FIAP

MBA Data Science

MBA Data Science: R for Data Scientists

Perfil Profissional

Prof. Edmar Caldas

Acadêmico

- MBA em Engenharia Financeira – POLI / USP.
- Pós Graduação em Análise de dados e Data Mining - FIA.
- Graduado em Ciência da Computação e Estatística.

Professor de Modelagem Estatística, Data Mining e Machine Learning dos Cursos - MBA Big Data, Data Science e Business Intelligence da Faculdade de Informática e Administração Paulista - FIAP com foco em linguagem de programação R e Python.

Professor do curso MBA Esalq/USP – Gestão de Vendas.

Profissional

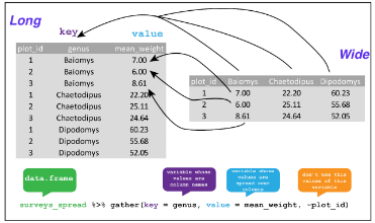
- CEO e consultor de negócios da Inteligência Analítica com foco em consultoria: Credit Scoring, Previsão de Vendas, Fraudes entre outas.
- Certificações: IBM SPSS Modeler e SPSS Statistics.

Objetivo da disciplina

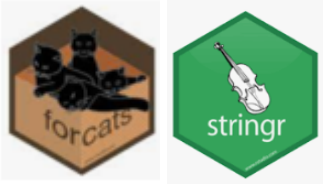
Ênfase no aprofundamento na linguagem R para Cientistas de Dados

Conteúdo da disciplina

- Exploratory Data Analysis



- Data transformation
- Tibbles
- TidyData



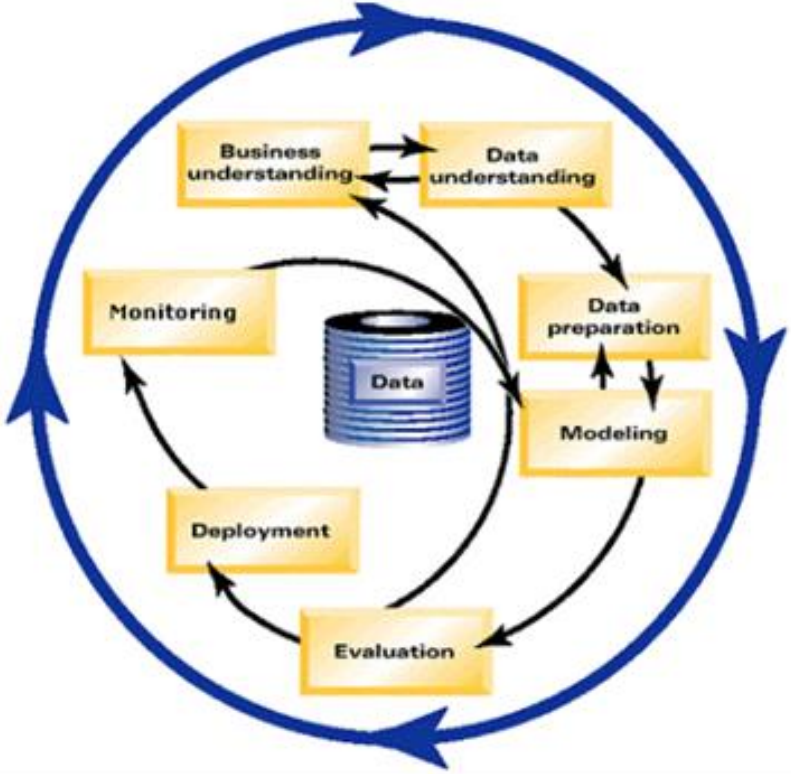
- Data transformation
- Stringr
- Factors

- Pipe
- Functions

R for Data Scientists

R para ciência de dados se concentra nos usos estatísticos e gráficos da linguagem. Ao aprender R para ciência de dados, você aprenderá a usar a linguagem para realizar análises estatísticas e desenvolver visualizações de dados. As funções estatísticas do R também facilitam a limpeza, importação e análise de dados.

Processo para Análise de Dados – Crisp - DM



Exploratory Data Analysis

DataExplorer

Background

Exploratory Data Analysis (EDA) is the initial and an important phase of data analysis/predictive modeling. During this process, analysts/modelers will have a first look of the data, and thus generate relevant hypotheses and decide next steps. However, the EDA process could be a hassle at times. This R package aims to automate most of data handling and visualization, so that users could focus on studying the data and extracting insights.

