R 프로그래밍 #5

2019. 04. 03

한국생명공학연구원
김하성

Summary of last classes

- **R objects**: vector, factor, **matrix**, **data.frame**, list *class()
- **R object types**: numeric, character, logical *typeof()
- **Indexing** of vector, matrix, data.frame, list
- How **to make/to use a function**
- How to use **if, for**

Matrix

```
matrix(0, nrow=2, ncol=2)
matrix(c(1,2,3,4), nrow=2, ncol=2)
matrix(c(1,2,3,4), 2, 2)
matrix(c(1,2,3,"4"), 2, 2)
matrix(c(1,2,3,TRUE), 2, 2)
```

if / if else


```
x <- 5
if(x > 0){
   print("Positive number")
}

if(x > 0) print("Positive number")


x <- -5
if(x > 0){
       print("Non-negative number")
} else {
       print("Negative number")
}

if(x > 0)
       print("Non-negative number")
else
       print("Negative number")
```

for loop

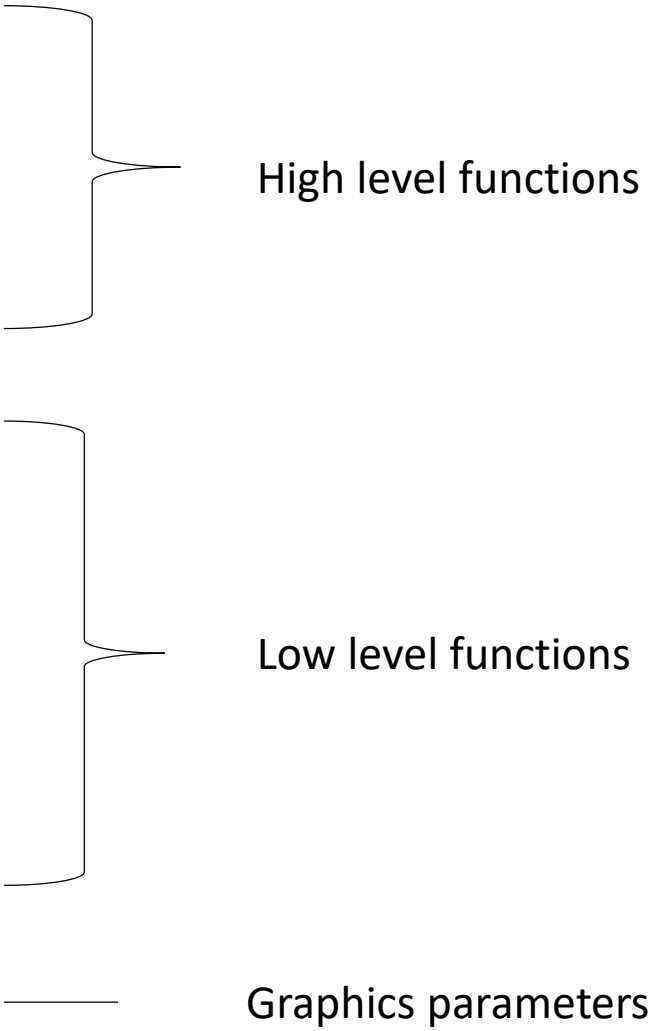
```
for(val in sequence){
    statement
}
```

Counting even numbers

```
x <- c(2,5,3,9,8,11,6)
count <- 0
for
    if

print(count)
```

Graphics in R

- plot()
- hist()
- barplot()
- boxplot()
- heatmap()
- abline()
- arrows()
- axis()
- lines()
- points()
- text()
- legend()
- par()



Scatter plot

```
x1 <- c(1,2,5,7)
y1 <- c(1,6,7,8)
xy<-data.frame(x1, y1)
write.table(xy, file="table_write.txt", quote=F)
myxy <- read.table(file="table_write.txt")
myxy
class(myxy)
names(myxy)
plot(myxy)
plot(myxy$x1, myxy$y1)
plot(x=myxy$x1, y=myxy$y1)
plot(y=myxy$x1, x=myxy$y1)
plot(y1~x1, data=myxy)*
```

