GEOG456_T1

Javier Arce

1/09/2020

Time Plotting

This is a tutorial for learning how to import data into R and make some figures that show temporal patterns. For this tutorial we will be using NOAA data from a sensor collecting temperature in Chapel Hill.

First we need to import the data.

```
the_file <- "C:/dev/geog456/geo_time_space/TMAX_chapel_hill.csv" ## identify where the file is located
df <- read.csv(the_file) # open a csv
head(df) ## to see the column names and the first 6 rows of the data.frame we opened
```

```
##      X.3 US1MISW0005 X20180101 PRCP  X0  X X.1 N X.2
## 1  21062 USC00311677  20180101 TMAX -33 NA  NA 7 800
## 2 116823 USC00311677  20180102 TMAX -28 NA  NA 7 800
## 3 213478 USC00311677  20180103 TMAX  -6 NA  NA 7 800
## 4 310682 USC00311677  20180104 TMAX   0 NA  NA 7 800
## 5 505407 USC00311677  20180106 TMAX -11 NA  NA 7 800
## 6 600882 USC00311677  20180107 TMAX -33 NA  NA 7 800
```

Notice that the column names are not very nice. Let's change that

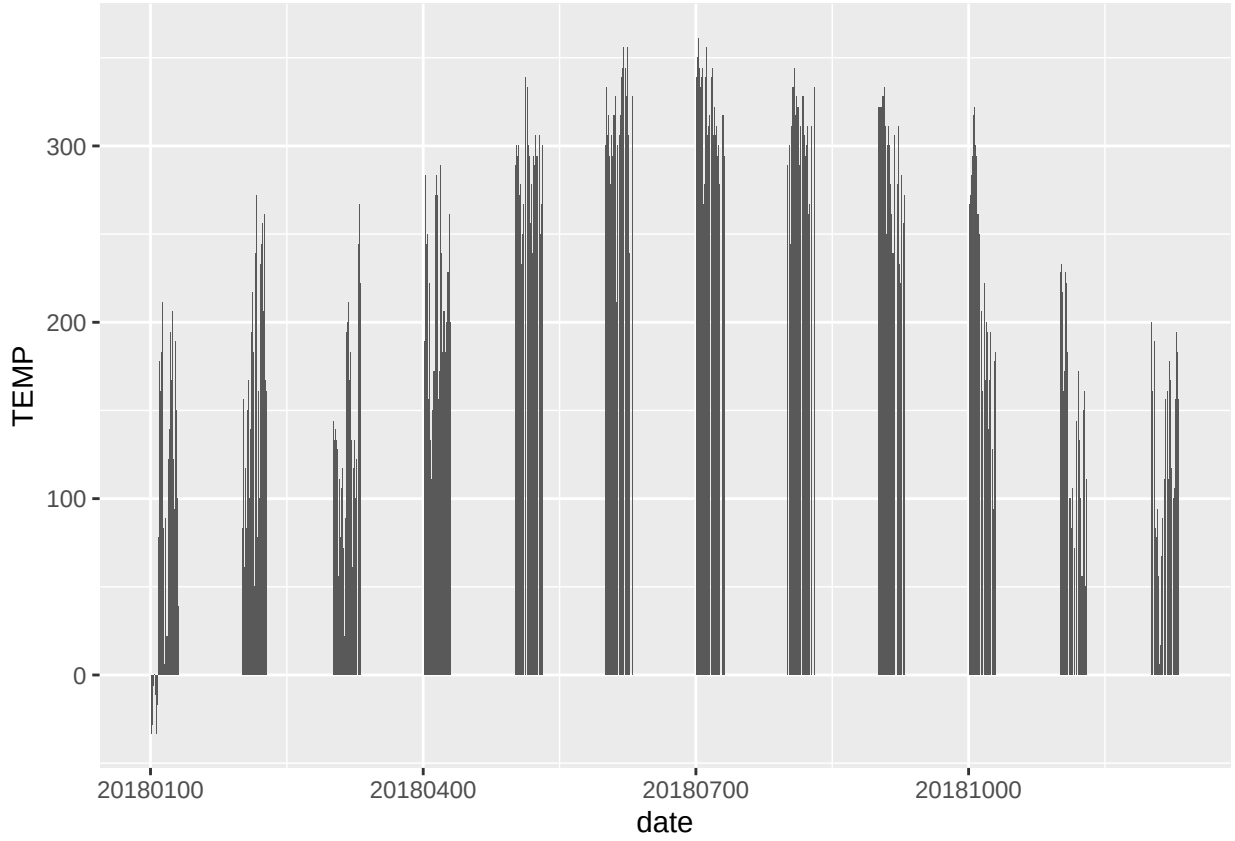
```
names(df) <- c("N","name","date","PRCP","TEMP") ## this will change the column names
head(df) ## notice that the column names have changed
```

```
##        N        name     date PRCP TEMP NA NA NA  NA
## 1  21062 USC00311677 20180101 TMAX  -33 NA NA  7 800
## 2 116823 USC00311677 20180102 TMAX  -28 NA NA  7 800
## 3 213478 USC00311677 20180103 TMAX   -6 NA NA  7 800
## 4 310682 USC00311677 20180104 TMAX    0 NA NA  7 800
## 5 505407 USC00311677 20180106 TMAX  -11 NA NA  7 800
## 6 600882 USC00311677 20180107 TMAX  -33 NA NA  7 800
```

```
df <- df[order(df$date),] ## I am making sure that the data.frame is sorted by date
```

