Statistical computing MATH10093
Computer lab 3
Solutions

Finn Lindgren

30/1/2019

Summary

In this lab session you will experiment with RMarkdown, and explore model assessment methods using predictive scores. You will not hand in anything, but you should keep your code script file for later use.

1. Download all the R files for Lab 3; `lab03code.R`, `RMdemo.Rmd`, and `RMtemplate.Rmd`. The questions and solutions (including the detailed marking scheme) from last year's Coursework A is also available.

   If you're running on your own computer, make sure you have the `rmarkdown` package installed; If `library(rmarkdown)` gives an error, run `install.packages("rmarkdown")`

2. Make sure that `Tools`→`Global Options`→`Sweave`→`Weave` is set to `knitr`.

3. Open `RMdemo.Rmd` in RStudio and press `Ctrl-Shift-K` to generate a PDF version.

   There is also an button labeled `Knit` at the top of the editor window that can be pressed; when the mouse hovers over it, it will say `Knit the current document`.

   A third option is `File`→`Knit Document`.

   The R function call `rmarkdown::render('RMdemo.Rmd')` should also work, or by providing the full pathname of the file.

4. Open `CWAcode.Rmd` and read through the code. This is a RMarkdown document, so you can generate and read a pdf version if you prefer to read that.

5. Make a copy of the `RMtemplate.Rmd` file and open it. Note that is contains a commented line that would automatically include the code from `CWAcode.R`. Change this to `source` the `lab03code.R` code.

6. Copy the commented plotting code from the end of `lab03code.R` (and only that code) into your new document, and generate a `pdf` report.

7. Use the estimation and prediction code from Lecture 3 estimate, predict, and check, for the simple model where y =`actual` and x =`cad` in the simple Lecture 3 model.

   First, split the data into estimation and testing parts:

```
obs <- sample(rep(c(TRUE, FALSE), c(0.75, 0.25) * nrow(printer_data)),
              size = nrow(printer_data),
              replace = FALSE)
data_obs <- printer_data[obs, ]
data_test <- printer_data[!obs, ]
```

   Then, define a new `model_Z` function that takes a `data.frame` parameter `data` instead of a vector `x`, and uses the `cad` column of the `data.frame` in the model matrix construction.

```
## Solution:
model_Z <- function(data) {
  Z0 <- model.matrix(~ 1 + cad, data = data)
  list(ZE = cbind(Z0, Z0 * 0), ZV = cbind(Z0 * 0, Z0))
}
```

   This model class isn't the true model for this data, so we don't have a `theta_true` value to test against. Instead, use the constant-variance model `theta_compare = c(0, 1, 6, 0)` as a reference model.

   Now follow the recipe from the lecture and longform notes. The main steps are

   (a) Estimate $\boldsymbol{\theta}$ and obtain $\boldsymbol{\Sigma}_\theta$.

   (b) Compute predictions using `model_predict()` to plot prediction intervals as functions of `cad`. Remember that you need to supply a `data.frame` with a column `cad` to predict in this model.

   (c) Predict for the estimation data and test data separately.

   (d) Compute and compare scores using `score_se()` and `score_qign()`.

   Note: This lab question has a close similarity to part of the coursework (for 2018); take the opportunity to aks questions about the lab and the code in the Prediction and Proper Scoring Rules notes (which you should feel free to liberally copy into your lab file, for the parts that you need that are not in the `lab03code.R` file). Remember that the lab solutions are also available.

```r
Z_obs <- model_Z(data_obs)
opt <- optim(rep(0, 4), fn = neg_log_lik, Z = Z_obs, y = data_obs$actual,
             method = "BFGS", hessian = TRUE)
theta_hat <- opt$par
Sigma_theta <- solve(opt$hessian)

## Prediction intervals:

theta_compare <- c(0, 1, 6, 0)
x_plot <- data.frame(cad = seq(10, 300, length=100))
pred_plot1 <- model_predict(theta_compare, x_plot, type = "observation")
pred_plot2 <- model_predict(theta_hat, x_plot, Sigma_theta = Sigma_theta, type = "observation")
plot(x_plot$cad, pred_plot1$mu, type = "l",
     ylim = range(pred_plot1[, c("lwr", "upr")],
                  pred_plot2[, c("lwr", "upr")]),
     xlab = "cad", ylab = "actual")
lines(x_plot$cad, pred_plot2$mu, lty = 1, col = 4)
lines(x_plot$cad, pred_plot1$lwr, lty = 2)
lines(x_plot$cad, pred_plot1$upr, lty = 2)
lines(x_plot$cad, pred_plot2$lwr, lty = 2, col = 4)
lines(x_plot$cad, pred_plot2$upr, lty = 2, col = 4)
points(actual ~ cad, data = data_obs, pch=20)
points(actual ~ cad, data = data_test, pch=20, col=2)

## Test scores:

opred1 <- model_predict(theta_compare, data_obs, type = "observation")
opred2 <- model_predict(theta_hat, data_obs, Sigma_theta = Sigma_theta, type = "observation")
tpred1 <- model_predict(theta_compare, data_test, type = "observation")
tpred2 <- model_predict(theta_hat, data_test, Sigma_theta = Sigma_theta, type = "observation")

## SE for observed and test data
rbind(
  c(mean(score_se(opred1, data_obs$actual)),
    mean(score_se(opred2, data_obs$actual))),
  c(mean(score_se(tpred1, data_test$actual)),
    mean(score_se(tpred2, data_test$actual)))
)

##          [,1]     [,2]
## [1,] 412.3290 372.5723
## [2,] 452.5479 378.7195

## DS for observed and test data
rbind(
  c(mean(score_ds(opred1, data_obs$actual)),
    mean(score_ds(opred2, data_obs$actual))),
  c(mean(score_ds(tpred1, data_test$actual)),
    mean(score_ds(tpred2, data_test$actual)))
```

```
)

##          [,1]     [,2]
## [1,] 7.022061 6.192742
## [2,] 7.121754 6.036440

# As should be expected, the model that knows about the variable variance
# scores better (lower) than the constant variance model.
# In the Coursework, the challenge is to take the colour information into
# account, to further improve the predictions.
```