EPIB 607: Inferential Statistics

Sahir Bhatnagar and James Hanley

2018-09-04

Contents

Part I

Preface

Welcome

Welcome to the course notes for EPIB 607: Inferential Statistics at McGill University.

Objectives

The aim of this course is to provide students with basic principles of statistical inference so that they can:

1. Understand the statistical methods section in a scientific paper.
2. Apply statistical methods in their own research.
3. Use the methods learned in this course as a foundation for more advanced biostatistics courses.

Audience

The principle audience is researchers in the natural and social sciences who have a basic understanding of differentiable and integral calculus, but haven't had an introductory course in statistics. This audience accepts that statistics has penetrated the life sciences pervasively and is required knowledge for both doing research and understanding scientific papers.

About these notes

These notes are a collection of useful links, videos, online resources and papers for an introductory course in statistics. The instructors have found that no single book sufficiently teaches all the topics covered in this course. Part of this is due to advancements in computing which have far outpaced the publication of modern textbooks. Indeed, the computer has replaced many of the calculations that were traditionally taught to be done by hand. We direct the readers to what we think is a good learning resource for a given topic (following the **Flipped Classroom** strategy). We also provide our own commentary and notes when we think its useful.

R Code Conventions

We use R code throughout these notes. When R code is displayed[1] it will be typeset using a `monospace` font with syntax highlighting enabled to ensure the differentiation of functions, variables, and so on. For example, the following adds 1 to 1

[1] https://raw.githubusercontent.com/coatless/spm/master/index.Rmd

Figure 1

```
a = 1L + 1L
a
```

Each code segment may contain actual output from R. Such output will appear in grey font prefixed by #>. For example, the output of the above code segment would look like so:

```
[1] 2
```

Rendering Mathematical Formulae

Throughout these notes, there will be mathematical symbols used to express the material. Depending on the version of the book, there are two different rendering engines.

- For the online version, the text uses MathJax to render mathematical notation for the web. In the event the formulae does not load for a specific chapter, first try to refresh the page. 9 times out of 10 the issue is related to the software library not loading quickly. You can also right-click to see the corresponding LaTeX code used to produce the equation.

- For the pdf version, the text is built using the recommended AMS LaTeX symbolic packages. As a result, there should be no issue displaying equations. An example of a mathematical rendering capabilities would be given as:

$$a^2 + b^2 = c^2$$

Development

This book is built with **bookdown** and is open source and freely available. This approach encourages contributions, ensures reproducibility and provides access to the material worldwide. The online version of the book is hosted at sahirbhatnagar.com/EPIB607 and kept up-to-date thanks to Travis. The entire source code is available at `https://github.com/sahirbhatnagar/EPIB607`.

If you notice any errors, we would be grateful if you would let us know by filing an issue here or making a pull request by clicking the edit button in the top-left corner of the text:

The version of the book you are reading now was built on 2018-09-04 and was built on Travis.

About the authors

Sahir Bhatnagar James Hanley

- Sahir R. Bhatnagar: Assistant Professor of Biostatistics - McGill University, Montreal, Canada.
  - Website: https://sahirbhatnagar.com/
  - Twitter: syfi_24
  - GitHub: https://github.com/sahirbhatnagar
- James A. Hanley: Professor of Biostatistics - McGill University, Montreal, Canada.
  - Webpage: http://www.medicine.mcgill.ca/epidemiology/hanley/

License

Course Information

- Instructor: Sahir Bhatnagar
- Teaching Assistants: Kody Crowell, Guanbo Wang, Himasara Marasinghe
- Website: `http://sahirbhatnagar.com/EPIB607/`
- Lectures: Monday 11:30am - 1:30pm, Thursday 8:30am - 10:30am
- Location: McMed 1034
- Office Hours: TBD
- Prerequisite(s): Calculus and Algebra
- Texts: *The Practice of Statistics in the Life Sciences*, 3nd Edition by Baldi & Moore.

Teaching strategy

This course will follow the **Flipped Classroom** model. Here, students are expected to have engaged with the material before coming to class (based on very precise pre-class instructions). The students will then be expected to answer a series of conceptual multiple choice questions using the DALITE online platform (Bhatnagar et al., 2016).

This allows the instructor to delegate the delivery of basic content and definitions to textbooks and videos, and enforces the idea that students cannot be simply passive recipients of information. This approach then allows the professor to focus valuable class time on nurturing efficient discussions surrounding the ideas within the content, guiding interactive exploration of typical misconceptions, and promoting collaborative problem solving with peers.