* Generic function: call plot.formula if input parameter is formula
call plot.default if input parameter is x or y

Histograms

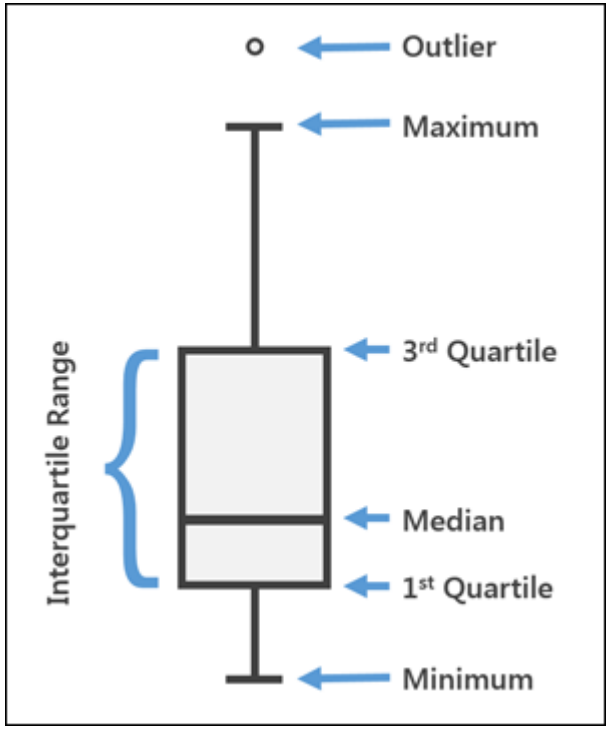
- Plots the frequencies that data appears within certain ranges

```
x <- rnorm(100)
?rnorm
hist(x)
hist(x, br=20)
hist(x, br=20, xlim=c(-3,3))
hist(x, br=20, xlim=c(-3,3), main="Main text", xlab="X label")
```

Boxplot

- A graphical view of the median, quartiles, maximum, and minimum of a data set

```
boxplot(x)
y <- rnorm(100, 1, 1)
boxplot(y)
xy <- data.frame(x, y)
boxplot(xy)
class(xy)
```



Barplot

```
x <- sample(1:12, 200, replace = T)
tab_x <- table(x)
y <- sample(1:12, 200, replace = T)
tab_y <- table(y)
tab_xy <- rbind(tab_x, tab_y)
tab_xy
barplot(tab_xy)
barplot(tab_xy, beside = T)
barplot(tab_xy, beside = T, col=c("darkblue","red"))
barplot(tab_xy, beside = T, col=c("darkblue","red"), xlab="Month")
barplot(tab_xy, beside = T, col=c("darkblue","red"), xlab="Month", horiz=TRUE)
```

We want to make a generally available function that reads excel formatted victor (plate reader) data.

Read excel data – Lecture 03

```
mydata <- read_excel("Rprog04-fl.xls", sheet=2, skip = 6, col_names=F)
myod <- as.data.frame(mydata[1:8, ])
mygfp <- as.data.frame(mydata[12:21, ])

## change datatype from character to numeric
myod[,1] <- as.numeric(myod[,1])
mygfp[,1] <- as.numeric(mygfp[,1])

# OD
myod_treat <- myod[2:4,]
myod_control <- myod[5:7,]

sample_names <- paste("Sample", c(1:12), sep="")
replicate_labels <- paste("Rep", c(1:3), sep="")

rownames(myod_treat) <- replicate_labels
colnames(myod_treat) <- sample_names
rownames(myod_control) <- replicate_labels
colnames(myod_control) <- sample_names

mean_treat <- colMeans(myod_treat)
mean_control <- colMeans(myod_control)
plot(mean_treat, type="h")
barplot(mean_treat, ylim=c(0,1))

mean_test <- data.frame(mean_treat, mean_control)
barplot(t(mean_test), ylim=c(0,1), beside=T)
```

Too case specific!

