

Introduction to R - session 2

Jee-Young Moon

Oct-29-2024

Warm-up 1: mean and standard deviation.

Create a vector called “test.score” and calculate the mean and standard deviation.

```
#####  
## Warm-up 1: mean and standard deviation  
#####  
## Create a vector, called test.score, with the values  
test.score <- c(80, 90, 70, 30, 40, 80, 75)  
  
## Use "mean()" function to get the mean  
mean(test.score)
```

```
## [1] 66.42857
```

```
## Use "sd()" function to get the standard deviation (SD)  
sd(test.score)
```

```
## [1] 22.49339
```

Warm-up 2: t-test

Create another vector called “test.score2” for the test scores of another group, and perform a t-test.

```
#####  
## Warm-up 2: t-test  
#####  
## Create another vector for another group  
test.score2 <- c(55, 80, 60, 90, 60, 90, 95)  
  
## Use "t.test()" function to perform a t-test  
t.test(test.score, test.score2)
```

```
##  
## Welch Two Sample t-test  
##  
## data: test.score and test.score2  
## t = -0.8725, df = 11.149, p-value = 0.4013
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -32.67182  14.10039
## sample estimates:
## mean of x mean of y
##  66.42857  75.71429
```

Getting help

```
## Getting help
?t.test
```

Data analysis example 1: Eye Bank (slide 9).

We will use Eye Bank data to learn to 1) read the data, 2) retrieve a variable (column), and 3) get basic statistics, and 4) perform a t-test and a Chi-squared test.

First, let's read the data into "dat".

```
#####
## Data analysis example 1: Eye Bank (slide 9)
#####
## Set the working directory
setwd('/Users/moon/EinsteinMed Dropbox/Jee Young Moon/class/R workshop/2024-Melissa')

## Read the data
library(readxl)
dat <- read_excel('EyeBank.xlsx', sheet=1)
dim(dat)
```

```
## [1] 2942    6
```

```
head(dat)
```

```
## # A tibble: 6 x 6
##   id age sex cellcount accepted.transplant diabetes
##   <dbl> <dbl> <chr>    <dbl> <chr>          <chr>
## 1     1   80 F      1403 No            No
## 2     2   80 M      2469 Yes          <NA>
## 3     3   80 F      2053 No            No
## 4     4   80 M      2618 Yes          Yes
## 5     5   80 M      3215 No            Yes
## 6     6   80 F      2037 Yes          No
```

```
tail(dat)
```

```
## # A tibble: 6 x 6
##   id age sex cellcount accepted.transplant diabetes
##   <dbl> <dbl> <chr>    <dbl> <chr>          <chr>
## 1 2937   3 M      3344 Yes            No
## 2 2938   3 F      4255 No            <NA>
## 3 2939   3 M      4367 No            No
## 4 2940   3 M      4255 No            No
## 5 2941   2 F      4329 Yes            No
## 6 2942   2 M      3704 Yes            No
```

```
str(dat)
```

```
## tibble [2,942 x 6] (S3: tbl_df/tbl/data.frame)
## $ id      : num [1:2942] 1 2 3 4 5 6 7 8 9 10 ...
## $ age     : num [1:2942] 80 80 80 80 80 80 80 80 80 80 ...
## $ sex     : chr [1:2942] "F" "M" "F" "M" ...
## $ cellcount : num [1:2942] 1403 2469 2053 2618 3215 ...
## $ accepted.transplant: chr [1:2942] "No" "Yes" "No" "Yes" ...
## $ diabetes  : chr [1:2942] "No" NA "No" "Yes" ...
```

```
summary(dat)
```

```
##           id           age           sex           cellcount
## Min.      : 1.0    Min.    : 2.00    Length:2942    Min.      : 696
## 1st Qu.: 736.2    1st Qu.:56.00    Class :character 1st Qu.:2273
## Median :1471.5    Median :65.00    Mode  :character Median :2571
## Mean     :1471.5    Mean     :61.26                    Mean     :2525
## 3rd Qu.:2206.8    3rd Qu.:71.00                    3rd Qu.:2841
## Max.     :2942.0    Max.      :80.00                    Max.     :4367
##                                           NA's      :103
## accepted.transplant  diabetes
## Length:2942          Length:2942
## Class :character     Class :character
## Mode  :character     Mode  :character
##
##
##
##
```