A focus on computation

Classic introductory statistics textbooks were written during a time when computers were still in their infancy. As such, even the newer editions heavily rely on *by-hand* computations such as looking up tables for tail probabilities. We take a modern approach and introduce computational methods in statistics with the statistical software program R.

Figure 2

DataCamp

This class is supported by DataCamp, the most intuitive learning platform for data science. Learn R, Python and SQL the way you learn best through a combination of short expert videos and hands-on-the-keyboard exercises. Take over 100+ courses by expert instructors on topics such as importing data, data visualization or machine learning and learn faster through immediate and personalised feedback on every exercise.

You will be asked to complete some of the courses in DataCamp for background reading or for assignments. You can sign up for a free account at this link. Note: you are required to sign up with a `@mail.mcgill.ca` or `@mcgill.ca` email address.

Grade Distribution

Assignments	30%
DALITE Quizzes	10%
Midterm	20%
Project	10%
Final Exam	30%

Target Syllabus

Abbreviation	Description
JH	James Hanley notes
EM	Erica Moodie notes
OS	Olli Saarela notes
AAO	Against all odds video series
B&M	The practice of statistics in the life sciences by Baldi and Moore, 3rd edition
Freedman	Statistics by Freedman, Pisani, Purves, Adhikari, 2nd edition
dataviz	Fundamentals of Data Visualization by Claus O. Wilke

Descriptive Statistics

Topic	Video	Readings
Histograms	[AAO unit 3](https://www.learner.org/courses/againstallodds/unitpages/unit03.html)	[AAO unit 3, pages 1-6](htt
Density Plots	–	[dataviz chapter 7](https:/
Measures of Center	[AAO unit 4](https://www.learner.org/courses/againstallodds/unitpages/unit04.html)	[AAO unit 4, pages 1-6](htt
Boxplots	[AAO unit 5](https://www.learner.org/courses/againstallodds/unitpages/unit05.html)	[AAO unit 5, pages 1-5](htt
Standard Deviation	[AAO unit 6](https://www.learner.org/courses/againstallodds/unitpages/unit06.html)	[AAO unit 6, pages 1-3](htt
Data Visualization	[Hans Rosling BBC](https://www.youtube.com/watch?v=jbkSRLYSojo)	–

Sampling Distributions

Topic	Video	Readings
Parameters and Statistics	–	[[JH section 1]](htt
Sampling Distributions	[AAO unit 22](https://www.learner.org/courses/againstallodds/unitpages/unit22.html)	[[AAO unit 22, pag
Central Limit Theorem	[AAO unit 22](https://www.learner.org/courses/againstallodds/unitpages/unit22.html)	[B&M pages 321-3
The Bootstrap	–	[[Computer-Intensi

Tests of Null Hypotheses

Topic	Video	Readings
Confidence Intervals	[AAO unit 24](https://www.learner.org/courses/againstallodds/unitpages/unit24.html)	[[AAO unit
Tests of Significance and P-values	[AAO unit 25](https://www.learner.org/courses/againstallodds/unitpages/unit25.html)	[[AAO unit

One-sample Inference

Topic	Video	Readings
Means	[AAO unit 26](https://www.learner.org/courses/againstallodds/unitpages/unit26.html)	[[AAO unit 26, pages 1-11]](https:
Proportions	[AAO unit 28](https://www.learner.org/courses/againstallodds/unitpages/unit28.html)	[[AAO unit 28, pages 1-11]](https:
Rates	–	–

Two-sample Inference

Topic	Video	Readings
Means	[AAO unit 27](https://www.learner.org/courses/againstallodds/unitpages/unit27.html)	[[AAO unit 27, pages 1-11]](https:/
Proportions	–	[[B&M chapter 20]](https://www.c
Rates	–	–

Inference for Counts

Topic	Video
Chi-square test	[AAO unit 29](https://www.learner.org/courses/againstallodds/unitpages/unit29.htm
Mantel-Haenszel, Fisher's Exact, McNemar's Test	–

Regression

Topic	Video	Readings
Linear Regression	[AAO unit 30](https://www.learner.org/courses/againstallodds/unitpages/unit30.html)	[[AAO unit 30, p
Logistic, Poisson Regression	–	[[OS notes]](htt

Nonparametric Statistics

Topic	Video	Readings	Exer
Wilcoxon Signed Rank Test	–	[[JH notes]](http://www.medicine.mcgill.ca/epidemiology/hanley/c607/ch14/jh_ch_14.pdf)	–
Kruskal-Wallis Test	–	[[EM notes]](https://www.dropbox.com/s/hotrocov75sm7q8/InfStatPart5.pdf?dl=0)	–

Prerequisites

Git

You need to first install the git version control system. Follow Chapter 1: Installing Git for step-by-step installation instructions with screenshots.

