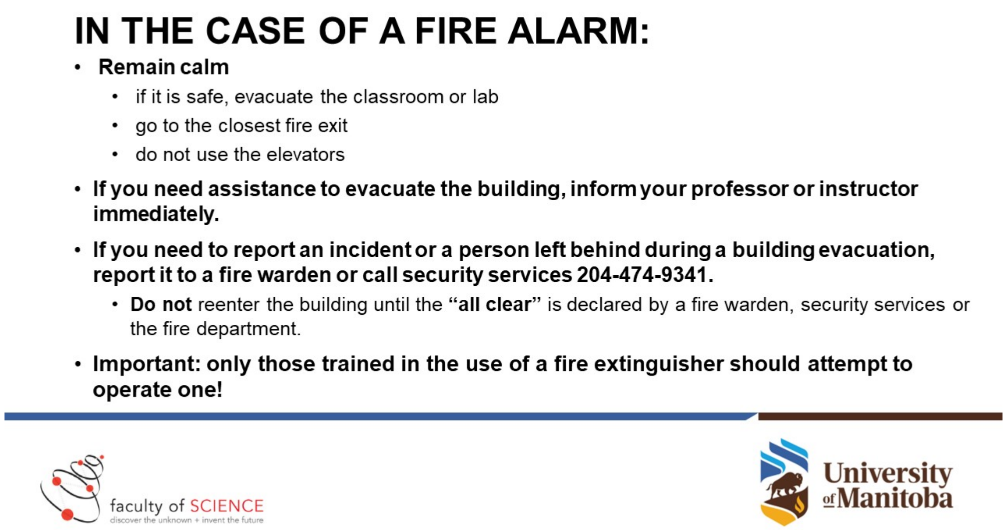
STAT 3690 Lecture Note

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

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

2023/Jan/11 10:38:20



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 10:30–11:20 am
- Assessments
  - Assignments (4/5 times)
  - 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 analying 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*
    - paenthesis () to enclose inputs for functions
    - square brackets [], [[]] for indexing
    - braces {} to enclose forloop or statements such as if or ifelse

---

```r
# 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]
```

```r
# 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))
```

```r
# 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
```

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

```r
# 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]])
```

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

```r
# 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!
```

```r
# 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
  - singlar 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()

```r
# 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)

```r
# Strip chart
stripchart(x)

# Histogram
hist(x)

# Box plot
boxplot(x)

# Side-bu-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))
```

```r
# 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.
```

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

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

- Singular value decomposition (SVD) for $n \times p$ matrix $\mathbf{B}$: $\mathbf{B} = \mathbf{USW}^\top$
  - $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]_{n\times n}$ with $\boldsymbol{u}_i$ the ith eigenvector of $\mathbf{BB}^\top$

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

$$\mathbf{S} = \left[\begin{array}{c:c} \mathbf{S}_1 & \mathbf{0}_{n\times(p-n)} \end{array}\right]_{n\times p} \text{ if } n \le p \text{ AND } \left[\begin{array}{c} \mathbf{S}_1 \\ \hdashline \mathbf{0}_{(n-p)\times p} \end{array}\right]_{n\times p} \text{ if } n > p$$

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

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

    * $r = \text{rank}(\mathbf{B}) \le \min\{n, p\}$
    * $s_1 \ge \cdots \ge s_r > 0$ 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

