Manipulação de dados em R

Leonardo Sangali Barone

March 27, 2017

Manipulação de dados com a gramática básica do R

A esta altura do campeonato, você já tem bastante recursos para programar em R. Combinando seu conhecimento sobre vetores, *data frames*, tipos de dados, loops, condicionais e funções dá para fazer um bocado de coisas legais. Depois deste tutorial, voltaremos a esteś tópicos para exercitar o que aprendemos.

Falta, porém, algo essencial e muito próprio da linguagem R: como manipular variáveis e observações em um *data frame*. Boa parte da tarefa de organização de dados para a pesquisa se resume ao que faremos neste tópico. Este é o uso da lingaugem R que se aproxima do uso de outras ferramantas como SPSS, Stata e SAS.

No tutorial anterior, sobre abertura de dados, mencionei que há "gramáticas" diferentes para a manipulação de dados em R. Nosso objetivo é aprender bem a "gramática" do *tidyverse* (que por si só seria suficiente) e vamos ignorar a do pacote *data.table* (por pura economia de tempo, pois ela pode ser bastante útil).

A "gramática" básica do R é pouco elegante e essa é uma das barreiras ao aprendizado da linguagem. Ela é bem mais confusa e "verbosa" (ou seja, tem que escrever muito para realizar pouco) do que as dos demais softwares de análise de dados e do *tidyverse*. Mas sem conhecer como funciona a "gramática" básica da linguagem R, nossa capacidade de aprender mais no futuro ficaria bastante limitada. Lembre-se que aprenderemos num futuro breve formas equivalentes de fazermos as mesmas coisas.

Variáveis e data frames

Para esta atividade, vamos trabalhar com um banco de dados falso que criei ("fake_data" – está em inglês, pois é reciclagem de outro curso que lecionei).

Vamos nos adaptar a abrir dados com funções dos pacotes *readr*, *data.table* e *haven*. Para este exercício, podemos usar *read_delim*:

```
library(readr)
url_fake_data <- "https://raw.githubusercontent.com/leobarone/FLS6397/master/data/fake_data.csv"
fake <- read_delim(url_fake_data, delim = ";", col_names = T)
```

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   educ = col_character(),
##   income = col_double(),
##   savings = col_double(),
##   marriage = col_character(),
##   kids = col_character(),
##   party = col_character(),
##   turnout = col_character(),
##   vote_history = col_integer(),
##   economy = col_character(),
##   incumbent = col_character(),
##   candidate = col_character()
## )
```

A descrição das variáveis do banco de dados está abaixo:

"Fakeland is a very stable democracy that helds presidential elections every 4 years. We are going to work with the fake dataset of Fakeland individual citizens that contains information about their basic fake characteristics and fake political opinions/positions. The variables that our fake dataset are:

- *age*: age
- *sex*: sex
- *educ*: educational level
- *income*: montly income measured in fake money (FM$)
- *savings*: total fake money (FM$) in savings account
- *marrige*: marriage status (yes = married)
- *kids*: number of children
- *party*: party affiliation
- *turnout*: intention to vote in the next election
- *vote_history*: numbers of presidential elections that turned out since 2002 elections
- *economy*: opinion about the national economy performance
- *incumbent*: opinion about the incumbent president performance
- *candidate*: candidate of preference"

Exercício

Utilize as funções que você já conhece – *head*, *dim*, *names*, *str*, etc – para conhecer os dados. Quantas linhas há no *data frame*? Quantas colunas? Como estão armazenadas cada variável (tipo de dados e classe dos vetores colunas)?

Data frame como conjunto de vetores

No primeiro tutorial construímos um *data frame* a partir de vetores de mesmo tamanho e "pareados", ou seja, com as posições das informações representando cada observação. Para trabalhar com variáveis do *data frame* como vetores usamos o símbolo "$" separando o nome do *data frame* da variável. Por exemplo, escrevemos fake$age para indicar a variável "age" no *data frame* "fake":

