HW6 - Lucas Fellmeth, Sven Bergmann

2023-11-29

1

The goal of this problem is to estimate the regression function of acceleration vs time for the `mcycle` data in the package `MASS`, using cubic smoothing splines.

```
library(MASS)
library(splines)
library(splines2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

A

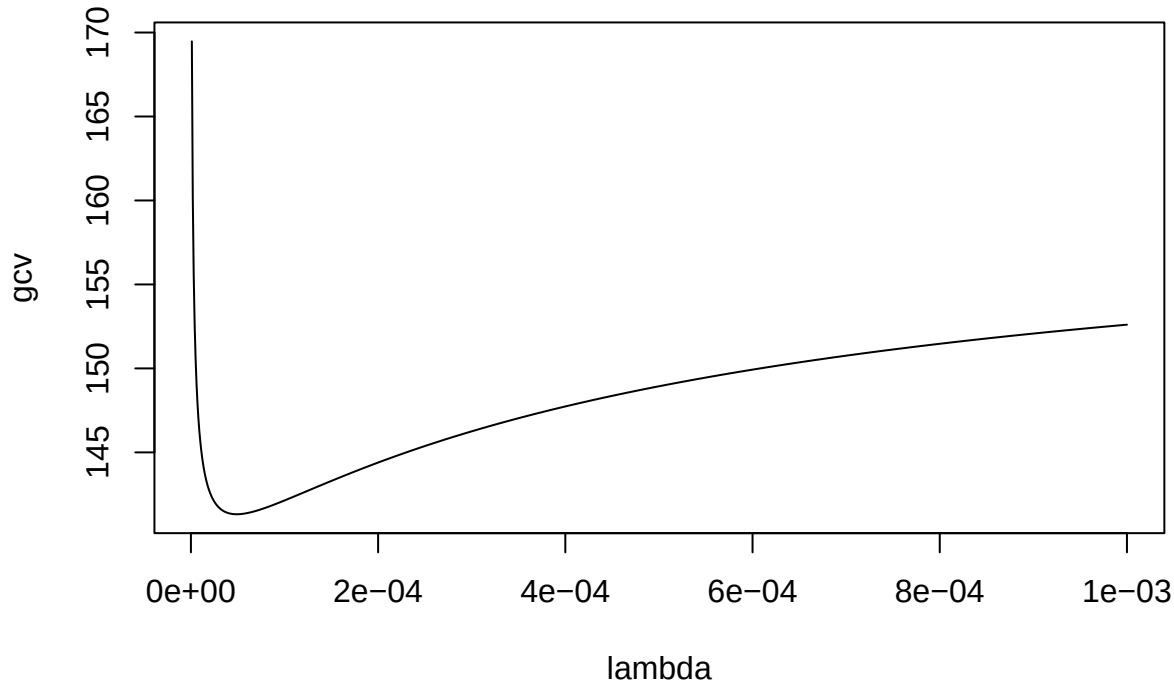
For a reasonable range of smoothing parameters λ, compute and plot the generalized cross validation measure $GCV(\lambda)$ and find the optimal smoothing parameter.

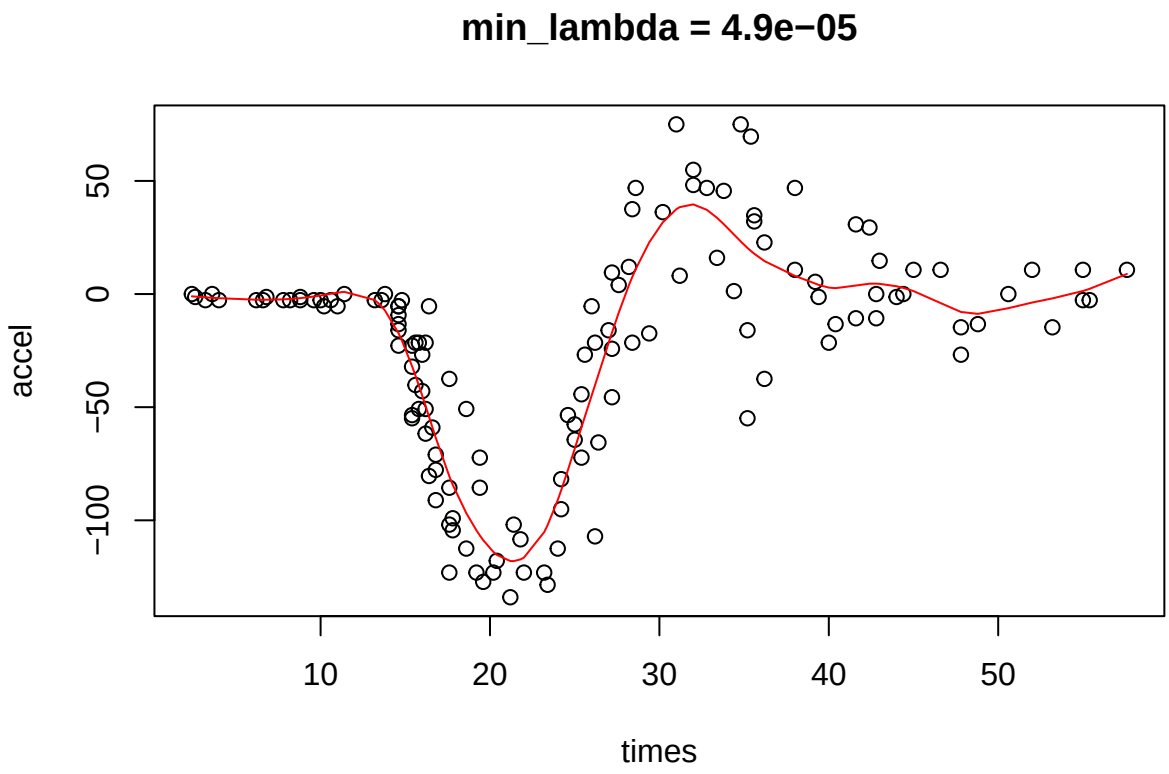
```
lambda_gcv_df <- function(data) {
    with(data, {
        lambda <- seq(10^(-6), 10^(-3), by = 10^(-6))

