Welcome to Data Science: R Basic

edx : HarvardX : PH125.1x

# Section 1: R Basic, Functions and Data Types

## Motivation

### Key Points

* In the first few videos we will go over logistics and introduce general concepts.
* Rather than cover every R skill you need, we will provide the building blocks so you can grow your knowledge as you go through the series.
* Because we better retain knowledge by learning through solving a specific problem, we will use a motivating case study - crime in the United States - to answer specific questions.

## Installing R

### Key points

* You need to install R before using RStudio, which is an interactive desktop environment.
* Select base subdirectory in CRAN and click download.
* Select all default choices in the installation process.
* We recommend selecting English for language to help you better follow the course.
* You can try using the R console, but for productivity purposes, we can switch to RStudio.

### Download R from CRAN

You can find the latest version of R for your operating system at the [CRAN website](https://cran.r-project.org/).

## Installing RStudio

### Key points

* You can download the latest version of RStudio at the [RStudio website](https://www.rstudio.com/products/rstudio/download/).
* The free desktop version is more than enough for this course.
* Make sure to choose the version for your own operating system.
* Choose "Yes" for all defaults in the installation process.

## Getting Started Using R

### Key Points

* R was developed by statisticians and data analysts as an interactive environment for data analysis.
* Some of the advantages of R are that (1) it is free and open source, (2) it has the capability to save scripts, (3) there are numerous resources for learning, and (4) it is easy for developers to share software implementation.
* Expressions are evaluated in the R console when you type the expression into the console and hit Return.
* A great advantage of R over point and click analysis software is that you can save your work as scripts.
* “Base R” is what you get after you first install R. Additional components are available via packages.

### Code: using install.packages and library()

# installing the dslabs package

install.packages("dslabs")

# loading the dslabs package into the R session

library(dslabs)

## Installing Packages

### Key points

* The base version of R is quite minimal, but you can supplement its functions by installing additional packages.
* We will be using **tidyverse** and **dslabs** packages for this course.
* Install packages from R console: install.packages("pkg\_name")
* Install packages from RStudio interface: Tools > Install Packages (allows autocomplete)
* Once installed, we can use library(pkg\_name) to load a package each time we want to use it

### Additional Notes

* If you try to load a package with library(blahblah) and get a message like Error in library(blahblah) : there is no package called 'blahblah', it means you need to install that package first with install.packages().
* On the DataCamp interface we use for some problems in the course, you cannot install additional packages. The problems have been set up with the packages you need to solve them.
* You can add the option dependencies = TRUE, which tells R to install the other things that are necessary for the package or packages to run smoothly. Otherwise, you may need to install additional packages to unlock the full functionality of a package.
* Throughout the course materials and textbook, package names are in **bold**.

### Code

install.packages("dslabs") # to install a single package

install.packages(c("tidyverse", "dslabs")） # to install two packages at the same time

installed.packages() # to see the list of all installed packages

## Running Commands While Editing Scripts

### Key points

* RStudio has many useful features as an R editor, including the ability to test code easily as we write scripts and several autocomplete features.
* Keyboard shortcuts:
  + Save a script: Ctrl+S on Windows and Command+S on Mac
  + Run an entire script:  Ctrl+Shift+Enter on Windows Command+Shift+Return on Mac, or click "Source" on the editor pane
  + Run a single line of script: Ctrl+Enter on Windows and Command+Return on Mac while the cursor is pointing to that line, or select the chunk and click "run"
  + Open a new script: Ctrl+Shift+N on Windows and Command+Shift+N on Mac

### Code

library(tidyverse)

## R Basics: Objects

### Key Points

* To define a variable, we may use the assignment symbol, <-.
* There are two ways to see the value stored in a variable: (1) type the variable name into the console and hit Return, or (2) use the print() function by typing print(variable\_name) and hitting Return.
* Objects are things that are stored in named containers in R.  They can be variables, functions, etc.
* The ls() function shows the names of the objects saved in your workspace.

