Escola Nacional de Saúde Pública – Fiocruz

Departamento de Epidemiologia e Métodos Quantitativos em Saúde

Aprendendo R

Antonio Guilherme Fonseca Pacheco
Geraldo Marcelo da Cunha
Valeska Lima Andreozzi

O objetivo desse material é introduzir o ambiente R para alunos de pós-graduação em Saúde Pública e mostrar suas vantagens e desvantagens. Estamos considerando que os alunos que estão fazendo uso deste material já tiveram algum contato mesmo que básico com o computador e que já tenham algum conhecimento de noções básicas de operação do Windows.

Gostaríamos de iniciar a apresentação do R a partir de algumas perguntas que são comuns (e que na maioria das vezes foram feitas por nós mesmos antes de termos nos tornados "amantes do R")
;-)

O que é o R?

O R é um sistema desenvolvido a partir da linguagem S (que também é usada numa versão comercial – o S-Plus), que tem suas origens nos laboratórios da AT&T no final dos anos 80. Em 1995 dois professores de estatística da Universidade de Auckland, na Nova Zelândia, iniciaram o "Projeto R", com o intuito de desenvolver um programa estatístico poderoso baseado em S, e de domínio público.

Com o R posso utilizar menus para fazer análises estatísticas, como no SPSS, SAS e S-Plus?

Não. O R em versão para Windows é até provido de menus, mas todos são usados para realizar tarefas não estatísticas (como atualizar a versão ou salvar um gráfico). Todas as funções estatísticas que acompanham o R devem ser chamadas a partir do cursor do programa (seja digitando um comando ou copiando e colando um comando previamente digitado).

O fato do R não possuir menus não seria uma desvantagem em relação a outros pacotes estatísticos?

Depende. Muitos irão certamente interpretar esse fato como uma desvantagem, mas a gente entende que na verdade esta é uma vantagem forte do R. A utilização do R para realizar análises estatísticas exige muito mais do que simplesmente apertar alguns botões em série e dar alguns cliques no mouse: para trabalhar dados com o R é preciso PENSAR e ENTENDER o que se está fazendo. Ao contrário de muitos pacotes estatísticos clássicos, o R permite uma grande flexibilidade em relação às funções estatísticas pré-existentes, i.e. as funções são "editáveis", além da possibilidade de você mesmo poder criar as suas próprias funções personalizadas (como será mostrado mais tarde).

Quanto custa para ter uma cópia oficial do R?

Não custa nada: ele é de graça MESMO, ou seja, ninguém precisa gastar US$ 1.349, o que seria necessário para comprar o módulo básico do SPSS, por exemplo; nem ser obrigado a cometer um pequeno delito para usar o R.

Se ninguém está ganhando dinheiro para manter o R atualizado, como posso ter certeza que se trata de um produto confiável?

Esta é uma outra vantagem do R: o Projeto R é de uma colaboração internacional de vários pesquisadores que se comunicam através de uma eficiente lista de discussão pela Internet. Com isso, não só "bugs" (defeitos de programação) são detectados e corrigidos, como também novos módulos contendo métodos estatísticos recentemente implementados são regularmente disponibilizados e atualizados na rede.

O que são esses módulos adicionais?

Os módulos adicionais funcionam da seguinte forma: um pesquisador em algum lugar do mundo precisou desenvolver uma aplicação numa área que não é coberta nem pelo módulo básico nem pelos módulos de colaboradores existentes. O que esse pesquisador faz é desenvolver o que é chamada de uma biblioteca para o R com as funções que ele criou e utilizou, disponibilizando-a na rede. A vantagem é que a biblioteca pode ser usada por diferentes pessoas, que irão eventualmente reportar erros nas funções, que podem então ser atualizadas pelo seu criador.

Que plataformas (sistemas operacionais) suportam o R?

Atualmente o R está disponível para a família UNIX (incluindo LINUX), a maior parte dos Mac OS e ainda Windows 95, 98, NT, 2000, Me, XP.

Onde posso conseguir o R?

O R está disponível na internet no *website* do CRAN – que é o *Comprehensive R Archive Network* ou "Rede Completa de Arquivos do R", no seguinte endereço: http://www.r-project.org/

Já sei, não gostou da tradução do *website,* né? Bem, se alguém tiver uma tradução melhor para *comprehensive,* por favor me avise...
;-)

Muito bem. Agora que já ganhamos uma certa noção do que vem a ser o R, vamos ver como esse material está dividido.

A idéia desse documento é separar em módulos diferentes assuntos estatísticos que são tratados pelo R; desse modo, pessoas que tenham interesses em tipos de análises diferentes poderão consultar partes específicas do material, sem ter o trabalho de paginá-lo exaustivamente para encontrar o que se quer.

O primeiro módulo, "Baixando e Instalando o R", é bastante curto e trata da página da internet que abriga todo o material necessário para baixar e instalar o R – chamada CRAN. O segundo módulo, chamado "Básico" serve para dar uma noção geral do funcionamento do R, desde uma simples calculadora até uma poderosa ferramenta de programação em estatísticas, sempre usando exemplos intuitivos e em um nível bastante inicial. Esses dois módulos não exigem o uso de qualquer dado externo ou pacotes que já não estejam incluídos na versão mais básica do R – aquela que você irá instalar inicialmente na sua máquina.

O terceiro módulo trata da entrada e saída de dados no R. Nesse caso não só será mostrado como o R lê dados externos, mas também como se exportam dados, saídas em texto e saídas gráficas. O uso de um pacote próprio para esse fim também é mostrada e dados externos serão necessários para a parte de importação de dados.

Esses três módulos são os únicos que podem ser encarados como seqüenciais e também fundamentais para uma compreensão inicial do ambiente. A partir daí, uma série de módulos já estão e serão mais tarde desenvolvidos para aulas ou assuntos específicos e ficarão guardados em arquivos separados, permitindo um acesso rápido e interativo aos assuntos de interesse.

Portanto, divirta-se...

Módulo Baixando e Instalando o R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimentos básicos de informática e acesso à internet (preferencialmente.)

Pacotes e arquivos necessários: Nenhum.

Esse módulo inicial tem o objetivo de capacitar pessoas a navegar mais facilmente no *webwsite* do R, chamado CRAN (*Comprehensive R Archive Network*) e ainda orientar a instalação do R para Windows e fazer uma configuração inicial básica do programa, incluindo a instalação de pacotes adicionais. Como esse material poderá estar acompanhado de um CD-ROM já com o programa de instalação do R e os seus pacotes, abordaremos também esse tópico. No CRAN é possível baixar não só o pacote principal do R, mas também os pacotes opcionais (chamados de contribuídos) e também uma série de manuais. Vale lembrar que não só a versão para Windows está disponível no CRAN, mas também versões para a família UNIX (incluindo Linux) e ainda para Mac.

Dispensável dizer que a correta utilização desse módulo pressupõe que a máquina do usuário esteja conectada à internet.

Conhecendo o CRAN

O primeiro macete é encontrar o *website* do CRAN, que não é muito fácil de achar se você não souber o endereço (que por sinal não é lá muito intuitivo). A dica é ir no *website* de uma dessas ferramentas de busca na internet (como por exemplo o Google) e procurar por "CRAN". Certamente uma das primeiras respostas será o endereço do CRAN. É claro que não somos tão maus assim para fazermos você voltar à primeira parte desse documento para procurar o endereço sem usar o Google... Vá em:

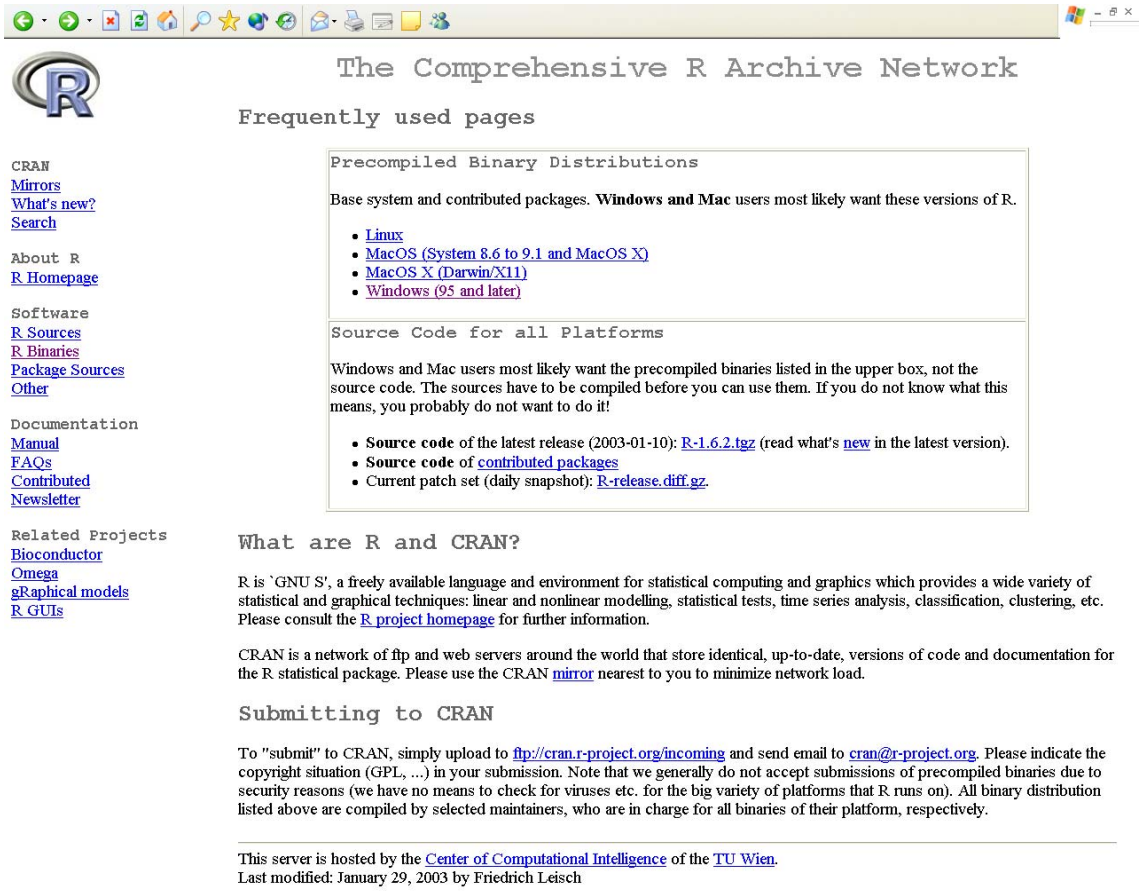
http://cran.r-project.org/

Uma vez lá, você verá algo como a figura abaixo.

Nessa página você vai encontrar praticamente tudo que você precisa saber sobre o R e tudo que você precisará baixar para a sua máquina para instalar o R e seus pacotes adicionais, além de informação sobre a excelente lista de discussão que é mantida na internet e da qual participam as pessoas do núcleo de desenvolvimento do R.

Na parte central da página você vai observar uma lista de páginas mais usadas e uma pequena explicação do R e do CRAN abaixo. Como essa parte da página pode mudar, vamos utilizar os atalhos que estão listados à esquerda, que acabarão levando para as mesmas páginas listadas como mais freqüentes.

À esquerda da página, então, você vai encontrar vários *links* para os diversos recursos disponíveis no CRAN... Vamos ver aonde alguns deles nos levarão e depois de aberto o caminho, sugerimos que você mesmo, imbuído de seu espírito aventureiro de surfista da rede, explore os demais atalhos. Inicialmente, repare que eles estão divididos em 5 grupos. Nós iremos abordar apenas os atalhos dos grupos *Software* e *Documentation*.



Apesar de você provavelmente estar fazendo esse curso em um laboratório que já tem instalado o R em uma plataforma Windows, é sempre bom instalar uma cópia do programa em uma outra máquina que você tenha acesso (seja em casa ou no trabalho). O atalho para baixar o programa que instala o R para Windows na sua máquina é o *R Binaries* que fica no grupo *Software*. Uma vez clicando nesse atalho, uma página se abrirá com o título de *Index of /bin*. Observe que os atalhos à sua esquerda permanecem lá. Nessa página (que mais parece uma estrutura de diretórios – o que de fato é), a escolha óbvia se você vai instalar o R para Windows é clicar em *windows/*.

Ao clicar no atalho, uma nova página será mostrada com 3 atalhos: *base, contrib* e *unsupported*. O primeiro atalho (que é o que estamos procurando) nos levará a uma nova página onde poderemos baixar o executável para a instalação. Os outros atalhos nós iremos deixar de lado, pois são para acessar pacotes contribuídos para o R, um assunto que vamos abordar em outros módulos. Apesar de ser possível baixar cada um dos pacotes e instalá-los manualmente no R, nós veremos que uma vez instalado o R, existe uma maneira muito mais simples de instalar pacotes do CRAN usando o próprio R como interface.

Muito bem, clicando então em *base* uma nova página se abrirá e você deve procurar pelo atalho *rwXXXX.exe*, onde os *XXXX* representam a versão atualmente disponível. Por exemplo, quando esse material estava sendo escrito, a versão disponível era a 1.6.2 e o atalho era *rw1062.exe*. Basta agora clicar nesse atalho e o seu *browser* irá automaticamente perguntar onde você deseja salvar o arquivo. Salve onde você estiver mais acostumado (e.g no *desktop* ou em *Meus Documentos*) e aguarde. Atenção: esse arquivo tem em torno de 20 Mb de tamanho e dependendo da sua velocidade de conexão com a internet, o *download* pode demorar até mais de uma hora.

Uma vez baixado o programa de instalação, basta dar um duplo-clique em seu ícone e seguir as instruções de instalação (veja as instruções mais detalhadas abaixo.)

Bem, vamos passar agora para a documentação do R. Observe os grupos de menus à esquerda e procure o grupo *Documentation*. Primeiramente, o atalho *Manual* irá levá-lo para uma

página de onde é possível baixar vários materiais em formato PDF (que pode ser lido pelo *Adobe Acrobat Reader*, um programa que pode ser baixado gratuitamente da rede – na verdade se a sua máquina já tem o *Reader* instalado, o documento aparecerá automaticamente na tela do seu *browser*. Não vamos nos estender aqui explicando cada um dos materiais. Experimente baixar alguns e divirta-se...

O atalho seguinte é chamado *FAQs*, que você já deve ter ouvido falar em algum lugar, mas que se não sabe o que significa, aqui vai: são "perguntas feitas com freqüência", sobre algum assunto que naturalmente aqui é sobre o R (*FAQ* em inglês é abreviação de *Frequently Asked Questions*). Ao clicar nesse atalho, vai se abrir uma página com *links* para a *FAQ* do R (que é uma *FAQ* "geral") e também *FAQs* específicas para usuários de Windows e Mac. Clique e boa leitura...

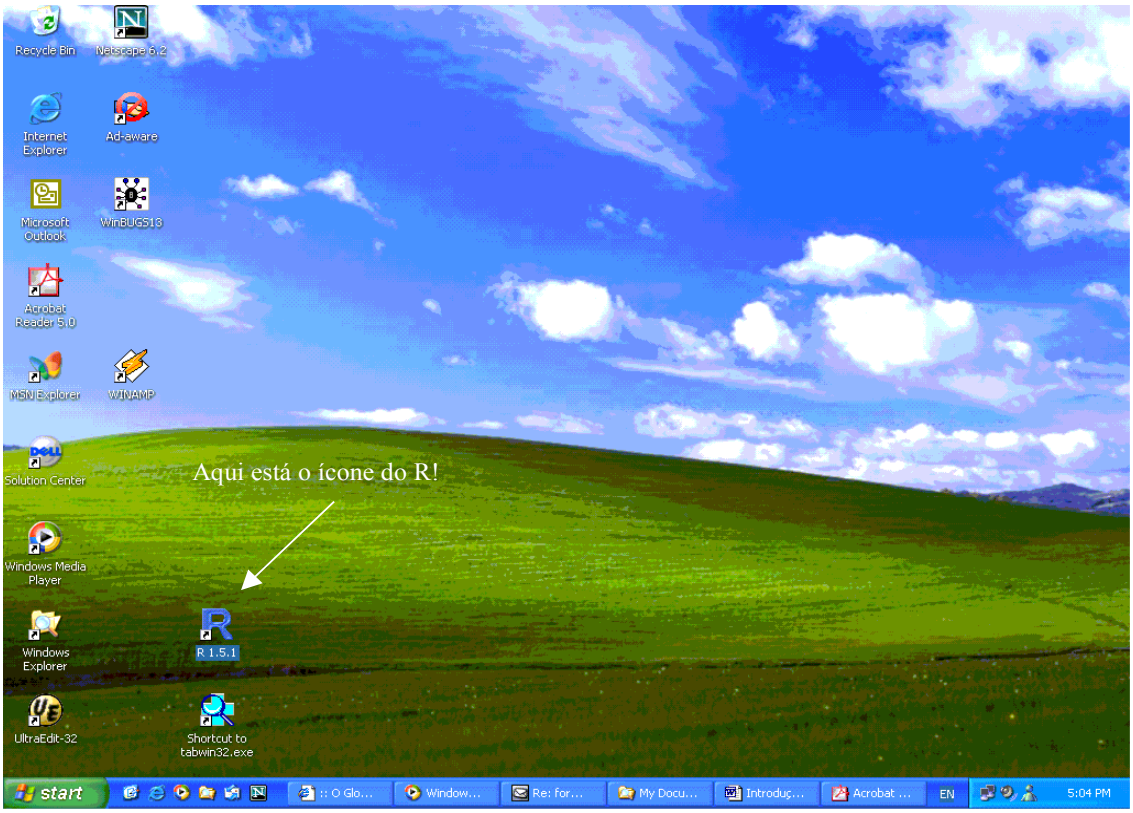
O terceiro atalho desse grupo é chamado *Contributed* e contém diversos manuais escritos por pessoas que não fazem parte do núcleo de desenvolvimento do R, mas que são reconhecidamente boas contribuições para o entendimento do programa. Você encontrará inclusive materiais em outras línguas que não o inglês. O leitor é convidado a explorar esses diversos materiais para ver se algum é do seu interesse.

Finalmente o último atalho nesse grupo é o *Newsletter*, onde você encontrará uma coleção de documentos sobre o desenvolvimento do R além de comentários sobre pacotes relevantes adicionados ao programa, dentre outras informações.

Por fim, não deixe de se aventurar pelos demais atalhos do CRAN que é uma fonte muito grande para informação sobre o R. Quanto à ajuda do programa, num outro módulo vamos falar sobre a documentação em html que acompanha o módulo básico do R. Até lá, boa sorte...

Instalando e Configurando o R para Windows

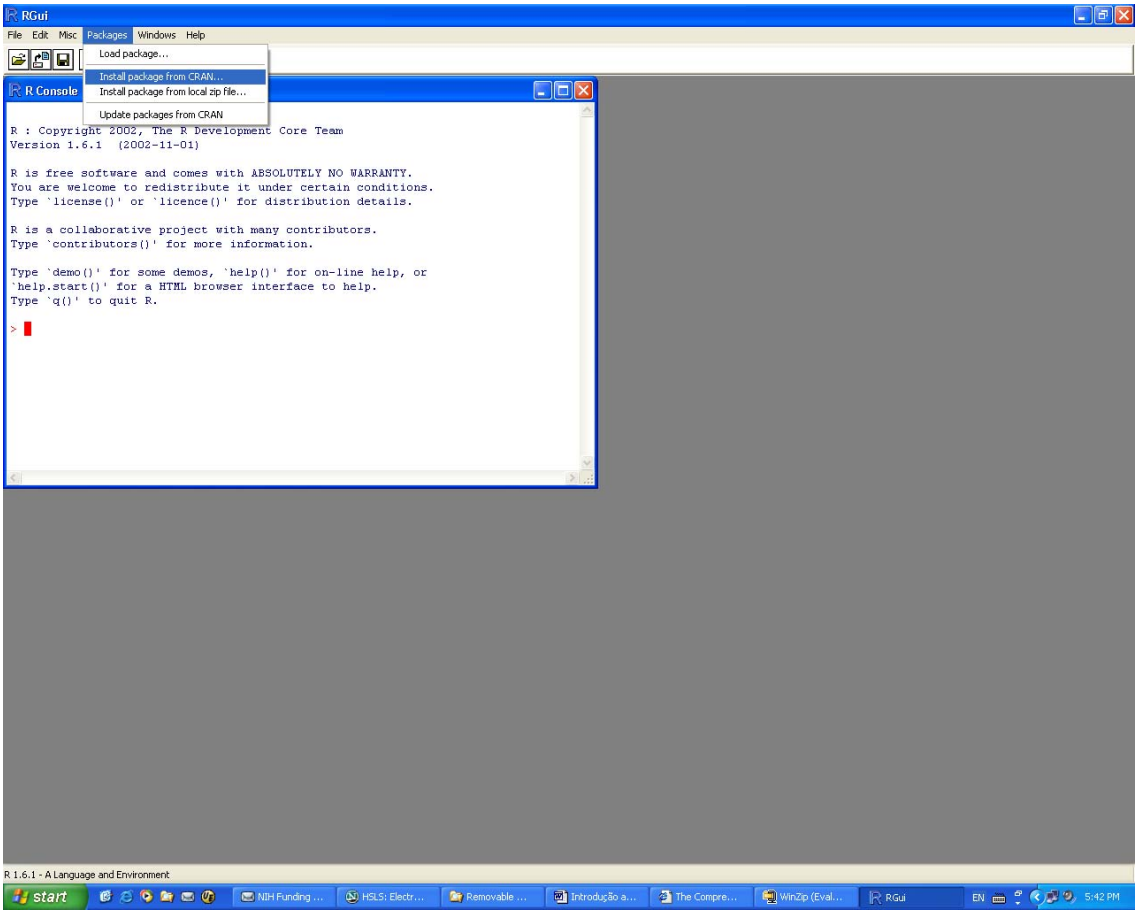
Como mencionado anteriormente, a instalação do R mesmo não tem grandes mistérios: basta dar um duplo-clique no ícone do programa de instalação e seguir as instruções que aparecem



na tela. Ao fazer isso, o programa será instalado no diretório padrão (geralmente "c:\Program Files\R\rwXXXX\") e será criada uma pasta dentro do seu menu Iniciar – Programas chamada "R". Adicionalmente, um ícone será colocado na sua área de trabalho.

Para iniciar o programa, dê um duplo-clique no ícone do R localizado na área de trabalho... Olhe um exemplo aí em cima em um computador qualquer...

Ao fazer isso, o programa será executado e uma janela com o programa será aberta. Você deve notar que o programa se parece bastante com outros programas que você já deve estar habituado a usar: ele tem uma barra de menus, uma de ícones e uma janela aberta, com o *prompt* de comando. Por enquanto, nós estaremos interessados no menu *Packages*, onde iremos ver como instalar pacotes adicionais diretamente do CRAN, usando o próprio R. Observe a figura:



Ao clicar na opção *Installing package from CRAN*, uma janela se abrirá com uma lista de todos os pacotes disponíveis no CRAN para *download*. Você pode então escolher um ou vários pacotes para serem baixados e instalados automaticamente na sua máquina, incluindo a ajuda em HTML para esses pacotes. As técnicas de seleção múltipla que você deve conhecer para programas como o Windows Explorer funcionam nessa janela, ou seja, para selecionar um intervalo de pacotes, clique no primeiro, pressione a tecla *Shift* e então clique no último pacote do intervalo; para selecionar pacotes salteados, clique nos pacotes, com a tecla *Control (Ctrl)* pressionada. Claro que uma opção pode ser selecionar todos os pacotes (selecionando do primeiro ao último com a ajuda da tecla *Shift*), mas atenção: ao selecionar todos os pacotes, você estará optando por baixar uma grande quantidade de dados, o que, num computador que está ligado à internet via modem, pode demorar muitas horas.

Ao final da instalação, o R vai perguntar se você deseja apagar o(s) pacote (s) baixados (a pergunta é mais ou menos: *Delete downloaded files (y/N)?* Para a qual você deve, geralmente responder “y”, a não ser que você deseje ter o pacote compactado para instalar em uma outra

máquina. Confundiu? Eu explico: os pacotes são na verdade arquivos compactados no formato ".zip", que pode ser lido com o popularíssimo Winzip. O que o programa faz é baixar esse arquivo do CRAN para um diretório temporário, e em seguida instalar o pacote a partir desse arquivo ".zip". O que ele pergunta no final é se você deseja manter esse arquivo gravado no seu computador ou não (claro que isso DEPOIS do pacote já ter sido instalado.)

Bem, uma vez instalados os pacotes, uma outra opção nesse mesmo menu pode ser usada para atualizar versões de pacotes instalados na sua máquina, mas que já possuem versões mais atualizadas no CRAN. É a última opção do menu *Update packages from CRAN* que ao ser acionada irá procurar os pacotes instalados na sua máquina, compará-los com as versões desses pacotes no CRAN e perguntar se você deseja atualizar (um a um). Note que ele não vai perguntar se você quer atualizar um pacote que ainda não está instalado na sua máquina!!! Para isso, você deve instalar o pacote, como descrito anteriormente.

Você agora deve estar se perguntando: mas e se eu não tiver acesso à internet e tiver tanto o programa de instalação do R quanto os pacotes em uma mídia de distribuição qualquer – um CD-ROM por exemplo?

Nesse caso, não há problema algum. Para instalar o R basta, a partir do próprio CD-ROM executar o programa de instalação do R (que deve ter um instrutivozinho junto com o CD, naturalmente, mas que deve consistir apenas de explicar em que diretório dentro do CD-ROM esse arquivo s encontra e dar um duplo-clique no seu ícone.

Já para instalar os pacotes (depois naturalmente de instalado o R) o procedimento é muito semelhante. A terceira opção do mesmo menu que estávamos falando *Install package from local zip files* irá abrir uma janela para você escolher pacotes compactados nas suas unidades locais (que pode ser o seu disco rígido ou uma outra mídia qualquer) para serem instalados na sua máquina. Esse procedimento também depende de onde estão guardados os arquivos. Nesse caso, se por exemplo esses arquivos estiverem em um CD-ROM distribuído para você, este deve vir acompanhando como instrutivo indicando a partir de que diretório os pacotes devem ser instalados, mas basicamente, tudo o que deve ser feito é escolher os pacotes e deixar o R fazer o resto.

Então, boa sorte...

Módulo Básico

Autores: Antonio Guilherme Fonseca Pacheco; Geraldo Marcelo da Cunha; Valeska Lima Andreozzi

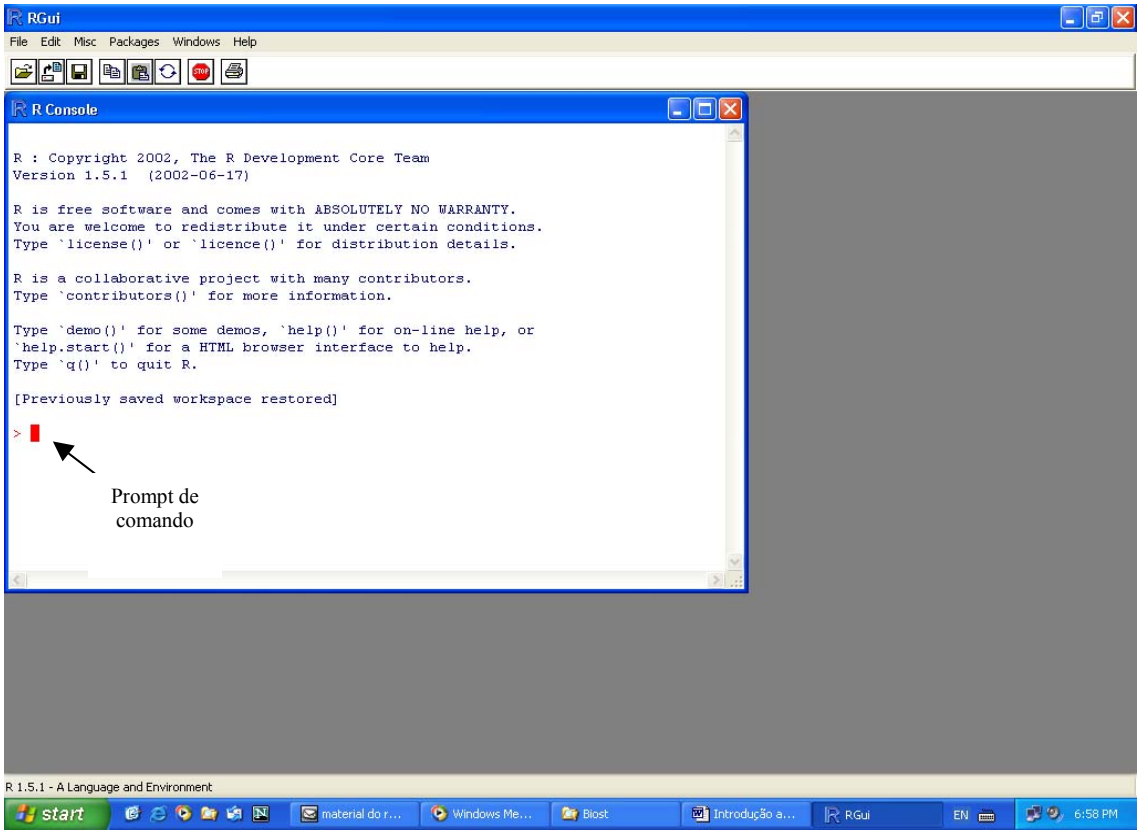
Pré-requisitos: Por se tratar de um módulo básico, não existem pré-requisitos para iniciar o treinamento em R com este documento, se o leitor tiver acesso ao R já instalado em sua máquina. Caso contrário, a leitura do Módulo "Baixando e Instalando o R" deve ser feita anteriormente (e claro, proceder a dita instalação). Foi feito um esforço também para tornar este material independente de qualquer arquivo ou banco que já não exista na distribuição básica do R, e por isso alguns exemplos ficaram um pouco "prejudicados"...
;-)

Pacotes e arquivos necessários: Nenhum.

Material extra: Um disquete (ou qualquer mídia regravável) para salvar o seu trabalho

Este módulo tem por objetivo apresentar as características básicas de funcionamento do R e introduzir o leitor a um primeiro contato com o ambiente R, sem referência específica a um tópico de estatística. Por ora, assumimos que o R já está instalado no seu computador, e que um ícone já aparece na área de trabalho (caso isso não seja verdade, leia o módulo "Baixando e Instalando o R" para instruções sobre como baixar, instalar e configurar o R em sua máquina.)

Para iniciar uma sessão do R, dê um duplo-clique no ícone do R que deve estar localizado na sua área de trabalho. Uma vez iniciado o programa, uma janela contendo o *prompt* de comando é aberta:



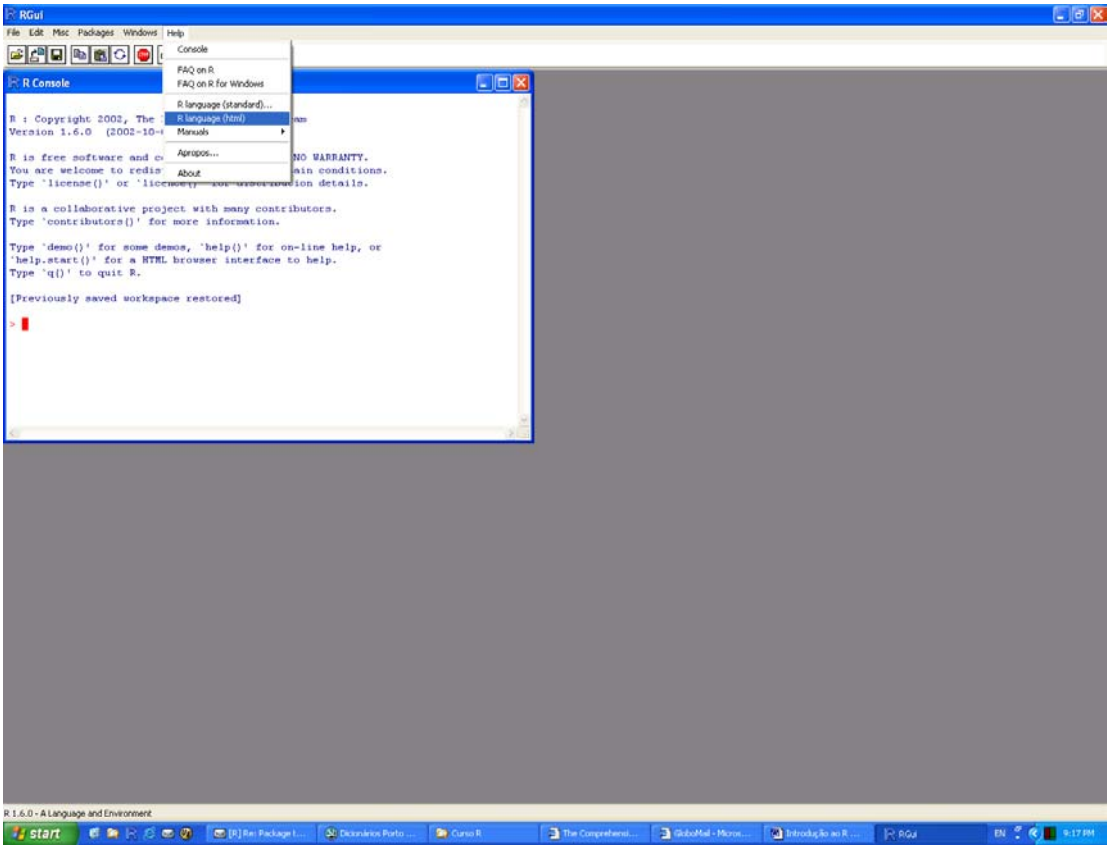
Observe que há uma janela dentro do programa com o *prompt* de comando. As saídas não gráficas no R aparecerão nesta mesma janela enquanto que as gráficas serão geradas em uma janela separada.

Entendendo a Ajuda do R

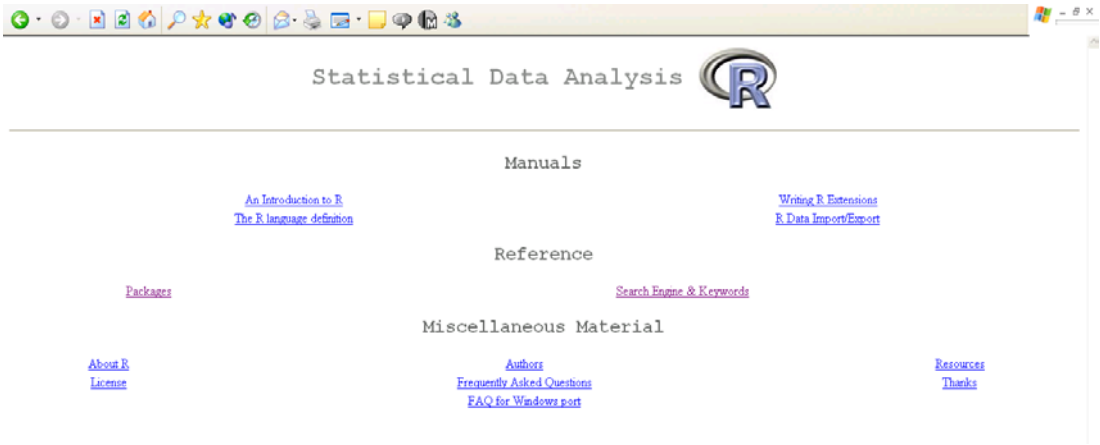
Se você me perguntasse qual é o conhecimento fundamental que uma pessoa que deseja saber usar qualquer programa de computador deve ter, a resposta seria contundente: o que diferencia a capacidade de um usuário se tornar familiarizado com qualquer *software* é a capacidade de usar de maneira racional e eficiente a **ajuda** desse programa. O R possui diversas formas diferentes de obter ajuda sobre o programa em geral e ainda sobre funções e conjuntos de dados disponíveis.

Uma característica muito útil do R é sem dúvida nenhuma a ajuda baseada em HTML que ele possui. Para quem não entendeu o que isso quer dizer, o R tem uma ajuda que pode ser acessada através do seu *browser* favorito... Ah, para quem não sabe, *browser* é aquele programa que você usa para navegar na internet, como o Internet Explorer ou o Netscape, para citar os mais usados...

Para acessar essa ajuda, clique na barra de menus em *Help* e, em seguida, *R language (html)*, como na figura abaixo:



Ao fazer isso, o seu *browser* será automaticamente executado e a página inicial da ajuda vai aparecer na sua tela. Você verá algo assim:

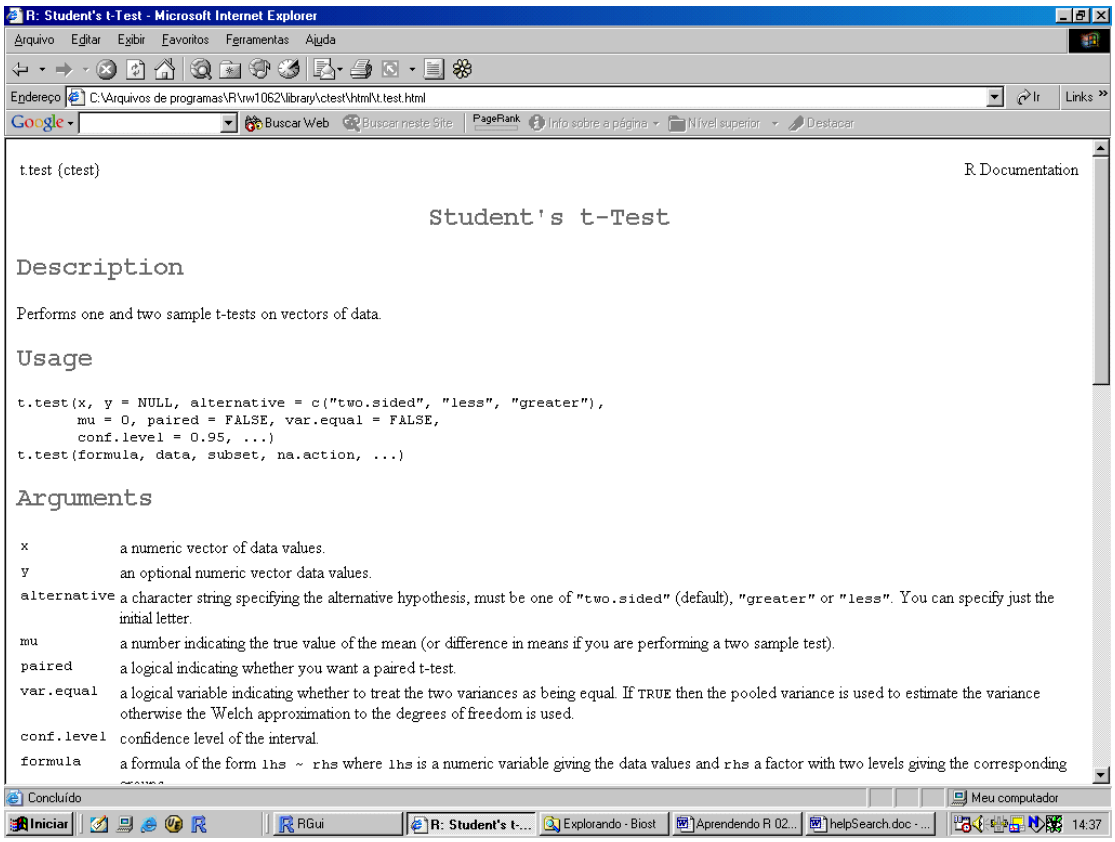


O grupo de *links* que dizem respeito à ajuda do R é o de referências (*Reference*). O *link* de *packages* vai levar você à pagina com todos os pacotes que estão instalados na sua máquina, além é claro do pacote *base* (que é o pacote básico do R que é automaticamente instalado quando você instala o R). Ao clicar em cada um dos *links* com os nomes dos pacotes, você será levado então a uma página com todas as funções e bancos de dados que compõem o pacote. Por que você não tenta acessar as funções do pacote *base*?

O segundo *link* em interesse para nós aqui é o de *Search Engine & Keywords*, algo como "Ferramenta de Procura e Palavras-chave".

Ao clicar neste segundo *link* você encontrará a seguinte tela que está dividida em duas partes: *Search* e *Keywords*.

Na primeira parte podemos procurar uma ou mais palavras que estejam no título das páginas de ajuda (*Help page titles*), por palavras-chave (*keywords*) ou pelo nome de funções ou objetos (*Object names*). Vamos experimentar procurar ajuda sobre como fazer um teste t de *Student*. Digite, em inglês claro, *t-test* e clique no botão *Search* ou aperte somente a tecla *enter* no seu teclado. O *help* HTML abrirá outra página com uma lista de comandos e uma pequena descrição do que cada uma delas faz. Clique no nome do comando e outra janela será aberta com todos os detalhes do respectivo comando, como mostra a figura abaixo:



No canto superior esquerdo encontramos o nome do comando e entre chaves o nome do pacote não qual este comando se encontra (mas a frente vamos entender um pouco melhor sobre o tópico pacotes). Esta janela é composta por 7 ítens:

1. *Description* (Descrição)
   Descreve o que o comando faz
2. *Usage* (Sintaxe – forma de uso)
   Mostra como devemos usar o comando, quais os seus argumentos e respectivos valores padrões
3. *Arguments*
   Descreve cada argumento listado na sintaxe do comando.
4. *Details* (Detalhes)
   Descreve com um pouco mais de detalhe o comando e/ou argumentos.
5. *Value* (Valores)
   Descreve o que o comando retorna
6. *See Also* (Ver também)
   Cita alguns comandos que tenham alguma ligação com o comando em questão
7. *Examples* (Exemplos)
   Exemplifica o comando. Podemos selecionar os comando listados no exemplo, depois colar e copiar na janela do R para executa-los.

O critério de procura do *help* HTML pode gerar dúvidas algumas vezes. Nas situações em que você não encontrar o que esteja procurando o conselho é: tente pesquisar de uma forma diferente. Só para exemplificar tente procurar a palavra *t test* (sem hífem) e *test t*. Você vai observar que o *help* irá retornar listas de comandos que são diferentes de quando usamos a palavra *t-test*.

Ah, mas como volto para a tela inicial (ou qualquer tela anterior)? Para voltar para a página inicial ou qualquer página anterior, use o botão “voltar para” (usualmente representado por uma setinha) do seu *web browser*.

A segunda parte da tela *Keyword* lista diversos *links* por assunto. Ao clicar em um desses *links* uma janela será aberta com todos os comandos relacionados aquele assunto.

No momento acho que este tópico pode não ser muito útil, mas com certeza você irá voltar nele mas a frente assim que começar a usar o R. Na verdade a gente aprende que a melhor maneira de trabalhar no R e estar sempre com o *help* HTML aberto, pois decorar todos os comando e suas sintaxes e algo impossível.

A partir deste ponto, vamos começar a discutir coisas que são feitas no R propriamente dito. Isto consiste basicamente em entrar comandos no *prompt* de comando, que já foi apresentado anteriormente. O leitor tem duas opções para conferir as saídas aqui apresentadas no R: ou escreve a partir do teclado o mesmo comando apresentado, ou então copia e cola o comando no R. Para copiar, basta marcar o texto com o *mouse* e então pressionar a tecla *Ctrl* seguida da tecla *c* no seu teclado. Para colar, pressione a tecla *Ctrl* seguida da tecla *v* no seu teclado. Para isso, evidentemente você deve ter o programa onde você está lendo este material aberto e também o R, alternando entre um e outro. Um detalhe importante deve ser mencionado de uma vez: o *prompt* do R, como você já deve ter notado, começa com o símbolo de maior (“>”). Sempre que neste material o comando for mostrado juntamente com a saída correspondente, o sinal “>” estará também no texto, como nesse primeiro exemplo abaixo:

```
> 3+7
[1] 10
```

Se você quiser copiar e colar esse comando de soma de dois números no R, NÃO COPIE O SINAL DE MAIOR, pois se o fizer, você receberá uma mensagem de erro. Veja só:

```
> > 3+7
Error: syntax error
```

E, claro, não copie a saída também, só o comando. Ah, e às vezes depois de colar, você deverá pressionar a tecla *enter* no teclado para o R executar o comando. É claro que você é livre para copiar o que quiser, mas sempre que a recomendação for copiar e colar, os comandos virão sem o símbolo “>” antes do comando.

Muito bem, então dê um duplo-clique no ícone do R para abri-lo e boa sorte...

Cálculos e Funções

Vamos iniciar mostrando o R como calculadora. Por exemplo, tente fazer uma conta simples:

```
> 3+7
[1] 10
```

Repare que o resultado da operação é precedido por um `[1]`. Isto é apenas para indicar que o resultado “10” é o primeiro (e nesse caso único) elemento retornado pelo programa. Bem, essa é uma excelente deixa para dizer que o R pode funcionar como uma calculadora, como a que você já deve ter usado em cursos passados. Porque a gente não experimenta outras operações simples? Tente:

```
> 3-7
[1] -4

> 3*7
[1] 21
```

```
> 3/7
[1] 0.4285714

> 3^7
[1] 2187
```

Como em outros programas (e.g. Excel), os operadores matemáticos simples são "+", "-", "*", "/" e "^" – este último para potenciação. Você pode estar se perguntando agora, depois de algumas aulas de matemática: como é o inverso do operador potência? Ou melhor, como descobrir a que potência um certo número foi elevado para obtermos um determinado resultado?

Bom, nesse caso temos que lançar mão de uma **função**, já que como você deve se lembrar o operador inverso da potenciação é a função logarítmica. Recordando:

Se $3^7 = 2187 \Rightarrow \log_3(2187) = 7$, ou seja, o logaritmo de 2187 na base 3 é igual a 7.

Lembrou? Será complicado obter esse resultado no R, sem a ajuda de um menuzinho? Bem, vamos tentar escrever exatamente como estamos interpretando a função?

```
> log(2187, base=3)
[1] 7
```

E não é que funcionou??? Quer dizer, precisamos elevar 3 à sétima potência para obter o resultado de 2187.

Vamos aproveitar esse exemplo para explicar algumas características básicas das funções disponíveis no R. Todas têm a forma:

```
> função (argumento(s) obrigatório(s), argumento(s) opcional(is))
```

Sendo que os argumentos opcionais podem ter um valor padrão pré-estabelecido ou não. Os argumentos estarão sempre entre parênteses sendo separados por vírgula. Se você deixar o primeiro argumento em branco, vai receber uma mensagem de erro:

```
> log(, base=3)
Error in log(, base = 3) : Argument "x" is missing, with no default
```

Repare que o programa já indica que o argumento (que ele chama de "x") obrigatório não foi fornecido e que não existe nenhum padrão (*default*). Agora, deixe o segundo argumento em branco:

```
> log(2187,)
[1] 7.690286
```

Dessa vez não houve mensagem de erro, mas o resultado é diferente do que a gente obteve anteriormente... O que terá acontecido? Bem, se não houve queixa do programa quanto ao segundo argumento, ele deve ter um *default* pré-estabelecido... Por que esse *default* seria 3?

Vamos descobrir, aproveitando para aprender a consultar a ajuda do R? Por que não tentamos assim:

```
> ?log
```

Note que uma nova janela se abriu dentro do próprio R com a informação de ajuda sobre a função "log". A essa altura você está se perguntando onde estará a ajuda em HTML, não? É que o R possui vários tipos de ajuda; se preferir, abra a ajuda em HTML e procure pela função "log". De qualquer modo, se você olhar a entrada "Usage", na página de ajuda sobre o "log", vai ver:

```
log(x, base = exp(1))
```

Isso quer dizer que a função `log()` precisa de um argumento `x` que não tem *default* e também de um argumento `base` que tem um valor pré-estabelecido, `exp(1)` que nada mais é que o número `e` (conhecido como algarismo Neperiano e aproximadamente igual ao valor 2.718282). Vamos tentar? Volte para a janela do *prompt* e veja o valor:

```
> exp(1)
[1] 2.718282
```

Lembrou dele? Vamos conferir então se foi isso mesmo que aconteceu. Vamos tentar assim:

```
> log(2187, base=2.718282)
[1] 7.690286
```

É o mesmo resultado!!! Parece ser isso mesmo!!!

A próxima pergunta que você pode estar fazendo é se existe uma maneira de economizar digitação omitindo o argumento `base =`. A resposta é sim, neste caso. Vamos tentar:

```
> log(2187, 3)
[1] 7
```

Neste caso deu certo porque a função só tem 2 argumentos possíveis e eles entraram na ordem certa. Se você trocar a ordem, o resultado é diferente:

```
> log(3,2187)
[1] 0.1428571
```

Se você trocar a ordem, porém especificando quem é quem, não haverá confusão:

```
> log(base=3,2187)
[1] 7
```

Moral da história: é sempre melhor especificarmos o que estamos escrevendo (embora muitas vezes nós tenhamos preguiça de fazer isso também...).

;-)

Vamos aproveitar e ver rapidamente algumas funções bastante usadas no R. Veja a tabela abaixo:

Função	Descrição
`sqrt()`	raiz quadrada
`abs()`	valor absoluto
`exp()`	exponencial
`log10()`	logaritmo na base 10
`log()`	Logaritmo na base e
`sin() cos() tan()`	funções trigonométricas
`asin() acos() atan()`	funções trigonométricas inversas
`sin() cos() tan()`	funções trigonométricas

Curioso para saber como essas funções funcionam? Tente usar a ajuda do R para descobrir. Por exemplo:

```
> help(sqrt)
```

Vai se abrir uma janela de ajuda que na verdade é genérica para a função "`abs()`" também. Esperamos que você tenha percebido que esta é uma forma alternativa de chamar a ajuda (em vez de `?sqrt`)

Como o R armazena Objetos e Comandos

Muito bem, até agora ganhamos uma noção de como o R realiza operações matemáticas e de como as funções pré-definidas (e também as que um dia você mesmo vai criar!!!) funcionam.

Mas você deve estar se perguntando: e os dados? Eu preciso mesmo é analisar dados com este programa... Como é que funciona? Como é que o R armazena dados, sejam eles digitados a partir do teclado, sejam eles importados de um arquivo externo?

Bom, antes de entrar nessa parte mais interessante, vamos começar a falar sobre a maneira diferente como o R guarda um **objeto.**

Objeto para o R significa tanto um banco de dados quanto a saída de uma função ou até mesmo uma fórmula. Para criar um objeto no R, seja ele qual for, você deverá sempre usar o operador de *asignment* (`<-`), apontando para o nome de um objeto. Por exemplo, para criar o objeto `x` com o valor 3, faça assim no R:

```
> x <- 3
```

O que fizemos foi colocar o número 3 em um objeto chamado `x` usando o operador *assignment* (`<-`). Este objeto pode então ser chamado e seu conteúdo revelado, simplesmente digitando o seu nome:

```
> x
[1] 3
```

Não se preocupe: ao longo deste documento vamos criar diversos objetos e você entenderá melhor como fazê-lo. Por ora, o importante é entender que tudo o que você cria no R é um objeto.

Ao contrário de outros programas que você já deve conhecer (como Excel, Word, etc), o R não armazena cada um dos seus objetos como um arquivo que fica em uma determinada localização no seu disco rígido. Ficou difícil de entender? Vamos tentar explicar de novo.

Quando você usa um editor de texto como o Word, por exemplo, e você salva o seu trabalho pela primeira vez, o programa pede para você dar um nome ao seu arquivo e o salva fisicamente em um diretório (que em geral é em "Meus Documentos" – "My Documents" se versão em inglês). Uma vez salvo, aquele único arquivo estará naquela localidade com o nome que você deu e portanto você poderá abri-lo no Word para futura edição ou para imprimir, etc.

O R trabalha um pouco diferente. Na verdade, quando você cria uma série de **objetos**, eles podem ser salvos também, porém EM GRUPO, ou seja, o R salva TODOS OS OBJETOS num mesmo arquivo. Mais tarde, você poderá abrir este arquivo e trabalhar com TODOS os objetos que foram salvos.

Você deve estar se perguntando se isso não vai atrapalhar muito a sua vida... Depende. Apesar do R salvar o arquivão com o mesmo nome sempre (".RData" – o nome é esquisitão assim mesmo), você pode salvá-lo com um nome diferente (mas mantendo a extensão) em diretórios (ou pastas, o nome moderno...;-) diferentes. A vantagem é que você pode armazenar todos os objetos relativos a um determinado projeto num diretório próprio do projeto.

Uma questão que deve ficar clara é que o R vai armazenar temporariamente todos os objetos que foram criados em uma sessão de trabalho e posteriormente salvá-los definitivamente no arquivo ".RData"; mas repare que serão salvos os OBJETOS, ou seja, tudo o que você guardou usando o símbolo de *assignment* (<-). Isso quer dizer que saídas de fórmulas, funções e dados externos serão armazenados como objetos (desde que designados para algum objeto com o símbolo <-), mas não as saídas na tela ou os gráficos gerados por funções. Mais tarde você vai aprender a salvar essas saídas de um modo mais conveniente.

Vamos ver um exemplo. Iremos salvar o nosso arquivão no *drive* de disquete (geralmente o *drive* A). É claro que para isso você deve ter um disquete...

Primeiro, vamos fazer com que o R mude o diretório de trabalho (que em geral é configurado para ser "C:\\Program Files\R\rwXXXX" Faça assim:

- Na barra de menus do R, clique em "File" e em seguida em "Change dir"
- Uma pequena janela chamada "Change directory" se abrirá. Clique em "Browse"
- A janela "Browse for folder" se abrirá. Dê um duplo clique em "Meu Computador", depois em "A:"
- Clique em "OK" e em "OK" novamente na outra janela.

Agora o nosso diretório de trabalho é o *drive* A, e agora podemos salvar a nossa sessão inteira no disquete:

- Vá em "File" de novo, mas clique em "Save Workspace"
- Note que o arquivo ".RData" está para ser salvo no nosso *drive* A
- Clique "Save"

Vamos agora verificar se o nosso arquivo está mesmo lá... Boa sorte!!!

Agora, experimente salvá-lo com outro nome, mas não se esquecendo de manter a extensão .Rdata (exemplo: "aluno.Rdata"). Quando você encerra uma sessão do R ele sempre pergunta se você deseja salvar as alterações, o que significa salvar todos os objetos novos que foram criados naquela sessão.

Uma outra característica do R é a capacidade de armazenar um histórico de comandos usados anteriormente. Como no caso dos objetos, o R armazena esses comandos num arquivo que tem a extensão ".Rhistory", também sem nome por *default* (e que também pode ser modificado, mantendo-se a extensão). Esse arquivo é na verdade um ASCII que contém todos os comandos que você salvou na sua última sessão. É claro que você pode editar esse arquivo, salvar com um nome diferente, etc.

O funcionamento do histórico é o seguinte: Na primeira vez que você salvar o seu *workspace*, o R vai automaticamente salvar no mesmo diretório um arquivo com o histórico; se ma próxima vez que você executar o R ele encontrar um arquivo ".Rhistory" na mesma pasta onde está o arquivo ".RData", ele vai carregar automaticamente esse arquivo. Caso contrário, você terá qeu carregá-lo manualmente, através do menu "File" e depois "Load History". Um detalhe importante é que caso você tenha carregado um arquivo de histórico (seja automaticamente ou manualmente), quando você salvar o histórico no final da sessão, o R vai acumular os comandos dessa sessão com os comandos previamente salvos. Tente brincar um pouco com o arquivo...

Por fim, uma vez que você tenha salvo o seu *workspace*, seja no disquete, seja em outro diretório qualquer, da próxima vez que for usá-lo, basta dar um duplo-clique no arquivo mesmo (a partir do Windows Explorer, por exemplo), que o R será automaticamente executado o *workspace* carregado.

Vetores

Certo. Não fique ansioso ainda... Antes de vermos a questão de ler dados externos, seria interessante ver como podemos entrar com algumas formas simples de dados a partir do teclado diretamente.
A forma mais simples de armazenamento de dados (i.e. o mais simples objeto) no R é um vetor. Um vetor é um seqüência em uma ordem específica de valores de um mesmo tipo de dados. Como os dados são mais freqüentemente números, os vetores numéricos são os mais usuais. Vetores não numéricos formados por caracteres (como nomes) são também bastante utilizados e podem também ser construídos no R.

Vamos criar um objeto chamado `x`, mas colocando diferentes coisas nesse objeto.

Um número:

```
> x <- 3
```

Você já deve ter duas perguntas prontinhas. A primeira é: que diabo é esse `<-` que apareceu entre o objeto que eu queria criar – o `x` – e o número 3? Já esqueceu, né?

;-)

Esse é o símbolo de atribuição (*assignment*) no R, lembra? Ele significa que o número 3 foi "colocado dentro" do objeto `x`. Toda vez que a gente quiser colocar alguma coisa em um objeto, esse símbolo vai ter que ser usado.

A segunda pergunta é: muito bem, colocamos o número 3 no objeto `x`, mas como é que eu verifico se o objeto `x` realmente possui o número 3? É simples, já fizemos isso também. Tente assim:

```
> x
[1] 3
```

Hummm... Quer dizer que para visualizar um objeto basta digitar o seu nome? É isso aí!!! Vamos aproveitar para mostrar um detalhe a mais do R. Para ele maiúsculas são DIFERENTES de minúsculas. Digite `X` em vez de `x`:

```
> X
Error: Object "X" not found
```

O programa informa que o objeto `X` não existe (existe o `x`).

Mas não era disso que estávamos falando. Vamos voltar para os diferentes conteúdos de um vetor... Que tal entrarmos com um caracter, por exemplo, com a palavra banana?

```
> x <- "banana"
> x
[1] "banana"
```

Percebeu a diferença deste caso para o caso numérico? Sempre que estivermos trabalhando com caracteres devemos usar aspas duplas. Você deve estar se perguntando o que teria ocorrido se não tivéssemos utilizado aspas na palavra banana... O problema é que se não fizermos isso, o R vai pensar que banana é o nome de um outro objeto...

```
> x <- banana
Error: Object "banana" not found
```

Viu só? O R reclama que o objeto banana não existe. Mas e se esse objeto existisse? O que aconteceria? O R copiaria o conteúdo do objeto `banana` para o objeto `x`. Estranho? Não. Essa é a

maneira de se fazer uma cópia de um objeto no R. Vamos criar um objeto chamado `banana`, contendo o número 7 e copiar o seu conteúdo para o objeto `x`:

```
> banana <- 7
> banana
[1] 7
> x <- banana
> x
[1] 7
```

Pegou? Legal. Agora, você deve ter notado um probleminha... a gente tinha colocado no objeto `x` primeiramente o número 3, depois a palavra banana e por fim o número 7, quando copiamos o conteúdo do objeto `banana` para `x`. Repare que as substituições de conteúdo do objeto `x` foram feitas sem nenhuma cerimônia pelo R. Pois é, isso pode ser um problema no R: ele não pergunta se você quer ou não substituir o conteúdo de um objeto com um nome, por outro. Para evitar esses acidentes, vamos aprender mais tarde uns macetes para economizar digitação... Aguardem...

;-)

Até agora só vimos exemplos que raramente usaremos, né? Quem é que vai entrar UM valor em um vetor??? Embora façamos isso muitas vezes no caso de programação avançada, vamos ver algo mais interessante. No R, para entrar com vários números (ou nomes, ou qualquer outro grupo de coisas), precisamos usar uma função para dizer ao programa que os valores serão *combinados* em um único vetor. Vamos tentar:

```
> x <- c(1,2,3,4,5)
> x
[1] 1 2 3 4 5
```

A essa altura, você já notou que a função usada foi `c()` e serve evidentemente para combinar elementos. Vamos ver com nomes:

```
> x <- c("banana", "laranja", "tangerina")
> x
[1] "banana"    "laranja"   "tangerina"
```

Por último, vamos abordar uma outra função que vai ser muito usada no R, e que a gente queria apresentar para você no contexto de vetores. É a função usada para gerar uma `seq`üência de números. Imagine como seria esta função... Acertou:

```
>x <- seq(from=1, to=12)
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Usamos a função `seq` para criar uma seqüência de (`from`) 1 a (`to`) 12. Bom, lembra quando a gente disse que era preguiçoso? Pois é: dá para fazer a mesma coisa de duas maneiras mais rápidas. Uma é omitindo os argumentos `from` e `to`:

```
> x<- seq(1, 12)
> x
[1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Agora, a campeã da preguiça mesmo seria:

```
> x<-1:12
> x
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12
```

É muita mesmo, não é???

Uma outra facilidade oferecida pelo R é a capacidade de selecionar valores armazenados em posições específicas dentro de um vetor (na verdade, como veremos adiante, essa capacidade se estende para outros objetos do R como matrizes e *data frames*). Vamos, por exemplo, criar o vetor `bicho` (só para treinar):

```
> bicho<-c("macaco","pato","galinha","porco")
> bicho
[1] "macaco"  "pato"    "galinha" "porco"
```

Que tal visualizarmos o conteúdo da posição 3? Vamos fazer assim:

```
> bicho[3]
[1] "galinha"
```

Repare que para selecionar uma posição específica dentro de um vetor, usamos o nome do vetor seguido de um número entre colchetes. Esse número corresponde à posição do elemento nesse vetor (confira acima, como `"galinha"` foi o terceiro elemento a ser entrado.) E se quiséssemos selecionar os conteúdos das posições 1, 3 e 4?

```
> bicho[c(1,3,4)]
[1] "macaco"  "galinha" "porco"
```

A gente utilizou a função `c()` para combinar os números correspondentes às coordenadas dos elementos (na verdade criamos um vetor com esses valores), ainda entre colchetes. E podemos ainda selecionar os conteúdos através de uma seqüência de posições, sempre entre os colchetes:

```
> bicho[2:4]
[1] "pato"    "galinha" "porco"
```

Bom, acho que deu para pegar a idéia de seleção, né? É só usar os colchetes. Porque você não tenta brincar um pouco agora, criando outros vetores e tentando selecionar elementos?

Operações vetoriais

Bem, vamos mudar um pouco de assunto agora. Já que estamos falando de vetores e já vimos como o R funciona como uma calculadora, que tal agora vermos como o R combina essas duas características para realizar operações vetoriais? Ih! Que palavrão é esse!!!??? Não se assuste, é só uma maneira que o R tem para facilitar a sua vida. Vamos pegar um exemplo qualquer bem simples, como um vetor de pesos de pessoas em kg e alturas em metros. Vamos fazer assim:

```
peso <- c(62, 70, 52, 98, 90, 70)
altura <- c(1.70, 1.82, 1.75, 1.94, 1.84, 1.61)
```

Muito bem, agora que tal calcularmos o índice de massa corporal (IMC) para essas pessoas? Não sabe como calcula? É fácil: ele é simplesmente o peso em kg dividido pela altura em metros ao quadrado. Vamos tentar fazer tudo de uma vez? Tente:

```
imc <- peso/altura^2
```

Vamos ver agora o resultado:

```
> imc
[1] 21.45329 21.13271 16.97959 26.03890 26.58318 27.00513
```

Percebeu o que aconteceu? O R calculou o IMC para cada posição dos vetores de peso e altura. Por exemplo, o primeiro resultado 21.45329 é o resultado da aplicação da fórmula proposta sobre o primeiro elemento de `peso` e o primeiro elemento de `altura`. Confira: $62/1.70^2 = 21.45329$. Confira os outros resultados.

Esse procedimento é uma operação vetorial, ou seja ela foi aplicada a um vetor inteiro.

Esse é um bom momento para falar sobre precedência de operações matemáticas no R. Bem, o R segue a precedência que a gente aprendeu lá no ensino fundamental em matemática, lembra? No acima, a potenciação tem precedência em relação à divisão e portanto a nossa conta foi feita corretamente. Lembre-se que se a divisão fosse feita antes da potenciação, o resultado seria diferente. Quer ver? Como se muda a precedência de operações? Isso mesmo: usando parênteses, igualzinho como você aprendeu:

```
> (peso/altura)^2
[1] 1330.1038 1479.2899  882.9388 2551.8121 2392.4858 1890.3592
```

Epa! Não era bem isso que a gente queria... Bem, assim fica claro que se a gente não quiser se confundir nunca com os resultados, o uso do parênteses é sempre uma boa política quando não se tem certeza de uma determinada operação...

Geração de Distribuições e Gráficos

Vamos ver agora algumas funções no R para a geração de distribuições estatísticas. Esta evidentemente é uma das especialidades do R, já que ele é voltado para funções estatísticas principalmente.

Comecemos com as distribuições contínuas. A mais badalada de todas, naturalmente é a Normal, que você já deve estar cansado de ouvir falar. Vamos gerar então um vetor com 100 valores de uma distribuição Normal, digamos com média 10 e variância 4 (estes são os parâmetros de uma distribuição Normal):

```
> x <- rnorm(100, mean=10, sd=2)
```

Repare que a função se chama `rnorm` e que os argumentos usados foram o número de valores a serem gerados, a média (`mean`) e o desvio-padrão (`sd` – de *standard deviation* em inglês). Cabem dois comentários: primeiro o nome da função. Bom, o `norm` vem de Normal, é claro e o `r` vem de *random*, aleatório em inglês, pois a gente está gerando aleatoriamente 100 valores de uma distribuição Normal(10, 4). A segunda observação é que a rigor (embora nem todos sejam rigorosos) o segundo parâmetro da Normal é a sua variância (e não o seu desvio-padrão). De qualquer maneira, o R recebe o argumento relativo ao desvio-padrão e não à variância.

Você pode estar pensando em duas coisas. Primeiro: e se eu só tivesse a média e a variância e não o desvio padrão em um problema e quisesse gerar a Normal? Eu teria que fazer a conta e colocar o valor? Você pode fazer isso... Entretanto, o R aceita colocar uma função dentro de outra função. Assim, o mesmo resultado pode ser obtido fazendo:

```
x <- rnorm(100, mean=10, sd=sqrt(4))
```

A segunda questão é: será que a função `rnorm()` tem um *default*? A resposta é sim, pelo menos para os argumentos `mean` e `sd`. Ganha um doce quem adivinhar quais são os padrões... Acertou quem pensou na mais famosa das Normais: A Normal Padronizada ou Normal (0, 1), ou

seja, média zero e variância (ou desvio-padrão) 1. Para verificar esse fato, vamos usar duas funções estatísticas, para calcular a média e o desvio-padrão de vetores. Vamos começar pelo vetor que a gente conhece:

```
> x <- rnorm(100)
> mean(x)
[1] -0.1358806
> sd(x)
[1] 1.053232
```

Epa!!! A média aqui no nosso exemplo não é exatamente 0. E o desvio-padrão também não é exatamente 1. Será que o *default* da função não é gerar a partir da Normal (0,1)?

Não é nada disso! O problema é que o vetor que nós criamos é gerado de maneira aleatória, de modo que, em média, esses valores convergem assintoticamente para os valores estabelecidos para os parâmetros. Ficou difícil? É o seguinte: se a gente aumentar o número de valores gerados, esses valores devem ir se aproximando cada vez mais dos valores dos parâmetros. Querem ver?

```
> x100 <- rnorm(100)
> mean(x100)
[1] -0.1111455
> sd(x100)
[1] 1.121295

> x1000 <- rnorm(1000)
> mean(x1000)
[1] -0.01045328
> sd(x1000)
[1] 1.024044

> x10000 <- rnorm(10000)
> mean(x10000)
[1] 0.005090138
> sd(x10000)
[1] 0.9953523
```

Ah, e se for fazer isso no R você mesmo, não digite tudo de novo: use a tecla de setinha para cima (↑) que você chama o último comando e pode editá-lo a partir do teclado... Mas atenção, para mover o curso na linha de comando você deve usar as setinhas para direita e esquerda (→ ou ←). Lembrando que você pode naturalmente sempre copiar e colar o texto – mas não se esqueça de não copiar o "`>`".

;-)

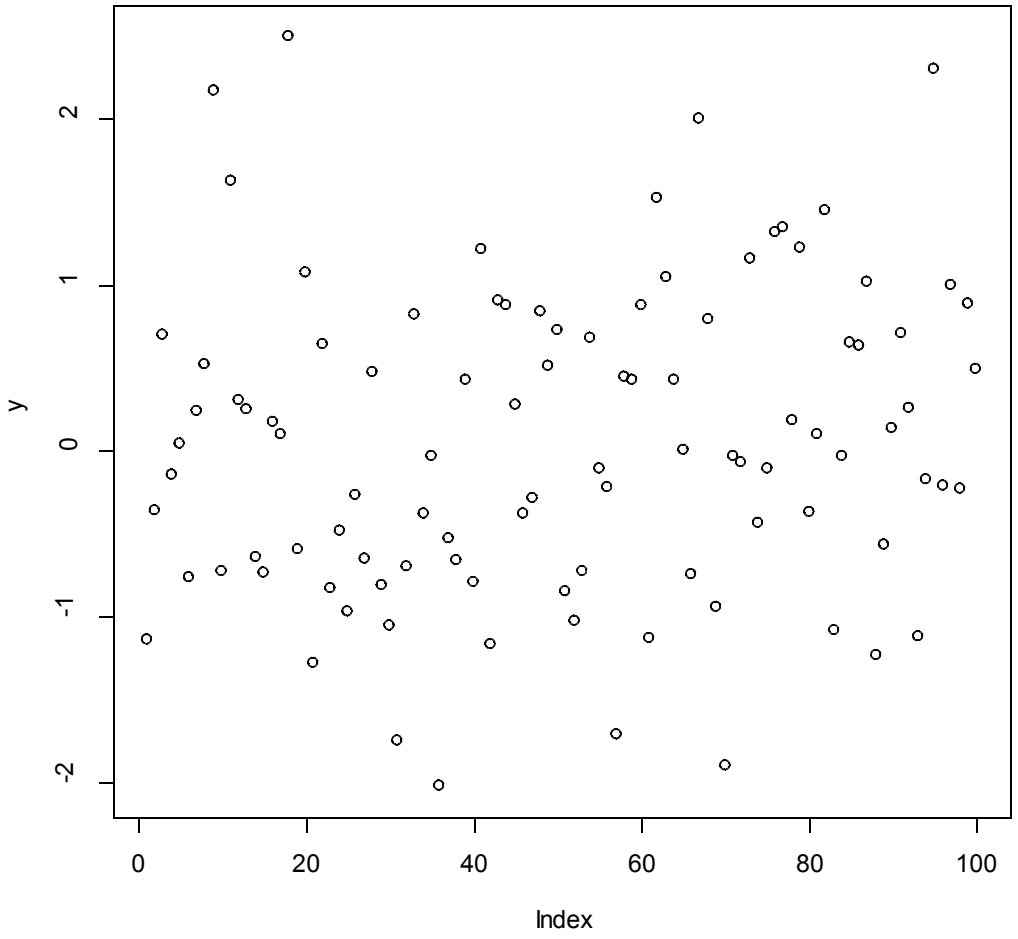
Muito bem, você acabou aprendendo a simular uma distribuição Normal com diferentes parâmetros e também a conferir a média e o desvio padrão do vetor que você gerou. Além disso, ganhou uma noção que a média e o desvio padrão de vetores maiores serão em princípio mais aproximados da média e desvio padrão das distribuições que os geraram.

Que tal uma visualização gráfica das nossas distribuições? Você já deve ter ouvido falar de tipos de gráficos comumente usados em estatística, como o *scatter plot* (ou gráfico de dispersão), o histograma, o *boxplot* (ou gráfico de caixas) e ainda um que gostamos muito, o *stem and leaves* (ou ramo e folhas). Por que a gente não aproveita então para dar uma olhada nesses gráficos usando os vetores que foram criados? Vamos lá. Primeiro vamos criar um vetor `y` com 100 valores de uma Normal Padronizada:

```
y <- rnorm(100)
```

Agora vamos *plotar* o nosso vetor:

```
plot(y)
```

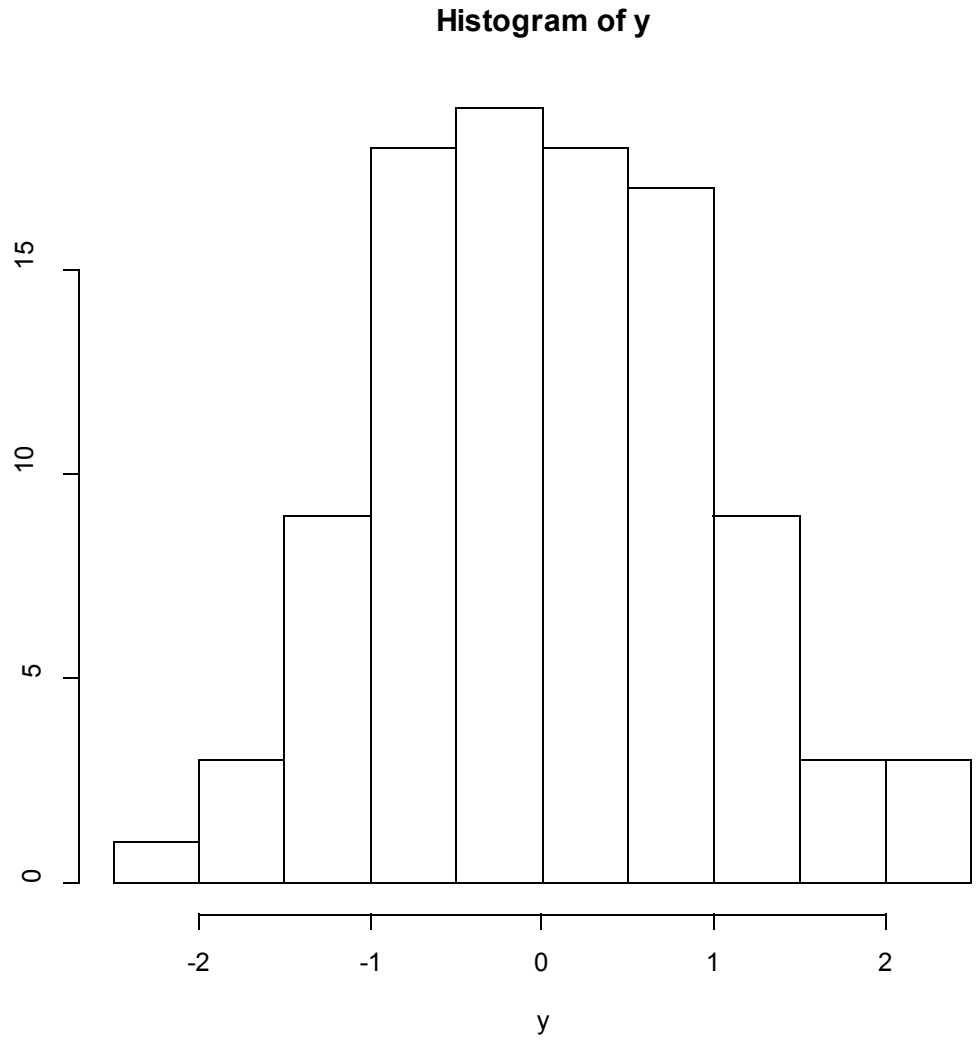


Se você digitou essa função no *prompt* do R, notou que aconteceu algo diferente. A saída dessa função não foi listada na mesma janela, mas uma outra janela diferente foi aberta e um gráfico foi mostrado. Os chamados gráficos de alto nível são sempre plotados nessa janela. O gráfico deve ser mais ou menos como esse aí em cima.

Alguns detalhes devem ser notados nessa figura. Primeiro, apesar da função ser `plot()`, o gráfico mostrado é um *scatter plot*. Esse é o *default* de uma função `plot()` para um vetor, onde o eixo *x* é o índice, ou seja a posição do elemento dentro do vetor, e o eixo do *y* é o valor dos elementos do vetor. O segundo detalhe são os valores do nosso vetor. Observe que, em se tratando de uma Normal Padronizada, cuja média é zero, a maior parte dos pontos se encontra concentrada em torno desse valor . Além disso, note que a esmagadora maioria dos pontos se concentra entre os valores –2 e 2. Você seria capaz de dizer qual a percentagem aproximada de pontos que deve estar entre esses valores?

Outro gráfico muito usado é o `hist`ograma. Vamos ver o jeitão dele:

```
hist(y)
```

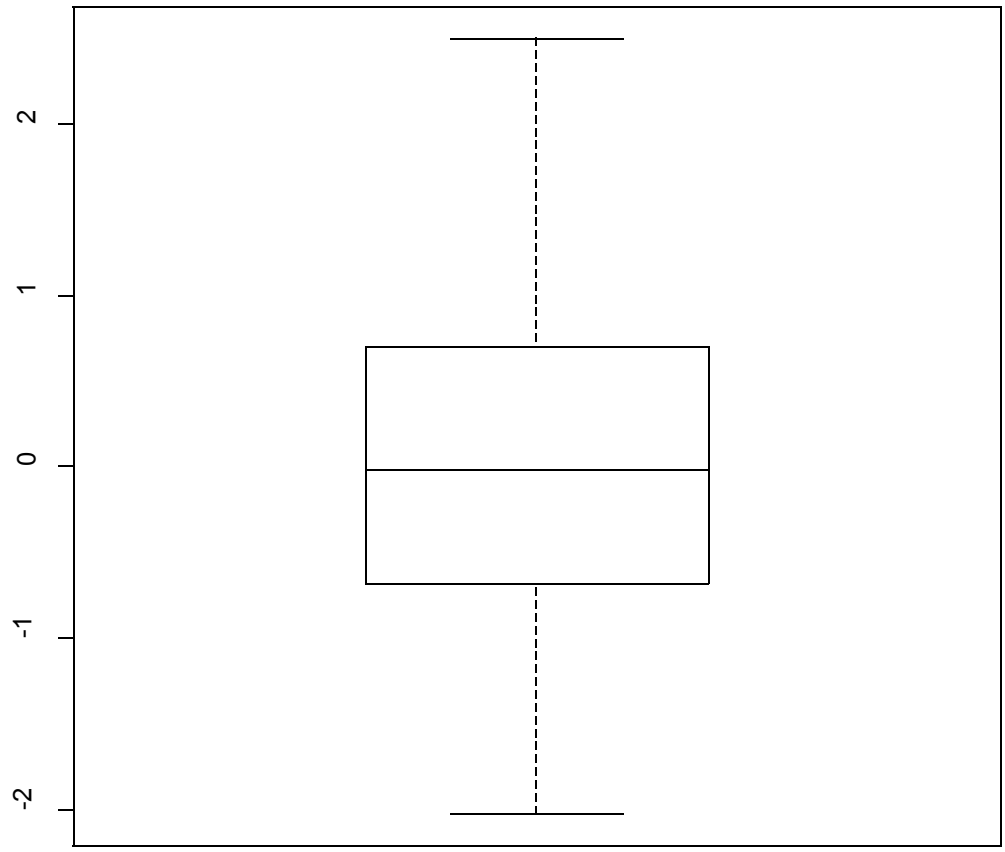


Repare como o histograma lembra bastante a densidade de uma distribuição Normal. Este gráfico tem os valores do vetor *y* no eixo horizontal e as freqüências (absolutas nesse caso) dos valores contidos nos intervalos pré-estabelecidos pelo programa no eixo vertical. Curioso para saber variações da função `hist()`? Consulte a ajuda. Lembra como?

```
> ?hist
```
OU
```
> help(hist)
```

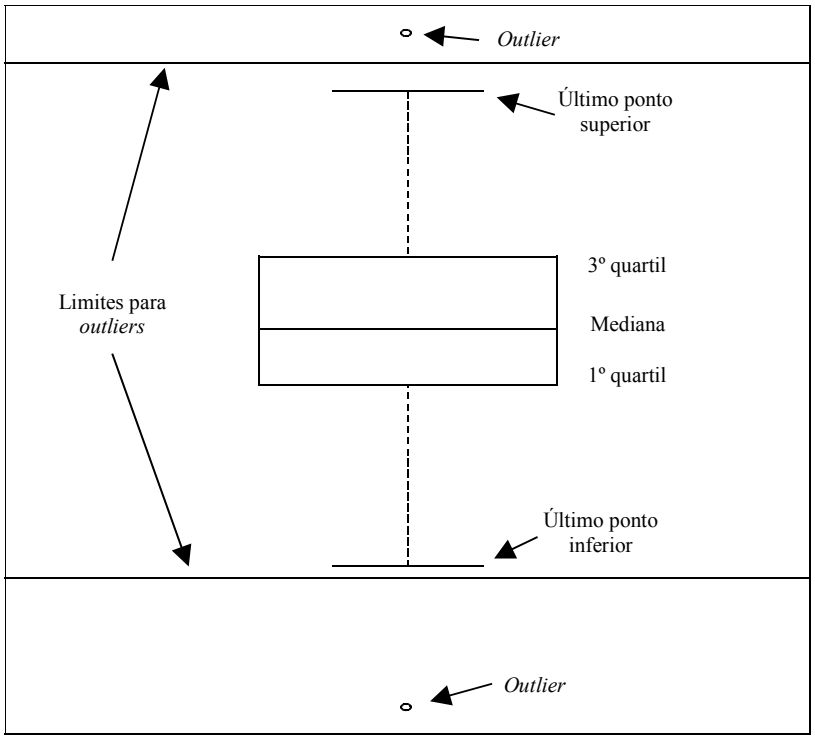
O *boxplot* é também um gráfico interessante principalmente para analisar a dispersão dos dados e também para detectar a presença de *outliers*. Para quem não sabe ou não se lembra, *outliers* são pontos que caem em uma região muito afastada do centro da distribuição (i.e. muito afastados da média e da mediana). Vamos explicar melhor com a visualização de um *boxplot* propriamente dito:

```
boxplot(y)
```



A linha central do retângulo (que seria a nossa "caixa") representa a **mediana** da distribuição (sabe o que significa mediana? Não? As bordas superior e inferior do retângulo representam o percentis 25 e 75, respectivamente (também conhecidos como primeiro e terceiro quartís, respectivamente). Logo, a altura deste retângulo é chamada de distribuição interquartilar (DI). Os traços horizontais ao final das linhas verticais são traçados sobre o último ponto (de um lado ou de outro) que não é considerado um *outlier*.

