Handling data in R

Timothée Bonnet

February 6, 2020

BDSI / RSB

About R-Stats workshops

- Schedule on Wattle

About R-Stats workshops

- Schedule on Wattle
- Slack RSB-R-Stats-Biology email me if you are interested

About R-Stats workshops

- Schedule on Wattle
- Slack RSB-R-Stats-Biology email me if you are interested
- Interested in Bayesian modeling? Let's chat during coffee break

Today

Some basic things you can do with data

- Read
- Access, subset, merge
- Iterate
- Write

Today

Some basic things you can do with data

- Read
- Access, subset, merge
- Iterate
- Write

Take notes in a R script. Comment using

```
# comment within R code
#' comment converted to Markdown
```

Compile HTML, Word or PDF with `CTRL + Maj + K`

Today

Some basic things you can do with data

- Read
- Access, subset, merge
- Iterate
- Write

Take notes in a R script. Comment using

```
# comment within R code
#' comment converted to Markdown
```

Compile HTML, Word or PDF with `CTRL + Maj + K`

If you are bored, go to the end for some exercises

Loading data

Data access, subset, merge

Repetitive tasks

R output

Extra practice

Type data

You can type short datasets

```
my_vector <- c(2,9.1, 4)

my_data_frame <- data.frame(weather=c("rain", "sun", "snow"),
                            temperature = c(12.2, 19.9, -1.4))
```

Or simulate random data

```
my_random_data <-
  data.frame(weather=sample(x = c("rain", "sun", "snow"),
                            size = 100, replace = TRUE),
             temperature = rnorm(n = 100, mean = 13, sd = 4))
```

Package data

R-base and R-packages contain data-sets

```
data("volcano")
volcano # you must "touch" the data to really load them
```

Package data

R-base and R-packages contain data-sets

```
data("volcano")
volcano # you must "touch" the data to really load them
```

```
library(ape)
data(cynipids)
cynipids
```

Load tables from file

```r
roo <- read.csv("Data/roo.csv")
ped <- read.table("Data/ped.txt", header = TRUE)
```

Load tables from file

```
roo <- read.csv("Data/roo.csv")
ped <- read.table("Data/ped.txt", header = TRUE)
```

What about Excel?

```
thorn <- read.csv("Data/thornexcel.xlsx") #ERROR
```

```
library(gdata)
thorn <- read.xls("Data/thornexcel.xlsx", sheet = 1, header = TRUE)
```

Online data

```
download.file(url="https://ndownloader.figshare.com/files/2292169"
              destfile = "dat.csv")
dat <- read.csv("dat.csv")
```

Loading data

Data access, subset, merge

Repetitive tasks

R output

Extra practice

Data overview

```
data(trees)
```

Data overview

```
data(trees)
```

```
str(trees)
```

```
'data.frame': 31 obs. of  3 variables:
 $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
 $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 .
```

Try also `summary`, `class`, `head`, `tail`, `plot`

Access

Bracket-syntax

- Row: dataframe[row,]
- Column: dataframe[, column]
- Cell: dataframe[row, column]

Access

Bracket-syntax

- Row: dataframe[row,]
- Column: dataframe[, column]
- Cell: dataframe[row, column]

```
trees[,1]
trees[1:8,]
trees[c(2,1,2), 3]
trees[, "Height"]
```

Dollar-syntax

- Column `dataframe$column_name`
- Element `dataframe$column_name[row]`

```
trees$Height
```

Time to think a tiny bit!

Calculate the mean for all three variables in trees, excluding the last (31st) record.

Solution for one column

Calculate the mean for all three variables in trees, excluding the last (31st) record.

```
mean(trees$Girth[1:30])
mean(trees[1:30, "Girth"])
mean(trees$Girth[-31])
mean(trees[-31, "Girth"])
```

subset

```
firstrees <- trees[1:10,]
```

subset

```r
firstrees <- trees[1:10,]
```

```r
smalltrees1 <- trees[trees$Height  < mean(trees$Height), ]

smalltrees2 <- subset(trees, subset = trees$Height  < mean(trees$He
```

```r
smalltrees1 == smalltrees2
identical(x=smalltrees1, smalltrees2)
```

Merging

Merge by row or column

```
trees2 <- trees*2

rbind(trees, trees2)
cbind(trees, trees2)
```

Merging

Merge columns by common id

```
roo <- read.csv("Data/roo.csv")
ped <- read.table("Data/ped.txt", header = TRUE)

AllData <- merge(x = roo, y = ped, by.x = "id", by.y = "animal")
```

Loading data

Data access, subset, merge

Repetitive tasks

R output

Extra practice

How to get the mean for every row?

```
data(rock)
str(rock)


'data.frame': 48 obs. of  4 variables:
 $ area : int  4990 7002 7558 7352 7943 7979 9333 8209 8393 6425 ..
 $ peri : num  2792 3893 3931 3869 3949 ...
 $ shape: num  0.0903 0.1486 0.1833 0.1171 0.1224 ...
 $ perm : num  6.3 6.3 6.3 6.3 17.1 17.1 17.1 17.1 119 119 ...
```

How to get the mean for every row?

```
data(rock)
str(rock)
```

```
'data.frame': 48 obs. of  4 variables:
 $ area : int  4990 7002 7558 7352 7943 7979 9333 8209 8393 6425 ..
 $ peri : num  2792 3893 3931 3869 3949 ...
 $ shape: num  0.0903 0.1486 0.1833 0.1171 0.1224 ...
 $ perm : num  6.3 6.3 6.3 6.3 17.1 17.1 17.1 17.1 119 119 ...
```

```
rowMeans(rock)
```

