

Getting Started with R and RStudio

Prof. Dr. Dennis Herhausen, KEDGE Business School

Getting Started with R and RStudio

R is a free and open source software environment for statistical computing and graphics

- Continuous improvement, maintained by top quality experts
- It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.

<https://www.r-project.org/>

R consists of a core and **packages** with functions that are not available in the core

https://cran.r-project.org/web/packages/available_packages_by_name.html

RStudio is an integrated development environment for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

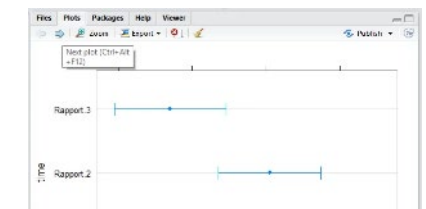
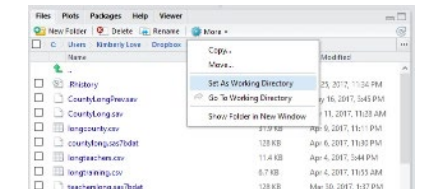
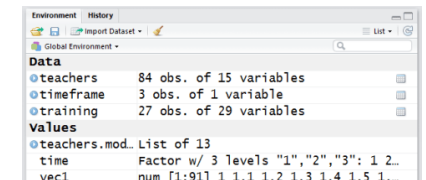
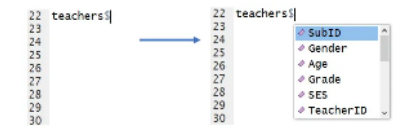
RStudio is available in free and open source editions and runs on the desktop (Windows, Mac, and Linux) or in a browser connected to RStudio Server or RStudio Server Pro.

<https://rstudio.com/>

Why R and RStudio?

Using RStudio for **data analysis and programming** in R provides many advantages:

- RStudio is designed to make it easy to write scripts
 - A text editor with features like color-coded syntax that helps you write clean scripts
 - Auto complete features that save time
- RStudio makes it convenient to view and interact with stored objects
 - An intuitive interface that lets you keep track of saved objects, scripts, and figures
- RStudio makes it easy to set your working directory and access files
 - Dedicated Project folders to keep everything in one place
 - Tools for creating documents containing a project's code, notes, and visuals
- RStudio makes graphics much more accessible for a casual user
- RStudio is **free and open source** (similar to R)



RStudio can also be used to program in other languages including **SQL, Python, Bash**, etc...

1. Install R

R is available to download from the official R website: <https://cran.r-project.org/>

Look for this section and select the most up-to-date version of R (new versions are released frequently):

The Comprehensive R Archive Network

Download and Install R
Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Windows

- Select the Download R for Windows option
- Select *base*, since this is your first installation of R on our computer
- Follow the standard instructions for installing programs for Windows. If you are asked to select *Customize Startup* or *Accept Default Startup Options*, choose the default options

MAC OS X

- Select the Download R for (Mac) OSX option
- Open the *.pkg* file and follow the standard instructions for installing applications on MAC OS X
- Drag and drop the R application into the Applications folder

2. Install RStudio

Now that R is installed, you can install RStudio. Navigate to the RStudio downloads page: <https://rstudio.com/products/rstudio/download/>

When you reach the RStudio downloads page, click the “Download” button of the *RStudio Desktop Open Source License Free* option:

	RStudio Desktop Open Source License Free	RStudio Desktop Commercial License \$995 /year	RStudio Server Open Source License Free	RStudio Server Pro Commercial License \$4,975 /year (5 Named Users)
	DOWNLOAD <small>Learn more</small>	BUY <small>Learn more</small>	DOWNLOAD <small>Learn more</small>	BUY <small>Evaluation Learn more</small>
Integrated Tools for R	✓	✓	✓	✓
Priority Support		✓		✓
Access via Web Browser			✓	✓
Enterprise Security				✓
Project Sharing				✓
Manage Multiple R Sessions & Versions				✓
Admin Dashboard				✓
Load Balancing				✓
Auditing and Monitoring				✓
Data Connectivity				✓
Launcher				✓
Tutorial API				✓
License	AGPL	Commercial	AGPL	Commercial

RStudio Desktop 1.3.1056 - [Release Notes](#)

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:



Requires macOS 10.13+ (64-bit)