A essas alturas você deve estar estranhando a falta de definição de *outlier* não é mesmo? A nossa primeira definição foi algo subjetiva, certo? É isso mesmo! Não há um consenso sobre a definição de um *outlier*. Porém, no caso do *boxplot* em geral, existe uma definição formal. A maior parte das definições considera que pontos acima do valor do 3º quartil somado a 1,5 vezes a DI ou os pontos abaixo do valor do 1º quartil diminuído de 1,5 vezes a DI são considerados *outliers*. Esses pontos são assinalados (no nosso exemplo, tivemos 2 *outliers*, um para cada lado). Vamos ver de novo o *boxplot,* para ficar mais claro:



Para o nosso próximo gráfico, que tal a gente tentar gerar outra distribuição também bastante comum em bioestatística, a binomial. Ao contrário da Normal, essa é uma distribuição *discreta*, ou seja, ela está definida para determinados valores (enumeráveis) num intervalo e não para *todos* os valores (não enumeráveis) em um determinado intervalo.

Assim como a Normal, a Binomial também possui dois parâmetros. O primeiro, *n* corresponde ao número de experimentos que serão realizados (também chamados de experimentos ou processos de Bernoulli); o segundo, *p* é a probabilidade de se obter um sucesso em cada um dos experimentos. A probabilidade de serem observados *k* sucessos é dada por:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

No R, a função para gerar números binomiais chama-se `rbinom()`, por motivos já óbvios. Evidentemente, nós precisamos especificar os dois parâmetros que mencionamos anteriormente:

```
y <- rbinom(100, size=200, p=0.05)
```

Geramos então também 100 pontos de uma Binomial cuja probabilidade de sucesso é de 5% e o número de experimentos (observações) é de 200. Esses parâmetros foram escolhidos propositadamente para fazer um paralelo com uma situação que é comum em epidemiologia: termos cerca de 200 participantes em um estudo (correspondendo ao nosso `size = 200`) e com uma prevalência de uma determinada doença de 5% (correspondente a uma probabilidade de sucesso de 0.05 por indivíduo). Dado este raciocínio, quantos participantes devem estar doentes, em média? A resposta vem mais tarde...

Para construir o *ramo e folhas* utilizamos a função `stem()`:

```
> stem(y)

  The decimal point is at the |

   2 | 0
   4 | 000000
   6 | 0000000000
   8 | 00000000000000000000
  10 | 0000000000000000000000000000
  12 | 000000000000000000000000000
  14 | 000000000
  16 | 00
  18 | 00
```

A idéia do ramo e folhas é separar um número (como 7,0) em duas partes. Neste caso, a primeiro parte inteira (7) chamada de ramo e a segunda, a parte decimal (0) chamada de folha. Além de separar os números em duas partes (inteira e decimal), o R agrupa os números em classes de tamanho 2. Por exemplo, o ramo 4 leva em conta os números 4 e 5. Não gostou dessa disposição? Não tem problema. Que tal dobrarmos a escala do gráfico para o R considerar cada número separadamente na sua própria classe?

```
> stem(y, scale=2)
  The decimal point is at the |
   3 | 0
   4 | 0000
   5 | 00
   6 | 00000
   7 | 00000000
   8 | 0000000000000000
   9 | 000000000000
  10 | 0000000000000000000
  11 | 0000000000
  12 | 00000000
  13 | 000000
  14 | 0000
  15 | 000
  16 | 00
  17 |
  18 |
  19 |
  20 |
  21 | 0
```

Já percebeu que o argumento extra para dobrar a escala é `scale=2`, né?

E por que a distribuição dos dados parece concentrada em torno de 10 (hum... nem tanto...)?

Bem, o que fizemos aqui foi gerar 100 amostras de 200 pessoas, contando o número de pessoas com a doença em cada amostra. Se a prevalência da doença é 0.05 a gente espera que em uma amostra de 200 pessoas, o número de pessoas com a doença seja igual a 10 = 200x0.05!! Mas como o processo é aleatório, existe a situação onde apenas 3 pessoas são afetadas pela doença e ainda situações extremas de 21 pessoas (mais que o dobro do esperado).

Vamos fazer um exercício interessante? O exercício consiste em observar o comportamento do número de pessoas afetadas por uma doença na medida em que aumentamos o valor da prevalência. Antes disso, vamos dividir o espaço gráfico em 6 partes através do comando:

```
par(mfrow=c(2,3))
```

Ih... O Que é essa função `par()`??? Como você deve ter observado, uma janela de gráficos foi aberta. Pois é, essa é a função para atribuir parâmetros gráficos. O argumento `mfrow=` estabelece o número de gráficos que serão visualizados em uma mesma janela gráfica e em que disposição. Quando escrevemos no comando `c(2,3)`, ele implicitamente divide essa janela em 6 partes: 2 linhas e 3 colunas (de gráficos, né?) e que você vai visualizar quando plotarmos 6 gráficos em seqüência.

Escolhendo valores de prevalência 0.02, 0.04, 0.06, 0.08, 0.1 e 0.5 podemos fazer:

```
hist(rbinom(100, size=200, p=0.02))
hist(rbinom(100, size=200, p=0.04))
hist(rbinom(100, size=200, p=0.06))
hist(rbinom(100, size=200, p=0.08))
hist(rbinom(100, size=200, p=0.1))
hist(rbinom(100, size=200, p=0.5))
par(mfrow=c(1,1))
```

Beleza? Podemos visualizar como se altera a distribuição do número de pessoas atingidas pela doença na medida em que aumentamos o valor da prevalência.

Matrizes

Bem, vamos passar para o próximo tipo de objeto que vamos aprender no R. São as matrizes. Como você deve saber, matrizes são objetos numéricos, que possuem elementos com coordenadas (que são simplesmente a linha e a coluna às quais o elemento pertence). Para construir um objeto que seja uma matriz no R, precisamos usar uma função... Adivinha o nome:

```
> x <- matrix(c(1,2,3,4,5,6,7,8,9,10,11,12), ncol=3)
> x
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Muito bem. Neste momento você deve estar desesperado. Por que usamos uma função que tinha sido usada para criar um vetor *dentro* de uma outra função do R?

É isso mesmo. Na verdade "criamos" um vetor temporário com o a função `c()` como tínhamos visto anteriormente e depois esse vetor foi transformado numa matriz com a função `matrix()`. Quer ver? Vamos fazer por partes. Primeiro, crie o vetor "y":

```
> y <- c(1,2,3,4,5,6,7,8,9,10,11,12)
> y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Lembre-se que esse vetor poderia ter sido gerado com a nossa seqüência:

```
> y <- 1:12
> y
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

Agora, aplique a função `matrix()` ao vetor `y`:

```
> x <- matrix(y, ncol=3)
> x
     [,1] [,2] [,3]
```

```
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

A próxima dúvida é quanto ao argumento `ncol`. Ele representa o número de colunas da nossa gloriosa matriz. Como se trata de um vetor, o R não pode adivinhar qual a dimensão (número de linhas e número de colunas desejada da matriz. Quer ver?

```
> x <- matrix(y)
> x
      [,1]
 [1,]    1
 [2,]    2
 [3,]    3
 [4,]    4
 [5,]    5
 [6,]    6
 [7,]    7
 [8,]    8
 [9,]    9
[10,]   10
[11,]   11
[12,]   12
```

A essa altura você já deve ter notado que o *default* do R é transformar o vetor em uma matriz com apenas uma coluna. Um outro detalhe importante a ser comentado é a ordem na qual ele entra com os elementos na matriz. Observe que o preenchimento da mesma é feito pelas colunas. Vamos ver de novo a nossa matriz:

```
> x <- matrix(y, ncol=3)
> x
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Viu? O programa preencheu a coluna 1 com os números 1 a 4, a segunda com os números de 5 a 8 e a terceira com os demais. Mais tarde vamos aprender a mudar este comportamento. Mas se você for curioso, use a ajuda.... e boa sorte!

Agora você deve estar se perguntando como é possível visualizar um elemento (ou um grupo de elementos) contido numa matriz. A lógica é a mesma que com vetores, sendo que no caso das matrizes, os elementos possuem 2 coordenadas: uma para a linha e outra para a coluna, usando ainda os nossos colchetes – lembra?. Quer ver um exemplo? Vamos visualizar o elemento da segunda linha, terceira coluna de `x`:

```
> x[2,3]
[1] 10
```

Como você já deve estar pensando, é possível selecionar, como nos vetores, um intervalo de valores. Digamos que você queira visualizar os 3 primeiros elementos da primeira coluna. Para isso façamos:

```
> x[1:3,1]
[1] 1 2 3
```

No caso das matrizes é possível selecionar uma linha (ou coluna) inteira, sem se preocupar em saber o número de colunas (ou linhas) da matriz. Não entendeu nada, né? Nem eu! Vamos tentar um exemplo: vamos selecionar as duas primeiras linhas da matriz, selecionando todas as colunas dessas linhas:

```
x[1:2,]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
```

Repare que o espaço depois da vírgula, que seria destinado às coordenadas das colunas, ficou em branco, indicando que TODAS as colunas deveriam ser selecionadas. Pode-se fazer um raciocínio semelhante para o caso de selecionar colunas inteiras, ou seja, com todas as linhas. Tente você mesmo!

Data Frames

O último tipo de objeto do R que vamos abordar neste material (é isso mesmo, tem mais...) são os chamados *data frames*. Esses objetos são equivalentes a um banco de dados que você provavelmente já viu em outros formatos (e.g. dbf), ou seja, trata-se de uma "tabela de dados" onde as colunas são as variáveis e as linhas são os registros.

Vamos ver um exemplo agora com um banco de dados que vem acompanhando o R (vários bancos de dados estão disponíveis para trabalhar exemplos no R). Para tal, teremos primeiro que invocar esse banco e depois salvá-lo num outro objeto para evitar "acidentes"

;-)

```
data(iris)
dados <- iris
```

Acho que não tem muito mistério. Nós usamos a função `data ()` para "chamar um banco de dados chamado `iris` e depois colocamos o conteúdo desse banco em um objeto chamado `dados`.

Aliás, essa mesma função serve para visualizar todos os bancos disponíveis como exemplo no R. Tente:

```
data()
```

E veja o que acontece!

Vamos agora dar uma olhadinha em alguns registros do nosso novo objeto; na verdade, vamos olhar apenas os primeiros 10 registros:

```
> dados[1:10, 1:5]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
```

Repare que não há nada de novo: continuamos usando os colchetes para selecionar os registros. Mas você deve estar se perguntando por que usamos esse `[1:10, 1:5]`. Se com as matrizes dá para selecionar uma linha inteira, porque não poderia com o *data frame* que nada mais é do que uma matriz com características especiais. Acertou: A mesma lógica funciona aqui. Vamos tentar:

```
> dados[1:10,]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
```

Que tal iniciarmos uma pequena exploração desse banco de dados? Vamos começar uma descrição dos dados, conhecendo o nome das variáveis que estão contidas neste banco (é claro que você já sabe esses nomes, né?)

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

A descrição dessas variáveis são:

- Sepal.Length – Tamanho da sépala
- Sepal.Width – Largura da sépala
- Petal.Length – Tamanho da pétala
- Petal.Width – Largura da pétala
- Species – Espécie observada

E se quisermos listar uma das variáveis apenas?

Você naturalmente respondeu que podemos selecionar uma das colunas inteiras do banco usando aquele nosso velho conhecido. Vamos selecionar a variável `Species`, que corresponde à terceira coluna:

```
dados[,5]
```

Vamos omitir a saída aqui para poupar espaço... Mas será que existe uma outra maneira de selecionar uma variável em um banco de dados no R? Uma maneira mais convencional, por exemplo, seria chamar o nome da variável e não tendo que saber especificamente a coluna correspondente.

Tente:

```
dados$Species
```

Viu? O símbolo `$` serve para indicar o nome de uma variável em um *data frame*.

Digamos agora que você queira listar todos os registros de flores que têm um tamanho de pétala acima de 6 cm. Nesse caso, estaremos selecionando todas as colunas do banco e somente as linhas que satisfizerem essa condição. Vamos ver como seria, usando os nossos colchetes:

```
> dados[dados$Petal.Length>6,]
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
106          7.6         3.0          6.6         2.1 virginica
108          7.3         2.9          6.3         1.8 virginica
110          7.2         3.6          6.1         2.5 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
123          7.7         2.8          6.7         2.0 virginica
131          7.4         2.8          6.1         1.9 virginica
132          7.9         3.8          6.4         2.0 virginica
136          7.7         3.0          6.1         2.3 virginica
```

Entendeu o que foi feito? Selecionamos (com os colchetes), do banco `dados` as linhas (primeiro índice dentro dos colchetes, antes da vírgula) onde a variável `Petal.Length` é > 6, e todas as colunas (o espaço em branco após a vírgula dentro dos colchetes – segundo índice).

Para terminar, vamos fazer uma brincadeira com o nosso banco de dados. Ele não possui nenhum valor faltante (*missing value*). No R, o "símbolo" para *missing* é "NA" de *not available* em inglês. Bem, vamos aproveitar o nosso exercício de seleção para ver como podemos também substituir valores selecionados por outro. Digamos que houve um engano e que todos os valores maiores que 6 cm para o tamanho da pétala não são confiáveis e devem ser considerados como faltantes (esse é um exemplo péssimo na verdade – normalmente nós substituímos valores que outros sistemas consideram *missing* por NA, como "-99" ou espaço em branco.) Neste caso a sintaxe será um pouco diferente:

```
dados$Petal.Length[dados$Petal.Length>6] <- NA
```

Explicando devagar: estamos substituindo (com o símbolo `<-`) os valores da variável `Petal.Length` (o primeiro `dados$Petal.Length`), que são maiores que 6 `(dados$Petal.Length>6)` por `NA`. Quer ver se funciona? Tente:

```
dados$Petal.Length
```

Repare que a saída é mesmo um vetor com os valores da variável `Petal.Length`.

Explorando um Data Frame

Vamos agora comparar os tamanhos de pétalas de acordo com 3 grupos: os das espécies setosa, versicolor e virgínica. Para isso, lançaremos mão de uma função especial:

```
by(data=dados$Petal.Length, INDICES=dados$Species, FUN=summary)
```

Essa função é especial porque ela aplica uma outra função (no caso a função `summary()` – argumento `FUN`) a uma variável (argumento `data=`) estratificado por uma outra variável (argumento `INDICES`). Pegou? Repare qual é a saída da função `summary()`: ela calcula a média, mediana mínimo, máximo, primeiro e terceiro quartís, além de reportar o número de *missings*.

Você deve ter notado que é muito trabalhoso digitar o nome do *data frame* toda vez que a gente quiser trabalhar com uma de suas variáveis. Para contornar esse problema existe uma função no R que permite acessar as variáveis diretamente. Vejamos

```
attach(dados)
```

Para testar vamos repetir a função `by()` omitindo o nome do *data frame* e também o `$`.

```
by(Petal.Length, Species, summary)
```

Viu como somos preguiçosos? Nem o nome dos argumentos foram usados neste caso. Lembre-se que a omissão dos nomes dos argumentos pode causar uma enorme confusão caso você esqueça a ordem correta (lembra do caso do log?).

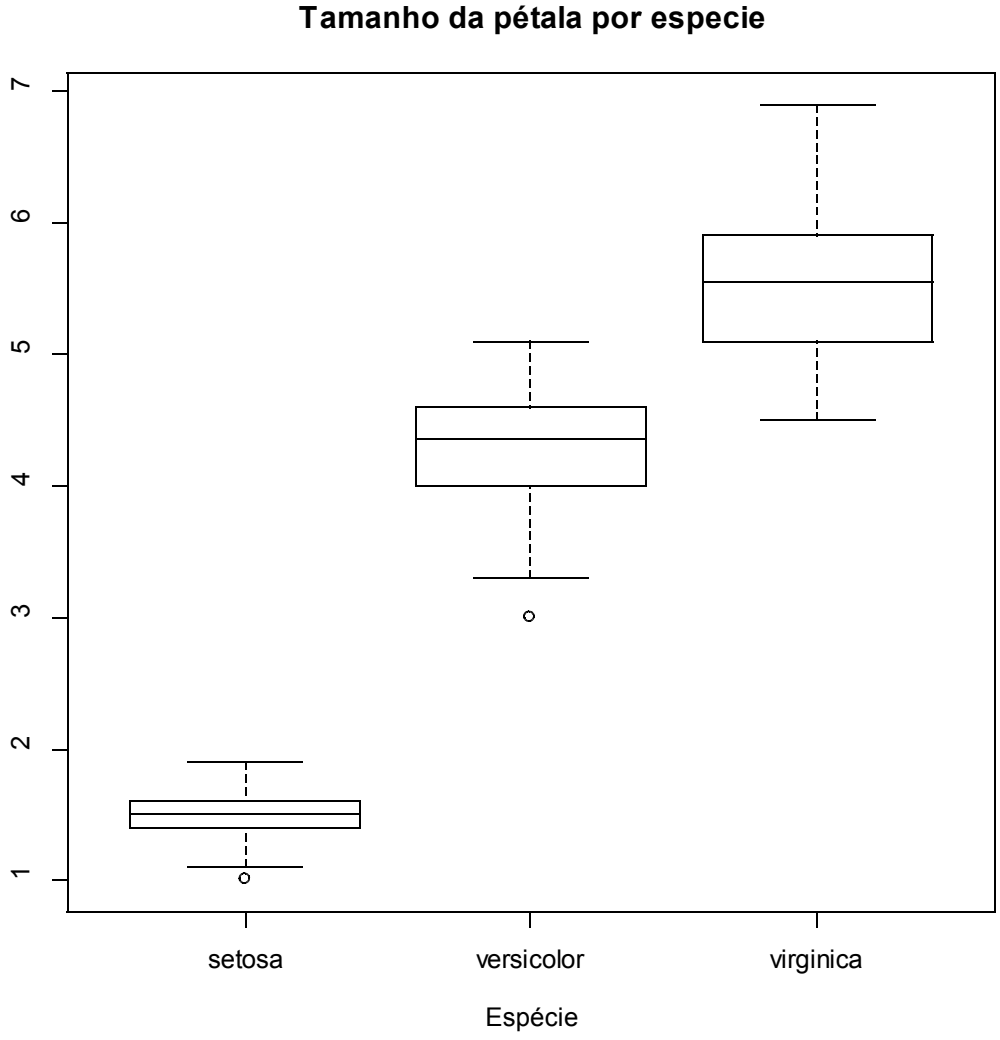
Vamos visualizar os dados com um gráfico que a gente já tinha visto anteriormente, que é o `boxplot()` com alguns argumentos avançados. Para isso escrevamos:

```
> boxplot(Petal.Length ~ Species)
```

Entendeu o que foi feito? O símbolo `~` (til) na função `boxplot()` funciona como a função `by()` que vimos anteriormente, ou seja, visualizamos o peso estratificado pela variável `Species`

Você pode adicionar alguns nomes aos gráficos para que ele fique mais apresentável

```
> boxplot(Petal.Length ~ Species, xlab="Espécie", ylab="Tamanho da
Pétala", main="Tamanho da pétala por especie")
```



As espécies parecem ter tamanhos de pétalas bastante diferentes. Suas medianas e distribuições interquartilares são bem diferentes. Para ter uma idéia se essas distribuições são aproximadamente normais podemos usar histogramas. Vamos tentar

```
par(mfrow=c(1,3))
hist(Petal.Length[Species=="setosa"])
hist(Petal.Length[Species=="versicolor"])
hist(Petal.Length[Species=="virginica"])
par(mfrow=c(1,1))
```

Epa! Parece que o nosso "tratamento" para *missing values* esculhambou o grupo das virgínicas (que são justamente as que têm um tamanho de pétala maior.)

;-)

Vamos "reiniciar" o nosso objeto `dados` e ver o que acontece. Faça assim:

```
detach(dados)
dados<-iris
attach(dados)
par(mfrow=c(1,3))
hist(Petal.Length[Species=="setosa"])
hist(Petal.Length[Species=="versicolor"])
hist(Petal.Length[Species=="virginica"])
par(mfrow=c(1,1))
```

Melhor assim, né? Repare que as distribuições não parecem ser lá muito aproximadamente normais para o tamanho da pétala, exceto para as setosas.

Um detalhe que não deve ter passado despercebido é o uso do operador "==" em vez de um sinal de igual somente para indicar igualdade. Não se desespere ainda: no R sempre que quisermos fazer uma comparação, seja numérica, seja com caracter, devemos usar os dois sinais de igual. Ah, e note também que `"virginica"` está entre aspas – isso mesmo, quando comparamos com um caracter, o que está sendo comparado deve estar entre aspas.

Para finalizar esse módulo básico, vamos aplicar um teste estatístico para inferir se o tamanho das pétalas são significativamente diferentes entre as espécies. Como não estou muito convencido da normalidade das distribuições, vamos usar um teste não-paramétrico (também chamado de livre de distribuição.) Esse tipo de teste não assume qualquer distribuição a priori e a rigor é um teste para saber se as medianas são diferentes. No caso de mais de dois grupos, devemos lançar mão de um teste bastante popular, chamado teste de Kruska-Wallis:

```
> kruskal.test(Petal.Length, Species)

        Kruskal-Wallis rank sum test

data:  Petal.Length and Species
Kruskal-Wallis chi-squared = 130.411, df = 2, p-value = < 2.2e-16
```

Como esperado após a inspeção do gráfico *boxplot*, de fato existe uma diferença significativa do nível de variação dos tamanhos de pétalas estratificados pela espécie de flor.

Muito bem. Esse foi o nosso módulo básico para dar uma noção geral e bastante superficial do R. Para quem estiver interessado em um estudo mais aprofundado do programa, futuramente disponibilizaremos módulos mais específicos para aprofundar os vários aspectos do R

Módulo Entrada e Saída de Dados

Autor: Geraldo Marcelo da Cunha e Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Saber como funcionam e como usar funções e pacotes no R. Saber selecionar elementos, linhas e colunas em *dataframes*.
ATENÇÃO: Este módulo também exige a obtenção de bancos de dados externos, que devem estar disponíveis junto com o meio de distribuição deste documento (ou seja, se foi em um CD-ROM, então deve estar em algum diretório desta mídia; se foi via internet deve estar disponível para *download* em algum lugar especificado)

Pacotes e arquivos necessários: Pacote `foreign`; arquivos "oswego.rec", "iris.sav".

Saída de Dados

Vamos iniciar a nossa discussão pela saída de dados. Apesar de parecer estranho estudarmos como o R exporta dados, em vez de como ele os importa, será mais fácil primeiro exportar para depois então reimportamos alguns formatos.

Como mencionado, o pacote `foreign` deve ser instalado na sua máquina. Se este pacote não estiver instalado, instale-o a partir do próprio R. Para instruções sobre como fazer isso, consulte o módulo Baixando e Instalando o R. Esse pacote, como sugere o nome, contém várias funções tanto para importar como para exportar alguns formatos de bancos de dados entre o R e diversos programas.

Uma vez instalado o pacote, você precisa carregá-lo. Faça assim:

```
library(foreign)
```

Aproveite para dar uma olhada nas funções contidas nesse pacote. Use a ajuda em HTML para facilitar a sua vida. Se você for curioso bastante para explorar o pacote `foreign`, notará que ele só possui na verdade uma função para exportar diretamente dados para um formato específico, que é o Stata, e que a maioria absoluta das funções são para importar dados.

Mas o R é capaz de exportar também para o formato ASCII, que pode ser considerado como um formato universal, ou seja, qualquer programa é capaz de ler um arquivo nesse formato. Nessa seção, vamos ver uma função que faz essa tarefa, usando diferentes argumentos para a exportação, convenientes para o programa que vai ler os dados. Não estranhe, mas a função utilizada para este fim – `write.table()` – faz parte do pacote `base`, que é instalada junto com a instalação básica do R.

O caso mais geral é exportar dados num formato ASCII delimitado por espaço ou tabulação. Vamos começar com a separação por tabulação. Este formato pode ser lido por diversos programas, como Excel, SPSS, SAS, dentre outros.

Se você já fez o módulo básico, deve se lembrar do banco de dados `iris` com algumas medidas de pétalas e sépalas de algumas espécies de flores. Bem, vamos invocar esse mesmo banco e trabalhar com ele:

```
data(iris)
dados <- iris
```

Vamos usar a função `write.table()`:

```
write.table(dados, file="iris.dat", sep="\t", row.names=F, quote = FALSE)
```

Os 3 primeiros argumentos da função `write.table()` não devem suscitar dúvidas: o primeiro é o nome do objeto a ser exportado, o segundo, o nome do arquivo onde o dado será armazenado e o terceiro o tipo de delimitador a ser usado, que neste caso é esse `"\t"` que significa tabulações. O argumento `row.names=F` serve para indicar que as linhas desse objeto não têm nome e ainda previne o R de colocar números como nomes (por *default* o R vai fazer uma numeração crescente, como se fosse o número do registro, se esse argumento for `T`). O argumento `quote = FALSE` serve para indicar ao R que variáveis tipo caracter devem ser exportadas sem estar entre aspas (porque se estiverem o SPSS por exemplo não vai levar isso em conta e vai importar as aspas junto).

O arquivo gerado "iris.dat" pode ser facilmente importado para diversos programas. A técnica de importação para outros programas não faz parte do escopo desse módulo e deverá ser dominado por pessoas que trabalham com os respectivos programas de uma forma regular.

Uma dúvida natural do leitor nesse momento seria: legal exportei os dados para um arquivo, mas onde está este arquivo??? Boa pergunta! Por *default*, o R salva o arquivo no mesmo diretório de onde você está trabalhando, ou seja no mesmo diretório onde está o seu arquivo ".Rdata". Se você quiser saber que diretório é esse, faça assim:

```
getwd()
```

Se você utilizou esse código, notou uma coisa estranha: o diretório veio com duas barras invertidas, em vez de uma só como é usual, se você tem alguma experiência com caminhos de diretórios do DOS, algo assim:

```
"C:\\Documents and Settings\\Owner\\My Documents\\ENSP\\Curso R"
```

É assim mesmo: no R nós temos que escrever os caminhos de diretórios com duas barras invertidas. Opcionalmente, podemo usar uma barra para frente, como veremos num exemplo adiante.

O outro formato que vamos ver nessa seção é o separado por vírguas, ou ".csv", um formato que é lido diretamente pelo Excel. A idéia é fundamentalmente a mesma, só diferindo o delimitador:

```
> write.table(dados, file="iris.csv", row.names=F, sep=",", quote=F)
```

O arquivo "iris.csv" pode ser aberto diretamente no Excel, simplesmente dando um duplo clique sobre o arquivo. Onde foi salvo este arquivo? No mesmo lugar que o anterior.

Mas e se você quiser salvar em um diretório específico? Bom, nesse caso, o argumento deve conter o caminho completo do diretório, seguido do nome do arquivo. Por exemplo, digamos que eu queira salvar esse arquivo no diretório "temp" que está na raiz do *drive* C da minha máquina. O código seria:

```
write.table(dados, file="c:/temp/iris.csv", row.names=F, sep=",", quote=F)
```

Observe que aqui utilizamos a barra para frente em vez das duas barras invertidas. O código abaixo teria o mesmo efeito:

```
write.table(dados, file="c:\\temp\\iris.csv", row.names=F, sep=",",
quote=F)
```

Entrada de Dados

O R é capaz de ler arquivos de dados salvos em diversos formatos diferentes, como ASCII (arquivo texto, delimitado por espaço, tabulação, vírgula, ponto-e-vírgula, entre outros), Excel,

SPSS, EpiInfo, etc. No caso do Excel, como veremos adiante, o processo de importação não é direto, tendo que se salvar inicialmente em um formato ".csv".

A função mais simples que lê dados externos no R faz parte do pacote básico e se chama `read.table()` e também pertence ao pacote `base`. Essa função vai importar dados em formato ASCII para um objeto do tipo *dataframe*. Vamos ver um exemplo, usando a base de dados que nós acabamos de exportar (note que o arquivo "iris.dat" deve estar presente no seu diretório de trabalho):

```
dados1 <- read.table(file="iris.dat", header=T, sep="\t")
```

Vamos ver por partes o que nós fizemos: criamos um objeto `dados1`, para não mexer no objeto `dados` que nós criamos na seção anterior, através do *assignment* `<-` pela função `read.table()`, que tem um argumento obrigatório (`file`) e dois argumentos opcionais (`header=T` e `sep="\t"`).

Bom, `file` é simplesmente o nome do arquivo externo onde os dados estão armazenados, com a extensão inclusive e entre aspas. O argumento `header=T` é para indicar para o R que a primeira linha contém o nome das variáveis nesse banco. O último argumento que usamos, `sep="\t"` é usado para indicar que a delimitação dos dados nesse caso é feita por tabulação. Uma importante observação é que se o separador fosse espaço e não tabulação, a função leria os dados da mesma forma, apenas mudando-se para `sep="\t"`. Que tal dar uma olhadinha na ajuda dessa função? Boa sorte...

Bom, a primeira providência após importar os dados é dar uma olhadinha neles para ver se tudo funcionou a contento. Se a gente quisesse ver esse banco inteiro, bastaria digitar o nome do objeto `dados` (como você deve se lembrar que acontece com qualquer objeto). Como esse banco tem 150 registros e 5 variáveis, que tal a gente ver somente os 10 primeiros registros? Vamos tentar:

```
> dados1[1:10,]
   slength swidth plength pwidth species
1      5.1    3.5     1.4    0.2  setosa
2      4.9    3.0     1.4    0.2  setosa
3      4.7    3.2     1.3    0.2  setosa
4      4.6    3.1     1.5    0.2  setosa
5      5.0    3.6     1.4    0.2  setosa
6      5.4    3.9     1.7    0.4  setosa
7      4.6    3.4     1.4    0.3  setosa
8      5.0    3.4     1.5    0.2  setosa
9      4.4    2.9     1.4    0.2  setosa
10     4.9    3.1     1.5    0.1  setosa
```

Antes de passarmos para o próximo assunto, um detalhe que incomoda um pouco no R é a impossibilidade de (por enquanto, pelo menos) nomear variáveis usando o caracter *underscore* (_). Isso acontece, porque esse caracter, para o R, corresponde ao caracter de *assignment*. Embora isso esteja mudando no R (e ao que parece isso será abolido no futuro), ele ainda entende "_" como um *assignment* (muito embora na atual versão você receba uma advertência quando usa). Vamos ver como funciona:

```
> x _ c(1,2,3)
Warning message:
The use of _ is deprecated: you will be warned only once per session
> x
[1] 1 2 3
```

Muito embora o meu objeto `x` tenha recebido o vetor (1,2,3), o programa adverte que o uso do "_" não é recomendado...

Agora vamos passar para o caso de um arquivo em Excel, que você gostaria que fosse lido pelo R. Como foi mencionado anteriormente, o R não é capaz de ler um arquivo em formato ".xls" diretamente; é preciso salvar o arquivo antes em um formato ASCII. Vamos ver como funciona?

Não faz parte do escopo deste material ensinar como salvar um arquivo do Excel no formato separado por vírgulas (ou CSV, *comma-separated values* em inglês), que será facilmente lido pelo R.

Bem, mas como somos muito bonzinhos, vamos descrever rapidamente como funciona: uma vez que você tenha aberto o arquivo que deseja salvar no Excel, pressione a tecla "F12" no teclado, o que abrirá o menu "Salvar Como". Na parte inferior do menu na opção "salvar arquivo do tipo", selecione "CSV": é simples assim mesmo.

Porém, meus amigos, nem tudo são flores... Existe um detalhe muito importante que os usuários do Excel devem saber: se o seu Excel for em português, provavelmente o delimitador será um ponto-e-vírgula e não uma vírgula, como deveria ser; se for em inglês, o delimitador será o certo, ou seja, com vírgula mesmo. Isso vai fazer diferença porque é preciso especificar o tipo de separador se você usar a função `read.table()`.

Bem, mas nós já temos um arquivo salvo no formato CSV que é o "iris.csv", que é um arquivo ASCII delimitado por vírgulas, lembra?

```
dados2 <- read.table(file="iris.csv", header=T, sep=",")
```

Acontece que o formato CSV é tão comum, que existem duas funções para ler esses arquivos diretamente; uma para aqueles separados por vírgulas e outro para aqueles separados por ponto-e-vírgula. Confira o usado para vírgula:

```
dados3 <- read.csv("iris.csv")
```

A função é basicamente a mesma, apenas com o *default* já trocado para o formato CSV. A outra função é `read.csv2()`. Consulte a ajuda sobre esta função para outras surpresas...

Vamos agora passar para os casos onde a leitura pode ser feita diretamente de um formato que não seja ASCII. Nesta situação é necessário carregar um pacote específico para importação e exportação de dados chamado `foreign`. Se você ainda não o fez, agora é a hora. Para isto digite:

```
library(foreign)
```

Vamos começar importando dados armazenados em formato ".sav", que é a extensão de banco de dados do SPSS. Aqui é também a primeira vez que você vai precisar de um arquivo externo. Nesse caso é o "iris.sav". Nós fizemos o seguinte: pegamos aquele arquivo que você exportou "iris.dat", importamos para o SPSS e fizemos algumas modificações nele para mostrarmos algumas outras características da função que vamos usar. Vamos ler então esse arquivo no R e vamos admitir nesse caso que o arquivo "iris.sav" foi salvo por você no seu diretório de trabalho:

```
dados4 <- read.spss(file= "iris.sav", use.value.label=F,  to.data.frame=T)
```

Vamos dar uma olhadinha nos primeiros 10 registros do nosso objeto:

```
> dados4[1:10,]
   SEPAL.LE SEPAL.WI PETAL.LE PETAL.WI SPECIES
1       5.1      3.5      1.4      0.2       1
2       4.9      3.0      1.4      0.2       1
3       4.7      3.2      1.3      0.2       1
4       4.6      3.1      1.5      0.2       1
5       5.0      3.6      1.4      0.2       1
6       5.4      3.9      1.7      0.4       1
7       4.6      3.4      1.4      0.3       1
8       5.0      3.4      1.5      0.2       1
```

```
9        4.4       2.9       1.4       0.2        1
10       4.9       3.1       1.5       0.1        1
```

Vamos fazer algumas observações. Primeiro, o nome das variáveis está diferente do que eram anteriormente: não só estão todos em maiúsculas, como parecem ter sido truncados para 8 espaços apenas. Bem, o SPSS não aceita nomes longos (com mais de 8 espaços) e trunca o nome mesmo. Além disso ele também converteu tudo para maiúsculo.

A outra diferença é que a variável `SPECIES` agora é numérica e não mais contém o nome das espécies. Essa foi a modificação que fizemos de propósito: convertemos os nomes para números: setosa para 1, versicolor para 2 e virgínica para 3. Depois colocamos os nomes como *labels* no SPSS (usuários deste programa sabem o que isso significa – espero...)

Vamos agora estudar os argumentos usados na função `read.spss()`: o primeiro não há dúvida e é o nome do arquivo; o segundo diz para o R não usar os *labels* das variáveis que eventualmente existam no SPSS ao importar as variáveis – é por isso que neste caso os números foram importados para a variável `SPECIES` e não os nomes das espécies. Por último, o argumento `to.data.frame=T`, indica para o R que o formato a ser lido é um *data frame*, visto que o *default* é importar para uma lista. Para importar os nomes das espécies, faça:

```
> dados4 <- read.spss(file= "iris.sav", use.value.label=T,
to.data.frame=T)
> dados4[1:10,]
   SEPAL.LE SEPAL.WI PETAL.LE PETAL.WI SPECIES
1       5.1      3.5      1.4      0.2  setosa
2       4.9      3.0      1.4      0.2  setosa
3       4.7      3.2      1.3      0.2  setosa
4       4.6      3.1      1.5      0.2  setosa
5       5.0      3.6      1.4      0.2  setosa
6       5.4      3.9      1.7      0.4  setosa
7       4.6      3.4      1.4      0.3  setosa
8       5.0      3.4      1.5      0.2  setosa
9       4.4      2.9      1.4      0.2  setosa
10      4.9      3.1      1.5      0.1  setosa
```

Quer saber mais detalhes sobre esta função? Consulte a ajuda...

Vamos agora ver um outro formato que é bastante usado em saúde pública, que é o ".rec", extensão usada por bancos armazenados pelo EpiInfo. Se você já usou o EpiInfo, deve conhecer o famosíssimo arquivo "oswego.rec", usado em muitos exercícios em epidemiologia. Para maiores informações sobre o arquivo, consulte o manual do EpiInfo, mas consiste de dados sobre uma intoxicação alimentar ocorrida em Oswego, NY em 1972 após um almoço beneficente em uma igreja. Nesse caso, podemos acessar o arquivo diretamente do diretório do próprio EpiInfo:

```
oswego <- read.epiinfo("c:\\epi6\\oswego.rec")
```

Se você não tiver o EpiInfo instalado, copie o arquivo "oswego.rec" para o diretório de trabalho e faça:

```
oswego <- read.epiinfo("oswego.rec")
```

Se você tem familiaridade com o EpiInfo, deve se lembrar que antigamente as datas eram armazenadas com os anos em 2 dígitos. No caso desse arquivo, na verdade, o ano foi omitido, já que tudo se passou em apenas dois dias. O problema é que nesses formatos, o R não vai entender esse campo como data, mas como texto. Vamos tentar o seguinte:

```
oswego$ONSETDATE
```

Note que a saída são as “datas” entre aspas, caracterizando uma variável tipo caracter. Duvida? Então faça:

```
is.character(oswego$ONSETDATE)
```

A resposta foi TRUE, como esperado. Para contornar esse problema, temos que lançar mão de alguns argumentos extras. Vamos tentar o seguinte:

```
oswego1 <- read.epiinfo("c:\\epi6\\oswego.rec", guess.broken.dates=TRUE,
thisyear="1972")
```

Bem, você deve ter entendido os dois argumetos extras: o primeiro pede para o R tentar acertar o ano da data que tenha um ano com 2 dígitos ou que não tenha ano algum (o nosso caso aqui). O segundo é para o R colocar um ano quando ele não estiver especificado. Como nós sabemos que esse surto ocorreu em 1972, esse foi o nosso argumento. Veja que agora temos datas de fato:

```
oswego1[1:10,1:5]
```

Além desses formatos, o pacote `foreign` também possui funções para ler arquivos em outros formatos como S-Plus, SAS, Minitab e Stata. Fique à vontade para explorar essas funções se for do seu interesse...

Saída de Resultados

Uma outra necessidade é a saída de resultados de funções usadas no R. Essas saídas são basicamente de dois tipos: Ou um gráfico ou uma saída de texto, que pode conter uma “tabela” ou resultados na forma de texto livre mesmo. Se você já tem alguma familiaridade com o R, deve saber que sua parte gráfica é bastante superior à sua parte de saída tipo tabela. Na verdade essas saídas só podem ser coladas como texto e não têm a mesma facilidade de manuseio como as tabelas avançadas do SPSS por exemplo (usuários do SPSS devem se lembrar que as saídas de uma regressão logística não têm essas facilidades também.)

Vamos então usar o mesmo exemplo visto no final do módulo básico com o nosso banco `iris` para mostrar como copiar e colar a saída de uma função e também um gráfico do R em um outro programa, como um editor de texto avançado (tipo Word ou Write) ou até mesmo para uma planilha eletrônica qualquer (como Excel, Lotus ou Calc.)

Primeiramente vamos reiniciar o objeto `dados` dados, evitando acidentes:

```
detach(dados)
data(iris)
dados <- iris
attach(dados)
```

Recordando os nomes das variáveis desse banco:

```
> names(dados)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

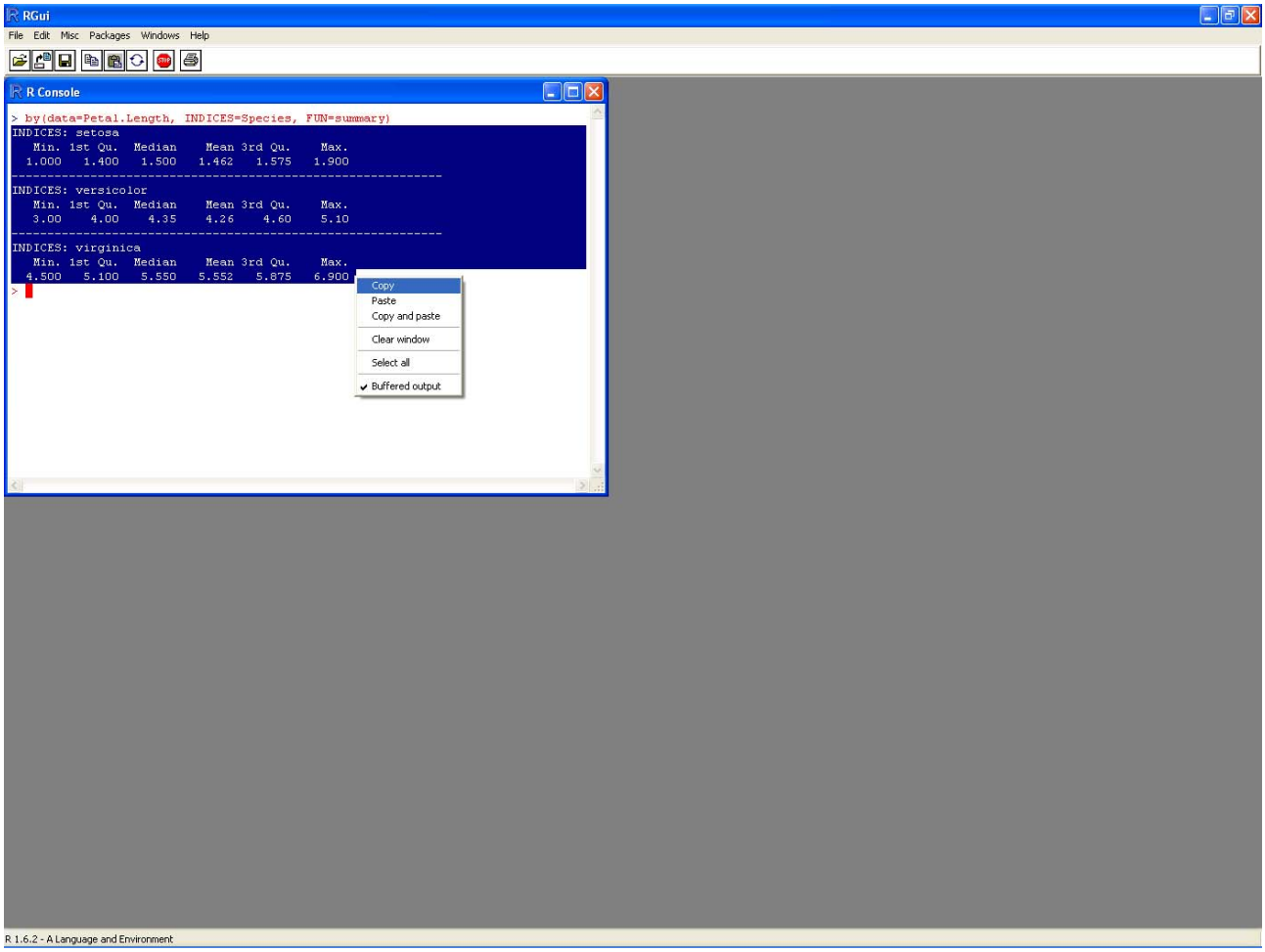
Você deve estar se lembrando da função utilizada no módulo básico chamada `by()`. Bem, se você não fez o módulo básico é porque já sabe um pouco de R, então consulte a ajuda... Vamos comparar os tamanhos de pétalas das flores por espécie:

```
by(data=Petal.Length, INDICES=Species, FUN=summary)
```

Temos aqui uma saída do tipo texto.

O primeiro passo para copiar e colar a saída dessa função é selecionar no R a saída ou parte da saída que seja de interesse. Para isso apertamos o botão esquerdo do *mouse* e o arrastamos sobre o texto. O tom azul sobre a tela indica a parte que está sendo selecionada.

O próximo passo é copiar a parte do texto selecionada no R. Isso pode ser feito selecionando no menu "Edit" a opção "Copy", ou simplesmente apertando sucessivamente a tecla "Ctrl" e a tecla "c". Outra possibilidade ainda é após selecionar o texto no R clicar com o botão direito do *mouse* e escolher "Copy":



Agora, é só colar o conteúdo no Word ou programa equivalente. Em qualquer editor de texto no Windows, a forma de se fazer isso é a mesma utilizada acima no processo de copiar os dados do R bastando trocar a opção "Copy" pela opção "Colar" (ou "Paste"). Para obter uma formatação no Word igual ao que aparece no R troque a fonte do texto colado para "courier new".

Uma outra necessidade de exportação de dados pode ser quando se deseja exportar uma tabulação qualquer (for exemplo uma tabela de freqüências, com uma ou mais variáveis) para um programa de planilha eletrônica. Bem, nós podemo usar neste caso a nossa já conhecida função `write.table()`, pois ela funciona também com outros tipos de objetos fora bancos de dados.

Para dar um exemplo, vamos dizer que desejamos fazer uma tabulação das espécies segundo o critério delas terem pétalas maiores ou menores do que a mediana do grupo inteiro. Inicialmente, vamos descobrir essa mediana:

```
> median(Petal.Length)
[1] 4.35
```

Muito bem, vamos então criar um vetor para agrupar as espécies segundo essa mediana (não se preocupe em entender a sintaxe por enquanto):

```
> g.petala <- ifelse(Petal.Length<4.35,"<4.35",">=4.35")
> table(Species, g.petala)
            g.petala
Species      <4.35 >=4.35
  setosa         50      0
  versicolor     25     25
  virginica       0     50
```

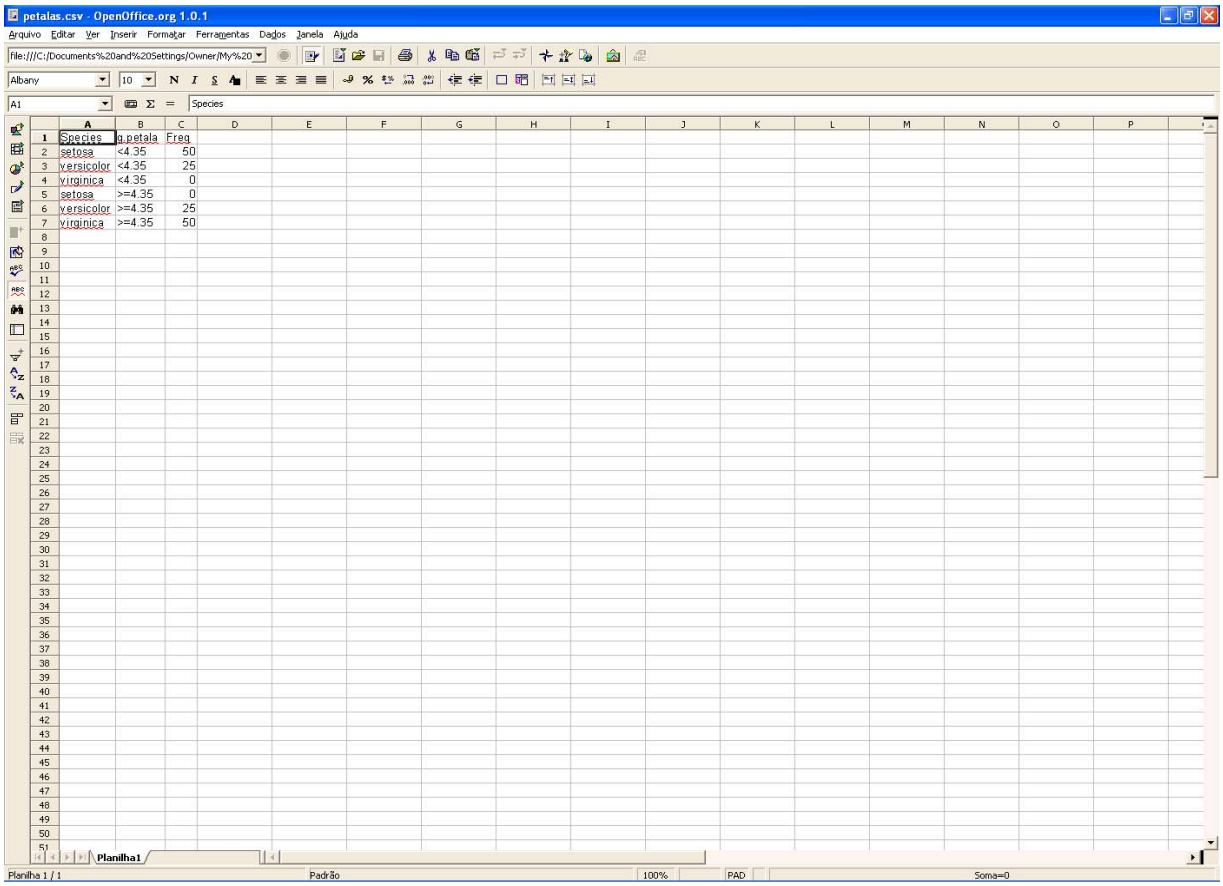
No primeiro comando, criamos o objeto `g.petala` que contém os agrupamentos, depois fizemos uma tabela para comparar as espécies com os grupos. Podemos guardar essa tabela em um objeto se quisermos. Faça assim:

```
g.petala <- ifelse(Petal.Length<4.35,"<4.35",">=4.35")
output <- table(Species, g.petala)
```

Confira o conteúdo do objeto `output`. Deverá ser a mesma tabela acima. Agora podemos usar a nossa velha amiga;

```
write.table(output, file="petalas.csv", row.names=F, sep=",", quote=F)
```

A única diferença é que o formato da tabela exportada é um pouco diferente (será uma tabela de freqüências por grupos em vez de uma tabela 3x2). Veja um exemplo após a importação para o Calc:

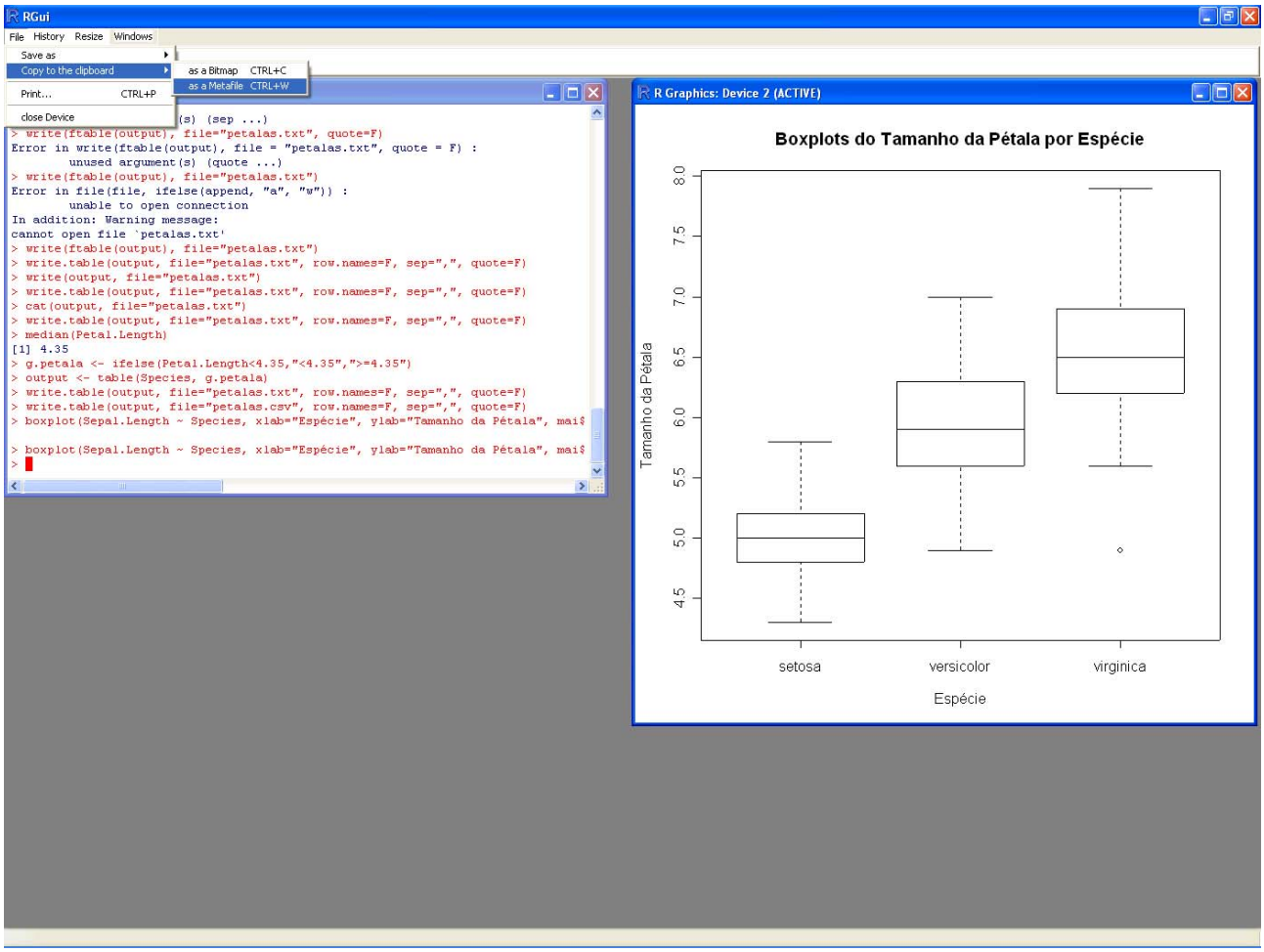


No caso da saída de uma função ser um gráfico, o procedimento de copiar e colar é outro. Vamos fazer boxplots para comparar os tamanhos das pétalas por espécie.

```
> boxplot(Sepal.Length ~ Species, xlab="Espécie", ylab="Tamanho da
Pétala", main="Boxplots do Tamanho da Pétala por Espécie")
```

Um arquivo gráfico pode ser copiado pelo R através de de dois formatos: Bitmap ou "metafile". Esses formatos se diferem entre si por vantagens e desvantagens que cada um possui. Uma dentre outras vantagens do formato "metafile" é o fato dele gerar arquivos de tamanho menor.

Para copiar um arquivo no formato "metafile", primeiro se certifique que a janela gráfica está selecionada (para tanto, apenas clique em qualquer área dentro da janela), e então escolha no menu "File" dentro de "Copy to the Clipboard" a opção "as Metafile":



Depois é só colar no seu editor de texto preferido, ou até mesmo na sua planilha. Colando no formato "metafile" podemos editar a figura através de um duplo clique sobre a figura permitindo alterar o título do gráfico, valores que aparecem nos eixos, etc, se você tiver um programa que permita editar "metafiles"

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Este é um módulo especial da série de módulos destinados ao aprendizado do ambiente R. A abrangência deste módulo é o conteúdo a ser dado em um curso básico de Bioestatística para alunos de pós-graduação, ministrado em qualquer instituição de ensino superior.

O objetivo desse módulo não é tornar o leitor um especialista em R, nem sequer um utilizador freqüente do programa, mas sim usar o R como uma ferramenta de ensino da Bioestatística. Alunos que já conhecem o R podem usar esse módulo sem recorrer a nenhum outro módulo básico sobre o R, mas para aqueles que nunca trabalharam com ele, é fundamental a leitura dos 3 módulos básicos contidos no material "Aprendendo R" e também a leitura prévia do módulo "Manuseando um Banco de Dados no R"

Este módulo é baseado no livro *Introductory Statistics with R*, escrito pelo professor Peter Dalgaard, que é um dos membros do *core* de desenvolvimento do R. A adoção deste livro como referência leva em conta o fato dele ter quase todos os seus exemplos em bioestatística (que é a área pela qual eu me interesso) e de ter disponível no próprio CRAN uma biblioteca com todos os bancos de dados necessários, chamada `ISwR`.

Os tópicos abordados nesse instrutivo acompanham aqueles que serão abordados nas aulas teóricas do curso de Estatística I, a saber:

- Probabilidade e distribuições
- Estatística descritiva
- Distribuições amostrais
- Teste de Hipóteses, poder e tamanho da amostra
- Testes para uma e duas amostras (contínuos)
- Proporções
- ANOVA
- Regressão e correlação

A programação de um curso como este é para ser dado em 8 aulas, cada uma abordando um destes temas citados, fora as aulas para o aprendizado do R. Sendo assim, vamos dividir esses tópicos em aulas não necessariamente do mesmo tamanho,como vocês poderão perceber no calendário do curso. Cada aula terá o tempo de 3 a 4 horas e meia, dependendo dos conteúdos a serem estudados. Esse é o tempo médio para que o aluno possa ler e treinar cada uma dessas aulas. Claro que isso é uma média mesmo, e algumas pessoas necessitarão de mais dedicação para acompanhar o ritmo das aulas.

Nem todas as aulas necessitarão da biblioteca , mas sempre que isso acontecer, será indicado no início da aula. Nem todas as aulas também estão contempladas no nosso livro de referência, ma sempre que estiverem, as páginas correspondentes estarão também indicadas.

Convenções e dicas

Se você já leu os outros tutoriais deste material, já deve estar acostumado com as convenções (que não são muitas) usadas neste material. Sempre que estivermos nos referindo a um código ou saída de texto do R, usaremos a fonte `Courier New tamanho 10` como em:

```
choose(1000, 30)
```

Existem dois tipos básicos de apresentação dos comandos. Esse acima, que não tem nenhuma saída de texto abaixo e também não é precedido do símbolo do *prompt* do R ">", será geralmente para ser copiado e colado no *prompt* do R para se obter a saída. Já os do tipo:

```
> mean(notas)
[1] 7.01
```

São em geral apenas para a observação dos resultados (claro que isso não impede que o comando seja usado, mas só tome cuidado para não colar o símbolo ">" junto, porque neste caso vai dar erro no R).

Existe ainda mais um tipo, que é igual ao primeiro, no sentido que é para ser copiado e colado também, mas de uma maneira mais complexa. São as funções. Elas têm esse jeitão:

```
var.pop <- function (x)
{
    sum((x-mean(x))^2)/length(x)
}
```

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: Nenhuma

Aula 1 – Probabilidade e distribuições
Livro: páginas 45 a 55

Essa aula na verdade conterá muita recordação do módulo básico onde falamos sobre distribuições e gráficos. Ficarei à vontade inclusive para copiar algumas partes do texto já escrito. O motivo dessa repetição é não só aprofundar um pouco essas noções que foram passadas, mas também permitir a introdução ao assunto para aqueles que se sentiram capazes de não estudar o módulo básico.

Probabilidade
A função `sample()`
Distribuições Discretas
Distribuições Contínuas
Distribuições no R
- Densidade
- Acumulativa
- Quantis
- Números pseudo-aleatórios

Exercícios

Probabilidade

Inicialmente, vamos falar de probabilidades e amostras aleatórias. Como você já deve ter notado, a inferência estatística é baseada sempre em uma amostra aleatória que é tirada de uma determinada população sobre a qual gostaríamos de inferir certas "características", como a sua média ou a sua variância, por exemplo.

A idéia é sempre a mesma das bolinhas numeradas em uma urna. Se nós tivermos por exemplo 10 bolinhas numeradas de 1 a 10 em uma urna, qual é a chance de nós escolhermos exatamente, por exemplo a bolinha com o número 5? Intuitivamente você vai responder que essa chance é de 1 em 10, 1/10 ou 10%. Nesse caso o que acontece é que as bolinhas têm uma chance igual de serem escolhidas, certo? Pois se eu perguntasse sobre a chance da bolinha com o número 3, a resposta seria a mesma.

Agora, e se eu perguntasse qual seria a probabilidade de se retirar a bolinha com o número 5 e a bolinha com o número 3? Nesse caso, você tem que me fazer duas perguntas pelo menos. A primeira é se a *ordem* de retirada deve ser levada em conta ou não (ou seja, se sortear a bolinha 5 e em seguida a 3 deve ser diferente de sortear a bolinha 3 e em seguida a 5, ou não). A segunda pergunta é se uma vez sorteada uma bolinha ela deve ser devolvida à urna ou não para ser sorteada a segunda bolinha. Essas perguntas são fundamentais porque elas vão alterar significativamente a resposta. Muito bem, vamos assumir a situação mais comum, que é quando a ordem não importa e calcular as probabilidades com e sem reposição.

Para o caso com reposição, nós teremos a probabilidade de sortear a bolinha 3 (0.1) *e* a bolinha 5 (0.1) *ou* a bolinha 5 (0.1) *e* a bolinha 3 (0.1). Em probabilidade existe um macete interessante: quando falamos da probabilidade de acontecer um evento *e* outro evento, estamos

falando de uma multiplicação. Se estamos falando da probabilidade de acontecer um evento *ou* outro evento, estamos falando de uma soma. Pescou? Pois é, essa probabilidade vai ser (0.1x0.1) + (0.1x0.1) = 0.02.

Já para o caso sem reposição, a cosa muda um pouco de figura. Agora, a probabilidade do segundo elemento a ser sorteado vai ser diferente da do primeiro, pois vai ter uma bolinha a menos na minha urna. Então, nesse caso, apesar do raciocínio ser o mesmo, os números mudam. Olha só, eu vou copiar a mesma frase lá de cima, só mudando as probabilidades: Para o caso sem reposição, nós teremos a probabilidade de sortear a bolinha 3 (1/10) *e* a bolinha 5 (1/9) *ou* a bolinha 5 (1/10) *e* a bolinha 3 (1/9). Então, essa probabilidade vai ser (1/10x1/9) + (1/10x1/9) = 0.0222.

Como você percebeu, a probabilidade no segundo caso é maior que no primeiro, já que a probabilidade de se sortear uma determinada bolinha na segunda tentativa é maior (o denominador é menor) que na primeira.

Agora um último problema antes de nós partirmos para a nossa parte prática de fato. Todos esses exemplos são compostos de bolinhas que têm uma probabilidade igual de ser sorteada. Mas isso não necessariamente acontece assim. Vamos supor que por exemplo nós só tivéssemos bolinhas com o número 1 e com o número 2, mas que temos 4 com o número 1 e 6 com o número 2. Bem, agora qual seria a probabilidade de se escolher uma bolinha com o número 1? A resposta também é intuitiva, e será 4/10 = 40% – o número de bolinhas com o número 1 dividido pelo total de bolinhas na urna... E da bolinha com o número 2? Nesse caso, seria 6/10 = 60%. Neste caso, as bolinhas têm uma probabilidade diferente de serem sorteadas. O caso de uma segunda bolinha, vamos deixar para um desafio para vocês que é um problema um pouco mais complicado.

Muito bem, esse papo todo é só para refrescar a sua memória sobre probabilidade, porque a nossa intenção aqui é ver coisas acontecendo na prática.

A função `sample()`

Tudo muito bonito esse papo de amostra aleatória e probabilidade, mas como o R pode nos ajudar com isso? Bem, o R possui uma função bastante interessante, que é a função `sample()` que para quem conhece essa palavra em inglês já deduziu que serve para amostrar alguma coisa. E é isso mesmo: ela serve para criar uma amostra aleatória de um vetor qualquer, com ou sem reposição e com probabilidades iguais ou não. Vamos ver então como essa função funciona e como ela vai nos ajudar a entender melhor esses problemas de probabilidade.

Primeiro, vamos simular a situação onde temos a urna com as bolinhas numeradas de 1 a 10. A maneira mais simples é criar um vetor chamado urna, com valores de 1 a 10:

```
urna <- 1:10
```

Agora, nós poderíamos pedir para a função sortear uma bolinha pra a gente, assim:

```
sample(urna, 1)
```

Experimente também tirar 2 amostras (ou seja, duas bolinhas) da nossa urna:

```
sample(urna, 1)
```

Faça várias vezes e veja o que acontece...

Bem, na verdade fica meio difícil ver o que realmente acontece quando essa função trabalha, não é mesmo? O programa está simplesmente atribuindo uma probabilidade igual a cada um dos elementos e retornando um ou dois deles. Bem, mas quando dizemos que a probabilidade de se sortear a bolinha *x*, o que realmente queremos dizer com isso?

Queremos dizer que se nós repetirmos este experimento (retirar uma bolinha) *n* vezes, onde *n* é um número grande (diz-se inclusive que tende para o infinito – grande para chuchu), então em média, a bolinha de número *x* será sorteada em uma fração *p* das vezes. Por que não tentamos então fazer isso? Vamos criar uma função para tirar várias amostras de tamanho 1 e guardá-las em um

vetor. Em seguida, vamos ver quantas vezes uma determinada bolinha (número) aparece no nosso vetor, dividindo este número pelo tamanho do vetor. Complicou? Vamos tentar na prática, então:

```
      probab <- function (x, size=length(bolinha), repos = FALSE, prob = NULL,
times=10000, bolinha=1, order=F)
      {
            el <- bolinha
            pr <- 0
            z<-0

            if(order){
                  for (i in 1:times)
            {
                  z <- sample(x, size=size, replace = repos, prob=prob)
                  if (sum(el==z)/length(el)==1)
                  {
                        pr[i] <- 1
                  }else{
                        pr[i] <- 0
                  }
            }

            }else{

            for (i in 1:times)
            {
                  z <- sample(x, size=size, replace = repos, prob=prob)
                  if (sum(el%in%z)/length(el)==1)
                  {
                        pr[i] <- 1
                  }else{
                        pr[i] <- 0
                  }
            }
      }
            sum(pr)/times
      }
```

O que estamos fazendo é apenas tirar uma amostra 10000 vezes e comparando com elementos que nós determinamos para ver qual a porcentagem (ou probabilidade) de se obter aquela(s) bolinha(s) indicada(s), quando a ordem não importa (por *default*). Vamos ver como isso funciona.

Vamos primeiro ver qual a probabilidade de se escolher a bolinha número 3 (a essa altura você já notou que para uma bolinha só não faz diferença se é com ou sem reposição, né? Para a bolinha número 3, teríamos:

```
> probab(urna, bolinha=3)
[1] 0.1017
```

Bastante próximo do que tínhamos calculado anteriormente, não? Bem, e agora para as bolinhas 3 e 5, como ficaria? Nesse caso depende se é com ou sem reposição. No caso sem reposição, a probabilidade deve ser em torno de 0.022. Vamos conferir:

```
> probab(urna, repos=F, bolinha=c(3,5))
[1] 0.0229
```

Repare que tivemos que usar a função `c()` para escolhermos as bolinhas 3 e 5. E agora no caso da com reposição? Esperamos uma probabilidade menor, em torno de 0.2. Vamos ver como funciona:

```
> probab(urna, repos=T, bolinha=c(3,5))
[1] 0.0206
```

Muito bem. Mas lembre-se que quando você for fazer isso no R, os resultados de cada uma das suas tentativas vai ser diferente do que eu obtive quando fiz, pois a função `sample()` escolhe as bolinhas aleatoriamente...

Repare também que esta função serve para calcular as probabilidades aproximadas quando a ordem de retirada importa, com o argumento `order=T`, se for necessário. Vamos ver só um exemplo rápido:

```
> probab(urna, repos=T, order=T, bolinha=c(3,5))
[1] 0.0099
```

Um resultado esperado, já que se a ordem importa, não nos interessa a amostra (5,3), mas apenas a amostra (3,5), e claro, a probabilidade deve ser aproximadamente a metade da anterior.

Distribuições Discretas

Como você já deve ter aprendido, as distribuições de fenômenos naturais dos quais queremos fazer inferências a respeito se dividem em dois grandes grupos: variáveis discretas e variáveis contínuas. Vamos começar pelas discretas. Elas são empregadas para descrever fenômenos que só podem assumir números inteiros. Um exemplo muito freqüente em epidemiologia é o número de pacientes com uma determinada doença em uma população. É claro que não pode existir um "meio" paciente, e portanto a distribuição deste evento é uma distribuição discreta.

Note que esta classificação é muito geral e nada é dito ou suposto sobre o *range* da distribuição, e nem ao seu formato. Essas características serão particulares de cada distribuição. A única característica que as une é o fato de assumirem números inteiros. Sendo discreta, podemos definir a probabilidade de ocorrência de um determinado evento ocorrer sem muita dificuldade:

$$P(X=x)=f(x)$$

Claro que isso é uma definição muito geral. Para o nosso exemplo das bolinhas com número 1 e 2, com probabilidades diferentes, teríamos:

$$P(X=x)=f(x)=\left\{\begin{array}{c} 0.4,\ se\ \ x=1 \\ 0.6,\ se\ \ x=2 \\ 0,\ caso\,contrário \end{array}\right\}$$

Como seria o gráfico desta função? (é isso mesmo, ela tem um gráfico... meio esquisitão, é verdade...)

```
x<-0:4
y<-c(0,0.4,0.6,0,0)
plot(x,y, type="h")
points(1,0.4)
points(2,0.6)
```

Esta função é chamada função de densidade de probabilidade, ou fdp (no bom sentido, é claro.) É conhecida também, por causa do inglês, como pdf.

Existe um outro tipo de função que mede a probabilidade acumulada de eventos, ou seja, ela mede a probabilidade da ocorrência de eventos em sucessão, e estaríamos falando não da probabilidade de ocorrer um evento, mas da probabilidade de ocorrerem x ou menos eventos:

$$P(X\le x)=F(x)$$

Nessa função anterior por exemplo, teríamos:

$$P(X\le x)=F(x)=\left\{\begin{array}{c} 0\ se\ \ x<1 \\ 0.4,\ se\ \ 1\le x<2 \\ 1,\ se\ \ x\ge 2 \end{array}\right\}$$

Vamos ver o gráfico?

```
x<-0:4
y<-c(0,0.4,1,1,1)
plot(x,y, type="s")
```

Esta função é chamada de função de densidade acumulada, e às vezes também chamada apenas função de probabilidade.

É claro que existem várias funções discretas famosas, e uma das que iremos usar com bastante freqüência é a distribuição Binomial. Ela descreve o número de "sucessos" que ocorrem em um determinado número de experimentos. Os sucessos têm uma certa probabilidade *p* de acontecer e, junto com o numero de experimentos *n* compõem os parâmetros desta distribuição. A fdp de uma Binomial (*n,p*) é dada por:

$$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

Nesse caso o *k* é usado apenas para não haver confusão com o *x,* e ele representa um valor qualquer de *x.* O gráfico desta função pode ser facilmente obtido no R. Vamos ver um exemplo para uma Binomial(100, 0.05):

```
hist(rbinom(1000, prob=.05, size=100), freq=F)
```

Também é possível obter-se a função acumulada de distribuição da Binomial:

$$P(X\le k)=\sum_{x=0}^{k}\binom{n}{x}p^x(1-p)^{n-x}$$

E o seu gráfico, como o do nosso exemplo, pode também ser facilmente obtido:

```
x<-seq(0,20,1)
plot(x, pbinom(x, prob=.05, size=100), type="s")
```

Você deve ter notado que a *F(x)* da Binomial nada mais é do que um somatório das probabilidades individuais de cada ponto da *f(x)* até que um determinado valor *k* seja obtido. Nós só somamos todos os pontos dentro do *range* da distribuição para conferir se ela pode mesmo ser uma distribuição. Como você deve estar cansado de saber, essa soma tem que ser sempre 1.

Aliás, se você não se lembra, o *range* de uma função de distribuição corresponde ao que você deve ter aprendido como domínio da função, ou seja, os valores de *x* para os quais a função está definida. No caso da Binomial, o seu domínio é $x=0,1,...n$ e $0\le p\le 1$. Ambos os valores são intuitivos, já que o número de sucessos *x* de uma Binomial em *n* experimentos, só pode ir de zero a *n*. Já a probabilidade *p* como toda probabilidade, só pode variar entre zero e um.

Distribuições Contínuas

Existem também variáveis aleatórias que seguem uma natureza contínua, ou seja ela pode assumir qualquer valor real. Por exemplo, a natureza da distribuição das pressões arteriais de uma população qualquer é uma variável que pode assumir qualquer valor real positivo, muito embora para uma pessoa viva alguns limites devam ser respeitados. Nesse caso, esta variável é dita contínua.

Como as variáveis discretas, as contínuas também possuem fdp's e distribuições acumuladas. A grande diferença é que como estamos falando de um espaço contínuo, não é possível calcular a probabilidade de um determinado valor que *x* assuma, mas sim de um pequeno intervalo entre dois *x* consecutivos. Então, a nossa *f(x)* vai ser apenas uma função contínua. Um exemplo muito conhecido nosso é a distribuição Normal:

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Como sabemos, a Normal possui também dois parâmetros: a sua média μ e a sua variância σ^2 (muito embora seja comum usar o desvio-padrão σ como o segundo parâmetro – incluindo o R, como veremos mais adiante.) Como no caso da Binomial, esta função pode também ser facilmente colocada em um gráfico no R. Vamos usar a Normal *default* no R que é a Normal (0,1):

```
curve(dnorm(x), from=-3, to=3)
```

Claro que como no caso das distribuições discretas, as distribuições contínuas também possuem funções de densidade acumuladas e a idéia do somatório das probabilidades individuais de cada valor que *x* pode assumir para a sua obtenção também permanece. Só que como estamos falando de uma soma de uma função contínua, em vez de somatório, teremos que usar uma integral para calcular a sua *F(x)*. De modo geral:

$$F(x) = \int_{-\infty}^{x} f(x)\, dx$$

É claro que esse limite inferior de menos infinito vai depender do domínio da distribuição – algumas são definidas para certos intervalos, como a Uniforme (0,1) por exemplo, mas para a Normal, é exatamente assim.

Podemos também obter um gráfico da função de distribuição acumulada no R. Para a Normal (0,1), seria assim:

```
x<-seq(-3,3,0.01)
plot(x, pnorm(x), type="l")
```

Repare que agora o gráfico não assume mais o formato de degraus, mas sim de uma distribuição contínua.

Vamos falar agora rapidamente de um fantasma que acabou de aparecer aí em cima que é a integração de uma função. Muitas pessoas se assustam com a notação e têm dificuldade de entender o que isso significa. Para o nosso nível de aprendizado, cremos que pelo menos uma compreensão bastante simples e básica é necessária. A primeira noção já foi passada, de que ela é uma soma contínua (se contrapondo ao somatório quando estamos trabalhando com números inteiros.)

A segunda noção simples é uma interpretação geométrica que a integral pode ter. Ela representa na verdade apenas a área sobre a curva da função que está sendo integrada, dentro dos limites estabelecidos. Quando nós falarmos adiante das funções do R para calcular funções acumuladas, vamos ver uns exemplos sobre isto, mas é legal já ter esta noção em mente.

Distribuições no R

O R possui uma série de funções para calcular valores para essas distribuições que nós acabamos de apresentar. Muitas delas já foram até usadas sem explicação alguma (o que pode ter deixado você um pouco boiando) quando fizemos os gráficos da Binomial e da Normal. São quatro os tipos básicos dessas funções: funções para o cálculo de densidade, funções para o cálculo de densidade acumulada, funções para o cálculo de quantis e funções que geram distribuições. Cada uma delas estão disponíveis para diferentes tipos de distribuições, sejam discretas (e.g. Binomial, Geométrica) ou contínuas (e.g. Uniforme, Normal, Exponencial.) Vamos ver em mais detalhe como cada uma destas funções funcionam.

Densidade

Como já foi mencionado anteriormente, a densidade de uma distribuição (o a sua fdp) tem conotações diferentes, dependendo se a distribuição é discreta ou contínua. No primeiro caso, a densidade representa de fato uma probabilidade pontual, a probabilidade do evento *x* ocorrer. Já no caso de uma distribuição contínua, a probabilidade para uma valor específico de *x* é zero e a

densidade representa a probabilidade de se obter um valor na vizinhança do valor *x*, definido por um intervalo qualquer na sua vizinhança (de *x*). Este conceito pode ser um pouco confuso, pois como veremos existe um valor definido para a densidade de uma função como a Normal por exemplo, só que esse valor NÃO CORRESPONDE À PROBABILIDADE DESSE EVENTO OCORRER!!!

Dos quatro tipos de funções que estudaremos, essa é certamente a que menos se usa no dia a dia, mas é muito útil para a construção de exemplos (aliás, nós já usamos esta função algumas vezes.) Essa família de funções (chamo de família porque existem várias delas para diversas distribuições no R) sempre começam com a letra `d`, seguida de uma abreviação do nome da distribuição. Por exemplo, quando construímos o gráfico da fdp da Normal, nós usamos o comando:

```
curve(dnorm(x), from=-3, to=3)
```

Como se tratava da Normal, a abreviação é `norm`, e a função se chama então `dnorm()`.

Veja um resumo das abreviações usadas para algumas distribuições na tabela abaixo:

Tabela 1.1 – Abreviações e argumentos usados pelo R para gerar distribuições. Antes da abreviação, deve-se acrescentar a letra `d` para a fdp, `p` para a função de densidade acumulada, `q` para a função de quantis e `r` para geração aleatória de uma amostra. Os argumentos são descritos juntamente com os valores *default*, quando houver. Veja texto para detalhes.

Distribuição	*Abreviação*	*Argumentos com default*
Binomial	`binom`	`n= p=`
Geométrica	`geom`	`prob=`
Hipergeométrica	`hyper`	`m= n= k=`
Binomial negativa	`nbinom`	`size= prob=`
Poisson	`pois`	`lambda=`
Uniforme	`unif`	`min=0 max=1`
Normal	`norm`	`mean=0 sd=1`
Exponencial	`exp`	`rate=1`
Qui-Quadrada	`chisq`	`df=`
t de Student	`t`	`df=`
F de Snedcor	`f`	`df1= df2=`
Weibull	`weibull`	`shape= scale=1`
Gama	`gamma`	`shape= rate=1`
Beta	`beta`	`shape1= shape2=`

Repare que apesar de na tabela estar assinalado que as funções para a Normal possuem 2 argumentos, eles foram omitidos no código acima. Isso ocorreu porque algumas destas funções têm valores *default*, e no caso da Normal, como você já deve ter observado são `mean=0` e `sd=1`.

