Formação Cientista de Dados

Vítor Wilher
Mestre em Economia
analisemacro.com.br

7 de março de 2019

Sumário

Lista de Tabelas

Lista de Figuras

Sobre o Curso

O curso de Formação Cientista de Dados será dividido em três grandes blocos. No primeiro, veremos uma introdução à *ciência de dados*, onde serão discutidos os grandes problemas pertinentes a essa área de estudo. Na segunda, faremos uma imersão no processo de análise de dados. Na última parte do curso, por fim, faremos uma introdução à *machine learning*.

Sobre a distribuição desse material

Material Complementar

Essa **apostila** conta com material complementar prático, que será disponibilizado ao longo das aulas do curso de Formação Cientista de Dados. O conjunto desses materiais faz parte do curso de **Formação Cientista de Dados**.

Sobre o Autor

Vítor Wilher é Bacharel e Mestre em Economia, pela Universidade Federal Fluminense, tendo se especializado na construção de modelos macroeconométricos, política monetária e análise da conjuntura macroeconômica doméstica e internacional. Tem, ademais, especialização em Data Science pela Johns Hopkins University. Sua dissertação de mestrado foi na área de política monetária, titulada "Clareza da Comunicação do Banco Central e Expectativas de Inflação: evidências para o Brasil", defendida perante banca composta pelos professores Gustavo H. B. Franco (PUC-RJ), Gabriel Montes Caldas (UFF), Carlos Enrique Guanziroli (UFF) e Luciano Vereda Oliveira (UFF). Já trabalhou em grandes empresas, nas áreas de telecomunicações, energia elétrica, consultoria financeira e consultoria macroeconômica. É o criador da Análise Macro, startup especializada em treinamento e consultoria em linguagens de programação voltadas para data analysis, sócio da MacroLab Consultoria, empresa especializada em cenários e previsões e fundador do hoje extinto Grupo de Estudos sobre Conjuntura Econômica (GECE-UFF). É também Visiting Professor da Universidade Veiga de Almeida, onde dá aulas nos cursos de MBA da instituição, Conselheiro do Instituto Millenium e um dos grandes entusiastas do uso do R no ensino. Leia os posts de Vítor Wilher no Blog da Análise Macro. Caso queira, mande um e-mail para ele: *vitorwilher@analisemacro.com.br*.

1 Primeiros Passos

Ao longo das aulas, utilizaremos o $\mathbf{R}$ como ferramenta principal, além de exemplos em $\mathbf{Python}$ nas exposições em sala de aula. Com efeito, de modo a nivelar a turma, essa primeira seção trata de uma introdução ao $\mathbf{R}$, dado que os alunos já conhecem $\mathbf{Python}$. Eu comecei a utilizar o $\mathbf{R}$ há alguns anos, influenciado por amigos. Minha introdução ao mundo dos programas estatísticos foi através do Eviews, ainda nos tempos da graduação, como provavelmente muitos dos matriculados nesse curso. Ainda que seja possível *programar* no Eviews e em outros pacotes estatísticos fechados (que precisam de licença), as vantagens do $\mathbf{R}$ são inúmeras, como comentarei mais à frente. Por ora, talvez seja necessário tecer algumas palavras sobre por que afinal é preciso aprender uma linguagem de programação.

Eu poderia falar que o mundo está mudando, que cada vez mais empregos e empresas têm exigido conhecimentos de programação. E isso de fato é verdade, o que por si só gera uma necessidade de saber programação. Mas, aqui entre nós, acho que é meio chato aprender algo por necessidade, não é mesmo?

Minha motivação para aprender a programar foi de fato outra. Eu estava um pouco cansado de *apertar botões* e fazer tarefas repetitivas em pacotes estatísticos como o Eviews, então parecia natural aprender uma forma de *automatizar* as coisas. Essa, afinal, era uma baita motivação para mim e talvez também seja para você. Mas por que o $\mathbf{R}$, você pode perguntar.

Essa é de fato uma boa pergunta. Por que não aprender a programar no próprio Eviews? E a resposta é bastante simples. Você já tentou encontrar alguma coisa de programação em Eviews na internet? E sobre $\mathbf{R}$? Pois é. Entre as inúmeras vantagens do $\mathbf{R}$, posso destacar:

- A existência de uma comunidade grande e bastante entusiasmada, que compartilha conhecimento todo o tempo;
- o $\mathbf{R}$ é gratuito, *open source*, de modo que você não precisa comprar licenças de software para instalá-lo;
- Tem inúmeras bibliotecas (pacotes) em estatística, *machine learning*, visualização,

importação e tratamento de dados;

- Possui uma linguagem estabelecida para *data analysis*;
- Ferramentas poderosas para comunicação dos resultados da sua pesquisa, seja em forma de um website ou em pdf.

Ao aprender $\mathbf{R}$, você conseguirá integrar as etapas de coleta, tratamento, análise e apresentação de dados em um único ambiente. Você vai esquecer ter de abrir o excel, algum pacote estatístico, depois o power point ou o word, depois um compilador de pdf para gerar seu relatório. Todas essas etapas serão feitas em um único ambiente. E essa talvez seja a grande motivação para você entrar de cabeça nesse mundo.

Certamente não será simples, tenho de lhe dizer. Haverá muitos momentos em que você pensará em desistir. Um erro inesperado em algum *script* poderá lhe tirar horas do seu tempo. No início, mais especificamente, qualquer pequeno problema pode parecer uma barreira intransponível. Nesses momentos, minha dica é *não se demorar muito nessas dificuldades*. Vá para outro problema ou mesmo descanse. Faça outras coisas. Depois volte mais tranquilo, procure nos canais de ajuda que colocaremos aqui e as coisas irão se acertar. Acredite: passado esse tempo inicial, com persistência, você certamente se beneficiará muito em ter aprendido uma linguagem de programação como o $\mathbf{R}$.

Nosso objetivo na Análise Macro, com nossos Cursos Aplicados de R é lhe proporcionar uma experência inovadora. Todos os nossos cursos apresentam sólida teoria, mas sempre recheada de exercícios e exemplos práticos. Nosso objetivo é trazer para o Brasil algo que já é bastante corriqueiro no resto do mundo: *você deve aprender fazendo*. Isso vai tornar o aprendizado mais interessante, mais divertido até.

Isso dito, vamos começar então? Nessa primeira seção, você verá alguns procedimentos básicos, sobre programas e um *overview* do $\mathbf{R}$. A partir da seção 02 começa o seu curso propriamente dito. Seja bem vindo a esse mundo e espero que não sai mais dele!

1.1 Instalando os programas

Antes de tudo, é preciso que você tenha os programas que utilizaremos ao longo das aulas instalados em seu computador. Serão três programas: $\mathbf{R}$, $\mathbf{RStudio}$ e $\mathbf{MikTex}$. Não se

preocupe, posto que são todos programas gratuitos e com *download* seguro. Desse modo, para que não tenhamos problemas, siga a sequência abaixo:

1. Baixe o $\mathbf{R}$ em http://cran-r.c3sl.ufpr.br/;
2. Baixe o $\mathbf{RStudio}$ em https://www.rstudio.com/products/rstudio/download/;
3. Baixe o $\mathbf{MikTex}$ *se você for usuário de Windows* em http://miktex.org/download;
4. Baixe o $\mathbf{MacTex}$ *se você for usuário de Mac* em http://www.tug.org/mactex/.

Para que você não tenha problemas no futuro, instale a versão do **MikTex** correspondente à versão do seu Windows. Isto é, se o seu Windows for 64 bits, instale a versão 64 bits do **MikTex**, se for 32 bits, instale a versão 32 bits do **MikTex**.

Por fim, alguns pacotes que utilizaremos fazem uso do **JAVA**. Assim, é importante que você o tenha instalado em sua máquina, **na versão correspondente do seu sistema operacional**, assim como o **MikTex**. Para instalar o JAVA, vá em Oracle.

1.2 Trabalhando com o R a partir do RStudio

Ao longo desse curso, nós não trabalharemos diretamente no $\mathbf{R}$. Ao invés disso, usaremos o $\mathbf{RStudio}$, uma interface mais amigável, que nos permite emular todos os códigos do $\mathbf{R}$, visualizar gráficos, ver o histórico de nossas operações, importar dados, criar scripts, etc. Com o $\mathbf{RStudio}$, poderemos otimizar o nosso curso, de maneira que o aluno terá mais facilidade para interagir com a linguagem.

A figura acima resume as quatro principais partes de uma tela do $\mathbf{RStudio}$. Na parte superior esquerda é onde ficará o nosso *editor de scripts*. Um *script* é uma sequência de comandos com um determinado objetivo. Por exemplo, você pode estar interessado em *construir um modelo univariado para fins de previsão do índice BOVESPA*. Para isso, terá de primeiro importar os dados do Ibovespa, bem como fazer uma análise descritiva inicial dos dados. Depois, com base nessa análise inicial, você terá de decidir entre alguns modelos univariados distintos. De posse da sua decisão, você enfim construirá um modelo de previsão para o índice BOVESPA. Essa sequência de *linhas de comando* pode ficar armazenada em um *script*, com extensão .R, podendo ser acessada posteriormente por

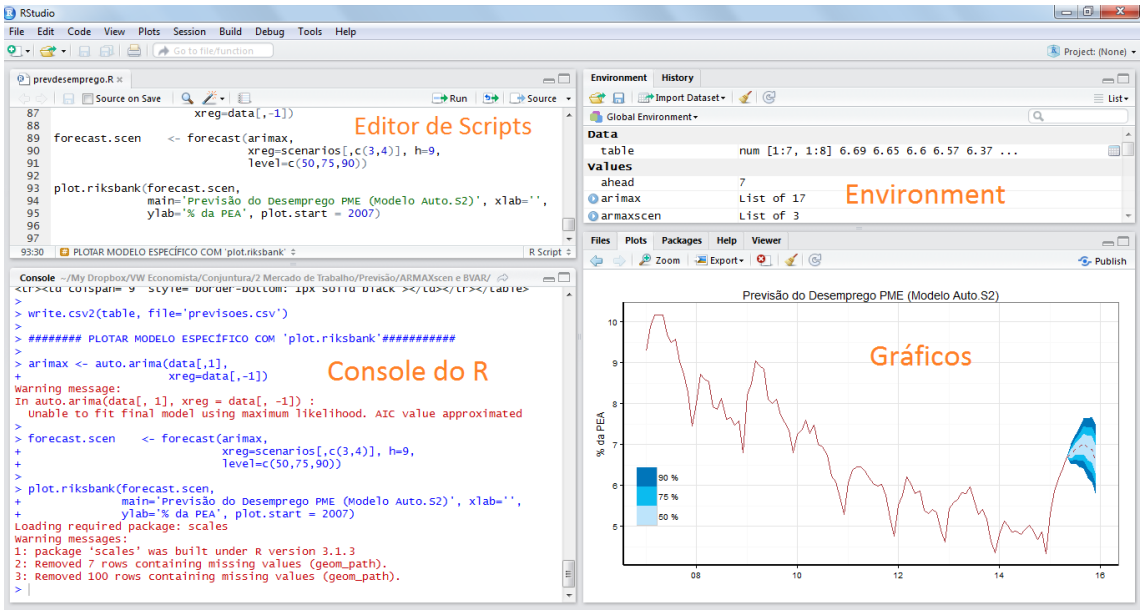


Figura 1: Ambiente do RStudio.

você mesmo ou compartilhada com outros colegas de trabalho. Para abrir um novo *script*, vá em *File*, *New File* e clique em *R Script*.

Na parte inferior esquerda, está o *console do R*, onde você poderá executar comandos rápidos, que não queira registrar no seu script, bem como será mostrados os *outputs* dos comandos que você executou no seu script.

Já na parte superior direita, está o *Environment*, onde ficam mostrados os objetos que você cria ao longo da sua seção no RStudio. Por fim, na parte inferior direita, ficarão os gráficos que você solicitar, bem como pacotes que você instalou, alguma ajuda sobre as funções e os arquivos disponíveis no seu diretório de trabalho.

1.3 Definindo seu diretório de trabalho

Agora que você já instalou os programas e já conhece um pouco do ambiente do **RStudio**, podemos começar a brincar um pouco. Para isso, antes de mais nada é preciso definir o seu *diretório de trabalho* ou a pasta onde ficará salvo o seu *script*. Uma vez definido, você poderá importar arquivos, colocar figuras no seu documento LaTeX, etc. Logo, dois comandos são importantes para isso. O primeiro é o **getwd**, para você ver o seu atual

working directory. O segundo é o **setwd**, para você *setar* o seu diretório de trabalho[1]

```
getwd()
```

[1] "C:/Users/Vítor Wilher/Dropbox/VW Economista/Análise Macro/01 - Cursos/Introdução à Data Science/Apostila"

```
setwd('C:/Users/Vítor Wilher/Dropbox/VW Economista')
```

Uma vez *setado* o seu diretório de trabalho, você poderá importar dados contidos naquela pasta. Assim, é um ponto importante realizar isso antes de qualquer coisa. Caso já tenha um *script* de R, por suposto, uma vez abrindo-o, o RStudio setará automaticamente o diretório onde o mesmo esteja.

1.4 Uma linguagem orientada a objetos

O $\mathbf{R}$ é uma linguagem orientada a objetos, de modo que o que você fará dentro do programa será, basicamente, manipulá-los. Seja lidando com objetos criados por terceiros, seja criando seus próprios objetos. As principais estruturas de dados dentro do $\mathbf{R}$ envolvem *vetores*, *matrizes*, *listas* e *data frames*. Abaixo colocamos um exemplo da estrutura mais simples do $\mathbf{R}$: um vetor que exprime a sequência de 1 a 10.

```
vetor <- c(1:10)

vetor
```

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

Ao longo do nosso curso, aprenderemos na prática a lidar com essas diferentes estruturas de dados. Em particular, aprenderemos a fazer o *subsetting* dessas estruturas, de modo a capturar subelementos das mesmas.

[1] Como você verá ao longo do nosso curso, as funções dentro do $\mathbf{R}$ são todas em inglês e bastante intuitivas, como *get something* ou *set something*.

1.5 Milhares de pacotes a sua disposição

O R é uma linguagem aberta, onde qualquer pessoa em qualquer parte do mundo pode dar a sua contribuição. Em geral, elas fazem isso através de *pacotes*, que são coleções de funções que fazem algum coisa dentro do R. Veremos muitos desses pacotes ao longo do nosso curso. A instalação de pacotes é feita primariamente pelo CRAN, através da seguinte função:

```
install.packages('fBasics')
```

O pacote é instalado corretamente quando aparece a mensagem *package 'fBasics' successfully unpacked and MD5 sums checked* no seu console. Ademais, no processo de instalação de um pacote, pode ser necessário instalar outros pacotes, chamados de *dependentes*, porque uma ou mais funções do pacote que você quer instalar fazem uso de funções de outros pacotes. Assim, **para que a instalação seja efetuada com sucesso, é preciso que todos os pacotes dependentes sejam instalados corretamente.** Por isso, procure verificar as mensagens no console de forma a verificar se o pacote foi instalado corretamente, como mostrado na figura 2.

```
> install.packages('fBasics')
Warning in install.packages :
  cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/src/contrib/PACKAGES.rds'
: HTTP status was '404 Not Found'
Installing package into 'C:/Users/Vitor Wilher/Documents/R/win-library/3.4'
(as 'lib' is unspecified)
Warning in install.packages :
  cannot open URL 'http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.4/P
ACKAGES.rds': HTTP status was '404 Not Found'
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.4/fBasics_3011.87.zi
p'
Content type 'application/zip' length 1559813 bytes (1.5 MB)
downloaded 1.5 MB

package 'fBasics' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Vitor Wilher\AppData\Local\Temp\Rtmp6rMrmM\downloaded_packages
```

Figura 2: Instalando pacotes.

Uma vez instalado, os seus pacotes ficam armazenados na pasta *library* da versão correspondente do seu R.[2] Você pode ver a lista de pacotes naturamente, diretamente no

[2]Toda vez que você instalar uma nova versão do R, uma dica é pegar os pacotes da sua pasta library e

RStudio, através da aba *Packages* no canto inferior direito.[3]

Uma outra forma muito comum de instalar pacotes é através do **GitHub**, uma plataforma bem bacana utilizada por desenvolvedores para compartilhar códigos. Ali ficam armazenados pacotes *em desenvolvimento*, que ainda não estão disponíveis no CRAN. Para instalar um pacote via GitHub, você deve ter instalado primeiro o pacote *devtools*. O código abaixo exemplifica com a instalação do pacote brasileiro BETS.[4]

```
install.packages('devtools')
require(devtools)
install_github("pedrocostaferreira/BETS")
```

No código acima, nos instalamos o pacote *devtools*, depois **carregamos** o mesmo com a função `require` - também pode ser utilizado a função `library` - e então utilizamos a função `install_github` para instalar um pacote armazenado no GitHub.

Às vezes pode ser necessário instalar uma versão antiga de um pacote, seja porque a versão atual tem algum *bug* - acontece! - seja porque ela é incompatível com o pacote que queremos usar. Para instalar versões antigas, você tem duas opções. Uma é subir o arquivo fonte zipado do pacote diretamente no RStudio, na opção *Tools* e *Install Packages*. Outra é utilizar a função `install_version` do pacote *devtools*, como abaixo.

```
install_version("DBI", version = "0.5", repos = "http://cran.us.r-project.org")
```

É muito comum os alunos receberem o erro de *caminho inválido* para a biblioteca de pacotes. Nesse caso, você pode especificar o caminho da biblioteca com o código abaixo.

```
library(withr) # É instalado e carregado junto com o devtools
withr::with_libpaths(new = "C:/Program Files/R/R-3.4.1/library", install_github("Stat
```

Por fim, vale ressaltar que todo pacote disponível no CRAN tem uma página com suas informações. Veja o exemplo do *devtools* aqui. Lá você pode ver um resumo do pacote, um pdf com as principais funções, versões antigas, etc.

---

copiar os mesmos para a pasta library da nova versão, de modo a não ter que instalar tudo novamente.

[3] Uma lista dos pacotes disponíveis pode ser encontrada aqui.

[4] Para saber mais sobre o BETS, vá em https://github.com/pedrocostaferreira/BETS.

1.6 Obtendo ajuda

Uma parte importante de aprender uma nova linguagem é saber conseguir resolver os pepinos que irão surgir ao longo do caminho. E, acredite: eles serão muitos! Mas não se desespere. Como há uma comunidade incrível trabalhando com **R**, existem inúmeros sites, blogs, tutoriais, apostilas, etc, que podem lhe ajudar. No próprio ambiente do **RStudio**, você pode invocar ajuda com os comandos abaixo.[5]

```
help.start() # Você terá acesso à página de ajuda do R.
help("read.csv") # Uma ajuda sobre a função 'read.csv'
?read.csv # A mesma coisa do comando acima.
```

Caso continue com dúvidas, entretanto, nada melhor do que o bom e velho Google. Recomendo que você jogue sempre sua dúvida lá, antes de qualquer coisa. Uma dica: jogue ela em inglês e verá logo na primeira página um monte de gente com o mesmo problema que você! Em particular, um fórum bastante conhecido é o Stackoverflow, onde pessoas mais experientes procuram ajudar os mais novos. Caso sua dúvida não seja resolvível via google, considere jogá-la nesse fórum. A versão em português do fórum está disponível em https://pt.stackoverflow.com/questions/tagged/r.

Outra coisa que você deve fazer a partir de agora é acompanhar blogs que falam de **R**. Modéstia ao largo, não se esqueça de acompanhar o meu próprio site www.analisemacro.com.br/blog. Lá faço alguns exercícios interessantes de como usar o **R** para resolver nossos problemas diários de análise de dados. No Brasil, há ainda os ótimos blogs do Cláudio Shikida e do Carlos Cinelli, que estão há bastante tempo nessa batalha também. No exterior, há uma infinidade de blogs reunidos no famoso R-bloggers.

[5] Observe que utilizamos o 'jogo da velha' # para colocar um comentário após a função. Isso é extremamente útil para que você lembre seus passos em um script no **R**, bem como no momento que você compartilhar seu script com outra pessoa.

1.7 Produzindo relatórios

1.7.1 Ei, você já instalou o MikTex/MacTex e o knitr?

Agora que você já conseguiu instalar e carregar pacotes no $\mathbf{R}$, vamos partir para um outro assunto. Uma das coisas mais legais do $\mathbf{R}$ é poder esquecer que o Word e o Power Point existem! Em um único ambiente você pode coletar, tratar, analisar e ... **apresentar** seus dados! Toda essa apostila, a propósito, é feita diretamente no $\mathbf{RStudio}$, dada a integração com o LaTeX. Para que isso seja possível, precisamos antes preparar o nosso ambiente. Siga os passos abaixo, de modo a fazer a integração entre o $\mathbf{R}$ e o LaTeX.[6]

1. Instale o pacote **knitr** com a `função` *install.packages*;
2. Vá em Tools, Global Options, Sweave e mude a opção *Weave Rnw files using*: to **knitr**;
3. Pronto, agora você está pronto para criar códigos LaTeX.

1.7.2 Tudo se resume a scripts

Uma vez que esteja tudo corretamente instalado, você deve abrir o $\mathbf{RStudio}$. Como vimos anteriormente, ao longo do nosso curso trabalharemos com *scripts*, sequências de comandos que têm por objetivo lidar com algum dado. Nesse caso, trabalharemos com extensões .R, enquanto no caso de produzir um documento, trabalharemos com extensões .Rnw. Criaremos muitos scripts para lidar com dados a partir da próxima seção - principalmente nas listas de exercícios -, de modo que por hora vamos nos concentrar nos scripts que geram documentos.

Para gerar um script que cria um documento, você deve ir em *File*, *New File* e clicar em *R Sweave*. Um *script* se abrirá com os seguintes comandos em LaTeX:

```
\documentclass{article}
```

[6] Caso queira conhecer desde já o ambiente do LaTeX, sugiro uma boa apostila do pessoal do PET de Telecomunicações da UFF aqui.

```
\begin{document}

\end{document}
```

Na primeira linha, nós definimos a classe do documento, pode ser um artigo, livro, apresentação, etc. Entre `\documentclass{article}` e `\begin{document}`, temos o *preâmbulo* do nosso documento, onde podemos listar pacotes que iremos utilizar, bem como outras definições de estilo do documento. Você irá escrever o seu documento entre `\begin{document}` e `\end{document}`, onde estará o *corpo* propriamente dito de tudo o que você fará. Uma referência para que você compreenda um documento LaTeX, desde a concepção do documento até diferentes comandos e ambientes que você pode utilizar, pode ser obtida aqui. Nessa nossa seção, tudo o que faremos é *facilitar* a sua vida, mostrando um exemplo simples de como gerar um relatório e uma apresentação, com códigos de R dentro.

1.7.3 Criando o seu primeiro documento LaTeX

Abra um arquivo .Rnw no RStudio seguindo as instruções da subseção anterior. Uma vez lá, coloque a sequência abaixo:

```
\documentclass[12pt, a4paper]{article}

\usepackage[T1]{fontenc}

\usepackage[portuguese]{babel}

\begin{document}

\begin{center}

\LARGE{Meu primeiro arquivo \LaTeX}

\end{center}

\subsection{Seção 1}

\subsubsection{Subseção 1.1}
```

Esse é o meu primeiro arquivo LaTeX.

```
\end{document}
```

Uma vez que tenha colocado essa sequência, clique em *Compile PDF* no cabeçalho do *script*. Caso não tenha salvo, o RStudio pedirá para salvar o arquivo e depois gerará o seu pdf. Caso você tenha conseguido gerar, é porque está tudo certo com os programas que recomendamos você instalar. Agora, vamos melhorar isso e colocar um código de R no seu documento LaTeX?

1.7.4 Adicionando códigos de R ao seu documento LaTeX

Para isso, precisamos utilizar o pacote **knitr**. É ele que vai emular códigos de R no seu documento. Você poderá controlar se quer mostrar os seus códigos ou se quer apenas o resultado dele, poderá controlar o espaço das figuras e inúmeras outras facilidades. Tudo isso é feito nos *chunks*, como o exemplo simples abaixo.

```
<<"chunk_example", eval=FALSE>>=
@
```

Ao longo das nossas listas de exercícios, vamos explorar algumas opções de *chunks*, mas você já pode conferir algumas aqui. Para adicionar um código de R no seu documento LaTeX, copie o *chunk* abaixo e coloque no arquivo que você acabou de criar acima.

```
<<"exemplo01", eval=T, echo=T, results='hide'>>=
@
```

Nessa área cinza, antes do @, adicione os seguintes comandos:

```
vetor <- 1:10
vetor
```

E agora, clique em *Compile PDF*. Você deverá ter gerado um PDF que mostra o código acima. Para mostrar o resultado do código, isto é, a sequência de 1 a 10, basta que você substitua *hide* por *asis* em `results`.

As listas de exercício do nosso curso são feitas nesse estilo. Procure utilizar o arquivo .Rnw da lista para respondê-la e enviar para mim - caso você seja aluno dos planos intermediário e premium. Isso vai facilitar muito a nossa interação.

1.7.5 Nosso curso já começou...

Pode não parecer, mas nosso curso já começou. Nossa ideia é mostrar para você como lidar com dados de forma real. Nas próximas semanas, tudo o que você fará será basicamente praticar, sempre. Mesmo que haja alguma teoria envolvida, em todas as seções, você será obrigado a fazer alguma coisa prática. Nessa seção não é diferente, você já tem um monte de coisa para fazer!

Senão, vejamos. Você já pode dar uma boa olhada naquela apostila de LaTeXque referenciamos. Você também já pode dar uma boa olhada nas opções de *chunk* no site do criador do pacote **knitr**.

A próxima seção, por suposto, é a primeira seção do seu curso propriamente dito. Bem vindo e mãos à obra!

2 O que é Data Science?

O avanço da informática e das telecomunicações possibilitou o armazenamento e a distribuição de conjuntos de dados cada vez mais complexos. Lidar com essas bases de dados exigiu a sistematização de diversas técnicas de coleta, tratamento, análise e apresentação de dados.

Essa sistematização de técnicas deu origem ao que hoje chamamos de **data science**, cujo objetivo principal é extrair informações úteis de conjuntos de dados aparentemente confusos. Abaixo, listamos algumas aplicações interessantes:

- Identificar mensagens indesejáveis em um e-mail (spam);
- Segmentação do comportamento de consumidores para propagandas direcionadas;
- Redução de fraudes em transações de cartão de crédito;
- Predição de eleições;
- Otimização do uso de energia em casas ou prédios;
- etc, etc, etc...

