R 프로그래밍 #6

2019. 04. 10

한국생명공학연구원
김하성

In today's lecture

- To know how to build and optimize a function (No worries if you don't understand all, it's enough to understand why this function is necessary and how to use it. This is why we learn programming!)
- To obtain (rather big) data using the function
- To understand ggplot
- To plot the data

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

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

condition A →

595nm_kk (A)
0.000

0.701	0.752	0.723	0.744	0.706			0.777	0.762			
0.803	0.775	0.780	0.800	0.758			0.787	0.808			
0.781	0.799	0.792	0.782	0.758			0.788	0.816			
0.774	0.805	0.785	0.787	0.739			0.835	0.840			
0.761	0.758	0.726	0.727	0.668							

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

A function for reading excel files

- **Name: multiple_plate_excel_reader**
- **input parameter: two (a vector of length two)**

```
design_file_name <- "exp_design.xlsx"
data_file_name <- "Rprog04-fl.xls“

multiple_plate_excel_reader <- function(design_file_name, data_file_name){
###
### code
###
}

multiple_plate_excel_read(file.names)
```

Code download & review

```
multiple_plate_excel_reader <- function(design_file_name, data_file_name){
  require(readxl)

  ## read excel files
  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))

  ## read data in all wells
  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)

  ## extract positions from position matrix by using design matrix indexes
  colnames(mydesign) <- as.character(1:12)
  colnames(well_position_matrix) <- as.character(1:12)
  bound_matrix <- rbind(mydesign, well_position_matrix)
  tmpv <- lapply(bound_matrix, extract_values_from_other_matrix)
  well_names <- unlist(tmpv)

  ## extract conditions from design matrix by design matrix indexes
  tmpv <- lapply(mydesign, extract_values_from_own_matrix)
  well_conditions <- unlist(tmpv)

  ## build well_info matrix
  well_info <- data.frame(well_names, well_conditions, stringsAsFactors = F)

  ## subset of the data filtered by the wells that we are interested in
  tmpidx <- match(mydata$Well, well_info$well_names)
  tmp_mydata_subset <- subset(mydata, !is.na(tmpidx))

  ## extract OD, GFP, etc if there is any
  sel_column <- c(3, seq(6, ncol(tmp_mydata_subset), by=2))
  mydata_subset <- tmp_mydata_subset[,sel_column]

  ## to make the condition column more readable
  tmp_final_data <- merge(well_info, mydata_subset, by.x="well_names", by.y="Well")
  tmpcond <- sapply(tmp_final_data$well_conditions, function(x){
      tmp <- unlist(strsplit(x, ";"))
      names(tmp) <- c("sample_names", "replication", "concentration")
      return(tmp)
    }
  )

  ## make final data matrix
  t_tmpcond <- t(tmpcond)
  t_tmpcond2 <- cbind(t_tmpcond, rownames(t_tmpcond))
  t_tmpcond2 <- cbind(t_tmpcond, well_conditions=rownames(t_tmpcond))
  final_data <- merge(tmp_final_data, t_tmpcond2, by="well_conditions")
  final_data <- final_data[,-1]

  return(final_data)
}
```

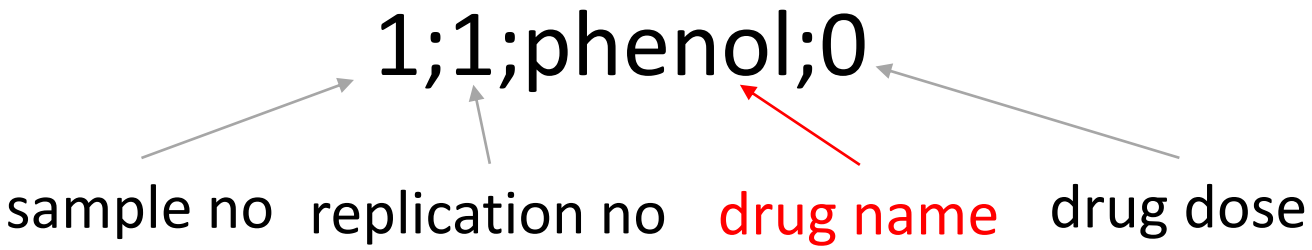
Usage

- Copy the function code to a new R file named “read_plate.R”
- Place design and data excel files in the same working directory (OR..)

```
source("read_plate.R")

design_file_name <- "exp_design.xlsx"
data_file_name <- "Rprog04-fl.xls"

mydata <- multiple_plate_excel_reader(design_file_name, data_file_name)
args(multiple_plate_excel_reader)
```

```
> head(mydata)
  well_names 595nm_kk (A) EGFP_sulim (Counts) sample_names replication concentration
1        B02    0.9241626                3882            1           1            10
2        B03    0.8793472               94601            1           1           100
3        B04    0.6583633               29836            1           1          1000
4        C02    0.9145258                3928            1           2            10
5        C03    0.8542024               95085            1           2           100
6        C04    0.6515582               41622            1           2          1000
```