You can access the cellcount variable by using “\$” sign

```
dat$cellcount
```

How many missing in the cellcount variable?

```
is.na(dat$cellcount)
sum(is.na(dat$cellcount))
```

You can access the age variable in the data by using “\$” sign.

```
summary(dat$age)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      2.00  56.00   65.00   61.26  71.00   80.00
```

You can create a new variable “age.group” from “age” grouped by ≥ 76 and < 76 .

```
## Create an age group ( $\geq 76$ ,  $< 76$ )
x <- dat$age  $\geq 76$ 

## age  $\geq 76$  to call 'old', and age  $< 76$  to call young
dat$age.group <- ifelse(dat$age $\geq 76$ , 'old', 'young')
```

I. Cell count analysis.

We will analyze the cell count (continuous) by age group.

- (a) We will summarize the cell count by mean and SD according to the age group.
- (b) T-test
- (c) Linear regression.

```
#####  
## I. Cell count analysis  
#####  
## a. Mean and SD of cell count in overall  
mean(dat$cellcount)
```

```
## [1] NA
```

As there is a missing value (NA) in cellcount, mean() function produces NA.

You can use “na.rm=T” option in mean() to calculate the mean among non-missing values.

```
mean(dat$cellcount, na.rm=T)
```

```
## [1] 2524.549
```

```
sd(dat$cellcount, na.rm=T)
```

```
## [1] 490.3624
```

To calculate the mean and SD of cell count by age group, first, you can create a subset of the data.

```
## Mean and SD of cell count by age group  
## Make a subset of the data of old group  
## You have to use double equal signs (==) when you are asking if the value is equal to something.  
## This is because one equal sign ("=") is used to assign the value, similar to "<-"  
dat.old <- subset(dat, age.group == 'old')  
mean(dat.old$cellcount, na.rm=T)
```

```
## [1] 2222.416
```

```
sd(dat.old$cellcount, na.rm=T)
```

```
## [1] 546.5434
```

Equivalently, you can use subset() and mean() functions together in one R command.

```
mean(subset(dat, age.group == 'old')$cellcount, na.rm=T)
```

```
## [1] 2222.416
```

```
mean(subset(dat, age.group == 'old')$cellcount, na.rm=T)
```

```
## [1] 2222.416
```

Small practice: mean and SD of cell count for young group

T-test

As was done in Warm-up II, you can use `t.test()` with two groups' cell counts as input.

```
## b. t-test on cell count by age group
dat.young <- subset(dat, age.group == 'young')

t.test(dat.old$cellcount, dat.young$cellcount)
```

```
##
## Welch Two Sample t-test
##
## data: dat.old$cellcount and dat.young$cellcount
## t = -8.2212, df = 225.2, p-value = 1.595e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -403.2436 -247.3111
## sample estimates:
## mean of x mean of y
## 2222.416 2547.693
```

A formula form makes it easier to use `t.test()`. See slide 13 on formula.

```
## A simpler way using a formula!
t.test(cellcount ~ age.group, dat)
```

```
##
## Welch Two Sample t-test
##
## data: cellcount by age.group
## t = -8.2212, df = 225.2, p-value = 1.595e-14
## alternative hypothesis: true difference in means between group old and group young is not equal to 0
## 95 percent confidence interval:
## -403.2436 -247.3111
## sample estimates:
## mean in group old mean in group young
## 2222.416 2547.693
```

Further, you can fit a linear regression using a formula.

`summary()` provides more information on the fitted regression (`fit1`) such as standard error and p-value.

```
## c. Linear regression
lm(cellcount ~ age.group, dat)
```

```
##
## Call:
## lm(formula = cellcount ~ age.group, data = dat)
##
## Coefficients:
##      (Intercept)  age.groupyoung
##           2222.4           325.3
```

```
fit1 <- lm(cellcount ~ age.group, dat)
fit1
```

```
##
## Call:
## lm(formula = cellcount ~ age.group, data = dat)
##
## Coefficients:
##      (Intercept)  age.groupyoung
##           2222.4           325.3
```

```
summary(fit1)
```

```
##
## Call:
## lm(formula = cellcount ~ age.group, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1851.69  -243.69   43.31   309.31  1819.31
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2222.42     34.00   65.36 <2e-16 ***
## age.groupyoung    325.28     35.28    9.22 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 483.3 on 2837 degrees of freedom
## (103 observations deleted due to missingness)
## Multiple R-squared:  0.02909,    Adjusted R-squared:  0.02875
## F-statistic:    85 on 1 and 2837 DF,  p-value: < 2.2e-16
```

In R, `factor()` is often used on binary or categorical variables to give an order on their categories.

