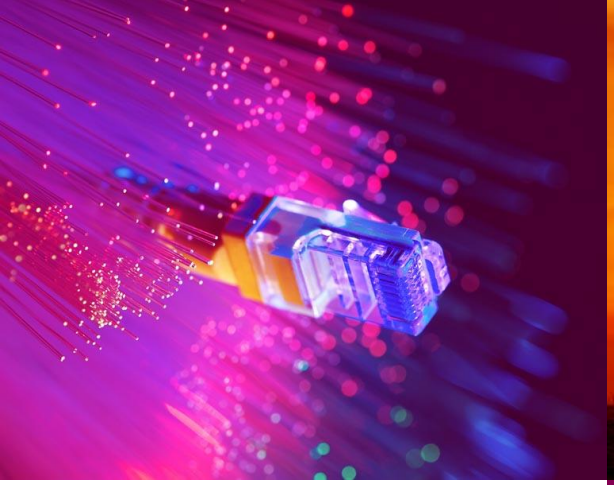


Data visualization using RStudio and ggplot

Bioweek 2018, 2018-02-06

CSC – Finnish research, education, culture and public administration ICT knowledge center

Foreword

This document is intended to go together with a course day with roughly the same name. Therefore, it is not self-supporting: it serves as reminder to both the teacher and the course participants of what will be or what was talked about. Most of the actual content is somewhere else: in the R documentation, cheat sheets, examples shown during the course etc. and most importantly in the "R for data science" book, by Garrett Grolemund and Hadley Wickham, available online.

So if you are reading this to learn on your own, treat it as a roadmap, not as study material. These things were deemed important enough to mention, and they were mentioned in this order, but you will need another source to actually study.

Which RStudio will you be using?

- The desktop version installed on the classroom computers
  - either on Linux or Windows
- The desktop version installed on your own laptop
- The server version on notebooks.csc.fi
  - using either your own laptop or the classroom computer

Getting started for the course is slightly different in all these cases, but the end experience is the same.

Getting the course material

- All of the material is available as a Github repository at https://github.com/csc-training/wrangling-with-R
  - If your RStudio knows about git, **clone the repo to a new RStudio project** (and if *you* know git, you might consider forking the repo and cloning your fork instead)
  - If your RStudio does not know about git, **manually download the repo as a zip, unzip it, and create a new RStudio project in the folder**
- The r4ds book is at http://r4ds.had.co.nz/
- Cheat sheets at https://www.rstudio.com/resources/cheatsheets/

RStudio interface

- Please also look at the cheat sheet and the keyboard shortcuts!
- Four panes:
  - top left: Scripts and data views
  - bottom left: Console
  - top right: Environment and history, data import wizard (and git)
  - bottom right: plots, help, file viewer, package viewer
- Scripts:
  - this is where you should preferably write your code, then run/source it, and not directly to the console!
  - plenty of nice editing tricks in the Edit and Code menus

RStudio interface

- You can install and load packages, and access their vignettes and other documentation, on the **Packages** tab
  - you can still also install and load them using the code commands
  - in fact, you might want to put all the *library(dplyr)* etc. commands at the start of your script

RStudio projects

- **Projects** will keep you organized when you need to switch from one context to another. See the R4ds chapter.
- A project corresponds to a folder (also the working directory)
  - and/or a version control repository, an R package in development etc
  - when you create a new project, you choose either a new or existing folder, where the .Rproj file then be created
- When you close a project, the currently open script tabs etc. will be saved, and restored when the project is reopened
  - … but not loaded packages, since the R session is restarted every time
  - the environment and history are saved to and loaded from the .RData and .Rhistory files just like when using only "base" R

Data import wizard

- The wizard opens a dialogue where you can easily browse for the file, or paste the URL, of the CSV you want to import
  - Instead of CSV you can (maybe?) also import some proprietary file formats
- You can make several choices on the wizard, about delimiters, encoding, etc.
  - choices match to the arguments of *readr::read_csv* function
  - things like row names will be a problem
- the wizard creates a piece of code that will actually do the importing
  - you might want to save, edit and rerun it in your script

Visualisation

3.2.2 Creating a ggplot

- `ggplot(data = mpg) +`
  `geom_point(mapping = aes(x = displ, y = hwy))`
- "Hey ggplot! I want to make a plot with this dataset mpg
  and
  I want there to be points with displ as x-coordinate and hwy as y-coordinate."

3.2.3 A graphing template

- `ggplot(data = <DATA>) +`
  `<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))`
- "Hey ggplot! I want to make a plot with this dataset <DATA>
  and
  I want there to be a <GEOM> with such and such
  <MAPPINGS>"

3.2.4 Exercises

1. Run ggplot(data = mpg). What do you see?
2. How many rows are in mpg? How many columns?
3. What does the drv variable describe? Read the help for ?mpg to find out.
4. Make a scatterplot of hwy vs cyl.
5. What happens if you make a scatterplot of class vs drv? Why is the plot not useful?

