Statistical Theory II
Practical Labs

Lab 1: Introduction

Paulius Kazlauskas

Essential Things First

Plan for today (03–03)

Introduction to the practical sessions

Introduction to R and its basic functions

If time permits, working on homework exercises

About the practical part

Bayesian statistics with a focus on R programming

Purpose of lab sessions:

- Implement Bayesian methods used in empirical work
- Learn programming in R
- Conduct analyses in R

Plan for practical labs – 6 labs in total

03–03 — Introduction to RStudio interface and R programming: variables, vectors, matrices, functions, conditional statements, loops.

03–10 — Bayesian inference for discrete random variables

03–26 (**Thursday**) — Continuous random variables

04–31 — Bayesian inference for Binomial proportion / Poisson

04–14 — Linear regression

05–05 — Assessment of the practical part

Plan for each lab

Homework solutions: first by hand and then in R.

Solutions uploaded after each lab.

Assessment of the practical part

In-person assessment (exam) worth **30%** of the final grade

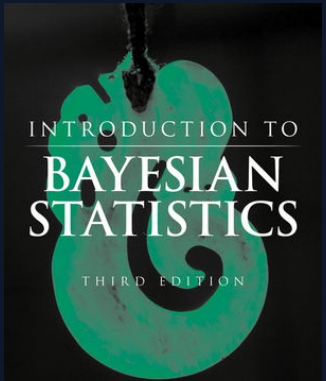
Conducted on the **5th of May** in **709**

Assessment information:

- Each task worth a number of points, total = 10
- Tasks cover course contents, similar to homework
- Conducted in R

Literature for the practical part

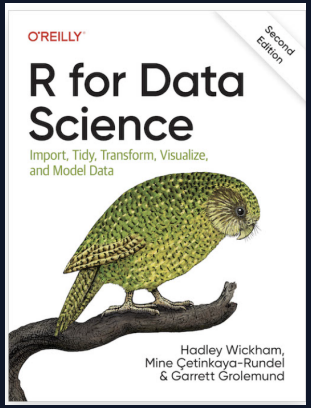
William M. Bolstad, James M. Curran
Introduction to Bayesian Statistics (3rd ed.)
Print ISBN: 9781118091562 Online ISBN: 9781118593165
DOI: 10.1002/9781118593165



Literature for the practical part

For mastering R skills

Great Interactive (FREE) online version **https://r4ds.hadley.nz/**



Software installation

Download and install R: cran.r-project.org

Download and install RStudio: posit.co/download/rstudio-desktop

To learn more about R

Introduction to R (manual): R-intro.pdf

Good coding style: style.tidyverse.org

It's open source! Just type *problem* in R

Or ask your favourite LLM

Motivation

Motivation for programming (economics angle)

Programming helps you **understand statistical theory**: writing algorithms makes assumptions explicit and removes ambiguity.

Modern economics is computational:

- empirical micro / macro, causal inference, time series, structural estimation
- simulation, optimization, Bayesian methods, reproducible research

In practice, **economists program**: you will read code, write code, and debug code in courses and in jobs.

Command-line and scripting feels harder at first than point-and-click tools, but gives:

- reproducibility, flexibility, automation, scalability
- ability to build *new* models/estimators rather than only using existing menus

Why R?

R is widely used by statisticians and economists

Strong for computation + graphics; rich ecosystem of packages

Results can be displayed, saved, exported, reused in workflows

Free and cross-platform (Windows / macOS / Linux)

Large community support (e.g., R-bloggers)

AI Policy

Yes, AI can code everything for you faster, and better (at least compared to your knowledge of R now)

AI still often fails at high level coding

Learn the basics at least! So you can proof-check, and don't rely on AI.

Use these labs to code!

Contacts

paulius.kazlauskas@evaf.vu.lt
VU TechHub (2nd floor)
Teams, Instagram or wherever you can find me

Bonus points

You can earn up to **4%** of the **final grade** for **free!**

How?

Actively participate.

Ask questions, answer questions, present your opinion or solutions, solve problems in front of the class, **whatever!**

Can everybody get bonus points?

YES!

The R part

Basic commands: using R as a calculator

R evaluates expressions entered in the console.

`+` Addition `-` Subtraction `*` Multiplication

`/` Division `^` Exponentiation

Operator precedence: `4 + 5 * 3` returns 19

```
> # Calculations work based on mathematical ordering:
> 4 + 5 * 3
> (4 + 5) * 3
> 2^5
```

R works with objects

R works with **objects** (variables).

Each object has:

- **mode** (numeric, character, complex, logical (TRUE or FALSE))
- **length** (number of elements)

```
> x <- 1
> mode(x)
> length(x)
> y <- c(1, 2, 3)
> mode(y)
> length(y)
```

x is a numeric object with length 1 while y is a numeric object (vector) with length 3.

Basic commands

> Create a character object and inspect it:

```
A <- "QE"; A; mode(A); length(A)
```

Output:

```
[1] "QE"
[1] "character"
[1] 1
```

> Create a logical object and inspect it:

```
compar <- TRUE; compar; mode(compar); length(compar)
```

```
[1] TRUE
[1] "logical"
[1] 1
```

Basic commands

> Use the help command to search for information about a function:

```
help(sum)
```

or

```
?sum
```

> List objects currently in memory:

```
ls()
```

> Remove objects from memory:

```
rm(x)
rm(A, y)
rm(list = ls())
```

Processing of vectors

Build a numeric vector a of dimension 5:

```
> a <- c(2,4,6,8,10)
> a
> Output: [1] 2 4 6 8 10
```

Show the first element:

```
> a[1]
> Output: [1] 2
```

Extract elements 3 to 5:

```
> b <- a[3:5]
> b
> Output: [1] 6 8 10
```

Processing of vectors

Natural logarithm `log(x)` (or `ln(x)`):

> `x <- c(2,4,6,8,10)`
> `log(x)`
> Output: `[1] 0.6931472 1.3862944 1.7917595 2.0794415 2.3025851`

Power:

> `x^2`
> Output: `[1] 4 16 36 64 100`

Exponential:

> `exp(x)`
> Output: `[1] 7.389056 54.598150 403.428793 2980.957987 22026.465795`

Processing of vectors: `sqrt`, `min`, `max`

```
> sqrt(x)  # square root
> Output: [1] 1.414214 2.000000 2.449490 2.828427 3.162278
> min(x); max(x)  # smallest, largest values
> Output: [1] 2
> Output: [1] 10
```

Processing of vectors

Calculations between two vectors x and y require consistent dimensions.

```
> x <- c(2,4,6,8,10)
> y <- c(1,3,5,7,9)
```

Summation:

```
> x + y
> Output: [1] 3 7 11 15 19
```

Subtraction:

```
> x - y
> Output: [1] 1 1 1 1 1
```

Processing of vectors

Product by items (element-wise):

```
> x * y
> Output: [1]  2 12 30 56 90
```

Scalar product:

```
> x %*% y
> Output: [,1] [1,] 190
> sum(x * y)  # same result as %*%
> Output: [1] 190
```

Processing of vectors

Compute summary measures of a:

```
> a <- c(2,4,6,8,10)
> sum(a)   Output: [1] 30
> mean(a)   Output: [1] 6
> length(a)   Output: [1] 5
> var(a)   Output: [1] 10
> sd(a)   Output: [1] 3.162278
> t(a)
```

Transpose output:

```
[,1] [,2] [,3] [,4] [,5][1,] 2 4 6 8 10
```

Processing of vectors: seq(), rep(), combining

Generate a sequence:

```
> x <- seq(1,5,0.5)
> x
```

Repeat values:

```
> rep(1,5)
```

Combine vectors (dimension must match):

```
> x <- 1:5
> y <- 6:10
> rbind(x,y)  # combine by rows
> cbind(x,y)  # combine by columns
```

Processing of matrices

Build a 5×4 matrix 'x1': for example, assign 20 numbers from 1 to 20 into 5 rows equally, with first row being 1, 6, 11, 16

```
> x1 <- matrix(1:20, nrow=5)
> x1
```

Output: **How does it look like?**

bonus point territory

Processing of matrices

Row-wise filling:

```
> x2 <- matrix(1:20, nrow=5, byrow=TRUE)
```

Matrix operations (consistent dimensions required):

```
> x1 + x2  # element-wise sum
> x1 - x2  # element-wise subtraction
> x1 * x2  # element-wise product
> t(x1) %*% x2  # matrix product
```

Processing of matrices

Build a 3×3 matrix:

```
> x3 <- matrix(c(5,1,0, 3,-1,2,4,0,-1), nrow=3, byrow=TRUE)
> dim(x3)  Output: [1] 3 3
> x3_inv <- solve(x3)  # inverse matrix
> x3[1,1]  Output: [1] 5
> x3[3,1] <- 500  # replace element (3,1)
> x3
```

Writing a function

General structure:

```
name <- function(arg) { expression }
```

Assume that we want to write a function: $f(x) = x^2$

```
> squarefunc <- function(y) { y^2 }
> squarefunc(4)   Output: [1] 16
```

Function with two arguments:

```
> add <- function(x,y) { return(x+y) }
> add(3,4)   Output: [1] 7
```

Writing a condition

General form:

```
if (expression1) expression2 else expression3
```

For example: the value of dd relies on the value of cc

```
> cc <- 1
> if (cc == 1) dd <- 4 else dd <- 5
> dd   Output: [1] 4
```

Block version:

```
> if (cc == 1) { dd <- 4 } else { dd <- 5 }
```

Writing a for-loop

General form:

```
for (name in expression) { expression }
```

Build a vector in which each element is the square of the order of the element:
$z0 = [1^2, 2^2, \ldots, 10^2]$

```
> z0 <- rep(0,10)
> v <- 1:10
> for (i in v) { z0[i] <- i^2 }
> z0
> Output: [1]  1  4  9 16 25 36 49 64 81 100
```

Writing a for-loop

Given $x = [1, 2, \ldots, 10]$, build $y = [1^3, 2^2, 3^3, 4^2, \ldots, 10^2]$

```
> x <- 1:10
> y <- rep(0,10)
> v1 <- seq(1,10,2)  # odd positions
> v2 <- seq(2,10,2)  # even positions
> for (i in v1) { y[i] <- i^3 }
> for (j in v2) { y[j] <- j^2 }
> y
```

Output: `[1] 1 4 27 16 125 36 343 64 729 100`

Writing a while-loop

Build a vector in which each element (when order is smaller than 5) is the cube of the order of the element, and the other elements are zero:

$$z1 = [1^3, 2^3, 3^3, 4^3, 0, 0, 0, 0, 0, 0]$$

```
> z1 <- rep(0,10)
> i <- 1
> while (i < 5) {
>   z1[i] <- i^3
>   i <- i + 1
> }
> z1
```

Output: `[1] 1 8 27 64 0 0 0 0 0 0`

Setting a working directory

Comments start with #

```
> getwd()  # show current working directory
> setwd("C:/Users/VU QE/Lab 1")
```

> Use forward slash / (not backslash \)

> Always use quotation marks around the path

Why is the working directory important?

R always reads and writes files relative to the working directory.

If you run:

```
> read.csv("data.csv")
```

R will look for `data.csv` inside the current working directory.

Similarly:

```
> write.csv(results, "output.csv")
> save(model, file="model.RData")
```

Files are saved into the working directory.

If the file is not there $\rightarrow$ R returns an error.

Creative task

Explore the dataset from `mtcars`

Understand the data

?mtcars

Look at the structure

head(mtcars) str(mtcars) summary(mtcars)

Goal:

- What kind of data is this?
- What do the variables represent?
- What patterns do you notice?

More on basics

> R people have made learning R ever so simple...
> package "swirl" allows you to learn the basics of R programming inside R console.
> install.packages("swirl")
> library(swirl)
> or if feeling fancy *pacman* :: *p_load*(*swirl*)
> When ready, type swirl()
> Follow instructions on the console

Credit

The lab slides are based on the slides of Dominykas Ragelis.