

# Rworkshop

## Data processing in R

---

Bolin Wu

Aging Research Center

2022-03-24 (updated: 2022-03-21)

# Reminder

- What we will talk about next usually can take 1-2 whole sessions for a master program in order to understand the technical details. Today the main purpose is to go through the tasks that R are capable of solving.
- R studio is very user-friendly. Whenever you have problem with a function, just type "?" + `function()`. For example:

```
?summary  
?lm  
?data.frame
```

--->R Studio

- If possible, please follow along with your computer, welcome to ask question at any time.

# Data

The example data we use are `paquid_cog.csv` and `paquid_cov.csv`.

## Load package

we can load the package by using `library()` function.

- `tidyverse`: A powerful package for data wrangling, which I will show later.
- `Hmisc`: We need this package for adding label to `data.frame()`.

```
library(tidyverse)
library(Hmisc)
```

## Import data

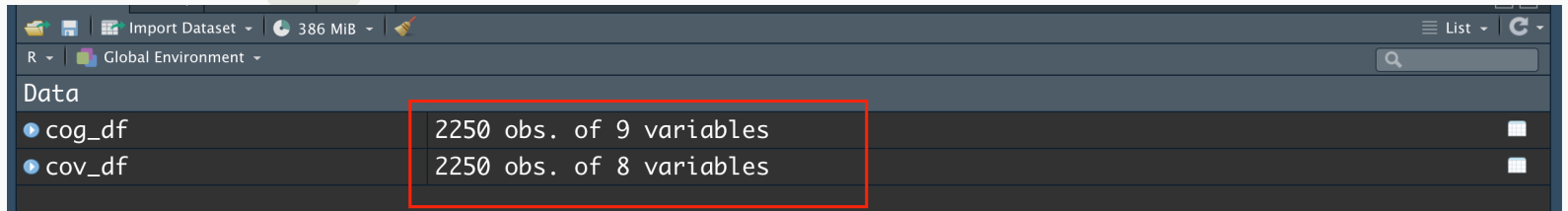
There are many different functions to read data, depending on the file's type. Since we have csv file, we can use `read.csv()` to read the data.

```
cog_df <- read.csv(file = "data/paquid_cog.csv")
cov_df <- read.csv(file = "data/paquid_cov.csv")
# Use ?read.csv() to check what other arguments you can put in
```

# Explore data

Usually the first thing we need to do is to check if the data are **correctly imported**.

- Looking at the Data panel



- Use `str()` function to check if the column types are as expected.

```
# check data's type
str(cog_df)
```

```
## 'data.frame':    2250 obs. of  9 variables:
## $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
## $ ID     : int  1 2 2 2 2 2 3 3 4 4 ...
## $ MMSE   : int  26 26 28 25 24 22 28 25 NA 22 ...
## $ BVRT   : int  10 13 13 12 13 9 13 6 NA 8 ...
## $ IST    : int  37 25 28 23 16 15 28 16 NA 27 ...
## $ dem    : int   0 1 1 1 1 1 0 0 0 0 ...
## $ agedem: num  68.5 85.6 85.6 85.6 85.6 ...
## $ age    : num  68.5 67 69.1 73.8 84.1 ...
```

# Explore data

```
# check data's type  
str(cov_df)
```

```
## 'data.frame':    2250 obs. of  8 variables:  
## $ X          : int  1 2 3 4 5 6 7 8 9 10 ...  
## $ ID         : int  1 2 2 2 2 2 3 3 4 4 ...  
## $ age_init   : num  67.4 65.9 65.9 65.9 65.9 ...  
## $ CEP       : int  1 1 1 1 1 1 1 1 1 1 ...  
## $ male      : int  1 0 0 0 0 0 1 1 0 0 ...  
## $ HIER      : int  2 1 1 1 3 3 1 1 1 1 ...  
## $ age       : num  68.5 67 69.1 73.8 84.1 ...  
## $ wave      : int  1 1 2 3 4 5 1 2 1 2 ...
```

- Here we can see the data types are as expected.
- However, there is a non-sense column "X", we need to get rid of it. The easiest way to use `select()` function from `tidyverse` package.

```
# get rid of the first column 'X'  
cog_df <- cog_df %>% select(-X)  
cov_df <- cov_df %>% select(-X)
```

