STAT5003 – Week 1

Review of Basic Statistical Concepts

Aims of this Course

- Learn **concepts** and **methods** related to statistical data analysis and statistical learning
- Apply the methods on **real datasets** in `R` (the statistical computing language)
- Aim is to teach you to use the methods, not necessarily to go into all the mathematical details

not a course that
looks into algebraic
concepts.

Basics of Statistical Computing

- Review basic statistical concepts
- Introduction to `R`
- Reproducible coding using Rmarkdown
- Recommended IDE via RStudio

Population

classical context: population of people.

- **Definition**:
  - The set of data (numeric or otherwise) corresponding to the entire collection of units about which information is sought
- **Examples**:
  - Blood pressure readings of **all** people in Australia
  - The number of languages spoken from **all** currently enrolled students in University of Sydney

doesn't have to be people...

Sample

- **Definition:**
  - A subset of the population data that are actually collected in the course of a study
- **Examples**:
  - Blood pressure readings of 1000 randomly selected people in Australia ✓
  - The number of languages spoken from 500 randomly selected students currently enrolled in University of Sydney ✓
    - In most studies, it is difficult to obtain information about the whole population. That is why we rely on samples to make estimates and inferences related to the whole population. ✓

Parameters vs. Statistic

- A **parameter** is a number that describes a population.
  - Notation usually denoted with Greek letters. e.g. μ, σ
- A **statistic** is a number that describes a sample.
  - Sample statistics are usually denoted using Roman letters, e.g. x, s.
- A parameter is a fixed number (usually unknown). A statistic is a variable whose value varies from sample to sample.

Descriptive Statistics: Numeric and Graphic

Numeric measures:

- Measure of location
  - Mean, Median, Mode for numeric data
  - Counts, proportions for categorical data
- Measure of spread
  - Standard deviation, MAD (median absolute deviation), IQR

never head of this...

- Others:
  - Min, Max, Quartile, Five number summaries (used later in boxplot)

↳ Q1, Q2, Q3, median etc.

THE UNIVERSITY OF
SYDNEY

Visualisation Packages

Types of Graphics Libraries Covered

- base `R` use the built in plotting functions → complex code required to make more beautiful visualisations in base R.
  - typically good for quick plots of simple datasets
- ggplot graphics `ggplot2`
  - Name meaning the grammar of graphics ✓
  - Typically better for more complicated datasets ✓
- Not covered but honorable mention with plotly
  - plotly is a powerful plotting library → more dynamic, through HTML browser.
  - Can do interactive graphics

Simple Example dataframe for Plots

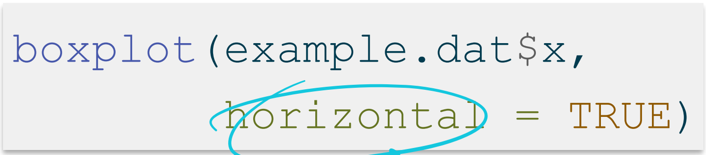
```
example.dat <- data.frame(x = rnorm(100),
                          y = runif(100),
                          cat = sample(LETTERS[1:2], prob =
c(1, 3), size = 100, replace = TRUE))
head(example.dat)
```

```
            x         y cat
1  0.37573068 0.8676167   A
2 -1.70387817 0.6760221   B
3 -1.64878643 0.7621811   A
4  0.09658172 0.2585820   B
5  0.74011371 0.2326891   A
6 -0.86970148 0.3605919   B
```

what does rnorm and runif do? Why are they different?

