BIOS 7720
Homework 1 Solutions

For all problems which involve simulation or fitting models to data, code to reproduce results must be included in your solutions. In addition, for problems 1-5, you may use *R* packages to help set up basis functions (e.g. *mgcv*), but estimation and smoothing parameter selection should be implemented manually. For these problems you may use either ordinary cross validation or generalized cross validation. For problems 7-8, students may use the *mgcv* package for model fitting and smoothing parameter selection.

Students are allowed to work together in groups of up to 3 on this assignment, though all students must submit their own independent write-ups. For coding problems, students may work together on designing and strategizing methods for implementation, but coding up of solutions must be done independently. Please include the name of any other students you worked with on this assignments in your submission.

Here I load all packages which are used in answering the homework questions. Then, I set the directories which contain code to be sources (R scripts) to answer specific homework questions. In addition, I set the path where I save results from R scripts which involve writing functions (problem 2) or have simulations/longer computation times (problems 5, 7, 8).

```r
## load packages
library("here")
library("ggplot2")
library("gridExtra")
library("tidyverse")
library("mgcv")
## set paths
code_path    <- here("assignments","HW1","solutions","code")
results_path <- here("assignments","HW1","solutions","results")
figure_path  <- here("assignments","HW1","solutions","figures")
```

P1. (10 points) Suppose we observe data of the form $\{(x_i, y_i); i = 1, \ldots, N\}$ and that we are interested in fitting the following model:

$$y_i = f(x_i) + \epsilon_i$$
$$\epsilon_i \sim N(0, \sigma_\epsilon^2)$$

where we model $f(x)$ using some basis expansion $f(x_i) = \sum_{k=1}^{K} \xi_k \phi_k(x)$. For the following problems, consider the penalized least squares criteria given by

$$\text{PENSSE}_\lambda = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

a. (2 points) Show that under this framework, $\lambda \int f''(x)^2 dx = \boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}$ where $\boldsymbol{S}$ is a $K \times K$ matrix and $\boldsymbol{\xi} = [\xi_1, \dots, \xi_k]^t$. State any necessary assumptions.
Write $f(x) = \sum_{k=1}^K \xi_k \phi_k(x)$. Assume that $\frac{d^2}{dx^2}\phi_k(x) = \phi_k''(x)$ exists over the range of potential x values for $k = 1, \dots, K$. Then $f''(x) = \sum_{k=1}^K \xi_k \phi_k''(x)$. Write $\boldsymbol{d}(x) = [\phi_1''(x) \dots \phi_k''(x)]^t$. Then

$$\begin{aligned}\int f''(x)^2 dx &= \int f''(x) f''(x) dx \\ &= \int \boldsymbol{\xi}^t \boldsymbol{d}(x) \boldsymbol{d}(x)^t \boldsymbol{\xi} dx \\ &= \boldsymbol{\xi}^t \left[\int \boldsymbol{d}(x) \boldsymbol{d}(x)^t dx \right] \boldsymbol{\xi} \\ &= \boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}\end{aligned}$$

where $\boldsymbol{S} = \boldsymbol{d}(x)\boldsymbol{d}(x)^t$ is the $K \times K$ matrix with the i^{th} row and j^{th} column equal to $\int \frac{d^2}{dx^2}\phi_i(x) \frac{d^2}{dx^2}\phi_j(x) dx$.

b. (1 points) Find $\boldsymbol{S}$ using the basis expansion $f(x) = \sum_{k=0}^3 \xi_k x^k$ where integration is over the domain $x \in (0, 10)$.
Here $\phi_k(x) = x^k$ so

$$\frac{d^2}{dx^2}\phi_k(x) = \begin{cases} k(k-1)x^{k-2} & k \geq 2 \\ 0 & k = 0, 1 \end{cases}$$

Then the i^{th} row and j^{th} column of $\boldsymbol{S}$ is defined by

$$\int_0^{10} \frac{d^2}{dx^2}\phi_i(x) \frac{d^2}{dx^2}\phi_j(x) dx = \begin{cases} \frac{ij(i-1)(j-1)}{i+j-3} x^{i+j-3} |_{x=0}^{x=10} & i, j \geq 2 \\ 0 & i \text{ or } j \in \{0, 1\} \end{cases}$$

It follows that

$$\boldsymbol{S}_{4\times 4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 600 \\ 0 & 0 & 600 & 12000 \end{bmatrix}$$

Code to create $\boldsymbol{S}$ is presented below.

```
k <- 0:3
## create a function for calcuating the S matrix for us
fn_S <- function(i, j, lb, ub) {
  if(any(c(i,j) %in% c(0,1))) return(0)
  i*j*(j-1)*(i-1)/(i+j-3) * (ub^(i+j-3) - lb^(i+j-3))
}
```

```r
## vectorize this function using the base::Vectorize() function
Vfn_S <- Vectorize(fn_S)
## then create the S matrix using the outer() function
## note that we vectorize fn_S because outer specifically requires
## a vectorized function
S <- outer(k, k, Vfn_S, lb=0, ub=10)
S

##      [,1] [,2] [,3]  [,4]
## [1,]    0    0    0     0
## [2,]    0    0    0     0
## [3,]    0    0   40   600
## [4,]    0    0  600 12000
```

c. (2 points) Show that the solution to the penalized least squares problem is $\hat{\boldsymbol{\xi}} = (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t\boldsymbol{y}$ where $\boldsymbol{\Phi}$ is the $N \times K$ spline basis matrix associated with the observed covariate data, and $\boldsymbol{y} = [y_1, \dots, y_N]^t$.

Here we find to find the $\hat{\boldsymbol{\xi}}$ which minimizes the penalized least squares problem for a fixed λ.

$$\begin{aligned}\text{PENSSE}_\lambda &= (\boldsymbol{y} - \boldsymbol{\Phi\xi})^t(\boldsymbol{y} - \boldsymbol{\Phi\xi}) + \lambda\boldsymbol{\xi}^t\boldsymbol{S\xi} \\ &= \boldsymbol{y}^t\boldsymbol{y} - 2\boldsymbol{\xi}^t\boldsymbol{\Phi}^t\boldsymbol{y} + \boldsymbol{\xi}^t\boldsymbol{\Phi}^t\boldsymbol{\Phi\xi} + \lambda\boldsymbol{\xi}^t\boldsymbol{S\xi}\end{aligned}$$

Take the derivative with respect to $\boldsymbol{\xi}$, set equal to 0, solve

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\xi}}\text{PENSSE}_\lambda &= -2\boldsymbol{\Phi}^t\boldsymbol{y} + 2\boldsymbol{\Phi}^t\boldsymbol{\Phi\xi} + 2\lambda\boldsymbol{S\xi} \overset{\text{set}}{=} 0 \\ (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})\boldsymbol{\xi} &= \boldsymbol{\Phi}^t\boldsymbol{y} \\ \hat{\boldsymbol{\xi}} &= (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t\boldsymbol{y}\end{aligned}$$

Check the second derivative

$$\frac{\partial^2}{\partial\boldsymbol{\xi}^2}\text{PENSSE}_\lambda = 2\boldsymbol{\Phi}^t\boldsymbol{\Phi} + 2\lambda\boldsymbol{S}$$

Since $\boldsymbol{\Phi}^t\boldsymbol{\Phi}$ is positive definite, $\boldsymbol{S}$ is positive semi-definite, and $\lambda \geq 0$, it follows that $2\boldsymbol{\Phi}^t\boldsymbol{\Phi} + 2\lambda\boldsymbol{S}$ is positive definite. Thus $\hat{\boldsymbol{\xi}}$ is a minimum.

d. (2 points) Given the least squares solution $\hat{\boldsymbol{\xi}} = (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t\boldsymbol{y}$, find $\text{Var}(\hat{f}(x))$ for a fixed λ.

