```
# Review of HW

# Exercise 2.####################################################################

# Download results.csv from HustkyCT (lecture11-data) in your computer.
# Import this dataset to R.

sc.game <- read.csv("results.csv", stringsAsFactors = FALSE, na.strings="")
head(sc.game)
str(sc.game)

sc.game$date <- as.Date(sc.game$date)
str(sc.game)

# Compare the average number of goals in 1930-1939 and in 2005-2014 and see
# whether it increased or decreased.

sc.game$year <- format(sc.game$date, format="%Y")
sc.game$score <- sc.game$home_score + sc.game$away_score

ave.score.1 <- colMeans(sc.game[sc.game$year>="1930" & sc.game$year<="1939",
                                c("home_score","away_score","score")])
ave.score.2 <- colMeans(sc.game[sc.game$year>="2005" & sc.game$year<="2014",
                                c("home_score","away_score","score")])

ave.score.1
ave.score.2

# What is the ratio of home winning games to the total number of games in 1950s?
game.1950s <- sc.game[sc.game$date >="1950-01-01" & sc.game$date <= "1959-12-31",]
home_win <- game.1950s$home_score > game.1950s$away_score
sum(home_win) / nrow(game.1950s)

# plot the number of games over each year from 1900 to 1960 and see whether
# there was decrease in game in world war I (1914-1919) and world war II (1939-1945)

games <- summary(factor(sc.game$year[sc.game$year>1900 & sc.game$year<1960]))
plot(games, type="s")


##############################################################################
# conditional statement: if and if-else

# Sometimes, we want to run one piece of code if some condition is true, but a
# different piece of code if it is false. We can do this using if-else satement.

# syntex of if statement
# if (condition) {
# statement
# }
#

if ( 3 != 4) {
  print ("3 is not equal to 4")
}

if ( 3 == 4) {
  print ("3 is equal to 4") # nothing executed because the condition doesn't hold.
}

if ("boy" < "girl") {
  print("Girl is taller than boy")
}
```

```
# syntex of if-else statement
# if (condition) {
# statement 1
# } else {
# statement 2
# }
#

# The else part is optional and is only evaluated if condition is false. It is
important to know that the
# else must be in the same line as the closing brace of the if statement

set.seed(4)
R <-
c("Zhifei","Daniel","Allison","Shenji","Utshaw","Zhenhao","Jonathan","Yuriy","Yuanming"
,

"Wnentao","Xiaohan","Huarui","Hanna","Jiachuan","Johan","Yifei","Zizhen","Xiaoying",
       "Claudia","John","Shilpa","Junfu","Yuan","Mofei","Qi","Kevin","Jiawen","Wenhui",
       "Kuanyue","Wei","Zichuan","Xin")
R.boy <-
c("Zhifei","Daniel","Utshaw","Zhenhao","Jonathan","Yuriy","Yuanming","Wnentao",

"Jiachuan","Johan","Yifei","John","Junfu","Mofei","Qi","Kevin","Zichuan","Xin")
R.girl <- R[!(R %in% R.boy)]

a <- sample(x=R, size=5, replace=FALSE)

if (sum(a %in% R.girl) > sum(a %in% R.boy)) {
  cat("Girls are more than boys in R class")
} else {
  cat("Boys are more than Girls in R class")
}

# We can have as many if-else statements as we want.
# Syntex

# if (condition 1) {
# statement 1
# } else if (condition 2) {
# statement 2
# } else if (condition 3) {
# statement 3
# } else {
# statement 4
# }

player1 <- "rock"
player2 <- "rock"

if ( (player1 == "rock"& player2 == "scissors") | (player1 == "scissors"& player2 ==
"paper")
     | (player1 == "paper"& player2 == "rock") ) {
  cat("Player1 win!")
} else if ((player1 == "scissors" & player2 == "rock") | (player1 == "paper"& player2
== "scissors")
           | (player1 == "rock"& player2 == "paper")) {
  cat("Player2 win!")
} else {
  cat("Tie!")
}

# While loop: Loops are used in programming to repeat a specific block of code.
# While loop repeat as long as the given condition is true.
# Draw a flowchart

# Syntex of while loop
# while (condition) {
# statement
# }

# Example: Initialize the speed variable
speed <- 64
# Code the while loop
while ( speed > 30 ) {
  cat("Slow down!")
