Aula 5 - Manipulação de dados, análise exploratória

Vitor Rios

11 de novembro de 2017

Função subset()

Divide um objeto em subconjuntos

`x` = objeto a ser dividido
`subset` = condição lógica indicando o que deve ser mantido
`select`= indica as colunas a serem selecioandas em um dataframe

```
dragoes_completo <- read.csv("..//..//arquivos/dragoes_completo.csv", sep=
vermelhos.grandes = subset(
  x=dragoes_completo
  ,subset = tamanho_asa >= 15 & cor == "vermelho" #subconjunto somente com
  , select=n_chifres:idade #manter apenas a coluna n_chifres
  )
vermelhos.grandes
```

```
##    n_chifres      cor tamanho_asa    idade
## 48         9 vermelho    16.35738 146.2257
## 60         5 vermelho    16.95939 194.3464
```

Familia apply()

Existem várias funções no R que aplicam funções sobre objetos, e elas variam principalmente no tipo de objeto que devolvem

Função apply()

Aplica uma funçao a todas as linhas ou todas as colunas de um array (dataframe numérico ou matriz), retorna um vetor ou matriz *ATENÇÃO* : só funciona para dados numéricos, não serve para fatores

`apply(X, MARGIN, FUN, ...)`

`x` é array que ser quer agrupar

`MARGIN` 1 = linhas, 2 = colunas, c(1,2), ambos

`FUN` é a função que se quer aplicar nas margens, sem parênteses

`...` são os argumentos que serão passados para a função `FUN`

Por exemplo, se queremos a soma de cada coluna

```
numericos= data.frame(dragoes_completo$peso,dragoes_completo$n_chifres,drag
somas = apply(numericos, MARGIN = 2, FUN =  sum, na.rm=T)
```

#Função lapply() Aplica uma função a cada elemento de uma lista, e retorna uma lista. Pode lidar com qualquer tipo de dado, a depender de FUN

`lapply(X, FUN, ...)`

`X` um lista ou objeto que será convertido em lista (pode ser data.frame ou matriz)

`FUN` é a função que se quer aplicar nos elementos de, sem parênteses

`...` são os argumentos que serão passados para a função `FUN`

```
lista.somas = lapply(numericos, FUN =  sum, na.rm=T)
lista.somas
```

```
## $dragoes_completo.peso
## [1] 10377.68
##
## $dragoes_completo.n_chifres
## [1] 560
##
## $dragoes_completo.tamanho_asa
## [1] 890.0223
##
## $dragoes_completo.idade
```

#Função sapply() Aplica uma função a cada elemento de uma lista, e retorna uma vetor com os resultados. Pode lidar com qualquer tipo de dado, a depender de `FUN`

`lapply(X, FUN, ...)`

`X` um lista ou objeto que será convertido em lista (pode ser data.frame ou matriz)

`FUN` é a função que se quer aplicar nos elementos de `X`, sem parênteses

`...` são os argumentos que serão passados para a função `FUN`

```
vetor.somas = sapply(numericos, FUN =  sum, na.rm=T)
vetor.somas
```

```
##        dragoes_completo.peso   dragoes_completo.n_chifres
##                   10377.6809                     560.0000
## dragoes_completo.tamanho_asa       dragoes_completo.idade
##                     890.0223                    9069.2594
```

```
vetor.log = sapply(numericos, FUN =  log)
```

#Função tapply() `tapply(X, INDEX, FUN, ...)`
Aplica uma função a subsets do objeto
`X` = um objeto, tipicamento um vetor
`INDEX` uma lista de fatores, com comprimento igual a `X`, usado para criar subconjuntos nos quais `FUN` será aplicada
`FUN` é a função que se quer aplicar nos elementos de `X`, sem parênteses
`...` são os argumentos que serão passados para a função `FUN`

```
x <- 1:20
y <- factor(rep(letters[1:5], each = 4))
tapply(x, INDEX=y, FUN = sum)
```

```
##  a  b  c  d  e
## 10 26 42 58 74
```

```
media.por.cor2=tapply(dragoes_completo$n_chifres, INDEX = dragoes_completo$
class(media.por.cor2)
```

```
## [1] "array"
```

Para facilitar:

- apply: genérica: aplica uma função a linhas ou colunas de uma matriz (ou às dimensões de um array), retorna vetor ou matriz
- lapply: “list apply”. Age em uma lista ou vetor e retorna uma lista
- sapply: “simple lapply”. Igual a lapply, mas retorna um vetor ou matriz sempre que possível
- tapply: “tagged apply”. subconjuntos (tags) identificam os grupos nos quais a função será aplicada. Tipo de retorno depende da função, geralmente array
- aggregate: tapply que converte o resultado para dataframe

Na maioria dos casos você vai usar aggregate ou tapply

Análise exploratória: verificando seus dados

Antes de qualquer análise estatística, é necessário verificar a distribuição dos dados, se há outliers, se a distribuição é normal, assimétrica, se há dados ausentes, erros de digitação, se as premissas dos testes são cumpridas, se é necessário transformar os dados, etc...

