Introduction to R Workshop for Researchers

University of Melbourne
Tuesday 3 and Friday 6 October 2017

9:00 am - 5:00 pm

Instructor: Pablo Franco (pablo.franco.dn@gmail.com)

Workshop website https://resbaz.github.io/introRcourse_oct2017/

DAY 1 CODE:
https://drive.google.com/open?id=0B1J2U_Uk1eoCNXdOTTROdVNkLU0

Workshop Requirements

Participants must bring own Laptop (and charger), with a Mac, Linux, or Windows operating system (not a tablet, Chromebook, etc.) that they have administrative privileges on.

First Download and install R (this will require around 160 Mb of space)

https://cloud.r-project.org/

- for Mac: https://cran.rstudio.com/bin/macosx/
  - select R-3.4.pkg
- for Windows: https://cran.rstudio.com/bin/windows/base/
  - select Download R 3.4 for Windows

then Download and install RStudio

http://www.rstudio.com/download

- RStudio 1.0.143 - Mac OS X 10.6+ (64-bit) (Mac)
- RStudio 1.0.143 - Windows Vista/7/8/10 (Windows Vista)

Decline the git developer tools prompt for now

Course material:
https://nikkirubinstein.gitbooks.io/resguides-introductory-r-workshop/content/content/01-rstudio-intro.html

Cheat Sheet for Base R
https://www.rstudio.com/wp-content/uploads/2016/05/base-r.pdf

https://goo.gl/4Gqsnz

TRUMP

https://drive.google.com/file/d/0B1J2U_Uk1eoCMzlpVzMzMl90dHM/view?usp=sharing

.csvfeline-data

htt

https://goo.gl/4Gqsnz

GGPLOT2 functions and description

http://ggplot2.tidyverse.org/reference/

 # Challenge 1.1

#

 # Which of the following are valid R variable names?

#

 # min_heightv T

 # max.height T surprisingly

 # _age F

 # .mass F

 # MaxLength T

 # min-length F

 # 2widths F

 # celsius2kelvin T

 # Challenge 1.2

#

 # What will be the value of each variable be after each statement

 # in the following program?

#

 # mass <- 47.5 47.5

 # age <- 122 122

*# mass <- mass * 2.3 109.25*

 # age <- age - 20 102

 # Challenge 1.3

#

 # Run the code from the previous challenge, and write a

*# command to **compare mass to age**. Is mass larger than age?*

mass==age3

mass>age

‘R

!=

the.truth = mass>=age

 # Challenge 1.4

#

 # Clean up your working environment by deleting the mass and

 # age variables.

rm(age)

rm(mass)

rm(list=ls())

rm(list=ls())

 # Challenge 1.5

#

#Install the following packages: dplyr, ggplot2, tidyr

 # Challenge 2.1

#

*# Look at the help for the **c()** function. What kind of vector*

*# do you expect you will create if you evaluate the following using **typeof()?***

#

c(1, 2, 3) double

c('d', 'e', 'f') char

c(1, 2, 'f')` char

 # Challenge 2.2

#

*# Look at the help for the **paste()** function. You'll need to use*

*# this later. What is the difference between the **sep** and*

*# **collapse** arguments?*

 # Challenge 3.1 5min
#
 # Start by making a vector with the numbers 11 to 20.
 # Then use the functions we just learned to extract the 3rd
 # through 5th element in that vector into a new vector; name
 # the elements in that new vector 'Res', 'Baz', 'R'.

```
chal31vc = seq(11,20,1)

ansvec=chal31vc[3:5]
names(ansvec) = c('Res','Baz','ARRR')

ansvec
```

 # Challenge 3.2
#
 # Is there a factor in our cats data.frame? what is its name?
*# Try using **?read.csv** to figure out how to keep text columns*
*# as **character** vectors **instead of factors**. Load feline-data.mes of the data types or*
 # and data structures we've seen so far.

Data frames and reading in data
csv into a
*# variable called **cats1** using this extra command. Write a command*
 # or two to show that the factor in cats1 is now a character
 # vector.

