R 프로그래밍 #9

2019. 05. 08

한국생명공학연구원
김하성

In the previous lecture

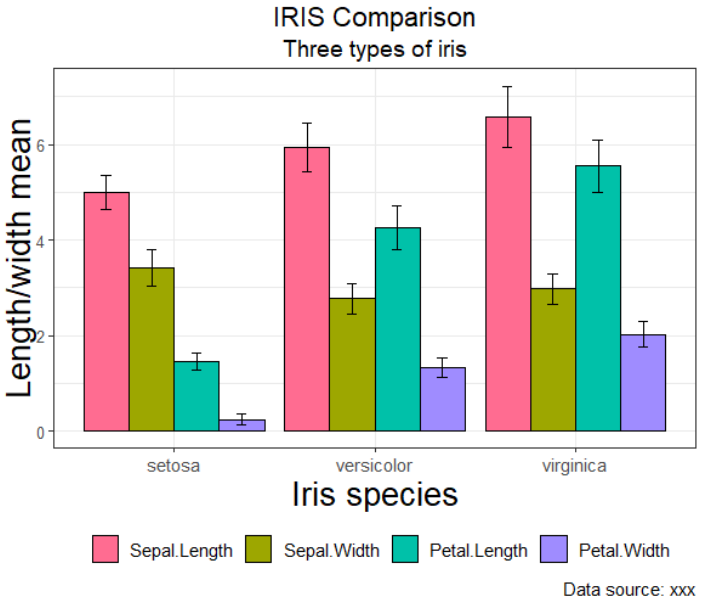
- Exercised how to manipulate and to visualize a dataset
  - ggplot2
    - geom_bar, geom_line
    - geom_errorbar
  - dplyr
    - %>%, group_by, summarize
    - mutate, select, join
  - reshape2

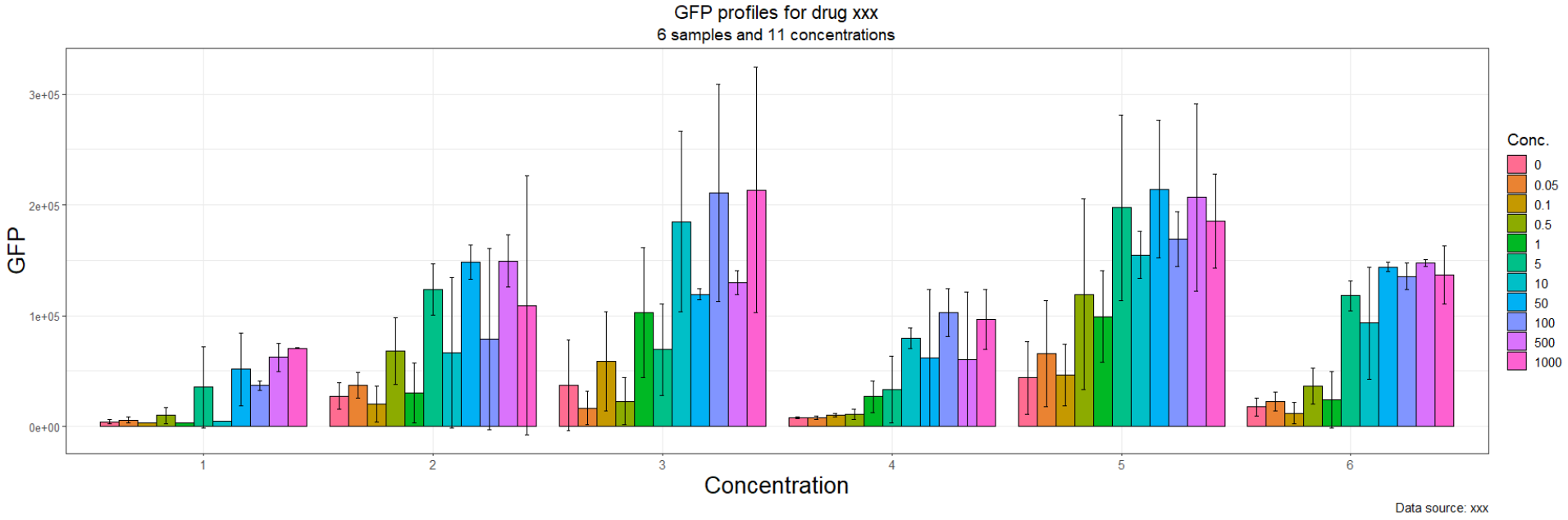
In today's lecture

- Exercised how to manipulate and to visualize a dataset
  - ggplot2
    - geom_bar, geom_line
    - geom_errorbar, scale, theme
    - 96well dataset
  - dplyr
    - %>%, group_by, summarize
    - mutate, select, join
  - reshape2