Antes de tudo, verifique a estrutura dos dados, NAs e erros de digitação

```
str(dragoes_completo)
```

```
## 'data.frame':    80 obs. of  7 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ dieta      : Factor w/ 4 levels "aventureiros",..: 3 3 3 3 3 3 3 3 3
##  $ peso       : num  121.7 103.8 130.2 98.3 103.4 ...
##  $ n_chifres  : int  9 8 8 8 4 8 7 12 4 3 ...
##  $ cor        : Factor w/ 10 levels "azul","banco",..: 3 8 6 9 3 9 1 3 9
##  $ tamanho_asa: num  15.95 4.31 10.23 13.3 7.12 ...
##  $ idade      : num  140.84 7.64 66.4 149.28 85.02 ...
```

```
head(dragoes_completo)
```

```
##   X dieta      peso n_chifres      cor tamanho_asa      idade
## 1 1 vacas 121.72355         9   branco   15.950930 140.837112
## 2 2 vacas 103.79754         8    verde    4.305912   7.640123
## 3 3 vacas 130.15442         8    preto   10.229693  66.399862
## 4 4 vacas  98.29305         8 vermelho   13.304952 149.276811
## 5 5 vacas 103.43365         4   branco    7.119846  85.021320
## 6 6 vacas 102.44998         8 vermelho    6.777281  43.396197
```

```r
sum(is.na(dragoes_completo))
```

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

```
summary(dragoes_completo)
```

```
##        X                  dieta         peso          n_chifres
##  Min.   : 1.00   aventureiros:20   Min.   : 65.10   Min.   : 1
##  1st Qu.:20.75   fazendeiros :20   1st Qu.: 82.74   1st Qu.: 5
##  Median :40.50   vacas       :20   Median :125.50   Median : 7
##  Mean   :40.50   virgens     :20   Mean   :131.36   Mean   : 7
##  3rd Qu.:60.25                     3rd Qu.:169.68   3rd Qu.: 9
##  Max.   :80.00                     Max.   :216.17   Max.   :13
##                                    NA's   :1
##        cor       tamanho_asa         idade
##  verde   :17   Min.   : 3.111   Min.   :  3.421
##  branco  :15   1st Qu.: 9.427   1st Qu.: 77.844
##  vermelho:13   Median :11.081   Median :115.519
##  azul    :11   Mean   :11.125   Mean   :113.366
##  preto   :11   3rd Qu.:13.431   3rd Qu.:142.198
##  dourado : 9   Max.   :17.244   Max.   :217.578
##  (Other) : 4
```

Precisamos remover a linha com NA e os erros de digitação

```
unique(dragoes_completo$cor)
```

```
##  [1] branco   verde    preto    vermelho azul     vremelho dourado
##  [8] banco    Preto    dorado
## 10 Levels: azul banco branco dorado dourado preto Preto verde ... vreme
```

```
dragoes_limpo = dragoes_completo #copie seus dados para um outro objeto
dragoes_limpo=dragoes_limpo[,-1] #coluna 1 é inútil
dragoes_limpo$cor[dragoes_limpo$cor =="dorado" ] = "dourado"
dragoes_limpo$cor[dragoes_limpo$cor =="vremelho" ] = "vermelho"
dragoes_limpo$cor[dragoes_limpo$cor =="banco" ] = "branco"
dragoes_limpo$cor = tolower(dragoes_limpo$cor) # maiúsculas para minuscula
unique(dragoes_limpo$cor)
```

```
## [1] "branco"   "verde"    "preto"    "vermelho" "azul"     "dourado"
```

```
dragoes_limpo$cor = factor(dragoes_limpo$cor)#transformando novamente em f
dragoes_limpo = dragoes_limpo[!is.na(dragoes_limpo$peso),]
unique(dragoes_limpo$cor)
```