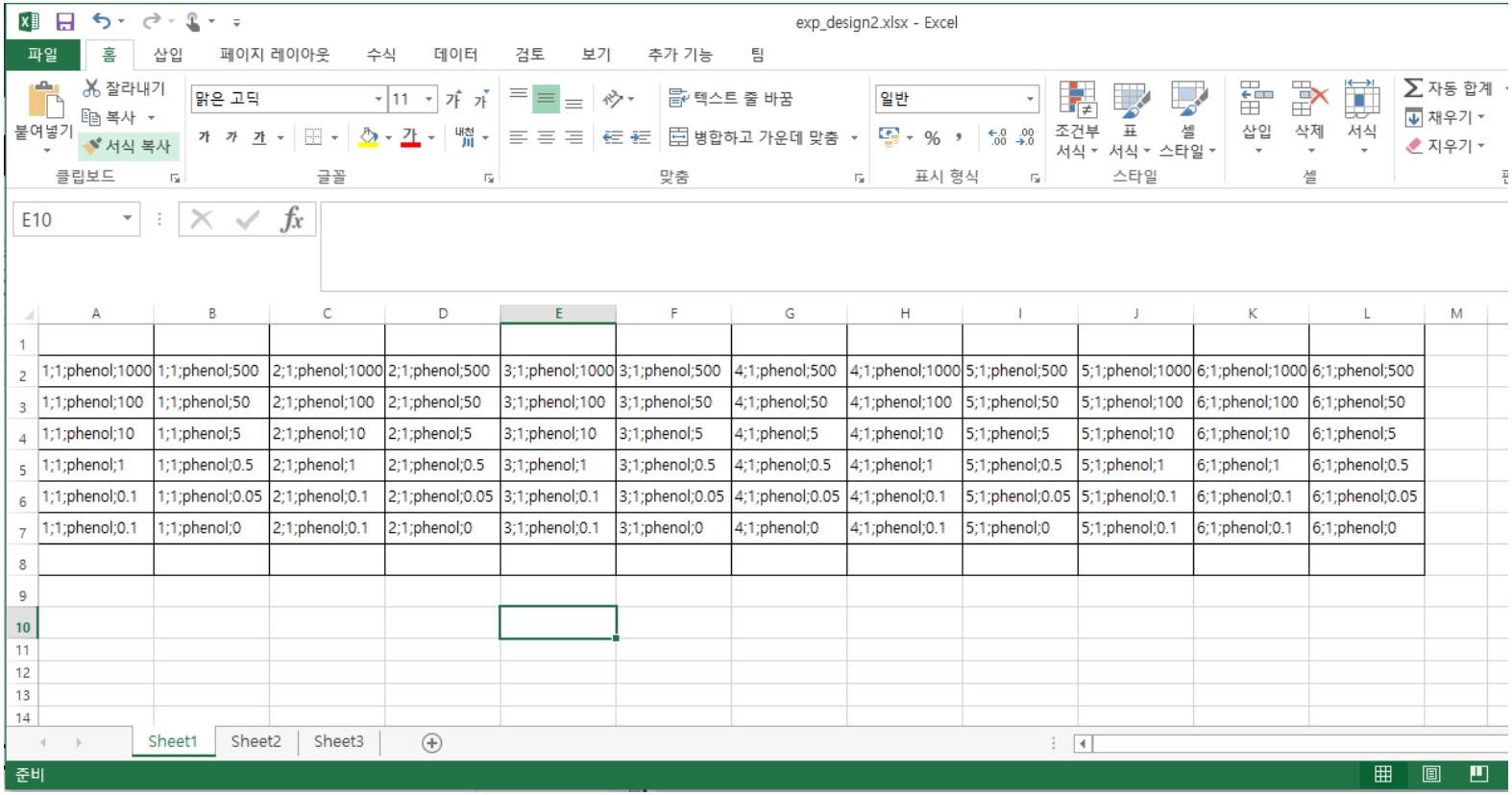
Experiment design notation rule



	1	2	3	4	5	6	7	8	9	10	11	12
A	1;1;phenol;1000	1;1;phenol;500	2;1;phenol;1000	2;1;phenol;500	3;1;phenol;1000	3;1;phenol;500	4;1;phenol;500	4;1;phenol;1000	5;1;phenol;500	5;1;phenol;1000	6;1;phenol;1000	6;1;phenol;500
B	1;1;phenol;100	1;1;phenol;50	2;1;phenol;100	2;1;phenol;50	3;1;phenol;100	3;1;phenol;50	4;1;phenol;50	4;1;phenol;100	5;1;phenol;50	5;1;phenol;100	6;1;phenol;100	6;1;phenol;50
C	1;1;phenol;10	1;1;phenol;5	2;1;phenol;10	2;1;phenol;5	3;1;phenol;10	3;1;phenol;5	4;1;phenol;5	4;1;phenol;10	5;1;phenol;5	5;1;phenol;10	6;1;phenol;10	6;1;phenol;5
D	1;1;phenol;1	1;1;phenol;0.5	2;1;phenol;1	2;1;phenol;0.5	3;1;phenol;1	3;1;phenol;0.5	4;1;phenol;0.5	4;1;phenol;1	5;1;phenol;0.5	5;1;phenol;1	6;1;phenol;1	6;1;phenol;0.5
E	1;1;phenol;0.1	1;1;phenol;0.05	2;1;phenol;0.1	2;1;phenol;0.05	3;1;phenol;0.1	3;1;phenol;0.05	4;1;phenol;0.05	4;1;phenol;0.1	5;1;phenol;0.05	5;1;phenol;0.1	6;1;phenol;0.1	6;1;phenol;0.05
F	1;1;phenol;0.1	1;1;phenol;0	2;1;phenol;0.1	2;1;phenol;0	3;1;phenol;0.1	3;1;phenol;0	4;1;phenol;0	4;1;phenol;0.1	5;1;phenol;0	5;1;phenol;0.1	6;1;phenol;0.1	6;1;phenol;0
G												
H												

Update the function

1. Add one more condition
2. Design file sheet



	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	1;1;phenol;1000	1;1;phenol;500	2;1;phenol;1000	2;1;phenol;500	3;1;phenol;1000	3;1;phenol;500	4;1;phenol;500	4;1;phenol;1000	5;1;phenol;500	5;1;phenol;1000	6;1;phenol;1000	6;1;phenol;500	
3	1;1;phenol;100	1;1;phenol;50	2;1;phenol;100	2;1;phenol;50	3;1;phenol;100	3;1;phenol;50	4;1;phenol;50	4;1;phenol;100	5;1;phenol;50	5;1;phenol;100	6;1;phenol;100	6;1;phenol;50	
4	1;1;phenol;10	1;1;phenol;5	2;1;phenol;10	2;1;phenol;5	3;1;phenol;10	3;1;phenol;5	4;1;phenol;5	4;1;phenol;10	5;1;phenol;5	5;1;phenol;10	6;1;phenol;10	6;1;phenol;5	
5	1;1;phenol;1	1;1;phenol;0.5	2;1;phenol;1	2;1;phenol;0.5	3;1;phenol;1	3;1;phenol;0.5	4;1;phenol;0.5	4;1;phenol;1	5;1;phenol;0.5	5;1;phenol;1	6;1;phenol;1	6;1;phenol;0.5	