  speed <- speed - 7
}

cat("Ok, now. The speed is", speed, "miles/hour")

# Newton-Raphson method for f(x) = x^3 + 2*x + 1
# The Newton-Raphson method is an approach for finding the roots of nonlinear
# equations and is one of the most common root-finding algorithms due to its
# relative simplicity and speed.

x <- seq(-2,2, by=0.01)
y <- x^3 + 2*x + 1
plot(x,y, type="l")

x <- c(0, 5)
i <- 2

while (abs(x[i] - x[i-1])>0.0001) {
  x[i+1] <- x[i] - (x[i]^3 + 2*x[i] + 1)/(3*x[i]^2 + 2)
  i <- i + 1
}
cat("The solution to the equation x^3 + 2*x = 0 is x=", x[i])

# for loop: repeat a block of code in the loop as many as specified.

# syntax

# for (var in sequence) {
# statement
# }

# Sequence is a vector. In each iteration, var takes on a value of sequence in order
# and statement is evaluated.

Econ5495 <-
c("Zhifei","Daniel","Allison","Shenji","Utshaw","Zhenhao","Jonathan","Yuriy","Yuanming"
,

"Wnentao","Xiaohan","Huarui","Hanna","Jiachuan","Johan","Yifei","Zizhen","Xiaoying",

"Claudia","John","Shilpa","Junfu","Yuan","Mofei","Qi","Kevin","Jiawen","Wenhui",
              "Kuanyue","Wei","Zichuan","Xin")

for (i in 1:length(Econ5495)) {
  print(paste("Hello", Econ5495[i]))
}

for (a in Econ5495) {
  print(paste("Hello",a))
}

# break statement:
```

```
# break is used inside a loop (for, while) to stop the iterations and get out of the
loop

x <- seq(-3,1, by=0.01)
y <- x^2 + 2*x
plot(x,y, type="l")
abline(h=0)

x <- c(5,rep(NA,999))

for (i in 1:1000) {
  x[i+1] <- x[i] - (x[i]^2 + 2*x[i])/(2*x[i] + 2)
  print(i)
  if (abs(x[i+1] - x[i])<0.0001) {
    break
  }
  i <- i + 1
}
print(paste("The solution to the equation x^3 + 2*x = 0 is x=", round(x[i],2)))

# Next statement

# A next statement is useful when we want to skip the current iteration of a loop
without
# terminating it. On encountering next, the R skips further evaluation and starts
# next iteration of the loop.

Econ5495 <-
c("Zhifei","Daniel","Allison","Shenji","Utshaw","Zhenhao","Jonathan","Yuriy","Yuanming"
,

"Wnentao","Xiaohan","Huarui","Hanna","Jiachuan","Johan","Yifei","Zizhen","Xiaoying",

"Claudia","John","Shilpa","Junfu","Yuan","Mofei","Qi","Kevin","Jiawen","Wenhui",
              "Kuanyue","Wei","Zichuan","Xin")
Econ5495.boy <-
c("Zhifei","Daniel","Utshaw","Zhenhao","Jonathan","Yuriy","Yuanming","Wnentao",

"Jiachuan","Johan","Yifei","John","Junfu","Mofei","Qi","Kevin","Zichuan","Xin")
Econ5495.girl <- NULL

for (i in 1:length(Econ5495)) {
  if (Econ5495[i] %in% Econ5495.boy) {
    next
  }
  Econ5495.girl <- c(Econ5495.girl, Econ5495[i])
}

Econ5495.girl

# Alternative way
Econ5495.girl <- Econ5495[!(Econ5495 %in% Econ5495.boy)]


# for loop in a matrix
# We can employ a for loop inside a for loop.

# syntax
# for (var1 in sequence 1) {
#   for (var2 in seqence 2) {
#     statement
#   }
# }
```

```
oil.shock <- rnorm(10, 0, 1)           # Ft
state.chracteristic <- runif(5, -1, 1)    # ai

Economy <- matrix(rep(NA, 50), 10, 5)
rownames(Economy) <- 2000:2009
colnames(Economy) <- state.abb[1:5]

for (i in 1:length(oil.shock)) {
  for (j in 1:length(state.chracteristic)) {
    Economy[i,j] <- as.numeric(oil.shock[i]*state.chracteristic[j] > 0)
  }
}

Economy

# Exercise 1: Alternative way to for loop: matrix operation
# Use the matrix operation to produce the same outcome above (Economy).


# Exercise 2 #################################################################

# Let's do simulation experiment about the omitted variable bias problem.
# Consider the following data generating process (DGP).

# attend <- runif(100,0,1)
# attitude <- 0.5 * attend + rnorm(100)
# error <- rnorm(100)
# score <- 0.5 * attend + attitude + error

# The true DGP indicates that the effect of attendance on the exam score is 0.5.
# Run the OLS and see whether the estimate is close to the true value.
# Using the for loop, replicate this simulation 500 times and take the mean of the
# estimate and compare the mean with the true value.


# Exercise 3 #################################################################
li <- 15
fb <- 9

# If both li and fb are 15 or higher, set sns (social network score) equal to double
# the sum of li and fb.
# If both li and fb are strictly below 10, set sns equal to half the sum of li and fb.
# In all other cases, set sns equal to li+fb.


# Exercise 4 Newton-Raphson method ############################################

# Using the Newton-Raphson method above, find the soluion of
# (x-1)^3 + 0.5*x^2 -x - 2= 0.
```