# Explore data

- Use `head()` function to check the first few rows of dataframe

```
head(cov_df)
```

```
##   ID age_init CEP male HIER      age wave
## 1  1  67.4167   1    1    2 68.50630    1
## 2  2  65.9167   1    0    1 66.99540    1
## 3  2  65.9167   1    0    1 69.09530    2
## 4  2  65.9167   1    0    1 73.80720    3
## 5  2  65.9167   1    0    3 84.14237    4
## 6  2  65.9167   1    0    3 87.09103    5
```

- Use `class()` function to check the type of `cov_df`: Is it a matrix, or `data.frame`, or list, or tibble?

```
class(cov_df)
```

```
## [1] "data.frame"
```

Personally I would love to convert `data.frame` to `tibble` for data wrangling. --> R studio...

# Explore data

We may also be interested in checking the descriptive statistics: `summary()`

```
# descriptive statistics
summary(cog_df)
```

```
##           ID           MMSE           BVRT           IST
##  Min.      : 1.0    Min.      : 0.00    Min.      : 0.00    Min.      : 5.00
## 1st Qu.:132.2    1st Qu.:25.00    1st Qu.: 9.00    1st Qu.:22.00
## Median :263.0    Median :27.00    Median :11.00    Median :27.00
## Mean     :256.2    Mean     :25.99    Mean     :10.78    Mean     :26.52
## 3rd Qu.:376.0    3rd Qu.:29.00    3rd Qu.:13.00    3rd Qu.:31.00
## Max.     :500.0    Max.     :30.00    Max.     :15.00    Max.     :40.00
##
##           NA's      :36    NA's      :300    NA's      :198
##
##           dem           agedem           age           wave
##  Min.      :0.0000    Min.      :66.87    Min.      : 66.28    Min.      :1.000
## 1st Qu.:0.0000    1st Qu.:82.03    1st Qu.: 75.09    1st Qu.:2.000
## Median :0.0000    Median :86.61    Median : 80.57    Median :3.000
## Mean     :0.3329    Mean     :85.89    Mean     : 80.53    Mean     :3.534
## 3rd Qu.:1.0000    3rd Qu.:89.94    3rd Qu.: 85.77    3rd Qu.:5.000
## Max.     :1.0000    Max.     :99.49    Max.     :102.16    Max.     :9.000
##
```

# Generate and label variables

1. Generate a variable “fu”, which means follow-up time and equals to age - age\_init

- Let's first review cov\_df:

```
head(cov_df)
```

```
##   ID age_init CEP male HIER      age wave
## 1  1  67.4167   1    1    2 68.50630    1
## 2  2  65.9167   1    0    1 66.99540    1
## 3  2  65.9167   1    0    1 69.09530    2
## 4  2  65.9167   1    0    1 73.80720    3
## 5  2  65.9167   1    0    3 84.14237    4
## 6  2  65.9167   1    0    3 87.09103    5
```

- How do we extract a specific column? By Google you can find many ways, one of them is to use `$` sign. “[1:10]” means take the elements one to ten.

```
cov_df$age_init[1:10]
```

```
## [1] 67.4167 65.9167 65.9167 65.9167 65.9167 65.9167 71.5000 71.5000
## [9] 66.0000 66.0000
```



# Generate and label variables

1. Generate a variable “fu”, which means follow-up time and equals to age - age\_init

- Now we are ready to generate the new variable:

```
fu ← cov_df$age - cov_df$age_init
# if you want to add the new variable to dataframe
cov_df$fu ← fu
head(cov_df)
```

##	ID	age_init	CEP	male	HIER	age	wave	fu
## 1	1	67.4167	1	1	2	68.50630	1	1.08960
## 2	2	65.9167	1	0	1	66.99540	1	1.07870
## 3	2	65.9167	1	0	1	69.09530	2	3.17860
## 4	2	65.9167	1	0	1	73.80720	3	7.89050
## 5	2	65.9167	1	0	3	84.14237	4	18.22567
## 6	2	65.9167	1	0	3	87.09103	5	21.17433

# Generate and label variables

2. Generate a variable “dem\_young”, which means age of dementia onset (variable “agedem”)  $\leq 70$  years old (use the if/else statement).

```
dem_young ← ifelse(cog_df$agedem ≤ 70, yes = 1, no = 0)
# put dem_young to cog_df
cog_df$dem_young ← dem_young
head(cog_df)
```

