# Using ggplot2 to recreate a line plot of annual temperature anomalies-CC217

William Komu

2023-02-15

### Load data and tidy

We need to load the data and tidy it to put the data in a long format... `pivot_longer` The code `mutate(month = factor(month, levels = month.abb))` turns the months into numerical order rather than the alphabetical order they come out with in the first instance!

### Preloaded vectors

The `month.abb` reports the abbreviated vec of the 12 months!

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

### Programming `glue` objects

We can program some elements of the charts and insert them into our ggplot

### Create subsets of last Dec and next Jan

Since the original plot kinda rep the previous Dec and preceding Jan, we need to create two more dfs to subset the these variables. These are then joined to the our main df `t_diff`! Oh! The Prof. calls it ***"engineer `last_Dec` and `nextJan` variables"***

We can then define the months index, where `last_Dec` and `next_Jan` are 0 and 13, respectively, and `month.abb` the usual `1:12` as follows:

`month_number = as.numeric(month)-1)`

The `scale_x_continuous(breaks = 1:12, labels = month.abb)` and `coord_cartesian(xlim = c(1, 12))` fixes our x-axis to indicate the extension to last_Dec and next_Jan

### Coloring

The original seems to have been made with `Python` and it uses the fairly common veridis palette which is common for countering the **red-green color deficiency**

R has a special gradient_(): `scale_color_veridis_c` the c gives us a continuous color gradient with each year having it's own color! which

### Theme-ing

The original has the background `darkgray` and plot panel `black`

We need to then rid the white gridlines

### Duplicate `ticks`

Since the original plot has `ticks` marks around it, we can use:

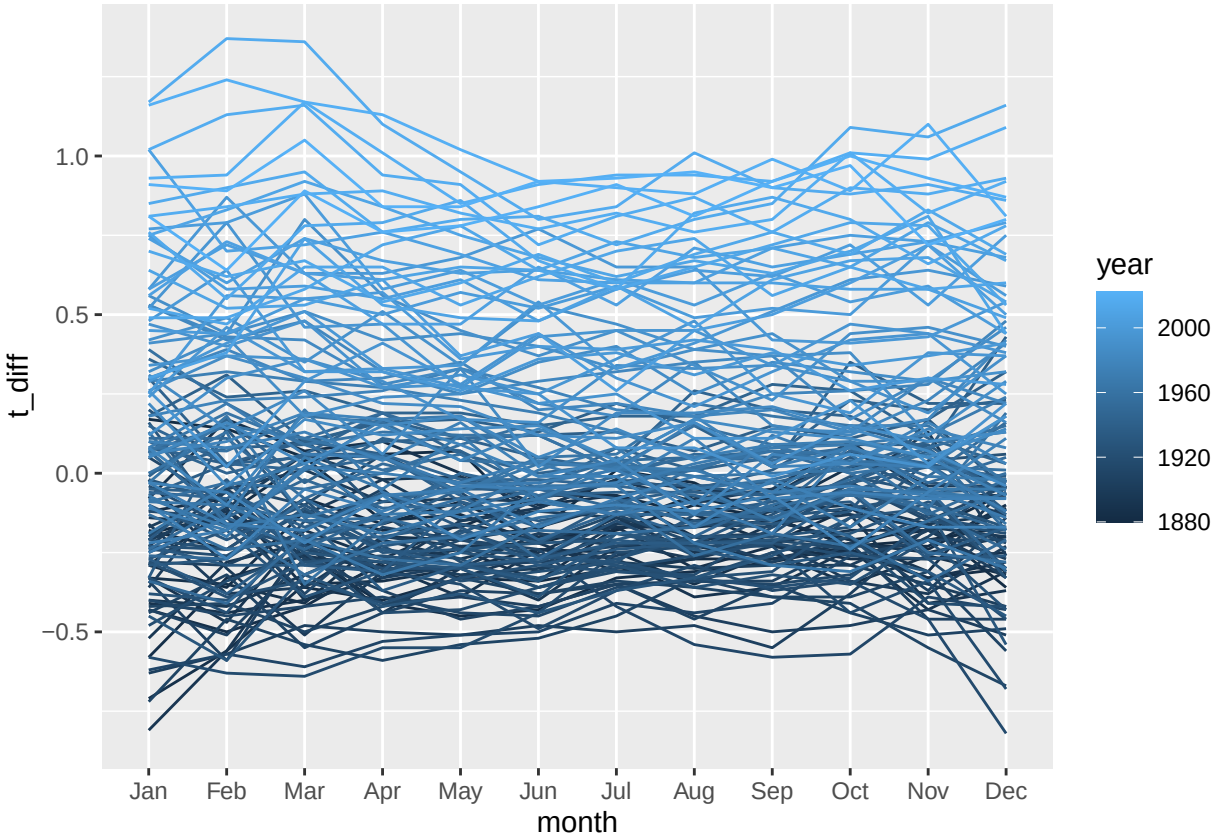
`scale_y_continuous(breaks = seq(-2, 2, 0.2), sec.axis = dup_axis(name = NULL, label = NULL))` to replicate that effect!

```
## # A tibble: 1 x 5
##    year month    t_diff month_number is_this_year
##   <dbl> <fct>     <dbl>        <dbl> <lgl>
## 1  2023 last_Dec    0.8            0 FALSE
```

