Getting started checklist

- Have both R and R Studio access (can download these online)
- Can open RStudio,
  - (instructions on how to launch the virtual R)
- Have access to the location of the class folder with class material

Getting started (1)

- Open window explorer, find the class folder and right click on it

Getting started (2)

- R objects needed for the class have been prepared. They are all in a one file with extension Day2.Rdata in folder Day2_Data

The R environment

Functions
(**recipe**)

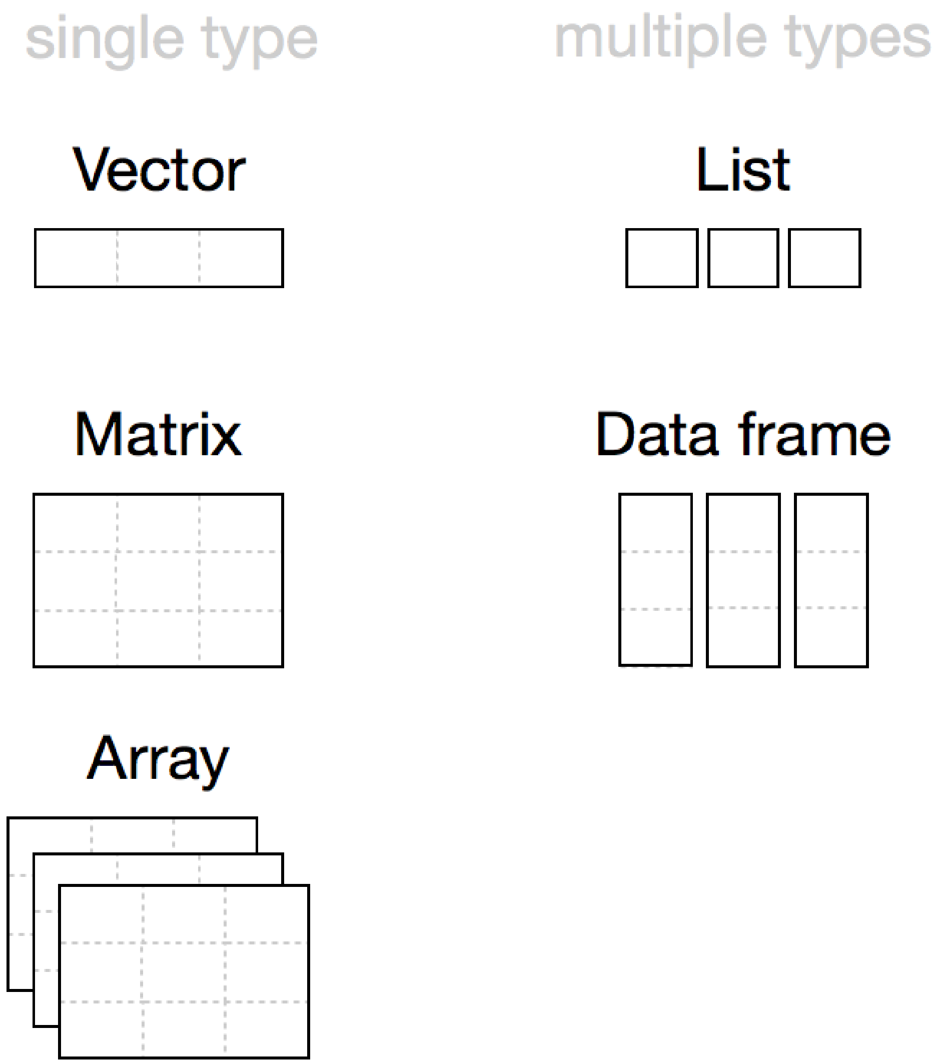
Data objects
(**ingredients**)

The R environment

- R programming is known as object-oriented programming, everything in R is an object
- Objects have characteristics known as attributes
- Objects have an associated label or name
- Objects can be divided into two
  - Functions (Recipes)
  - Data objects (ingredients)

Data Structures

- Basic data types
  - Character 'a', '1', '*', '/'
  - Integer e.g. -1, 0, 1, 2, …
  - Numeric e.g. -2.4467 (includes integers)
  - Boolean (TRUE, FALSE)
- Structured data
  - Vectors
  - Matrix
  - List (mix of data types)
  - Data frame

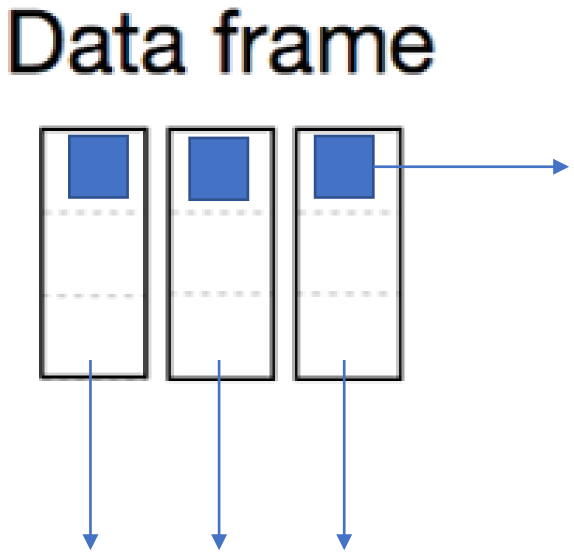


Functions (“Recipes”)

- Takes in R data structures/element and ‘do something’ to produces other data structures/element
- E.g. most basic function: + , - , /, *
- Some other functions: `floor()`, `ceiling()`, `round()`, `mean()`, `var()`, `cor()`, `plot()`
- Class focused on functions that can help you organize and analyze your data

Data frame objects

- Most convenient way of storing data
- Data-frame consists of vectors(columns) of the same length
- Vectors can be of different types
- Columns have names
- Row’s can have names



Column names

Vectors of the same length, elements within a vector are the same type

Class exercise 1: Inbuilt dataframes

- Click on the environment tab on the top right side
- Click on the down arrow next to the purple “*P*”
- You should get a list of inbuilt datasets

  Click on **`chickwts`**
- You can learn more about the dataset by entering the following in the console window

  `?chickwts`
- Click on the dataset **`cars`** find out more about this data frame

Review: Referencing for dataframes

Referencing a single column:

```
> cars$speed
 [1]  4  4  7  7  8  9 10 10 10 11 11
[27] 16 16 17 17 17 18 18 18 18 19 19
> cars[,1]
 [1]  4  4  7  7  8  9 10 10 10 11 11
[27] 16 16 17 17 17 18 18 18 18 19 19
> cars[,'speed']
 [1]  4  4  7  7  8  9 10 10 10 11 11
[27] 16 16 17 17 17 18 18 18 18 19 19
> 
```

Referencing a sequence of columns:

```
> cars[ ,1:2]
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
```

or

```
> cars[, c('speed','dist')]
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
```

Review: Referencing for dataframes

Referencing a specific row using the row number

```
> cars[1,]
  speed dist
1     4    2
> cars[8,]
  speed dist
8    10   26
```

Referencing a row using sequences of rows numbers

```
> cars[c(1:2, 8:10), ]
   speed dist
1      4    2
2      4   10
8     10   26
9     10   34
10    11   17
> cars[c(1, 2, 8, 9, 10), ]
   speed dist
1      4    2
2      4   10
8     10   26
9     10   34
10    11   17
> 
```

Review: Referencing for dataframes

- Boolean arguments for referencing, want to select all cars with speed of 20
- Step 1

```
> cars.selection <- cars$speed==20
> cars.selection
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[14] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[27] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[40]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

- Step 2

```
> cars[ cars.selection, ]
   speed dist
39    20   32
40    20   48
41    20   52
42    20   56
43    20   64
```

R functions for manipulating data frames

- The **attach()** function allows you to reference copies of the columns of a data frame directly,

```
> attach(cars)
> speed
 [1]  4  4  7  7  8  9 10 10 10 11 11 12 12 12 12 13 13 13 13 14
[27] 16 16 17 17 17 18 18 18 18 19 19 19 20 20 20 20 20 22 23 24
> dist
 [1]   2  10   4  22  16  10  18  26  34  17  28  14  20  24  28
[20]  26  36  60  80  20  26  54  32  40  32  40  50  42  56  76
[39]  32  48  52  56  64  66  54  70  92  93 120  85
> travel.time <- dist*0.000189394/speed
>
```

To stop referencing columns names directly use **detach()**

```
> detach(cars)
> speed
Error: object 'speed' not found
>
```

Case for cleaning data

- Ensure relevant data is accessible to software, i.e.you can read your data into R
- Standardization for example missing data is denoted the same way
- Detect and correct errors
- Portability across software
- Assess the quality of your data and if this might impact your analysis

Some pointer when data cleaning

- Study units e.g. patient, encounter, hospital are uniquely identifiable
- Variable information in columns, study units information in rows
- Rows in dataset are unique and identifiable
- Columns contain unique information on the study unit
- Consistent and specific method of denoting missing data
- Consistent data format, e.g. if date format is mm/dd/yyyy do not use dd/mm/yy or dd/mm/yy
- No empty rows/no empty columns
- Always have each dataset having a data dictionary indicating coding, date formats, calculations e.t.c.

Class exercise 2: Data cleaning exercise

- From the dataset pick out three issues

StudyID	1786	1953	1431	1650	1814
DOB	4/24/21	7-15-21	Mar/9/2021	9/8/20	7/11/21
Race	White	Other	white	Black or African American	Other
Insurance	Missing	Private	Public		Public

Functions for getting dataframe information

- **attributes()**

