

# Principal Component Analysis

---

Max Turgeon

STAT 7200–Multivariate Statistics

# Objectives

- Discuss population and sample principal component analysis.
- Explain how PCA can be used for visualizing high-dimensional data.
- Present a geometric view on PCA.
- Discuss asymptotic results about PCA.
- Explain how PCA can be used for model selection.

## Population PCA i

- PCA: Principal Component Analysis
- Dimension reduction method:
  - Let  $\mathbf{Y} = (Y_1, \dots, Y_p)$  be a random vector with covariance matrix  $\Sigma$ . We are looking for a transformation  $h : \mathbb{R}^p \rightarrow \mathbb{R}^k$ , with  $k \ll p$  such that  $h(\mathbf{Y})$  retains “as much information as possible” about  $\mathbf{Y}$ .
  - In PCA, we are looking for a **linear transformation**  $h(y) = w^T y$  with **maximal variance** (where  $\|w\| = 1$ )
  - More generally, we are looking for  $k$  linear transformations  $w_1, \dots, w_k$  such that  $w_j^T \mathbf{Y}$  has maximal variance and is uncorrelated with  $w_1^T \mathbf{Y}, \dots, w_{j-1}^T \mathbf{Y}$ .

## Population PCA ii

- First, note that  $\text{Var}(w^T \mathbf{Y}) = w^T \Sigma w$ . So our optimisation problem is

$$\max_w w^T \Sigma w, \quad \text{with } w^T w = 1.$$

- We will solve this constrained optimisation problem using the method of **Lagrange multipliers**.

## Lagrange Multipliers i

Let  $f, g : \mathbb{R}^p \rightarrow \mathbb{R}$  be differentiable functions. Let  $c$  be a real number, and assume that we are looking for an extremum point  $\mathbf{y}_0 \in \mathbb{R}^p$  of  $f$  subject to the constraint  $g(\mathbf{y}_0) = 0$ . If such an extremum point exists, then there exists a scalar  $\lambda$  such that

$$\nabla f(\mathbf{y}_0) = \lambda \nabla g(\mathbf{y}_0).$$

## Lagrange Multipliers ii

- In practice, this means that we can replace the constrained optimisation problem in  $p$  variables

$$\max_{\mathbf{y}} f(\mathbf{y}), \quad \text{such that } g(\mathbf{y}) = c,$$

with the *unconstrained* optimisation problem in  $p + 1$  variables

$$\max_{\mathbf{y}, \lambda} \{f(\mathbf{y}) - \lambda(g(\mathbf{y}) - c)\}.$$

## Population PCA (cont'd) i

- From the theory of Lagrange multipliers, we can look at the *unconstrained* problem

$$\max_{w,\lambda} w^T \Sigma w - \lambda(w^T w - 1).$$

- Write  $\phi(w, \lambda)$  for the function we are trying to optimise. We have

$$\begin{aligned}\frac{\partial}{\partial w} \phi(w, \lambda) &= \frac{\partial}{\partial w} w^T \Sigma w - \lambda(w^T w - 1) \\ &= 2\Sigma w - 2\lambda w;\end{aligned}$$

$$\frac{\partial}{\partial \lambda} \phi(w, \lambda) = w^T w - 1.$$

## Population PCA (cont'd) ii

- From the first partial derivative, we conclude that

$$\Sigma w = \lambda w.$$

- From the second partial derivative, we conclude that  $w \neq 0$ ; in other words,  $w$  is an eigenvector of  $\Sigma$  with eigenvalue  $\lambda$ .
- Moreover, at this stationary point of  $\phi(w, \lambda)$ , we have

$$\text{Var}(w^T \mathbf{Y}) = w^T \Sigma w = w^T (\lambda w) = \lambda w^T w = \lambda.$$

- In other words, to maximise the variance  $\text{Var}(w^T \mathbf{Y})$ , we need to choose  $\lambda$  to be the *largest* eigenvalue of  $\Sigma$ .

## Population PCA (cont'd) iii

- By induction, and using the extra constraints  $w_i^T w_j = 0$ , we can show that all other linear transformations are given by eigenvectors of  $\Sigma$ .

## PCA Theorem

Let  $\lambda_1 \geq \dots \geq \lambda_p$  be the eigenvalues of  $\Sigma$ , with corresponding unit-norm eigenvectors  $w_1, \dots, w_p$ . To reduce the dimension of  $\mathbf{Y}$  from  $p$  to  $k$  such that every component of  $W^T \mathbf{Y}$  is uncorrelated and each direction has maximal variance, we can take  $W = (w_1 \quad \dots \quad w_k)$ , whose  $j$ -th column is  $w_j$ .

## Properties of PCA i

- Some vocabulary:
  - $Z_i = w_i^T \mathbf{Y}$  is called the  $i$ -th **principal component** of  $\mathbf{Y}$ .
  - $w_i$  is the  $i$ -th vector of **loadings**.
- Note that we can take  $k = p$ , in which case we do not reduce the dimension of  $\mathbf{Y}$ , but we *transform* it into a random vector with *uncorrelated* components.
- Let  $\Sigma = P\Lambda P^T$  be the eigendecomposition of  $\Sigma$ . We have

$$\sum_{i=1}^p \text{Var}(w_i^T \mathbf{Y}) = \sum_{i=1}^p \lambda_i = \text{tr}(\Lambda) = \text{tr}(\Sigma) = \sum_{i=1}^p \text{Var}(Y_i).$$

- Therefore, each linear transformation  $w_i^T \mathbf{Y}$  contributes  $\lambda_i / \sum_j \lambda_j$  as percentage of the overall variance.

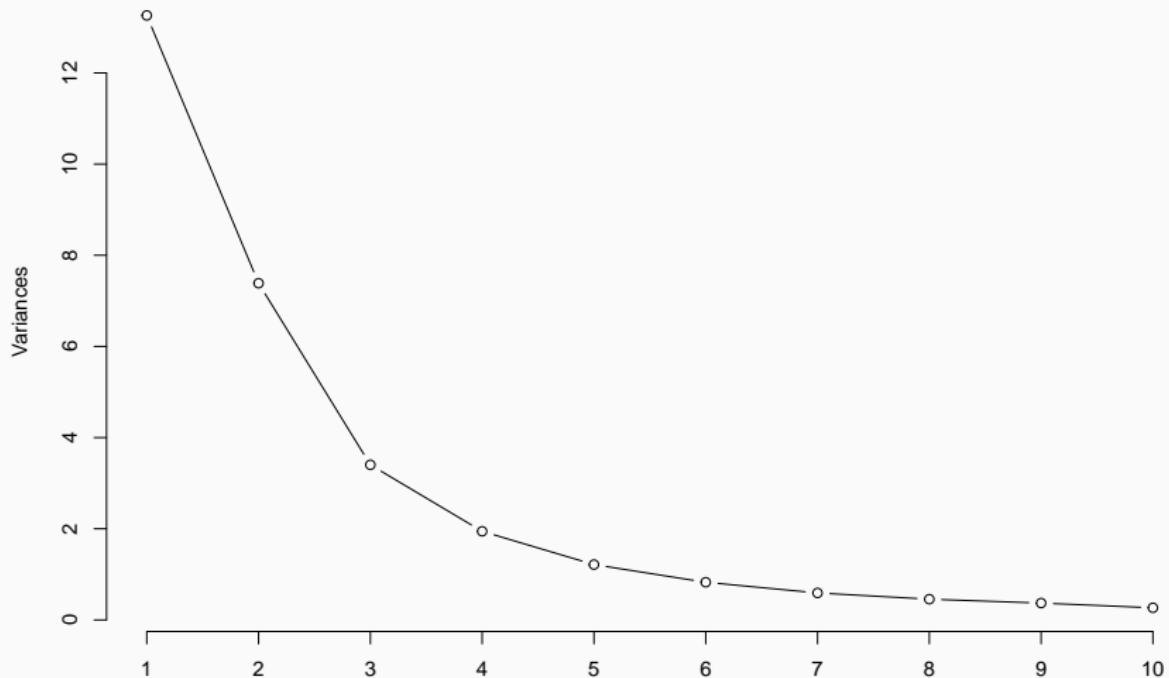
## Properties of PCA ii

- Selecting  $k$ : One common strategy is to select a threshold (e.g.  $c = 0.9$ ) such that

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i} \geq c.$$

## Scree plot

- A **scree plot** is a plot with the sequence  $1, \dots, p$  on the x-axis, and the sequence  $\lambda_1, \dots, \lambda_p$  on the y-axis.
- Another common strategy for selecting  $k$  is to choose the point where the curve starts to flatten out.
  - **Note:** This inflection point does not necessarily exist, and it may be hard to identify.



## Correlation matrix

- When the observations are on the different scale, it is typically more appropriate to normalise the components of  $\mathbf{Y}$  before doing PCA.
  - The variance depends on the units, and therefore without normalising, the component with the “smallest” units (e.g. centimeters vs. meters) could be driving most of the overall variance.
- In other words, instead of using  $\Sigma$ , we can use the (population) correlation matrix  $R$ .
- **Note:** The loadings and components we obtain from  $\Sigma$  are **not** equivalent to the ones obtained from  $R$ .

## Sample PCA

- In general, we do not have the population covariance matrix  $\Sigma$ .
- Therefore, in practice, we estimate the loadings  $w_i$  through the eigenvectors of the sample covariance matrix  $S_n$ .
- As with the population version of PCA, if the units are different, we should normalise the components or use the sample correlation matrix.