Problems that need to be solved by programming!

Excel data

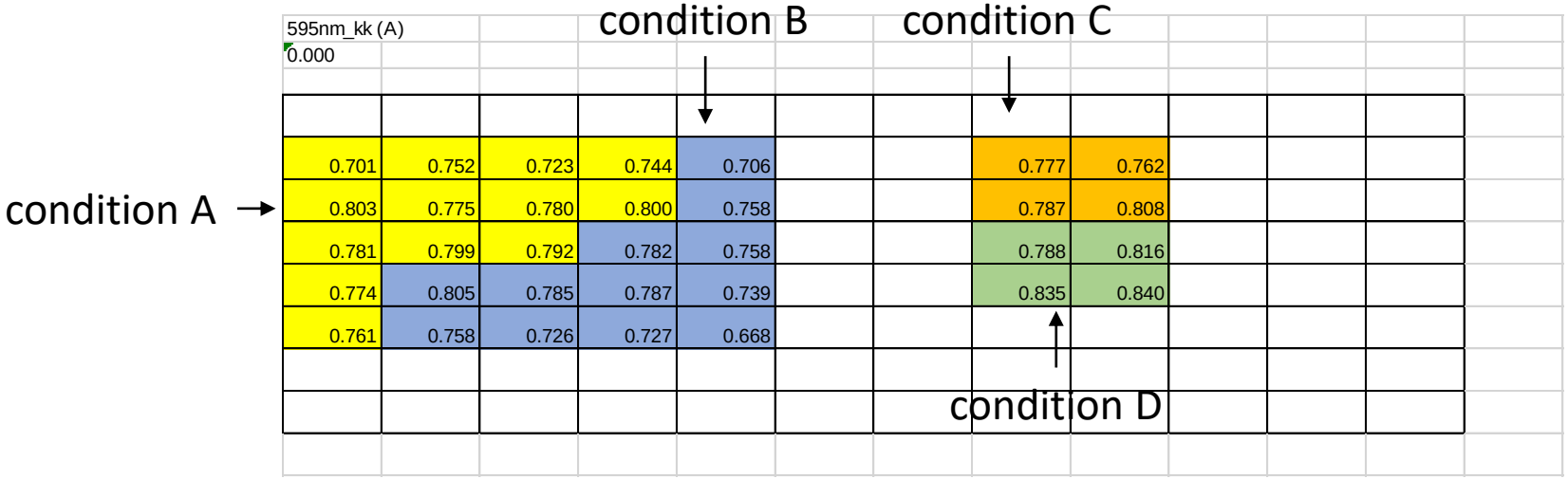
595nm_kk (A)
0.000

0.701	0.752	0.723	0.744	0.706	0.723	0.767	0.777	0.762	0.798	0.793	0.821
0.803	0.775	0.780	0.800	0.758	0.749	0.807	0.787	0.808	0.826	0.824	0.814
0.781	0.799	0.792	0.782	0.758	0.756	0.838	0.788	0.816	0.852	0.834	0.842
0.774	0.805	0.785	0.787	0.739	0.713	0.827	0.835	0.840	0.846	0.863	0.870
0.761	0.758	0.726	0.727	0.668	0.691	0.791	0.803	0.819	0.837	0.820	0.846
0.793	0.779	0.778	0.727	0.703	0.685	0.810	0.805	0.831	0.834	0.851	0.851

EGFP_sulim (Counts)
94

67809	60025	102745	99979	108175	109575	76531	72137	154549	128498	151693	130526
42654	33957	104464	103331	115580	115359	72935	68912	118882	117575	120961	118888
15117	11422	97913	93222	112280	107634	62202	49677	111322	110489	114973	109902
5881	5325	67768	54317	70586	53319	19434	20773	84612	77549	85990	72300
5071	5151	31184	27357	22038	20188	9510	11416	38419	41307	45109	46451
5221	5133	29389	26134	20092	23702	9122	9580	30837	42795	50058	53168

What if..