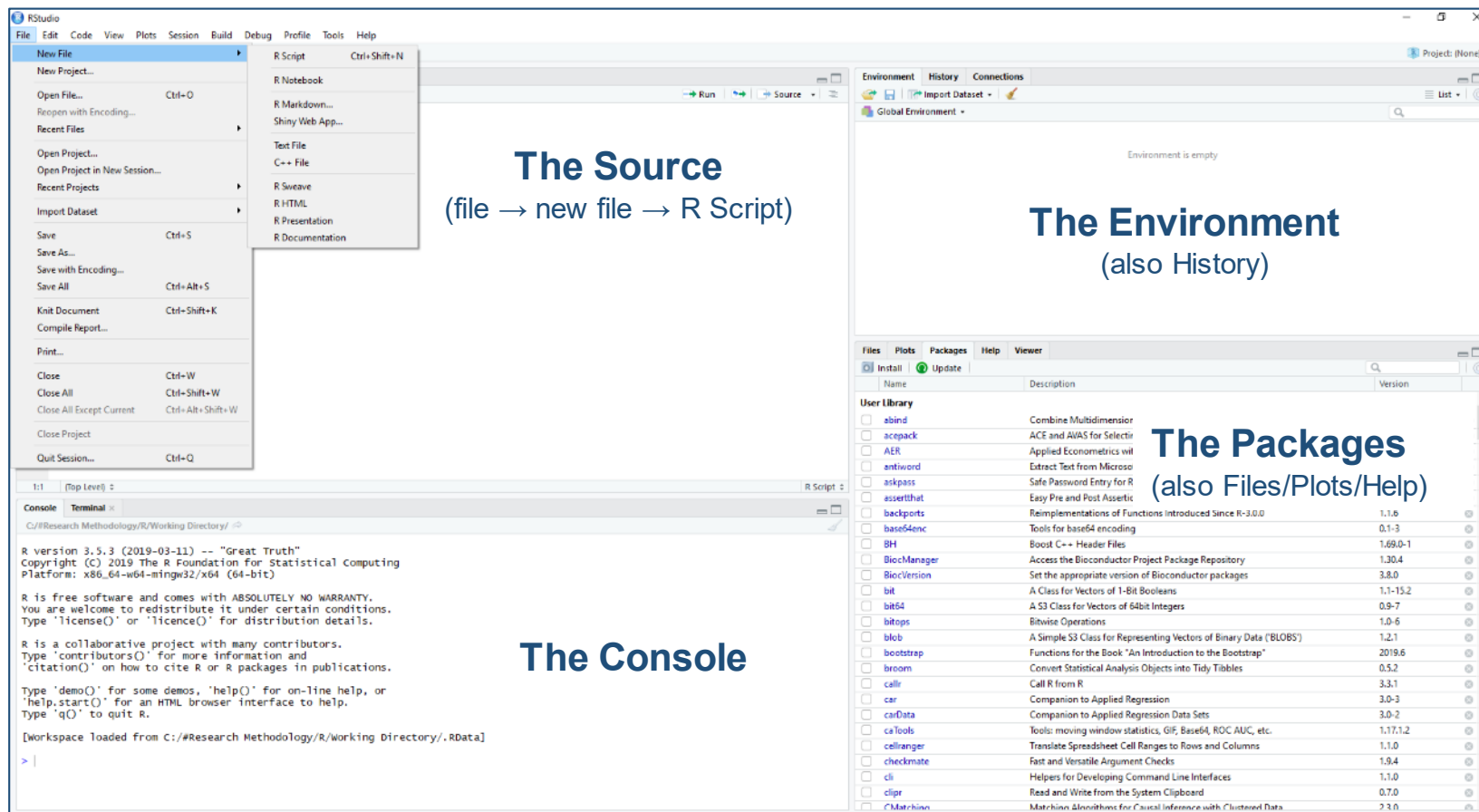
The operating system is usually **detected automatically** - you can directly download the correct version by clicking the “*Download Rstudio*” button.

If you want to download RStudio for another operating system navigate down to the “*All installers*” section of the page.

3. Launching RStudio

When you open RStudio, R is launched as well. A common mistake by new users is to open R instead of RStudio.

To open RStudio, search for RStudio on the desktop, and pin the RStudio icon to the preferred location (e.g. Desktop or toolbar)



4. Navigating in RStudio

The Source

The source pane is where you create and edit R Scripts” – your collections of code. R scripts are just text files with the “.R” extension. When you open RStudio, it will automatically start a new Untitled script. Before you start typing in an untitled R script, you should always save the file under a new file name. That way, if something on your computer crashes while you are working, RStudio will have your code waiting for you when you re-open RStudio, as it has recovered the code that you were editing.

The Console

The console is the heart of R. By default, It is present at the bottom left of the window. It is also called a command window. Here is where R actually evaluates code. At the beginning of the console you will see the character. This is a prompt that tells you that R is ready for new code. You can type code directly into the console after the prompt and get an immediate response.

For example, if you type 3+5 into the console and press enter, you will see that R immediately gives an output of 8.