Exploratory Data Analysis

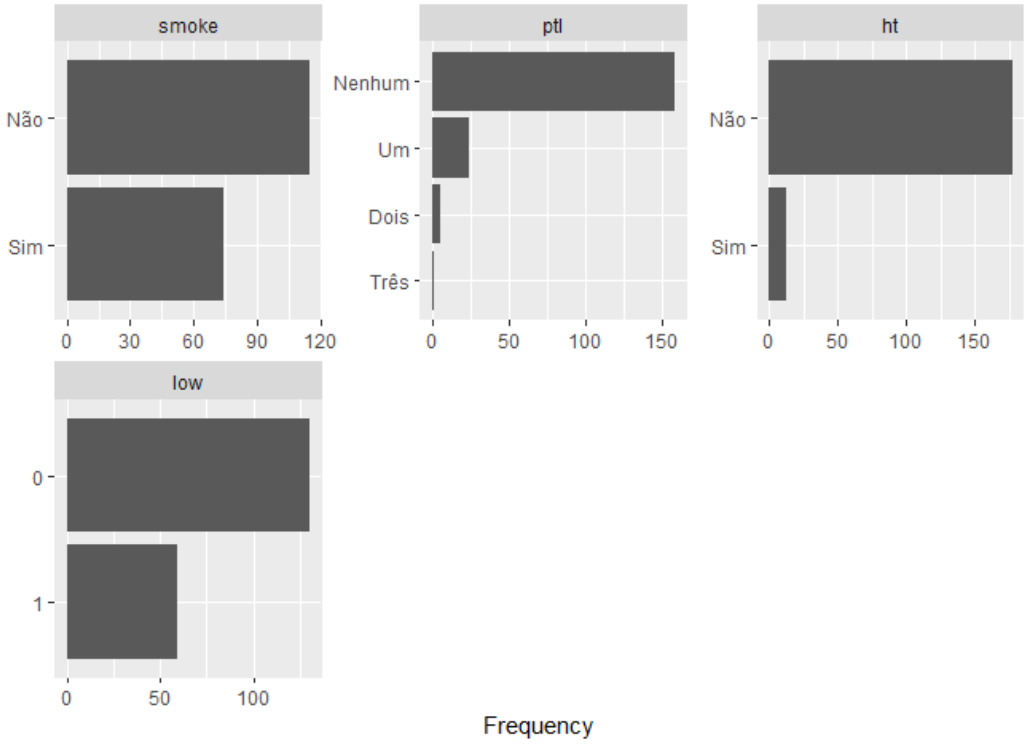
```
#install.packages('DataExplorer')
library(DataExplorer)

# Describe basic information
introduce(bebes)

# This function creates a data profiling report
create_report(bebes)
```

Data Profiling Report

- Basic Statistics
  - Raw Counts
  - Percentages
- Data Structure
- Missing Data Profile
- Univariate Distribution
  - Histogram
  - Bar Chart (with frequency)
  - QQ Plot
- Correlation Analysis
- Principal Component Analysis

Exploratory Data Analysis

```
# cria grafico no visualizador
plot_bar(bebes)
```



Exploratory Data Analysis

```
# cria uma matrix de correlacao
plot_correlation(bebes)

boxplot(bebes$age ~ bebes$low)
plot_boxplot(bebes, by = "low",geom_boxplot_args = list("outlier.color" = "red"))
plot_boxplot(bebes, by = "low", ncol = 2L)
plot_qq(bebes, by = "low")
table(bebes$low, bebes$smoke)
barplot(table(bebes$low))


#options(scipen=999)
#options(digits=2)
#plot_bar(bebes, with = "lwt")

# Plot missing value profile
plot_missing(bebes)

# sampled_rows number of rows to sample if data has too many rows. Default is all rows
plot_scatterplot(split_columns(bebes)$continuous, by = "lwt", sampled_rows = 189)

# Plot introduction
plot_intro(bebes)
```

Exploratory Data Analysis

cran/**SmartEDA**

Summarize and Explore the Data

R package that automates most of exploratory analyses tasks in modeling

The document introduces the **SmartEDA** package and how it can help you to build exploratory data analysis.

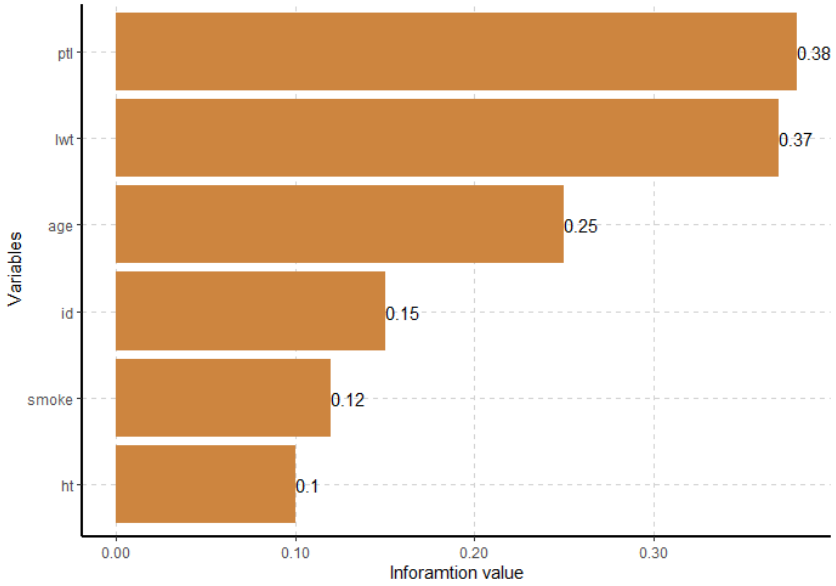
SmartEDA includes multiple custom functions to perform initial exploratory analysis on any input data describing the structure and the relationships present in the data. The generated output can be obtained in both summary and graphical form. The graphical form or charts can also be exported as reports.

https://cran.r-project.org/web/packages/SmartEDA/vignettes/SmartEDA.html

Exploratory Data Analysis

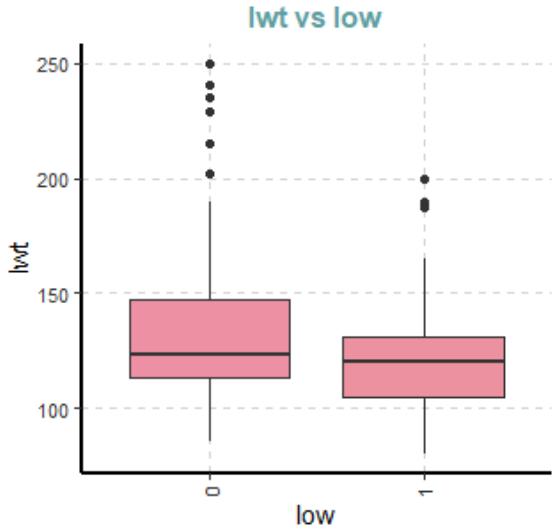
```
#install.packages('SmartEDA')
library(SmartEDA)

ExpData(data=bebes, type=1)
ExpData(data=bebes, type=2)
ExpCatStat(bebes, Target="low", Pclass="1", plot=T)
```

```
  ExpCatStat(bebes, Target="low", Pclass="1", plot=T)
  Variable Target Unique Chi-squared p-value df IV Value Cramers V
1    smoke    low      2         4.2   0.040  1     0.12       0.15
2      ptl    low      4        16.9   0.001  3     0.38       0.30
3       ht    low      2         3.1   0.076  1     0.10       0.13
4       id    low     10       180.7   0.000  9     0.15       0.98
5      age    low      9         7.4   0.388  7     0.25       0.22
6      lwt    low     10        15.0   0.092  9     0.37       0.28
  Degree of Association    Predictive Power
1                  Weak Somewhat Predictive
2                Strong   Highly Predictive
3                  Weak Somewhat Predictive
4                Strong Somewhat Predictive
5              Moderate   Medium Predictive
6              Moderate   Highly Predictive
```



