

13pt

 para estudantes de marketing
KU Leuven Marketing department

Traduzido por [Rodrigo Hermont Ozon](#)*

Maio, 2020

*Economista e Mestre em Desenvolvimento Econômico pela UFPR.




Sobre o autor desta tradução:

Rodrigo Hermont Ozon, economista e apaixonado por econometria, pelas aplicações de modelos econômicos a problemas reais e cotidianos vivenciados na sociedade e na realidade do mundo empresarial e corporativo.

Seus contatos podem ser acessados em:

-  [Github](#)
-  [Linkedin](#)

Resumo

O objetivo de traduzir esse tutorial consiste em facilitar o aprendizado e utilização da linguagem estatística  para os profissionais de marketing e demais áreas de negócio que precisam se adequar a uma realidade mutante e movida por um fluxo significativo de informações por todos os lados. Trabalhar e interpretar bem os dados é um desafio computacional para muitos profissionais dessa área; e esta tradução visa cobrir (ainda que superficialmente) tal lacuna. Este e-book foi escrito no [overleaf](#) com o pacote knitr [para a página interativa de autoria de KU Leven Marketing Department](#).

Ao meu amado paizão e professor pra vida inteira, Ronaldo – *”Ensina a criança no caminho em que deve andar, e, ainda quando for velho, não se desviará dele.”*

Provérbios 22:6


Sumário

1	Sobre esse tutorial	8
1.1	Download e instalação do R e RStudio	8
1.2	Obtendo familiaridade com o RStudio	8
1.2.1	Console vs. script	8
1.2.2	Comentários	9
1.2.3	Pacotes	9
2	Introdução ao R	10
2.1	Importando dados	10
2.1.1	Importando arquivos .csv	10
2.1.2	Ajustando seu diretório de trabalho	11
2.1.3	Atribuindo dados a objetos	11
2.1.4	Importando arquivos do Excel	12
2.1.5	Lendo os dados do Airbnb	12
2.2	Manipulando dataframes	14
2.2.1	Transformando variáveis	14
2.2.2	Transformações numéricas	15
2.2.3	Transformando variáveis com a função mutate	15
2.2.4	Incluindo ou excluindo e renomeando variáveis (colunas)	15
2.2.5	Incluindo ou excluindo observações (linhas)	16
2.3	O operador pipe	16
2.3.1	Uma maneira de escrever o código	16
2.4	Agrupando e resumindo	18
2.4.1	Tabelas de frequência	18
2.4.2	Estatísticas Descritivas	19
2.5	Exportando (summaries) dos dados	21
2.6	Gráficos	21
2.6.1	Diagrama de dispersão (scatterplot)	21
2.6.2	Jitter	22
2.6.3	Histograma	23
2.6.4	Transformação logarítmica	24
2.6.5	Plotando a mediana	24
2.6.6	Plota a média	25
2.6.7	Salvando imagens	26
3	Análise básica de dados: analisando dados secundários	27
3.1	Dados	27
3.1.1	Importação	27
3.1.2	Manipulação	27
3.1.3	Mesclando datasets	28
3.1.4	Recapitulando: importação e manipulação	29
3.2	Amostras independentes: teste t	29
3.2.1	ANOVA univariada	32
3.2.2	Suposição: normalidade de resíduos	33
3.2.3	Suposição: homogeneidade de variâncias	34
3.3	ANOVA	36
3.4	Teste de Tuckey de diferença significativa verdadeira	36
4	Regressão linear	37
4.1	Regressão linear simples	37
4.2	Correlação	39
4.2.1	Regressão linear múltipla, com interação	41
4.2.2	Premissas	43
4.3	Teste qui-quadrado	46
4.4	Regressão logística (opcional)	47
4.4.1	Medindo o ajuste de uma regressão logística: porcentagem classificada corretamente	49

5	Análise básica de dados: experimentos	51
5.1	Dados	53
5.1.1	Importação	53
5.2	Manipulação	53
5.2.1	Fatorar algumas variáveis	54
5.2.2	Calcular a consistência interna e a média de perguntas que medem o mesmo conceito	54
5.2.3	Recapitulando: importando e manipulando	55
5.3	Teste t	56
5.3.1	Teste t para amostras independentes	56
5.3.2	Teste t para amostras dependentes	57
5.3.3	Teste t para amostra única	58
5.4	ANOVA bivariada	58
5.4.1	Seguindo com contrastes	60
5.5	Análise de moderação: interação entre variáveis independentes contínuas e categóricas	63
5.6	ANCOVA	66
5.7	Medidas repetidas ANOVA	67
6	Análise de componentes principais para mapas perceptivos (office dataset)	70
6.1	Dados	70
6.1.1	Importação	70
6.1.2	Manipulação	70
6.1.3	Recapitulação: importação e manipulação	70
6.2	Quanto fatores devemos considerar ?	70
6.3	Análise de Componentes Principais	71
6.3.1	Cargas fatoriais	71
6.3.2	Plotando as cargas fatoriais	72
7	Análise de componentes principais para mapas perceptivos (toothpaste dataset)	74
7.1	Dados	74
7.1.1	Importação	74
7.1.2	Manipulação	74
7.1.3	Recapitulação: importação e manipulação	74
7.2	Quanto fatores devemos considerar ?	74
7.3	Análise de Componentes Principais	76
7.3.1	Cargas fatoriais	77
7.3.2	Gráfico das cargas fatoriais	77
8	Análise de cluster para segmentação	79
8.1	Dados	79
8.1.1	Recapitulação: importação e manipulação	80
8.2	Análise de Cluster	80
8.2.1	Padronizar ou não ?	80
8.2.2	Cluster hierárquico	80
8.3	Cluster não-hierárquico	82
8.4	LDA Canônico	83
9	Análise Conjunta	84
9.1	Dados	84
9.1.1	Importação	84
9.1.2	Manipulação	84
9.1.3	Recapitulando: importação e manipulação	85
9.2	Design de experimentos	85
9.3	Um respondente	92
9.3.1	Estimar valores de peça e pesos de importância	92
9.3.2	Profiles: utilitários previstos	94
9.4	Muitos respondentes	96
9.4.1	Estimar valores de peça e pesos de importância	96
9.4.2	Profiles: utilitários previstos	97
9.5	Simulação de Mercado	98

Nota do tradutor

.

Esse e-book traduzido é oriundo de  *for marketing students*.

Para que os scripts funcionem corretamente recomendo que você faça a integração do seu  Studio com o Overleaf observando  *esse tutorial aqui*.

**As traduções aqui são somente as transcrições. Não me preocupei em aperfeiçoá-las para a língua portuguesa com maior nível de clareza nos textos. As figuras e imagens não foram traduzidas.*

1 Sobre esse tutorial

Neste tutorial, exploraremos o R como uma ferramenta para analisar e visualizar dados. R é uma linguagem de programação estatística que rapidamente ganhou popularidade em muitos campos científicos. A principal diferença entre o R e outro software estatístico como o SPSS é que o R não possui interface gráfica com o usuário. Não há botões para clicar. R é executado inteiramente digitando comandos em uma interface de texto. Isso pode parecer assustador, mas, esperançosamente, no final deste tutorial, você verá como o R pode ajudá-lo a fazer uma melhor análise estatística.

Então, por que estamos usando R e não um dos muitos outros pacotes estatísticos como SPSS, SAS ou Microsoft Excel? Algumas das razões mais importantes:

Ao contrário de outros softwares, o R é gratuito e de código aberto, e sempre será! R é uma linguagem de programação e não uma interface gráfica como o SPSS. Ele realiza análises ou visualizações executando algumas linhas de código. Essas linhas de código podem ser salvas como scripts para repetição futura das análises ou visualizações. Também facilita o compartilhamento de seu trabalho com outras pessoas, que podem aprender ou corrigi-lo se houver algum erro.

R tem uma comunidade online muito ativa e útil. Quando você se depara com um problema, muitas vezes basta uma rápida pesquisa no Google para encontrar uma solução de origem coletiva.

Todas as principais empresas de pesquisa de marketing indicam que estão experimentando o R e que o R é o software do futuro. Este tutorial se concentra em análises estatísticas relevantes para estudantes de marketing. Se você quiser uma introdução mais extensa, porém acessível, ao R, confira o excelente e gratuito livro "R for Data Science". Este capítulo introdutório e o próximo são baseados na introdução ao R1, encontrada nos tutoriais do [Coding Club](#), que também possui muitos outros ótimos tutoriais de R.

Este tutorial foi escrito no RMarkdown, com a ajuda do incrível pacote [bookdown](#).

Questões? Comentários? Sugestões? Envie-me um e-mail: samuel.franssens@kuleuven.be

1.1 Download e instalação do R e R Studio

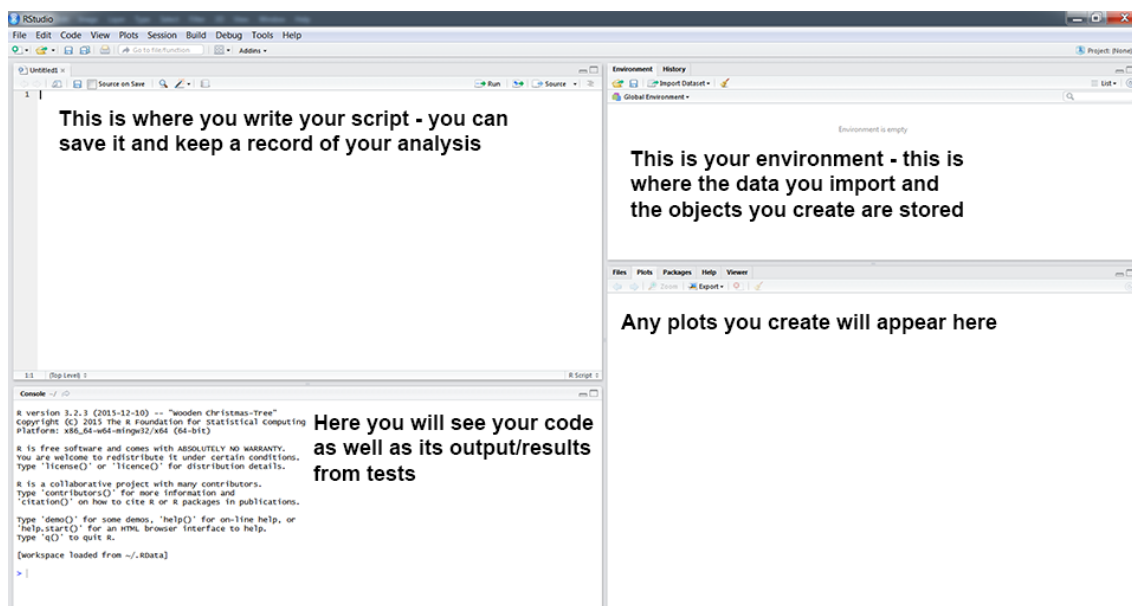
Para fornecer algumas funcionalidades extras e facilitar um pouco a transição, usaremos um programa chamado R Studio como um front-end gráfico para R.

Você pode fazer o download do R em <https://cloud.r-project.org/>. Selecione o link apropriado para o seu sistema operacional e instale o R no seu computador (no Windows, você primeiro precisa clicar em "base").

Em seguida, faça o download do R Studio em <https://www.rstudio.com/products/rstudio/download/>. Selecione o instalador para a versão gratuita e instale o R Studio (nota: você precisa ter o R instalado primeiro).

1.2 Obtendo familiaridade com o R Studio

1.2.1 Console vs. script



Ao abrir o R Studio, você verá uma janela como a acima. Você pode digitar o código diretamente no console (janela inferior esquerda) - basta digitar seu código após o prompt (`>`) e pressionar enter no final da linha para executar o código. Você também pode escrever seu código no arquivo de script (a janela superior esquerda). Se você não ver uma janela com um arquivo de script, abra uma clicando em Arquivo, Novo arquivo, R Script. Para executar uma linha de código a partir do seu script, pressione `Ctrl + R` ou `Ctrl + Enter` no Windows e `Cmd + Enter` no Mac ou use o botão 'Executar' no canto superior direito da janela do script.

O código digitado diretamente no console não será salvo pelo R. O código digitado em um arquivo de script pode ser salvo como um registro reproduzível de sua análise. Se você estiver trabalhando no console e quiser editar ou executar novamente uma linha de código anterior, pressione a seta para cima. Se você estiver trabalhando em um script, lembre-se de clicar em Salvar frequentemente (Arquivo, Salvar), para que você realmente salve o seu script!

É melhor trabalhar em arquivos de script. Também é altamente recomendável salvar seu arquivo de script em uma pasta que é automaticamente copiada pelo software de compartilhamento de arquivos que oferece a funcionalidade "versões anteriores" (o [Dropbox](#) é provavelmente o mais famoso; aqui estão algumas [alternativas](#)). Isso lhe dará a opção de restaurar versões salvas anteriormente de seus arquivos sempre que você salvar algo por engano. Como qualquer peça escrita, os roteiros se beneficiam de estrutura e clareza - a [Coding Etiquette do Coding Club](#) oferece mais conselhos sobre isso.

1.2.2 Comentários

Ao escrever um script, é muito importante adicionar comentários para descrever o que você está fazendo e por quê. Você pode fazer isso inserindo um `#` na frente de uma linha de texto. Comece seu script gravando quem está escrevendo o roteiro, a data e o objetivo principal - no capítulo introdutório, aprenderemos sobre as acomodações do Airbnb na Bélgica. Aqui está um exemplo:

```
1 # Aprendendo a importar e explorar dados e criar graficos investigando as acomodacoes do
   Airbnb na Belgica
2 # Escrito por Samuel Franssens 28/01/2018
```

1.2.3 Pacotes

As próximas linhas de código geralmente carregam os pacotes que você usará em sua análise ou visualização.

O R carrega automaticamente várias funções para executar operações básicas, mas os pacotes fornecem funcionalidade extra. Eles geralmente consistem em várias funções que podem lidar com tarefas específicas. Por exemplo, um pacote poderia fornecer funções para fazer análises de cluster ou para fazer biplots. Para instalar um pacote, digite `install.packages("nome do pacote")` (e pressione enter ao trabalhar no console ou pressione `Ctrl + Enter`, `Ctrl + R`, `Cmd + Enter` ou o botão 'Executar' ao trabalhar em um script).

Você só precisa instalar pacotes uma vez; depois, basta carregá-los usando a biblioteca (nome do pacote). Aqui, usaremos o popular pacote tidyverse que fornece muitas funções úteis e intuitivas (<https://www.tidyverse.org/>).

O pacote tidyverse é na verdade uma coleção de outros pacotes; portanto, durante a instalação ou o carregamento, você verá que vários pacotes são instalados ou carregados. Instale e carregue o pacote tidyverse executando as seguintes linhas de código:

```
1 install.packages("tidyverse") # instala o pacote tidyverse
2 library(tidyverse) # carrega o pacote tidyverse
```

Observe que há aspas ao instalar um pacote, mas não ao carregá-lo.

A instalação de um pacote normalmente produz muita saída no console. Você pode verificar se instalou um pacote com êxito, carregando o pacote. Se você tentar carregar um pacote que não foi instalado com sucesso, você receberá o seguinte erro:

```
1 library(marketing) # Estou tentando instalar o pacote inexistente 'marketing'

1 ## Error in library(marketing): there is no package called 'marketing'
```

Nesse caso, tente reinstalar o pacote.

Quando você tenta usar uma função de um determinado pacote que ainda não foi carregado, você pode receber o seguinte erro:

```
1 # agnes eh uma funcao do pacote cluster para rodar analise de cluster.
2 agnes(dist(data), metric = "euclidean", method = "ward")
```

```
1 ## Error in agnes(dist(data), metric = "euclidean", method = "ward"): could not find
  function "agnes"
```

O R nos dirá que não pode encontrar a função solicitada (neste caso, `agnes`, uma função do pacote de cluster para análises de cluster). Geralmente, isso ocorre porque você ainda não carregou (ou instalou) o pacote ao qual a função pertence.

Após instalar e carregar o pacote `tidyverse`, você poderá usar as funções incluídas no pacote `tidyverse`. Como você usará o pacote `tidyverse` com tanta frequência, é melhor sempre carregá-lo no início do seu script.

2 Introdução ao R

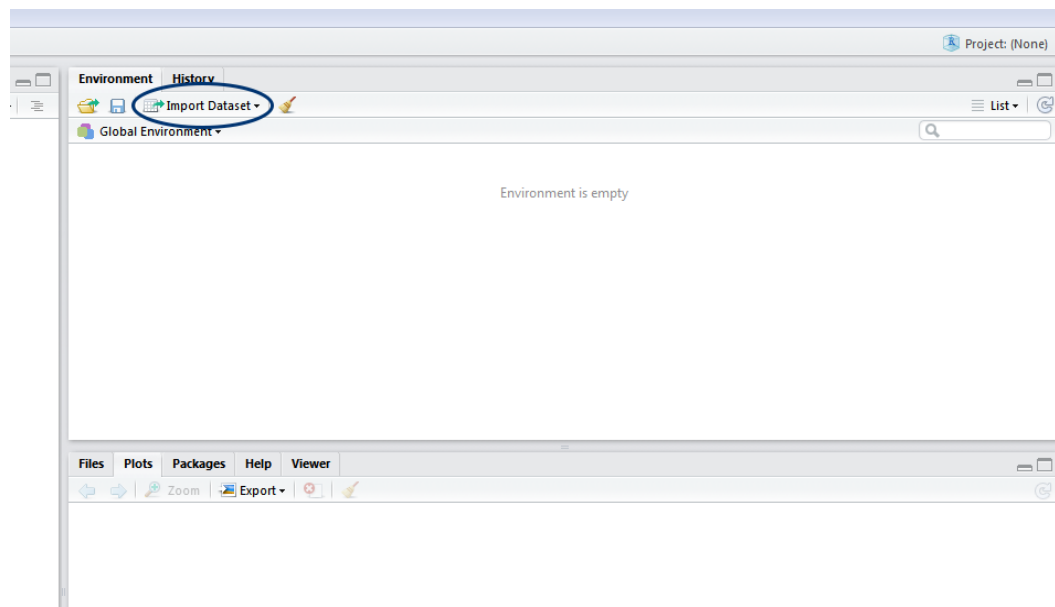
Neste capítulo introdutório, você aprenderá:

- como importar dados
- como manipular um conjunto de dados com o operador de canal
- como resumir um conjunto de dados
- como fazer gráficos de dispersão e histogramas

2.1 Importando dados

Neste capítulo, exploraremos um conjunto de dados publicamente disponível dos dados do Airbnb. Encontramos esses dados [aqui](#). (Estes são dados reais “raspados” do `airbnb.com` em julho de 2017. Isso significa que o proprietário do site criou um script para coletar automaticamente esses dados no site `airbnb.com`. Essa é uma das muitas coisas que você também pode fazer no R. Mas primeiro vamos aprender o básico.) Você pode baixar o conjunto de dados clicando com o botão direito do mouse [neste link](#), selecionando “Salvar link como...” (ou algo semelhante) e salvando o arquivo `.csv` em um diretório no disco rígido. Como mencionado na [introdução](#), é uma boa ideia salvar seu trabalho em um diretório que é automaticamente copiado pelo software de compartilhamento de arquivos. Mais tarde, salvaremos nosso script no mesmo diretório.

2.1.1 Importando arquivos `.csv`



Para importar dados para o R, clique em `Import Dataset` e depois em `From text (readr)`. Uma nova janela será exibida. Clique em `Procurar` e encontre seu arquivo de dados. Certifique-se de que a primeira linha como nomes esteja selecionada (isso diz ao R para tratar a primeira linha dos seus dados como os títulos das colunas) e clique em `Importar`. Após clicar em `importar`, o RStudio abre uma guia Visualizador. Isso mostra seus dados em uma planilha.

Alguns computadores salvam arquivos `.csv` com ponto e vírgula (;) em vez de vírgulas (,) como separadores ou “delimitadores”. Isso geralmente acontece quando o inglês não é o primeiro ou o único idioma do

seu computador. Se seus arquivos estiverem separados por ponto e vírgula, clique em Importar conjunto de dados e encontre seu arquivo de dados, mas agora escolha Ponto e vírgula no delimitador do menu suspenso.

Nota: se você não salvou o conjunto de dados clicando com o botão direito do mouse no link e selecionando “Salvar link como...”, mas clicou com o botão esquerdo do mouse no link, seu navegador pode ter acabado abrindo o conjunto de dados. Você pode salvar o conjunto de dados pressionando Ctrl + S. Observe, no entanto, que seu navegador pode acabar salvando o conjunto de dados como um arquivo .txt. É importante alterar a extensão do seu arquivo nos argumentos para o comando read_csv abaixo.

2.1.2 Ajustando seu diretório de trabalho

Depois de importar seus dados com Import Dataset, verifique a janela do console. Você verá o comando para abrir o Visualizador (View()) e, uma linha acima, verá o comando que lê os dados. Copie o comando que lê os dados do console para o seu script. No meu caso, fica assim:

```
1 tomslee_airbnb_belgium_1454_2017_07_14 <- read_csv("c:/Dropbox/work/teaching/R/data/tomslee_
  _airbnb_belgium_1454_2017-07-14.csv")
2 # Mude .csv para .txt se necessario
```

Esta linha tem a seguinte leitura (da direita para a esquerda): a função read_csv deve ler o arquivo tomslee_airbnb_belgium_1454_2017-07-14.csv no diretório c: / Dropbox / work / teaching / R / data / (você verá um diretório diferente aqui) Em seguida, R deve atribuir (<-) esses dados a um objeto chamado tomslee_airbnb_belgium_1454_2017_07_14.

Antes de explicar cada um desses conceitos, vamos simplificar esta linha de código:

```
1 setwd("c:/Dropbox/work/teaching/R/data/") # Ajusta o diretório de trabalho para onde o R
  precisa apontar para o arquivo .csv
2 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv")
3 # read_csv agora não precisa mais de um diretório e somente precisa de um nome de arquivo
4 # Atribuímos os dados a um objeto com um nome mais simples: airbnb em vez de tomslee_airbnb_
  _belgium_1454_2017_07_14
```

O comando setwd informa ao R onde está o seu diretório de trabalho. Seu diretório de trabalho é uma pasta no seu computador onde o R procurará dados, onde as plotagens serão salvas etc. Defina seu diretório de trabalho na pasta em que os dados foram armazenados. Agora, o arquivo read_csv não requer mais um diretório.

Você só precisa definir seu diretório de trabalho uma vez, na parte superior do seu script. Você pode verificar se está definido corretamente executando getwd(). Observe que em um computador com Windows, os caminhos de arquivo possuem barras invertidas que separam as pastas

```
("C: \ folder \ data").
```

No entanto, o caminho do arquivo digitado no R deve usar barras ("C: / folder / data").

Salve este script no diretório de trabalho (no meu caso: c: / Dropbox / trabalho / ensino / R / dados /). No futuro, você pode simplesmente executar essas linhas de código para importar seus dados em vez de clicar em Importar conjunto de dados (a execução de linhas de código é muito mais rápida do que apontar e clicar - uma das vantagens do uso do R).

Não se esqueça de carregar o pacote tidyverse na parte superior do seu script (mesmo antes de definir o diretório de trabalho) com a biblioteca (tidyverse).

2.1.3 Atribuindo dados a objetos

Observe a seta <- no meio da linha que importou o arquivo .csv:

```
1 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv")
```

<- é o operador de atribuição. Nesse caso, atribuímos o conjunto de dados (ou seja, os dados que lemos do arquivo .csv) a um objeto chamado airbnb. Um objeto é uma estrutura de dados. Todos os objetos que você criar serão exibidos no painel Ambiente (a janela superior direita). O R Studio fornece um atalho para escrever <-: Alt + - (no Windows). É uma boa ideia aprender esse atalho de cor.

Quando você importa dados para o R, ele se torna um objeto chamado quadro de dados. Um quadro de dados é como uma tabela ou uma planilha do Excel. Tem duas dimensões: linhas e colunas. Geralmente, as linhas representam suas observações, as colunas representam as diferentes variáveis. Quando seus dados consistem em apenas uma dimensão (por exemplo, uma sequência de números ou palavras), eles são armazenados em um segundo tipo de objeto chamado vetor. Mais tarde, aprenderemos como criar vetores.

2.1.4 Importando arquivos do Excel

R funciona melhor com arquivos .csv (valores separados por vírgula). No entanto, os dados geralmente são armazenados como um arquivo do Excel (você pode baixar o conjunto de dados do Airbnb como um arquivo do Excel aqui). O R também pode lidar com isso, mas você precisará carregar primeiro um pacote chamado `readxl` (este pacote faz parte do pacote `tidyverse`, mas não é carregado com a biblioteca `(tidyverse)` porque não é um pacote `tidyverse` principal):

```
1 library(readxl) # carrega o pacote
2
3 airbnb.excel <- read_excel(path = "tomslee_airbnb_belgium_1454_2017-07-14.xlsx", sheet = "
  Sheet1")
4 # verifique se o arquivo do Excel está salvo no seu diretório de trabalho
5
6 # você também pode deixar de fora o path = & sheet =
7 # então o comando se torna: read_excel("tomslee_airbnb_belgium_1454_2017-07-14.xlsx", "
  Sheet1")
```

`read_excel` é uma função do pacote `readxl`. São necessários dois argumentos: o primeiro é o nome do arquivo e o segundo é o nome da planilha do Excel que você deseja ler.

2.1.5 Lendo os dados do Airbnb

Nosso conjunto de dados contém informações sobre quartos na Bélgica listados no [airbnb.com](https://www.airbnb.com). Sabemos para cada sala (identificada por `room_id`): quem é o hóspede (`host_id`), que tipo de sala é (`room_type`), onde está localizada (`country`, `city`, `neighborhood`) e até a `latitude` e `longitude` exata), como muitas críticas que recebeu (`reviews`), como as pessoas estavam satisfeitas (`overall_satisfaction`), preço (`price`) e características dos quartos (`accommodates`, `bedrooms`, `bathrooms`, `minstay`).

Uma etapa realmente importante é verificar se seus dados foram importados corretamente. É uma boa prática sempre inspecionar seus dados. Você vê algum valor ausente, os números e os nomes fazem sentido? Se você começar imediatamente com a análise, corre o risco de ter que refazê-la porque os dados não foram lidos corretamente, ou pior, analisando dados errados sem perceber.

```
1 airbnb # Visualiza o conteúdo do conjunto de dados da Airbnb
2
3 ## # A tibble: 17,651 x 20
4 ##   room_id survey_id host_id room_type country city borough neighborhood
5 ##   <int>    <int>    <int> <chr>    <chr>    <chr> <chr>    <chr>
6 ## 1  5.14e6      1454    2.07e7 Shared r~ <NA>    Belg~ Gent      Gent
7 ## 2  1.31e7      1454    4.61e7 Shared r~ <NA>    Belg~ Brussel Schaarbeek
8 ## 3  8.30e6      1454    3.09e7 Shared r~ <NA>    Belg~ Brussel Elsene
9 ## 4  1.38e7      1454    8.14e7 Shared r~ <NA>    Belg~ Oosten~ Middelkerke
10 ## 5  1.83e7      1454    1.43e7 Shared r~ <NA>    Belg~ Brussel Anderlecht
11 ## 6  1.27e7      1454    6.88e7 Shared r~ <NA>    Belg~ Brussel Koekelberg
12 ## 7  1.55e7      1454    9.91e7 Shared r~ <NA>    Belg~ Gent      Gent
13 ## 8  3.91e6      1454    3.69e6 Shared r~ <NA>    Belg~ Brussel Elsene
14 ## 9  1.49e7      1454    3.06e7 Shared r~ <NA>    Belg~ Vervie~ Baelen
15 ## 10 8.50e6      1454    4.05e7 Shared r~ <NA>    Belg~ Brussel Etterbeek
16 ## # ... with 17,641 more rows, and 12 more variables: reviews <int>,
17 ## #   overall_satisfaction <dbl>, accommodates <int>, bedrooms <dbl>,
18 ## #   bathrooms <chr>, price <dbl>, minstay <chr>, name <chr>,
19 ## #   last_modified <dtm>, latitude <dbl>, longitude <dbl>, location <chr>
```

O R nos diz que estamos lidando com uma `tibble` (essa é apenas outra palavra para quadro de dados) com 17651 linhas ou observações e 20 colunas ou variáveis. Para cada coluna, é fornecido o tipo da variável: int (inteiro), chr (caractere), dbl (duplo), dtm (data e hora). Variáveis inteiras e duplas armazenam números (inteiro para números redondos, duplicam para números com decimais), variáveis de caracteres armazenam letras, variáveis de data e hora armazenam datas e / ou horas.

O R imprime apenas os dados das dez primeiras linhas e o número máximo de colunas que cabem na tela. Se, no entanto, você deseja inspecionar todo o conjunto de dados, clique duas vezes no objeto `airbnb` no painel Ambiente (a janela superior direita) para abrir uma aba Visualizador ou executar a Visualização (`airbnb`). Observe o V maiúsculo no comando Visualizar. O R sempre diferencia maiúsculas de minúsculas!

Você também pode usar o comando `print` para solicitar mais (ou menos) linhas e colunas na janela do console:

```

1 # Imprima 25 linhas (defina como Inf para imprimir todas as linhas) e defina a largura como
  100 para ver mais colunas.
2 # Observe que as colunas que nao cabem na primeira tela com 25 linhas
3 # sao impressos abaixo das 25 linhas iniciais.
4
5 print (airbnb, n = 25, width = 100)
6
7 ## # A tibble: 17,651 x 20
8 ##   room_id survey_id host_id room_type country city borough neighborhood
9 ##   <int>   <int>   <int> <chr>   <chr>   <chr> <chr>   <chr>
10 ## 1 5.14e6 1454 2.07e7 Shared r ~ <NA> Belg~ Gent Gent
11 ## 2 1.31e7 1454 4.61e7 Shared r ~ <NA> Belg~ Brussel Schaarbeek
12 ## 3 8.30e6 1454 3.09e7 Shared r ~ <NA> Belg~ Brussel Elsene
13 ## 4 1.38e7 1454 8.14e7 Shared r ~ <NA> Belg~ Oosten~ Middelkerke
14 ## 5 1.83e7 1454 1.43e7 Shared r ~ <NA> Belg~ Brussel Anderlecht
15 ## 6 1.27e7 1454 6.88e7 Shared r ~ <NA> Belg~ Brussel Koekelberg
16 ## 7 1.55e7 1454 9.91e7 Shared r ~ <NA> Belg~ Gent Gent
17 ## 8 3.91e6 1454 3.69e6 Shared r ~ <NA> Belg~ Brussel Elsene
18 ## 9 1.49e7 1454 3.06e7 Shared r ~ <NA> Belg~ Vervie~ Baelen
19 ## 10 8.50e6 1454 4.05e7 Shared r ~ <NA> Belg~ Brussel Etterbeek
20 ## 11 1.94e7 1454 1.87e7 Shared r ~ <NA> Belg~ Tournai Brunehaut
21 ## 12 1.99e7 1454 1.29e8 Shared r ~ <NA> Belg~ Brussel Etterbeek
22 ## 13 6.77e6 1454 3.50e7 Shared r ~ <NA> Belg~ Gent Gent
23 ## 14 1.39e7 1454 8.18e7 Shared r ~ <NA> Belg~ Arlon Arlon
24 ## 15 1.16e7 1454 5.00e7 Shared r ~ <NA> Belg~ Kortri~ Waregem
25 ## 16 3.65e6 1454 1.84e7 Shared r ~ <NA> Belg~ Antwer~ Boom
26 ## 17 1.20e7 1454 6.37e7 Shared r ~ <NA> Belg~ Vervie~ B llingen
27 ## 18 1.20e7 1454 6.37e7 Shared r ~ <NA> Belg~ Vervie~ B llingen
28 ## 19 4.28e5 1454 1.33e6 Shared r ~ <NA> Belg~ Gent Gent
29 ## 20 1.42e7 1454 8.61e7 Shared r ~ <NA> Belg~ Brussel Sint-Jans-M~
30 ## 21 1.93e7 1454 1.07e8 Shared r ~ <NA> Belg~ Leuven Rotselaar
31 ## 22 1.21e7 1454 6.21e7 Shared r ~ <NA> Belg~ Brugge Jabbeke
32 ## 23 4.42e6 1454 2.29e7 Shared r ~ <NA> Belg~ Ath Ath
33 ## 24 1.56e7 1454 2.05e7 Shared r ~ <NA> Belg~ Leuven Leuven
34 ## 25 1.33e6 1454 3.51e6 Shared r ~ <NA> Belg~ Tonger~ Voeren
35 ##   reviews overall_satisfa~ accommodates bedrooms bathrooms price minstay
36 ##   <int>         <dbl>         <int>   <dbl> <chr>         <dbl> <chr>
37 ## 1 9 4.5 2 1 <NA> 59 <NA>
38 ## 2 2 0 2 1 <NA> 53 <NA>
39 ## 3 12 4 2 1 <NA> 46 <NA>
40 ## 4 19 4.5 4 1 <NA> 56 <NA>
41 ## 5 5 5 2 1 <NA> 47 <NA>
42 ## 6 28 5 4 1 <NA> 60 <NA>
43 ## 7 2 0 2 1 <NA> 41 <NA>
44 ## 8 13 4 2 1 <NA> 36 <NA>
45 ## 9 2 0 8 1 <NA> 18 <NA>
46 ## 10 57 4.5 3 1 <NA> 38 <NA>
47 ## 11 1 0 4 1 <NA> 14 <NA>
48 ## 12 0 0 2 1 <NA> 37 <NA>
49 ## 13 143 5 2 1 <NA> 28 <NA>
50 ## 14 0 0 1 1 <NA> 177 <NA>
51 ## 15 1 0 4 1 <NA> 147 <NA>
52 ## 16 3 4.5 2 1 <NA> 177 <NA>
53 ## 17 0 0 2 1 <NA> 129 <NA>
54 ## 18 0 0 2 1 <NA> 140 <NA>
55 ## 19 9 5 2 1 <NA> 141 <NA>
56 ## 20 0 0 5 1 <NA> 136 <NA>
57 ## 21 1 0 2 1 <NA> 132 <NA>
58 ## 22 0 0 1 1 <NA> 117 <NA>
59 ## 23 0 0 6 1 <NA> 106 <NA>
60 ## 24 3 5 1 1 <NA> 116 <NA>
61 ## 25 13 4.5 2 1 <NA> 106 <NA>
62 ## # ... with 1.763e+04 more rows, and 5 more variables: name <chr>,
63 ## # last_modified <dtm>, latitude <dbl>, longitude <dbl>, location <chr>

```

2.2 Manipulando dataframes

2.2.1 Transformando variáveis

Fatoração

Vamos observar nosso dataset novamente:

```
1 airbnb
2
3 ## # A tibble: 17,651 x 20
4 ##   room_id survey_id host_id room_type country city borough neighborhood
5 ##   <int>    <int>    <int> <chr>    <chr>    <chr> <chr>    <chr>
6 ## 1  5.14e6    1454  2.07e7 Shared r ~ <NA>    Belg ~ Gent      Gent
7 ## 2  1.31e7    1454  4.61e7 Shared r ~ <NA>    Belg ~ Brussel Schaarbeek
8 ## 3  8.30e6    1454  3.09e7 Shared r ~ <NA>    Belg ~ Brussel Elsene
9 ## 4  1.38e7    1454  8.14e7 Shared r ~ <NA>    Belg ~ Oosten ~ Middelkerke
10 ## 5  1.83e7    1454  1.43e7 Shared r ~ <NA>    Belg ~ Brussel Anderlecht
11 ## 6  1.27e7    1454  6.88e7 Shared r ~ <NA>    Belg ~ Brussel Koekelberg
12 ## 7  1.55e7    1454  9.91e7 Shared r ~ <NA>    Belg ~ Gent      Gent
13 ## 8  3.91e6    1454  3.69e6 Shared r ~ <NA>    Belg ~ Brussel Elsene
14 ## 9  1.49e7    1454  3.06e7 Shared r ~ <NA>    Belg ~ Vervie ~ Baelen
15 ## 10 8.50e6    1454  4.05e7 Shared r ~ <NA>    Belg ~ Brussel Etterbeek
16 ## # ... with 17,641 more rows, and 12 more variables: reviews <int>,
17 ## # overall_satisfaction <dbl>, accommodates <int>, bedrooms <dbl>,
18 ## # bathrooms <chr>, price <dbl>, minstay <chr>, name <chr>,
19 ## # last_modified <dtm>, latitude <dbl>, longitude <dbl>, location <chr>
```

Vimos que `room_id` e `host_id` são "identificadores" ou rótulos que identificam as observações. São nomes (neste caso, apenas números) para as salas(quartos) e hóspedes específicos. No entanto, vemos que o R os trata como números inteiros, ou seja, como números. Isso significa que poderíamos adicionar os `room_id`'s de duas salas diferentes e obter um novo número. No entanto, isso não faria muito sentido, porque os `room_id` são apenas rótulos.

Certifique-se de que R trate os identificadores como rótulos, em vez de números, fatorando-os. Observe o operador \$. Este operador muito importante nos permite selecionar variáveis específicas de um quadro de dados, neste caso `room_id` e `host_id`.

```
1 airbnb$room_id_F <- factor(airbnb$room_id)
2 airbnb$host_id_F <- factor(airbnb$host_id)
```

Uma variável de fator é semelhante a uma variável de caractere, pois armazena letras. Os fatores são mais úteis para variáveis que podem assumir apenas um número de categorias pré-determinadas. Eles devem, por exemplo, ser usados para variáveis dependentes categóricas - por exemplo, se uma venda foi feita ou não: venda *versus* não venda. Você pode pensar em fatores como variáveis que armazenam rótulos. Os rótulos reais não são tão importantes (não nos importamos se uma venda é chamada de venda ou sucesso ou algo mais), apenas os usamos para fazer uma distinção entre categorias diferentes. É muito importante fatorar variáveis inteiras que representam variáveis independentes ou dependentes categóricas, porque, se não fatorarmos essas variáveis, elas serão tratadas como contínuas em vez de variáveis categóricas nas análises. Por exemplo, uma variável pode representar uma venda como 1 e uma não-venda como 0. Nesse caso, é importante informar ao R que essa variável deve ser tratada como uma variável categórica em vez de contínua.

As variáveis de caractere são diferentes das variáveis de fator, pois não são apenas rótulos para categorias. Um exemplo de variável de caractere seria uma variável que armazena as respostas dos entrevistados para uma pergunta em aberto. Aqui, o conteúdo real é importante (nós nos importamos se alguém descreve sua estadia no Airbnb como muito boa ou excelente ou outra coisa).

No conjunto de dados do airbnb, os `room_id` não são rigorosamente determinados de antemão, mas definitivamente são rótulos e não devem ser tratados como números. Por isso, pedimos para o R convertê-los em fatores. Vamos dar uma olhada no conjunto de dados do airbnb novamente para verificar se o tipo dessas variáveis mudou após fatorar:

```
1 airbnb
2 ## # A tibble: 17,651 x 22
3 ##   room_id survey_id host_id room_type country city borough neighborhood
4 ##   <int>    <int>    <int> <chr>    <chr>    <chr> <chr>    <chr>
5 ## 1  5.14e6    1454  2.07e7 Shared r ~ <NA>    Belg ~ Gent      Gent
6 ## 2  1.31e7    1454  4.61e7 Shared r ~ <NA>    Belg ~ Brussel Schaarbeek
7 ## 3  8.30e6    1454  3.09e7 Shared r ~ <NA>    Belg ~ Brussel Elsene
8 ## 4  1.38e7    1454  8.14e7 Shared r ~ <NA>    Belg ~ Oosten ~ Middelkerke
9 ## 5  1.83e7    1454  1.43e7 Shared r ~ <NA>    Belg ~ Brussel Anderlecht
```



```

10 ## 6 1.27e7 1454 6.88e7 Shared r~ <NA> Belg~ Brussel Koekelberg
11 ## 7 1.55e7 1454 9.91e7 Shared r~ <NA> Belg~ Gent Gent
12 ## 8 3.91e6 1454 3.69e6 Shared r~ <NA> Belg~ Brussel Elsene
13 ## 9 1.49e7 1454 3.06e7 Shared r~ <NA> Belg~ Vervie~ Baelen
14 ## 10 8.50e6 1454 4.05e7 Shared r~ <NA> Belg~ Brussel Etterbeek
15 ## # ... with 17,641 more rows, and 14 more variables: reviews <int>,
16 ## # overall_satisfaction <dbl>, accommodates <int>, bedrooms <dbl>,
17 ## # bathrooms <chr>, price <dbl>, minstay <chr>, name <chr>,
18 ## # last_modified <dtm>, latitude <dbl>, longitude <dbl>, location <chr>,
19 ## # room_id_F <fct>, host_id_F <fct>

```

Vemos que o tipo de `room_id` e `host_id` agora é `fct` (fator).

2.2.2 Transformações numéricas

Vamos dar uma olhada nas classificações das acomodações:

```

1 # Uso a funcao head para garantir que o R mostre apenas as primeiras classificacoes.
2 # Caso contrario, teremos uma lista muito longa de classificacoes..
3 head(airbnb$global_satisfac o)
4 ## [1] 4.5 0.0 4.0 4.5 5.0 5.0

```

Vemos que as classificações estão em uma escala de 0 a 5. Se preferirmos ter classificações em uma escala de 0 a 100, poderíamos simplesmente multiplicar as classificações por 20:

```

1
2 airbnb$overall_satisfaction_100 <- airbnb$overall_satisfaction * 20
3 # Perceba que criamos uma nova variavel overall_satisfaction_100.
4 # A variavel original overall_satisfaction continua inalterada.
5
6
7 # Voc tambem pode inspecionar todo o conjunto de dados com o Visualizador
8 # e veja se ha uma nova coluna a direita.
9 head(airbnb$overall_satisfaction_100)
10 ## [1] 90 0 80 90 100 100

```

2.2.3 Transformando variáveis com a função mutate

Também podemos transformar variáveis com a função `mutate`:

```

1 airbnb <- mutate(airbnb,
2                 room_id_F = factor(room_id), host_id_F = factor(host_id),
3                 overall_satisfaction_100 = overall_satisfaction * 20)

```

Isso instrui R a pegar o conjunto de dados do `airbnb`, criar uma nova variável `room_id_F` que deve ser a fatoração de `room_id`, uma nova variável `host_id_F` que deve ser a fatoração de `host_id` e uma nova variável `overall_satisfaction_100` que deve ser a satisfação geral vezes 20. O conjunto de dados com essas mutações (transformações) devem ser atribuídas ao objeto `airbnb`. Observe que não precisamos usar o operador `$` aqui, porque a função `mutate` sabe desde seu primeiro argumento (`airbnb`) onde procurar determinadas variáveis e, portanto, não precisamos especificá-lo posteriormente com `airbnb $`. Uma vantagem do uso da função `mutate` é que ela mantém bem todas as transformações desejadas dentro de um comando. Outra grande vantagem do uso do `mutate` será discutida na seção sobre o [operador pipe](#).

