Iniciando Nossos Trabalhos

jameshunter R. Hunter

10 de fevereiro de 2017

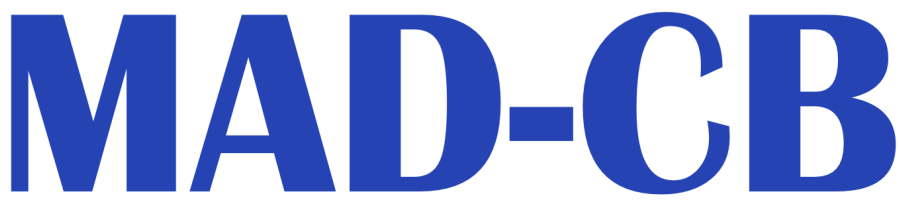


Figure 1:

Setup dos Monitores – em Casa

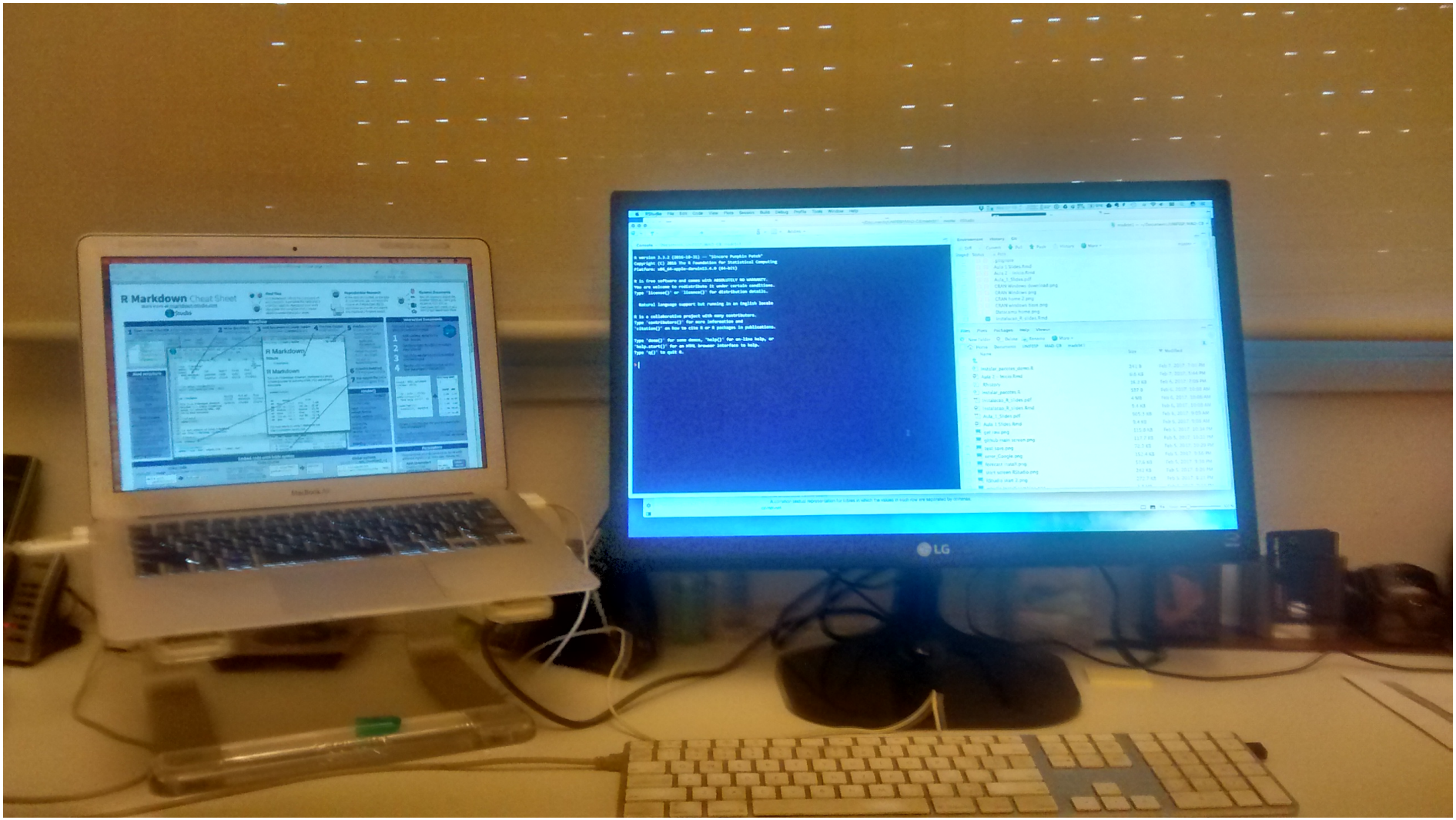


Figure 2:

O Ínicio

- Dados médicos sobre tratamentos para artrite e inflamação
- Com isso, podemos aprender
  - Como inserir dados nos objetos R
  - Como manipular esses dados
  - Gráficos básicos
  - Resumos descritivos de dados
  - Funções em R
  - Loops
- Muito deste lição e os dados vêm do site “Software Carpentry”
  - Agradeço eles

Verificar da Pasta

- Queremos estar trabalhando na pasta para matéria
- Pode usar a função `getwd()` para ver o que é o working directory

```
## [1] "/Users/jameshunter/Documents/UNIFESP/MAD-CB/madcbt1"
```