6	1;1;phenol;0.1	1;1;phenol;0.05	2;1;phenol;0.1	2;1;phenol;0.05	3;1;phenol;0.1	3;1;phenol;0.05	4;1;phenol;0.05	4;1;phenol;0.1	5;1;phenol;0.05	5;1;phenol;0.1	6;1;phenol;0.1	6;1;phenol;0.05	
7	1;1;phenol;0.1	1;1;phenol;0	2;1;phenol;0.1	2;1;phenol;0	3;1;phenol;0.1	3;1;phenol;0	4;1;phenol;0	4;1;phenol;0.1	5;1;phenol;0	5;1;phenol;0.1	6;1;phenol;0.1	6;1;phenol;0	
8													
9													
10													
11													
12													
13													
14													

Update the function

3. Column names (OD, GFP, and possibly RFP…)

	A	B	C	D	E	F	G	H	I
1	Plate	Repeat	Well	Type	Time	595nm_kk (A)	Time	EGFP_sulim (Counts)	
2	1	1	B02	M	00:00:20.68	0.924	00:01:18.04	3882	
3	1	1	B03	M	00:00:21.05	0.879	00:01:18.45	94601	
4	1	1	B04	M	00:00:21.42	0.658	00:01:18.86	29836	
5	1	1	B05	M	00:00:21.79	0.895	00:01:19.26	7152	
6	1	1	B06	M	00:00:22.16	0.962	00:01:19.67	8285	
7	1	1	B07	M	00:00:22.53	0.919	00:01:20.08	4088	
8	1	1	B08	M	00:00:22.90	0.935	00:01:20.49	95802	
9	1	1	B09	M	00:00:23.27	0.709	00:01:20.90	31417	
10	1	1	B10	M	00:00:23.64	0.877	00:01:21.31	7100	
11	1	1	B11	M	00:00:24.01	0.895	00:01:21.72	7947	
12	1	1	C02	M	00:00:26.84	0.915	00:01:24.59	3928	
13	1	1	C03	M	00:00:27.21	0.854	00:01:25.00	95085	
14	1	1	C04	M	00:00:27.58	0.652	00:01:25.41	41622	
15	1	1	C05	M	00:00:27.95	0.884	00:01:25.82	6952	
16	1	1	C06	M	00:00:28.32	0.977	00:01:26.22	9096	
17	1	1	C07	M	00:00:28.71	0.897	00:01:26.63	3800	
18	1	1	C08	M	00:00:29.08	0.867	00:01:27.04	89744	
19	1	1	C09	M	00:00:29.45	0.635	00:01:27.45	31886	
20	1	1	C10	M	00:00:29.82	0.861	00:01:27.86	6410	
21	1	1	C11	M	00:00:30.19	0.870	00:01:28.27	8227	
22	1	1	D02	M	00:00:33.02	0.899	00:01:31.14	4280	
23	1	1	D03	M	00:00:33.39	0.866	00:01:31.55	173203	
24	1	1	D04	M	00:00:33.76	0.638	00:01:31.96	102397	
25	1	1	D05	M	00:00:34.13	0.866	00:01:32.37	7328	
26	1	1	D06	M	00:00:34.50	0.887	00:01:32.78	29236	
27	1	1	D07	M	00:00:34.87	0.901	00:01:33.19	4096	
28	1	1	D08	M	00:00:35.24	0.874	00:01:33.60	125130	
29	1	1	D09	M	00:00:35.61	0.642	00:01:34.01	76001	
30	1	1	D10	M	00:00:35.98	0.852	00:01:34.42	6780	
31	1	1	D11	M	00:00:36.35	0.875	00:01:34.83	15117	
32	1	1	E02	M	00:00:39.18	0.923	00:01:37.70	44070	

