Statistical computing MATH10093
Coursework A
Solutions and marking guidance

Finn Lindgren

March 2018 (comments 2019)

Marking guidance in italics. The answers need not be flowing text; in many cases just the R code is sufficient.

General marks:

- *PDF produced by RMarkdown (or knitr) [+3 marks]*
- *Typeset non-trivial mathematical formula (trivial example: θ, non-trivial examples: $y_i = \theta_1 + \theta_2$, $f(\theta) = \exp(\theta)$) [+1 marks per non-trivial formula, in total at most +4 marks]*
- *Readable code, useful code comments (comments only needed where code is non-self-explanatory) [+3 marks]*

When reasonable, initial mistakes should not give mark deductions when used in later questions (in particular, if Q2 is not solved correctly, Q3 should be marked as if it were, even if the solution uses a mock-up in place of the actual Q2 answer).

Make sure to look at the question document, hints document, and lab 4 (2019: lab 3) solutions in combination with these coursework solutions.

The second order finite difference formula was provided in an announcement on Learn, as it wasn't explicitly stated in the lectures.

```
source("CWAcode.R") # 2019 lab 3: lab03code.R
```

Question 1

Only the function definition is required here. The remaining 4 marks are awarded in the result discussion of Q3.
A direct solution, directly from on the definition [+4 marks]
Including the `alpha` *parameter (with or without default value) [+2 marks]*
Correctness of code is the important part here. It does not have to be pretty.

```
score_interval <- function(pred, y, alpha = 0.05) {
  L <- pred$lwr
  U <- pred$upr
  U - L + 2 / alpha * (L - y) * (y < L) + 2 / alpha * (y - U) * (y > U)
}
```

An alternative solution:

```
score_interval <- function(pred, y, alpha = 0.05) {
  L <- pred$lwr
  U <- pred$upr
  U - L + 2 / alpha * pmax(0, L - y) + 2 / alpha * pmax(0, y - U)
}
```

Note: Using the `I()` function, as `I(y > U)`, is out of place, but is not contradicting the documentation, so no mark deduction should be made.

Question 2

Determine what the 12 parameter model structure is. [+3 marks]

The 4-parameter model from the lab had $E(y_i) = \theta_1 + x_i\theta_2$ and $\log V(y_i) = \theta_3 + x_i\theta_4$, where x_i is the CAD prediction for observation i. From the code that generated the data we can see that that same model structure now holds within each of the three material colours, black, red, and blue. This means that for the expectation model we should have three intercept parameters and three slope parameters, and the same for the log-variance model.

Some answers may include attempts at writing the entire model as a formula. If the $y = Z_E\theta + \epsilon$ formulation is used, a note on the variance model for ϵ_i should be included.

Construct the `model_Z_cw` function for the determined 12 parameter model [+3 marks].
A brute force solution for `model_Z` (only the `Z0` definition differs from the model in the lab):

```
model_Z_cw <- function(data) {
  Z0 <- cbind(data$colour == "black",
              data$colour == "red",
              data$colour == "blue")
  Z0 <- cbind(Z0, data$cad * Z0) ## Using the vector vector-reusing feature
  list(ZE = cbind(Z0, Z0 * 0), ZV = cbind(Z0 * 0, Z0))
}
```

An alternative, compact solution, based on the lab and hints:

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

The possible choice of using the default intercept (leaving out `-1` or including `1`) is also valid, but gives a different parameter interpretation. The code should not contradict the textual/mathematical explanation for the model.

Split the data, as in lab 4 [+2 marks]

```
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, ]
```

Estimate θ [+2 marks]:

```
model_Z <- model_Z_cw
Z_obs <- model_Z(data_obs)
opt <- optim(rep(0, 12), fn = neg_log_lik, Z = Z_obs, y = data_obs$actual,
             method = "BFGS", hessian = TRUE)
theta_hat_cw <- opt$par
Sigma_theta_cw <- solve(opt$hessian)
theta_hat_cw

##  [1]  1.301341638 -0.958994899  0.073126843  1.060764934  0.932765641
##  [6]  1.105957125  2.100559527 -0.511852349 -0.912861487  0.007710524
## [11]  0.028489087  0.028103715
```

Question 3

Re-estimate the model from lab 4 [+3 marks]
The cleanest solution is to ignore the `model_Z` function from `CWAcode.R` for the lab model, since it doesn't take a `data.frame` as input, and define a new version that works just the same as `model_Z_cw` from Q2:

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

Code from the lab, with updated variable names:

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

Compute predictions. Code from the lab, with some extra copy/modify [+3 marks]

```
## Test scores:

model_Z <- model_Z_lab
opred1 <- model_predict(theta_hat_lab, data_obs, Sigma_theta = Sigma_theta_lab,
                        type = "observation", alpha=1/3)
tpred1 <- model_predict(theta_hat_lab, data_test, Sigma_theta = Sigma_theta_lab,
                        type = "observation", alpha=1/3)
model_Z <- model_Z_cw
opred2 <- model_predict(theta_hat_cw, data_obs, Sigma_theta = Sigma_theta_cw,
                        type = "observation", alpha=1/3)
tpred2 <- model_predict(theta_hat_cw, data_test, Sigma_theta = Sigma_theta_cw,
                        type = "observation", alpha=1/3)
```

Compute the SE, DS, and Interval scores [+2 marks here, and +2 marks for Q1]
From the lab, plus an extra block for `score_interval`*:*

```
## 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,] 372.5723 147.9068
## [2,] 378.7195 268.0411
```

```
## 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,] 6.192742 4.824360
## [2,] 6.036440 5.680567
```

```
## Interval Score for observed and test data
rbind(
  c(mean(score_interval(opred1, data_obs$actual, alpha = 1/3)),
    mean(score_interval(opred2, data_obs$actual, alpha = 1/3))),
  c(mean(score_interval(tpred1, data_test$actual, alpha = 1/3)),
    mean(score_interval(tpred2, data_test$actual, alpha = 1/3)))
)
```

```
##          [,1]     [,2]
## [1,] 51.89367 27.82312
## [2,] 51.47399 30.34178
```

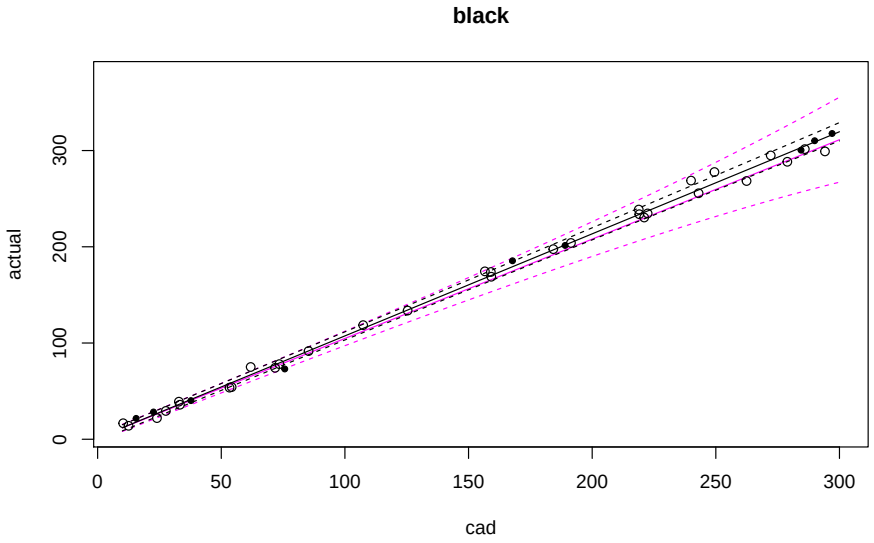
Briefly discuss the meaning of the results [+2 marks here, and +2 marks for Q1]
Note: In some cases, the randomness of the data splitting gives an unusually small number of observations for some colours for large CAD values, which leads to scores not necessarily being strictly lower for the full model. If done as in the code here (where the random seed is set in `CWAcode.R`) should in theory generate the same split on all systems in every run, but there have been indications that this is not the case (possibly due to differences in how the splitting is done).