$$
\begin{aligned}
\mathrm{Var}(\hat{f}(x)) &= \mathrm{Var}(\phi(x)^t \hat{\boldsymbol{\xi}}) \\
&= \mathrm{Var}(\phi(x)^t \hat{\boldsymbol{\xi}}) \\
&= \mathrm{Var}(\phi(x)^t (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\boldsymbol{\Phi}^t \boldsymbol{y}) \\
&= \phi(x)^t (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\boldsymbol{\Phi}^t \mathrm{Var}(\boldsymbol{y})\boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\phi(x) \\
&\text{Since } (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1} \text{ is symmetric} \\
&= \sigma_\epsilon^2 \phi(x)^t (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\boldsymbol{\Phi}^t \boldsymbol{I}_{N\times N}\boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\phi(x) \\
&\text{Since } y|x \overset{\text{iid}}{\sim} N(f(x), \sigma_\epsilon^2) \\
&= \sigma_\epsilon^2 \phi(x)^t (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\boldsymbol{\Phi}^t \boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda \boldsymbol{S})^{-1}\phi(x)
\end{aligned}
$$

e. (2 points) In lecture we discussed looping over candidate smoothing parameters λ and choosing the optimal λ based on leave-one-out cross-validated prediction error. In practice, the procedure shown in the lecture notes can be quite slow for large K (relative to N).

We can potentially speed this procedure up and increase numerical stability up by creating an augmented design matrix ($\boldsymbol{\Phi}^*$) and response vector ($\boldsymbol{y}^*$):

$$
\boldsymbol{y}^* = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{a} \end{bmatrix}, \qquad \boldsymbol{\Phi}^* = \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{B} \end{bmatrix}
$$

and using these augmented data to obtain $\hat{\boldsymbol{\xi}}$ via linear regression (using, e.g. .lm.fit). This allows us to take advantage of methods for estimating linear regression via orthogonal decomposition. Find $\boldsymbol{a}$ and $\boldsymbol{B}$. Hint: note that $\boldsymbol{S}$ can be written as $\boldsymbol{D}^t\boldsymbol{D}$.

Note that because $\boldsymbol{S}$ is positive semi-definite, we an find $\boldsymbol{D}$ using the cholesky decomposition of $\boldsymbol{S}$

$$
\begin{aligned}
\mathrm{PENSSE}_\lambda &= \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx \\
&= (\boldsymbol{y} - \boldsymbol{\Phi\xi})^t(\boldsymbol{y} - \boldsymbol{\Phi\xi}) + \lambda \boldsymbol{\xi}^t \boldsymbol{S}\boldsymbol{\xi} \\
&= (\boldsymbol{y} - \boldsymbol{\Phi\xi})^t(\boldsymbol{y} - \boldsymbol{\Phi\xi}) + \lambda \boldsymbol{\xi}^t \boldsymbol{D}^t\boldsymbol{D}\boldsymbol{\xi} \\
&= (\boldsymbol{y} - \boldsymbol{\Phi\xi})^t(\boldsymbol{y} - \boldsymbol{\Phi\xi}) + (\boldsymbol{0}_{K\times 1} - \sqrt{\lambda}\boldsymbol{D\xi})^t(\boldsymbol{0}_{K\times 1} - \sqrt{\lambda}\boldsymbol{D\xi}) \\
&= (\boldsymbol{y}^* - \boldsymbol{\Phi}^*\boldsymbol{\xi})^t(\boldsymbol{y}^* - \boldsymbol{\Phi}^*\boldsymbol{\xi})
\end{aligned}
$$

where

$$
\boldsymbol{y}^* = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0}_{K\times 1} \end{bmatrix}, \qquad \boldsymbol{\Phi}^* = \begin{bmatrix} \boldsymbol{\Phi} \\ \sqrt{\lambda}\boldsymbol{D} \end{bmatrix}
$$

f. (1 point) Why don't we optimize the least squares criteria jointly for $\boldsymbol{\xi}$ and λ?

Optimizing the penalized least squares criteria for $\boldsymbol{\xi}$ and λ jointly will always select $\lambda = 0$ since $\int f''(x)^2 dx \geq 0$ for all possible $f(\cdot)$, resulting in an unpenalized fit.

P2. (10 points) Write a function which takes as inputs a response vector $\boldsymbol{y}$, a spline basis matrix $\boldsymbol{\Phi}$, a penalty matrix $\boldsymbol{S}$, and a candidate smoothing parameter λ (under the notation from problem 1 above) and obtains both point and variance estimates for $\boldsymbol{\xi}$ (conditional on the smoothing parameter). Using the data simulated by the code below, apply your function to find the optimal smoothing parameter using cross-validation as your smoothing parameter selection criteria.

```
set.seed(5520)
N <- 100
f <- function(x) cos(2*pi*x)
x <- runif(N, min=-1, max=1)
y <- f(x) + rnorm(N, mean=0, sd=0.5)
```

You may use the code from the lecture 2 notes (*mgcv::smoothCon()*) to set up your basis matrix $\boldsymbol{\Phi}$ and obtain $\boldsymbol{S}$ using either B-splines or cubic regression splines.

Here we do smoothing parameter selection using the GCV criteria presented in lecture:

$$\begin{aligned}\mathcal{V}_g &= N\sum_{i=1}^{N}(y_i - \hat{y}_i)^2/(N - \text{trace}(H))^2 \\ H &= \boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t\end{aligned}$$

where $\boldsymbol{\Phi}$ is our design matrix containing our spline basis functions. Note that when N is large, calculating H directly can be costly. For this problem N is small, but later problems require re-calculating this quantity many times or calculating it for large N. Of course, you can directly use the code from class, I'll just be presenting a more efficient approach here.

Since we only need the trace of H, we can simplify the GCV calculation by using the fact that the trace is invariant to cyclic permutations. Specifically:

$$\begin{aligned}\text{trace}(H) &= \text{trace}(\boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t) \\ &= \text{trace}(\boldsymbol{\Phi}^t\boldsymbol{\Phi}(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \lambda\boldsymbol{S})^{-1})\end{aligned}$$

In addition to calculating H, we need predicted values $\hat{y}$. To obtain fast and stable estimates for $\hat{y}$, we can use the results from 1e which allow us to reformulate the penalized least squares problem as standard linear regression which can be estimated using fast, numerically stable QR decomposition of the design matrix.

Combining our results from from 1c and 1e, we know that we obtain our estimate $\hat{\boldsymbol{\xi}}$ as the solution to the set of equations

$$\boldsymbol{\Phi}^{*,t}\boldsymbol{\Phi}^{*,t}\boldsymbol{\xi} = \boldsymbol{\phi}^{*,t}\boldsymbol{y}$$

I wrote up two approaches for doing smoothing parameter selection this way, both of which are contained in "HW1_P2.R".

