# Fatores

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

vec1 <- c("Macho","Femea","Femea","Macho","Macho")

vec1

fac\_vec1 <- factor(vec1)

fac\_vec1

class(vec1)

class(fac\_vec1)

# Variáveis categóricas nominais

# Não existe uma ordem implícita

animais <- c("Zebra", "Pantera", "Rinoceronte", "Macaco", "Tigre")

animais

class(animais)

fac\_animais <- factor(animais)

fac\_animais

class(fac\_animais)

levels(fac\_animais)

# Variáveis categóricas ordinais

# Possuem uma ordem natural

grad <- c("Mestrado", "Doutorado", "Bacharelado", "Mestrado", "Mestrado")

grad

fac\_grad <- factor(grad, order = TRUE, levels = c("Doutorado", "Mestrado", "Bacharelado"))

fac\_grad

levels(fac\_grad)

# Sumarizar os dados fornece uma visão geral sobre o conteúdo das variáveis

summary(fac\_grad)

summary(grad)

vec2 <- c("M", "F", "F", "M", "M", "M", "F", "F", "M", "M", "M", "F", "F", "M", "M")

vec2

fac\_vec2 <- factor(vec2)

fac\_vec2

levels(fac\_vec2) <- c("Femea", "Macho")

fac\_vec2

summary(fac\_vec2)

summary(vec2)

# Mais exemplos

data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)

fdata = factor(data)

fdata

rdata = factor(data, labels = c("I","II","III"))

rdata

# Fatores Não-Ordenados

set1 <- c("AA", "B", "BA", "CC", "CA", "AA", "BA", "CC", "CC")

set1

# Transformando os dados. O R apenas criou os níveis, o que não significa que exista uma hierarquia

f.set1 <- factor(set1)

f.set1

class(f.set1)

is.ordered(f.set1)

# Fatores Ordenados

o.set1 <- factor(set1,

levels = c("CA", "BA", "AA", "CC", "B"),

ordered = TRUE)

o.set1

is.ordered(o.set1)

as.numeric(o.set1)

table(o.set1)

# Fatores e Dataframes

df <- read.csv2("etnias.csv", sep = ',')

df

# Variáveis do tipo fator

str(df)

# Níveis dos fatores

# Internamente, o R armazena valores inteiros e faz um mapeamento para as strings (em ordem alfabética)

# e agrupa as estatísticas por níveis. Agora, se fizermos sumarização de estatísticas, é possível visualizar

# a contabilização de para cada categoria

levels(df$Etnia)

summary(df$Etnia)

# Plot

# Agora se fizermos um plot, temos um boxplot para estas variáveis categóricas

plot(df$Idade~df$Etnia, xlab = 'Etnia', ylab = 'Idade', main = 'Idade por Etnia')

# Regressão

summary(lm(Idade~Etnia, data = df))

# Convertendo uma coluna em variável categórica. Isso criará um fator não-ordenado

df

str(df)

df$Estado\_Civil.cat <- factor(df$Estado\_Civil, labels = c("Solteiro", "Casado", "Divorciado"))

df

str(df)

# Compreendendo a ordem dos fatores

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Níveis dos fatores

# Internamente, o R armazena valores inteiros e faz um mapeamento para as strings (em ordem alfabética)

# e agrupa as estatísticas por níveis.

# Criando vetores

vec1 <- c(1001, 1002, 1003, 1004, 1005)

vec2 <- c(0, 1, 1, 0, 2)

vec3 <- c('Verde','Laranja','Azul','Laranja','Verde')

# Unindo os vetores em um dataframe

df <- data.frame(vec1, vec2, vec3)

df

# Verificando que o R categorizou a última coluna como fator

str(df)

# Verificando os níveis do fator. Perceba que os níveis estão categorizados em ordem alfabética

levels(df$vec3)

# Criando uma outra coluna e atribuindo labels

df$cat1 <- factor(df$vec3, labels = c("cor2", "cor1", "cor3"))

df

# Internamente, os fatores são registrados como inteiros, mas a ordenação segue a ordem alfabética

# das strings

str(df)

# Veja como foi feita a atribuição:

# Azul = cor2

# Laranja = cor1

# Verde = cor3

# Ou seja, os vetores com os labels, seguiram a ordem alfabética dos níveis classificados pelo R

# Criando uma outra coluna e atribuindo labels

# Ao aplicarmos a função factor() a coluna vec2, internamente o R classificou em ordem alfabética

# e quando atribuímos os labels, foi feita a associação.

df$cat2 <- factor(df$vec2, labels = c("Divorciado", "Casado", "Solteiro"))

df

str(df)

levels(df$cat2)

# A família apply - uma forma elegante de fazer loops

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# apply() - arrays e matrizes