The first level of the category is used as a reference group.

```
## factor
dat$age.group <- factor(dat$age.group, levels=c('young', 'old'))
# Now, the reference group is the 'young' group.
fit2 <- lm(cellcount ~ age.group, dat)
summary(fit2)
```

```
##
## Call:
## lm(formula = cellcount ~ age.group, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1851.69  -243.69   43.31   309.31  1819.31
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2547.693      9.411   270.72  <2e-16 ***
## age.groupold -325.277     35.281   -9.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 483.3 on 2837 degrees of freedom
## (103 observations deleted due to missingness)
## Multiple R-squared:  0.02909,    Adjusted R-squared:  0.02875
## F-statistic:    85 on 1 and 2837 DF,  p-value: < 2.2e-16
```

You can include more predictors in the linear regression.

```
fit3 <- lm(cellcount ~ age.group + sex+diabetes, dat)
summary(fit3)
```

```
##
## Call:
## lm(formula = cellcount ~ age.group + sex + diabetes, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1819.15  -247.94   50.35   310.00  1784.70
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2544.30      15.82  160.816  < 2e-16 ***
## age.groupold -322.77     35.72   -9.037  < 2e-16 ***
## sexM          65.35      18.50    3.532 0.000419 ***
## diabetesYes  -94.50     19.07   -4.955 7.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 478.4 on 2764 degrees of freedom
## (174 observations deleted due to missingness)
## Multiple R-squared:  0.04204,    Adjusted R-squared:  0.041
## F-statistic: 40.43 on 3 and 2764 DF,  p-value: < 2.2e-16
```


II. Transplant acceptance.

We will analyze the transplant acceptance (binary) by age group.

- (a) We will summarize transplant acceptance by count and percentage according to age group.
- (b) Chi-square test, Fisher's exact test.
- (c) Logistic regression.

```
#####  
## II. Transplant acceptance  
#####  
## a. Get the count using table() function
```

```
table(dat$accepted.transplant)
```

```
##  
##    No   Yes  
##  929 2013
```

```
table(dat$diabetes)
```

```
##  
##    No   Yes  
## 1855 1000
```

If you want to know the number of NA (missing) values in diabetes,

```
table(dat$diabetes, useNA='ifany')
```

```
##  
##    No   Yes <NA>  
## 1855 1000   87
```

A cross-tabulation between accepted.transplant and age.group

```
table(dat[, c( 'accepted.transplant', 'age.group')])
```

```
##                age.group  
## accepted.transplant young  old  
##                No      827  102  
##                Yes     1907  106
```

```
tab <- table(dat[, c( 'accepted.transplant', 'age.group')])  
colSums(tab)
```

```
## young  old  
## 2734   208
```

```
rowSums(tab)
```

```
##    No  Yes  
##  929 2013
```

```
## a little advanced.. to get %  
tab/rep(colSums(tab), each=2)
```

```
##                age.group  
## accepted.transplant  young      old  
##                No  0.3024872 0.4903846  
##                Yes 0.6975128 0.5096154
```

```
## b. Chi-squared test  
chisq.test(tab)
```

```
##  
## Pearson's Chi-squared test with Yates' continuity correction  
##  
## data:  tab  
## X-squared = 30.722, df = 1, p-value = 2.978e-08
```

```
fisher.test(tab)
```

```
##  
## Fisher's Exact Test for Count Data  
##  
## data:  tab  
## p-value = 6.612e-08  
## alternative hypothesis: true odds ratio is not equal to 1  
## 95 percent confidence interval:  
##  0.3359471 0.6051543  
## sample estimates:  
## odds ratio  
##  0.4508012
```

```
## If you have cross-tabulated counts, you can create a matrix in this way.  
## By default, the first column is filled first, then second column, etc.  
mat <- matrix(c(827, 1907, 102, 106), ncol=2)  
mat
```

```
##      [,1] [,2]  
## [1,]  827 1907  
## [2,]  102  106
```

```
## If you want to fill the first row first, you can use byrow=T  
matrix(c(827, 1907, 102, 106), ncol=2, byrow=T)
```

```
##      [,1] [,2]  
## [1,]  827 1907  
## [2,]  102  106
```

```
chisq.test(mat)
```

```
##
## Pearson's Chi-squared test with Yates' continuity correction
##
## data:  mat
## X-squared = 30.722, df = 1, p-value = 2.978e-08
```