```
> attributes(cars)
$names
[1] "speed" "dist"

$class
[1] "data.frame"
```

- **summary()**

```
> summary(cars)
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
```

- **dim()**

```
> dim(cars)
[1] 50  2
```

R functions for manipulating data frames

Sorting a data frame using **`order()`** which returns the ranks of the data,

```
> order(cars$speed)
 [1]  1  2  3  4  5  6  7  8
[27] 27 28 29 30 31 32 33 34
> order(cars$dist)
 [1]  1  3  2  6 12  5 10  7
[27] 28 30 32 19 37 40 31 41
```

to sort the dataset we reference the rows in the order given

```
> cars.dist.order <- order(cars$dist)
> cars.dist.sort <- cars[cars.dist.order, ]
> cars.dist.sort
   speed dist
1      4    2
3      7    4
2      4   10
6      9   10
```

Can sort on multiple columns with subsequent columns used to break ties

```
> cars.all.order <- order(cars$dist, cars$speed)
> cars[cars.all.order, ]
```

R functions for recoding

- Can recode using variables which is in library car **`recode()`** , for examples in the cars dataset, create a new variable

  speed.level = High if speed > 15, otherwise "Low"

  = Low otherwise

```
> speed.recode <- car::recode(cars$speed, "0:14 = 'Low'; 15:120 = 'High' ")
> speed.recode
 [1] "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"
[12] "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"  "Low"
[23] "Low"  "High" "High" "High" "High" "High" "High" "High" "High" "High" "High"
[34] "High" "High" "High" "High" "High" "High" "High" "High" "High" "High" "High"
[45] "High" "High" "High" "High" "High" "High"
> cars2 <- cbind(cars, speed.recode)
```

R function for recoding

- The function **`cut()`** helps recode continuous variables into quantitative/class variable, for example to recode the vector below into two groups: less than or equal to 0.5 and greater than 0.5:

```
> random.vec <- c(0, 0.3, 0.2, 0.8, 1.2, 1.5)

> cut(random.vec, breaks = c(-1, 0.5, 1.5))
[1] (-1,0.5]  (-1,0.5]  (-1,0.5]  (0.5,1.5] (0.5,1.5] (0.5,1.5]
Levels: (-1,0.5] (0.5,1.5]
```

From this argument R creates categories (-1, 0.5] and (0.5, 1.5]

- Can have nicer labels for the categories

```
> cut(random.vec, breaks = c(-2, 0.5, 1.5), labels = c('0 - 0.5', '> 0.5'))
[1] 0 - 0.5 0 - 0.5 0 - 0.5 > 0.5   > 0.5   > 0.5
Levels: 0 - 0.5 > 0.5
```

Class exercise on R function for recoding

- For data `chickwts` recode weight in the following groups:
    1. Less than and including the 1st quantile = Low
    2. Greater than the 1st quantile but less or equal to 3rd quantile = Medium
    3. Greater than the 3rd quantile = High

Data type Factors

In R vectors containing quantitative data can be coded as a type known as factors where information falls into distinct categories

- This format is to allow you to label categories which helps in data summaries
  - Enables you to change reference group in analyses involving categorical variables
  - Helps keep track of categories of interest, e.g. missing information
  - Some functions output this type of data
- You can change a numeric vector by using `factor()` or change back using `as.numeric()`

```
> Sex <- c( 1, 2, 1, 2, NA)
> factor(Sex)
[1] 1    2    1    2    <NA>
Levels: 1 2
> Sex.f <- factor(Sex, labels=c('Female', 'Male'))
> Sex.f
[1] Female Male   Female Male   <NA>
Levels: Female Male
> table(Sex.f)
Sex.f
Female   Male
     2      2
```

Summary statistics in R

For totals and means:

`sum()` **and** `mean( , na.rm=T),`

For variance:

`var(),`

For quantiles:

`quantile(),`

```
> attach(cars)
> mean(dist)
[1] 42.98
> sum(dist)/nrow(cars)
[1] 42.98

> var(dist)
[1] 664.0608
> sqrt(var(dist))
[1] 25.76938

> quantile(dist)
  0%  25%  50%  75% 100% 
   2   26   36   56  120 
> quantile(dist, 0.1)
 10% 
15.8 
```

Note: Modify function calls if you have missing data

Summary statistics in R

1. Frequency tables, use **`tables()`**, check for missing first!