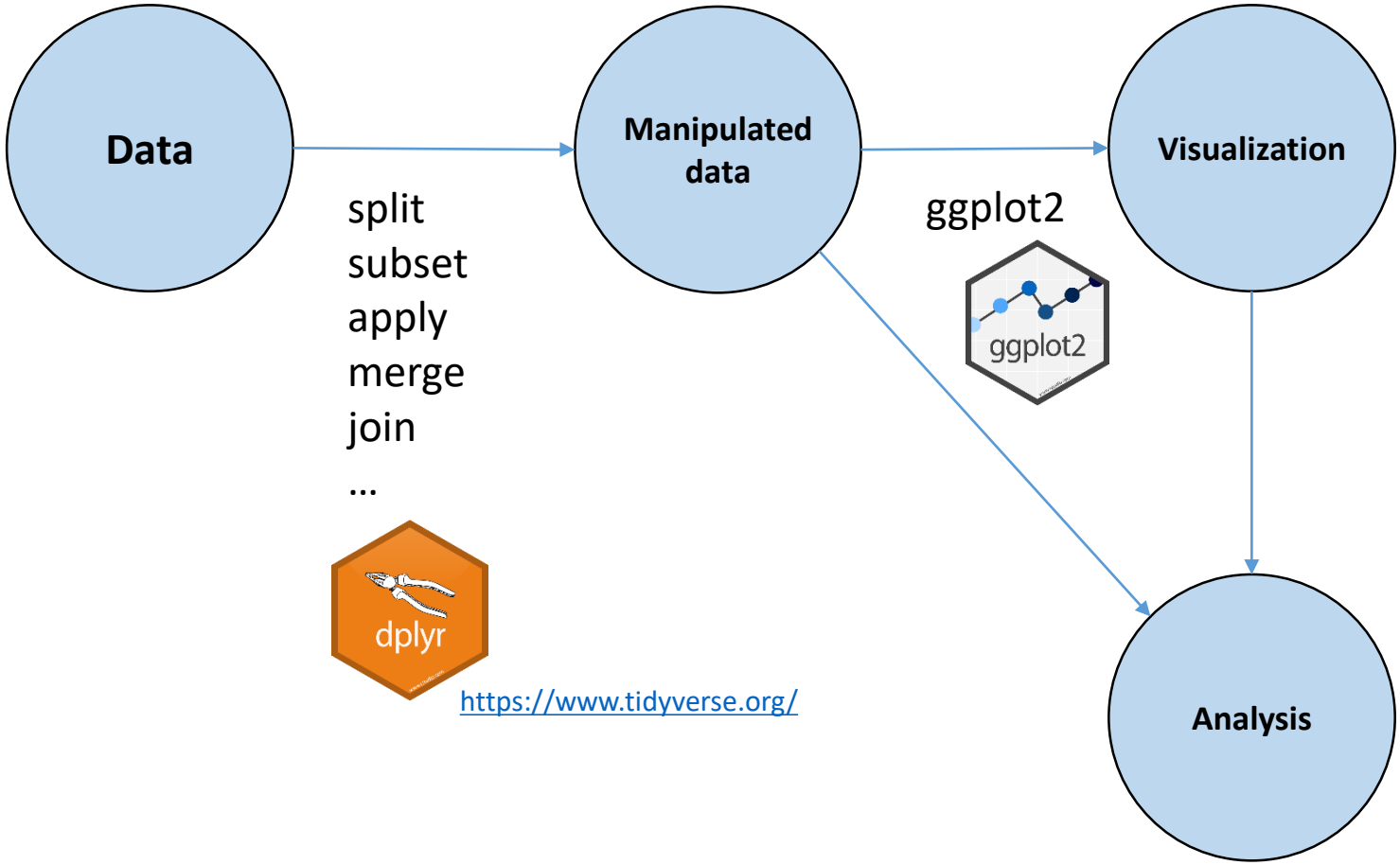
https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

Start at the end





Data analysis in R



Introducing dplyr

Hadley Wickham

2014-01-17

Categories: Packages

`dplyr` is a new package which provides a set of tools for efficiently manipulating datasets in R. `dplyr` is the next iteration of `plyr`, focussing on only data frames. `dplyr` is faster, has a more consistent API and should be easier to use. There are three key ideas that underlie `dplyr`:

1. Your time is important, so Romain Francois has written the key pieces in Rcpp to provide blazing fast performance. Performance will only get better over time, especially once we figure out the best way to make the most of multiple processors.

2. Tabular data is tabular data regardless of where it lives, so you should use the same functions to work with it. With `dplyr`, anything you can do to a local data frame you can also do to a remote database table. PostgreSQL, MySQL, SQLite and Google bigquery support is built-in; adding a new backend is a matter of implementing a handful of S3 methods.

3. The bottleneck in most data analyses is the time it takes for you to figure out what to do with your data, and dplyr makes this easier by having individual functions that correspond to the most common operations (`group_by`, `summarise`, `mutate`, `filter`, `select` and `arrange`). Each function does one only thing, but does it well.