Let's fit a logistic regression as accepted.transplant is a binary variable.

A logistic regression can further adjust for other covariates.

```
## c. logistic regression
```

```
glm(accepted.transplant ~ age.group, dat, family=binomial('logit'))
```

```
## Error in eval(family$initialize): y values must be 0 <= y <= 1
```

```
dat$accepted.transplant <- factor(dat$accepted.transplant, levels=c('No', 'Yes'))
```

```
fit.transplant <- glm(accepted.transplant ~ age.group, dat, family=binomial('logit'))
summary(fit.transplant)
```

```
##
## Call:
## glm(formula = accepted.transplant ~ age.group, family = binomial("logit"),
##      data = dat)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.83548    0.04164  20.066 < 2e-16 ***
## age.groupold -0.79702    0.14482  -5.504 3.72e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 3669.5  on 2941  degrees of freedom
## Residual deviance: 3639.9  on 2940  degrees of freedom
## AIC: 3643.9
##
## Number of Fisher Scoring iterations: 4
```

```
fit.transplant3 <- glm(accepted.transplant ~ age.group+sex+ diabetes, dat, family=binomial('logit'))
```

Practice to fill the table 1 according to diabetes status.

```
#####
## Practice
## Fill the table in slide 14 by diabetes
#####
```

table1 and tableone package to expedite the Table 1 creation

Now, we learned how to get basic stats. We want to expedite this process over multiple variables.

table1() and CreateTableOne() functions give good Table 1.

Both functions provide summary statistics.

table1() does not produce p-values but shows how many missing values are; CreateTableone() produces p-values.

```
#####  
## table1 and tableone example  
#####  
## Create a table 1 using table1()  
library(table1)
```

```
##  
## Attaching package: 'table1'  
  
## The following objects are masked from 'package:base':  
##  
##      units, units<-
```

```
table1(~age+sex+cellcount+accepted.transplant+diabetes | age.group, dat)
```

```
## Get nicer 'table1' LaTeX output by simply installing the 'kableExtra' package
```

	young	old	Overall
	(N=2734)	(N=208)	(N=2942)
age			
Mean (SD)	60.0 (13.5)	78.0 (1.46)	61.3 (13.8)
Median [Min, Max]	64.0 [2.00, 75.0]	78.0 [76.0, 80.0]	65.0 [2.00, 80.0]
sex			
F	1119 (40.9%)	86 (41.3%)	1205 (41.0%)
M	1615 (59.1%)	122 (58.7%)	1737 (59.0%)
cellcount			
Mean (SD)	2550 (478)	2220 (547)	2520 (490)
Median [Min, Max]	2590 [696, 4370]	2330 [721, 3570]	2570 [696, 4370]
Missing	97 (3.5%)	6 (2.9%)	103 (3.5%)
accepted.transplant			
No	827 (30.2%)	102 (49.0%)	929 (31.6%)
Yes	1907 (69.8%)	106 (51.0%)	2013 (68.4%)
diabetes			
No	1737 (63.5%)	118 (56.7%)	1855 (63.1%)
Yes	919 (33.6%)	81 (38.9%)	1000 (34.0%)
Missing	78 (2.9%)	9 (4.3%)	87 (3.0%)

```
tab <- table1(~age+sex+cellcount+accepted.transplant+diabetes | age.group, dat)  
write.table(tab, "table1.csv", col.names=T, row.names=F, sep=',')
```

```
## exclude missing values
tab.nomissing <- table1(~age+sex+cellcount+accepted.transplant+diabetes | age.group, dat,
  render.missing=NULL)
write.table(tab.nomissing, "table1-nomissing.csv", col.names=T, row.names=F, sep=',')

## Another way to create a table 1 with p-values using CreateTableOne() in tableone package
library(tableone)

## Create a master table 1 for vars according to the "strata"
## The function calculates all following stats.
## Continuous variables: mean (SD), t-test, median (IQR), Wilcoxon test
## Categorical variables: count (%), Chi-square test, Fisher's exact test
tab2 <- CreateTableOne(vars=c('age', 'sex', 'cellcount', 'accepted.transplant', 'diabetes'),
  strata='age.group', data=dat,
  argsNormal=list(var.equal=FALSE))
tab2
```

```
##
##              Stratified by age.group
##              young      old      p      test
##  n              2734      208
##  age (mean (SD))  59.99 (13.53)  78.02 (1.46)  <0.001
##  sex = M (%)      1615 (59.1)   122 (58.7)   0.964
##  cellcount (mean (SD))  2547.69 (478.09)  2222.42 (546.54)  <0.001
##  accepted.transplant = Yes (%)  1907 (69.8)   106 (51.0)   <0.001
##  diabetes = Yes (%)   919 (34.6)    81 (40.7)   0.096
```

