충남대학교
Chungnam National University

투자론

- *R*과 Excel을 통한 금융데이터 분석 -

7주차
R(입문 및 데이터 포맷)과 Data Management

충남대학교
장호규 교수

Unit 03

Data Management

Overview

- Data Management
- Basic Data Analysis
- Graphics

Preparing Data for Analysis

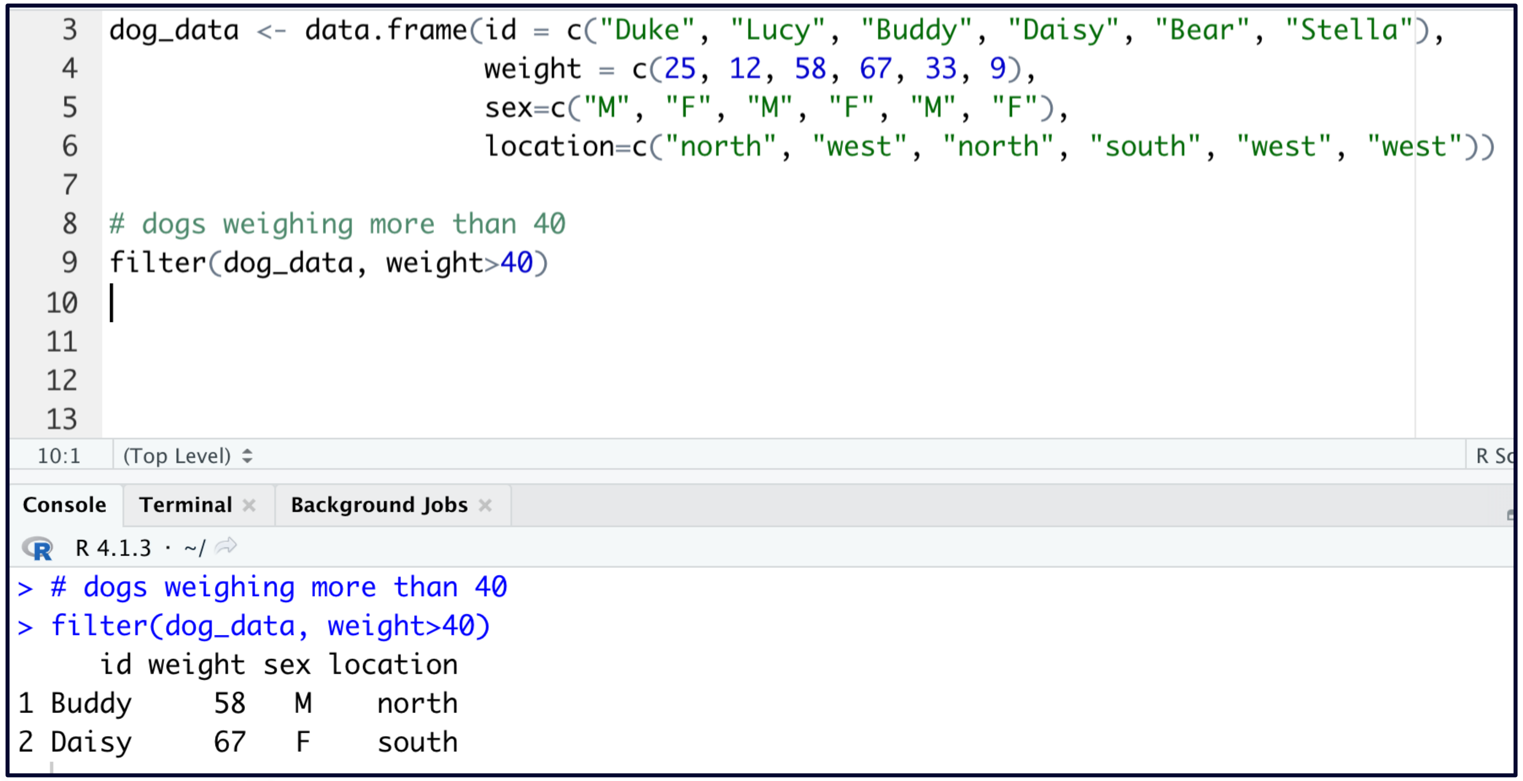
- Most widely used package is dplyr
- Load package by using library(dplyr)
- Key functions in dplyr

`filter()`	extract specific rows
`select()`	extract specific columns
`arrange()`	arrange
`mutate()`	add a variabe
`summarise()`	summarize statistics
`group_by()`	divide by specific groups
`left_join()`	merge data sets
`bind_rows()`	merge data by rows

Subsetting Rows of a Data Frame

Example

- `filter()`



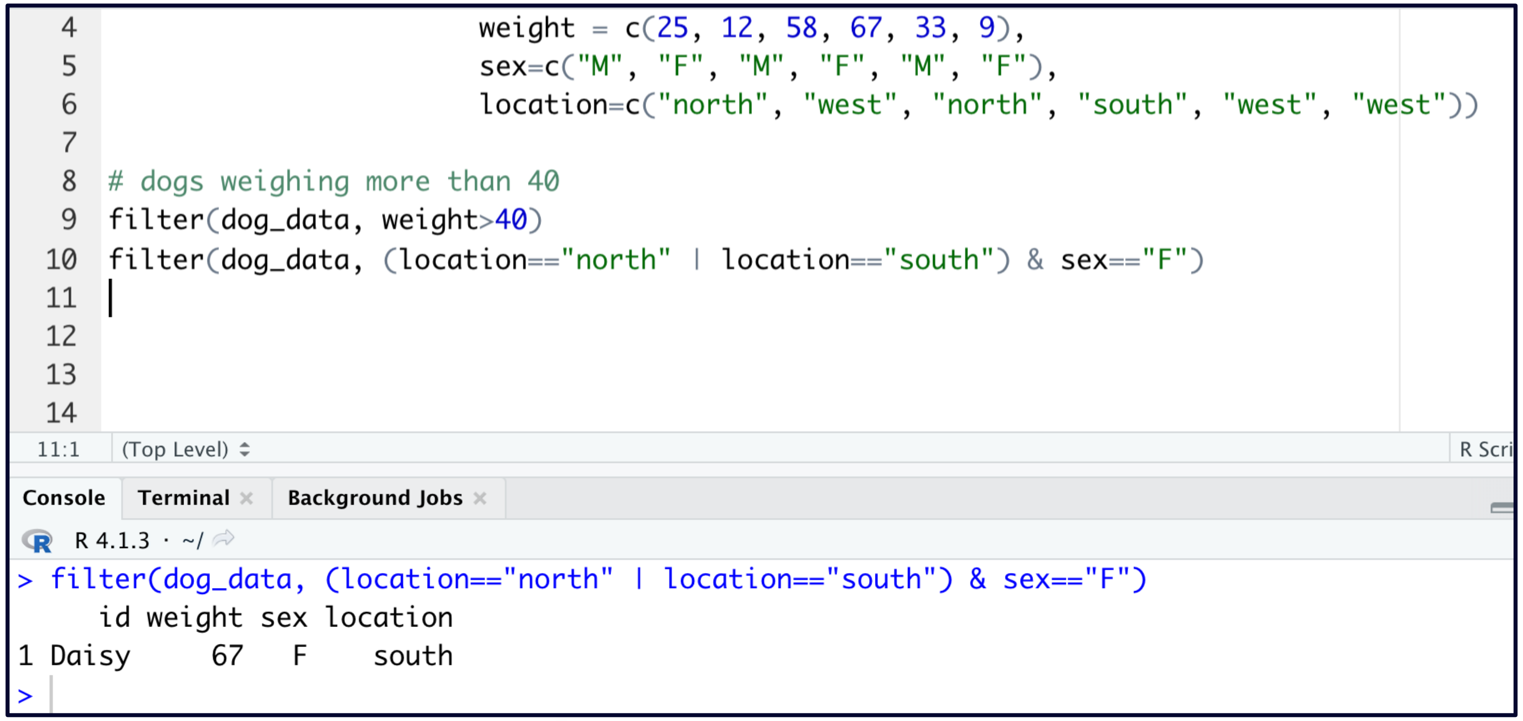
```
dog_data <- data.frame(id = c("Duke", "Lucy", "Buddy", "Daisy", "Bear", "Stella"),
                       weight = c(25, 12, 58, 67, 33, 9),
                       sex=c("M", "F", "M", "F", "M", "F"),
                       location=c("north", "west", "north", "south", "west", "west"))

# dogs weighing more than 40
filter(dog_data, weight>40)
```

```
> # dogs weighing more than 40
> filter(dog_data, weight>40)
     id weight sex location
1 Buddy     58   M    north
2 Daisy     67   F    south
```

Subsetting Rows of a Data Frame

Example

- `filter()`



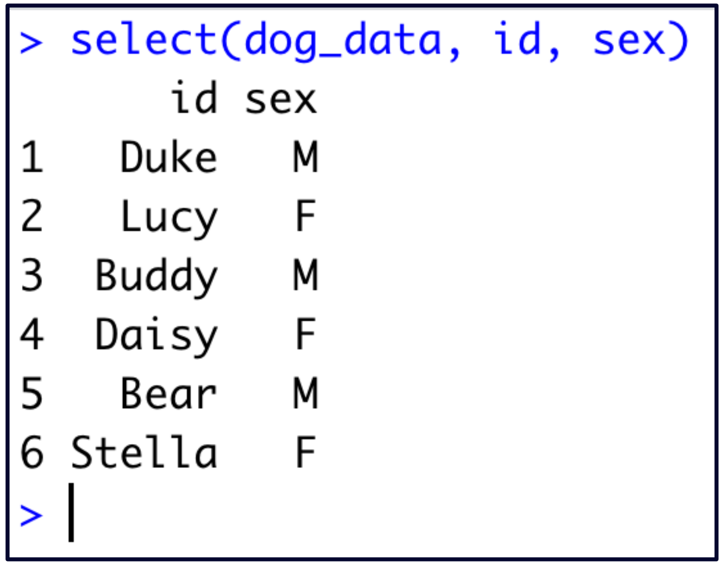
```
                         weight = c(25, 12, 58, 67, 33, 9),
                         sex=c("M", "F", "M", "F", "M", "F"),
                         location=c("north", "west", "north", "south", "west", "west"))

# dogs weighing more than 40
filter(dog_data, weight>40)
filter(dog_data, (location=="north" | location=="south") & sex=="F")
```

```
> filter(dog_data, (location=="north" | location=="south") & sex=="F")
     id weight sex location
1 Daisy     67   F    south
>
```