Dados para Working Directory

- Coloque os dados no working directory
- Salvé-los da Github para o working directory
- Nome de dados: “r-novice-inflammation-data.zip”
- No OS, expande o arquivo ‘zip’ com um dupla-clique
- Vai criar uma nova pasta chamada “`data`”
- Mude o nome desta pasta para “`artrite_data`”

Entre os Dados em Memoria

- Precisa abrir alguns pacotes para ler e manipular os dados
- Uso da função `library()`
- Dados são carregados no disco no formato ".csv"
- ".csv" ("Comma Separated Values") – Formato de Excel
- A função `read_csv()` faz parte do pacote `readr`
- `readr` faz parte do "tidyverse"
  - Podemos chamar isso e outros carregando o pacote `tidyverse`

Pacotes Carregados com o Pacote tidyverse

```
sessionInfo() ## Comando para mostrar o estado do sistema R neste momento
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] backports_1.0.5 magrittr_1.5    rprojroot_1.2   tools_3.3.2
##  [5] htmltools_0.3.5 yaml_2.1.14     Rcpp_0.12.9     stringi_1.1.2
##  [9] rmarkdown_1.3   knitr_1.15.1    stringr_1.1.0   digest_0.6.12
## [13] evaluate_0.10
```

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_0.5.0     purrr_0.2.2     readr_1.0.0     tidyr_0.6.1
## [5] tibble_1.2      ggplot2_2.2.1   tidyverse_1.1.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.9       plyr_1.8.4        forcats_0.2.0     tools_3.3.2
##  [5] digest_0.6.12     jsonlite_1.2      lubridate_1.6.0   evaluate_0.10
##  [9] nlme_3.1-131      gtable_0.2.0      lattice_0.20-34   psych_1.6.12
## [13] DBI_0.5-1         yaml_2.1.14       parallel_3.3.2    haven_1.0.0
## [17] xml2_1.1.1        stringr_1.1.0     httr_1.2.1        knitr_1.15.1
## [21] hms_0.3           rprojroot_1.2     grid_3.3.2        R6_2.2.0
## [25] readxl_0.1.1      foreign_0.8-67    rmarkdown_1.3     modelr_0.1.0
## [29] reshape2_1.4.2    magrittr_1.5      backports_1.0.5   scales_0.4.1
## [33] htmltools_0.3.5   rvest_0.3.2       assertthat_0.1    mnormt_1.5-5
## [37] colorspace_1.3-2  stringi_1.1.2     lazyeval_0.2.0    munsell_0.4.3
```

Carregar os Dados em Memoria

```
dados <- read_csv(file = "artrite_data/inflammation-01.csv",
                  col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

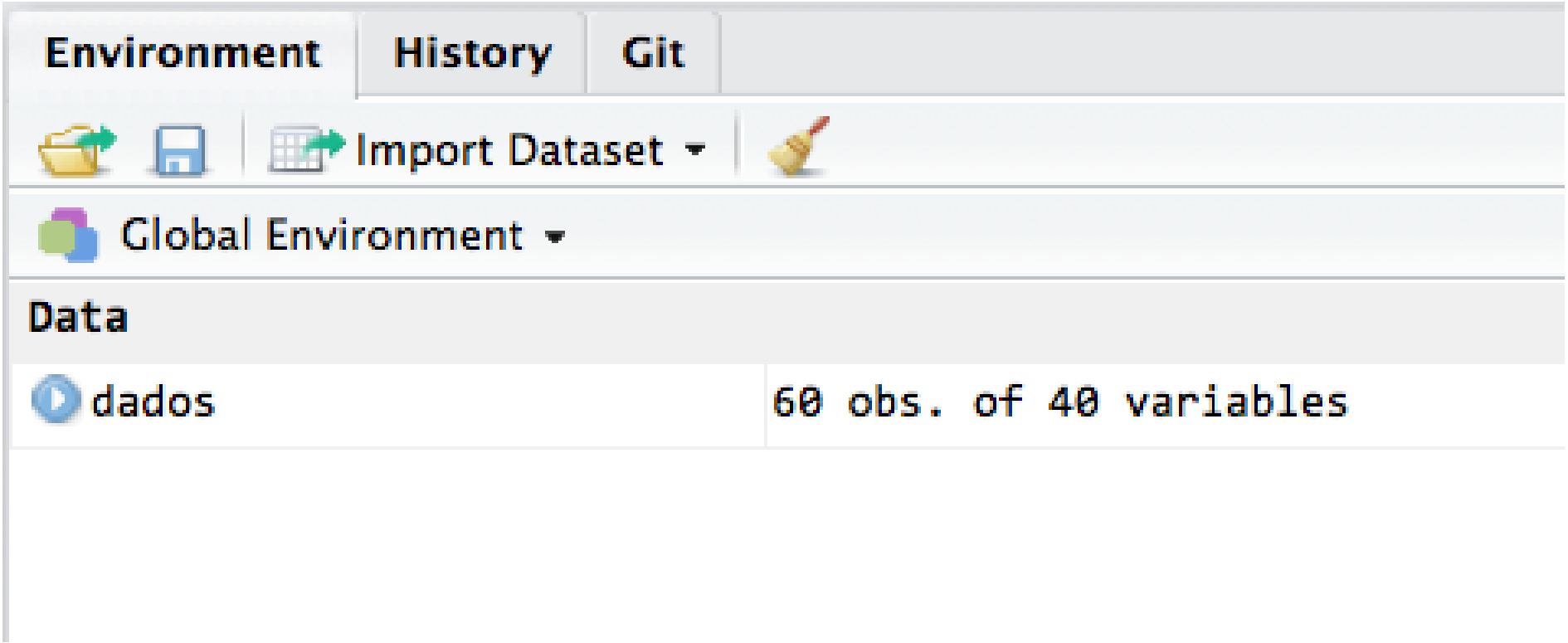

Environment
History
Git
Import Dataset
Global Environment
Data
dados
60 obs. of 40 variables