```
> attach(chickwts)
> feed.table <- table(feed)
> feed.table
feed
   casein horsebean   linseed  meatmeal   soybean 
       12        10        12        11        14 
sunflower 
       12 
```

2. For better formatting, perform (1), then convert to data frame

```
> data.frame(feed.table)
       feed Freq
1    casein   12
2 horsebean   10
3   linseed   12
4  meatmeal   11
5   soybean   14
6 sunflower   12
```

Summary statistics in R

For cross-tabulation use `table()`, note the childHealth data has missing observations

```
> attach(childHealth)
> table(ACE3, ACE5, useNA = 'ifany')
        ACE5
ACE3     1  2 <NA>
  1      3 16    0
  2      0 74    0
  <NA>   0  1    6
```

Can also cross-tabulate > 2 variables,

```
> table(ACE3, ACE5, ACE6, useNA = 'ifany')
, , ACE6 = 1

      ACE5
ACE3    1  2 <NA>
  1     3  0    0
  2     0  1    0
  <NA>  0  0    0

, , ACE6 = 2

      ACE5
ACE3    1  2 <NA>
  1     0 15    0
  2     0 73    0
  <NA>  0  1    0

, , ACE6 = NA

      ACE5
ACE3    1  2 <NA>
  1     0  1    0
  2     0  0    0
  <NA>  0  0    6
```

`ftable()` is another cross-tabulation function try: `ftable(ACE3, ACE5, ACE6, exclude = NULL)`

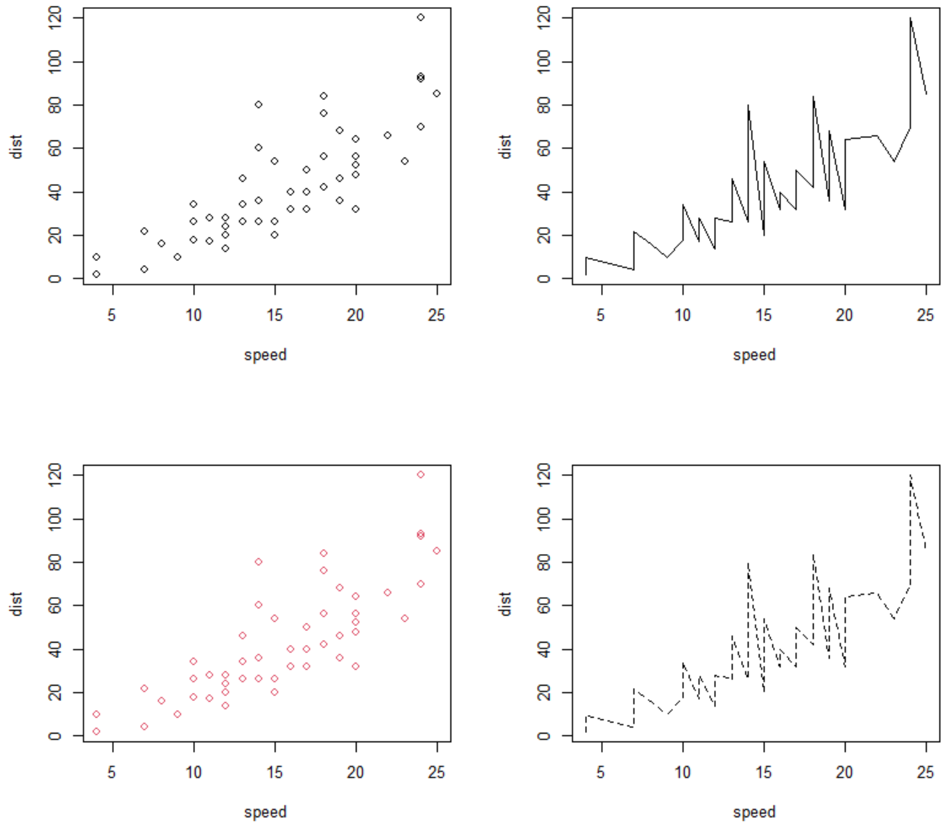
Quantitative data analysis

Scatter plot (scatter plot matrix)

plot(x, y,…)