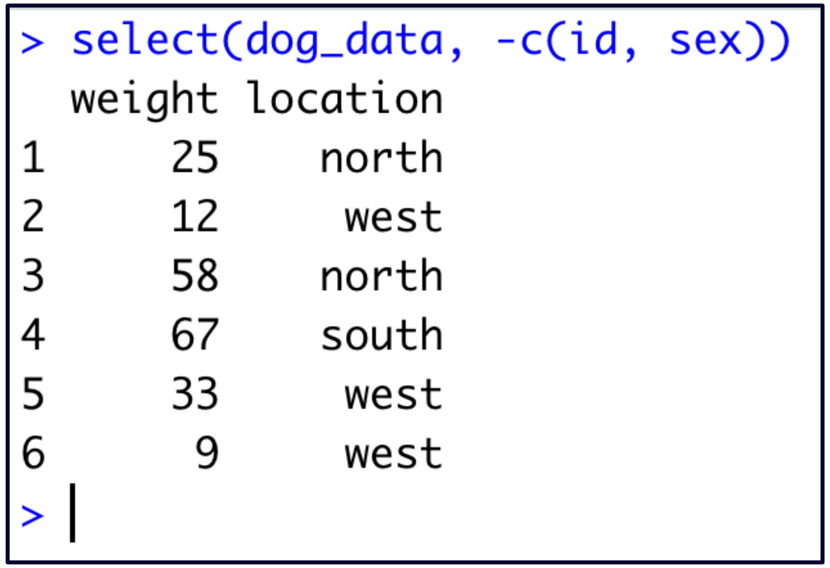
Subsetting Rows of a Data Frame

Example

- select()



```
> select(dog_data, id, sex)
      id sex
1   Duke   M
2   Lucy   F
3  Buddy   M
4  Daisy   F
5   Bear   M
6 Stella   F
> |
```

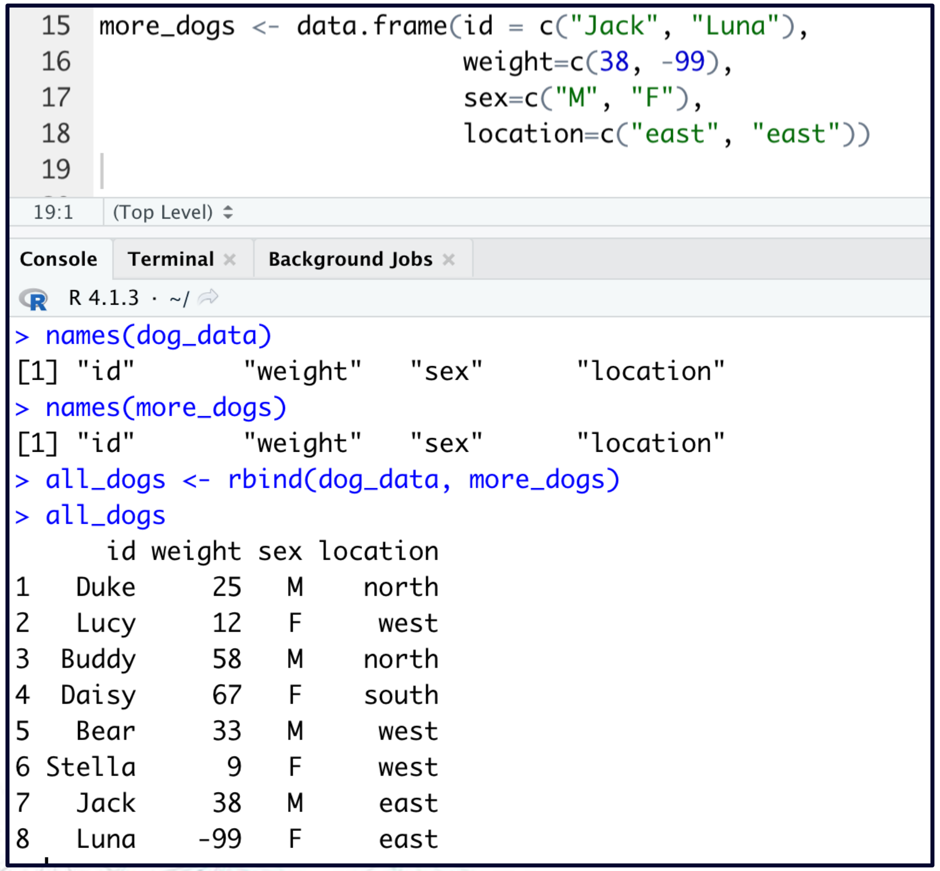


```
> select(dog_data, -c(id, sex))
  weight location
1     25    north
2     12     west
3     58    north
4     67    south
5     33     west
6      9     west
> |
```

Subsetting Rows of a Data Frame

Example

- `rbind()`



```
more_dogs <- data.frame(id = c("Jack", "Luna"),
                        weight=c(38, -99),
                        sex=c("M", "F"),
                        location=c("east", "east"))
```

```
> names(dog_data)
[1] "id"       "weight"   "sex"      "location"
> names(more_dogs)
[1] "id"       "weight"   "sex"      "location"
> all_dogs <- rbind(dog_data, more_dogs)
> all_dogs
      id weight sex location
1   Duke     25   M    north
2   Lucy     12   F     west
3  Buddy     58   M    north
4  Daisy     67   F    south
5   Bear     33   M     west
6 Stella      9   F     west
7   Jack     38   M     east
8   Luna    -99   F     east
```

Merging Data

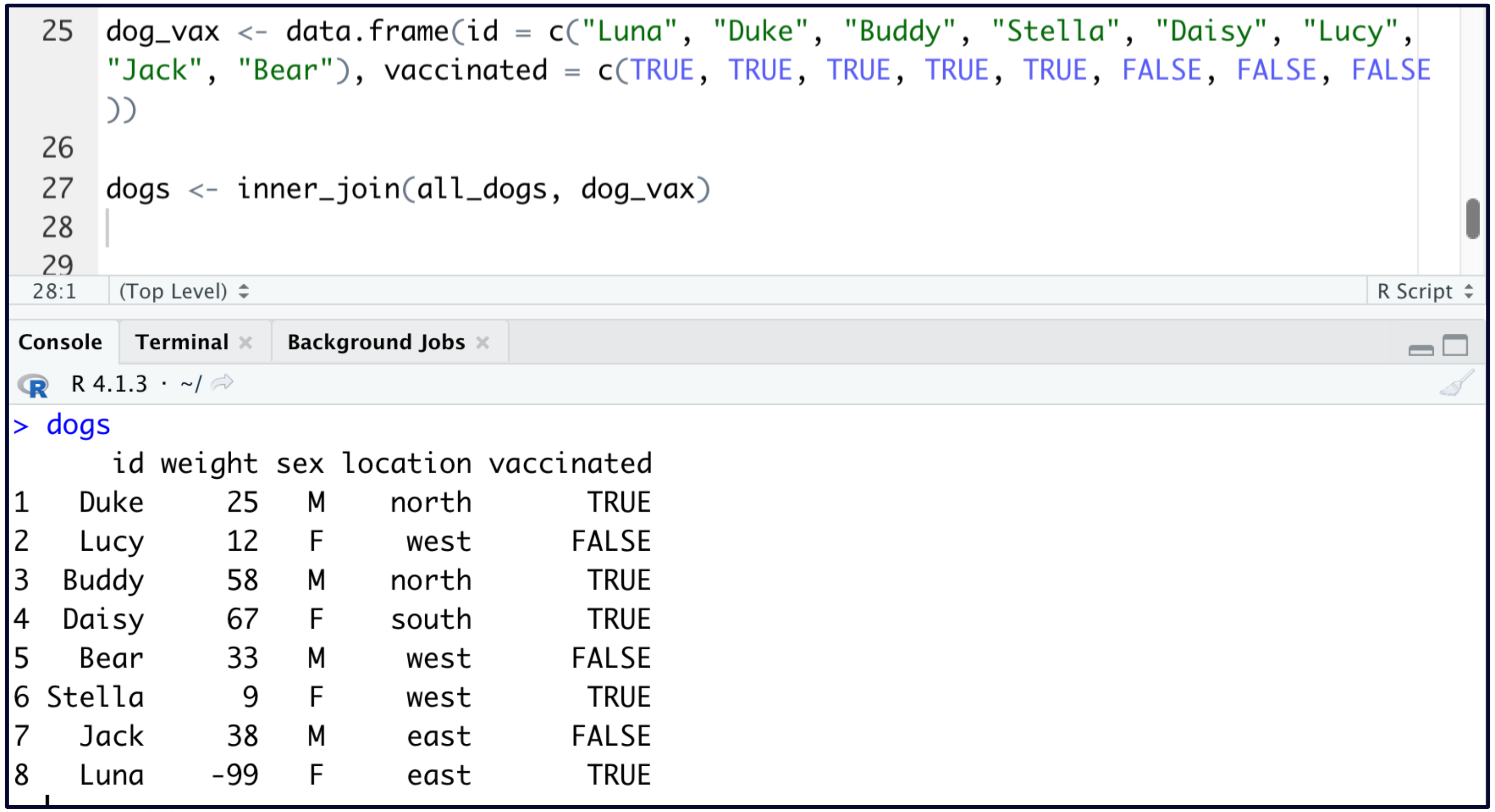
- We often <u>receive separate datasets with different variables (columns) that must be merged on a key variable</u>

- Many different kinds of merges are possible, depending on whether every observation in one dataset can be matched to an observation in the other dataset. Sometimes, you'll want to keep observations in one dataset, even if it is not matched. Other times, you will not

- We will use the `dplyr` function `inner_join()` to perform the merge
  The base R function `merge()` can be used for the same type of merge

- `inner_join()` will search both datasets for any variables with the same name, and will use those as matching variables. If you need to control which variables are used to match, use the `by=` argument

Merging Data

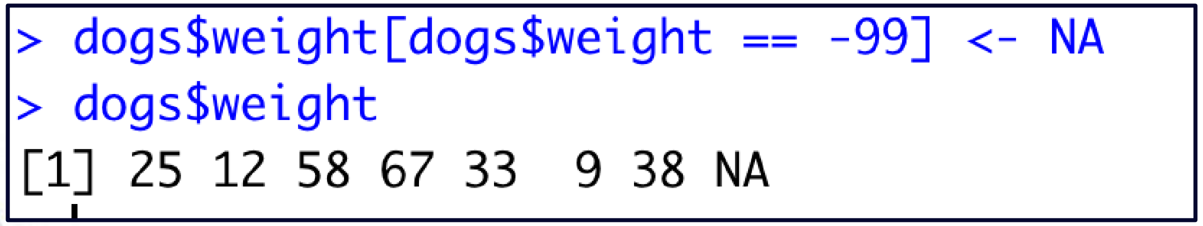


```
dog_vax <- data.frame(id = c("Luna", "Duke", "Buddy", "Stella", "Daisy", "Lucy",
"Jack", "Bear"), vaccinated = c(TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE
))

dogs <- inner_join(all_dogs, dog_vax)
```

```
> dogs
      id weight sex location vaccinated
1   Duke     25   M    north       TRUE
2   Lucy     12   F     west      FALSE
3  Buddy     58   M    north       TRUE
4  Daisy     67   F    south       TRUE
5   Bear     33   M     west      FALSE
6 Stella      9   F     west       TRUE
7   Jack     38   M     east      FALSE
8   Luna    -99   F     east       TRUE
```