R and RStudio

Complete the following DataCamp courses:

Topic	DataCamp Courses
Working with the RStudio IDE (Part 1) This short course will guide you through installing both R and RStudio. RStudio is a software application that facilitates how you interact with R.	
Introduction to R In this course you will get a hands-on introduction to the basic commands in R. With the knowledge gained in this course, you will be ready to perform a data analysis.	
Reporting with R Markdown You will learn how to create reproducible reports using R and Markdown. All assignments for this course must be submitted in this format.	
Version Control with RStudio IDE (Chapter 2 only) You will learn how to use RStudio to version control your code. All assignments for this course must be submitted to a GitHub repository.	

Part II

Part I

Chapter 1

Icebreakers

1.1 The Lady Tasting Tea

This example is adapted from *Start Teaching with R* (Pruim et al., 2015) and JH notes unit 8_B.

There is a famous story about a lady who claimed that tea with milk tasted different depending on whether the milk was added to the tea or the tea added to the milk. The story is famous because of the setting in which she made this claim. She was attending a party in Cambridge, England, in the 1920s. Also in attendance were a number of university dons and their wives. The scientists in at- tendance scoffed at the woman and her claim. What, after all, could be the difference? All the scientists but one, that is. Rather than simply dismiss the woman's claim, he proposed that they decide how one should test the claim. The tenor of the conversa- tion changed at this suggestion, and the scientists began to discuss how the claim should be tested. Within a few minutes cups of tea with milk had been prepared and presented to the woman for tasting. At this point, you may be wondering who the innova- tive scientist was and what the results of the experiment were. The scientist was R. A. Fisher, who first described

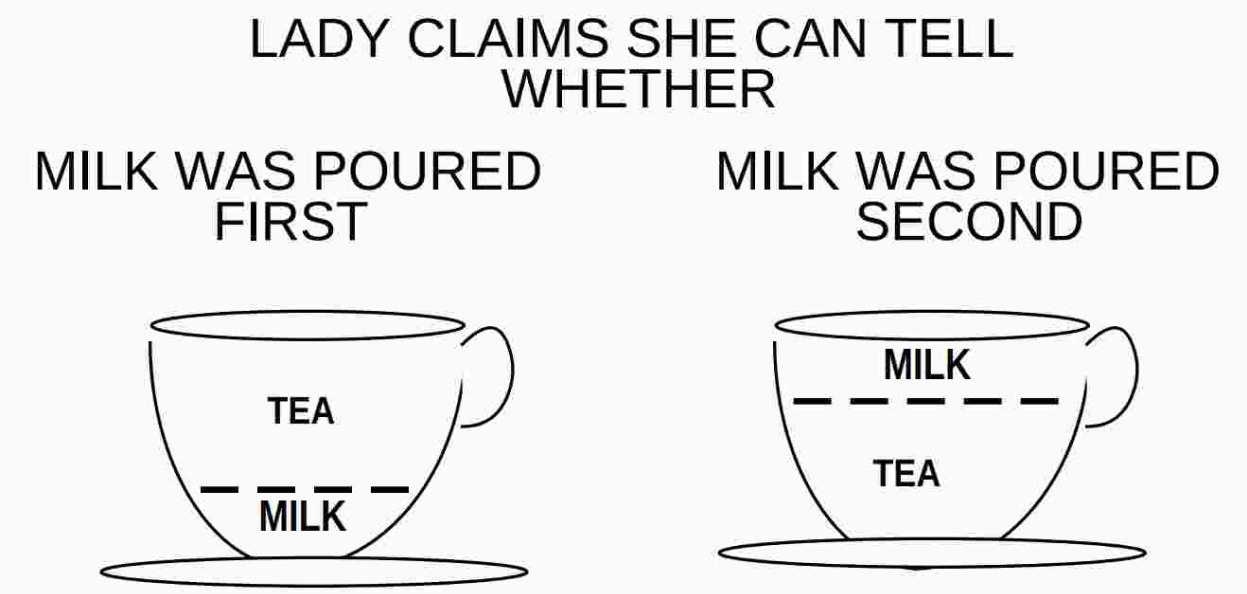


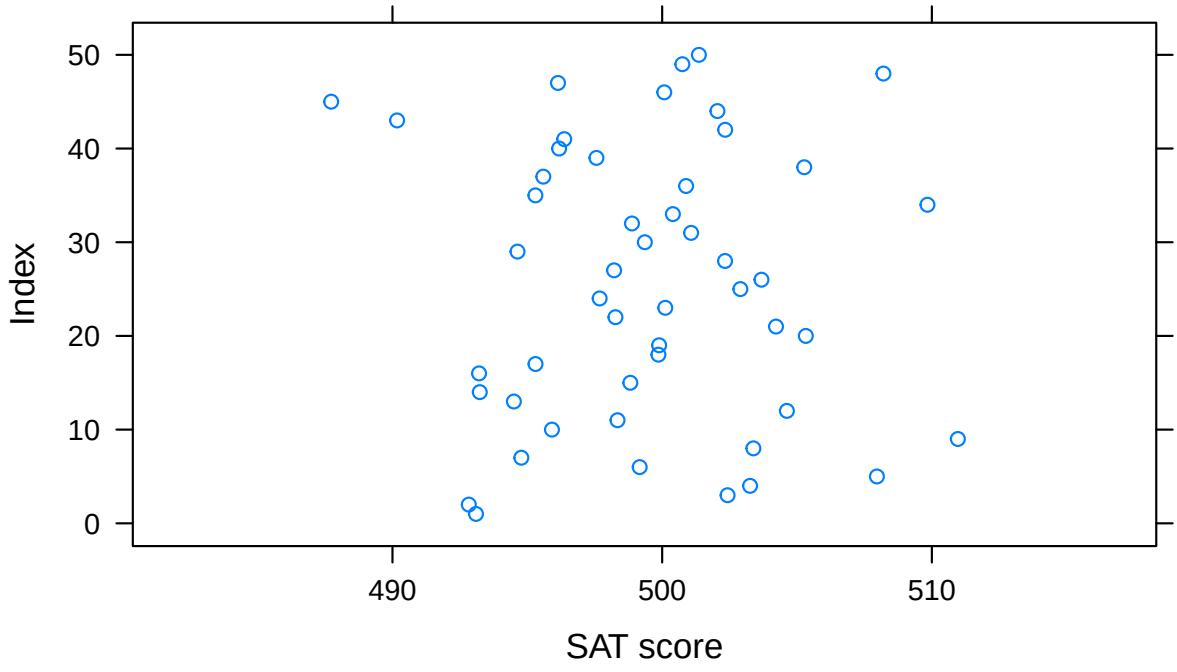
Figure 1.1

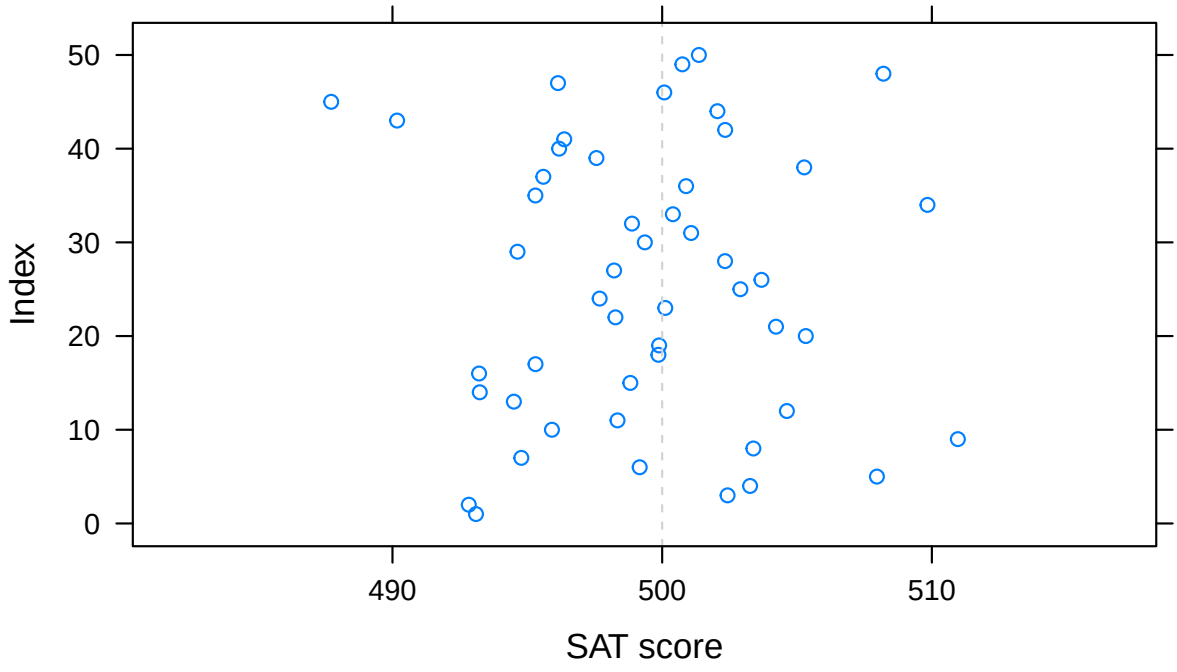
```
library(mosaic)
rflip()
#>
#> Flipping 1 coin [ Prob(Heads) = 0.5 ] ...
#>
#> H
#>
#> Number of Heads: 1 [Proportion Heads: 1]
```