Exploratory Data Analysis

```
# Resumo estatistico
ExpNumStat(bebes,by="A",Qnt=seq(0,1,0.1),MesofShape=2,Outlier=TRUE,round=2)
ExpNumStat(bebes,by="G",gp="low",Qnt=seq(0,1,0.1),MesofShape=2,Outlier=TRUE,round=2)

# Visualizacao
ExpNumViz(bebes,target="low",type=2,nlim=25,Page = c(2,2))
```



Exploratory Data Analysis

```
library(ggplot2)
# Cria tabela de frequencia
ExpCTable(bebes,Target=NULL,margin=1,clim=10,nlim=5,round=2,bin=NULL,per=T)

# Cria uma distribuicao por categoria
ExpCatViz(bebes,target="low",fname=NULL,clim=10,col=NULL,margin=2,Page = c(2,2))#,sample=2)

# Funcao de resumo
ExpCatStat(bebes,Target="low",result="Stat",Pclass="Yes",plot=TRUE,top=20,Round=2)

# Cria um HTML
ExpReport(bebes,Target="low",label=NULL,op_file="test.html",op_dir=getwd(),sc=2,sn=2,Rc="Yes")

# Cria grafico QQPLT
ExpOutQQ(bebes,nlim=10,fname=NULL,Page=c(3,1))#,sample=4)

#Identifying outliers mehtod - 3 Standard Deviation
ExpOutliers(bebes, varlist = c("lwt","low","age"), method = "3xStDev")

#Identifying outliers mehtod - 2 Standard Deviation
ExpOutliers(bebes, varlist = c("lwt","low","age"), method = "2xStDev")
```

Exploratory Data Analysis

```
# Customizacao resumo estatistico
ExpCustomStat(bebes,Cvar=c("smoke","ptl"),gpby=T)
ExpCustomStat(bebes,Cvar=c("smoke","ptl"),gpby=T,filt="age > 30")
ExpCustomStat(bebes,Cvar=c("smoke","ptl"),gpby=T,filt="low=='0' & age > 30")
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum','var','min','max'))
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('min','p0.25','median','p0.75','max'))
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum','var'),filt="smoke=='Não'")
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum'),filt="smoke=='Sim' & age > 30")
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum','min'),filt="All %ni% c(999,-9)")
ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum','var','sd','IQR','median'),
              filt=c("smoke=='Sim'^ age > 20  ^ lwt >= 100 ^ ht == 'Não'"))
```

```
  ExpCustomStat(bebes,Nvar=c("lwt","age"),stat = c('Count','mean','sum','var','sd','IQR','median'),
+               filt=c("smoke=='Sim'^ age > 20  ^ lwt >= 100 ^ ht == 'Não'"))
   Attribute       Filter Count mean  sum  var   sd IQR median
1:       lwt smoke=='Sim'    74  128 9482 1142 33.8  30    120
2:       age     age > 20   118   26 3095   18  4.2   6     25
```

Exploratory Data Analysis

cran/**desctable**

Stata command for descriptive statistics tables

desctable creates formatted descriptive statistics tables using Stata. The table is automatically exported to Excel, where it can be easily copied and pasted to Word without losing the formatting.

desctable treats continuous, binary, and nominal variables differently — providing formatting, labeling, and statistics that are most appropriate for the measurement level of each variable.

https://www.trentonmize.com/software/desctable

Exploratory Data Analysis

```
desctable(bebes)

bebes %>%
  desctable() %>%
  pander()

teste = bebes[,c('low','lwt','age','ht')]
teste %>%
  group_by(low)%>%
  desc_table() %>%
  desc_tests(lwt =  ~ t.test(., var.equal = T),
             age =  ~ t.test(., var.equal = F),
              ht =  ~ chisq.test)%>%
  desc_output("pander")

bebes %>%
  desctable(stats = stats_auto) %>%
  datatable()
```

```
desctable(bebes)
        Min  Q1    Med   Mean  Q3 Max    sd IQR
1    id    4  68    123 121.08 176 226 63.30 108
2   low    0   0      0   0.31   1   1  0.46   1
3   age   14  19     23  23.26  26  45  5.30   7
4   lwt   80 110    121 130.10 141 250 30.52  31
5 smoke  Não  NA    Não     NA  NA Sim    NA  NA
6   ptl Dois  NA Nenhum     NA  NA  Um    NA  NA
7    ht  Não  NA    Não     NA  NA Sim    NA  NA
```

Exploratory Data Analysis

```
bebes %>%
  group_by(low) %>%
  desctable() %>%
  pander()

bebes %>%
  group_by(lwt >=100) %>%
  desctable() %>%
  datatable()

bebes %>%
  dplyr::mutate(low = factor(low, labels = c("normal", "abaixo"))) %>%
  group_by(ht, low) %>%
  desctable() %>%
  datatable()

bebes %>%
  group_by(low) %>%
  desctable(tests = tests_auto) %>%
  datatable()
```

Copy Excel

low: 0 (n=130)

	Min	Q1	Med	Mean	Q3	Max	sd	IQR
id	85	119	154	156	192	226	42	72
age	14	19	23	24	28	45	5.6	9

Data transformation – package tidyr

Na computação, a transformação de dados é o processo de conversão de dados de um formato ou estrutura para outro formato ou estrutura. É um aspecto fundamental da maioria das tarefas de integração e gerenciamento de dados, como data wrangling, data warehousing, integração de dados e integração de aplicativos.