## Example 1 i

```
library(mvtnorm)
Sigma <- matrix(c(1, 0.5, 0.1,
                  0.5, 1, 0.5,
                  0.1, 0.5, 1),
                  ncol = 3)

set.seed(17)
X <- rmvnorm(100, sigma = Sigma)
pca <- prcomp(X)
```

## Example 1 ii

```
summary(pca)
```

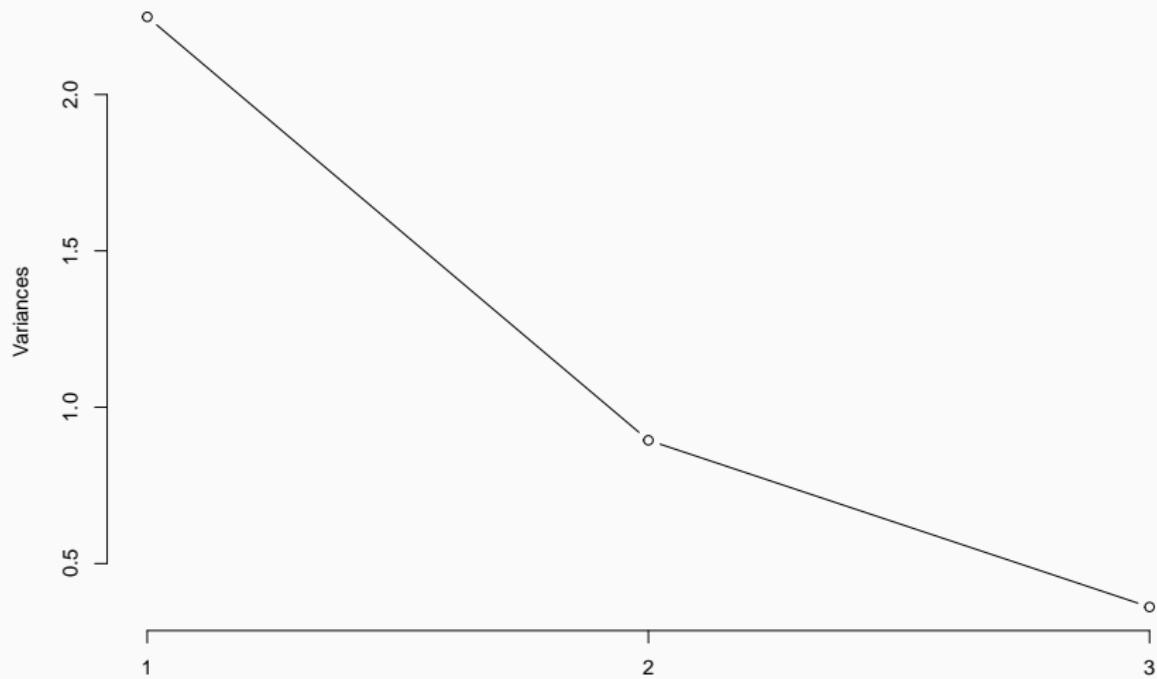
```
## Importance of components:
```

```
##                               PC1      PC2      PC3  
## Standard deviation     1.4994  0.9457  0.6009  
## Proportion of Variance 0.6417  0.2552  0.1031  
## Cumulative Proportion  0.6417  0.8969  1.0000
```

```
screeplot(pca, type = 'l')
```

## Example 1 iii

pca



## Example 2 i

```
pca <- prcomp(USArrests, scale = TRUE)
```

```
summary(pca)
```

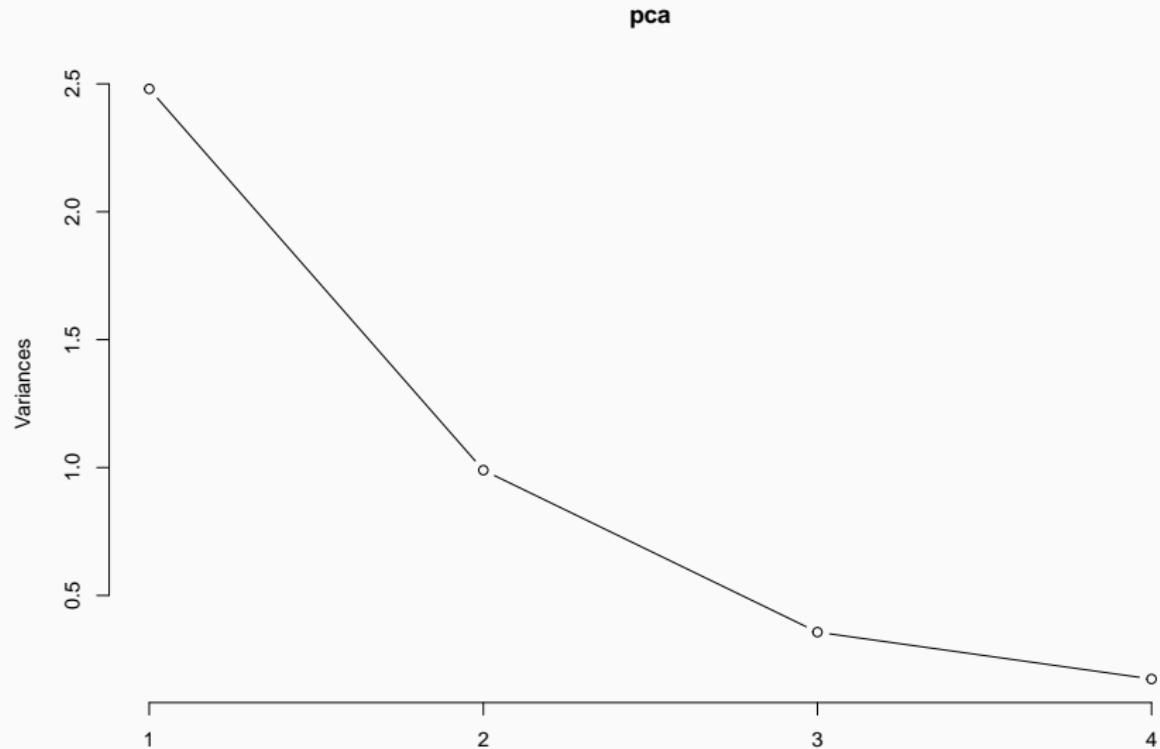
```
## Importance of components:
```

	PC1	PC2	PC3	PC4
## Standard deviation	1.5749	0.9949	0.59713	0.41645
## Proportion of Variance	0.6201	0.2474	0.08914	0.04336
## Cumulative Proportion	0.6201	0.8675	0.95664	1.00000

## Example 2 ii

```
screeplot(pca, type = 'l')
```

## Example 2 iii



## Additional comments about sample PCA i

- Let  $\mathbf{Y}_1, \dots, \mathbf{Y}_n$  be a sample from a distribution with covariance matrix  $\Sigma$ . Write  $\mathbb{Y}$  for the  $n \times p$  matrix whose  $i$ -th row is  $\mathbf{Y}_i$ .
- Let  $S_n$  be the sample covariance matrix, and write  $W_k$  for the matrix whose columns are the first  $k$  eigenvectors of  $S_n$ .
- You can define the matrix of  $k$  principal components as

$$\mathbb{Z} = \mathbb{Y}W_k.$$

## Additional comments about sample PCA ii

- On the other hand, it is much more common to define it as

$$\mathbb{Z} = \tilde{\mathbb{Y}}W_k,$$

where  $\tilde{\mathbb{Y}}$  is the centered version of  $\mathbb{Y}$  (i.e. the sample mean has been subtracted from each row).

- This leads to sample principal components with mean zero.

## Example 1 (revisited) i

```
library(mvtnorm)
Sigma <- matrix(c(1, 0.5, 0.1,
                  0.5, 1, 0.5,
                  0.1, 0.5, 1),
                 ncol = 3)
mu <- c(1, 2, 2)
```

## Example 1 (revisited) ii

```
set.seed(17)
X <- rmvnorm(100, mean = mu,
              sigma = Sigma)
pca <- prcomp(X)

colMeans(X)

## [1] 0.8789229 2.0517403 2.0965127

colMeans(pca$x)
```

## Example 1 (revisited) iii

```
##          PC1          PC2          PC3
## -5.523360e-17 4.454770e-17 7.077672e-18
```

# On the other hand

```
pca <- prcomp(X, center = FALSE)
colMeans(pca$x)
```

```
##          PC1          PC2          PC3
## 3.058960918 0.142358612 0.001050088
```

## Data Visualization i

```
library(tidyverse)
library(dslabs)

mnist <- read_mnist()

dim(mnist$train$images)

## [1] 60000    784

dim(mnist$test$images)
```

## Data Visualization ii

```
## [1] 10000    784

head(mnist$train$labels)

## [1] 5 0 4 1 9 2

img <- matrix(mnist$train$images[1,], ncol = 28)
# Switch columns
img <- img[,rev(1:28)]
image(img, col = gray.colors(12, rev = TRUE),
      axes = FALSE, asp = 1)
```

