Data manipulation with R, Part 1

Sameh MAGDELDIN M.V.Sc, 2Ph.D

November, 2024

Data manipulation

Topics to be covered in this module

1. Data normalization
2. Data preparation
3. The apply family
4. Finding and removing duplicate hits

Required packages i will use in this tutorial

```
library(MASS)
library(ade4)
library(rgl)
library(e1071)
library(R2HTML)
library(cluster) # last 3 should be loaded before cluster
library(clusterSim) # all above should be loaded before clusterSim
library(reshape)
library(ggplot2)
```

Data Normalization

> Normalization is a term that is used to describe the process of eliminating such variations to allow appropriate comparison of data obtained from the two samples.
>
> Data normalization is very critical prior to statistical analysis. without it, a probable false statistical result might be what you get.

Note that normalization is a case by case. i mean it depends on experiments and stat. PCA and regression require data normalization.

I will demonstrate an example before working with R to normalize the data

{Data Normalization} Normalization based on intensity

When working with PCR ,for example, you assume that you are loading the same amount of mRNA (template). However, if starting template changed at beginning the final result are not comparable.

When you want to compare 2 images parts, you need to make sure that total intensity of are same but not specified parts only.

A simple example that Ahmed earned 50 points out of 100, Mohamed earned 500 out of 1000. you can not say Mohamed is better than Ahmed because he got 500 point while other got 50. you need to normalize the data.

Lets do the simple one.

When counting sub-cellular fraction by and b, these was the result.

```
a <- c(1,2,3,4,5)
b <- c(1,2,3,4,10)
data <- data.frame(a=a,b=b)
data
```

```
##   a  b
## 1 1  1
## 2 2  2
## 3 3  3
## 4 4  4
## 5 5 10
```

To compare between these fractions in both reads, we need to normalize the data

```
norm1 <- data.Normalization(data,type="n10",normalization="column")
norm1
```

```
##           a    b
## 1 0.06666667 0.05
## 2 0.13333333 0.10
## 3 0.20000000 0.15
## 4 0.26666667 0.20
## 5 0.33333333 0.50
```

The data was corrected based on the sum, (n10), type ?data.Normalization to see what normalization method you need.

```
norm1 * 20
```

```
##         a  b
## 1 1.333333  1
## 2 2.666667  2
## 3 4.000000  3
## 4 5.333333  4
## 5 6.666667 10
```