(a) Grid search (as shown in lecture): *fn_PENSSE_grid()*

(b) Direct numeric optimization: *fn_PENSSE_opt()*

The code is sourced below. I compare computation times to *mgcv::gam*. Note: even this approach is not as fast as it could be! I'll try to add a third option which actually does the correct "numerically stable" approach to this document at a later date.

```
## set up the bases for model fitting
## and extract relevant quantities Phi and S
K   <- 50
smX <- smoothCon(s(x, bs="cr", k=K), data=data.frame(x=x))
Phi <- smX[[1]]$X
S   <- smX[[1]]$S[[1]]

## source the code for problem 2
## containing the grid search and optimization
## functions for GCV
source(here(code_path, "HW1_P2.R"))

start_time_p2_grid <- Sys.time()
ests_grid <- fn_PENSSE_grid(y=y, Phi=Phi, S=S, loglambda=seq(-3,20,len=1000))
end_time_p2_grid <- Sys.time()
difftime(end_time_p2_grid, start_time_p2_grid, units="mins")


## Time difference of 0.009092283 mins


start_time_p2_opt <- Sys.time()
ests_opt <- fn_PENSSE_opt(y=y, Phi=Phi, S=S)
end_time_p2_opt <- Sys.time()
difftime(end_time_p2_opt, start_time_p2_opt, units="mins")


## Time difference of 0.0007016182 mins


start_time_p2_gam <- Sys.time()
ests_gam <- gam(y ~ s(x, bs="cr",k=50))
end_time_p2_gam <- Sys.time()
difftime(end_time_p2_gam, start_time_p2_gam, units="mins")


## Time difference of 0.000412631 mins
```

We obtain similar smoothing parameters using either approach, though the optimization approach is closer to that produced by *mgcv::gam*. This is unsurprising as a grid search is generally not expected to find the "exact" minimum/maximum.

```
ests_grid$sp

## [1] 1934.429

ests_opt$sp

## [1] 1943.2

ests_gam$sp

##     s(x)
## 1943.202
```

P3. (10 points) Thus far we have discussed estimating spline coefficients using penalized least squares. However, because of our distributional assumptions on the residual term ($\epsilon_i \sim N(0, \sigma^2_\epsilon)$), for a fixed smoothing parameter λ, we can obtain point and uncertainty estimates on spline coefficients using maximum likelihood. Using an optimization routine (e.g. the *optim()* function in *R*):

a. (5 points) Obtain the maximum likelihood estimate for $\hat{\boldsymbol{\xi}}$ and $\text{var}(\hat{\boldsymbol{\xi}})$ for the data simulated in problem 2 above.

Note: This question did not ask you to derive the MLE for $\boldsymbol{\xi}$ and specified using the smoothing parameter obtained from the previous problem. That's fine and will result in full credit. However, we can observe something interesting if we solve for $\hat{\boldsymbol{\xi}}$.

Let $\boldsymbol{\theta} = \{\boldsymbol{\xi}, \sigma_\epsilon\}$ and denote the penalized log likelihood as $\text{pl}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}, \lambda)$. Then we have

$$\text{pl}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}, \lambda) = -0.5\left[N\log(2\pi\sigma^2_\epsilon) + (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi})^t(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\xi})/\sigma^2_\epsilon + \lambda\boldsymbol{\xi}^t\boldsymbol{S}\boldsymbol{\xi}\right]$$

We can find the MLE $\hat{\boldsymbol{\xi}}$ by following the same procedure as with Problem 1c. That is, take the first derivative, set equal to zero, solve, check the second derivative.

$$\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\xi}}\text{pl}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}, \lambda) &= -2\boldsymbol{\Phi}^t\boldsymbol{y}/\sigma^2_\epsilon + 2\boldsymbol{\Phi}^t\boldsymbol{\Phi}\boldsymbol{\xi}/\sigma^2_\epsilon + 2\lambda\boldsymbol{S}\boldsymbol{\xi} \overset{\text{set}}{=} 0 \\
(\boldsymbol{\Phi}^t\boldsymbol{\Phi}/\sigma^2_\epsilon + \lambda\boldsymbol{S})\boldsymbol{\xi} &= \boldsymbol{\Phi}^t\boldsymbol{y}/\sigma^2_\epsilon \\
(\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \sigma^2_\epsilon\lambda\boldsymbol{S})\boldsymbol{\xi} &= \boldsymbol{\Phi}^t\boldsymbol{y} \\
\hat{\boldsymbol{\xi}} &= (\boldsymbol{\Phi}^t\boldsymbol{\Phi} + \sigma^2_\epsilon\lambda\boldsymbol{S})^{-1}\boldsymbol{\Phi}^t\boldsymbol{y}
\end{aligned}$$

Below I provide code for obtaining the maximum likelihood estimates and asymptotic variances for our spline coefficients by minimizing the negative penalized log-likelihood of our data (maximizing the log-likelihood would be equally valid).

Note that the asymptotic variance estimates from maximum likelihood estimation are the negative second derivative of the log likelihood with respect to the model parameters $\boldsymbol{\xi}, \sigma_\epsilon$ evaluated at their maximal values. Let $\boldsymbol{\theta} = \{\boldsymbol{\xi}, \sigma_\epsilon\}$:

$$I_0(\boldsymbol{\theta}) = -\frac{\partial^2}{\partial \boldsymbol{\theta}} l(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{x})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$

where $l(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{x})$ is the log-likelihood. Then recall from your stat theory class that the asymptotic variance of $\hat{\boldsymbol{\theta}}$ is $\text{Var}(\hat{\boldsymbol{\theta}}) = I_0(\boldsymbol{\theta})^{-1}$.

```r
## construct the spline basis
K <- 50
sm <- smoothCon(s(x, bs="cr", k=K), data=data.frame(x=x))
Phi <- sm[[1]]$X
## function for MINIMIZING the penalized normal log-likelihood
norm_l <- function(params, Phi, y, S, lambda) {
    ## get spline coefficients and sd from params vector
    xi    <- params[-1]
    sigma <- params[1]
    ## get the negative penalized log-likelihood
    -sum(dnorm(y, mean=Phi %*% xi, sd=sigma, log=TRUE)) +
      0.5*lambda * t(xi) %*% S %*% xi / sigma^2
}

## get some initial estimates for "good" starting values
init_fit <- lm(y ~ Phi-1)
inits    <- c(mean(init_fit$residuals^2), coef(init_fit))

## fit using L-BFGS-B optimization method -- this allows us
## to place bounds on sigma (BFGS would work fine, but we'd get some error messages
## when the optimizer tries negative values for \sigma)
ests_fixed_lambda  <-
  optim(par=inits, fn=norm_l, y=y, Phi=Phi, S=sm[[1]]$S[[1]], lambda=ests_opt$sp,
        method="L-BFGS-B",control=list(maxit=10000),
        lower=c(0, rep(-Inf,K)), upper=rep(Inf,K=1), hessian=TRUE)
## get the MLE for \xi (note that the first element in my code is the \sigma_\epsilon, so we remove this
## from the returned parameter vector)
xi_mle      <- ests_fixed_lambda$par[-1]
## get the asyymptotic variance of \hat{\xi} as the inverse of the Hessian
## again, we remove the first row and first column
## because we minimized the negative log likelihood, we just inver the Hessian
## if we had maximized the log likelihood, we would invert the negative Hessian
var_xi_mle  <- solve(ests_fixed_lambda$hessian)[-1,-1]
```