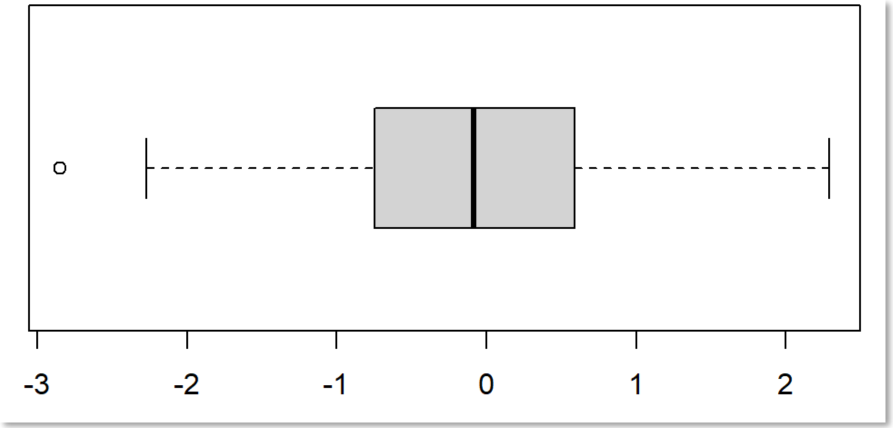
go through lab on lecture video...

Single Numeric Variable: Boxplot in base R

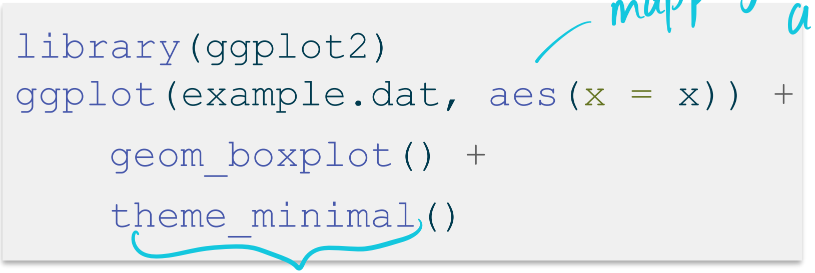


```
boxplot(example.dat$x,
        horizontal = TRUE)
```

by default, the boxplot is vertical



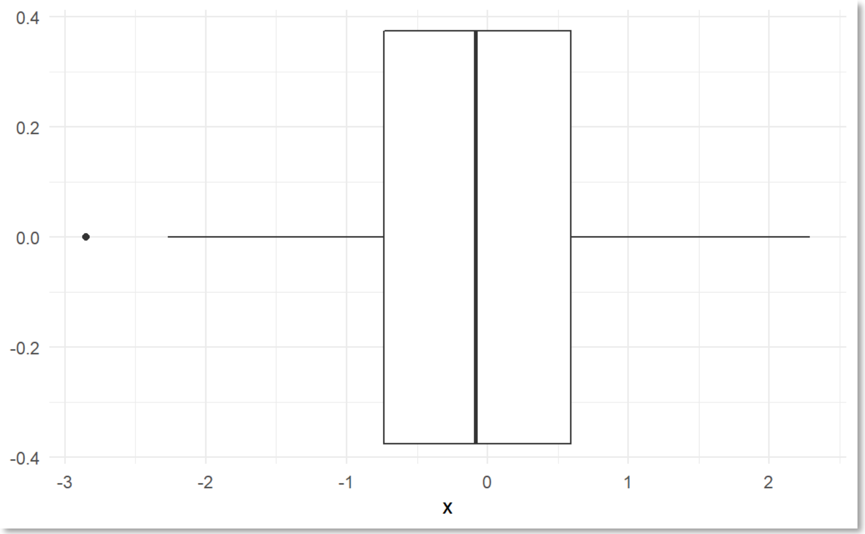
Single Numeric Variable: Boxplot in `ggplot2`



```
library(ggplot2)
ggplot(example.dat, aes(x = x)) +
    geom_boxplot() +
    theme_minimal()
```