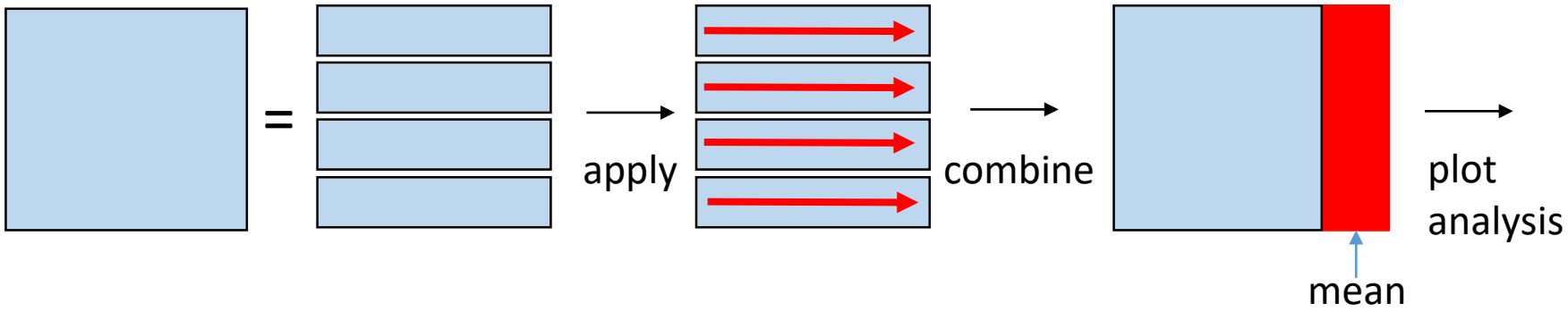
dplyr functions

dplyr Function	Description
select()	Selecting columns (variables)
filter()	Filter (subset) rows.
group_by()	Group the data
summarise()	Summarise (or aggregate) data
arrange()	Sort the data
join()	Joining data frames (tables)
mutate()	Creating New Variables

```
str(iris)
tmp <- select(iris, Sepal.Length)
head(tmp)

# to select variables
iris_sepal <- select(iris, Sepal.Length, Sepal.Width)
head(iris_sepal)

# to select a variable and divide into groups
iris_sepal <- select(iris, Sepal.Length, Sepal.Width, Species)
iris_group <- group_by(tmp, Species)
iris_group

# to get means
iris_mean <- summarize(iris_group,
                       mean(Sepal.Length),
                       mean(Sepal.Width))
iris_mean

# to get means for all columns
iris_mean <- summarize_all(iris_group, mean)
iris_mean
```

```
# to get standard deviations for all columns
iris_sd <- summarize_all(iris_group, sd)
iris_sd

# join iris_mean and iris_sd with the same species
iris_join <- inner_join(iris_mean, iris_sd, by="Species")
iris_join

mutate(iris_join, Sepal.Length.x+2)
colnames(iris_join) <- c("Specis",
                         "Sepal.Length.mean",
                         "Sepal.Width.mean",
                         "Sepal.Length.sd",
                         "Sepal.Width.sd")

# use pipe operator
iris_mean <- iris %>%
              select(Sepal.Length, Sepal.Width, Species) %>%
              group_by(Species) %>%
              summarize_all(mean)
```

The Pipe Operator: %>% (dplyr package)



- %>% takes the output of its lhs statement and makes it the input of the rhs (next) statement

$$f(x) == x \%>\% f$$

- Short cut in Rstudio: Shift + Ctl + m (Alt+_ for <-)
- Placeholder . operator

example 9-1) ggplot and dplyr

- Draw a bar graph of the iris_join$Sepal.length.mean
  - using plot function
  - using ggplot function
- Draw a bar graph of the all the mean variables in iris_join
  - ~~using plot function~~
  - ~~using ggplot function~~
- Dataset of mean and sd for all variables of iris