Data transformation – package tidyr

spread {tidyr} R Documentation

Spread a key-value pair across multiple columns

Description

lifecycle superseded

Development on `spread()` is complete, and for new code we recommend switching to `pivot_wider()`, which is easier to use, more featureful, and still under active development. `df %>% spread(key, value)` is equivalent to `df %>% pivot_wider(names_from = key, values_from = value)`

Data transformation – package tidyr

```
library(tidyr)
df <- data.frame(player=rep(c('A', 'B'), each=4),
                 year=rep(c(1, 1, 2, 2), times=2),
                 stat=rep(c('points', 'assists'), times=4),
                 amount=c(14, 6, 18, 7, 22, 9, 38, 4))

df
player year    stat amount
     A    1  points     14
     A    1 assists      6
     A    2  points     18
     A    2 assists      7
     B    1  points     22
     B    1 assists      9
     B    2  points     38
     B    2 assists      4


# Spread a key-value pair across multiple columns
spread(df, key=stat, value=amount)
player year assists points
     A    1       6     14
     A    2       7     18
     B    1       9     22
     B    2       4     38
```

Data transformation – package tidyr

```
# 2 exemplo
df2 <- data.frame(player=rep(c('A'), times=8),
                  year=rep(c(1, 2), each=4),
                  stat=rep(c('points', 'assists', 'steals', 'blocks'), times=2),
                  amount=c(14, 6, 2, 1, 29, 9, 3, 4))


df2
player year    stat amount
     A    1  points     14
     A    1 assists      6
     A    1  steals      2
     A    1  blocks      1
     A    2  points     29
     A    2 assists      9
     A    2  steals      3
     A    2  blocks      4

spread(df2, key=stat, value=amount)
player year assists blocks points steals
     A    1       6      1     14      2
     A    2       9      4     29      3
```

Data transformation – package tidyr

```
# exemplo 3
df3 <- data.frame(player_category=rep(c('A', 'B'), each=4),
                  experience_in_years=rep(c(1, 1, 2, 2), times=2),
                  statistics=rep(c('points', 'assists'), times=4),
                  amount=c(13, 7, 10, 3, 21, 8, 58, 2))

df3
player_category experience_in_years statistics amount
              A                    1     points     13
              A                    1    assists      7
              A                    2     points     10
              A                    2    assists      3
              B                    1     points     21
              B                    1    assists      8
              B                    2     points     58
              B                    2    assists      2


# Invoking spread() function
# spreading statistics column across multiple columns
spread(df3, key=statistics, value=amount)
player_category experience_in_years assists points
              A                    1       7     13
              A                    2       3     10
              B                    1       8     21
              B                    2       2     58
```

Data transformation – package tidyr

gather {tidyr} R Documentation

Gather columns into key-value pairs

Description

Development on `gather()` is complete, and for new code we recommend switching to `pivot_longer()`, which is easier to use, more featureful, and still under active development. `df %>% gather("key", "value", x, y, z)` is equivalent to `df %>% pivot_longer(c(x, y, z), names_to = "key", values_to = "value")`

Data transformation – package tidyr

```
# exemplo 4
df4 <- data.frame(player=c('A', 'B', 'C', 'D'),
year1=c(12, 15, 19, 19),
year2=c(22, 29, 18, 12))

df4
player year1 year2
     A    12    22
     B    15    29
     C    19    18
     D    19    12

# Gather columns into key-value pairs
gather(df4, key="year", value="points", 2:3)
player  year points
     A year1     12
     B year1     15
     C year1     19
     D year1     19
     A year2     22
     B year2     29
     C year2     18
     D year2     12
```

Data transformation – package tidyr

separate {tidyr} R Documentation

Separate a character column into multiple columns with a regular expression or numeric locations

Description

Given either a regular expression or a vector of character positions, `separate()` turns a single character column into multiple columns.

Data transformation – package tidyr

```
# separete
df5 <- data.frame(player=c('A', 'A', 'B', 'B', 'C', 'C'),
                  year=c(1, 2, 1, 2, 1, 2),
                  stats=c('22-2', '29-3', '18-6', '11-8', '12-5', '19-2'))

#view data frame
df5
player year stats
     A    1  22-2
     A    2  29-3
     B    1  18-6
     B    2  11-8
     C    1  12-5
     C    2  19-2

# Separate a character column into multiple columns with a regular expression

separate(df5, col=stats, into=c('dia', 'mes'), sep='-')
player year dia mes
     A    1  22   2
     A    2  29   3
     B    1  18   6
     B    2  11   8
     C    1  12   5
     C    2  19   2
```

Data transformation – package tidyr

```
df6 <- data.frame(player=c('A', 'A', 'B', 'B', 'C', 'C'),
                  year=c(1, 2, 1, 2, 1, 2),
                  stats=c('22/2/3', '29/3/4', '18/6/7', '11/1/2', '12/1/1', '19/2/4'))


df6
player year  stats
     A    1 22/2/3
     A    2 29/3/4
     B    1 18/6/7
     B    2 11/1/2
     C    1 12/1/1
     C    2 19/2/4

separate(df6, col=stats, into=c('dia', 'mes', 'ano'), sep='/')
player year dia mes ano
     A    1  22   2   3
     A    2  29   3   4
     B    1  18   6   7
     B    2  11   1   2
     C    1  12   1   1
     C    2  19   2   4
```

Data transformation – package tidyr

unite {tidyr} R Documentation

Unite multiple columns into one by pasting strings together

Description

Convenience function to paste together multiple columns into one.

