Understand basic programming concepts

- **Video:** Programming using RStudio
  2 min
- **Video:** Programming fundamentals
  8 min
- **Reading:** Vectors and lists in R
  10 min
- **Reading:** Dates and times in R
  10 min
- **Reading:** Other common data structures
  10 min
- **Practice Quiz:** Test your knowledge on programming concepts
  4 questions

Explore coding in R

Learning about R packages

Explore the tidyverse

Weekly challenge 2

Other common data structures

In this reading, you will continue on the topic of data structures with an introduction to data frames and matrices. You will learn about the basic properties of each structure, and simple ways to make use of them using R code. You will also briefly explore **files**, which are often used to access and store data and related information.

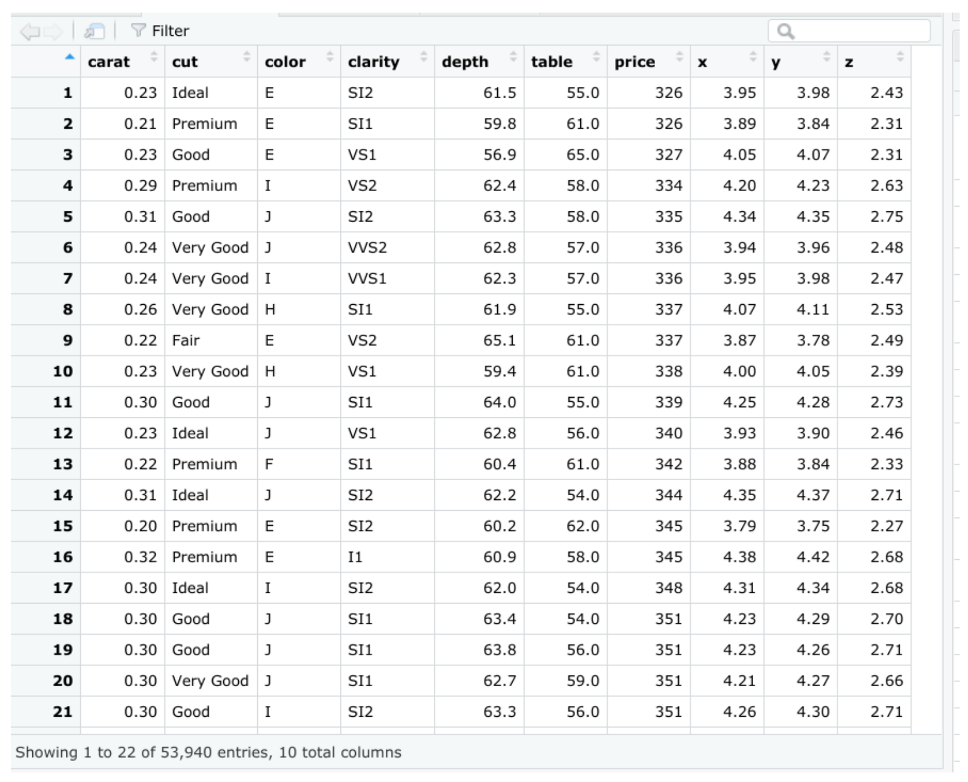
Data structures

Recall that a data structure is like a house that contains your data.

Data frames

Data frames are the most common way of storing and analyzing data in R, so it's important to understand what they are and how to create them. A **data frame** is a collection of columns–similar to a spreadsheet or SQL table. Each column has a name at the top that represents a variable, and includes one observation per row. Data frames help summarize data and organize it into a format that is easy to read and use.

For example, the data frame below shows the "diamonds" dataset, which is one of the preloaded datasets in R. Each column contains a single variable that is related to diamonds: carat, cut, color, clarity, depth, and so on. Each row represents a single observation.



	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
11	0.30	Good	J	SI1	64.0	55.0	339	4.25	4.28	2.73
12	0.23	Ideal	J	VS1	62.8	56.0	340	3.93	3.90	2.46
13	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33
14	0.31	Ideal	J	SI2	62.2	54.0	344	4.35	4.37	2.71
15	0.20	Premium	E	SI2	60.2	62.0	345	3.79	3.75	2.27
16	0.32	Premium	E	I1	60.9	58.0	345	4.38	4.42	2.68
17	0.30	Ideal	I	SI2	62.0	54.0	348	4.31	4.34	2.68
18	0.30	Good	J	SI1	63.4	54.0	351	4.23	4.29	2.70
19	0.30	Good	J	SI1	63.8	56.0	351	4.23	4.26	2.71
20	0.30	Very Good	J	SI1	62.7	59.0	351	4.21	4.27	2.66
21	0.30	Good	I	SI2	63.3	56.0	351	4.26	4.30	2.71

Showing 1 to 22 of 53,940 entries, 10 total columns

There are a few key things to keep in mind when you are working with data frames:

- First, columns should be named.
- Second, data frames can include many different types of data, like numeric, logical, or character.
- Finally, elements in the same column should be of the same type.

You will learn more about data frames later on in the program, but this is a great starting point.

If you need to manually create a data frame in R, you can use the **data.frame()** function. The data.frame() function takes vectors as input. In the parentheses, enter the name of the column, followed by an equals sign, and then the vector you want to input for that column. In this example, the *x* column is a vector with elements 1, 2, 3, and the *y* column is a vector with elements 1.5, 5.5, 7.5.

```
data.frame(x = c(1, 2, 3) , y = c(1.5, 5.5, 7.5))
```

If you run the function, R displays the data frame in ordered rows and columns.

```
   x y

1  1 1.5

2  2 5.5

3  3 7.5
```

In most cases, you won't need to manually create a data frame yourself, as you will typically import data from another source, such as a .csv file, a relational database, or a software program.

Files

Let's go over how to create, copy, and delete files in R. For more information on working with files in R, check out R documentation: files. **R documentation** is a tool that helps you easily find and browse the documentation of almost all R packages on CRAN. It's a useful reference guide for functions in R code. Let's go through a few of the most useful functions for working with files.

Use the **dir.create** function to create a new folder, or directory, to hold your files. Place the name of the folder in the parentheses of the function.

```
dir.create ("destination_folder")
```

Use the **file.create()** function to create a blank file. Place the name and the type of the file in the parentheses of the function. Your file types will usually be something like .txt, .docx, or .csv.

```
file.create ("new_text_file.txt")

file.create ("new_word_file.docx")

file.create ("new_csv_file.csv")
```

If the file is successfully created when you run the function, R will return a value of `TRUE` (if not, R will return `FALSE`).

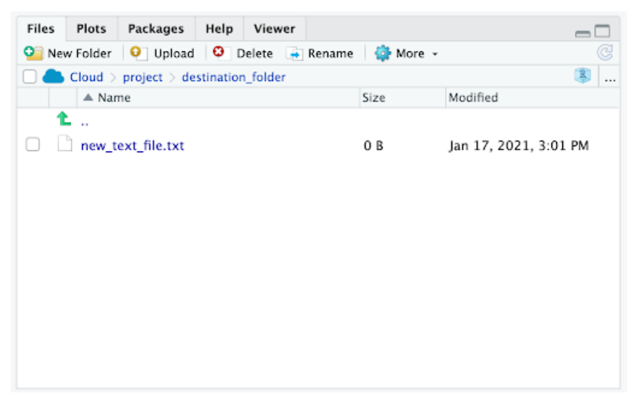
```
file.create ("new_csv_file.csv")

[1] TRUE
```

Copying a file can be done using the **file.copy()** function. In the parentheses, add the name of the file to be copied. Then, type a comma, and add the name of the destination folder that you want to copy the file to.

```
file.copy ("new_text_file.txt" , "destination_folder")
```

If you check the Files pane in RStudio, a copy of the file appears in the relevant folder:



Cloud > project > destination_folder

Name	Size	Modified
..		
new_text_file.txt	0 B	Jan 17, 2021, 3:01 PM

You can delete R files using the **unlink()** function. Enter the file's name in the parentheses of the function.

```
unlink ("some_.file.csv")
```

Additional resource

If you want to learn more about working with data frames, matrices, and arrays in R, check out the Data Wrangling section of Stat Education's Introduction to R course. The section includes modules on data frames, matrices, and arrays (and more), and each module contains helpful examples of key coding concepts.

Optional: Matrices

A **matrix** is a two-dimensional collection of data elements. This means it has both rows and columns. By contrast, a vector is a one-dimensional sequence of data elements. But like vectors, matrices can only contain a single data type. For example, you can't have both logicals and numerics in a matrix.

To create a matrix in R, you can use the **matrix()** function. The matrix() function has two main arguments that you enter in the parentheses. First, add a vector. The vector contains the values you want to place in the matrix. Next, add at least one matrix dimension. You can choose to specify the number of rows or the number of columns by using the code `nrow =` or `ncol =`.

For example, imagine you want to create a 2x3 (two rows by three columns) matrix containing the values 3-8. First, enter a vector containing that series of numbers: `c(3:8)`. Then, enter a comma. Finally, enter `nrow = 2` to specify the number of rows.

```
matrix(c(3:8), nrow = 2)
```

If you run the function, R displays a matrix with three columns and two rows (typically referred to as a "2x3") that contain the numeric values 3, 4, 5, 6, 7, 8. R places the first value (3) of the vector in the uppermost row, and the leftmost column of the matrix, and continues the sequence from left to right.

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

[1,]    3    5    7

[2,]    4    6    8
```

You can also choose to specify the number of columns (`ncol =`) instead of the number of rows (`nrow =`).

```
matrix(c(3:8), ncol = 2)
```

When you run the function, R infers the number of rows automatically.

```
     [,1] [,2]

[1,]    3    6

[2,]    4    7

[3,]    5    8
```

Mark as completed

Like Dislike Report an issue