MATH 697

Sahir Rai Bhatnagar

2017-08-17

Contents

Syllabus

General Information

- Instructor: Sahir Bhatnagar
- Email: sahir.bhatnagar@mail.mcgill.ca
- Website: http://sahirbhatnagar.com/MATH697/
- Lectures: Tuesdays 9am - 12pm
- Office: TBD
- Office Hours: By appointment only
- Prerequisite(s): Calculus and Algebra
- Texts: *Modern Mathematical Statistics with Applications*, 2nd Edition by Jay L. Devore and Kenneth N. Berk

Course Description

The main learning outcomes of this course are to get a broad idea about some frequently used probability models and to learn basic results and techniques in probability theory and statistical inference. Most of the materials for the course will be drawn from the first seven chapters of the textbook. The book does not, however, contain all the materials we intend to cover in this course. Some extra notes will therefore be given on those topics not in the text book. We will also introduce computational methods in statistics with the statistical software program `R`.

Grade Distribution

Assignments	10%
Quizzes	40%
Final Exam	50%

Target Syllabus

Overview and Descriptive Statistics (Weeks 1-4)

- 1.1 Populations and Samples
- 1.2 Pictorial and Tabular Methods in Descriptive Statistics
- 1.3 Measures of Location
- 1.4 Measures of Variability

Probability (Weeks 1-4)

- 2.1 Sample Spaces and Events
- 2.2 Axioms, Interpretations, and Properties of Probability
- 2.3 Counting Techniques
- 2.4 Conditional Probability
- 2.5 Independence

Discrete Random Variables and Probability Distributions (Weeks 1-4)

- 3.1 Random Variables
- 3.2 Probability Distributions for Discrete Random Variables
- 3.3 Expected Values of Discrete Random Variables
- 3.4 Moments and Moment Generating Functions
- 3.5 The Binomial Probability Distribution
- 3.7 The Poisson Probability Distribution

Continuous Random Variables and Probability Distributions (Weeks 5-8)

- 4.1 Probability Density Functions and Cumulative Distribution Functions
- 4.2 Expected Values and Moment Generating Functions
- 4.3 The Normal Distribution
- 4.7 Transformations of a Random Variable

Joint Probability Distributions (Weeks 5-8)

- 5.1 Jointly Distributed Random Variables
- 5.2 Expected Values, Covariance, and Correlation
- 5.3 Conditional Distributions
- 5.4 Transformations of Random Variables

Statistics and Sampling Distributions (Weeks 5-8)

- 6.1 Statistics and Their Distributions
- 6.2 The Distribution of the Sample Mean
- 6.3 The Mean, Variance, and MGF for Several Variables
- 6.4 Distributions Based on a Normal Random Sample

Point Estimation (Weeks 9-12)

- 7.1 General Concepts and Criteria
- 7.2 Methods of Point Estimation

Statistical Intervals Based on a Single Sample (Weeks 9-12)

- 8.1 Basic Properties of Confidence Intervals
- 8.2 Large-Sample Confidence Intervals for a Population Mean and Proportion
- 8.3 Intervals Based on a Normal Population Distribution
- 8.4 Confidence Intervals for the Variance and Standard Deviation of a Normal Population

- 8.5 Bootstrap Confidence Intervals

Tests of Hypotheses Based on a Single Sample (Weeks 9-12)

- 9.1 Hypotheses and Test Procedures
- 9.2 Tests About a Population Mean
- 9.3 Tests Concerning a Population Proportion
- 9.4 P-Values
- 9.5 Some Comments on Selecting a Test Procedure

Prerequisites

Install R and RStudio

All examples in this book are run in an R environment. You also need a recent version of RStudio, which is a software application that facilitates how you interact with `R`. It is developed by data enthusiasts who consider statistics to be more than just simulations, formulas and proofs. RStudio emphasizes the following:

1. **Version control**: Why I should use version control especially for the solo data analyst.
2. **Reproducible research**: seamless integration with RMarkdown for creating dynamic documents and presentations
3. **Creating R Packages**: seamless integration with the devtools package for creating software that implements your statistical method or analysis.