```
print(fake$age)
```

```
##  [1] 37 26 35 43 36 38 29 33 42 41 25 35 35 28 32 40 37 34 33 29 43 35 30
## [24] 29 32 38 35 37 32 23
```

Podemos fazer uma cópia do vetor "age" que não seja variável "fake"? Sim:

```
idade <- fake$age
print(idade)
```

```
##  [1] 37 26 35 43 36 38 29 33 42 41 25 35 35 28 32 40 37 34 33 29 43 35 30
## [24] 29 32 38 35 37 32 23
```

Porque não podemos simplesmente usar "age" e precisamos colocar o nome do *data frame* seguido de "$" para indicar o vetor do conjunto de dados? Por que podemos ter mais de um *data frame* no mesmo **workspace** com uma variável de nome "age". Pense no *data frame* + nome da variável como um endereço composto da variável no seu workspace que evita ambiguidade. Para quem está aconstumad@ a trabalhar com SPSS, Stata ou SAS, ter que indicar qual é o *data frame* ao qual a variável pertence parece estranho, mas faz todo sentido para o R.

Outros exemplos simples de como usar variáveis de um *data frame* em outras funções (algumas das quais veremos no futuro, mas você já pode ir se acostumando à linguagem).

Gráfico de distribuição de uma variável contínua:

```r
plot(density(fake$age), main = "Distribuição de Idade", xlab = "Idade")
```

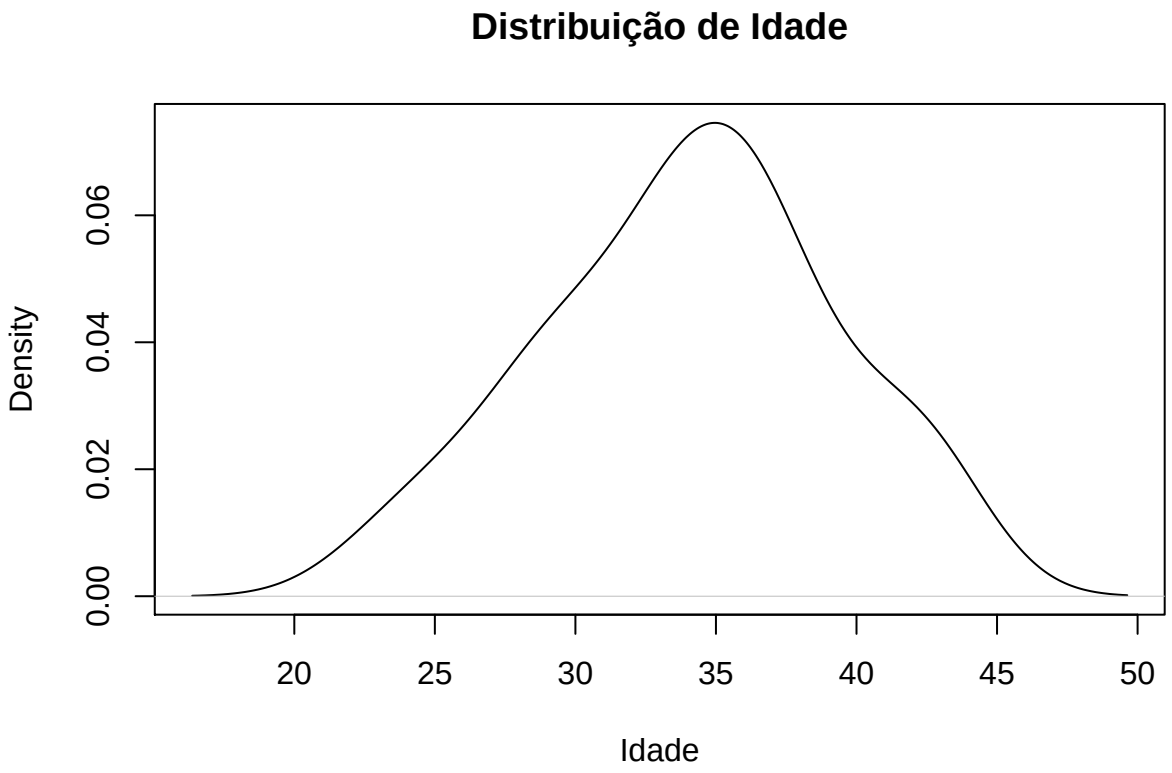

Gráfico de dispersão de duas variáveis contínuas:

