

PSTAT 126

Lab 1

Roupen Khanjian

Spring 2021

R Review

Have 2 options on how to use Rstudio

- 1) Use the “RStudio on Cloud link” on the home page of Gauchospace.
- 2) Use Rstudio on your own local computer.

First Download R, then Rstudio.

- **R for Mac:** <https://cran.r-project.org/bin/macosx/>
- **R for windows:** <https://cran.r-project.org/bin/windows/base/>
- **RStudio** download: <https://www.rstudio.com/products/rstudio/download/>

How to install packages

```
install.packages("alr4")  
install.packages(pkgs = c("faraway", "tidyverse"))
```

- OR go to the *packages* pane click on *install* then type in package name(s) and click install.

How to load packages

```
library(alr4)  
library(faraway)
```

- OR go to the *packages* pane click on the white box next to the package you are trying to load.

help

```
?seq # fast way to go the help page for a function  
help(seq) # another way to go to help page for function
```

Vectors

```

x <- c(1,2,3,4,5)
(x <- 1:5) # can assign and print out object by putting() around code.

## [1] 1 2 3 4 5
seq(from = 1, to = 5, by = 1) # sequence of numbers starting from 1 until 5 by 1

## [1] 1 2 3 4 5
seq(from = 2, to = 10 , by = 2) # sequence of numbers starting from 2 until 10 by 2

## [1] 2 4 6 8 10
seq(1,5,1)

## [1] 1 2 3 4 5

```

A few other useful functions

```

rep(3,times=5) # repeats 3 five times

## [1] 3 3 3 3 3
rev(x) # reverse the order of your vector

## [1] 5 4 3 2 1
abs(-8) # absolute value

## [1] 8
log(3) # natural log

## [1] 1.098612
exp(1) # e^()

## [1] 2.718282
summary(x) # prints out 5 number summary along with mean

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##         1         2         3         3         4         5
sample(x, 1) # takes a random sample of size 1

## [1] 5

Strings:
y <- "PSTAT 126" # use "" for strings
y

## [1] "PSTAT 126"
class(y) # strings are referred to as characters in R

## [1] "character"
paste("Hello", "World") # a way to put 2 strings together

## [1] "Hello World"

```

Working with vectors

```
x <- 8:18
x

## [1] 8 9 10 11 12 13 14 15 16 17 18
x[2] # print out second element

## [1] 9
x[2] <- 19 # reassign second element
x[2] # print out second element

## [1] 19
x[2:5] # print out second to the fifth element

## [1] 19 10 11 12
x[c(2,5)] # print out the 2nd and 5th elements

## [1] 19 12
length(x) # length of vector

## [1] 11
x <- c(2,5,7,9)
x+1 # add 1 to every element in the vector

## [1] 3 6 8 10
x/2 # divide every element by 2

## [1] 1.0 2.5 3.5 4.5
x*2 # multiply every element by 2

## [1] 4 10 14 18
x^2 # square every element

## [1] 4 25 49 81
x**2 # another way to square every element

## [1] 4 25 49 81
sum(x) # Sum all vector elements

## [1] 23
x[4] <- NA # assign the 4th element in the vector as a missing value
sum(x) # does not work with missing values

## [1] NA
sum(x, na.rm = TRUE) # use argument na.rm to remove missing values

## [1] 14
x <- 2:4
prod(x) # Multiply all vector elements
```

```
## [1] 24
x <- 5:9
sqrt(x) # square root each element

## [1] 2.236068 2.449490 2.645751 2.828427 3.000000
```

And some more useful functions

```
mean(x) # mean

## [1] 7
sd(x) # standard deviation

## [1] 1.581139
var(x) # variance

## [1] 2.5
sort(x) # sort the vector in ascending order

## [1] 5 6 7 8 9
sort(x, decreasing = TRUE) # sort the vector in descending order

## [1] 9 8 7 6 5
min(x) # minimum value for the vector

## [1] 5
max(x) # maximum value for the vector

## [1] 9
range(x) # range (min max)

## [1] 5 9
x <- sqrt(x)
x

## [1] 2.236068 2.449490 2.645751 2.828427 3.000000
round(x, 2) # how to round values

## [1] 2.24 2.45 2.65 2.83 3.00
```

