Universidade Federal da Bahia
Instituto Multidisciplinar em Saúde
Curso de extensão: Introdução ao uso do programa R
Prof. Leandro Martins de Freitas e Profª Leila Cruz

Introdução ao uso do programa R

Dia 2: Fundamentos da Linguagem R

9. Operadores de Comparação - Comparison operators

9.1. a == b # igualdade, retorna TRUE se os dois termos forem iguais, retorna FALSE se os dois termos forem diferentes.

```
>10 == 11
>"Normal" == "Normal"
>vetor_num <- c(1,10,2,3,5,2,10)
>vetor_num == 2
>vetor_char <-
c("Normal","Cancer","Normal","Normal","Cancer","Normal")
>vetor_char == "Normal"
```

9.2. a != b # diferente, retorna TRUE se os dois termos forem diferentes, retorna FALSE se os dois termos forem iguais.

```
>10 != 5
>10 != 10
>vetor_num != 10
>vetor_char != "Cancer"
```

9.3. a < b #menor, retorna TRUE se o termo `a` é menor que `b`.

```
>20 < 5
>20 < 21
>vctor_num > 5
```

9.4. a > b #maior, retorna TRUE se o termo `a` é maior que `b`.

```
>15 > 10+4
>2 > 4-1
>vetor_num*5 >  10
```

9.5. a <= b #menor ou igual, retorna TRUE se o termo `a` é menor ou igual a `b`.

```
>4 <= 4
>4 <= 6
```

```
>vetor_num <= 4
```

9.6. a >= b #maior ou igual, retorna TRUE se o termo `a` é maior ou igual a `b`.

```
>3 >=0
>3 >= NaN
>vetor_num-4 >= 5
```

10. Diretório de trabalho

O R possui um diretório de trabalho, onde devemos colocar os arquivos que serão analisados e onde serão salvos os arquivos de saida e gráficos. Para salvar os arquivos e gráficos devemos usar funções especificas para cada tipo.

Podemos fazer leitura dos arquivos que estão nesse diretório de trabalho e carregar esses arquivos no programa.

10.1. Para saber qual é o diretório de trabalho devemos usar a função `getwd()`.

```
>getwd()
```

10.2. Podemos mudar o diretório de trabalho usando a função `setwd()`. Para cada conjunto de dados podemos reservar um diretório e passar a informação do caminho do diretório para o R. Isso facilita carregar os arquivos e salvar os resultados das análises.

```
>setwd()
#isso é só um exemplo de caminho para o diretório , você deve usar o
caminho que escolher no seu computador
>setwd("C:/Documents and Settings/leandro/Documents/curso_R_2015")

#veja se o agora você está no diretório desejado
>getwd()
```

11. Arquivos existentes no diretório

```
>dir()
```

11.1. Dados

11.1.1. Leitura

Existem funções que fazem a leitura de arquivos e importam os dados para uma variável do R. Vamos usar aqui a função `read.csv()`.

`read.csv()`

file: nome_do_arquivo.csv.
header: se o arquivo apresenta nome de colunas. Padrão é FALSE.
sep: qual caracter é usado para separar os campos.
dec: separador decimal. Padrão ".".

row.names: vetor com o nome das linhas ou número indicando qual coluna do arquivo possui os nomes das linhas.

```
>respostas <- read.csv(file=respostas.csv, header = T, sep=";",
row.names=1)
>dim(respostats)
>head(respostas)
```

11.1.2. Escrita

Existem funções que fazem a gravação dos dados de uma variável em arquivos. Vamos usar aqui a função `writre.csv()`.

`write.csv()`

x: objeto (variável) que será salvo em arquivo.

file: nome_do_arquivo.csv.

sep: qual caracter é usado para separar os campos.

row.names: valores logicos (T ou F) informando se o nome das linhas devem ser gravados. Pode ser passado também um vetor com o nome das linhas.

col.names: valores logicos (T ou F) informando se o nome das colunas devem ser gravados. Pode ser passado também um vetor com o nome das colunas.

dec: separador decimal. Padrão, ponto ".".

```
>write.csv(x= data_exemplo, file= "data_exemplo.csv", sep=";",
row.names=T, col.names=T)
```

12. Subsetting Data

O *Subsetting Data* é a criação de um subconjunto contendo elementos e variáveis do nosso interesse. Frequentemente podemos remover dados que não nos interessam de uma base de dados e análisar somente uma parte.

Vamos ver exemplos de como fazer *Subsetting Data* considerando observações nas linhas e variáveis nas colunas

	var1	var2	va3
Obser1	0.31	10.4	Pop1
Obser2	0.35	9.0	Pop2
Obser3	0.3	9.5	Pop1

12.1. Removendo observações (linhas)

Para fazer um *Subsetting Data* de observações devemos indicar na primeira parte do `[,]` quais elementos queremos selecionar.