##	ID	MMSE	BVRT	IST	dem	agedem	age	wave	dem_young
## 1	1	26	10	37	0	68.5063	68.50630	1	1
## 2	2	26	13	25	1	85.6167	66.99540	1	0
## 3	2	28	13	28	1	85.6167	69.09530	2	0
## 4	2	25	12	23	1	85.6167	73.80720	3	0
## 5	2	24	13	16	1	85.6167	84.14237	4	0
## 6	2	22	9	15	1	85.6167	87.09103	5	0

# Generate and label variables

3. **Rename** variable “CEP” as “education” and change the variable class to **factor**. **Label** the variable values as 0=“Below primary school”, 1=“Primary school and above”.

- We can rename a specific column "CEP" as follows:

```
colnames(cov_df)[colnames(cov_df) == "CEP"] <- "education"  
head(cov_df)
```

```
##   ID age_init education male HIER      age wave      fu  
## 1  1  67.4167          1     1    2 68.50630    1  1.08960  
## 2  2  65.9167          1     0    1 66.99540    1  1.07870  
## 3  2  65.9167          1     0    1 69.09530    2  3.17860  
## 4  2  65.9167          1     0    1 73.80720    3  7.89050  
## 5  2  65.9167          1     0    3 84.14237    4 18.22567  
## 6  2  65.9167          1     0    3 87.09103    5 21.17433
```

- Wait, what is going on here? ---> R studio

# Generate and label variables

3. **Rename** variable “CEP” as “education” and change the variable class to **factor**. **Label** the variable values as 0=“Below primary school”, 1=“Primary school and above”.

- To label the variable we need to use the fore-mentioned `Hmisc` package.

Actually this is also new for me...

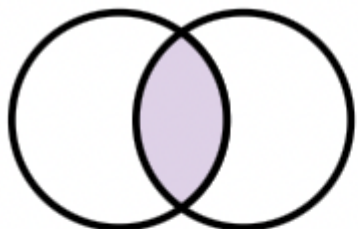
```
label(cov_df[["education"]]) ← "0='Below primary school', 1='Primary school and above"
```

And then you can use View panel to check the label in the dataset. ---> R Studio.

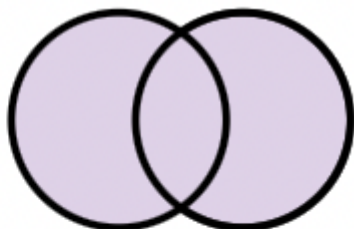
# Merge and reshape data sets

4. **Merge** datasets “paquid\_cog” and “paquid\_cov” to a data frame named “paquid”.

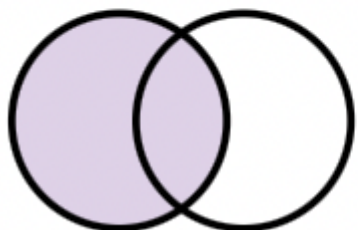
- With `join_` function: There are 4 common types of joins.



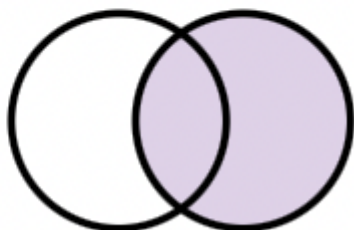
`inner_join(x, y)`



`full_join(x, y)`



`left_join(x, y)`



`right_join(x, y)`

# Merge and reshape data sets

4.**Merge** datasets “paquid\_cog” and “paquid\_cov” to a data frame named “paquid”.

```
paquid <- full_join(x = cog_df, y = cov_df, by = c("ID", "wave", "age"))  
head(paquid)
```

```
##   ID MMSE BVRT IST dem   agedem      age wave dem_young age_init  
## 1  1   26   10  37   0 68.5063 68.50630     1         1  67.4167  
## 2  2   26   13  25   1 85.6167 66.99540     1         0  65.9167  
## 3  2   28   13  28   1 85.6167 69.09530     2         0  65.9167  
## 4  2   25   12  23   1 85.6167 73.80720     3         0  65.9167  
## 5  2   24   13  16   1 85.6167 84.14237     4         0  65.9167  
## 6  2   22    9  15   1 85.6167 87.09103     5         0  65.9167  
##   education male HIER      fu  
## 1          1    1    2  1.08960  
## 2          1    0    1  1.07870  
## 3          1    0    1  3.17860  
## 4          1    0    1  7.89050  
## 5          1    0    3 18.22567  
## 6          1    0    3 21.17433
```