Matrices

```
(mat <- matrix(c(3,2,5,3,1,4,7,4,9), nrow = 3))

##      [,1] [,2] [,3]
## [1,]    3    3    7
## [2,]    2    1    4
## [3,]    5    4    9
mat[2,2]

## [1] 1
```

```

mat[2,2]<-100
sqrt(mat) # square root of each element in matrix

##          [,1]      [,2]      [,3]
## [1,]  1.732051  1.732051  2.645751
## [2,]  1.414214 10.000000  2.000000
## [3,]  2.236068  2.000000  3.000000

mat^2 # square of each entry in matrix

##          [,1] [,2] [,3]
## [1,]      9      9     49
## [2,]      4 10000     16
## [3,]     25     16     81

mat%*%mat # matrix multiplication

##          [,1] [,2] [,3]
## [1,]      50   337   96
## [2,]     226 10022  450
## [3,]      68   451  132

solve(mat) # Matrix inversion for non-singular matrices

##          [,1]      [,2]      [,3]
## [1,] -1.124681934 -0.001272265  0.875318066
## [2,] -0.002544529  0.010178117 -0.002544529
## [3,]  0.625954198 -0.003816794 -0.374045802

diag(mat) # extract the diagonal elements of the matrix

## [1]      3 100      9

mat

##          [,1] [,2] [,3]
## [1,]      3      3      7
## [2,]      2  100      4
## [3,]      5      4      9

t(mat) # transpose of matrix

##          [,1] [,2] [,3]
## [1,]      3      2      5
## [2,]      3  100      4
## [3,]      7      4      9

sum(mat) # sum of all entries in matrix

## [1] 137

mat[2,] # extract second row of matrix

## [1]      2 100      4

sum(mat[2,]) # sum up all the element of the second row of the matrix

## [1] 106

mat[2,] <- c(2,5,10)
mat

```

```
##      [,1] [,2] [,3]
## [1,]    3    3    7
## [2,]    2    5   10
## [3,]    5    4    9
```

Working with data sets

```
data('faithful')
?faithful # information on the dataset
nrow(faithful) # number of rows in the dataset
```

```
## [1] 272
```

```
ncol(faithful) # number of columns in the dataset
```

```
## [1] 2
```

```
dim(faithful) # dimension of dataset
```

```
## [1] 272    2
```

```
head(faithful) # first 6 rows (observations) of the dataset
```

```
##      eruptions waiting
## 1      3.600      79
## 2      1.800      54
## 3      3.333      74
## 4      2.283      62
## 5      4.533      85
## 6      2.883      55
```

```
tail(faithful) # last 6 rows (observations) of the dataset
```

```
##      eruptions waiting
## 267      4.750      75
## 268      4.117      81
## 269      2.150      46
## 270      4.417      90
## 271      1.817      46
## 272      4.467      74
```

```
names(faithful) # name of columns
```

```
## [1] "eruptions" "waiting"
```

```
faithful[1:5,2] # extract the first 5 rows and 2nd column of data set
```

```
## [1] 79 54 74 62 85
```

```
faithful$waiting[1:5] # extract the first 5 rows and `waiting` column of data set
```

```
## [1] 79 54 74 62 85
```

```
apply(faithful, 2, mean) # obtain the mean of each column in a dataset
```

```
##      eruptions    waiting
## 3.487783 70.897059
```

```
x <- 1:5
```

```
y <- 2:6
```

```
cbind(x,y) # create a matrix by combining vectors column-wise
```

```
##      x y
## [1,] 1 2
## [2,] 2 3
## [3,] 3 4
## [4,] 4 5
## [5,] 5 6
```

```
rbind(x,y) # create a matrix by combining vectors row-wise
```

```
##      [,1] [,2] [,3] [,4] [,5]
## x      1   2   3   4   5
## y      2   3   4   5   6
```

Categorical variables

```
x <- c(1,2,3,2,3,4,3,2,1,2,3,4,4)
class(x)
```

```
## [1] "numeric"
```

```
x <- factor(x) # how to convert a numerical vector into a categorical one
class(x)
```

```
## [1] "factor"
```

```
levels(x) # prints out the levels of the categories
```

```
## [1] "1" "2" "3" "4"
```

