DSA 8070 R Session 7: Inference for Covariance Matrix

Whitney Huang, Clemson University

Contents


```r
library(MASS)
library(corpcor)
library(fields)
library(PerformanceAnalytics)
library(ggplot2)
library(glasso)
library(boot)
library(Matrix)
```

Introduction

This notebook demonstrates covariance matrix estimation techniques, including:

- Sample covariance matrix
- Wishart distribution and inference
- Shrinkage estimation (Ledoit-Wolf)
- Sparse estimation (Graphical Lasso)

Simulating data

```r
# Simulate multivariate normal data
set.seed(123)
Sigma_true <- matrix(c(1, 0.8, 0.5,
                       0.8, 1, 0.3,
                       0.5, 0.3, 1), ncol = 3)
data <- mvrnorm(n = 100, mu = c(0, 0, 0), Sigma = Sigma_true)
colnames(data) <- c("X1", "X2", "X3")
head(data)
```

```
##             X1          X2          X3
## [1,] 0.2405360 -0.76694589 -1.09474911
## [2,] 0.1465984 -0.64944798 -0.08115701
## [3,] 1.4224811  1.53013080  0.88431809
## [4,] 0.2803955  0.06408771 -0.25760205
## [5,] 0.1289004  0.61572797 -0.57887479
## [6,] 1.4773313  1.63774739  1.15650887
```

Sample Covariance and Correlation

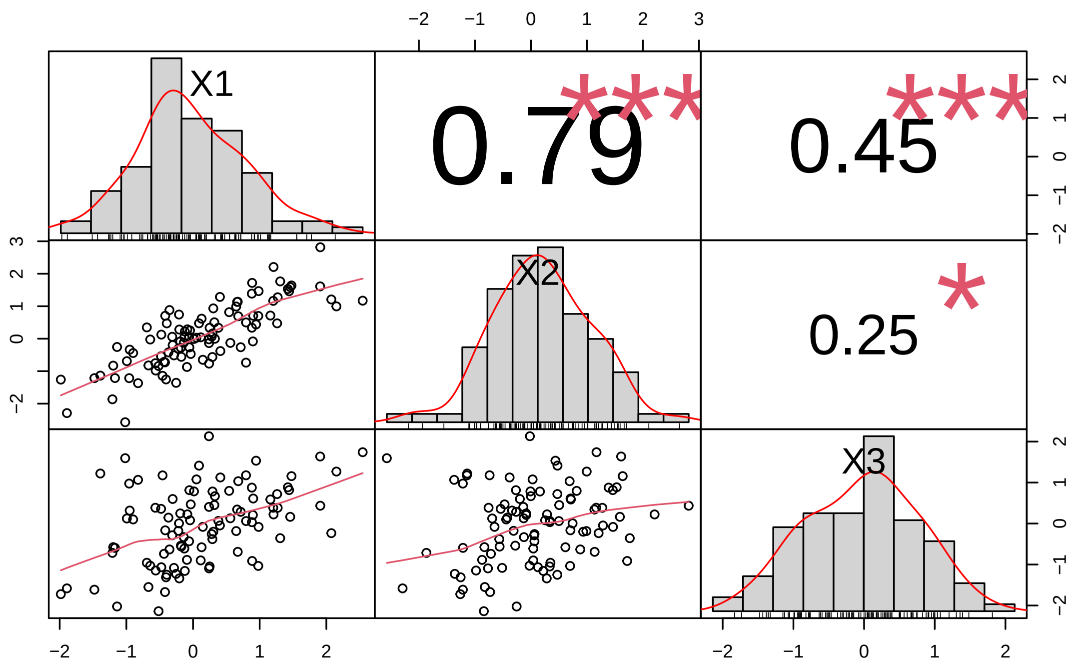
```r
(S_sample <- cov(data))
```

```
##           X1        X2        X3
## X1 0.7865232 0.6847376 0.3711648
## X2 0.6847376 0.9440958 0.2282530
## X3 0.3711648 0.2282530 0.8557030
```

```r
(R_sample <- cor(data))
```

```
##           X1        X2        X3
## X1 1.0000000 0.7946216 0.4524280
## X2 0.7946216 1.0000000 0.2539493
## X3 0.4524280 0.2539493 1.0000000
```

```r
chart.Correlation(data, histogram = TRUE)
```



Wishart Distribution

The *Wishart* distribution is the sampling distribution of the sample covariance matrix of a multivariate normal distribution. If $X_1, \ldots, X_n \sim N_p(\mu, \Sigma)$, then:

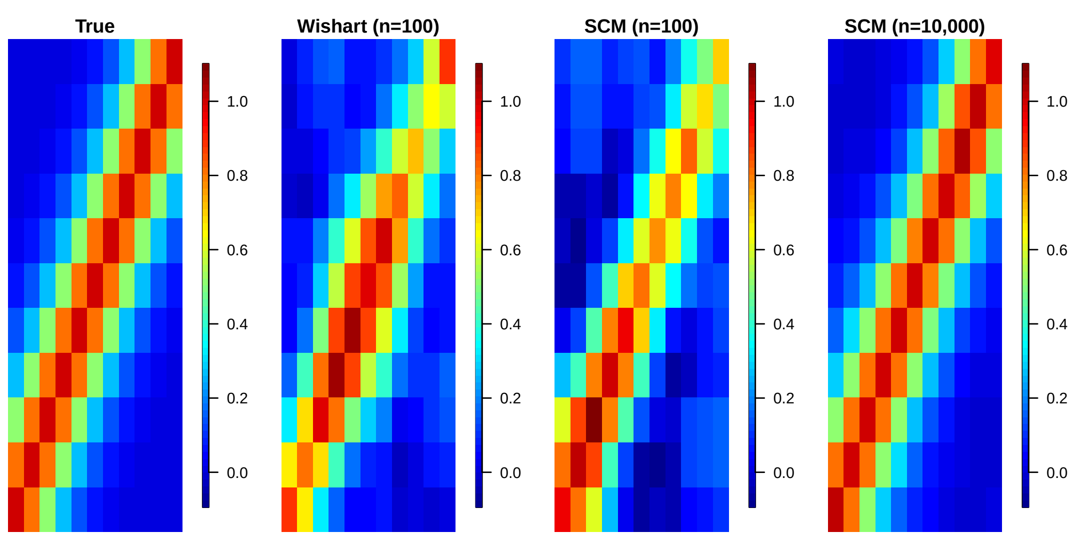
$$S = \sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})^\top \sim W_p(n-1, \Sigma)$$