Transform data structure for ggplot

x	y
1	1
2	6
5	7
7	8

=

variable	value
x	1
x	2
x	5
x	7
y	1
y	6
y	7
y	8

x	y	z
1	1	3
2	6	4
5	7	5
7	8	6

=

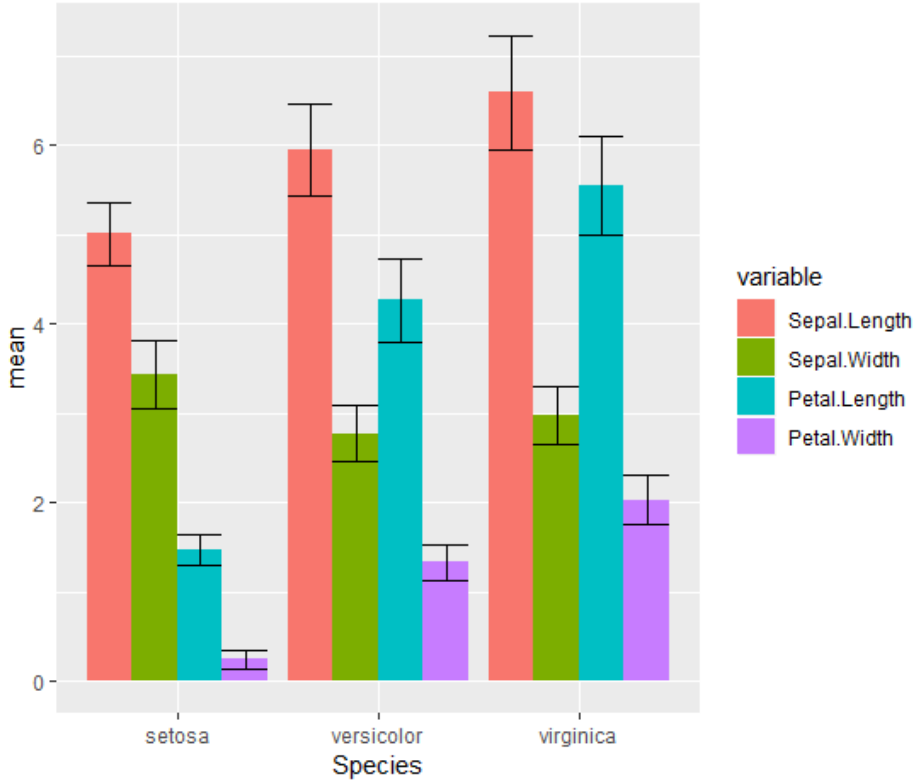
variable	value
x	1
x	2
x	5
x	7
y	1
y	6
y	7
y	8
z	3
z	4
z	5
z	6

Melt iris data

```
library(reshape2)
iris_melt <- melt(iris_mean)

iris_mean <- iris %>% group_by(Species) %>% summarize_all(mean) %>% melt
iris_sd <- iris %>% group_by(Species) %>% summarize_all(sd) %>% melt
iris_join <- inner_join(iris_mean, iris_sd, by=c("Species", "variable"))
colnames(iris_join)[c(3,4)] <- c("mean", "sd")


## change column name value → mean or sd
iris_mean <- iris %>% group_by(Species) %>% summarize_all(mean) %>% melt(value.name=c("mean"))
iris_sd <- iris %>% group_by(Species) %>% summarize_all(sd) %>% melt(value.name=c("sd"))
iris_join <- inner_join(iris_mean, iris_sd, by=c("Species", "variable"))
iris_join
```

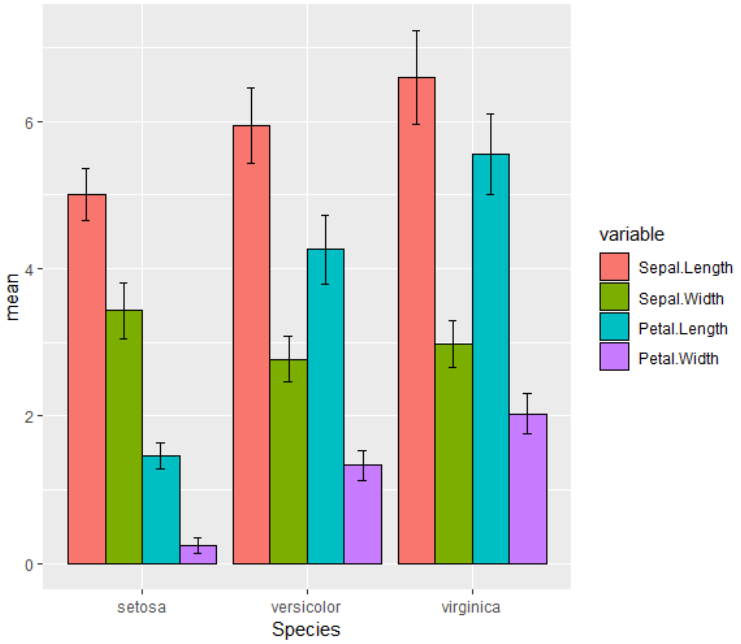
iris data bar graph

```
ggplot(iris_join, aes(x=Species, y=mean, fill=variable)) +
  geom_bar(stat="identity", position="dodge")

ggplot(iris_join, aes(x=Species, y=mean, fill=variable)) +
  geom_bar(stat="identity", position="dodge") +
  geom_errorbar(aes(min=mean-sd, max=mean+sd), position="dodge")
```



Draw error bars

```
p1 <- ggplot(iris_join, aes(x=Species, y=mean, fill=variable)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), width=.2, position=position_dodge(0.9))
p1
```

