

STAT 8670 - Computational Methods in Statistics

Chi-Kuang Yeh

2025-07-06

Table of contents

Preface	4
Description	4
Prerequisites	4
Instructor	4
Office Hour	4
Assignment	4
Midterm	5
Topics and Corresponding Lectures	5
Recommended Textbooks	5
Side Readings	5
1 Data Structure and R Programming	6
1.1 Data type	6
1.1.1 To change data type	7
1.2 Operators	7
1.2.1 Comparison Operator	7
1.2.2 Logical Operator	8
1.3 Indexing	8
1.4 Naming	8
1.5 Array and Matrix	8
1.6 Key and Value Pair	8
1.7 Data Frame	8
1.8 Tidyverse	9
2 Optimization	10
2.1 Speed comparison	10
2.2 Type of Optimization Algorithms	11
3 Resampling, Jackknife and Bootstrap	12
3.1 Introduction	12
3.2 Jackknife	12
3.3 Bootstrap	12
3.4 Applications	12
References	13

I	Appendix	14
4	Appendix: Introduction to R?	15
4.1	R	15
4.2	IDE	15
4.2.1	Rstudio	15
4.2.2	Visual Studio Code (VS Code)	15
4.2.3	Positron	16
4.3	RStudio Layout	16
4.4	R Scripts	16
4.5	R Help	16
4.6	R Packages	16
4.7	R Markdown	17
4.8	Vectors	17
4.9	Data Sets	17

Preface

Description

Topics included are optimization, numerical integration, bootstrapping, cross-validation and Jackknife, density estimation, smoothing, and use of the statistical computer package of S-plus/R.

Prerequisites

MATH 4752/6752 – Mathematical Statistics II, and the ability to program in a high-level language.

Instructor

[Chi-Kuang Yeh](#), I am a postdoctoral scholar at the Department of Statistics and Actuarial Science, McGill University.

- Office: 1216 Burnside Hall.
- Email: chi-kuang.yeh@mail.mcgill.ca.

Office Hour

[By appointment and a online link will be provided later]

Assignment

- Assignment 1: Date and topics TBA

Midterm

- Midterm 1: Date and topics TBA

Topics and Corresponding Lectures

Those chapters are based on the lecture notes. This part will be updated frequently.

Topic	Lecture Covered
Optimization	TBA
Numerical integration	TBA
Jackknife	TBA
Bootstrap	TBA
Cross-validation	TBA
Smoothing	TBA
Density estimation	TBA
Monte Carlo Methods	TBA

Recommended Textbooks

- Givens, G.H. and Hoeting, J.A. (2012). *Computational Statistics*. Wiley, New York.
- Rizzo, M.L. (2007) *Statistical Computing with R*. CRC Press, Boca Raton.
- Hothorn, T. and Everitt, B.S. (2006). *A Handbook of Statistical Analyses Using R*. CRC Press, Boca Raton.

Side Readings

- Wickham, H., Çetinkaya-Rundel, M. and Grolemund, G. (2023). *R for Data Science*. O'Reilly.

1 Data Structure and R Programming

Data types, operators, variables

Two basic types of objects: (1) data & (2) functions

- Data: can be a number, a vector, a matrix, a dataframe, a list or other datatypes
- Function: a function is a set of instructions that takes input, processes it, and returns output. Functions can be built-in or user-defined.

1.1 Data type

- Boolean/Logical: Yes or No, Head or Tail, True or False
- Integers: Whole numbers \mathbb{Z} , e.g., 1, 2, 3, -1, -2, -3
- Characters: Text strings, e.g., “Hello”, “World.”
- Floats: Noninteger fractional numbers, e.g., π , e .
- Missing data: NA in R, which stands for “Not Available.” It is used to represent missing or undefined values in a dataset.

```
day <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
weather <- c("Raining", "Sunny", NA, "Windy", "Snowing")
data.frame(rbind(day, weather))
```

	X1	X2	X3	X4	X5
day	Monday	Tuesday	Wednesday	Thursday	Friday
weather	Raining	Sunny	<NA>	Windy	Snowing

- Other more complex types

1.1.1 To change data type

You may change the data type using the following functions, but the chance is that some of the information will be missing. Do this with caution!

```
x <- pi  
print(x)
```

```
[1] 3.141593
```

```
x_int <- as.integer(x)  
print(x_int)
```

```
[1] 3
```

Some of the conversion functions:

- `as.integer()`: Convert to integer.
- `as.numeric()`: Convert to numeric (float).
- `as.character()`: Convert to character.
- `as.logical()`: Convert to logical (boolean).
- `as.Date()`: Convert to date.
- `as.factor()`: Convert to factor (categorical variable).
- `as.list()`: Convert to list.
- `as.matrix()`: Convert to matrix.
- `as.data.frame()`: Convert to data frame.
- `as.vector()`: Convert to vector.
- `as.complex()`: Convert to complex number.

