Overview
Using R
Exercise 1
R Language
Importing Data
Exercise 2
Managing Data
Graphics in R
Exercise 3
t Tests
ANOVA
Regression
Final Words

R: Introduction Continued

Okan Bulut

Centre for Research in Applied Measurement and Evaluation (CRAME)

February 28, 2020

(http://bit.ly/rworkshop2020)

Workshop Details

- This workshop will introduce participants to data science procedures using **R** (`https://cran.r-project.org/`) and **RStudio** (`https://rstudio.com/`) that are widely used in social sciences, public health, and other similar areas.

- My contact information:
  - Dr. Okan Bulut
  - Program: Measurement, Evaluation, and Data Science
  - Email: bulut@ualberta.ca
  - Website: https://okan.cloud

- Workshop materials can be downloaded from:
  - http://bit.ly/rworkshop2020

Outline

1. Part I: An overview of **R** and **RStudio**
    - What is **R**?
    - What is **RStudio**?
    - Basics of the **R** language
    - Importing Data into R
2. Break
3. Part II: Descriptive and inferential statistics with **R**
    - Data Management in in **R**
    - Data visualization in **R**
    - Inferential statistics (e.g., t tests, correlation, regression)

Learning Path in **R** Programming

"Patience is bitter, but its fruit is sweet." – Aristotle

What is R?

R:

- is a free, open source program for statistical computing and data visualization.
- is cross-platform (e.g., available on Windows, Mac OS, and Linux).
- is maintained and regularly updated by the Comprehensive R Archive Network (CRAN; https://cran.r-project.org/).
- is capable of running all types of statistical analyses.
- has amazing visualization capabilities (high-quality, customizable figures).
- enables reproducible research.
- has many other capabilities, such as web programming.
- supports user-created packages (currently, more than 10,000)

Some R Resources (1)

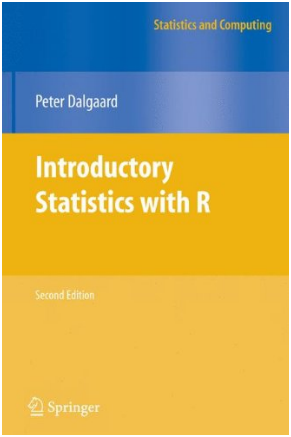
Websites and e-books:

- An Introduction to R
- Using R for Introductory Statistics
- R for SAS and SPSS Users
- Quick R: `https://www.statmethods.net/`
- R Cookbook: `http://www.cookbook-r.com/`
- R for Data Science: `https://r4ds.had.co.nz/`

Training:

- Statistical Analysis and Visualizations Using R (U of A)
- Coursera
- DataCamp

Some R Resources (2)



Statistics and Computing

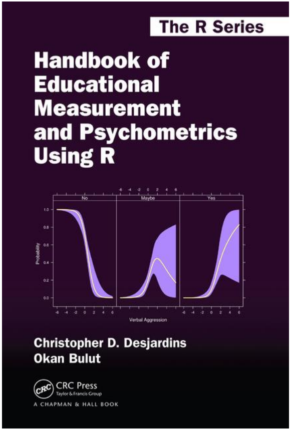
Peter Dalgaard

Introductory Statistics with R

Second Edition

Springer

https://goo.gl/zt7wc7



The R Series

Handbook of Educational Measurement and Psychometrics Using R

Christopher D. Desjardins
Okan Bulut

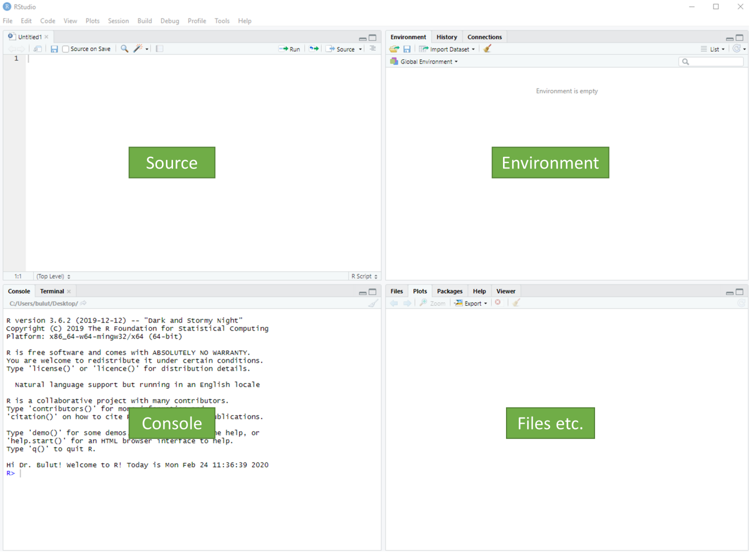
CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

https://goo.gl/Y6X3Hq

What is RStudio?

- **RStudio** is a free program available from the RStudio website.
- **RStudio** provides a more user-friendly interface for **R**.
- **RStudio** includes a set of tools to help you be more productive with **R**, such as:
  - A syntax-highlighting editor for highlighting your **R** codes
  - Functions for helping you type the **R** codes (auto-completion)
  - A variety of tools for creating and saving various plots (e.g., histograms, scatterplot)
  - A workspace management tool for importing or exporting data
- To benefit from **RStudio**, both **R** and **RStudio** should be installed in your computer.