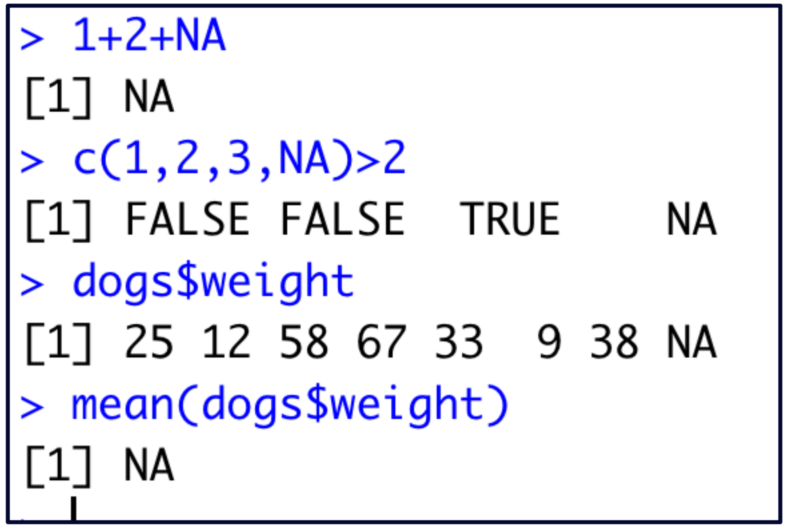
Missing Values

- Missing values in R are represented by the reserved symbol `NA` (cannot be used for variable names)
- Blank fields in a text file will generally be converted to `NA` when loaded into R
- Often, datasets use codes, such as impossible numeric values (e.g. `-99`) to denote missing values
- We can convert missing data codes like `-99` in variables to `NA` with conditional selection

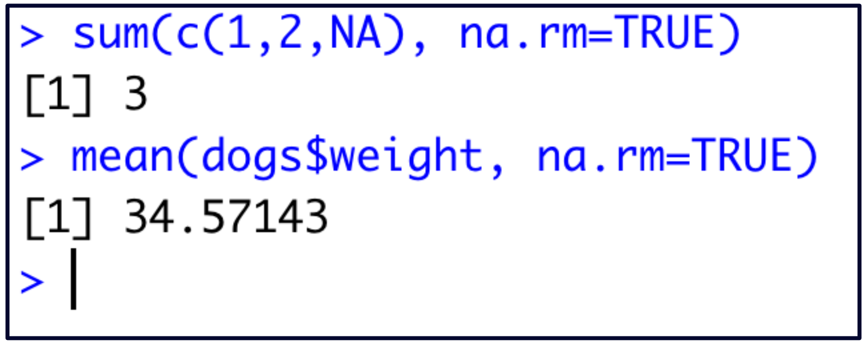


```
> dogs$weight[dogs$weight == -99] <- NA
> dogs$weight
[1] 25 12 58 67 33  9 38 NA
```

Dealing with Missing Values



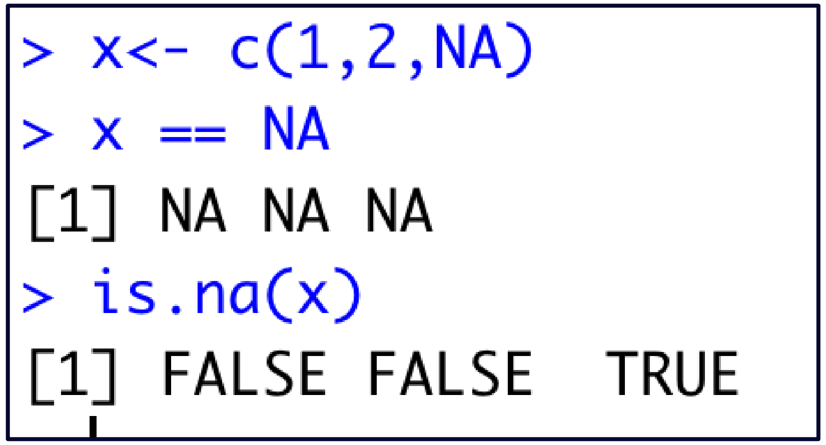
```
> 1+2+NA
[1] NA
> c(1,2,3,NA)>2
[1] FALSE FALSE  TRUE    NA
> dogs$weight
[1] 25 12 58 67 33  9 38 NA
> mean(dogs$weight)
[1] NA
> 
```



```
> sum(c(1,2,NA), na.rm=TRUE)
[1] 3
> mean(dogs$weight, na.rm=TRUE)
[1] 34.57143
> 
```

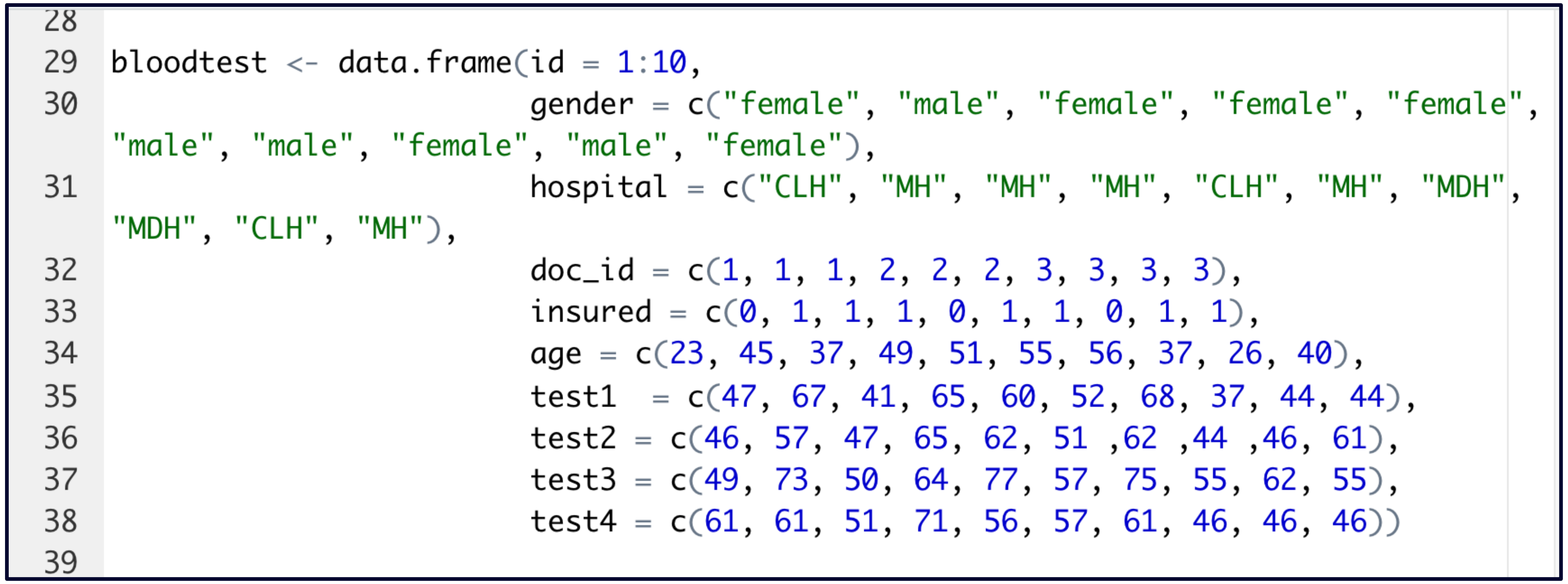
is.na()

- You cannot check for equality to **`NA`** because means "undefined". It will always result in **`NA`**
- Use **`is.na()`** instead.



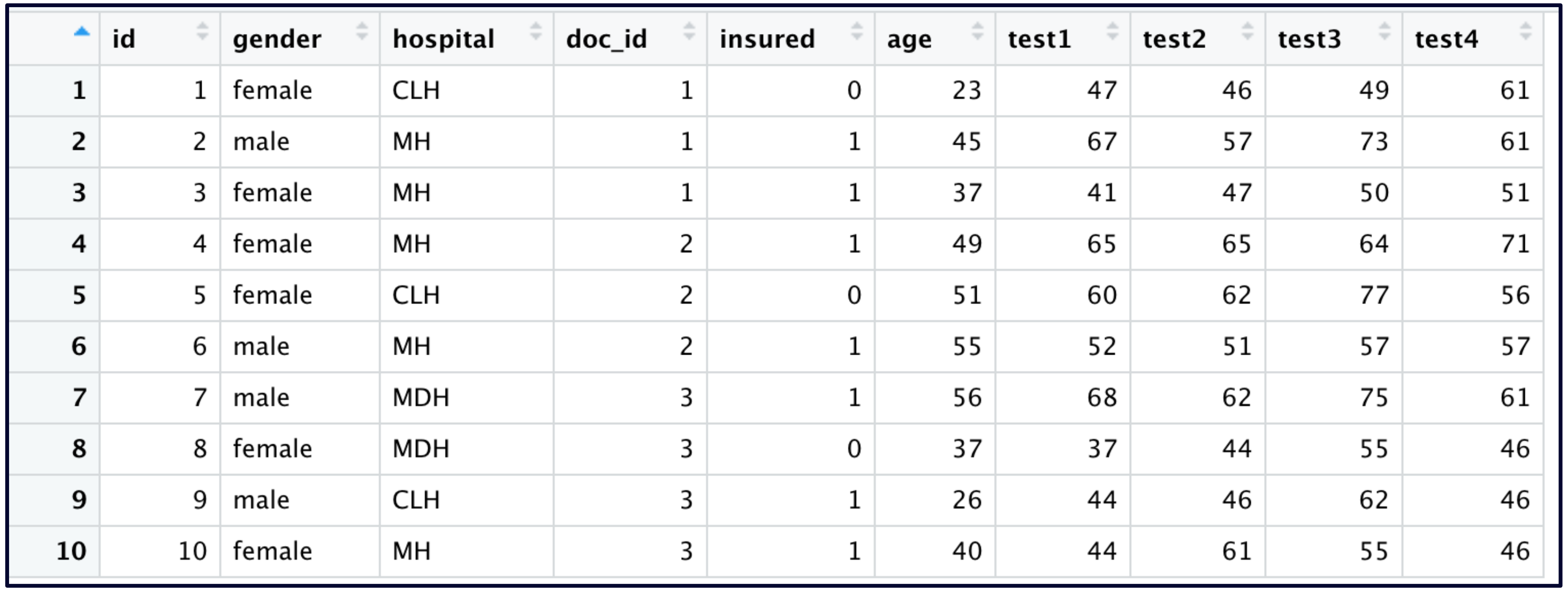
```
> x<- c(1,2,NA)
> x == NA
[1] NA NA NA
> is.na(x)
[1] FALSE FALSE  TRUE
```