1.2 Operators

- Unary: With only **one** argument. E.g., `-x` (negation), `!x` (logical negation).
- Binary: With **two** arguments. E.g., `x + y` (addition), `x - y` (subtraction), `x * y` (multiplication), `x / y` (division).

1.2.1 Comparison Operator

Comparing two objects. E.g., `x == y` (equal), `x != y` (not equal), `x < y` (less than), `x > y` (greater than), `x <= y` (less than or equal to), `x >= y` (greater than or equal to).

1.2.2 Logical Operator

Logical operators are used to combine or manipulate logical values (TRUE or FALSE). E.g., `x & y` (logical AND), `x | y` (logical OR), `!x` (logical NOT).

We shall note that the logical operators in R are vectorized, `x | y` and `x || y` are different. The former is vectorized, while the latter is not.

```
x <- c(TRUE, FALSE, FALSE)
y <- c(TRUE, FALSE, FALSE)
x | y # [1] TRUE FALSE FALSE
x || y # This will return an error
```

1.3 Indexing

Indexing is a way to access or modify specific elements in a data structure. In **R**, indexing can be done using square brackets `[]` for vectors and matrices, or the `$` operator for data frames. Note that the index starts from **0** in **R**, which is different from some other programming languages like Python.

1.4 Naming

1.5 Array and Matrix

1.6 Key and Value Pair

1.7 Data Frame

Dataframe is a two-dimensional, tabular data structure in R that can hold different types of variables (numeric, character, factor, etc.) in each column. It is similar to a spreadsheet or SQL table.

```
iris <- datasets::iris
head(iris)
```


	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

1.8 Tidyverse

The tidyverse is a collection of open source packages for the R programming language introduced by Hadley Wickham and his team that “share an underlying design philosophy, grammar, and data structures” of tidy data. Characteristic features of tidyverse packages include extensive use of non-standard evaluation and encouraging piping.

```
## Load all tidyverse packages
library(tidyverse)

## Or load specific packages in the tidy family
library(dplyr) # Data manipulation
library(ggplot2) # Data visualization
library(readr) # Data import
library(tibble) # Tidy data frames
library(tidyr) # Data tidying
# ...
```

Some of the materials are adapted from [CMU Stat36-350](#).

2 Optimization

The optimization plays an important role in statistical computing, especially in the context of maximum likelihood estimation (MLE) and other statistical inference methods. This chapter will cover various optimization techniques used in statistical computing.

For instance, for a linear regression

$$y = X\beta + \varepsilon.$$

From regression class, we know that the (ordinary) least-squares estimation (OLE) for β is given by $\hat{\beta} = (X^\top X)^{-1} X^\top y$. It is convenient as the solution is in the **closed-form**! However, in the most case, the closed-form solutions will not be available.

For GLMs or non-linear regression, we need to do this **iteratively**!

2.1 Speed comparison

```
set.seed(2017-07-13)
X <- matrix(rnorm(5000 * 100), 5000, 100)
y <- rnorm(5000)
library(microbenchmark)
microbenchmark(solve(t(X) %*% X) %*% t(X) %*% y)
```

Warning in microbenchmark(solve(t(X) %*% X) %*% t(X) %*% y): less accurate
nanosecond times to avoid potential integer overflows
Unit: milliseconds

	expr	min	lq	mean	median	uq
	solve(t(X) %*% X) %*% t(X) %*% y	28.57097	29.23575	30.04752	29.87713	30.62448
	max neval					
	32.21641	100				

```
microbenchmark(solve(t(X) %*% X) %*% t(X) %*% y,  
               solve(crossprod(X), crossprod(X, y)))
```

Unit: milliseconds

	expr	min	lq	mean	median
solve(t(X) %*% X) %*% t(X) %*% y		28.39886	29.03231	29.73657	29.41512
solve(crossprod(X), crossprod(X, y))		24.96937	25.01504	25.25663	25.05457
uq	max	neval			
30.23379	32.94231	100			
25.16457	28.50168	100			

2.2 Type of Optimization Algorithms

There are in general two types of the optimization algorithms: (1). **deterministic** and (2). **metaheuristic**. Deterministic and metaheuristic algorithms represent two distinct paradigms in optimization. Deterministic methods, such as gradient descent, produce the same solution for a given input and follow a predictable path toward an optimum. In contrast, metaheuristic approaches—like genetic algorithms—incorporate randomness and do not guarantee the best possible solution. However, they are often more effective at avoiding local optima and exploring complex search spaces.

Examples are borrowed from the following sources:

- Peng, R.D. [Advanced Statistical Computing](#).

3 Resampling, Jackknife and Bootstrap

3.1 Introduction

This chapter covers resampling methods including the jackknife and bootstrap techniques.

3.2 Jackknife

The jackknife is a resampling technique used to estimate the bias and variance of a statistic.

3.3 Bootstrap

The bootstrap is a resampling method that allows estimation of the sampling distribution of almost any statistic using random sampling methods.

3.4 Applications

These methods are widely used in statistical inference and have applications in various fields.

References

Part I

Appendix

4 Appendix: Introduction to R?

4.1 R

For conducting analyses with data sets of hundreds to thousands of observations, calculating by hand is not feasible and you will need a statistical software. **R** is one of those. **R** can also be thought of as a high-level programming language. In fact, **R** is [one of the top languages](#) to be used by data analysts and data scientists. There are a lot of analysis packages in **R** that are currently developed and maintained by researchers around the world to deal with different data problems. Most importantly, **R** is free! In this section, we will learn how to use **R** to conduct basic statistical analyses.

4.2 IDE

4.2.1 Rstudio

RStudio is an integrated development environment (IDE) designed specifically for working with the **R** programming language. It provides a user-friendly interface that includes a source editor, console, environment pane, and tools for plotting, debugging, version control, and package management. RStudio supports both R and Python and is widely used for data analysis, statistical modeling, and reproducible research. It also integrates seamlessly with tools like R Markdown, Shiny, and Quarto, making it popular among data scientists, statisticians, and educators.

4.2.2 Visual Studio Code (VS Code)

VS Code is a versatile code editor that supports multiple programming languages, including R. With the R extension for VS Code, users can write and execute R code, access R's console, and utilize features like syntax highlighting, code completion, and debugging. While not as specialized as RStudio for R development, VS Code offers a lightweight alternative with extensive customization options and support for various programming tasks.

4.2.3 Positron

Positron IDE is the next-generation integrated development environment developed by Posit, the company behind RStudio. Designed to be a modern, extensible, and language-agnostic IDE, Positron builds on the strengths of RStudio while supporting a broader range of languages and workflows, including R, Python, and Quarto.

4.3 RStudio Layout

RStudio consists of several panes: - **Source**: Where you write scripts and markdown documents. - **Console**: Where you type and execute R commands. - **Environment/History**: Shows your variables and command history. - **Files/Plots/Packages/Help/Viewer**: For file management, viewing plots, managing packages, accessing help, and viewing web content.

4.4 R Scripts

R scripts are plain text files containing R code. You can create a new script in RStudio by clicking **File > New File > R Script**.

4.5 R Help

Use `?function_name` or `help(function_name)` to access help for any R function. For example:

```
?mean  
help(mean)
```

4.6 R Packages

Packages extend R's functionality. Install a package with:

```
install.packages("package_name")
```

Load a package with:


```
library(package_name)
```

4.7 R Markdown

R Markdown allows you to combine text, code, and output in a single document. Create a new R Markdown file in RStudio via **File > New File > R Markdown...**

Recently, the posit team has developed a new version of the R Markdown called quarto document, with the file extension `.qmd`. It is still under rapid development.

4.8 Vectors

Vectors are the most basic data structure in R.

```
x <- c(1, 2, 3, 4, 5)
x
```

```
[1] 1 2 3 4 5
```

You can perform operations on vectors:

```
x * 2
```

```
[1] 2 4 6 8 10
```

4.9 Data Sets

Data frames are used for storing data tables. Create a data frame:

```
df <- data.frame(Name = c("Alice", "Bob"), Score = c(90, 85))
df
```

```
  Name Score
1 Alice   90
2  Bob   85
```

You can import data from files using `read.csv()` or `read.table()`.

This appendix is adapted from [Why R?](#).