CreateTableOne() calculates stats and p-value in both parametric and non-parametric ways.

Options in print() can choose which stats and statistical tests are used for your table 1.

```
## To print out the table. You can choose which statistical test and statistics to be used.
## For continuous variables, you want to present
##   (a) mean (SD) and p-value by t-test [default]
##   (b) median (IQR) and Wilcoxon test
## For categorical variables, you want to present
##   (a) count (%) and p-value by Chi-square test [default],
##   (b) count (%) and p-value by Fisher's exact test
print(tab2, nonnormal=c('age', 'cellcount'), exact=c('sex', 'accepted.transplant', 'diabetes'), quote=T)
```

```
## Median (IQR) and Wilcoxon test for continuous variables
## Fisher's exact test for categorical variables.
write.csv(print(tab2, nonnormal=c('age', 'cellcount'),
  exact=c('sex', 'accepted.transplant', 'diabetes'),
  quote=T), file='table1-tableone-nonparametric.csv')

## Mean (SD) and t-test for continuous variables
## Fisher's exact test for categorical variables.
write.csv(print(tab2, exact=c('sex', 'accepted.transplant', 'diabetes'), quote=T),
  file='table1-tableone-parametric.csv')
```

Practice to create table 1 by diabetes using table1() or CreateTableOne()

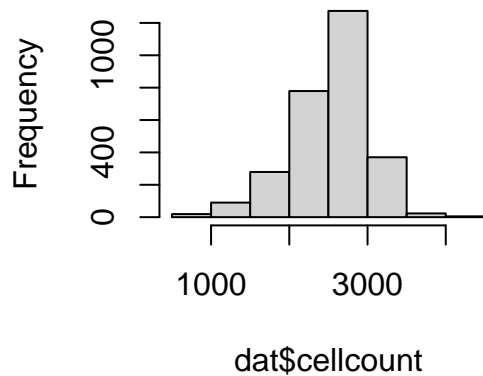
```
#####  
## Practice  
## Create Table 1 by diabetes  
#####
```

Plots in R

R is very good at making plots. We will show how make basic plots in R.

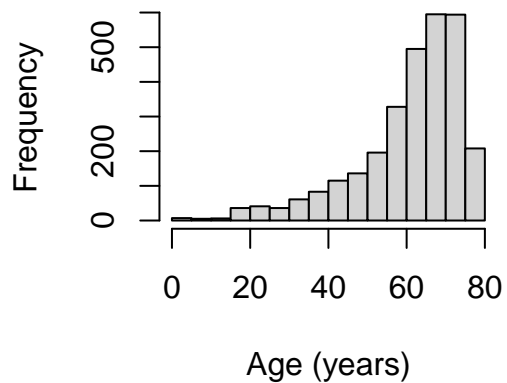
```
#####  
## Plots: slide 15  
#####  
## 1-variable plots  
## Histogram  
hist(dat$cellcount)
```

