Programming for Practical Data Analysis

Prof. Dr. Johannes Binswanger

Fall 2016

Lecture 2: Variable types, IDEs, command prompt, and some other concepts


Variable types, data structures, and the organization of memory

DATA STRUCTURES

Source: Eneas De Troya - Flickr, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=3407867

Source: By Gyga - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=969037

DATA STRUCTURES

- A very nice article on data structures and why they are so useful can be found on https://www.topcoder.com/community/data-science/data-science-tutorials/data-structures/
  - It compares a computer's memory to a library: If the library is well ordered, then you will be much faster finding the books you need.
  - Its order has several layers: Books are stored in bookshelves, bookshelves have "addresses" (e.g. "G9", or was this Ikea…?) etc.
    - On top of this, there may be catalogues, maybe they contain sub-catalogues.
  - Thus, the "library system" is much more then books standing on shelves, as it includes the addressing system, the catalogues etc.

DATA STRUCTURES

- From a very basic perspective, a computer's memory is just like a giant shelf containing bits (instead of books).
  - A bit is the smallest unit of information; physically, storing a bit in memory means either a low charge or high charge (or low/high magnetization, light intensity etc.)
- Actually, think of a *bit* as a little square on a page in a book which either contains dried ink or not.
  - From this point of view, a library actually consists of a giant collection of just those little squares…
    - They are very well organized to make up a library, and augmented by auxiliary systems like the addressing system and catalogue system.
  - Books are just particular combinations of these little squares.
  - In the language of programming, books are variables of certain types (monographs, dictionaries, magazines), and they have a variable name (the book title).
  - Books can be combined to greater objects (single shelves, collection of shelves with books on similar topics etc. etc.). These would also be variable types.

DATA STRUCTURES

- In order for a the whole system to be useful, there needs to be a lot of structure indeed, organized in a hierarchical way.
- Once more:
  - Squares are grouped to letters, letters are grouped to words, words are grouped to paragraphs, …, … grouped to books.
  - Books are put into shelves, grouped together with other books that are on a similar subtopics (e.g. industrial economics as a part of microeconomics).
  - Books on subtopics in microeconomics are grouped together into the section of microeconomics, … economics, … social sciences, … science, …

DATA STRUCTURES

- And again, remember that a library system not only contains books!
  - It also contains addresses of the shelves,
  - A plan of the location of the shelves,
  - A catalogue system…
- The catalogue and written addresses again consist of words, letters, dots…
- This is a hugely complex system
  - If the structure of its components is well chosen and efficient, a book search takes just minutes.
  - If the structure is chaotic, the catalogues are not well organized, it may take days or, in the worst case, weeks.

DATA STRUCTURES

- Also in computer memory, bits are hierarchically structured into more and more higher-level units.
- The currently smallest unit of interest in a memory are – as in books – called “words”!
  - Those have a length of 32 bits in a 32-bit system.
  - And a length of 64 bits in a 64-bit system.
  - Heureka!
- Words are then grouped together to variables, to code instructions, to an addressing system etc., just like in a library.

DATA STRUCTURES

- You have just seen variables of different types in R
  - Vectors of characters
  - Vectors of numeric information
  - Factors
  - Vectors and factors bundled into data.frames
  - Lists
  - … And there are also matrices, arrays, data.tables
- In other languages you have sometimes different structures
  - In particular, in object-oriented languages there are classes and objects.
  - A class is a blueprint for a particular object type
  - Objects are user-defined data structures that fit particular real-world objects (hotel or flight bookings, a regression estimation, or an Angry Bird…)

DATA STRUCTURES

- Data structures differ in complexity, scope, and what they can contain.
  - Think of a vector as a linear shelf where all books are on the same level along just one aisle. All books are on the same topic.
  - A data.frame combines several vectors by collecting books in particular shelves into a section of a library that has its own address.
  - A list is the most encompassing category that can combine sections of the library on wildly different topics located in different shelves.

VARIABLES

- In programming, variables are objects that you assign an name that refers collectively to information stored at various addresses (“books in various shelves”) in memory.

PROGRAMMING VS. MENU-DRIVEN SOFTWARE

- When you work with Excel and you choose something from a menu, you are, in fact, also manipulating data.
- Internally, these data are also organized in structures and variables with types and names.
- However, you are shielded from the internal structure and cannot access is directly (unless you program in Visual Basic or C#)
- When you program, you administer your library yourself!
  - But you have millions of well trained-library assistants that help you doing this (the internal structure of the program, the interpreter, operating system etc.).
  - You only talk to the “heads” of the library assistants in each domain, so to speak.


