

PH 716 Applied Survival Analysis

Part I: R basics

Zhiyang Zhou (zhou67@uwm.edu, zhiyanggeezhou.github.io)

2024/Jan/21 09:16:18

Syllabus

Contact

- Instructor: Zhiyang (Gee) Zhou, PhD, Asst. Prof. (Biostatistics)
 - Email: zhou67@uwm.edu
 - Homepage: zhiyanggeezhou.github.io

Timeline

- Lectures
 - Mon/Wed 12:45–14:00
- Office Hour
 - TBD
- Assessments
 - 4 or 5 Assignments
 - Midterm
 - Final project

Grading

- Assignments (30%)
 - Digital copies submitted
 - 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 NOT later than Mar. 13, 2024
- Final project (35%)
 - Individual report analyzing recently collected data
 - See the guideline posted at Canvas

Materials

- Reading list (recommended but not required)
 - [DM] D. F. Moore. (2016). *Applied Survival Analysis Using R*. Switzerland: Springer.

- * Accessible via UWM library <http://ebookcentral.proquest.com/lib/uwm/detail.action?docID=4526865>
- [KM] J. P. Klein & M. L. Moeschberger. (2003). *Survival analysis : techniques for censored and truncated data*, 2nd Ed. New York: Springer.
- 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](https://github.com/zhiyanggeezhou)
 - Canvas

Outline

- Topics to be covered
 - R basics
 - Basic quantities of survival models
 - Kaplan-Meier and Nelson-Altschuler(-Aalen-Fleming-Harrington) estimators
 - Comparisons of several multivariate means
 - Accelerated failure time model
 - Principal component analysis
 - Cox proportional hazards (CPH) model
 - CPH model with time dependent covariates
 - Model selection and interpretation
 - Model diagnostics
 - Competing risks
 - 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 the fusion of *R* and statistical methods
 - G. James, D. Witten, T. Hastie and R. Tibshirani (2023) *An Introduction to Statistical Learning: with Applications in R*, 2nd Ed.
 - M. L. Rizzo (2019) *Statistical Computing with R*, 2nd Ed.
 - 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 graphics
 - 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)

```