2.2.4 Incluindo ou excluindo e renomeando variáveis (colunas)

Se olharmos para os dados, também podemos ver que `country` é NA, o que significa que não está disponível ou está ausente. `city` é sempre a Bélgica (o que está errado porque a Bélgica é um país, não uma cidade) e o `borough` contém as informações da cidade. Vamos corrigir esses erros removendo a variável `country` de nosso conjunto de dados e renomeando `city` e `borough`. Também excluiríamos o `survey_id` porque essa variável é constante nas observações e não a usaremos no restante da análise:

```

1 airbnb <- select(airbnb, -country, -survey_id)
2 # Diga R para remover country & survey_id do quadro de dados do airbnb incluindo um sinal
3 # de menos antes dessas variaveis.
4 # Atribua novamente esse novo quadro de dados ao objeto airbnb.
5 airbnb # Agora voc ver que o country e o survey_id se foram.

```

```

6 airbnb <- rename(airbnb, country = city, city = borough)
7 # Diga ao R para renomear algumas variáveis do quadro de dados do airbnb e reatribuir esse
  novo quadro de dados ao objeto do airbnb.
8 # Nota: a sintaxe um pouco contra-intuitiva: novo nome de variável (country) = nome da
  variável antiga (city)!
9 airbnb # country = Bélgica agora e cidade se refere a cidades

```

2.2.5 Incluindo ou excluindo observações (linhas)

Criando um vetor com `c()`

Mais adiante, faremos um gráfico dos preços do Airbnb nas dez maiores cidades da Bélgica (em termos de população): Bruxelas, Antuérpia, Gent, Charleroi, Liège, Bruges, Namur, Lovaina, Mons e Aalst.

Para isso, precisamos criar um objeto de dados que tenha apenas dados para as dez maiores cidades. Para fazer isso, primeiro precisamos de um vetor com os nomes das dez maiores cidades, para que, na próxima seção, possamos dizer ao R para incluir apenas os dados dessas cidades:

```

1 topten <- c("Brussel", "Antwerpen", "Gent", "Charleroi", "Liege", "Brugge", "Namur", "Leuven", "
  Mons", "Aalst") # Cria um vetor com as 10 maiores cidades
2 topten # Mostra esse vetor.
3
4 ## [1] "Brussel" "Antwerpen" "Gent" "Charleroi" "Liege"
5 ## [6] "Brugge" "Namur" "Leuven" "Mons" "Aalst"

```

Lembre-se de que um vetor é uma estrutura de dados unidimensional (diferente de um quadro de dados que possui duas dimensões, isto é, colunas e linhas). Usamos o operador `c()` para criar um vetor que chamamos de topten. `c()` é uma abreviação de concatenar, que significa juntar as coisas. O vetor `topten` é um vetor de strings (palavras). Deve haver aspas entre as strings. Um vetor de números, no entanto, não requer aspas:

```

1 number_vector <- c(0, 2, 4, 6)
2 number_vector
3 ## [1] 0 2 4 6

```

Qualquer vetor que você criará aparecerá como um objeto no painel Ambiente (janela superior direita).

Incluindo ou excluindo observações com a função `filter`

Para armazenar apenas os dados das dez maiores cidades, precisamos do operador `%in%` do pacote `Hmisc`:

```

1 install.packages("Hmisc")
2 library(Hmisc)

```

Agora podemos usar a função de filtro para instruir o R a reter os dados apenas das dez maiores cidades:

```

1 airbnb.topten <- filter(airbnb, cidade %in% topten)
2 # Filtre o quadro de dados do airbnb para manter apenas as cidades no vetor topten.
3 # Armazene o conjunto de dados filtrado em um objeto chamado airbnb.topten.
4
5 # Então, estamos criando um novo conjunto de dados airbnb.topten, que é um subconjunto do
  conjunto de dados airbnb.
6 # Verifique o painel Ambiente para ver se o conjunto de dados airbnb.topten tem menos
  observacoes que o conjunto de dados airbnb,
7 # porque só possui dados para as dez maiores cidades.

```

2.3 O operador pipe

2.3.1 Uma maneira de escrever o código

Até agora, aprendemos (entre outras coisas) como ler um arquivo .csv e atribuí-lo a um objeto, como transformar variáveis com a função `mutate`, como remover variáveis (colunas) do nosso conjunto de dados com a função `select`, como renomear variáveis com a função `rename` e como remover observações (linhas) do nosso conjunto de dados com a função de `filter`:


```

1 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv")
2 airbnb <- mutate(airbnb, room_id_F = factor(room_id), host_id_F = factor(host_id), overall_
  satisfaction_100 = overall_satisfaction * 20)
3 airbnb <- select(airbnb, -country, -survey_id)
4 airbnb <- rename(airbnb, country = city, city = borough)
5 airbnb <- filter(airbnb, city %in% c("Brussel", "Antwerpen", "Gent", "Charleroi", "Liege", "
  Brugge", "Namur", "Leuven", "Mons", "Aalst"))

```

Ao ler este código, vemos que em cada linha substituímos o objeto `airbnb`. Não há nada de fundamentalmente errado com essa maneira de escrever, mas estamos repetindo elementos do código porque as últimas quatro linhas consistem em uma atribuição (`airbnb <-`) e em funções (`mutate`, `select`, `rename`, `filter`) que têm o mesmo primeiro argumento (o objeto `airbnb` criado na linha anterior).

Uma maneira melhor de escrever seus códigos

Existe uma maneira mais elegante de escrever código. Envolve um operador chamado *pipe* (`%>%`). Ele nos permite reescrever nossa sequência usual de operações:

```

1 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv")
2 airbnb <- mutate(airbnb, room_id_F = factor(room_id), host_id_F = factor(host_id), overall_
  satisfaction_100 = overall_satisfaction * 20)
3 airbnb <- select(airbnb, -country, -survey_id)
4 airbnb <- rename(airbnb, country = city, city = borough)
5 airbnb <- filter(airbnb, city %in% c("Brussel", "Antwerpen", "Gent", "Charleroi", "Liege", "
  Brugge", "Namur", "Leuven", "Mons", "Aalst"))

```

como:

```

1 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv") %>%
2   mutate(room_id_F = factor(room_id), host_id_F = factor(host_id), overall_satisfaction_100
  = overall_satisfaction * 20) %>%
3   select(-country, -survey_id) %>%
4   rename(country = city, city = borough) %>%
5   filter(city %in% c("Brussel", "Antwerpen", "Gent", "Charleroi", "Liege", "Brugge", "Namur", "
  Leuven", "Mons", "Aalst"))

```

Isso pode ser lido de maneira natural: “leia o arquivo csv, depois faça a mutação, selecione, renomeie e depois filtre”. Começamos lendo um arquivo .csv. Em vez de armazená-lo em um objeto intermediário, fornecemos como o primeiro argumento para a função `mutate` usando o *operador pipe*: `%>%`. É uma boa idéia aprender o atalho para `%>%` de cór: **Ctrl + Shift + M**.

A função `mutate` usa os mesmos argumentos acima (crie `room_id_F`, que deve ser uma fatoração de `room_id`, etc), mas agora não o fazemos precisamos fornecer o primeiro argumento (em qual conjunto de dados queremos que o `mutate` funcione). O primeiro argumento seria o quadro de dados resultante da leitura do arquivo .csv na linha anterior, mas isso é automaticamente transmitido como primeiro argumento a ser alterado pelo operador pipe. O operador pipe obtém a saída do que está no lado esquerdo do tubo e fornece isso como o primeiro argumento para o que está no lado direito do pipe (ou seja, a próxima linha de código).

Depois de criar novas variáveis com `mutate`, descartamos algumas variáveis com `select`. Novamente, a função `select` usa os mesmos argumentos acima (soltar país e `survey_id`), mas não fornecemos o primeiro argumento (de qual conjunto de dados devemos retirar variáveis), porque ele já é fornecido pelo pipe na linha anterior. Continuamos da mesma maneira e renomeamos algumas variáveis com `rename` e descartamos algumas observações com o `filter`.

A escrita de código com o operador de pipe explora a estrutura semelhante de `mutate`, `select`, `rename`, `filter`, que são as funções mais importantes para manipulação de dados. O primeiro argumento para todas essas funções é o quadro de dados no qual ela deve operar. Agora, esse primeiro argumento pode ser deixado de fora, porque é fornecido pelo operador pipe. No restante deste tutorial, escreveremos código usando o operador de pipe, pois melhora consideravelmente a legibilidade do nosso código.

2.4 Agrupando e resumindo

Vamos trabalhar no conjunto de dados completo novamente. Até agora, seu script deve ficar assim:

```
1 library(tidyverse)
2 setwd("c:/Dropbox/work/teaching/R/data/") # Direciona seu diretorio de trabalho
3
4 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv") %>%
5   mutate(room_id = factor(room_id), host_id = factor(host_id)) %>% # N o criamos uma nova
6     # variavel room_id_F, mas substitui mos room_id com sua fatora o. O mesmo para host_
7     # id.
8   select(-country, -survey_id) %>% # dropa country e survey_id
9   rename(country = city, city = borough) # renomeia city & borough
10
11 # Deixamos de lado a transformacao da overall_satisfaction
12 # e deixamos de fora o comando filter para garantir que nao retenhamos apenas os dados das
13 # dez cidades mais populosas
```

2.4.1 Tabelas de frequência

Cada observação em nosso conjunto de dados é uma sala ou quarto; portanto, sabemos que nossos dados contêm informações sobre 17651 quartos. Digamos que queremos saber quantos quartos existem por cidade:

```
1 airbnb%>%
2   group_by(city)%>% # Use a funcao group_by para agrupar o quadro de dados do airbnb (
3   # fornecido pelo pipe na linha anterior) por cidade
4   summarise(nr_per_city = n()) # Resuma este objeto agrupado (fornecido pelo pipe na
5   # linha anterior): peca ao R para criar uma nova variavel nr_per_city que possua o numero
6   # de observacoes em cada grupo (cidade)
7
8   ## # A tibble: 43 x 2
9   ##   city          nr_per_city
10  ##   <chr>          <int>
11  ## 1 Aalst             74
12  ## 2 Antwerpen       1610
13  ## 3 Arlon             46
14  ## 4 Ath              47
15  ## 5 Bastogne        145
16  ## 6 Brugge          1094
17  ## 7 Brussel         6715
18  ## 8 Charleroi        118
19  ## 9 Dendermonde      45
20  ## 10 Diksmuide       27
21  ## # ... with 33 more rows
```

Dizemos ao R para pegar o objeto `airbnb`, agrupá-lo por cidade e resumi-lo (`summarise()`). O resumo que queremos é o número de observações por grupo. Nesse caso, as cidades formam os grupos. Os grupos sempre serão a primeira coluna em nossa saída. Obtemos o número de observações por grupo com a função `n()`. Esses números são armazenados em uma nova coluna denominada `nr_per_city`.

Como você pode ver, essas frequências são classificadas em ordem alfabética por cidade. Em vez disso, podemos classificá-los pelo número de quartos por cidade:

```
1 airbnb %>%
2   group_by(city) %>%
3   summarise(nr_per_city = n()) %>%
4   arrange(nr_per_city) # Usa a funcao arrange para classificar em uma coluna selecionada
5
6   ## # A tibble: 43 x 2
7   ##   city          nr_per_city
8   ##   <chr>          <int>
9   ## 1 Tielt             24
10  ## 2 Diksmuide        27
11  ## 3 Moeskroen        28
12  ## 4 Roeselare        41
13  ## 5 Eeklo            43
14  ## 6 Dendermonde      45
15  ## 7 Arlon            46
16  ## 8 Ath              47
17  ## 9 Waremmes         51
18  ## 10 Sint-Niklaas    52
19  ## # ... with 33 more rows
```

Mostra a cidade com o menor número de quartos no topo. Para exibir a cidade com mais quartos no topo, classifique em ordem decrescente:

```
1 airbnb %>%
2   group_by(city) %>%
3   summarise(nr_per_city = n()) %>%
4   arrange(desc(nr_per_city)) # Classifica por ordem decrescente
5
6 ## # A tibble: 43 x 2
7 ##   city          nr_per_city
8 ##   <chr>          <int>
9 ## 1 Brussel          6715
10 ## 2 Antwerpen         1610
11 ## 3 Gent             1206
12 ## 4 Brugge           1094
13 ## 5 Li ge             667
14 ## 6 Verviers           631
15 ## 7 Oostende           527
16 ## 8 Nivelles           505
17 ## 9 Halle-Vilvoorde    471
18 ## 10 Leuven            434
19 ## # ... with 33 more rows
```

Você verá que a capital Bruxelas tem mais quartos em oferta, seguidos por Antwerpen e Gent. Observe que isso é muito parecido com trabalhar com a Tabela Dinâmica no Excel. Você poderia ter feito tudo isso no Excel, mas isso tem várias desvantagens, especialmente ao trabalhar com grandes conjuntos de dados como o nosso: você não tem registro do que clicou, de como classificou os dados e do que pode ter copiado ou excluído. No Excel, é mais fácil cometer erros acidentais sem perceber do que no R. No R, você tem seu script, para poder voltar e verificar todas as etapas de sua análise.

Nota: você também poderia ter feito isso sem o operador pipe:

```
1 airbnb.grouped <- group_by(airbnb, city)
2 airbnb.grouped.summary <- summarize(airbnb.grouped, nr_per_city = n())
3 arrange(airbnb.grouped.summary, desc(nr_per_city))
4
5 ## # A tibble: 43 x 2
6 ##   city          nr_per_city
7 ##   <chr>          <int>
8 ## 1 Brussel          6715
9 ## 2 Antwerpen         1610
10 ## 3 Gent             1206
11 ## 4 Brugge           1094
12 ## 5 Li ge             667
13 ## 6 Verviers           631
14 ## 7 Oostende           527
15 ## 8 Nivelles           505
16 ## 9 Halle-Vilvoorde    471
17 ## 10 Leuven            434
18 ## # ... with 33 more rows
```

Mas espero que você concorde que o código que usa o operador de pipe é mais fácil de ler. Além disso, sem o operador pipe, você acabará criando muitos objetos desnecessários, como `airbnb.grouped` e `airbnb.grouped.summary`.

2.4.2 Estatísticas Descritivas

Digamos que, além das frequências por cidade, também desejemos o preço médio por cidade. Queremos que isso seja classificado em ordem decrescente pelo preço médio. Além disso, agora queremos armazenar as frequências e médias em um objeto (na seção anterior, não armazenamos a tabela de frequências em um objeto):

```
1 airbnb.summary <- airbnb %>% # Armazena este resumo em um objeto chamado airbnb.summary.
2   group_by(city) %>%
3   summarise(nr_per_city = n(), average_price = mean(price)) %>% # Aqui informamos ao R para
4     criar outra variavel chamada average_price que nos fornece a media dos precos por
5     grupo (city)
6
7 arrange(desc(average_price)) # Agora organiza por average_price e mostra o maior preco
8   praticado dentre os demais
9
10 # Veja o painel de Ambiente para visualizar se ha agora um novo objeto chamado airbnb.
11   summary.
```

```

8
9 # Ao inves de apenas rodar airbnb.summary,
10 # Eu o envolvi em um comando de print e defini n como Inf para ver todas as linhas.
11
12 print(airbnb.summary, n = Inf)
13
14 ## # A tibble: 43 x 3
15 ##   city                nr_per_city average_price
16 ##   <chr>                <int>         <dbl>
17 ## 1 Bastogne             145          181.
18 ## 2 Philippeville        85          162.
19 ## 3 Verviers             631          159.
20 ## 4 Ieper                143          151.
21 ## 5 Waremmme              51          150.
22 ## 6 Dinant                286          144.
23 ## 7 Oudenaarde           110          142.
24 ## 8 Neufchateau          160          141.
25 ## 9 Ath                   47          134.
26 ## 10 Tielt                 24          129.
27 ## 11 Tongeren             173          127.
28 ## 12 Brugge              1094          126.
29 ## 13 Huy                   99          125.
30 ## 14 Marche-en-Famenne    266          124.
31 ## 15 Veurne               350          119.
32 ## 16 Eeklo                 43          115.
33 ## 17 Diksmuide             27          114.
34 ## 18 Moeskroen             28          113.
35 ## 19 Mechelen             190          112.
36 ## 20 Namur                286          111.
37 ## 21 Thuin                 81          107.
38 ## 22 Kortrijk             107          103.
39 ## 23 Oostende             527          102.
40 ## 24 Hasselt              151           99.6
41 ## 25 Maaseik              93           98.1
42 ## 26 Antwerpen          1610           95.7
43 ## 27 Aalst                 74           94.9
44 ## 28 Nivelles             505           94.1
45 ## 29 Gent                 1206           90.5
46 ## 30 Sint-Niklaas         52           86.7
47 ## 31 Virton                56           86.5
48 ## 32 Tournai              97           86.4
49 ## 33 Halle-Vilvoorde     471           85.4
50 ## 34 Dendermonde          45           81.4
51 ## 35 Mons                129           79.3
52 ## 36 Liège                 667           79.1
53 ## 37 Turnhout            130           78.1
54 ## 38 Soignies             58           77.7
55 ## 39 Charleroi           118           76.9
56 ## 40 Arlon                46           76.0
57 ## 41 Leuven              434           75.7
58 ## 42 Brussel            6715           75.1
59 ## 43 Roeselare            41           74.9

```

Talvez surpreendentemente, as três principais cidades mais caras são Bastogne, Philippeville e Verviers. Talvez o preço médio dessas cidades seja alto por causa de discrepâncias.

Vamos calcular algumas estatísticas mais descritivas para ver se nosso palpite está correto:

```

1 airbnb %>%
2   group_by(city) %>%
3   summarise(nr_per_city = n(),
4             average_price = mean(price),
5             median_price = median(price), # calcula a mediana dos preços por grupo (city)
6             max_price = max(price)) %>% # calcula o preço máximo por grupo (city)
7   arrange(desc(median_price),
8             desc(max_price)) # ordena em decendente pela mediana de preço então pelo preço
9   maximo
10
11 ## # A tibble: 43 x 5
12 ##   city                nr_per_city average_price median_price max_price
13 ##   <chr>                <int>         <dbl>         <dbl>         <dbl>
14 ## 1 Tielt                 24          129.          112           318
15 ## 2 Ieper                143          151.          111           695
16 ## 3 Verviers             631          159.          105          1769
17 ## 4 Brugge              1094          126.          105          1414
18 ## 5 Bastogne            145          181.          100          1650

```

```

18 ## 6 Veurne 350 119. 100 943
19 ## 7 Marche-en-Famenne 266 124. 100 472
20 ## 8 Dinant 286 144. 95 1284
21 ## 9 Tongeren 173 127. 95 990
22 ## 10 Neufchâteau 160 141. 95 872
23 ## # ... with 33 more rows

```

Vemos que duas das três cidades com o preço médio mais alto (Verviers e Bastogne) também estão entre as cinco principais cidades com as medianas de preços; portanto, o seu preço médio alto não se deve apenas a alguns quartos com preços extremamente altos (embora tenham o preço mais alto, quartos nessas cidades são muito caros).

2.5 Exportando (summaries) dos dados

Às vezes, você pode querer exportar dados ou um resumo dos dados. Vamos salvar nossos dados ou resumo em um arquivo .csv (no Excel, podemos convertê-lo em um arquivo do Excel, se quisermos):

```

1 # o primeiro argumento eh o objeto que voce deseja armazenar, o segundo eh o nome que voce
  # deseja atribuir ao arquivo (nao esqueca a extensao .csv)
2 # use write_csv2 quando voce tiver um computador belga (AZERTY), caso contrario, os
  # n meros decimais n o ser o armazenados como n meros
3
4 # armazenamento de dados
5 write_excel_csv(airbnb, "airbnb.csv")
6 write_excel_csv2(airbnb, "airbnb.csv")
7
8 # armazenamento de summary
9 write_excel_csv(airbnb.summary, "airbnb_summary.csv")
10 write_excel_csv2(airbnb.summary, "airbnb_summary.csv")

```

O arquivo será salvo no seu diretório de trabalho.

2.6 Gráficos

Faremos gráficos dos dados das dez cidades mais populosas da Bélgica. Se você possui o conjunto de dados completo do Airbnb em sua memória (verifique o painel Ambiente), basta filtrá-lo:

```

1 airbnb.topten <- airbnb %>%
2   filter(city %in% c("Brussel", "Antwerpen", "Gent", "Charleroi", "Liege", "Brugge", "Namur",
  # Leuven", "Mons", "Aalst")) # lembre-se de que voce tera que carregar o pacote Hmisc para
  # usar o operador %in%.

```

Se você acabou de iniciar uma nova sessão R, também pode reler o arquivo .csv executando o código na seção da seção anterior.

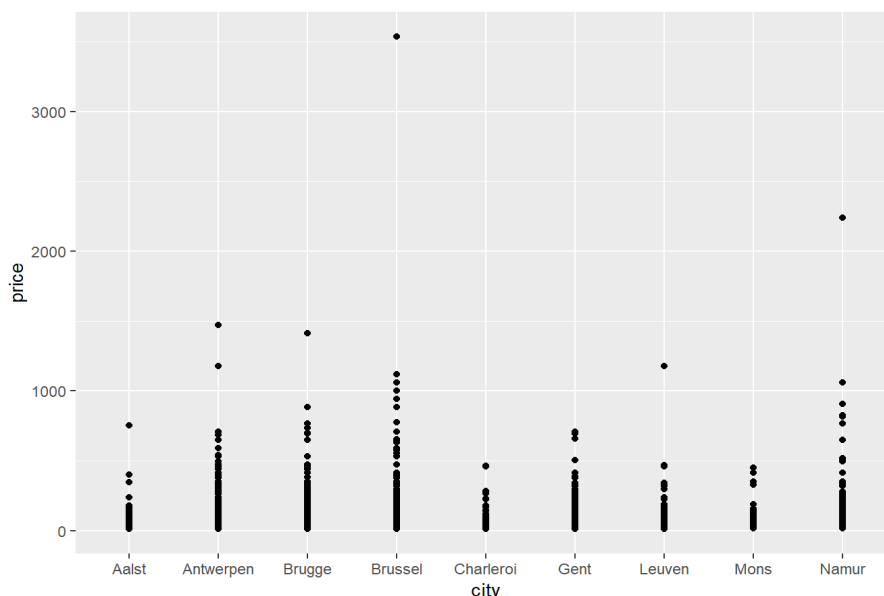
2.6.1 Diagrama de dispersão (scatterplot)

Vamos criar um scatterplot dos preços por cidade:

```

1 ggplot(data = airbnb.topten, mapping = aes(x = city, y = price)) +
2   geom_point()

```



Se tudo correr bem, uma plotagem deve aparecer no canto inferior direito da tela. As figuras são feitas com o comando `ggplot`. Na primeira linha, você diz ao `ggplot` quais dados devem ser usados para criar um gráfico e quais variáveis devem aparecer no eixo X e no eixo Y. Dizemos para colocar cidade no eixo X e preço no eixo Y. A especificação do eixo X e do eixo Y sempre deve vir como argumentos para uma função `aes`, que por sua vez é fornecida como um argumento para a função `mapping` (mapeamento). Na segunda linha, você diz ao `ggplot` para desenhar pontos (`geom_point`).

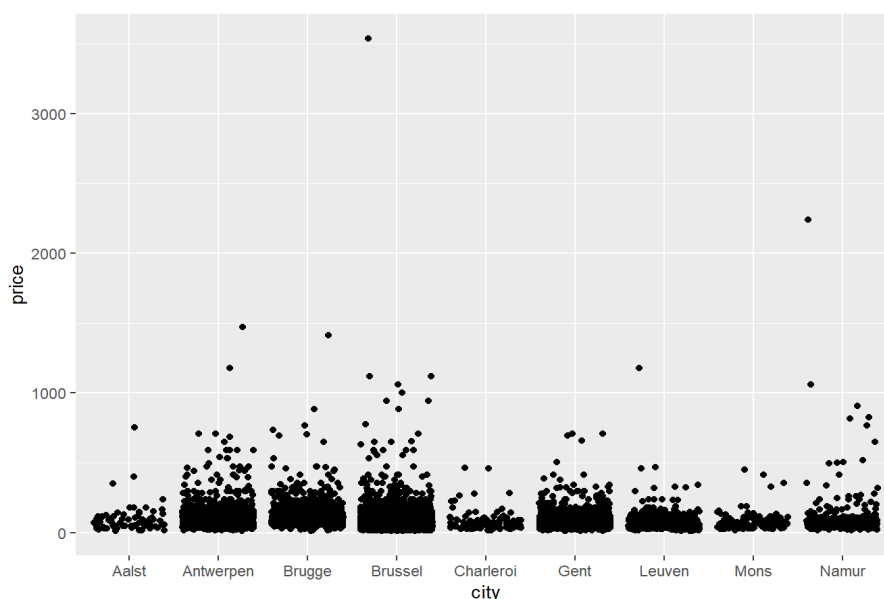
Ao criar um gráfico, lembre-se de sempre adicionar um `+` no final de cada linha de código que compõe o gráfico, exceto o último (adicionar o `+` no início de uma linha não funcionará).

O gráfico não é muito informativo porque muitos pontos são desenhados um sobre o outro.

2.6.2 Jitter

Vamos adicionar `jitter` aos nossos pontos:

```
1 ggplot(data = airbnb.topten, mapping = aes(x = city, y = price)) +
2   geom_jitter() # O mesmo código de antes mas agora mudamos geom_point para geom_jitter.
```



Em vez de solicitar pontos com `geom_point()`, agora solicitamos pontos com `jitter` adicionado com `geom_jitter()`. Jitter é um valor aleatório que é adicionado a cada coordenada X e Y, de modo que os

pontos de dados não sejam desenhados um sobre o outro. Observe que fazemos isso apenas para tornar o gráfico mais informativo (compare-o com o gráfico de dispersão anterior, onde muitos pontos de dados são desenhados um sobre o outro); não altera os valores reais em nosso conjunto de dados.

2.6.3 Histograma

Ainda não está claro. Parece que a distribuição do preço está correta. Isso significa que a distribuição do preço não é normal. Uma distribuição normal tem dois recursos principais.

Uma primeira característica é que existem mais valores próximos à média do que valores distantes da média.

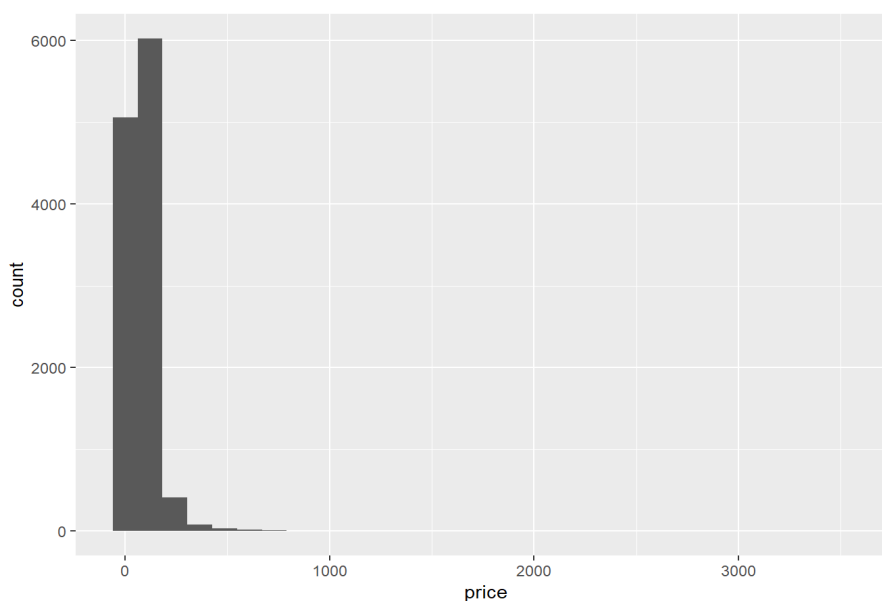
Em outras palavras, valores extremos não ocorrem com muita frequência.

Uma segunda característica é que a distribuição é simétrica. Em outras palavras, o número de valores abaixo da média é igual ao número de valores acima da média. Em uma distribuição distorcida, existem valores extremos em apenas um lado da distribuição. No caso de inclinação à direita, isso significa que existem valores extremos no lado direito da distribuição.

No nosso caso, isso significa que existem algumas listagens do Airbnb com preços muito altos. Isso aumenta a média da distribuição, de modo que as listagens não sejam mais normalmente distribuídas em torno da média.

Vamos desenhar um histograma dos preços:

```
1 ggplot(data = airbnb.topten, mapping = aes(x = price)) + # Observe que nao temos mais uma
  cidade x =. 0 preco deve estar no eixo X e as frequencias dos precos devem estar no
  eixo Y
2 geom_histogram() # Eixo Y = frequencia dos valores no eixo X
```

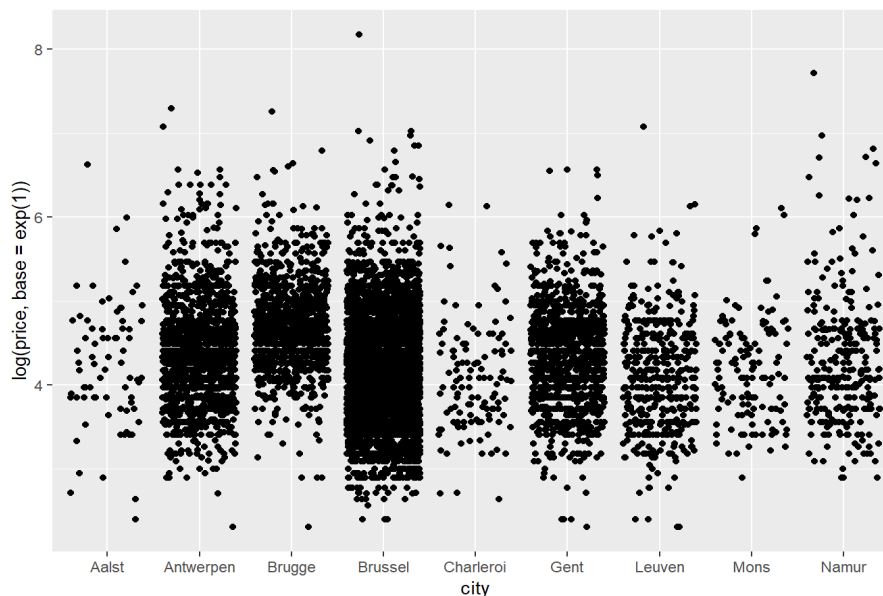


De fato, existem alguns preços extremamente altos (em comparação com a maioria dos preços), portanto, os preços estão inclinados à direita. Nota: o `stat_bin()` usando `compartimentos = 30`. Escolha um valor melhor com o aviso de largura de caixa no console que possa ser ignorado com segurança.

2.6.4 Transformação logarítmica

Como a variável `price` está inclinada à direita, podemos transformá-la em log para torná-la mais normal:

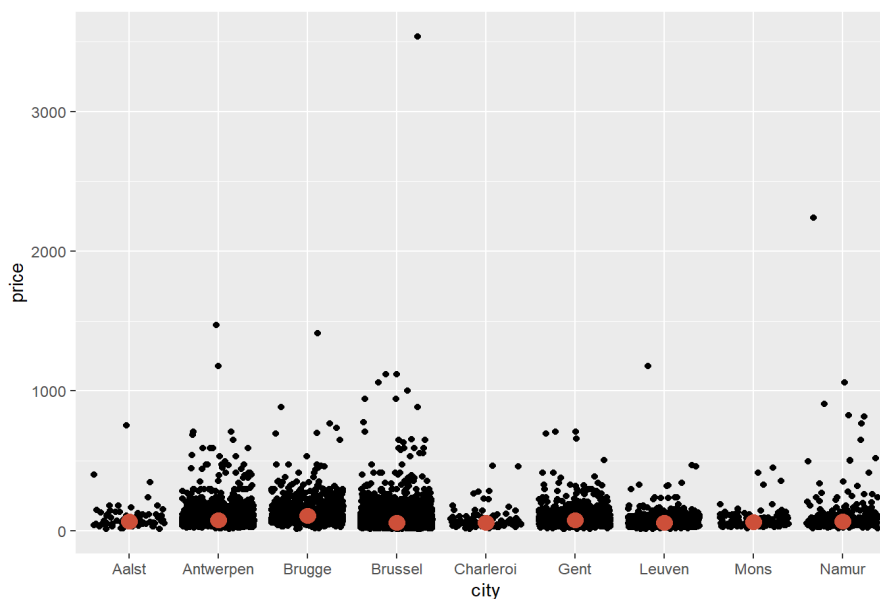
```
1 # No eixo y agora temos log(price, base=exp(1)) ao inves de price. log(price, base=exp(1))
2   = assume o log natural, i.e., o log com base = exp(1) = e.
3 ggplot(data = airbnb.topten, mapping = aes(x = city, y = log(price, base=exp(1)))) +
4   geom_jitter()
```



2.6.5 Plotando a mediana

Vamos ter uma idéia melhor da mediana de preço por cidade:

```
1 ggplot(data = airbnb.topten, mapping = aes(x = city, y = price)) +
2   geom_jitter() +
3   stat_summary(fun.y=median, colour="tomato3", size = 4, geom="point")
```

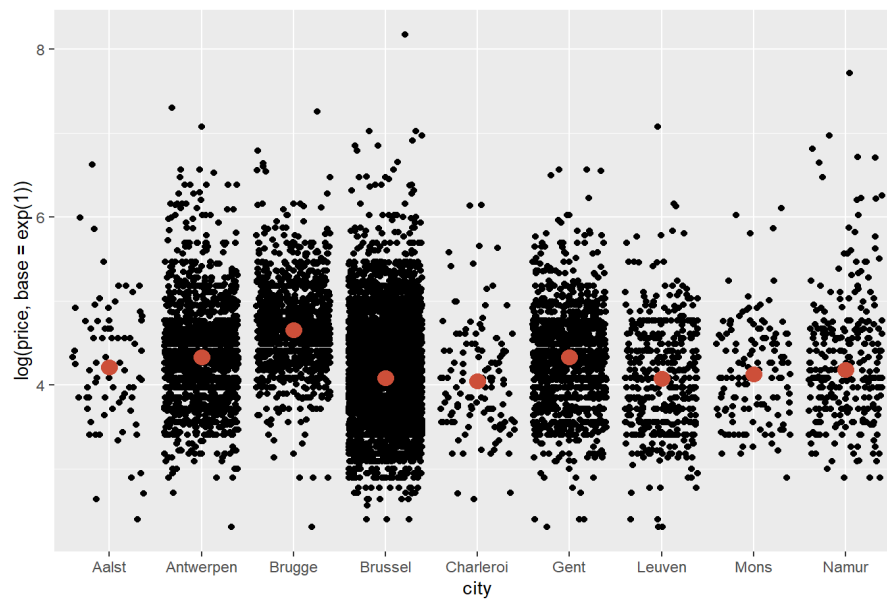


A linha de código para obter a mediana pode ser lida da seguinte forma: `stat_summary` solicitará um resumo estatístico. A estatística que queremos é a mediana em uma cor chamada `"tomato3"`, com tamanho 4. Ela deve ser representada como um "ponto". Vemos que Bruges é a cidade com o preço mediano mais alto. É muito mais fácil ver isso quando transformamos o preço por log:


```

1 ggplot(data = airbnb.topten, mapping = aes(x = city, y = log(price, base = exp(1)))) +
2   geom_jitter() +
3   stat_summary(fun.y=median, colour="tomato3", size = 4, geom="point")

```



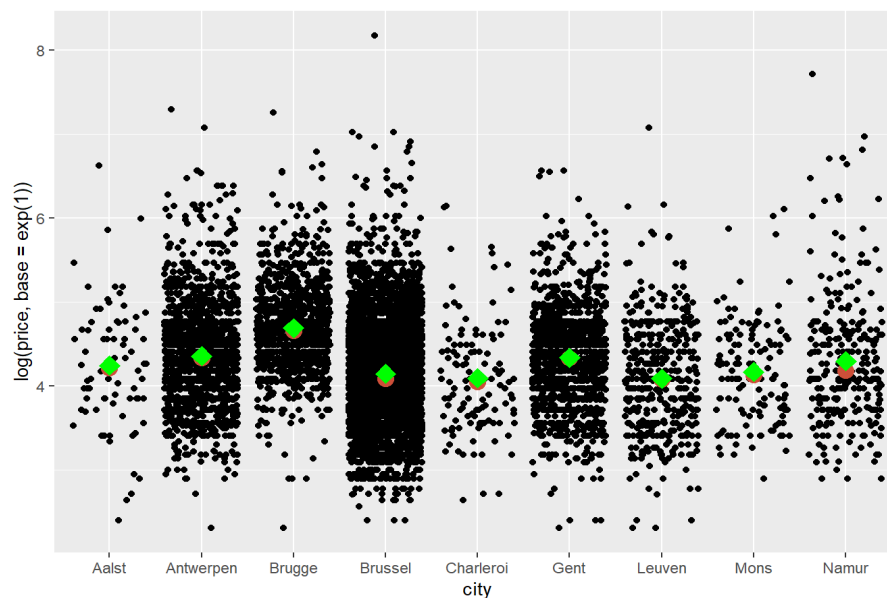
2.6.6 Plota a média

Vamos adicionar a média também, mas com uma cor e forma diferentes da média:

```

1 ggplot(data = airbnb.topten, mapping = aes(x = city, y = log(price, base = exp(1)))) +
2   geom_jitter() +
3   stat_summary(fun.y=median, colour="tomato3", size = 4, geom="point") +
4   stat_summary(fun.y=mean, colour="green", size = 4, geom="point", shape = 23, fill = "green")

```



O código para obter a média é muito semelhante ao usado para obter a mediana. Simplesmente alteramos a estatística, a cor e adicionamos a forma = 23 para obter diamantes em vez de círculos e preencher = "green" para preencher os diamantes (pontos do gráfico). Vemos que os meios e medianas são bastante semelhantes.

2.6.7 Salvando imagens

Podemos salvar esse gráfico em nosso disco rígido. Para fazer isso, clique em Exportar / Salvar como imagem. Se você não alterar o diretório, o arquivo será salvo no seu diretório de trabalho. Você pode redimensionar a plotagem e também fornecer um nome de arquivo significativo - Rplot01.png não será útil quando você tentar encontrar o arquivo posteriormente.

Uma maneira diferente (reproduzível) de salvar seu arquivo é agrupar o código nas funções `png()` e `dev.off()`:

```
1 png("price_per_city.png", width=800, height=600)
2 # Isso ira preparar o R para salvar o grafico a seguir.
3 # Fornece um nome de arquivo e dimensoes para largura e altura da figura em pixels
4
5 ggplot(data = airbnb.topten, mapping = aes(x = city, log(price, base = exp(1)))) +
6   geom_jitter() +
7   stat_summary(fun.y=mean, colour="green", size = 4, geom="point", shape = 23, fill = "
8     green") # Somente mantivemos a media aqui
9
10 dev.off() # Isso dira ao R que terminamos a plotagem e que ela deve salvar a plotagem no
11    disco rigido.
```

Embora o R tenha uma interface não gráfica, ele pode criar gráficos muito bons. Praticamente todos os pequenos detalhes no gráfico podem ser ajustados. Muitos dos gráficos que você vê em "jornalismo de dados" (por exemplo, em <https://www.nytimes.com/> ou em <http://fivethirtyeight.com/>) são feitos em R.

3 Análise básica de dados: analisando dados secundários

Neste capítulo, analisaremos os dados do Airbnb.com. A introdução tem mais informações sobre esses dados.

3.1 Dados

3.1.1 Importação

Você pode baixar o conjunto de dados clicando com o botão direito do mouse [nesse link](#), selecionando “Salvar link como...” (ou algo semelhante) e salvando o arquivo .csv em um diretório no disco rígido. Como mencionado na introdução, é uma boa ideia salvar seu trabalho em um diretório que é automaticamente copiado pelo software de compartilhamento de arquivos. Vamos importar os dados:

```
1 library(tidyverse)
2 setwd("c:/Dropbox/work/teaching/R/") # Ajusta seu diretorio de trabalho
3
4 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv") %>%
5   mutate(room_id = factor(room_id), host_id = factor(host_id)) %>%
6   select(-country, -survey_id) %>% # dropa country & survey_id, veja a introdução de por
   que fazemos isso
7   rename(country = city, city = borough) # renomeia city & borough, veja a introdução de
   por que fazemos isso
```

Não se esqueça de salvar seu script no diretório de trabalho.

3.1.2 Manipulação

Se você abrir o quadro de dados do airbnb em uma guia do Visualizador, verá que os `bathrooms` e o `minstay` são colunas vazias e que o `local` e `last_modified` não são muito informativos. Vamos remover estas variáveis:

```
1 airbnb <- airbnb %>%
2   select (-bathrooms, -minstay, -location, -last_modified)
```

Agora, dê uma olhada na variável `overall_satisfaction`:

```
1 # use head() para imprimir apenas os primeiros valores de um vetor, para evitar uma lista
   muito longa
2 # tail() imprime apenas os últimos valores de um vetor
3 head(satisfação_geral_do_airbnb$)
4
5 ## [1] 4.5 0.0 4.0 4.5 5.0 5.0
```

A segunda classificação é zero. Provavelmente, isso significa que a classificação está faltando, em vez de ser realmente zero. Vamos substituir os valores zero na `overall_satisfaction` por NA:

```
1 airbnb <- airbnb %>%
2   mutate(overall_satisfaction = replace(overall_satisfaction, overall_satisfaction == 0, NA
   ))
3
4 # crie uma variável "nova" overall_satisfaction que seja igual a overall_satisfaction com
   valores de NA em que overall_satisfaction seja igual a zero.
5
6 # Digamos que desejássemos substituir NA por 0, então o comando se tornaria: substitute(
   overall_satisfaction, is.na(overall_satisfaction), 0)
7 # overall_satisfaction == NA não funciona
8
9 head(airbnb$overall_satisfaction)
10
11 ## [1] 4.5 NA 4.0 4.5 5.0 5.0
```

3.1.3 Mesclando datasets

Posteriormente, testaremos se o preço está relacionado a determinadas características dos tipos de quartos. As características potencialmente interessantes são: `room_type`, `city`, `reviews`, `overall_satisfaction`, etc. Para torná-lo ainda mais interessante, podemos aumentar os dados, por exemplo, com dados disponíveis publicamente nas cidades. Reuni os tamanhos de população das cidades belgas mais populosas [deste site](#). Faça o download desses dados [aqui](#) e importe-os para o R:

```
1 population <- read_excel("population.xlsx", "data")
2 population
3
4 ## # A tibble: 183 x 2
5 ##   place      population
6 ##   <chr>      <dbl>
7 ## 1 Brussels  1019022
8 ## 2 Antwerpen  459805
9 ## 3 Gent      231493
10 ## 4 Charleroi  200132
11 ## 5 Liège     182597
12 ## 6 Brugge    116709
13 ## 7 Namur     106284
14 ## 8 Leuven    92892
15 ## 9 Mons      91277
16 ## 10 Aalst    77534
17 ## # ... with 173 more rows
```

Agora, queremos vincular esses dados ao nosso quadro de dados do airbnb. Isso é muito fácil no R (mas é muito difícil, por exemplo, no Excel):

```
1 airbnb.merged <- left_join(airbnb, population, by = c("city" = "place"))
2 # o primeiro argumento eh o conjunto de dados que queremos aumentar
3 # o segundo argumento eh onde encontramos os dados para aumentar o primeiro conjunto de
4 # o terceiro argumento sao as variaveis que usamos para vincular um conjunto de dados ao
  # outro (cidade eh uma variavel no airbnb, local eh uma variavel na populacao)
```

Confira as colunas mais relevantes do quadro de dados `airbnb.merged`:

```
1 airbnb.merged %>% select(room_id, city, price, population)
2
3 ## # A tibble: 17,651 x 4
4 ##   room_id city      price population
5 ##   <fct>   <chr>    <dbl>      <dbl>
6 ## 1 5141135 Gent      59      231493
7 ## 2 13128333 Brussel   53      NA
8 ## 3 8298885 Brussel   46      NA
9 ## 4 13822088 Oostende  56      NA
10 ## 5 18324301 Brussel   47      NA
11 ## 6 12664969 Brussel   60      NA
12 ## 7 15452889 Gent      41      231493
13 ## 8 3911778 Brussel   36      NA
14 ## 9 14929414 Verviers  18      52824
15 ## 10 8497852 Brussel   38      NA
16 ## # ... with 17,641 more rows
```

Vemos que há uma `population` de colunas em nosso conjunto de dados `airbnb.merged`. Você também pode ver isso no painel Ambiente: `airbnb.merged` tem uma variável a mais que `airbnb` (mas o mesmo número de observações).