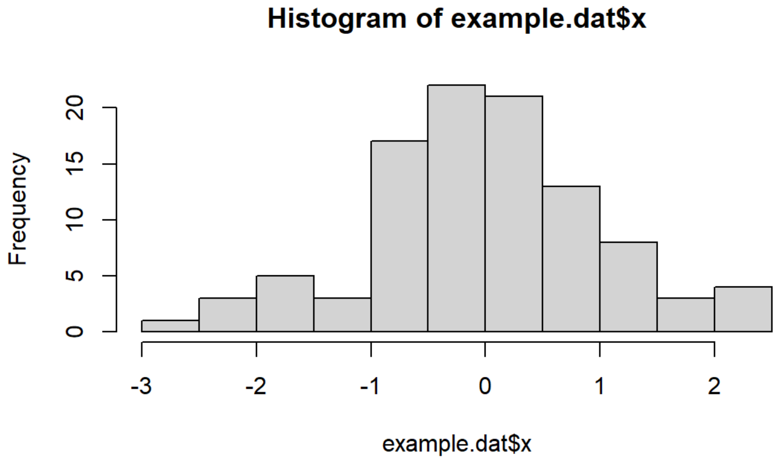
mapping using aes

gives a basic background.



Single Numeric Variable: Histogram in base R

```r
hist(example.dat$x)
```

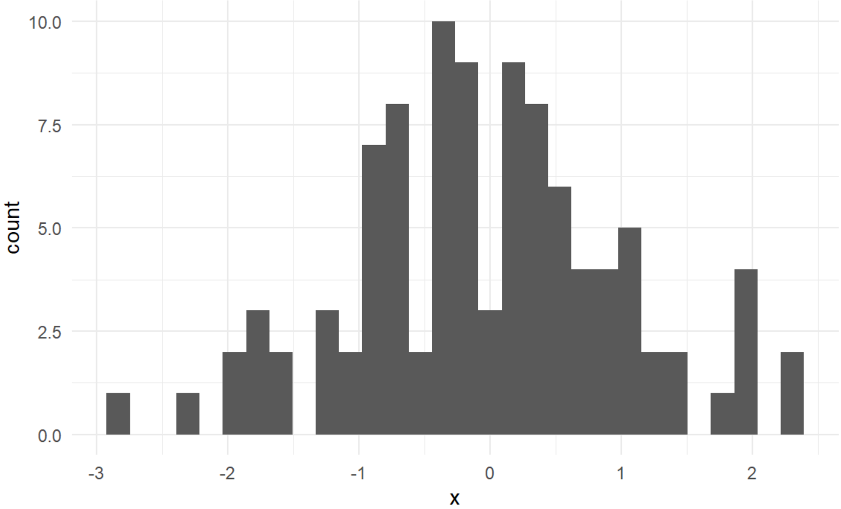


Same as previous
but we use ggplot2

Single Numeric Variable: Histogram in ggplot2

```
ggplot(example.dat, aes(x = x)) +
    geom_histogram() +
    theme_minimal()
```

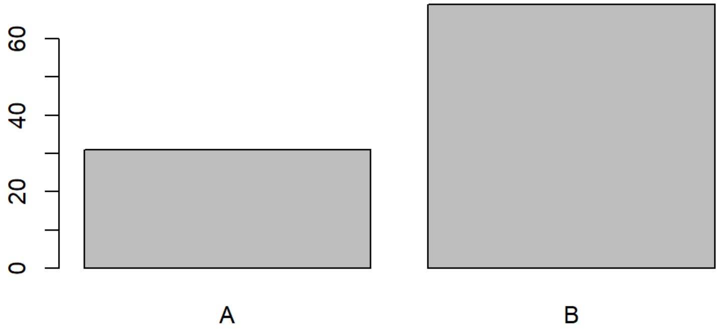
you can specify the
number of bins, which
controls the bandwidth
of the histogram



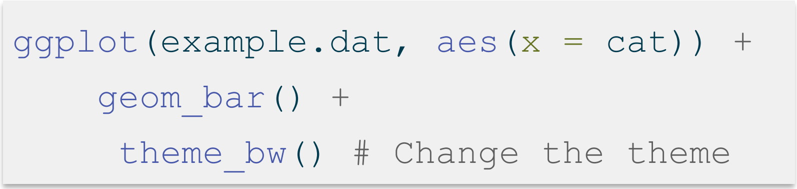
Single Categorical Variable: Bar Plot in base R

```
barplot(table(example.dat$cat))
```

