Computing for Big Data (BST-262)

Christine Choirat

2017-09-15

Contents

Chapter 1

Introduction

1.1 Prerequisites

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

1.2 Syllabus

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, continuous integration, code coverage

Week 3 - Software optimization

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

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)

Week 5 - Analyzing data that does not fit in memory

- Lecture 9. Pure R solutions (sampling, ff and ffbase, other interpreters). JVM solutions (h20, 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)

1.3 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.

1.4 Software packages

We will mostly use R in this course. Some examples will be run in Python. In genral, we will use free and open-source software programs such as PostgreSQL or Spark.

1.5 Datasets

1.5.1 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, notably:

1. Small (1MB)
2. Benchmark (~190MB zipped)

```
ratings <- read.csv("data/ml-latest-small/ratings.csv")
head(ratings)
```

```
##   userId movieId rating  timestamp
## 1      1      31    2.5 1260759144
## 2      1    1029    3.0 1260759179
## 3      1    1061    3.0 1260759182
## 4      1    1129    2.0 1260759185
## 5      1    1172    4.0 1260759205
## 6      1    1263    2.0 1260759151
```

```
links <- read.csv("data/ml-latest-small/links.csv")
head(links)
```

```
##   movieId imdbId tmdbId
## 1       1 114709    862
## 2       2 113497   8844
## 3       3 113228  15602
## 4       4 114885  31357
## 5       5 113041  11862
## 6       6 113277    949
```

```
movies <- read.csv("data/ml-latest-small/movies.csv")
head(movies)
```

```
##   movieId                                title
## 1       1                    Toy Story (1995)
## 2       2                      Jumanji (1995)
## 3       3             Grumpier Old Men (1995)
## 4       4            Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                           Heat (1995)
##                                         genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                              Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller
```

```
tags <- read.csv("data/ml-latest-small/tags.csv")
head(tags)
```

```
##   userId movieId                     tag  timestamp
## 1     15     339 sandra 'boring' bullock 1138537770
## 2     15    1955                  dentist 1193435061
## 3     15    7478                 Cambodia 1170560997
## 4     15   32892                  Russian 1170626366
## 5     15   34162              forgettable 1141391765
## 6     15   35957                    short 1141391873
```

1.5.2 Airlines data

1.5.3 Census

1.5.4 Health claims

1.5.5 GIS: $PM_{2.5}$ exposure

$PM_{2.5}$ exposure

1.5.6 Methylation?

1.5.7 Genomics??

1.5.8 GWAS

Chapter 2

Basic tools

2.1 Command line tools

2.2 Git and GitHub

Chapter 3

Packages

3.1 Why?

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

3.2 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`

3.4 `R CMD build`

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

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

3.5 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.

3.6 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.

3.7 Building steps with devtools

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

3.8 Creating an R package

3.8.1 utils::package.skeleton

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

3.8.2 devtools::create

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

3.9 Submitting to CRAN

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

3.10 Using GitHub

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

3.11 RStudio and GitHub integration (1 / 7)

3.12 RStudio and GitHub integration (2 / 7)

3.13 RStudio and GitHub integration (3 / 7)

3.14 RStudio and GitHub integration (4 / 7)

3.15 RStudio and GitHub integration (5 / 7)

3.16 Command line

```
git init
git add *
git commit -m "First commit"
git remote add origin git@github.com:harvard-P01/pkgtemplate.git
git push -u origin master
```

3.17 RStudio and GitHub integration (6 / 7)

3.18 RStudio and GitHub integration (7 / 7)

3.19 Installing from GitHub

```
devtools::install_github("harvard-P01/pkgtemplate")
```

```
devtools::install_github("harvard-P01/pkgtemplate",
                         build_vignettes = TRUE)
```

3.20 `.gitignore` (RStudio default)

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

3.21 `.gitignore` (GitHub default)

```
# History files
.Rhistory
.Rapp.history

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

# RStudio files
.Rproj.user/

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

3.22 RStudio projects

- `.Rproj` file extension, in our example `pkgtemplate.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

3.23 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
```