Faltam dados para Bruxelas, no entanto. Isso ocorre porque Bruxelas está escrito em holandês no conjunto de dados `airbnb`, mas em inglês no conjunto de dados da `population`.

Vamos substituir Bruxelas por Brussel no conjunto de dados da `population` (e também alterar a ortografia de duas outras cidades) e vincular os dados novamente:

```
1 population <- population %>%
2   mutate(place = replace(place, place == "Brussels", "Brussel"),
3           place = replace(place, place == "Ostend", "Oostende"),
4           place = replace(place, place == "Mouscron", "Moeskroen"))
5
6 airbnb.merged <- left_join(airbnb, population, by = c("city" = "place"))
7
8 airbnb.merged %>% select(room_id, city, price, population)
9
10 ## # A tibble: 17,651 x 4
11 ##   room_id city      price population
12 ##   <fct>   <chr>    <dbl>      <dbl>
13 ## 1 5141135 Gent        59      231493
14 ## 2 13128333 Brussel      53     1019022
15 ## 3 8298885 Brussel      46     1019022
16 ## 4 13822088 Oostende     56       69011
17 ## 5 18324301 Brussel      47     1019022
18 ## 6 12664969 Brussel      60     1019022
19 ## 7 15452889 Gent        41      231493
20 ## 8 3911778 Brussel      36     1019022
21 ## 9 14929414 Verviers     18       52824
22 ## 10 8497852 Brussel      38     1019022
23 ## # ... with 17,641 more rows
```

3.1.4 Recapitulando: importação e manipulação

Aqui está o que fizemos até agora, em uma sequência ordenada de operações pipe (faça o download dos dados [aqui](#) e [aqui](#)):

```
1 library(tidyverse)
2 setwd("c:/Dropbox/work/teaching/R") # Configura seu diretorio de trabalho
3
4 airbnb <- read_csv("tomslee_airbnb_belgium_1454_2017-07-14.csv") %>%
5   mutate(room_id = factor(room_id), host_id = factor(host_id),
6           overall_satisfaction = replace(overall_satisfaction, overall_satisfaction == 0, NA
7           )) %>%
8   select(-country, -survey_id, -bathrooms, -minstay, -location, -last_modified) %>%
9   rename(country = city, city = borough)
10
11 population <- read_excel("population.xlsx", "data") %>%
12   mutate(place = replace(place, place == "Brussels", "Brussel"),
13           place = replace(place, place == "Ostend", "Oostende"),
14           place = replace(place, place == "Mouscron", "Moeskroen"))
15
16 airbnb <- left_join(airbnb, population, by = c("city" = "place"))
```

3.2 Amostras independentes: teste t

Digamos que queremos testar se os preços diferem entre cidades grandes e pequenas. Para fazer isso, precisamos de uma variável que indique se um Airbnb está em uma cidade grande ou pequena. Na Bélgica, consideramos cidades com uma população de pelo menos cem mil como grande:

```
1 airbnb <- airbnb %>%
2   mutate(large = population > 100000,
3           size = factor(large, labels = c("small", "large")))
4
5 # Nos poderiamos tambem ter escrito: mutate(size = factor(population > 100000, labels = c("
6   small", "large")))
7
8 # observando a variavel populacao
9 head(airbnb$population)
10
11 ## [1] 231493 1019022 1019022 69011 1019022 1019022
12
13 # olhando a maior variavel
14 head(airbnb$large)
15
16 ## [1] TRUE TRUE TRUE FALSE TRUE TRUE
```

```

17 # e o tamanho da variavel
18 head(airbnb$size)
19
20 ## [1] large large large small large large
21 ## Levels: small large

```

No script acima, primeiro criamos uma variável lógica (esse é outro tipo de variável; discutimos outras aqui). Chamamos essa variável de grande e é `TRUE` quando a população é maior que 100000 e `FALSE`, se não. Depois, criamos um novo tamanho de variável que é a faturação de grande porte. Observe que adicionamos outro argumento à função `factor`, ou seja, `labels`, para fornecer os valores `large` de nomes mais intuitivos. `FALSE` vem em primeiro lugar no alfabeto e obtém o primeiro rótulo pequeno, `TRUE` fica em segundo lugar no alfabeto e obtém o segundo rótulo grande.

Para saber quais cidades são grandes e quais são pequenas, podemos solicitar frequências de combinações de tamanho (grande versus pequeno) e `city` (a própria cidade). Aprendemos como fazer isso no capítulo introdutório (consulte as tabelas de frequência e as estatísticas descritivas):

```

1  airbnb %>%
2    group_by(size, city) %>%
3    summarize(count = n(), population = mean(population)) %>% # Cidades formam os grupos.
4    # Portanto, a populacao media de um grupo = a media de observacoes com a mesma populacao,
5    # porque elas vem da mesma cidade = a populacao da cidade
6    arrange(desc(size), desc(population)) %>% # maior cidade no topo
7    print(n = Inf) # mostra a distribuicao completa das frequencias
8
9  ## # A tibble: 43 x 4
10  ## # Groups:   size [3]
11  ##   size city          count population
12  ##   <fct> <chr>         <int>      <dbl>
13  ## 1 large Brussel         6715    1019022
14  ## 2 large Antwerpen        1610    459805
15  ## 3 large Gent             1206    231493
16  ## 4 large Charleroi         118    200132
17  ## 5 large Li ge             667    182597
18  ## 6 large Brugge           1094    116709
19  ## 7 large Namur             286    106284
20  ## 8 small Leuven            434     92892
21  ## 9 small Mons             129     91277
22  ## 10 small Aalst             74     77534
23  ## 11 small Mechelen         190     77530
24  ## 12 small Kortrijk         107     73879
25  ## 13 small Hasselt          151     69222
26  ## 14 small Oostende          527     69011
27  ## 15 small Sint-Niklaas      52     69010
28  ## 16 small Tournai           97     67721
29  ## 17 small Roeselare         41     56016
30  ## 18 small Verviers          631     52824
31  ## 19 small Moeskroen         28     52069
32  ## 20 small Dendermonde       45     43055
33  ## 21 small Turnhout          130     39654
34  ## 22 small Ieper            143     35089
35  ## 23 small Tongeren          173     29816
36  ## 24 small Oudenaarde        110     27935
37  ## 25 small Ath              47     26681
38  ## 26 small Arlon             46     26179
39  ## 27 small Soignies          58     24869
40  ## 28 small Nivelles          505     24149
41  ## 29 small Maaseik           93     23684
42  ## 30 small Huy              99     19973
43  ## 31 small Tielt            24     19299
44  ## 32 small Eeklo             43     19116
45  ## 33 small Marche-en-Famenne 266     16856
46  ## 34 small Diksmuide         27     15515
47  ## 35 <NA> Bastogne           145      NA
48  ## 36 <NA> Dinant             286      NA
49  ## 37 <NA> Halle-Vilvoorde    471      NA
50  ## 38 <NA> Neufchateau        160      NA
51  ## 39 <NA> Philippeville      85      NA
52  ## 40 <NA> Thuin              81      NA
53  ## 41 <NA> Veurne             350      NA
54  ## 42 <NA> Virton             56      NA
55  ## 43 <NA> Waremmes           51      NA

```

Vemos que algumas cidades têm um valor de NA para tamanho. Isso ocorre porque não temos população para essas cidades (e, portanto, também não sabemos se é uma cidade grande ou pequena). Vamos filtrar essas observações e verificar as médias e os desvios padrão de preço, dependendo do tamanho da cidade:

```
1 airbnb.cities <- airbnb %>%
2   filter(!is.na(population))
3   # Filtre as observacoes para as quais nao temos a populacao.
4   # 0 ponto de exclamacao deve ser lido como NAO. Entao, queremos manter as observacoes para
5   # as quais a populacao NAO eh NA.
6   # Visite https://r4ds.had.co.nz/transform.html#filter-rows-with-filter para conhecer mais
7   # sobre operadores logicos (veja secao 5.2.2).
8
9 airbnb.cities %>%
10  group_by(size) %>%
11  summarize(mean_price = mean(price),
12            sd_price = sd(price),
13            count = n())
14
15 ## # A tibble: 2 x 4
16 ##   size mean_price sd_price count
17 ##   <fct>      <dbl>    <dbl> <int>
18 ## 1 small      110.     122.   4270
19 ## 2 large       85.4     82.5 11696
```

Vemos que os preços são mais altos nas pequenas e nas grandes cidades, mas queremos saber se essa diferença é significativa. Um teste t de amostras independentes pode fornecer a resposta (as listagens nas grandes cidades e as listagens nas pequenas cidades são as amostras independentes), mas precisamos verificar primeiro uma suposição: as variâncias das duas amostras independentes são iguais?

```
1 install.packages("car") # Para o teste de igualdade de variancias precisaremos do pacote
2   car.
3   library(car)
4   # Teste de Levene para variancias iguais
5   # Baixo valor p significa que as variancias nao sao iguais.
6   # Primeiro argumento = variavel dependente continua, segundo argumento = variavel
7   # independente categorica.
8
9 leveneTest(airbnb.cities$price, airbnb.cities$size)
10
11 ## Levene's Test for Homogeneity of Variance (center = median)
12 ##           Df F value    Pr(>F)
13 ## group      1  139.76 < 2.2e-16 ***
14 ##           15964
15 ## ---
16 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A hipótese nula de variâncias iguais é rejeitada ($p < 0,001$), portanto, devemos continuar com um teste t que pressupõe variâncias desiguais:

```
1 # Teste se os pre os m dios das cidades grandes e pequenas s o diferentes.
2 # Indique se o teste deve assumir varia es iguais ou n o (defina var.equal = TRUE para
3   # um teste que assume varia es iguais).
4
5 t.test(airbnb.cities$price ~ airbnb.cities$size, var.equal = FALSE)
6
7 ##
8 ##   Welch Two Sample t-test
9 ##
10 ## data:  airbnb.cities$price by airbnb.cities$size
11 ## t = 12.376, df = 5762.8, p-value < 2.2e-16
12 ## alternative hypothesis: true difference in means is not equal to 0
13 ## 95 percent confidence interval:
14 ##   20.95129 28.83782
15 ## sample estimates:
16 ## mean in group small mean in group large
17 ##   110.31265      85.41809
```

Você pode relatar o seguinte: “As cidades grandes ($M = 85,42$, $DP = 82,46$) tinham um preço mais baixo ($t(5762,79) = 12,376$, $p < 0,001$, variação desigual assumida) do que as cidades pequenas ($M = 110,31$, $DP = 121,63$). ”

3.2.1 ANOVA univariada

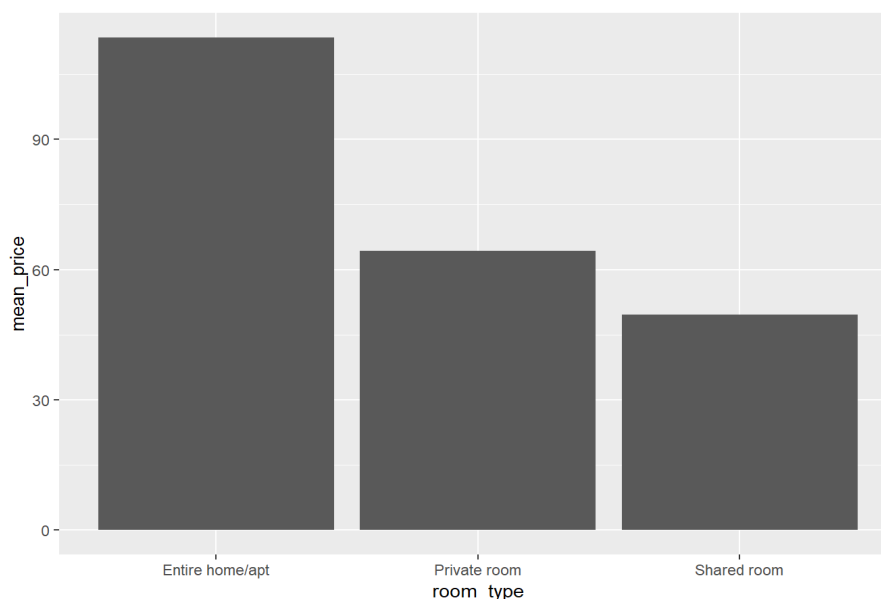
Quando sua variável independente (categórica) possui apenas dois grupos, é possível testar se as médias da variável dependente (contínua) são significativamente diferentes ou não com um teste t . Quando sua variável independente possui mais de dois grupos, você pode testar se as médias são diferentes com uma ANOVA.

Por exemplo, digamos que queremos testar se há uma diferença significativa entre os preços médios de casas e apartamentos inteiros, salas privadas e quartos compartilhados. Vamos dar uma olhada nos meios por tipo de quarto:

```
1 airbnb.summary <- airbnb %>%
2   group_by(room_type) %>%
3   summarize(count = n(), # obtenha as frequências dos diferentes tipos de quartos
4             mean_price = mean(price), # o preço medio por tipo de quarto
5             sd_price = sd(price)) # e o desvio padrao do preço por tipo de quarto
6
7 airbnb.summary
8
9 ## # A tibble: 3 x 4
10 ##   room_type      count mean_price sd_price
11 ##   <chr>         <int>     <dbl>   <dbl>
12 ## 1 Entire home/apt 11082     113.    118.
13 ## 2 Private room   6416      64.3     46.5
14 ## 3 Shared room    153      49.6     33.9
```

Também podemos traçar esses meios em um gráfico de barras:

```
1 # Ao criar um grafico de barras, o conjunto de dados que serve como entrada para o ggplot
2   eh o resumo com os meios, nao o conjunto de dados completo.
3 # (Eh por isso que salvamos o resumo acima em um objeto airbnb.summary)
4 ggplot(data = airbnb.summary, mapping = aes(x = room_type, y = mean_price)) +
5   geom_bar(stat = "identity", position = "dodge")
```



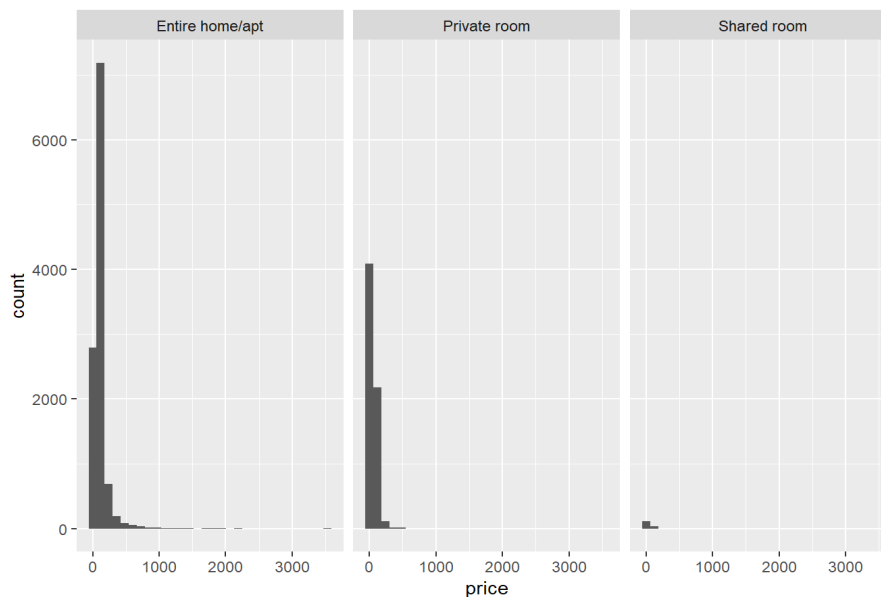
Não é de surpreender que casas ou apartamentos inteiros tenham preços mais altos do que quartos particulares, que, por sua vez, têm preços mais altos que quartos compartilhados. Também vemos que há quase o dobro de casas e apartamentos inteiros do que quartos privados disponíveis e quase não há quartos compartilhados disponíveis. Além disso, o desvio padrão é muito mais alto na categoria de casas ou apartamentos inteiros do que nas categorias de quarto particular ou compartilhado.

Uma ANOVA pode testar se há diferenças significativas nos preços médios por tipo de quarto. Porém, antes de executar uma ANOVA, precisamos verificar se as premissas da ANOVA são atendidas.

3.2.2 Suposição: normalidade de resíduos

A primeira suposição é que a variável dependente (`price`) é normalmente distribuída em cada nível da variável independente (`room_type`). Primeiro, vamos inspecionar visualmente se essa suposição será válida:

```
1 # Ao criar um histograma, o conjunto de dados que serve como entrada para o ggplot eh o
2 # conjunto de dados completo, nao o resumo com os meios
3
4 ggplot(data = airbnb, mapping = aes(x = price)) + # Queremos price no eixo x.
5   facet_wrap(~ room_type) + # Queremos que isso seja dividido por room_type.
6   #facet_wrap garantira que o ggplot crie paineis diferentes no seu gr fico.
7   geom_histogram() # geom_histogram garante que as frequencias dos valores no eixo X sejam
8   ## `stat_bin()` using `bins = 30`. Pega o melhor valor com `binwidth`.
```



Vemos que há inclinação correta para cada tipo de sala. Também podemos testar formalmente, dentro de cada tipo de sala, se as distribuições são normais com o teste Shapiro-Wilk. Por exemplo, para as salas compartilhadas:

```
1 airbnb.shared <- airbnb %>%
2   filter(room_type == "Shared room") # reter dados apenas das salas compartilhadas
3
4 shapiro.test(airbnb.shared$price)
5
6 ##
7 ## Shapiro-Wilk normality test
8 ##
9 ## data:  airbnb.shared$price
10 ## W = 0.83948, p-value = 1.181e-11
```

O valor-p deste teste é extremamente pequeno, portanto a hipótese nula de que a amostra provém de uma distribuição normal deve ser rejeitada. Se tentarmos o teste Shapiro-Wilk para as salas privadas:

```
1
2 airbnb.private <- airbnb %>%
3   filter(room_type == "Private room") # armazenar dados apenas das salas compartilhadas
4
5 shapiro.test(airbnb.private$price)
6
7 ## Error in shapiro.test(airbnb.private$price): sample size must be between 3 and 5000
```

Ocorreu um erro ao dizer que o tamanho da amostra é muito grande. Para contornar esse problema, podemos tentar o teste Anderson-Darling do pacote `nortest`:

```

1 install.packages("nortest")
2 library(nortest)
3 ad.test(airbnb.private$price)
4
5 ##
6 ## Anderson-Darling normality test
7 ##
8 ## data:  airbnb.private$price
9 ## A = 372.05, p-value < 2.2e-16

```

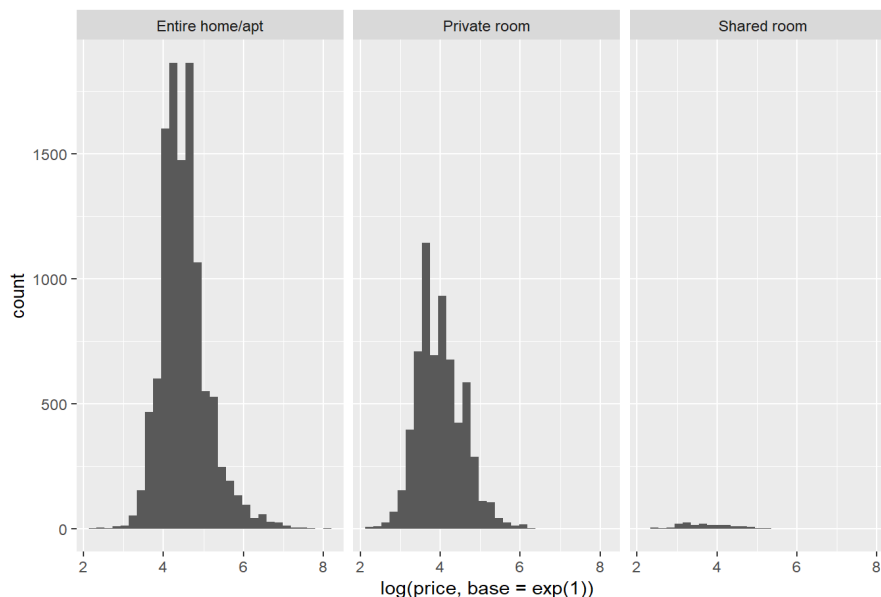
Mais uma vez, rejeitamos a hipótese nula de normalidade. Deixo como exercício para testar a normalidade dos preços de casas e apartamentos inteiros.

Agora que sabemos que a suposição de normalidade é violada, o que podemos fazer? Podemos considerar transformar nossa variável dependente com uma transformação de log:

```

1 ggplot(data=airbnb, mapping=aes(x=log(price, base = exp(1)))) + # Queremos o preço
  transformado em log no eixo X.
2 facet_wrap(~ room_type) + # Queremos que isso seja dividido por room_type. Facet_wrap
  garantira que o ggplot crie painéis diferentes no seu grafico.
3 geom_histogram() # geom_histogram garante que as frequencias dos valores no eixo X
  sejam plotadas.
4
5 ## `stat_bin()` using `bins = 30`. Pega um valor melhor com `binwidth`.

```



Como você pode ver, uma transformação de log normaliza uma distribuição inclinada à direita. Poderíamos então executar a ANOVA na variável dependente transformada em log. No entanto, na realidade, muitas vezes é seguro ignorar violações da suposição de normalidade (a menos que você esteja lidando com pequenas amostras, o que nós não somos). Vamos simplesmente continuar com o preço não transformado como variável dependente.

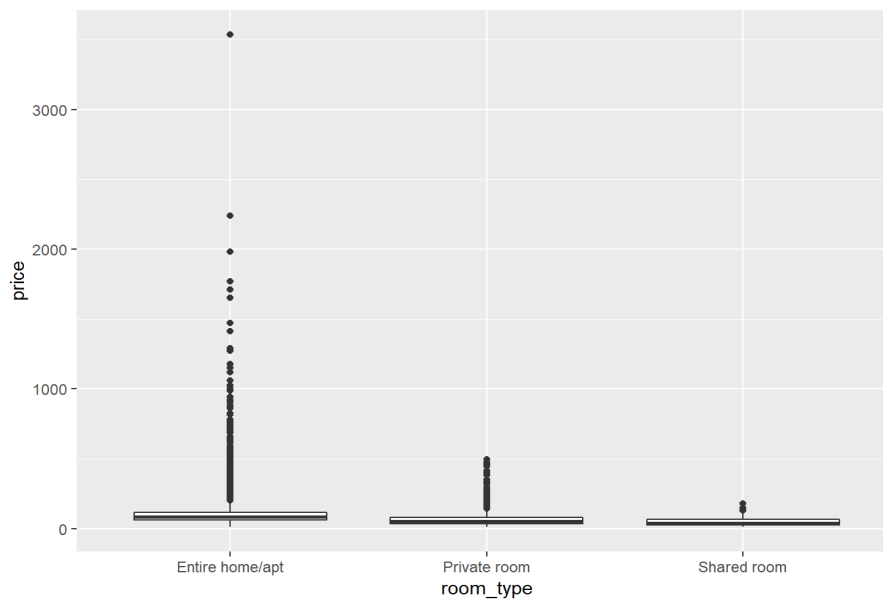
3.2.3 Suposição: homogeneidade de variâncias

Uma segunda suposição que precisamos verificar é se as variações de nosso preço variável dependente são iguais nas categorias de nossa variável independente `room_type`. Normalmente, um gráfico boxplot é informativo:

```

1 ggplot(data = airbnb, mapping = aes(x = room_type, y = price)) +
2   geom_boxplot()

```



Mas, neste caso, os intervalos interquartis (as alturas das caixas), que normalmente nos dariam uma idéia da variação dentro de cada tipo de quarto, são muito estreitos. Isso ocorre porque o intervalo de valores Y a ser plotado é muito amplo devido a alguns valores extremos. Se observarmos os desvios padrão, porém, veremos que estes são muito maiores para todos as salas e apartamentos do que para os quartos privativo e compartilhado:

```
1 airbnb %>%
2   group_by(room_type) %>%
3   summarize(count = n(), # obtenha as frequencias dos diferentes tipos de quartos
4             mean_price = mean(price), # o preco medio por tipo de quarto
5             sd_price = sd(price)) # e o desvio padrao do preco por tipo de quarto
6
7 ## # A tibble: 3 x 4
8 ##   room_type      count mean_price sd_price
9 ##   <chr>         <int>     <dbl>   <dbl>
10 ## 1 Entire home/apt 11082      113.    118.
11 ## 2 Private room    6416       64.3    46.5
12 ## 3 Shared room     153       49.6    33.9
```

Também podemos realizar um teste formal de homogeneidade de variâncias. Para isso, precisamos da função `leveneTest` do pacote `car`:

```
1 install.packages("car") # Para o teste de variancias iguais, precisamos de um pacote
2                           chamado car. Instalamos isso antes, portanto, nao eh necessario reinstala-lo se voce ja
3                           o tiver feito.
4
5 library(car)
6
7 #Teste de Levene de variancias iguais.
8 # Valor baixo de p significa que as variancias nao sao iguais.
9 # Primeiro argumento = variavel dependente continua, segundo argumento = variavel
10   independente categorica.
11
12 leveneTest(airbnb$price, airbnb$room_type)
13
14 ## Levene's Test for Homogeneity of Variance (center = median)
15 ##      Df F value    Pr(>F)
16 ## group  2  140.07 < 2.2e-16 ***
17 ##      17648
18 ## ---
19 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como o valor p é extremamente pequeno, rejeitamos a hipótese nula de variâncias iguais. Assim como no pressuposto da normalidade, as violações do pressuposto de variâncias iguais podem, no entanto, ser frequentemente ignoradas e o faremos neste caso.

3.3 ANOVA

Para realizar uma ANOVA, precisamos instalar alguns pacotes:

```
1 install.packages("remotes") #O pacote de controles remotos nos permite instalar pacotes
  armazenados no GitHub, um site para desenvolvedores de pacotes.
2 install.packages("car") #Também precisaremos do pacote do carro para executar a ANOVA (
  não é necessário reinstalá-lo se você já tiver feito isso).
3
4 library(remotes)
5 install_github('samuelfrassens/type3anova') # Instala o pacote type3anova. Esta e as
  etapas anteriores precisam ser executadas apenas uma vez.
6
7 library(type3anova) # Carregue o pacote type3anova.
```

Agora podemos prosseguir com a ANOVA verdadeira:

```
1 # Primeiro cria um modelo linear
2 # A formula lm() toma os argumentos de dados
3 # A formula tem a seguinte sintaxe: variável dependente ~ variável(s) independente
4
5 linearmodel <- lm(price ~ room_type, data=airbnb)
6
7 # Em seguida, peça a saída no formato ANOVA. Isso fornece a soma dos quadrados do Tipo
  III.
8 # Observe que isso é diferente da anova (modelo linear), que fornece a soma dos quadrados
  do tipo I.
9
10 type3anova(linearmodel)
11
12 ## # A tibble: 3 x 6
13 ##   term                ss    df1    df2    f      pvalue
14 ##   <chr>              <dbl> <dbl> <int> <dbl>    <dbl>
15 ## 1 (Intercept)      7618725.     1 17648  803.  7.31e-173
16 ## 2 room_type       10120155.     2 17648  534.  1.02e-225
17 ## 3 Residuals      167364763. 17648 17648    NA      NA
```

Nesse caso, o valor-p associado ao efeito de `room_type` é praticamente 0, o que significa que rejeitamos a hipótese nula de que o preço médio é igual para cada `room_type`. Você pode relatar o seguinte: “Houve diferenças significativas entre os preços médios das diferentes tipos de salas ($F(2, 17648) = 533,57, p < 0,001$).”

3.4 Teste de Tuckey de diferença significativa verdadeira

Observe que a ANOVA testa a hipótese nula de que as médias em todos os nossos grupos são iguais. A rejeição desta hipótese nula significa que há uma diferença significativa em pelo menos um dos possíveis pares de médias (ou seja, em casa / apartamento inteiro vs. privado e / ou em casa / apartamento inteiro vs. compartilhado e / ou privado) vs. compartilhado). Para ter uma ideia de qual par de médias contém uma diferença significativa, podemos acompanhar o teste de Tukey, que nos dará todas as comparações pareadas.

O teste de Tukey corrige os valores de p para cima - portanto, é mais conservador decidir que algo é significativo - porque as comparações são post-hoc ou exploratórias:

```
1 TukeyHSD(aov(price ~ room_type, data=airbnb),
2           "room_type") # O primeiro argumento é um objeto "aov", o segundo é a nossa
  variável independente.
3
4 ##   Tukey multiple comparisons of means
5 ##   95% family-wise confidence level
6 ##
7 ## Fit: aov(formula = price ~ room_type, data = airbnb)
8 ##
9 ## $room_type
10 ##
11 ##           diff          lwr          upr      p adj
12 ## Private room-Entire home/apt -49.11516 -52.69593 -45.534395 0.000000
13 ## Shared room-Entire home/apt  -63.79178 -82.37217 -45.211381 0.000000
14 ## Shared room-Private room    -14.67661 -33.34879   3.995562 0.155939
```

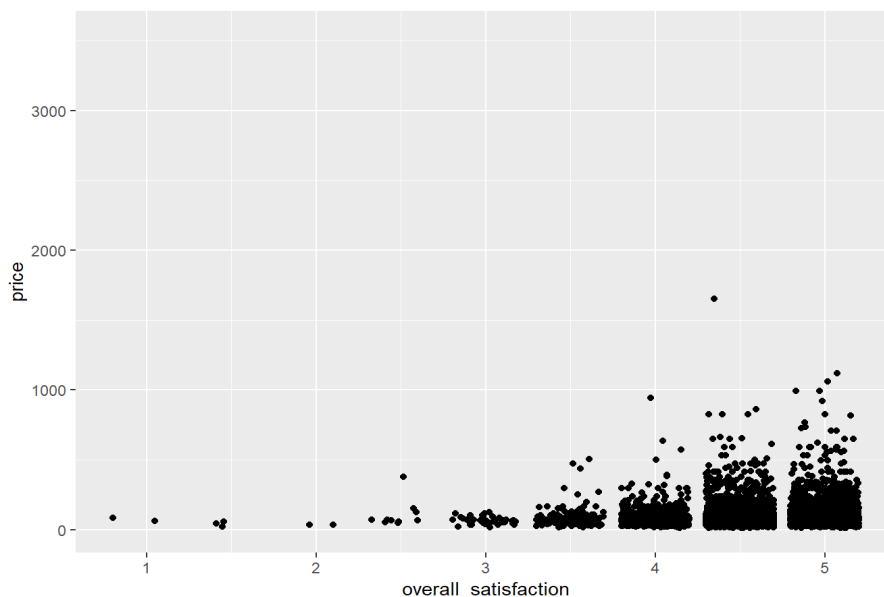
Isso nos mostra que não há diferença significativa no preço médio de quartos compartilhados e privados, mas que quartos compartilhados e quartos particulares diferem significativamente de casas e apartamentos inteiros.

4 Regressão linear

4.1 Regressão linear simples

Digamos que desejamos prever o preço com base em várias características do quarto. Vamos começar com um caso simples em que prevemos preço com base em um preditor: `overall_satisfaction` (satisfação geral). A satisfação geral é a classificação que uma listagem recebe no airbnb.com. Vamos fazer um gráfico de dispersão primeiro:

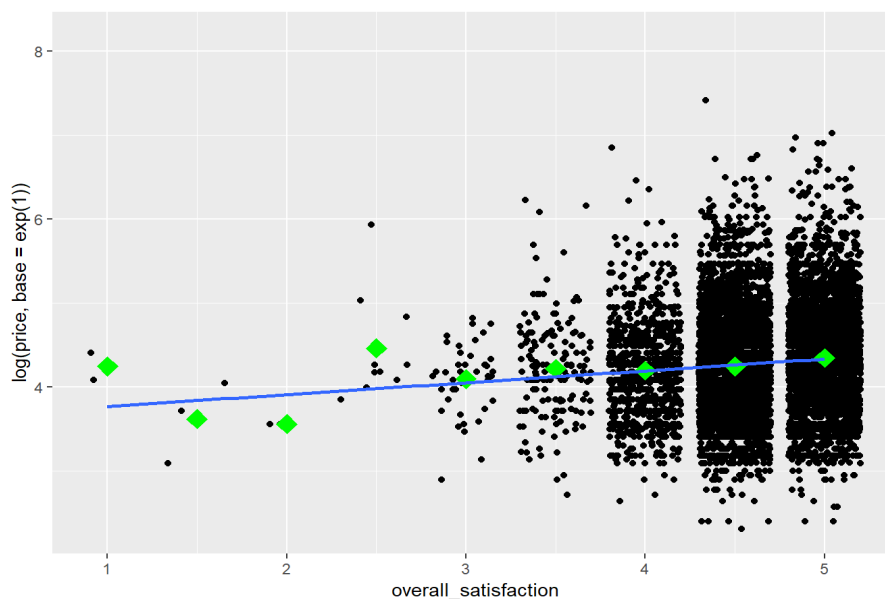
```
1 ggplot(data = airbnb, mapping = aes(x = overall_satisfaction, y = price)) +
2   geom_jitter() # jitter em vez de pontos, caso contrario, muitos pontos sao desenhados um
3                 sobre o outro
4 ## Warning: Removed 7064 rows containing missing values (geom_point).
```



(Recebemos uma mensagem de erro informando que várias linhas foram removidas. Essas são as linhas com valores ausentes para a `overall_satisfaction`, portanto, não há necessidade de se preocupar com essa mensagem de erro. Consulte as [manipulações de dados](#) para saber por que faltam valores para a `overall_satisfaction`.)

Os outliers de preço reduzem a informatividade do gráfico, portanto, vamos transformar a variável `price`. Também vamos adicionar alguns meios ao gráfico, como aprendemos [aqui](#), e uma linha de regressão:

```
1 ggplot(data = airbnb, mapping = aes(x = overall_satisfaction, y = log(price, base = exp(1))
2   )) +
3   geom_jitter() + # jitter em vez de pontos, caso contrario, muitos pontos sao desenhados
4                 um sobre o outro
5   stat_summary(fun.y=mean, colour="green", size = 4, geom="point", shape = 23, fill = "
6                 green") + # medias
7   stat_smooth(method = "lm", se=FALSE) # reta de regressao
```



Vemos que o preço tende a aumentar um pouco com maior satisfação. Para testar se a relação entre preço e satisfação é realmente significativa, podemos realizar uma regressão simples (simples refere-se ao fato de haver apenas um preditor):

```
1 linearmodel <- lm(price ~ overall_satisfaction, data = airbnb) # criamos um modelo linear.
  0 primeiro argumento eh o modelo que assume a forma de variavel dependente - variavel (
  s) independente (s). O segundo argumento sao os dados que devemos considerar.
2
3 summary(linearmodel) # solicite um resumo dos resultados desse modelo linear
4
5 ##
6 ## Call:
7 ## lm(formula = price ~ overall_satisfaction, data = airbnb)
8 ##
9 ## Residuals:
10 ##      Min       1Q   Median       3Q      Max
11 ## -80.51  -38.33  -15.51   14.49 1564.67
12 ##
13 ## Coefficients:
14 ##              Estimate Std. Error t value Pr(>|t|)
15 ## (Intercept)      29.747      8.706   3.417 0.000636 ***
16 ## overall_satisfaction 12.353      1.864   6.626 3.62e-11 ***
17 ## ---
18 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
19 ##
20 ## Residual standard error: 71.47 on 10585 degrees of freedom
21 ## (7064 observations deleted due to missingness)
22 ## Multiple R-squared:  0.00413,    Adjusted R-squared:  0.004036
23 ## F-statistic: 43.9 on 1 and 10585 DF, p-value: 3.619e-11
```

Vemos dois parâmetros neste modelo:

- β_0 ou intercepto (29.75)
- β_1 inclinação de overall_satisfaction (12.35)

Esses parâmetros têm as seguintes interpretações. A interceptação (β_0) é o preço estimado para uma observação com satisfação geral igual a zero. A inclinação (β_1) é o aumento estimado do preço para cada aumento na satisfação geral. Isso determina a inclinação da linha de regressão ajustada no gráfico. Portanto, para uma listagem com uma satisfação geral de, por exemplo, 3,5, o preço estimado é $29,75 + 3,5 \times 12,35 = 72,98$.

A inclinação é positiva e significativa. Você pode relatar o seguinte: “Havia uma relação significativamente positiva entre preço e satisfação geral ($\beta = 12,35$, $t(10585) = 6,63$, $p < 0,001$).”

Na saída, também obtemos informações sobre o modelo geral.

O modelo vem com um valor F de 43,9 com 1 grau de liberdade no numerador e 10585 graus de liberdade no denominador. Essa estatística F nos diz se nosso modelo com um preditor (`overall_satisfaction`) prediz a

variável dependente (`price`) melhor do que um modelo sem preditores (o que simplesmente preveria a média do preço para todos os níveis de satisfação geral). Os graus de liberdade nos permitem encontrar o valor p correspondente ($<0,001$) da estatística F (43,9). Os graus de liberdade no numerador são iguais ao número de preditores em nosso modelo. Os graus de liberdade no denominador são iguais ao número de observações menos o número de preditores menos um. Lembre-se de que temos 10587 observações para as quais temos valores para `price` e `overall_satisfaction`. Como o valor p é menor que 0,05, rejeitamos a hipótese nula de que nosso modelo não prediz melhor a variável dependente do que um modelo sem preditores. Observe que, no caso de regressão linear simples, o valor p do modelo corresponde ao valor p do preditor único. Para modelos com mais preditores, não existe essa correspondência.

Por fim, observe também a estatística do R quadrado do modelo. Esta estatística é igual a 0,004. Essa estatística nos diz quanto da variação na variável dependente é explicada por nossos preditores. Quanto mais preditores você adicionar a um modelo, maior será o R quadrado.

4.2 Correlação

Observe que na regressão linear simples, a inclinação do preditor é uma função da correlação entre o preditor e a variável dependente. Podemos calcular a correlação da seguinte maneira:

```
1 # Certifique-se de incluir o argumento use, caso contrario, o resultado sera NA devido aos
  # valores ausentes na overall_satisfaction.
2 # O argumento use instrui o R para calcular a correlacao com base apenas nas observacoes
  # para as quais temos dados sobre price e overall_satisfaction.
3
4 cor(airbnb$price, airbnb$overall_satisfaction, use = "pairwise.complete.obs")
5
6 ## [1] 0.06426892
```

Vemos que a correlação é positiva, mas muito baixa ($r = 0,064$).

Elevando ao quadrado essa correlação, você obterá o R quadrado de um modelo com apenas esse preditor ($0,064 \times 0,064 = 0,004$).

Ao lidar com múltiplos preditores (como na próxima seção), podemos gerar uma matriz de correlação. Isso é especialmente útil ao verificar a multicolinearidade. Digamos que desejamos que as correlações entre, `price`, `overall_satisfaction`, `reviews`, `accommodates`:

```
1 airbnb.corr <- airbnb%>%
2   filter(! is.na (overall_satisfaction))%>% # caso contrario, voc vera NAs no
  # resultado
3   select(price, overall_satisfaction, reviews, accommodates)
4
5 cor(airbnb.corr) # obter a matriz de correlacao
6
7 cor(airbnb.corr) # obtenha a matriz de correlacao
8
9 ##
10 ## price overall_satisfaction reviews
11 ## price 1.0000000 0.06426892 -0.05827489
12 ## overall_satisfaction 0.06426892 1.00000000 0.03229339
13 ## reviews -0.05827489 0.03229339 1.00000000
14 ## accommodates 0.63409855 -0.04698709 -0.03862767
15 ##
16 ## price 0.63409855
17 ## overall_satisfaction -0.04698709
18 ## reviews -0.03862767
19 ## accommodates 1.0000000
```

Você pode visualizar facilmente essa matriz de correlação:

```
1 install.packages("corrplot") # instala e carrega o pacote corrplot
2 library(corrplot)
3
4 corrplot(cor(airbnb.corr), method = "number", type = "lower", bg = "grey") # apresente numa
  tabela
```



As cores das correlações dependem de seus valores absolutos.