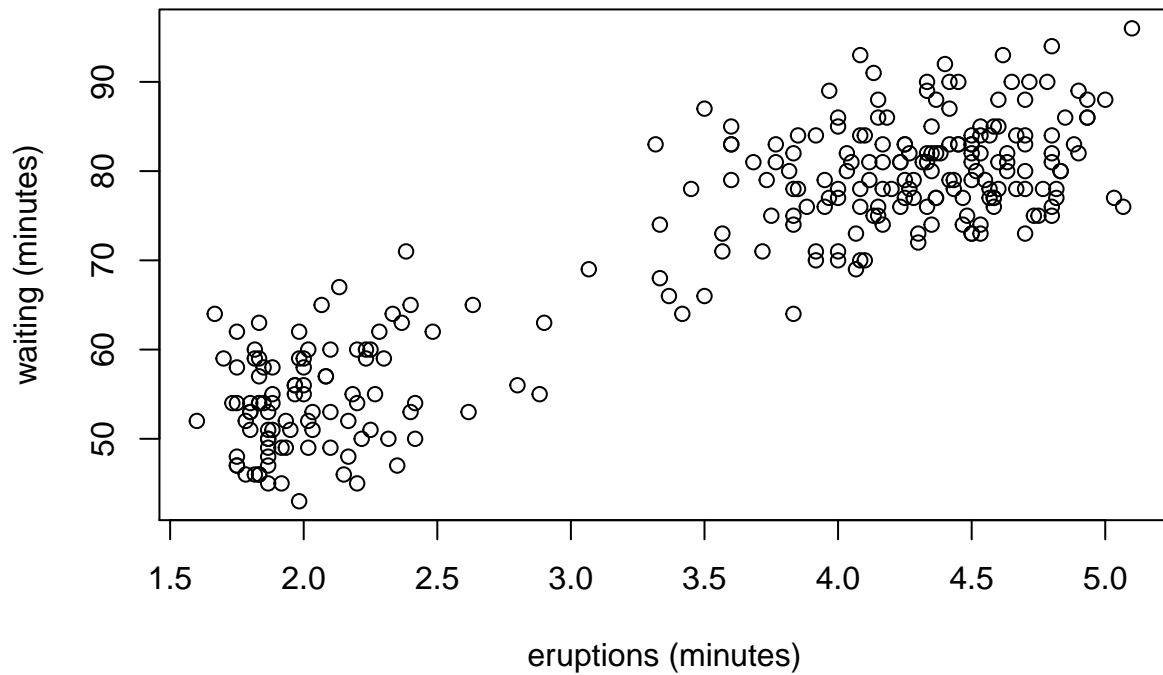
```
nlevels(x) # prints out the number of levels
```

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

Base R plotting

```
plot(faithful$eruptions, faithful$waiting,
     xlab = "eruptions (minutes)", ylab = "waiting (minutes)",
     main = "A plot of time between eruptions and duration of eruptions")
```