A função `curve()` vai desenhar um gráfico de uma função qualquer de *x*. Nesse caso a função é justamente a função `dnorm()`, que gera a *d*ensidade de uma *norm*al. Além disso, a função `curve()` também toma os argumentos `from` e `to`, para estabelecer os limites do gráfico.

Mas afinal de contas, que valor é retornado por esta função? É simplesmente o resultado da fdp no ponto *x*. Quer ver um exemplo? Vamos ver a fdp da Normal (0,1):

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Vamos agora no R calcular o valor dessa função quando *x = 0*:

```
> dnorm(0)
[1] 0.3989423
```

Lembre-se que esse valor, por se tratar de uma distribuição contínua, NÃO corresponde à probabilidade de se obter o valor *x = 0* em uma Normal (0,1)!!!

Bem, vamos fazer este cálculo na mão agora e conferir. Basta substituir por *0* onde existe *x* na equação acima:

$$f(0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{0^2}{2}} = \frac{1}{\sqrt{2\pi}} e^{-0} = \frac{1}{\sqrt{2\pi}} \simeq 0.3989$$

Função de distribuição acumulada

Já a função de distribuição acumulada é bastante usada, não em exemplos, como nós também vimos anteriormente, mas também para cálculos corriqueiramente utilizados em estatística.

Para a construção de um gráfico da função de distribuição acumulada da Normal, nós usamos o comando:

```
x<-seq(-3,3,0.01)
plot(x, pnorm(x), type="l")
```

Onde nós criamos um vetor `x` como uma seqüência de -3 a 3, de 0.01 a 0.01 e depois plotamos esse vetor contra os valores retornados pela função `pnorm()`. Como você deve ter percebido, para gerarmos densidades acumuladas, acrescentamos a letra `p` antes de uma das abreviações descritas na Tabela 1.1.

Aliás, é aqui que entra a nossa compreensão superficial sobre a integral de uma função. O que a `pnorm()` faz é calcular o resultado desta conta:

$$P(X \leq x) = F(x) = \int_{-\infty}^{x} f(x)\, dx$$

Onde a *f(x)* vai ser a fdp de alguma distribuição, neste caso, a Normal (0,1) e que nós já vimos que o resultado é:

$$P(X \leq x) = F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx$$

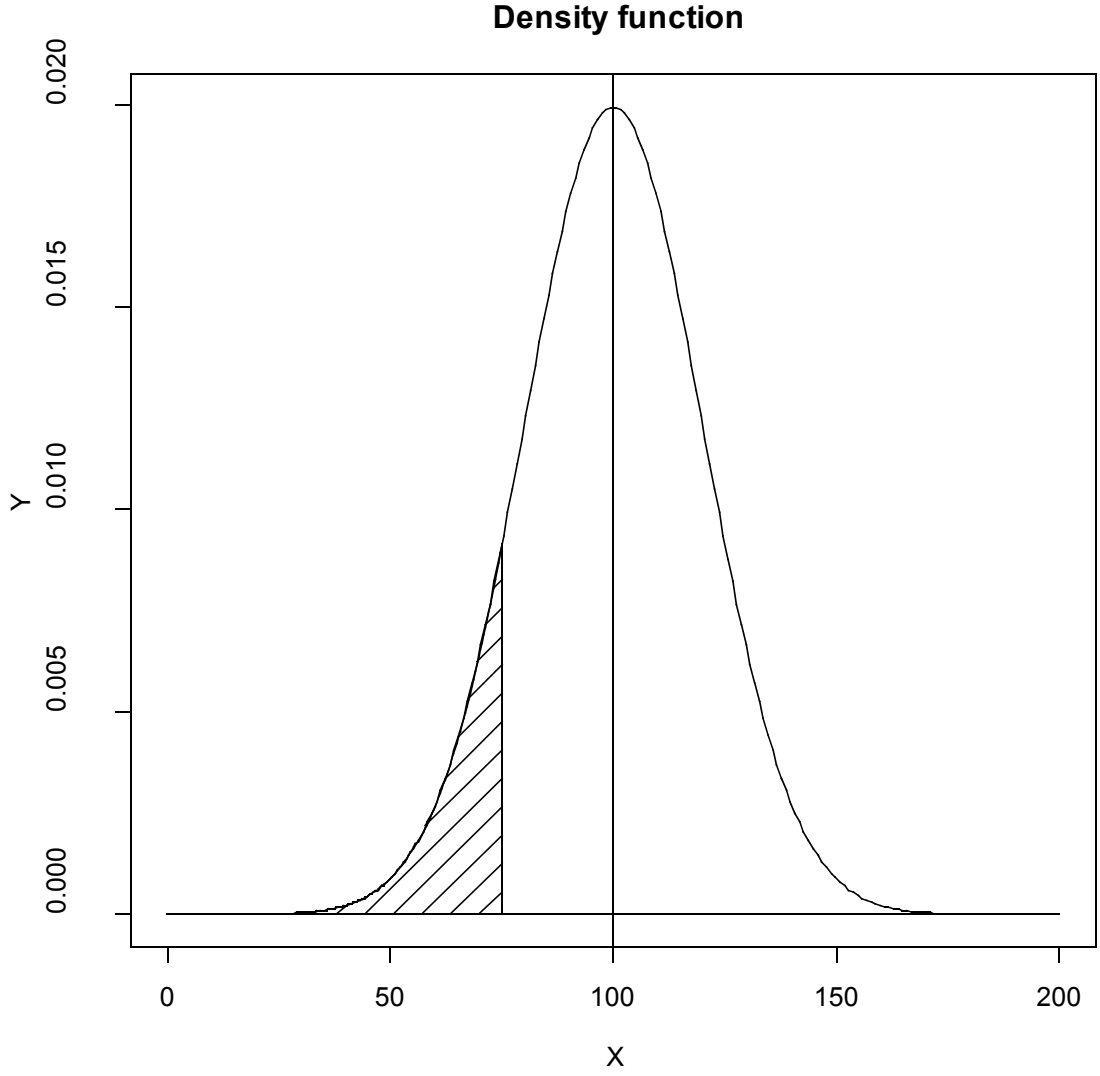
Agora você deve estar se perguntando: mas porque existem aquelas imensas tabelas para a distribuição Normal (0,1) que vêm nas costas de todo livro de estatística, se o valor da função de distribuição acumulada é apenas uma conta, uma função de *x* aplicada a um determinado valor? Nesse caso, a gente poderia calcular na mão, com fizemos com a fdp, não?

Bem... não. O problema é que a "conta" $\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx$ não tem resultado algébrico definido (ou seja, não é possível obter uma outra função a partir deste cálculo) e então os resultados têm que ser obtidos por cálculo numérico. Daí a grande importância da implementação de funções do tipo da `pnorm()`.

Além então dos gráficos que nós vimos, com estas funções é possível calcular uma série de coisas úteis para estatística, que substituem a consulta das tabelas. Digamos, por exemplo, que uma certa característica de uma população siga uma distribuição Normal, com média 100 e DP de 20. Uma pergunta pertinente sobre esta população seria: qual a percerntagem de pessoas nesta população que possuem um valor igual ou menor a 75? Para responder, podemos fazer:

```
> pnorm(75, mean=100, sd=20)
[1] 0.1056498
```

Ou seja, cerca de 10.56% das pessoas possuem um valor igual ou menor que 75. Mas afinal de contas, como podemos visualizar esses resultados? Como foi mencionado, esse valor que foi encontrado nada mais é do que a área abaixo da curva da fdp desta Normal (100, 400). Vamos ver como isso funciona, mas desta vez só mostrarei o resultado final (seria um pouco complicado pedir para você fazer esse gráfico, mas se estiver interessado, não se acanhe em entrar em contato.)



O que observamos na figura acima é a fdp desta normal, com a área hachurada de menos infinito (embora não dê pra visualizar muito bem) até o valor que queríamos de 75. Esta área vale exatamente o valor que nós achamos acima, ou seja, 0.1056. Para reforçar, essa conta para esta normal é dada por:

$$F(75) = \int_{-\infty}^{75} \frac{1}{\sqrt{2\pi} \times 20} \exp\left(-\frac{(x-100)^2}{2 \times 20^2} \right) dx$$

Significa que estou calculando a integral (que é a área sob a curva) de menos infinito até 75 da fdp da Normal (100, 400.) Note que eu apenas substituí os valores de μ e de σ^2 na equação acima.

Testes estatísticos também são uma aplicação direta dessas funções. Por exemplo, mais tarde você vai aprender a usar e interpretar o famoso teste *t* de Student. Sem entrar em detalhes, basicamente será calculada uma estatística *T*, a qual terá uma distribuição *t* com *n-1* graus de liberdade (onde *n* é o tamanho da amostra.) Para se calcular o famoso p-valor associado a esta

estatística, usamos a função `pt()`. Digamos que para uma amostra de 100 pacientes, a estatística que você calculou foi -2.55. Para calcular o p-valor:

```
> pt(-2.55, df=99)
[1] 0.006152768
```

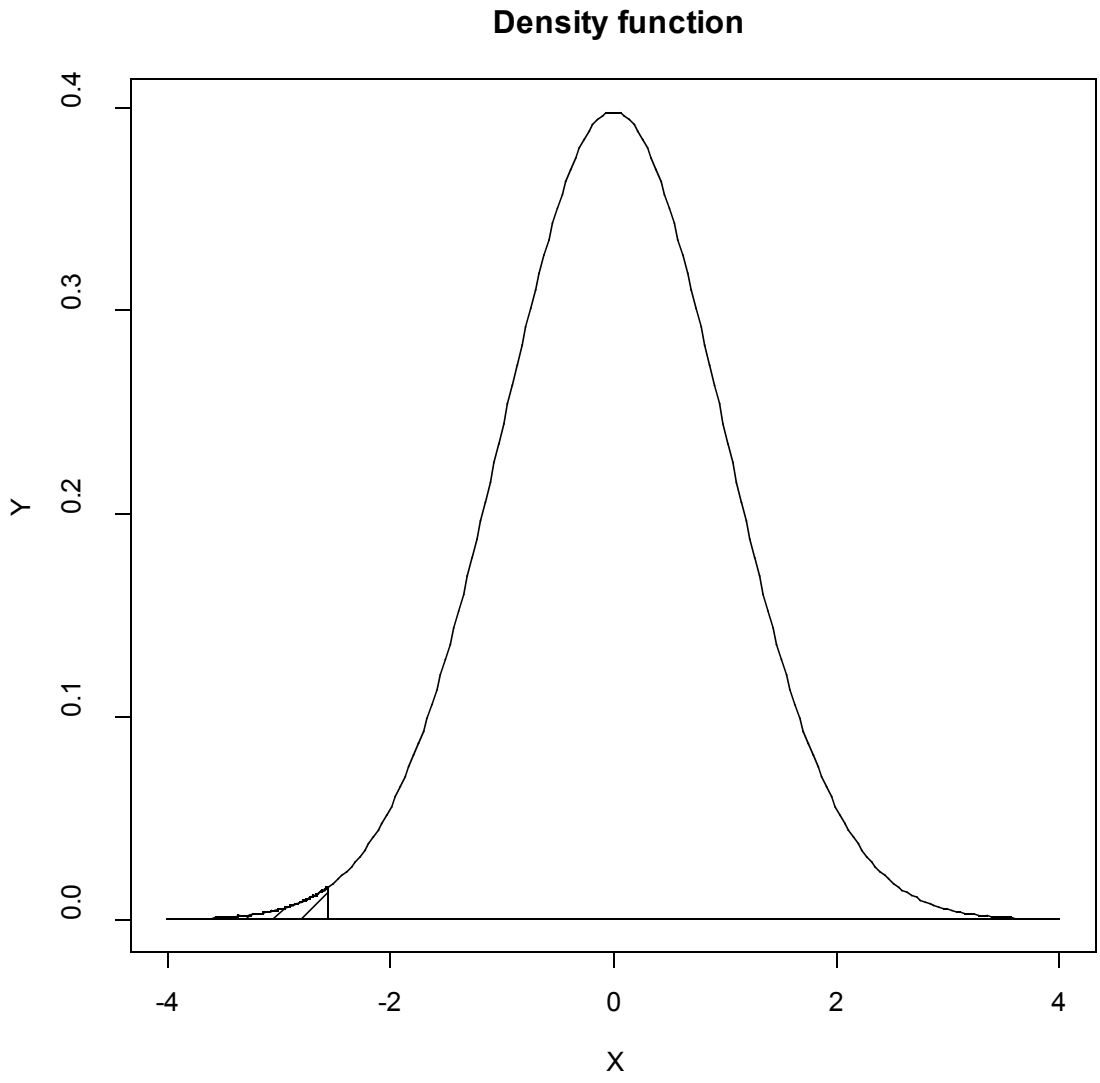
Como você aprenderá mais tarde, a distribuição *t* tem um único parâmetro, que são os graus de liberdade (*degrees of freedom* em inglês – daí o argumento `df`). Mas não se preocupe com nada disso agora, é apenas para já ter em mente a utilidade da função.

Novamente, esse valor é também uma área, mas sob a curva da fdp de uma distribuição *t* com 99 graus de liberdade, indo de menos infinito até -2.55, como mostrado na curva abaixo.

Só para fixar, sem querer complicar muito, vamos admitir que a distribuição t_{99} tenha uma fdp genérica, do tipo $f(t)$, usando *t* só para indicar que se trata de uma distribuição *t*. Então, a área assinalada abaixo corresponde à conta:

$$F(-2.55) = \int_{-\infty}^{-2.55} f(t)\, dt$$

Cuja solução, como você já deve ter adivinhado, também não tem um resultado definido algebricamente.



Quantis

A próxima família de funções são as que geram quantis. Se você está achando isso um palavrão, não posso culpá-lo. Essa função é na verdade apenas a função inversa da função de densidade acumulada. Assim, ela responde à seguinte pergunta: a que valor de *x* corresponde uma probabilidade acumulada de 0.975, por exemplo. Aliás, numa Normal (0,1), qual seria mesmo esse valor?

```
> qnorm(0.975, mean=0, sd=1)
[1] 1.959964
```

Isso mesmo, o nosso bom e velho 1.96...

A utilização dos quantis será bastante importante quando estudarmos também intervalos de confiança, pois como veremos, precisaremos do valor desta função para calcular quanto deve ser somado à média amostral por exemplo para obter-se o limite superior do seu intervalo de confiança (e subtrair-se também para obter-se o limite inferior.) Vamos ver um rápido exemplo: Por exemplo, um IC 95% para uma média amostral com variância da população desconhecida é dada por (não se assuste se você não entendeu o que isso quer dizer, você vai aprender mais tarde):

$$\bar{x} \pm t_{n-1,1-\alpha/2} \times \sqrt{s^2/n}$$

Onde $\bar{x}$ é a média da amostra, s^2 é a variância da amostra, *n* é o tamanho da amostra e $t_{n-1,1-\alpha/2}$ é exatamente o valor de *t* com *n-1* graus de liberdade no ponto $1-\alpha/2$, que para um nível de confiança de 5% $(\alpha = 0.05)$ corresponde a 0.975. Vamos chutar alguns valores e ver como isso funcionaria no R. Digamos que a média das diferenças de pressão arterial de uma amostra de 100 pacientes, antes e após um determinado tratamento, seja -10mmHg e que a variância da diferença tenha sido de, digamos 9mmHg^2. Nesse caso teríamos:

```
> -10-(qt(0.975, df=99)*sqrt(9/100))
[1] -10.59527
> -10+(qt(0.975, df=99)*sqrt(9/100))
[1] -9.404735
```

Nesse caso tivemos que usar a função `qt()` para calcular este IC 95%. Assim, nesse experimento a pressão arterial dos pacientes foi reduzida em média 10mmHg, com um IC95% de (-10.595,-9.405).

Números pseudo-aleatórios

Geralmente nos referimos à geração de números pelo computador como números aleatórios apenas. Essa noção é pouco intuitiva, porém, pois é de se esperar que um computador realize apenas cálculos e operações exatas, sem nenhum tipo de aleatoriedade envolvida. Essa dúvida procede, já que na verdade, algoritmos foram desenvolvidos para que o computador possa gerar números como se fossem aleatórios, mas na verdade eles não são genuinamente aleatórios, e por isso devem receber a denominação mais precisa de pseudo-aleatórios. A explicação de como isso é obtido foge do escopo deste material e não será abordada.

O que importa é que o R é capaz de gerar amostras aleatórias de várias distribuições diferentes, que podem ser usadas em simulações e também em exemplos, como nós fizemos anteriormente. Por exemplo, a fdp de uma binomial foi feita com essa função, gerando uma amostra de tamanho 1000:

```
hist(rbinom(1000, prob=.05, size=100), freq=F)
```

Vamos ver alguns dos números gerados, digamos, 10:

```
> rbinom(10, prob=.05, size=100)
 [1] 8 4 4 4 2 3 6 8 6 5
```

Esses números deveriam estar (e estão) distribuídos mais ou menos em torno de 5, que é a média desta Binomial – temos nesse caso a probabilidade de 5% (`prob=.05`) de sucessos, como o número de experimentos é 100 (`size=100`), esperamos que ocorram 5 sucessos em cada 100 experimentos.

Exercícios

1. Tomando o mesmo exemplo da urna com 10 bolinhas numeradas, qual seria a probabilidade para o caso com e sem reposição, com ordem indiferente para se retirar a bolinha com o número 5 duas vezes?

2. Suponha que uma urna contenha 7 bolinhas azuis e 3 bolinhas verdes. Digamos que estamos interessados na probabilidade de tirarmos uma segunda bolinha verde da urna, sem reposição. Calcule as seguintes probabilidades
   a. A segunda bolinha ser verde, sendo que eu desconheço a cor da primeira bolinha
   b. A segunda bolinha ser verde, dado que a primeira bolinha é verde
   c. A segunda bolinha ser verde, dado que a primeira bolinha é azul
   Que conclusões você pode tirar sobre essas probabilidades condicionais?

3. Calcule a probabilidade desses eventos:
   a. Uma variável distribuída como uma Normal (0,1) ser *menor que* 3
   b. Uma variável distribuída como uma Normal com média 35 e DP de 6 ser *maior que* 42
   c. Obter-se 10 sucessos em 10 experimentos em uma Binomial com probabilidade de sucesso de 0.8
   d. X < 0.9, sendo que X é uma Normal-padrão
   e. X > 6.5 numa distribuição χ^2 com 2 graus de liberdade

4. Como vimos na aula, o intervalo de confiança 95% de uma normal gira em torno da média amostral com um certo afastamento. Quando a variância da população é conhecida (coisa que nunca acontecerá), podemos usar a distribuição Normal (0,1) em vez da distribuição *t*. Nesse caso, o IC $100\times(1-\alpha)$ % será dado por $\bar{x} \pm z_{1-\alpha/2} \times \sqrt{\sigma^2/n}$. Para um IC 95%, nós chegamos a calcular o valor de $z_{1-\alpha/2}$ - lembra? Era aproximadamente 1.96. Calcule $z_{1-\alpha/2}$ para os seguintes níveis de confiança: 10%, 1%, 0.05%, 0.01%

5. Um cirurgião propõe uma nova técnica operatória para uma doença cujo método clássico sabidamente cursa com complicações pós-operatórias em 20% dos pacientes. Ele aplica a nova técnica em 10 pacientes e nenhum apresenta complicações pós-operatórias. Qual é a probabilidade de se usar o método tradicional em 10 pacientes e obter-se o mesmo resultado (i.e. não haver complicação em nenhum paciente)?

Exercícios

Aula 1 – Probabilidade e distribuições

Livro: páginas 45 a 55

1. Tomando o mesmo exemplo da urna com 10 bolinhas numeradas, qual seria a probabilidade para o caso com e sem reposição, com ordem indiferente para se retirar a bolinha com o número 5 duas vezes?

Sem reposição, a probabilidade é zero, pois não é possível tirar a bolinha 5 uma segunda vez se ela não for recolocada na urna.

Com reposição, nós teremos (1/10)x(1/10) = 0.01, ou 1%, já que só existe uma possibilidade de se sortear duas vezes a bolinha 5.

Uma outra forma de se chegar ao mesmo resultado é calcular todas as possíveis amostras de tamanho 2 com reposição que podem ser retiradas de uma população de 10 bolinhas. Esse número é calculado assim: Quantas bolinhas diferentes podem ser sorteadas na primeira escolha? A resposta é óbvia: qualquer uma das 10. Mas se é com reposição, quantas bolinhas diferentes podem ser sorteadas na segunda escolha? 10 novamente. Logo o total de amostras possíveis será 10x10 = 100. Esse resultado pode ser generalizado para o caso de uma população de N elementos e uma amostra de tamanho n. Nesse caso, pode-se mostrar que o número total de amostras possíveis com reposição será N^n . Esse conceito é muito importante e toda a teoria de distribuições amostrais é baseada nesse tipo de amostragem, como você verá em breve.

Muito bem, mas então dentre as 100 possíveis amostras, sem se levar em conta a ordem do sorteio, quantas amostras terão duas vezes o número 5? Apenas uma! Então a probabilidade é 1/100 = 0.01.

Mas você deve estar perguntando agora e se fosse a bolinha 3 e a 5, sem importar a ordem. Bem, nesse caso, nós teríamos duas possibilidades, 3 primeiro e 5 depois ou vice-versa. Então a probabilidade seria 0.02.

2. Suponha que uma urna contenha 7 bolinhas azuis e 3 bolinhas verdes. Digamos que estamos interessados na probabilidade de tirarmos uma segunda bolinha verde da urna, sem reposição. Calcule as seguintes probabilidades

Vamos inicialmente definir as probabilidades de alguns eventos:
Seja:
P(1A) = Probabilidade da primeira bolinha ser azul
P(1V) = Probabilidade da primeira bolinha ser verde
P(2V) = Probabilidade da segunda bolinha ser verde

a. A segunda bolinha ser verde, sendo que eu desconheço a cor da primeira bolinha

Para começar, se nós não sabemos a cor da primeira bolinha, de cara nós teremos duas possibilidades: OU a primeira bolinha era verde OU a primeira bolinha era azul. Já deu para sentir que teremos que SOMAR probabilidades, né? Além disso, cada parcela desta soma envolverá uma multiplicação, pois estaremos lidando com 2 eventos que aconteceram, ou seja, se a primeira bolinha foi verde e a segunda também, temos que fazer a multiplicação das probabilidades destes eventos. Vamos ver como fica:

P(2V) = [P(1V) x P(2V)] + [P(1A) x P(2V)] = (3/10 x 2/9) + (7/10 x 3/9) = 0,3

Note que esse valor é igual à probabilidade de tirarmos uma bolinha verde da primeira vez! Isso significa que o fato de apenas sabermos que uma bolinha já foi tirada da urna, sem saber nada sobre ela, não ajuda em nada no cálculo desta probabilidade.

b. A segunda bolinha ser verde, dado que a primeira bolinha é verde

Agora a história é diferente. Nós sabemos que a primeira bolinha é verde:

P(2V | 1V) = 2/9

c. A segunda bolinha ser verde, dado que a primeira bolinha é azul

Bem, ficou fácil, né?

P(2V | 1A) = 3/9

Que conclusões você pode tirar sobre essas probabilidades condicionais?

A conclusão é que nesses casos, como os eventos são dependentes, a informação sobre um deles modifica a probabilidade de ocorrência do outro. Já percebeu como os mesmos eventos seriam independentes? Uma questão de bônus para quem me procurar até a próxima aula e me explicar como...

3. Calcule a probabilidade desses eventos:

a. Uma variável distribuída como uma Normal (0,1) ser *menor que* 3

Para resolver esta questão, precisamos apenas aplicar diretamente a função de densidade acumulada:

```
> pnorm(3)
[1] 0.9986501
```

b. Uma variável distribuída como uma Normal com média 35 e DP de 6 ser *maior que* 42

Repare que nesse caso, a aplicação direta da função de densidade acumulada não vai funcionar. Você deve usar uma propriedade simples das funções de densidade acumuladas para resolver facilmente esse problema. A área total abaixo da fdp é sempre 1, pois esse é valor máximo de uma probabilidade. Isso significa que a área total abaixo da curva da Normal vale 1. Ora, a função de densidade acumulada nos dá $P(X \leq z)$, então $P(X \leq z) + P(X > z) = 1$, logo:

$P(X > z) = 1 - P(X \leq z)$

No R:

```
> 1-pnorm(42, mean=35, sd=6)
[1] 0.1216725
```

A outra maneira de se obter o mesmo resultado era se você se lembrar da característica da Normal ser simétrica em torno da sua média. O valor sobre o qual eu quero a probabilidade está a 7 unidades à direita da média (42-35). Ora, então a área abaixo da curva para valores acima de 42, terá de ser a mesma que a área abaixo da curva para valores abaixo de 35-7 = 28 unidades. Vamos ver no R:

```
> pnorm(28, mean=35, sd=6)
[1] 0.1216725
```

De maneira geral, para uma Normal (0,1), cuja média é zero, essa propriedade pode ser descrita:

$1 - \Phi(z_p) = \Phi(-z_p)$

Onde Φ representa a função de densidade acumulada da Normal (0,1), z_p representa um quantil da Normal (0,1)

c. Obter-se 10 sucessos em 10 experimentos em uma Binomial com probabilidade de sucesso de 0.8

Neste caso a pergunta se refere a exatamente 10 sucessos (repare que isso é diferente de *no máximo* 10 sucessos. Estamos procurando $P(X=0)$ em uma Binomial (p = 0.8, n = 10). No R:

```
> dbinom(10, prob=.8, size=10)
[1] 0.1073742
```

Note que a probabilidade é relativamente alta (~10%), o que é esperado, já que a média dessa Binomial é 0.8x10 = 8.

Se a pergunta fosse sobre no máximo 10 sucessos, a resposta seria óbvia e de cabeça: 1, pois 10 *é* o número máximo de sucessos, logo, tudo o que pode acontecer está entre 0 e 10 sucessos. Mas quem quiser conferir no R será bem-vindo:

```
> pbinom(10, prob=.8, size=10)
[1] 1
```

d. X < 0.9, sendo que X é uma Normal-padrão

Alguma dúvida? No R:

```
> pnorm(0.9)
[1] 0.8159399
```

e. X > 6.5 numa distribuição χ^2 com 2 graus de liberdade

Aqui teremos que usar a propriedade da função de densidade acumulada (cuidado: A distribuição qui-quadrada NÃO É SIMÉTRICA!!!) Novamente:

$P(X>x)$ = 1- $P(X \leq x)$

```
> 1-pchisq(6.5, df=2)
[1] 0.03877421
```

Confira no R como a χ^2_2 não é simétrica:

```
curve(dchisq(x, df=2), from=0, to=10)
```

4. Como vimos na aula, o intervalo de confiança 95% de uma normal gira em torno da média amostral com um certo afastamento. Quando a variância da população é conhecida (coisa que nunca acontecerá), podemos usar a distribuição Normal (0,1) em vez da distribuição *t*. Nesse caso, o IC $100\times(1-\alpha)$ % será dado por $\bar{x} \pm z_{1-\alpha/2} \times \sqrt{\sigma^2/n}$. Para um IC 95%, nós chegamos a calcular o valor de $z_{1-\alpha/2}$ - lembra? Era aproximadamente 1.96. Calcule $z_{1-\alpha/2}$ para os seguintes níveis de confiança: 10%, 1%, 0.05%, 0.01%

Esse exercício é uma aplicação direta da fórmula de quantis. A única atenção tem que ser dada ao fato de os números estarem em porcentagem:

Para 10%, $1-\alpha/2$ = 0.95

Para 1%, $1-\alpha/2$ = 0.995

Para 0.05%, $1-\alpha/2$ = 0.99975

Para 0.01%, $1-\alpha/2$ = 0.99995

Então no R:

```
> qnorm(0.95)
```

```
[1] 1.644854
> qnorm(0.995)
[1] 2.575829
> qnorm(0.99975)
[1] 3.480756
> qnorm(0.99995)
[1] 3.890592
```

5. Um cirurgião propõe uma nova técnica operatória para uma doença cujo método clássico sabidamente cursa com complicações pós-operatórias em 20% dos pacientes. Ele aplica a nova técnica em 10 pacientes e nenhum apresenta complicações pós-operatórias. Qual é a probabilidade de se usar o método tradicional em 10 pacientes e obter-se o mesmo resultado (i.e. não haver complicação em nenhum paciente)?

Neste caso, temos duas possibilidades de calcular esta probabilidade. Repare que o que estamos tentando descobrir é $P(X=0)$, ou seja, a densidade de uma Binomial no ponto 0, mas para uma distribuição Binomial, cujo *range* é de $k = 0,1,2,...,n$, isso equivale a calcular a função de densidade acumulada também, $P(X \leq 0)$. Pelas características do problema, essa é uma Binomial($p = 0.2, n = 10$). Para obter o valor no R com a fdp:

```
> dbinom(0, prob=0.2, size=10)
[1] 0.1073742
```

Usando a função de densidade acumulada:

```
> pbinom(0, prob=0.2, size=10)
[1] 0.1073742
```

É claro que nesse caso, poderíamos facilmente calcular esse valor na mão. Lembre-se da fdp da Binomial:

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Substituindo os valores de *k, p* e *n* nessa equação:

$$P(X=0) = \binom{10}{0} 0.2^0 (1-0.2)^{10-0} = 0.8^{10} = 0.1073742$$

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 2 – Estatística descritiva
Livro: páginas 57 a 80

Nesta e na próxima aula também usaremos alguns elementos já descritos no módulo básico, e a repetição mais uma vez se justifica pelos motivos já mencionados. Novamente, parte do material será tomada "emprestada" do módulo básico.

Uma diferença importante deve ser mencionada é que a partir desta aula, nós iremos efetivamente usar a biblioteca `ISwR`. Essa biblioteca é composta basicamente de bancos de dados que usaremos para exemplificar alguns procedimentos. Isso significa que teremos que inicializar a biblioteca, chamar alguma base de dados dessa biblioteca e eventualmente anexar essa base de dados para poupar digitação. Digamos que iremos usar a base de dados `juul` anexada. Nesse caso, precisaremos invocar a seguinte seqüência de dados:

```
library(ISwR)
data(juul)
attach(juul)
```

Nenhuma dessas funções é nova para você se você já fez os módulos iniciais do R, mas só para relembrar, a primeira carrega a biblioteca, a segunda inicializa a base de dados e a terceira anexa o banco.

O importante é lembrar que esses comando somente ficarão ativos durante uma sessão do R, ou seja, se você fechar o programa, esses comandos serão perdidos e numa próxima sessão, terão que ser invocados novamente. Isso é importante porque as aulas são feitas como se o aluno estivesse em uma única sessão do R. Isso significa que dentro de uma mesma aula, esses comandos não serão repetidos, e se o aluno resolver seguir a aula em várias sessões, essa inicialização deverá ser feita toda vez que a sessão for iniciada (obviamente substituindo o nome do banco de dados, que não será apenas o `juul` , naturalmente.)

Descrição univariada
- Estatísticas-resumo
- Gráficos
  - Histogramas
  - Distribuição acumulativa empírica
  - Q-Q plots
  - Boxplots

Descrição univariada por grupos
- Estatísticas-resumo
- Gráficos
  - Histogramas
  - Boxplots paralelos

Descrição de dados categóricos

- Tabelas
- Gráficos

Descrição univariada

Apesar do palavrão aí do título, ele apenas quer dizer que vamos ver a descrição de variáveis uma a uma, sem estratificar ou cruzar esta variável com outras variáveis.

Estatísticas-resumo

No R é fácil obter-se estatísticas-resumo básicas. Na verdade nós já fizemos isso no Módulo Básico, quando estudamos distribuições e gráficos. Vamos aliás pegar emprestado o que já fizemos: vamos gerar uma amostra de 100 observações de uma Normal (0,1):

```
x <- rnorm(100)
```

Agora, vamos calcular algumas estatísticas:

```
> mean(x)
[1] -0.05940595
> sd(x)
[1] 0.9533712
> var(x)
[1] 0.9089166
> median(x)
[1] -0.08017842
> sqrt(var(x))
[1] 0.9533712
```

A primeira observação, claro, é que a média e a mediana não são iguais a zero e a variância e desvio-padrão também não são exatamente iguais a 1. Bom, como já comentamos no Módulo Básico, isso acontece porque nós geramos esses números, então eles devem ter uma distribuição *aproximadamente* igual a uma Normal (0,1). A segunda observação é que apesar de ser intuitivo, parece ser absolutamente não prático obter esses valores... Será que não existe um modo mais fácil??? Relaxe: existe e veremos já-já...

Antes, vamos ver algumas outras estatísticas, como por exemplo quantis:

```
> quantile(x)
         0%         25%         50%         75%        100%
-2.34662031 -0.65581393 -0.08017842  0.57501559  2.61270000
```

Repare que essa função retorna o valor mínimo, o máximo e ainda a mediana (por definição o quantil 0.5, ou centil 50), o primeiro e o terceiro quartis (que são os centis 25 e 75, respectivamente). Essa função aceita também um argumento para fornecer o centil que se desejar. Por exemplo, para obter-se o 10º e o 90º. centis, pode-se fazer:

```
> quantile(x, c(0.1, 0.9))
      10%       90%
-1.185000  1.160362
```

Muito bem, vamos agora pela primeira vez usar dados contidos na biblioteca `IswR`, para podermos estudar descrições em bancos de dados reais:

```
library(ISwR)
```

Vamos seguir o nosso livro de referência e usar o banco juul, que contém valores repetidos de IGF-I para um grupo de crianças em idade escolar, além de outras variáveis. Para uma descrição mais detalhada desse banco, digite:

```
?juul
```

Muito bem. Vamos agora usar o nosso banco. Mas antes, vamos salvá-lo com outro nome, para facilitar a nossa vida:

```
data(juul)
aula2.juul <- juul
attach(aula2.juul)
```

O primeiro comando é usado para carregar o banco `juul` para a memória do R e o segundo você deve se lembrar, é para não termos que digitar o nome do banco toda vez que quisermos chamar uma variável deste banco. Aliás, vamos conferir o nome destas variáveis:

```
> names(aula2.juul)
[1] "age"      "menarche" "sex"      "igf1"     "tanner"   "testvol"
```

Bom, vamos calcular uma média então da variável IGF-I:

```
> mean(igf1)
[1] NA
```

Epa!!! Como é isso? A média da IGF-I é não disponível??? (Lembra? `NA` no R significa *missing*.) Calma: isso acontece pelo modo como o R lida com valores faltantes. Depende da função, mas em geral, o R não remove automaticamente os valores faltantes para calcular estatísticas – isso funciona na verdade como um "alarme" para informar que existem registros sem informação. Então, nós precisamos indicar para o R que ele deve remover os `NA`s. Assim:

```
mean(igf1, na.rm=T)
[1] 340.168
```

Entendeu? Usamos a opção `na.rm=T` para `r`emover os `Na`s. Isso funciona para outras funções também:

```
> median(igf1)
[1] NA
> median(igf1, na.rm=T)
[1] 313.5
```

Certo, mas e a nossa função que nos dá uma saída mais resumida, mais "jeitosa"? Vamos lá; que tal um resumo da nossa variável?

```
> summary(igf1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   25.0   202.3   313.5   340.2   462.8   915.0   321.0
```

Com essa função temos então algumas estatísticas básicas (porém não todas as que você gostaria de calcular, talvez) e ainda o número de registros com *missing* para aquela variável. Mais tarde veremos como personalizar algumas saídas.

Na verdade, essa mesma função pode ser usada para obter um resumo de um banco inteiro:

```
> summary(aula2.juul)
```

```
      age              menarche           sex              igf1
 Min.   : 0.170   Min.   :  1.000   Min.   :1.000   Min.   : 25.0
 1st Qu.: 9.053   1st Qu.:  1.000   1st Qu.:1.000   1st Qu.:202.3
 Median :12.560   Median :  1.000   Median :2.000   Median :313.5
 Mean   :15.095   Mean   :  1.476   Mean   :1.534   Mean   :340.2
 3rd Qu.:16.855   3rd Qu.:  2.000   3rd Qu.:2.000   3rd Qu.:462.8
 Max.   :83.000   Max.   :  2.000   Max.   :2.000   Max.   :915.0
 NA's   : 5.000   NA's   :635.000   NA's   :5.000   NA's   :321.0
     tanner          testvol
 Min.   :  1.000   Min.   :  1.000
 1st Qu.:  1.000   1st Qu.:  1.000
 Median :  2.000   Median :  3.000
 Mean   :  2.640   Mean   :  7.896
 3rd Qu.:  5.000   3rd Qu.: 15.000
 Max.   :  5.000   Max.   : 30.000
 NA's   :240.000   NA's   :859.000
```

Observando essas estatísticas-resumo, observamos um outro problema: o R está considerando variáveis de classificação, como menarca (`menarche`), sexo (`sex`), e classificação de Tanner (`tanner`) como sendo contínuas. Além disso, a falta de um significado para os códigos também atrapalha um pouco. Vamos usar então uma função para transformar estas variáveis em fatores (lembra?) e atribuir alguns *labels* para os seus valores. Faça assim:

```
detach(aula2.juul)
aula2.juul$sex <- factor(juul$sex, labels=c("M","F"))
aula2.juul$menarche <- factor(juul$menarche, labels=c("Não","Sim"))
aula2.juul$tanner <- factor(juul$tanner, labels=c("I","II", "III", "IV",
"V"))
attach(aula2.juul)
summary(aula2.juul)
```

E veja como a saída agora é bem mais agradável. Note que tivemos que remover o banco da memória com a função antes de fazermos as modificações. Isso é necessário porque a versão que você carrega para a memória fica preservada e não é alterada quando você faz alterações no banco original.

Gráficos

Já sei: está preocupado com a função que eu prometi mostrar que tem uma saída mais palatável, não é? Mas vamos esperar mais um pouquinho, pois ela também fornece uns gráficos interessantes...

Histogramas

Primeiramente vamos falar dos nossos conhecidos histogramas. Como já vimos anteriormente, o histograma apresenta a distribuição de freqüências de uma variável qualquer, seja ela gerada pelo R, como uma Normal (0,1), seja de uma variável contínua qualquer.

```
hist(igf1, freq=F)
```

Hum... Não parece muito normal, não é? Vamos comparar com uma Normal teórica?

```
hist(igf1, freq=F)
curve(dnorm(x, mean=mean(igf1, na.rm=T), sd=sd(igf1, na.rm=T)), from=0,
to=1000, add=T)
```

Até que não é tão ruim assim... O que você acha? Repare que eu usei a função `curve()` que nós já tínhamos usado anteriormente, usando como parâmetros da Normal a média e o desvio-padrão da variável `igf1` calculados pelo R – observe que eu tive que retirar os *missings* também! Além disso, a opção `add=T` tem que ser usada para a cure aparecer na mesma janela do histograma. Ah, e para esse caso, temos que usar o histograma com a densidade e não com a freqüência.

Os histogramas são muito úteis para observar o "jeitão" da nossa variável. Existe um outro recurso, que muita gente não gosta, mas que eu considero mais informativo do que o histograma para este tipo de descrição, que é o gráfico de ramo-e-folhas (que já foi visto também no Módulo Básico.) Vamos ver como ficaria:

```
> stem(igf1, scale=2)

  The decimal point is 1 digit(s) to the right of the |

   2 | 5968
   4 | 233346123
   6 | 478811699
   8 | 0678899000234556789
  10 | 00111344455666678011235577 79
  12 | 12234666701234455567888999
  14 | 00111223444566889999001112223334466677788 8899
  16 | 000013333444566677788889990011111123444444466778899999 9
  18 | 00011234455666677777777888000001133334556667789999
  20 | 00111122223334455666789990012222444555577 8889
  22 | 000111122223333333444555678899912222223334444555667888999 9
  24 | 000122222233445566667778899001122244555666777788899999 9
  26 | 111112222233466778899901111112222344578888999 9
  28 | 000111133444456667888890000001222344555666899
  30 | 0144457789912233456889
  32 | 11222223445667888899001266 8889
  34 | 014555677777888999122222445566 68899
  36 | 000122333334444466778801124444456678 9
  38 | 033356777778899900111233445567788899 99
  40 | 00122244455666778990112224456667788999 99
  42 | 00111255688890000112344455666 8999
  44 | 00122223334455677788902233337 79
  46 | 11223444455678990123334445566677 88899
  48 | 012333455678888889000011112233334455566777 889
  50 | 01223334444455689044777888
  52 | 02234456666778890355555
  54 | 1357889901223336688 89
  56 | 004455800344689
  58 | 112780125569
  60 | 034889991556889
  62 | 124500125799
  64 | 81235
  66 | 7890
  68 | 02679069
  70 | 1146
  72 | 0222383778
  74 | 137
  76 | 2897
  78 | 158
  80 | 1
  82 | 1028
  84 | 23
  86 | 8
  88 | 4
  90 | 45
```

Observe que com essa saída visualizamos o "jeitão" quase normal, que observamos no histograma e temos uma idéia da distribuição dos valores no banco também. Por exemplo, o valor mínimo pode ser estimado, que é 25 (observe a mensagem dizendo que o ponto decimal está uma casa à direita da barra, então o primeiro valor será 25 ou 35 – isso por causa do intervalo estabelecido pelo gráfico – cada ramo engloba 2 dezenas, ou seja, o ramo 2 engloba todos os valores de 20 a 39; mas como a seqüência é `5968` esses valores são 25, 29, 36, 38.) O mesmo acontece para o valor máximo, que é 905 ou 915, pelo mesmo motivo já comentado, mas aqui não podemos dizer exatamente qual. Além disso, podemos ver que os valores mais freqüentes estão em torno de 220, 260, mais ou menos.

Distribuição acumulativa empírica

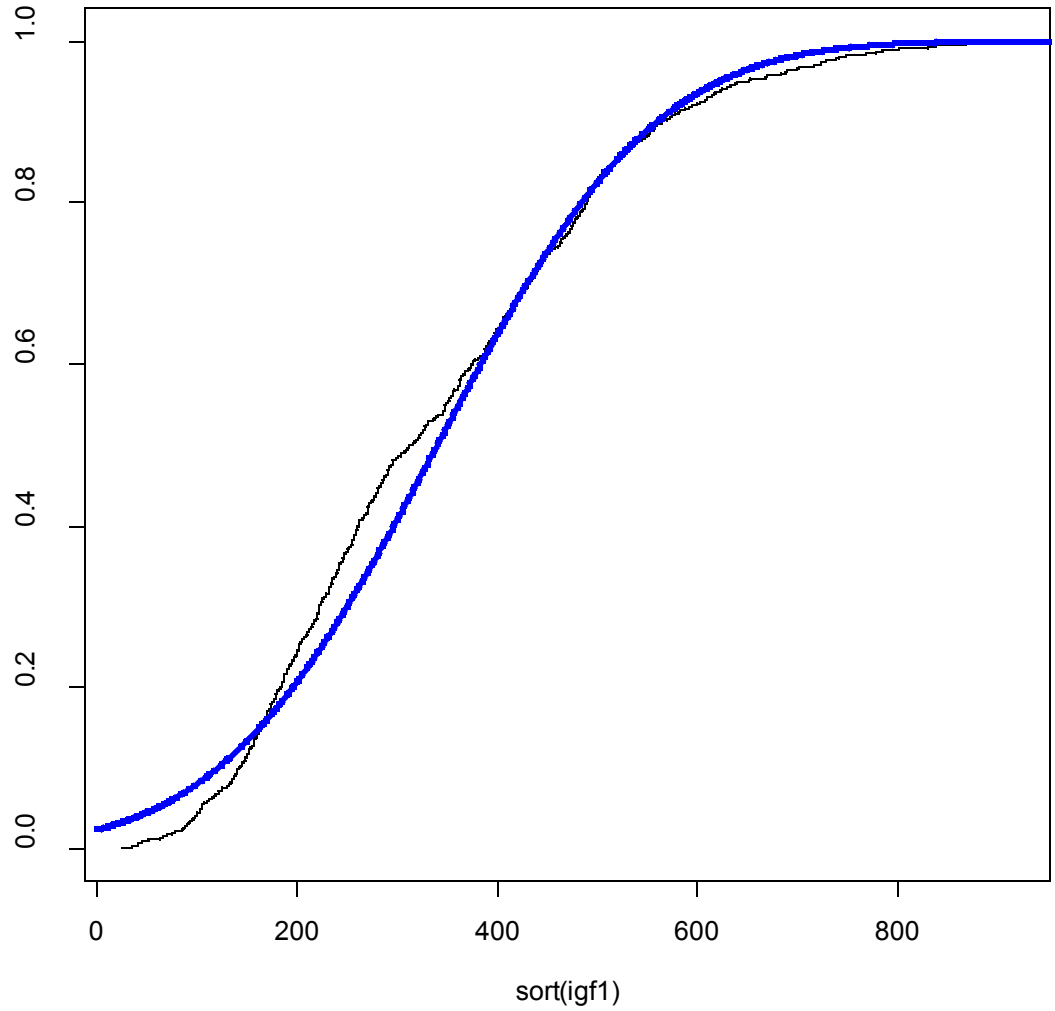
Esse gráfico nós já vimos na aula passada, quando construímos os gráficos das funções de densidade acumulada para distribuições, usando, por exemplo a função `pnorm()` para a Normal (0,1). Isso é possível de ser feito para uma variável também.

Na prática mesmo, isso não é muito usado, já que um dos seus objetivos, que é testar normalidade, contra uma distribuição simulada, por exemplo pode ser feita de uma maneira mais eficiente como veremos a seguir, mas vamos ver como funciona:

```
      n<-sum(!is.na(igf1))
      x<-seq(1,n,1)
      plot(sort(igf1), (1:n)/n, type="s")
      lines(x, pnorm(x, mean=mean(igf1, na.rm=T), sd=sd(igf1, na.rm=T)),
col="blue", lwd=3)
```

Bem, não vou entrar em muitos detalhes sobre o que foi feito e colocamos o resultado final no gráfico abaixo. Apenas a primeira linha será comentada. Muitas vezes, queremos saber o número de observações de uma variável que não são *missing* e usar este número para alguma coisa, como foi o caso aqui, para construir a distribuição empírica. Bem, você se lembra que a IGF-I tem 321 valores faltantes, e que o total de registros é de 1339. Ora, então efetivamente usaremos 1339-321 = 1018 valores para serem usados. Mas olhe o que acontece quando chamamos a função para medir o tamanho de um objeto:

```
      > length(igf1)
      [1] 1339
```



E o pior é que a opção para remover os *missing* não é aceita pela função `length()`. Então, tivemos que usar esse macete:

```
> sum(!is.na(igf1))
[1] 1018
```

Não é difícil entender esse código: eu fiz uma pergunta lógica – que elementos em `igf1` são diferentes de `NA` – com o código `!is.na(igf1)`. Esse teste vai retornar um vetor cheio de valores `TRUE` e `FALSE` que como você já aprendeu, correspondem aos valores 1 e zero, respectivamente. Ora, quantos desses valores serão `TRUE`? Todos os que não forem faltantes, certo? Então a soma total desses valores será igual ao número de valores não faltantes.

Muito bem, mas veja só a comparação da distribuição empírica (em preto) com a teórica (em azul). O que acha? Normalzinha?

Q-Q plots

Bem, vamos dar uma rápida olhada no *q-q plot*. Literalmente significa gráfico quantil-quantil. Ficou difícil? Não se preocupe. Na verdade é a mesma coisa que nós já fizemos na nossa distribuição empírica, só que desenhados em uma única curva (em vez de 2, como fizemos na seção anterior.) Vamos tentar:

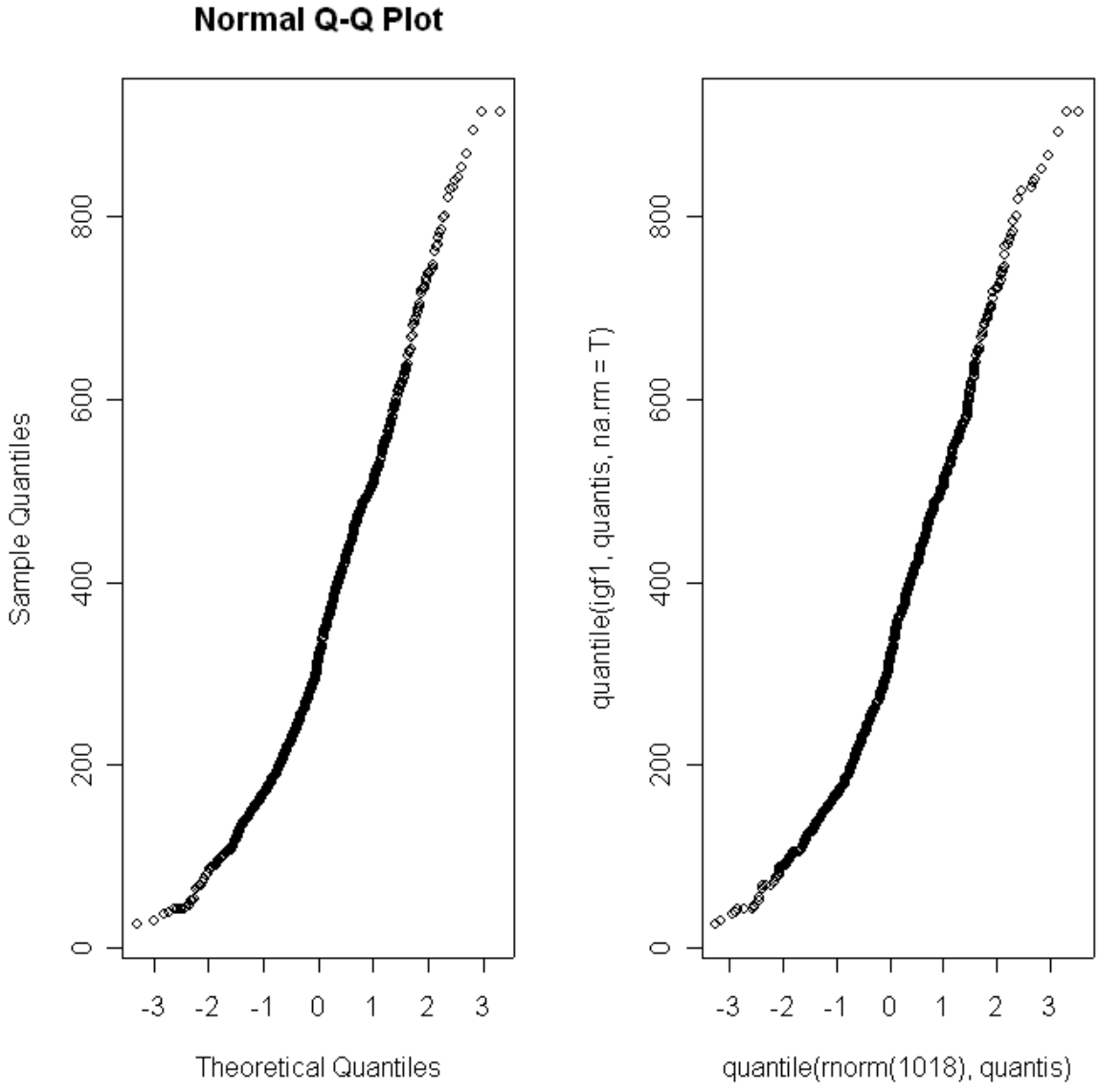
```
qqnorm(igf1)
```

Repare nos nomes dos eixos do gráfico: estamos comparando os quantis observados com os quantis teóricos. Foi exatamente o que fizemos antes. Vamos tentar? Vamos usar então aquela função que nós vimos anteriormente, mas vamos calcular vários quantis e depois vamos plotar quantis de uma Normal (0,1) contra os quantis reais de :

```
quantis <- seq(0,1,0.001)
plot(quantile(rnorm(1018),quantis), quantile(igf1,quantis,na.rm=T))
```

Achou parecido com o gráfico anterior? Não lembra? Vamos ver lado a lado:

```
par(mfrow=c(1,2))
qqnorm(igf1)
plot(quantile(rnorm(1018),quantis), quantile(igf1,quantis,na.rm=T))
par(mfrow=c(1,1))
```



Elas não são exatamente iguais porque nós geramos a normal ao fazer o gráfico, mas a idéia é a mesma. Para ser normal, a curva deveria ser uma reta diagonal.

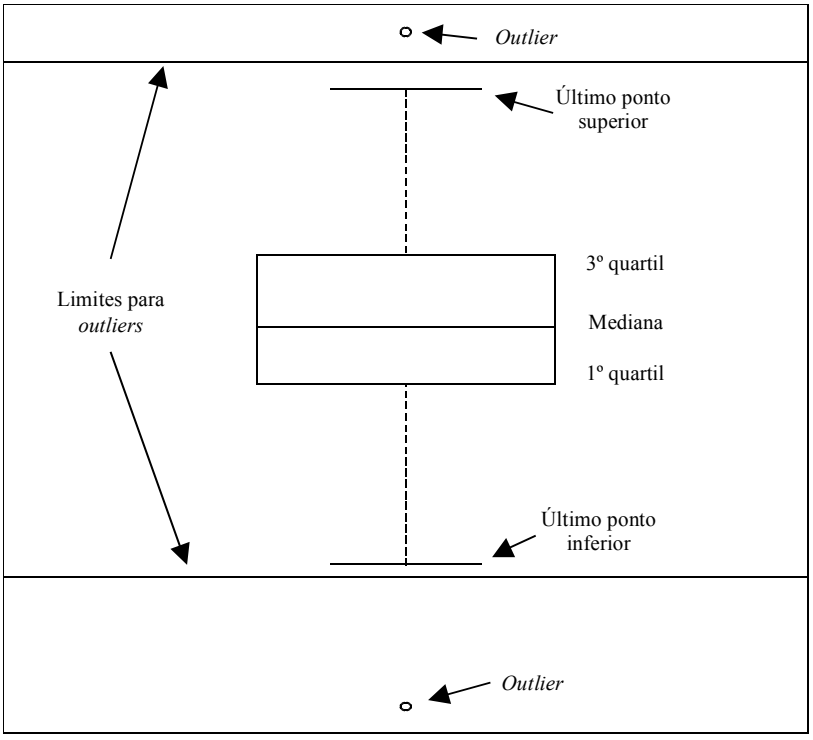
Boxplots

Bem, o boxplot foi bem explicado no módulo básico, mas para aqueles que não pegaram a explicação, vamos repetir aqui. Vamos fazer um primeiro:

```
boxplot(igf1)
```

A linha central do retângulo (que seria a nossa "caixa") representa a mediana da distribuição. As bordas superior e inferior do retângulo representam os percentis 25 e 75, respectivamente (também conhecidos como primeiro e terceiro quartís, respectivamente). Logo, a altura deste retângulo é chamada de distribuição interquartilar (DI). Os traços horizontais ao final das linhas verticais são traçados sobre o último ponto (de um lado ou de outro) que não é considerado um *outlier*.

No caso do boxplot em geral, existe uma definição formal de *outlier*, que é adotada pelo R. A maior parte das definições considera que pontos acima do valor do 3º quartil somado a 1,5 vezes a DI ou os pontos abaixo do valor do 1º quartil diminuído de 1,5 vezes a DI são considerados *outliers*. Esses pontos são assinalados (no nosso exemplo, tivemos 2 *outliers*, um para cada lado). A figura acima mostra como isso funciona esquematicamente.

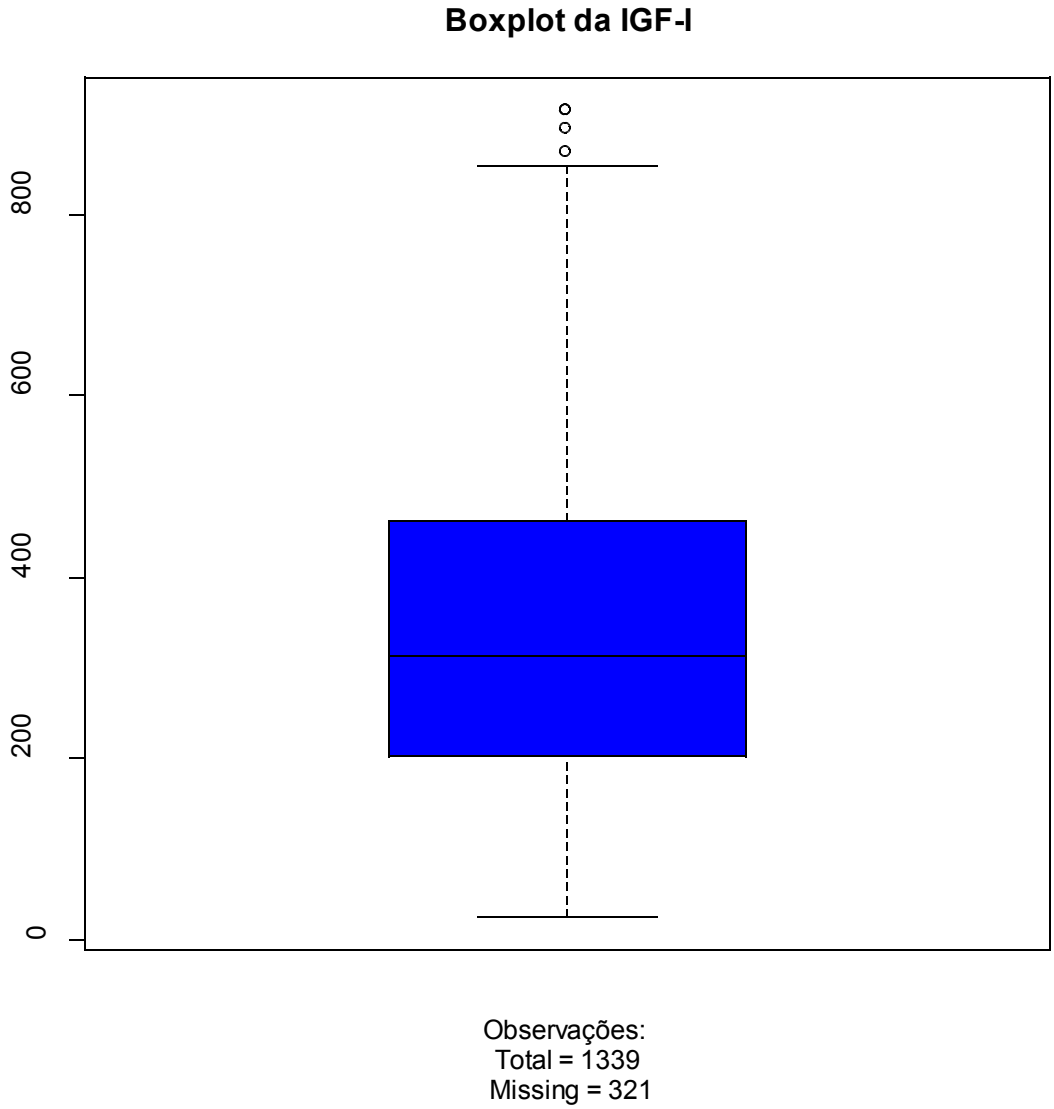


Agora você deve estar se perguntando se esse boxplot que você fez no R poderia conter mais informação e ser esteticamente mais apresentável, por exemplo se você quiser utilizá-lo num

documento ou numa apresentação em uma aula, etc. Claro que podemos. Vamos ver algumas opções para o nosso gráfico:

```
      boxplot(igf1, main="Boxplot da IGF-I", ylab="microgramas/litro",
sub=paste("Observações:", "\n", "Total =",length(igf1), "\n", "Missing =",
sum(is.na(igf1)), sep=" " ), col="blue")
```

Veja o resultado:



Para finalizar, vamos dar uma olhada em uma função muito interessante que foi escrita inicialmetne para o S-Plus, mas que foi passada para o R e disponibilizada pelo nosso guru, o Professor Oswaldo Cruz:

```
eda <- function(x, na.rm=FALSE)
{
      require(ctest)
      if (!is.numeric(x) || !is.vector(x))  {
            stop("argumento não é vetor ou não é numerico ")
      }
      old.par <- par(mfrow = c(2, 2), oma = c(1, 1, 2, 1))
```

```
            densi  <- density(x, na.rm=na.rm)
            xli  <- range(densi$x)
            yli  <- range(densi$y)
            hist(x,col="red",probability = T,xlim = xli, ylim =
yli,main=paste("Histograma de ",deparse(substitute(x))), ylab="Densidade",
xlab="")
            lines(densi,lwd=2)
            boxplot(x,col="limegreen", main=paste("Boxplot de
",deparse(substitute(x))))
            points(1, mean(x), pch = 16,cex=1,col="blue")
            qqnorm(x, xlab="Quantis Teóricos", ylab="Quantis da Amostra",
sub="",main="qq-plot Normal")
            qqline(x,col="orange",lwd=2)
            plot(seq(1:50),seq(1:50),axes=F,type="n",ylab="",xlab="")
            lugarnomes <- list(x=c(24,24,24,24,24,24),y=c(50,42,34,26,18,10))
            text(lugarnomes,c("Minimo =","1°Quartil =","Mediana =","Média
=","3°Quartil=","Maximo =),adj=1)
            lugx=c(25,25,25,25,25,25)
            lugy=c(50,42,34,26,18,10)
            text(lugx,lugy,summary(x),adj=0)
            par(old.par)
            title("Sumário")
      }
```

Veja só o resultado muito bom que se consegue pra um rápido resumo descritivo da nossa variável `igf1`:

```
eda(igf1, na.rm=T)
```

Bem legal, não?

Descrição univariada por grupos

Nesta seção, discutiremos a descrição de uma variável por grupos. Esta abordagem é usada para mostrar variáveis contínuas por determinados subgrupos. Por exemplo, digamos que você quer saber se os níveis de IGF-I varia de acordo com as categorias de sexo e classificação de Tanner. Na verdade, queremos a descrição desses dados por grupos de sexo ou de classificação.

Estatísticas-resumo

Comecemos pelas estatísticas-resumo. A maneira mais fácil de se fazer isso no R é usando a função `tapply()` . Ela vai aplicar uma outra função a uma tabela qualquer. Esta tabela vai ser definida como o cruzamento de duas variáveis: a que você quer aplicar a função e a variável de agrupamento. Assim, se eu quiser saber a média de IGF-I por sexo:

```
> tapply(igf1,sex,mean)
 M  F
NA NA
```

Hum... parece que o nosso problema com dados faltantes está presente aqui também... Vamos pedir então para a função retirá-los:

```
> tapply(igf1,sex,mean,na.rm=T)
       M        F
310.8866 368.1006
```

Bem melhor... Repare que a função `mean()` é passada sem os parênteses. Vamos fazer o mesmo com a classificação:

```
> tapply(igf1,tanner,mean,na.rm=T)
       I       II      III       IV        V
207.4727 352.6714 483.2222 513.0172 465.3344
```

Ah, e podemos também usar a nossa função `summary()`:

```
> tapply(igf1,tanner,summary)
$I
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   29.0   151.0   201.0   207.5   259.0   624.0   204.0

$II
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  157.0   269.5   341.5   352.7   443.3   682.0    33.0

$III
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  167.0   383.0   474.0   483.2   553.0   868.0    27.0

$IV
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  321.0   447.5   500.0   513.0   574.0   915.0    23.0

$V
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  100.0   389.8   452.0   465.3   528.0   914.0    20.0
```

Nada mal... E com o R nós podemos criar funções personalizadas para apresentar várias estatísticas por grupos. Vamos a um exemplo:

```
resumo.grupo <- function (x, y, na.rm=F)
{
      min <- tapply(x, y, min, na.rm=na.rm)
      max <- tapply(x, y, max, na.rm=na.rm)
      mediana <- tapply(x, y, median, na.rm=na.rm)
      media <- tapply(x, y, mean, na.rm=na.rm)
      sd <- tapply(x, y, sd, na.rm=na.rm)
      cbind(Mínimo=min, Máximo=max, Mediana=mediana, Média=media, DP=sd)
}
```

Esta função vai nos dar uma saída composta de estatísticas básicas, num formato mais confortável. Vamos testar:

```
> resumo.grupo(igf1,sex,na.rm=T)

  Mínimo Máximo Mediana    Média       DP
M     29    915     280 310.8866 169.7136
F     25    914     352 368.1006 167.3476
```

Que tal?

Gráficos

Obviamente, podemos utilizar gráficos também para descrever variáveis por grupos. Na verdade são os mesmos gráficos que nós já vimos antes, porém separados, ou estratificados por grupos.

Histogramas

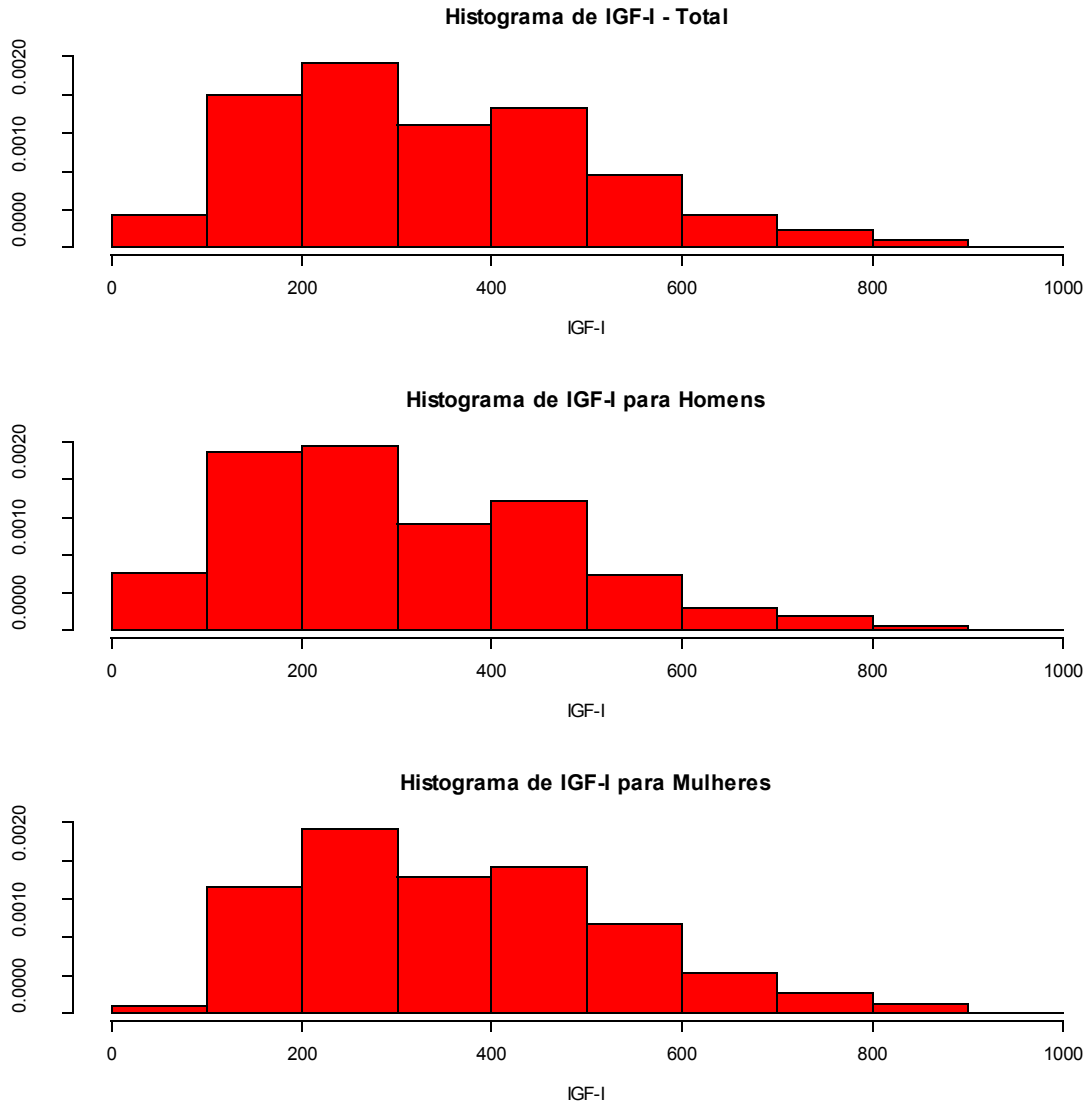
Muitas vezes é interessante observar a distribuição de uma variável por grupos. Para este fim, podemos utilizar histogramas estratificados por categoria também. Vamos ver uma maneira "suja e rápida" de obter histogramas por grupos

```
par(mfrow=c(3,1))
hist(igf1)
by(igf1,sex,hist)
par(mfrow=c(1,1))
```

Veja o que aconteceu: Primeiro nós dividimos a tela gráfica em 3 linhas e uma coluna, depois fizemos o histograma para a IGF-I sem dividir por grupos e em seguida usamos a função `by()` para fazer os histogramas da IGF-I por categoria de sexo. Eu chamei de rápida e suja, porque na saída não temos os títulos apropriados para os gráficos, e além disso, o R "cospe" uma saída em texto na tela... Não se preocupe: esse texto contém todas as informações sobre os histogramas que você acabou de plotar, além de dizer que grupos foram usados: primeiro foram os homens e depois as mulheres. Para apresentar não está muito bom, de jeito nenhum, mas dá para ter uma idéia. Para uma saída mais elegante, poderíamos usar:

```
par(mfrow=c(3,1))
hist(igf1, main="Histograma de IGF-I - Total", xlab="IGF-I",
ylab="Densidade", col="red", freq=F)
hist(igf1[sex=="M"], main="Histograma de IGF-I para Homens", xlab="IGF-I",
ylab="Densidade", col="red", freq=F)
hist(igf1[sex=="F"], main="Histograma de IGF-I para Mulheres", xlab="IGF-
I", ylab="Densidade", col="red", freq=F)
par(mfrow=c(1,1))
```

Veja o resultado:



Boxplots

Assim como histogramas, os boxplots também podem ser feitos por grupos. Vamos ver como funciona a sintaxe do boxplot, com alguns detalhes para ficar mais apresentáveis:

```
boxplot(igf1~sex, col="blue", main="Boxplot de IGF-I por Sexo")
```

Repare no til que nós usamos entre a variável `igf1` e o sexo, que é a variável de agrupamento. Esse til significa "por" neste caso. Nós ainda vamos usá-lo muito quando aprendermos modelos lineares mais adiante.

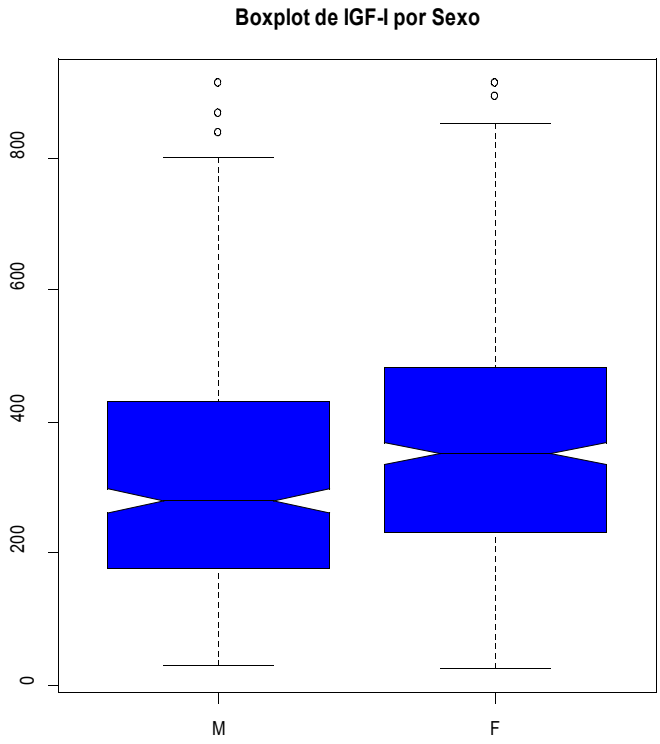
Às vezes também é interessante dispor os boxplots em conformação horizontal. Veja um exemplo:

```
boxplot(igf1~tanner, col="blue", main="Boxplot de IGF-I por Classificação
de Tanner", horizontal=T)
```

Por último, vou mencionar uma característica muito interessante disponível no R. É a colocação de uma indentação no boxplot, chamada *notch*. A utilidade disso é que os boxplots de

grupos cujas indentações não se sobrepuserem têm suas medianas significativamente diferentes para um nível de 0.05. Veja como é:

```
boxplot(igf1~sex, col="blue", main="Boxplot de IGF-I por Sexo", notch=T)
```



Como se observa, a mediana de IGF-I para os homens é significativamente diferente da mediana de IGF-I para as mulheres (as indentações não se sobrepõem.)

Descrição de dados categóricos

Até agora discutimos como descrever variáveis contínuas, seja o seu total ou dividida por categorias de uma outra variável. Mas e quando queremos estudar também a própria variável categórica? Nesse caso, vamos usar ferramentas diferentes daquelas usadas para as variáveis contínuas.

Tabelas

A forma mais usual e simples de se descrever dados categóricos é com o uso de tabelas de contingência. Com o R é possível construir essas tabelas, mas não há um instrumental adequado para uma formatação de alto nível para apresentação de dados em um documento ou para a preparação de *slides*. Como já mostrado no módulo "Entrada e Saída de Dados no R", essas tabelas podem ser exportadas facilmente para o formato CSV que pode ser lido por diversos programas tanto estatísticos como planilhas eletrônicas.

Vamos usar o mesmo banco que vínhamos usando para exemplificar. Vamos estudar as relações entre as variáveis categóricas nesse banco. Começando por uma tabulação para cada variável:

```
> table(sex)
sex
  M   F
621 713

> table(menarche)
menarche
Não Sim
369 335

> table(tanner)
tanner
  I  II III  IV   V
515 103  72  81 328
```

Podemos também fazer tabelas de dupla entrada, claro:

```
> table(sex,tanner)
   tanner
sex   I II III IV   V
  M 291 55  34 41 124
  F 224 48  38 40 204

> table(menarche,tanner)
        tanner
menarche   I II III IV   V
     Não 221 43  32 14   2
     Sim   1  1   5 26 202
```

E até mesmo de tripla entrada, embora neste caso isso não faça sentido...

```
table(menarche,tanner, sex)
```

A essas alturas, você deve estar estranhando um pouco essas tabelas que o R está fazendo, não? Pois é: elas não têm as marginais, ou seja os totais das linhas e das colunas. Outra questão é se você estivesse querendo uma tabela de proporções e não de números absolutos. Bem, para facilitar a vida, vamos criar uma função para que possamos escolher o tipo de tabela que queremos:

```
tabela.2x2 <- function(..., margens=F, prop=c("no", "colunas", "linhas",
"total"), arred=3)
{
      if (margens && prop!="no") {stop("Não ajuda muito fazer proporções
com marginais... Tente de novo")}
      z <- table(...)
      nnm <- names(dimnames(x))
      if (prop=="colunas")
      {
            z <- prop.table(z,2)
      }
      if (prop=="linhas")
      {
            z <- prop.table(z,1)
      }
      if (prop=="total")
      {
            z <- z/sum(z)
      }
      if (margens && prop=="no")
      {
            z <- cbind(z, Total=apply(z,1,sum))
            z<-rbind(z, Total=apply(z,2,sum))
      }
```

```
        names(dimnames(z))<-nnm
        round(z, arred)
}
```

Não chega a ser um primor de função, mas dá para fazer algumas brincadeiras. Por exemplo, vamos tentar fazer uma tabela com as margens:

```
> tabela.2x2(menarche,tanner, margens=T)
         tanner
menarche   I II III IV   V Total
   Não   221 43  32 14   2   312
   Sim     1  1   5 26 202   235
   Total 222 44  37 40 204   547
```

Podemos também obter tabelas com proporções, tanto em relação às colunas como linhas e também com relação ao total geral da tabela. Vamo tentar para as colunas:

```
> tabela.2x2(menarche,tanner, prop="colunas")
         tanner
menarche     I    II   III   IV    V
     Não 0.995 0.977 0.865 0.35 0.01
     Sim 0.005 0.023 0.135 0.65 0.99
```

Repare que as colunas é que somam 1 e não as linhas. Você é capaz de sugerir como poderíamos obter percentagens em vez de proporções para estas saídas?

Gráficos

Ao contrário dos gráficos que vimos para variáveis contínuas, que são realmente muito bons tanto sob o ponto de vista de utilidade quanto sob o ponto de vista estético, já no caso das representações gráficas de tabelas, que seriam os gráficos tipo barra e pizza, que em geral precisam de legendas, o R fica um pouco mais complicado para obter-se um resultado satisfatório. Muitas vezes será mais útil exportar a tabela para uma planilha eletrônica. De qualquer forma, vamos ver como isso pode ser feito no R.

Vamos iniciar pelo gráfico de barras. Podemos usar a nossa função para tabelas 2x2 personalizadas para nos ajudar, por exemplo a fazer um gráfico de barras com as proporções de sexo e Classificação de Tanner em relação ao total. Vamos ver como fica, fazendo de duas maneiras diferentes e com barras verticais e horizontais:

```
      par(mfrow=c(2,2))
      barplot(100*tabela.2x2(sex,tanner, prop="total"), legend.text=T,
main="Distribuição do Sexo pela Classificação de Tanner", ylim=c(0,50), ylab="%
do Total")
      barplot(100*tabela.2x2(tanner,sex, prop="total"), legend.text=T,
main="Distribuição da Classificação de Tanner por Sexo", ylim=c(0,50), ylab="%
do Total", beside=T)
      barplot(100*tabela.2x2(sex,tanner, prop="total"), legend.text=T,
main="Distribuição do Sexo pela Classificação de Tanner", xlim=c(0,50), xlab="%
do Total", horiz = T)
      barplot(100*tabela.2x2(tanner,sex, prop="total"), legend.text=T,
main="Distribuição da Classificação de Tanner por Sexo", xlim=c(0,50), xlab="%
do Total", beside=T, horiz = T)
      par(mfrow=c(1,1))
```

Se você estiver curioso para saber o que realmente está sendo plotado, veja o resultado das tabelas, por exemplo, a primeira:

```
100*tabela.2x2(sex,tanner, prop="total")
```

Observe que o argumento `beside=T` faz com que as barras não sejam aglomeradas para a mesma categoria (o *default* é aglomerar) e que o argumento `horiz = T` faz com que as barras sejam horizontais (*default* vertical.)

A essas alturas, você deve estar se perguntando porque eu fui tão duro com esse tipo de gráfico no R... Bem, é que o controle da posição da legenda não é uma tarefa muito fácil e o R nem sempre acerta o local ideal para colocá-la. Por exemplo, tente fazer um gráfico de proporções por colunas:

```
barplot(100*tabela.2x2(sex,tanner, prop="colunas"), legend.text=T,
main="Distribuição do Sexo pela Classificação de Tanner", ylab="% sexo")
```

Entendeu o que eu quis dizer? A maneira de driblar o R seria aumentar os limites do eixo y:

```
barplot(100*tabela.2x2(sex,tanner, prop="colunas"), legend.text=T,
main="Distribuição do Sexo pela Classificação de Tanner", ylab="% sexo",
ylim=c(0,120))
```

Mas isso não é lá muito elegante, é?

Bem, o outro tipo de gráfico é o famoso gráfico de pizza (ou torta, depende do freguês...) Também é um gráfico meramente ilustrativo, e podemos por exemplo usá-lo para mostrar a mesma coisa que já mostramos com o gráfico de barras. Vamos a um exemplo:

```
par(mfrow=c(1,2))
pie(tabela.2x2(sex,tanner)["M",], main="Homens")
pie(tabela.2x2(sex,tanner)["F",], main="Mulheres")
par(mfrow=c(1,1))
```

Essas lindas cores podem ser alteradas, com o argumento `col`. Tente brincar com esses gráficos para ver se eles lhe agradam...

Exercícios

1. Para uma distribuição Normal, calcule a massa de probabilidade contida no intervalo interquartilar e também entre os limites superior e inferior de um *boxplot* (corresponderia à probabilidade de um ponto não ser um *outlier*.) Exemplifique com um gráfico.