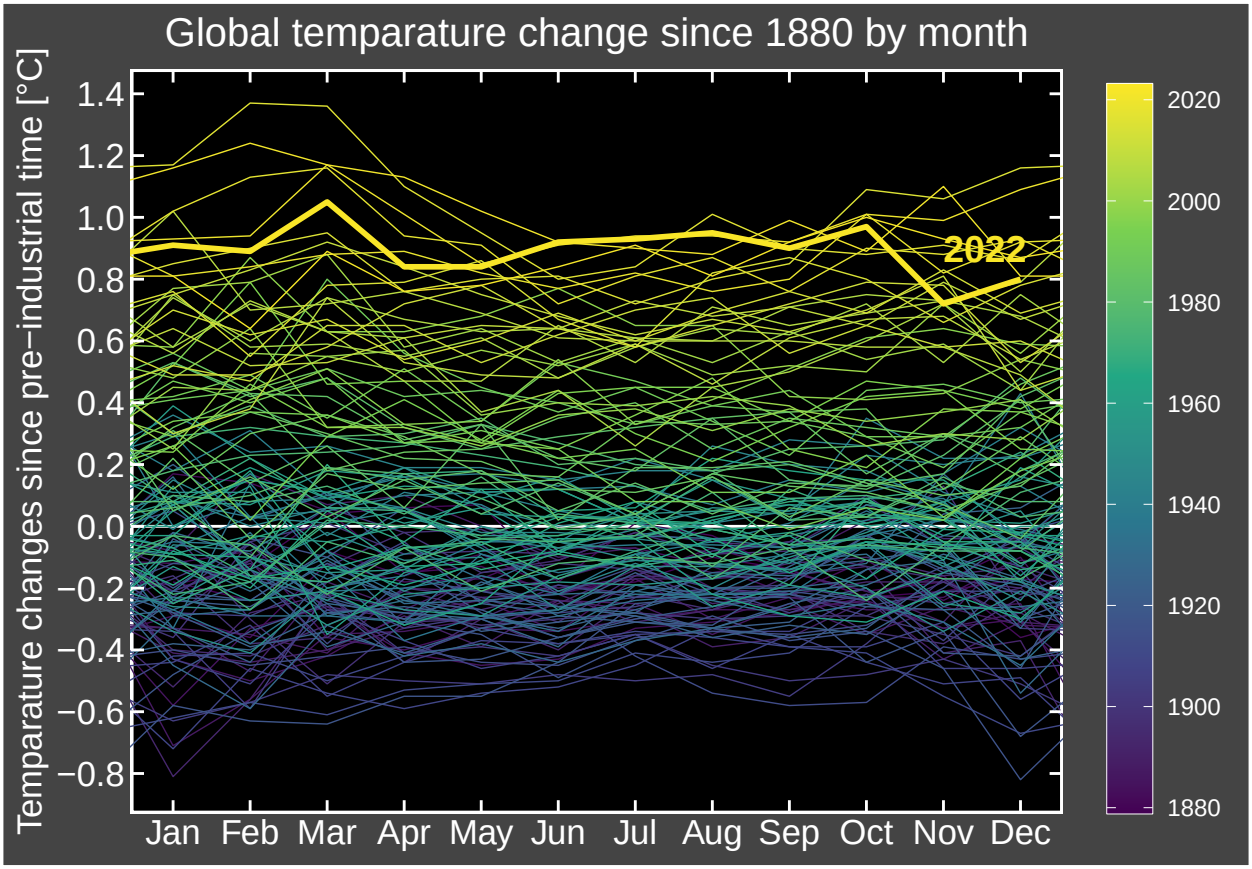
### Initial draft plot

Let's plug the data to ggplot and display a skeleton of our draft plot, so that we can tell a kind of a story of the transition to the final draft.

Note: the `group` allows for plotting lines in `geom_line()`



**Final plot**




# Commentary

To keep my code chucks clean, I'll start doing commentaries - explanations of the ***not-so-obvious*** **code lines and tricks** or nifty tricks that I have a feeling are **outta this world** - at the end of the document.

**My adaptation:**

In subsetting the data for the annotation trick, I adapted `is_this_year` because I am in 2023 which has no data other than the `last_Dec` we created!

In subsetting this year, I had to deduct one (-1) so that I can get last year's as this year! I will amend this line once i have a dataset that has data for 2023! NOTE: last_Dec takes the previous years data and put it as a month 0, helping as to showcased a kind of transition/action in the edge of the plot!

**Read the end of df**

Print the tail end of a df, where n = # of rows! `data %>% tail(n=#)`

```
## # A tibble: 10 x 3
##     year month t_diff
##    <dbl> <fct>  <dbl>
```

```
##  1  2022 Mar     1.05
##  2  2022 Apr     0.84
##  3  2022 May     0.84
##  4  2022 Jun     0.92
##  5  2022 Jul     0.93
##  6  2022 Aug     0.95
##  7  2022 Sep     0.9
##  8  2022 Oct     0.97
##  9  2022 Nov     0.72
## 10  2022 Dec     0.8
```