Properties:

1. Mean: $\mathbb{E}[\mathbf{S}] = (n-1)\mathbf{\Sigma}$
2. Variance:$\mathrm{Var}[\mathbf{S}_{ij}] = (n-1)[\mathbf{\Sigma}_{ij} + \mathbf{\Sigma}_{ii}\mathbf{\Sigma}_{jj}]$
3. $\hat{\mathbf{\Sigma}}_n = \frac{\mathbf{S}}{n-1} \sim \frac{W_p(n-1,\mathbf{\Sigma})}{n-1} \Rightarrow$ fundamental for statistical inference on $\mathbf{\Sigma}$

```r
# ---- Grid and Matérn covariance ----
t <- seq(0, 1, 0.1)            # 11 grid points in [0,1]
p <- length(t)                 # dimension p = 11

# Wrapper for a Matérn covariance; 'pars' = (variance, range, smoothness)
cov.Matern <- function(h, pars) Matern(h, phi = pars[1], range = pars[2], smoothness = pars[3])

dist <- rdist(t)  # pairwise distances between grid points (11x11)
Sigma_Matern <- cov.Matern(dist, c(1, 0.1, 2))  # true covariance matrix (p x p)

# ---- Wishart samples and sample covariance (n = 100) ----
n <- 100
# rWishart(m, df, Sigma) returns an array p x p x m of Wishart draws
```

```
wishart_samples <- rWishart(100, df = n - 1, Sigma_Matern)  # 100 Wishart draws with df=99

set.seed(124)                    # reproducibility
# Simulate n i.i.d. N_p(0, Sigma_Matern) observations
data <- mvrnorm(n = 100, mu = rep(0, p), Sigma = Sigma_Matern)
S_sample <- cov(data)            # sample covariance matrix (SCM) for n=100

# ---- Larger-sample SCM (n = 10,000) for comparison ----
set.seed(124)
data1 <- mvrnorm(n = 10000, mu = rep(0, p), Sigma = Sigma_Matern)
S_sample1 <- cov(data1)          # SCM for n=10,000 (closer to truth)

# ---- Visualization setup ----
par(mfrow = c(1, 4),                    # 1 row, 4 panels
    mar = c(1.5, 1.5, 1.5, 4),
    mgp = c(2, 1, 0), font = 3)

# Common color scale across panels for fair comparison.
# Wishart(df, Sigma) has E[ W / df ] = Sigma, so we scale by df (=99 here).
zlim <- range(S_sample)

# ---- Panels: true Sigma vs. estimates ----
image.plot(Sigma_Matern, zlim = zlim, axes = FALSE, main = "True",
           legend.mar = 2, legend.cex = 0.5)

image.plot(wishart_samples[,, 1] / 99, axes = FALSE, zlim = zlim,
           legend.mar = 2, main = "Wishart (n=100)", legend.cex = 0.5)
# ^ One Wishart draw scaled by df to be on the Sigma scale.

image.plot(S_sample, zlim = zlim, axes = FALSE, main = "SCM (n=100)",
           legend.mar = 2, legend.cex = 0.5)

image.plot(S_sample1, zlim = zlim, axes = FALSE, legend.mar = 2,
           main = "SCM (n=10,000)", legend.cex = 0.5)
```



Statistical Inference

Bootstrap Confidence Intervals for Covariance Elements

```
boot_cov <- function(data, indices) {
  d <- data[indices, ]
  return(cov(d)[1, 2])  # Cov(X1, X2)
}

set.seed(123)
boot_obj <- boot(data, statistic = boot_cov, R = 1000)
boot.ci(boot_obj, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_obj, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.5859,  1.1292 )
## Calculations and Intervals on Original Scale
```

Hypothesis Test for Covariance Matrix

```
# H0: Sigma = Identity
n <- nrow(data)
p <- ncol(data)
Sigma0 <- diag(p)
LRT_stat <- n * (log(det(Sigma0)) - log(det(S_sample)) + sum(diag(solve(Sigma0) %*% S_sample)) - p)
LRT_stat
```

```
## [1] 1266.969
```

