STAT 3690 Lecture 01

zhiyanggeezhou.github.io

Zhiyang Zhou (zhiyang.zhou@umanitoba.ca)

Jan 24th, 2022

Contact

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

Timeline

- Lectures
  - Mon/Wed/Fri 9:30–10:20 via Zoom (tentatively)
- Office Hour
  - (Instructor) Wed/Fri 10:20–11:20 via Zoom (tentatively)
  - (Marker) TBA
- Exam
  - Midterm: (tentatively) Mar. 7, 2022
  - Final project: TBD

Grading

- Assignments (20%)
  - Scanned/photographed and submitted to Crowdmark
  - Attaching both outputs and sourse codes if R is used in computation
  - Including necessary interpretation
  - Organized in a clear and readable way
  - Accepting NO late submission
- Midterm (30%)
  - Take-home
  - Open-book
  - Time-sensitive
- Final project (50%)
  - Individual report with an analysis of recent dataset(s)
  - To be detailed later

Meterials

- Reading list (recommended but not required)
  - R. A. Johnson & D. W. Wichern. (2007). *Applied Multivariate Statistical Analysis*, 5/6th Ed. London: Pearson Education.
    * Textbook, abbr. J&W

        * 2HR print reserve in the Sciences and Technology Library
    - A. C. Rencher & W. F. Christensen. (2012). *Methods of Multivariate Analysis*, 3rd Ed. Hoboken: Wiley.
        * Electronically accessible via 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
    - 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 <- "stat3690Lec01"
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 cheep
    - 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 encoutered 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]
```