2. Escolha um banco de dados qualquer – pode ser um banco com o qual você trabalha, ou um banco que você baixou da internet ou até mesmo um banco de exemplo do R, qualquer um. Apresente uma análise descritiva das principais variáveis encontradas nesse banco, como se fosse o material que você enviaria para uma publicação em uma revista. Note que não há nenhuma obrigação em se fazer este trabalho no R. Note também que o material a ser entregue é o mesmo que você mandaria para uma publicação, o que exclui completamente respostas escritas a mão.

Exercícios

Aula 2 – Estatística descritiva

Livro: páginas 57 a 80

1. Para uma distribuição Normal, calcule a massa de probabilidade contida no intervalo interquartilar e também entre os limites superior e inferior de um *boxplot* (corresponderia à probabilidade de um ponto não ser um *outlier*.) Exemplifique com um gráfico.

Bem, a primeira parte da pergunta é para ser feita de cabeça. A resposta é 0.5 e acho que dispensa maiores explicações. A segunda parte é mais complicada e devemos usar algum código para resolvê-la (claro que isso poderia ter sido feito à mão também.) Inicialmente vamos calcular os quantis de uma Normal (0,1) teórica que delimitam a distância interquartilar:

```
> qnorm(.25)
[1] -0.6744898
> qnorm(.75)
[1] 0.6744898
```

Como esperado, são valores simétricos. Para conferir, sabendo que teoricamente a densidade dessa área deve ser 0.5, vamos fazer:

```
> pnorm(qnorm(.75))-pnorm(qnorm(.25))
[1] 0.5
```

Confere. Agora vamos calcular a distância entre os quartis. A forma mais fácil é:

```
> 2*qnorm(.75)
[1] 1.348980
```

Em seguida, devemos multiplicar essa distância por 1.5:

```
> 1.5*2*qnorm(.75)
[1] 2.023469
```

E em seguida temos que subtrair esse valor do primeiro quartil e somá-lo ao terceiro quartil, ambos calculados acima, para obter os limites que estamos procurando:

```
> qnorm(.25)-1.5*2*qnorm(.75)
[1] -2.697959
> qnorm(.75)+1.5*2*qnorm(.75)
[1] 2.697959
```

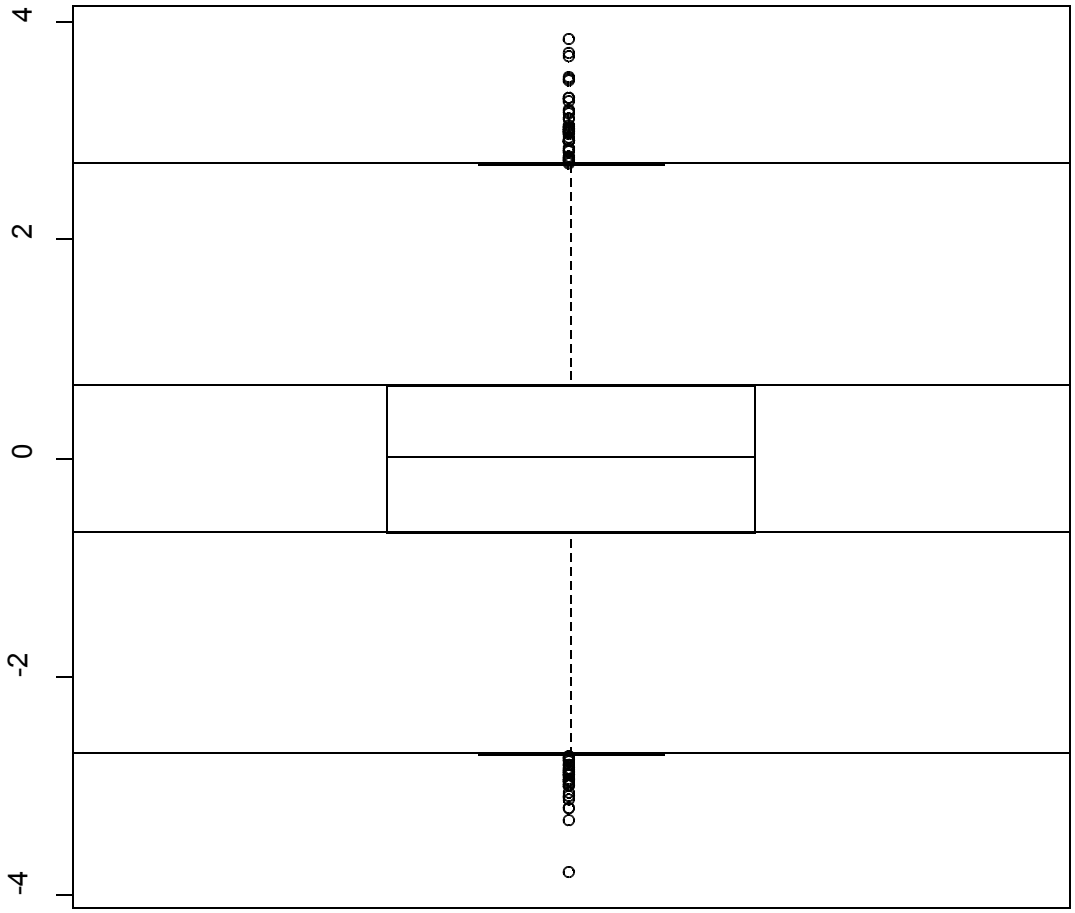
Novamente, como esperado, são valores simétricos. Agora basta calcular a área sob a curva entre esses limites

```
> pnorm(qnorm(.75)+1.5*2*qnorm(.75))-pnorm(qnorm(.25)-1.5*2*qnorm(.75))
[1] 0.9930234
```

O gráfico deve ser obviamente um *boxplot*, que para ilustrar essas questões poderia ser feito com uma simulação de muitos pontos de uma Normal (0,1) e poderíamos traçar linhas horizontais demarcando os limites que calculamos sobre o *boxplot.* Algo assim:

```
boxplot(rnorm(10000))
```

```
abline(h=qnorm(.25))
abline(h=qnorm(.75))
abline(h=qnorm(.25)-1.5*2*qnorm(.75))
abline(h=qnorm(.75)+1.5*2*qnorm(.75))
```



Notou algo de anormal com esse *boxplot*? Alguma explicação para isso? Quem me entregar essa resposta certa e **por escrito na próxima aula**, ganhará 0.1 de bônus.

2. Escolha um banco de dados qualquer – pode ser um banco com o qual você trabalha, ou um banco que você baixou da internet ou até mesmo um banco de exemplo do R, qualquer um. Apresente uma análise descritiva das principais variáveis encontradas nesse banco, como se fosse o material que você enviaria para uma publicação em uma revista. Note que não há nenhuma obrigação em se fazer este trabalho no R. Note também que o material a ser entregue é o mesmo que você mandaria para uma publicação, o que exclui completamente respostas escritas a mão.

Evidentemente esse exercício não tem uma resposta fixa. Ele consiste em escolher um banco de dados qualquer e fazer uma descrição de variáveis contidas nesse banco, como se fosse parte dos resultados que você entregaria para publicação, o que significa que a qualidade gráfica é importante também. Basta usar as saídas de qualquer programa. Por exemplo, usando o R para descrever o banco `juul`:

Por exemplo, para uma descrição univariada das variáveis contínuas, poderíamos usar a função `eda()`:

Figura 1 – Resumo Estatístico da Quantidade de IGF-I

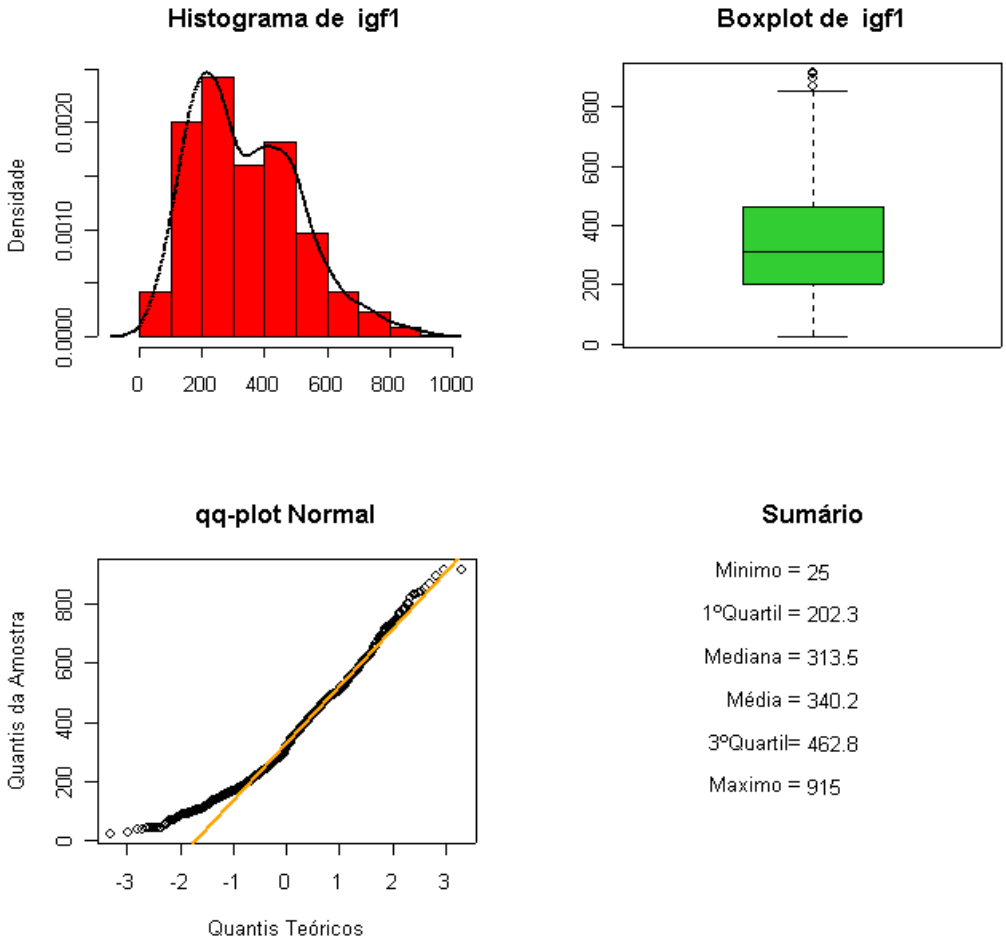
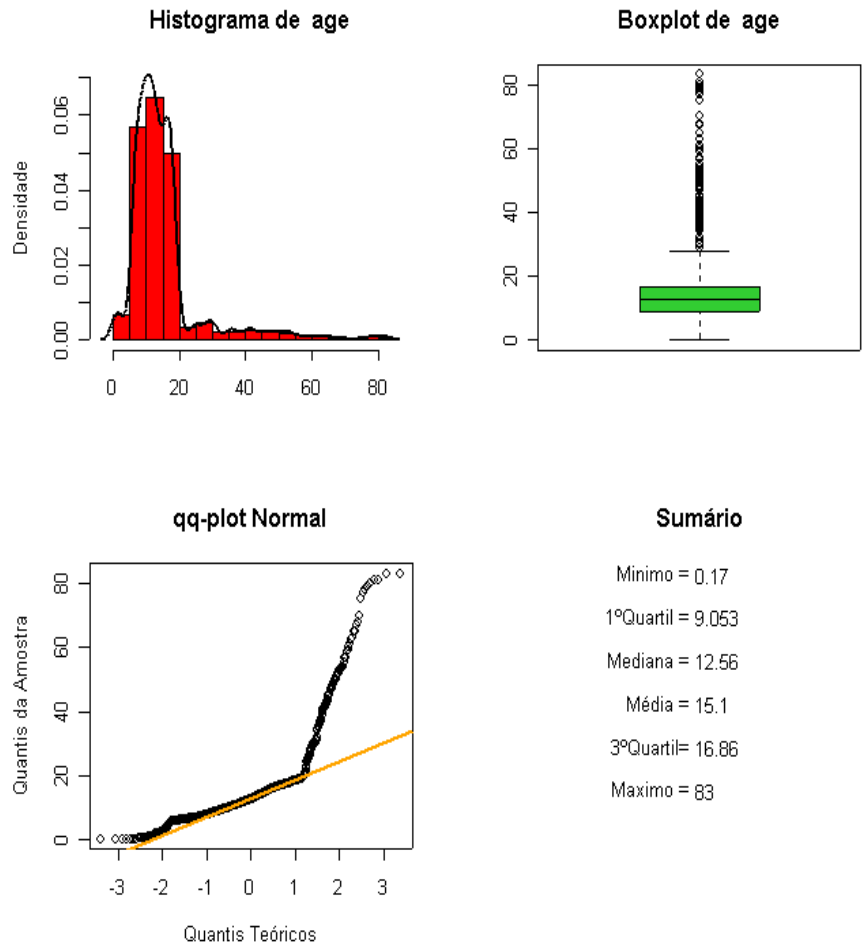
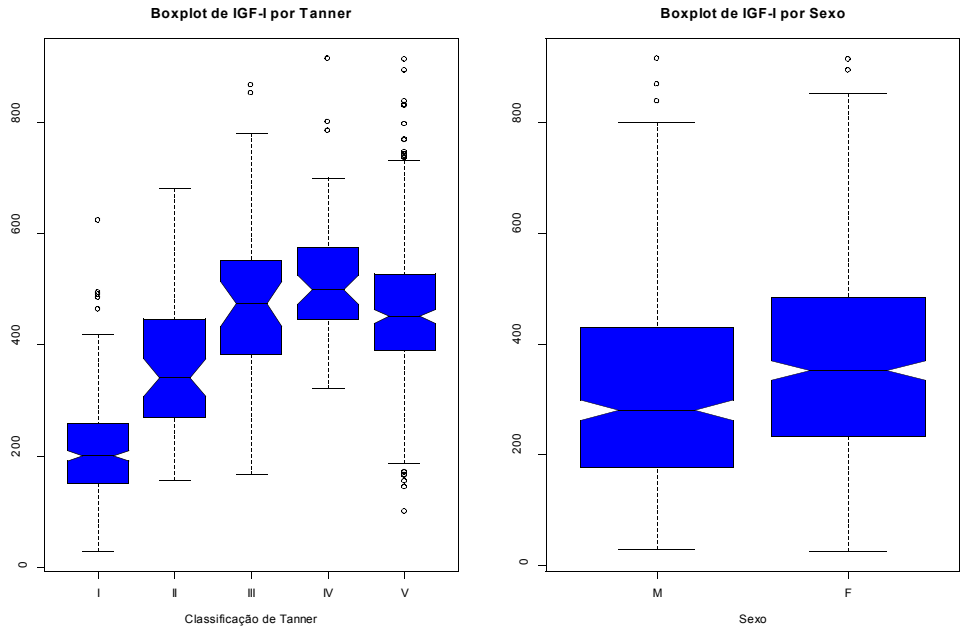


Figura 2 – Resumo Estatístico da Idade dos Participantes



Poderíamos também usar comparações por categorias. Como o nosso interesse neste banco é a IGF-I, podemos usar Box-plots para comparar:



Claro que poderíamos usar tabelas também:

Quadro 1 – Resumo Estatístico da IGF-I por Categorias de Sexo e Classificação de Tannner

Variável	*Categoria*	*Mínimo*	*Máximo*	*Mediana*	*Média*	*DP*
Sexo	Masculino	29	915	280	310.9	169.71
	Feminino	25	914	352	368.1	167.35
Tanner	I	29	624	201	207.47	90.27
	II	157	682	341.5	352.67	122.59
	III	167	868	474	483.22	152.29
	IV	321	915	500	513.02	119.1
	V	100	914	452	465.33	134.42

Enfim, são apenas exemplos. Você pode inventar a apresentação que quiser!!!

Claro que isso deve ser acompanhado de uma descrição propriamente dita desses gráficos e tabelas. Deixo isso para um exercício extra para você... Quem me entregar essa resposta certa e **por escrito até Sexta-Feira que vem**, ganhará 0.1 de bônus. Obs.: Nesse caso, por escrito não é escrito a mão!!!

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 3 – Distribuições amostrais
Livro: NA

Mistérios da Estatística
- O TLC
- A distribuição amostral da média e o Intervalo de Confiança
- As variâncias da população e da amostra

Exercícios

O conteúdo desta aula não está disponível no nosso livro texto de referência. Nós achamos, porém, que ele vai nos ajudar a explicar melhor alguns conceitos, fundamentais para se entender com mais clareza questões de inferência e testes de hipóteses que serão vistos mais adiante, usando o que nós já vimos como base para esta aula.

Uma primeira questão que deve surgir muito cedo na cabeça de qualquer pessoa que começa a aprender estatística é como é possível acreditar no Teorema do Limite Central (TLC) sem nunca ter visto como ele funciona na prática. Bem, quando você lê em um livro de estatística que a distribuição da média amostral de uma população normal é normal, acredito que intuitivamente você concorde (sem talvez entender muito bem o que isso quer dizer), mas e quando você lê que a distribuição da média amostral de qualquer distribuição converge assintoticamente (na verdade converge em distribuição, mas isso é outro assunto...) para uma normal, eu não creio que esse conceito fique entendido e assimilado na cabeça de quem lê.

A primeira questão aqui é entender que, apesar da média amostral ser bem intuitiva e, como você deve se lembrar, o seu cálculo é feito da mesma maneira que o cálculo da média da população, ela é na verdade uma variável aleatória (e não um número!!!) e que portanto ela tem uma determinada distribuição. A sua próxima pergunta é: mas então quando eu calculo a pressão arterial média dos meus pacientes, essa medida não é um número???

Vamos por partes... É um número, sem dúvida, mas ele representa a média de uma única amostra da sua população e a ele estará associado uma probabilidade de estar suficientemente próximo da verdadeira média da população (a qual eu nunca conhecerei com certeza, a menos que realize um censo nesta população). É por causa desta noção – que eu usei de forma livre aqui e espero que nenhum estatístico esteja por perto, pois ele pode ter um troço – que nós usamos sempre um estimador intervalar (o nosso conhecido intervalo de confiança.)

Como vocês devem saber, o intervalo de confiança representa um intervalo numérico tal que, se nós retirássemos k amostras de tamanho n de uma população, para um k suficientemente grande, aproximadamente 95% dos k intervalos, calculados a partir de cada uma das amostras, conteriam o verdadeiro valor da média da população (claro que você já concluiu que isso é para um intervalo de confiança 95% da média, né?)

Ah, e tem ainda a misteriosa variância da amostra, que por incrível que pareça não é estimada corretamente pela mesma fórmula usada para a variância da população, mas que como a média também é uma variável aleatória (alguém lembra qual a sua distribuição???)

O TLC

Bom, chega de papo e vamos tentar entender alguns desses mistérios com a ajuda do R. Vamos começar pelo TLC. Vamos verificar que, dada uma distribuição *qualquer*, a distribuição da média amostral converge para uma distribuição normal, com média $\bar{x}$, igual à média da população e variância σ^2/n, ou seja, a variância da população sobre o tamanho da amostra. Bem, vamos ver como isso funciona, então?

A idéia inicial é gerar uma amostra de uma distribuição qualquer e depois usar uma função que você já deve ter visto para retirar amostras dessa distribuição, que será considerada a minha população sobre a qual queremos fazer inferências. Para visualizarmos a distribuição da média amostral, não precisamos retirar todas as possíveis amostras (até porque seria muito trabalhoso, por exemplo obter todas as possíveis amostras de $n = 30$ de uma população de $N = 1000$. Isso daria nada menos que $\binom{1000}{30}$ – experimente fazer a conta no R com a função:

```
choose(1000, 30)
```

Grandinho o número, né?

;-)

E olha que isso seria sem reposição!!! Na verdade a toda a teoria amostral é desenvolvida para casos com reposição. Alguém se arrisca dizer qual seria o número total de amostras com reposição neste caso? Pois é: 1000^{30} ou 10^{32}, diria que bastante trabalhoso...

Em vez disso, vamos retirar umas 500 amostras mais ou menos e ver o que acontece com esta distribuição. Para facilitar a nossa vida, vamos usar uma função que eu inventei, e vamos ver se conseguimos entender o que se passa:

```
histo.mean <- function(x, n=2, b=500)
#x é um vetor com uma distribuição populacional; n é o tamanho de cada amostra e b o número de amostras
{
        z <- mat.or.vec(0,1)   #Inicialização do vetor z
        for (i in 1:b) #Loop para obter as médias das b amostras
        {
                z[i] <- mean(sample(x,n))     #O verdadeiro truque
        }
        w <- list("n" = n, "mean.pop"=mean(x), "var.pop" = (length(x)-1)*var(x)/length(x))
#Adicionando a média e a variância da população
        w$mean.sampl <- mean(z) #Acrescentando a média amostral
        w$var.mean.sampl <- var(z)    #Acrescentando a variância amostral
        hist(z, main="Histograma da média amostral de x")    #Fazendo o Histograma
        w
}
```

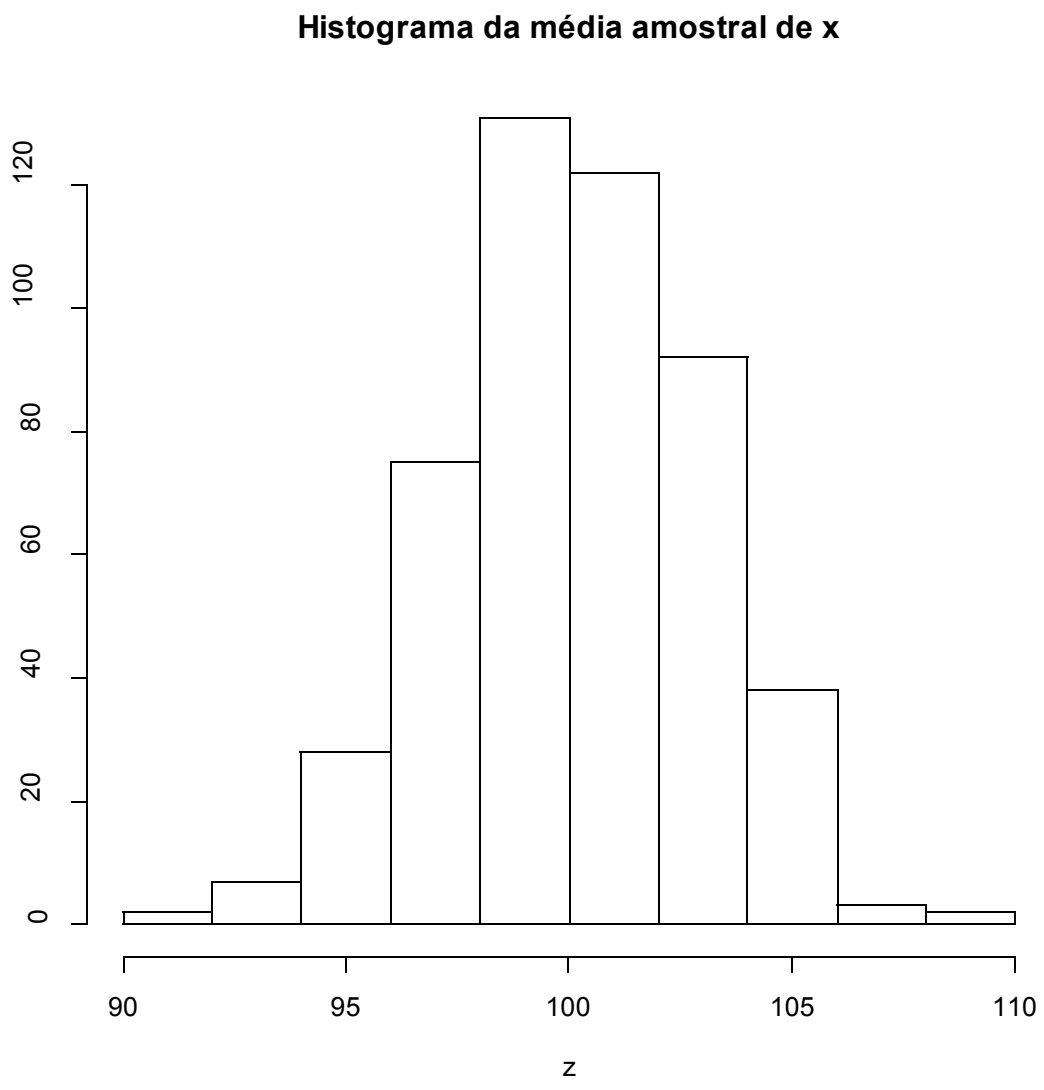
Sinto que neste momento o desespero tomou conta de você... Calma, nem tudo está perdido... Eu coloquei alguns comentários (tudo o que for seguido do símbolo #) que explica cada um dos passos dessa função. Tente entender o que está acontecendo. Em resumo, essa função pega um vetor *x* com uma distribuição qualquer, tira *b* amostras (500 por *default*) de tamanho *n* (2 por *default*.) Com isso, a função calcula a média dessas 500 amostras e as armazena num vetor *z*. Então uma lista é criada com algumas informações úteis tanto com respeito à população quanto às amostras (experimente a função mais tarde para entender melhor.) Por último, um histograma das médias amostrais é desenhado.

Você já deve saber que para utilizar esta função, basta marcá-la, copiá-la e colá-la no *prompt* do R. Bem, então vamos começar a brincar. Vamos "fabricar" uma distribuição, digamos normal para começar... Vamos assumir que temos 1000 idosos, dos quais nós queremos estudar a pressão arterial média (PAM), onde a média é em torno de 100 mmHg e a variância é de 16 $mmHg^2$:

```
x <- rnorm(1000, mean=100, sd=4)
```

Agora vamos usar a nossa função para construir uma distribuição de 500 amostras dessas 1000 PAMs d $n = 2$ (é isso mesmo, a nossa amostrinha é só de 2):

```
> histo.mean(x)
$n
[1] 2

$mean.pop
[1] 100.078

$var.pop
[1] 16.74125

$mean.sampl
[1] 100.1585

$var.mean.sampl
[1] 8.1367
```



Você deve ter notado que além dessa saída acima (que deve ser diferente para você, é claro, já que a nossa população foi gerada aleatoriamente!!!), um histograma também foi gerado, como você deve ter visto na sua sessão do R. Surpreso com os resultados? É isso mesmo: ainda que a nossa amostrinha tenha sido de apenas 2 observações, a sua distribuição é bastante normal, sua média é bastante próxima da média da população (compare a saída `mean.pop` e `mean.sampl` acima.) Além disso, a sua variância (da média) é bastante próxima da metade da variância da população (ou seja, a variância da população dividida pelo tamanho da amostra.)

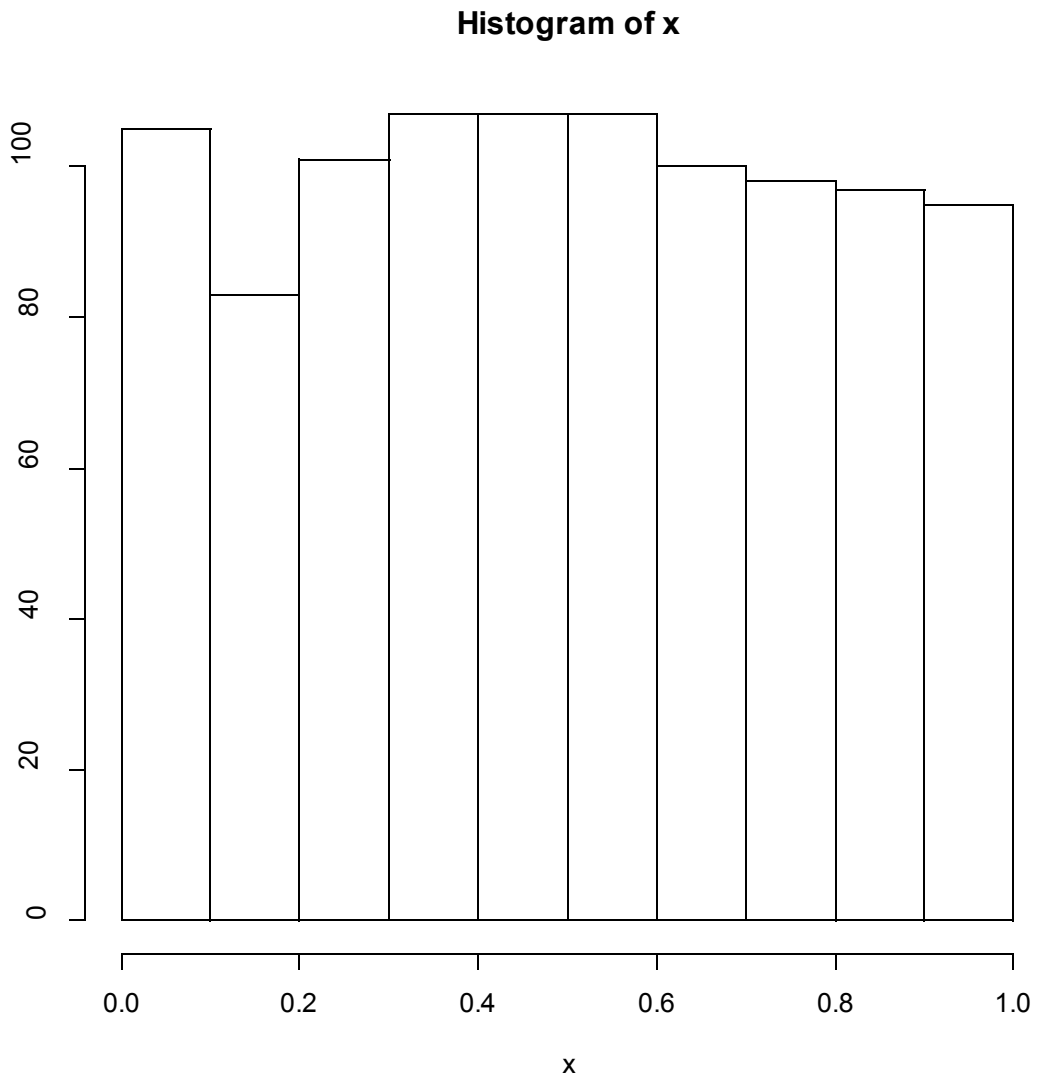
Bem, mas por enquanto não teve graça, né? A população já tinha uma distribuição normal, então não surpreende muito que a distribuição da média amostral seja normal também...

Vamos brincar um pouco agora com distribuições populacionais bem diferentes da normal. Que tal começarmos pela uniforme?

```
x <- runif(1000)
```

Como você deve se lembrar, esta distribuição é muito diferente da normal. Confira:

```
> hist(x)
```



Veja agora o que acontece com a distribuição amostral da média com o $n = 2$:

```
histo.mean(x)
```

Bom, agora eu acho que você ficou surpreso: aposto como o resultado foi muito parecido com o da população com distribuição normal... Na verdade, existe um motivo para isso, mas a explicação foge ao escopo deste material.

Vamos radicalizar então e simular uma distribuição bastante assimétrica, e vamos aproveitar para usar uma distribuição discreta. Sugestões? Que tal uma Binomial com $p = 0.2$ e $n = 1$? Não confunda este n com o tamanho da amostra: aqui trata-se do parâmetro da Binomial, significando que estamos realizando 1 experimento apenas e medindo o número de sucessos – ou seja, o vetor gerado será composto de números 0 e 1.) Lembra o nome dessa Binomial especial?

```
x <- rbinom(1000, p=0.2, size=1)
```

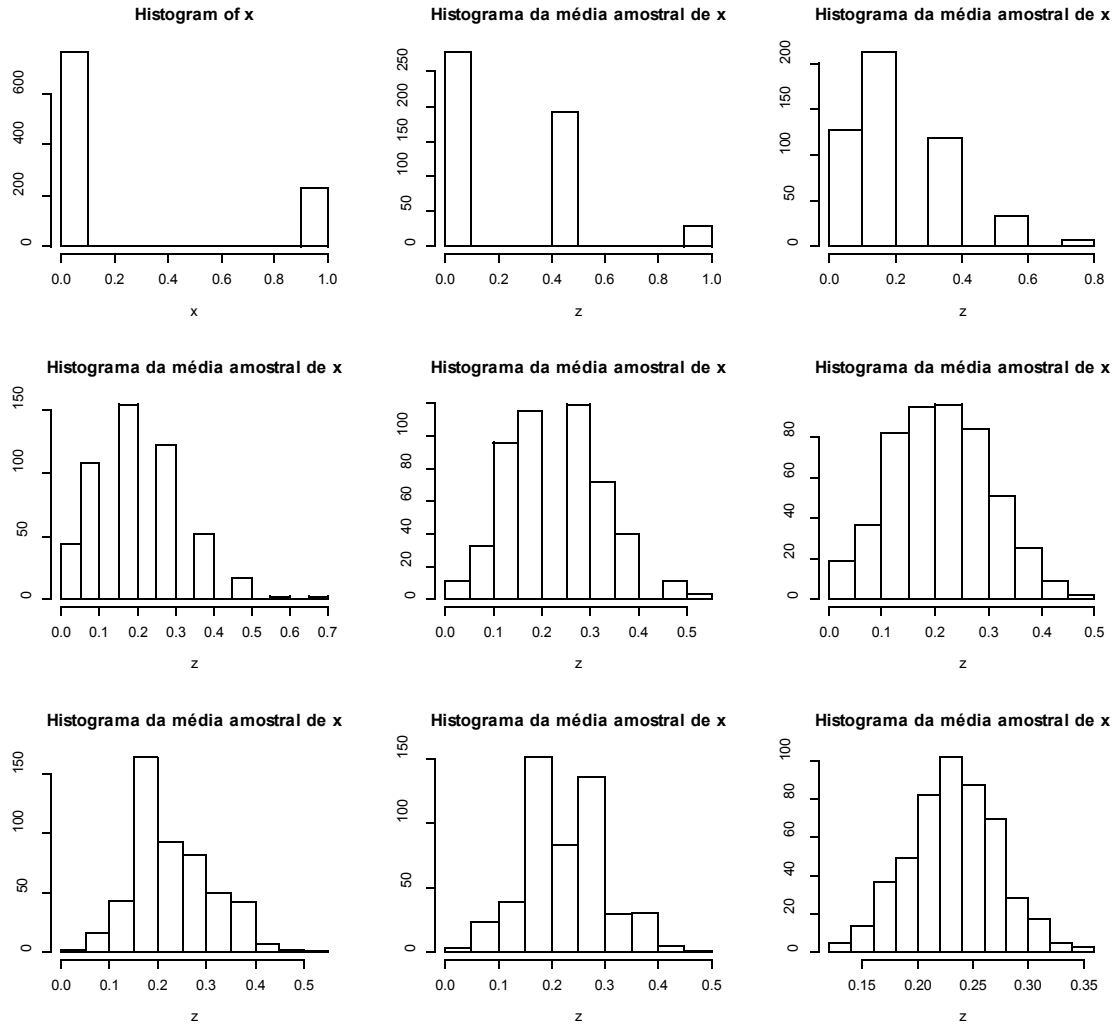
Confira o histograma desta distribuição. Observe como ela é assimétrica, e que ela só assume 2 valores inteiros. Faça:

```
table(x)
```

Essa poderia ser a distribuição de um desfecho binário qualquer do tipo doente/sadio, por exemplo. Vamos fazer então a seguinte brincadeira: vamos usar a nossa função mágica `par` para plotarmos 9 gráficos em uma mesma janela. O primeiro vai ser o histograma da distribuição de `x` e os outros, histogramas amostrais da média com tamanhos da amostra crescentes:

```
par(mfrow=c(3,3))
hist(x)
histo.mean(x)
histo.mean(x,5)
histo.mean(x,10)
histo.mean(x,15)
histo.mean(x,20)
histo.mean(x,25)
histo.mean(x,30)
histo.mean(x,100)
par(mfrow=c(1,1))
```

Além da saída que nós já discutimos, o gráfico final que eu obtive é mostrado abaixo – apesar da falta de legenda para saber qual é o n para cada um deles, você deve ter decifrado isso pelo código acima. Veja o que aconteceu: Ao aumentarmos o tamanho da amostra, a distribuição da média amostral foi se aproximando de uma distribuição normal. Aproveite para conferir também as médias e variâncias dessas distribuições. Veja se também não seguem a nossa regrinha do TLC.



Achou pouco os exemplos? Não se preocupe, tem um exercício bem legal sobre isso para você treinar e se divertir...

Muito bem. Até agora mostramos como o TLC funciona na prática, mas ainda falta mostrar, para a média, como funciona essa história do intervalo de confiança para a média, que é justamente baseado nesse teorema.

A distribuição amostral da média e o Intervalo de Confiança

Vamos agora trabalhar com uma situação menos real, mas que nos permitirá tirar algumas conclusões importantes a respeito ainda da média amostral, e também do seu intervalo de confiança, fazendo as distribuições completas da média amostral – claro que agora precisamos de uma população e amostras menores para não termos que lidar com números gigantescos.

Digamos que nós queiramos a partir de uma população de 10 notas de alunos, estimar a média da turma com amostras de 2 ou de 3 alunos apenas. Vamos assumir que essas notas têm distribuição normal, com média aproximadamente 7 e desvio-padrão de 1. Como somos preguiçosos, vamos usar o R para nos fazer uma distribuição assim:

```
notas <- round(rnorm(10,7,1),1)
notas
[1] 6.8 5.0 7.4 6.3 7.2 7.1 7.0 6.4 9.4 7.5
```

Note que como eu gerei essas notas, não necessariamente elas terão média 7 e DP de 1. Vamos ver o que temos nesse caso:

```
> mean(notas)
[1] 7.01
> sqrt((length(notas)-1)*var(notas)/length(notas))
[1] 1.053992
```

Para minha sorte, até que ficou bem perto... mas isso não importa, pois agora a média da minha POPULAÇÃO é 7.01 e o DP é 1.05.

Você reparou que eu não usei a função `sd()` disponível no R, não é mesmo? Isso porque por *default* o R calcula tanto a variância quanto o desvio padrão de uma amostra e não de uma população – você lembra a diferença? Não? Não faz mal, vamos falar bastante sobre isso mais tarde, mas por ora, vamos logo usar uma função para calcular a variância da população que eu gerei, e que vai facilitar a nossa vida bastante:

```
var.pop <- function (x)
{
    sum((x-mean(x))^2)/length(x)
}
```

Confira se o resultado é o mesmo:

```
> sqrt(var.pop(notas))
[1] 1.053992
```

Acho que não preciso lembrar que o desvio-padrão é a raiz quadrada da variância (a raiz positiva, por convenção.) Não se preocupe em entender o código ainda, que ele será melhor explicado mais adiante.

Se você quiser gerar a sua própria população de notas, apenas copie o código acima, mas se você quiser seguir os exemplos exatamente como eu vou escrever aqui, copie o vetor:

```
notas <- c(6.8, 5.0, 7.4, 6.3, 7.2, 7.1, 7.0, 6.4, 9.4, 7.5)
```

Vamos começar a nossa brincadeira tomando todas as possíveis amostras de tamanho 2 e 3 com reposição nesta população. Quantas amostras são possíveis?

Bem, de tamanho 2, são possíveis 100 amostras, o que corresponde a todas as permutações possíveis 2 a 2 de 10 observações.

No caso do tamanho 3, teremos 1000 amostras, também correspondendo a todas as possíveis permutações 3 a 3 de 10 observações.

Repare que estamos fazendo a nossa brincadeira a partir de uma amostragem com reposição, pois assim é que é desenvolvida toda a teoria amostral, e não como se faz geralmente na prática, onde não iremos entrevistar um mesmo paciente duas vezes, por exemplo... Mais tarde, veremos qual a conseqüência dessa mudança.

Bem, vamos usar a ajuda do R para gerar as amostras e você poderá verificar os valores mais tarde. Para isso vamos usar uma função que calcula essas permutações automaticamente:

```
permuta2.ou.3 <- function (x, N, n)
{
       if (n!=2 && n!=3) stop("Esta função é para obter permutações 2 a 2 ou 3 a 3 apenas")
       z <- matrix(0, nrow=N^n,ncol=n)
       z[,1] <- rep(x,each=N^(n-1))
       z[,2] <- rep(x, times=N^(n-2), each=N^(n-2))
       if (n==3)
       {
              z[,3] <- rep(x, times=N^(n-1))
       }
       z
}
```

Bem, agora basta você fazer:

```
amostras.2 <- permuta2.ou.3(notas, 10,2)
amostras.3 <- permuta2.ou.3(notas, 10,3)
```

Com isso, criamos uma matriz 100x2 e outra 1000x3, sendo que cada linha dessas matrizes é uma das combinações de amostras de tamanho 2 e 3, respectivamente. Não entendeu? Então verifique o conteúdo do objeto `amostras.2` e `amostras.3`!

Bem, agora que temos todas as possíveis amostras, vamos calcular a média de cada uma dessas amostras... Se você está pensando que vai ter que calcular cada média na mão, enganou-se. Vamos usar uma função especial, bem parecida com uma outra função que nós já utilizamos (lembra?), para calcular a média de cada uma das linhas da nossa matriz (que são as amostras):

```
medias.2 <- apply(amostras.2,1,mean)
medias.3 <- apply(amostras.3,1,mean)
```

Aproveite para conferir se esses objetos contêm o esperado...
Vamos ver qual é a média e a variância dessas médias amostrais:

```
> mean(medias.2)
[1] 7.01
> mean(medias.3)
[1] 7.01
> > var.pop(medias.2)
[1] 0.55545
> var.pop(medias.3)
[1] 0.3703
```

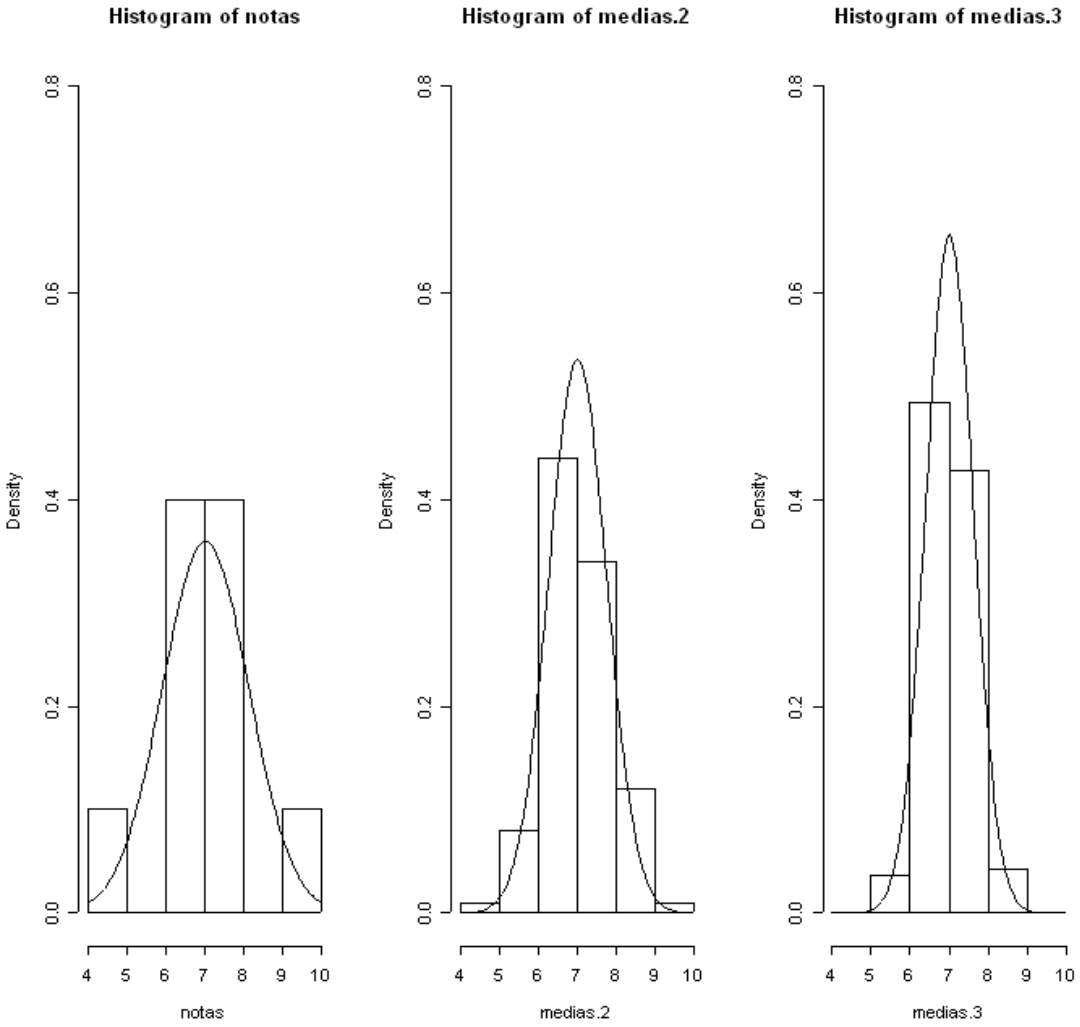
Alguma surpresa? Confira... Veja que agora os números são exatos e não mais aproximados.
Bem, agora vamos comparar a distribuição desta população de notas com as suas distribuições amostrais da média. Vamos fazer isso de uma maneira um pouco elaborada, com o código abaixo, cuja explicação será deixada como exercício:

```
par(mfrow=c(1,3))
hist(notas, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,1.111006), from=4, to=10, add=T)
hist(medias.2, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/2)), from=4, to=10, add=T)
hist(medias.3, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/3)), from=4, to=10, add=T)
par(mfrow=c(1,1))
```

Vamos ver o resultado:

Ainda nenhuma surpresa: nada diferente do que já havíamos feito com as amostras anteriormente, ou seja as distribuições amostrais da média são normais, assim como a população e sua variância diminui quando aumentamos o tamanho da amostra.

Tudo muito bonito, mas agora você deve estar se perguntando: mas acontece que eu não tiro todas as amostras possíveis de uma população (seria mais fácil fazer um censo, bolas!) O que nós fazemos é tirar UMA ÚNICA amostra de tamanho 2 (e não 100 amostras) ou de tamanho 3 (e não 1000 amostras.)



E tem mais: você deve estar pensando também no seguinte: como eu só tiro UMA amostra, eu obterei UMA dessas médias... Bem nós vimos que a média das médias amostrais é exatamente igual à média da população, mas e a média que eu obterei da minha única amostra? Qual será a chance dela ser igual à média da população? Isso é fácil de calcular. Basta nós inspecionarmos o vetor de médias e ver quantas delas são iguais à média da população de notas. Depois, basta dividir pelo total de médias (100 para amostra de 2 e 1000 para amostra de 3.) Vamos usar um código que nos diz em que posição em um vetor se encontra um valor que queremos procurar:

```
> which(medias.2==mean(notas))
numeric(0)
> which(medias.3==mean(notas))
numeric(0)
```

EPA!!! Esse resultado está indicando que NENHUMA das médias das amostras de tamanhos 2 e 3 são iguais à média da população???!!!

Nem precisa tanta surpresa: ninguém disse que a média de uma amostra será igual à média da população. Aliás, para os nosso exemplos, não é possível obter-se a verdadeira média de jeito nenhum... O que nós garantimos é que em média esses valores são iguais à média da população. Pode até parecer brincadeira, mas não é...

Mas então, como podemos saber se a média que estamos obtendo de uma amostra está pelo menos próxima da verdadeira média? De que me adianta todo esse trabalho, se eu não puder inferir algo a esse respeito?

Bem, é aí que temos que lançar mão dos intervalos de confiança (IC), que representam um intervalo numérico, construído a partir da amostra, através do qual poderemos ter uma idéia de o quanto próximo da verdadeira média a nossa média amostral pode estar.

Nós vamos tentar demonstrar como isso funciona, através da própria definição de um intervalo de confiança. Por exemplo, a definição de um IC 95% para a média é a seguinte: se nós retirássemos *k* amostras de tamanho *n* de uma população, para um *k* suficientemente grande, aproximadamente 95% dos *k* ICs, calculados a partir de cada uma das amostras, conteriam o verdadeiro valor da média da população.

Duas observações podem ser feitas. A primeira é que nada é dito sobre *n* nesse caso, isto é, essa definição não depende do tamanho da amostra, ela será válida para qualquer tamanho (embora para o caso de variância da população desconhecida, *n* deve ser no mínimo 2 – você sabe explicar porque?)

A segunda observação é sobre a afirmação "para um *k* suficientemente grande". Essa forma mais abrangente serve para incluir populações consideradas infinitas, onde evidentemente o número de amostras possíveis será igualmente infinito. Você já notou que para o nosso exemplo aqui das notas, essa população é finita, e fomos inclusive capazes de tirar TODAS as amostras possíveis de tamanhos 2 e 3.

Vamos começar então a nossa brincadeira, e para facilitar a nossa vida, vamos trabalhar com o nosso velho conhecido de IC 95% para a média.

Você se lembra como se calcula um IC 95% para a média, usando a distribuição Normal? Aliás, por que mesmo posso usar a Normal aqui e posso dispensar a distribuição *t* de Student?

Bom, vamos refrescar o cálculo:

$$\bar{x} \pm z_{1-\alpha/2} \times \sqrt{\sigma^2/n}$$

Bem, para demonstrarmos como o IC funciona, segundo a definição que conhecemos, poderíamos construir todos os 100 IC 95% possíveis para todas as amostras de tamanho 2 e comparar isso com a verdadeira média da população (lembrando que isso nunca mais vai acontecer na sua vida profissional!!!)

Vamos usar o seguinte código:

```
plot(medias.2,rep(mean(medias.2),100), type="b", ylim=c(0,15))
arrows(medias.2, medias.2+(qnorm(0.975)*sqrt(var.pop(notas)/2)), medias.2,
medias.2-(qnorm(0.975)*sqrt(var.pop(notas)/2)), angle=90, code=3, length=0.1)
points(medias.2, medias.2, pch=19)
```

O primeiro comando, como você deve ter facilmente decifrado apenas vai desenhar o gráfico das médias de cada uma das amostras contra um valor fixo, que é a média total de todas as médias amostrais, feito pela função `rep()`, que repete esse número 100 vezes. A opção `type="b"` vai forçar o aparecimento de pontos ligados por uma linha.

O segundo comando acrescenta a esse gráfico pontos do tipo "alto e baixo", que é na verdade os limites superiores e inferiores de cada um dos ICs para cada uma das amostras.

Finalmente o último comando insere no gráfico as médias de cada uma das amostras.

Sem entrar em detalhes como funciona essa função `arrows()`, vamos nos deter um pouco mais no código que foi usado para criar o intervalo o gráfico:

```
medias.2+(qnorm(0.975)*sqrt(var.pop(notas)/2))
```

```
medias.2-(qnorm(0.975)*sqrt(var.pop(notas)/2))
```

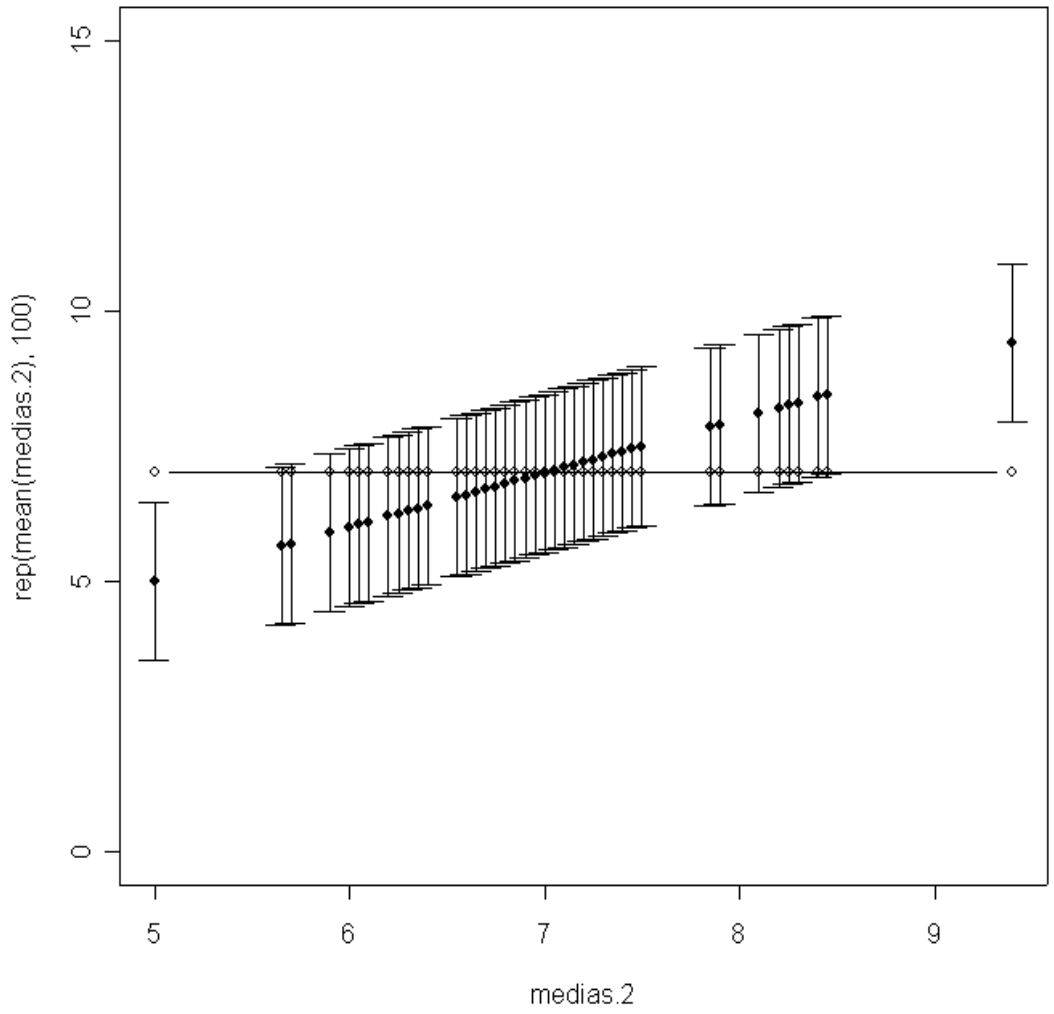
e comparar com a a equação que nós conhecemos

$$\bar{x} \pm z_{1-\alpha/2} \times \sqrt{\sigma^2 / n}$$

Acho que não há dúvida quanto a se escrever esta equação em suas linhas, separando o "+" do "-". O que pode ser um pouco confuso se você se esqueceu da capacidade do R de fazer operações vetoriais é o que nós estamos fazendo com o vetor `medias.2`. Como nós precisamos de todos os 100 ICs nesse caso, nós vamos aplicar a soma e a subtração da parcela da direita (que nesse caso são números fixos e simétricos) ao vetor inteiro, o que vai nos dar os 100 intervalos de confiança (lembre-se que a operação nesse caso é feita elemento-a-elemento).

A parcela da direita começa com o valor de *z* para o qual a área da normal é 1- α . No R a função que calcula esse quantil é a `qnorm()`, como você deve se lembrar. Como nosso nível de confiança é 0,05, usamos o valor 0,975. Por fim, esse valor é multiplicado pela raiz quadrada do quociente da variância pelo tamanho da amostra.

O gráfico é mostrado abaixo. Que conclusões você tiraria dele, em respeito ao nosso IC 95%?



Você deve ter notado que na verdade não aparecem neste gráfico os 100 ICs que nós esperávamos... O que será que aconteceu? Uma possibilidade é que tenhamos na verdade vários intervalos sobrepostos uns aos outros. Vamos criar uma matriz com os valores, para podermos inspecionar o que está acontecendo. Faça assim:

```
ic.2<-cbind(medias.2-(qnorm(0.975)*sqrt(var.pop(notas)/2)), medias.2,
medias.2+(qnorm(0.975)*sqrt(var.pop(notas)/2)))
ic.2[order(ic.2[,2]),]
```

O resultado é simplesmente uma matriz onde a primeira coluna é o limite inferior, a segunda a média e a terceira o limite superior do IC. Além disso, a matriz foi ordenada ascendentemente pela coluna do meio (pela média.) Verifique se de fato não existem linhas rigorosamente iguais...

Mas agora você deve estar se perguntando se não há mais ICs que não contenham a média... Quem sabe esses dois que estamos visualizando não estão também sobrepostos? Para tirar essa dúvida, vamos usar um código para saber quantos ICs não contêm a média:

```
length(c(which(mean(notas)<ic.2[,1]), which(mean(notas)>ic.2[,3])))
```

Se tudo funcionou, o resultado foi 2, ou seja apenas aqueles dois intervalos mesmo não continham a média da população. Repare que nesse exemplo, 98% dos ICs contêm a média (e não 95%) Isso não chega a causar surpresa já que a garantia é de **aproximadamente** 95% e não exatamente 95%

Bem, é claro que essa situação de conhecer a variância da população nunca vai acontecer na vida real, e nós teremos que usar um estimador para esta variância. Como você deve se lembrar o estimador indicado neste caso é s^2. Lembra como se calcula?

$$s^2 = \frac{\sum_{i=0}^{n} \left(x_i - \bar{x} \right)^2}{n-1}$$

Olha aí um dos nossos mistérios que nós vamos falar mais tarde: por que esse *n-1* no denominador?

De qualquer maneira, neste caso o desvio-padrão usado não será constante como no caso de usarmos a normal, para variância conhecida e cada amostra terá a seu próprio s^2 para ser calculado

Vamos usar este código para gerar os desvios-padrão de todas as amostras:

```
sem.2<-apply(amostras.2,1,var)
sem.2<-sqrt(sem.2/2)
sem.3<-apply(amostras.3,1,var)
sem.3<-sqrt(sem.3/3)
```

Agora vamos fazer algo bem semelhante ao que fizemos anteriormente, mas usando agora s^2 para calcular o IC 95%, usando para isso a distribuição *t*. A equação, bastante semelhante será:

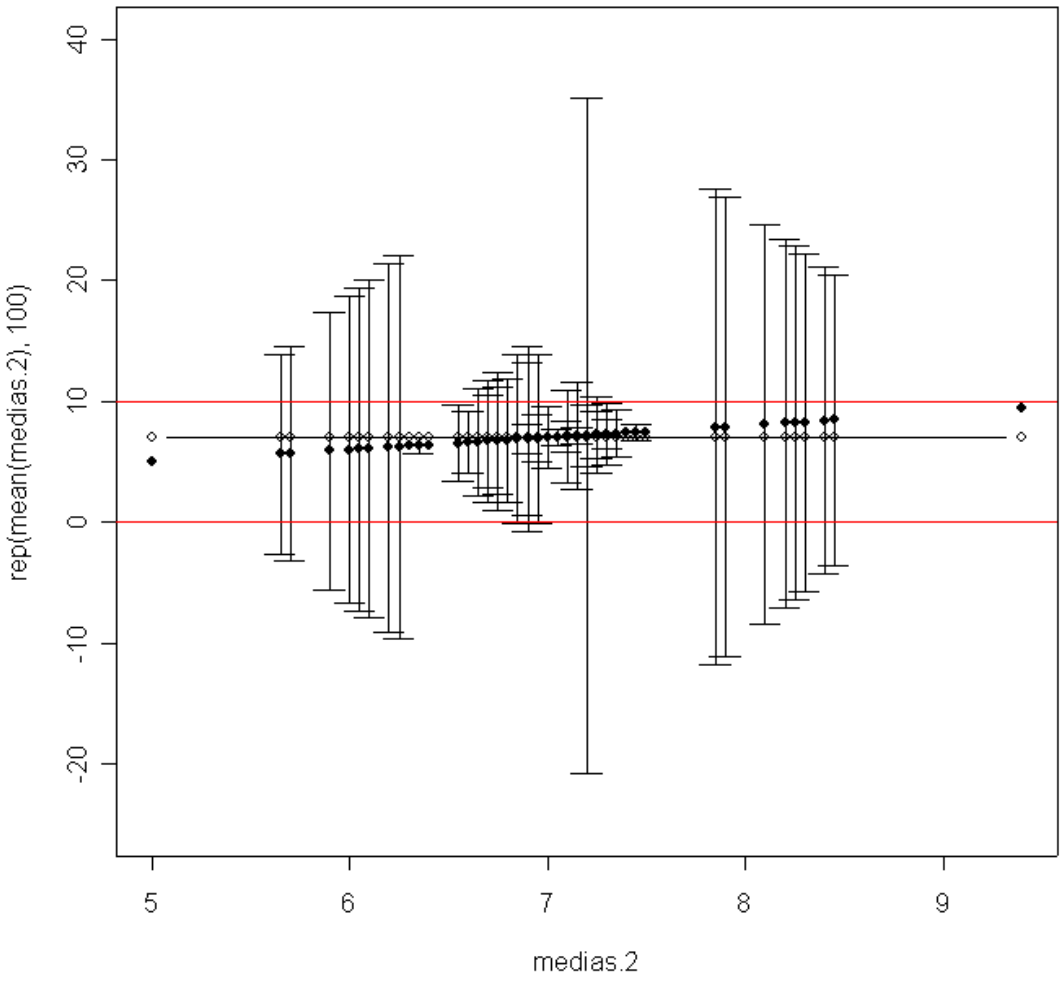
$$\bar{x} \pm t_{n-1,1-\alpha/2} \times \sqrt{s^2/n}$$

Vamos ver como fica, com o código abaixo:

```
plot(medias.2,rep(mean(medias.2),100), type="b", ylim=c(-25,40))
arrows(medias.2, medias.2+(qt(0.975,1)*sem.2), medias.2, medias.2-
(qt(0.975, 1)*sem.2), angle=90, code=3, length=0.1)
points(medias.2, medias.2, pch=19)
abline(h=0, col="red")
abline(h=10, col="red")
```

A diferença é que agora temos que usar a distribuição *t* de Student com *n*-1 graus de liberdade em vez da normal. Outras diferenças são os limites do eixo *y* e duas linhas horizontais acrescentadas nos pontos *y* = 0 e *y* = 10. Olha o bicho aí embaixo...

Notou algo de diferente? Você tem alguma explicação para isso?



Você deve estar se perguntando agora por que é que fizemos isso tudo para uma amostra de tamanho 3 e não usamos ainda... Isso mesmo: será um exercício para você...

As variâncias da população e da amostra

Finalmente vamos entrar no nosso último mistério: o *n-1*. Mas afinal de contas, por que cargas d'água o denominador é *n-1* e não *n*? Bom, primeiro vamos definir algumas coisas para a gente entender melhor o que está acontecendo.

Em primeiro lugar, vamos definir melhor o que vem a ser a variância da população, também conhecida como S^2 (maiúsculo, para indicar que é da população.) Ele é a média do quadrado dos afastamentos das observações em relação à média geral. Complicou? Vamos ver então a definição matemática:

$$S^2 = \frac{\sum_{i=0}^{n} \left(X_i - \bar{X} \right)^2}{n}$$

Piorou? Ih... Vamos indo adiante para ver se dá para entender, então.

Bem, mas e a variância amostral? Essa é chamada de s^2 e ela é quase igual ao nosso amigo S^2, só que o seu denominador é um pouco diferente:

$$s^2 = \frac{\sum_{i=0}^{n} \left(X_i - \bar{X} \right)^2}{n-1}$$

Percebeu a diferença? Sutil, né? Tão sutil que a gente pode até aplicar uma correção ao s^2 para ele ficar igual ao S^2.

$$S^2 = \frac{(n-1)s^2}{n}$$

Nós já até fizemos isso, lembra?

```
> sqrt((length(notas)-1)*var(notas)/length(notas))
```

Só que aqui foi para o DP, então eu acrescentei a raiz quadrada.

Legal, mas então qual é o problema? Por que a gente usa S^2 para calcular a variância da população e s^2 para a da amostra, se para a média se usa a mesma conta?

Bom, para aqueles que gostam de uma resposta mais formal, é porque s^2 é um estimador não-enviesado de σ^2, o que não é o caso de S^2. Por falar nisso, você se lembra a definição de um estimador não-enviesado? Vou dar uma pista... Pode-se provar que:

$$E(s^2) = E\left(\frac{\sum_{i=0}^{n} \left(x_i - \bar{x} \right)^2}{n-1} \right) = \sigma^2$$

Isto aí em cima quer dizer que a esperança (que corresponde à média) de s^2 é igual a σ^2.

Mas como a nossa aula é prática, não nos interessa provar isso, e sim mostrar como isso funciona na prática.

Olhando as definições acima, podemos agora entender melhor a nossa função `var.pop()`, lembra?

```
var.pop <- function (x)
{
    sum((x-mean(x))^2)/length(x)
}
```

Compare com $\frac{\sum_{i=0}^{n} \left(X_i - \bar{X} \right)^2}{n}$ e veja se não é a mesma coisa...

Vamos fazer o seguinte: vamos calcular agora todas as variâncias de todas as nossas 100 e 1000 amostras, tanto as da população quanto as da amostra:

```
> var.2.pop <- apply(amostras.2,1,var.pop)
> var.3.pop <- apply(amostras.3,1,var.pop)
> var.2.smpl <- apply(amostras.2,1,var)
```

```
> var.3.smpl <- apply(amostras.3,1,var)
```

Agora vamos calcular as médias dessas variâncias (que são na verdade as esperanças, lembra?):

```
> mean(var.2.pop)
[1] 0.55545
> mean(var.3.pop)
[1] 0.7406
> mean(var.2.smpl)
[1] 1.1109
> mean(var.3.smpl)
[1] 1.1109
```

E vamos agora comparar com a variância da nossa população:

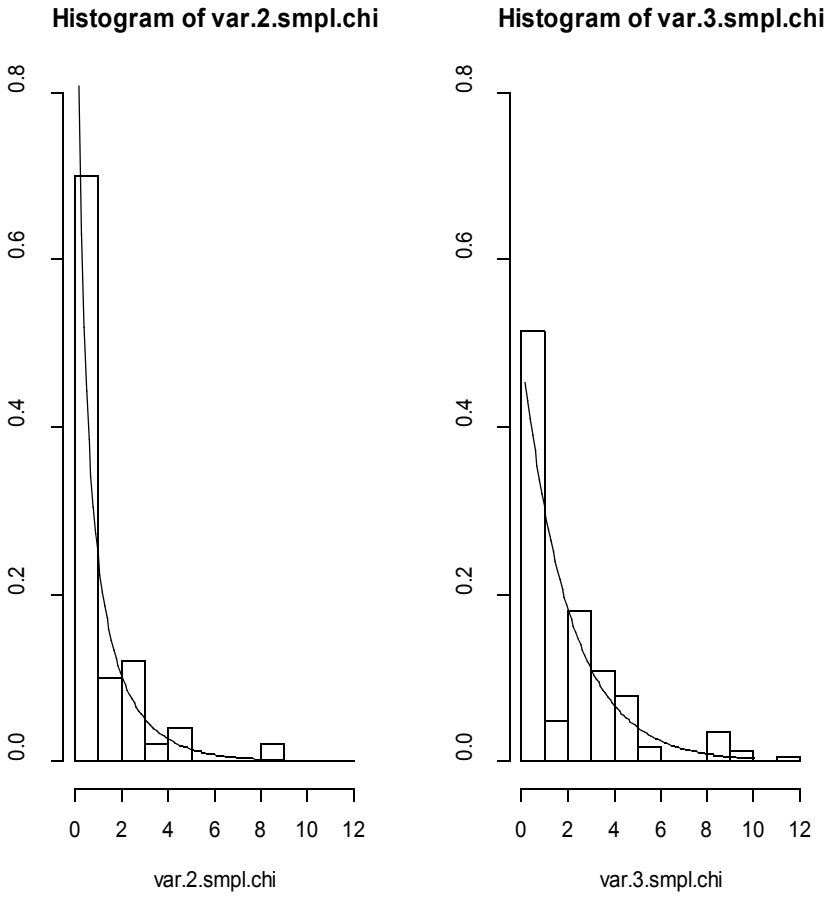
```
> var.pop(notas)
[1] 1.1109
```

Alguma conclusão interessante?
;-)
Para finalizar, você deve estar familiarizado com a distribuição que uma pequena modificação da variância amostral segue uma distribuição qui-quadrada, com *n-1* graus de liberdade (χ^2_{n-1}), se a população for Normal. Observe que neste caso, em se tratando da variância, não temos o TLC para nos ajudar. Vamos recordar esse resultado:

$$\frac{(n-1)\times s^2}{\sigma^2} \sim \chi^2_{n-1}$$

Ou seja, se multiplicarmos s^2 por *n-1* e dividirmos pela variância da população, obteremos uma distribuição chi-quadrada com *n-1* graus de liberdade. Vamos usar o R para nos ajudar a verificar esse fato. Primeiro vamos transformar os nossos vetores em distribuições qui-quadradas:

```
var.2.smpl.chi <- 1*var.2.smpl/var.pop(notas)
var.3.smpl.chi <- 2*var.3.smpl/var.pop(notas)
```



E agora vamos fazer os histogramas comparando com as distribuições teóricas:

```
par(mfrow=c(1,2))
hist(var.2.smpl.chi, freq=F, breaks=seq(0,12), ylim=c(0,0.8))
curve(dchisq(x,1), from=0.2, to=10, add=T)
hist(var.3.smpl.chi, freq=F, breaks=seq(0,12), ylim=c(0,0.8))
curve(dchisq(x,2), from=0.2, to=10, add=T)
par(mfrow=c(1,1))
```

O resultado é essa figura aí em cima. Que tal? Convenceu?

Exercícios

1. Reveja a função que nós usamos para mostrar o TLC, `histo.mean`. Todas as linhas dessa função estão comentadas. Você seria capaz de explicar a seguinte linha com o seu respectivo comentário?

```
z[i] <- mean(sample(x,n))      #O verdadeiro truque
```

   O que significa esse "verdadeiro truque"?

2. Escolha duas distribuições, uma discreta e uma contínua, gere 1000 valores destas distribuições e use o mesmo código que nós usamos para a Bernoulli na Aula para gerar uma seqüência de histogramas, o primeiro com a distribuição gerada e os demais com as distribuições das médias amostrais para um número crescente de tamanho da amostra.

Mostre os gráficos e descreva o que está acontecendo. Dicas: 1) Consulte a Tabela 1.1 da Aula 1 para recordar algumas distribuições disponíveis no R (ou, claro, use um outro programa qualquer para gerá-las); 2) O código ao qual estamos nos referindo é:

```
par(mfrow=c(3,3))
hist(x)
histo.mean(x)
histo.mean(x,5)
histo.mean(x,10)
histo.mean(x,15)
histo.mean(x,20)
histo.mean(x,25)
histo.mean(x,30)
histo.mean(x,100)
par(mfrow=c(1,1))
```

3. Explique com as suas palavras o que o código abaixo faz:

```
par(mfrow=c(1,3))
hist(notas, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,1.111006), from=4, to=10, add=T)
hist(medias.2, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/2)), from=4, to=10, add=T)
hist(medias.3, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/3)), from=4, to=10, add=T)
par(mfrow=c(1,1))
```

Nota: Não é para explicar o código, mas apenas o que ele faz, interpretar a sua saída. Dica: nós conhecemos a variância da população nesse caso?

4. Afirmamos na seção sobre Intervalos de Confiança que "duas observações podem ser feitas. A primeira é que nada é dito sobre *n* nesse caso, isto é, essa definição não depende do tamanho da amostra, ela será válida para qualquer tamanho (embora para o caso de variância da população desconhecida, *n* deve ser no mínimo 2...)".

Explique porque o tamanho de *n* deve ser no mínimo 2 nesta situação. Dica: lembre-se que o IC depende basicamente de 3 valores: o nível de significância, o tamanho da amostra e a variabilidade da amostra, como mostrado na equação abaixo, que é o IC para a média amostral com variância da população desconhecida:

$$\bar{x} \pm t_{n-1,1-\alpha/2} \times \sqrt{s^2/n}$$

5. Quantos ICs 95% calculados para as amostras de tamanho 3 das notas dos alunos de fato continham a verdadeira média, no caso da variância conhecida? Mostre um gráfico para ilustrar.

Dica: Basta modificar o código já dado para as amostras de tamanho 2, mas em todo caso aqui está a cola:

```
      plot(medias.3,rep(mean(medias.3),1000), type="b", ylim=c(0,15))
      arrows(medias.3, medias.3+(qnorm(0.975)*sqrt(var.pop(notas)/3)), medias.3,
medias.3-(qnorm(0.975)*sqrt(var.pop(notas)/3)), angle=90, code=3, length=0.1)
      points(medias.3, medias.3, pch=19)
      ic.3<-cbind(medias.3-(qnorm(0.975)*sqrt(var.pop(notas)/3)), medias.3,
medias.3+(qnorm(0.975)*sqrt(var.pop(notas)/3)))
      length(c(which(mean(notas)<ic.3[,1]), which(mean(notas)>ic.3[,3])))
```

6. Nós vimos que com a média das variâncias das amostras usando S^2 obtemos um resultado enviesado, isto é, diferente da verdadeira variância da população de notas:

```
> mean(var.2.pop)
[1] 0.55545
> mean(var.3.pop)
[1] 0.7406
```

Baseado no que nós discutimos existe alguma maneira de se obter a variância da população a partir desses valores?

7. Nós vimos na aula teórica que a média amostral é um estimador consistente, ou seja, ela:
- É assintoticamente não-enviesada (aliás ela é não-enviesada mesmo, não só assintoticamente)
- Sua variância tende para zero quando o tamanho da amostra tende para o infinito

Mostre graficamente essa segunda condição para um caso onde a variância da população seja conhecida

8. A fdp de uma distribuição *t* de Student é dada por:

$$f(t) = \frac{\Gamma\left(\nu + \frac{1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi}} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

, onde ν são os graus de liberdade e Γ é a função gama. Pode ser mostrado que para uma distribuição *t* com $\nu > 2$, sua média é fixa e igual a zero e sua variância é dada por: $\frac{\nu}{\nu - 2}$. Mostre porque a distribuição *t* torna-se bastante semelhante a uma Normal quando $\nu \to \infty$. Dica: uma maneira de mostrar isso é usando a função `curve()` para comparar gráficos de Normais e *t*'s.

Aula 3 – Distribuições amostrais
Livro: NA

1. Reveja a função que nós usamos para mostrar o TLC, `histo.mean`. Todas as linhas dessa função estão comentadas. Você seria capaz de explicar a seguinte linha com o seu respectivo comentário?

```
z[i] <- mean(sample(x,n))       #O verdadeiro truque
```

O que significa esse "verdadeiro truque"?

Esse truque é apenas a escolha aleatória de uma amostra de tamanho `n` do vetor `x` , aproveitando para se calcular a média de cada uma dessas amostras, que são guardadas em um vetor `z`. Com isso, não é necessário armazenar cada uma das amostras, mas apenas os valores das suas médias. Essa é uma boa aplicação da função `sample()`.

2. Escolha duas distribuições, uma discreta e uma contínua, gere 1000 valores destas distribuições e use o mesmo código que nós usamos para a Bernoulli na Aula para gerar uma seqüência de histogramas, o primeiro com a distribuição gerada e os demais com as distribuições das médias amostrais para um número crescente de tamanho da amostra. Mostre os gráficos e descreva o que está acontecendo. Dicas: 1) Consulte a Tabela 1.1 da Aula 1 para recordar algumas distribuições disponíveis no R (ou, claro, use um outro programa qualquer para gerá-las); 2) O código ao qual estamos nos referindo é:

```
par(mfrow=c(3,3))
hist(x)
histo.mean(x)
histo.mean(x,5)
histo.mean(x,10)
histo.mean(x,15)
histo.mean(x,20)
histo.mean(x,25)
histo.mean(x,30)
histo.mean(x,100)
par(mfrow=c(1,1))x<-rpois(1000, 3)
```

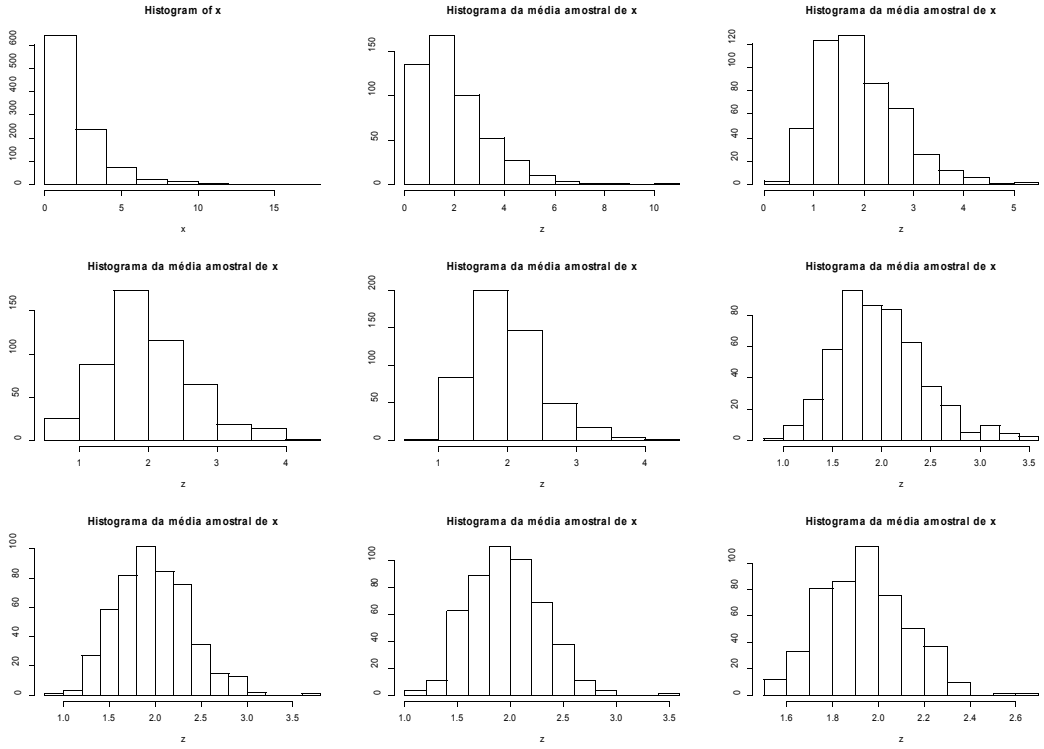
Esse exercício é apenas uma aplicação direta de conhecimentos e o resultado será o mesmo que o discutido em aula. Por exemplo, vamos fazer para uma Qui-quadrada com 2 graus de liberdade e para uma Poisson com média 3:

```
x <- rchisq(1000, df=2)
x <- rpois(1000, lambda=3)
```

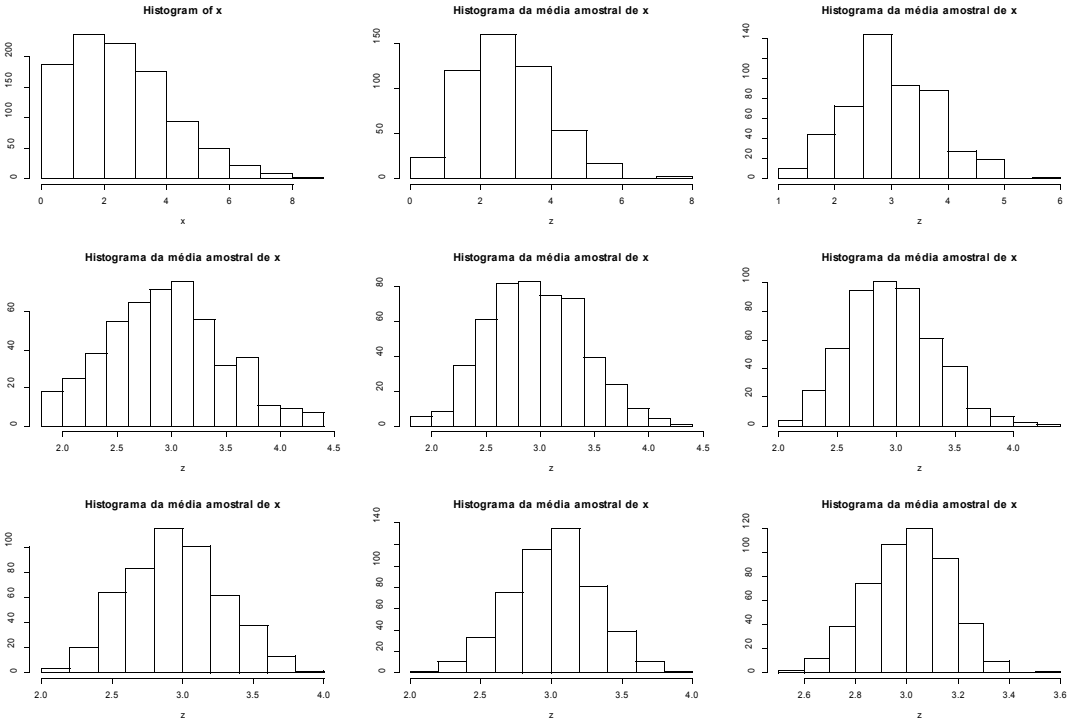
Aplicando o código acima, você visualizaria o mesmo conjunto de gráficos que vimos na aula e com a mesma tendência de aproximação da média amostral da média verdadeira e diminuição da dispersão da distribuição da média amostral, por alcunha o erro-padrão.