The Environment

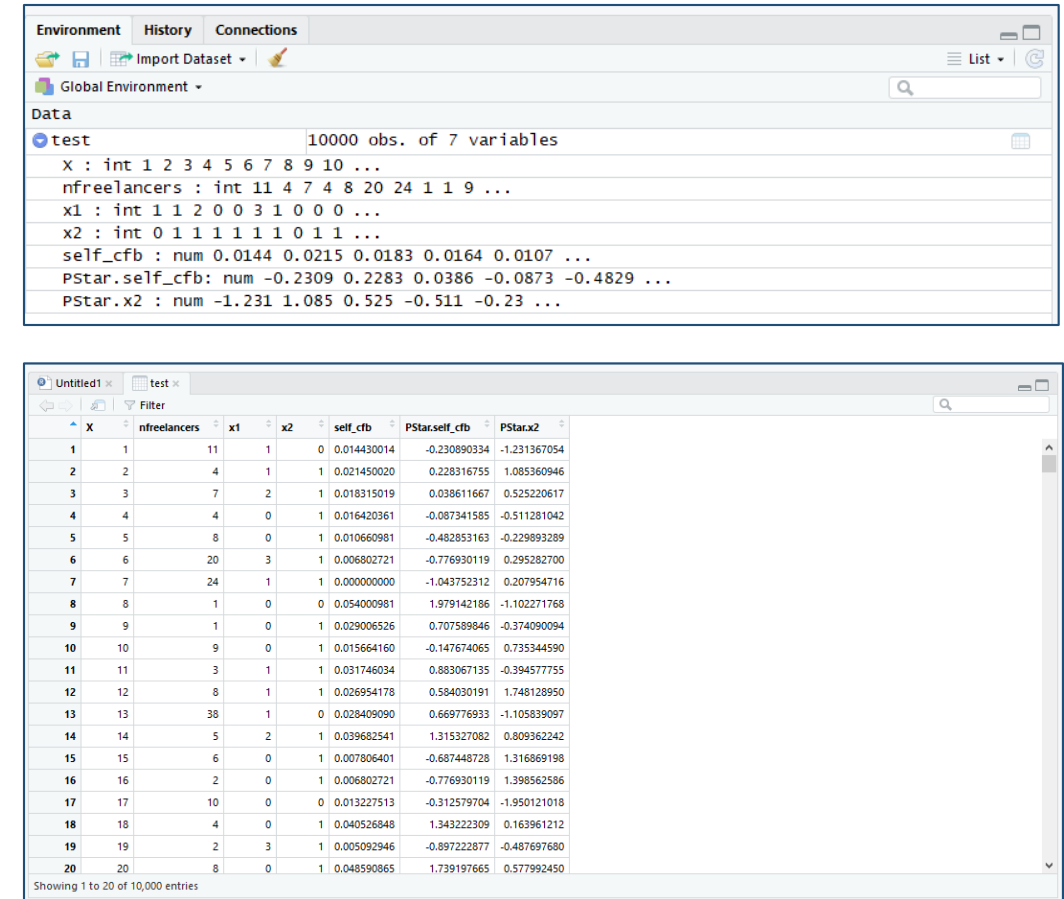
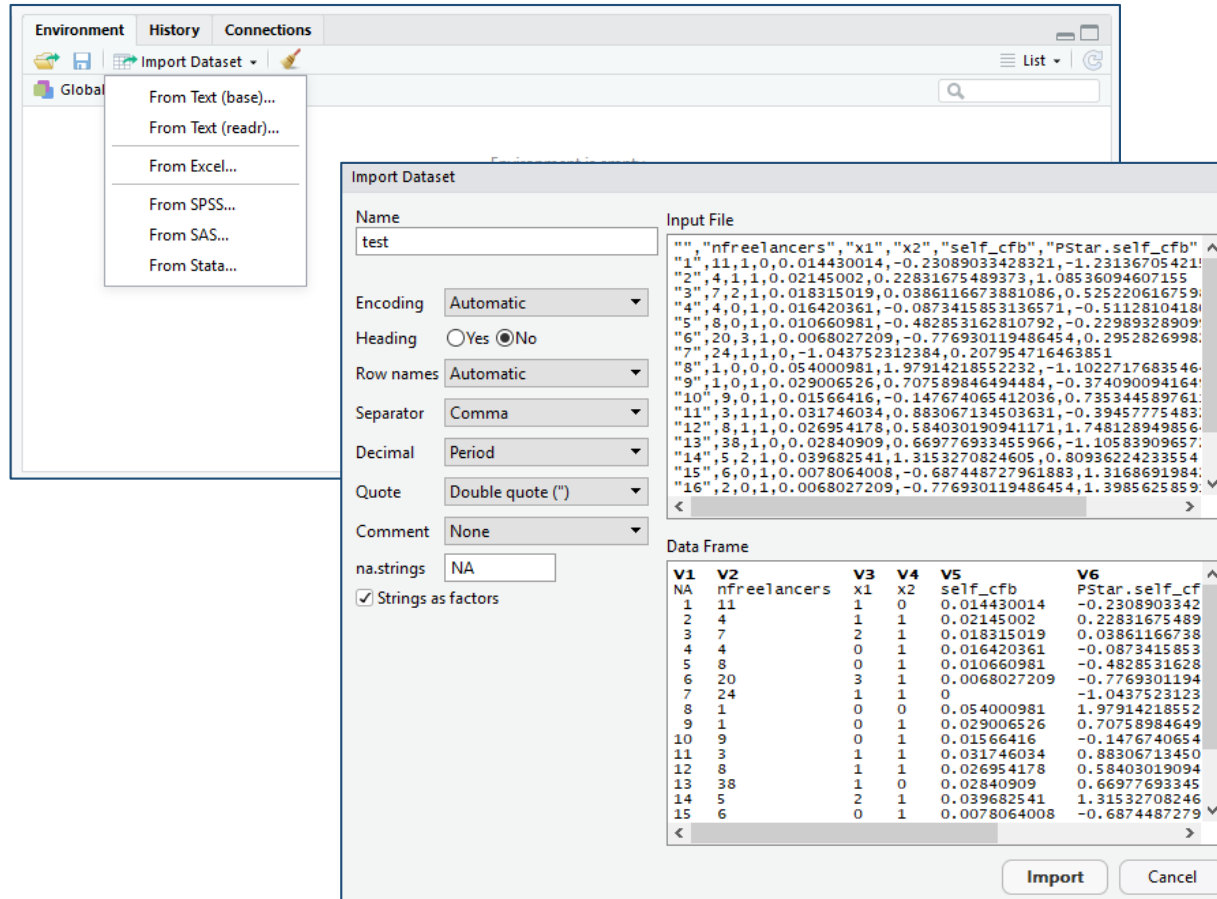
The Environment tab of this panel shows you the names of all the data objects (like a data frame) that you have defined in your current R session. You can also see information like the number of observations and rows in data objects.

The Packages

Shows a list of all the R packages installed on your hard drive and indicates whether or not they are currently loaded. Packages that are loaded in the current session are checked while those that are installed but not yet loaded are unchecked.

Packages can be installed by selecting “Install” and search for a package name

5. Getting Data into RStudio



This is the most basic approach to import a dataset into a **"Data Frame"**...

6. Install a Package



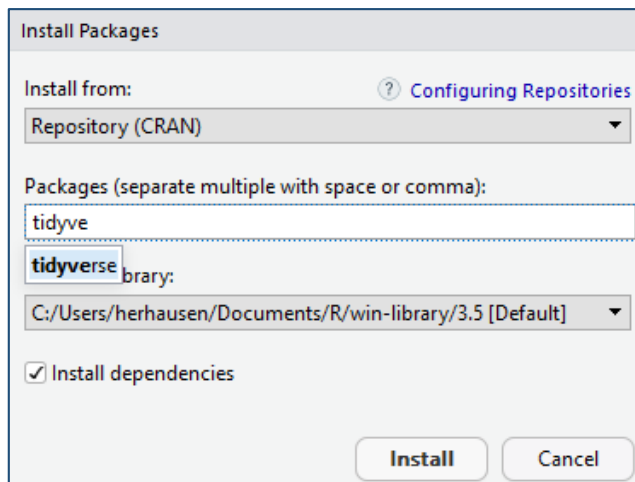
What is a package?

