`./img/subway.gif`

Exploring Data Science With »R«

May 19, 2021

EXPLORING DATA SCIENCE WITH »R« By Marcus Birkenkrahe, Berlin

Title GIF: Looking at the title image begs the question: how can I get on that train? *(Image: Robbie Khan, London subway).*

Thank you for giving me the opportunity to give this talk. I was originally asked to give a "class". It took me a while to find a topic that seemed suitable given the circumstances of virtual delivery.

They say (I think Stephen Hawking first said it in "A Brief History of Time"?) that every mathematical equation cuts the readership of a book in half.

I think there is a similar law for online presentations: if you cannot grasp the listener's imagination in the first 2 minutes, you will lose him or her. It may only be 1 minute, and my first minute is almost up.

So let me ask you this: how many programming languages do you think existed in the year 1966 (that's 55 years ago)? Type your answer in the chat! Everybody! Come on, it can't be so difficult. [...] The answer is: 700 (Landin, 1966).

If you thought this was fascinating, puzzling and interesting, stay on the line!

What will you learn?

Two ways of looking at data
Data science $\leftarrow$ modeling $\cup$ coding
Data science examples using R
R vs. Python for data science

NOTES:

- We begin by looking at the **purpose** of data science through the lense of two typical setups.
- Each of these setups comes with its own **workflow**. This is a process that can be modeled, and we'll look at these models together with two examples - text mining and recommender algorithms.
- Data science will now emerge as the marriage of two skills - modeling and coding[1]. The resulting framework allows us to place practical examples, programming languages and different disciplines on one and the same map.
- In the second half of my talk, I'm going to look more closely at the examples mentioned earlier: text mining the novels of Jane Austen, and predicting success in speed dating, and I will present a list of projects from two of my current data science courses.
- Then we're going to take a closer look at R itself, and at its greatest contender in the data science world, Python.
- At the end, I also talk about the tools I used to create this presentation. This will cover the aspect of productivity.

  So the first half is conceptual, the second half is rather technical, though because this is a singular event of limited time, I will not be able to give a lot of detail compared to a regular classroom lecture.

  Also, it's going to be a lot less interactive, not only because we're in digital teaching mode. But I wanted to show you a spectrum of things today. Which is in line with the immaturity of data science as a discipline - there are many ways of teaching it well.

The Genesis of this talk:

Originally, I wanted to give an introduction to R, and I thought it would be fun to contrast it with Python, because this is something students always ask when I teach them R instead of Python.

[1]Practitioners like to distinguish between programming and coding: programming is the creation of algorithm-based software solutions, while coding is only an aspect of programming focused on simpler solutions and smaller pieces of code. Learners mostly code, developers program (cp. GeeksforGeeks 09/08/20).

Then I realized that introducing R as a programming language in 45 minutes, without the necessary followup of practice, would be pointless and quite possibly terribly boring. I needed something short and snappy to hang both a data science introduction and R out to dry.

Then something came back to me that one of the professors had said in a meeting with me last week, to explain why Lyon College has a data science program in the first place. Out of that remark came the abovementioned data science framework.

Now, I needed some examples to illustrate the concepts, and, in the end, carry out my comparison of R and Python. Voilá!

What can you do?

- Answer polls, like "do you know R?"
- Leave comments/questions in the chat
- Download the slides (PDF) here
- Learn R, e.g. here

NOTES:

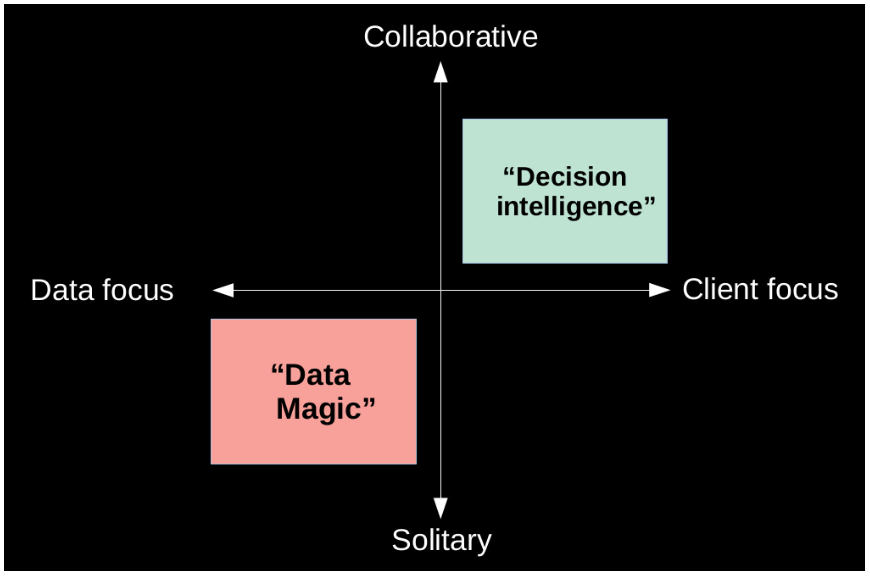
Please go to the poll to let me know if you already know R - not just know of it, but how to use it. Who knows, perhaps I am facing a group of real experts, in which case I am on a road to nowhere!

You can leave feedback, comments and questions in the chat. I will periodically look at it. You can interrupt me any time for questions though this may implode my time management. Happens to me all the time, especially during online teaching.

You can download the slides as PDF. I'm going to post the URL later into the chat. You can also ask me for the script - my lectures usually come with a script.

If you need to know more about me, check LinkedIn. I am the only one of my name there.

Two ways of looking at data



NOTES:

There are two distinct ways of looking at data. I am sure you’re aware of them and that I only have to spell it out. I think it’s an important foundation for what follows.

One scenario is called "Data Magic". It is characterized by a strong focus on data (instead of clients, or problems that clients may have), and a strong focus on solitary activities, for example programming, or building models.

The other scenario, "Decision intelligence", is at the opposite end of this world: here, there is a strong emphasis on clients and their problems, and on working collaboratively.

Let’s see how these scenarios play out in terms of data science.

Scenario 1: "Data magic"

- We’ve got data!
- We’ve got computers!
- Let’s compute something!

NOTES:

Yes, "data is the new oil", perhaps even "big data"[2]. But "data are profoundly dumb," (Pearl, 2018). Something's missing in this approach - and it's not the power of the machines. You probably know this from your own experience: if your car drives well and is fast, it is hard not to drive fast all the time, no matter if you actually need to be quickly where you want to go. Likewise, powerful computing equipment demands to be challenged and used.

I totally get this attitude, too: I love tech gadgets as much as the next IT guy[3]. But I also know that any tool gets its power from being the right tool used in the right way - you surely know the saying "If all you have is a hammer, everything looks like a nail." This is the same thing.

Good question, bad setup

What's the story behind the data?

NOTES:

Having big customer data - like a successful B2C e-commerce company would - is great. Access to superfast computers is great, too. And who doesn't want more customers?

The "story" that is hidden here could be the story of a company wanting to predict user preference for or against certain products. This is called a recommender algorithm. You've all suffered through the output of these algorithms during your own online shopping journeys!

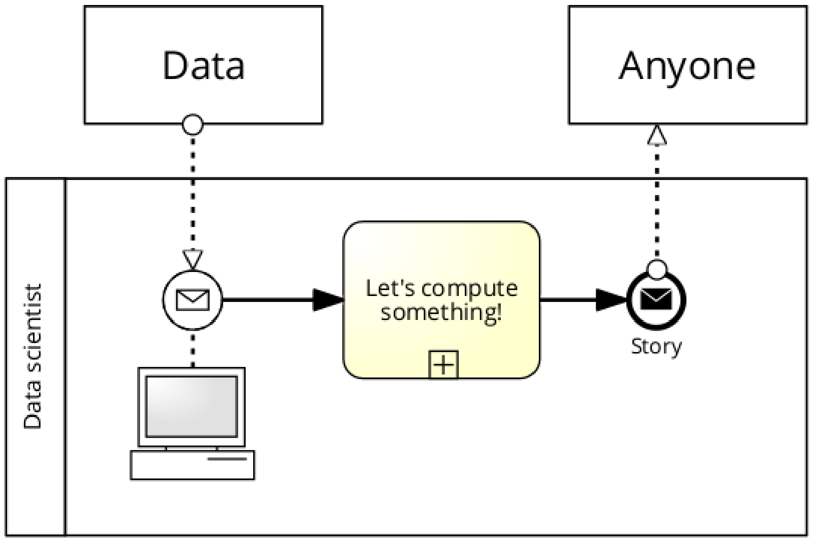
But with this little preparation, the question "What does the data tell us", or the battle cry "Let the data tell a story," is near pointless. It's as if we're treating data science as an **oracle**, asking it to perform **magic**. We

[2]The expression "Data is the new oil" goes back to an article in the Economist (2017). This article was not, however, about data as such, but about regulations of the big tech companies. By contrast, the expression "Big Data" has a technical meaning, usually summarized as "3V" for data of (1) high volume, (2) high velocity, and (3) high variety.