Basic Data Analysis

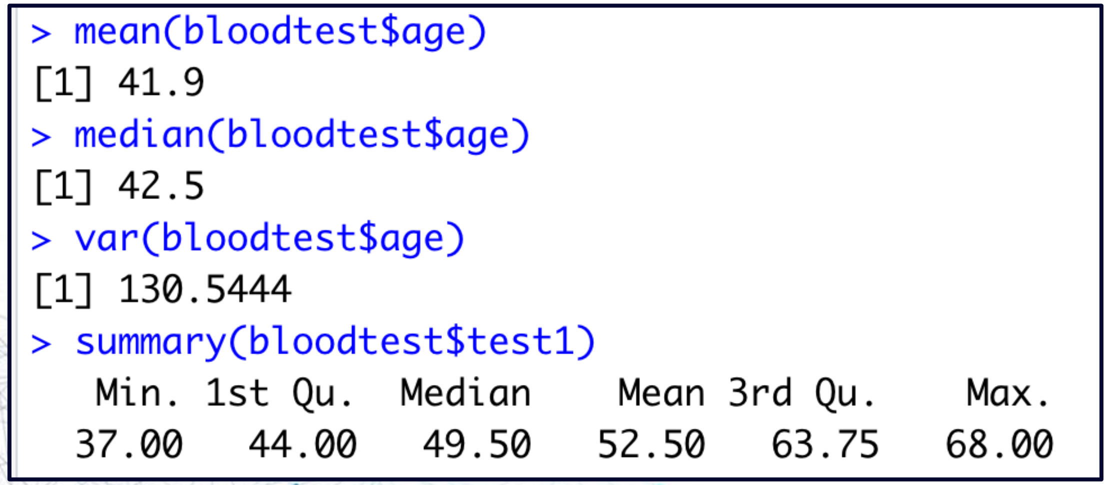


```r
bloodtest <- data.frame(id = 1:10,
                        gender = c("female", "male", "female", "female", "female",
"male", "male", "female", "male", "female"),
                        hospital = c("CLH", "MH", "MH", "MH", "CLH", "MH", "MDH",
"MDH", "CLH", "MH"),
                        doc_id = c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3),
                        insured = c(0, 1, 1, 1, 0, 1, 1, 0, 1, 1),
                        age = c(23, 45, 37, 49, 51, 55, 56, 37, 26, 40),
                        test1  = c(47, 67, 41, 65, 60, 52, 68, 37, 44, 44),
                        test2 = c(46, 57, 47, 65, 62, 51 ,62 ,44 ,46, 61),
                        test3 = c(49, 73, 50, 64, 77, 57, 75, 55, 62, 55),
                        test4 = c(61, 61, 51, 71, 56, 57, 61, 46, 46, 46))
```

Descriptive Statistics



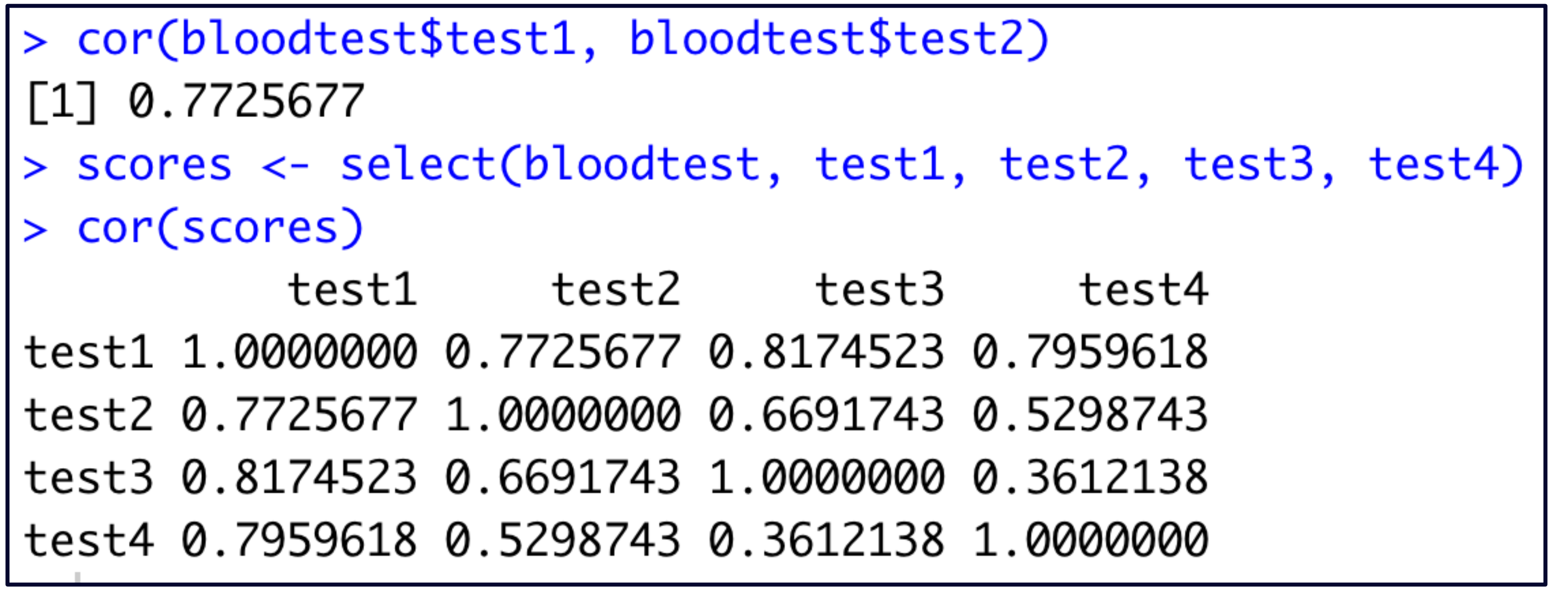
	id	gender	hospital	doc_id	insured	age	test1	test2	test3	test4
1	1	female	CLH	1	0	23	47	46	49	61
2	2	male	MH	1	1	45	67	57	73	61
3	3	female	MH	1	1	37	41	47	50	51
4	4	female	MH	2	1	49	65	65	64	71
5	5	female	CLH	2	0	51	60	62	77	56
6	6	male	MH	2	1	55	52	51	57	57
7	7	male	MDH	3	1	56	68	62	75	61
8	8	female	MDH	3	0	37	37	44	55	46
9	9	male	CLH	3	1	26	44	46	62	46
10	10	female	MH	3	1	40	44	61	55	46



```
> mean(bloodtest$age)
[1] 41.9
> median(bloodtest$age)
[1] 42.5
> var(bloodtest$age)
[1] 130.5444
> summary(bloodtest$test1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  37.00   44.00   49.50   52.50   63.75   68.00 
```

Correlations

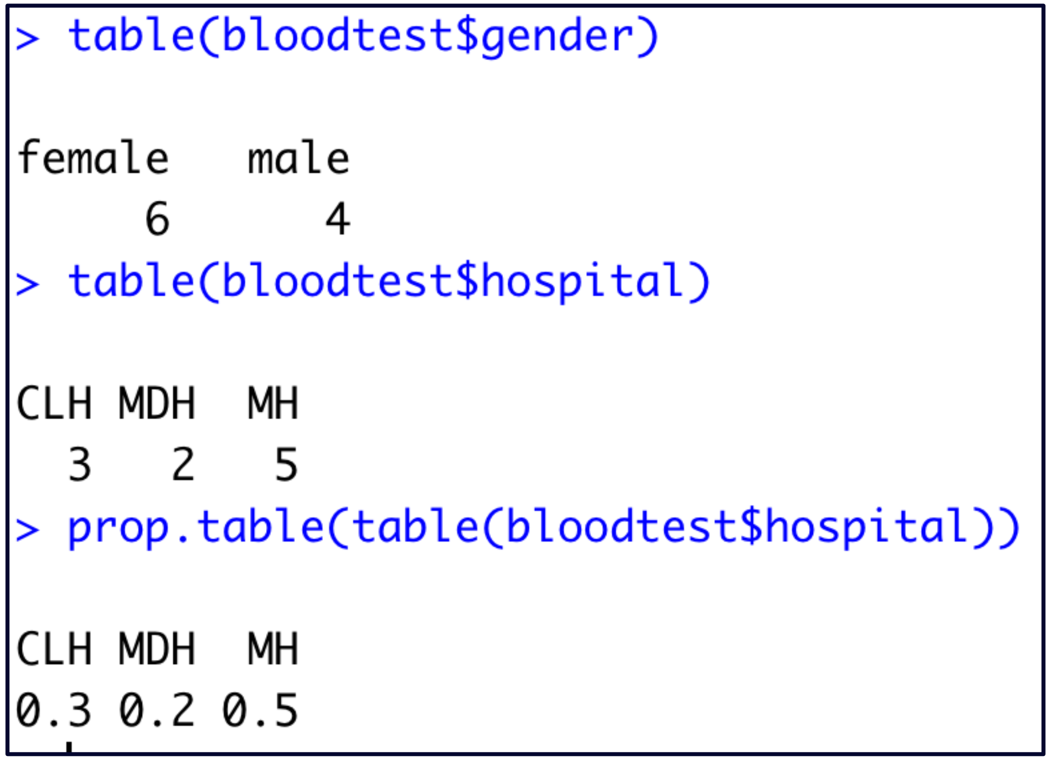
- Correlations provide quick assessments of whether two continuous variables are linearly related to one another



```
> cor(bloodtest$test1, bloodtest$test2)
[1] 0.7725677
> scores <- select(bloodtest, test1, test2, test3, test4)
> cor(scores)
          test1     test2     test3     test4
test1 1.0000000 0.7725677 0.8174523 0.7959618
test2 0.7725677 1.0000000 0.6691743 0.5298743
test3 0.8174523 0.6691743 1.0000000 0.3612138
test4 0.7959618 0.5298743 0.3612138 1.0000000
```

Frequency Table

- The statistics mean, median and variance cannot be calculated meaningfully for categorical variables (unless just 2 categories)
- Instead, we often present frequency tables of the distribution of membership to each category
- Use `table()` to produce frequency tables
- Use `prop.table()` on the tables produced by `table()` (i.e. the output) to see the frequencies expressed as proportions



```
> table(bloodtest$gender)

female   male
     6      4
> table(bloodtest$hospital)

CLH MDH  MH
  3   2   5
> prop.table(table(bloodtest$hospital))

CLH MDH  MH
0.3 0.2 0.5
```

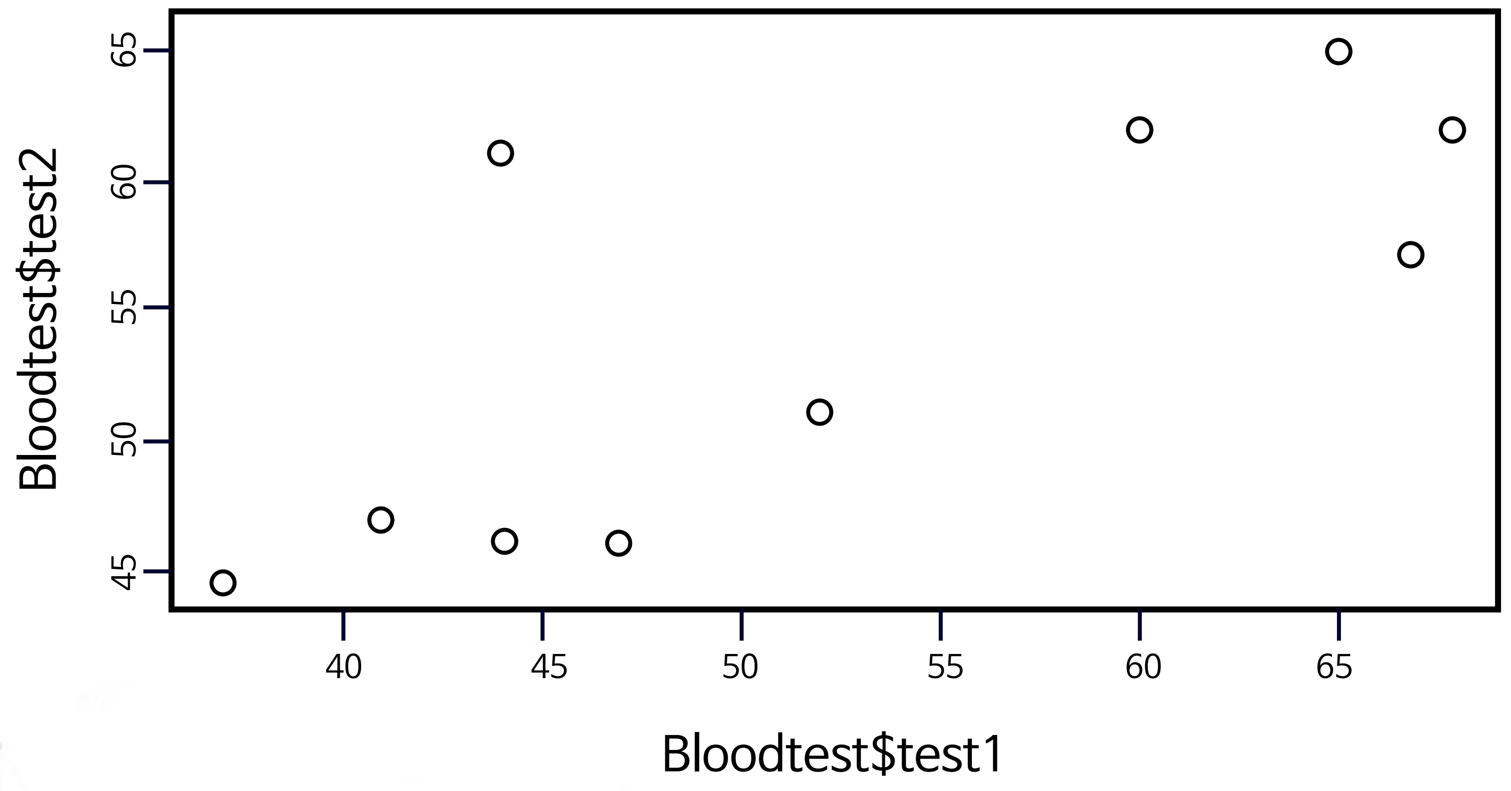
Linear Regression

- The `lm()` function is used to fit linear regression models
- Numeric and character variable predictors are acceptable
  Character variables are essentially treated as factors (categorical variables), where by default, a dummy (0/1) variable is entered into the model for each level except for the first

```
> m1 <- lm(test1 ~ age + gender, data=bloodtest)
> m1

Call:
lm(formula = test1 ~ age + gender, data = bloodtest)

Coefficients:
(Intercept)          age   gendermale
    24.4871       0.6206       5.0265
```

Scatter Plots

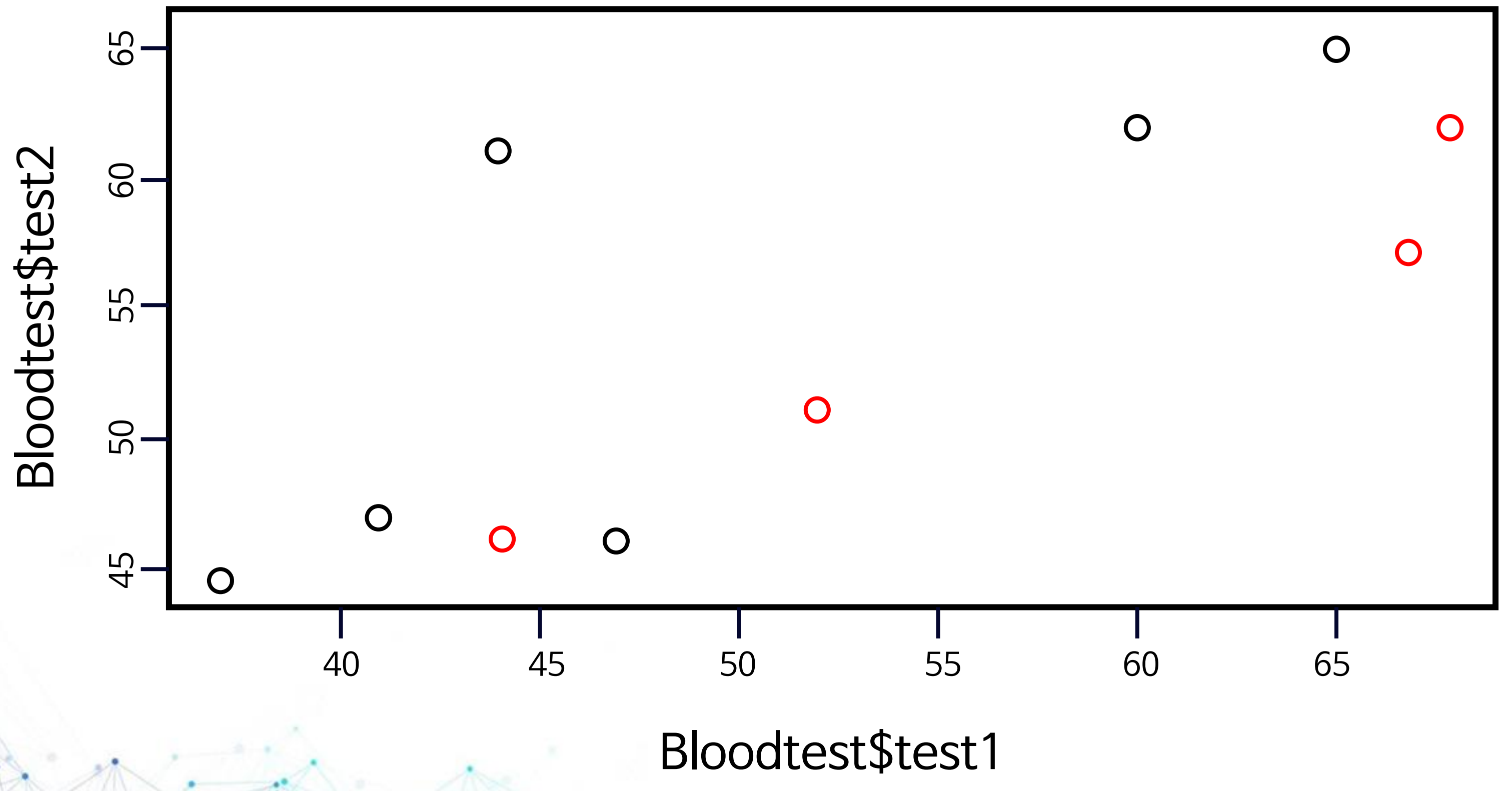
- `plot(bloodtest$test1,bloodtest$test2)`



Change Appearance by Grouping

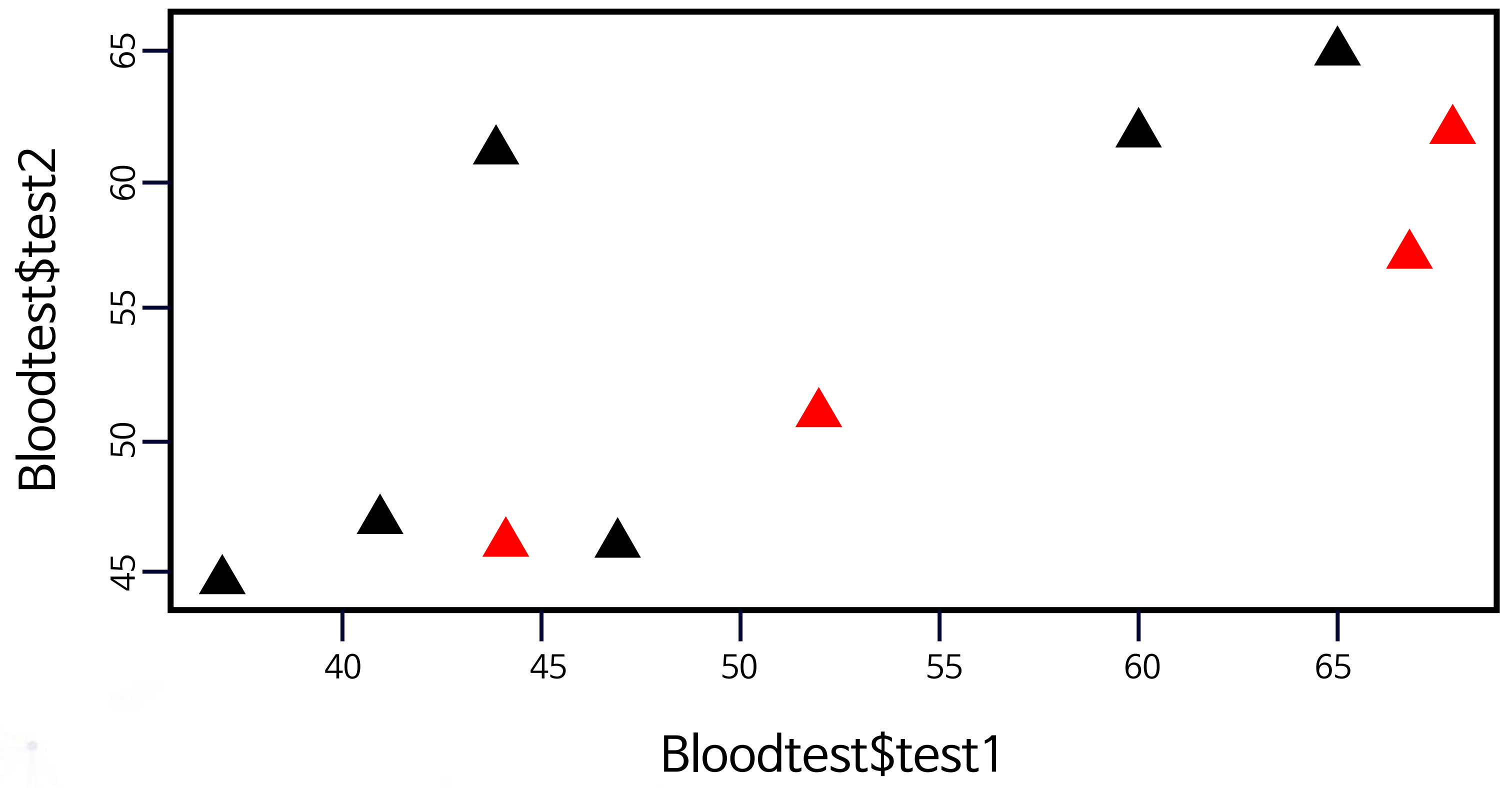
- Grouping variable must be made into a factor first

```
> bloodtest$gender <- factor(bloodtest$gender)
> plot(bloodtest$test1, bloodtest$test2, col=bloodtest$gender)
```