Você também pode obter valores de p para as correlações ($p < 0,05$ indica que a correlação difere significativamente de zero):

```
1 # O comando para os valores-p eh cor.mtest(airbnb.corr)
2 # Mas queremos apenas os valores-p, portanto $ p
3 # E arredondamos para cinco digitos, portanto arredondamos (, 5)
4
5 round(cor.mtest(airbnb.corr)$p, 5)
6 ##      [,1]      [,2]      [,3]      [,4]
7 ## [1,]      0 0.00000 0.00000 0e+00
8 ## [2,]      0 0.00000 0.00089 0e+00
9 ## [3,]      0 0.00089 0.00000 7e-05
10 ## [4,]      0 0.00000 0.00007 0e+00
```


4.2.1 Regressão linear múltipla, com interação

Frequentemente, estamos interessados em interações entre preditores (por exemplo, `overall_satisfaction` e `reviews`). Uma interação entre preditores nos diz que o efeito de um preditor depende do nível do outro preditor:

```

1 # overall_satisfaction + reviews: a interacao nao eh incluida como preditor
2 # overall_satisfaction * reviews: a interacao entre os dois preditores eh incluida como
  preditora
3
4 ##
5 ## Call:
6 ## lm(formula = price ~ overall_satisfaction * reviews, data = airbnb)
7 ##
8 ## Residuals:
9 ##      Min       1Q   Median       3Q      Max
10 ## -82.17  -36.71  -16.08   13.47 1561.53
11 ##
12 ## Coefficients:
13 ##              Estimate Std. Error t value Pr(>|t|)
14 ## (Intercept)      48.77336    10.14434   4.808 1.55e-06 ***
15 ## overall_satisfaction      8.91437     2.17246   4.103 4.10e-05 ***
16 ## reviews          -0.99200     0.26160  -3.792 0.00015 ***
17 ## overall_satisfaction:reviews  0.18961     0.05573   3.402 0.00067 ***
18 ## ---
19 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
20 ##
21 ## Residual standard error: 71.31 on 10583 degrees of freedom
22 ## (7064 observations deleted due to missingness)
23 ## Multiple R-squared:  0.008861, Adjusted R-squared:  0.00858
24 ## F-statistic: 31.54 on 3 and 10583 DF, p-value: < 2.2e-16

```

Com esse modelo, preço estimado = $\beta_0 + \beta_1 \text{overall_satisfaction} + \beta_2 \text{reviews} + \beta_3 \text{overall_satisfaction} \times \text{reviews}$, em que:

- β_0 é o intercepto (48.77)
- β_1 representa a relação entre `overall_satisfaction` e `price` (8.91) controlando todas as outras variáveis em nosso modelo
- β_2 representa a relação entre revisões e preço (-0.99), controlando todas as outras variáveis em nosso modelo
- β_3 é a interação entre satisfação geral e revisões (0.19)

Para um determinado nível de `reviews`, o relacionamento entre `overall_satisfaction` e `price` pode ser reescrito como:

$$= [\beta_0 + \beta_2 \text{reviews}] + (\beta_1 + \beta_3 \text{reviews}) \times \text{overall_satisfaction}$$

Vemos que tanto a interceptação ($\beta_0 + \beta_2 \text{reviews}$) e a inclinação ($\beta_1 + \beta_3 \text{reviews}$) a relação entre `overall_satisfaction` e `price` depende de `reviews`. No modelo sem interações, apenas a interceptação da relação entre `overall_satisfaction` e `price` dependia de `reviews`. Como adicionamos ao nosso modelo uma interação entre `overall_satisfaction` e `reviews`, a inclinação agora também depende de `reviews`.

Da mesma forma, para um determinado nível de `overall_satisfaction`, o relacionamento entre `reviews` e `price` pode ser reescrito como:

$$= [\beta_0 + \beta_1 \text{overall_satisfaction}] + (\beta_2 + \beta_3 \text{overall_satisfaction}) \times \text{reviews}$$

Aqui também vemos que tanto a interceptação quanto a inclinação da relação entre revisões (`reviews`) e preço (`price`) dependem da satisfação geral (`overall_satisfaction`).

Como dito, quando o relacionamento entre uma variável independente e uma variável dependente depende do nível de outra variável independente, temos uma interação entre as duas variáveis independentes. Para esses dados, a interação é altamente significativa ($p < 0,001$). Vamos visualizar essa interação. Nós nos concentramos na relação entre satisfação geral e preço. Planejaremos isso para um nível de comentários que possa ser considerado baixo, médio e alto:

```

1 airbnb %>%
2   filter(!is.na(overall_satisfaction)) %>%
3   summarize(min = min(reviews),
4             Q1 = quantile(reviews, .25), # primeiro quartil
5             Q2 = quantile(reviews, .50), # segundo quartil ou mediana
6             Q3 = quantile(reviews, .75), # terceiro quartil
7             max = max(reviews),
8             mean = mean(reviews))
9
10 ## # A tibble: 1 x 6
11 ##   min      Q1      Q2      Q3      max      mean
12 ##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
13 ## 1      3      6     13     32     708    28.5

```

Vimos que 25% das listagens têm 6 ou menos comentários, 50% das listagens tem 13 ou menos comentários e 75% das listagens tem 32 ou menos comentários.

Queremos três grupos, no entanto, para que possamos pedir quantis diferentes:

```

1 airbnb %>%
2   filter(!is.na(overall_satisfaction)) %>%
3   summarize(Q1 = quantile(reviews, .33), # baixo
4             Q2 = quantile(reviews, .66), # medio
5             max = max(reviews))         # alto
6
7 ## # A tibble: 1 x 3
8 ##   Q1      Q2      max
9 ##   <dbl> <dbl> <dbl>
10 ## 1      8     23     708

```

e crie grupos com base nesses números:

```

1 airbnb.reviews <- airbnb %>%
2   filter(!is.na(overall_satisfaction)) %>%
3   mutate(review_group = case_when(reviews <= quantile(reviews, .33) ~ "low",
4                                   reviews <= quantile(reviews, .66) ~ "medium",
5                                   TRUE ~ "high"),
6   review_group = factor(review_group, levels = c("low", "medium", "high")))

```

Por isso, pedimos ao R para criar uma nova variável `review_group` que deve ser igual a "low" quando o número de revisões for menor ou igual ao 33º percentil, "medium" quando o número de revisões for menor ou igual ao 66º percentil e "high", caso contrário. Depois, fatoramos a variável `review_group` recém-criada e fornecemos um novo argumento, `levels`, que especifica a ordem dos níveis dos fatores (caso contrário, a ordem seria alfabética: alta, baixa, média). Vamos verificar se o agrupamento foi bem-sucedido:

```

1 # checagem:
2 airbnb.reviews %>%
3   group_by(review_group) %>%
4   summarize(min = min(reviews), max = max(reviews))
5
6 ## # A tibble: 3 x 3
7 ##   review_group  min      max
8 ##   <fct>      <dbl> <dbl>
9 ## 1 low          3      8
10 ## 2 medium      9     23
11 ## 3 high      24    708

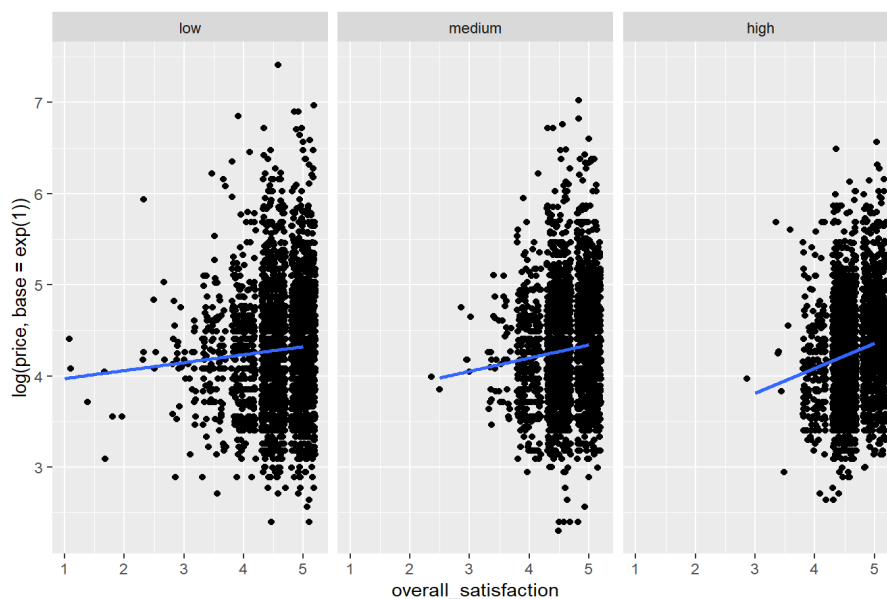
```

De fato, o número máximo de revisões em cada grupo corresponde aos pontos de corte definidos acima. Agora, podemos solicitar a R um gráfico da relação entre `overall_satisfaction` e `price` para os três níveis de revisão:

```

1 ggplot(data = airbnb.reviews, mapping = aes(x = overall_satisfaction, y = log(price, base =
2   exp(1)))) + # transformacao log de preco
3   facet_wrap(~ review_group) + # peca paineis diferentes para cada grupo de revisao
4   geom_jitter() +
5   stat_smooth(method = "lm", se = FALSE)

```



Vemos que a relação entre `overall_satisfaction` e `price` é sempre positiva, mas é mais positiva para listagens com muitas críticas do que para listagens com poucas críticas. Pode haver muitas razões para isso. Talvez seja o caso de listagens com críticas positivas aumentarem o preço, mas somente depois de receberem uma certa quantidade de críticas.

Também vemos que as listagens com muitas avaliações quase nunca têm uma classificação de satisfação menor que 3. Isso faz sentido, porque é difícil continuar atraindo pessoas quando a classificação de uma listagem é baixa. Listas com poucas críticas tendem a ter baixos índices de satisfação geral.

Portanto, parece que nossos preditores estão correlacionados: quanto mais avaliações uma listagem tiver, maior será seu índice de satisfação. Isso potencialmente apresenta um problema que discutiremos em uma das próximas seções sobre [multicolinearidade](#).

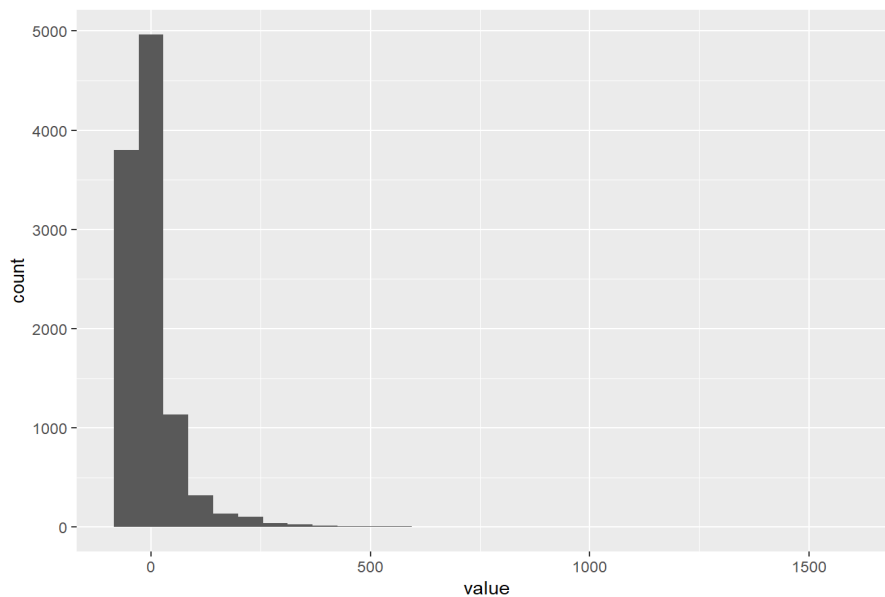
4.2.2 Premissas

Antes de tirar conclusões de uma análise de regressão, é preciso verificar várias suposições. Essas premissas devem ser atendidas independentemente do número de preditores no modelo, mas continuaremos com o caso de dois preditores.

Normalidade dos resíduos

Os resíduos (a diferença entre os valores observados e os estimados) devem ser normalmente distribuídos. Podemos inspecionar visualmente os resíduos:

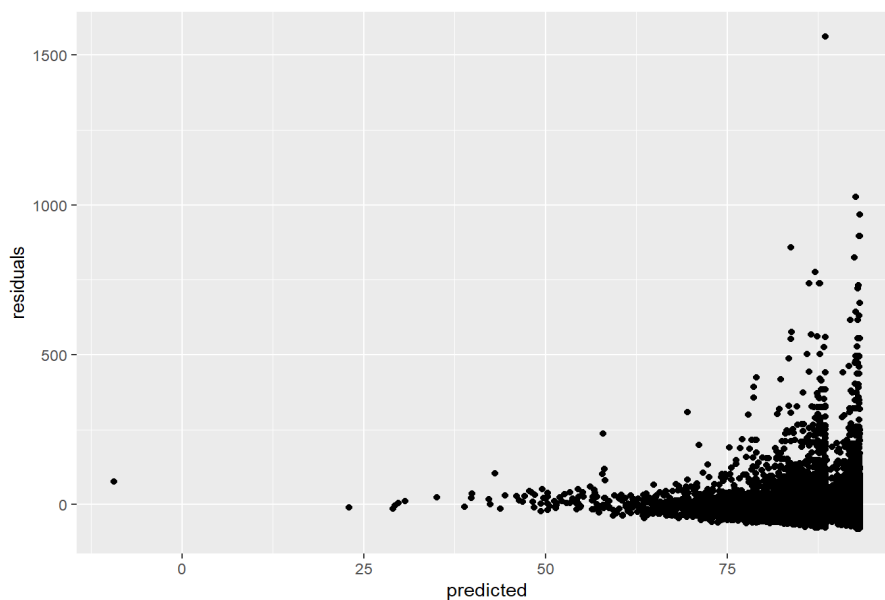
```
1 linearmodel <- lm(price ~ overall_satisfaction * reviews, data = airbnb)
2 residuals <- as_tibble(resid(linearmodel))
3
4 Atenção: Chamar `as_tibble()` em um vetor desencorajado, porque provavelmente que o
  comportamento mude no futuro. Use `enframe(name = NULL)` em seu lugar.
5 ## Este aviso exibido uma vez por sessão.
6
7 # veja os resíduos do modelo linear com resid(linearmodel)
8 # e mude isso em seu dataframe com as_tibble()
9
10 ggplot(data = residuals, mapping = aes(x = value)) +
11   geom_histogram()
12
13 ## `stat_bin()` usando `bins = 30`. Obtenha um valor melhor com `binwidth`.
```



Homocedasticidade dos resíduos

Os resíduos (a diferença entre os valores observados e os estimados) devem ter uma variação constante. Podemos verificar isso plotando os resíduos versus os valores previstos:

```
1 linearmodel <- lm(price ~ overall_satisfaction * reviews, data = airbnb)
2
3 # cria um dataframe (a tibble)
4 residuals_predicted <- tibble(residuals = resid(linearmodel), # a primeira variável são os
5                               predicted = predict(linearmodel)) # a segunda variável é a
6                               prevista, quais são os valores previstos do nosso modelo linear
7 ggplot(data = residuals_predicted, mapping = aes(x = predicted, y = residuals)) +
8   geom_point()
```



Essa suposição é violada porque, quanto maiores nossos valores previstos, maior a variação que vemos nos resíduos.

Multicolinearidade

A multicolinearidade existe sempre que dois ou mais dos preditores em um modelo de regressão são moderadamente ou altamente correlacionados (portanto, é claro que isso não é um problema no caso de regressão simples). Vamos testar a correlação entre `overall_satisfaction` e `reviews`:

```
1 # Certifique-se de incluir o argumento use, caso contrario, o resultado sera NA devido aos
2 # 0 argumento use instrui o R para calcular a correlacao com base apenas nas observacoes
3 # para as quais temos dados sobre price e overall_satisfaction.
4 cor.test(airbnb$overall_satisfaction,airbnb$reviews, use = "pairwise.complete.obs") # teste
5 # para correlacao
6 ##
7 ## Pearson's product-moment correlation
8 ##
9 ## data:  airbnb$overall_satisfaction and airbnb$reviews
10 ## t = 3.3242, df = 10585, p-value = 0.0008898
11 ## alternative hypothesis: true correlation is not equal to 0
12 ## 95 percent confidence interval:
13 ## 0.01325261 0.05131076
14 ## sample estimates:
15 ## cor
16 ## 0.03229339
```

Nossos preditores são de fato significativamente correlacionados ($p < 0,001$), mas a correlação é realmente baixa (0,03). Ao lidar com mais de dois preditores, é uma boa ideia criar uma matriz de correlação.

O problema da multicolinearidade é que ela infla os erros padrão dos coeficientes de regressão. Como resultado, os testes de significância desses coeficientes terão mais dificuldade em rejeitar a hipótese nula. Podemos facilmente ter uma ideia do grau em que os coeficientes são inflados. Para ilustrar isso, vamos estimar um modelo com preditores correlacionados: acomodações (`accommodates`) e preço (`price`) ($r = 0,56$).

```
1 linearmodel <- lm(overall_satisfaction ~ accommodates * price, data = airbnb)
2 summary(linearmodel)
3
4 ##
5 ## Call:
6 ## lm(formula = overall_satisfaction ~ accommodates * price, data = airbnb)
7 ##
8 ## Residuals:
9 ##      Min       1Q   Median       3Q      Max
10 ## -3.6494 -0.1624 -0.1105  0.3363  0.6022
11 ##
12 ## Coefficients:
13 ##              Estimate Std. Error t value Pr(>|t|)
14 ## (Intercept)    4.622e+00  9.938e-03  465.031  < 2e-16 ***
15 ## accommodates   -1.640e-02  2.039e-03  -8.046  9.48e-16 ***
16 ## price          1.356e-03  1.159e-04  11.702  < 2e-16 ***
17 ## accommodates:price -5.423e-05  9.694e-06  -5.595  2.26e-08 ***
18 ## ---
19 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
20 ##
21 ## Residual standard error: 0.3689 on 10583 degrees of freedom
22 ## (7064 observations deleted due to missingness)
23 ## Multiple R-squared:  0.0199, Adjusted R-squared:  0.01963
24 ## F-statistic: 71.64 on 3 and 10583 DF, p-value: < 2.2e-16
```

Vemos que todos os preditores são significativos. Vamos dar uma olhada nos fatores de inflação da variância:

```
1 library(car) # a funcao vif eh do pacote cars
2
3 vif(linearmodel)
4
5 ##      accommodates      price accommodates:price
6 ##      2.090206      5.359203      6.678312
```

Os fatores VIF informam até que ponto os erros padrão são inflados. Uma regra prática é que VIFs de 5 e acima indicam inflação significativa.

4.3 Teste qui-quadrado

Suponha que tenhamos interesse em encontrar uma verdadeira jóia (`gem`) de uma lista. Por exemplo, estamos interessados em listagens com uma classificação de 5 em 5 e pelo menos 30 avaliações:

```
1 airbnb <- airbnb %>%
2   mutate(gem = (overall_satisfaction == 5 & reviews >= 30), # duas condicoes devem ser
3     atendidas antes de dizer que uma listagem eh uma joia
4     gem = factor(gem, labels = c("no gem", "gem"))) # d      variavel logica
5     r tulos mais intuitivos
```

Agora, digamos que estamos interessados em saber se é mais provável encontrar "jóias" (`gem`) em cidades pequenas ou grandes (criamos a variável de tamanho aqui). O teste do qui-quadrado pode fornecer uma resposta a essa pergunta testando a hipótese nula de não haver relação entre duas variáveis categóricas (tamanho da cidade: `large` vs. `small` | `gem`: `yes` vs. `no`).

Ele compara a tabela de frequências observada com a tabela de frequências que você esperaria quando não houvesse relação entre as duas variáveis. Quanto mais as tabelas de frequência observada e esperada divergem, maior a estatística qui-quadrado, menor o valor de p e menos provável é que as duas variáveis não sejam relacionadas.

Antes de realizarmos um teste qui-quadrado, lembre-se de que algumas cidades têm um valor em falta para o tamanho porque têm um valor em falta para a `population`. Vamos filtrar isso primeiro:

```
1 airbnb.cities <- airbnb %>%
2   filter(!is.na(size))
3
4 # queremos apenas aquelas observacoes em que tamanho nao eh NA. ! significa 'nao'
5 # veja https://r4ds.had.co.nz/transform.html#filter-rows-with-filter para mais funcoes
6   logicas (desca ate a secao 5.2.2)
```

Agora, imprima as frequências do tamanho da cidade e combinações de gems:

```
1 airbnb.cities %>%
2   group_by(size, gem) %>%
3   summarize(count = n())
4
5 ## # A tibble: 4 x 3
6 ## # Groups:   size [?]
7 ##   size gem    count
8 ##   <fct> <fct> <int>
9 ## 1 small no gem    4095
10 ## 2 small gem      175
11 ## 3 large no gem  10755
12 ## 4 large gem     941
```

Esta informação está correta, mas o formato em que a tabela é apresentada é um pouco incomum. Gostaríamos de ter uma variável como linhas e a outra como colunas:

```
1 table(airbnb.cities$size, airbnb.cities$gem)
2
3 ##
4 ##           no gem    gem
5 ##   small    4095    175
6 ##   large   10755    941
```

Isso é um pouco mais fácil de interpretar. Uma tabela como essa é frequentemente chamada de tabela cruzada. É fácil pedir porcentagens em vez de contagens:

```
1 crosstable <- table(airbnb.cities$size, airbnb.cities$gem) #Precisamos salvar a tabela
2   cruzada primeiro.
3
4 prop.table(crosstable) # Use a funcao prop.table () para solicitar porcentagens.
5
6 ##
7 ##           no gem    gem
8 ##   small 0.25648253 0.01096079
9 ##   large 0.67361894 0.05893774
10 prop.table(crosstable,1) # This gives percentages conditional on rows, i.e., the
11   percentages in the rows sum to 1.
12 ##
13 ##           no gem    gem
14 ##   small 0.95901639 0.04098361
15 ##   large 0.91954514 0.08045486
16 prop.table(crosstable,2) # This gives percentages conditional on columns, i.e., the
17   percentages in the columns sum to 1.
```

```

15 ##
16 ##           no gem      gem
17 ##   small 0.2757576 0.1568100
18 ##   large 0.7242424 0.8431900

```

Com base nessas frequências ou porcentagens, não devemos esperar uma forte relação entre `size` e `gem`. Vamos realizar o teste do qui-quadrado para testar nossa intuição:

```

1 chisq.test(crosstable)
2
3 ##
4 ##   Pearson's Chi-squared test with Yates' continuity correction
5 ##
6 ## data:  crosstable
7 ## X-squared = 74.355, df = 1, p-value < 2.2e-16

```

O valor da estatística qui é 74,35 e o valor p é praticamente 0, por isso rejeitamos a hipótese nula de nenhum relacionamento. Não é o que esperávamos, mas o valor p é baixo porque nossa amostra é bastante grande (15966 observações).

Você pode relatar o seguinte: “Havia uma relação significativa entre o tamanho da cidade e se uma listagem era ou não uma jóia ($\chi^2(1, N = 15966) = 74,35, p < 0,001$), de modo que as cidades grandes (8,05%) tinham uma porcentagem maior de jóias (raridades) do que as cidades pequenas (4,1%).

4.4 Regressão logística (opcional)

Às vezes, queremos prever uma variável dependente binária, ou seja, uma variável que pode assumir apenas dois valores, com base em várias variáveis independentes contínuas ou categóricas. Por exemplo, digamos que estamos interessados em testar se uma listagem é ou não uma jóia depende do preço e do tipo de quarto da listagem.

Não podemos usar ANOVA ou regressão linear aqui porque a variável dependente é uma variável binária e, portanto, normalmente não é distribuída. Outro problema com a ANOVA ou regressão linear é que ela pode prever valores que não são possíveis (por exemplo, nosso valor previsto pode ser 5, mas apenas 0 e 1 fazem sentido para essa variável dependente). Portanto, usaremos regressão logística. A regressão logística primeiro transforma a variável dependente Y com a transformação do logit. A transformação do logit usa o logaritmo natural das chances de que a variável dependente seja igual a 1:

$$probabilidades = \frac{P(Y = 1)}{P(Y = 0)} = \frac{P(Y = 1)}{1 - P(Y = 1)}$$

$$\text{então o logit } P(Y = 1) = \ln \frac{P(Y = 1)}{1 - P(Y = 1)}$$

Isso garante que nossa variável dependente seja normalmente distribuída e não seja restrita a ser 0 ou 1. Vamos realizar a regressão logística:

```

1 logistic.model <- glm(gem ~ price * room_type, data=airbnb, family="binomial") # o
  argumento family = "binomial" diz R para tratar a variavel dependente como uma variavel
  0/1
2 summary(logistic.model) # saida da regressao
3
4 ##
5 ## Call:
6 ## glm(formula = gem ~ price * room_type, family = "binomial", data = airbnb)
7 ##
8 ## Deviance Residuals:
9 ##      Min       1Q   Median       3Q      Max
10 ## -0.4909  -0.3819  -0.3756  -0.3660   2.5307
11 ##
12 ## Coefficients:
13 ##              Estimate Std. Error z value Pr(>|z|)
14 ## (Intercept)    -2.5270702   0.0570207  -44.318  <2e-16 ***
15 ## price          -0.0007185   0.0004030   -1.783   0.0746 .
16 ## room_typePrivate room    -0.0334362   0.1072091   -0.312   0.7551
17 ## room_typeShared room     1.0986318   1.1278159    0.974   0.3300
18 ## price:room_typePrivate room -0.0011336   0.0012941   -0.876   0.3810
19 ## price:room_typeShared room  -0.0562610   0.0381846   -1.473   0.1406
20 ## ---
21 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

22 ##
23 ## (Dispersion parameter for binomial family taken to be 1)
24 ##
25 ##      Null deviance: 8663.8  on 17650  degrees of freedom
26 ## Residual deviance: 8648.6  on 17645  degrees of freedom
27 ## AIC: 8660.6
28 ##
29 ## Number of Fisher Scoring iterations: 7

```

Vemos que o único preditor marginalmente significativo de uma listagem ser ou não uma jóia é o preço da listagem. Você pode relatar o seguinte: “Controlando o tipo de quarto e a interação entre preço e tipo de quarto, havia uma relação negativa marginalmente significativa entre o preço e a probabilidade de uma listagem ser uma jóia ($\beta = -0.0007185$, $\chi(17645) = -1,783$, $p = 0,075$).

A interpretação dos coeficientes de regressão na regressão logística é diferente da do caso da regressão linear:

```

1 summary(logistic.model)
2 ##
3 ## Call:
4 ## glm(formula = gem ~ price * room_type, family = "binomial", data = airbnb)
5 ##
6 ## Deviance Residuals:
7 ##      Min       1Q   Median       3Q      Max
8 ## -0.4909  -0.3819  -0.3756  -0.3660   2.5307
9 ##
10 ## Coefficients:
11 ##              Estimate Std. Error z value Pr(>|z|)
12 ## (Intercept)    -2.5270702   0.0570207  -44.318  <2e-16 ***
13 ## price          -0.0007185   0.0004030   -1.783   0.0746 .
14 ## room_typePrivate room    -0.0334362   0.1072091   -0.312   0.7551
15 ## room_typeShared room     1.0986318   1.1278159    0.974   0.3300
16 ## price:room_typePrivate room -0.0011336   0.0012941   -0.876   0.3810
17 ## price:room_typeShared room  -0.0562610   0.0381846   -1.473   0.1406
18 ## ---
19 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
20 ##
21 ## (Dispersion parameter for binomial family taken to be 1)
22 ##
23 ##      Null deviance: 8663.8  on 17650  degrees of freedom
24 ## Residual deviance: 8648.6  on 17645  degrees of freedom
25 ## AIC: 8660.6
26 ##
27 ## Number of Fisher Scoring iterations: 7

```

O coeficiente de regressão do preço é -0.0007185. Isso significa que um aumento de uma unidade no preço levará a um aumento de -0.0007185 nas chances de log de joia ser igual a 1 (ou seja, de uma listagem sendo uma joia). Por exemplo:

$$\text{logit}(P(Y = 1 \mid \text{price} = 60)) = \text{logit}(P(Y = 1 \mid \text{price} = 59)) - 0.0007185$$

$$\Leftrightarrow \text{logit}(P(Y = 1 \mid \text{price} = 60)) - \text{logit}(P(Y = 1 \mid \text{price} = 59)) = -0.0007185$$

$$\Leftrightarrow \ln(\text{probabilidades}(P(Y = 1 \mid \text{price} = 60))) - \ln(\text{probabilidades}(P(Y = 1 \mid \text{price} = 59))) = -0.0007185$$

$$\Leftrightarrow \ln\left(\frac{\text{probabilidades}(P(Y = 1 \mid \text{price} = 60))}{\text{probabilidades}(P(Y = 1 \mid \text{price} = 59))}\right) = -0.0007185$$

$$\Leftrightarrow \frac{\text{probabilidades}(P(Y = 1 \mid \text{price} = 60))}{\text{probabilidades}(P(Y = 1 \mid \text{price} = 59))} = e^{-0.0007185}$$

$$\Leftrightarrow \text{probabilidades}(P(Y = 1 \mid \text{price} = 60)) = e^{-0.0007185} * \text{probabilidades}(P(Y = 1 \mid \text{price} = 59))$$

Assim, o coeficiente de regressão em uma regressão logística deve ser interpretado como o aumento relativo nas chances da variável dependente ser igual a 1, para cada aumento de unidade no preditor,

controlando todos os outros fatores em nosso modelo. Nesse caso, as probabilidades de uma listagem ser uma gema devem ser multiplicadas por $e^{0,0007185} = 0,999$ ou diminuídas em 0,1%, para cada aumento de preço unitário. Em outras palavras, listagens mais caras têm menos probabilidade de serem jóias. No exemplo específico acima, as chances de ser uma joia de uma listagem com preço de 60 são $e^{(-0,00071855)} = 0,996$ vezes a chance de ser uma jóia de uma listagem com preço de 59.

4.4.1 Medindo o ajuste de uma regressão logística: porcentagem classificada corretamente

Nosso modelo usa o preço e o tipo de quarto da listagem para prever se a listagem é uma joia ou não. Ao fazer previsões, é natural nos perguntarmos se nossas previsões são boas. Em outras palavras, usando preço e tipo de quarto, com que frequência prevemos corretamente se uma listagem é uma jóia ou não? Para ter uma idéia da qualidade de nossas previsões, podemos pegar o preço e o tipo de quarto das listagens em nosso conjunto de dados, prever se as listagens são gemas e comparar nossas previsões com o status real da gema das listagens. Vamos primeiro fazer as previsões:

```
1 airbnb <- airbnb %>%
2   mutate(prediction = predict(logistic.model, airbnb))
3 # Crie uma nova coluna chamada previsao no quadro de dados do airbnb e armazene nela a
4   previsao,
5 # baseado em logistic.model, para os dados do airbnb
6 # De uma olhada nessas previsoes:
7 head(prediction)
8 ## 1 2 3 4 5 6
9 ## -4.790232 -4.448355 -4.049498 -4.619293 -4.106477 -4.847211
10 # Compare com as observacoes:
11 head(airbnb$gem)
12 ## [1] no gem no gem no gem no gem no gem no gem
13 ## Levels: no gem gem
```

Você vê o problema? As previsões são logits, ou seja, logaritmos de chances de que as listagens sejam gemas, mas as observações simplesmente nos dizem se uma listagem é uma gema ou não. Para uma comparação significativa entre previsões e observações, precisamos transformar os logits em uma decisão: gema ou não gema. É fácil transformar logits em probabilidades usando o exponencial do logit. A relação entre probabilidades e probabilidades é a seguinte:

$$\begin{aligned} \text{probabilidades} &= \frac{P(Y=1)}{P(Y=0)} = \frac{P(Y=1)}{1 - P(Y=1)} \\ \Leftrightarrow \frac{\text{probabilidades}}{1 + \frac{P(Y=1)}{1 - P(Y=1)}} &= \frac{\frac{P(Y=1)}{1 - P(Y=1)}}{1 + \frac{P(Y=1)}{1 - P(Y=1)}} \\ \Leftrightarrow \frac{\text{probabilidades}}{1 + \text{probabilidades}} &= \frac{\frac{P(Y=1)}{1 - P(Y=1)}}{\frac{1 - P(Y=1) + P(Y=1)}{1 - P(Y=1)}} \\ \Leftrightarrow \frac{\text{probabilidades}}{1 + \text{probabilidades}} &= \frac{P(Y=1)}{1 - P(Y=1) + P(Y=1)} \\ \Leftrightarrow \frac{\text{probabilidades}}{1 + \text{probabilidades}} &= P(Y=1) \end{aligned}$$

Agora vamos calcular, para cada listagem, a probabilidade de a listagem ser uma jóia (`gem`):

```
1 airbnb <- airbnb %>%
2   mutate(prediction.logit = predict(logistic.model, airbnb),
3          prediction.odds = exp(prediction.logit),
4          prediction.probability = prediction.odds / (1+prediction.odds))
5
6 # Inspeccionando as probabilidades preditas
7 head(airbnb$prediction.probability)
```

```

8
9 ##           1           2           3           4           5           6
10 ## 0.008242034 0.011562542 0.017132489 0.009763496 0.016198950 0.007789092

```

Os primeiros números são probabilidades muito baixas, mas também existem probabilidades mais altas e todas as previsões estão entre 0 e 1, como deveriam. Agora precisamos decidir qual probabilidade é suficiente para prevermos que uma listagem é uma `gem` ou não.

Uma escolha óbvia é uma probabilidade de 0,5: uma probabilidade maior que 0,5 significa que prevemos que é mais provável que uma listagem seja uma `gem` do que não. Vamos converter probabilidades em previsões:

```

1 airbnb <- airbnb %>%
2   mutate(prediction = case_when(prediction.probability <= .50 ~ "no gem",
3                                 prediction.probability > .50 ~ "gem"))
4
5 # Inspecinando as predicoes
6 head(airbnb$prediction)
7 ## [1] "no gem" "no gem" "no gem" "no gem" "no gem" "no gem"

```

Uma etapa final é comparar previsões com observações:

```

1 table(airbnb$prediction, airbnb$gem)
2 ##
3 ##      no gem    gem
4 ##  no gem 16471  1180

```

Normalmente, vemos uma tabela 2x2, mas vemos uma tabela com um valor previsto nas linhas e dois valores observados nas colunas. Isso ocorre porque todas as probabilidades previstas estão abaixo de 0,50 e, portanto, sempre previmos que não há gemas. Vamos reduzir o limite para prever que uma listagem é uma `gem`:

```

1 airbnb <- airbnb %>%
2   mutate(prediction = case_when(prediction.probability <= .07 ~ "no gem",
3                                 prediction.probability > .07 ~ "gem"),
4         prediction = factor(prediction, levels = c("no gem", "gem")) # verifique se nenhuma
5                               joia eh o primeiro nivel do nosso fator
6
7 # Observando a tabela
8 table(airbnb$prediction, airbnb$gem)
9 ##
10 ##      no gem    gem
11 ##  no gem 11240  854
12 ##    gem   5231  326

```

Podemos ver que, com esta regra de decisão (prever `gems` sempre que a probabilidade prevista de `gems` for superior a 7%), obtivemos 11240 + 326 corretas e 5231 + 854 previsões erradas, que é uma taxa de acerto de $(11240 + 326) / (11240 + 326 + 5231 + 854) = 65,5\%$.

5 Análise básica de dados: experimentos

Neste capítulo, analisaremos os dados de um experimento que testou se o senso de poder das pessoas afeta sua disposição de pagar (WTP) por produtos relacionados ao status (ou seja, por consumo conspícuo) e se essa relação é diferente quando a WTP desses produtos é visível para os outros versus não.

Os participantes vieram ao nosso laboratório em grupos de oito ou sete. Estavam sentados em frente a um computador em cubículos semi-fechados. Na introdução, os participantes leram que primeiro teriam que preencher um questionário de personalidade e uma pesquisa sobre como eles lidavam com dinheiro. Depois disso, eles teriam que trabalhar juntos em grupos de dois em alguns quebra-cabeças.

A primeira parte da sessão foi um questionário de personalidade avaliando dominância e aspirações de status (Cassidy & Lynn, 1989; Mead & Maner, 2012). Os participantes leram 18 declarações e indicaram se cada uma delas se aplicava a elas ou não. Após o preenchimento deste questionário, os participantes foram lembrados de que, no final da sessão, teriam que trabalhar juntos com outro participante em alguns quebra-cabeças. Cada diáde consistiria em um gerente e um trabalhador. Os participantes leram que a atribuição a esses papéis foi baseada em seus resultados no questionário de personalidade, mas, na realidade, a atribuição a papéis foi aleatória.

Os participantes na condição de alta potência então leram que eram mais adequados para serem gerentes, enquanto os participantes na condição de baixa potência liam que eram mais adequados para serem trabalhadores (Galinsky, Gruenfeld e Magee, 2003). As instruções deixaram claro que os gerentes teriam mais poder na tarefa de resolver quebra-cabeças do que os trabalhadores (eles poderiam decidir como um bônus em potencial de 20 euros seria dividido entre gerente e trabalhador). Antes de iniciar os quebra-cabeças, no entanto, os participantes foram convidados a participar de um estudo diferente.

Em um estudo ostensivamente diferente, a disposição dos participantes de gastar em produtos conspícuos e discretos foi medida. Na introdução desta parte do experimento, a presença do público foi manipulada. Na condição privada, os participantes foram informados simplesmente de que estávamos interessados em seus padrões de consumo. Eles foram questionados quanto gastariam em dez produtos que diferiam na medida em que poderiam ser usados para sinalizar o status. Os produtos conspícuos ou aprimoradores de status eram: um carro novo, uma casa, viagens, roupas e um relógio de pulso (para homens) ou jóias (para mulheres). Os produtos discretos ou com status neutro eram produtos de higiene pessoal básicos, medicamentos domésticos, despertador de quarto, utensílios de cozinha e limpeza doméstica (Griskevicius, et al., 2007). Os participantes responderam em uma escala de nove pontos, variando de 1: "Eu compraria itens muito baratos" a 9: "Eu compraria itens muito caros".

Na condição pública, os participantes foram informados de que estávamos trabalhando em um site onde as pessoas pudessem se encontrar. Este site nos ajudaria a investigar como as pessoas formam impressões entre si com base nos padrões de consumo. Os participantes leram que primeiro teriam que indicar quanto gastariam em alguns produtos. Suas escolhas seriam resumidas em um perfil. Os outros participantes da sessão teriam que formar impressões sobre eles com base nesse perfil. Depois de ver um exemplo da aparência do perfil, os participantes passaram para a mesma medida de consumo da condição privada.

Em suma, o experimento tem um design 2 (poder: alto vs. baixo) x 2 (público: público vs. privado) x 2 (consumo: conspícuo vs. discreto) com poder e audiência manipulados entre os sujeitos e consumo manipulado entre os sujeitos.

As hipóteses neste experimento foram as seguintes:

- Na condição de privado, esperávamos que os participantes de baixa potência tivessem uma WTP maior do que os participantes de alta potência para produtos visíveis, mas não para produtos discretos. Esse padrão de resultados replicaria os resultados de Rucker e Galinsky (2008).
- Esperávamos que a manipulação pública versus privada reduzisse a WTP para produtos visíveis para participantes de baixa potência, mas não para participantes de alta potência. Não esperávamos um efeito da manipulação pública versus privada na WTP para produtos discretos para participantes de baixa ou alta potência.

Este experimento é descrito com mais detalhes em minha tese de doutorado (Franssens, 2016)

Referências

- Cassidy, T., Lynn, R. (1989). *A multifactorial approach to achievement motivation: The development of a comprehensive measure*. Journal of Occupational Psychology, 62(4), 301-312.
- Franssens, S. (2016). *Essays in consumer behavior (Doctoral dissertation)*. KU Leuven, Leuven, Belgium.
- Galinsky, A. D., Gruenfeld, D. H., Magee, J. C. (2003). *From Power to action*. Journal of Personality and Social Psychology, 85(3), 453-466. <https://doi.org/10.1037/0022-3514.85.3.453>
- Griskevicius, V., Tybur, J. M., Sundie, J. M., Cialdini, R. B., Miller, G. F., Kenrick, D. T. (2007). *Blatant benevolence and conspicuous consumption: When romantic motives elicit strategic costly signals*. Journal of Personality and Social Psychology, 93(1), 85-102. <https://doi.org/10.1037/0022-3514.93.1.85>
- Mead, N. L., Maner, J. K. (2012). *On keeping your enemies close: Powerful leaders seek proximity to in-group power threats*. Journal of Personality and Social Psychology, 102(3), 576-591. <https://doi.org/10.1037/a0025755>
- Rucker, D. D., Galinsky, A. D. (2008). *Desire to acquire: Powerlessness and compensatory consumption*. Journal of Consumer Research, 35(2), 257-267. <https://doi.org/10.1086/588569>

5.1 Dados

5.1.1 Importação

Faça o download dos dados [aqui](#). Como sempre, salve os dados em um diretório (de preferência um backup automático do software de compartilhamento de arquivos) e inicie seu script carregando o `tidyverse` e definindo o diretório de trabalho no diretório em que você acabou de salvar seus dados:

```
1 library(tidyverse)
2 library(readxl) # precisamos deste pacote pois nossos dados estao num arquivo Excel
3 setwd("c:/dropbox/work/teaching/R/") # mudando para o nosso diretorio de trabalho
4
5 powercc <- read_excel("power_conspicuous_consumption.xlsx", "data") # Importe o arquivo
  Excel. Perceba que o nome da aba do Excel eh data
```

Não se esqueça de salvar seu script no diretório de trabalho.

5.2 Manipulação

```
1 powercc # mostra os dados
2
3 ## # A tibble: 147 x 39
4 ##   subject start_date      end_date      duration finished power
5 ##   <dbl> <dtm>          <dtm>          <dbl>    <dbl> <chr>
6 ## 1      1 2012-04-19 09:32:56 2012-04-19 09:49:42 1006.      1 high
7 ## 2      2 2012-04-19 09:31:26 2012-04-19 09:51:13 1187.      1 low
8 ## 3      3 2012-04-19 09:29:50 2012-04-19 09:53:10 1400.      1 low
9 ## 4      4 2012-04-19 09:26:25 2012-04-19 09:53:21 1616.      1 low
10 ## 5      5 2012-04-19 09:20:55 2012-04-19 09:54:21 2006.      1 high
11 ## 6      6 2012-04-19 09:28:02 2012-04-19 09:55:50 1668.      1 high
12 ## 7      7 2012-04-19 09:17:54 2012-04-19 09:58:49 2455.      1 low
13 ## 8      8 2012-04-19 09:22:26 2012-04-19 10:01:40 2354.      1 high
14 ## 9      9 2012-04-19 10:13:12 2012-04-19 10:31:03 1071.      1 low
15 ## 10     10 2012-04-19 10:12:55 2012-04-19 10:31:29 1114.      1 high
16 ## # ... with 137 more rows, and 33 more variables: audience <chr>,
17 ## #   group_size <dbl>, gender <chr>, age <dbl>, dominance1 <dbl>,
18 ## #   dominance2 <dbl>, dominance3 <dbl>, dominance4 <dbl>,
19 ## #   dominance5 <dbl>, dominance6 <dbl>, dominance7 <dbl>, sa1 <dbl>,
20 ## #   sa2 <dbl>, sa3 <dbl>, sa4 <dbl>, sa5 <dbl>, sa6 <dbl>, sa7 <dbl>,
21 ## #   sa8 <dbl>, sa9 <dbl>, sa10 <dbl>, sa11 <dbl>, inconspicuous1 <dbl>,
22 ## #   inconspicuous2 <dbl>, inconspicuous3 <dbl>, inconspicuous4 <dbl>,
23 ## #   inconspicuous5 <dbl>, conspicuous1 <dbl>, conspicuous2 <dbl>,
24 ## #   conspicuous3 <dbl>, conspicuous4 <dbl>, conspicuous5 <dbl>,
25 ## #   agree <dbl>
```

Temos 39 colunas ou variáveis em nossos dados:

- `subject` identifica os participantes
- `start_date` e `end_date` indicam o início e o fim da sessão experimental.
- `duration` indica a duração da sessão experimental
- `finished`: os participantes concluíram todo o experimento?
- `power` (alto vs. baixo) e público (privado vs. público) são as condições experimentais
- `group_size`: em grupos de quantos participantes compareceram ao laboratório?
- `gender` e `age` do participante
- `dominance1`, `dominance2`, etc. são as perguntas que mediram a dominância. Um exemplo é "Eu acho que gostaria de ter autoridade sobre outras pessoas". Os participantes responderam com sim (1) ou não (0).
- `sa1`, `sa2` etc. são as perguntas que medem as aspirações de status. Um exemplo é: "Gostaria de um trabalho importante, onde as pessoas me admirassem". Os participantes responderam com sim (1) ou não (0).