A package is a bundle of code that a generous person has written, tested, and then given away.

Most of the time packages are designed to solve a specific problem, so they pull together functions related to a particular data science problem (e.g., data wrangling, visualization, inference). Anyone can write a package, and you can get packages from lots of different places, but for beginners the best thing to do is get packages from **CRAN**, the **Comprehensive R Archive Network**.

It's easier than any of the alternatives, and people tend to wait until their package is stable before submitting it to CRAN, so you're less likely to run into problems. You can find a list of all the packages on CRAN [here](https://cran.r-project.org/).

Some packages are bundles of packages. For example, the **tidyverse** is an umbrella package that pulls together lots of individual data wrangling and visualization packages, so that when you install **tidyverse** you actually get eight packages (<https://www.tidyverse.org/>).



Installing packages is all well and good - but knowing what they do is pretty important when you want to use them! CRAN requires that package authors write documentation that goes with their package and these documents are designed to give you an idea of what functions are included and what the package can be used for.

Best ways to get started with a package:

- Look on CRAN (<https://cran.r-project.org/>)
- See if the author wrote a package vignette (*click on the package*)
- Simply google it – you will most probably find many “how to” guides ;)

7. Load a Package a.k.a. “Write your First Script”

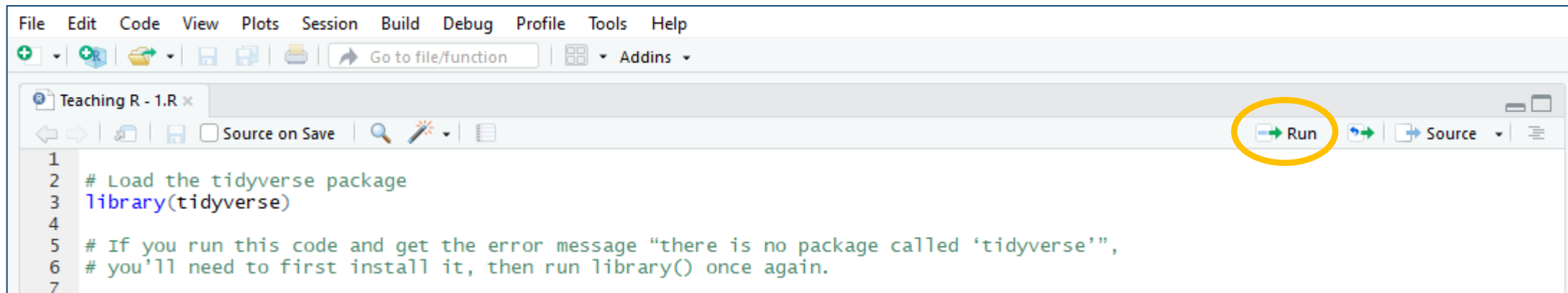
Install the **tidyverse** package (see previous page) and load it to use it.

You only need to install a package once, but **you need to reload it every time you start a new session!**

```
# Load the tidyverse package  
library(tidyverse)
```

```
# If you run this code and get the error message “there is no package called ‘tidyverse’”,  
# you’ll need to first install it, then run library() once again.
```

Select the text and “Run”



8. Start to Work with Data 😊

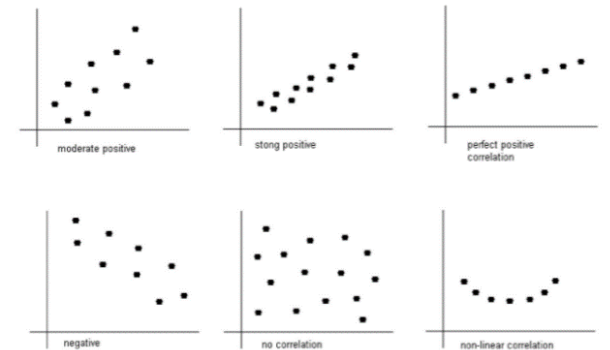


Do cars with big engines use more fuel than cars with small engines?

You probably already have an answer, but try to make your answer precise.

What does the relationship between engine size and fuel efficiency look like?

Is it positive? Negative? Linear? Nonlinear?



<https://r4ds.had.co.nz/data-visualisation.html>

Example: Data Visualization with *ggplot2*

“The simple graph has brought more information to the data analyst’s mind than any other device.” John Tukey

R has several systems for making graphs, but *ggplot2* is one of the most elegant and most versatile (and a core member of the *tidyverse*). If you have installed and loaded the *tidyverse* you can access the datasets, help pages, and functions that we will use in the following.