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

여기서 잠깐; String manipulation

```
s <- "This is the sixth lecture of R programming"
substr(s, 0, 11)
nchar(s)
toupper(s)
tolower(s)
strsplit(s, split=" ")
paste(s, " at UST", sep="")
sub("This", "That", s)
## regular expression
sub("This is.+of ", "", s)
grep("This", s)
grepl("This", s)
regexpr(“lecture", s)
```

Exercise 6-1) String functions

s <- "aatgctgtaga"

- Change the letters to capital and save it to ‘ss’
- Find start codon “ATG”
- Find stop codon “TAG”
- Extract the ORF

function update

1. Add one more condition
2. Design file sheet

```
## to make the condition column more readable
 tmp_final_data <- merge(well_info, mydata_subset, by.x="well_names", by.y="Well")
 tmpcond <- sapply(tmp_final_data$well_conditions, function(x){
     tmp <- unlist(strsplit(x, ";"))
     names(tmp) <- c("sample_names", "replication", "treatment", "concentration")
     return(tmp)
   }
 )
```

function update

3. Column names (OD, GFP, and possibly RFP…)

```
## column names
 cnames <- colnames(final_data)
 idx <- grep("595", cnames)
 if(length(idx)>0){
   cnames[idx] <- "OD"
 }
 idx <- grep("EGFP", cnames)
 if(length(idx)>0){
   cnames[idx] <- "GFP"
 }
 colnames(final_data) <- cnames
```

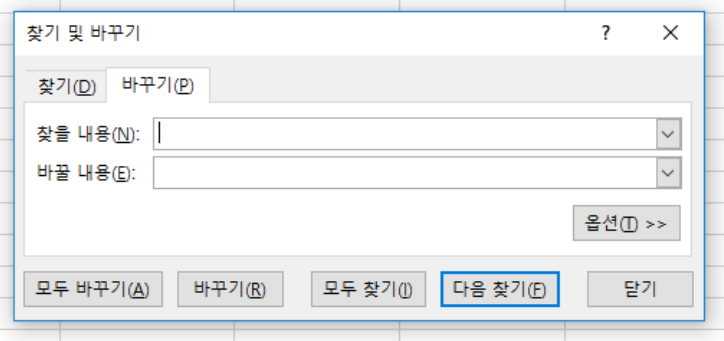
Read data

```
source("read_plate.R")

design_file_name <- "exp_design2.xlsx"
data_file_names <- c("20171012-phenol-1.xls",
                     "20171012-phenol-2.xls",
                     "20171227-phenol-1.xls",
                     "20171227-phenol-2.xls")

mydata1 <- multiple_plate_excel_reader2(design_file_name, data_file_names[1], sheet4design=1)
mydata2 <- multiple_plate_excel_reader2(design_file_name, data_file_names[2], sheet4design=2)
mydata3 <- multiple_plate_excel_reader2(design_file_name, data_file_names[3], sheet4design=3)
# mydata4 <- multiple_plate_excel_reader3(design_file_name, data_file_names[4], sheet4design=4)

mydata <- rbind(mydata1, mydata2, mydata3, mydata4)
```

```
> head(mydata)
  well_names        OD    GFP sample_names replication treatment concentration
1        G02 0.9042823 124002            1           1    phenol          0e+00
2        F02 0.9368631 127999            1           1    phenol          5e-02
3        E02 0.9228352  44070            1           1    phenol          5e-01
4        D02 0.8994368   4280            1           1    phenol          5e+00
5        C02 0.9145258   3928            1           1    phenol          5e+01
6        B02 0.9241626   3882            1           1    phenol          5e+02
> dim(mydata)
[1] 224   7
```

