Biostatisitcs 285: Project 2

Emilie Campos

2020-02-27

Problem 1. Consider the Diffusion tensor Imaging (DTI) data set in the R pacakge `refund`. DTI is a magnetic resonance methodology. The DTI data set in `refund` provides a measure of the quantity of white matter (fractional anisotropy) along 93 equally spaced locations in the corpus callosum (`cca` - corpus callosum anysotropy). While this measure is essentially spatial, we will model `cca` as a function from $[0, 1] \to L^2$ evaluated at 93 sampling points. Patients undergo multiple visits, where a measure of their cognitive function is assessed via PASAT scores, and structure brain imaging is performed via DTI. As you examine the data-set, consider a complete case analysis by defining:

```
DTI1 <- DTI[complete.cases(DTI) & visit == 1, ]
```

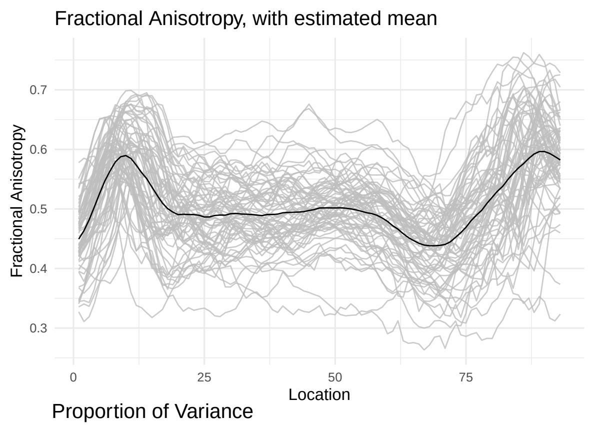
a. Consider the fractional anisotropy in the corpus callosum at visit 1. Carry out a Functional Principal Components analysis of the cca data, using the sample mean and covariance as estimators of the related functional quantities. Compare your results in 1.a, with the same analysis carried out using the sandwich smoother in `fpca.face`. Interpret your results.

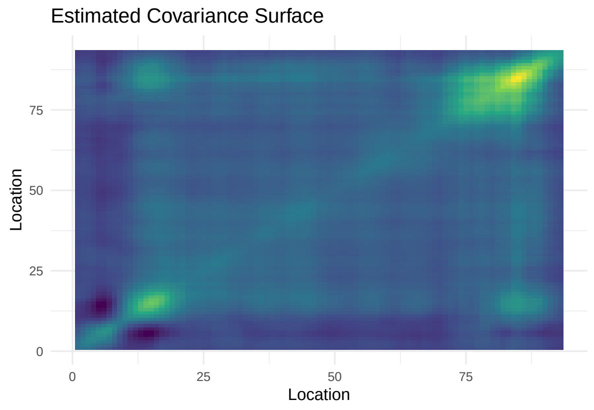
   Solution:

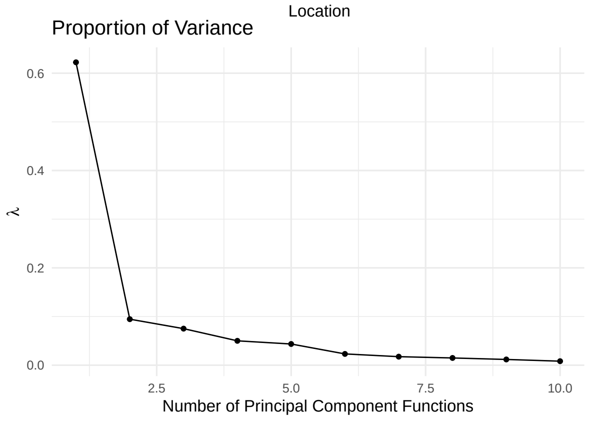
```
# filter by visit 1 and select the cca column (which is a dataframe)
cca <- data.frame(DTI1[, "cca"])

# estimate the mean by taking the row means of the cca_visit1 dataset
mu_hat <- colMeans(cca)

# center the cca_visit1 data
cca_center <- cca
for (i in 1:66) {
  cca_center[i, ] <- cca[i, ] - mu_hat
}

# calculate the covariance using the centered data
c_hat <- t(as.matrix(cca_center)) %*% as.matrix(cca_center)
```

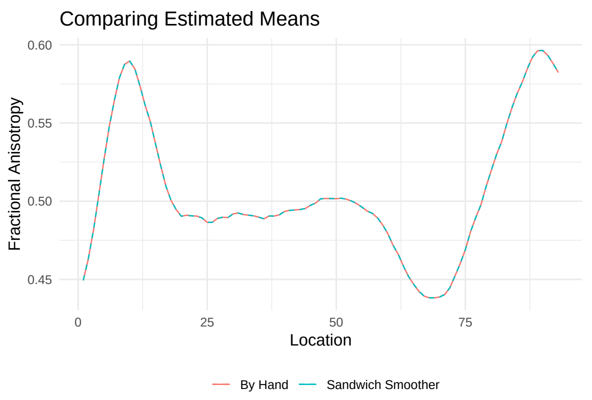
```
# use the singular value decomposition to find the eigendecomposition of
# the covariance
svd_out <- svd(c_hat)
est_FPCs <- svd_out$v
est_lambdas <- svd_out$d
```

