

Principal Component Analysis

Max Turgeon

STAT 4690—Applied Multivariate Analysis

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 Σ . 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 $\|w\| = 1$)

Population PCA ii

- 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 $w_1^T \mathbf{Y}, \dots, w_{j-1}^T \mathbf{Y}$.
- First, note that $\text{Var}(w^T \mathbf{Y}) = w^T \Sigma w$. So our optimisation problem is

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

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

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

Population PCA iii

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

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

- From the first partial derivative, we conclude that

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

Population PCA iv

- From the second partial derivative, we conclude that $w \neq 0$; in other words, w is an eigenvector of Σ with eigenvalue λ .
- 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 λ to be the *largest* eigenvalue of Σ .
- 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 Σ .

Population PCA v

PCA Theorem

Let $\lambda_1 \geq \dots \geq \lambda_p$ be the eigenvalues of Σ , 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 $W = (w_1 \ \dots \ w_k)$, 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 Σ . We have

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

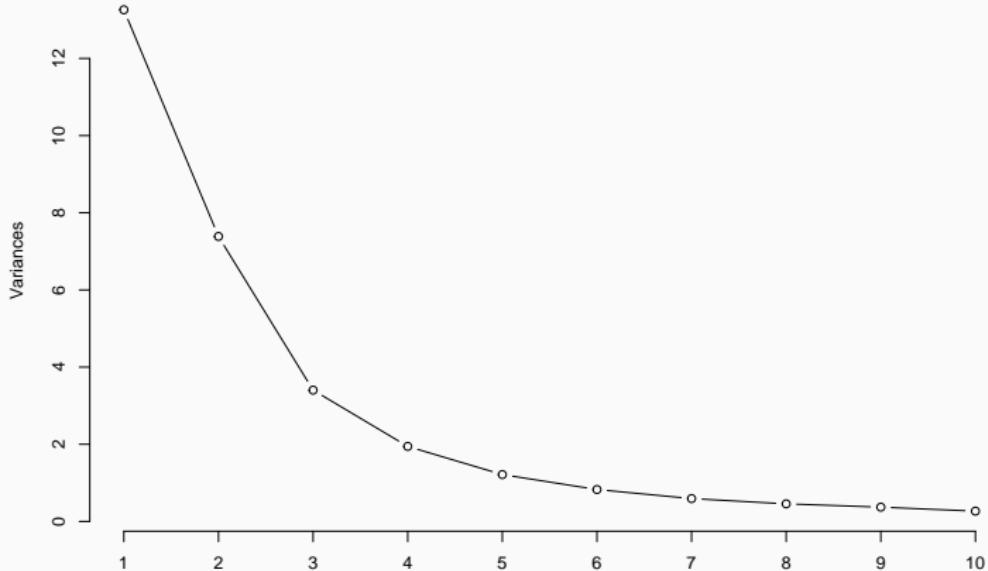
Properties of PCA ii

- Therefore, each linear transformation $w_i^T \mathbf{Y}$ contributes $\lambda_i / \sum_j \lambda_j$ as percentage of the overall variance.
- **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 Σ , we can use the (population) correlation matrix R .
- **Note:** The loadings and components we obtain from Σ are **not** equivalent to the ones obtained from R .

Sample PCA

- In general, we do not use the population covariance matrix Σ .
- 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

