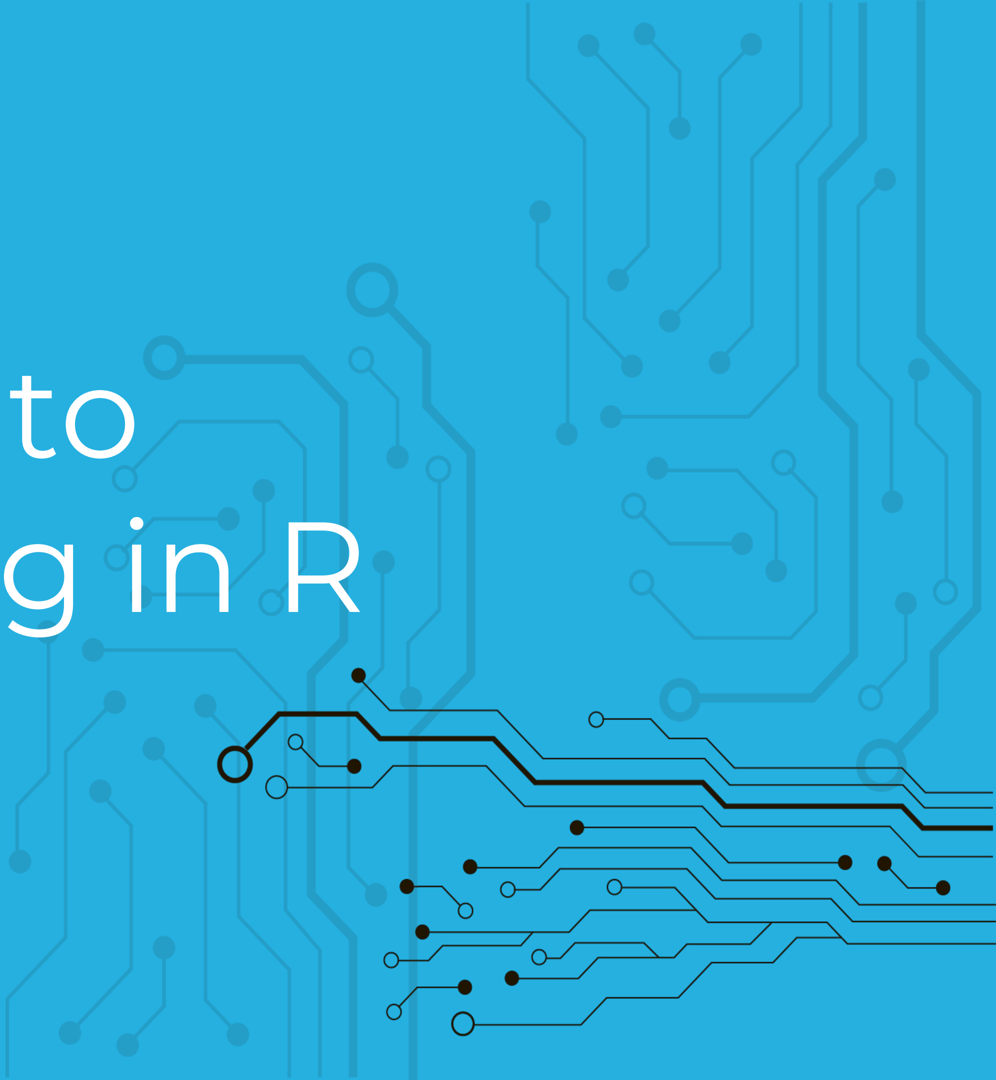
Introduction to Programming in R



Topics for today

- Basics of coding
- Program structure
- Control flow and functions
- Visualization and reporting
- Git
- Resources

```
CPU:  Two Z80 microprocessors
Clock:  2.048MHz
Memory:  16K bytes RAM expandable to 48K
Display:  25 lines by 80 characters
Characters:  5 x 7 dot matrix, upper & lower case
Graphics:  33 special graphics characters

Keyboard:  72 keys, 60 alphanumeric & 12 function
           control (8 user-definable)
Numeric keypad:  12 keys
Cursor Addressing:  relative & direct for full cursor control
Edit Functions: Insert or delete character or line
Video:  Normal and reverse by character for highlighting _
```

Let’s get set up...

Connect to GitHub with SSH

- Setting up SSH means that you won’t need to enter an email address and password every time you want to push your code to GitHub
- You only need to perform this step one time on each machine you will be working from

