# Introduction to R

Ruihan Lu

GradQuant
University of California, Riverside, CA

October 12, 2020

# Table of Contents



# Table of Contents



# Introduction

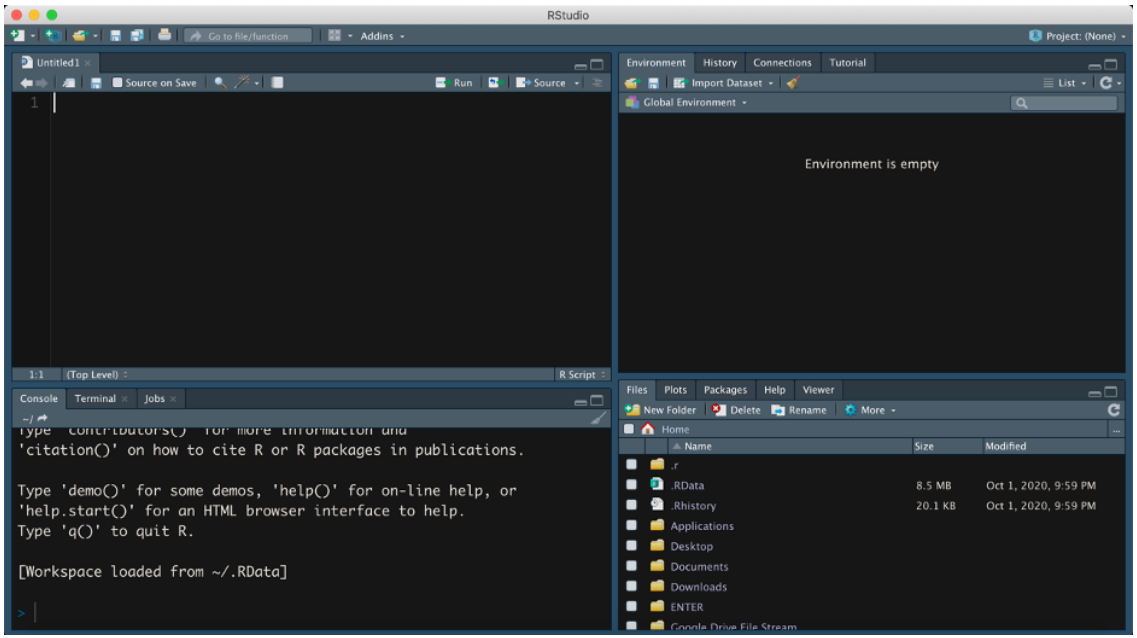
- R is an open source software that offers tremendous flexibility and capability.
- Accessible and easy to download, no license restrictions
- Powerful enough for complex computing tasks, like machine learning
- Graphics capabilities surpass many other programs
- Works well with other programs like Python, SQL and Excel

# Installing and Using R

- Installation
  - To obtain R for Windows (or Mac) go to The Comprehensive R Archive Network (CRAN) at `http://www.r-project.org`. Just download the executable file, and it will install itself.
- Recommended Software in R
  - RStudio
  - RMarkdown
- Notes
  - R can be run without RStudio, but RStudio is a really nice GUI that will make it easier to work in R.
  - R must be installed for RStudio to run.