Histogram of dat\$cellcount

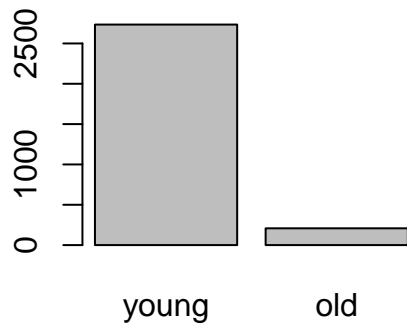


```
hist(dat$age, xlab='Age (years)', main='Histogram')
```

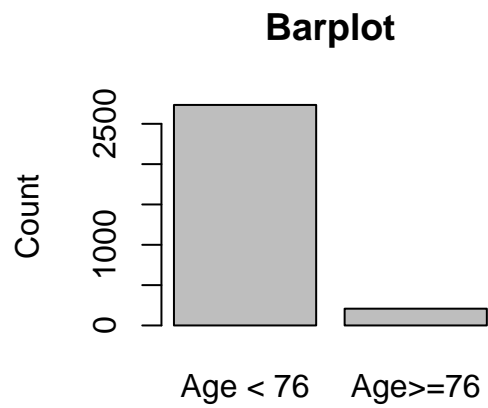
Histogram



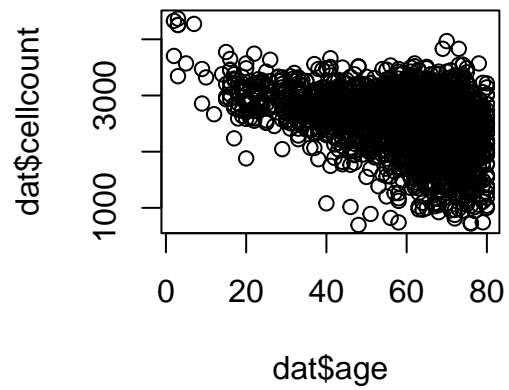
```
## barplot  
barplot(table(dat$age.group))
```



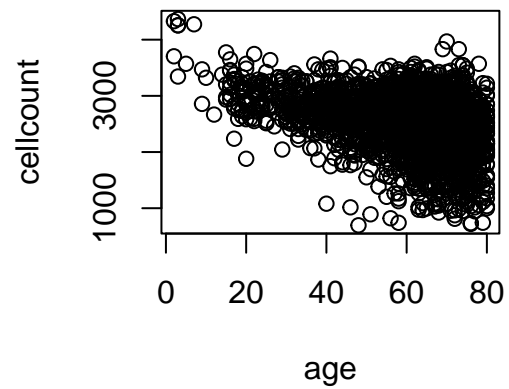
```
barplot(table(dat$age.group), names=c('Age < 76', 'Age>=76'), ylab='Count', main='Barplot')
```



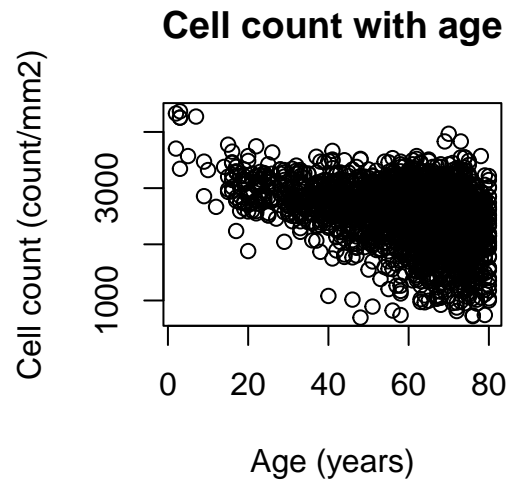

```
## between 2 variables: slide 16
## Scatter plot
plot(dat$age, dat$cellcount)
```



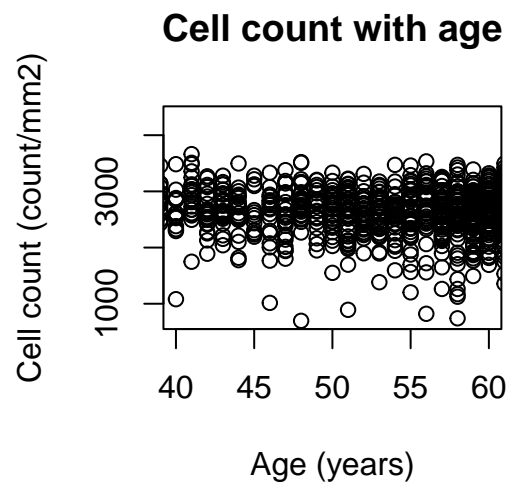
```
plot(cellcount~age, dat)
```



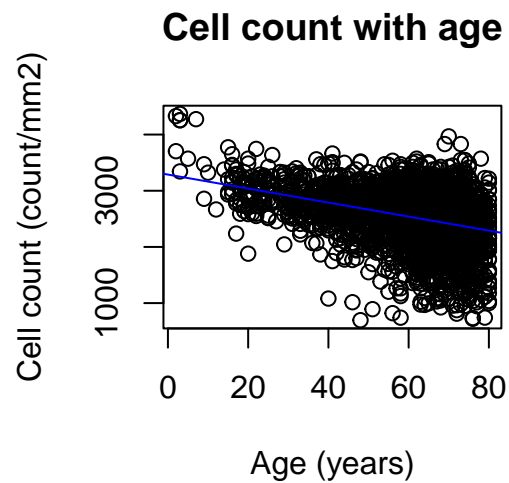
```
plot(cellcount~age, dat, xlab='Age (years)', ylab='Cell count (count/mm2)',
     main='Cell count with age')
```



```
plot(cellcount~age, dat, xlab='Age (years)', ylab='Cell count (count/mm2)',
     main='Cell count with age', xlim=c(40,60))
```



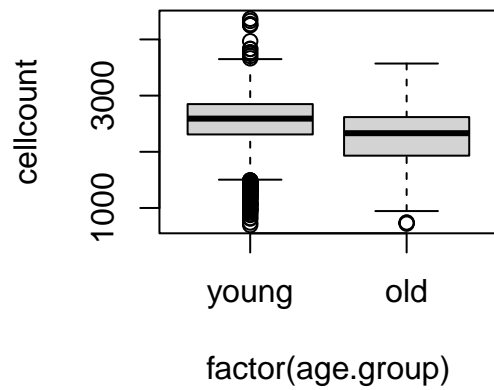
```
fit <- lm(cellcount~age, dat)
plot(cellcount~age, dat, xlab='Age (years)', ylab='Cell count (count/mm2)',
     main='Cell count with age')
abline(fit, col='blue') ## add a regression line
```



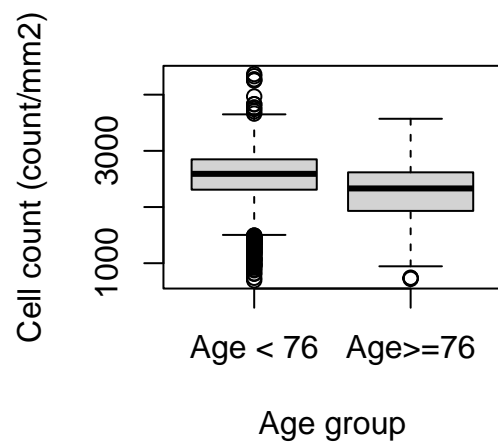
```
## save into pdf
pdf('cellcount-age.pdf', width=5, height=5)
plot(cellcount~age, dat, xlab='Age (years)', ylab='Cell count (count/mm2)', main='Cell count with age')
abline(fit, col='blue') ## add a regression line
dev.off()
```

```
## pdf
## 2
```

```
## Box plot
plot(cellcount ~ factor(age.group), dat)
```



```
plot(cellcount ~ age.group, dat, names=c('Age < 76', 'Age>=76'), ylab='Cell count (count/mm2)',
      xlab = 'Age group')
```



Save the data

After working with your data (cleaning), you might want to save the data (slide 18)

You can save the cleaned data into .csv or .txt files.

Another option is to save the R objects into .RData file. This will preserve the special attributes on the data such as factor. `load()` function can read .RData file in R.

```
#####  
## Export data (slide 18)  
#####  
write.csv(dat, file='eyebank-updated-10082024.csv')  
write.table(dat, file='eyebank-updated-10082024.txt', sep='\t')  
  
save(dat, file='eyebank-10082024.RData')  
load('eyebank-10082024.RData', verbose=T)
```

To quit R

```
#####  
## Quit R  
#####  
q()
```