### Code: solving the equation **x2+x−1=0**

# assigning values to variables

a <-1

b <-1

c <--1

# solving the quadratic equation

(-b + sqrt(b^2 - 4\*a\*c))/(2\*a)

(-b - sqrt(b^2 - 4\*a\*c))/(2\*a)

## Functions

### Key points

* In general, to evaluate a function we need to use parentheses. If we type a function without parenthesis, R shows us the code for the function. Most functions also require an argument, that is, something to be written inside the parenthesis.
* To access help files, we may use the help function, help(function\_name), or write the question mark followed by the function name, ?function\_name.
* The help file shows you the arguments the function is expecting, some of which are required and some are optional. If an argument is optional, a default value is assigned with the equal sign. The args() function also shows the arguments a function needs.
* To specify arguments, we use the equals sign. If no argument name is used, R assumes you’re entering arguments in the order shown in the help file.
* Creating and saving a script makes code much easier to execute.
* To make your code more readable, use intuitive variable names and include comments (using the “#” symbol) to remind yourself why you wrote a particular line of code.

## Data Types

### Key Points

* The function class() helps us determine the type of an object.
* Data frames can be thought of as tables with rows representing observations and columns representing different variables.
* To access data from columns of a data frame, we use the dollar sign symbol, $, which is called the accessor.
* A vector is an object consisting of several entries and can be a numeric vector, a character vector, or a logical vector.
* We use quotes to distinguish between variable names and character strings.
* Factors are useful for storing categorical data, and are more memory efficient than storing characters.

### Code

# loading the dslabs package and the murders dataset

library(dslabs)

data(murders)

# determining that the murders dataset is of the "data frame" class

class(murders)

# finding out more about the structure of the object

str(murders)

# showing the first 6 lines of the dataset

head(murders)

# using the accessor operator to obtain the population column

murders$population

# displaying the variable names in the murders dataset

names(murders)

# determining how many entries are in a vector

pop <- murders$population

length(pop)

# vectors can be of class numeric and character

class(pop)

class(murders$state)

# logical vectors are either TRUE or FALSE

z <- 3 == 2

z

class(z)

# factors are another type of class

class(murders$region)

# obtaining the levels of a factor

levels(murders$region)

### **Question 1**

2/2 points (graded)

To find the solutions to an equation of the format ax2+bx+c, use the quadratic equation: x=−b±b2−4ac√2a.

What are the two solutions to 2x2−x−4=0 ? Use the quadratic equation.

Report the positive solution first, using 3 significant digits for both solutions.

  correct

1.686 Loading

  correct

−1.186 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 2**

1/1 point (graded)

Use R to compute log base 4 of 1024. You can use the help() function to learn how to use arguments to change the base of the log() function.

Submit your answer AS A NUMBER below.

  correct

5 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

Install the **dslabs** package if you have not done so:

install.packages("dslabs")

**Note that any time you get an error that a package is not found, try installing that package.**

The movielens dataset in the **dslabs** package includes data on a variety of movies and their rating by a  
particular user. Load the movielens dataset:

library(dslabs)  
data(movielens)

Begin your exploration of this dataset by looking at the structure of the dataset and variable types.

### **Question 3a**

1/1 point (graded)

How many rows are in the dataset?  correct

100004 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3b**

1/1 point (graded)

How many different variables are in the dataset?  correct

7 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3c**

1/1 point (graded)

What is the variable type of title ?

It is a text (txt) variable

It is a chronological (chr) variable

It is a string (str) variable

It is a numeric (num) variable

It is an integer (int) variable

It is a factor (Factor) variable

It is a character (chr) variable

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3d**

1/1 point (graded)

What is the variable type of genres ?

It is a text (txt) variable

It is a chronological (chr) variable

It is a string (str) variable

It is a numeric (num) variable

It is an integer (int) variable

It is a factor (Factor) variable

It is a character (chr) variable

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 4**

1/1 point (graded)

We already know we can use the levels() function to determine the levels of a factor. A different function, nlevels(), may be used to determine the number of levels of a factor.