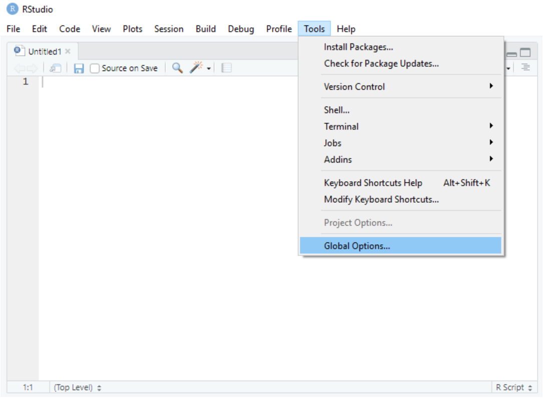
Preview of RStudio



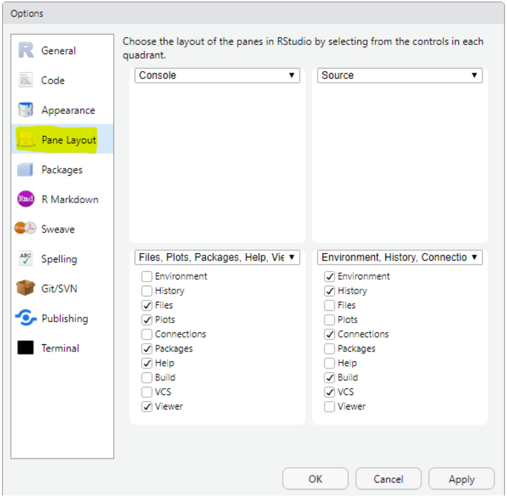
```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for mor
'citation()' on how to cite R                     ublications.

Type 'demo()' for some demos                     he help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Hi Dr. Bulut! Welcome to R! Today is Mon Feb 24 11:36:39 2020
R> 
```

Changing the Pane Layout

The pane layout can be updated using “Global Options”. I personally prefer console on the top-left, source on the top-right, files on the bottom-left, and environment on the bottom-right.



Changing the Pane Layout



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Console ▼

Source ▼

Files, Plots, Packages, Help, Vie ▼

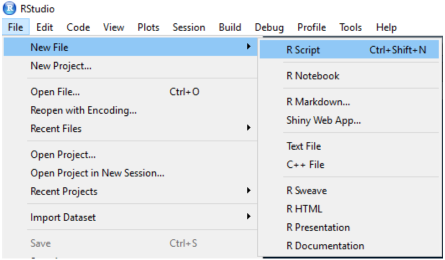
- ☐ Environment
- ☐ History
- ☑ Files
- ☑ Plots
- ☐ Connections
- ☑ Packages
- ☑ Help
- ☐ Build
- ☐ VCS
- ☑ Viewer

Environment, History, Connectio ▼

- ☑ Environment
- ☑ History
- ☐ Files
- ☐ Plots
- ☑ Connections
- ☐ Packages
- ☐ Help
- ☑ Build
- ☑ VCS
- ☐ Viewer

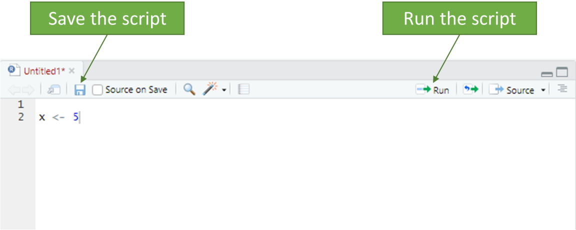
Creating a New Script

- In **R**, we can type our commands in the console; but once we close **R**, everything we have typed will be gone. Therefore, we should create an empty script, write the codes in the script, and save it for future use.
- The **R** script file has the .R extension, but it is essentially a text file. Thus, any text editor (e.g., Microsoft Word) can be used to open a script file.



Using the Script

When we type some codes in the script, we can select the lines we want to run and then hit the run button.



Save the script

Run the script

```
x <- 5
```

Alternatively, we can bring the cursor at the beginning of the line and hit the run button (which runs only a single line).

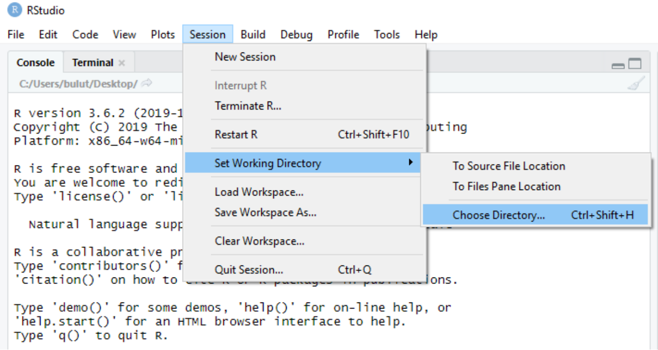
Working Directory

- An important feature of **R** is “working directory”, which refers to a location or a folder in your computer where you keep your **R** script, your data files, etc.

- Once we define a working directory in **R**, any data file or script within that directory can be easily imported into **R** without specifying where the file is located.

- We can set the working directory in two ways:
  1. Using the “Session” options menu in **RStudio**
  2. Using the `setwd` command in the console

Working Directory



C:/Users/bulut/Desktop/

```
R version 3.6.2 (2019-1
Copyright (C) 2019 The                                      puting
Platform: x86_64-w64-mi

R is free software and
You are welcome to redi
Type 'license()' or 'li

  Natural language supp

R is a collaborative pr
Type 'contributors()' f
'citation()' on how to                                     ons.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Working Directory

- Typing the following code in the console will set the "workshop2020" folder on my desktop as the working directory. If the folder path is correct, **R** changes the working directory without giving any error messages in the console.