Os gráficos:

Para a Qui-quadrada:



Para a Poisson:



3. Explique com as suas palavras o que o código abaixo faz:

```
par(mfrow=c(1,3))
hist(notas, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,1.111006), from=4, to=10, add=T)
```

```
hist(medias.2, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/2)), from=4, to=10, add=T)
hist(medias.3, freq=F, breaks=seq(4,10), ylim=c(0,0.8))
curve(dnorm(x,7.01,sqrt(1.111006/3)), from=4, to=10, add=T)
par(mfrow=c(1,1))
```

Nota: Não é para explicar o código, mas apenas o que ele faz, interpretar a sua saída. Dica: nós conhecemos a variância da população nesse caso?

Bem, essa é fácil demais. Basta colocar o código para rodar que veremos que estamos simplesmente plotando histogramas da população e das distribuições amostrais das médias de tamanhos 2 e 3, juntamente com a Normal teórica, com a sua variância conhecida, para comparação.

4. Afirmamos na seção sobre Intervalos de Confiança que "duas observações podem ser feitas. A primeira é que nada é dito sobre *n* nesse caso, isto é, essa definição não depende do tamanho da amostra, ela será válida para qualquer tamanho (embora para o caso de variância da população desconhecida, *n* deve ser no mínimo 2...)".

Explique porque o tamanho de *n* deve ser no mínimo 2 nesta situação. Dica: lembre-se que o IC depende basicamente de 3 valores: o nível de significância, o tamanho da amostra e a variabilidade da amostra, como mostrado na equação abaixo, que é o IC para a média amostral com variância da população desconhecida:

$$\bar{x} \pm t_{n-1,1-\alpha/2} \times \sqrt{s^2/n}$$

Nesse caso, nós precisamos de alguma variabilidade nessa amostra para podermos calcular o s^2. Com uma amostra de tamanho 1, o nosso s^2 será igual a zero e não será possível calcular um IC para essa média.

5. Quantos ICs 95% calculados para as amostras de tamanho 3 das notas dos alunos de fato continham a verdadeira média, no caso da variância conhecida? Mostre um gráfico para ilustrar.

Dica: Basta modificar o código já dado para as amostras de tamanho 2, mas em todo caso aqui está a cola:

```
plot(medias.3,rep(mean(medias.3),1000), type="b", ylim=c(0,15))
arrows(medias.3, medias.3+(qnorm(0.975)*sqrt(var.pop(notas)/3)), medias.3,
medias.3-(qnorm(0.975)*sqrt(var.pop(notas)/3)), angle=90, code=3, length=0.1)
points(medias.3, medias.3, pch=19)
ic.3<-cbind(medias.3-(qnorm(0.975)*sqrt(var.pop(notas)/3)), medias.3,
medias.3+(qnorm(0.975)*sqrt(var.pop(notas)/3)))
length(c(which(mean(notas)<ic.3[,1]), which(mean(notas)>ic.3[,3])))
```

Bem, aqui é só aplicação direta do código acima para quem já vinha fazendo a aula. O resultado para mim foram 47 ICs, o que nos dá 95,3% dos ICs contendo a verdadeira média. Bastante aproximadinho, até... Vou omitir o gráfico, por uma questão de espaço, pois o código acima o fará automaticamente.

6. Nós vimos que com a média das variâncias das amostras usando S^2 obtemos um resultado enviesado, isto é, diferente da verdadeira variância da população de notas:

```
> mean(var.2.pop)
[1] 0.55545
> mean(var.3.pop)
[1] 0.7406
```

Baseado no que nós discutimos existe alguma maneira de se obter a variância da população a partir desses valores?

Bem, essa nós fizemos na aula: é só repetir. Vamos lá. Se $S^2 = \frac{(n-1)s^2}{n}$, basta tirarmos o valor de s2, certo? Então:

$$s^2 = \frac{n}{n-1} S^2$$

No R, poderíamos fazer:

```
> 2*mean(var.2.pop)/1
[1] 1.1109
> 3*mean(var.3.pop)/2
[1] 1.1109
```

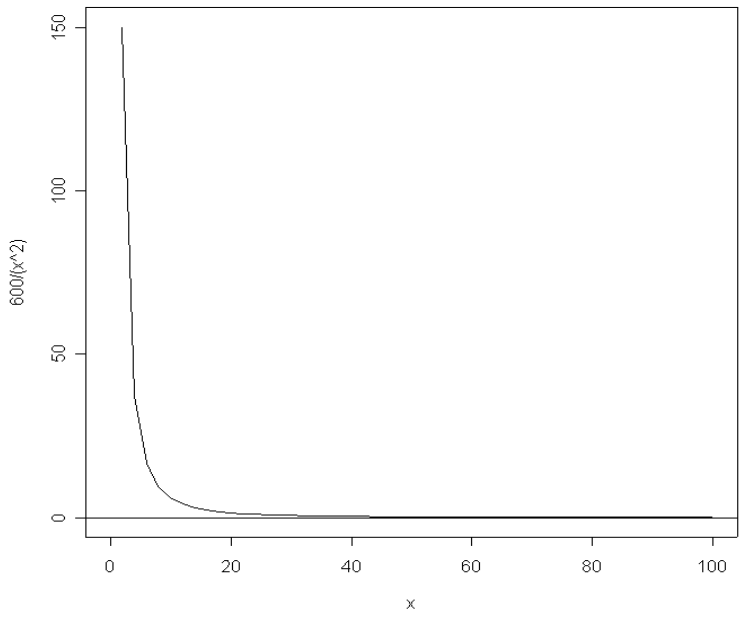
7. Nós vimos na aula teórica que a média amostral é um estimador consistente, ou seja, ela:
- É assintoticamente não-enviesada (aliás ela é não-enviesada mesmo, não só assintoticamente)
- Sua variância tende para zero quando o tamanho da amostra tende para o infinito

Mostre graficamente essa segunda condição para um caso onde a variância da população seja conhecida.

Para essa questão basta mostrar um gráfico com um número de amostras crescentes no eixo *x* e com a variância da média amostral no eixo *y*. Isso pode ser feito até mesmo à mão. Vamos admitir que a variância conhecida é de 600. No R seria algo assim:

```
x <- seq(0,100,2)
plot(x, 600/(x^2), type="l")
abline(h=0)
```

Veja o resultado:



8. A fdp de uma distribuição *t* de Student é dada por:

$$f(t) = \frac{\Gamma\left(\nu + \frac{1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi}} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$, onde ν são os graus de liberdade e Γ é a função gama. Pode ser mostrado que para uma distribuição *t* com $\nu > 2$, sua média é fixa e igual a zero e sua variância é dada por: $\frac{\nu}{\nu - 2}$. Mostre porque a distribuição *t* torna-se bastante semelhante a uma Normal quando $\nu \to \infty$. Dica: uma maneira de mostrar isso é usando a função `curve()` para comparar gráficos de Normais e *t*'s.

Seguindo a dica, poderíamos plotar diversas *t*'s e sobrepor diversas Normais a essas curvas e ver o que acontece. Vamos plotar então a *t* pontilhada e a Normal com linha cheia, para graus de liberdade crescentes de *t*. No R:

```
par(mfrow=c(3,3))
curve(dt(x,1), from=-4, to=4, lty=2, ylim=c(0,0.4))
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,5), from=-4, to=4, lty=2, ylim=c(0,0.4))
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,10), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,15), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,20), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,30), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,50), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,100), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
curve(dt(x,200), from=-4, to=4, lty=2)
curve(dnorm(x),from=-4, to=4, add=T)
par(mfrow=c(1,1))
```

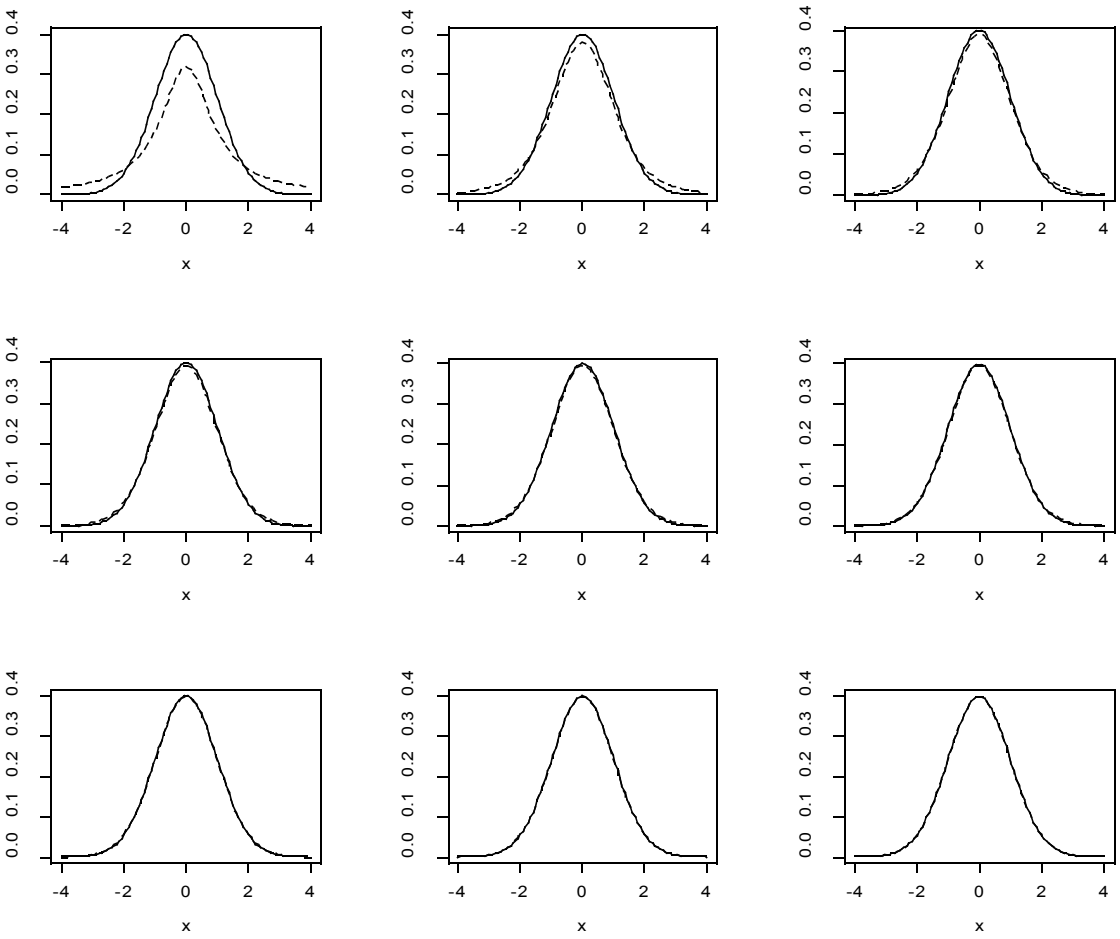
Com o seguinte resultado abaixo. Repare que para um $\nu > 100$, praticamente já não há diferença em relação à Normal.

Uma outra maneira de se mostrar isso era afirmar que como a *t* é simétrica em torno de zero para qualquer ν e sua variância tende a 1 conforme $\nu \to \infty$, ela vai se aproximando da Normal. Como mostrar que a variância tende a 1? Muito semelhante com o exercício anterior. Como *Var(t)* = $\frac{\nu}{\nu - 2}$, é só fazer:

```
x<-seq(3,200,2)
plot(x, x/(x-2), ylim=c(0,3))
abline(h=1)
```

Faça e veja o que acontece!!!

Mostre uma terceira forma de mostrar essa aproximação da *t* pela Normal, sem levar em conta a sua variância propriamente dita. Quem entregar **por escrito até Terça-Feira**, vale 0.1 ponto.


0.0 0.1 0.2 0.3 0.4
-4 -2 0 2 4
x

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 4 - Teste de Hipóteses, poder e tamanho da amostra
Livro: páginas 139 a 147

Muito embora o livro esteja referido nesta aula, a parte de teste de hipóteses propriamente dito não é abordado no livro. A compreensão do significado do teste de hipóteses é central para uma melhor compreensão dos testes estatísticos que são rotineiramente empregados em bioestatística e em estatística em geral. Além disso, permite compreender melhor os conceitos fundamentais de poder de um teste e tamanho da amostra, que podem influenciar em muito o tipo de estudo e o gasto que esse estudo vai exigir no mundo real, caso você esteja planejando um estudo epidemiológico qualquer.

Embora o teste de hipóteses seja bem abordado nesta aula, os outros dois assuntos que trataremos serão apresentados de uma maneira bastante geral, já que os testes específicos ainda não foram apresentados para você neste ponto do curso. É claro que exemplos serão usados, e nas aulas específicas de cada tipo de teste, esses temas poderão ser abordados de uma maneira mais específica. Eu digo poderão, porque o essencial é entender como a coisa funciona e não a especificidade para cada caso particular.

Teste de Hipóteses
- O teste
- O julgamento
- Relação entre o teste de hipóteses e o intervalo de confiança

Poder de um teste
Tamanho da Amostra
Simulações
Exercícios

Teste de Hipóteses

Você já deve ter sido apresentado ao clássico exemplo do julgamento, comparado a um teste de hipóteses, para passar uma compreensão intuitiva da construção de um teste de hipóteses. Muitas vezes, porém, a passagem desse exemplo para a estatística propriamente dita apresenta algumas dificuldades, que nós vamos tentar amenizar nesta aula.

O teste

Você se lembra como se monta um teste? De maneira geral:
H_0: $\theta = 0$
H_1: $\theta \neq 0$
Nesse caso, usamos um teta, que é uma representação geral, poderia tanto ser uma média, como uma variância, como uma diferença de médias. Também não definimos a direção do teste, e a hipótese alternativa é simplesmente ser diferente de zero, isto é trata-se de um teste bilateral.

O primeiro conceito-chave que deve ser frisado nesse caso é que teta é um parâmetro e não uma estatística, ou seja, é uma característica da população (que nós não conhecemos e jamais conheceremos, a não ser que realizemos um censo) e não da amostra.

O segundo conceito importante é que este parâmetro possui uma determinada distribuição teórica, e é sobre esta distribuição que os testes são feitos. Epa! Tem algo errado aqui. Se tudo nesse teste de hipóteses se refere à população que eu não conheço, inclusive a distribuição de teta, onde entra a minha amostra?

Pois é: essa é a parte difícil. Isso mais parece um beco sem saída... Nem tanto. Os parâmetros da população serão na verdade *inferidos* a partir de estatísticas tiradas da amostra. Bem, a pergunta que vem a seguir seria: mas se não temos informação alguma sobre a população, como posso ter a garantia que uma amostra desta população vai ser capaz de me dar informações suficientes para inferir os valores dos parâmetros? Bem, é aí que entram alguns resultados que nós até já discutimos. É claro que vamos nos basear no mais fácil. Que tal o Teorema do Limite Central (TLC)? Lembra dele? Será que ele poderia nos ajudar de alguma maneira nesse nosso problema?

Digamos que o nosso θ seja a média de uma população qualquer... E agora, alguma pista que nos ajude? É isso mesmo, podemos usar o TLC para nos ajudar a conhecer alguma coisa sobre este parâmetro. Como já vimos anteriormente, a média amostral de qualquer distribuição, para um tamanho de amostra suficientemente grande seguirá uma distribuição Normal com média μ e variância σ^2/n . Hum... parece mesmo que esse negócio pode me ajudar, já que nesse caso, nós temos pistas sobre parâmetros da população (μ e σ^2) a partir da nossa amostra... E lembre-se: esse resultado *independe* da distribuição da população para um *n* suficientemente grande.

Vamos ver um exemplo mais específico, então. Digamos que uma população de indivíduos sadios tenha uma pressão arterial média (PAM) de 100 mmHg, com variância de 625mmHg^2. Vamos desenhar um estudo para selecionar uma amostra de 100 idosos internados em uma unidade hospitalar e avaliar se a PAM desse grupo pode ser considerada normal. Como não temos nenhuma informação sobre a PAM desses idosos, vamos usar um teste bilateral:

H_0: $\mu - \mu_0 = 0$

H_1: $\mu - \mu_0 \neq 0$

Como o μ_0 é uma constante e é conhecido nesse caso, podemos reescrever esse teste:

H_0: $\mu - 100 = 0$

H_1: $\mu - 100 \neq 0$

Antes de bater o desespero, você entendeu que o "100" são os 100mmHg de média da população, né? Bom... Agora, o que diabos esse $\mu - 100$ está fazendo no lugar do teta lá em cima? Não se preocupe com isso, pois é apenas um algebrismo. Você pode diminuir uma constante de um parâmetro para fazer um teste de hipóteses. Aliás, como você deve se lembrar, podemos até testar a diferença de dois parâmetros, no caso de estarmos testando médias de dois grupos diferentes – confundiu? Tudo bem, veremos isso mais tarde...

Bem, voltando ao nosso teste, por que agora nós temos esse μ menos uma constante no lugar de teta? Isso é bastante confuso na minha opinião. Repare bem o que estamos testando aqui: eu tenho uma população de idosos internados (que é uma sub-população de uma população mais geral, por exemplo a população de uma cidade.) Ora, essa sub-população tem uma PAM média (olha a cacofonia...) que eu nunca conhecerei de fato, a não ser que faça um censo de todos os idosos internados em todos os hospitais digamos dessa cidade. Ao invés disso, vamos selecionar 100 deles e inferir a sua PAM, a qual eu quero comparar com a PAM de uma população geral para ver se elas diferem significativamente... Ufa!!!

Aliás, meus amigos, pensem bem e vejam que essa situação aqui na prática mesmo vai ser bem difícil de acontecer... mas enfim, deixa prá lá...

Certo, então esse μ sobre o qual queremos fazer inferências é a PAM média dessa sub-população. Quando selecionarmos a nossa amostra e calcularmos a sua média, estaremos evidentemente diante de $\bar{x}$ e usaremos o TLC para nos ajudar nessa tarefa.

Uma observação importante antes de seguirmos adiante é que nesse momento nós ainda não conhecemos a média amostral, mas no entanto já estabelecemos o teste de hipóteses que iremos usar. Nós estamos na fase do desenho do experimento, antes dele se realizar. Você deve estar imaginando agora que na prática do dia-a-dia muitas vezes os dados já foram colhidos e você estará analisando esses dados sem ter planejado experimento algum... Essa é uma discussão muito interessante, e voltaremos a falar sobre ela mais tarde nessa aula, embora seja impossível esgotar esse assunto, que é de fato um tema de intenso debate.

O julgamento

Agora que o nosso teste está entendido, vamos passar ao nosso julgamento. Vamos repetir as regras desse julgamento só para refrescar a memória:

- Toda pessoa é inocente até que se prove o contrário – aqui esse fato é traduzido pela hipótese nula, que trata da igualdade da diferença entre a verdadeira média da sub-população, e a constante que queremos testar com o valor zero;
- Os jurados ainda não conhecem as evidências que serão apresentadas (os dados), mas já sabem as possíveis hipóteses, inocente ou culpado – inocente é a igualdade e culpado é a diferença. Note que as hipóteses devem ser estabelecidas *antes* da apresentação das evidências;
- Existem dois enganos que podem acontecer neste julgamento: a absolvição de um culpado ou a condenação de um inocente – são os erros Tipo II e Tipo I, respectivamente;
- O pior engano possível é sem dúvida condenar um inocente (muito embora algumas pessoas possam discordar...) Seria tão grave, que gostaríamos de ter total controle sobre esse tipo de engano, estabelecendo um limite para ele, antes mesmo de conhecermos as evidências – é o estabelecimento do alfa – que geralmente é de 0.05 ou 5%. Não estranhe: isso quer dizer exatamente o que você entendeu, ou seja, até 5% dos inocentes podem ir para a cadeia...
- O engano de absolver um culpado é considerado menos grave, e nos damos o direito de nos preocuparmos com ele após a apresentação das evidências, aliás, preferimos até fazer o contrário: estabelecer a nossa capacidade de, apresentadas as evidências, sermos capazes de provar que um culpado é realmente culpado, o que dependerá da qualidade e da quantidade de evidências apresentadas – É o famoso poder do teste, também conhecido como $1-\beta$ e que como veremos vai depender da qualidade e quantidade dos dados coletados;

Passemos então à apresentação das evidências coletadas:

$n = 100$

$\bar{x} = 112.28$

$s^2 = 22.72$

Claro que essa mágica foi feita no R. Caso queira fazer também, com resultados algo diferentes, crie um vetor assim:

```
pam.idosos<-rnorm(100, mean=110, sd=25)
```

E depois calcule a média e desvio-padrão desse vetor.

Voltando ao nosso julgamento, os advogados de defesa pedem a inclusão da variância da população, que é conhecida e igual a 625 $mmHg^2$, pedido que é imediatamente indeferido pelo juiz (alguém sabe explicar porque?).

Bem, na impossibilidade de usar a variância da população, teremos que lançar mão de um teste *t* para uma amostra, que como você deve se lembrar, usa a variância da amostra no seu lugar, e

então e passaremos a compar a média amostral com o valor da população geral de 100 mmHg. A maneira mais fácil de fazer isso é usar a função `t.test()` no R:

```
> t.test(pam.idosos, mu=100)

        One Sample t-test

data:  pam.idosos
t = 5.4056, df = 99, p-value = 4.474e-07
alternative hypothesis: true mean is not equal to 100
95 percent confidence interval:
 107.7729 116.7887
sample estimates:
mean of x
 112.2808
```

Observe que nesse caso tivemos que "dizer" para o R que o valor contra o qual queremos comparar a média é 100, com o argumento `mu=100` .

Ué, mas já acabou o nosso julgamento, então? Não deu nem para perceber o que aconteceu... Bem, é melhor irmos devagar. Vamos esquecer a saída aí em cima por enquanto e tentar entender o que está acontecendo.

Primeiramente, as nossas evidências precisam ser processadas para que o júri possa dar o seu veredicto. Neste nosso caso, baseado no TLC e na utilização da variância da amostra para inferir a variância da população, esse processamento consiste em construir a distribuição da média amostral, sob a hipótese nula... Ihhhh! Complicou, hein? O que isso quer dizer? Vamos reescrever a hipótese nula, só para melhorar a compreensão:

H_0: $\mu = 100$

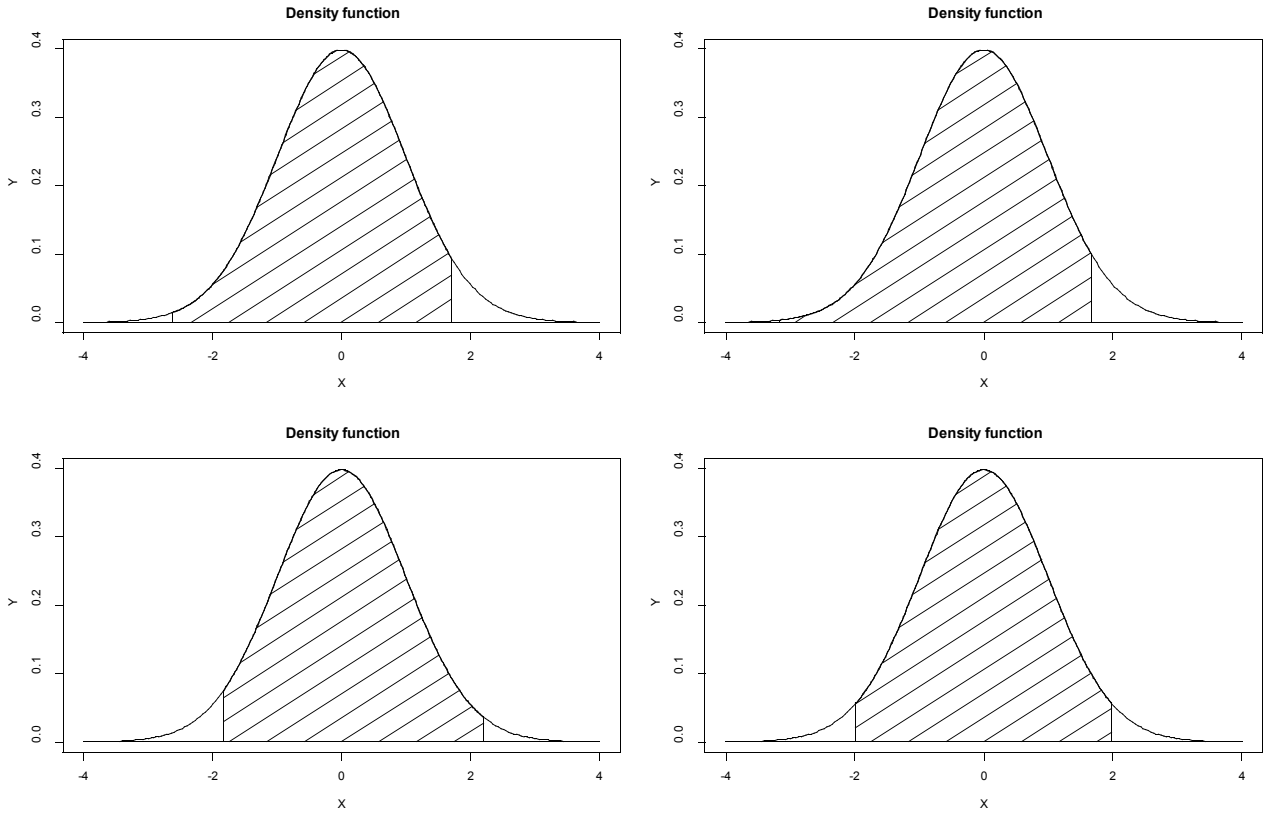
Hum, já melhorou! Quer dizer que sob a hipótese nula, a (verdadeira) média é igual a 100. Mas o que é que nós realmente temos? Ah, nós temos a *amostra*. E o que podemos dizer sobre a média dessa amostra? Que ela segue uma distribuição *t* com *n-1* graus de liberdade e que a sua média amostral é igual a 110.64. Bem, com essas informações, podemos já fazer algo que fizemos anteriormente, que é calcular e desenhar um IC 95% para essa média amostral... Você seria capaz de fazer isso com a ajuda do R?

Além disso, o que poderíamos fazer? Como entender melhor o processamento das nossas evidências? Pelo que estamos vendo, isso não é a mesma coisa que calcular o IC 95%. Repare que no teste de hipóteses estamos afirmando que sob a hipótese nula a verdadeira média é igual a 100. Isso quer dizer que para construirmos a curva da média sob a hipótese nula, devemos fazê-lo para uma distribuição com média 100 e não 112.28, como você poderia estar pensando (pois esse seria o raciocínio do IC 95%. o qual seria construído ao redor do valor 112.28, como você deve se lembrar – no caso do IC, nós estamos trabalhando apenas com valores amostrais, e nada é dito sobre a população, entendeu a diferença?)

Muito bem. Mas agora para construirmos uma curva, vamos ter que voltar à nossa hipótese escrita como uma diferença e igual a zero – isso porque como estamos lidando com uma distribuição *t* não faz muito sentido usarmos uma distribuição com média 100. Para a construção da curva, não necessitamos ainda do valor da média calculada a partir da amostra, pois podemos visualizar uma densidade da distribuição *t* com o auxílio apenas do seu único parâmetro, que é o número de graus de liberdade (você deve se lembrar que nesse caso será *n-1*, ou seja, 99 para o nosso problema.) Essa curva seria então a distribuição da amostra sob a hipótese nula.

Com o nosso erro do tipo I fixado em 5% e bilateral, para separar a área de não rejeição da área de rejeição, basta calcular uma área abaixo da curva de densidade (fdp) que tenha 95% de toda a massa de probabilidade. O que sobrar será a nossa área de rejeição... Mas será que isso é suficiente para o nosso veredicto, ou seja, para definir a nossa área de rejeição?

Não... Como você deve estar pensando, existem infinitas áreas com valor 0.95 sob a curva de densidade de uma t_{99}, não é verdade?. Veja a figura abaixo:



A área hachurada de todas essas curvas têm uma área de 0.95. Porém, a área da curva no canto inferior direito tem uma outra característica que a fará ser a única dentre as infinitas áreas de não rejeição que será usada no nosso julgamento. Alguém arrisca dizer que característica é essa?

Isso mesmo: a **simetria** das áreas de rejeição. Esta é a única área de não rejeição que proporciona áreas de rejeição simétricas, ou seja, com área iguais (a 0.025 cada, no caso de um alfa de 0.05). Como o nosso teste é bilateral, e nós não sabíamos *a priori* se a estatística seria maior ou menor que o valor sob a hipótese nula, é justo que a massa de probabilidade destinada aos valores menores que 100 fosse a mesma que a dos valores maiores que 100.

Aposto que agora surgiu uma nova dúvida... Estávamos felizes, falando de média igual a 100, 112, etc, e me aparecem umas curvas com valores de -4 a 4???!!! Não se desespere ainda... Como já mencionei, se trata de uma distribuição *t*, e não faz sentido falarmos em médias que não sejam iguais a zero – ao contrário da Normal, que pode assumir várias médias diferentes (afinal ela é um dos seus parâmetros), a distribuição *t* tem sempre média zero.

Tudo bem, mas então como é que eu posso comparar alguma coisa com essa curva? Os valores são muito diferentes... É aí que entra o famoso cálculo da estatística *T*, que você já deve ter ouvido falar, e que vai ser na verdade parte do nosso processamento das evidências apresentadas. Vamos recordar o seu cálculo:

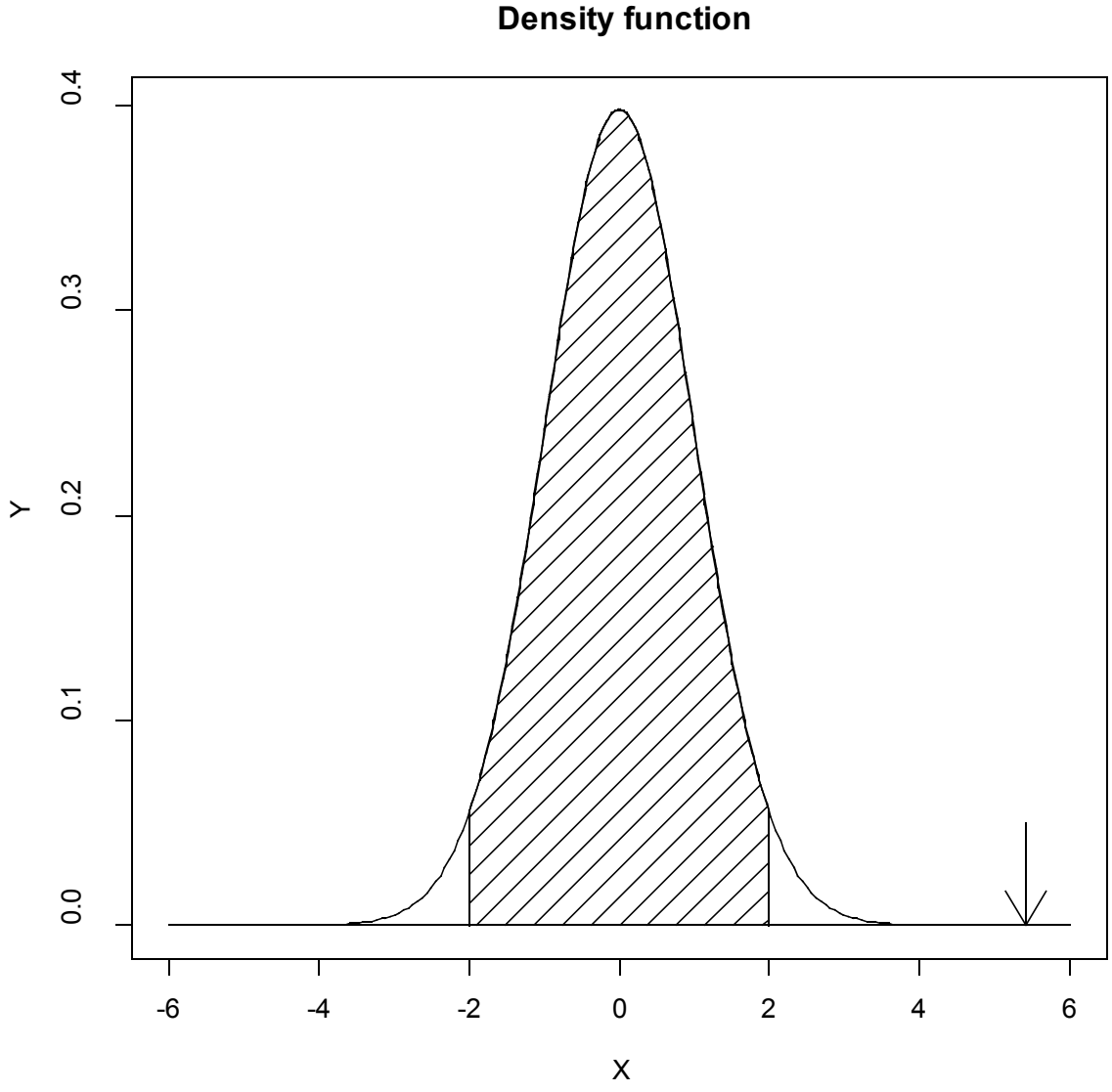
$$T = \frac{\bar{x} - \mu_0}{\sqrt{s^2/n}}$$

Repare que a obtenção do *T* envolve todas as evidências colhidas: a média amostral, a variância da amostra e o tamanho da amostra. Essa conta pode ser feita facilmente no R:

```
(mean(pam.idosos)-100)/sqrt(var(pam.idosos)/100)
[1] 5.405571
```

Confira com o resultado obtido com o `t.test()` acima... Esse processo de obtenção do *T* é equivalente à padronização que fazemos quando estamos trabalhando com a distribuição Normal (lembra?) A única diferença é que em vez de usar a variância da população, estamos usando a da amostra.

É claro que neste caso, com o gráfico (aquele inferior direito que calculamos acima) e com esse valor, é possível de cara dar o veredicto de rejeitar a hipótese nula e condenar a média desta sub-população a ser diferente da média da população geral, não é mesmo? Afinal 5.4 está bastante distante dos cerca de 2 (valor no "olhômetro") para o limite da área de rejeição à direita do gráfico, concorda? Vamos ver, só para conferir:



A seta ali à direita representa o 5.4. Ele está bastante na área de rejeição, não é mesmo? Mas e se a nossa estatística *T* estivesse muito próxima do valor 2, por exemplo? Ficaria difícil ver no olho se deveríamos condenar ou absolver a nossa média... Bem, existem duas maneiras de se contornar este problema...

A primeira é calculando-se exatamente os valores críticos, que nada mais são do que os limites do intervalo para o qual nós construímos a área de não rejeição. Ora, se a estatística calculada cair dentro deste intervalo, ela estará na área de não rejeição e a nossa média será

absolvida. Se cair fora do intervalo, estará na área de rejeição e a média será condenada (rejeitaremos a hipótese nula.)

O cálculo desse intervalo é fácil, e só depende na verdade da distribuição em questão e seus parâmetros. Basta calcularmos dois quantis: o primeiro, que será o limite inferior desse intervalo, que no nosso caso aqui será para uma *t* onde teremos uma área de 0.025, contando desde menos infinito até este quantil. Isso nós já aprendemos a fazer no R, não é mesmo?

```
> qt(0.025, df=99)
[1] -1.984217
```

O limite superior deste intervalo é calculado para a mesma t_{99} para se obter uma área de também 0.025 a partir desse valor até mais infinito. Claro que pela simetria da distribuição *t* podemos também calcular o quantil que corresponde a uma área de 1 – 0.025 = 0.975, indo de menos infinito até esse valor. Bem para dizer a verdade não precisamos calcular nada, já que devido à simetria em torno de zero este valor obrigatoriamente é igual ao valor acima, só que com sinal positivo. Duvida? Então faça:

```
qt(0.025, df=99, lower.tail=F)
```

ou

```
qt(0.975, df=99)
```

A outra maneira de se fazer isso é calculando-se o famoso p-valor. Mas para facilitar as nossas contas, vamos agora comparar a nossa população de idosos com uma média de 106 mmHg e não mais com 100 mmHg como estávamos fazendo anteriormente. Veja bem que a nossa curva não muda, ela continua sendo a mesma e o intervalo de (-1.98,1.98) ainda é o mesmo – eles não dependem do valor que queremos testar. O nosso teste porém mudou:

H_0: $\mu - 106 = 0$

H_1: $\mu - 106 \neq 0$

Obviamente a nossa estatística *T* mudou, pois ela depende de μ_0, que mudou agora. Vamos fazer um novo teste *t*:

```
> t.test(pam.idosos, mu=106)

        One Sample t-test

data:  pam.idosos
t = 2.7646, df = 99, p-value = 0.006799
alternative hypothesis: true mean is not equal to 106
95 percent confidence interval:
 107.7729 116.7887
sample estimates:
mean of x
 112.2808
```

Repare que obviamente o IC 95% para a média amostral também não mudou. Aliás, somente dois valores foram alterados: o valor da estatística *T* e justamente o p-valor. Como funciona então o p-valor? É simples: o p-valor é a área sob a curva (nesse caso a t_{99}) a partir do quantil do mesmo valor da estatística *T* que foi calculada até mais infinito (para um *T* positivo), ou a área de menos infinito até a estatística *T*, se ela for negativa. Como recaímos no primeiro caso, vamos ver como fica:

$$p - valor = \int_{T}^{+\infty} f(t)\, dt$$

Onde a *f(t)* é a fdp de uma t_{99}. Vamos conferir no R, usando funções já conhecidas nossas:

```
> pt(2.7646, df=99, lower.tail=F)
[1] 0.003399186
```

Epa! Esse p-valor não é igual ao p-valor calculado no nosso teste *t* acima (0.006799)!!!! O que está acontecendo aqui? Calma. O problema aqui é que nós estamos fazendo um teste bilateral, lembra? Nós não sabíamos *a priori* se a média da amostra seria maior ou menor que o meu valor de teste (você pode dizer que para este segundo caso, nós já sabíamos... bom, mas temos que fingir que não sabíamos, ou estaríamos apresentando evidências para o nosso júri com informações privilegiadas, o que não é justo...)

Quando estávamos construindo a nossa área de não rejeição, o fato de acharmos duas áreas de rejeição simétricas fez sentido, para este teste bilateral, mas não dá para entender muito bem para o caso do p-valor, Não é mesmo? É mesmo! Tanto que para calcular o p-valor para um teste bilateral, usamos uma convenção, que é multiplicar o valor encontrado por 2 (o que na verdade corresponde a calcular a integral para os intervalos simétricos e somar os dois valores.) Complicou? Vamos ver devagar:

$$p - valor = 2 \times \int_{T}^{+\infty} f(t)\, dt \; = \int_{T}^{+\infty} f(t)\, dt + \int_{-\infty}^{-T} f(t)\, dt$$

Conferindo no R:

```
> 2*pt(2.7646, df=99, lower.tail=F)
[1] 0.006798372
> pt(2.7646, df=99, lower.tail=F)+pt(-2.7646, df=99)
[1] 0.006798372
```

Muito bem, conferindo com o valor obtido no teste *t* acima, o resultado bate. Mas afinal de contas, qual é a interpretação desse p-valor? Não parece tão intuitivo quanto a nossa área de rejeição... E de fato não é...

O que ele representa é a probabilidade de se selecionar uma amostra, cuja média amostral é pelo menos $\bar{x} - \mu_0$ (pode ser esta diferença ou uma diferença maior, mais para o lado do infinito), se a verdadeira média da população fosse μ_0 . Entendeu? É aquela área que a gente calculou...

Repare que este conceito é bastante diferente daquele que nós estudamos para os intervalos de confiança e bem menos intuitivo também. Ainda assim é a medida mais usada em estatística.

E assim o nosso julgamento entra em recesso, com a condenação incondicional do nosso réu. Voltaremos a ele ainda mais tarde quando falarmos de poder e tamanho de amostra

Relação entre o teste de hipóteses e o intervalo de confiança

Você agora deve estar pensando: bom, o teste de hipóteses com as suas áreas de rejeição e p-valores são conceitualmente diferentes dos intervalos de confiança. Mas é muito comum, especialmente em Epidemiologia vermos relatadas ambas as medidas, lado a lado, e tem uma coisa que sempre acontece: sempre que o p-valor é menor que 0.05, o IC 95% não contém o valor do parâmetro sendo testado na hipótese nula. Por exemplo, uma *odds ratio* (OR) que tenha um p-valor relatado < 0.05 nunca contém a unidade, sim, porque o que estamos falando aqui vale para qualquer teste de hipóteses e qualquer IC, para qualquer parâmetro.

Claro que isso não é coincidência. De fato existe uma relação entre essas duas medidas apesar delas serem construídas de maneiras diferentes e terem interpretações próprias.

Não vamos perder tempo com provas algébricas formais aqui (que aliás nem são tão complicadas assim), e vamos partir para uma demonstração prática. Vamos pegar a nossa amostra de idosos novamente. Se o que acabei de afirmar é verdade, então todas as estatísticas *T* calculadas para qualquer valor fora do IC 95% para o nosso vetor de PAMs (107.7729,116.7887) deverá cair na área de rejeição do nosso teste de hipóteses (e, por conseqüência, ter um p-valor menor que 0.05). Vamos fazer o seguinte: vamos criar um vetor seqüencial cujos limites são o IC 95%, com espaço de 0.1:

```
x <- seq(107.7729,116.7887,0.1)
x[92] <- 116.7887
```

Tivemos que acrescentar o último valor do vetor só para podermos ter um vetor exatamente igual ao do intervalo. Se quiser, confira o vetor. Vamos agora calcular todas as estatísticas *T* associadas a esses valores do vetor `x`:

```
T <- (mean(pam.idosos)-x)/sqrt(var(pam.idosos)/100)
```

Não creio que seja necessária explicação. Esta é a mesma equação que usamos anteriormente, apenas colocando o vetor `x` no lugar do valor 100. Claro que aqui nos valemos de um cálculo vetorial no R. Vamos agora ver se esses valores de *T* caem mesmo na área de rejeição. Lembra do intervalo? É mais fácil fazer o contrário: tínhamos calculado o intervalo de não rejeição - (-1.98,1.98) e podemos verificar se os valores caem fora deste intervalo. Para isso, vamos ver os valores máximo e mínimo para o nosso recém-calculado vetor `T`:

```
> min(T)
[1] -1.984238
> max(T)
[1] 1.984207
```

Como o menor valor deste vetor é menor que -1.98 e o maior valor é maior que 1.98, podemos concluir que de fato todos esses valores caem na área de rejeição. Mas e o p-valor? Como será que ele se comportaria? Vamos calcular:

```
pvalor <- 2*pt(abs(T), df=99, lower.tail=F)
```

A explicação desse código fica para um exercício. Basta agora ver qual é o menor p-valor que foi calculado e ele deve ser maior que 0.05:

```
> min(pvalor)
[1] 0.04999757
```

Epa! Esse p-valor aqui ainda é menor que 0.05. Como fica isso? Bem, meus amigos, isso é a estatística... Certamente é um problema de de arredondamento. Esse valor corresponde ao valor do limite superior, usado com 4 dígitos (116.7887.) Daria muito trabalho calcular exatamente, sem arredondamento esse valor. Vamos então ver o menor valor possível que arredondado daria 116.7887. Bem, 116.78865 seria o caso... Confira:

```
> round(116.78865,4)
[1] 116.7887
```

Agora vamos ver o que teríamos nesse caso (não se assuste: eu apenas coloquei tudo ao mesmo tempo agora):

```
> 2*pt(abs((mean(pam.idosos)-116.78865)/sqrt(var(pam.idosos)/100)), df=99,
lower.tail=F)
[1] 0.05000006
```

Até que ficou aproximadinho...
;-)
Convenceu? Não? Faça então com outro valor qualquer e confira...

Poder de um teste

Vamos agora voltar para o nosso teste e também para o nosso julgamento para discutir o famoso poder de um teste. Para facilitar a nossa vida porém, vamos agora usar uma Normal em vez de uma distribuição *t*. Nesse caso, vamos trabalhar com uma variância conhecida e igual a 625 $mmHg^2$, uma situação bastante irreal, mas é só para ficar mais palpável. No nosso julgamento, a única garantia que queríamos era estabelecer um erro máximo para não condenar um inocente, ou seja para não rejeitar a hipótese nula com uma probabilidade qualquer. Para tal, em geral, estabelece-se um limite de 5%.

Como nós já comentamos também, o poder nada mais é do que a probabilidade de um acerto e não de um erro: é a probabilidade de rejeitar a hipótese nula, quando de fato ela é falsa, ou seja, é a probabilidade de condenar um culpado. Isso evidentemente é o complemento de se absolver um culpado, ou seja o complemento do erro tipo II. A probabilidade do erro tipo II é também conhecida como β. Logo, o poder será a probabilidade de não cometermos o erro tipo II ou $1-\beta$. Ao contrário da probabilidade de cometer o erro tipo I, ou o nosso famoso alfa, o beta não é fixo e ele depende de três fatores: o próprio alfa, o valor testado na hipótese alternativa e o tamanho da amostra.

O tamanho da amostra não deve causar estranheza a você, já que você deve estar se lembrando da nossa discussão sobre intervalos de confiança, que quanto maior o tamanho da amostra, menor a variância da média amostral. Nesse caso, intuitivamente você poderia pensar que faz sentido um teste ter uma precisão maior quanto maior o tamanho da amostra e por conseqüência um maior poder. O erro tipo I também não deve ser intuitivamente difícil de entender, já que se nós aumentarmos o limite do erro tipo I, deveremos diminuir o limite do erro tipo II – é como se o júri ficasse mais inclinado a condenar o nosso réu, e portanto menos inclinado a absolvê-lo.

Mas e quanto ao valor testado? Como é isso?

Para início de conversa, qual seria o valor testado sob a hipótese alternativa no nosso exemplo acima? Recordando:

H_0: $\mu - \mu_0 = 0$

H_1: $\mu - \mu_0 \neq 0$

Não parece haver um valor definido não é mesmo? É isso aí. No nosso caso, existem infinitos números sendo testados, aliás, qualquer valor que não seja o número zero cabe na nossa hipótese alternativa. Complicou, né? Bem, é que na verdade o poder do teste é uma função e não um número exato, e dependerá do valor contra o qual queremos testar a hipótese nula; para dizer a verdade ele depende mesmo da *diferença* que queremos testar. Vamos ver como isso funciona na prática, então.

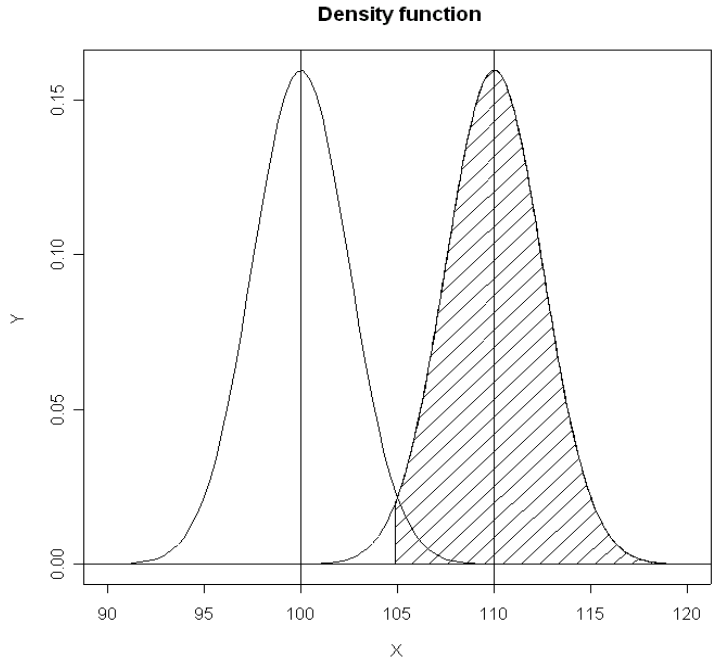
Para começar, vamos recordar o que significa o poder, em termos intuitivos: como temos que rejeitar a hipótese nula, quando ela é de fato falsa, estamos procurando uma probabilidade a partir de um ponto qualquer (um quantil qualquer) que está certamente mais extrema do que o nosso alfa, sob a hipótese nula, para podermos rejeitá-la, certo? Se temos que rejeitar a hipótese nula, esse valor tem que estar na área de rejeição (sob a hipótese nula.)

Acontece que ao mesmo tempo queremos obter uma probabilidade sob a hipótese alternativa, na verdade, concorda? Ora, se estamos afirmando que a hipótese nula é falsa, isso

significa que a hipótese alternativa é que é verdadeira e portanto é sobre a distribuição da hipótese alternativa que nos interessa calcular essa probabilidade. Complicou bem, não?

É que você não está levando em conta o fato de ser possível que essas distribuições se sobreponham. Aliás, no caso da Normal e da *t* elas vão sempre se sobrepor, não é mesmo? Vamos ver um exemplo? Vamos usar a hipótese alternativa de 110 mmHg e ver o que acontece.

Repare na figura abaixo a área que estamos procurando. A curva da esquerda é a distribuição da média sob a hipótese nula e a da direita, sob a hipótese alternativa. Nós marcamos então o limite do nosso erro do tipo I na curva da esquerda, mas aí teremos que calcular a área sob a curva a partir deste valor, mas na curva da direita (a curva da hipótese alternativa).



A área hachurada é o poder do teste para uma hipótese alternativa de 110 mmHg. Obviamente isso é só um exemplo, pois como já mencionado, não temos uma hipótese alternativa simples, com apenas um número, mas sim infinitos números. Desse modo, é comum representarmos o poder de um teste através de uma curva, a curva de poder. É como se tivéssemos várias curvas diferentes para a hipótese alternativa, com várias áreas diferentes, que variam em função da hipótese alternativa.

Vamos usar o R para construir uma curva de poder para esse teste – lembre-se que estamos mantendo fixos o nosso tamanho de amostra de 100 e ainda o nosso alfa de 0.05!!!

Bem, uma maneira intuitiva (diferente da fórmula do livro...) de se desenhar essa curva de poder para vários valores da hipótese alternativa poderia ser o seguinte: calculamos o valor crítico para a hipótese nula, demarcando a nossa área de rejeição. Toma-se esse quantil calculado para a hipótese nula e calcula-se a área mostrada no gráfico acima para vários valores diferentes. Vamos começar criando esses valores, digamos de 100 a 120:

```
x <- seq(100,120,1)
```

Agora vamos calcular o poder para esses vários valores, usando o raciocínio acima, com a ajuda do R:

```
poder <- pnorm(qnorm(0.975, mean=100, sd=2.5), mean=x, sd=2.5,
lower.tail=F)
```

Ihhhh! Complicou... Vamos explicar devagar. Estamos calculando uma área com a função `pnorm()` e esta área é de um ponto até o infinito (com o agumento `lower.tail=F`.) Muito bem, o ponto a partir do qual queremos calcular esta área é o limite da região crítica sob a hipótese nula, certo? Este ponto corresponde ao quantil para uma área de 0.975, para uma Normal de média 100 (hipótese nula) e um desvio-padrão de 2.5 (de onde vem esse valor? Dica: lembre-se que a variância da população é conhecida e igual a 625 $mmHg^2$). Mas essa área é para ser calculada sob a hipótese alternativa, e portanto os demais argumentos são a média sob a hipótese alternativa (que é o vetor `x`) e o desvio-padrão que é o mesmo que sob a hipótese nula. Agora basta desenhar o gráfico:

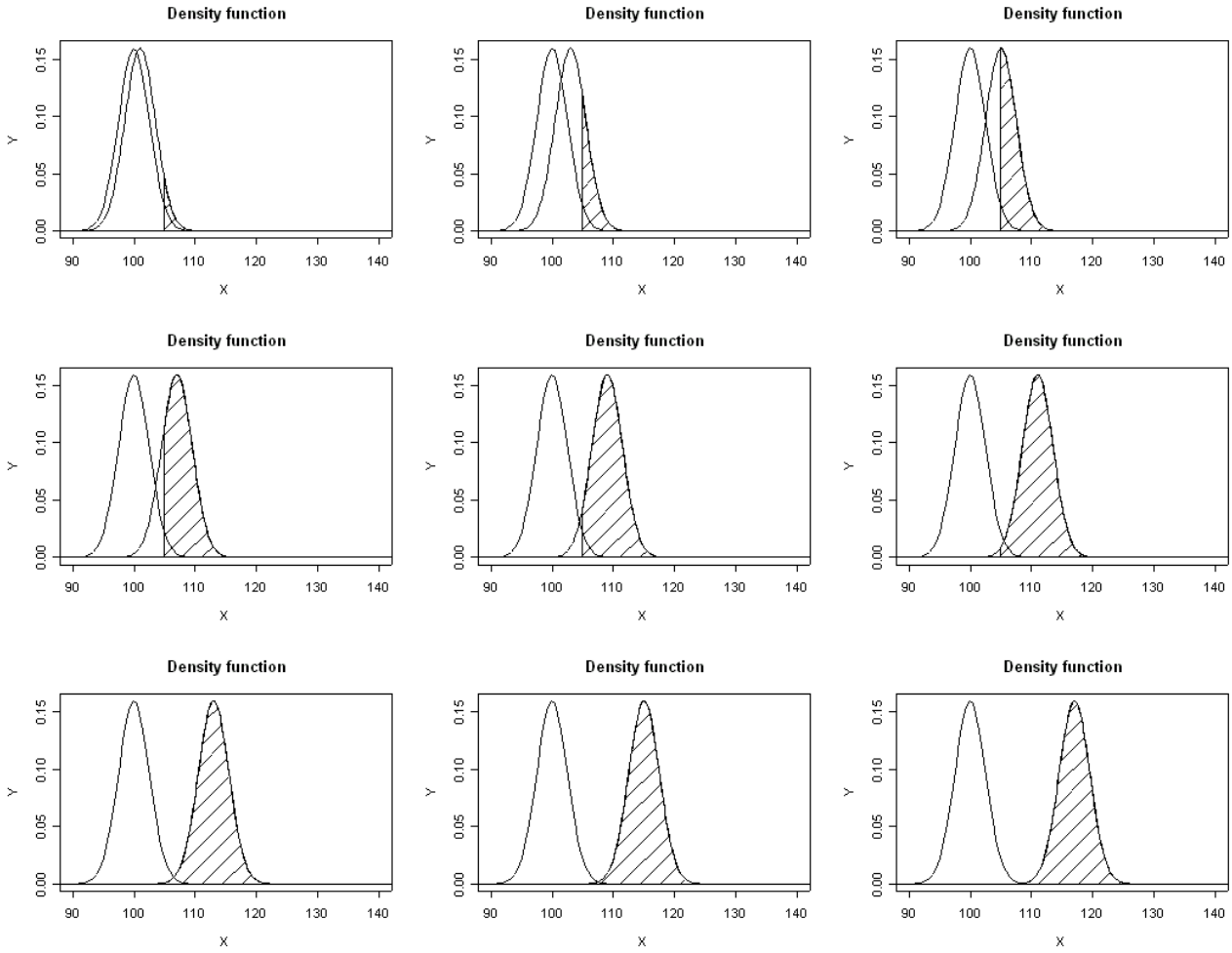
```
plot(x,poder, type="l")
```

Que tal? Que conclusões você pode tirar dessa curva?

Se você quiser conferir essa curva com a que você aprendeu na aula ou no livro, use este código:

```
poder1 <- pnorm(qnorm(0.025)+abs((100-x))/2.5)
points(x,poder1)
```

Confere? Entendeu o que está acontecendo? Aquela curva da hipótese alternativa está se movendo no eixo *x*, produzindo diferentes áreas, com nessas curvas abaixo, construídas para hopóteses alternativas variando de 101 a 117 com um incremento de 2:



Reparou como a área aumenta à medida que a hipótese alternativa se afasta da hipótese nula?

Agora, uma coisa que pode estar estranha é que apesar de estarmos lidando com um teste bilateral, nós só tratamos de poder para um dos lados. Nesse caso nós escolhemos valores maiores que o valor para a hipótese nula. Na verdade, uma vez que estabelecemos os valores, não faz diferença se o valor foi menor ou maior. Com o uso da equação, que leva em conta valores absolutos, não importa a direção, que o valor do poder será o mesmo. Visualmente fica mais fácil fazer para um dos lados só. Note que isso não funcionaria para o meu raciocínio intuitivo, e a equação teria que ser modificada um pouco para o cálculo do poder para valores menores que 100.

Agora que já vimos que quanto mais distante do valor da hipótese nula maior o poder, vamos ver o que acontece quando alteramos os outros dois parâmetros, o alfa e o tamanho da amostra. Vamos fazer o seguinte: criaremos uma função para desenhar uma matriz de curvas de poder para uma hipótese nula fixa e várias hipóteses alternativas (as mesmas que nós já construímos) em relação a uma média. Cada curva terá valores variáveis para o seu alfa e também para o seu tamanho de amostra. Veja esta função:

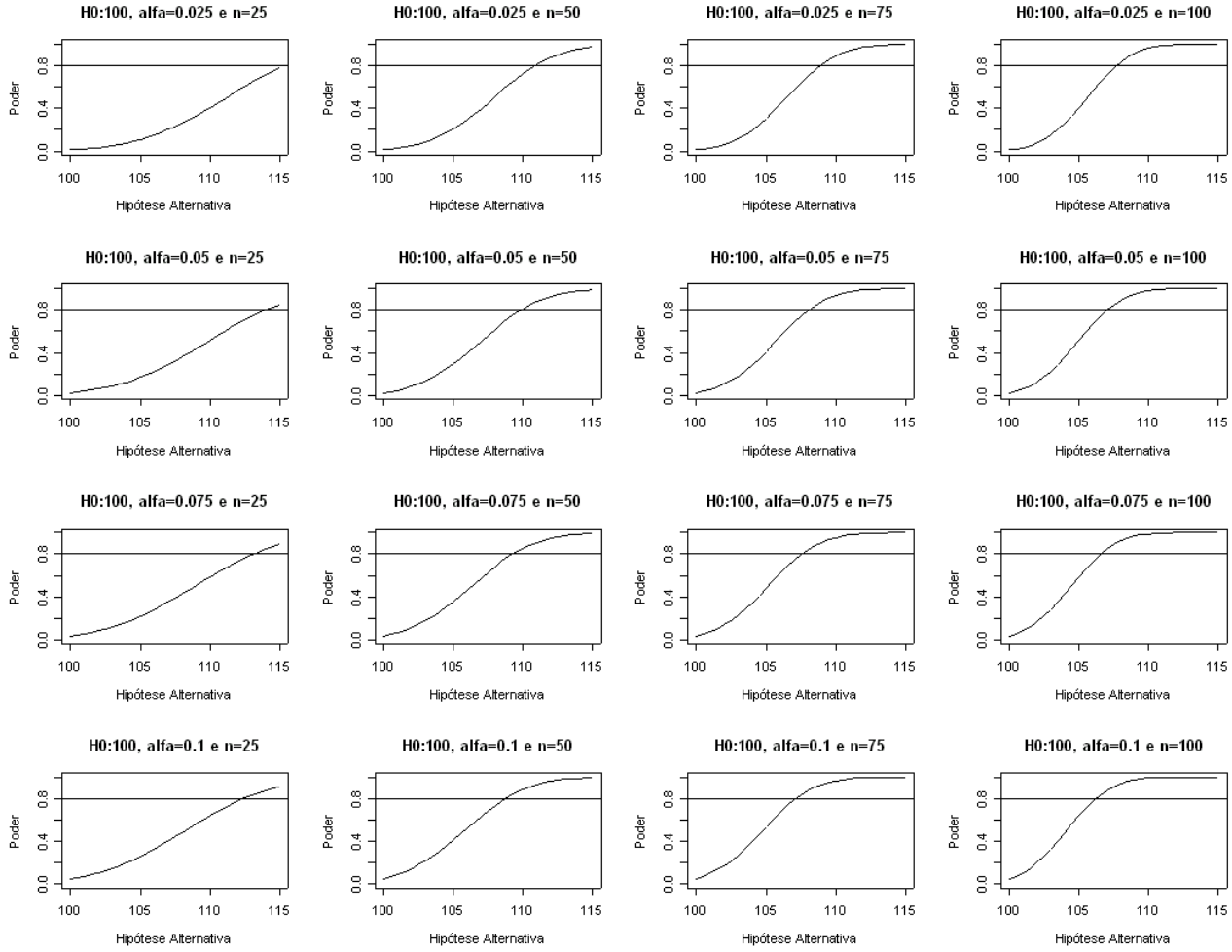
```
curva.poder <- function(H1, H0, pop.var, alfa, n)
{
      a <- length(alfa)
      b <- length(n)
      par(mfrow=c(a,b))
      for (i in 1:a){
            for (j in 1:b){
                  plot(H1,pnorm(qnorm(alfa[i]/2)+abs((H0-
H1))/sqrt(pop.var/n[j])), type="l", ylim=c(0,1), ylab="Poder", xlab="Hipótese
Alternativa", main=paste(c("H0:", H0, ", alfa=", alfa[i],  " e n=", n[j]),
collapse=""))
                  abline(h=0.8)
            }
      }
      par(mfrow=c(1,1))
}
```

Ela vai pegar um vetor de hipóteses alternativas, e plotar a curva de poder em relação a uma hipótese nula, dada uma variância da população e ainda vetores de alfas e tamanhos de amostra. Os gráficos também têm uma linha horizontal em 80% de poder para referência. Note que o resultado é para alternativa bicaudal.

Veja um resultado abaixo, para a nossa mesma hipótese nula de uma média de 100 mmHg, várias alternativas e a variância de 625 $mmHg^2$:

```
curva.poder(H1=seq(100,115,0.5), H0=100, pop.var=625,
alfa=seq(0.025,0.10,0.025), n=seq(25,100,25))
```

As curvas obtidas podem ser vistas na figura abaixo. Que conclusões você poderia tirar? Façam uma linha vertical onde a curva cruza com a linha traçada em 80% de poder. O que acontece com a diferença detectada? Você consegue perceber que algumas vezes teremos que absolver o nosso réu por falta de provas?



Mas eu tenho certeza que agora você está se perguntado: bom, se isso é uma área sob uma curva, deve ter uma daquelas cobrinhas (nome "carinhoso" da integral) que representa essa joça, não é mesmo? É mesmo! Tanto que eu vou deixar essa como exercício para vocês!!!

Tamanho da amostra

Por último, vamos ver um pouco mais sobre o tamanho da amostra, que como você já notou influencia bastante o poder do teste. Isso tudo é muito bonito, mas na prática mesmo de um desenho de estudo, a decisão final que terá que ser tomada é quanto ao tamanho da amostra a ser coletada da sua população. Isso porque, como você deve ter notado, os outros parâmetros que influenciam nessa nossa equação de poder são de uma maneira ou de outra chutados. Vamos ver a equação do poder novamente. Eis uma de suas representações, para um teste bilateral para a média. Encontrado em qualquer livro de estatística:

$$1-\beta = \Phi\left[-z_{1-\alpha/2} + \frac{|\mu_0 - \mu_1|\sqrt{n}}{\sigma}\right]$$

Essa equação envolve todos os parâmetros aos quais nos referimos anteriormente. Tente identificá-los (repare que nós usamos essa equação, com uma pequena modificação). Ah, lembre-se que $\Phi(z)$ é a área de uma normal desde menos infinito até o ponto *z*, ou seja, é aquela cobrinha... Alguma semelhança com uma função do R?

Bem, a partir desta equação, podemos "facilmente" isolar o valor do tamanho da amostra, *n*. Para não dizer que sou muito exigente, o primeiro passo é aplicar a função inversa à $\Phi(z)$ em ambos os lados da equação, o que daria:

$$z_{1-\beta} = -z_{1-\alpha/2} + \frac{|\mu_0 - \mu_1|\sqrt{n}}{\sigma}$$

Bem, daqui para frente ficou fácil obter uma equação para o *n*, não é mesmo?

Mas afinal, por que eu digo que os parâmetros serão chutados? É simples: porque na prática, você não sabe a verdadeira média nem a verdadeira variância – se soubesse não precisaria da estatística, não é mesmo? Como o alfa é fixo e você tem que chutar os outros parâmetros, o único cálculo interessante na prática é o tamanho de sua amostra para obter-se um teste com um determinado poder, chutados os demais parâmetros.

Observe também nessa equação que o que importa de fato é a diferença entre as médias que estamos testando e não os seus valores propriamente ditos. Assim, tanto faz estarmos comparando 100 mmHg com 110 mmHg ou 95 mmHg com 105 mmHg, que mantidos os outros parâmetros fixos, o tamanho da amostra será o mesmo.

De fato, ao desenhar um estudo, o que se faz em geral é apresentar diversos cenários para tamanhos de amostra diferentes, e ver como o poder desse teste vai variar. É verdade que a diferença que se quer detectar também pode entrar na questão, especialmente se você não tem a menor idéia de qual seja essa diferença. É claro também que no caso de um ensaio clínico por exemplo, uma diferença que seja clinicamente significativa seria a de interesse para você.

O principal motivo para isso é que coletar informações de pessoas custa tempo e dinheiro, e o que queremos, em um mundo com recursos escassos (que papo de economista, hein?) é o melhor custo-benefício em termos de poder e tamanho de amostra, isto é, um tamanho de amostra que não seja muito caro de se obter tanto em termos de tempo quanto de dinheiro, mas que ao mesmo tempo seja capaz de responder à pergunta proposta pelo estudo de forma adequada.

Bem, nesse caso seria interessante termos uma função para calcular o tamanho de uma amostra não é mesmo? Então vamos lá...

```
tamanho.amostra <- function(alfa=0.05, poder=0.8, dif, var, bilateral=T)
{
      if (bilateral){
            zalfa<-qnorm(1-(alfa/2))
            }else{
            zalfa<-qnorm(1-alfa)
      }
      ceiling(((qnorm(poder)+zalfa)^2)*var/(dif^2))
}
```

Essa é uma função bem simples, apenas para o caso de uma amostra que nós vínhamos discutindo, mas funciona. Experimente um tamanho de amostra para o nosso exemplo:

```
tamanho.amostra(alfa=0.05, poder=0.8, dif=10, var=625, bilateral=T)
```

Muito bem. Repare que nós mantivemos aqui a nossa variância conhecida, para podermos usar a Normal em vez da distribuição *t*. Apesar de existirem métodos (inclusive implementados no R) para um cálculo mais preciso de tamanho de amostra (e conseqüentemente do poder também), em geral essas aproximações pela Normal são usadas e funcionam bem, especialmente se estivermos falando de tamanhos de amostra suficientemente grandes (por que será?) Esses métodos mais específicos serão abordados na próxima aula... Aguardem!

Apesar de nós só termos abordado o caso de uma amostra para a diferença de médias, toda esta teoria sobre teste de hipóteses, poder e tamanho de amostra pode e deve ser aplicada para outros problemas, como testes pareados, testes para mais de uma amostra, testes para proporções,

etc. Como o raciocínio é o mesmo, se você entendeu como funciona, a extrapolação para as demais situações é bastante intuitiva também.

Simulações

Mas nem tudo são flores... Nem sempre é possível obter-se funções algébricas para calcular tamanhos de amostra para determinados problemas, como vocês poderão se deparar no futuro... Em situações como essas, podemos lançar mão de resultados aproximados, ou então usar um recurso que vem sendo empregado mais e mais freqüentemente em estatística que é o uso de simulações para o cálculo de poder de um teste.

Neste caso, faz-se o caminho inverso: nós simulamos amostras sob a hipótese alternativa para vários tamanhos de amostra diferentes e calculamos o poder obtido para cada um desses tamanhos de amostras. Assim, é possível, para um determinado poder, estabelecermos um tamanho de amostra adequado.

Mas como isso funciona, afinal de contas? Não dá para entender muito bem como isso funciona, não é mesmo? Mas é mais simples do que parece. Vamos lá: queremos calcular o poder do teste, certo? Isto significa que quero saber com que freqüência o meu teste é capaz de rejeitar a hipótese nula, quando a hipótese alternativa é verdadeira (um acerto).

Muito bem, na simulação nós fazemos o caminho inverso: nós criamos uma base de dados que será uma amostra de uma população que tenha os parâmetros de uma hipótese alternativa. Complicou? Vamos ver o nosso caso: quero testar se a PAM é diferente de 100mmHg. Bom, para uma determinada hipótese alternativa, por exemplo 106mmHg, nós vamos gerar amostras de tamanhos diversos a partir de uma Normal (106, 625), que é a distribuição sob a hipótese alternativa e então vamos usar o nosso teste para ver se ele é capaz de detectar a diferença (que de fato existe.) Com isso teremos uma freqüência que corresponde mesmo ao poder desse teste para esta diferença.

Vamos então implementar uma simulação no R, para esse problema. Vamos pensar: o poder do teste é a capacidade desse teste detectar uma real diferença quando ela realmente existe. Bem, baseado no que já vimos em termos de testes de hipóteses, uma das maneiras de se fazer isso, é calcular o p-valor e rejeitar H_0 para valores menores que 0.05, por exemplo. Ora, então, isso quer dizer que se eu testar um valor fixo (como fizemos aqui, de 100 mmHg, nossa H_0) contra uma distribuição que seja gerada de uma população com uma média de, digamos 106 mmHg, para diferentes tamanhos de amostra, teremos diferentes probabilidades de rejeitarmos H_0 (pois nesse caso, estamos assumindo que de fato a média é mesmo 106 mmHg)

Como poderíamos então fazer isso? Bem, uma maneira é simular várias vezes uma amostra de tamanho variável (mas conhecido, estabelecido por nós), aplicar o teste estatístico que empregaremos no nosso estudo e contar quantas vezes o p-valor desse teste será menor que 0.05. Se dividirmos esse número pelo número total de simulações, estaremos calculando a freqüência com a qual o teste rejeita a hipótese nula ($p < 0.05$), quando a hipótese alternativa é verdadeira (a amostra é simulada sob a hipótese nula), ou seja, teremos o poder desse teste.

Vamos então criar uma função para simular as nossas pressões a partir de uma Normal com média igual à média sob a hipótese alternativa e com a variância da população – $625mmHg^2$ e então calcular quantas vezes o p-valor para um teste *t* vai ser menor que 0.05. O *default* para o número de simulações será 1000. Veja o código abaixo:

```
poder.sim <- function(x, k=1000, H0, H1, sd)
{
      #Iniciando as variáveis
      p <- 0
      poder.t<-0
      #Loop para vários tamanhos de amostra
      for (m in 1: length(x))
      {
            #Loop para o número de simulações
```

```
                  for (i in 1:k)
                  {
                        data <- rnorm(x[m], mean=H1, sd=sd) #Geração da amostra
sob H1
                        p[i] <- t.test(data, mu=H0)$p.value #Cálculo do p-valor
para o teste t
                  }
            poder.t[m] <- sum(p<=0.05)/k #Cálculo do Poder para cada tamanho de
amostra - o número de vezes que o p-valor é menor que 0.05, dividido pelo número
de simulações
            }
            plot(x, poder.t, ylab="Poder", xlab="n", type="b") #Plotando a curva
de poder
            cbind(x, poder.t) #Retornando os tamanhos e os poderes
      }
```

Agora podemos calcular uma curva de poder para o nosso problema. Vamos tentar então calcular a nossa curva para amostras de tamanhos 25 até 350 com incremento de 25 em 25, para as nossas hipóteses H_0=100mmHg e H_1=106mmHg:

```
poder.sim(seq(25,350,25), k=1000, H0=100, H1=106, sd=25)
```

Possivelmente a sua curva não é tão redondinha quanto uma curva calculada a partir de uma conta exata, como fizemos anteriormente. Acontece que para a distribuição *t* também não existe uma equação exata muito tratável, apesar de existir um modo de fazê-lo, e que é descrito no livro-texto, para quem estiver interessado. O que a maioria dos livros descreve geralmente é admitir que a variância da população é conhecida e aproximar os resultados usando a Normal mesmo para calcular o poder. Vamos então comparar com os poderes usando a Normal para o nosso caso, no próprio gráfico que acabamos de criar, assumindo a variância da população como 625mmHg^2:

```
x<-seq(25,350,25)
lines(x, pnorm(qnorm(0.975, mean=100, sd=sqrt(625/x)), mean=106,
sd=sqrt(625/x), lower.tail=F), pch=20, type="b")
```

E então, os resultados são bem parecidos?

Exercícios

1. Explique porque o juiz indeferiu o pedido de se utilizar a variância da população geral como evidência para o nosso julgamento.

2. Calcule o IC 95% para a média amostral calculada no nosso julgamento. Apresente o código do software que você utilizou ou as contas feitas à mão. Explique cada um dos passos do cálculo. Obs.: Repare que o vetor `pam.idosos` foi gerado aleatoriamente por cada um de vocês. Portanto, espera-se que os resultados sejam diferentes para cada aluno.

3. Para o nosso exemplo da PAM, apresente uma proposta de tamanho de amostra para um estudo como esse, discutindo diferentes poderes e diferentes possíveis diferenças.

4. Escreva as equações matemáticas que correspondem ao código `((qnorm(poder)+zalfa)^2)*var/(dif^2)` na função para calcular o tamanho de uma amostra acima. Dica: o plural está bem empregado, pois é uma equação para o caso bilateral e outra para o unilateral. Observe o código acima, para tirar (ou aumentar?) a dúvida:

```
if (bilateral){
            zalfa<-qnorm(1-(alfa/2))
            }else{
            zalfa<-qnorm(1-alfa)
```

5. Vimos na aula uma das maneiras de escrevermos a equação para obter o poder de um teste: $1-\beta=\Phi\left[-z_{1-\alpha/2}+\frac{|\mu_0-\mu_1|\sqrt{n}}{\sigma}\right]$. Identifique nessa equação cada um dos elementos presentes, descrevendo o significado de cada um deles (i.e. descreva cada uma das letrinhas dessa equação). Dica: essa equação tem uma integral disfarçada. Aponte onde ela está e o que ela representa também.

6. Utilize o banco de dados velho conhecido nosso, `juul` e teste se a média de IGF-I é diferente de 330 µg/l. Indique qual é o teste de hipóteses e interprete a saída do programa que você utilizou das três maneiras possíveis: comparação com um valor crítico, p-valor e intervalo de confiança.

Questão extra (bônus de 0.1 ponto):

Use simulações para desenhar uma curva de poder para diversos tamanhos de amostra para um teste de hipóteses qualquer, envolvendo uma distribuição *t* e escolha um tamanho de amostra adequado para este estudo. Isto significa que você vai estabelecer o que testar e que pressuposições são necessárias. Indique também explicitamente o teste de hipóteses a ser realizado. Dica: Use a função para simulação de poder que nós criamos para ajudar você a escolher o tamanho da amostra. Obs.: **Não use o mesmo exemplo da questão 3.**

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 5 - Testes para uma e duas amostras (contínuos)
Livro: páginas 81 a 93

A partir dessa aula vamos abordar diferentes tipos de testes usados em situações específicas comumente encontrados no dia-a-dia da Epidemiologia e da Bioestatística.

Na presente aula vamos conversar sobre testes para inferir basicamente diferenças entre médias de variáveis contínuas, seja como nós já vimos, apenas com uma amostra ou com duas amostras. Além disso, alternativas não-paramétricas, isto é, estatísticas que não assumem uma distribuição conhecida dos parâmetros também será abordada. O caso de mais de duas amostras para variáveis contínuas será abordado na aula sobre ANOVA

Teste *t* para uma amostra
Teste do sinal do posto de Wilcoxon
Teste *t* para duas amostras
    Comparação de variâncias
    Testes para normalidade
Teste de Wilcoxon para duas amostras
Teste *t* pareado
Teste de Wilcoxon pareado
Poder e tamanho de amostra para testes *t*
Exercícios

Teste *t* para uma amostra

Bem, o teste *t* para uma amostra não deve apresentar problemas ou surpresas para você, já que vínhamos usando esse teste como exemplos nas duas últimas aulas. Iremos apenas formalizar as coisas aqui e também descrever um pouco melhor a saída da função `t.test()`, que é usada também para esse fim, como já fizemos anteriormente.

Recordando um pouco, o teste *t* será usado na esmagadora maioria dos casos, pois em geral a variância da população não é conhecida, e teremos que lançar mão da variância amostral. O conceito básico de um teste *t* é o cálculo do erro padrão da média (EPM), que nada mais é do que o desvio-padrão da média amostral que nós já exploramos bastante. Recordando:

EPM = $s/\sqrt{(n)}$

Não creio que haja dúvidas a respeito do EPM acima. O próximo conceito é o cálculo da estatística *T*, que é baseado, como você deve estar lembrado também no EPM. Nós já fizemos esse cálculo para o teste *t* para uma amostra, mas sem usar o termo EPM. Recorde:

$$T = \frac{\bar{x} - \mu_0}{EPM} = \frac{\bar{x} - \mu_0}{s/\sqrt{(n)}}$$

E é claro que toda aquela teoria que nós já discutimos para testes de hipóteses, p-valores e ICs se aplicam aos nossos testes, e portanto não vamos discutir esse assunto novamente. Vamos partir para um exemplo prático, só para reforçar também.

Vamos usar um banco de dados do pacote `ISwR` como exemplo. O banco chama-se `intake` e contém informações sobre ingesta calórica (em kilojoules) antes (primeira coluna) e após (segunda coluna) a menstruação de 11 mulheres, valores esses pareados para cada uma dessas mulheres.

Vamos chamar e inspecionar esse banco:

```
library(ISwR)
data(intake)
intake
```

Bem, como por enquanto vamos trabalhar com uma amostra, vamos inicialmente usar somente a primeira coluna desse banco. Selecione este vetor:

```
energia<-intake[,1]
```

Aproveite para fazer algumas estatísticas-resumo deste vetor...