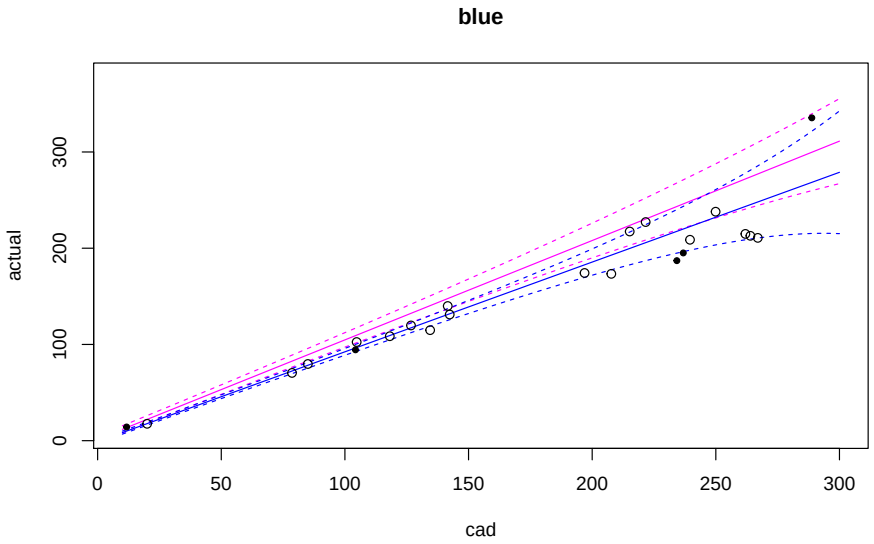
The full, colour aware model consitently gives lower (better) scores than the simpler, colour blind model. The difference is largest when scoring the predictions for the estimation/observation data. This is to be expected, since the full model is more flexible, and the maximum likelihood estimator tries to find optimal parameters for that particular set of observations. As seen in the figures, the colour blind model doesn't adapt as well to the data as the full model, neither in terms of the point predictions (assessed by the all the scores, but exclusively by the SE score) nor the uncertainties (assessed by the DS and Interval scores).

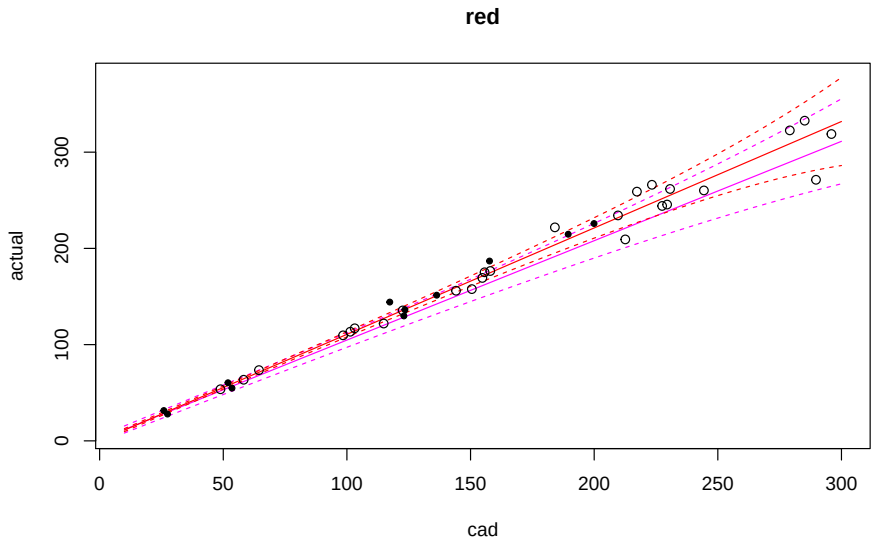
Plotting was not required, but this entire block of code, except for the `model_predict` calls, was included in the hints document as a helpful tool for the discussion.

```
## Prediction intervals:
data_plot <- data.frame(cad = rep(seq(10, 300, length=100), times=3),
                        colour=rep(c("black", "blue", "red"), each=100))
model_Z <- model_Z_lab
ppred_lab <- model_predict(theta_hat_lab, data_plot, Sigma_theta = Sigma_theta_lab,
                           type = "observation", alpha=1/3)
model_Z <- model_Z_cw
ppred_cw <- model_predict(theta_hat_cw, data_plot, Sigma_theta = Sigma_theta_cw,
                          type = "observation", alpha=1/3)
for (colour in c("black", "blue", "red")) {
  data_plot_sub <- data_plot[data_plot$colour == colour, ]
  ppred_lab_sub <- ppred_lab[data_plot$colour == colour, ]
  ppred_cw_sub <- ppred_cw[data_plot$colour == colour, ]
  plot(data_plot_sub$cad, ppred_lab_sub$mu, type = "l", col = "magenta",
       ylim = range(ppred_lab[, c("lwr", "upr")],
                    ppred_cw[, c("lwr", "upr")]),
       xlab = "cad", ylab = "actual",
       main = colour)
  lines(data_plot_sub$cad, ppred_lab_sub$lwr, lty = 2, col = "magenta")
  lines(data_plot_sub$cad, ppred_lab_sub$upr, lty = 2, col = "magenta")
  lines(data_plot_sub$cad, ppred_cw_sub$mu, lty = 1, col = colour)
  lines(data_plot_sub$cad, ppred_cw_sub$lwr, lty = 2, col = colour)
  lines(data_plot_sub$cad, ppred_cw_sub$upr, lty = 2, col = colour)
  points(actual ~ cad, data = data_obs[data_obs$colour==colour, ],
         pch = 1, col = "black")
  points(actual ~ cad, data = data_test[data_test$colour==colour, ],
         pch = 20, col = "black")
```

```
}
```