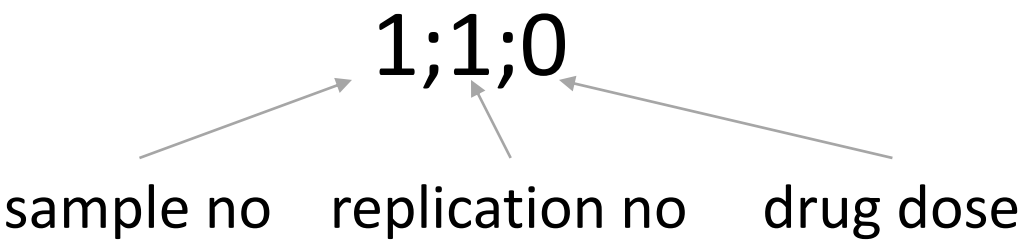


Programming) Generalized data read

- Experiment design
  - 2 types of samples (cells)
  - 2 replicates
  - 4 points of drug doses
  - Use a 96 well plate
- The results are stored in “Rprog04-fl.xls”

- Code writing
  - Make a function that read two excel files, one is for design and the other is for the data

Experiment Design



	1	2	3	4	5	6	7	8	9	10	11	12
A												
B	1;1;0	1;1;10	1;1;100	1;1;1000								
C	1;2;0	1;2;10	1;2;100	1;2;1000								
D	2;1;0	2;1;10	2;1;100	2;1;1000								
E	2;2;0	2;2;10	2;2;100	2;2;1000								
F												
G												
H												

Save the excel file as “exp_design.xlsx”

Top-down design in programming

1. Divide a big problem into smaller problems
2. keep dividing until the small problem can be solved easily
3. Solve the small problems
4. Merge the solutions to solve the big problem

Make a function that read two excel files, one is for design and the other is for the data

smaller problems

1. Read the excel files
2. Extract well positions using indexes
3. Extract the data using the well positions
4. Return the analysis ready data

Strategy

	1	2	3	4	5	6	7	8	9	10	11	12
A												
B	1;1;0	1;1;10	1;1;100	1;1;1000								
C	1;2;0	1;2;10	1;2;100	1;2;1000								
D	2;1;0	2;1;10	2;1;100	2;1;1000								
E	2;2;0	2;2;10	2;2;100	2;2;1000								
F												
G												
H												

Design file

Extract index

A01, A02, …, B01, …

Extract data

	A	B	C	D	E	F	G	H
	Plate	Repeat	Well	Type	Time	595nm_kk (A)	Time	EGFP_suli
	1	1	B01	M	00:00:15.93	0.701	00:01:11.48	67809
	1	1	B02	M	00:00:16.32	0.752	00:01:11.89	60025
	1	1	B03	M	00:00:16.69	0.723	00:01:12.30	102745
	1	1	B04	M	00:00:17.06	0.744	00:01:12.71	99979
	1	1	B05	M	00:00:17.43	0.706	00:01:13.12	108175
	1	1	B06	M	00:00:17.80	0.723	00:01:13.53	109575
	1	1	B07	M	00:00:18.17	0.767	00:01:13.94	76531
	1	1	B08	M	00:00:18.54	0.777	00:01:14.35	72137
	1	1	B09	M	00:00:18.91	0.762	00:01:14.76	154549
	1	1	B10	M	00:00:19.28	0.798	00:01:15.17	128498
	1	1	B11	M	00:00:19.65	0.793	00:01:15.58	151693
	1	1	B12	M	00:00:20.02	0.821	00:01:15.99	130526
	1	1	C01	M	00:00:22.85	0.803	00:01:18.86	42654
	1	1	C02	M	00:00:23.22	0.775	00:01:19.27	33957
	1	1	C03	M	00:00:23.59	0.780	00:01:19.68	104464
	1	1	C04	M	00:00:23.96	0.800	00:01:20.09	103331
	1	1	C05	M	00:00:24.33	0.758	00:01:20.50	115580