# Data Visualization iii

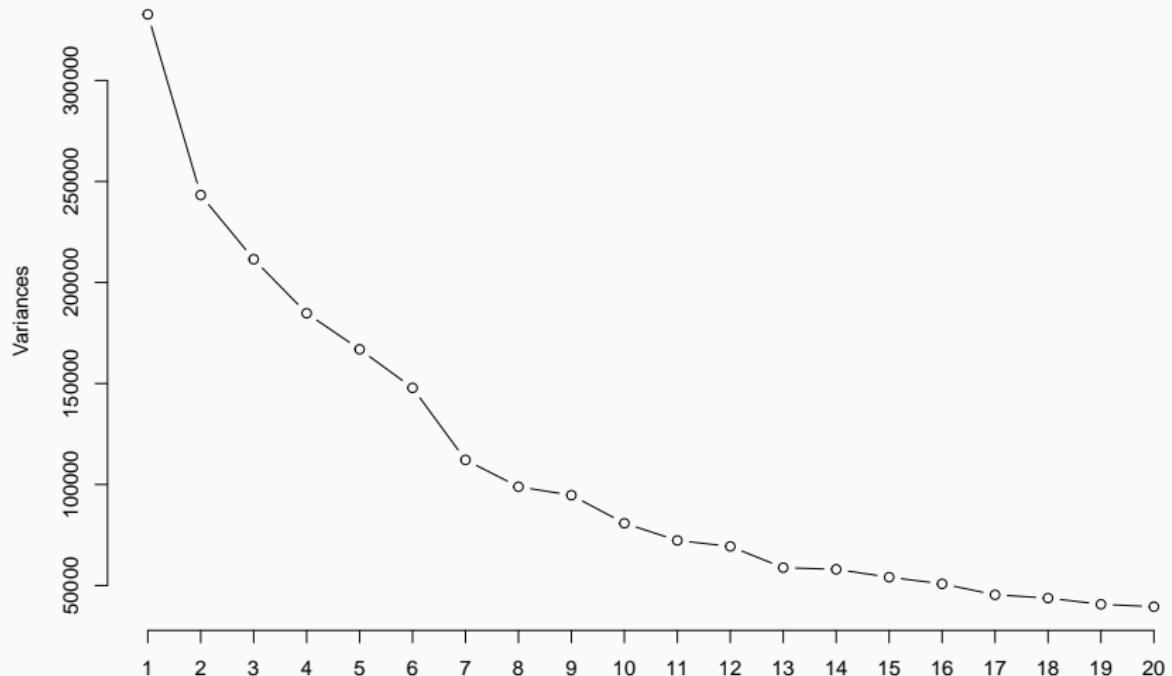


## Data Visualization iv

```
decomp <- prcomp(mnist$train$images)
```

```
screeplot(decomp, type = 'lines',
          n pcs = 20, main = "")
```

# Data Visualization v



## Data Visualization vi

```
decomp$x[,1:2] %>%  
  as.data.frame() %>%  
  mutate(label = factor(mnist$train$labels)) %>%  
  ggplot(aes(PC1, PC2, colour = label)) +  
  geom_point(alpha = 0.5) +  
  theme_minimal() +  
  coord_fixed()
```

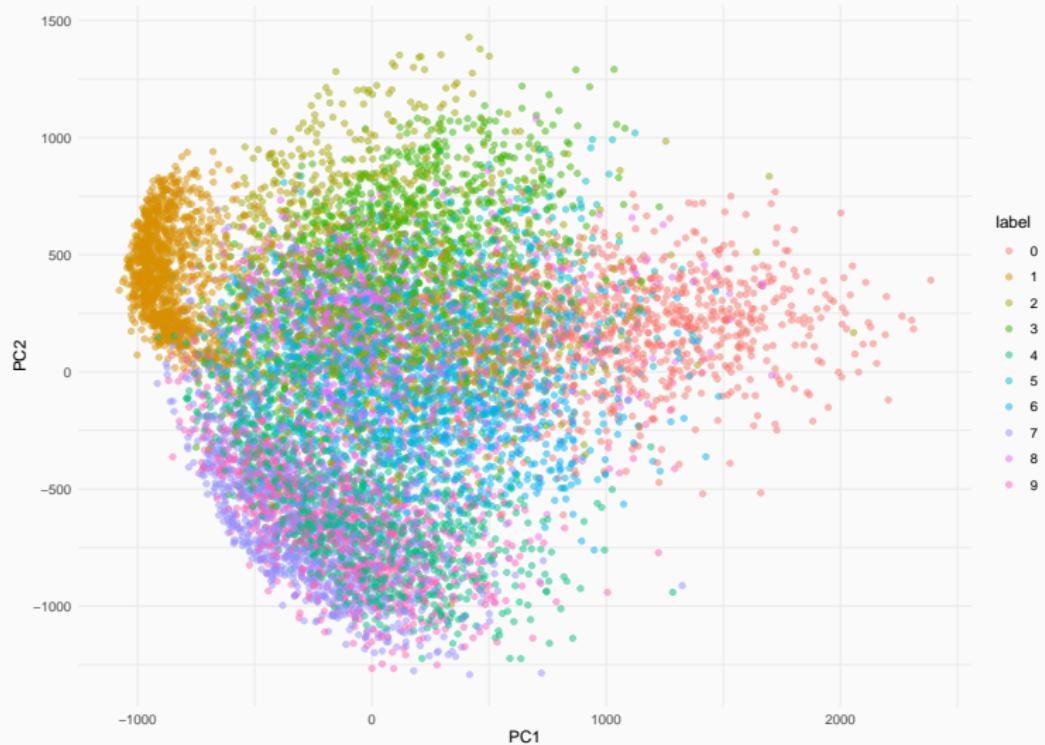
# Data Visualization vii



## Data Visualization viii

```
# And on the test set
decomp %>%
  predict(newdata = mnist$test$images) %>%
  as.data.frame() %>%
  mutate(label = factor(mnist$test$labels)) %>%
  ggplot(aes(PC1, PC2, colour = label)) +
  geom_point(alpha = 0.5) +
  theme_minimal() +
  coord_fixed()
```

# Data Visualization ix

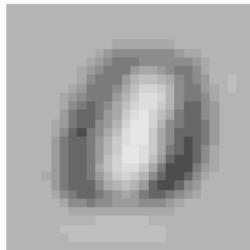


## Data Visualization x

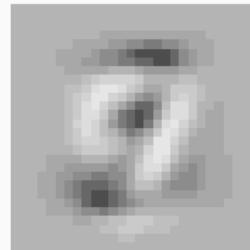
```
par(mfrow = c(2, 2))
for (i in seq_len(4)) {
  img <- matrix(decomp$rotation[,i], ncol = 28)
  img <- img[,rev(1:28)]
  image(img, col = gray.colors(12, rev = TRUE),
        axes = FALSE, main = paste0("PC", i), asp = 1)
}
```

# Data Visualization xi

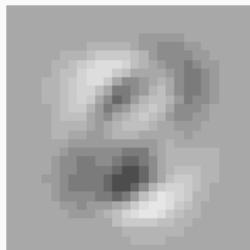
PC1



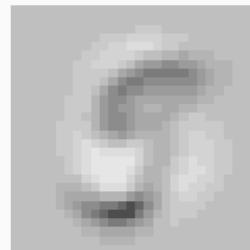
PC2



PC3



PC4



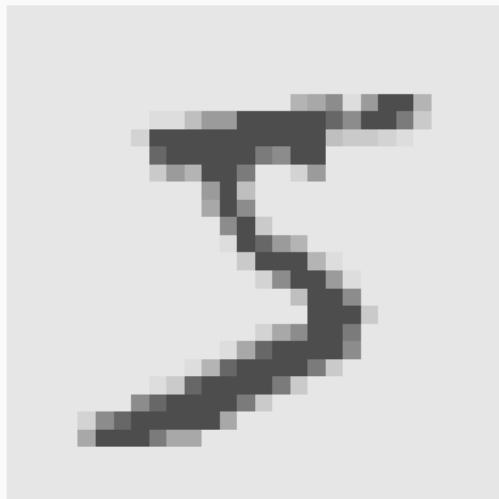
```
# Approximation with 90 PCs
approx_mnist <- decomp$rotation[, seq_len(90)] %*%
  decomp$x[1, seq_len(90)]

# Original image
img1 <- matrix(mnist$train$images[1, ], ncol = 28)
img1 <- img1[, rev(1:28)]
# Approximation
img2 <- matrix(approx_mnist, ncol = 28)
img2 <- img2[, rev(1:28)]
```

```
par(mfrow = c(1, 2))
image(img1, col = gray.colors(12, rev = TRUE),
      axes = FALSE, main = "Original", asp = 1)
image(img2, col = gray.colors(12, rev = TRUE),
      axes = FALSE, main = "Approx", asp = 1)
```

# Data Visualization xiv

Original



Approx



## Geometric interpretation of PCA i

- The definition of PCA as a linear combination that maximises variance is due to Hotelling (1933).
- But PCA was actually introduced earlier by Pearson (1901)
  - *On Lines and Planes of Closest Fit to Systems of Points in Space*
- He defined PCA as the **best approximation of the data by a linear manifold**
- Let's suppose we have a lower dimension representation of  $\mathbb{Y}$ , denoted by a  $n \times k$  matrix  $\mathbb{Z}$ .

## Geometric interpretation of PCA ii

- We want to *reconstruct*  $\mathbb{Y}$  using an affine transformation

$$f(z) = \mu + W_k z,$$

where  $W_k$  is a  $p \times k$  matrix whose columns are orthogonal vectors of unit length.

- We want to find  $\mu, W_k, \mathbf{Z}_i$  that minimises the **reconstruction error**:

$$\min_{\mu, W_k, \mathbf{Z}_i} \sum_{i=1}^n \|\mathbf{Y}_i - \mu - W_k \mathbf{Z}_i\|^2.$$

## Geometric interpretation of PCA iii

- First, treating  $W_k$  constant and minimising over  $\mu, \mathbf{Z}_i$ , we find

$$\hat{\mu} = \bar{\mathbf{Y}},$$

$$\hat{\mathbf{Z}}_i = W_k^T(\mathbf{Y}_i - \bar{\mathbf{Y}}).$$

- Putting these quantities into the reconstruction error, we get

$$\min_{W_k} \sum_{i=1}^n \|(\mathbf{Y}_i - \bar{\mathbf{Y}}) - W_k W_k^T (\mathbf{Y}_i - \bar{\mathbf{Y}})\|^2.$$

## Geometric interpretation of PCA iv

### Eckart–Young theorem

The reconstruction error is minimised by taking  $W_k$  to be the matrix whose columns are the first  $k$  eigenvectors of the sampling covariance matrix  $S_n$ .

Equivalently, we can take the matrix whose columns are the first  $k$  right singular vectors or the centered data matrix  $\tilde{\mathbb{Y}}$ .