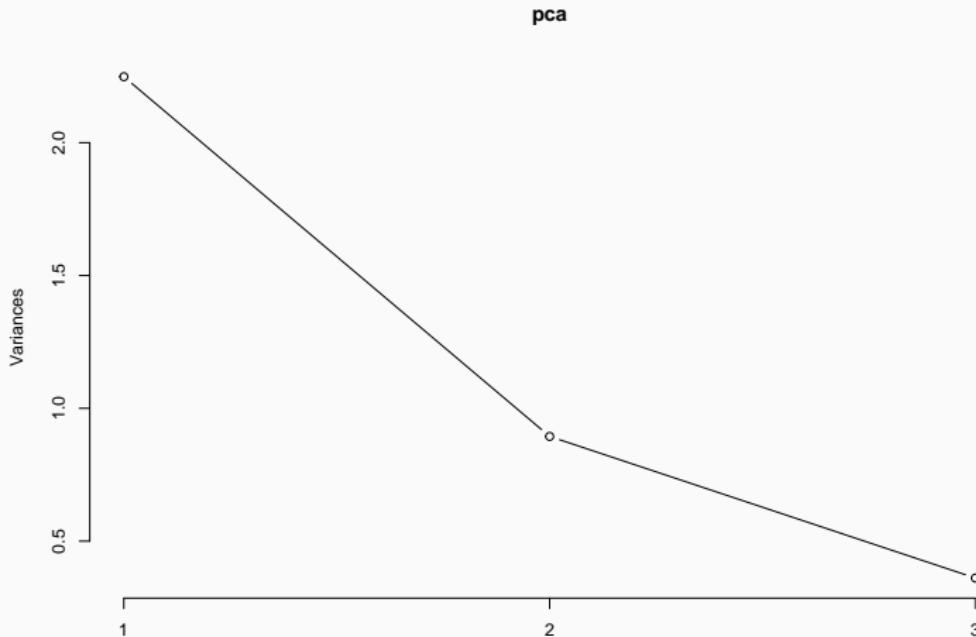
```
summary(pca)
```

```
## 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



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 0.41649
```

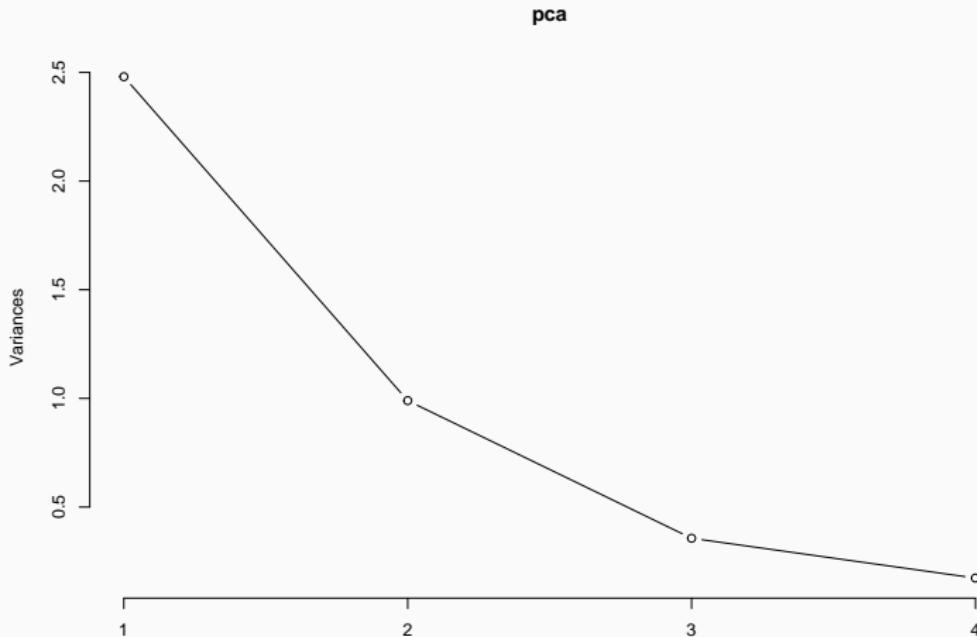
```
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
```

```
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

Example 2 ii

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

Example 2 iii



Applications of PCA

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

```
library(ElemStatLearn)
library(tidyverse)
train <- subset(prostate, train == TRUE,
                 select = -train)
test  <- subset(prostate, train == FALSE,
                 select = -train)