```
setwd("C:/Users/bulut/Desktop/workshop2020")
```

- To ensure that the working directory is properly set, we can use the `getwd` command:

```
getwd()
```

- Note that **R** does not accept any backslashes in the file path. Instead of a backslash, we need to use a front slash. This is particularly important for Windows computers since the file paths involve backslashes (Mac OS X doesn't have this problem).

Installing Packages

- The base **R** program comes with many built-in functions to compute a variety of statistics and to create graphics (e.g., histograms, scatterplots, etc.)

- What makes **R** more powerful than other software programs is that **R** users can write functions and share them with other **R** users via the CRAN website.

- For example, **ggplot2** is a well-known **R** package, created by Hadley Wickham and Winston Chang. This package allows users to create elegant data visualizations. To download and install the **ggplot2** package, we need to use the `install.packages` command.

```
install.packages("ggplot2")
```

Installing Packages

- Once a package is downloaded and installed, it is permanently in your **R** folder (no need to re-install it).

- Although a package is already in the **R** folder, it is not accessible until we activate it.

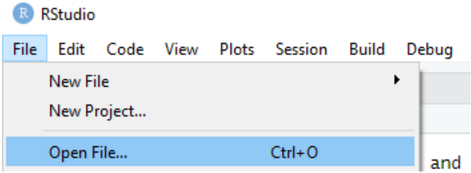
- Whenever we need to access a package in **R**, the `library` command should be used.

- For example, to access the **ggplot2** package, we would use:

```
library("ggplot2")
```

Exercise 1

- In **RStudio**, open "workshop2020script" using File –> Open File:



- Next, you will set the "workshop2020" as your working directory using either the `setwd` command or the Session options menu in **RStudio**.

- Make sure that you've successfully changed the working directory using in the console:

```
getwd()
```

Exercise 1

- Lastly, install and activate the **lattice** and **ggplot2** packages using the `install.packages` and `library` commands.

```
install.packages("lattice")
install.packages("ggplot2")

library("lattice")
library("ggplot2")
```

- You can simply select these lines in the workshop2020script file that you've just opened and hit the “Run” button.

Creating Variables in R

To create a new variable in **R**, we use the assignment operator “<-” or the equal sign “=”. To create a variable “x” that equals 1, we need to type:

```
x <- 1
```

If we want to print x, we just type “x” in the console and hit enter. **R** returns the value assigned to x.

```
x

[1] 1
```

We can also create a variable that holds multiple values in it, using the “c” command (c standards for “combine”).

```
weight <- c(60, 72, 80, 84, 56)
weight

[1] 60 72 80 84 56

height <- c(1.70, 1.75, 1.80, 1.90, 1.60)
height

[1] 1.70 1.75 1.80 1.90 1.60
```

Creating Variables in R

Once we create a variable, we can do further calculations with it. Let's say we want to transform the weight variable (in kg) to a new variable called weight2 (in lbs).

```
weight2 <- weight*2.20462
weight2

[1] 132.3 158.7 176.4 185.2 123.5
```

Note that we named the variable as "weight2". So, both weight and weight2 exist in the active **R** session now. If we used the following, this would overwrite the existing "weight" variable.

```
weight <- weight*2.20462
```

We can define a new variable based on the existing variables.

```
reading <- c(80, 75, 50, 44, 65)
math <- c(90, 65, 60, 38, 70)
total <- reading + math
total

[1] 170 140 110  82 135
```

Creating Variables in R

If a variable is not numerical, we need to use double quotation marks. In the example below, we create a new variable called “cities” with four city names. Each city name is written with double quotation marks.

```r
cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
cities
```

```
[1] "Edmonton"     "Calgary"      "Red Deer"     "Spruce Grove"
```

We can also treat numerical values as character strings. For example, assume that we have a gender variable where 1=Male and 2=Female. We want **R** to know that these values are not actual numbers; instead, they are just numerical labels for gender groups.

```r
gender <- c("1", "2", "2", "1", "2")
gender
```

```
[1] "1" "2" "2" "1" "2"
```

Some Rules...

R is case-sensitive!

R codes written in lowercase would **NOT** refer to the same codes written in uppercase.

```
cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
Cities
CITIES
Error: object 'Cities' not found
Error: object 'CITIES' not found
```

Variable names in **R**

- A variable name **CAN'T** begin with a number.
- A variable name **CAN'T** include a space.

```
4cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
my cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
Error: unexpected symbol in "4cities"
Error: unexpected symbol in "my cities"
```

Some Rules...

Naming conventions in **R**

- All lowercase: e.g. mycities
- Period.separated: e.g. my.cities
- Underscore_separated: e.g. my_cities
- Numbers at the end: e.g. mycities2018
- Combination of some of these rules: my.cities.2018

- Not to create messy code that is difficult to read and understand, I recommend using consistent and clear naming conventions.
- I personally prefer all lowercase with underscore (e.g., my_variable).

Some Rules...

Commenting in **R**

The hashtag symbol (#) is used for commenting in **R**. Any words, codes, etc. coming after a hashtag are just ignored.

```r
# Here I define four cities in Alberta
cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
```

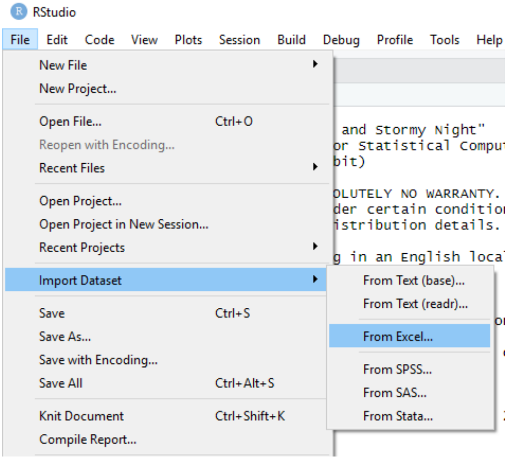
- I strongly recommend using comments throughout your codes.
- These annotations would remind you what you did and why you did it that way.
- You can easily comment out a line without having to remove it from your codes.