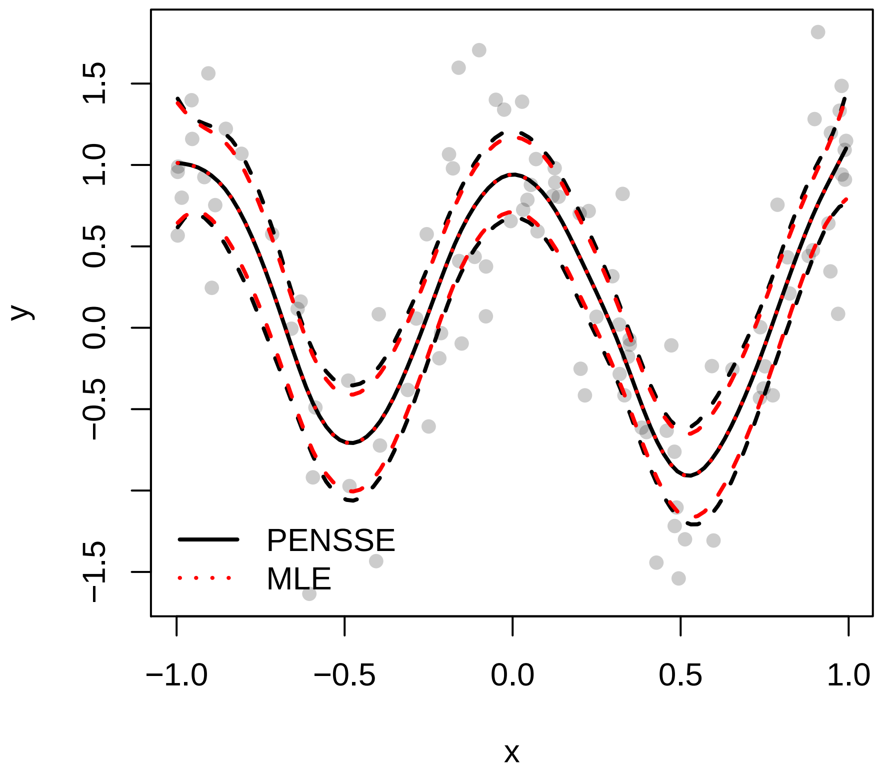
b. (5 points) Use your results for (a) to compare the estimated $\hat{f}(x)$ and $\hat{f}(x) \pm 2\text{SE}(\hat{f}(x))$ for $x \in (-1, 1)$ obtained using maximum likelihood to those from problem 2 (penalized least squares) at the optimal value of λ you found in problem 2.

Hint(s): 1) by default the *optim()* function performs minimization (as opposed to maximization); 2) optimize on the log-likelihood scale; 3) using the *optim()* function, set the argument hessian = TRUE to obtain the Hessian matrix.

Below I show how to obtain point and pointwise 95% confidence intervals for $\hat{f}(x)$ for a range of x values between -1 and 1. Note that I obtain variance estimates for the MLE fit using the fact that the Hessian matrix returned by *optim()* is the second derivative of the objective function with respect to the parameters evaluated at the final values.

We see that the point estimates (PENSSE = solid black line, MLE = short dashed red line) are nearly identical. This is because I divided the penalty term by σ^2_ϵ to make the MLE solution equivalent to the PENSSE solution. The 95% CIs (long dashed lines) are shorter for the MLE fit than the PENSSE.

```r
## set up a new grid to obtain predictions on
xind        <- seq(min(x), max(x), len=100)
## get the corresponding basis matrix
Phi_pred    <- PredictMat(sm[[1]], data=data.frame(x=xind))
## get \hat{f}(x) and SE(\hat{f}(x)) at the new values
fhat_mle    <- Phi_pred %*% xi_mle
se_fhat_mle <- sqrt(diag(Phi_pred %*% var_xi_mle %*% t(Phi_pred)))

## get variance and point estimates from the PENSSE fit
fhat_PENSSE    <- Phi_pred %*% ests_opt$xi
se_fhat_PENSSE <- sqrt(diag(Phi_pred %*% ests_opt$var_xi %*% t(Phi_pred)))

## do the plotting
par(mfrow=c(1,1))
plot(x,y, pch=16, col=rgb(0,0,0,0.2), xlab='x',ylab='y')
matplot(xind, cbind(fhat_mle, fhat_mle + 1.96*se_fhat_mle, fhat_mle - 1.96*se_fhat_mle),
        col="black", lty=c(1,2,2),type='l',add=TRUE,lwd=2)
matplot(xind, cbind(fhat_PENSSE, fhat_PENSSE + 1.96*se_fhat_PENSSE,
                    fhat_PENSSE - 1.96*se_fhat_PENSSE),
        col="red", lty=c(3,2,2),type='l',add=TRUE,lwd=2)
legend("bottomleft", c("PENSSE","MLE"), col=c("black","red"), lty=c(1,3,2), lwd=2, bty='n')
```



P4. (10 points) This question considers the polynomial basis expansion $f(x) = \sum_{k=0}^{K} \xi_k \phi_k(x) = \sum_{k=0}^{K} \xi_k x^k$. Suppose we observe we observe data of the form $\{(x_{i1}, x_{i2}, y_{i1}, y_{i2}); i = 1, \ldots, N\}$ where the data are generated as follows:

$$\begin{aligned} y_{ip} &= f_p(x_{ip}) + \epsilon_{ip} \\ &= \sum_{k=1}^{K} \xi_{kp} \phi_{kp}(x_{ip}) + \epsilon_{ip} \end{aligned}$$

for $p = 1, 2$ where $x_{i1} \sim \text{Uniform}(0, 1)$, $x_{i2} \sim \text{Exponential}(\lambda = 0.01)$, $\epsilon_{i1} \sim N(0, \sigma^2 = 0.5^2)$, $\epsilon_{i2} \sim N(0, \sigma^2 = 0.5^2)$, and $f_p(x) = a_p x^2$ where $a_1 = 5, a_2 = 1 \times 10^{-5}$. Simulate data for $N = 1000$ using the code below and use these simulated data to answer questions (a)-(d) below.

```r
set.seed(19870)
N <- 100
f  <- function(x, a) a*x^2
x1 <- runif(N, 0, 1)
```

```
y1 <- f(x1, a=5) + rnorm(N, sd=0.5)
x2 <- rexp(N, rate=1/100)
y2 <- f(x2, a=1e-05) + rnorm(N, sd=0.5)
```

a. (2 points) Set up the polynomial basis for $K = 5$ for both covariate vectors $\boldsymbol{x}_1 = [x_{11}, \ldots, x_{N1}]^t$ and $\boldsymbol{x}_2 = [x_{12}, \ldots, x_{N2}]^t$ and provide the code to do so.

   Here

$$\boldsymbol{\Phi}_p = \begin{bmatrix} x_{1p}^0 & x_{1p}^1 & \cdots & x_{1p}^5 \\ x_{2p}^0 & \ddots & \cdots & x_{2p}^5 \\ \vdots & \ddots & \cdots & \vdots \\ x_{Np}^0 & \cdots & \cdots & x_{Np}^5 \end{bmatrix}$$

   for $p = 1, 2$. See code below for setting up the bases for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.

```
K   <- 5
Phi1 <- t(vapply(x1, function(x) x^(0:K), numeric(K+1)))
Phi2 <- t(vapply(x2, function(x) x^(0:K), numeric(K+1)))
```

b. (2 points) Estimate the coefficient vectors $\boldsymbol{\xi}_p = [\xi_{1p}, \ldots, \xi_{Kp}]^t$ for $p = 1, 2$ using unpenalized least squares.

   The code below fits unpenalized least squares to the basis matrix created in Problem 4a. The coefficients can be extracted using the *coef.lm()* function.

```
fit4_1_up <- lm(y1 ~ Phi1-1)
fit4_2_up <- lm(y2 ~ Phi2-1)
```

c. (2 points) Estimate the coefficient vectors $\boldsymbol{\xi}_p = [\xi_{1p}, \ldots, \xi_{Kp}]^t$ for $p = 1, 2$ using penalized least squares where the smoothing parameter is selected using cross validation.