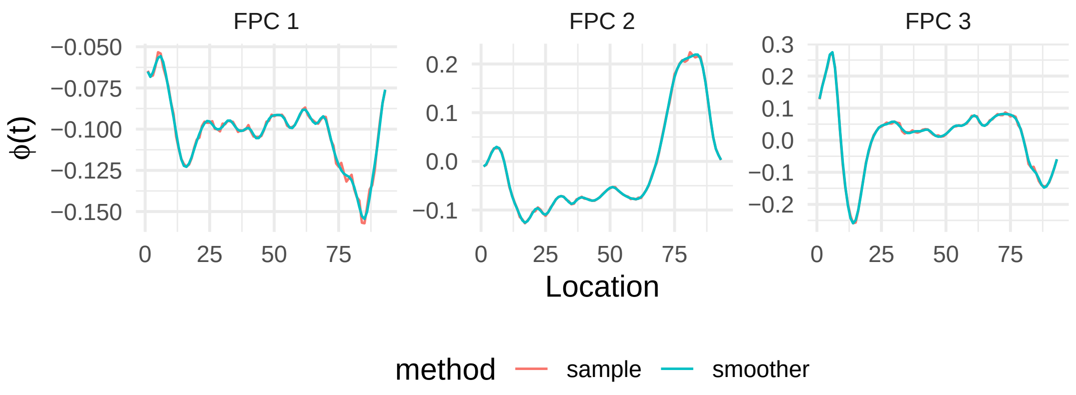
```
fpca_out <- fpca.face(as.matrix(cca))
```











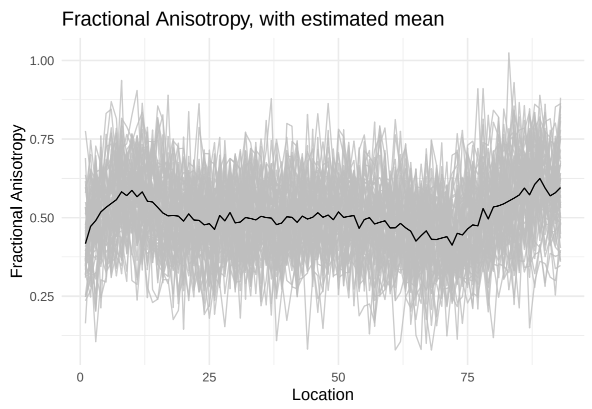
Up to a sign, the eigenfunctions estimated by hand and using the sandwich smoother are nearly identical. Although, the eigenfunctions estimated using the sandwich smoother are smoother than the ones estimated by hand. As for interpretation of the eigenfunctions, the first eigenfunction characterizes the direction of the most variation (62.2%). The estimated eigenfunction is relatively flat with a dip at the beginning and end implying there is constant variance across the middle locations and the locations 1-25 and 75-93 are more variable.

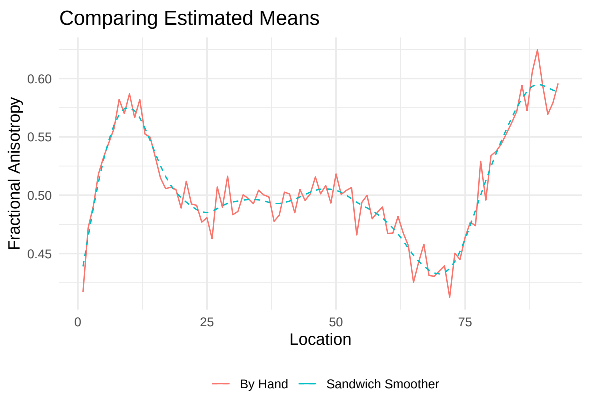
b. Construct a contaminated version of the cca data by adding independent $\mathrm{N}(0, sd = 0.1)$ errors to the data. Using the sample covariance estimator examine the ensuing eigenfunctions in comparison to the estimator in `fpca.face`. Construct your own bivariate smoother of the empirical covariance, describe the details of your implementation, and compare your results with the FACE estimator.

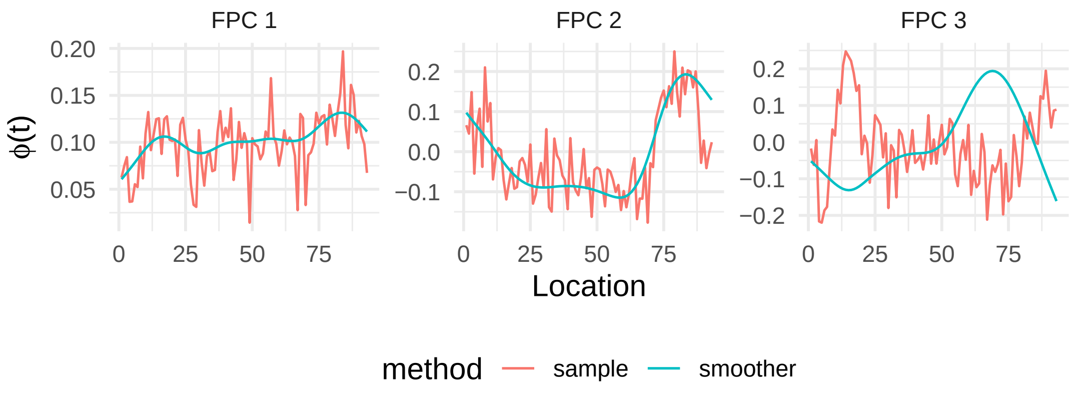
Solution:

```
# contaminate cca data with iid n(0, 0.1)
noisy_cca <- cca + matrix(rnorm(66 * 93, mean = 0, sd = 0.1), ncol = 93)
```

```
# estimate the mean by taking the row means of the noisy_cca dataset
noisy_mu_hat <- colMeans(noisy_cca)

# center the cca_visit1 data
noisy_cca_center <- noisy_cca
for (i in 1:66) {
  noisy_cca_center[i, ] <- noisy_cca[i, ] - noisy_mu_hat
}

# calculate the covariance using the centered data
noisy_c_hat <- t(as.matrix(noisy_cca_center)) %*%
  as.matrix(noisy_cca_center)

# use the singular value decomposition to find the eigendecomposition of
# the covariance
noisy_svd_out <- svd(noisy_c_hat)
noisy_est_FPCs <- noisy_svd_out$v
noisy_est_lambdas <- noisy_svd_out$d

noisy_fpca_out <- fpca.face(as.matrix(noisy_cca))
```