# Merge and reshape data sets

- With `merge` function: In merge function, it finds columns with the same name by default. Therefore we only need to specify x and y data sets. You can recheck this by reading `?merge`.

```
paquid2 <- merge(x = cog_df, y = cov_df)
head(paquid2, n = 3)
```

```
##   ID      age wave MMSE BVRT IST dem   agedem dem_young age_init
## 1  1 68.5063    1   26   10  37   0 68.50630          1  67.4167
## 2 10 81.0601    1   30   12  35   0 88.01694          0  77.7500
## 3 10 83.0861    2   28    8  22   0 88.01694          0  77.7500
##   education male HIER      fu
## 1          1    1    2 1.0896
## 2          0    0    1 3.3101
## 3          0    0    1 5.3361
```

- This result is difficult to compare with paquid. We can reorder paquid2 so that it has the same column/row order as paquid.

```
paquid2 <- paquid2[order(paquid2$ID), names(paquid)]
head(paquid2,n = 3)
```

```
##      ID MMSE BVRT IST dem  agedem      age wave dem_young age_init
## 1      1   26   10   37   0 68.5063 68.5063     1         1 67.4167
## 449    2   26   13   25   1 85.6167 66.9954     1         0 65.9167
## 450    2   28   13   28   1 85.6167 69.0953     2         0 65.9167
##      education male HIER      fu
## 1              1     1     2 1.0896
## 449             1     0     1 1.0787
## 450             1     0     1 3.1786
```

```
head(paquid,n = 3)
```

```
##      ID MMSE BVRT IST dem  agedem      age wave dem_young age_init
## 1      1   26   10   37   0 68.5063 68.5063     1         1 67.4167
## 2      2   26   13   25   1 85.6167 66.9954     1         0 65.9167
## 3      2   28   13   28   1 85.6167 69.0953     2         0 65.9167
##      education male HIER      fu
## 1              1     1     2 1.0896
## 2              1     0     1 1.0787
## 3              1     0     1 3.1786
```



# Merge and reshape data sets

5. Reshape the “paquid” data to **wide** format.

- Reshaping a data set is pretty tricky, usually I need to test out several times before I can arrive the final interested format.

# Merge and reshape data sets

- Points to consider:
  1. What is the variable that measures the time span?
  2. What variables we would like to spread? Is it only one column to spread, or multiple?
  3. What variables are unchanged?

# Merge and reshape data sets

- Time span: column `wave` represents the time span.

```
summary.factor(paquid$wave)
```

```
##    1    2    3    4    5    6    7    8    9  
## 500 424 346 288 232 173 134 100  53
```

- Columns to change
  - If only spread `MMSE` column

```
paquid_wide = spread(data = paquid, value = "MMSE", key = "wave", sep = "MMSE")
```

```
datatable(paquid_wide)
```

Show  entries

Search:

	ID ⬆	BVRT ⬆	IST ⬆	dem ⬆	agedem ⬆	age ⬆	dem_young ⬆	age_init ⬆	ec
1	1	10	37	0	68.5063	68.5063	1	67.4167	
2	2	13	25	1	85.61670089	66.9954	0	65.9167	

- Columns to change
  - If only spread `MMSE` column.
  - If we want to change all the columns varying at waves. (`reshape()` function)

```
# timevar: the variable in long format that differentiates multiple records from the same ID
# idvar: Columns that will not be affected, stay the same
unchange_column <- c("ID", "age_init", "education", "male", "agedem", "dem", "dem_young")
wide_paquid <- reshape(data = paquid, timevar = "wave", idvar = unchange_column, direction = "wide")
datatable(wide_paquid)
```

Show  entries

Search:

	ID ↕	dem ↕	agedem ↕	dem_young ↕	age_init ↕	education ↕	male ↕	MMSE_1 ↕
1	1	0	68.5063	1	67.4167	1	1	26
2	2	1	85.61670089	0	65.9167	1	0	26
7	3	0	74.7334	0	71.5	1	1	28
9	4	0	87.633127995	0	66	1	0	

