

Week1: Basics

Hakan Mehmetcik

Collective Statistical Illiteracy: Understanding the Challenge

In contemporary society, almost every news article or broadcast includes some form of scientific or numerical data. Headlines routinely mention statistics about public health, economic indicators, environmental changes, and more. However, a significant portion of the audience struggles to interpret these numbers and to understand how they were derived. This difficulty can have profound implications for personal decision-making, organizational strategy, and public policy.

The concept of **collective statistical illiteracy** highlights this widespread inability to grasp the meaning behind statistical facts and figures. When individuals lack the skills to critically evaluate quantitative information, they are more susceptible to misunderstanding important issues, making poorly informed decisions, or being swayed by misleading statistics.

Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write.

H.G. Wells (1903, paraphrased by S.S. Wilks, see [link](#))

Why Study Data Science?

Data science drives innovation and decision-making in virtually every industry—from health-care and finance to social media and marketing. By combining computational methods with statistical analysis, data science uncovers hidden patterns and insights that help organizations craft evidence-based strategies, predict future trends, and make more informed decisions. As the volume of data grows exponentially, the ability to collect, process, and interpret this information becomes ever more critical.

How Is Data Science Different from Math and Statistics?

- **Interdisciplinary Approach:** While mathematics and statistics focus on theoretical models and the science of uncertainty, data science integrates computer programming, domain knowledge, and data engineering alongside statistical techniques.
- **Practical, Data-Driven Focus:** Data scientists often work with real-world data that may be messy or unstructured. They prioritize building reproducible workflows and scalable analyses, which is not the central concern in pure math or traditional statistics.
- **Technology and Tools:** Data science heavily relies on programming languages (e.g., Python, R) and tools (e.g., machine learning frameworks, data visualization platforms) to handle large, complex datasets—capabilities not typically emphasized in standard math or statistics coursework.

Data science—the science of extracting meaningful information from data. As such, data science is a broader field than statistics, encompassing data gathering, preparation, modeling, visualization, computing, and meta-analysis of data science itself.

i Note

An assumption underlying these efforts is that data is the foundation of various information types that can eventually be turned into knowledge and wisdom. Figure below shows their arrangement in a hierarchical structure of a [DIKW pyramid](#):



The key distinction between the lower and the upper layers of the pyramid is that data needs to address some hypothesis or answer some question, and has to be interpreted and understood to become valuable. Here to say, another reason for the close interaction between data and theory is that we need theoretical models for understanding and interpreting data. When analyzing data, we are typically interested in the underlying mechanisms (i.e., the causal relationships between variables). Importantly, any pattern of data can be useless and misleading, when the data-generating process is unknown or ignored. Knowledge or at least assumptions regarding the causal process illuminate the data and are required for its sound interpretation. The importance of theoretical assumptions for data analysis cannot be underestimated (see, e.g., the notion of causal models and counterfactual reasoning in [Pearl & Mackenzie, 2018](#)). Thus, **pitting data against theory is nonsense**: Using and understanding data is always based on theory.

Our aim is to prepare **well-documented and reproducible** analysis since data literacy is the ability and skill of making sense of data. This includes numeracy, risk-literacy, and the ability of using tools to collect, transform, analyze, interpret, and present data, in a transparent, reproducible, and responsible fashion.

Information is what we want, but data are what we've got. The techniques for transforming data into information is the *data science!*

Data scientists, therefore, are individuals who strive to transform the plentiful data available today into actionable information, which often appears to be in short supply

Think with data = desire to solve problems using data

A **data scientist** is “a knowledge worker” who is principally occupied with analyzing complex and massive data resources.

Demand for data skills is strong! Data Scientist is one of the best job in the world since 2016.

The Key components that are part of *data acumen* include mathematical, computational, and statistical foundations, data management and curation, data description and visualization, data modeling and assessment, workflow and reproducibility, communication and teamwork, domain-specific considerations, and ethical problem solving.

Therefore, contemporary data science requires **tight integration of statistical, computational, and communication skills.**

Data Wrangling a process of preparing data for visualization and other modern techniques of statistical interpretations.