```r
#cities <- c("Edmonton", "Calgary", "Red Deer", "Spruce Grove")
```

Importing Data into R

- We often save our data sets in convenient data formats, such as Excel, SPSS, or text files (.txt, .csv, .dat, etc.).
- **R** is capable of importing (i.e., reading) various data formats.
- There are two possible ways to read a data file in **R**:
  1. By using the "Import Dataset" menu option
  2. By reading the file with an **R** command
     - **R** has some built-in functions, such as read.csv and read.table
     - There are some **R** packages for specific data formats; "foreign" for SPSS files and "xlsx" for Excel files

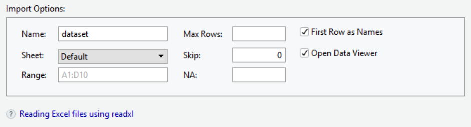
Importing Data into R



```
 and Stormy Night"
or Statistical Compu
bit)

OLUTELY NO WARRANTY.
der certain conditio
istribution details.

g in an English loca
```

Importing Excel and SPSS Files

- Give a name for the data set
- Choose the sheet to be imported
- “First Row as Names” if the variable names are in the first row

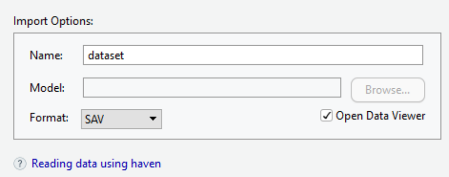


Import Options:

Name: dataset
Sheet: Default
Range: A1:D10
Max Rows:
Skip: 0
NA:
☑ First Row as Names
☑ Open Data Viewer

Reading Excel files using readxl

- Give a name for the data set
- Choose the SPSS data format (SAV)



Import Options:

Name: dataset
Model:
Format: SAV
☑ Open Data Viewer

Reading data using haven

Importing Data Files with R Commands

Excel files:

```
install.packages("xlsx")
library("xlsx")

my_excel_file <- read.xlsx("path to the file/filename.xlsx",
                           sheetName="sheetname")
```

SPSS files:

```
install.packages("foreign")
library("foreign")

my_spss_file <- read.spss("path to the file/filename.sav",
                          to.data.frame=TRUE)
```

Comma-separated files:

```
my_csv_file <- read.csv("path to the file/filename.csv", header=TRUE)
```

- `header=TRUE` if the variable names are in the first row

Exercise 2

- In the workshop materials, you will find two data files:
  - `medical.xlsx` (Excel)
  - `medical.csv` (Comma Separated Values)
- The `medical` dataset comes from a clinical trial for adult inpatients recruited from a detoxification unit.
- Patients received either a multidisciplinary assessment and a brief motivational intervention or usual care.
- The dataset has several demographic variables and scores from physical, mental, and depression assessments for the patients.
- **Source:** https://nhorton.people.amherst.edu/help/

Exercise 2

- Using the codes under "Exercise 2", you will read the `medical` dataset in **R**.

- You can use:
  - the codes to read in either medical.csv or medical.xlsx, or
  - use the "Import Dataset" menu option in **RStudio**.

- If there is no error messages in the console, then your dataset has been properly read into **R**!

Viewing and Indexing Data

- Once a dataset is imported into **R**, we should ensure that **R** was able to read the data properly. The `head` command prints out the first six rows of the dataset.

```
medical <- read.csv("medical.csv", header=TRUE)
head(medical)
```

- To see how many columns and rows exist in the dataset, we can use the `dim` (i.e., dimension) command:

```
dim(medical)

[1] 246  16
```

- The medical dataset has 246 rows (i.e., individuals) and 16 columns (i.e., variables).

Viewing and Indexing Data

- The `str` command prints the content of the dataset.

```
str(medical)

'data.frame': 246 obs. of  16 variables:
 $ id         : int  1 2 3 4 5 6 8 9 10 12 ...
 $ age        : int  37 37 26 39 32 47 28 50 39 58 ...
 $ sex        : chr  "male" "male" "male" "female" ...
 $ race       : chr  "black" "white" "black" "white" ...
 $ homeless   : chr  "housed" "homeless" "housed" "housed" ...
 $ substance  : chr  "cocaine" "alcohol" "heroin" "heroin" ...
 $ avg_drinks : int  13 56 0 5 10 4 12 71 20 13 ...
 $ max_drinks : int  26 62 0 5 13 4 24 129 27 13 ...
 $ suicidal   : chr  "yes" "yes" "no" "no" ...
 $ treat      : chr  "yes" "yes" "no" "no" ...
 $ physical1  : num  58.4 36 74.8 61.9 37.3 ...
 $ mental1    : num  25.11 26.67 6.76 43.97 21.68 ...
 $ depression1: int  49 30 39 15 39 6 32 50 46 49 ...
 $ physical2  : num  54.2 59.6 58.5 46.6 31.4 ...
 $ mental2    : num  52.2 41.7 56.8 14.7 40.7 ...
 $ depression2: int  7 11 14 44 26 23 18 33 37 8 ...
```

Viewing and Indexing Data

- If we want to see a specific variable, we can either specify the dataset name (followed by $) and the variable name.

- For example, we can print the first six rows of the variable “age” as follows:

```
head(medical$age)

[1] 37 37 26 39 32 47
```

- It is also possible to view multiple variables together.