# Row-wise calculation

6. Generate a variable named “MMSE\_M”, which is the number of missing values across variables “MMSE\_1”, “MMSE\_2”, ..., “MMSE\_9” per individual. Label the variable as “the number of missing values in MMSE”.

```
# convert dataframe to tibble for faster data cleaning
wide_paquid <- as_tibble(wide_paquid)
paquid_MMSE <- wide_paquid %>% select(c("ID", contains("MMSE")))
datatable(paquid_MMSE)
```

Show  entries

Search:

	ID ⚙	MMSE_1 ⚙	MMSE_2 ⚙	MMSE_3 ⚙	MMSE_4 ⚙	MMSE_5 ⚙	MMSE_6 ⚙	MMSE_7 ⚙
1	1	26						
2	2	26	28	25	24	22		
3	3	28	25					
4	4		22					
5	5	29	27	30	28	28	27	26

# Row-wise calculation

- Use `rowSums` combined with `is.na`.

```
MMSE_M <- rowSums(is.na(paquid_MMSE))  
length(MMSE_M)
```

```
## [1] 500
```

```
MMSE_M[1:10]
```

```
## [1] 8 4 7 8 2 3 6 2 6 5
```

```
# merge the vector to the tibble  
paquid_MMSE$MMSE = MMSE_M
```

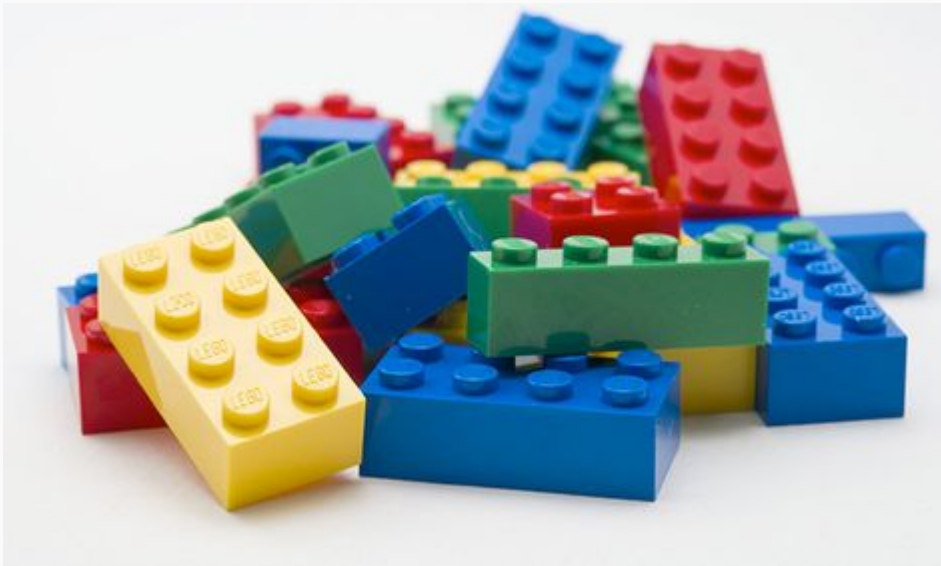
# Row-wise calculation

7.View variables that contain “MMSE”.

- Use the `selection` from `tidyverse` package.

```
wide_paquid %>%  
  select(contains("MMSE"))
```

Beautiful syntax with pipeline, just like playing LEGO. ---> R studio



# Row-wise calculation

8. Generate variables “MEM\_1”, “MEM\_2”, ..., “MEM\_9” which equals the mean of “BVRT” and “IST” at each time point.

- The original wide format data set has too many columns... Let's start with one babystep first:

```
# this tibble is the "ingredient"
wide_paquid %>% select(contains(c("ID", "BVRT", "IST")))
```

```
## # A tibble: 500 x 19
```

```
##       ID BVRT_1 BVRT_2 BVRT_3 BVRT_4 BVRT_5 BVRT_6 BVRT_7 BVRT_8
##   <int> <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
##  1     1     10    NA    NA    NA    NA    NA    NA    NA
##  2     2     13    13    12    13     9    NA    NA    NA
##  3     3     13     6    NA    NA    NA    NA    NA    NA
##  4     4    NA     8    NA    NA    NA    NA    NA    NA
##  5     5     10    13    12    11    11     9    12    NA
##  6     6     12    14    11    10     9     8    NA    NA
##  7     7      5     2    NA    NA    NA    NA    NA    NA
##  8     8     11    11    12     8    10     9     8    NA
##  9     9     11    12    12    NA    NA    NA    NA    NA
## 10    10     12     8     8    NA    NA    NA    NA    NA
```



