Regression in R

CMED6020 – Session 2

Eric Lau (ehylau@hku.hk)

School of Public Health
The University of Hong Kong

18 Jan 2021

Outline

- 18:30 to 19:15 – R Graphics and generating random numbers
- 19:15 to 20:00 – Practical
- 20:00 to 20:30 – Linear regression in R
- 20:30 to 21:00 – Practical

Session 2 learning objectives

After this session, students should be able to

- Produce clear and well-formatted graphs in R
- Generating random numbers
- Fit linear regression in R
- Interpret regression results

R Graphics

R Graphics

- **High-level plotting functions** create a new plot on the graphics device, possibly with axes, labels, titles and so on.
- **Low-level plotting functions** add more information to an existing plot, such as extra points, lines and labels.

High-level plotting functions

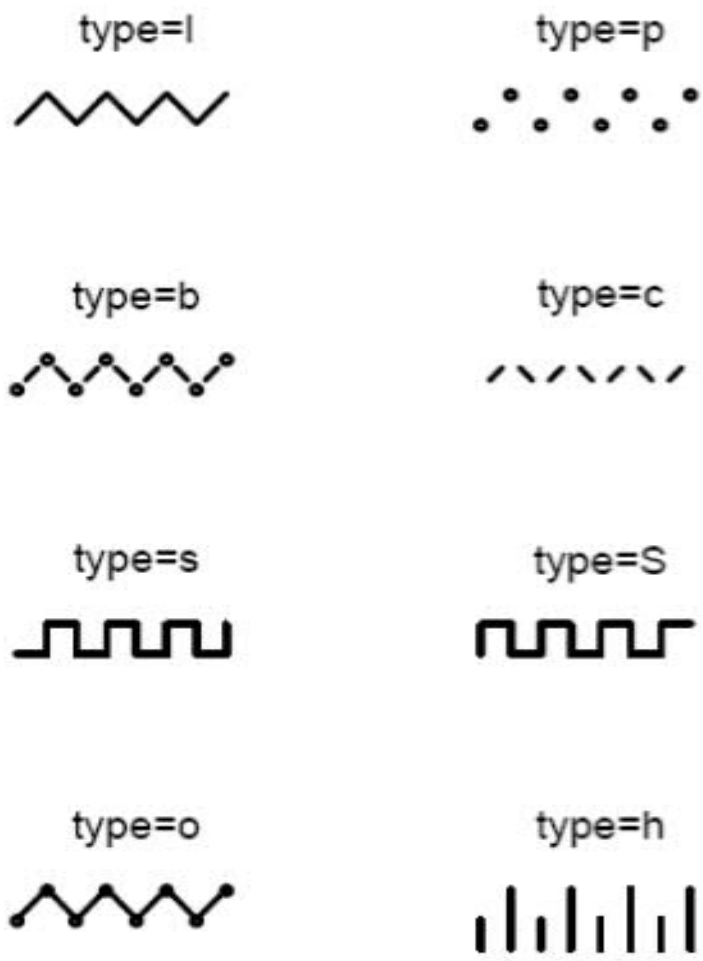
- plot(x, y, …)
- pairs(data)
- hist(x)
- boxplot(x) and boxplot(y~x)

The plot() function

- plot(x, y)
- plot(x, y, type="p", axes=TRUE, xlim=c(0,1), ylim=c(0,1), main="Main title", xlab="x-axis title", ylab="y-axis title")
  - **type** indicates type of plot (points, line, both, …).
  - **axes** indicates whether axes should be plotted.
  - **xlim, ylim** specify the limits of the x- and y-axes.
  - **main, xlab, ylab** specify the main and axes titles.

Plot types

- type="n" plots nothing, but sets up the plot region and coordinate system (this can be very useful!)
- Other plot types shown on the right