Data transformation – package tidyr

```
df7 <- data.frame(player=c('A', 'A', 'B', 'B', 'C', 'C'),
                  year=c(1, 2, 1, 2, 1, 2),
                  points=c(22, 29, 18, 11, 12, 19),
                  assists=c(2, 3, 6, 8, 5, 2))


df7
player year points assists
     A    1     22       2
     A    2     29       3
     B    1     18       6
     B    2     11       8
     C    1     12       5
     C    2     19       2

unite(df7, col='dia_mes', c('points', 'assists'), sep='-')
player year dia_mes
     A    1    22-2
     A    2    29-3
     B    1    18-6
     B    2    11-8
     C    1    12-5
     C    2    19-2
```

Data transformation – package tidyr

```
df8 <- data.frame(player=c('A', 'A', 'B', 'B', 'C', 'C'),
                  year=c(1, 2, 1, 2, 1, 2),
                  points=c(22, 29, 18, 11, 12, 19),
                  assists=c(2, 3, 6, 8, 5, 2),
                  blocks=c(2, 3, 3, 2, 1, 0))
```

```
df8
player year points assists blocks
     A    1     22       2      2
     A    2     29       3      3
     B    1     18       6      3
     B    2     11       8      2
     C    1     12       5      1
     C    2     19       2      0
```

```
unite(df8, col='data_completa', c('points', 'assists', 'blocks'), sep='/')
player year data_completa
     A    1        22/2/2
     A    2        29/3/3
     B    1        18/6/3
     B    2        11/8/2
     C    1        12/5/1
     C    2        19/2/0
```

Data transformation – package reshape

```
# reestruturar
library(reshape)

head(mtcars)
head(t(mtcars))
```

```
  head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
  head(t(mtcars))
     Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout
mpg      21.00         21.000      22.80         21.400             18.70
cyl       6.00          6.000       4.00          6.000              8.00
disp    160.00        160.000     108.00        258.000            360.00
hp      110.00        110.000      93.00        110.000            175.00
drat      3.90          3.900       3.85          3.080              3.15
```

Data transformation – package reshape

```
id <-c(1,1,2,2)
time <-c(1,2,1,2)
x1 <-c(5,3,6,2)
x2 <-c(6,5,1,4)

mydata <-data.frame(id,time,x1,x2)
mydata
id time x1 x2
 1    1  5  6
 1    2  3  5
 2    1  6  1
 2    2  2  4

mdata <-melt(mydata, id=c("id","time"))
mdata
id time variable value
 1    1       x1     5
 1    2       x1     3
 2    1       x1     6
 2    2       x1     2
 1    1       x2     6
 1    2       x2     5
 2    1       x2     1
 2    2       x2     4
```

Data transformation – package reshape

```
means_var <- cast(mdata, id~variable, mean)
means_var
id x1  x2
 1  4 5.5
 2  4 2.5

means_time <- cast(mdata, time~variable, mean)
means_time
time  x1  x2
   1 5.5 3.5
   2 2.5 4.5

aggdata <-aggregate(mtcars, by=list(mtcars$cyl,mtcars$vs),
                    FUN=mean, na.rm=TRUE)
head(aggdata)
Group.1 Group.2      mpg cyl   disp       hp     drat       wt
      4       0 26.00000   4 120.30  91.0000 4.430000 2.140000
      6       0 20.56667   6 155.00 131.6667 3.806667 2.755000
      8       0 15.10000   8 353.10 209.2143 3.229286 3.999214
      4       1 26.73000   4 103.62  81.8000 4.035000 2.300300
      6       1 19.12500   6 204.55 115.2500 3.420000 3.388750
```

Data transformation – package dplyr

```
View(Banco)

library(dplyr)
Banco %>%
  group_by(catemp) %>%
  tally()
```

```
A tibble: 3 x 2
catemp     n
<chr>  <int>
A         94
B         27
C        390
```

```
Banco %>%
  count(catemp)
```

```
A tibble: 3 x 2
catemp     n
<chr>  <int>
A         94
B         27
C        390
```

```
Banco %>%
  group_by(catemp) %>%
  group_size()
```

```
 94  27 390
```

Data transformation – pacakge tibble

Tibbles são formatos de dataframe, a vantagem é que ao chamar um banco de dados tibble ele não enche a tela do R com todas as linhas. São projetados para que você não sobrecarregue seu console.

Data transformation – pacakge tibble

```
mtcars %>% add_column(z = -1:30, w = 0)
```

```
                  mpg cyl  disp  hp before_y drat after_x    wt  qsec vs am gear carb  z w
Mazda RX4        21.0   6 160.0 110       -1 3.90      -1 2.620 16.46  0  1    4    4 -1 0
Mazda RX4 Wag    21.0   6 160.0 110        0 3.90       0 2.875 17.02  0  1    4    4  0 0
Datsun 710       22.8   4 108.0  93        1 3.85       1 2.320 18.61  1  1    4    1  1 0
Hornet 4 Drive   21.4   6 258.0 110        2 3.08       2 3.215 19.44  1  0    3    1  2 0
Hornet Sportabout 18.7  8 360.0 175        3 3.15       3 3.440 17.02  0  0    3    2  3 0
Valiant          18.1   6 225.0 105        4 2.76       4 3.460 20.22  1  0    3    1  4 0
Duster 360       14.3   8 360.0 245        5 3.21       5 3.570 15.84  0  0    3    4  5 0
```

```
mtcars <-mtcars %>%
  add_column(before_y=-1:30, .before="drat")

mtcars <-mtcars %>%
  add_column(after_x=-1:30, .after="drat")

mtcars
```

```
                  mpg cyl  disp  hp before_y before_y.1 drat after_x.1 after_x    wt  qsec vs am gear carb
Mazda RX4        21.0   6 160.0 110       -1         -1 3.90        -1       -1 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21.0   6 160.0 110        0          0 3.90         0        0 2.875 17.02  0  1    4    4
Datsun 710       22.8   4 108.0  93        1          1 3.85         1        1 2.320 18.61  1  1    4    1
Hornet 4 Drive   21.4   6 258.0 110        2          2 3.08         2        2 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7  8 360.0 175        3          3 3.15         3        3 3.440 17.02  0  0    3    2
Valiant          18.1   6 225.0 105        4          4 2.76         4        4 3.460 20.22  1  0    3    1
```

