15 pts : Part I, general R commands

 # Create sequence

```
y<-seq(2,200,by=2)
```

```
# with y[i]=T if x[i] is divisible by 10, otherwise F
 y <- ifelse(x%%10==0, T, F) ###ifelse(test, yes, no)
```

```
> y = (x %% 10 == 0)
> y[10]
[1] TRUE
> y[11]
[1] FALSE
```

```
# Standard normal with increasing order
z <- sort(rnorm(100), decreasing=FALSE)
```

 # Create data frame

```
recipe= data.frame (amount=1:3,
                    unit=c("cup", "cups", "oz"),
                    ingredient = c("flour", "salt", "sugar"))
```

```
> recipe
  amount unit ingredient
1      1  cup      flour
2      2 cups       salt
3      3   oz      sugar
```

 # Create vector

```
vec <- c(1, "abc", “TRUE”)
vec[1]  ## Select elements from a vector
```

 # Create character vector/use of paste

```
tests <- paste("test", 1:200, sep='')
tests.all <- paste(tests, collapse=" ")
even <- paste("even", seq(from=2, by=2, length=1000), sep='')
```

 # Create a list

```
l <- list(1,2,3)
names(l) <- c("a","b","c")
# select from a list
l[[1]]  #####list 当中第一列数
l[["a"]]  ####OR  l$a

l <- list(); for(i in 1:12) l[[i]] <- rpois(n=100, lambda=5)
```

OR
l = list()
for(i in 1:12){
  l[[i]] = rpois(100, lambda=5)}

 # Create matrix

b <- matrix(c(seq(10, 120, by=10)), nrow=3, ncol=4, byrow=T)
> t(B) # transpose of B

```
> m <- matrix(seq(10, 120, by=10), nrow=3, byrow=T)
> m
     [,1] [,2] [,3] [,4]
[1,]   10   20   30   40
[2,]   50   60   70   80
[3,]   90  100  110  120
> n <- matrix(seq(10, 120, by=10), nrow=3, byrow=F)
> n
     [,1] [,2] [,3] [,4]
[1,]   10   40   70  100
[2,]   20   50   80  110
[3,]   30   60   90  120
```

 # Create table

athTab <- table(athletes$Sex, athletes$Sport)
t <- table(infants$ed[infants$married=="Married" & infants$parity==1]) ##midterm2

 # Data frame manipulation

# add a column “bmi” to the data frame
family2 <- data.frame(family, bmi)
# how many rows in the data frame
n.wr <- nrow(wr1500m)
# find row names of a data frame
model <- row.names(mtcars)
# Change NA data
wr1500m$month[is.na(wr1500m$month)] <- 6

 # Subset data

- mtcars$disp ####OR mtcars[,"disp"] ####OR mtcars[,3] ##return 第几列
- family.men<-subset(family, gender=="m") OR family.men <- family[family$gender=="m",]
- family.30y68i<-subset(family, age>"30" & height<"68")
- housing.less3.berkeley <- housing[(housing$br < 3) & (housing$city=="Berkeley"), c("city", "br", "date")]
- wr1500m$month[is.na(wr1500m$month)] <- 6

double brackets used to select elements in a list

- rainfall.subset <- rain[[1]][yr.2000]

find specific value in the data frame

- wr.name <- wr1500m[which.min(wr1500m$times), 4] #[, 4] select the 4th column; [,] select all data in that row
- length(athletes$Name[which(athletes$Sex == "F")])

Create [f3] a subset of family of people whose name starts with T
- f3 <- family[substr(family$name, 1, 1)=="T" ,] ##从第一个到第一个

Select certain values of a column in a data
- mw <- mean(infants$bwt[infants$gestation>=259])

Select top 5 data in a data frame
- top5 <- as.character(SO2012Ctry$Country[order(SO2012Ctry$Total, decreasing = TRUE)[1:5]])

Select a row with minimum or maximum value
- f4 <- family[family$age == min(family$age),]

Select 1st column of the datas
- data[,1]

Remove 1st column
- mtcars[mtcars$gear==4, -1] # return everything where gear=4, but remove 1st column

Remove rows from data set
- mtcars[-c(2, 4, 6),]

Remove certain data from the data frame using subset.
- mtcars2<-subset(mtcars, gear!=4)

 # Use of %in% and %%

%in% whether an element of ….

```
> all(1:6 %in% 0:36)
[1] TRUE
> all(1:60 %in% 0:36)
[1] FALSE
```

On a similar note, if you want to check whether any of the elements is TRUE you can use `any`

```
> any(1:6 %in% 0:36)
[1] TRUE
> any(1:60 %in% 0:36)
[1] TRUE
> any(50:60 %in% 0:36)
[1] FALSE
```

%% whether 被某数整除

```
> y = (x %% 10 == 0)
> y[10]
[1] TRUE
> y[11]
[1] FALSE
```

20 pts : Part II, plotting

 # Make box plot

boxplot(iris$Sepal.Length ~ iris$Species) # Make a box plot of Sepal Length by Species, midterm2

 # Make bar plot

- athTab <- table(athletes$Sex, athletes$Sport)
- barplot(athTab[, orderSport], beside = TRUE, las=3, cex.names = 0.8, main="London 2012 Male vs. Female athlets") ##hw2

 # Scatter plot with color

Make a scatterplot of petal width (y-axis) versus petal length (x-axis); plot(y~x)

plot(iris$Petal.Length, iris$Petal.Width, main="graph", xlab="Petal Length", ylab="Petal Width", col=as.numeric(iris$Species))

Use "*" as the plotting symbol

plot(infants$gestation, infants$bwt, pch='*')

Make a scatterplot of (sepal length / petal length) (y-axis) as a function of index (order, x-axis); Color the plotting symbol by Species (any 3 colors)

ratio <- iris$Sepal.Length / iris$Petal.Length

plot(ratio ~ seq(1, length(iris$Sepal.Length)),
xlab = "Index", ylab = "Ratio of Sepal to Petal length", col = iris$Species)

 # Step plot

plot(wr1500m$new_year, wr1500m$times, type="s")

 # Connect two points (2000,5) and (2014, 6)

lines (x=c(2000, 2014), y= c(5, 6))

 # Add text on the plot

text(x=3, y=3, "China", cex=1 ,col="blue") ##x&y indicate the coordinate position of the text

text(x=mds[,1], y=mds[,2], labels=as.character(unique(speechesDF$Pres)), col = cols[presParty[rownames(presDist)]]) #hw7

 # Make histogram and density curve

hist(infants$age, prob=T, xlab="Mother's age")

Add density plot to the historgram

lines(density(infants$age, na.rm=T), col="blue")

 # Use of liner model

```
### use the lm function to fit a line of a quadratic
### e.g. y ~ x or y ~ x + I(x^2)
fitModel = function(x, y, degree = 1){
if(degree == 1){
   coeff = coef(lm(y ~ x))
 }
 if(degree == 2){
   coeff = coef(lm(y ~ x + I(x^2)))
 }
 return(coeff)
}
```

15 pts : Part III, apply statements (also know how to use by and replicate)

 # The use of 'by'

Create a variable [max.petal.width] _a numeric vector of length 3 that has the maximum petal length for each iris species.

```
max.petal.width <- as.vector(by(iris$Petal.Width, iris$Species, max)) ???? ##找每个 species 中 width 最大的 by(data, list of factors each length nrow(data),function )
```

 # Use of sample and replicate

```
# k 个骰子扔 B 次，求每次骰子的和
dice_sum <- function(k=2, B=100){
  replicate(B, sum(sample(1:6, size=k, replace=T)) )
}  ## replicate: apply a function multiple times

replicate (10, sample(c(1:6), 1)) 重复这个 process 10 次
replicate (sample(c(1:6), 10)) 复制这个结果 10 次
# different from rep.
> n= rep(1:4, 2)
> n
[1] 1 2 3 4 1 2 3 4
> z<- rep(1:5, each=4)
> z
 [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5
```

 # sapply

# Create [first.cache], a vector where each entry is the _first_ element of the corresponding vector in the list Cache500
first.cache <- sapply(Cache500, "[", 1) ###OR
first.cache <- sapply(Cache500, function(x) x[1])

# a vector of length 5 with the number of measurements at each station in the year 1989 (use [day])
n1989.rain <- sapply(day, function(x) sum(floor(x)==1989)) ##floor(x) 舍去法 OR
n1989.rain <- sapply(day, function(x) sum(x >= 1989 & x<1990))

 # sapply with function

Create [mean.long.cache], a vector where mean.long.cache[i] is the mean of Cache500[[i]] IF it has 50 or more entries. NA IF Cache500[[i]] has less than 50 entries.

- mean.long.cache <- sapply(Cache500, function(x){ if(length(x)>=50) return(mean(x)) else return(NA)}) ##OR
- mean.long.cache <- sapply(Cache500, function(x) if(length(x) >= 50) mean(x) else NA)
- words <- sapply(speechWords, function(x) length(unlist(x)))

 # tapply:

We want to break it up into groups; Within each group, we want to apply a function；分别找几个 sub group 的 mean, sd etc.

tapply(Summary Variable(最后要 return 的数据), Group Variable(不同的 group), Function)

```
genBootY = function(x, y, rep = TRUE){
result = tapply(y, x, function(y) sample(y,size = 10, replace = T))
 result = unlist(result)
 return(result)
}
```

```
> ## Generic Example
> ## tapply(Summary Variable, Group Variable, Function)
>
> ## Medical Example
> tapply(medical.example$age, medical.example$treatment, mean)
Treatment   Control
 62.26883  60.30371
> ## Baseball Example
> tapply(baseball.example$batting.average, baseball.example$team,
        max)
   Team A    Team B    Team C    Team D    Team E
0.3784396 0.3012680 0.3488655 0.2962828 0.3858841
```

 # lapply:

Create [l], a list with 12 elements, each a vector of length 100. Each vector of length 100 of Poisson (hint:rpois) random variables with mean 5

```
l <- lapply(1:12, function(x) rpois(100, 5))
lapply(food, "[",  1) #1st element of each vector in the list
```

15 pts : Part IV, functions

 # Remove columns/rows of a matrix

```
m <- matrix(c(1, 5, 3, 8, 2, 5, 7, 9), ncol=4, byrow=T)
```

```
> m
     [,1] [,2] [,3] [,4]
[1,]    1    5    3    8
[2,]    2    5    7    9
> m[,-1]
     [,1] [,2] [,3]
[1,]    5    3    8
[2,]    5    7    9
> m[,-2]
     [,1] [,2] [,3]
[1,]    1    3    8
[2,]    2    7    9
```

```
firstColToNames <- function(m){
if(length(dim(m))!=2) print("m is not a matrix or dataframe")
 else{
   names <- as.character(m[,1])
   m <- m[, -1]
   rownames(m) <- names
   return(m)
 }
}

# a function to convert temperature.
TempConv <- function(t, scale){
 if(scale=="F") return((t - 32) * 5/9)
 else if(scale=="C") return(t * 9/5 + 32)
}


# a function to convert cup to gram, oz to ml; hw4
recipeConversion <- function(recipe){
    if (!all(colnames(recipe)==c("amount", "unit", "ingredient")))
       stop("Error unexpected column names")
    recipe$unit=as.character(recipe$unit)
    for (i in 1:nrow(recipe))
    if(recipe$unit[i] == "cup" | recipe$unit[i] == "cups") {
     recipe$unit[i] = "ml"
     recipe$amount[i]= 5*round(236.5*recipe$amount[i]/5)
```

```
    }
     else if (recipe$unit[i]=="oz"){
       recipe$amount[i]=5*round(28.3*recipe$amount[i]/5)
     }
    recipe.metric = recipe
    return(recipe.metric)
}

# Bootstrap example; hw4
bootstrapVarEst <- function(x, B){
  bootstrap.mean=c()
  for(i in 1:B){
    new.sample = sample(x, length(x), replace=T)
    bootstrap.mean[i]= mean(new.sample)
  }
  boot.sigma2.est = var(bootstrap.mean)
  return(boot.sigma2.est)
}

# Sample without one value; hw4
jackknifeVarEst <- function(x){
  jack.mean=c()
  for(i in 1:length(x)){
    new_sample = sample(x[-i], length(x)-1, replace=T)
    jack.mean[i]= mean(new_sample)
  }
  jack.sigma2.est = var(jack.mean)
  return(jack.sigma2.est)
}


# While loop; hw5
bml.sim <- function(r, c, p){
  m = bml.init (r,c,p)
  count=0
  while(count < 5000) {
    if (bml.step(m)[[2]] == TRUE) {
      m=bml.step(m)[[1]]
      count=count+1
    }else{
      return(count)
    }
  }
  return(count)
}

# Create an empty matrix first; hw6
sim.doctors <- function(initial.doctors, n.doctors, n.days, p){
  m = matrix(0, nrow=n.doctors, ncol=n.days)
  for (k in 1:n.days){
    n= sample (1:n.doctors, 2, replace = F)
```

```
   if ( !(m[n[1],k]== m[n[2],k])){
     x=n[m[n, k]==0]
     m[x, k:n.days] = sample (c(1,0),1,prob=c(p, 1-p))
    }
  }
  return(m)
}
```

```
# Create a power matrix; <k> : an integer
-    powers <- function(x, k){
     x.powers <- sapply (1:k, function(a) {x^a})
     colnames(x.powers) <- colnames(x.powers, do.NULL = FALSE, prefix = "x^")
     return(x.powers)
     }
```

15 pts : Part V, simulations

 # Use of sample and replicate

```
# k 个骰子扔 B 次，求每次骰子的和
dice_sum <- function(k=2, B=100){
   replicate(B, sum(sample(1:6, size=k, replace=T)) )
}   ## replicate: apply a function multiple times

# Create [w], a random permutation of the numeric values of a deck of cards
w <- sample(rep(seq(1, 13), each = 4), 52, replace = F)

# Sampling the data other than i
-    new_sample = sample(x[-i], length(x)-1, replace=T)


# Bootstrap
n <- 100
boot_mean <- rep(0,100)
for (i in 1:n){
  dat <- sample(x=iris$Sepal.Length, size=nrow(iris), replace=T)
  boot_mean[i] <- mean(dat)
}
var(boot_mean)

-    bootstrapVarEst <- function(x, B){
     bootstrap.mean=c()
```

```
for(i in 1:B){ new.sample = sample(x, length(x), replace=T)
bootstrap.mean[i]= mean(new.sample)}
boot.sigma2.est = var(bootstrap.mean)
return(boot.sigma2.est)
}
```

20 pts : Part VI, string manipulation and regular expressions

Table 11.1: Some POSIX regular expression character classes.

`[:alpha:]`	Alphabetic (only letters)
`[:lower:]`	Lowercase letters
`[:upper:]`	Uppercase letters
`[:digit:]`	Digits
`[:alnum:]`	Alphanumeric (letters and digits)
`[:space:]`	White space
`[:punct:]`	Punctuation

 # Meta characters:

"[[:lower:]]at" means: any lowercase letter in any language, followed by the letter *`a'*, followed by the letter *`t'*. Eg. [[:lower:]]at ⇒ The cat sat on the mat .

"[a-z]at" means: any (English) lowercase letter, followed by the letter *`a'*, followed by the letter *`t'*.

^ means: the beginning of the string;

- ^[Tt]he ⇒ The cat sat on the mat. vs. [Tt]he ⇒ The cat sat on the mat.
- text2 <- grep("^d.+", phrases) ##a vector [text2] that lists the elements in phrases that START with the letter "d"

"[^c]at" ⇒ The cat sat on the mat . [^c]at means: any letter *except `c'*, followed by the letter *`a'*, followed by the letter *`t'*.

$ The dollar character matches the end of a piece of text. the letter *`a'*, followed by the letter *`t'*, followed by any character, *at the end of the text*.

at.$ ⇒ The cat sat on the mat.

text1 <- grep("d.$", phrases) ## phrases that 2nd last letter is “d”, midterm4

\\< at the beginning of the phrase
\\> at the end of every single phrase;

```
> chvec
[1] "him"  "thim" "hims" "ahim"
> grep("\\<him", chvec)
[1] 1 3
> grep("him\\>", chvec)
[1] 1 2 4
> grep("\\<(him)\\>", chvec)
[1] 1
```

\\: 不运行
Eg: chvec=c('star', 'st*r', '***'); a= grep("*", chvec)

. The full stop character matches any single character ".at" ⇒ The cat sat on the mat .
at. ⇒ The cat sat on the mat.
at[.] ⇒ The cat sat on the mat.

[] Within square brackets, most metacharacters revert to their literal meaning. For example, [.] means a literal full stop.
"at[.]" means: the letter *`a'*, followed by the letter *`t'*, followed by a full stop.
"at[.]" ⇒ The cat sat on the mat.
at[.]? ⇒ The cat sat on the mat. *optionally* followed by a full stop.

| : 或者 ； cat|sat ⇒ The cat sat on the mat.
cats = c("diplocat", "Hi cat", "mat", "at", "t!", "ct")
grep("\\<(cat|at|t)\\>", cats)
[1] 2 4 5

+ means that the subpattern can occur one or more times
grep("c.+t", test)
[a-z]+ ⇒ The cat sat on the mat .
grep("I{2,}", movies) 出现I两次以上的位置
grep(".+((at){2,}.+)", temp) #something in front of and after a match to any multiple of "at", _two or more times_ (atat" or "atatat" etc.)
"(.at)+" ##one or more repetitions of: any letter at all, followed by the letter `a', followed by the letter `t', followed by a space.

* means that the subpattern can occur zero or more times

 # Use of strsplit (create a list)

strsplit(example, "") ##split string by character
strsplit(example, " ") ##split string by word
Select the 3rd element from left in the splitted string; 在 split 的 string 中截取第三个. hw7 (split string by space)

- speechYr <- as.numeric(sapply(strsplit(tempDates, " "), "[[", 3))
- words = unlist(strsplit(sen, "[[:space:]]+|[[:punct:]]+"))
- strsplit(".a.b.c","\\.") ##表示不运行

[[1]]
[1] "" "a" "b" "c"

- strsplit("11/03/2013","/")

[[1]]
[1] "11" "03" "2013"

 # Remove certain lines of text
randj <- randj[-(1:65)]

 # Use of grep/gsub
# "cat" change to "cot"
gsub("(.)at", "\\1ot", text)
"The cot sot on the mot." ### \1 refers to the first subpattern (reading from the left), \2 refers to the second subpattern, and so on.

```
>x<-c("abc","bcd","cde","def")
>grep("bc",x)
[1] 1 2

>grep("bc",x,value=TRUE)
[1] "abc" "bcd"
>x[grep("bc",x)]
[1] "abc" "bcd"
```

 # Use of gregexpr/reexpr
regexpr("\\(.*\\)", movies[1]) ####找一个 string 中的第几个
The gregexpr() does the same thing as regexpr("D", x), except that its returned object is a list rather than a vector.

```
> x <- c("ABCDE", "CDEFG", "FGHIJ")
> regexpr("D", x)
[1]  4  2 -1
attr(,"match.length")
[1]  1  1 -1
```

 # Use of substr

```
# Write a vector of the same length with only the first [k] characters from the original vector
abbreviate <- function(vector, k){return(substr(vector, start=1, stop=k))}

# Create [f3] a subset of family of people whose name starts with T
–    f3 <- family[substr(family$name, 1, 1)=="T" , ]  ##从第一个到第一个
```

```
# Find the most common letter in a string; midterm3; chvec=c('s', 'st*r', '***')
mostCommonLetter <- function(chvec){
  new.chvec = tolower(chvec)
  new.chvec = gsub("'", "", new.chvec)
  new.chvec = gsub("[[:digit:]]+", "", new.chvec)
  new.chvec = gsub("[[:punct:]]+", "", new.chvec)
  new.chvec = gsub("/", "", new.chvec)
  x = strsplit(new.chvec, "")
  tab = table(unlist(x))
  lettermax = max(tab)
  letter = names(tab)[tab==max(tab)]
  return(letter)
}
```

hw1: sequence/ data frame
hw2: plot/symbol/barplot
hw3: ggplot/apply
hw4: fxn & bootstrap (recipe)
hw5: bml function
hw6: fxn / traffic simulation
hw7: text analysis
hw8: genboot &simulation

```
# Linear model: hw8
mod <- lm(mpg~., data=mtcars)
mod$fitted
mod$coef
mod$residuals
```