Computing in Epidemiology and Biostatistics

/* This is a non-synchronous online course taught in English. ***/**

Instructors: Prof. Wan-Yu Lin, Prof. Wen-Chung Lee

TAs: Xue-Yong Chang (r10849044@ntu.edu.tw), Ying-Hsuan Hsieh (r09849034@ntu.edu.tw)

Course: Video will be uploaded to NTU COOL every **Thursday afternoon**.

Online office Hour: Should you have any question, you may register your NTU mail on Microsoft Teams. Here is how: https://www.cc.ntu.edu.tw/chinese/epaper/0053/20200620_5308.html
And then please mail your registered NTU mail to our TA (r10849044@ntu.edu.tw).
The online office hour will be held every **Monday 2 pm – 3 pm**. Welcome to join the online discussions and get advice from TA (starting from Sep. 12).

[Students' Background] Prerequisite course: at least one biostatistics (or statistics) or epidemiology course. You should have learned ANOVA, regression, etc. **These will not be taught again in this course. You may forget some details but you need to review them by yourselves.**

[Course introduction]

In most Biostatistics courses, lecturers usually introduce theoretical models, and then use statistical software (such as SAS and R) to analyze the data. However, there is a black box between these two parts. To fill in this gap, we will teach the numerical computation process involved in statistical models. Students will learn matrix operations, numerical analyses, Monte-Carlo simulations, etc. We will teach how to construct a log-likelihood function according to a statistical distribution, how to obtain maximum likelihood estimates for a logistic regression and a Poisson regression, how to build exact confidence intervals, and how to design Monte-Carlo simulations for a research topic, etc.

The main purpose of this course is to build logical thinking, and to train students to write their own functions, rather than data analysis.

[References]

1. Introduction to scientific programming and simulation using R (second edition). Owen Jones, Robert Maillardet, and Andrew Robinson (2014). Chapman & Hall/CRC.

2. Principles of Biostatistics. 2nd edition. Marcello Pagano and Kimberlee Gauvreau (2000). Duxbury Press.

[How we evaluate students?]

Homework (100%): Weekly assignments and homework, in the end of this semester, semester grade of each student will be adjusted according to the grade distribution of the whole class.

1. Strict due time: If we assign the first homework today, please submit it to **NTU COOL** before next Thursday 12:00 noon.
2. No late homework can be accepted. All late homework will be scored as a zero.
3. Plagiarism in Homework will be scored as a zero, including "you use someone's answers" or "someone uses your answers". Therefore, please don't mail your homework answers to anyone else.

Install and start R

林菀俞 (Wan-Yu Lin)

Originators of the R programming language: Prof. ***R**oss Ihaka* and Prof. ***R**obert Gentleman*

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

R is a language and environment for statistical computing and graphics. It is similar to the S language and environment. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R website: http://www.r-project.org/
R editor: File > New script; File > Open script
Professional R editors: Tinn-R (http://sourceforge.net/projects/tinn-r/),
RStudio (http://www.rstudio.com/)

Basic commands in R:

```
> getwd()                          (get working directory)
[1] "C:/Users/user/Documents"
> setwd("D:/wanyu")      (set working directory)
```

```
> help(mean)
> ?mean
```

Help > Manuals (in PDF)

Arithmetic in R:

R uses the usual symbols for addition +, subtraction -, multiplication *, division /, and exponentiation ^. Parentheses ( ) can be used to specify the order of operations. R also provides %% for taking the modulus and %/% for integer division.

```
> (1 + 1/100)^100     or       > (1 + 1/100)**100
[1] 2.704814
```

$$\exp(1) = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

```
> 17%%5       # remainder
[1] 2
> 17%/%5      # quotient
[1] 3
```

```
> exp(1)
[1] 2.718282

> options(digits = 16)
> exp(1)
[1] 2.718281828459045
> pi
[1] 3.141592653589793
> options(digits = 7)        # default setting
```

The functions floor(x) and ceiling(x) round down and up respectively, to the nearest integer.

```
> floor(pi)          # find the nearest integer < pi
[1] 3
> ceiling(pi)         # find the nearest integer > pi
[1] 4
> round(pi)          # rounding pi
[1] 3
```