```
> plot(speed, dist)
> plot(speed, dist, type='l')
> plot(speed, dist, col = 2)
> plot(speed, dist, type='l', lty=2)
```

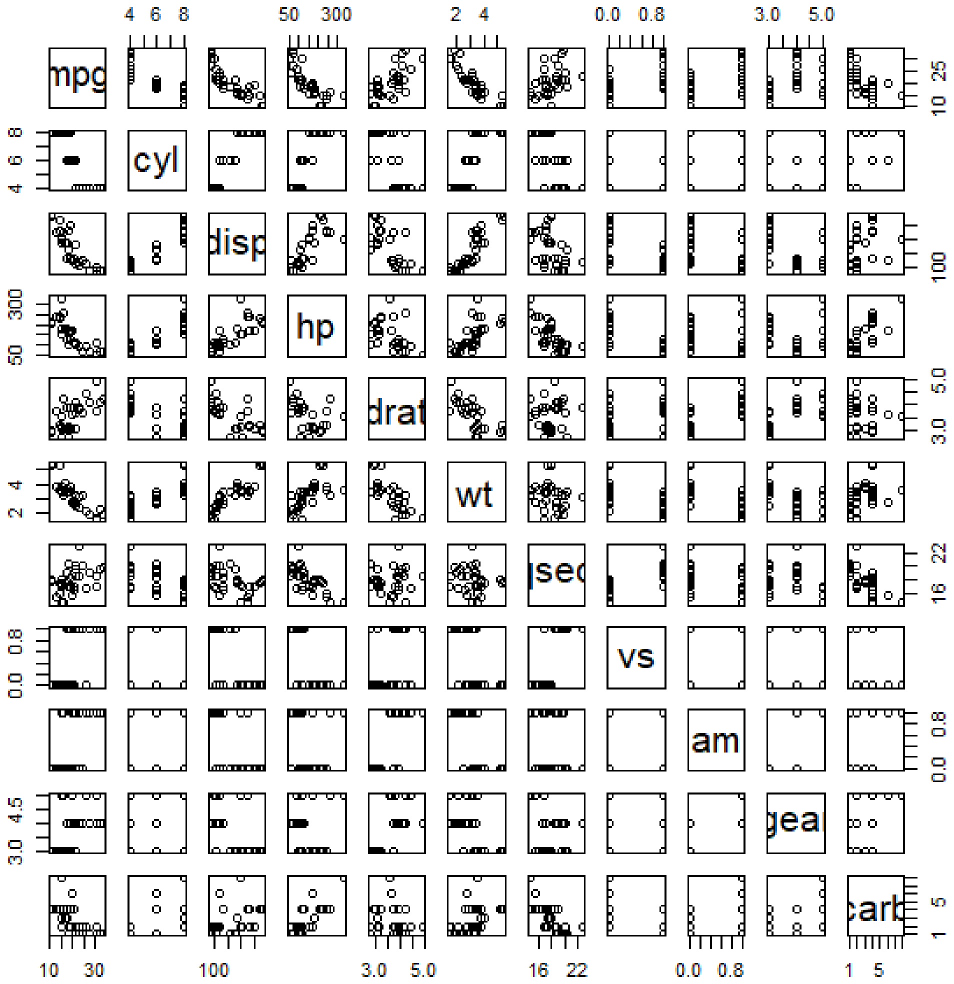
Caution for lines: Make sure data is sorted by x-axis variable



Analyzing quantitative data

A scatterplot matrix is a matrix of all possible pairwise scatter plots of a group of variables use **`pairs()`**

```
> pairs(mtcars)
```



Analyzing quantitative data

- Correlation assessed with the function **`cor()`**
  - Pearson measures linear association,
  - Spearman and is rank correlation,
- To test significance (null hypothesis that correlation coefficient equal to zero) use **`cor.test()`**

```
> cor(speed, dist)
[1] 0.8068949
> cor(speed, dist, method = 'spearman')
[1] 0.8303568
```

```
> cor.test(speed, dist, method='pearson')

        Pearson's product-moment correlation

data:  speed and dist
t = 9.464, df = 48, p-value = 1.49e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.6816422 0.8862036
sample estimates:
      cor
0.8068949
```

Analyzing quantitative data

- Can also obtain all pairwise correlation using `cor()` on a matrix or dataframe