Figure 3:

A Classe de dados? (Usando str())

- str() - função que elabora a estrutura do objeto
  - Inclusive dos colunas (variáveis)
  - Mostra a classe de cada elemento

```
str(dados[, 1:5])
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    60 obs. of  5 variables:
##  $ X1: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X2: int  0 1 1 0 1 0 0 0 0 1 ...
##  $ X3: int  1 2 1 2 1 1 2 1 0 1 ...
##  $ X4: int  3 1 3 0 3 2 2 2 3 2 ...
##  $ X5: int  1 2 3 4 3 2 4 3 1 1 ...
```

O Que Significa Dados

- Dando um nome para os dados na planilha
- <- $=$ Assignment
- <- $\neq =$ (igual)
- <- quer dizer que o valor a direta está sendo associado ao nome a esquerda
  - <-

Exemplo Mais Simples

```
## 1ª Versão
peso <- 55  ## Pessoa pesa 55 kg.

## 2ª Versão
peso_kg <- 55 ## Mais claro

## Pode Converter à Libra
(peso_lb <- peso_kg * 2.2)
```

```
## [1] 121
```

```
peso_lb
```

```
## [1] 121
```

Qual Tipo de Dados Temos?

- Grau de inflamação de 60 pacientes durante 40 dias recebendo um novo tratamento para artrite
- Pacientes são linhas
- Dias são colunas

```
class(dados)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dim(dados)
```

```
## [1] 60 40
```

Pegar Dados Específicos do Conjunto

- Subset por *Índice*

```
dados[1, 1]
```

```
## # A tibble: 1 × 1
##      X1
##   <int>
## 1     0
```

```
dados[20, 20] # valor de linha 20, coluna 20
```

```
## # A tibble: 1 × 1
##     X20
##   <int>
## 1    16
```

```
dados[1:3, 1:5] # valores dos primeiros 3 pacientes para primeiro 5 dias
```

```
## # A tibble: 3 × 5
##      X1    X2    X3    X4    X5
##   <int> <int> <int> <int> <int>
## 1     0     0     1     3     1
## 2     0     1     2     1     2
## 3     0     1     1     3     3
```

Pode Ver Subsets dos Dados Não-Contiguos? SIM

- Índices não precisam começar com 1

```
dados[5:8, 6:9] # valores dos pacientes 5 - 8 para dias 6 até 9
```

```
## # A tibble: 4 × 4
##      X6    X7    X8    X9
##   <int> <int> <int> <int>
## 1     1     3     5     2
## 2     4     2     1     6
## 3     2     2     5     5
## 4     1     2     3     5
```

Usando o ":" para Adicionar um Grupo ("Slice") das Linhas/Colunas

- ":" num índice, permite que nós estamos chamando um grupo das linhas ou colunas *contiguas*
- Também, chamado um "slice"
- Pode usar a function `slice()` dentro do pacote `dplyr`
  - `dplyr::slice(dados, 1:3)` significa selecione as primeiras 3 linhas do objeto `dados`
  - O dupla ":" acima diz que `slice()` fica dentro do pacote `dplyr`

O Que É Um "**Tibble**"?

- `data.frame` é classe de dados mais comum para conjuntos de dados estatísticos
  - Deve ser retangular, i.e., todos as colunas deve ter o mesmo # de linhas
- `tibble` é a versão mais nova do data.frame
  - Não muda o tipo de um valor na base original – tem a ver com "strings"
  - Facilita a criação das colunas com a classe de listas (`list`)
  - Não mexe com os nomes de variáveis (colunas)
  - Mais eficiente na avaliação das funções (coisa de Inside Baseball)
- Consistente com os princípios de tidyverse

Colocando Nomes nas Colunas e nos Pacientes

- Nomes “X1”, “X2”, etc. não explicam nada
- Sabemos que as variáveis são dias e que as linhas são pacientes
- Vamos criar estes nomes
- Passo 1: criar um vetor (seqüência dos valores) dos dias
- Passo 2: criar um vetor dos Pacientes
  - Esses 2 são “strings”, ou, em R, vetores de classe “*character*”
- Passo 3: Usar “colnames()” para colocar os nomes de variáveis no lugar
- Passo 4: Aumentar o vetor dos Pacientes ao conjunto

```r
## vetor dos dias
dias <- c(paste0("dia", 1:ncol(dados)))

## vetor dos pacientes
pacientes <- c(paste0("pac", 1:nrow(dados)))

## Colocar os nomes nas variáveis
colnames(dados) <- dias

## Colocar os IDs dos pacientes na base
dados <- bind_cols(tibble(pacientes), dados)
```

O Resultado

```
dados[1:6, 1:8]
```

```
## # A tibble: 6 × 8
##   pacientes  dia1  dia2  dia3  dia4  dia5  dia6  dia7
##       <chr> <int> <int> <int> <int> <int> <int> <int>
## 1      pac1     0     0     1     3     1     2     4
## 2      pac2     0     1     2     1     2     1     3
## 3      pac3     0     1     1     3     3     2     6
## 4      pac4     0     0     2     0     4     2     2
## 5      pac5     0     1     1     3     3     1     3
## 6      pac6     0     0     1     2     2     4     2
```