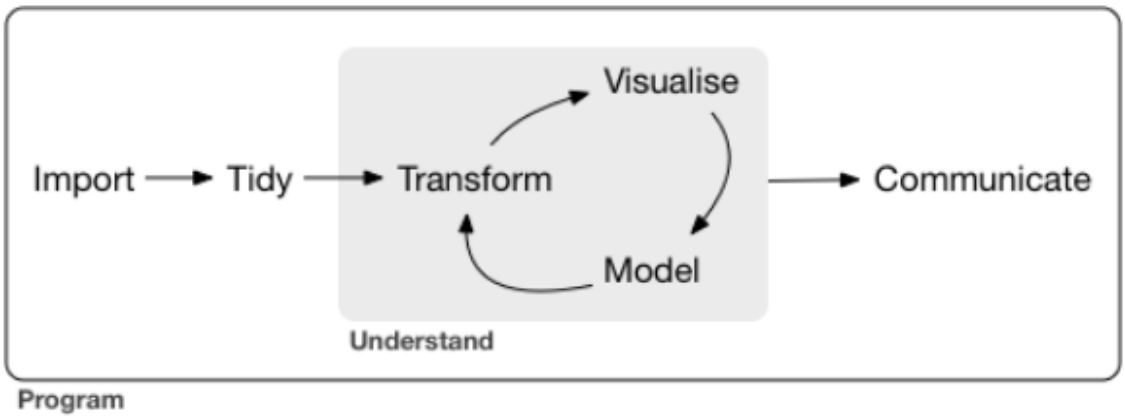


Figura 3: O processo de compreensão dos dados (Grolemund and Wickham (2017))

Ao coletar dados, introduzimos em uma plataforma de Análise de Dados (como o R) informações coletadas no mundo real. Seja vindas de base de dados prontas ou adquiridas minerando dados abertos em sites. Depois tratamos as informações para que sejam devidamente legíveis para um computador e bem formatadas para nosso próprio entendimento.

Então começamos a análise propriamente dita. Visualizamos os dados, procurando perguntas interessantes e padrões, depois avaliamos nossas intuições com modelos estatísticos. Por fim, comunicamos nossos achados - afinal de pouco importa achar algo somente para si.

2.1 Coleta e tratamento

Dados podem estar dispostos em diferentes formatos:

- Excel;
- XML;
- JSON;
- txt;
- HTML;
- MySQL;
- Formatos proprietários (Weka, Stata, Minitab, Octave, SPSS, SAS, etc).

Dados precisam ser tratados:

- Limpeza de dados;
- Tratamento de *missing values*;
- Construção de números índices;
- Deflacionar valores correntes;
- Obtenção de taxas de crescimento, a partir de comparações mensais, interanuais, acumuladas em 12 meses, etc;
- Tratando tendências;
- Dessazonalização;
- Obtendo subconjuntos (*subsetting*) relevantes;
- Classificando dados de acordo com algum critério;

- Transformando dados de acordo com alguma operação.

2.2 Exploração de dados

A exploração de dados é a arte de analisar seus dados, gerando hipóteses rapidamente, testando-os rapidamente, repetindo-os várias vezes. O objetivo da exploração de dados é gerar muitos leads promissores que você poderá explorar mais tarde com mais profundidade.

2.3 Modelagem

O objetivo da modelagem é capturar a essência de um conjunto de dados. Alguns exemplos de modelos que podem ser utilizados para isso:

- Modelos ARIMA;
- Regressão linear;
- Árvores de regressão;
- Neural Network;
- Support Vector Machine;
- Naive Bayes;
- etc, etc, etc.

2.4 Comunicação

Os resultados encontrados devem ser compartilhados com gestores, clientes ou colaboradores. Para os primeiros grupos, não se faz necessário mostrar o `código` utilizado, enquanto para o segundo isso deve ser uma característica importante. Por isso, é preciso encontrar uma plataforma que consiga integrar as quatro etapas da análise de dados, gerando como `produto` um documento reprodutíve, que unam **código** e **texto**. Seja o código visível ou não para a ponta final.

2.5 Tipos de Cientistas de Dados

Com a difusão da área de *data science*, é muito comum que as empresas estejam atrás de profissionais multidisciplinares, que consigam cumprir aquelas quatro etapas do processo. Como começamos a ver, contudo, essas quatro etapas envolvem diversos conhecimentos que são difíceis de serem encontrados em um único profissional. Não é que eles não existam, de fato, há alguns *unicórnios* por aí, mas são raros de serem encontrados.

De maneira geral, há *tipos* de cientistas de dados. Podemos diferenciá-lo da seguinte forma;

- *Analistas de BI*: raramente codificam (podem até utilizar GUIs para acessar bancos de dados, para que nem sequer escrevam consultas SQL - a ferramenta faz isso para eles; no entanto, eles precisam entender o esquema do banco de dados.), são os responsáveis por definir métricas e trabalhar com o gerenciamento para identificar fontes de dados ou para criar dados. Eles também trabalham na criação de *dashboards* de dados com vários usuários finais em mente, desde segurança, finanças, vendas, marketing até executivos. Muitos têm um diploma de MBA;
- *Engenheiros de Dados*: obtêm os requisitos desses analistas de BI para configurar os pipelines de dados e têm o fluxo de dados em toda a empresa e fora dele, com pequenos pedaços (normalmente dados resumidos) terminando em vários laptops de funcionários para análise ou relatório. Eles trabalham com administradores de sistema para configurar o acesso a dados, personalizado para cada tipo de usuário. Eles estão familiarizados com o data warehousing, os diferentes tipos de infraestrutura em nuvem (interna, externa, híbrida) e sobre como otimizar as transferências e o armazenamento de dados, equilibrando a velocidade com o custo e a segurança. Eles estão muito familiarizados com o funcionamento da Internet, bem como com a integração e padronização de dados. Eles são bons em programar e implantar sistemas projetados pelo terceiro tipo de cientistas de dados, descrito abaixo. Às vezes, particularmente para funções seniores, eles são chamados de arquitetos de dados;
- *ML Data Scientists*: Os cientistas de dados de aprendizado de máquina projetam e monitoram sistemas preditivos e de pontuação, têm um grau avançado, são especialistas em todos os tipos de dados (grandes, pequenos, em tempo real, não-

estruturados etc.) Eles executam muitos algoritmos, testes, ajustes e manutenção. Eles sabem como selecionar / comparar ferramentas e fornecedores, e como decidir entre o aprendizado de máquina caseiro ou ferramentas (fornecedor ou código aberto). Eles geralmente desenvolvem protótipos ou provas de conceitos, que acabam sendo implementados no modo de produção por engenheiros de dados. Suas linguagens de programação de escolha são Python e R;

- *Analistas de Dados*: Os analistas de dados são cientistas juniores de dados que fazem muita análise de números, limpeza de dados e trabalho em análises únicas e geralmente projetos de curto prazo. Eles interagem e suportam cientistas de dados de BI ou ML. Eles às vezes usam técnicas de modelagem estatística mais avançadas.

2.6 Conhecimentos necessários

Uma pergunta muito frequente associada à data science é justamente o que é preciso saber para se tornar um cientista de dados. Como vimos acima, nem todos os profissionais que trabalham com ciência de dados são iguais. Logo, não há uma cesta de conhecimento única para se trabalhar na área. Analistas de BI, por exemplo, vão estar muito mais preocupados com a criação e apresentação de indicadores de desempenho; enquanto engenheiros de dados estarão mais frequentemente envolvidos com a infraestrutura necessária para armazenar e distribuir conjuntos de dados entre diferentes perfis de usuários.

De maneira geral, cientistas de dados são altamente qualificados, tendo mestrado ou doutorado em áreas como matemática, estatística, economia, ciência da computação, etc, de modo que disciplinas como cálculo, estatística descritiva e inferência estatística são conhecidas. Ademais, têm conhecimento de uma ou mais linguagens de programação, como R e Python. Outros conhecimentos específicos, como Hadoop, SQL, Apache Spark ou algoritmos de machine learning vão depender do tipo de tarefa e/ou estrutura de dados a que o profissional está submetido.

3 Dados Univariados

O R possui cinco classes básicas (ou *atômicas*) de objetos, a saber: *character*, *numeric*, *integer*, *complex* e *logical*. Os objetos no R, por sua vez, são divididos em algumas `estruturas de dados`, a saber: *vetores atômicos*, *matrizes*, *arrays*, *listas* e *data frames*. Cada uma dessas estruturas possui características próprias, servindo a determinados tipos de dados. Elas podem, entretanto, ser organizadas, como propõe Wickham (2014), pela *dimensionalidade* e pelo fato de poderem receber dados homogêneos (todos os elementos precisam ser do mesmo tipo) ou heterogêneos, da seguinte forma:

Organização de estruturas de dados no R

	Homogêneos	Heterogêneos
1d	Atomic Vector	List
2d	Matrix	Data frame
nd	Array	

O mais básico tipo de objeto no R é o `vetor`. Vetores podem ser de dois tipos: `vetores atômicos` ou `listas`. Eles possuem três propriedades comuns:

- Tipo, `typeof()`, o que é;
- Tamanho, `length()`, quantos elementos possui;
- Atributos, `attributes()`, metadados arbitrários adicionais.

A diferença entre eles, como visto acima, é que todos os elementos em um vetor atômico precisam ser da mesma classe, enquanto na lista, não necessariamente. Abaixo um exemplo de vetor numérico de tamanho 10 no R.

```
x = seq(1:10) # criando uma sequência de 1 a 10
vetor = rnorm(x, 34, 12) # geração aleatório de números
class(vetor) # verificando a classe do objeto

## [1] "numeric"

length(vetor) # verificando o tamanho do objeto

## [1] 10
```

```
typeof(vetor) # o tipo do objeto

## [1] "double"

str(vetor) # A estrutura do objeto

##  num [1:10] 31.6 37.8 27.8 61.6 51 ...
```

Sobre um vetor, é possível fazer operações de *redução* ou de *vetorização*, como abaixo.

```
sum(vetor)/length(vetor) # Obtendo a média dos dados

## [1] 39.01533

vetor - mean(vetor) # O quanto cada observação se distancia da média

##  [1]  -7.415040  -1.255253 -11.167355  22.605764  12.007812  -8.475240
##  [7]  16.347699 -22.483062   3.701048  -3.866373
```

No primeiro caso, estamos fazendo uma operação que irá resultar em único número. Já no segundo, estamos fazendo uma operação com cada elemento do vetor, o que naturalmente irá resultar em um novo vetor.

Vamos avançar um pouco no entendimento e manipulação dessa estrutura básica no **R** por meio de um exemplo mais real. Para isso, precisamos instalar o pacote `UsingR`, que acompanha o livro Verzani (2014), nossa referência bibliográfica básica nesse curso. O código abaixo ilustra a instalação e posterior carregamento do pacote.[7]

```
install.packages('UsingR')
require(UsingR)
```

Feito isso, podemos acessar todos os `data sets` do pacote com o comando abaixo.

```
data(package='UsingR')
```

Escolhemos *brincar* com o *data set* `coldvermont`, que traz a temperatura mínima diária, em *Fahrenheit*, em Woodstock Vermont entre os anos de 1980 e 1985.

```
library(UsingR)
class(coldvermont)

## [1] "ts"
```

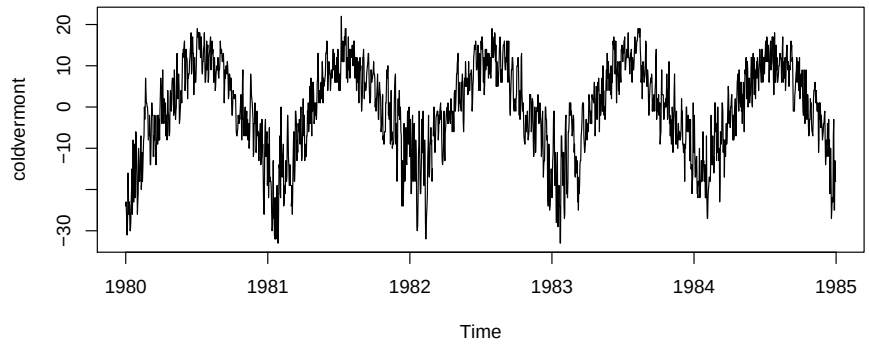
[7] Caso tenha dificuldades nesse processo, dê uma olhada na subseção **??**.

Observe que a classe do objeto é `ts`, de série temporal. Podemos, para facilitar a interpretação, passar de graus Fahrenheit para graus Celsius, através da fórmula abaixo.

```
coldvermont = round((coldvermont-32)/1.8,0)
```

E abaixo, plotamos um gráfico rápido do nosso conjunto de dados.

```
plot(coldvermont)
```



Observe que as temperaturas mínimas apresentam uma *sazonalidade* ao longo do ano. Isto é, elas aumentam durante a primeira metade e diminuem na segunda metade do ano.

3.1 Medidas de tendência central

De modo a compreender melhor o nosso conjunto de dados `coldvermont`, podemos agora fazer uso de algumas estatísticas descritivas, tais como as de **medida central**, **amplitude** e **forma**. Para começar, vamos verificar a média e a mediana do nosso conjunto de dados.[8]

```
mean(coldvermont, na.rm=TRUE)
```

[1] -0.641955

```
median(coldvermont, na.rm=TRUE)
```

[1] 1

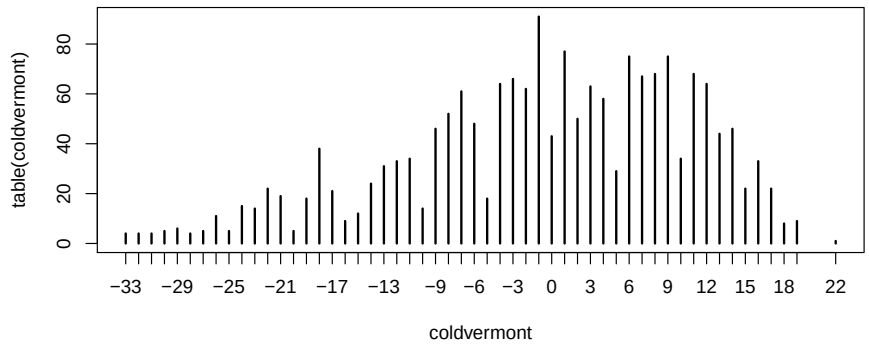
[8] Observe que estamos utilizando o argumento *na.rm*, de modo a ignorar os valores faltantes (*missing values*) dentro da nossa amostra.

Enquanto a média do nosso conjunto de dados é -0.64, a mediana é 1. A mediana, por suposto, divide a amostra em dois, isto é, valores menores do que ela e valores maiores do que ela. Em termos gerais, podemos dividir a amostra em *quantis*, de forma a estabelecer *medidas de posição*, tais como:

```
quantile(coldvermont, seq(0,1,0.25), na.rm=TRUE)

##   0%  25%  50%  75% 100%
##  -33   -7    1    8   22
```

Observe que o primeiro quartil da nossa amostra vai até a temperatura -7 graus Celsius, enquanto o terceiro quartil vai até 8 graus Celsius. Uma outra medida importante para analisarmos o nosso conjunto de dados é a **moda**, isto é, o valor mais frequente que podemos encontrar. Não há uma função direta no R que faz isso, infelizmente. Mas podemos brincar um pouco para descobrir. Podemos usar a função `table` de modo a tabular o número de vezes que cada temperatura ocorre na nossa amostra. Fazemos isso com o código abaixo. Para calculá-lo no R, podemos utilizar o código abaixo.

```
plot(table(coldvermont))
```



Fica fácil visualizar que o número -1 é a temperatura mais comum no nosso conjunto de dados. De modo a obter esse número, podemos utilizar a linha de código abaixo.

```
as.numeric(names(which(table(coldvermont)==max(table(coldvermont)))))

## [1] -1
```

3.2 Medidas de dispersão e formatos de distribuição

A amplitude ou variabilidade de um conjunto de dados é uma importante característica. Uma medida simples disso poderia ser o intervalo de um determinado conjunto de dados, que nada mais é do que a distância entre o mínimo e o máximo. Isto é,

```
c(min(coldvermont, na.rm=TRUE), max(coldvermont, na.rm=TRUE))

## [1] -33  22
```

Nosso conjunto de temperaturas *varia* assim de -33 a 22, sendo o valor mais comum -1, a média -0.64 e a mediana 1. Pode ser chato, ademais, toda hora ter de aplicar o argumento *na.rm* nas funções. Nesse caso, podemos simplesmente retirar os valores faltantes do nosso conjunto de dados com o comando abaixo.

```
coldvermont = coldvermont[complete.cases(coldvermont)]
```

Os dados *sumarizados* acima, podem, ademais, ser obtidos de forma direta com a função *summary*, como abaixo.

```
summary(coldvermont)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -33.000  -7.000   1.000  -0.642   8.000  22.000
```

A distância entre o menor e o maior valor de um conjunto de dados também pode ser visto de forma direta com a função `range`, como abaixo.

```
range(coldvermont)

## [1] -33  22

diff(range(coldvermont))

## [1] 55
```

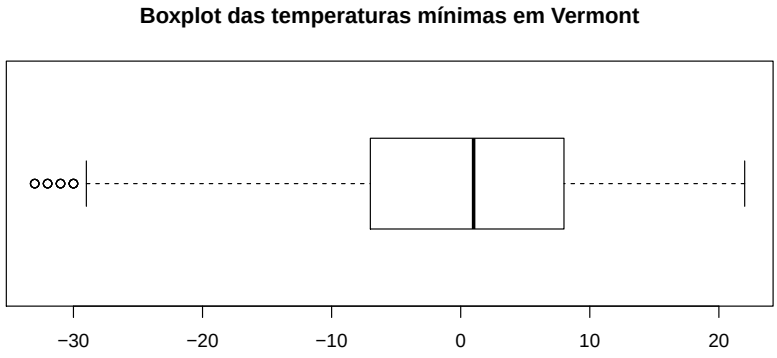
Já a diferença entre o primeiro e o terceiro quartil, denominado como *Interquartile Range (IQR)*, pode ser obtido da seguinte forma.

```
IQR(coldvermont)

## [1] 15
```

Esses cinco valores de um conjunto de dados, ademais, pode ser visualizado de forma direta por meio de *boxplots*. No R, está implementado na função de mesmo nome, como

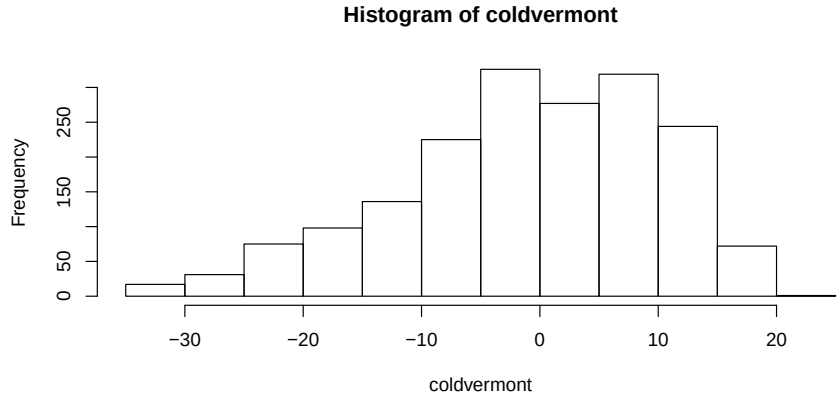
abaixo.

```
boxplot(coldvermont,
        main='Boxplot das temperaturas mínimas em Vermont',
        horizontal=TRUE)
```



Uma outra forma de visualizar a amplitude do nosso conjunto de dados é através de *histogramas*, como abaixo.

```
hist(coldvermont)
```



Através do boxplot e do histograma, a propósito, é possível ilustrar a **forma** como as observações de um determinado conjunto de dados se distribui. Elas podem apresentar, basicamente, três formatos: distribuição simétrica, uma assimetria à esquerda e uma assimetria à direita. No caso das temperaturas diárias mínimas de Vermont, vemos claramente

uma assimetria à esquerda.[9] É o caso típico quando a média é menor do que a mediana.[10]

3.2.1 Variância e desvio-padrão

Por fim, de modo a identificar melhor a variabilidade de um determinado conjunto de dados, podemos calcular a distância entre uma determinada observação e a média do mesmo. De forma a considerar esses desvios independentes do sinal, devemos somar os desvios ao quadrado, controlando pelo tamanho da amostra. Isto é,

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{1}$$

De modo que, quanto mais espalhado for o conjunto de dados, maior será a sua **variância**. Podemos, ademais, padronizar a unidade de medida através da raiz quadrada da variância, o que dará o **desvio-padrão**. No R, isso é facilmente implementado conforme o código abaixo.

```
var(coldvermont)

## [1] 122.0113

sd(coldvermont)

## [1] 11.04587
```

Quanto maiores forem os valores para a variância, mais amplo será o espalhamento dos dados em relação à média. Já o desvio-padrão indica, em média, quanto cada valor se diferencia da média.

3.3 Dados categóricos

Sumarizar dados categóricos é algo bastante direto. A ferramenta básica é **tabular** os valores e apresentá-los, seja de forma gráfica ou descritiva. Para ilustrar, vamos considerar

[9] Alguns livros se referem a isso como **assimetria negativa**.

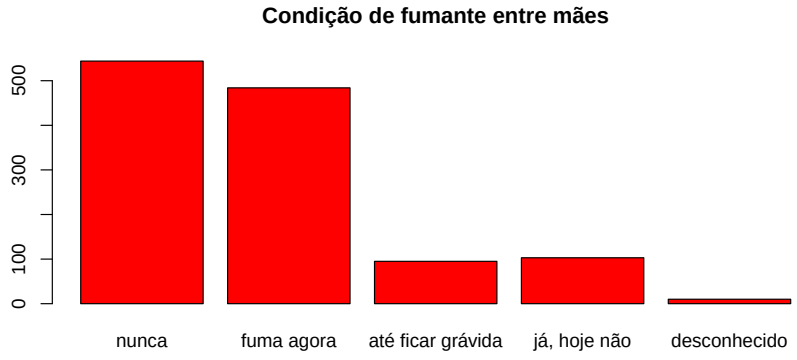
[10] Quando a média é maior do que a mediana, ocorre uma *assimetria à direita*. Já quando a média é igual a mediana, ocorre uma simetria.

a variável `smoke` no conjunto de dados `babies`, que representa a condição de fumante entre mães.[11]

```
x = babies$smoke
x = factor(x, labels=c('nunca', 'fuma agora', 'até ficar grávida',
                       'já, hoje não', 'desconhecido'))
table(x)

## x
##             nunca        fuma agora até ficar grávida      já, hoje não
##               544               484                95               103
##      desconhecido
##                10
```

Como podemos ver, a função `table` relaciona cada categoria à frequência encontrada. Podemos visualizar isso através de um gráfico de barras, como abaixo.

```
barplot(table(x),
        col='red', main='Condição de fumante entre mães',
        horiz = FALSE)
```



[11] Os códigos utilizados estão disponíveis na ajuda do conjunto de dados.

4 Dados Bivariados

Nessa seção, veremos os conjuntos de dados bivariados, dados envolvendo duas variáveis. Quando estamos tratando desse tipo de conjunto, nós podemos indagar se há relação entre as variáveis analisadas. Para *amostras independentes*, nós podemos nos perguntar se diversas métricas, como medidas centrais, de variabilidade ou forma são similares ou diferentes. Já para *dados emparelhados*, podemos verificar a existência ou ausência de relacionamento entre as variáveis.

4.1 Amostras Independentes

Um experimento estatístico bastante comum consiste em separar uma determinada coorte em grupos de controle e de tratamento. Enquanto o grupo de tratamento recebe algum tipo de tratamento, o de controle não recebe nada. Depois, medidas para cada um dos grupos são tomadas. De modo a controlar o viés ao separar a coorte, uma alocação aleatória costuma ser utilizada. Esse tipo de procedimento leva a independência - onde o conhecimento sobre a performance de um determinado participante não sugere nada sobre outro. Ademais, para controlar o *efeito placebo*, o grupo de controle geralmente recebe algum tratamento, mas que não é esperado ter qualquer efeito.

Por exemplo, suponha que estamos investigando quais alimentos podem ter impacto sobre a performance esportiva. O grupo de tratamento foi alimentado com $\frac{3}{2}$ copos de beterraba 75 minutos antes de uma atividade física, enquanto o grupo de controle não foi. A atividade física foi uma corrida cronometrada. O pesquisador acredita que o nitrato encontrado na beterraba ampliaria os vasos sanguíneos do indivíduo, aumentando o fluxo sanguíneo nos músculos exercicitados, diminuindo o tempo de duração da atividade. Suponha que as medidas em minutos das atividades sejam dadas por:

Beterrabas	41	40	41	42	44	35	41	36	47	45
Sem Beterrabas	51	51	50	42	40	31	43	45		

O que podemos dizer sobre esses dados? Os três maiores tempos estão no grupo *sem beterrabas*, mas este também possui o menor tempo. Algumas questões pertinentes:

- Os dados parecem ter sido extraídos da mesma população, os dois possuem a mesma forma?
- Os dados parecem ter as mesmas medidas centrais e de variabilidade?

Vamos para o **R** responder essas questões...

```
beets = c(41,40,41,42,44,35,41,36,47,45)
nobeets = c(51,51,50,42,40,31,43,45)

c(media_beets=mean(beets), media_nobeets=mean(nobeets),
  sd_beets=sd(beets), sd_nobeets=sd(nobeets))
```

```
##   media_beets media_nobeets      sd_beets    sd_nobeets
##     41.200000     44.125000      3.705851      6.812541
```

O grupo que recebeu as beterrabas possui uma média de 3 minutos a menos do que o grupo de controle, com uma variabilidade menor. Se essa diferença se dá por termos corredores mais rápidos no grupo de tratamento ou se é resultado da ingestão das beterrabas, nós não sabemos.

Assim como na análise univariada, pode ser importante visualizar os dados.

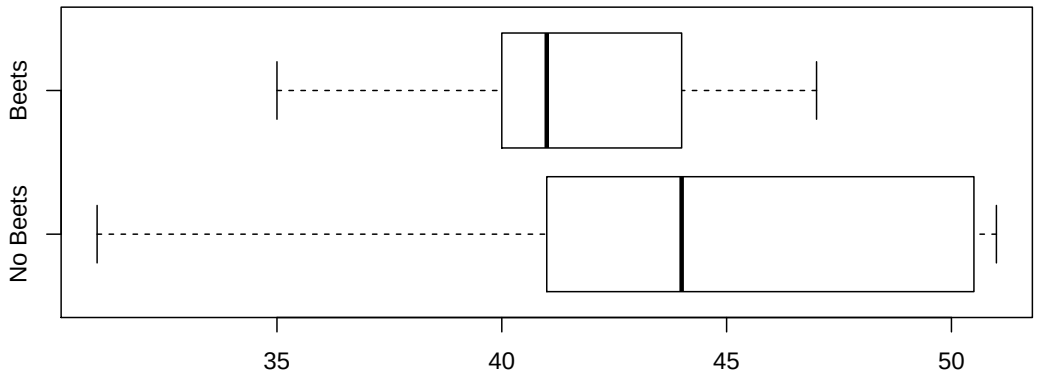
```
library(aplpack)
stem.leaf.backback(beets, nobeets, rule.line='Sturges')
```

```
## ------------------------------
##   1 | 2: represents 12, leaf unit: 1
##         beets      nobeets
## ------------------------------
##              | 3* |1        1
##    2       65| 3. |
##   (6)  421110| 4* |023      4
##    2       75| 4. |5       (1)
##              | 5* |011      3
##              | 5. |
##              | 6* |
## ------------------------------
## n:         10      8
## ------------------------------
```

Os lados mais à esquerda e à direita registram a posição da observação no conjunto de

dados, as hastes ficam no meio. Após focarmos no meio, percebe-se que o grupo de controle (sem beterrabas) possui maior variabilidade, mas a diferença no centro não é tão óbvia. De forma a complementar a análise, colocamos abaixo um `boxplot`. Observamos medianas diferentes, mas ambas estão dentro da amplitude da outra. A ideia de olhar as diferenças no centro de uma escala determinada pela amplitude é a chave para a inferência estatística.

```
boxplot(nobeets, beets, names=c('No Beets', 'Beets'),
        horizontal = TRUE)
```

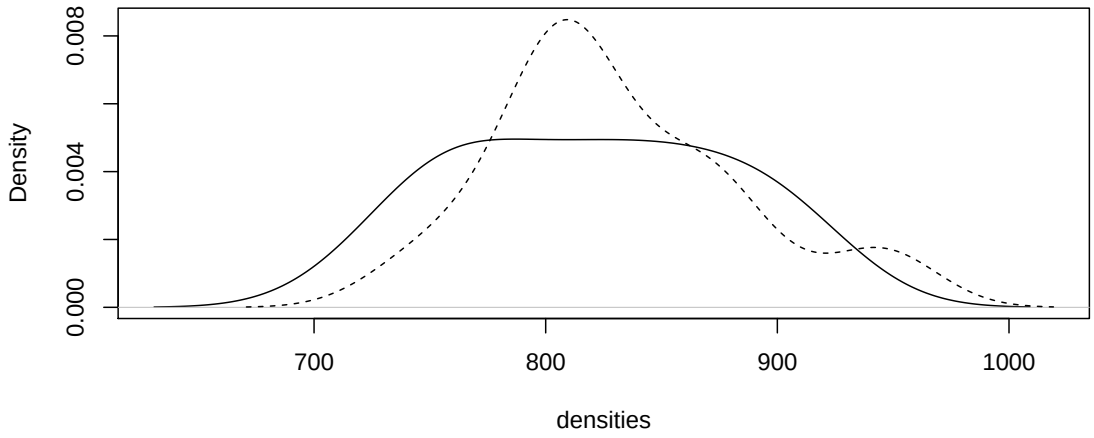


4.1.1 Gráficos de densidade

Gráficos de densidade são semelhantes aos `histogramas` que vimos na seção anterior, mas difere por permitir que mostremos dois ou mais conjuntos de dados. Vamos considerar o conjunto de dados `michelson` do pacote `MASS`, que contém medidas de velocidade da luz no ar produzidas por Michelson em 1879. O conjunto de dados contém cinco experimentos. Nós comparamos o quarto e quinto abaixo.

```
library(MASS)
speed <- michelson$Speed; expt <- michelson$Expt
fourth <- speed[expt==4]
fifth <- speed[expt==5]
d_4 <- density(fourth)
```

```r
d_5 <- density(fifth)
xrange <- range(c(d_4$x, d_5$x))
yrange <- range(c(d_4$y, d_5$y))
plot(d_4, xlim=xrange, ylim=yrange, xlab='densities', main='')
lines(d_5, lty=2)
```



4.2 Manipulação de Dados no R

O R possui duas estruturas que podem simplificar o trabalho com dados bivariados: listas e data frames. As listas têm uma vantagem sobre vetores atômicos, como vimos, por permitir a alocação de dados com categorias diferentes. Podemos construí-las com o código abaixo.

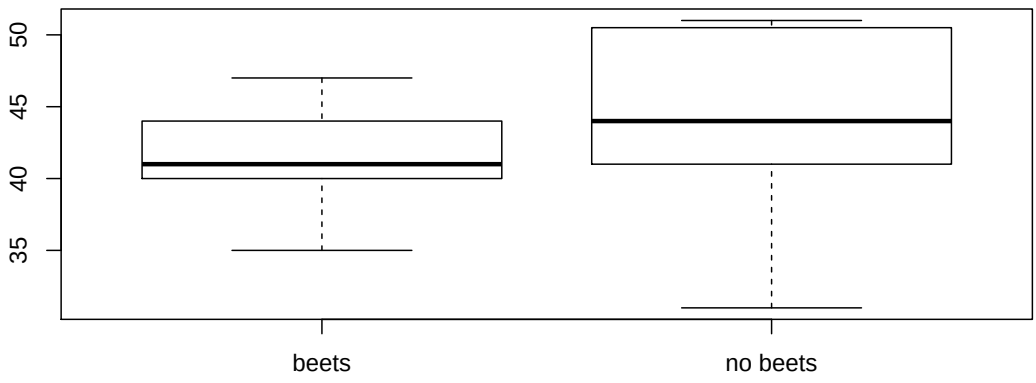
```r
beets = c(41,40,41,42,44,35,41,36,47,45)
nobeets = c(51,51,50,42,40,31,43,45)

b = list(beets=beets, 'no beets'=nobeets)
b

## $beets
##  [1] 41 40 41 42 44 35 41 36 47 45
##
## $`no beets`
## [1] 51 51 50 42 40 31 43 45
```

As listas, a propósito, podem ser subdivididas.

```
b$beets

##  [1] 41 40 41 42 44 35 41 36 47 45

b$`no beets`

## [1] 51 51 50 42 40 31 43 45

b[1]

## $beets
##  [1] 41 40 41 42 44 35 41 36 47 45

b[2]

## $`no beets`
## [1] 51 51 50 42 40 31 43 45

b[[1]][1]

## [1] 41
```

E facilitam a construção do *boxplot*.

```
boxplot(b)
```



`Data frames`, por sua vez, são uma forma muito utilizada para armazenar variáveis, de forma que as colunas representam variáveis e as observações ficam nas linhas. Abaixo um exemplo.

```r
student = c('Alice', 'Bob')
grade = c('A', 'B')
attendance = c('awesome', 'bad')
data = data.frame(student, grade, attendance)
data
```

```
##   student grade attendance
## 1   Alice     A    awesome
## 2     Bob     B        bad
```

4.3 Dados Emparelhados

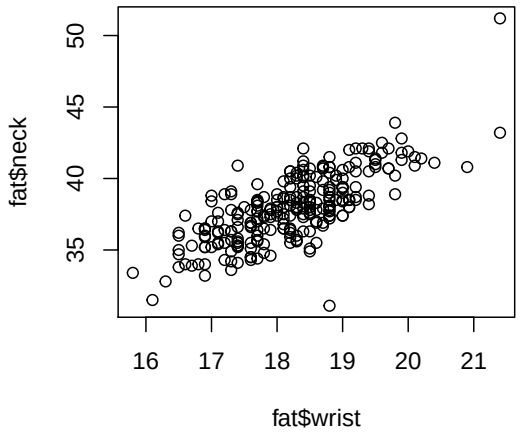
Considere o conjunto de dados `fat` do pacote `UsingR` que contém medidas de diferentes dimensões do corpo de uma coorte de 252 homens. O objetivo desse conjunto de dados foi o de verificar se alguma previsão do índice de gordura corporal pode ser feita a partir de algumas variáveis que podem ser obtidas a partir de uma fita métrica. Para esse exemplo particular, estamos interessados apenas nas relações entre as variáveis.

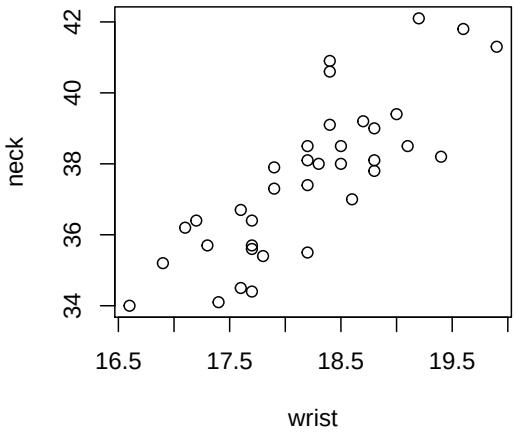
```r
library(UsingR)
names(fat)
```

```
##  [1] "case"          "body.fat"      "body.fat.siri" "density"
##  [5] "age"           "weight"        "height"        "BMI"
##  [9] "ffweight"      "neck"          "chest"         "abdomen"
## [13] "hip"           "thigh"         "knee"          "ankle"
## [17] "bicep"         "forearm"       "wrist"
```

E abaixo um primeiro `gráfico de correlação` entre as medidas do pulso (*wrist*) e do pescoço (*neck*).

```r
par(mfrow=c(1,2))
plot(fat$wrist, fat$neck)
plot(neck~wrist, data=fat, subset= 20 <= age & age < 30)
```





A correlação é um sumário numérico que diz o quão perto estão relacionadas as medidas de duas variáveis numéricas quando elas estão uma relação linear. O *coeficiente de correlação de Pearson* pode ser definido como

$$cor(x, y) = \frac{1}{n-1} \sum \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right) \tag{2}$$

de modo que o número permaneça no intervalo entre 1 e -1, representando, respectivamente, perfeita correlação positiva e negativa. Se o número for 0, por outro lado, isso significa que as variáveis são perfeitamente não correlacionadas.

5 Dados Multivariados

Um processo de análise de dados típico consiste em algumas etapas: tratamento dos dados, transformação dos dados, visualização exploratória, sumários numéricos e modelagem. Aprender de modo eficiente a como trabalhar no R com `data frames` torna essas etapas muito mais diretas e gerenciáveis. Nessa seção, aprenderemos sobre como trabalhar com `data frames`.

5.1 Trabalhando com data frames

`Data frames` se situam na fronteira entre matrizes e listas de modo que nós temos duas interfaces para trabalhar dentro da mesma estrutura. Ademais, é possível utilizar determinadas funções para tornar o gerenciamento dessa estrutura de dados ainda mais conveniente. Vamos organizar nossa discussão em alguns aspectos, como se segue.

5.1.1 Construindo um data frame

Data frames são listas de variáveis. Eles podem ser construídos como se segue.

```
df = data.frame(nm1=vec1, nm2=vec2)
```

A combinação `key=value` equivale às colunas, enquanto as linhas são criadas automaticamente. Observe, por suposto, que os vetores precisam ter o mesmo tamanho para estarem no mesmo data frame. Com a função `as.data.frame` é possível ainda forçar uma estrutura a ser um data frame.

5.1.2 Pesquisa de dados

É possível acesssar as variáveis de um data frame através da notação $, da seguinte forma:

```
data(mtcars)
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
```

```
## [29] 15.8 19.7 15.0 21.4
```

É possível ainda pesquisar dados como em matrizes, da seguinte forma:

```
mtcars[,1]
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

Ou fazer operações dentro do mesmo data frame com a função `with`:

```
with(mtcars, mean(mpg) - sd(mpg))
```

```
## [1] 14.06368
```

5.1.3 Modificando linhas, colunas e nomes

A notação de matriz é útil para modificar blocos de linhas em um data frame.

```
library(UsingR)
d = Cars93[1:3,1:4]
d[1,1] = d[3,4] = NA
d
```

```
##   Manufacturer   Model    Type Min.Price
## 1         <NA> Integra   Small      12.9
## 2        Acura  Legend Midsize      29.2
## 3         Audi      90 Compact        NA
```

Observe que o objeto d acima é composto por quatro colunas, sendo três delas recebendo *Factors*. Observe abaixo:

```
str(d)
```

```
## 'data.frame': 3 obs. of  4 variables:
##  $ Manufacturer: Factor w/ 32 levels "Acura","Audi",..: NA 1 2
##  $ Model       : Factor w/ 93 levels "100","190E","240",..: 49 56 9
##  $ Type        : Factor w/ 6 levels "Compact","Large",..: 4 3 1
##  $ Min.Price   : num  12.9 29.2 NA
```

Assim, se quisermos, por exemplo, adicionar uma linha precisamos primeiro criar os *levels* para o *Factor* envolvido:

```
levels(d$Model) = c(levels(d$Model), c('A3', 'A4', 'A6'))
d[4,] = list('Audi', 'A4', 'Midsize', 35)
d
```

```
##   Manufacturer   Model    Type Min.Price
## 1         <NA> Integra   Small      12.9
## 2        Acura  Legend Midsize      29.2
## 3         Audi      90 Compact        NA
## 4         Audi      A4 Midsize      35.0
```

Uma outra forma de adicionar uma linha, por suposto, é com a função `rbind`:

```
d = rbind(d, list('Audi', 'A6', 'Large', 45))
d
```

```
##   Manufacturer   Model    Type Min.Price
## 1         <NA> Integra   Small      12.9
## 2        Acura  Legend Midsize      29.2
## 3         Audi      90 Compact        NA
## 4         Audi      A4 Midsize      35.0
## 5         Audi      A6   Large      45.0
```

Adicionar uma coluna pode ser feito da seguinte forma:

```
d[,5] = d$Min.Price*1.3 # Preço em Euro
d
```

```
##   Manufacturer   Model    Type Min.Price    V5
## 1         <NA> Integra   Small      12.9 16.77
## 2        Acura  Legend Midsize      29.2 37.96
## 3         Audi      90 Compact        NA    NA
## 4         Audi      A4 Midsize      35.0 45.50
## 5         Audi      A6   Large      45.0 58.50
```

Ou, de forma a nomear a coluna:

```
d = d[,-5]
d$Min.Price.Euro = d$Min.Price*1.3
d
```

```
##   Manufacturer   Model    Type Min.Price Min.Price.Euro
## 1         <NA> Integra   Small      12.9          16.77
## 2        Acura  Legend Midsize      29.2          37.96
## 3         Audi      90 Compact        NA             NA
```

```
## 4         Audi      A4 Midsize      35.0          45.50
## 5         Audi      A6   Large      45.0          58.50
```

Podemos colocar todos os nomes em minúsculo com o código abaixo:

```
names(d) = tolower(names(d))
d

##   manufacturer   model    type min.price min.price.euro
## 1         <NA> Integra   Small      12.9          16.77
## 2        Acura  Legend Midsize      29.2          37.96
## 3         Audi      90 Compact        NA             NA
## 4         Audi      A4 Midsize      35.0          45.50
## 5         Audi      A6   Large      45.0          58.50
```

E podemos modificar o nome de uma coluna como abaixo:

```
names(d)[3] = 'car type'
d

##   manufacturer   model car type min.price min.price.euro
## 1         <NA> Integra    Small      12.9          16.77
## 2        Acura  Legend  Midsize      29.2          37.96
## 3         Audi      90  Compact        NA             NA
## 4         Audi      A4  Midsize      35.0          45.50
## 5         Audi      A6    Large      45.0          58.50
```

5.1.4 A função `subset`

Como vimos acima, podemos extrair blocos de um data frame especificando as linhas e colunas do mesmo. Isso, entretanto, pode ser complicado se o seu data frame possuir muitas linhas e colunas. Um modo mais interessante de fazê-lo pode ser utilizando a função `subset` como abaixo:

```
aq = airquality[1:5,]
aq

##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
```

```
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
```

```
subset(aq, select=Ozone:Wind)
```

```
##   Ozone Solar.R Wind
## 1    41     190  7.4
## 2    36     118  8.0
## 3    12     149 12.6
## 4    18     313 11.5
## 5    NA      NA 14.3
```

```
subset(aq, select=-c(Month,Day))
```

```
##   Ozone Solar.R Wind Temp
## 1    41     190  7.4   67
## 2    36     118  8.0   72
## 3    12     149 12.6   74
## 4    18     313 11.5   62
## 5    NA      NA 14.3   56
```

```
subset(aq, subset=!is.na(Ozone), select=Ozone:Wind)
```

```
##   Ozone Solar.R Wind
## 1    41     190  7.4
## 2    36     118  8.0
## 3    12     149 12.6
## 4    18     313 11.5
```

5.1.5 is.na, complete.cases

Quando estamos extraindo dados de um data frame, pode ser interessante ignorar linhas com NA. Observe abaixo:

```
DF = data.frame(a=c(NA,1,2), b=c('one', NA, 'three'))
DF
```

```
##    a     b
## 1 NA   one
## 2  1  <NA>
## 3  2 three
```

```
subset(DF, !is.na(a))
```

```
##   a     b
## 2 1  <NA>
## 3 2 three
```

```
subset(DF, complete.cases(DF))
```

```
##   a     b
## 3 2 three
```

5.1.6 Transformando valores

Em geral, a maioria das funções no R são *funções puras* no sentido de que elas não modificam os argumentos passados para elas. A integridade dos dados é respeitada dentro do ambiente R, de modo que modificações inesperadas são evitadas. Em alguns momentos, entretanto, alguma transformação dos dados pode ser importante para o analista. Vimos acima um exemplo com a taxa de câmbio. Abaixo uma outra forma de fazer essa mudança com a função `within`.

```
d = within(d, {min.price.euro = min.price*1.5})
d
```

```
##   manufacturer   model car type min.price min.price.euro
## 1         <NA> Integra    Small      12.9          19.35
## 2        Acura  Legend  Midsize      29.2          43.80
## 3         Audi      90  Compact        NA             NA
## 4         Audi      A4  Midsize      35.0          52.50
## 5         Audi      A6    Large      45.0          67.50
```

Também é possível utilizar a função `transform` como abaixo.

```
d = transform(d, min.price.euro = min.price*1.5)
d
```

```
##   manufacturer   model car.type min.price min.price.euro
## 1         <NA> Integra    Small      12.9          19.35
## 2        Acura  Legend  Midsize      29.2          43.80
## 3         Audi      90  Compact        NA             NA
## 4         Audi      A4  Midsize      35.0          52.50
## 5         Audi      A6    Large      45.0          67.50
```

5.1.7 Reshaping

Dados podem ser apresentados em um formato amplo. Por exemplo, podemos ter um conjunto de dados da seguinte forma:

```
head(speed)
```

```
##   Speed.1 Speed.2 Speed.3 Speed.4 Speed.5
## 1     850     960     880     890     890
## 2     740     940     880     810     840
## 3     900     960     880     810     780
## 4    1070     940     860     820     810
## 5     930     880     720     800     760
## 6     850     800     720     770     810
```

Os dados acima são extraídos do conjunto de dados `morley`, que possui dados sobre um experimento com a velocidade da luz.[12] **Dados amplos**, onde diferentes medidas de algum valor são arquivados como variáveis separadas, é comumente utilizado para visualização de dados. Entretanto, **dados longos** onde diferentes valores são arquivados com fatores é mais útil para uso com funções do R. Desse modo, fazer a conversão entre uma e outra forma é conhecido como *reshaping* dos dados. Abaixo um exemplo.

```
m = reshape(speed, varying = names(speed)[1:5], direction = 'long')
head(m)
```

```
##     time Speed id
## 1.1    1   850  1
## 2.1    1   740  2
## 3.1    1   900  3
## 4.1    1  1070  4
## 5.1    1   930  5
## 6.1    1   850  6
```

[12] Você pode consultar as informações desse `data set` com ?morley.

5.2 Aplicando uma função sobre uma coleção

Nós nos voltamos agora para construções no R que objetivam aplicar uma função sobre um grupo formado para algum processo de *splitting*. Vimos anteriormente que algumas funções no R são *vetorizadas*. Por exemplo:

```
x = 1:5
x - 3

## [1] -2 -1  0  1  2
```

Uma outra forma de fazer a mesma operação segue abaixo.

```
x = 1:5
res = integer(length(x))
for(i in 1:length(x)){
  res[i] = x[i] - 3
}
res

## [1] -2 -1  0  1  2
```

O código acima faz uma iteração sobre um conjunto de números que são os índices de x e usa esse *index* para verificar o correspondente valor de x de modo a conseguir subtrair 3 das mesmas. Esse tipo de iteração sobre um conjunto ou coleção de elementos usando um loop `for` é elguns momentos desejável.

Se uma função não é vetorizada, a função `Vectorize` irá fazê-lo. Veja um exemplo abaixo.

```
vmedian = Vectorize(median)
vmedian(homedata)

##  y1970  y2000
##  68900 251700
```

5.2.1 Map

Vetorização é um exemplo de *map*. Um *map* toma uma função f e aplica a mesma sobre cada elemento de uma coleção, retornando uma nova coleção. Abaixo um exemplo.

```
collection = c(4,9,16)
Map(sqrt, collection)
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 3
##
## [[3]]
## [1] 4
```

5.2.2 A função `sapply`

A função `sapply` é um exemplo de `simplify2array`, um processo de fazer o trabalho de combinar o *output* de uma forma aprazível. A chamada da função `sapply` é o inverso da que vimos a função `Map` acima.

```
sapply(collection, sqrt)

## [1] 2 3 4
```

Na sequência, nós *splitamos* um conjunto de dados por um fator de modo a obter uma lista. Depois, nós aplicamos a função da média a cada uma.

```
lst = with(ToothGrowth, split(len, supp))
sapply(lst, mean)

##       OJ       VC
## 20.66333 16.96333
```

5.2.3 A função `tapply`

Uma forma de fazer a mesma coisa acima, de forma particular:

```
with(ToothGrowth, tapply(len, supp, mean))

##       OJ       VC
## 20.66333 16.96333
```

5.2.4 A função `aggregate`

A função `aggregate` é uma função alternativa que provê uma interface para fórmulas. Uma sintaxe mais conveniente do que aquela proposta pela função `with`. Veja abaixo um exemplo.

```
aggregate(len~supp, data=ToothGrowth, mean)

##   supp      len
## 1   OJ 20.66333
## 2   VC 16.96333
```

5.2.5 A função `apply`

Para matrizes (ou *arrays*), funções vetorizadas são aplicadas a cada elemento, tratando a matriz como um vetor com alguns atributos de forma. Por exemplo,

```
m = rbind(c(1,2), c(3,4))
m

##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4

sqrt(m)

##          [,1]     [,2]
## [1,] 1.000000 1.414214
## [2,] 1.732051 2.000000
```

Pode ser desejável, entretanto, aplicar a função à cada coluna, como em um data frame. Ou, como as linhas de uma matriz são do mesmo tipo, pode ser interessante aplicar uma função a cada linha. Vamos ilustrar alguns exemplos abaixo.

```
m = replicate(5, rnorm(3))
m

##             [,1]      [,2]       [,3]       [,4]       [,5]
## [1,] -0.04598935 1.0561260  2.4928894 -1.2629394 -0.0575376
## [2,]  0.18977922 0.5084391 -1.6630999 -0.3517931 -1.3672294
## [3,]  0.42189720 0.3308661  0.3163945 -0.4082029 -0.1928928

rowSums(m) # Soma as linhas
```

```
## [1]  2.1825491 -2.6839041  0.4680621

colSums(m) # Soma as colunas

## [1]  0.5656871  1.8954312  1.1461839 -2.0229354 -1.6176597
```

Como a matriz m é tratada como um vetor quando vista como uma coleção, a função `sapply` não funciona para esse tipo de tarefa.

```
sapply(m, sum)

##  [1] -0.04598935  0.18977922  0.42189720  1.05612603  0.50843906
##  [6]  0.33086610  2.49288939 -1.66309994  0.31639449 -1.26293942
## [11] -0.35179306 -0.40820289 -0.05753760 -1.36722938 -0.19289275
```

Dado que precisamos aplicar a função a uma dimensão específica, precisaremos de uma outra forma. Veja abaixo.

```
apply(m, 1, mean) # um rowMeans alternativo

## [1]  0.43650981 -0.53678082  0.09361243

apply(m, 2, mean) # um colMeans alternativo

## [1]  0.1885624  0.6318104  0.3820613 -0.6743118 -0.5392199
```

5.2.6 A função `mapply`

Mostramos até aqui funções que tomam uma única variável e mostram algum *output*. Há funções, entretanto, que tomam duas ou mais variáveis. Veja um exemplo abaixo.

```
min(c(1,3), c(2,3))

## [1] 1

mapply(min, c(1,4), c(2,3))

## [1] 1 3
```

5.2.7 A função `do.call`

```
body = Animals$body; brain = Animals$brain
do.call(cor, Map(rank, list(body, brain)))

## [1] 0.7162994
```

```
do.call(cor, Map(rank, setNames(Animals, NULL)))
```

```
## [1] 0.7162994
```

5.2.8 A função `Filter`

Outro conceito bastante útil é filtrar ou extrair elementos de uma coleção. Por exemplo,

```
m = Cars93[1:2, 1:15]
Filter(is.factor, m)
```

```
##   Manufacturer   Model    Type            AirBags DriveTrain Cylinders
## 1        Acura Integra   Small               None      Front         4
## 2        Acura  Legend Midsize Driver & Passenger      Front         6
```

5.3 Dados externos

Para terminar essa seção, fazemos uma breve descrição sobre como importar dados para o R. A fonte mais comum de dados está disponível em planilhas, do tipo EXCEL. Abaixo uma forma de importá-las e tratá-las, de modo a obter um *dado limpo*.

```
library(XLConnect)
url1 = 'http://www.anp.gov.br/images/Precos/Mensal2001-2012/Brasil.xlsx'
temp = tempfile()
download.file(url1, destfile=temp, mode='wb')
data1 = loadWorkbook(temp)
url2 = 'http://www.anp.gov.br/images/Precos/Mensal2013/MENSAL_BRASIL-DESDE_Jan2013.xlsx'
temp = tempfile()
download.file(url2, destfile=temp, mode='wb')
data2 = loadWorkbook(temp)
tabela1 = readWorksheet(data1, sheet = 1, header = TRUE, startRow = 13)
gasolina1 = tabela1$PRECO.MÉDIO.DISTRIBUIÇÃO[tabela1$PRODUTO=='GASOLINA COMUM']
tabela2 = readWorksheet(data2, sheet = 1, header = TRUE, startRow = 15)
gasolina2 = tabela2$PREÇO.MÉDIO.DISTRIBUIÇÃO[tabela2$PRODUTO=='GASOLINA COMUM']
gasolina = ts(c(gasolina1, gasolina2), start=c(2001,07), freq=12)
```

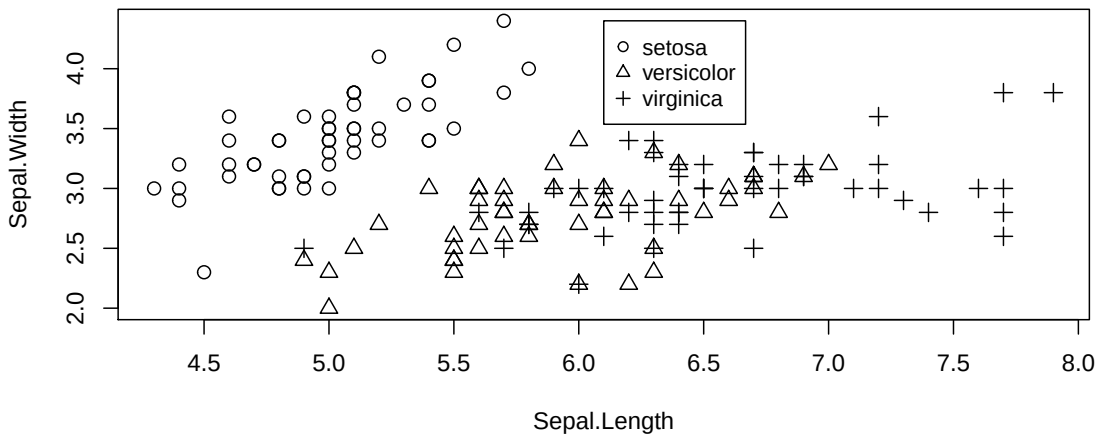
6 Gráficos Multivariados

Essa seção do nosso Curso de Formação Cientista de Dados cobre algumas formas de visualização de dados multivariados. Vamos ver, basicamente, gráficos que podem ser obtidos com o código-base do R, gráficos a partir do pacote `lattice` e do pacote `ggplot2`.

6.1 Gráficos básicos

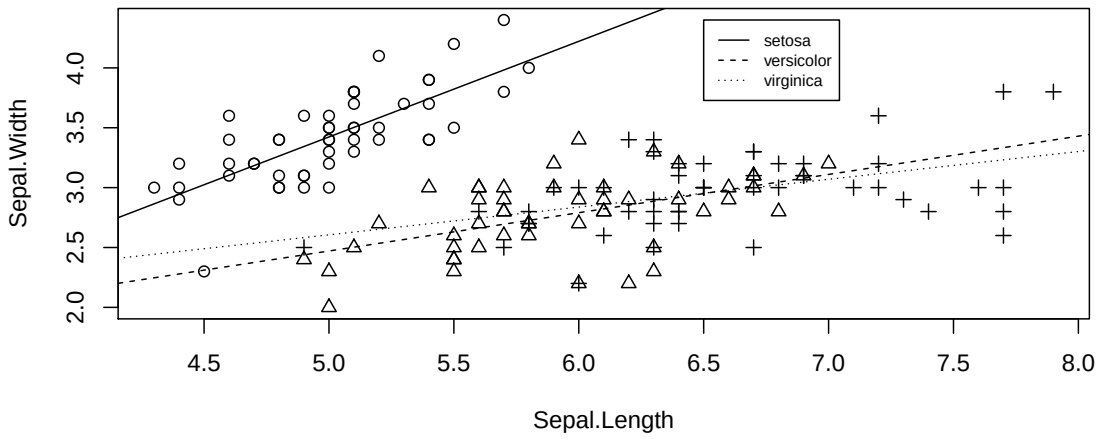
Para ilustrar a construção de gráficos básicos, vamos utilizar o conjunto de dados `iris`. O código abaixo cria um gráfico de correlação entre duas das variáveis do *dataset*, diferenciando ainda pelas três espécies de iris.

```
with(iris, plot(Sepal.Length, Sepal.Width,
                pch=as.numeric(Species), cex=1.2))
legend(6.1, 4.4, c('setosa', 'versicolor', 'virginica'),
       cex=0.9, pch=1:3)
```



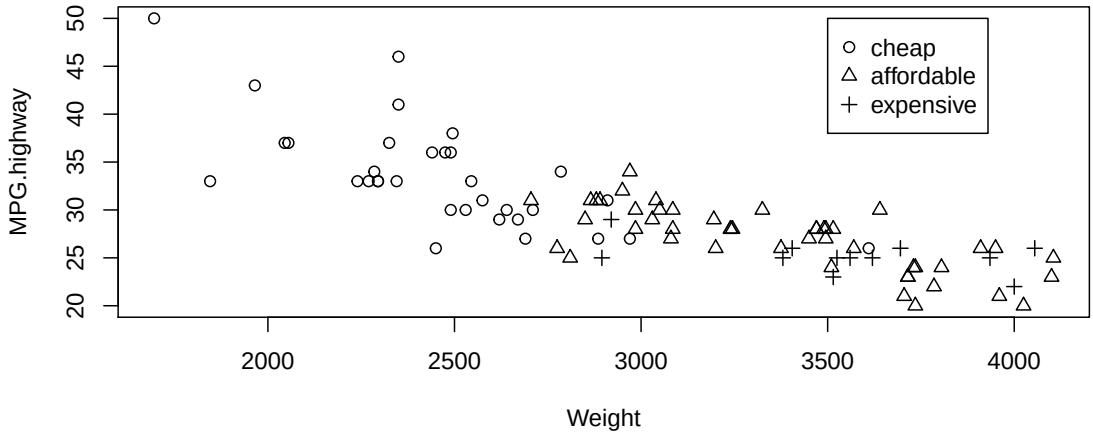
Isto é, além de plotar os pares (x, y), o código acima utiliza o argumento `pch=as.numeric(Species)` de modo a forçar diferentes formatos para cada uma das três espécies. A ideia aqui é adicionar uma terceiro variável ao gráfico, que neste caso seriam as variáveis categóricas representadas pelas espécies. Ademais, é adicionada uma legenda ao gráfico, de acordo com as três espécies. Podemos aumentar a complexidade do nosso gráfico ao adicionar uma *reta de regressão*:

```
fm = Sepal.Width ~ Sepal.Length
plot(fm, iris, pch=as.numeric(Species))
out = mapply(function(i, x) abline(lm(fm, data=x), lty=i),
             i=1:3, x=split(iris, iris$Species))
legend(6.5, 4.4, levels(iris$Species), cex=0.7, lty=1:3)
```



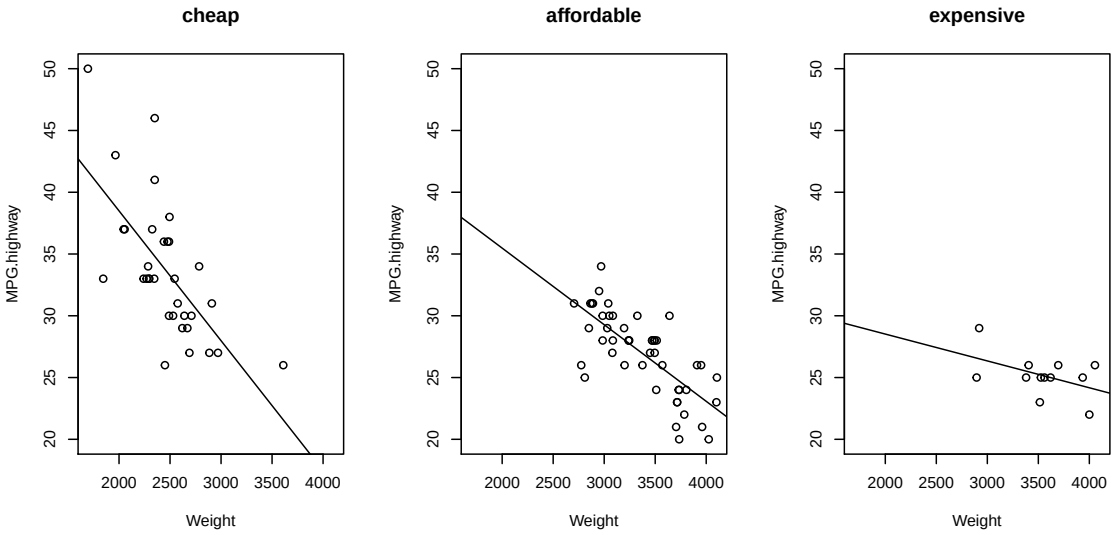
O gráfico torna visível a diferença na relação para a espécie *setosa*, dado que a inclinação é bastante distinta. Ainda nessa linha, a propósito, variáveis numéricas podem ser transformadas em variáveis categóricas para propósitos de agrupamento. Para ilustrar, vamos utilizar abaixo o conjunto de dados `Cars93`.

```
library(UsingR)
Cars93 = transform(Cars93, price=cut(Price, c(0,15,30,75),
                                     labels=c('cheap', 'affordable',
                                              'expensive')))
plot(MPG.highway ~ Weight, Cars93, pch=as.numeric(price))
legend(3500, 50, levels(Cars93$price), pch=1:3)
```



O problema do gráfico acima é que fica difícil identificar a tendência dos diferentes grupos. Isso pode ser resolvido se plotamos um gráfico para cada um. O código abaixo operacionaliza.

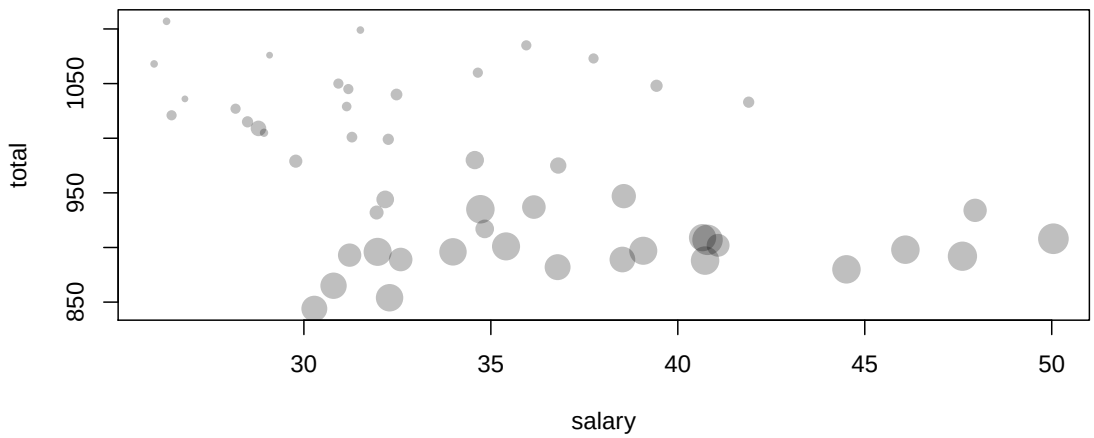
```
l = split(Cars93, Cars93$price)
par(mfrow=c(1, length(l)))
fm = MPG.highway ~ Weight
xlim = range(Cars93$Weight)
ylim = range(Cars93$MPG.highway)
mapply(function(x, nm) {
  plot(fm, data=x, main=nm, xlim=xlim, ylim=ylim)
  abline(lm(fm, data=x))
}, l, names(l))
```



6.1.1 Gráfico de bolhas

Vamos mostrar agora uma outra forma bastante comum de inserir uma terceira variável numérica em um gráfico. Para isso, vamos utilizar o conjunto de dados SAT no código abaixo.

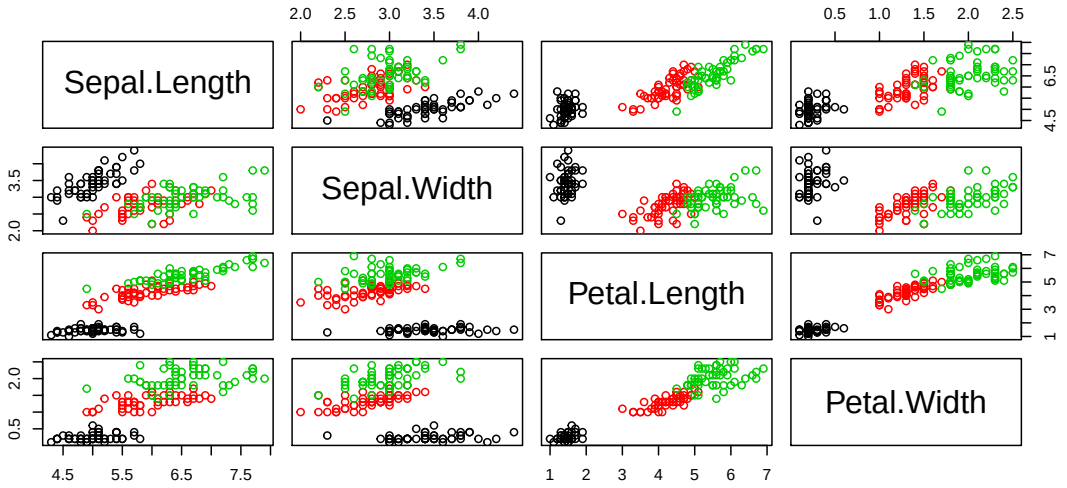
```
plot(total ~ salary, data=SAT, cex=sqrt(perc/10),
     pch=16,
     col=rgb(red=0, green=0, blue=0, alpha=0.250))
```



6.1.2 *Pairs plots*

Um *pairs plot* mostra gráficos de correlação para cada *pair* de uma lista de variáveis numéricas. Por exemplo,

```
species = iris$Species
values = Filter(is.numeric, iris)
pairs(values, col=species)
```



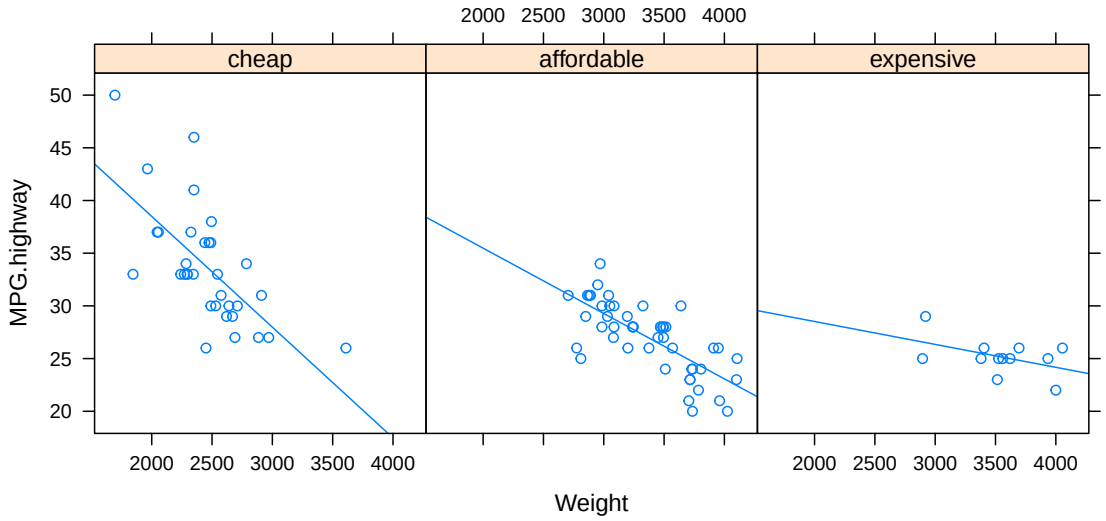
6.2 Gráficos `lattice`

O pacote `lattice` é um poderoso e elegante sistema de visualização de dados multivariados. A elegância provém do uso de fórmulas de R de modo a especificar dados *splitados*. A fórmula básica é extendida de momo a incluir variáveis condicionais como

$$y \quad \sim \quad x \quad | \quad g1 + g2 + ...$$

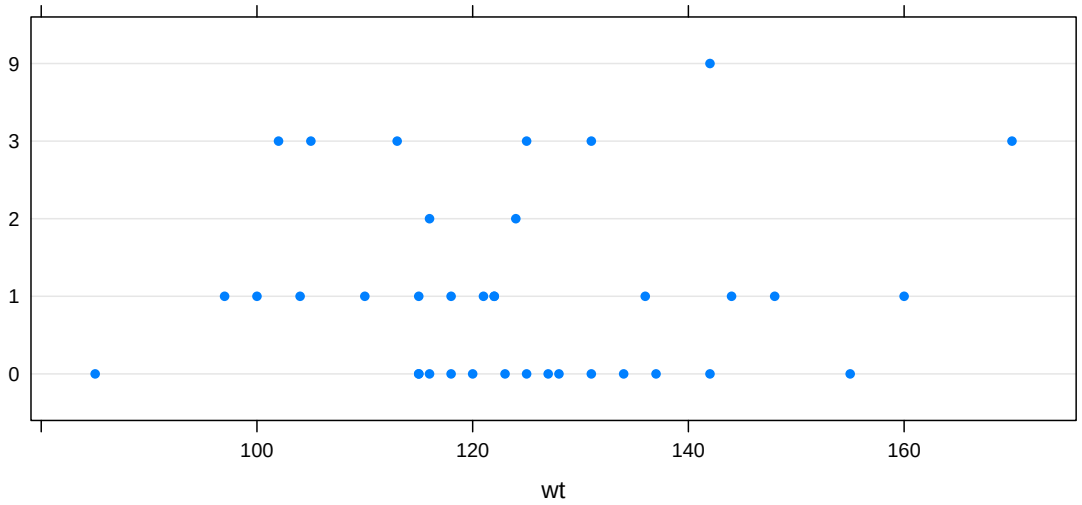
de modo que as variáveis condicionais $g1, g2, ...$ são utilizadas para dividir os dados. Por exemplo, o gráfico que fizemos acima pode ser facilmente replicado com o código abaixo.

```
require(lattice)
Cars93 = transform(Cars93, price=cut(Price, c(0,15,30,75),
                                     labels=c('cheap', 'affordable',
                                              'expensive')))
xyplot(MPG.highway ~ Weight | price, data=Cars93,
       layout=c(3,1), type=c('p', 'r'))
```



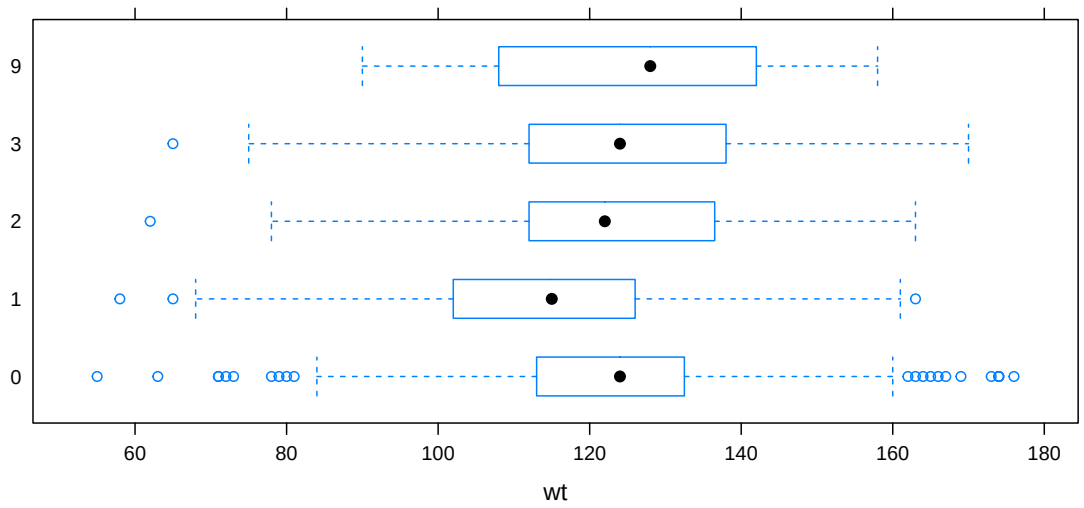
6.2.1 *Dot charts*

```
dotplot(factor(smoke) ~ wt, data=babies, subset=wt<999 & ded==3)
```



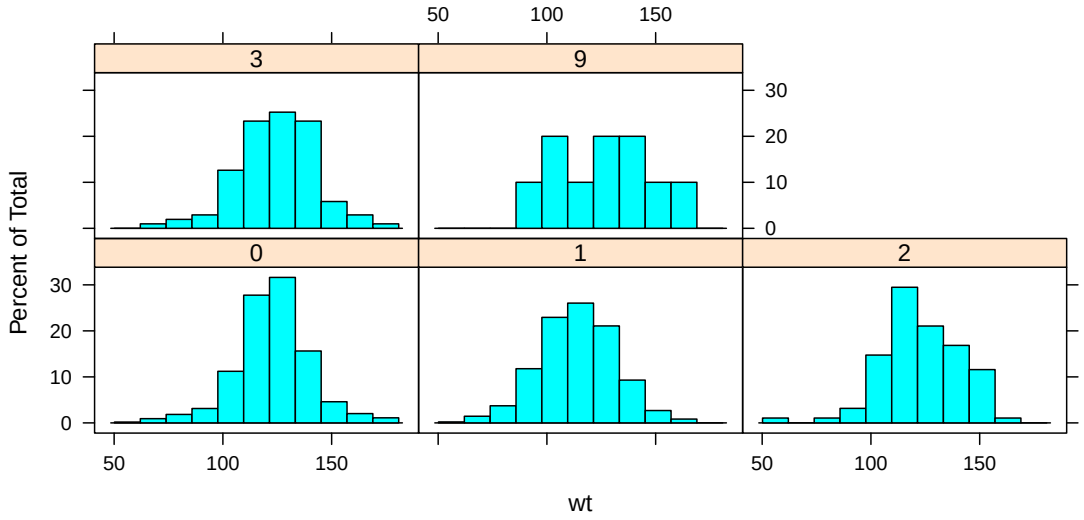
6.2.2 *Boxplots*

```
bwplot(factor(smoke) ~ wt, data=babies, subset=wt<999)
```



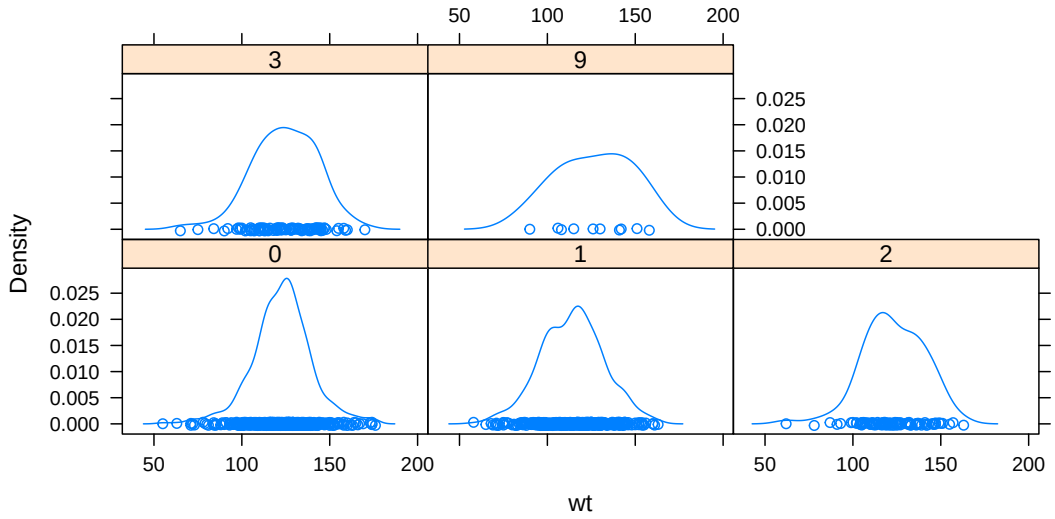
6.2.3 Histogramas

```
histogram(~wt|factor(smoke), data=babies, subset=wt<999)
```



6.2.4 Gráficos de densidade

```
densityplot(~wt|factor(smoke), data=babies, subset=wt<999)
```



6.3 Gráficos com `ggplot2`

O pacote `ggplot2` utiliza a *gramática dos gráficos* de Wilkinson, de modo que um gráfico é o resultado da combinação de diversos componentes.[13] Vamos introduzir o pacote com um exemplo detalhado.

Aesthetics ou *Atributos Estéticos*: Começamos definindo um objeto `ggplot` para um conjunto de dados.

```
require(ggplot2)
p = ggplot(Cars93)
```

Criamos acima apenas um objeto, nenhum gráfico ainda é gerado. Para fazê-lo, precisamos de dois componentes principais: **aesthetics** e **geometries** - objetos geométricos. Aesthetics mapeiam as variáveis em um conjunto de dados de modo que suas propriedades possam ser colocadas em um gráfico. Por exemplo, tamanho, forma e cor são

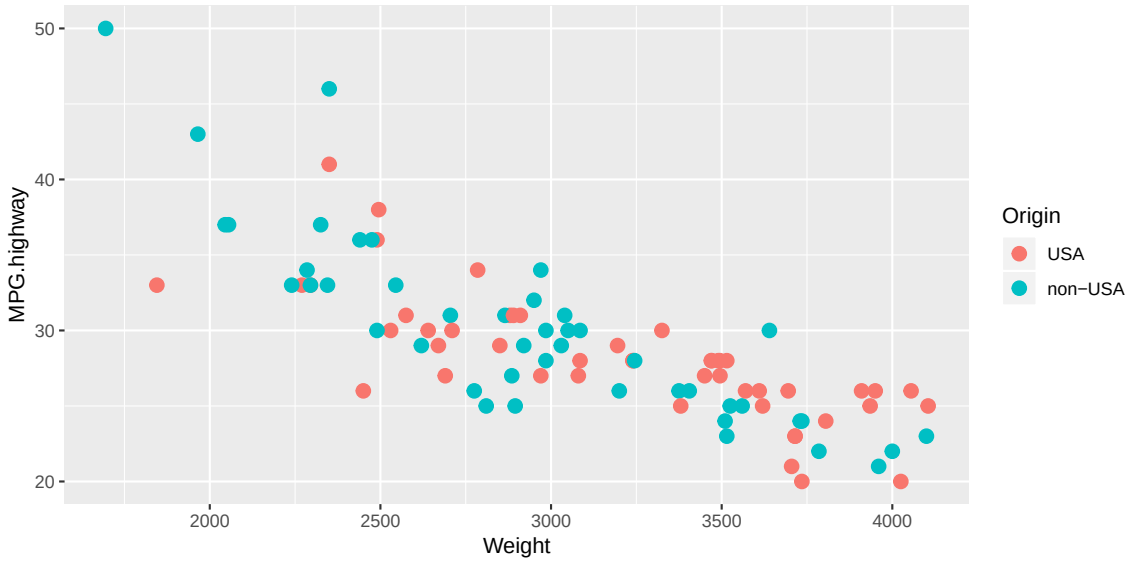
[13] Sobre a gramática dos gráficos, ver Wilkinson (2005). Para uma introdução completo ao pacote, ver Wickham (2009).

aesthetics. Ademais, valores para x e y também serão aesthetics. Abaixo declaramos.

```
p = p + aes(x=Weight, y= MPG.highway, color=Origin)
```

Geoms: uma vez definidos os atributos estéticos, precisamos definir o objeto geométrico que estamos interessados. Com efeito, já temos um gráfico para mostrar.

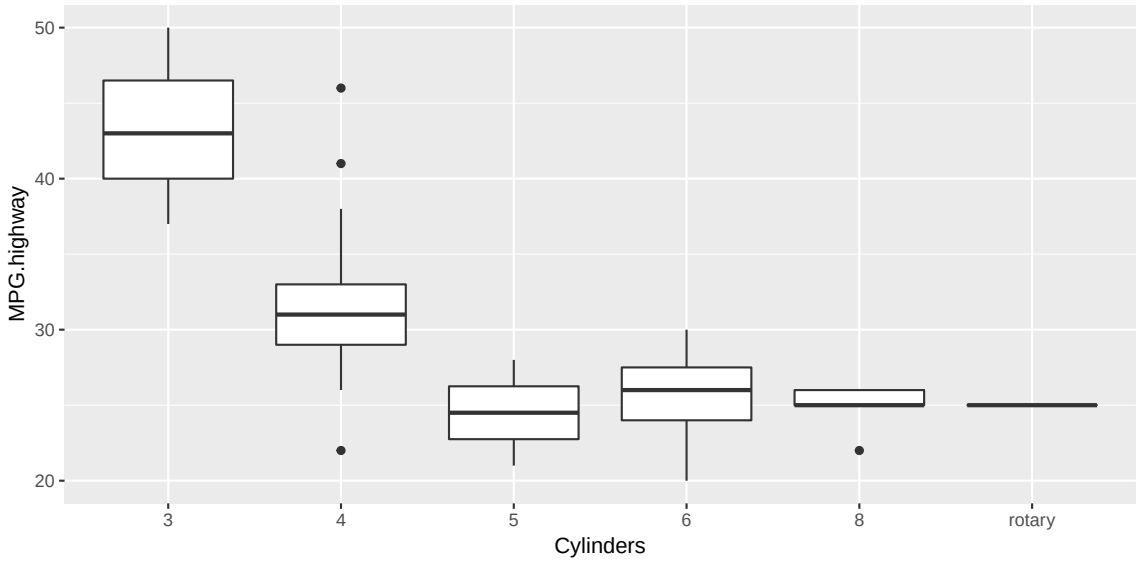
```
p + geom_point(cex=3)
```



6.3.1 Grouping

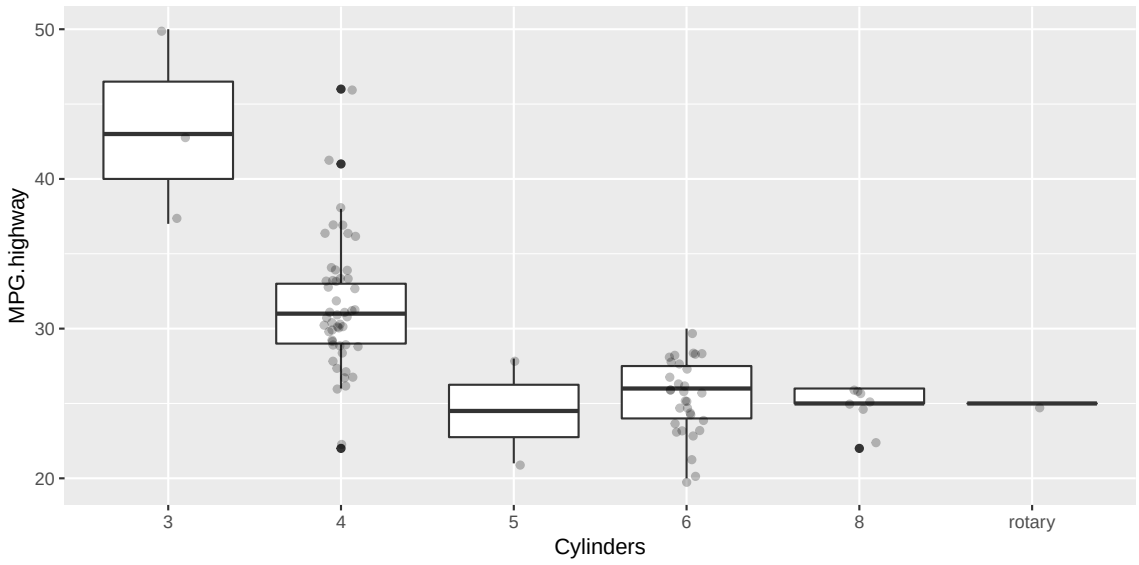
O objeto geométrico `geom_point` mapeia cada ponto de dados em uma representação gráfica. Esse, entretanto, não é o caso para gráficos que reduzem a visualização de dados, tal como histogramas ou boxplots. Para esses, os objetos correspondem a grupos de dados. Abaixo um exemplo.

```
p = ggplot(Cars93, aes(x=Cylinders, y=MPG.highway))
p + geom_boxplot()
```



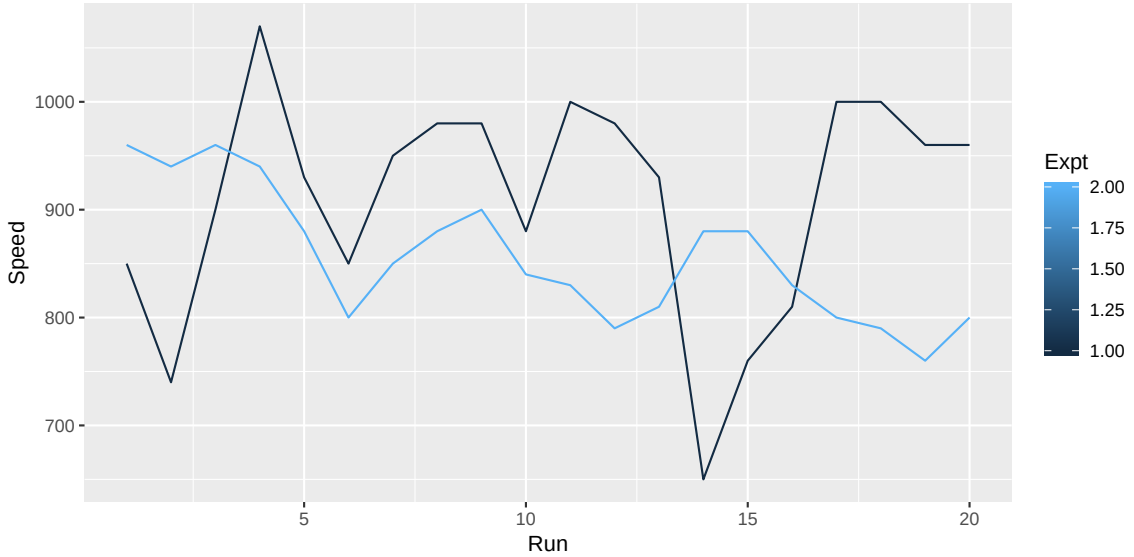
Ou

```
p + geom_boxplot() +
  geom_jitter(position=position_jitter(w=0.1), alpha=.25)
```



Dados longitudinais, por suposto, podem gerar uma situação onde o agrupamento padrão pode não ser suficiente. O conjunto de dados `morley`, por exemplo, guarda 100 diferentes medidas de velocidade da luz. Há 20 rodadas para cada um dos cinco experimentos. Nós podemos então traçar as medidas de cada rodada ao plotar um gráfico de linha com `Run` no eixo x e `Speed` no eixo y. Nós podemos então construir o gráfico por diferentes grupos de acordo com a variável `Expt`. Veja abaixo.

```
m = subset(morley, Expt %in% 1:2) # apenas os dois primeiros exper
p = ggplot(m, aes(x=Run, y=Speed, group=Expt, color=Expt))
p + geom_line()
```

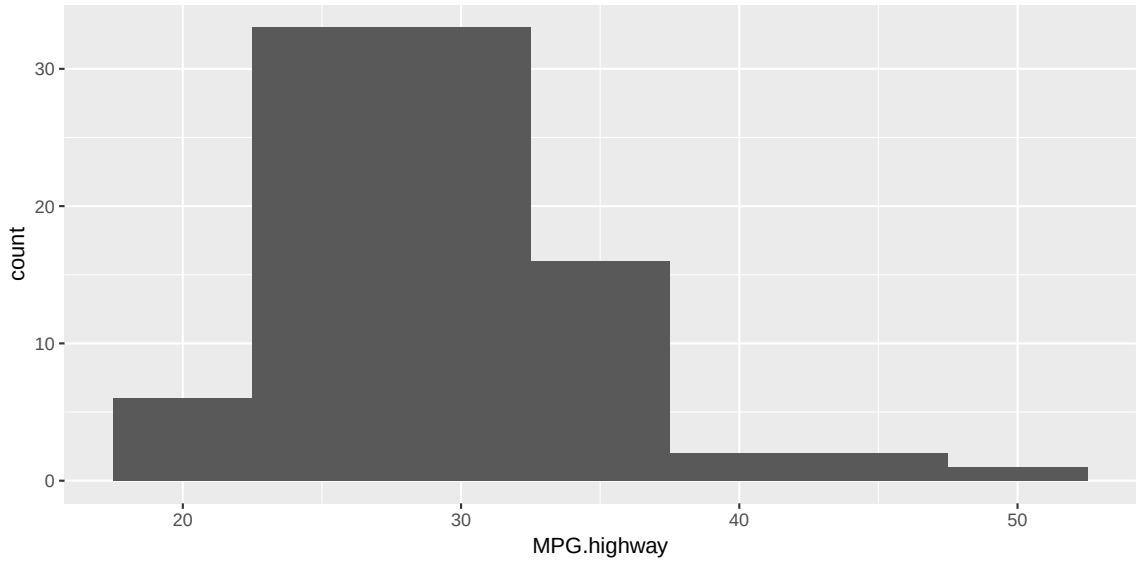


6.3.2 Transformações estatísticas

Estatísticas sumarizam os dados, usualmente reduzindo sua dimensão. Por exemplo, a média sumariza um conjunto de dados com n valores para um único número.

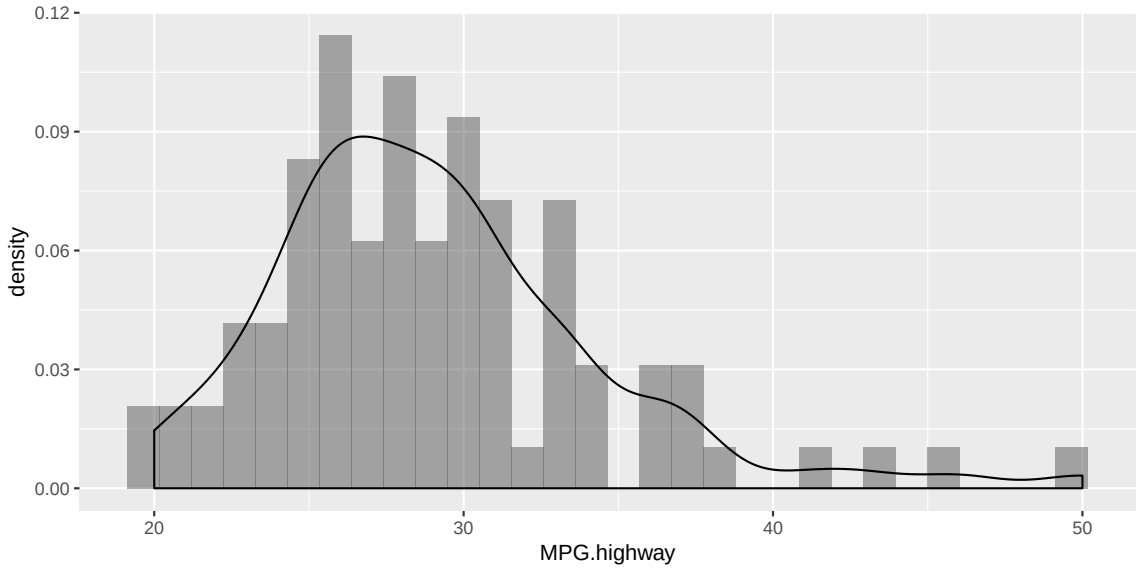
stat_bin: Na gramática do `ggplot2`, transformações estatísticas (**stats**) sumarizam os dados antes deles serem processados. Abaixo um exemplo.

```
p = ggplot(Cars93, aes(x=MPG.highway))
p + stat_bin(binwidth = 5)
```



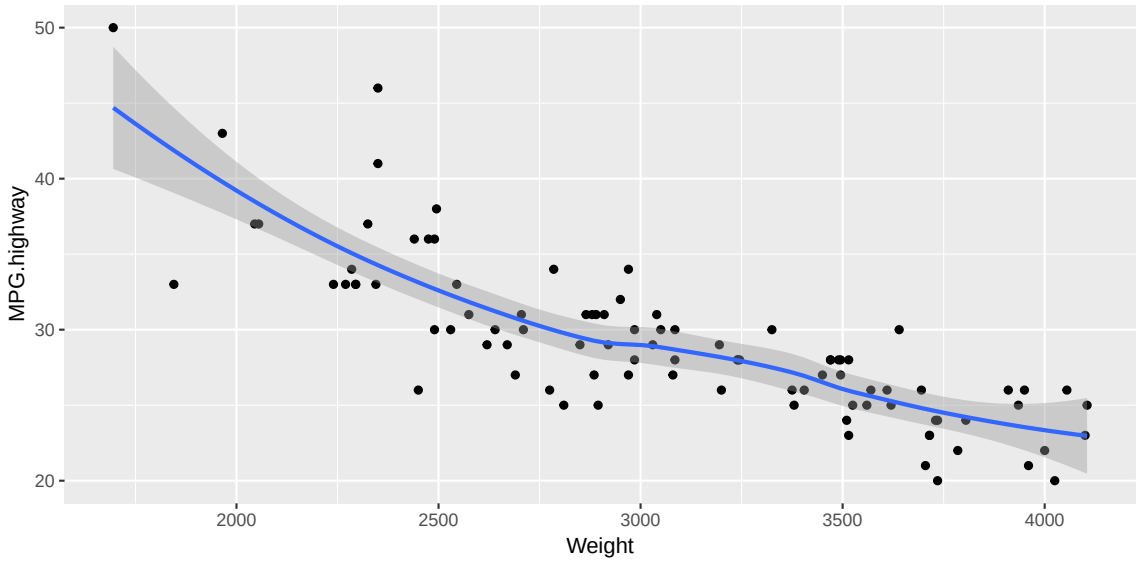
stat_density: Gráficos de densidade são feitos de maneira similar.

```
p = ggplot(Cars93, aes(x=MPG.highway, y=..density..))
p + geom_histogram(alpha=.5)+geom_density()
```



stat_smooth: Uma linha de tendência é um sumário estatístico de uma relação bivariada. Abaixo um exemplo com o `ggplot2`.

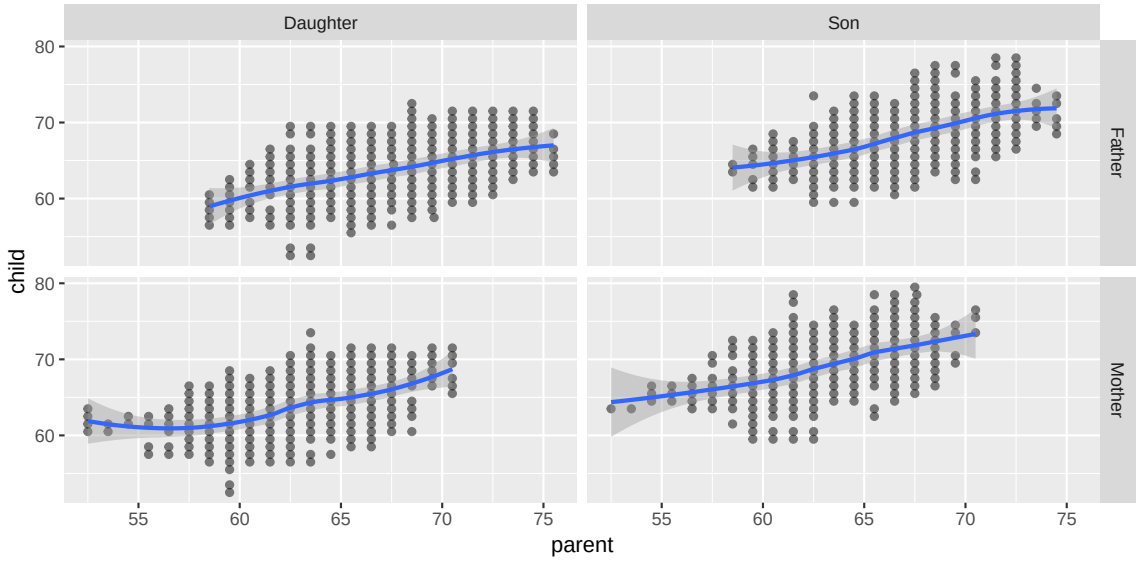
```
p = ggplot(Cars93, aes(x=Weight, y=MPG.highway))
p + geom_point() + geom_smooth()
```



6.3.3 Faceting

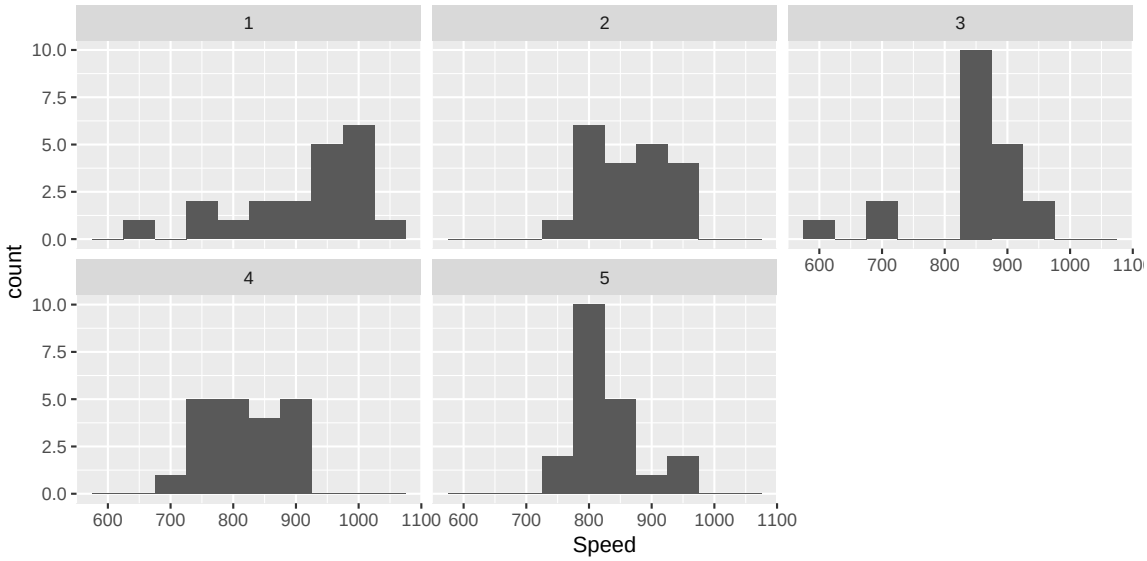
No pacote `lattice` podemos fazer comparações através de grupos por meio de painéis. No pacote `ggplot2`, essa facilidade está implementada através de *facets*. Há dois tipos: `facet_grid` e `facet_wrap`. Abaixo, exemplos.

```
p = ggplot(PearsonLee, aes(y=child, x=parent))
p + geom_point(alpha=.5) + geom_smooth(method='loess') +
  facet_grid(par ~ chl)
```



Ou,

```
p = ggplot(morley, aes(x=Speed)) + geom_histogram(binwidth=50)
p + facet_wrap(~Expt)
```



7 Populações

Nas seções anteriores do nosso Curso de Formação Cientista de Dados, vimos uma introdução à exploração e tratamento de dados. A partir dessa seção, faremos uma iniciação ao processo de inferência estatística propriamente dito. **Inferência estatística**, em termos simplificados, é o processo de formar julgamento sobre uma determinada população com base em uma amostra dessa população. Aqui, nós descrevemos populações e amostras de populações usando a linguagem da probabilidade.[14]

De modo a fazer inferência estatística com base em dados nós utilizamos um **modelo probabilístico** para os dados. E aqui devemos diferenciá-lo de um **modelo determínistico**. Modelos determinísticos são aqueles que estipulam que as condições sob as quais um experimento seja executado determinam o resultado do experimento. Por exemplo, se introduzirmos uma bateria em um circuito simples, o modelo matemático que descreveria o fluxo de corrente elétrica seria presumivelmente a *lei de Ohm*, de modo que para obter a intensidade de corrente elétrica, basta que tenhamos a tensão e a resistência elétrica.

Considere, por outro lado, um conjunto de dados univariado que consiste em medidas de alguma variável. Um ponto qualquer desse conjunto será a realização de um intervalo de valores. Chamaremos assim a *população* de um variável como sendo a descrição do intervalo de possibilidades para esse valor. Utilizaremos o termo *variável aleatória* para descrever o número aleatório de uma população. Assim, um ponto dentro daquele conjunto de dados nada mais é que a realização de uma determinada variável aleatória. Importante dizer que fazemos uma distinção entre quando nós temos uma variável aleatória observada ou realizada. Uma vez observada, o valor da variável aleatória é conhecido. Antes de ser observada, contudo, ela pode ser qualquer valor dentro daquele intervalo possível da população. Para a maioria dos casos, nem todos os valores da população possuem a mesma *probabilidade* de ocorrerem. Assim, antes de observar uma variável aleatória, nós precisaremos indicar a probabilidade dela exercer um determinado valor ou um intervalo de

[14] Essa seção procura introduzir o tema de *inferência estatística* de modo despreocupado, baseado sobretudo em Verzani (2014). Para uma leitura intermediária do assunto, ver Meyer (2011), que tem uma boa introdução à probabilidade. Para uma leitura completa, ver Casella and Berger (2016).

valores. Nós nos referimos a essa descrição de intervalo e suas respectivas probabilidades como *distribuição de uma variável aleatória*.

Por probabilidade, nós queremos dizer algum número entre 0 e 1 que descreve a possibilidade da nossa variável aleatória assumir algum valor. A intuição aqui vem da necessidade de se entender como os números são gerados. Por exemplo, quando jogamos uma moeda para o ar, a probabilidade de dar *cara* ou *coroa* é de 50% para cada. Para situações onde as possibilidades são igualmente prováveis e o total de possibilidades é finito, a definição de probabilidade é dada por

$$P(E) = \frac{\text{Eventos em E}}{\text{Total de Eventos}}. \tag{3}$$

Para essa definição, as seguintes regras serão satisfeitas:

- $P(E) \geq 0$ para todos os eventos;
- O evento de todos os possíveis resultados tem probabilidade igual a 1;
- Se A e B são dois eventos disjuntos, então $P(AouB) = P(A) + P(B)$.

Uma consequência matemática disso é que para qualquer dois eventos, $P(AouB) = P(A) + P(B) - P(AeB)$.

Seleção aleatória

A tabela a seguir mostra a relação entre gênero e fumo para uma coorte no conjunto de dados `survey` do pacote `MASS`:

```
data(survey, package='MASS')
tbl = xtabs(~Sex + Smoke, data = survey)
tbl
##          Smoke
## Sex      Heavy Never Occas Regul
##   Female     5    99     9     5
##   Male       6    89    10    12
```

Existem 237 participantes no estudo, mas apenas 235 estão representados acima. Se nós selecionarmos um deles *de forma aleatória*, a probabilidade de selecionarmos uma mulher será o número de mulheres dividido pelo número total de pessoas representadas. Isto é,

```r
margin.table(tbl, margin=1)
## Sex
## Female   Male
##    118    117
sum(tbl[1,])/sum(tbl)
## [1] 0.5021277
```

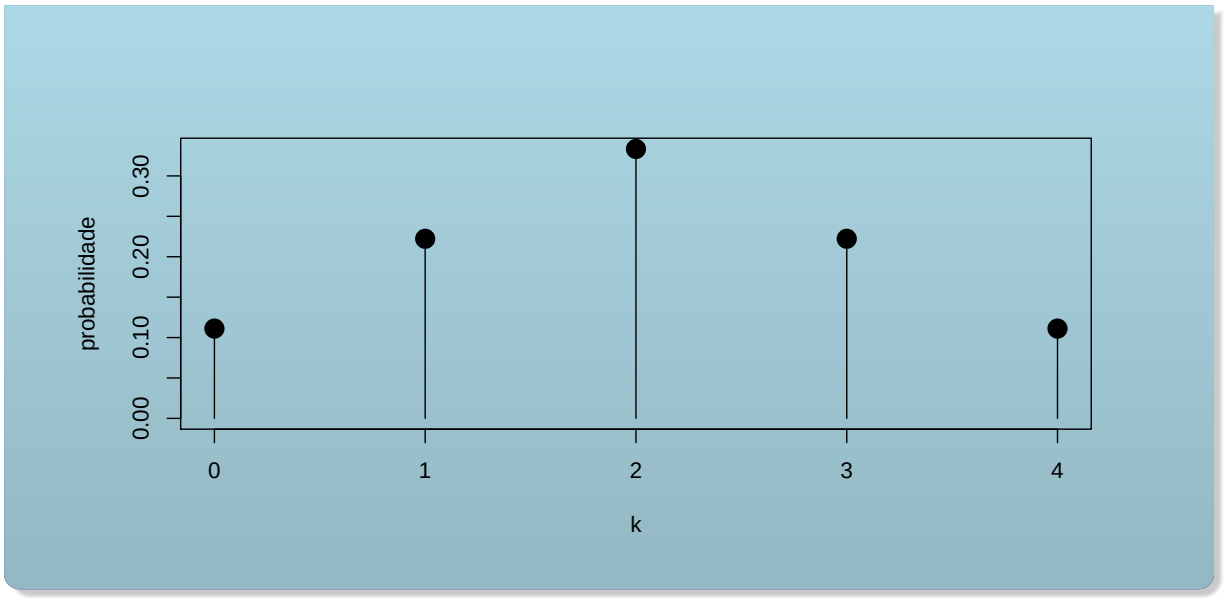
7.1 Variáveis aleatórias discretas

Dados numéricos podem ser discretos ou contínuos, de modo que podemos ter igualmente variáveis aleatórias discretas ou contínuas. Deixemos que X seja uma variável aleatória discreta. Isto é, uma variável aleatória cujos possíveis resultados sejam algo como $\{sim, no\}$ ou $\{0, 1, 2, ...\}$. A extensão de X será o conjunto de todos os k, onde $P(X = k) > 0$. A distribuição de X será uma especificação dessas probabilidades. As regras da probabilidade implicam que distribuições não são arbitrárias, tal que para cada k no intervalo, $P(X = k) > 0$ e $P(X = k) \leq 1$. Ademais, como X tem algum valor, nós teremos $\sum_k P(X = k) = 1$.

Especificando uma distribuição

Nós podemos especificar a distribuição de uma variável aleatória discreta primeiro especificando o intervalo de valores e depois associando a cada k um número $p_k = P(X = k)$, de modo que $\sum p_k = 1$ e $p_k \leq 0$.

```r
k = 0:4
p = c(1,2,3,2,1); p = p/sum(p)
plot(k, p, type='h', xlab='k', ylab='probabilidade', ylim=c(0,max(p)))
points(k,p,pch=16,cex=2)
```



7.1.1 Gerando valores aleatórios

O R possui a função `sample` de modo a gerar observações para uma variável aleatória discreta com uma distribuição específica. Se o vetor k contém os valores amostrados, e p contém as probabilidades dos valores selecionados, assim a função `sample` irá selecionar um dos k valores com a probabilidade especificada por p, como abaixo.

```
k = 0:2
p = c(1,2,1); p = p/sum(p)
sample(k, size=1, prob=p)

## [1] 1
```

7.1.2 A média e o desvio-padrão

Para um determinado conjunto de dados, a média e o desvio-padrão são sumários do centro e da amplitude. Para variáveis aleatórias esses conceitos se transferem, de modo que suas definições são diferentes.

A *média populacional* será denotada por μ. Se X for uma variável aleatória com essa população, a média será também chamada de o *valor esperado de* X, de modo que

$$\mu = E(X) = \sum kP(X = k).$$

Isto é, a média ponderada dos valores no intervalo de X com pesos $p_k = P(X = k)$. Já o *desvio-padrão populacional* será denotado por σ. O desvio-padrão será a raiz quadrada da variância. Se X for uma variável aleatória discreta, sua variância será definida por $\sigma^2 = VAR(X) = E((X - \sigma)^2)$.

7.2 Variáveis aleatórias contínuas

Dados contínuos são modelados por variáveis aleatórias contínuas. Devido aos valores possíveis serem contínuos, uma nova forma de definir probabilidades deve ser utilizada. Ao invés de tentar especificar $P(X = k)$ para todo k, defini-se $P(a < X \leq b)$. Isto pode ser feito por meio de uma função, $F(b) = P(X \leq b)$ ou por meio de uma função relacionada $f(x)$ que possui $P(a < X \leq b)$ igualando a área abaixo do gráfico de $f(x)$, entre a e b. Para uma dada variável aleatória X, a função $f(x)$ é referida como a *densidade* de X.

7.2.1 f.d.p. e f.d.a.

Para uma variável aleatória discreta é comum definir a função $f(k)$ por $f(k) = P(X = k)$. De maneira similar, para uma variável aleatória contínua X, é comum definir a densidade de X por $f(x)$. Em ambos os casos, elas serão chamadas de f.d.p. No caso discreto, significa *função de distribuição de probabilidade*, enquanto no caso contínuo significa *função densidade de probabilidade*. A função de distribuição acumulada, por seu turno, será $F(b) = P(X \leq b)$. No caso discreto, será dada por $\sum_{k \leq b} P(X = k)$; no caso contínuo, será a área à esquerda de b sob a densidade $f(x)$.

7.2.2 A média e o desvio-padrão

Os conceitos de média e desvio-padrão também se aplicam a variáveis aleatórias contínuas, embora suas definições requeiram cálculo. A noção intuitiva para a média de X é que esta será o ponto de equilíbrio para a densidade de X. Ficam válidas as mesmas letras. Se X tem, por exemplo, uma distribuição uniforme sobre $[0, 1]$, a média será $\frac{1}{2}$ e o desvio-padrão será aproximadamente 0.289.

7.3 Amostragem de uma população

Nosso modelo de probabilidade para um ponto do conjunto de dados será uma observação de uma variável aleatória cuja distribuição é descrita pela população parental. Para performar inferência estatística sobre uma população parental, precisamos de uma *amostra* da população. Isto é, uma sequência de variáveis aleatórias $X_1, X_2, ..., X_n$. Uma sequência será *identicamente distribuída* se cada variável aleatória possuir a mesma distribuição, obviamente. A sequência será independente se ao conhecer o valor de alguma das variáveis aleatórias não implicar em nenhuma informação adicional sobre a distribuição das outras. Uma sequência que é independente e identicamente distribuída (i.i.d.) é chamada de amostra aleatória.

Para ilustrar, considere jogar uma moeda para o alto n vezes. Se dissermos que X_i é 1 para uma possibilidade e 0, caso contrário, então claramente $X_1, X_2, X_3, ..., X_n$ será uma sequência *i.i.d.*. De modo geral, se nós geramos nossos números aleatórios por meio de uma seleção aleatória a partir de uma população finita, então os valores serão independentes se a amostragem for feita com reposição. Por outro lado, caso não tenhamos reposição, as variáveis aleatórias $X_1, X_2, X_3, ..., X_n$ ainda terão a mesma distribuição, mas serão dependentes.

Amostras aleatórias com a função `sample`

A função `sample` tomará amostras de tamanho n de uma distribuição discreta com a especificação `size = n`. Abaixo um exemplo.

```
sample(0:1, size=10, replace=T)
##  [1] 0 0 1 1 1 0 0 0 0 0
```

7.4 Distribuições de amostragem

Uma *estatística* é um valor numérico que sumariza uma amostra aleatória. Um exemplo disso é a média amostral $\bar{X} = \sum_{i=1}^{n} \frac{X_n}{n}$. Quando uma estatística depende de uma amostra aleatória, ela também será uma variável aleatória. Daí a descrição $\bar{X}$. A distribuição de uma estatística é chamada de distribuição de amostragem.

7.5 Famílias de Distribuições

Em estatística existem distribuições que se apresentam em famílias. Cada família é descrita por meio de uma função que possui um número de parâmetros que caracterizam a distribuição. Por exemplo, a distribuição uniforme é uma distribuição contínua no intervalo $[a, b]$ que atribui probabilidade igual para áreas de tamanho igual no intervalo. Os parâmetros a e b serão os *limites* desse intervalo. A densidade e a população de uma família de distribuições são em geral representadas em termos desses parâmetros.

7.5.1 As funções d, p, q e r

O R possui quatro tipos de funções para tomar informação sobre uma família de distribuições:

- A função d retorna a *f.d.p.* da distribuição;
- A função p retorna a *f.d.a.* da distribuição;
- a função q retorna os *quantis*;
- a função r retorna amostras aleatórias de uma distribuição.

Essas funções são usadas de forma similar. Cada família tem um nome e alguns parâmetros. O nome da função é encontrado pela combinação de d, p, q ou r com o nome da família. Os nomes dos parâmetros variam de família para família mas são consistentes com a família.

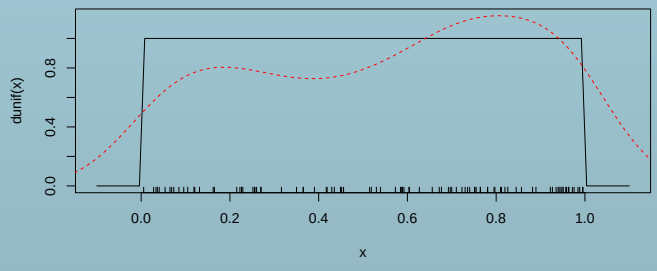
Por exemplo, a distribuição uniforme sobre $[a, b]$ tem dois parâmetros. O nome da família é unif. No R os parâmetros são nomeados como `min` e `max`. Abaixo um exemplo para a distribuição uniforme sobre $[0, 3]$.

```
dunif(x=1, min=0, max=3)

## [1] 0.3333333

punif(q=2, min=0, max=3)

## [1] 0.6666667

qunif(p=1/2, min=0, max=3)

## [1] 1.5

runif(n=1, min=0, max=3)

## [1] 1.84567
```

Relacionando as funções d e r

Para distribuições contínuas, a função d descreve uma densidade teórica. Vamos abaixo relacioná-la com a função r.

```
x = runif(100)
d = density(x)
curve(dunif, -0.1,1.1,ylim=c(0,max(d$y,1)))
lines(d, lty=2, col='red')
rug(x)
```



7.5.2 Variáveis aleatórias de Bernoulli

Uma variável aleatória de *Bernoulli* X é uma que possui apenas dois valores: 0 e 1. A distribuição de X é caracterizada por $p = P(X = 1)$. Se utiliza `Bernoulli(p)` para se referir a essa distribuição. Em geral, atribui-se *sucesso* quando $X = 1$ e *fracasso* quando $X = 0$. Se jogarmos uma moeda e deixemos que X seja 1 caso dê cara, então X será uma variável aleatória Bernoulli onde o valor de p seria $\frac{1}{2}$ se a moeda for honesta. Uma sequência de moedas jogadas seria uma sequência *i.i.d.* de variáveis aleatórias de Bernoulli, também conhecida como *processo ou ensaio de Bernoulli*. Uma variável aleatória de Bernoulli tem média $\mu = p$ e variância $\sigma^2 = p(1 - p)$. No R, a função `sample` pode ser utilizada para gerar uma amostra aleatória a partir de uma distribuição de Bernoulli. Abaixo um exemplo.

```
n = 10; p = 1/4
sample(0:1, size=n, replace=TRUE, prob=c(1-p,p))

##  [1] 0 0 1 0 0 0 0 1 1 0
```

7.5.3 Variáveis aleatórias Binomiais

Uma variável aleatória Binomial X conta o número de sucessos em n processos de Bernoulli. Existem dois parâmetros que descrevem a distribuição de X: o número de processos, n, e a probabilidade de sucesso, p. A distribuição é representada por `Binomial(n,p)`. O intervalo possível para X será $0, 1, ..., n$. A distribuição de X será dada por

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

O termo $\binom{n}{k}$ é conhecido como coeficiente binomial e é definido por

$$\binom{n}{k} = \frac{n!}{(n - k)!k!}.$$

A notação padrão $n!$ é para o *fatorial* de n ou simplesmente $n * (n - 1) * * * 2 * 1$. Por convenção, $0! = 1$. O coeficiente binomial conta o número de maneiras que k objetos podem ser escolhidos de n objetos distintos e é lido como *n escolhe k*. A função `choose` resulta no coeficiente binomial. A média de uma variável aleatóri `Binomial(n,p)` será $\mu = np$ e o desvio-padrão será $\sigma = \sqrt{np(1 - p)}$. No R, o nome da família será `binom` e os parâmetros são rotulados como `size = n` e `prob = p`.

Jogando 10 moedas

Joque uma moeda dez vezes para o alto. Deixe que X seja o número de caras. Se a moeda for honesta, X possui uma distribuição `Binomial(10,1/2)`. A probabilidade que $X = 5$ pode ser encontrada diretamente a partir da distribuição com a função `choose`:

```
choose(10,5)*(1/2)^5*(1/2)^(10-5)
## [1] 0.2460938
```

Isso, a propósito, pode ser melhor representado usando a função d, isto é, `dbinom`:

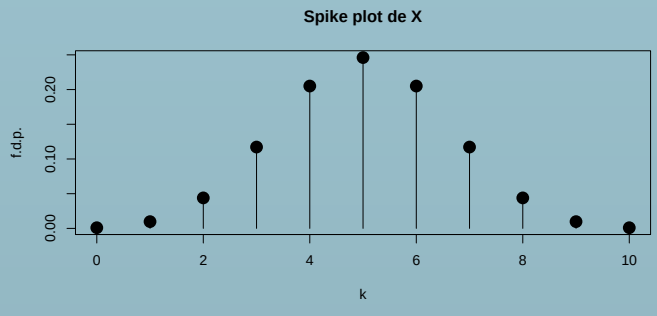
```
dbinom(5, size=10, prob=1/2)
## [1] 0.2460938
```

A probabilidade de que seja seis ou menos caras, $P(X \leq 6) = \sum_{k \leq 6} P(X = k)$, pode ser dada de duas maneiras:

```
sum(dbinom(0:6, size = 10, prob=1/2))
## [1] 0.828125
pbinom(6, size=10, p=1/2)
## [1] 0.828125
```

Um gráfico da distribuição pode ser gerado usando `dbinom`:

```
n = 10; p = 1/2
heights = dbinom(0:10, size=n, prob=p)
plot(0:10, heights, type='h',
     main='Spike plot de X', xlab='k', ylab='f.d.p.')
points(0:10, heights, pch=16, cex=2)
```



7.5.4 Variável aleatória Normal

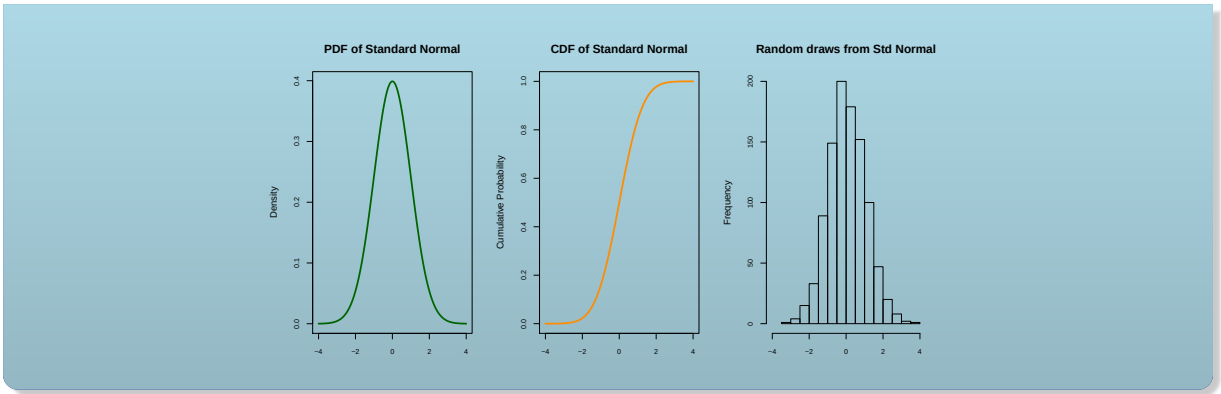
A distribuição normal é uma distribuição contínua que dá sentido à expressão *em forma de sino*. É utilizada para descrever diversas populações na natureza, tal qual a distribuição de alturas, e adicionalmente descreve a distribuição de amostragem de diferentes estatísticas. A distribuição normal é uma família de distribuições com densidade dada por

$$f(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\epsilon^{-\frac{1}{2\sigma^2}(x-\mu)^2}.$$

Os dois parâmetros são a média, μ, e o desvio padrão, σ. Nós utilizamos `Normal(`μ,σ`)` para representar a distribuição, embora muitos livros utilizem a variância σ^2 para representar o segundo parâmetro. No R, o nome da família é norm e os parâmetros são `mean` e sd.

Um exemplo de distribuição normal

```
set.seed(3000)
xseq<-seq(-4,4,.01)
densities<-dnorm(xseq, 0,1)
cumulative<-pnorm(xseq, 0, 1)
randomdeviates<-rnorm(1000,0,1)
par(mfrow=c(1,3), mar=c(3,4,4,2))
plot(xseq, densities, col="darkgreen",xlab="", ylab="Density",
     type="l",lwd=2, cex=2, main="PDF of Standard Normal",
     cex.axis=.8)
plot(xseq, cumulative, col="darkorange", xlab="",
     ylab="Cumulative Probability",type="l",lwd=2, cex=2,
     main="CDF of Standard Normal", cex.axis=.8)
hist(randomdeviates, main="Random draws from Std Normal",
     cex.axis=.8, xlim=c(-4,4))
```



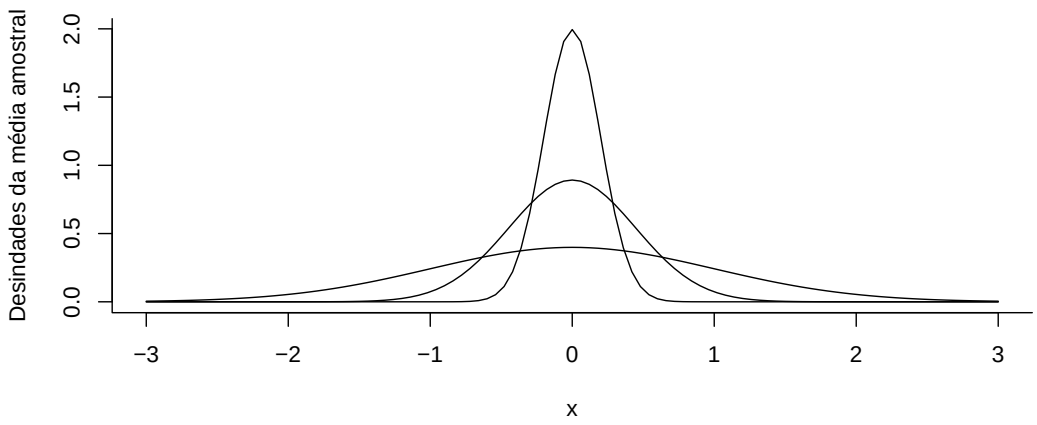
7.6 O Teorema Central do Limite

Para uma amostra *i.i.d.* obtida de uma população, a distribuição da média amostral tem valor esperado μ e desvio-padrão $\frac{\sigma}{\sqrt{n}}$, onde μ e σ são parâmetros populacionais. Para um n *grande o suficiente*, a distribuição de amostragem de $\bar{X}$ será normal ou aproxidamente normal.

7.6.1 População parental Normal

Quando a amostra $X_1, X_2, ..., X_n$ é desenhada a partir de uma população $N(\mu, \sigma)$, a distribuição de $\bar{X}$ será precisamente a distribuição normal. Abaixo, desenhamos densidades para a população e distribuição de amostragem de $\bar{X}$ para $n = 5$ e $n = 25$ quando $\mu = 0$ e $\sigma = 1$.

```
n = 25; curve(dnorm(x, mean=0, sd=1/sqrt(n)), -3, 3,
              xlab='x',
              ylab='Desindades da média amostral', bty='l')
n = 5; curve(dnorm(x, mean=0, sd=1/sqrt(n)), add=TRUE)
n = 1; curve(dnorm(x, mean=0, sd=1/sqrt(n)), add=TRUE)
```



Conquanto o centro permanece o mesmo, a variabilidade de $\bar{X}$ fica menor à medida que n aumenta. Se o tamanho da amostra aumenta por um fator 4, o desvio-padrão será $\frac{1}{2}$ de suas populações. A densidade se concentrará na média. Isto é, com maiores e maiores probabilidades, o valor aleatório de $\bar{X}$ será mais perto da média, μ, da população parental. Esse fênomeno de concentração ao redor da média é conhecido como *lei dos grandes números*.

7.6.2 População parental não-Normal

O *teorema central do limite* diz que para qualquer população parental com média μ e desvio-padrão σ, a distribuição de amostragem de $\bar{X}$ para n grande satisfaz

$$P(\frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \leq b) \approx P(Z \leq b),$$

onde Z é uma variável aleatória normal padrão. Isto é, para n grande o suficiente, a distribuição padronizada de $\bar{X}$ será aproxidamente uma normal padrão.

8 Inferência Estatística

Essa seção do nosso Curso Formação Cientista de Dados foca em conceitos básicos de inferência estatística usando simulação - evitando cálculos probabilísticos - de modo a produzir respostas. Como vimos, inferência estatística é o processo de extrair inferências sobre uma população com base em dados amostrais dessa população. Nesse processo, por suposto, são quatro os conceitos-chaves que devem ser levados em consideração:

- **populações** Nossas populações são modeladas por meio de distribuições que descrevem a aleatoriedade de cada ponto dos dados amostrais;
- **parâmetros** Um parâmetro é um número que descreve a população, tal como a média (μ) ou o desvio-padrão (σ);
- **amostras** Uma amostra é uma coleção de observações da população, onde uma observação é a realização de uma variável aleatória com a distribuição da população. Para nossos propósitos, uma amostra é usualmente assumida como sendo uma amostra aleatória da população, o que implica em independência;
- **estatística** Um sumário numérico da amostra. Por exemplo, a média amostral $\bar{x}$ ou o desvio-padrão amostral s.

A população é descrita pelos parâmetros e é representada pelas amostras. Dado que uma estatística sumariza a amostra, uma questão estatística central é como nós podemos inferir informação sobre os parâmetros a partir de uma estatística?

Ao comparar a densidade interna para a distribuição de $\bar{x}$ com a distribuição teórica de um valor individual, nós podemos ser levados a dizer que a distribuição de $\bar{x}$ está centrada no mesmo lugar que a da população - isto é, a aleatoriedade de uma estatística está centrada na aleatoriedade da população e a distribuição é bem comportada. Entretanto, é evidente que a distribuição de $\bar{x}$ tem uma variabilidade menor do que a da população. Em geral, para a simulação de $\bar{x}$ as seguintes observações podem ser explicitadas:

- Para qualquer amostra aleatória, a distribuição amostral de $\bar{x}$ é centrada na média populacional, μ;
- Para qualquer amostra aleatória, a distribuição amostral de $\bar{x}$ possui um desvio-padrão de $\frac{\sigma}{\sqrt{n}}$ onde σ é o desvio padrão da população;

- Ademais, se a população for normalmente distribuída, a distribuição de $\bar{x}$ também será normalmente distribuída.

O ponto é, dados simulados podem ser utilizados para investigar relacionamentos, de modo a produzir respostas que embora não sejam sempre precisas, podem dar excelentes *insights*. Essa seção, por suposto, cobre os conceitos básicos para performar simulações com o R e posteriormente aplica isso para introduzir a abordagem mais utilizada para construir inferência estatística.

8.1 Simulação

Para nossas simulações, o ponto inicial é especificar um modelo probabilístico para os dados. O R possui, como vimos, funções específicas para produzir amostras aleatórias a partir da distribuição da população. Nssas simulações usualmente envolvem sumarizar uma amostra aleatória com uma estatística. Por exemplo, encontrar uma realização da média amostral de uma amostra aleatória de tamanho 16 obtida de uma população normal.

```
mu = 100; sigma = 16 # parâmetros populacionais
x = rnorm(16, mu, sigma) # nossa amostra
mean(x) # média amostral

## [1] 98.86523
```

O modelo é especificado nesse exemplo pela escolha da família de distribuição (`rnorm`) e pela escolha dos parâmetros. Com isso, o R pode ser utilizado para produir uma amostra aleatória de um tamanho determinado. O valor *mean(x)* é o valor produzido. Ao rodar o código acima repetidas vezes, tudo o que você terá, a propósito, serão números diferentes. Isso traz *insights* interessantes sobre a forma da distribuição, seus mínimos, suas médias, variâncias, probabilidades relacionadas, etc.

Repetindo uma simulação

Repetir uma simulação, a propósito, é a chave para obter insights sobre essa simulação. Há diversas formas de repetir uma expressão no R. Vamos aqui mostrar isso através da função `for`.

```
mu = 100; sigma = 16
M = 10; n = 16
```

```
res = numeric(M)
for (i in 1:M){

  res[i] = mean(rnorm(n, mean=mu, sd=sigma))

}

res
```

```
##  [1] 104.03563 102.75109  93.70237 102.05196 100.36116  98.57477 104.51395
##  [8]  99.14903 107.93251  90.29412
```

Essa é uma forma bastante direta, mas há outras formas. Por exemplo, podemos criar uma função para chamar nossa expressão a criar um único $\bar{x}$ para utilizar com `sapply`:

```
xbar = function(i)
  mean(rnorm(n, mean=mu, sd=sigma))

sapply(1:M, xbar)
```

```
##  [1] 104.21484 102.80604 105.88470  96.37395 103.01796  98.23108 103.81314
##  [8] 104.56875 103.49757  95.75595
```

Ou simplesmente,

```
replicate(M, mean(rnorm(n, mean=mu, sd=sigma)))
```

```
##  [1] 100.27518  98.66045 108.15381 102.64012 105.21034 104.36471 112.54579
##  [8] 100.78001  99.25324  96.20105
```

Com a função `apply`:

```
x = matrix(rnorm(M*n, mean=mu, sd=sigma), nrow=n)
dim(x)
```

```
## [1] 16 10
```

```
apply(x, 2, mean)
```

```
##  [1] 107.76388 102.66261 104.75218 100.91053 102.37037  99.29138 106.64731
##  [8]  97.65304 103.03606 101.28149
```

8.2 Testes de significância

Imagine agora um estudo de algum novo tratamento que melhora a performance. Por exemplo, consumir uma porção de mel durante os exercícios aumenta a performance? De modo a testar esse tipo de tratamento, uma coorte de sete pessoas convenientemente selecionadas é obtida. O pesquisador aleatoriamente atribui a três como grupo de controle e a 4 como grupo de tratamento. O grupo de controle deveria dar uma base de comparação para o grupo de tratamento. Os dados coletadas são alguma medida de modo que valores menores são melhores:

```
controle = c(23, 33, 40)
tratamento = c(19, 22, 25, 26)
data = stack(list(controle=controle, tratamento=tratamento))
aggregate(values~ind, data, mean)

##          ind values
## 1   controle     32
## 2 tratamento     23
```

Uma diferença de 9 é obtida. O pesquisador fica então emocionado. O estudo parece mostrar que uma simples porção de mel pode aumentar a performance. É preciso, contudo, ser cauteloso. Os resultados podem ser derivados simplesmente da aleatoriedade da escolha. Isto é, pessoas que performam melhor podem estar no grupo de tratamento, enquanto que pessoas que naturalmente performam pior pode estar no grupo de controle, de modo que a porção de mel não impactou em nada. A partir da simulação, podemos investigar se é esse o caso. Em particular, nós podemos enumerar todas as possibilidades de diferentes combinações. A função `combn` irá listar então através dos seus índices.

```
cmbs = combn(7, 3) # 35 possibilidades
cmbs[,1:6]

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    1    1    1
## [2,]    2    2    2    2    2    3
## [3,]    3    4    5    6    7    4
```

O primeiro seria apenas a aleatorização observada.

```
i = 1
ind = cmbs[,1]
```

```
obs = mean(data$values[ind] - mean(data$values[-ind]))
obs

## [1] 9
```

O uso da indexação negativa é conveniente nesse caso, dado que estamos parcionando nosso conjunto de 7 elementos em grupos de 3 e 4. De modo a computar as 35 combinações, podemos utilizar a função `apply`.

```
res = apply(cmbs, 2, function(ind) {
    mean(data$values[ind]) - mean(data$values[-ind])
})
```

Os valores em `res` representam a distribuição para a diferença no grupo das médias. Agora que nós temos 35 diferentes valores, nós podemos ver o quão extremo o número 9 é nós podemos contar quantos valores são iguais ou maiores do que ele.

```
sum(res >= obs)

## [1] 3

sum(res >= obs)/length(res)

## [1] 0.08571429
```

Apenas três das 35 atribuições aleatórias ou simples 8.57% irão produzir uma diferença igual ou maior do que 9. Esse valor parece ser, portanto, improvável.

8.3 Estimação e intervalos de confiança

A última subseção procurou verificar se um tratamento induzia melhor performance. Nessa, temos uma questão diferente. Como uma estatística amostral pode produzir uma boa estimativa de um parâmetro?

Uma notícia comum é um relatório sobre uma proporção em uma amostra. Por exemplo, em meados de 2012, seguindo as decisões da Suprema Corte norte-americana sobre a defesa do casamento gay, uma pesquisa de opinião foi feita pela Princeton Survey Research. Os pesquisadores perguntaram para uma amostra aleatória de 1.003 norte-americanos se o casamento gay deveria ser reconhecido como válido, tendo os mesmos atributos do casamento entre pessoas de sexos distintos. O resultado foi que 55% eram a favor da

validade do casamento entre pessoas do mesmo sexo, uma resposta bastante elevada.

O valor 55%, por suposto, representa a amostra, de modo que ele é uma estatística. A implicação é que de alguma maneira esse valor representa um parâmetro - a proporção de todos os norte-americanos. De que modo então essa estatística estima um parâmetro desconhecido?

Novamente, nós começamos com algum modelo probabilístico para os dados. No cenário acima, um modelo simples seria aquele em que cada pessoa escolhida aleatoriamente possui uma probabilidade p de responder sim. Isto é, as variáveis aleatórias possuem uma distribuição `Bernoulli(p)`. Outros cenários podem assumir uma distribuição normal para as variáveis aleatórias que produzem a amostra. Os modelos que consideramos são definidos em meio a uma família e um ou mais parâmetros.

A questão de uma estimação estatística de um parâmetro é, então, respondida por meio da observação de valores tais como $\bar{x}$ em relação μ, ou $\hat{p}$ em relação a $\hat{p}$. Existem várias maneiras de comparar uma relação, uma simples é olhar para suas diferenças.

Vamos focar nas diferenças, por enquanto. O quanto podemos dizer? Se nós tivermos uma amostra grande, a intuição nos diria que a média amostral é uma estimativa melhor para μ do que um valor único. Por que? Ambas possuem um valor esperado - μ. O segredo é a variabilidade. Para um amostra aleatória de tamanho n, $VAR(\bar{x}) = \frac{\sigma^2}{n}$, onde σ^2 é a variância da população.

Deixemos que θ seja algum parâmetro e que $\hat{\theta}$ seja alguma estatística para estimar θ. Olhar para a variabilidade, ou $E(\hat{\theta} - \theta)^2$, é razoável. Para uma variável aleatória, isso pode ser escrito de duas formas

$$E((\hat{\theta} - \theta)^2) = VAR(\hat{\theta}) + \left[E(\hat{\theta} - \theta)^2\right] = \quad \text{variância} + \text{viés}^2. \tag{4}$$

O viés é simplesmente a diferença entre o valor esperado do estimador e do parâmetro. A maioria das estatísticas que nós encontramos como não-viesada, significa que essa diferença é zero. Exemplos são, $E(\bar{x}) = \mu$ e $E(\hat{p}) = p$.

Uma questão mais sutil do que olhar para a expectativa é olhar para a distribuição de amostragem. Para um modelo de população normal, nós vimos que nós podemos simular a distribuição de $\bar{x} - \mu$ da seguinte forma.

```
mu = 100; sigma = 16
M = 1000; n = 4
res = replicate(4, mean(rnorm(n, mu, sigma)) - mu)
```

O desvio-padrão de $\bar{x}$, como vimos, é $\frac{\sigma/\sqrt{nn}}{}$, de modo que se substituírmos σ por s, o que obteremos será o erro-padrão, isto é, o estimador substitui o parâmetro desconhecido. A distribuição depende de n, mas de posse de uma amostra grande o suficiente nós podemos fazer perguntas do tipo: onde está a maioria dos dados? Sendo preciso, qual intervalo de cerca de 0 contém 95% dos dados?

Podemos simular a *estatística t* para responder essa pergunta:

```
mu = 100; sigma = 16
M = 1000; n = 4

res = replicate(M, {
  x = rnorm(n, mu, sigma)
  SE = sd(x)/sqrt(n)      # erro padrão
  (mean(x) - mu)/SE
})
```

De modo a encontrar um intervalo, nós usamos a função `quantile`:

```
quantile(res, c(0.025, 0.975))
```

```
##      2.5%     97.5%
## -2.985613  3.336426
```

Então, basicamente, nós temos com probabilidade aproximada de 95% que

$$-3.05 * SE < \bar{x} - \mu < 3.19 * SE.$$

Ou, em outras palavras, o valor de $\bar{x}$ não é muitos erros-padrão distante da média, μ, com probabilidade elevada. O segredo dessa declaração é comparar distâncias usando uma escala definida pelo erro-padrão.

Quando nós rodamos uma simulação, nós conhecemos parâmetros como μ a partir da especificação do nosso modelo e pela geração de valores repetidos de uma estatística como $\bar{x}$. Na vida real, existe uma perspectiva diferente: nós temos um único valor para a estatística e nós não conhecemos o parâmetro. O que nós podemos dizer sobre um parâmetro baseado em um único valor de $\bar{x}$?

Nós podemos simplesmente inverter a algebra e resolver para μ de modo a dizer que com probabilidade 0.95,

$$\bar{x} - 3.19 * SE < \mu < \bar{x} + 3.05 * SE.$$

Essa fórmula está descrevendo a relação como variáveis aleatórias. Nós temos uma única realização, de modo que nós precisamos ser cautelosos com o termo *probabilidade*, de modo que a frase se torna *nós estamos 95% confiantes de que o intervalo* $\bar{x} - 3.19 * SE, \bar{x} + 3.05 * SE$ *contém o parâmetro desconhecido* μ.

9 Intervalos de Confiança

Uma pesquisa de opinião da Gallup é sumarizada como *46% dos norte-americanos acreditam na visão criacionista de que Deus criou os seres-humanos*. Ao ler a pesquisa mais profundamente, contudo, vemos que nem todos os norte-americanos foram perguntados sobre a questão, mas que a mesma foi baseada em uma amostra de 1.012 norte-americanos. Isso dito, como a Gallup pode utilizar uma amostra desse tamanho para fazer uma generalização sobre toda uma população? Esse é justamente o foco dessa seção do nosso Curso Formação Cientista de Dados. A chave aqui é que *amostras aleatórias* não são *aleatórias* em um aspecto casual, mas obedecem as leis da probabilidade.

Isso introduz uma linguagem que faz declarações tais como os intervalos de confiança mencionados na seção anterior. Nesta, por suposto, nós veremos que se determinadas premissas sobre uma população de onde uma amostra aleatória é obtida são válidas, então nós podemos ter uma certa precisão sobre como uma *estatística amostral* pode ser utilizada para estimar ou fazer inferência sobre um *parâmetro populacional*.

9.1 Intervalos de Confiança para uma proporção populacional p

Vamos supor que na pesquisa da Gallup tivéssemos apenas duas possíveis respostas. Nós poderíamos então nomear uma como `sucesso` e a outra como `fracasso`. A probabilidade de uma pessoa escolhida da população produzir um `sucesso` é um parâmetro da população, tradicionalmente simbolizado por p. Como já vimos, um modelo para o número de `sucessos` poderia ser o modelo binomial, `Binomial(1012,p)`.[15] Deixemos, assim, que $\hat{p}$ seja a proporção de sucessos na amostra aleatória. Ela será 0,46 ou 466 de 1.012.

A questão a se perguntar aqui, como na seção anterior, é sobre a acurácia dessa estimativa de p. Deixemos que $\hat{p}$ denote uma variável aleatória com a distribuição `Binomial(1012,p)`. Assim, $\hat{p}$ tem uma distribuição dependente de p, de modo que o erro da estimativa será

[15] Esse modelo seria melhor apropriado se a amostragem fosse feita com reposição, assim o resultado seria verdadeiramente independente. Ainda, porém, que não seja o caso, é uma boa aproximação.

dado por $\hat{p} - p$. As regras da probabilidade, por suposto, nos dizem que

$$SD(\hat{p}) = \sqrt{\frac{p(1-p)}{n}}, \tag{5}$$

e o teorema central do limite para variáveis aleatórias binomiais nos diz que para um n suficientemente grante, a distribuição de $\frac{(\hat{p}-p)}{SD(\hat{p})}$ é uma normal padrão. Com essa informação, nós poderíamos usar `qnorm(0,025)` para nos dizer que com probabilidade 0,95, nós temos:

$$-1,96 * SD(\hat{p}) < \hat{p} - p < 1,96 * SD(\hat{p}). \tag{6}$$

O resultado acima se aplica à variável aleatória $\hat{p}$, mas nós temos uma realização da variável aleatória, $\hat{p} = 0,46$. Nós estamos apenas 95% confiantes em nosso processo de que o resultado acima é o caso, dado que nós observamos nossa variável aleatória, $\hat{p}$, nossa linguagem probabilística não é muito aproriada.

Com esse pequeno mas importante resultado feito, a próxima tarefa consiste em resolver o par de desigualdades envolvendo p. Relembre que nós assumimos que n é grande o suficiente para que o TLC funcione. Para n grande o suficiente, uma versão diferente se aplica. O erro padrão para $\hat{p}$ é o desvio padrão com parâmetros desconhecidos sendo substituídos por estimativas amostrais. Isto é,

$$SE(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}. \tag{7}$$

Assim a variável aleatória a seguir também tem uma distribuição normal padrão para n grande o suficiente:

$$\frac{\hat{p} - p}{SE(\hat{p})} = \frac{observado - esperado}{SE}. \tag{8}$$

Disso, nós podemos dizer com 95% de confiança que:

$$-1,96 * SE(\hat{p}) < \hat{p} - p < 1,96 * SE(\hat{p}). \tag{9}$$

Isso pode ser então invertido para p de modo a produzir um intervalo contendo p baseado no valor amostral de $\hat{p}$

$$\hat{p} - 1,96 * SE(\hat{p}) < p < \hat{p} + 1,96 * SE(\hat{p}), \tag{10}$$

ou, simplesmente, $\hat{p} \pm 1,96 * SE$. No nosso exemplo da Gallup, $\hat{p} = 0,46$ e $n = 1.012$. Isso gera:

```
phat = 0.46
n = 1012
SE = sqrt(phat*(1-phat)/n)
c(lower=phat - 1.96*SE, upper=phat+1.96*SE)

##     lower     upper
## 0.4292927 0.4907073
```

Margem de Erro

O valor $1,96 * SE$ é costumeiramente chamado de *margem de erro*. O fator multiplicativo 1,96 é, por sua vez, relacionado ao nível de confiança. No exemplo acima, nós usamos 95%. Deixemos que $\alpha = 1 - 0.95$ meça o nível de confiança. Abaixo, como podemos achar nosso fator multiplicativo a partir de α:

```
conf.level = c(0.80, 0.90, 0.95, 0.99)
alpha = 1 - conf.level
multipliers = qnorm(1 - alpha/2)
round(setNames(multipliers, conf.level),2)
##  0.8  0.9 0.95 0.99
## 1.28 1.64 1.96 2.58
```

Intervalo de Confiança para uma proporção

Deixemos que $\hat{p}$ seja uma proporção amostral de uma amostra aleatória e suponha que n é grande o suficiente para que a variável aleatória $\frac{(\hat{p}-p)}{SE}$ seja aproximadamente normal. Assim, um $(1-\alpha) * 100\%$ intervalo de confiança para p baseado em $\hat{p}$ será dado por

$$\hat{p} \pm z * \sqrt{\frac{p(1-p)}{n}}. \tag{11}$$

A função `prop.test`, a propósito, irá computar um intervalo de confiança baseado em valores de x e n.

9.2 Intervalos de Confiança para a média populacional

Deixemos que $x_1, x_2, ..., x_3$ seja uma amostra aleatória de uma população normal com média μ e desvio padrão σ. Assim, essa estatística

$$Z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} = \frac{\bar{x} - \mu}{SD(\bar{x})} \tag{12}$$

nós teremos uma distribuição normal padrão. Isso implica, por exemplo, que mais ou menos 95% do tempo, Z não é maior do que 2 em termos absolutos. Quando σ é conhecido, nós podemos usar isso para criar um intervalo de confiança para μ baseado em $\bar{x}$.

Infelizmente, isso é raro, ou seja, quando conhecemos o desvio padrão, mas não conhecemos a média. Como na subseção anterior, nós substituímos o desvio padrão pelo erro padrão, simplesmente estimando σ com desvio padrão amostral s. $(SE(\bar{x} = s/\sqrt{n}))$ Isso vai gerar:

$$T = \frac{\bar{x} - \mu}{s/\sqrt{n}} = \frac{\bar{x} - \mu}{SE(\bar{x})} = \frac{observado - esperado}{SE}. \tag{13}$$

Na subseção anterior, quando encontramos uma estatística similar, foi assumido que n era grande o suficiente de modo que o TLC implica que a distribuição amostral era aproximadamente a normal padrão. Para essa estatística, será também o caso para valores grandes o suficiente de n. Esse será também o caso para populações que são *não-normal*.

Entretanto, para pequenos valores de n a distribuição de T não é normal. Ela terá a distribuição t com $n - 1$ graus de liberdade. A distribuição t é simétrica, desenhada em forma de sino, que se aproxima assintoticamente da distribuição normal padrão, mas para valores pequenos de n possui caudas mais gordas. O grau de liberdade, por suposto, é um parâmetro para essa distribuição no sentido que a média e o desvio padrão são para a distribuição normal. Assim como na distribuição normal, a distribuição t permite que resolvamos para t^* ou para α a seguinte equação, assumindo que um ou outro é conhecido:

$$P(-t^* < \frac{\bar{x} - \mu}{SE(\bar{x})} < t^*) = 1 - \alpha. \tag{14}$$

Isso pode ser reescrito algebricamente como

$$P(\bar{x} - t^*SE < \mu < \bar{x} + t^*SE) = 1 - \alpha, \tag{15}$$

de modo a facilmente encontrarmos um intervalo de confiança.

Intervalos de Confiança para a média

Deixemos que $x_1, x_2, ..., x_n$ seja uma amostral aleatória de uma população normal com média μ e variância σ^2. Um $(1-\alpha)100\%$ intervalo de confiança para μ será dado por

$$\bar{x} \pm t^* SE(\bar{x}),$$

onde o multiplicador t^* está relacionado com α através da distribuição t com $n-1$ graus de liberdade.

Para dados não sumarizados, a função `t.test` irá computar intervalos de confiança.

Encontrando t^* com o R

Computar valores de t^* para um dado α e vice-versa é feito de maneiro similar a z^*, com a exceção de que uma densidade diferente é usada. Mudar para uma nova densidade requer apenas que utilizamos o nome da família apropriada - t para distribuição t e norm para normal - e especificar os valores dos parâmetros. Em particular, se n é o tamanho da amostra, então os dois estão relacionados como abaixo:

```
tstar = qt(1 - alpha/2, df=n-1)
alpha = 2*pt(-tstar, df=n-1)
```

Para z^*:

```
zstar = qnorm(1 - alpha/2)
alpha = 2*pnorm(-zstar)
```

O intervalo de confiança para a média depende do fato de que a distribuição amostral de $T = (\bar{x} - \mu)/SE$ é a distribuição t com $n-1$ graus de liberdade. Isso é verdade quando x_i for uma amostra aleatória de uma população normal. Mas e se essa suposição em nosso modelo para os dados não for apropriada?

Se n é pequeno, nós podemos fazer simulações de modo a ver se a distribuição de T ainda é aproximadamente a distribuição t se a distribuição parental de x_i não for muito distinta da normal. Isto é, as caudas não podem ser muito longas, ou o viés não pode ser muito alto. Quando n é grande, o TLC se aplica.

Uma estatística onde a distribuição amostral não muda dramaticamente para moderadas mudanças na distribuição populacional é chamada de estatística robusta. No caso, a estatística T é robusta.

Intervalos de Confiança unilaterais

Quando encontramos um intervalo de confiança para a média dado α, nós encontramos t^*, de modo que $P(-t^* \leq T_{n-1} \leq t^*) = 1 - \alpha$. Esse método irá retornar um intervalo de confiança simétrico. A ideia básica é que a área abaixo da densidade da distribuição amostral que fica de fora do intervalo de confiança é uniformemente dividida em cada lado. Isso leva a uma área $\alpha/2$ em cada cauda.

Esse método, contudo, não é o único. Essa área extra pode ser alocada em qualquer proporção à esquerda ou à direita do intervalo de confiança. Intervalos de confiança unilaterais colocam toda a área de um lado ou de outro. Para intervalor de confiança para a média, baseado na estatística T, isso poderia ser encontrado para um dado α ao encontrar t^* de modo que $P(t^* \leq T) = 1 - \alpha$ ou $P(T \leq t^*) = 1 - \alpha$.

No R, as funções `prop.test`, `binom.test` e `t.test` podem retornar intervalos de confiança unilaterais. Quando o argumento `alt='less'` é usado, um intervalo do tipo $(-\infty, b]$ é gerado.

9.3 Outros intervalos de confiança

Para formar intervalos de confiança, nós temos usado o fato de que certas estatísticas,

$$\frac{\hat{p} - p}{SE} \quad \text{e} \quad \frac{\bar{x} - \mu}{SE},$$

tem distribuições amostrais conhecidas que não envolvem nenhum parâmetro populacional. A partir disso, nós podemos resolver para intervalos de confiança para os parâmetros em termos de quantidades conhecidas. Em geral, esse tipo de estatística é chama de *quantidade pivotal* e pode ser usada para gener um número de intervalos de confiança em várias situações.

9.3.1 Intervalo de Confiança para σ^2

Por exemplo, se x_i for i.i.d normal, então a distribuição de

$$\frac{(n-1)s^2}{\sigma^2}$$

será a distribuição qui-quadrado χ com $n-1$ graus de liberdade. Esse fato nos permite resolver para intervalos de confiança para σ^2 em termos da variância amostral s^2.

Em particular, um $(1-\alpha)*100\%$ intervalo de confiança pode ser encontrado como se segue. Para um dado α, deixemos que l^* e r^* resolvam

$$P(l^* \leq \chi^2_{n-1} \leq r^*) = 1-\alpha \tag{16}$$

Se nós escolhemos l^* e r^* de modo a gerar áreas iguais nas caudas, nós podemos encontrá-los com

```
n = 10; alpha = 1 - 0.9
lstar = qchisq(alpha/2, df=n-1)
rstar = qchisq(1-alpha/2, df=n-1)
```

Então,

$$P(l^* \leq \frac{(n-1)s^2}{\sigma^2} \leq r^*) = 1-\alpha \tag{17}$$

pode ser reescrito como

$$P(\frac{(n-1)s^2}{r^*} \leq \sigma^2 \leq \frac{(n-1)s^2}{t^*}) = 1-\alpha. \tag{18}$$

Em outras palavras, o intervalo $((n-1)s^2/r^*, (n-1)s^2/l^*)$ dará um $(1-\alpha)*100\%$ intervalo de confiança para σ^2.

10 Testes de Significância

Imagine o seguinte cenário simplificado: um réu é acusado de um crime e deve ser julgado. Durante o julgamento, um promotor e um advogado de defesa, respectivamente, tentam convencer o júri de que o réu é culpado ou inocente. O júri deveria ser imparcial. Quando decidindo o destino do réu, os jurados são instruídos a assumir que o réu está inocente, a menos que se prove culpado, sem sombra de dúvida. No final do julgamento, os jurados decidem a culpa ou inocência do réu com base na força de sua crença no pressuposto de sua inocência, dada a evidência. Se os jurados acreditam ser muito improvável que uma pessoa inocente possa ter provas em contrário, eles encontrarão réu "culpado". Se não for tão improvável, eles vão decidir "não culpado". O sistema não é infalível. Um homem culpado pode ser libertado se for considerado inocente, e um homem inocente pode ser erroneamente condenado. A frequência com que esses erros ocorrem depende do limiar usado para encontrar a culpa. Em um julgamento criminal, para diminuir a chance de um veredicto de culpado errôneo, o critério é mais rigoroso. Em um julgamento civil, é relaxado para uma preponderância da evidência. Este último torna mais fácil errar quando uma pessoa verdadeiramenté inocente, mas mais difícil errar com um verdadeiramente culpado.

Essa questão pode ser refraseada como um teste de hipótese. A suposição de inocência é substituída pela hipótese nula, H_0. Isto está em contraste à hipótese alternativa, H_A. Isso seria uma suposição de culpa na analogia do julgamento. Em um julgamento, essa alternativa não é usada como uma suposição; só norteia a interpretação das provas. A determinação da culpa por um júri não é prova da alternativa, apenas uma percepção de falha da suposição de inocência para explicar bem o suficiente a evidência disponível.

Um veredicto de culpado é mais precisamente chamado veredicto de "não inocente". Quem realiza um teste de significância procura determinar se a hipótese nula é razoável dados os dados disponíveis. A evidência é substituída por uma experiência que produza uma estatística de teste. A probabilidade de que a estatística de teste seja o valor observado ou seja mais extremo, como implícito pela hipótese alternativa, é calculado usando as premissas da hipótese nula. Isso é chamado de **p-valor**. Isto é como a pesagem da evidência - o júri calcula a probabilidade de que a evidência esteja de acordo com o suposição de inocência. O cálculo do p-valor é chamado de **teste de significância**. O p-valor é baseado

tanto na distribuição amostral da estatística de teste em H_0 quanto o único valor observado durante o julgamento.

10.1 Testes de Significância para proporções na população

Um pesquisador pode querer saber se a aprovação de um político está caindo ou se a taxa de desemprego está aumentando, por exemplo. Em muitos casos, existe uma proporção conhecida. O que se pede é uma comparação com esta proporção. Um teste de proporção pode ser usado para ajudar a responder a essas perguntas. Suponha que p_0 reflita a proporção historicamente verdadeira de alguma variável de interesse. O pesquisador pode querer testar se a proporção desconhecida atual, p, é diferente de p_0.

Um teste de proporção verificaria a hipótese nula, $H_0 : p = p_0$ contra possibilidades alternativas, de que p é meramente diferente de p_0, maior ou menor.

10.2 Testes de Significância para Médias, o teste t

No teste t, temos uma hipótese nula de que a média populacional μ é igual a μ_0. A estatística t é computada da seguinte maneira:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} \sim t_{n-1} \tag{19}$$

10.3 Testes de Significância para Medianas

Um dos principais problemas dos testes t é que eles assumem normalidade ou uma amostra grande, pontos nem sempre alcançáveis. Nesses casos, talvez seja mais interessante usar testes não-paramétricos, que não carregam suposições sobre como é a população da qual tiramos a amostra.

10.3.1 O teste do Sinal

O teste do Sinal somente supõe que a distribuição populacional é contínua e positiva - o que é bem razoável. A hipótese nula é de que a mediana é um número m. Se contarmos as observações maiores que a mediana, teremos um número com distribuição Binominal com parâmetros n e $1/2$.

10.3.2 O teste do Sinal Ordenado

Para testes de sinal ordenados, usamos principalmente o Teste se Wilcoxson para Mediana. Supomos uma amostra i.i.d., hipótese nula de que a mediana é m e alternativa de que não é m. A estatística do teste é:

$$T = \sum_{i:x_i>m} \text{rank}(|x_i - m|) \tag{20}$$

T segue uma distribuição probabilística conhecida, porém complexa. A função que implementa esse teste no R é "wilcox.test".

10.4 Testes para Duas Amostras

10.4.1 Para Proporções

Nas subseções anteriores, os testes comparavam uma amostra com valores presumidos de certos parâmetros na população. Isso supõe que temos conhecimento sobre esses parâmetros populacionais, o que pode ser problemático, mas também podemos querer comparar uma amostra com outra e não com a população. Voltando ao teste de proporções, queremos podemos testar se em duas amostras elas são diferentes, teríamos uma hipótese nula $H_0 : p_1 = p_2$, que é equivalente a $H_0 : p_1 - p_2 = 0$. Temos uma estatística de teste apropriada:

$$Z = \frac{(\hat{p_1} - \hat{p_2}) - \mathbb{E}[\hat{p_1} - \hat{p_2}|H_0]}{SE[\hat{p_1} - \hat{p_2}|H_0]} \tag{21}$$

Estamos supondo que as amostras são aleatórias e representativas da população, então sob a hipótese nula, a esperança de Z é meramente o diferencial de p_1 e p_2, 0. Sabendo disso, a estatística do teste agora tem outra cara:

$$Z = \frac{(\hat{p_1} - \hat{p_2})}{\sqrt{[\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}} \tag{22}$$

Este teste está implementado na função prop.test, com o parâmetro alternative em "two.sided".

10.4.2 Para Centros

Um médico pode estar interessado em saber se o tempo de recuperação de uma cirurgia é maior do que o de outra, ou um taxista pode querer saber se existe diferença no tempo de viagem entre duas rotas. Então, suponha x_i e y_j duas amostras de duas populações de interesse. Um teste de significância para comparar os centros de suas duas distribuições subjacentes teria hipótese nula $H_0 : \mu_x = \mu_y$.

Um teste razoável para essa hipótese dependeria nas hipóteses feitas sobre as populações subjacentes. Se elas são normalmente distribuídas (ou algo próximo disso), então a estatística t é razoável. Se não, talvez o teste de Wilcoxson seja mais interessante. Com populações normais, teremos algo como Esperado - Observado / Erro Padrão:

$$t = \frac{(\bar{x} - \bar{y}) - \mathbb{E}[\bar{x} - \bar{y}|H_0]}{SE[\bar{x} - \bar{y}|H_0]} \tag{23}$$

Sob a hipótele nula o valor esperado da diferença é 0. Esse teste também está implementado - como no teste t anterior - na função t.test, dado que o parâmetro alternativo é "two.sided".

11 Qualidade do Ajuste

Agora nos voltamos a avaliar o quão bem uma distribuição probabilística teórica se encaixa em dados reais.

11.1 O teste chi-quadrado

Em uma pesquisa de opinião pública, normalmente há mais do que duas opiniões possíveis. Por exemplo, se nos EUA perguntam a um entrevistado qual partido deve ser eleito. O entrevistdo pode falar Republicano, Democrata ou ficar indeciso. Se então tivermos um resultado como, de 100 entrevistados, 35 responderam Republicano, 40 Democratas e 25 foram indecisos, a diferença entre Republicanos e Democratas é estatísticamente significante?

11.1.1 A distribuição Multinomial

Precisamos de um modelo probabilístico apropriado. A distribuição Multinomial nos serve e podemos encará-la como uma generalização da distribuição Binomial.

Imagine que temos k categorias, cada uma com probabilidade $p_1, p_2, ..., p_k$ de ser escolhida. É óbvio que $\sum_{i=1}^{k} p_i = 1$. Se definirmos n como o número de "tiradas", y_i como o número de realizações da k-ésima possibilidade, então a probabilidade conjunta de uma certa combinação de realizações é:

$$P(y_1 = y_1, ..., y_k = y_k) = \binom{n}{y_1}\binom{n-y_1}{y_2}...\binom{n-y_1-y_2-...-y_{k-1}}{y_k} p_1^{y_1}...p_k^{y_k} \qquad (24)$$

VOltando ao exemplo, imagine que esperávamos 35 para Republicanos, 35 para Democratas e 30 indecisos. Se fosse o caso então que de 100 entrevistados, 35 responderam Republicano, 40 Democratas e 25 foram indecisos, a probabilidade desse evento é:

$$P(y_1 = 35, y_2 = 40, y_3 = 25) = \binom{100}{35}\binom{65}{40}\binom{25}{25}(0.35)^{35}(0.35)^{40}(0.3)^{25} \qquad (25)$$

Você pode operacionalizar essa conta no R com a função *choose* para as escolhas. Se o fizer, vai ver que é de 0,38%. A função *dmultinom* implementa essa distribuição diretamente.

11.1.2 A estatística χ^2 de Pearson

Usar a distribuição Multinomial diretamente para resolver um problema de cálculo de p-valor é difícil, já que provavelmente várias das variáveis têm correlação - dado que existe um tamanho de amostra n, então se uma tem mais recorrência, as outras precisam ter menos.

No entanto, podemos fazer como antes: calcular o observado, o que era esperado e normalizar isso para que tenha alguma distribuição probabilística conhecida. Se focarmos somente em uma variável y_i, podemos vê-la como uma Binomial(n, p_i) com uma esperança de np_i. Uma estatística seria:

$$\sum_{i=1}^{k}(y_i - np_i)^2$$

Normalizando essa estatística ao dividir pelo valor esperado, temos a estatística chi-quadrado de Pearson:

$$\chi^2 = \sum_{i=1}^{k}\frac{(y_i - np_i)^2}{np_i} \tag{26}$$

Se o modelo multinomial está correto, então a distribuição assintótica de y_i é chi-quadrado com $n - 1$ graus de liberdade.

11.2 O teste chi-quadrado de independência

É comum estarmos interessados em saber se duas variáveis aleatórias são independentes. Por exemplo: as chances de pais e filhos em um mesmo carro usarem cintos são independentes? Seja i um índice de resultados possíveis em uma variável aleatória que

varia de 1 a n_r e j um índice análogo para a outra variável aleatória, que vai de 1 a n_c. n é o tamanho da amostra. Então podemos computar a estatística chi-quadrado como:

$$\chi^2 = \sum_{i=1}^{n_r} \sum_{j=1}^{n_c} \frac{(y_i - np_{ij})^2}{np_{ij}} \tag{27}$$

Sob a hipótese de dados multinomiais e variáveis independentes, essa estatística teria distribuição chi-quadrado com $(n_r - 1)(n_c - 1)$ graus de liberdade.

11.2.1 O teste chi-quadrado de Homogeneidade

Como endereçar a efetividade de um tratamento de uma droga? Tipicamente, existem ensaios clínicos com alocação aleatória de participantes em cada grupo. Quando os resultados são categóricos, podemos usar a estatística chi-quadrado para testar se as distribuições dos resultados são iguais.

O teste descrito está implementado na função "chisq.test".

11.3 Teste Kolmogorov-Smirnof

Suponha que temos uma amostra aleatória $X_1, X_2, ..., X_3$ para alguma distribuição contínua. Então a função de distribuição acumulada de X é $F(x) = P(X \leq x)$. Para uma amostra, $X_1, X_2, ...X_n$, a distribuição empírica é a distribuição gerada por amostragem a partir dos pontos de dados. A probabilidade de que um número escolhido aleatoriamente de um amostra ser menor ou igual a x é o número de observações na amostra menor ou igual a x dividido por n. Usamos a notação $F_n(x)$ para isso:

$$F_n(x) = \frac{\{i : X_i < x\}}{n} \tag{28}$$

Se definirmos D como o máximo em relação a x do módulo da diferença entre $F(x)$ teórica e empírica, então meramente precisamos supor que F seja contínua para D tenha uma distribuição conhecida. Usamos essa distribuição então para testar se uma amostra tem

uma certa distribuição. Esse é o teste de Kolmogorov-Smirnof, implementado pela função "ks.test".

11.4 Teste Shapiro-Wilk de Normalidade

O teste de Kolmogorov-Smirnof serve para diagnosticar se dados se encaixam em uma distribuição qualquer que devemos especificar. O teste de Shapiro-Wilk (SW) serve especificamente para diagnosticar se uma amostra tem distribuição Normal. Isso é útil porque o teste KS depende de uma distribuição teórica com parâmetros especificados *sem olhar os dados*. Por consequência, só podemos testar normalidade com o teste KS se conhecermos de antemão os parâmetros populacionais - que raramente conhecemos. O teste de SW não depende disso, já que se baseia na ideia de quantil-quantil. Em essência, observamos os quantis amostrais e comparamos com os quantis que essa distribuição teoricamente teria se fosse normal.

Esse teste é implementado na função "shapiro.test".

12 Regressão Linear

Uma maneira simples e poderosa de resumir o relacionamento (de preferência linear) de duas variáveis é através de um modelo linear, que podemos expressar como:

$$Y_i = \beta_0 + \beta X_i + \epsilon_i \tag{29}$$

Onde Y_i é o que chamamos de variável de reposta, variável explicada ou variável dependente. β_0 é uma constante, β é o parâmetro que dá a magnitude em que uma variação de X, que chamamos de variável explicativa, covariada ou variável independente, leva a variável de resposta a variar. Por fim, ϵ_i é um termo de erro, i é um índice para indexar observações da amostra. Normalmente assumimos que $\epsilon \sim N(0, \sigma^2)$. Essa hipótese de normalidade dos erros faz com que a distribuição da variável de resposta seja normal, com média $\mu_{Y|X} = \beta_0 + \beta_i$ e variância σ^2.

Em um modelo de regressão linear simples, podemos estimar o parâmetro β rapidamente através do estimador de Mínimos Quadrados Ordinários:

$$\hat{\beta} = \text{arg min}_\beta \sum_{i=1}^{n} (y_i - \beta x_i)^2 \tag{30}$$

Solucionando o problema de otimização, temos outros resultado - que decorre diretamente do estimador apresentado:

$$\hat{\beta} = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} \tag{31}$$

Esse estimador está implementado no R, através da função "lm".

12.1 Inferência no Modelo Linear

O modelo linear de que tratamos faz certas hipóteses sobre o comportamento dos dados que devemos verificar. Em particular, o modelo deve ser apropriado para a média da

variável de resposta e os resíduos devem ter distribuição normal e independente. Podemos fazer uma inspeção visual dos resíduos.

12.1.1 Testes t marginais

Podemos construir intervalos de confiança e executar testes de hipótese em estimadores. Se por exemplo, tivermos a hipótese nula $H_0 : \beta = b$ então podemos computar a estatística t:

$$T = \frac{\hat{\beta} - b}{SE(\hat{\beta})} \tag{32}$$

Sob a hipótese nula, essa estatística tem distribuição t de Student com $n - 2$ graus de liberdade. No caso particular em que $b = 0$, dizemos que há Teste t Marginal.

12.1.2 O teste F

Podemos usar um teste alternativo para testar a hipótese anterior com uma abordagem similar, mas que generaliza o modelo de regressão linear. Um dos objetivos de modelar é explicar variância na variável de resposta com um ou mais regressores. Podemos definir a *variação total* da variável resposta como a Soma dos Quadrados Totais (SQT):

$$SQT = \sum (y_i - \bar{y})^2$$

A SQT pode ser decomposta em uma soma: dos quadrados dos resíduos (SQR), que são a diferença entre o estimado e o observado - e dos quadrados explicados (SQE), diferença do estimado para a média do observado. A fórmula abaixo resume:

$$SQT = \sum (y_i - \hat{y}_i)^2 + \sum (\hat{y}_i - \bar{y})^2 \tag{33}$$

A cada um desses termos associamos um grau de liberdade, a depender do tamanho da amostra. Para a SQT, temos uma amostra de tamanho n e um valor estimado, sua média

amostral, deixando $n-1$ graus de liberdade. Para o SQR, temos de novo n observações, mas dois valores estimados, os coeficientes β_0 e β, deixando $n-2$ graus de liberdade. O que deixa somente 1 grau de liberdade para SQE. Se dividirmos uma soma de quadrados por seus graus de liberdade, temos o que se chama de *soma média de quadrados*.

Isso pode ser informativo se conseguirmos padronizar alguma estatística que envolva soma de quadrados para algo cuja distribuição probabilística conhecemos, o que nos permitiria fazer testes de hipótese. A estatística F é definida como a razão da soma média dos quadrados explicados pela soma média dos quadrados dos resíduos:

$$F = \frac{\frac{SQE}{1}}{\frac{SQR}{n-2}} \, F_{1,n-2} \tag{34}$$

Podemos reescrever a equação como:

$$F = (\frac{\hat{\beta}}{SE(\hat{\beta})})^2 \tag{35}$$

12.1.3 O coeficiente de determinação R^2

A decomposição em somas de quadrados de que falamos anteriormente nos permite interpretar o quão bem um modelo se ajusta aos dados. Se os dados estão muito dispersos ao redor da linha de regressão estimada, então teremos um SQR maior e podemos dizer que explicamos uma fração pequena da variância da variável de resposta. Existem duas formas equivalente de computar o coeficiente de determinação:

$$R^2 = 1 - \frac{SQR}{SQT} = \frac{SQE}{SQT} \tag{36}$$

O R^2 sempre é um número de zero a 1 e podemos interpreta-lo como uma porcentagem: qual proporção da variância da variável de resposta o modelo consegue explicar. Podemos também definir o R^2 *ajustado* ao levar em conta os graus de liberdade de SQR e de SQT, para penalizar modelos que conseguem coeficientes de determinação melhor ao adicionar regressores.

12.2 Regressão Linear Múltipla

A regressão múltipla nos abre portas para tentar modelar variáveis que dependem de várias outras. Essas outras variáveis podem ser diferentes, potências de uma variável que já temos, funções delas... As possibilidades são grandes. Boa parte do aparato da regressão linear simples se mantém no caso com várias variáveis.

De uma maneira bem geral, os modelos agora terão essa forma:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \epsilon_i \tag{37}$$

Temos agora $p + 1$ parâmetros em uma forma funcional linear. Agora entendemos que os valores de y_i são amostras independentes de uma distribuição normal com média $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$ e variância σ^2. Se os valores das variáveis x_i são aleatórios, isso também é verdade para as médias condicionais.

Agora não temos mais fórmulas fechadas e simples para estimar cada parâmetro, mas ainda temos uma para a variância da regressão:

$$\hat{\sigma}^2 = \frac{SQR}{n - (p + 1)} \tag{38}$$

13 Análise de Variância

Análise de Variância, ou *ANOVA*, é um método de comparar médias através de amostras baseado nas variações das médias.

13.1 ANOVA de um sentido

Uma análise de variância de um sentido é uma generalização do teste t para duas amostras independentes, nos permitindo comparar médias populacionais de várias amostras independentes. Suponha que temos k populações de interesse e de cada uma destas tirados uma amostra aleatória. Vamos notar que para a i-ésima amostra, x_{in} será o n-ésimo elemento dessa amostra.

Suponha que a média da i-ésima popualaçõ é μ_i e seu desvio-padrão é σ_i - que será simplesmente σ se o desvio-padrão for consistente entre os grupos. Um modelo estatístico para os dados com um desvio-padrão comum seria:

$$x_{ij} = \mu_i + \epsilon_{ij} \tag{39}$$

onde os termos de erro ϵ_{ij} são independentes, normalmente distribuídos com média zero e variância σ^2.

Se quisermos testar se várias amostras têm uma mesma média, podemos considerar o modelo linear apresentado. Ao estima-lo, teremos SQT e SQR, dos quais podemos construir uma estatística que já vimos, a F:

$$F = \frac{SQT/(k-1)}{SQR/(n-k)} \sim F_{(k-1),(n-K)} \tag{40}$$

Como conhecemos a distribuição F com $(k-1)$ e $(n-K)$ graus de liberdade, podemos realizar um teste de hipótese para as médias de cada amostra, em que a hipótese nula é de que são todas iguais e a alternativa alguma negativa disso. A função oneway.test implementa esse teste no R.

É conveniente usar fatores para fazer esses testes. Se armazenamos a variável que indica em qual das i amostras está a observação como um fator "f", então podemos especificar o teste como $x \sim f$ que o R interpretará isso corretamente. Uma outra função para implementar análise de variância é "aov".

Um exemplo: Pesquisadores da Montana State University realizaram um estudo sobre como os vários tipos de esqui afetam o desempenho no esqui cross-country. Existem três básicos: clássico, moderno e integrado. Suponha que 9 esquiadores sejam designados em aleatório para os três tipos de aderência e para cada um, o esquiador tem sua força no tronco superior medida. Podemos investigar a hipótese nula de que os três tipos produzirão médias iguais com uma análise de variância. Nós assumimos que os erros são todos independentes e que os dados são amostrados a partir de populações normalmente distribuídas com variância comum, mas talvez médias diferentes.

13.1.1 O teste não-paramétrico de Kruskal-Wallis

O teste da soma de postos de Wilcoxon foi discutido como uma alternativa não-paramétrica para o teste t de duas amostras para amostras independentes. Embora não fizéssemos suposições sobre os parâmetros da população, assumimos que elas tinham densidades de mesma forma funcional e talvez centros diferentes. O teste de Kruskal-Wallis, um teste não-paramétrico, é análogo ao de Wilcoxson para comparar as médias populacionais de k amostras independentes. Em particular, se $f(x)$ é uma densidade de uma variável aleatória contínua com com média 0, supomos que x_{ij} são tiradas independentemente dos outros, de uma população com densidade $f(x - \mu_i)$. As hipóteses testadas são $H_0 : \mu_1 = \mu_2 = ... = \mu_i$ e $H_1 : \mu_i \neq \mu_j$ para algum par de amostras i e j.

A estatística de teste envolve o posto de todos os dados. Seja r_{ij} o respectivo posto de uma observação quando todos os dados são classificados do menor para o maior, $\bar{r}_i$ a média do posto de cada grupo, e $\bar{r}$ a média total. A estatística de teste é:

$$T = \frac{12}{n(n+1)} \sum_i n_i(\bar{r}_i - \bar{r})^2 \sim \chi^2_{k-1} \tag{41}$$

Como essa estatística tem distribuição conhecida, uma chi-quadrado com $k - 1$ graus de

liberdade, podemos usa-la para testes de hipótese.

13.2 Comparando múltiplas diferenças

Quando a análise de variância é executada com a função "lm", a saída do resumo exibe inúmeros testes estatísticos. O teste F realizado é para a hipótese nula de que $\beta_2 = \beta_3 = = \beta_k = 0$ contra uma alternativa que um ou mais parâmetros diferem de 0. Ou seja, que uma ou mais das variáveis tem efeitos de tratamento em comparação com o nível de referência. Os testes t marginais que são executados são testes de dois lados com uma hipótese nula de que $\beta_i = \beta_1$, cada um é feito para $i = 2, 3, ..., k$. Estes testam se algum dos tratamentos adicionais tem um efeito de tratamento quando controlado pelas outras variáveis.

No entanto, podemos querer fazer outras perguntas sobre os vários parâmetros. Por exemplo, comparações que não são informadas por padrão são testes mais específicos como β_2 e β_3 diferem? e β_1 e β_2 são metade de β_3?. Vamos avaliar agora diferentes múltiplas de parâmetros.

Se sabemos de antemão que estamos procurando uma diferença entre dois parâmetros, então um teste t simples é apropriado (como no caso em que estamos considerando apenas duas amostras independentes). No entanto, se olharmos para os dados e depois decidirmos para testar se o segundo e terceiro parâmetros diferem, então o nosso teste t é instável. Por quê? Lembre-se de que qualquer teste está correto apenas com alguma probabilidade, mesmo que os modelos estejam corretos. Isso significa que às vezes eles falham e quanto mais testes realizamos, mais provavelmente um ou mais falhará.

Podemos, por exemplo, nos perguntar se linhas aéreas diferentes estão sujeitas a tempos diferentes de espera em um mesmo aeroporto. Estaríamos comparando dois parâmetros entre si e com o a *opção nula* de que ambos sejam na verdade 0.

13.3 ANCOVA

Análise de Covariância (ANCOVA) é o nome dado aos modelos em que tanto variáveis categóricas quanto numéricas são usadas como preditoras. Também rodamos ANCOVAs

com a função "lm". Para comparar a performance de dois modelos dessa maneira, precisamos estimar dois modelos lineares, salva-los como objetos no R e depois alimenta-los à função "anova".

14 Duas Extensões do Modelo Linear

As idéias de regressão linear são blocos de construção para muitos outros modelos estatísticos. O repositório da Fundação R contém centenas de pacotes com adições, muitos dos quais implementam extensões para o modelo de regressão linear abrangidos nos dois últimos capítulos. Neste capítulo analisamos duas extensões: modelos de regressão logística e modelos não-lineares. Nosso objetivo é ilustrar que a maioria das técnicas usadas para modelos lineares são transferidas para esses (e outros) modelos.

O modelo de regressão logística abrange a situação em que a variável de resposta é uma variável binária. Regressão logística, que é um caso particular de um Modelo Linear Generalizado, surge em diversas áreas, incluindo por exemplo, a análise de dados de pesquisas. Em modelos não-lineares discutimos usar uma função para descrever a resposta média que não é linear nos parâmetros.

14.1 Modelos de Regressão Logística

Uma variável binária é aquela que pode ter apenas dois valores, *sucesso* ou *falha*, muitas vezes codificado como 1 ou 0. No modelo ANOVA vimos que podemos usar variáveis binárias como preditores em um modelo de regressão linear usando fatores. E se quisermos usar um variável binária como uma variável de resposta?

Esse tipo de problema tipicamente surge quando estamos tentando classificar observações com um modelo. Dadas certas observações clínicas, um paciente tem que probabilidade de ter uma certa doença? Conseguimos inferir se um e-mail é spam ou não a partir de características objetivas do corpo do texto, se precisar de alguém para isso, somente com um modelo? Um cliente de um banco vai pedir um empréstimo ou não?

Suponha que temos duas variáveis X e Y, onde Y é a variável de resposta, que assume 0 ou 1. Se escrevermos um modelo linear como vimos anteriormente, $Y_i = \beta_0 + \beta X_i + \epsilon_i$, precisamos impor seríssimas restrições no comportamento dos erros nesse modelo, que não podem mais ser normais. É melhor então trocar Y_i pela probabilidade de Y_i ser um sucesso, $\pi_i = P[Y_i = 1]$. Agora podemos escrever $\pi_i = \beta_0 + \beta X_i + \epsilon_i$. No entanto observe que apesar dos erros ficarem mais bem comportados, ainda não como

gostaríamos. Enquanto a probabilidade varia entre 0 e 1, o lado direito da equação varia muito mais amplamente.

Se condicionarmos a X_i, então podemos trocar a probabilidade pelo valor esperado $E[Y_i|X_i] = \pi_i$. A última alteração necessária é como tratamos a regressão binária. Vamos quebrar o procedimento em dois passos. Primeiro, estimaremos $\eta = \beta_0 + \beta X_i + \epsilon_i$, então assumimos que podemos transformar η para dar a média condicional de uma função $m()$, que chamamos de função de ligação. Quando $m = \frac{e^x}{1+e^x}$, dizemos que o modelo é de regressão logística. Resolvendo, teríamos:

$$\pi_i = \frac{e^{\beta_0 + \beta X_i}}{1 + e^{\beta_0 + \beta X_i}} \tag{42}$$

A função logística $m()$ transforma qualquer valor real em valores entre 0 e 1, de forma que seus valores possam ser lidos como probabilidades. Quando $m()$ é invertida, temos:

$$\log \frac{\pi_i}{1 - \pi_i} = \beta_0 + \beta X_i \tag{43}$$

Esse termo em log é chamado *log da razão de chances*. Para clarificar, as chances de uma probabilidade p são $p/1 - p$, e entendemos que se um evento tem chances a em b então se tivermos $a + b$ amostras, esperamos que o evento aconteça a vezes.

14.1.1 Modelo Linear Generalizado

Podemos escrever o Modelo Linear Generalizado como, em princípio:

$$\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k \tag{44}$$

A média de Y dados os valores de X então é relacionada a η por uma função de ligação invertível $m()$ como $\mu = m(\eta)$. A implementação de modelos assim é pela função "glm".

14.2 Modelos Não-Lineares

Chamamos os modelos que vimos até agora de *lineares* porque sempre que estimamos um coeficiente ele é uma constante multiplicando um termo (esse sim, não precisa ser linear). Modelos não-lineares permitem relações mais complexas, como por exemplo o exponencial:

$$Y_i = \beta_0 e^{-\beta_1 x_i} + \epsilon_i \tag{45}$$

Observe aqui que esse modelo não é mais linear nos parâmetros.

Modelos mais gerais poderiam ter mais preditores e outros tipos de erros, como multiplicativos. As possibilidades parecem infinitas, mas na verdade são limitadas pelo problema que estamos modelando. Ao usar modelos não-lineares, normalmente temos uma ideia de quais tipos de modelos são apropriados para os dados e cabem apenas aqueles. Se o modelo tiver erros i.i.d. que são normalmente distribuídos, então usando o método dos mínimos quadrados podemos encontrar estimativas de parâmetros e usar o AIC para comparar modelos.

Podemos estimar modelos dessa natureza com a função "nls".

15 Introdução à Machine Learning

Nessa última seção do nosso Curso Formação Cientista de Dados, vamos fazer uma introdução à *machine learning*. Esperamos que essa introdução os ajude a compreender os conceitos fundamentais que definem e diferenciam as abordagens mais utilizadas de machine learning.[16]

Você aprenderá nessa introdução:

- As origens e aplicações práticas de machine learning;
- Como transformar dados e conhecimento em ação;
- Como adequar algoritmos de machine learning a dados reais.

15.1 Origens

Diversos aspectos da nossa vida são registrados. Governos, administradores e indivíduos estão todo o tempo registrando e reportando informação. Esse dilúvio de dados tem levado muitos a constatar que vivemos uma era de **Big Data**, mas isso pode ser um termo impróprio. Isto porque, temos estado desde sempre cercados de um amontado de dados. O que diferencia a era atual é que temos um vasto conjunto de *dados registrados*. Essa riqueza de informações tem o potencial de gerar ação, dada uma sistematização que transforme conjuntos de informação aparentemente confusos que façam sentido.

O campo de estudo interessado no desenvolvimento de algoritmos que transformam dados em ações inteligentes é conhecido como **machine learning**. Esse campo se originou em um ambiente onde dados disponíveis, métodos estatísticos e poder computacional rápido e simultaneamente se desenvolveram. O crescimento dos dados gera uma necessidade de poder computacional, o que por sua vez transborda para o desenvolvimento de métodos estatísticos para analisar grandes conjuntos de dados. Isso criou um ciclo de progresso, permitindo que conjuntos ainda maiores e mais interessantes de dados pudessem ser coletados.

[16] Essa introdução é baseada principalmente em Lantz (2013).

15.2 Aplicações

Machine learning é mais bem sucedido quando ela aumenta ao invés de substituir um conhecimento especializado de um assunto específico. Ela funciona com médicos em busca da cura do câncer, programadores comprometidos em construir casas e automovéis inteligentes ou ajudando cientistas sociais a construir conhecimento sobre como as sociedades funcionam. Algumas outras aplicações podem ser listadas abaixo:

- Identificação de mensagens indesejáveis;
- Segmentação do comportamento de consumidores para propaganda direcionada;
- Previsão do tempo e de mudanças climáticas de longo-termo;
- Redução de fraudes em transações de cartão de crédito;
- Estimativas atuariais de danos financeiros associados a tempestades e desastres naturais;
- Previsão de eleições;
- Desenvolvimento de algoritmos para drones e carros sem motoristas;
- Otimização do consumo de energia em casas e escritórios;
- Projeção de áreas onde é mais provável ocorrer crimes;
- Descobrimento de sequências genéticas associadas a doenças.

15.3 Como as máquinas aprendem?

Uma definição formal de machine learning foi proposta por Tom M. Mitchell: `Maquinas aprendem sempre que são capazes de utilizar suas experiências de modo que essa performance melhora uma experiência similar no futuro.` Embora essa definição seja intuitiva, ela ignora por completo o processo exato sobre como essa experiência é transformada em ação futura.

Enquanto o cérebro humano é naturalmente capaz de aprender desde o nascimento, as condições necessárias para que computadores aprendam precisam ser especificadas. Por

essa razão, embora não seja estritamente necessário entender as bases teóricas do processo de aprendizado, essa fundação ajuda a entender, distinguir e implementar algoritmos de machine learning.

Em termos gerais, o processo de aprendizado pode ser dividido em quatro componentes:

- **Armazenamento de dados**: utiliza observação, memória e recordações para prover uma base factual para aumentar o raciocínio;
- **Abstração**: envolve a tradução de dados armazenados em representações e conceitos mais amplos;
- **Generalização**: usa dados abstraídos para criar conhecimento e inferências que direcionam ações em novos contextos;
- **Avaliação**: provê um mecanismo de feedback para medir a utilidade do conhecimento aprendido e informar possíveis melhorias.

15.4 Machine Learning na prática

Até aqui, temos focado como machine learning trabalha em teoria. De modo a aplicar o processo de aprendizado a tarefas do mundo real, nós podemos utilizar um processo de cinco etapas:

- **Coleta de dados**: envolve reunir o material de aprendizado que o algoritmo irá usar para gerar conhecimento tangível;
- **Exploração de dados e preparação**: A qualidade de qualquer projeção de machine learning está de longe baseada na qualidade dos dados imputados. Por isso, é importante aprender mais sobre os dados e seus nuances durante o processo de *exploração*. Um trabalho adicional é necessário para preparar os dados para o processo de aprendizagem. Isso envolve corrigir ou limpar dados desestruturados, eliminando dados desnecessários e guardando os dados conforme as expectativas de aprendizado;
- **Treinamento do modelo**: Desde que os dados estão preparados para a análise, você está pronto para ter alguma dimensão sobre o que pode ser aprendido a partir

dos dados. A tarefa específica de machine learning irá informar o algoritmo apropriado e este irá representar os dados na forma de um modelo;

- **Avaliação do modelo**: Dado que cada modelo de machine learning resulta em uma solução viesada para o problema de aprendizado, é importante avaliar quão bem o algoritmo aprende a partir dessa experiência. A depender do tipo de modelo usado, você pode ser capaz de avaliar a acurácia do modelo usando um conjunto de dados de teste (*dataset test*) ou criar medidas de performance específicas para uma dada aplicação;
- **Aperfeiçoamento do modelo**: Se uma melhor performance for necessário, pode ser importante utilizar estratégias mais avançadas de modo a aumentar a acurácia do modelo. Em alguns casos, pode ser necessário mudar o tipo de modelo, adicionar outras variáveis ou mesmo refazer o trabalho de preparação dos dados.

15.5 Tipos de algoritmos

Algoritmos de machine learning são divididos em categorias de acordo com os seus propósitos. Entender as categorias dos algoritmos de aprendizado é um primeiro passo essencial na direção de usar os dados para direcionar uma ação desejada.

Um **modelo de previsão** é utilizado para tarefas que como o nome diz exigem a previsão de um valor utilizando outros valores do conjunto de dados. O algoritmo de aprendizado busca descobrir e modelar a relação entre o *objetivo*, a variável a ser prevista, e outras variáveis. Dado que em modelos de previsão está muito claro sobre o que e como eles precisam aprender, o processo de treinar um modelo de previsão é conhecido como **aprendizado supervisionado**. Dado um conjunto de dados, um algoritmo de aprendizado supervisionado busca otimizar um função - o modelo - de modo a encontrar uma combinação de valores que resultam no *output* esperado.

A tarefa de machine learning supervisionada frequentemente utilizada para prever a qual categoria pertence um exemplo é conhecida como **classificação**. Potenciais usos:

- Um e-mail é um spam;
- Uma pessoa tem câncer;

- Um time de futebol irá perder ou ganhar;
- Um tomador não irá pagar um empréstimo.

Em algoritmos de classificação, o objetivo a ser previsto é um aspecto categórico conhecido como **classe** e é dividida em categorias chamadas de **níveis**. Algoritmos supervisionados podem ser utilizados também para **previsões numéricas**.

Um **modelo descritivo**, por outro lado, é utilizado para tarefas que se beneficiam dos insights gerados pela sumarização dos dados em um novo e interessante modo. Como oposição aos modelos preditivos que prevem um objetivo de interesse, em um modelo descritivo, uma variável não é mais importante do que outra. Dado que não um objetivo implícito a ser aprendido, o processo de treinamento de um modelo descritivo é conhecimento como **aprendizado não supervisionado**.

Uma tarefa de modelagem descritiva conhecida como **detecção de padrões** é utilizada, por exemplo, para identificar associações úteis nos dados. Já a tarefa de dividir um conjunto de dados em grupos homogêneos é chamada de **clustering**.

Por fim, uma classe de algoritmos de machine learning conhecida como **meta-aprendizagem** não está associada a uma tarefa de aprendizado específica, mas por outro lado está focada em como aprender mais efetivamente.

15.6 Dados de entrada e algoritmos

Abaixo, listamos os tipos gerais de algoritmos de machine learning de acordo com as tarefas de aprendizado. Primeiro os algoritmos de aprendizado supervisionado com os nomes em inglês:

Nearest Neighbor	Classificação
Naive Bayes	Classificação
Decision Trees	Classificação
Classification Rule Learners	Classificação
Linear Regression	Previsão numérica
Regression Trees	Previsão numérica
Model Trees	Previsão numérica
Neural Networks	Uso dual
Support Vector Machine	Uso dual

Abaixo, algoritmos de aprendizado não supervisionado.

Association Rules	Detecção de padrões
k-means clustering	Clustering

Por fim, os algoritmos de meta-aprendizagem.

Bagging	Uso dual
Boosting	Uso dual
Random Forests	Uso dual

15.7 Conclusão

Machine learning se origina da intersecção entre estatística, bases de dados e ciência da computação. É uma poderosa ferramenta, capaz de encontrar insights interessantes em grandes conjuntos de dados.

Conceitualmente, o aprendizado envolve a abstração dos dados em uma representação estruturada e a generalização dessa estrutura em uma ação em que sua utilidade pode ser avaliada. Em termos práticos, utiliza-se dados contendo exemplos e amostras de onde pode ser aprendido algo útil. Sumarizamos esses dados na forma de um modelo, que pode ser utilizado para previsão ou propósitos descritivos. Esses propósitos podem ser agrupar em tarefas, incluindo classificação, previsão numérica, detecção de padrões e *clustering*. Algoritmos de machine learning são escolhidos de acordo com os dados de entrada e a tarefa de aprendizagem.

Referências Bibliográficas

Casella, G. and Berger, R. L. *Inferência Estatística*. Cengage Learning, 2016.

Grolemund, G. and Wickham, H. *R for Data Science*. O'Reilly Media, 2017.

Lantz, Brett. *Machine Learning with R*. Packt Publishing, 2013.

Meyer, P. L. *Probabilidade - Aplicações á Estatística*. LTC, 2011.

Verzani, J. *Using R for Introductory Statistics*. CRC Press, 2014.

Wickham, H. *ggplot2*. Springer, 2009.

Wickham, H. *Advanced R*. CRC Press, 2014.

Wilkinson, L. *The Grammar of of Graphics*. Springer, 2005.