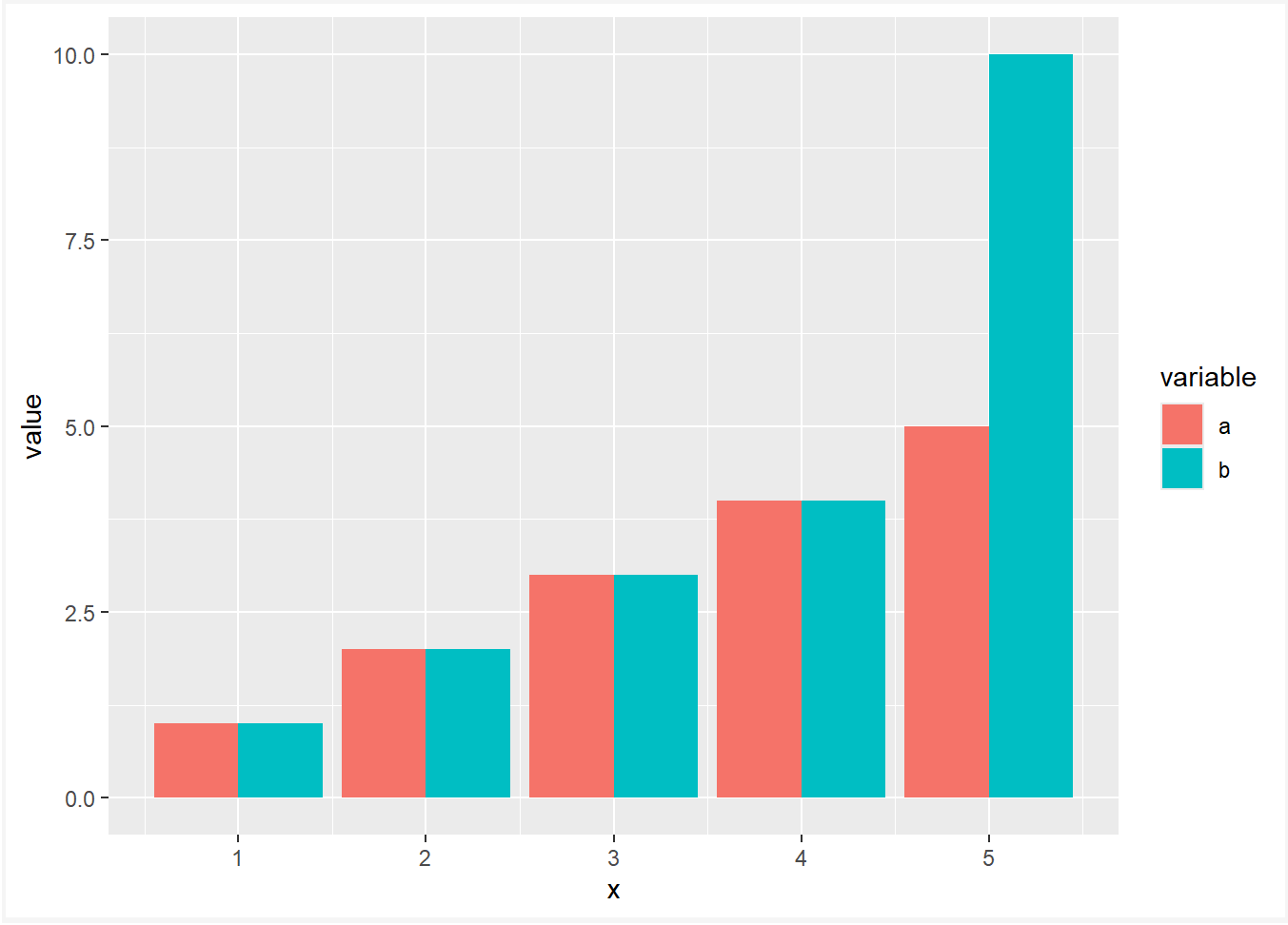
OK there is a big difference in interpreting both result, lets see the graphs for both of them

Graphing the original data

```
melted <- melt(data)
```

```
## Using  as id variables
```

```
melted$x <- c(1,2,3,4,5,1,2,3,4,5)
ggplot(melted, aes(x, value, fill=variable)) + geom_bar(stat="identity", position="dodge")
```

