

Computing for Big Data (BST-262)

Christine Choirat

2017-10-30

Contents

1	Introduction	5
1.1	Logistics	5
1.2	Prerequisites	5
1.3	Rationale	5
1.4	Big data bottlenecks	5
1.5	Syllabus	7
1.6	Evaluation	7
1.7	Software tools and packages	8
1.8	Datasets	8
1.9	Contributing with GitHub	9
1.10	Before we start...	9
1.11	Style	9
2	Basic tools	11
2.1	Command line tools	11
2.2	Makefiles	13
2.3	Git and GitHub	17
3	Packages	19
3.1	Why?	19
3.2	Package structure	19
3.3	Building steps	19
3.4	Create an R package	20
3.5	R packages on GitHub	21
3.6	RStudio projects	30
3.7	Package workflow example	31
3.8	Unit testing	39
3.9	Continuous integration	39
3.10	Code coverage	42
3.11	Back to GitHub	44
3.12	Vignettes	44
4	Optimization	47
4.1	Measuring performance	47
4.2	Improving performance	55
4.3	Vectorization	55
4.4	Parallelization	57
4.5	Introduction to C++	58
4.6	Rcpp packages	61
5	Databases	63
5.1	Overview	63

5.2	SQL	63
5.3	noSQL	63
5.4	R interfaces	63
6	Big data	65
6.1	List of tools	65
6.2	Data that fits in memory	65
6.3	Data that doesn't fit in memory (but fits on drive)	67
6.4	Pure R solutions	67
6.5	Scaling up	67
7	Visualization	69
7.1	Principles of visualization	69
7.2	Maps and GIS	69

Chapter 1

Introduction

1.1 Logistics

- Fall 2 course
- Tuesday and Thursday, 11:30am-1pm
- Contact info: Christine Choirat (cchoirat@iq.harvard.edu). Please use BST232 in the email title.
- TA's: Qian Di (qiandi@mail.harvard.edu) and Ben Sabath (mbsabath@hsph.harvard.edu)
- Office hours:
 - Ben: Tuesday 1:30-2:30pm
 - Qian: Thursday 10:30-11:30am
 - Christine: Tuesday 10:30-11:30am (office 437A)
- Course GitHub repository <https://github.com/cchoirat/bigdata17>
- Open file in folder `_book/index.html`
- These course notes are **work in progress**.

1.2 Prerequisites

For BST262 (Computing for Big Data), we assume familiarity with the material covered in BST260 (Introduction to Data Science).

We will use R to present concepts that are mostly language-agnostic. We could have used Python, as in BST261 (Data Science II).

1.3 Rationale

1. Available data grows at a much faster rate than available computing capacity.
2. Statistical software programs such as R were not designed to handle datasets of massive size.

1.4 Big data bottlenecks

As described by Lim and Tjhi (2015), there are three bottlenecks:

- CPU
- RAM
- I/O

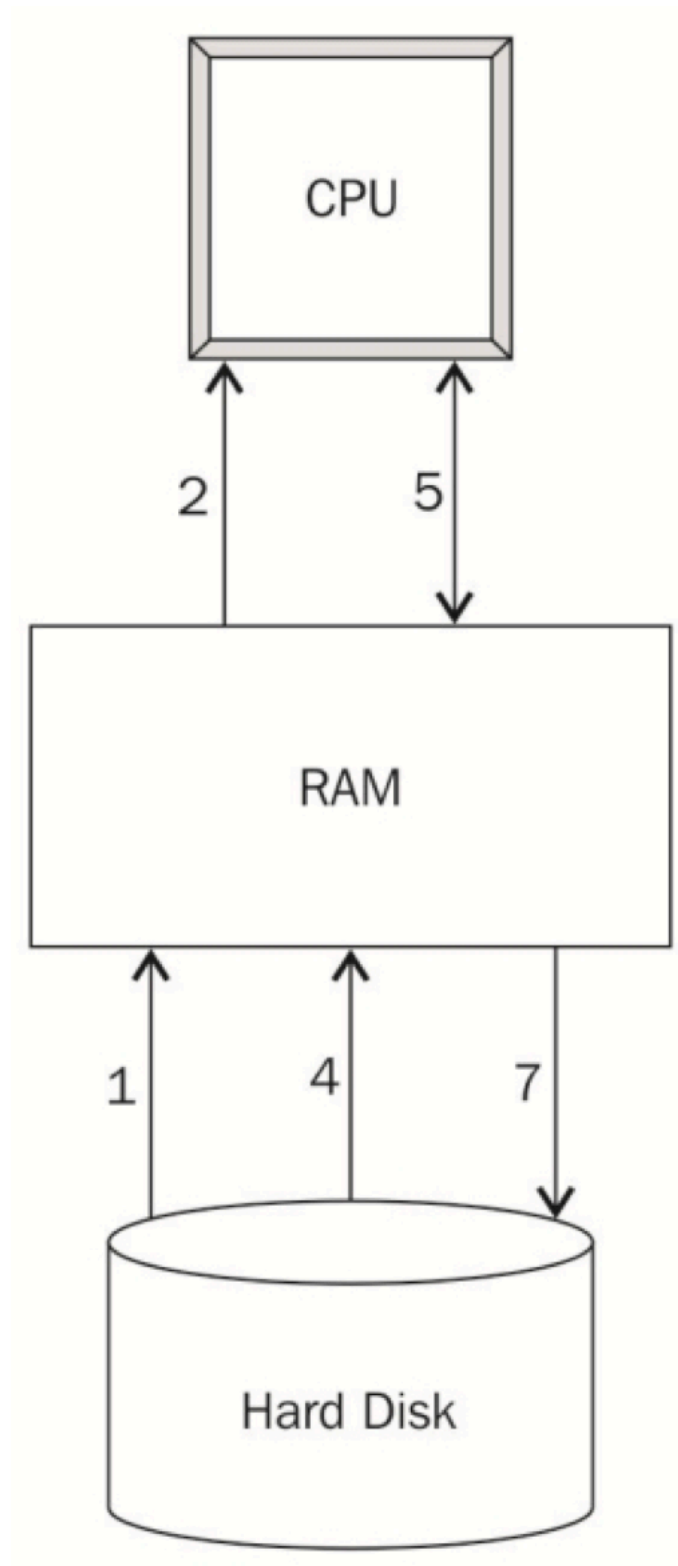


Figure 1.1: Steps to execute an R program, from @Lim2015, Chapter 1.

Exercise 1.1. Can you identify points 1–7 in the following code snippet?

```
data <- read.csv("mydata.csv")
totals <- colSums(data)
write.csv(totals, "totals.csv")
```

1.5 Syllabus

Part I – Good code still matters (*even with lots of computing resources*)

Week 1 - Basic tools

- Lecture 1. Unix scripting, make
- Lecture 2. Version control: Git and GitHub (guest lecture: Ista Zhan)

Week 2 - Creating and maintaining R packages

- Lecture 3. Rationale, package structure, available tools
- Lecture 4. Basics of software engineering: unit testing, code coverage, continuous integration

Week 3 - Software optimization

- Lecture 5. Measuring performance: profiling and benchmarking tools
- Lecture 6. Improving performance: an introduction to C/C++, Rcpp

Part II – Scaling up (*don't use big data tools for small data*)

Week 4 – Databases

- Lecture 7. Overview of SQL (SQLite, PostgreSQL) and noSQL databases (HBase, MongoDB, Cassandra, BigTable, ...)
- Lecture 8. R database interfaces (in particular through dplyr and mongolite)

Week 5 - Analyzing data that does not fit in memory

- Lecture 9. Pure R solutions (sampling, **ff** and **bigmemory**, other interpreters). JVM solutions (h2o, Spark)
- Lecture 10. An introduction to parallel computing; clusters and cloud computing. “Divide and Conquer” (MapReduce approaches)

Week 6 – Visualization

- Lecture 11. Principles of visualization (guest lecture: James Honaker)
- Lecture 12. Maps and GIS: principles of GIS, using R as a GIS, PostGIS

Weeks 7 & 8 - Guest lectures (order and precise schedule TBD)

- Software project management (Danny Brooke)
- R and Spark (Ellen Kraffmiller and Robert Treacy)
- Advanced GIS and remote sensing (TBD)
- Cluster architecture (William J. Horka)

1.6 Evaluation

Grades will be based on **two mandatory problem sets**. Each problem set will correspond to 50% (= 50 points) of the final grade. The first problem set will be available by the end of week 3 and the second problem set by the end of week 6.

You will be required to submit problem set solutions within two weeks. Grades, and feedback when appropriate, will be returned two weeks after submission.

You will submit a markdown document that combines commented code for data analysis and detailed and structured explanations of the algorithms and software tools that you used.

1.7 Software tools and packages

We will mostly use R in this course. Some examples will be run in Python.

In general, we will use free and open-source software programs such as PostgreSQL / PostGIS or Spark.

1.8 Datasets

We have collected datasets to illustrate concepts. They are hosted on a Dropbox folder.

1.8.1 MovieLens

MovieLens by Harper and Konstan (2015, <https://grouplens.org/datasets/movielens/>) collects datasets from the website <https://movielens.org/>.

There are datasets of different sizes. We will use:

1. Small (1MB): <https://grouplens.org/datasets/movielens/latest/>
2. Benchmark (~190MB zipped): <https://grouplens.org/datasets/movielens/20m/>

1.8.2 Airlines data

The airlines dataset comes from the U.S. Department of Transportation and were used in the 2009 Data Expo of the American Statistical Association (ASA).

We will use a version curated by h2o: <https://github.com/h2oai/h2o-2/wiki/Hacking-Airline-DataSet-with-H2O>.

1.8.3 Insurance claims

Claims data contain Protected Health Information (PHI). There are strong privacy restrictions to store, use and share this type of data.

We will use synthetic data (Sample 1) from the Centers for Medicare and Medicaid Services (CMS).

1.8.4 Census

Census data is commonly merged with administrative claims data such as Medicare. We will use data from the Census Bureau.

1.8.5 PM_{2.5} exposure

We will use PM_{2.5} exposure data from the EPA Air Quality System (AQS) to illustrate GIS linkage concepts.

1.8.6 Methylation

If there is enough interest, we might present methylation examples.

1.9 Contributing with GitHub

If you have suggestions, you can open a GitHub issue at <https://github.com/cchoirat/bigdata17/issues>.

If you want to contribute, we welcome pull requests.

1.10 Before we start...

How much R do you know?

Introduction to R: <http://tutorials.iq.harvard.edu/R/Rintro/Rintro.html>

Regression models in R: <http://tutorials.iq.harvard.edu/R/Rstatistics/Rstatistics.html>

R graphics: <http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html>

R programming: <http://tutorials.iq.harvard.edu/R/RProgramming/Rprogramming.html>

1.11 Style

Reading: <http://adv-r.had.co.nz/Style.html>

Chapter 2

Basic tools

In this Chapter, we present basic tools that will be important when interacting with big data systems: the command-line interface (CLI) in a Unix shell and several utilities (**less**, **awk**, **vi** and **make**).

2.1 Command line tools

We assume some familiarity with the Unix shell, for example as in <http://swcarpentry.github.io/shell-novice/>.

We also assume that you have access to a shell, either because you use Linux or OS X or because you have the right tools on Windows (for example Cygwin or the Bash shell in Windows 10).

2.1.1 Why use the command line?

- Batch processing
- Cluster and cloud computing

2.1.2 Basic Unix tools

2.1.3 Useful tools

2.1.3.1 **less**

less is a pager that lets you view one page at a time files that can be very large.

File `DE1_0_2008_to_2010_Carrier_Claims_Sample_1A.csv` in `Data17/SyntheticMedicare` is 1.2GB. Even if we have enough RAM to process the data, **less** helps get a very quick sense of the data (variable names, separators, etc.)

2.1.3.2 **awk**

awk is a text-processing programming language available on most Unix systems. It can be used for data extraction.

2.1.3.3 vi

vi is a screen-based text editor available on almost all Unix systems. Most versions are actually Vim (that stands for “Vi IMproved”).

There are many cheat sheets and tutorials available on-line (for example, the interactive <http://www.openvim.com/>). I invite you to learn basics vi commands.

