Naniar and annotations

Susannah Cowtan

1 July 2019

Get yourself set up

You'll need these packages loaded. You don't have to use `suppressPackageStartupMessages()`, but I do, because none of the package conflicts where one package's "filter", say, masks another package's are going to cause a problem here, so I'd rather not have those messages cluttering up my output.

```
# I am an incorrigible tidyverse programmer.  Sorry not sorry.
suppressPackageStartupMessages(library(dplyr))
library(readr)
suppressPackageStartupMessages(library(tidyr))

# Libraries for graphics
suppressPackageStartupMessages(library(scales))
library(ggplot2)
library(ggrepel)
library(naniar)
```

Define some global variables

My data are in a directory called `talk_data`, which is in the same directory as the directory holding my scripts. I always access data using relative paths because they are less likely to get broken. You may still need to change the path, depending on where you put the data!

```
# My "target species"
species <- "Hermania scabra"

# "target year" - we look at the observations before and including/after this
target_year <- 1992
```

Read in the data

I use `readr::read_csv` rather than read.csv because it is faster and makes fewer assumptions - in particular, it doesn't assume anything is a factor. Then I contradict myself by telling it to make assumptions, using `col_types = cols()`.

```
shark_df <- read_csv("../talk_data/shark.csv", col_types = cols())
```

Sadly, there are no actual sharks in these data that I know of; it's mostly molluscs.

Set up a theme

You don't need to set up the theme for the workshop, but you have it if you want it for reference. I suggest you do define the colours, because those will appear in the code later.

```
# Custom colours (colourblind friendly, supposedly!) -
# Pick widely spaced range, and omit colours nearest black and white
prof_colours <- scales::viridis_pal(option = "viridis")(8)[c(2, 5, 7)]
# Nice and contrasty (for a colour vision person - should really check R/G vis)
highlight_colour <- "violetred3"

# Look of ggplot
theme_prof <- function (base_size = 10, base_family = "") {
  theme_bw(base_size = base_size, base_family = base_family) %+replace%
    theme(
      axis.text = element_text(size = 8),
      axis.title = element_text(size = 10),
      axis.title.x = element_text(margin = margin(t = 14)),
      axis.title.y = element_text(margin = margin(l = 0, r = 14),
        angle = 90),
      legend.text = element_text(size = 10),
      legend.title = element_text(size = 10)
    )
}

# Make this theme the default
theme_set(theme_prof())
```

Data munging: classify observations as pre or post the target year

You don't need to type the `packagename::` parts of this as you have the packages loaded; I include them to show which package different commands come from, in case you're unfamiliar with the tidyverse.

A big part of making good plots is getting the data in the right format beforehand. `ggplot` likes one column per variable. Here my variables are `post` and `pre`, referring to whether the observation was made after the target year or not, and `scientificName` for the labels.

```
## Pre & post target_year
shark_gg_df <- shark_df %>%
  # Up to and including target year or after?
  dplyr::mutate(pre_post = ifelse(year > target_year, "post", "pre")) %>%
  dplyr::select(scientificName, pre_post) %>%
  # Group the data by pre or post target_year
  # Any operation after this will be performed on each group separately
  dplyr::group_by(pre_post) %>%
    # Count the number of records for each species (puts in column "n")
    dplyr::count(scientificName) %>%
  dplyr::ungroup() %>%
  # ggplot likes a "wide" format, so pre and post need to be separate columns
  # get columns scientificName, pre (count), post (count)
  tidyr::spread(pre_post, n)
```

Making a terrible approximation that survey effort was equal for all years, calculate the slope of a reference line.

```
# Calculate slope of comparison line by how many years either side of target_year
# (approximation - takes no account of survey effort)
slope <- length(which(unique(shark_df$year) > target_year)) /
```

```r
length(which(unique(shark_df$year) <= target_year))
```