Bem, um problema interessante que podemos testar é se a ingesta dessas mulheres difere significativamente ou não no período pré-menstrual do valor recomendado, de 7725 kJ. Vamos assumir que essa amostra foi tirada de uma distribuição Normal e aplicar o nosso já conhecido teste *t*:

```
> t.test(energia, mu=7725)

        One Sample t-test

data:  energia
t = -2.8208, df = 10, p-value = 0.01814
alternative hypothesis: true mean is not equal to 7725
95 percent confidence interval:
 5986.348 7520.925
sample estimates:
mean of x
 6753.636
```

Vamos então explicar agora cada parte da saída desse teste:

```
        One Sample t-test
```

Bem, essa é só a descrição do tipo de teste empregado. Repare que o R entende, só por você ter colocado a opção `mu=7725` que se trata de um teste *t* para uma amostra.

```
data:  energia
```

Sem problemas aqui também. Trata-se apenas do nome do banco de dados que foi usado pela função. Pode ser útil em algumas situações.

```
t = -2.8208, df = 10, p-value = 0.01814
```

Aqui começa a brincadeira. Nós temos a estatística *T* (`t = -2.8208`), os graus de liberdade da distribuição (`df = 10`) e o p-valor associado a essa estatística (`p-value = 0.01814`). Na verdade esta linha basta para você interpretar o seu teste de hipóteses, não é mesmo? Lembra da relação entre o p-valor e o teste de hipóteses? Pois é, mas é claro que você poderia calcular o valor crítico e compará-lo com o valor de *T* como nós fizemos anteriormente.

```
alternative hypothesis: true mean is not equal to 7725
```

Aqui o R nos informa que o teste é bilateral (bicaudal) – isso por causa do `not equal to` que significa diferente e não maior ou menor que. Temos também a informação sobre o valor contra o qual estamos comparando essa amostra, `7725`.

```
95 percent confidence interval:
 5986.348 7520.925
```

E é claro que podemos ainda usar o IC 95% para esse fim também. Ora, se o valor testado (7725) não estiver contido no IC 95% para a média da amostra, por causa da mesma relação, o p-valor será < 0.05 e a hipótese nula será rejeitada, como é o caso. Lembra-se como se calcula esse IC?

```
sample estimates:
mean of x
 6753.636
```

E aqui está a estimativa da média, ou seja a média amostral do nosso vetor. Você gostaria de ver alguma outra estimativa aqui para o nosso problema?

Se você foi curioso o bastante, já deve ter consultado a ajuda do R para saber mais sobre a função `t.test()`, que é realmente muito usada em estatística e conseqüentemente por nós também. Você deve então ter notado que ela apresenta diversos argumentos possíveis para vários tipos de testes e também opções de testes.

Neste caso, para definir o teste como para uma amostra, o argumento usado foi o `mu=7725`, mas outros argumentos, dos quais nós usamos os *defaults*, definem características do teste em questão. O primeiro é a direção desejada do teste, cujo valor-padrão é`"alternative=”two.sided”`, o que nós exatamente usamos. Se quiséssemos um teste unidirecional, poderíamos lançar mão das alternativas, que tenho certeza que você já procurou na ajuda para esse teste (caso contrário, faça-o agora.)

O outro parâmetro é o nível de confiança que desejamos usar para o teste. O valor padrão, como você deve estar imaginando é `conf.level = 0.95`. Mas se quiséssemos poderíamos usar outros níveis. Por exemplo, tente fazer um teste bilateral para um alfa de 0.01 e veja o que acontece...

Teste do sinal do posto de Wilcoxon

Para início de conversa, que nomezinho mais infeliz, esse não é mesmo? Pois é, mas o nome original dele é “Wilcoxon signed-rank test” e se alguém vier com uma tradução melhorzinha, será bem-vinda.

Bem, esse teste pertence a uma família de testes chamados testes não-paramétricos, também conhecidos como livres de distribuição. Significa que nós não precisamos assumir uma distribuição qualquer para os parâmetros que estamos querendo estimar.

Ué, mas então como é que isso funciona? Calma. Ele é baseado no que chamamos de estatísticas de ordem. É o seguinte: lembra da mediana e dos quartis, quantis, etc? Pois é, eles são chamados de estatísticas de ordem porque eles criam uma ordenação ou um *rank* das observações. A partir dessas ordens ou postos (*ranks*) é possível então fazer-se inferências sobre o nível de uma variável qualquer.

Peraí, que história é essa de “nível”? Nós não estávamos falando de médias??? Pois é, aqui há uma discussão. Muito embora à rigor os testes baseados em estatísticas de ordem estejam testando a diferença de ordens mesmo, muitos autores tomam a liberdade de dar um salto e dizer que estes testes estariam mesmo inferindo diferenças entre médias.

A rigor, o que estamos realmente testando são diferenças de medianas, que é umas dessas ordens, mas que têm a característica de dividir nossa distribuição em partes iguais, ou seja exatamente na metade. A fim de evitar confusões, eu acabo optando por ficar em cima do muro e dizer que estamos testando diferenças nos níveis (conveniente, não?)
;-)

Tudo muito bonito, mas para que servem estes testes afinal de contas? O teste *t* não é relativamente robusto em relação à distribuição Normal, por conta do TLC? Por que então eu usaria um teste desses? Bem, é que nem sempre o TLC vai ajudar muito, especialmente se estamos tratando de amostras pequenas.

Ah, mas isso é para essas variáveis que poderiam ser muito bem testadas pelo teste *t*, mas temos também situações onde a primeira opção seria mesmo um teste desse tipo. Alguém arrisca dizer que situação é essa?

Bem, mas vamos adiante. O R tem implementado também funções que realizam esses testes não-paramétricos. Vamos aplicar o teste que corresponde ao teste *t* para uma amostra, para o mesmo exemplo que utilizamos anteriormente:

```
> wilcox.test(energia, mu=7725)

        Wilcoxon signed rank test with continuity correction

data:  energia
V = 8, p-value = 0.0293
alternative hypothesis: true mu is not equal to 7725

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(energia, mu
= 7725)
```

Como você deve ter notado, a função que calcula esse tipo de teste se chama `wilcox.test()` e os argumentos que usamos são essencialmente os mesmos que tínhamos usado anteriormente. A saída da função também é bem semelhante, se bem que algo mais resumida.

Vamos ver a saída em maior detalhe.

```
        Wilcoxon signed rank test with continuity correction

data:  energia
```

Essa parte é igualzinha ao do teste *t*, exceto pelo nome do teste que é diferente e por essa correção de continuidade, que veremos mais tarde o que significa.

```
V = 8, p-value = 0.0293
```

Essa é a parte da saída que nos interessa, com a estatística calculada e com o p-valor que corresponde a esse valor para essa estatística. Vamos ver o funcionamento disso mais adiante.

```
alternative hypothesis: true mu is not equal to 7725
```

Também igual ao teste *t* anteriormente.

```
Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(energia, mu
= 7725)
```

E aqui aparece um problema inerente ao teste não-paramétrico. É a questão dos empates nos postos (*ranks*). Como nós veremos mais tarde, para esse teste é possível calcular o valor exato,

construindo a distribuição da estatística *V*. Porém quando o número de *ranks* é muito elevado ou na presença de empates, esse cálculo exato fica bem mais complicado. No primeiro caso, essa distribuição pode ser muito bem aproximada pela Normal, já que é uma distribuição simétrica em torno de μ_0 . Mas para o caso dos empates, e para um número reduzido de observações, isso não é verdade e muitas vezes o resultado assintótico pode ficar prejudicado.

Repare que a função `wilcox.test()` não calculou o valor exato por causa de empates, apesar do tamanho da amostra ser bastante reduzido.

Mas vamos votar à nossa estatística *V* e tentar entender o que ela significa. Ele representa a soma dos *ranks* positivos da diferença entre a nossa amostra e o valor a ser testado. Ihhh, complicou bem... Vamos começar vendo a diferença qual é:

```
> energia-7725
 [1] -2465 -2255 -2085 -1545 -1335 -1210  -920  -210  -210   505  1045
```

Repare que das 11 observações, temos duas diferenças positivas e as demais negativas. Repare também que temos um empate: dois valores são -210. Bem, o próximo passo é ordenarmos essas diferenças, **sem considerar o sinal**, ou seja, vamos ordenar os seus valores *absolutos*. No R podemos fazer assim:

```
> rank(abs(energia-7725))
 [1] 11.0 10.0  9.0  8.0  7.0  6.0  4.0  1.5  1.5  3.0  5.0
```

Tudo muito bonito, mas o que diabos são esses dois 1.5 que aparecem no nosso vetor??? Isso ocorreu justamente por causa do empate que nós tínhamos. Quando acontece um empate, atribui-se o *rank* médio aos elementos empatados, o que seria apenas a média de fato. Nesse caso, em termos absolutos, o 210 seria o primeiro elemento, *rank* número 1. como temos um segundo 210, que seria o *rank* número 2, temos que fazer (1 + 2)/2 = 1.5.

Muito bem, agora a nossa estatística *V* será a soma dos *ranks* para os valores cujas diferenças haviam sido originalmente positivas. Isso pode ser feito "facilmente" no R:

```
> sum(rank(abs(energia-7725))[energia-7725>0])
[1] 8
```

Conseguiu entender esse código? Não? Então tente mais uma vez... Pode ser um exercício interessante. Repare que estamos selecionando do vetor de *ranks* os valores cuja diferença é positiva (`[energia-7725>0]`) e somando esse resultado.

Conferiu com o valor achado pelo R lá na saída?

Bom, o resto da teoria do cálculo dos p-valores ou áreas de rejeição para esta estatística, deixo para a curiosidade de cada um procurar em um livro-texto.

A questão da aproximação pela Normal, em geral, para um número de diferenças não nulas (i.e. diferentes de zero) de pelo menos 16 os resultados são bastante satisfatórios. Para os demais resultados, existem valores tabelados para valores críticos do teste de Wilcoxon.

Só para lembrar da aula teórica, sob a hipótese nula, a estatística *V* tem média $\frac{n+(n+1)}{4}$ e variância $\frac{n(n+1)(2n+1)}{24}$. Bem, se houver empates, aí temos que descontar um valor dessa variância, cujo cálculo é um pouco complicado, e que veremos com mais detalhes no teste para duas amostras.

Teste *t* para duas amostras

Como você já deve ter depreendido da aula teórica, o teste *t* para duas amostras não difere muito do teste para uma amostra apenas. A diferença é que testaremos a diferença entre duas

médias diferentes, entre duas amostras independentes e não mais a diferença entre uma média de uma amostra e um número fixo. É claro que isso trará conseqüências em termos de cálculos, mas não em termos de conceito.

O cálculo da estatística *T* nesse caso deve ser feito levando-se em conta ambas as médias a serem testadas, e também ambas as variâncias das médias amostrais. De maneira geral:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

O denominador dessa equação é o erro-padrão da diferença das médias, que nós podemos apelidar aqui de s_{dif}.

A questão é que classicamente se usa a condição onde as variâncias das amostras são homogêneas, situação onde é possível calcular-se uma variância conjunta (é na verdade uma média ponderada das variâncias de cada grupo.) Isso era bastante importante antigamente porque, para calcular esses resultados na mão essa pressuposição é tratável. Assim, se assumirmos a homogeneidade, podemos calcular a variância conjunta:

$$s_{conj}^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

E então poderíamos reduzir a nossa equação para:

$$T = \frac{\bar{x}_1 - \bar{x}_2}{s_{conj}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Acontece que nem sempre é possível obter-se variâncias homogêneas e aproximações para esses casos foram desenvolvidas para podermos obter resultados confiáveis mesmo sem essa pressuposição. Uma dessas aproximações (existem outras) foi descrita por Welch, e é implementada no R. O que acontece é que nós simplesmente usamos a equação lá de cima, para calcular o *T*, que passa a não ser mais distribuído como uma $t_{n_1+n_2-2}$ mas a sua distribuição pode ser aproximada por uma *t* com ν' graus de liberdade calculados assim:

$$\nu' = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1 - 1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{s_2^2}{n_2}\right)^2}$$

Bem, vamos usar então um banco de exemplo do `ISwR` para estudar o nosso teste *t*. O banco `energy` contém o gasto energético de mulheres obesas (*obese*) e magras (*lean*), em Megajoules (MJ).

```
data(energy)
attach(energy)
```

Verifique a disposição dos dados nesse *data frame*. Repare que nós temos uma variável para os valores e outra para a classificação dos grupos (*obese* e *lean*). Vamos então usar a nossa função `t.test()` nesse banco:

```
> t.test(expend~stature)

        Welch Two Sample t-test

data:  expend by stature
```

```
t = -3.8555, df = 15.919, p-value = 0.001411
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.459167 -1.004081
sample estimates:
 mean in group lean mean in group obese
           8.066154           10.297778
```

Repare que nós usamos o til para indicar que a variável `expend` está sendo classificada pela variável `stature`, da mesma maneira que nós usamos no *boxplot*, lembra?

Repare também que o *default* do R é usar mesmo o procedimento de Welch, para variâncias heterogêneas. As únicas diferenças dessa saída para a que nós já vimos anteriormente são:

```
data:  expend by stature
```

Indicando que é uma variável classificada pela outra.

```
t = -3.8555, df = 15.919, p-value = 0.001411
```

Os graus de liberdade que devem ser calculados segundo aquela equação lá em cima.

```
alternative hypothesis: true difference in means is not equal to 0
```

O teste que nesse caso é a diferença das médias ser ou não igual a zero

```
sample estimates:
 mean in group lean mean in group obese
           8.066154           10.297778
```

E duas estimativas de médias, uma para cada grupo a ser comparado.

Vamos agora ver como o R calcula esse teste de Welch. Bem, primeiramente vamos calcular os graus de liberdade, segundo a equação acima:

```
s1 <- var(expend[stature=="obese"])
s2 <- var(expend[stature=="lean"])
n1 <- length(expend[stature=="obese"])
n2 <- length(expend[stature=="lean"])
gl <- ((s1/n1)+(s2/n2))^2 / (((1/(n1-1))*(s1/n1)^2) + ((1/(n2-
1))*(s2/n2)^2))
gl
```

Veja se não bate com o resultado lá de cima... Se quiser pode fazer isso tudo na mão também, mas eu recomendaria que você tentasse identificar a equação acima nesse código. Você seria capaz de explicar o que foi feito antes de calcularmos os graus de liberdade propriamente ditos?

Bem, agora falta ainda calcular a estatística *T*. Mas isso é fácil, também baseado na fórmula mais geral lá de cima:

```
media1 <- mean(expend[stature=="obese"])
media2 <- mean(expend[stature=="lean"])
T <- (media1-media2)/sqrt((s1/n1)+(s2/n2))
T
```

Algo de errado com o valor de *T*? Está com o sinal trocado? Isso faz alguma diferença? Como você obteria o resultado com o mesmo sinal?

Bom, já que o resultado bate com o esperado, agora ficou fácil obter o p-valor no R, não? É a mesma coisa que para uma amostra:

```
> 2*pt(T, df=gl, lower.tail=F)
[1] 0.001410692
```

Ora, não é que bateu direitinho... Ah, se fosse fazer com o resultado de *T* negativo, precisaríamos omitir o último argumento:

```
> 2*pt(-T, df=gl)
[1] 0.001410692
```

Claro que podemos também testar as nossas hipóteses baseado nesse mesmo teste. Para isso, vamos primeiro estabelecer:

H_0: $\mu_1 - \mu_2 = 0$

H_1: $\mu_1 - \mu_2 \neq 0$

E agora vamos usar o mesmo raciocínio que usamos na última aula: vamos calcular um valor crítico para essa nossa *t* sob a hipótese nula. Lembra como é? Vamos lá:

```
> qt(0.025, df=gl)
[1] -2.120785
> qt(0.025, df=gl, lower.tail=F)
[1] 2.120785
```

Lembrou? A diferença é que tivemos que usar os graus de liberdade corrigidos. Como a nossa *T* é maior que 2.12 (ou -*T* é menor que -2.12, não importa), vamos rejeitar a hipótese nula, para um alfa de 5%, corroborando o resultado que obtivemos através do p-valor.

Ah, já ia me esquecendo. Tem ainda o IC 95% para a diferença dessas médias. Vamos usar o nosso erro-padrão da diferença para a nossa notação (o denominador da equação lá em cima). É assim:

$$\text{IC 95\%} = \left(\bar{x}_1 - \bar{x}_2 - t_{\nu',0.975} s_{diff}, \bar{x}_1 - \bar{x}_2 + t_{\nu',0.975} s_{diff} \right)$$

Isso para o nosso procedimento de Welch, é claro – por isso estamos usando s_{dif} e ν'.

No R:

```
> media1-media2-qt(0.975, df=gl)*sqrt((s1/n1)+(s2/n2))
[1] 1.004081
> media1-media2+qt(0.975, df=gl)*sqrt((s1/n1)+(s2/n2))
[1] 3.459167
```

Bateu? Não? Ihhhh, esse sinal...

Mas é claro que o R permite que você use também o método clássico para variâncias supostamente iguais. Basta um argumento para isso:

```
>  t.test(expend~stature, var.equal=T)

        Two Sample t-test

data:  expend by stature
t = -3.9456, df = 20, p-value = 0.000799
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.411451 -1.051796
sample estimates:
 mean in group lean mean in group obese
```

```
          8.066154          10.297778
```

Repare que agora houve uma mudança em uma das saídas:

```
t = -3.9456, df = 20, p-value = 0.000799
```

Todos esses valores são diferentes, pois *T* foi calculado a partir da variância conjunta, os graus de liberdade da *t* são um número inteiro (usual) e iguais ao tradicional $n_1 + n_2 - 2$ e o p-valor é evidentemente calculado a partir desta distribuição e deste valor de *T* e portanto também é bastante diferente. Nossa conclusão sobre o teste de hipóteses não mudou, porém.

Já sei: você está achando que eu vou repetir toda aquela contaria que eu fiz para o procedimento de Welch agora para o método clássico, não é? Se enganou, vai ficar para um longo exercício...

Comparação de variâncias

Bem, de qualquer maneira, precisamos ter um meio de testar se as variâncias são homogêneas ou não, para podermos usar o teste clássico. O R oferece como opção o teste *F* para razão de variâncias. O procedimento no R é bem simples:

```
> var.test(expend~stature)

        F test to compare two variances

data:  expend by stature
F = 0.7844, num df = 12, denom df = 8, p-value = 0.6797
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.1867876 2.7547991
sample estimates:
ratio of variances
          0.784446
```

Esse teste é bem intuitivo e ele testa se a razão das variâncias das amostras é ou não igual a 1 (caso em que elas seriam iguais, obviamente.) Ele é baseado no fato de uma razão de variâncias seguir uma distribuição *F* de Snedecor. Isso não é difícil de entender se você souber que na verdade a distribuição *F* é uma razão de duas distribuições χ^2_{gl} divididas pelos seus respectivos graus de liberdade. Além disso, a distribuição *F* tem como parâmetros justamente os graus de liberdade do numerador e do denominador, que correspondem aos graus de liberdade das χ^2_{gl} do numerador e do denominador dessa razão, respectivamente:

$$F_{gl_1, gl_2} = \frac{\frac{\chi^2_{gl_1}}{gl_1}}{\frac{\chi^2_{gl_2}}{gl_2}}$$

Vamos ver como isso funciona então. Como você deve se lembrar, a variância da amostra segue uma distribuição χ^2_{n-1}, se for feito um pequeno ajuste. Lembra?

$$\frac{(n-1)\times s^2}{\sigma^2} \sim \chi^2_{n-1}$$

Então, para comparar duas variâncias, podemos fazer o seguinte: vamos comparar as variâncias s_1^2 e s_2^2 de duas amostras, inicialmente fazendo uma divisão de duas expressões como a acima:

$$\frac{\frac{(n_1-1)\times s_1^2}{\sigma_1^2}}{\frac{(n_2-1)\times s_2^2}{\sigma_2^2}}$$

Bem, por enquanto, nada podemos dizer desta expressão acima, apenas que representa a divisão de duas distribuições Qui-quadradas. Mas se nós dividirmos o numerador por n_1*-1* e o denominador por n_2*-1*, que são os graus de liberdade dessas distribuições, teremos o que estávamos procurando, ou seja, uma F_{n_1-1,n_2-1} :

$$\frac{s_1^2/\sigma_1^2}{s_2^2/\sigma_2^2} \sim F_{n_1-1,n_2-1}$$

Mas há um outro detalhe: o tipo de teste de hipóteses que vamos fazer neste caso é seguinte:

H_0: $\sigma_1^2/\sigma_2^1 = 1$

H_1: $\sigma_1^2/\sigma_2^1 \neq 1$

Ora, isso quer dizer que sob a hipótese nula, $\sigma_1^2 = \sigma_2^2$ e podemos então cancelá-los na expressão acima. Sendo assim, a razão entre s_1^2 e s_2^2 segue uma distribuição F_{n_1-1,n_2-1} . No nosso caso:

```
> s2/s1
[1] 0.784446
```

O mesmo resultado da saída lá em cima. Mas e o nosso p-valor? Como ele foi calculado? Antes disso, vamos ver o jeitão dessa distribuição *F*

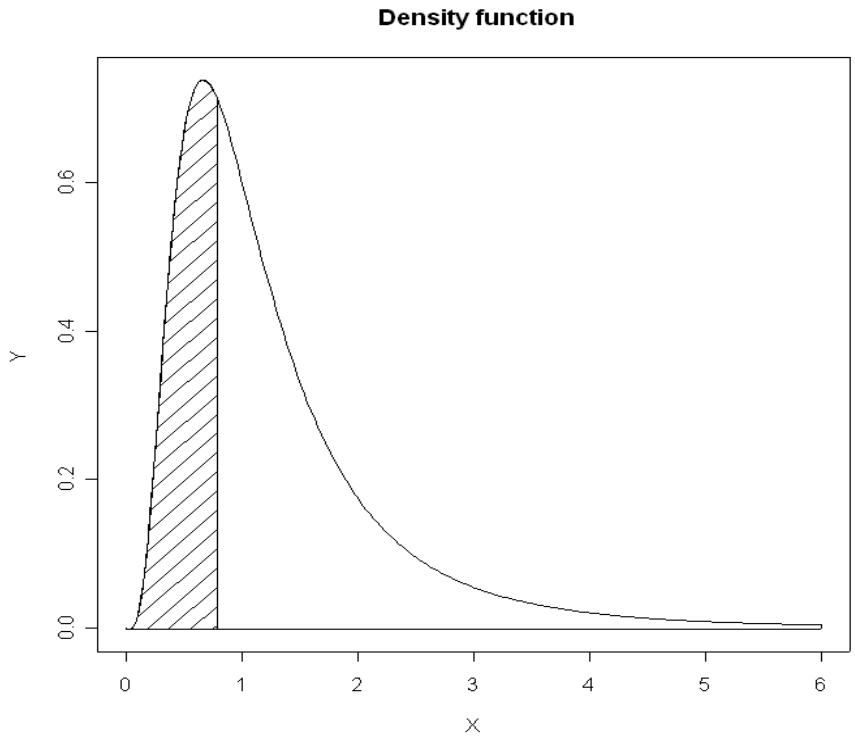
```
curve(df(x, df1=n2-1, df2=n1-1), from=0, to=5)
abline(v=1)
```

Repare que nós estamos testando uma igualdade com a unidade e não com zero. Bem, se a estatística obtida for menor que 1, estaremos procurando um p-valor que vai desde zero (limite inferior da *F*) até este ponto e depois, claro, multiplicar por dois, para obtermos um p-valor bicaudal. Veja no gráfico abaixo.

A área hachurada representa a área de zero até 0.784, o valor da nossa estatística. Para calcular essa área, basta então usarmos uma função para calcular a função de densidade acumulada de uma *F* com esses parâmetros e multiplicar por 2:

```
> 2*pf(s2/s1, df1=n2-1, df2=n1-1)
[1] 0.679746
```

Confere? Ótimo. Repare que tanto faz qual variância está no numerador e qual está no denominador. Você vai conferir como o resultado é o mesmo como um exercício.



Testes para normalidade

Muito embora o teste *t* seja bastante robusto em relação à normalidade da população de onde provém a amostra, especialmente pra um tamanho de amostra suficientemente grande, muitas vezes desejamos verificar se a amostra vem ou não de uma distribuição Normal.

Nós já aprendemos a fazer essa verificação visualmente, através especialmente do Q-Q plot, mas ainda não definimos nenhum teste formal para isso.

Um dos testes mais usados para esse fim é o teste de Shapiro-Wilk, que nada mais faz do que testar a linearidade de um Q-Q plot. O R tem uma implementação desse teste. Vamos ver como se comporta o nosso banco `energy` para cada uma das classes:

```
> shapiro.test(expend[stature=='lean'])

        Shapiro-Wilk normality test

data:  expend[stature == "lean"]
W = 0.8673, p-value = 0.04818

> shapiro.test(expend[stature=='obese'])

        Shapiro-Wilk normality test

data:  expend[stature == "obese"]
W = 0.876, p-value = 0.1426
```

Repare que o grupo de magras tem um p-valor que rejeita a normalidade (essa é a hipótese nula), mas de maneira bem fronteiriça. Compare com o Q-Q plot para esses grupos e vejam se você se sentiria à vontade de usar um teste paramétrico para esse banco.

Teste de Wilcoxon para duas amostras

Assim como a idéia do teste *t* para duas amostras tem a mesma lógica do que o para uma amostra, o mesmo acontece com o teste não paramétrico. Para esse teste vamos descrever um pouco mais detalhdamente o processo de obtenção do p-valor, usando uma aproximação Normal.

Resumindo, você pode lançar mão de um teste não paramétrico caso não se sinta seguro em assumir a normalidade da população de onde veio a sua amostra, ou a sua amostra é pequena e você não está muito certo se o TLC vai mesmo ajudar.

Vamos ver um exemplo com a mesma base acima:

```
> wilcox.test(expend~stature)

        Wilcoxon rank sum test with continuity correction

data:  expend by stature
W = 12, p-value = 0.002122
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(x = c(7.53,
7.48, 8.08, 8.09, 10.15, 8.4,
```

Repare que essa saída é bastante parecida com aquela do teste para uma amostra somente. A diferença está na estatística calculada – agora é *W* e não mais *V*. Note também que o problema dos empates continuam a nos incomodar nesse caso também.

O cálculo da estatística *W* é um pouco diferente e consiste em ordenar as observações em ambos os grupos, independentemente do grupo ao qual a observação pertence e então calcular a soma dos *ranks* em um dos grupos, diminuindo-se a soma do mínimo teórico para esse mesmo grupo. Não faz diferença que grupo escolhemos, mas vamos adotar o mesmo que o R usou, que foi o grupo de magros. Primeiro vamos ordenar a variável `expend`:

```
> rank(expend)
 [1] 14.0  5.0  3.5  8.0  9.0 18.0 11.0 19.0  1.0  7.0 20.0 22.0  2.0 21.0
17.0  3.5 12.0 16.0 15.0  6.0 13.0 10.0
```

Note que existem de fato empates, denotados pelo *rank* 3.5 presente nesse vetor acima. Agora, temos que pegar a soma dos *ranks* no grupo de magros:

```
> soma.lean <- sum(rank(expend)[stature=='lean'])
> soma.lean
[1] 103
```

Bem, agora temos que diminuir esse valor da soma do mínimo teórico. Parece enigmático? Mas não é, isso significa apenas que esse é o caso onde todos os *ranks* desse grupo são os menores possíveis, logo esse valor nada mais é do que a soma de um vetor que vai de 1 até o tamanho do vetor formado apenas pelo grupo de magros, entendeu? É como se os todos os magros tivessem os menores valores para a variável `expend`. Vamos ver como fica:

```
> soma.teorica <- sum(1:length(expend[stature=='lean']))
> soma.teorica
[1] 91
```

Se você não entendeu o código acima, veja o que acontece dentro dessa soma:

```
> 1:length(expend[stature=='lean'])
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13
```

Entendeu? É só um vetor seqüencial de 1 até o número de observações do grupo de magros. Agora ficou fácil computar a estatística *W*, não é mesmo?

```
> soma.lean-soma.teorica
[1] 12
```

Muito bem. Agora vamos precisar de alguma teoria para podermos obter o p-valor que o R calculou. Sem entrar em detalhes, pode ser mostrado que para um tamanho de amostra suficientemente grande – estamos falando de pelo menos 10 obervações em cada grupo, o que *quase* acontece no nosso caso, teremos:

$$E(W) = \frac{n_1 n_2}{2}$$ onde, por convenção n_1 é o tamanho da amostra no grupo para o qual calculamos W e n_2 para o outro grupo. Além disso:

$$Var(W) = \left(\frac{n_1 n_2}{12} \right) (n_1 + n_2 + 1)$$

Existe um pequeno problema, porém. Na presença de empates, a média esperada de *W* não é alterada, mas a sua variância tem que sofrer uma correção, que consiste em subtrair um valor, calculado segundo o número de empates presentes na amostra. Vamos ver como fica, na presença de empates:

$$Var(W) = \left(\frac{n_1 n_2}{12} \right) \left[n_1 + n_2 + 1 - \frac{\sum_{i=1}^{g} t_i (t_i^2 - 1)}{(n_1 + n_2)(n_1 + n_2 - 1)} \right]$$ onde *g* é o número de grupos de *ranks* empatados e *t* é o número de valores empatados nesse grupo.

Ihhhh! Complicou, né? Mas é mais fácil do que parece. Isso só quer dizer que podemos ter mais de um grupo de valores iguais em uma mesma amostra. Por exemplo, podemos ter 2 valores iguais a, digamos 10. Se esses fossem os *ranks* 3 a 4, cada um deles receberia o valor 3.5. Esses dois valores seriam um grupo, com 2 valores empatados. Poderíamos então ter outro grupo de valores, por exemplo 20, que seriam os *ranks* de 10 a 13, e cada um deles receberia o valor de 5.75, e teríamos então 4 valores empatados nesse grupo.

Nós já desconfiávamos que havia pelo menos um grupo de empates no nosso banco. Vamos conferir:

```
> sort(expend)
 [1]  6.13  7.05  7.48  7.48  7.53  7.58  7.90  8.08  8.09  8.11  8.40
8.79  9.19  9.21  9.68  9.69  9.97 10.15 10.88 11.51 11.85 12.79
```

Parece que nós só temos mesmo um grupo, correspondendo ao valor 7.48, com o *rank* de 3.5 e com apenas 2 valores para eles. Ora, então a nossa conta vai ser bem simples, não é verdade? Vamos ver mais adiante...

Bem, com as informações com essas informações, fica fácil nós calcularmos um *z* de uma Normal e calcular um p-valor para esse escore. Vamos aos cálculos. Primeiro vamos colocar os nossos tamanhos dos grupos em vetores para facilitar a nossa vida:

```
n1 <- length(expend[stature=='lean'])
n2 <- length(expend[stature=='obese'])
```

Agora vamos calcular a média de *W*:

```
media<-(n1*n2)/2
```

Para facilitar a vida, vamos primeiro calcular aquele fator de correção lá de cima. Observe que não haverá soma alguma, pois só temos um único grupo, e que o valor de *t* é 2:

```
correcao<-(2*((2^2)-1))/((n1+n2)*(n1+n2-1))
```

Agora vamo à variância, não nos esquecendo de acrescentar (ou melhor, diminuir) o fator de correção para os empates:

```
variancia<-(n1*n2/12)*(n1+n2+1-correcao)
```

Bem, agora o cálculo de *z* ficou muito fácil , não é mesmo? Todos já cansamos de fazer isso:

```
> (12-media)/sqrt(variancia)
[1] -3.106057
```

E agora você deve estar pensando: basta eu ver a área sob a curva da Normal que corresponde a esse valor de *z*, multiplicar por 2 e obter o p-valor, certo? Bem, não é bem assim... Vamos tentar fazer isso:

```
> 2*pnorm((12-media)/sqrt(variancia))
[1] 0.001896004
```

Esse valor não bate com o p-valor achado por causa de um detalhe que ainda não vimos, mas que será abordado em uma outra aula. Repare o que essa linha da saída diz:

```
Wilcoxon rank sum test with continuity correction
```

É por causa dessa "correção de continuidade" que não obtemos o mesmo resultado. Rapidamente, o que isso significa é que como estamos aproximando uma distribuição discreta (a soma dos *ranks* são na verdade distribuídas como uma variável aleatória discreta), por uma distribuição contínua (a Normal), é necessário fazer uma correção para que o valor seja melhor aproximado. Na verdade essa correção é bastante fácil de se fazer: basta nós acrescentarmos meia unidade à diferença da média observada e da média teórica, sempre em direção à hipótese nula, i.e. se a diferença for negativa, vamos somar 0.5; se for positiva, vamos diminuir 0.5. Como já vimos que no nosso caso o *z* foi negativo, vamos somar:

```
> 2*pnorm((12-media+0.5)/sqrt(variancia))
[1] 0.002121613
```

Ufa! Chegamos ao resultado finalmente. Mas agora, alguém arrisca dizer como poderíamos fazer esta conta dar certo com a correção de continuidade, mesmo sem sabermos se o *z* é positivo ou negativo? Exercício à vista...

Teste *t* pareado

Testes pareados são empregados quando temos duas medidas em uma mesma unidade experimental. Um exemplo seria a mensuração de uma certa característica antes e após uma determinada intervenção, ou mesmo em dois períodos distintos no tempo, como o banco de dados que já vimos de aporte energético no período pré- e pós-menstrual (`intake`).

A idéia do teste é simplesmente subtrair uma observação da outra em uma mesma unidade e então tratar o teste como se fosse um teste *t* de amostra única, que já vimos anteriormente. Nesse caso, obviamente estaremos usando menos graus de liberdade, pois o nosso tamanho de amostra

será o número de unidades experimentais, ou seja, o número de pares e não o número de observações.

Existe porém um pré-requisito para a validade do teste que é a independência entre a distribuição das diferenças e os seus níveis, ou seja, a distribuição das diferenças não pode ter nenhum padrão definido em relação, por exemplo à média das observações dos pares. Aliás, um método gráfico usado para verificar este fato é exatamente um gráfico de dispersão (*scatter plot*) exatamente assim, conhecido como gráfico de Bland-Altman.

Vamos usar então o banco `intake` para esse nosso exemplo:

```
data(intake)
attach(intake)
```

Verifique o seu conteúdo e note que são as mesmas 11 observações que já tínhamos usado, mas agora com as observações pareadas também.

Vamos começar verificando o gráfico de Bland-Altman

```
diferenca <- post-pre
medias <- apply(intake,1,mean)
plot(diferenca,medias)
```

A distribuição parece estar bem aleatória, não sendo preciso nenhum tipo de transformação desses dados para adequação. Para fazer esse teste no R, o código é bem parecido:

```
> t.test(post, pre, paired=T)

        Paired t-test

data:  post and pre
t = -11.9414, df = 10, p-value = 3.059e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1566.838 -1074.072
sample estimates:
mean of the differences
              -1320.455
```

Não creio que seja necessário explicar essa saída, não é mesmo? Não tem nada de muito diferente em relação aos outros tipos de teste *t*. Repare que os graus de liberdade nesse caso é igual a 10 e não 20, como seria no caso de um teste *t* para duas amostras.

É sempre bom frisar que uma vez tendo observações pareadas, não podemos analisar esses dados como se eles não fossem pareados. Isso porque uma das pré-suposições básicas do teste *t* para duas amostras é que elas sejam independentes, o que obviamente não acontece no caso do teste pareado. O que acontece nesta situação é um inflacionameto da variância da nossa estimativa com uma conseqüente perda de poder do teste.

Vale a pena também mostrar que esse teste é exatamente igual a fazermos a diferença entre as observações e aplicar um teste *t* para uma amostra, comparando com uma média igual a zero (se a diferença em média for zero, não haveria diferença entre os grupos). Veja como ficaria:

```
> t.test(diferenca)

        One Sample t-test

data:  diferenca
t = -11.9414, df = 10, p-value = 3.059e-07
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -1566.838 -1074.072
```

```
sample estimates:
mean of x
-1320.455
```

Teste de Wilcoxon pareado

O teste não paramétrico para observações pareadas também é igual a se aplicar o teste do sinal do posto às diferenças, o mesmo caso do teste *t* que acabamos de ver. Os argumentos são inclusive bem parecidos com o anterior. Veja:

```
> wilcox.test(post, pre, paired=T)

        Wilcoxon signed rank test with continuity correction

data:  post and pre
V = 0, p-value = 0.00384
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(post, pre,
paired = T)
```

A única possível surpresa neste caso seria o fato de o p-valor ser bem mais alto que no caso do teste paramétrico. Isso acontece porque como o teste *t* usa as médias de fato, a sua significância pode ser muito grande, dependendo de quão grande é a diferença entre as médias. Já o teste não-paramétrico é limitado, pois ele usa os *ranks* e não os valores mesmo. Por exemplo, esse p-valor que foi calculado é o menor p-valor possível para esse tamanho de amostra, pois todas as diferenças são negativas (veja que o `V = 0`).

Repare que nesse caso, como no teste *t* pareado, o teste de Wilcoxon pareado se reduz também a um teste a uma amostra das diferenças. Confira:

```
> wilcox.test(pre-post)

        Wilcoxon signed rank test with continuity correction

data:  pre - post
V = 66, p-value = 0.00384
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(pre - post)
```

Poder e tamanho de amostra para testes *t*

Na aula passada, nós vimos como calcular o poder de um teste e também o tamanho de amostras, usando o exemplo de um teste *t* para uma amostra, e através de métodos aproximados e simulações (que como dissemos muitas vezes é o único recurso que temos para alguns casos.) Comentamos então que existem sim métodos específicos para o teste *t*, mas que em geral usa-se a aproximação pela Normal.

A idéia geral do poder já foi passada, e não entraremos em detalhes aqui sobre o que significa cada uma dessas coisas, e procuraremos apenas mostras as funções disponíveis no R para esse fim. Para dizer a verdade, nessa seção falaremos apenas da função `power.t.test()`, cujo nome não deixa dúvidas sobre o que ela faz.

Como discutimos anteriormente, o poder de um teste depende basicamente de 4 fatores: o tamanho da amostra, a diferença a ser detectada, a dispersão da população e o nível de significância estabelecido (o nosso alfa). Obviamente, se nós estabelecermos 4 desses 5 fatores envolvidos, o

quinto pode ser calculado (reveja a função de poder que nós discutimos na aula passada.) É exatamente isso que esta função faz. Dê uma olhada na sua ajuda:

```
?power.t.test
```

Repare que ela tem justamente esses 5 argumentos e mais dois que estabelecem que tipo de teste *t* está sendo aplicado e se se trata de um teste uni ou bicaudal. Vamos ver então como ele funciona, usando o mesmo exemplo que usamos na última aula. Tínhamos criado uma função para calcular aproximadamente o tamanho de uma amostra, lembra?

```
tamanho.amostra <- function(alfa=0.05, poder=0.8, dif, var, bilateral=T)
{
      if (bilateral){
            zalfa<-qnorm(1-(alfa/2))
            }else{
            zalfa<-qnorm(1-alfa)
      }
      ceiling(((qnorm(poder)+zalfa)^2)*var/(dif^2))
}
```

E achamos o seguinte tamanho de amostra para esse problema:

```
> tamanho.amostra(alfa=0.05, poder=0.8, dif=10, var=625, bilateral=T)
[1] 50
```

Agora vamos conferir com a função que não usa a aproximação Normal:

```
> power.t.test(n=NULL, delta=10, sd=25, sig.level=0.05, power=0.8,
type="one.sample")

     One-sample t test power calculation

              n = 51.00957
          delta = 10
             sd = 25
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

A primeira coisa a ser notada é que tivemos que modificar um pouquinho as coisas aqui, já que os argumentos desta função são diferentes dos da função que nós criamos anteriormente. Repare que o argumento `n` ganhou o valor `NULL`, significando que eu quero achar esse valor (ele poderia simplesmente se omitido também.) O valor foi bem aproximado, não é mesmo? Tente comparar com outros valores.

É claro que você pode brincar com qualquer um dos valores, basta deixar um deles sem valor algum, que ele será calculado. Agora podemos também fazer uma outra brincadeira que é comparar os valores obtidos com essa função como os valores simulados para o poder do teste, como fizemos na aula passada. Para este exemplo a única modificação é que a diferença era de 6 e não de 10mmHg. Vamos comparar. Primeiro, vamos recordar a função:

```
poder.sim <- function(x, k=1000, H0, H1, sd)
{
      #Iniciando as variáveis
      p <- 0
      poder.t<-0
      #Loop para vários tamanhos de amostra
      for (m in 1: length(x))
      {
```

```
                  #Loop para o número de simulações
                  for (i in 1:k)
                  {
                        data <- rnorm(x[m], mean=H1, sd=sd) #Geração da amostra
sob H1
                        p[i] <- t.test(data, mu=H0)$p.value #Cálculo do p-valor
para o teste t
                  }
            poder.t[m] <- sum(p<=0.05)/k #Cálculo do Poder para cada tamanho de
amostra - o número de vezes que o p-valor é menor que 0.05, dividido pelo número
de simulações
            }
            plot(x, poder.t, ylab="Poder", xlab="n", type="b") #Plotando a curva
de poder
            cbind(x, poder.t) #Retornando os tamanhos e os poderes
      }
```

E agora vamos à simulação, com o gráfico:

```
poder.t.sim<-poder.sim(seq(25,350,25), k=1000, H0=100, H1=106, sd=25)
```

Agora, acrescentando a aproximação normal no gráfico:

```
x<-seq(25,350,25)
lines(x, pnorm(qnorm(0.975, mean=100, sd=sqrt(625/x)), mean=106,
sd=sqrt(625/x), lower.tail=F), pch=20, type="b")
```

Por enquanto, só cola do que fizemos na aula passada, certo? Estamos vendo as curvas que batem bastante bem. Agora, vamos comparar também com a nossa nova função. Vamos usar um "bacalhau" em vez de uma fórmula, usando o mesmo vetor `x`:

```
poder<-0
for (i in 1:length(x)){
poder[i] <- power.t.test(n=x[i], delta=6, sd=25, sig.level=0.05,
power=NULL, type="one.sample")$power}
```

Agora vamos acrescentar os poderes calculados ao nosso gráfico:

```
points(x, poder, pch='t', cex=2)
```

Que tal? Se quiser, compare visualmente os vetores `poder.t.sim` e `poder` para uma visualização melhor, faça um gráfico de dispersão, para ver se parecem uma linha reta:

```
plot(poder.t.sim[,2], poder)
```

Muito, bem. Esta mesma função pode então ser usada para os outros testes *t* que nós vimos, apenas mudando-se as opções necessárias. Vamos ver só mais um exemplo para o teste *t* para duas amostras.

Vamos por exemplo ver que tamanho de amostra precisaríamos para realizar o estudo que nós vimos no banco `energy` se quiséssemos detectar uma diferença de 2 Mj com um poder de 80% e uma significância de 0.05. Baseado em estudos anteriores, a variância foi estimada em torno de 1.7 Mj:

```
> power.t.test(n=NULL, delta=2, sd=1.7, sig.level=0.05, power=0.8)

     Two-sample t test power calculation

              n = 12.37967
```

```
           delta = 2
              sd = 1.7
       sig.level = 0.05
           power = 0.8
     alternative = two.sided

NOTE: n is number in *each* group
```

Precisaríamos de 13 pessoas em cada grupo neste caso (é claro que sempre vamos arredondar para cima esse número.)

Exercícios

1. Verifique se o tamanho de amostra para o banco `ashima` (na biblioteca `ISwR`) é adequado para o estudo em questão. Use também simulações para descrever as suas conclusões.

2. Mostre que o teste de variâncias independe de qual delas está no numerador ou no denominador (i.e. Mostre que o resultado será o mesmo se invertermos o nosso teste na aula). Mostre o código do programa que você usou ou os cálculos feitos à mão.

3. Mostre passo-a-passo como o R obteve o IC 95%, a estatística *T* e o p-valor para o teste *t* para duas amostras com variâncias homogêneas. Use o mesmo exemplo da aula. Mostre o código do programa que você usou ou os cálculos feitos à mão.

4. Como poderíamos calcular o p-valor para o teste de Wilcoxon com a correção de continuidade, mesmo sem sabermos se o *z* é positivo ou negativo?

5. Aplique, erradamente, um teste *t* para duas amostras independentes no banco `intake` e descreva o que acontece. Você tem alguma explicação para isso?

6. Faça as análises necessárias nos bancos (a) `intake`, (b) `vitcap` e (c) `react` da biblioteca `ISwR`, para comparar os níveis de energia e de capacidade vital e tamanho do teste tuberculínico entre os grupos, respectivamente. Use tanto o método paramétrico quanto o não-paramétrico adequado para o caso. Para informações sobre os bancos, consulte a ajuda do R. Obs: É para fazer a análise completa, o que significa descrição do banco, verificação de pré-suposições, etc.

Questão extra (0.1 ponto):

Observe o "bacalhau" para o poder que nós usamos acima e faça uma função que calcule poderes para vários tamanhos de amostra para o teste *t* para duas amostras. Entregue o código que você criou.

Aula 5 - Testes para uma e duas amostras (contínuos)
Livro: páginas 81 a 93

1. Verifique se o tamanho de amostra para o banco `ashina` é adequado para o estudo em questão. Use também simulações para descrever as suas conclusões.

A primeira providência nesse caso é conhecer melhor o banco `ashina`, o que pode ser feito através da ajuda do R. Vemos então que se trata de um estudo tipo *cross-over*, onde todos os participantes fizeram uso da substância ativa ou de um placebo, aleatoriamente, mas todos usaram ambos. Isso caracteriza de cara um estudo pareado, já que as colunas com os valores a serem testados não são independentes.

Uma observação que você deve ter feito logo de saída é que estamos lidando aqui com dados derivados de uma escala (muito embora seja uma diferença média de escalas em diferentes pontos no tempo.) Ainda assim, podemos encarar isso como um ranqueamento, o que nos remeteria a um teste não-paramétrico.

Muito bem, vamos tratar disso já. Mas agora, a pergunta que foi feita pode parecer um pouco estranha a princípio, não é mesmo? O que significa verificar se um tamanho de amostra foi adequado ou não para o que já testamos? Significa que eu gostaria de verificar se o poder desse teste seria adequado ou não para o que estou testando. Peraí! Mas o poder não depende do verdadeiro valor da hipótese alternativa, o qual não conhecemos? O poder não é simplesmente uma função?

É isso mesmo!!! Por isso não faz muito sentido falarmos em calcular apenas um único poder para o tamanho de amostra e a diferença em questão nesse estudo. Precisamos estudar essa função...

Tudo muito bonito, mas vamos então verificar o nosso banco:

```
> library(ISwR)
> data(ashina)
> attach(ashina)
> summary(ashina)
   vas.active         vas.plac            grp
 Min.   :-167.00   Min.   :-102.00   Min.   :1.000
 1st Qu.: -81.25   1st Qu.: -36.75   1st Qu.:1.000
 Median : -51.50   Median :  -5.00   Median :1.000
 Mean   : -56.81   Mean   : -13.94   Mean   :1.375
 3rd Qu.: -14.25   3rd Qu.:  11.25   3rd Qu.:2.000
 Max.   :  29.00   Max.   :  32.00   Max.   :2.000
```

O que nos interessa aqui são as variáveis `vas.active` e `vas.plac`. Aliás, para ser mais exato o que realmente nos interessa é a diferença entre essas variáveis, certo? Dado o desenho deste estudo, estamos lidando com um grupo pareado, logo podemos testar a diferença desses valores com um teste *t* para uma amostra, por exemplo. Vamos ver o que acontece com esse teste, nesse caso:

```
> t.test(vas.active-vas.plac)

        One Sample t-test

data:  vas.active - vas.plac
t = -3.2269, df = 15, p-value = 0.005644
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -71.1946 -14.5554
sample estimates:
mean of x
```

```
  -42.875
```

Tudo bem, mas se olharmos o gráfico de Bland-Altman, veremos uma certa tendência das médias serem maiores quanto maiores forem as diferenças. Além disso, a amostra é pequena e estamos falando de um escore. É... vamos usar um teste não-paramétrico:

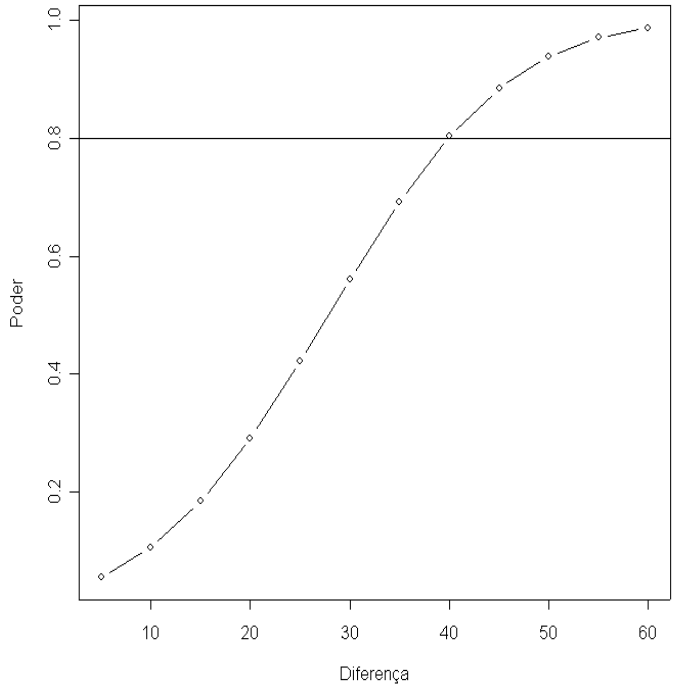
```
> wilcox.test(vas.active-vas.plac)

        Wilcoxon signed rank test

data:  vas.active - vas.plac
V = 20, p-value = 0.01099
alternative hypothesis: true mu is not equal to 0
```

Bem, até que não mudou o nosso resultado, certo? Tá, mas e cálculo do poder afinal de contas? Pois é... precisamos abordar isso de alguma maneira... Que tal tentarmos usar aquela função `power.t.test()` que nós tínhamos usado anteriormente, usando o *n* fixo, uma diferença variável e um desvio-padrão qualquer? Epa! O *n* tudo bem, mas e os outros dois, como fazer? É fácil: vamos ver qual foi a média calculada da amostra, e fazer deltas em torno dessa média. Para o DP, vamos usar o melhor que temos que é o DP da amostra mesmo.

Vamos começar fazendo uma brincadeira, mas admitindo que estivéssemos falando de um teste *t* então, usando uma pequena modificação do código que vimos na aula, poderíamos fazer:

```
x<-seq(5,60,5)
poder<-0
for (i in 1:length(x)){
poder[i] <- power.t.test(n=16, delta=x[i], sd=53, sig.level=0.05,
power=NULL, type="one.sample")$power}
plot(x,poder, type="b", xlab="Diferença", ylab="Poder")
abline(h=0.8)
```

Repare que a única diferença é que agora o vetor refere-se a várias diferenças, mantendo o *n* fixo e as diferenças indo de 5 a 60, o que inclui a diferença de 42.8 (em módulo) calculada lá no nosso teste *t*. Para o SD, usamos o SD da diferença da amostra. Veja o resultado:



Repare que para uma diferença em torno de 40, o poder desse teste fica próximo de 80%.

Mas por que eu falei em usar simulações se nós temos essa função exata para o cálculo do poder? Isso mesmo: não aprendemos a fazer isso para um teste não-paramétrico. Bem, nós tínhamos a nossa função `poder.sim()` que poderia nos ajudar bastante com isso. O problema é que ela foi criada para variar os tamanhos de amostra apenas, para uma hipótese alternativa fixa. No nosso caso, seria mais interessante variar bastante a hipótese alternativa e testar alguns tamanhos de amostra próximos ao que foi utilizado. Teríamos que fazer uma "pequena" modificação. Que tal essa:

```
poder.sim1 <- function(x, k=1000, H0, H1, sd)
{
     #Iniciando as variáveis
     p <- 0
     poder.w<-matrix(nrow=length(x), ncol=length(H1))
     #Loop para vários tamanhos de amostra
     for (m in 1: length(x))
     {
               #Loop para o número de diferenças
               for (n in 1:length(H1))
               {
                         #Loop para o número de simulações
                         for (i in 1:k)
                         {
                                   data <- rnorm(x[m], mean=H1[n], sd=sd)
#Geração da amostra sob cada H1
                                   p[i] <- wilcox.test(data,
mu=H0)$p.value #Cálculo do p-valor
                         }
               poder.w[m,n] <- sum(p<=0.05)/k #Cálculo do Poder para cada
tamanho de amostra e delta
               }
     }
     dimnames(poder.w) <- list(n=x, delta= H1)
     t(poder.w) #Retornando os tamanhos e os poderes
```

```
}
```

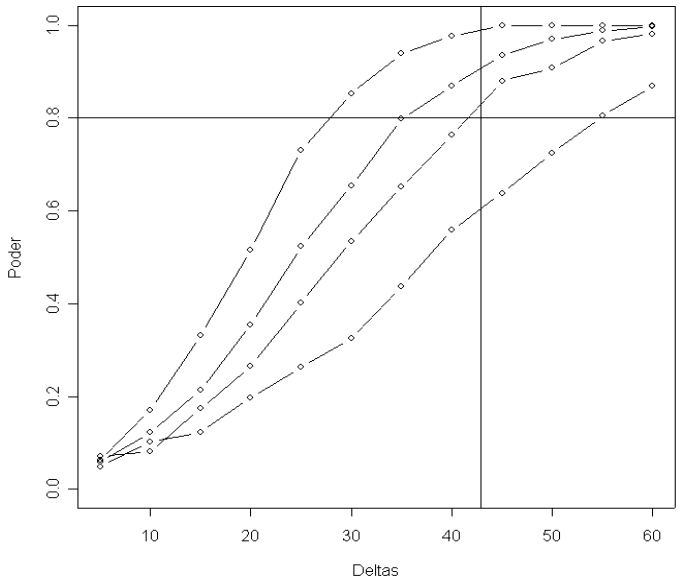
Repare que algumas coisas foram assumidas para isso: primeiro, a diferença entre os escores foi considerada normal. Essa não é uma pressuposição muito longe da realidade, não é mesmo? Lembre-se que ela é usada quando calculamos os p-valores dos testes não-paramétricos. A segunda, nós já falamos: é sobre o DP da distribuição, que usaremos o estimado a partir da amostra.

A partir daí poderíamos fazer um gráfico para ver como o poder se comporta para vários deltas e alguns tamanhos de amostra. Primeiro vamos criar um objeto com a saída da nossa função, para tamanhos de amostra 10, 16 20 e 32, e com os mesmos deltas que usamos lá em cima:

```
      poderes<-poder.sim1(c(10,16,20,32), k=10, H0=0, H1=seq(5,60,5),
sd=sd(vas.active-vas.plac))
```

Agora vamos ao gráfico:

```
      plot(dimnames(poderes)$delta, poderes[,1], type="b", ylab="Poder",
xlab="Deltas", ylim=c(0,1))
      lines(dimnames(poderes)$delta,poderes[,2], type="b")
      lines(dimnames(poderes)$delta,poderes[,3], type="b")
      lines(dimnames(poderes)$delta,poderes[,4], type="b")
      abline(v=abs(mean(vas.active-vas.plac)))
      abline(h=0.8)
```



Como você pode perceber pelo gráfico, o poder para uma hipótese alternativa igual à média da amostra, o poder desse teste seria acima de 80%. A questão a ser discutida é se essa diferença de mais de 40 pontos seria de fato o mínimo que se queria encontrar. Repare que um tamanho de 32 daria um poder de 80% para uma diferença de cerca de 25 pontos.

Compare agora o resultado dessas curvas de poder usando o teste *t* em vez do teste de Wilcoxon. Quais são as suas conclusões em relação ao que deu de diferente entre eles? Quem entregar até a próxima aula ganha 0.1 ponto.

2. Mostre que o teste de variâncias independe de qual delas está no numerador ou no denominador (i.e. Mostre que o resultado será o mesmo se invertermos o nosso teste na aula). Mostre o código do programa que você usou ou os cálculos feitos à mão.

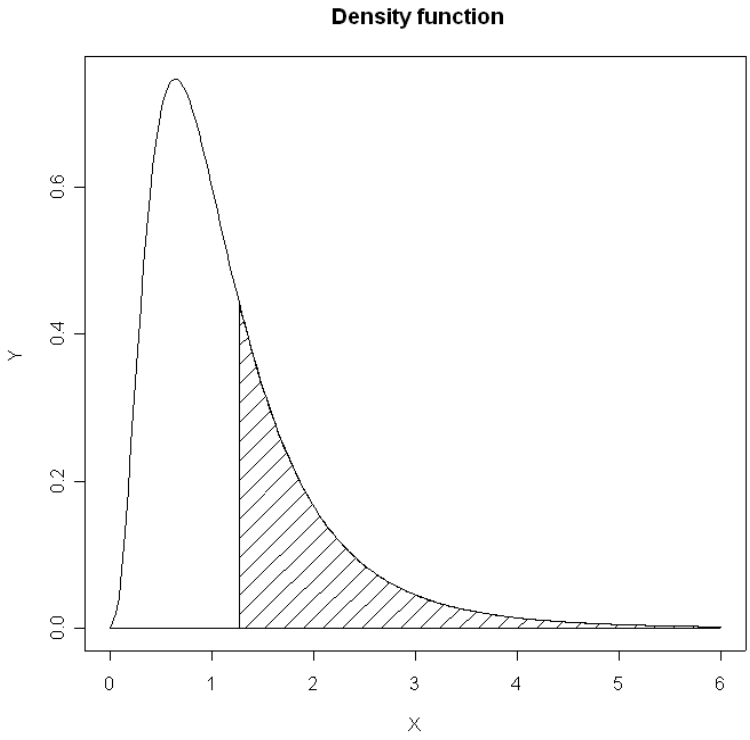
Essa é fácil, né? Temos que inverter as variâncias e os graus de liberdade na nossa fórmula. Ah, e também a cauda! Repare que agora a razão é maior que 1, logo estamos procurando uma área na causa superior, certo? Vamos ver:

```
> s1/s2
[1] 1.274785
```

Isso vai me obrigar a usar aquela opção:

```
> 2*pf(s1/s2, df1=n1-1, df2=n2-1, lower.tail=F)
[1] 0.679746
```

Confere? Veja a área que estamos procurando (e depois multiplicando por 2):



3. Mostre passo-a-passo como o R obteve o IC 95%, a estatística *T* e o p-valor para o teste *t* para duas amostras com variâncias homogêneas. Use o mesmo exemplo da aula. Mostre o código do programa que você usou ou os cálculos feitos à mão.

Bom, essa você deve estar já cansado de fazer, não é mesmo? Além disso, é praticamente copiar e colar o código da aula, mudando apenas os graus de liberdade e o erro padrão, certo? Vamos ver o que não mudaria:

```
s1 <- var(expend[stature=="obese"])
s2 <- var(expend[stature=="lean"])
n1 <- length(expend[stature=="obese"])
n2 <- length(expend[stature=="lean"])
media1 <- mean(expend[stature=="obese"])
media2 <- mean(expend[stature=="lean"])
```

E os graus de liberdade e a variância da média:

```
gl <- n1+n2-2
```

```
var.media<-(s12/n1)+(s12/n2)
```

Agora é só calcular o IC (de novo):

```
> (media1-media2)+(c(-1,1)*qt(0.975, df=gl)*sqrt(var.media))
[1] 1.051796 3.411451
```

Confira com a saída do R. Ainda o problema do sinal? Ihhhhhhhhh...

4. Como poderíamos calcular o p-valor para o teste de Wilcoxon com a correção de continuidade, mesmo sem sabermos se o *z* é positivo ou negativo?

Essa é só uma questão algébrica mesmo. Dada a simetria da Normal, podemos trabalhar sempre com o quantil positivo. Para isso, basta aplicarmos o módulo à diferença, diminuir sempre 0.5 e calcular sempre a cauda superior da fdp:

```
2*pnorm((abs(12-media)-0.5)/sqrt(variancia), lower.tail=F)
[1] 0.002121613
```

Confere?

5. Aplique, erradamente, um teste *t* para duas amostras independentes no banco `intake` e descreva o que acontece. Você tem alguma explicação para isso?

Bem, inicialmente, vamos acrregar o banco:

```
data(intake)
attach(intake)
```

E agora, basta aplicar o teste sem a opção `paired=T`:

```
> t.test(post, pre)

        Welch Two Sample t-test

data:  post and pre
t = -2.6242, df = 19.92, p-value = 0.01629
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2370.3458  -270.5633
sample estimates:
mean of x mean of y
 5433.182  6753.636
```

O que se observa é que o p-valor é bem menor que no teste parado e o IC é muito mais alargado. Isso demonstra a perda de eficiência quando analisamos dados que têm estrutura de dependência, como se fossem independentes (há um inflacionamento da variância).

6. Faça as análises necessárias nos bancos (a) `intake`, (b) `vitcap` e (c) `react` da biblioteca `ISwR`, para comparar os níveis de energia e de capacidade vital e tamanho do teste tuberculínico entre os grupos, respectivamente. Use tanto o método paramétrico quanto o não-paramétrico adequado para o caso. Para informações sobre os bancos, consulte a ajuda do R. Obs: É para fazer a análise completa, o que significa descrição do banco, verificação de pré-suposições, etc.

(a)

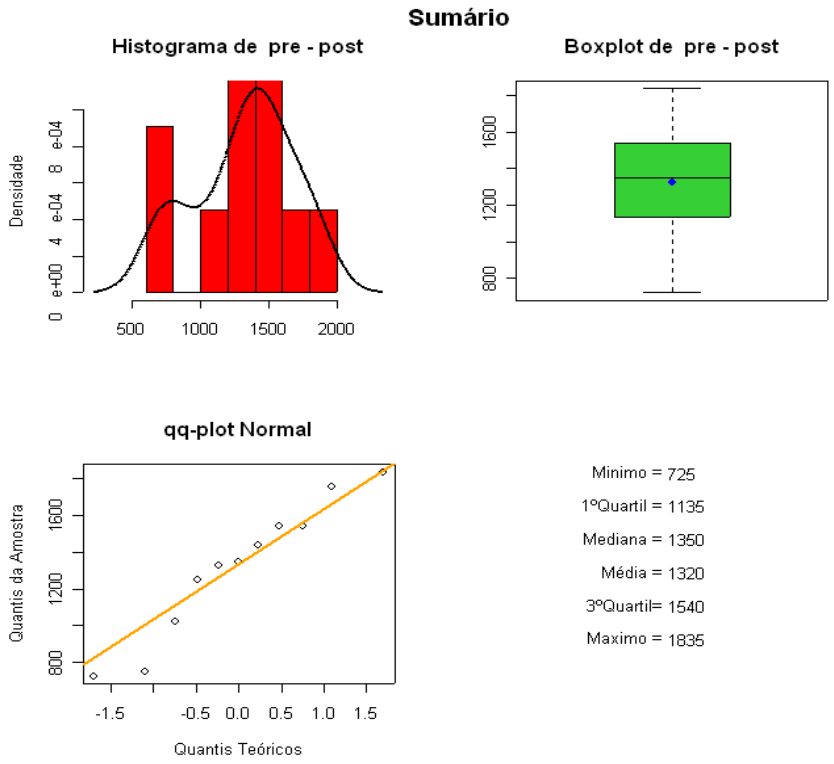
Bem, esse banco nós efetivamente usamos na aula, então facilita bastante. Trata-se de um

banco com aferições de ingesta energética de mulheres nas fases pré e pós menstruais em kJ.

Podemos descrever o banco da maneira usual, usando a função `summary()` para o banco todo:

```
> summary(intake)
      pre              post
 Min.   :5260   Min.   :3885
 1st Qu.:5910   1st Qu.:4450
 Median :6515   Median :5265
 Mean   :6754   Mean   :5433
 3rd Qu.:7515   3rd Qu.:6383
 Max.   :8770   Max.   :7335
```

Ou ainda, mais interessante, usar a função `eda()` na diferença:



Apesar do histograma não aparentar uma distribuição muito normal, o Boxplot e o Q-Q plot parecem indicar alguma normalidade. Conferindo:

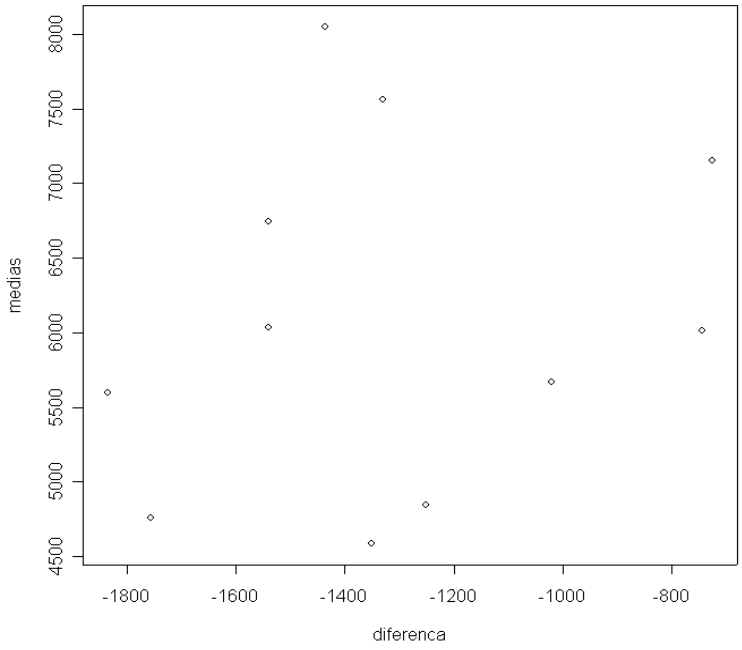
```
> shapiro.test(pre-post)

        Shapiro-Wilk normality test

data:  pre - post
W = 0.9374, p-value = 0.4901
```

Enfim, é um estudo pareado, como vimos e vamos testar se a ingesta difere nessas duas fases:

H_0: $\mu_{pre} - \mu_{pos} = 0$

H_1: $\mu_{pre} - \mu_{pos} \neq 0$

Além da normalidade, o gráfico de Bland-Altman parece não mostrar tendência alguma:



Mesmo assim, vamos usar o teste não-paramétrico também:

```
> t.test(post-pre)

        One Sample t-test

data:  post - pre
t = -11.9414, df = 10, p-value = 3.059e-07
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -1566.838 -1074.072
sample estimates:
mean of x
-1320.455

> wilcox.test(post-pre)

        Wilcoxon signed rank test with continuity correction

data:  post - pre
V = 0, p-value = 0.00384
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(post - pre)
```

Apesar da menor eficiência do teste não-paramétrico (esperado), ambos os testes rejeitam a hipótese nula de que a ingesta é igual em ambas as fases. Logo, a conclusão é que a ingesta pré-menstrual é menor que a pós, e esse resultado é estatisticamente significativo para um alfa de 5%.

(b)

Esse banco contém informações sobre a capacidade vital de trabalhadores da indústria de

cádmio. As variáveis descrevem, além da capacidade vital, a idade do trabalhador e ainda o tipo de exposição (nesse caso somente não exposto e expostos há mais de 10 anos). Evidentemente o que queremos estudar nesse caso é a associação do tempo de exposição ao cádmio com a capacidade vital. A idade é importante porque pode ser uma variável de confundimento nesse caso.

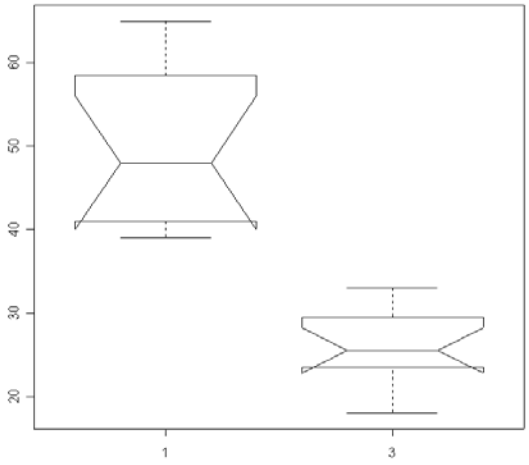
A descrição pode ser feita com a função também, sem problemas para a capacidade vital e para a idade, e pode-se inclusive separar por grupo. Não vou nem mostrar aqui, que isso já é básico... Os comandos:

```
eda(vital.capacity[group==1])
eda(vital.capacity[group==3])
```

Outra maneira é usando aquela função que nós vimos na aula 2, lembra?

```
> resumo.grupo(vital.capacity,group)
  Mínimo Máximo Mediana    Média        DP
1   2.70   5.52   3.865 3.949167 1.0330578
3   3.67   5.86   4.985 4.992500 0.6796272
> resumo.grupo(age,group)
  Mínimo Máximo Mediana    Média       DP
1     39     65    48.0 49.75000 9.106691
3     18     33    25.5 25.91667 4.776045
```

Bem, dá para perceber que a idade tem uma relação importante com a exposição, não é mesmo? Você saberia dizer porque? Vamos ver um boxplot com indentações:



As idades parecem ser bem diferentes, não?

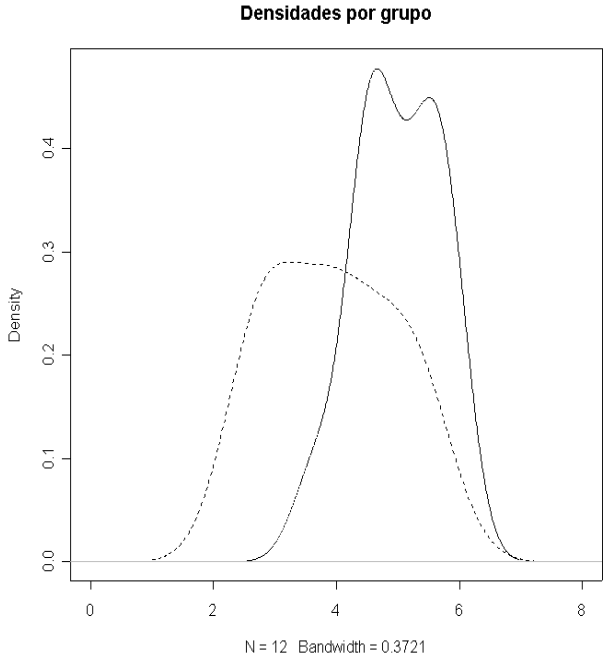
Muito bem, mas agora, com o que aprendemos, só podemos testar mesmo se a exposição afeta ou não a capacidade vital desses trabalhadores, não é mesmo? Vamos ver se podemos assumir normalidade (repare que o *n* é pequeno nesse caso:

```
> shapiro.test(vital.capacity[group==1])

        Shapiro-Wilk normality test

data:  vital.capacity[group == 1]
W = 0.9163, p-value = 0.257

> shapiro.test(vital.capacity[group==3])
```

```
        Shapiro-Wilk normality test

data:  vital.capacity[group == 3]
W = 0.9342, p-value = 0.4269
```

Hummm. Que tal uma visualização gráfica?

```
> plot(density(vital.capacity[group==3]), xlim=c(0,8), main="Densidades
por grupo")
> lines(density(vital.capacity[group==1]), lty=2)
```



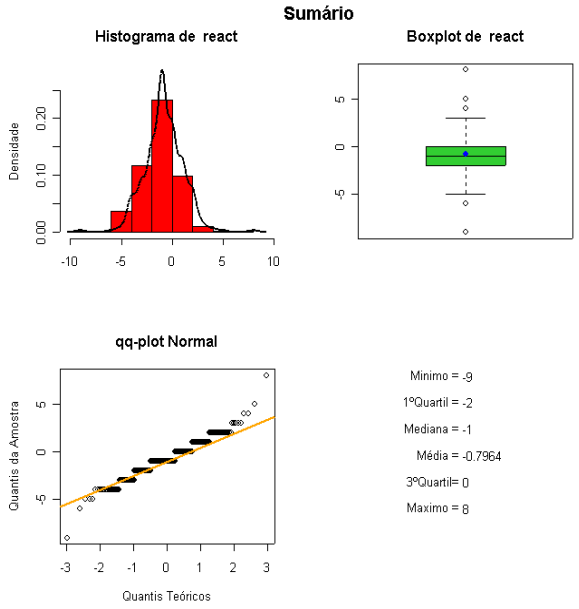
É melhor fazer os dois, não é?

```
> t.test(vital.capacity~group)

        Welch Two Sample t-test

data:  vital.capacity by group
t = -2.9228, df = 19.019, p-value = 0.008724
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.7904211 -0.2962456
sample estimates:
mean in group 1 mean in group 3
       3.949167        4.992500

> wilcox.test(vital.capacity~group)

        Wilcoxon rank sum test with continuity correction

data:  vital.capacity by group
W = 30.5, p-value = 0.01783
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test.default(x = c(4.62,
5.29, 5.52, 3.71, 4.02, 5.09,
```

A conclusão é que existe uma associação estatisticamente significativa entre o tempo de exposição ao cádmio e a capacidade vital desses trabalhadores (que é menor nos expostos). Você ficaria satisfeito com esse resultado? Por que? Responda e proponha uma solução (não é para fazer, só propor) até a próxima aula e ganhe 0.1 ponto.

(c)

Esse é o mais simples de todos os bancos e contém apenas as diferenças entre 2 testes de PPD na mesma enfermeira em tempos diferentes. Nesse caso ele é apenas um vetor, e podemos apenas descrevê-lo de maneira básica.



Apesar do estranho Q-Q plot (por que?) parece que podemos aproximar bem pela normal, não é mesmo? Mesmo assim, vamos ver as duas maneiras de se fazer:

```
> t.test(react)

        One Sample t-test

data:  react
t = -7.7512, df = 333, p-value = 1.115e-13
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.9985214 -0.5942930
sample estimates:
 mean of x
-0.7964072

> wilcox.test(react)

        Wilcoxon signed rank test with continuity correction

data:  react
V = 9283.5, p-value = 2.075e-13
alternative hypothesis: true mu is not equal to 0
```

Deixo as conclusões para você....

Questão extra (0.1 ponto):

Observe o “bacalhau” para o poder que nós usamos acima e faça uma função que calcule poderes para vários tamanhos de amostra para o teste *t* para duas amostras. Entregue o código que você criou.

```
      poderes.t <- function(n, delta, sd, sig.level=0.05, power=NULL,
type="two.sample")
      {
           poder<-0
           for (i in 1:length(n))
           {
                     poder[i] <- power.t.test(n=n[i], delta=delta, sd=sd,
sig.level=sig.level, power=NULL, type=type)$power
           }
           poder
      }
```

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 6 - Proporções
Livro: páginas 129 a 138

Nesta aula vamos ver como podemos analisar dados categóricos. O R possui diversas funções para tratar desse tipo de dados, mas só abordaremos as comparações mais simples, deixando as mais complexas para um curso mais avançado em estatística.

Proporções Simples
Duas proporções independentes
Teste de McNemar
k proporções independentes
Teste de tendência (linear)
Tabelas *r* x *c*
Poder para proporções
Exercícios

Proporções Simples

Os testes para comparar proporções simples (ou seja, uma proporção conhecida contra a proporção de uma amostra) são feitos a partir da distribuição Binomial. Pode-se usar tanto uma aproximação pela distribuição Normal, para um tamanho de amostra suficientemente grande, ou então usar um teste exato, baseado na própria Binomial.

Existem várias regrinhas para se saber se a amostra é suficientemente grande ou não e você já deve ter ouvido várias delas. Para citar duas, uma é se o valor esperado de sucessos e de fracassos na sua amostra é maior que 5; a outra é se o produto de n por $p_0(1 - p_0)$ for maior ou igual a 5. Repare que no caso de uma amostra, estamos falando do p e do q teóricos e do n da amostra. Vamos ver isso brevemente.

Vamos apenas refrescar a memória sobre a nossa aproximação. A Normal em questão terá média np_0 e variância $np_0(1 - p_0)$, para o número absoluto de sucessos dessa Binomial, ou, se quisermos falar de proporções propriamente ditas, dividimos ambas as medidas por n e então teremos que:

$z = \frac{\hat{p} - p_0}{\sqrt{p_0(1 - p_0)/n}}$ será distribuída como uma Normal-padrão, onde $\hat{p}$ é a proporção da amostra e p_0 é a proporção contra a qual queremos testar a nossa amostra.

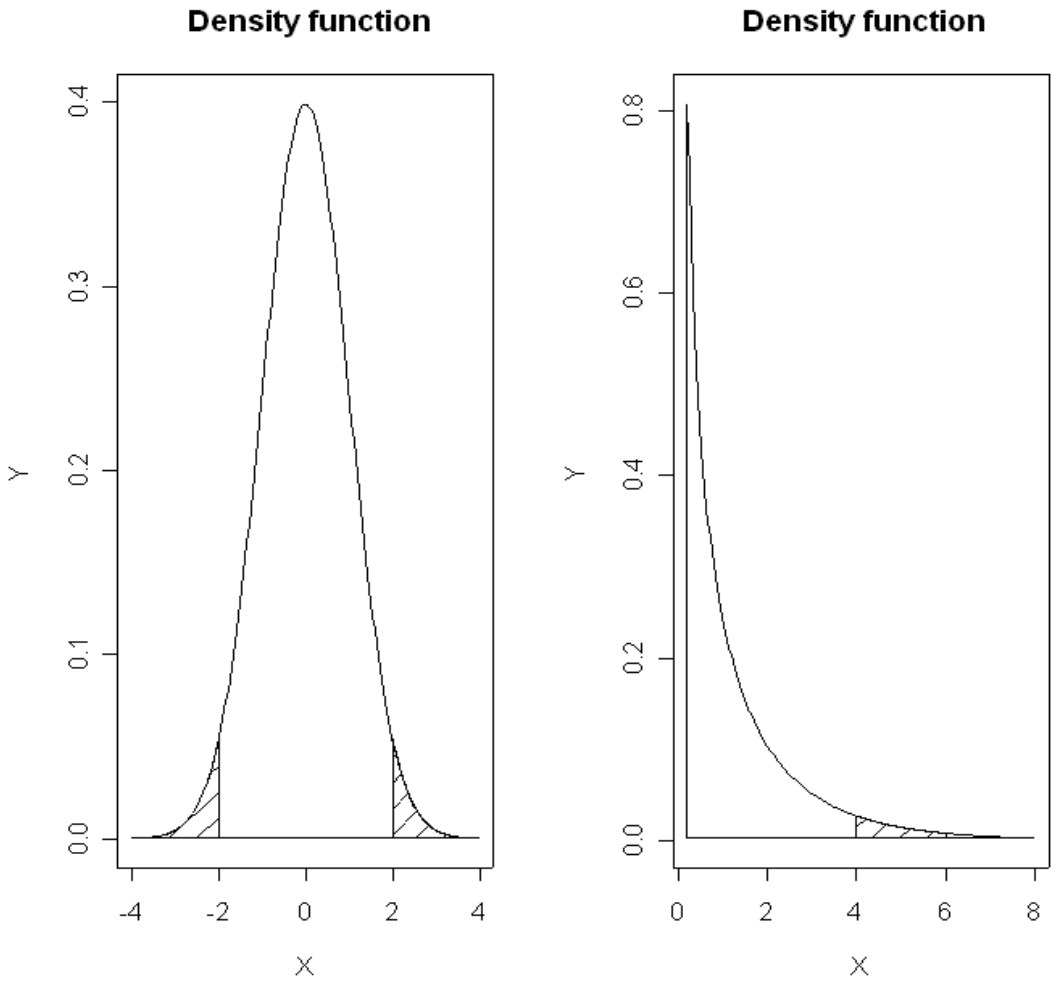
Cuidado: o denominador foi dividido por n^2. Por que?

Uma informação nova que pode e deve ser introduzida neste momento é a relação que existe entre a Normal e a distribuição Qui-quadrada (nós já vimos a relação desta com a distribuição F, lembra?) Pois é, a distribuição Qui-quadrada é uma soma de distribuições Normais (0,1) ao quadrado, sendo que o seu parâmetro que são os graus de liberdade correspondem ao número de

Normais sendo somadas. No caso de apenas elevarmos uma única Normal-padrão ao quadrado (sem somar outra), estaremos diante de uma χ^2_1 .

Você pode estar achando estranho eu falar em elevar a distribuição Normal-padrão ao quadrado, mas é isso mesmo. Isso se chama transformar uma distribuição em outra distribuição. Podemos inclusive testar esse fato no R. Vamos fazer o seguinte: se o que estou dizendo é verdade, então a densidade acumulada para um quantil qualquer da χ^2_1 tem que ser igual à densidade acumulada desse mesmo quantil ao quadrado em uma Normal (0,1), certo? Bem, não exatamente... Há uma complicação ao elevarmos uma Normal ao quadrado, especialmente no que diz respeito ao seu *range*, que obviamente passará a não admitir números negativos.

Sem entrar em muitos detalhes, sempre que esta transformação for feita, a relação que vamos facilmente encontrar é entre os p-valores de um teste bilateral para a Normal. O que isso significa é que se nós pegarmos as áreas das caudas de uma Normal determinadas por um quantil e seu simétrico, essa área corresponderá à área determinada por este quantil ao quadrado em uma Qui-quadrada com 1 grau de liberdade, ou seja, as áreas no gráfico abaixo são equivalentes:



Não se engane por conta da escala, onde parece que a área da Normal (à esquerda) é maior que a da Qui-quadrada (à direita). Vamos conferir isso numericamente. Os quantis em questão são o -2 e o 2 para a Normal e o 4 para a χ^2_1 :

```
> 2*pnorm(-2)
[1] 0.04550026
```

```
> 1-pchisq(4,df=1)
[1] 0.04550026
```

Entendeu a relação entre as duas? Simples, não? Essa é uma relação importante de se entender, pois muitas vezes vocês vão se deparar com este tipo de transformação.

Bom, mas isso foi apenas um parênteses, pois estávamos falando do teste para uma proporção simples, aproximado por uma Normal (0,1). Vamos usar o mesmo exemplo da livro, onde foi feito um estudo com 215 pessoas e feito o diagnóstico de asma em 39 delas. Vamos testar se para esta amostra a proporção de pessoas com asma seria significativamente diferente de 0.15. No R:

```
> prop.test(39,215,.15)

        1-sample proportions test with continuity correction

data:  39 out of 215, null probability 0.15
X-squared = 1.425, df = 1, p-value = 0.2326
alternative hypothesis: true p is not equal to 0.15
95 percent confidence interval:
 0.1335937 0.2408799
sample estimates:
        p
0.1813953
```

Esta função tem como argumentos o número de sucessos (atenção, não é a proporção!!!), o número de experimentos e, neste caso, a proporção contra a qual queremos testar a amostra em questão. Ainda usaremos esta função de novo, mas se você preferir, dê uma olhada na ajuda...

Vamos ver a nossa saída devagar:

```
        1-sample proportions test with continuity correction
```

Indica que trata-se de um teste de proporção para uma amostra e que foi usada uma correção de continuidade. Nós já falamos nela na aula passada, mas vamos dissecá-la um pouco mais daqui a pouco, aguarde.