```r
plot(fake$age, fake$savings, main = "Idade x Poupança", xlab = "Idade", ylab = "Poupança")
```

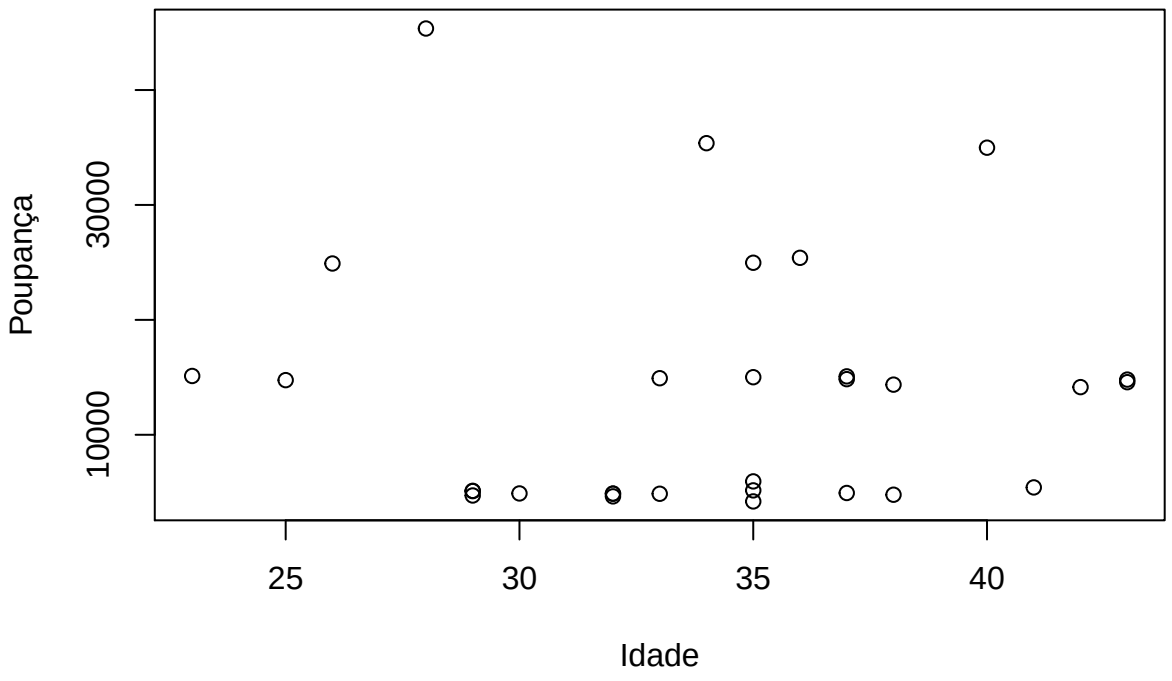

Tabela de uma variável categórica (contagem):

```
table(fake$party)
```

```
##
## Conservative Party        Independent    Socialist Party
##                  6                 15                  9
```

Tabela de duas entradas para duas variávels categóricas (contagem):

```
table(fake$party, fake$candidate)
```

```
##
##                       None Other Rilari Trampi
##   Conservative Party    0     0      3      3
##   Independent           1     4      5      5
##   Socialist Party       0     0      5      4
```

No começo pode parecer um pouco irritante usar o "endereço" completo da variável, mas você logo se acostuma.

Dimensões em um data frame

Tal como uma matriz, um *data frame* tem duas dimensões: linha e coluna. Se queremos selecionar elementos de um *data frame*, podemos usar colchetes separados por uma vírgula e inserir antes da vírgula uma seleção de linhas e depois da vírgula uma seleção de colunas – [linhas, colunas]. Vejamos alguns exemplos de seleção de linhas:

Quinta linha fazemos:

```
fake[5, ]
```

```
## # A tibble: 1 × 13
##     age    sex                   educ   income  savings marriage  kids
##   <int>  <chr>                  <chr>    <dbl>    <dbl>    <chr> <chr>
## 1    36 Female College Degree or more 4072.067 25398.73       No     1
## # ... with 6 more variables: party <chr>, turnout <chr>,
## #   vote_history <int>, economy <chr>, incumbent <chr>, candidate <chr>
```

Quinta e a oitava linhas:

```
fake[c(5,8), ]
```

```
## # A tibble: 2 × 13
##     age    sex                   educ   income  savings marriage  kids
##   <int>  <chr>                  <chr>    <dbl>    <dbl>    <chr> <chr>
## 1    36 Female College Degree or more 4072.067 25398.73       No     1
## 2    33   Male College Degree or more 2826.062 14922.29      Yes     0
## # ... with 6 more variables: party <chr>, turnout <chr>,
## #   vote_history <int>, economy <chr>, incumbent <chr>, candidate <chr>
```

As linhas 4 a 10:

```
fake[4:10,]
```

```
## # A tibble: 7 × 13
##     age    sex                   educ    income   savings marriage  kids
##   <int>  <chr>                  <chr>     <dbl>     <dbl>    <chr> <chr>
## 1    43   Male     High School Degree 4619.9733 14576.175       No     0
```

```
## 2    36 Female College Degree or more 4072.0674 25398.732       No     1
## 3    38   Male     College Incomplete 6463.0189 14363.654       No     0
## 4    29 Female     High School Degree 2219.0141  5118.181       No     1
## 5    33   Male College Degree or more 2826.0615 14922.285      Yes     0
## 6    42 Female     College Incomplete  448.0594 14147.852       No     0
## 7    41   Male     High School Degree 6540.1966  5419.379       No     0
## # ... with 6 more variables: party <chr>, turnout <chr>,
## #   vote_history <int>, economy <chr>, incumbent <chr>, candidate <chr>
```

Agora alguns exemplos de colunas. Segunda coluna:

```
fake[, 2]
```

```
## # A tibble: 30 × 1
##       sex
##     <chr>
## 1  Female
## 2  Female
## 3    Male
## 4    Male
## 5  Female
## 6    Male
## 7  Female
## 8    Male
## 9  Female
## 10   Male
## # ... with 20 more rows
```

Note que o resultado é semelhante ao de:

```
fake$sex
```

```
##  [1] "Female" "Female" "Male"   "Male"   "Female" "Male"   "Female"
##  [8] "Male"   "Female" "Male"   "Female" "Female" "Female" "Female"
## [15] "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Male"
## [22] "Male"   "Female" "Male"   "Female" "Male"   "Female" "Male"
## [29] "Female" "Male"
```

No entanto, no primeiro caso estamos produzindo um *data frame* de uma única coluna, enquanto no segundo estamos produzinho um vetor. Exceto pela classe, são idênticos.

Segunda e sétima colunas:

```
fake[, c(2,7)]
```

```
## # A tibble: 30 × 2
##       sex  kids
##     <chr> <chr>
## 1  Female     0
## 2  Female     0
## 3    Male     1
## 4    Male     0
## 5  Female     1
## 6    Male     0
## 7  Female     1
## 8    Male     0
## 9  Female     0
## 10   Male     0
```

```
## # ... with 20 more rows
```

Três primeiras colunas:

```
fake[, 1:3]
```

```
## # A tibble: 30 × 3
##      age    sex                   educ
##    <int>  <chr>                  <chr>
## 1     37 Female     College Incomplete
## 2     26 Female  No High School Degree
## 3     35   Male  No High School Degree
## 4     43   Male     High School Degree
## 5     36 Female College Degree or more
## 6     38   Male     College Incomplete
## 7     29 Female     High School Degree
## 8     33   Male College Degree or more
## 9     42 Female     College Incomplete
## 10    41   Male     High School Degree
## # ... with 20 more rows
```

Exercício

Qual é a idade do 17o. indivíduo? Qual é o candidato de preferência do 25o. indivíduo?

Seleção de colunas com nomes das variáveis

Neste *data frame* as linhas não têm nomes (mas poderiam ter). As colunas, no entanto, sempre têm. A regra é trabalharmos com muito mais linhas do que colunas e por esta razão os nomes das colunas costumam ser mais úteis do que os das linhas. Podemos usar os nomes das colunas no lugar de suas posições para selecioná-las.

```
fake[, c("age", "income", "party")]
```

```
## # A tibble: 30 × 3
##      age    income           party
##    <int>     <dbl>           <chr>
## 1     37 2595.2523     Independent
## 2     26 2166.7405     Independent
## 3     35 8213.8103 Socialist Party
## 4     43 4619.9733     Independent
## 5     36 4072.0674 Socialist Party
## 6     38 6463.0189     Independent
## 7     29 2219.0141     Independent
## 8     33 2826.0615     Independent
## 9     42  448.0594     Independent
## 10    41 6540.1966     Independent
## # ... with 20 more rows
```

Mas o código seguinte não é válido, pois o operador ":" serve somente para gerar sequências de números inteiros.

```
fake[, "age":"sex"]
```

```
## Warning in check_names_df(j, x): NAs introduced by coercion
```

```
## Warning in check_names_df(j, x): NAs introduced by coercion
```

```
## Error in "age":"sex": NA/NaN argument
```

Vamos super que acabamos de abrir os resultados eleitorais do Rio Grande do Sul nas eleições de 2016 retirados do Repositório de Dados Eleitorais do TSE (exatamente como faremos na atividade que segue este tutorial). Há um número grande de colunas desnecessárias a análise dos resultados (por exemplo, o ano da eleição, a hora da extração dos dados, etc). Para liberar memória do computador e trabalhar com um *data frame* menor, fazemos uma seleção de colunas exatamente como acima, seja usando sua posição ou seu nome e geramos um *data frame* novo (ou sobrescrevemos o atual). Veja um exemplo com "fake":

```
new_fake <- fake[, c("age", "income", "party", "candidate")]
```

E se quiseremos todas as colunas menos "turnout" e "vote_history"? Podemos usar a função *setdiff*, que gera a diferença entre dois vetores, por exemplo, o vetor com todos os nomes de colunas (gerado com a função *names*) e o vetor com as colunas que desejamos excluir. Vamos guardar o resultado em "new_fake2"

```
selecao_colunas <- setdiff(names(fake), c("turnout", "vote_history"))
print(selecao_colunas)
```

```
##  [1] "age"       "sex"       "educ"      "income"    "savings"
##  [6] "marriage"  "kids"      "party"     "economy"   "incumbent"
## [11] "candidate"
```

```
new_fake2 <- fake[,selecao_colunas]
```

Selecionando linhas com o operadores relacionais

No item acima fizemos uma seleção de colunas nos dados usando os nomes das colunas. Bancos de dados com muitas colunas, como os Censos Populacional e Escolar, ou o Latinobarômetro, não são tão comuns e raramente o número de colunas ultrapassa as poucas centenas.

O que fazer, então, com linhas, que são normalmente muito mais numerosas, as vezes na casa dos milhões? Precisamos utilizar operadores relacionais. Vamos fazer isso dando passos curtos para entendermos todo o processo.

Vamos supor que queremos selecionar apenas os indivíduos que pretendem votar na próxima eleição (variável "turnout"). Podemos gerar um vetor lógico que represente essa seleção:

```
fake$turnout == "Yes"
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [12]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [23] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

Vamos guardar esse vetor lógico em um objeto denominado "selecao_linhas"

```
selecao_linhas <- fake$turnout == "Yes"
print(selecao_linhas)
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [12]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## [23] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

Agora, podemos inserir esse vetor lógico na posição das linhas dentro dos colchetes para gerar um novo conjunto de dados que atenda à condição (intenção de votar):

```
fake_will_vote <- fake[selecao_linhas, ]
```

Basicamente, para fazermos uma seleção podemos usar a posição das linhas (ou das colunas), seus nomes ou um vetor lógico do mesmo tamanho das linhas (ou colunas). Sequer precisamos fazer o passo a passo acima. Veja um exemplo que gera um *data frame* de indivíduos que se identificam como "Independent":