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

```
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 of positive (semi-)definite matrix

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

- $\Lambda^{1/2} = \text{diag}(\lambda_1^{1/2}, \ldots, \lambda_n^{1/2})$
- $\mathbf{A}^{1/2}\mathbf{A}^{1/2} = \mathbf{A}$

- If $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^\top$ is the spectral decomposition of positive definite $\mathbf{A}$, then $\mathbf{A}^{-1/2} = \mathbf{V}\Lambda^{-1/2}\mathbf{V}^\top$ satisfies that
  - $\Lambda^{-1/2} = \text{diag}(\lambda_1^{-1/2}, \ldots, \lambda_n^{-1/2})$
  - $\mathbf{A}^{-1/2}\mathbf{A}^{-1/2} = \mathbf{A}^{-1}$
  - $\mathbf{A}^{1/2}\mathbf{A}^{-1/2} = \mathbf{I}_n$

Determinant and trace

- Applicable only 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
  - $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$ 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
  - $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$ for $m \times n$ matrix $\mathbf{A}$ and $n \times m$ matrix $\mathbf{B}$
- Remark:$\det(\mathbf{A})$ and $\text{tr}(\mathbf{A})$ can be taken as measures of the size of $\mathbf{A}$ when $\mathbf{A}$ is positive definite (i.e., its eigenvalues are all positive).

- Exercise: Prove that
  1. $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$ for $m \times n$ $\mathbf{A}$ and $n \times m$ $\mathbf{B}$.
  2. (The 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 $1 < k' < k$.
  3. $\text{tr}(\mathbf{A}) = \sum_i \lambda_i$.
  4. $\det(\mathbf{A}) = \prod_i \lambda_i$. Hint: Jordan matrix decomposition, i.e., there exists a Jordan normal (or canonical) form $\mathbf{J}$ and invertible $\mathbf{U}$ such that $\mathbf{A} = \mathbf{UJU}^{-1}$ for any square $\mathbf{A}$.

Block/partitioned matrix

- A partition of matrix: Suppose $\mathbf{A}_{11}$ is of $p \times r$, $\mathbf{A}_{12}$ is of $p \times s$, $\mathbf{A}_{21}$ is of $q \times r$ and $\mathbf{A}_{22}$ is of $q \times s$. Make a new $(p+q) \times (r+s)$-matrix by organizing $\mathbf{A}_{ij}$'s in a 2 by 2 way:

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array}\right]$$

  e.g.,

$$\mathbf{A} = \left[\begin{array}{cc|c} 1 & 0 & 2 \\ 0 & 1 & 3 \\ \hline 4 & 5 & 6 \end{array}\right]$$

  if

$$\mathbf{A}_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{A}_{12} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{A}_{21} = \begin{bmatrix} 4 & 5 \end{bmatrix}, \quad \text{and} \quad \mathbf{A}_{22} = \begin{bmatrix} 6 \end{bmatrix}.$$

- Operations with block matrices
  - Working with partitioned matrices just like ordinary matrices

– Matrix addition: if dimensions of $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ are quite the same, then

$$\mathbf{A}+\mathbf{B}=\left[\begin{array}{cc}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{array}\right]+\left[\begin{array}{cc}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{array}\right]=\left[\begin{array}{cc}\mathbf{A}_{11}+\mathbf{B}_{11} & \mathbf{A}_{12}+\mathbf{B}_{12}\\ \mathbf{A}_{21}+\mathbf{B}_{21} & \mathbf{A}_{22}+\mathbf{B}_{22}\end{array}\right]$$

– Matrix multiplication: if $\mathbf{A}_{ij}\mathbf{B}_{jk}$ makes sense for each i, j, k, then

$$\mathbf{AB}=\left[\begin{array}{cc}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{array}\right]\left[\begin{array}{cc}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{array}\right]=\left[\begin{array}{cc}\mathbf{A}_{11}\mathbf{B}_{11}+\mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12}+\mathbf{A}_{12}\mathbf{B}_{22}\\ \mathbf{A}_{21}\mathbf{B}_{11}+\mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12}+\mathbf{A}_{22}\mathbf{B}_{22}\end{array}\right]$$

– Inverse: if $\mathbf{A}$, $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are all invertible, then

$$\mathbf{A}^{-1}=\left[\begin{array}{cc}\mathbf{A}_{11.2}^{-1} & -\mathbf{A}_{11.2}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{A}_{11.2}^{-1} & \mathbf{A}_{22.1}^{-1}\end{array}\right]$$

  * $\mathbf{A}_{11.2}=\mathbf{A}_{11}-\mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$
  * $\mathbf{A}_{22.1}=\mathbf{A}_{22}-\mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$