You can find an answer to the question of **fuel efficiency** with the *mpg* data frame found in *ggplot2*.

mpg contains observations collected by the US Environmental Protection Agency on 38 models of car, including:

- *displ*, a car’s engine size, in litres.
- *hwy*, a car’s fuel efficiency on the highway, in miles per gallon.

```
# Have a first look at the dataset
```

```
mpg
```

```
# Learn more about the dataset and open the “Help” tap
```

```
?mpg
```

```
# Load the dataset into your “Environment” tap
```

```
mpg <- mpg
```

A data frame is a rectangular collection of variables (in the columns) and observations (in the rows).

```
> mpg
# A tibble: 234 x 11
  manufacturer model      displ  year   cyl trans      drv   cty   hwy fl   class
  <chr>         <chr>    <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 audi         a4          1.8   1999     4 auto(l5) f      18    29 p    compact
2 audi         a4          1.8   1999     4 manual(m5) f      21    29 p    compact
3 audi         a4          2     2008     4 manual(m6) f      20    31 p    compact
4 audi         a4          2     2008     4 auto(av) f      21    30 p    compact
5 audi         a4          2.8   1999     6 auto(l5) f      16    26 p    compact
6 audi         a4          2.8   1999     6 manual(m5) f      18    26 p    compact
7 audi         a4          3.1   2008     6 auto(av) f      18    27 p    compact
8 audi         a4 quattro  1.8   1999     4 manual(m5) 4      18    26 p    compact
9 audi         a4 quattro  1.8   1999     4 auto(l5) 4      16    25 p    compact
10 audi         a4 quattro  2     2008     4 manual(m6) 4      20    28 p    compact
# ... with 224 more rows
> |
```

<https://r4ds.had.co.nz/data-visualisation.html>

Correlation Analysis and First Plot

Start with a simple correlation analysis (syntax for *cor* is `dataset_name$variable_name`)

```
# Correlate displ with hwy from the dataset mpg
cor(mpg$displ, mpg$hwy)
```