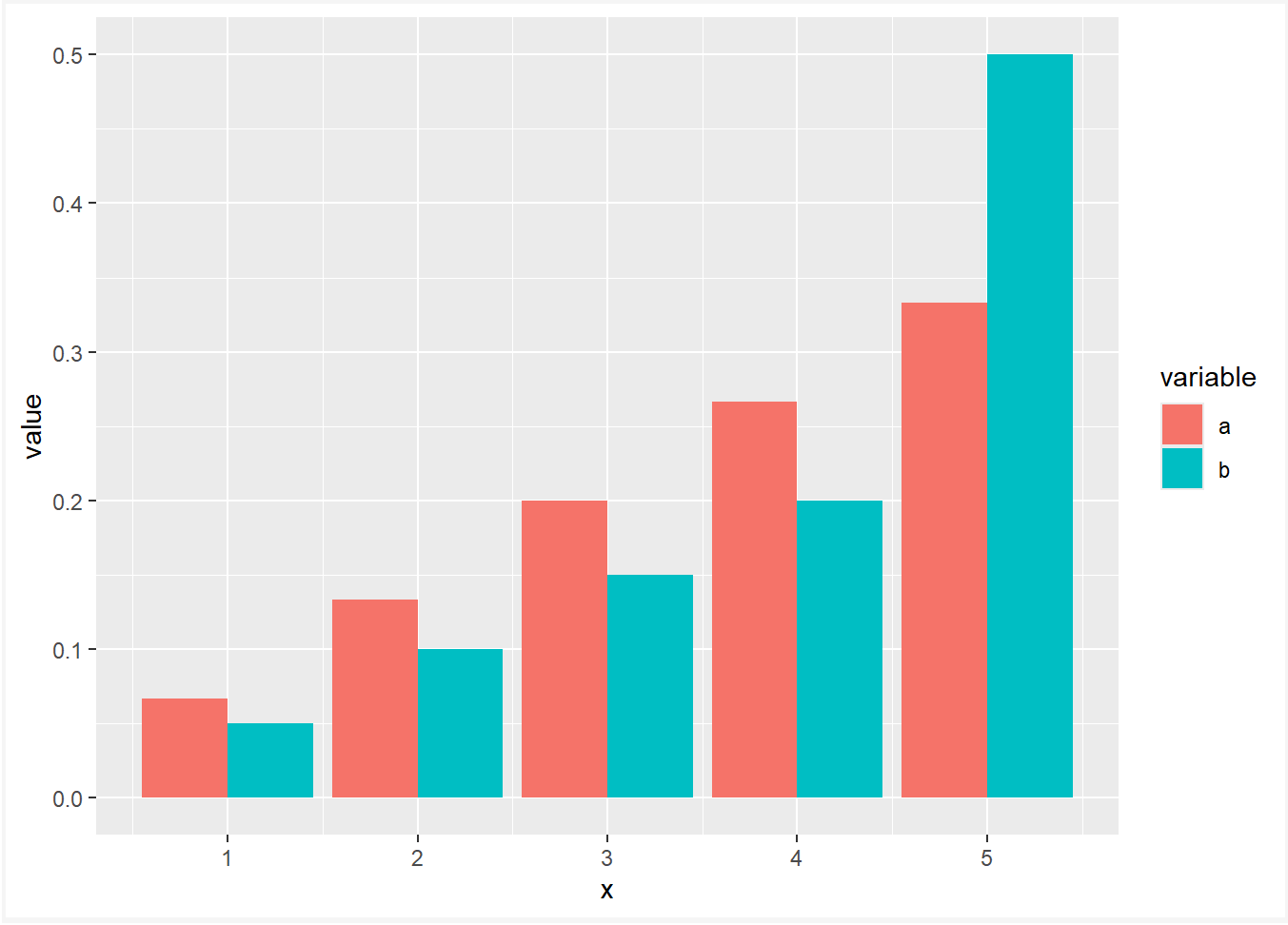


Graphing the normalized data

```
melted1 <- melt(norm1)

## Using  as id variables

melted1$x <- c(1,2,3,4,5,1,2,3,4,5)
ggplot(melted1, aes(x, value, fill=variable)) + geom_bar(stat="identity", position="dodge")
```



As you can see, using a built in function is classy:), always stay classy

my graph

{Data Normalization} Normalization based on mean and SD [Z score normalization]

This is another common method of normalization. where a simple formula { x-mean / sd } is used. so the mean of the generated data is 0

and sd is 1. This method is called scaling and centering.

when to use?

Highly recommended when doing regression and PCA

```
data
```

```
##   a  b
## 1 1  1
## 2 2  2
## 3 3  3
## 4 4  4
## 5 5 10
```

```
scale(data)
```

```
##              a          b
## [1,] -1.2649111 -0.8485281
## [2,] -0.6324555 -0.5656854
## [3,]  0.0000000 -0.2828427
## [4,]  0.6324555  0.0000000
## [5,]  1.2649111  1.6970563
## attr(,"scaled:center")
## a b
## 3 4
## attr(,"scaled:scale")
##        a        b
## 1.581139 3.535534
```

```
sd(data$a)# calling SD of column a
```

```
## [1] 1.581139
```

```
mean(data$a) # calling mean of column a
```

```
## [1] 3
```

```
# Applying the formula in a very primitive "non-classy" way
```

```
(1-3)/1.5 # not 1-3/1.5 -> **PEMDAS** [Parentheses,Exponents,Multiplication and Division (left-to-right),Addition and Subtraction (left-to-right).
```

```
## [1] -1.333333
```

```
(2-3)/1.5
```

```
## [1] -0.6666667
```

```
(3-3)/1.5
```

```
## [1] 0
```

```
(4-3)/1.5
```

```
## [1] 0.6666667
```

{Data Normalization} Log transformation and square -root transformation

> Log transformation is useful for data which are resulted by the multiplication of various factors.
>
> If you have zero or negative values in your data. You need to add a constant to each data point to make them larger positive and non-zero

```
x <- c(1,2,3,32,433,12,54,343,88,75)
log(x)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 3.4657359 6.0707377 2.4849066 3.9889840
##  [8] 5.8377304 4.4773368 4.3174881
```

```
log(x) + 0.5
```

```
##  [1] 0.500000 1.193147 1.598612 3.965736 6.570738 2.984907 4.488984 6.337730
##  [9] 4.977337 4.817488
```

```
sqrt(x)
```

```
##  [1]  1.000000  1.414214  1.732051  5.656854 20.808652  3.464102  7.348469
```

```
## [8] 18.520259 9.380832 8.660254
```