Data transformation – package stringr

Strings não são componentes glamorosos e de alto perfil do R, mas desempenham um papel importante em muitas tarefas de limpeza e preparação de dados. O pacote stringr fornece um conjunto coeso de funções projetadas para tornar o trabalho com strings o mais fácil possível.

Data transformation – package stringr

```
library(stringr)

#Esta função recebe como argumento um vetor de strings e retorna o
# número de caracteres de cada string. Repare que o espaço " " é
#considerado um caractere.

str_length(bebes$ptl)
```

```
 [1] 6 6 6 6 6 6 6 6 6 6 6 6 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 2 2 6 6 6 6 6 6
[40] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 2 6 6 4 2 4 6 6 6 6 6 2
[79] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 6 6 6 6 6 6 6 6 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6
[118] 6 6 6 6 6 6 6 6 6 6 2 6 6 6 2 6 6 2 6 6 6 2 6 6 6 4 6 6 6 6 6 2 6 6 4 6 6 2 6 2
[157] 6 2 6 6 6 2 6 6 2 2 2 6 2 6 6 6 6 6 6 2 6 6 2 6 2 6 6 2 6 6 6 6 6
```

```
str_to_lower(bebes$smoke)
str_to_upper(bebes$smoke)
str_to_title(bebes$smoke)
```

```
str_to_lower(bebes$smoke)
```

```
 [1] "não" "não" "sim" "sim" "sim" "não" "não" "não" "sim" "sim" "não" "não" "não" "não" "sim"
[16] "sim" "não" "sim" "não" "sim" "não" "não" "não" "não" "não" "não" "sim" "não" "sim" "não"
[31] "não" "sim" "sim" "não" "não" "sim" "sim" "sim" "sim" "sim" "sim" "não" "não" "não" "sim"
[46] "sim" "não" "não" "não" "sim" "não" "não" "sim" "sim" "não" "não" "sim" "não" "não" "não"
[61] "não" "não" "não" "não" "sim" "não" "não" "sim" "não" "não" "sim" "sim" "sim" "não" "sim"
[76] "não" "não" "sim" "sim" "não" "não" "não" "não" "não" "não" "sim" "não" "não" "não" "não"
[91] "não" "não" "sim" "sim" "sim" "não" "não" "sim" "sim" "não" "não" "sim" "não" "não" "não"
[106] "não" "não" "não" "sim" "não" "não" "não" "sim" "não" "sim" "não" "não" "não" "não" "não"
[121] "não" "não" "não" "não" "não" "não" "não" "sim" "não" "não" "sim" "não" "sim" "não" "não"
[136] "não" "não" "não" "não" "sim" "sim" "sim" "não" "não" "sim" "sim" "não" "sim" "não" "não"
[151] "não" "não" "sim" "sim" "não" "sim" "sim" "sim" "não" "sim" "sim" "não" "não" "não" "sim"
[166] "sim" "não" "não" "sim" "não" "sim" "sim" "sim" "não" "não" "sim" "sim" "sim" "sim" "não"
[181] "não" "não" "sim" "sim" "sim" "não" "sim" "não" "sim"
```

Data transformation – package stringr

```
# str_trim - é comum encontrar textos que vêm com espaços a mais
de formulários em que cada usuário escreve da forma que prefere.
sexo <-c('M','F','M', ' F', '  M')
sexo
[1] "M"    "F"    "M"    " F"   "  M"

sexo_correto <-str_trim(sexo)
sexo_correto
[1] "M" "F" "M" "F" "M"


final <-str_sub(texto, end = 2)
final
[1] "Fe" "Ma" "In"

texto <-c("Feminino-01", "Masculino-02", "Indefinido-03")

teste <-str_sub(texto, end= -4)
teste
[1] "Feminino"   "Masculino"  "Indefinido"

teste <-str_sub(texto, start= -2)
teste
[1] "01" "02" "03"
```

Data transformation – package stringr

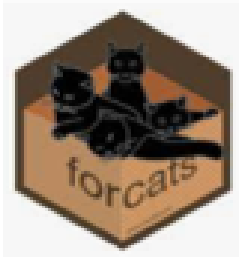
```
  # expressões regulares - ex/pattern “paulo$” indica que o texto deve ser terminado em “pa
  str_detect('sao paulo', pattern = 'paulo$')
[1] TRUE
  str_detect('sao paulo sp', pattern = 'paulo$')
[1] FALSE

  # str_replace() substitui somente a primeira
  # str_replace_all() substitui todas
  cidades <- c("S. José do Rio Preto", "São Paulo", "S. José dos Campos", "São Roque", "S.
  cidades
[1] "S. José do Rio Preto" "São Paulo"            "S. José dos Campos"   "São Roque"
[5] "S. S. da Grama"


  # colocar o . entre colchetes. Se não tivéssemos colocado, ele seria interpretado como um
drão procurado seria “S” seguido de qualquer caracter:
  str_replace(cidades, "S[.]","São")
[1] "São José do Rio Preto" "São Paulo"             "São José dos Campos"   "São Roque"
[5] "São S. da Grama"
```