#exemplo de como usar o Subsetting Data de linhas

```
>data_base[linhas,]

#ATENÇÃO
#carregar os dados ChickWeight.csv em uma variável chamada ChickWeight
#carregar os dados chickwts.csv emu ma variável chamada chickwts
#usar função read.csv()
```

Vamos fazer um novo `data.frame` com somente as observações que possuam dieta do tipo 3 e 4. Essa informação se encontra na variável Diet que possui 4 tipos diferentes de dietas `c(1,2,3,4)`.

```
#criação de um vetor com valores logicos, contendo quais observações
possuem Diet igual a 3 ou 4
>dieta_3_4 <- ChickWeight$Diet == c(3,4)
>sum(dieta_3_4)

#uso do vetor criado para fazer o Subsetting Data
>ChickWeight_diet_3_4 <- ChickWeight[dieta_3_4,]
>dim(ChickWeight_diet_3_4)#para saber as dimenssões de uma matriz ou
data frame usamos a função dim().
```

Vamos fazer um novo data.frame com somente as observações que possuam tempo menor ou igual a 10. Essa informação se encontra na variável Time que possui 12 tipos diferentes de tempos de amostragem do peso `c(0,2,4,6,8,10,12,14,16,18,20,21)`.

```
#criação de um vetor com valores logicos, contendo quais observações
possuem tempo menor ou igual 10
>time_0_10 <- ChickWeight$Time <= 10
>sum(time_0_10)
#uso do vetor criado para fazer o Subsetting Data
>ChickWeight_time_0_10 <- ChickWeight[time_0_10,]
>dim(ChickWeight_time_0_10)#para saber as dimenssões de uma matriz ou
data frame usamos a função dim().
```

Podemos combinar informações de duas ou mais variáveis para fazer o Subsetting Data

Vamos fazer um novo `data.frame` com somente as observações que possuam dieta do tipo 3 e 4 e que possuam tempo menor ou igual a 10.

Já fizemos dois vetores com valores logicos, com informação de quais elementos tem dieta do tipo 3 ou 4 e outro com informação de tempo de amostragem menor ou igual a 10. Vamos agora combinar as informações desses dois vetores.

```
#aqui usamos o simbolo & que significa que o valor deve ser igual
(TRUE) nos dois vetores para ser considerado verdadeiro.
>time_0_10_dieta_3_4 <- (dieta_3_4 & time_0_10)
>ChickWeight_time_0_10_dieta_3_4<-  ChickWeight[time_0_10_dieta_3_4,]
>dim(ChickWeight_time_0_10_dieta_3_4) #para saber as dimenssões de uma
matriz ou data frame usamos a função dim().
```

12.2. Removendo variaveis (colunas)

Vamos fazer um novo `data.frame` com somente as variáveis peso (weight), colunas 1

```
>ChickWeight_tempo <- ChickWeight[,1]
>length() # para saber o comprimento de um vetor usamos a função
>length(ChickWeight_tempo)
>head(ChickWeight_tempo)
```

Vamos fazer um novo `data.frame` com somente as variáveis tempo (Time), colunas 2

```
>ChickWeight_tempo <- ChickWeight[,2]
>length(ChickWeight_tempo) # para saber o comprimento de um vetor
usamos a função length()
>head(ChickWeight_tempo)
```

Vamos fazer um novo `data.frame` com somente as variáveis peso (weight) e tempo (Time), colunas 1,2

```
>ChickWeight_peso_tempo <- ChickWeight[,c(1,2)]
>dim(ChickWeight_peso_tempo)
>head(ChickWeight_peso_tempo)
```

O Subsetting Data pode ser feito combinando linhas e colunas.

Vamos usar o exemplo das observações com Diet do tipo 3 ou 4 e selecionar somente as colunas (variáveis) peso e tempo.

```
>ChickWeight_diet_3_4_peso_tempo <- ChickWeight[dieta_3_4,c(1,2)]
>dim(ChickWeight_diet_3_4_peso_tempo)
```

#novo `data.frame` com somente peso(weight) e alimentação (Diet), colunas 1,4

```
>ChickWeight_peso_alim <- ChickWeight[,c(1,4)]
>head(ChickWeight_peso_alim)
```

Tarefas

#armazenar em um vetor somente os dados da coluna 1 do arquivo `ChickWeight.csv`

#armazenar em um data frame somente os dados das colunas 1,2,3 do arquivo `ChickWeight.csv`

#armazenar em um data frame somente os dados dos dias 0 até o dia 6 do arquivo `ChickWeight.csv`

#armazenar em um data frame somente os dados de peso dos dias 8 até o dia 21 do arquivo

#armazenar em um data frame somente os dados dos dias 8 até o dia 14 do arquivo

Exercício 1. Faça o *Subsetting Data* da variavel peso (weight) do dataset chickwts.csv usando como subsetting cada um dos tipos de alimentação (feed)
Exercício 2. Qual a média (mean) de peso (weight) para cada um dos subset produzidos?
Exercício 3. Qual o desvio padrão (sd) de peso(weight) para cada um dos subset produzidos?
Exercício 4. Qual a diferença da média de peso (weight) de cada subset e a média do peso total? Faça a diferença das médias usando programação R.

13. Dados qualitativos

Distribuição de Frequência para Variável Qualitativa

A distribuição de frequência é um sumário da ocorrência dos dados em categorias que não apresentam sobreposição.

Podemos apresentar essa frequência na forma de tabela ou graficamente.

Vamos investigar a distribuição de frequências de variáveis tipo "*factor*". Poderíamos também realizar essa investigação usando variáveis do tipo "*intenger*".

#exemplos de frequência de distribuição

```
>head(ChickWeight)
>lapply(ChickWeight, class)
```

```
#distribuição de frequência da variável Diet
>table(ChickWeight$Diet)
#distribuição de frequência da variável Chick
>table(ChickWeight$Chick)

#busca por qual o valor minimo da variável dieta
>var1 <- table(ChickWeight$Diet)
#comparação de valores da var1 com o minimo da var1, gerando vetor
logico
>elementos1 <- var1 == min(var1)
#uso do vetor lógico para retornar somente o valor minimo
>var1[elementos1]
```

#exemplo 2

```
>head(chickwts)
>lapply(chickwts, class)
>table(chickwts$feed)

#busca por qual o valor minimo da variável dieta
>var2 <- table(chickwts$feed)
#comparação de valores da var2 com o máximo da var2, gerando vetor
lógico
>elementos_max <- var2 == min(var2)
#uso do vetor lógico para retornar somente o valor minimo
>var2[elementos_max]
```

```
#carregar os dados esoph.csv em uma variável chamada esoph
#usar função read.csv()
```

Exercício 1. Quais tipos de variáveis existem no `dataset esoph`?

Exercício 2. Quais variáveis são qualitativas (`dataset esoph`)?

Exercíco 3. Encontrar a frequência absoluta das variáveis qualitativas do `data.frame (dataset esoph).`

Exercício 4. Qual a classe e quantidade de casos nas classe mais frequentes em cada variável `(dataset esoph)`?

Exercício 5. Transformar cada resultado da frequência absoluta em uma matrix usando a função `cbind() (dataset esoph)`.

Exercício 6. Encontrar quais classes apresentam o maior valor para cada variável usando programação `(dataset esoph)`.

Exercício 7. Encontrar quais classes apresentam valor menor ou igual a média para cada variável usando programação `(dataset esoph)`.

---

14. Distribuição de Frequência Relativa para Variável Qualitativa

A frequência relativa é a frequência absoluta de cada classe dividida pelo tamanho da amostra.

```
#exemplos de frequência de distribuição relativa
>head(ChickWeight)
>lapply(ChickWeight, class)
>var3 <- table(ChickWeight$Diet)
>freq_relativa_var3 <- var3/sum(var3)

#melhorando a apresentação
#melhorando a apresentação menor numero de casas
>round(freq_relativa_var3,digits=2)

#melhorando a apresentação em porcentagem
>round(freq_relativa_var3,digits=2)*100


#exemplo 4
>head(chickwts)
>lapply(chickwts, class)
>var4 <- cbind(table(chickwts$feed))
>freq_relativa_var4 <- var4/sum(var4)
>freq_relativa_var4

#melhorando a apresentação menor numero de casas
>round(freq_relativa_var4,digits=2)

#melhorando a apresentação em porcentagem
>round(freq_relativa_var4,digits=2)*100
```

Exercíco 8. Encontrar a frequência relativas das variáveis qualitativas do `data.frame (dataset esoph)`.

Exercício 9. Qual a classe e quantidade relativa de casos nas classe mais frequentes em cada variável `(dataset esoph)`?

Exercício 10. Encontrar quais classes apresentam valor de frequencia relativa maior, para cada variável usando programação `(dataset esoph)`.

Exercício 11. Encontrar quais classes apresentam valor de frequencia relativa menor ou igual a média para cada variável, usando programação `(dataset esoph)`.