Classe `int` Não Serve para os Cálculos

- As variáveis dos dias (int) não serve para cálculos de muitos parâmetros estatísticos
  - Mean não faz sentido com números inteiros
  - Precisa as casas decimais

Formula para Um Mean

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

- Quando dividimos por n (número de valores), precisa números decimais, não inteiros
- Mas, agora temos `int`, não `num`
- Precisamos forçar ("coerce") as variáveis para ser números decimais
  - "numeric" em R
- Comando para forçar conversão dos valores em R: "`as.<tipo de objeto>`"
  - Nós queremos usar `as.numeric`

Aplicando as.numeric a Uma Coluna Única

- dia1 como exemplo
- N.B. unlist necessário para razões internas de como R trata tibbles e data.frames
- Veremos uma alternativa preferida que não precisa unlist

```
dados[, 'dia1'] <- as.numeric(unlist(dados[, 'dia1']))
str(dados[, 'dia1'])
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    60 obs. of  1 variable:
##  $ dia1: num  0 0 0 0 0 0 0 0 0 0 ...
```

Índice Mais Eficiente - Índice por Nome

- As variáveis têm nomes
- Aproveitar desses nomes com a indexação com `$` e o nome da variável
- Para `dia1`: `dados$dia1`

```r
str(dados$dia1)
```

```
##  num [1:60] 0 0 0 0 0 0 0 0 0 0 ...
```

```r
## Retornar os IDs dos primeiros 10 pacientes
dados$pacientes[1:10] # Pode combinar os nomes das variáveis com
```

```
##  [1] "pac1"  "pac2"  "pac3"  "pac4"  "pac5"  "pac6"  "pac7"  "pac8"
##  [9] "pac9"  "pac10"
```

```r
                        # índices números para as linhas/os casos
```

Como Revisar Todos as 40 Variáveis dos Dias?

- Nossa tarefa é converter *todas* as 40 variáveis a classe "numeric"
- Pode fazer um "loop", uma ferramenta de programação para repetir algo múltiplas vezes
- Loops – não muito eficiente
- Alternativa melhor: Use uma versão da função `dplyr::mutate`
- `dplyr` pacote chave do manuseio dos dados do tidyverse
  - `mutate` cria novas variáveis ou muda variáveis existentes

Novo Símbolo – O “Pipe” (%>%)

- Funções no tidyverse podem ser ligados dentro de um comando usando %>%
- Quando usa o Pipe, o 1º argumento do comando depois %>% usa o resultado do comando anterior
- Segue um exemplo

Aplicar as.numeric às Variáveis de Dia

```
dados <- dados %>% mutate_at(vars(dia1:dia40), funs(as.numeric))
str(dados[, 2:6])
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    60 obs. of  5 variables:
##  $ dia1: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ dia2: num  0 1 1 0 1 0 0 0 0 1 ...
##  $ dia3: num  1 2 1 2 1 1 2 1 0 1 ...
##  $ dia4: num  3 1 3 0 3 2 2 2 3 2 ...
##  $ dia5: num  1 2 3 4 3 2 4 3 1 1 ...
```

Resumo de Comandos e Funções de Manuseio dos Dados

- getwd()/setwd()
- library()
- readr::read_csv()
- tidyverse
- sessionInfo()
- str()
- <-
- class()

Resumo – 2

- dim()
- dplyr::slice()
- data.frame/tibble
- paste()/paste0()
- dplyr::bind_cols()
- as.numeric()
- $
- %>%
- unlist()

Estatística Começa

Vamos Começar Descrever os Dados

- Vamos ver o que é o máximo grau de inflamação de Paciente 1 e a média (mean) durante os 40 dias
- Novas funções: `max()` e `mean()`
- Fazer subset dos dados com somente as variáveis do dia, sem o ID
- Novo comando de `dplyr` – select() – selecione colunas/variáveis para o subset

```
dadosDia <- dados %>% select(-pacientes)
## "-" em selecionar quer dizer omitir variável

## Create variable/vector for Paciente 1
pac1 <- dadosDia %>% slice(1) %>% unlist()
max(pac1)
```

```
## [1] 18
```

```
mean(pac1)
```

```
## [1] 5.45
```

Nosso Primeiro Resultado Estatístico

Que Mais Que Nós Podemos Tirar desses Dados?

- Qual é a relação entre Paciente 1 e os Outros
- Quão consistente é esse média? Tem muito variança?
  - Ou, 5.45 é um resumo bom de estado de inflamação?

Evolução de Grau de Inflamação

- Calcular a média para todos os dias usando todos os pacientes
- É um resumo (summary) dos resultados
- Uso de `summarize()` e `summarize_all()` de `dplyr`
- Com `summarize_all()`, so precisa dar o nome de função que quer usar como argumento