- `inconspicuous1`, `inconspicuous2`, etc. contêm a `WTP` para os produtos `inconspicuous`. Escala de 1: eu compraria itens muito baratos a 9: eu compraria itens muito caros.
- `conspicuous1`, `conspicuous2`, etc. contêm a `WTP` para os produtos conspícuos. Escala de 1: eu compraria itens muito baratos a 9: eu compraria itens muito caros.
- `agree`: uma questão exploratória que mede se as pessoas concordam que elas são mais adequadas ao papel de trabalhador ou gerente. Os participantes responderam em uma escala de 1: muito mais adequado para a função de trabalhador (gerente) a 7: muito mais adequado para a função de gerente (trabalhador). Números mais altos indicam concordância com a atribuição de função no experimento.

5.2.1 Fatorar algumas variáveis

Após a inspeção dos dados, vemos que o tipo de `subject` é duplo, o que significa que o `subject` deve ser fatorado para que seus valores não sejam tratados como números. Também fatoraremos nossas condições experimentais:

```
1 powercc <- powercc %>% # nos criamos o objeto powercc anteriormente
2   mutate(subject = factor(subject),
3         power = factor(power, levels = c("low","high")), # note os níveis dos argumentos
4         audience = factor(audience, levels = c("private","public"))) # note os níveis dos
   argumentos
```

Observe que fornecemos novos `levels` de argumento ao fatorar poder e público. Este argumento especifica a ordem dos `levels` de um fator. No contexto desse experimento, é mais natural falar sobre o efeito da alta versus baixa potência no consumo do que falar sobre o efeito da baixa versus alta potência no consumo. Portanto, dizemos ao R que o baixo nível de energia deve ser considerado como o primeiro nível. Mais adiante, veremos que o resultado das análises pode ser interpretado como efeitos de alta potência (segundo nível) vs. baixa potência (primeiro nível). O mesmo raciocínio se aplica ao fator público, embora não seja necessário fornecer os níveis para esse fator porque o privado vem antes do público em ordem alfabética. Sua escolha do nível para o primeiro ou nível de referência influencia apenas a interpretação, não o resultado real da análise.

Em uma sessão experimental, o alarme de incêndio disparou e tivemos que sair do laboratório. Vamos remover os participantes que não concluíram a experiência:

```
1 powercc <- powercc %>% # nos ja criamos o objeto powercc anteriormente
2   filter(finished == 1) # somente mantenha as observacoes que sao terminadas ou iguais a 1
```

Observe o `dobro ==` ao testar a igualdade. Confira o livro [R4 Data Science para outros operadores lógicos](#) (role para baixo para chegar à Seção 5.2.2).

5.2.2 Calcular a consistência interna e a média de perguntas que medem o mesmo conceito

Gostaríamos de calcular a média das perguntas que medem a dominância para obter um único número indicando se o participante tem uma personalidade dominante ou não dominante. Antes de fazer isso, devemos ter uma idéia da consistência interna das perguntas que medem o domínio. Isso nos dirá se todas essas perguntas medem o mesmo conceito. Uma medida da consistência interna é o alfa de Cronbach. Para calcular, precisamos de um pacote chamado `psych`:

```
1 install.packages("psych")
2 library(psych)
```

Depois que o pacote for carregado, podemos usar a função `alfa` para calcular o alfa de Cronbach para um conjunto de perguntas:

```
1 dominance.questions <- powercc %>%
2   select(starts_with("dominance")) # pegue o dataframe powercc e selecione todas as
   variaveis com o nome que se inicia com dominancia
3
4 alpha(dominance.questions) # calcula o alfa de cronbach para essas variaveis
5
6 ##
7 ## Reliability analysis
8 ## Call: alpha(x = dominance.questions)
9 ##
10 ##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd median_r
11 ##      0.69      0.67      0.67      0.23 2.1 0.038 0.63 0.27      0.21
```

```

12 ##
13 ## lower alpha upper      95% confidence boundaries
14 ## 0.61 0.69 0.76
15 ##
16 ## Reliability if an item is dropped:
17 ##      raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
18 ## dominance1      0.68      0.67      0.66      0.25 2.0      0.039 0.016 0.22
19 ## dominance2      0.65      0.63      0.62      0.22 1.7      0.043 0.018 0.21
20 ## dominance3      0.59      0.58      0.56      0.19 1.4      0.050 0.014 0.20
21 ## dominance4      0.62      0.60      0.59      0.20 1.5      0.047 0.018 0.21
22 ## dominance5      0.65      0.64      0.63      0.23 1.7      0.043 0.021 0.21
23 ## dominance6      0.70      0.70      0.68      0.28 2.4      0.038 0.009 0.25
24 ## dominance7      0.65      0.64      0.62      0.23 1.8      0.043 0.017 0.20
25 ##
26 ## Item statistics
27 ##      n raw.r std.r r.cor r.drop mean sd
28 ## dominance1 143 0.52 0.49 0.33 0.29 0.53 0.50
29 ## dominance2 143 0.59 0.61 0.52 0.42 0.80 0.40
30 ## dominance3 143 0.75 0.74 0.71 0.59 0.52 0.50
31 ## dominance4 143 0.68 0.68 0.61 0.50 0.58 0.50
32 ## dominance5 143 0.61 0.59 0.48 0.40 0.55 0.50
33 ## dominance6 143 0.29 0.38 0.19 0.14 0.91 0.29
34 ## dominance7 143 0.62 0.59 0.48 0.41 0.53 0.50
35 ##
36 ## Non missing response frequency for each item
37 ##      0 1 miss
38 ## dominance1 0.47 0.53 0
39 ## dominance2 0.20 0.80 0
40 ## dominance3 0.48 0.52 0
41 ## dominance4 0.42 0.58 0
42 ## dominance5 0.45 0.55 0
43 ## dominance6 0.09 0.91 0
44 ## dominance7 0.47 0.53 0
45 ##
46 # Observe que tamb m poder amos ter escrito isso da seguinte maneira:
47 # powercc %>% select(starts_with("dominance")) %>% cronbach()

```

Isso produz muita saída. Em `raw_alpha`, vemos que o alfa é 0,69, que fica no lado inferior (0,70 é geralmente considerado o mínimo necessário), mas ainda está ok. A tabela abaixo nos diz qual seria o alfa se retirássemos uma pergunta de nossa medida. A queda da `dominance6` aumentaria o alfa para 0,7. Comparado ao alfa original de 0,69, esse aumento é pequeno e, portanto, não perdemos a `dominance6`. Se houvesse uma pergunta com um alto "alfa se descartado", isso indicaria que esta pergunta está medindo algo diferente das outras perguntas. Nesse caso, você pode considerar remover esta pergunta da sua medida.

Podemos proceder calculando a média das respostas sobre a questão do domínio:

```

1 powercc <- powercc %>%
2   mutate(dominance = (dominance1+dominance2+dominance3+dominance4+dominance5+dominance6+
3     dominance7)/7,
4     cc = (conspicuous1+conspicuous2+conspicuous3+conspicuous4+conspicuous5)/5,
5     icc = (inconspicuous1+inconspicuous2+inconspicuous3+inconspicuous4+inconspicuous5)
6     /5) %>%
7   select(-starts_with("sa"))

```

Também calculei a média das perguntas sobre consumo conspícuo e consumo discreto, mas não sobre as aspirações de status porque o alfa de Cronbach era muito baixo. Excluí as perguntas sobre aspirações de status do conjunto de dados. Deixo como um exercício verificar os alfa de Cronbach de cada um desses conceitos (faça isso antes de excluir as perguntas sobre as aspirações de status, é claro).

5.2.3 Recapitulando: importando e manipulando

Aqui está o que fizemos até agora, em uma sequência ordenada de operações canalizadas (faça o [download dos dados aqui](#)):

```

1 library(tidyverse)
2 library(readxl)
3 setwd("c:/dropbox/work/teaching/R/") # mudando para seu proprio diretorio
4
5 powercc <- read_excel("power_conspicuous_consumption.xlsx", "data") %>%
6   filter(finished == 1) %>%
7   mutate(subject = factor(subject),
8     power = factor(power, levels = c("low", "high")),
9     audience = factor(audience, levels = c("private", "public")),

```

```

10 dominance = (dominance1+dominance2+dominance3+dominance4+dominance5+dominance6+
11 dominance7)/7,
12 cc = (conspicuous1+conspicuous2+conspicuous3+conspicuous4+conspicuous5)/5,
13 icc = (inconspicuous1+inconspicuous2+inconspicuous3+inconspicuous4+inconspicuous5)
/5) %>%
select(-starts_with("sa"))

```

5.3 Teste *t*

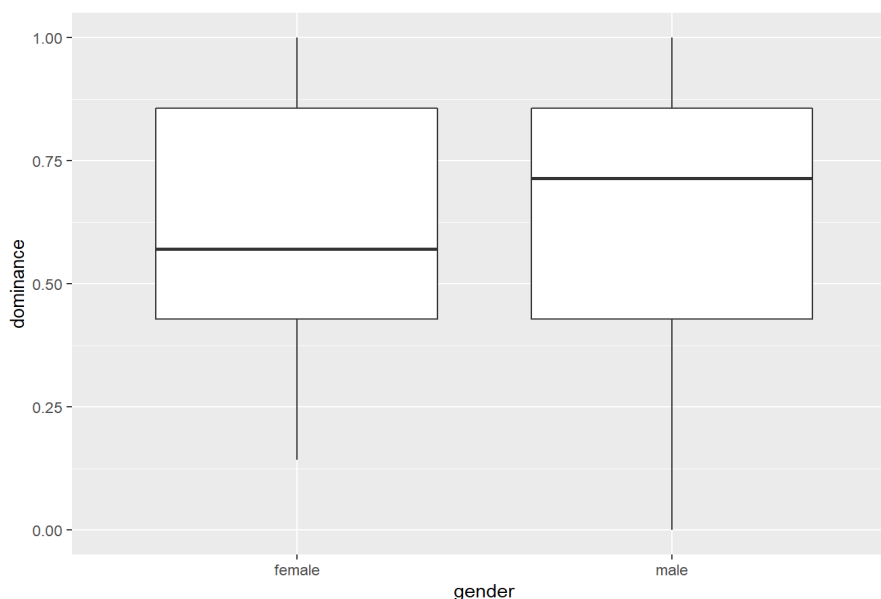
5.3.1 Teste *t* para amostras independentes

Digamos que queremos testar se homens e mulheres diferem no grau em que são dominantes. Vamos criar um boxplot primeiro e depois verificar as médias e os desvios padrão:

```

1 ggplot(data = powercc, mapping = aes(x = gender, y = dominance)) +
2   geom_boxplot()

```



```

1 powercc %>%
2   group_by(gender) %>%
3   summarize(mean_dominance = mean(dominance),
4             sd_dominance = sd(dominance))
5
6 ## # A tibble: 2 x 3
7 ##   gender mean_dominance sd_dominance
8 ##   <chr>         <dbl>         <dbl>
9 ## 1 female         0.614         0.247
10 ## 2 male          0.646         0.296

```

Os homens pontuam um pouco mais alto que as mulheres, mas queremos saber se essa diferença é significativa. Um teste *t* de amostras independentes pode fornecer a resposta (os homens e as mulheres em nosso experimento são amostras independentes), mas precisamos verificar primeiro uma suposição: as variações das duas amostras independentes são iguais?

```

1 install.packages("car") # para o teste de variancias iguais, precisamos de um pacote
   chamado car
2 library(car)
3
4 # Teste Levene para igualdade de variancias.
5 # Baixo valor-p indica que as variancias nao sejam iguais.
6 # Primeiro argumento = variavel dependente continua, segundo argumento = variavel
   independente categorica
7
8 leveneTest(powercc$dominance, powercc$gender)
9
10 ## Levene's Test for Homogeneity of Variance (center = median)
11 ##      Df F value Pr(>F)
12 ## group  1  2.1915  0.141
13 ##

```


A hipótese nula de variâncias iguais não é rejeitada ($p = 0,14$), para que possamos continuar com um teste t que assume variâncias iguais:

```
1 # Teste se os meios de dominancia diferem entre os sexos.
2 # Indique se o teste deve assumir variacoes iguais ou nao (define var.equal = FALSE para um
  teste que nao assume variacoes iguais).
3
4 t.test(powercc$dominance ~ powercc$gender, var.equal = TRUE)
5
6 ##
7 ## Two Sample t-test
8 ##
9 ## data: powercc$dominance by powercc$gender
10 ## t = -0.6899, df = 141, p-value = 0.4914
11 ## alternative hypothesis: true difference in means is not equal to 0
12 ## 95 percent confidence interval:
13 ## -0.12179092 0.05877722
14 ## sample estimates:
15 ## mean in group female mean in group male
16 ## 0.6142857 0.6457926
```

Você pode relatar o seguinte: “Homens ($M = 0,65$, $DP = 0,3$) e mulheres ($M = 0,61$, $DP = 0,25$) não diferiram no grau em que se classificaram como dominantes ($t(141) = -0,69$, $p = 0,49$). ”

5.3.2 Teste t para amostras dependentes

Digamos que queremos testar se as pessoas estão mais dispostas a gastar em itens conspícuos do que em itens discretos. Vamos verificar os meios e os desvios padrão primeiro:

```
1 powercc %>% # nao ha necessidade de agrupar! nao estamos dividindo nossa amostra em
  subgrupos
2 summarize(mean_cc = mean(cc), sd_cc = sd(cc),
3            mean_icc = mean(icc), sd_icc = sd(icc))
4
5 ## # A tibble: 1 x 4
6 ##   mean_cc sd_cc mean_icc sd_icc
7 ##   <dbl> <dbl> <dbl> <dbl>
8 ## 1    6.01  1.05    3.60  0.988
```

As médias são mais altas para produtos conspícuos do que para produtos discretos, mas queremos saber se essa diferença é significativa e, portanto, realizar um teste t de amostras dependentes (cada participante classifica produtos conspícuos e discretos, portanto, essas classificações são dependentes):

```
1 t.test(powercc$cc, powercc$icc, paired = TRUE) # Teste se as medias de cc e icc sao
  diferentes. Indique que este eh um teste t de amostras dependentes com emparelhado =
  TRUE.
2
3 ##
4 ## Paired t-test
5 ##
6 ## data: powercc$cc and powercc$icc
7 ## t = 25.064, df = 142, p-value < 2.2e-16
8 ## alternative hypothesis: true difference in means is not equal to 0
9 ## 95 percent confidence interval:
10 ## 2.214575 2.593816
11 ## sample estimates:
12 ## mean of the differences
13 ## 2.404196
```

Você pode relatar o seguinte: “As pessoas indicaram que estavam dispostas a pagar mais ($t(142) = 25.064$, $p < 0,001$) por produtos conspícuos ($M = 6,01$, $DP = 1,05$) do que por produtos discretos ($M = 3,6$, $DP = 0,99$). ”

5.3.3 Teste t para amostra única

Digamos que queremos testar se a disposição média de pagar pelos itens visíveis foi significativamente maior que 5 (o ponto médio da escala):

```
1 t.test(powercc$cc, mu = 5) # Indique a variavel cuja media queremos comparar com um valor
  especifico (5).
2
3 ##
4 ## One Sample t-test
5 ##
6 ## data: powercc$cc
7 ## t = 11.499, df = 142, p-value < 2.2e-16
8 ## alternative hypothesis: true mean is not equal to 5
9 ## 95 percent confidence interval:
10 ## 5.833886 6.180100
11 ## sample estimates:
12 ## mean of x
13 ## 6.006993
```

Na verdade, é significativamente maior que 5. Você pode relatar o seguinte: "A WTP média para produtos conspícuos (M = 6,01, DP = 1,05) estava significativamente acima de 5 ($t(142) = 11,499$, $p < 0,001$)."

5.4 ANOVA bivariada

Quando você lê um artigo acadêmico que relata um experimento, a seção de resultados geralmente começa com uma discussão dos principais efeitos dos fatores experimentais e sua interação, conforme testado por uma Análise de variância ou ANOVA. Vamos nos concentrar primeiro na **WTP** para produtos que melhoram o status. (Consideramos brevemente a **WTP** para produtos com status neutro em nossa discussão sobre testes t de amostras dependentes e discutiremos mais detalhadamente na seção ANOVA de medidas repetidas.)

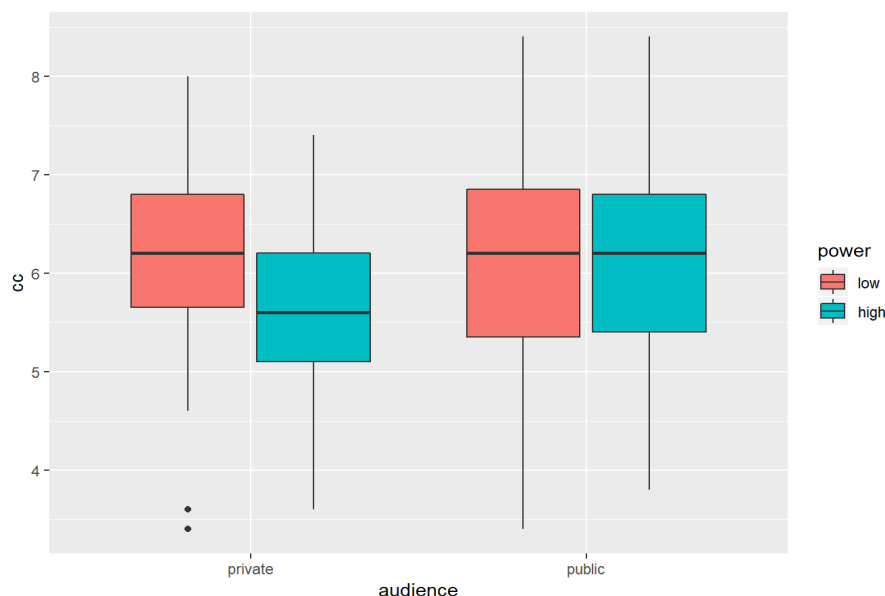
Vamos inspecionar algumas estatísticas descritivas primeiro. Gostaríamos de ver a **WTP** média para produtos visíveis, o desvio padrão dessa **WTP** e o número de participantes em cada célula experimental. Já aprendemos como fazer isso no capítulo introdutório (consulte as tabelas de frequência e as estatísticas descritivas):

```
1 powercc.summary <- powercc %>% # estamos atribuindo o resumo a um objeto, porque
  precisaremos desse resumo para fazer um grafico de barras
  group_by(audience, power) %>% # agora agrupamos por duas variaveis
  summarize(count = n(),
            mean = mean(cc),
            sd = sd(cc))
2
3
4
5
6
7 powercc.summary
8 ## # A tibble: 4 x 5
9 ## # Groups:   audience [?]
10 ##   audience power count mean sd
11 ##   <fct>   <fct> <int> <dbl> <dbl>
12 ## 1 private low    34  6.08  1.04
13 ## 2 private high   31  5.63  0.893
14 ## 3 public low    40  6.17  1.15
15 ## 4 public high   38  6.07  1.03
```

Vemos que a diferença na **WTP** entre os participantes de alta e baixa potência é maior na condição privada do que na pública.

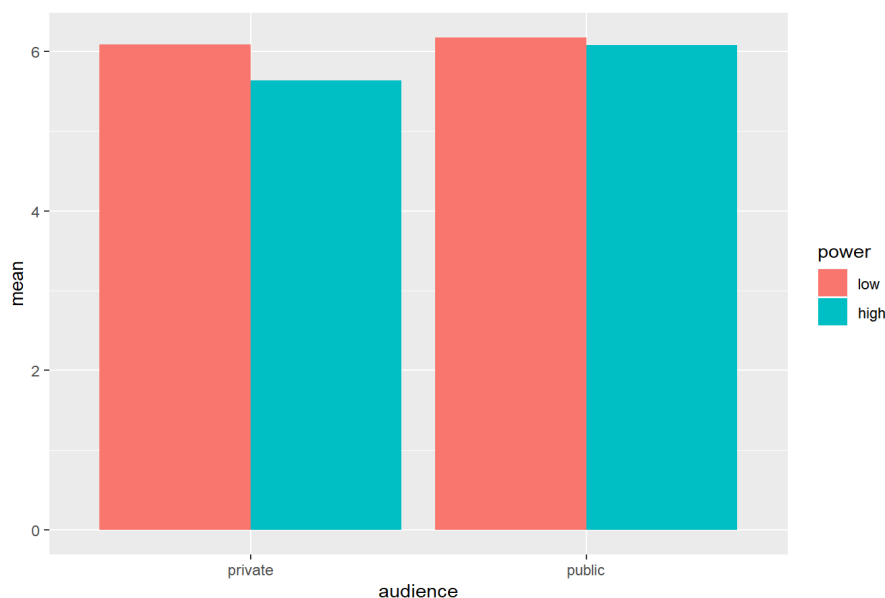
Vamos resumir esses resultados em um gráfico de caixa:

```
1 # ao criar um grafico de caixa, o conjunto de dados que serve como entrada para o ggplot eh
  o conjunto de dados completo, nao o resumo com os meios
2 ggplot(data = powercc, mapping = aes(x = audience, y = cc, fill = power)) +
3   geom_boxplot() # argumentos sao os mesmos que para geom_bar, mas agora fornecemos a geom_
  boxplot
```



Os gráficos boxplot são informativos porque nos dão uma idéia da mediana e da distribuição dos dados. No entanto, em trabalhos acadêmicos, os resultados das experiências são tradicionalmente resumidos em gráficos de barras (mesmo que os gráficos boxplots sejam mais informativos).

```
1 # ao criar um grafico de barras, o conjunto de dados que serve como entrada para o ggplot
  eh o resumo com os meios, nao o conjunto de dados completo
2 ggplot(data = powercc.summary, mapping = aes(x = audience, y = mean, fill = power)) +
3   geom_bar(stat = "identity", position = "dodge")
```



Para criar um gráfico de barras, informamos ao `ggplot` as variáveis que devem estar no eixo X e no eixo Y e também usamos cores diferentes para diferentes níveis de potência com o comando `fill`. Então pedimos um gráfico de barras com `geom_bar`. `stat = "identity"` e `position = "dodge"` devem ser incluídos como argumentos para `geom_bar`, mas uma discussão sobre esses argumentos está além do escopo deste tutorial.

Antes de prosseguirmos com a ANOVA, devemos verificar se suas premissas são atendidas. Discutimos as suposições da ANOVA anteriormente, mas ignoramos aqui.

Para realizar uma ANOVA, precisamos instalar alguns pacotes:

```
1 install.packages("remotes") # O pacote de controles remotos nos permite instalar pacotes
  armazenados no GitHub, um site para desenvolvedores de pacotes.
```

```

2 install.packages("car") # Também precisaremos do pacote do carro para executar a ANOVA (
  n o      necess rio reinstala-lo se voc  j  tiver feito isso).
3
4 library(remotes)
5 install_github('samuelfrasssens/type3anova') # Instale o pacote type3anova. Esta e as
  etapas anteriores precisam ser executadas apenas uma vez.
6
7 library(type3anova) # Carrega o pacote type3anova

```

Agora podemos prosseguir com a ANOVA real:

```

1 # Cria um modelo linear primeiramente
2 # A funcao lm() toma os dados e os argumentos
3 # A formula tem a seguinte sintaxe: variavel dependente ~ variavel independente
4 linearmodel <- lm(cc ~ power*audience, data=powercc)
5 # power*audience: interacao eh calculada
6 # power+audience: interacao nao eh incluida
7
8 type3anova(linearmodel) # Em seguida, pe a a sa da no formato ANOVA. Isso fornece a soma
  dos quadrados do Tipo III. Observe que isso  diferente da anova (modelo linear), que
  fornece a soma dos quadrados do tipo I.
9
10 ## # A tibble: 5 x 6
11 ##   term                ss    df1    df2      f      pvalue
12 ##   <chr>             <dbl> <dbl> <int>   <dbl>   <dbl>
13 ## 1 (Intercept)      5080.      1    139  4710.   4.21e-109
14 ## 2 power              2.64      1    139   2.45   1.20e- 1
15 ## 3 audience           2.48      1    139   2.30   1.32e- 1
16 ## 4 power:audience    1.11      1    139   1.03   3.13e- 1
17 ## 5 Residuals        150.     139    139    NA     NA

```

Você pode relatar esses resultados da seguinte forma: “Nem o principal efeito do poder ($F(1, 139) = 2,45$, $p = 0,12$), nem o principal efeito do público ($F(1, 139) = 2,3$, $p = 0,13$), nem a interação entre poder e público ($F(1, 139) = 1,03$, $p = 0,31$) foi significativa. ”

5.4.1 Seguindo com contrastes

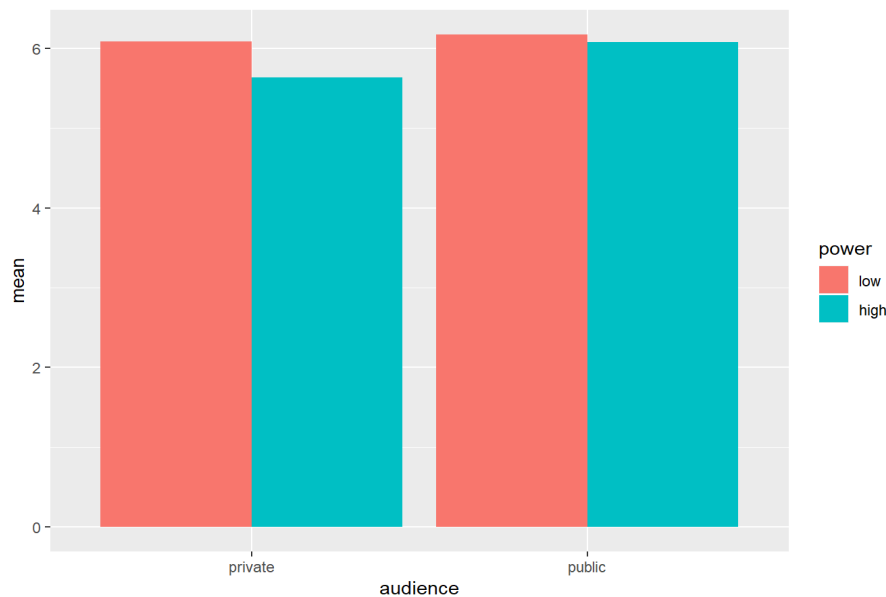
Ao realizar um experimento 2 x 2, geralmente fazemos a hipótese de uma interação. Uma interação significa que o efeito de uma variável independente é diferente dependendo dos níveis da outra variável independente. Na seção anterior, a ANOVA nos disse que a interação entre poder e público não era significativa. Isso significa que o efeito de alta versus baixa potência foi o mesmo na condição privada e pública e que o efeito de público versus privado foi o mesmo na condição de alta e baixa potência. Normalmente, a análise para com a não significância da interação. No entanto, faremos os testes de acompanhamento que um faria se a interação fosse significativa.

Vamos considerar o significado da célula novamente:

```

1 powercc.summary <- powercc %>%
2   group_by(audience, power) %>%
3   summarize(count = n(),
4             mean = mean(cc),
5             sd = sd(cc))
6
7 # ao criar um grafico de barras, o conjunto de dados que serve como entrada para o ggplot
  eh o resumo com os meios, nao o conjunto de dados completo
8 ggplot(data = powercc.summary, mapping = aes(x = audience, y = mean, fill = power)) +
9   geom_bar(stat = "identity", position = "dodge")

```



```
1 powercc.summary
2
3 ## # A tibble: 4 x 5
4 ## # Groups:   audience [2]
5 ##   audience power count  mean    sd
6 ##   <fct>      <fct> <int> <dbl> <dbl>
7 ## 1 private  low      34  6.08  1.04
8 ## 2 private  high     31  5.63  0.893
9 ## 3 public   low      40  6.17  1.15
10 ## 4 public   high     38  6.07  1.03
```

Queremos testar se a diferença entre alta e baixa potência na condição privada é significativa (baixa: 6,08 vs. alta: 5,63) e se a diferença entre alta e baixa potência na condição pública é significativa (baixa: 6,17 vs. alta : 6,07). Para fazer isso, podemos usar a função de contraste da biblioteca `type3anova` que instalamos anteriormente. A função de contraste usa dois argumentos: um modelo linear e uma especificação de contraste. O modelo linear é o mesmo de antes:

```
1 linearmodel <- lm(cc ~ power*audience, data=powercc)
```

Mas, para saber como devemos especificar nosso contraste, precisamos dar uma olhada nos coeficientes de regressão em nosso modelo linear:

```
1 summary(linearmodel)
2
3 ##
4 ## Call:
5 ## lm(formula = cc ~ power * audience, data = powercc)
6 ##
7 ## Residuals:
8 ##      Min       1Q   Median       3Q      Max
9 ## -2.7700 -0.6737  0.1177  0.7177  2.3263
10 ##
11 ## Coefficients:
12 ##              Estimate Std. Error t value Pr(>|t|)
13 ## (Intercept)      6.08235    0.17812   34.148  <2e-16 ***
14 ## powerhigh       -0.45009    0.25792   -1.745  0.0832 .
15 ## audiencepublic  0.08765    0.24226    0.362  0.7181
16 ## powerhigh:audiencepublic 0.35378    0.34910    1.013  0.3126
17 ## ---
18 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
19 ##
20 ## Residual standard error: 1.039 on 139 degrees of freedom
21 ## Multiple R-squared:  0.03711,    Adjusted R-squared:  0.01633
22 ## F-statistic: 1.786 on 3 and 139 DF,  p-value: 0.1527
```

Temos 4 coeficientes de regressão:

- β_0 ou intercepto
- β_1 ou o coeficiente de "powerhigh" ou o alto nível do fator de potência (-0,45)
- β_2 ou o coeficiente para "audiencepublic" ou o nível público do fator público (0,09)
- β_3 ou coeficiente para: "powerhigh:audiencepublic" (0,35)

Vemos que a estimativa para a interceptação corresponde à média observada na célula privada de baixa potência. Adicione a isso a estimativa de "powerhigh" e obteremos a média observada na célula privada de alta potência. Adicione à estimativa para a interceptação, a estimativa de "audiencepublic" e obtenha a média observada na célula pública de baixa potência. Adicione à estimativa para a interceptação, a estimativa de "powerhigh", "audiencepublic" e "powerhigh:audiencepublic" e obteremos a média observada na célula pública de alto poder. Geralmente visualizo isso em uma tabela (e salvo no meu script) para facilitar a especificação do contraste que quero testar:

```
1 #           private           public
2 # low power      b0            b0+b2
3 # high power     b0+b1         b0+b1+b2+b3
```

Digamos que estamos contrastando energia alta versus baixa na condição privada e testamos se $(\beta_0 + \beta_1)$ e β_0 diferem significativamente ou se $(\beta_0 + \beta_1) - \beta_0 = \beta_1$ é significativamente diferente de zero. Nós especificamos isso da seguinte maneira:

```
1 contrast_specification <- c(0, 1, 0, 0)
2
3 # os quatro numeros correspondentes a b0, b1, b2, b3.
4 # queremos testar se b1 eh significativo, entao colocamos 1 em 2 lugar (o 1 lugar
5   para b0)
6
7 contrast(linearmodel, contrast_specification)
8
9 ##
10 ##      Simultaneous Tests for General Linear Hypotheses
11 ##
12 ## Fit: aov(formula = linearmodel)
13 ##
14 ## Linear Hypotheses:
15 ##      Estimate Std. Error t value Pr(>|t|)
16 ## 1 == 0   -0.4501      0.2579  -1.745   0.0832 .
17 ## ---
18 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
19 ## (Adjusted p values reported -- single-step method)
```

A saída nos diz que a estimativa do contraste é -0,45, que é realmente a diferença entre as médias observadas na condição privada (alta potência: 5,63 vs. baixa potência: 6,08). Você pode relatar esse resultado da seguinte forma: “O efeito de alta (M = 5,63, DP = 0,89) vs. baixa potência (M = 6,08, DP = 1,04) na condição privada foi marginalmente significativo ($t(139) = -1,745$, $p = 0,083$).” Para obter os graus de liberdade, faça um contraste `(linearmodel, contrast_specification)$df`

Digamos que queremos testar um contraste mais complicado, ou seja, se a média na célula privada de alta potência é diferente da média das médias nas outras células. Estamos testando:

$$\begin{aligned} & (\beta_0 + \beta_1) - 1/3 * [(\beta_0) + (\beta_0 + \beta_2) + (\beta_0 + \beta_1 + \beta_2 + \beta_3)] \\ & = \beta_1 - 1/3 * (\beta_1 + 2\beta_2 + \beta_3) \\ & = 2/3 * \beta_1 - 2/3 * \beta_2 - 1/3 * \beta_3 \end{aligned}$$

A especificação de contraste é a seguinte:

```
1 contrast_specification <- c(0, 2/3, -2/3, -1/3)
2 contrast(linearmodel, contrast_specification)
3
4 ##
5 ## Simultaneous Tests for General Linear Hypotheses
6 ##
7 ## Fit: aov(formula = linearmodel)
8 ##
9 ## Linear Hypotheses:
10 ## Estimate Std. Error t value Pr(>|t|)
11 ## 1 == 0 -0.4764 0.2109 -2.259 0.0254 *
12 ## ---
13 ## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
14 ## (Adjusted p values reported -- single-step method)
```

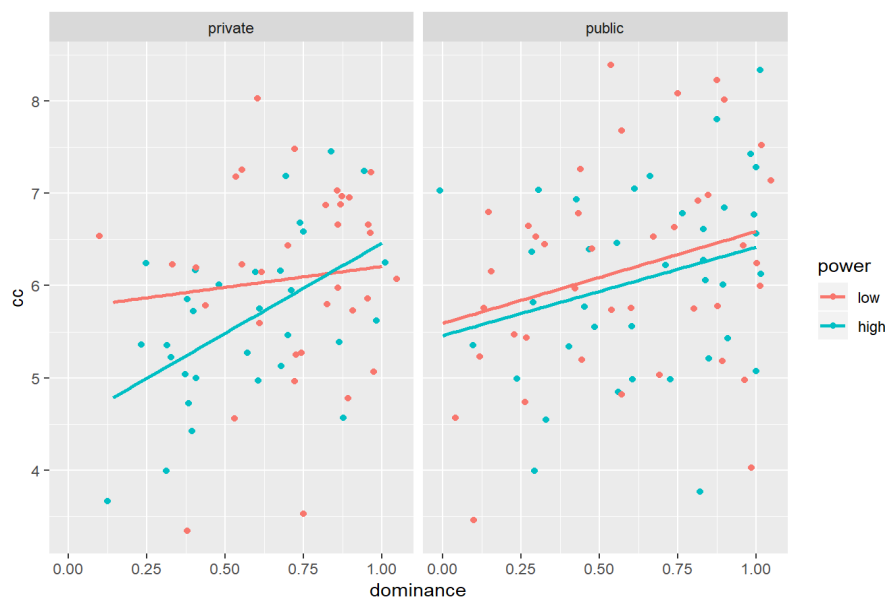
Verifique se a estimativa realmente corresponde à diferença entre a média de alta potência, privada e a média das médias nas outras células. Você pode relatar o seguinte: “A célula privada de alta potência tinha uma WTP significativamente menor do que as outras células experimentais em nosso projeto (est = -0,476, t (139) = -2,259, p = 0,025)”.

5.5 Análise de moderação: interação entre variáveis independentes contínuas e categóricas

Digamos que queremos testar se os resultados do experimento dependem do nível de domínio das pessoas. Em outras palavras, os efeitos do poder e da audiência são diferentes para participantes dominantes versus participantes não dominantes? Ainda em outras palavras, existe uma interação de três vias entre poder, audiência e domínio?

Vamos criar um gráfico primeiro:

```
1 # Nossa variavel dependente eh o consumo conspicuo, nossa variavel independente (no eixo X)
2 # eh a dominancia.
3 # Entao, estamos tracando a relacao entre dominancia e consumo conspicuo.
4 # O argumento da cor diz a R que queremos um relacionamento para cada nivel de poder.
5 ggplot(data = powercc, mapping = aes(x = dominance, y = cc, color = power)) +
6   facet_wrap(~ audience) + # Diga a R que tambem queremos dividir por audiencia.
7   geom_jitter() + # Use geom_jitter em vez de geom_point, caso contrario, os pontos serao
8   geom_smooth(method='lm', se=FALSE) # Desenhe uma linha de regressao linear atraves dos
   pontos.
```



Este gráfico é informativo. Isso nos mostra que, na condição privada, o efeito da alta versus baixa potência no consumo conspicuo é mais negativo para participantes menos dominantes do que para participantes mais dominantes. Na condição pública, o efeito da alta versus baixa potência no consumo conspicuo não difere

entre os participantes menos versus os mais dominantes. Também vemos que, tanto na condição privada quanto na pública, os participantes mais vs. menos dominantes estão mais dispostos a gastar em consumo conspícuo.

Vamos verificar se a interação de três vias é significativa:

```
1 linearmodel <- lm(cc ~ power * audience * dominance, data = powercc)
2 type3anova(linearmodel)
3
4 ## # A tibble: 9 x 6
5 ##   term                ss    df1    df2      f    pvalue
6 ##   <chr>          <dbl> <dbl> <int>   <dbl>   <dbl>
7 ## 1 (Intercept)    525.      1    135  519.  4.35e-48
8 ## 2 power           2.21      1    135   2.18  1.42e- 1
9 ## 3 audience        0.715     1    135   0.706  4.02e- 1
10 ## 4 dominance       10.5      1    135  10.3   1.63e- 3
11 ## 5 power:audience   1.44      1    135   1.42  2.35e- 1
12 ## 6 power:dominance   1.18      1    135   1.17  2.82e- 1
13 ## 7 audience:dominance 0.113     1    135   0.111  7.39e- 1
14 ## 8 power:audience:dominance 1.30      1    135   1.29  2.58e- 1
15 ## 9 Residuals      137.     135   135    NA     NA
```

Somente o efeito da dominância é significativa. Se a interação de três vias fosse significativa, poderíamos acompanhar com mais testes. Por exemplo, poderíamos testar se a interação bidirecional entre dominância e poder é significativa na condição privada, como sugere o gráfico. Para fazer isso, vamos primeiro examinar os coeficientes de regressão do nosso modelo linear:

```
1 summary(linearmodel)
2
3 ##
4 ## Call:
5 ## lm(formula = cc ~ power * audience * dominance, data = powercc)
6 ##
7 ## Residuals:
8 ##      Min       1Q   Median       3Q      Max
9 ## -2.58617 -0.63357  0.09751  0.68310  2.24067
10 ##
11 ## Coefficients:
12 ##              Estimate Std. Error t value Pr(>|t|)
13 ## (Intercept)      5.7553     0.6019   9.561  <2e-16
14 ## powerhigh       -1.2490     0.7703  -1.621   0.107
15 ## audiencepublic  -0.1651     0.6918  -0.239   0.812
16 ## dominance        0.4526     0.7980   0.567   0.572
17 ## powerhigh:audiencepublic  1.1162     0.9355   1.193   0.235
18 ## powerhigh:dominance    1.5021     1.1111   1.352   0.179
19 ## audiencepublic:dominance  0.5434     0.9514   0.571   0.569
20 ## powerhigh:audiencepublic:dominance -1.5394     1.3561  -1.135   0.258
21 ##
22 ## (Intercept)          ***
23 ## powerhigh
24 ## audiencepublic
25 ## dominance
26 ## powerhigh:audiencepublic
27 ## powerhigh:dominance
28 ## audiencepublic:dominance
29 ## powerhigh:audiencepublic:dominance
30 ## ---
31 ## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
32 ##
33 ## Residual standard error: 1.006 on 135 degrees of freedom
34 ## Multiple R-squared:  0.1225, Adjusted R-squared:  0.07703
35 ## F-statistic: 2.693 on 7 and 135 DF, p-value: 0.01211
```

Temos oito termos em nosso modelo:

- β_0 ou intercepto;
- β_1 ou o coeficiente para “powerhigh”
- β_2 ou o coeficiente para “audiencepublic”
- β_3 ou o coeficiente para “dominance”
- β_4 ou o coeficiente para “powerhigh:audiencepublic”

- β_5 ou o coeficiente para “powerhigh:dominance”
- β_6 ou o coeficiente para “audiencepublic:dominance”
- β_7 ou o coeficiente para “powerhigh:audiencepublic:dominance”

Resultando nessa tabela de coeficientes:

	private	public
# low power	[b0] + (b3)	[b0+b2] + (b3+b6)
# high power	[b0+b1] + (b3+b5)	[b0+b1+b2+b4] + (b3+b5+b6+b7)

Como conseguimos essa tabela? Com o modelo linear especificado acima, cada valor estimado de consumo conspícuo (a variável dependente) é uma função da condição experimental do participante e do nível de dominância do participante. Digamos que tenhamos um participante na condição pública de baixa potência com um nível de dominância de 0,5. O valor estimado do consumo conspícuo para esse participante é:

- β_0 ou intercepto x 1;
- β_1 “powerhigh” (porque o participante não está na condição de alta potência)
- β_2 “audiencepublic” $\times 1$ (na condição pública)
- β_3 “dominance” $\times 0.5$ (dominance = 0.5)
- β_4 “powerhigh:audiencepublic” $\times 0$ (não no poder superior e na condição pública)
- β_5 “powerhigh:dominance” $\times 0$ (não na condição de alta potência)
- β_6 “audiencepublic:dominance” $\times 1 \times 0,5$ (na condição pública e dominância = 0,5)
- β_7 “powerhigh:audiencepublic:dominance” $\times 0$ (não no poder superior e na condição pública)

ou

$$\begin{aligned}
 & \beta_0 \times 1 + \beta_1 \times 0 + \beta_2 \times 1 + \beta_3 \times 0.5 + \beta_4 \times 0 + \beta_5 \times 0 + \beta_6 \times 0.5 + \beta_7 \times 0 \\
 & = [\beta_0 + \beta_2] + (\beta_3 + \beta_6) \times 0.5 \\
 & = [5.76 + -0.17] + (0.45 + 0.54) \times 0.5 = 6.09
 \end{aligned}$$

Verifique o gráfico para ver se isso realmente corresponde ao consumo conspícuo estimado de um participante com dominância = 0,5 na célula pública de baixa potência.

