



Week 1: Intro to R & Programming

Sierra & Maria

8/29/2023

Record

Who we are!







Maria (she/her/hers)
2nd year, Social-Personality
RASCL & SIR Lab



Sierra (she/her/hers)
4th year, Social-Personality
RASCL & Okonofua (EDENS) Lab



Today's agenda

-  Welcome & introductions
-  Check-in + any questions about the program?
-  Structure and goals of the workshop
-  Why R?
-  Introduction to RStudio interface
-  Data types, vectors, and data frames

Why we offer QuACK!

Programming skills! Learning to code is an essential skill for research and many other jobs!

Data skills! There's a lot more to working with data than just statistics!

Hit the ground running! You will be using these skills in 205 in the spring AND you will probably be doing research before that!

Community building! Get to know your cohort and other grad students and learn from each other!

We want to help make learning R and programming feel accessible and fun!

Goals for QuACK

- Learn foundational R and programming skills so you can learn more on your own and in your lab
 - Don't just do, **understand**!
- Feel confident in your skills moving forward
- Build an inclusive, welcoming, and supportive community around programming and data skills for people at all skill levels and from all backgrounds

Goals for QuACK

- Learn foundational R and programming skills so you can learn more on your own and in your lab
 - Don't just do, **understand**!
- Feel confident in your skills moving forward
- Build an inclusive, welcoming, and supportive community around programming and data skills for people at all skill levels and from all backgrounds

Day 1



Goals for QuACK

- Learn foundational R and programming skills so you can learn more on your own and in your lab
 - Don't just do, **understand!**
- Feel confident in your skills moving forward
- Build an inclusive, welcoming, and supportive community around programming and data skills for people at all skill levels and from all backgrounds

Day 1



Day 100



And it is always a work in progress!
Be patient with yourself!

Why teach R? And why is it so great?!

R is built for working with and analyzing data frames. And that is exactly what we need!

Check out these sites for more information:

- [Why is R so great?](#)
- [Why R is like a relationship...](#)

From: [YaRrr! The Pirate's Guide to R](#)



Enrolling

Post-bacs should audit the course

Grad students may audit or take the course for credit. It is your choice.

- Course code: 14178
- Units: 2
- Grading: Pass/No Pass (everyone passes)

Structure of the workshop

We've structured the workshop to parallel the data analysis pipeline:

1. Intro to data types and data exploration
2. Data wrangling: prepare your data for analysis*
3. Data visualization: get to know your data
4. Foundational skills for analysis and future programming
5. Putting it together

* The step that takes the longest!

Class structure

1. Live coding demo
2. Individual practice
3. Group work on real data*
4. Discussion

**We will be using the same dataset throughout the course to get a feel for the full process of exploring data and prep for analysis.*

Before every class:

- Download the materials you need each week from our [course page](#).
- Review the answer key from the previous week and see if you have questions.
- Catch up on any missed work or practice problems from the previous week.

Important reminders and some motivation

- Now is the time to learn! There is no time like the present!
- No question is a dumb question. If you are wondering something, chances are, at least one other person is, too.
ASK!
- TELL US TO SLOW DOWN! Ask us to repeat or review! This workshop is for you!
- Don't just do, understand!



Keep in mind...



- People are coming in from different starting points.
- Some things we talk about may be easy for you, and others may be new.
- Be open to brushing up on things you already know and supporting your peers while they learn something new!
- We have tried to make this accessible and interesting for people at various different levels.

Let's dig into some data!

Course page: <https://ucb-psychology-quack.github.io/site/QuACK2023/QuACK2023>

Data: covid_attitudes.csv.

Questions to keep in mind

- *What are some things you notice about these data just by looking at the spreadsheet ?*
- *What are some questions we could ask with this data ?*

Data types

Think about the data we just looked at:

What kind of information do we want to represent?

- Numbers
- Words (also called “strings” or “characters”)
- Logical - True/False or 0/1 (also called “booleans”)
- Categorical

That’s all that data types are: the different kinds of information we want to be able to work with and handle.

Let's get started in R!

Let's take a look at the RStudio interface

4 major components:

- Script
- Console
- Global Environment
- Multi-functional box: help, directory, plots, etc.

Basic data types in all programming languages

Think about when you've worked with data: What kind of information do you think we want to store?

- Numbers
- Words
- True/False

That's all that data types are. They are the different kinds of information that we want to be able to handle.

Integer, Double, and Numeric Types

This data type takes care of storing numbers

Integers: 0, -1, 5

Numeric: 0.5, 4.5, -3.1415

This distinction is not super important in R, but it can be in other programming languages because the computer stores this information slightly differently.

Character or String

This data type takes care of storing letters and words

Character: "C", "e",

String: "hello", "world", "party!", "1"

Note the use of single and double quotation marks! These are important to indicate to the computer that the words are strings!

In R, all the examples above are type character. Some other programming languages distinguish between character and a string of characters, so it is good to know both words.

Boolean or Logical

This data type takes care of storing TRUE/FALSE. It can only have these two values.

We will learn more about why this type is important as we go.

This data type exists in all programming languages. It is foundational to how programming works!

Variables and Variable Assignment

Creating a variable names a piece of information so that it can be used again and referred to more easily! Just like in algebra.

E.g., `x <- 5` or `x <- "hello"`

We “assign” data to a name with the assignment operator, “<-” (hot key: alt + “-” on windows or opt + “-” on mac)

Let’s check it out in RStudio!

Vectors - The building block of R

Vectors are a 1D collection of items that are all the same type. They are all numbers, or all characters, or all booleans. But not a mix of data types.

Syntax to create a vector: `c()`

Examples:

- `c(1, 2, 3, 4)`
- `c("a", "b", "c", "hello")`
- `c(TRUE, FALSE, TRUE, TRUE)`

Creating a vector

There are many different ways to create a vector. Here are a few:

Syntax	Explanation	Example
<code>c()</code>	Standard syntax	<code>c(1, 2, 3, 4)</code>
<code>:</code> notation	<i>From : to</i>	<code>1:4</code> yields <code>c(1, 2, 3, 4)</code>
<code>rep()</code>	Create a vector repeating the input vector a certain number of times	<code>rep(1, 5)</code> or <code>rep(1:4, 2)</code>
<code>seq()</code>	Create a sequence from a value to a value, incrementing by a given number	<code>seq(0, 1, .1)</code>

Indexing a vector

Sometimes we want to access the information inside of a vector.

For example, what if we want the first element in the vector:
`s <- c("Hi", "my", "name", "is", "Elena")`?

Each element has an *index*, or a position number that locates it, starting at 1.

Index	1	2	3	4	5	
c("Hi"	"my"	"name"	"is"	"Elena")

Indexing a vector

To get the first element in the vector:

```
s <- c("Hi", "my", "name", "is", "Elena")
```

s[1]  This is called bracket notation! []

Index	1	2	3	4	5	
s <- c("Hi"	"my"	"name"	"is"	"Elena")

R is “Vectorized”

This means that R is built to work with vectors of data.

We won't go into too many details here, it isn't necessary to fully understand the implications of this.

But main takeaway is that working with data is easier and writing code to work with this data is more efficient because R was written to work well with vectors.

Data Frames - The bread and butter of R

A data frame is a 2D collection of vectors

Each column is a vector of the same length

Each columns can contain a different type of data

Number of rows is the length of each vector

Example data

The dimensions of a data frame are described as row x column.

subjid	age	gender	bilingual
1	10	m	0
2	15	m	1
3	12	f	0
4	11	other	1
5	13	f	0

This is a 5x4 data frame

What are the data types of each column?

subjid	age	gender	bilingual
1	10	m	0
2	15	m	1
3	12	f	0
4	11	other	1
5	13	f	0

Special R data type: Factor

A factor is another way of saying a categorical variable

Factors have a finite number of category options, known as *levels* in R

E.g., bilingual is a factor with two levels can only be 0 or 1

This data type is special to R because of its utility in dealing with data

Now let's make a data frame in R!

Indexing data frames

Remember that data frames are described as row x columns

To index a data frame:

`df[row, col]` Returns the element in the cell at the intersection of that row# and col#

`df[row,]` Returns that row

`df[, col]` Returns that column

Practice indexing

subjid	age	gender	bilingual
1	10	m	0
2	15	m	1
3	12	f	0
4	11	other	1
5	13	f	0

\$ notation

What if you want to access a whole column in a data frame? R makes this easy, too!

```
df$subjid
```

This returns a vector!

For sake of completeness: Lists and Matrices

Lists

- Like vectors but can have mixed data types (e.g., `list(1, "a", TRUE)`)
- If you take a single row of a data frame the type would be a list
- Indexing lists is slightly different than vectors, just search it!

Matrices

- Like a data frame but can only contain one type of data
- Indexing matrices is the same as for data frames