"By reference, by value…"

PASSING ARGUMENTS “BY REFERENCE” OR “BY VALUE”

- With the library analogy, we can shed light on an otherwise tricky subject that we briefly already discussed during the first class.
- Suppose we have the following program:

```
a = 5
b = a
a = 10
b
```

- What will be the value of b? 5 or 10?

PASSING ARGUMENTS “BY REFERENCE” OR “BY VALUE”

Let’s consider a library… a is a reference to the book at a shelf with a particular address, where we put a book with title “5”.

- Suppose we “pass **by reference**” :
  - In this case, the “=“ in “a=b” means: b is a reference to the book that has the **same address** as a.
  - So if we change a by replacing the book on “5” with the book on “10”, then the content of b will also change!
  - b points to the same address as a!
- Suppose we “pass by **value**” :
  - In this case, the “=“ in “a=b” means: b is a reference to the **book itself** that is stored at the shelf which a refers to at the moment of assignment.
  - So if we change b replacing the book on “5” with the book on “10”, the content of a will *not* change since it only matters what a was referencing to at the moment we assigned b the value of a!

PASSING ARGUMENTS “BY REFERENCE” OR “BY VALUE”

- On a purely mathematical level, passing by reference would be the natural thing

```
a = 5
b = a
a = 10
```

- The mathematical conclusion is: `b = 10`. After all, `b` is set equal to `a`…
- However, interpreting code just as mathematical equations can lead to serious misunderstandings. For instance consider

```
a = 5
a = a + 1
```

From a mathematical point of view, this would never make sense.

PASSING ARGUMENTS “BY REFERENCE” OR “BY VALUE”

- Code does not just represent mathematical equations.
- The “=“ in “a=x” in coding means: **assign** a the value x **.**
- And as you have just seen, there are sometimes two ways of understanding this assignment

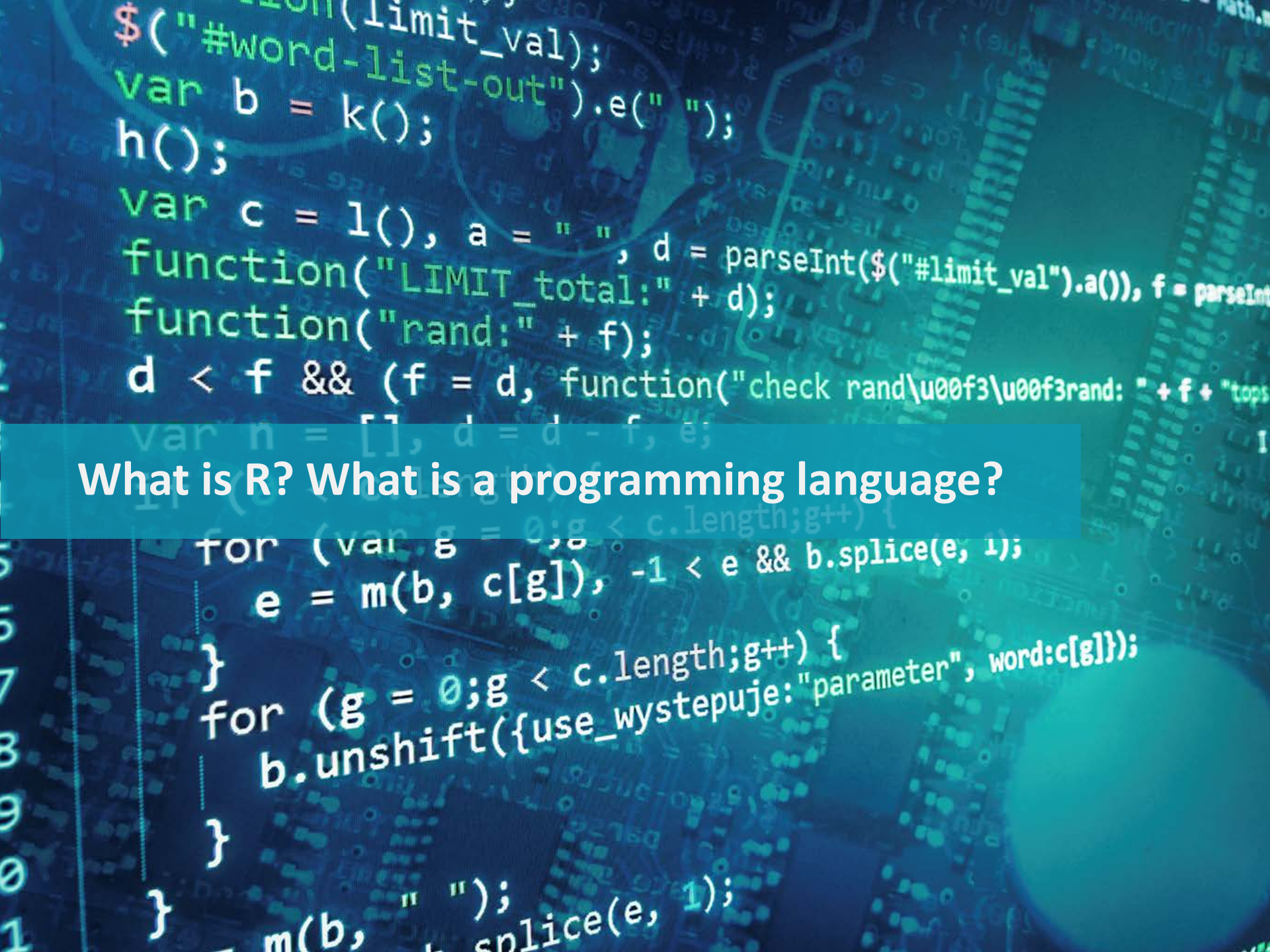
PASSING ARGUMENTS “BY REFERENCE” OR “BY VALUE”