Use this function to determine how many levels are in the factor genres in the movielens data frame.  correct

901 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

# Section 2:Vectors and Sorting

## Vectors

### Key Points

* The function c(), which stands for concatenate, is useful for creating vectors.
* Another useful function for creating vectors is the seq() function, which generates sequences.
* Subsetting lets us access specific parts of a vector by using square brackets to access elements of a vector.

### Code

# We may create vectors of class numeric or character with the concatenate function

codes <- c(380, 124, 818)

country <- c("italy", "canada", "egypt")

# We can also name the elements of a numeric vector

# Note that the two lines of code below have the same result

codes <- c(italy = 380, canada = 124, egypt = 818)

codes <- c("italy" = 380, "canada" = 124, "egypt" = 818)

# We can also name the elements of a numeric vector using the names() function

codes <- c(380, 124, 818)

country <- c("italy","canada","egypt")

names(codes) <- country

# Using square brackets is useful for subsetting to access specific elements of a vector

codes[2]

codes[c(1,3)]

codes[1:2]

# If the entries of a vector are named, they may be accessed by referring to their name

codes["canada"]

codes[c("egypt","italy")]

## Vector Coercion

### Key Points

* In general, coercion is an attempt by R to be flexible with data types by guessing what was meant when an entry does not match the expected. For example, when defining x as

x <- c(1, "canada", 3)

R coerced the data into characters. It guessed that because you put a character string in the vector, you meant the 1 and 3 to actually be character strings, "1" and "3".

* The function as.character() turns numbers into characters.
* The function as.numeric() turns characters into numbers.
* In R, missing data is assigned the value NA.

## Sorting

### Key Points

* The function sort() sorts a vector in increasing order.
* The function order() produces the indices needed to obtain the sorted vector, e.g. a result of  2 3 1 5 4 means the sorted vector will be produced by listing the 2nd, 3rd, 1st, 5th, and then 4th item of the original vector.
* The function rank() gives us the ranks of the items in the original vector.
* The function max() returns the largest value, while which.max() returns the index of the largest value. The functions min() and which.min() work similarly for minimum values.

### Code

library(dslabs)

data(murders)

sort(murders$total)

x <- c(31, 4, 15, 92, 65)

x

sort(x) # puts elements in order

index <- order(x) # returns index that will put x in order

x[index] # rearranging by this index puts elements in order

order(x)

murders$state[1:10]

murders$abb[1:10]

index <- order(murders$total)

murders$abb[index] # order abbreviations by total murders

max(murders$total) # highest number of total murders

i\_max <- which.max(murders$total) # index with highest number of murders

murders$state[i\_max] # state name with highest number of total murders

x <- c(31, 4, 15, 92, 65)

x

rank(x) # returns ranks (smallest to largest)

## Vector Arithmetic

### Key Points

* In R, arithmetic operations on vectors occur element-wise.

### Code

# The name of the state with the maximum population is found by doing the following

murders$state[which.max(murders$population)]

# how to obtain the murder rate

murder\_rate <- murders$total / murders$population \* 100000

# ordering the states by murder rate, in decreasing order

murders$state[order(murder\_rate, decreasing=TRUE)]

## Section 2 Assessment

 Bookmark this page

### **Question 1**

4.0/4.0 points (graded)

Consider the vector x:

x <- c(2, 43, 27, 96, 18)

Match the following outputs to the function which produces that output. Options include sort(x), order(x), rank(x) and none of these.

1, 2, 3, 4, 5                                                 

correct

1, 5, 3, 2, 4                                                 

correct

1, 4, 3, 5, 2                                                 

correct

2, 18, 27, 43, 96                                                 

correct

Submit

You have used 2 of 3 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 2**

4/4 points (graded)

Continue working with the vector x defined in question 1.

Match the following functions to their output. Options include integers 1 through 5 and none of these.

min(x)                                                                       

correct

which.min(x)                                                                       

correct

max(x)                                                                       

correct

which.max(x)                                                                       

correct

Submit

You have used 1 of 3 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3**

3.0/3.0 points (graded)

Mandi, Amy, Nicole, and Olivia all ran different distances in different time intervals. Their distances (in  
miles) and times (in minutes) are as follows:

name <- c("Mandi", "Amy", "Nicole", "Olivia")  
distance <- c(0.8, 3.1, 2.8, 4.0)  
time <- c(10, 30, 40, 50)

Write a line of code to convert time to hours. Remember there are 60 minutes in an hour. Then write a line of code to calculate the speed of each runner in miles per hour. Speed is distance divided by time.