## Example i

```
set.seed(1234)
# Random measurement error
sigma <- 5

# Exact relationship between
# Celsius and Fahrenheit
temp_c <- seq(-40, 40, by = 1)
temp_f <- 1.8*temp_c + 32
```

## Example ii

```
# Add measurement error
temp_c_noise <- temp_c + rnorm(n = length(temp_c),
                                 sd = sigma)
temp_f_noise <- temp_f + rnorm(n = length(temp_f),
                                 sd = sigma)

# Linear model
(fit <- lm(temp_f_noise ~ temp_c_noise))
```

### Example iii

```
##  
## Call:  
## lm(formula = temp_f_noise ~ temp_c_noise)  
##  
## Coefficients:  
## (Intercept)  temp_c_noise  
##           34.256          1.662  
  
confint(fit)
```

## Example iv

```
##                      2.5 %    97.5 %
## (Intercept) 32.152891 36.35921
## temp_c_noise 1.577228  1.74711

# PCA
pca <- prcomp(cbind(temp_c_noise, temp_f_noise))
pca$rotation
```

```
##                  PC1          PC2
## temp_c_noise 0.5012360 -0.8653106
## temp_f_noise 0.8653106  0.5012360
```

## Example v

```
pca$rotation[2,"PC1"] / pca$rotation[1,"PC1"]
```

```
## [1] 1.726354
```

## Large sample inference i

- If we impose distributional assumptions on the data  $\mathbf{Y}$ , we can derive the sampling distributions of the sample principal components.
- Assume  $\mathbf{Y} \sim N_p(\mu, \Sigma)$ , with  $\Sigma$  positive definite. Let  $\lambda_1 > \dots > \lambda_p$  be the eigenvalues of  $\Sigma$ ; in particular we assume they are *distinct*. Finally let  $w_1, \dots, w_p$  be the corresponding eigenvectors.
- Given a random sample of size  $n$ , let  $S_n$  be the sample covariance matrix,  $\hat{\lambda}_1, \dots, \hat{\lambda}_p$  its eigenvalues, and  $\hat{w}_1, \dots, \hat{w}_p$  the corresponding eigenvectors.

## Large sample inference ii

- Define  $\Lambda$  to be the diagonal matrix whose entries are  $\lambda_1, \dots, \lambda_p$ , and define

$$\Omega_i = \lambda_i \sum_{k=1, k \neq i}^p \frac{\lambda_k}{(\lambda_k - \lambda_i)^2} w_k w_k^T.$$

## Asymptotic results

1. Write  $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_p)$  and similarly for  $\hat{\boldsymbol{\lambda}}$ . As  $n \rightarrow \infty$ , we have

$$\sqrt{n} (\hat{\boldsymbol{\lambda}} - \boldsymbol{\lambda}) \rightarrow N_p(0, 2\Lambda^2).$$

2. As  $n \rightarrow \infty$ , we have

$$\sqrt{n} (\hat{w}_i - w_i) \rightarrow N_p(0, \Omega_i).$$

3. Each  $\hat{\lambda}_i$  is distributed independently of  $\hat{w}_i$ .

## Comments i

- These results **only** apply to principal components derived from the covariance matrix.
  - Some asymptotic results are available for those derived from the correlation matrix, but we will not cover them in class.
- Asymptotically, all eigenvalues of  $S_n$  are independent.
- You can get a confidence interval for  $\lambda_i$  as follows:

$$\frac{\hat{\lambda}_i}{(1 + z_{\alpha/2}\sqrt{2/n})} \leq \lambda_i \leq \frac{\hat{\lambda}_i}{(1 - z_{\alpha/2}\sqrt{2/n})}.$$

- Use Bonferroni correction if you want CIs that are simultaneously valid for all eigenvalues.

## Comments ii

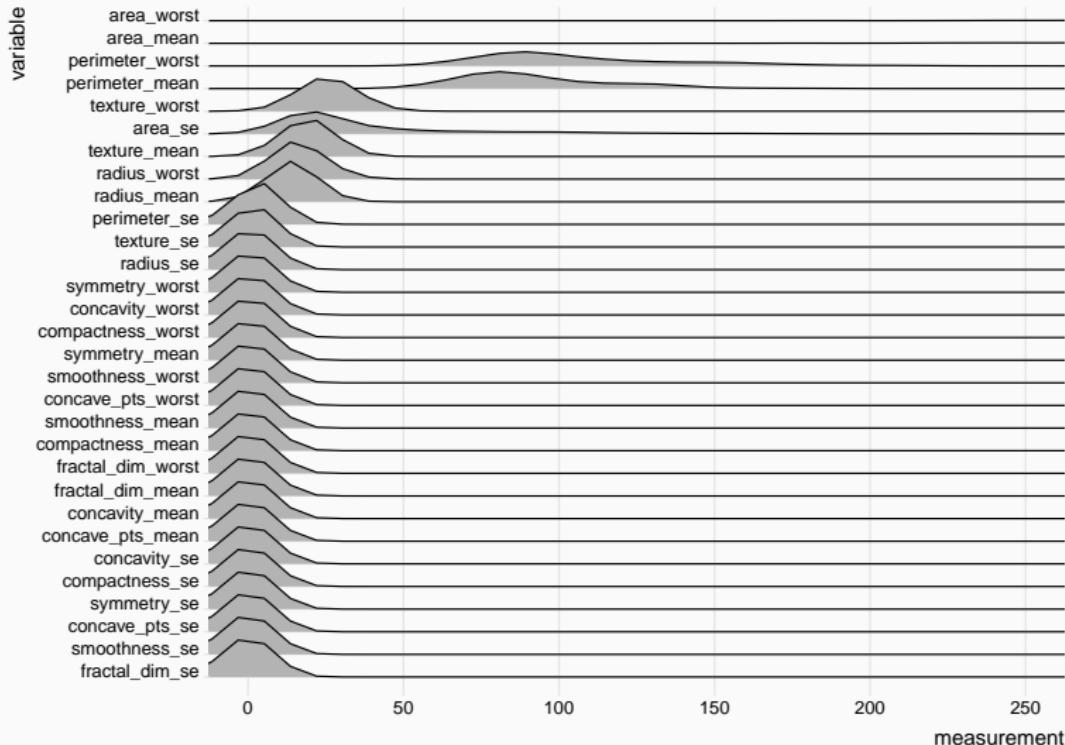
- The matrices  $\Omega_i$  have rank  $p - 1$ , and therefore they are *singular*.
- The entries of  $\hat{w}_i$  are correlated, and this correlation depends on the *separation* between the eigenvalues.
  - Good separation  $\implies$  smaller correlation

## Example i

```
library(dslabs)
library(ggridges)

# Data on Breast Cancer
as.data.frame(brca$x) %>%
  gather(variable, measurement) %>%
  mutate(variable = reorder(variable, measurement,
                            median)) %>%
  ggplot(aes(x = measurement, y = variable)) +
  geom_density_ridges() + theme_ridges() +
  coord_cartesian(xlim = c(0, 250))
```

## Example ii



### Example iii

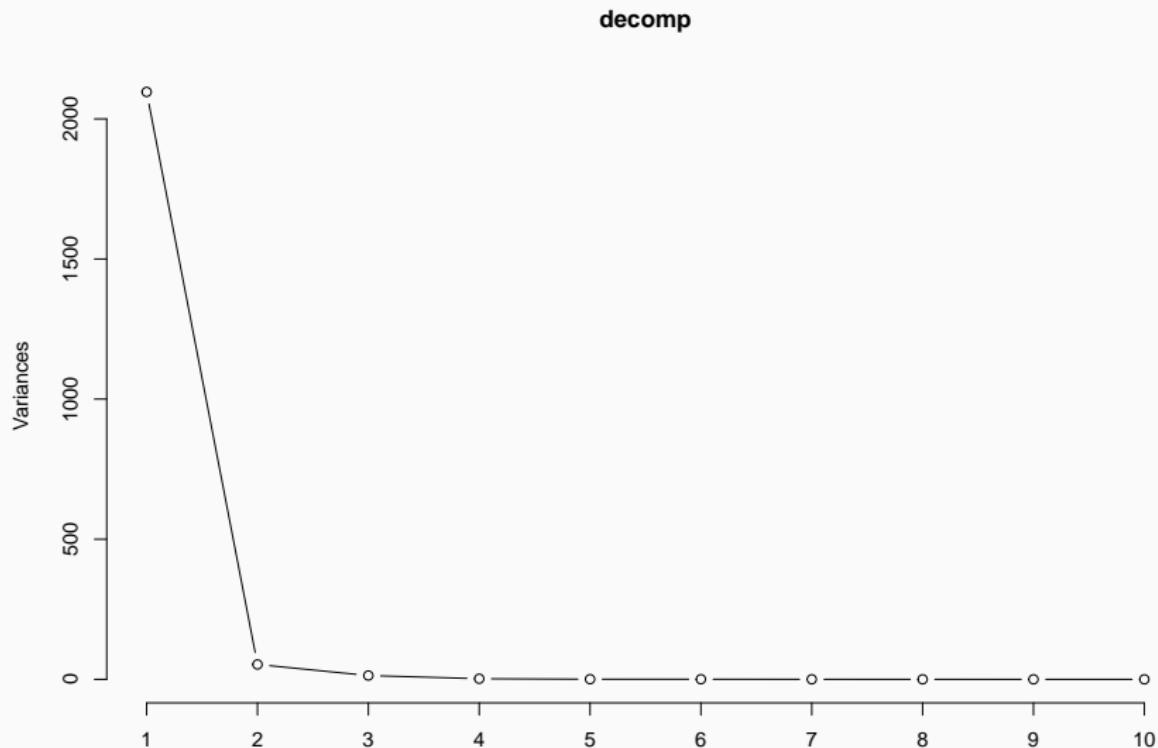
```
# Remove some variables
rem_index <- which(colnames(brca$x) %in%
                     c("area_worst", "area_mean",
                       "perimeter_worst",
                       "perimeter_mean"))
dataset <- brca$x[, -rem_index]
decomp <- prcomp(dataset)
summary(decomp)$importance[, 1:3]
```