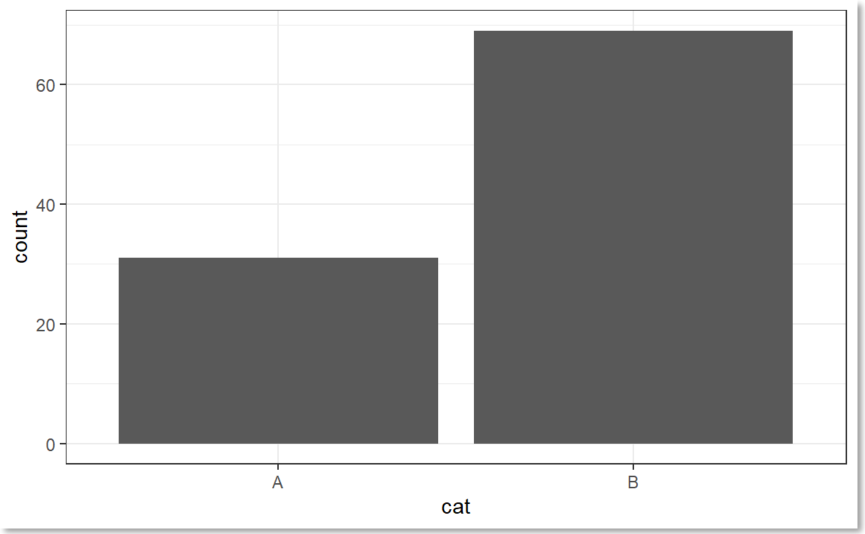
this command creates the number of counts for each variable.



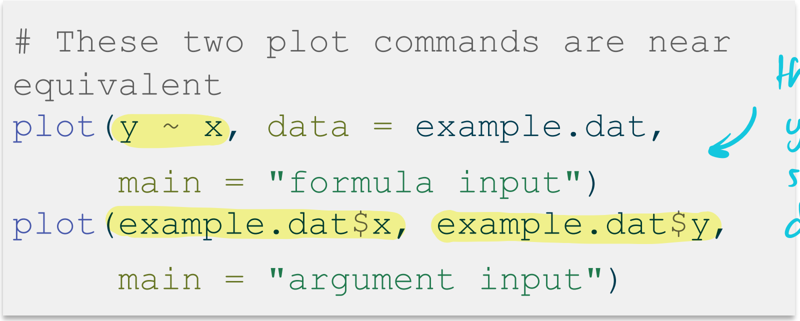
Single Categorical Variable: Bar Plot `ggplot2`



```
ggplot(example.dat, aes(x = cat)) +
    geom_bar() +
     theme_bw() # Change the theme
```

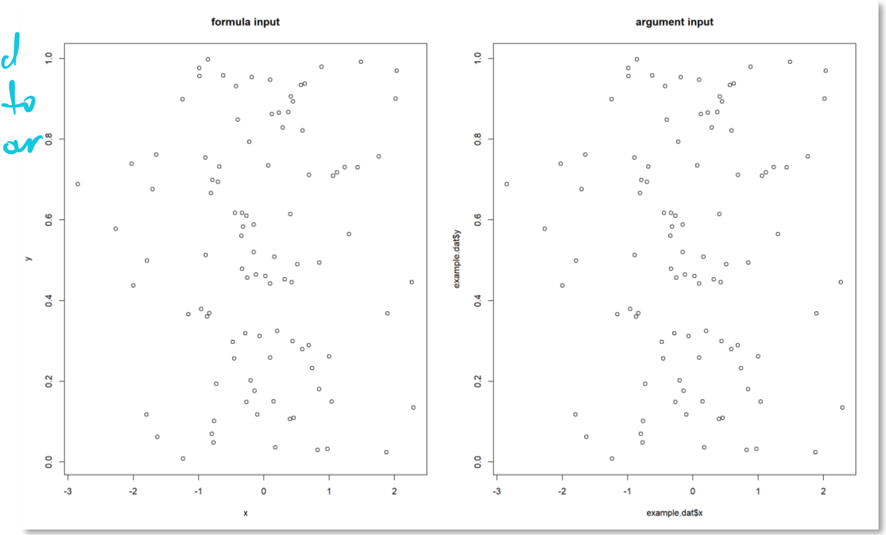


Two Numeric Variables: Scatterplot in base R



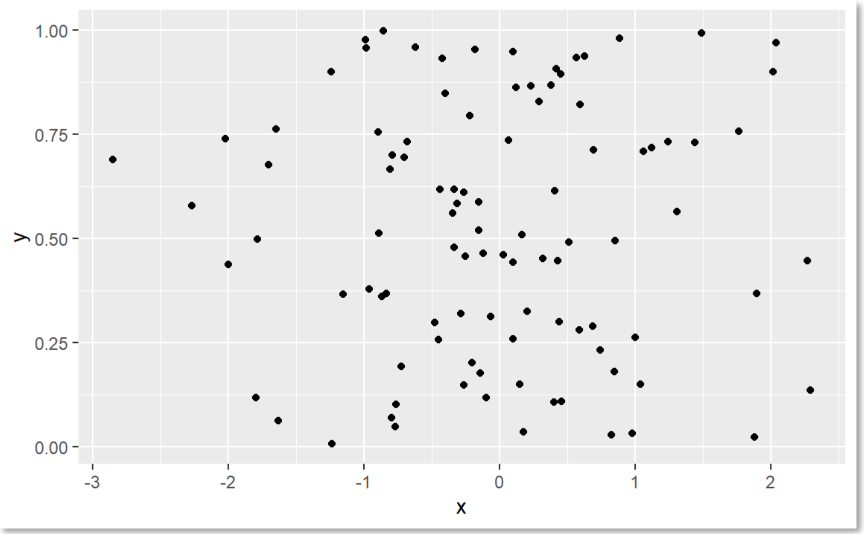
```
# These two plot commands are near
equivalent
plot(y ~ x, data = example.dat,
     main = "formula input")
plot(example.dat$x, example.dat$y,
     main = "argument input")
```

this method you need to specify your dataset



Two Numeric Variables: Scatterplot in `ggplot2`

```
ggplot(example.dat, aes(x = x, y =
y)) + geom_point() # default theme
here
```



THE UNIVERSITY OF
SYDNEY

Coding with R

What is `R`?

need to use laptop...

- Free, open source software designed for statistical computing ✓✓
- Runs on Windows, Mac, Linux and other flavours of Unix ✓✓
- Provides an interactive environment, but it is also an interpreted programming language ✓✓
- Its power lies in the thousands of contributed packages on CRAN, Bioconductor and github ✓✓

↓

Central Run Archived Network.

Base `R` and the tidyverse

- The `tidyverse` popularised by Hadley Wickham and the team at
  - Has a (somewhat) standardised syntax (pipes `|>` or `%>%` are king except for `+` in `ggplot2`)
  - Produces more human readable code ✓✓
  - Not as stable as base, breaking changes occur as `tidyverse` develops.
  - Good for interactive data analyst ✓✓

Base `R` and the tidyverse, (cont.)

- Core base `R`
  - Good for production level code
  - Stable
  - Function syntax inconsistent

Quirks of R Syntax

- `<-` is the symbol for 'assign'
  - Example: `x <- 14`
  - which is equivalent to: `x = 14` when used at the prompt
- Should use `=` for argument matching in a function
- The period symbol `.` can be used in variable names
  - Example: `new.vector <- c("A", "B", "C")`
- Element indexing starts at 1

```
new.vector <- c("A", "B", "C")
new.vector[0]     0     1     2
```

```
 character(0)
```

```
 new.vector[1]
 [1] "A"
```

can use identical(x,y)
to determine if
variables have same assigned
value or not. Returns true or
false.