How many hours did Olivia run?

Report 3 significant digits.

  correct

0.833 Loading

What was Mandi's speed in miles per hour?  correct

4.8 Loading

Which runner had the fastest speed?                                                 

correct

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

# Section 3: Indexing, Data Wrangling and Plots

## Indexing

### Key Points

* We can use logicals to index vectors.
* Using the function sum()on a logical vector returns the number of entries that are true.
* The logical operator “&” makes two logicals true only when they are both true.

### Code

# defining murder rate as before

murder\_rate <- murders$total / murders$population \* 100000

# creating a logical vector that specifies if the murder rate in that state is less than or equal to 0.71

index <- murder\_rate <= 0.71

# determining which states have murder rates less than or equal to 0.71

murders$state[index]

# calculating how many states have a murder rate less than or equal to 0.71

sum(index)

# creating the two logical vectors representing our conditions

west <- murders$region == "West"

safe <- murder\_rate <= 1

# defining an index and identifying states with both conditions true

index <- safe & west

murders$state[index]

## Indexing Functions

### Key Points

* The function which() gives us the entries of a logical vector that are true.
* The function match() looks for entries in a vector and returns the index needed to access them.
* We use the function %in% if we want to know whether or not each element of a first vector is in a second vector.

### Code

x <- c(FALSE, TRUE, FALSE, TRUE, TRUE, FALSE)

which(x) # returns indices that are TRUE

# to determine the murder rate in Massachusetts we may do the following

index <- which(murders$state == "Massachusetts")

index

murder\_rate[index]

# to obtain the indices and subsequent murder rates of New York, Florida, Texas, we do:

index <- match(c("New York", "Florida", "Texas"), murders$state)

index

murders$state[index]

murder\_rate[index]

x <- c("a", "b", "c", "d", "e")

y <- c("a", "d", "f")

y %in% x

# to see if Boston, Dakota, and Washington are states

c("Boston", "Dakota", "Washington") %in% murders$state

## Basic Data Wrangling

### Key Points

* To change a data table by adding a new column, or changing an existing one, we use the mutate() function.
* To filter the data by subsetting rows, we use the function filter().
* To subset the data by selecting specific columns, we use the select() function.
* We can perform a series of operations by sending the results of one function to another function using the pipe operator, %>%.

### Code

# installing and loading the dplyr package

install.packages("dplyr")

library(dplyr)

# adding a column with mutate

library(dslabs)

data("murders")

murders <- mutate(murders, rate = total / population \* 100000)

# subsetting with filter

filter(murders, rate <= 0.71)

# selecting columns with select

new\_table <- select(murders, state, region, rate)

# using the pipe

murders %>% select(state, region, rate) %>% filter(rate <= 0.71)

## Creating Data Frames

### Key Points

* We can use the data.frame() function to create data frames.
* By default, the data.frame() function turns characters into factors.  To avoid this, we utilize the stringsAsFactors argument and set it equal to false.

### Code

# creating a data frame with stringAsFactors = FALSE

grades <- data.frame(names = c("John", "Juan", "Jean", "Yao"),

exam\_1 = c(95, 80, 90, 85),

exam\_2 = c(90, 85, 85, 90),

stringsAsFactors = FALSE)

## Basic Plots

### Key Points

* We can create a simple scatterplot using the function plot().
* Histograms are graphical summaries that give you a general overview of the types of values you have.  In R, they can be produced using the hist() function.
* Boxplots provide a more compact summary of a distribution than a histogram and are more useful for comparing distributions. They can be produced using the boxplot() function.