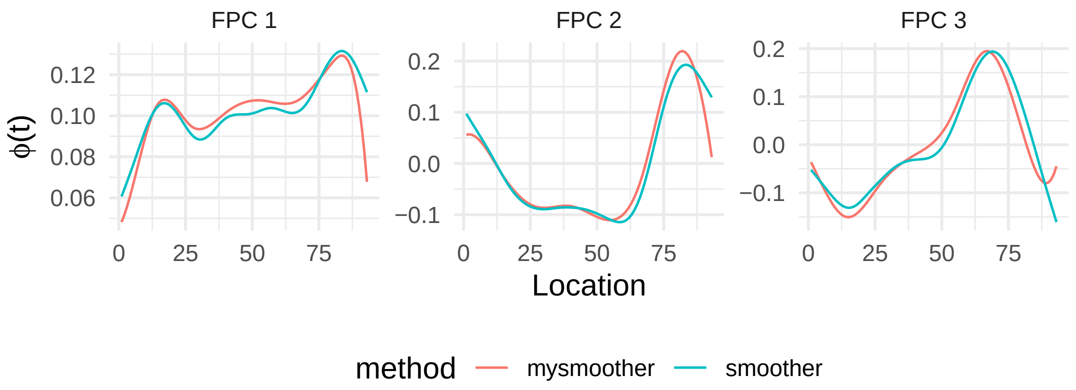




The eigenfunctions estimated using the sample covariance are entirely useless since they are so noisy.

```
# 1. create a cubic b-spline basis of (0, 93)
bs_basis <- fda::create.bspline.basis(
  rangeval = c(1,93),
  nbasis = 10,
  norder = 4
)
```

```r
B <- eval.basis(seq(1, 93), bs_basis)

# 2. construct smoother matrix s(lambda) = B(B^TB + lambda P)^{-1} B^T
P <- diag(1, nrow = 10)
lambdas <- seq(0, 10, length.out = 100)
s_lambda <- map(lambdas, ~ B %*% solve(t(B) %*% B + .x * P) %*% t(B))

# 3. use cross validation to choose lambda
c_hat_smoothed_list <- map(s_lambda, ~ .x %*% noisy_c_hat %*% .x)
# f_hat <- map_dfc(c_hat_smoothed, ~ .x %*% )

# 4. face estimator c_hat = s(lambda) k_hat s(lambda), k_hat = sample
#    covariance
c_hat_smoothed <- s_lambda[[15]] %*% noisy_c_hat %*% s_lambda[[15]]

# 5. eigendecomp of c_hat
smoothed_svd_out <- svd(c_hat_smoothed)
smoothed_est_FPCs <- smoothed_svd_out$v
smoothed_est_lambdas <- smoothed_svd_out$d
```



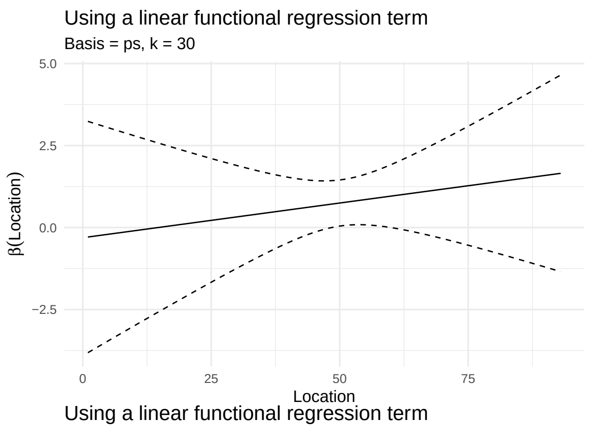
Using my own bivariate smoother, which is based on the FACE estimator, the estimates are nearly identical.

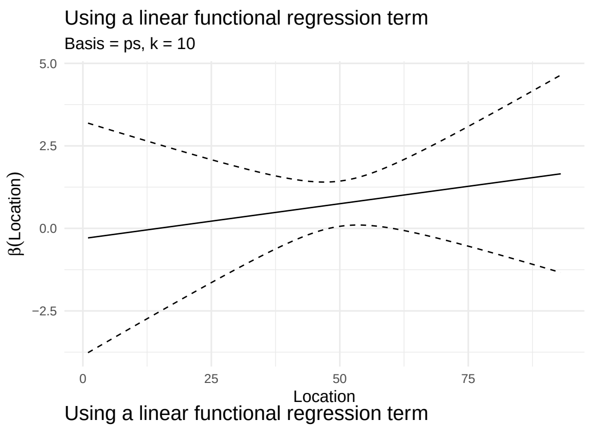
c. We will now consider a scalar on function regression model, aiming to estimate the effect of cca on the pasat cognitive score (first visit only). Specifically, let y_i = pasat score for subject i and $x_i(t)$ be the cca along 93 equally spaced locations in the corpus callosum. We consider the model

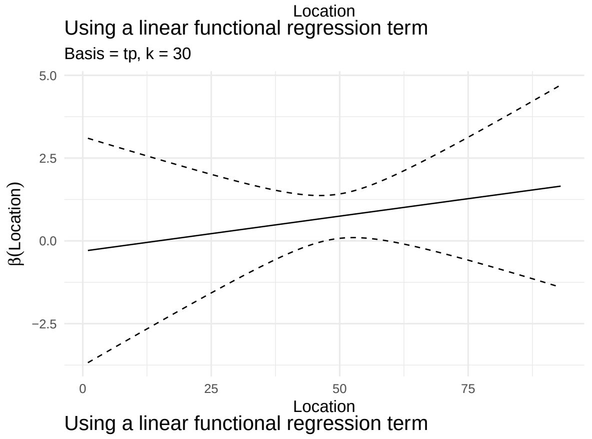
$$y_i = \alpha + \int \beta(t)x_i(t)dt + \epsilon_i; \quad \epsilon_i \sim_{\text{iid}} \mathrm{N}(0, \sigma^2).$$