```
> round( cor(mtcars), 2)
       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
mpg   1.00 -0.85 -0.85 -0.78  0.68 -0.87  0.42  0.66  0.60  0.48 -0.55
cyl  -0.85  1.00  0.90  0.83 -0.70  0.78 -0.59 -0.81 -0.52 -0.49  0.53
disp -0.85  0.90  1.00  0.79 -0.71  0.89 -0.43 -0.71 -0.59 -0.56  0.39
hp   -0.78  0.83  0.79  1.00 -0.45  0.66 -0.71 -0.72 -0.24 -0.13  0.75
drat  0.68 -0.70 -0.71 -0.45  1.00 -0.71  0.09  0.44  0.71  0.70 -0.09
wt   -0.87  0.78  0.89  0.66 -0.71  1.00 -0.17 -0.55 -0.69 -0.58  0.43
qsec  0.42 -0.59 -0.43 -0.71  0.09 -0.17  1.00  0.74 -0.23 -0.21 -0.66
vs    0.66 -0.81 -0.71 -0.72  0.44 -0.55  0.74  1.00  0.17  0.21 -0.57
am    0.60 -0.52 -0.59 -0.24  0.71 -0.69 -0.23  0.17  1.00  0.79  0.06
gear  0.48 -0.49 -0.56 -0.13  0.70 -0.58 -0.21  0.21  0.79  1.00  0.27
carb -0.55  0.53  0.39  0.75 -0.09  0.43 -0.66 -0.57  0.06  0.27  1.00
```

Randomization by example

I want to perform a study to investigate the effect of 4 types of feed additives (A, B, C, D) in the diet of mice. The study plans on using 32 lab mice, 16 males and 16 females as shown. I assign treatment as shown on the right.

Class discussion: Examine on the data, what is the problem with this treatment assignment?

ID	Sex	Weight	Tmt
1	M	276	A
2	M	292	A
3	M	297	A
4	M	345	A
5	M	360	A
6	M	369	A
7	M	270	A
8	M	284	A
9	M	303	B
10	M	332	B
11	M	301	B
12	M	250	B
13	M	263	B
14	M	285	B
15	M	285	B
16	M	180	B
17	F	278	C
18	F	273	C
19	F	278	C
20	F	276	C
21	F	278	C
22	F	270	C
23	F	276	C
24	F	276	C
25	F	276	D
26	F	273	D
27	F	275	D
28	F	271	D
29	F	273	D
30	F	282	D
31	F	269	D
32	F	268	D

Randomization by example: Complete Randomization

- A completely random assignment, we can use `sample()` to randomly assign the treatment

```
Tmt <-  c("A","B","C","D")
mice$Tmt.Rand1.a <- sample(Tmt, 32, replace=TRUE)
```

- What are the issues with this randomization approach?

Randomization by example: Stratified Randomization

- A random assignment within sex (stratified by sex),

```
mice$Tmt.Str.Rand <- c(sample(Tmt, 16, replace = TRUE),
    sample(Tmt, 16, replace = TRUE))
```

Still have issue with balance (tabulate assignment with sex)

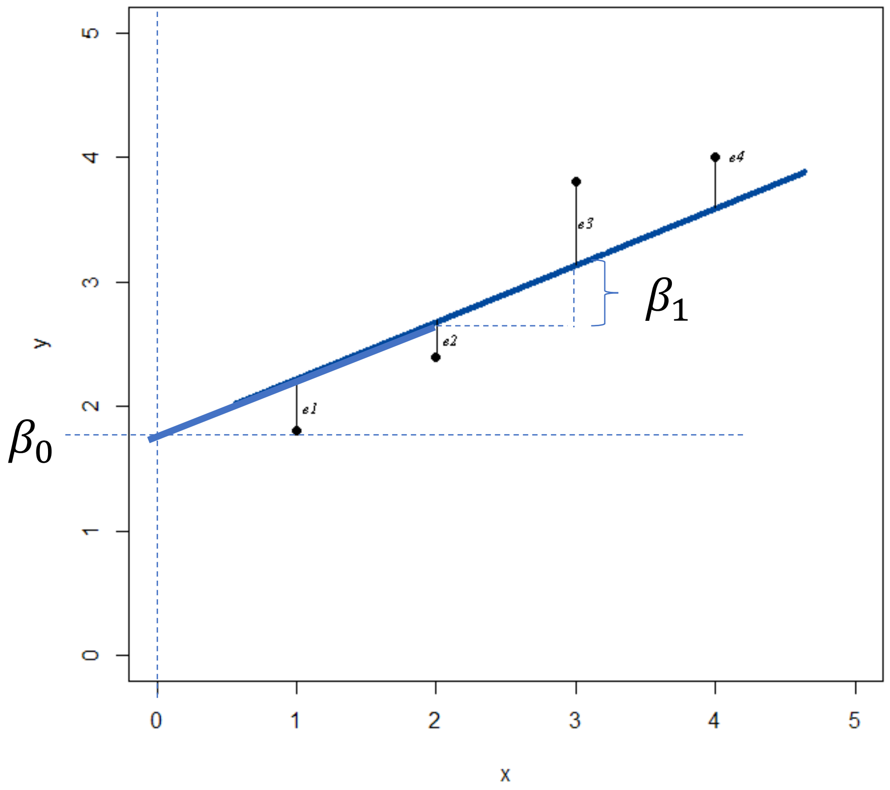
Randomization by example

- The function `runif()` generates random numbers in the interval 0-1,
- Randomization blocks of size four (stratified by sex) can help achieve balance:

