

Fundamentals of R

Pre-survey: tinyurl.com/rcac-pre-survey



Rosen Center for
Advanced Computing

Pre-survey



1

Getting Started

Agenda, Connecting to Training Account

Getting Started

Agenda

Time	Activity
9:00am	Introduction to R and RStudio, Data Structures in R
10:45am	Break
11:00am	Control Structures and Functions, Data Import/Export
12:30pm	Break
1:30pm	Data Manipulation and Visualization
3:30pm	Break
3:45pm	Statistical Analysis
4:30pm	Q&A

Getting Started

Connecting to RStudio Session on Negishi

1. <https://gateway.negishi.rcac.purdue.edu/>
2. Interactive Apps > GUIs > RStudio Server
3. Request for the following job -
 - R version – 4.4.1
 - Queue – Workshop
 - Walltime in Hours – 8
 - Number of cores for your job – 2
4. Launch
5. My Interactive Sessions > Connect to RStudio Server
6. Open Shell
 - `cp -r /depot/workshop/data/r-fundamentals/ .`

2

Introduction to R

What is R and RStudio?

Introduction to R

What is R?

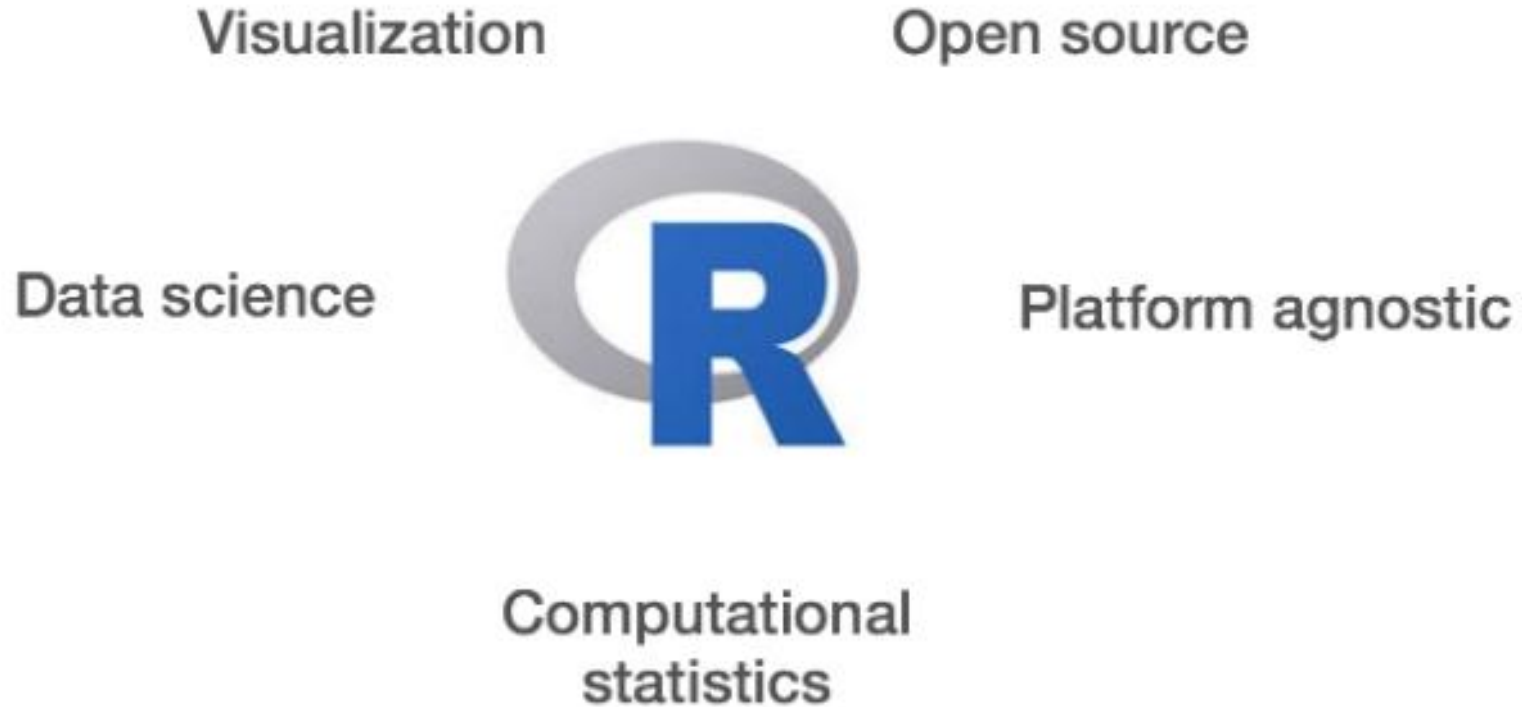
R is a free, open-source programming language for statistical computing and data visualization.

The R environment combines:

- effective handling of big data
- collection of integrated tools
- graphical facilities
- simple and effective programming language

Introduction to R

Why use R?



Introduction to R

What is RStudio?



Graphical user interface, not just a command prompt



Great learning tool



Free for academic use



Platform agnostic



Open source

3

R Basics

R syntax, variables, data types and math operations

R Basics

R Syntax and Basic Operations

The main "parts of speech" in R (syntax) include:

- The comments `#` and how they are used to document function and its content
- Variables and functions
- The assignment operator `<-`
- The `=` for arguments in functions

NOTE: indentation and consistency in spacing is used to improve clarity and legibility

R Basics

Variables and Assignment

Variables

- Symbolic names for storing data, similar to "buckets".
- name of the variable acts as a reference to the data stored in the bucket.

Assignment Operator (<-)

- Assign values to variables with the assignment operator
- Example: `x <- 3` assigns the value 3 to the variable x

Creating and Using Variables

- Example: Create a variable y and assign it a value of 5: `y <- 5`.
- Variables do not display their values in the console after assignment.

Performing Operations with Variables

- Use variable names to perform operations:
`x + y`.
- Assign operation results to another variable:
`z <- x + y`.

Printing and Viewing Variables

- Print a variable's value by simply typing its name in the console.
- Use parentheses to force the printing of a value during assignment, e.g., `(y <- 5)`.
- View variable information in RStudio Environment window.

R Basics

Variable Naming Conventions

- **Start names correctly:** Avoid names starting with a number; use x2 instead of 2x
- **Case Sensitivity:** Remember that R is case sensitive; genome_length and Genome_length are different
- **Clarity and Length:** Choose names that are explicit yet concise
- **Consistency in Style:** Adhere to a style guide, such as Hadley Wickham's or Google's style guide for R
- **Avoid Reserved Words:** Do not use names of fundamental functions like if, else, for, etc
- **Avoid Dots in Names:** Refrain from using dots (e.g., my.dataset) because they have special meanings in R and can be confused with methods
- **Nouns and Verbs:** Use nouns for variable names and verbs for function names

R Basics

Data Types

Data Type	Examples
Numeric	1, 1.5, 20, pi
Character	"sample text", "5", "TRUE"
Integer	2L, 500L, -17L
Logical	TRUE, FALSE, T, F
Complex	9i + 3



You can check the data type of a variable using the function `class()`

R Basics

Basic Math Operations

Basic Arithmetic Operations

- Perform standard arithmetic: +, -, *, / for addition, subtraction, multiplication, and division.
- Use ^ for exponentiation, e.g., 2^3 results in 8.

Built-in Math Functions

- Common functions include sum(), mean(), median(), etc.
- Calculate the sum of elements with sum(x), average with mean(x), and median with median(x).

Generating Sequences

- Create sequences using seq(): seq(from = 1, to = 10, by = 1) generates numbers from 1 to 10.
- Use : for simple sequences, e.g., 1:10 for the same effect.

Generating Random Numbers

- Use runif(n) for n uniform random numbers between 0 and 1.
- Generate normally distributed random numbers with rnorm(n, mean = 0, sd = 1).

R Basics

Hands On



1_basics.R

4

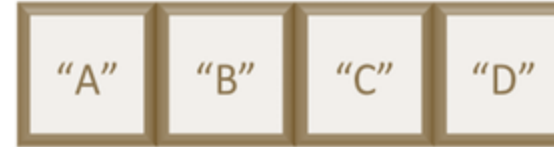
Data Structures

Vectors, Matrices, Arrays, Data Frames, Lists

Data Structures

Vectors

- Vectors are a basic data structure in R that store elements of the same data type (eg. character, logical, numeric)
- Common properties of vectors:
 - `typeof()`: what it is
 - `length()`: how many elements it contains
 - `attributes()`: additional arbitrary metadata



Data Structures

Factors

- Factors are a special type of vector that is used to store categorical data.
- Factors enable R to treat categorical variables differently from continuous variables.
- You can create factors in R using `factor()` or `as.factor()` functions.



Data Structures

Matrices and Arrays

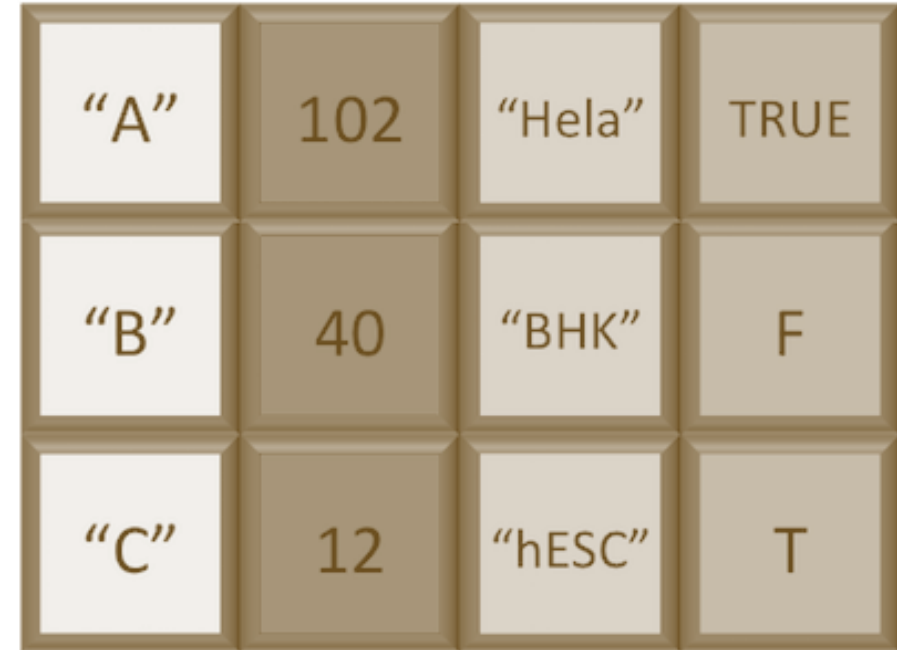
- Adding a *dim* attribute to an atomic vector allows it to behave like a multi-dimensional array
- A special case of array is the matrix, which has two dimensions i.e. rows and columns

90	5	137	9
87	40	2	52
4	102	32	41

Data Structures

Data Frames

- A Data Frame is a two-dimensional data structure, similar to a matrix but with a key difference: each column in a data frame can hold different types of data.
- Under the hood, a data frame is a list of equal-length vectors.

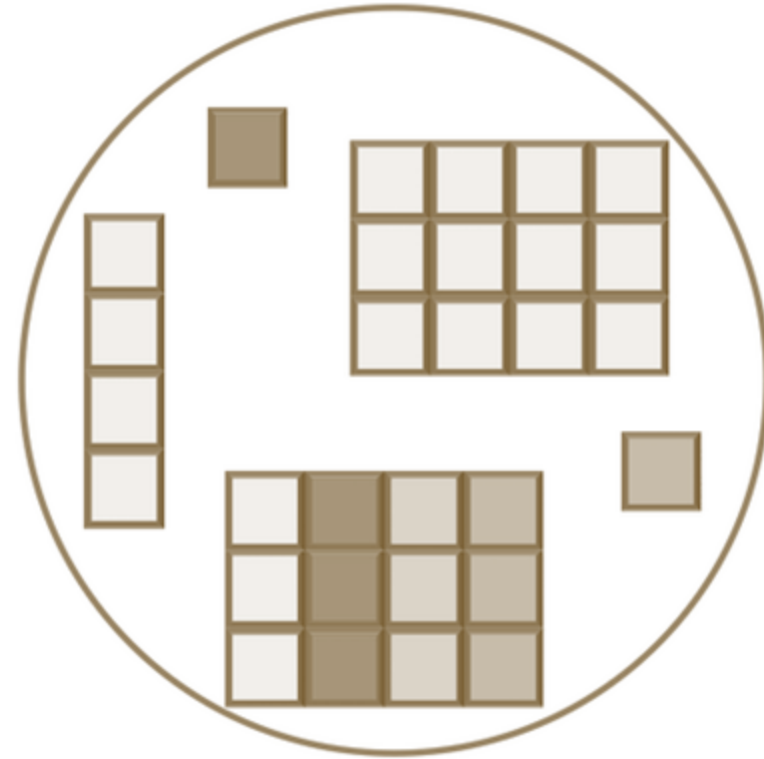


"A"	102	"Hela"	TRUE
"B"	40	"BHK"	F
"C"	12	"hESC"	T

Data Structures

Lists

- A list in R is an ordered collection of objects, which can be of different types and sizes.
- Lists can contain other lists.



Data Structures

Summary

Dimensions	Homogeneous	Heterogeneous
1	Atomic Vector	List
2	Matrix	Data Frame
n	Array	

Data Structures

Hands On



2_data_structures.R

Break

Up next : Control Structures, Functions, Working with Data

5

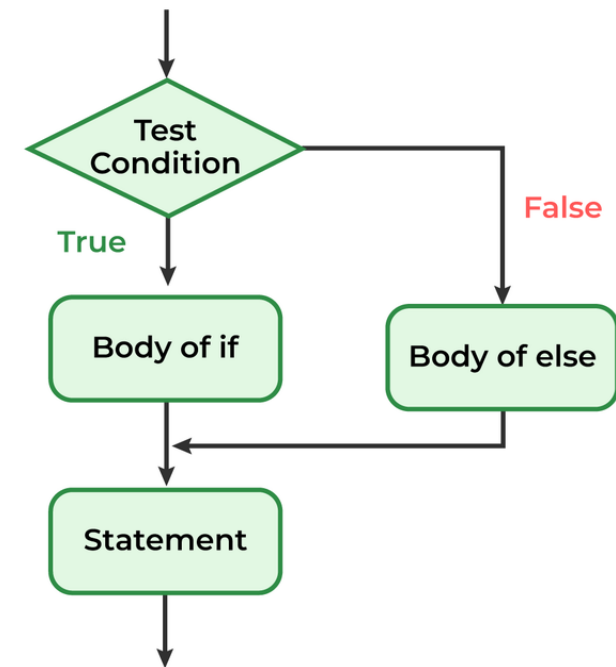
Control Structures

If-else, for loops, while loops, functions, anonymous functions

Control Structures

If-else

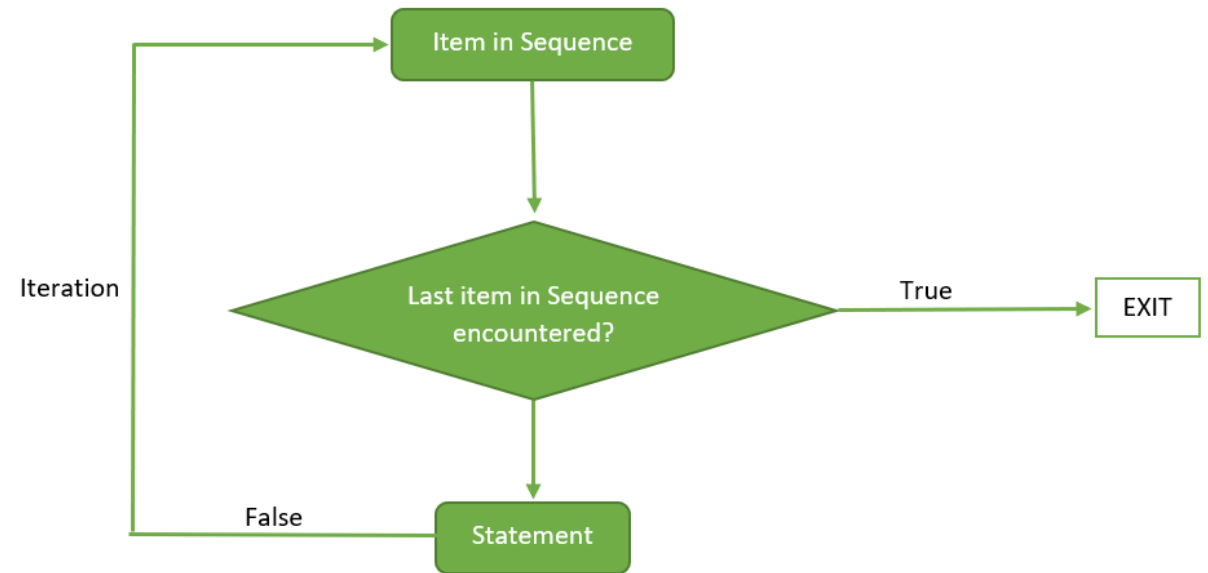
- **Simple If Statements:**
 - Execute a block of code conditionally.
 - Example: `if (condition) { execute this code }.`
- **If-Else Statements:**
 - Add an alternative action if the condition is false.
 - Example: `if (condition) { execute this } else { execute that }.`
- **Nested If-Else Statements:**
 - Place if-else structures inside another if-else, allowing for multiple conditions and outcomes.



Control Structures

For loops

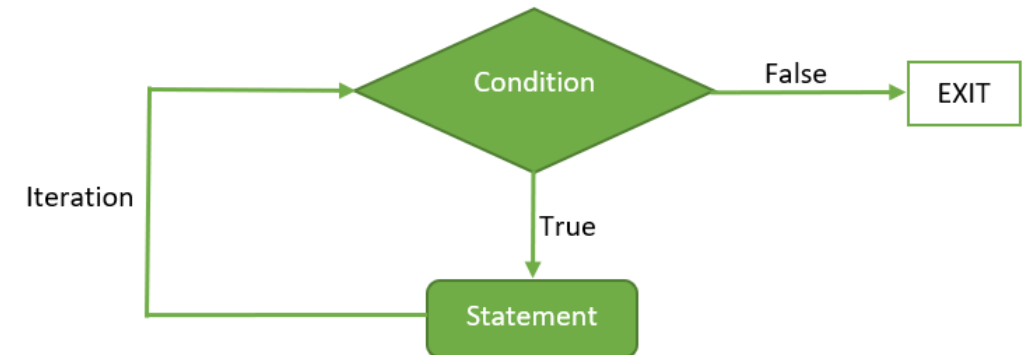
- **Basic Syntax and Usage:**
 - Loop through a block of code multiple times.
 - Syntax: `for (var in sequence) { execute code }.`
- **Iterating Over Vectors and Lists:**
 - Use for loops to perform operations on each element of a vector or list.
- **Nested For Loops:**
 - Include one for loop inside another to handle multi-dimensional data structures.



Control Structures

While loops

- **Basic Syntax and Usage:**
 - Repeat a block of code as long as a condition is true.
 - Syntax: `while (condition) { execute code }.`
- **Importance of Setting Proper Conditions:**
 - Ensure the loop has a stopping condition to prevent infinite loops.



6

Functions

Working with Functions and Anonymous Functions

Functions

Creating and Using Functions

- **Function Syntax in R:**
 - Define functions using
`function_name <- function(arg1, arg2) { code }.`
- **Parameters and Return Values:**
 - Pass data into functions via parameters and return a value using `return(value)`.
- **Default Arguments:**
 - Set default values for parameters to make functions more flexible.
- **Scope in R Functions:**
 - Variables created inside functions are local to those functions unless explicitly defined as global.

Functions

Anonymous Functions

- **Creating and Using Lambda Function:**
 - Define short, unnamed functions on the fly, e.g., `function(x) x^2`.
- **Applications in Apply Family of Functions:**
 - Use anonymous functions within `apply()`, `sapply()`, `lapply()`, etc., to perform operations across elements in data structures.

Control Structures and Functions

Hands On



3_control_structures_and
_functions.R

7

R Packages

Package Installation and Usage

R Packages

Installation and Usage

- Packages are collections of functions, data, and code designed to add specific features
- Over 10,000 user-contributed packages available
- Packages for R can be installed from the CRAN package repository using the `install.packages` function
- Installation example: `install.packages("ggplot2")`
- Loading example: `library(ggplot2)`

8

Working with Data

Reading Files, Inspecting the Dataset

Working with Data

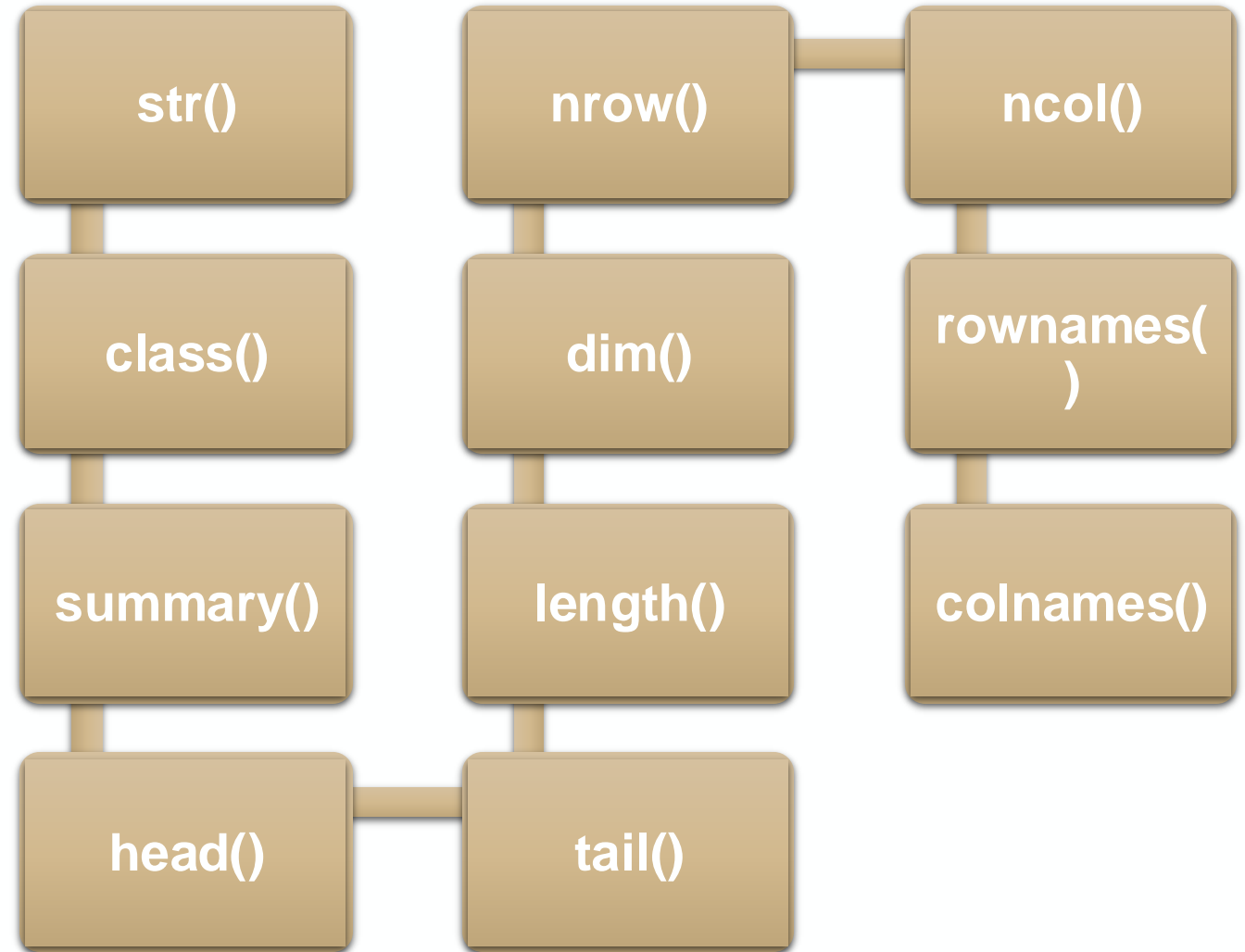
Reading Files

Data type	Extension	Function	Package
Comma separated values	csv	read.csv()	utils (default)
		read_csv()	readr (tidyverse)
Tab separated values	tsv	read_tsv()	readr
Other delimited formats	txt	read.table()	utils
		read_table()	readr
		read_delim()	readr

Working with Data

Inspecting the Dataset

- To understand the characteristics and to identify potential problems or issues with the data



Working with Data

Data Wrangling



how to subset vectors and
factors



use of logical operators when
subsetting vectors and factors



how to relevel factors in a
desired order

Working with Data

Hands On



4_data_import_export.R

Break

Up next : Data Manipulation, Data Visualization

9

Data Manipulation

Using dplyr and tidyr, Common dplyr functions

Data Manipulation

Using dplyr and tidyr

dplyr

- Provides easy tools for common data manipulation tasks.
- Built to work directly with data frames.
- Can work with data stored in an external database.
 - This feature helps overcome R's in-memory operation limitation.
 - Allows you to connect to large databases, conduct queries, and pull only the needed data into R for analysis.

tidyr

- Helps reshape data for plotting and use by different R functions.
- Facilitates moving between different data formats:
 - One row per measurement.
 - Data frames where each measurement type has its own column and rows are aggregated groups.
- Offers tools for sophisticated data manipulation.

Data Manipulation

Common dplyr Functions

Function	Use
<code>select()</code>	To subset columns
<code>filter()</code>	To subset rows on conditions
<code>mutate()</code>	Create new columns by using information from other columns
<code>group_by()</code> & <code>summarize()</code>	To create summary statistics on grouped data
<code>arrange()</code>	To sort results
<code>count()</code>	To count discrete values

Data Manipulation

Hands On



5_data_manipulation.R

10

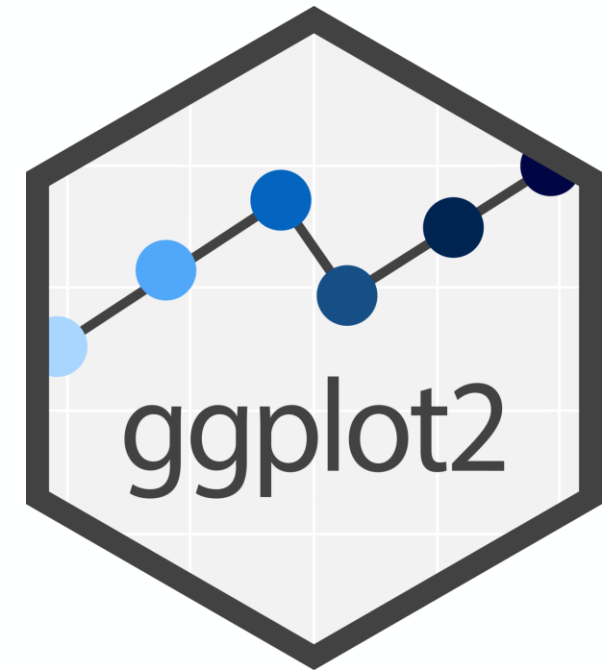
Data Visualization

ggplot2, creating plots with aesthetics, faceting and multi-layer plots

Data Visualization

ggplot2

- ggplot2 is part of the tidyverse, a collection of R packages designed for data science
- The functions in the ggplot2 package build up a graph in layers. We'll build a a complex graph by starting with a simple graph and adding additional elements, one at a time.

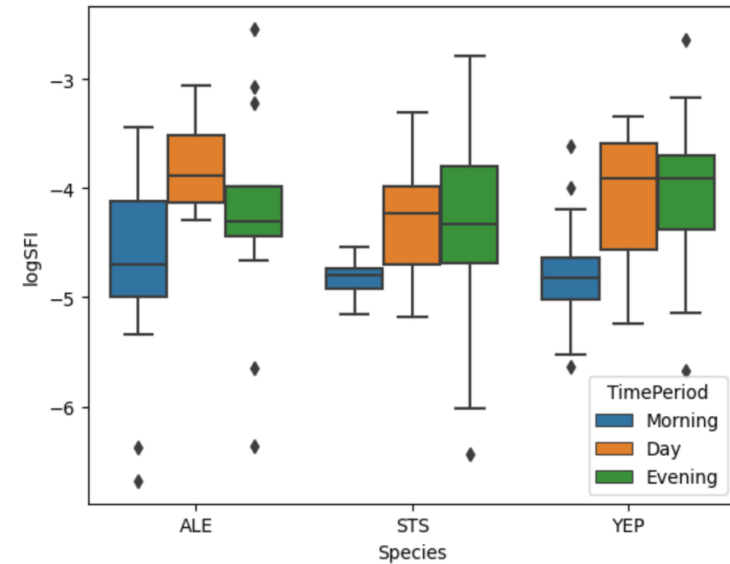
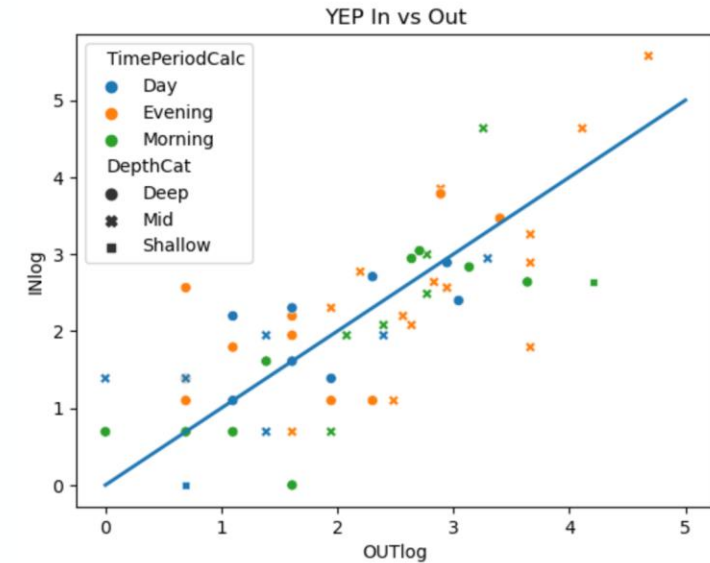


Data Visualization

The Power of ggplot2

Some key features of ggplot2:

- Versatility in plot types
- Layered approach in building plots
- Easy customization of aesthetics
- Built-in themes for quick styling
- Automatic legends and labels



Data Visualization

Grammar of Graphics Concept

- Layered approach to building plots:
 - Data : The dataset that is being plotted
 - Aesthetics : The scale onto which we map our data
 - Geometries : The visual elements used for the data
- Other grammatical elements:
 - Themes : allows you to exercise fine control over the non-data elements of your plot
 - Statistics : to understand the data
 - Coordinates : The space where the data will be plotted
 - Facets : if we need small multiples of the plots (Multiple plots - like scatterplot)



ggplot2 cheatsheet can be found at <https://ggplot2.tidyverse.org/>

Data Visualization

Creating Plots and Customizing Aesthetics

Creating Basic Plots

- Scatter Plots: Use `geom_point()` to visualize relationships between two continuous variables.
- Line Plots: Apply `geom_line()` to depict trends over time or ordered categories.
- Bar Plots: Implement `geom_bar()` for categorical data and `geom_col()` for plotting pre-summarized data.

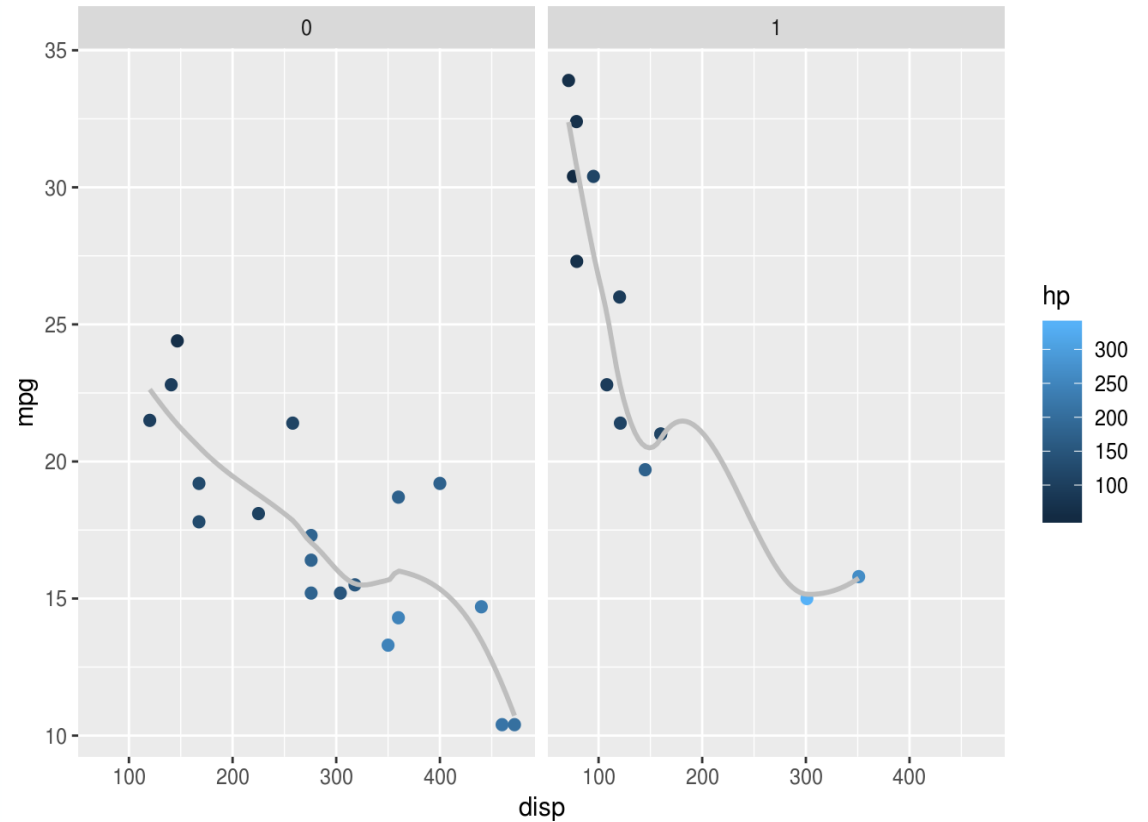
Customizing Plot Aesthetics

- Changing Colors, Shapes, and Sizes: Customize these aesthetics to enhance plot readability and appeal.
- Modifying Axes and Labels: Adjust text labels, tick marks, and axis titles to improve clarity.
- Adding Titles and Subtitles: Include descriptive titles and subtitles to provide context and guide interpretation.

Data Visualization

Faceting and Multi-layer Plots

- **Facets**
 - Use `facet_wrap()` for arbitrary layouts or `facet_grid()` for a matrix of plots based on factor variables, allowing comparisons across subsets of data.
- **Creating Complex, Multi-layer Plots**
 - Layer different geometric objects (geoms) to build complex plots.
 - Integrate statistical transformations like `stat_summary()` to compute summaries directly in plots.
 - Apply and modify themes to tailor the visual style of plots to specific requirements or aesthetics.



Data Visualization

Hands On



6_data_visualization.R

Break

Up next : Statistical Analysis in R

11

Statistical Analysis in R

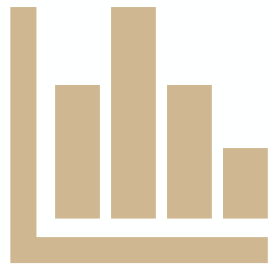
Descriptive Statistics, t-tests and ANOVA, Linear Regression, Logistic Regression, Introduction to Machine Learning in R

Statistical Analysis in R

Descriptive Statistics and Comparative tests

Descriptive Statistics

- Measures of Central Tendency and Dispersion: Understand mean, median, mode, variance, and standard deviation.
- Using summary() Function: Get a quick statistical summary of data sets including min, max, quartiles.



T-tests and ANOVA

- Conducting t-tests: Use t.test() to compare means between two groups.
- One-way ANOVA: Apply aov() to analyze differences among three or more group means.

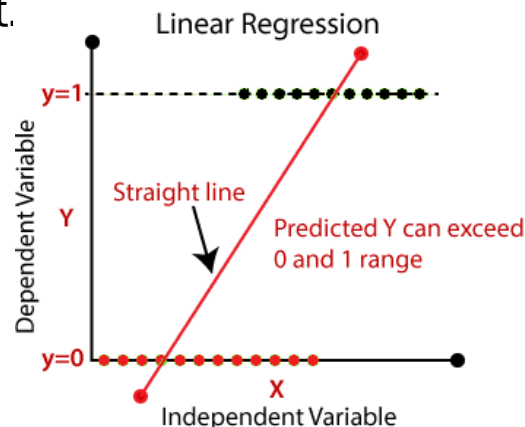


Statistical Analysis in R

Linear Regression and Logistic Regression

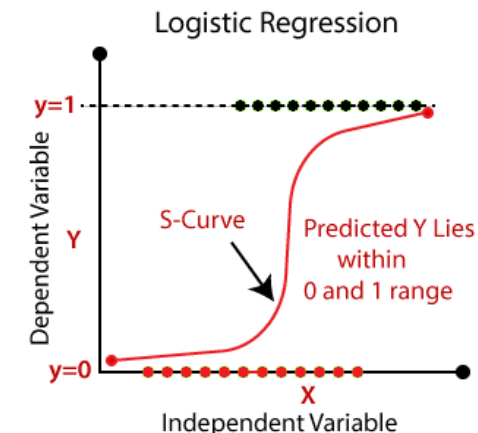
Linear Regression

- Simple Linear Regression: Use `lm()` to model the relationship between two continuous variables.
- Multiple Regression: Extend `lm()` to include multiple predictors.
- Interpreting Regression Output: Focus on coefficients, R-squared, and p-values for model assessment.



Logistic Regression

- Binary Outcomes: Model binary outcomes using `glm()` with a binomial family.
- Interpreting Logistic Regression Results: Analyze coefficients, odds ratios, and significance to understand effect sizes and predictive accuracy.



Statistical Analysis in R

Some Common R Packages for Statistical Analysis

- MASS (Modern Applied Statistics with S)
 - Common methods - `lm()`, `glm()`, `stepAIC()`, `polr()`
- caret (Classification And REgression Training)
- lme4 (linear mixed-effects models)
 - Common methods - `lmer()`, `glmer()`,
- nlme (NonLinear Mixed-Effects models)
 - Common methods - `nlme()`
- survival (Survival analysis)
 - Common methods - `Surv()`, `survfit()`, `coxph()`

Statistical Analysis in R

Hands On



7_statistical_analysis.R

12

Q&A

Thank You

Training catalog: rcac.purdue.edu/training

Women in HPC: rcac.purdue.edu/whpc

Consulting hours: rcac.purdue.edu/coffee

Careers: rcac.purdue.edu/careers

Post-survey: tinyurl.com/rcac-post-survey

Post-survey