2.1.4 Example

Let’s apply some of the techniques described in Blackwell and Sen (2012) on Fisher’s Iris data set saved in tab-delimited format. Of course, it is a small dataset easily processed with R:

```
iris <- read.table("~/Dropbox/Data17/iris/iris.tab")
head(iris, n = 5)
```

```
## 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
```

In a shell, we can use:

```
head -n 6 ~/Dropbox/Data17/iris/iris.tab
```

```
## "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
## "1" 5.1 3.5 1.4 0.2 "setosa"
## "2" 4.9 3 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 3.6 1.4 0.2 "setosa"
```

Suppose we only need to select two variables in our model, Sepal.Length and Species. In R, we can use:

```
iris_subset <- iris[, c("Sepal.Length", "Species")]
```

or

```
iris_subset <- iris[, c(1, 5)]
head(iris_subset)
```

```
## Sepal.Length Species
## 1          5.1 setosa
## 2          4.9 setosa
## 3          4.7 setosa
## 4          4.6 setosa
## 5          5.0 setosa
## 6          5.4 setosa
```

With the tidyverse, we can use *pipes*. The %>% operator allows for performing chained operations.

```
suppressMessages(library(dplyr))
```

```
iris %>%
  select(1, 5) %>%
  head()
```

```
## Sepal.Length Species
## 1      5.1 setosa
## 2      4.9 setosa
## 3      4.7 setosa
## 4      4.6 setosa
## 5      5.0 setosa
## 6      5.4 setosa
```

In a shell, the pipe operator to combine shell commands is `|` and we can use:

```
cut -f 1,5 ~/Dropbox/Data17/iris/iris.tab | head -n 7
```

```
## "Sepal.Length" "Species"
## "1" 0.2
## "2" 0.2
## "3" 0.2
## "4" 0.2
## "5" 0.2
## "6" 0.4
```

To keep observations with “Sepal.Length” greater than 5:

```
iris %>%
  filter(Sepal.Length > 5) %>%
  head()
```

```
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1      5.1      3.5      1.4      0.2 setosa
## 2      5.4      3.9      1.7      0.4 setosa
## 3      5.4      3.7      1.5      0.2 setosa
## 4      5.8      4.0      1.2      0.2 setosa
## 5      5.7      4.4      1.5      0.4 setosa
## 6      5.4      3.9      1.3      0.4 setosa
```

In the shell, we can use the AWK programming language. We start from row NR 2 (we could start from row 1, it contains variable names) and select rows such that the second variable (Sepal.Length) is greater than 5.

```
awk 'NR == 2 || $2 > 5' ~/Dropbox/Data17/iris/iris.tab | head
```

```
## "1" 5.1 3.5 1.4 0.2 "setosa"
## "6" 5.4 3.9 1.7 0.4 "setosa"
## "11" 5.4 3.7 1.5 0.2 "setosa"
## "15" 5.8 4 1.2 0.2 "setosa"
## "16" 5.7 4.4 1.5 0.4 "setosa"
## "17" 5.4 3.9 1.3 0.4 "setosa"
## "18" 5.1 3.5 1.4 0.3 "setosa"
## "19" 5.7 3.8 1.7 0.3 "setosa"
## "20" 5.1 3.8 1.5 0.3 "setosa"
## "21" 5.4 3.4 1.7 0.2 "setosa"
```

Exercise 2.1. The iris dataset is also saved in .csv format at ~/Dropbox/Data17/iris/iris.csv. Use AWK and tail to select the last 5 observations where Sepal.Width is larger than 3.5 and Petal.Length is smaller than 1.5.

2.2 Makefiles

make is a tool that helps put all the (interdependent) pieces of an analytic workflow together:

- data retrieving
- data cleaning
- analysis
- graphs
- reports
- ...

2.2.1 Simulate data in R

```
set.seed(123)
```

File `simulate_data.R`

```
# set.seed(123)
N <- 1000 # sample size

X1 <- rpois(n = N, lambda = 50)
X2 <- 10 + rbinom(n = N, prob = 0.8, size = 1)
Y <- 10 + 3 * X1 + -5 * X2 + 3 * rnorm(n = N)

write.csv(data.frame(Y = Y, X1 = X1, X2 = X2),
          "sample_data.csv", row.names = FALSE)

head(data.frame(Y = Y, X1 = X1, X2 = X2))
```

```
##           Y X1 X2
## 1  88.74430 46 11
## 2 125.77081 58 11
## 3  70.76396 38 10
## 4 110.32157 50 10
## 5 145.79546 62 11
## 6 109.45403 53 11
```

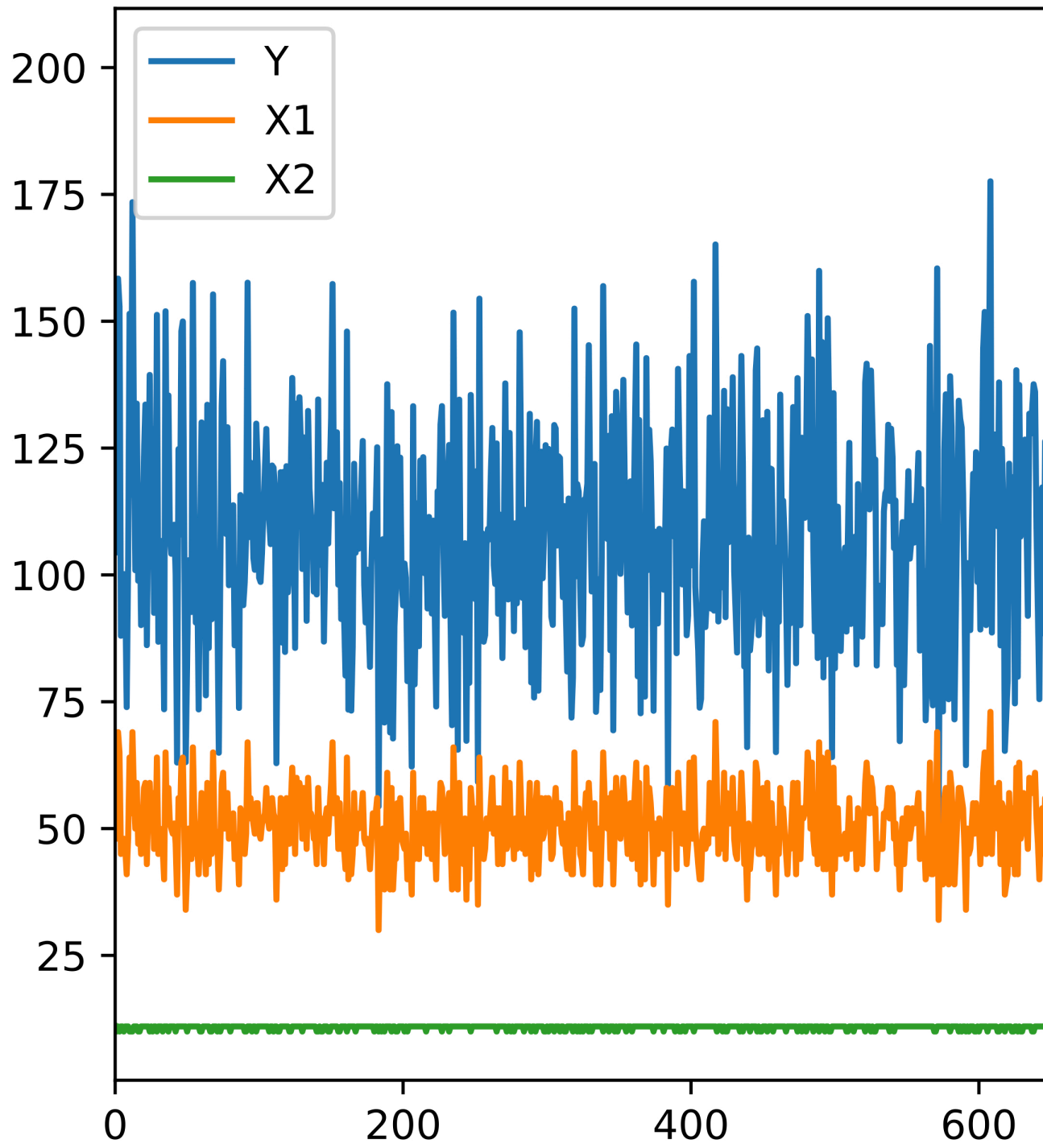
2.2.2 Create a plot in Python

File `create_graph.py`

```
import pandas as pd
import matplotlib.pyplot as plt

sim_data = pd.read_csv("sample_data.csv")

plt.figure()
sim_data.plot()
plt.savefig("plot.pdf", format = "pdf")
```



2.2.3 Run statistical model in R

We can estimate the model with R:

```
sim_data <- read.csv("sample_data.csv")
summary(lm(Y ~ X1 + X2, data = sim_data))

##
## Call:
## lm(formula = Y ~ X1 + X2, data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.3988 -1.9452 -0.0261  2.0216  9.1066
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.09087    2.54667    3.57 0.000374 ***
## X1             3.00531    0.01326  226.68 < 2e-16 ***
## X2            -4.94658    0.22876  -21.62 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.936 on 997 degrees of freedom
## Multiple R-squared:  0.9811, Adjusted R-squared:  0.981
## F-statistic: 2.585e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

2.2.4 Run statistical model in R

To save the output, we use the `sink` function.

File `estimate_model.R`

```
sink("estimation_summary.txt")
summary(lm(Y ~ X1 + X2, data = sim_data))
sink()
```

2.2.5 Makefile syntax

- `make` is a *command* that runs on a text file often named `Makefile`.
- A `Makefile` contains one or several blocks with the following structure:

```
targetfile: sourcefile(s)
[tab] command
```

2.2.6 Naive version

File: `Makefile`

```
sample_data.csv: simulate_data.R
    R CMD BATCH simulate_data.R
```

```
plot.pdf: create_graph.py
    python create_graph.py
```



```
estimation_summary.txt: estimate_model.R
    R CMD BATCH estimate_model.R
```

A simple call to `make` only builds the first target (`sample_data.csv`). To build the other targets, we have to use: `make plot.pdf` and `make estimation_summary.txt`.

2.2.7 Making all targets

File: Makefile

```
all: analysis
```

```
analysis: sample_data.csv plot.pdf estimation_summary.txt
```

```
sample_data.csv: simulate_data.R
    R CMD BATCH simulate_data.R
```

```
plot.pdf: create_graph.py
    python create_graph.py
```

```
estimation_summary.txt: estimate_model.R
    R CMD BATCH estimate_model.R
```

New data is simulated and saved in `sample_data.csv`. But `plot.pdf` and `estimation_summary.txt` are not updated.

2.2.8 Dealing with dependencies

- Problem `plot.pdf` and `estimation_summary.txt` depend on `sample_data.csv`.
- Solution: explicit dependencies.

File: Makefile

```
all: analysis
```

```
analysis: sample_data.csv plot.pdf estimation_summary.txt
```

```
sample_data.csv: simulate_data.R
    R CMD BATCH simulate_data.R
```

```
plot.pdf: sample_data.csv create_graph.py
    python create_graph.py
```

```
estimation_summary.txt: sample_data.csv estimate_model.R
    R CMD BATCH estimate_model.R
```

2.3 Git and GitHub

Guest lecture by Ista Zahn.

Chapter 3

Packages

We strongly recommend Wickham (2015).

We assume the following packages are installed:

```
install.packages(c("devtools", "roxygen2", "testthat", "knitr"))
```

3.1 Why?

- Organize your code
- Distribute your code
- Keep versions of your code

3.2 Package structure

- Folder hierarchy
 - NAMESPACE: package import / export
 - DESCRIPTION: metadata
 - R/: R code
 - man/: object documentation (with short examples)
 - tests/
 - data/
 - src/: compiled code
 - vignettes/: manual-like documentation
 - inst/: installed files
 - demo/: longer examples
 - exec, po, tools

3.3 Building steps

- R CMD build
- R CMD INSTALL
- R CMD check

3.3.1 R CMD build

```
R CMD build --help
```

Build R packages from package sources in the directories specified by 'pkgdirs'

3.3.2 R CMD INSTALL

```
R CMD INSTALL --help
```

Install the add-on packages specified by pkgs. The elements of pkgs can be relative or absolute paths to directories with the package sources, or to gzipped package 'tar' archives. The library tree to install to can be specified via '-library'. By default, packages are installed in the library tree rooted at the first directory in .libPaths() for an R session run in the current environment.

3.3.3 R CMD check