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

Feature Extraction ii

```
## [1] 0.521274

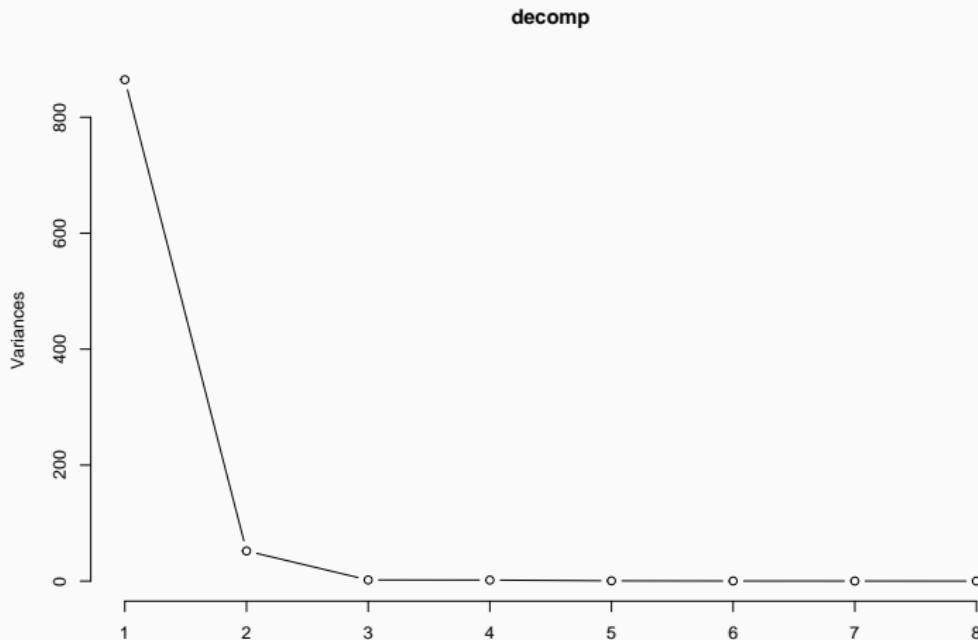
# PCA
decomp <- train %>%
  subset(select = -lpsa) %>%
  as.matrix() %>%
  prcomp
summary(decomp)$importance[, 1:3]
```

Feature Extraction iii

```
##                                     PC1        PC2        PC3
## Standard deviation      29.40597 7.211721 1.410789
## Proportion of Variance 0.93844 0.056440 0.002160
## Cumulative Proportion  0.93844 0.994890 0.997050
```

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

Feature Extraction iv



Feature Extraction v

```
# Second model: PCs for predictors
train_pc <- train
train_pc$PC1 <- decomp$x[,1]
pc_model <- lm(lpsa ~ PC1, data = train_pc)

test_pc <- as.data.frame(predict(decomp, test))
pc_pred <- predict(pc_model,
                     newdata = test_pc)
(pc_mse <- mean((test$lpsa - pc_pred)^2))

## [1] 0.9552741
```

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 = train[,c(keep, "lpsa")])  
fs_pred <- predict(fs_model, newdata = test)  
(fs_mse <- mean((test$lpsa - fs_pred)^2))  
  
## [1] 0.5815571
```

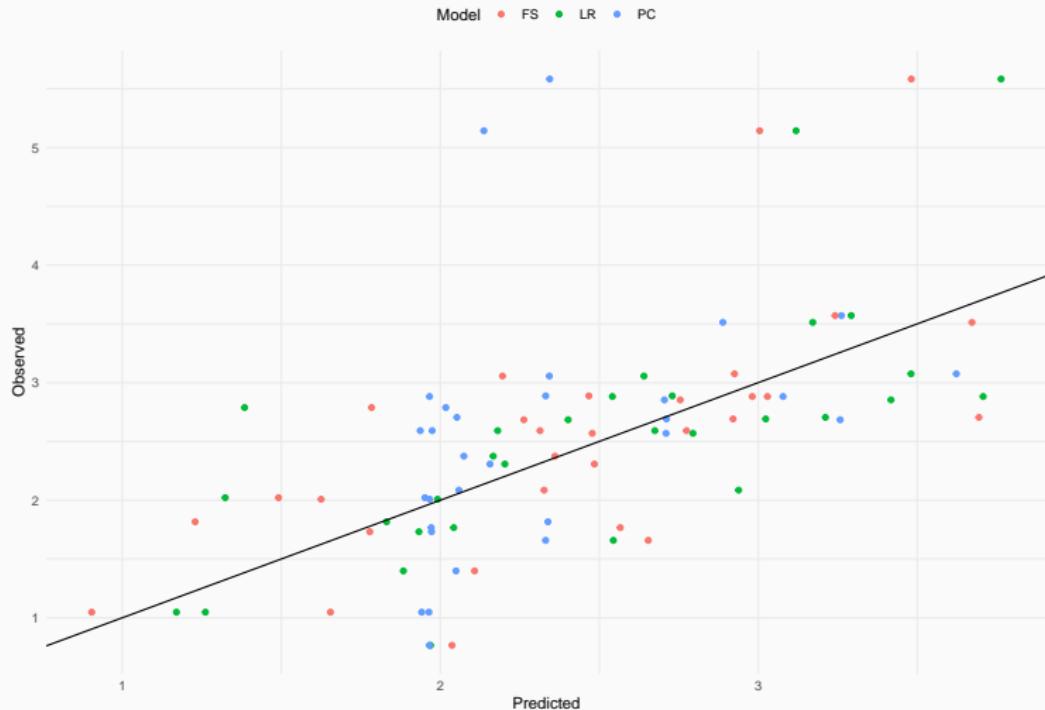
Feature Selection iii

```
model_plot <- data.frame(  
  "obs" = 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.