## Example iv

```
##                                     PC1      PC2      PC3
## Standard deviation      45.78445 7.281664 3.677815
## Proportion of Variance  0.96776 0.024480 0.006240
## Cumulative Proportion   0.96776 0.992240 0.998490

screeplot(decomp, type = 'l')
```

## Example v



## Example vi

```
# Let's put a CI around the first eigenvalue
first_ev <- decomp$sdev[1]^2
n <- nrow(dataset)

# Recall that TV = 2166
c("LB" = first_ev/(1+qnorm(0.975)*sqrt(2/n)),
  "Est." = first_ev,
  "UP" = first_ev/(1-qnorm(0.975)*sqrt(2/n)))

##          LB      Est.       UP
## 1877.992 2096.216 2371.822
```

## Simulations i

```
B <- 1000; n <- 100; p <- 3

results <- purrr::map_df(seq_len(B), function(b) {
  X <- matrix(rnorm(p*n, sd = sqrt(c(1, 2, 3))),  

              ncol = p, byrow = TRUE)
  tmp <- eigen(cov(X), symmetric = TRUE,  

              only.values = TRUE)
  tibble(ev1 = tmp$values[1],  

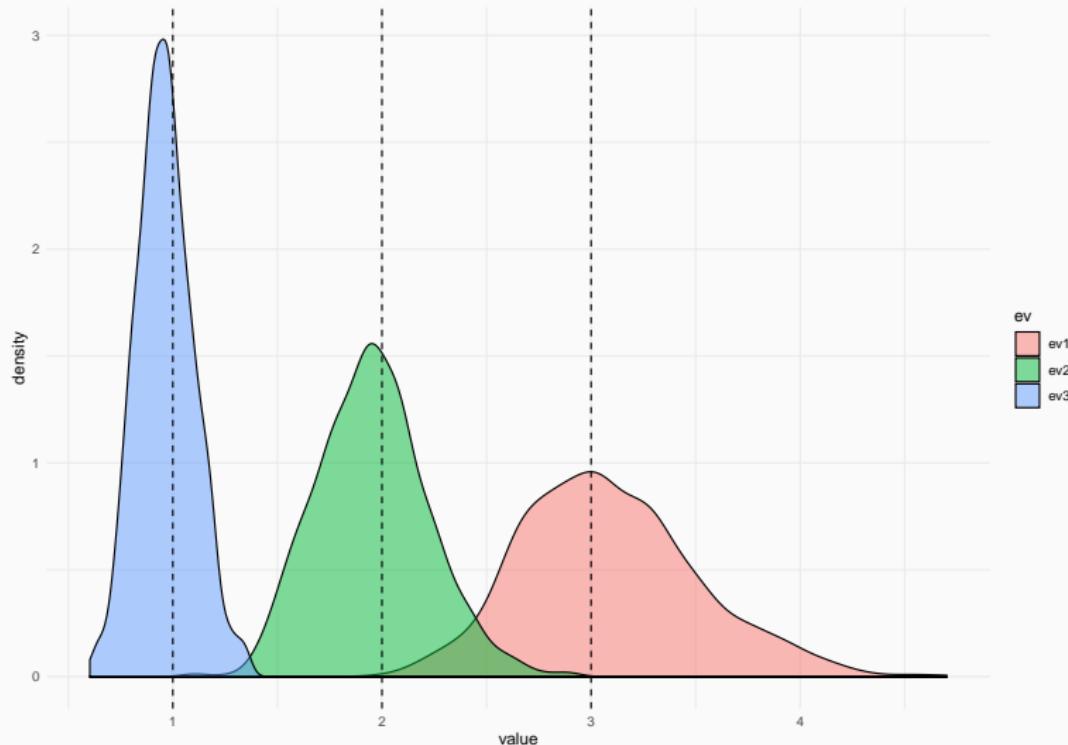
         ev2 = tmp$values[2],  

         ev3 = tmp$values[3])
})
```

## Simulations ii

```
results %>%  
gather(ev, value) %>%  
ggplot(aes(value, fill = ev)) +  
geom_density(alpha = 0.5) +  
theme_minimal() +  
geom_vline(xintercept = c(1, 2, 3),  
linetype = 'dashed')
```

# Simulations iii



## Simulations iv

```
results %>%  
  summarise_all(mean)  
  
## # A tibble: 1 x 3  
##       ev1     ev2     ev3  
##   <dbl> <dbl> <dbl>  
## 1  3.09  1.95  0.963
```

## Simulations v

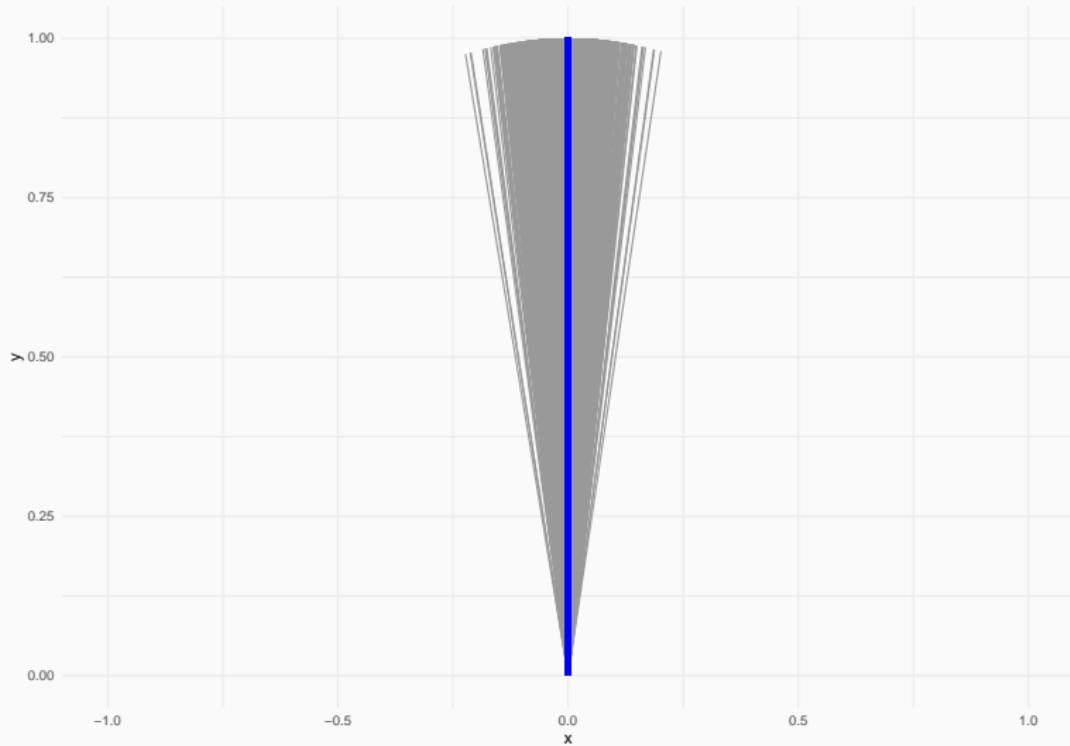
```
p <- 2
results <- purrr::map_df(seq_len(B), function(b) {
  X <- matrix(rnorm(p*n, sd = c(1, 2)), ncol = p,
              byrow = TRUE)
  tmp <- eigen(cov(X), symmetric = TRUE)

  tibble(
    xend = tmp$vectors[1,1],
    yend = tmp$vectors[2,1]
  )
})
```

## Simulations vi

```
results %>%  
  ggplot() +  
  geom_segment(aes(xend = xend, yend = yend),  
               x = 0, y = 0, colour = 'grey60') +  
  geom_segment(x = 0, xend = 0,  
               y = 0, yend = 1,  
               colour = 'blue', size = 2) +  
  expand_limits(y = 0, x = c(-1, 1)) +  
  theme_minimal()
```

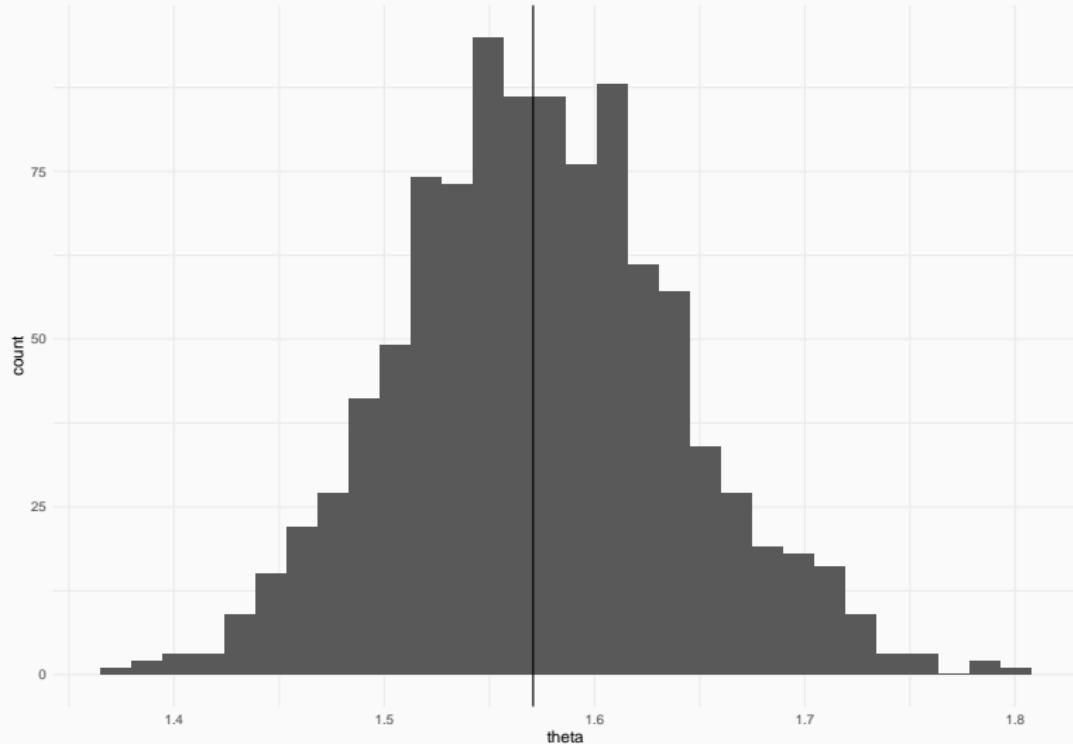
## Simulations vii



## Simulations viii

```
# Or looking at angles
results %>%
  transmute(theta = atan2(yend, xend)) %>%
  ggplot(aes(theta)) +
  geom_histogram() +
  theme_minimal() +
  geom_vline(xintercept = pi/2)
```

## Simulations ix



## Test for structured covariance i

- The asymptotic results above assumed distinct eigenvalues.
- But we may be interested in *structured* covariance matrices; for example:

$$\Sigma_0 = \sigma^2 \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}.$$

- This is called an **exchangeable** correlation structure.

