BIOS 7720: Applied Functional Data Analysis

Leture 5: (f)PCA

Andrew Leroux

March 18, 2021

Roadmap

1. Review PCA for multivariate data
2. Introduce fPCA

PCA: Problem Set Up

- Suppose we have observe N iid realizations of $\boldsymbol{x} \in \mathbb{R}^p$
- Typically used for large p
- Considers (orthogonal) linear combinations of $\boldsymbol{x}$ that explain maximal variation in the data
- Used for (unsupervised) dimensionality reduction
  - Clustering
  - Feature extraction
- Useful when $\text{cor}(x_i, x_j) \neq 0$ for $i \neq j$

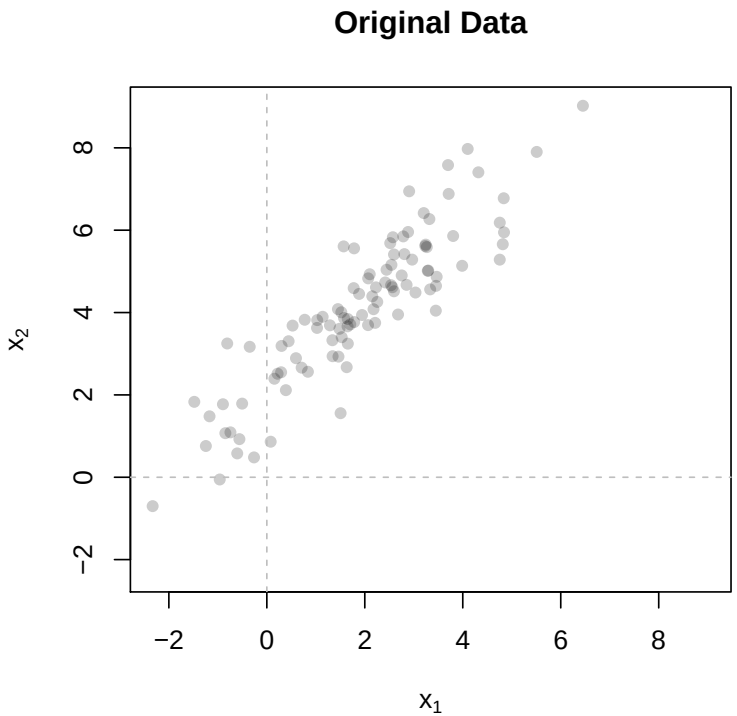
PCA: Problem Set Up

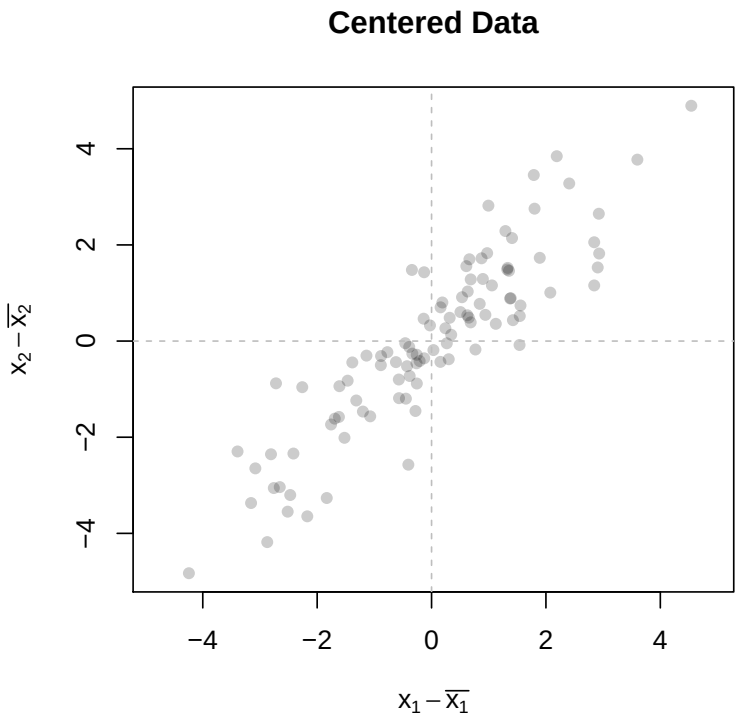
- Observe $\boldsymbol{x}_i \in \mathbb{R}^p$ for $x = 1, \ldots, N$
- Goals: find linear combinations (principal directions) of $\boldsymbol{X}$ that
  - maximize variance in the data
  - minimize the data reconstruction error^2
- Subject to the constraint that principal directions are orthogonal
- So, to find the first such vector, $\boldsymbol{v}_1$, find $\boldsymbol{v}_1$ that
  - maximizes $\text{Var}(\boldsymbol{X}\boldsymbol{v}_1)$ subject to
  - $\boldsymbol{v}_1^t \boldsymbol{v}_1 = 1$ (normalization)
- Repeat for $v_2, \ldots, v_p$ where $\boldsymbol{v}_m^t \boldsymbol{v}_n = 0$ for $n \neq m$
- Solution turns out to be the eigenvectors of the sample covariance matrix

PCA Example: 2 Dimensions

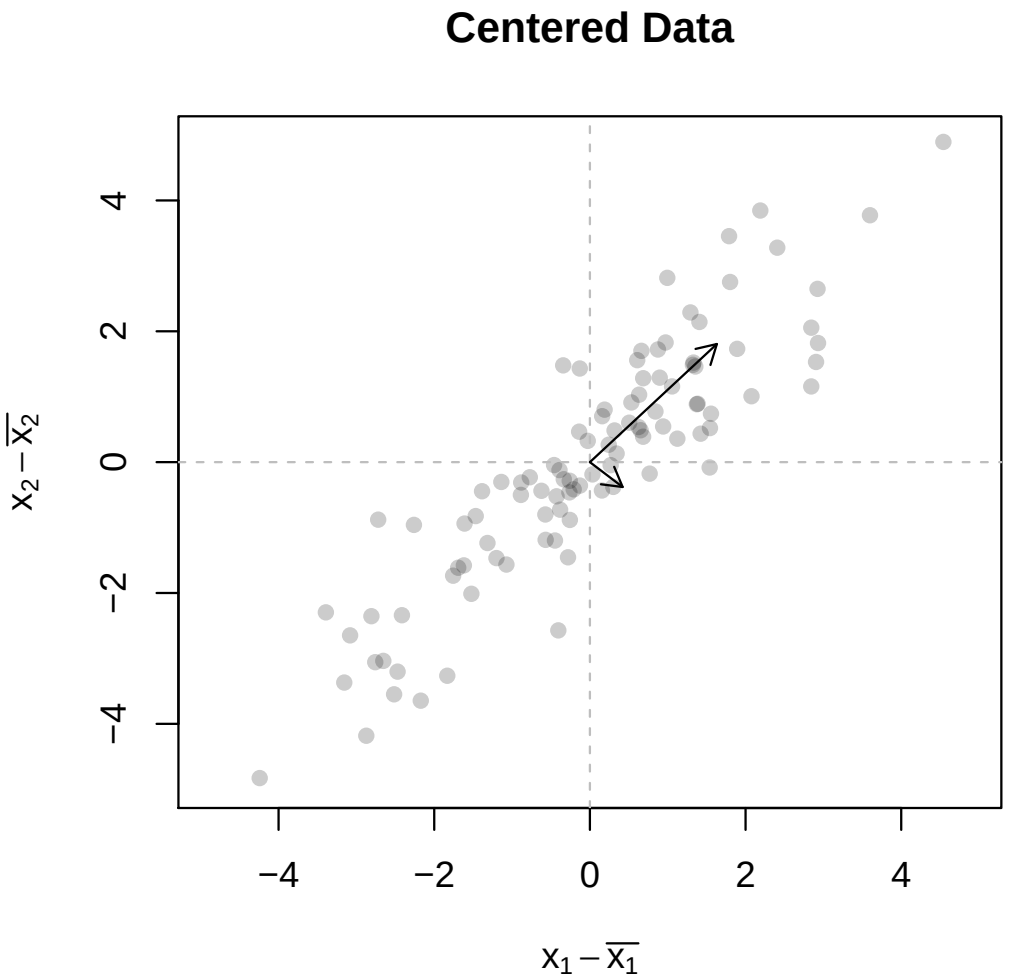

```r
set.seed(100)
N <- 100              ## number of samples
p <- 2                ## number of predictors to simulate
mu_vec <- c(2,4)      ## vector of means
sd_vec <- c(2,2)      ## vector of standard deviations
rho_mat       <- matrix(0.9 , p, p)          ## correlation matrix
diag(rho_mat) <- 1
Sigma_mat  <- tcrossprod(sd_vec) * rho_mat ## variance/covariance matrix
## simulate the data
X <- mvtnorm::rmvnorm(N, mean=mu_vec, sigma=Sigma_mat, method="svd")
## column center the data
col_mns <- colMeans(X)
X_cn    <- sweep(X, MARGIN=2, STATS=col_mns, FUN="-")
## do pca using stats::prcomp()
fit_pca <- prcomp(X, center=TRUE, scale=FALSE)
```

