

# STAT 3690 Lecture Note

Week One (Jan 9, 11, & 13, 2023)

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca, zhiyanggeezhou.github.io)

2023/Jan/13 11:59:23

---

## IN THE CASE OF A FIRE ALARM:

- **Remain calm**
  - if it is safe, evacuate the classroom or lab
  - go to the closest fire exit
  - do not use the elevators
- **If you need assistance to evacuate the building, inform your professor or instructor immediately.**
- **If you need to report an incident or a person left behind during a building evacuation, report it to a fire warden or call security services 204-474-9341.**
  - **Do not** reenter the building until the “all clear” is declared by a fire warden, security services or the fire department.
- **Important: only those trained in the use of a fire extinguisher should attempt to operate one!**



## Syllabus

### Contact

- Instructor: Zhiyang (Gee) Zhou, PhD, Asst. Prof.
  - Email: zhiyang.zhou@umanitoba.ca
  - Homepage: zhiyanggeezhou.github.io
- Marker: Mr. Masudul Islam
  - Email: islamm8@myumanitoba.ca

### Timeline

- Lectures
  - Mon/Wed/Fri 9:30–10:20 am
- Office Hour
  - Wed 10:30–11:30 am
- Assessments
  - 4 or 5 Assignments
  - Midterm
  - Final project

## Grading

- Assignments (30%)
  - Scanned/photographed and submitted to Crowdmark
  - Attaching both outputs and source codes (if applicable)
  - Including necessary interpretation
  - Organized in a clear and readable way
  - Accepting NO late submission
- Midterm (35%)
  - Open-book
  - In-person on Mar 10 6–8 pm OR take-home (webcam-invigilated) NOT later than Mar. 20
- Final project (35%)
  - Individual report analyzing recently collected datasets
  - See the Project Guideline posted at UM Learn

## Materials

- Reading list (recommended but not required)
  - [J&W] R. A. Johnson & D. W. Wichern. (2007). *Applied Multivariate Statistical Analysis*, 5/6th Ed. London: Pearson Education.
    - \* 2HR print reserve in the Sciences and Technology Library
  - [R&C] A. C. Rencher & W. F. Christensen. (2012). *Methods of Multivariate Analysis*, 3rd Ed. Hoboken: Wiley.
    - \* Digital copy accessible via the library
  - D. Salsburg (2001). *The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century*. New York: WH Freeman.
- Lecture notes and beyond
  - zhiyanggeezhou.github.io
  - UM Learn

## Outline

- Topics to be covered
  - Matrix manipulation
  - Basics of statistical modeling
  - Multivariate normal distribution
  - Inference on a mean vector
  - Comparisons of several multivariate means
  - Multivariate linear regression
  - Principal component analysis
  - Factor analysis
  - Canonical correlation analysis
  - and so forth

## R basics

- Installation
  - download and install BASE *R* from <https://cran.r-project.org>
  - download and install *Rstudio* from <https://www.rstudio.com>
  - download and install packages via *Rstudio*
- Working directory
  - When you ask *R* to open a certain file, it will look in the working directory for this file.
  - When you tell *R* to save a data file or figure, it will save it in the working directory.

```
getwd()
mainDir <- "c:/"
subDir <- "stat3690"
dir.create(file.path(mainDir, subDir), showWarnings = FALSE)
setwd(file.path(mainDir, subDir))
```

- Packages
  - installation: `install.packages()`
  - loading: `library()`

```
install.packages('nlme')
library(nlme)
```

- Help manual: `help()`, `?`, google, stackoverflow, etc.

- 
- *R* is free but not cheap
    - Open-source
    - Citing packages
    - NO quality control
    - Requiring statistical sophistication
    - Time-consuming to become a master

- 
- References for *R*
    - M. L. Rizzo (2019) Statistical Computing with R, 2nd Ed. (forthcoming)
    - O. Jones, R. Maillardet, A. Robinson (2014) Introduction to Scientific Programming and Simulation Using R, 2nd Ed.
    - .....
  - Courses online
    - <https://www.pluralsight.com/search?q=R>
    - .....
  - Data types: let `str()` or `class()` tell you
    - numbers (integer, real, or complex)
    - characters (“abc”)
    - logical (TRUE or FALSE)
    - date & time
    - factor (commonly encountered in this course)
    - NA (different from Inf, “ ”, 0, NaN etc.)

- 
- Data structures: let `str()` or `class()` tell you
    - vector: an ordered collection of the same data type
    - matrix: two-dimensional collection of the same data type
    - array: more than two dimensional collection of the same data type
    - data frame: collection of vectors of same length but of arbitrary data types
    - list: collection of arbitrary objects

- 
- Data input and output
    - create
      - \* vector: `c()`, `seq()`, `rep()`
      - \* matrix: `matrix()`, `cbind()`, `rbind()`
      - \* data frame

- output: write.table(), write.csv(), write.xlsx()
- import: read.table(), read.csv(), read.xlsx()
  - \* header: whether or not assume variable names in first row
  - \* stringsAsFactors: whether or not convert character string to factors
- scan(): a more general way to input data
- save.image() and load(): save and reload workspace
- source(): run R script

- 
- Parenthesis in R
    - parenthesis () to enclose inputs for functions
    - square brackets [], [[]] for indexing
    - braces {} to enclose forloop or statements such as if or ifelse
- 

```
# Create numeric vectors
```

```
v1 = c(1,2,3); v1
v2 = seq(4,6,by=0.5); v2
v3 = c(v1,v2); v3
v4 = rep(pi,5); v4
v5 = rep(v1,2); v5
v6 = rep(v1,each=2); v6
# Create Character vector
v7 <- c("one", "two", "three"); v7
# Select specific elements
v1[c(1,3)]
v7[2]
```

```
# Create matrices
```

```
m1 = matrix(-1:4, nrow=2); m1
m2 = matrix(-1:4, nrow=2, byrow=TRUE); m2
m3 = cbind(m1,m2); m3
(m4 = cbind(m1,m2))
```

```
# Create a data frame
```

```
e <- c(1,2,3,4)
f <- c("red", "white", "black", NA)
g <- c(TRUE,TRUE,TRUE,FALSE)
mydata <- data.frame(e,f,g)
names(mydata) <- c("ID", "Color", "Passed") # name variable
mydata
```

```
# Output
```

```
write.csv(mydata, file='mydata.csv', row.names=F)
```

```
# Import
```

```
(simple = read.csv('mydata.csv', header=TRUE, stringsAsFactors=TRUE))
class(simple)
class(simple[[1]])
class(simple[[2]])
class(simple[[3]])
(simple = read.csv('mydata.csv', header=FALSE, stringsAsFactors=FALSE))
class(simple[[3]])
```

```
# EXERCISE
```

```
# Create a matrix with 2 rows and 6 columns such that it contains the numbers 1,4,7,...,34.
```

```
# Make sure the numbers are increasing row-wise; ie, 4 should be in the second column.  
# Use the seq() function to generate the numbers. Do NOT type them out by hand!
```

```
# ANSWER
```

```
matrix(seq(from=1, to=34, by=3), nrow=2)
```

- 
- Elementary arithmetic operators
    - +, -, \*, /, ^
    - log, exp, sin, cos, tan, sqrt
    - FALSE and TRUE becoming 0 and 1, respectively
    - sum(), mean(), median(), min(), max(), var(), sd(), summary()
  - Matrix calculation
    - element-wise multiplication: A \* B
    - matrix multiplication: A %\*% B
    - singular value decomposition: eigen(A)
  - Loops: for() and while()

- 
- Probabilities
    - normal distribution: dnorm(), pnorm(), qnorm(), rnorm()
    - uniform distribution: dunif(), punif(), qunif(), runif()
    - multivariate normal distribution: dmvnorm(), rmvnorm()

```
# Generate two datasets
```

```
set.seed(100)  
x = rnorm(250, mean=0, sd=1)  
y = runif(250, -3, 3)
```

- 
- Basic plots
    - strip chart, histogram, box plot, scatter plot
    - Package ggplot2 (RECOMMENDED)

```
# Strip chart
```

```
stripchart(x)
```

```
# Histogram
```

```
hist(x)
```

```
# Box plot
```

```
boxplot(x)
```

```
# Side-by-side box plot
```

```
xy = data.frame(normal=x, uniform=y)
```

```
boxplot(xy)
```

```
# Scatter Plot with fitted line
```

```
plot(x, y, xlab="x", ylab = "y", main = "scatter plot between x and y")
```

```
abline(lm(y~x))
```

```

# EXERCISE
# Play with a data set called "Gasoline" included in the package "nlme".
# 1. How many variables are contained in this data set? What are they?
# 2. Generate a histogram of yield and calculate the five number summary for it.
#   What is the shape of the histogram?
# 3. Generate side-by-side boxplots,
#   comparing the temperature at which all the gasoline is vaporized (endpoint) to sample.
#   Does it seem that the temperatures at which all the gasoline is vaporized differ by sample?
# 4. Generate a plot that illustrates the relationship between yield and endpoint.
#   Describe the relationship between these two variables.
# 5. What if the plot created in Q4 were separated by sample?
#   Generate a plot of yield v.s. endpoint, separated by sample.

# ANSWER
attach(nlme::Gasoline)
# 1. Six variables: yield, endpoint, sample, API, vapor, ASTM
# 2.
summary(yield)
hist(yield, nclass=50)
# 3.
boxplot(endpoint ~ Sample)
anova(lm(endpoint ~ Sample))
# 4.
plot(x=endpoint, y=yield, xlab="endpoint", ylab = "yield",
     main = "scatter plot between endpoint and yield")
abline(lm(yield~endpoint))
# 5.
par(mfrow=c(2,5))
for (i in 1:10){
  plot(x=endpoint[Sample==i], y=yield[Sample==i], xlab='', ylab='', main=paste('Sample=', i))
  abline(lm(yield[Sample==i]~endpoint[Sample==i]))
}
# Do not forget to detach the dataset after using it.
detach(nlme::Gasoline)

```

## Matrix basics

### Matrix decomposition

- Eigen-decomposition (for square matrix  $\mathbf{A}_{n \times n}$ ):  $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$ 
  - $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$ 
    - $\lambda_1 \geq \dots \geq \lambda_n$  are the eigenvalues of  $\mathbf{A}$ , i.e.,  $n$  roots of characteristic equation  $\det(\lambda \mathbf{I}_n - \mathbf{A}) = 0$
  - $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_n]_{n \times n}$ 
    - $\mathbf{v}_1, \dots, \mathbf{v}_n$  are (right) eigenvectors of  $\mathbf{A}$ , i.e.,  $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
  - Implementation in *R*: `eigen()`

- 
- Spectral decomposition (for symmetric  $\mathbf{A}$ ):  $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ 
    - $\mathbf{V}$  is orthogonal, i.e.,  $\mathbf{V}^T = \mathbf{V}^{-1}$

- 
- Singular value decomposition (SVD) for  $n \times p$  matrix  $\mathbf{B}$ :  $\mathbf{B} = \mathbf{U}\mathbf{S}\mathbf{W}^T$ 
    - $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n]_{n \times n}$  with  $\mathbf{u}_i$  the  $i$ th eigenvector of  $\mathbf{B}\mathbf{B}^T$

- \*  $\mathbf{U}$  is orthogonal
- $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_p]_{p \times p}$  with  $\mathbf{w}_i$  the  $i$ th eigenvector of  $\mathbf{B}^\top \mathbf{B}$
- \*  $\mathbf{W}$  is orthogonal

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{0}_{n \times (p-n)} \end{bmatrix}_{n \times p} \text{ if } n \leq p \text{ AND } \begin{bmatrix} \mathbf{S}_1 & \dots & \mathbf{0}_{(n-p) \times p} \end{bmatrix}_{n \times p} \text{ if } n > p$$

- \*  $\mathbf{S}_1 = \text{diag}(s_1, \dots, s_n)$  if  $n \leq p$  and  $\text{diag}(s_1, \dots, s_p)$  if  $n > p$
- \*  $s_1 \geq \dots \geq s_n$  are square roots of eigenvalues of  $\mathbf{B}\mathbf{B}^\top$
- \*  $s_1 \geq \dots \geq s_p$  are square roots of eigenvalues of  $\mathbf{B}^\top \mathbf{B}$
- Thin/compact SVD for  $n \times p$  matrix  $\mathbf{B}$ :

$$\mathbf{B} = [\mathbf{u}_1, \dots, \mathbf{u}_r] \text{diag}(s_1, \dots, s_r) [\mathbf{w}_1, \dots, \mathbf{w}_r]^\top = s_1 \mathbf{u}_1 \mathbf{w}_1^\top + \dots + s_r \mathbf{u}_r \mathbf{w}_r^\top$$

- \*  $r = \text{rank}(\mathbf{B}) \leq \min\{n, p\}$
- \*  $s_1 \geq \dots \geq s_r > 0$  are square roots of non-zero eigenvalues of  $\mathbf{B}^\top \mathbf{B}$  or  $\mathbf{B}\mathbf{B}^\top$
- \* Implementation via  $R$ : `svd()`

- The connection of decompositions

$$\begin{array}{ccc} \text{Spectral decomposition} & \Leftrightarrow & \text{Eigen-decomposition} \\ \text{(for symmetric matrices)} & & \text{(for square matrices)} \\ \uparrow & & \\ \text{(Thin) SVD} & & \\ \text{(for any matrices)} & & \end{array}$$

```
options(digits = 4) # control the number of significant digits
set.seed(1)
# Generate a symmetric matrix
A = matrix(runif(12), nrow = 2, ncol = 6)
B = t(A) %*% A # guaranteed to be symmetric
isSymmetric(B) # check symmetry
# Eigen-decomposition
(res_eigen = eigen(B))
res_eigen$vectors %*% diag(res_eigen$values) %*% t(res_eigen$vectors) - B # diff between B and decompos
# SVD
(res_svd = svd(B))
res_svd$u %*% diag(res_svd$d) %*% t(res_svd$v) - B # diff between B and decomposed B
# Thin SVD
r = qr(B)$rank # rank
res_svd$u[,1:r] %*% diag(res_svd$d[1:r]) %*% t(res_svd$v[,1:r]) - B # diff between B and decomposed B
# Comparing eigen-decomposition and SVD
res_eigen$values - res_svd$d
res_eigen$vectors - res_svd$u
res_eigen$vectors - res_svd$v
```

## Square root and inverse of positive (semi-)definite matrix

- $\mathbf{A}$  is positive semi-definite (say  $\mathbf{A} \geq 0$ ) iff  $\mathbf{A}$  is symmetric and its eigenvalues are all non-negative
  - Equiv.,  $\mathbf{u}^\top \mathbf{A} \mathbf{u} \geq 0$  for any non-zero real  $n$ -vector  $\mathbf{u}$  (i.e.,  $n \times 1$  real matrix, say  $\mathbf{u} \in \mathbb{R}^{n \times 1}$  OR  $\mathbf{u} \in \mathbb{R}^n$ )
- If  $\mathbf{A} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$  is the spectral decomposition of positive semi-definite  $\mathbf{A}$ , then  $\mathbf{A}^{1/2} = \mathbf{V} \mathbf{\Lambda}^{1/2} \mathbf{V}^\top$ , where

- $\Lambda^{1/2} = \text{diag}(\lambda_1^{1/2}, \dots, \lambda_n^{1/2})$
- $\mathbf{A}^{1/2} \mathbf{A}^{1/2} = \mathbf{A}$
- $\mathbf{A}$  is positive definite (say  $\mathbf{A} > 0$ ) iff  $\mathbf{A}$  is symmetric and its eigenvalues are all positive
  - Equiv.,  $\mathbf{u}^\top \mathbf{A} \mathbf{u} > 0$  for all non-zero  $\mathbf{u} \in \mathbb{R}^n$
- If  $\mathbf{A} = \mathbf{V} \Lambda \mathbf{V}^\top$  is the spectral decomposition of positive definite  $\mathbf{A}$ , then
  - $\mathbf{A}^{-1} = \mathbf{V} \Lambda^{-1} \mathbf{V}^\top$ , where  $\Lambda^{-1} = \text{diag}(\lambda_1^{-1}, \dots, \lambda_n^{-1})$
  - $\mathbf{A}^{-1/2} = \mathbf{V} \Lambda^{-1/2} \mathbf{V}^\top$  is the inverse of  $\mathbf{A}^{1/2}$  and also the root of  $\mathbf{A}^{-1}$ , where  $\Lambda^{-1/2} = \text{diag}(\lambda_1^{-1/2}, \dots, \lambda_n^{-1/2})$

---

```
options(digits = 4) # control the number of significant digits
set.seed(1)
## Generate a demo of positive semi-definite matrices
A = matrix(runif(12), nrow = 2, ncol = 6)
B = t(A) %*% A # guaranteed to be positive semi-definite
# Get the root of B via the eigen-decomposition of B
res_eigen_B = eigen(B)
B_root1 = res_eigen_B$vectors %*%
  diag((res_eigen_B$values*(res_eigen_B$values>1e-6))^.5) %*%
  t(res_eigen_B$vectors)
# Get the root of B via an existing function
B_root2 = expm::sqrtm(B)
# Comparing
B_root1 - B_root2

## Generate a demo of positive definite matrices
C = A %*% t(A) # (almost surely) guaranteed to be positive definite
# Get the inverse of C via the eigen-decomposition of B
res_eigen_C = eigen(C)
C_inv1 = res_eigen_C$vectors %*%
  diag(res_eigen_C$values^-1) %*%
  t(res_eigen_C$vectors)
# Get the inverse of C via an existing function
C_inv2 = solve(C)
# Comparing
C_inv1 - C_inv2
```

---

## Determinant and trace

- Merely applicable to square matrices
- Properties for determinant
  - $\det(\mathbf{A}) = \prod_i \lambda_i$
  - $\det(\mathbf{A}^\top) = \det(\mathbf{A})$
  - $\det(\mathbf{A}^{-1}) = 1/\det(\mathbf{A})$
  - $\det(c \cdot \mathbf{A}) = c^n \det(\mathbf{A})$  for  $n \times n$  matrix  $\mathbf{A}$  and scalar  $c$
  - $\det(\mathbf{A}\mathbf{B}) = \det(\mathbf{A})\det(\mathbf{B})$  if  $\mathbf{A}$  and  $\mathbf{B}$  are square matrices of the identical dimension
- Properties for trace
  - $\text{tr}(\mathbf{A}) = \sum_i \lambda_i$
  - $\text{tr}(c \cdot \mathbf{A}) = c \cdot \text{tr}(\mathbf{A})$  for scalar  $c$
  - $\text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B})$  if  $\mathbf{A}$  and  $\mathbf{B}$  are square matrices of the identical dimension



- (Trace trick)  $\text{tr}(\mathbf{A}_1 \cdots \mathbf{A}_k) = \text{tr}(\mathbf{A}_{k'+1} \cdots \mathbf{A}_k \mathbf{A}_1 \cdots \mathbf{A}_{k'})$  for  $1 < k' < k$ .
    - \* Specifically,  $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$
  - Remark:  $\det(\mathbf{A})$  and  $\text{tr}(\mathbf{A})$  can be taken as measures of the size of  $\mathbf{A}$  when  $\mathbf{A} > 0$
-