STAT5003

Week 1: Basics of statistical computing and visualization

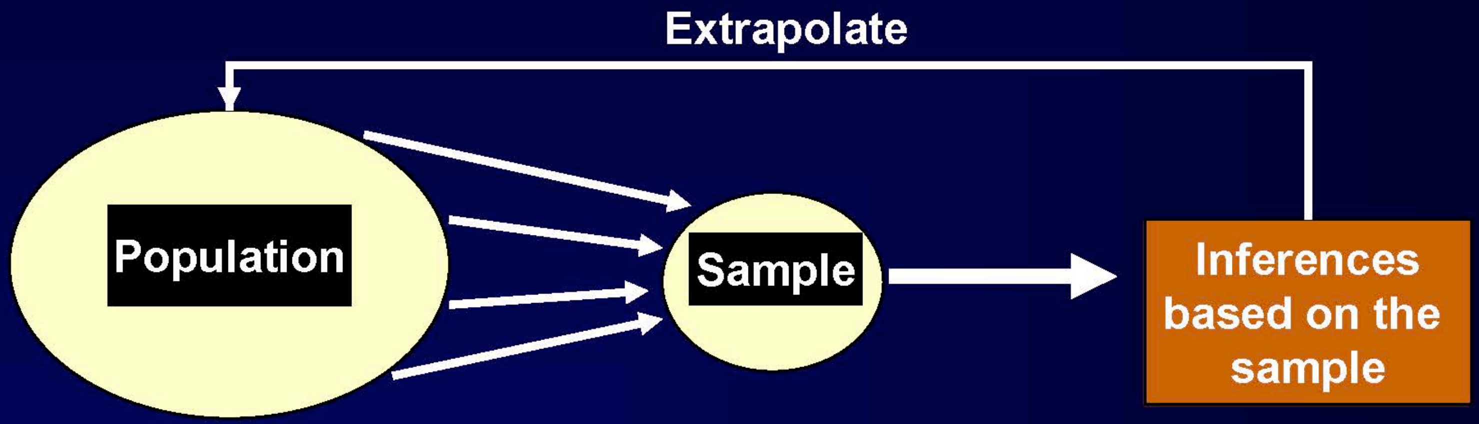
Jaslene Lin

Basics of Statistical Computing

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

Review of basic statistical concepts

Population and Sample



e.g. all adults in a population of interest

e.g. 300 adults chosen at random

e.g. at least one-third of adults have high cholesterol

Population

- **Definition**:
  - The set of data (numeric or otherwise) corresponding to the entire collection of units about which information is sought.
- **Examples**:
  - Cholesterol levels– Cholesterol levels of ALL adults in Australia.
  - The number of languages spoken from ALL currently enrolled students in University of Sydney.

Sample

- **Definition:**
  - ➠ A subset of the population data that are actually collected in the course of a study.
- **Examples**:
  - ➠ Cholesterol levels of 300 adults chosen at random 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 Statistics

- 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.

Quiz

You are conducting a survey to understand **the study habits your peers in STAT5003**. The survey includes questions about the time spent studying, preferred study methods, and familiarity with basic statistical concepts.

In the context of your survey, what is the definition of a population?

○ A subset of students who responded to the survey

○ All students in STAT5003

○ A numerical summary of the survey responses

○ A graphical representation of the survey data

Quiz

In your survey, what is a sample?

- ○ All students in STAT5003
- ○ A subset of students who responded to the survey
- ○ A numerical summary of the survey responses
- ○ A graphical representation of the survey data

Quiz

In your survey, the number of hours students study per week is **likely** an example of what type of data?

- ○ Continuous
- ○ Discrete
- ○ Categorical
- ○ Ordinal

Descriptive statistics – numeric and graphics

Many methods are available for summarising data in both **numeric** and **graphical** form

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

Basic statistical graphics

Types of graphics covered in this course

- Become familiar with simple graphics using base**R** and ggplot graphics
- base**R** use the built in plotting functions
  - Typically good for quick plots of simple datasets
- ggplot graphics
  - Name meaning the grammar of graphics
  - Typically better for more complicated datasets

Simple example dataframe for plots

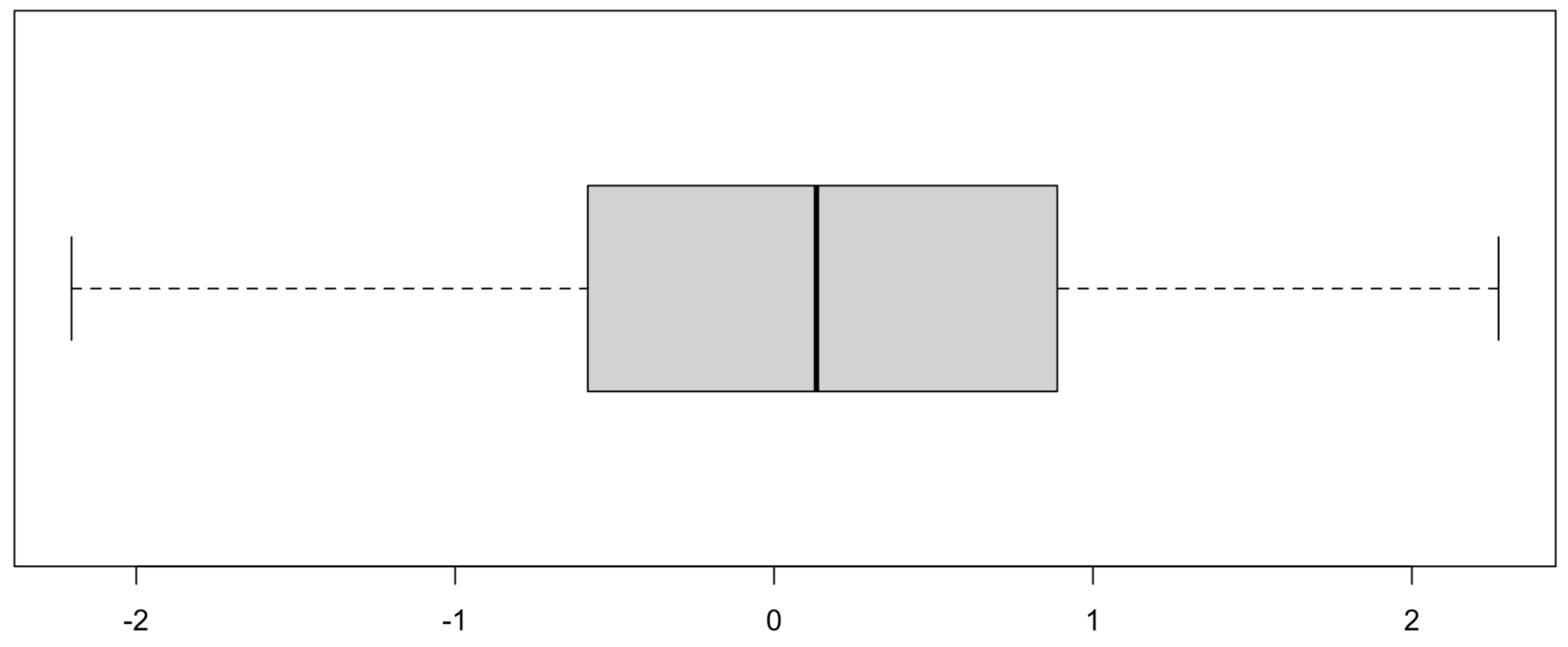
```r
set.seed(5003)

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 -0.08569102 0.6601036   B
2  0.12462451 0.4534993   B
3  0.87579853 0.4029725   B
4  0.77128325 0.9148725   B
5 -0.29414040 0.1930455   A
6 -0.64272686 0.5727217   B
```

```r
library(tidyverse)
```

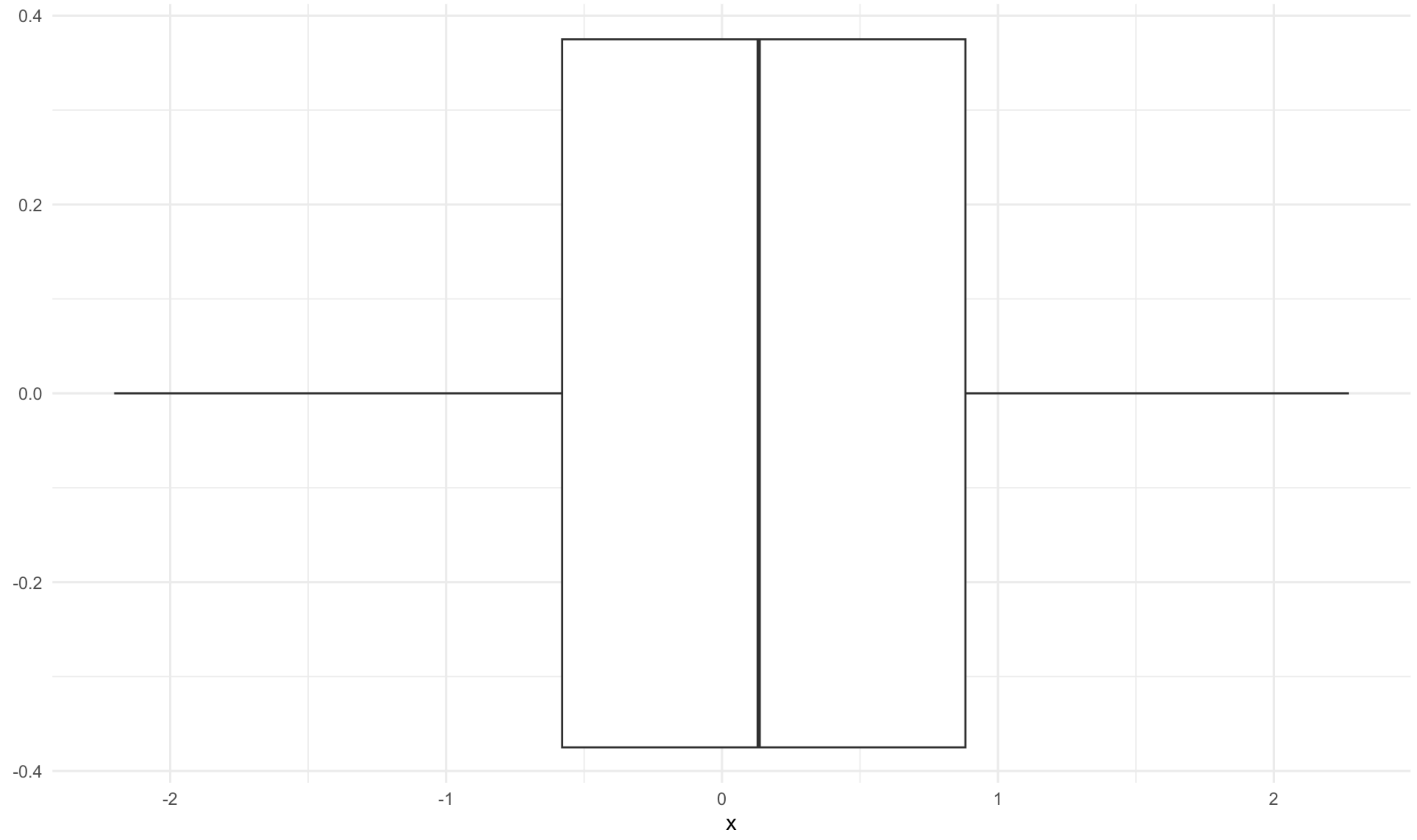
Single numeric variable: Boxplot in base**R**

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



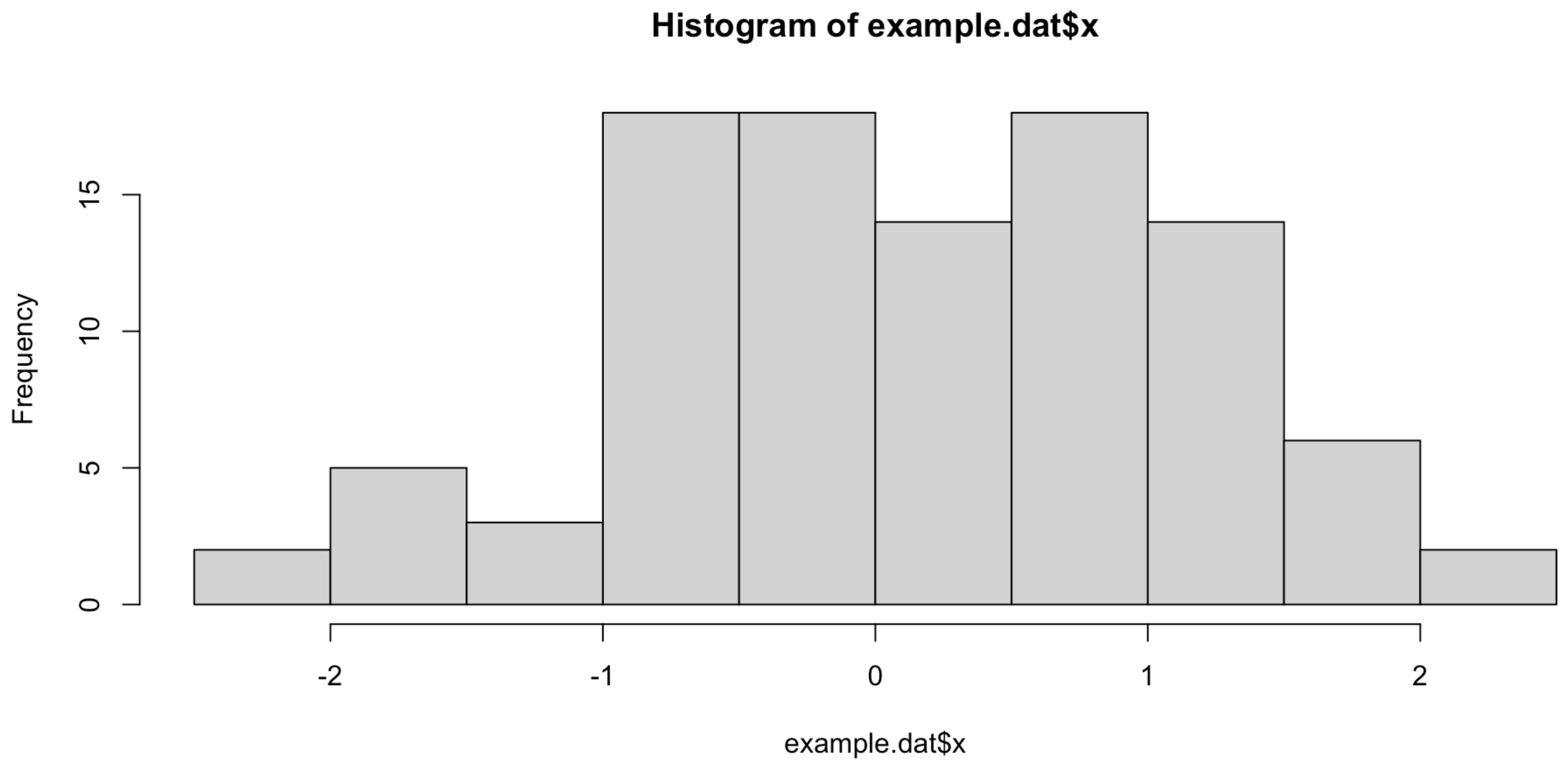
Single numeric variable: Boxplot in ggplot

```r
library(ggplot2) # Only need to load the library once in an R session
ggplot(example.dat, aes(x = x)) + geom_boxplot() + theme_minimal()
```