A fórmula geral para a célula pública de baixa potência é a seguinte:

$$[\beta_0 + \beta_2] + (\beta_3 + \beta_6) \times \text{dominance}$$

e podemos obter as fórmulas para as diferentes células de maneira semelhante. Vemos que em cada célula, o coeficiente entre colchetes é o valor estimado na medida de consumo conspícuo para um participante que pontua 0 em dominância. O coeficiente entre parênteses arredondados é o aumento estimado na medida de consumo conspícuo para cada aumento de uma unidade no domínio. Em outras palavras, os coeficientes entre colchetes representam a interceptação e os coeficientes entre colchetes representam a inclinação da linha que representa a regressão do consumo conspícuo (Y) na dominância (X) dentro de cada célula experimental (ou seja, cada potência e público-alvo) combinação).

Um teste da interação entre poder e dominância dentro da condição privada se resumiria a testar se as linhas azul e vermelha no painel esquerdo da figura acima devem ser consideradas paralelas ou não. Se eles são paralelos, o efeito da dominância no consumo conspícuo é o mesmo nas condições de baixa e alta potência. Se eles não são paralelos, o efeito da dominância é diferente nas condições de baixa e alta potência. Em outras palavras, devemos testar se os coeficientes de regressão são iguais dentro de baixa potência, privado e dentro de alta potência, privado. Em outras palavras, testamos se $(\beta_3 + \beta_5) - \beta_3 = \beta_5$ é igual a zero:

```

1 # agora temos oito numeros correspondentes aos oito coeficientes de regressao
2 # queremos testar se b5 eh significativo, entao colocamos 1 em 6 lugar (o 1 lugar
   para b0)
3
4 contrast_specification <- c(0, 0, 0, 0, 0, 1, 0, 0)
5
6 contrast(linearmodel, contrast_specification)
7
8 ##
9 ## Simultaneous Tests for General Linear Hypotheses
10 ##
11 ## Fit: aov(formula = linearmodel)
12 ##
13 ## Linear Hypotheses:
14 ## Estimate Std. Error t value Pr(>|t|)
15 ## 1 == 0 1.502 1.111 1.352 0.179
16 ## (Adjusted p values reported -- single-step method)

```

A interação não é significativa, no entanto. Você pode relatar o seguinte: “Dentro da condição privada, não houve interação entre poder e dominância ($t(135) = 1,352$, $p = 0,18$).”

Análise spotlight

Em breve.

5.6 ANCOVA

Descobrimos que nossas condições experimentais não afetam significativamente o consumo conspícuo:

```

1 linearmodel1 <- lm(cc ~ power*audience, data = powercc)
2 type3anova(linearmodel1)
3
4 ## # A tibble: 5 x 6
5 ##   term          ss    df1    df2      f    pvalue
6 ##   <chr>      <dbl> <dbl> <int>   <dbl>   <dbl>
7 ## 1 (Intercept)  5080.      1    139  4710.  4.21e-109
8 ## 2 power         2.64      1    139   2.45  1.20e- 1
9 ## 3 audience       2.48      1    139   2.30  1.32e- 1
10 ## 4 power:audience 1.11      1    139   1.03  3.13e- 1
11 ## 5 Residuals    150.     139    139    NA     NA

```

Por um lado, isso pode significar que simplesmente não há efeitos das condições experimentais no consumo conspícuo. Por outro lado, isso poderia significar que as manipulações experimentais não são fortes o suficiente ou que há muita variação inexplicada em nossa variável dependente (ou ambas). Podemos reduzir a variação inexplicável em nossa variável dependente, no entanto, incluindo uma variável em nosso modelo que suspeitamos estar relacionada à variável dependente. No nosso caso, suspeitamos que a disposição de gastar em consumo discreto (`icc`) esteja relacionada à disposição de gastar em consumo conspícuo. Embora `icc` seja uma variável contínua, podemos incluí-la como uma variável independente em nossa ANOVA e isso nos permitirá reduzir a variação inexplicável em nossa variável dependente:

```

1 linearmodel2 <- lm(cc ~ power*audience + icc, data = powercc)
2 type3anova(linearmodel2)
3
4 ## # A tibble: 6 x 6
5 ##   term          ss    df1    df2      f    pvalue
6 ##   <chr>      <dbl> <dbl> <int>   <dbl>   <dbl>
7 ## 1 (Intercept)  209.      1    138  221.  1.85e-30
8 ## 2 power         1.88      1    138   1.99  1.60e- 1
9 ## 3 audience       1.14      1    138   1.21  2.74e- 1
10 ## 4 icc          19.6      1    138  20.8  1.13e- 5
11 ## 5 power:audience 1.96      1    138   2.08  1.52e- 1
12 ## 6 Residuals    130.     138    138    NA     NA

```

Vemos que `icc` está relacionado à variável dependente e, portanto, que a soma dos quadrados dos resíduos desse modelo, ou seja, a variação inexplicada em nossa variável dependente, é menor (130,32) do que a do modelo sem `icc` (149,93). Os valores p dos fatores experimentais não diminuem, no entanto. Você pode relatar o seguinte: “Controlando a disposição de gastar em consumo discreto, nem o principal efeito do poder ($F(1, 138) = 1,99$, $p = 0,16$), nem o principal efeito do público ($F(1, 138) = 1,21$, $p = 0,27$), nem a interação entre poder e público ($F(1, 138) = 2,08$, $p = 0,15$) foi significativa.”

Chamamos essa análise de ANCOVA porque `icc` é uma covariável (abrange ou está relacionada à nossa variável dependente).

5.7 Medidas repetidas ANOVA

Neste experimento, temos mais de uma medida por unidade de observação, ou seja, disposição para gastar em produtos conspícuos e disposição para gastar em produtos discretos. Uma ANOVA de medidas repetidas pode ser usada para testar se os efeitos das condições experimentais são diferentes para produtos conspícuos versus inconspícuos.

Para executar uma ANOVA de medidas repetidas, precisamos reestruturar nosso quadro de dados de amplo a longo. Um amplo quadro de dados possui uma linha por unidade de observação (em nosso experimento: uma linha por participante) e uma coluna por observação (em nosso experimento: uma coluna para os produtos conspícuos e uma coluna para os produtos inconspícuos). Um quadro de dados longo possui uma linha por observação (em nosso experimento: duas linhas por participante, uma linha para o produto conspícuo e uma linha para o produto inconspícuo) e uma coluna extra que indica com que tipo de observação estamos lidando (conspícua ou imperceptível). Vamos converter o quadro de dados e ver como os quadros de dados amplos e longos diferem um do outro.

```
1 powercc.long <- powercc %>%
2   gather(consumption_type, wtp, cc, icc)
3
4 # Converter de grande para longo significa que estamos empilhando varias colunas umas sobre
5   # as outras. Para isso, precisamos de uma variavel extra para acompanhar qual coluna
6   # estamos lidando.
7 # A funcao de coleta converte conjuntos de dados de amplos para longos.
8 # O primeiro argumento (consumer_type) nos dira com qual coluna estamos lidando. Essa eh a
9   # variavel que armazenara os nomes das colunas que estamos empilhando.
10 # O segundo argumento (wtp) armazenara as colunas reais empilhadas umas sobre as outras.
11 # Os argumentos a seguir sao as colunas que queremos empilhar.
12
13 # Entao, dizemos ao gather para criar duas novas variaveis: tipo_de_consumo e vontade de
14   # pagar, para representar o empilhamento de um determinado numero de colunas.
15
16 # Vamos dizer ao R para nos mostrar apenas as colunas relevantes (isto eh apenas para fins
17   # de apresentacao):
18
19 powercc.long %>%
20   select(subject, power, audience, consumption_type, wtp) %>%
21   arrange(subject)
22
23 ## # A tibble: 286 x 5
24 ##   subject power audience consumption_type wtp
25 ##   <fct>   <fct> <fct>   <chr>          <dbl>
26 ## 1 1      high public    cc              5
27 ## 2 1      high public    icc             2.6
28 ## 3 2      low  public    cc              7.6
29 ## 4 2      low  public    icc              4
30 ## 5 3      low  public    cc              5.4
31 ## 6 3      low  public    icc              3.4
32 ## 7 4      low  public    cc              8.4
33 ## 8 4      low  public    icc              5.2
34 ## 9 5      high public    cc              7.8
35 ## 10 5     high public    icc              3
36 ## # ... with 276 more rows
```

Temos duas linhas por assunto, uma coluna de disposição para pagar e outra coluna (tipo de consumo) que indica se está disposta a pagar por produtos visíveis ou imperceptíveis. Compare isso com o amplo conjunto de dados:

```
1 powercc %>%
2   select(subject, power, audience, cc, icc) %>%
3   arrange(subject) # Mostre somente as colunas relevantes
4
5 ## # A tibble: 143 x 5
6 ##   subject power audience    cc    icc
7 ##   <fct>   <fct> <fct>   <dbl> <dbl>
8 ## 1 1      high public     5     2.6
9 ## 2 2      low  public    7.6     4
10 ## 3 3      low  public    5.4    3.4
11 ## 4 4      low  public    8.4    5.2
12 ## 5 5      high public    7.8     3
13 ## 6 6      high public    7.2     5
14 ## 7 7      low  public    4.8     4
15 ## 8 8      high public    6.6    4.4
16 ## 9 9      low  public    5.8    4.2
```

```

17 ## 10 10      high public      6.8      3.4
18 ## # ... with 133 more rows

```

Temos uma linha por assunto e duas colunas, uma para cada tipo de produto.

Agora podemos realizar uma medida repetida ANOVA. Para isso, precisamos do pacote `ez`.

```

1 install.packages("ez") # Precisamos do pacote ez para RM ANOVA
2 library(ez)

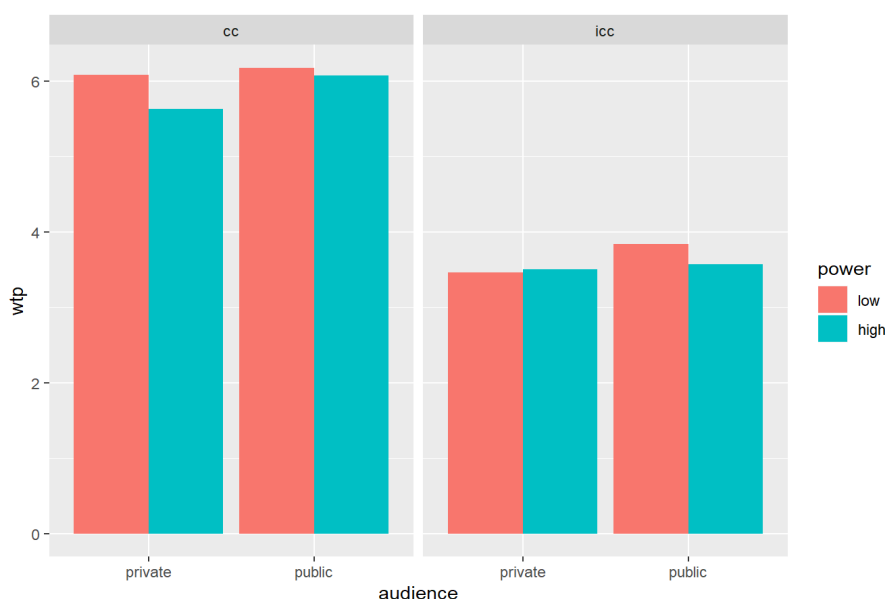
```

Queremos testar se a interação entre poder e público é diferente para produtos conspícuos e inconspícuos. Vamos dar uma olhada em um gráfico primeiro:

```

1 powercc.long.summary <- powercc.long %>% # para um grafico de barras precisamos de summary
  primeiro
2 group_by(power, audience, consumption_type) %>% # agrupa por tres variaveis independentes
3 summarize(wtp = mean(wtp)) # obtem a media de wtp
4
5 ggplot(data = powercc.long.summary, mapping = aes(x = audience, y = wtp, fill = power)) +
6   facet_wrap(~ consumption_type) + # create a panel for each consumption type
7   geom_bar(stat = "identity", position = "dodge")

```



Agora podemos testar formalmente a interação de três vias:

```

1 # Especifique os dados, a variavel dependente, o identificador (wid),
2 # a variavel que representa a condicao dentro dos sujeitos e as variaveis que representam
  as condicoes entre os sujeitos.
3
4 ezANOVA(data = powercc.long, dv = wtp, wid = subject, within = consumption_type, between =
  power*audience)
5
6 ## $ANOVA
7 ##
8 ##           Effect DFn DFd           F          p p<.05
9 ## 2              power      1 139 1.867530e+00 1.739647e-01
10 ## 3             audience      1 139 2.977018e+00 8.667716e-02
11 ## 5      consumption_type      1 139 6.303234e+02 1.687352e-53 *
12 ## 4      power:audience      1 139 5.801046e-03 9.393977e-01
13 ## 6  power:consumption_type      1 139 4.719313e-01 4.932445e-01
14 ## 7  audience:consumption_type      1 139 2.700684e-02 8.697041e-01
15 ## 8 power:audience:consumption_type      1 139 2.982598e+00 8.638552e-02
16 ##
17 ## ges
18 ## 2 9.061619e-03
19 ## 3 1.436772e-02
20 ## 5 5.915501e-01
21 ## 4 2.840440e-05
22 ## 6 1.083172e-03
23 ## 7 6.204919e-05
24 ## 8 6.806406e-03

```

Vemos nesses resultados que a interação de três vias entre poder, público e tipo de consumo é marginalmente significativa. Você pode relatar o seguinte: "Uma ANOVA de medidas repetidas estabeleceu que a

interação de três vias entre poder, público e tipo de consumo era marginalmente significativa ($F(1, 139) = 2,98, p = 0,086$).

6 Análise de componentes principais para mapas perceptivos (office dataset)

Neste capítulo, você aprenderá como realizar uma análise de componentes principais e visualizar os resultados em um mapa perceptivo.

Digamos que tenhamos um conjunto de observações que diferem entre si em várias dimensões; por exemplo, temos várias marcas de uísque (observações) classificadas em vários atributos, como corpo, doçura, sabor, etc. (dimensões). Se algumas dessas dimensões estiverem fortemente correlacionadas, deve ser possível descrever as observações por um número menor (que o original) de dimensões sem perder muita informação. Por exemplo, doçura e frutificação podem ser altamente correlacionadas e, portanto, podem ser substituídas por uma variável. Essa redução de dimensionalidade é o objetivo da análise de componentes principais.

6.1 Dados

6.1.1 Importação

Analisaremos os dados de uma pesquisa na qual os entrevistados foram solicitados a classificar quatro marcas de equipamentos de escritório em seis dimensões. Faça o download dos dados [aqui](#) e importe-os para o R:

```
1 library(tidyverse)
2 library(readxl)
3
4 office <- read_excel("perceptual_map_office.xlsx", "attributes") # não esqueça de carregar o
  pacote readxl
```

6.1.2 Manipulação

```
1 office # visualiza os dados
```

O conjunto de dados consiste em um identificador, a marca do equipamento de escritório (`brand`) e a classificação média (entre os entrevistados) de cada marca em seis atributos: escolha grande (`large-choice`), preços baixos (`low_prices`), qualidade do serviço (`service-quality`), qualidade do produto (`product-quality`), conveniência (`convenience`) e pontuação da preferência (`preference-score`). Vamos fatorar o identificador:

```
1 office <- office %>%
2   mutate(brand = factor(brand))
```

6.1.3 Recapitulação: importação e manipulação

Aqui está o que fizemos até agora, em uma sequência ordenada de operações canalizadas/pipe (faça o download dos dados [aqui](#)):

```
1 library(tidyverse)
2 library(readxl)
3
4 office <- read_excel("perceptual_map_office.xlsx", "attributes") %>% #
5   mutate(brand = factor(brand))
```

6.2 Quantos fatores devemos considerar ?

O objetivo da análise de componentes principais é reduzir o número de dimensões que descrevem nossos dados, sem perder muitas informações. O primeiro passo na análise de componentes principais é decidir o número de componentes ou fatores principais que queremos manter. Para nos ajudar a decidir, usaremos a função PCA do pacote FactoMineR:

```
1 install.packages("FactoMineR")
2 library(FactoMineR)
```

Para poder usar a função PCA, precisamos primeiro transformar o quadro de dados:

```
1 office.df <- office %>%
2   select(- brand) %>% # A entrada para a análise de componentes principais deve ser apenas
3     as.data.frame() # altere o tipo do objeto para 'data.frame'. Isso eh necessario para a
4     funcao PCA
5 rownames(office.df) <- office$brand # Defina os nomes das linhas do data.frame para as
6   marcas (isso eh importante mais tarde ao fazer um biplot)
```

Agora podemos prosseguir com a análise de componentes principais:

```
1 office.pca <- PCA(office.df, graph=FALSE) # Realizar a analise de componentes principais
2
3 office.pca$eig # e veja a tabela com informacoes sobre variancia explicada
4
5 ##          eigenvalue percentage of variance cumulative percentage of variance
6 ## comp 1    4.2656310           71.093850           71.09385
7 ## comp 2    1.6197932           26.996554           98.09040
8 ## comp 3    0.1145758            1.909596          100.00000
```

Se olharmos para esta tabela, veremos que dois componentes explicam 98,1% da variância nas classificações. Isso já é bastante e sugere que podemos fazer com segurança duas dimensões para descrever nossos dados. Uma regra prática aqui é que a variância cumulativa explicada pelos componentes deve ser de pelo menos 70%.

6.3 Análise de Componentes Principais

Vamos reter apenas dois componentes ou fatores:

```
1 office.pca.two <- PCA(office.df, ncp = 2, graph=FALSE) #Peca dois fatores preenchendo o
2   argumento ncp.
```

6.3.1 Cargas fatoriais

Agora podemos inspecionar a tabela com as cargas fatoriais:

```
1 office.pca.two$var$cor %>% #tabela com cargas fatoriais varimax
2   #, mas solicite uma rotacao varimax para melhorar a interpretacao
3
4 ## $loadings
5 ##
6 ## Loadings:
7 ##          Dim.1 Dim.2
8 ## large_choice    0.516 -0.850
9 ## low_prices      -0.990
10 ## service_quality  0.912 -0.410
11 ## product_quality  0.964
12 ## convenience     0.175  0.978
13 ## preference_score 0.708 -0.706
14 ##
15 ##          Dim.1 Dim.2
16 ## SS loadings  3.538  2.347
17 ## Proportion Var 0.590  0.391
18 ## Cumulative Var 0.590  0.981
19 ##
20 ## $rotmat
21 ##          [,1]      [,2]
22 ## [1,] 0.8515627 -0.5242528
23 ## [2,] 0.5242528  0.8515627
```

Essas cargas são as correlações entre as dimensões originais (`large_choice`, `low_prices`, etc.) e dois fatores são extraídos (`Dim.1 and Dim.2`). Nós vemos que `low_prices`, `service_quality`, e `product_quality` pontuação alta no primeiro fator, enquanto `large_choice`, `convenience`, e `preference_score` pontuação alta no segundo fator. Poderíamos, portanto, dizer que o primeiro fator descreve o preço e a qualidade da marca e que o segundo fator descreve a conveniência das lojas da marca.

Também queremos saber quanto cada uma das seis dimensões é explicada pelos fatores extraídos. Para isso, precisamos calcular a comunalidade e / ou seu complemento, a singularidade das dimensões:

```

1 loadings <- as_tibble(office.pca.two$var$cor) %>% # Precisamos capturar os carregamentos
  como um quadro de dados em um novo objeto. Use as_tibble(), caso contrario, nao podemos
  acessar os diferentes fatores
2
3 mutate(variable = rownames(office.pca.two$var$cor), # manter o controle dos nomes das
  linhas (eles sao removidos ao converter para tibble)
4   communality = Dim.1^2 + Dim.2^2,
5   uniqueness = 1 - communality) # O operador ^ eleva um valor a uma determinada
  potencia. Para calcular a comunalidade, precisamos somar os quadrados das cargas em
  cada fator.
6
7 loadings
8 ## # A tibble: 6 x 5
9 ##   Dim.1 Dim.2 variable      communality uniqueness
10 ##   <dbl> <dbl> <chr>          <dbl>      <dbl>
11 ## 1  0.885 -0.453 large_choice    0.988      0.0116
12 ## 2 -0.845 -0.516 low_prices      0.980      0.0198
13 ## 3  0.991  0.128 service_quality  0.999      0.000669
14 ## 4  0.841  0.473 product_quality  0.930      0.0696
15 ## 5 -0.364  0.925 convenience    0.988      0.0124
16 ## 6  0.973 -0.230 preference_score  0.999      0.000524

```

A comunalidade de uma variável é a porcentagem da variação dessa variável que é explicada pelos fatores. Seu complemento é chamado de exclusividade. A exclusividade (unicidade) pode ser puro erro de medição ou pode representar algo que é medido de forma confiável por essa variável específica, mas não por nenhuma das outras variáveis. Quanto maior a exclusividade, maior a probabilidade de que seja mais do que apenas erro de medição. Uma exclusividade superior a 0,6 é geralmente considerada alta. Se a exclusividade for alta, a variável não será bem explicada pelos fatores. Vemos que, para todas as dimensões, a comunalidade é alta e, portanto, a singularidade é baixa; portanto, todas as dimensões são bem capturadas pelos fatores extraídos.

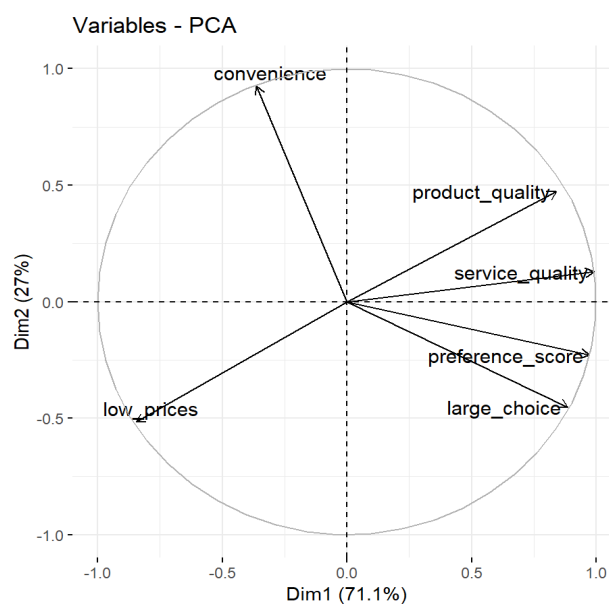
6.3.2 Plotando as cargas fatoriais

Também podemos plotar as cargas. Para isso, precisamos de outro pacote chamado factoextra:

```

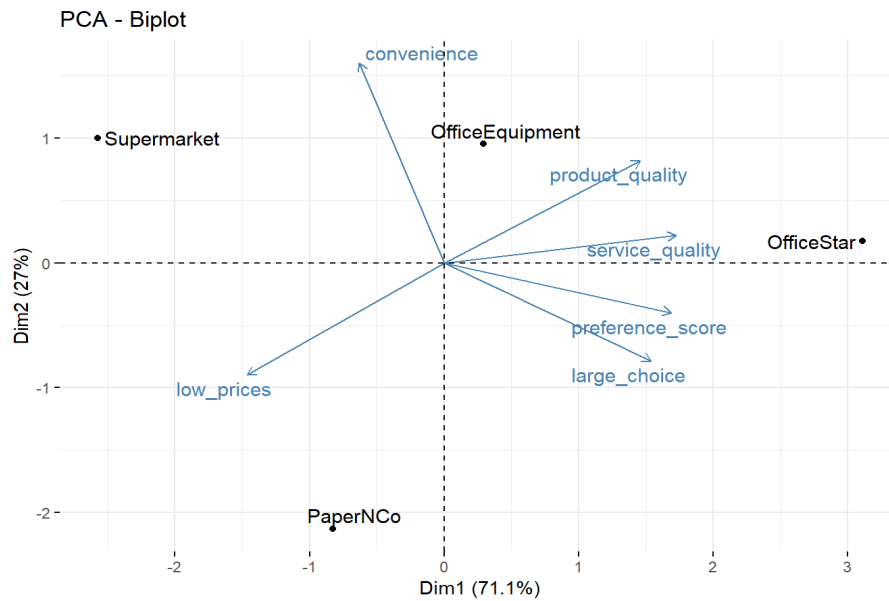
1 install.packages("factoextra")
2 library(factoextra)
3 fviz_pca_var(office.pca.two, repel = TRUE) # o argumento repel = TRUE garante que o texto
  seja exibido corretamente no grafico

```



Nós vimos que large_choice, service_quality, product_quality e preference_score têm pontuações altas no primeiro fator (o Dim1 do eixo X) e essa conveniência tem uma pontuação alta no segundo fator (o Dim2 do eixo Y). Também podemos adicionar as observações (as diferentes marcas) a esse gráfico:


```
1 fviz_pca_biplot(office.pca.two, repel = TRUE) # tracar as cargas e as marcas juntas em uma trama
```



Isso também é chamado de biplot. Podemos ver, por exemplo, que o OfficeStar tem uma pontuação alta no primeiro fator.

7 Análise de componentes principais para mapas perceptivos (toothpaste dataset)

Neste capítulo, você aprenderá como realizar uma análise de componentes principais e visualizar os resultados em um mapa perceptivo.

Digamos que tenhamos um conjunto de observações que diferem entre si em várias dimensões; por exemplo, temos várias marcas de uísque (observações) classificadas em vários atributos, como corpo, doçura, sabor, etc. (dimensões). Se algumas dessas dimensões estiverem fortemente correlacionadas, deve ser possível descrever as observações por um número menor (que o original) de dimensões sem perder muita informação. Por exemplo, doçura e frutificação podem ser altamente correlacionadas e, portanto, podem ser substituídas por uma variável. Essa redução de dimensionalidade é o objetivo da análise de componentes principais.

7.1 Dados

7.1.1 Importação

Analisaremos os dados de uma pesquisa na qual 60 consumidores foram convidados a responder a seis perguntas sobre pasta de dente. Esses dados foram coletados pelos criadores do `Radiant`, que é um pacote do R para análise de negócios que usaremos posteriormente. Faça o download dos dados [aqui](#) e importe-os para o R:

```
1 library(tidyverse)
2 library(readxl)
3
4 toothpaste <- read_excel("toothpaste.xlsx")
```

7.1.2 Manipulação

```
1 toothpaste # Visualiza os dados
```

O conjunto de dados consiste em um identificador, `consumer`, e as classificações do entrevistado sobre a importância de seis atributos de pasta de dente: `prevents_cavities`, `shiny_teeth`, `strengthens_gums`, `freshens_breath`, `decay_prevention_unimportant`, and `attractive_teeth`. Nos também temos os respondentes `age` e `gender`.

Vamos fatorar o identificador e o `gender`:

```
1 toothpaste <- toothpaste %>%
2   mutate(consumer = factor(consumer),
3          gender = factor(gender))
```

7.1.3 Recapitulação: importação e manipulação

Aqui está o que fizemos até agora, em uma sequência ordenada de operações canalizadas (faça o download dos dados [aqui](#)):

```
1 library(tidyverse)
2 library(readxl)
3
4 toothpaste <- read_excel("toothpaste.xlsx")
5   mutate(consumer = factor(consumer),
6          gender = factor(gender))
```

7.2 Quantos fatores devemos considerar ?

O objetivo da análise de componentes principais é reduzir o número de dimensões que descrevem nossos dados, sem perder muitas informações. O primeiro passo na análise de componentes principais é decidir o número de componentes ou fatores principais que queremos manter. Para nos ajudar a decidir, usaremos a função `pre_factor` do pacote `radiant`:

```
1 install.packages("radiant")
2 library(radiant)
3
4 # armazene os nomes das dimensoes em um vetor para que nao tenhamos que digita-las
   repetidamente
```

```

5 dimensions <- c("prevents_cavities", "shiny_teeth", "strengthens_gums", "freshens_breath",
6 "decay_prevention_unimportant", "attractive_teeth")
7
8 # dica: tambem poderiamos fazer o seguinte:
9
10 # dimensions <- toothpaste %>% select(-consumer, -gender, -age) %>% names()
11
12 summary(pre_factor(toothpaste, vars = dimensions))
13
14 ## Pre-factor analysis diagnostics
15 ## Data : toothpaste
16 ## Variables : prevents_cavities, shiny_teeth, strengthens_gums, freshens_breath, decay_
17 ## prevention_unimportant, attractive_teeth
18 ## Observations: 60
19 ##
20 ## Bartlett test
21 ## Null hyp. : variables are not correlated
22 ## Alt. hyp. : variables are correlated
23 ## Chi-square: 238.93 df(15), p.value < .001
24 ##
25 ## KMO test: 0.66
26 ##
27 ## Variable collinearity:
28 ##
29 ## Rsq KMO
30 ## prevents_cavities 0.86 0.62
31 ## shiny_teeth 0.48 0.70
32 ## strengthens_gums 0.81 0.68
33 ## freshens_breath 0.54 0.64
34 ## decay_prevention_unimportant 0.76 0.77
35 ## attractive_teeth 0.59 0.56
36 ##
37 ## Fit measures:
38 ##
39 ## Eigenvalues Variance % Cumulative %
40 ## PC1 2.73 0.46 0.46
41 ## PC2 2.22 0.37 0.82
42 ## PC3 0.44 0.07 0.90
43 ## PC4 0.34 0.06 0.96
44 ## PC5 0.18 0.03 0.99
45 ## PC6 0.09 0.01 1.00

```

Nas `Fit measures`, vemos que dois componentes explicam 82% da variação nas classificações. Isso já é bastante e sugere que podemos reduzir com segurança o número de dimensões para dois componentes. Uma regra prática aqui é que a variação cumulativa explicada pelos componentes deve ser de pelo menos 70%.

7.3 Análise de Componentes Principais

Vamos extrair somente dois componentes ou fatores:

```

1 summary(full_factor(toothpaste, dimensions, nr_fact = 2))
2 # Pe a dois fatores preenchendo o argumento nr_fact.
3
4
5 ## Factor analysis
6 ## Data      : toothpaste
7 ## Variables : prevents_cavities, shiny_teeth, strengthens_gums, freshens_breath, decay_
8 ##            prevention_unimportant, attractive_teeth
9 ## Factors   : 2
10 ## Method    : PCA
11 ## Rotation   : varimax
12 ## Observations: 60
13 ##
14 ## Factor loadings:
15 ##              RC1    RC2
16 ## prevents_cavities      0.96 -0.03
17 ## shiny_teeth            -0.05  0.85
18 ## strengthens_gums       0.93 -0.15
19 ## freshens_breath        -0.09  0.85
20 ## decay_prevention_unimportant -0.93 -0.08
21 ## attractive_teeth       0.09  0.88
22 ##
23 ## Fit measures:
24 ##              RC1    RC2
25 ## Eigenvalues    2.69  2.26
26 ## Variance %     0.45  0.38
27 ## Cumulative %   0.45  0.82
28 ##
29 ## Attribute communalities:
30 ## prevents_cavities      92.59%
31 ## shiny_teeth            72.27%
32 ## strengthens_gums      89.36%
33 ## freshens_breath       73.91%
34 ## decay_prevention_unimportant 87.78%
35 ## attractive_teeth      79.01%
36 ##
37 ## Factor scores (max 10 shown):
38 ##              RC1    RC2
39 ##      1.15 -0.30
40 ##     -1.17 -0.34
41 ##      1.29 -0.86
42 ##      0.29  1.11
43 ##     -1.43 -1.49
44 ##      0.97 -0.31
45 ##      0.39 -0.94
46 ##      1.33 -0.03
47 ##     -1.02 -0.64
48 ##     -1.31  1.56

```

7.3.1 Cargas fatoriais

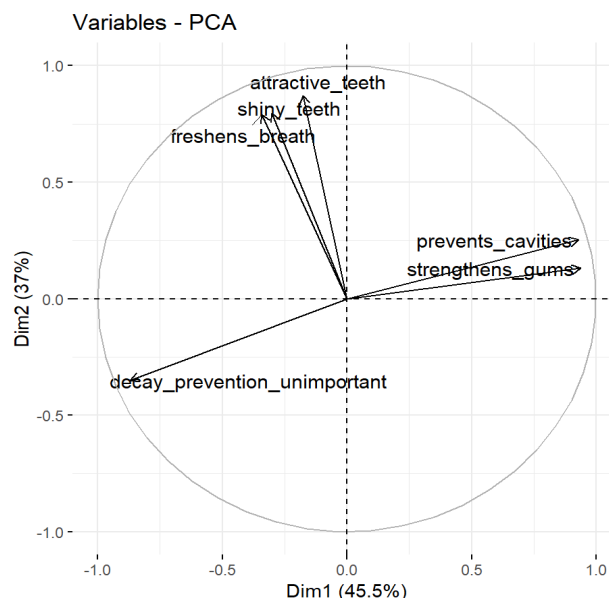
Veja a tabela abaixo do Factor loadings. Essas cargas são as correlações entre as dimensões originais (`prevents_cavities`, `shiny_teeth`, etc.) e os dois fatores que são retidos (RC1 e RC2). Nós vemos que `prevents_cavities`, `strengthens_gums`, e `decay_prevention_unimportant` pontuação alta no primeiro fator, enquanto `shiny_teeth`, `strengthens_gums`, e `freshens_breath` pontuação alta no segundo fator. Poderíamos, portanto, dizer que o primeiro fator descreve preocupações relacionadas à saúde e que o segundo fator descreve preocupações relacionadas à aparência.

Também queremos saber quanto cada uma das seis dimensões é explicada pelos fatores extraídos. Para isso, podemos observar a comunalidade das dimensões (cabeçalho: `Attribute communalities`). A comunalidade de uma variável é a porcentagem da variação dessa variável que é explicada pelos fatores. Seu complemento é chamado de exclusividade ($= 1 - \text{comunalidade}$). A exclusividade pode ser puro erro de medição ou pode representar algo que é medido de forma confiável por essa variável específica, mas não por nenhuma das outras variáveis. Quanto maior a exclusividade, maior a probabilidade de que seja mais do que apenas erro de medição. Uma exclusividade superior a 0,6 é geralmente considerada alta. Se a exclusividade for alta, a variável não será bem explicada pelos fatores. Vemos que, para todas as dimensões, a comunalidade é alta e, portanto, a singularidade é baixa; portanto, todas as dimensões são bem capturadas pelos fatores extraídos.

7.3.2 Gráfico das cargas fatoriais

Também podemos traçar as cargas. Para isso, usaremos dois pacotes:

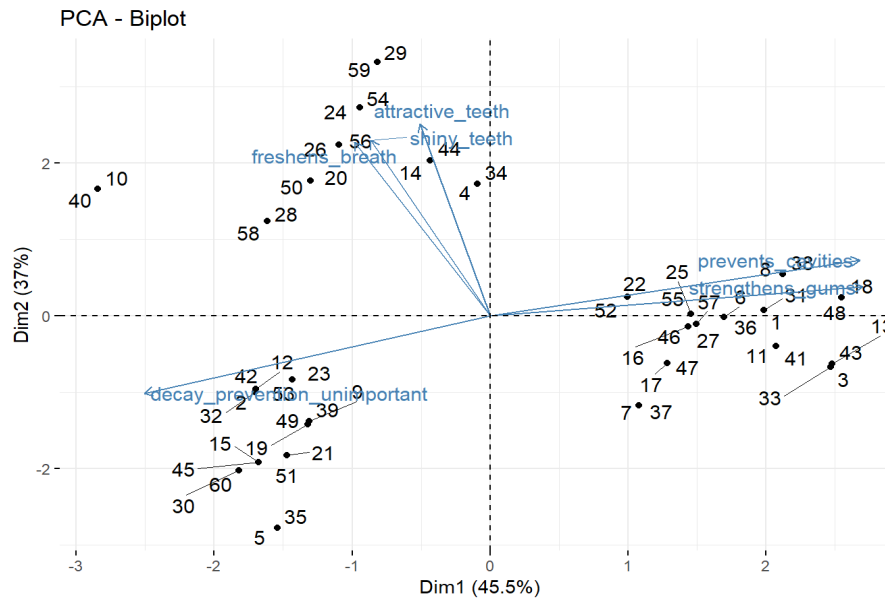
```
1 install.packages("FactoMineR")
2 install.packages("factoextra")
3 library(FactoMineR)
4 library(factoextra)
5 toothpaste %>% # take dataset
6   select(-consumer, -age, -gender) %>% # somente duas dimensoes
7   as.data.frame() %>% # converter em um objeto data.frame, caso contrario, o PCA não o
   aceitar
8   PCA(npc = 2, graph = FALSE) %>% # fa a uma anlise de componentes principais e retenha
   2 fatores
9   fviz_pca_var(repel = TRUE) # pegue essa anlise e a transforme em uma visualiza o
```



Vemos que `attractive_teeth`, `shiny_teeth`, `freshens_breath`, têm pontuações altas no segundo fator (o Dim2 do eixo X). `prevents_cavities` e `strengthens_gums` têm pontuações altas no primeiro fator (o eixo Y Dim2) e `decay_prevention_unimportant` tem uma pontuação baixa nesse fator (essa variável mede a importância da prevenção da cárie).

Também podemos adicionar as observações (os diferentes consumidores) a esse gráfico:

```
1 toothpaste %>% # pega os dados
2 select(-consumer,-age,-gender) %>% # obtem as dimensoes somente
3 as.data.frame() %>% # converte em data.frame object, caso contrario PCA nao aceita
4 PCA(npc = 2, graph = FALSE) %>% # faz o pca e retém 2 fatores
5 fviz_pca_biplot(repel = TRUE) #faz o grafico
```



Isto também é conhecido como biplot.

8 Análise de cluster para segmentação

Neste capítulo, você aprenderá como realizar uma análise de cluster e uma análise discriminante linear.

Uma análise de cluster trabalha em um grupo de observações que diferem entre si em várias dimensões. Ele encontrará clusters de observações no espaço n-dimensional, de modo que a semelhança de observações dentro de clusters seja a mais alta possível e a similaridade de observações entre clusters seja a mais baixa possível. Você sempre pode executar uma análise de cluster e pode solicitar qualquer número de clusters. O número máximo de clusters é o número total de observações. Nesse caso, cada observação será um cluster, mas isso não seria um cluster muito útil. O objetivo do agrupamento em cluster é encontrar um pequeno número de clusters que possam ser descritos de forma significativa por suas pontuações médias nas n dimensões. Em outras palavras, o objetivo é encontrar diferentes 'perfis' de observações.

A análise discriminante linear tenta prever uma variável categórica com base em várias variáveis independentes contínuas ou categóricas. É semelhante à regressão logística. Nós o usaremos para prever a associação de cluster de uma observação (conforme estabelecido pela análise de cluster) com base em algumas variáveis de segmentação (ou seja, outras informações que temos sobre as observações que não serviram de entrada na análise de cluster).

Analisaremos os dados de 40 entrevistados que avaliaram a importância de vários atributos da loja ao comprar equipamentos de escritório. Usaremos a análise de cluster para encontrar agrupamentos de observações, neste caso, agrupamentos de respondentes. Esses clusters terão perfis diferentes (por exemplo, um cluster pode atribuir importância à política de preços e devoluções, o outro pode atribuir importância à variedade de opções e qualidade de serviço). Em seguida, usaremos a análise discriminante linear para testar se podemos prever a associação ao cluster (ou seja, que tipo de comprador de equipamento de escritório alguém é) com base em várias características dos entrevistados (por exemplo, sua renda).

8.1 Dados

Importação

Analisaremos os dados de uma pesquisa na qual 40 entrevistados foram solicitados a avaliar a importância de vários atributos da loja ao comprar equipamentos. Faça o download dos dados aqui e importe-os para o R:

```
1 library(tidyverse)
2 library(readxl)
3
4 equipment <- read_excel("segmentation_office.xlsx", "SegmentationData") # Importa o arquivo
   Excel
```

Manipulação

```
1 equipment
2
3 ## # A tibble: 40 x 10
4 ##   respondent_id variety_of_choi~ electronics furniture quality_of_serv~
5 ##           <dbl>           <dbl>           <dbl>           <dbl>           <dbl>
6 ## 1             1             8             6             6             3
7 ## 2             2             6             3             1             4
8 ## 3             3             6             1             2             4
9 ## 4             4             8             3             3             4
10 ## 5             5             4             6             3             9
11 ## 6             6             8             4             3             5
12 ## 7             7             7             2             2             2
13 ## 8             8             7             5             7             2
14 ## 9             9             7             7             5             1
15 ## 10            10             8             4             0             4
16 ## # ... with 30 more rows, and 5 more variables: low_prices <dbl>,
17 ## #   return_policy <dbl>, professional <dbl>, income <dbl>, age <dbl>
18 ## # Check out the data
```

Temos 10 colunas ou variáveis em nossos dados:

- `respondent_id` é um identificador para nossas observações
- Os entrevistados classificaram a importância de cada um dos seguintes atributos em uma escala de 1 a 10: `variety_of_choice`, `electronics`, `furniture`, `quality_of_service`, `low_prices`, `return_policy`.
- `professional`: 1 para profissionais, 0 para não profissionais

- `income`: expresso em milhares de dólares
- `age`

A análise de cluster tentará identificar clusters com padrões semelhantes de classificações. A análise discriminante linear preverá a associação do cluster com base nas variáveis de segmentação (`professional`, `income`, e `age`).

Como sempre, vamos fatorar as variáveis que devem ser tratadas como categóricas:

```
1 equipment <- equipment %>%
2   mutate(respondent_id = factor(respondent_id),
3     professional = factor(professional, labels = c("non-professional", "professional")))
4 )
```

8.1.1 Recapitulação: importação e manipulação

Aqui está o que fizemos até agora, em uma sequência ordenada de operações canalizadas/pipe (faça o download dos dados [aqui](#)):

```
1 library(tidyverse)
2 library(readxl)
3
4 equipment <- read_excel("segmentation_office.xlsx", "SegmentationData") %>%
5   mutate(respondent_id = factor(respondent_id),
6     professional = factor(professional, labels = c("non-professional", "professional")))
7 )
```

8.2 Análise de Cluster

Primeiro, realizaremos uma análise hierárquica de cluster para encontrar o número ideal de clusters. Depois disso, realizaremos uma análise de cluster não hierárquica e solicitaremos o número de clusters considerados ideais pela análise de cluster hierárquica. As variáveis que servirão de entrada para a análise de cluster são as classificações de importância dos atributos da loja.