```
head(medical[,c("age", "sex", "homeless")])

  age    sex homeless
1  37   male   housed
2  37   male homeless
3  26   male   housed
4  39 female   housed
5  32   male homeless
6  47 female   housed
```

Viewing and Indexing Data

- In **R**, square brackets ([<rows>, <columns>]) are used for indexing. Within the square brackets, the first part shows the row number(s) and the second part shows the column(s).

- For example; if we wanted to see the second variable for the third and fifth persons in the dataset, then we would do:

```
medical[c(3,5), 2]

[1] 26 32
```

- Or, if we wanted to see the first three variables for the first five persons, we would do:

```
medical[1:5, 1:3]

  id age    sex
1  1  37   male
2  2  37   male
3  3  26   male
4  4  39 female
5  5  32   male
```

Viewing and Indexing Data

- Instead of `medical[1:5,1:3]`, we could also use:

```
medical[c(1,2,3,4,5),c(1,2,3)]

  id age    sex
1  1  37   male
2  2  37   male
3  3  26   male
4  4  39 female
5  5  32   male
```

or

```
medical[1:5, c("id", "age", "sex")]

  id age    sex
1  1  37   male
2  2  37   male
3  3  26   male
4  4  39 female
5  5  32   male
```

Subsetting Data

- Assume that we want to create a new dataset with only females from the medical dataset. We can subset the females in two ways:

```
medical_female <- subset(medical, sex=="female")
```

or

```
medical_female <- medical[medical$sex=="female", ]
```

- Both options make a conditional selection where sex is equal to "female" ("==" makes a conditional selection).

- Another example of subsetting with sex and age:

```
male_50 <- subset(medical, sex=="male" & age < 50)
```

Recoding Variables

- There are multiple ways of recoding variables in **R**.
- For a binary recoding, the `ifelse` command is very easy to use.
- Assume that we want to create a new age variable identifying those with age > 50:

```
medical$age2 <- ifelse(medical$age > 50, "50+", "50 or younger")
```

- To create three age groups, we may need a few `ifelse` statements:

```
medical$age3 <- ifelse(medical$age > 50, "50+",
                       ifelse(medical$age <= 50 & medical$age > 30, "31-50",
                       "30 or younger"))
```

Summarizing Data

- In **R**, there are many ways for summarizing the data.
- The `summary` command returns the min and max, mean, median, and first and third quartiles for numerical variables and frequencies for categorical variables.
- Using the same dataset, we can summarize four variables (age, sex, homeless, and depression1):

```
summary(medical[,c("age", "sex", "homeless", "depression1")])

      age            sex              homeless          depression1
 Min.   :20.0   Length:246         Length:246         Min.   : 1.0
 1st Qu.:31.0   Class :character   Class :character   1st Qu.:25.2
 Median :35.0   Mode  :character   Mode  :character   Median :34.0
 Mean   :36.3                                         Mean   :32.6
 3rd Qu.:41.0                                         3rd Qu.:41.0
 Max.   :60.0                                         Max.   :57.0
```

Frequency Tables

- For categorical variables, the `table` command can be used for creating frequency tables:

```
table(medical$sex)


female   male
    57    189
```

```
table(medical$homeless)


homeless   housed
     118      128
```

```
table(medical$sex, medical$homeless)


        homeless housed
  female       22     35
  male         96     93
```

Descriptive Statistics

- We can calculate mean, median, variance, standard deviation, minimum, and maximum for individual variables.

```r
mean(medical$age)
[1] 36.31
median(medical$age)
[1] 35
var(medical$age)
[1] 63.75
sd(medical$age)
[1] 7.984
min(medical$age)
[1] 20
max(medical$age)
[1] 60
```

Descriptive Statistics by Group

- We can also calculate the descriptive statistics for each value of a categorical variable (e.g., gender) using the `tapply` command.

```
# Mean age by sex
tapply(medical$age, medical$sex, mean)

female   male
 37.07  36.08

# Median age by sex
tapply(medical$age, medical$sex, median)

female   male
    35     36

# Standard deviation of depression scores by homeless status
tapply(medical$depression1, medical$homeless, sd)

homeless   housed
   12.22    11.92

# Variance of depression scores by homeless status
tapply(medical$depression1, medical$homeless, var)

homeless   housed
   149.4    142.1
```

Descriptive Statistics by Group

- For a detailed summary of variables, I strongly recommend the **skimr** package.

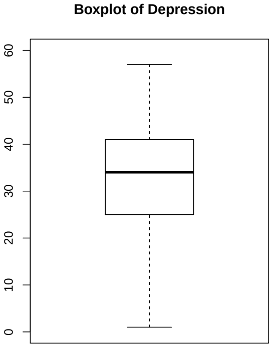
```r
install.packages("skimr")
library("skimr")
skim(medical[,c("age","sex","homeless","depression1","mental1")])
```

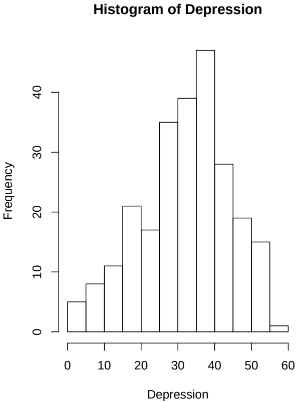
```
Skim summary statistics
 n obs: 246
 n variables: 5

-- Variable type:character -----------------------------------------------------
 variable missing complete   n min max empty n_unique
 homeless       0      246 246   6   8     0        2
      sex       0      246 246   4   6     0        2

-- Variable type:integer -------------------------------------------------------
    variable missing complete   n  mean    sd p0   p25 p50 p75 p100
         age       0      246 246 36.31  7.98 20 31      35  41   60
 depression1       0      246 246 32.59 12.11  1 25.25  34  41   57

-- Variable type:numeric -------------------------------------------------------
 variable missing complete   n  mean    sd   p0   p25   p50   p75  p100
  mental1       0      246 246 31.68 12.49 6.76 21.95 29.15 40.62 60.54
```

Base Graphics in R

Boxplots and histograms