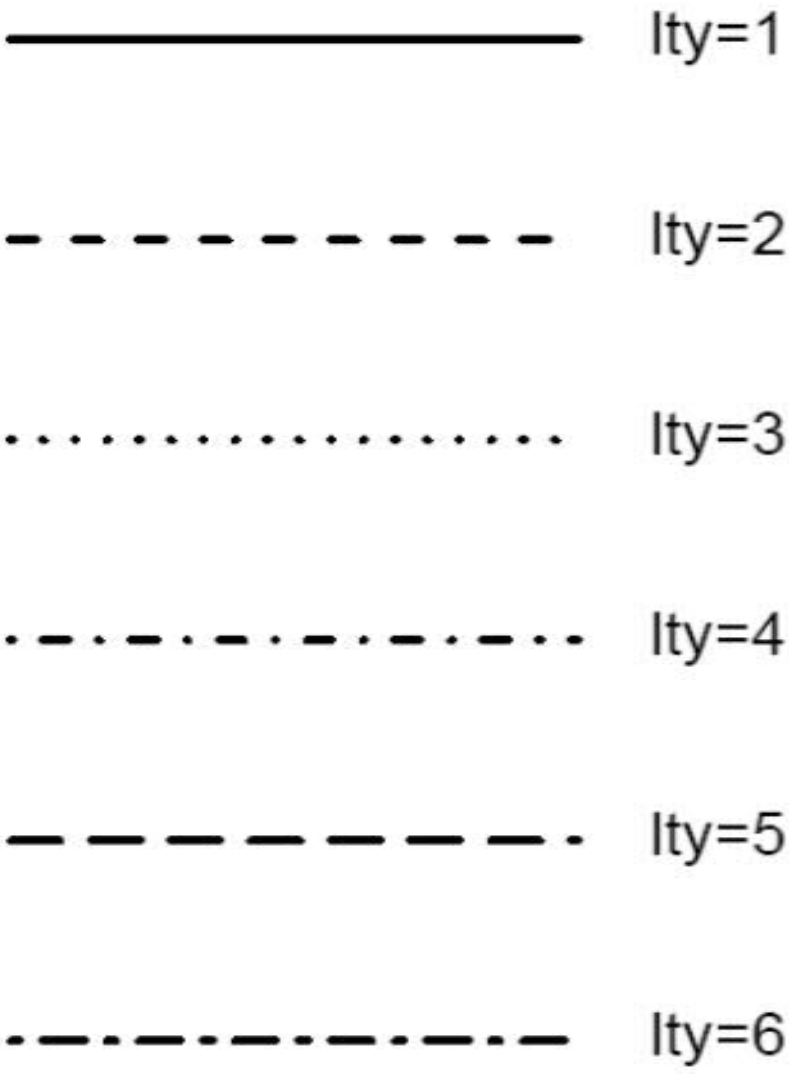
Lower level plotting commands

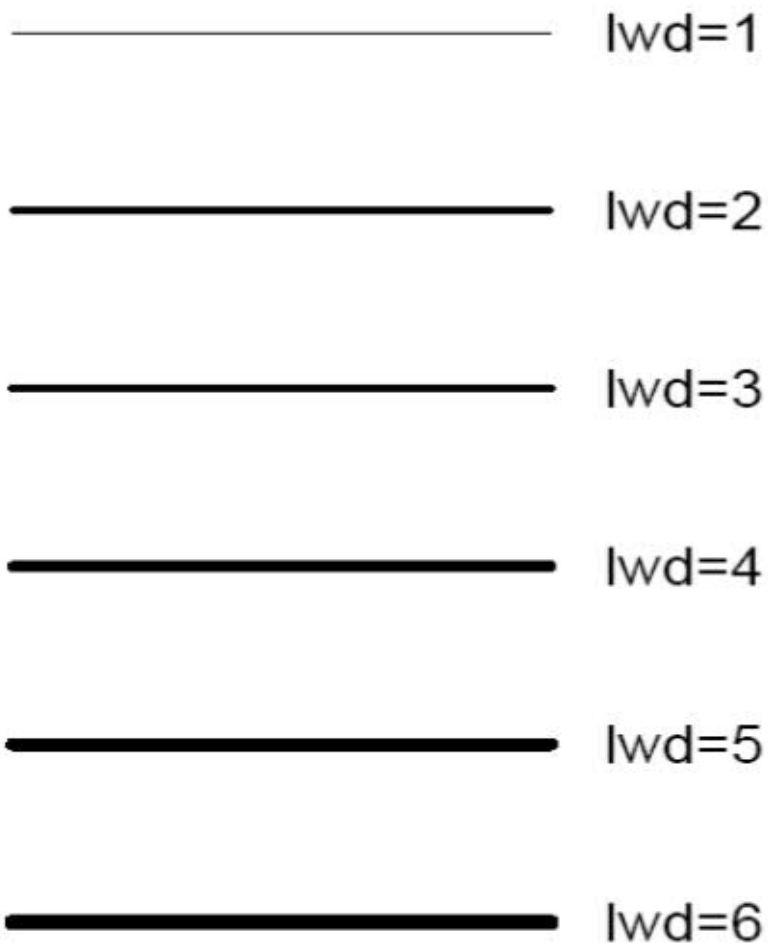
- points(x, y)
- points(x, y, pch=1, cex=1, col=1)
  - **pch** is the plot character
  - **cex** is the character expansion (relative size)
  - **col** is the colour
- lines(x, y)
- lines(x, y, lty=1, lwd=1)
  - **lty, lwd** are the line type and width respectively

Points – pch and cex

□ pch=0	✡ pch=11	· cex=0.5
○ pch=1	⊞ pch=12	
△ pch=2	⊠ pch=13	• cex=1
+ pch=3	⊠ pch=14	
× pch=4	■ pch=15	● cex=1.5
◇ pch=5	● pch=16	
▽ pch=6	▲ pch=17	● cex=2
⊠ pch=7	◆ pch=18	
✳ pch=8	a pch='a'	● cex=2.5
⊕ pch=9	b pch='b'	
⊕ pch=10	c pch='c'	● cex=3

Lines – lty and lwd





Colours

Black	col=1
Red	col=2
Green	col=3
Blue	col=4
Cyan	col=5
Purple	col=6
Yellow	col=7
Grey	col=8
Blue	col='blue'
50% grey	col=grey(0.5)

Other plotting commands

- text(x, y, “text”) … adds “text” at x, y
- abline(a, b) … adds line of slope b, intercept a
- abline(h=y) … adds a horizontal line at y
- abline(v=x) … adds a vertical line at x
- polygon(x, y) … adds a polygon with corners at each x_i, y_i

  (x and y should be vectors)
- legend(x, y, text) … adds a legend at x,y
- title(“title text”) … adds a title

Adding an axis

- axis(side, pos, at, labels)
  - **side** =1 for bottom, 2 for left, 3 for top, 4 for right
  - **pos** = positioning of axis
  - **at** = where tick marks should be drawn
  - **labels** = what to write next to each tick mark
- e.g. axis(1, pos=0, at=0:5, labels=0:5)
  - This adds an x-axis at the horizontal location y=0 with tick marks at x=0,1,…,5 and corresponding labels.
- For a y-axis add the option “las=1” to get horizontal rather than vertical text labels (looks nicer!)

Further customisation

- par() … used to access and modify graphics parameters for the current device
- par(col="red", lty=2) … change the default colour to red and the default line type to 2 (dotted)
- par(mar=c(5,4,1,1)) … change the default plot margins to 5,4,1,1 on the bottom, left, top, right (the default is 5,4,2,2).
- cex.axis, cex.lab, cex.main
  - Character expansion (relative sizes) for axis annotation, axis titles and main titles respectively

Plot windows

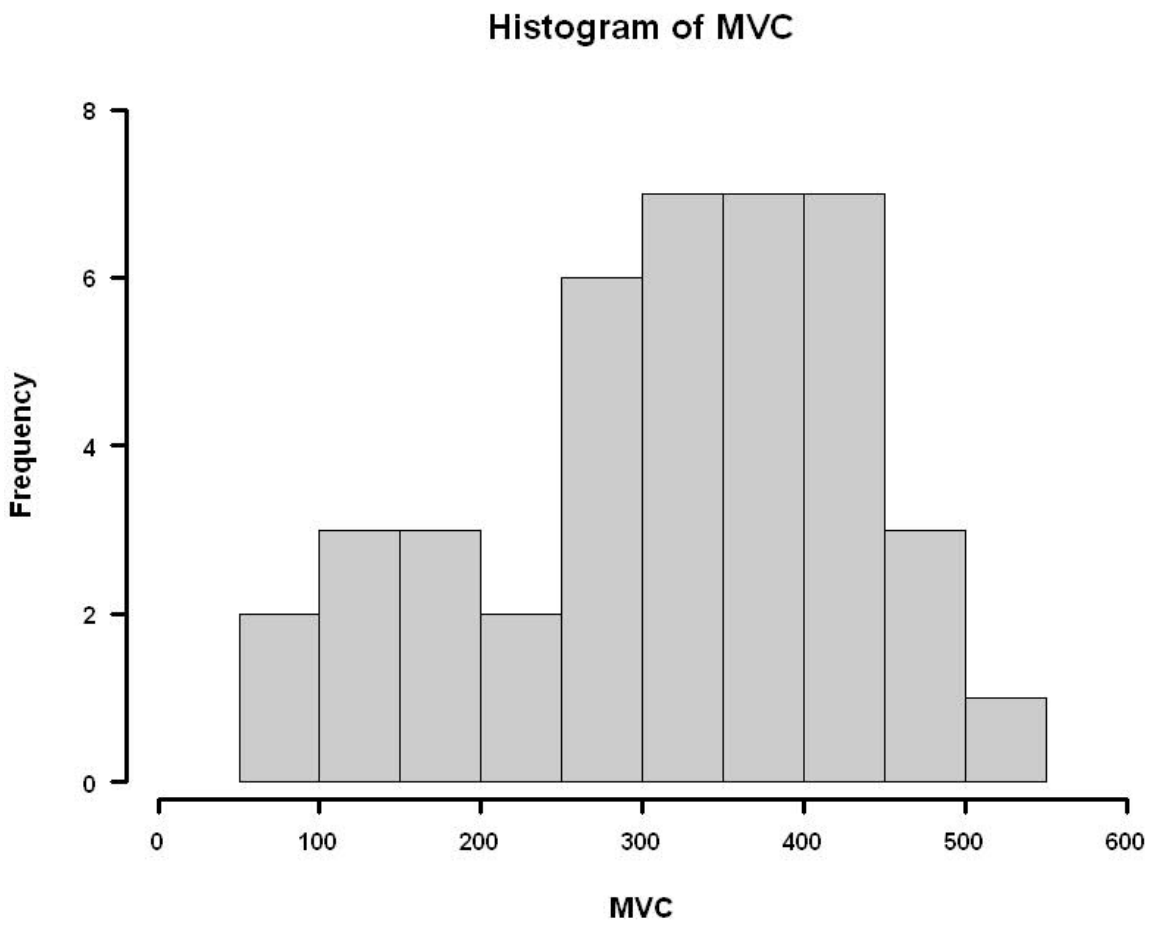
- windows(width=6, height=5)
  - Opens a new graphics window with width=6 inches and height=5 inches
  - quartz("Quartz", width=6 , height=5) [for Mac OS]
- layout(matrix(1:4, nrow=2, byrow=TRUE), widths=c(1,1), heights=c(1,3))
  - Split the graphics window into 4 subplots where the top two occupy 25% and bottom two occupy 75% of the plot region, respectively
- par(mfrow=c(2,2)) / par(mfcol=c(2,2))
  - Split the graphics window evenly into 4 subplots, shown in an order by rows (mfrow) or columns (mfcol)