```
Blocks <- rep(1:8, each=4 ) #create blocks of size four
r.unif <- runif(32)
Tmt.all <- rep(c("A","B","C","D"), 8)
mice$Tmt.Blk.rand <- Tmt.all[order(Blocks, r.unif)]
```

Check for balance (tabulate assignment with sex)

Simple linear regression



Linear regression: Finding a line that passes as close as possible to data points

A linear model $Y = \beta_0 + \beta_1 X + \epsilon$ be fit by Least Squares (LS) using the function **`lm()`**

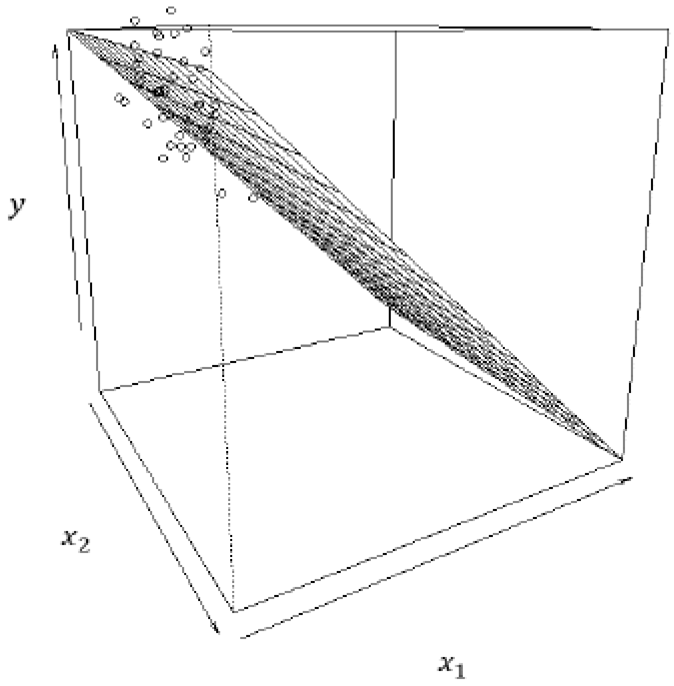
```
> lm(dist~speed, data = cars)

Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)        speed
    -17.579        3.932
```

Simple linear regression, LS regression output

- Inference from fit use **`summary()`** on fitted object

```
> cars.fit <- lm(dist~speed, data=cars)
> summary(cars.fit)

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-29.069  -9.525  -2.272   9.215  43.201 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -17.5791     6.7584  -2.601   0.0123 *  
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,	Adjusted R-squared:  0.6438 
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Multiple linear regression



Extension of simple linear regression

A linear plane $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$ can be obtained by least squares using the function `lm()` in a similar manner as the simple linear regression

Class Exercise

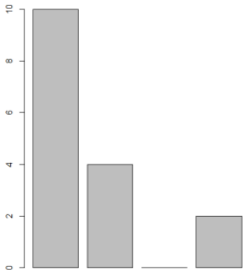
Data from the Berkeley guidance study of children born in 1928-29 in Berkeley, CA is in the library alr4 and also in the class folder. You can find out more about the dataset using ?BGSall

- Draw a scatterplot matrix of the data, use different colors for males and females, what conclusion do you draw about the variables,
- Estimate the correlation matrix for boys.
- For only boys, fit a simple linear regression model regressing Soma on WT9 and LG9. Is there evidence in the data that body type is determined by weight and or leg girth at age 9?

Categorical data analysis: Bar charts

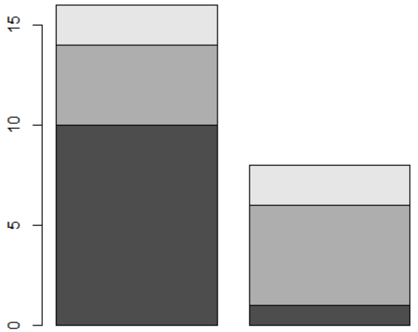
Bar chart, use **`barplot()`**, main argument is a vector, each element represents counts of a class