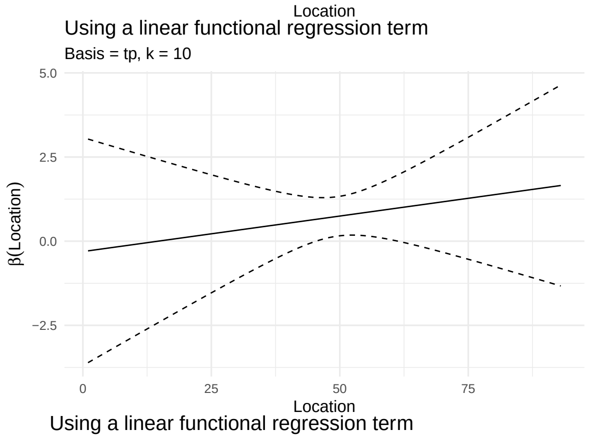
Estimate the model parameters using the `refund` function `pfr`, using the `lf` and the `fpc` options. Comment on the estimation strategy used in these functions and assess sensitivity of estimation to these different estimation strategies.

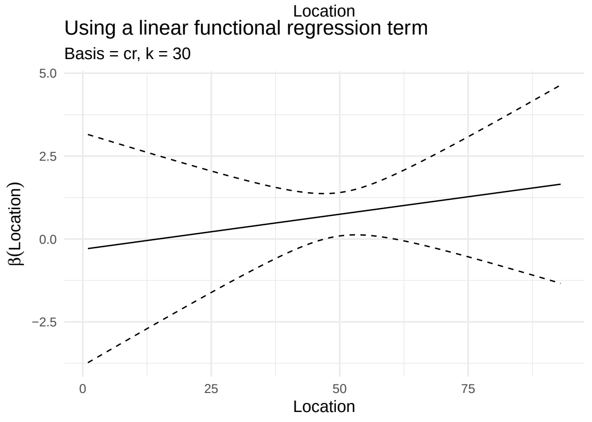
Solution:

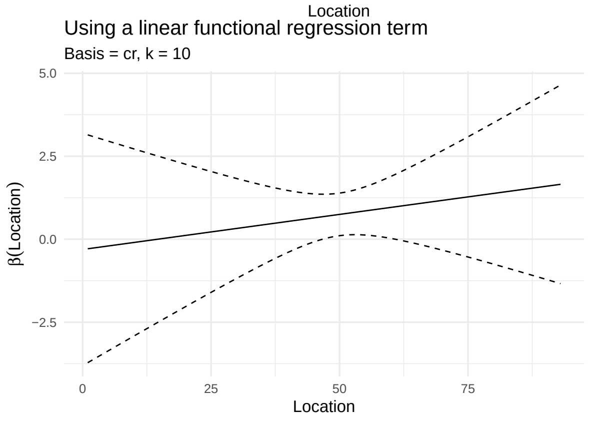




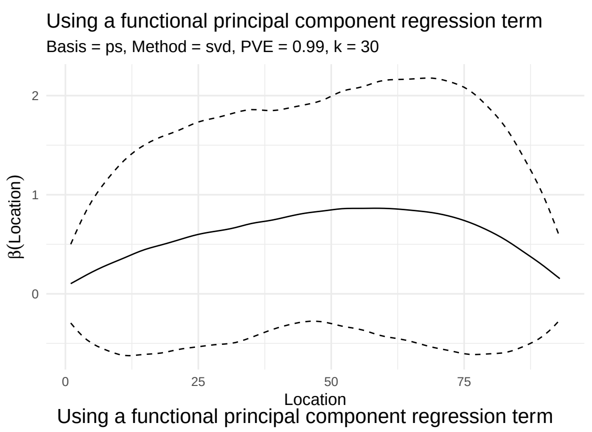
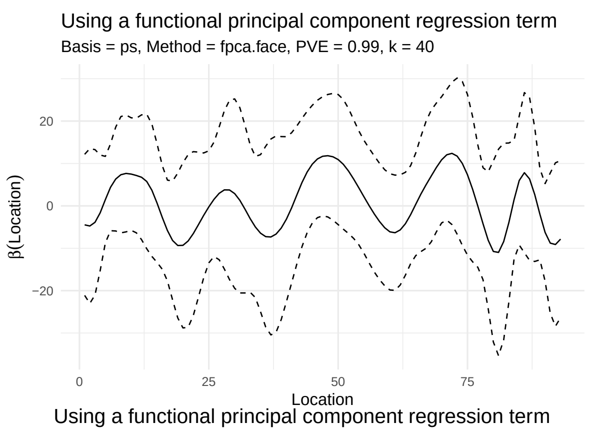
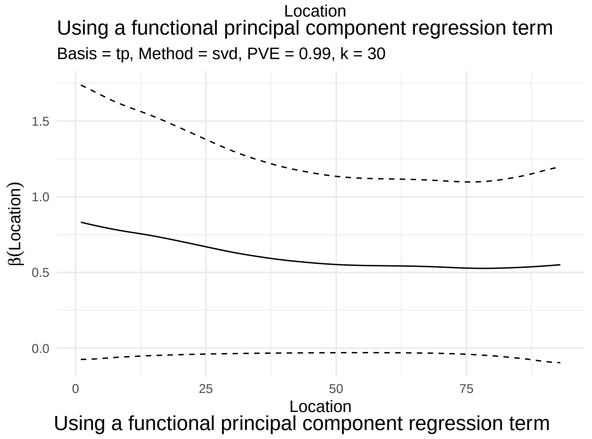
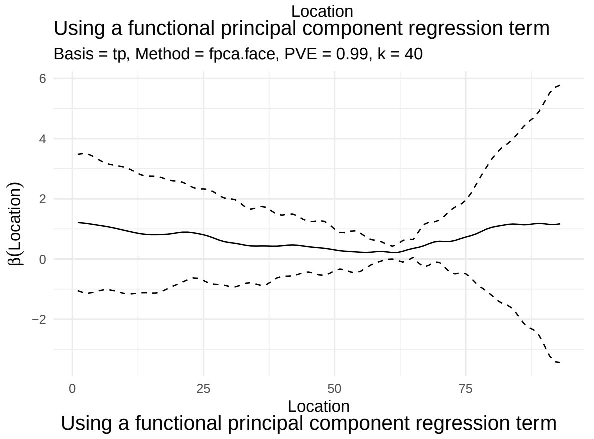
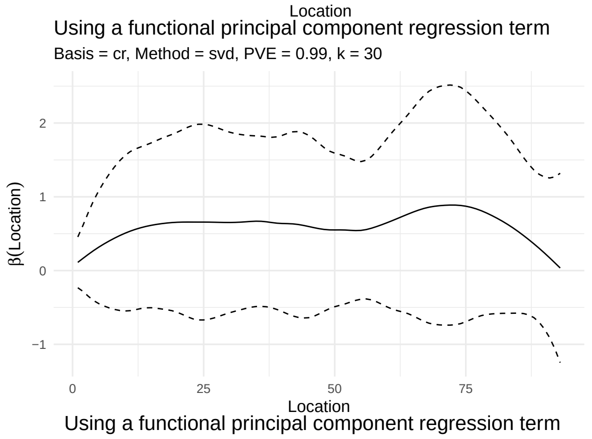
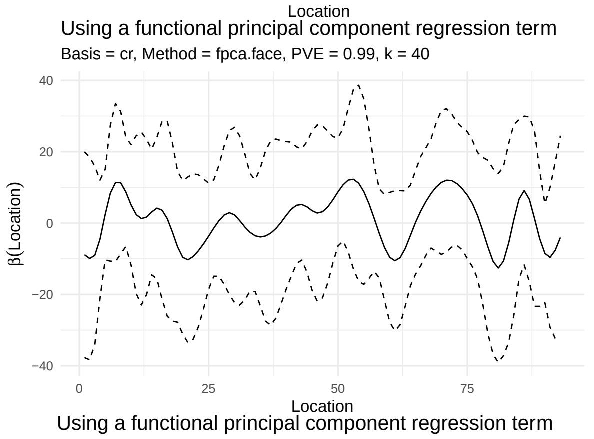
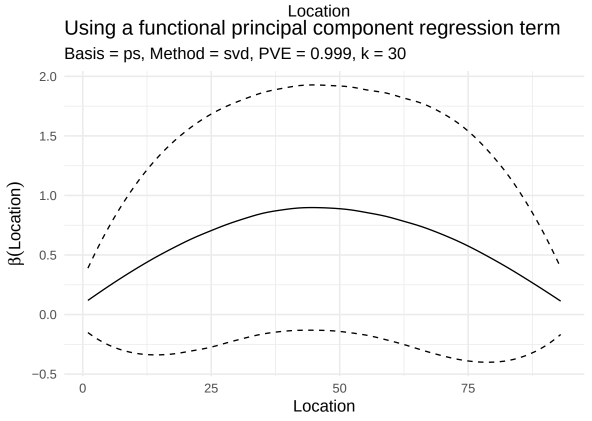
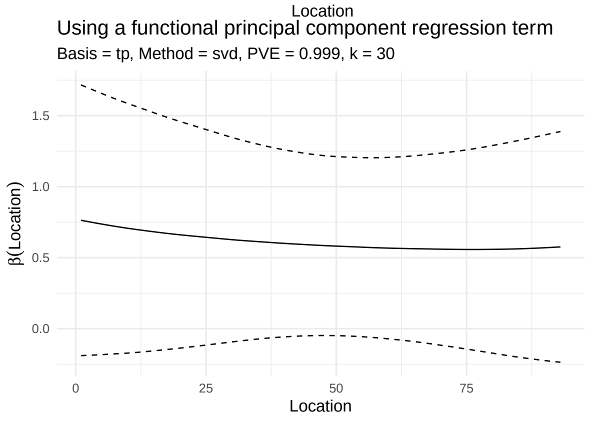


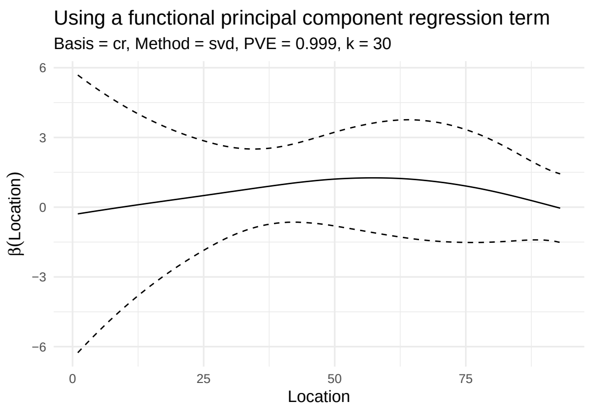






The `lf` function defines a term $\int_T \beta(t)X_i(t)dt$, where $\beta(t)$ is an unknown coefficient function and $X_i(t)$ is a functional predictor on the closed interval T. The `bs` option is choosing the smooth terms. The `k` option is choosing the dimension of the basis used to represent the smooth term. Lastly, the `argvals` option is the indices of evaluation, where I used `c(1, 93)` since these are the locations of the `cca` data. The `lf` function is not senstive to changes in these parameters.