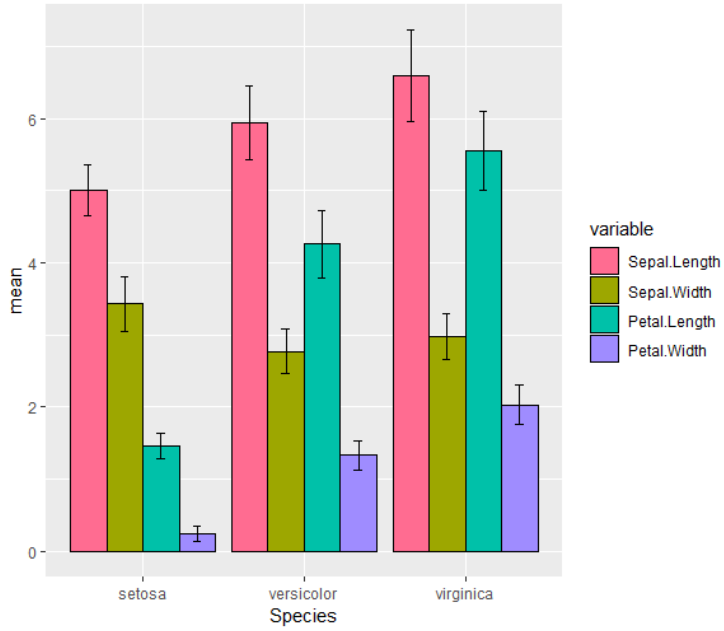


Scale

- aes mapping data to variable. Scale sets detail indications (axis, label, legend, …)
  - position
  - color and fill
  - size
  - shape
  - line type
- Scales are modified with a series of functions using a scale_<aesthetic>_<type> naming scheme. Try typing scale_<tab> to see a list of scale modification functions.
- Common scale arguments
  - name: the first argument gives the axis or legend title
  - limits: the minimum and maximum of the scale
  - breaks: the points along the scale where labels should appear
  - labels: the labels that appear at each break

Scale

```
p1 + scale_fill_brewer(palette = "Greens")

p1 + scale_fill_hue(h = c(0, 360))

p2 <- p1 + scale_fill_hue(h = c(0, 360)) +
  scale_y_continuous(name="Length/width Mean") +
  scale_x_discrete(name="Iris species")
p2

p2 <- p1 + scale_fill_hue(h = c(0, 360)) +
  ylab("Length/width Mean2") +
  xlab("Iris species2") +
  labs(title = "IRIS Comparison", subtitle="Three types of iris", caption="Data source: xxx", fill="Types")
p2

?scale_fill_hue
```



Theme

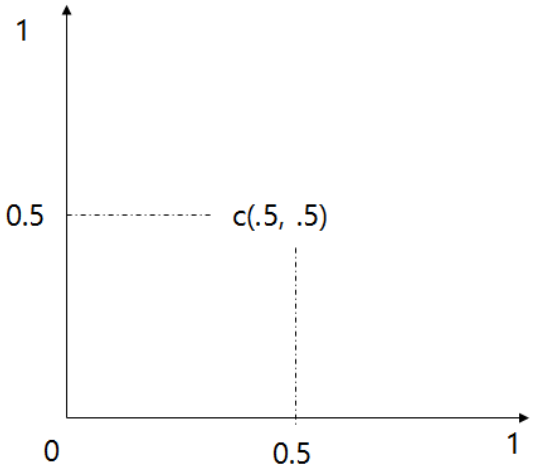
- Axis label
- Plot background
- Facet label backround
- Legend appearance

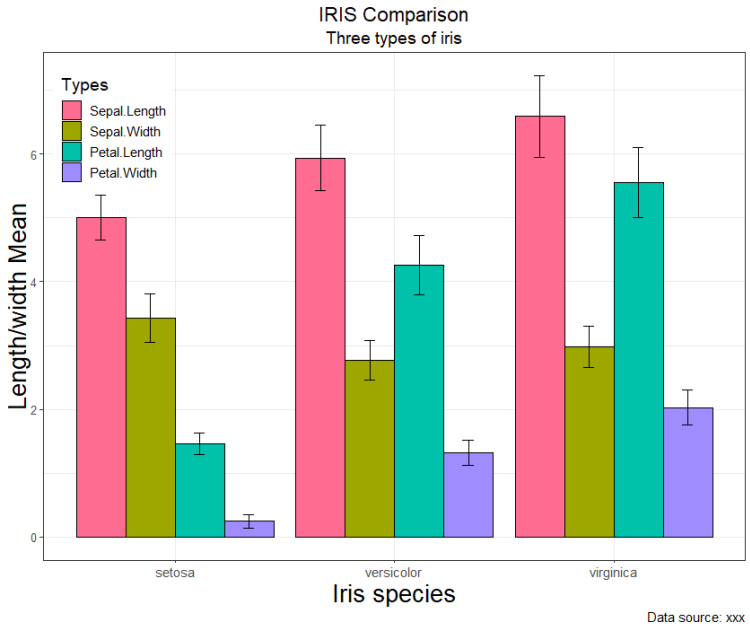
Usage

```
theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x,
  axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left,
  axis.title.y.right, axis.text, axis.text.x, axis.text.x.top,
  axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right,
  axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom,
  axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length,
  axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y,
  axis.line.y.left, axis.line.y.right, legend.background, legend.margin,
  legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,
  legend.key.size, legend.key.height, legend.key.width, legend.text,
  legend.text.align, legend.title, legend.title.align, legend.position,
  legend.direction, legend.justification, legend.box, legend.box.just,
  legend.box.margin, legend.box.background, legend.box.spacing,
  panel.background, panel.border, panel.spacing, panel.spacing.x,
  panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor,
  panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x,
  panel.grid.minor.y, panel.ontop, plot.background, plot.title,
  plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin,
  strip.background, strip.background.x, strip.background.y,
  strip.placement, strip.text, strip.text.x, strip.text.y,
  strip.switch.pad.grid, strip.switch.pad.wrap, ..., complete = FALSE,
  validate = TRUE)
```