List ; Plates 1 - 1 | Plate_Page1 | Protocol | Errors | Notes

Data file

Read files

```
design_file_name <- "exp_design.xlsx"
data_file_name <- "Rprog04-fl.xls"
mydesign <- read_excel(design_file_name, sheet=1)
mydesign <- as.data.frame(read_excel(design_file_name, sheet=1, range="A1:L8", skip = 0,
col_names=F))
mydata <- as.data.frame(read_excel(data_file_name, sheet=1))

head(mydesign)
head(mydata)
```

```
> head(mydesign)
    ..1    ..2     ..3      ..4 ..5 ..6 ..7 ..8 ..9 ..10 ..11 ..12
1  <NA>   <NA>    <NA>     <NA>  NA  NA  NA  NA  NA    NA    NA    NA
2 1;1;0 1;1;10 1;1;100 1;1;1000  NA  NA  NA  NA  NA    NA    NA    NA
3 1;2;0 1;2;10 1;2;100 1;2;1000  NA  NA  NA  NA  NA    NA    NA    NA
4 2;1;0 2;1;10 2;1;100 2;1;1000  NA  NA  NA  NA  NA    NA    NA    NA
5 2;2;0 2;2;10 2;2;100 2;2;1000  NA  NA  NA  NA  NA    NA    NA    NA
6  <NA>   <NA>    <NA>     <NA>  NA  NA  NA  NA  NA    NA    NA    NA
> head(mydata)
  Plate Repeat Well Type             Time..5 595nm_kk (A)             Time..7 EGFP_sulim (Counts)
1     1      1  B01    M 1899-12-31 00:00:16    0.7012086 1899-12-31 00:01:12                67809
2     1      1  B02    M 1899-12-31 00:00:17    0.7518509 1899-12-31 00:01:12                60025
3     1      1  B03    M 1899-12-31 00:00:17    0.7232866 1899-12-31 00:01:13               102745
4     1      1  B04    M 1899-12-31 00:00:18    0.7440926 1899-12-31 00:01:13                99979
5     1      1  B05    M 1899-12-31 00:00:18    0.7056004 1899-12-31 00:01:14               108175
6     1      1  B06    M 1899-12-31 00:00:18    0.7228400 1899-12-31 00:01:14               109575
```

2. Extract positions using indexes

```
# make a position matrix
pos1 <- rep(LETTERS[1:8], time=12)
pos2 <- rep(sprintf("%02d", 1:12), each=8)
well_position_labels <- paste(pos1, pos2, sep="")
well_position_matrix <- matrix(well_position_labels, nrow=8, ncol=12)
```

Vector level

```
tmpi <- mydesign[, 1]
tmpv <- well_position_matrix[, 1]
!is.na(tmpi)
which(!is.na(tmpi))
tmpi[!is.na(tmpi)]
tmpv[!is.na(tmpi)]
```

2. Extract positions using indexes

data.frame (list) level

```
extract_values <- function(x){
  flag <- which(!is.na(x))
  return(x[flag])
}
extract_values(tmpi)

extract_values2 <- function(x){
  flag <- which(!is.na(x[1:8]))
  return(x[9:16][flag])
}
extract_values2(c(tmpi, tmpv))
```

Apply – last lectures

mydata

	Sample1	Sample2	Sample3	Sample4	Sample5	Sample6	Sample7	Sample8	Sample9	Sample10	Sample11	Sample12
Rep1	0.7738588	0.8049214	0.7846458	0.7871608	0.7393147	0.7132604	0.8267264	0.8352386	0.8397562	0.8459013	0.8631678	0.8699542
Rep2	0.7607952	0.7582134	0.7259247	0.7272937	0.6677032	0.6911640	0.7911676	0.8031119	0.8193607	0.8374564	0.8198136	0.8460365
Rep3	0.7925900	0.7791847	0.7780503	0.7274179	0.7033402	0.6846401	0.8104981	0.8053088	0.8314057	0.8338089	0.8511754	0.8506644

```
apply(mydata, 1, mean)
apply(mydata, 2, mean)
```