PCA Example: 2 Dimensions

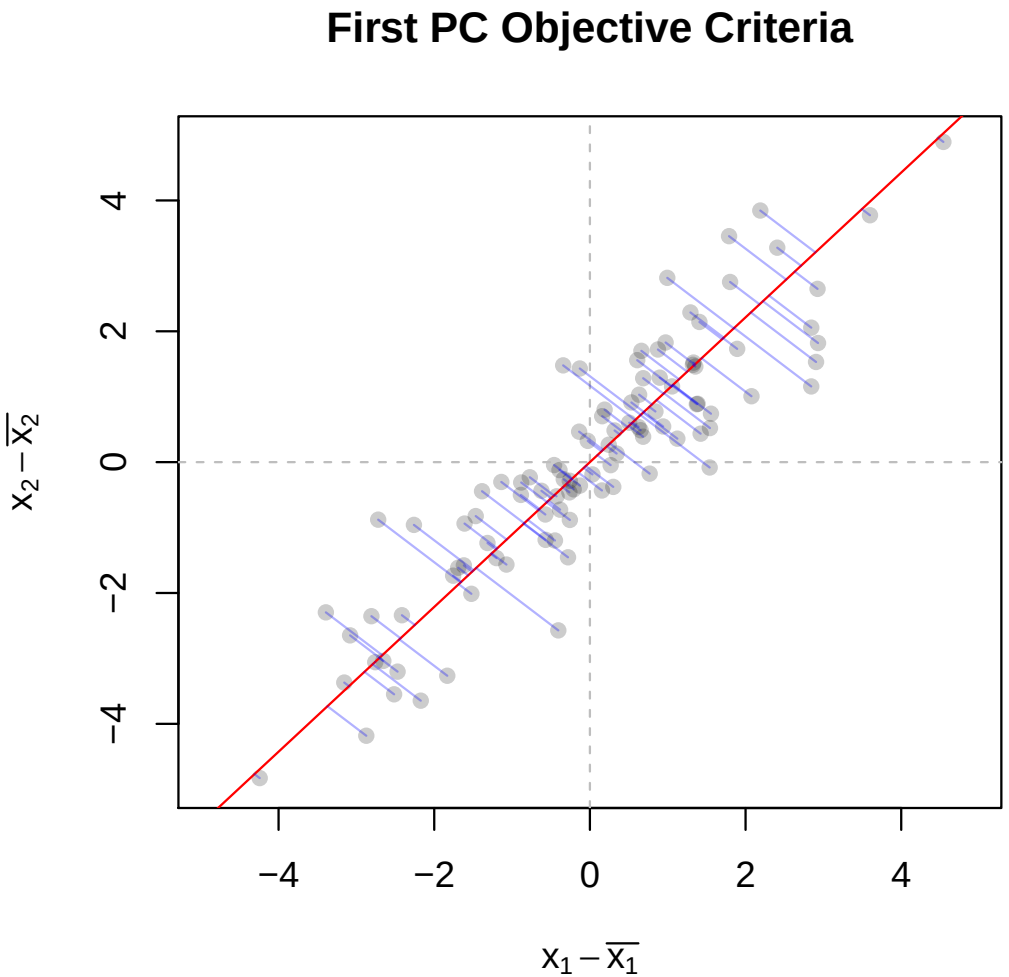




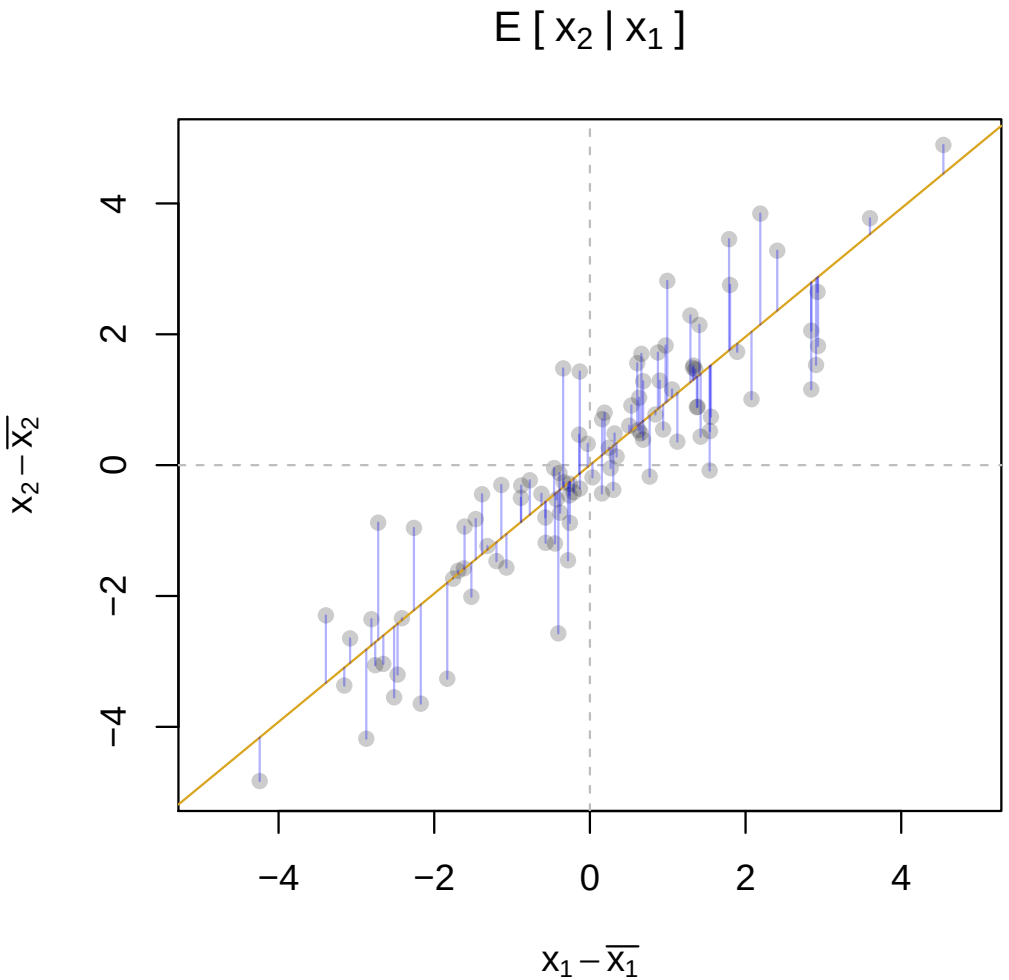
PCA Example: 2 Dimensions



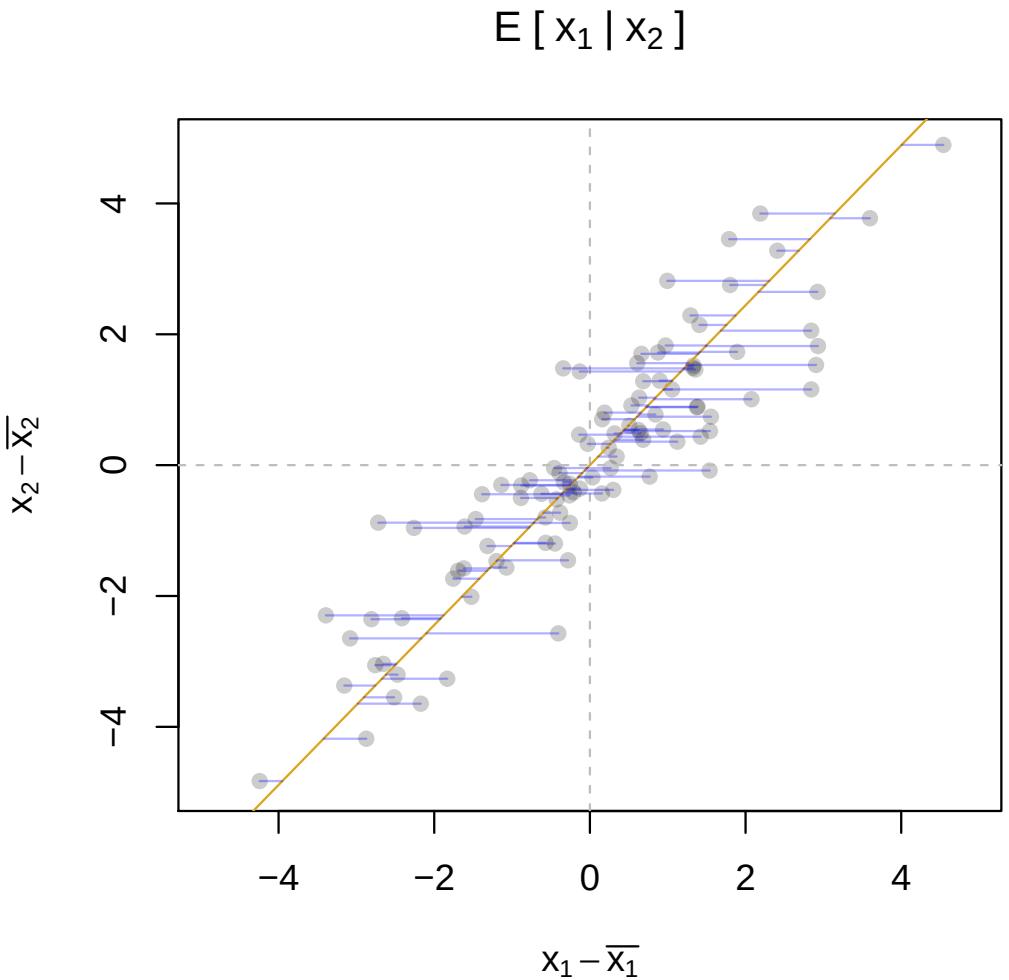
PCA Example: 2 Dimensions



Linear Regression: $E[x_2|x_1] = \beta x_1$



Linear Regression: $E[x_1|x_2] = \beta x_2$



PCA: Eigendecomposition of $\hat{\Sigma}$

- Observe $\boldsymbol{x}_i \in \mathbb{R}^P$ for $x = 1, \ldots, N$
- Combine vectors into a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^t$
- Center each of the p variables $\boldsymbol{X}_{\text{cn}} = (\boldsymbol{I} - 11^t)\boldsymbol{X}$
- Estimate covariate $\hat{\Sigma} = (N-1)^{-1}\boldsymbol{X}_{\text{cn}}^t\boldsymbol{X}_{\text{cn}}$
- Get the eigen-decomposition of $\hat{\Sigma} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{V}^t$
  - columns of $\boldsymbol{V}$ are the principal directions ("loadings")
  - $\boldsymbol{S}$ is diagonal with entries λ_p
  - $\boldsymbol{\xi}_p = \boldsymbol{X}_{\text{cn}}\boldsymbol{v}_p$ are the p^{th} principal component scores

PCA: Eigendecomposition of $\hat{\Sigma}$

- The loadings p^{th} loading (p^{th} column of $\boldsymbol{V}$)
  - Determine how much "important" a variable is for the p^{th} principal direction