```
fake_independents <- fake[fake$party == "Independent", ]
```

Podemos, obviamente, combinar condições e usar os operador lógicos ("ou", "e" e "não") para fazer seleções mais complexas:

```
fake_married_no_college_yong <- fake[fake$marriage == "Yes" &
                                     fake$age <= 30 &
                                     !(fake$educ == "College Degree or more"), ]
```

Exercício

Produza um novo *data frame* com apenas 4 variáveis – "age", "income", "economy" e "candidate" – e que contenha apenas eleitores homens, ricos ("income" maior que FM$ 3 mil, que é dinheiro pra caramba em Fakeland) e inclinados a votar no candidato "Trampi".

Quais as dimensões do novo *data frame*? Qual é a idade média dos eleitores no novo *data frame*? Qual é a soma da renda no novo *data frame*?

Função subset

Uma maneira alternativa de fazer a seleção de linhas é usar a função *subset*. Veja como (repetindo o exemplo de indivíduos identificados como "independentes"):

```
fake_independents <- subset(fake, party == "Independent")
```

O resultado é o mesmo e você pode achar essa maneira mais elegante. Veremos, no futuro, outra ainda mais simples com o pacote *dplyr*.

Criando uma nova coluna

Criar uma nova coluna em um *data frame* é trivial. Por exemplo, vamos criar a coluna "vazia", onde colocaremos apenas "missing values", que são representados no R por "NA"

```
fake$vazia <- NA
```

Podemos criar uma coluna a partir de outra(s). Por exemplo, vamos criar duas novas colunas, "poupança", que será igual a coluna "savings" mas em real (cotação de um FM$ – fake money – é R$ 17) e a coluna "savings_year", que será a divisão de "savings" por anos do indíviduo a partir dos 18.

```
fake$poupanca <- fake$savings / 17
fake$savings_year <- fake$savings / (fake$age - 18)
```

Você pode fazer qualquer operação com vetores que vimos em tutoriais anteriores para criar novas variáveis. A única imposição é que os vetores tenham sempre o mesmo tamanho, o que não é um problema em um *data frame*.

Se quiser substituir o conteúdo de uma variável em vez de gerar uma nova, o procedimento é o mesmo. Basta atribuir o resultado da operação entre vetores à variável existente, tal qual no exemplo, que transforma "age" em uma variável medida em meses:

```
fake$age <- fake$age  * 12
```

Vamos ver agora como substituir valores em uma variável para depois aprendermos a recodificarmos variáveis.

Substituindo valores em um variável

Vamos "traduzir" para o português a variável "party". Faremos isso alterando cada uma das categorias individualmente e, por enquanto, sem usar nenhuma função que auxilie a substituição de valores. Começemos com uma tabela simples da variável "party"

```
table(fake$party)
```

```
##
## Conservative Party        Independent    Socialist Party
##                  6                 15                  9
```

Agora, observe o resultado do código abaixo:

```
fake$party[fake$party == "Independent"]
```

```
##  [1] "Independent" "Independent" "Independent" "Independent" "Independent"
##  [6] "Independent" "Independent" "Independent" "Independent" "Independent"
## [11] "Independent" "Independent" "Independent" "Independent" "Independent"
```

Fizemos um subconjunto de apenas uma variável do *data frame*, e não do *data frame* todo. Note a ausência da vírgula dentro do colchetes, pois Se atribuirmos algo a essa selação, por exemplo, o texto "Independentes", substituiremos os valores da seleção:

```
fake$party[fake$party == "Independent"] <- "Independente"
```

Importante: a seleção do vetor (colchetes) está à direita do símbolo de atribuição.

Observe o resultado na tabela:

```
table(fake$party)
```

```
##
## Conservative Party       Independente    Socialist Party
##                  6                 15                  9
```

Exercício

Traduza para o português as demais categorias da variável "party".

Substituição com o comando replace

O procedimento de substituir valores em uma mesma variável pode ser alternativamente realizado com a função *replace*. Vamos traduzir para o português a variável "sex"