# Row-wise calculation

- What should  $MEM_1$  look like?

$$MEM_1 = (BVRT_1 + IST_1)/2$$

```
wide_paquid %>%  
  select(contains(c("ID", "BVRT_1", "IST_1"))) %>%  
  rowwise("ID") %>% # indicate the row index  
  mutate(MEM_1 = mean(c(BVRT_1, IST_1))) %>% # mutate: create new column  
  ungroup() # need to ungroup for rowwise calculation
```

```
## # A tibble: 500 x 4  
##       ID BVRT_1 IST_1 MEM_1  
##   <int> <int> <int> <dbl>  
## 1     1     1    10    37  23.5  
## 2     2     2    13    25   19  
## 3     3     3    13    28  20.5  
## 4     4     4    NA     NA   NA  
## 5     5     5    10    34   22  
## 6     6     6    12    37  24.5  
## 7     7     7     5    16  10.5  
## 8     8     8    11    29   20  
## 9     9     9    11    25   18  
## 10    10    10    12    35  23.5
```

# Row-wise calculation

- Can we do the above process directly on the whole wide dataframe? --- Yes!

```
wide_paquid %>%  
  rowwise("ID") %>%  
  mutate(MEM_1 = mean(c(BVRT_1,IST_1))) %>%  
  ungroup() %>%  
  datatable()
```

Show  entries

Search:

	ID ↕	dem ↕	agedem ↕	dem_young ↕	age_init ↕	education ↕	male ↕	MMSE_1 ↕
1	1	0	68.5063	1	67.4167	1	1	26
2	2	1	85.61670089	0	65.9167	1	0	26
3	3	0	74.7334	0	71.5	1	1	28
4	4	0	87.633127995	0	66	1	0	
5	5	0	88.128678987	0	67.3333	1	0	29
6	6	1	86.543361157	0	70.0833	1	1	45 / 71 <sup>29</sup>

# Row-wise calculation

- We can do the same procedure for the rest eight variables by loop and assign() function. This flexibility renders R an advantage concerning repetitive work.

```
for (i in 1:9) {  
  BVRT_i = paste0("BVRT_", i)  
  IST_i = paste0("IST_", i)  
  # create new variable name  
  MEM_i = paste0("MEM_", i)  
  # assign value to each new variable  
  wide_paquid = wide_paquid %>%  
    rowwise("ID") %>%  
    mutate(!!sym(MEM_i) := mean(c(get(BVRT_i),get(IST_i)))) %>%  
    ungroup()  
}  
# sym(new_col_name) := is a dynamic way of writing MEM_1 = , MEM_2 = ,etc  
# when using functions like mutate()  
# in the tidyr package
```

# Row-wise calculation

9.View variables that contain “MEM”, “BVRT”, or “IST”:

```
wide_paquid %>%  
  select(contains(c("ID", "MEM", "BVRT", "IST"))) %>%  
  datatable()
```

Show  entries

Search:

	ID	MEM_1	MEM_2	MEM_3	MEM_4	MEM_5	MEM_6	MEM_7	MEM_8
1	1	23.5							
2	2	19	20.5	17.5	14.5	12			
3	3	20.5	11						
4	4		17.5						
5	5	22	24.5	24	19.5	23	17	19.5	
6	6	24.5	26	19.5	20	17.5	17		
7	7	10.5	9						

# Summarizing data

10. Summarize variable “age\_init” (mean, sd, quantiles, etc), summarize “age\_init” by variable “male”.

- As is shown before, we can use `summary` function.

```
summary(wide_paquid$age_init)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  65.25   68.42   73.83   74.23   78.42   92.33
```

```
# sd is missing, we can calculate it with sd() function
sd(wide_paquid$age_init)
```

```
## [1] 6.392525
```

# Summarizing data

- If we would like to summary by a specific group: Use `group_by` function and `summarise` function, connected by pipeline:

```
wide_paquid %>%  
  group_by(male) %>%  
  summarise(max = max(age_init),  
            q3 = quantile(age_init, 0.75),  
            mean = mean(age_init),  
            q1 = quantile(age_init, 0.25),  
            min = min(age_init),  
            sd = sd(age_init)  
            )
```

```
## # A tibble: 2 x 7  
##   male    max    q3  mean    q1   min    sd  
##   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1     0  91.9  79.6  74.8  68.2  65.5  6.83  
## 2     1  92.3  77.5  73.5  68.5  65.2  5.68
```

# Summarizing data

11. Summarize variable “MMSE\_1” (mean, sd, quantiles, etc), summarize “age\_init” by variable “male”. Note how R deals with missing values.