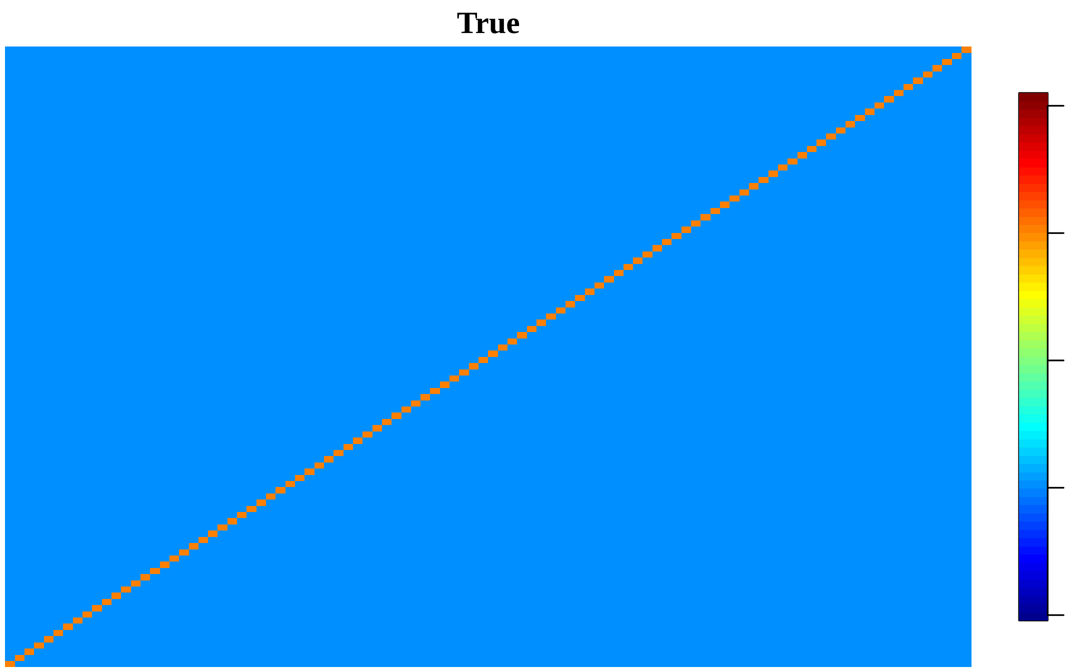
```
# Asymptotic chi-square distribution (df = p(p+1)/2)
df <- p * (p + 1) / 2
p_value <- 1 - pchisq(LRT_stat, df)
p_value
```

```
## [1] 0
```

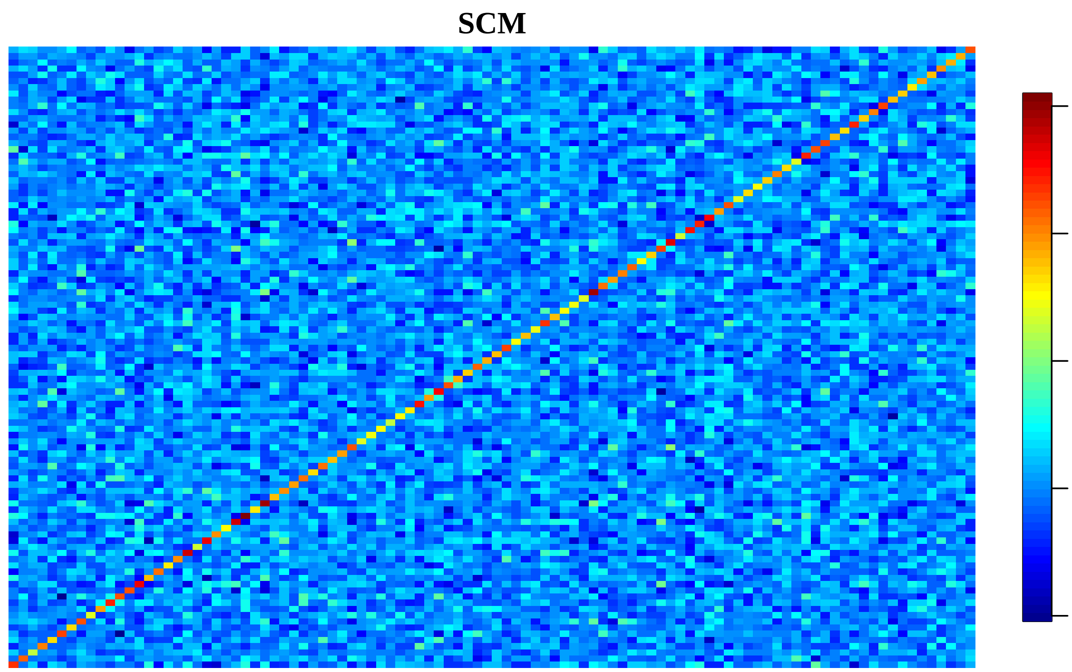
High-Dimensional Case: SCM Breakdown

Simulate high-dimensional case where $n = 50$ but the dimension $p = 100$ is “high” w.r.t. sample size here

```r
# Simulate high-dimensional case: n = 50, p = 100
set.seed(456)
data_hd <- mvrnorm(n = 50, mu = rep(0, 100), Sigma = diag(100))
S_hd <- cov(data_hd)

par(mar = c(1.5, 1.5, 1.5, 2), mgp = c(2, 1, 0), font = 3,
    family = "serif")
zlim <- range(S_hd)
image.plot(diag(100), zlim = zlim, axes = F, main = "True",
           legend.mar = 1, legend.cex = 0.25, legend.shrink = 0.85)
```