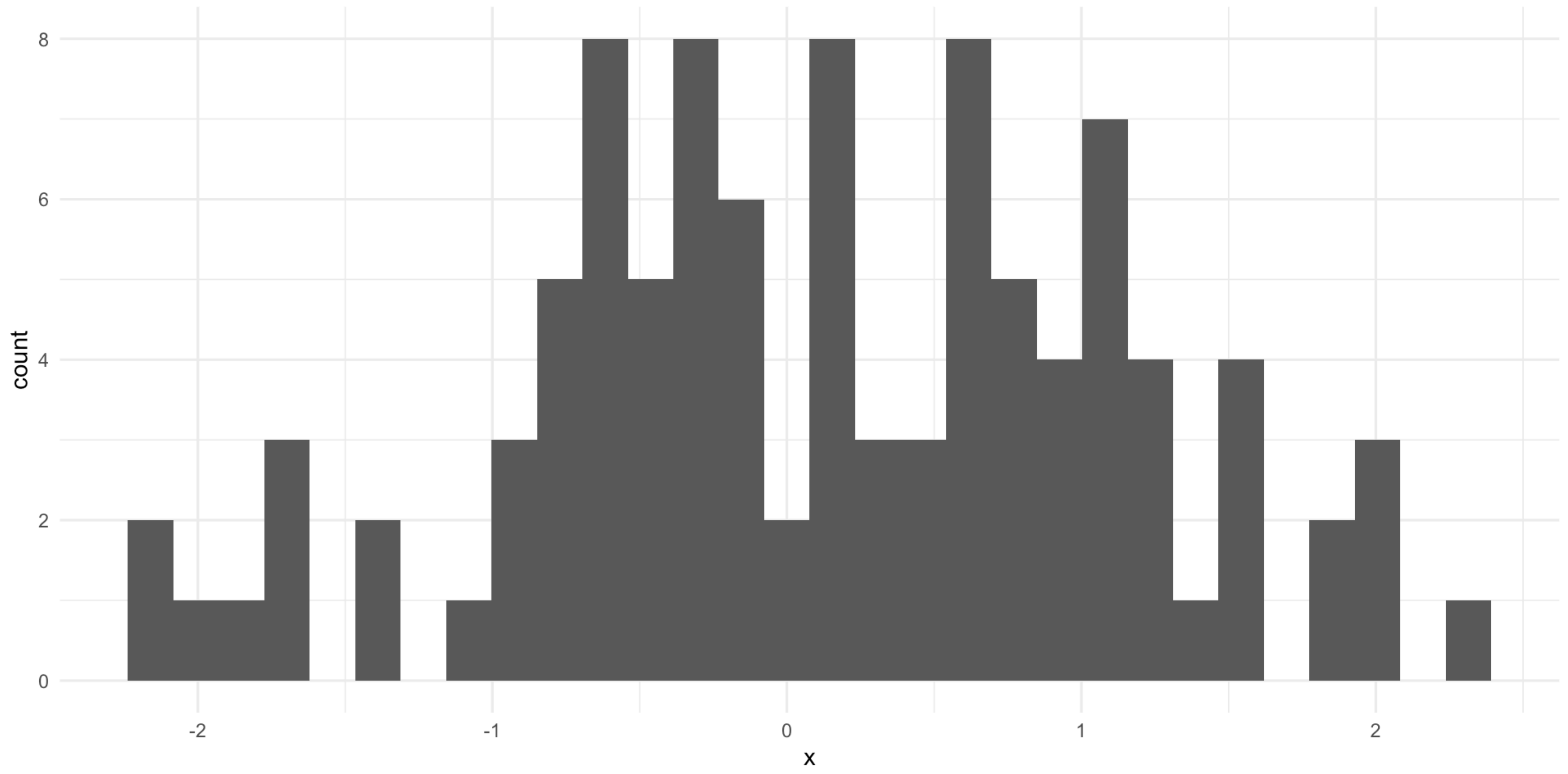
Single numeric variable: Histogram in base**R**

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



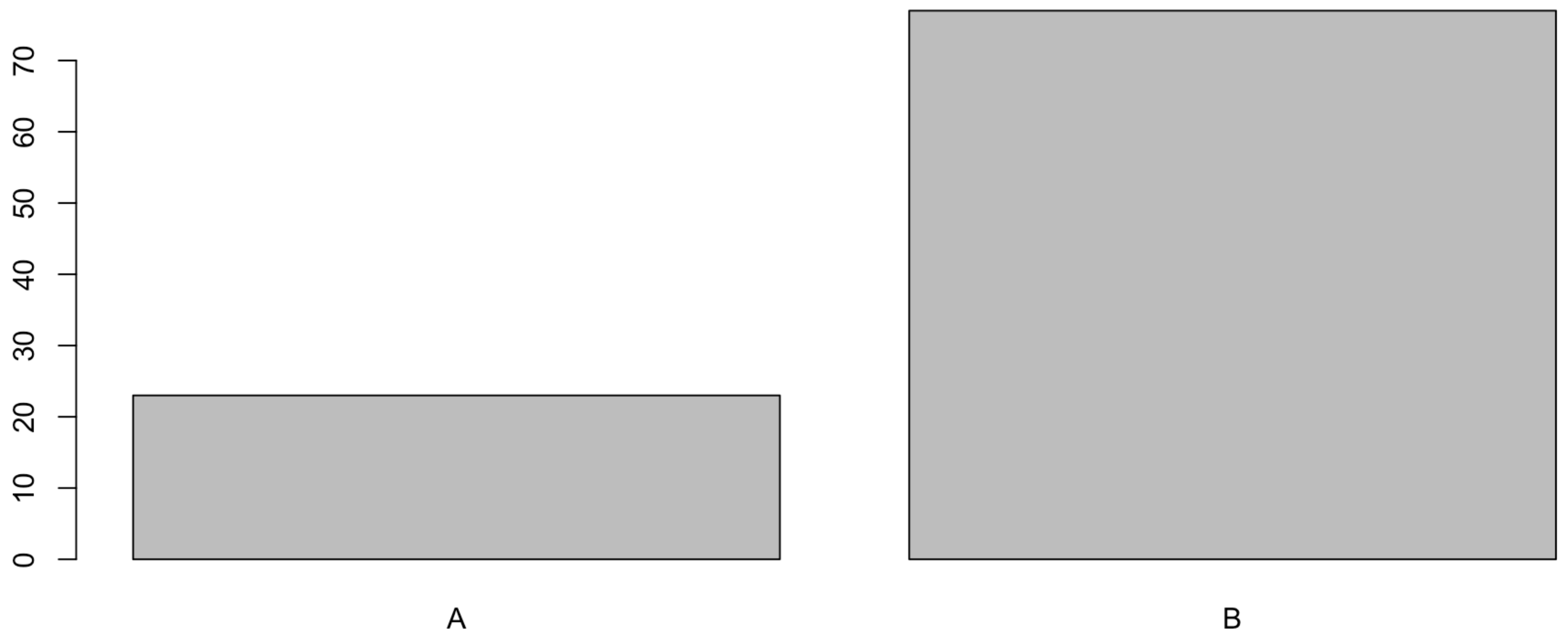
Single numeric variable: Histogram in ggplot

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



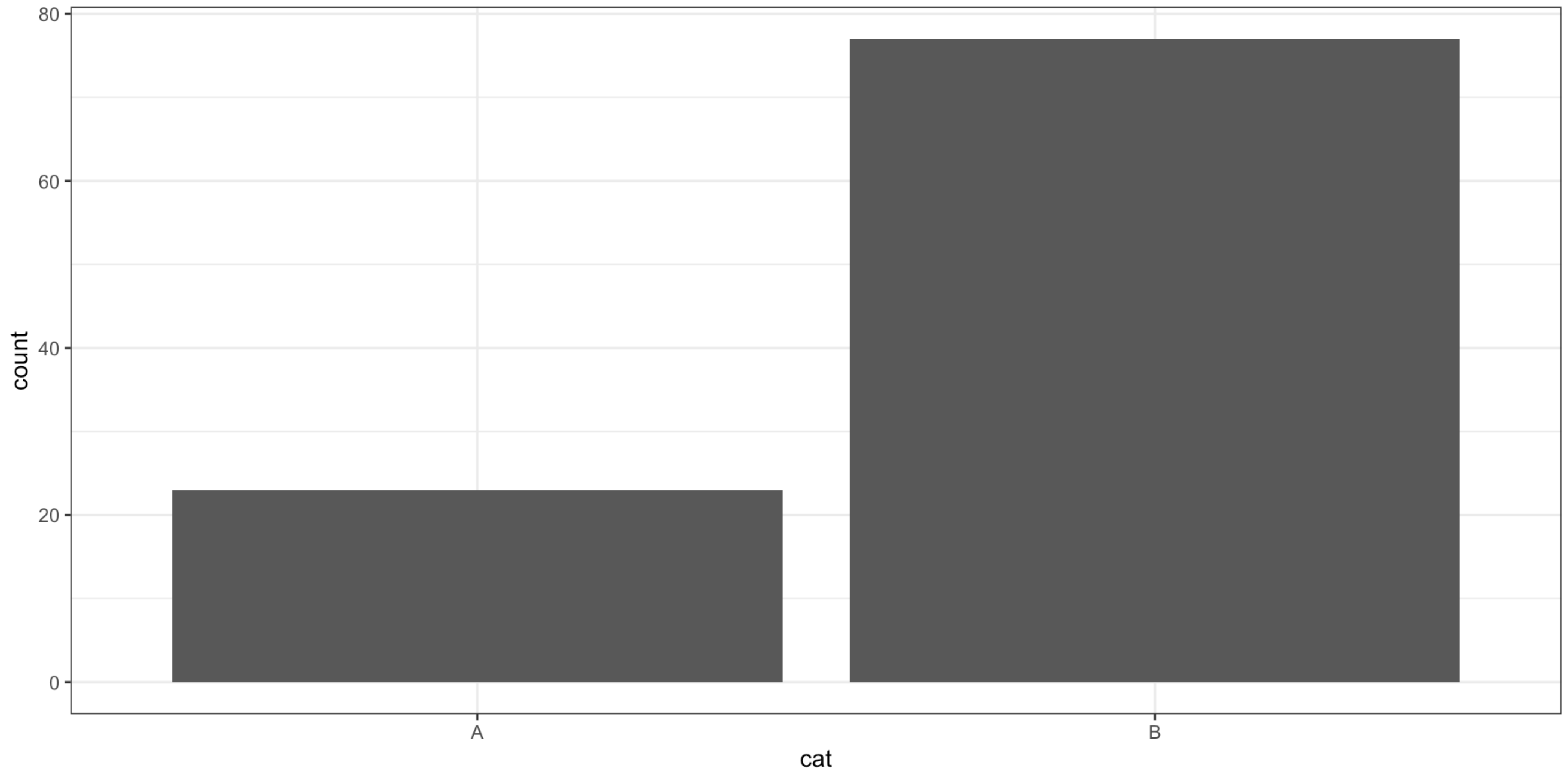
Single categorical variable: Bar plot in base **R**

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



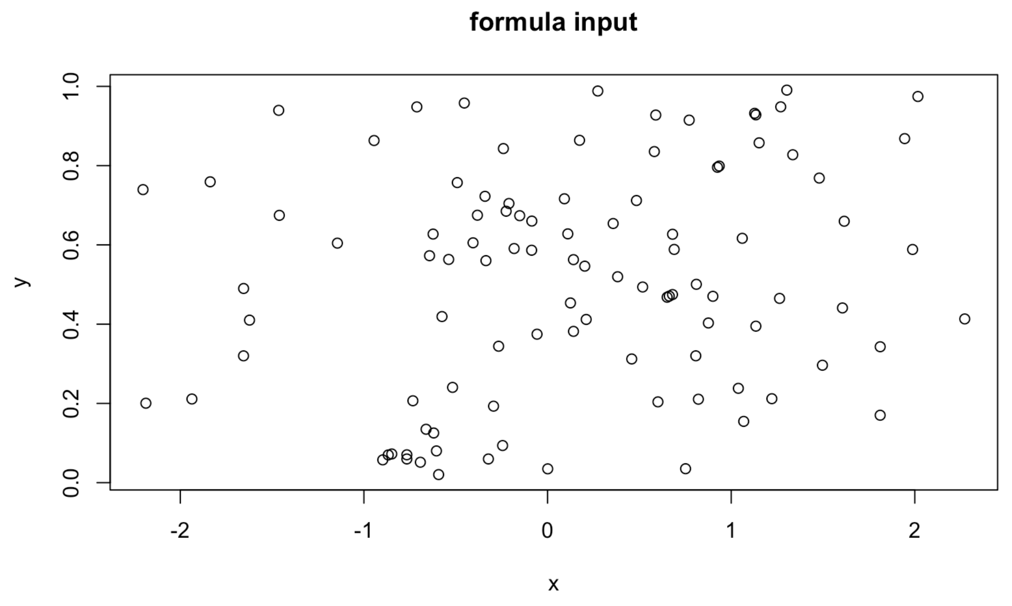
Single categorical variable: Bar plot in ggplot

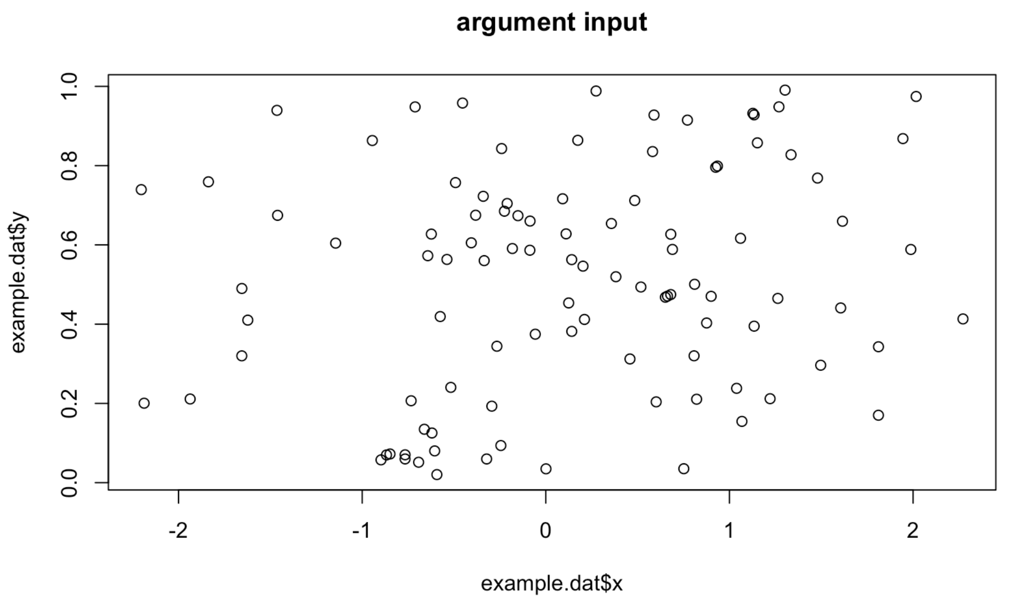
```r
ggplot(example.dat, aes(x = cat)) + geom_bar() + theme_bw() # Change the theme
```



Two numeric variables: Scatterplot in base **R**

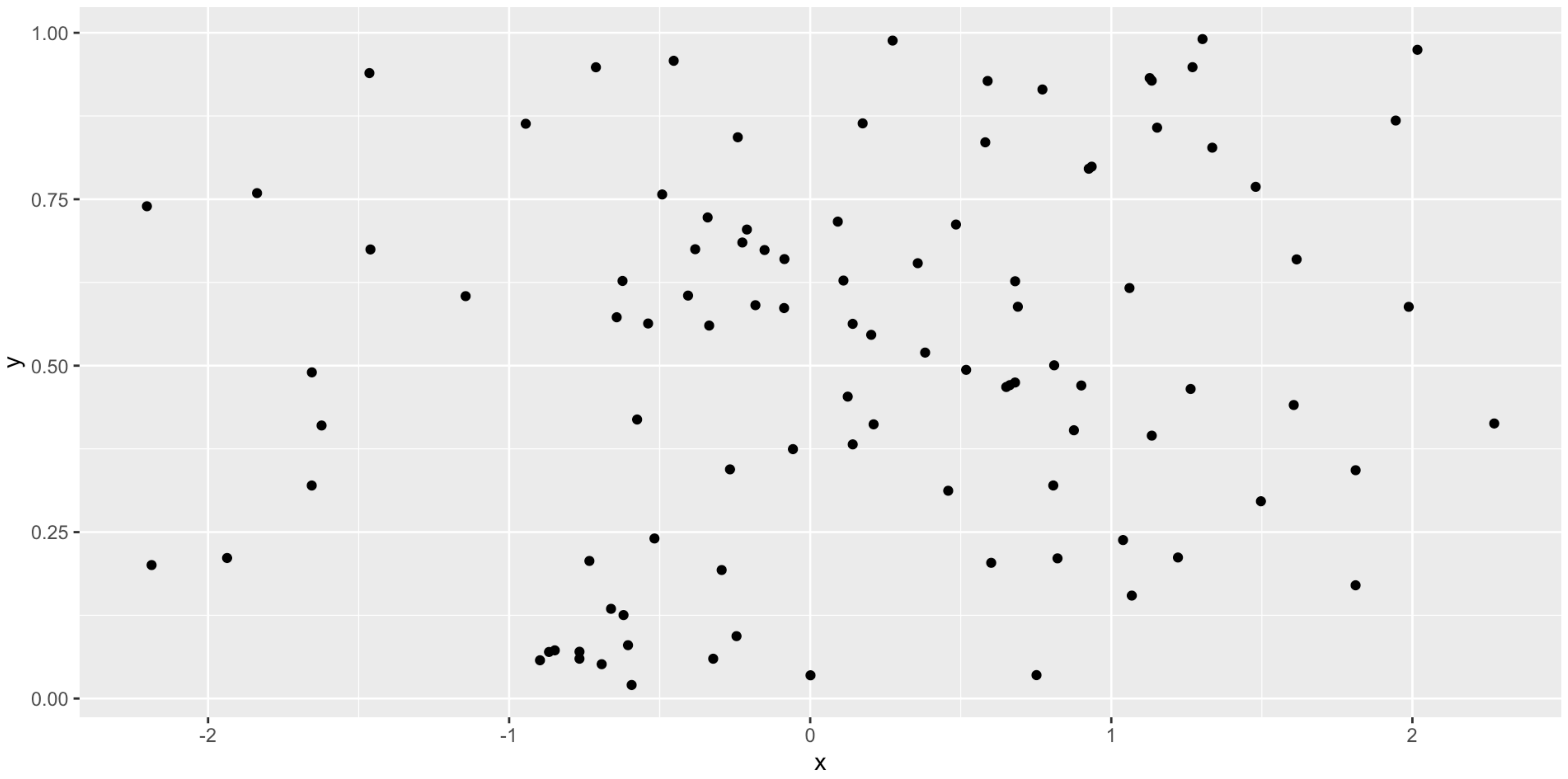
```r
# These two plot commands are near equivalent
plot(y ~ x, data = example.dat, main = "formula input")
plot(example.dat$x, example.dat$y, main = "argument input")
```





Two numeric variables: Scatterplot in ggplot

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



Quiz

What is the purpose of a histogram in the context of your survey data?

○ To show the relationship between two variables

○ To display the frequency distribution of study hours

○ To summarize the central tendency of study hours

○ To compare the means of different study methods

What is R?

- 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

What is RStudio?

- RStudio is an integrated development environment (IDE) for **R**
- Think of it as the front-end, like a powerful text editor
- R needs to be installed first before RStudio

R: Engine	RStudio: Dashboard

Base R versus Tidyverse

- Core Base **R**
  - ➟ Syntax and Functions: Good for production level code, stable but function syntax inconsistent.
  - ➟ Data Manipulation: using functions like `apply()`, `lapply()`, `sapply()`, and `tapply(),` which can be powerful but sometimes complex and less readable.
  - ➟ Data Visualization: Base R provides basic plotting functions such as `plot()`, `hist()`, and `boxplot()`.
- **Tidyverse**
  - ➟ Has a (somewhat) standardised syntax. Pipes (base `▷` and the magrittr `%>%`) 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

R Session

- **Working Directory**: Each session runs from a working directory
  - `getwd()` shows the current working directory for R
  - `setwd(<path>)` to change the working directory where `<path>` is a string
    - Example: `setwd("C:/Users/usyd-student")` for a windows user
- **Workspace**: Includes
  - Global environment: Data and variables loaded or defined
  - package environments: Any loaded packages with their functions/data
- **History**:
  - Can view your recent commands in the History pane
  - Can navigate previous commands using up and down arrows at your prompt

Packages

- Inspect the current environment
  - `sessionInfo()`: shows everything in the current R session
- To install a new package
  - `install.packages("cluster")` will install the `cluster` package from the Comprehensive R Archive Network (CRAN) repository
- To load a package to use in the environment `library(cluster)`
  - After loading a package, you will be able to use the functions provided in that package
- If name conflicts arise you can specify the desired function using `::`
  - Example: `dplyr::filter` will use the `filter` function from the `dplyr` package instead of the `stats::filter` function

Help

- CRAN requires every exported function in every contributed package to have a help file
    - `help.start()`
    - `help(plot)`
    - `? plot`
- In general, the help file for each function gives a brief description of what the function does, the required inputs, the expected outputs and some examples
    - Some packages also include a "vignette", a short document with guidance on using the package

Quirks of R syntax

- `←` is the symbol for 'assign', e.g. `x ← 14`
  - ➟ 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

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

```
character(0)
```

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

```
[1] "A"
```

Basic data types in R

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

```r
data("ToothGrowth")
levels(ToothGrowth$supp)
```

```
[1] "OJ" "VC"
```

```r
class(ToothGrowth$supp)
```

```
[1] "factor"
```

```r
str(ToothGrowth$supp)
```

```
 Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
```

Classical data types

- Numeric
- Integer
- Logical
- Character
- Complex

Homogeneous vs non-homogenous data types in R

Homogeneous

- 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-homogeneous

- 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"
```

```r
is.vector(new.vector)
```

```
[1] TRUE
```

Matrix

```r
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
```

```r
A[2, 1]
```

```
[1] 4
```

```r
A[1, ]
```

```
[1] 2 1
```

```r
A[5]
```

```
[1] 7
```

List

```r
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" "!!"
```

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

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

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

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

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

Data frames

```r
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
```

```r
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 ...
```

```r
names(warpbreaks)
```

```
[1] "breaks"  "wool"    "tension"
```

```r
class(warpbreaks)
```

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

```r
head(warpbreaks$wool)
```

```
[1] A A A A A A
Levels: A B
```

Rprojects

- Each Rstudio project has its own working R session, workspace, history, and source documents
- You can either create an Rproject in a new directory or within and existing directory
- When you open an .Rproj file, the following happens:
    - ➠ A new R session (process) is started
    - ➠ If it exists, the .Rprofile in the project's main directory is loaded
    - ➠ If applicable, the .RData file in the main directory is loaded
    - ➠ If set, the .Rhistory file in the main directory is loaded
    - ➠ The current working directory is set to the project directory
    - ➠ If set, previously edited documents appear in the source editor