Next, create a plot of the negative relationship between *displ* and *hwy*

```
# Creating a first ggplot for displ and hwy from mpg in the "Plots" tab
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

The first argument of *ggplot()* is the dataset to use in the graph.

You complete your graph by adding one or more layers to *ggplot()*.

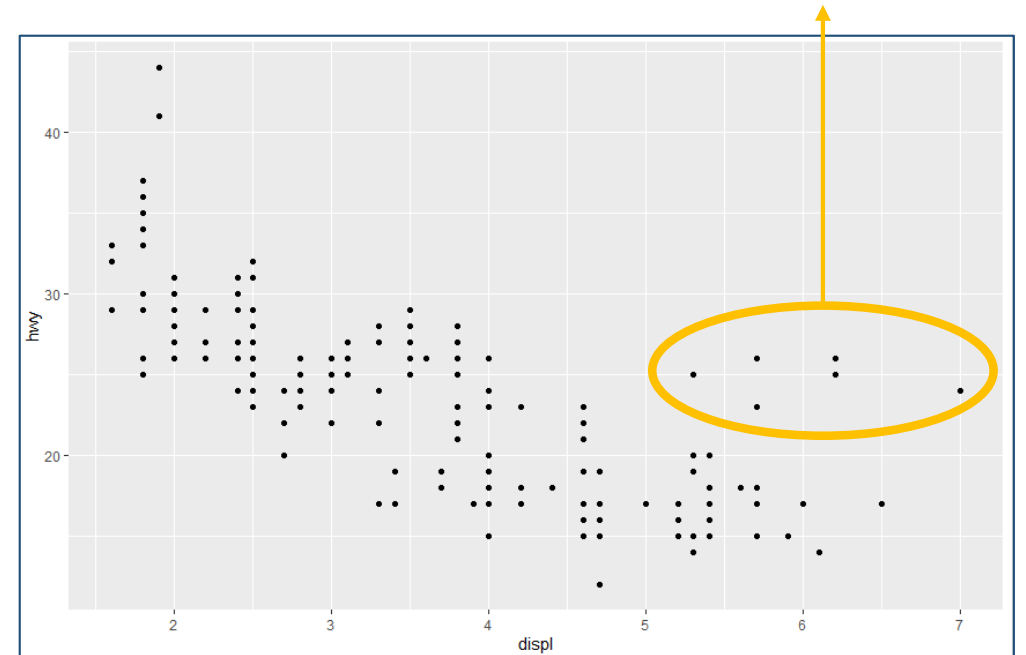
The function *geom_point()* adds a layer of points to your plot, which creates a scatterplot.

Each *geom* function in **ggplot2** takes a mapping argument. This defines how variables in your dataset are mapped to visual properties.

The mapping argument is always paired with *aes()*, and the *x* and *y* arguments of *aes()* specify which variables to map to the *x* and *y* axes.

ggplot2 looks for the mapped variables in the data argument, in this case, *mpg*.

Some data points do not fit to the trend...



Adding Complexity to the Analysis

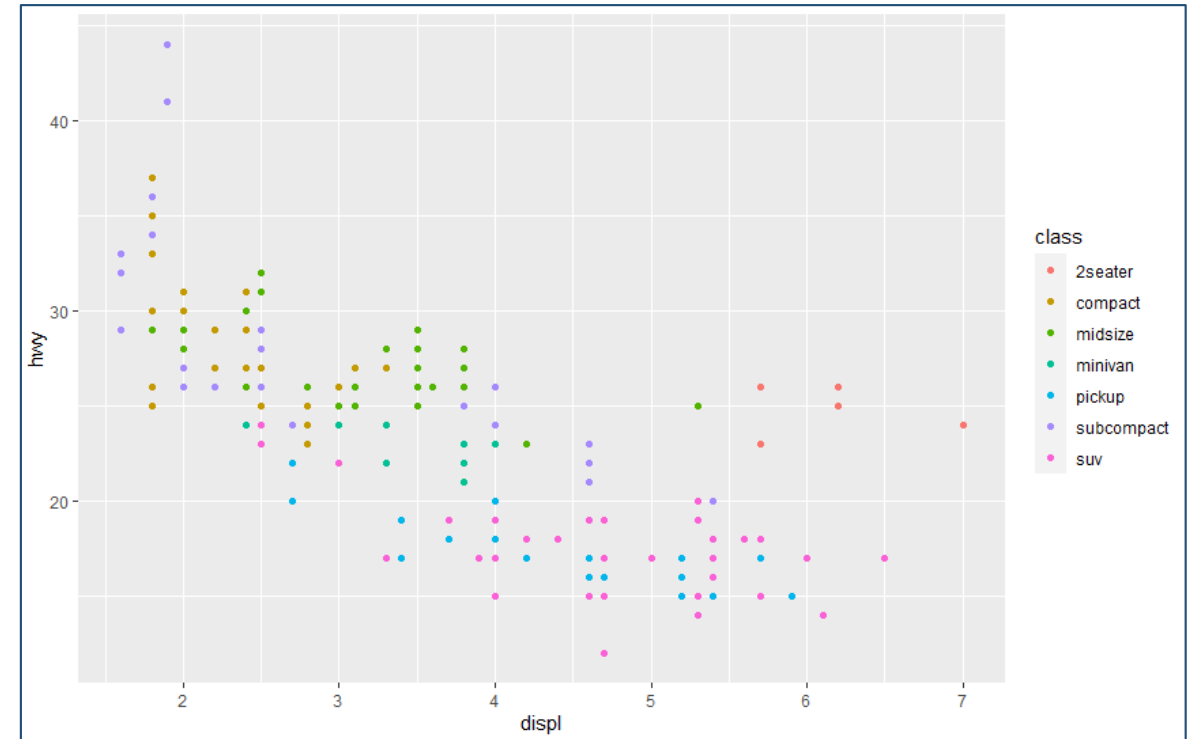
Some cars have a higher mileage than you might expect.

How can you explain these cars?

One way to test this hypothesis is to look at the *class* value for each car. The *class* variable of the *mpg* dataset classifies cars into groups such as compact, midsize, and SUV. Maybe the outlying points are classified as compact cars opposed to trucks or SUVs?

You can add a **third variable**, like *class*, to a two dimensional scatterplot by mapping it to an aesthetic. An aesthetic is a visual property of the objects in your plot. Aesthetics include things like the size, the shape, or the color of your points. You can display a point in different ways by changing the values of its aesthetic properties.

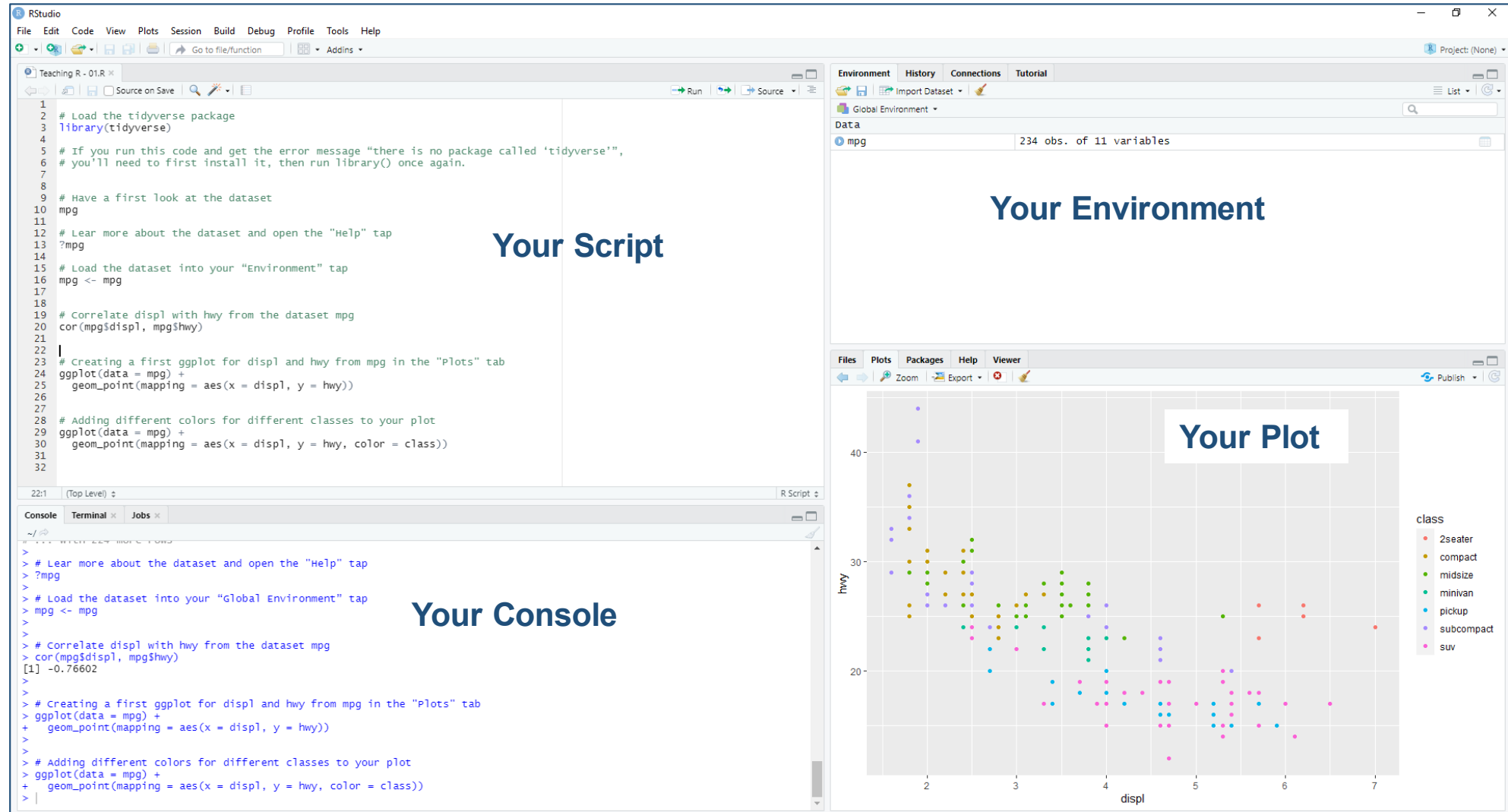
To map an aesthetic to a variable, associate the name of the aesthetic to the name of the variable inside *aes()*. **ggplot2** will automatically assign a unique level of the aesthetic (here a unique color) to each unique value and add a legend for explanation.