black
actual
cad
0
100
200
300
0
50
100
150
200
250
300


blue
actual
cad
0
100
200
300
0
50
100
150
200
250
300



Question 4

The self-chosen function needs to have non-zero derivatives up to order 4 so that the bounds near the self-chosen θ are non-zero. The bounds need to fulfil $L_0 \geq |f(\cdot)|$, $L_k \geq |f^{(k)}(\cdot)|$, $k = 1, 2, 3, 4$, in the vicinity of θ to work in the expressions for optimal steps h. In theory, the bounds need to hold between θ and $\theta + h$, but since h is small, it's allowed to use the actual values at θ itself. Using the simplistic $h_1 = \epsilon_0^{1/2}$ approximations should not give full marks. Using $L_k \equiv 1$ as bound approximation should not give full marks (unless those bounds actually hold for the chosen function).

Example solution:

Find true derivatives and bounds [+3 marks]:

$$f(\theta) = \exp(2\theta),$$

$$f'(\theta) = 2\exp(2\theta),$$

$$f''(\theta) = 4\exp(2\theta),$$

$$f^{(3)}(\theta) = 8\exp(2\theta),$$

$$f^{(4)}(\theta) = 16\exp(2\theta)$$

```
theta <- 3
f0 <- function(theta) exp(2 * theta)
f1 <- function(theta) 2 * exp(2 * theta)
f2 <- function(theta) 4 * exp(2 * theta)
```

```r
f3 <- function(theta) 8 * exp(2 * theta) # Not used or needed.
f4 <- function(theta) 16 * exp(2 * theta) # Only used for the L4 bound.
```

```r
L0 <- abs(f0(theta))
L1 <- abs(f1(theta))
L2 <- abs(f2(theta))
L3 <- abs(f3(theta)) # Not used or needed.
L4 <- abs(f4(theta))
```

Find optimal steps [+2 marks]

```r
e0 <- .Machine$double.eps
# Optimal step for asymmetric first order difference
h1 <- (2 * e0 * (2 * L0 + abs(theta) * L1) / L2) ^ 0.5
# Optimal step for second order difference
h2 <- (24 * e0 * (2 * L0 + abs(theta) * L1) / L4) ^ 0.25
```

Calculate finite differences [+2 marks]

```r
# Theoretical error bounds (NOT required)
B1 <- e0 * (2 * L0 + abs(theta) * L1) / h1 + h1 * L2 / 2
B2 <- e0 * (4 * L0 + 2 * abs(theta) * L1) / h2 ^ 2 + h2 ^ 2 * L4 / 12
# Calculate finite differences
df1 <- (f0(theta + h1) - f0(theta)) / h1
df2 <- (f0(theta + h2) - 2 * f0(theta) + f0(theta - h2)) / h2 ^ 2
df2suboptimal <- (f0(theta + h1) - 2 * f0(theta) + f0(theta - h1)) / h1 ^ 2
```

Calculate errors and compare [+3 marks]

```r
# First order derivative error
c(B1, abs(df1 - f1(theta)))

## [1] 4.809246e-05 2.371704e-05

# Second order derivative error
c(B2, abs(df2 - f2(theta)), abs(df2suboptimal - f2(theta)))

## [1] 5.553239e-05 2.887523e-05 1.371517e+01
```

The error in the second order finite difference is many orders of magnitude smaller when using the(approximately) optimal h-values than when using the h-value that would have been optimal for a first order difference.