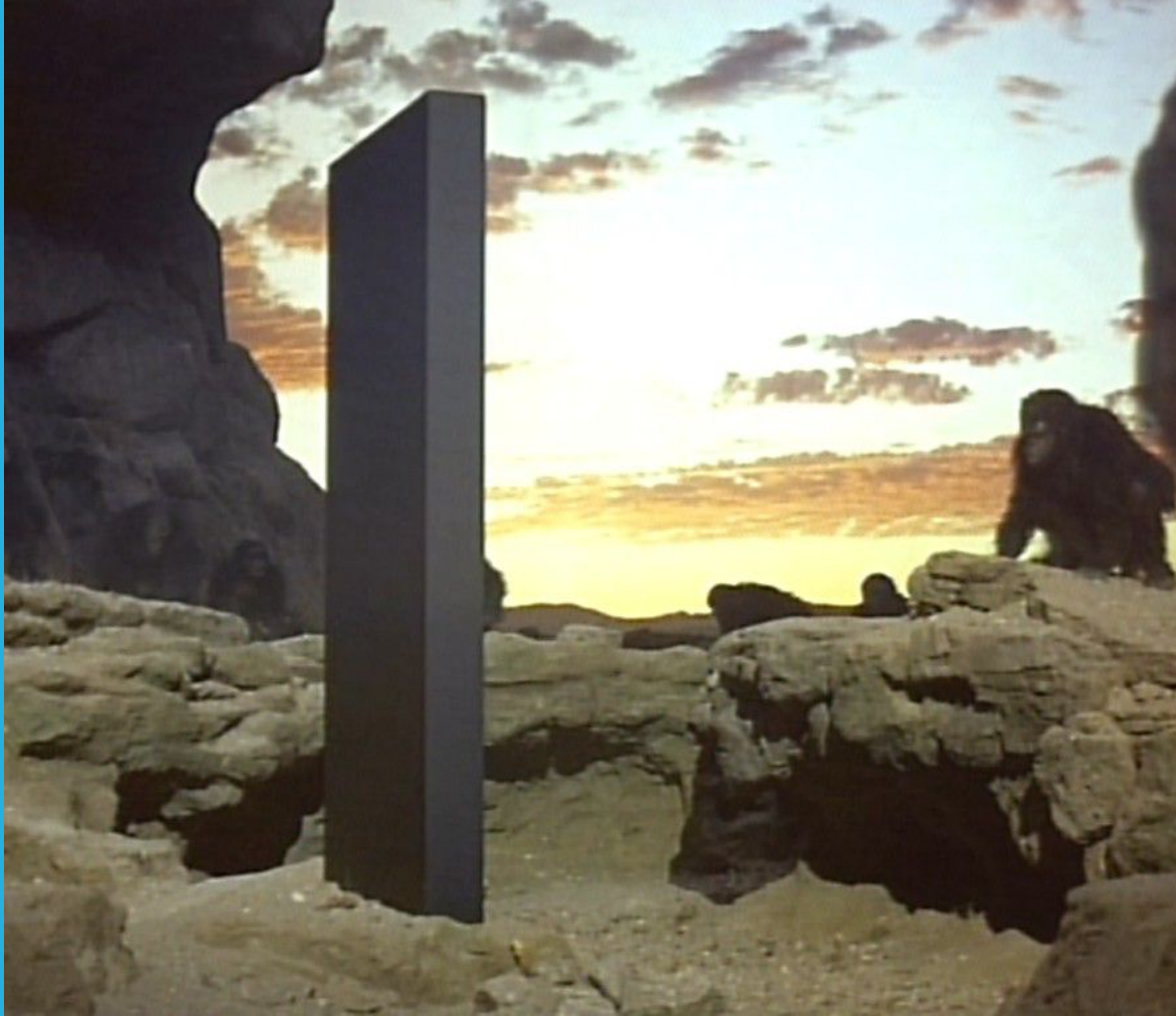
Fork the workshop repository

- Forking a repository creates a personal copy in your own account
- This allows you to make changes that do not affect the original repository, while still controlling versions and tracking your work

The Basics of Coding

What is **programming**?

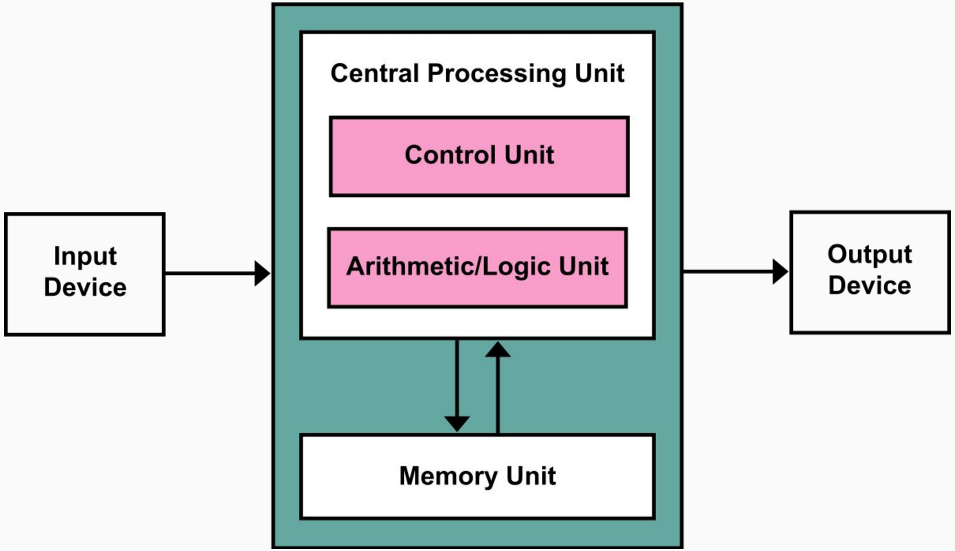
What is a programming **language**?



How does a computer process your code?

- Programming languages provide a structured, human-readable syntax for communicating with computers

- Code written by humans is compiled to machine-readable code, which serves as a set of instructions for a computer to execute



What defines a programming language?

- Programming languages are characterized and distinguished by a number of key properties including:
  - **Syntax**
  - Programming paradigm
  - Typing discipline
  - Implementation

What defines a programming language?

- Programming languages are characterized and distinguished by a number of key properties including:
  - Syntax
  - **Programming paradigm**
  - Typing discipline
  - Implementation

Paradigms

Imperative

- Programs consist of a series of sequential statements
- These statements serve as an instruction set to move from some initial state to another final state
    - Example:
        - `a = 1;`
        - `b = 2;`
        - `c = a + b;`
        - `print c;`

Declarative

- Programs are comprised of a set of things you wish to accomplish
- This set does not include instructions for how the computer should execute those tasks
    - Example:
        - `INSERT INTO Table_1 (col_a,col_b,col_c,col_d) VALUES (1,2,3,4);`

Paradigms continued

Procedural

- Subtype of imperative programming
- Adds concept of subroutines
- Subroutines contain their own sequential set of operations for the computer to perform
- These subroutines are called by a main routine which is executed in order
- Enables code reuse

Functional

- Programs consist of a set of functions (think math)
- Take some input, perform a set of operations, and return a new value
- No side effects
- The same output is produced for any given input every time a specific function is executed
- Functions are composed in an arrangement which leads to desired output

Paradigms continued

Object-oriented

- Programs consist of objects which may exhibit both state and behavior
- These objects are typically defined as **classes**
- Classes contain:
  - Attributes
    - Similar to variables
  - Methods
    - Similar to functions
- Program flow is manipulated through the instantiation of classes followed by manipulation of their attributes and execution of their methods

Key Differences

- Modularity
- Scope
- Execution order

Overlap…

- Most modern languages employ multiple paradigms
- Usage patterns often matter more than language features

What defines a programming language?

- Programming languages are characterized and distinguished by a number of key properties including:
    - Syntax
    - Programming paradigm
    - **Typing discipline**
    - Implementation

Typing discipline

Dynamic	Static
Weak	Strong
Implicit	Explicit

What defines a programming language?

- Programming languages are characterized and distinguished by a number of key properties including:
  - Syntax
  - Programming paradigm
  - Typing discipline
  - **Implementation**

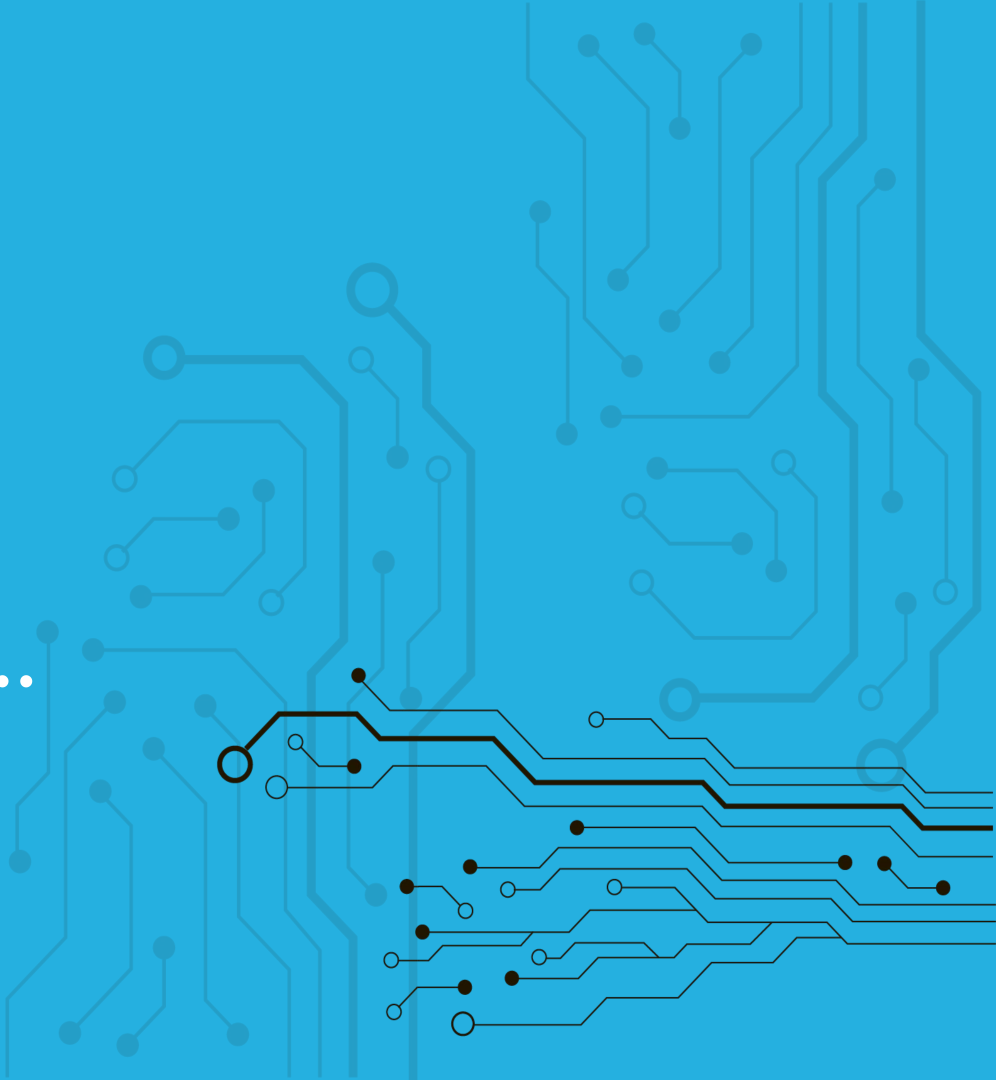
Implementation

- Implementation deals with **how** and **when** code is executed
- All programming languages must ultimately be translated to language that a computer can understand (typically machine code)
- Two main strategies:
  - **Interpretation** -- programs are read by an interpreter (another program), which in turn executes the programs instruction set on a machine
  - **Compilation** -- programs are converted to machine-readable code. That code can then be executed on a computer.
- Many languages have multiple implementations which are optimized for specific applications

Disclaimer

These descriptions are simplified generalizations. Programming paradigms, typing disciplines, and implementations are fluid concepts, and may differ between sources.

Let’s look at R…



The basics in terms of R

- Typing discipline
  - Variable type constraints
  - Dynamic type checking
  - Implicit variable declaration
- Programming paradigm:
  - Multi-paradigm
    - Imperative / Procedural
    - Functional / Declarative
    - Object-oriented

- Interpreted
  - Portable
  - Rapid development cycle
  - “Slow” execution
- Libraries for almost everything

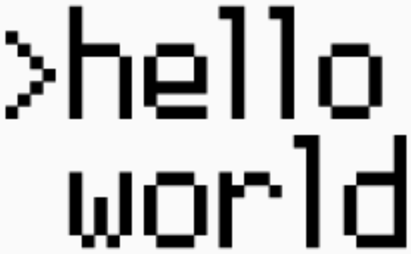
RStudio

...Just use it

Basic Program Structure

Statements and expressions

- A statement is a segment of syntactically correct code which performs some action. Statements do not necessarily have an associated value
- An expression is a segment of code which, when evaluated, produces some value

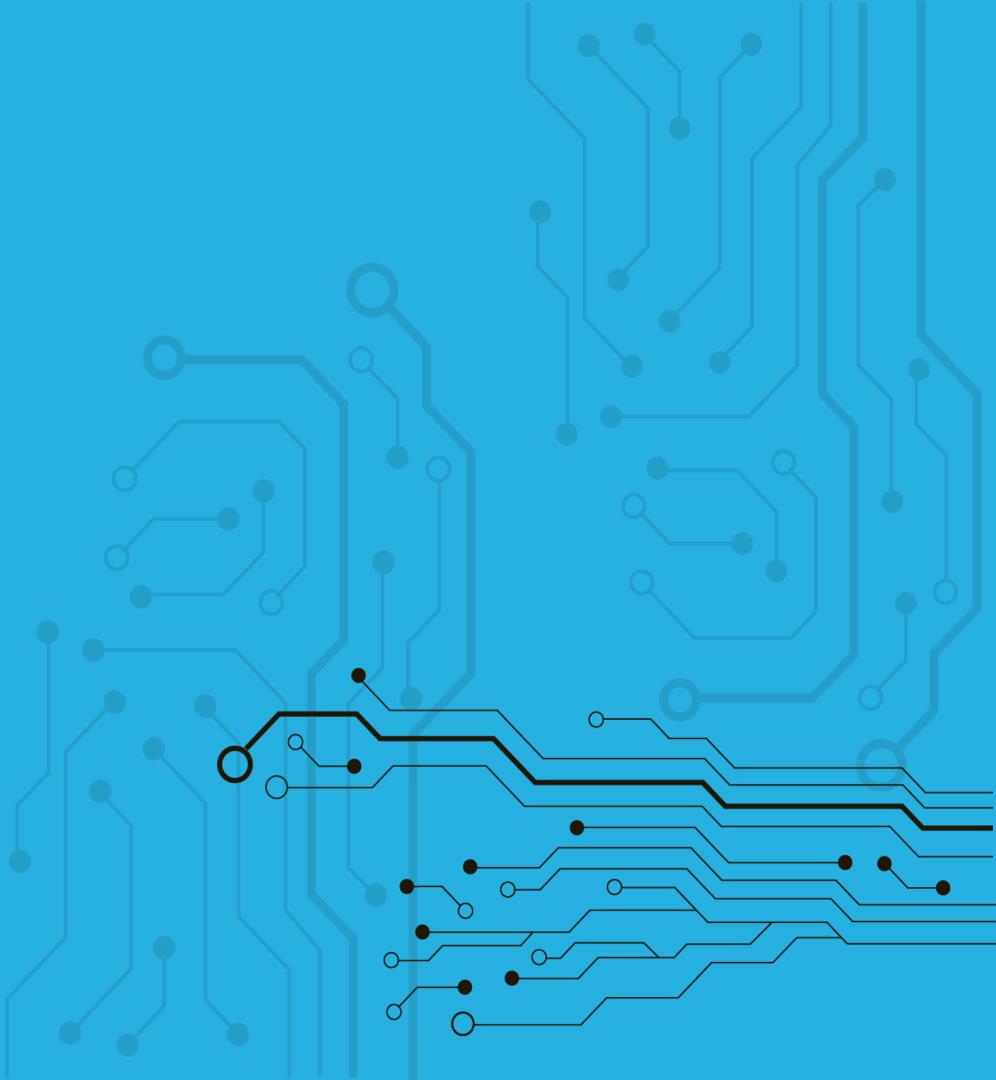


Operators

- Symbols associated with a particular operation
- Recognized by the interpreter / compiler
- Instruct the computer to perform an arithmetic, logical, or relational task
- When used in context with data, operators result in a value

+%()-</=>+%()-</=>+%()-

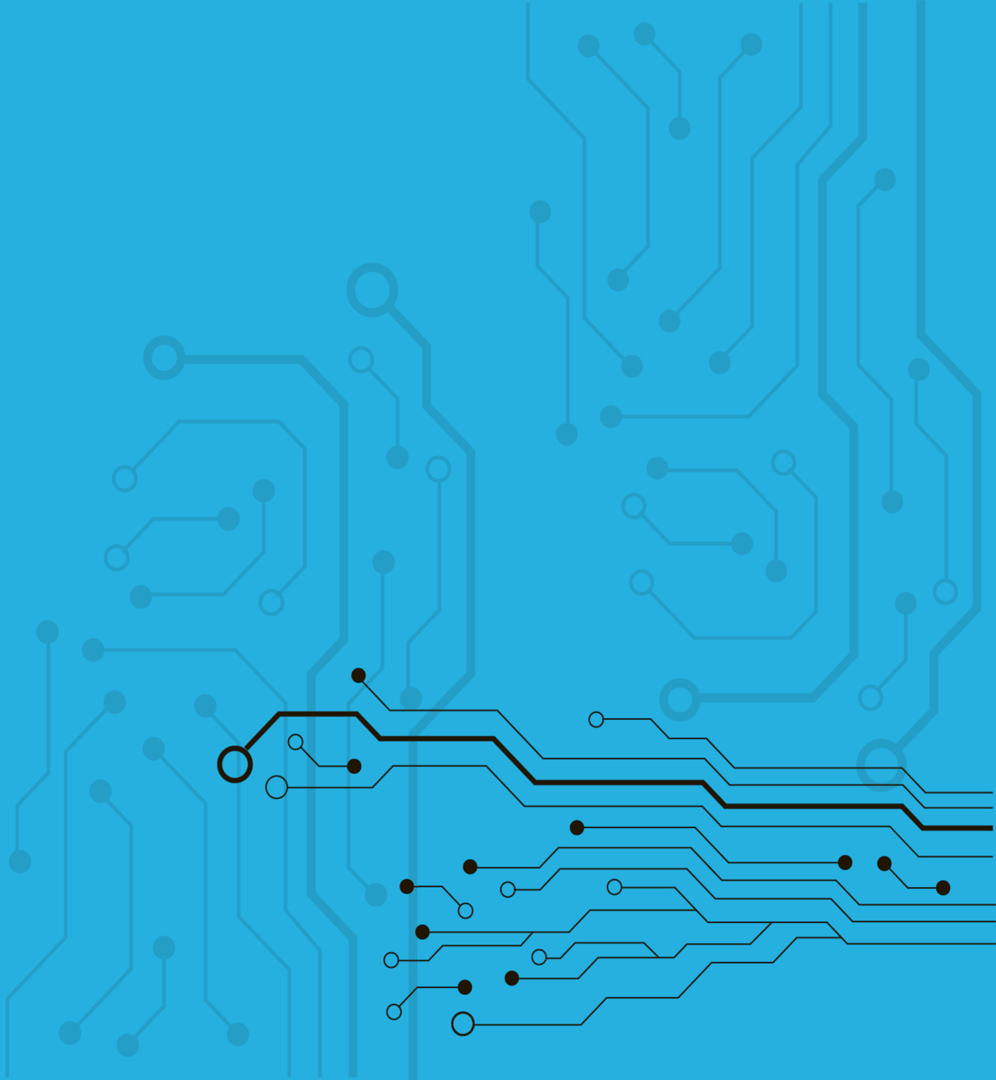
Exercise 1



Variables

- Variables are used to store data in memory
- This allows data to be referenced or changed at another point in a program
- Assignment in R
    - **=** vs. **<-**

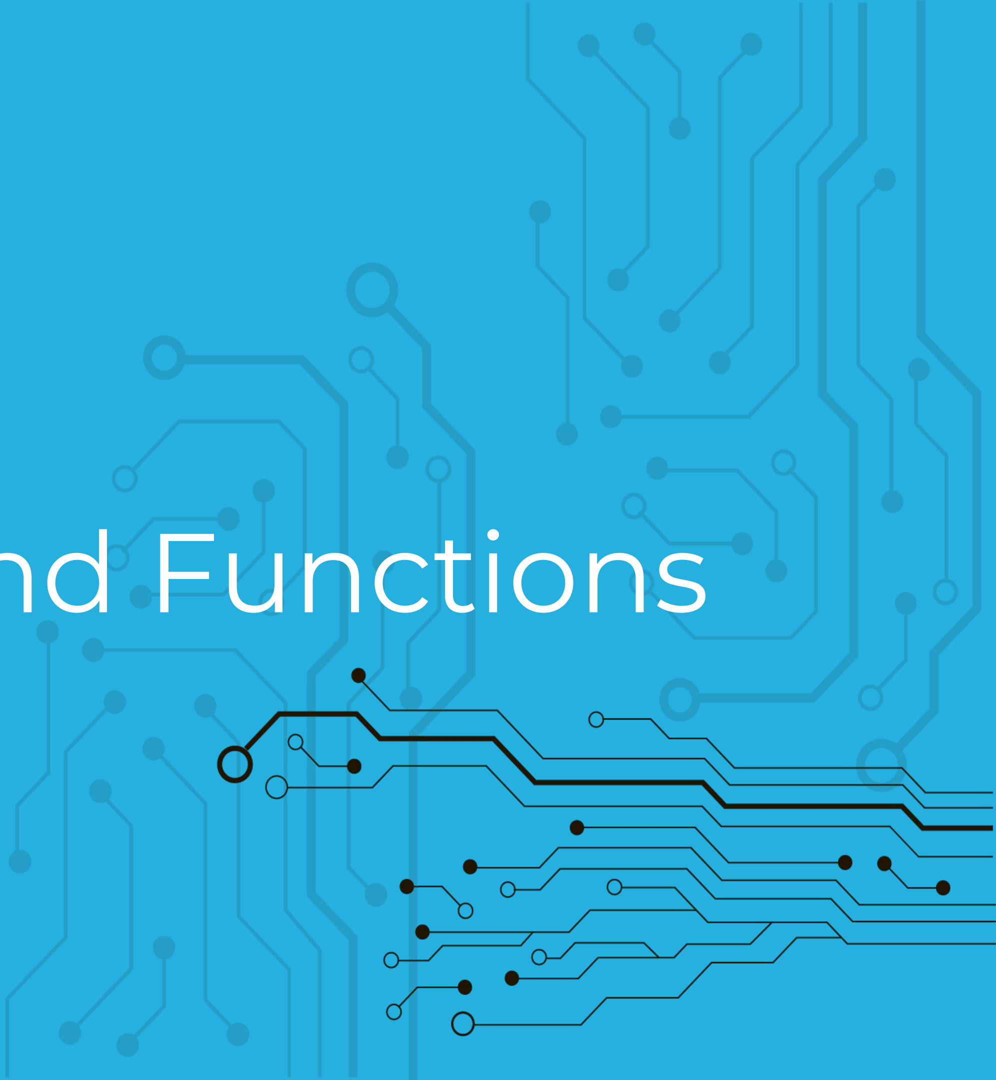
Exercise 2



Data structures in R

- **Vectors**
  - 1D sequences of data containing the same type of data in every cell
- **Lists**
  - Extensible with named elements
  - Can store multiple data types
- **Matrices**
  - 2D vectors
- **Arrays**
  - N-dimensional vectors
- **Data frames**
  - Rectangular data structures (think tables) with row and column names

Control Flow and Functions



Control flow

- **Conditional logic**
    - Statements:
        - **if (**condition**) {}**
        - **else if (**condition**) {}**
        - **else (**condition**) {}**
    - Logical operators:
        - and: **&**
        - or: **|**
        - not: **!**
    - Comparison operators:
        - greater than: **>**
        - greater than or equal to: **>=**
        - less than: **<**
        - less than or equal to: **<=**
        - equal to: **==**
        - not equal to: **!=**

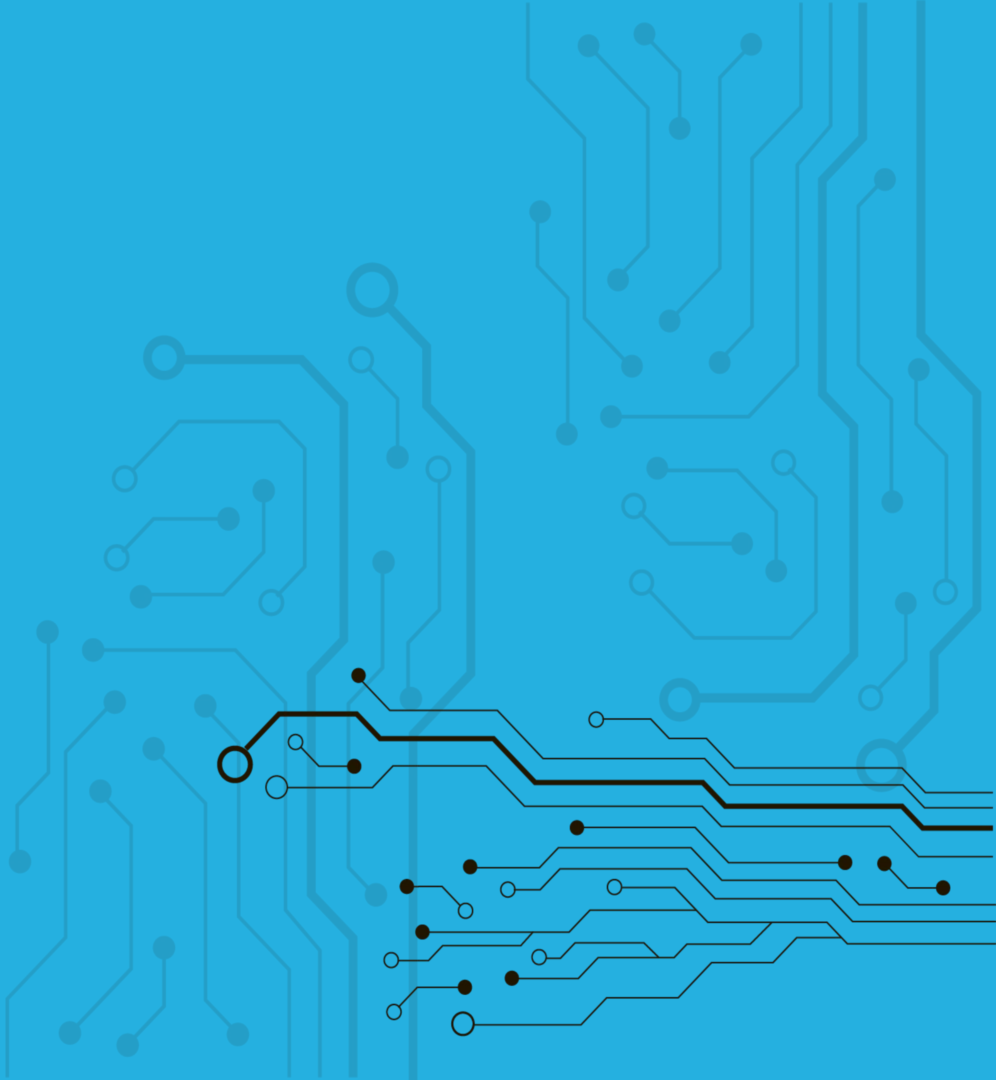
Control flow continued

Loops

- Types:
  - **for** (element **in** iterable) **{}**
  - **while** (condition) **{}**

- Continuation:
  - **next**
  - **break**

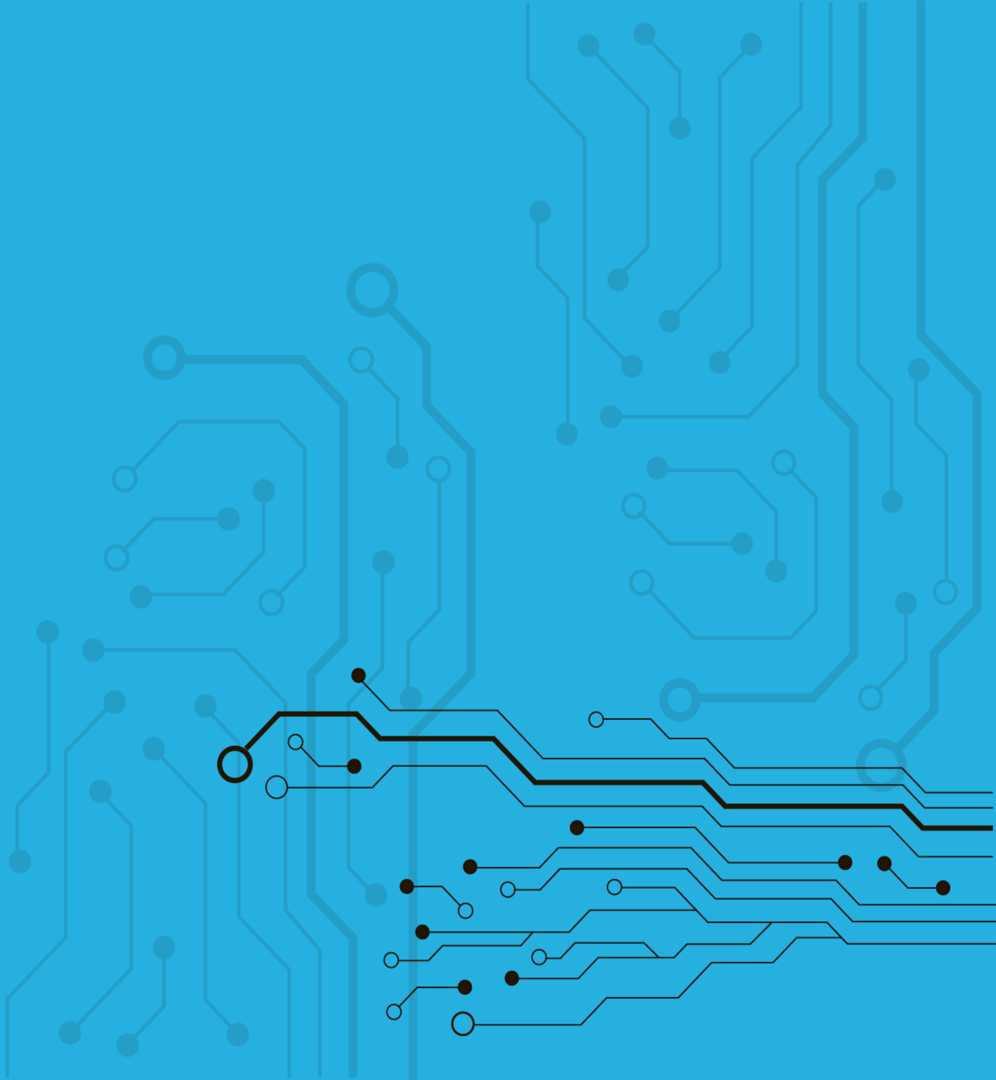
Exercise 3



Functions

- Enhance code reuse
- Add modularity
- Result in more readable code
- Introduce variable scope

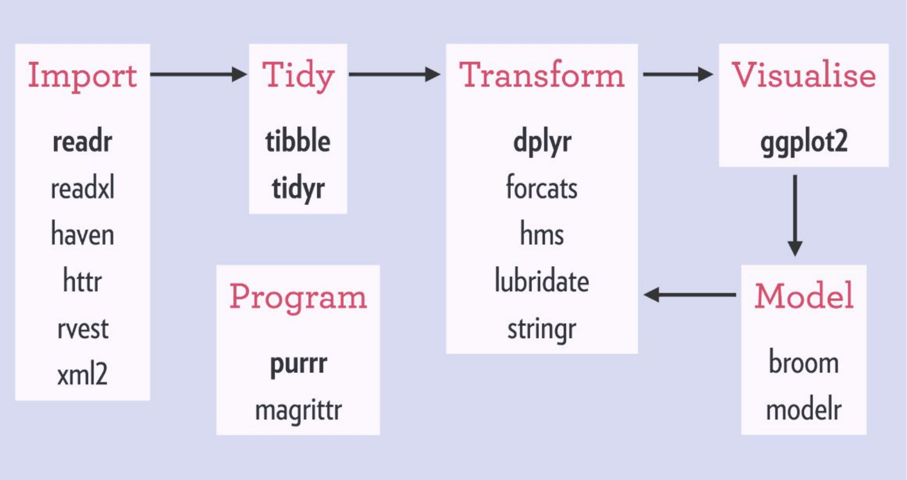
Exercise 4



Visualization and Reporting

Tidyverse

- Tidyverse is **the** essential collection for data science in R
- From the same group of developers that produced RStudio
- Simplifies and standardizes interaction with data in R

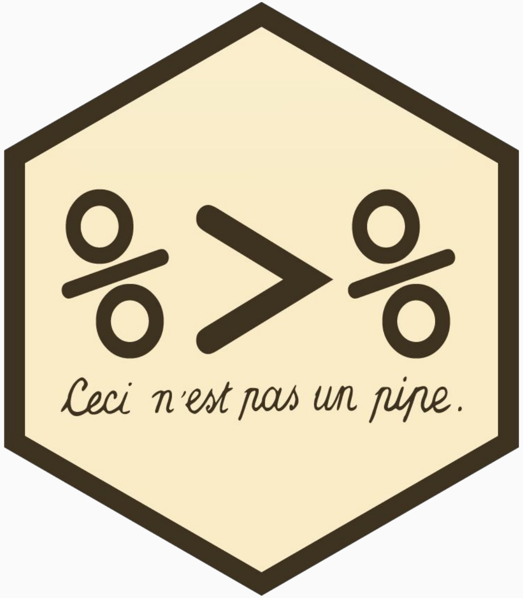


dplyr: data manipulation

- **Select**
  - Extract variables of interest from a data frame
- **Filter**
  - Subset data on specific criteria
  - Multiple conditions linked by logical operators
- **Mutate**
  - Edit or add new columns
- **Summarize**
  - Reduce a data frame to a set of summary statistics
- **Arrange**
  - Alter the order of a data frame

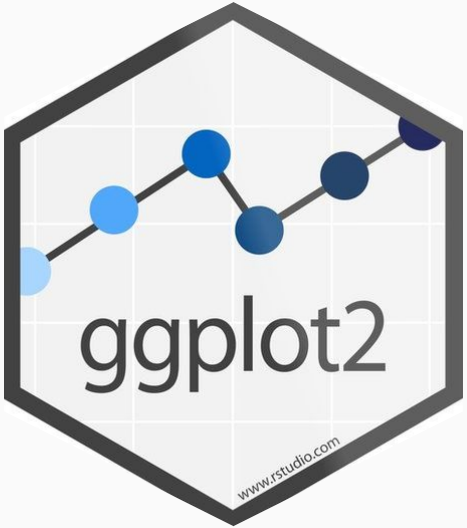
magrittr: cleaner code

- **Pipes**
  - Allows construction of “pipelines,” connecting the output of one expression to the input of the next
  - Pipe operator: **%>%**
  - Improves readability of code
  - Reduces verbosity
  - Encourages thoughtful consideration of code



ggplot2: information visualization

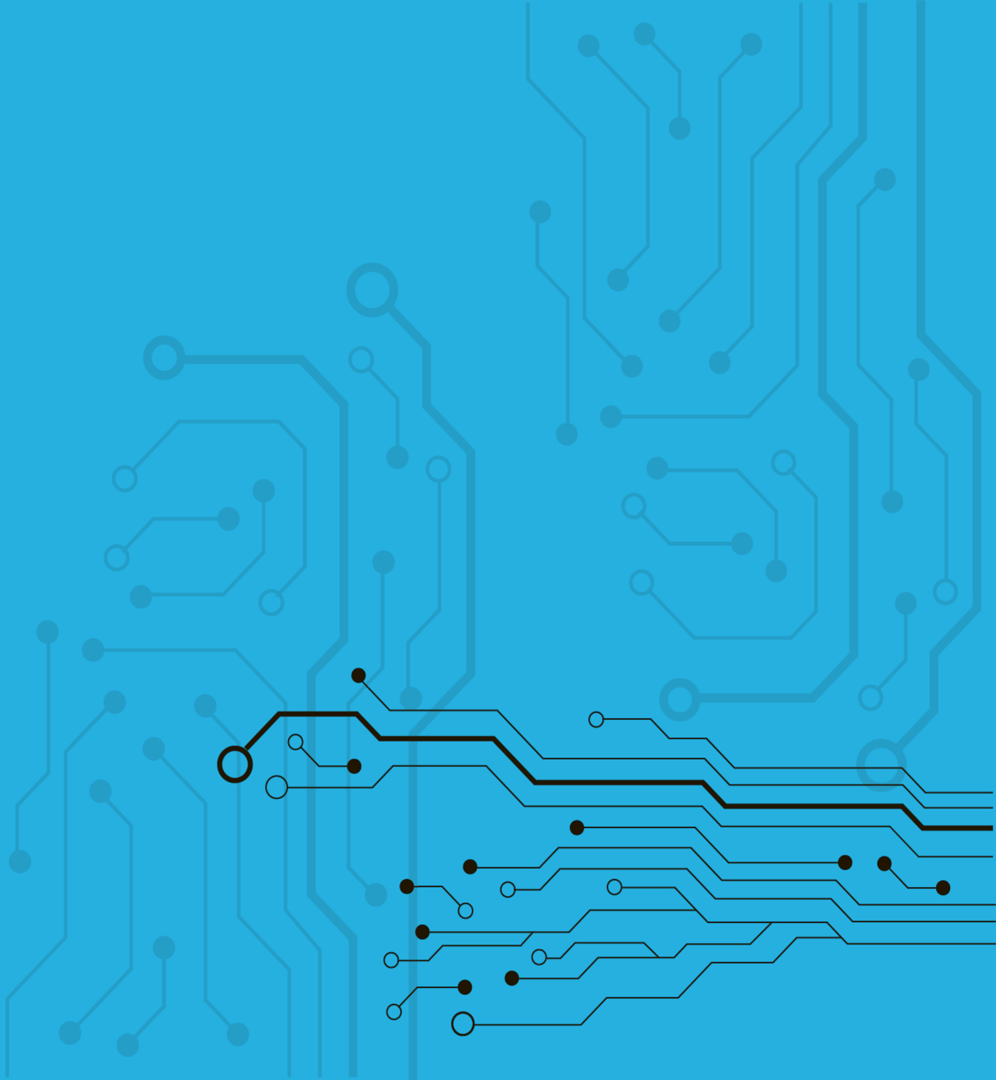
- **Grammar of graphics** (that’s the gg)
  - Syntax intended to encourage conformation to proper graphical thought process
  - Graphics are built up from a series of layers
- **Layers**
  - Data
  - Aesthetics
  - Geometries
  - Statistics
  - Scale
  - Facets
  - Themes
  - Legends and labels



Additional Tidyverse packages

- **tibble**: modernized data frames
  - All variables are columns
  - All columns are variables
  - Methods for standardizing and reshaping data frames
- **readr**: reading rectangular data
- **purrr**: functional programming
- **tidyr**: tidying data
- **stringr**: string manipulations
- **forcats**: factors

Exercise 5

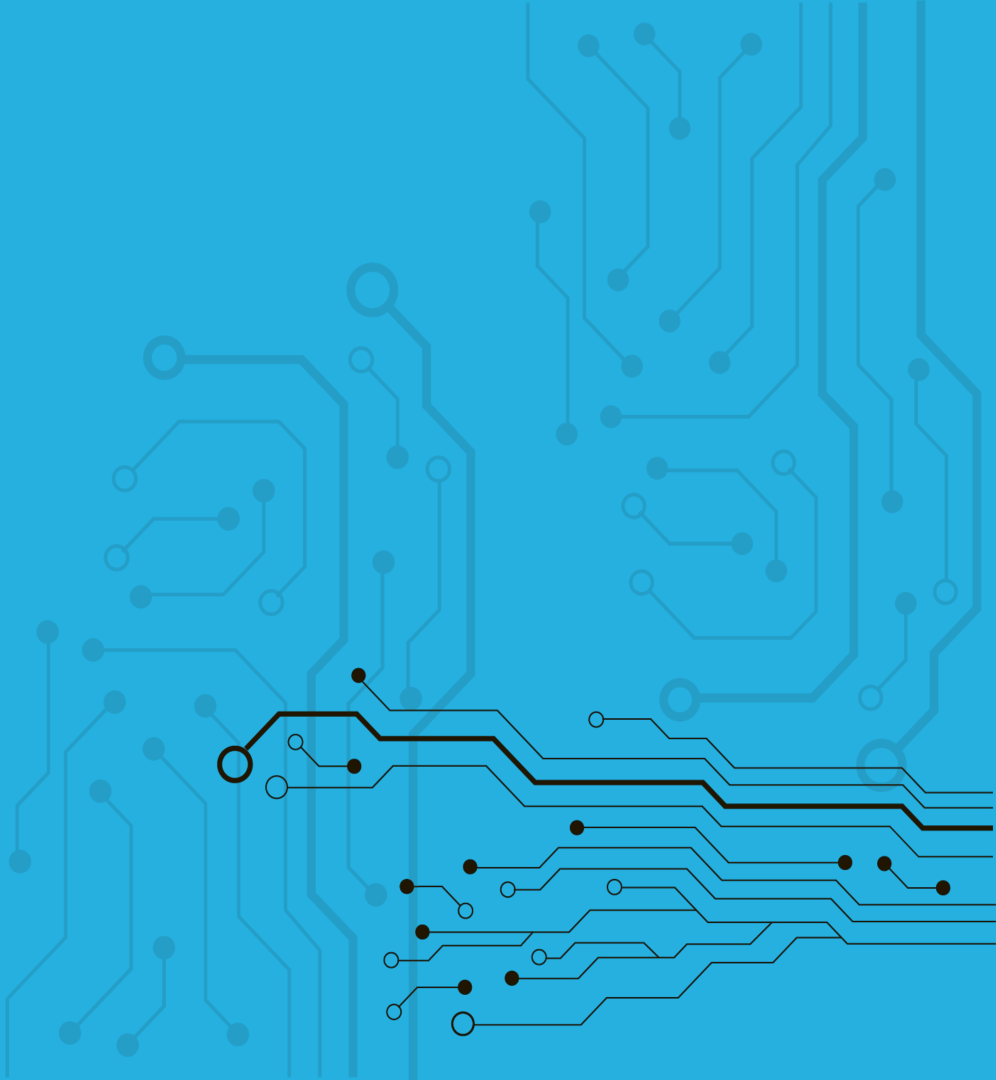


Git

What is Git?

- Essential Git commands
  - Clone
  - Pull
  - Commit
  - Push
- Gitignore
  - GitHub is intended for code, not data
  - Data is more likely to be sensitive than code
  - Use private repositories for things you don’t want to be public facing
  - Use local repositories only for particularly sensitive / controlled projects

Exercise 6



Additional resources

- R documentation
- CRAN
- Library documentation
- Google
- Wikipedia
- Stack overflow
- Quora
- GitHub student package
- Useful libraries
  - Bioconductor
  - jsonlite
  - Rtsne
  - caret

Thank you!!

Please fill out the feedback sheet

Theodore Smith
smithtg@email.arizona.edu