        smoothing <- function(l, x, y) {
            return(smooth.spline(x, y, lambda = l)$cv.crit)
        }

        gcv <- sapply(lambda, smoothing, x = accel, y = times)
        return(data.frame(lambda = lambda, gcv = gcv))
    })
}
```

```r
data(mcycle)
with(mcycle, {
    lambda_gcv_df_a <- lambda_gcv_df(mcycle)
    lambda_opt <- lambda_gcv_df_a[which.min(lambda_gcv_df_a$gcv), ]$lambda
    plot(lambda_gcv_df_a, type = "l")
    plot.new()
    plot(times, accel, main = paste("min_lambda =", lambda_opt))
    lines(smooth.spline(x = times, y = accel, lambda = lambda_opt), col = "red",
        type = "l")
})
```





B

Using the Demmler-Reinsch basis representation, how many basis functions contribute significantly to the fit, as determined, for instance, by accumulating 90% of the total degrees of freedom?

First, we averaged over the accel values when there were the same times values. We did that because otherwise we could not perform Cholesky decomposition because of a singularity issue. When we tried doing SVD we had issues with imaginary values.

```
with(mcycle, {
    df <- data.frame(times = times, accel = accel)
    df$times <- as.character(df$times)
    df_grouped_times <- df %>%
        group_by(times) %>%
        summarise(accel = mean(accel))
    df_grouped_times$times <- as.numeric(df_grouped_times$times)
    df_grouped_times <<- df_grouped_times[order(df_grouped_times$times), ]
})
```

Now we went on computing the Demmler-Reinsch basis representation.

```
with(df_grouped_times, {
    # cut two values at the ends for the knots
    knots <- times[3:92]
    upper <- ceiling(max(times))
```

```
    Phi <- bSpline(times, knots = knots, degree = 3, intercept = T, Boundary.knots = c(0,
        upper))
    R <- chol(t(Phi) %*% Phi)
    t <- seq(0, upper, length.out = 200)
    delta <- t[2] - t[1]
    Phi2 <- bSpline(t, knots = knots, degree = 3, intercept = T, Boundary.knots = c(0,
        upper), derivs = 2)
    Omega <- delta * t(Phi2) %*% Phi2
    Phi0 <- bSpline(t, knots = knots, degree = 3, intercept = T, Boundary.knots = c(0,
        upper))
    Sigma <- solve(t(R)) %*% Omega %*% solve(R)
    S <<- eigen(Sigma)$values
    U <- eigen(Sigma)$vectors
    Psi <- Phi %*% solve(R) %*% U
    Psi0 <- Phi0 %*% solve(R) %*% U
})
```

We calculate lambda optimal again because we averaged some numbers in the dataset as described above.