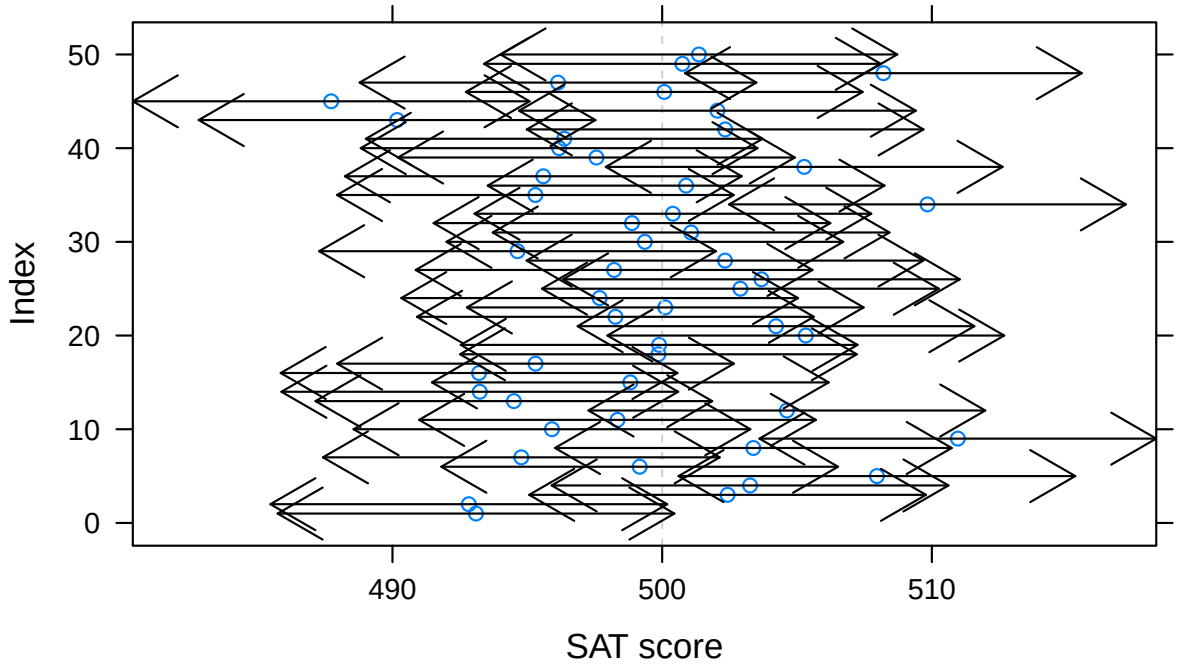
```
rflip(10)
#>
#> Flipping 10 coins [ Prob(Heads) = 0.5 ] ...
#>
#> H T T T T H T T T H
#>
#> Number of Heads: 3 [Proportion Heads: 0.3]
```

```
mu = 500
sigma = 100
x = rnorm(500, mean=mu, sd=sigma)
favstats(x)
#>  min    Q1 median    Q3   max  mean    sd   n missing
#>  117 431.4  497.4 563.1 811.6 500.2 99.16 500       0
meanconfint = function (x, sigma, level = 0.95, ...) {
  se = sigma / sqrt(length(x))
  mu = mean(x)
  z = qnorm(1 - (1 - level)/2)
  out = c(mu, mu - z * se, mu + z * se)
  names(out) = c("mean", "lower", "upper")
  return(out)
}
meanconfint(x, sigma = sigma)
#>  mean lower upper
#> 500.2 491.4 509.0
randomx = do(50) * rnorm(500, mean=mu, sd=sigma)
ci = data.frame(t(apply(randomx, 1, meanconfint, sigma=sigma, level=0.90)))
head(ci, 3)
#>    mean lower upper
#> 1 493.1 485.7 500.5
#> 2 492.8 485.5 500.2
#> 3 502.4 495.1 509.8

xyplot(1:nrow(ci) ~ mean, data=ci, xlim=range(ci), xlab="SAT score", ylab="Index")
ladd(panel.abline(v=500, col="lightgray", lty=2))
ladd(with(ci, panel.arrows(x0 = lower, y0=1:nrow(ci), y1=1:nrow(ci), cex=0.5,
                           x1=upper, code=3)))

head(t(apply(randomx, 1, meanconfint, sigma=sigma, level=0.9)), 3)
#>       mean lower upper
#> [1,] 493.1 485.7 500.5
#> [2,] 492.8 485.5 500.2
```