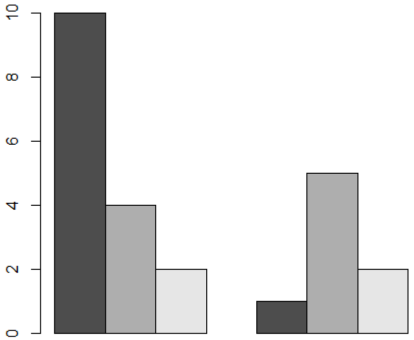
```
> vec1 <- c(10, 4, 0, 2)
> barplot(vec1)
```



Or a matrix, each row belonging to the same class

```
> mat1 <- cbind(c(10,4,2),c(1,5,2))
> barplot(mat1)
> barplot(mat1, beside=TRUE)
```

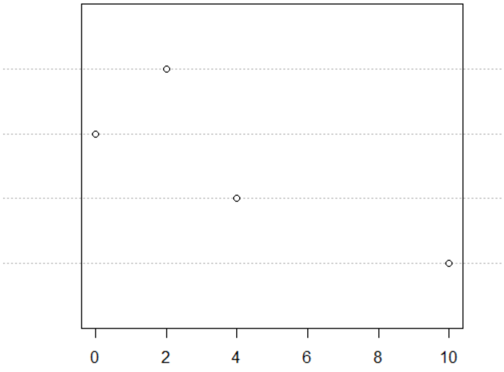


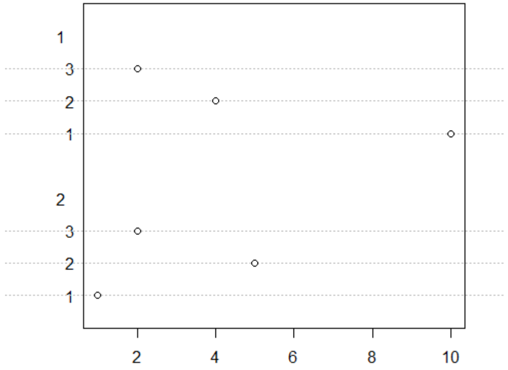


Categorical data analysis: Graphics (Dot plots)

`dotchart()` takes vector and matrix, performs group dot plot if main argument is matrix

```
> dotchart(vec1)
> dotchart(mat1)
```





Can also do grouped dot plots for a vector if grouping variable is specified

```
> dotchart(chickwts$weight, groups = chickwts$feed)
```

Bivariate association of quantitative with qualitative variables

- Difference in the mean in two populations using a t-test

```
> girls.Soma <- BGSall[ BGSall$Sex == 1,'Soma']
> boys.Soma <- BGSall[ BGSall$Sex == 0,'Soma']
> t.test(girls.Soma, boys.Soma)

        Welch Two Sample t-test

data:  girls.Soma and boys.Soma
t = 8.326, df = 100.41, p-value = 4.39e-13
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.285534 2.089790
sample estimates:
mean of x mean of y 
 4.778571  3.090909 

> t.test(Soma ~ Sex, data=BGSall)
```

- Analysis of variance for grouped response

```
> summary(anova.chicks)
            Df Sum Sq Mean Sq F value   Pr(>F)    
feed         5 231129   46226   15.37 5.94e-10 ***
Residuals   65 195556    3009                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Bivariate association quantitative with quantitative variables

- Qualitative response
  - Chi-square test for independence (in example: null depression is independent of parent divorce)

    `chisq.test()`

  - Fisher exact test (low frequency counts)

    `fisher.test()`

```
> ACE3.f <- factor(ACE3, labels=c('Divorce=Yes', 'Divorce=No'))
> Depression.f <- factor(K2Q32A, labels=c('Depression=Yes', 'Depression=No'))
> table(ACE3.f, Depression.f)
             Depression.f
ACE3.f        Depression=Yes Depression=No
  Divorce=Yes              2            17
  Divorce=No               1            73
> chisq.test(ACE3.f, Depression.f)

        Pearson's Chi-squared test with Yates' continuity correction

data:  ACE3.f and Depression.f
X-squared = 1.6674, df = 1, p-value = 0.1966

Warning message:
In chisq.test(ACE3.f, Depression.f) :
  Chi-squared approximation may be incorrect
> fisher.test(ACE3.f, Depression.f)

        Fisher's Exact Test for Count Data

data:  ACE3.f and Depression.f
p-value = 0.105
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.4108367 511.4464913
sample estimates:
odds ratio
  8.310283
```

Resources

- For any questions on any R-related topic, solution exists in stack exchange: https://stackexchange.com/
- Gentle introduction into using R to fit linear models: Applied Linear Regression : Statistics : University of Minnesota (umn.edu)