```
R CMD check --help
```

<http://r-pkgs.had.co.nz/check.html>

Check R packages from package sources, which can be directories or package 'tar' archives with extension 'tar.gz', 'tar.bz2', 'tar.xz' or 'tgz'.

A variety of diagnostic checks on directory structure, index and control files are performed. The package is installed into the log directory and production of the package PDF manual is tested. All examples and tests provided by the package are tested to see if they run successfully. By default code in the vignettes is tested, as is re-building the vignette PDFs.

3.3.4 Building steps with devtools

- devtools::build
- devtools::install
- devtools::check
- and many others: load_all, document, test, run_examples, ...

3.4 Create an R package

3.4.1 utils::package.skeleton

```
package.skeleton() # "in "clean" session ("anRpackage")
package.skeleton("pkgname") # in "clean" session

set.seed(02138)
f <- function(x, y) x+y
g <- function(x, y) x-y
d <- data.frame(a = 1, b = 2)
e <- rnorm(1000)
package.skeleton(list = c("f","g","d","e"), name = "pkgname")
```



Figure 3.1: Submitting to CRAN. It's not that bad...

3.4.2 devtools::create

```
devtools::create("path/to/package/pkgname")
```

Also from RStudio ('File -> New Project').

3.4.3 Submit to CRAN

Reading: <http://r-pkgs.had.co.nz/release.html>

3.5 R packages on GitHub

Reading: <http://r-pkgs.had.co.nz/git.html>

- Version control
- Website, wiki, project management
- Easy install: `install_github` from `devtools`
- Collaboration
- Issue tracking

3.5.0.1 RStudio and GitHub integration

Command line

```
# git init # already run when creating package with RStudio
git add *
git commit -m "First commit"
git remote add origin https://github.com/cchoirat/Linreg
git push -u origin master
```

3.5.1 .gitignore

RStudio default

```
.Rproj.user
.Rhistory
.RData
```

GitHub default

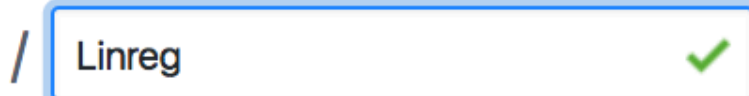
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about **upg**

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you already have a repository.

Add .gitignore: **None** ▼

Add a license: **None** ▼



Create repository

Figure 3.2: Create a new Linreg repository on GitHub

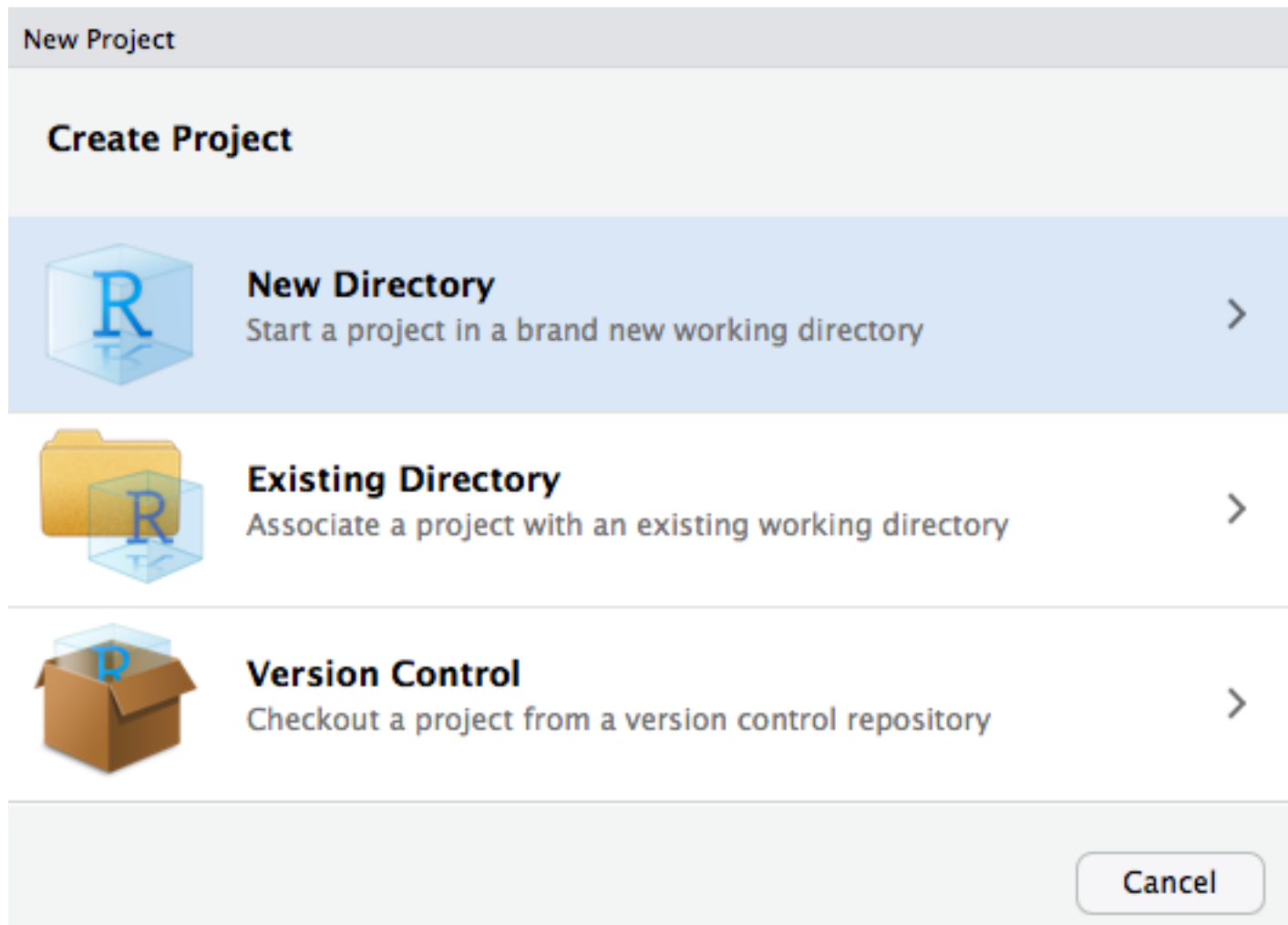


Figure 3.3: Create a new project in RStudio

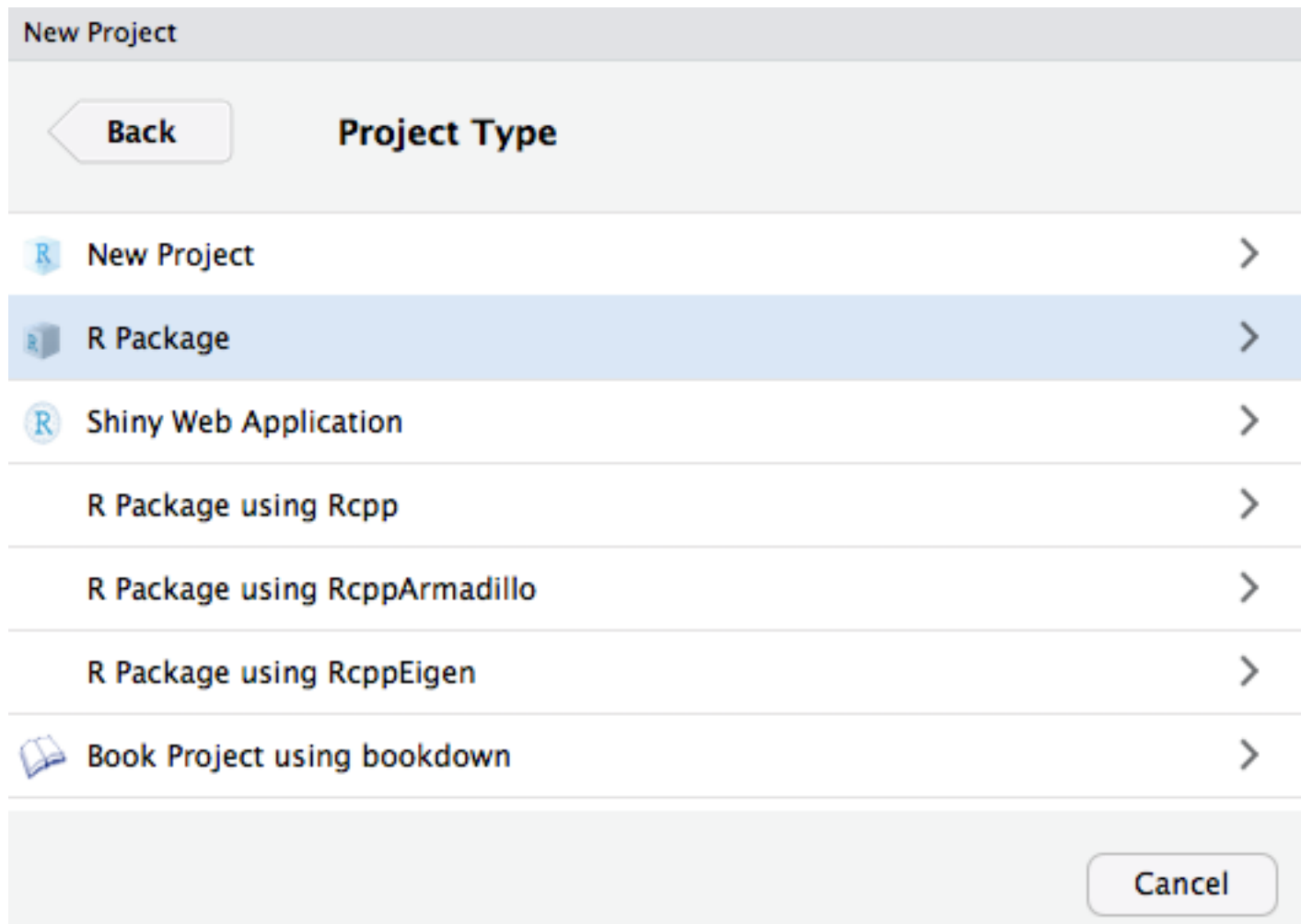



Figure 3.4: Select R package

New Project

[Back](#) **Create R Package**



Type: Package name:

Create package based on source files:

[Add...](#)
[Remove](#)

Create project as subdirectory of:

[Browse...](#)

☒ Create a git repository

☐ Open in new session

[Create Project](#) [Cancel](#)

Figure 3.5: Create the Linreg R package as a Git repository

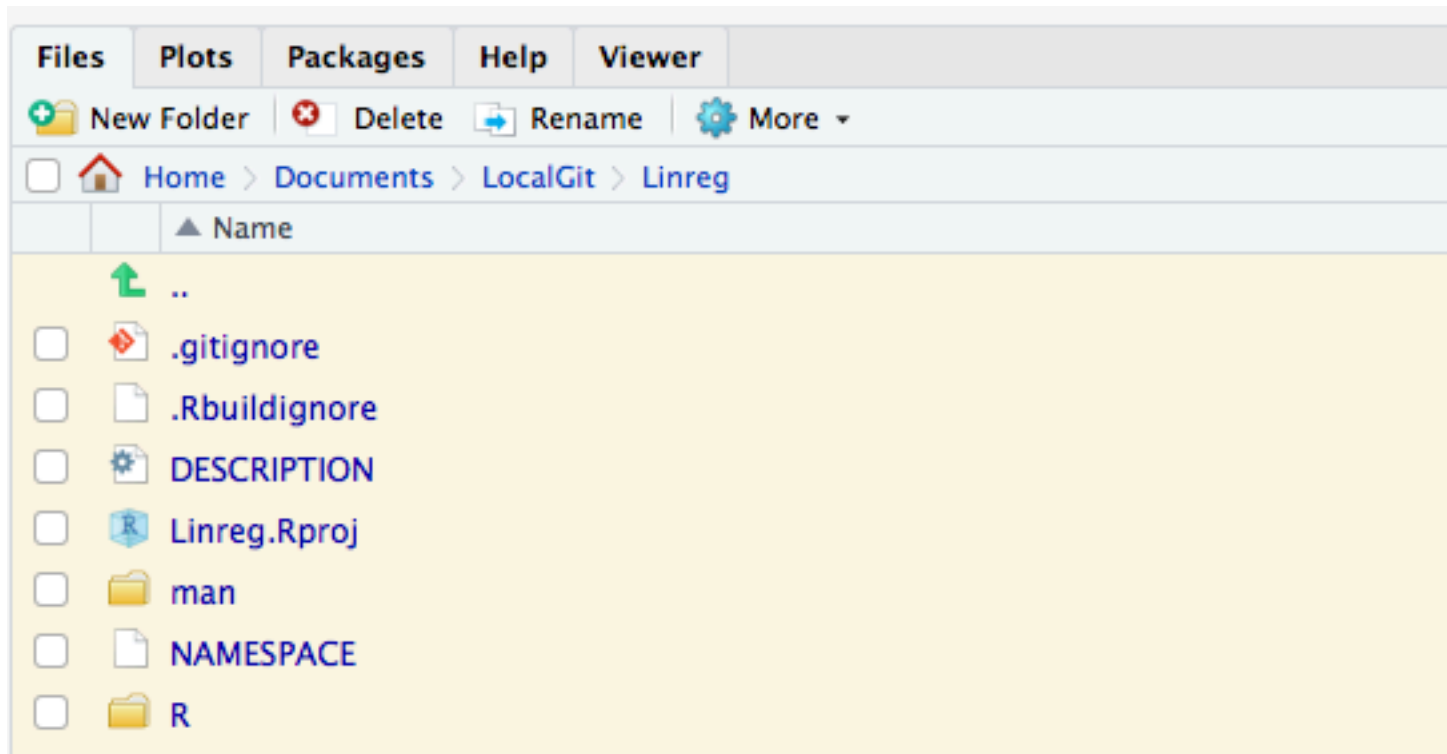


Figure 3.6: Automatically created files

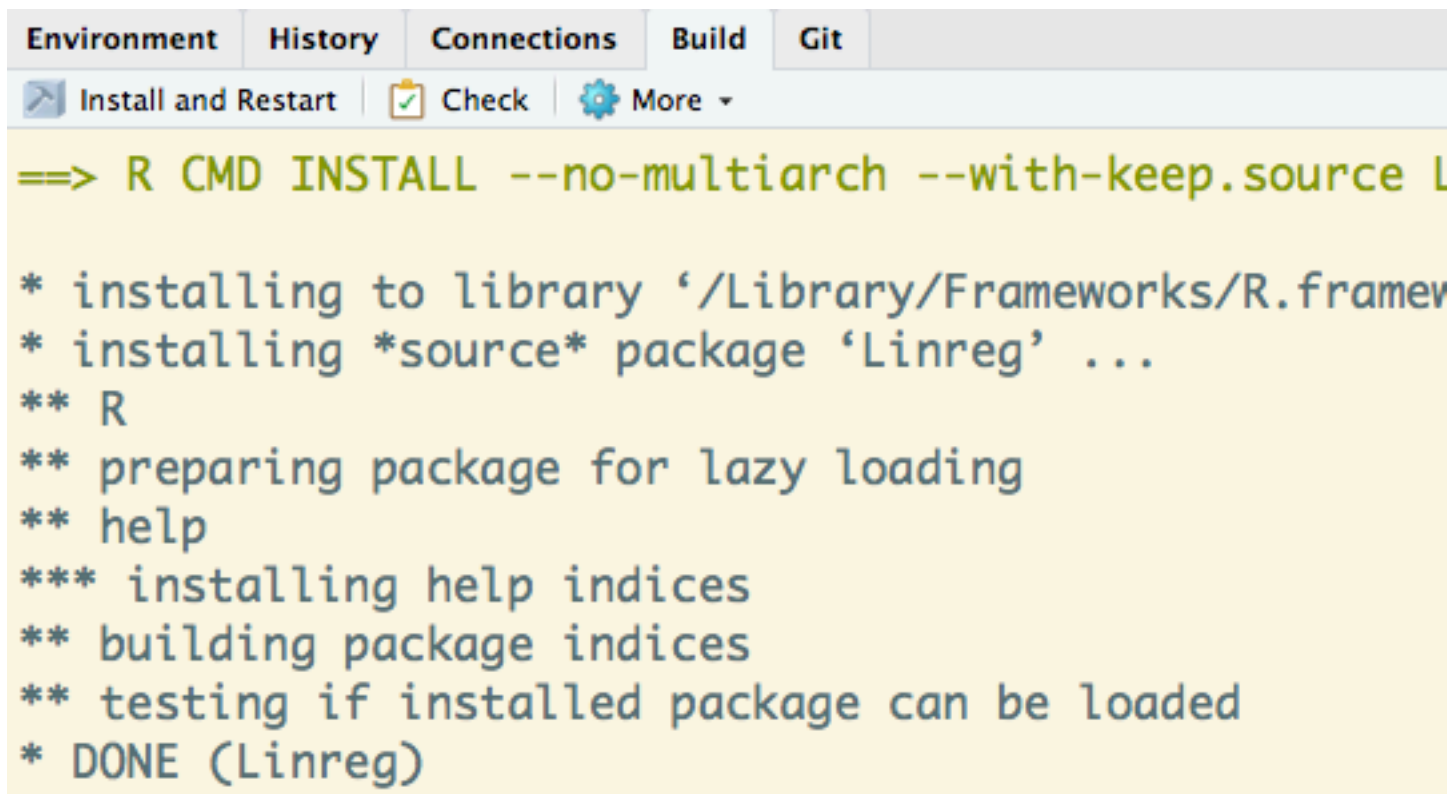



Figure 3.7: Build tab in RStudio

 **cchoirat / Linreg**

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📊 Insights

Quick setup — if you've done this kind of thing before

📄 Set up in Desktop

 or

HTTPS

SSH

`git@github.com:cchoirat/Linreg.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Linreg" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:cchoirat/Linreg.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:cchoirat/Linreg.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Figure 3.8: Github webpage

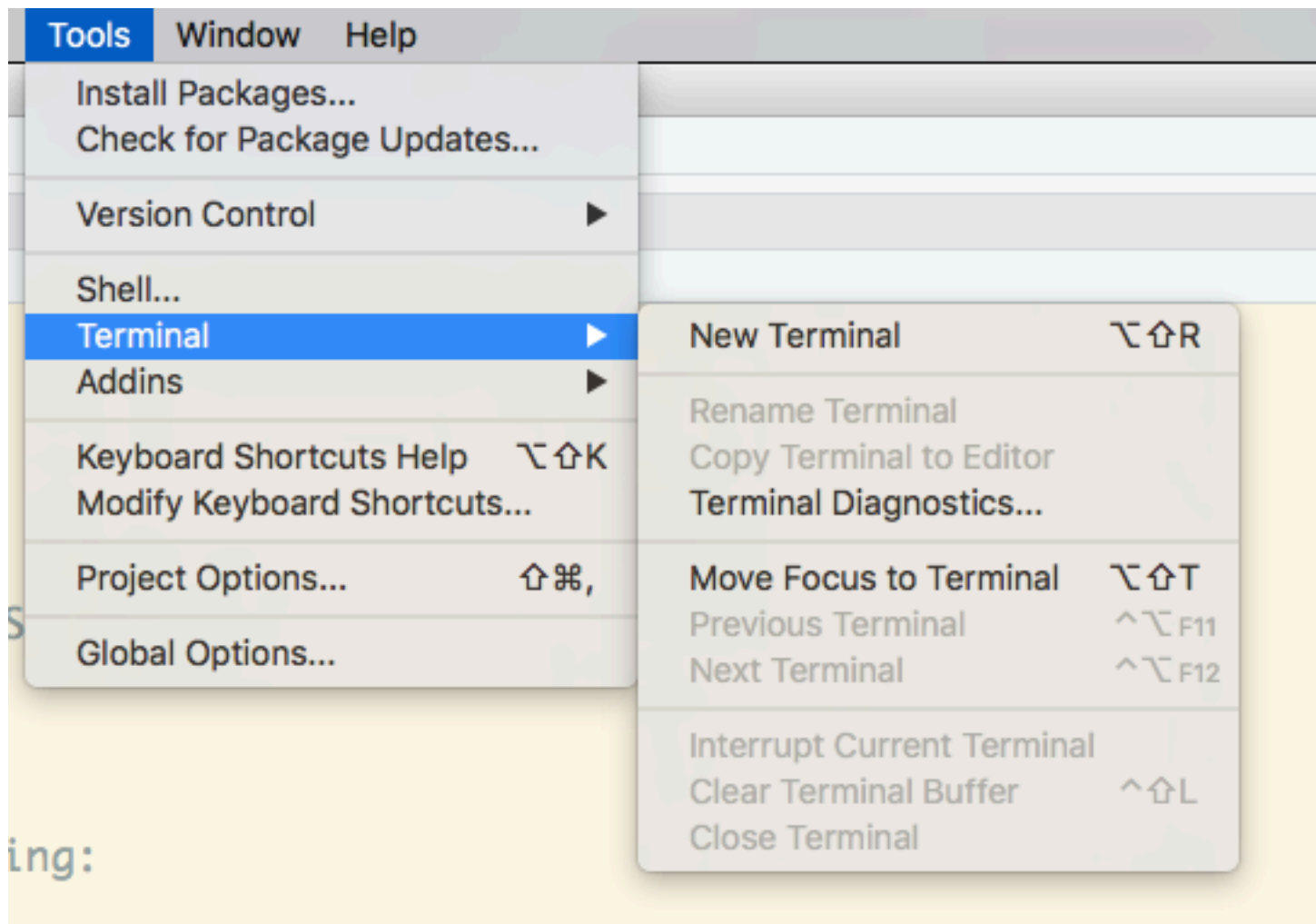



Figure 3.9: Open a terminal

 **cchoirat / Linreg**

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki


No description, website, or topics provided.

[Add topics](#)

🕒 1 commit

🔗 1 branch

Branch: master ▼ New pull request

 **cchoirat** First commit

📁 R	First com
📁 man	First com
📄 .Rbuildignore	First com
📄 .gitignore	First com
📄 DESCRIPTION	First com
📄 Linreg.Rproj	First com
📄 NAMESPACE	First com

Help people interested in this repository understand your project by adding a README

Figure 3.10: Github webpage is updated