# RStudio

- R script
- R console (a.k.a. terminal)
- Workspace
- Help



# Table of Contents

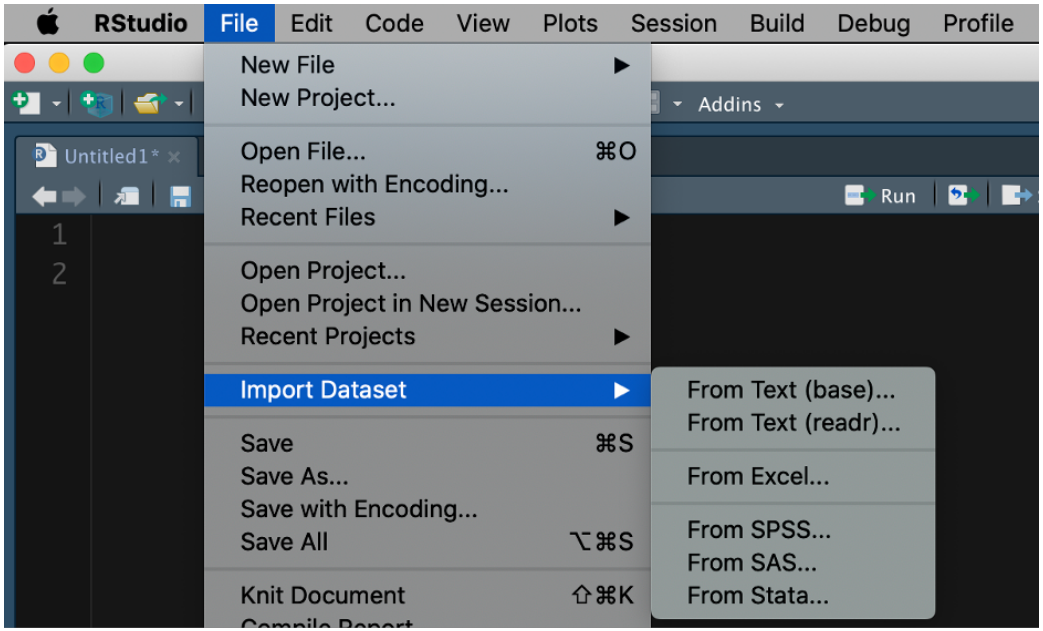


# Using R as a Scientific Calculator

Let's start with some simple mathematical operations.

- ```
  > 2+2
  [1] 4
  ```
- ```
  > 2^3
  [1] 8
  ```
- ```
  > 3+4*(3^2)
  [1] 39
  ```
- ```
  > sqrt(2)
  [1] 1.414214
  ```
- ```
  > exp(2)
  [1] 7.389056
  ```

Notes: R uses standard order of operations.


  ```

# Test for (in)equality

- ```
  > 2 == 3
  [1] FALSE
  ```
- ```
  > 2 < 3
  [1] TRUE
  ```

Notes: We use "==" instead of "=" to test equality. The "=" operator is for setting things equal to one another.

# Table of Contents



# Working Derectory

- A working directory is the default location where R looks for files.
- To see a current working directory where all of your packages and other R files will be downloaded to / uploaded from.
  - ```
    > getwd()
    [1] "/Users/ravenlu/Desktop/GQ_Introduction to R"
    ```
- To change working directory
  - ```
    setwd("deseired_path")
    ```
- Notes: the direction of the slashes for different operating systems.

# Importing Data Into R by Using File Tab

- RStudio has a File tab that can import different types of data files, including text, Excel, SPSS, SAS and Stata.



# Importing Data Into R by Using Packages

- From A Comma Delimited Text File
  - ```
    dat <- read.table('~/Desktop/GQ_introduction to R/test_csv.csv', sep = ',', header = T)
    ```
  - Or
    ```
    dat <- read.csv('~/Desktop/GQ_introduction to R/test_csv.csv')
    ```
- From Excel file
  - ```
    library(readxl)
    dat <- read_excel('~/Desktop/GQ_introduction to R/test_excel.xlsx')
    ```
- From SPSS
  - ```
    library(foreign)
    dat <- read.spss("~/Desktop/GQ_introduction to R/test_spss.sav",to.data.frame = T)
    ```


    ```

# Table of Contents



# Data Structure

- Single value.
  - R can store and use objects that we create, like the single value.
  - 
```
> x <- 2*3
> print(x)
[1] 6
```
- Vector.
  - A vector is a sequence of values, all of the same type;
  - **c()** function returns a vector containing all its arguments in order.
  - 
```
> y <- c(2, 4, 6, 8)
> print(y)
[1] 2 4 6 8
> is.vector(y)
[1] TRUE
```
  - We can also access all or part of our vector.
  - 
```
> y[1]
[1] 2
> y[2:3]
[1] 4 6
```

# Data Structure: Vectors

We can now work with an entire vector instead of individual values.