Using a functional principal component regression term
Basis = ps, Method = svd, PVE = 0.99, k = 30
β(Location)
Location
2
1
0
0
25
50
75

Using a functional principal component regression term
Basis = ps, Method = fpca.face, PVE = 0.99, k = 40
β(Location)
Location
20
0
−20
0
25
50
75

Using a functional principal component regression term
Basis = tp, Method = svd, PVE = 0.99, k = 30
β(Location)
Location
1.5
1.0
0.5
0.0
0
25
50
75

Using a functional principal component regression term
Basis = tp, Method = fpca.face, PVE = 0.99, k = 40
β(Location)
Location
6
4
2
0
−2
0
25
50
75

Using a functional principal component regression term
Basis = cr, Method = svd, PVE = 0.99, k = 30
β(Location)
Location
2
1
0
−1
0
25
50
75

Using a functional principal component regression term
Basis = cr, Method = fpca.face, PVE = 0.99, k = 40
β(Location)
Location
40
20
0
−20
−40
0
25
50
75

Using a functional principal component regression term
Basis = ps, Method = svd, PVE = 0.999, k = 30
β(Location)
Location
2.0
1.5
1.0
0.5
0.0
−0.5
0
25
50
75

Using a functional principal component regression term
Basis = tp, Method = svd, PVE = 0.999, k = 30
β(Location)
Location
1.5
1.0
0.5
0.0
0
25
50
75



The `fpc` function contstructs a functional principal component regression term. The options I used are the same as the ones in `lf` with the addition of the `pve` argument which controls the proportion of variation explained by the functional principal components and the `method` argument which controls the method of finding principal components. Adding more components produced smoother estimates in general. The default method is "svd" and the default for `penalize` is TRUE when the method is "svd" so the estimates are very smooth compared to the estimates when the method is "fpca.face". The `fpc` estimates are very sensitive to the choice of basis, knots, proprotion of variance explained, and method.

d. Let $B(t)$ be a set of cubic B-spline basis with k equally spaced knots in $[0,1]$. Assume a function $f(t)$ is represented as $f(t) = B(t)'\beta$; $\beta = (\beta_1, \dots, \beta_{k+4})'$. Define Δ, such that $\Delta\beta_j = \beta_j - \beta_{j-1}$, and $\Delta^2\beta_j = \Delta\Delta\beta_j = \beta_j - 2\beta_{j-1} + \beta_{j-2}$. A common penalization criterion for L^2 functions assumes a penalty of the form:

$$pen(\lambda) = \lambda \sum_{j=3}^{k+4} (\Delta^2\beta_j)^2.$$

Derive the matrices D and P, such that

$$pen(\lambda) = \lambda \sum_{j=3}^{k+4} (\Delta^2\beta_j)^2 = \beta' D' D \beta = \beta' P \beta.$$

Solution: We can write the penalty function as

$$
\begin{aligned}
pen(\lambda) &= \lambda \sum_{j=3}^{k+4} (\Delta^2 \beta_j)^2 \\
&= \lambda \sum_{j=3}^{k+4} (\beta_j - 2\beta_{j-1} + \beta_{j-2})^2 \\
&= \lambda \beta' \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ -2 & 1 & 0 & 0 & & \\ 1 & -2 & 1 & 0 & & \\ 0 & 1 & -2 & 1 & & \\ 0 & 0 & \ddots & \ddots & \ddots & \\ \vdots & \vdots & & & & \end{bmatrix}}_{(k+4)\times(k+1)} \underbrace{\begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & & \\ 0 & 0 & 1 & -2 & 1 & 0 & & \\ 0 & 0 & 0 & 1 & -2 & 1 & & \\ 0 & 0 & 0 & & \ddots & \ddots & \ddots & \\ \vdots & \vdots & & & & & & \end{bmatrix}}_{(k+1)\times(k+4)} \beta \\
&= \lambda \beta' D' D \beta \\
&= \lambda \beta' \underbrace{\begin{bmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ \vdots & \vdots & & & & & & & \\ 0 & \cdots & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix}}_{(k+4)\times(k+4)} \beta \\
&= \lambda \beta' P \beta.
\end{aligned}
$$

e. Let $B(t)$ be a set of B-spline basis with 20 equally spaced knots in $[0, 1]$. Derive an estimator of the regression function $\beta(t) = B(t)'\beta$, minimizing the PLS criterion, with penalty $\lambda\beta' P\beta$. Implement your estimation and compare to the relvant `refund` fit for several values of a smoothing parameter λ, starting at $\lambda = 0$.

Solution: To estimate using basis expansions, we are considering the model

$$Y_i = \alpha + \int \beta(t) X_i(t) dt + \epsilon_i$$

where $\beta(t)$ is estimated using a set of bases $B_1(t), \ldots, B_k(t)$

$$\beta(t) \approx \sum_{j=1}^{k} c_j B_j(t).$$

Then the model becomes

$$
\begin{aligned}
Y_i &= \alpha + \int \sum_{j=1}^{k} c_j B_j(t) X_i(t) dt + \epsilon_i \\
&= \alpha + \sum_{j=1}^{k} c_j \int B_j(t) X_i(t) dt + \epsilon_i \\
&= \alpha + \sum_{j=1}^{k} c_j \tilde{X}_{ij} + \epsilon_i
\end{aligned}
$$

and the estimates of $(\widehat{\alpha}, \widehat{c}) = [(1\tilde{X})'(1\tilde{X})]^{-1}(1\tilde{X})'Y$ and