---

15. Qualitativa - Gráficos

Vamos usar aqui os gráficos de pizza e barras para fazer representações das frequências de distribuição.

```
#carregar os dados ChickWeight.csv
>freq_diet <- table(ChickWeight$Diet)
>barplot(freq_diet)

#mudando o rotulo do eixo y
>barplot(freq_diet, ylab="freq.")

#mudando o rotulo de cada barra no gráfico
>barplot(freq_diet, names.arg=c("dieta 1","dieta 2","dieta 2","dieta
3"))

#outras configurações
>barplot(freq_diet,main="Frequência do tipo de alimentação")

#mudando o limite do eixo Y
>barplot(freq_diet,main="Frequência do tipo de alimentação",
ylim=c(0,250))
>barplot(freq_diet,main="Frequência do tipo de alimentação",
ylim=c(0,250), col="lightblue")

#mudando a cor
>barplot(freq_diet,main="Frequência do tipo de alimentação",
ylim=c(0,250), col="lightblue", space=0.1)

#mudando o espaço entre as barras
```

```
>barplot(freq_diet,main="Frequência do tipo de alimentação",
ylim=c(0,250), col="lightblue", space=0.3, names.arg=c("milho",
"soja", "farelo", "mistura"))

#carregar os dados chickwts.csv
>freq_feed <- table(chickwts$feed)
>barplot(freq_feed)

#colocando em ordem crescente
>barplot(sort(freq_feed))

#mudando a cor
>barplot(sort(freq_feed), col = "beige")

#adicionando, rotulo x (alimendação), título(Avaliação da alimentação)
e subtítulo (tratamento 1)
>barplot(sort(freq_feed), col = "beige", xlab="alimendação", ylab =
"freq. ", main="Avaliação da alimentação", sub="tratamento 1")

#diminuindo o tamanho do texto
>barplot(sort(freq_feed), col = "beige", ylab = "freq.",
main="Avaliação da alimentação", cex.names=0.8, cex.axis=0.8, las=0)

#mudando a configuração dos números no eixo x e y
>barplot(sort(freq_feed), col = "beige", ylab = "freq.",
main="Avaliação da alimentação", cex.names=0.8, cex.axis=0.8, las=1)
>barplot(sort(freq_feed), col = "beige", ylab = "freq.",
main="Avaliação da alimentação",  cex.names=0.8, cex.axis=0.8, las=2)
>barplot(sort(freq_feed), col = "beige", , ylab = "freq.",
main="Avaliação da alimentação",  cex.names=0.8, cex.axis=0.8, las=3)

#mudando a orientação
>barplot(sort(freq_feed), col = "beige", , xlab = "freq.",
main="Avaliação da alimentação", cex.names=0.8, cex.axis=0.8, las=1,
horiz=T)

#frequencia relativa
>freq_relativa_var_resp4 <- freq_diet/sum(freq_diet)
>barplot(freq_relativa_var_resp4)

#a diferença no gráfico da absoluta e relativa vai aparecer no eixo y
>barplot(freq_diet)
>freq_relativa_var_resp3 <- freq_feed/sum(freq_feed)

#relativa
```

```
>barplot(freq_relativa_var_resp3)

#absoluta
>barplot(freq_feed)
```

Apresentando frequência relativa usando gráfico de pizza

#gráfico pizza - pie()

```
>pie(freq_diet)
#mudando a cor
>pie(freq_diet, col=c("red","blue", "darkgreen", "magenta"))

#mudando os rotulos
>pie(freq_diet,labels=c("dieta 1","dieta 2","dieta 2","dieta 3"),
col=c("red","blue", "darkgreen", "magenta"))

#titulo
>pie(freq_diet,labels=c("dieta 1","dieta 2","dieta 2","dieta 3"),
col=c("red","blue", "darkgreen", "magenta"), main="Frequência da
dieta")

#usando um pacote para fazer pizza 3D
# 3D Exploded Pie Chart
>install.packages(pkgs="plotrix", dependencies=T)
>library(plotrix)
>pie3D(freq_diet,labels=c("dieta 1","dieta 2","dieta 2","dieta 3"),
       col=c("red","blue", "darkgreen", "magenta"),
       main="Tipo de dieta", explode=0.05)

#mudando a distância entre as fatias
>pie3D(freq_diet,labels=c("dieta 1","dieta 2","dieta 2","dieta 3"),
      col=c("red","blue", "darkgreen", "magenta"),
      main="Tipo de dieta", explode=0.2)
```

Desafio

Encontrar quais são as variáveis qualitativas, frequência absoluta, frequência relativa (somente duas casa depois do ".") , fazer gráficos de barra e pizza para os dataset abaixos:

```
>quakes.csv
>ToothGrowth.csv
>volcano.csv
>BOD.csv
```

```
>cars.csv
>CO2.csv
>DNase.csv
>ability.cov.csv
>airquality.csv
>PlantGrowth.csv
>iris.csv
>chickwts.csv
>esoph.csv
>mtcars.csv
>ChickWeight.csv
```