[3]Though I admit leaning towards an "ascetic" position of wanting to extract as much value as possible from technology that I use or own before moving on to the next gadget. Two examples: (1) I do almost all my work - including this presentation right now - on the computer inside GNU Emacs, the LISP-based extensible text editor, which was originally released in 1976. (2)

have needlessly turned the computer into a black box. Needlessly, because computers are actually simple, much, much simpler than humans, and they are begging to be **understood** so that we can use them better for what they were created.

"Data Magic" workflow



NOTES:

This scenario leaves an imprint on people doing work in it. It's reasonable to sketch the workflow or process that follows.

Figure shows the defective workflow as a BPMN[4] model. All kinds of interesting activities and findings may be hidden behind the central task, and the "Story" may be interesting, too, but data and potential beneficiary are effectively decoupled from one another - a "client" as a person (or organization) with an authentic need was never defined.

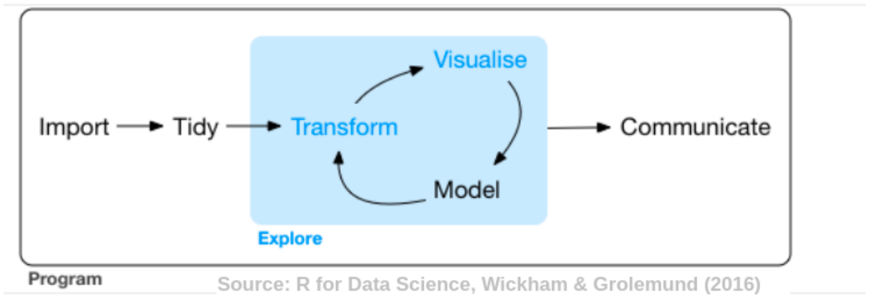
Of course this is a shortened description, it illustrates an extreme. However, other widely accepted descriptions of the data science workflow look

[4]BPMN or Business Process Model and Notation is an industry standard for process model diagrams. It's much more than a drawing though. E.g. you can take a correct model, run it through a so-called BPEL engine and get some executable Java code at the other end. So this is potentially powerful stuff which is why it's a standard like UML.

just the same, e.g. in Wickham & Grolemund (2016), or Wing (2019): data scientists largely talking to themselves, wrapped up in "wrangling", "tidying", and exploring data.

- Link: Wickham & Grolemund (2016).
- Link: BPMN diagram following Wing (2019).

Exploratory Data Analysis



NOTES:

Here you can clearly see the same structure as in the model diagram earlier - even though the central bit, "explore", exhibits some important substructure.

Compare the video "Hadley Wickham "Data Science with R" (Reed college, 2016)

Scenario 2: "Decision intelligence"

- A client got a problem!
- A client needs a solution!

Better question, good setup

What are your options?

NOTES:

This is a good setup to ask an even better, clearer question, namely "What are our options?"

For this question, we start with a different entity - a **problem**! There's nothing wrong with exploring away, but that's an academic luxury. When the rubber hits the road, as the saying goes, there are some real problems that require real solutions. Here, "real" is short for: has a client, that is an identifiable owner or beneficiary, a person or an organization (ideally both - faceless organizations are creepy!).

We can also assume that the problem is not trivial: the **solution** does not present itself. The data are **messy**, but they are a means to an end, and so are the machines. The solution is only a solution if it leads to practical decisions, to an actionable insight. Just a thought, musings or untethered insights, no matter how precious, are not enough to justify our efforts.

Having said this, there is nothing shameful in having the leisure to go after raw insight, especially in education, when training data scientists. However, insight and "tidy" data are useless to practitioners in industry. This is not the only area where education creates a paradox - we know this in other sciences from the lab.

The **humanities** have their own set of emergencies and problems. They are not less pressing or problematic because they are (often) not of the material world. However, the data sciences can only target them, if the problems are quantified, if suitable metrics are available.

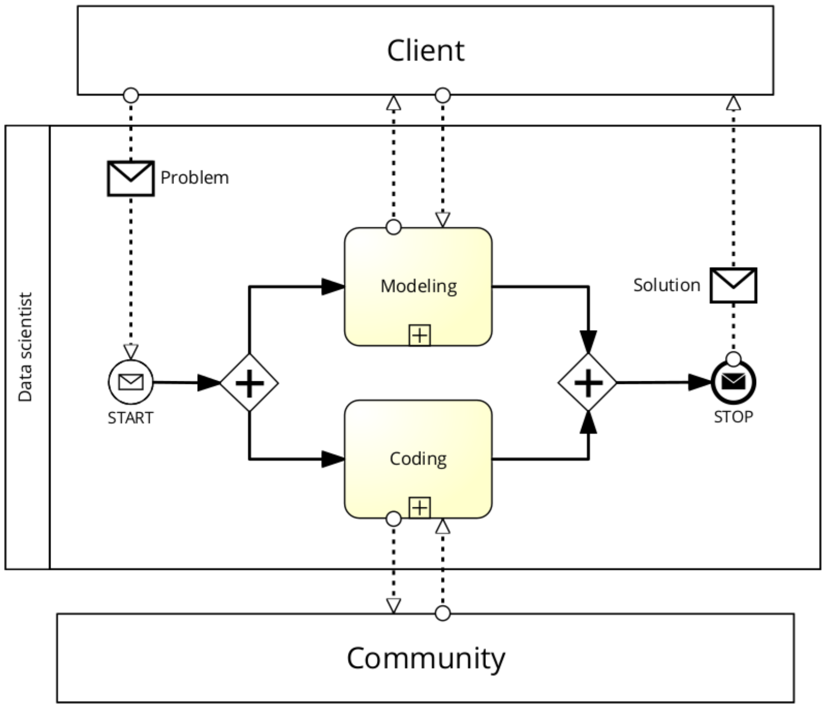
Let's take language learning as an example: I might be interested in automatically annotating a large body of documents for learners of the language in which the documents were written based on their individual profiles. This could become a text mining challenge.

So, we've moved from the "oracle" and its "data magic" to "decision intelligence."[5]. Let's see how the modified workflow looks like.

---

[5]The term is used by Google, which also has a "Chief decision scientist", the well-known data scientist Cassie Kozyrkov.

Problem-based Collaborative Exploratory Data Analysis



NOTES:

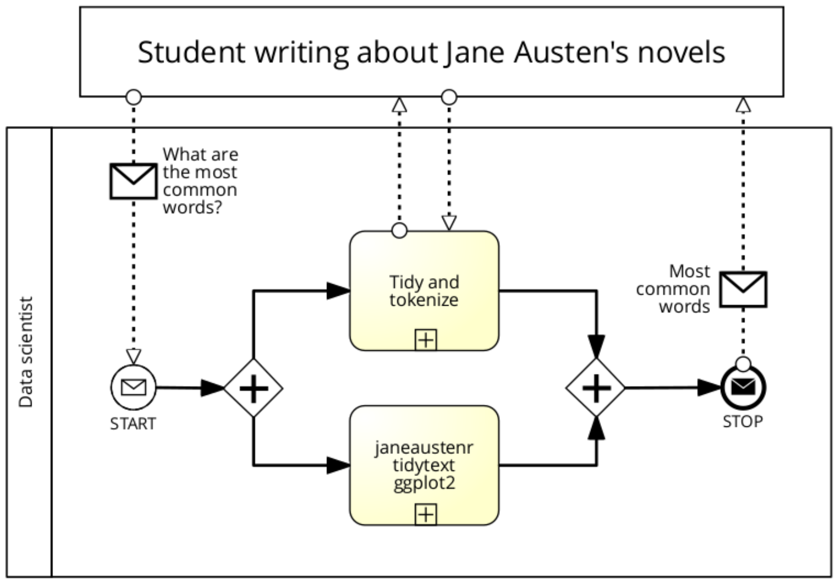
Figure is a modification of with several new elements: (1) a "client", someone who benefits from data science by getting the solution to a problem; (2) "modeling" and "coding" as parallel processes either of which can bring about the solution[6] though they mostly work in tandem. We will later take a closer look at modeling, and then at coding, especially with R. The data and the computer have assumed supporting roles for both modeling and coding - they're a given. The story is a means to an end (the solution)[7].

[6]This is a tricky bit of the process: strictly speaking, there is no coding without modeling, which is why the coding activity is generally ignored or called "visualizing" (as if creating visuals were the main, or only coding activity worth pursuing). Code requires a programming language and a system, it is often carried out in a shell using an interpreter, and every function has an underlying model. The ambiguity is expressed here by modeling both tasks as collapsed subprocesses.

[7]There always is some story (in the sense of an UML-type use case or an agile user story), but that fact, and the necessary processes to establish and communicate it, are part of a specific methodology. The term "data-driven storytelling" is used to capture different

Another important aspect that we did not mention before: coding is not a solitary activity but usually involves community. In the case of R, this is especially important. The community that drives the development of R is a major "selling" point for adopting the language in the first place.

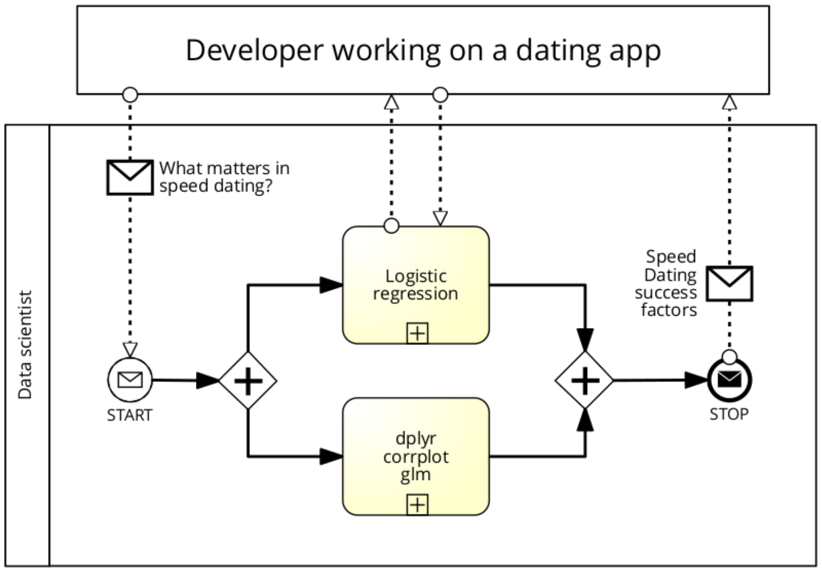
Example: text mining



Student writing about Jane Austen's novels
Data scientist
What are the most common words?
START
Tidy and tokenize
janeaustenr tidytext ggplot2
Most common words
STOP

methodologies for creating stories based on data. It relies strongly on visualization and thereby, once again, bypasses the crucial issue of problem and solution as necessary start and end point of the data science process.

Example: speed dating


Developer working on a dating app
Data scientist
What matters in speed dating?
START
Logistic regression
dplyr corrplot glm
Speed Dating success factors
STOP

Summary

- "Data magic" vs. "decision intelligence"
- Data/solitary vs. Client/collaborative
- Text mining and recommender systems
- Modeling/coding as collaborative activities

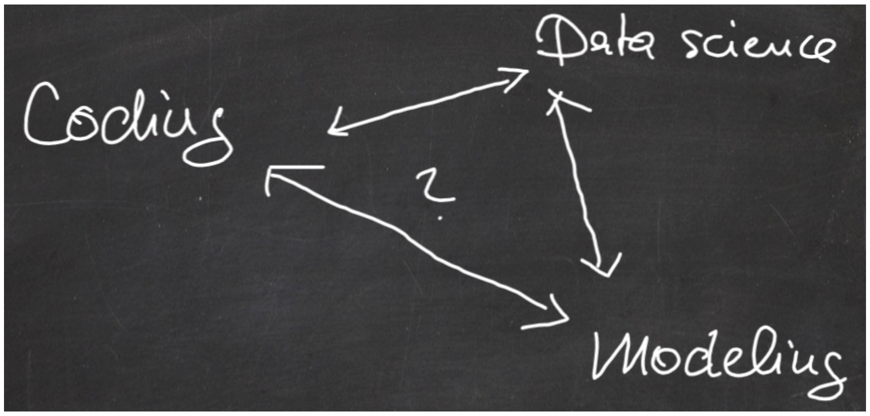
  NOTES:

Let's take a breather: what have we seen so far?

- One way of looking at data science stays within a narrow corridor of meaning as a kind of "l'art pour l'art" activity. I called this the "data magic" scenario.
- Another way considers client focus and collaborative work essential, in line with the general demands of digital transformation - this I called "decision intelligence"
- Problems in either scenario are equally challenging, both conceptually and technically.

- We briefly looked at text mining, a natural language processing application, and a speed dating, which could be used for a recommender system like a dating app. We'll look at these in more detail later in the "technical" part.
- The second scenario, "decision intelligence", begs the question how "modeling" and "coding" are connected, which is what we're going to look at now.

DS $\leftarrow$ modeling $\cup$ coding



Coding skills

- Algorithmic thinking
- Fixed at run-time
- Procedure oriented
- Depends on data structures
- Object-oriented programming (OOP)
- Functional Programming (FP)

Notes:

- We can imagine "coding" and "modeling" as skill **dimensions**.
- Coding requires primarily **algorithmic** thinking - thinking in fixed procedures that are suited to be executed by machines. Algorithms can usually not be changed at run-time. They take different form based on the input variable values. Their form and their success for a given problem critically depends on available data structures and programming paradigms - e.g. Object-Oriented Programming. Coding and programming are ways to design **procedures** to be executed by a machine.

Modeling skills

- Heuristic thinking
- Adaptive at run-time
- Pattern oriented
- Cognitive bias
- Analogy, induction
- Petri nets, BPMN, UML

Notes:

By contrast, modeling requires primarily **heuristic** thinking. It is adaptive at run-time. When it is formalized, it is informed mostly by mathematics (however, see UML diagrams, BPMN diagrams from Petri nets - direct graphs with places and transitions, or events and flow between them)[8].

[8]One formalism, yes, but modeling comes in many more forms than mathematical models. Partly because it is hard to imagine mathematical models, and partly probably because of laziness or ignorance, many process modeling standards exist. Earlier, we saw Business Process Model Notation (BPMN), a graphical notation that is derived from Petri nets (a true mathematical modelling language for the description of distributed systems - mathematically, Petri nets are directed bipartite graphs that consist of places and transitions, or events and flow between them). In IT and programming, Unified Modeling Language (UML) is the most common graphic notation - but many aspects of BPMN and UML do not have an exact mathematical equivalent.

Heuristics works via **patterns**, a form of "procedures with wiggle room". There is a subjectivity to heuristics, which is hard to put into mathematics, except for simple logic problems, and just as hard to put into words[9]. Examples for such heuristics are analogy, generalization and specialization, decomposition and recombination, induction and diagrammatic reasoning ("draw the problem"). Some of these have found their way into graphical notations (e.g. UML class diagrams).

Four scenarios

- Data Visualization
- Application Building
- Numerical Computation
- Machine Learning

Notes:

Between coding and modeling, we can define four different scenarios:

1. LOW MODELING / LOW CODING: Data Visualization. The purpose is to make data structures visible. This is the domain of Data Scientists. We'll see some examples in R later on.

2. LOW MODELING / HIGH CODING: Application building. This is the domain of software developers, and this is what most computer scientists are either groomed for, or find themselves doing at the end of their very long days and nights.

3. HIGH MODELING / LOW CODING: numerical computation depends on formulas. Sounds traditional, but it isn't. For example, modeling and data mining gave rise to a completely new discipline, process mining, less than 10 years ago.

4. HIGH MODELING / HIGH CODING: this is the end game, where many interesting problems are at home.

[9]It's not impossible though. The best book I know about this issue is "How to solve it" by George Polya (1945). It contains a set of heuristics. See also "The Model Thinker" by Page (2018).

There is a lot of potential **mathematical complexity** at home here - it's not all probability and statistics. E.g. category theory (describing generalized functions as functors in deep learning), or group theory (addressing the curse of dimensionality[10] with the Renormalization Group known from particle physics).

The **computational complexity** is e.g. related to parallelization, and to highly distributed operations, or messy or unstructured data (e.g. graph databases). This is mainly because the size of the problems do not scale linearly with the number of parameters or dimensions, but often exponentially.

Programming languages

- Apps: Java, C++
- Numerics: FORTRAN, C, Octave
- DataViz: R / Python, Snap!, Processing
- Machine Learning: Python / R, Julia

Notes:

You know now that there are many, many programming languages. But of course, they are not all equally in use. We're going to look at two of these, R and Python, later on. In general, the coding world is systemically multi-lingual.

For application building, I have listed **Java** and **C++** though both can be, and are used for maching learning, too. C++ e.g. has an edge on embedded systems in the "Internet of Things" (IoT) world, and as machine learning engines get mobile, there will be more C++ implementations. Java has some interesting libraries that allow for Natural Language Processing.

For numerical computations, FORTRAN and C are among the oldest, and most reliable, most used languages. In fact, most of R is written in C and FORTRAN. There are also newer languages here, like MATLAB or GNU Octave. These, too, are used for machine learning. GNU Octave e.g. was the

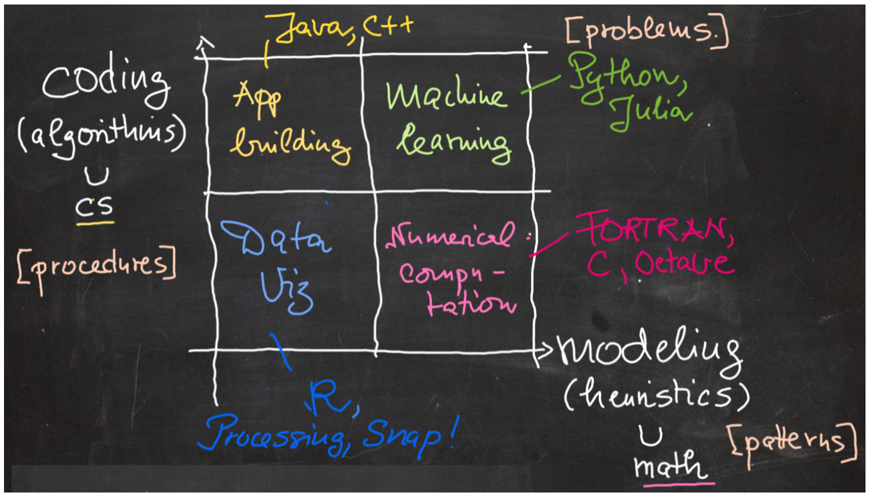
[10]When analyzing data in higher-dimensional spaces - the dimensions correspond to features in the dataset. For machine learning systems, the number of training samples to generalize a higher-dimensional model increases exponentially. So I must reduce the features (e.g. with Principal Component Analysis/PCA).

basis in Andrew Ng's seminal Stanford machine learning course (Coursera, 2008).

Data Visualization is a playground for languages that often, like Snap! (Berkeley's HTML5 implementation of MIT's block-based language Scratch, bought by SAP), or Processing (for coding within the visual arts) were created by, and for non-scientists. This trend is likely to become more important as (and if) data literacy and software literacy increase.

As I said, many if not all of these can in principle be used for machine learning: I have programmed neural nets in C++. But Python and R rule this space - partly because it is still pretty small. There are also hybrids[11], e.g. IBM Watson is a suite of machine learning tools on a Linux frame, written in Java, C++ and PROLOG ("PROgramming in LOGic", specialized on symbolic or non-numeric computation). A comparatively new kid on the block is Julia, which was designed for high performance (aka parallel) numerical computing. But also Java has developed into this direction since Java 8.

Data science framework



Notes:

[11] In 2019, IBM Watson ran on a massive parallel computing architecture of 2,880 concurrently running processor cores - by comparison, your laptop probably contains 4 cores of which only 2 are running concurrently.

There you have it - one map of many languages and data science aspects. There are two different pathways here to get from the lower left corner to the upper right: one is more traditional and goes through computer science, software development and coding, the other is more aligned with data mining and numerical computation and modeling. Of course, you don't have to go to the upper right - you can also stay put anywhere along the way. Otherwise we'll all be machine learning scientists, and how boring (and dangerous) would that be?

By the way, the reason why it's still possible to get away with drawing such diagrams is that data science is still very young. Computer science, too, of course, but CS is at least old enough to have begun to understand itself as a science. Data science is still not quite there yet and does not yet know who it's going to be when it grows up, as it were. Will it be a craft, like database management or software development? Or a theoretical science? Or something in between? This is one of the things that attracted me to it in the first place: so much is still possible!

Summary

- Coding is algorithmic, modeling is heuristic
- Paradigms: OOP/FP, BPMN/UML
- Scenarios: DataViz, Apps, Numerics, ML
- Languages: R, Python, and many more
- Two pathways: CS/coding vs. DS/modeling

Let's take a breather: what have we seen so far?

- We have defined coding vs. modeling as more algorithmic vs. heuristic, and procedure vs. pattern oriented.
- We identified major paradigms in coding and modeling, such as OOP/FP for coding, BPMN/UML for modeling.
- We introduced four scenarios with different coding/modeling ratios for: DataViz, numerics, apps, Machine Learning.

- We aligned several programming languages with these scenarios, and we found that R and Python play a key role both in Data Visualization and in Machine Learning.

Data science examples using R

`./img/gapminder.gif`

- Text mining Jane Austen
- What matters in speed dating
- Current student projects

NOTES:

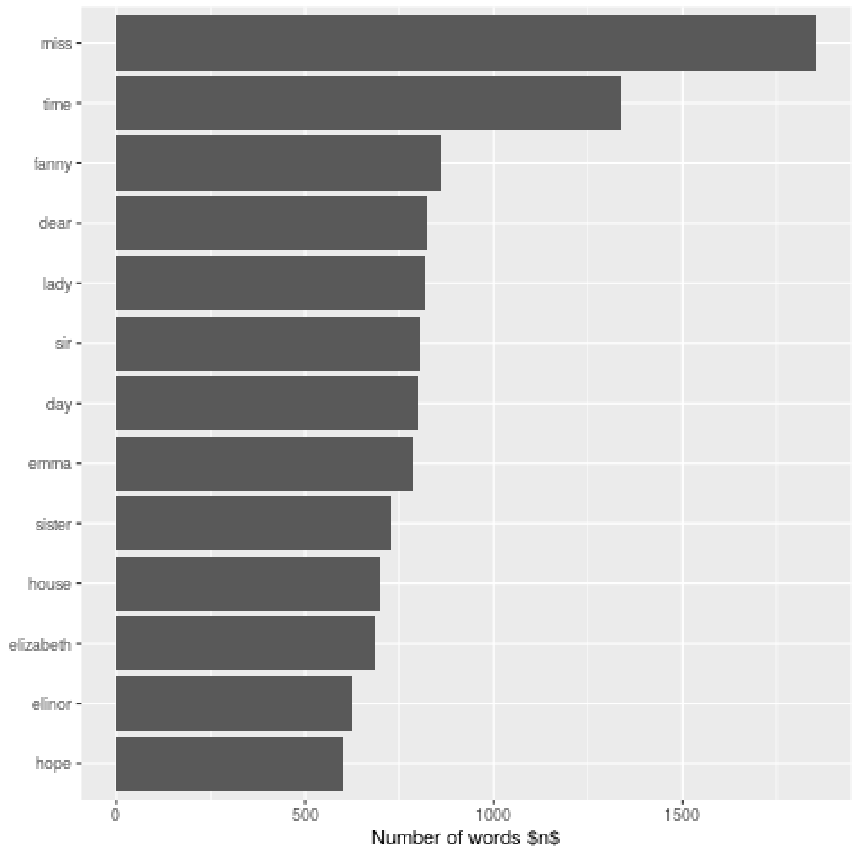
Let's look at some examples. The image shows the language part of the framework we introduced earlier. I'm going to show: (1) a text mining example with Jane Austen's novels (Silge & Robinson 2017); (2) building and running a linear regression model on speed dating data (McNulty 2020 and GitHub).

The added charm of these examples: (1) would be a beginning case in my special topics class on Natural Language Processing, and (2) would be part of a machine learning course[12], and (3) are sample projects from an introductory data science class.

---

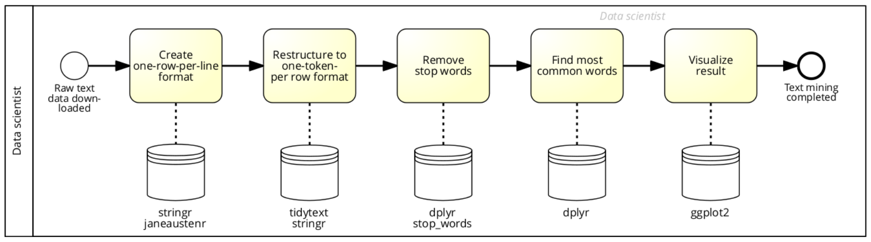
[12]Though the example is not actually machine learning, but an Exploratory Data Analysis (EDA) example. However, it illustrates how to build and run a statistical model (in this case binomial logistic regression). This could be one step when developing a recommender engine for a dating service (which is a form of digital speed dating).

Text mining Jane Austen



The most common words in Jane Austen's novels

Text mining Jane Austen: steps



BPMN diagram showing required R libraries

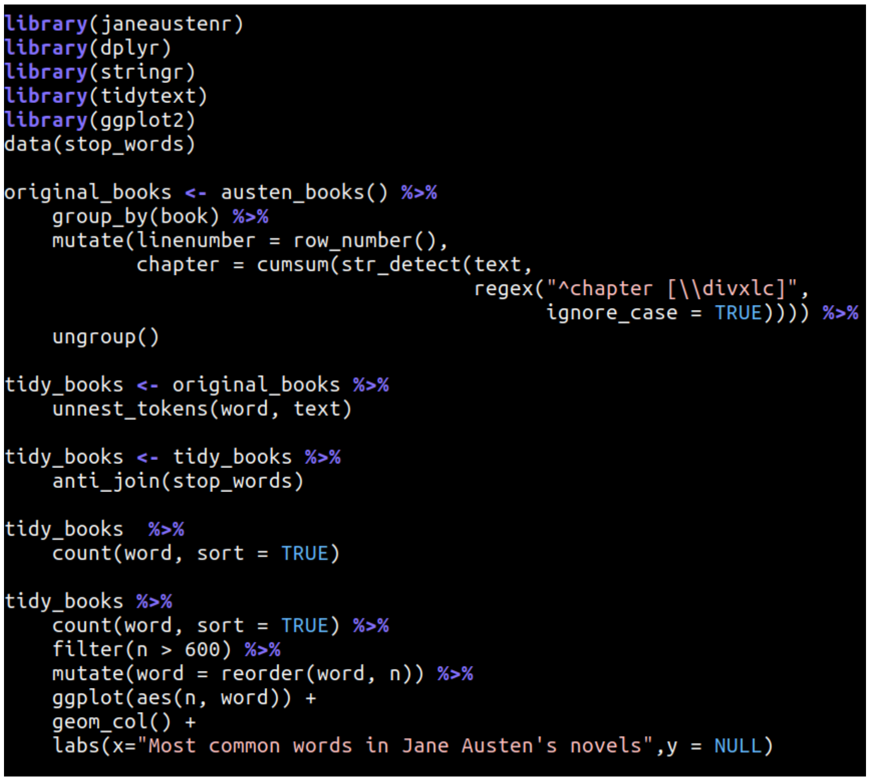
NOTES:

Steps:

Download raw textual data	`stringr`, `janeaustenr`
Create one-row-per-line format	`tidytext`, `stringr`
Restructure in one-token-per-row format	`dplyr`
Remove stop words	`stop_words`
Count and filter for minimum occurrences	`dplyr`
Create labeled barplot	`ggplot2`

R libraries are called "packages".

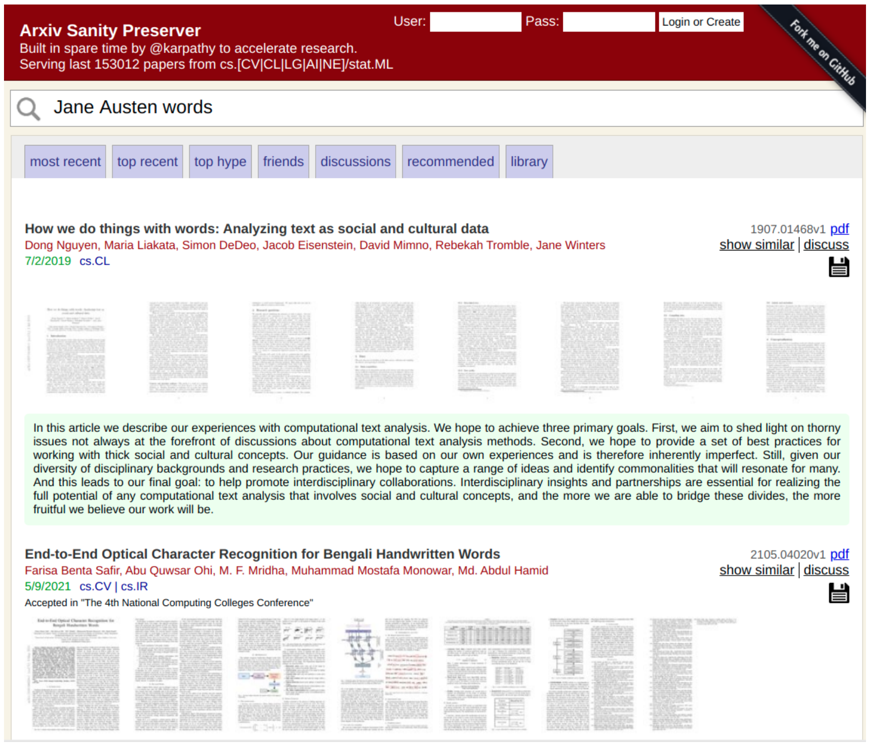
Text mining Jane Austen: code



```
library(janeaustenr)
library(dplyr)
library(stringr)
library(tidytext)
library(ggplot2)
data(stop_words)

original_books <- austen_books() %>%
    group_by(book) %>%
    mutate(linenumber = row_number(),
           chapter = cumsum(str_detect(text,
                                       regex("^chapter [\\divxlc]",
                                             ignore_case = TRUE)))) %>%
    ungroup()

tidy_books <- original_books %>%
    unnest_tokens(word, text)

tidy_books <- tidy_books %>%
    anti_join(stop_words)

tidy_books  %>%
    count(word, sort = TRUE)

tidy_books %>%
    count(word, sort = TRUE) %>%
    filter(n > 600) %>%
    mutate(word = reorder(word, n)) %>%
    ggplot(aes(n, word)) +
    geom_col() +
    labs(x="Most common words in Jane Austen's novels",y = NULL)
```

Source: Silge & Robinson (2017)

Real world application



Arxiv Sanity Preserver
Built in spare time by @karpathy to accelerate research.
Serving last 153012 papers from cs.[CV|CL|LG|AI|NE]/stat.ML

Jane Austen words

How we do things with words: Analyzing text as social and cultural data 1907.01468v1 pdf
Dong Nguyen, Maria Liakata, Simon DeDeo, Jacob Eisenstein, David Mimno, Rebekah Tromble, Jane Winters
7/2/2019 cs.CL

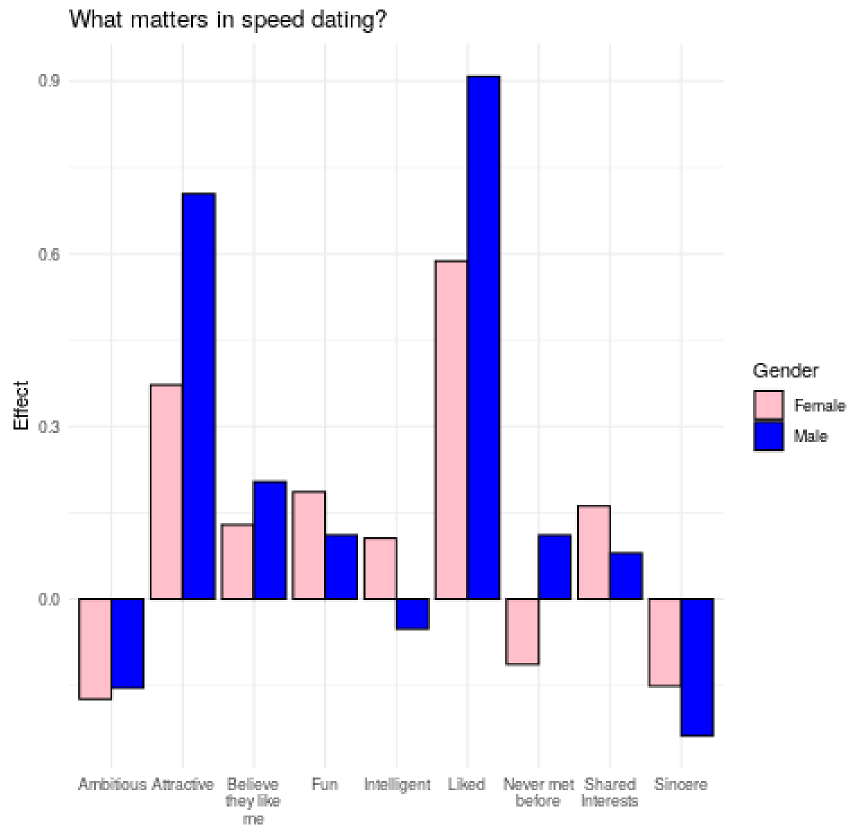
In this article we describe our experiences with computational text analysis. We hope to achieve three primary goals. First, we aim to shed light on thorny issues not always at the forefront of discussions about computational text analysis methods. Second, we hope to provide a set of best practices for working with thick social and cultural concepts. Our guidance is based on our own experiences and is therefore inherently imperfect. Still, given our diversity of disciplinary backgrounds and research practices, we hope to capture a range of ideas and identify commonalities that will resonate for many. And this leads to our final goal: to help promote interdisciplinary collaborations. Interdisciplinary insights and partnerships are essential for realizing the full potential of any computational text analysis that involves social and cultural concepts, and the more we are able to bridge these divides, the more fruitful we believe our work will be.

End-to-End Optical Character Recognition for Bengali Handwritten Words 2105.04020v1 pdf
Farisa Benta Safir, Abu Quwsar Ohi, M. F. Mridha, Muhammad Mostafa Monowar, Md. Abdul Hamid
5/9/2021 cs.CV | cs.IR
Accepted in "The 4th National Computing Colleges Conference"

Arxiv Sanity Preserver (recommender)

From the project description: "This project is a web interface that attempts to tame the overwhelming flood of papers on Arxiv. It allows researchers to keep track of recent papers, search for papers, sort papers by similarity to any paper, see recent popular papers, to add papers to a personal library, and to get personalized recommendations of (new or old) Arxiv papers."

What matters in speed dating



NOTES:

2002-2004, Columbia Univ. ran a speed-dating experiment: they tracked 21 speed dating sessions for adults meeting people of the opposite sex. In this analysis, we want to find out what it was that determined if someone viewed someone else as a match.

Because the outcome variable was binary, a single decision value - match yes/no - an binomial (logistic) regression analysis was run[13]. This analysis is used to determine, which factor has which influence when all the other values

[13]A binomial GLM (Generalized Linear Model) is used when the response variable is binary numerical data. The outcome variables can be discrete (like here) or continuous. The values given are on the log odds scale and need to be transformed using the exponential to be given on the odds ratio scale, which is why this model is called "logistic regression" or "logit". (For more, see e.g. Cox 2017 p.239.)

are held still. Also, correlation was tested to exclude that variables measure the same thing (using a correlation matrix and drawing a correlation plot).

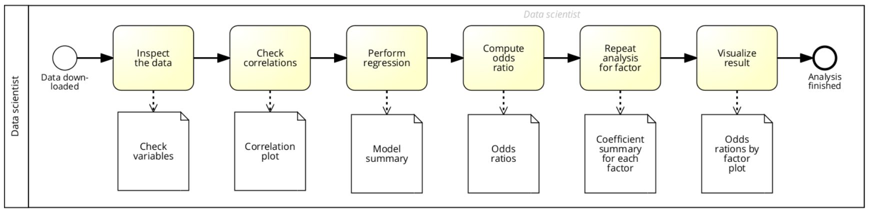
Analysis results without considering `gender` as a factor:

1. The respondents' overall rating (`like`) is the greatest indicator for a "match" decision.
2. Attractiveness (`attr`) is the primary positive indicator.
3. Sincerity (`sinc`) and ambitiousness (`amb`) **decreased** the likelihood of a match (their estimators are negative).
4. Interestingly, if the respondent believed their interest to be reciprocated (`prob`) was not decisive.

The results when comparing `gender` (see barplot):

1. Physical attractiveness matters a lot more to males.
2. Intelligence matters more to women, little to men.
3. If the speed dating partners had met has a significant effect, but we could not see it before, because it was averaging out as insignificant: men prefer new interactions, women like to see a familiar face.

What matters in speed dating: steps

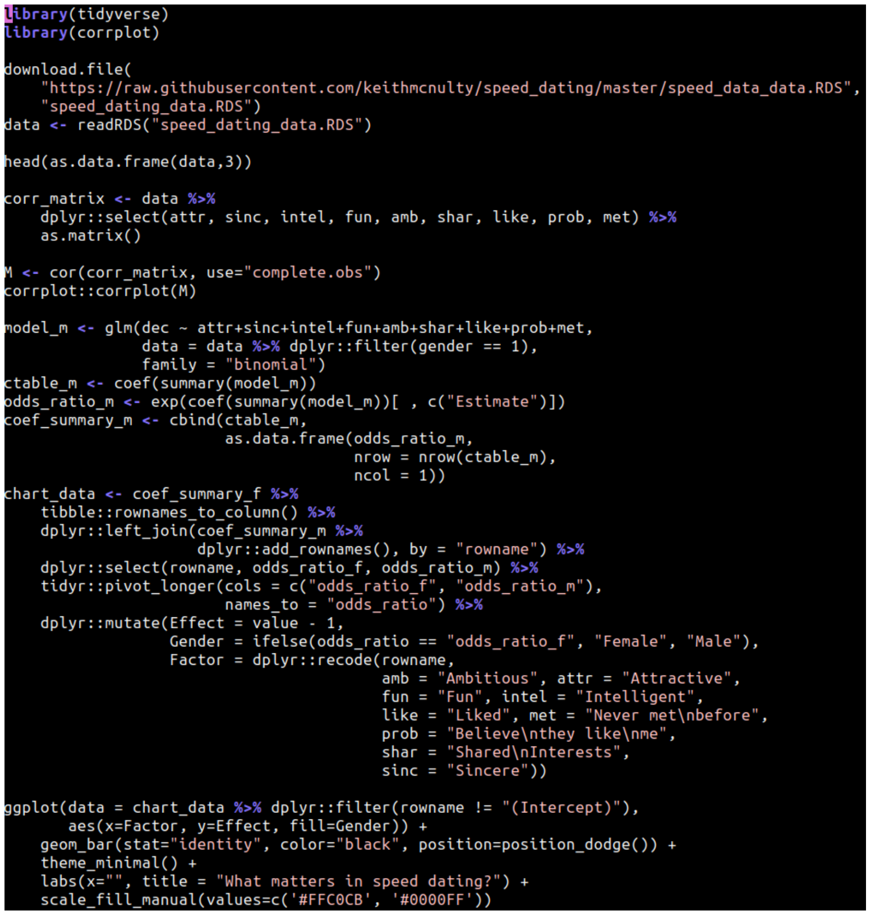


BPMN diagram showing process & output

1. Create data set for analysis - (compressed) RDS format
2. Inspect the data - look at variables' meaning and value range
3. Check correlation (plot) - needs `corrplot`

4. Perform binomial logistic regression
5. Compute regular (non-log) odds
6. Repeat analysis for factored subsets
7. Create barplot

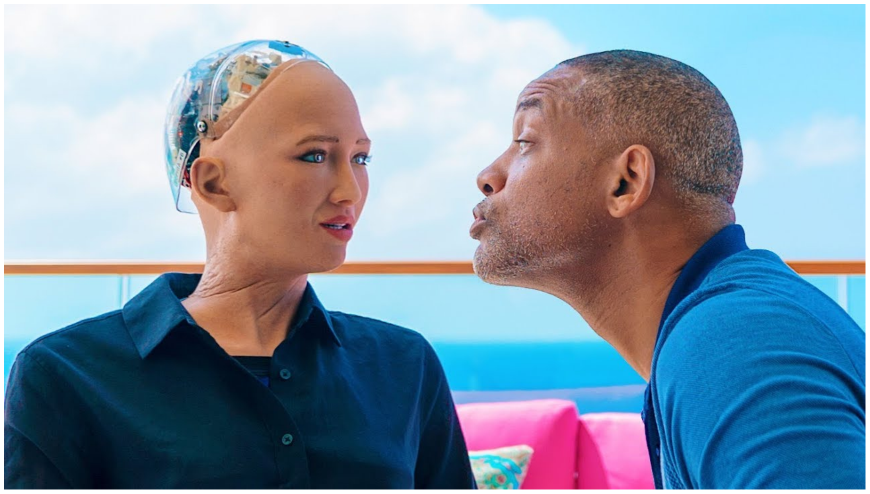
What matters in speed dating: code



```r
library(tidyverse)
library(corrplot)

download.file(
    "https://raw.githubusercontent.com/keithmcnulty/speed_dating/master/speed_data_data.RDS",
    "speed_dating_data.RDS")
data <- readRDS("speed_dating_data.RDS")

head(as.data.frame(data,3))

corr_matrix <- data %>%
    dplyr::select(attr, sinc, intel, fun, amb, shar, like, prob, met) %>%
    as.matrix()

M <- cor(corr_matrix, use="complete.obs")
corrplot::corrplot(M)

model_m <- glm(dec ~ attr+sinc+intel+fun+amb+shar+like+prob+met,
               data = data %>% dplyr::filter(gender == 1),
               family = "binomial")
ctable_m <- coef(summary(model_m))
odds_ratio_m <- exp(coef(summary(model_m))[ , c("Estimate")])
coef_summary_m <- cbind(ctable_m,
                        as.data.frame(odds_ratio_m,
                                      nrow = nrow(ctable_m),
                                      ncol = 1))
chart_data <- coef_summary_f %>%
    tibble::rownames_to_column() %>%
    dplyr::left_join(coef_summary_m %>%
                     dplyr::add_rownames(), by = "rowname") %>%
    dplyr::select(rowname, odds_ratio_f, odds_ratio_m) %>%
    tidyr::pivot_longer(cols = c("odds_ratio_f", "odds_ratio_m"),
                        names_to = "odds_ratio") %>%
    dplyr::mutate(Effect = value - 1,
                  Gender = ifelse(odds_ratio == "odds_ratio_f", "Female", "Male"),
                  Factor = dplyr::recode(rowname,
                                         amb = "Ambitious", attr = "Attractive",
                                         fun = "Fun", intel = "Intelligent",
                                         like = "Liked", met = "Never met\nbefore",
                                         prob = "Believe\nthey like\nme",
                                         shar = "Shared\nInterests",
                                         sinc = "Sincere"))

ggplot(data = chart_data %>% dplyr::filter(rowname != "(Intercept)"),
       aes(x=Factor, y=Effect, fill=Gender)) +
    geom_bar(stat="identity", color="black", position=position_dodge()) +
    theme_minimal() +
    labs(x="", title = "What matters in speed dating?") +
    scale_fill_manual(values=c('#FFC0CB', '#0000FF'))
```

Source: McNulty (2020)

Real world application



"Will Smith Tries Online Dating"
NOTES:

For a dating recommender engine, many profiles and parameters have to be correlated. This has to be done based on a model that makes decisions about which parameters have which influence. This problem - matching people - could also be used for matching jobs and profiles (using different data of course). The example also shows the critical importance of the real data set: where it came from, what exactly was measured and how, etc.

The image shows Will Smith hitting on the robot "Sophia" of Hanson Robotics.

Current student projects

- Avocado sales in different US cities
- Rental prices in German cities
- What is my Netflix consumption like?
- Popularity of different US bills
- Lifestyle habits and weight issues

- Musical consumption during a pandemic
- Influence of Elon Musk tweets on bitcoin
- Influence of Queens Gambit on Chess.com

NOTES:

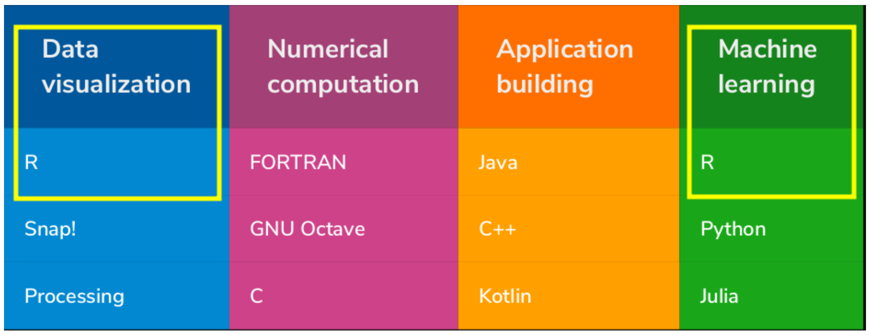
Projects:

1. Avocado **sales** in different US cities: data sets from Kaggle. Committed Avocado fans wanting to find out the correlation between Avocado prices and sales volume.
2. Rental **prices** in German cities: how do the conditions of an apartment affect the rental prices? How have prices changed over time? Students are always looking for affordable housing.
3. What is my Netflix consumption like? How has it changed over 2 years? What have I watched and when?
4. Popularity of different US bills: Checking data from The Economist about the popularity of different bills - students want to improve on a badly designed graph.
5. Exploring the relationship between lifestyle habits and weight using as variables: water consumption, physical activity, eating habits, smoking habits.
6. Musical consumption during a pandemic: using own Spotify usage data and the `spotifyr` package.
7. Influence of Elon Musk tweets on bitcoin using the `Twitter` API.
8. Influence of Queens Gambit (US mini series) on Chess.com user activity.

   Projects 1-5: Master of International Business Management, course "Introduction to Data Science" (840122).

   Projects 6-10: MA Accounting and Controlling, "Data-driven business process analysis & maching learning" (504092)

Which language?



Data visualization	Numerical computation	Application building	Machine learning
R	FORTRAN	Java	R
Snap!	GNU Octave	C++	Python
Processing	C	Kotlin	Julia

R vs. Python

`./img/RVSP/fight.gif`

- Product
- Popularity
- Prediction
- Proposal

NOTES:

I looked through a lot of surveys and articles to come up with this comparison. It's not systematic or scientific, but I think it's a fair assessment of both languages. The most interesting article on this subject is "Python vs (and) R for Data Science" by Brian Ray. It is a little dated (2018) (and slightly favors Python). Since then, R and Python have become even more similar, I would say. Here is a 2020 update that argues in the same direction: "How R Still Excels Compared to Python," by Michael Grogan.

Apparently, someone on Kaggle built a predictive model to find out whether a developer uses R or Python, and found out, among other things:

- "If you're more worried than excited about AI, then you're more likely to be an R user"
- "If you studied statistics you're more likely R, and if computer science then Python"

Product - Differences

	R [4.1.0]	**Python [3.9.5]**
Release:	1991 (1976)	2008 (1989)
Written in:	FORTRAN, C, R	C
Purpose:	Math & Statistics	Productivity
License:	GPL	PSF
Libraries:	17,634 (CRAN)	137,000 (PyPI)
Vectors:	Start at `1`	Start at `0`
Typedness:	Weak	Strong

Product - Similarities

Access:	Free	Easy to learn
Paradigms:	OOP	Functional
Memory:	Dynamical[14]	Garbage[15]
Translation:	Interpreter	REPL/shell
Analytics:	Visualization	Machine Learning
Creativity:	Packages	Portability
Interfaces:	Shiny/Django	SQL/SQLite

NOTES:

There are many more similarities than differences!

- Both R and Python are **free to use** (though with difference licenses - the Python Software Foundation License does not force you to make you modified code open source, while the GNU Public License is more strict.

[14] A programming language is dynamically-typed if the type of a variable is checked during run-time. Other languages like this are: PHP, Lisp, or JavaScript. Code is generally less optimized than in statically-checked languages, which check at compile-time - examples are: Java, C++, C or FORTRAN. The other two categories are strong-typedness (variables are bound to specific data types) vs. weak-typedness. Python, C++ and Jave are strong-typed, but R and C are weak-typed. Here is a good overview with brief examples.

[15] Garbage collection (GC) relates to memory management. Memory that is no longer needed is reclaimed to be reused. This concept was first established in Lisp. Languages that are not garbage-collecting force the programmer to think more about memory management (which is a good thing IMO). Most scripting languages (Python, R) require GC. C and C++ are designed to be used with manual memory management (but there are implementations with automatic GC. One danger e.g. in C/C++ is that pointers (a specific data structure) can be dangling - point to memory that has been freed and is already being reused, which leads to unpredictable results that may be hard to discover.

- Both are **easy to learn** though I think R is a little easier to learn, at least if you learn languages at university, in an academic environment.
- Both Python and R are **interpreted** rather than compiled.
- Both languages have a **REPL** (Read-Evaluate-Programming-Loop), the fancy word for an console, i.e. they are set up for interactive work. REPLs are so neat that even Java has one now (JShell, since Java 9).
- For **data science**, both are equally matched. They're so similar in fact that interfaces exist - you can e.g. use `ggplot` for graphics in Python, too, though it's an R package, and the `reticulate` package in R allows you to use Python.
- Library **creation** is a favorite pastime for both R and Python programmers - you can do it in parallel (though it's not as common as dual development in Android and iOS) - though perhaps R is better positioned here if you are not a programmer.
- For R, the environment could be anywhere where statistics is used and needed - psychology, linguistics, biology - any science or discipline that is naturally data rich (e.g. because it is experimental). R is more of a statistician's language.
- Python is more of a general purpose language - hence the focus on productivity. A popular text book is called "Automate the boring stuff with Python". Python is arguably more of a programmer's language.
- I knew Python before I knew R. I discovered R when I created my first introductory data science course - not for computer scientists, but for business students. These students had no difficulty picking up R. The focus away from programming and IT, the ease of creating **visualizations**, and the
- My first real programming language was FORTRAN - it's the language in which I learnt to **debug**, and so there is a natural affinity with functional programming perhaps. But the language in which I wrote most of my code as a student is **C++**, which made it easier for me to embrace Python.
- The hard (or sweet, depending on your preferred pastime, which for me includes coding and learning new programming languages) truth is: for data science you need to learn both languages. For example if

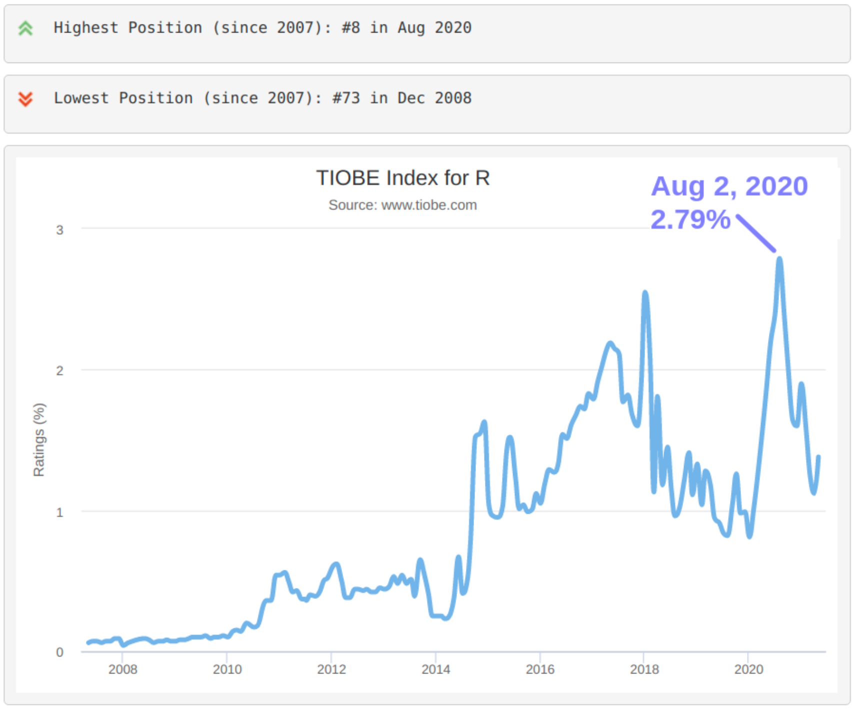
I have time in class, I will introduce the Python way as well as the R way, and many textbooks do this, too, e.g. here and here [16].

- If you already know Java or C, Python will come natural to you. If you come from MATLAB, GNU Octave or FORTRAN, R will seem more natural.
- Interfacing with other languages is another issue - most importantly, with relational databases using SQL or SQLite - both R and Python are equally good at this (though Python may be faster).
- Due to their interactive nature, the popular way to develop in both R and Python is via notebooks (like Jupyter or Colaboratory) where you can mix documentation and code, including other languages. This is nothing but the old "Literate Programming" paradigm invented by Don Knuth.

---

[16]Certainly there are more books written for R and Python in parallel than, say, for Java and C++ in parallel. In fact, Java and C++ have a similar feud, with another core issue (which language is more relevant for industrial applications). Java has no pointers and is friendler to the beginner, but C++ has a greater edge when it comes to learning algorithms and data structures. And for modern industrial parallel applications, both are currently racing each other all over again! Language wars are very popular and sort of fun (some say this is because there isn't much fun in CS otherwise but I am sure that's a myth).

Popularity - search



Highest Position (since 2007): #8 in Aug 2020

Lowest Position (since 2007): #73 in Dec 2008

Popularity - no contest

	R	**Python**
TIOBE Index	1.38%	11.74%
Loved (Dev)	44.5%	66.7%
Dreaded (Dev)	55.5%	33.3%
Wanted (Dev)	5.1%	30.0%
Salary (US)	$109k	$120k
Salary (World)	$59k	$57k

»After a consistent rise over the last five years, Python fell from second last year to third this year on the list of most loved technologies, being beat out by TypeScript. Rust held the top spot for most loved technology for the fifth year in a row.« (Stackoverflow Deverlopers Survey 2020 - 65k developers from 186 countries)

- Most dreaded languages: Virtual Basic, Objective-C, Perl
- Most loved languages: Rust, Typescript, Python

- Most wanted languages: Python, Javascript, Go

To put these numbers in perspective: most developers on Stackoverflow come from web technologies. This is not an area where R has ever had great followers. There is a web app framework ("Shiny"), which is neat, but it's limited to users of RStudio, the most popular R IDE, co-created by the creator of the "Tidyverse" bundle of packages, Hadley Wickham.

Also interesting: the loved/dreaded/wanted platforms [to develop for] - the top 10 are Linux, Docker, Kubernetes, AWS, Raspberri Pi, MacOS, MS Azure, iOS, Google Cloud Platform, Windows, Android, Arduino and Slack.

Salaries: top spots in the US (in this order) for Scala, Go, Objective-C (Apple), Kotlin, Perl, Ruby, Rust, C, Swift (Apple), Haskell, Assembly, Bash, C++, Java, Python…R. The difference between R and Python is $19k.

More meaningful for both learners and practitioners is the community support

Prediction - the war is over!

`./img/RVSP/bench.gif`

NOTES:

In a document on choosing R or Python at data analytics consulting company CAN, their senior data scientist Summers says: »To do development is to use the application and to use the application is to do development. There is no IT person and no business user. The person is both a developer and a business user. One of the reasons that larger organization have struggled to embrace Python and R is that frequently there is an organizational barrier between IT and Business.«

This is one of the smartest things about the need to know both R and Python that I've read in a long time. Business is still trying to figure out where data science fits in, which questions to ask, while IT is steaming ahead. To know where IT is going is not easy even if you've been in the business for a while as I have.

Proposal - what you should do

- `R`

- `R` + `Python`
- `R` + `Python` + `Java`
- `R` + `Python` + `Java` + `C++`
- `R` + `Python` + `Java` + `C++` + `^[\w]+$`

NOTES:

Remember how many languages there were in 1966 already? 700! There is no end in sight: hardware, software are going to change and adapt to ever-changing problems and solution requirements. Especially in young, interdisciplinary fields like computer science and data science whose potential has not yet fully been uncovered, languages are likely to change. There are demographic aspects to this, historical and cultural aspects[17]

The upshot is: your future as a computer scientist or computational mathematician, as a digital historian or librarian, or as an experimental physicist or psychologist, will be a multi-lingual future - get ready for it!

Summary

Data science:	Decision intelligence
Core skills:	Modeling and coding
Applications:	ML / EDA / NLP
R vs. Python:	R + Python

Tools

`./img/tools.gif`

- BPMN: Signavio Process Manager
- R: Emacs + ESS + Org-Mode (+ Python)

[17]E.g. demographic: people who learnt COBOL have died out. Historic: some languages are old and established, and will be harder to replace (e.g. Java). Cultural: each language fosters a particular community that often defends its language fiercely. None of these issues are 100% rational, none are 100% irrational, all of them have to be weighed and heard before the field of CS or DS as a whole can move on.

- Slides: GNU Emacs + Org-Mode + reveal.js
- Computer: Dell 7300 i7 1.9GHz (2019)
- OS: Ubuntu 18.04 LTS Linux (2018)

NOTES:

No tools were harmed in the making of this presentation! I am a self-confessed tool fanatic, and this is nothing compared to the firework of tools that I like to light in my regular classes! After all, IT is largely about gadgets and tools, too, so why not enjoy it while we can?

- Signavio Process Manager was used to create the Business Process Model and Notation (BPMN) diagrams. The Berlin-based company Signavio was bought by SAP in 2021.
- For programming, task management, presentation and everything really, I use the extensible text editor Emacs in connection with Org-mode. For R, I use the Emacs-Speaks-Statistics module.
- The slides were created in Emacs using the reveal.js framework. With Org-mode, I can also create PDF, Markdown, Open Office (WORD), $\LaTeX$ etc. on the fly, while the content will always be highly portable (as text).
- I also created one mindmap (with XMind).
- I have a Mac and a Windows laptop and several Raspberry Pi and microbits at home, but my work computer is a trusty Dell running Ubuntu.

Thank you! Questions? Comments?

`./img/sunflower.gif`
Contact: birkenkrahe@outlook.com

References

1. Cox. Translating Statistics to Make Decisions. Apress 2017.

2. Huang, Evans & Chattopadhyay. Deep Learning Without Neural Networks: Fractal-nets for Rare Event Modeling. Preprint. `doi:10.21203/rs.3.rs-86045/v1`

3. Landin. The next 700 programming languages. In: Communications of the ACM 9(3) (1966). `doi:10.1145/365230.365257`

4. McNulty. What Matters in Speed Dating? Online: towardsdatascience.com (02/14/2020)

5. Pearl & Mackenzie. The Book of Why: The New Science of Cause and Effect. New York: Basic Books (2018).

6. Page. The Model Thinker. Basic Books (2018).

7. Polya. How to solve it. Doubleday (1957).

8. Programming vs Coding - A Short Comparison Between Both. Online: GeeksforGeeks (09/08/2020)

9. R: A language and environment fo statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

10. Silge & Robinson. Text Mining with R. O'Reilly (2017). Online: tidytextmining.com

11. Wickham & Grolemund. R for Data Science: Visualize, Model, Transform, Tidy, And Import Data. O'Reilly (2016). Online:r4ds.had.co.nz (2016)]]

12. Wing. The Data Life Cycle. In: Harvard Data Science Review 1(1) (2019). doi:10.1162/99608f92.e26845b4