```
lambda_gcv_df_a <- lambda_gcv_df(df_grouped_times)
lambda_opt <- lambda_gcv_df_a[which.min(lambda_gcv_df_a$gcv), ]$lambda
cat("lambda_opt = ", lambda_opt)
```

```
## lambda_opt =  5.2e-05
```

In the following, we tried to compute the basis functions with lambda_opt which we calculated above but that gave strange values and a strange plot. So we decided to use 10^{-1} for lambda because then we are getting more reasonable values.

```
x_lambda_opt <- rev(1/(1 + lambda_opt * S))
degrees_of_freedom_lambda_opt <- sum(x_lambda_opt)
cat("degrees of freedom with lambda =", lambda_opt, ":", degrees_of_freedom_lambda_opt)
```

```
## degrees of freedom with lambda = 5.2e-05 : 91.67041
```

```
lmd <- 10^(-1)
x <- rev(1/(1 + lmd * S))
degrees_of_freedom <- sum(x)
cat("degrees of freedom with lambda =", lmd, ":", degrees_of_freedom)
```

```
## degrees of freedom with lambda = 0.1 : 37.97324
```

```
idx_lambda_opt <- length(which(cumsum(x_lambda_opt) < 0.9 * degrees_of_freedom_lambda_opt))
cat("We need", idx_lambda_opt, "basis functions for lambda =", lambda_opt)
```

```
## We need 82 basis functions for lambda = 5.2e-05
```

```
cat("Now we have", sum(x[1:idx_lambda_opt]), "degrees of freedom for lambda =", lambda_opt)
```

```
## Now we have 37.888 degrees of freedom for lambda = 5.2e-05
```

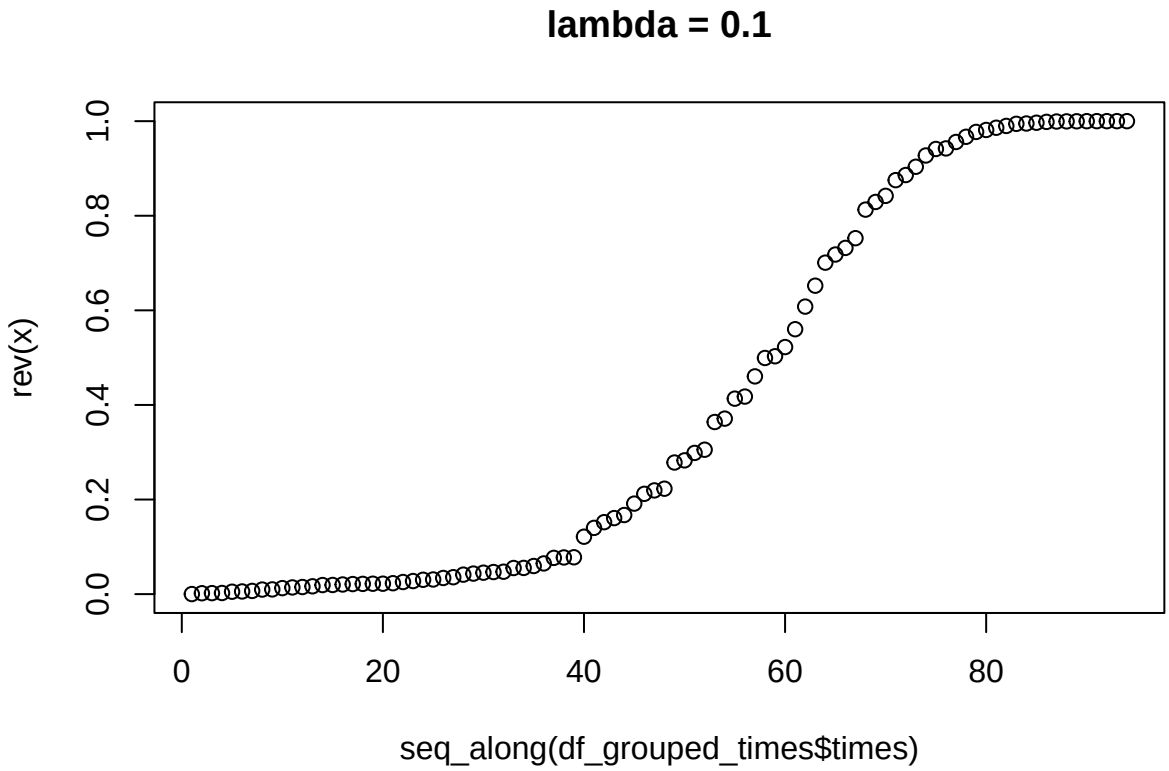
```r
idx <- length(which(cumsum(x) < 0.9 * degrees_of_freedom))
cat("We need", idx, "basis functions for lambda = ", lmd)
```