Note

Recommended Background Readings

1. Baumer, B. S., Kaplan, D. T., & Horton, N. J. (2021). *Modern Data Science with R*.
 - **Chapter 1: Prologue: Why Data Science?**
 - This chapter introduces the motivation behind data science as a discipline. It discusses how the field has evolved, its interdisciplinary nature, and why it is relevant today. The reading helps students understand the **philosophical and practical reasons for studying data science.**
2. Donoho, D. (2017). “50 Years of Data Science.” *Journal of Computational and Graphical Statistics*, 26(4), 745-766.
 - A historical perspective on the evolution of **data science as a field**, emphasizing how it extends beyond traditional statistics.
 - Discusses the distinction between **data science and statistics**, highlighting the importance of computational tools.

- Advocates for a more **inclusive and interdisciplinary** approach to data science education.
3. De Veaux, R. D., Agarwal, M., Averett, M., Baumer, B. S., Bray, A., Bressoud, T. C., & others. (2017). “Curriculum Guidelines for Undergraduate Programs in Data Science.” *Annual Review of Statistics and Its Application*, 4, 15-30.
- Provides an overview of the **core competencies needed for data science education**, including programming, statistical inference, and domain expertise.
 - Defines **guidelines for structuring an undergraduate data science curriculum**, which is useful for understanding how your course fits within broader educational frameworks.
 - Emphasizes the importance of **ethics, reproducibility, and communication skills** in data science training.

What is data?

Today, the manner in which we extract meaning from data is different in two ways—both due primarily to advances in computing:

- we are able to compute many more things than we could before, and,
- we have a *lot* more data than we had before.

the traditional two-dimensional representation of data: rows and columns in a data table, and horizontal and vertical in a data graphic. For instance, if someone aimed to collect or compare health-related characteristics of some people, he or she would measure and record these characteristics in some file. In such a file of health records, some values may identify persons (e.g., by name or some ID code), while others describe them by numbers (e.g., their age, height, etc.) or various codes or text labels (e.g., their address, profession, diagnosis, notes on allergies, medication, vaccinations, etc.). The variables and values are typically stored in tabular form: If each of the table’s rows describes an individual person as our unit of observation, its columns are called *variables* and the entries in the cells (which can be referenced as combinations of rows and columns) are *values*. The contents of the entire table (i.e., the observations, variables, and values) are typically called “data”.

Non-traditional data types (e.g., geospatial, text, network, “big”) and interactive data graphic are the new normal! Thus, we can argue that the gap between generating scientific insights and understanding them is widening.

The increasing complexity and heterogeneity of modern data means that each data analysis project needs to be custom-built. Simply put, the modern data analyst needs to be able to read and write computer instructions, the “code” from which data analysis projects are built.

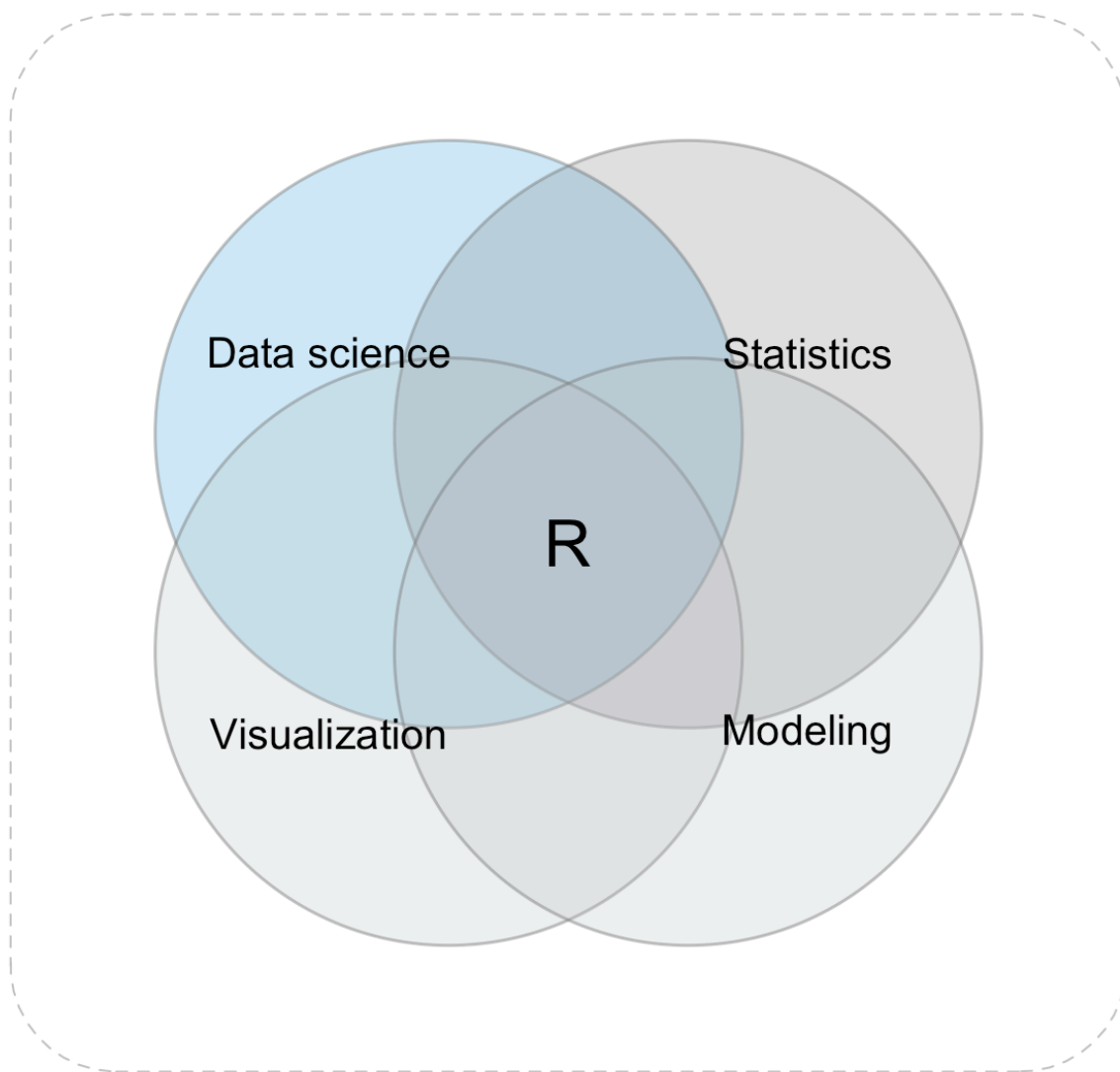
Domain knowledge is always useful in data science since data science is best applied in the context of expert knowledge about the domain from which the data originate. For data scientists of all application domains, creativity, domain knowledge, and technical ability are absolutely essential.