3.24 Package documentation

- Functions and methods
- Vignettes
  - PDF
  - `knitr` (or `Sweave`)

3.25 Process example

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

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

3.26 Adding `linreg.R` in `R/` directory

```
linmodEst <- function(x, y) {
  ## compute QR-decomposition of x
  qx <- qr(x)
  ## compute (x'x)^(-1) x'y
  coef <- solve.qr(qx, y)
  ## 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 * chol2inv(qx$qr)
  colnames(vcov) <- rownames(vcov) <- colnames(x)
  list(
    coefficients = coef,
    vcov = vcov,
    sigma = sqrt(sigma2),
    df = df
  )
}
```

3.27 Running our function

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

```
## $coefficients
## [1] -0.3566624  4.0340627
##
## $vcov
##            [,1]        [,2]
## [1,]  0.4792475 -0.17058197
## [2,] -0.1705820  0.06263081
##
## $sigma
## [1] 1.452373
##
## $df
## [1] 142
```

3.28 And compare with `lm` (1 / 2)

```
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
```

3.29 And compare with `lm` (2 / 2)

```
vcov(lm1)
```

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

```
summary(lm1)$sigma
```

```
## [1] 1.452373
```

3.30 Adding 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) {
  ## compute QR-decomposition of x
  qx <- qr(x)
  ## compute (x'x)^(-1) x'y
  coef <- solve.qr(qx, y)
  ## 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 * chol2inv(qx$qr)
  colnames(vcov) <- rownames(vcov) <- colnames(x)
  list(
    coefficients = coef,
    vcov = vcov,
    sigma = sqrt(sigma2),
    df = df
  )
}
```

3.31 Configure Build Tools

3.32 man/linmodEst.Rd

```
% Generated by roxygen2 (4.1.1): do not edit by hand
% Please edit documentation in R/linmodEst.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"]
linmodEst(y, X)
}
```

3.33 Formatted output

3.34 DESCRIPTION

```
Package: pkgtemplate
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1
Date: 2015-10-24
Author: Who wrote it
Maintainer: Who to complain to <yourfault@somewhere.net>
Description: More about what it does (maybe more than one line)
License: What license is it under?
LazyData: TRUE
```

3.35 NAMESPACE

export's automatically generated when parsing ROxygens2 snippets

```
export(linmodEst)
```

3.36 S3 basics

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

hello()
```

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

3.37 S3 basics

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

hello()
```

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

3.38 S3 and S4 generics

```
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)
}
```

3.39 print

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

3.40 print

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

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

3.41 Other methods

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

3.42 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)
}
```

3.43 Unit tests and `testthat`

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.

3.44 `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(l1$coefficients, l2$coefficients)
})
```

```
==> devtools::test()

Loading pkgtemplate
Loading required package: testthat
Testing pkgtemplate
.
Woot!
```

3.45 Vignettes

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

```
devtools::use_vignette("linmod")
```

https://github.com/harvard-P01/pkgtemplate/blob/master/vignettes/linmod.Rmd

Chapter 4

Optimization

In this Chapter, we are going to see how to measure and improve code performance

4.1 Measuring performance

4.1.1 Profiling

4.1.2 Benchmarking

4.2 Improving performance

4.2.1 Introduction to C/C++

4.2.2 Rcpp

Chapter 5

Databases

5.1 Overview

5.2 SQL

5.3 noSQL

5.4 R interfaces

Chapter 6

Big data

In this Chapter, we are going to review different approaches to handle and perform analyses on *data that does not fit in memory.*

6.1 Reading big data (that fits in memory)

6.1.1 R package comparison

6.1.2 Python

6.2 Sampling (can be read, not analyzed easily)

6.3 Pure R solutions

6.4 JVM solutions

6.4.1 h20

6.4.2 Spark

Chapter 7

Visualization

We have finished a nice book.

Bibliography

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.