- Vector arithmetic.
  - Operators apply to vectors "pairwise" or "elementwise".
  - ```
    > x+y
    [1]  8 10 12 14
    ```
  - Recycling repeat elements in shorter vector when combined with longer.
  - ```
    > y + c(1,2)
    [1]  3  6  7 10
    ```
  - Single numbers are vectors of length 1 for purposes of recycling.
  - ```
    > y*2
    [1]  4  8 12 16
    ```


    ```

# Data Structure: Vectors

Functions on vectors:

- mean(), median(), sd(), var(), max(), min(), length(), sum(): return single numbers.
- sort() returns a new vector.
- hist() takes a vector of numbers and produces a histogram, a highly structured object, with the side-effect of making a plot.
- ecdf() produces a cumulative-density-function object.
- summary() gives a six-number summary of numerical vectors.

# Data Structure: Matrices

- A matrix is a specialization of a 2D array in R.
  - Matrices can be ordered by different directions.
  - 
```
> mtx <- matrix(c(1:9), nrow = 3, ncol = 3)
> mtx
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```
  - 
```
> mtx1 <- matrix(c(1:9), nrow = 3, ncol = 3, byrow = T)
> mtx1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```
- We can also access all or part of matrices.
  - 
```
> mtx[1,2]
[1] 4
```

Notes: R numbers its vector and matrices starting at 1.
If you're familiar with programming languages like C, this is something to be mindful of.

# Data Structure: Matrix Algebra

- Element-wise multiplication.
  - 
```
> mtx * mtx1
     [,1] [,2] [,3]
[1,]    1    8   21
[2,]    8   25   48
[3,]   21   48   81
```
- Matrix multiplication.
  - 
```
> mtx%*%mtx1
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126
```
- Transpose
  - 
```
> t(mtx)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

# Data Structure: Matrix Algebra

- Creates diagonal matrix with elements of x in the principal diagonal.
  - ```
    > diag(3)
         [,1] [,2] [,3]
    [1,]    1    0    0
    [2,]    0    1    0
    [3,]    0    0    1
    ```
- Inverse of A where A is a square matrix.
  - ```
    > A <- matrix(c(2, 3, 5, 7), nrow = 2)
    > solve(A)
         [,1] [,2]
    [1,]   -7    5
    [2,]    3   -2
    ```
- Returns vector of row means.
  - ```
    > rowMeans(A)
    [1] 3.5 5.0
    ```
- Returns vector of column means.
  - ```
    > colMeans(A)
    [1] 2.5 6.0
    ```

# Simple functions

- A simple linear or quadratic function can be write in R.
- For example: $y = 2x + 3x^2$
  - ```
    > y<-function(x) {2*x+3*(x^2)}
    > y(1)
    [1] 5
    ```
- Function with more than one unknown variable.
- For example: $y = 2xt + 3x^2t$
  - ```
    > y2 <- function(x, t) {2*x*t+3*(x^2)*t}
    > y2(1,1)
    [1] 5
    > y2(1,2)
    [1] 10
    ```

# If/else Statement

- The if/else statement executes a block of code if a specified condition is true.
- 

```
> subset1 <- ifelse(dat$Z1==1, 1, 0)
> subset1
 [1] 0 0 0 0 0 1 1 1 1 1 0
```