   Here I fit penalized least squares using the function written for Problem 2. The penalty matrix $\boldsymbol{S}$ is calculated using the function written for Problem 1b. Three key points:

   i. The *base::chol()* function is used to calculate $\boldsymbol{S}^{0.5}$. Because the *base::chol()* function by default does not work by default for positive semi definite matrices (which our $\boldsymbol{S}$ matrices are here), we need to pre-calculate them and pass them to the function vectorized version of fn_S.

   ii. The code for creating $\boldsymbol{S}^{0.5}$ below will generate a warning which you can safely ignore in this case.

   iii. Smoothing parameter selection:

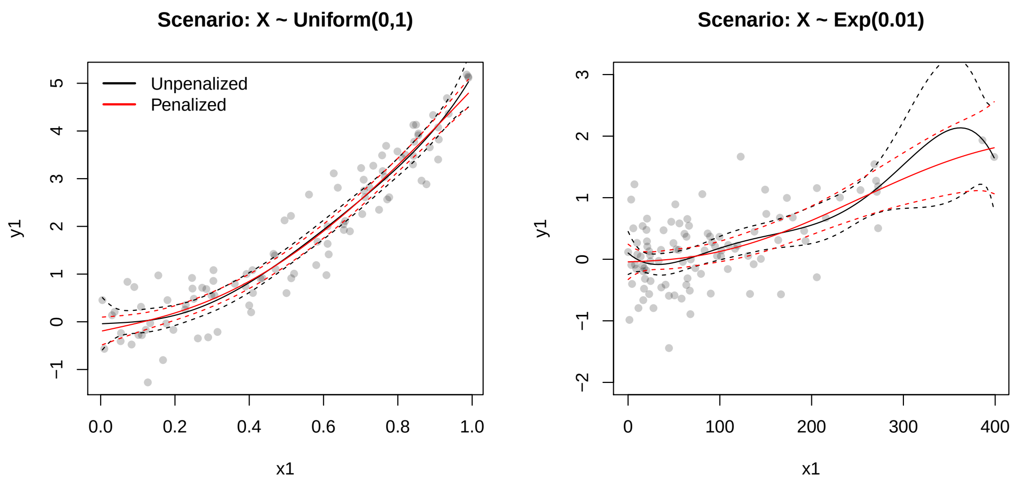
      - The "optimal" smoothing parameter for $X \sim \text{Unif}(0, 1)$ is < -3! This means that if you use the code from lecture which searches over a grid $\log(\lambda) \in [-3, 20]$ you'll miss the optimal λ based on the GCV criteria (no points were deducted for missing this, and the estimated fits are nearly identical, but boundary issues are something to be aware of).

- The GCV function for $X \sim \text{Exponential}(0.01)$ is **very** poorly behaved for low values **and** does not vary much (many possible fits perform equally well). You can see this yourself by using my *fn_PENNSE_grid()* function which returns the GCV criteria over the grid searched. This poor behavior means that the *optim()* function is likely to get stuck in a local minimum if your starting value is low. This is why I use a relatively large value of $\log(\lambda) = 10$ as a starting value. In this example the grid search is actually better than direct numeric optimization.

```r
## get penalty matrices for the first and second data (x,y) pairs
S1 <- outer(0:K, 0:K, Vfn_S, lb=min(x1), ub=max(x1))
S2 <- outer(0:K, 0:K, Vfn_S, lb=min(x2), ub=max(x2))
## get S1^(0.5)
S1_sqrt <- chol(S1, pivot=TRUE)
S1_sqrt <- S1_sqrt[,order(attr(S1_sqrt, "pivot"))]
## get S2^(0.5)
S2_sqrt <- chol(S2, pivot=TRUE)
S2_sqrt <- S2_sqrt[,order(attr(S2_sqrt, "pivot"))]

fit4_1_pen <- fn_PENSSE_opt(y=y1, Phi=Phi1, S=S1, S_sqrt=S1_sqrt)
fit4_2_pen <- fn_PENSSE_opt(y=y2, Phi=Phi2, S=S2, S_sqrt=S2_sqrt,loglambda_st=10)
```

d. (2 points) Plot both unpenalized and penalized estimates of $\hat{f}_p(x)$ along with $\hat{f}_p(x) + / - 2\text{SE}(\hat{f}_p(x))$ over the range of observed x values.
Below I plot the unpenalized and penalized estimates for the two data generating mechanisms: $x \sim \text{Uniform}(0,1)$ (left panel) and $x \sim \text{Exponential}(0.01)$ (right panel).

```r
## set up the basises
x1ind_pred <- seq(min(x1), max(x1), len=100)
x2ind_pred <- seq(min(x2), max(x2), len=100)
Phi1_pred <- t(vapply(x1ind_pred, function(x) x^(0:K), numeric(K+1)))
Phi2_pred <- t(vapply(x2ind_pred, function(x) x^(0:K), numeric(K+1)))
## get point estimates
# X ~ Unif(0,1)
f1hat_up  <- Phi1_pred %*% coef(fit4_1_up)
f1hat_pen <- Phi1_pred %*% fit4_1_pen$xi
# X ~ Exp(0.01)
f2hat_up  <- Phi2_pred %*% coef(fit4_2_up)
f2hat_pen <- Phi2_pred %*% fit4_2_pen$xi
## get pointwise SE
# X ~ Unif(0,1)
SE_f1hat_up  <- sqrt(diag(Phi1_pred %*% vcov(fit4_1_up) %*% t(Phi1_pred)))
SE_f1hat_pen <- sqrt(diag(Phi1_pred %*% fit4_1_pen$var_xi %*% t(Phi1_pred)))
# X ~ Exp(0.01)
SE_f2hat_up  <- sqrt(diag(Phi2_pred %*% vcov(fit4_2_up) %*% t(Phi2_pred)))
SE_f2hat_pen <- sqrt(diag(Phi2_pred %*% fit4_2_pen$var_xi %*% t(Phi2_pred)))

par(mfrow=c(1,2))
plot(x1,y1,pch=16,col=rgb(0,0,0,0.2),xlab="x1",ylab="y1",main="Scenario: X ~ Uniform(0,1)")
matplot(x1ind_pred, cbind(f1hat_up, f1hat_up - 1.96*SE_f1hat_up, f1hat_up + 1.96*SE_f1hat_up),
        col='black',type='l',add=TRUE,lty=c(1,2,2))
matplot(x1ind_pred, cbind(f1hat_pen, f1hat_pen - 1.96*SE_f1hat_pen, f1hat_pen + 1.96*SE_f1hat_pen),
        col='red',type='l',add=TRUE,lty=c(1,2,2))
legend("topleft", c("Unpenalized","Penalized"), col=c("black","red"), lty=1,lwd=2,bty='n')

plot(x2,y2,pch=16,col=rgb(0,0,0,0.2),xlab="x1",ylab="y1",main="Scenario: X ~ Exp(0.01)",
     ylim=c(-2,3))
matplot(x2ind_pred, cbind(f2hat_up, f2hat_up - 1.96*SE_f2hat_up, f2hat_up + 1.96*SE_f2hat_up),
        col='black',type='l',add=TRUE,lty=c(1,2,2))
matplot(x2ind_pred, cbind(f2hat_pen, f2hat_pen - 1.96*SE_f2hat_pen, f2hat_pen + 1.96*SE_f2hat_pen),
        col='red',type='l',add=TRUE,lty=c(1,2,2))
```



e. (2 points) Suppose we modified the polynomial basis to be of the form $f(x) = \sum_{k=1}^{K} \xi_k \phi_k(x) = \sum_{k=2}^{K} \xi_k x^k$. What would happen to the estimated function $\hat{f}(x)$ as λ approaches ∞?

If we remove the constant and linear components of the basis, the as $\lambda \to \infty$ all remaining coefficients will be penalized to 0. Thus as $\lambda \to \infty$, the estimated $\hat{f}(x)$ will be a horizontal line at 0 ($\hat{f}(x) = 0$ for all x).

P5. (10 points) Design and implement a simulation study which simulates data according to the model:

$$y_i = f(x_i) + \epsilon_i$$
$$\epsilon_i \sim N(0, \sigma_\epsilon^2)$$

Using either a cubic regression spline or B-spline basis for $f(x)$, assess the bias, mean squared error, and coverage probability of $\hat{f}(x)$ estimated using both penalized and unpenalized least squares for various choices of number of observations (N) and association structure (f) and number of basis functions (K). At least three combinations of N, f, and K must be included in the simulation study (at least $3 \times 3 \times 3 = 27$ total scenarios). Summarize your results and support your claims using either tables or figures.