This question is similar to the previous one. I would leave it to the audience.

# Summarizing data

12. Tabulate variable “male”, tabulate variable “male” and “education”, add row-wise and column-wise proportions.

- Find frequency of elements in male

```
table(wide_paquid$male)
```

```
##  
##      0      1  
## 288 212
```

- Male and education

```
tab_male_edu = table(wide_paquid %>% select("male", "education"))  
prop.table(tab_male_edu)
```

```
##      education  
## male      0      1  
##      0 0.202 0.374  
##      1 0.088 0.336
```



# Summarizing data

- It seems difficult to add column to table() directly, so we convert it to tibble first

```
tib_male_edu = as_tibble(table(wide_paquid %>% select("male", "education")))
tib_male_edu
```

```
## # A tibble: 4 x 3
##   male education      n
##   <chr> <chr>    <int>
## 1 0      0      101
## 2 1      0       44
## 3 0      1      187
## 4 1      1      168
```

# Summarizing data

- Add proportion column to the tibble:

```
tib_male_edu$proportion = tib_male_edu$n / 500
tib_male_edu
```

```
## # A tibble: 4 x 4
##   male education      n proportion
##   <chr> <chr>    <int>    <dbl>
## 1 0      0      101    0.202
## 2 1      0       44    0.088
## 3 0      1      187    0.374
## 4 1      1      168    0.336
```