```
fake$sex <- replace(fake$sex, fake$sex == "Female", "Mulher")
fake$sex <- replace(fake$sex, fake$sex == "Male", "Homem")
table(fake$sex)
```

```
##
##  Homem Mulher
##     15     15
```

Recodificando uma variável

Vamos supor que não nos interessa trabalhar com renda ("income") como variável contínua. Vamos transformá-la na variável "rich", que receberá valor "rich" se a renda for maior que FM$ 3 mil e "not rich" caso contrário.

Seguiremos o seguinte procedimento: criaremos uma variável com "missing values"; substituiremos os valores para os indivíduos ricos; e depois substituiremos os valores para os não-ricos. Note que a seleção da variável "rich" para a substituição de valores é feita utilizando a variável "income":

```
fake$rich <- NA
fake$rich[fake$income > 3000] <- "rich"
fake$rich[fake$income <= 3000] <- "not rich"
table(fake$rich)
```

```
##
## not rich     rich
##       19       11
```

Exercício

Utilize o que você aprendeu sobre transformações de variáveis neste tutorial e o sobre fatores ("factors") no tutorial 2 para transformar a variável "rich" em fatores.

Exercício (mais um)

Crie a variável "kids2" que indica se o indivíduo tem algum filho (TRUE) ou nenhum (FALSE). Dica: essa é uma variável de texto, e não numérica.

Recodificando uma variável contínua com a função cut

Quando vamos recodificar uma variável contínua, podemos usar a função *cut*. Vamos repetir o exemplo da criação da variável "rich", agora com "rich2":

```
fake$rich2 <- cut(fake$income,
                  breaks = c(-Inf, 3000, Inf),
                  labels = c("não rico", "rico"))
table(fake$rich2)
```

```
##
## não rico     rico
##       19       11
```

Algumas observações importantes: se a nova variável tiver 2 categorias, precisamos de 3 "break points"; "-Inf" e "Inf" são os símbolos do R para menos e mais infinito, respectivamente; por padrão, o R nao inclui o primeiro "break point" na primeira categoria, exceto se o argumento "include.lowest" for alterado para TRUE; também por padrão, os intervalos são fechados à direita, e abertos à esquerda (ou seja, incluem o valor superior que delimita o intervalor, mas não o inferior), exceto se "right" o argumento for alterado para FALSE.

Exercício

Crie a variável "poupador", gerada a partir de avings_year (que criamos anteriormente, antes de transformar "age" em meses), e que separa os indivíduos que poupam muito (mais de FM/$ 1000 por ano) dos que poupam pouco. Use a função *cut*.

Recodificando uma variável contínua com a função recode

O equivalente da função *cut* para variáveis categóricas, estejam elas como texto ou como fatores é a função *recode*, do pacote *dplyr*. Seu uso é simples e intuitivo. Vamos recodificar como exemplo a variável "educ":

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
fake$college <- recode(fake$educ,
                       "No High School Degree" = "No College",
                       "High School Degree" = "No College",
                       "College Incomplete" = "No College",
                       "College Degree or more" = "College")
table(fake$college)
```

```
##
##    College No College
##          7         23
```

Podemos comparar as mudanças com uma tabela de 2 entradas:

```
table(fake$college, fake$educ)
```

```
##
##              College Degree or more College Incomplete High School Degree
##   College                         7                  0                  0
##   No College                      0                  6                 14
##
##              No High School Degree
##   College                        0
##   No College                     3
```

Exercício

Crie a variável "economia", que os indivíduos que avaliam a economia (variável "economy") como "Good" ou melhor recebem o valor "positivo" e os demais recebem "negativo".