R Packages

The following packages will be called upon at some point, so please install them before getting started with the tutorials. Enter the following command in `R`:

```
install.packages(c("pacman", "knitr", "data.table", "rmarkdown", "tidyverse",
    "boot", "Hmisc"))
```

Introduction to R

Try out the interactive tutorial: `http://swirlstats.com/`

Background Reading

The greatest thing about `R` is that there are so many people out there willing to help you. `R` users are constantly writing tutorials and creating packages to make your analysis tasks easier. Here is a very targeted list that I suggest reading prior to starting the tutorials

1. Writing Functions
2. `for` loops
3. `apply` vs. `for`

Part I

Part I

Chapter 1

Overview and Descriptive Statistics

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 1. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 3.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, 0.1, 0.1))
plot(pressure, type = "b", pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 1.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 1.1.

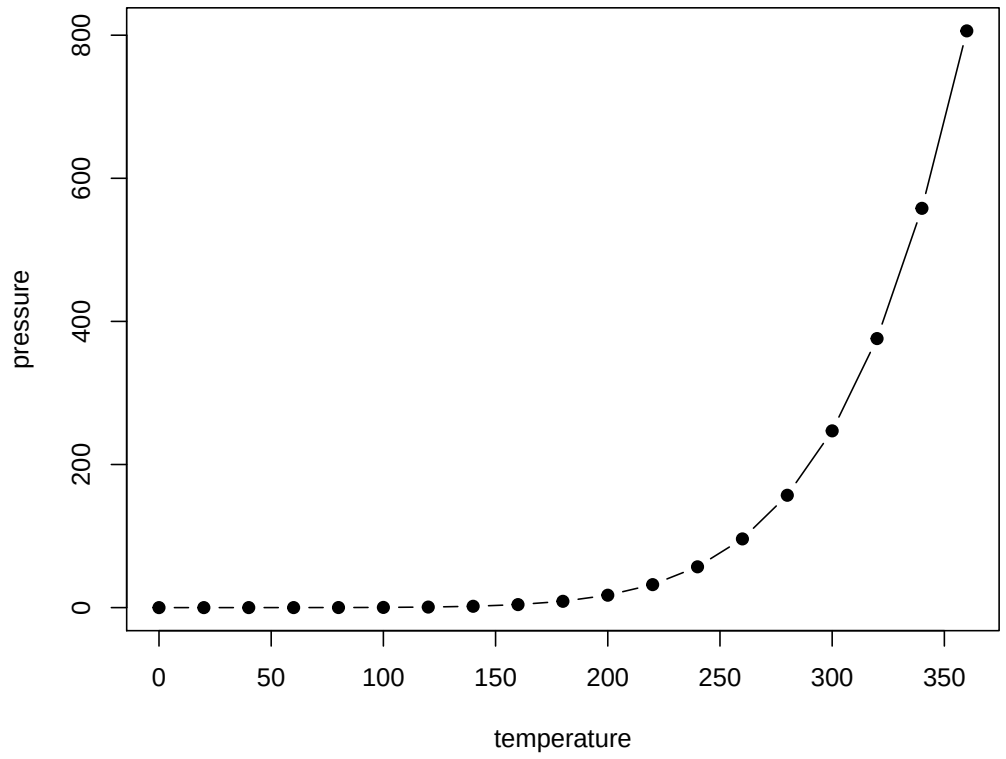

Figure 1.1: Here is a nice figure!