```

# History files
.Rhistory
.Rapp.history

# Example code in package build process
*-Ex.R

# RStudio files
.Rproj.user/

# produced vignettes
vignettes/*.html
vignettes/*.pdf

```

3.6 RStudio projects

- .Rproj file extension, in our example `Linreg.Rproj`
- A project has its own:
 - R session
 - .Rprofile (*e.g.*, to customize startup environment)
 - .Rhistory
- Default working directory is project directory
- Keeps track of project-specific recent files

3.6.1 Project options

```

Version: 1.0

RestoreWorkspace: Default
SaveWorkspace: Default
AlwaysSaveHistory: Default

EnableCodeIndexing: Yes
UseSpacesForTab: Yes
NumSpacesForTab: 2
Encoding: UTF-8

RnwWeave: knitr
LaTeX: pdfLaTeX

AutoAppendNewline: Yes
StripTrailingWhitespace: Yes

BuildType: Package
PackageUseDevtools: Yes
PackageInstallArgs: --no-multiarch --with-keep.source

```

3.6.2 Package documentation

- Functions and methods
- Vignettes
 - PDF
 - knitr

3.7 Package workflow example

Creating R Packages: A Tutorial (Friedrich Leisch, 2009)

Our example is adapted from <https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf>.

3.7.1 Add linreg.R to R/ directory

```
linmodEst <- function(x, y) {
  ## CC: crossprod or a QR decomposition (as in the original version) are more efficient
  coef <- solve(t(x) %*% x) %*% t(x) %*% y
  print(coef)
  ## degrees of freedom and standard deviation of residuals
  df <- nrow(x) - ncol(x)
  sigma2 <- sum((y - x %*% coef) ^ 2) / df
  ## compute sigma^2 * (x'x)^-1
  vcov <- sigma2 * solve(t(x) %*% x)
  colnames(vcov) <- rownames(vcov) <- colnames(x)
  list(
    coefficients = coef,
    vcov = vcov,
    sigma = sqrt(sigma2),
    df = df
  )
}
```

3.7.2 Run our function

```
data(cats, package = "MASS")
linmodEst(cbind(1, cats$Bwt), cats$Hwt)

##           [,1]
## [1,] -0.3566624
## [2,]  4.0340627
## $coefficients
##           [,1]
## [1,] -0.3566624
## [2,]  4.0340627
##
## $vcov
##           [,1]      [,2]
## [1,]  0.4792475 -0.17058197
## [2,] -0.1705820  0.06263081
```

```
##
## $sigma
## [1] 1.452373
##
## $df
## [1] 142
```

We can compare the output with `lm`.