Data transformation – package stringr

```
  # fixed padrão e não uma regex
  str_replace(cidades, fixed("S."),"São")
[1] "São José do Rio Preto" "São Paulo"             "São José dos Campos"   "São Roque"
[5] "São S. da Grama"
  str_replace(cidades, "S.","São")
[1] "São José do Rio Preto" "Sãoo Paulo"            "São José dos Campos"   "Sãoo Roque"
[5] "São S. da Grama"

  # substitui todas
  str_replace_all(cidades, "S[.]","São")
[1] "São José do Rio Preto" "São Paulo"             "São José dos Campos"   "São Roque"
[5] "São São da Grama"

  # CPF
  cpf <- c("303.030.111-33", "102-177-011-20", "987.220.199.00")
  str_replace_all(cpf, "[.-]"," ")
[1] "303 030 111 33" "102 177 011 20" "987 220 199 00"
  str_replace_all(cpf, "[.-]","")
[1] "30303011133" "10217701120" "98722019900"
```

Data transformation – package forcats

As principais funções do forcats servem para alterar a **ordem** e modificar os **níveis** de um fator. Fatores são uma classe de objetos no R criado para representar as variáveis categóricas numericamente.



Data transformation – package forcats

```
library(dplyr)
library(forcats)

fct_count(bebes$ptl)
# A tibble: 4 x 2
f           n
<fct>   <int>
1 Dois        5
2 Nenhum    159
3 Três        1
4 Um         24

bebes$smoke <- as.factor(bebes$smoke)
bebes$ptl <- as.factor(bebes$ptl)
fct_c(bebes$smoke, bebes$ptl) %>% levels()
[1] "Não"    "Sim"    "Dois"   "Nenhum" "Três"   "Um"
```

Data transformation – magrittr %>%pipe

O *pacote magrittr* (a ser pronunciado com um sotaque francês sofisticado) tem dois objetivos: diminuir o tempo de desenvolvimento e melhorar a legibilidade e a manutenção do código.

Data transformation – magrittr %>%pipe

```
library(magrittr)

bebes <- bebes %>%
  transform(testa = lwt * 2)

bebes %>% filter(lwt > 200) %>% select(id:lwt)
   id low age lwt
1 108   0  36 202
2 126   0  31 215
3 159   0  28 250
4 168   0  18 229
5 187   0  19 235
6 202   0  25 241

bebes$teste <-sqrt(sum(bebes$lwt))
bebes$teste1 <- bebes$lwt%>% sum() %>% sqrt()
bebes
     id low age lwt smoke    ptl  ht testa    teste   teste1
1    85   0  NA 182   Não Nenhum Não   364 156.8056 156.8056
2    86   0  NA 155   Não Nenhum Não   310 156.8056 156.8056
3    87   0  NA 105   Sim Nenhum Não   210 156.8056 156.8056
4    88   0  NA 108   Sim Nenhum Não   216 156.8056 156.8056
5    89   0  18 107   Sim Nenhum Não   214 156.8056 156.8056
```

Data transformation – package purrr

purrr aprimora o kit de ferramentas de programação funcional (FP) do R, fornecendo um conjunto completo e consistente de ferramentas para trabalhar com funções e vetores. Se você nunca ouviu falar de FP antes, o melhor lugar para começar é a família de map() funções que permitem substituir muitos loops for por um código mais sucinto e fácil de ler.

Data transformation – package purrr

```
library(purrr)

square <- function(x){
  return(x*x)
}

bebes$agee <-map(bebes$age, square)

to_Power <- function(x, y){
  return(x*y)
}

bebes$teste <-map2(bebes$age, bebes$lwt, to_Power)

bebes
```

id	low	age	lwt	smoke	ptl	ht	agee	teste
85	0	*NA*	182	Não	Nenhum	Não	NA	NA
86	0	*NA*	155	Não	Nenhum	Não	NA	NA
87	0	*NA*	105	Sim	Nenhum	Não	NA	NA
88	0	*NA*	108	Sim	Nenhum	Não	NA	NA
89	0	18	107	Sim	Nenhum	Não	324	1926

Data transformation – package lubridate

As funções do R base são, em grande parte, contraintuitivas e podem mudar de acordo com o tipo do objeto que você está usando (data, hora, data/hora). Além disso, características como fusos horários, anos bissextos, horários de verão, entre outras, podem não estar bem especificadas ou nem mesmo sendo levadas em conta.

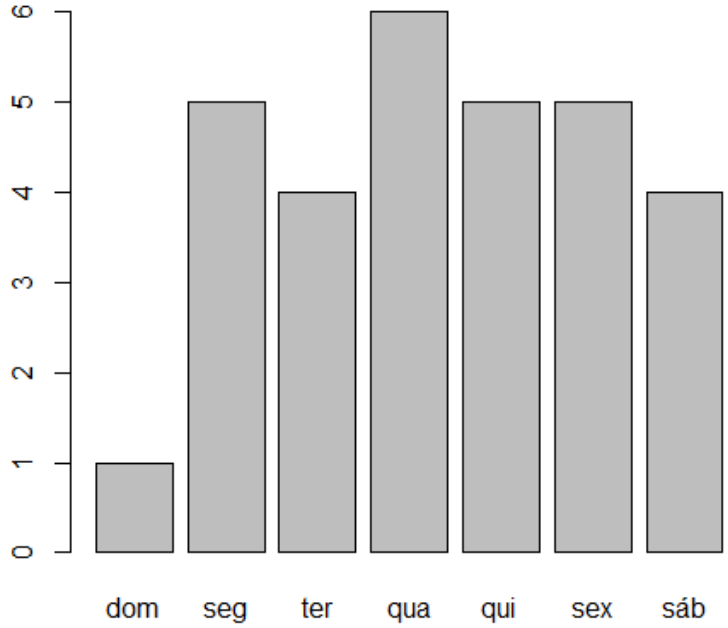
O pacote {lubridate}, criado por Garrett Grolemund e Hadley Wickham, surgiu para lidar com esses problemas, fazendo o trabalho com datas ser muito mais fácil.