# tapply() - os vetores podem ser divididos em diferentes subsets

# lapply() - vetores e listas

# sapply() - versão amigável da lapply

# vapply() - similar a sapply, com valor de retorno modificado

# rapply() - similar a lapply()

# eapply() - gera uma lista

# mapply() - similar a sapply, multivariada

# by

# Se você estiver trabalhando com os objetos:

# list, numeric, character (list/vecor) => sapply ou lapply

# matrix, data.frame (agregação por coluna) => by / tapply

# Operações por linha ou operações específicas => apply

# Usando um Loop

lista1 <- list(a = (1:10), b = (45:77))

# Calculando o total de cada elemento da lista com loop for

valor\_a = 0

valor\_b = 0

for (i in lista1$a){

valor\_a = valor\_a + i

}

for (j in lista1$b){

valor\_b = valor\_b + j

}

print(valor\_a)

print(valor\_b)

# Calculando cada elemento da lista com sapply

?sapply

sapply(lista1, sum)

# Aplicando funções com sapply

sapply(lista1, mean)

# apply()

?apply

x <- matrix(rnorm(9), nr = 3, byrow = T)

x

apply(x, 1, mean)

apply(x, 2, mean)

apply(x, 1, plot)

resultapply <- apply(x, 1, mean)

resultapply

escola <- data.frame(Aluno = c('Alan', 'Alice', 'Alana', 'Aline', 'Alex', 'Ajay'),

Matematica = c(90, 80, 85, 87, 56, 79),

Geografia = c(100, 78, 86, 90, 98, 67),

Quimica = c(76, 56, 89, 90, 100, 87))

escola

escola$Geografia

# Calculando a média por aluno

escola$Media = NA

escola

escola$Media = apply(escola[, c(2, 3, 4)], 1, mean)

escola

escola$Media = round(escola$Media)

escola

# tapply() vs sqldf

install.packages('sqldf')

require(sqldf)

escola2 <- data.frame(Aluno = c('Alan', 'Alice', 'Alana', 'Alan', 'Alice', 'Alana'),

Semestre = c(1, 1, 1, 2, 2, 2),

Matematica = c(90, 80, 85, 87, 56, 79),

Geografia = c(100, 78, 86, 90, 98, 67),

Quimica = c(76, 56, 89, 90, 100, 87))

escola2

# Obs: Esta mensagem de warning aparece somente na primeira execução do comando sqldf():

# Warning message: Quoted identifiers should have class SQL, use DBI::SQL() if the caller performs the quoting.

sqldf("select aluno, sum(Matematica), sum(Geografia), sum(Quimica) from escola2 group by aluno")

tapply(c(escola2$Matematica), escola2$Aluno, sum)

# by

?by

# A coluna "aluno" aqui não é necessária

sqldf("select aluno, sum(Matematica), sum(Geografia), sum(Quimica) from escola2 group by semestre")

sqldf("select sum(Matematica), sum(Geografia), sum(Quimica) from escola2 group by semestre")

by(escola2[, c(2, 3, 4)], escola2$Semestre, colSums)

# lapply()

?lapply

lista1 <- list(a = (1:10), b = (45:77))

lista1

lapply(lista1, sum)

sapply(lista1, sum)

# vapply()

?vapply

vapply(lista1, fivenum, c(Min. = 0, "1stQu." = 0, Median = 0, "3rd Qu." = 0, Max = 0))

# replicate

?replicate

replicate(7, runif(10))

# mapply()

?mapply

mapply(rep, 1:4, 4:1)

# rapply()

?rapply

lista2 <- list(a = c(1:5), b = c(6:10))

lista2

rapply(lista2, sum)

rapply(lista2, sum, how = "list")

# Text Mining

# \*\*\*\*\* Esta é a versão 3.0 deste script, atualizado em 02/10/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3, 3.4.0 e 3.4.1 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.2 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Exemplo 1

install.packages(c("tm", "SnowballC", "wordcloud", "RColorBrewer"))

library(tm)

library(SnowballC)

library(wordcloud)

library(RColorBrewer)

# Carregando o dataset

df <- read.csv('questoes.csv', stringsAsFactors = FALSE)

head(df)

# Criando um Corpus

dfCorpus <- Corpus(VectorSource(df$Question))

class(dfCorpus)

# Imprimindo uma linha do Corpus

dfCorpus[[1]][[1]]

# Remove pontuação

dfCorpus <- tm\_map(dfCorpus, removePunctuation)

# Remover palavras específicas do inglês

dfCorpus <- tm\_map(dfCorpus, removeWords, stopwords('english'))

# Neste processo, várias versões de uma palavras são convertidas em uma única instância

dfCorpus <- tm\_map(dfCorpus, stemDocument)

# Removendo palavras específicas

dfCorpus <- tm\_map(dfCorpus, removeWords, c('the', 'this', stopwords('english')))

# wordcloud

wordcloud(dfCorpus, max.words = 100, random.order = FALSE)