### Code

# a simple scatterplot of total murders versus population

x <- murders$population /10^6

y <- murders$total

plot(x, y)

# a histogram of murder rates

hist(murders$rate)

# boxplots of murder rates by region

boxplot(rate~region, data = murders)

## Section 3 Assessment

 Bookmark this page

For questions 1-8, load the **dslabs** dataset heights:

library(dslabs)

data(heights)

options(digits = 3) # report 3 significant digits for all answers

### **Question 1**

1/1 point (graded)

First, determine the average height in this dataset. Then create a logical vector ind with the indices for those individuals who are above average height.

How many individuals in the dataset are above average height?  correct

532 Loading

Submit

You have used 2 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 2**

1/1 point (graded)

How many individuals in the dataset are above average height and are female?  correct

31 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3**

1/1 point (graded)

If you use mean() on a logical (TRUE/FALSE) vector, it returns the proportion of observations that are TRUE.

What proportion of individuals in the dataset are female?

Report 3 significant digits.

  correct

0.227 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### Question 4

This question takes you through three steps to determine the sex of the individual with the minimum height.

### **Question 4a**

1/1 point (graded)

Determine the minimum height in the heights dataset.  correct

50 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 4b**

1/1 point (graded)

Use the match() function to determine the index of the first individual with the minimum height.  correct

1032 Loading

Submit

You have used 2 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 4c**

1/1 point (graded)

Subset the sex column of the dataset by the index in 4b to determine the individual’s sex.                           

correct

Submit

You have used 1 of 1 attemptSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Show Answer

### Question 5

This question takes you through three steps to determine how many of the integer height values between the minimum and maximum heights are not actual heights of individuals in the heights dataset.

### **Question 5a**

1/1 point (graded)

Determine the maximum height.

Report 3 significant digits.

  correct

82.7 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 5b**

1/1 point (graded)

Which integer values are between the maximum and minimum heights? For example, if the minimum height is 10.2 and the maximum height is 20.8, your answer should be x <- 11:20 to capture the integers in between those values. (If either the maximum or minimum height are integers, include those values too.)

Write code to create a vector x that includes the integers between the minimum and maximum heights.

There are multiple ways to solve this problem, but the grader expects you to use the format in the problem description.

  correct

Submit

You have used 2 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 5c**

1/1 point (graded)

How many of the integers in x are NOT heights in the dataset?

Use the sum() and %in% functions in addition to the ! operator.

  correct

3 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### Question 6

Using the heights dataset, create a new column of heights in centimeters named ht\_cm. Recall that 1 inch =  
2.54 centimeters. Save the resulting dataset as heights2.

### **Question 6a**

1/1 point (graded)

What is the height in centimeters of the 18th individual (index 18)?  correct

163 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 6b**

1/1 point (graded)

What is the mean height in centimeters?  correct

174 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

Create a data frame females by filtering the heights2 data to contain only female individuals.

### **Question 7a**

1/1 point (graded)

How many females are in the heights2 dataset?  correct

238 Loading

Submit

You have used 1 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 7b**

1/1 point (graded)

What is the mean height of the females in centimeters?  correct

165 Loading

Submit

You have used 2 of 10 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 8**

1/1 point (graded)

The olive dataset in **dslabs** contains composition in percentage of eight fatty acids found in the lipid fraction of 572 Italian olive oils:

library(dslabs)

data(olive)

head(olive)

Plot the percent palmitic acid versus palmitoleic acid in a scatterplot. What relationship do you see?

There is no relationship between palmitic and palmitoleic.

There is a positive linear relationship between palmitic and palmitoleic.

There is a negative linear relationship between palmitic and palmitoleic.

There is a positive exponential relationship between palmitic and palmitoleic.

There is a negative exponential relationship between palmitic and palmitoleic.

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 9**

1/1 point (graded)

Create a histogram of the percentage of eicosenoic acid in olive.

Which of the following is true?

The most common value of eicosenoic acid is below 0.05%.

The most common value of eicosenoic acid is greater than 0.5%.

The most common value of eicosenoic acid is around 0.3%.

There are equal numbers of olive oils with eicosenoic acid below 0.05% and greater than 0.5%.

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 10**

2/2 points (graded)

Make a boxplot of palmitic acid percentage in olive with separate distributions for each region.

Which region has the highest median palmitic acid percentage?                                      

correct

Which region has the most variable palmitic acid percentage?                                      

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

# Section 4: Programming Basics

## Basic Conditionals

### Key Points

* The most common conditional expression in programming is an if-else statement, which has the form "if [condition], perform [expression], else perform [alternative expression]".
* The ifelse() function works similarly to an if-else statement, but it is particularly useful since it works on vectors by examining each element of the vector and returning a corresponding answer accordingly.
* The any() function takes a vector of logicals and returns true if any of the entries are true.
* The all() function takes a vector of logicals and returns true if all of the entries are true.

### Code

# an example showing the general structure of an if-else statement

a <- 0

if(a!=0){

print(1/a)

} else{

print("No reciprocal for 0.")

}

# an example that tells us which states, if any, have a murder rate less than 0.5

library(dslabs)

data(murders)

murder\_rate <- murders$total / murders$population\*100000

ind <- which.min(murder\_rate)

if(murder\_rate[ind] < 0.5){

print(murders$state[ind])

} else{

print("No state has murder rate that low")

}

# changing the condition to < 0.25 changes the result

if(murder\_rate[ind] < 0.25){

print(murders$state[ind])

} else{

print("No state has a murder rate that low.")

}

# the ifelse() function works similarly to an if-else conditional

a <- 0

ifelse(a > 0, 1/a, NA)

# the ifelse() function is particularly useful on vectors

a <- c(0,1,2,-4,5)

result <- ifelse(a > 0, 1/a, NA)

# the ifelse() function is also helpful for replacing missing values

data(na\_example)

no\_nas <- ifelse(is.na(na\_example), 0, na\_example)

sum(is.na(no\_nas))

# the any() and all() functions evaluate logical vectors

z <- c(TRUE, TRUE, FALSE)

any(z)

all(z)

## Functions

### Key points

* The R function called function() tells R you are about to define a new function.
* Functions are objects, so must be assigned a variable name with the arrow operator.
* The general way to define functions is: (1) decide the function name, which will be an object, (2) type function() with your function's arguments in parentheses, (3) write all the operations inside brackets.
* Variables defined inside a function are not saved in the workspace.

### Code

# example of defining a function to compute the average of a vector x

avg <- function(x){

s <- sum(x)

n <- length(x)

s/n

}

# we see that the above function and the pre-built R mean() function are identical

x <- 1:100

identical(mean(x), avg(x))

# variables inside a function are not defined in the workspace

s <- 3

avg(1:10)

s

# the general form of a function

my\_function <- function(VARIABLE\_NAME){

perform operations on VARIABLE\_NAME and calculate VALUE

VALUE

}

# functions can have multiple arguments as well as default values

avg <- function(x, arithmetic = TRUE){

n <- length(x)

ifelse(arithmetic, sum(x)/n, prod(x)^(1/n))

}

## For Loops

### Key points

* For-loops perform the same task over and over while changing the variable.  They let us define the range that our variable takes, and then changes the value with each loop and evaluates the expression every time inside the loop.
* The general form of a for-loop is: "For i in [some range], do operations".  This i changes across the range of values and the operations assume i is a value you're interested in computing on.
* At the end of the loop, the value of i is the last value of the range.

### Code

# creating a function that computes the sum of integers 1 through n

compute\_s\_n <- function(n){

x <- 1:n

sum(x)

}

# a very simple for-loop

for(i in 1:5){

print(i)

}

# a for-loop for our summation

m <- 25

s\_n <- vector(length = m) # create an empty vector

for(n in 1:m){

s\_n[n] <- compute\_s\_n(n)

}

# creating a plot for our summation function

n <- 1:m

plot(n, s\_n)

# a table of values comparing our function to the summation formula

head(data.frame(s\_n = s\_n, formula = n\*(n+1)/2))

# overlaying our function with the summation formula

plot(n, s\_n)

lines(n, n\*(n+1)/2)

## Other Functions

* Apply
* Tapply
* Sapply
* Mapply
* Split
* Cut
* Quantile
* Reduce
* Identical
* Uniquexc