A plot of time between eruptions and duration of eruptions



Brief Tidyverse intro

```
library(tidyverse)
faithful %>%
  filter(eruptions > 3) %>% # filter eruptions above 3
  slice(1:5) # print out only first 5 rows

##   eruptions waiting
## 1     3.600      79
## 2     3.333      74
## 3     4.533      85
## 4     4.700      88
## 5     3.600      85

faithful %>%
  select(eruptions) %>% # select eruptions column
  filter(eruptions <= 2) %>% # filter eruptions equal to or below 2
  slice(1:5) # filter eruptions above 3

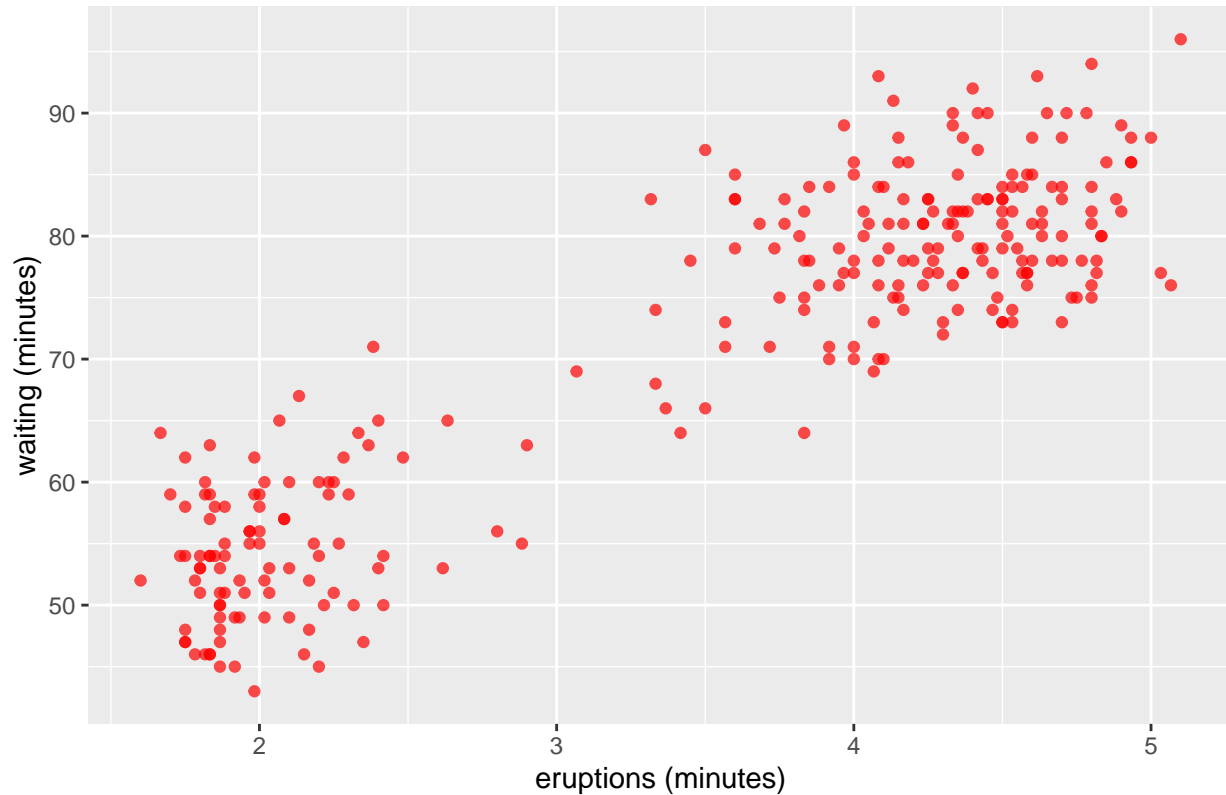
##   eruptions
## 1     1.800
## 2     1.950
## 3     1.833
## 4     1.750
## 5     1.750

# scatterplot with eruptions on x axis and waiting on y
ggplot(data = faithful,
  aes(x = eruptions, y = waiting)) +
  # red color data points. alpha changes the transparency of the points
  geom_point(color = "red", alpha = 0.7) +
```



```
labs(x = "eruptions (minutes)",
     y = "waiting (minutes)",
     title = "A ggplot of time between eruptions and duration of eruptions")
```

A ggplot of time between eruptions and duration of eruptions



Other useful functions in the tidyverse are `group_by()`, `arrange()`, `mutate()`, and `summarise()`. For more info on tidyverse function go to <https://r4ds.had.co.nz/transform.html/>. Can use `glimpse()` (from tidyverse package) or `str()` to get a quick look at dataset.

```
str(faithful)
```

```
## 'data.frame': 272 obs. of 2 variables:
## $ eruptions: num 3.6 1.8 3.33 2.28 4.53 ...
## $ waiting : num 79 54 74 62 85 55 88 85 51 85 ...
```

```
glimpse(faithful)
```

```
## Rows: 272
## Columns: 2
## $ eruptions <dbl> 3.600, 1.800, 3.333, 2.283, 4.533, 2.883, 4.700, 3.600, 1.95~
## $ waiting <dbl> 79, 54, 74, 62, 85, 55, 88, 85, 51, 85, 54, 84, 78, 47, 83, ~
```

Distributions

Can obtain random sample from a specific distribution:

Type in `?distribution` in console to see all the different distributions in R.

- `p` for “probability”, the cumulative distribution function (c. d. f.)