## Test for structured covariance ii

- Assuming  $\rho > 0$ , the eigenvalues of  $\Sigma_0$  are

$$\lambda_1 = \sigma^2(1 + (p - 1)\rho),$$

$$\lambda_2 = \sigma^2(1 - \rho),$$

$$\vdots \quad \vdots$$

$$\lambda_p = \sigma^2(1 - \rho).$$

- Let's assume  $\sigma^2 = 1$ . We are interested in testing whether the correlation matrix is equal to  $\Sigma_1$ .
- Let  $\bar{r}_k = \frac{1}{p-1} \sum_{i=1, i \neq k}^p r_{ik}$  be the average of the off-diagonal value of the  $k$ -th column of the sample correlation matrix.

## Test for structured covariance iii

- Let  $\bar{r} = \frac{2}{p(p-1)} \sum_{i < j} r_{ij}$  be the average of all off-diagonal elements (we are only looking at entries below the diagonal).
- Finally, let  $\hat{\gamma} = \frac{(p-1)^2[1-(1-\bar{r})^2]}{p-(p-2)(1-\bar{r})^2}$ .
- We reject the null hypothesis that the correlation matrix is equal to  $\Sigma_0$  if

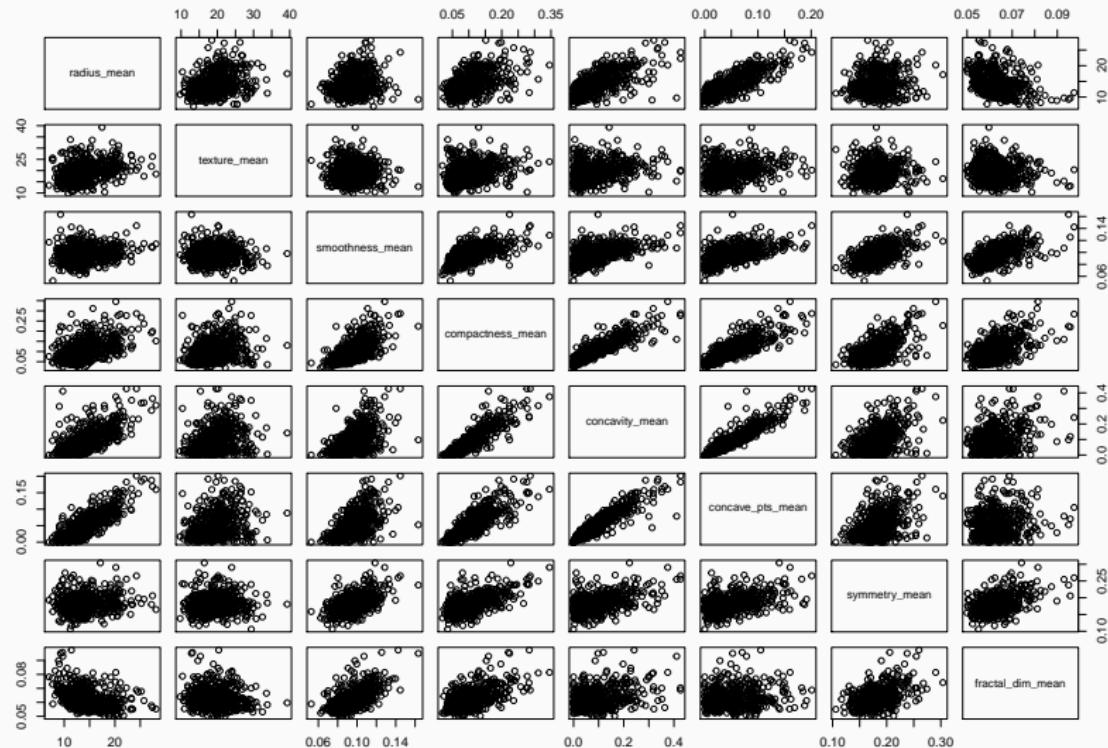
$$\frac{(n-1)}{(1-\bar{r})^2} \left[ \sum_{i < j} (r_{ij} - \bar{r})^2 - \hat{\gamma} \sum_{k=1}^p (\bar{r}_k - \bar{r})^2 \right] > \chi_{\alpha}^2((p+1)(p-2)/2)$$

## Example i

```
# Keep only mean measurements
rem_index2 <- grep("mean", colnames(brca$x),
                     invert = TRUE)
dataset <- brca$x[,-c(rem_index,
                      rem_index2)]
R <- cor(dataset)

pairs(dataset)
```

## Example ii



### Example iii

```
# Overall mean  
r_bar <- mean(R[upper.tri(R, diag = FALSE)])  
  
# Column specific means  
r_cols <- (colSums(R) - 1)/(nrow(R) - 1)  
  
# Extra quantities  
p <- ncol(dataset)  
n <- nrow(dataset)  
gamma_hat <- (p - 1)^2*(1 - (1 - r_bar)^2)/  
(p - (p - 2)*(1 - r_bar)^2)
```

## Example iv

```
# Test statistic
Tstat <- sum((R[upper.tri(R,
                           diag = FALSE)] - r_bar)^2) -
  gamma_hat*sum((r_cols - r_bar)^2)
Tstat <- (n-1)*Tstat/(1-r_bar)^2

Tstat > qchisq(0.95, 0.5*(p+1)*(p-2))

## [1] TRUE
```

## Selecting the number of PCs i

- We already discussed two strategies for selecting the number of principal components:
  - Look at the scree plot and find where the curve starts to be flat;
  - Retain as many PCs as required to explain the desired proportion of variance.
- There is a **vast** literature on different strategies for selecting the number of components. Two good references:
  - Peres-Neto *et al.* (2005) *How many principal components? stopping rules for determining the number of non-trivial axes revisited*
  - Jolliffe (2012) *Principal Component Analysis* (2nd ed)

## Selecting the number of PCs ii

- We will discuss one more technique based on resampling.
- The idea is to try to estimate the distribution of eigenvalues if there was no correlation between the variables.

### Algorithm

1. Permute the observations of each column **independently**.
2. Perform PCA on the permuted data.
3. Repeat  $B$  times and collect the eigenvalues  $\hat{\lambda}_1^{(b)}, \dots, \hat{\lambda}_p^{(b)}$ .
4. Keep the components whose observed  $\hat{\lambda}_i$  is greater than  $(1 - \alpha)\%$  of the values  $\hat{\lambda}_i^{(b)}$  obtained through permutations.

## Example (cont'd) i

```
decomp <- prcomp(dataset)
```

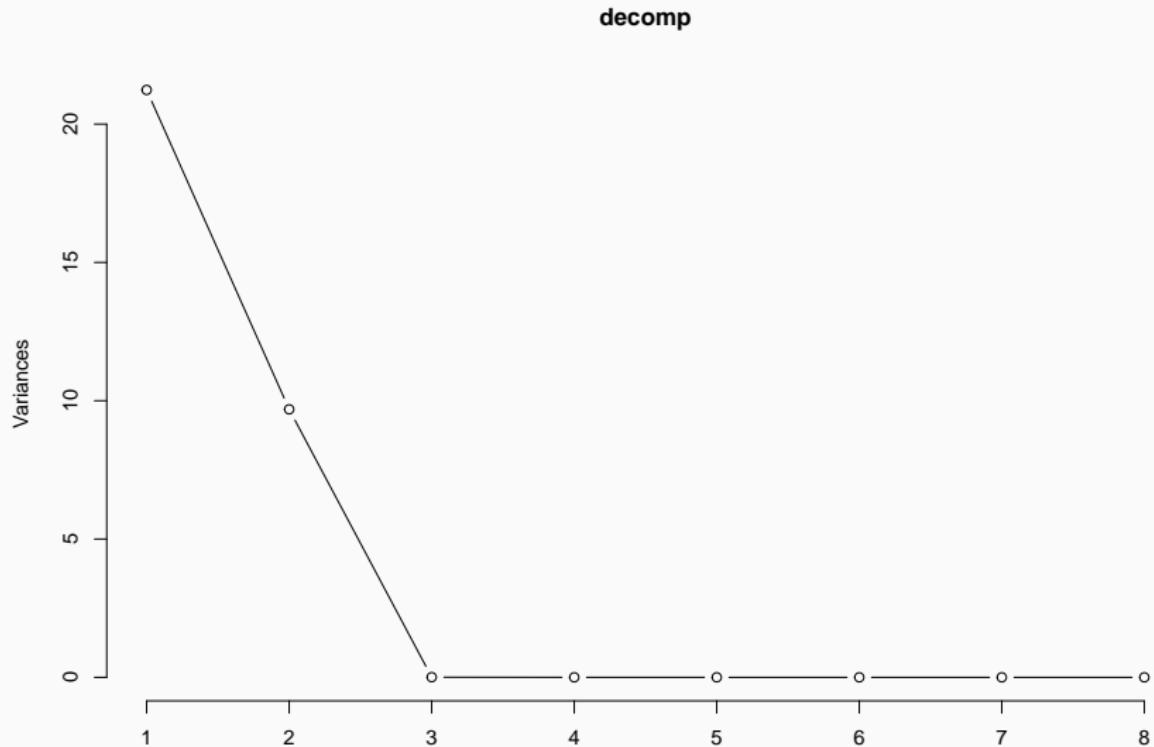
```
summary(decomp)$importance[, seq_len(3)]
```