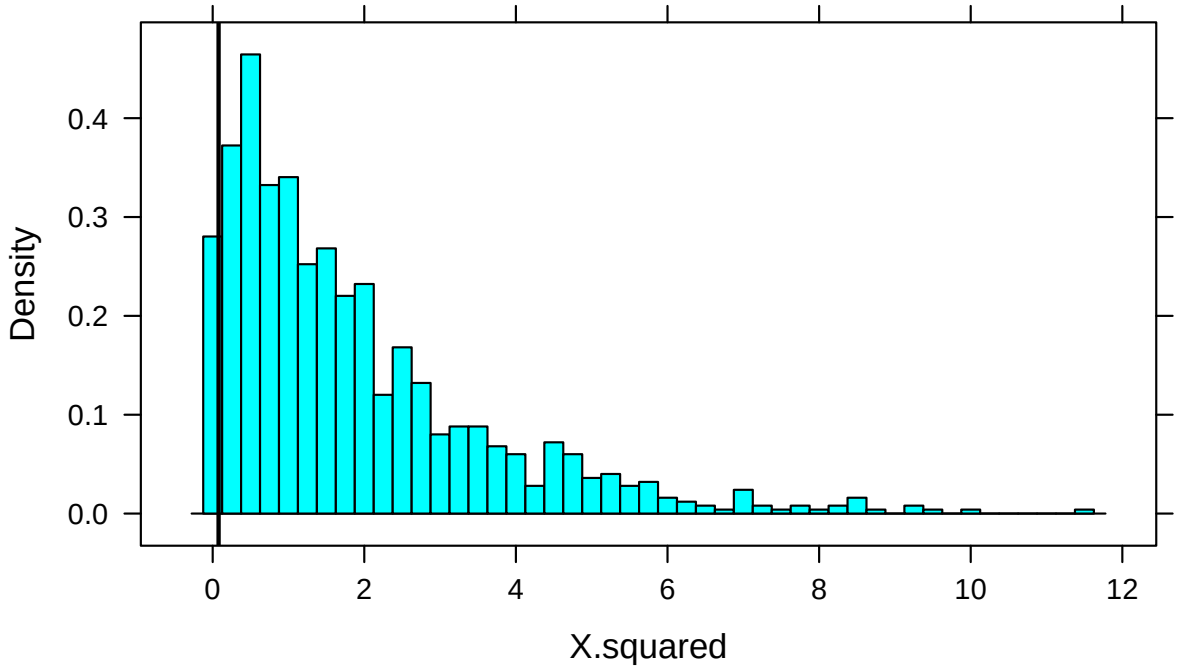
```
#> [3,] 502.4 495.1 509.8
```








```
T <- chisq(substance ~ shuffle(sex), data = HELPrct); T
#> X.squared
#>   0.07796
Substance.Null <- do(999) * chisq(substance ~ shuffle(sex), data = HELPrct)
histogram( ~ X.squared, data = Substance.Null, v = T, width = 0.25)
prop1( ~(X.squared >= T), data = Substance.Null)
#> prop_TRUE
#>     0.956
```


Density
0.0
0.1
0.2
0.3
0.4
0
2
4
6
8
10
12
X.squared

Appendix A

Vectorization, *apply and for loops

This section will cover the basics of vectorizations, the `*apply` family of functions and `for` loops.