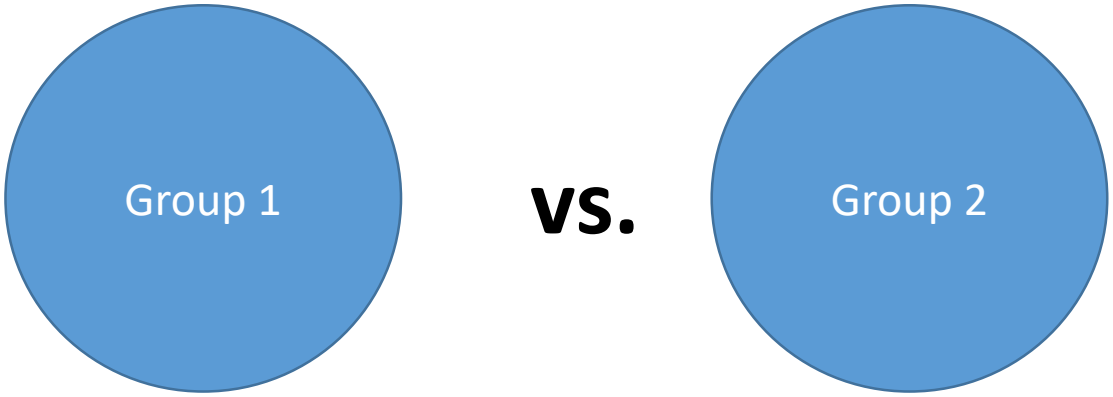
Experiment design

A	B	C	D	E	F	G	H	I	J	K	
1;3;phenol;1000	1;3;phenol;500	2;3;phenol;1000	2;3;phenol;500	3;3;phenol;1000	3;3;phenol;500	4;3;phenol;500	4;3;phenol;1000	5;3;phenol;500	5;3;phenol;1000	6;3;phenol;1000	6;3;p
1;3;phenol;100	1;3;phenol;50	2;3;phenol;100	2;3;phenol;50	3;3;phenol;100	3;3;phenol;50	4;3;phenol;50	4;3;phenol;100	5;3;phenol;50	5;3;phenol;100	6;3;phenol;100	6;3;p
1;3;phenol;10	1;3;phenol;5	2;3;phenol;10	2;3;phenol;5	3;3;phenol;10	3;3;phenol;5	4;3;phenol;5	4;3;phenol;10	5;3;phenol;5	5;3;phenol;10	6;3;phenol;10	6;3;p
1;3;phenol;1	1;3;phenol;0.5	2;3;phenol;1	2;3;phenol;0.5	3;3;phenol;1	3;3;phenol;0.5	4;3;phenol;0.5	4;3;phenol;1	5;3;phenol;0.5	5;3;phenol;1	6;3;phenol;1	6;3;p
1;3;phenol;0.1	1;3;phenol;0.05	2;3;phenol;0.1	2;3;phenol;0.05	3;3;phenol;0.1	3;3;phenol;0.05	4;3;phenol;0.05	4;3;phenol;0.1	5;3;phenol;0.05	5;3;phenol;0.1	6;3;phenol;0.1	6;3;p
1;3;phenol;0.1	1;3;phenol;0	2;3;phenol;0.1	2;3;phenol;0	3;3;phenol;0.1	3;3;phenol;0	4;3;phenol;0	4;3;phenol;0.1	5;3;phenol;0	5;3;phenol;0.1	6;3;phenol;0.1	6;3;p