```
boxplot(medical$depression1, main="Boxplot of Depression")
hist(medical$depression1, main="Histogram of Depression", xlab="Depression")
```

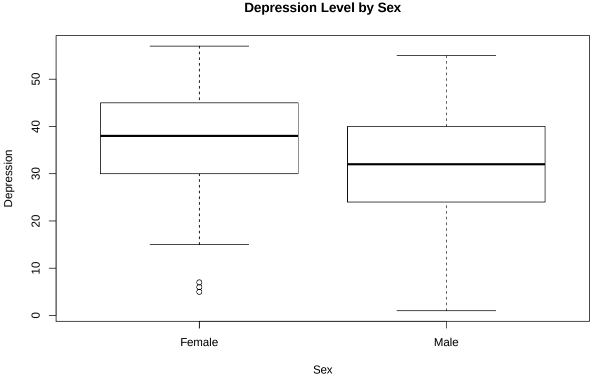




Base Graphics in R

- Boxplots by group

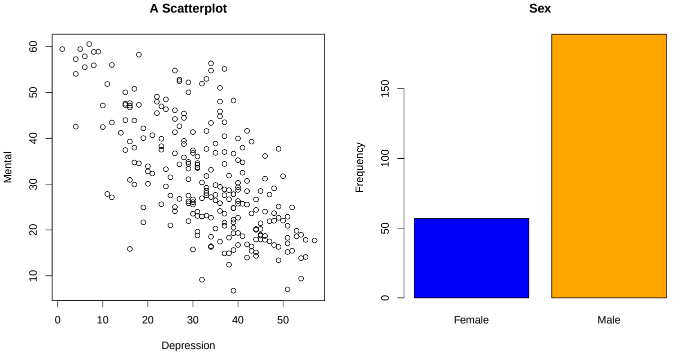
```
boxplot(medical$depression1 ~ medical$sex, xlab="Sex", ylab="Depression",
main="Depression Level by Sex", names = c("Female", "Male"))
```



Base Graphics in R

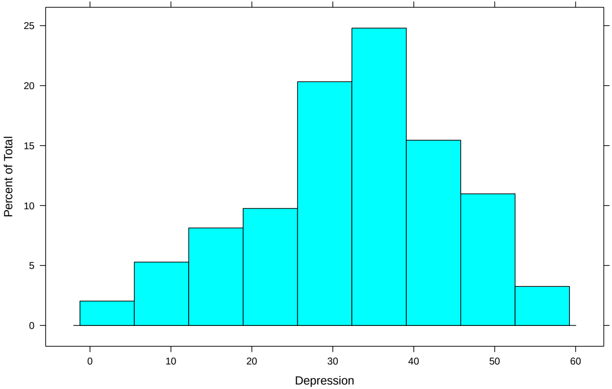
Scatterplots and bar plots

```
plot(medical$depression1, medical$mental1, main="A Scatterplot",
xlab="Depression", ylab="Mental")
barplot(table(medical$sex), main = "Sex", names = c("Female", "Male"),
ylab = "Frequency", col = c("blue", "orange"))
```



Graphics with lattice

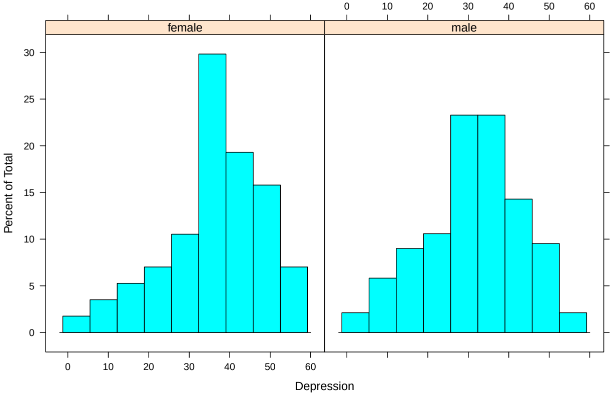
```r
library("lattice")
histogram(~ depression1, data = medical, xlab="Depression")
```



Graphics with lattice

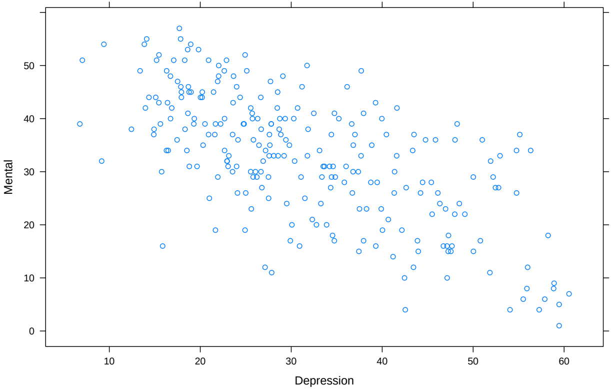
- Histograms by group