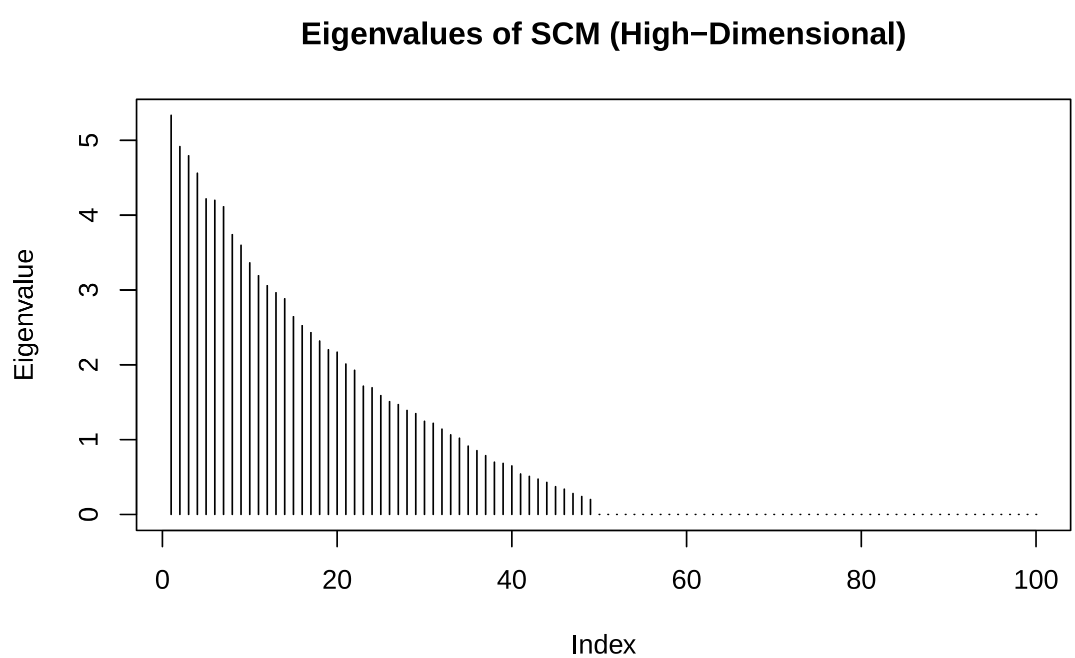

```r
image.plot(S_hd, zlim = zlim, axes = F, main = "SCM",
           legend.mar = 1, legend.cex = 0.25, legend.shrink = 0.85)
```



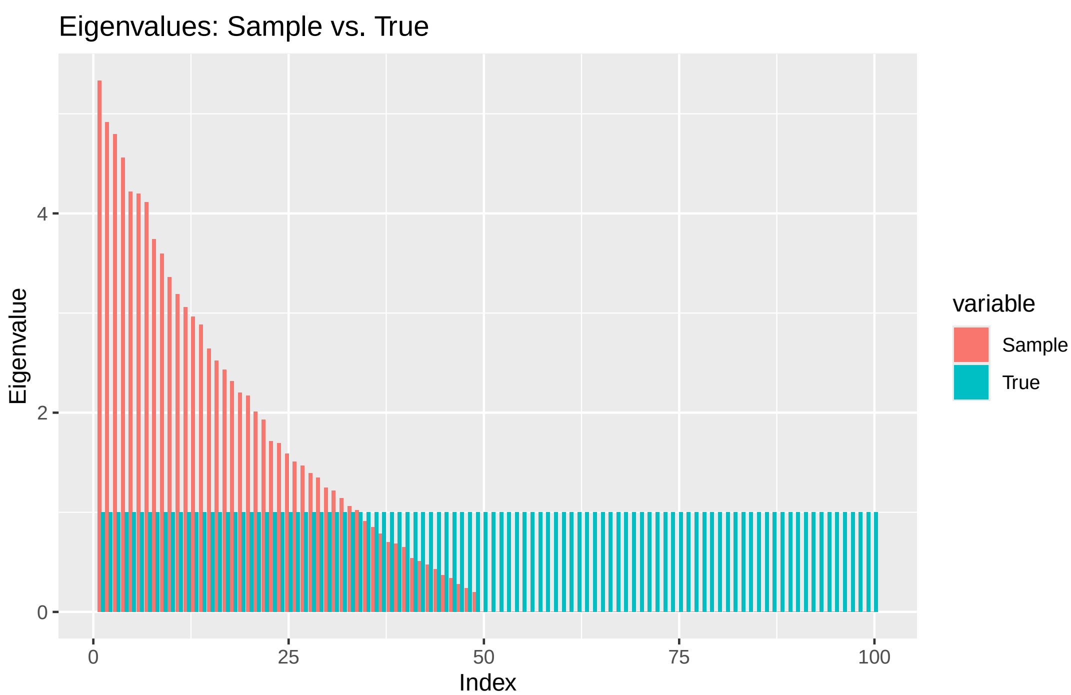

```r
eigen_hd <- eigen(S_hd)$values
summary(eigen_hd)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.9751  1.5281  5.3342
```

```r
plot(eigen_hd, main = "Eigenvalues of SCM (High-Dimensional)", ylab = "Eigenvalue", type = "h")
```