1: row (가로)
2: column (세로)

function

```
mysd <- function(x){
  xmean <- sum(x)/length(x)
  tmpdiff <- x-xmean
  xvar <- sum(tmpdiff^2)/(length(x)-1)
  xsd <- sqrt(xvar)
  return(xsd)
}
```

```
apply(mydata, 2, mysd)
```

2. Extract positions using indexes

data.frame (list) level

```
tmpdata <- rbind(mydesign, well_position_matrix)

colnames(mydesign) <- as.character(1:12)
colnames(well_position_matrix) <- as.character(1:12)
tmpdata <- rbind(mydesign, well_position_matrix)

tmpv <- lapply(tmpdata, extract_values2)
well_names <- unlist(tmpv)

tmpv <- lapply(mydesign, extract_values)
well_conditions <- unlist(tmpv)

well_info <- data.frame(well_names, well_conditions, stringsAsFactors = F)
well_info
```

3. Extract values using well names

A	B	C	D	E	F	G	H
Plate	Repeat	Well	Type	Time	595nm_kk (A)	Time	EGFP_suli
1	1	B01	M	00:00:15.93	0.701	00:01:11.48	67809
1	1	B02	M	00:00:16.32	0.752	00:01:11.89	60025
1	1	B03	M	00:00:16.69	0.723	00:01:12.30	102745
1	1	B04	M	00:00:17.06	0.744	00:01:12.71	99979
1	1	B05	M	00:00:17.43	0.706	00:01:13.12	108175
1	1	B06	M	00:00:17.80	0.723	00:01:13.53	109575
1	1	B07	M	00:00:18.17	0.767	00:01:13.94	76531
1	1	B08	M	00:00:18.54	0.777	00:01:14.35	72137
1	1	B09	M	00:00:18.91	0.762	00:01:14.76	154549
1	1	B10	M	00:00:19.28	0.798	00:01:15.17	128498
1	1	B11	M	00:00:19.65	0.793	00:01:15.58	151693
1	1	B12	M	00:00:20.02	0.821	00:01:15.99	130526
1	1	C01	M	00:00:22.85	0.803	00:01:18.86	42654
1	1	C02	M	00:00:23.22	0.775	00:01:19.27	33957
1	1	C03	M	00:00:23.59	0.780	00:01:19.68	104464
1	1	C04	M	00:00:23.96	0.800	00:01:20.09	103331
1	1	C05	M	00:00:24.33	0.758	00:01:20.50	115580

List ; Plates 1 - 1 | Plate_Page1 | Protocol | Errors | Notes

```
> mydata[1:10,]
   Plate Repeat Well Type              Time..5 595nm_kk (A)              Time..7 EGFP_sulim (Counts)
1      1      1  B01    M 1899-12-31 00:00:16    0.7012086 1899-12-31 00:01:12                67809
2      1      1  B02    M 1899-12-31 00:00:17    0.7518509 1899-12-31 00:01:12                60025
3      1      1  B03    M 1899-12-31 00:00:17    0.7232866 1899-12-31 00:01:13               102745
4      1      1  B04    M 1899-12-31 00:00:18    0.7440926 1899-12-31 00:01:13                99979
5      1      1  B05    M 1899-12-31 00:00:18    0.7056004 1899-12-31 00:01:14               108175
6      1      1  B06    M 1899-12-31 00:00:18    0.7228400 1899-12-31 00:01:14               109575
7      1      1  B07    M 1899-12-31 00:00:19    0.7668599 1899-12-31 00:01:14                76531
8      1      1  B08    M 1899-12-31 00:00:19    0.7765085 1899-12-31 00:01:15                72137
9      1      1  B09    M 1899-12-31 00:00:19    0.7624659 1899-12-31 00:01:15               154549
10     1      1  B10    M 1899-12-31 00:00:20    0.7980559 1899-12-31 00:01:16               128498
```