Data Visualization i

```
library(dslabs)

mnist <- read_mnist()

dim(mnist$train$images)

## [1] 60000    784

dim(mnist$test$images)

## [1] 10000    784
```

Data Visualization ii

```
head(mnist$train$labels)  
  
## [1] 5 0 4 1 9 2  
  
matrix(mnist$train$images[1,], ncol = 28) %>%  
  image(col = gray.colors(12, rev = TRUE),  
        axes = FALSE)
```

Data Visualization iii

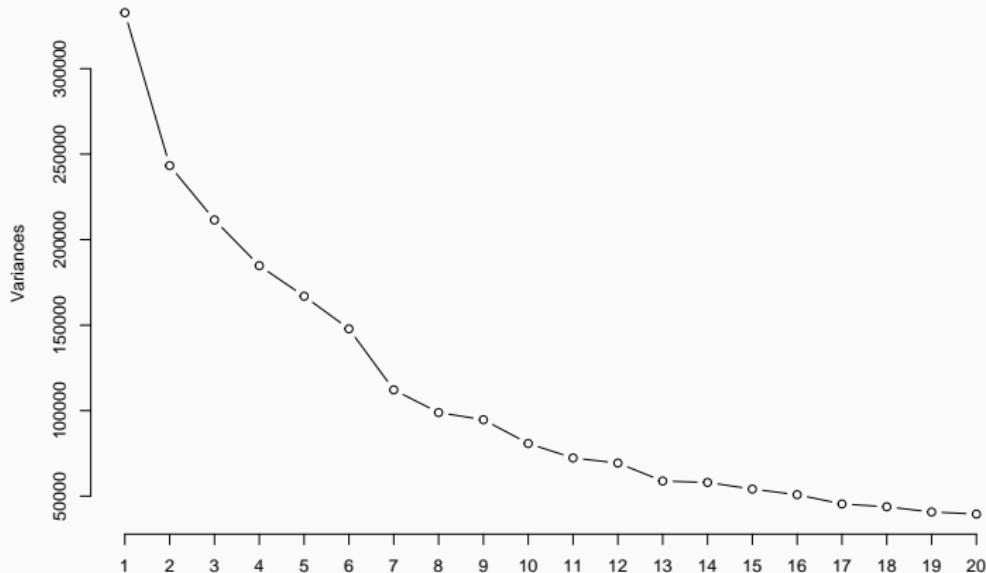


Data Visualization iv

```
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()
```