Table 1.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2017) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see http://www.gnu.org/licenses/.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{1.1}$$

See (1.1)

Proof. The characteristic function of $X \sim \text{Pois}(\lambda)$ is $\varphi_X(t) = e^{\lambda(e^{it}-1)}$. Let $P_n = \sum_{i=1}^{n} X_i$. We know from Theorem **??** that

$$\begin{aligned}\varphi_{P_n}(t) &= \prod_{i=1}^{n} \varphi_{X_i}(t) \\ &= \prod_{i=1}^{n} e^{\lambda_i(e^{it}-1)} \\ &= e^{\sum_{i=1}^{n} \lambda_i(e^{it}-1)}\end{aligned}$$

This is the characteristic function of a Poisson random variable with the parameter $\lambda = \sum_{i=1}^{n} \lambda_i$. From Lemma **??**, we know the distribution of P_n is Pois($\sum_{i=1}^{n} \lambda_i$). □

Theorem 1.1 (Pythagorean theorem)**.** *For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have*

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

Theorem 1.1

Chapter 2

Literature

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see http://www.gnu.org/licenses/.

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see http://www.gnu.org/licenses/.

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see http://www.gnu.org/licenses/.

R is free software and comes with ABSOLUTELY NO WARRANTY. You are welcome to redistribute it under the terms of the GNU General Public License versions 2 or 3. For more information about these matters see http://www.gnu.org/licenses/.

Part II

Part II

Chapter 3

Methods

Example 3.1 (Example theorem)**.** For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the other two sides, we have

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

Chapter 4

Applications

Some *significant* applications are demonstrated in this chapter.

4.1 Example one

4.2 Example two

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

```r
# 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

```r
# 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 0.1356226 0.21527199
## 2 0.2716021 0.26700884
## 3 0.4793413 0.53208314
## 4 0.7397965 0.05376706
## 5 0.3435368 0.53268188
## 6 0.4487223 0.01851926
```

```r
# 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
##  65.0000 136.7333
```

```
# 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 ...
```

```
# 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,] 5.851127 16.981559 -1.995026
## [2,] 3.613288 15.208544  7.608420
## [3,] 3.705362  6.663188  7.639518
## [4,] 6.304945 11.361456 -7.565889
## [5,] 5.267800 10.431325  3.637958
## [6,] 5.588270  7.622221  5.763472
```

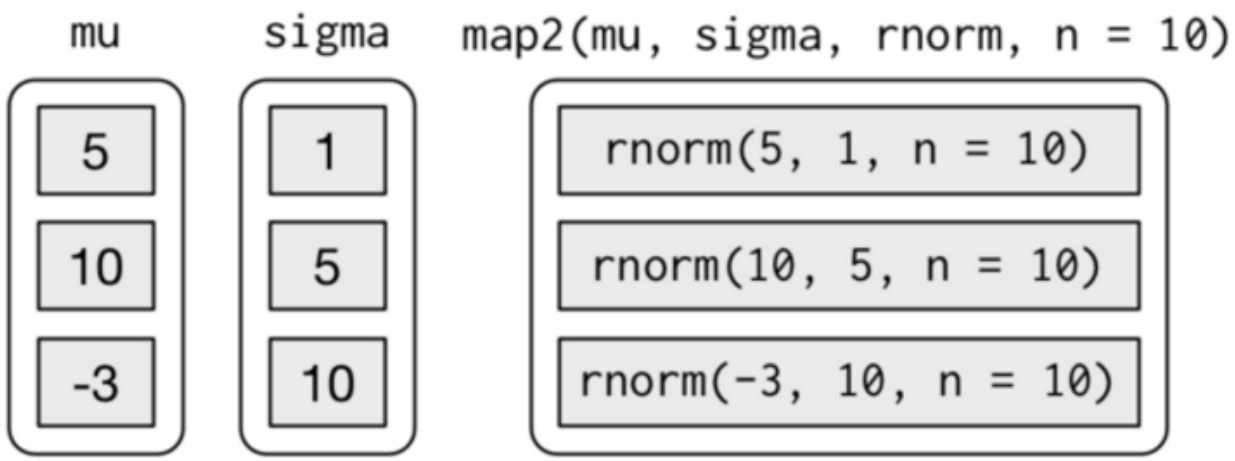

Figure A.1

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
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., 2017) 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., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A., Hyndman, R., and Arslan, R. (2017). *rmarkdown: Dynamic Documents for R*. R package version 1.6.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2017). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.4.