```
data:  39 out of 215, null probability 0.15
```

Indica o que foi fornecido quando chamamos a função: o número de sucessos em tantos experimentos e a probabilidade a ser testada.

```
X-squared = 1.425, df = 1, p-value = 0.2326
```

Repare que a estatística fornecida pelo R é na verdade a Qui-quadrada e não a *z*. De cara você percebeu que este teste foi feito bilateralmente, não é mesmo? Vamos conferir rapidamente no R como seria com a Normal? Veja:

```
> 1-pchisq(1.425,1)
[1] 0.2325822
> 2*(1-pnorm(sqrt(1.425)))
[1] 0.2325822
```

Mas você deve estar curioso para saber como esse valor de 1.425 foi achado, não é mesmo? É fácil: basta aplicar a transformação lá de cima para uma Normal e então elevar este número ao quadrado. Podemos usar tanto o número de sucessos quanto as proporções, certo (este último é obtido simplesmente dividindo os sucessos por *n*). Vamos usar o número de sucessos para ficar igual ao R:

```
> z<-(abs((215*0.15)-39)-0.5)/(sqrt(215*0.15*(1-0.15)))
> z^2
[1] 1.424989
```

A única diferença em relação à fórmula acima é esse que foi colocado no numerador e que como já vimos corresponde à correção de continuidade... Não se afobe, a explicação já vem... Continuando:

```
alternative hypothesis: true p is not equal to 0.15
```

Nada de novo....

```
95 percent confidence interval:
 0.1335937 0.2408799
```

O IC 95% para a proporção da amostra, construído também a partir da Normal. Você já deve estar cansado de calcular isso, mas deixo para você conferir se for do seu interesse.

```
sample estimates:
        p
0.1813953
```

E a estimativa da proporção amostral, que obviamente é:

```
> 39/215
[1] 0.1813953
```

Bem, mas como você já devia desconfiar, podemos muito bem lançar mão de um teste exato para esta proporção. Claro, podemos usar a distribuição Binomial propriamente dita, sem usar aproximação alguma:

```
> binom.test(39,215,.15)

        Exact binomial test

data:  39 and 215
number of successes = 39, number of trials = 215, p-value = 0.2135
alternative hypothesis: true probability of success is not equal to 0.15
95 percent confidence interval:
 0.1322842 0.2395223
sample estimates:
probability of success
             0.1813953
```

Bem, como você pode notar, a chamada é exatamente igual ao teste anterior e a saída é bem parecida também. Há no entanto diferenças nos valores do p-valor e do IC 95%. Vamos ver como é que eles foram obtidos. Bem, é claro que tudo isso é baseado na nossa velha Binomial, que já estamos carecas de conhecer. Vamos ver o jeitão dessa aí, sob a hipótese nula:

```
x<-0:75
plot(x, dbinom(x, size=215, prob=0.15), type="h")
```

Bom, mas isso não me ajuda muito, pois para calcular o p-valor precisamos, como na Normal, calcular a área sob essa curva, que venha de 0 até o valor a ser testado vezes dois (se o

valor a ser testado for < 39) ou desse valor até 215 vezes dois também, se o valor for > 39. Mas afinal, que valor é esse? Fácil, como queremos testar o valor 0.15:

```
> 0.15*215
[1] 32.25
```

Epa! Mas esse valor não é inteiro! Como é que vamos testar isso? Pois é, começam os problemas... Bem, mas pelo menos sabemos que caímos no primeiro caso, ou seja o valor procurado é menor que a média... Só que isso não ajuda também muito. Repare o que está acontecendo: Lembra que para calcular o p-valor nós construímos a distribuição **sob a hipótese nula** e depois calculamos a área? Pois é, acontece que a nossa hipótese nula é uma Binomial com 32.25 sucessos... Isso não existe!!! Que enrascada, hein??? Calma. Essa é a média da Binomial, e pode ser um número não inteiro, sim!

Para sair dessa, existem várias maneiras propostas por diferentes autores. Uma delas, menos complicada, defende o cálculo de um p-valor unilateral, multiplicando-se por dois o valor encontrado. Para encontrar esse valor, teremos que fazer assim: se o número de sucessos observados for maior que o número de sucessos esperados (nosso caso aqui – 39 e 32.25, respectivamente), calculamos 2 vezes o valor da probabilidade do número de sucessos ser **maior ou igual** ao valor observado numa Binomial sob a hipótese nula. Caso contrário será a mesma coisa, mas para a probabilidade de ser **menor ou igual** ao valor esperado.

Repare bem que no nosso caso a probabilidade de ser igual a 39 deve estar incluída, e o R não vai ser nosso amigo neste momento. Lembre-se que a função `pbinom()` vai nos dar $P(X \leq k)$ e portanto o seu complemento nos fornecerá $P(X > k)$. Mas o que queremos é $P(X \geq k)$ e portanto temos que tomar cuidado aqui. Uma maneira de fazer isso com facilidade é calcular a soma das densidades de todos os valores possíveis, que no nosso caso seriam de 39 a 215 sucessos:

```
> sum(dbinom(39:215, size=215, prob=0.15))
[1] 0.1178394
```

Esta então é a soma de todos os valores maiores ou iguais a 39 sucessos em uma Binomial (0.15, 215). Agora, basta multiplicar por 2:

```
> 2*sum(dbinom(39:215, size=215, prob=0.15))
[1] 0.2356789
```

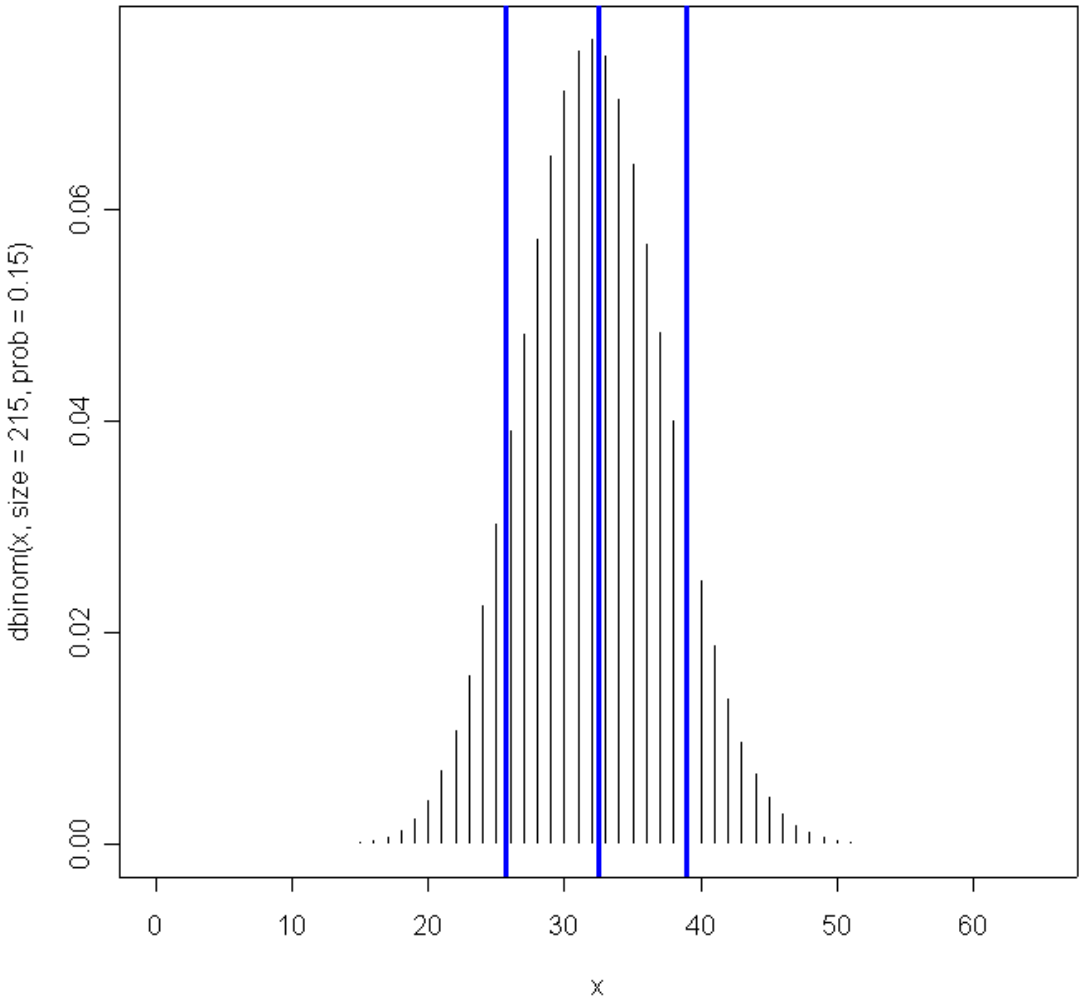
E teremos então o nosso p-valor. Acontece que o R usa um método algo mais complicado para esse cálculo. Em vez de simplesmente multiplicar por 2, ele calcula a área para um valor teórico, simétrico à média da Binomial sob a hipótese nula. Ficou difícil, né? Nós já fizemos algo parecido no exercício da primeira aula, com a Normal. É assim: calculamos a diferença entre o número observado e o esperado e aí adicionamos essa diferença para o outro lado da reta, para obter um número simétrico em torno da média teórica. No nosso caso:

```
> 39-32.25
[1] 6.75
```

Agora, basta diminuir esse valor de 32.5:

```
> 32.25-(39-32.25)
[1] 25.5
```

Vamos olhar no gráfico o que foi feito para ficar mais claro:



A linha azul central é a média dessa Binomial, a linha da direita é o valor 39 (sua densidade está inclusive encoberta pela linha azul) e a linha da esquerda é o seu simétrico em relação à média. Bem o que o R faz é calcular a soma das densidades à esquerda da linha azul da esquerda e à direita da linha azul da direita. Assim:

```
> pbinom(25.5, size=215, prob=0.15)+(sum(dbinom(39:215, size=215,
prob=0.15)))
[1] 0.2135205
```

Que na verdade é o valor que já tínhamos calculado somado à $P(X \leq 25.5)$.

Tudo muito bonito, mas e o IC 95%, que é calculado na verdade a partir da amostra, como é que se calcula isso? Bem, baseado no que acabamos de ver, já deu para perceber que será meio difícil calcular um IC 95% que seja exatamente 95%, concorda? Ou ele será um pouquinho mais, ou um pouquinho menos...

Acontece que o cálculo desses intervalos, mesmo os exatos ainda são um assunto de intensa discussão em estatística e existem várias formas de se calcular esses intervalos. Vamos ver aqui um exemplo.

Se nós fôssemos calcular um IC 95% conservador para esse valor, nós poderíamos simplesmente estabelecer que valores desta Binomial corresponderiam a uma área de 0.025 e 0.975 à esquerda da nossa curva:

```
> qbinom(0.025, size=215, prob=39/215)
[1] 28
> qbinom(0.975, size=215, prob=39/215)
[1] 50
```

Agora, poderíamos conferir se esse intervalo de fato comporta 95% da massa de probabilidade:

```
> sum(dbinom(28:50, size=215, prob=39/215))
[1] 0.9587535
```

Muito bem, um pouquinho mais que 95%, logo o intervalo (28/215,50/215) – aberto mesmo – seria um IC 95% exato, conservativo para essa amostra. Isso dá:

```
> 28/215
[1] 0.1302326
> 50/215
[1] 0.2325581
```

Repare que esses valores são diferentes dos obtidos pelo R:

```
95 percent confidence interval:
 0.1322842 0.2395223
```

Os intervalos calculados pelo R são os chamados intervalos de Clopper-Pearson, e que são considerados bastante conservadores, ou seja eles têm no mínimo 1-alfa de massa de probabilidade. A explicação da teoria por trás do cálculo desse intervalo foge totalmente ao escopo dessa aula, mas como o cálculo em si é bem fácil no R, vou explicar rapidamente.

É o seguinte: para calcular um IC 95% para uma Binomial com *k* sucessos observados, basta usarmos uma distribuição Beta (aliás duas, uma para o limite inferior e outra para o superior.) É assim: $L(k) \sim Beta(k, n-k+1)$ e $U(k) \sim Beta(k+1, n-k)$, onde *L(k)* é o limite inferior (de *lower*) e *U(k)* é o limite superior (de *upper*) do intervalo. Nesse caso, basta calcular os quantis para 0.025 na primeira distribuição e o quantil para 0.975 na segunda, para termos os nossos limites. Como temos *k* (39) e também *n* (215), ficou fácil. No R:

```
> qbeta(0.025, 39, 215-39+1)
[1] 0.1322842
> qbeta(0.975, 39+1, 215-39)
[1] 0.2395223
```

Bem, agora vamos voltar para a nossa correção de continuidade. Como havíamos visto, ela é necessária para que distribuições discretas sejam melhor aproximadas por distribuições contínuas. Acontece sempre quando aproximamos uma Binomial por uma Normal ou por uma Qui-quadrada. Tínhamos visto também na aproximação das distribuições das estatísticas de ordem, nos testes não-paramétricos. A idéia é simples: como são distribuições discretas, em vez de calcularmos a probabilidade contida entre dois pontos (no caso de um intervalo de confiança.)

Nós já tivemos muitos problemas para entender o IC 95% aí em cima, então vamos trabalhar com a distribuição acumulada de uma binomial, para entendermos melhor o que acontece. Primeiro, vamos plotar a função acumulada da Binomial que a gente acabou de trabalhar:

```
x<-0:100
```

```
acumulada.binom<-pbinom(x, size=215, prob=0.15)
plot(x, acumulada.binom, type="s")
```

É igual ao gráfico anterior, ,lembrando que fizemos só até 100 para poupar espaço. Agora vamos calcular a aproximação Normal sem correção e ver como fica a sua distribuição acumulada no gráfico:

```
acumulada.norm<-pnorm(x, mean=215*0.15, sd=sqrt(215*0.15*(1-0.15)))
points(x, acumulada.norm)
```

Repare que vários dos pontos estão claramente *entre* os degraus da escada, ou seja eles estão correspondendo a números diferentes no eixo *y*. Lembre-se que o valor da função acumulada de uma distribuição discreta é na verdade o degrau mesmo. Veja qual seria o valor para 30 sucessos, no gráfico:

```
abline(v=30)
arrows(20, pbinom(30, size=215, prob=0.15), 30, pbinom(30, size=215,
prob=0.15) )
```

Repare que o valor correto é o assinalado pela seta, ou seja, o degrau superior da escada. Para este ponto, a Normal estimou uma probabilidade acumulada menor, quase na metade do degrau. Vamos ver as diferenças?

```
> pbinom(30, size=215, prob=0.15)
[1] 0.3767186
> pnorm(30, mean=215*0.15, sd=sqrt(215*0.15*(1-0.15)))
[1] 0.3336915
```

Uma diferença considerável, dependendo do que estamos fazendo com essa distribuição. Isso pode fazer por exemplo um IC 95% conter um valor para um caso e não contê-lo no outro. Aí entra a correção de cotinuidade. Ela serve para chegar esse ponto da Normal mais para perto daquele degrau. Vamos ver como funciona: basta acrescentar 0.5 aos pontos de x:

```
acumulada.norm.corr<-pnorm(x+0.5, mean=215*0.15, sd=sqrt(215*0.15*(1-
0.15)))
points(x, acumulada.norm.corr, pch=19)
```

Veja como todos os pontos fechados estão muito mais próximos do degrau do que os pontos abertos. Vamos ver agora a diferença:

```
> pbinom(30, size=215, prob=0.15)
[1] 0.3767186
> pnorm(30.5, mean=215*0.15, sd=sqrt(215*0.15*(1-0.15)))
[1] 0.3690977
```

Duas proporções independentes

Teste para proporções aproximado pela Normal

A mesma função `prop.test()` pode ser usada para comparar também duas ou mais proporções. Nesse caso, em vez de entrar números, devemos entrar vetores com o número de sucessos e o número de experimentos em cada grupo, respectivamente.

A teoria por trás desse teste é simples e consiste em considerar que a diferença *d* entre as proporções sendo testadas seguirá uma distribuição aproximadamente Normal, com média zero e

variância $V(p_d) = (1/n_1 + 1/n_2) \times p(1-p)$, para um p igual para ambas as amostras – o que é verdade sob a nossa hipótese nula, certo?

H_0: $p_1 = p_2$

H_1: $p_1 \neq p_2$

Logo, a quantia

$$u = \frac{d}{\sqrt{\hat{V}(p_d)}}$$ pode ser testada em relação a uma Normal-padrão (e, claro, u^2 pode ser testada em relação a uma Qui-quadrada com 1 grau de liberdade). O p comum nesse caso é simplesmente a soma do número total de sucessos em ambos os grupos dividido pelo tamanho total da amostra, ou seja, uma média ponderada pelo tamanho: $\hat{p} = (x_1 + x_2)/(n_1 + n_2)$

Vamos usar um exemplo novamente do livro, mas sem entrar no mérito do contexto aqui. Vamos comparar um grupo com 9 sucessos em 12 tentativas e outro com 4 sucessos em 13 tentativas):

```
> prop.test(c(9,4),c(12,13))

        2-sample test for equality of proportions with continuity
correction

data:  c(9, 4) out of c(12, 13)
X-squared = 3.2793, df = 1, p-value = 0.07016
alternative hypothesis: two.sided
95 percent confidence interval:
 0.01151032 0.87310506
sample estimates:
   prop 1    prop 2
0.7500000 0.3076923
```

Há algo de estranho nessa saída, não é mesmo? Você saberia dizer o que é? As aproximações usadas nesse teste apresentam problemas quando queremos calcular um IC 95% para a diferença das proporções, e portanto uma aproximação que foge do escopo desta aula é usada. Mas para o p-valor é possível fazer-se o cálculo e deixaremos as explicações para um exercício.

Teste do Qui-quadrado

Até agora, nós vimos as proporções serem testadas como diferenças entre proporções propriamente ditas. Você deve até estar estranhando: onde estão as tabelas 2 x 2? E o teste do Qui-quadrado para estas tabelas que a gente vê a toda hora relatado em trabalhos, artigos, etc? Pois é, se você estava ansioso, chegou a hora.

Vamos usar o mesmo exemplo acima para trabalhar, mas agora digamos que se trata de um estudo de caso-controle, onde teremos uma certa condição sendo estudada em relação a uma exposição qualquer. Faça assim:

```
tabela<-matrix(c(9,4,3,9),2)
tabela<-cbind(tabela,apply(tabela,1,sum))
tabela<-rbind(tabela,apply(tabela,2,sum))
rownames(tabela)<-c("Casos","Controles", "Total")
colnames(tabela)<-c("Expostos","Não Expostos", "Total")
```

Teremos então a nossa tabela 2 x 2:

```
> tabela
          Expostos Não Expostos Total
```

```
Casos            9            3    12
Controles        4            9    13
Total           13           12    25
```

Muito bem. Vamos ver algumas características dessa tabela. Primeiro ela contém 4 células no seu corpo (daí o nome 2 x 2) e ainda uma terceira linha e uma terceira coluna, com os totais das linhas e das colunas, respectivamente. Elas se chamam as *marginais* da tabela.

Mas será que existe alguma relação entre esta tabela e as proporções que estávamos estudando? Claro que existe: se eu disser que ser exposto é um sucesso em uma Binomial, então nós podemos recuperar exatamente as proporções que estávamos vendo anteriormente, veja:

```
> tabela/tabela[,3]
          Expostos Não Expostos Total
Casos     0.7500000    0.2500000     1
Controles 0.3076923    0.6923077     1
Total     0.5200000    0.4800000     1
```

Confira com a saída acima e veja se as proporções de exposição entre casos e controles não correspondem exatamente ao que acabamos de testar. Bem simples, não é? Mas repare que neste caso as proporções são de fato diferentes e eu tenho que testar uma hipótese nula qualquer. Nós mencionamos anteriormente que sob a hipótese nula, ambas as proporções seriam iguais, e que uma maneira de fazermos isso era calcular uma proporção única para os dois grupos "ponderada" pelo tamanho da amostra em cada um deles. Isso equivale a somar todos os sucessos e dividir pelo número total da amostra, não é? Nesse caso seria 13/25, certo? Exatamente o que aparece na marginal dessa tabela aí em cima: 0.52.

Ora, mas nesse caso, nós podemos reconstruir essa tabela, sem alterar obviamente o número de casos ou de controles e nem o de expostos e não expostos, mas apenas rearranjando o corpo da tabela para que ela nos diga os números esperados nessa tabela, caso as proporções em ambos os grupos fossem iguais, certo? Basta aplicarmos a nossa proporção esperada àquela coluna de totais, não é mesmo? Ora, se temos um total de 12 casos e 0.52 ou 52% deles seriam esperados terem sido expostos, então, 6.24 pessoas, em média estariam expostas nesse grupo. Da mesma forma, para o grupo de 13 controles, 52% estariam expostos, ou seja, 6.76 pessoas, em média.

Na verdade, poderíamos fazer a conta para cada uma das caselas, mas acontece que se mantivermos as marginais fixas e calcularmos essa proporção para apenas uma delas, as outras necessariamente terão valores fixos também, bastando subtrair as marginais para obtê-los (menos trabalho que multiplicar, né?) Ou seja, mantidas as marginais fixas, só temos a liberdade de alterar 1 casela, as outras serão fixas... Hummm... 1 grau de liberdade... Eu já ouvi falar nisso...

Acontece que este procedimento corresponde àquelas milhares de multiplicações e divisões que você já deve ter visto em algum lugar para calcular a tabela esperada. Nós não faremos isso aqui. Vamos deixar o R fazer para nós:

```
> chisq.test(tabela[1:2,1:2])$expected
          Expostos Não Expostos
Casos         6.24         5.76
Controles     6.76         6.24
```

Tivemos que retirar as marginais, pois esta função só aceita o corpo da tabela para os cálculos.

Muito bem, e para esta disposição você deve ter aprendido que o somatório para as 4 células desta tabela para as diferenças ao quadrado dos valores observados e esperados ao quadrado sobre os valores esperados seguem uma distribuição Qui-quadrada com 1 grau de liberdade (olha ele aí, gente!!!), isto é:

$$\sum \frac{(O-E)^2}{E} \sim \chi^2_1$$

Vamos fazer a conta no R, para ver o que temos:

```
> esperado<-chisq.test(tabela[1:2,1:2])$expected
> sum(((tabela[1:2,1:2]-esperado)^2)/esperado)
[1] 4.890902
```

Agora, basta calcular um p-valor de uma Qui-quadrada (1):

```
> pchisq(sum(((tabela[1:2,1:2]-esperado)^2)/esperado),1,lower.tail=F)
[1] 0.02699857
```

Sem problemas, né? Vamos só conferir agora com o R:

```
> chisq.test(tabela[1:2,1:2])

        Pearson's Chi-squared test with Yates' continuity correction

data:  tabela[1:2, 1:2]
X-squared = 3.2793, df = 1, p-value = 0.07016
```

Eu tenho certeza que você já sabia que não ia dar certo... Afinal, cadê a correção de continuidade? Pois é, na verdade nós fazemos:

$$\sum \frac{(|O-E|-0.5)^2}{E} \sim \chi_1^2$$

No R:

```
> sum(((abs(tabela[1:2,1:2]-esperado)-0.5)^2)/esperado)
[1] 3.279350
> pchisq(sum(((abs(tabela[1:2,1:2]-esperado)-
0.5)^2)/esperado),1,lower.tail=F)
[1] 0.07015673
```

Teste Exato de Fisher

É claro que temos uma versão exata para testar duas proporções, mas devido às dificuldades de se testar proporções (nós já vimos várias delas), o teste empregado para este fim é um pouco diferente do que estávamos já acostumados a lidar aqui. É o famoso teste exato de Fisher. Ele é baseado na verdade em distribuições de tabelas 2 x 2, essas que nós acabamos de ver.

Soa estranho falar em distribuição de tabelas, não é mesmo? Mas é isso mesmo: É possível calcular-se probabilidades de termos uma determinada configuração em uma tabela 2 x 2, sob a hipótese nula de que não há diferença entre as proporções a serem testadas.

Essa idéia é baseada nessa mesma propriedade de podermos mudar a configuração do corpo de uma tabela 2 x 2, mantidas as marginais fixas e alterando apenas uma das caselas, já que as outras estarão determinadas automaticamente. Aliás, tente fazer esta brincadeira e veja se não é verdade.

Ora, nesse caso, podemos determinar probabilidades para determinados valores a serem encontrados em uma casela, por exemplo a casela superior esquerda (chamada de "a") em relação a margens fixas. Acontece que este tipo de experimento poder ser descrito por uma distribuição muito importante, porém pouco citada em cursos de estatística básica. É a distribuição Hipergeométrica.

Para entender o que esta distribuição faz, vamos voltar um pouco ao nosso exemplo da primeira aula, onde tínhamos uma urna com bolinhas verdes e azuis. Digamos que eu queira retirar bolinhas desta urna, sem reposição, e estou interessado em saber a probabilidade de retirar x bolinhas azuis, de uma urna com m bolinhas azuis e n bolinhas verdes, em k retiradas. Uma

pergunta então que essa distribuição responderia é: qual a probabilidade de eu retirar 1 (*x*) bolinha azul, dentre 2 (*k*) tentativas, sendo que há 7 bolinhas azuis (*m*) e 3 bolinhas verdes (*n*) sem reposição.

Essa podemos fazer sem problema, né? Para sair somente uma em duas tentativas temos duas possibilidades: ou saiu na primeira ou saiu na segunda tentativa; sem reposição, seria o seguinte:

```
> ((7/10)*(3/9))+((3/10)*(7/9))
[1] 0.4666667
```

Certo? Parecido com o que a gente viu naquele exercício, né? Pois bem, podemos usar a distribuição Hipergeométrica para nos ajudar a ver esse número:

```
> dhyper(x=1, m=7, n=3, k=2)
[1] 0.4666667
```

Tudo muito bonito, mas o que isso tem a ver com as tabelas 2 x 2??? Que papo estranho é esse de bolinhas?

Acontece que podemos fazer uma analogia entre estas tabelas e as bolinhas. Vamos ver. Imagine que os expostos são o total de bolinhas verdes e os não expostos, as azuis e que os casos são o número total de bolinhas que eu vou sortear, sem reposição. Ora, os controles seriam apenas o total menos os casos, veja:

```
                Verdes Azuis Total
Sorteados            9     3    12
Total-Sorteados      4     9    13
Total               13    12    25
```

Veja só: eu pergunto agora: qual é a probabilidade de eu sortear 9 bolinhas verdes, em 12 tentativas, sendo que temos 13 bolinhas verdes e 12 bolinhas azuis, sem reposição? A sua dúvida pode surgir aqui: por que sem reposição? É fácil: como os 12 sorteados são fixos, se o número de verdes mudar, o número de azuis também muda, ou seja, há dependência entre esses fenômenos, caracterizando a não reposição.

Bem, seguindo esse raciocínio, podemos então calcular qual a probabilidade de termos 9 ou mais bolinhas verdes sorteadas, não é mesmo? Seria a soma de termos 9 ou 10 ou 11 ou 12:

```
> sum(dhyper(x=9:12, m=13, n=12, k=12))
[1] 0.03406053
```

Ora, isso seria equivalente a testar a probabilidade de, dada a hipótese nula, a proporção de casos expostos ser maior ou igual a 0.75, lembra da tabela lá em cima?

```
           Expostos Não Expostos Total
Casos     0.7500000    0.2500000     1
Controles 0.3076923    0.6923077     1
Total     0.5200000    0.4800000     1
```

Bem, vamos então aplicar um teste de Fisher unidirecional, para ver a chance de uma observação ser tão extrema ou mais do que essa. Vamos usar a opção `alternative="greater"`:

```
> fisher.test(tabela[1:2,1:2], alternative="greater")

        Fisher's Exact Test for Count Data

data:  tabela[1:2, 1:2]
```

```
p-value = 0.03406
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 1.148427      Inf
sample estimates:
odds ratio
  6.180528
```

Hummm. Bateu direitinho o p-valor. Mas e o bilateral? Como é que se faz. Bem, aí fica parecido com a Binomial, que nós vimos. Calculamos o afastamento desse valor (9) para o valor esperado nessa casela (6.24) e calculamos o simétrico de 9 em relação a esse valor:

```
> 6.24-(9-6.24)
[1] 3.48
```

Como 3.48 não é inteiro, vamos calcular para valores iguais ou mais extremos que 3. No total, teríamos:

```
> sum(dhyper(x=c(0:3,9:12), m=13, n=12, k=12))
[1] 0.04717997
```

Vamos comparar agora com o teste bilateral:

```
> fisher.test( )

        Fisher's Exact Test for Count Data

data:  tabela[1:2, 1:2]
p-value = 0.04718
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.9006803 57.2549701
sample estimates:
odds ratio
  6.180528
```

Bem, para finalizar precisamos fazer algumas considerações sobre essa saída, especialmente as linhas abaixo do p-valor. Vamos devagar:

```
alternative hypothesis: true odds ratio is not equal to 1
```

Repare que agora o nosso teste de hipóteses não diz respeito a proporções, mas sim à *odds ratio* (OR) da qual você já deve ter ouvido falar bastante. Para refrescar a memória, uma *odds* é um valor que indica quantas chances de ganhar algo versus a chance de perder algo. Assim, a *odds* de um caso ser exposto é de 9:3, ou se preferir 3:1 ou 3 simplesmente, enquanto a *odds* de um controle ser exposto é de 4:9, ou 0.44. Essas chances podem também ser expressas através de suas proporções. Assim, a chance de um caso ser exposto é 0.75:1-0.75, ou 0.75:0.25 = 3. São equivalentes.

São equivalentes de tal sorte que comparar se p1 – p2 = 0 é o mesmo que comparar se $\frac{p_1/(1-p_1)}{p_2/(1-p_2)} = 1$, ou seja, se a razão das *odds* é igual a 1. Esta é a famosa OR.

```
95 percent confidence interval:
  0.9006803 57.2549701
```

Tudo estaria indo bem, não fosse esse IC para a OR, não é mesmo? Notou algo de estranho? Não? Então olhe o p-valor calculado de novo. E agora? Pois é, não bate, não é mesmo? Bem, esse é um problema que temos por causa das nossas aproximações, mesmo em um teste exato. Mas tem um outro problema. Olha só a estimativa que o R dá para a OR:

```
sample estimates:
odds ratio
  6.180528
```

Você que já está cansado de calcular ORs no seu curso de Epidemiologia, calcule para mim a OR da nossa tabela, por favor:

```
           Expostos Não Expostos Total
Casos             9            3    12
Controles         4            9    13
Total            13           12    25
```

Calculou? Conferiu para ver se está certo? Bateu com esse resultado aqui do R?

Pois é, não bate né? Alguém já tinha te dito que existe mais de um tipo de OR? Não? Aconteceu o mesmo comigo quando descobri que existe. Não se preocupe, eu conheço essa sensação de decepção. Parece muito quando você descobre que Papai Noel não existe...

;-)

Acontece que o que todos nós aprendemos e usamos, a famosa razão dos produtos cruzados é chamada de estimador de máxima verossimilhança incondicional da OR. Quer dizer que existe uma estimador condicional? Isso mesmo! E é essa que o R calcula... Isso porque vários estudos mostram que este estimador é mais acurado que a razão dos produtos cruzados para estimar o risco; mas isso é uma discussão para a Epidemio, não é mesmo? Bom, a "má" notícia é que esse estimador não pode ser calculado na mão, pois depende de métodos iterativos, e portanto não poderemos refazê-lo aqui... O IC então, nem me pergunte!!!

:-(

Mas não fique tão triste. Vamos implementar uma função para calcular a OR que você conhece e de quebra calcular um IC aproximado para essa OR.

O método aproximado mais popular para se calcular o IC de uma OR é a que você provavelmente já viu, que se chama método de Woolf, que é baseada como você já deve ter desconfiado em uma aproximação normal. Vamos criar uma função no R para calcular isso:

```
or.woolf <- function(x, alfa=0.05)
{
      y<-c((x[1,1]*x[2,2])/(x[1,2]*x[2,1])) # Calculando a OR
      z<-exp(log(y)+(c(-1,1)*qnorm(1-
(alfa/2))*sqrt((1/x[1,1])+(1/x[1,2])+(1/x[2,1])+(1/x[2,2])))) # Calculando o IC
      round(c("OR"=y, "IC"=z),3)
}
```

Bem, o cálculo da OR no código acima não deve deixar dúvidas (se ainda deixa, preocupe-se!!! Já estava na hora de ser entendido.) O cálculo do IC propriamente dito, ficará para um exercício para você.

Tudo muito bonito, mas eu mencionei que esse é o método mais popular. Isso: significa que existem outros... Pelo menos um outro é também badalado por aí, e foi proposto por Miettinen, e esse é baseado no resultado do teste Qui-quadrado. Vamos ver como isso funciona. A equação para o cálculo desse IC é até bem simples:

$IC\,100\times(1-\alpha)\% = \widehat{OR}^{1\pm\sqrt{X^2_{1,1-\alpha}/X^2}}$, onde X^2 nada mais é do que o valor da estatística do Qui-quadrado com correção de continuidade que o R calcula para nós. Bem, a implementação é também fácil:

```
or.miett <- function(x, alfa=0.05)
{
      y<-c((x[1,1]*x[2,2])/(x[1,2]*x[2,1])) # Calculando a OR
      qui<-chisq.test(x)$statistic
      z<-y^(1+(c(-1,1)*sqrt(qchisq(1-alfa, 1)/qui)))
      round(c("OR"=y, "IC"=z),3)
}
```

Não fique aflito, vamos usar essas funções em um exercício...

Teste de McNemar

Ainda falando em duas amostras para proporções, precisamos também de um teste para dar conta de estudos pareados, como vimos na última aula, no caso de um teste *t* pareado. Esse tipo de estudo é muito comum em Epidemiologia, como a maioria de vocês já deve ter percebido.

Não abordaremos os estimadores para a razão dos pares discordantes, assunto certamente abordado pela Epidemio, mas vamos ver um teste Qui-quadrado específico para testar dados pareados, que é chamado de teste de McNemar.

Como o nosso livro não comenta esse teste, vamo usar um exemplo do livro "Fundamentals of Biostatistics", de Bernard Rosner, 5ª edição, página 376, Exemplo 10.21, que trata de dois regimes diferentes de quimioterapia em mulheres, sendo que o desfecho em questão é número de mulheres que sobrevivem em 5 anos após a cirurgia. Para tornar os grupos sob diferentes tratamentos mais "comparáveis", pares são formados em respeito à idade e a uma classificação clínica.

Vamos ver o resultado desse estudo na clássica apresentação de uma tabela para dados pareados. No R:

```
      sobrevida<-matrix(c(526, 5, 16, 90), nr=2, dimnames = list("Tratamento A"
= c("Sobreviveu", "Faleceu"), "Tratamento B" = c("Sobreviveu", "Faleceu")))
```

O objeto `sobrevida` ficou assim:

```
> sobrevida
            Tratamento B
Tratamento A Sobreviveu Faleceu
  Sobreviveu         526      16
  Faleceu              5      90
```

Não causa nenhuma surpresa para quem já viu esse tipo de resultado, onde a tabela retrata o resultado dos pares e não dos indivíduos em relação aos tratamentos e também aos desfechos. Para aplicar o teste de McNemar, basta fazermos:

```
> mcnemar.test(sobrevida)

        McNemar's Chi-squared test with continuity correction

data:  sobrevida
McNemar's chi-squared = 4.7619, df = 1, p-value = 0.02910
```

A interpretação da saída é exatamente a mesma que a do teste não pareado. A que conclusão você chega, nesse caso?

Não vamos entrar em detalhes para esse teste.

k proporções independentes

É claro que muitas vezes você pode estar interessado em comparar mais de duas proporções. Neste caso estaremos diante de uma distribuição Multinominal e não mais uma Binomial apenas. Os detalhes não serão comentados, mas para uma Multinomial, nós teremos várias categorias com probabilidades diferentes de ocorrência.

As categorias em questão podem ser tanto nominais, ou seja, desprovidas de qualquer ordenação natural, quanto ordinais. Um exemplo do primeiro caso seria etnia, local de residência, etc. No segundo caso, grupo etário, escolaridade, renda, etc.

Para categorias nominais, a única alternativa é testar se as distribuições dessas categorias são independentes entre si, ou seja, não faz sentido fazer inferências sobre possíveis tendências presentes entre essas categorias. Claro que o contrário não é verdadeiro, e podemos testar categorias ordenadas dessa forma também.

Para usar um exemplo único, vamos seguir o mesmo do nosso livro-texto e usar o banco `caesarian` que contém informações sobre realização de cesarianas e tamanho de calçado de mulheres (medida britânica). Faça

```
data(caesarean)
caesar.shoe
```

E veja a disposição dos dados. Muito bem, agora podemos usar a nossa já conhecida função `prop.test()` para realizar um teste aproximado para independência entre as proporções. Claro que você já deve estar imaginando que o teste é bem semelhante ao que nós já vimos há pouco.

O detalhe é que teremos que entrar com dois vetores, um para os sucessos em cada grupo de tamanho de sapato e outra para o total de experimentos, da mesma forma que fizemos com duas proporções apenas. Nesse caso, teremos que selecionar a primeira linha e somar a primeira com a segunda, para termos os vetores que precisamos:

```
cesaria.sim<-caesar.shoe["Yes",]
cesaria.total<-caesar.shoe["Yes",]+caesar.shoe["No",]
```

Agora, basta aplicar o nosso já conhecido teste:

```
> prop.test(cesaria.sim,cesaria.total)

        6-sample test for equality of proportions without continuity
correction

data:  cesaria.sim out of cesaria.total
X-squared = 9.2874, df = 5, p-value = 0.09814
alternative hypothesis: two.sided
sample estimates:
    prop 1     prop 2     prop 3     prop 4     prop 5     prop 6
0.22727273 0.20000000 0.14285714 0.14583333 0.14814815 0.06666667

Warning message:
Chi-squared approximation may be incorrect in: prop.test(cesaria.sim,
cesaria.total)
```

Vamos olhar a saída desse teste devagar.

```
        6-sample test for equality of proportions without continuity
correction
```

Repare então que estamos testando 6 amostras independentes e que o teste foi feito sem correção de continuidade. Isso acontece porque no caso específico desse teste, quando estamos testando mais de duas proporções, demonstrou-se que a correção de continuidade não acrescenta nenhum ganho de precisão, como acontece nos demais casos.

```
data:  cesaria.sim out of cesaria.total
```

Apenas o que foi analisado: os eventos dentre os experimentos

```
X-squared = 9.2874, df = 5, p-value = 0.09814
```

O nosso Qui-quadrado, que é calculado da mesma forma que o para a tabela 2 x 2, mas agora a soma é em relação a todas as células e não mais as quatro somente, e repare que a gora estamos lidando com uma Qui-quadrada com 5 graus de liberdade. Não é coincidência, claro: essa estatística segue mesmo uma Qui-quadrada com $k - 1$ graus de liberdade, k sendo o número de grupos. Como temos 6 grupos, teremos 5 graus de liberdade. O p-valor, não tem mistério. Vamos até conferir esse:

```
> 1-pchisq(9.2874, 5)
[1] 0.0981354

alternative hypothesis: two.sided
sample estimates:
    prop 1     prop 2     prop 3     prop 4     prop 5     prop 6
0.22727273 0.20000000 0.14285714 0.14583333 0.14814815 0.06666667
```

A indicação do teste ser bilateral e as proporções para cada grupo

```
Warning message:
Chi-squared approximation may be incorrect in: prop.test(cesaria.sim,
cesaria.total)
```

E a indicação que o teste pode não estar correto. Alguém chutaria por que?

Vamos adiante, para facilitar essa resposta. Como no caso de duas proporções, aqui podemos também usar a função `chisq.test()` que apesar de ter uma saída mais resumida, nos permite calcular outras coisas. Vamos experimentar, lembrando que não precisamos ajeitar o objeto `caesar.shoe` nesse caso:

```
> chisq.test(caesar.shoe)

        Pearson's Chi-squared test

data:  caesar.shoe
X-squared = 9.2874, df = 5, p-value = 0.09814

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(caesar.shoe)
```

Repare que a saída é a mesma e que a mensagem continua lá. Vamos calcular esse Qui-quadrado? É igualzinho ao que a gente já fez antes:

```
> esperado<-chisq.test(caesar.shoe)$expected
Warning message:
Chi-squared approximation may be incorrect in: chisq.test(caesar.shoe)
> sum(((caesar.shoe-esperado)^2)/esperado)
[1] 9.287382
```

Que interessante! O R reclamou de novo! Acho que a resposta desse mistério da advertência deve estar no objeto `esperado`. Experimente verificar esse objeto...

É claro que podemos fazer também um teste exato de Fisher para esses dados, mas isso eu vou deixar para você se divertir...

Teste de tendência (linear)

Ainda para esse tipo de dados, podemos lançar mão de um teste de tendência, já que as nossas classes são ordenáveis. Esse teste será apresentado à guisa de curiosidade, e não exploraremos ele mais a fundo, embora considere importante mencioná-lo.

Esse teste na verdade pondera as proporções segundo um escore qualquer, que representa a ordem de cada uma das categorias. Em geral, se não temos nenhuma outra razão para fazer diferente, a ordem natural é uma seqüência simples indo de 1 até *k*, o número de classes. Não é à toa que o *default* da função que usaremos é exatamente essa.

A entrada da função é igual a da função `prop.test()`. Vamos ver como fica:

```
> prop.trend.test(cesaria.sim,cesaria.total)

        Chi-squared Test for Trend in Proportions

data:  cesaria.sim out of cesaria.total ,
 using scores: 1 2 3 4 5 6
X-squared = 8.0237, df = 1, p-value = 0.004617
```

Fácil, não é? Bastou acrescentar a palavra `trend` no meio da nossa velha conhecida, que aliás quer dizer tendência em inglês.

Não há muito o que comentar na saída dessa função. Lá está a nossa velha estatística e o nosso p-valor. De diferente só os escores, que como já disse são o *default* e os graus de liberdade, que agora voltou a ser igual a 1 de novo.

Dada a complexidade do assunto e o tempo disponível, não vamos detalhar os cálculos para esse teste e nem demonstrar a sua relação com o teste não-paramétrico de Wilcoxon para duas amostras (que é um caso particular desse teste), ficando isso para os mais curiosos.

Vamos apenas comentar o "linear" entre parênteses acima. É que esse teste funciona como se fosse uma regressão linear ponderada dessas proporções em relação aos escores. Sendo assim, temos que assumir que o efeito do tamanho do sapato é linear entre os grupos em questão, muito embora isso não precise ser garantido matematicamente, mas apenas assumido mesmo, dispensando qualquer tipo de verificação, por exemplo.

Quem quiser mais informações, procure em um livro de Bioestatística, com o Rosner, que eu mencionei anteriormente.

Tabelas *r* x *c*

Assim como o caso de duas proporções, que nada mais é do que uma tabela 2 x *k*, tabelas maiores, com *r* linhas e *c* colunas também podem ser testadas para independência tanto com a função `chisq.test()` quanto com a função `fisher.test()`. Esse tipo de teste serve para vários tipos diferentes de desenhos, o que modifica a interpretação dos resultados, mas não o teste em si. Esse assunto porém, foge do escopo desta aula, e vamos nos ater apenas no teste em si e não em aspectos de desenho de experimentos.

A idéia novamente é a mesma do teste anterior, e teremos que calcular valores esperados para todas as células, mantidas as marginais fixas e calcular a mesmíssima coisa de antes: $\sum \frac{(O-E)^2}{E}$, só que com um detalhe. Agora a distribuição não segue mais uma χ^2_1, mas sim

uma $\chi^2_{(r-1)(c-1)}$. Isso acontece, como você já deve ter adivinhado porque numa tabela desse tipo, fixadas as margens, podemos alterar até *(r-1)* x *(c-1)* caselas da tabela à vontade, antes das outras não poderem mais ser alteradas, ou seja, esses são os seus graus de liberdade. Aliás isso já aconteceu no caso da 2 x *k* , não é mesmo? Faça a conta.

Para exemplificar, vamos usar os dados referentes ao grau da resposta ao tratamento para Gonorréia com 3 diferentes esquemas terapêuticos. Este é um exercício do livro "Fundamentals of Biostatistics", de Bernard Rosner, 5ª edição, tabela 10.25 na página 415. Essa teremos que digitar também:

```
      gono.trat<-matrix(c(40,10,15,30,20,40,130,70,45), nr=3, dimnames =
list("Tratamento" = c("Penicilina", "Spectinomicina (baixa)", "Spectinomicina
(alta)"), "Resposta" = c("Esfregaço +", "Esfregaço +, Cultura +", "Ambos -")))
      gono.trat
```

Para fazer o teste:

```
> chisq.test(gono.trat)

        Pearson's Chi-squared test

data:  gono.trat
X-squared = 29.1401, df = 4, p-value = 7.322e-06
```

Não há muito a se comentar sobre essa saída. Poderíamos conferir os cálculos, como antes:

```
> esperado<-chisq.test(gono.trat)$expected
> sum(((gono.trat-esperado)^2)/esperado)
[1] 29.14007
```

Confira o p-valor. Conferem os graus de liberdade? Muito bem. Aqui também não temos a nossa correção de continuidade, pelos mesmos motivos já mencionados. Experimente fazer um teste exato também.

Bem, agora eu aposto que alguns de vocês gostariam de saber como identificar qual desses valores é de fato diferente dos demais, não é mesmo? Ainda que fosse para uma tabela 2 x *k*, como é que podemos satisfatoriamente saber qual grupo difere de qual? Bem, infelizmente esse não é um problema trivial, e modelos muito mais avançados dos que estamos lidando aqui devem ser usados. Vocês serão apresentados a esses modelos no próximo curso de estatística.

Uma advertência que deve ser feita é que não é correto, ao contrário do que muitos acham, testar um dos grupos contra a soma dos demais, repetindo esse procedimento para todos os grupos. Existem aí vários problemas entre comparações múltiplas e dependência entre os grupos.

Poder para proporções

O R também possui uma função para calcular poder, tamanho de amostra, etc para proporções, no mesmo estilo da que nós vimos para o teste *t*. A implementação, porém é só para duas amostras. O nome da função também é bastante parecido, `power.prop.test()` e os seus argumentos são bastante parecidos também, exceto pela ausência do desvio-padrão, já que no caso de proporções, ele depende dos valores de *p*. Por exemplo, para saber o poder de um teste para a diferença de uma proporção p_1 de 0.15 e p_2 de 0.30, com um alfa de 0.05 e 140 indivíduos em cada grupo, basta fazer:

```
> power.prop.test(p1=0.15, p2=0.30, n=140)

     Two-sample comparison of proportions power calculation
```

```
              n = 140
             p1 = 0.15
             p2 = 0.3
      sig.level = 0.05
          power = 0.8560383
    alternative = two.sided

NOTE: n is number in *each* group
```

Claro que este procedimento também serve para calcular tamanhos de amostra para poderes pré-estabelecidos.

Exercícios

1. Faça todos os passos do teste para diferenças de duas proporções independentes, de modo a obter o p-valor para a aproximação Normal e Qui-quadrada. Faça (a) sem a correção de continuidade e (b) com a correção de continuidade. (c) O que está estranho na saída mostrada na aula? Use o mesmo exemplo da aula. Mostre o código do software que você usou, ou os cálculos feitos à mão. Dica: para a correção de continuidade nesse caso, some, ao módulo da diferença das proporções a quantia $(1/2n_1)+(1/2n_2)$

2. Por que o teste do Qui-quadrado para as *k* proporções não é recomendado para o nosso exemplo da aula. Faça um teste que contorne esse problema.

3. Explique o código usado para o cálculo do IC da OR pelo método de Woolf

4. Volte ao exemplo do teste de Fisher. Calcule agora a OR e os ICs pelos métodos de Woolf e Miettinen. Todos os resultados são coerentes uns com os outros? Discuta esses resultados.

5. Usando a função `power.prop.test()` faça uma curva de poder para um experimento que testa duas proporções quaisquer. Estabeleça o problema, o teste de hipóteses e discuta diferentes tamanhos de amostra para o seu problema. Que sugestão você teria para construir essa curva para um experimento que usasse apenas uma amostra?

Do livro:

6. (7.2.) Em 747 casos de febre das Montanhas Rochosas, foram registrados 210 óbitos em uma determinada região dos EUA. Em outra região, de 661 casos, 122 faleceram. A diferença de letalidade da doença entre essas regiões é estatisticamente significante?

7. (7.3) Duas drogas foram testadas para úlcera péptica e comparadas quanto à sua efetividade. Os resultados foram:

	Curou	***Não curou***	***Total***
Pirenzepina	23	7	30
Tritiozina	18	13	31
Total	41	20	61

Estabeleça o teste de hipóteses para este experimento e faça um teste Qui-quadrado e também o teste exato de Fisher sobre esses dados e discuta as diferenças entre eles. Baseie-se tanto no p-valor quanto no IC 95%.

Do Rosner:

8. A tabela abaixo apresenta o resultado de um estudo feito no Líbano para aferir o efeito da viuvez na mortalidade. Para isso, Viúvos e viúvas foram pareados com pessoas casadas na mesma época, com idade semelhante (mais ou menos 2 anos) e mesmo sexo. Na tabela, o

número de pares onde pelo menos uma das pessoas faleceu até um certo período de seguimento:

TABLE 10.27 Effect of widowhood on mortality

	Males		Females	
Age (years)	n_1[a]	n_2[b]	n_1	n_2
36–45	4	8	3	2
46–55	20	17	17	10
56–65	42	26	16	15
66–75	21	10	18	11
Unknown	0	2	3	2
Total	87	63	57	40

[a] n_1 = number of pairs in which the widowed subject is deceased and the married subject is alive.
[b] n_2 = number of pairs in which the widowed subject is alive and the married subject is deceased.
Source: Reprinted with the permission of the *American Journal of Epidemiology*, *125*(1), 127–132, 1987.

O total de pares no estudo foi de 151 viúvos e 544 viúvas, sempre na proporção de 1:1. Considerando que todos os pares foram considerados para a análise, formula o teste de hipóteses, aplique o teste adequado, e comente o resultado.

Obs.: Como está meio apagado: n_1 são os pares onde o viúvo ou viúva faleceu e o casado está vivo e o n_2, o contrário.

Aula 6 - Proporções
Livro: páginas 129 a 138

1. Faça todos os passos do teste para diferenças de duas proporções independentes, de modo a obter o p-valor para a aproximação Normal e Qui-quadrada. Faça (a) sem a correção de continuidade e (b) com a correção de continuidade. (c) O que está estranho na saída mostrada na aula? Use o mesmo exemplo da aula. Mostre o código do software que você usou, ou os cálculos feitos à mão. Dica: para a correção de continuidade nesse caso, some, ao módulo da diferença das proporções a quantia $(1/2n_1)+(1/2n_2)$

(a) Primeiro vamos ver a saída sem correção, para conferir depois:

```
> prop.test(c(9,4),c(12,13), correct=F)

        2-sample test for equality of proportions without continuity
correction

data:  c(9, 4) out of c(12, 13)
X-squared = 4.8909, df = 1, p-value = 0.027
alternative hypothesis: two.sided
95 percent confidence interval:
 0.09163853 0.79297686
sample estimates:
   prop 1    prop 2
0.7500000 0.3076923
```

Agora basta seguir as equações apresentadas na aula mesmo. Precisamos calcular o *p* comum, a diferença e a variância:

```
prop<-13/25
dif<-(9/12)-(4/13)
varian<-((1/12)+(1/13))*prop*(1-prop)
```

Tranqüilo? Só segui as instruções. Agora podemos calcular o nosso *z*:

```
z<-dif/sqrt(varian)
```

E então calcular um p-valor:

```
> 2*pnorm(z, lower.tail=F)
[1] 0.02699857
```

Claro que isso seria o mesmo que fazer *z* ao quadrado e calcular o p-valor pela Qui-quadrada, certo?

```
> pchisq(z^2, df=1, lower.tail=F)
[1] 0.02699857
```

Aliás, podemos também conferir o valor dessa estatística com a saída acima:

```
> z^2
[1] 4.890902
```

(b) Novamente a saída para conferir:

```
> prop.test(c(9,4),c(12,13))

        2-sample test for equality of proportions with continuity
correction

data:  c(9, 4) out of c(12, 13)
X-squared = 3.2793, df = 1, p-value = 0.07016
alternative hypothesis: two.sided
95 percent confidence interval:
 0.01151032 0.87310506
sample estimates:
   prop 1    prop 2
0.7500000 0.3076923
```

É pena que haja um erro no enunciado e na verdade nós temos que diminuir a quantidade do módulo da diferença e não somar. Bem, faz parte... Vamos calcular a correção:

```
corr<-(1/24)+(1/26)
```

E agora vamos usar o mesmo código, apenas calculando o módulo e diminuindo a correção:

```
z1<-(abs(dif)-corr)/sqrt(varian)
```

Conferindo como na letra (a):

```
> 2*pnorm(z1, lower.tail=F)
[1] 0.07015673
> pchisq(z1^2, df=1, lower.tail=F)
[1] 0.07015673
> z1^2
[1] 3.279350
```

(c) O que há de estranho nessa saída é que o IC não é concordante com o p-valor relatado quando usamos a correção de continuidade: para um p-valor de 0.07, que não nos permitiria rejeitar a hipótese nula, temos um IC 95% que não contém zero, ou seja, pelo IC nós poderíamos rejeitar H_0.

2. Por que o teste do Qui-quadrado para as *k* proporções não é recomendado para o nosso exemplo da aula. Faça um teste que contorne esse problema.

Pelo motivo mais popular de todos: pelo menos um dos valores esperados para uma das caselas foi menor que 5. Vamos ver:

```
> chisq.test(caesar.shoe)$expected
           <4         4       4.5         5       5.5        6+
Yes  2.695157  4.287749  5.145299  5.880342  6.615385  18.37607
No  19.304843 30.712251 36.854701 42.119658 47.384615 131.62393
Warning message:
Chi-squared approximation may be incorrect in: chisq.test(caesar.shoe)
```

Isso necessita um teste exato: Fisher. Como tínhamos usado o exemplo `caesar.shoe`, vamos aplicá-lo:

```
> fisher.test(caesar.shoe)

        Fisher's Exact Test for Count Data

data:  caesar.shoe
```

```
p-value = 0.05766
alternative hypothesis: two.sided
```

Ainda assim não podemos rejeitar H_0.

3. Explique o código usado para o cálculo do IC da OR pelo método de Woolf

O código é simplesmente a aplicação dos cálculos da OR e sua variância e estabelecimento dos ICs. Vamos ver:

```
or.woolf <- function(x, alfa=0.05)
{
     y<-c((x[1,1]*x[2,2])/(x[1,2]*x[2,1])) # Calculando a OR
     z<-exp(log(y)+(c(-1,1)*qnorm(1-
(alfa/2))*sqrt((1/x[1,1])+(1/x[1,2])+(1/x[2,1])+(1/x[2,2])))) # Calculando o IC
     round(c("OR"=y, "IC"=z),3)
}
```

Primeiro a função é definida com 2 argumentos, um sem *default*, que é o `x` e outro com um valor pré-estabelecido, `alfa=0.05`. Em seguida, o objeto `y` recebe o cálculo da razão dos produtos cruzados (como está indicado, inclusive). Depois o objeto recebe aquela contaria toda para calcular o IC. O interessante é que foi possível fazer tudo em um passo apenas, o que pode parecer confuso inicialmente, mas foi possível criar apenas um objeto para isso. Repare que o último passo é exponenciar o IC calculado para o log da OR. Finalmente, a saída é arredondada para 3 casas decimais e o objeto ganha os nomes nas colunas "OR" e "IC".

4. Volte ao exemplo do teste de Fisher. Calcule agora a OR e os ICs pelos métodos de Woolf e Miettinen. Todos os resultados são coerentes uns com os outros? Discuta esses resultados.

Essa questão acabou ficando com um peguinha por causa de um descuido na hora de criar a função para calcular o IC pelo método de Miettinen. O problema é que para funcionar, você tem que entrar só o corpo da tabela como fizemos para o teste de Fisher:

```
or.miett(tabela[1:2,1:2]) #certo
    OR    IC1    IC2
 6.750  0.855 53.318

> or.miett(tabela) #ERRADO!!!
    OR    IC1    IC2
 6.750  1.243 36.667
```

O primeiro é que está certo. Lembre-se que o IC de Miettinen tem sempre que concordar com o teste do Qui-quadrado:

```
> chisq.test(tabela[1:2,1:2])

        Pearson's Chi-squared test with Yates' continuity correction

data:  tabela[1:2, 1:2]
X-squared = 3.2793, df = 1, p-value = 0.07016
```

No qual não rejeitamos H_0. Logo o esse IC tem que conter o zero. Para o IC de Woolf, tanto faz usar o corpo da tabela somente ou a tabela com as marginais:

```
> or.woolf(tabela[1:2,1:2])
    OR    IC1    IC2
 6.750  1.162 39.200
```

```
> or.woolf(tabela)
    OR    IC1    IC2
 6.750  1.162 39.200
```

Bem, mas o importante é notar como os intervalos são diferentes, e se compararmos com o teste de Fisher, veremos que na verdade o IC de Miettinen é bem mais próximo do IC calculado para o IC da OR de máxima verossimilhança condicional que o de Woolf. Além disso, e mais importante, eles são coerentes, ou seja ambos apontam para não rejeição da hipótese nula. Vamos conferir:

```
> fisher.test(tabela[1:2,1:2])

        Fisher's Exact Test for Count Data

data:  tabela[1:2, 1:2]
p-value = 0.04718
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.9006803 57.2549701
sample estimates:
odds ratio
  6.180528
```

Na verdade, recentemente o método de Woolf vem sendo bastante criticado por causa da sua instabilidade e falta de acurácia em algumas situações.

5. Usando a função `power.prop.test()` faça uma curva de poder para um experimento que testa duas proporções quaisquer. Estabeleça o problema, o teste de hipóteses e discuta diferentes tamanhos de amostra para o seu problema. Que sugestão você teria para construir essa curva para um experimento que usasse apenas uma amostra?

Vamos testar se a freqüência alélica do alelo delta-32 do gene codificador do receptor CCR5 é diferente entre caucasianos e negros. Nosso teste de hipóteses seria:

$H_0: p_1 = p_2$

$H_0: p_1 \neq p_2$

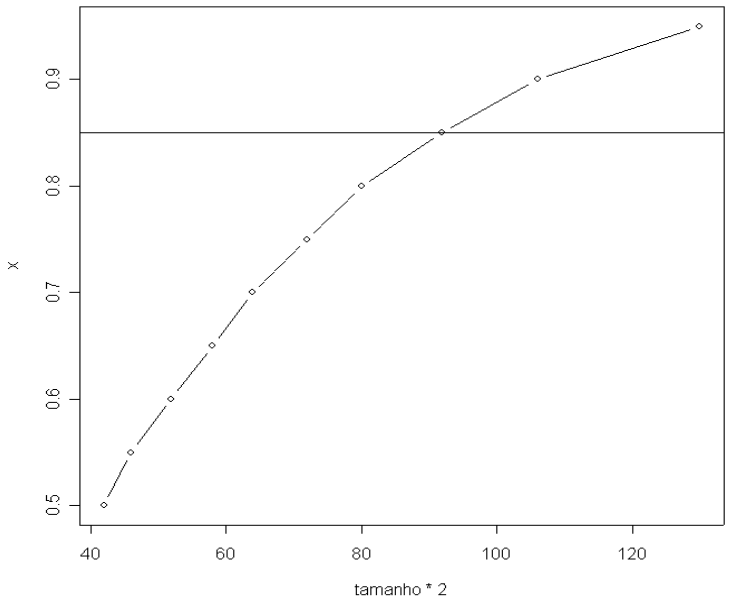
Baseado na literatura, a diferença de proporções é bastante grande para esse problema, da ordem de 20% para os caucasianos e 1% para os negros. Para manter tudo bem simples, vamos admitir que a diferença que queremos é essa mesmo e vamos estudar apenas o poder que teremos para tamanhos de amostra diferentes, mas vamos fazer ao contrário: fornecer o poder e ver o tamanho necessário:

```
x<-seq(0.5,0.95,0.05)
tamanho<-0
for (i in 1:length(x))
tamanho[i]<-ceiling(power.prop.test(p1=0.2, p2=0.01, power=x[i])$n)
```

O resultado em é na verdade o tamanho em cada grupo. Podemos então fazer um gráfico, multiplicando por 2 esse vetor:

```
> plot(tamanho*2,x, type="b")
> abline(h=0.85)
```

Veja o resultado abaixo:



Para conhecer exatamente o tamanho de amostra por grupo para termos 85% de poder:

```
> tamanho[x=="0.85"]
[1] 46
```

Esse tamanho de amostra parece bastante razoável para o nosso problema.

A sugestão para uma amostra, como não está implementado no R seria ou implementar ou então usar simulações para calcular os poderes para diferentes tamanhos de amostra.

Do livro:

6. (7.2.) Em 747 casos de febre das Montanhas Rochosas, foram registrados 210 óbitos em uma determinada região dos EUA. Em outra região, de 661 casos, 122 faleceram. A diferença de letalidade da doença entre essas regiões é estatisticamente significante?

Aqui temos várias alternativas para testar. Primeiro pela proporção

```
> prop.test(c(210,122),c(747,661))

        2-sample test for equality of proportions with continuity
correction

data:  c(210, 122) out of c(747, 661)
X-squared = 17.612, df = 1, p-value = 2.709e-05
alternative hypothesis: two.sided
95 percent confidence interval:
 0.05138139 0.14172994
sample estimates:
   prop 1    prop 2
0.2811245 0.1845688
```

Poderíamos criar uma tabelinha para usar o teste Qui-quadrado e Fishher também:

```
tabelinha<-matrix(c(210,122,747-210,661-122), nrow=2)
```

E agora podemos aplicar os testes:

```
> chisq.test(tabelinha)

        Pearson's Chi-squared test with Yates' continuity correction

data:  tabelinha
X-squared = 17.612, df = 1, p-value = 2.709e-05

> fisher.test(tabelinha)

        Fisher's Exact Test for Count Data

data:  tabelinha
p-value = 2.39e-05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.331814 2.246053
sample estimates:
odds ratio
  1.727031
```

Nesse caso, não há dúvida: todos os testes apontam para a rejeição de H0, portanto a diferença das proporções é estatisticamente significante. Fique à vontade para aplicar as nossas funções para OR também.

7. (7.3) Duas drogas foram testadas para úlcera péptica e comparadas quanto à sua efetividade. Os resultados foram:

	Curou	***Não curou***	***Total***
Pirenzepina	23	7	30
Tritiozina	18	13	31
Total	41	20	61

Estabeleça o teste de hipóteses para este experimento e faça um teste Qui-quadrado e também o teste exato de Fisher sobre esses dados e discuta as diferenças entre eles. Baseie-se tanto no p-valor quanto no IC 95%

Vamos criar a nossa tabela para esse problema

```
drogas<-matrix(c(23,18,7,13), nrow=2)
```

Agora podemos aplicar os testes. Primeiro o Qui-quadrado:

```
> chisq.test(drogas)

        Pearson's Chi-squared test with Yates' continuity correction

data:  drogas
X-squared = 1.6243, df = 1, p-value = 0.2025
```

Acontece que o R não fornece IC algum para esse teste. Vamos então usar o IC de Miettinen para a OR, até para ficar comparável com o teste de Fisher:

```
> or.woolf(drogas)
   OR   IC1   IC2
2.373 0.785 7.177
```

Tudo sem problemas. Os teste (claro) são coerentes e não nos permitem rejeitar H0. Vamos agora fazer o teste de Fisher:

```
> fisher.test(drogas)

        Fisher's Exact Test for Count Data

data:  drogas
p-value = 0.1737
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.6936416 8.4948588
sample estimates:
odds ratio
  2.339104
```

Sem problemas, né? Tudo bate direitinho, inclusive as ORs são bastante próximas.

Do Rosner:

8. A tabela abaixo apresenta o resultado de um estudo feito no Líbano para aferir o efeito da viuvez na mortalidade. Para isso, Viúvos e viúvas foram pareados com pessoas casadas na mesma época, com idade semelhante (mais ou menos 2 anos) e mesmo sexo. Na tabela, o número de pares onde pelo menos uma das pessoas faleceu até um certo período de seguimento:

TABLE 10.27 Effect of widowhood on mortality

	Males		Females	
Age (years)	n_1[a]	n_2[b]	n_1	n_2
36–45	4	8	3	2
46–55	20	17	17	10
56–65	42	26	16	15
66–75	21	10	18	11
Unknown	0	2	3	2
Total	87	63	57	40

[a] n_1 = number of pairs in which the widowed subject is deceased and the married subject is alive.
[b] n_2 = number of pairs in which the widowed subject is alive and the married subject is deceased.
Source: Reprinted with the permission of the *American Journal of Epidemiology*, 125(1), 127–132, 1987.

O total de pares no estudo foi de 151 viúvos e 544 viúvas, sempre na proporção de 1:1. Considerando que todos os pares foram considerados para a análise, formula o teste de hipóteses, aplique o teste adequado, e comente o resultado.

Obs.: Como está meio apagado: n_1 são os pares onde o viúvo ou viúva faleceu e o casado está vivo e o n_2, o contrário.

Nesse caso evidentemente estamos lidando com um estudo pareado. O teste de hipóteses pode ser estabelecido de diferentes maneiras. Vamos ver um bastante geral:

H_0: não há associação entre viuvez e mortalidade

H_1: há associação entre viuvez e mortalidade

O peguinha dessa questão aqui é que a tabela só mostra na verdade os pares discordantes, sendo que os demais têm que ser calculados. O problema é que não sabemos nada sobre os pares concordantes. Mas não tem problema: eles não contribuem em nada para o teste de McNemar. Logo, teremos um total de 247 pares discordantes. Logo, não importa o número que joguemos na

tabela. Vamos fazer o seguinte: vamos colocar metade dos pares concordantes em cada uma das 2 caselas. Eles são 695-247=378, logo colocaremos 189 em cada uma delas:

```
      viuvez<-matrix(c(189, 144, 103, 189), nr=2, dimnames = list("Viuvo" =
c("Sobreviveu", "Faleceu"), "Casado" = c("Sobreviveu", "Faleceu")))
      > viuvez
                  Casado
      Viuvo         Sobreviveu Faleceu
        Sobreviveu        189     103
        Faleceu           144     189
```

Podemos então calcular a OR para ver o risco de um viúvo falecer em relação aos casados. Isso seria apenas n1/n2 (repare que a tabela está ao contrário. Isso importa no resultado?):

```
      > 144/103
      [1] 1.398058
```

Bem, o risco é maior. Agora vamos ver se é significativamente maior:

```
      > mcnemar.test(viuvez)

              McNemar's Chi-squared test with continuity correction

      data:  viuvez
      McNemar's chi-squared = 6.4777, df = 1, p-value = 0.01092
```

A nossa conclusão é que há associação entre a viuvez e a mortatlidade, e que o risco de mortalidade é 40% maior entre os viúvos.

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 7 - ANOVA
Livro: páginas 111 a 121

Esta aula e também a próxima, nós vamos seguir o livro-texto mais de perto, e então nós vamos apenas comentar cada uma das seções de interesse, juntamente com o código utilizado em cada uma das passagens. Como não há muito o que modificar nesses capítulos, o aluno deverá ler os capítulos correspondentes e seguir a aula com os comentários.

ANOVA
- Comparações 2 a 2 e testes múltiplos
- Relaxando a pressuposição de homoscedasticidade
- Apresentação gráfica
- Teste de Bartlett

Teste de Kruskal-Wallis
- Comparações 2 a 2 e testes múltiplos

Exercícios

ANOVA

Vamos começar pela compreensão da breve teoria que o livro introduz para em seguida entrarmos na parte do teste propriamente dito.

É fundamental que se entenda o que acontece nessa decomposição a qual o livro se refere, tanto em relação às observações, como em relação às somas dos quadrados. Para nos ajudar nessa tarefa, vamos gerar um banquinho de exemplo:

```
groupa<-rnorm(5, 200, 25)
groupb<-rnorm(5, 110, 30)
groupc<-rnorm(5, 50, 30)
group<-c(rep("A",5),rep("B",5),rep("C",5))
grupos<-data.frame(valor=c(groupa, groupb, groupc), grupo=group)
attach(grupos)
```

Não acho que seja a maneira mais elegante de se fazer isso, mas criamos um *data frame* com uma variável chamada `valor` gerada a partir de Normais para 3 grupos diferentes com 5 observações por grupo e também uma variável `grupo` com os nomes dos grupos – A, B e C. Depois anexamos o objeto para nos facilitar a vida. Confira como ficou.

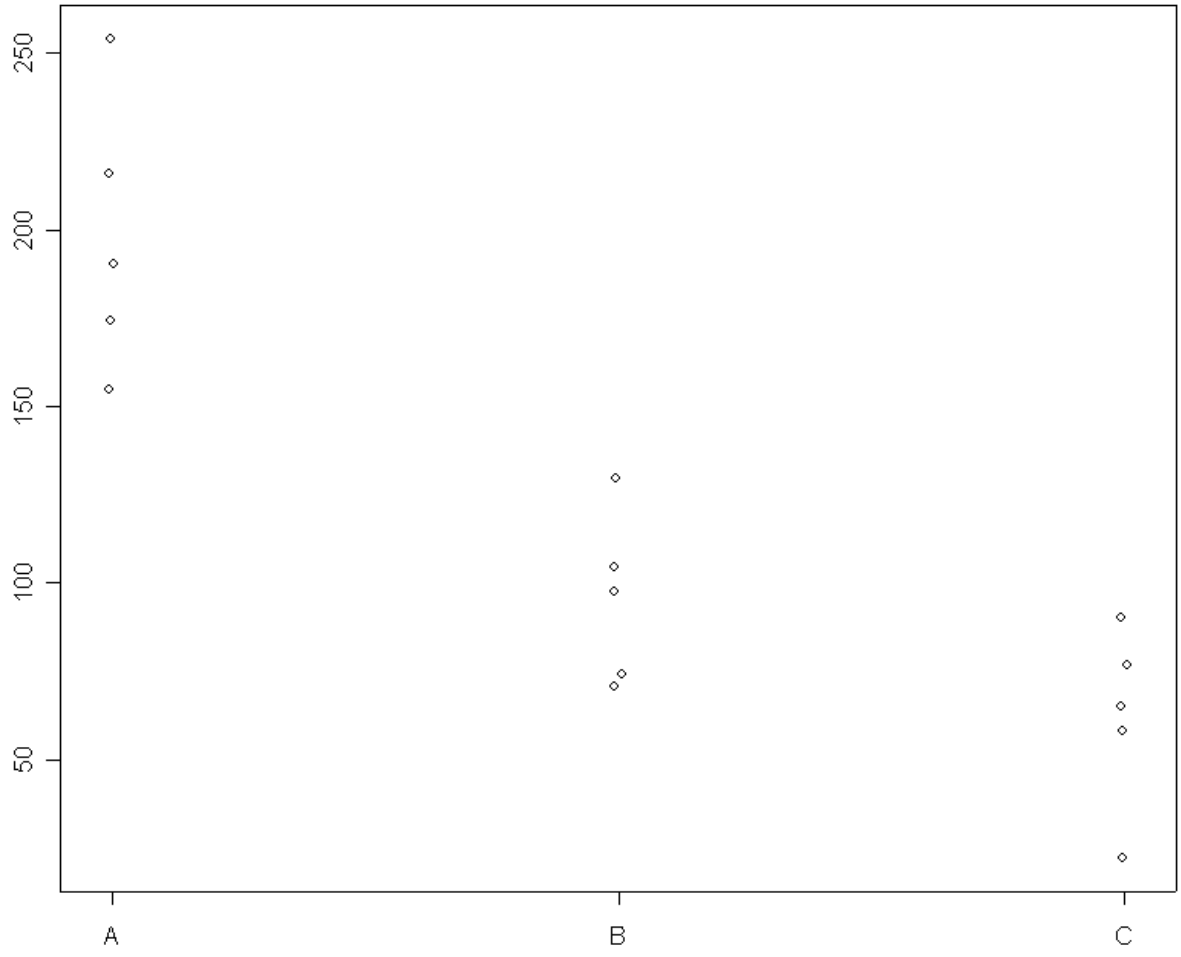
```
summary(grupos)
```

Vamos agora usar um gráfico que nunca usamos antes, mas que vai dar uma boa visualização desses dados. Os pontos parecem um pouquinho afastados uns dos outros em relação

ao eixo horizontal, mas isso é um artifício, gerado pelo argumento `"jitter"` na função abaixo, já que todos devem estar alinhados em relação ao seu grupo:

```
stripchart(valor~grupo, vert=T, "jitter", jit=0.01, pch=21)
```

Você deve estar vendo algo como esse gráfico aí embaixo: as observações por cada um dos grupos (não igualzinho, é claro, pois nós geramos isso aleatoriamente).



Muito bem, vamos guardar então esse gráfico por enquanto. Vamos tentar entender a relação que ele coloca. Ele diz que um ponto qualquer desse gráfico, x_{ij} (i.e. a j-ésima observação do grupo i) é determinado por 3 elementos diferentes: a média geral (sem levar em conta os grupos), a diferença entre a média do grupo e a média geral e a diferença entre a própria observação e a média do grupo. Agora, repare que isso é apenas um truque matemático. Repare o que ele fez. Começamos de uma igualdade:

$$x_{ij} = x_{ij} \quad .$$

Aí, somamos e diminuímos $\bar{x}_{\bullet}$, a média geral:

$$x_{ij} = \bar{x}_{\bullet} - \bar{x}_{\bullet} + x_{ij}$$

Ainda ficou a mesma coisa, certo? Agora, para finalizar, ele soma e subtrai $\bar{x}_i$, a média de cada grupo *i*:

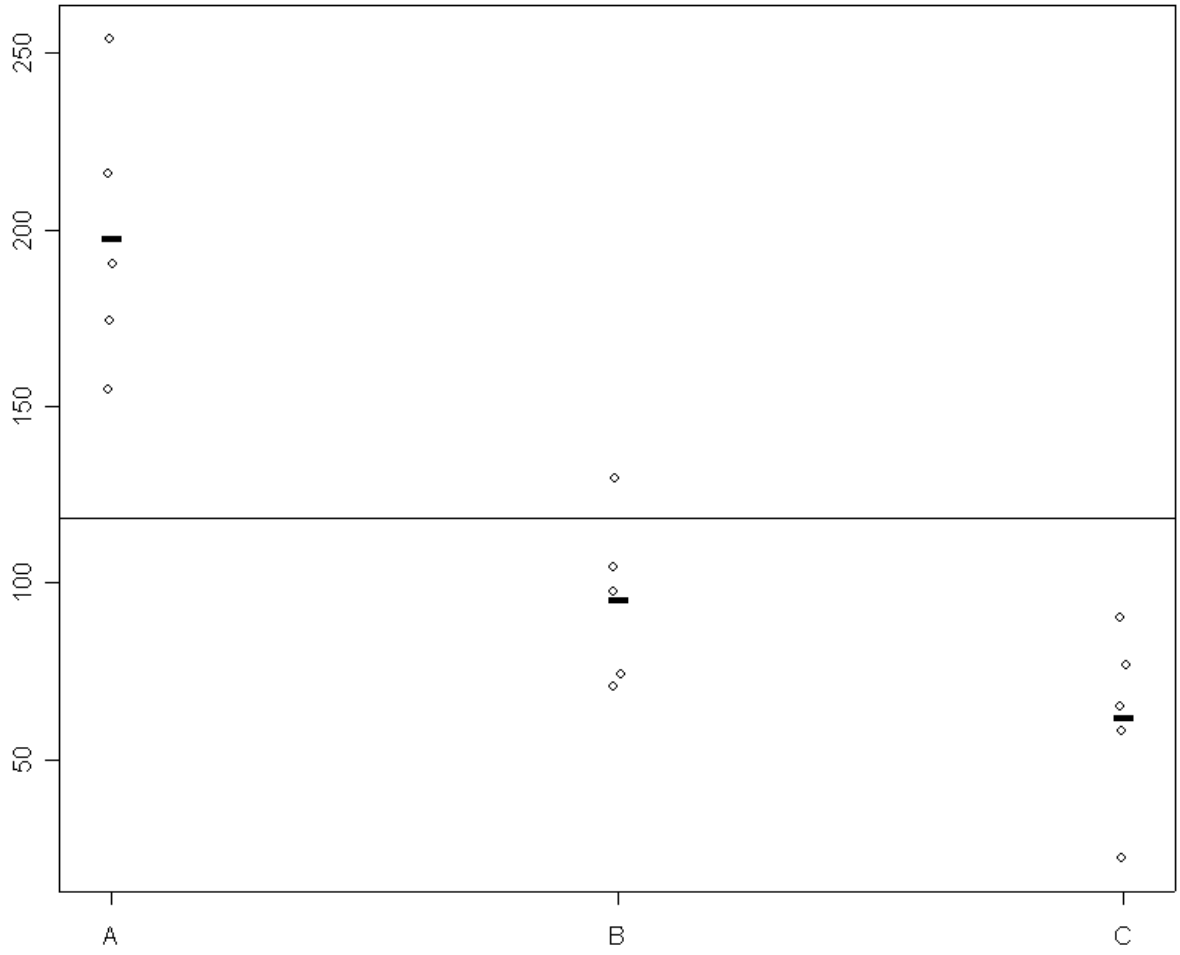
$$x_{ij} = \bar{x}_\bullet + (\bar{x}_i - \bar{x}_\bullet) + (x_{ij} - \bar{x}_i) \quad .$$

Isso é simples de conferir também no nosso gráfico, quer ver? Vamos calcular a média geral e a média de cada um dos grupos para o nosso banquinho de exemplo:

```
xbarra<-tapply(valor, grupo, mean)
media<-mean(valor)
```

Agora, vamos acrescentar esses cálculos no nosso gráfico:

```
points(1:3, xbarra, pch="-", cex=3)
abline(h=media)
```



A linha horizontal é a média geral e os tracinhos a média de cada grupo. Pegue um lápis e desenhe nesse gráfico agora as distâncias que acabamos de mencionar e veja se as relações não se mantêm.

Agora vamos ver a questão da variabilidade, também comentada no texto. Não fica muito claro como dessa relação se chega nas variabilidades. Sem entrar em detalhes, podemos reescrever essa igualdade assim, certo?

$$(x_{ij} - \bar{x}_{\bullet}) = (\bar{x}_i - \bar{x}_{\bullet}) + (x_{ij} - \bar{x}_i)$$

Pode-se então provar que se elevarmos esses termos ao quadrado e somarmos todas as observações em todos os grupos, essa relação se mantém. Vamos usar uma notação mais rígida que a do livro, especificando os *k* grupos e os n_i elementos de cada grupo:

$$\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_{\bullet})^2 = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(\bar{x}_i - \bar{x}_{\bullet})^2 + \sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2$$

O primeiro termo seria a dispersão de cada observação em relação à media geral, o segundo, a dispersão entre a média de cada grupo e essa mesma média geral e o terceiro a dispersão entre cada observação e a média do seu próprio grupo. Hummm, poderia dizer então que são as dispersões totais, entre os grupos e dentro dos grupos, respectivamente, não é mesmo?

Nós aprendemos que uma medida de variabilidade com boas características matemáticas era a variância, seja da população, seja da amostra, como nós vimos anteriormente. Ambas nada mais são do que a soma dos quadrados dos afastamentos (ou diferenças) entre as observações e a média dessas observações, só que dividido pelo número de observações (ou *n-1*, se for a amostral). Bem, mas acontece que se nós não dividíssemos por *N* ou *n-1*, ainda assim isso seria uma medida de variabilidade, certo? Só para lembrar a variância da amostra por exemplo:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Essa seria logicamente a variabilidade média total da nossa amostra. Mas agora nós temos uma amostra dividida em grupos, e nós precisamos de uma medida de variabilidade também. Mas espere aí. Essas relações lá de cima nada mais são do que variabilidades, não é mesmo? São somatórios de desvios das observações em relação a médias ao quadrado. Então o primeiro termo dessa relação é uma medida de dispersão total das observações. Usando a notação do livro:

$$SS_{total} = \sum_{i}\sum_{j}(x_{ij} - \bar{x}_{\bullet})^2$$

Eu sei que desde lá de cima já tinha complicado um pouco a notação por causa desse somatório duplo, não é? Mas isso ocorreu porque a gente tem que dividir os grupos e as observações, mas na verdade isso só quer dizer que é o somatórios das *j* observações dos *i* grupos, mais nada. Ela corresponde à nossa variabilidade total, mas não é a variância propriamente dita, pois ao contrário da variância, não representa uma variabilidade média. Agora você deve estar pensando por que não se usa a variância que nós já vimos que tem propriedades tão boas... Espere um pouco e veremos que essa também possui.

Ah, e *SS* é soma dos quadrados (de *sum of squares* em inglês.) No livro eles usam *SSD* soma dos quadrados das diferenças – mais descritivo...

Bem, acontece que como as observações, essa variabilidade pode também ser dividida em dois componentes, um que diz respeito ao próprio grupo (a variabilidade interna do grupo, as distâncias entre as observações pertencentes a um grupo e a média desse grupo) e ainda a variabilidade entre os grupos (as diferenças entre as médias dos grupos e a média total.) Aliás, para o nosso teste é nessa que estamos interessados, não é mesmo? Bem, estamos interessados em ambas...

Então podemos escrever esses componentes, como no livro, não é? Ah, mas já está lá, não precisa escrever de novo. Vamos logo fazer algo que ele não faz, que é conferir se isso é verdade ou não. Vamos começar conferindo quanto vale o SS_{total}, que é só fazer a soma das diferenças das observações para a média geral ao quadrado. Essa é fácil, pois nós já calculamos a média geral:

```
sum((valor-media)^2)
[1] 60884.83
```

Vamos agora calcular a soma dentro dos grupos. Esse é um pouquinho mais chato de fazer, porque a gente tem que subtrair as observações das médias dos seus grupos. Então temos que criar um vetor com esses valores. Veja se isso funciona:

```
rep(xbarra, each=5)
```

Nós repetimos os valores de `xbarra` 5 vezes cada, que é o número de observações em cada grupo, justamente. Ah, agora podemos fazer e fica bem parecido com a fórmula, veja:

```
> sum((valor-rep(xbarra, each=5))^2)
[1] 10910.57
```

Repare que com esse macete não precisamos nos preocupar com o duplo somatório, pois essa nossa função `rep()` "substituiu" um deles para nós.

E agora finalmente a soma entre os grupos. Para facilitar os cálculos, vamos usar a forma simplificada do livro:

$$SS_{entre} = \sum_i n_i (\bar{x}_i - \bar{x}_\bullet)^2$$

Aqui o macete são os n_i's que no nosso caso são 3 e todos iguais a 5. Aí facilita até:

```
> sum(rep(5,times=3)*(xbarra-media)^2)
[1] 49974.26
```

Repare que aqui fizemos uma operação vetorial, de outra maneira não daria certo. Veja se a soma desses dois não dá exatamente a soma total. Vou deixar como exercício indicar no gráfico acima onde estão esses componentes das variabilidades.

Muito bem. Sobre o teste F, além das explicações do livro, que me parecem claras, vamos recordar o que aprendemos na aula 5 sobre aquele teste para homogeneidade das variâncias, lembra? Pois é, isso é a mesma coisa, são somas de diferenças ao quadrado sobre os seus graus de liberdade. Só que nesse caso, nos interessa a direção desse teste, pois o que estamos procurando é se a variabilidade entre os grupos explica uma parte maior da variância total do que a variabilidade dentro dos grupos. Assim, como os graus de liberdade serão sempre *k-1* (as médias dos *k* grupos menos a média geral) para a soma entre e *n-k* (todas as observações menos as *k* médias dos grupos) para a soma dentro dos grupos, se dividirmos essas somas pelos seus graus de liberdade e fizermos a divisão, teremos exatamente o mesmo teste F para diferença de variâncias, que é mostrado no alto da página 113.

O detalhe é que neste caso será um teste unidirecional, pois só nos interessa testar se a variânca entre os grupos é maior do que a dentro dos grupos.

Repare também que estranhamente o que estamos testando aqui são variâncias, mas estamos interessados em saber se as médias dos grupos são diferentes.

Bem, vamos então ao exemplo do livro:

```
library (ISwR)
data (red.cell.folate)
attach(red.cell.folate)
summary(red.cell.folate)
```

Pela ordem do livro, a regressão linear já foi vista e é por isso que ele fala tanto sobre ela, comparando com a ANOVA. Não se preocupe com isso, na próxima aula nós vamos falar sobre isso.

Indo adiante com a análise mesmo:

```
anova(lm(folate~ventilation))
```

Segue então a explicação da saída do R, não creio que haja problemas quanto a isso. Note que esse banco tem 22 observações em 3 grupos, e portanto é sempre bom conferir os graus de liberdade. Claro que podemos usar aquelas continhas que fizemos anteriormente para conferir as contas, apenas mudando algumas coisas:

```
xbarra<-tapply(folate, ventilation, mean)
media<-mean(folate)
```

Até aqui nada mudou, só os nomes. Agora nós vamos ter um pouco de dificuldade em criar aquelas nossas repetições para as médias, não é? Vamos ver. Os grupos têm 8, 9 e 5 observações respectivamente e a gente precisa do `xbarra` repetido nessas freqüências, certo? Que tal tentarmos:

```
rep(xbarra, times=c(8,9,5))
```

Parece que funciona, né? Então vamos fazer assim para a soma dentro dos grupos:

```
ssw<-sum((folate-rep(xbarra, times=c(8,9,5)))^2)
```

A seguinte é até fácil, pois precisamos apenas de um vetor com esses valores de n_i:

```
ssb<-sum(c(8,9,5)*(xbarra-media)^2)
```

Confira os valores dos objetos `ssw` e `ssb` para ver se bate com os valores na tabela. Bem, agora para obter o valor de *F*, ficou fácil, né?