The code for running my simulation study is provided in the separate R script "HW1_P5.R". I considered the following simulation settings:

- $X \sim \text{Unif}(-2, 2)$

- $\sigma_\epsilon^2 = 1$
- $N \in \{100, 1000, 5000\}$
- $K \in \{5, 20, 50\}$
- $f_1(x) = x^2$, $f_2(x) = \sin(2\pi x)$, $f_3(x) = \cos^2(x) + x^3$

For each setting I simulated $1,000$ datasets. The results of my simulation study are presented in Figures 1. Specifically, Figure 1 (A), (B), and (C) presents the bias, MSE, and 95% CI coverage probability across the 1000 simulations for each simulation scenario. Within each of Figure 1 (A), (B), and (C), each panel corresponds to different combinations of sample size and penalized versus unpenalized fits (columns) and true association structure (rows). Note that I have truncated the y-axis for bias and MSE to be between $(0, 0.1)$ and $(-0.1, 0.1)$ to allow for visual comparison between scenarios. Poor performance at the bondaries is much more severe than we see plotted here in some scenarios.

First, looking at Figure 1 (A), we see that bias generally decreases across the range of observed x values with increasing sample size for both penalized and unpenalized fits. In addition, we see that the bias is most severe at the boundaries of the covariate values. This is to be expected and is known as a "boundary effect". The boundary effect is most notable when we have a low sample size and a large number of basis functions, but diminishes with increasing sample sizes. In addition, we can see that there appears to be a non-trivial bias for $K = 5$ using penalized splines even with large sample sizes. In contrast, this observable bias is only apparent for f3 in the unpeanlized fits for larger sample sizes (N=500, 1000). The bias across true association structures in the penalized fits for $K = 5$ is likely due to the asymptotics of penalized splines (not discussed in class, theory is quite dense) which require K increase with N to avoid bias. In contrast, unpenalized splines are unbiased. For f3, the bias in the unpeanlzied fit is likely due to $K = 5$ being insufficiently flexible to capture the true association structure.

Next, looking at Figure 1 (B), we see that mean squared error of the estimated function decreases with sample size for both penalized and unpenalized fits, and that for small sample sizes we are estimating the function at the boundaries poorly. As sample size increases we estimate the boundaries better. This is consistent with our observations from Figure 1 (A). We also see that MSE decreases consistently with increasing sample size. Interestingly, $K = 50$ performs worse across all simulation scenarios while $K = 5$ performs best. This is possibly because we used GCV for smoothing parameter selection and tend to undersmooth the data for large K. Lastly, the penalzied fits perform notably better than unpenalized fits across all scenarios.

Turning to Figure 1 (C), for penalized splines, we see that we achieve nominal coverage probabilities for our 95% CIs for $K = 20$ or $K = 50$ across simulation scenarios, except at the boundaries. However, for $K = 5$ we only achieve nominal coverage for smaller sample sizes ($N = 100$). The poor coverage for $K = 5$ for $N = 500, 1000$ is likely due to the bias we observed in Figure 1 (A). For the unpenalzied fits we acheive nominal coverage across simulation scenarios except for larger sample sizes $N = 500, 1000$, likely due to the bias we see in Figure 1.

```
## check to see if the simulation study has been run
## if not, source the relevant code for running
## the simulations
if(!file.exists(here(results_path, "HW1_P5_results.rds"))){
  source(here(code_path, "HW1_P5.R"))
}
```

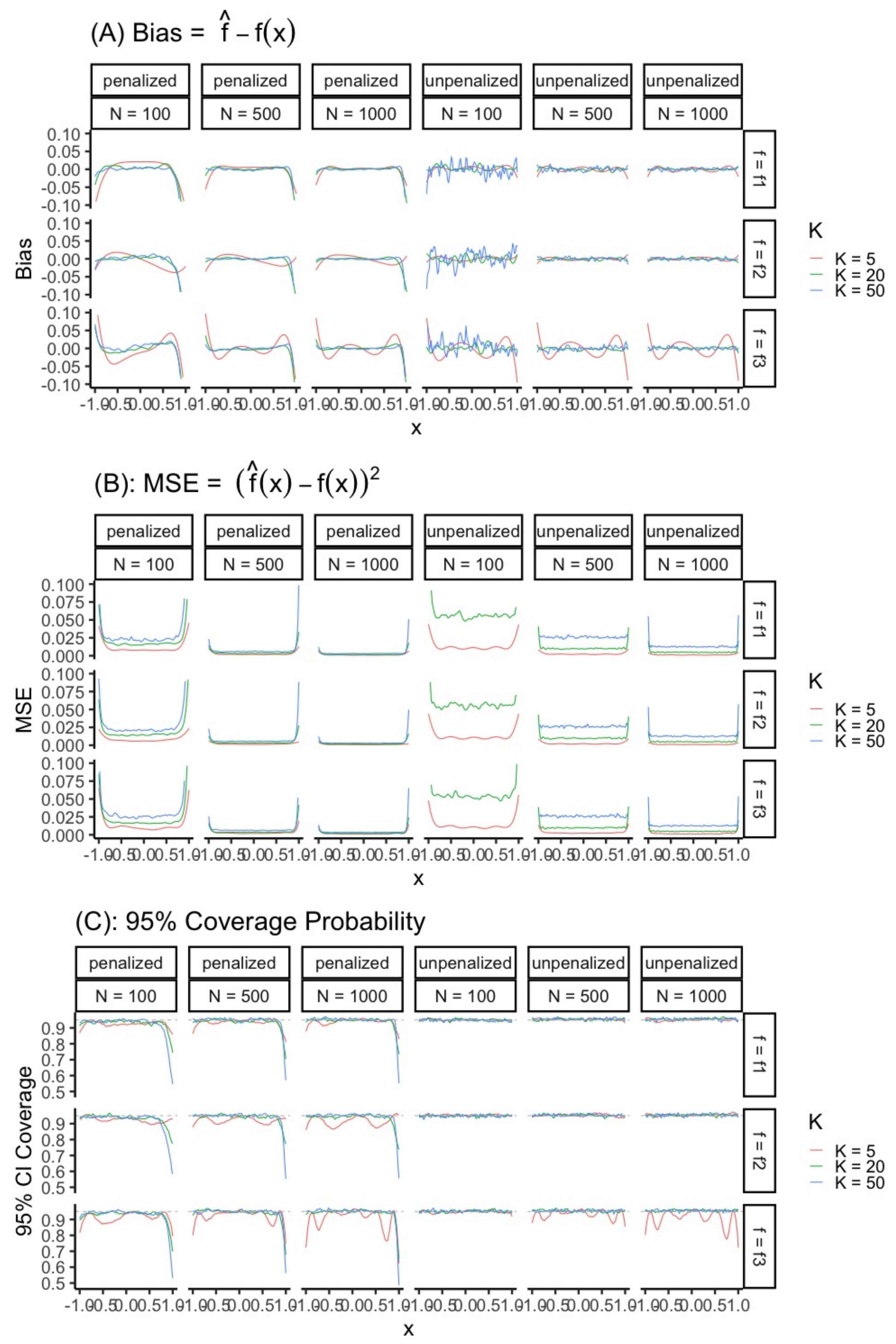


Figure 1: Estimated performance measures evaluated across the range of x values $[-1, 1]$. (A) Bias of the estimated $\hat{f}$; (B) Mean squared error of the estimated $\hat{f}$; (C) 95% Confidence interval coverage.