- q for “quantile”, the inverse c. d. f.
- d for “density”, the density function (p. f. or p. d. f.)
- r for “random”, a random variable having the specified distribution

```
rnorm(10, mean = 0, sd = 1)
```

```
## [1] 1.5674999 -0.4401595 0.4770275 -0.7170414 0.4228052 -0.2388130
## [7] -0.9213240 -0.4610122 -0.9095698 -0.3097701
```

- random sample from a normal distribution with mean = 0, standard deviation = 1

```
runif(10, min = 0, max = 1)
```

```
## [1] 0.20106905 0.07790612 0.89374740 0.85772452 0.55152465 0.34622191
## [7] 0.80477189 0.14596829 0.45475421 0.45699612
```

- random sample from a uniform distribution with minimum value = 0, maximum value = 1

```
rpois(10, lambda = 7)
```

```
## [1] 8 5 6 5 5 4 6 4 9 13
```

- random sample from a poisson distribution with lambda parameter = 7

```
sample <- rnorm(1000, mean = 0, sd = 1)
mean(sample) # mean of sample
```

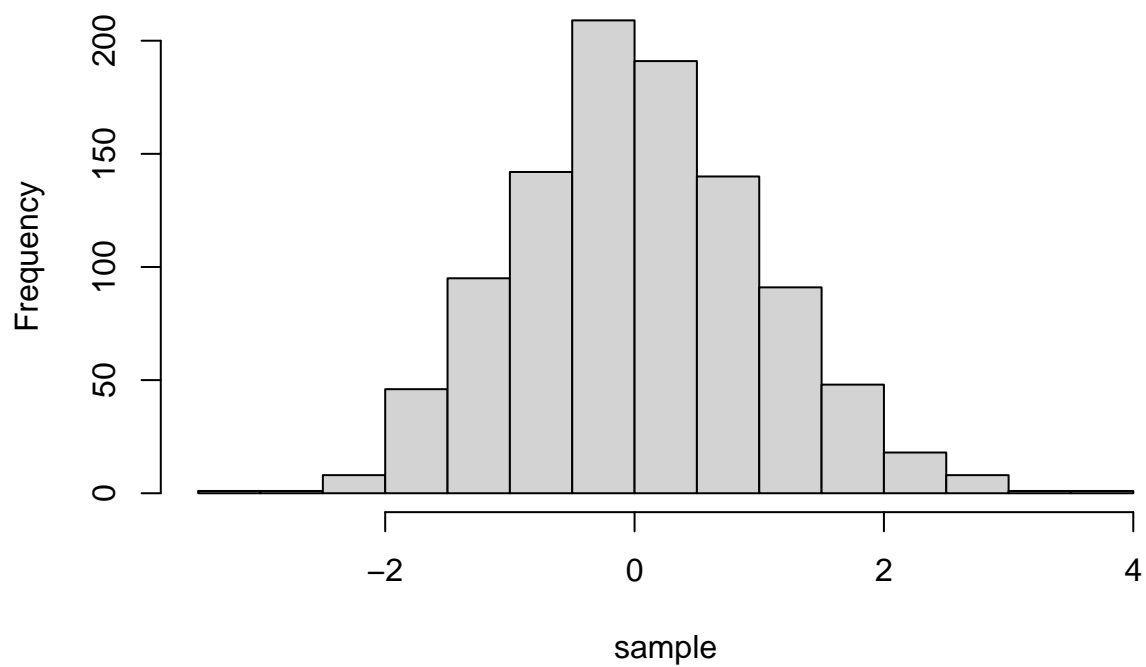
```
## [1] 0.03298275
```

```
sd(sample) # standard deviation of sample
```