```
# Adding different colors for different classes to your plot
```

```
ggplot(data = mpg) +  
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

RStudio should look like this now



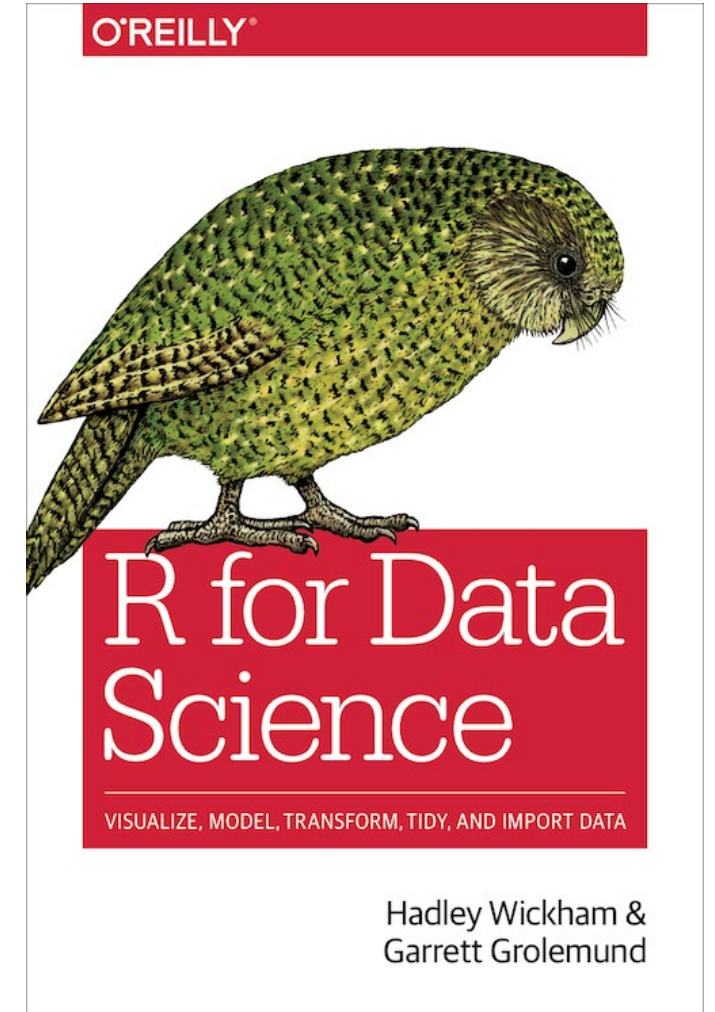
Now it is Time to Practice

<https://r4ds.had.co.nz/data-visualisation.html>


The example is part of the website for “**R for Data Science**”.

This website will teach you how to do data science with R: You will learn how to get your data into R, get it into the most useful structure, transform it, visualize it, and model it.

On the website, you will find a practicum of skills for data science. Just as a chemist learns how to clean test tubes and stock a lab, you will learn how to clean data and draw plots—and many other things besides. These are the skills that allow data science to happen, and here you will find the best practices for doing each of these things with R. You will learn how to use the grammar of graphics, literate programming, and reproducible research to save time. You’ll also learn how to manage cognitive resources to facilitate discoveries when wrangling, visualizing, and exploring data.



Remember the Basic R Logic

- 
1. Load the necessary packages
 2. Import the data
 3. Pre-process the data (if necessary)
 4. Run the analysis
 5. Save and export the results

Common Problems with R

As you start to write and run R code, you're likely to run into problems...

Don't worry — it happens to everyone 😊

Start by carefully comparing the code that you're running to the code in the examples.

- Try to **use and adapt the code from the examples!**
- R is **extremely picky**, and a misplaced character can make all the difference.
- Make sure that every (is matched with a) and every " is paired with another ".
- Sometimes you'll run the code and nothing happens. Check the left-hand of your console: if it's a +, it means that R doesn't think you've typed a complete expression and it's waiting for you to finish it.
- If you're still stuck, try the help. You can get **help about any R function** by running `?function_name` in the console. Don't worry if the help doesn't seem that helpful - instead skip down to the examples and look for code that matches what you're trying to do.
- If that doesn't help, carefully read the error message. Sometimes the answer will be buried there! But when you're new to R, the answer might be in the error message but you don't yet know how to understand it.
- Try **googling** the error message, as it's likely someone else has had the same problem, and has gotten help online.
- Use the **Troubleshooting Guide**: <https://support.rstudio.com/hc/en-us/articles/200488498-Troubleshooting-Guide-Using-RStudio>



Style Guide: Practice Good Habits in your Script

Good coding style is like using **correct punctuation**. You can manage without it, but it sure makes things easier to read.

As with styles of punctuation, there are many possible variations. But it is important to use a **consistent style**.

Good style is important because while your code only has one author, it'll usually have **multiple readers**.

This is especially true when you're writing code with others. In that case, it's a good idea to **agree on a common style up-front**.