Ordenar linhas e remover linhas duplicadas:

Finalmente, vamos ordenar linhas em um banco de dados e aprender a remover duplicidades. Com a função *order*, podemos gerar um vetor que indica qual a posição que cada linha deveria receber no ordenamento desejado. Vamos ordernar "fake" por renda.

```
ordem <- order(fake$income)
print(ordem)
```

```
##  [1] 12 13  9 28 15 20 25 11 29  2 19  7 21  1 16 30  8 26 22 24 27  5 17
## [24] 18  4 23  6 10 14  3
```

Se aplicarmos um vetor numérico com um novo ordenaemnto à parte destinada às linhas no colchetes, receberemos o *data frame* ordenado:

```
fake_ordenado <- fake[ordem, ]
head(fake_ordenado)
```

```
## # A tibble: 6 × 19
##     age    sex              educ    income   savings marriage  kids
##   <dbl>  <chr>             <chr>     <dbl>     <dbl>    <chr> <chr>
## 1   420 Mulher High School Degree  43.69933  4203.021      Yes     0
## 2   420 Mulher High School Degree 266.92899  5162.188       No     0
## 3   504 Mulher College Incomplete 448.05938 14147.852       No     0
## 4   444  Homem High School Degree 508.59934  4935.762      Yes     2
## 5   384 Mulher High School Degree 524.24489  4639.687       No     0
## 6   348 Mulher High School Degree 652.70715  4718.129       No     2
## # ... with 12 more variables: party <chr>, turnout <chr>,
## #   vote_history <int>, economy <chr>, incumbent <chr>, candidate <chr>,
## #   vazia <lgl>, poupanca <dbl>, savings_year <dbl>, rich <chr>,
## #   rich2 <fctr>, college <chr>
```

Poderíamos ter aplicado a função order diretamente dentro dos colchetes:

```
fake_ordenado <- fake[order(fake$income), ]
```

Para encerrar, vamos duplicar articialmente parte dos nossos dados (as 10 primeiras linhas) usando o comando *rbind*, que "empilha" dois *data frames*:

```
fake_duplicado <- rbind(fake, fake[1:10, ])
```

Vamos ordenar para ver algumas duplicidades:

```
fake_duplicado[order(fake_duplicado$income), ]
```

```
## # A tibble: 40 × 19
##      age    sex              educ     income   savings marriage  kids
##    <dbl>  <chr>             <chr>      <dbl>     <dbl>    <chr> <chr>
## 1    420 Mulher High School Degree   43.69933  4203.021      Yes     0
## 2    420 Mulher High School Degree  266.92899  5162.188       No     0
## 3    504 Mulher College Incomplete  448.05938 14147.852       No     0
## 4    504 Mulher College Incomplete  448.05938 14147.852       No     0
## 5    444  Homem High School Degree  508.59934  4935.762      Yes     2
## 6    384 Mulher High School Degree  524.24489  4639.687       No     0
## 7    348 Mulher High School Degree  652.70715  4718.129       No     2
## 8    384 Mulher High School Degree  874.76081  4884.785      Yes     0
## 9    300 Mulher College Incomplete 1407.38431 14755.751      Yes     0
## 10   384 Mulher High School Degree 2108.53667  4872.189       No     0
## # ... with 30 more rows, and 12 more variables: party <chr>,
## #   turnout <chr>, vote_history <int>, economy <chr>, incumbent <chr>,
## #   candidate <chr>, vazia <lgl>, poupanca <dbl>, savings_year <dbl>,
## #   rich <chr>, rich2 <fctr>, college <chr>
```

E agora removemos da seguinte maneira:

```
fake_novo <- fake_duplicado[!duplicated(fake_duplicado),]
```

Note que precisamos da exclamação (operador lógico "não") para ficar com todas as linhas **não** duplicadas.

Item Adicional - Renomeando variáveis

Em breve veremos como renomear variáveis de uma maneira bastante mais simples. Por enquanto, vamos aprender o jeito trabalhoso de renomear um variável.

Podemos observar os nomes das variáveis de um *data frame* usando o comando *names*:

```
names(fake)
```

```
##  [1] "age"          "sex"          "educ"         "income"
##  [5] "savings"      "marriage"     "kids"         "party"
##  [9] "turnout"      "vote_history" "economy"      "incumbent"
## [13] "candidate"    "vazia"        "poupanca"     "savings_year"
## [17] "rich"         "rich2"        "college"
```

Os nomes das colunas são um vetor. Para renomear as variáveis, basta substituir o vetor de nomes por outro:

```
names(fake) <- c()
```