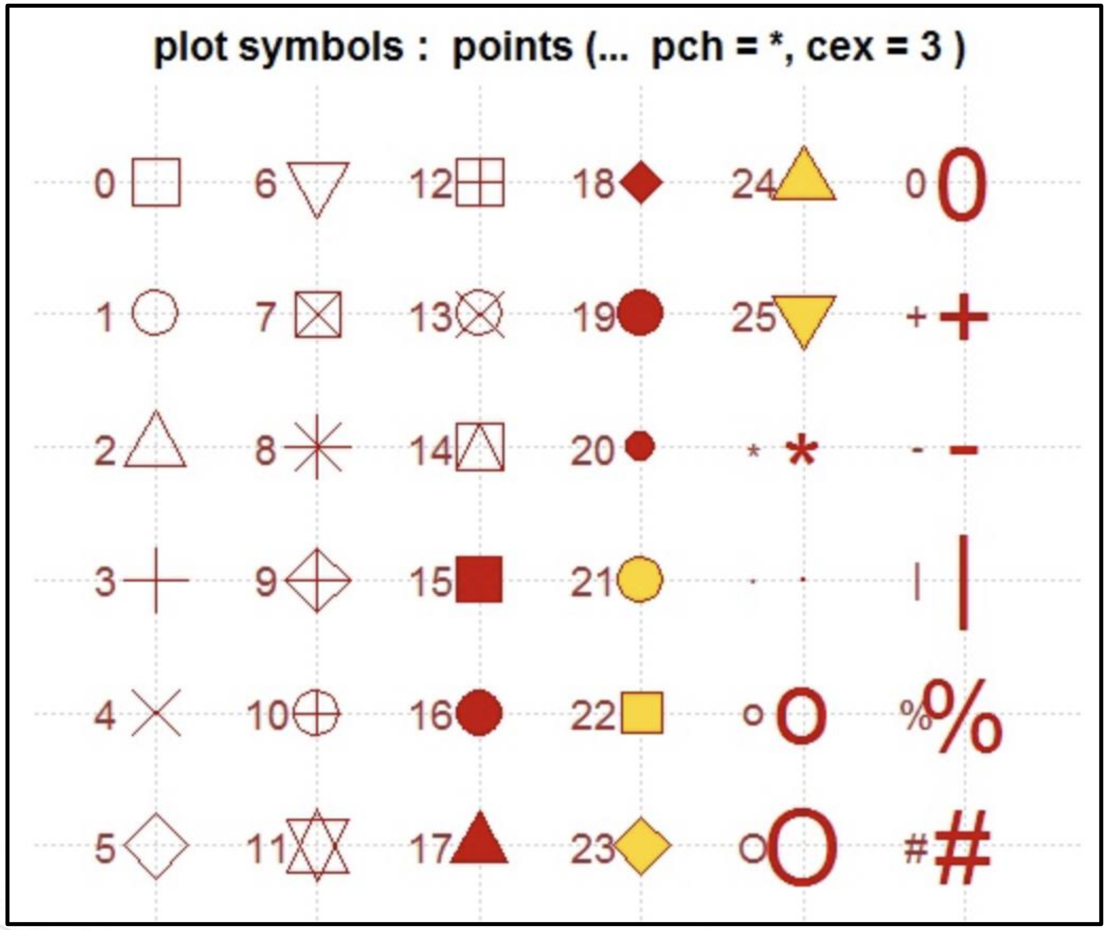
Change the Plotting Symbol

- `plot(bloodtest$test1,bloodtest$test2,col=bloodtest$gender,pch=17)`



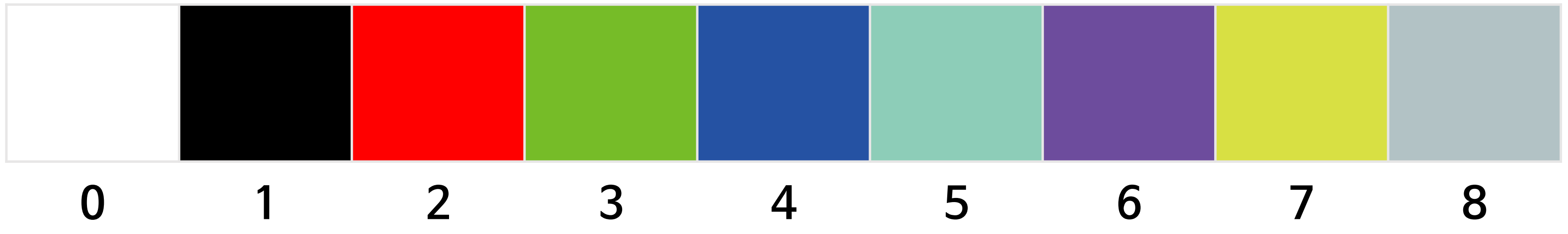
Data Management

- **pch =**



출처 : https://rfriend.tistory.com/149

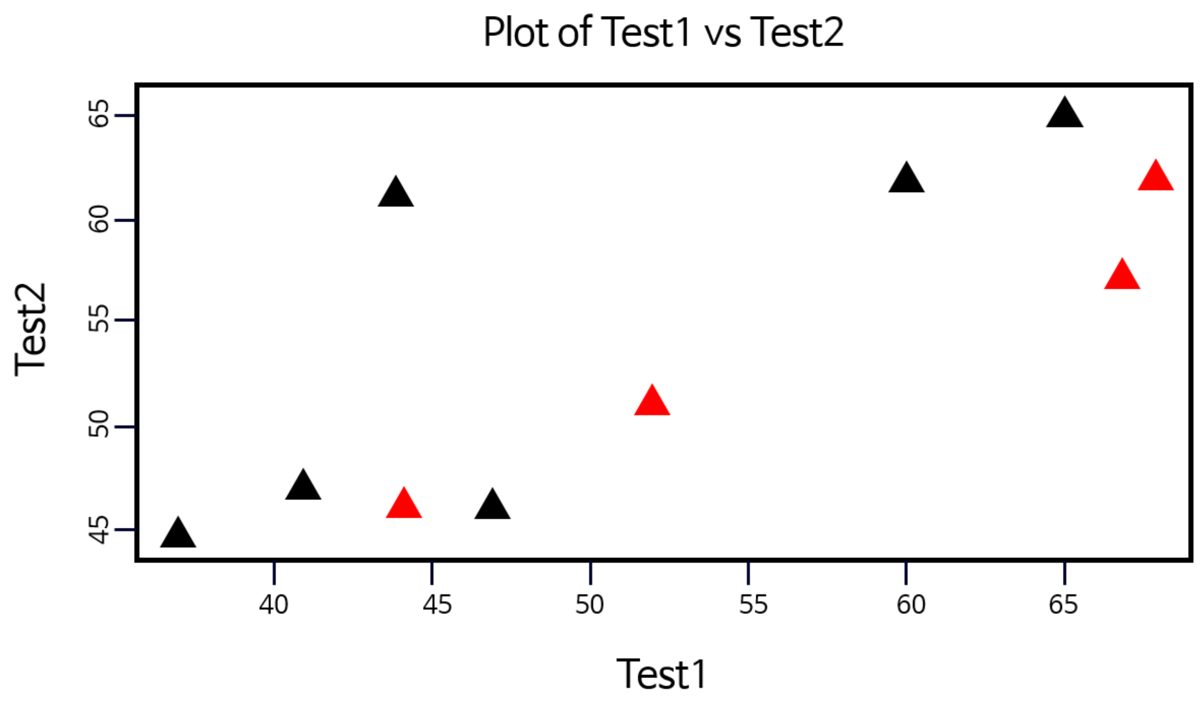
Color number



```
Set.seed(100)
z <-  sample(1:4,100,TRUE)
x <-  rnorm(100)
y <-  rnorm(100)
Plot(x,y,pch=15)
Plot(x,y,pch=15, col="red")
Plot(x,y,pch=15, col=3)
```

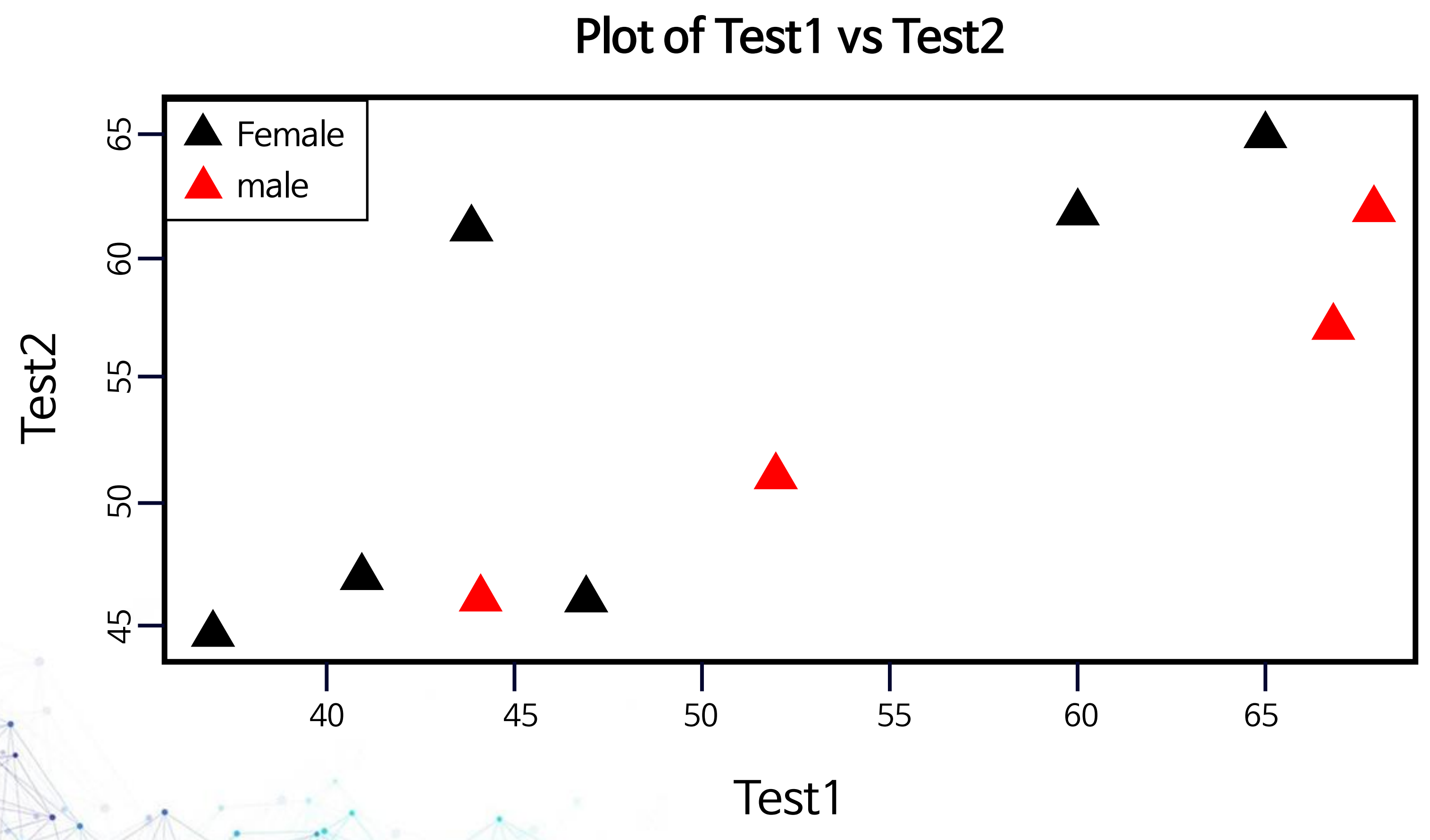
Changing the Axis

```
plot(bloodtest$test1,
bloodtest$test2,
      col=bloodtest$gender,
      pch=17,
      xlab="Test 1",
      ylab="Test 2",
      main="Plot of Test1 vs
Test2")
```