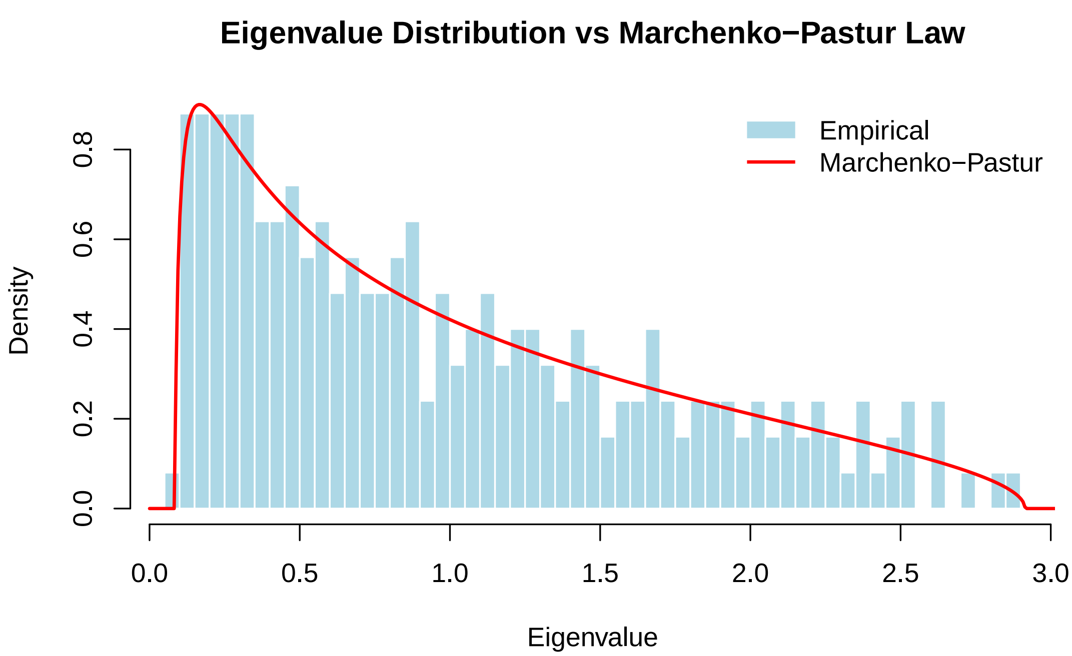
```r
eig_scm <- eigen(S_hd)$values
eig_true <- eigen(diag(100))$values
df_eig <- data.frame(Index = 1:100,
                     Sample = eig_scm,
                     True = eig_true)

library(reshape2)
df_melt <- melt(df_eig, id = "Index")

p <- ggplot(df_melt, aes(x = Index, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Eigenvalues: Sample vs. True", y = "Eigenvalue")

print(p)
```



Marchenko–Pastur law

```r
# Eigenvalue distribution vs Marchenko-Pastur law
set.seed(123)

# Dimensions
n <- 500   # sample size
p <- 250   # dimension (so p/n = 0.5)

# Generate data from multivariate normal with Sigma = I
X <- matrix(rnorm(n * p), nrow = n, ncol = p)

# Sample covariance matrix (scaled by 1/n)
S <- (1/n) * t(X) %*% X
```

```r
# Eigenvalues
eigvals <- eigen(S, symmetric = TRUE, only.values = TRUE)$values

# Marchenko-Pastur density function
mp_density <- function(x, c) {
  a <- (1 - sqrt(c))^2
  b <- (1 + sqrt(c))^2
  dens <- rep(0, length(x))
  inside <- (x >= a & x <= b)
  dens[inside] <- sqrt((b - x[inside]) * (x[inside] - a)) / (2 * pi * c * x[inside])
  dens
}

c_ratio <- p/n
x_grid <- seq(0, max(eigvals) * 1.1, length.out = 500)
mp_dens <- mp_density(x_grid, c_ratio)

hist(eigvals, breaks = 50, freq = FALSE, col = "lightblue",
     border = "white", main = "Eigenvalue Distribution vs Marchenko-Pastur Law",
     xlab = "Eigenvalue")
lines(x_grid, mp_dens, col = "red", lwd = 2)
legend("topright", legend = c("Empirical", "Marchenko-Pastur"),
       col = c("lightblue", "red"), lwd = c(10, 2), bty = "n", pch = c(15, NA))
```



Shrinkage Example: Ledoit-Wolf vs Sample Covariance

```r
set.seed(123)

library(corpcor)  # for cov.shrink (Ledoit-Wolf shrinkage)
```

```r
# Parameters
p <- 50
n <- 30

# Simulate data: multivariate normal with identity covariance
X <- matrix(rnorm(n * p), nrow = n, ncol = p)

# Sample covariance matrix
S <- cov(X)

# Ledoit-Wolf shrinkage covariance
S_shrink <- cov.shrink(X)
```

```
## Estimating optimal shrinkage intensity lambda.var (variance vector): 1
##
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.9931
```

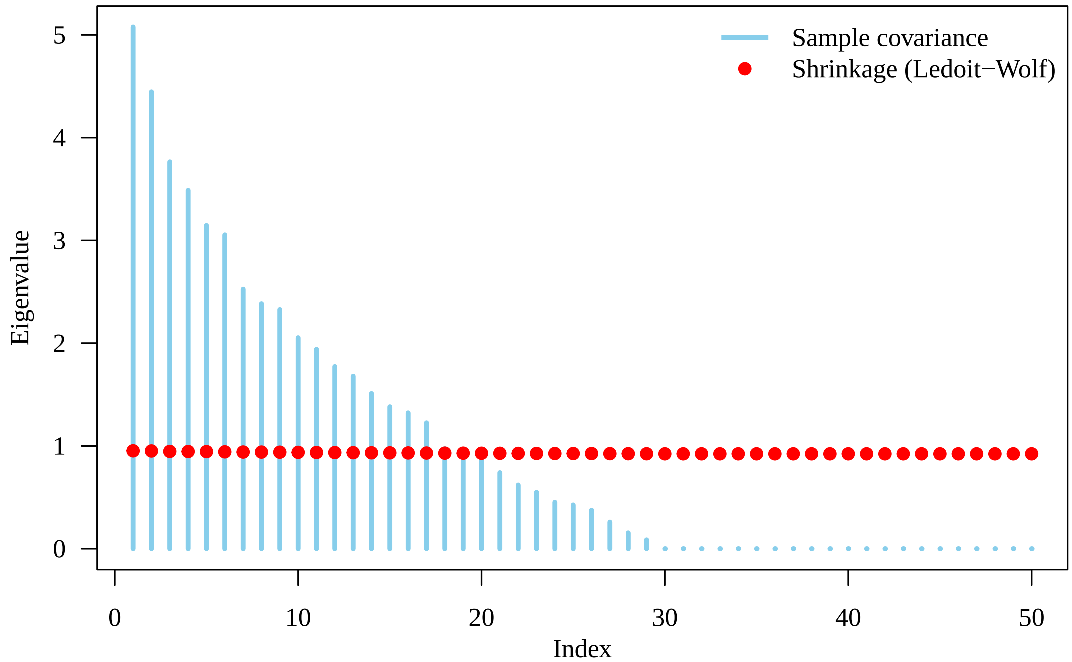
```r
# Eigenvalues
eig_sample <- eigen(S, symmetric = TRUE, only.values = TRUE)$values
eig_shrink <- eigen(S_shrink, symmetric = TRUE, only.values = TRUE)$values

# Condition numbers
cond_sample <- max(eig_sample) / min(eig_sample)
cond_shrink <- max(eig_shrink) / min(eig_shrink)

cat("Condition number (sample):", cond_sample, "\n")
```

```
## Condition number (sample): -4.027124e+15
```

```r
cat("Condition number (shrinkage):", cond_shrink, "\n")
```

```
## Condition number (shrinkage): 1.031591
```

```r
# Plot eigenvalues
par(mar = c(3.5, 3.5, 1, 0.5), mgp = c(2, 1, 0), family = "serif",
    las = 1)
plot(eig_sample, type = "h", lwd = 3, col = "skyblue",
     ylim = c(0, max(c(eig_sample, eig_shrink))),
     xlab = "Index", ylab = "Eigenvalue",
     main = "")
points(eig_shrink, col = "red", pch = 19)
legend("topright", legend = c("Sample covariance", "Shrinkage (Ledoit-Wolf)"),
       col = c("skyblue", "red"), lwd = c(3, NA), pch = c(NA, 19), bty = "n")
```