Data transformation – package lubridate

```
library(readxl)
cliente <- read_excel("cliente.xlsx")
View(cliente)

str(cliente)

names(cliente)

library(lubridate)

cliente$dia <-day(cliente$Data_Compra)
cliente$meses <-month(cliente$Data_Compra, label = T)
cliente$ano <- year(cliente$Data_Compra)
cliente$dia_semana <- wday(cliente$Data_Compra, label = T)
cliente$data_hoje <- Sys.Date()
cliente$intervalo <-interval(cliente$Data_Compra, cliente$data_hoje)
cliente$diferenca_anos <-round(cliente$intervalo/ dyears(1))
cliente$diferenca_meses <-round(cliente$diferenca_anos * 12)
cliente$diferenca_minutos <- round(cliente$intervalo/ dminutes(1))
```

Data transformation – package lubridate

```
barplot(table(cliente$dia_semana))
```



Family Apply

A família Apply representa um conjunto de funções básicas do R que permite realizar operações sobre os dados contidos nas várias estruturas disponíveis (vetor, data frame, listas).

Family Apply

```
set.seed(123)
precos <- runif(20, min = 2, max = 10)
vendas <- runif(20, min = 20, max = 100)
estoque <- sample(c(20:50), size = 20, replace = TRUE)

dataset_loja <- data.frame(id_produto = c(1:20),preco = precos,
+                           total_vendas = vendas, total_estoque = estoque)

head(dataset_loja)
  id_produto    preco total_vendas total_estoque
1          1 4.300620     91.16315            34
2          2 8.306441     75.42427            29
3          3 5.271815     71.24055            32
4          4 9.064139     99.54158            26
5          5 9.523738     72.45646            28
6          6 2.364452     76.68244            28

# Apply - soma das linhas
# margin 2 vertical
apply(X = dataset_loja[,-c(1)], MARGIN = 2, FUN = sum)
      preco  total_vendas total_estoque
   128.1293     1322.7534      682.0000
```

Family Apply

```
  # lapply - coluna
  # média das variáveis/ colunas
  lapply(X = dataset_loja[,-c(1)], FUN = mean)
$preco
[1] 6.406467

$total_vendas
[1] 66.13767

$total_estoque
[1] 34.1


  # Sapply - mesma coisa que lapply - porem outro formato
  sapply(dataset_loja[,-c(1,2)], mean)
 total_vendas total_estoque
    66.13767      34.10000
```

Family Apply

```
# tapply - aplica uma função a um subconjunto de um vetor que é construído
#a partir de um outro vetor (normalmente, um fator).
set.seed(123)
fornecedores <- sample(c('Fornecedor A','Fornecedor B','Fornecedor C',
                         'Fornecedor D'), size = 20, replace = TRUE)


# novo dataset
dataset_loja <- data.frame(cbind(dataset_loja), fornecedor = fornecedores)
head(dataset_loja)
  id_produto    preco total_vendas total_estoque   fornecedor
1          1 4.300620     91.16315            34 Fornecedor C
2          2 8.306441     75.42427            29 Fornecedor C
3          3 5.271815     71.24055            32 Fornecedor C
4          4 9.064139     99.54158            26 Fornecedor B
5          5 9.523738     72.45646            28 Fornecedor C
6          6 2.364452     76.68244            28 Fornecedor B


# aplico a função tapply para descobrir a média de preço de vendas
# para cada fornecedor:
# média de preço praticado para cada fornecedor
tapply(dataset_loja[,c('preco')], dataset_loja[,ncol(dataset_loja)], mean)
Fornecedor A Fornecedor B Fornecedor C Fornecedor D
    4.856820     6.094775     7.159011     6.811684
```

Family Apply

```
# vapply() similar ao sapply, porém tem um tipo de valor de retorno pré-estabelecido.
# obter o resumo estatístico das variáveis
# fivenu retorna 5 estatistica
tabela <-vapply(dataset_loja[,-c(1,ncol(dataset_loja))], FUN = fivenum,
       FUN.VALUE =
         c(Min. = 0,
           '1stQu.' = 0,
           Media = 0,
           '3rd Qu.' = 0,
           Max = 0))

View(tabela)
```

tibble.R* | tabela | dataset_loja

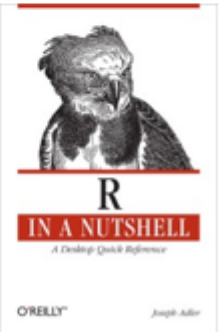
Filter

	preco	total_vendas	total_estoque
Min.	2.336476	21.96909	21.0
1stQu.	4.461993	44.29363	26.5
Media	6.318162	71.84850	30.5
3rd Qu.	9.101746	82.15708	43.0
Max	9.654667	99.54158	48.0

Referências Bibliográficas

Teetor, Paul. 2011. 25 Recipes for Getting Started with R. O'Reilly Media.
Teetor, Paul. 2011. R Cookbook. O'Reilly Media.
Adler, Joseph. 2012. R in a Nutshell. O'Reilly Media.

Tutorial sobre o R: http://tryr.codeschool.com/
Rmarkdown em http://rmarkdown.rstudio.com/
Lista de discussão r-br-request@listas.c3sl.ufpr.br



OBRIGADO


FIAP