 # Challenge 3.3
#
*# Create a **list** of length two containing a character vector for*
 # each of the sections in this part of the workshop:
#
 # - Data types
 # - Data structures
#
 # Create data_types and data_structures as separate vectors, and
 # populate each with the names of the data types or
 # and data structures we've seen so far

```
data_types = as.character(c('logical', 'integer', 'numeric', 'complex',
'character'))
data_structures = as.character(c('list', 'vector', 'dataframe'))
g = list(data_types, data_structures)
```

 # Challenge 4.1

#

You can create a new data.frame right from within R

with the following syntax:

#

df <- data.frame(id = c('a', 'b', 'c'),

x = 1:3,

y = c(TRUE, TRUE, FALSE),

stringsAsFactors = FALSE)

#

 # Make a data.frame that holds the following

 # information for yourself:

#

- first name

- last name

- lucky number

#

 # Then use rbind to add an entry for the people

 # sitting beside you. Finally, use cbind to add

 # a column with each person's answer to the question,

 # "Is it time for coffee break?"

```
df <- data.frame(firstname = c('Tolga'),lastname = c('Ozdogan'),luckynumber = c('9'),
stringsAsFactors = FALSE)

newLine = list('Leo', 'Mccomb', '9')
df = rbind(df, newLine)

newColumn = c('Is it time for coffee break?')
coffeeBreak = c('No', 'Yes')
df = cbind(df, coffeeBreak)
View(df)
```

 # Challenge 5.1

```
#
# Given the following code:
#
#         x <- c(5.4, 6.2, 7.1, 4.8, 7.5)
#         names(x) <- c('garfield', 'nermel', 'odie', 'jon', 'pooky')
#         print(x)

# which gives the following output:
#
#         garfield   nermel     odie         jon       pooky
#         5.4        6.2         7.1      4.8       7.5
#
# 1. Come up with at least 3 different commands that
# will produce the following output:
#
#     nermel odie  jon
#           6.2    7.1   4.8
#
# 2. Compare notes with your neighbour. Did you have
# different strategies?
x[c(-1,-5)
x[2:4]
x[c(-1,-5)]
x[c(2,3,4)]
x[c("nermel","odie","jon")]


x[-c(1,5)]
```

```
          x[2:4]
          x[c('nermel','odie','jon')]
          x[-c(1,5)]
```

 # Challenge 5.2

```
#
# Run the following code to define vector x as above:
```

```
#
#       x <- c(5.4, 6.2, 7.1, 4.8, 7.5)
#       names(x) <- c('garfield', 'nermel', 'odie', 'jon', 'pooky')
#       print(x)
#
# Given this vector x, what would you expect the
# following to do?
#
#       x[-which(names(x) == "scratchy")]
#
# Try out this command and see what you get.
# Did this match your expectation?
# Why did we get this result?
# (Tip: test out each part of the command on it's
# own like we just did above - this is a useful
# debugging strategy)
#
# Which of the following are true:

#       A) if there are no TRUE values passed to which,
#       an empty vector is returned
#       B) if there are no TRUE values passed to which,
#       an error message is shown
#       C) integer() is an empty vector
#       D) making an empty vector negative produces an
#       "everything" vector
#       E) x[] gives the same result as x[integer()]

 \
# Challenge 5.3

# Subset the data that satisfy the following conditions:
    A. Males and are older than 35
    B. Males between the age of 5 and 60

titanicM35=titanicNew[titanicNew$Sex=="male" & titanicNew$Age>35,]
titanic560=titanicNew[titanicNew$Sex=="male" & titanicNew$Age>5 & titanicNew$Age<60,]

# Challenge 5.4
#
# Consider:
#
#       xlist <- list(a = "Research Bazaar", b = 1:10, data = head(iris))
```

```
#
# Using your knowledge of both list and vector
# subsetting, extract the number 2 from xlist.
# Hint: the number 2 is contained within the "b" item
# in the list.




# Challenge 5.7
#
# Fix each of the following common data frame subsetting errors:
#
# 1. Extract rows (i.e., passengers) where age = 35
#         titanic[titanic$Age = 35, ]
          titanicNew[titanicNew$Age==35, ]
#
# 2. Extract all columns except 1 through to 4
#         titanic[ ,-1:4]
          titanic[ ,-c(1:4)]
titanic[ ,c(-1:-4)]

#
# 3. Extract the rows where the passenger class (Pclass) is 1.
#         titanic[titanic$Pclass == 1 ]
          titanic[titanic$Pclass == 1, ]
#
# 4. Extract the first row, and the fourth and fifth
# columns (Name and Sex).
#         titanic[c(1),c( 4, 5)]
tNS = titanic[1,4:5]
          tNS = titanic[1,4:5]
#
# 5. Advanced: extract rows that contain information for
# those aged 14 and 16.
titanic[titanic$Age == 14 | 16,  ]
#
 titanic [titanic$Age == 14 | titanic$Age ==16,]

titanic[titanic$Age %in% c(14, 16),  ]

# Challenge 5.8
#
```

 # 1. Why does titanic[1:20] return an error? How does
 # it differ from titanic[1:20,]?
#
 # 2. Create a new ***data.frame called titanic_small***
 # that only contains rows 1 through 9, and 19 through
 # 23. *You can do this in one or two steps.*

```
titanic_small=titanic[c(1:9,19:23),]
```

```
titanic_small = titanic[c(1:9,19:23),]
```

```
fahr_to_kelvin <- function(temp) {
  kelvin <- ((temp - 32) * (5 / 9)) + 273.15
  return(kelvin)
}
```

Now it's your turn to define a function. Create a function to convert temperatures in kelvin to celsius. Hint: to convert from kelvin to celsius substract 273.15.

```
kelvin_to_celcius <- function(temp) {
  celcius <- (temp - 273.15)
  return(celcius)
}

kelvin_to_celcius(273.15)

kelvin_to_celsius <- function(temp) {
  celsius <-(temp -273.15)
  return(celsius)
```

Challenge 2: Build Your own nested function

Using the two functions we've defined to convert fahrenheit to kelvin, and kelvin to celsius, define a function that converts fahrenheit to celsius.

```
fahr_to_celcius <- function(temp) {
  kelvin<-fahr_to_kelvin(temp)
  celcius<-kelvin_to_celcius(kelvin)
  return(celcius)


fahr_to_celc = function(temp){
  celcius = kelvin_to_celc(fahr_to_kelvin(temp))
  return(celcius)
}

}
```

```
calcAgeAverage <- function(dat) {
  ageAverage <- mean(dat$Age, na.rm = TRUE)
  return(ageAverage)
}
```

```
calcAgeAverage <- function(dat, sex = "female") {
  ageAverage <- mean(dat[dat$Sex == sex, ]$Age, na.rm = TRUE)
  return(ageAverage)
}
```

Challenge 3: Subset by age and passenger class

Define the function to calculate the average age for specific classes of a single sex. Hint: Look up the function %in%, which will allow you to subset by multiple classes.

```
calcAgeAverage <- function(dat, Pclass = 1, sex = "female") {
  ageAverage <- mean(dat[dat$Sex == sex & dat$Pclass %in% Pclass, ]$Age, na.rm = TRUE)
  return(ageAverage)
}
```

Challenge 1

Can the points be colour coded according to passenger class? (Hint: use factor(Pclass))
Hint: use colour as aesthetic property within geom_points
Time: 3 minutes

Challenge 2

Can we show lines and points on the same plot?
Time: 3 minutes

Challenge 3

Can we make color coding global?
Time: 3 minutes

research Q: Did passengers on Titanic survive equally based on Sex and Class?

Challenge 4:

Plot the frequency distribution of 'Age' of Passengers on Titanic. (Hint: histogram)
Can we plot it by gender?
Time: 5 minutes

```
# Challenge 8.1
#
# Let's try this on the Fare column of the titanic dataset.
#
# Make a new column in the titanic data frame that contains
# Fare rounded to the nearest integer. Check the head or tail
# of the data frame to make sure it worked.
#
# Hint: R has a round() function

titanic$roundfare <- round(titanic$Fare)
head(titanic$roundfare)

FareRounded=round(titanic$Fare)
titanic2=cbind(titanic,FareRounded)

# Challenge 8.2
#
# We're interested in looking at the sum of the
# following sequence of fractions:
#
#         Sum = 1/1 + 1/2 + 1/3 + ... + 1/n
```

#
This would be tedious to type out, and impossible
for high values of n.
*# Use vectorisation to compute **the sum of squares (Sumsq) when n=100.** What is # the sum of squares when **n=10,000**?*
TIP: START WITH x=1:100 and apply functions to that vector

```
x=1:100
x
y=1/x
v1=1:100
v2=1/v1
sum(v2)
```

 # Challenge 9.1
#
 # Use an if statement to print a suitable message
 # reporting:
 # 1. Whether there are any people with age
# of 100 in the titanic dataset.
 # 2. Now do the same to report if all passengers are older than 10.
#Tip: Look for ?all
#
HINT: you could use the **na.rm** argument in the
any() function. OR **na.omit**(any()).

```
if(any(titanic$Age==100)){
  print("There are people with age of 100 in Titanic")
} else {
  print("There is no one with age of 100 in Titanic")
}
if (any(titanic$Age<=10)){
  print("Not all the passengers are older than 10")
} else {
  print("All the passengers are older than 10")
}

if (all(titanic$Age>10)){
  print("All the passengers are older than 10")
} else {
```

```
  print("Not all the passengers are older than 10")
}

# Challenge

# Calculate the mean of survival for each passenger class in the titanic
# dataset and print a statement declaring if the mean is >= 0.5 or < 0.5


# Challenge

Loop through the vector (1:20) add 2 to each number and save th
e result into a vector


# Challenge 10.1
#
# Rewrite your 'pdf' command to print again a page for each class. This time # in each page
show a facet plot (hint: use facet_grid) of the same data with # one panel per sex.
p
pdf(file="myPlot4.pdf",width=10,height=7)
 print(
   ggplot(data=titanic[titanic$Pclass==class,],
          aes(x=Age,y=Fare,col=as.factor(Pclass)))+
     geom_point()+
     facet_grid(titanic$Sex,)
     geom_smooth(method="lm",se=TRUE)+
     scale_y_log10()
 )
 dev.off
pdf(file="myplot.pdf",width=10, height=7)

for (class in unique(titanic$Pclass)){
  print(
    ggplot(data=titanic[titanic$Pclass==class,],aes(x=Age,y=Fare,col=as.factor(Pclass)))+
      geom_point()+ facet_grid(Pclass~Sex)+
      geom_smooth(method="lm",se=TRUE)+
      scale_y_log10()
  )}
dev.off()
```

 # Challenge 10.2
#
 # Make a subset the titanic data to include only data for children
 # (below the age of 18).
#
 # Write the new subset to a file in your IntroductiontoR/ directory.

```
install.packages(“dplyr”)
library(dplyr

titanicChil=titanic[titanic$Age<18,]
write.table(titanicChil,file="titanicChildren.txt",sep="\t",quote=FALSE,row.names = FALSE)
```

AgeSiblingsFareMales=titanic %>% filter(Sex=="male") %>% select(Age,SibSp,Fare)

```
A=select(filter(titanic,Sex=="male"),Sex,Age,SibSp,Fare)
View(A)
```

 # Challenge 11.2
#
 # Calculate the average Survived value per Pclass and Sex.
 # Which combination of Pclass and Sex had the highest
 # Survived value and which combination had the lowest?

```
SurvivedbyClass=titanic%>% select(Pclass,Sex,Survived) %>% group_by(Pclass,Sex) %>%
  summarise(meanSurvived=mean(Survived))

Survivedbyclasssex=titanic %>% group_by(Pclass, Sex) %>%
  summarise(meanSurvived=mean(Survived))
```

 # Advanced Challenge
#
 # Calculate the average Survived value for a group of 20
 # randomly selected females from each Pclass group.
 # Then arrange the classes in descending order.
#
Hint: Use the dplyr functions arrange() and sample_n(),

they have similar syntax to other dplyr functions.
Look at the help!

fem20class=titanic %>% filter(Sex=="female") %>% sample_n(20,replace=TRUE) %>% group_by(Pclass,Sex) %>% summarise(meanSurvived=mean(Survived)) %>% arrange(desc(Pclass))

#Challenge T.1: Vectorization
Create a column that finds shows TRUE if "america" is present in the tweet irrespective of Case
Add the column to the TRUMP dataframe
Hint: Use grepl() with ignore.case
#How many times has he used America in a tweet in 2017?
#How many of those original tweets (not retweets) (Hint: False is different to FALSE)

 # Challenge T.2: FOR/WHILE
 # Part A.
Repeat the previous exercise but now for the following vector of words
words = c("great","again","trump","loser") #YOU CAN ADD UR OWN WORDS!
 # Part B.
How many times has Trump tweeted each of those words (in 2017)?
Save the results in a data frame,
where a column represents the word and the other is the number of times that the word was tweeted
Hint:R Remember that trump$text=trump[["text"]]

#Challenge T.3: GGPLOT
#Present the trumpCount data in a plot. Choose the plot type which you think is more convenient

Gemo_col?
You have one Discrete variable (words)
And one continuous (wordCounts)
plotWord=ggplot(data=trumpCounts, aes(x=words,y=wordCount))+geom_col()

```
ggplot(data=trumpCount,aes(x=words,y=wordCount))+
  geom_col()
```