Some related discussions in this contexts include:

1. The distinction between *raw data* (e.g., measurements, inputs) and *processed data* (results or interpretations): Data are often portrayed as a potential resource: Something that can be collected or generated, harvested, and — by processing — refined or distilled to turn it into something more valuable. The related term *data mining* also suggests that data is some passive raw material, whereas the active processing and interpretation of data (by algorithms, rules, instructions) can generate interest and revenue.
2. The distinction between *data* and *information*: In contrast to data, the term *information* has a clear formal definition (see [Wikipedia: Information theory](#)). And while data appears to be neutral, information is usually viewed as something positive and valuable. Hence, when data is being used and perceived as useful, we can generate *insight*, *knowledge*, or perhaps even *wisdom* from it?
3. The field of signal detection theory (SDT, see [Wikipedia: Detection theory](#)) distinguishes between *signal* and *noise*. Does *data* include both signal and noise, or should we only count the signal parts as data?

What is Data Science (DS)?

DS is not a single and homogeneous discipline. Instead, it overlaps and is intricately interwoven with several other academic fields and requires corresponding skills. For instance, becoming an expert in DS requires considerable knowledge in statistics, but the discipline of data science is not as mathematical as statistics. Similarly, data science involves computers, but is usually not as much concerned with abstract formalisms as computer science. We could describe DS as applied statistics or applied computer science, but this would suggest that it is a sub-discipline of these other fields — and any serious application ususally assumes knowledge of the discipline’s foundations.



As R is defined as a “software environment for statistical computing and graphics” ([R Core Team, 2023](#)), it supports and provides tools for all topics mentioned and is thus pictured at the center. However, this central position does not imply that R is the only or a necessary tool for DS (e.g., a popular alternative is the programming language [Python](#)).

i Note

A striking example of bad software choices is the Public Health England (PHE)’s recent decision to import CSV-files on COVID-19 test results into Microsoft Excel’s XLS file format. Due to this file format’s artificial limit to a maximum of 65.536 rows of data, nearly 16.000 Covid cases went unreported in the U.K. (which amounts to almost 24% of the cases recorded in the time span from Sep 25 to Oct. 2, 2020).

The skills for successfully dealing with data are not confined to one discipline or talent. As data scientists must discover, mine, select, organize, transform, analyze, understand, communicate and present information, they tend to be generalists, rather than specialists. Beyond a set of skills from a diverse range of areas, getting DS done requires the familiarity with and mastery of suitable tools.

The *same* data can be represented in many *different types* and *shapes*.

Types of data

As variables and values depend on what we want to measure (i.e., our goals), it is impossible to provide a comprehensive list of data types. Nevertheless, computer scientists typically categorize data into different types. This is made possible by distinguishing between different ways in which data is represented. The three most basic types of data are:

1. *Truth values* (aka. *logicals*): either TRUE or FALSE
2. *Numbers*: e.g., 2,12, $\sqrt{22}$,12,2
3. *Text* (aka. *characters* or *strings*): e.g., “Donald Duck” or “War and Peace”

Most computer languages have ways to represent these three elementary types. Exactly how truth values, numbers, or text relate to the actual phenomena being described involves many issues of representation and measurement.

In addition to these three basic data types, there are common data types that have precise and widely-shared definitions, like

- *dates and times*: e.g., 2025-02-10, 11:55
- *locations*: e.g., the summit of Mount Everest, or N40 44.9064’, W073 59.0735’

but even more potential data types that we could define, if we had or wanted to, for instance

- *temporal terms*: e.g., Monday, noon, or tomorrow
- *visualizations*: e.g., a bar chart, or Venn diagram

As we can express more complex measures in terms of simpler ones (e.g., as numbers or by some descriptive text), we can get quite far by combining our three basic data types (e.g., dates, times and locations can be described as combinations of characters and text).

Shapes of data

Beyond distinguishing data types, we can also ask about the *shape* of data or representations. While it gets challenging to answer questions like “How is the number 2 represented in the brain?” or “What is the shape of yellow?”, we can simplify our lives by asking: “In which shape do computers represent data?”.

1. *scalars*: e.g., 1 or TRUE
2. 1-dimensional vectors and lists: e.g., 1, 2, 3 or 'x', 'y', 'z'
3. 2-dimensional matrices: e.g., a table of health records for different people

Just as for data types, we can easily extend these basic shapes into more complex data formats, like

- n-dimensional arrays, e.g., the number of Titanic passengers by **age**, **sex**, and **survival**
- non-rectangular data, e.g., a list of sentences, what someone did last summer

For more see: **Introduction to Data Science** <https://bookdown.org/hneth/i2ds/intro.html>

An introduction to R and RStudio:

Introduction to R and RStudio

R is a powerful and versatile programming language and environment for statistical computing and data analysis. RStudio is a popular integrated development environment (IDE) that provides a user-friendly interface for working with R.

Note

The R language is a free, open-source software environment for statistical computing and graphics. Download and install **R** from the Comprehensive **R** Archive Network (CRAN, <http://www.r-project.org>)

RStudio is an open-source integrated development environment (IDE) for R created by Posit that adds many features and productivity tools for R. . Download and install (RStudio) from <https://posit.co/products/open-source/rstudio>.

R Basics

Using R as a Calculator

You can use R as a calculator to perform basic arithmetic operations. Just type in your calculations, and R will provide the results.

```
# Addition
```

```
3 + 5
```

```
[1] 8
```

```
# Subtraction
```

```
10 - 2
```

```
[1] 8
```

```
# Multiplication
```

```
4 * 7
```

```
[1] 28
```

```
# Division
```

```
15 / 3
```

```
[1] 5
```

Creating New Objects

In R, you can create and store values in objects. This is done using the assignment operator `<-`. You can name the object on the left and assign a value or expression to it on the right.

```
# Creating objects
```

```
x <- 3 * 4
```

```
y <- 3 + 4
```

```
z <- 12 / 6

a <- x * y - z
```

Functions

R has a vast collection of built-in functions, each with specific purposes. Functions are called by their names followed by arguments in parentheses.

```
# Using built-in functions

seq(from = 10, to = 68, by = 3) # Sequence of numbers
```

```
[1] 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67
```

```
rep(3, 6) # Repeat a value
```

```
[1] 3 3 3 3 3 3
```

You can also create your own functions in R.

```
# Defining a custom function

ifnegative <- function(x) {

  if (x < 0)

    print("negative")

  else

    print("positive")

}

# Calling the custom function

ifnegative(-5)
```

```
[1] "negative"
```

```
ifnegative(2)
```

```
[1] "positive"
```

Operators

R supports various operators for arithmetic, comparison, and logical operations. Here are some commonly used operators:

Arithmetic Operators

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- ^ or ** (Exponentiation)
- %% (Modulus)

Comparison Operators

- < (Less than)
- > (Greater than)
- <= (Less than or equal to)
- = (Greater than or equal to)
- == (Equal)
- != (Not equal)

Logical Operators

- `!` (NOT)
- `|` (OR)
- `&` (AND)
- `isTRUE` (Test if an expression is TRUE)
- `isFALSE` (Test if an expression is FALSE)

Note

The `|` is an “or” operator that operates on each element of a vector, while the `||` is another “or” operator that stops evaluation the first time that the result is true!

Colon Operator

The colon `:` operator is used to create sequences of numbers, making it helpful for creating numeric vectors.

```
# Creating a sequence of numbers  
  
1:10 # Generates numbers from 1 to 10
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

Checker `%in%` Operator

The `%in%` operator checks if elements belong to a vector and is often used for data manipulation.

```
# Checking if elements belong to a vector  
  
a <- c(1, 2, 3, 4, 5)  
b <- c(3, 4, 5, 6, 7)  
  
a %in% b # Check if elements in 'a' belong to 'b'
```

```
[1] FALSE FALSE TRUE TRUE TRUE
```

Data Types in R

R supports different data types, including:

- **Numeric:** These are numeric values (e.g., 3.14, 42).

```
x <- c(1, 2, 3, 4.5, 6.7)
```

Character: These are text values (e.g., “apple,” “banana”).

```
y <- c("apple", "banana", "cherry")
```

Logical: These are binary values (e.g., TRUE or FALSE).

```
z <- c(TRUE, FALSE, TRUE, FALSE, TRUE)
```

Data Structures

Table 1: R offers various data structures to store and manipulate data:

	Homogeneous	Heterogeneous
1d	Atomic Vector	List
2d	Matrix	Data Frame
nd	Array	

Atomic Vectors

- Atomic vectors store homogeneous data, meaning all elements are of the same data type.

```
# Create an atomic vector  
var1 <- c(1, 2.5, 4, 6)
```

Lists

- Lists store heterogeneous data, allowing different data types in the same list.

```
# Create a list  
var2 <- list("name", 2, 3, c("item1", "item2"), 16, 75)
```

Matrices

- Matrices are two-dimensional data structures.

```
# Create a matrix

a <- matrix(1:6, ncol = 3, nrow = 2)
```

- Data frames are a versatile and commonly used data structure in R. They are similar to matrices but can store both numeric and non-numeric data. Data frames are often used to store datasets. Data frames are particularly useful for working with tabular data, and you can perform a wide range of operations on them.

```
# create a data frame
data <- data.frame(
  Name = c("Alice", "Bob", "Charlie"),
  Age = c(25, 30, 22),
  Score = c(95, 88, 73)
)
```

Get your data into R

R have multiple in-built data

```
# to see which data you have
data()

# for instance, you can check out data
mtcars
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Sportabout											
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Fleetwood											
Lincoln	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Continental											
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Reading existing local data

```
# read csv file
mydata <- read.csv("~/Desktop/mac_projects 2/data-science-with-R/data/cars.csv")
View(mydata)
# write csv
write.csv(mydata, file =
           "~/Desktop/mac_projects 2/data-science-with-R/data/mydata.csv")

# read csv-file from net
mydata <- read.csv(
  "https://gist.githubusercontent.com/MorganZhang100/0c489d1f376a04d5436a/raw/7c335ebe48e575"
)
View(mydata)
```



```
# read excel
library(readxl)
mydata1 <- read_excel(
  "~/Desktop/mac_projects 2/data-science-with-R/data/datacom.xlsx")
View(mydata1)
```

Tip

You can use `here()` package in order to short-cut reading and writing data! The “here” package in R is a useful tool for creating file paths in a project-agnostic way. It provides a simple and consistent method for specifying file paths in your R scripts and projects, making your code more portable and less prone to errors when you share it with others or move it between different machines.

Here are some key aspects and benefits of the “here” package:

1. Platform-Independent Paths:

- “here” automatically generates file paths that are platform-independent. This means your code will work seamlessly on Windows, macOS, and Linux without the need to manually adjust path separators (e.g., using “\\” or “/” depending on the operating system).

2. Project-Agnostic Paths:

- “here” is designed to work within R projects. It helps you create paths relative to the root directory of your project, making your code independent of the specific folder structure on your machine.

3. Consistency:

- By using “here” to specify file paths, you ensure consistency across your project. This reduces the likelihood of errors caused by incorrect or inconsistent path specifications.

4. Easy Integration:

- The “here” package can be seamlessly integrated into your R scripts and projects. You don’t need to install any additional dependencies or libraries.

Here’s how you can use the “here” package in R:

1. Installation and Loading:

- You can install the “here” package from CRAN using the following command:

```
# install.packages("here")
library(here)
```

2. Creating & using Paths:

- To create file paths relative to your project's root directory, use the 'here()' function. For example, to specify a path to a data file in your project, you can do the following:

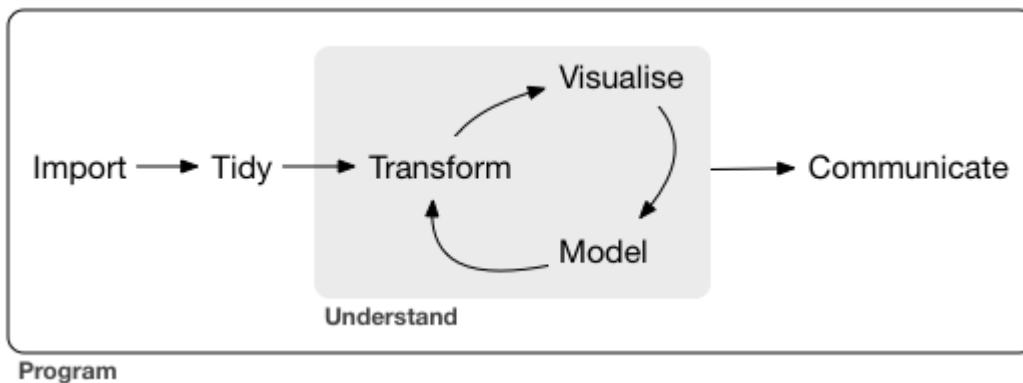
```
dpath <- "~/Desktop/mac_projects 2/data-science-with-R/data"
mydata2 <- read.csv(here(dpath, "/cars.csv"))
write.csv(mydata, here(dpath, "cars2.csv"))
```

The “here” package simplifies path management in R, making your code more robust and portable. It's especially helpful in larger projects where consistent path specifications are essential for reproducibility and collaboration.

Tidy Data

First you must **import** your data into R. This typically means that you take data stored in a file, database, or web application programming interface (API), and load it into a data frame in R. If you can't get your data into R, you can't do data science on it!

Once you've imported your data, it is a good idea to **tidy** it. Tidying your data means storing it in a consistent form that matches the semantics of the dataset with the way it is stored. In brief, when your data is tidy, each column is a variable, and each row is an observation. Tidy data is important because the consistent structure lets you focus your struggle on questions about the data, not fighting to get the data into the right form for different functions.



Once you have tidy data, a common first step is to **transform** it. Transformation includes narrowing in on observations of interest (like all people in one city, or all data from the last

year), creating new variables that are functions of existing variables (like computing speed from distance and time), and calculating a set of summary statistics (like counts or means). Together, tidying and transforming are called **wrangling**, because getting your data in a form that's natural to work with often feels like a fight!

Once you have tidy data with the variables you need, there are two main engines of knowledge generation: visualisation and modelling. These have complementary strengths and weaknesses so any real analysis will iterate between them many times.

Visualisation is a fundamentally human activity. A good visualisation will show you things that you did not expect, or raise new questions about the data. A good visualisation might also hint that you're asking the wrong question, or you need to collect different data. Visualisations can surprise you, but don't scale particularly well because they require a human to interpret them.

Models are complementary tools to visualisation. Once you have made your questions sufficiently precise, you can use a model to answer them. Models are a fundamentally mathematical or computational tool, so they generally scale well. Even when they don't, it's usually cheaper to buy more computers than it is to buy more brains! But every model makes assumptions, and by its very nature a model cannot question its own assumptions. That means a model cannot fundamentally surprise you.

The last step of data science is **communication**, an absolutely critical part of any data analysis project. It doesn't matter how well your models and visualisation have led you to understand the data unless you can also communicate your results to others.

Surrounding all these tools is **programming**. Programming is a cross-cutting tool that you use in every part of the project. You don't need to be an expert programmer to be a data scientist, but learning more about programming pays off because becoming a better programmer allows you to automate common tasks, and solve new problems with greater ease.

For more see: [Modern Data Science with R - Appendix B — Introduction to R and RStudio \(mdsr-book.github.io\)](https://mdsr-book.github.io)