summary() para estatísticas básicas

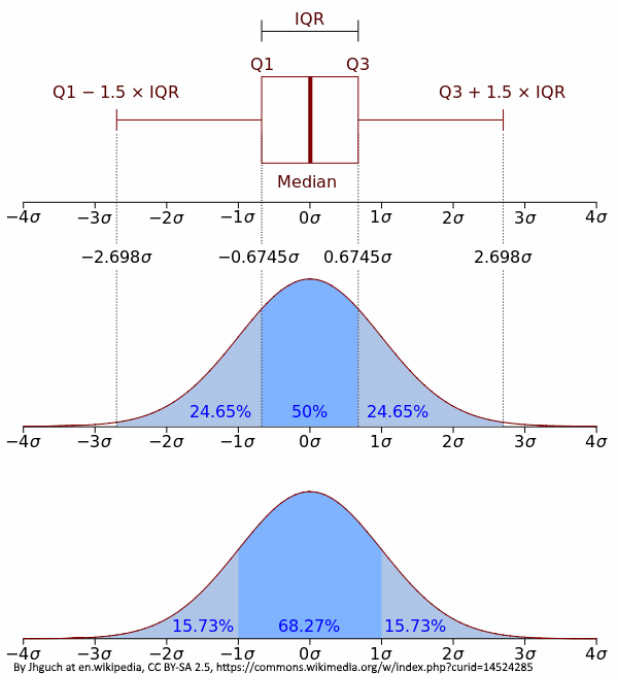
```
summary(dragoes_limpo)
```

```
##           dieta         peso           n_chifres            cor
##  aventureiros:20   Min.   : 65.10   Min.   : 1.000   azul    :11
##  fazendeiros :20   1st Qu.: 82.74   1st Qu.: 5.000   branco  :16
##  vacas       :20   Median :125.50   Median : 7.000   dourado :10
##  virgens     :19   Mean   :131.36   Mean   : 6.975   preto   :11
##                    3rd Qu.:169.68   3rd Qu.: 9.000   verde   :17
##                    Max.   :216.17   Max.   :13.000   vermelho:14
##   tamanho_asa         idade
##  Min.   : 3.111   Min.   :  3.421
##  1st Qu.: 9.385   1st Qu.: 77.825
##  Median :11.059   Median :114.258
##  Mean   :11.103   Mean   :113.240
##  3rd Qu.:13.435   3rd Qu.:143.541
##  Max.   :17.244   Max.   :217.578
```

Essas estatísticas são suficientes? Para cada coluna, summary nos dá

- Min. : valor mínimo dos dados
- 1st Qu. : primeiro quartil
- Median : mediana
- Mean : média
- 3rd Qu. : terceiro quartil
- Max. : valor máximo dos dados
- NA's : quantidade de NAs nos dados

O que falta?



Para conseguirmos outras estatísticas, usamos as funções da família apply

```
sds = lapply(X = dragoes_limpo,FUN = sd ,na.rm=T ) #sd não funciona para fa

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na
##   Use something like 'all(duplicated(x)[-1L])' to test for a constant ve

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na
##   Use something like 'all(duplicated(x)[-1L])' to test for a constant ve
sds

## $dieta
## [1] 1.119303
##
## $peso
## [1] 47.24073
##
## $n_chifres
## [1] 2.684116
```

também podemos usar funcões que nós mesmo escrevemos

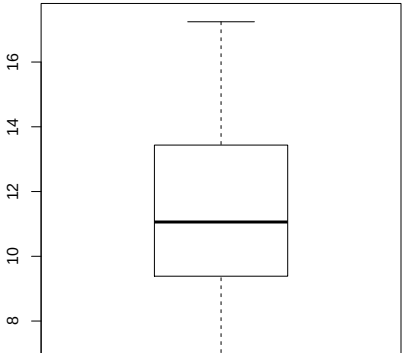
```
erro.padrao.media = function(x) {
  #erro padrão da média, é igual ao desvio padrão dividido pela raiz do nú
                                    sd(x)/sqrt(length(x))
}
erros = aggregate(dragoes_limpo$idade
                  ,by=list(dragoes_limpo$dieta)
                  ,FUN = erro.padrao.media
                                  )

erros
```

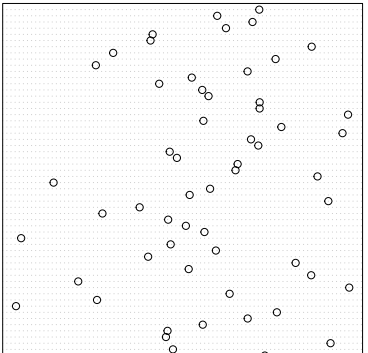
```
##        Group.1         x
## 1 aventureiros 10.711285
## 2  fazendeiros  9.928882
## 3        vacas 11.602347
## 4      virgens 11.599718
```

Detectando outliers: Cleveland dotplot

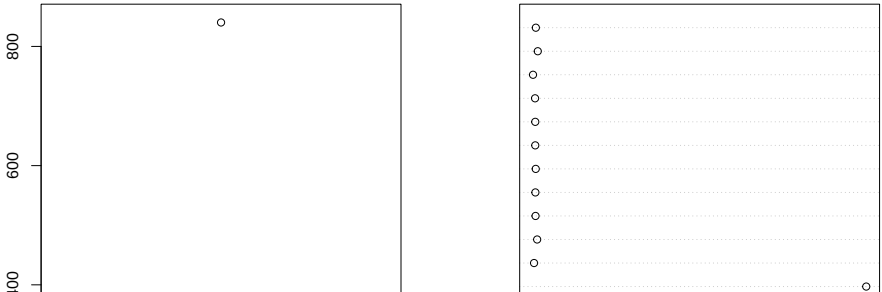
Arruma os pontos de acordo com o valor (eixo x), e a ordem nos dados (eixo y)

```
par(mfrow=c(1,2))
boxplot(dragoes_limpo$tamanho_asa)
dotchart(dragoes_limpo$tamanho_asa)
```





```
fazendeiros = c(77.91352, 78.07251, 81.95604, 75.64862, 78.45213, 79.11058
                79.98952, 79.18127, 840.1635, 74.86860, 82.01886, 78.26936
                77.94691, 78.75372, 77.64901, 77.64097, 77.19803, 72.48175
                83.45336, 78.99681
)
par(mfrow=c(1,2))
boxplot(fazendeiros)
dotchart(fazendeiros)
```



Homogeneidade da variância

Teste de Bartlett ou de Levene (mais robusto, pacote car)

```
library(car)
```

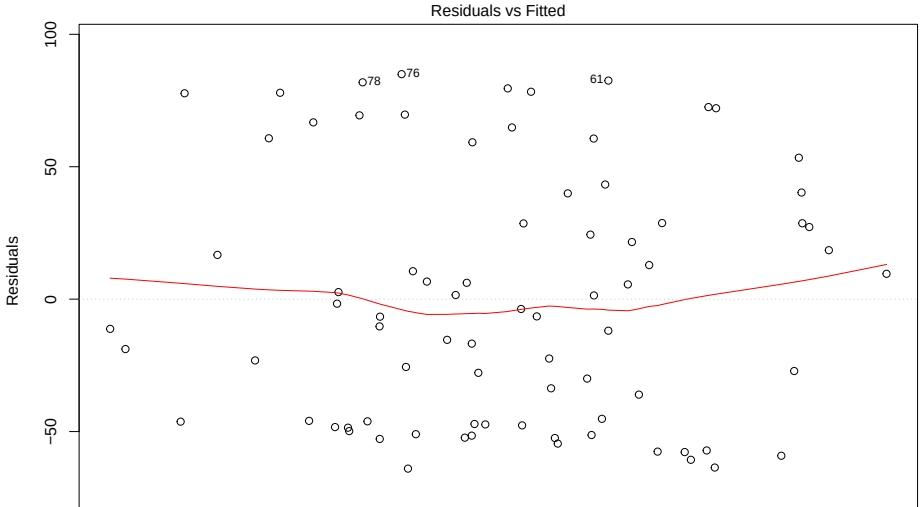
```
## Warning: package 'car' was built under R version 3.4.3
```

```
leveneTest(dragoes_limpo$peso ~dragoes_limpo$dieta)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value    Pr(>F)
## group  3  7.0987 0.0002912 ***
##       75
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

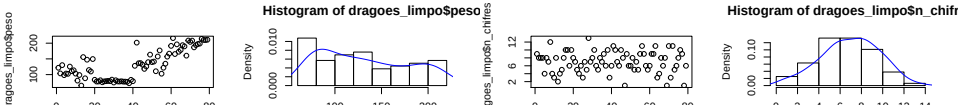
ou observando os resíduos