```
library("lattice")
histogram(~ depression1 | sex, data = medical, xlab="Depression")
```



Graphics with lattice

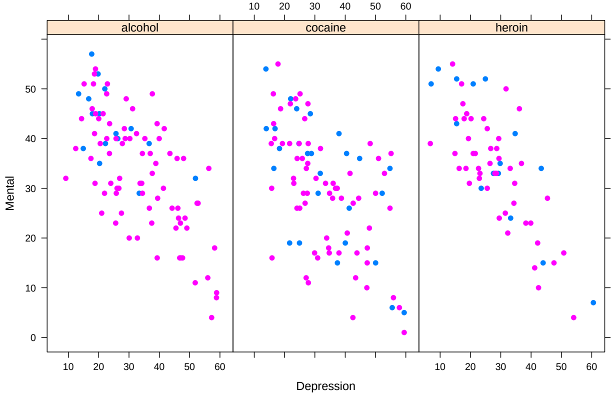
```r
xyplot(depression1 ~ mental1, data = medical, xlab="Depression", ylab="Mental")
```



Graphics with lattice

- Scatterplots by group

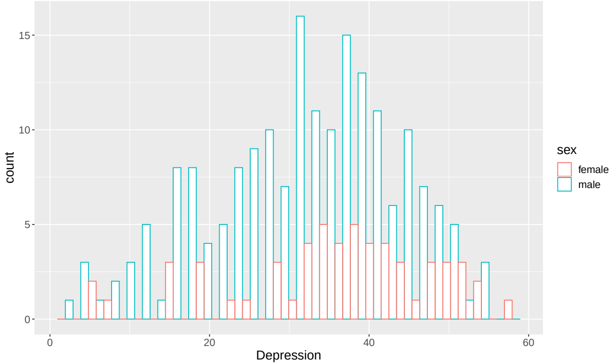
```
xyplot(depression1 ~ mental1 | substance, group = sex, data = medical,
xlab="Depression", ylab="Mental")
```



Graphics with ggplot2

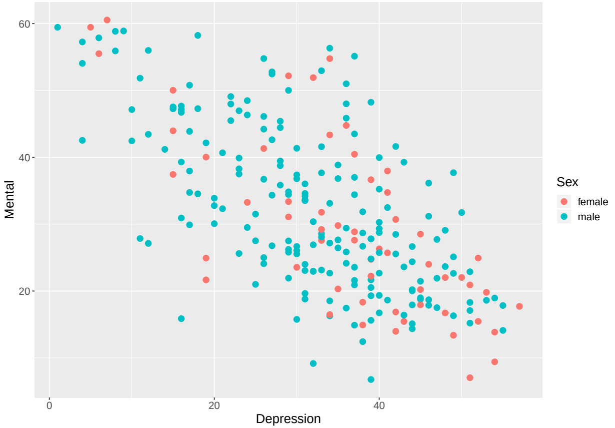
- The **ggplot2** package is capable of creating more elegant and complex graphics (check out http://ggplot2.tidyverse.org/).

```
library("ggplot2")
ggplot(data = medical, aes(x=depression1, color=sex)) +
geom_histogram(fill="white", position="dodge") + xlab("Depression")
```



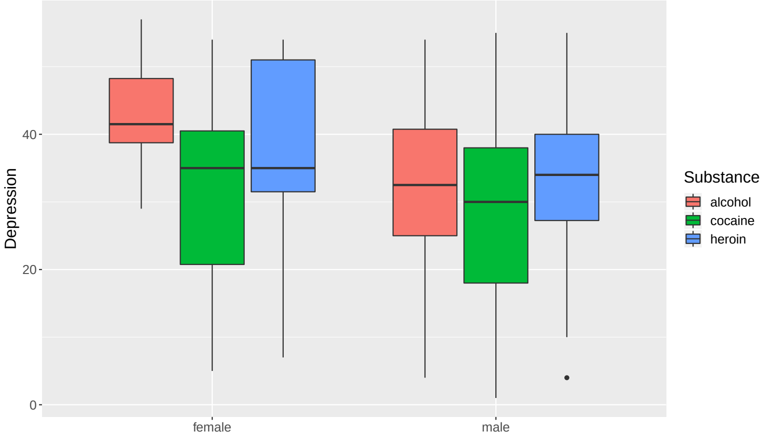
Graphics with ggplot2

```
library("ggplot2")
ggplot(data = medical, aes(depression1, mental1, colour = sex)) +
geom_point(size = 3) + labs(colour = "Sex", x = "Depression", y = "Mental")
```



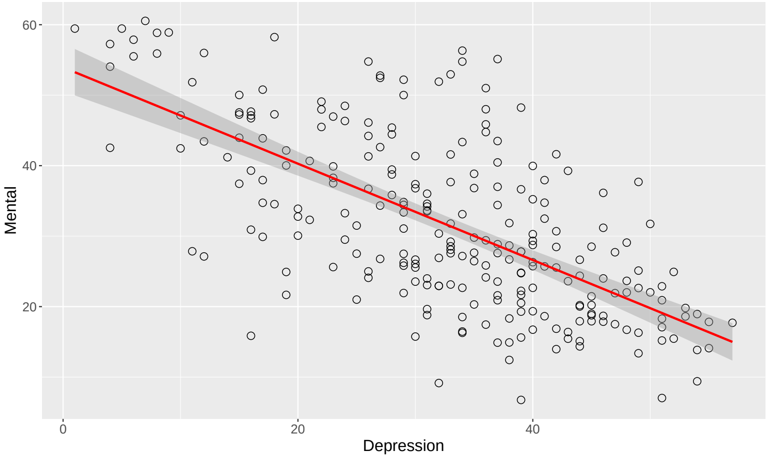
Graphics with ggplot2

```
ggplot(data = medical, aes(sex, depression1, fill=substance)) +
labs(x="" , y="Depression", fill="Substance") + geom_boxplot()
```



Graphics with ggplot2

```
ggplot(data = medical, aes(depression1, mental1)) + geom_point(shape=1, size=3) +
geom_smooth(method=lm , color="red", se=TRUE) + labs(x="Depression" , y="Mental")
```



Exercise 3

- Using the codes under “Exercise 3”, you will create two plots:

  1. A histogram of **depression2** by **substance**
  2. A scatterplot of **depression2** and **mental2**, colored by **substance**

- Label the axes clearly in each plot by using the variable names

Inferential Statistics

With hypothesis testing, we can calculate several inferential statistics:

1. Whether a sample mean is equal to a particular value:
   - One sample t test ($H_0 : \mu = \text{value}$)
2. Whether two sample means are equal:
   - Independent samples t test ($H_0 : \mu_1 = \mu_2$ or $H_0 : \mu_1 - \mu_2 = 0$)
   - Repeated measures t test ($H_0 : \mu_D = 0$ or $H_0 : \mu_1 - \mu_2 = 0$)
3. Whether three or more groups have equal means:
   - ANOVA ($H_0 : \mu_1 = \mu_2 = \mu_3 = \cdots = \mu_k$)
4. Whether two variables are correlated
5. Whether a dependent variable can be predicted by other variables

One Sample *t* Test

Hypothesis: Does the average depression score in the sample differ from the average depression score ($\mu = 25$) in the population?