  - Used for interpretation
  - Sometimes these group nicely, sometimes they don't
- The scores determine how highly a particular observation $\boldsymbol{x}_i$
- The λ_p are measure of variability

$$\begin{aligned}\hat{\text{Var}}(X\boldsymbol{v}_1) &= \boldsymbol{v}_1^t \hat{\text{Var}}(\boldsymbol{X})\boldsymbol{v}_1 \\ &= \boldsymbol{v}_1^t \boldsymbol{V}\boldsymbol{S}\boldsymbol{V}^t \boldsymbol{v}_1 \\ &= \lambda_1\end{aligned}$$

- "Reconstructed" data are $\tilde{\boldsymbol{x}}_i^k = \sum_{p=1}^{K} \xi_{ip}\boldsymbol{v}_p$

PCA: SVD of $\boldsymbol{X}_{\text{cn}}$

- It turns out we can obtain all the components of PCA from SVD on the centered data matrix $\boldsymbol{X}_{\text{cn}}$
- Recall, SVD decomposes $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{R}\boldsymbol{V}^t$, $\boldsymbol{U}, \boldsymbol{V}$ are orthogonal

$$\begin{aligned} \boldsymbol{X}_{\text{cn}} &= \boldsymbol{U}\boldsymbol{R}\boldsymbol{V}^t \\ \boldsymbol{X}_{\text{cn}}^t \boldsymbol{X}_{\text{cn}} &= \boldsymbol{V}\boldsymbol{R}\boldsymbol{U}^t\boldsymbol{U}\boldsymbol{R}\boldsymbol{V}^t \\ &= \boldsymbol{V}\boldsymbol{R}^2\boldsymbol{V}^t \end{aligned}$$

- $\boldsymbol{R}^2$ is a diagonal matrix with $\lambda_p^2/(N-1)$ on the diagonal
- SVD is more numerically stable, faster than eigendecomposition of $\boldsymbol{\Sigma}$

PCA Example Revisited: 2 Dimensions


```
str(fit_pca)

## List of 5
##  $ sdev    : num [1:2] 2.433 0.567
##  $ rotation: num [1:2, 1:2] 0.67 0.742 0.742 -0.67
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "PC1" "PC2"
##  $ center  : num [1:2] 1.91 4.13
##  $ scale   : logi FALSE
##  $ x       : num [1:100, 1:2] -0.742 1.56 0.818 0.253 -2.332 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:2] "PC1" "PC2"
##  - attr(*, "class")= chr "prcomp"

fit_pca$rotation

##            PC1        PC2
## [1,] 0.6704739  0.7419332
## [2,] 0.7419332 -0.6704739
```

PCA in *R*: Simulating Data


```r
set.seed(982734)
N <- 1000             ## number of samples
p <- 500              ## number of predictors to simulate
mu_vec <- rep(0, p) ## vector of means
sd_vec <- sqrt(1:p) ## vector of standard deviations
rho_mat       <- matrix(0.9, p, p)          ## correlation matrix
diag(rho_mat) <- 1
Sigma_mat  <- tcrossprod(sd_vec) * rho_mat ## variance/covariance matrix
## simulate the data
X <- mvtnorm::rmvnorm(N, mean=mu_vec, sigma=Sigma_mat, method="svd")
```

PCA in *R*: *stats::prcomp()*


```
## do pca using stats::prcomp()
pca_fit <- prcomp(X, center=TRUE)
vars_pca <- pca_fit$sdev^2
dirs_pca <- pca_fit$rotation
str(pca_fit)

## List of 5
##  $ sdev    : num [1:500] 320.45 9.71 9.64 9.49 9.44 ...
##  $ rotation: num [1:500, 1:500] -0.00285 -0.00395 -0.00496 -0.00562 -0.00622 ..
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:500] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : num [1:500] 0.000723 0.039661 -0.005911 0.035688 0.003728 ...
##  $ scale   : logi FALSE
##  $ x       : num [1:1000, 1:500] -72.2 -223 -512.1 -315.1 246.6 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:500] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

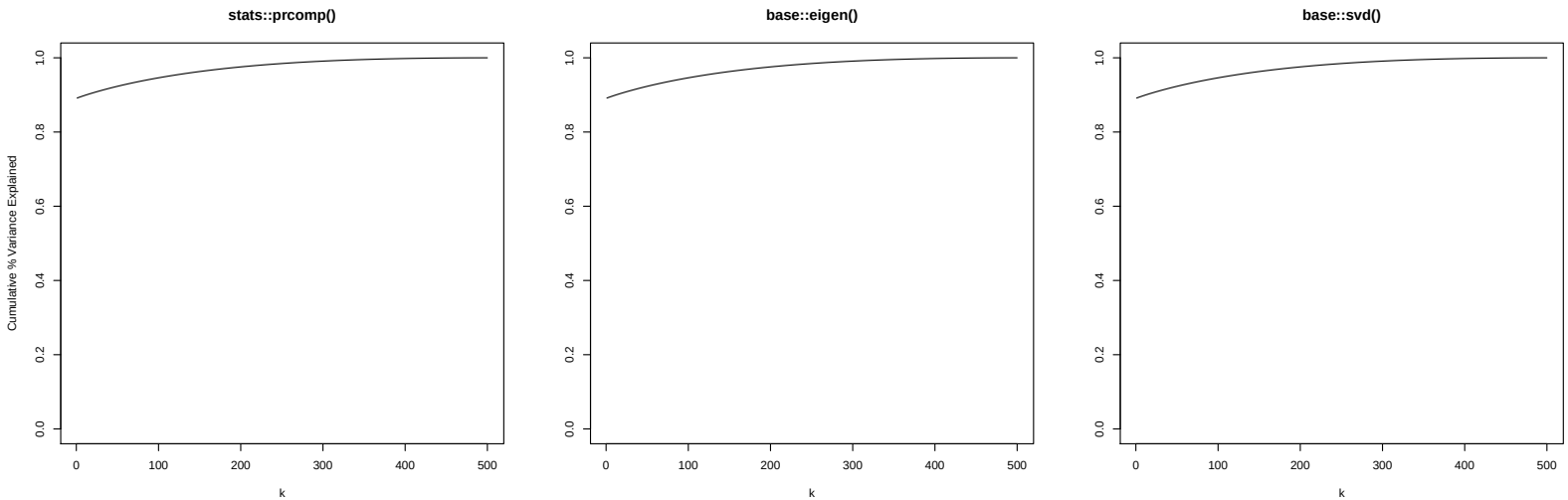
PCA in *R*: *base::eigen()*


```r
## do pca eigen decomposition of the covariance matrix
X_mn      <- colMeans(X)
X_cn      <- sweep(X, MARGIN=2, STATS=X_mn, FUN="-")
Sigma_hat <- crossprod(X_cn)/(N-1)
eigen_fit <- eigen(Sigma_hat)
## sort by variance
inx_srt    <- order(eigen_fit$values, decreasing=TRUE)
dirs_eigen <- eigen_fit$vectors[inx_srt,]
vars_eigen <- eigen_fit$values[inx_srt]
```

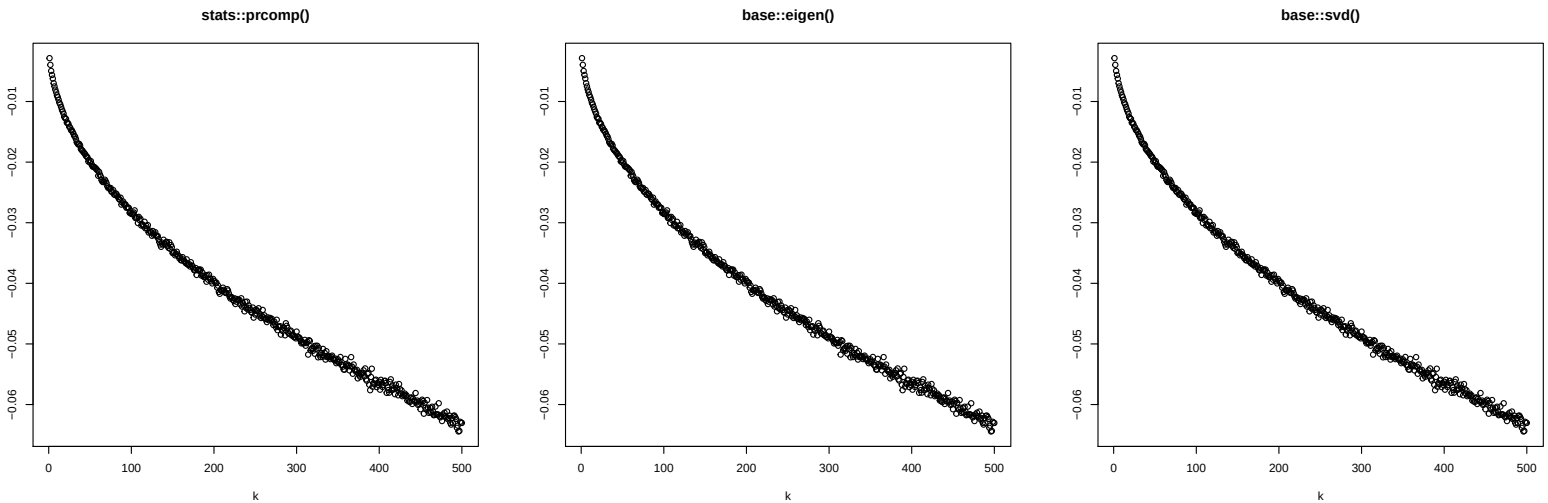
PCA in *R*: *base::svd()*