- In R, the rule is generally “by value”.
- So the output in R of

```
a=5; b = a; a = 10; b
```

  is 5!

- However, especially in object-oriented languages including Python, there are cases where arguments are passed by reference, especially for more complicated data structures.
  - This can lead to confusion.
- I have made you aware of this since you may want to acquire some general programming skills, not just of R.
- Btw., R is also an “object-oriented” language. But as a data analyst, you are mostly shielded from this, unless you dig very deep.
  - This may make R a little more accessible than Python. You can do intermediate-level programming in R without touching object-oriented stuff, while, for intermediate-level programming in Python, you will already need to master classes and objects.


What is R? What is a programming language?

PROGRAMMING LANGUAGES

- R is the programming language. What does that mean?
  - R is a specific way to write instructions to a computer.
  - (Python would be another way to write instructions, as would be Java etc.)
  - The computer only understands “machine language”, i.e. physical combinations of binary signals.
  - Software engineers had the brilliant idea to generate “high-level” language that are
    - Relatively easily understandable to humans
    - But the translation into machine language can be easily automated.
  - That’s what we understand today by “programming language”.

PROGRAMMING LANGUAGES

- So R (or any programming language) is a system that
  - Accepts commands, written in code,
  - Translates the commands into machine language,
  - Works closely together with a computer's operating system (Windows, OS X, Linux),
  - Together with the operating system, takes care that the machine executes the commands,
  - And that the user gets the results.

PROGRAMMING LANGUAGES VS. SOFTWARE

- If course, you can interact with a computer other than a programming language.
  - If you format text in a Word document or make some graphic plots in Excel or go to a website in your browser you do not write any code.
  - Instead, you use what is called a graphical user interface (GUI) with menus tabs and the like.
- A GUI is much more intuitive for simple things, but it also has some important disadvantages.
  - If you want to repeat 1000 times what you have done in a GUI, you have to literally do it 1000 times (with all the clicks etc.)
  - It is trivial to repeat chunks of written code 1000 times.
- The solution in modern computing environments is a combination of both:
  - Use a GUI for simple things that are not repeated many times in a row in the same way.
  - Use code for things that are complex and/or that you want to automate.

PROGRAMMING LANGUAGES VS. SOFTWARE

- Although the difference between “software” and “programming language” is fuzzy, you can think about it as follows:
  - Software provides IT solutions where the interaction between you and the machine is mostly based on GUIs.
  - A programming language lets you interact with your machine with code.