# Exemplo 2

install.packages("tm")

install.package("SnowballC")

install.packages("wordcloud")

install.packages("RColorBrewer")

library("tm")

library("SnowballC")

library("wordcloud")

library("RColorBrewer")

# Lendo o arquivo

arquivo <- "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt"

texto <- readLines(arquivo)

# Carregando os dados como Corpus

docs <- Corpus(VectorSource(texto))

# Pré-processamento

inspect(docs)

toSpace <- content\_transformer(function (x , pattern ) gsub(pattern, " ", x))

docs <- tm\_map(docs, toSpace, "/")

docs <- tm\_map(docs, toSpace, "@")

docs <- tm\_map(docs, toSpace, "\\|")

# Converte o texto para minúsculo

docs <- tm\_map(docs, content\_transformer(tolower))

# Remove números

docs <- tm\_map(docs, removeNumbers)

# Remove as palavras mais comuns do idioma inglês

docs <- tm\_map(docs, removeWords, stopwords("english"))

# Você pode definir um vetor de palavras (stopwords) a serem removidas do texto

docs <- tm\_map(docs, removeWords, c("blabla1", "blabla2"))

# Remove pontuação

docs <- tm\_map(docs, removePunctuation)

# Elimina espaços extras

docs <- tm\_map(docs, stripWhitespace)

# Text stemming

docs <- tm\_map(docs, stemDocument)

dtm <- TermDocumentMatrix(docs)

m <- as.matrix(dtm)

v <- sort(rowSums(m),decreasing=TRUE)

d <- data.frame(word = names(v), freq = v)

head(d, 10)

# wordcloud

set.seed(1234)

wordcloud(words = d$word, freq = d$freq, min.freq = 1,

max.words = 200, random.order = FALSE, rot.per = 0.35,

colors = brewer.pal(8, "Dark2"))

# Tabela de frequência

findFreqTerms(dtm, lowfreq = 4)

findAssocs(dtm, terms = "freedom", corlimit = 0.3)

head(d, 10)

# Gráficos de barras com as palavras mais frequentes

barplot(d[1:10,]$freq, las = 2, names.arg = d[1:10,]$word,

col ="lightblue", main ="Most frequent words",

ylab = "Word frequencies")

# Time e Datas

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Hora e Data do sistema

hoje <- Sys.Date()

hoje

class(hoje)

Sys.time()

Sys.timezone()

# Data - representado por Date

# Armazenados como número de dias desde 1 de Janeiro de 1970

# Time - representado por POSIXct

# Armazenados como número de segundos desde 1 de Janeiro de 1970

# Formatando Datas

# %d: dia do mês em 2 dígitos (13)

# %m: mês em 2 digitos (01)

# %y: ano em 2 dígitos (82)

# %Y: ano em 4 dígitos (1982)

# %A: dia da semana (Friday)

# %a: dia da semana abreviado (Fri)

# %B: mês (July)

# %b: mês abreviado (Jul)

# Formatando Time

# %H: hora (00-23)

# %M: minuto

# %S: segundo

# %T: formado reduzido para %H:%M:%S

?strptime

# Formatando a saída - as.Date()

as.Date("2016-06-28")

as.Date("20160628434")

as.Date("Jun-28-16", format = "%b-%d-%y")

as.Date("28 June, 2016", format = "%d %B, %Y")

# Função format()

Sys.Date()

format(Sys.Date(), format = "%d %B, %Y")

format(Sys.Date(), format = "Hoje é %A!")

# Convertendo Datas - as.POSIXct

date1 <- "Jun 13, '96 hours:23 minutes:01 seconds:45"

date1\_convert <- as.POSIXct(date1, format = "%B %d, '%y hours:%H minutes:%M seconds:%S")

date1\_convert

# Operações com Datas

data\_de\_hoje <- as.Date("2016-06-25", format = "%Y-%m-%d")

data\_de\_hoje

data\_de\_hoje + 1

my\_time <- as.POSIXct("2016-05-14 11:24:134")

my\_time

my\_time + 1

data\_de\_hoje - as.Date(my\_time)

data\_de\_hoje - my\_time

# Convertendo Data em formato específico

dts = c(1127056501,1104295502,1129233601,1113547501,1119826801,1132519502,1125298801,1113289201)

mydates = dts

class(mydates) = c('POSIXt','POSIXct')

mydates

mydates = structure(dts, class = c('POSIXt','POSIXct'))

mydates

# Função ISODate

b1 = ISOdate(2011,3,23)

b1

b2 = ISOdate(2012,9,19)

b2

b2 - b1

difftime(b2, b1, units = 'weeks')