A few suggestions (see <http://adv-r.had.co.nz/Style.html> for more details):

- File names for scripts should be meaningful and end in .R
- Strive to limit your code to 80 characters per line. This fits comfortably on a printed page with a reasonably sized font
- Comment your code: Each line of a comment should begin with the comment symbol and a single space: #
- Use commented lines of - and = to break up your file into easily readable chunks
- Variable and function names should be lowercase. Use an underscore (_) to separate words within a name
- Strive for names that are concise and meaningful (this is not easy!)
- Place spaces around all infix operators (=, +, -, <-, etc.)
- Always put a space after a comma, and never before (just like in regular English)

If you Want to Practice Yourself

Learn the basics

Visit Try R to learn how to write basic R code (<https://www.pluralsight.com/search?q=R>). These interactive lessons will get you writing real code in minutes, and they'll tell you immediately when you go wrong.

Broaden your skills

Work through The Beginner's Guide to R by Computerworld Magazine (<https://www.computerworld.com/article/2497143/business-intelligence-beginner-s-guide-to-r-introduction.html>). This 30 page guide will show you how to install R, load data, run analyses, make graphs, and more.

Look up help

When you need to learn more about an R function or package, visit <https://www.rdocumentation.org/>, a searchable database of R documentation. You can search for R packages and functions, look at package download statistics, and leave and read comments about R functions.

Ask questions

Seek help at StackOverflow, a searchable forum of questions and answers about computer programming. StackOverflow has answered (and archived) over 240,000 questions related to R programming (<https://stackoverflow.com/questions/tagged/r>).

If you have a question that is more about statistical methodology there are also plenty of R users active on the CrossValidated Q&A community (<https://stats.stackexchange.com/>).

You may ask for help from R and RStudio users on <https://community.rstudio.com/latest>.

Keep tabs on the R community

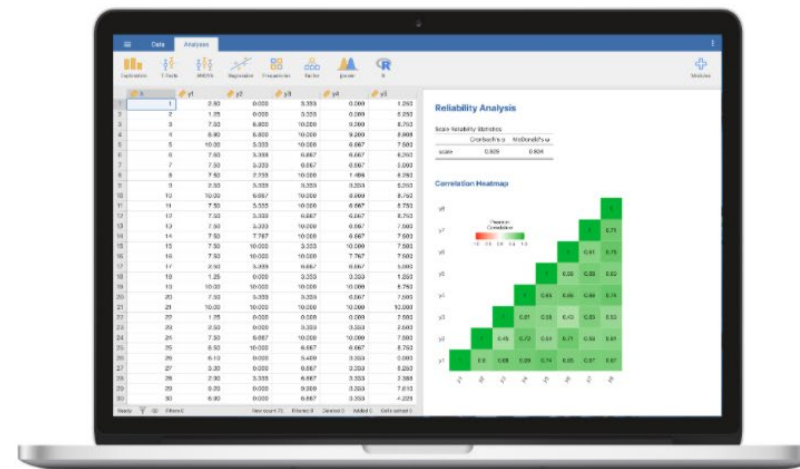
Read R bloggers, a blog aggregator that reposts R related articles from across the web (<https://www.r-bloggers.com/>). A good place to find R tutorials, announcements, and other random happenings.

Subscribe to R-Weekly (<https://rweekly.org/>), a collaboratively maintained community newsletter summarizing each week in R.

Do you Prefer to Click for Simple Analyses?



features download news about resources ▾ contribute



free and open statistical software to bridge the gap between researcher and statistician



STATS MADE SIMPLE

jamovi is a new "3rd generation" statistical spreadsheet. designed from the ground up to be easy to use, jamovi is a compelling alternative to costly statistical products such as SPSS and SAS.



R INTEGRATION

jamovi is built on top of the R statistical language, giving you access to the best the statistics community has to offer. would you like the R code for your analyses? jamovi can provide that too.



FREE AND OPEN

jamovi will always be free and open - that's one of our core values - because jamovi is made by the scientific community, for the scientific community.

Download from:
<https://www.jamovi.org/>

jamovi can operate some R code with an Interface

The screenshot displays the jamovi software interface. On the left is a data table with columns: No, manufact..., model, displ, and year. The main panel shows the 'Correlation Matrix' configuration. The 'Correlation Coefficients' section has 'Pearson' selected. The 'Additional Options' section has 'Report significance' selected. The 'Hypothesis' section has 'Correlated' selected. The 'Plot' section has 'Correlation matrix' selected. The 'References' section lists two sources: [1] The jamovi project (2020). jamovi. (Version 1.2) [Computer Software]. Retrieved from <https://www.jamovi.org>. [2] R Core Team (2019). R: A Language and environment for statistical computing. (Version 3.6) [Computer software]. Retrieved from <https://cran.r-project.org/>.

Correlation Matrix

		displ	hwy
displ	Pearson's r	—	—
	p-value	—	—
hwy	Pearson's r	-0.766	—
	p-value	< .001	—

The jamovi project is a free and open statistical platform which is intuitive to use.

But not all analyses are possible!

However it is a recent project and continuously growing in functionality.

Further Reading and More Details

R for Data Science

This book will teach you how to do data science with R: You'll learn how to get your data into R, get it into the most useful structure, transform it, visualise it and model it.

<https://r4ds.had.co.nz/>

Hands-On Programming with R

An introduction to programming in R. The book uses three hands-on projects to teach every aspect of R programming, from loading data to writing fast, vectorized functions.

<https://rstudio-education.github.io/hopr/index.html>

An Introduction to R

Most R novices will start with the introductory session in Appendix A. This should give some familiarity with the style of R sessions and more importantly some instant feedback on what actually happens.

<https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>

R pour les débutants

Introduction to R in French 😊

https://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf

Contact

Prof. Dr. Dennis Herhausen

Associate Professor of Marketing

KEDGE Business School

Domaine de Luminy

Rue Antoine Bourdelle

13009 Marseille, France

dennis.herhausen@kedgebs.com

To connect: **Linked** 