Basic Data Types in R

Classical data types

- Numeric ✓✓
- Integer ✓✓
- Logical ✓✓
- Character ✓✓
- Complex ✓✓

- Factor: categorical data type
  - Unique to R (integer with some attributes)

```
data("ToothGrowth")
levels(ToothGrowth$supp)
[1] "OJ" "VC"
class(ToothGrowth$supp)
[1] "factor"
str(ToothGrowth$supp)
 Factor w/ 2 levels
"OJ","VC": 2 2 2 2 2 2 2 2
2 2 ...
```

Homogeneous vs. Non-homogenous Data Types in R

Homogenous

- Vector
  - Sequence of data elements of the same basic data type
- Matrix
  - Collection of data elements in a 2-dimensional array with rows and columns

Non-homogenous

- List
  - More general structure containing other objects (including possibly other lists)
- Data frame
  - Used for storing data, each column can be a different basic type
  - All columns must have the same length

Vectors

```r
new.vector <- c(1, 2, 3)
class(new.vector)
```

```
[1] "numeric"
```

```r
length(new.vector)
```

```
[1] 3
```

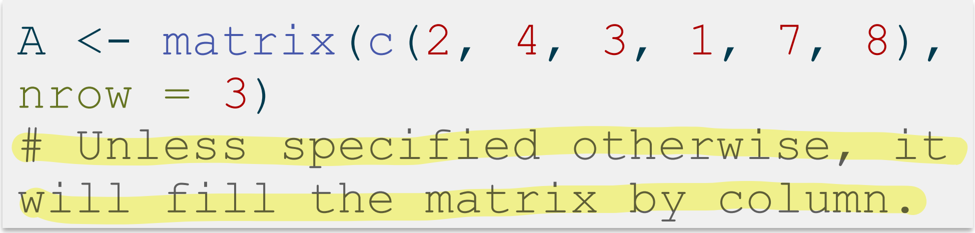
```r
new.vector[1:2]
```

```
[1] 1 2
```

```r
new.vector <- c(1, 2, "hello")
class(new.vector)
```

```
[1] "character"
```

Matrix



```
A <- matrix(c(2, 4, 3, 1, 7, 8),
nrow = 3)
# Unless specified otherwise, it
will fill the matrix by column.
```

```
A
     [,1] [,2]
[1,]    2    1
[2,]    4    7
[3,]    3    8
A[2, 1]
[1] 4
A[1, ]
[1] 2 1
A[5]
[1] 7
```

List

```
vector.a <- c(1, 2, 3)
vector.b <- c("hello", "world", "!!")
new.list <- list(c(vector.a,
vector.b))
new.list
```

```
[[1]]
[1] "1"      "2"      "3"      "hello"
"world" "!!"
```

```
new.list <- list(vector.a, vector.b)
new.list
```

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


[[2]]
[1] "hello" "world" "!!"
```

```
new.list[[1]]
```

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

Data Frames

we can check if a variable is a list using: is.list(warpbreaks) — in this case.

```
head(warpbreaks)
  breaks wool tension
1     26    A       L
2     30    A       L
3     54    A       L
4     25    A       L
5     70    A       L
6     52    A       L
class(warpbreaks)
[1] "data.frame"
head(warpbreaks$wool)
[1] A A A A A A
Levels: A B
```

should be true, since a dataframe is a special kind of list.

```
str(warpbreaks)
'data.frame':   54 obs. of  3
variables:
 $ breaks : num  26 30 54 25 70
52 51 26 67 18 ...
 $ wool   : Factor w/ 2 levels
"A","B": 1 1 1 1 1 1 1 1 1 1
...
 $ tension: Factor w/ 3 levels
"L","M","H": 1 1 1 1 1 1 1 1 1
2 ...
names(warpbreaks)
[1] "breaks"  "wool"
"tension"
```

THE UNIVERSITY OF
SYDNEY