Now let's make a simple bar graph to see how the maximum temperature changed. For that we will install two libraries (ggplot2 and colorbrewer)

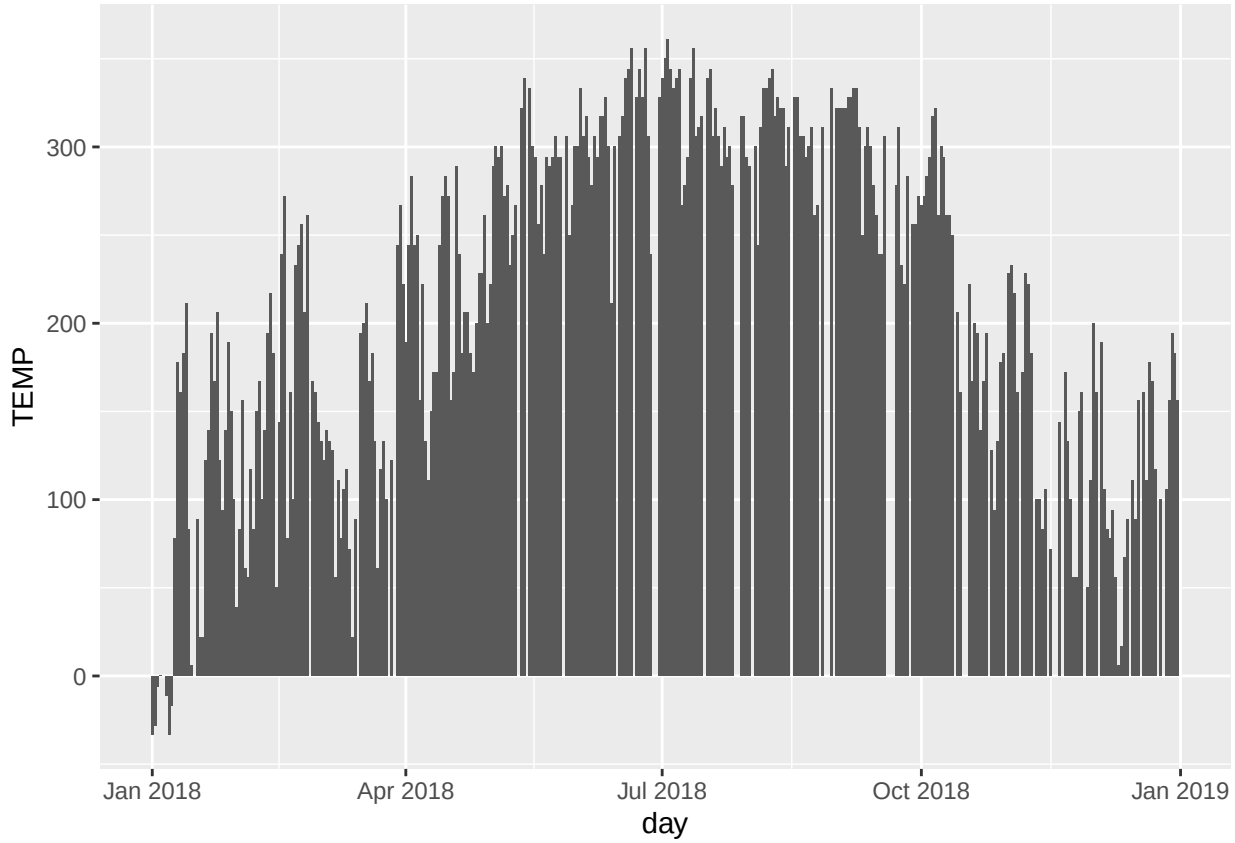
```r
#install.packages("ggplot2") ## take out the hash-symbol at the beginning of this line if you want to r
#install.packages("RColorBrewer")
#After installing the packages you need to open the libraries.
library(ggplot2)
library(RColorBrewer)

## now let's make our bar plot
p <- ggplot(df, aes(x=date, y=TEMP)) + geom_bar(stat="identity") ## this is giving the instructions to
# the + geom_bar(stat = "identity") is giving the instruction to make a bar graph where the heights of

p ## will print the plot
```



Congratulations, you have made your first bar plot! If you do not see a plot like the one above, check that you have the ggplot2 libraries installed and that you have opened the csv as a data.frame.

Notice two things. 1. That the vertical axis goes from 0 to almost 400. This is because the temperature values are recorded as decimal centrigrades. So you should divide the values by 10 if you want to have a more normal representation of the temperature values. But for now we will use the numbers as they come from NOAA. 2. Notice that the bar plots are clustered. This is beacuase we gave R the value dates, hence Jan 31 2018 (20180131) will appear as it is 70 units away from February 01 2018 (20180201). For this reason we need to change this values to a date format.

```r
df$day <- as.Date(as.character(df$date), "%Y%m%d") ## this will create a new column named day by interp
## now lets make the plot again with the value day in the x axis
p <- ggplot(df, aes(x=day, y=TEMP)) + geom_bar(stat="identity")
p
```



Notice that there are some gaps in the data. You could have also noticed that some data was missing by looking at the number of rows in the data and noticing that we have only 322 counts and not 365 counts.

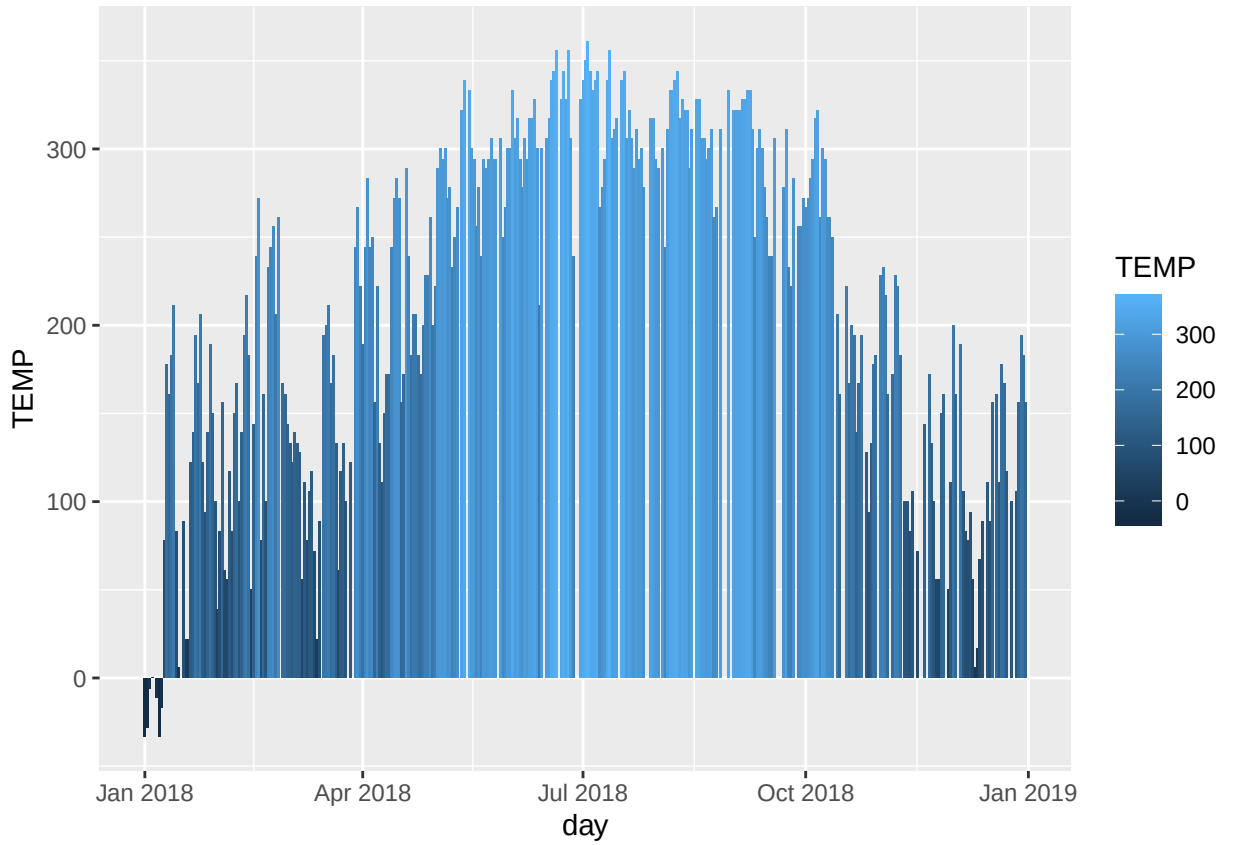
```
nrow(df)
```

```
## [1] 322
```

Today I am going to overlook the missing 43 counts as they seem to be missing throughout many months. Missing data is going to be a common problem, and there are different ways to address it, but that is not the purpose of this tutorial.