# Pacote lubridate

?lubridate

install.packages("lubridate")

require(lubridate)

ymd("20160604")

mdy("06-04-2016")

dmy("04/06/2016")

chegada <- ymd\_hms("2016-06-04 12:00:00", tz = "Pacific/Auckland")

partida <- ymd\_hms("2011-08-10 14:00:00", tz = "Pacific/Auckland")

chegada

partida

second(chegada)

second(chegada) <- 23

chegada

wday(chegada)

wday(chegada, label = TRUE)

class(chegada)

interval(chegada, partida)

tm1.lub <- ymd\_hms("2016-05-24 23:55:26")

tm1.lub

tm2.lub <- mdy\_hm("05/25/16 08:32")

tm2.lub

year(tm1.lub)

week(tm1.lub)

tm1.dechr <- hour(tm1.lub) + minute(tm1.lub)/60 + second(tm1.lub)/3600

tm1.dechr

force\_tz(tm1.lub, "Pacific/Auckland")

# Gerando um dataframe de datas

sono <- data.frame(bed.time = ymd\_hms("2013-09-01 23:05:24", "2013-09-02 22:51:09",

"2013-09-04 00:09:16", "2013-09-04 23:43:31", "2013-09-06 00:17:41", "2013-09-06 22:42:27",

"2013-09-08 00:22:27"), rise.time = ymd\_hms("2013-09-02 08:03:29", "2013-09-03 07:34:21",

"2013-09-04 07:45:06", "2013-09-05 07:07:17", "2013-09-06 08:17:13", "2013-09-07 06:52:11",

"2013-09-08 07:15:19"), sleep.time = dhours(c(6.74, 7.92, 7.01, 6.23, 6.34, 7.42, 6.45)))

sono

sono$eficiencia <- round(sono$sleep.time/(sono$rise.time - sono$bed.time) \* 100, 1)

sono

# Gerando um plot a partir de datas

par(mar = c(5, 4, 4, 4))

plot(round\_date(sono$rise.time, "day"), sono$eficiencia, type = "o", col = "blue", xlab = "Manhã", ylab = NA)

par(new = TRUE)

plot(round\_date(sono$rise.time, "day"), sono$sleep.time/3600, type = "o", col = "red", axes = FALSE, ylab = NA, xlab = NA)

axis(side = 4)

mtext(side = 4, line = 2.5, col = "red", "Duração do Sono")

mtext(side = 2, line = 2.5, col = "blue", "Eficiência do Sono")

# Scatterplots

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Define os dados

set.seed(67)

x = rnorm(10,5,7)

y = rpois(10,7)

z = rnorm(10,6,7)

t = rpois(10,9)

# Cria o Plot

plot(x, y, col = 123, pch = 10, main = "Multi Scatterplot",

col.main = "red", cex.main = 1.5, xlab = "indep",

ylab = "depend")

# Adiciona outros dados

points(z, t, col = "blue", pch = 4)

# Adiciona outros dados

points(y, t, col = 777, pch = 9)

# Cria legenda

legend(-6,5.9, legend = c("Nível 1", "Nível 2", "Nível 3"),

col = c(123, "blue", 777), pch = c(10,4,9),

cex = 0.65, bty = "n")

# Boxplots

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

?boxplot

?sleep

# Permite utilizar as colunas sem especificar o mome do dataset

attach(sleep)

# Construção do boxplot

sleepboxplot = boxplot(data = sleep, extra ~ group,

main = "Duração do Sono",

col.main = "red", ylab = "Horas", xlab = "Droga")

# Cálculo da média

means = by(extra, group, mean)

# Adiciona a média ao gráfico

points(means, col = "red")

# Boxplot horizontal

horizontalboxplot = boxplot(data = sleep, extra ~ group,

ylab = "", xlab = "", horizontal = T)

horizontalboxplot = boxplot(data = sleep, extra ~ group,

ylab = "", xlab = "", horizontal = T,

col = c("blue", "red") )

# Histogramas

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

?hist

?cars

# Definindo os dados

dados = cars$speed

# Construindo um histograma

hist(dados)

hist(dados, breaks = 10)

hist(dados, labels = T, breaks = c(0,5,10,20,30))

hist(dados, labels = T, breaks = 10)

hist(dados, labels = T, ylim = c(0,10), breaks = 10)

# Adicionando linhas ao histograma

hey = hist(dados, breaks = 10)

xaxis = seq(min(dados), max(dados), length = 10)

yaxis = dnorm(xaxis, mean = mean(dados), sd = sd(dados))

yaxis = yaxis\*diff(hey$mids)\*length(dados)

lines(xaxis, yaxis, col = "red")

# Gráficos de Pizza

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

.Platform$GUI

?pie

# Criando as fatias

fatias = c(40, 20, 40)

# Nomeando os labels

paises = c("Brasil", "Argentina", "Chile")

# Unindo paises e fatias

paises = paste(paises, fatias)

# to further specify the labels

paises = paste(paises, "%", sep = "")

colors()

# Construindo um gráfico

pie(fatias, labels = paises,

col = c("darksalmon", "gainsboro", "lemonchiffon4"),

main ="Dsitribuição de Vendas")

# Trabalhando com dataframes

?iris

attach(iris)

Values = table(Species)

labels = paste(names(Values))

pie(Values, labels = labels, main = "Distribuição de Espécies")

# 3D

install.packages("plotrix")

library(plotrix)

pie3D(fatias, labels = paises, explode = 0.05,

col = c("steelblue1", "tomato2", "tan3"),

main = "Dsitribuição de Vendas")

# Bar Plots

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

?barplot

# Preparando os dados

casamentos <- matrix(c(652,1537,598,242,36,46,38,21,218,327,106,67),

nrow = 3, byrow = T)

# Nomeando a matriz

colnames(casamentos) <- c("0","1-150","151-300",">300")

rownames(casamentos) <- c("Casado","Divorciado","Solteiro")

# Construindo o plot

barplot(casamentos)

barplot(casamentos, beside = T)

# Transposta da matriz

barplot(t(casamentos), beside = T)

# ggplot2

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Um sistema gráfico completo, alternativo ao sistema básico

# de gráficos do R

# Oferece mais opções de modificação, legendas prontas,

# formatação mais sólida

install.packages("ggplot2")

library(ggplot2)

# Plotando um gráfico básico com qplot()

data(tips, package = 'reshape2')

qplot(total\_bill, tip, data = tips, geom = "point")

# Camada 1

camada1 <- geom\_point(

mapping = aes(x = total\_bill, y = tip, color = sex),

data = tips,

size = 3

)

ggplot() + camada1

?aes

??aes

# Contruindo um modelo de regressão

modelo\_base <- lm(tip ~ total\_bill, data = tips)

modelo\_fit <- data.frame(

total\_bill = tips$total\_bill,

predict(modelo\_base, interval = "confidence")

)

head(modelo\_fit)

# Camada 2

camada2 <- geom\_line(

mapping = aes(x = total\_bill, y = fit),

data = modelo\_fit,

color = "darkred"

)

ggplot() + camada1 + camada2

# Camada 3

camada3 <- geom\_ribbon(

mapping = aes(x = total\_bill, ymin = lwr, ymax = upr),

data = modelo\_fit,

alpha = 0.3

)

ggplot() + camada1 + camada2 + camada3

# Versão final otimizada

ggplot(tips, aes(x = total\_bill, y = tip)) +

geom\_point(aes(color = sex)) +

geom\_smooth(method = 'lm')

# Gravando o gráfico em um objeto

myplot <- ggplot(tips, aes(x = total\_bill, y = tip)) +

geom\_point(aes(color = sex)) +

geom\_smooth(method = 'lm')

class(myplot)

print(myplot)

# ScatterPlot com linha de regressão

# Dados

data = data.frame(cond = rep(c("Obs 1", "Obs 2"),

each = 10), var1 = 1:100 +

rnorm(100,sd = 9), var2 = 1:100 +

rnorm(100,sd = 16))

# Plot

ggplot(data, aes(x = var1, y = var2)) +

geom\_point(shape = 1) +

geom\_smooth(method = lm , color = "red", se = FALSE)

?lm

# Bar Plot

# Dados

data = data.frame(grupo = c("A ","B ","C ","D ") ,

valor = c(33,62,56,67) ,

num\_obs = c(100,500,459,342))

# Gerando a massa de dados

data$right = cumsum(data$num\_obs) + 30 \* c(0:(nrow(data)-1))

data$left = data$right - data$num\_obs

# Plot

ggplot(data, aes(ymin = 0)) +

geom\_rect(aes(xmin = left, xmax = right,

ymax = valor, colour = grupo, fill = grupo)) +

xlab("Número de obs") + ylab("Valor")

# Usando mtcars

head(mtcars)

ggplot(data = mtcars, aes(x = disp, y = mpg)) + geom\_point()

# Outro aspecto que pode ser mapeado nesse gráfico é a cor dos pontos

ggplot(data = mtcars,

aes(x = disp, y = mpg,

colour = as.factor(am))) + geom\_point()

# No entanto, tambem podemos mapear uma variável contínua à cor dos pontos:

ggplot(mtcars, aes(x = disp, y = mpg, colour = cyl)) + geom\_point()

# Também podemos mapear o tamanho dos pontos à uma variável de interesse:

# A legenda é inserida no gráfico automaticamente

ggplot(mtcars, aes(x = disp, y = mpg, colour = cyl, size = wt)) + geom\_point()

# Os geoms definem qual forma geométrica será utilizada para a visualização dos dados no gráfico.

ggplot(mtcars, aes(x = as.factor(cyl), y = mpg)) + geom\_boxplot()

# Histogramas

ggplot(mtcars, aes(x = mpg), binwidth = 30) + geom\_histogram()

# Gráfico de Barras

ggplot(mtcars, aes(x = as.factor(cyl))) + geom\_bar()