How to get the mean for every row?

```
data(rock)
str(rock)
```

```
'data.frame': 48 obs. of  4 variables:
 $ area : int  4990 7002 7558 7352 7943 7979 9333 8209 8393 6425 ..
 $ peri : num  2792 3893 3931 3869 3949 ...
 $ shape: num  0.0903 0.1486 0.1833 0.1171 0.1224 ...
 $ perm : num  6.3 6.3 6.3 6.3 17.1 17.1 17.1 17.1 119 119 ...
```

```
rowMeans(rock)
```

By the way:

```
colMeans(rock)
```

More flexible alternative: apply functions

```
apply(X = dataframe, MARGIN = 1 (row) or 2 (col), FUN = function)
```

More flexible alternative: apply functions

```
apply(X = dataframe, MARGIN = 1 (row) or 2 (col), FUN = function)
```

```
apply(X = rock, MARGIN = 1, FUN = mean)#by row
apply(X = rock, MARGIN = 2, FUN = mean)#by column
```

More flexible alternative: apply functions

```
apply(X = dataframe, MARGIN = 1 (row) or 2 (col), FUN = function)
```

```
apply(X = rock, MARGIN = 1, FUN = mean)#by row
apply(X = rock, MARGIN = 2, FUN = mean)#by column
```

Flexibility example:

```
  apply(X = rock, MARGIN = 1, FUN = function(x) var(x^(1/2) ) )
```

The ultimate tool: For-loops

```
for (i in 1:N)
{
  something as a function of i
}#end of the loop
```

The ultimate tool: For-loops

```
for (i in 1:N)
{
  something as a function of i
}#end of the loop
```

```
for (i in 1:31)
{
  print(i)
}
```

How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

We will want to store the result somewhere:

```
  ResultMean <- vector() # we will store the results there
  ResultMean[22] <- mean(as.numeric(trees[22,]))
```

How to get the row means? For-loops

Start by buiding the code for 1 iteration (1 "i" value, e.g., 22):

```
mean(as.numeric(trees[22,]))
```

We will want to store the result somewhere:

```
  ResultMean <- vector() # we will store the results there
  ResultMean[22] <- mean(as.numeric(trees[22,]))
```

Now change 22 to "i" and write a loop around:

```
ResultMean <- vector() # we will store the results there
for (i in 1:31)
{
  ResultMean[i] <- mean(as.numeric(trees[i,]))
}
```

For-loops: your turn!

Load rock data

```
rock <- read.csv("rock.csv")
```

Use a for loop to obtain column averages

Solution

Load rock data.

```
rock <- read.csv("rock.csv")
```

Use a for loop to obtain column averages

```
storage <- vector(length = ncol(rock))
for (i in 1:ncol(rock))
{
  storage[i] <- mean(rock[,i])
}
```

Trade-off concision / flexibility

- colMeans shortest, but does only means
- apply very flexible, but does only array/matrix/data-frame
- for-loop looks complex, but infinitely flexible
- (NB: your computer does a for-loop whether you see it or not)

NB: The last value

Sometimes a function takes very long to run

```
    rowMeans(rock[rep(1:nrow(rock),100000),])
```

NB: The last value

Sometimes a function takes very long to run

```
    rowMeans(rock[rep(1:nrow(rock),100000),])
```

What if you forgot to save the output to an object??

NB: The last value

Sometimes a function takes very long to run

```
    rowMeans(rock[rep(1:nrow(rock),100000),])
```

What if you forgot to save the output to an object??

```
ourlasthope <- .Last.value
```

Loading data

Data access, subset, merge

Repetitive tasks

R output

Extra practice

Write data-frames and matrices

```
write.csv(rock, file = "rock.csv", quote = FALSE, row.names = FALSE
write.table(rock, file = "rock.txt", row.names = FALSE)
```

Write figures

functions pdf(); jpeg(); png();...

```
pdf(filename = "myplot.pdf", width = 4, height = 6)
plot(x=rock$area, y=rock$peri)
dev.off()
```

Write R-environments (DANGER)

Save all objects in your environmnet Non-reproducible!!!

```
save.image(file = "Feb6_2020_rock.RData")


load("Feb6_2020_rock.RData")
```

Write R-objects

Save one or a few objects. Good reason:

- To share with someone else
- To avoid re-doing long computations (big models)

```
a <- 1:10
save(list = c("a", "rock"), file = "ubu")
load("ubu")
```

Loading data

Data access, subset, merge

Repetitive tasks

R output

Extra practice

Exercise 1

Recover this online dataset:

```
download.file(url="https://ndownloader.figshare.com/files/2292169"
              destfile = "dat.csv")
dat <- read.csv("dat.csv")
```

- Create a subset with the first 200 entries
- Pull out that last row using `nrow()`
- Use nrow() to extract the row that is in the middle of the data frame. Store the content of this row in an object named surveysmiddle.
- Combine nrow() with the - notation above to reproduce the behavior of head(surveys), keeping just the first through 6th rows of the surveys dataset.

Exercise 2

- Create a subset containing only the records from the year 2000.
- What are the dimentions of this subset?
- What columns contain missing values?
- Discard all rows containing missing values (you may use the function na.omit())
- Save that table as a csv file.

Exercise 3 (difficult)

- Create a new data-frame containing the mean weight and hindfoot length for each genus.
- Write the data-frame to a csv file.
- Make a graph of weight by hindfoot length and export it as a pdf (using code)