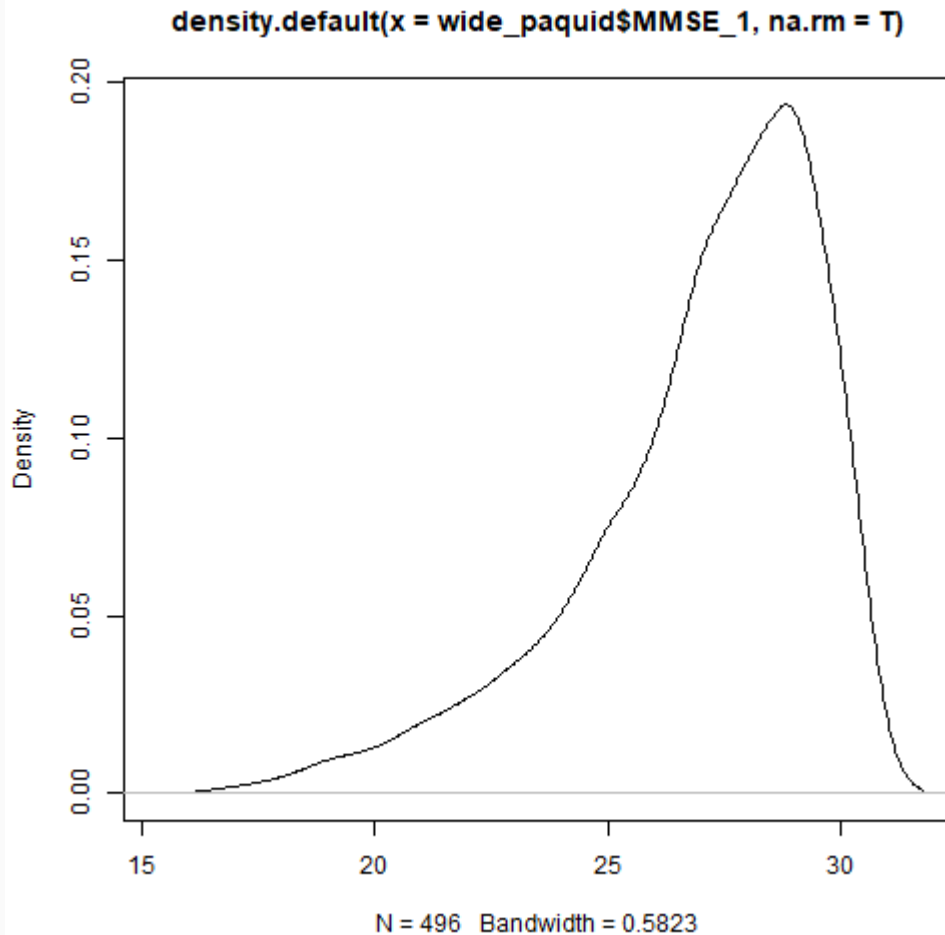
# Summarizing data

13. Draw a histogram and a density plot of “MMSE\_1”: Use `hist` and `plot` function:

```
hist(wide_paquid$MMSE_1)
```

# Summarizing data

```
plot(density(wide_paquid$MMSE_1, na.rm = T))
```



# Run simple models and check model

14. Run a linear regression, with “**MMSE\_1**” as **dependent** variable and “**age\_init**” and “**male**” as the **independent** variables, assuming “MMSE\_1” has a normal distribution. Check model output.

- Use `lm` function. ---> R studio

```
linear_m = lm(formula = MMSE_1 ~ age_init + male, data = wide_paquid)
```

# Run simple models and check model

- Output from `summary` function:

```
summary(linear_m)
```

```
##  
## Call:  
## lm(formula = MMSE_1 ~ age_init + male, data = wide_paquid)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -8.9013 -1.0920  0.5365  1.7564  4.6739   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept) 36.69160    1.31023  28.004 < 2e-16 ***   
## age_init    -0.13127    0.01740  -7.542 2.24e-13 ***   
## male         0.06449    0.22498   0.287  0.775      
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##
```

# Run simple models and check model

- Extract certain information from `summary` function:

```
summary(linear_m)$coefficients
```

```
##              Estimate Std. Error   t value    Pr(>|t|)
## (Intercept) 36.69160077 1.31022670 28.0040093 5.656499e-104
## age_init    -0.13126706 0.01740399 -7.5423524 2.238581e-13
## male        0.06448892 0.22498083  0.2866418 7.745070e-01
```

This output is super useful, when we have many combinations of dependent ~ independent.  
For example:

- Assuming we have 12 air pollution variables & 10 MRI variables. There are 120 pairs in total.
- We want to find which air pollutants could be significant in linear regression with MRI. Controlling for age, education, gender, etc...
- A loop can easily solve this problem! (Code script is available if anyone needs.)

# Run simple models and check model

- Check the residual for model evaluation: ---> R studio

```
# residuals  
linear_m$residuals  
plot(linear_m)
```



# Run simple models and check model

15. Run a logistic regression, with “dem\_young” as dependent variable and “male” as the independent variables.

- Use `glm` function, set `family = binomial`. It is similar to `lm` function.

```
logi_m = glm(formula = dem_young ~ male, data = wide_paquid, family = binomial)
logi_m
```

```
##
## Call:  glm(formula = dem_young ~ male, family = binomial, data = wide_paquid)
##
## Coefficients:
## (Intercept)          male
##    -3.6924         0.4538
##
## Degrees of Freedom: 499 Total (i.e. Null);  498 Residual
## Null Deviance:          134.7
## Residual Deviance: 134      AIC: 138
```

# Run simple models and check model

```
# output
summary(logi_m)
summary(logi_m)$coefficients
# similar to lm function's output

# residuals
logi_m$residuals

# fitted values
fitted.values(logi_m)
```

# Run simple models and check model

- For binary estimation, we are usually interested in the predicted probability for each observation and the overall prediction accuracy.

$$\text{accuracy} = \frac{\text{Number of correct prediction}}{\text{Number of ovservation}}$$

```
# probability
probabilities = predict(logi_m,type = "response")
probabilities[1:5]
```

```
##           1           2           3           4           5
## 0.03773585 0.02430556 0.03773585 0.02430556 0.02430556
```

```
# overall accuracy, assume threshold = 0.5
dicted.classes <- ifelse(probabilities > 0.5, 1, 0)
mean(predicted.classes == wide_paquid$dem_young)
```

```
## [1] 0.97
```

# Ending

- At first you may feel difficult. But don't worry, R has a strong user community, basically you can solve most of the problems by Google.
- After using R several years, I still rely heavily on Google and I keep learning new things.
- Because of its strong community, R now has many extensions. For example, this whole slide is written by R...
- R code for all the exercises is available on my github page: [https://github.com/Bolin-Wu/Rworkshop\\_KI](https://github.com/Bolin-Wu/Rworkshop_KI). If you have any question please feel free to ask and we can discuss together!