3. Extract values using well names

match
subset
merge

```
mydata[1:10,]

dim(mydata)
match(mydata$Well, well_info$well_names)
tmpidx <- match(mydata$Well, well_info$well_names)
mydata_subset <- subset(mydata, !is.na(tmpidx))[,c(3,6,8)]

final_data <- merge(well_info, mydata_subset, by.x="well_names", by.y="Well")
```

	well_names	well_conditions	595nm_kk (A)	EGFP_sulim (Counts)
1	B01	1;1;0	0.7012086	67809
2	B02	1;1;10	0.7518509	60025
3	B03	1;1;100	0.7232866	102745
4	B04	1;1;1000	0.7440926	99979
5	C01	1;2;0	0.8026616	42654
6	C02	1;2;10	0.7750938	33957

Parse the conditions

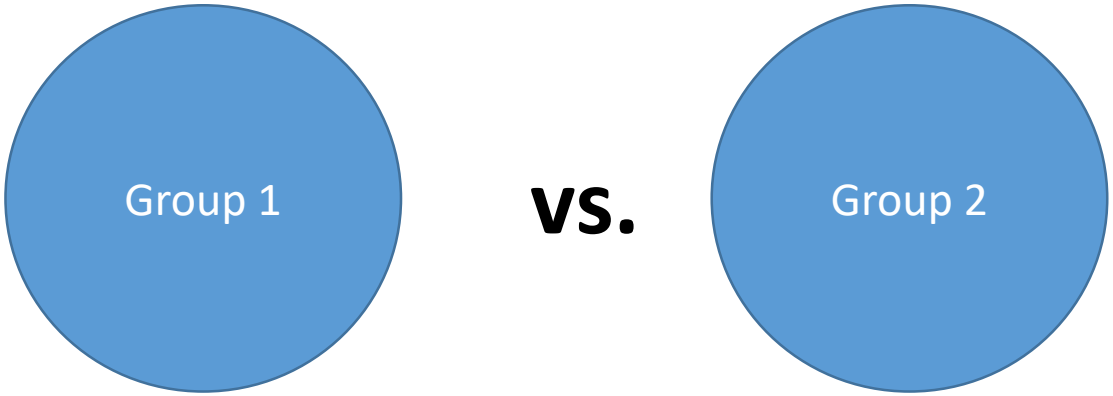
```
strsplit("1;1;0", ";")
unlist(strsplit("1;1;0", ";"))
strsplit(final_data$well_conditions, ";")
myparse <- function(x){
  tmp <- unlist(strsplit(x, ";"))
  names(tmp) <- c("sample_names", "replication", "concentration")
  return(tmp)
}

tmpcond <- sapply(final_data$well_conditions, myparse)
t_tmpcond <- t(sapply(final_data$well_conditions, myparse))
t_tmpcond2 <- cbind(t_tmpcond, rownames(t_tmpcond))
t_tmpcond2 <- cbind(t_tmpcond, well_conditions=rownames(t_tmpcond))
final_data <- merge(final_data, t_tmpcond2, by="well_conditions")
final_data <- final_data[,-1]
```

Let’s make all into the function

- Do it yourself

Quiz 4-1) Data comparison

- Which object or data structure will you use to compare two groups of datasets?
- How many variables do we need for the comparison in this example?



Data structure for data analysis

x1	y1
1	1
2	6
5	7
7	8

=

variable	value
x1	1
x1	2
x1	5
x1	7
y1	1
y1	6
y1	7
y1	8

Formula

```
a <- y~x
class(a)
```

Next

- R visualization
  - ggplot2
- Data manipulation
  - dplyr