Data Preparation

> 80% of the effort on a typical project is spent on finding, cleaning, and preparing data for analysis. Less than 5% of the effort is devoted to analysis. “R in a nutshell”

simple concatenation using paste function (in base package)

```
c <- c(100,200,300,400,500)
d <- c(11,12,13,14,15)
paste(c,d)
```

```
## [1] "100 11" "200 12" "300 13" "400 14" "500 15"
```

you can add separator

```
paste(c,d,sep="-")
```

```
## [1] "100-11" "200-12" "300-13" "400-14" "500-15"
```

rbind and cbind or using $ sign

```
data1 <- data # just make a copy of data to work on

data1$newcol <- c # should be same length
data1
```

```
##   a  b newcol
## 1 1  1    100
## 2 2  2    200
## 3 3  3    300
## 4 4  4    400
## 5 5 10    500
```

```
cbind(data1, d) # cbind combine data by column thats why ots called c bind
```

```
##   a  b newcol  d
## 1 1  1    100 11
```

```
## 2 2  2    200 12
## 3 3  3    300 13
## 4 4  4    400 14
## 5 5 10    500 15
```

On the other hand , rbind combines rows, lets see

```
data1
```

```
##   a  b newcol
## 1 1  1    100
## 2 2  2    200
## 3 3  3    300
## 4 4  4    400
## 5 5 10    500
```

ill construct another data called data2 [just assume u have 2 excel files]

```
data2 <- data.frame(a=c(23,53,45),b=c(11,12,13),newcol=c(992,993,995))
data2
```

```
##    a  b newcol
## 1 23 11    992
## 2 53 12    993
## 3 45 13    995
```

now ill merge them together using rbind and call the data mergedDB

```
mergedDB <- rbind(data1, data2)
mergedDB
```

```
##    a  b newcol
## 1  1  1    100
## 2  2  2    200
## 3  3  3    300
## 4  4  4    400
## 5  5 10    500
## 6 23 11    992
## 7 53 12    993
## 8 45 13    995
```

Lets go classy

my graph

Merging using a common field

suppose that i have these 2 data

```
group1 <- data.frame(name=c("ali","amr","soha"), age=c(15,20,27), height=c(140,122,162),
gender=c("m","m","f"))
group1
```

```
##   name age height gender
## 1  ali  15    140      m
## 2  amr  20    122      m
## 3 soha  27    162      f
```

and

```
group2 <- data.frame(name=c("ali","amr","soha"), birthday=c("Nov","Jan","Dec"), weight=c(43,40,62))
```

I would like to see the relation between length and weights. To do this, you need to merge the data from the 2 dataframes

lets do this, ill use merge function here

```
merge(group1, group2)
```

```
##   name age height gender birthday weight
## 1  ali  15    140      m      Nov     43
## 2  amr  20    122      m      Jan     40
## 3 soha  27    162      f      Dec     62
```

** note that if the common variable differed, it will ignore the unique and show only the common one

you can try it, make a group3 and add another variable then merge the data.

??? combine data of different variables????????

The apply family

This is one of the most powerful functions for data manipulation. apply family is found in plyr and dplyr packages. please make sure you laded first plyr then dplyr

Lets see how it works [i will select some functions of these family not all]

a. apply

```
data
```

```
##   a  b
## 1 1  1
## 2 2  2
## 3 3  3
## 4 4  4
## 5 5 10
```

```
apply(data, 2, sum) # apply(data name, 1 for raw , 2 for column, your function to be applied)
```

```
##  a  b
## 15 20
```

```
# here im calling the sum of columns
```

```
apply(data, 1, sum) # applied to the rows
```

```
## [1]  2  4  6  8 15
```

More professionally, i can make my own function and pass it to apply function

```
se <- function(x) sd(x) / sqrt(length(x))
apply(data, 2, se)
```

```
##        a        b
```

```
## 0.7071068 1.5811388

#or i can make it in one line
apply(data, 2, function(x) sd(x) / sqrt(length(x)))

##         a         b
## 0.7071068 1.5811388
```

you can load psych package and code describe (data) to confirm SE.