Add a Legend

```
> # specifies placement, labels, color, and symbol in legend box
> legend("topleft", legend=levels(bloodtest$gender), col=c(1:2), pch=17)
```



Statistical Graphs

Try followings yourself

- `hist(bloodtest$test1)`
- `hist(bloodtest$test1, breaks=2)`
- `boxplot(bloodtest$test2 ~ bloodtest$insured)`
- `boxplot(bloodtest$test2 ~ bloodtest$insured, xlab="Insured", ylab="Test 2", main = "Boxplots of Test2 by Insurance Status", col="lightblue")`
- `tab <- table(bloodtest$gender, bloodtest$hospital)`
  - `barplot(tab)`
  - `barplot(tab, legend.text = TRUE)`

Statistical Graphs

- Try followings yourself

```
- barplot(tab,
          legend.text = TRUE,
          beside=TRUE,
          col=c("lawngreen", "sandybrown"),
          xlab="Hospital",
          ylab="Frequency",
          main="Frequencies of gender by hospital")
```

General Graphics

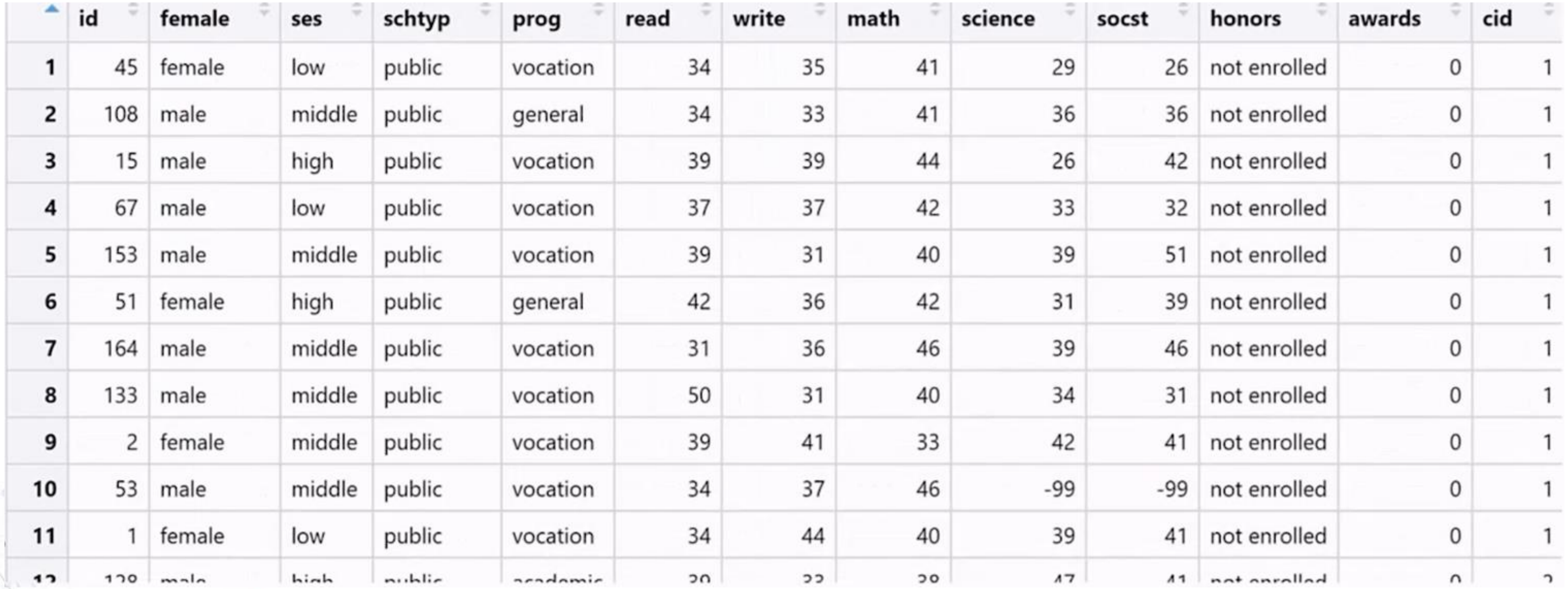
- For general purpose for the graphics, we use ggplot2 package
- The basic specification for a ggplot2 plot is to specify which variables are mapped to which aspects of the graph (called aesthetics) and then to choose a shape (called a geom) to display on the graph
- we can produce many plots with some variation of the following syntax

```
- ggplot(dataset, aes(x=xvar, y=yvar)) + geom_function()
```

General Graphics

- Again, we use the following data:

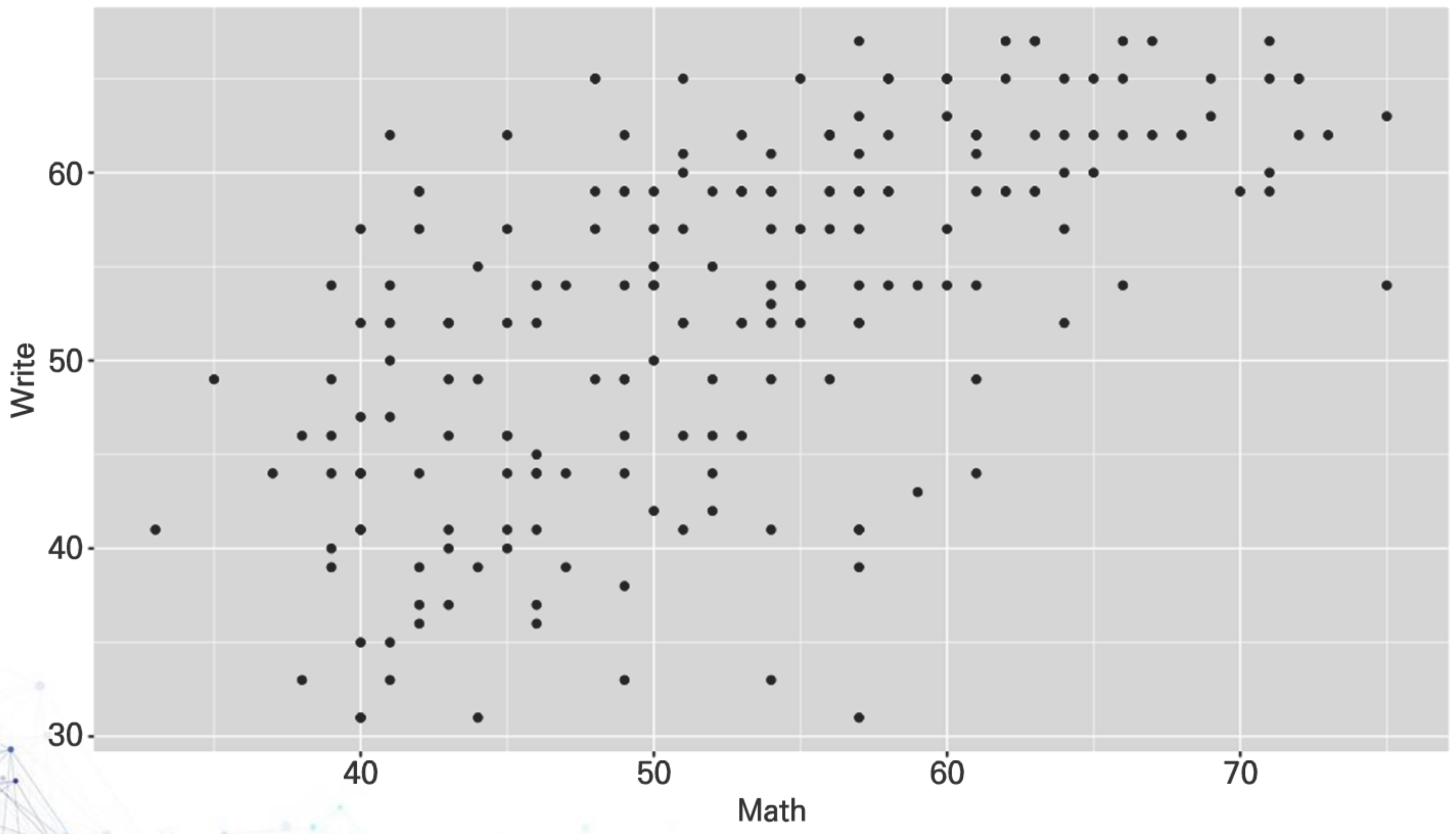
```
Dat_csv <-
Read.csv(“http://stats.idre.ucla.edu/stat/data/hsbraw.csv”)
```



	id	female	ses	schtyp	prog	read	write	math	science	socst	honors	awards	cid
1	45	female	low	public	vocation	34	35	41	29	26	not enrolled	0	1
2	108	male	middle	public	general	34	33	41	36	36	not enrolled	0	1
3	15	male	high	public	vocation	39	39	44	26	42	not enrolled	0	1
4	67	male	low	public	vocation	37	37	42	33	32	not enrolled	0	1
5	153	male	middle	public	vocation	39	31	40	39	51	not enrolled	0	1
6	51	female	high	public	general	42	36	42	31	39	not enrolled	0	1
7	164	male	middle	public	vocation	31	36	46	39	46	not enrolled	0	1
8	133	male	middle	public	vocation	50	31	40	34	31	not enrolled	0	1
9	2	female	middle	public	vocation	39	41	33	42	41	not enrolled	0	1
10	53	male	middle	public	vocation	34	37	46	-99	-99	not enrolled	0	1
11	1	female	low	public	vocation	34	44	40	39	41	not enrolled	0	1
12	128	male	high	public	academic	39	33	38	47	41	not enrolled	0	2