	PC1	PC2	PC3
## Standard deviation	4.60806	3.112611	0.07664969
## Proportion of Variance	0.68654	0.313240	0.00019000
## Cumulative Proportion	0.68654	0.999780	0.99997000

## Example (cont'd) ii

```
screeplot(decomp, type = 'l')
```

## Example (cont'd) iii



## Example (cont'd) iv

```
permute_data <- function(data) {  
  p <- ncol(data)  
  data_perm <- data  
  for (i in seq_len(p)) {  
    ind_sc <- sample(nrow(data))  
    data_perm[,i] <- data[ind_sc, i]  
  }  
  return(data_perm)  
}
```

## Example (cont'd) v

```
set.seed(123)
B <- 1000
alpha <- 0.05
results <- matrix(NA, ncol = B,
                  nrow = ncol(dataset))

results[,1] <- decomp$sdev
results[,-1] <- replicate(B - 1, {
  data_perm <- permute_data(dataset)
  prcomp(data_perm)$sdev
})
```

## Example (cont'd) vi

```
cutoff <- apply(results, 1, function(row) {  
  mean(row >= row[1])  
})  
which(cutoff < alpha)  
  
## [1] 1
```

## Biplots i

- In our example with the MNIST dataset, we plotted the first principal component against the second component.
  - This gave us a sense of how much discriminatory ability each PC gave us.
  - E.g. the first PC separated 1s from 0s
- What was missing from that plot was how the PCs were related to the original variables.
- A **biplot** is a graphical display of both the original observations and original variables *together* on one scatterplot.
  - The prefix “bi” refers to two modalities (i.e. observations and variables), not to two dimensions.

## Biplots ii

- One approach to biplots relies on the Eckart-Young theorem:
  - The “best” 2-dimensional representation of the data passes through the plane containing the first two eigenvectors of the sample covariance matrix.

## Biplots iii

### Construction

- Let  $\tilde{\mathbb{Y}}$  be the  $n \times p$  matrix of centered data, and let  $w_1, \dots, w_p$  be the  $p$  eigenvectors of  $\tilde{\mathbb{Y}}^T \tilde{\mathbb{Y}}$ .
- For each row  $\mathbf{Y}_i$  of  $\mathbb{Y}$ , add the point  $(w_1^T \mathbf{Y}_i, w_2^T \mathbf{Y}_i)$  to the plot.
- The  $j$ -th column of  $\mathbb{Y}$  is represented by an arrow from the origin to the point  $(w_{1j}, w_{2j})$ .
- It may be necessary to rescale the PCs and/or the loadings in order to see the relationship better.

## Example (cont'd) i

```
# Continuing with our example on breast cancer
decomp <- prcomp(dataset)

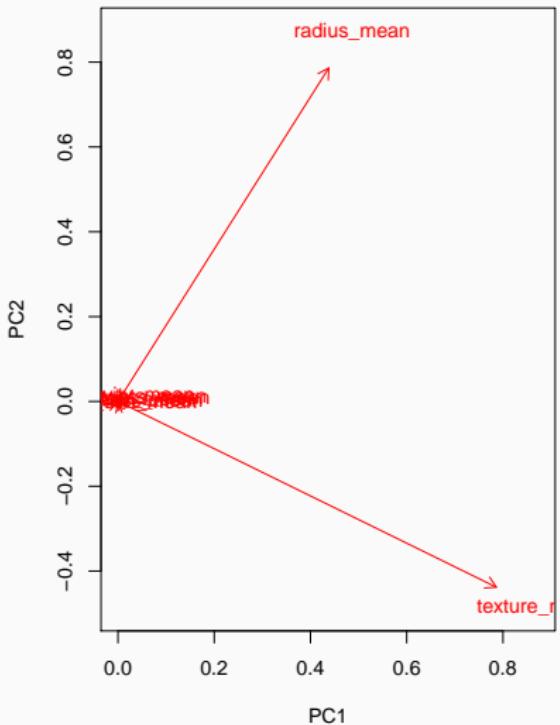
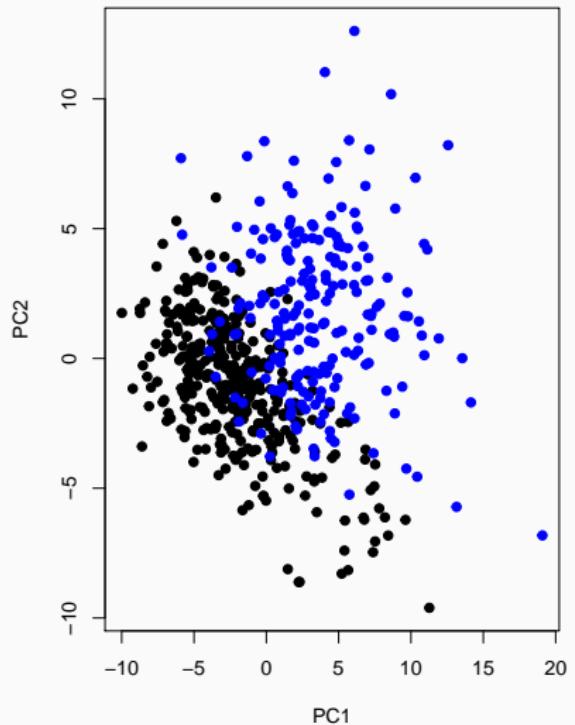
# Extract PCs and loadings
PCs <- decomp$x[, 1:2]
loadings <- decomp$rotation[, 1:2]

# Extract data on tumour type
colour <- ifelse(brca$y == "B", "black", 'blue')
```

## Example (cont'd) ii

```
par(mfrow = c(1,2))
plot(PCs, pch = 19, col = colour)
plot(loadings, type = 'n')
text(loadings,
      labels = colnames(dataset),
      col = 'red')
arrows(0, 0, 0.9 * loadings[, 1],
       0.9 * loadings[, 2],
       col = 'red',
       length = 0.1)
```

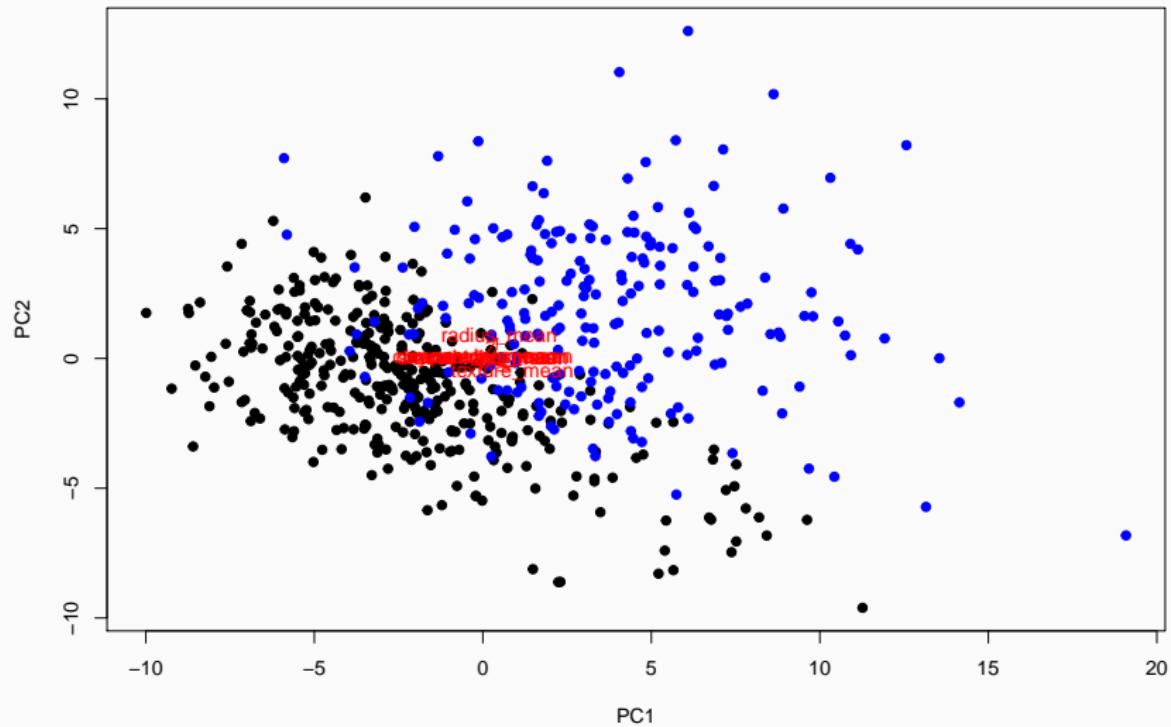
### Example (cont'd) iii



## Example (cont'd) iv

```
# Or both on the same plot
plot(PCs, pch = 19, col = colour)
text(loadings,
      labels = colnames(dataset),
      col = 'red')
arrows(0, 0, 0.9 * loadings[, 1],
       0.9 * loadings[, 2],
       col = 'red',
       length = 0.1)
```