# Table of Contents



# Simple Linear Model

- R has a huge number of built-in datasets (most of which are in specific packages).
- We will use the **cars** dataset which, is a default R dataset. Thus, we do not need to load a package first; it is immediately available.
  - ```
    > View(cars)
    ```
- Before we analyze any dataset, we should get some basic statsical information about the dataset.
  - ```
    > dim(cars)
    [1] 50  2
    ```
  - There are totally fifty rows and two columns in the **cars**.
  - ```
    > names(cars)
    [1] "speed" "dist"
    ```
  - One column named *Speed* that represents the speed of cars and another column named *Dist* that measured the distance of cars taken to stop.


GradQuant
    ```

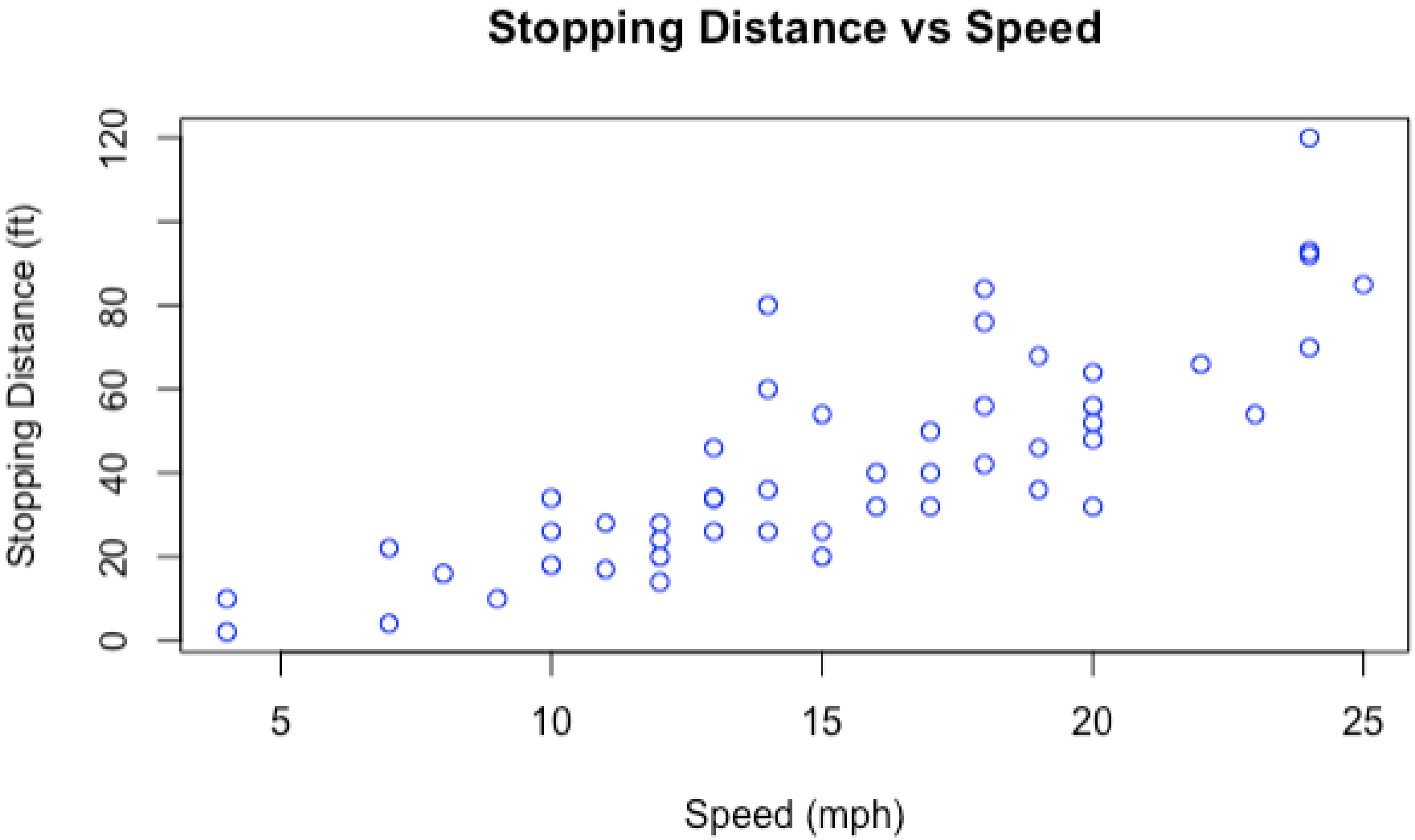
# Simple Linear Model

Reading the documentation in help window we learn that this dataset is used to analysis speed and stopping distances of cars. The interesting task here is to determine how far a car travels before stopping, when traveling at a certain speed.

- We create a scatter plot the check whether there exists any linear or non-linear relationships between speed and distance.
  - 
```
> plot(dist ~ speed, data = cars,
+      xlab = "Speed (mph)",
+      ylab = "Stopping Distance (ft)",
+      main = "Stopping Distance vs Speed",
+      col  = "blue")
```

# Simple Linear Model

- Here is the plot shown in the Plots window.



# Simple Linear Model

- In R, there is a function named **lm()** that fit a simple linear model for dataset.
- **lm()** function is a built-in function in **stats** package. So we need to call the library first.
- Distance is our response variable and our independent variable is the speed.

```
library(stats)
mod1 <- lm(dist ~ speed)
summary(mod1)
```

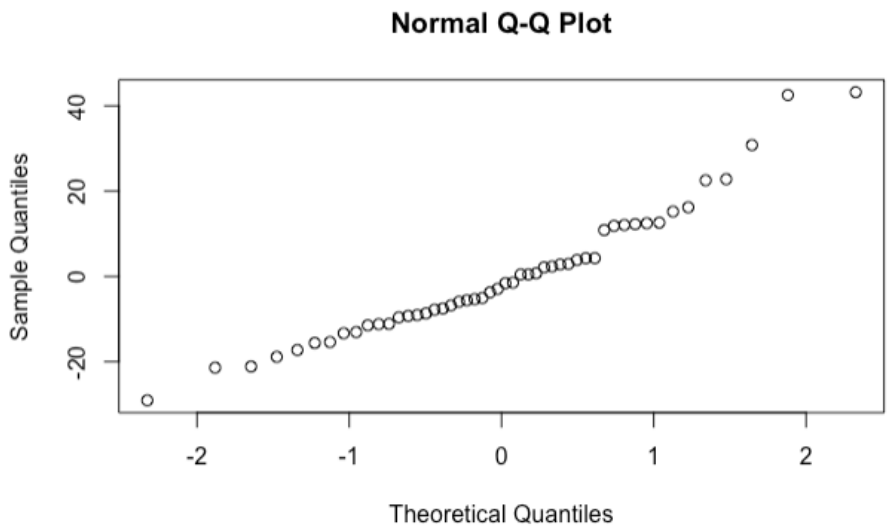
# Simple Linear Model

- How can we read the summary result from R?

```
Call:
lm(formula = dist ~ speed)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

- 

# Simple Linear Model

- After you fit a liear model, do not forget to check the residuals!!!
- In R, there are two ways to check the normality assumption of residuals.
  - Normal QQ plot.
  - 
```
> qqnorm(mod1$residuals)
```
  - 

# Simple Linear Model

- Another way in R to check the normality assumption is Shapiro-Wilk test.

```
> shapiro.test(mod1$residuals)

        Shapiro-Wilk normality test

data:  mod1$residuals
W = 0.94509, p-value = 0.02152
```

# Related Workshops

- **Data Manipulation in R**
- November 10, 10:00 to 11:50 AM.
  - We will discuss how to clean and organize data in R and how to create, subset, reshape, and order a data frame or a list.
- **Data Visualization in R**
- December 9, 14:00 to 15:50.
  - We will show how to use R to plot fantastic graphics.
- Please check the detailed description via `https://gradquant.ucr.edu` and help yourself register via `https://ucr.mywconline.com`.

# Services in GradQuant

- GradQuant offers individual consultations on Skype. Always welcome to make an appointment with us.
- Weekly Drop-in Hours on Monday from 1 pm to 3 pm. Graduate students and postdocs to meet with GradQuant consultants without an appointment, on a 'first-come, first-served' basis.
- Weekly Hacky Hours on Tuesday from 2 pm to 4 pm. Hacky Hours open to the whole campus, serving undergraduate and graduate students, faculty, and staff. No appointment is required.
- For detailed information about how to make an appointment and attend drop-in hours, you can visit via `https://gradquant.ucr.edu`.

# Question?