A.1 Vectorization

Almost everything in `R` is a vector. A scalar is really a vector of length 1 and a `data.frame` is a collection of vectors. An nice feature of `R` is its vectorized capabilities. Vectorization indicates that a function operates on a whole vector of values at the same time and not just on a single value[1]. If you have have ever taken a basic linear algebra course, this concept will be familiar to you.
Take for example two vectors:

$$\begin{bmatrix}1\\2\\3\end{bmatrix} + \begin{bmatrix}1\\2\\3\end{bmatrix} = \begin{bmatrix}2\\4\\6\end{bmatrix}$$

The corresponding `R` code is given by:

```
a <- c(1,2,3)
b <- c(1,2,3)
a+b
#> [1] 2 4 6
```

Many of the `base` functions in `R` are already vectorized. Here are some common examples:

```
# generate a sequence of numbers from 1 to 10
(a <- 1:10)
#>  [1]  1  2  3  4  5  6  7  8  9 10

# sum the numbers from 1 to 10
sum(a)
#> [1] 55
```

[1] http://www.dummies.com/how-to/content/how-to-vectorize-your-functions-in-r.html

```
# calculate sums of each column
colSums(iris[, -5])
#> Sepal.Length  Sepal.Width Petal.Length  Petal.Width
#>        876.5        458.6        563.7        179.9
```

Exercise: What happens when you sum two vectors of different lengths?

A.2 Family of `*apply` functions

- `apply`, `lapply` and `sapply` are some of the most commonly used class of functions in `R`
- `*apply` functions are not necessarily faster than loops, but can be easier to read (and vice cersa)
- `apply` is used when you need to perform an operation on every row or column of a matrix or data.frame
- `lapply` and `sapply` differ in the format of the output. The former returns a list while the ladder returns a vector
- There are other `*apply` functions such as `tapply`, `vapply` and `mapply` with similar functionality and purpose

A.2.1 Loops vs. Apply

```
# Getting the row means of two columns
# Generate data
N <- 10000
x1 <- runif(N)
x2 <- runif(N)
d <- as.data.frame(cbind(x1, x2))
head(d)
#>        x1     x2
#> 1 0.57632 0.9615
#> 2 0.56474 0.1950
#> 3 0.07399 0.3001
#> 4 0.45387 0.3823
#> 5 0.37328 0.1197
#> 6 0.33132 0.9891

# Loop:
# create a vector to store the results in
rowMeanFor <- vector("double", N)

for (i in seq_len(N)) {
        rowMeanFor[[i]] <- mean(c(d[i, 1], d[i, 2]))
}

# Apply:
rowMeanApply <- apply(d, 1, mean)

# are the results equal
all.equal(rowMeanFor,rowMeanApply)
#> [1] TRUE
```

A.2.2 Descriptive Statistics using *apply

```
data(women)
# data structure
str(women)
#> 'data.frame':    15 obs. of  2 variables:
#>  $ height: num  58 59 60 61 62 63 64 65 66 67 ...
#>  $ weight: num  115 117 120 123 126 129 132 135 139 142 ...

# calculate the mean for each column
apply(women, 2, mean)
#> height weight
#>   65.0  136.7

# apply 'fivenum' function to each column
vapply(women, fivenum, c("Min." = 0, "1st Qu." = 0, "Median" = 0,
                         "3rd Qu." = 0, "Max." = 0))
#>         height weight
#> Min.      58.0  115.0
#> 1st Qu.   61.5  124.5
#> Median    65.0  135.0
#> 3rd Qu.   68.5  148.0
#> Max.      72.0  164.0
```

A.2.3 Creating new columns using sapply

You can apply a *user defined function* to columns or the entire data frame:

```
# the ouput of sapply is a vector
# the 's' in sapply stands for 'simplified' apply
mtcars$gear2 <- sapply(mtcars$gear,
                       function(i) if (i==4) "alot" else "some")

head(mtcars)[,c("gear","gear2")]
#>                   gear gear2
#> Mazda RX4            4  alot
#> Mazda RX4 Wag        4  alot
#> Datsun 710           4  alot
#> Hornet 4 Drive       3  some
#> Hornet Sportabout    3  some
#> Valiant              3  some
```

A.2.4 Applying functions to subsets using tapply

```
# Fisher's famous dataset
data(iris)
```

```
str(iris)
#> 'data.frame':    150 obs. of  5 variables:
#>  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#>  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#>  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#>  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#>  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

# mean sepal length by species
tapply(iris$Sepal.Length, iris$Species, mean)
#>     setosa versicolor  virginica
#>      5.006      5.936      6.588
```

A.2.5 Nested for loops using `mapply`

`mapply` is my favorite `base R` function and here are some reasons why:

- Using `mapply` is equivalent to writing nested `for` loops except that it is 100% more human readable and less prone to errors
- It is an effective way of conducting simulations because it iterates of many arguments