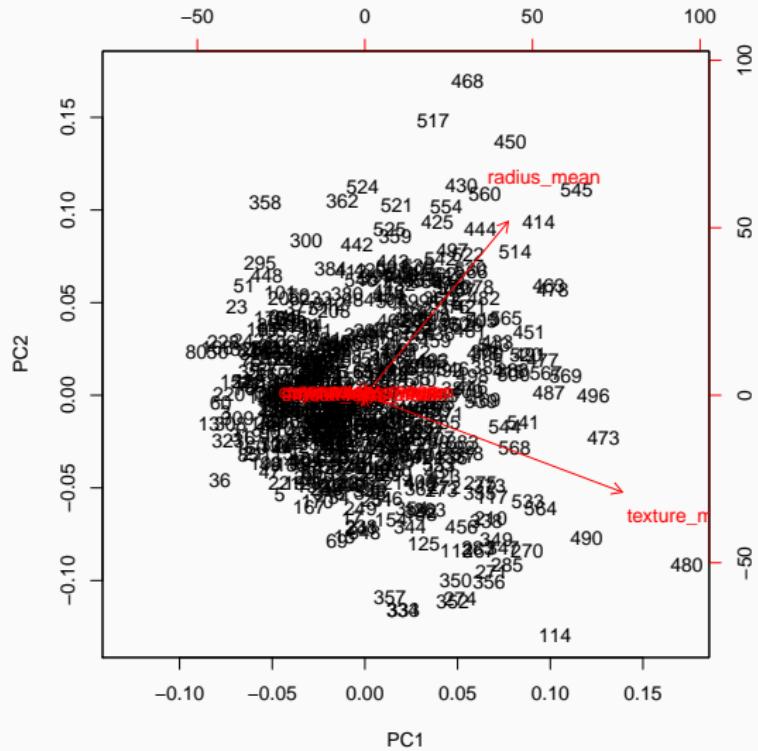
## Example (cont'd) v



## Example (cont'd) vi

```
# The biplot function rescales for us  
biplot(decomp)
```

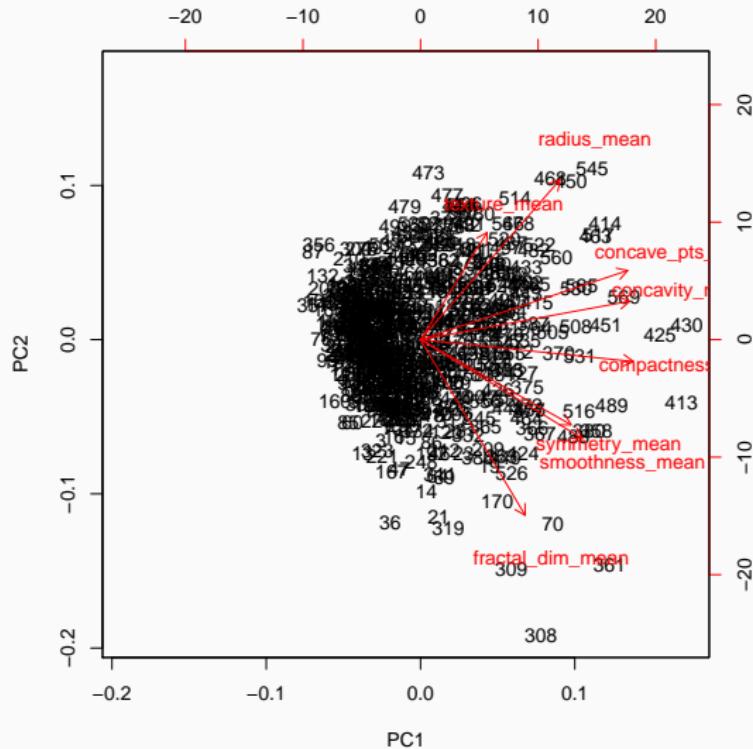
## Example (cont'd) vii



## Example (cont'd) viii

```
# With scaled data  
biplot(prcomp(dataset, scale = TRUE))
```

## Example (cont'd) ix



## Summary of graphical displays

- When we plot the first PC against the second PC, we are looking for similarity between *observations*.
- When we plot the first loading against the second loading, we are looking for similarity between *variables*.
  - Orthogonal loadings  $\implies$  Uncorrelated variables
  - Obtuse angle between loadings  $\implies$  Negative correlation
- A **biplot** combines both pieces of information.
  - You can think of it as a projection of the  $p$ -dimensional scatter plot (points and axes) onto a 2-dimensional plane.
- A **scree plot** displays the amount of variation in each principal component.

# Applications of PCA to Model Building

---

## Training and testing i

- Recall: Mean Squared Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2,$$

where  $Y_i, \hat{Y}_i$  are the *observed* and *predicted* values.

- It is good practice to separate your dataset in two:
  - **Training** dataset, that is used to build and fit your model (e.g. choose covariates, estimate regression coefficients).
  - **Testing** dataset, that it used to compute the MSE or other performance metrics.

## Training and testing ii

- PCA can be used for predictive model building in (univariate) linear regression:
  - **Feature extraction:** Perform PCA on the covariates, extract the first  $k$  PCs, and use them as predictors in your model.
  - **Feature selection:** Perform PCA on the covariates, look at the first PC, find the covariates whose loadings are the largest (in absolute value), and only use those covariates as predictors.

## Feature Extraction i

## Feature Extraction ii

```
library(tidyverse)
url <- "https://maxturgeon.ca/w20-stat7200/prostate.csv"
prostate <- read_csv(url)

# Separate into training and testing sets
data_train <- filter(prostate, train == TRUE) %>%
  dplyr::select(-train)
data_test <- filter(prostate, train == FALSE) %>%
  dplyr::select(-train)

# First model: Linear regression
lr_model <- lm(lpsa ~ ., data = data_train)
lr_pred <- predict(lr_model, newdata = data_test)
(lr_mse <- mean((data_test$lpsa - lr_pred)^2))
```

## Feature Selection i

```
contribution <- decomp$rotation[, "PC1"]
round(contribution, 3)[1:6]

## lcavol lweight      age      lbph      svi      lcp
##  0.021   0.001   0.075  -0.001   0.007   0.032

round(contribution, 3)[7:8]

## gleason    pgg45
##  0.018    0.996
```

## Feature Selection ii

```
(keep <- names(which(abs(contribution) > 0.01)))  
  
## [1] "lcavol"   "age"       "lcp"       "gleason"  "pgg45"  
  
fs_model <- lm(lpsa ~ ., data = data_train[,c(keep, "lpsa")])  
fs_pred <- predict(fs_model, newdata = data_test)  
(fs_mse <- mean((data_test$lpsa - fs_pred)^2))  
  
## [1] 0.5815571
```

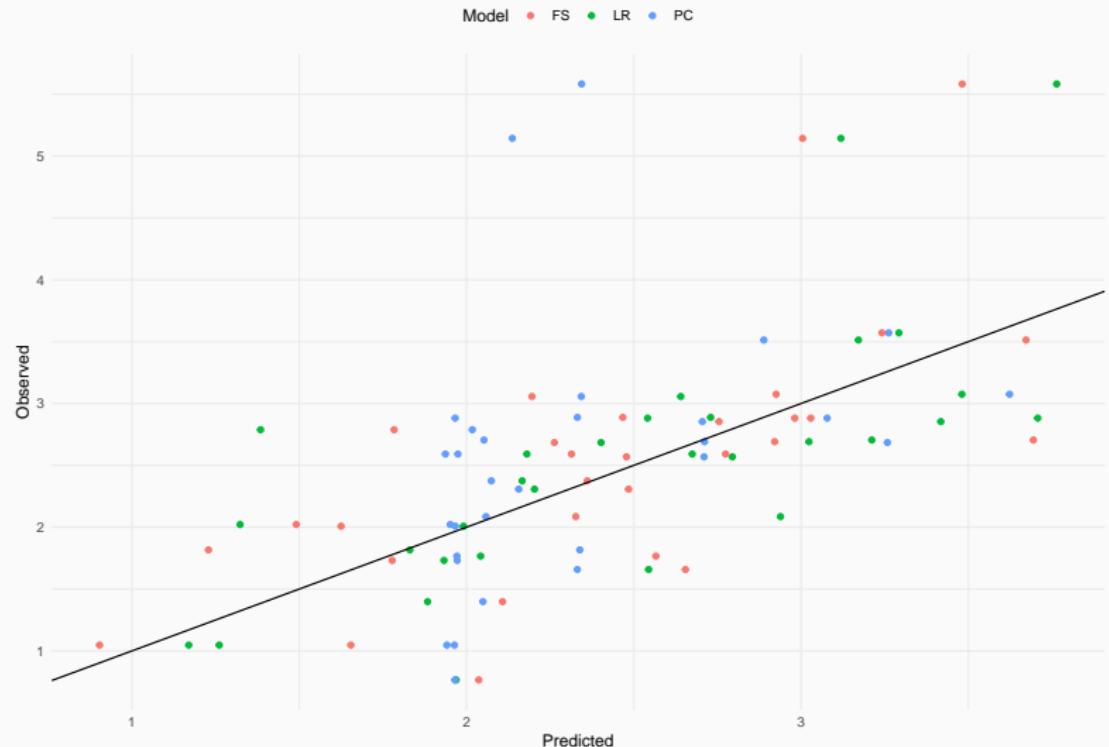
## Feature Selection iii

```
model_plot <- data.frame(  
  "obs" = data_test$lpsa,  
  "LR" = lr_pred,  
  "PC" = pc_pred,  
  "FS" = fs_pred  
) %>%  
  gather(Model, pred, -obs)
```

## Feature Selection iv

```
ggplot(model_plot,  
       aes(pred, obs, colour = Model)) +  
  geom_point() +  
  theme_minimal() +  
  geom_abline(slope = 1, intercept = 0) +  
  theme(legend.position = 'top') +  
  xlab("Predicted") + ylab("Observed")
```

# Feature Selection v



## Comments

- The full model performed better than the ones we created with PCA
  - It had a lower MSE
- On the other hand, if we had multicollinearity issues, or too many covariates ( $p > n$ ), the PCA models could outperform the full model.
- However, note that PCA does not use the association between the covariates and the outcome, so it will never be the most efficient way of building a model.