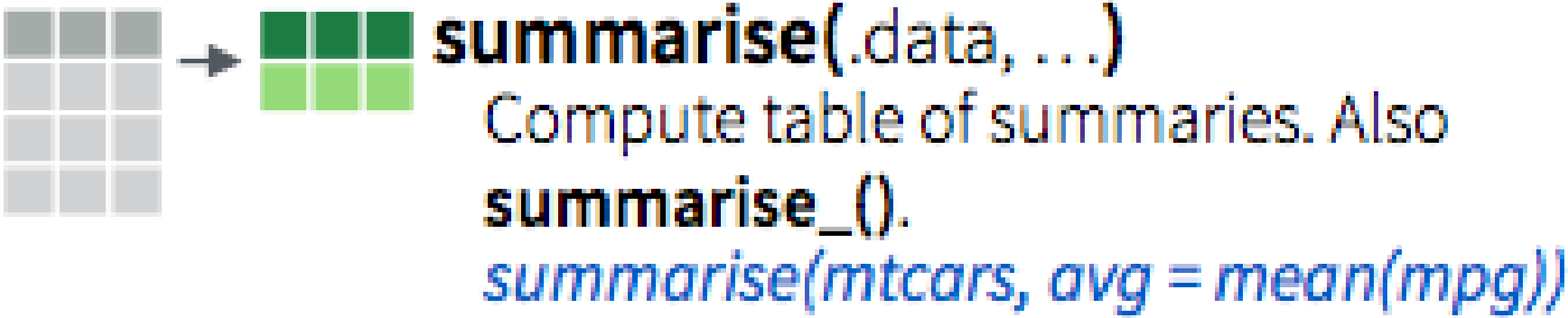


Figure 4:

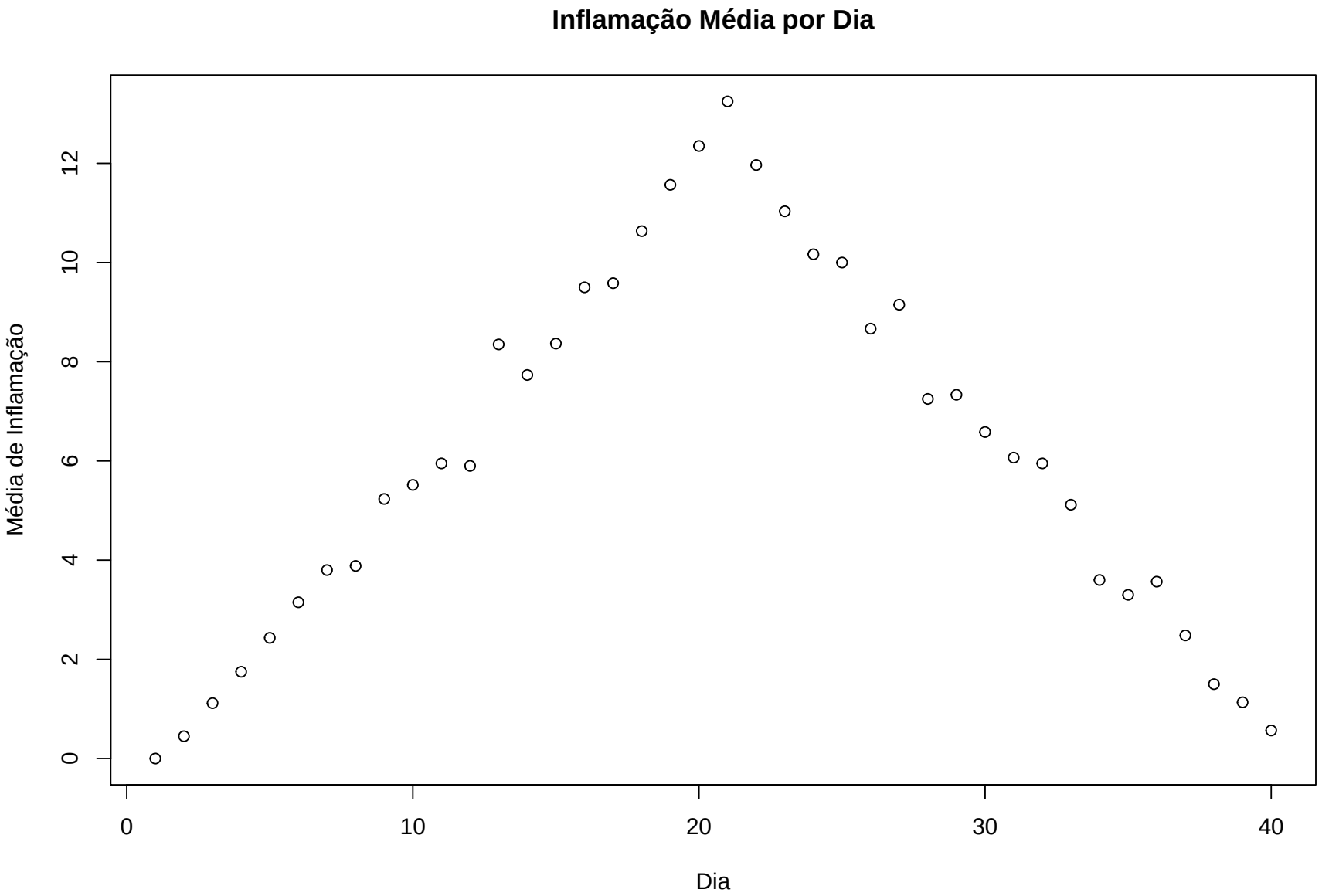
```r
meanDia <- dadosDia %>% summarize_all(mean)
unlist(meanDia)
```

```
##       dia1       dia2       dia3       dia4       dia5       dia6
##  0.0000000  0.4500000  1.1166667  1.7500000  2.4333333  3.1500000
##       dia7       dia8       dia9      dia10      dia11      dia12
##  3.8000000  3.8833333  5.2333333  5.5166667  5.9500000  5.9000000
##      dia13      dia14      dia15      dia16      dia17      dia18
##  8.3500000  7.7333333  8.3666667  9.5000000  9.5833333 10.6333333
##      dia19      dia20      dia21      dia22      dia23      dia24
## 11.5666667 12.3500000 13.2500000 11.9666667 11.0333333 10.1666667
##      dia25      dia26      dia27      dia28      dia29      dia30
## 10.0000000  8.6666667  9.1500000  7.2500000  7.3333333  6.5833333
##      dia31      dia32      dia33      dia34      dia35      dia36
##  6.0666667  5.9500000  5.1166667  3.6000000  3.3000000  3.5666667
##      dia37      dia38      dia39      dia40
##  2.4833333  1.5000000  1.1333333  0.5666667
```