P6. (10 points) This question relates to the identifiability constraints imposed by the *mgcv* package in *R*. Suppose we have a full rank spline basis matrix $\mathbf{\Phi}_{N\times K}$ which contains the constant vector ($\boldsymbol{c}_{N\times 1}$) in it's column space. Show that column mean centering $\mathbf{\Phi}$ results in a new basis matrix $\mathbf{\Phi}^*$ which does not contain the constant vector $\boldsymbol{c}$ in it's column space.

P7. (10 points) Thus far we have assumed independent observations. This problem investigates the behavior of automatic smoothing parameter methods when data are correlated. Here, we will take this idea to the extreme where the correlation between observations is 1. To do so, we will be investigating the association between age and average total daily volume of physical activity measured via wearable hip worn accelerometers in the NHANES 2003-2006 accelerometry cohort.

Specifically, we will calculate average daily total volume of physical activity within individuals, and then fit our model to two datasets. The first dataset will contain duplicated rows (duplicated (x_i, y_i) pairs). The second dataset will contain a "correct" dataset with one row per subject. The code to set up the two datasets is below.

```r
library("readr"); library("dplyr"); library("mgcv")
df       <- read_rds("NHANES_AC_processed.rds")
min_cols <- paste0("MIN",1:1440)
df$TAC   <- rowSums(df[,min_cols], na.rm=TRUE)
df_fit   <-
    df %>%
    filter(good_day == 1, !is.na(Age)) %>%
    group_by(SEQN) %>%
    mutate(TAC_mn = mean(TAC))
df_full <- df_fit
df_ind  <- df_fit[!duplicated(df_fit$SEQN),]
```

The code for fitting the models and plotting results for this problem are included in the R script "HW1_P7.R".

a. (5 points) Fit the additive model $\bar{\text{TAC}}_i = f(\text{Age}_i) + \epsilon_i$ where $\bar{\text{TAC}}_i$ is the subject average daily total activity count (TAC_mn in the code above) using both the full dataset with repeated observations ("df_full") and the dataset with one row per subject ("df_ind"). Use *mgcv::gam()* for model estimation with smoothing parameter selection done by generalized cross validation. Plot the estimated average TAC as a function of age plus/minus two standard errors for each fit. Comment on any differences you see.

The estimated fits plus/minus two standard errors are presented in Figure 2 below. The estimate from the dataset with repeated observations ("df_full") are presented in black while the results from the dataset with one observation per participant ("df_ind") are presented in red. We can observe three primary differences:

i. The estimated function $\hat{f}(\text{Age})$ is much less smooth when the independence model is fit to the data with repeated observations.
ii. The estimated standard errors are smaller when the independence model is fit to the data with repeated observations.
iii. The estimated function appears to be slightly lower (lower total activity) for individuals between the ages of 20-40 when fit to the data with one row per participant ("df_ind")

That the estimated function is both more wiggly and has narrower confidence intervals (smaller standard errors) is entirely expected. First, the point of using GCV to select our smoothing parameter is to approximate out-of-sample prediction error to prevent overfitting to the data ($\hat{f}$ which are too wiggly) using leave-one-observation-out cross validation. Because leaving one observation out still leaves multiple records of the same individual in the dataset (repeated records), the GCV criteria does not accurately estimate out-of-sample error. Second, because the model does not account for the (perfect) dependence of observations in the data, standard errors are underestimated as the model thinks we have many more data points than we actually do.

```
## check to see if the code for P7 has been run
## if not, source the relevant code
if(!file.exists(here(figure_path, "fig_HW1_P7a.jpeg"))){
  source(here(code_path, "HW1_P7.R"))
}
```

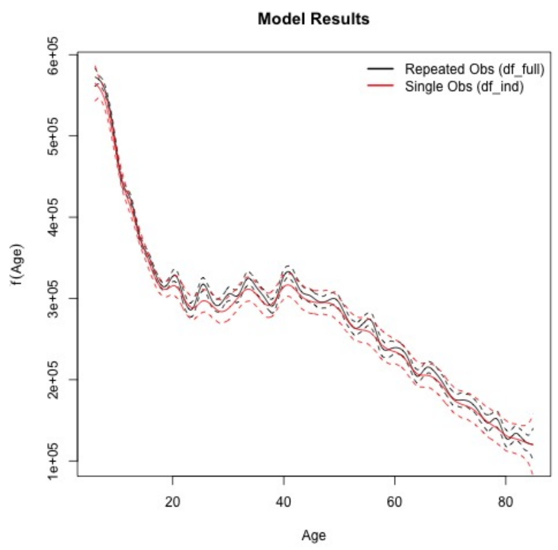


Figure 2: Estimated average daily total activity count as a function of age (solid lines) plus/minus two (pointwise) standard errors (dashed lines) using data with either repeated observations (black lines) or one observation per participant (red lines).

b. (5 points) Choose the optimal λ using leave-one-subject-out cross validation fit on the full dataset with repeated observations ("df_full") and plot the estimated average TAC as a function of age plus/minus two standard errors. Comment on any differences you see with the results from part (a) and hypothesize on the reasons for any observed differences.

Note that because of long computation times, I allowed for the use of leave-several-subject out cross validation (LSS CV) and provided some template code on the Canvas announcements. Figure 3 presents the results from leave-several subject out cross-validation created using the template code I provided which is contained in the HW1_P7.R script. Figure 3 (A) presents the estimated coefficient with pointwise 95% confidence intervals. Figure 3 (B) presents the estimated wiggliness of each of the three models fit in this problem as measured by $\boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}$. From left to right we have the wiggliness of the model fit: (left) to the data with repeated rows where smoothing parameter selection is done using GCV, (center) to the data with one row per participant, (right) to the data with repeated rows where smoothing parameter selection is done using LSS CV. Figure 3 (C) presents the estimated difference in estimated average TAC comparing the model fit to the data with one row per participant and the model fit to the full data with smoothing parameter selection done using LSS CV.

First, consider Figure 3 (A). Clearly the estimated function is visually less wiggly than when smoothing parameter selection is done using GCV. This arises precisely because of the point raised in my answer to the previous part of this question. When we formalize our measure of wiggliness as $\boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}$, presented in Figure 3 (B), we see that LSS CV fit to the repeated data is actually slightly less wiggly than the model fit to the data with one row per participant. However, looking at Figure 3 (C), we see that the estimated average TAC is still consistently higher when using LSS CV on the repeated data as compared to the fit to the data with one row per participant. This is likely due to individuals with higher average TAC having more days of data as their "extra" days pull the estimated population average upward. Finally, from Figure 3 (A) we see while using LSS for smoothing parameter selection yields more "appropriate" smoothness of the estimated function, the standard errors are still too small.

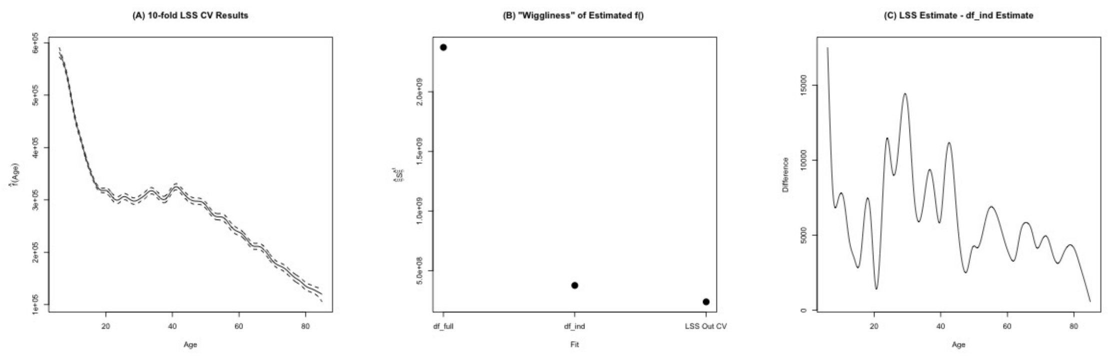


Figure 3: (A) The estimated coefficient (solid line) with pointwise 95% confidence intervals (dashed lines) from LSS CV. (B) The estimated wiggliness of each of the three models fit in this problem as measured by $\boldsymbol{\xi}^t \boldsymbol{S} \boldsymbol{\xi}$. (C) The estimated difference in estimated average TAC comparing the model fit to the data with one row per participant and the model fit to the full data with smoothing parameter selection done using LSS CV.