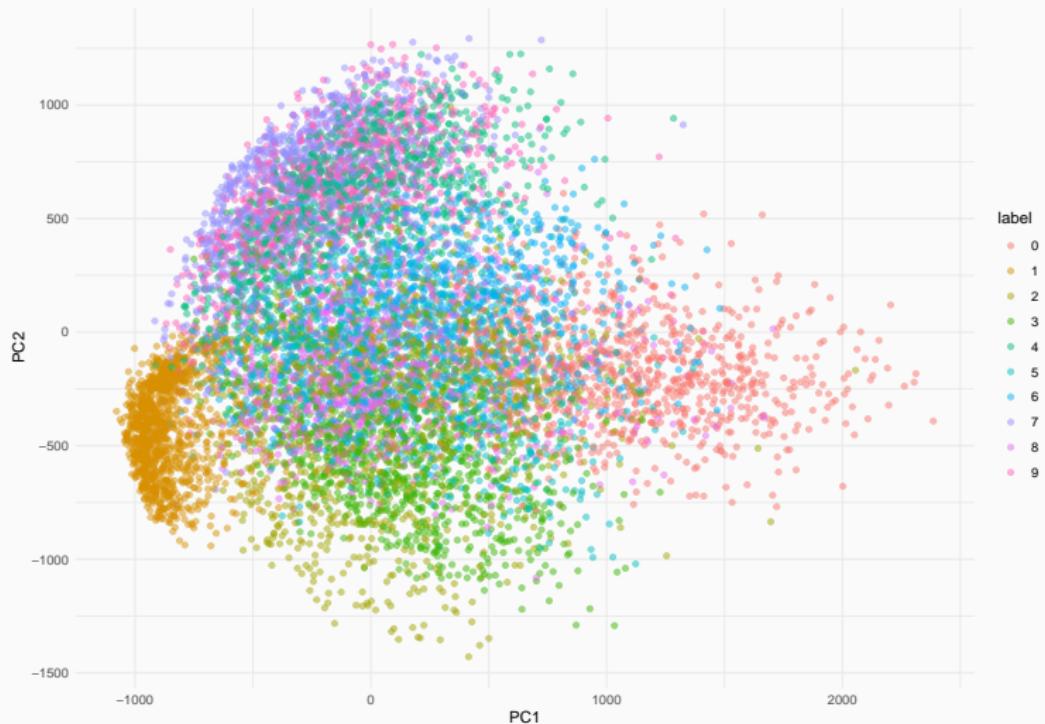
Data Visualization vii



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()
```

Data Visualization ix

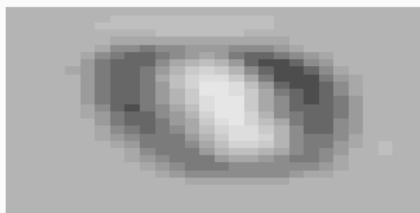


Data Visualization x

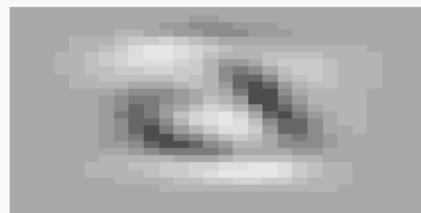
```
par(mfrow = c(2, 2))
for (i in seq_len(4)) {
  matrix(decomp$rotation[,i], ncol = 28) %>%
    image(col = gray.colors(12, rev = TRUE),
          axes = FALSE, main = paste0("PC", i))
}
```

Data Visualization xi

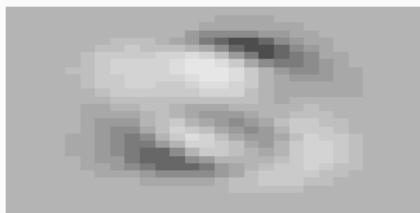
PC1



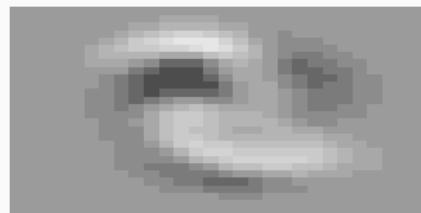
PC2



PC3



PC4



Data Visualization xii

```
# Approximation with 90 PCs
approx_mnist <- decomp$rotation[, seq_len(90)] %*%
  decomp$x[1, seq_len(90)]
par(mfrow = c(1, 2))

matrix(mnist$train$images[1,], ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE),
        axes = FALSE, main = "Original")
matrix(approx_mnist, ncol = 28) %>%
  image(col = gray.colors(12, rev = TRUE),
        axes = FALSE, main = "Approx")
```

Data Visualization xiii

Original



Approx



Additional comments about sample PCA i

- Let $\mathbf{Y}_1, \dots, \mathbf{Y}_n$ be a sample from a distribution with covariance matrix Σ . 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
## -6.169544e-17 5.433154e-17 9.228729e-18
```

On the other hand

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

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

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.

- We want to find μ, 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 μ, \mathbf{Z}_i , we find

$$\begin{aligned}\hat{\mu} &= \bar{\mathbf{Y}}, \\ \hat{\mathbf{Z}}_i &= W_k^T(\mathbf{Y}_i - \bar{\mathbf{Y}}).\end{aligned}$$

- 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 assumption 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 Σ positive definite. Let $\lambda_1 > \dots > \lambda_p$ be the eigenvalues of Σ ; 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 Λ to be the diagonal matrix whose entries are $\lambda_1, \dots, \lambda_p$, and define

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

Large sample inference iii

Asymptotic results

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

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

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 λ_i as follows:

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

Comments ii

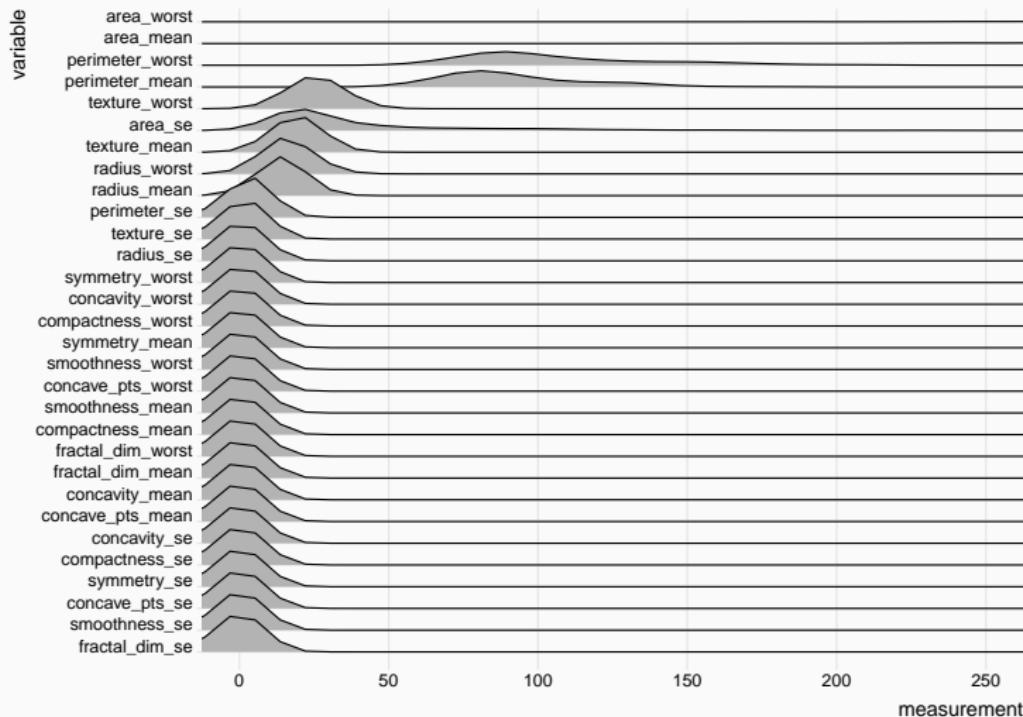
- Use Bonferroni correction if you want CIs that are simultaneously valid for all eigenvalues.
- The matrices Ω_i have rank $p - 1$, and therefore they are *singular*.
- The entries of \hat{w}_i are correlated, and this correlation depends on the *separation* between the eigenvalues.
 - Good separation \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

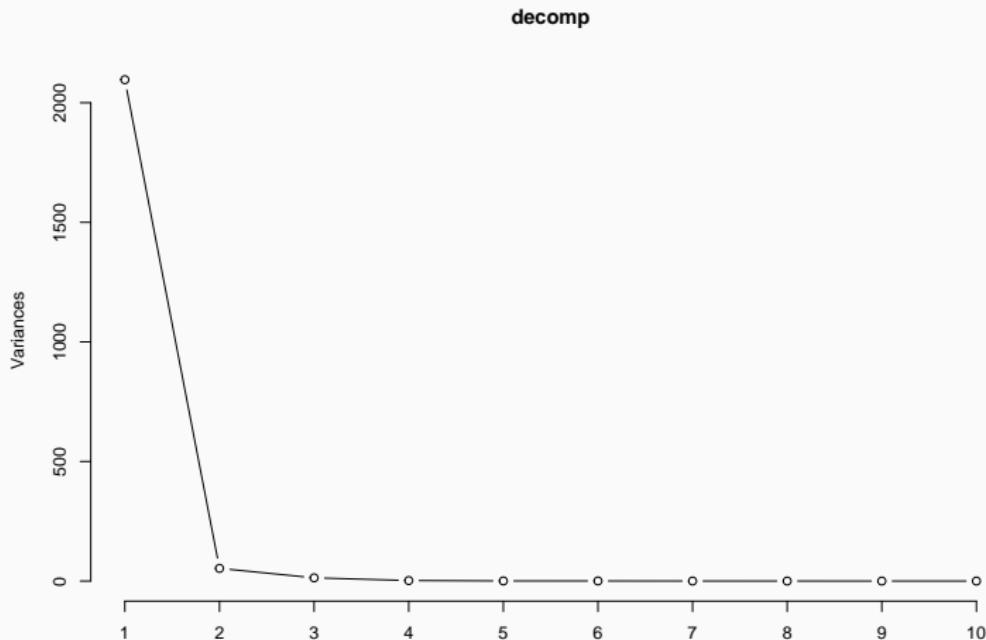
```
# Remove some variables
rem_index <- which(colnames(brca$x) %in%
                     c("area_worst", "area_mean",
                       "perimeter_worst",
                       "perimeter_mean"))
dataset <- brca$x[,-rem_index]
decomp <- prcomp(dataset)
summary(decomp)$importance[,1:3]
```

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],  

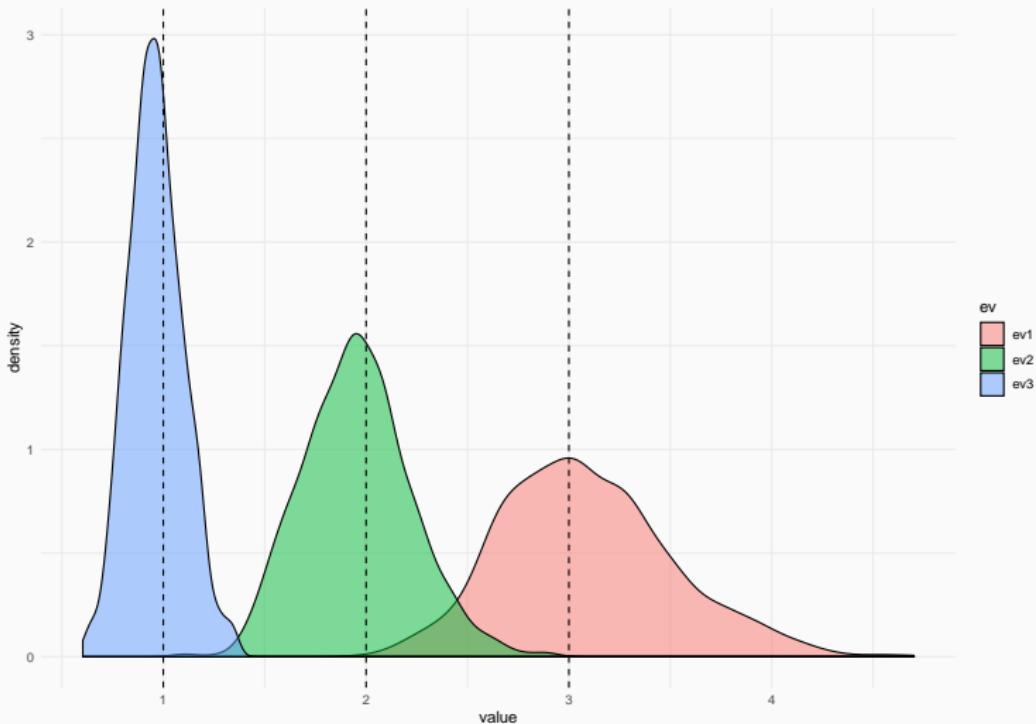
         ev2 = tmp$values[2],  

         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

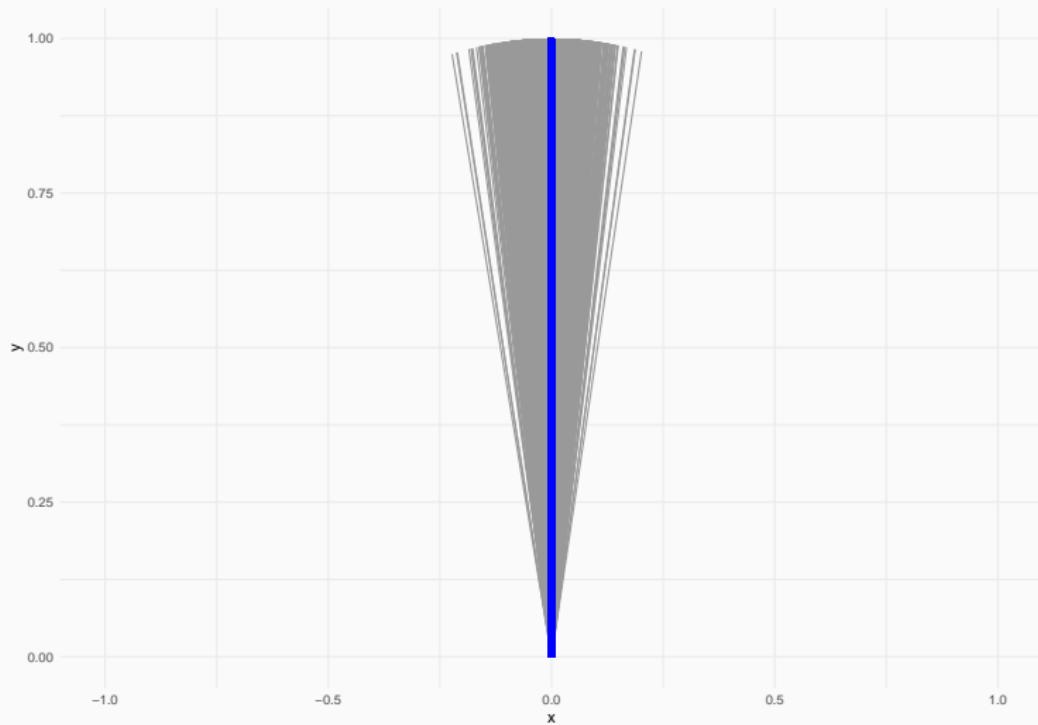
```
p <- 2
results <- purrr::map_df(seq_len(B), function(b) {
  X <- matrix(rnorm(p*n, sd = c(1, 2)), ncol = p,
              byrow = TRUE)
  tmp <- eigen(cov(X), symmetric = TRUE)