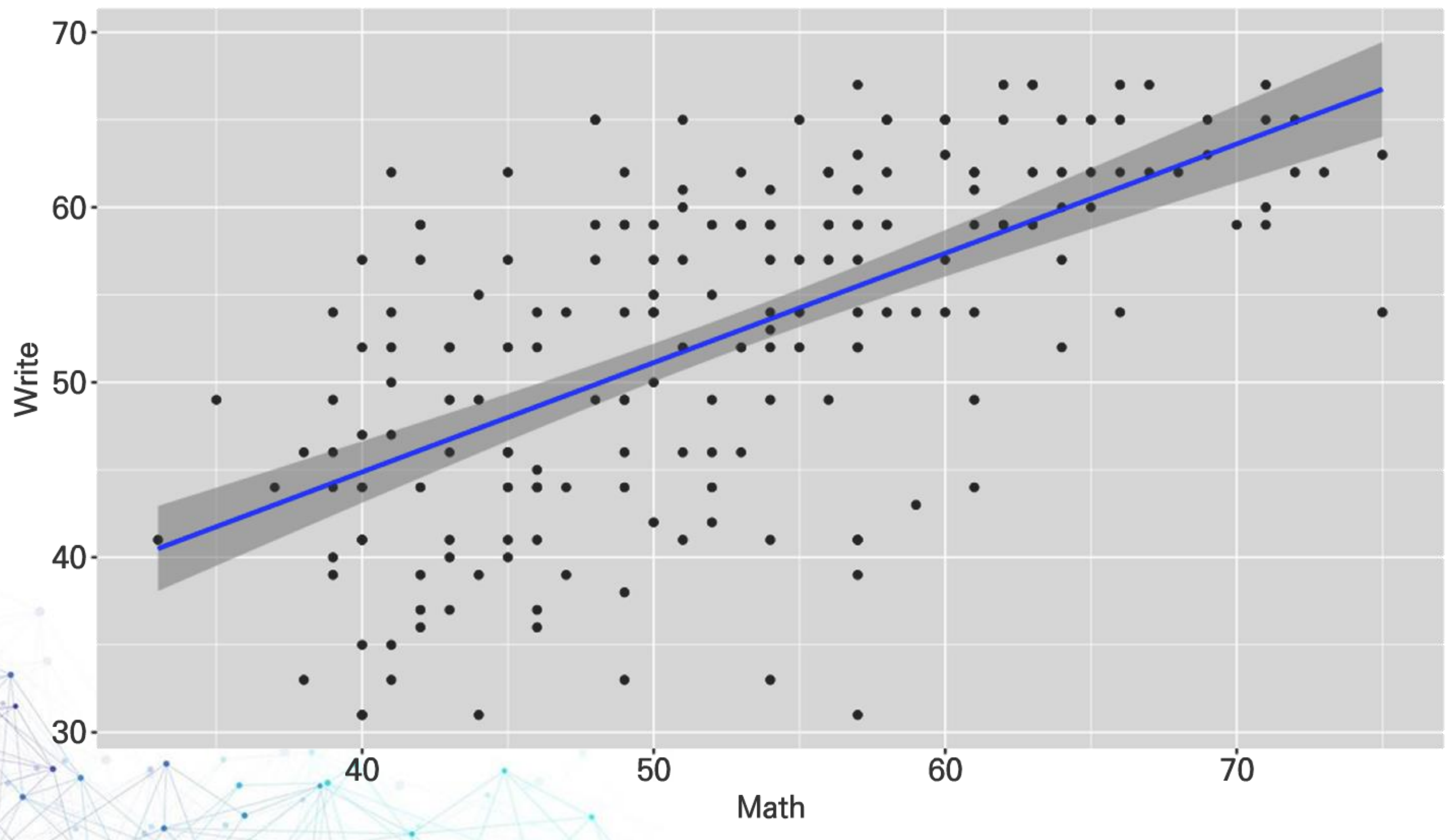
Example

```
> ggplot(data=dat_csv, aes(x=math, y=write)) + geom_point()
```



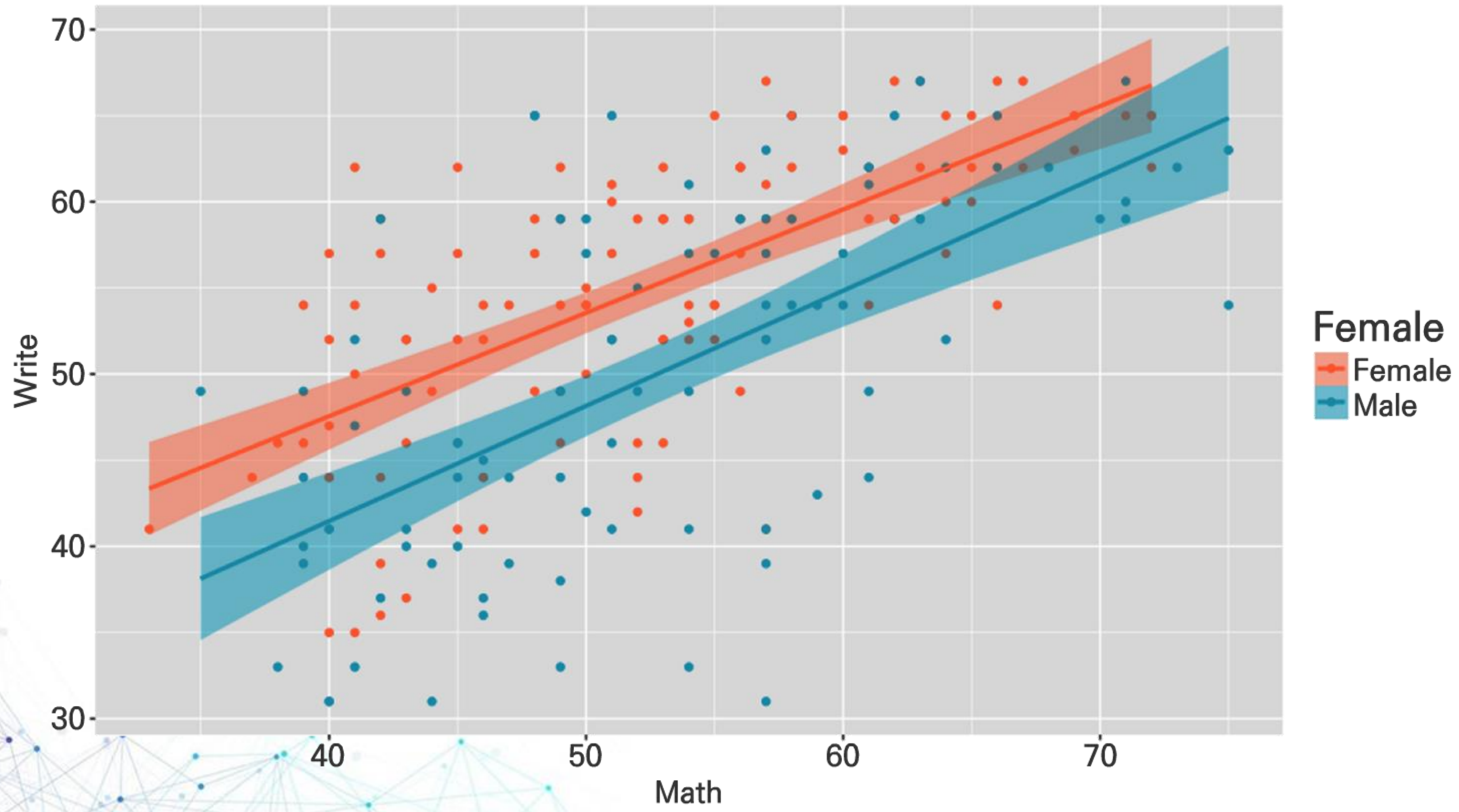
Example

```
> # a scatterplot of math vs write with best fit line
> ggplot(dat_csv, aes(x=math, y=write)) + geom_point() + geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'
```



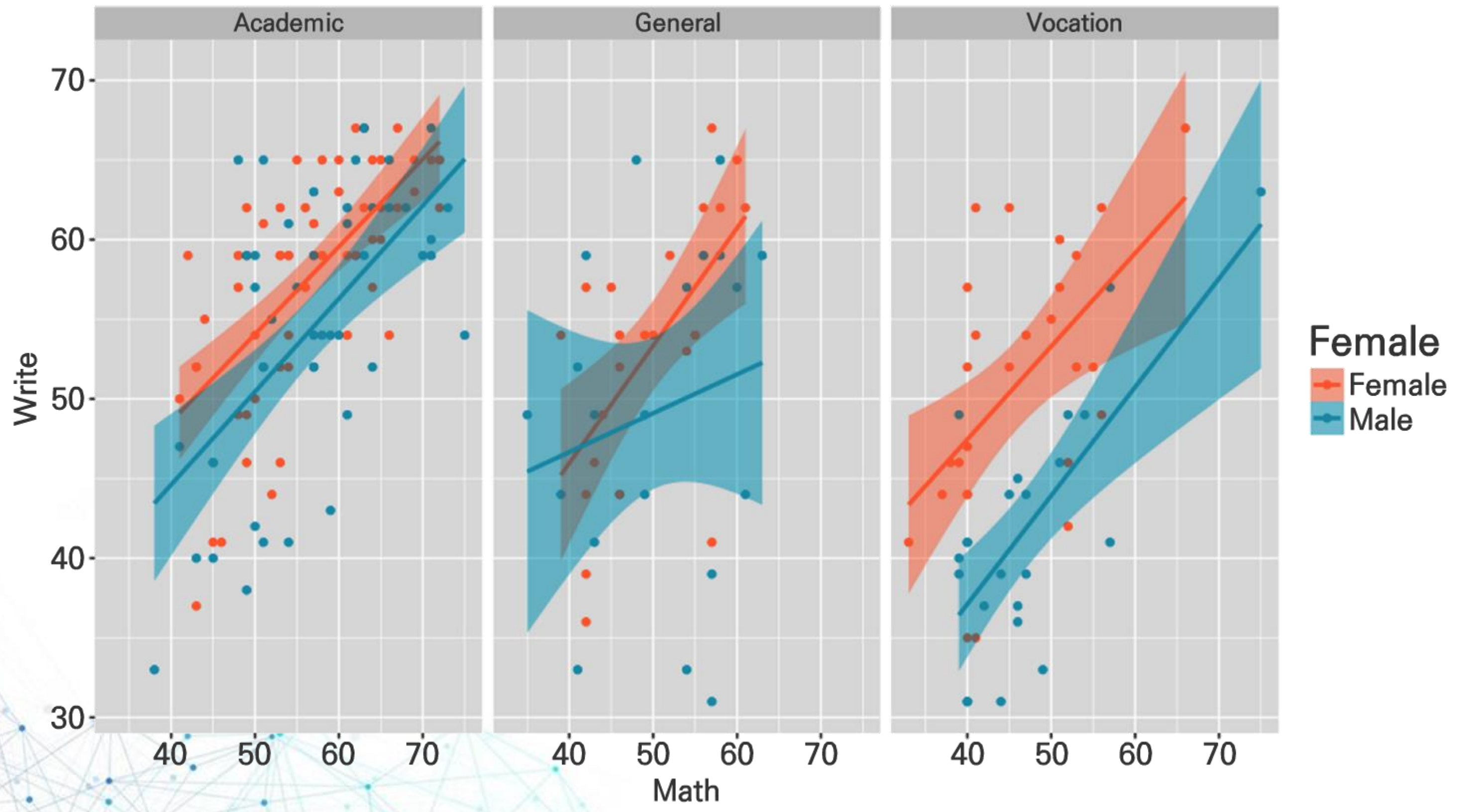
Example

```
> ggplot(dat_csv, aes(x=math, y=write, color=female, fill=female)) +
+   geom_point() +
+   geom_smooth(method="lm")
`geom_smooth()` using formula 'y ~ x'
```



Example

```
> ggplot(dat_csv, aes(x=math, y=write, color=female, fill=female)) +
+   geom_point() +
+   geom_smooth(method="lm") +
+   facet_wrap(~prog)
`geom_smooth()` using formula 'y ~ x'
```


Academic
General
Vocation
Write
Math
Female
Female
Male