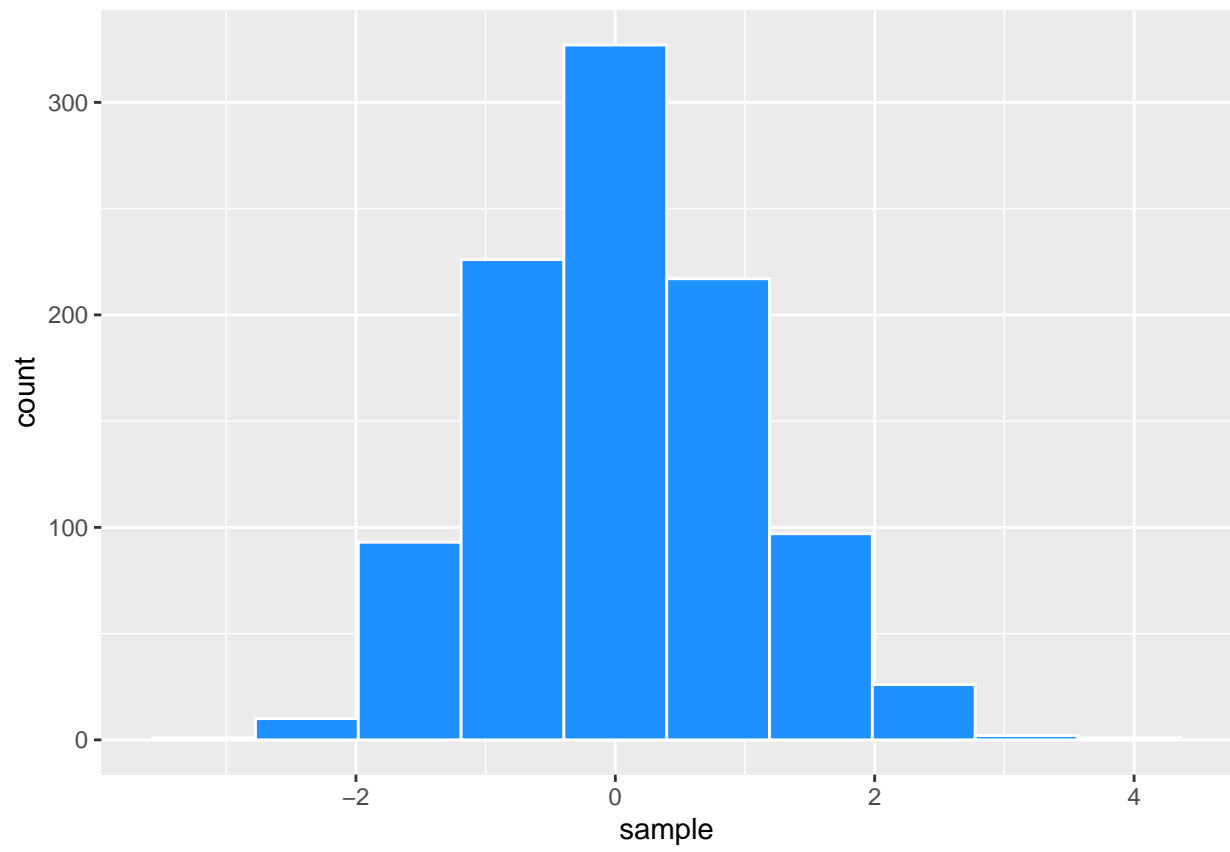
```
## [1] 0.9848796
```

```
hist(sample) # make a simple histogram from base r plotting
```

Histogram of sample



```
ggplot() +  
  geom_histogram(aes(x = sample), # using ggplot to make histograms  
                 bins = 10, color = "white", # control the number of bins  
                 fill = "dodgerblue")
```



Rmarkdown

In the menu bar of RStudio, click on File, then New File, and choose R Markdown. Select the default option (Document), and click Ok.

Rmd files are a special type of file, referred to as a dynamic document, that allows to combine narrative (text) with R code.

For more info go to <https://rmarkdown.rstudio.com/> Also see “RMarkdown Reference Guide”

- Don't name 2 chunks the same!
- Can write in LaTeX in Rmarkdown like this: $\bar{x} = \mu$ or:

$$\sum_{x=1}^5 x^2 = 55$$

Useful Rstudio shortcuts

- alt(option) + - = assignment operator (<-)
- Ctrl(Cmd) + alt(option) + I = new code chunk in Rmarkdown
- Ctrl(Cmd) + shift + C = Comment or uncomment lines highlighted
- Ctrl(Cmd) + shift + A = Reformat code in a neat way (most of the time)
- Ctrl(Cmd) + shift + M = pipe operator (%>%)
- alt(option) + shift + K = see all of the other fun shortcuts!

Some other useful tips

- In order to keep your files organized (on your local computer) try using R projects. <https://teachdatascience.com/projects/>
- <https://rstudio.com/resources/cheatsheets/> for cheatsheets