```
> (ssb/2)/(ssw/19)
[1] 3.711336
```

E o p-valor é derivado de uma distribuição $F_{2,19}$, lembra?

```
> pf((ssb/2)/(ssw/19), 2, 19, lower.tail=F)
[1] 0.04358933
```

Lembrando que o teste é unilateral e é claro que estamos olhando para a cauda superior dessa *F*. Bateu, né? Confira:

```
> anova(lm(folate~ventilation))
Analysis of Variance Table

Response: folate
            Df Sum Sq Mean Sq F value  Pr(>F)
ventilation  2  15516    7758  3.7113 0.04359 *
Residuals   19  39716    2090
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Bem, em seguida ele comenta sobre um possível erro se a sua variável não estiver corretamente codificada. Vamos só ver o exemplo certo aqui:

```
data(juul)
juul$tanner<-factor(juul$tanner, labels=c("I", "II", "III", "IV", "V"))
attach(juul)
```

```
summary(tanner)
anova(lm(igf1~tanner))
```

O passo-a-passo dessa fica como exercício para você. Note que não precisamos desanexar e anexar o banco nesse caso, pois não fizemos o exemplo errado que ele propõe.

Comparações 2 a 2 e testes múltiplos

Bem, vamos pular essa parte da regressão linear, já que nós não vimos esse assunto ainda e falar das comparações 2 a 2, já no quarto parágrafo da página 116.

O que queremos fazer na verdade é comparar os nossos grupos 2 a 2 para ver onde esta diferença existe. O teste *F* que fizemos nos diz apenas que pelo menos 2 desses grupos são diferentes entre si, mas não quais.

O problema é que estamos comparando várias coisas ao mesmo tempo e cada vez que comparamos, estamos sujeitos a cometer um erro. Esse erro nada mais é do que uma probabilidade de se achar um valor muito extremo em uma distribuição (pensando no p-valor, por exemplo). Acontece que essa probabilidade é multiplicada cada vez que nós comparamos um grupo com outro grupo. Isso quer dizer que se formos comparar pro exemplo uma variável com 10 grupos, teríamos 45 pares de grupos, e poderíamos ter então 0.05x(45) = 2.25, ou seja, cerca de 2 pares que tenham uma diferença estatisticamente significante apenas devido ao acaso, só porque eu multipliquei o meu erro.

Existem então várias maneiras de se corrigir esse fenômeno e uma das mais comuns é a correção de Bonferroni, que é também a mais simples e a mais conservativa. O procedimento é apenas multiplicar o p-valor calculado pelo número de combinações. A única diferença é que a variância conjunta dos grupos (que é igual à soma dos quadrados média dentro dos grupos) é usada para todos os pares, sob a pré-suposição que as variâncias são homogêneas.

Vamos ver como fica a variância conjunta para mais de duas amostras (para duas nós já vimos na aula 5, lembra?): $s^2_{conj} = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}$. Mas repare que se temos *k* grupos em vez de 2, poderíamos reescrever essa equação:

$$s^2_{conj} = \frac{\sum_{i=1}^{k}(n_i-1)s_i^2}{\sum_{i=1}^{k}(n_i-1)} \quad .$$

Mas podemos expandir essa equação para s_i^2 :

$$s^2_{conj} = \frac{\sum_{i=1}^{k}(n_i-1)\sum_{j=1}^{n_i}(x_{ij}-\bar{x}_j)^2/(n_i-1)}{n-k}$$. Veja que o denominador foi também alterado (só fiz a conta, é o somatório dos *n*'s de todos os grupos, que nada mais é do que o próprio *n* total, menos *k* vezes 1, que é *k*.) Agora, vamos deslocar esses somatórios e cortar os (n_i-1) :

$$s^2_{conj} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij}-\bar{x}_j)^2}{n-k}$$

Hummmm. Isso está parecendo a doma dos quadrados entre os grupos, dividido pelos seus graus de liberdade. E é isso mesmo!!! A variância conjunta é igual àquela saída da tabela da ANOVA, e podemos usá-la para os nossos cálculos. Podemos até conferir isso numericamente para

o nosso exemplo aqui. Vamos calcular $s^2_{conj} = \frac{\sum_{i=1}^{k} (n_i - 1) s_i^2}{\sum_{i=1}^{k} (n_i - 1)}$ no R e comparar com a tabela da ANOVA. Vamos primeiro criar um vetor para os n_i's e outro para os s_i^2 's:

```
ni<-c(8,9,5)
si<-tapply(folate, ventilation, var)
```

Agora podemos facilmente calcular:

```
> sum((ni-1)*si)/sum((ni-1))
[1] 2090.321
```

Arredondando confere, né? Bem, é essa variância que temos que usar para fazer os nossos cálculos 2 a 2. Mas agora, vamos ver o que acontece quando fazemos o nosso teste no R com a correção de Bonferroni:

```
> pairwise.t.test(folate, ventilation, p.adj="bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.042      -
O2,24h    0.464      1.000

P value adjustment method: bonferroni
```

Usamos então a função `pairwise.t.test()` com o argumento `p.adj="bonferroni"`, já que como você percebeu no livro o *default* não é esse.

Bem, o que essa correção faz na verdade é multiplicar o p-valor originalmente calculado pelo número de combinações. No nosso caso, para 3 grupos:

```
> choose(3,2)
[1] 3
```

Legal, então vamos ver quais seriam os p-valores sem a correção. Para isso basta usar o argumento `p.adj="none"`:

```
> pairwise.t.test(folate,ventilation, p.adj="none")

        Pairwise comparisons using t tests with pooled SD

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.014      -
O2,24h    0.155      0.408

P value adjustment method: none
```

Confira se valores corrigidos são esses valores aqui multiplicados por 3. Um deles não é, por que?

Bem, é claro que esses p-valores podem ser facilmente obtidos a mão, e esse será um exercício para você

Relaxando a pressuposição de homoscedasticidade

Como você deve ter aprendido, a ANOVA também exige que as variâncias entre os grupos seja homogênea, ou seja que haja homoscedasticidade entre os grupos. Assim como no caso do este *t* para duas amostras, e também graças a Welch, é possível fazermos um teste que não leva a homoscedaticidade em conta. Seria muito longo e tedioso mostrar, como fizemos para o teste *t* o caminho a ser seguido para essas contas, e portanto nos limitaremos a mostrar como o R faz esse teste.

Para o caso da comparação geral da ANOVA, teremos que usar uma função diferente da que usamos anteriormente. Trata-se da `oneway.test()` que implementa a correção de Welch. Vamos ver como fica:

```
> oneway.test(folate~ventilation)

        One-way analysis of means (not assuming equal variances)

data:  folate and ventilation
F = 2.9704, num df = 2.000, denom df = 11.065, p-value = 0.09277
```

Note que a saída é mais resumida que o teste anterior, mas que agora os graus de liberdade do denominador são fracionários. Repare que agora o p-valor não é mais significativo, o que pode significar, como diz o livro que a diferença que estávamos vendo era na verdade devida ao fato de um dos grupos terem não uma média maior, mas uma variância maior.

Esse mesmo procedimento pode ser usado para comparações múltiplas, usando a mesma função de antes, mas com um argumento extra, indicando que não queremos a variância (ou o desvio-padrão) agrupada. Vamos fazer diferente do livro e usar a correção de Bonferroni também:

```
> pairwise.t.test(folate, ventilation, pool.sd=F, p.adj="bonferroni")

        Pairwise comparisons using t tests with non-pooled SD

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.087      -
O2,24h    0.482      0.893

P value adjustment method: bonferroni
```

O argumento `pool.sd=F` pede então que usemos desvios-padrão não agrupados. Repare que agora nenhum p-valor é significativo também. O problema das variâncias diferentes entre os grupos também ocorre aqui, como vimos no teste geral acima.

Apresentação gráfica

Essa seção do livro é mais uma curiosidade de como apresentar esse tipo de dados no R. Nós já até usamos a função `stripchart()` no início dessa aula. Vou então fornecer o código para você não ter que digitar, mas não tenho nada a acrescentar ao texto:

```
xbar <- tapply(folate, ventilation, mean)
```

```
s <- tapply(folate, ventilation, sd)
n <- tapply(folate, ventilation, length)
sem <- s/sqrt(n)
stripchart(folate~ventilation, "jitter", jit=0.05,pch=16, vert=T)
arrows(1:3, xbar+sem, 1:3, xbar-sem, angle=90, code=3, length=0.1)
lines(1:3, xbar, pch=4, type="b", cex=2)
```

O código é bem parecido com o que usamos anteriormente para construir os gráficos dos intervalos de confiança, e essas funções podem ser melhor entendidas através da ajuda para cada uma delas, além da explicação do livro.

Teste de Bartlett

Não entendi porque esta seção está aqui e não antes da seção de relaxamento da homoscedasticidade, mas como estou seguindo o livro, deixa para lá. Como no caso de duas amostras, você pode querer comparar as variâncias de 3 ou mais amostras ao mesmo tempo para saber se deve ou não fazer a ANOVA com ou sem homoscedasticidade. O R tem implementado o teste de Bartlett (usuários do Epiinfo vão se lembrar dele).

Não vamos entrar em detalhes também sobre este teste, mas estaremos testando a homogeneidade das variâncias entre os grupos, e este teste, assim como o teste F que já tínhamos visto para duas variâncias na aula 5, também é bastante sensível a distribuições não-normais. Vamos ver como fica:

```
> bartlett.test(folate~ventilation)

        Bartlett test for homogeneity of variances

data:  folate by ventilation
Bartlett's K-squared = 2.0951, df = 2, p-value = 0.3508
```

Surpreendentemente, o teste não nos permite concluir que alguma das variâncias sejam diferentes, apesar dos nossos testes terem se modificado bastante, quando não assumimos homoscedasticidade anteriormente e apesar também do primeiro grupo parecer mais disperso que os demais (veja no livro ou faça o gráfico.) A normalidade pode ter nos atrapalhado aqui...

Teste de Kruskal-Wallis

Também não vamos nos estender muito nesse teste que nada mais é do que uma generalização do teste de Wilcoxon para mais de duas amostras, ou seja um teste não-paramétrico para mais de dois grupos independentes.

Os cálculos são na verdade bastante parecidos com o teste de Wilcoxon, mas vamos poupá-los das contas infindáveis. A idéia dos *ranks* é a mesma e agora estamos comparando vários conjuntos de *ranks* em vez de apenas 2.

A função `kruskal.test()` implementa esse teste no R:

```
> kruskal.test(folate~ventilation)

        Kruskal-Wallis rank sum test

data:  folate by ventilation
Kruskal-Wallis chi-squared = 4.1852, df = 2, p-value = 0.1234
```

Acrescento somente o comentário do livro sobre a eficiência do teste não-paramétrico em relação ao seu correspondente paramétrico, e é bom frisar que isso é verdade para qualquer teste não paramétrico, ou seja, se o teste paramétrico for aplicado sobre amostras que obedecem aos pré-

requisitos (e.g. normalidade), então o teste paramétrico será mais eficiente que o não-paramétrico. O problema sempre será assumir que esses pré-requisitos são verdadeiros.

Comparações 2 a 2 e testes múltiplos

O livro não comenta nada sobre comparações 2 a 2 para testes não paramétricos, mas eles também estão implementados no R. Vamos ver um exemplo com o mesmo banco de dados:

```
> pairwise.wilcox.test(folate,ventilation, p.adj="bonferroni")

        Pairwise comparisons using Wilcoxon rank sum test

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.18       -
O2,24h    0.85       1.00

P value adjustment method: bonferroni
```

Usamos então a função `pairwise.wilcox.test()` e usamos também a correção de Bonferroni. Não há surpresas sobre os resultados.

Exercícios

1. Indique no gráfico que fizemos no primeiro exemplo os componentes das variabilidades totais dentro e entre os grupos. Faça com um lápis.

2. (a)Faça a ANOVA do banco `juul` e interprete o resultado. Apresente o código que você usou ou os cálculos feitos a mão. (b) Faça as comparações múltiplas necessárias entre todos os grupos. Use a correção de Bonferroni. Quantas comparações estaremos fazendo nesse caso? Quais as conclusões que você pode tirar dos resultados? Obs.: estabeleça os testes de hipóteses que você realizou.

3. Calcule passo-a-passo as comparações 2 a 2 para o banco `red.cell.folate` que usamos como exemplo na aula. Faça também a correção de Bonferroni. Por que uma delas não é exatamente 3 vezes a não corrigida. Dica: Não calcule a s^2_{conj} a mão. Use a saída da tabela de ANOVA, ou calcule com a ajuda de um software qualquer.

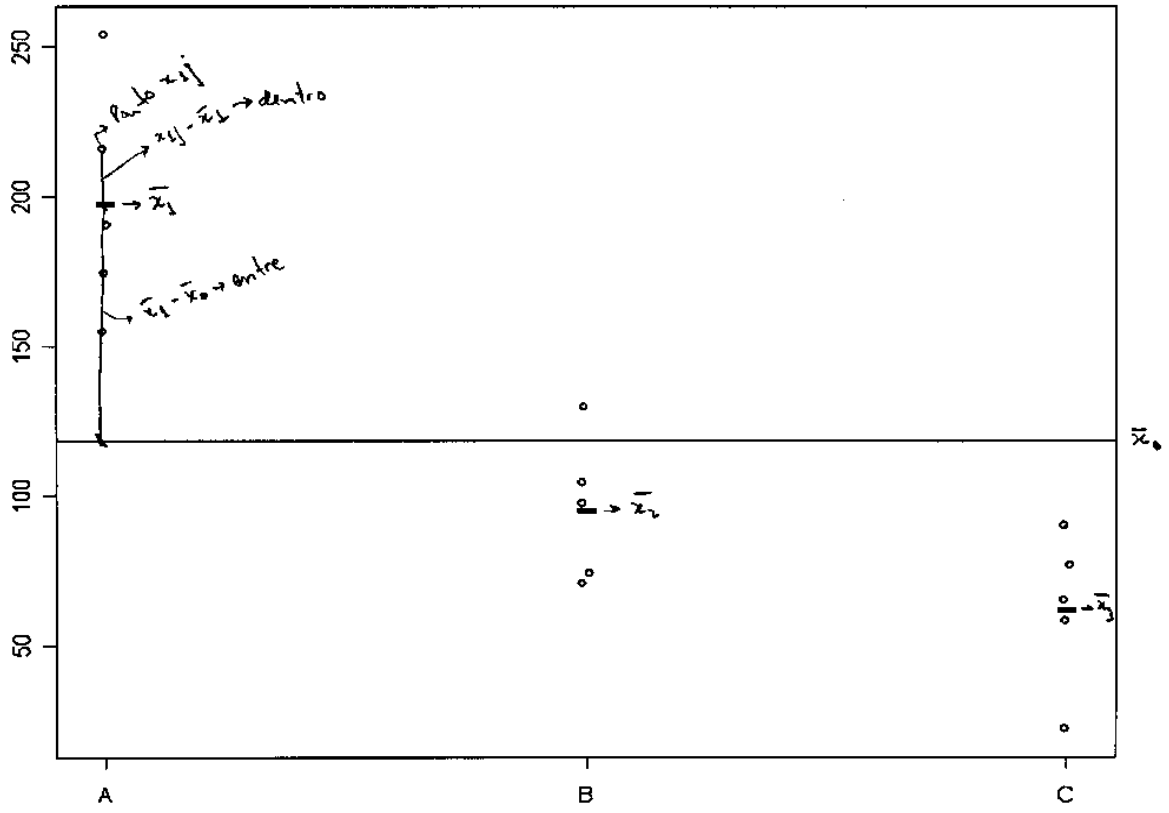
4. Exercício 6.1 do livro (página 127)

5. Repita o exercício acima usando testes não-paramétricos. Quais as suas conclusões?

Aula 7 - ANOVA
Livro: páginas 111 a 121

1. Indique no gráfico que fizemos no primeiro exemplo os componentes das variabilidades totais dentro e entre os grupos. Faça com um lápis.

Claro que nesse caso você pode escolher um ponto como exemplo, e já saberemos que o somatório de todas essas distâncias ao quadrado será a nossa variabilidade. Veja como seria um exemplo bem simples:



2. (a)Faça a ANOVA do banco `juul` e interprete o resultado. Apresente o código que você usou ou os cálculos feitos a mão. (b) Faça as comparações múltiplas necessárias entre todos os grupos. Use a correção de Bonferroni. Quantas comparações estaremos fazendo nesse caso? Quais as conclusões que você pode tirar dos resultados? Obs.: estabeleça os testes de hipóteses que você realizou.

Bem, o primeiro passo é identificar que variáveis do banco `juul` poderiam ser usadas para a aplicação da ANOVA. Isso não é lá muito difícil, já que a variável de interesse é contínua, a IGF-I e a única variável com mais de 2 categorias é a classificação de Tanner.

(a)

Nesse caso, o nosso teste de hipóteses será:

$H_0: \mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$

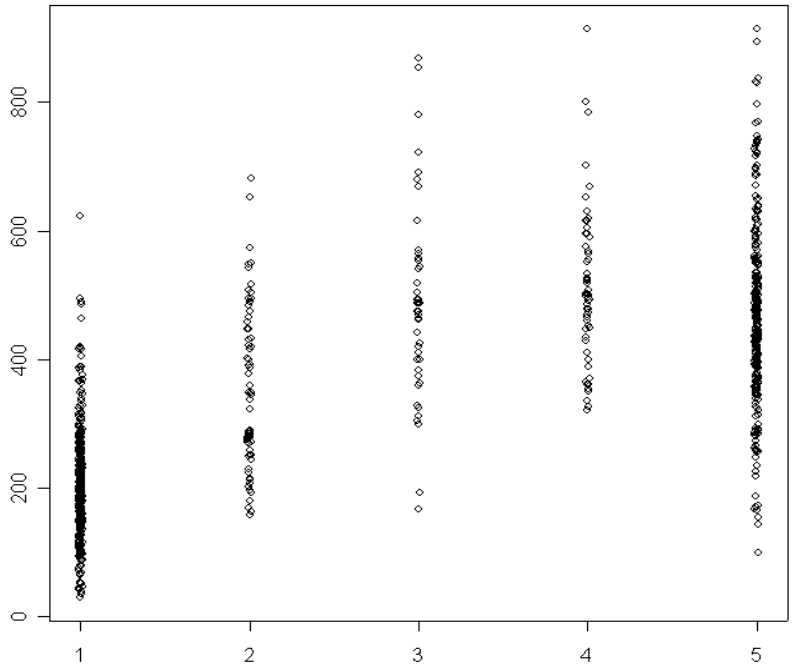
$H_1:$ Pelo menos um par de médias é diferente

Onde as médias se referem à média de concentração de IGF-I por cada umas das cinco

categorias da classificação de Tanner. Bem, vamos dar uma olhada nas nossas pressuposições:

Primeiro as variâncias devem ser homogêneas. Vamos ver. Primeiro vamos dar uma olhada no gráfico de distribuições dos pontos:

```
stripchart(igf1~tanner, vert=T, "jitter", jit=0.01, pch=21)
```



Não me parecem lá muito homogêneas não... Vamos aplicar o teste:

```
 > bartlett.test(igf1,tanner)

         Bartlett test for homogeneity of variances

 data:  igf1 and tanner
 Bartlett's K-squared = 55.6603, df = 4, p-value = 2.362e-11
```

É... nada homogêneas. Vamos ver os valores dos DPs:

```
 > by(igf1, tanner, sd, na.rm=T)
INDICES: 1
[1] 90.27237
------------------------------------------------------------
INDICES: 2
[1] 122.5933
------------------------------------------------------------
INDICES: 3
[1] 152.2866
------------------------------------------------------------
INDICES: 4
[1] 119.0959
------------------------------------------------------------
INDICES: 5
[1] 134.4187
```

São de fato bastante diferentes. Bem, teremos que relaxar essa pressuposição. Vamos à normalidade. Vamos dar uma olhada no teste de Shapiro-Wilk em cada um dos grupos para ver como estamos em termos de distribuição normal:

```
> by(igf1, tanner, shapiro.test)
INDICES: 1

        Shapiro-Wilk normality test

data:  data[x, ]
W = 0.9695, p-value = 3.764e-06

------------------------------------------------------------
INDICES: 2

        Shapiro-Wilk normality test

data:  data[x, ]
W = 0.9606, p-value = 0.02704

------------------------------------------------------------
INDICES: 3

        Shapiro-Wilk normality test

data:  data[x, ]
W = 0.9635, p-value = 0.1657

------------------------------------------------------------
INDICES: 4

        Shapiro-Wilk normality test

data:  data[x, ]
W = 0.9469, p-value = 0.01309

------------------------------------------------------------
INDICES: 5

        Shapiro-Wilk normality test

data:  data[x, ]
W = 0.9783, p-value = 0.0001284
```

Estamos mal. Só uma das categorias parece ser normalzinha... Sinto um cheirinho de teste não-paramétrico no ar...

Mas vamos fazer a ANOVA como pedido, mas relaxando a variância homogênea:

```
> oneway.test(igf1~tanner)

        One-way analysis of means (not assuming equal variances)

data:  igf1 and tanner
F = 258.5494, num df = 4.000, denom df = 154.677, p-value = < 2.2e-16
```

Foi rejeitado mesmo, sem dúvida nenhuma, né? Mas vamos ver o teste de Kriskal-Wallis:

```
 > kruskal.test(igf1~tanner)

        Kruskal-Wallis rank sum test

data:  igf1 by tanner
```

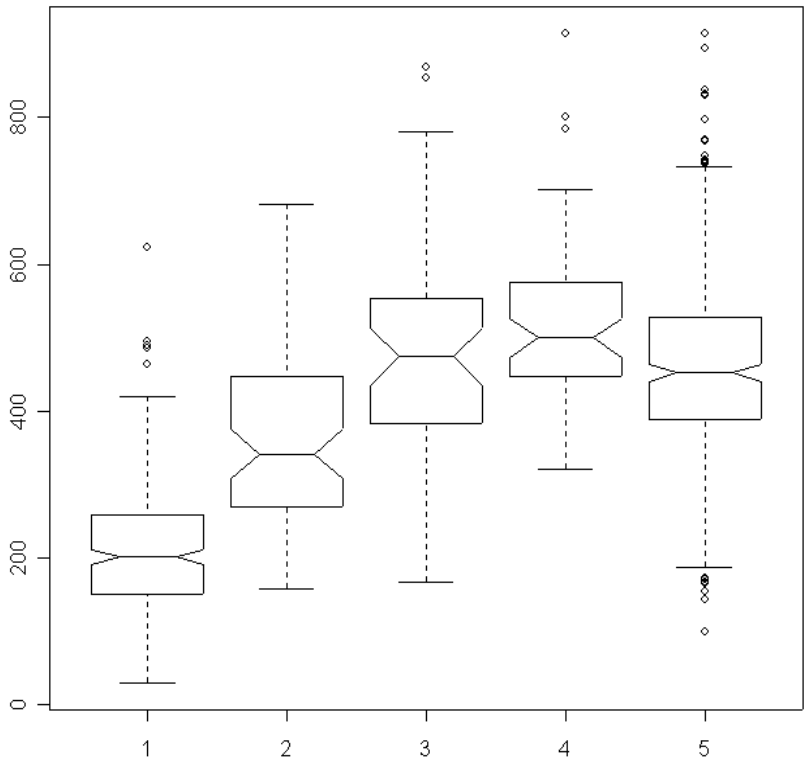
```
Kruskal-Wallis chi-squared = 462.3461, df = 4, p-value = < 2.2e-16
```

É, parece que rejeitamos mesmo a hipótese nula e concluimos que pelo menos um par dessas médias é diferente. Vamos ver quais...

(b)

Podemos começar com boxplots com indentações para termos uma idéia do que está acontecendo:

```
boxplot(igf1~tanner, notch=T)
```



Pelos gráficos, parece que os grupos 1 e 2 são diferentes de todos os outros (incluindo eles mesmos) e os grupos 3, 4 e 5 são similares (dúvida entre o 4 e o 5 talvez). Repare que esse gráfico permite também inspecionar a dispersão da variável por grupo, e que realmente parece diferente. Bem, vamos fazer então os testes 2 a 2. Vou poupar o leitor dos testes de hipóteses que faremos, pois são muitos... Repare que estou usando a correção de Bonferroni e que estou considerando as variâncias diferentes:

```
> pairwise.t.test(igf1,tanner, "bonferroni", pool.sd=F)

        Pairwise comparisons using t tests with non-pooled SD

data:  igf1 and tanner

  1       2       3     4
2 8.8e-14 -       -     -
3 6.2e-15 6.5e-05 -     -
4 < 2e-16 1.2e-10 1.000 -
```

```
5 < 2e-16 5.3e-09 1.000 0.075

P value adjustment method: bonferroni
```

Repare que os resultados corroboram o nosso método gráfico: os grupos 1 e 2 são diferentes de todos, inclusive entre si, os grupos 3, 4 e 5 são semelhantes, sendo que o 4 e 5 tiveram um resultado *borderline*.

Podemos, por via das dúvidas aplicar a versão não-paramétrica também:

```
 > pairwise.wilcox.test(igf1,tanner, "bonferroni")

        Pairwise comparisons using Wilcoxon rank sum test

data:  igf1 and tanner

  1       2       3     4
2 < 2e-16 -       -     -
3 < 2e-16 5.7e-05 -     -
4 < 2e-16 1.6e-09 1.000 -
5 < 2e-16 7.6e-09 1.000 0.052

P value adjustment method: bonferroni
```

Nada mudou, exceto o nosso *borderline* que ficou ainda mais limítrofe..

Para isso tudo fizemos 10 comparações. Confira:

```
> choose(5,2)
[1] 10
```

3. Calcule passo-a-passo as comparações 2 a 2 para o banco `red.cell.folate` que usamos como exemplo na aula. Faça também a correção de Bonferroni. Por que uma delas não é exatamente 3 vezes a não corrigida. Dica: Não calcule a s^2_{conj} a mão. Use a saída da tabela de ANOVA, ou calcule com a ajuda de um software qualquer.

Vamos então seguir a dica e calcular a tabela de ANOVA:

```
> anova(lm(folate~ventilation))
Analysis of Variance Table

Response: folate
            Df Sum Sq Mean Sq F value  Pr(>F)
ventilation  2  15516    7758  3.7113 0.04359 *
Residuals   19  39716    2090
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Então a nossa variância connjunta é de 2090. Vamos calcular as médias agora:

```
medias<-tapply(folate, ventilation, mean)
```

E agora vamos aplicar a fórmula que nós aprendemos:

```
> 2*pt(abs(medias[1]-medias[2])/sqrt(2090*((1/8)+(1/9))), 19,
lower.tail=F)
N2O+O2,24h
0.01391336
> 2*pt(abs(medias[1]-medias[3])/sqrt(2090*((1/8)+(1/5))), 19,
lower.tail=F)
N2O+O2,24h
```

```
 0.1547313
> 2*pt(abs(medias[2]-medias[3])/sqrt(2090*((1/9)+(1/5))), 19,
lower.tail=F)
N2O+O2,op
0.4084441
```

Não se preocupe com os títulos das saídas, são apenas por causa do `tapply()` que eu tinha aplicado antes.

Repare que esses p-valores são aqueles não corrigidos que vimos na aula:

```
> pairwise.t.test(folate,ventilation, p.adj="none")

        Pairwise comparisons using t tests with pooled SD

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.014      -
O2,24h    0.155      0.408

P value adjustment method: none
```

Para a correção, temos que multiplicar esses valores por 3. Vamos ver:

```
> 3*2*pt(abs(medias[1]-medias[2])/sqrt(2090*((1/8)+(1/9))), 19,
lower.tail=F)
N2O+O2,24h
0.04174008
> 3*2*pt(abs(medias[1]-medias[3])/sqrt(2090*((1/8)+(1/5))), 19,
lower.tail=F)
N2O+O2,24h
 0.4641939
> 3*2*pt(abs(medias[2]-medias[3])/sqrt(2090*((1/9)+(1/5))), 19,
lower.tail=F)
N2O+O2,op
 1.225332
```

Quase igual à saída da aula:

```
> pairwise.t.test(folate, ventilation, p.adj="bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  folate and ventilation

          N2O+O2,24h N2O+O2,op
N2O+O2,op 0.042      -
O2,24h    0.464      1.000
```

Exceto pelo último valor que calculamos ser 1.22 e que o R relata como sendo 1. Isso acontece obviamente porque o p-valor, como você já está cansado de saber é uma probabilidade, e portanto não pode ser maior que 1. Logo, quando se faz essa correção, faz-se na verdade o máximo entre a multiplicação e a unidade.

4. Exercício 6.1 do livro (página 127)

A parte mais difícil desse exercício é na verdade passar esta lista para um formato adequado para aplicar os testes. Trata-se de uma lista, então podemos combinar os seus elementos em um vetor de idades:

```
idades<-c(zelazo$active, zelazo$passive, zelazo$none, zelazo$ctr.8w)
```

Agora podemos criar um vetor de grupos (1, 2, 3 e 4) correspondendo ao tamanho de cada um dos componentes da lista, repetindo esse número:

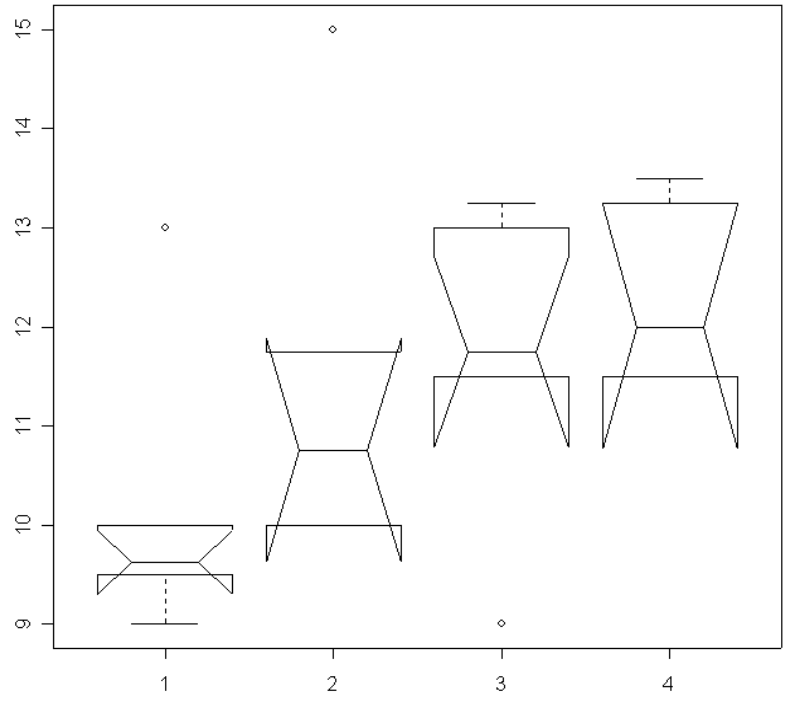
```
grupo<-c(rep(1, length(zelazo$active)), rep(2, length(zelazo$passive)),
rep(3, length(zelazo$none)), rep(4, length(zelazo$ctr.8w)))
```

Aplicando logo a ANOVA:

```
> anova(lm(idades~as.factor(grupo)))
Analysis of Variance Table

Response: idades
                 Df Sum Sq Mean Sq F value Pr(>F)
as.factor(grupo)  3 14.778   4.926  2.1422 0.1285
Residuals        19 43.690   2.299
```

Parece estar longe de rejeitar alguma coisa. Mas como o enunciado fala em usar testes *t*, inclusive combinando grupos, vamos dar uma olhada no *boxplot*:

```
boxplot(idades~grupo, notch=T)
```



Hummm. Números pequenos mesmo... Acho que vale no máximo juntar os grupos 3 e 4, não só porque são semelhantes, mas por ambos representarem controles:

```
grupo1<-c(rep(1, length(zelazo$active)), rep(2, length(zelazo$passive)),
rep(3, length(zelazo$none)), rep(3, length(zelazo$ctr.8w)))
```

E agora podemos aplicar novamente:

```
> anova(lm(idades~as.factor(grupo1)))
Analysis of Variance Table

Response: idades
                  Df Sum Sq Mean Sq F value  Pr(>F)
as.factor(grupo1)  2 13.655   6.827  3.0471 0.06996 .
Residuals         20 44.812   2.241
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Para esse podemos aplicar um teste pareado, só por fazer:

```
> pairwise.t.test(idades, grupo1, "bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  idades and grupo1

  1     2
2 0.491 -
3 0.068 1.000

P value adjustment method: bonferroni
```

Como esperado, nenhum significativo.

Ah, já ia me esquecendo: as variâncias são homogêneas e eu nem vi a normalidade porque vamos ter que fazer os teste não-paramétricos de qualquer maneira na próxima questão.

```
> bartlett.test(idades, grupo1)

        Bartlett test for homogeneity of variances

data:  idades and grupo1
Bartlett's K-squared = 1.035, df = 2, p-value = 0.596
```

5. Repita o exercício acima usando testes não-paramétricos. Quais as suas conclusões?

Bem, esses devem ser os testes mais recomendados de fato, especialmente porque os grupos são bem pequenos. Começando pelos 4 grupos:

```
> kruskal.test(idades~grupo)

        Kruskal-Wallis rank sum test

data:  idades by grupo
Kruskal-Wallis chi-squared = 6.8805, df = 3, p-value = 0.0758
```

Já obtivemos um resultado mais significativo. Vamos ver juntando os últimos:

```
> kruskal.test(idades~grupo1)

        Kruskal-Wallis rank sum test

data:  idades by grupo1
Kruskal-Wallis chi-squared = 6.2929, df = 2, p-value = 0.043
```

Hummm. Já fica até significativo. Vamos ver os testes 2 a 2:

```
> pairwise.wilcox.test(idades, grupo1, "bonferroni")

        Pairwise comparisons using Wilcoxon rank sum test

data:  idades and grupo1

  1    2
2 0.19 -
3 0.11 0.62

P value adjustment method: bonferroni
Warning messages:
1: Cannot compute exact p-value with ties in: wilcox.test.default(xi, xj,
...)
2: Cannot compute exact p-value with ties in: wilcox.test.default(xi, xj,
...)
3: Cannot compute exact p-value with ties in: wilcox.test.default(xi, xj,
...)
```

Bem, apesar desses valores não serem muito bons, pois o R não fez os testes exatos, o p-valor geral foi limítrofe e provavelmente a correção para o teste exato não rejeitaria a hipótese nula de qualquer maneira.

Módulo Estatística I no R

Autor: Antonio Guilherme Fonseca Pacheco

Pré-requisitos: Conhecimento prévio do ambiente R. Especificamente, o leitor deve estar familiarizado com os módulos "Básico", "Entrada e Saída de Dados" e também "Manuseando dados no R".

Bibliotecas necessárias: `ISwR`

Aula 8 - Regressão e correlação
Livro: páginas 95 a 110

Esta aula, assim como a anterior, nós vamos seguir o livro-texto mais de perto, e então nós vamos apenas comentar cada uma das seções de interesse, juntamente com o código utilizado em cada uma das passagens. Como não há muito o que modificar nesses capítulos, o aluno deverá ler os capítulos correspondentes e seguir a aula com os comentários.

Regressão linear Simples
- Resíduos e valores ajustados
- Tabela de ANOVA
- Predição e bandas de confiança

Correlação
- Correlação de Pearson
- Correlação de Spearman
- Correlação de Kendall

Exercícios

Regressão linear Simples

A idéia da regressão linear simples é identificar se uma variável (y) é linearmente dependente de uma outra variável (x) e também mensurar o quanto dependente ela é. Nos interessa saber então se os valores de y aumentam, diminuem ou não não se alteram quando os de x aumentam.

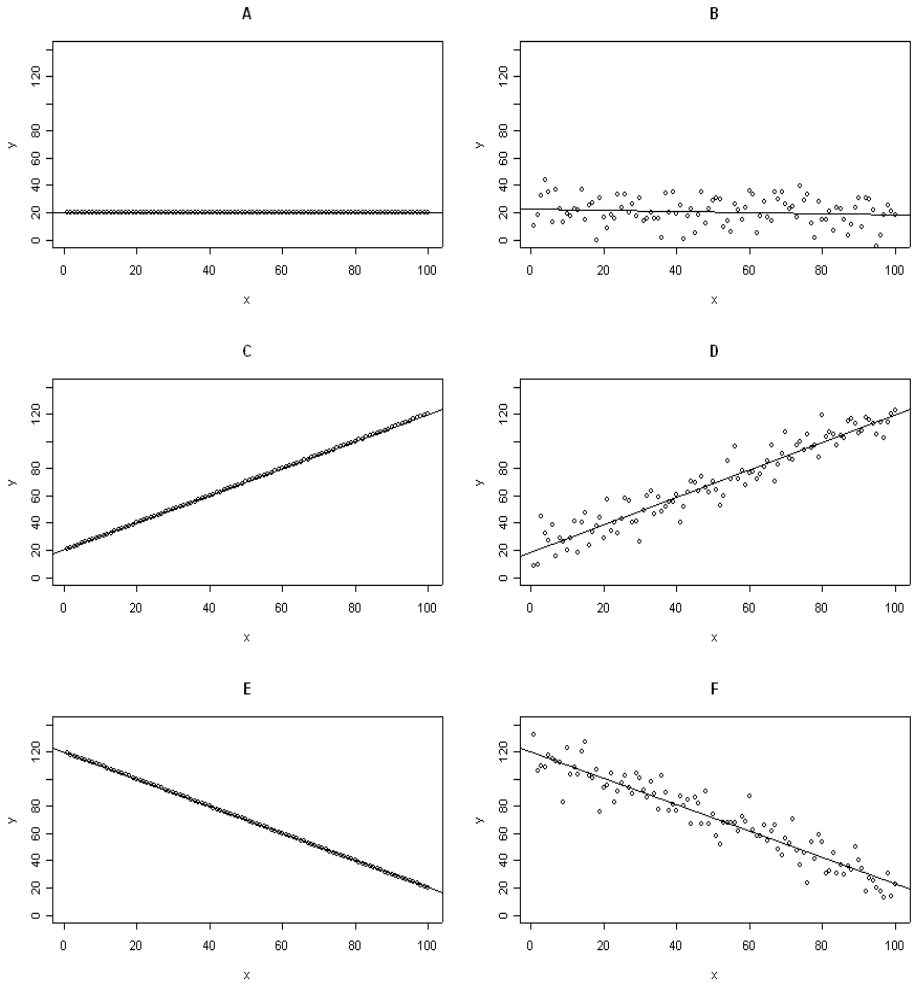
Na verdade isso nada mais é do que uma generalização do teste t que nós vimos para duas amostras ou a ANOVA que nós vimos para mais de duas amostras. Agora, estudaremos a associação de uma variável contínua com outra variável contínua. Mas como isso funciona?

Imagine a seguinte situação: temos uma variável x qualquer que varia de 1 a 100 e uma variável y que queremos ver como se comporta em relação a x. Repare que se trata de um par, ou seja, x e y são mensurados na mesma unidade (e.g. são duas variáveis do mesmo paciente). Vamos olhar para a figura abaixo. Que relação seria essa se estivéssemos falando dos gráficos A e B?

O que parece é que o valor de y é o mesmo, não importa o valor de x, ou seja, os valores de y são independentes dos valores de x. A diferença do gráfico A para o B é que o primeiro quase não apresenta variabilidade dos valores de y, enquanto que o segundo apresenta alguma variabilidade, que nós poderíamos nos referir como "erro". As outras duas duplas representam uma relação positiva entre x e y, ou seja, à medida que x aumenta, y também aumenta (C e D), e na seguinte é o contrário, ou seja a relação é negativa e à medida que x aumenta, y diminui (E e F). Também estão presentes exemplos com "erros" pequenos (C e E) e grandes (D e F).

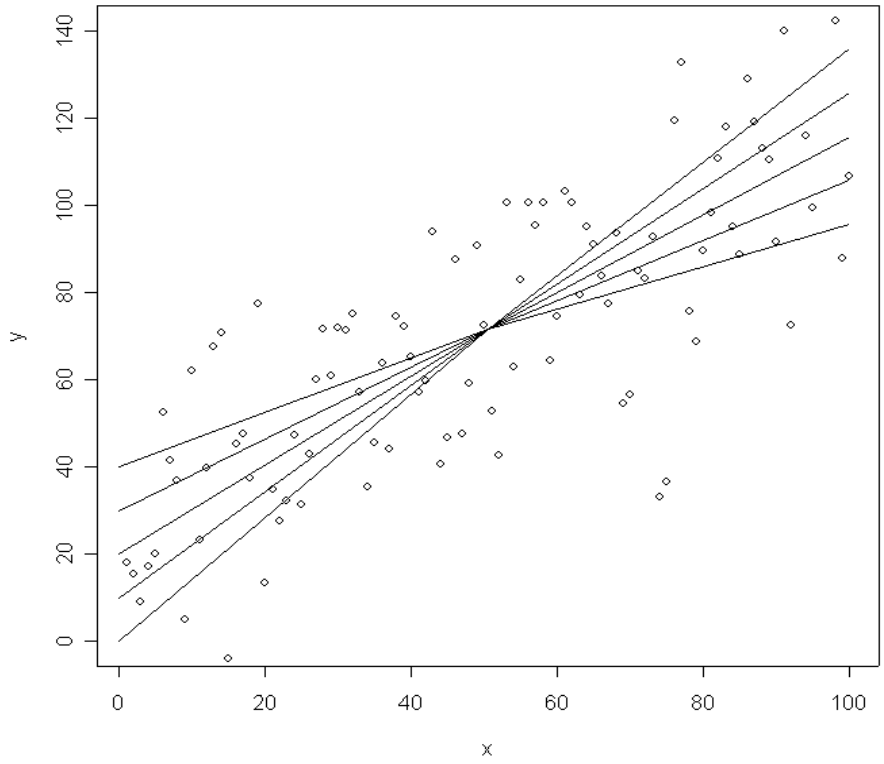
Mas e essa linha reta que aparece nos gráficos? Você vai dizer: Hummm, como é uma aula de regressão linear, deve ser a linha que corresponde a essa regressão, ou seja, a "melhor" linha que descreve a relação entre x e y.

Afinal de contas, que reta é essa? Por que justamente essa reta, calculada a partir de uma regressão linear é a “melhor”?



Bem, é aí que entram aqueles “erros” entre aspas mesmo. Vamos ver esse outro gráfico. Ele contém também uma nuvem de pontos. Estamos procurando uma reta que melhor explique essa relação entre as variáveis *x* e *y*. Então, qual dessas retas mostradas melhor respondem a essa questão? Todas elas descreveriam uma relação entre essas variáveis... Aí é que entra o nosso “erro”. O que se convenciona é que a melhor reta é aquela que minimiza o “erro”. Repare que é curioso, pois a realidade é o dado que temos, certo? Mas como assumimos que existe uma reta que descreve essa relação, o que se afasta dessa reta teórica, é chamado de erro.

Mas como é possível achar uma reta que minimize o erro? É um conceito parecido com o da variância. Digamos que a reta seja uma média (o que de fato é, você verá que ela é a média condicional da variável *y*, dado *x*) e que a variabilidade é medida entre os pontos e essa reta. Como no caso da variância, a soma dos afastamentos ao quadrado de todos esses pontos para essa reta seria uma medida dessa dispersão. A reta que proporcionar a mínima variabilidade é então considerada a melhor reta. É o famoso do método dos mínimos quadrados.



Uma dúvida inevitável que vem a seguir é: mas como é possível obter-se essa reta? Será que eu tenho que calcular esses afastamentos para todas as possíveis retas? Na verdade, não. É possível obter-se estimadores para esta reta, a partir de cálculos que fogem do escopo desta aula. Mas vamos só entender melhor. Primeiro, uma equação qualquer para essa reta (sem conhecer ainda) seria, como você já aprendeu em matemática: $y_i = \alpha + \beta x_i$, onde α é o ponto onde essa reta corta o eixo *y* e β é o coeficiente angular ou a tangente do ângulo que a reta faz com o eixo *x*. No caso da nossa reta para a regressão linear, podemos ainda incluir o nosso erro: $y_i = \alpha + \beta x_i + \epsilon_i$.

Esse erro no caso é o afastamento dos pontos para a reta calculada, certo? Parece muito com a idéia da ANOVA, não é mesmo? E é bastante parecido sim. Mais adiante vamos inclusive mostrar uma tabela de ANOVA para a regressão e vamos ver que graficamente essa questão dos erros poderá ser melhor visualizada.

Mas repare que para essa nossa equação da reta, podemos tirar o valor de ϵ_i :

$\epsilon_i = y_i - (\alpha + \beta x_i)$. E a partir daqui podemos calcular a soma dos quadrados dos erros, não é mesmo?

$\sum_{i=0}^{n} \epsilon_i^2 = \sum_{i=0}^{n} [\, y_i - (\alpha + \beta x_i)]^2$. A partir dessa equação é que chegamos nos estimadores de α e β :

$$\hat{\beta} = \frac{\sum_{i=0}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n} (x_i - \bar{x})^2} \quad \text{e} \quad \hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Então, o que estamos fazendo é estimar valores para esta melhor reta, que minimiza os erros. Acho que não há muita dúvida, não é?

Um outro conceito importante que deve ser mencionado, mas não provado nesta aula é o fato dessa reta ter uma outra conotação também. Ela representa na verdade a média da variável *y* condicional aos valores observados de *x*. Isso significa que o nosso modelo de regressão linear simples pode ser escrito da seguinte forma:

$$E(Y \mid X) = \alpha + \beta x_i$$

Na verdade isso pode ser percebido também nas equações que mostramos acima para os estimadores $\hat{\alpha}$ e $\hat{\beta}$. Repare que este último representa a relação entre a variação conjunta de *x* e *y* (no numerador) e a variância de *x* (no denominador). Note como isso nos remete à idéia de uma probabilidade condicional, sendo a probabilidade de *x* e *y* sobre a probabilidade de *x*, só que nesse caso estamos lidando com variabilidades e não probabilidades.

Bem, essa leitura complementa os conceitos da seção 5.1 do livro-texto. Vamos apenas colocar o código usado para facilitar a sua vida:

```
library(ISwR)
data(thuesen)
attach(thuesen)
lm(short.velocity~blood.glucose)
summary(lm(short.velocity~blood.glucose))
```

Vamos acrescentar mais dois detalhes que são discutidos no livro, só para não ficar muito solto. O primeiro é sobre erro-padrão residual, que o livro explica tratar-se do estimador de σ. Só para não confundir, esse é o desvio-padrão dos erros, ou seja, é a variabilidade de ϵ_i e ele é igual, na verdade à raiz quadrada da nossa expressão lá de cima para a variabilidade do erro dividido pelos seus graus de liberdade, que no caso da regressão simples será sempre *n-2*:

$$\hat{\sigma}^2 = \frac{\sum_{i=0}^{n} [y_i - (\alpha + \beta x_i)]^2}{n-2}$$

, que como já vimos pode ser escrita também da seguinte forma:

$$\hat{\sigma}^2 = \frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{n-2}$$

Claro que o livro refere-se à raiz quadrada da expressão acima. Isso vai aparecer de novo quando virmos ANOVA no contexto de regressão.

O outro detalhe, que por sinal também veremos depois é o R^2. Ele é conhecido como o coeficiente de regressão e denota o quanto da variabilidade total do modelo pode ser explicado pela(s) variável(is) incluída(s) no modelo. Ou seja, é uma medida também de *fitness,* ajuste do modelo. Isso ficará mais claro quando virmos a tabela de ANOVA.

Vamos ao gráfico poroposto:

```
plot(blood.glucose,short.velocity)
abline(lm(short.velocity~blood.glucose))
```

Por que o nome das variáveis têm que ser invertidos nesses códigos acima? Podemos também acrescentar uma média incondicional de *y* nesse gráfico, não é verdade?

```
abline(h=mean(short.velocity, na.rm=T))
```

Resíduos e valores ajustados

Para a seção 5.2 teremos o seguinte código (vou fazer comentários também):

```
lm.velo<-lm(short.velocity~blood.glucose)
fitted(lm.velo)
resid(lm.velo)
```

A partir dessa observação dos valores, segue-se uma discussão grande sobre valores faltantes no R (que pode ser bem chato na verdade). Vamos usar somente a última opção que ele sugere para nós:

```
options(na.action=na.exclude)
lm.velo<-lm(short.velocity~blood.glucose)
fitted(lm.velo)
```

Bem, agora podemos fazer o gráfico que é sugerido pelo livro para os valores ajustados, mas cuidado que tem uns peguinhas. Para fazer exatamente o que o livro faz, devemos fazer a seguinte seqüência:

```
plot(blood.glucose,short.velocity)
lines(blood.glucose,fitted(lm.velo))
segments(blood.glucose, fitted(lm.velo), blood.glucose, short.velocity)
```

Muito bem. Repare que os afastamentos que estão representados nesse gráfico são exatamente o erro do modelo, ou seja o somatório do quadrado desses segmentos, ou $\sum_{i=0}^{n} (y_i - \hat{y}_i)^2$ é que representam a nossa variabilidade.

Bom, para finalizar essa seção, o livro comenta muito superficialmente sobre inspeção dos resíduos. Isso é muito importante em qualquer modelo, para podermos avaliar se as pressuposições do modelo não são desrespeitadas, avaliando assim a adequação do modelo escolhido. A verificação mais básica de todas é estudar se os resíduos têm uma distribuição aproximadamente normal e com uma variabilidade constante ao longo dos valores ajustados. Para isso o livro sugere 2 gráficos:

```
plot(fitted(lm.velo), resid(lm.velo))
qqnorm(resid(lm.velo))
```

Mas faça um de cada vez, por favor... Você sugere algum teste para esta normalidade dos resíduos?

No primeiro gráfico, estamos na verdade procurando tendências nesses resíduos, o que poderia denotar não normalidade, heteroscedasticidade ou ambos.

Agora, é claro que o R tem umas facilidades também que vocês verão com mais detalhes quando virem modelos lineares em um curso futuro. Mas se nós pedirmos para o R fazer um *plot* de um objeto classe `lm`, ele vai fazer por *default* 4 gráficos. Veja:

```
par(mfrow=c(2,2))
plot(lm.velo)
par(mfrow=c(1,1))
```

Os dois primeiros (de cima) são exatamente os que acabamos de fazer, só que com os *outliers* marcados para nós. O de baixo à esquerda é parecido com o primeiro, só que os valores do erro são padronizados, para remover o efeito de possíveis assimetrias presentes na distribuição e facilitar a visualização de tendências. O último gráfico mostra o cálculo de uma medida de análise de influência de pontos extremos, cuja teoria foge do escopo dessa aula.

Um último comentário sobre diagnóstico do modelo é que uma outra pressuposição que você poderia estar interessado em testar é a autocorrelação dos erros (lembra? Os erros de um ponto são independentes dos erros em outros pontos). Podemos testar isso com um gráfico também, mas você vai precisar da biblioteca `ts` (de séries temporais):

```
library(ts)
acf(resid(lm.velo)[!is.na(resid(lm.velo))])
```

O gráfico mostra os limites de confiança em azul. Repare que para o ponto 0, a correlação é 1 (claro, pois isso significa a correlação do ponto com ele mesmo):

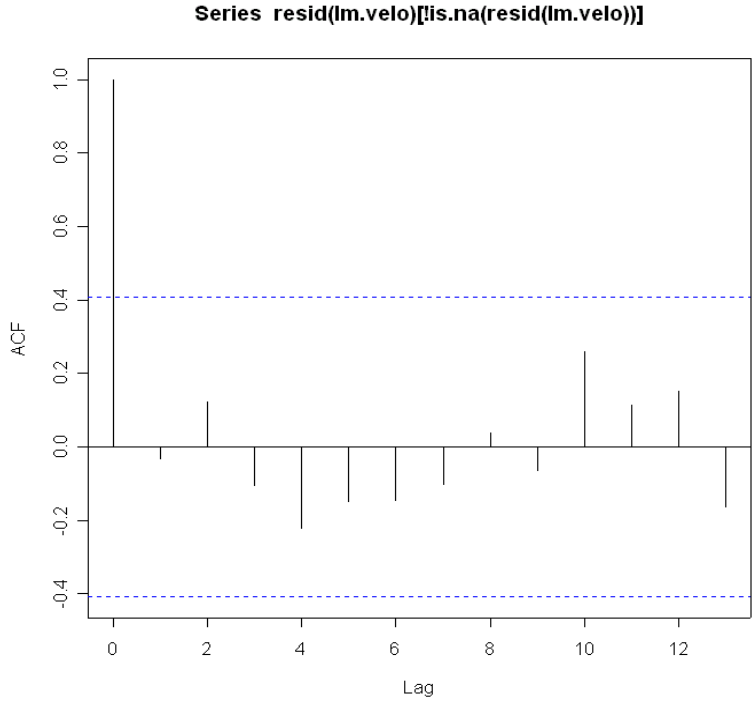


Tabela de ANOVA

Vamos partir agora para a tabela de ANOVA aplicada à regressão linear, que está no nosso livro no capítulo de ANOVA, seção 6.5, página 126. Toda a explicação vem do simples comando:

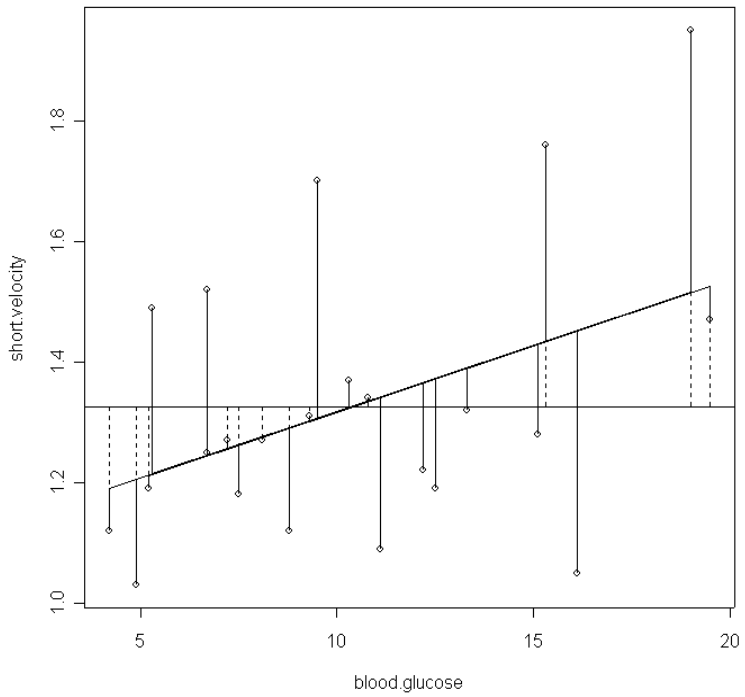
```
anova(lm.velo)
```

Os comentários são bastante completos, inclusive sobre o R^2, mas eu queria complementar algumas coisas. Primeiro uma visualização gráfica do que está acontecendo nessa tabela de ANOVA. Vamos fazer assim:

```
plot(blood.glucose,short.velocity)
segments(blood.glucose, rep(mean(short.velocity,
na.rm=T),length(blood.glucose)), blood.glucose, short.velocity, lty=2)
abline(h=mean(short.velocity, na.rm=T))
lines(blood.glucose,fitted(lm.velo))
segments(blood.glucose, fitted(lm.velo), blood.glucose, short.velocity)
```

Repare que nesse gráfico nós teremos as distâncias dos pontos até a média de *y* e também até a média condicional de *y*. Faça o seguinte: identifique na figura abaixo os componentes da

variabilidade total, i.e. a variabilidade do modelo e a variabilidade dos resíduos. Percebeu como é a mesma idéia da ANOVA?



A outra observação é sobre a relação entre os p-valores para a estatística *t* e a *F* que é comentada no texto. Essa relação entre essas distribuições existe mesmo e é a seguinte: Seja *X* uma distribuição *t* com ν graus de liberdade. Então X^2 terá uma distribuição $F_{1,\nu}$, ou seja uma *F* com 1 grau de liberdade no denominador e o mesmo número de graus de liberdade no denominador. Vamos conferir usando o próprio exemplo do livro. Antes tínhamos um *t* de 2.101 para o nosso beta, lembra? Vamos ver:

```
2*pt(2.101,21, lower.tail=F)
[1] 0.04789171
```

Confere com o valor. Vamos ver agora para uma F:

```
> pf(2.101^2,df1=1,df2=21, lower.tail=F)
[1] 0.04789171
```

Claro que agora não vamos multiplicar por 2, não é? O nosso quadrado já fez isso para nós. Conferiu?

Predição e bandas de confiança

Essa seção do livro é só para curiosidade e então não farei comentários a respeito.

Correlação

Vamos passar agora para os 3 tipos de correlação abordados no livro. Conforme destacado na parte inicial do texto, o coeficiente de correlação denota a associação entre duas variáveis aleatórias. Note que isso é um conceito diferente da regressão, onde estávamos estudando o

comportamento de uma variável aleatória (*y*) dados valores fixos de *x*. Comenta também da associação direta – ou positiva (quando o sinal é positivo) e da inversa – ou negativa (quando o sinal é negartivo.)

Correlação de Pearson

Bem ele já começa de cara com esse papo de elipse, reta... ihhhhhh, uma confusão danada... Não vamos abordar isso aqui, mas quem tiver curiosidade a respeito, grite.

Um comentário importante é você observar como a equação do coeficiente de correlação do livro se parece com o estimador do beta na regressão. Olhe só:

$$r = \frac{\sum_{i=0}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n} (x_i - \bar{x})^2 \sum_{i=0}^{n} (y_i - \bar{y})^2}} \quad \text{e} \quad \hat{\beta} = \frac{\sum_{i=0}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=0}^{n} (x_i - \bar{x})^2}$$

Notou a diferença? O numerador é igual – é a covariância de *x* e *y*. Mas o denominador de *r* não é a variância de *x*, mas sim o produto dos desvios-padrão de *x* e *y*. Você conseguiu enxergar isso? Ótimo, então... Isso tem alguma implicação quando fazemos testes de hipóteses para *r* ou para o beta?

Acho que o resto está bem explicado. O código usado vai abaixo:

```
cor(blood.glucose, short.velocity)
cor(blood.glucose, short.velocity, use="complete.obs")
cor(thuesen, use="complete.obs")
cor.test(blood.glucose, short.velocity)
```

Você desconfia por que ele menciona que o p-valor do teste para correlação é o mesmo para o valor obtido na regressão?

Uma observação importante é que não estamos tocando no caso de querermos testar a correlação de suas variáveis contra um valor fixo qualquer. Não vimos, porque isso não é muito usado e o método é algo complicado (bem, nem tanto...). Mas para quem for curioso, basta procurar em qualquer livro a transformação *z* de Fisher (de novo!!!).

Correlação de Spearman

Não acho que tenha algo a acrescentar aqui. A idéia é substituir o valor das observações pelos *ranks* e calcular normalmente o *r* que nós vimos anteriormente. O código usado:

```
cor.test(blood.glucose, short.velocity, method="spearman")
```

Correlação de Kendall

Também não tenho nada a acrescentar. O livro descreve também pouco, mas eu concordo que a interpretação é mais fácil... mais ou menos... O código:

```
cor.test(blood.glucose, short.velocity, method="kendall")
```

Exercícios

Do livro (página 110):

1. 5.1 - Não precisa calcular o IC 95%

2. 5.2
3. 5.3
4. 5.4

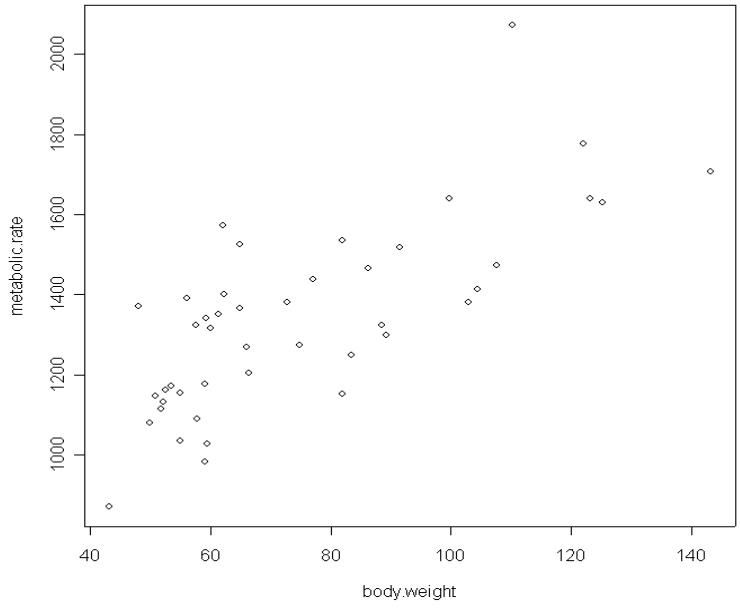
Exercícios - Respostas

Aula 8 - Regressão e correlação
Livro: páginas 95 a 110

Do livro (página 110):

1. 5.1 - Não precisa calcular o IC 95%

Segundo a ajuda do R esse banco contém dados sobre peso corporal e taxa de metabolismo de 44 mulheres. Pelo que entendi, o estudo visa saber se o peso corporal determina a taxa de metabolismo dessas mulheres. Vamos dar uma olhada nessa relação:



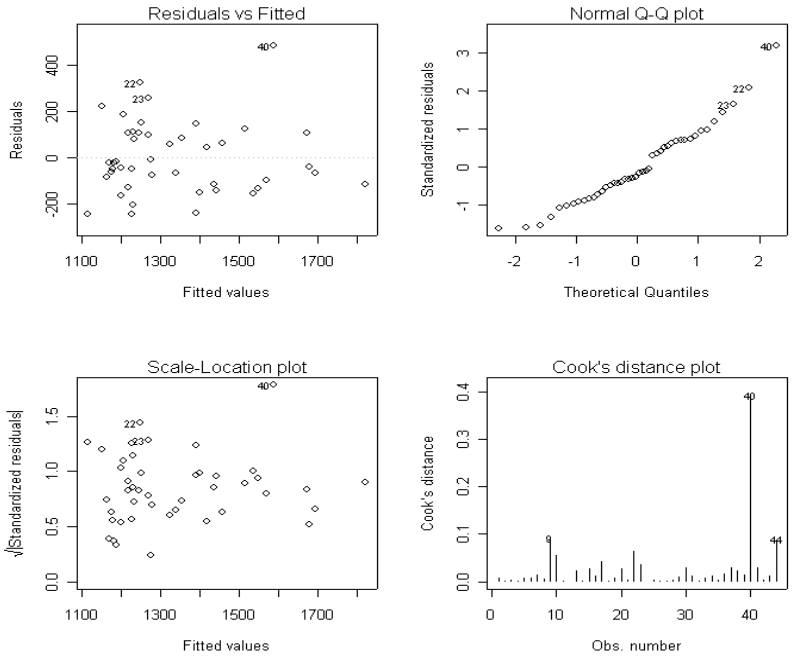
Até que parece bastante linear a relação...

O exercício é simples e pede simplesmente para ajustar um modelo de regressão linear a essas variáveis. Para isso, basta o código:

```
rmr.lm<-lm(metabolic.rate~body.weight,data=rmr)
```

Dê uma verificada na saída, e se quiser veja os gráficos para a análise dos resíduos primeiro:

```
par(mfrow=c(2,2))
plot(rmr.lm)
par(mfrow=c(1,1))
```



Nada mau, eu diria... Vamos ver o modelo mesmo:

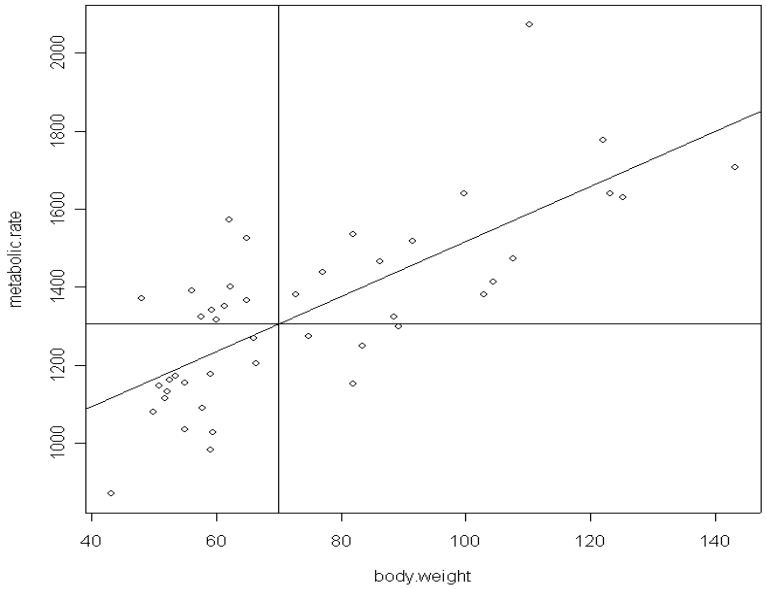
```
> summary(rmr.lm)

Call:
lm(formula = metabolic.rate ~ body.weight, data = rmr)

Residuals:
    Min      1Q  Median      3Q     Max
-245.74 -113.99  -32.05  104.96  484.81

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 811.2267    76.9755  10.539 2.29e-13 ***
body.weight   7.0595     0.9776   7.221 7.03e-09 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 157.9 on 42 degrees of freedom
Multiple R-Squared: 0.5539,     Adjusted R-squared: 0.5433
F-statistic: 52.15 on 1 and 42 DF,  p-value: 7.025e-09
```

Significativo, tudo nos conformes. Para encontrar o valor ajustado para um peso de 70 kg, basta usarmos a constante e o coeficiente preditos pelo modelo, no ponto $x = 70$:

```
> 811.2267+(7.0595*70)
[1] 1305.392
```

Tranqüilo, né? Vamos ver um gráfico desse ponto:

```
plot(metabolic.rate~body.weight,data=rmr)
abline(rmr.lm)
abline(v=70)
abline(h=1305.392)
```

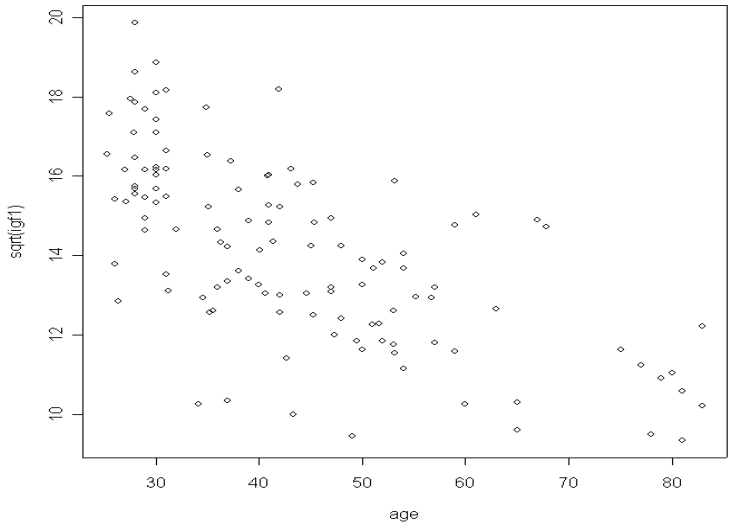


2. 5.2

Esse exercício também parece bem simples, com o nosso velho conhecido `juul`. A única dificuldade seria para alguns selecionar o subconjunto que ele pede (maiores de 25 anos). Vamos ver um gráfico dessas variáveis:

```
plot(sqrt(igf1)~age, data=juul, subset=age>25)
```

Repare que eu usei a opção `subset=age>25` para selecionar apenas parte dos dados. Além disso, a opção `data=juul` permite que você acesse um banco de dados com o nome das variáveis apenas, sem ter que anexar o objeto. A relação parece ser boa e inversa:



O modelo poderia ser ajustado da mesma forma:

```
juul.lm<-lm(sqrt(igf1)~age, data=juul, subset=age>25)
```

Faça a mesma verificação agora:

```
par(mfrow=c(2,2))
plot(juul.lm)
par(mfrow=c(1,1))
```

Agora, faça o seguinte: ajuste um modelo sem usar a raiz quadrada. O que você notou de diferente?

```
juul.lm1<-lm(igf1~age, data=juul, subset=age>25)
par(mfrow=c(2,2))
plot(juul.lm1)
par(mfrow=c(1,1))
```

Bem, o nosso modelo final aqui ficaria assim:

```
> summary(juul.lm)

Call:
lm(formula = sqrt(igf1) ~ age, data = juul, subset = age > 25)

Residuals:
    Min      1Q  Median      3Q     Max
-4.8642 -1.1661  0.1018  0.9450  4.1136

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 18.71025    0.49462  37.828   <2e-16 ***
age         -0.10533    0.01072  -9.829   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 1.741 on 120 degrees of freedom
Multiple R-Squared: 0.446,      Adjusted R-squared: 0.4414
F-statistic:  96.6 on 1 and 120 DF,  p-value: < 2.2e-16
```

A interpretação é que para cada acréscimo de uma unidade na idade do indivíduo, a raiz quadrada da IGF-I decresce 0.1053 unidade.

3. 5.3

Esse banco, segundo a ajuda do R descreve níveis de um determinado anticorpo em crianças de 3-15 anos em Gana. O exercício pede uma análise da relação entre os níveis desses anticorpos, transformados (log) e a idade. Para estudar essa relação podemos usar novamente a regressão linear.

```
ab.lm<-lm(log(ab)~age, data=malaria)
summary(ab.lm)

Call:
lm(formula = log(ab) ~ age, data = malaria)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0753 -1.0622  0.1181  1.1012  2.7335

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.83697    0.38021  10.092   <2e-16 ***
age          0.10350    0.03954   2.618   0.0103 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

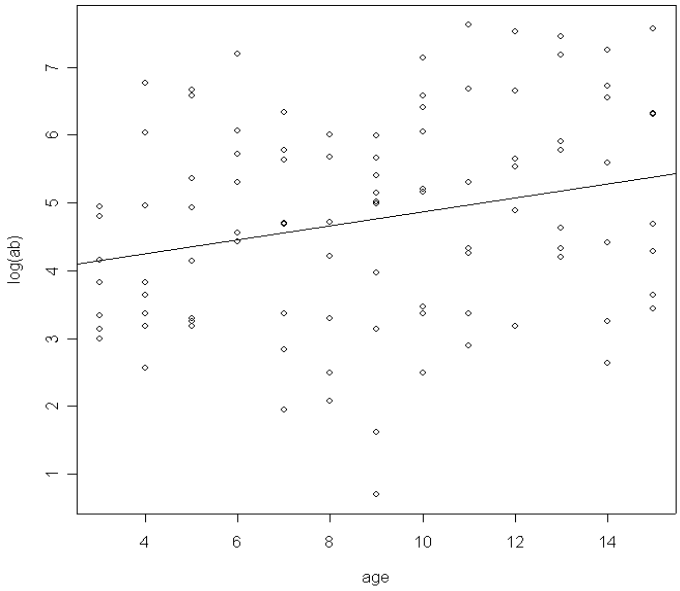
```
Residual standard error: 1.478 on 98 degrees of freedom
Multiple R-Squared: 0.06536,    Adjusted R-squared: 0.05582
F-statistic: 6.853 on 1 and 98 DF,  p-value: 0.01025
```

A relação parece significativa. Os gráficos para diagnóstico:

```
par(mfrow=c(2,2))
plot(ab.lm)
par(mfrow=c(1,1))
```

A disposição dos dados parece estranha mesmo... Vamos ver como é o gráfico com a linha de regressão:

```
plot(log(ab)~age, data=malaria)
abline(ab.lm)
```



A única peculiaridade que eu vejo é que a idade parece uma variável discreta, já que ela foi anotada como um número inteiro.

4. 5.4
   ANULADA