Example – MVC data

- mvc <- read.csv("http://web.hku.hk/~ehylau/mvc.csv")
- Data includes ages, heights and maximum voluntary contraction of the quadriceps muscle (MVC) in a group of male alcoholics
- Can MVC be predicted from the other variables / what is the association between them?

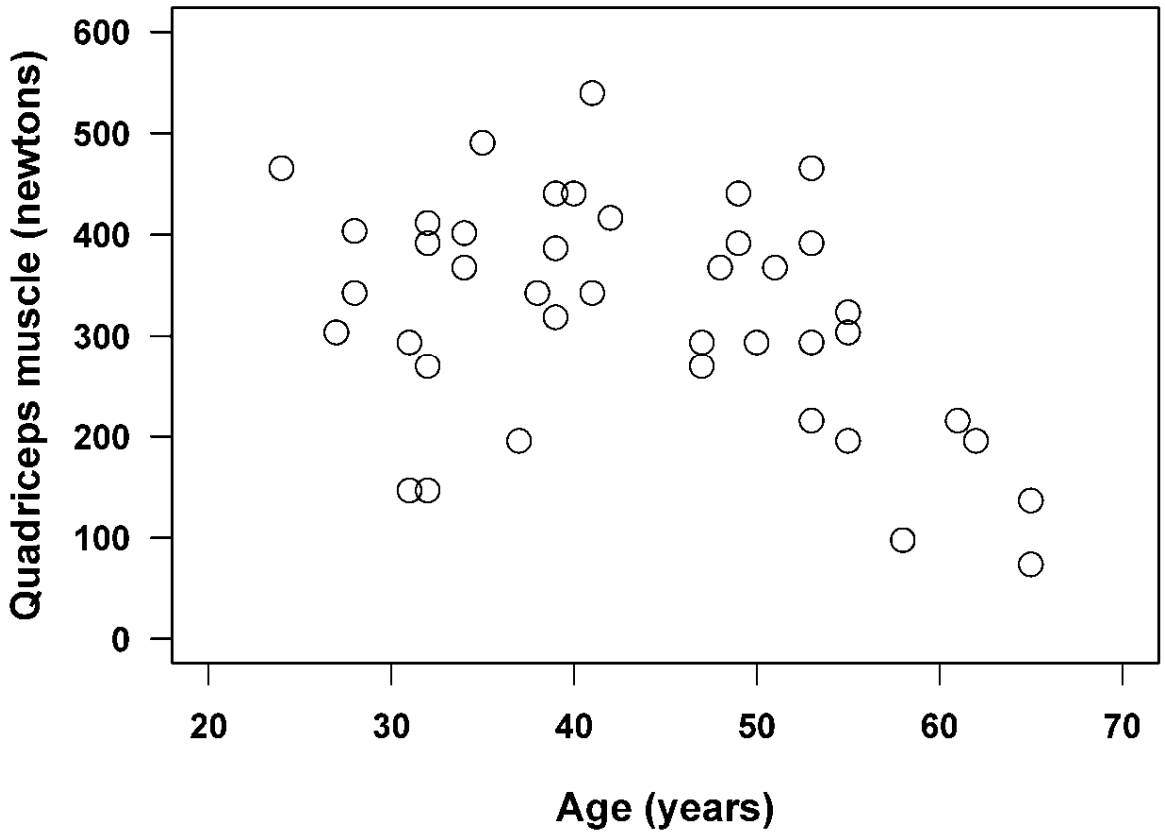
Plot a histogram

```
hist(mvc$MVC, axes=FALSE,
   xlim=c(0, 600), ylim=c(0,
   8), font.lab=2,
   cex.lab=1.2, cex.main=1.5,
   col=grey(0.8),
   xlab="MVC",
   ylab="Frequency",
   main="Histogram of MVC")
axis(1, pos=-0.2, lwd=3.5,
   font=2)
axis(2, pos=-20, lwd=3.5,
   font=2, las=1)
```