```
lm1 <- lm(Hwt ~ Bwt, data = cats)
lm1

##
## Call:
## lm(formula = Hwt ~ Bwt, data = cats)
##
## Coefficients:
## (Intercept)          Bwt
##    -0.3567         4.0341

coef(lm1)

## (Intercept)          Bwt
## -0.3566624     4.0340627

vcov(lm1)

##          (Intercept)          Bwt
## (Intercept)  0.4792475 -0.17058197
## Bwt         -0.1705820  0.06263081

summary(lm1)$sigma

## [1] 1.452373
```

3.7.3 Add ROxygen2 documentation

```
##' Linear regression
##'
##' Runs an OLS regression not unlike \code{\link{lm}}
##'
##' @param y response vector (1 x n)
##' @param X covariate matrix (p x n) with no intercept
##'
##' @return A list with 4 elements: coefficients, vcov, sigma, df
##'
##' @examples
##' data(mtcars)
##' X <- as.matrix(mtcars[, c("cyl", "disp", "hp")])
##' y <- mtcars[, "mpg"]
##' linreg(y, X)
##'
##' @export
##'
linmodEst <- function(x, y) {
  ## CC: crossprod or a QR decomposition (as in the original version) are more efficient
  coef <- solve(t(x) %*% x) %*% t(x) %*% y
```

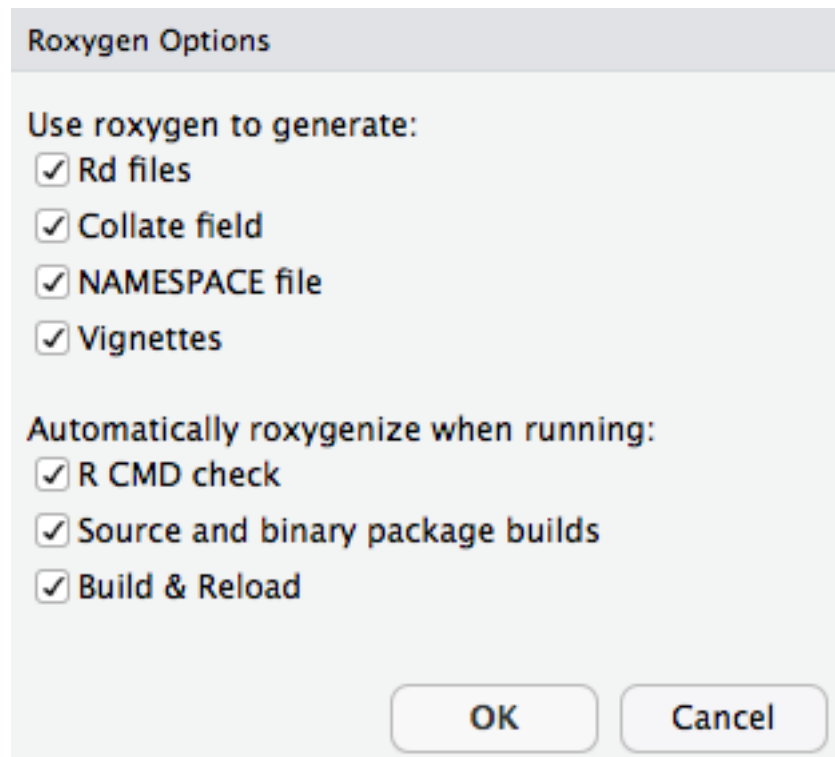



Figure 3.11: Roxygen options

```
print(coef)
## degrees of freedom and standard deviation of residuals
df <- nrow(x) - ncol(x)
sigma2 <- sum((y - x %*% coef) ^ 2) / df
## compute sigma^2 * (x'x)^-1
vcov <- sigma2 * solve(t(x) %*% x)
colnames(vcov) <- rownames(vcov) <- colnames(x)
list(
  coefficients = coef,
  vcov = vcov,
  sigma = sqrt(sigma2),
  df = df
)
}
```

3.7.4 Configure Build Tools

3.7.5 man page

File `man/linmodEst.Rd` contains:

```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/linreg.R
\name{linmodEst}
\alias{linmodEst}
\title{Linear regression}
```

```

\usage{
  linmodEst(x, y)
}
\arguments{
  \item{y}{response vector (1 x n)}

  \item{X}{covariate matrix (p x n) with no intercept}
}
\value{
  A list with 4 elements: coefficients, vcov, sigma, df
}
\description{
  Runs an OLS regression not unlike \code{\link{lm}}
}
\examples{
  data(mtcars)
  X <- as.matrix(mtcars[, c("cyl", "disp", "hp")])
  y <- mtcars[, "mpg"]
  linreg(y, X)
}

```

3.7.6 Formatted output

3.7.7 DESCRIPTION

Reading: <http://r-pkgs.had.co.nz/description.html>

```

Package: Linreg
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
  Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
RoxygenNote: 6.0.1

```

3.7.8 NAMESPACE

Reading: <http://r-pkgs.had.co.nz/namespace.html>, in particular Imports vs Suggests

export's automatically generated when parsing ROxygen2 snippets

```
export(linmodEst)
```

- A scary hack
- A scary tree

```
linmodEst {Linreg}
```

Linear regression

Description

Runs an OLS regression not unlike [lm](#)

Usage

```
linmodEst(x, y)
```

Arguments

y response vector (1 x n)
x covariate matrix (p x n) with no intercept

Value

A list with 4 elements: coefficients, vcov, sigma, df

Examples

```
data(mtcars)  
X <- as.matrix(mtcars[, c("cyl", "disp", "hp")])  
y <- mtcars[, "mpg"]  
linreg(y, X)
```







3.7.9 S3 basics

Reading: <http://adv-r.had.co.nz/S3.html>

```
hello <- function() {
  s <- "Hello World!"
  class(s) <- "hi"
  return(s)
}
```

```
hello()
```

```
## [1] "Hello World!"
## attr(,"class")
## [1] "hi"
```

```
print.hi <- function(...) {
  print("Surprise!")
}
```

```
hello()
```

```
## [1] "Surprise!"
```

3.7.10 S3 and S4 generics

Reading: <http://adv-r.had.co.nz/S4.html>

```
linmod <- function(x, ...)
  UseMethod("linmod")
```

```
linmod.default <- function(x, y, ...) {
  x <- as.matrix(x)
  y <- as.numeric(y)
  est <- linmodEst(x, y)
  est$fitted.values <- as.vector(x %*% est$coefficients)
  est$residuals <- y - est$fitted.values
  est$call <- match.call()
  class(est) <- "linmod"
  return(est)
}
```

3.7.11 print

```
print.linmod <- function(x, ...) {
  cat("Call:\n")
  print(x$call)
  cat("\nCoefficients:\n")
  print(x$coefficients)
}
```

```
x <- cbind(Const = 1, Bwt = cats$Bwt)
y <- cats$Hw
mod1 <- linmod(x, y)
```

```
##           [,1]
## Const -0.3566624
## Bwt    4.0340627
```

```
mod1
```

```
## Call:
## linmod.default(x = x, y = y)
##
## Coefficients:
##           [,1]
## Const -0.3566624
## Bwt    4.0340627
```

3.7.12 Other methods

- summary.linmod
- print.summary.linmod
- predict.linmod
- plot.linmod
- coef.linmod, vcov.linmod, ...

3.7.13 Formulas and model frames

```
linmod.formula <- function(formula, data = list(), ...) {
  mf <- model.frame(formula = formula, data = data)
```

```

x <- model.matrix(attr(mf, "terms"), data = mf)
y <- model.response(mf)
est <- linmod.default(x, y, ...)
est$call <- match.call()
est$formula <- formula
return(est)
}

linmod(Hwt ~ - 1 + Bwt * Sex, data = cats)

Call:
linmod.formula(formula = Hwt ~ -1 + Bwt * Sex, data = cats)

Coefficients:
      Bwt      SexF      SexM  Bwt:SexM
 2.636414  2.981312 -1.184088  1.676265

```

3.8 Unit testing

3.8.1 Unit tests and testthat

Reading: <http://r-pkgs.had.co.nz/tests.html>

In package directory:

```
devtools::use_testthat()
```

pre-populates test/testthat/

Test files should start with test to be processed.

3.8.2 test_coef.R

```

data(cats, package = "MASS")
l1 <- linmod(Hwt ~ Bwt * Sex, data = cats)
l2 <- lm(Hwt ~ Bwt * Sex, data = cats)

test_that("same estimated coefficients as lm function", {
  expect_equal(round(l1$coefficients, 3), round(l2$coefficients, 3))
})

> devtools::test()
Loading Linreg
Loading required package: testthat
Testing Linreg
.
DONE =====

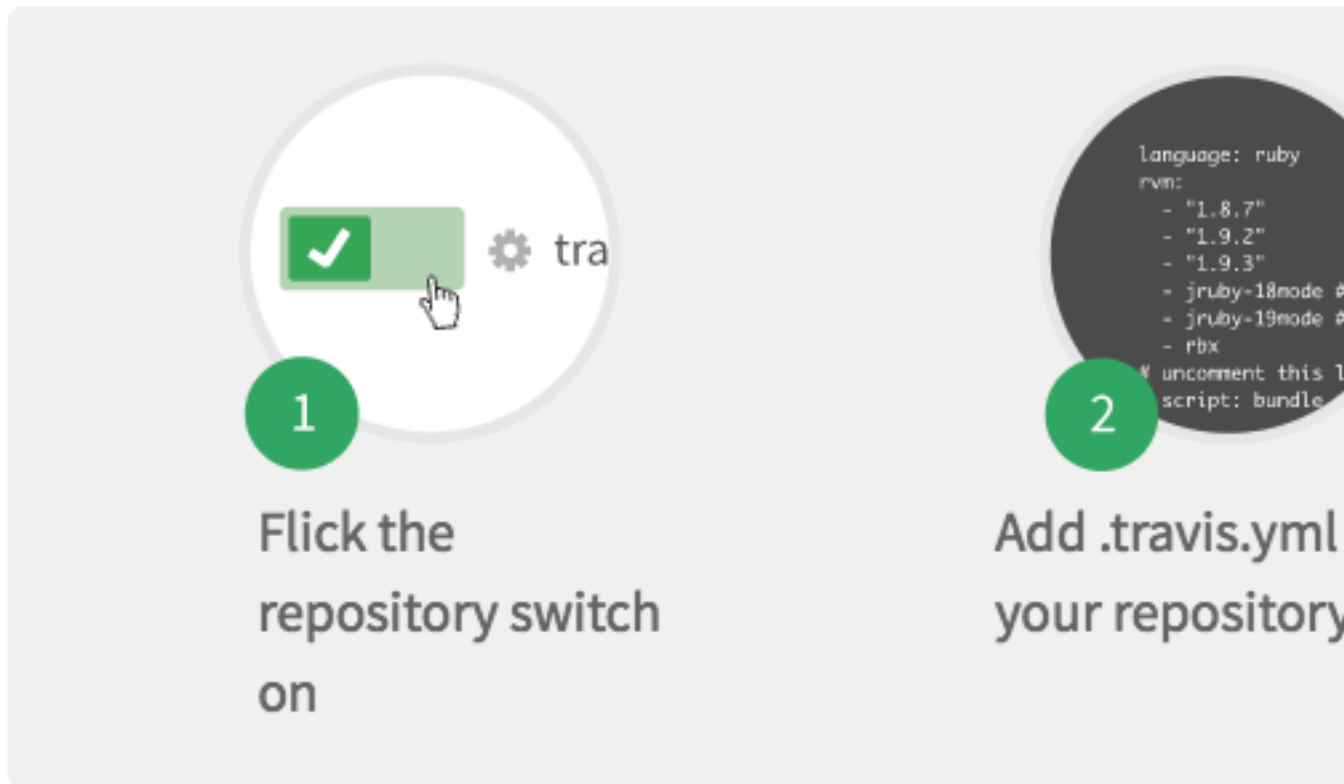
```

3.9 Continuous integration

Reading: <http://r-pkgs.had.co.nz/check.html#travis>

Website: <https://travis-ci.org/>

We're only showing your public repositories. You can



First step is to create a Travis account and link it to your GitHub account.

Travis will list all your public GitHub repositories for you to select the ones you want to test.

Calling

```
devtools::use_coverage(pkg = ".", type = c("codecov"))
```

creates the .travis.yml file:

```
# R for travis: see documentation at https://docs.travis-ci.com/user/languages/r

language: R
sudo: false
cache: packages
```

and pushing Linreg code to GitHub will automatically trigger a Travis build... which fails!

To be continued...



cchoirat/Linreg



cchoirat / Linreg (master)

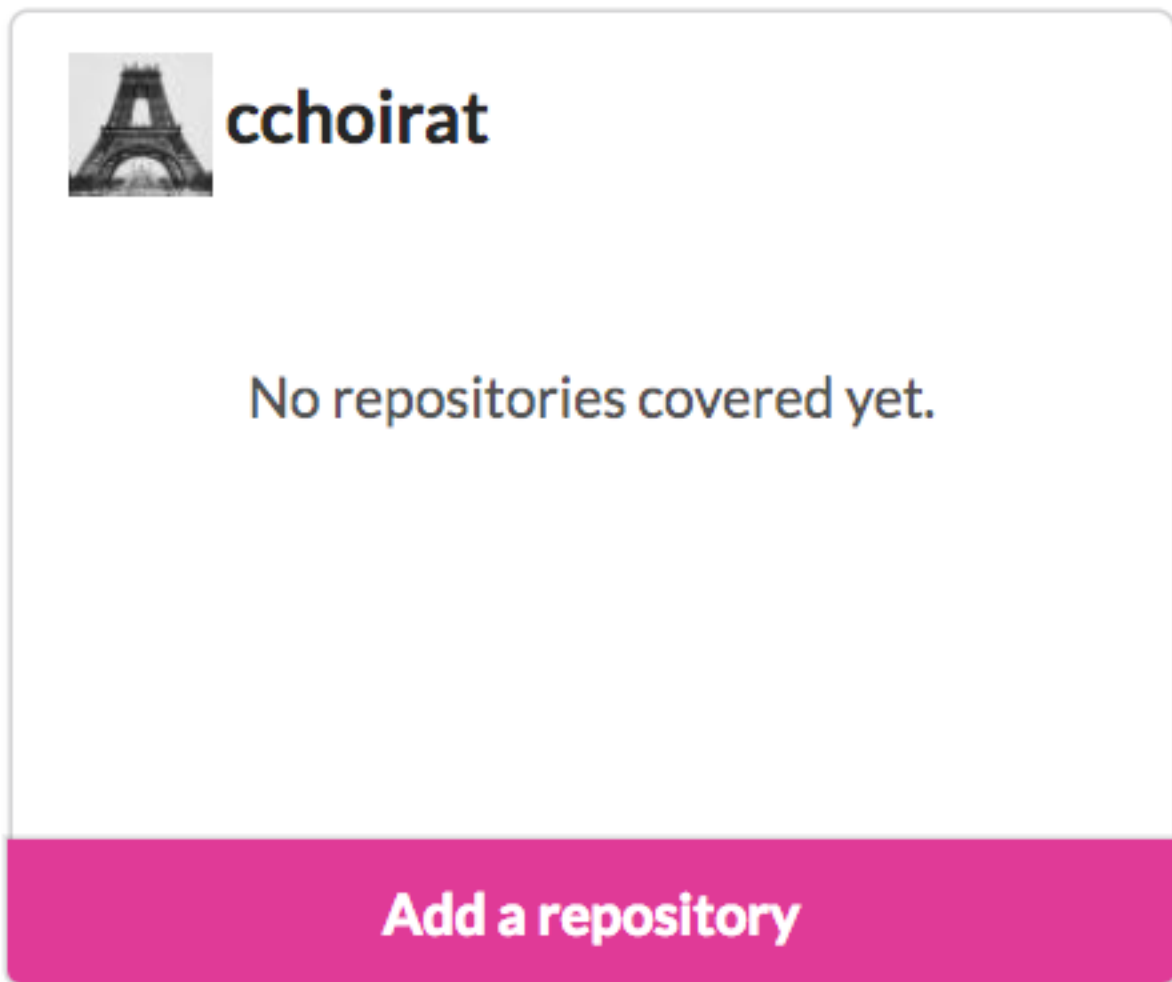


Build #1 failed.



cchoirat

Trying to trigger a build



3.10 Code coverage

Reading: <https://walczak.org/2017/06/how-to-add-code-coverage-codecov-to-your-r-package/>

Website: <https://codecov.io/>

Like Travis, codecov has to be linked to a GitHub account:

```
devtools::use_coverage(pkg = ".", type = c("codecov"))
```

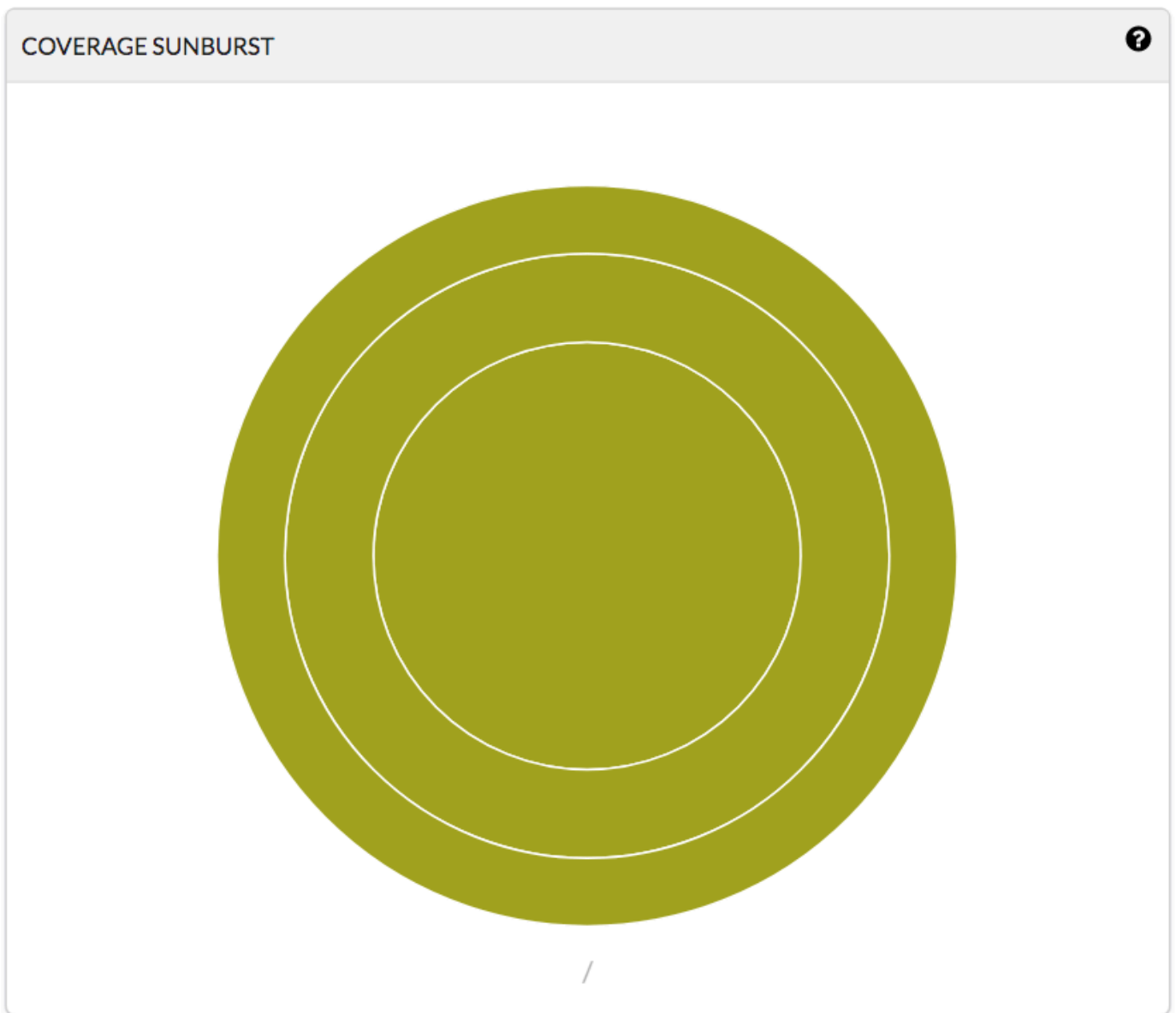
creates the `codecov.yml` file:

```
comment: false
```

A call to

```
covr::codecov(token = "YOUR_TOKEN")
```

will give you code coverage information:



Files	
	R/linreg.R
Project Totals (1 files)	



3.11 Back to GitHub

Badges can be added to README.md:

```
<!-- Badges ----->
[![Travis (LINUX) Build Status](https://travis-ci.org/cchoirat/Linreg.svg?branch=master)](https://travis-ci.org/cchoirat/Linreg)
[![codecov](https://codecov.io/gh/cchoirat/Linreg/branch/master/graph/badge.svg)](https://codecov.io/gh/cchoirat/Linreg)

## `Linreg` package template

Based on "Creating R Packages: A Tutorial" (Friedrich Leisch, 2009)

- https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf
```

are are automatically displayed on GitHub:

3.12 Vignettes

Reading: <http://r-pkgs.had.co.nz/vignettes.html>

Reading: http://kbroman.org/pkg_primer/pages/vignettes.html

Even if all the functions and datasets of your package are documented, it is still useful to have a more detailed illustration on how to use your package. A *vignette* is the right place to explain a workflow and a statistical method.

Running:

```
devtools::use_vignette("my-linear-regression")
```

creates a `vignettes` folder and provide a template in RMarkdown format `my-linear-regression.Rmd`:

<https://github.com/cchoirat/Linreg/blob/master/vignettes/my-linear-regression.Rmd>

It also indicates in DESCRIPTION that vignettes should be built with `knitr`.

```
VignetteBuilder: knitr
```

The vignette is built into a HTML document with

```
devtools::build_vignettes()
```

Building Linreg vignettes

Moving `my-linear-regression.html`, `my-linear-regression.R` to `inst/doc/`

Copying `my-linear-regression.Rmd` to `inst/doc/`

The vignette is accessible with

```
vignette("my-linear-regression")
vignette("my-linear-regression", package = "Linreg")
```

Vignette Title

Vignette Author

2017-10-21

Vignettes are long form documentation commonly included in packages. Because of the distribution of the package, they need to be as compact as possible. The `html_vignette` custom style sheet (and tweaks some options) to ensure that the resulting html is as `html_vignette` format:

- Never uses retina figures
- Has a smaller default figure size
- Uses a custom CSS stylesheet instead of the default Twitter Bootstrap style

Vignette Info

Note the various macros within the `vignette` section of the metadata block above. These instruct R how to build the vignette. Note that you should change the `title` field and match the title of your vignette.

Styles

The `html_vignette` template includes a basic CSS theme. To override this theme you can in the document metadata as follows:

```
output:
  rmarkdown::html_vignette:
    css: mystyles.css
```


Chapter 4

Optimization

In this Chapter, we will see how to measure and improve code performance.

4.1 Measuring performance

4.1.1 Benchmarking

Reading: <http://adv-r.had.co.nz/Performance.html#microbenchmarking>

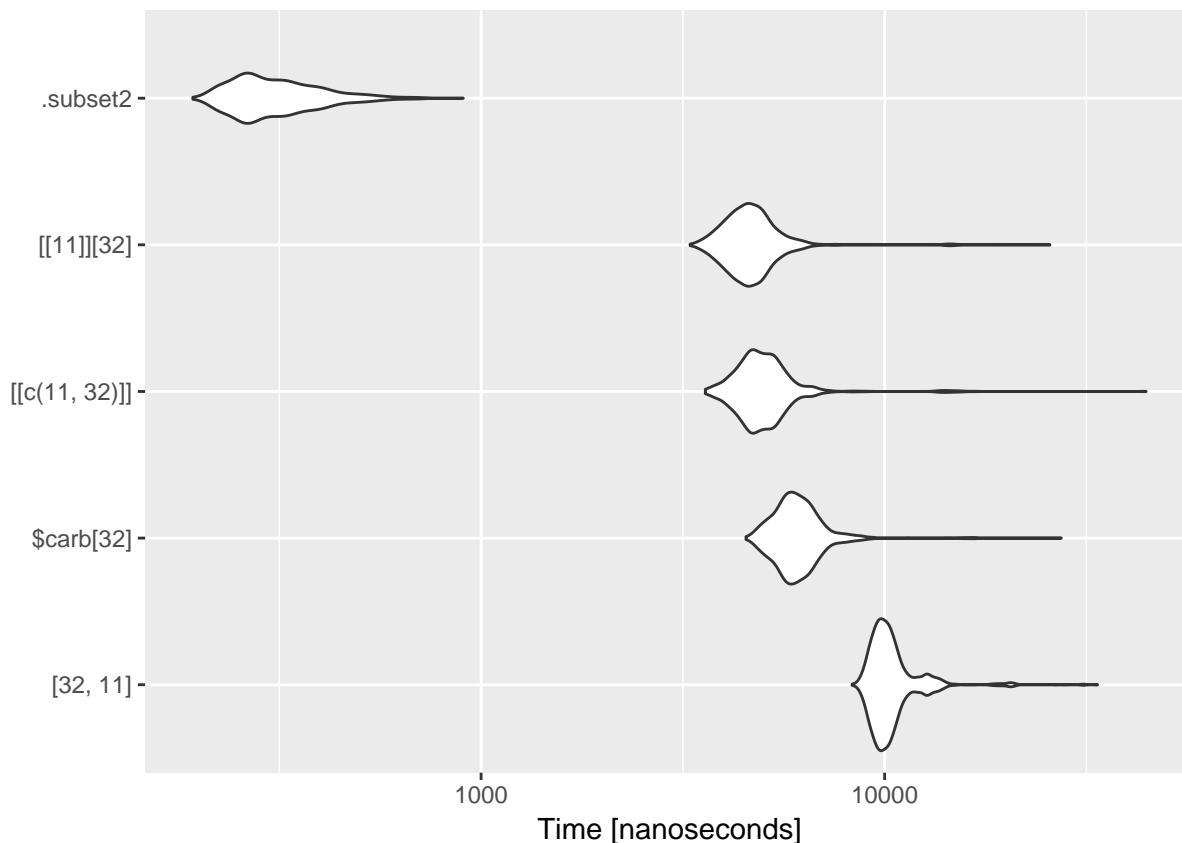
There are several ways to benchmark code (see http://www.alexejgossmann.com/benchmarking_r/) from `system.time` to dedicated packages such as `rbenchmark` (Kusnierczyk (2012)) or `microbenchmark` (Mersmann (2015)).

Let's start with an example from Wickham (2014).

```
library(microbenchmark)
m <- microbenchmark(
  times = 1000, # default is 100
  "[32, 11]"    = mtcars[32, 11],
  "$carb[32]"   = mtcars$carb[32],
  "[[c(11, 32)]]" = mtcars[[c(11, 32)]],
  "[[11]][32]"  = mtcars[[11]][32],
  ".subset2"    = .subset2(mtcars, 11)[32]
)
m
```

```
## Unit: nanoseconds
##      expr   min      lq      mean  median      uq   max neval  cld
##   [32, 11] 8304 9559.0 10559.731 10066.0 10689.5 33687  1000    e
##   $carb[32] 4533 5576.0  6198.955  5994.5  6510.5 27395  1000    d
##  [[c(11, 32)]] 3595 4534.5  5132.417  4916.0  5368.5 44446  1000    c
##  [[11]][32] 3295 4225.5  4761.957  4599.5  5007.0 25681  1000    b
##    .subset2   193  258.0   320.960   297.0   361.0   903  1000    a
```

```
ggplot2::autoplot(m)
```



4.1.2 Profiling and optimization

Reading: <http://adv-r.had.co.nz/Profiling.html#measure-perf>

Let's compare three ways of estimating a linear regression: with built-in `lm` and with two functions we defined in package `Linreg` in Chapter 3.

```
data(cats, package = "MASS")
fit1 <- lm(Hwt ~ Bwt, data = cats)
fit2 <- linmod(Hwt ~ Bwt, data = cats)
fit3 <- linmodEst(cbind(1, cats$Bwt), cats$Hwt)
```

```
##           [,1]
## [1,] -0.3566624
## [2,]  4.0340627
```

```
all.equal(round(coef(fit1), 5), round(coef(fit2), 5))
```

```
## [1] TRUE
```

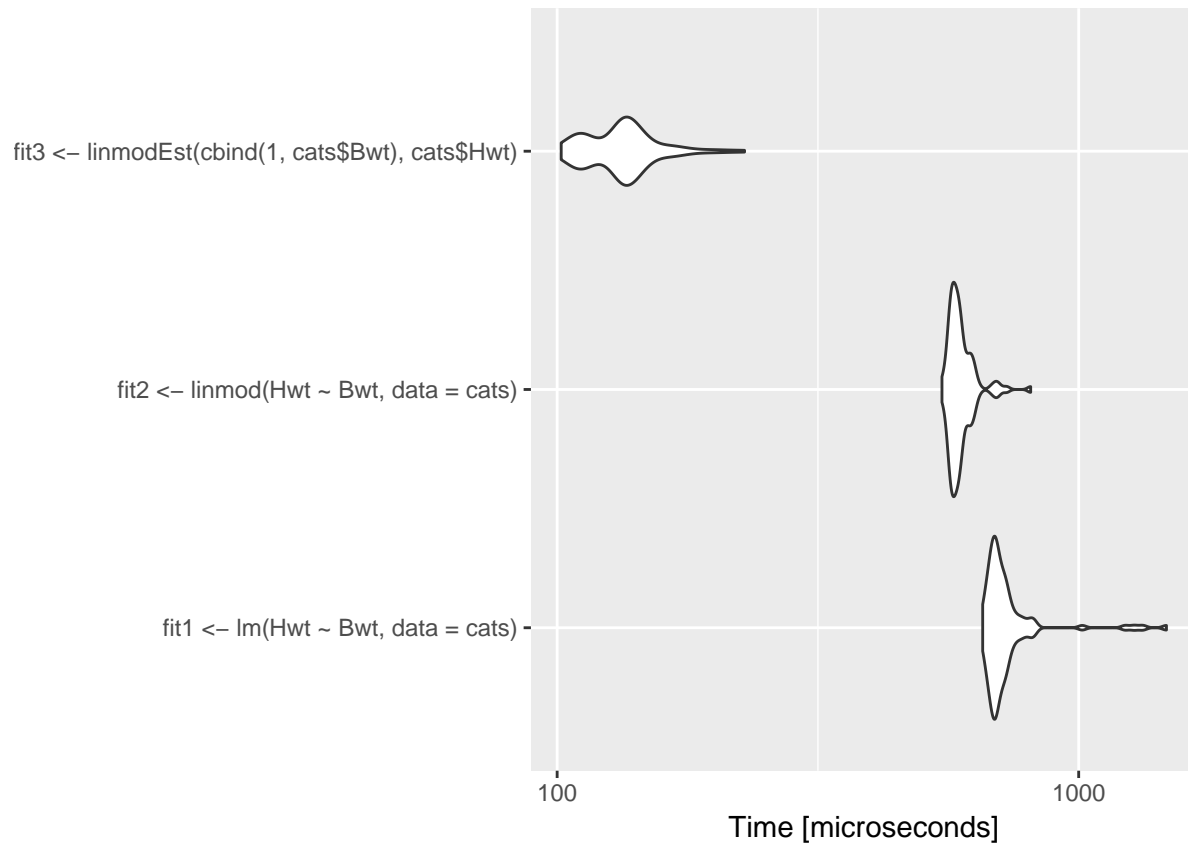
```
all.equal(round(coef(fit1), 5), round(fit3$coefficients, 5), check.names = FALSE)
```

```
## [1] "Attributes: < names for current but not for target >"
## [2] "Attributes: < Length mismatch: comparison on first 0 components >"
## [3] "target is numeric, current is matrix"
```

```
m <- microbenchmark(
  fit1 <- lm(Hwt ~ Bwt, data = cats),
  fit2 <- linmod(Hwt ~ Bwt, data = cats),
```


[illegible]

[illegible]



4.2 Improving performance

- Vectorize
- Parallelize
- Use a faster language (C/C++, Fortran, ...)
- Use different tools (as in Chapter 6)

4.3 Vectorization

Let's take an example from a blog post (that seems to be gone). It's used in Wickham (2014, Section Case studies).

```

vacc1a <- function(age, female, ily) {
  p <- 0.25 + 0.3 * 1 / (1 - exp(0.04 * age)) + 0.1 * ily
  p <- p * if (female) 1.25 else 0.75
  p <- max(0, p)
  p <- min(1, p)
  p
}

set.seed(1959)
n <- 1000
age <- rnorm(n, mean = 50, sd = 10)

```

```

female <- sample(c(T, F), n, rep = TRUE)
ily <- sample(c(T, F), n, prob = c(0.8, 0.2), rep = TRUE)

vacc1a(age[1], female[1], ily[1])

## [1] 0.1667005
vacc1a(age[2], female[2], ily[2])

## [1] 0.4045439
vacc1a(age[3], female[3], ily[3])

## [1] 0.2699324
vacc1a is not designed for vector inputs
vacc1a(age, female, ily)

## Warning in if (female) 1.25 else 0.75: the condition has length > 1 and
## only the first element will be used
## [1] 0.2526293
It should be called
vacc1a(age[1], female[1], ily[1])

## [1] 0.1667005
vacc1a(age[2], female[2], ily[2])

## [1] 0.4045439
vacc1a(age[3], female[3], ily[3])

## [1] 0.2699324
We can use a loop:
out <- numeric(n)
for (i in 1:n)
  out[i] <- vacc1a(age[i], female[i], ily[i])

or one of the apply functions:
vacc0<- function(age, female, ily) {
  sapply(1:n, function(i) vacc1a(age[i], female[i], ily[i]))
}

out0 <- vacc0(age, female, ily)

all.equal(out, out0)

## [1] TRUE

```

But, it's convenient for the function to support vector inputs, instead of relying on users writing their own wrappers. We can loop inside the function body.

```

vacc1 <- function(age, female, ily) {
  n <- length(age)
  out <- numeric(n)
  for (i in seq_len(n)) {
    out[i] <- vacc1a(age[i], female[i], ily[i])
  }
}

```



```

}
out
}

```

or we can rely on base R functions that accept vector inputs

```

vacc2 <- function(age, female, ily) {
  p <- 0.25 + 0.3 * 1 / (1 - exp(0.04 * age)) + 0.1 * ily
  p <- p * ifelse(female, 1.25, 0.75)
  p <- pmax(0, p)
  p <- pmin(1, p)
  p
}

```

4.4 Parallelization

```

library(parallel)
cores <- detectCores()
cores

```

```
## [1] 8
```

```

vacc3 <- function(age, female, ily) {
  mcmapply(function(i) vacc1a(age[i], female[i], ily[i]), 1:n, mc.cores = cores - 1)
}

```

```
out3 <- vacc3(age, female, ily)
```

```

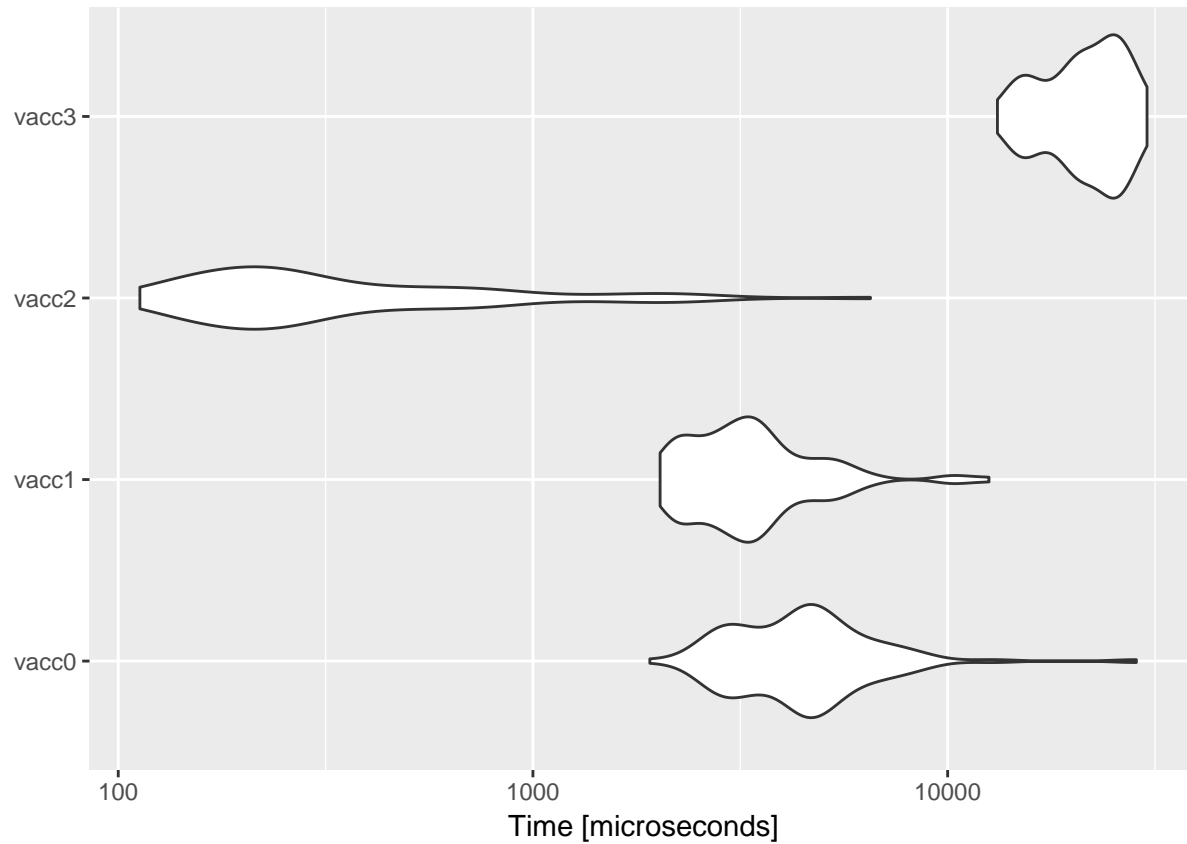
library(microbenchmark)
m <- microbenchmark(
  vacc0 = vacc0(age, female, ily),
  vacc1 = vacc1(age, female, ily),
  vacc2 = vacc2(age, female, ily),
  vacc3 = vacc3(age, female, ily)
)
m

```