```
## do pca eigen decomposition of the covariance matrix
svd_fit <- svd(X_cn)
## sort by variance
inx_srt <- order(svd_fit$d, decreasing=TRUE)
## do pca using singular value decomposition of X
dirs_svd <- svd_fit$v[inx_srt,]
vars_svd <- svd_fit$d[inx_srt]^2/(N-1)
```

PCA in *R*: Scree Plots



PCA in *R*: First Principal Direction

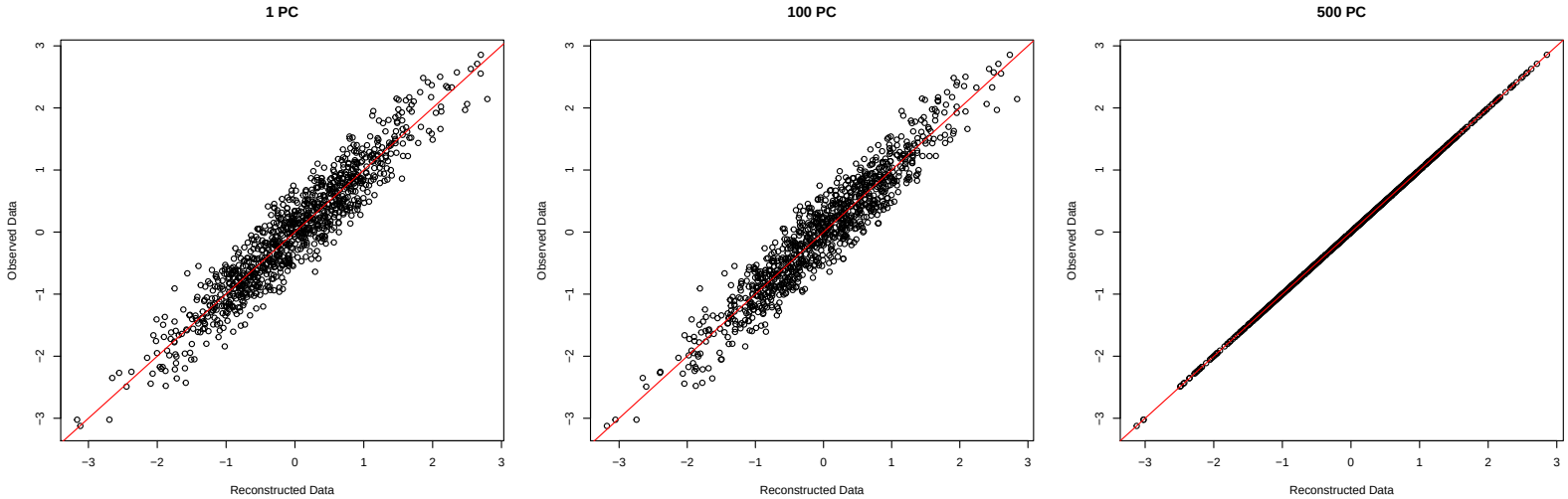


PCA in *R*: Predictions Using The First k PCs


```r
Ks <- c(1, 50, 100, 500)
ests_k <- vector("list", length(Ks))
for(k in seq_along(Ks)){
  ests_k[[k]] <- pca_fit$x[,1:Ks[k]] %*% t(pca_fit$rotation[,1:Ks[k]])
}
(mse_k <- vapply(ests_k, function(x) mean((x-X_cn)^2), numeric(1)))

## [1] 2.495128e+01 1.767536e+01 1.241312e+01 7.539309e-28
```

PCA in *R*: Predictions Using The First *k* PCs



PCA: Summary

- PCA is **unsupervised**
- To scale predictors or not?
- Interpretability versus parsimony
- Translation across different samples

Functional Principal Component Analysis (fPCA)

- Observe N realizations of $x(s)$ a function

$$\text{Cov}(x(s), x(u)) = \Sigma(s, u); \quad s, u \in \mathcal{S}$$

- $x(s)$ theoretically continuous, observed discretely

$$\begin{aligned} x_{\text{obs}}(s) &= \mu(s) + x_d(s) + \epsilon(s) \\ &= \mu(s) + \sum_{k=1}^{\infty} \xi_k \phi_k(s) + \epsilon(s) \quad \text{Karhunen-Loève Theorem} \\ &\approx \mu(s) + \sum_{k=1}^{K} \xi_k \phi_k(s) + \epsilon(s) \end{aligned}$$

- $\mu(s)$ is the mean function
- $\text{Cov}(\xi_l, \xi_k) = 0$ for $l \neq k$, ϕ_k are orthogonal

Functional Principal Component Analysis (fPCA)

- Basically, PCA with smoothness on the eigenfunctions
- Principal directions are $\phi_k(s)$
- Scores are $\xi_k = \int_S (x(s) - \mu(s))\phi_k(s)ds$
- Normalized eigenfunctions $\int \phi_k(s)\phi_k(s) = 1$
- Orthogonality $\int \phi_j(s)\phi_k(s) = 0$ for $j \neq k$
- How to choose K?
  - Fix K a-priori
  - Percent variance explained (e.g. 95%)

Uses of (fPCA)

- Smoothing
- Imputation
- Clustering
- Regression

fPCA

- Estimation approaches
  - Smoothing of eigenfunctions from “raw" data (columns of $\boldsymbol{V}$)
  - Data smoothing
  - Covariance smoothing
  - Maximum likelihood `partially observed data.`
- Some *R* implementations
  - *refund*:
    - *fpca.face()* (fast covariance smoothing)
    - *fpca.sc()* (covariance smoothing)
    - *fpca2s()* (fast SVD, smoothing eigenvectors)
    - *fpca* (SVD, smoothing eigenvectors)
  - *face*: *face.sparse()* (covariance smoothing, sparse/irregular data)
  - *fdapace*: *FPCA()* (covariance smoothing)
  - *fpca*: *fpca.mle()* (ML)

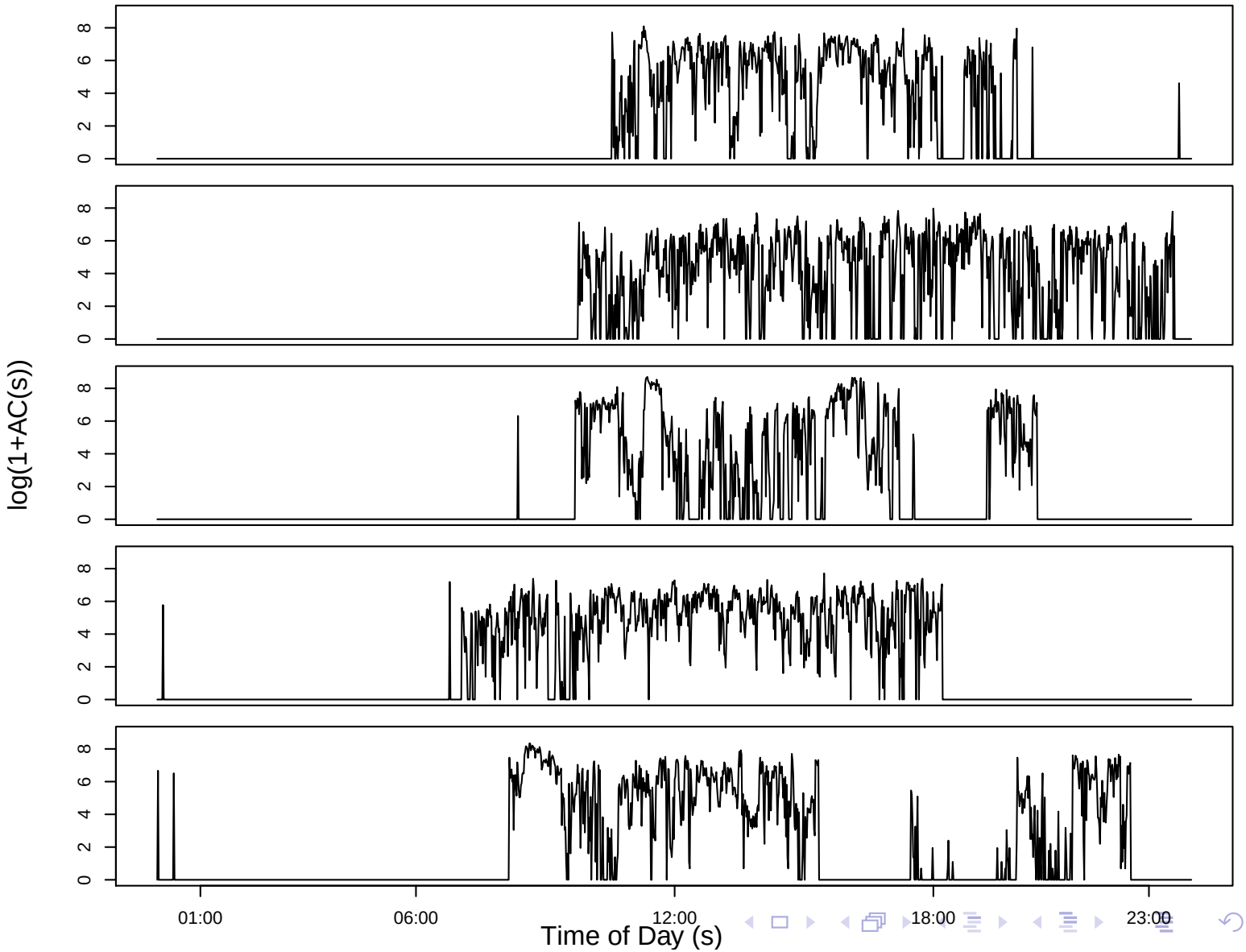
fPCA: NHANES PA Data

- Let's fit fPCA to the NHANES activity count data
- For now, ignore distributional assumptions
- Here we'll use *refund::fpca.face()*
  - Extremely fast
  - Intuitive inputs/outputs
- Note: *fpca.face()* and most FDA software assume the data are observed on $[0, 1]$
  - Can be changed (e.g. via the "argvals" argument)
  - Need to be careful with how this affects ϕ_k, λ_k

fPCA: NHANES PA Data


```
library("tidyverse");library("here");  ## data manuplation/path setting
library("refund")                      ## fpca
## load the data
df <- read_rds(here::here("data","data_processed","NHANES_AC_processed.rds"))
## subset to "good" days of data
df <-
  df %>%
  filter(good_day %in% 1)
## get a matrix of activity counts
X <- as.matrix(df[,paste0("MIN",1:1440)])
## impute 0s for the few NA values
X[is.na(X)] <- 0
## log transform
lX <- log(1+X)
```

fPCA: NHANES PA Data

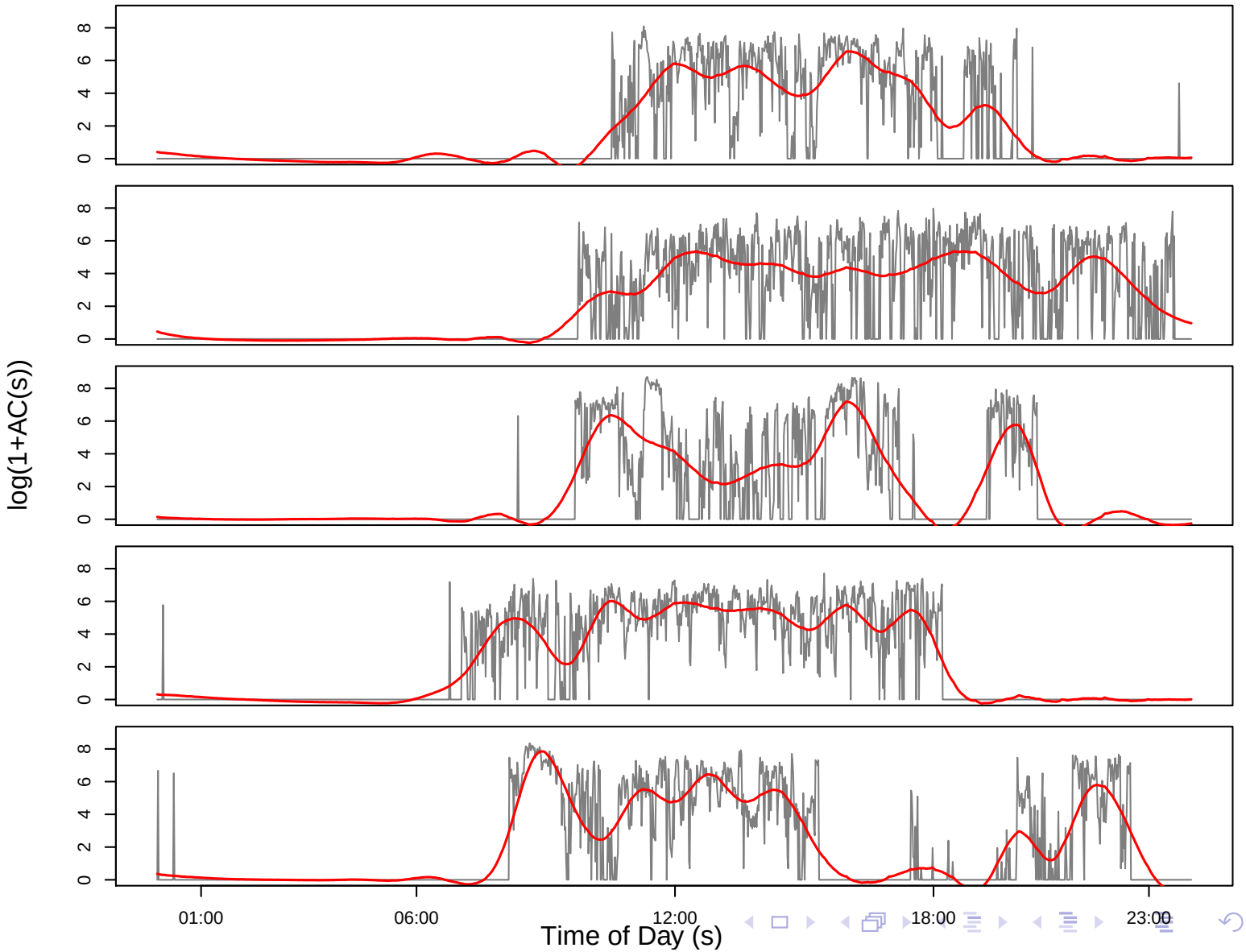


fPCA: NHANES PA Data


