Introductory Module - R

GIS for Transport

Christine Okeyo
Development Impact (DIME)

April, 2024

Contents

About this module

About this module

- This module will review key R fundamentals that will be useful and be applied in the subsequent modules.

Format

- These are hands-on sessions. You are strongly encouraged to follow along in your computer what the presenter is doing
- The sessions include exercises. You will be given 3-4 minutes to solve each exercise before we show the solution

R programming basics

R Objects

Objects are the **building blocks of R programming.**

What is an object?

- An object is **something you can refer to later** in your code to get a value
- **You can put anything into an object** : scalars, strings, datasets, vectors, plots, functions, etc.
- Objects also have attributes that can be used to manipulate them

Object classes

Some essential object classes include;

- **Vectors:** an uni-dimensional object that **stores a sequence of values of the same class**
- **Data frames:** a combination of different vectors of the same length (the same as your dataset in Excel)
- **Lists:** a multidimensional object that can store several objects **of different classes and dimensions**

File paths

File paths

- What about working directories? We set directories every time we start a new script:

```
1  setwd("your/path")
```

- However, we recommend not using it unless it's absolutely necessary (never, if possible)

File paths

Instead, use an absolute directory

```
1 data_dir <- file.path("~", "Documents", "DIME - BRT", "GIS for Transport","gis-transport-training","DataWork'
2
3 print(data_dir)
```

```
[1] "~/Documents/DIME - BRT/GIS for Transport/gis-transport-training/DataWork/DataSets/Final"
```

File paths

Load the schools and country data.

- The school names data contains names of all the schools in Kenya
- The school coordinates data contains geographical information of the respective schools

```
1 data_dir <- file.path("~", "Documents", "DIME - BRT", "GIS for Transport","gis-transport-training","DataWork'
2
3 school_names <- read.csv(file.path(data_dir, "school_names.csv"))
4
5 school_coordinates <- read.csv(file.path(data_dir, "school_coordinates.csv"))
```

Installing and Loading Packages

Packages

Another important aspect to consider is R packages. Consider the following:

Packages

To install a package you can run the following command:

```r
# To install
install.packages("tidyverse")
install.packages("sf")
install.packages("purrr")
```

- R packages need to be loaded in each R session that will use them.
- That means that, for example, a function that comes from the `sf` package cannot be used if the package has not been installed and loaded first.

To load a package you can run the following command:

```r
# To load
library(dplyr) # To wrangle data
library(sf) # Package for manipulating gis data
library(purrr)
```

> *Notes*: Remember you should always load your packages before your start coding.

Functions

Quick intro to functions

- `head()`, `View()`, `filter()` and `read.csv()` are functions.
- Functions in R take **named arguments**
- Type `help(filter)` in the console to check the arguments of the `filter()` function

Quick intro to functions

- When we used `filter(school_names, name == "Consolata School")` we're implicitly telling R that the `x` argument is `school_names` and the `subset` argument is `name == "Consolata School"`
- For example, see below:
  - `filter(school_names, name == "Consolata School")`

Quick intro to functions

- Arguments are always enclosed in parentheses
- Usually the first argument is the object you want to use the function on, e.g. `filter(school_names, ...)`
- Functions usually return values that you can store in an object, print or use directly as an argument of another function.

We will explore these ideas in depth in a later session.

Data in R

Loading a dataset in R

Before we start wrangling our data, let's read our datasets. We can use the `read.csv` function if we want to load the csv files with the school names and coordinates

Exercise 1: Loading data in R

Use either of the functions mentioned above and load the two datasets from your working directory

Solution:

```
school_names <- read.csv(file.path(data_dir, "school_names.csv"))

school_coordinates <- read.csv(file.path(data_dir, "school_coordinates.csv"))
```

Exploring data

Exploring a data set

These are some useful functions from base R:

- `View()`: open the data set.
- `class()`: reports object type of type of data stored.
- `dim()`: reports the size of each one of an object's dimension.
- `names()`: returns the variable names of a dataset.
- `str()`: general information on an R object.
- `summary()`: summary information about the variables in a data frame.
- `head()`: shows the first few observations in the dataset.
- `tail()`: shows the last few observations in the dataset.

Some other useful functions from the tidyverse:

- `glimpse()`: get a glimpse of your data.

Load and show a dataset

We can just show our dataset using the name of the object; in this case, `school_names`.

```
1 school_names
```

	osm_id	name
1	30312225	Consolata School
2	674552830	<NA>
3	1399125354	Galitos restaurant
4	1764153756	Makini Schools
5	1867185524	Bohra Primary School
6	2061462027	<NA>
7	2061462030	<NA>
8	2061462031	<NA>
9	2061462033	<NA>
10	2468180197	<NA>
11	2468180199	<NA>
12	2468180201	<NA>
13	2468180203	<NA>
14	2468180205	<NA>
15	2468180207	<NA>
16	2468180209	<NA>
17	2468180211	<NA>
18	2468180213	<NA>
19	2468180215	<NA>
20	2468189973	<NA>
21	2468189975	<NA>

Glimpse your data

Use `glimpse()` to get information about your variables (e.g., type, row, columns,)

```
1  school_names %>%
2    glimpse() #<<
```

```
Rows: 3,549
Columns: 3
$ osm_id  <dbl> 30312225, 674552830, 1399125354, 1764153756, 1867185524, 20614…
$ name    <chr> "Consolata School", NA, "Galitos restaurant", "Makini Schools"…
$ amenity <chr> "school", NA, "school", "school", "school", NA, NA, NA, NA, NA…
```

Dimensions of your data

Let's see first how many columns and observations the dataset has:

- **Dimensions of your data** (Rows and Columns):

```
1 dim(school_names)
```

```
[1] 3549    3
```

- **The number of distinct values of a particular variable**:

The `$` sign is a subsetting operator. In R, we have three subsetting operators (`[[`, `[`, and `$`.). It is often used to access variables in a dataframe.

The `n_distinct` function allows us to count the number of unique values of a variable length of a vector. We included `na.rm = TRUE`, so we don't count missing values.

Dimensions of your data

Exercise 2: Identify distinct values of a variable in a dataset. Using the `n_distinct` function, can you tell how many unique values these variables in the `school_names` dataset have?

1. name
2. amenity

Solution:

```r
## R Code
n_distinct(school_names$name, na.rm = TRUE)
```

```
[1] 1406
```

```r
n_distinct(school_names$amenity, na.rm = TRUE)
```

```
[1] 1
```

Data Cleaning

Renaming variables

If we want to to rename our variables, we could use `rename()` function from `dplyr` package:

```
1  school_names <- school_names %>%
2    rename( #<<
3      var_newname = var_oldname #<<
4    ) #<<
```

Subsetting data

We can subset data using the `filter` function from **dplyr package** ### dplyr::filter

Let's try filtering all the rows where school name is "Baptist Primary School" from the `school_names` data.

```
  1   filter(school_names, name=="Baptist Primary School")
```

```
      osm_id                    name amenity
1  949597326 Baptist Primary School  school
2  949598174 Baptist Primary School  school
3 1446059238 Baptist Primary School  school
```

Piping

- Ever heard of piping? It's this: %>%
- Piping is a way of doing metaprogramming
- The actual meaning of the pipes is: *Pipes take the **output** of the function at the left and pass it as the **first argument** of the function at the right*
- The advantages of using piping is that it allows to have a cleaner division of successively applied functions in R code, **drastically improving code readability**

Piping

```
  # 1: Doing it the long way -----------------------------
school_names <- filter(school_names, name =="Baptist Primary School")
school_names <- select(school_names, name)

  # 2: With pipes ----------------------------------
school_names %>%
  filter(name=="Baptist Primary School") %>%
  select(name)
```

Just remember:

- `x %>% f()` is the same as `f(x)`
- `x %>% f() %>% g()` is the same as `g(f(x))`

Merging data sets

We can use the `left_join()` function to merge two dataframes. The function syntax is: `left_join(a_df, another_df, by = c("id_col1"))`.

> A left join takes all the values from the first table, and looks for matches in the second table. If it finds a match, it adds the data from the second table; if not, it adds missing values.

Merging data sets

Exercise 3: Join the dataframes: `school_names` and `school_coordinates`.

Solution:

```
1  schools_df <- school_names %>% #<<
2    left_join(school_coordinates,by = "osm_id")
```

Merging data sets

Exercise 4: Check if there is any schools with no gis information:

- Only use pipes %>%
- And `filter()`
- Do not assign it to an object.

Solution:

```
1  schools_df %>%
2    filter(is.na(latitude))
```

So we ended up with no NAs for the school coordinates

This is because all schools have their respective GIS details stored in the `school_coordinates` dataset.

Saving a dataset

Saving a dataset

- The dataset you have is the same data set we've been using for earlier sessions, so we can save it now.
- To save a dataset we can use the `write_csv` function from the `readr` package, or `write.csv` from base R.

The function takes the following syntax:

`write_csv(x, file, append = FALSE)`:

- `x:` the object (usually a data frame) you want to export to CSV
- `file:` the file path to where you want to save it, including the file name and the format (".csv")

Saving a dataset

Exercise 5: Saving the dataset as csv format.

- Use `write_csv()`

```
write_csv(
  schools_df, file.path(data_dir, "schools_final.csv")
  )
```

Saving a dataset

- The problem with CSVs is that they cannot differentiate between `strings` and `factors`
- They also don't save factor orders
- Data attributes (which are beyond the scope of this training, but also useful to document data sets) are also lost.

Saving a dataset

It can be saved and loaded using the following commands:

- `write_rds(object, file = "")`: Writes a single R object to a file.
- `read_rds(file)`: Load a single R object from a file.

Save the data set

```
write_rds(schools_df,
          file.path(data_dir, "schools_final.Rds"))
```

Thank you!

Appendix

Appendix - R and RStudio Installation

This training requires that you have R and RStudio installed in your computer:

Instructions

- To install R, visit (https://cran.r-project.org) and select a Comprehensive R Archive Network (CRAN) mirror close to you.
- To install RStudio, go to https://www.rstudio.com/. Note that you need to install R first.

Appendix - More on R

Here are some advantages of R:

- R is a free and open source software!
- It allows you to have several datasets open simultaneously.
- It can run complex Geographic Information System (GIS) analyses.
- You can use it for web scrapping.
- You can run machine learning algorithms with it.
- You can create complex Markdown documents. This presentation, for example, is entirely done in R.
- You can create interactive dashboards and online applications with the Shiny package.

Appendix - Syntax

Some of the highlights on R's syntax include:

- Parentheses to separate function names from its arguments.
- Commas to separate arguments.
- For comments we use the # sign.
- You can have line breaks inside function statements.
- In R, functions can be treated much like any other object. Therefore, they can be passed as arguments to other functions.

Appendix - RStudio interface

Script

Where you write your code.

Console

Where your results and messages will be displayed. But you can also type commands directly into the console.

Environment

What's in R's memory.

The 4th pane

Can display different things, including plots you create, packages loaded and help files.

Appendix - RStudio vs R GUI

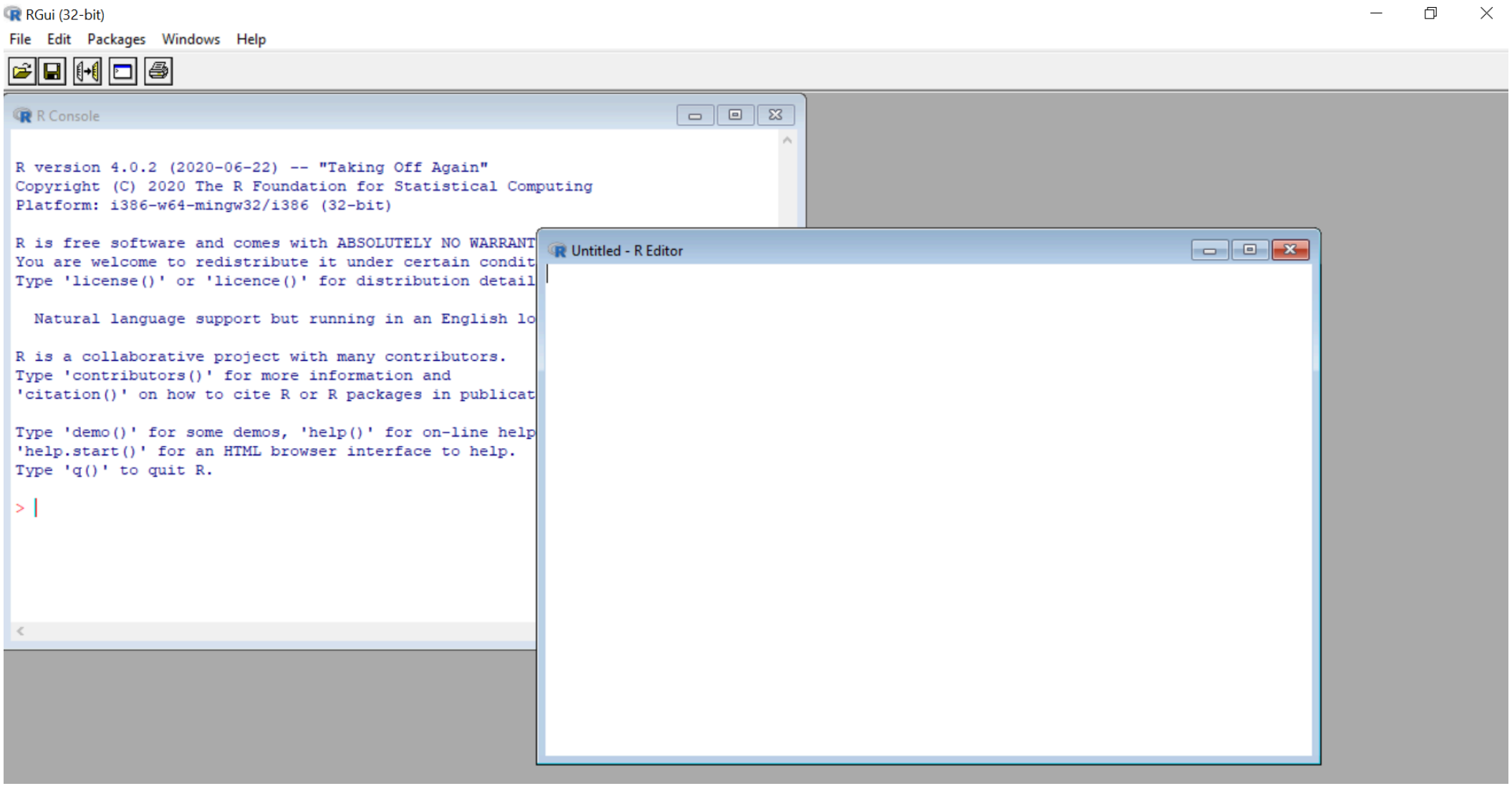
RStudio

- RStudio is an integrated development environment for R
- It's a software that uses the base R installation of your computer and provides an expanded interface that greatly facilitates R programming

R GUI

- The basic R Graphic User Interface (GUI) can also be used to program in R. You will find it in your computer with a name similar to `R<version>`, as in `R4.2.2`
- Opening the R GUI allows to work with R in a command line format, where you introduce one R command and the interface executes it and prints any message if needed
- It's very similar to the console panel of RStudio and it also allows to open a script editor, but it will not show you a list of the variables loaded on your environment

Appendix - RStudio vs R GUI



Appendix - Help, Google and Stack Overflow

R help files usually start with a brief description of the function, explain its syntax and arguments and list a few examples. There are two ways to access help files:

Exercise 6: Use help to get more information on the `summary` function

```
# You can use the help() function
help(summary)

# or its abbreviation
?summary
```

Appendix - Help, Google and Stack Overflow

- Some of the best things about R, is that **R has a wide user community** and it has **a lot of online resources**.
- The most powerful problem-solving tool in R, however, is Google. Searching the something yields tons of results.
- Often that means a Stack Overflow page where someone asked the same question and several people gave different answers. Here's a typical example: https://stackoverflow.com/questions/1660124/how-to-sum-a-variable-by-group

Appendix - Useful resources

Blogs, courses and resources:

- Surviving graduate econometrics with R: https://thetarzan.wordpress.com/2011/05/24/surviving-graduate-econometrics-with-r-the-basics-1-of-8/
- CRAN's manuals: https://cran.r-project.org/manuals.html
- R programming in Coursera: https://www.coursera.org/learn/r-programming
- R programming for dummies: http://www.dummies.com/programming/r/
- R bloggers: https://www.r-bloggers.com/
- R statistics blog: https://www.r-statistics.com/
- The R graph gallery: https://www.r-graph-gallery.com/
- R Econ visual library: (developed and maintained by DIME Analytics!) https://worldbank.github.io/r-econ-visual-library/

Appendix - Useful resources

Books:

- R for Stata Users - Robert A. Muenchen and Joseph Hilbe
- R Graphics Cookbook - Winston Chang https://r-graphics.org/
- R for Data Science - Hadley Wickham and Garrett Grolemund https://r4ds.had.co.nz/