Quiz 6-1) Data structure

- 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

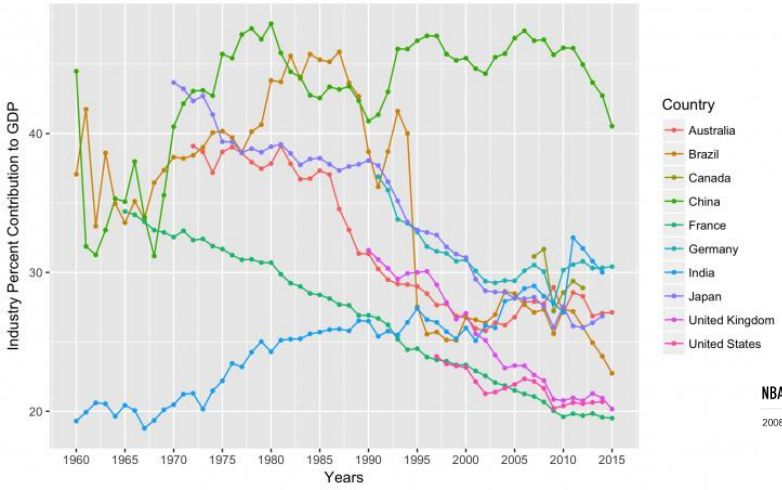
=

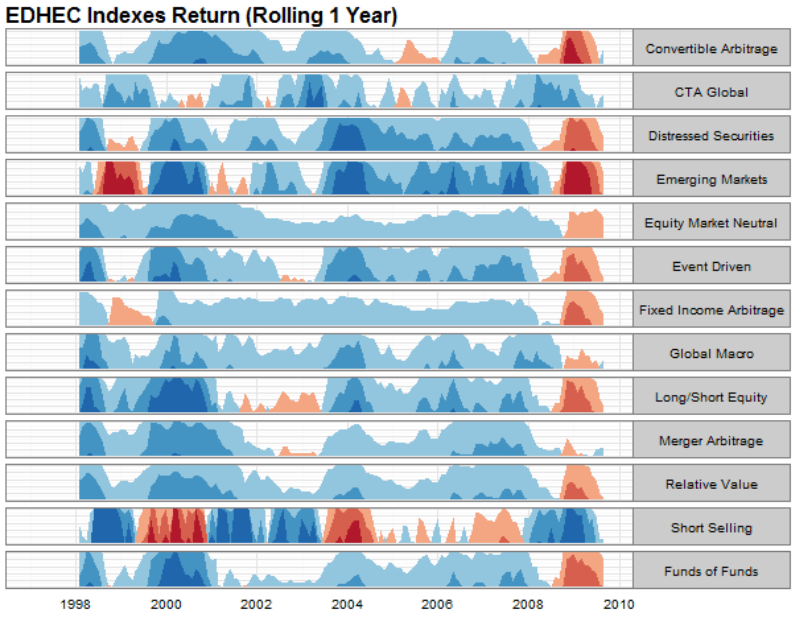
variable	value
x1	1
x1	2
x1	5
x1	7
y1	1
y1	6
y1	7
y1	8

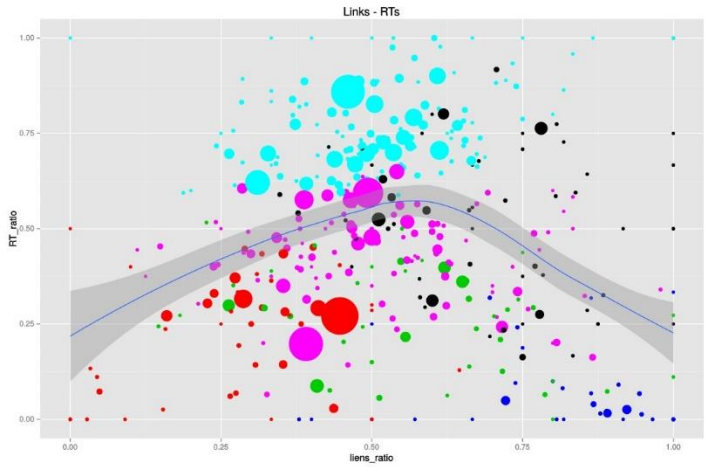
Formula

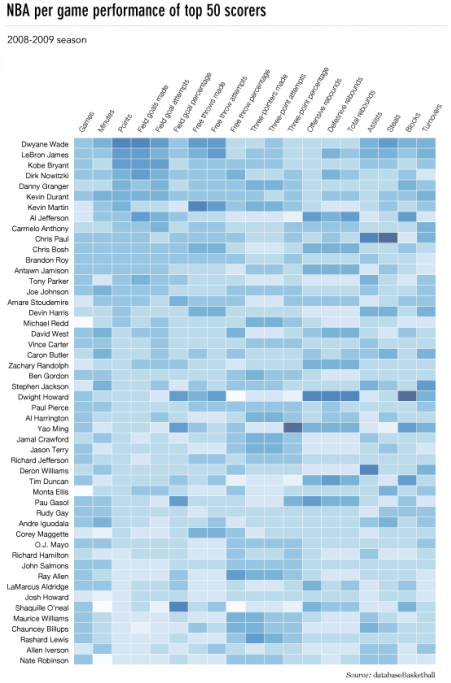
```
a <- y1~x1
class(a)
lm(y1~x1)
lm(y1~x1, data=xymlt)
```

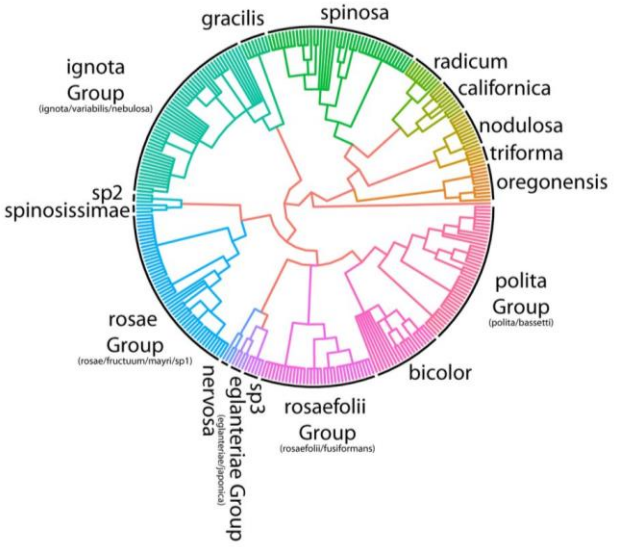
ggplot2


Industry Percent Contribution to GDP
40
30
20
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
Years
Country
Australia
Brazil
Canada
China
France
Germany
India
Japan
United Kingdom
United States