Gráfico BÁSICO das Médias por Dia - Comando

```
plot(unlist(meanDia), main = "Inflamação Média por Dia",
     ylab = "Média de Inflamação", xlab = "Dia")
```

Gráfico BÁSICO das Médias por Dia



Agora, Sabemos Mais sobre Nossos Dados

- Inflamação aumenta até dia 20 e depois diminua até 0

Agora, Sabemos Mais sobre Nossos Dados

- Inflamação aumenta até dia 20 e depois diminua até 0
- Aumento e descida parecem de ser quase linear (i.e. em linhas retas)

Agora, Sabemos Mais sobre Nossos Dados

- Inflamação aumenta até dia 20 e depois diminua até 0
- Aumento e descida parecem de ser quase linear (i.e. em linhas retas)
- Este é uma idéia que vamos explorar um pouco para frente

Agora, Sabemos Mais sobre Nossos Dados

- Inflamação aumenta até dia 20 e depois diminua até 0
- Aumento e descida parecem de ser quase linear (i.e. em linhas retas)
- Este é uma idéia que vamos explorar um pouco para frente
- Quer dizer que não tem muito variância através dos pacientes

Perspectiva de Pacientes

- Vale a pena calcular a média de inflamação para cada paciente?

Perspectiva de Pacientes

- Vale a pena calcular a média de inflamação para cada paciente?
- Sim ou Não? Por quê?

Perspectiva de Pacientes

- Vale a pena calcular a média de inflamação para cada paciente?
- Sim ou Não? Por quê?
- Para mim, não.

