Introduction to R and Rmarkdown

Xuemao Zhang
East Stroudsburg University

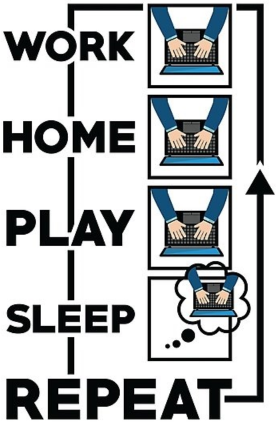
August 29, 2023

What’s covered in this lecture?

- Why programming?
- R, RStudio and Rstudio.cloud
- What is R Markdown?
- Header
- Markdown text
- Code chunks

Why programming?

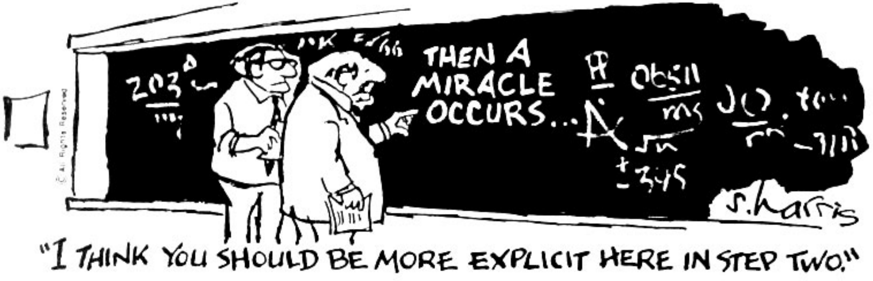
- To be able to easily repeat your own work.



Source: https://www.redbubble.com

Why programming?

- The workflow of using a script makes your research reproducible.



Source: Malanris.ru

Why programming?

- Programming isn't scary. If you've written formulas in Excel, you've already done "programming".

R

- It's a software environment for statistical computing and graphics, free and open source.
- It is availalbe for three plat forms: Linux, (Mac) OS X, and Windows.

R: https://www.r-project.org/

RStudio(an IDE, integrated development environment, for R): https://posit.co/ (https://www.rstudio.com/)

R

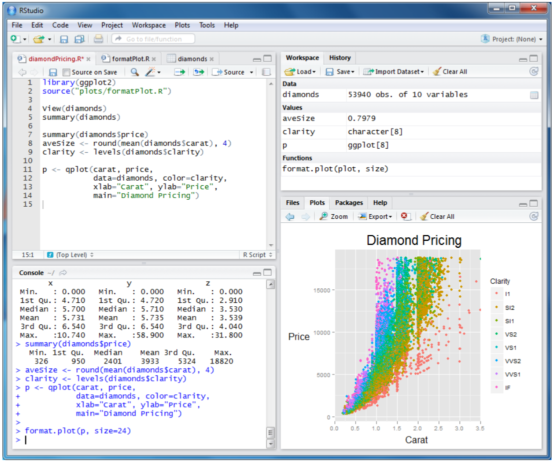
- It's designed to analyze data. The spreadsheet-like data structure “data frame” makes it easy to apply calculations.

```
##   emp_id emp_name salary start_date
## 1      1     Rick 623.30 2012-01-01
## 2      2      Dan 515.20 2013-09-23
## 3      3 Michelle 611.00 2014-11-15
## 4      4     Ryan 729.00 2014-05-11
## 5      5     Gary 843.25 2015-03-27
```

RStudio IDE

- RStudio is a popular IDE (Integrated Development Environment) for R programming
- It is a powerful editor for R coding and debugging.
- It is a powerful generator for HTML, PDF, dynamic documents and slide shows.
- RStudio can be run on both Desktop and Cloud (https://rstudio.cloud/).
    - You may use Rstudio Cloud only if you have difficulties installing R and Rstudio.
- Check out more nice features of RStudio at its official website(https://posit.co/products/open-source/rstudio/)

RStudio IDE



```r
library(ggplot2)
source("plots/formatPlot.R")

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")

```

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
> |
```

- To install R and Rstudio, please go to Installing R and RStudio https://rstudio-education.github.io/hopr/starting.html

Rstudio Cloud

- If you have difficulties installing R and Rstudio, you can use Rstuido cloud https://posit.cloud/learn/guide
- Watch the first 6 minutes of Posit Cloud Essentials https://youtu.be/-fzwm4ZhVQQ
  - We skip the `Data Connections`

What is R Markdown?

- Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.
- An R Markdown document is written in markdown (an easy-to-write plain text format) and contains **chunks of embedded R code**.
- An R Markdown file has name extension `.Rmd`.
- When you click the **Knit** button, a document will be generated that includes both content as well as the output of any embedded **R code chunks** within the document.
- Installation: First, install R and the RStudio IDE; Then in the R console, type `install.packages('rmarkdown')`.

What is R Markdown?

- Detailed information about RMarkdown can be found in the book R Markdown: The Definitive Guide by Yihui Xie.
- An R Markdown file generally contains three things.
  - A header at the top of the document.
  - Markdown text.
  - Code chunks.
    - Code chunks are used to render R (and code from other programming languages!) output into a document.

Header

- To create an R Markdown file,
  - Creat a plain text file and save it with the extension `.Rmd`.
  - Or you can click File > New File > R Markdown... in the RStudio toolbar.
- There are two types of output formats in the rmarkdown package: **documents**, and **presentations**.
- You can specify the output format in the YAML (originally meant Yet Another Markup Language now stands for YAML Ain't Markup Language) **header** at the top of the document.

Header

- The following is a header of a Markdown file. The header is enclosed by two sets of three dashes ---. This block allows you to fine-tune the output of your document.

```
---
title: "Writing documents with R Markdown"
author: John
date: "09/09/2023"
output: html_document
---
```

- Create an Rmarkdown file with this header only and click Knit, you can see how the output looks like.

Header

- The following is a list of some common output formats
  - beamer_presentation
  - powerpoint_presentation
  - html_document
  - pdf_document
- You can see the full list of YAML header options for a HTML document in the book R Markdown: The Definitive Guide by Yihui Xie.

Header

- You can add a table of contents (TOC) using the `toc` option and specify the depth of headers that it applies to using the `toc_depth` option. For example:

```
---
title: "Writing documents with R Markdown"
author: John
date: "09/09/2023"
output:
  html_document:
    toc: true
    toc_depth: 2
---
```

Header

- You can specify the toc_float option to float the table of contents to the left of the main document content. The floating table of contents will always be visible even when the document is scrolled.

```
---
title: "Writing documents with R Markdown"
author: John
date: "09/09/2023"
output:
    html_document:
      toc: true
      toc_float: true
      toc_depth: 3
---
```

Markdown text

- **Headers**: Place one or more hashtags at the start of a line that will be a header (or sub-header). For example,
    - # Say Hello to markdown. A single hashtag creates a first level header.
    - Two hashtags, ##, creates a second level header, and so on.
- **Italicized and bold text**:
    - Surround italicized text with asterisks, like this `*italicized text*`.
    - Surround bold text with two asterisks, like this `**bold text**`.
- **Lists**: Group lines into bullet points that begin with asterisks, dashes - or plus signs +. Leave a blank line before the first bullet, like this

```
This is a list

* item 1
* item 2
* item 3
```

Markdown text

- **Hyperlinks**: Surround links with brackets, and then provide the link target in parentheses, like this `[Github](https://github.com/)`.
- **Plain code blocks**: Plain code blocks are used to show R code without runing it. They can be written after three or more backticks, and ended with three or more backticks.

~~~~
```

install.packages('ggplot2');

library(ggplot2);

help(ggplot);

```
~~~~

Markdown text

- The following is an example of R Markdown file

```
---
title: "Writing documents with R Markdown"
author: John
date: "09/09/2023"
output:
  html_document:
    toc: true
    toc_float: true
    toc_depth: 3
---


# Header 1

This is an R Markdown document.


## Header 2

Use an asterisk mark to provide emphasis,
such as *itlatics* and **bold**.
```

Markdown text

~~~
Create lists with a dash

- Item 1

    - item 1.1

    - item 1.2 [Github](https://github.com/).

- Item 2

- Item 3

```
Use back ticks to create a block of code
```
~~~

Markdown text

Formatting:

- Italic. `*italic*`.

  This is *italic*.

- Bold. `**bold**`.

  This is **bold**.

- Superscripts. `y^2^`.

  This is y^2.

Code chunks

- The `knitr` package extends the basic markdown syntax to include chunks of executable code. When you render the report, knitr will run the code and add the results to the output file.
- Code chunks are used to render R (and code from other programming languages!) output into a document.

A code chunk delimiter looks like:

~~~
```{r}

```
~~~

- All code falls between the triple back tick marks, e.g:

~~~
```{r}

sin(3.1416/2);

```
~~~

Code chunks

- To omit the results from your final report (and not run the code) add the argument `eval = FALSE` inside the brackets and after r. This will place a copy of your code into the report.

~~~
```{r eval = FALSE}

# An example without running the code

sin(3.1416/2);

```
~~~

Code chunks

- To omit the code from the final report (while including the results) add the argument `echo = FALSE`. This will place a copy of the results into your report whthout showing the code.

~~~
```{r echo = FALSE}

# The dimensions of iris data are

dim(iris);

```
~~~

- For more other code chunk options, see section 2.6 R code chunks and inline R code of the book R Markdown: The Definitive Guide by Yihui Xie.

Code chunks

Inline code:

- To embed R code in a line of text, surround the code with a pair of backticks and the letter r, like this.

The dimensions of iris data are ` ` r dim(iris) ` `. (please remove the two spaces).

- `knitr` will replace the inline code with its result in your final document (inline code is always replaced by its result). The result will appear as if it were part of the original text.

Code chunks

- Add the following code trunks to the previous R Markdown file, knit and see the results.

~~~
```{r}
sin(3.1416/2);
```
```{r eval = FALSE}
# An example without running the code
sin(3.1416/2);
```
```{r echo = FALSE}
# The dimensions of iris data are
dim(iris);
```
~~~

The dimensions of iris data are r dim(iris) . (sorry, I could not type the back ticks)

Code chunks

- We can also create plots.
- By default, figures produced by R code will be placed immediately after the code chunk they were generated from. For example

~~~
```{r fig.align="center", out.width = '60%', echo=TRUE}

library(ggplot2);

qplot(data = mpg, displ, cty, geom = "point");

```
~~~

Code chunks

- We can use figure options to customise the output of the plot, e.g:
  - fig.align='center' to set the alignment to the middle of the document
  - fig.height=8 to set the height of the figure
  - fig.width=8 to set the width of the figure
  - fig.cap="Fig 1." to add a caption describing the plot
- Again, for more information read the book R Markdown: The Definitive Guide by Yihui Xie.

Code chunks

- Add the following code to the RMarkdown file and knit

~~~
```{r fig.align="center", out.width = '60%', echo=TRUE}
library(ggplot2);
qplot(data = mpg, displ, cty, geom = "point");
```
~~~

License

R Vectors and Operators

Xuemao Zhang
East Stroudsburg University

October 5, 2023

Outline

- R Console
- R Vectors
- R Operators
- An Example of Analysis of Univariate Data

R console

- You can enter commands one at a time at the command prompt (>). For example,

```
3+5
```

```
## [1] 8
```

The above is the form of code in our presentations: The first line is what I typed into the console; the second line is my result.

Note: # is the comment symbol in R.

When you run the code 3+5 in your local console, you type after the command prompt > and it will look like this:

```
> 3+5
[1] 8
```

R console

- But if we want to do more than one thing with the same data, it's best to store that data in a variable.

```
x=3;
y=7;
x;      #print(x)
```

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

```
y;      #print(y)
```

```
## [1] 7
```

```
X;      # R is case sensitive.
```

- R can be used as a caculator. Try the following after class.

```
>3**2
>3^2
>exp(1)
>log(3)    #the base of the log function is e
>pi
>sin(pi)   # sin(pi) is supposed to be zero
>round(sin(pi),4)   #Round sin(pi) to four decimal places
```

R console: need help?

- use google!
- use help() or ? to seek help.

```
help(log); #This asks for information about the log function
```

```
## starting httpd help server ... done
```

```
?log; #a  shorter way of asking for help
example(log); #This will give some  examples
```

```
##
## log> log(exp(3))
## [1] 3
##
## log> log10(1e7) # = 7
## [1] 7
##
## log> x <- 10^-(1+2*1:9)
##
## log> cbind(deparse.level=2, # to get nice column names
## log+       x, log(1+x), log1p(x), exp(x)-1, expm1(x))
##            x    log(1 + x)     log1p(x)   exp(x) - 1     expm1(x)
##  [1,] 1e-03 9.995003e-04 9.995003e-04 1.000500e-03 1.000500e-03
```

R console: clear work space

- ls() lists all variables in the worksapce
- rm() removes a variable
- rm(list=ls(all=TRUE)) deletes all variables in the worksapce

```
ls();
```

```
## [1] "x" "y"
```

```
rm(x);  # x;
y;
```

```
## [1] 7
```

```
rm(list=ls(all=TRUE)); # y;
```

R Vectors

- There are several data types in R. We can use the function `class()` to check the data type of an R object.
- The basic **data structure** in R is the **vector**, a sequence of data elements of the same basic type. In order to create a vector in R Programming, `c()` function is used.

```r
A=c(2, 3, 5);
A;
```

```
## [1] 2 3 5
```

```r
C=c("aa", "bb", "cc", "dd", "ee");
C;
```

```
## [1] "aa" "bb" "cc" "dd" "ee"
```

```r
length(C);  #how many elements it contains
```

```
## [1] 5
```

R Vectors

- Refer to elements of a vector using subscripts.

```
A=c(2, 3, 5);
A[1];
```

```
## [1] 2
```

```
A[c(1,3)];
```

```
## [1] 2 5
```

- Combining Vectors

```
A=c(2, 3, 5);
B = c("aa", "bb", "cc"); # a vector of characters
c(A, B);
```

```
## [1] "2"  "3"  "5"  "aa" "bb" "cc"
```

Note: In the above, numeric values are being coerced into character strings when the two vectors are combined. This is necessary so as to maintain the same primitive data type for elmentss in the same vector.

R operators

Arithmetic Operators: +,−, ∗, ˆ, /

- The operators act on each element of a vector.

```
v = c(2,5,6);
w = c(8,3,6);
v+w;      #sum of two vectors
```

```
## [1] 10  8 12
```

```
v-w;      #difference of two vectors
```

```
## [1] -6  2  0
```

```
v*w;      #product of two vectors
```

```
## [1] 16 15 36
```

```
v/w;      #quotient
```

```
## [1] 0.250000 1.666667 1.000000
```

```
v^2;      #square of a vector
```

```
## [1]  4 25 36
```

R operators

Relational Operators: >,<, ==, >=, <=, !=

```
v = c(2,5,6);
w = c(8,3,6);
v>w;
```

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

```
v<w;
```

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

```
v==w;
```

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

```
v>=w;
```

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

```
v!=w;
```

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

Note *TRUE* (or *T*) and *FALSE* (or *F*) are two **logical** values in R.

R operators

Assignment Operators: =, <-, c

```
v1 <- c(3,1,TRUE,2);
v1;
```

```
## [1] 3 1 1 2
```

```
v2 = c(3,1,TRUE,2);
v2;
```

```
## [1] 3 1 1 2
```

Right Assignment Operator: ->

```
c(3,1,F,2) -> v3;
v3;
```

```
## [1] 3 1 0 2
```

R operators

- : Colon operator. It creates the series of numbers in sequence for a vector.

```
v = 2:8;
v;
```

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

R operators

- %in% It is used to identify if an element belongs to a vector.

```r
v1 = 8;
v2 = 12;
w = 1:10;
print(v1 %in% w);
```

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

```r
print(v2 %in% w);
```

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

An example

```
x=c(10,30,5,25,40,20,10,15,30,20,15,20,85,15,65,15,60,60,40,45)
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   15.00   22.50   31.25   41.25   85.00
```

```
mean(x)
```

```
## [1] 31.25
```

```
sum(x)
```

```
## [1] 625
```

```
sum(x)/length(x)
```

```
## [1] 31.25
```

```
median(x)
```

```
## [1] 22.5
```

An example

```
var(x) #sample variance
```

```
## [1] 478.6184
```

```
sd(x) #sample sd
```

```
## [1] 21.87735
```

```
min(x)
```

```
## [1] 5
```

```
max(x)
```

```
## [1] 85
```

- To find the mode of a data set, we can create our own function or use an R package.
  - To install the package `modeest`, type `install.packages("modeest")` in the R console.

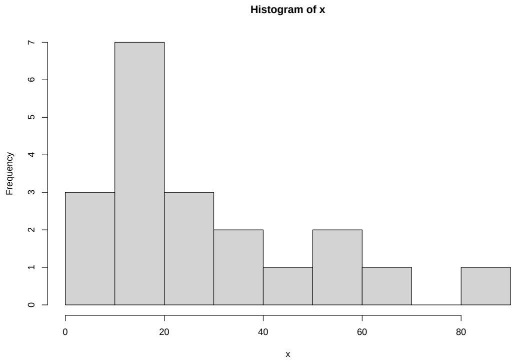
```
library(modeest)
mfv(x)
```

```
## [1] 15
```

An example

- Histogram

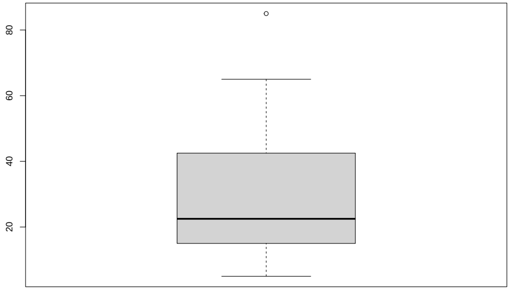
```
hist(x, breaks = 6)
```



An example

- Box plot

```
boxplot(x)
```



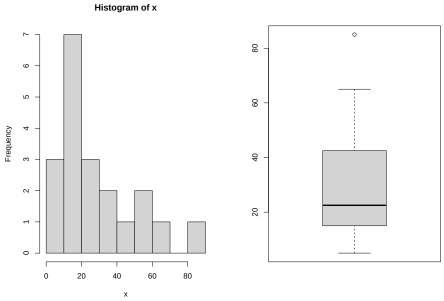
An example

- The par() function can be used to divide the graph frame into the desired grid

```
par( mfrow= c(m,n) )  # divide window into a mXn grid
```

- Let's put the two graphs above in a same window

```
par( mfrow= c(1,2) )
hist(x, breaks = 6)
boxplot(x)
```



```
par(mfrow=c(1,1)) #  reset the mfrow parameter
# or use dev.off()
```

An example

- Calculate the lower fence and upper fence in the above box plot
  - Recall that there are various algorithms to calculate quantiles.

```r
Q=quantile(x, probs=c(0.25,0.5,0.75))
Q= as.numeric(Q) #convert the object to a vector;
# so the labels `25%   50%   75%` will be removed
print(Q)
```

```
## [1] 15.00 22.50 41.25
```

An example

```
Q1=Q[1]
Q3=Q[3]
IQR= c(Q3-Q1)
Lower_fence=Q1-1.5*IQR
Upper_fence=Q3+1.5*IQR
print(Lower_fence)
```

```
## [1] -24.375
```

```
print(Upper_fence)
```

```
## [1] 80.625
```

License

Probability Distributions

Xuemao Zhang
East Stroudsburg University

October 26, 2023

Outline

- Finite distributions
- Binomial distributions
- Normal distributions
- Student-t distributions

All R code in this handout can be found in the lecture slides.

Finite distributions

- When a discrete random variable can take finite number of values, we call the distribution finite distribution.
- We calculate the parameters of a finte distribution with the help of R.
- **Example 1:** Find the mean, variance and standard deviation of a frequency/relative frequency table

Y	Freq
5	4
15	9
25	6
35	4
45	2

Finite distributions

- Data analysis

```
x = c(5, 15, 25, 35, 45)
w = c(4, 9, 6, 4, 2)
mu = sum(x * w) / sum(w)
cat("Population/Sample mean:", mu)
```

```
## Population/Sample mean: 21.4
```

```
sigma2 = sum((x - mu)^2 * w) / sum(w)
cat("Population variance:", sigma2)
```

```
## Population variance: 135.04
```

```
sigma = sqrt(sigma2)
cat("Population standard deviation:", sigma)
```

```
## Population standard deviation: 11.62067
```

Finite distributions

- Data analysis continued

```
s2 = sum((x - mu)^2 * w)/(sum(w) - 1)
cat("Sample variance:", s2)
```

```
## Sample variance: 140.6667
```

```
s = sqrt(s2)
cat("Sample standard deviation:", s)
```

```
## Sample standard deviation: 11.8603
```

Finite distributions

- **Example 2:** Find the mean, variance and standard deviation of a relative frequency table.

Y	Prob
95	0.15
83	0.25
76	0.25
84	0.35

```
x=c(95, 83, 76, 84)
w=c(0.15, 0.25, 0.25, 0.35)
mu=sum(x*w)

sigma2=sum(w*(x-mu)^2)
sigma2
```

```
## [1] 34.04
```

```
sigma=sqrt(sigma2)
sigma
```

```
## [1] 5.834381
```

- Therefore, if a relative fequency is given, we cannot find the sample variance (or sd) without knowing the size of the data.

Binomial Distributions

- Let X be a binomial random variable with number of trials $n = 12$ and probability of success $p = 0.67$. Calculate the following.
- 1. Find $P(X = 5)$.

```
dbinom(5, size = 12, prob = 0.67)
```

```
## [1] 0.04557186
```

```
dbinom(5,12,0.67)
```

```
## [1] 0.04557186
```

Binomial Distributions

- ② Find $P(X \leq 5)$ and $P(X < 5)$.

```
pbinom(5, size = 12, prob = 0.67, lower.tail = TRUE)
```

```
## [1] 0.06316753
```

```
pbinom(5, 12, 0.67)
```

```
## [1] 0.06316753
```

```
#tail is lower of cumulative by default

pbinom(4, 12, 0.67)
```

```
## [1] 0.01759567
```

Binomial Distributions

- (3) Find $P(X \geq 7)$

```
1-pbinom(6, 12, 0.67)
```

```
## [1] 0.828887
```

```
pbinom(6, 12, 0.67,lower.tail = FALSE)
```

```
## [1] 0.828887
```

```
# it is calculating P(X > 6)
```

- (4) Find $P(X > 7)$.

```
pbinom(7, 12, 0.67,lower.tail = FALSE)
```

```
## [1] 0.6410338
```

```
1-pbinom(7, 12, 0.67) #1- P(X <= 7)
```

```
## [1] 0.6410338
```

Binomial Distributions

- 5. Find $P(4 \leq X \leq 8)$.

```
pbinom(8, 12, 0.67)-pbinom(3, 12, 0.67)
```

```
## [1] 0.5937734
```

```
#difference between two cumulative probabilities
```

- 6. Find $P(4 < X \leq 8)$.

```
pbinom(8, 12, 0.67)-pbinom(4, 12, 0.67)
```

```
## [1] 0.5797448
```

```
#P(X<= 8) - P(X<= 4)
```

Binomial Distributions

- (7) Find $P(4 \leq X < 8)$

```
pbinom(7, 12, 0.67)-pbinom(3, 12, 0.67)
```

```
## [1] 0.3553991
```

```
#P(X<= 7) - P(X<= 3)
```

- (8) Find $P(4 < X < 8)$.

```
pbinom(7, 12, 0.67)-pbinom(4, 12, 0.67)
```

```
## [1] 0.3413705
```

```
#P(X<= 7) - P(X<= 4)
```

Normal distributions

- **Example:** Let X be normal random variable with mean of 24.8 and std dev of 6.2. Calculate the following.
  - Again, `lower.tail = TRUE` by default.
- 1. $P(X < 22)$ or $P(X \leq 22)$.

```
pnorm(22, 24.8, 6.2,lower.tail = TRUE)
```

```
## [1] 0.3257739
```

```
pnorm(22, 24.8, 6.2)
```

```
## [1] 0.3257739
```

Normal distributions

- (2) $P(X \geq 26)$ or $P(X > 26)$

```
1-pnorm(26, 24.8, 6.2)
```

```
## [1] 0.4232648
```

```
pnorm(26, 24.8, 6.2, lower.tail = FALSE)
```

```
## [1] 0.4232648
```

- (3) $P(22 < X < 26)$

```
pnorm(26, 24.8, 6.2)-pnorm(22, 24.8, 6.2)
```

```
## [1] 0.2509613
```

```
# difference between two lower tail areas
```

Normal distributions

- (4) what is the data value for the 40th percentile (40% of the population is below what value)?

```
qnorm(0.4, 24.8, 6.2)
```

```
## [1] 23.22925
```

- (5) 70% of the population is above what data value?

```
qnorm(1-0.7, 24.8, 6.2)
```

```
## [1] 21.54872
```

Normal distributions

- **Example** Find $Z_{0.025}$. Z is the symbol for standard normal distribution.

```
qnorm(0.975, 0,1) #lower tail is True
```

```
## [1] 1.959964
```

```
qnorm(0.975)  #standard normal by default
```

```
## [1] 1.959964
```

```
qnorm(0.025, lower.tail=FALSE)
```

```
## [1] 1.959964
```

Student-t distributions

- Function `pt()` is used to calculate t-probabilities.
- Function `qt()` is used to calculate t-quantiles.
  - There is only one parameter `df` which means degress of freedom.
  - Again, `lower.tail = TRUE` by default.
- **Example** Given a t-distribution t with `df=9`,

1. $P(-1 < t < 1)$

```
pt(1,9) - pt(-1,9)
```

```
## [1] 0.6565636
```

2. Find $t_{0.025}$

```
qt(0.975, 9)
```

```
## [1] 2.262157
```

```
qt(0.025, 9, lower.tail =FALSE)
```

```
## [1] 2.262157
```

License

Data Import and Export

Xuemao Zhang
East Stroudsburg University

November 7, 2023

Outline

- Common new user mistakes
- Working Directories
- Data Import
  - Data Summaries
  - Data type conversion
- Subsetting Data
  - Subsetting columns
  - Subsetting rows
- Data Export

Common new user mistakes

1. **Working directory problems: trying to read files that R “can’t find”**
   - RStudio can help, and so do RStudio Projects
2. Lack of comments in code
3. Typos (R is **case sensitive**, `x` and `X` are different)
   - RStudio helps with “tab completion” which can help correct your typos
4. Data type problems (is that a string or a number?)
5. Open ended quotes, parentheses, and brackets

Working Directories

- R “looks” for files on your computer relative to the **working** directory
  - Many people recommend not setting a directory in the scripts
  - If you open an R project to start a new RStudio session, the working directory is set for you.
- Getting the working directory

```
getwd() #obtain the current working directory
```

- We use function `setwd()` to set the working directory.

Working Directories

- An **absolute or full path** points to the same location in a file system, regardless of the current working directory. To do that, it must include the **root directory** when you use `setwd()`.
  - This means if I try your code, and you use absolute paths, it won't work unless we have the exact same folder structure where R is looking (bad).
- By contrast, a **relative path starts from some given working directory**, avoiding the need to provide the full absolute path. A filename can be considered as a relative path based at the current working directory.
- Typical directory structure syntax applies
  - ".." - goes up one level
  - "./" - is the current directory
  - "∼" - is your "home" directory

Data Import

- Easy way: R Studio features some nice "drop down" support, where you can run some tasks by selecting them from the toolbar.
  - For example, you can easily import text datasets using the "File –> Import Dataset" command. Selecting this will bring up a new screen that lets you specify the formatting of your text file.
  - After importing a datatset, you get the corresponding R commands that you can enter in the console if you want to re-import data.
- Write your code directly.

Data Import

- R can read almost any file format, especially via add-on packages.
- We are going to focus on comma separated (that is, '.csv') files.
  - We use base R functions like `read.csv()` or `read.table()`.

```
read.csv(file, header = TRUE, sep = ",",
quote = "\"", dec = ".",
fill = TRUE, comment.char = "", ...)
```

Data Import

- Let's read in the data set `fat.csv` from the command line

```r
fat = read.csv("fat.csv")
str(fat) # structure of the data
```

```
## 'data.frame':    252 obs. of  19 variables:
##  $ case         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ body.fat     : num  12.6 6.9 24.6 10.9 27.8 20.6 19 12.8 5.1 1
##  $ body.fat.siri: num  12.3 6.1 25.3 10.4 28.7 20.9 19.2 12.4 4.1
##  $ density      : num  1.07 1.09 1.04 1.08 1.03 ...
##  $ age          : int  23 22 22 26 24 24 26 25 25 23 ...
##  $ weight       : num  154 173 154 185 184 ...
##  $ height       : num  67.8 72.2 66.2 72.2 71.2 ...
##  $ BMI          : num  23.7 23.4 24.7 24.9 25.6 26.5 26.2 23.6 24
##  $ ffweight     : num  135 161 116 165 133 ...
##  $ neck         : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 4
##  $ chest        : num  93.1 93.6 95.8 101.8 97.3 ...
##  $ abdomen      : num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5
##  $ hip          : num  94.5 98.7 99.2 101.2 101.9 ...
##  $ thigh        : num  59 58.7 59.6 60.1 63.2 66 58.4 60 62.9 63.
##  $ knee         : num  37.3 37.3 38.9 37.3 42.2 42 38.3 39.4 38.3
```

Data Summaries

- The imported data is a **dataframe**, a spreadsheet-like table.
- Show the header of the data

```r
head(fat)
```

```
##   case body.fat body.fat.siri density age weight height  BMI ffwe
## 1    1     12.6          12.3  1.0708  23 154.25  67.75 23.7    1
## 2    2      6.9           6.1  1.0853  22 173.25  72.25 23.4    1
## 3    3     24.6          25.3  1.0414  22 154.00  66.25 24.7    1
## 4    4     10.9          10.4  1.0751  26 184.75  72.25 24.9    1
## 5    5     27.8          28.7  1.0340  24 184.25  71.25 25.6    1
## 6    6     20.6          20.9  1.0502  24 210.25  74.75 26.5    1
##   chest abdomen   hip thigh knee ankle bicep forearm wrist
## 1  93.1     85.2  94.5  59.0 37.3  21.9  32.0    27.4  17.1
## 2  93.6     83.0  98.7  58.7 37.3  23.4  30.5    28.9  18.2
## 3  95.8     87.9  99.2  59.6 38.9  24.0  28.8    25.2  16.6
## 4 101.8     86.4 101.2  60.1 37.3  22.8  32.4    29.4  18.2
## 5  97.3    100.0 101.9  63.2 42.2  24.0  32.2    27.7  17.7
## 6 104.5     94.4 107.8  66.0 42.0  25.6  35.7    30.6  18.8
```

Data Summaries

- show the last several rows

```
tail(fat)
```

```
##     case body.fat body.fat.siri density age weight height  BMI ff
## 247  247     29.1          30.2  1.0308  69 215.50  70.50 30.5
## 248  248     11.5          11.0  1.0736  70 134.25  67.00 21.1
## 249  249     32.3          33.6  1.0236  72 201.00  69.75 29.1
## 250  250     28.3          29.3  1.0328  72 186.75  66.00 30.2
## 251  251     25.3          26.0  1.0399  72 190.75  70.50 27.0
## 252  252     30.7          31.9  1.0271  74 207.50  70.00 29.8
##     chest abdomen   hip thigh knee ankle bicep forearm wrist
## 247 113.7   107.6 110.0  63.3 44.0  22.6  37.5    32.6  18.8
## 248  89.2    83.6  88.8  49.6 34.8  21.5  25.6    25.7  18.5
## 249 108.5   105.0 104.5  59.6 40.8  23.2  35.2    28.6  20.1
## 250 111.1   111.5 101.7  60.3 37.3  21.5  31.3    27.2  18.0
## 251 108.3   101.3  97.8  56.0 41.6  22.7  30.5    29.4  19.8
## 252 112.4   108.5 107.1  59.3 42.2  24.6  33.7    30.0  20.9
```

Data Summaries

- `nrow()` displays the number of rows of a data frame
- `ncol()` displays the number of columns
- `dim()` displays a vector of length 2: # rows, # columns

```
nrow(fat)     # number of data rows
```

```
## [1] 252
```

```
ncol(fat)     # number of columns
```

```
## [1] 19
```

```
dim(fat)      # dimensions of the data
```

```
## [1] 252  19
```

Data Summaries

- colnames() displays the column names (if any) and rownames() displays the row names (if any)

```
rownames(mtcars)
```

```
##  [1] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
##  [4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
##  [7] "Duster 360"          "Merc 240D"           "Merc 230"
## [10] "Merc 280"            "Merc 280C"           "Merc 450SE"
## [13] "Merc 450SL"          "Merc 450SLC"         "Cadillac Fleetw
## [16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
## [19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
## [22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
## [25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
## [28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
## [31] "Maserati Bora"       "Volvo 142E"
```

```
colnames(fat)
```

```
##  [1] "case"          "body.fat"      "body.fat.siri" "density"
##  [5] "age"           "weight"        "height"        "BMI"
##  [9] "ffweight"      "neck"          "chest"         "abdomen"
```

Data type conversion

- Use `is.Type()` to test for data Type.
- Use `as.Type` to explicitly convert it.

```
is.numeric(),    as.numeric()
is.character(),  as.character()
is.factor(),  as.factor()
is.vector(),     as.vector()
is.matrix(),     as.matrix()
is.data.frame(), as.data.frame()
```

Data type conversion

```r
fat$age=as.character(fat$age)
str(fat)
```

```
## 'data.frame':    252 obs. of  19 variables:
##  $ case          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ body.fat      : num  12.6 6.9 24.6 10.9 27.8 20.6 19 12.8 5.1 1
##  $ body.fat.siri: num  12.3 6.1 25.3 10.4 28.7 20.9 19.2 12.4 4.1
##  $ density       : num  1.07 1.09 1.04 1.08 1.03 ...
##  $ age           : chr  "23" "22" "22" "26" ...
##  $ weight        : num  154 173 154 185 184 ...
##  $ height        : num  67.8 72.2 66.2 72.2 71.2 ...
##  $ BMI           : num  23.7 23.4 24.7 24.9 25.6 26.5 26.2 23.6 24
##  $ ffweight      : num  135 161 116 165 133 ...
##  $ neck          : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 4
##  $ chest         : num  93.1 93.6 95.8 101.8 97.3 ...
##  $ abdomen       : num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5
##  $ hip           : num  94.5 98.7 99.2 101.2 101.9 ...
##  $ thigh         : num  59 58.7 59.6 60.1 63.2 66 58.4 60 62.9 63.
##  $ knee          : num  37.3 37.3 38.9 37.3 42.2 42 38.3 39.4 38.3
##  $ ankle         : num  21.9 23.4 24 22.8 24 25.6 22.9 23.2 23.8 2
```

Subsetting columns

- We can extract specific column from a data frame using column name.

```
names1=c("body.fat","weight","height","abdomen")
fat1=fat[names1] #extract the above variables
str(fat1)
```

```
## 'data.frame':    252 obs. of  4 variables:
##  $ body.fat: num  12.6 6.9 24.6 10.9 27.8 20.6 19 12.8 5.1 12 ..
##  $ weight  : num  154 173 154 185 184 ...
##  $ height  : num  67.8 72.2 66.2 72.2 71.2 ...
##  $ abdomen : num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5 88.6
```

Subsetting columns

- A single column can be extracted using the $ operator

```r
wt=fat$weight
str(wt)
```

```
##  num [1:252] 154 173 154 185 184 ...
```

Subsetting columns

- We can remove a column as well.

```
fat$case=NULL
colnames(fat)
```

```
##  [1] "body.fat"      "body.fat.siri" "density"       "age"
##  [5] "weight"        "height"        "BMI"           "ffweight"
##  [9] "neck"          "chest"         "abdomen"       "hip"
## [13] "thigh"         "knee"          "ankle"         "bicep"
## [17] "forearm"       "wrist"
```

Subsetting columns

- A new column can be added

```
fat$new_col = fat$weight/ fat$height
colnames(fat)
```

```
##  [1] "body.fat"      "body.fat.siri" "density"       "age"
##  [5] "weight"        "height"        "BMI"           "ffweight"
##  [9] "neck"          "chest"         "abdomen"       "hip"
## [13] "thigh"         "knee"          "ankle"         "bicep"
## [17] "forearm"       "wrist"         "new_col"
```

Subsetting rows

- We can extract specific rows from a data frame.

```
fat2=fat1[1:50,]     #extract the first 50 rows
fat2
```

```
##    body.fat weight height abdomen
## 1      12.6 154.25  67.75    85.2
## 2       6.9 173.25  72.25    83.0
## 3      24.6 154.00  66.25    87.9
## 4      10.9 184.75  72.25    86.4
## 5      27.8 184.25  71.25   100.0
## 6      20.6 210.25  74.75    94.4
## 7      19.0 181.00  69.75    90.7
## 8      12.8 176.00  72.50    88.5
## 9       5.1 191.00  74.00    82.5
## 10     12.0 198.25  73.50    88.6
## 11      7.5 186.25  74.50    83.6
## 12      8.5 216.00  76.00    90.9
## 13     20.5 180.50  69.50    91.6
## 14     20.8 205.25  71.25   101.8
## 15     21.7 187.75  69.50    96.4
```

Subsetting rows

- We can remove some rows (using - sign) as well.

```
fat3=fat2[-c(49,50),]  #remove the last two rows
tail(fat3)
```

```
##    body.fat weight height abdomen
## 43     30.4 217.00  70.00   111.2
## 44     30.8 212.00  71.50   104.3
## 45      8.4 125.25  68.00    76.0
## 46     14.1 164.25  73.25    81.5
## 47     11.2 133.50  67.50    73.7
## 48      6.4 148.50  71.25    79.5
```

Subsetting rows

- We can subset rows based on the value of a variable or more variables

```
#obtain a subset with age >=50
dim(fat)
```

```
## [1] 252  19
```

```
fat4=subset(fat, age>=50)
dim(fat4)
```

```
## [1] 83 19
```

```
fat4$age
```

```
##  [1] 50 50 51 54 58 62 54 61 62 56 54 61 57 55 54 55 54 55 62 55
## [26] 69 81 66 67 64 64 70 72 67 72 64 53 50 52 51 52 50 50 52 50
## [51] 53 54 54 54 55 55 55 55 55 56 56 57 57 58 58 60 62 62 63 64
## [76] 67 68 69 70 72 72 72 74
```

Subsetting rows

You can have multiple logical conditions using the following:

- & : AND
- | : OR

```
#obtain a subset with age >=50 and  BMI >=30
fat5=subset(fat, age>=50 & BMI >=30)
dim(fat5)
```

```
## [1] 10 19
```

```
fat5[c('age', 'BMI')]
```

```
##     age  BMI
## 40   50 31.8
## 216  51 37.6
## 222  54 31.0
## 238  63 31.9
## 240  65 30.9
## 242  65 33.9
## 243  66 31.8
## 244  67 30.3
```

Data Export

While its nice to be able to read in a variety of data formats, it's equally important to be able to output data in the R workspace to your hard/usb drive.

`write.table()`: prints its required argument `x` (after converting it to a `data.frame` if it is not one nor a `matrix`) to a file or connection.

```
write.table(x,file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
```

Data Export

`x`: the R `data.frame` or `matrix` you want to write

`file`: the file name where you want to R object written. It can be an absolute path, or a filename (which writes the file to your working directory)

`sep`: what character separates the columns?

- `sep =","` = .csv - Note there is also a `write.csv()` function
- `sep ="\t"` = tab delimited

`row.names`: setting this to TRUE or FALSE

Data Export

- Note that `row.names=TRUE` would make the first column contain the row names, which is not very useful for Excel.

```
write.table(fat2, file = "fat2.txt", sep = "\t",
            row.names = TRUE, col.names = NA);
#saved as tab-separated text file
write.csv(fat3, file = "fat3.csv",row.names = TRUE)
```

You will find that the two files fat2.txt and fat3.csv are saved on your current directory.

License

Univariate and Bivariate Data Analysis

Xuemao Zhang
East Stroudsburg University

November 23, 2023

Outline

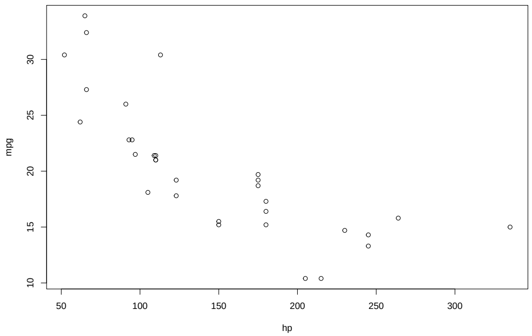
- SLR Models
  - Scatter plot
  - Fitting an SLR model
- Statistical inferences about population means
  - Confidence interval estimations
  - Hypothesis testing

SLR models

When two variables are measured (not always but usually on a single experimental unit), the resulting data are called bivariate data (or Paired data). When both of the variables (X, Y) are quantitative, call the variable X - the *independent variable*, and Y - the *dependent variable*. A random sample is of the form

$$(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n).$$

A typical scatter plot is like



SLR models

Assume that visual examination of the scatter plot confirms that the points approximate a straight-line pattern

$$y = \beta_0 + \beta_1 x.$$

This model is called a **deterministic** mathematical model because it does not allow for any error in predicting y as a function of x.

However, the bivariate measurements that we observe do not generally fall exactly on a straight line, we choose to use a **probabilistic** model: for any fixed value of x,

$$E(Y|X = x) = \beta_0 + \beta_1 x.$$

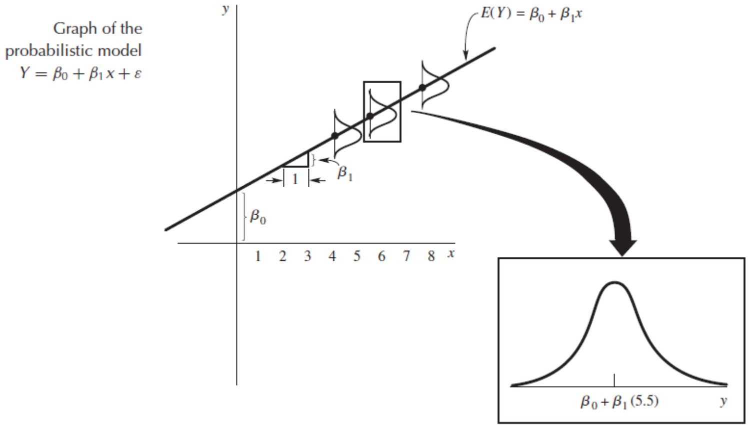
or, equivalently,

$$Y|_{X=x} = \beta_0 + \beta_1 x + \varepsilon,$$

where ε is a random variable possessing a specified probability distribution with mean 0.

For example, assume that ε's are independent normal random variables with mean 0 and common variance σ^2.

SLR models



Graph of the probabilistic model $Y = \beta_0 + \beta_1 x + \varepsilon$
y
$E(Y) = \beta_0 + \beta_1 x$
β_1
1
β_0
1 2 3 4 5 6 7 8 x
$\beta_0 + \beta_1 (5.5)$
y

We estimate the population parameters β_0 and β_1 using sample information.

Scatter plot

- Let's consider the data set `fat.csv` again

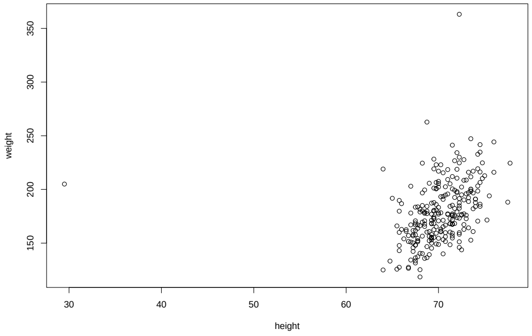
```r
fat = read.csv("fat.csv")
str(fat) # structure of the data
```

```
## 'data.frame':    252 obs. of  19 variables:
##  $ case         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ body.fat     : num  12.6 6.9 24.6 10.9 27.8 20.6 19 12.8 5.1 1
##  $ body.fat.siri: num  12.3 6.1 25.3 10.4 28.7 20.9 19.2 12.4 4.
##  $ density      : num  1.07 1.09 1.04 1.08 1.03 ...
##  $ age          : int  23 22 22 26 24 24 26 25 25 23 ...
##  $ weight       : num  154 173 154 185 184 ...
##  $ height       : num  67.8 72.2 66.2 72.2 71.2 ...
##  $ BMI          : num  23.7 23.4 24.7 24.9 25.6 26.5 26.2 23.6 24
##  $ ffweight     : num  135 161 116 165 133 ...
##  $ neck         : num  36.2 38.5 34 37.4 34.4 39 36.4 37.8 38.1 4
##  $ chest        : num  93.1 93.6 95.8 101.8 97.3 ...
##  $ abdomen      : num  85.2 83 87.9 86.4 100 94.4 90.7 88.5 82.5
##  $ hip          : num  94.5 98.7 99.2 101.2 101.9 ...
##  $ thigh        : num  59 58.7 59.6 60.1 63.2 66 58.4 60 62.9 63.
```

Scatter plot

- Consider the relationship between `height` and `weight`
- Scatter plot of `height` and `weight`

```
plot(fat$height, fat$weight, xlab = 'height', ylab = 'weight')
```



Scatter plot

- We see two influential points. Let's remove them.
  - We update the data directly without saving a copy

```
nrow(fat)
```

```
## [1] 252
```

```
fat=subset(fat, height>50 & weight<300)
nrow(fat)
```

```
## [1] 250
```

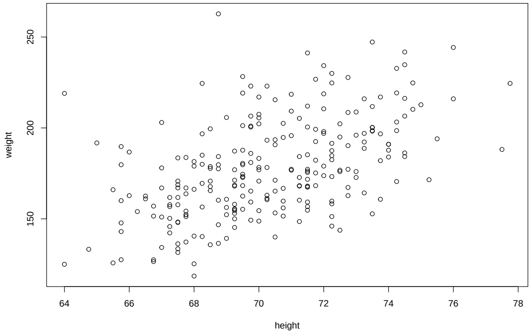
- To let R find the variables without using $: `fat$height` and `fat$weight`, we can use the `attach` function

```
attach(fat)
```

Scatter plot

- Scatter plot again

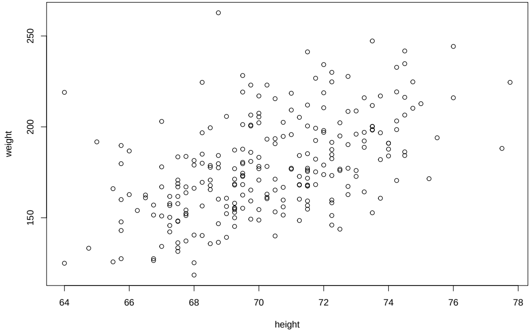
```
plot(height, weight, xlab = 'height', ylab = 'weight')
```



Scatter plot

- Scatter plot again

```
plot(weight~height, data=fat)
```



- Linear correlation coefficient

```
cor(fat$height, fat$weight)
```

Fitting an SLR model

- Model fit

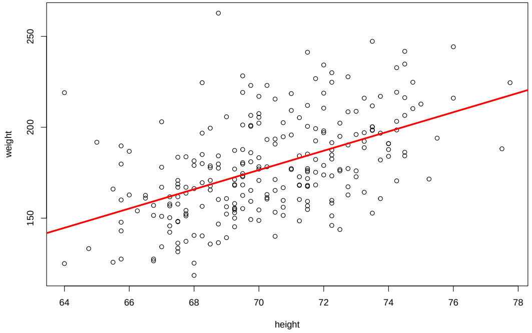
```
fit = lm(weight~height, data=fat)
summary(fit)
```

```
##
## Call:
## lm(formula = weight ~ height, data = fat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.378 -16.032  -1.465  13.816  92.897
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -194.4861    39.6230  -4.908 1.67e-06 ***
## height         5.2995     0.5632   9.409  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Fitting an SLR model

- Scatter plot with the LS line

```r
plot(weight~height, data=fat)
abline(fit, col = "red",lwd=3)  #lwd is line width
```



Fitting an SLR model

- To get the estimates of the parameters

```
fit$coefficients
```

```
## (Intercept)      height
##  -194.48613     5.29948
```

- Coefficient of determination

```
summaryfit=summary(fit)
summaryfit$r.squared
```

```
## [1] 0.2630798
```

```
cor(fat$height, fat$weight)^2
```

```
## [1] 0.2630798
```

Fitting an SLR model

- ANOVA table

```
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: weight
##            Df Sum Sq Mean Sq F value    Pr(>F)
## height      1  47880   47880  88.536 < 2.2e-16 ***
## Residuals 248 134118     541
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fitting an SLR model

- Fitted values and residuals

```
Fitted= fit$fitted
Resd= fit$residuals
```

- Check the first several values

```
Fitted[1:4]
```

```
##        1        2        3        4
## 164.5536 188.4013 156.6044 188.4013
```

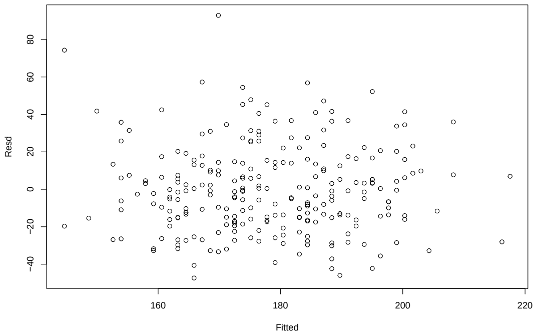
```
Resd[1:4]
```

```
##          1          2          3          4
## -10.303628 -15.151287  -2.604408  -3.651287
```

Fitting an SLR model

- Residual plot

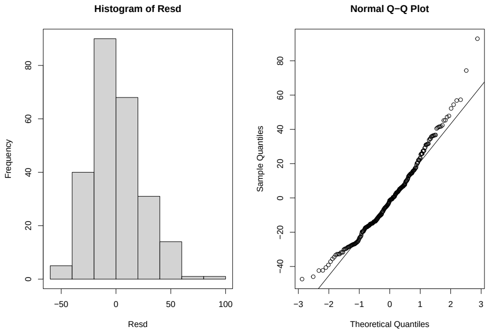
```
plot(Fitted, Resd)   #plot(fat$height, Resd)
```



Fitting an SLR model

- Normality assumption can be checked using historgram and QQ plot of the residuals

```
par(mfrow = c(1, 2))
hist(Resd)
box()
qqnorm(Resd)
qqline(Resd)
```



```
dev.off()
```

CI estimations and t-test

- When $\overline{X}$ is the mean of a random sample of size n from a **normal distribution** with mean μ, the random variable

$$T = \frac{\overline{X} - \mu}{S/\sqrt{n}}$$

  has a t-distribution with $n - 1$ degrees of freedom (df).
- If the population is not normally distributed, but the sample size n is large (> 30), then the statistics T above is approximately t-distributed with $n - 1$ df.
- The t-distribution of T is robust to small or even moderate departures from normality unless the sample size n is quite small.

CI estimations and t-test

- Confidence Intervals

Let $X_1, \ldots, X_n$ be a random sample from a normal population with mean μ. Then

- A two-sided $1-\alpha$ CI of μ is $\overline{X} \pm t_{\alpha/2}\left(\frac{S}{\sqrt{n}}\right)$,

where $t_{\alpha/2}$ is determined from the t-distribution with $df = n-1$.

CI estimations and t-test

- One Sample t-Test

$$
\boxed{
\begin{array}{l}
H_0 : \mu = \mu_0 \\
H_a : \begin{cases} \mu > \mu_0, & \text{upper-tail alternative;} \\ \mu < \mu_0, & \text{lower-tail alternative;} \\ \mu \neq \mu_0, & \text{two-tailed alternative.} \end{cases} \\
\text{Test Statistic: } t_0 = \dfrac{\overline{X} - \mu_0}{S/\sqrt{n}} \\
\text{Rejection Region: } RR = \begin{cases} \{t : t \geq t_\alpha\}, & \text{upper-tail RR;} \\ \{t : t \leq -t_\alpha\}, & \text{lower-tail RR;} \\ \{t : |t| \geq t_{\alpha/2}\}, & \text{two-tailed RR.} \end{cases} \\
\text{where the t-distribution has } df = n - 1.
\end{array}
}
$$

- p-value method:
  - Test statistic: $t_0 = \frac{\overline{X} - \mu_0}{S/\sqrt{n}}$
  - $H_a : \mu > \mu_0$: p-value$=P(t \geq t_0)$
  - $H_a : \mu < \mu_0$: p-value$= P(t \leq t_0)$
  - $H_a : \mu \neq \mu_0$: p-value$= 2P(t \geq |t_0|)$
- Confidence interval method: We reject $H_0 : \mu = \mu_0$ at significance level α if μ_0 lies outside the interval. Especially
  - $\overline{X} \pm t_{\alpha/2}\left(\frac{S}{\sqrt{n}}\right)$ is for the two-sided test: $H_a : \mu \neq \mu_0$

CI estimations and t-test

- Let's consider the `weight` variable in the `fat` data set
- Find a 95% confidence interval of `weight`
  - We studied 2-sided CI only, you must specify `alternative ="two.sided"`

```
weight = fat$weight
t.test(weight, alternative ="two.sided", conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  weight
## t = 104.15, df = 249, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  174.7155 181.4509
## sample estimates:
## mean of x
##  178.0832
```

CI estimations and t-test

- Hypothesis test of $H_0 : weight = 182$ versus $H_0 : weight \neq 182$
  - It is a two-sided test

```
weight = fat$weight
t.test(weight, alternative ="two.sided", mu=182)
```

```
##
##  One Sample t-test
##
## data:  weight
## t = -2.2907, df = 249, p-value = 0.02282
## alternative hypothesis: true mean is not equal to 182
## 95 percent confidence interval:
##  174.7155 181.4509
## sample estimates:
## mean of x
##  178.0832
```

CI estimations and t-test

- Hypothesis test of $H_0 : weight = 182$ versus $H_0 : weight > 182$
  - It is a right-sided test

```
weight = fat$weight
t.test(weight, alternative ="greater", mu=182)
```

```
##
##  One Sample t-test
##
## data:  weight
## t = -2.2907, df = 249, p-value = 0.9886
## alternative hypothesis: true mean is greater than 182
## 95 percent confidence interval:
##  175.2602      Inf
## sample estimates:
## mean of x
##  178.0832
```

CI estimations and t-test

- Hypothesis test of $H_0 : weight = 182$ versus $H_0 : weight < 182$
  - It is a left-sided test

```
weight = fat$weight
t.test(weight, alternative ="less", mu=182)
```

```
##
##  One Sample t-test
##
## data:  weight
## t = -2.2907, df = 249, p-value = 0.01141
## alternative hypothesis: true mean is less than 182
## 95 percent confidence interval:
##      -Inf 180.9062
## sample estimates:
## mean of x
##  178.0832
```

License