EDHEC Indexes Return (Rolling 1 Year)
Convertible Arbitrage
CTA Global
Distressed Securities
Emerging Markets
Equity Market Neutral
Event Driven
Fixed Income Arbitrage
Global Macro
Long/Short Equity
Merger Arbitrage
Relative Value
Short Selling
Funds of Funds
1998
2000
2002
2004
2006
2008
2010

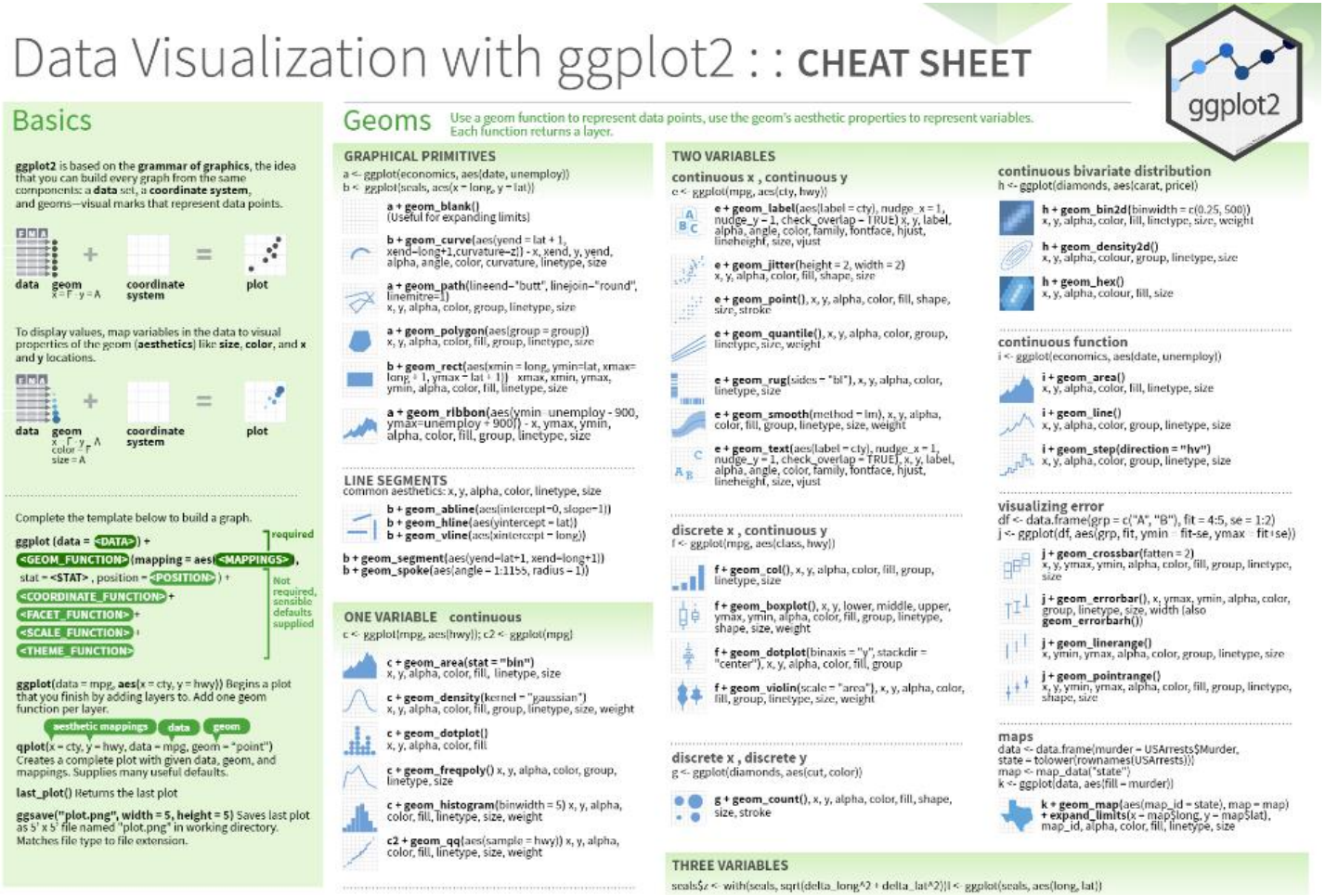

Links - RTs
RT_ratio
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
liens_ratio


