Introduction to Data Science with R

Björn Herder

A little bit about myself

- Data Scientist at iZettle
  - Building Machine Learning models for decision automation
- Former Data Scientist at Scania
  - Education of the organisation, creation of the Data Science stack and prototyping
- Passionate about solving problems using data and machine learning
  - Mostly working in R, Python, Scala and Spark
- Skier, runner and (aspiring) golfer

Overview

- R - The Language (60 +15 min)
  - CRAN, Syntax and language specifics
  - Functional Programing
  - Exercises - Functional programing in R (15 min)
- Introduction to the Tidyverse (45 + 15 min)
  - Tidy Data
  - R for Data Science
  - Exercise - Data wrangling with tidyr (15 min)
- End-to-end workflow
  - Importing, (tidying), modelling, presenting
  - Your turn!

CRAN, Syntax and language specifics

Overview of R

- Modern implementation of the S language
  - In 2018 it was 25 years since its creation
- Is an interpreted language
  - RStudio being the most popular GUI
- Multi Paradigm:
  - **Array**, **functional**, object-oriented, procedural, imperative, reflective
- CRAN - The Comprehensive R Archive Network
  - Over 13.000 packages available
- Used within both Business and Academia
- Interfaces to other languages include:
  - C, C++, Java, Python, Tensorflow, JavaScript, with many more.

My thoughts about using R

- Excels in being an interactive tool for data analysis
- In comparison to the *Zen of Python* where it's stated that there should be **one** obvious way to solve a problem, R is more flexible and providing multiple ways to interact with objects and functions.
- In accordance to the *Zen of Python* explicit is prefered over implicit. Here R is much more implicit and **often** does what you want.
- RStudio, the tidyverse and CRAN are the most important ingredients in making R as good as it is.
- R is forgiving when it comes to casting between types and seldoms warns or throws errors. This is both a strength and a common source of logical errors.
- R includes many programming paradigms, including over three object oriented systems, and by learning the language one can get many skills that are transferable to other languages and frameworks.

CRAN - The Comprehensive R Archive Network

- Repository for packages written for use in R and precompiled binaries of R distributions
- The Task Views are a good starting point when searching for packages
    - https://cran.r-project.org/web/views/
- Packages in CRAN are followed by both documentation and often an Vignette detailing intended usage of packages
- Tips:
    - There are often overlap in what is included in the packages, some packages are significantly faster, and it's worthwhile to compare the ones listed in the different views
        - An example ist he DBSCAN implementation in the package dbscan is much faster than the implementation in the package fpc
    - Vignettes are often easier to follow than the reference manual
    - The Task Views are good place to look for finding new ways of tackling problems

Syntax fundamentals

- Scoping in R
  - R uses lexical scoping
  - Values are looked up based on how the function was nested at creation
- The assignment operators:
  - <- and = Assigns to the environment in which they are evaluated
  - <- is prefered for assignment, = and <- behave differently in function calls
  - <<- Can be used to reach a parent environment (to allow mutable state)
- Indexes start at one
  - Compared to 0 in python and (many) other languages
- Sequences can be created with 1:n
  - 1:3 gives a vector with values 1,2,3
  - seq() is also a useful function
- Subsetting is done with brackets
  - vector[1]
  - matrix[1,2]
  - array[1,2,3, ...]
  - dataframe[1,] (An empty indicates selection of all elements in that dimension)
- c() is the generic combination function
  - Combines values to either vector or list
  - cbind and rbind are useful for
- The class() can be used to determine the class of an object
  - R equivalent of type() in Python

Data Structures

- R has no scalar (0-dimensional) type
  - Individual numbers/strings are vectors of length one
- Homogenous data structures include:
  - Atomic vectors (1-dimension)
  - Matrix (2-dimensions)
  - Array (n-dimensions)
- Heterogeneous data structures include:
  - List (1-dimension)
  - Data Frame (2-dimensions), compare to pandas DataFrame

- A data frame is a list of vectors
  - df <- data.frame(
    x = 1:3,
    y = c("a", "b", "c"))
- Functions are first-class citizens
  - Functions can be saved as items in a list and passed to other functions
- Attributes - Factors
  - All objects in R can be assigned metadata
  - Factors is one of the most used - simplifies work with categorical data
    - Factors is a common source of errors when working with strings

Functional programing in R

What is a function?