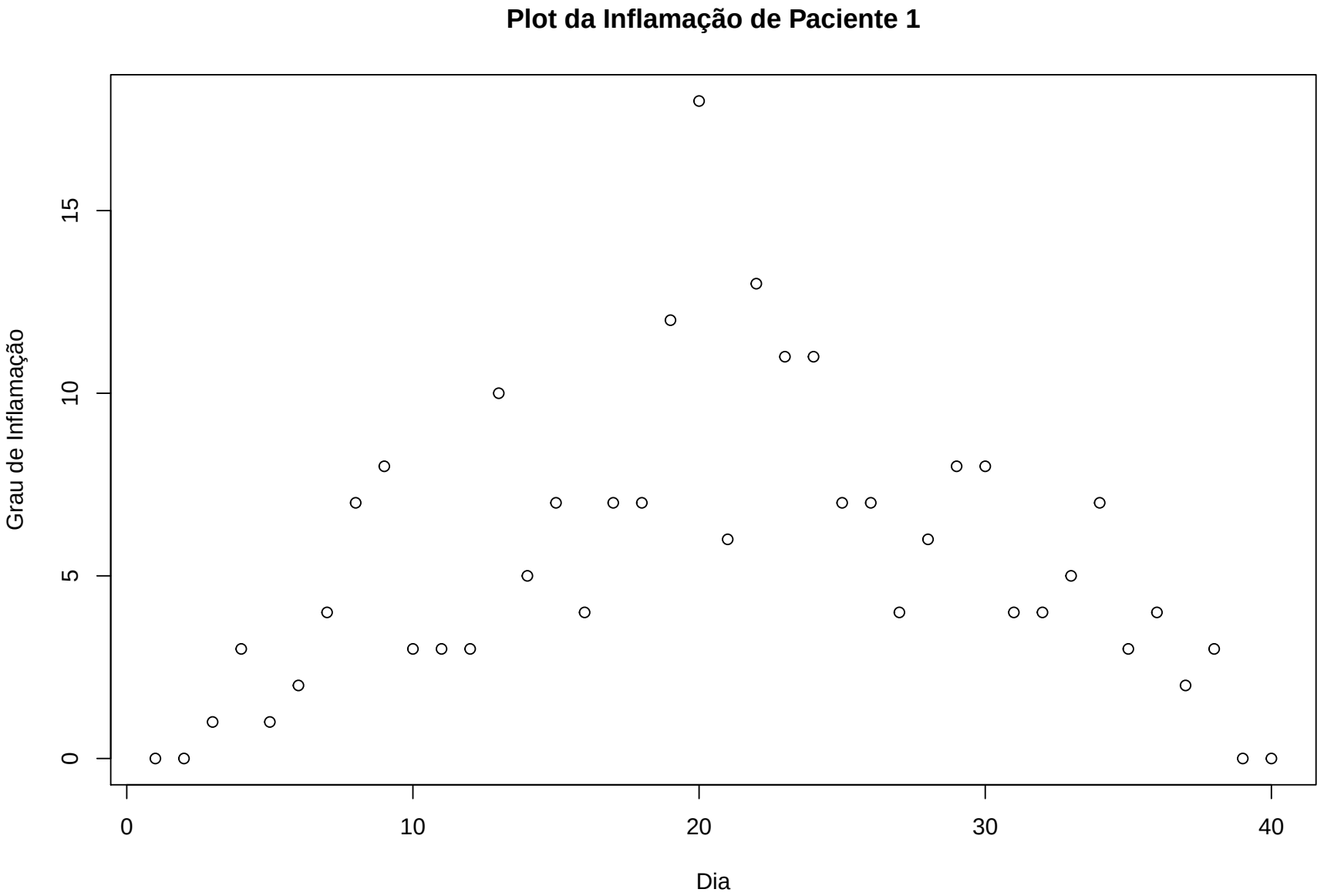
Perspectiva de Pacientes

- Vale a pena calcular a média de inflamação para cada paciente?
- Sim ou Não? Por quê?
- Para mim, não.
- Mas, gostaria de ver a dispersão de inflamação para cada paciente

Perspectiva de Pacientes

- Vale a pena calcular a média de inflamação para cada paciente?
- Sim ou Não? Por quê?
- Para mim, não.
- Mas, gostaria de ver a dispersão de inflamação para cada paciente
- Tentar um gráfico como o último primeiro

Gráfico de Paciente 1

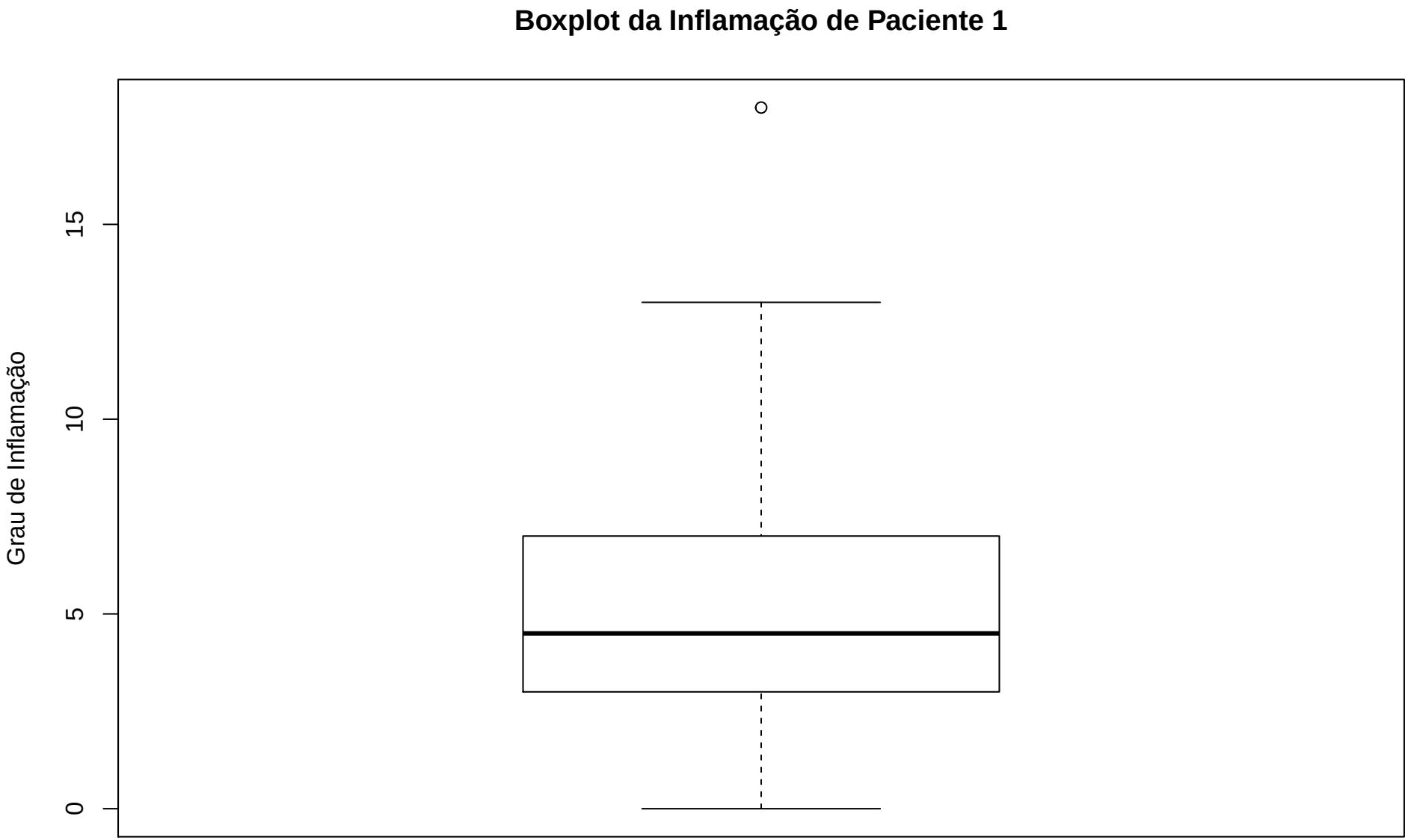


Resultado Não Muito Claro

- Sei que existe dispersão entre 0 e acima de 15
- Mas, perceber uma tendência é difícil
- Boxplot é um gráfico que organiza esse informação bem

Boxplot de Inflamação do Paciente 1

```
boxplot(pac1, main = "Boxplot da Inflamação de Paciente 1",
        ylab = "Grau de Inflamação")
```


Boxplot da Inflamação de Paciente 1
Grau de Inflamação
0
5
10
15

O Que Pode Entender deste Gráfico, Agora?