NBA per game performance of top 50 scorers
2008-2009 season
Games
Minutes
Points
Field goals made
Field goal attempts
Field goal percentage
Free throws made
Free throw attempts
Free throw percentage
Three-points made
Three-point attempts
Three-point percentage
Offensive rebounds
Defensive rebounds
Total rebounds
Assists
Steals
Blocks
Turnovers
Dwyane Wade
LeBron James
Kobe Bryant
Dirk Nowitzki
Danny Granger
Kevin Durant
Kevin Martin
Al Jefferson
Carmelo Anthony
Chris Paul
Chris Bosh
Brandon Roy
Antawn Jamison
Tony Parker
Joe Johnson
Amare Stoudemire
Devin Harris
Michael Redd
David West
Vince Carter
Caron Butler
Zachary Randolph
Ben Gordon
Stephen Jackson
Dwight Howard
Paul Pierce
Al Harrington
Yao Ming
Jamal Crawford
Jason Terry
Richard Jefferson
Deron Williams
Tim Duncan
Monta Ellis
Pau Gasol
Rudy Gay
Andre Iguodala
Corey Maggette
O.J. Mayo
Richard Hamilton
John Salmons
Ray Allen
LaMarcus Aldridge
Josh Howard
Shaquille O'neal
Maurice Williams
Chauncey Billups
Rashard Lewis
Allen Iverson
Nate Robinson
Source: databasefbasketball


gracilis
spinosa
radicum
californica
nodulosa
triforma
oregonensis
polita
Group
(polita/bassetti)
bicolor
rosaefolii
Group
(rosaefolii/fusiformans)
sp3
eglanteriae Group
(eglanteriae/japonica)
nervosa
rosae
Group
(rosae/fructuum/mayri/sp1)
sp2
spinosissimae
ignota
Group
(ignota/variabilis/nebulosa)

ggplot2

https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf



Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points.

data + geom (x = F, y = A) = coordinate system + plot

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system + plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
  stat = <STAT>, position = <POSITION>) +                  Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

- a + geom_blank() (Useful for expanding limits)
- b + geom_curve(aes(yend = lat + 1, xend=long+1, curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size
- a + geom_path(lineend="butt", linejoin="round", linemitre=1) x, y, alpha, color, group, linetype, size
- a + geom_polygon(aes(group = group)) x, y, alpha, color, fill, group, linetype, size
- b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

- b + geom_abline(aes(intercept=0, slope=1))
- b + geom_hline(aes(yintercept = lat))
- b + geom_vline(aes(xintercept = long))
- b + geom_segment(aes(yend=lat+1, xend=long+1))
- b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

- c + geom_area(stat = "bin") x, y, alpha, color, fill, linetype, size
- c + geom_density(kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight
- c + geom_dotplot() x, y, alpha, color, fill
- c + geom_freqpoly() x, y, alpha, color, group, linetype, size
- c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight
- c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

- e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- e + geom_jitter(height = 2, width = 2) x, y, alpha, color, fill, shape, size
- e + geom_point(), x, y, alpha, color, fill, shape, size, stroke
- e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight
- e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size
- e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight
- e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

- f + geom_col(), x, y, alpha, color, fill, group, linetype, size
- f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group
- f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

- g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

- h + geom_bin2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight
- h + geom_density2d() x, y, alpha, colour, group, linetype, size
- h + geom_hex() x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

- i + geom_area() x, y, alpha, color, fill, linetype, size
- i + geom_line() x, y, alpha, color, group, linetype, size
- i + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

- j + geom_crossbar(fatten = 2) x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also geom_errorbarh())
- j + geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size
- j + geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

- k + geom_map(aes(map_id = state), map = map) + expand_limits(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))
```

ggplot grammer

- Components
  - data frame (ggplot)
  - aesthetic factors such as color, size, etc (aes)
  - geometric factors such as point, line, bar, etc (geoms)
  - statistical factors (stats)
  - theme or scale to be used in aes
- How to draw
  - Determine what graph do you want
  - Use ggplot to indicate dataset and its aesthetic factors
  - Add layers to indicate geometric factors and appropriate statistics
  - Add layers for scale/theme

barplot

```
> head(mydata)
  well_names        OD    GFP sample_names replication treatment concentration
1        G02 0.9042823 124002            1           1    phenol          0e+00
2        F02 0.9368631 127999            1           1    phenol          5e-02
3        E02 0.9228352  44070            1           1    phenol          5e-01
4        D02 0.8994368   4280            1           1    phenol          5e+00
5        C02 0.9145258   3928            1           1    phenol          5e+01
6        B02 0.9241626   3882            1           1    phenol          5e+02
```

```
library(ggplot2)
ggplot(data=mydata, aes(x=sample_names, y=GFP))

ggplot(data=mydata, aes(x=sample_names, y=GFP)) +
  geom_bar()

?geom_bar

ggplot(data=mydata, aes(x=sample_names, y=GFP)) +
  geom_bar(stat="identity")

ggplot(data=mydata, aes(x=sample_names, y=GFP, fill=concentration)) +
  geom_bar(stat="identity", position="dodge")
```

Next

- R visualization II
  - ggplot2
- Data manipulation
  - dplyr