Let's say you want to generate random samples from a normal distribution with varying means and standard deviations. Of course the brute force way would be to write out the command once, copy paste as many times as you want, and then manually change the arguments for `mean` and `sd` in the `rnorm` function as so:

```
v1 <- rnorm(100, mean = 5, sd = 1)
v2 <- rnorm(100, mean = 10, sd = 5)
v3 <- rnorm(100, mean = -3, sd = 10)
```

This isn't too bad for three vectors. But what if you want to generate many more combinations of means and sds ? Furthermore, how can you keep track of the parameters you used? Now lets consider the `mapply` function:

```
means <- c(5,10,-3) ; sds <- c(1,5,10)

# MoreArgs is a list of arguments that dont change
randomNormals <- mapply(rnorm, mean = means, sd = sds,
                        MoreArgs = list(n = 100))

head(randomNormals)
#>       [,1]   [,2]    [,3]
#> [1,] 3.836  8.771   5.144
#> [2,] 4.525  9.376  -2.280
#> [3,] 4.072 13.144   2.940
#> [4,] 4.737 18.210 -13.118
#> [5,] 5.690 22.951  -7.008
#> [6,] 4.826  7.615 -15.323
```

The following diagram (from r4ds) describes exactly what is going on in the above function call to `mapply`:

Advantages:

1. Result is automatically stored in a matrix

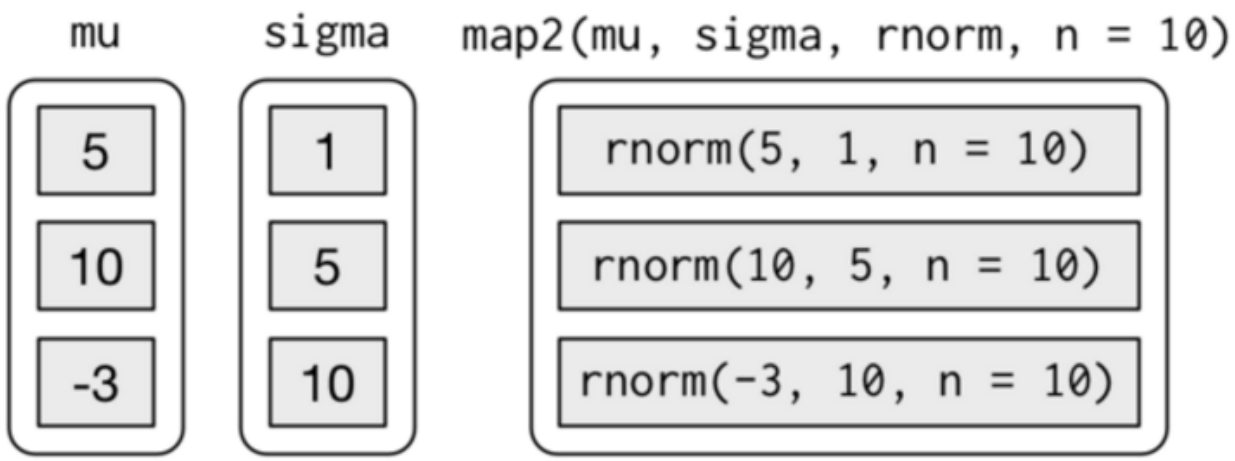


Figure A.1

2. The parameters are also saved in `R` objects so that they can be easily manipulated and/or recovered

Consider a more complex scenario where you want to consider many possible combinations of means and sds. We take advantage of the `expand.grid` function to create a `data.frame` of simulation parameters:

```
simParams <- expand.grid(means = 1:10,
                         sds = 1:10)

randomNormals <- mapply(rnorm, mean = simParams$means,
                        sd = simParams$sds,
                        MoreArgs = list(n = 100))

dim(randomNormals)
#> [1] 100 100
```

A.3 Creating dynamic documents with `mapply`

`mapply` together with the `rmarkdown` package (Allaire et al., 2018) can be very useful to create dynamic documents for exploratory analysis. We illustrate this using the Motor Trend Car Road Tests data which comes pre-loaded in `R`.

> The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Copy the code below in a file called `mapplyRmarkdown.Rmd` :

Copy the code below in a file called `boxplotTemplate` :

Appendix B

Appendix B

Bibliography

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., and Chang, W. (2018). *rmarkdown: Dynamic Documents for R*. R package version 1.10.

Bhatnagar, S., Lasry, N., Desmarais, M., and Charles, E. (2016). Dalite: Asynchronous peer instruction for moocs. In *European Conference on Technology Enhanced Learning*, pages 505–508. Springer.

Pruim, R., Horton, N. J., and Kaplan, D. T. (2015). Start teaching with r.