Shrinkage Estimation (Ledoit-Wolf)

```r
S_shrink <- cov.shrink(data, lambda = 0.2, verbose = TRUE)
```

```
## Estimating optimal shrinkage intensity lambda.var (variance vector): 0.8628
##
## Specified shrinkage intensity lambda (correlation matrix): 0.2
```

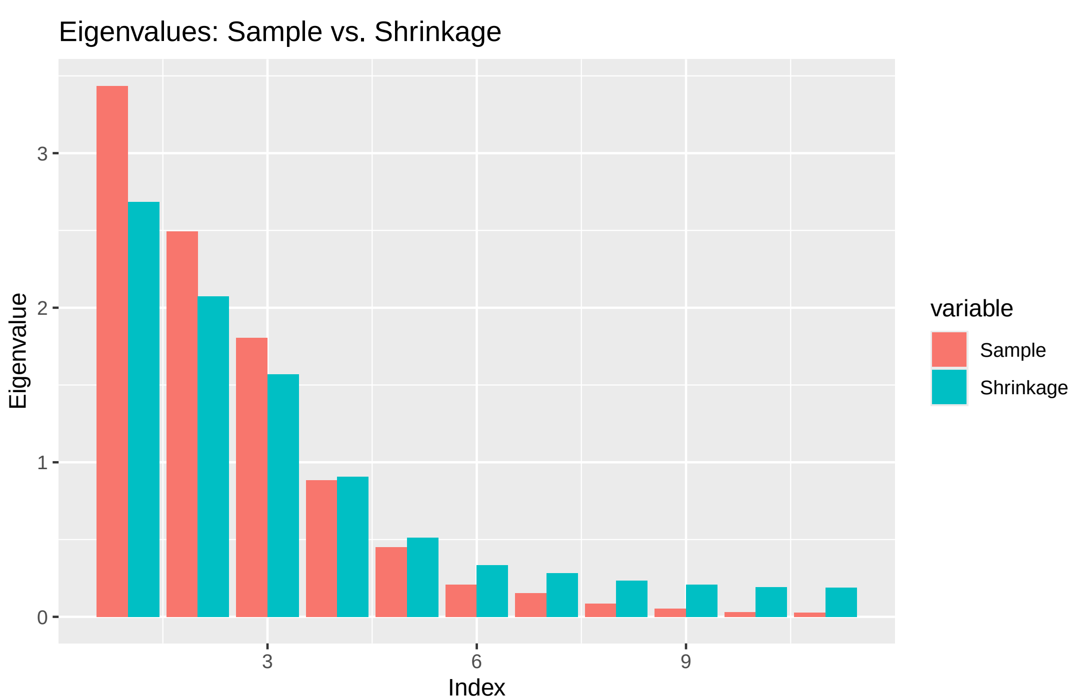
```r
S_shrink
```

```
##              [,1]        [,2]          [,3]        [,4]        [,5]        [,6]
##  [1,]  0.84714490  0.55832958  0.4018342168  0.19070994 0.022876275 -0.05709991
##  [2,]  0.55832958  0.85487403  0.5684408262  0.28926751 0.081807202 -0.04294704
##  [3,]  0.40183422  0.56844083  0.8647851336  0.51836983 0.297117763  0.09919590
##  [4,]  0.19070994  0.28926751  0.5183698261  0.85238768 0.555992749  0.31056215
##  [5,]  0.02287627  0.08180720  0.2971177632  0.55599275 0.845873697  0.53087671
##  [6,] -0.05709991 -0.04294704  0.0991958959  0.31056215 0.530876714  0.82612388
##  [7,] -0.02862804 -0.06381759  0.0006732619  0.08560092 0.247321428  0.49677857
##  [8,] -0.03400519 -0.02861341 -0.0136763742 -0.04315885 0.044459528  0.27598647
##  [9,]  0.02932403  0.09070457  0.0887433889 -0.01566876 0.005757439  0.13678750
## [10,]  0.05328648  0.11757578  0.1082462348  0.04944503 0.057607184  0.10759118
## [11,]  0.07820854  0.12295560  0.1171212567  0.07138615 0.102612708  0.12347245
##                [,7]        [,8]         [,9]       [,10]      [,11]
##  [1,] -0.0286280402 -0.03400519  0.029324027 0.05328648 0.07820854
##  [2,] -0.0638175902 -0.02861341  0.090704565 0.11757578 0.12295560
##  [3,]  0.0006732619 -0.01367637  0.088743389 0.10824623 0.11712126
##  [4,]  0.0856009235 -0.04315885 -0.015668762 0.04944503 0.07138615
##  [5,]  0.2473214280  0.04445953  0.005757439 0.05760718 0.10261271
##  [6,]  0.4967785659  0.27598647  0.136787498 0.10759118 0.12347245
```

```
##  [7,]  0.8224395698  0.52631738  0.302628850 0.11688104 0.05746415
##  [8,]  0.5263173759  0.82473903  0.519816006 0.28383147 0.17481116
##  [9,]  0.3026288504  0.51981601  0.828653385 0.50605525 0.30560032
## [10,]  0.1168810411  0.28383147  0.506055250 0.80786826 0.46873426
## [11,]  0.0574641452  0.17481116  0.305600318 0.46873426 0.81103568
## attr(,"lambda")
## [1] 0.2
## attr(,"lambda.estimated")
## [1] FALSE
## attr(,"class")
## [1] "shrinkage"
## attr(,"lambda.var")
## [1] 0.862776
## attr(,"lambda.var.estimated")
## [1] TRUE
```