```
plot(lm(dragoes_limpo$peso ~dragoes_limpo$idade))
```



Premissas de normalidade

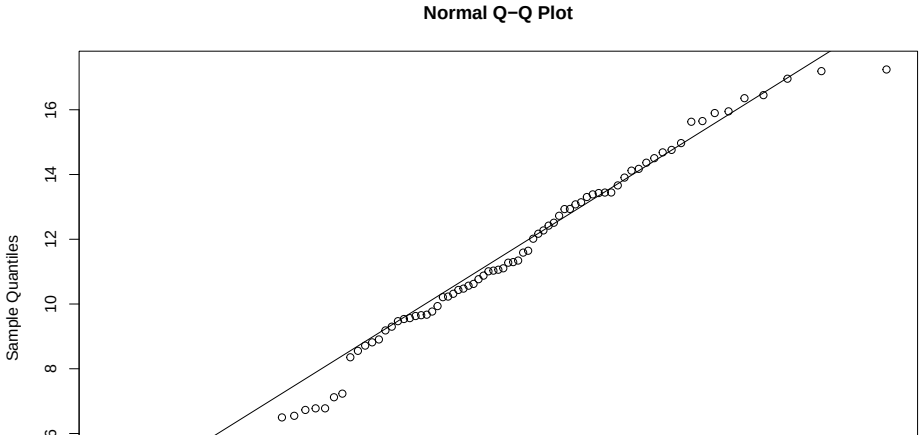
```
par(mfrow = c(4,4))
plot(dragoes_limpo$peso)
hist(dragoes_limpo$peso, prob=T)
lines(density(dragoes_limpo$peso),col="blue")#densidade probabilística dos
plot(dragoes_limpo$n_chifres)
hist(dragoes_limpo$n_chifres, prob=T)
lines(density(dragoes_limpo$n_chifres),col="blue")#densidade probabilística
plot(dragoes_limpo$tamanho_asa)
hist(dragoes_limpo$tamanho_asa, prob=T)
lines(density(dragoes_limpo$tamanho_asa),col="blue")#densidade probabilísti
plot(dragoes_limpo$idade)
hist(dragoes_limpo$idade, prob=T)
lines(density(dragoes_limpo$idade),col="blue")#densidade probabilística dos
```



qqnorm() é uma investigação rápida de normalidade da variável

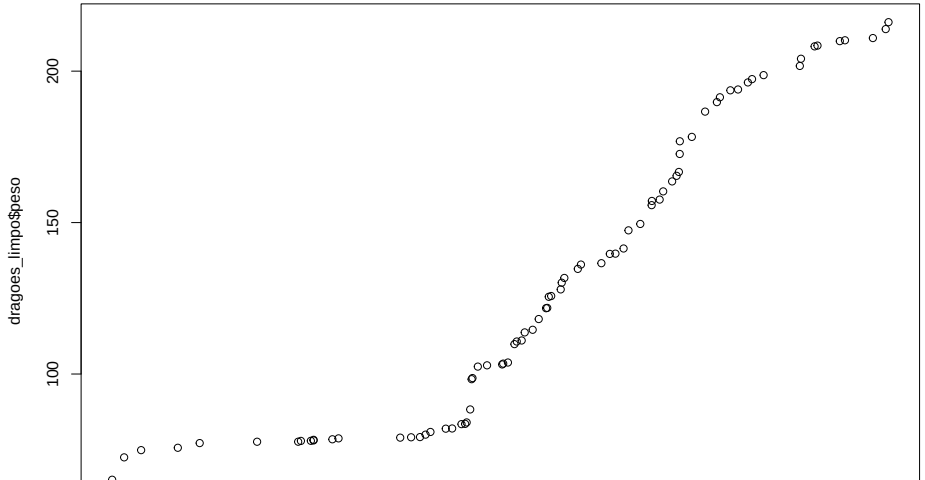
qqnorm()

```
qqnorm(dragoes_limpo$tamanho_asa) #plota os dados contra uma distribuição n
qqline(dragoes_limpo$tamanho_asa) #plota uma linha para facilitar a compara
```



qqplot() compara duas distribuições qualquer

```
qqplot(dragoes_limpo$tamanho_asa,dragoes_limpo$peso)
```



Funções matemáticas para gerar distribuições de dados

Usadas para gerar dados artificais com as propriedades desejadas. O R possui as seguintes distribuições por padrão: beta, binomial, Cauchy, qui-quadrado, exponencial, F, gamma, geométrica, hipergeométrica, log-normal, multinomial, binomial negativa, normal, Poisson, t de Student, uniforme, Weibull

```
#distribuição normal
rnorm(n, mean = 0, sd = 1) #gera n observações amostradas de uma normal de
dnorm(3, mean = 0, sd = 1, log = FALSE)# retorna a probabilidade de um valo
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE) # retorna a pr
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE) # retorna o va
```