```
## We need 42 basis functions for lambda =  0.1
```

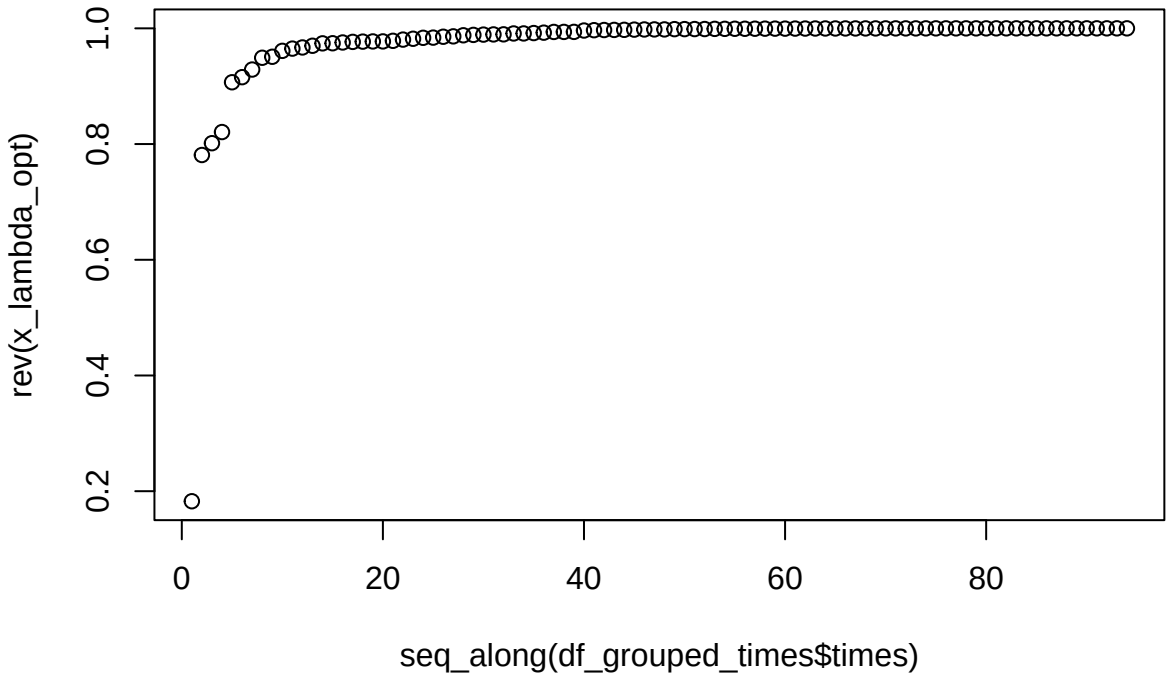
```r
cat("Now we have", sum(x[1:idx]), "degrees of freedom for lambda = ", lmd)
```

```
## Now we have 34.07568 degrees of freedom for lambda =  0.1
```

```r
plot(seq_along(df_grouped_times$times), rev(x), main = paste("lambda =", lmd))
```




```r
plot(seq_along(df_grouped_times$times), rev(x_lambda_opt), main = paste("lambda =",
    lambda_opt))
```


lambda = 5.2e−05
rev(x_lambda_opt)
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
seq_along(df_grouped_times$times)