Theme overriding

```
p2 + theme_bw() +
  theme(
    text=element_text(size=14),
    axis.text.y = element_text(size=10),
    axis.title.y = element_text(size=20),
    axis.title.x = element_text(size=20),
    #legend.position = "bottom",
    legend.position = c(0.1,0.9),
    plot.title=element_text(hjust=0.5),
    plot.subtitle=element_text(hjust=0.5)
    )
```





Dataset for exercise

Experiment conditions

- **Cell types: 1~4**
- **Drug type: 1 (phenol)**
- **Drug concentrations: 11 points**
- **Replications: 4 times**

Dataset

```
setwd("C:\\Rprog\\07")
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_reader2(design_file_name, data_file_names[4], sheet4design=4)

mydata <- rbind(mydata1, mydata2, mydata3, mydata4)
mydata2 <- mydata
mydata2$concentration <- as.factor(mydata2$concentration)
str(mydata2)
head(mydata2)
```

```
> head(mydata)
  well_names        OD    GFP sample_names replication drugname 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] 308   7
```

Dataset

Experiment conditions

Cell types: 1~6

Drug type: 1 (phenol)

Drug concentrations: 11 points

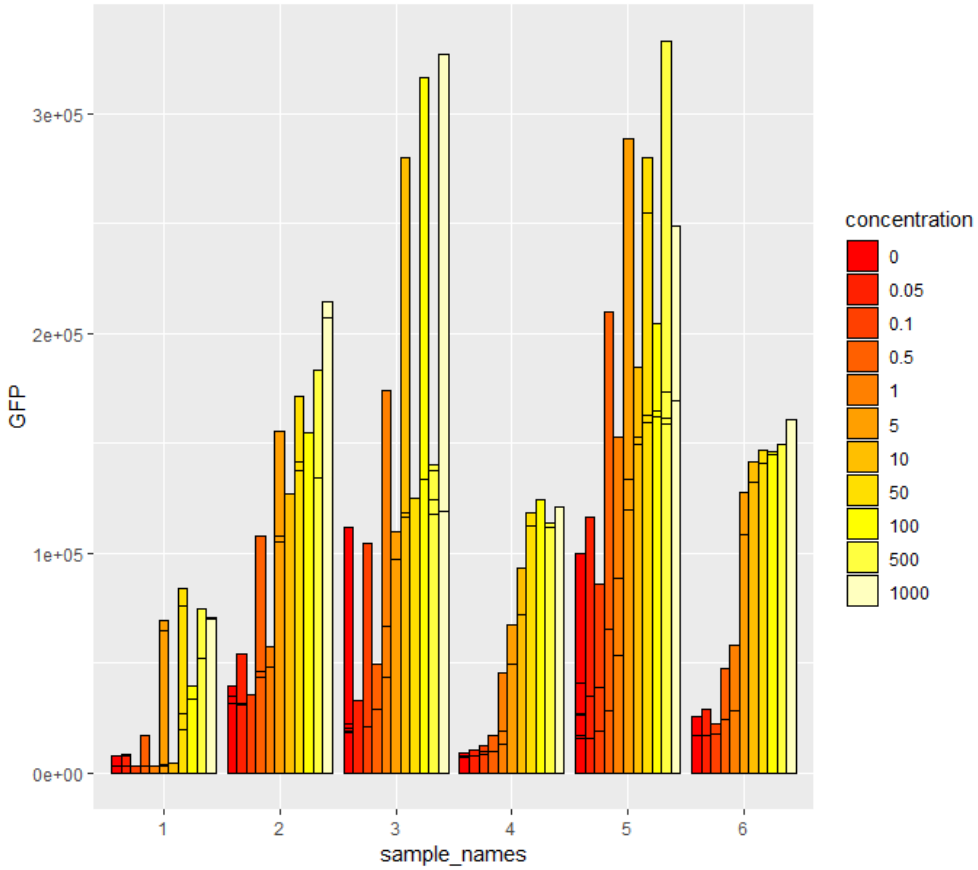
Replications: 4 times

```
> head(mydata)
  well_names        OD    GFP sample_names replication drugname 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] 308   7
```

```
> str(mydata2)
'data.frame':   308 obs. of  7 variables:
 $ well_names   : chr  "G02" "F02" "E02" "D02" ...
 $ OD           : num  0.904 0.937 0.923 0.899 0.915 ...
 $ GFP          : num  124002 127999 44070 4280 3928 ...
 $ sample_names : Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ replication  : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ drugname     : Factor w/ 1 level "phenol": 1 1 1 1 1 1 1 1 1 1 ...
 $ concentration: Factor w/ 11 levels "0","0.05","0.1",..: 1 2 4 6 8 10 1 2 3 3 ...
```