# Personalizando os gráficos

colors()

ggplot(mtcars, aes(x = as.factor(cyl), y = mpg,

colour = as.factor(cyl))) + geom\_boxplot()

ggplot(mtcars, aes(x = as.factor(cyl), y = mpg,

fill = as.factor(cyl))) + geom\_boxplot()

ggplot(mtcars,

aes(x = as.factor(cyl), y = mpg)) +

geom\_boxplot(color = "blue", fill = "seagreen4")

# Podemos alterar os eixos

ggplot(mtcars, aes(x = mpg)) +

geom\_histogram() +

xlab("Milhas por galão") + ylab("Frequência")

# Alterar os limites do gráfico

ggplot(mtcars, aes(x = mpg)) + geom\_histogram() + xlab("Milhas por galão") + ylab("Frequência") +

xlim(c(0, 40)) +

ylim(c(0,8))

# Legendas

ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +

geom\_bar() +

labs(fill = "cyl")

# Trocando a posição da legenda

ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +

geom\_bar() +

labs(fill = "cyl") +

theme(legend.position = "top")

# Sem legenda

ggplot(mtcars, aes(x = as.factor(cyl), fill = as.factor(cyl))) +

geom\_bar() +

guides(fill = FALSE)

# Facets

ggplot(mtcars, aes(x = mpg, y = disp, colour = as.factor(cyl))) +

geom\_point() +

facet\_grid(am~.)

ggplot(mtcars, aes(x = mpg, y = disp, colour = as.factor(cyl))) +

geom\_point() +

facet\_grid(.~am)

# Plots diferentes juntos (diferente de Facet)

install.packages("gridExtra")

library(gridExtra)

library(ggplot2)

# Dataset diamonds

data(diamonds)

# Histograma como plot1

plot1 <- qplot(price, data = diamonds, binwidth = 1000)

# ScatterPlot como plot2

plot2 <- qplot(carat, price, data = diamonds, colour = cut)

# Combina os 2 plots na mesma área

grid.arrange(plot1, plot2, ncol = 1)

# Gráficos de Densidade

ggplot(data = diamonds, aes(x = price, group = cut, fill = cut)) +

geom\_density(adjust = 1.5)

ggplot(data = diamonds, aes(x = price, group = cut, fill = cut)) +

geom\_density(adjust = 1.5 , alpha = 0.2)

ggplot(data = diamonds,aes(x = price, group = cut, fill = cut)) +

geom\_density(adjust = 1.5, position = "fill")

# Facets com reshape

install.packages("reshape2")

install.packages("plotly")

library(reshape2)

library(plotly)

sp <- ggplot(tips, aes(x = total\_bill, y = tip/total\_bill)) + geom\_point(shape = 1)

sp + facet\_grid(sex ~ .)

ggplotly()

sp + facet\_grid(. ~ sex)

ggplotly()

sp + facet\_wrap( ~ day, ncol = 2)

ggplotly()

ggplot(mpg, aes(displ, hwy)) + geom\_point() + facet\_wrap(~manufacturer)

ggplotly()

# Lattice

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# O pacote Lattice é um sistema de visualização de dados

# de alto nível poderoso e elegante, com ênfase em dados

# multivariados.

# Na criação de gráficos, condições e agrupamentos são 2 conceitos

# importantes, que nos permitem compreender mais facilmente

# os dados que temos em mãos. O conceito por trás do Lattice

# é agrupar os dados e criar visualizaçãoes de forma que fique

# mais fácil a busca por padrões

install.packages('lattice')

library(lattice)

xyplot(data = iris, groups = Species, Sepal.Length ~ Petal.Length)

# ScatterPlot

splom(trees)

?Titanic

barchart(Class ~ Freq | Sex + Age, data = as.data.frame(Titanic),

groups = Survived, stack = T, layout = c(4, 1),

auto.key = list(title = "Dados Titanic", columns = 2))

barchart(Class ~ Freq | Sex + Age, data = as.data.frame(Titanic),

groups = Survived, stack = T, layout = c(4, 1),

auto.key = list(title = "Dados Titanic", columns = 2),

scales = list(x = "free"))

x = equal.count(rivers)

x

PetalLength <- equal.count(iris$Petal.Length, 4)

PetalLength

xyplot(Sepal.Length~Sepal.Width | PetalLength, data = iris)

xyplot(Sepal.Length~Sepal.Width | PetalLength, data = iris,

panel = function(...) {

panel.grid(h = -1, v = -1, col.line = "skyblue")

panel.xyplot(...)})

xyplot(Sepal.Length~Sepal.Width | PetalLength, data = iris,

panel = function(x,y,...) {

panel.xyplot(x,y,...)

mylm <- lm(y~x)

panel.abline(mylm)})

histogram(~Sepal.Length | Species, xlab = "",

data = iris, layout=c(3,1), type = "density",

main = "Lattice Histogram", sub = "Iris Dataset, Sepal Length")

qqmath(~ Sepal.Length | Species, data = iris, distribution = qunif)

# Boxplot

bwplot(Species~Sepal.Length, data = iris)

# ViolinPlot

bwplot(Species~Sepal.Length, data = iris,

panel = panel.violin)

par(mfrow=c(1,3))

cyls <- split(mtcars, mtcars$cyl)

for (ii in 1:length(cyls)) {

tmpdf <- cyls[[ii]]

sname <- names(cyls)[ii]

plot(tmpdf$wt, tmpdf$mpg,

main = paste("MPG vs Wt",sname,"Cyl"),

ylim = c(0,40), xlab = "Wt / 1,000",

ylab = "MPG", pch=19, col="blue")

grid()

}

# Mapas

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# ggmap

install.packages("ggmap")

install.packages("mapproj")

library(ggmap)

library(mapproj)

map1 <- get\_map(location = 'Brazil', zoom = 4)

map2 <- get\_map(location = 'Europe', zoom = 4)

ggmap(map1)

ggmap(map2)

?get\_map

?ggmap

options(warn=-1)

qmap('Recife')

qmap('L69 3GP', zoom = 16)

qmap('L69 3GP', zoom = 16, maptype = 'satellite')

# Estádio Olímpico de Londres

mapImageData1 <- get\_map(location = c(lon = -0.016179, lat = 51.538525),

color = "color",

source = "google",

maptype = "satellite",

zoom = 17)

ggmap(mapImageData1,

extent = "device",

ylab = "Latitude",

xlab = "Longitude")

mapImageData2 <- get\_map(location = c(lon = -0.016179, lat = 51.538525),

color = "color",

source = "google",

maptype = "hybrid",

zoom = 15)

ggmap(mapImageData2,

extent = "device",

ylab = "Latitude",

xlab = "Longitude")

# Maps

install.packages("mapdata")

install.packages("maps")

library(maps)

library(mapdata)

map("worldHires","Canada",

xlim=c(-141,-53),

ylim=c(40,85),

col="gray90",

fill=TRUE)

?map

help(package='maps')

# Gráficos de Rede, interativos

install.packages("networkD3")

library(networkD3)

data(MisLinks, MisNodes)

forceNetwork(Links = MisLinks, Nodes = MisNodes, Source = "source",

Target = "target", Value = "value", NodeID = "name",

Group = "group", opacity = 0.4)

?networkD3

library(networkD3)

# Criando dados

src <- c("A", "A", "A", "A",

"B", "B", "C", "C", "D")

target <- c("B", "C", "D", "J",

"E", "F", "G", "H", "I")

networkData <- data.frame(src, target)

# Plot

simpleNetwork(networkData)

# Heatmap

install.packages("heatmaply")

library(heatmaply)

heatmaply(mtcars, k\_col = 2, k\_row = 3) %>% layout(margin = list(l = 130, b = 40))

# Gráfico Interativo

install.packages("dygraphs")

library(dygraphs)

dygraph(nhtemp, main = "Temperaturas") %>%

dyRangeSelector(dateWindow = c("1920-01-01", "1960-01-01"))

# rMaps

# <http://rmaps.github.io>

# googleVis - Visualizações Interativas

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Baseado no Google Chart (visualização de dados na web)

# googlevis é um pacote que fornece interface entre R e o Google Charts

# Utiliza JavaScript e arquivos JSON. Os dados são transformados em formato JSON

# O resultado é gerado em HTML5 ou Flash

# Pose-se criar os mais variados tipos de gráficos e mapas. Inclusive Google Maps

# Você pode incorporar os gráficos criados em páginas HTML ou apps

# Verificar termos de uso, antes de utilizar

install.packages("googleVis")

library(googleVis)

?googleVis

# Criando um Datafrane

df = data.frame(País = c("BR", "CH", "AR"),

Exportações = c(10,13,14),

Importações = c(23,12,32))

# Gráfico de Linha

Line <- gvisLineChart(df)

plot(Line)

# Gráfico de Barras

Column <- gvisColumnChart(df)

plot(Column)

# Gráfico de Barras Horizontais

Bar <- gvisBarChart(df)

plot(Bar)

# Gráfico de Pizza

Pie <- gvisPieChart(CityPopularity)

plot(Pie)

# Gráficos Combinados

Combo <- gvisComboChart(df, xvar = "País",

yvar = c("Exportações", "Importações"),

options = list(seriesType = "bars",

series='{1: {type:"line"}}'))

plot(Combo)

# Scatter Chart

Scatter <- gvisScatterChart(women,

options=list(

legend="none",

lineWidth=2, pointSize=0,

title="Women", vAxis="{title:'weight (lbs)'}",

hAxis="{title:'height (in)'}",

width=300, height=300))

plot(Scatter)

# Bubble

Bubble <- gvisBubbleChart(Fruits, idvar="Fruit",

xvar="Sales", yvar="Expenses",

colorvar="Year", sizevar="Profit",

options=list(

hAxis='{minValue:75, maxValue:125}'))

plot(Bubble)