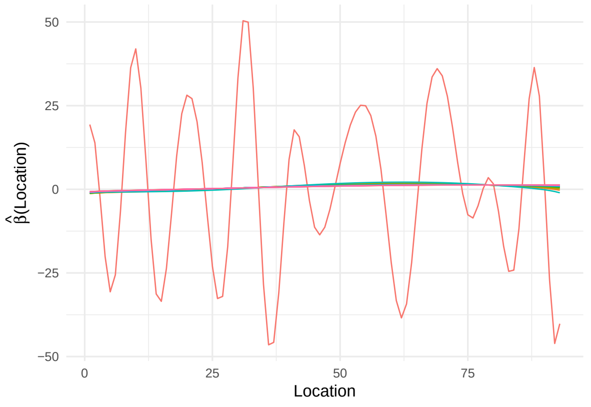
$$
\widehat{\beta}(t) = \sum_{j=1}^{k} \widehat{c}_j B_j(t).
$$

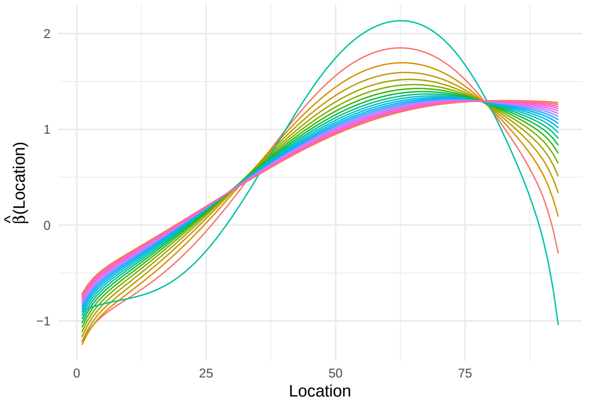
If we want to penalize the set of basis functions, then using the penalization matrix

$$
R = \begin{bmatrix} 0 & 0 \\ 0 & P \end{bmatrix}
$$

the estimates of $(\widehat{\alpha}, \widehat{c}) = [(1\tilde{X})'(1\tilde{X}) + \lambda R]^{-1}(1\tilde{X})'Y$

```r
# create a b-spline basis with 20 equally spaced knots in [1, 93]
B <- fda::create.bspline.basis(
  rangeval = c(1, 93),
  norder = 4,
  breaks = seq(1, 93, length.out = 22)
)

# discretize B to use in further analysis
B <- eval.basis(seq(1, 93, length.out = 93), B)

# construct D matrix
D <- diag(1, nrow = 22, ncol = 24)
for (j in 2:23) {
  D[j - 1, j] <- -2
}
for (j in 2:24) {
  D[j - 2, j] <- 1
}

# construct P matrix using D'D
P <- t(D) %*% D

# construct R matrix as the penalization matrix bound with a column and row
# of zeros (we are not penalizing the intercept)
R <- rbind(rep(0, 24), P) %>% cbind(rep(0, 25), .)

# calculate X_ij as \int B_j(t) X_i(t) dt
xtilde <- matrix(nrow = nrow(cca), ncol = ncol(B))
for (j in 1:ncol(B)) {
  for (i in 1:nrow(cca)) {
    xtilde[i, j] <- trapz(
      x = seq(1, 93, length.out = 93),
      y = B[, j] * as.matrix(cca[i, ])
    )
  }
}
xtilde <- cbind(rep(1, 66), xtilde)

y <- DTI1$pasat[DTI1$visit == 1]

lambda <- seq(0, 10000, by = 500)
names(lambda) <- paste0(
  "lambda_",
  seq(0, 10000, by = 500)
)
c_hat <- map_dfc(
  lambda,
  ~ data.frame(solve(t(xtilde) %*% xtilde + .x * R, t(xtilde) %*% y))
  ) %>%
  setNames(names(lambda))
```





We can compare these estimates to the `lf` estimates in (c) which are a straight line. As λ increases, the estimate of $\beta(t)$ approaches a straight line. However, even for very large λ, the estimate is not a straight line.

f. Let $\Phi(t)$ be a set of FPCs for $x_i(t)$. Derive an estimator of the regression function $\beta(t) = \Phi(t)'\eta$, minimizing the LS criterion, for a fixed number of FPCs. Implement your estimation and compare to the relevant `refund` fit for several numbers of FPCs.

Solution: If we want to use the functional principal components, we use the KL expansion

$$X_i(t) \approx \mu(t) + \sum_{j=1}^{p} \eta_{ij}\nu_j(t)$$

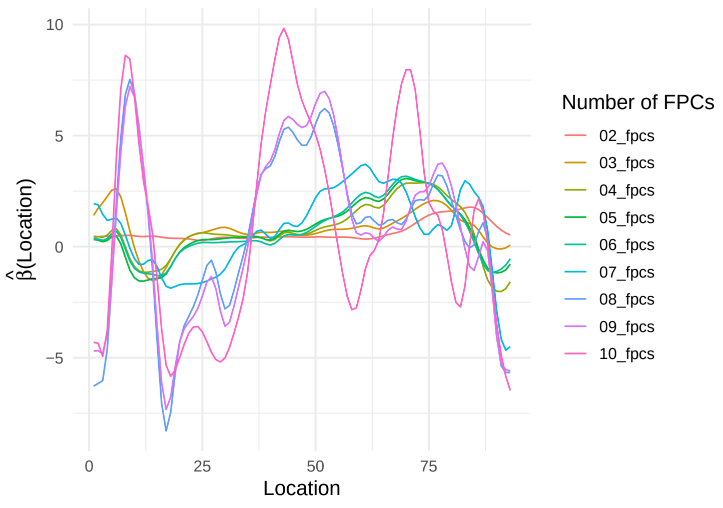
and the model becomes

$$
\begin{aligned}
Y_i &= \alpha + \int \beta(t) \left\{ \mu(t) + \sum_{j=1}^{p} \widehat{\eta}_{ij} \nu(t) \right\} dt + \epsilon_i \\
&= \alpha + \int \beta(t)\widehat{\mu}(t)dt + \sum_{j=1}^{p} \widehat{\eta}_{ij} \int \beta(t)\widehat{\nu}_j(t)dt + \epsilon_i \\
&= \beta_0 + \sum_{j=1} \widehat{\eta}_{ij}\beta_j + \epsilon_i.
\end{aligned}
$$

The estimate $\widehat{\beta} = [(1\eta)'(1\eta)]^{-1}(1\eta)'Y$ and

$$
\begin{aligned}
\widehat{\beta}(t) &= \sum_{j=1}^{p} \widehat{\beta}_j \widehat{\nu}_j(t) \\
\widehat{\alpha} &= \widehat{\beta}_0 - \sum_{j=1}^{p} \widehat{\beta}_j \int \widehat{\nu}_j(t)\widehat{\mu}(t)dt.
\end{aligned}
$$

```r
fpcs <- seq(2, 10)
names(fpcs) <- paste0(str_pad(fpcs, width = 2, pad = "0", side = "left"),
                      "_fpcs")

beta_hat_t <- map_dfr(
  fpcs,
  function(x) {
    phi <- fpca.face(as.matrix(cca), npc = x)$efunctions
    eta <- fpca.face(as.matrix(cca), npc = x)$scores

    eta <- cbind(rep(1, 66), eta)

    y <- DTI1$pasat[DTI1$visit == 1]

    beta_hat <- solve(t(eta) %*% eta, t(eta) %*% y)

    data.frame(
      "t" = seq(1, 93, length.out = 93),
      "beta_hat_t" = phi %*% as.matrix(beta_hat[-1])
    )
  },
  .id = "npcs"
)
```



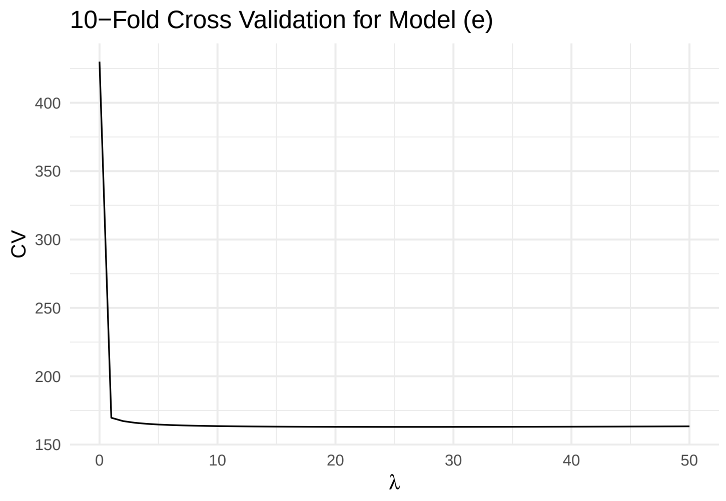
We can compare this estimate to the `fpc` estimates in (c). The estimates are not very similar since the `fpc` function smooths the data using B-splines before conducting FPCA on the covariance matrix.

g. For the models in (e) and (f), determine the optimal value of λ using 10-fold-CV.

Solution:

```
folds <- as.numeric(
  cut(
    sample(nrow(cca)),
    breaks = c(0, 7, 14, 21, 28, 35, 42, 48, 54, 60, 66)
  )
)
cca$fold <- folds


lambdas <- seq(0, 50, by = 1)
names(lambdas) <- paste0(
  "lambda_",
  seq(0, 50, by = 1)
)

s_n <- vector("numeric", length(lambdas))
for(lambda in lambdas) {
  s_n[match(lambda, lambdas)] <- 0
  for (fold in seq(1, 10)) {
    cca_i <- cca[cca$fold != fold, ]

    xtilde <- matrix(nrow = nrow(cca_i), ncol = ncol(B))
    for (j in 1:ncol(B)) {
      for (i in 1:nrow(cca_i)) {
        xtilde[i, j] <- trapz(
          x = seq(1, 93, length.out = 93),
          y = B[, j] * as.matrix(cca_i[i, -94])
        )
      }
    }
    xtilde <- cbind(rep(1, nrow(cca_i)), xtilde)
```

```r
    y <- data.frame(
      y = DTI1$pasat[DTI1$visit == 1],
      group = folds
    ) %>%
      filter(group != fold) %>%
      pull(y)


    c_hat <- solve(t(xtilde) %*% xtilde + lambda * R, t(xtilde) %*% y)

    x_i <- cca[cca$fold == fold, -94]
    y_i <- data.frame(
      y = DTI1$pasat[DTI1$visit == 1],
      group = folds
    ) %>%
      filter(group == fold) %>%
      pull(y)

    y_hat_i <- vector("numeric", nrow(x_i))

    for (sub in 1:nrow(x_i)) {
      y_hat_i[sub] <- c_hat[1] + trapz(
        x = seq(1, 93, length.out = 93),
        y = (B %*% as.matrix(c_hat[2:25])) * as.matrix(t(x_i[sub, ]))
      )
    }

    s_n[match(lambda, lambdas)] <- s_n[match(lambda, lambdas)] +
      mean(y_i - y_hat_i)^2
  }
}
```



The optimal λ in model (e) is 26.

h. Attempt the derivation of a bootstrap procedure aimed at the construction of Confidence bands around the function $\beta(t)$, for both your implementations in (e) and (f), and compare your findings with the relvant results obtained through `refund`.

Solution: