Lesson 7:
Data Manipulation

---useful functions and packages in R

Chris Qi

Part1:
for loops in R (循环)

- For loops are for iterations

```
> primes_list <- list(2, 3, 5, 7, 11, 13)

> for (i in 1:length(primes_list)) {
    print(primes_list[[i]])
  }
[1] 2
[1] 3
[1] 5
[1] 7
[1] 11
[1] 13
```

parts of a for loop

- sequence:

```
for (i in 1:length(primes_list)) {   sequence
  print(primes[[i]])
}
```

- body:

```
for (i in 1:length(primes_list)) {
  print(primes[[i]])                    body
}
```

- Output:

```
for (i in 1:length(primes_list)) {
  print(primes[[i]])
}
                                  output?
```

looping over columns in a data frame

```
> df <- data.frame(
    a = rnorm(10),
    b = rnorm(10),
    c = rnorm(10),
    d = rnorm(10)
  )

> for (i in 1:ncol(df)) {
    print(median(df[[i]]))
  }
```

Part 2:
apply functions

- apply
- lapply
- sapply

apply(object, margin, function, ...)

- margin specifies which dim to iterate.
- 1=row
- 2=column
- c(1,2) both
- ... additional function options

apply(object, margin, function, ...)

```
> Duckweed.mat<-matrix(c(10,20,30,40,50,60,70,80,90,100,10,30,50,80,100,150,200,250,2
70,300,10,30,36,80,96,106,110,130,136,144,10,15,30,50,70,86,95,100,105,190,10,40,50,6
6,78,96,107,120,144,157,10,30,57,98,106,130,160,177,189,198),nrow=10,byrow=FALSE)
> 
```

```
> rownames(Duckweed.mat)<-c("Day1","Day2","Day3","Day4","Day5","Day6","Day7","Day8","
Day9","Day10")
> colnames(Duckweed.mat)<-c("R1","R2","R3","R4","R5","R6")
> Duckweed.mat
       R1  R2  R3  R4  R5  R6
Day1   10  10  10  10  10  10
Day2   20  30  30  15  40  30
Day3   30  50  36  30  50  57
Day4   40  80  80  50  66  98
Day5   50 100  96  70  78 106
Day6   60 150 106  86  96 130
Day7   70 200 110  95 107 160
Day8   80 250 130 100 120 177
Day9   90 270 136 105 144 189
Day10 100 300 144 190 157 198
```

lappy and sapply

```
> max(Duckweed.mat[1,])
[1] 10
>
```

```
> max(Duckweed.mat[2,])
[1] 40
> max(Duckweed.mat[3,])
[1] 57
> max(Duckweed.mat[4,])
[1] 98
> max(Duckweed.mat[5,])
[1] 106
> max(Duckweed.mat[6,])
[1] 150
> max(Duckweed.mat[7,])
[1] 200
> max(Duckweed.mat[8,])
[1] 250
> max(Duckweed.mat[9,])
[1] 270
```

```
> for (i in 1:10){
+ row<-Duckweed.mat[i,]
+ max<-max(row)
+ print(max)
+ }
[1] 10
[1] 40
[1] 57
[1] 98
[1] 106
[1] 150
[1] 200
[1] 250
[1] 270
[1] 300
```

```
> apply(Duckweed.mat,1,max)
 Day1  Day2  Day3  Day4  Day5  Day6  Day7  Day8  Day9 Day10
   10    40    57    98   106   150   200   250   270   300
```

```
> apply(Duckweed.mat,2,max)
 R1  R2  R3  R4  R5  R6
100 300 144 190 157 198
>
```

```
> Duckweed.df<-data.frame(R1=c(10,20,30,40,50,60,70,80,90,100), R2=c(10,30,50,80,100,
150,200,250,270,300), R3=c(10,30,36,80,96,106,110,130,136,144), R4=c(10,15,30,50,70,8
6,95,100,105,190), R5=c(10,40,50,66,78,96,107,120,144,157), R6=c(10,30,57,98,106,130,
160,177,189,198))
> class(Duckweed.df)
[1] "data.frame"
>
```

```
> Duckweed.df
    R1  R2  R3  R4  R5  R6
1   10  10  10  10  10  10
2   20  30  30  15  40  30
3   30  50  36  30  50  57
4   40  80  80  50  66  98
5   50 100  96  70  78 106
6   60 150 106  86  96 130
7   70 200 110  95 107 160
8   80 250 130 100 120 177
9   90 270 136 105 144 189
10 100 300 144 190 157 198
```

```
> apply(Duckweed.df,1,mean)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333 104.66667 123.66667 142.83333
 [9] 155.66667 181.50000
```

```
> Duckweed.df$Day<-as.factor(1:10)
> apply(Duckweed.df,1,mean)
3: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
4: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
5: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
6: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
7: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
8: In mean.default(newX[, i], ...) :
  argument is not numeric or logical: returning NA
9: In mean.default(newX[, i], ...) :
```

```
> apply(Duckweed.df[,2:7],1,mean)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333 104.66667 123.66667 142.83333
 [9] 155.66667 181.50000
```

```
> apply(Duckweed.df[,2:7],1,mean)
 [1]  10.00000  27.50000  42.16667  69.
 [9] 155.66667 181.50000
> Duckweed.df
   Day  R1  R2  R3  R4  R5  R6
1    1  10  10  10  10  10  10
2    2  20  30  30  15  40  30
3    3  30  50  36  30  50  57
4    4  40  80  80  50  66  98
5    5  50 100  96  70  78 106
6    6  60 150 106  86  96 130
7    7  70 200 110  95 107 160
8    8  80 250 130 100 120 177
9    9  90 270 136 105 144 189
10  10 100 300 144 190 157 198
```

```
> apply(Duckweed.df[,-1],1,mean)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333 104.66667 123.66667 142.83333
 [9] 155.66667 181.50000
```

```
> apply(Duckweed.df[,-c(1,2,4,6)],1,mean)
 [1]  10.00000  25.00000  45.66667  76.00000  92.00000 122.00000 151.66667 175.66667
 [9] 188.00000 229.33333
```

```
> Duckweed.df
    Day  R1  R2  R3  R4  R5  R6
1     1  10  10  10  10  10  10
2     2  20  30  30  15  40  30
3     3  30  50  36  30  50  57
4     4  40  80  80  50  66  98
5     5  50 100  96  70  78 106
6     6  60 150 106  86  96 130
7     7  70 200 110  95 107 160
8     8  80 250 130 100 120 177
9     9  90 270 136 105 144 189
10   10 100 300 144 190 157 198
```

```
> Duckweed.df
    Day  R1  R2  R3  R4  R5  R6
1     1  10  10  10  10  10  10
2     2  20  30  30  15  40  30
3     3  30  50  36  30  50  57
4     4  40  80  80  50  66  98
5     5  50 100  96  70  78 106
6     6  60 150 106  86  96 130
7     7  70 200 110  95 107 160
8     8  80 250 130 100 120 177
9     9  90 270 136 105 144 189
10   10 100 300 144 190 157 198
```

```
> Duckweed.df[6,7]<-NA
> Duckweed.df
   Day  R1  R2  R3  R4  R5  R6
1    1  10  10  10  10  10  10
2    2  20  30  30  15  40  30
3    3  30  50  36  30  50  57
4    4  40  80  80  50  66  98
5    5  50 100  96  70  78 106
6    6  60 150 106  86  96  NA
7    7  70 200 110  95 107 160
8    8  80 250 130 100 120 177
9    9  90 270 136 105 144 189
10  10 100 300 144 190 157 198
```

```
> Duckweed.df
   Day  R1  R2  R3  R4  R5  R6
1    1  10  10  10  10  10  10
2    2  20  30  30  15  40  30
3    3  30  50  36  30  50  57
4    4  40  80  80  50  66  98
5    5  50 100  96  70  78 106
6    6  60 150 106  86  96  NA
7    7  70 200 110  95 107 160
8    8  80 250 130 100 120 177
9    9  90 270 136 105 144 189
10  10 100 300 144 190 157 198
> apply(Duckweed.df[,-1],1,mean)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333        NA 123.66667 142.83333
 [9] 155.66667 181.50000
```

na.rm

- apply(object, margin, function, ...)

```
> apply(Duckweed.df[,-1],1,mean)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333        NA 123.66667 142.83333
 [9] 155.66667 181.50000
> apply(Duckweed.df[,-1],1,mean,na.rm=TRUE)
 [1]  10.00000  27.50000  42.16667  69.00000  83.33333  99.60000 123.66667 142.83333
 [9] 155.66667 181.50000
```

lapply() and sapply()

- lapply(object, function, ...)
- object: list, vector, data frame
- OUTPUT: only list

```
> CAGO.list<-list(Diet1=c(2,5,4,5,3,5,4,4,4,5), Diet2=c(8,5,6,5,7,7,6,8,8,3), Diet3=c(3,4
,2,5,2,6,5,6,2,4), Diet4=c(2,2,3,2,5,2,4,3,5,7))
> CAGO.list
$Diet1
 [1] 2 5 4 5 3 5 4 4 4 5

$Diet2
 [1] 8 5 6 5 7 7 6 8 8 3

$Diet3
 [1] 3 4 2 5 2 6 5 6 2 4

$Diet4
 [1] 2 2 3 2 5 2 4 3 5 7
```

```
> lapply(CAGO.list,mean)
$Diet1
[1] 4.1

$Diet2
[1] 6.3

$Diet3
[1] 3.9

$Diet4
[1] 3.5
```

```
> CAGO.df<-data.frame(Diet1=c(2,5,4,5,3,5,4,4,4,5), Diet2=c(8,5,6,5,7,7,6,8,8,3), Diet3=c
(3,4,2,5,2,6,5,6,2,4), Diet4=c(2,2,3,2,5,2,4,3,5,7))
> CAGO.df
   Diet1 Diet2 Diet3 Diet4
1      2     8     3     2
2      5     5     4     2
3      4     6     2     3
4      5     5     5     2
5      3     7     2     5
6      5     7     6     2
7      4     6     5     4
8      4     8     6     3
9      4     8     2     5
10     5     3     4     7
```

- lapply assumes we are working with column

```
> lapply(CAGO.df,mean)
$Diet1
[1] 4.1

$Diet2
[1] 6.3

$Diet3
[1] 3.9

$Diet4
[1] 3.5
```

```
> Random<-c("This","is","a","random","vector")
> Random
[1] "This"   "is"     "a"      "random" "vector"
>
```

```
> lapply(Random,nchar)
[[1]]
[1] 4

[[2]]
[1] 2

[[3]]
[1] 1

[[4]]
[1] 6

[[5]]
[1] 6
```

sapply()

- sapply(object, function, ...)
- OUTPUT IS SIMPLIFIED (vector, matrix, list)

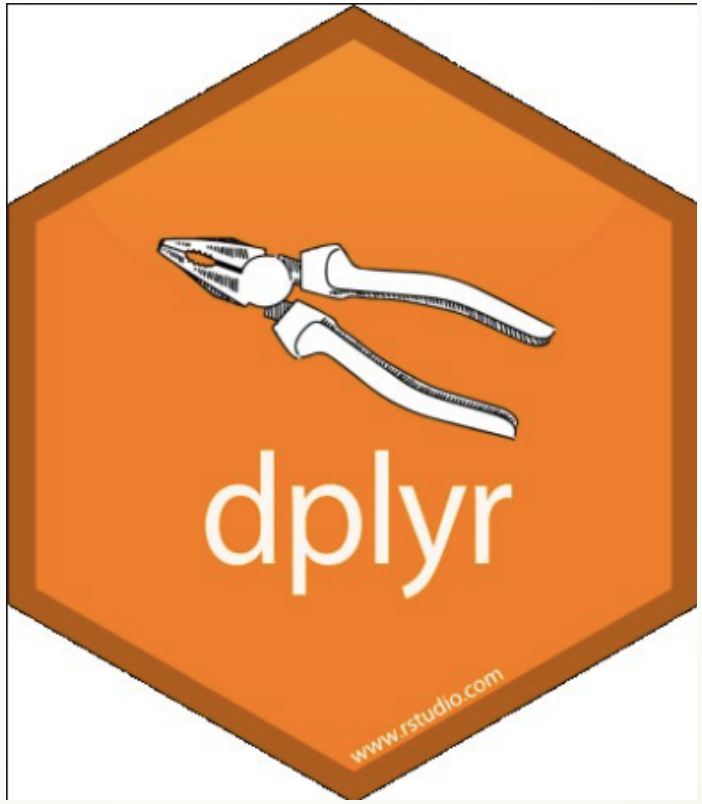
```
> CAGO.list
$Diet1
 [1] 2 5 4 5 3 5 4 4 4 5

$Diet2
 [1] 8 5 6 5 7 7 6 8 8 3

$Diet3
 [1] 3 4 2 5 2 6 5 6 2 4

$Diet4
 [1] 2 2 3 2 5 2 4 3 5 7
```

```
> sapply(CAGO.list,mean)
Diet1 Diet2 Diet3 Diet4
  4.1   6.3   3.9   3.5
```

```
> CAGO.df
   Diet1 Diet2 Diet3 Diet4
1      2     8     3     2
2      5     5     4     2
3      4     6     2     3
4      5     5     5     2
5      3     7     2     5
6      5     7     6     2
7      4     6     5     4
8      4     8     6     3
9      4     8     2     5
10     5     3     4     7
> sapply(CAGO.df,mean)
Diet1 Diet2 Diet3 Diet4
  4.1   6.3   3.9   3.5
```

```
> Random
[1] "This"   "is"     "a"      "random" "vector"
> sapply(Random,nchar)
  This     is      a random vector
     4      2      1      6      6
```

Part 3:
dplyr package



What is dplyr

- Tools for data exploration and transformation
- Intuitive to write and easy to read
- Super-fast on data frames

- install.packages("hflights")
- hflights: Dataset on commercial domestic flights that departed Houston (IAH and HOU) in 2011.

Basic single table (df) verbs

1. `filter`: for subsetting variables
2. `select`: for subsetting rows
3. `arrange`: for re-ordering rows
4. `mutate`: for adding new columns
5. `summarise` or `summarize`: for reducing each group to a smaller number of summary statistics

R markdown

dplyr demo in R markdown

Resources

- Kevin Markham's tutorial
- Official dplyr reference manual and vignettes on CRAN: vignettes are well-written and cover many aspects of dplyr
- July 2014 webinar about dplyr (and ggvis) by Hadley Wickham and related slides/code: mostly conceptual, with a bit of code
- dplyr tutorial by Hadley Wickham at the useR! 2014 conference: excellent, in-depth tutorial with lots of example code (Dropbox link includes slides, code files, and data files)
- dplyr GitHub repo and list of releases