Mais uma Perspectiva sobre um Conjunto de Dados – Resumo de 5 Números

- em R, tem na função `summary()`
- Mostra em ordem:
  - Valor mínimo
  - Valor de 25º percentil (1º quartil)
  - Valor no meio de todos os valores (Mediana)
  - Valor de 75º percentil (3º quartil)
  - Valor máximo
- `summary` aumenta a média para a lista
- `IQR` = diferença entre o 3º quartil e o 1º quartil
  - $IQR = Q_3 - Q_1$
  - IQR – "Interquartile Range"

summary e IQR de Paciente 1

```
summary(pac1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    3.00    4.50    5.45    7.00   18.00
```

```
IQR(pac1)
```

```
## [1] 4
```

Entendendo o Boxplot Melhor

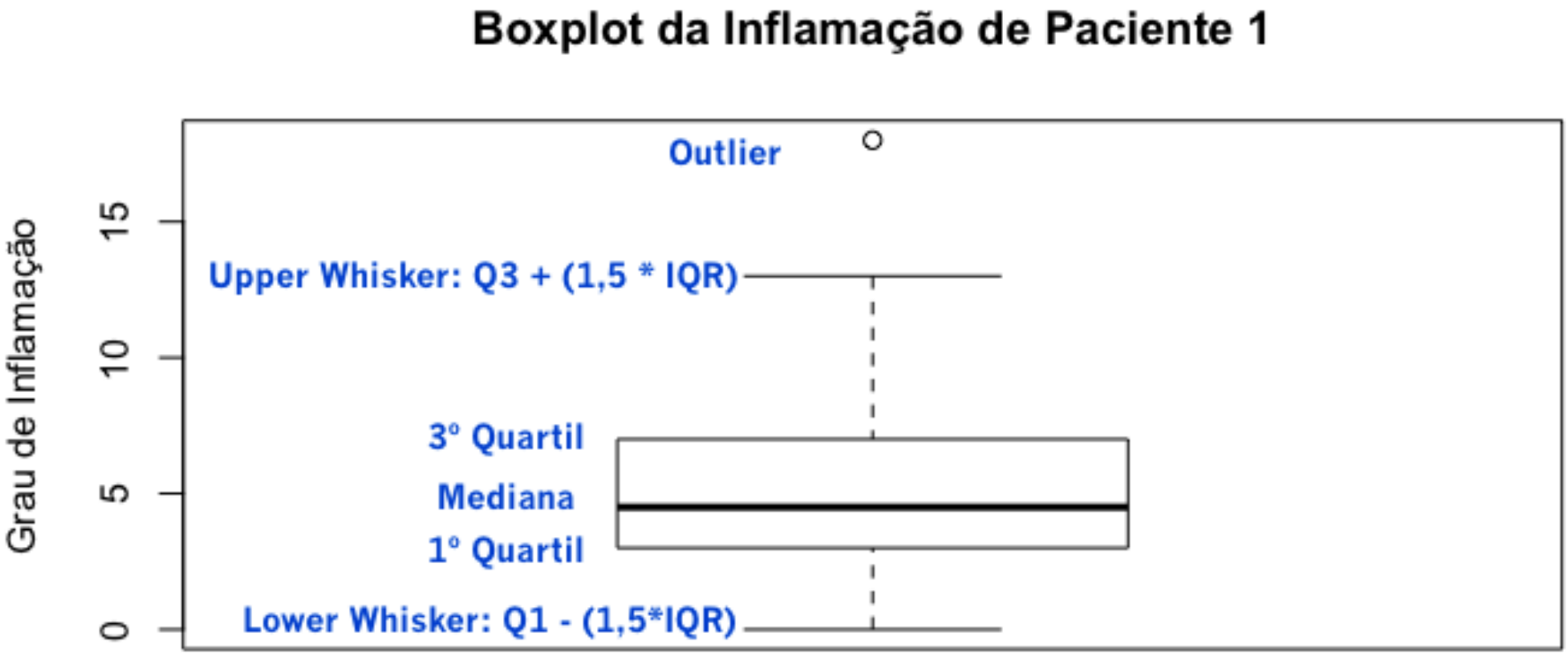


Figure 5:

Resumo das Medidas e Funções Estatísticas Que Usamos

- max()/min()
- mean()
- select()
- plot(x, main = , xlab = , ylab =)
- boxplot()
- summary()
- IQR()

Deixei Fora uma Medida Descritiva Super Importante

- Desvio padrão (standard deviation, sd, ou *sigma*) merece mais foco
  - Semana que vem
- *sigma* mede a dispersão dos dados em volta da média e forma um dos paramétros da distribuição mais conhecida em estatística — a Gaussiana ou Normal

Hoje, Fizemos Muito

- Começou de 0
- Estudamos como preparar dados para análise
- Fizemos vários exercícios de manuseio dos dados (“data wrangling”)
- Começamos de fazer análise descritivo de estatística com nossos dados
- Objectivo da aula: Mostre um pouco de tudo
  - Semana que vem: mais foco
  - Terça na teoria de estatística e probabilidade
  - Sexta nas ferramentas de R e RStudio para manipulação e análise

2 Artigos para Ler

- *Can we predict flu deaths with Machine Learning and R?* no Github
- *Code Alert*, Nature, Vol. 541, 26/1/17, <doi: 10.1038/nj7638-563a>
- Vamos conversar na terça sobre eles