A function is a mapping from a space X to a space Y

Functional programing

- Programming paradigm that “treats computations as the evaluation of mathematical functions” - Wikipedia
- Benefits of the paradigm include:
  - Modularity
  - Composability
  - Avoiding state and mutability
  - Simplifies parallelization of operations

- Concepts we will cover are:
- Higher Order Functions (HOF)
- Anonymous functions
- Closures
- Partial application
- List of functions
- split-apply-combine pattern

The Pipe Operator

- Used to chain functions together
- The output on the LHS is given as first argument to the function on the right hand side
- Example :

  sum(c(1,2,3)) becomes c(1,2,3) %>% sum with the pipe

- Makes nested function calls easier to read

Higher Order Functions in R

- Map
  - Applies a function to each element of a list/array
- Reduce
  - Combines element of a list using a supplied combination function (an example is a sum)
  - Sometimes called fold in other languages
- Filter
  - Selects elements given by a statement that evaluates to a boolean
- Find
  - Like filter but only return the first found element
- Position
  - Finds the position of the element that would be found using find
- Negate
  - Inverts a predicate
- Other HOF:s are included in the purrr package

Anonymous functions

- An anonymous function is a function that is not bound to an identifier
- Useful for short functions such as specific data manipulation
- Useful in combination with HOF:s
- Example:
  - 1:5 %>% (function(x) x*x) %>% reduce((function(x,y) x + y))

Closures

- Functions that return functions
- Example:
  - closure <- function(x) {function(y) x+y}
  - plus_two <- closure(2)
  - plus_two(1) gives 3

Partial Application

- In partial application the some of the variables in a function called are fixed and the result is a new function with fever inputs
- Useful when passing functions to other functions
- The equivalent in OOP is to have objects with a state

split-apply-combine

- The pattern involves
  - Splitting on identifier or statement
  - Applying a transformation/function
  - Combining the results
- Is the pattern behind MapReduce
- Example - compute mean of each column
  - mtcars %>% map(function(x) mean(x)) %>% data.frame

Vectorized operations

- Since R uses vectors as it basic data type multiplication is in fact element wise multiplication
  - c(1,2,3,4) * c(1,2,3,4) gives c(1, 4, 9, 16)
- The same is true for addition as well as most functions in R
- Examples:
  - c(1,2,3,4) + 1
  - c(1,2,3,4) + c(1, 2)
  - c(1,2,3,4) + c(1, 2, 3)
  - c(1,2) + c(1, 2, 3)
- The results can sometimes be unexpected, especially for vectors of different lengths

Notes:

- Not all functions are vectorized, however they can be converted to vectorized functions using the function Vectorize()
- It's almost never necessary to use for loops in R
- Using it correctly will give faster and less verbose code

Exercises

Introduction to the tidyverse

“Happy families are all alike; every unhappy family is unhappy in its own way.”
Leo Tolstoy

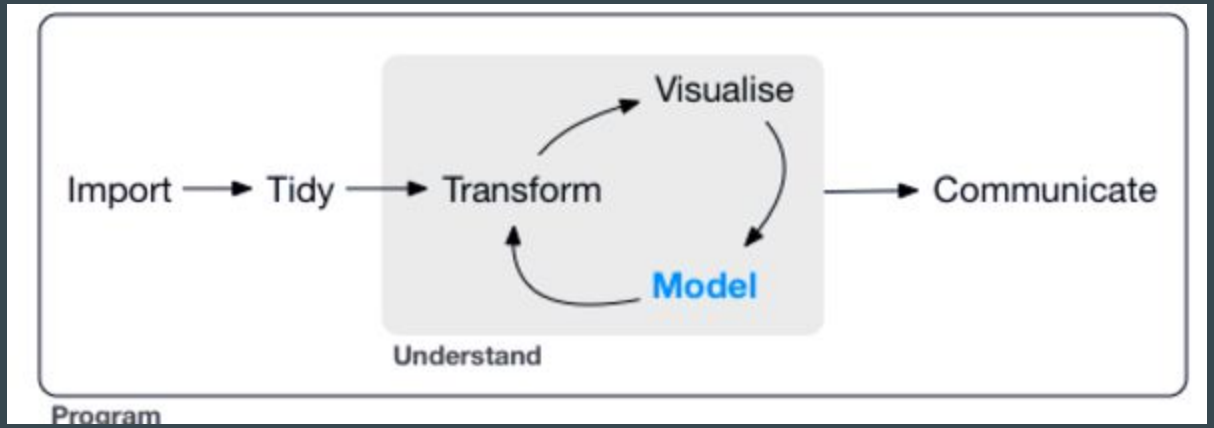
The Concept of tidy data

- Tidy data is data that is:
  - Each variable forms a column
  - Each observation forms a row
  - Each type of observational unit forms a table
- All data that does not fulfill the tidy data criteria are called untidy
- Data in untidy format can sometimes be useful for interpretation
  - Transform to untidy when presenting the data but not between function calls

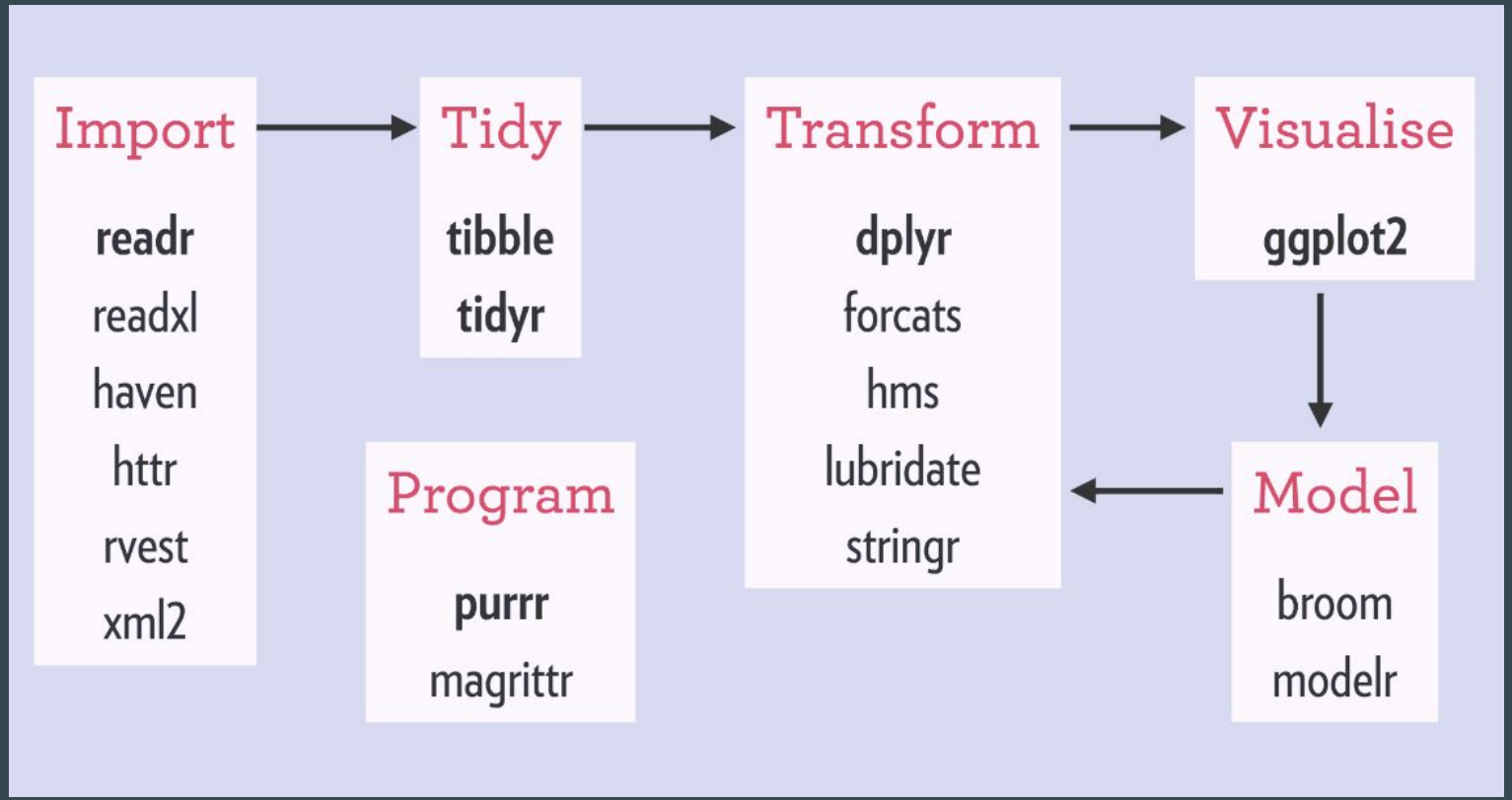
- Example of tidy data:

```
> mtcars
                   mpg cyl  disp  hp
Mazda RX4         21.0   6 160.0 110
Mazda RX4 Wag     21.0   6 160.0 110
Datsun 710        22.8   4 108.0  93
Hornet 4 Drive    21.4   6 258.0 110
Hornet Sportabout 18.7   8 360.0 175
Valiant           18.1   6 225.0 105
Duster 360        14.3   8 360.0 245
```

Overview of the tidy verse



Overview of the tidy verse



Non-Standard Evaluation - NSE

- In R some functions allows two types of syntax to the same result
- An example of this is the loading of libraries
  - library(“tidyverse”)
  - library(tidyverse)
- Or subsetting a data frame using the subset function
  - subset(df, column_1 >3)
- These functions can be written using the quote() and eval() functions
  - Allows for building functions with lazy evaluation
- Non-Standard Evaluation is used throughout the tidyverse
- Saves on typing as quotes are not needed when selecting columns
- Example
  - mtcars %>% select(mpg) %>% summarise_all(mean)

dplyr - Grammar of data manipulation

- filter()
  - Select rows based on their values.
- arrange()
  - Reorder rows
- select()
  - Select columns based on their names
- mutate()
  - Add new columns that are functions of existing variables

- group_by()
  - Groups variables in SQL like fashion
- summarise()
  - Reduces multiple values down to a single summary
  - Used in conjunction with group_by()
- These expression are chained together using the pipe operator %>%

ggplot2 - Grammar of Graphics

- Assumes data is tidy
- Works by providing standard ways of plotting based on commands
    - Plots are customized by overriding the standard behaviour
- ggplot plots are built with layers where a layer combines data, aesthetic mapping, a geom, a stat, and a position adjustment.

- Geoms - geometric object
    - line plots, histograms etc.
- Stats - statistical transformation
    - log transform, scaling, centering etc.
- Annotations
    - add text to plots
- Position adjustment
    - defines how to solve overlapping geoms

Exercises

References and Additional Resources

- The material in this course is mainly inspired by four books: *R for Data Science*, *Advanced R*, *Efficient R programming* and *Applied Predictive Modelling* (see caret)
- *R for Data Science* gives a good foundation for working with the tidyverse and related concepts
- *Advanced R* digs deeper into how R works under the hood and is a good reference for those wanting to get a deeper understanding of the language
- *Efficient R programming* is a good book regarding being an efficient programmer in R as well as other languages

Other good resources are:

- Google's style guide
- Views at CRAN
- The caret package
- RStudio
- tidyverse
- dplyr
- ggplot2 reference
- sparklyr - Spark API with dplyr grammar
- shiny - interactive web applications
- leaflet - API to the leaflet JS library
- The tidy data paper
- caret package - library for modelling

Extra

Formulas

- Used to denote variables and target variable
  - y ~ x
- Used extensively throughout R
- Can also be used in Spark ML

- Examples
  - as.formula("Species ~ Petal.Length")
  - as.formula("Species ~ .")
  - as.formula("Species ~ . -Petal.Length")
  - Fitting a linear model:
    - lm(mpg ~ cyl, mtcars)