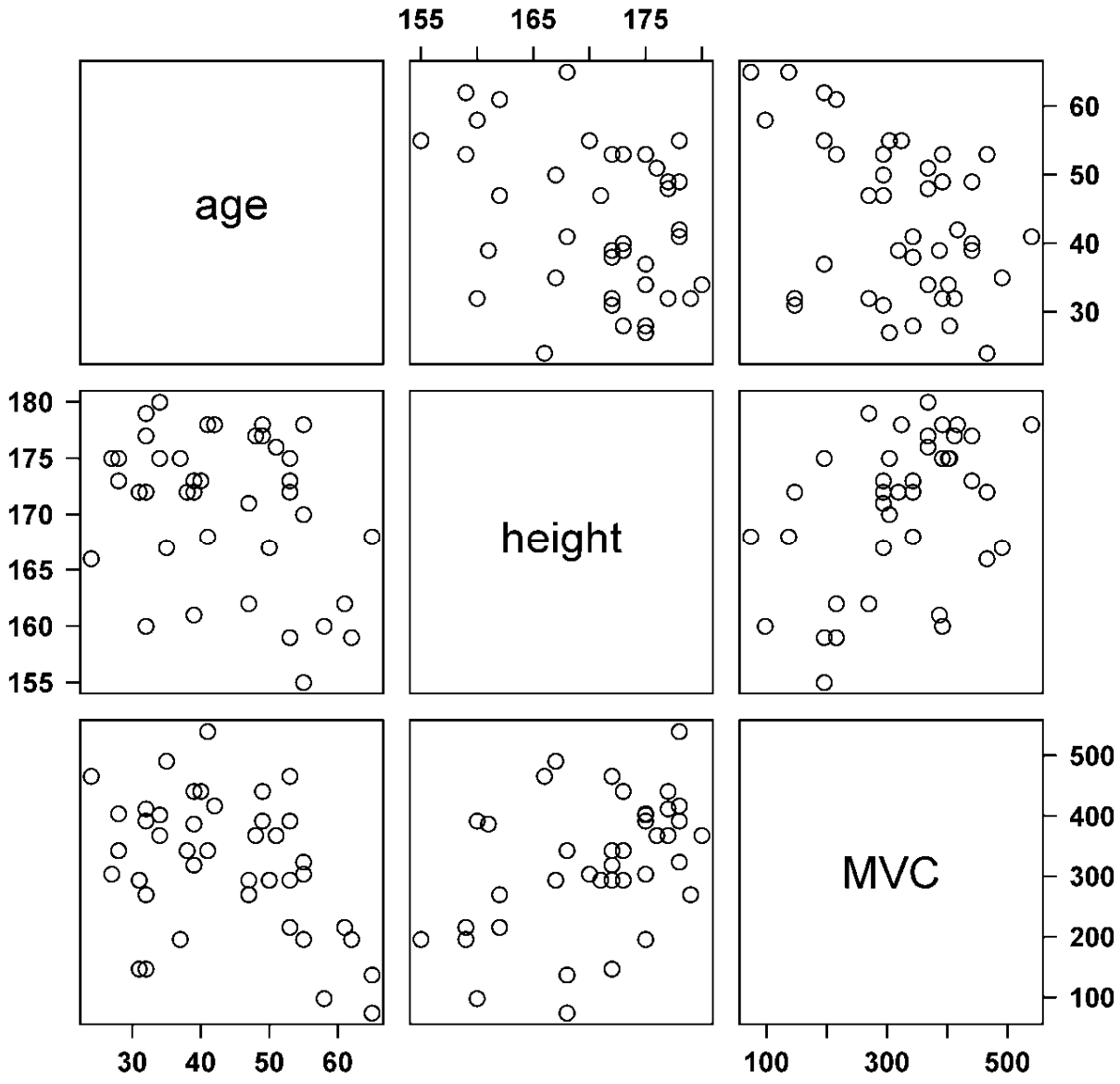
Plot two variables side by side

```
plot(MVC ~ age, data=mvc,
   xlab="Age (years)",
   ylab="Quadriceps muscle
   (newtons)", xlim=c(20,
   70), ylim=c(0, 600),
   cex=1.5, cex.lab=1.2,
   font.lab=2, font.axis=2,
   las=1)
```



Scatter plot matrix

```
pairs(mvc, cex =1.5,
   font=2, las=1,
   cex.axis=1.2)
```



Save graphical output

- pdf(*file*, *width*, *height*)
  - this will start the graphics device driver
  - **file** is the filename and its directory
- can also save in other format: e.g. bmp, jpeg, png, tiff
- dev.off()
  - close the graphics device
- Example:

```
pdf("d:/figure1.pdf", width=6, height=4)
plot(MVC ~ age, data=mvc, xlab="Age (years)", ylab="Quadriceps
   muscle (newtons)", xlim=c(20, 70), ylim=c(0, 600), cex=1.5,
   cex.lab=1.2, font.lab=2, font.axis=2, las=1)
dev.off()
```

Graphing package: ggplot2 syntax structure

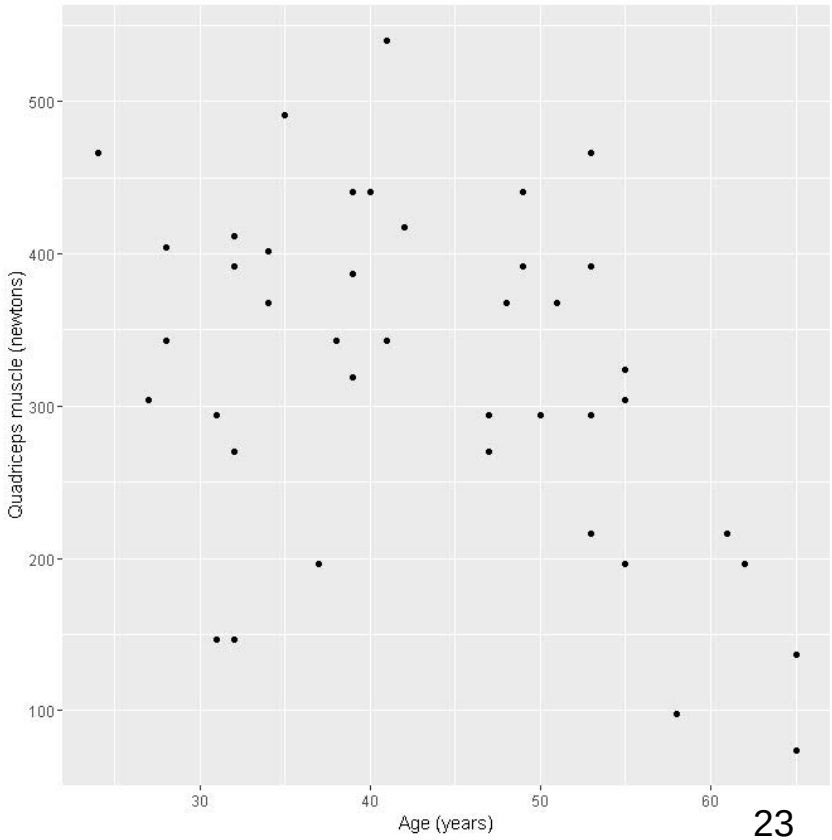
- gg: grammar of graphics
- basic structure: aesthetics and geometric shapes
- asthetics
  - x, y position, size/shape/color of elements
- geometric shapes
  - points, lines, line segments, bars, text
- a nice reference for ggplot2: http://www.cookbook-r.com/Graphs/index.html

ggplot: scatter plot

- Example: ggplot(mvc, aes(x=age, y=MVC)) + geom_point() + xlab("Age (years)") + ylab("Quadriceps muscle (newtons)")

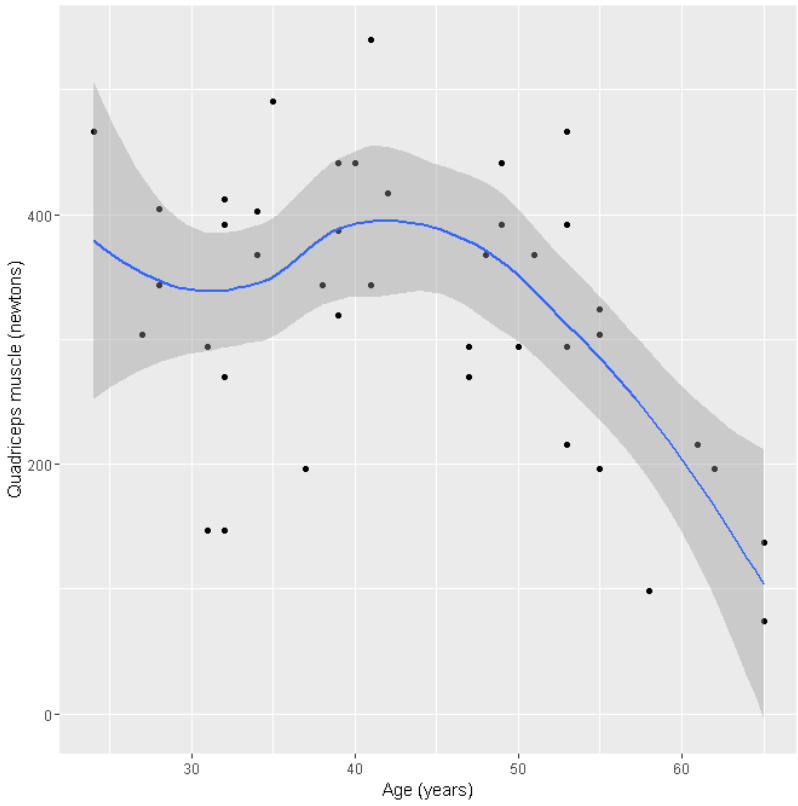
Or in separate steps:

- p <- ggplot(mvc, aes(x=age, y=MVC))
- p + geom_point() + xlab("Age (years)") + ylab("Quadriceps muscle (newtons)")



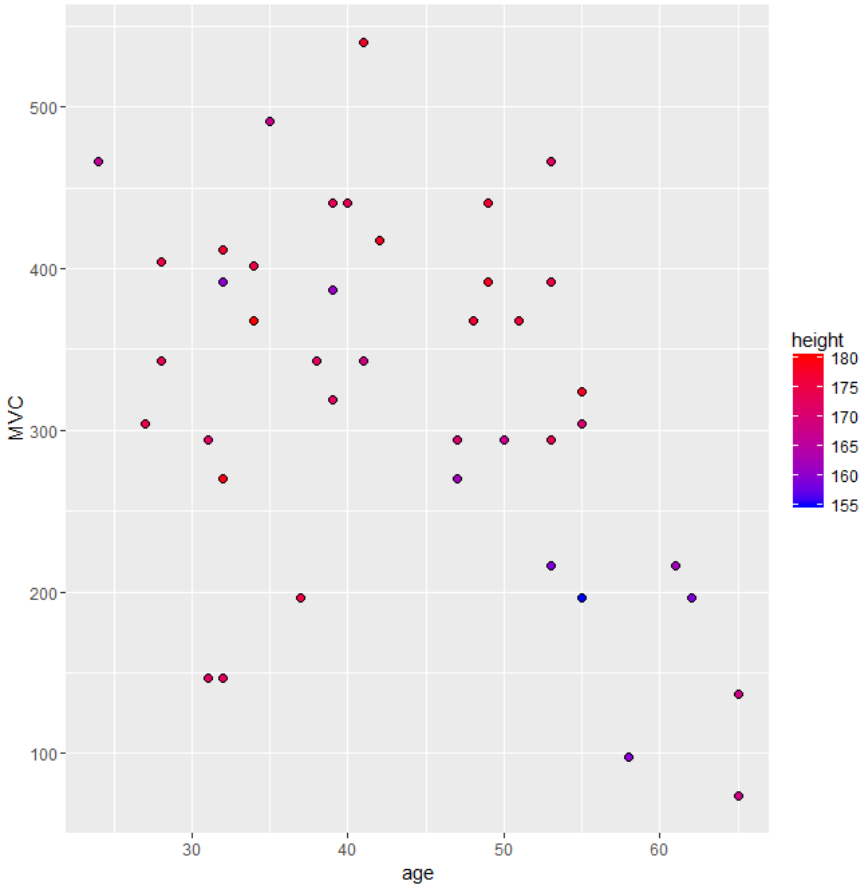
Some variations

- p <- ggplot(mvc, aes(age, MVC)) + xlab("Age (years)") + ylab("Quadriceps muscle (newtons)")
- p + geom_point() + stat_smooth(method=loess)



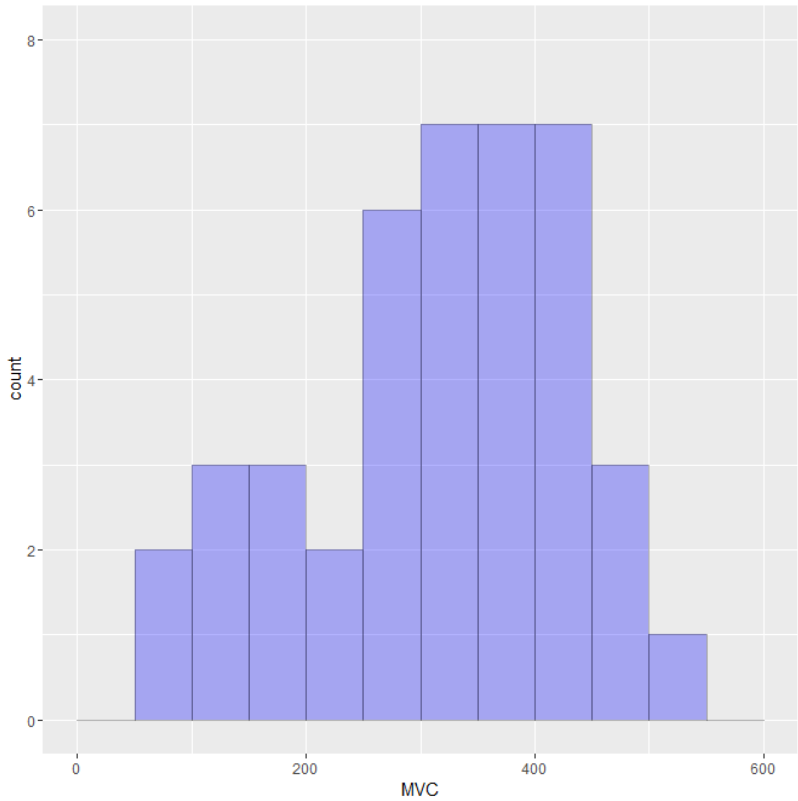
Some variations

- ggplot(mvc, aes(x=age, y=MVC, fill=height)) + geom_point(shape=21, size=2) + scale_fill_gradient(low='blue', high='red')



ggplot histogram for MVC

- Ploting a histogram for the variable MVC, with a binwidth of 50
- ggplot(mvc, aes(MVC)) + geom_histogram(binwidth=50, boundary=50, col="black", fill="blue", alpha=0.3) + ylim(c(0,8))



Generating random numbers

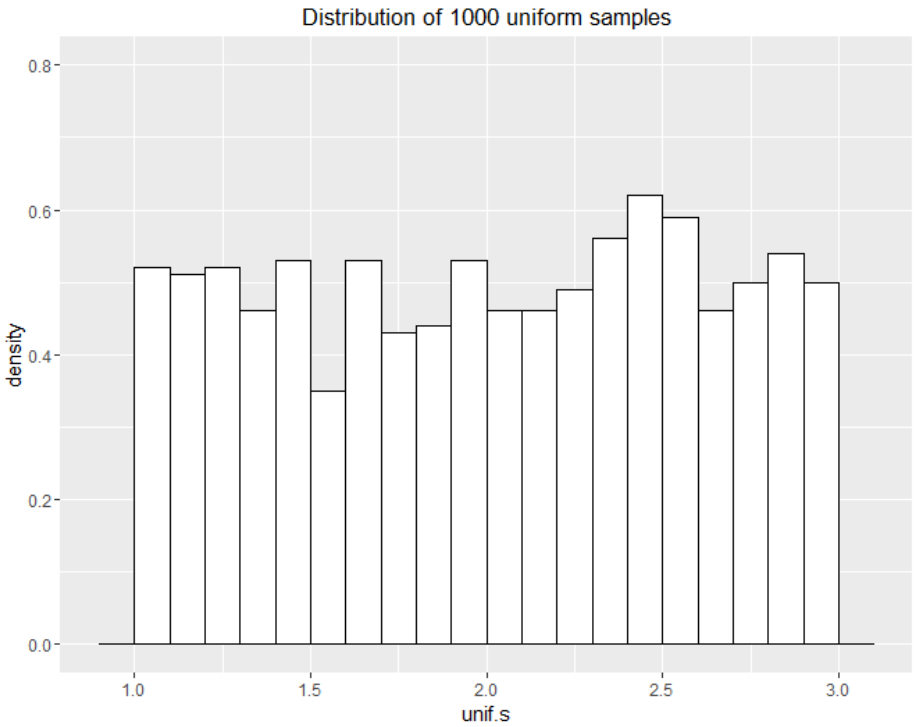
Generating Random Numbers

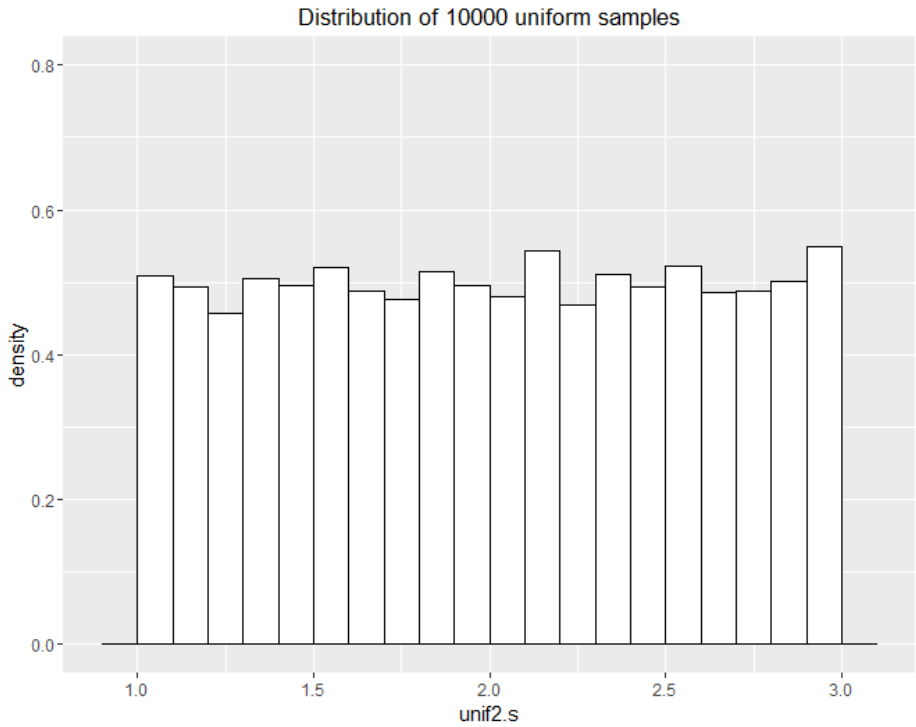
- Draw random samples from discrete / continuous probability distributions
- e.g. Uniform distribution
- runif(*n*, *min*, *max*)
  - **n** is the number of samples
  - **min / max** are the lower and upper limits of the uniform distribution
- Useful for simulation of random process

Uniform Distribution

Try to simulate and visualize 10000 or 100000 random numbers

```
unif.s <- runif(1000, min=1, max=3)
ggplot() + geom_histogram(aes(x=unif.s,
   y=..density..), binwidth=0.1, boundary=1,
   col="black", fill="white") + ggtitle('Distribution
   of 1000 uniform samples') + ylim(c(0,0.8))
```



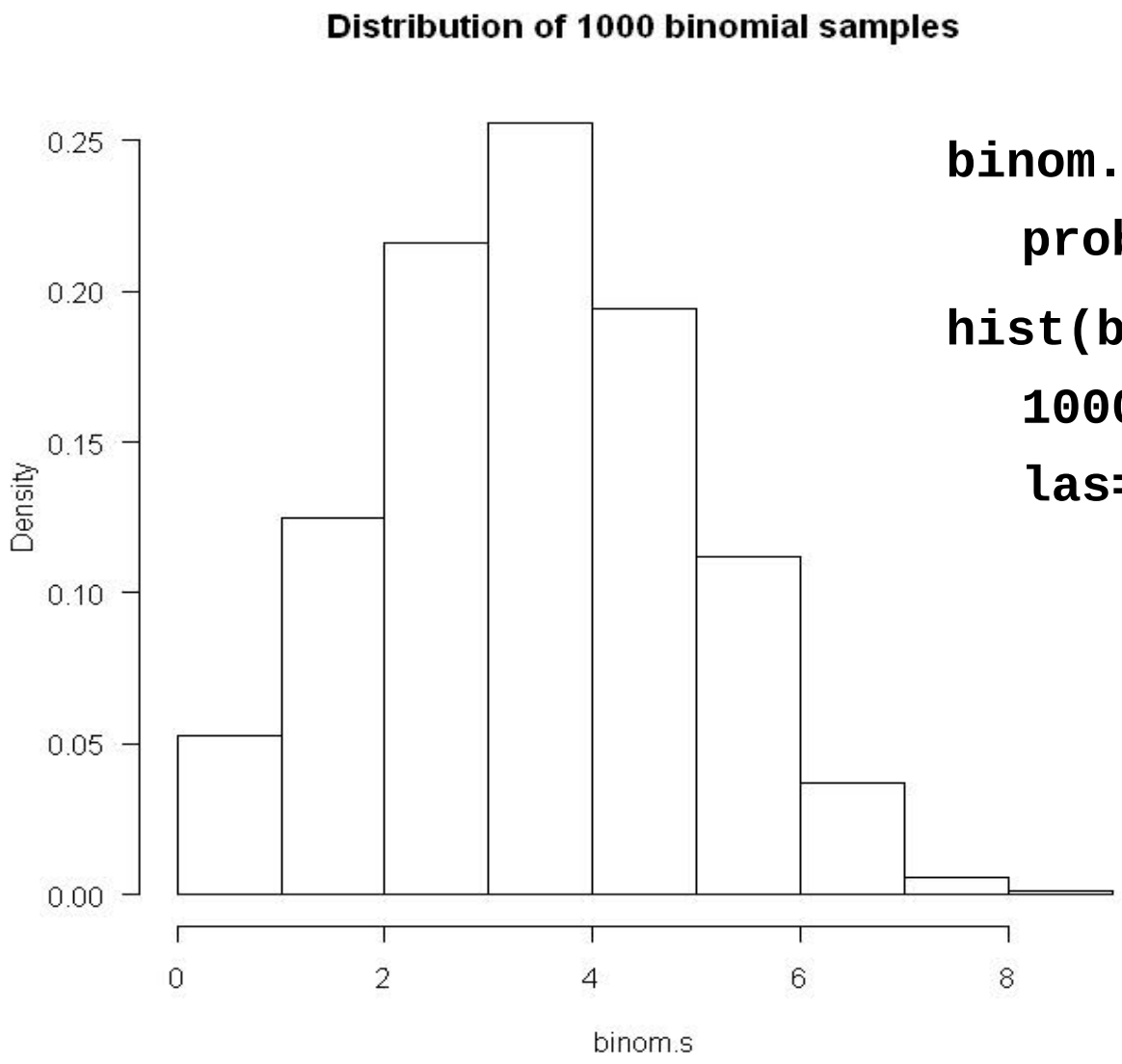


Binomial Distribution

- rbinom(*n*, *size*, *prob*)
  - **n** is the number of samples
  - **size** is the number of trials
  - **prob** is the probability of success for each trial

```
binom.s <- rbinom(1000, size=10, prob=0.4)
hist(binom.s, main="Distribution of 1000 binomial
   samples", freq=F, las=1)
```

Binomial Distribution



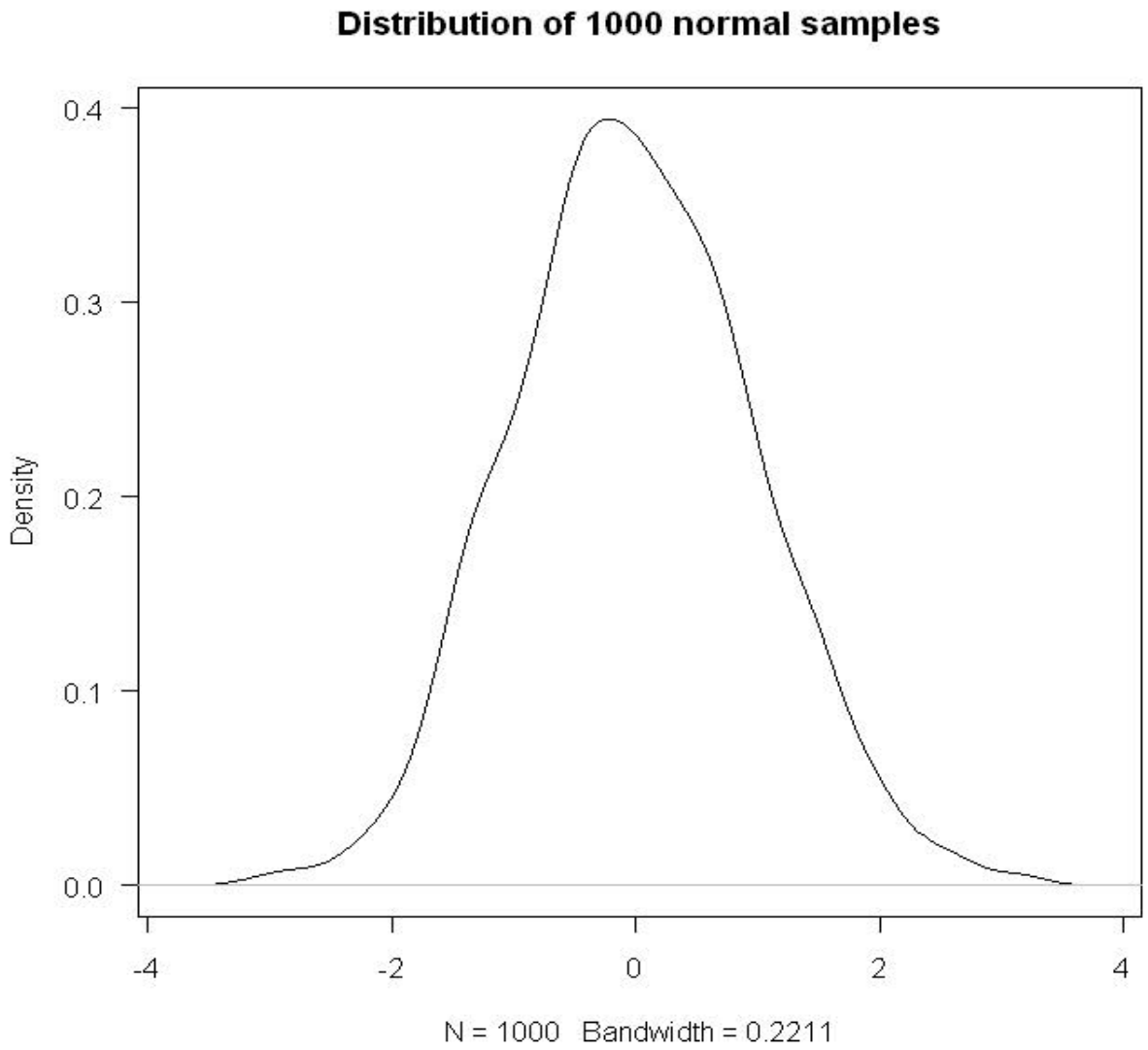

```
binom.s <- rbinom(1000, size=10,
   prob=0.4)
hist(binom.s, main="Distribution of
   1000 binomial samples", freq=F,
   las=1)
```

Normal Distribution

- rnorm(*n*, *mean*, *sd*)
  - **n** is the number of samples
  - **mean** is the mean / vector of means
  - **sd** is the standard deviation / vector of standard deviations

```
norm.s <- rnorm(1000, mean=0, sd=1)
plot(density(norm.s), main="Distribution of 1000 normal
   samples", las=1)
```

Normal Distribution




```
norm.s <- rnorm(1000, mean=0,
   sd=1)
plot(density(norm.s),
   main="Distribution of 1000
   normal samples", las=1)
```

Random samples

- sample(*x*, *size*, *replace*)
  - **x** is a vector from which samples are drawn
  - **size** is the number of samples to be drawn
  - **replace** indicates if the samples are drawn with replacement

```
sample(seq(1,50), 10, replace=F)
[1] 40 21 33 13 31  7 23 49 39 35
sample(seq(1,50), 10, replace=T)
[1] 15 45 35 28 28 38 49 29  5  4
```

Specify “seed” for randomization

- set.seed(*seed*)
  - **seed** is an integer which specifies the state of random number generator

```
runif(3)
[1] 0.7264811 0.3704220 0.5149238
runif(3)
[1] 0.53229524 0.43216062 0.09368152

set.seed(111)
runif(3)
[1] 0.5929813 0.7264811 0.3704220
set.seed(111)
runif(3)
[1] 0.5929813 0.7264811 0.3704220
```

Linear regression in R

Recap of linear regression

- If Y is the response variable and X is the explanatory variable, then a simple linear regression states that

$$Y = \alpha + \beta X + \varepsilon$$

- ε is distributed $N(0, \sigma^2)$, so that

$$Y \sim N(\alpha + \beta X, \sigma^2)$$

- When there is more than one explanatory variable, i.e. more than one X, then the linear regression is not “simple”, but “multiple”.
- Now $Y \sim N(\alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k, \sigma^2)$
- Can also include interactions, i.e. terms where two or more of the explanatory variables are multiplied together, and higher-order powers of the variables, e.g. X_1^2, $X_1 X_2$.

Regression in R

lm(*formula*, *data*, *subset*, *weights*, *na.action*…)

- ***formula*** is written as y~x+z
- ***data*** indicates the data frame to be used
- ***subset*** indicates that only some of the data should be used, e.g. *subset=(gender=="M")*
- ***weights*** weights the observations
- ***na.action*** describes what to do with missing data. The default is *na.omit* (listwise deletion)

Formula format

Term	Interpretation	Example
A+B	Include both A and B	height + age
A-B	Exclude B from A	height*age – height:age
	Dropping the intercept	height + age - 1
A:B	Interaction of A and B	height:age
A*B	A + B + A:B	height*age
A+B+C+…	remaining variables	.

R output

```
> summary(lm(MVC ~ height + age, data=mvc))
or
> mvc.lm <- lm(MVC ~ height + age, data=mvc)
> summary(mvc.lm)

Call:
lm(formula = MVC ~ height + age, data = mvc)

Residuals:
     Min        1Q   Median        3Q       Max
-220.523  -33.483    5.665   50.917  170.841

...
```

R output

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -465.626    460.333  -1.011   0.3182
height         5.398      2.545   2.121   0.0405 *
age           -3.075      1.467  -2.096   0.0428 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 98.92 on 38 degrees of freedom
Multiple R-squared:  0.2612, Adjusted R-squared:  0.2224
F-statistic: 6.719 on 2 and 38 DF,  p-value: 0.003173
```

The estimated regression equation is:

$$MVC = -465.63 + 5.40 \times height - 3.08 \times age$$

Convert numeric to factor

cut(*x*, *breaks*, *labels* = NULL, *include.lowest* = FALSE, *right* = TRUE, ...)

- ***x*** is the variable to be converted to factor
- ***breaks*** defines the cut point
- ***labels*** gives the label for each category level
- ***right*** = TRUE indicates the intervals should be close on the right, i.e., intervals of ( ].
- ***include.lowest*** = TRUE includes values at the first (when ***right*** = TRUE) or last (when ***right*** = FALSE) cut point with an open interval

Examples: cut function

```
> x = 1:8
> cut(x,c(1,3,6,8))
 [1] <NA>  (1,3] (1,3] (3,6] (3,6] (3,6] (6,8] (6,8]
Levels: (1,3] (3,6] (6,8]
```

Default value: *include.lowest* = FALSE, *right* = TRUE

```
> cut(x,c(1,3,6,8), include.lowest=T)
[1] [1,3] [1,3] [1,3] (3,6] (3,6] (3,6] (6,8] (6,8]
Levels: [1,3] (3,6] (6,8]
```

Examples: cut function

```
> x = 1:8
> cut(x, c(1,3,6,8), include.lowest=T, right=F)
[1] [1,3) [1,3) [3,6) [3,6) [3,6) [6,8] [6,8] [6,8]
Levels: [1,3) [3,6) [6,8]

> cut(x, c(1,3,6,8), include.lowest=T)
[1] [1,3] [1,3] [1,3] (3,6] (3,6] (3,6] (6,8] (6,8]
Levels: [1,3] (3,6] (6,8]

> cut(x, c(1,3,6,8), label=c('low','med','high'),
   include.lowest=T, right=F)
[1] low  low  med  med  med  high high high
Levels: low med high
```

Reordering categorical variables

relevel(*x*, *ref*)

- ***x*** is the categorical variable to be re-ordered

factor(*x*, *levels*)

- ***x*** is the categorical variable to be re-ordered
- ***levels*** is the specified order

Examples: reordering factor levels

```
> x.f <- cut(x,c(1,3,6,8), label=c('low','med','high'),
   include.lowest=T, right=F)
> x.f
[1] low  low  med  med  med  high high high
Levels: low med high

relevel(x.f, ref='med')
[1] low  low  med  med  med  high high high
Levels: med low high

> factor(x.f, c('high','med','low'))
[1] low  low  med  med  med  high high high
Levels: high med low
```

Confidence intervals for the estimated coefficients

- The two-sided 100(1-α)% confidence interval (CI) of an estimated coefficient is given by

$$\beta \pm t(1-\alpha/2, n-k) \times s_\beta$$

- where t(1-α/2, n-k) is the critical value of the student's t distribution with degree of freedom n-k
  - for α = 0.05, t(0.975, n-k) → z(0.975) = 1.96 when n is large
- k is the number of parameters in the model
- s_β is the standard error of the estimated coefficient

Confidence intervals for the estimated coefficients

To obtain 95% CI for β_{height}:

```
> confint(mvc.lm)
                    2.5 %       97.5 %
(Intercept) -1397.5226142 466.2701576
height          0.2460489  10.5503147
age            -6.0459158  -0.1048933
```

```
> summary(mvc.lm)

Call:
lm(formula = MVC ~ height + age, data = mvc)

Residuals:
     Min       1Q   Median       3Q      Max
-220.523  -33.483    5.665   50.917  170.841

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -465.626    460.333  -1.011   0.3182
height         5.398      2.545   2.121   0.0405 *
age           -3.075      1.467  -2.096   0.0428 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 98.92 on 38 degrees of freedom
Multiple R-squared:  0.2612,    Adjusted R-squared:  0.2224
F-statistic: 6.719 on 2 and 38 DF,  p-value: 0.003173
```

alternatively

```
> mvc.out <- summary(mvc.lm)
> mvc.out$coef[,"Estimate"] %o% c(1,1) + mvc.out$coef[,"Std.
  Error"] %o% qt(c(0.025,0.975), mvc.out$df[2])
(Intercept) -1397.5226142 466.2701576
height          0.2460489  10.5503147
age            -6.0459158  -0.1048933
```

$$\beta \pm t(1-\alpha/2, n-k) \times s_\beta$$

Useful R materials

- Paradis E. R for beginners.
  http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

- Contributed packages. http://cran.r-project.org/web/packages

- Zheng T. Colors in R.
  http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf