KTP Coding Workshop

Coding in R: Basics

Schedule

01

What is R?

02

Basic Syntax

03

Data Structures

04

Graphics

05

Statistics

06

Practice

01

What is R?

What is R?

- **Definition:** R is an open-source programming language developed specifically for statistical analysis, data visualization, and data manipulation
- **Core Features:** Data analysis, data visualization, data manipulation
- **Strengths:**
  - Tailored for statistical computing and graphics
  - Thousands of packages for various applications

Who uses R?

→ **People using data:** Used by data scientists, statisticians, and researchers in fields such as business, healthcare, finance, and academia.

→ **Benefits :**

- Powerful tools for managing and visualizing data.
- Extensive libraries for statistical tests, data manipulation, and predictive modeling.

Download R + R Studio

https://posit.co/download/rstudio-desktop/

- Different setup for everyone!

02

Basic Syntax

Syntax

Printing:

Printing a number:
1, 2 ,3

Printing words: Just use quotes
"Hello World!"

Comments:

Use a # to indicate a comment

Variables:

You can create a variable and assign it a value with
<- or =

Data Types

Numeric

(5, 55, 555.5)
Ex: x <- 1

Integer

(5G, 55G, 555G, where the letter "G" declares this as an integer)
Ex: t <- 1G

Char/String

(A,B,C, This is amazing)
Ex: word <- ‘A’

Booleans

Represents a true or false value
Ex: z<- true

Fun Functions

Basic math

Addition: +
Subtraction: -
Multiplication : *
Division: /

Max & Mins

max(1, 10, 100)

min(1, 10, 100)

Roots & abs

Square Root =sqrt()
Abs value= abs()

Operators

Arithmetic

All the same math functions!

Logical

&& - and
|| - or
! - not

Assignment

<-
=
<<- (global)

Comparison

= - equal to
<= - less than or equal
>= - greater than or equal
> - greater
< - less

Loops

Loops are essential for automating repetitive tasks, processing data, and iterating over elements in structures like vectors and lists.

For Loop

Use When: Iterating over a fixed sequence, like elements in a vector or a range of numbers.

Ex:

```
for (i in 1:5) {
   print(i)
}
```

While Loop

Use When: The number of iterations is unknown; loop continues as long as a condition is TRUE.

Ex: (note must have an initial index)

```
x <- 1  # Initial index
while (x <= 5) {
   print(x)
   x <- x + 1  # Update index to prevent infinite loop
}
```

Definition: Functions are reusable blocks of code that perform specific tasks.

Purpose: Organize code, reduce repetition, and make programs modular and readable.

Calling Functions

```
my_function <- function() { # create a function with the name my_function
  print("Hello World!")
}
```

Syntax

Calling on function

Function:

```
greet <- function(name
= "R User") {
print(paste("Hello,",
name)) }
```

Call

```
greet()
greet(“Aaron”)
```

03

Data Structures

What happens when you want to store multiple pieces of data?

Vectors:A one-dimensional data structure with elements of the same type.

Creating a vector:

numbers <- c(1, 2, 3, 4)

vector names <- c("Alice", "Bob", "Charlie")

Accessing elements:

Use square brackets
Numbers[2]

Vector names [3]

Lists:One-dimensional structure that can contain elements of different types (e.g., numeric, character, vectors).

Creating lists:

person <- list(name="Alice", age=25, scores=c(80, 90, 85))

Accessing elements:

Use $ or [[ ]]

person$name
person[[2]]

Matrices:Two-dimensional data structure with rows and columns, all elements of the same type.

Creating matrix:

matrix_data <- matrix(1:9, nrow=3, ncol=3)

- This creates a 3x3 filled from 1-9

Accessing elements:

Use [row,col]

matrix_data[2, 3

04

Graphics

Array:Matrices but longer

Creating array:

array_data <- array(1:12, dim = c(3, 2, 2))

- This creates a 3x3x2 filled 1-12

Accessing elements:

Use [spot,spot,spot]

array_data[2, 1, 1]

Plots + line plots:

Overview of `plot()`:

- The `plot()` function in R is a versatile tool for creating basic diagrams by plotting points.
- **Usage**: At its simplest, `plot()` plots coordinates on a graph, with parameters for specifying points along the x- and y-axes.

Basic Syntax:

```
plot(x, y)
```

- **Parameter 1** (x): Specifies the position on the x-axis.
- **Parameter 2** (y): Specifies the position on the y-axis.

Example: Plotting a Single Point

To plot a single point at (1, 3):

```
plot(1, 3)
```

```
NOTE:To make it a line plot, just add a ,type="l"
```

Scatter Plots:

Requires two vectors of the same length: one for the x-axis and one for the y-axis.

Same syntax of plot: plit (x,y) where x and y are the vector values

Ex :

X <- c(1,2,3,4,5)
Y <- c(1,2,3,4,5)

Plot (x,y)

Pie Charts

The pie function creates a pie graph pie()

You create a vector of values and call on it

Ex:

```
X<-c(1,2,3)

pie(X)
```

Bars

Similar to the scatter plots of using two vectors using barplot(y,name.arg=x)

Ex:

```
X <- c (“A”,”B”,”C”)

Y < - c(1,2,3)

barplot(y,name.arg=x)
```

Changing Aspects of Graph

Titles and axis labels:

- Set new parameter within parenthesis and do main = “custom title”
- Set new parameters of xlab and ylab for x and y labels where its = “label”

Colors:

- Set new parameters within parenthesis using col =”color”
- Background color is bg =”color”

Adjusting ranges:

- Set xlim and ylim as limits for x and y axes where x/y lim =c(limit,limit)

05

Statistics

Descriptive Stats

Maximum and Minimum values can be found using

1. min (data)
2. max(data)

Mean Medians Modes are found the same way

1. mode(data)
2. median(data)
3. mean(data)

Ex:

```
data <- c(4, 8, 6, 5, 3, 9, 12)
mode(data)
median(data)
mean(data)
min (data)
max(data)
```

T tests

T tests: tests if two sample means are significantly different

Syntax: t.test(x,y,)

Ex:

```
sample1 <- c(5, 6, 7, 8, 9)
sample2 <- c(7, 8, 9, 10, 11)
t.test(sample1, sample2)
```

06

Review!

Review

Section 1: Why Use R?

- Advantages: Statistical analysis, data visualization, data manipulation.

Section 2: Basic Syntax

- **Variables**: Declare using `<-` or `=`.
- **Data Types**: Numeric, Integer, Character, Boolean.
- **Loops**: `for` and `while` loops for iteration.
- **Functions**: Defined using `function()`. Example: `my_function <- function() {print("Hello!")}`.

Section 3: Data Structures

- **Vectors**: 1D data, same type.
- **Lists**: 1D data, mixed types.
- **Matrices**: 2D, same type.
- **Arrays**: Multi-dimensional, same type.

Review

Section 4: Graphics

- **Basic Plotting**: `plot(x, y)`.
- **Customization**: Title (`main`), axes labels (`xlab`, `ylab`), color (`col`).

Section 5: Statistics

- **Descriptive Stats**: Mean, Median, Mode, Min, Max.
- **T-tests**: `t.test(x, y)` to compare sample means.

Moving forward!

- There’s so much more to R than what we covered here!
- Try experimenting with what you’ve learned and explore new functions and packages.
- R has tools for everything, from quick data summaries to complex visualizations.
- Check out more resources online, try real-world projects, and practice often.
- Remember: the more you play around with R, the more you'll discover!

ΚΘΠ