  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()
```

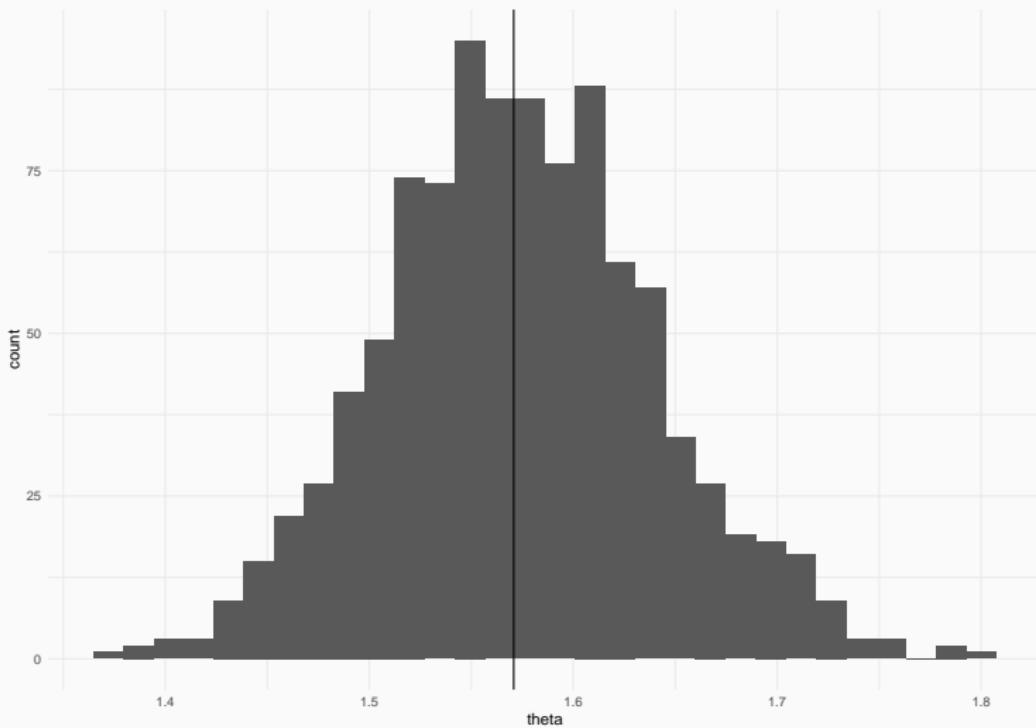
Simulations vii



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 Σ_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 Σ_1 .

Test for structured covariance iii

- 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.
- 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 Σ_0 if

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