```
t.test(medical$depression1, mu=25, conf.level = 0.95, alternative = "two.sided")


One Sample t-test

data:  medical$depression1
t = 9.8, df = 245, p-value <2e-16
alternative hypothesis: true mean is not equal to 25
95 percent confidence interval:
 31.06 34.11
sample estimates:
mean of x
    32.59
```

Conclusion: With the significance level of $\alpha = .05$, we reject the null hypothesis that the average depression score in the medical dataset is the same as the average depression score in the population, $t(245) = 9.8,\ p < .05,\ CI_{95} = [31.06, 34.11]$.

Independent Samples *t* Test

Hypothesis: Does the average depression score differ between males and females?

```
male <- medical[medical$sex=="male", "depression1"]
female <- medical[medical$sex=="female", "depression1"]
t.test(male, female, conf.level = 0.95, alternative = "two.sided")


Welch Two Sample t-test

data:  male and female
t = -2.5, df = 87, p-value = 0.02
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8.418 -0.928
sample estimates:
mean of x mean of y
    31.50     36.18
```

Conclusion: With the significance level of $\alpha = .05$, we reject the null hypothesis that the average depression score is the same for males and females, $t(87) = 2.5$, $p < .05$.

Paired Samples *t* Test

Hypothesis: Does the average depression score differ between time 1 and time 2?

```
t.test(medical$depression1, medical$depression2, paired = TRUE,
conf.level = 0.95, alternative = "two.sided")


Paired t-test

data:  medical$depression1 and medical$depression2
t = 11, df = 245, p-value <2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  8.021 11.719
sample estimates:
mean of the differences
                   9.87
```

Conclusion: With the significance level of $\alpha = .05$, we reject the null hypothesis that the average depression score is the same between time 1 and time 2, $t(245) = 11$, $p < .05$.

Analysis of Variance (ANOVA)

Hypothesis: Do substance groups differ in their depression levels?

```
tapply(medical$depression1, medical$substance, mean)

alcohol cocaine  heroin
  34.32   29.69   34.23

anova(lm(medical$depression1 ~ medical$substance))

Analysis of Variance Table

Response: medical$depression1
                   Df Sum Sq Mean Sq F value Pr(>F)
medical$substance   2   1209     605    4.23  0.016 *
Residuals         243  34733     143
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Conclusion: With the significance level of $\alpha = .05$, we reject the null hypothesis that the depression level is the same across three substance groups.

Correlations

What are the correlations among the depression, mental, and physical assessment scores, the average number of drinks, and age?

```
cor(medical[,c("depression1","mental1","physical1", "avg_drinks", "age")],
    method = "pearson")

            depression1  mental1 physical1 avg_drinks      age
depression1     1.00000 -0.66289  -0.32000    0.11794 -0.01374
mental1        -0.66289  1.00000   0.05698    0.01601  0.06336
physical1      -0.32000  0.05698   1.00000   -0.20996 -0.23111
avg_drinks      0.11794  0.01601  -0.20996    1.00000  0.27921
age            -0.01374  0.06336  -0.23111    0.27921  1.00000
```

- Now assume that a researcher wants to predict depression scores using the variables in the correlation table above.
- Except for age, the other three variables (mental1, physical1, and avg_drinks) appear to be correlated with depression1.

Regression

We should run a regression model to predict depression1.

```
model <- lm(depression1 ~ mental1 + physical1 + avg_drinks, data = medical)
summary(model)


Call:
lm(formula = depression1 ~ mental1 + physical1 + avg_drinks,
    data = medical)

Residuals:
    Min      1Q  Median      3Q     Max
-22.744  -5.745   0.108   5.605  20.899

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  65.5275     2.7862   23.52  < 2e-16 ***
mental1      -0.6294     0.0431  -14.60  < 2e-16 ***
physical1    -0.2886     0.0490   -5.90  1.2e-08 ***
avg_drinks    0.0411     0.0259    1.59     0.11
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.41 on 242 degrees of freedom
Multiple R-squared:  0.524, Adjusted R-squared:  0.518
F-statistic: 88.9 on 3 and 242 DF,  p-value: <2e-16
```

Regression

The output from the regression model shows that:

- mental1 is a **significant**, negative predictor of depression1.
- physical1 is a **significant**, negative predictor of depression1.
- avg_drinks is a **not** significant predictor of depression1.
- The overall regression model is **significant**, $F(3,22) = 88.9, p < .05$.
- $R^2 = 0.524$ suggests that our model explains 52.4% of the total variation in the depression scores.

Other Statistical Methods in R

- Regression for categorical variables (e.g., logistic regression, ordinal regression)
- Longitudinal model
- Multilevel (i.e., hierarchical) models
- Mixed-effect models
- Factor analysis and its variants
- Structural equation modeling
- And tons of other things...

Other Software Working with R

- markdown (https://rmarkdown.rstudio.com/) to create fully reproducible and elegant HTML-style documents
- shiny (https://shiny.rstudio.com/) to create interactive applications and dashboards with **R**
- plotly (https://plot.ly/r/) to create interactive data visualizations
- H2O (https://www.h2o.ai/) to apply advanced machine learning algorithms
- Apache Spark (https://spark.apache.org/) and Hadoop (https://hadoop.apache.org/) for processing and modeling big data

THANK YOU!

For questions/comments: bulut@ualberta.ca