```
## Unit: microseconds
##   expr      min       lq      mean     median        uq      max neval
## vacc0  1913.810  3272.610  4860.9381  4511.481  5360.9050 28508.167   100
## vacc1   2025.942  2617.384  3589.6382  3248.014  3810.6850 12578.677   100
## vacc2   112.774   186.666   518.0049   260.377   490.4655  6513.684   100
## vacc3 13182.759 18173.791 21784.7975 21765.854 25633.0875 30251.287   100
##   cld
##    c
##   b
##   a
##   d

```

```
ggplot2::autoplot(m)
```



So, what's going on?

We will talk more about parallelization tools and techniques in Chapter ‘?(bigdata).

4.5 Introduction to C++

- C++ is a very powerful object-oriented language.
- Many tutorials are available on-line, for example <http://www.cplusplus.com/doc/tutorial/>.
- R is *intepreted*, C++ is *compiled* and typically much faster (in loops for examples).
- Our introduction to C++ is from an R perspective. Python (and most interpreted languages) can be extended with C++ too.

4.5.1 Rcpp

Reading: <http://adv-r.had.co.nz/Rcpp.html>

- Rcpp Eddelbuettel (2013) makes it very easy to use C++ code in R (for example to speed up a function or to wrap methods already implemented in C++).
- Rcpp provides “syntactic sugar” that makes is easy to leverage C++ even without a deep knowledge of it.
- To use Rcpp, you need a C++ compiler:

- Windows: Rtools
- OS X: Xcode
- Linux: `r-base-dev` from package manager

4.5.2 Hello World!

```
library(Rcpp)
cppFunction('void hello(){
  Rprintf("Hello, world!");
}')
hello
```

```
## function ()
## invisible(.Primitive(".Call")(<pointer: 0x106cbc100>))
hello()
```

```
## Hello, world!
```

`Rprintf` is the counterpart of C++ `printf` function.

Let's take the first example of Wickham (2014), Section Getting started with C++.

```
cppFunction('int add(int x, int y, int z) {
  int sum = x + y + z;
  return sum;
}')
```

We have to specify the input type and the output type. As expected

```
add(1, 2, 3)
```

returns 6. How about?

```
add(1.1, 2.2, 3.3)
```

```
cppFunction('double addd(double x, double y, double z) {
  double sum = x + y + z;
  return sum;
}')
```

With `addd` we do get 6.6:

```
addd(1.1, 2.2, 3.3)
```

4.5.3 sourceCpp

When C++ code takes more than a couple of lines, it's more convenient to create a stand-alone C++ source file.

From the RStudio default template:

```
#include <Rcpp.h>
using namespace Rcpp;

NumericVector timesTwo(NumericVector x) {
  return x * 2;
}
```

```
/** R
timesTwo(42)
*/
```

From R, we can use `sourceCpp` to access `timesTwo` in R:

```
sourceCpp("src/times-two.cpp")
timesTwo(100)
```

4.5.4 Data types

```
int double bool string
NumericVector LogicalVector IntegerVector CharacterVector
NumericMatrix IntegerMatrix LogicalMatrix CharacterMatrix
NA_REAL NA_INTEGER NA_STRING NA_LOGICAL
List DataFrame Function
...
```

4.5.5 Sugar

Reading: <http://adv-r.had.co.nz/Rcpp.html#rcpp-sugar>.

- Vectorization of `+`, `*`, `-`, `/`, `pow`, `<`, `<=`, `>`, `>=`, `==`, `!=`, `!`

-x‘

Exercise 4.1. Can you write an Rcpp function similar to `addd` but accepting vector arguments?

```
cppFunction('NumericVector addv(NumericVector x, NumericVector y, NumericVector z) {
  NumericVector sum = x + y + z;
  return sum;
}')
```

4.5.6 Example (continued)

```
#include <Rcpp.h>
using namespace Rcpp;

double vacc3a(double age, bool female, bool ily){
  double p = 0.25 + 0.3 * 1 / (1 - exp(0.04 * age)) + 0.1 * ily;
  p = p * (female ? 1.25 : 0.75);
  p = std::max(p, 0.0);
  p = std::min(p, 1.0);
  return p;
}

// [[Rcpp::export]]
NumericVector vacc3(NumericVector age, LogicalVector female,
                   LogicalVector ily) {
  int n = age.size();
  NumericVector out(n);
```

```

for(int i = 0; i < n; ++i) {
  out[i] = vacc3a(age[i], female[i], ily[i]);
}

return out;
}

```

4.5.7 Back to Linreg

- `armadillo` is a very powerful C++ linear algebra library: <http://arma.sourceforge.net/>
- It can be used in `Rcpp` via the `RcppArmadillo` package.

Exercise 4.2. Can you write an `Rcpp` function similar to `linmodEst`?

```

linmodEst <- function(x, y) {
  ## CC: crossprod or a QR decomposition (as in the original version) are more efficient
  coef <- solve(t(x) %*% x) %*% t(x) %*% y
  print(coef)
  ## degrees of freedom and standard deviation of residuals
  df <- nrow(x) - ncol(x)
  sigma2 <- sum((y - x %*% coef) ^ 2) / df
  ## compute sigma^2 * (x'x)^-1
  vcov <- sigma2 * solve(t(x) %*% x)
  colnames(vcov) <- rownames(vcov) <- colnames(x)
  list(
    coefficients = coef,
    vcov = vcov,
    sigma = sqrt(sigma2),
    df = df
  )
}

```

4.6 Rcpp packages

Readings: - <https://cran.r-project.org/web/packages/Rcpp/vignettes/Rcpp-package.pdf> - <http://adv-r.had.co.nz/Rcpp.html#rcpp-package>

Chapter 5

Databases

5.1 Overview

5.2 SQL

5.3 noSQL

5.4 R interfaces

Chapter 6

Big data

6.1 List of tools

Reading: Varian (2014) (PDF available)

Spark? h2o? More? Let's go back to the bottlenecks

- CPU
- RAM
- I/O

6.2 Data that fits in memory

6.2.1 Faster I/O

Reading: <https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html>

`data.table` provides an enhanced of a `data.frame` and faster I/O with `fread` and `fwrite`.

To read the 0.5GB ratings file from MovieLens

```
library(data.table)
system.time(ratings <- fread("~/Dropbox/Data17/ml-20m/ratings.csv"))
```

takes

```
Read 20000263 rows and 4 (of 4) columns from 0.497 GB file in 00:00:05
```

user	system	elapsed
4.007	0.229	4.244

while

```
system.time(ratings <- read.csv("~/Dropbox/Data17/ml-20m/ratings.csv"))
```

takes

user	system	elapsed
85.199	2.711	90.997

There are ways to improve the speed of `read.csv` (for example, by specifying column types). But in general `fread` is much faster.

Tools for Manipulating Big Data

<i>Google name</i>	<i>Analog</i>	<i>Description</i>
Google File System	Hadoop File System	This system supports files stored distributed across hundreds of computers.
Bigtable	Cassandra	This is a table of data that is part of the System. It too can stretch across many machines.
MapReduce	Hadoop	This is a system for accessing and processing data in large data structures. MapReduce allows you to access data using hundreds or thousands of machines. The data you are interested in is sent to the machines and is then processed. The different shards of the data are then combined ("reduced") into a summary table you are interested in.
Sawzall	Pig	This is a language for creating and manipulating data.
Go	None	Go is flexible open-source, compiled computer language that supports parallel data processing.
Dremel, BigQuery	Hive, Drill, Impala	This is a tool that allows data to be queried in a simplified form of of Structured Query Language (SQL). With Dremel it is possible to query on a petabyte of data in a few seconds.

Table 6.1: I/O comparison

package	function.	speed	output
base	read.csv	slow	data.frame
data.table	fread	very fast	data.table
readr	read_csv	fast	tibble

```
library(readr) # in tidyverse
system.time(ratings <- read_csv("~/Dropbox/Data17/ml-20m/ratings.csv"))
```

```
      user  system elapsed
10.290    3.037   18.450
```

also tends to perform better than `read.csv`.

6.2.2 Reference vs copy

6.3 Data that doesn't fit in memory (but fits on drive)

6.4 Pure R solutions

6.4.1 Sampling

6.4.2 bigmemory

6.4.3 Database connections and lazy evaluation

6.5 Scaling up

6.5.1 Parallel computing and clusters

6.5.2 Cloud computing

6.5.3 Spark

Reading: <https://spark.rstudio.com/>

```
library(sparklyr)
spark_install(version = "2.1.0")
```

```
conf <- spark_config()
conf$`sparklyr.shell.driver-memory` <- "32G"
conf$spark.memory.fraction <- 0.5
sc <- spark_connect(master = "local")
```

```
library(dplyr)
iris_tbl <- copy_to(sc, iris)
flights_tbl <- copy_to(sc, nycflights13::flights, "flights")
batting_tbl <- copy_to(sc, Lahman::Batting, "batting")
src_tbls(sc)
```

```
top_rows <- read.csv("~/Dropbox/Data17/AirFlights/allyears.csv", nrow = 5)
file_columns <- top_rows %>%
  purrr::map(function(x) "character")
rm(top_rows)
```

```
sp_flights <- spark_read_csv(sc,
                             name = "flights2",
                             path = "~/Dropbox/Data17/AirFlights/allyears.csv",
                             memory = FALSE,
                             columns = file_columns,
                             infer_schema = FALSE)
```

```
flights_table <- sp_flights %>%
  mutate(DepDelay = as.numeric(DepDelay),
         ArrDelay = as.numeric(ArrDelay),
         SchedDeparture = as.numeric(CRSDepTime)) %>%
  select(Origin, Dest, SchedDeparture, ArrDelay, DepDelay, Month, DayofMonth)

flights_table %>% head
```

Cache data:

```
sp_flights %>%
  tally # takes a loooooong time
```

123534969...

```
subset_table <- flights_table %>%
  compute("flights_subset")
```

```
subset_table %>%
  tally # a bit faster.
```

123534969 as well!

```
lm(arr_delay ~ distance, data = flights_tbl)
ml_linear_regression(subset_table, response = "ArrDelay", features = "SchedDeparture")
```

TODOL change the config arguments of the connection

6.5.4 h2o and Sparkling Water

Reading: <https://spark.rstudio.com/h2o.html>

6.5.5 More?

GPU

Chapter 7

Visualization

7.1 Principles of visualization

7.2 Maps and GIS

Bibliography

- Blackwell, M. and Sen, M. (2012). Large datasets and you: A field guide. *The Political Methodologist*, 20(1):2–5.
- Eddelbuettel, D. (2013). *Seamless R and C++ Integration with Rcpp*. Springer, New York. ISBN 978-1-4614-6867-7.
- Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19.
- Kusnierczyk, W. (2012). *rbenchmark: Benchmarking routine for R*. R package version 1.0.0.
- Lim, A. and Tjhi, W. (2015). *R High Performance Programming*. Community experience distilled. Packt Publishing.
- Mersmann, O. (2015). *microbenchmark: Accurate Timing Functions*. R package version 1.4-2.1.
- Varian, H. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, 28(2):3–28.
- Wickham, H. (2014). *Advanced R*. Chapman & Hall/CRC The R Series. Taylor & Francis.
- Wickham, H. (2015). *R Packages*. O’Reilly Media, Inc., 1st edition.