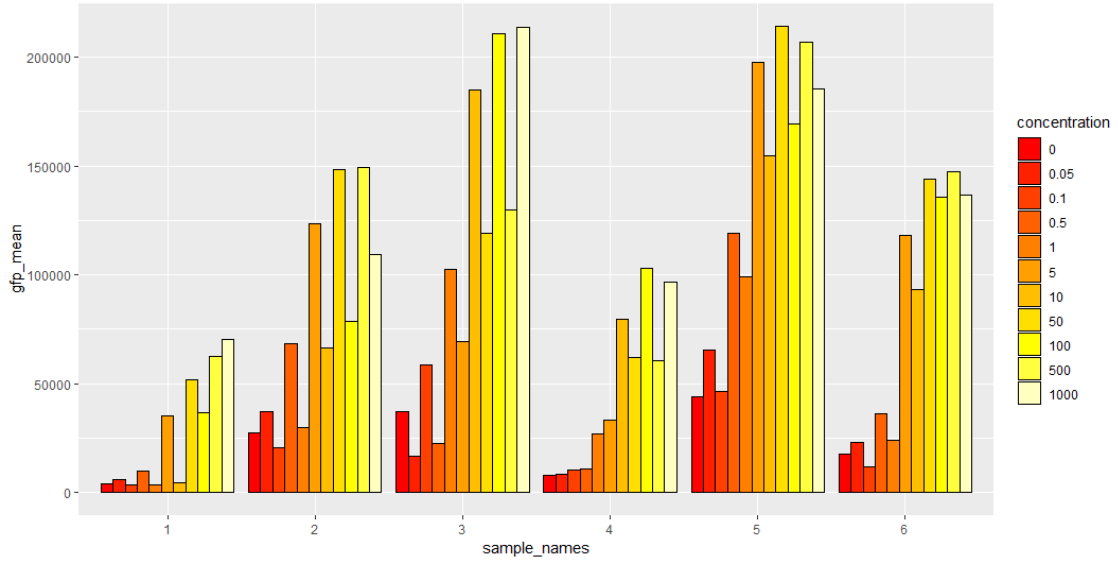
barplot - ggplot

```
ggplot(data=mydata2, aes(x=sample_names, y=GFP, fill=concentration)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  scale_fill_manual(values = heat.colors(11))
```



Plot gfp mean values

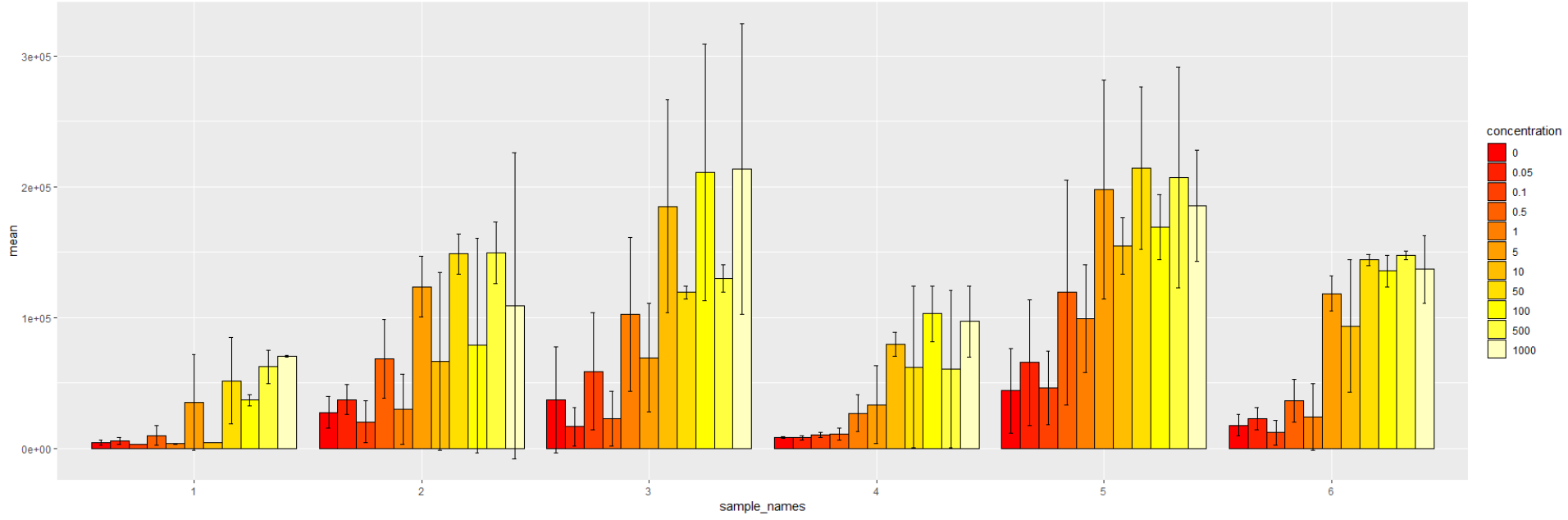
```
grouped_data <- group_by(mydata2, sample_names, drugname, concentration)
data_mean <- summarize(grouped_data, gfp_mean=mean(GFP))
ggplot(data_mean, aes(x=sample_names, y=gfp_mean, fill=concentration)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  scale_fill_manual(values = heat.colors(11))
```




```
save(mydata2, file="mydata2.Rdata")
load(mydata2)
```

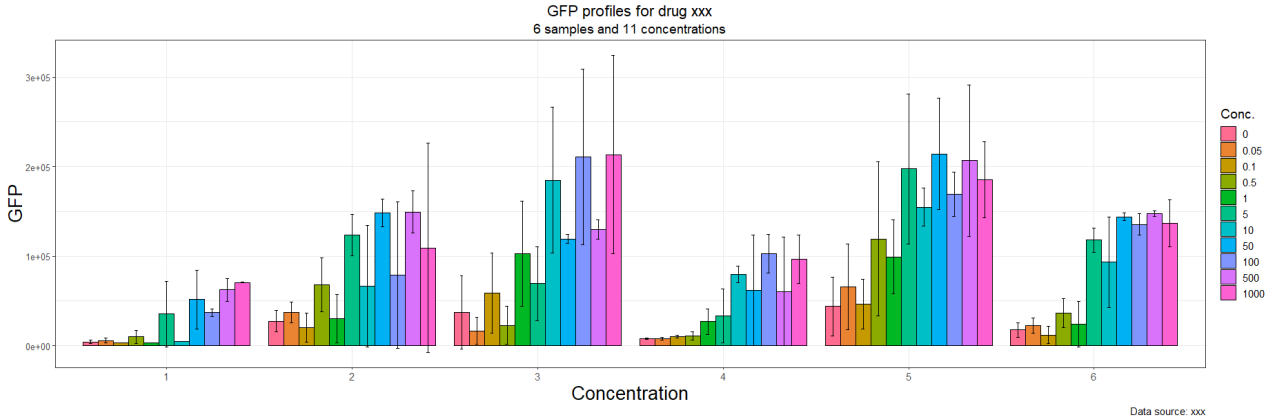
bar graph with error bars

```
grouped_data <- group_by(mydata2, sample_names, drugname, concentration)
data_mean <- summarize(grouped_data, mean=mean(GFP))
data_sd <- summarize(grouped_data, sd=sd(GFP))
data_join <- inner_join(data_mean, data_sd, by=c("sample_names", "drugname", "concentration"))

ggplot(data_join, aes(x=sample_names, y=mean, fill=concentration)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  scale_fill_hue(h = c(0, 360)) +
  geom_errorbar(aes(min=mean-sd, max=mean+sd), width=.2, position=position_dodge(0.9))
```



bar graph with error bars

```
p1 <- ggplot(data_join, aes(x=sample_names, y=mean, fill=concentration)) +
  geom_bar(stat="identity", position="dodge", color="black") +
  geom_errorbar(aes(min=mean-sd, max=mean+sd), width=.2, position=position_dodge(0.9))

p1 + scale_fill_hue(h = c(0, 360)) +
  ylab("GFP") +
  xlab("Concentration") +
  labs(title = "GFP profiles for drug xxx",
       subtitle="6 samples and 11 concentrations",
       caption="Data source: xxx", fill="Conc.") +
  theme_bw() +
  theme(
    text=element_text(size=14),
    axis.text.y = element_text(size=10),
    axis.title.y = element_text(size=20),
    axis.title.x = element_text(size=20),
    plot.title=element_text(hjust=0.5),
    plot.subtitle=element_text(hjust=0.5)
  )
```



Next

- Sequence analysis in R
- Install packages
  - shiny
  - Biostrings
  - rentrez