```
fit_fpca <- fpca.face(Y=lX, pve=0.95, knots=30)
str(fit_fpca)

## List of 7
##  $ Yhat       : num [1:65777, 1:1440] -0.6239 0.1877 -0.0472 1.8408 0.6017 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Y          : num [1:65777, 1:1440] -0.484 -0.484 -0.484 3.846 -0.484 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:1440] "MIN1" "MIN2" "MIN3" "MIN4" ...
##  $ scores     : num [1:65777, 1:21] 33.4 -25.6 21.2 -36.2 -44.7 ...
##  $ mu         : num [1:1440] 0.484 0.473 0.463 0.456 0.45 ...
##  $ efunctions: num [1:1440, 1:21] -0.0104 -0.0103 -0.0103 -0.0103 -0.0102 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ evalues    : num [1:21] 946 779 343 242 211 ...
##  $ npc        : int 21
##  - attr(*, "class")= chr "fpca"
```

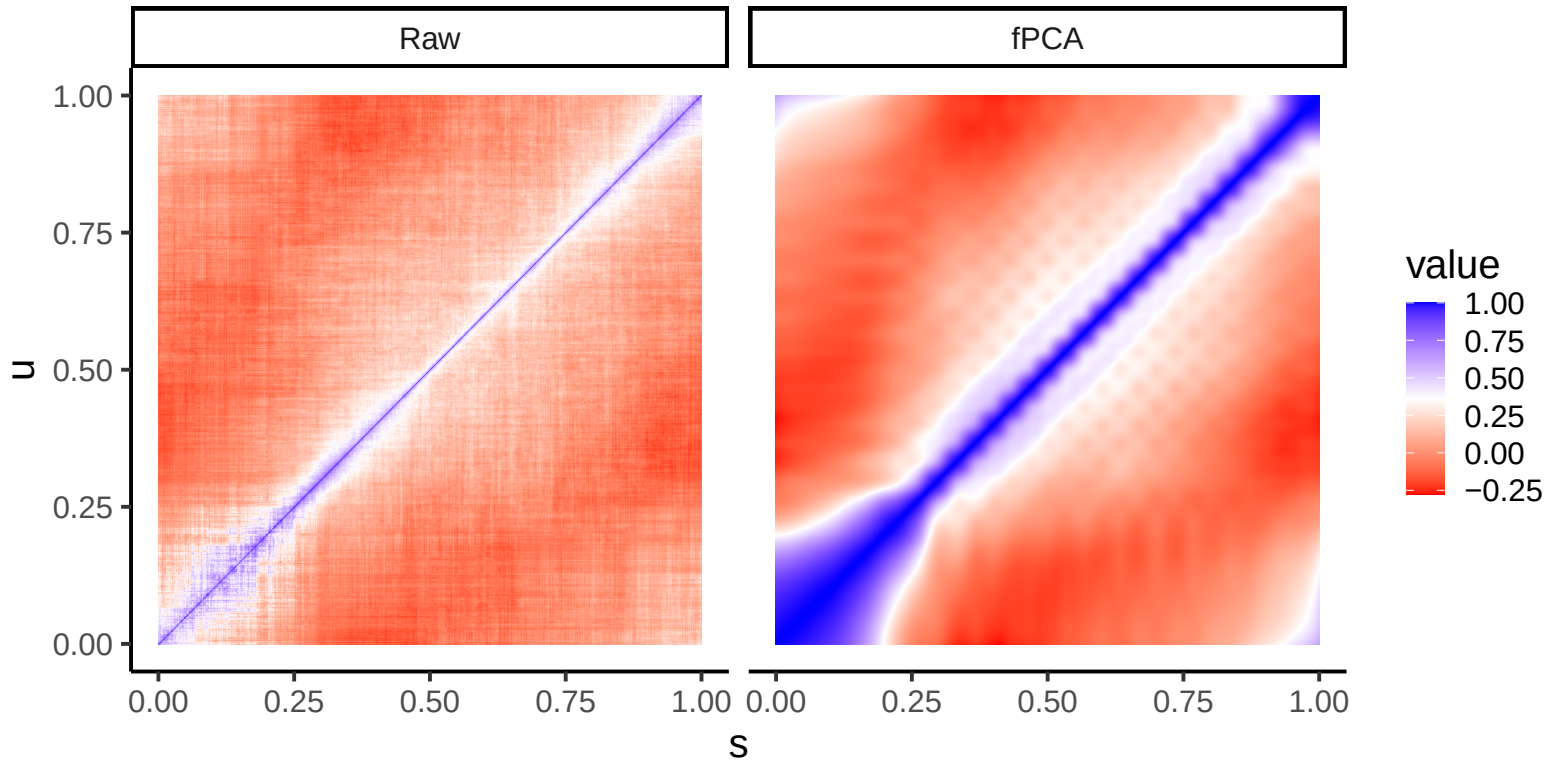
fPCA: NHANES PA Data, Predictions/Fitted Values



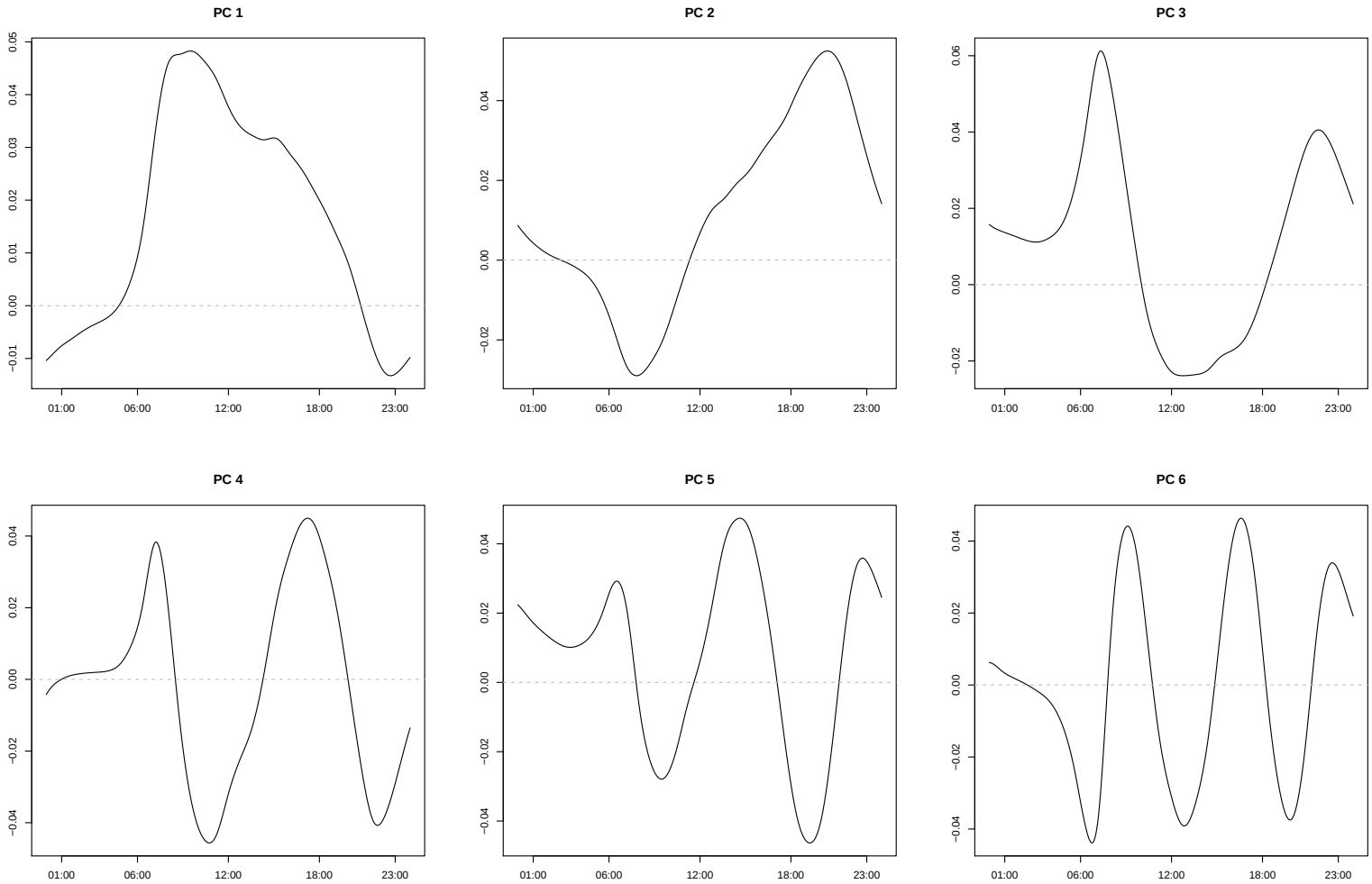
fPCA: NHANES PA Data, Covariance


```
library("ggplot2")
inx_cov_samp <- sample(1:nrow(lX), size=500, replace=FALSE)
cov_raw  <- cov(lX[inx_cov_samp,])
cor_raw  <- cov2cor(cov_raw)
cov_fpca <- fit_fpca$efunctions %*% diag(fit_fpca$evalues) %*% t(fit_fpca$efunctio
cor_fpca <- cov2cor(cov_fpca)
tind <- seq(0,1,len=1440)
df_plt <- data.frame(cor_raw = as.vector(cor_raw),
                     cor_fpca = as.vector(cor_fpca),
                     s = rep(tind, length(tind)),
                     u = rep(tind, each=length(tind)))
plt_cov <-
  df_plt %>%
  pivot_longer(cols=c("cor_raw","cor_fpca")) %>%
  mutate(name = factor(name, levels=c("cor_raw","cor_fpca"), labels=c("Raw","fPCA"
  ggplot() +
  geom_raster(aes(s,u,fill=value)) + facet_grid(~name) +
  scale_fill_gradientn(colours=c("red","white","blue")) +
  theme_classic(base_size=18)
```

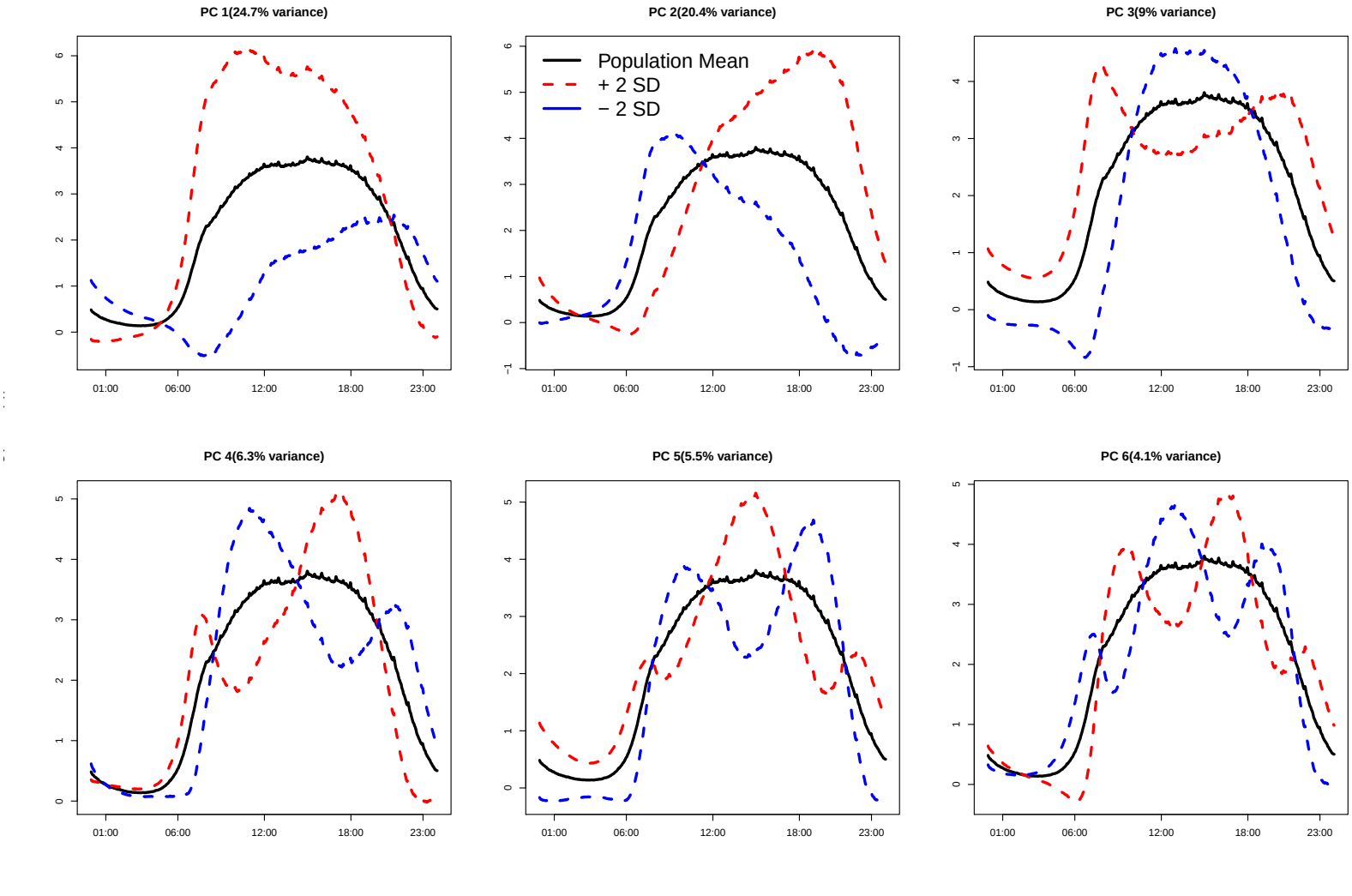
fPCA: NHANES PA Data, Correlation



fPCA: NHANES PA Data, Eigenfunctions



fPCA: NHANES PA Data, Eigenfunctions



Next Class

- Provide details on several estimation approaches
  - Covariance smoothing
  - Maximum likelihood
- In-class exercises using fPCA