```
# 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
  - 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()

```
# 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-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))
```

```
# 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")
```

```r
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 properties

- Determinant and trace
  - Applicable only to square matrices
  - Properties for determinant
    * $|\mathbf{A}^\top| = |\mathbf{A}|$
    * $|\mathbf{A}^{-1}| = |\mathbf{A}|^{-1}$
    * $|c\mathbf{A}| = c^n|\mathbf{A}|$ for $n \times n$ matrix $\mathbf{A}$ and scalar c
    * $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$ if $\mathbf{A}$ and $\mathbf{B}$ are square matrices of the identical dimension
    * $|\mathbf{A}| = \prod_i \lambda_i$
  - Properties for trace
    * $\mathrm{tr}(c\mathbf{A}) = c\mathrm{tr}(\mathbf{A})$ for scalar c
    * $\mathrm{tr}(\mathbf{A}+\mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B})$ if $\mathbf{A}$ and $\mathbf{B}$ are square matrices of the identical dimension
    * $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ for $m \times n$ $\mathbf{A}$ and $n \times m$ $\mathbf{B}$
    * $(\mathrm{tr}(\mathbf{AA}^\top))^{1/2} = (\sum_{i,j} a_{ij}^2)^{1/2}$ Frobenius norm (a generilization of Euclidean norm)
    * $\mathrm{tr}(\mathbf{A}) = \sum_i \lambda_i$

- Exercise: Prove that
  1. $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ for $m \times n$ $\mathbf{A}$ and $n \times m$ $\mathbf{B}$.
  2. $\mathrm{tr}(\mathbf{A}_1 \cdots \mathbf{A}_k) = \mathrm{tr}(\mathbf{A}_{k'+1} \cdots \mathbf{A}_k \mathbf{A}_1 \cdots \mathbf{A}_{k'})$ for $1 < k' < k$.
  3. $\mathrm{tr}(\mathbf{A}) = \sum_i \lambda_i$.
  4. $|\mathbf{A}| = \prod_i \lambda_i$.
- Hint: Jordan matrix decomposition: there exists a Jordan normal (or canonical) form $\mathbf{J}$ and invertible $\mathbf{U}$ such that $\mathbf{A} = \mathbf{UJU}^{-1}$ for any square $\mathbf{A}$.
- Remark:$|\mathbf{A}|$ and $\mathrm{tr}(\mathbf{A})$ can be taken as measures of the size of $\mathbf{A}$ when $\mathbf{A}$ is positive definite.

- Proof:
  1. $\mathrm{tr}(\mathbf{AB}) = \sum_i \sum_j a_{ij}b_{ji} = \sum_j \sum_i b_{ji}a_{ij} = \mathrm{tr}(\mathbf{BA})$.
  2. Take $\mathbf{A}_1 \cdots \mathbf{A}_{k'}$ and $\mathbf{A}_{k'+1} \cdots \mathbf{A}_k$ as a whole, respectively.
  3. $\mathrm{tr}(\mathbf{UJU}^{-1}) = \mathrm{tr}(\mathbf{JU}^{-1}\mathbf{U}) = \mathrm{tr}(\mathbf{J}) = \sum_i \lambda_i$.
  4. $|\mathbf{A}| = |\mathbf{UJU}^{-1}| = |\mathbf{U}||\mathbf{J}||\mathbf{U}^{-1}| = |\mathbf{J}|$.

- Singular value decomposition (SVD)
  - SVD: $\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^\top = \mathbf{U}\Lambda\mathbf{V}^{-1}$
    * any $m \times n$ (real) matrix $\mathbf{A}$
    * $m \times m$ matrix $\mathbf{U}$ and $n \times n$ matrix $\mathbf{V}$, both orthogonal
    * $m \times n$ Λ with λ_i being the (i,i)-entry and zero elsewhere
      · λ_i are eigenvalues of $\mathbf{A}$
      · $|\lambda_1| \geq \cdots \geq |\lambda_{\min\{m,n\}}| \geq 0$
  - Thin SVD: $\mathbf{A} = \mathbf{U}\Lambda\mathbf{V}^\top$

- any $m \times n$ (real) matrix $\mathbf{A}$
- $m \times r$ matrix $\mathbf{U}$ and $r \times n$ matrix $\mathbf{V}$, both semi-orthogonal, i.e., $\mathbf{U}^\top \mathbf{U} = \mathbf{V}^\top \mathbf{V} = \mathbf{I}_r$
- $r \times r$ $\Lambda = \text{diag}\{\lambda_1, \dots, \lambda_r\}$
  - $r = \text{rk}(\mathbf{A})$
  - $|\lambda_1| \geq \cdots \geq |\lambda_r| > 0$
- Implementation in *R*: svd()

- Spectral decomposition (eigendecomposition)
  - Special case of SVD specific for symmetric $\mathbf{A}$
    - $\mathbf{U} = \mathbf{V}$
  - Special interest in
    - Positive definite: symmetric $\mathbf{A}$ with $\lambda_i > 0$ for all i
    - Semi-positive (or non-negative) definite: symmetric $\mathbf{A}$ with $\lambda_i \geq 0$ for all i
  - Further results
    - If eigenvalues λ_i are all nonzero, then
      - $\mathbf{A}^{-1} = \mathbf{U}\Lambda^{-1}\mathbf{U}^\top$.
    - If $\mathbf{A}$ is semi-positive, then
      - $\mathbf{A}^{1/2} = \mathbf{U}\Lambda^{1/2}\mathbf{U}^\top$.
    - If $\mathbf{A}$ is positive definite, then
      - $\mathbf{A}^{-1/2} = \mathbf{U}\Lambda^{-1/2}\mathbf{U}^\top$.
  - Implementation in *R*: eigen()

- Exercise: Is it feasible to apply eigen() only in conducting the thin SVD for a matrix with non-negative singular values (λ_i's)?

```r
options(digits = 4) # control the number of significant digits
set.seed(1)
A = matrix(runif(12), nrow = 2, ncol = 6)
svdResult = svd(A)
eigenResult = eigen(tcrossprod(A))
# respective set of eigenvalues from each method
svdResult$d; eigenResult$values^.5
# respective eigenvectors from each method
svdResult$u; eigenResult$vectors
# respective eigenvectors from each method
svdResult$v; t(diag(eigenResult$values^-.5) %*% t(eigenResult$vectors) %*% A)
```