3.3 Aesthetic mapping

- ```
  ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy,
  color = class))
  ```
- "Hey ggplot! I want to make a plot with this dataset mpg
  and
  I want there to be points with displ as x-coordinate and hwy as y-coordinate and **class as color**."
- There's also size, shape, alpha...
- NB: "and they should be blue" is not an aesthetic mapping!
  ```

3.3.1 Exercises

1. What's gone wrong with this code? Why are the points not blue?
   ```
   ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
   ```
2. Which variables in mpg are categorical? Which variables are continuous? (Hint: type ?mpg to read the documentation for the dataset). How can you see this information when you run mpg?
3. Map a continuous variable to color, size, and shape. How do these aesthetics behave differently for categorical vs. continuous variables?
4. What happens if you map the same variable to multiple aesthetics?
5. What does the stroke aesthetic do? What shapes does it work with? (Hint: use ?geom_point)
6. What happens if you map an aesthetic to something other than a variable name, like aes(colour = displ < 5)?

- Suggested order/importance: 1 through 3

3.5 Facets

- ```
  ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
  ```
- "Hey ggplot! I want to make a plot with this dataset mpg
  and
  I want there to be points with displ as x-coordinate and hwy as y-coordinate.
  and
  they should be **faceted by class**, wrapping on two rows."
  ```

3.5.1 Exercises

1. What happens if you facet on a continuous variable?
2. What do the empty cells in plot with facet_grid(drv ~ cyl) mean? How do they relate to this plot?
   ```
   ggplot(data = mpg) + geom_point(mapping = aes(x = drv, y = cyl))
   ```
3. What plots does the following code make? What does . do?
   - ```
     ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy)) + facet_grid(drv ~ .)
     ```
   - ```
     ggplot(data = mpg) + geom_point(mapping = aes(x = displ, y = hwy)) + facet_grid(. ~ cyl)
     ```

- Suggested order/importance: these three first, you can look at the rest too if you have time.

3.6 Geometric objects

- `ggplot(data = mpg) +`
  `geom_smooth(mapping = aes(x = displ, y = hwy))`
- "Hey ggplot! I want to make a plot with this dataset mpg
  and
  I want there to be **a smoothing curve** with displ as x-coordinate and hwy as y-coordinate..."
- `ggplot(data = mpg) +`
  `geom_smooth(mapping = aes(x = displ, y = hwy, linetype = drv))`
- "...and a different curve per drv, marked by linetype."