# Customizando

M <- matrix(nrow=6,ncol=6)

M[col(M)==row(M)] <- 1:6

dat <- data.frame(X=1:6, M)

SC <- gvisScatterChart(dat,

options=list(

title="Customizing points",

legend="right",

pointSize=30,

series="{

0: { pointShape: 'circle' },

1: { pointShape: 'triangle' },

2: { pointShape: 'square' },

3: { pointShape: 'diamond' },

4: { pointShape: 'star' },

5: { pointShape: 'polygon' }

}"))

plot(SC)

# Gauge

Gauge <- gvisGauge(CityPopularity,

options=list(min=0, max=800, greenFrom=500,

greenTo=800, yellowFrom=300, yellowTo=500,

redFrom=0, redTo=300, width=400, height=300))

plot(Gauge)

# Mapas

Intensity <- gvisIntensityMap(df)

plot(Intensity)

# Geo Chart

Geo=gvisGeoChart(Exports, locationvar="Country",

colorvar="Profit",

options=list(projection="kavrayskiy-vii"))

plot(Geo)

# Google Maps

AndrewMap <- gvisMap(Andrew, "LatLong" , "Tip",

options=list(showTip=TRUE,

showLine=TRUE,

enableScrollWheel=TRUE,

mapType='terrain',

useMapTypeControl=TRUE))

plot(AndrewMap)

# Dados em Gráfico. Nível de analfabetismo nos EUA

require(datasets)

states <- data.frame(state.name, state.x77)

GeoStates <- gvisGeoChart(states, "state.name", "Illiteracy",

options=list(region="US",

displayMode="regions",

resolution="provinces",

width=600, height=400))

plot(GeoStates)

# Mais exemplos

demo(googleVis)

# qplot

# \*\*\*\*\* Esta é a versão 2.0 deste script, atualizado em 23/05/2017 \*\*\*\*\*

# \*\*\*\*\* Esse script pode ser executado nas versões 3.3.1, 3.3.2, 3.3.3 e 3.4.0 da linguagem R \*\*\*\*\*

# \*\*\*\*\* Recomendamos a utilização da versão 3.4.0 da linguagem R \*\*\*\*\*

# Configurando o diretório de trabalho

# Coloque entre aspas o diretório de trabalho que você está usando no seu computador

setwd("~/Dropbox/DSA/BigDataAnalytics-R-Azure/Cap03")

getwd()

# Um sistema gráfico completo, alternativo ao sistema básico de gráficos do R

# Oferece mais opções de modificação, legendas prontas, formatação mais sólida

library(ggplot2)

x = runif(1000)

plot(x)

qplot(x)

?qplot

qplot(x, main = "Dist Uniforme 1000", ylab = 'y', xlab = 'x')

head(iris)

qplot(x = Sepal.Length, y = Sepal.Width, data = iris, color = Species, geom = "line")

qplot(x = Sepal.Length, y = Sepal.Width, data = iris, color = Species) + geom\_line()

qplot(x = Sepal.Length, y = Sepal.Width, data = iris, color = Species, geom = "point")

qplot(x = Sepal.Length, y = Sepal.Width, data = iris, color = Species, geom = "smooth", method = 'lm', se = F)

qplot(x = Sepal.Length, y = Sepal.Width, data = iris, color = Species, geom = "jitter")

# Histogramas

x=rnorm(10000)

options(warn=-1)

qplot(x, geom="histogram")

qplot(x, geom="histogram", binwidth=0.5, xlim=c(-3,3))

qplot(x, geom="histogram", breaks=c(-3,-1.5,-1,-0.5,0,1.5,2,3), xlim=c(-3,3))

qplot(x, geom="density")

qplot(Sepal.Length, data=iris, geom="density", colour=Species)

qplot(Sepal.Length, data=iris, geom="density", facets=Species~. )

qplot(Sepal.Length, data=iris, geom="density", facets=.~Species )

qplot(x=Sepal.Length, y=Sepal.Width, data=iris, color=Species)

qplot(x=Sepal.Length, y=Sepal.Width, data=iris, size=Species)

qplot(x=Sepal.Length, y=Sepal.Width, data=iris, shape=Species)

qplot(x=Sepal.Length, y=Sepal.Width, data=iris, alpha=Species)

qplot(x=Sepal.Length, y=Sepal.Width, data=iris, color=Species, size=Species, shape=Species)

# ggplot

mydata <- data.frame(x=1:5, y=11:15, group=c("F", "F", "M", "M", "F"))

mydata

?ggplot

p <- ggplot(data=mydata, aes(x=x,y=y))

p+geom\_point()

p+geom\_point(aes(color=group), shape=3)

p+geom\_point(aes(color=group), shape=3) + scale\_color\_manual(values=c(123,569))

p+geom\_point() + scale\_x\_continuous(limits=c(0,10))