Lets use other functions [try changing the function to length, sd,range,quantile, max, min]

b. sapply

sapply returns vector or matrix mainly

```
sapply(data, FUN=function(x) x + 3)

##      a  b
## [1,] 4  4
## [2,] 5  5
## [3,] 6  6
## [4,] 7  7
## [5,] 8 13
```

c. tapply

This is a very classy function to apply function based on a certain variable

OK, let me recall data

```
data

##   a  b
## 1 1  1
## 2 2  2
## 3 3  3
## 4 4  4
## 5 5 10
```

Ill add a column and ill tell you why later

```
data$f <- c("M", "M", "M", "F", "F")
```

```
data
```

```
##   a  b f
## 1 1  1 M
## 2 2  2 M
## 3 3  3 M
## 4 4  4 F
## 5 5 10 F
```

OK now i need to get the sum of a and b based on f. in another word, male and females

```
tapply(data$b, data$f, FUN=mean) # tapply(what, based on what, function)
```

```
## F M
## 7 2
```

Finding and removing duplicate hits

> In many times, you may need to remove duplication in a certain data. here is how to do it

Lets make some duplicate data

```
data
```

```
##   a  b f
## 1 1  1 M
## 2 2  2 M
## 3 3  3 M
## 4 4  4 F
## 5 5 10 F
```

ill add some duplicates

```
added <-data.frame(a=c(2,3,5), b=c(11,12,13), f=c("M", "F", "F"))
ddata <- rbind(data, added)
ddata
```

```
##   a  b f
## 1 1  1 M
```

```
## 2 2  2 M
## 3 3  3 M
## 4 4  4 F
## 5 5 10 F
## 6 2 11 M
## 7 3 12 F
## 8 5 13 F
```

```
duplicated(ddata$a)
```

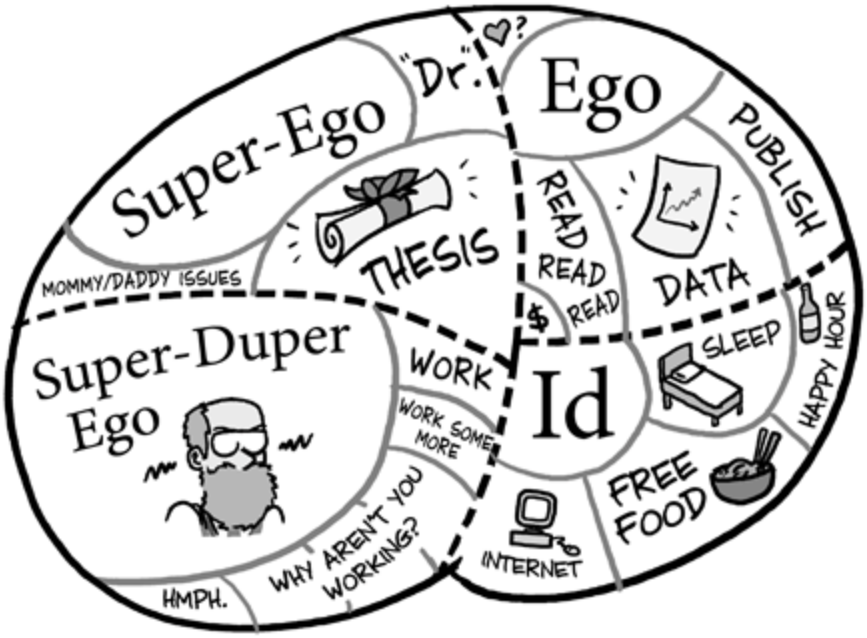
```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

If you want to remove the duplicated data

```
unique <- ddata[!duplicated(ddata$a), ]
unique
```

```
##   a  b f
## 1 1  1 M
## 2 2  2 M
## 3 3  3 M
## 4 4  4 F
## 5 5 10 F
```

Scientist mind


Super-Ego
"Dr."
Ego
PUBLISH
THESIS
READ
READ
READ
DATA
MOMMY/DADDY ISSUES
Super-Duper
Ego
WORK
Id
SLEEP
HAPPY HOUR
WORK SOME MORE
FREE FOOD
WHY AREN'T YOU WORKING?
INTERNET
HMPH.

my graph