P8. (10 points) This problem investigates REML versus generalized cross-validation as a method for smoothing parameter selection. Using the same NHANES dataset from the previous problem, calculate the total daily activity counts (TAC) for each subject-day, filter out "bad days" (days with fewer than 10 hours of estimated wear time or individuals with poor quality data) using the "good_day" column, remove individuals with missing age data, and then obtain subject-specific average TAC. The code to do so is presented below.

```
library("readr"); library("dplyr"); library("mgcv")
df       <- read_rds("NHANES_AC_processed.rds")
min_cols <- paste0("MIN",1:1440)
df$TAC   <- rowSums(df[,min_cols], na.rm=TRUE)
df_ind   <-
    df %>%
    dplyr::select(-one_of(min_cols)) %>%
    filter(good_day == 1, !is.na(Age)) %>%
    group_by(SEQN) %>%
    summarize(Age=Age[1], TAC_mn = mean(TAC))
```

a. (5 points) Fit the additive model $\text{TAC}_i = f(\text{Age}_i) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma_\epsilon^2)$ to the full data using penalized cubic regression splines with $K = 50$ using both REML and GCV for smoothing parameter selection. The code for model fitting can be found below. Using the fitted models, plot the estimated average TAC plus/minus two standard errors over the range of observed ages.

```
fit_GCV  <- gam(TAC_mn ~ s(Age, k=50, bs="cr"), method="GCV.Cp", data=df_ind)
fit_REML <- gam(TAC_mn ~ s(Age, k=50, bs="cr"), method="REML", data=df_ind)
```

Below I source the code for answering both parts of this question.

Figure 4 presents the estimated average TAC as a function of age from the model fit using either GCV (left panel) or REML (right panel). Overall the estimated trends with age ares similar, though the estimate is much more smooth for REML than for GCV.

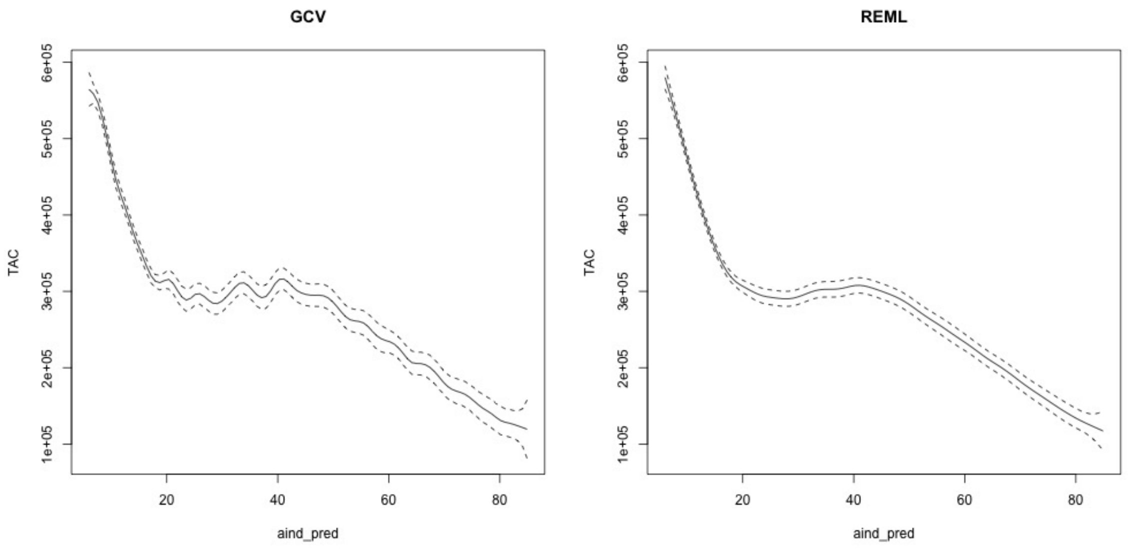


Figure 4: Estimated average TAC (solid lines) with 95% pointwise confidence intervals (dashed lines) derived from models fit using GCV (left panel) or REML (right panel) for smoothing parameter selection.

b. (5 points) Using these data, obtain 1000 randomly selected sub-samples for each of $N \in \{100, 500, 1000, 2000\}$. Fit the additive model from the previous sub-question for each of these re-sampled datasets using both REML and GCV as smoothing parameter selection criteria. Compare the shapes of the estimated functions $\hat{f}$. Does either method tend to yield smoother versus more wiggly estimated functions? How does the comparison change with sample size? Support your claims using numeric summaries of the estimated functions.

Figure 5 plots the estimated wiggliness of $\hat{f}$(Age) estimated using both GCV and REML from 1000 randomly selected subsamples of size $N = 100, 500, 1000, 2000$ from the NHANES data. Wiggliness is assessed using $\hat{\boldsymbol{\xi}}^t \boldsymbol{S} \hat{\boldsymbol{\xi}}$. The actual plot is on the $\log_{10}$ scale due to very large outlying values dominating the visual distribution. We see that for smaller sample sizes (N=100, 500) REML is substantially smoother than GCV on average and comparable or smoother at the median/quartiles. For larger sample sizes (N=1000,2000), the median wiggliness for GCV smoothing parameter selection is smoother than for REML.

However, because the wiggliness of estimates using GCV is substantially more variable and right skewed across all sample sizes, the average wiggliness is consistently less for REML than GCV.

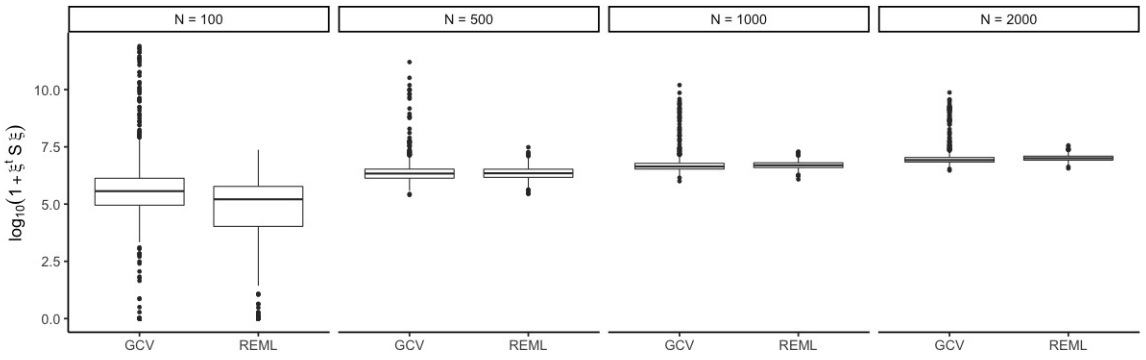


Figure 5: Wiggliness of the estimated $\hat{f}(\text{Age})$ from models estimated using either GCV or REML for a variety of subsamples of various size ($N = 100, 500, 1000, 2000$) from the NHANES 2003-2006 dataset. Note that wiggliness is defined as $\hat{\boldsymbol{\xi}}^t \boldsymbol{S} \hat{\boldsymbol{\xi}}$ and the plots are on the $\log_{10}$ scale due to extremely large values skewing the distributions.