Compare Eigenvalues

```r
eig_sample <- eigen(S_sample)$values
eig_shrink <- eigen(S_shrink)$values
df_eig <- data.frame(Index = 1:11,
                     Sample = eig_sample,
                     Shrinkage = eig_shrink)

library(reshape2)
df_melt <- melt(df_eig, id = "Index")

ggplot(df_melt, aes(x = Index, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Eigenvalues: Sample vs. Shrinkage", y = "Eigenvalue")
```

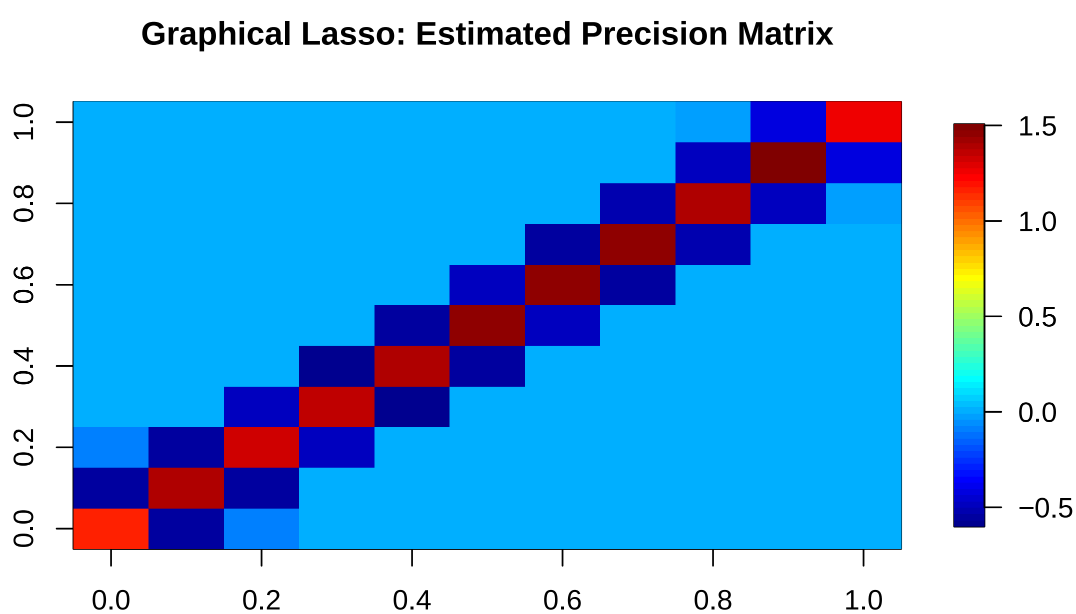


Sparse Estimation (Graphical Lasso)

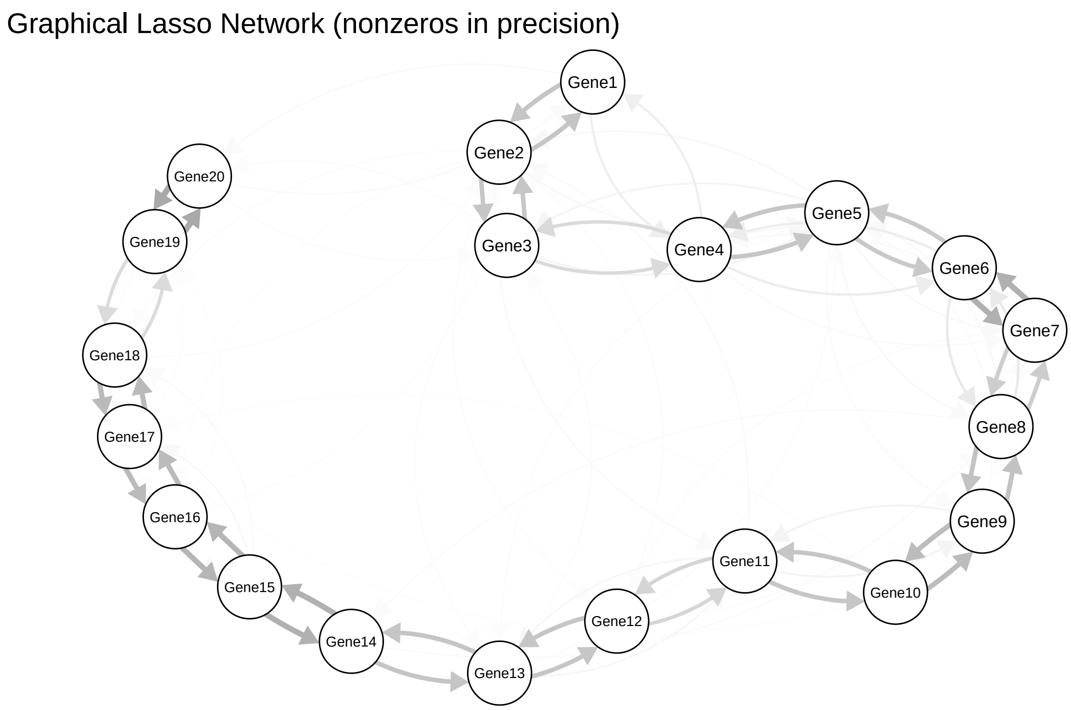
```r
S <- cov(data)
S_glasso <- glasso(S, rho = 0.2)
S_glasso$wi  # Precision matrix
```

```
##              [,1]       [,2]        [,3]       [,4]       [,5]       [,6]
##  [1,]  1.17510177 -0.5453156 -0.08190783  0.0000000  0.0000000  0.0000000
##  [2,] -0.54531584  1.4022764 -0.55978201  0.0000000  0.0000000  0.0000000
##  [3,] -0.08190066 -0.5597838  1.31369962 -0.4889460  0.0000000  0.0000000
##  [4,]  0.00000000  0.0000000 -0.48895530  1.3639429 -0.5845411  0.0000000
##  [5,]  0.00000000  0.0000000  0.00000000 -0.5845449  1.4044481 -0.5424634
##  [6,]  0.00000000  0.0000000  0.00000000  0.0000000 -0.5424608  1.4475223
##  [7,]  0.00000000  0.0000000  0.00000000  0.0000000  0.0000000 -0.4802424
##  [8,]  0.00000000  0.0000000  0.00000000  0.0000000  0.0000000  0.0000000
##  [9,]  0.00000000  0.0000000  0.00000000  0.0000000  0.0000000  0.0000000
## [10,]  0.00000000  0.0000000  0.00000000  0.0000000  0.0000000  0.0000000
## [11,]  0.00000000  0.0000000  0.00000000  0.0000000  0.0000000  0.0000000
##             [,7]       [,8]        [,9]      [,10]       [,11]
##  [1,]  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
##  [2,]  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
##  [3,]  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
##  [4,]  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
##  [5,]  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
##  [6,] -0.4802518  0.0000000  0.00000000  0.0000000  0.00000000
##  [7,]  1.4510863 -0.5429077  0.00000000  0.0000000  0.00000000
##  [8,] -0.5429045  1.4663876 -0.52683235  0.0000000  0.00000000
##  [9,]  0.0000000 -0.5268316  1.38285390 -0.4875178 -0.03760863
## [10,]  0.0000000  0.0000000 -0.48751768  1.4902885 -0.41015602
## [11,]  0.0000000  0.0000000 -0.03760863 -0.4101560  1.25323238
```

```r
image.plot(S_glasso$wi, main = "Graphical Lasso: Estimated Precision Matrix")
```



```r
# Graphical Lasso Application: simulate, fit, and plot network
set.seed(1)

library(glasso)   # graphical lasso
library(qgraph)   # easy network plotting

# ---- Simulate gene expression data (p = 20 genes) ----
p <- 20
n <- 120

# AR(1)-type covariance to mimic correlated genes
rho <- 0.6
Sigma <- rho ^ abs(outer(1:p, 1:p, "-"))

# Simulate N(0, Sigma) data
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma)

# Standardize to zero mean, unit variance (scale-invariant graph)
Xsc <- scale(X)
S <- cov(Xsc)

# ---- Graphical lasso (sparse precision estimate) ----
# Tune rho (penalty) to control sparsity (try 0.05-0.25)
lambda <- 0.12
fit <- glasso(s = S, rho = lambda)

Theta_hat <- fit$wi     # sparse precision matrix (inverse covariance)
Adj <- (abs(Theta_hat) > 1e-6)         # adjacency from nonzeros
diag(Adj) <- FALSE                      # remove self-loops

# Edge weights for plotting (use absolute precision off-diagonals)
W <- abs(Theta_hat)
diag(W) <- 0

# ---- Plot network ----
# Node labels: Gene1, Gene2, ...
gene_labels <- paste0("Gene", 1:p)

qgraph(W,
       labels = gene_labels,
       layout = "spring",          # force-directed layout
       cut = 0,                    # show all nonzero edges
       edge.color = "darkgray",
       asize = 4,                  # arrow size (kept small; undirected visual)
       vsize = 6,                  # node size
       esize = 4,                  # edge scaling
       minimum = min(W[W > 0]),    # normalize edge thickness
       theme = "colorblind",
       title = "Graphical Lasso Network (nonzeros in precision)")
```




```
# ---- Simple summary for your notes/console ----
num_edges <- sum(Adj[upper.tri(Adj)])
cat("Nonzero off-diagonal entries (edges):", num_edges, "\n")
```

```
## Nonzero off-diagonal entries (edges): 50
```

```
cat("Saved figure: glasso_network.pdf\n")
```

```
## Saved figure: glasso_network.pdf
```

Summary

This R Markdown covered key concepts and tools for covariance estimation including:

- Sample covariance and correlation
- Wishart distribution and inference
- Bootstrap and likelihood-based inference
- Regularized covariance estimation (shrinkage)
- Sparse precision estimation and motivation in high-dimensional settings