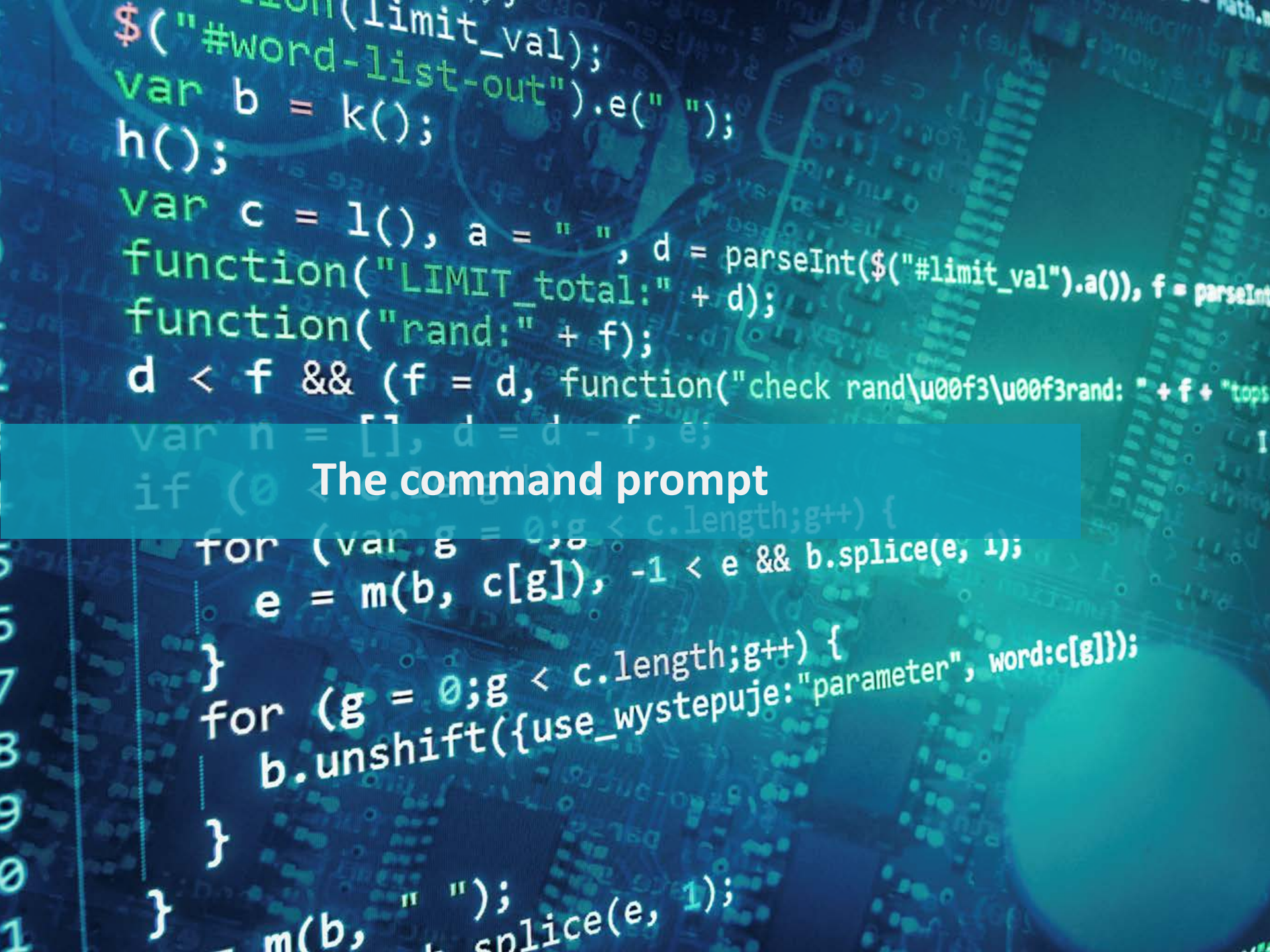
PROGRAMMING LANGUAGES VS. SOFTWARE

- Data analysis contains both elements: complexity and automation (you will see).
- So for all both the most trivial ways to analyze data, coding is vastly better than GUI-style analysis (e.g. with Excel).


R vs. RStudio

RSTUDIO

- Now you understand what R does, and why R is a “programming language” rather than just a “software”.
- But what does RStudio do?
- RStudio is just an environment, containing GUI elements, that makes it easier to write code.
- This type of environment is called an **editor** or an **Integrated Development Environment (IDE).**
- RStudio is not necessary for running R!
- To really understand that you could work without RStudio, and how this would feel, let’s actually try to do it!


The command prompt

THE COMMAND PROMPT ON WINDOWS

- Before computers had GUI environments to do things like double-clicking to open a document, they were run via “command lines” from “command prompts”.
- These things are still there, and sometimes they are still useful today.
- Go to the search lens and type “command prompt” (or “cmd”) and then click on “Command Prompt”
- We want to invoke R and computer ‘2+2’
- So first type “R”
- You will get an error message. The problem is that your system does not know where to find R.
- So we’ll tell it.

WHERE IS R ON WINDOWS?

- On your system, you find R most likely in a directory like

  C (or “This PC”/“Windows”) / Program Files / R / R-3.3.1 / bin

- There you see the files

  `R.exe` and `Rscript.exe`

- This is the “heart” of R.
- To tell the operating system where these files are, such that we can invoke them from the command prompt, we need to do something called “Set the PATH”.
  - This shows you some of the inner life of your machine.

SETTING THE "PATH" IN WINDOWS

- Go to Control Panel (types this into the search lens)
- Click on "System"
- Click on "Advanced system settings" on the left.
- Click on "Environment Variables"
- Make sure that "PATH" is selected in the top panel, then click on "Edit"
- Click "New" and add the directory of the bin folder mentioned above (where R.exe and Rscript.exe are located).
  - To do so, click in to the address bar in the file finder, copy the resulting address, and paste as a new PATH.
- Click OK and close all related windows.

RUNNING R IN COMMAND PROMPT ON WINDOWS

- Close the command prompt and reopen it. Now type R, you get some output
- Then type 2+2, what do you get?
- Before you proceed with the commands on the next slide, write `quit()` into the command prompt, and confirm with `n`.

R WITHOUT IDE IN WINDOWS

- Now we want to run an R script from the command prompt.
- Make a new folder within the folder you have the documents for this course. Call it, say, `RfromCMD`.
- Now go to the command prompt.
- Navigate to the directory `RfromCMD`:
  - If necessary, type first the root directory, i.e. type `cd C:\` or `cd D:\` This step is not necessary if the command prompt shows that you are already in the partition of your system that you need to go!
  - Then go to the file explorer and copy the directory of `RfromCMD` by clicking into the address bar and then ctrl + c.
  - Go to the command prompt. Write `cd` ("change directory"). Then, if your address to be inserted contains any blanks, enter an opening quote". Then paste the address, add the closing quote and hit enter:

```
cd "C:\bla bla\bli blu\RfromCMD"
```

- Type

```
fsutil file createnew hello.R 0
```

- Professional IT people use this a lot. You can use similar tricks to manage whole IT systems…

R WITHOUT IDE IN WINDOWS

- Now open the `hello.R`, but NOT with Rstudio but just a text editor (e.g. Notepad)
- Type “Hey there, you can run an R script without Rstudio!!”
- Then save the file.
- Go to the command prompt and type

```
Rscript hello.R.
```

RSTUDIO?!

- So, you see, we didn’t need Rsudio. A simple text file was enough as an **editor**.
- But was it convenient? It would be to have something that is more integrated with our working environment, in other words an IDE.
- And so, that is what RStudio does.
- And, by the way, you can run R without Rstudio, but the opposite is not possible, obviously!

THE COMMAND PROMPT ON A MAC

- On a Mac, the command prompt is called “terminal”. You get it as follows:
  - Open the Finder
  - Select “Applications” on the left side
  - Click on “Utilities” and scroll down to terminal.
  - It looks white, but you can give it more fancy looks by pressing command and “,” at the same time.

RUNNING R FROM THE TERMINAL ON A MAC

- And now you see some of the things that are better on OS X than on Windows:
  - You can just type R, and then '2+2', everything is ready!
  - Quite R by writing `quit()`, and confirm with `n`; this step is important, otherwise, the commands on the following slide will not work!

HOW NAVIGATE TO A DIRECTORY IN TERMINAL ON A MAC

- In the finder, navigate to the folder where you want to have your R script located
  - Say a folder called `RfromCMD` inside the folder that you use for this course.
  - Navigate to folder containing `RfromCMD`. Right-click on the folder and press the Option key.
  - Click on `Copy "RfromCMD" as Pathname`.
- Go to the terminal, write `cd "`, then copy the pathname, and add a ":

```
cd "/Users/bla bla/bli blu/…/RfromCMD"
```

- You are there…

HOW CREATE A FILE FOR YOUR R SCRIPT ON A MAC?

- Now we need to create a file containing your R script. Just type `touch hello.R`
- You will see in the finder how this file appears; cool, isn't it?

R FROM THE COMMAND PROMPT ON A MAC

- Go to the new file hello.R and open it with TextEdit. Now write some code like
  - `exp(3.5)`
  - You can also write text like: “Working from the terminal is the coolest thing I’ve ever done.” But there can be some difficulties with the quotes and you may get an error for reasons that we do not go into now.
- Save it, go to the terminal and type
  - `Rscript hello.R`
- And there we go, you do not need RStudio…
- But again, RStudio is pretty convenient, in comparison!

IDES

- Initially, there were no good IDEs for R (or Python).
  - So you got a flavor how people were working at these times.
  - Of course, there are more sophisticated ways of working with R and the command prompt/terminal without an IDE. But we will not need it, you got the idea.