8.2.1 Padronizar ou não ?

A primeira etapa de uma análise de cluster é decidir se padronizará as variáveis de entrada. A padronização não é necessária quando as variáveis de entrada são medidas na mesma escala ou quando as variáveis de entrada são os coeficientes obtidos por uma análise conjunta. Em outros casos, a padronização é necessária.

No nosso exemplo, todas as variáveis de entrada são medidas na mesma escala e, portanto, a padronização não é necessária. Se necessário, isso pode ser feito facilmente com `mutate(newvar = scale(oldvar))`.

8.2.2 Cluster hierárquico

Em seguida, criamos um novo conjunto de dados que inclui apenas as variáveis de entrada, ou seja, as classificações:

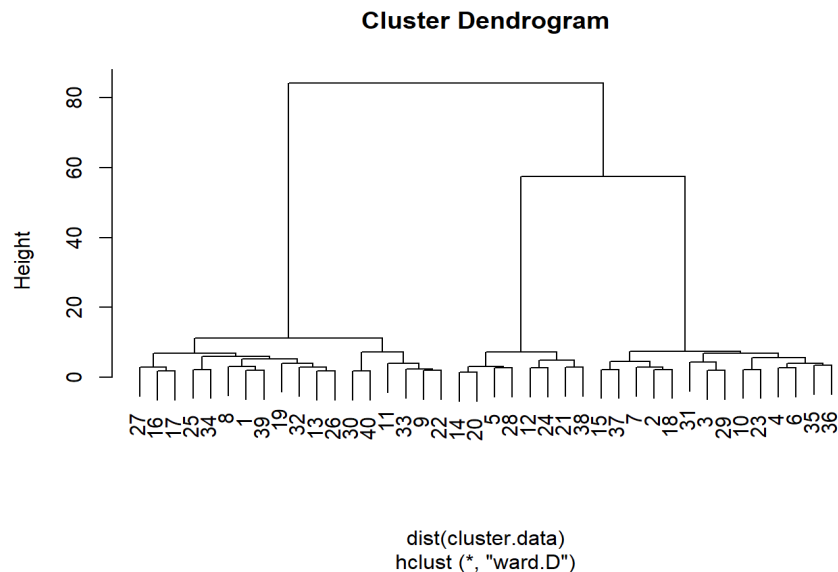
```
1 cluster.data <- equipment %>%
2   select(variety_of_choice, electronics, furniture, quality_of_service, low_prices, return_
3     policy) # Seleciona o conjunto de dados do equipamento apenas as variáveis com
4     classificacoes padronizadas
```

Agora, podemos prosseguir com o cluster hierárquico para determinar o número ideal de clusters:

```
1 # A funcao dist() cria uma matriz de dissimilaridade do nosso conjunto de dados e deve ser
2   o primeiro argumento para a funcao hclust().
3 # No argumento do metodo, voce pode especificar o metodo a ser usado para armazenamento em
4   cluster.
5 hierarchical.clustering <- hclust(dist(cluster.data), method = "ward.D")
```


A análise de cluster é armazenada no objeto `hierarchical.clustering` e pode ser facilmente visualizada por um dendrograma:

```
1 plot(hierarchical.clustering)
```



A partir desse dendrograma, parece que podemos dividir as observações em dois, três ou seis grupos de observações. Vamos realizar um teste formal, a regra de parada de Duda-Hart, para ver quantos clusters devemos reter. Para isso, precisamos (instalar e) carregar o pacote `NbClust`:

```
1 install.packages("NbClust")
2 library(NbClust)
```

A tabela de regras de parada de Duda-Hart pode ser obtida da seguinte maneira:

```
1 duda <- NbClust(cluster.data, distance = "euclidean", method = "ward.D2", max.nc = 9, index
  = "duda")
2 pseudot2 <- NbClust(cluster.data, distance = "euclidean", method = "ward.D2", max.nc = 9,
  index = "pseudot2")
3
4 duda$All.index
5
6 ##      2      3      4      5      6      7      8      9
7 ## 0.3746 0.7277 0.7530 0.5062 0.4169 0.5948 0.7536 0.8615
8 pseudot2$All.index
9 ##      2      3      4      5      6      7      8      9
10 ## 33.3846 5.9872 3.9369 5.8519 5.5945 3.4055 3.2695 1.2865
```

A sabedoria convencional para decidir o número de grupos com base na regra de parada de Duda-Hart é encontrar um dos maiores valores de Duda que corresponda a um baixo valor de pseudo- T^2 . No entanto, você também pode solicitar o número ideal de clusters, conforme sugerido pela regra de parada:

```
1 duda$Best.nc
2 ## Number_clusters Value_Index
3 ##      3.0000      0.7277
```

Nesse caso, o número ideal é três.

8.3 Cluster não-hierárquico

Agora, realizamos uma análise de cluster não hierárquica na qual solicitamos três clusters (conforme determinado pela análise de cluster hierárquica):

```
1 # existe um elemento aleatorio na analise de cluster
2 # isso significa que voce nem sempre obtera a mesma saida toda vez que fizer uma analise de
  cluster
3 # se voce deseja obter sempre a mesma saida, eh necessario corrigir o gerador de numeros
  aleatorios de R com o comando set.seed
4 set.seed (1)
5
6 # o argumento nstart deve ser incluido e definido como 25, mas sua explicacao esta fora do
  escopo deste tutorial
7 kmeans.clustering <- kmeans(cluster.data, 3, nstart = 25)
```

Adicione ao conjunto de `dados equipament` uma variável que indique a qual cluster uma observação pertence:

```
1 equipament <- equipament %>%
2   mutate(km.group = factor(kmeans.clustering$cluster, labels=c("c11","c12","c13"))) #
  Fatore o indicador de cluster a partir do quadro de dados clustering k e adicione-o ao
  quadro de dados do equipamento.
```

Inspeciona os clusters:

```
1 equipament %>%
2   group_by(km.group) %>% # agrupado por cluster (km.group)
3   summarise(count = n(),
4             variety = mean(variety_of_choice),
5             electronics = mean(electronics),
6             furniture = mean(furniture),
7             service = mean(quality_of_service),
8             prices = mean(low_prices),
9             return = mean(return_policy)) #Em seguida, pergunte pelo n mero de
  entrevistados e pelos meios das classifica es.
10
11 ## # A tibble: 3 x 8
12 ##   km.group count variety electronics furniture service prices return
13 ##   <fct>   <int>   <dbl>      <dbl>      <dbl>   <dbl>   <dbl>   <dbl>
14 ## 1 c11         8     5        4.38        1.75    8.5     2.5    4.38
15 ## 2 c12        18    9.11       6.06        5.78    2.39    3.67    3.17
16 ## 3 c13        14    6.93       2.79        1.43    3.5     8.29    6.29
```

Vemos que:

- o cluster 1 atribui mais importância (do que outros clusters) à qualidade do serviço
- o cluster 2 atribui mais importância à variedade de opções
- o cluster 3 atribui mais importância a preços baixos

Também podemos testar se há diferenças significativas entre os clusters, por exemplo, na variedade de opções. Para isso, usamos uma ANOVA unidirecional:

```
1 # remotes::install_github("samuelfranssens/type3anova") # para instalar o pacote type3anova
2 # Voce precisa do pacote de controles remotos para isso e o pacote para carro precisa ser
  instalado para que o pacote type3anova funcione
3
4 library(type3anova)
5 type3anova(lm(variety_of_choice ~ km.group, data=equipament))
6
7 ## # A tibble: 3 x 6
8 ##   term      ss df1 df2      f pvalue
9 ##   <chr>   <dbl> <dbl> <int>   <dbl>   <dbl>
10 ## 1 (Intercept) 1757.      1    37 1335. 1.24e-30
11 ## 2 km.group    101.      2    37   38.5 9.20e-10
12 ## 3 Residuals   48.7     37    37    NA     NA
```

Existem diferenças significativas entre os clusters em importância associadas à variedade de opções, e isso faz sentido porque o objetivo da análise de clusters é maximizar as diferenças entre os clusters. Vamos acompanhar isso com o HSD de Tukey para ver exatamente quais meios diferem um do outro:

```

1 TukeyHSD(aov(variety_of_choice ~ km.group, data=equipment),
2           "km.group") # 0 primeiro argumento eh um objeto "aov", o segundo eh a nossa
3           variavel independente.
4 ## Tukey multiple comparisons of means
5 ## 95% family-wise confidence level
6 ##
7 ## Fit: aov(formula = variety_of_choice ~ km.group, data = equipment)
8 ##
9 ## $km.group
10 ##           diff           lwr           upr           p adj
11 ## c12-cl1  4.111111  2.9208248  5.301397  0.0000000
12 ## c13-cl1  1.928571  0.6870668  3.170076  0.0015154
13 ## c13-cl2 -2.182540 -3.1807470 -1.184332  0.0000145

```

Vemos que em todos os meios, a diferença é significativa.

8.4 LDA Canônico

Na vida real, geralmente não sabemos o que os potenciais compradores consideram importante, mas temos uma ideia, por exemplo, de sua renda, idade e status profissional. Portanto, seria útil testar quão bem podemos prever a associação ao cluster (perfil de classificações de importância) com base nas características dos respondentes (renda, idade, profissional), que também são chamados de variáveis de segmentação. A fórmula preditiva poderia então ser usada para prever a associação ao cluster de novos compradores em potencial. Para encontrar a fórmula correta, usamos a análise discriminante linear (LDA). Mas primeiro vamos dar uma olhada nas médias de income, age, e professional por cluster:

```

1
2 equipment %>%
3   group_by(km.group) %>% # Agrupar equipamentos por cluster.
4   summarize(income = mean(income),
5             age = mean(age),
6             professional = mean(as.numeric(professional)-1))
7
8 ## # A tibble: 3 x 4
9 ##   km.group income  age professional
10 ##   <fct>    <dbl> <dbl>         <dbl>
11 ## 1 cl1      47.5  49          0.75
12 ## 2 cl2      48.3  44.2        0.333
13 ## 3 cl3      32.1  30.9         0.5
14 # Nao podemos assumir a media do profissional porque e uma variavel fator.
15 # Pedimos, portanto, que R o trate como uma variavel numerica.
16 # acessado por $class, para os new_data com base na formula do LDA com base nos dados
17   antigos (use o LDA onde CV = FALSE).
18
19 #visualiza as predicoes
20
21 new_data
22 ## # A tibble: 4 x 4
23 ##   income  age professional prediction
24 ##   <dbl> <dbl> <chr>         <fct>
25 ## 1    65    20 professional cluster 3
26 ## 2    65    35 non-professional cluster 2
27 ## 3    35    45 non-professional cluster 2
28 ## 4    35    60 professional cluster 1

```

9 Análise Conjunta

Neste capítulo, você aprenderá como realizar uma análise conjunta. A análise conjunta começa a partir de uma pesquisa na qual as pessoas avaliam ou escolhem entre produtos (por exemplo, carros) que diferem em vários atributos (por exemplo, segurança, eficiência de combustível, conforto etc.). A partir dessas classificações ou escolhas, a análise determina o valor que as pessoas atribuem aos diferentes atributos do produto (por exemplo, quanto peso as pessoas atribuem à segurança ao escolher entre carros). Essas informações podem ser usadas no desenvolvimento de produtos.

9.1 Dados

9.1.1 Importação

Analisaremos os dados de uma pesquisa em que 15 consumidores foram convidados a avaliar dez sorvetes. Cada sorvete tinha um 'perfil' diferente, ou seja, uma combinação diferente de níveis de quatro atributos: sabor (framboesa, chocolate, morango, manga, baunilha), embalagem (waffle caseiro, casquinha, caneca), leve (com pouca gordura ou não) e orgânico (orgânico ou não). Todos os 15 entrevistados classificaram os dez perfis, fornecendo uma pontuação entre 1 e 10.

Usamos os dados fornecidos pelo www.xlstat.com, descritos em seu [tutorial](#) sobre como fazer análises conjuntas no Excel. Faça o download dos dados [aqui](#).

```
1 library(tidyverse)
2 library(readxl)
3
4 icecream <- read_excel("icecream.xlsx") # Não é necessário incluir o argumento da planilha
    quando houver apenas uma planilha no arquivo do Excel
```

9.1.2 Manipulação

```
1 icecream
2
3 ## # A tibble: 10 x 20
4 ##   Observations Flavor Packaging Light Organic `Individual 1`
5 ##   <chr>         <chr>   <chr>   <chr> <chr>          <dbl>
6 ## 1 Profile 1    Raspb~ Homemade~ No 1~ Not or~      1
7 ## 2 Profile 2    Choco~ Cone     No 1~ Organic      4
8 ## 3 Profile 3    Raspb~ Pint     Low ~ Organic      2
9 ## 4 Profile 4    Straw~ Pint     No 1~ Organic      7
10 ## 5 Profile 5    Straw~ Cone     Low ~ Not or~     9
11 ## 6 Profile 6    Choco~ Homemade~ No 1~ Not or~     3
12 ## 7 Profile 7    Vanil~ Pint     Low ~ Not or~     5
13 ## 8 Profile 8    Mango  Homemade~ Low ~ Organic     10
14 ## 9 Profile 9    Mango  Pint     No 1~ Not or~     6
15 ## 10 Profile 10 Vanil~ Homemade~ No 1~ Organic     8
16 ## # ... with 14 more variables: `Individual 2` <dbl>, `Individual 3` <dbl>,
17 ## # `Individual 4` <dbl>, `Individual 5` <dbl>, `Individual 6` <dbl>,
18 ## # `Individual 7` <dbl>, `Individual 8` <dbl>, `Individual 9` <dbl>,
19 ## # `Individual 10` <dbl>, `Individual 11` <dbl>, `Individual 12` <dbl>,
20 ## # `Individual 13` <dbl>, `Individual 14` <dbl>, `Individual 15` <dbl>
```

Quando inspecionamos os dados, vemos que temos uma coluna para cada entrevistado. Essa é uma maneira incomum de armazenar dados (normalmente temos uma linha por respondente), então vamos reestruturar nosso conjunto de dados com a função de coleta (como fizemos [anteriormente](#)):

```
1 icecream <- icecream %>%
2   gather(respondent, rating, starts_with("Individual")) %>% # o entrevistado acompanha o
    respondente, a classificacao armazena as classificacoes do entrevistado e queremos
    empilhar todas as variaveis que comecam com Individual
3   rename("profile" = "Observations") %>% # renomeia Observations para profile
4   mutate(profile = factor(profile), respondent = factor(respondent), # fatorar
    identificadores
5     Flavor = factor(Flavor), Packaging = factor(Packaging), Light = factor(Light),
    Organic = factor(Organic)) # fatorar os atributos do sorvete
6
7
8 # Amplo conjunto de dados: uma linha por unidade de observacao (aqui: perfil) e varias
    colunas para as diferentes observacoes (aqui: respondentes)
9 # Conjunto de dados longo: uma linha por observacao (aqui: combinacao de perfil x
    respondente)
```

```

11 # Converter de grande para longo significa que estamos empilhando varias colunas umas sobre
    # as outras. Para isso, precisamos de uma variavel extra para acompanhar qual coluna
    # estamos lidando.
12 # A funcao de coleta converte conjuntos de dados de amplos para longos.
13 # O primeiro argumento (respondente) nos dira com qual coluna estamos lidando. Essa eh a
    # variavel que armazenara os nomes das colunas que estamos empilhando.
14 # O segundo argumento (classificacao) armazenara as colunas reais empilhadas umas sobre as
    # outras.
15 # Os argumentos a seguir (todas as variaveis com nomes que comecam com Individual) sao as
    # colunas que queremos empilhar.
16
17 icecream
18
19 ## # A tibble: 150 x 7
20 ##   profile Flavor Packaging Light Organic respondent rating
21 ##   <fct> <fct> <fct> <fct> <fct> <fct> <dbl>
22 ## 1 Profile 1 Raspberry Homemade waf ~ No low ~ Not orga ~ Individual ~ 1
23 ## 2 Profile 2 Chocolate Cone No low ~ Organic Individual ~ 4
24 ## 3 Profile 3 Raspberry Pint Low fat Organic Individual ~ 2
25 ## 4 Profile 4 Strawber ~ Pint No low ~ Organic Individual ~ 7
26 ## 5 Profile 5 Strawber ~ Cone Low fat Not orga ~ Individual ~ 9
27 ## 6 Profile 6 Chocolate Homemade waf ~ No low ~ Not orga ~ Individual ~ 3
28 ## 7 Profile 7 Vanilla Pint Low fat Not orga ~ Individual ~ 5
29 ## 8 Profile 8 Mango Homemade waf ~ Low fat Organic Individual ~ 10
30 ## 9 Profile ~ Mango Pint No low ~ Not orga ~ Individual ~ 6
31 ## 10 Profile ~ Vanilla Homemade waf ~ No low ~ Organic Individual ~ 8
32 ## # ... with 140 more rows

```

É melhor usar o Visualizador aqui (clique duas vezes no `icecream` objeto no painel Ambiente ou digite `View(icecream)`) para ver que existem dez linhas (10 perfis) por respondente agora.

As demais variáveis são:

- `profile` é um identificador para os diferentes sorvetes
- `Flavor`, `Packaging`, `Light`, `Organic` são os quatro atributos que compõem o perfil de um sorvete

9.1.3 Recapitulando: importação e manipulação

Aqui está o que fizemos até agora, em uma sequência ordenada de operações canalizadas (faça o download dos dados [aqui](#)):

```

1 library(tidyverse)
2 library(readxl)
3
4 icecream <- read_excel("icecream.xlsx") %>%
5   gather(respondent, rating, starts_with("Individual")) %>% # o entrevistado acompanha o
    # respondente, a classificacao armazena as classificacoes do entrevistado e queremos
    # empilhar todas as variaveis que comecam com Individual
6   rename("profile" = "Observations") %>%
7   mutate(profile = factor(profile), respondent = factor(respondent),
8     Flavor = factor(Flavor), Packaging = factor(Packaging), Light = factor(Light),
9     Organic = factor(Organic))

```

9.2 Design de experimentos

Quando inspecionamos nosso conjunto de dados, vemos que o `flavor` possui 5 níveis (framboesa, chocolate, morango, manga, baunilha), `packaging` possui 3 níveis (waffle caseiro, cone, cerveja), a `Light` possui 2 níveis (baixo teor de gordura versus não), e `Organic` tem 2 níveis (orgânico vs. não). O objetivo de uma análise conjunta é estimar até que ponto cada nível de atributo afeta a classificação do sorvete.

Para fazer isso, o fabricante de sorvete poderia criar $5 \times 3 \times 2 \times 2 = 60$ sorvetes diferentes e peça às pessoas para avaliarem tudo isso. Isso fornecerá ao fabricante uma boa estimativa da importância de cada atributo e de todas as possíveis interações. No entanto, classificar 60 sorvetes é difícil para os participantes e um estudo tão grande seria caro para o fabricante financiar. Na prática, os pesquisadores nessa situação solicitarão que as pessoas classifiquem um subconjunto desses 60 sorvetes. Nesta seção, discutiremos como selecionar um subconjunto (por exemplo, 10 sorvetes) de todas as combinações possíveis de nível de atributo (ou seja, 60 sorvetes) que ainda nos permitirão obter boas estimativas dos efeitos mais importantes.

No conjunto de dados, já temos as classificações para dez perfis, portanto a decisão de quais sorvetes para teste já foi tomada. No entanto, vamos desconsiderar o fato de já termos os dados e considerar as decisões

que precisam ser tomadas antes da coleta de dados. Em outras palavras, vamos discutir como passamos de um *fatorial completo* (todas as 60 combinações) para um design *fracionário* (menos de 60 combinações).

A função `doe` (projeto de experimentos) do pacote `radiant` nos ajudará a decidir sobre os projetos de estudo. `Radiant` é um pacote do R para business analytics.

A discussão a seguir da função `doe` é baseada na discussão da `Radiant` sobre essa função.

```
1 install.packages("radiant")
2 library(radiant)
```

Para usar a `doe`, precisamos inserir as informações sobre nossos atributos e seus níveis de uma maneira específica:

```
1 # attribute1, attribute2, etc. sao vetores com um elemento no qual fornecemos primeiro o
  # nome do atributo seguido por um ponto e virgula e depois fornecemos todos os niveis dos
  # atributos separados por ponto e virgula
2 attribute1 <- "Flavor; Raspberry; Chocolate; Strawberry; Mango; Vanilla"
3 attribute2 <- "Package; Homemade waffle; Cone; Pint"
4 attribute3 <- "Light; Low fat; No low fat"
5 attribute4 <- "Organic; Organic; Not organic"
6
7 # agora combine esses diferentes atributos em um vetor com c()
8 attributes <- c(attribute1, attribute2, attribute3, attribute4)
```

Agora podemos pedir possíveis projetos experimentais:

```
1 summary(doe(attributes, seed = 123)) # Seed: fixa o gerador de numeros aleatorios
2
3 ## Experimental design
4 ## # trials for partial factorial: 60
5 ## # trials for full factorial      : 60
6 ## Random seed                     : 123
7 ##
8 ## Attributes and levels:
9 ## Flavor: Raspberry, Chocolate, Strawberry, Mango, Vanilla
10 ## Package: Homemade_waffle, Cone, Pint
11 ## Light: Low_fat, No_low_fat
12 ## Organic: Organic, Not_organic
13 ##
14 ## Design efficiency:
15 ##   Trials D-efficiency Balanced
16 ##      9      0.105      FALSE
17 ##     10      0.389      FALSE
18 ##     11      0.411      FALSE
19 ##     12      0.614      FALSE
20 ##     13      0.542      FALSE
21 ##     14      0.479      FALSE
22 ##     15      0.762      FALSE
23 ##     16      0.738      FALSE
24 ##     17      0.748      FALSE
25 ##     18      0.756      FALSE
26 ##     19      0.644      FALSE
27 ##     20      0.895      FALSE
28 ##     21      0.848      FALSE
29 ##     22      0.833      FALSE
30 ##     23      0.790      FALSE
31 ##     24      0.827      FALSE
32 ##     25      0.787      FALSE
33 ##     26      0.768      FALSE
34 ##     27      0.759      FALSE
35 ##     28      0.736      FALSE
36 ##     29      0.702      FALSE
37 ##     30      0.984      TRUE
38 ##     31      0.952      FALSE
39 ##     32      0.933      FALSE
40 ##     33      0.928      FALSE
41 ##     34      0.900      FALSE
42 ##     35      0.871      FALSE
43 ##     36      0.893      FALSE
44 ##     37      0.866      FALSE
45 ##     38      0.843      FALSE
46 ##     39      0.836      FALSE
47 ##     40      0.922      FALSE
48 ##     41      0.899      FALSE
49 ##     42      0.904      FALSE
```

```

50 ##      43      0.882  FALSE
51 ##      44      0.861  FALSE
52 ##      45      0.949  FALSE
53 ##      46      0.919  FALSE
54 ##      47      0.912  FALSE
55 ##      48      0.911  FALSE
56 ##      49      0.891  FALSE
57 ##      50      0.959  FALSE
58 ##      51      0.939  FALSE
59 ##      52      0.944  FALSE
60 ##      53      0.925  FALSE
61 ##      54      0.924  FALSE
62 ##      55      0.906  FALSE
63 ##      56      0.902  FALSE
64 ##      57      0.884  FALSE
65 ##      58      0.872  FALSE
66 ##      59      0.855  FALSE
67 ##      60      1.000   TRUE
68 ##
69 ## Partial factorial design correlations:
70 ##      Flavor Package Light Organic
71 ## Flavor      1      0      0      0
72 ## Package      0      1      0      0
73 ## Light        0      0      1      0
74 ## Organic      0      0      0      1
75 ##
76 ## Partial factorial design:
77 ## trial      Flavor      Package      Light      Organic
78 ##      1 Raspberry Homemade_waffle Low_fat Organic
79 ##      2 Raspberry Homemade_waffle Low_fat Not_organic
80 ##      3 Raspberry Homemade_waffle No_low_fat Organic
81 ##      4 Raspberry Homemade_waffle No_low_fat Not_organic
82 ##      5 Raspberry Cone Low_fat Organic
83 ##      6 Raspberry Cone Low_fat Not_organic
84 ##      7 Raspberry Cone No_low_fat Organic
85 ##      8 Raspberry Cone No_low_fat Not_organic
86 ##      9 Raspberry Pint Low_fat Organic
87 ##     10 Raspberry Pint Low_fat Not_organic
88 ##     11 Raspberry Pint No_low_fat Organic
89 ##     12 Raspberry Pint No_low_fat Not_organic
90 ##     13 Chocolate Homemade_waffle Low_fat Organic
91 ##     14 Chocolate Homemade_waffle Low_fat Not_organic
92 ##     15 Chocolate Homemade_waffle No_low_fat Organic
93 ##     16 Chocolate Homemade_waffle No_low_fat Not_organic
94 ##     17 Chocolate Cone Low_fat Organic
95 ##     18 Chocolate Cone Low_fat Not_organic
96 ##     19 Chocolate Cone No_low_fat Organic
97 ##     20 Chocolate Cone No_low_fat Not_organic
98 ##     21 Chocolate Pint Low_fat Organic
99 ##     22 Chocolate Pint Low_fat Not_organic
100 ##     23 Chocolate Pint No_low_fat Organic
101 ##     24 Chocolate Pint No_low_fat Not_organic
102 ##     25 Strawberry Homemade_waffle Low_fat Organic
103 ##     26 Strawberry Homemade_waffle Low_fat Not_organic
104 ##     27 Strawberry Homemade_waffle No_low_fat Organic
105 ##     28 Strawberry Homemade_waffle No_low_fat Not_organic
106 ##     29 Strawberry Cone Low_fat Organic
107 ##     30 Strawberry Cone Low_fat Not_organic
108 ##     31 Strawberry Cone No_low_fat Organic
109 ##     32 Strawberry Cone No_low_fat Not_organic
110 ##     33 Strawberry Pint Low_fat Organic
111 ##     34 Strawberry Pint Low_fat Not_organic
112 ##     35 Strawberry Pint No_low_fat Organic
113 ##     36 Strawberry Pint No_low_fat Not_organic
114 ##     37 Mango Homemade_waffle Low_fat Organic
115 ##     38 Mango Homemade_waffle Low_fat Not_organic
116 ##     39 Mango Homemade_waffle No_low_fat Organic
117 ##     40 Mango Homemade_waffle No_low_fat Not_organic
118 ##     41 Mango Cone Low_fat Organic
119 ##     42 Mango Cone Low_fat Not_organic
120 ##     43 Mango Cone No_low_fat Organic
121 ##     44 Mango Cone No_low_fat Not_organic
122 ##     45 Mango Pint Low_fat Organic
123 ##     46 Mango Pint Low_fat Not_organic
124 ##     47 Mango Pint No_low_fat Organic

```

```

125 ##      48      Mango      Pint No_low_fat Not_organic
126 ##      49      Vanilla Homemade_waffle Low_fat Organic
127 ##      50      Vanilla Homemade_waffle Low_fat Not_organic
128 ##      51      Vanilla Homemade_waffle No_low_fat Organic
129 ##      52      Vanilla Homemade_waffle No_low_fat Not_organic
130 ##      53      Vanilla      Cone Low_fat Organic
131 ##      54      Vanilla      Cone Low_fat Not_organic
132 ##      55      Vanilla      Cone No_low_fat Organic
133 ##      56      Vanilla      Cone No_low_fat Not_organic
134 ##      57      Vanilla      Pint Low_fat Organic
135 ##      58      Vanilla      Pint Low_fat Not_organic
136 ##      59      Vanilla      Pint No_low_fat Organic
137 ##      60      Vanilla      Pint No_low_fat Not_organic
138 ##
139 ## Full factorial design:
140 ##      trial      Flavor      Package      Light      Organic
141 ##          1  Raspberry Homemade_waffle Low_fat Organic
142 ##          2  Raspberry Homemade_waffle Low_fat Not_organic
143 ##          3  Raspberry Homemade_waffle No_low_fat Organic
144 ##          4  Raspberry Homemade_waffle No_low_fat Not_organic
145 ##          5  Raspberry      Cone Low_fat Organic
146 ##          6  Raspberry      Cone Low_fat Not_organic
147 ##          7  Raspberry      Cone No_low_fat Organic
148 ##          8  Raspberry      Cone No_low_fat Not_organic
149 ##          9  Raspberry      Pint Low_fat Organic
150 ##         10  Raspberry      Pint Low_fat Not_organic
151 ##         11  Raspberry      Pint No_low_fat Organic
152 ##         12  Raspberry      Pint No_low_fat Not_organic
153 ##         13  Chocolate Homemade_waffle Low_fat Organic
154 ##         14  Chocolate Homemade_waffle Low_fat Not_organic
155 ##         15  Chocolate Homemade_waffle No_low_fat Organic
156 ##         16  Chocolate Homemade_waffle No_low_fat Not_organic
157 ##         17  Chocolate      Cone Low_fat Organic
158 ##         18  Chocolate      Cone Low_fat Not_organic
159 ##         19  Chocolate      Cone No_low_fat Organic
160 ##         20  Chocolate      Cone No_low_fat Not_organic
161 ##         21  Chocolate      Pint Low_fat Organic
162 ##         22  Chocolate      Pint Low_fat Not_organic
163 ##         23  Chocolate      Pint No_low_fat Organic
164 ##         24  Chocolate      Pint No_low_fat Not_organic
165 ##         25  Strawberry Homemade_waffle Low_fat Organic
166 ##         26  Strawberry Homemade_waffle Low_fat Not_organic
167 ##         27  Strawberry Homemade_waffle No_low_fat Organic
168 ##         28  Strawberry Homemade_waffle No_low_fat Not_organic
169 ##         29  Strawberry      Cone Low_fat Organic
170 ##         30  Strawberry      Cone Low_fat Not_organic
171 ##         31  Strawberry      Cone No_low_fat Organic
172 ##         32  Strawberry      Cone No_low_fat Not_organic
173 ##         33  Strawberry      Pint Low_fat Organic
174 ##         34  Strawberry      Pint Low_fat Not_organic
175 ##         35  Strawberry      Pint No_low_fat Organic
176 ##         36  Strawberry      Pint No_low_fat Not_organic
177 ##         37      Mango Homemade_waffle Low_fat Organic
178 ##         38      Mango Homemade_waffle Low_fat Not_organic
179 ##         39      Mango Homemade_waffle No_low_fat Organic
180 ##         40      Mango Homemade_waffle No_low_fat Not_organic
181 ##         41      Mango      Cone Low_fat Organic
182 ##         42      Mango      Cone Low_fat Not_organic
183 ##         43      Mango      Cone No_low_fat Organic
184 ##         44      Mango      Cone No_low_fat Not_organic
185 ##         45      Mango      Pint Low_fat Organic
186 ##         46      Mango      Pint Low_fat Not_organic
187 ##         47      Mango      Pint No_low_fat Organic
188 ##         48      Mango      Pint No_low_fat Not_organic
189 ##         49      Vanilla Homemade_waffle Low_fat Organic
190 ##         50      Vanilla Homemade_waffle Low_fat Not_organic
191 ##         51      Vanilla Homemade_waffle No_low_fat Organic
192 ##         52      Vanilla Homemade_waffle No_low_fat Not_organic
193 ##         53      Vanilla      Cone Low_fat Organic
194 ##         54      Vanilla      Cone Low_fat Not_organic
195 ##         55      Vanilla      Cone No_low_fat Organic
196 ##         56      Vanilla      Cone No_low_fat Not_organic
197 ##         57      Vanilla      Pint Low_fat Organic
198 ##         58      Vanilla      Pint Low_fat Not_organic
199 ##         59      Vanilla      Pint No_low_fat Organic

```



```
200 ##      60      Vanilla      Pint No_low_fat Not_organic
```

Observe a saída no cabeçalho `Design efficiency`. Mostra 52 linhas. As linhas representam projetos experimentais com diferentes números de `Trials` ou diferentes números de sorvetes (ou seja, combinações de nível de atributo) que seriam testados. Uma palavra melhor para julgamento é perfil. Para cada projeto experimental, ele mostra a `D-efficiency` do projeto - uma medida de como poderemos estimar com clareza os efeitos do interesse após a execução do experimento (pontuações mais altas são melhores) - e se o projeto está ou não equilibrado - se cada nível está incluído no mesmo número de tentativas ou perfis. Idealmente, procuramos projetos balanceados com alta `D-efficiency` (acima de 0,80 é considerado razoável). Temos dois candidatos, um delineamento experimental com 60 perfis, que é apenas o delineamento fatorial completo ou um delineamento com 30 perfis. Vamos dar uma olhada no design com 30 perfis:

```
1 summary(doe(attributes, seed = 123, trials = 30))
2
3 ## Experimental design
4 ## # trials for partial factorial: 30
5 ## # trials for full factorial : 60
6 ## Random seed : 123
7 ##
8 ## Attributes and levels:
9 ## Flavor: Raspberry, Chocolate, Strawberry, Mango, Vanilla
10 ## Package: Homemade_waffle, Cone, Pint
11 ## Light: Low_fat, No_low_fat
12 ## Organic: Organic, Not_organic
13 ##
14 ## Design efficiency:
15 ## Trials D-efficiency Balanced
16 ## 30 0.984 TRUE
17 ##
18 ## Partial factorial design correlations:
19 ## Flavor Package Light Organic
20 ## Flavor 1 0 0.000 0.000
21 ## Package 0 1 0.000 0.000
22 ## Light 0 0 1.000 -0.105
23 ## Organic 0 0 -0.105 1.000
24 ##
25 ## Partial factorial design:
26 ## trial Flavor Package Light Organic
27 ## 1 Raspberry Homemade_waffle Low_fat Organic
28 ## 4 Raspberry Homemade_waffle No_low_fat Not_organic
29 ## 6 Raspberry Cone Low_fat Not_organic
30 ## 7 Raspberry Cone No_low_fat Organic
31 ## 10 Raspberry Pint Low_fat Not_organic
32 ## 11 Raspberry Pint No_low_fat Organic
33 ## 13 Chocolate Homemade_waffle Low_fat Organic
34 ## 14 Chocolate Homemade_waffle Low_fat Not_organic
35 ## 19 Chocolate Cone No_low_fat Organic
36 ## 20 Chocolate Cone No_low_fat Not_organic
37 ## 22 Chocolate Pint Low_fat Not_organic
38 ## 23 Chocolate Pint No_low_fat Organic
39 ## 26 Strawberry Homemade_waffle Low_fat Not_organic
40 ## 28 Strawberry Homemade_waffle No_low_fat Not_organic
41 ## 29 Strawberry Cone Low_fat Organic
42 ## 32 Strawberry Cone No_low_fat Not_organic
43 ## 33 Strawberry Pint Low_fat Organic
44 ## 35 Strawberry Pint No_low_fat Organic
45 ## 39 Mango Homemade_waffle No_low_fat Organic
46 ## 40 Mango Homemade_waffle No_low_fat Not_organic
47 ## 41 Mango Cone Low_fat Organic
48 ## 42 Mango Cone Low_fat Not_organic
49 ## 46 Mango Pint Low_fat Not_organic
50 ## 47 Mango Pint No_low_fat Organic
51 ## 49 Vanilla Homemade_waffle Low_fat Organic
52 ## 51 Vanilla Homemade_waffle No_low_fat Organic
53 ## 53 Vanilla Cone Low_fat Organic
54 ## 56 Vanilla Cone No_low_fat Not_organic
55 ## 58 Vanilla Pint Low_fat Not_organic
56 ## 60 Vanilla Pint No_low_fat Not_organic
57 ##
58 ## Full factorial design:
59 ## trial Flavor Package Light Organic
60 ## 1 Raspberry Homemade_waffle Low_fat Organic
```

```

61 ##      2 Raspberry Homemade_waffle Low_fat Not_organic
62 ##      3 Raspberry Homemade_waffle No_low_fat Organic
63 ##      4 Raspberry Homemade_waffle No_low_fat Not_organic
64 ##      5 Raspberry Cone Low_fat Organic
65 ##      6 Raspberry Cone Low_fat Not_organic
66 ##      7 Raspberry Cone No_low_fat Organic
67 ##      8 Raspberry Cone No_low_fat Not_organic
68 ##      9 Raspberry Pint Low_fat Organic
69 ##     10 Raspberry Pint Low_fat Not_organic
70 ##     11 Raspberry Pint No_low_fat Organic
71 ##     12 Raspberry Pint No_low_fat Not_organic
72 ##     13 Chocolate Homemade_waffle Low_fat Organic
73 ##     14 Chocolate Homemade_waffle Low_fat Not_organic
74 ##     15 Chocolate Homemade_waffle No_low_fat Organic
75 ##     16 Chocolate Homemade_waffle No_low_fat Not_organic
76 ##     17 Chocolate Cone Low_fat Organic
77 ##     18 Chocolate Cone Low_fat Not_organic
78 ##     19 Chocolate Cone No_low_fat Organic
79 ##     20 Chocolate Cone No_low_fat Not_organic
80 ##     21 Chocolate Pint Low_fat Organic
81 ##     22 Chocolate Pint Low_fat Not_organic
82 ##     23 Chocolate Pint No_low_fat Organic
83 ##     24 Chocolate Pint No_low_fat Not_organic
84 ##     25 Strawberry Homemade_waffle Low_fat Organic
85 ##     26 Strawberry Homemade_waffle Low_fat Not_organic
86 ##     27 Strawberry Homemade_waffle No_low_fat Organic
87 ##     28 Strawberry Homemade_waffle No_low_fat Not_organic
88 ##     29 Strawberry Cone Low_fat Organic
89 ##     30 Strawberry Cone Low_fat Not_organic
90 ##     31 Strawberry Cone No_low_fat Organic
91 ##     32 Strawberry Cone No_low_fat Not_organic
92 ##     33 Strawberry Pint Low_fat Organic
93 ##     34 Strawberry Pint Low_fat Not_organic
94 ##     35 Strawberry Pint No_low_fat Organic
95 ##     36 Strawberry Pint No_low_fat Not_organic
96 ##     37 Mango Homemade_waffle Low_fat Organic
97 ##     38 Mango Homemade_waffle Low_fat Not_organic
98 ##     39 Mango Homemade_waffle No_low_fat Organic
99 ##     40 Mango Homemade_waffle No_low_fat Not_organic
100 ##     41 Mango Cone Low_fat Organic
101 ##     42 Mango Cone Low_fat Not_organic
102 ##     43 Mango Cone No_low_fat Organic
103 ##     44 Mango Cone No_low_fat Not_organic
104 ##     45 Mango Pint Low_fat Organic
105 ##     46 Mango Pint Low_fat Not_organic
106 ##     47 Mango Pint No_low_fat Organic
107 ##     48 Mango Pint No_low_fat Not_organic
108 ##     49 Vanilla Homemade_waffle Low_fat Organic
109 ##     50 Vanilla Homemade_waffle Low_fat Not_organic
110 ##     51 Vanilla Homemade_waffle No_low_fat Organic
111 ##     52 Vanilla Homemade_waffle No_low_fat Not_organic
112 ##     53 Vanilla Cone Low_fat Organic
113 ##     54 Vanilla Cone Low_fat Not_organic
114 ##     55 Vanilla Cone No_low_fat Organic
115 ##     56 Vanilla Cone No_low_fat Not_organic
116 ##     57 Vanilla Pint Low_fat Organic
117 ##     58 Vanilla Pint Low_fat Not_organic
118 ##     59 Vanilla Pint No_low_fat Organic
119 ##     60 Vanilla Pint No_low_fat Not_organic

```

Em Projeto fatorial parcial (ou projeto fatorial fracionário), encontramos os perfis que poderíamos executar em um experimento com 30 em vez de 60 perfis. Sob correlações parciais do projeto fatorial, vemos que dois atributos estão correlacionados, a saber, `Light` e `Organic` ($r = -0,105$). Esse sempre será o caso em projetos fatoriais fracionários. Isso significa que algumas combinações de níveis de atributo serão mais prevalentes que outras. Somente em um planejamento fatorial completo todos os atributos serão não correlacionados ou ortogonais.

Um design possível com apenas 10 perfis seria desequilibrado e teria a seguinte aparência:

```

1 summary(doe(attributes, seed = 123, trials = 10))
2
3 ## Experimental design
4 ## # trials for partial factorial: 10
5 ## # trials for full factorial : 60
6 ## Random seed : 123

```

```

7 ##
8 ## Attributes and levels:
9 ## Flavor: Raspberry, Chocolate, Strawberry, Mango, Vanilla
10 ## Package: Homemade_waffle, Cone, Pint
11 ## Light: Low_fat, No_low_fat
12 ## Organic: Organic, Not_organic
13 ##
14 ## Design efficiency:
15 ## Trials D-efficiency Balanced
16 ##      10      0.389      FALSE
17 ##
18 ## Partial factorial design correlations:
19 ##      Flavor Package Light Organic
20 ## Flavor   1.000   0.121 0.000   0.000
21 ## Package  0.121   1.000 0.000   0.000
22 ## Light    0.000   0.000 1.000   0.309
23 ## Organic  0.000   0.000 0.309   1.000
24 ##
25 ## Partial factorial design:
26 ##      trial      Flavor      Package      Light      Organic
27 ##      4  Raspberry  Homemade_waffle  No_low_fat  Not_organic
28 ##      5  Raspberry           Cone    Low_fat    Organic
29 ##     20  Chocolate           Cone  No_low_fat  Not_organic
30 ##     21  Chocolate           Pint    Low_fat    Organic
31 ##     25  Strawberry  Homemade_waffle  Low_fat    Organic
32 ##     36  Strawberry           Pint  No_low_fat  Not_organic
33 ##     39      Mango  Homemade_waffle  No_low_fat    Organic
34 ##     46      Mango           Pint    Low_fat  Not_organic
35 ##     50  Vanilla  Homemade_waffle  Low_fat  Not_organic
36 ##     59  Vanilla           Pint  No_low_fat    Organic
37 ##
38 ## Full factorial design:
39 ##      trial      Flavor      Package      Light      Organic
40 ##      1  Raspberry  Homemade_waffle  Low_fat    Organic
41 ##      2  Raspberry  Homemade_waffle  Low_fat  Not_organic
42 ##      3  Raspberry  Homemade_waffle  No_low_fat    Organic
43 ##      4  Raspberry  Homemade_waffle  No_low_fat  Not_organic
44 ##      5  Raspberry           Cone    Low_fat    Organic
45 ##      6  Raspberry           Cone    Low_fat  Not_organic
46 ##      7  Raspberry           Cone  No_low_fat    Organic
47 ##      8  Raspberry           Cone  No_low_fat  Not_organic
48 ##      9  Raspberry           Pint    Low_fat    Organic
49 ##     10  Raspberry           Pint    Low_fat  Not_organic
50 ##     11  Raspberry           Pint  No_low_fat    Organic
51 ##     12  Raspberry           Pint  No_low_fat  Not_organic
52 ##     13  Chocolate  Homemade_waffle  Low_fat    Organic
53 ##     14  Chocolate  Homemade_waffle  Low_fat  Not_organic
54 ##     15  Chocolate  Homemade_waffle  No_low_fat    Organic
55 ##     16  Chocolate  Homemade_waffle  No_low_fat  Not_organic
56 ##     17  Chocolate           Cone    Low_fat    Organic
57 ##     18  Chocolate           Cone    Low_fat  Not_organic
58 ##     19  Chocolate           Cone  No_low_fat    Organic
59 ##     20  Chocolate           Cone  No_low_fat  Not_organic
60 ##     21  Chocolate           Pint    Low_fat    Organic
61 ##     22  Chocolate           Pint    Low_fat  Not_organic
62 ##     23  Chocolate           Pint  No_low_fat    Organic
63 ##     24  Chocolate           Pint  No_low_fat  Not_organic
64 ##     25  Strawberry  Homemade_waffle  Low_fat    Organic
65 ##     26  Strawberry  Homemade_waffle  Low_fat  Not_organic
66 ##     27  Strawberry  Homemade_waffle  No_low_fat    Organic
67 ##     28  Strawberry  Homemade_waffle  No_low_fat  Not_organic
68 ##     29  Strawberry           Cone    Low_fat    Organic
69 ##     30  Strawberry           Cone    Low_fat  Not_organic
70 ##     31  Strawberry           Cone  No_low_fat    Organic
71 ##     32  Strawberry           Cone  No_low_fat  Not_organic
72 ##     33  Strawberry           Pint    Low_fat    Organic
73 ##     34  Strawberry           Pint    Low_fat  Not_organic
74 ##     35  Strawberry           Pint  No_low_fat    Organic
75 ##     36  Strawberry           Pint  No_low_fat  Not_organic
76 ##     37      Mango  Homemade_waffle  Low_fat    Organic
77 ##     38      Mango  Homemade_waffle  Low_fat  Not_organic
78 ##     39      Mango  Homemade_waffle  No_low_fat    Organic
79 ##     40      Mango  Homemade_waffle  No_low_fat  Not_organic
80 ##     41      Mango           Cone    Low_fat    Organic
81 ##     42      Mango           Cone    Low_fat  Not_organic

```

```

82 ##      43      Mango      Cone No_low_fat      Organic
83 ##      44      Mango      Cone No_low_fat Not_organic
84 ##      45      Mango      Pint  Low_fat      Organic
85 ##      46      Mango      Pint  Low_fat Not_organic
86 ##      47      Mango      Pint No_low_fat      Organic
87 ##      48      Mango      Pint No_low_fat Not_organic
88 ##      49      Vanilla Homemade_waffle Low_fat      Organic
89 ##      50      Vanilla Homemade_waffle Low_fat Not_organic
90 ##      51      Vanilla Homemade_waffle No_low_fat      Organic
91 ##      52      Vanilla Homemade_waffle No_low_fat Not_organic
92 ##      53      Vanilla      Cone  Low_fat      Organic
93 ##      54      Vanilla      Cone  Low_fat Not_organic
94 ##      55      Vanilla      Cone No_low_fat      Organic
95 ##      56      Vanilla      Cone No_low_fat Not_organic
96 ##      57      Vanilla      Pint  Low_fat      Organic
97 ##      58      Vanilla      Pint  Low_fat Not_organic
98 ##      59      Vanilla      Pint No_low_fat      Organic
99 ##      60      Vanilla      Pint No_low_fat Not_organic

```

Comparado ao design com 30 perfis, agora existem mais e mais fortes correlações entre os atributos.

Observe que os perfis não são exatamente iguais aos da experiência usada para coletar os dados do sorvete. Isso ocorre porque, para projetos desequilibrados, existe alguma aleatoriedade na definição das combinações reais de nível de atributo. É também por isso que definimos `seed = 123`. `seed` é usado para consertar o gerador de números aleatórios do R. Configurá-lo para um número fixo (123 ou 456 ou qualquer outra coisa) garantirá que o R gere sempre a mesma saída. Sem definir a `seed`, doe com `trials = 10` não daria o mesmo design fracionário toda vez que você a executar.

Observe também que o `pacote radiant` instala um suplemento que você pode acessar por meio de suplementos (encontre este botão à direita da linha abaixo de "Arquivo" etc.) -> Iniciar radiant (navegador). Isso abrirá um aplicativo no seu navegador que permitirá executar as etapas acima em uma interface visual intuitiva. Para obter ajuda, confira a discussão da `Radiant` sobre o módulo Design de experiências [aqui](#).

9.3 Um respondente

9.3.1 Estimar valores de peça e pesos de importância

Embora alguns softwares exijam que você primeiro crie variáveis fictícias representando os níveis de atributo e execute uma regressão, o `Radiant` não exige que você faça isso. Você pode simplesmente usar os atributos (cada um com vários níveis) como variáveis. Primeiro, faremos uma análise conjunta dos dados de um respondente (indivíduo 1):

```

1 respondent1 <- icecream %>% filter(respondent == "Individual 1")
2
3 # salve a análise conjunta em um objeto, porque a usaremos como entrada para summary(),
  plot(), and predict()
4
5 conjoint_respondent1 <- conjoint(respondent1, rvar = "rating", evar = c("Flavor", "Packaging",
  "Light", "Organic"))
6
7 summary(conjoint_respondent1)
8
9 ## Conjoint analysis
10 ## Data : respondent1
11 ## Response variable : rating
12 ## Explanatory variables: Flavor, Packaging, Light, Organic
13 ##
14 ## Conjoint part-worths:
15 ## Attributes Levels PW
16 ## Flavor Chocolate 0.000
17 ## Flavor Mango 5.000
18 ## Flavor Raspberry -1.500
19 ## Flavor Strawberry 5.000
20 ## Flavor Vanilla 3.500
21 ## Packaging Cone 0.000
22 ## Packaging Homemade waffle 0.000
23 ## Packaging Pint -2.000
24 ## Light Low fat 0.000
25 ## Light No low fat -1.000
26 ## Organic Not organic 0.000
27 ## Organic Organic 1.400
28 ## Base utility ~ 3.800

```

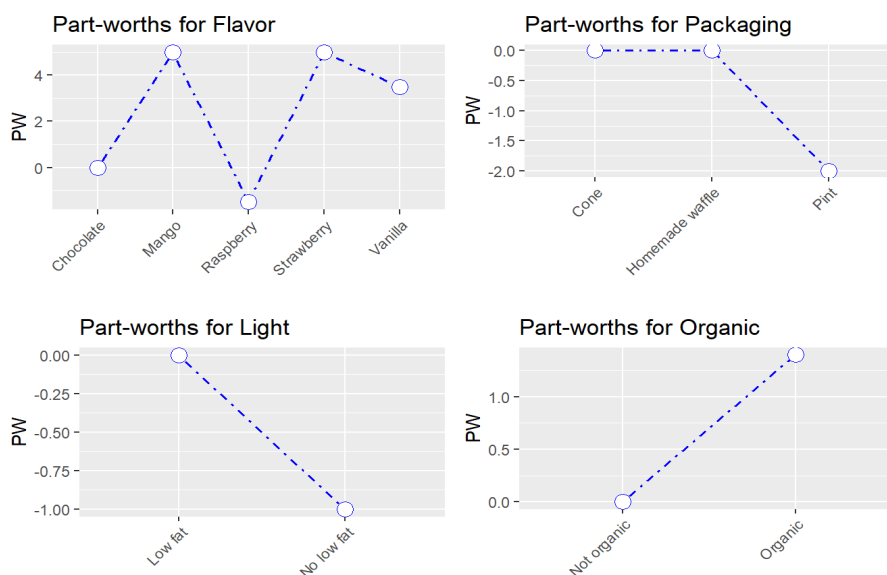
```

29 ##
30 ## Conjoint importance weights:
31 ##   Attributes    IW
32 ##   Flavor      0.596
33 ##   Packaging    0.183
34 ##   Light       0.092
35 ##   Organic     0.128
36 ##
37 ## Conjoint regression results:
38 ##
39 ##                               coefficient
40 ## (Intercept)                   3.800
41 ## Flavor|Mango                   5.000
42 ## Flavor|Raspberry              -1.500
43 ## Flavor|Strawberry              5.000
44 ## Flavor|Vanilla                 3.500
45 ## Packaging|Homemade waffle       0.000
46 ## Packaging|Pint                  -2.000
47 ## Light|No low fat               -1.000
48 ## Organic|Organic                1.400

```

A saída fornece valores de peça, pesos de importância e coeficientes de regressão. Os valores das partes e os coeficientes de regressão fornecem as mesmas informações: comparado ao nível de referência (o primeiro nível de um atributo; você verá que os valores das partes são sempre zero para este nível), quanto aumenta cada nível de atributo ou diminuir a classificação de um sorvete? Podemos traçar estes resultados:

```
1 plot(conjoint_respondent1)
```



E então vemos facilmente que essa pessoa desfrutaria de um sorvete com baixo teor de gordura, orgânico, manga ou morango em um cone ou em um waffle caseiro.

Observe que os resultados da regressão conjunta são simplesmente os resultados de uma [regressão linear múltipla](#):

```

1 # Execute essa regressão se estiver interessado em aprender qual preditor é significativo
  ou qual é o R quadrado do modelo geral.
2
3 summary(lm(rating ~ Flavor + Packaging + Light + Organic, data = respondent1))
4
5 ##
6 ## Call:
7 ## lm(formula = rating ~ Flavor + Packaging + Light + Organic, data = respondent1)
8 ##
9 ## Residuals:
10 ##      1      2      3      4      5      6      7      8      9     10
11 ## -0.3 -0.2  0.3 -0.2  0.2  0.2 -0.3 -0.2  0.2  0.3
12 ##
13 ## Coefficients:
14 ##                               Estimate Std. Error t value Pr(>|t|)

```

```

15 ## (Intercept)          3.800e+00  8.426e-01  4.510    0.139
16 ## FlavorMango          5.000e+00  9.747e-01  5.130    0.123
17 ## FlavorRaspberry     -1.500e+00  9.747e-01 -1.539    0.367
18 ## FlavorStrawberry     5.000e+00  8.660e-01  5.774    0.109
19 ## FlavorVanilla        3.500e+00  9.747e-01  3.591    0.173
20 ## PackagingHomemade waffle 1.570e-15  8.944e-01  0.000    1.000
21 ## PackagingPint         -2.000e+00  8.660e-01 -2.309    0.260
22 ## LightNo low fat      -1.000e+00  5.916e-01 -1.690    0.340
23 ## OrganicOrganic       1.400e+00  4.899e-01  2.858    0.214
24 ##
25 ## Residual standard error: 0.7746 on 1 degrees of freedom
26 ## Multiple R-squared:  0.9927, Adjusted R-squared:  0.9345
27 ## F-statistic: 17.06 on 8 and 1 DF,  p-value: 0.1852

```

Finalmente, os pesos de importância nos dizem com que intensidade cada atributo determina a classificação de um sorvete. Para esse respondente, sabor é o atributo mais importante e luz é o atributo menos importante. A classificação deste respondente é determinada em 59,6% por sabor e em 9,2% por luz.

9.3.2 Profiles: utilitários previstos

Prever as classificações (utilitários) dos diferentes sorvetes é muito fácil em R. Primeiro, verifique se temos um conjunto de dados com os diferentes perfis que foram testados:

```

1 profiles <- icecream %>%
2   filter(respondent == "Individual 1") %>%
3   select(Flavor, Packaging, Light, Organic)
4
5 profiles
6 ## # A tibble: 10 x 4
7 ##   Flavor      Packaging      Light      Organic
8 ##   <fct>      <fct>      <fct>      <fct>
9 ## 1 Raspberry  Homemade waffle No low fat Not organic
10 ## 2 Chocolate  Cone         No low fat Organic
11 ## 3 Raspberry  Pint         Low fat    Organic
12 ## 4 Strawberry Pint         No low fat Organic
13 ## 5 Strawberry Cone         Low fat    Not organic
14 ## 6 Chocolate  Homemade waffle No low fat Not organic
15 ## 7 Vanilla    Pint         Low fat    Not organic
16 ## 8 Mango      Homemade waffle Low fat    Organic
17 ## 9 Mango      Pint         No low fat Not organic
18 ## 10 Vanilla   Homemade waffle No low fat Organic

```

Em seguida, pedimos à função `predict` para prever as classificações dos perfis com base na função de regressão:

```

1 predict(conjoint_respondent1, profiles) # prever as classificacoes para os perfis com base
2   na analise conjunta
3
4 ## Conjoint Analysis
5 ## Data           : respondent1
6 ## Response variable : rating
7 ## Explanatory variables: Flavor, Packaging, Light, Organic
8 ## Prediction dataset : profiles
9 ##
10 ##   Flavor      Packaging      Light      Organic Prediction
11 ##   Raspberry  Homemade waffle No low fat Not organic    1.300
12 ##   Chocolate  Cone         No low fat Organic      4.200
13 ##   Raspberry  Pint         Low fat    Organic      1.700
14 ##   Strawberry Pint         No low fat Organic      7.200
15 ##   Strawberry Cone         Low fat    Not organic    8.800
16 ##   Chocolate  Homemade waffle No low fat Not organic    2.800
17 ##   Vanilla    Pint         Low fat    Not organic    5.300
18 ##   Mango      Homemade waffle Low fat    Organic     10.200
19 ##   Mango      Pint         No low fat Not organic    5.800
20 ##   Vanilla    Homemade waffle No low fat Organic      7.700

```

A classificação prevista é mais alta para sorvetes orgânicos com pouca gordura e manga em um waffle caseiro. Mas essas são previsões para sorvetes que o entrevistado realmente classificou. Se quiséssemos saber qual sorvete o entrevistado mais gostava, poderíamos apenas olhar para as classificações observadas (em vez das previstas). É mais interessante obter previsões para sorvetes que o entrevistado não avaliou. Para isso, precisamos dos perfis para todos os sorvetes possíveis. Podemos criar esses perfis com a função `expand.grid`.

A função `expand.grid` usa dois ou mais vetores e cria todas as combinações possíveis de elementos desses vetores:

```
1 Flavor <- c("Raspberry", "Chocolate", "Mango", "Strawberry", "Vanilla")
2 Organic <- c("Organic", "Not organic")
3
4 expand.grid(Flavor, Organic)
5 ##           Var1           Var2
6 ## 1   Raspberry   Organic
7 ## 2   Chocolate   Organic
8 ## 3     Mango     Organic
9 ## 4 Strawberry   Organic
10 ## 5    Vanilla   Organic
11 ## 6 Raspberry Not organic
12 ## 7   Chocolate Not organic
13 ## 8     Mango Not organic
14 ## 9 Strawberry Not organic
15 ## 10    Vanilla Not organic
```

Vamos fazer isso para todos os nossos níveis de atributo:

```
1 # existe uma maneira mais facil de obter niveis de atributo do que criar os vetores
  manualmente:
2
3 levels(icecream$Flavor) # certifique-se de que o sabor seja fatorado primeiro!
4
5 ## [1] "Chocolate" "Mango" "Raspberry" "Strawberry" "Vanilla"
6 # agora crie todos os perfis
7
8 profiles.all <- expand.grid(levels(icecream$Flavor), levels(icecream$Packaging), levels(
  icecream$Light), levels(icecream$Organic)) %>%
9   rename("Flavor" = "Var1", "Packaging" = "Var2", "Light" = "Var3", "Organic" = "Var4") #
  rename the variables created by expand.grid (don't forget this, otherwise predict won't
  know where to look for each attribute)
10
11 # prever as classifica es de todos os perfis
12
13 predict(conjoint_respondent1, profiles.all) %>%
14   arrange(desc(Prediction)) # mostrar os sorvetes com a classifica o mais alta prevista
  no topo
15
16 ##           Flavor           Packaging           Light           Organic Prediction
17 ## 1 Strawberry Homemade waffle Low fat Organic 10.2
18 ## 2 Strawberry Cone Low fat Organic 10.2
19 ## 3 Mango Homemade waffle Low fat Organic 10.2
20 ## 4 Mango Cone Low fat Organic 10.2
21 ## 5 Strawberry Homemade waffle No low fat Organic 9.2
22 ## 6 Strawberry Cone No low fat Organic 9.2
23 ## 7 Mango Homemade waffle No low fat Organic 9.2
24 ## 8 Mango Cone No low fat Organic 9.2
25 ## 9 Strawberry Homemade waffle Low fat Not organic 8.8
26 ## 10 Strawberry Cone Low fat Not organic 8.8
27 ## 11 Mango Homemade waffle Low fat Not organic 8.8
28 ## 12 Mango Cone Low fat Not organic 8.8
29 ## 13 Vanilla Homemade waffle Low fat Organic 8.7
30 ## 14 Vanilla Cone Low fat Organic 8.7
31 ## 15 Strawberry Pint Low fat Organic 8.2
32 ## 16 Mango Pint Low fat Organic 8.2
33 ## 17 Strawberry Homemade waffle No low fat Not organic 7.8
34 ## 18 Strawberry Cone No low fat Not organic 7.8
35 ## 19 Mango Homemade waffle No low fat Not organic 7.8
36 ## 20 Mango Cone No low fat Not organic 7.8
37 ## 21 Vanilla Homemade waffle No low fat Organic 7.7
38 ## 22 Vanilla Cone No low fat Organic 7.7
39 ## 23 Vanilla Homemade waffle Low fat Not organic 7.3
40 ## 24 Vanilla Cone Low fat Not organic 7.3
41 ## 25 Strawberry Pint No low fat Organic 7.2
42 ## 26 Mango Pint No low fat Organic 7.2
43 ## 27 Strawberry Pint Low fat Not organic 6.8
44 ## 28 Mango Pint Low fat Not organic 6.8
45 ## 29 Vanilla Pint Low fat Organic 6.7
46 ## 30 Vanilla Homemade waffle No low fat Not organic 6.3
47 ## 31 Vanilla Cone No low fat Not organic 6.3
48 ## 32 Strawberry Pint No low fat Not organic 5.8
49 ## 33 Mango Pint No low fat Not organic 5.8
50 ## 34 Vanilla Pint No low fat Organic 5.7
```



```

51 ## 35 Vanilla Pint Low fat Not organic 5.3
52 ## 36 Chocolate Homemade waffle Low fat Organic 5.2
53 ## 37 Chocolate Cone Low fat Organic 5.2
54 ## 38 Vanilla Pint No low fat Not organic 4.3
55 ## 39 Chocolate Homemade waffle No low fat Organic 4.2
56 ## 40 Chocolate Cone No low fat Organic 4.2
57 ## 41 Chocolate Homemade waffle Low fat Not organic 3.8
58 ## 42 Chocolate Cone Low fat Not organic 3.8
59 ## 43 Raspberry Homemade waffle Low fat Organic 3.7
60 ## 44 Raspberry Cone Low fat Organic 3.7
61 ## 45 Chocolate Pint Low fat Organic 3.2
62 ## 46 Chocolate Homemade waffle No low fat Not organic 2.8
63 ## 47 Chocolate Cone No low fat Not organic 2.8
64 ## 48 Raspberry Homemade waffle No low fat Organic 2.7
65 ## 49 Raspberry Cone No low fat Organic 2.7
66 ## 50 Raspberry Homemade waffle Low fat Not organic 2.3
67 ## 51 Raspberry Cone Low fat Not organic 2.3
68 ## 52 Chocolate Pint No low fat Organic 2.2
69 ## 53 Chocolate Pint Low fat Not organic 1.8
70 ## 54 Raspberry Pint Low fat Organic 1.7
71 ## 55 Raspberry Homemade waffle No low fat Not organic 1.3
72 ## 56 Raspberry Cone No low fat Not organic 1.3
73 ## 57 Chocolate Pint No low fat Not organic 0.8
74 ## 58 Raspberry Pint No low fat Organic 0.7
75 ## 59 Raspberry Pint Low fat Not organic 0.3
76 ## 60 Raspberry Pint No low fat Not organic -0.7

```

Mesma conclusão que a da seção anterior: essa pessoa desfrutaria de um sorvete com baixo teor de gordura, orgânico, manga ou morango em um cone ou um waffle caseiro.

9.4 Muitos respondentes

9.4.1 Estimar valores de peça e pesos de importância

Agora, vamos realizar a análise conjunta no conjunto de dados completo para ter uma idéia de quais sorvetes os 15 entrevistados, em média, gostaram mais e qual a importância de cada atributo:

```

1 conjoint_allrespondents <- conjoint(icecream, rvar = "rating", evar = c("Flavor","Packaging",
2   "","Light","Organic")) # como antes, mas com um conjunto de dados diferente.
3 summary(conjoint_allrespondents)
4
5 ## Conjoint analysis
6 ## Data : icecream
7 ## Response variable : rating
8 ## Explanatory variables: Flavor, Packaging, Light, Organic
9 ##
10 ## Conjoint part-worths:
11 ## Attributes Levels PW
12 ## Flavor Chocolate 0.000
13 ## Flavor Mango 1.522
14 ## Flavor Raspberry 0.522
15 ## Flavor Strawberry 0.767
16 ## Flavor Vanilla 1.389
17 ## Packaging Cone 0.000
18 ## Packaging Homemade waffle -0.244
19 ## Packaging Pint -0.100
20 ## Light Low fat 0.000
21 ## Light No low fat 0.478
22 ## Organic Not organic 0.000
23 ## Organic Organic 0.307
24 ## Base utility ~ 4.358
25 ##
26 ## Conjoint importance weights:
27 ## Attributes IW
28 ## Flavor 0.597
29 ## Packaging 0.096
30 ## Light 0.187
31 ## Organic 0.120
32 ##
33 ## Conjoint regression results:
34 ##
35 ## coefficient
36 ## (Intercept) 4.358

```



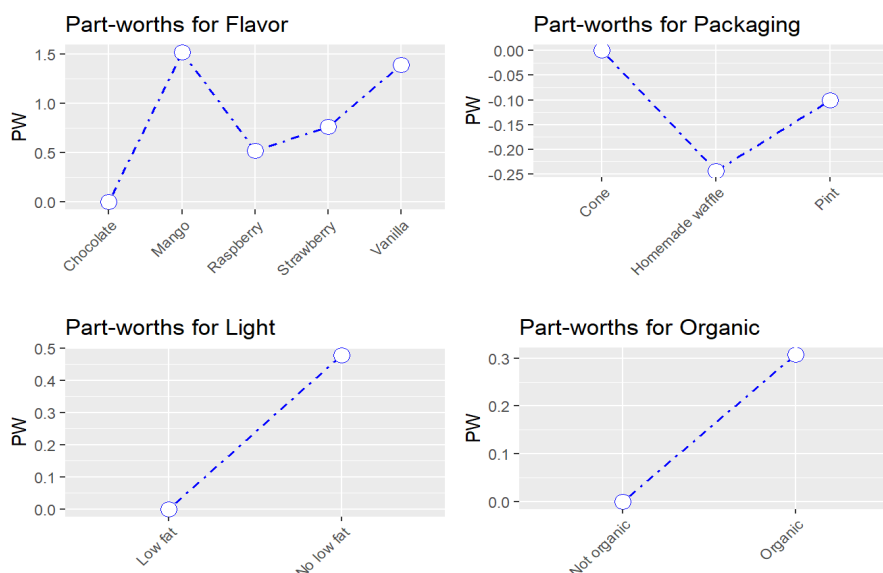
```

37 ## Flavor|Mango 1.522
38 ## Flavor|Raspberry 0.522
39 ## Flavor|Strawberry 0.767
40 ## Flavor|Vanilla 1.389
41 ## Packaging|Homemade waffle -0.244
42 ## Packaging|Pint -0.100
43 ## Light|No low fat 0.478
44 ## Organic|Organic 0.307

```

O sabor é de longe o atributo mais importante. Vamos traçar estes resultados:

```
1 plot(conjoint_allrespondents)
```



A partir disso, prevemos que, em média, as pessoas mais gostariam de um sorvete de manga orgânico, sem pouca gordura, em um cone.

Os pesos de importância nos dizem com que intensidade cada atributo determina a classificação média de um sorvete. O sabor é o atributo mais importante e a embalagem é o atributo menos importante. A classificação deste respondente é determinada para 59,7% por sabor e para 9,6% por embalagem.

9.4.2 Profiles: utilitários previstos

Vamos prever as classificações de todos os sorvetes possíveis:

```

1 predict(conjoint_allrespondents, profiles.all) %>% # verifique as secoes anteriores para
  profiles.all
2
3 arrange(desc(Prediction)) # mostrar os sorvetes com a classificacao mais alta prevista no
  topo
4
5 ## Flavor Packaging Light Organic Prediction
6 ## 1 Mango Cone No low fat Organic 6.664444
7 ## 2 Mango Pint No low fat Organic 6.564444
8 ## 3 Vanilla Cone No low fat Organic 6.531111
9 ## 4 Vanilla Pint No low fat Organic 6.431111
10 ## 5 Mango Homemade waffle No low fat Organic 6.420000
11 ## 6 Mango Cone No low fat Not organic 6.357778
12 ## 7 Vanilla Homemade waffle No low fat Organic 6.286667
13 ## 8 Mango Pint No low fat Not organic 6.257778
14 ## 9 Vanilla Cone No low fat Not organic 6.224444
15 ## 10 Mango Cone Low fat Organic 6.186667
16 ## 11 Vanilla Pint No low fat Not organic 6.124444
17 ## 12 Mango Homemade waffle No low fat Not organic 6.113333
18 ## 13 Mango Pint Low fat Organic 6.086667
19 ## 14 Vanilla Cone Low fat Organic 6.053333
20 ## 15 Vanilla Homemade waffle No low fat Not organic 5.980000
21 ## 16 Vanilla Pint Low fat Organic 5.953333
22 ## 17 Mango Homemade waffle Low fat Organic 5.942222
23 ## 18 Strawberry Cone No low fat Organic 5.908889

```

```

24 ## 19      Mango      Cone      Low fat Not organic 5.880000
25 ## 20      Vanilla  Homemade waffle      Low fat      Organic 5.808889
26 ## 21 Strawberry      Pint      No low fat      Organic 5.808889
27 ## 22      Mango      Pint      Low fat Not organic 5.780000
28 ## 23      Vanilla      Cone      Low fat Not organic 5.746667
29 ## 24 Raspberry      Cone      No low fat      Organic 5.664444
30 ## 25 Strawberry  Homemade waffle      No low fat      Organic 5.664444
31 ## 26      Vanilla      Pint      Low fat Not organic 5.646667
32 ## 27      Mango  Homemade waffle      Low fat Not organic 5.635556
33 ## 28 Strawberry      Cone      No low fat Not organic 5.602222
34 ## 29 Raspberry      Pint      No low fat      Organic 5.564444
35 ## 30      Vanilla  Homemade waffle      Low fat Not organic 5.502222
36 ## 31 Strawberry      Pint      No low fat Not organic 5.502222
37 ## 32 Strawberry      Cone      Low fat      Organic 5.431111
38 ## 33 Raspberry  Homemade waffle      No low fat      Organic 5.420000
39 ## 34 Raspberry      Cone      No low fat Not organic 5.357778
40 ## 35 Strawberry  Homemade waffle      No low fat Not organic 5.357778
41 ## 36 Strawberry      Pint      Low fat      Organic 5.331111
42 ## 37 Raspberry      Pint      No low fat Not organic 5.257778
43 ## 38 Raspberry      Cone      Low fat      Organic 5.186667
44 ## 39 Strawberry  Homemade waffle      Low fat      Organic 5.186667
45 ## 40 Chocolate      Cone      No low fat      Organic 5.142222
46 ## 41 Strawberry      Cone      Low fat Not organic 5.124444
47 ## 42 Raspberry  Homemade waffle      No low fat Not organic 5.113333
48 ## 43 Raspberry      Pint      Low fat      Organic 5.086667
49 ## 44 Chocolate      Pint      No low fat      Organic 5.042222
50 ## 45 Strawberry      Pint      Low fat Not organic 5.024444
51 ## 46 Raspberry  Homemade waffle      Low fat      Organic 4.942222
52 ## 47 Chocolate  Homemade waffle      No low fat      Organic 4.897778
53 ## 48 Raspberry      Cone      Low fat Not organic 4.880000
54 ## 49 Strawberry  Homemade waffle      Low fat Not organic 4.880000
55 ## 50 Chocolate      Cone      No low fat Not organic 4.835556
56 ## 51 Raspberry      Pint      Low fat Not organic 4.780000
57 ## 52 Chocolate      Pint      No low fat Not organic 4.735556
58 ## 53 Chocolate      Cone      Low fat      Organic 4.664444
59 ## 54 Raspberry  Homemade waffle      Low fat Not organic 4.635556
60 ## 55 Chocolate  Homemade waffle      No low fat Not organic 4.591111
61 ## 56 Chocolate      Pint      Low fat      Organic 4.564444
62 ## 57 Chocolate  Homemade waffle      Low fat      Organic 4.420000
63 ## 58 Chocolate      Cone      Low fat Not organic 4.357778
64 ## 59 Chocolate      Pint      Low fat Not organic 4.257778
65 ## 60 Chocolate  Homemade waffle      Low fat Not organic 4.113333

```

Mesmas conclusões de antes: prevemos que, em média, as pessoas mais gostariam de um sorvete de manga orgânico, sem pouca gordura, em um cone.

9.5 Simulação de Mercado

Digamos que criamos um pequeno número de sorvetes e queremos estimar a participação de mercado de cada um desses sorvetes. Digamos que selecionamos os quatro perfis a seguir:

```

1 # use slice() para selecionar as linhas
2
3 market_profiles <- profiles.all %>%
4   slice(c(4, 16, 23, 38)) # de profiles.all, selecione as linhas 4, 16, 23, 38 como quatro
5   profiles
6
7 market_profiles
8
9 ##      Flavor      Packaging      Light      Organic
10 ## 1 Strawberry      Cone      Low fat Not organic
11 ## 2 Chocolate      Cone      No low fat Not organic
12 ## 3 Raspberry  Homemade waffle      No low fat Not organic
13 ## 4 Raspberry  Homemade waffle      Low fat      Organic
14
15 # J sabemos como estimar qual sorvete ser mais apreciado:
16
17 conjoint_allrespondents <- conjoint(icecream, rvar = "rating", evar = c("Flavor", "Packaging",
18   "Light", "Organic"))
19
20 predict(conjoint_allrespondents, market_profiles) %>%
21   arrange(desc(Prediction))
22
23 ##      Flavor      Packaging      Light      Organic Prediction

```

```

22 ## 1 Strawberry Cone Low fat Not organic 5.124444
23 ## 2 Raspberry Homemade waffle No low fat Not organic 5.113333
24 ## 3 Raspberry Homemade waffle Low fat Organic 4.942222
25 ## 4 Chocolate Cone No low fat Not organic 4.835556

```

O sorvete de morango com baixo teor de gordura e não orgânico em um cone tem a classificação mais alta prevista entre todos os entrevistados. Mas isso não nos diz qual será a participação de mercado de cada um dos quatro perfis. Para isso, precisamos saber, para cada participante, qual perfil ele escolheria. Em outras palavras, precisamos prever as classificações para cada indivíduo separadamente:

```

1 # mesmo modelo de antes, mas agora adicione por = "respondent"
2 conjoint_perrespondent <- conjoint(icecream, rvar = "rating", evar = c("Flavor", "Packaging",
3   , "Light", "Organic"), by = "respondent")
4
5 predict(conjoint_perrespondent, market_profiles) %>%
6   arrange(respondent, desc(Prediction)) # classificar por respondente e depois por
7     classificacao prevista
8
9 ##      respondent      Flavor      Packaging      Light      Organic
10 ## 1 Individual 1 Strawberry Cone Low fat Not organic
11 ## 2 Individual 1 Raspberry Homemade waffle Low fat Organic
12 ## 3 Individual 1 Chocolate Cone No low fat Not organic
13 ## 4 Individual 1 Raspberry Homemade waffle No low fat Not organic
14 ## 5 Individual 10 Raspberry Homemade waffle No low fat Not organic
15 ## 6 Individual 10 Raspberry Homemade waffle Low fat Organic
16 ## 7 Individual 10 Chocolate Cone No low fat Not organic
17 ## 8 Individual 10 Strawberry Cone Low fat Not organic
18 ## 9 Individual 11 Strawberry Cone Low fat Not organic
19 ## 10 Individual 11 Raspberry Homemade waffle Low fat Organic
20 ## 11 Individual 11 Chocolate Cone No low fat Not organic
21 ## 12 Individual 11 Raspberry Homemade waffle No low fat Not organic
22 ## 13 Individual 12 Raspberry Homemade waffle No low fat Not organic
23 ## 14 Individual 12 Raspberry Homemade waffle Low fat Organic
24 ## 15 Individual 12 Chocolate Cone No low fat Not organic
25 ## 16 Individual 12 Strawberry Cone Low fat Not organic
26 ## 17 Individual 13 Strawberry Cone Low fat Not organic
27 ## 18 Individual 13 Raspberry Homemade waffle Low fat Organic
28 ## 19 Individual 13 Chocolate Cone No low fat Not organic
29 ## 20 Individual 13 Strawberry Cone Low fat Not organic
30 ## 21 Individual 14 Raspberry Homemade waffle No low fat Not organic
31 ## 22 Individual 14 Raspberry Homemade waffle Low fat Organic
32 ## 23 Individual 14 Strawberry Cone Low fat Not organic
33 ## 24 Individual 14 Chocolate Cone No low fat Not organic
34 ## 25 Individual 15 Strawberry Cone Low fat Not organic
35 ## 26 Individual 15 Chocolate Cone No low fat Not organic
36 ## 27 Individual 15 Raspberry Homemade waffle No low fat Not organic
37 ## 28 Individual 15 Raspberry Homemade waffle Low fat Organic
38 ## 29 Individual 2 Strawberry Cone Low fat Not organic
39 ## 30 Individual 2 Chocolate Cone No low fat Not organic
40 ## 31 Individual 2 Raspberry Homemade waffle Low fat Organic
41 ## 32 Individual 2 Raspberry Homemade waffle No low fat Not organic
42 ## 33 Individual 3 Chocolate Cone No low fat Not organic
43 ## 34 Individual 3 Raspberry Homemade waffle Low fat Organic
44 ## 35 Individual 3 Raspberry Homemade waffle No low fat Not organic
45 ## 36 Individual 3 Strawberry Cone Low fat Not organic
46 ## 37 Individual 4 Raspberry Homemade waffle Low fat Organic
47 ## 38 Individual 4 Chocolate Cone No low fat Not organic
48 ## 39 Individual 4 Strawberry Cone Low fat Not organic
49 ## 40 Individual 4 Raspberry Homemade waffle No low fat Not organic
50 ## 41 Individual 5 Strawberry Cone Low fat Not organic
51 ## 42 Individual 5 Chocolate Cone No low fat Not organic
52 ## 43 Individual 5 Raspberry Homemade waffle No low fat Not organic
53 ## 44 Individual 5 Raspberry Homemade waffle Low fat Organic
54 ## 45 Individual 6 Raspberry Homemade waffle Low fat Organic
55 ## 46 Individual 6 Raspberry Homemade waffle No low fat Not organic
56 ## 47 Individual 6 Chocolate Cone No low fat Not organic
57 ## 48 Individual 6 Strawberry Cone Low fat Not organic
58 ## 49 Individual 7 Strawberry Cone Low fat Not organic
59 ## 50 Individual 7 Raspberry Homemade waffle No low fat Not organic
60 ## 51 Individual 7 Raspberry Homemade waffle Low fat Organic
61 ## 52 Individual 7 Chocolate Cone No low fat Not organic
62 ## 53 Individual 8 Chocolate Cone No low fat Not organic
63 ## 54 Individual 8 Strawberry Cone Low fat Not organic
64 ## 55 Individual 8 Raspberry Homemade waffle No low fat Not organic
65 ## 56 Individual 8 Raspberry Homemade waffle Low fat Organic

```

```

64 ## 57 Individual 9 Strawberry Cone Low fat Not organic
65 ## 58 Individual 9 Chocolate Cone No low fat Not organic
66 ## 59 Individual 9 Raspberry Homemade waffle Low fat Organic
67 ## 60 Individual 9 Raspberry Homemade waffle No low fat Not organic
68 ## Prediction
69 ## 1 8.8000000
70 ## 2 3.7000000
71 ## 3 2.8000000
72 ## 4 1.3000000
73 ## 5 9.9500000
74 ## 6 8.9666667
75 ## 7 5.3666667
76 ## 8 2.0333333
77 ## 9 5.8000000
78 ## 10 5.2000000
79 ## 11 3.8000000
80 ## 12 2.8000000
81 ## 13 9.6000000
82 ## 14 7.7333333
83 ## 15 5.9333333
84 ## 16 3.2666667
85 ## 17 6.5500000
86 ## 18 6.2000000
87 ## 19 5.3000000
88 ## 20 1.3000000
89 ## 21 9.8000000
90 ## 22 6.8666667
91 ## 23 5.1333333
92 ## 24 3.4666667
93 ## 25 8.5333333
94 ## 26 6.8666667
95 ## 27 5.7000000
96 ## 28 1.9666667
97 ## 29 9.6333333
98 ## 30 5.9666667
99 ## 31 4.8666667
100 ## 32 3.5500000
101 ## 33 5.5666667
102 ## 34 4.7666667
103 ## 35 4.6500000
104 ## 36 2.2333333
105 ## 37 4.2000000
106 ## 38 3.3000000
107 ## 39 2.3000000
108 ## 40 2.0500000
109 ## 41 4.9333333
110 ## 42 4.2666667
111 ## 43 2.1000000
112 ## 44 2.0666667
113 ## 45 9.4000000
114 ## 46 7.6000000
115 ## 47 2.6000000
116 ## 48 1.6000000
117 ## 49 9.1666667
118 ## 50 6.7500000
119 ## 51 5.3333333
120 ## 52 4.8333333
121 ## 53 4.9333333
122 ## 54 4.2666667
123 ## 55 3.1000000
124 ## 56 0.7333333
125 ## 57 7.8666667
126 ## 58 7.5333333
127 ## 59 2.1333333
128 ## 60 1.2000000

```

Vamos reter para cada indivíduo apenas seu perfil mais bem classificado. Podemos fazer isso agrupando por entrevistado e adicionando uma variável denominada ranking que nos dirá o ranking de perfis, com base na classificação prevista, para cada entrevistado:

```

1 highest_rated <- predict(conjoint_perrespondent, market_profiles) %>%
2   group_by(respondent) %>%
3   mutate(ranking = rank(Prediction))
4
5 # dando uma olhada

```

```

6 highest_rated %>%
7   arrange(respondent, ranking)
8
9 ## # A tibble: 60 x 7
10 ## # Groups:   respondent [15]
11 ##   respondent Flavor Packaging Light Organic Prediction ranking
12 ##   <chr> <fct> <fct> <fct> <fct> <dbl> <dbl>
13 ## 1 Individual 1 Raspber~ Homemade wa~ No low~ Not orga~ 1.3 1
14 ## 2 Individual 1 Chocola~ Cone No low~ Not orga~ 2.80 2
15 ## 3 Individual 1 Raspber~ Homemade wa~ Low fat Organic 3.7 3
16 ## 4 Individual 1 Strawbe~ Cone Low fat Not orga~ 8.8 4
17 ## 5 Individual ~ Strawbe~ Cone Low fat Not orga~ 2.03 1
18 ## 6 Individual ~ Chocola~ Cone No low~ Not orga~ 5.37 2
19 ## 7 Individual ~ Raspber~ Homemade wa~ Low fat Organic 8.97 3
20 ## 8 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 9.95 4
21 ## 9 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 2.8 1
22 ## 10 Individual ~ Chocola~ Cone No low~ Not orga~ 3.80 2
23 ## # ... with 50 more rows
24 # precisamos reter apenas o sorvete mais bem classificado
25
26 highest_rated <- highest_rated %>%
27   arrange(respondent, ranking) %>%
28   filter(ranking == 4)
29
30 highest_rated
31
32 ## # A tibble: 15 x 7
33 ## # Groups:   respondent [15]
34 ##   respondent Flavor Packaging Light Organic Prediction ranking
35 ##   <chr> <fct> <fct> <fct> <fct> <dbl> <dbl>
36 ## 1 Individual 1 Strawbe~ Cone Low fat Not orga~ 8.8 4
37 ## 2 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 9.95 4
38 ## 3 Individual ~ Strawbe~ Cone Low fat Not orga~ 5.80 4
39 ## 4 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 9.60 4
40 ## 5 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 6.55 4
41 ## 6 Individual ~ Raspber~ Homemade wa~ No low~ Not orga~ 9.8 4
42 ## 7 Individual ~ Strawbe~ Cone Low fat Not orga~ 8.53 4
43 ## 8 Individual 2 Strawbe~ Cone Low fat Not orga~ 9.63 4
44 ## 9 Individual 3 Chocola~ Cone No low~ Not orga~ 5.57 4
45 ## 10 Individual 4 Raspber~ Homemade wa~ Low fat Organic 4.2 4
46 ## 11 Individual 5 Strawbe~ Cone Low fat Not orga~ 4.93 4
47 ## 12 Individual 6 Raspber~ Homemade wa~ Low fat Organic 9.40 4
48 ## 13 Individual 7 Strawbe~ Cone Low fat Not orga~ 9.17 4
49 ## 14 Individual 8 Chocola~ Cone No low~ Not orga~ 4.93 4
50 ## 15 Individual 9 Strawbe~ Cone Low fat Not orga~ 7.87 4

```

Agora podemos estimar a participação de mercado:



```

1 market_share <- highest_rated %>%
2   group_by(Flavor, Packaging, Light, Organic) %>%
3   summarize(count = n()) %>%
4   arrange(desc(count))
5
6 market_share
7
8 ## # A tibble: 4 x 5
9 ## # Groups:   Flavor, Packaging, Light [4]
10 ##   Flavor Packaging Light Organic count
11 ##   <fct> <fct> <fct> <fct> <int>
12 ## 1 Strawberry Cone Low fat Not organic 7
13 ## 2 Raspberry Homemade waffle No low fat Not organic 4
14 ## 3 Chocolate Cone No low fat Not organic 2
15 ## 4 Raspberry Homemade waffle Low fat Organic 2

```

Vimos que o sorvete de morango, cone, baixo teor de gordura e não orgânico é preferido por 7 em cada 15 participantes, o framboesa, waffle caseiro, sem baixo teor de gordura e sorvete não orgânico é favorecido por 4 em cada 15 participantes e assim por diante .

Referências

- [1]  for Marketing Students. Disponível em: <https://bookdown.org/content/1340/>
- [2] Overleaf, online \LaTeX editor. Disponível em [Overleaf.com](https://overleaf.com)
- [3] Xie, Y. **Dynamic Documents with  and knitr** 2nd edition, 2015.
- [4] **Reproducible Research using and Overleaf**. Disponível em [Reproducible Research using RMarkdown and Overleaf](#)