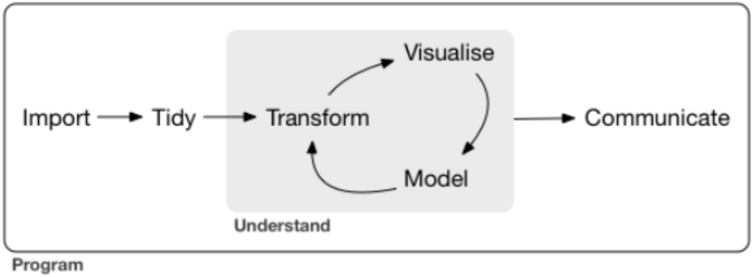
Stat 260, Lecture 13, Review

Brad McNeney

Topics



- We didn't cover the most important topic: communicate

Cheatsheets

- visualization [https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf]
- data import and tidy [https://github.com/rstudio/cheatsheets/raw/master/data-import.pdf]
- relational data and data transformation [https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf]
- strings [https://github.com/rstudio/cheatsheets/raw/master/strings.pdf]
- factors [https://github.com/rstudio/cheatsheets/raw/master/factors.pdf]
- dates [https://github.com/rstudio/cheatsheets/raw/master/lubridate.pdf]
- base R [http://github.com/rstudio/cheatsheets/raw/master/base-r.pdf]
- advanced R [https://www.rstudio.com/wp-content/uploads/2016/02/advancedR.pdf]
- iteration [https://github.com/rstudio/cheatsheets/raw/master/purrr.pdf]

Visualization with `ggplot2`

- Build a plot with `ggplot` layers:
  - Start with `ggplot()` to specify default data and aesthetic mapping of `x`, y, color, shapes, etc.
  - Add `geom_()`'s, which can override default data and mapping.
  - `stat_()`'s calculate statistical summaries such as smooths that we can add. Rather than use the `stat_()`s directly, we tended to add summaries with built-in geoms, such as `geom_smooth()`.
  - Faceting builds multiple plots by values of a faceting variable.
- Advanced topics we did not emphasize include:
  - Position adjustment such as jitter or dodge to avoid overplotting.
  - Scales specify the mapping of data to what we see on the plot.
  - Coordinate system can be changed from Cartesian to, e.g., polar.
  - Themes change the overall look-and-feel of plots
- Visualization is often cookbook

Data import and tidy

- Import with the `read_` functions, such as `read_csv()`.
  - remember `skip` and `comment` arguments
  - `read_` functions guess at how to parse columns of input and use `parse_` functions.
  - Best bet is to specify column types with `col_types=cols()`.
- `tibbles` are an improved version of the R `data.frame`.
  - Implemented as lists, so subset with `[` and extract elements with `[[`
- Use `dplyr`'s five key verbs to wrangle:
  1. `filter()` to select subsets of observations
  2. `arrange()` to reorder rows
  3. `select()` to select variables (remember helper functions like `starts_with()`, `ends_with()` and `contains()`)
  4. `mutate()` to create new variables from existing ones, and
  5. `summarize()` to calculate summary statistics (useful with `group_by()` to do split-apply-combine)

Tidy Data

- In a tidy dataset,
  - each variable has its own column,
  - each observation has its own row, and
  - each value has its own cell.
- Use `gather()` to make “wide” data “tall” and `spread()` to make “tall” data “wide”.

Relational data: multiple tables

- Modern data comes in multiple tables, called relational data.
- Keys are variables present in two tables that can be used to join them.
- The most common type of join is a “mutating join”, such as a `left_join()` or `inner_join()`.
- `semi_join()` can be used for a “filtering join” in which we filter one table based on characteristics of another.

Working with strings

- Fixed, or literal strings, like `fish`:
  - count the number of characters in a string
  - detect (yes/no) or find (starting position) substrings
  - extract and substitute substrings
  - split and combine strings
- Regular expressions specify string patterns, like `f[aeiou]sh`:
  - detect, find, extract and substitute
- Use tools from the `stringr` package

Factors

- Factors are categorical variables, implemented as an integer vector with levels.
- The `forcats` package provides tools for working with factor levels.
- Use `fct_recode()` to rename or collapse factor levels.
- Use `fct_relevel()` to partially or completely re-order a factor's levels.
- Use `fct_reorder()` to reorder levels by a second variable.

Dates and Times

- Moments in time can be dates, times, or date-times.
- The `lubridate` package contains functions to coerce strings to date, time, or date-time objects:
  - `ymd()` to coerce data in year-month-date, `mdy()` to coerce data in month-day-year, `ymd_hm()` to coerce data in year-month-date-hour-minute, etc.
- `make_datetime()` makes a date-time object from components.
- `hour()`, `minute()`, etc. extract components.
- Time data includes time zone. To set a time zone with the lubridate time functions, use the `tz` argument.
- Easy to summarize and plot date-time objects.

Pipes and functions

- The forward pipe %>% is useful for combining a linear sequence of data processing steps, when we won't need the intermediate steps.
- Encapsuling code in a function has several advantages:
  - can be used multiple times on different inputs
  - can compartmentalize computations and give them a name
- We discussed when to write a function and the components of a function:
  - the code inside the function, or `body`,
  - the list of arguments to the function, and
  - a data structure called an `environment` inside the function
- Generic functions behave differently depending on the class of input.

Vectors and iteration

- Vectors can be either atomic or list
  - The elements of an atomic vector must be the **same** type.
  - Lists can be comprised of **multiple** data types
- Use `vector()` to create an empty vector, or `c()` and `list()` to construct from data.
  - vector elements can be named
- Subset with `[` or by name.
- Extract individual elements with `[[`, or `$` for named objects
- Combine subsetting and assignment to change the value of vectors
- Iterate over a vector with a `for()` loop, `lapply()` or `map()` functions
  - Remember shortcuts for specifying a function to use with a `map()` function.