Variable in R

To assign a value to a variable we use the assignment command <-. Variables are created the first time you assign a value to them. You can give a variable any name made up of letters, numbers, and . or _, provided it starts with a letter, or . then a letter. Note that names are case sensitive. (AA and aa are two different objects in R)

```
> x <- 100
> x
    [1] 100
> (1 + 1/x)^x
    [1] 2.704814
> x <- 200
> (1 + 1/x)^x
    [1] 2.711517
> (y <- (1 + 1/x)^x)
    [1] 2.711517
> n <- 1
> n <- n + 1
> n
    [1] 2
```

We understand the assignment n <- n + 1 by thinking of n as the name of a data location in the computer memory, whose contents change as the assignment is processed. Contrast this with the usual mathematical interpretation of n = n +1, where the variable n is thought of as having the same value on both sides (so this equation has no finite solution).

Function in R

```
> seq(from = 1, to = 9, by = 2)
[1] 1 3 5 7 9
> demo(graphics)
```

Vector in R

```
> x <- seq(from = 1, to = 9, by = 2)
> length(x)
[1] 5
```

The function length(x) gives the number of elements of x.

Algebraic operations on vectors act on each element separately, that is elementwise.

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)
> x * y
[1] 4 10 18
> x + y
[1] 5 7 9
> y^x
[1] 4 25 216
```

When you apply an algebraic expression to two vectors of unequal length, R automatically repeats the shorter vector until it has something the same length as the longer vector.

```
> c(1, 2, 3, 4) + c(1, 2)
[1] 2 4 4 6
```

```
> c(1, 2, 3, 4) + c(1, 2, 3)
[1] 2 4 6 5
Warning message:
In c(1, 2, 3, 4) + c(1, 2, 3) :
  longer object length is not a multiple of shorter object length
```

Ex 0: Plot the probability density function (p.d.f.) of the standard normal distribution

```
x <- seq(-3, 3, 0.01)
y <- dnorm(x, 0, 1)
plot(x, y)
```

Ex 1: Plot the probability density function (p.d.f.) and cumulative distribution function (c.d.f.) of the chi-square distribution with degrees of freedom 1

Use chi-square distribution as an example	
dchisq(x, df)	probability density function, p.d.f.
pchisq(q, df)	cumulative distribution function, c.d.f.
qchisq(p, df)	quantile function
rchisq(n, df)	generate random numbers

These functions provide information about the chi-square distribution with **df** degrees of freedom. **dchisq** gives the density, **pchisq** gives the distribution function, **qchisq** gives the quantile function, and **rchisq** generates random variables.

Ex 2: In regression analysis, if you observe an *F* test statistic = 3.2 (right-tailed test), and under the null hypothesis this statistic should follow an *F* distribution with a numerator d.f. of 3 and a denominator d.f. of 194, please find the p-value? Please plot the pdf and cdf of this *F* distribution.

Ex 3: In regression analysis, if you observe a *t* test statistic = -2.08 (two-tailed test), and under the null hypothesis this statistic should follow a *t* distribution with a d.f. of 136, please find the p-value? What's the p-value if you observe a *t* test statistic = 2.45 (two-tailed test). Please plot the pdf and cdf of this *t* distribution.

Matrix in R:

```
> Y <- matrix(NA, 3, 2)

> matrix(c(1,2,3,4),2,2)
      [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
> matrix(c(1,2,3,4),2,2,byrow=T)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Ex 4: Please make a matrix as follows:

$$\begin{pmatrix} 0 & 2 & 3 \\ 0 & 5 & 0 \\ 7 & 0 & 0 \end{pmatrix}$$

Loop in R :

```
ss <- 0
for(i in 1:100){     #index number
   ss <- ss+i
}
ss

sum(1:100)

i=1, ss <- ss+i=0+1=1
i=2, ss <- ss+i=1+2=3
i=3, ss <- ss+i=3+3=6
i=4, ss <- ss+i=6+4=10
i=5, ss <- ss+i=10+5=15
```

Read files in R:

```
Seizure1 <- read.csv('seizure.csv')
Seizure2 <- read.table('Seizure.txt')
head(Seizure1)
head(Seizure2)
dim(Seizure1)
dim(Seizure2)
```

Ex 5: Please rearrange the data in Seizure1 as the format in Seizure2. (Please answer this question with “for” loop)