3.6 Geometric objects (contd.)

- ```
  ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
  ```
- "Hey ggplot! I want to make a plot **with this dataset mpg with displ as x-coordinate and hwy as y-coordinate**
  and I want there to be points
  and I want there to be a smoothing curve."
  ```

3.6.1 Exercises

1. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart?
2. Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions.
   ```
   ggplot(data = mpg, mapping = aes(x = displ, y = hwy, color = drv)) + geom_point() + geom_smooth(se = FALSE)
   ```
3. What does show.legend = FALSE do? What happens if you remove it? Why do you think I used it earlier in the chapter?
4. What does the se argument to geom_smooth() do?
5. Will these two graphs look different? Why/why not? [see codes in the book]
6. **Recreate the R code necessary to generate the following graphs. [see graphs in the book]**

- Suggested order/importance: 1 (see your cheatsheet), then 2, and **6 as much as you can!** If 6 seems like too much trouble, feel free to look at 3-5 as well.

geom_bar (chapters 3.7-3.9)

- `ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut))`
- "Hey ggplot! I want to make a plot with this dataset diamonds and I want it to be a **barplot** with cut on the x-axis..."
- `geom_bar(mapping = aes(x = cut, fill = clarity), position = "dodge")`
- "with the bars color-filled by clarity, and put side by side."
- `+ coord_flip()`
- "and with the coordinate axes flipped."

Data transformation

5.1.3 dplyr basics

- We have 6 *verbs*: `filter`, `arrange`, `select`, `mutate`, `summarise` and `group_by`
- Each verb is a function that behaves in the same way:

1. The first argument is a data frame.
2. The subsequent arguments describe what to do with the data frame, using the variable names (without quotes).
3. The result is a new data frame.

5.2 Filter rows with `filter()`

- The first argument is the name of the data frame. The second and subsequent arguments are the expressions that filter the data frame
- Remember the usual comparison operators and the shorthand `%in%`

5.2.4 Exercises

1. Find all flights that
   - Had an arrival delay of two or more hours
   - Flew to Houston (IAH or HOU)
   - Were operated by United, American, or Delta
   - Departed in summer (July, August, and September)
   - Arrived more than two hours late, but didn't leave late
   - Were delayed by at least an hour, but made up over 30 minutes in flight
   - Departed between midnight and 6am (inclusive)
2. Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the code needed to answer the previous challenges?
3. How many flights have a missing dep_time? What other variables are missing? What might these rows represent?
4. Why is NA ^ 0 not missing? Why is NA | TRUE not missing? Why is FALSE & NA not missing? Can you figure out the general rule? (NA * 0 is a tricky counterexample!)

- Suggested order/importance: 1, then 2 and 3. 4 is for math geeks.

5.4 Select columns with `select()`

- The first argument is the name of the data frame. After that come the names of the columns to select or a more complicated expression:
  - a range of columns using :
  - "everything but" using -
  - helper functions that deal with patterns in column names: starts_with, ends_with etc
- `rename()` is a variant of `select()` for easy renaming of columns

5.4.1 Exercises

1. Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from flights.
2. What happens if you include the name of a variable multiple times in a select() call?
3. What does the one_of() function do? Why might it be helpful in conjunction with this vector?
   vars <- c("year", "month", "day", "dep_delay", "arr_delay")
4. Does the result of running the following code surprise you? How do the select helpers deal with case by default? How can you change that default?
   select(flights, contains("TIME"))

- Suggested order/importance: 1 and 2 mainly

5.5 Add new variables with `mutate()`

- The first argument is the name of the data frame. After that come the names of the new columns and how to create them
- All the arithmetic and math you can think of is in use!
- Also all kinds of ranks, lags, lead, cumulative sums…

5.4.1 Exercises

1. Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because they're not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.
2. Compare air_time with arr_time - dep_time. What do you expect to see? What do you see? What do you need to do to fix it?
3. Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be related?
4. Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully read the documentation for min_rank().
5. What does 1:3 + 1:10 return? Why?
6. What trigonometric functions does R provide?

- Suggested order/importance: 1, 2 and 3 mainly

5.6 Grouped summaries with `summarise()`

- On its own collapses the whole data frame in to a single result of one function taken on a single column - not very useful!
- Together with `group_by()` calculates the function by group - really useful!
- `group_by()` does work with more than one grouping variable
- Occasionally you might want to `ungroup()` afterwards

5.6.1 Combining multiple operations with the pipe

- Piping improves readability of the R code, and also helps simultaneous thinking and writing. Read it as "and then".
- the result on the left becomes the first argument on the right: *firstfun(x) %>% secondfun(y)* is the same as *secondfun(firstfun(x),y)*
- If you know the Linux/UNIX pipe, this is basically the same thing
- If you know your math, this is exactly the same thing as function composition
- Shift + Ctrl + M is the keyboard shortcut

5.4.1 Exercises

1. Brainstorm at least 5 different ways to assess the typical delay characteristics of a group of flights. Consider the following scenarios:
   - A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.
   - A flight is always 10 minutes late.
   - A flight is 30 minutes early 50% of the time, and 30 minutes late 50% of the time.
   - 99% of the time a flight is on time. 1% of the time it's 2 hours late.

   Which is more important: arrival delay or departure delay?

2. **Come up with another approach that will give you the same output as not_cancelled %>% count(dest) and not_cancelled %>% count(tailnum, wt = distance) (without using count()).**

3. Our definition of cancelled flights (is.na(dep_delay) | is.na(arr_delay)) is slightly suboptimal. Why? Which is the most important column?

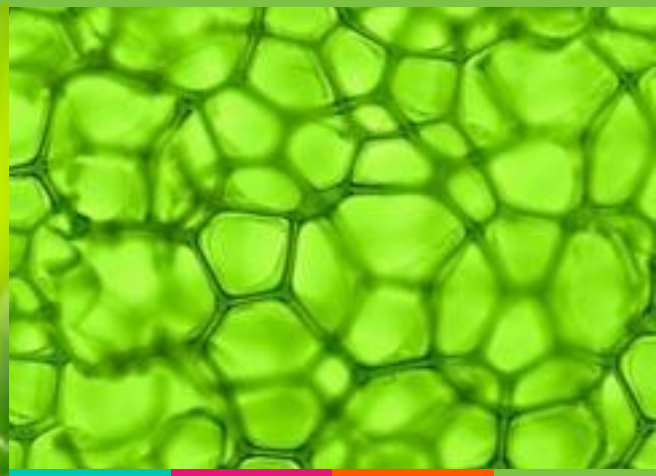
4. **Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled flights related to the average delay?**

5. Which carrier has the worst delays? Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not? (Hint: think about flights %>% group_by(carrier, dest) %>% summarise(n()))

6. What does the sort argument to count() do. When might you use it?

- Suggested order/importance: 1, 2 and 3 mainly

Extra!



Tidy data

- The short version:
  - you want single columns to be single variables
  - you want single rows to be single observations
  - (you want one type of data in one table)
- All of *tidyverse* (*ggplot2* in particular) relies on this idea
  - but some other packages / software might not, and you have to deal with this
- The relevant functions are *gather* and *spread*, and *separate*

Combining data sets

- When you have two data frames with one (or more) common variable, and you need to combine the data from the two sets, matching the common variable, use one of the *join* functions (left, right, inner or full)
  - or *merge* from the base R
  - the versions differ in what they do in case there are unmatched values
- When you have two data frames with the same variables (columns), you can stick them together with *bind_rows*
  - note that the columns need not be in the same order
  - *bind_cols* also exists, but maybe join is what you actually need in that case?

Seija Sirkiä

PhD, data scientist, application specialist

seija.sirkia@csc.fi

https://www.facebook.com/CSCfi

https://twitter.com/CSCfi

https://www.youtube.com/c/CSCfi

https://www.linkedin.com/company/csc---it-center-for-science