Let's get back to making plots and giving these plots some color.

```
p <- p + aes(fill = TEMP) ## fill = TEMP is indicating that it will fill the bars with a color gradient
p
```



Here we can see the magic of ggplot2. Notice that the code was very simple. Since you already have a plot object p, then we can make changes to what we already have done. So you should get the same result if you write the following code.

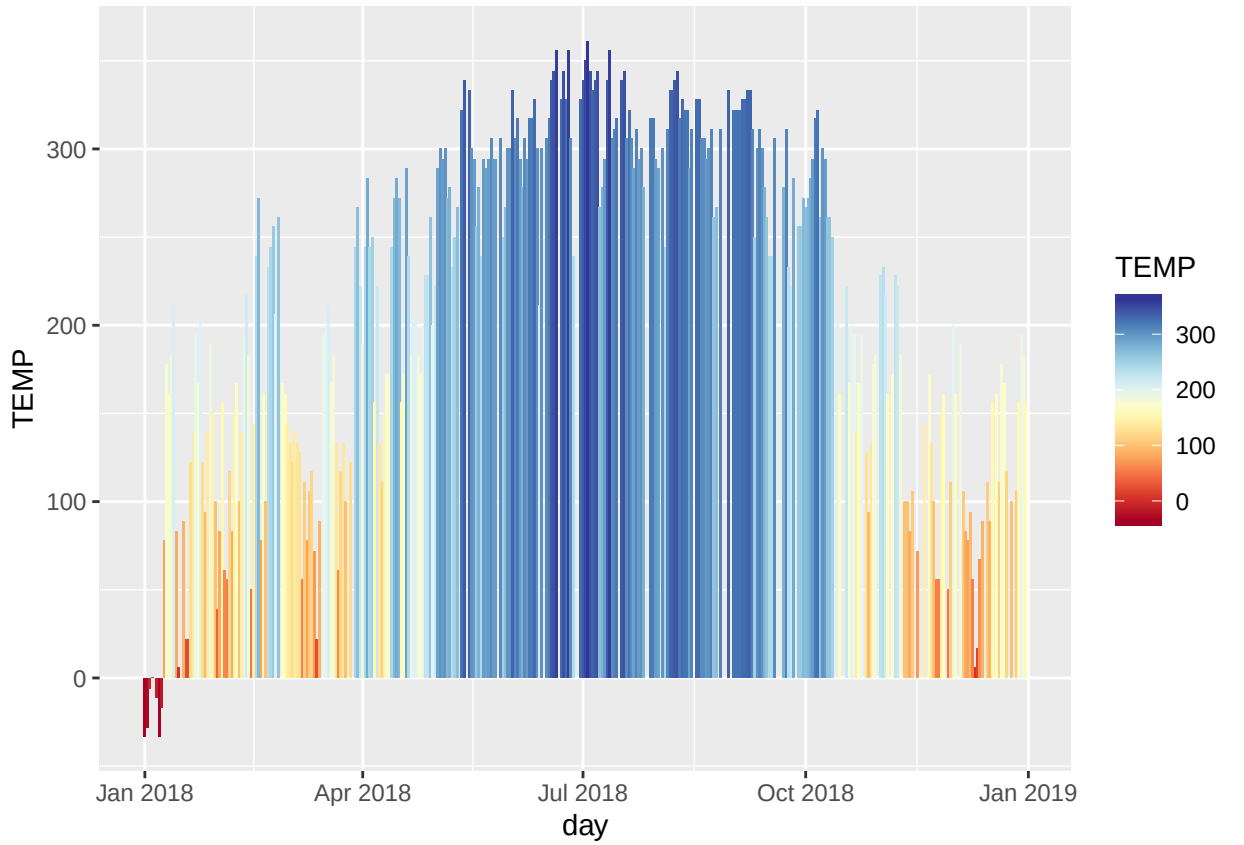
```
ggplot(df, aes(x=day, y=TEMP, fill = TEMP)) + geom_bar(stat="identity")
```

But I am not happy with the colors. I want colors that relate to temperature: cold should be blue and hot should be red. For this task I am going to use RcolorBrewer. You can check the available combinations using the following code.

```
display.brewer.all()
```

I like the RdYlBu combination for this task. So I am going to add the colors to the table fill.

```
p + scale_fill_gradientn(colours = colorRampPalette(brewer.pal(11,"RdYlBu"))(11))
```




```
#### this code is indicating that the fill should be done with a color ramp palette using the brewer.pa
```

Much better! But wait, the cold temperatures are red and the hot are cold. So we need to invert the color ramp. We do this by using function **rev.** For example look what **rev** does to a vector or list of numbers.

```
x <- letters[1:5] ## this will create a vector of letter a-e
x
```

```
## [1] "a" "b" "c" "d" "e"
```

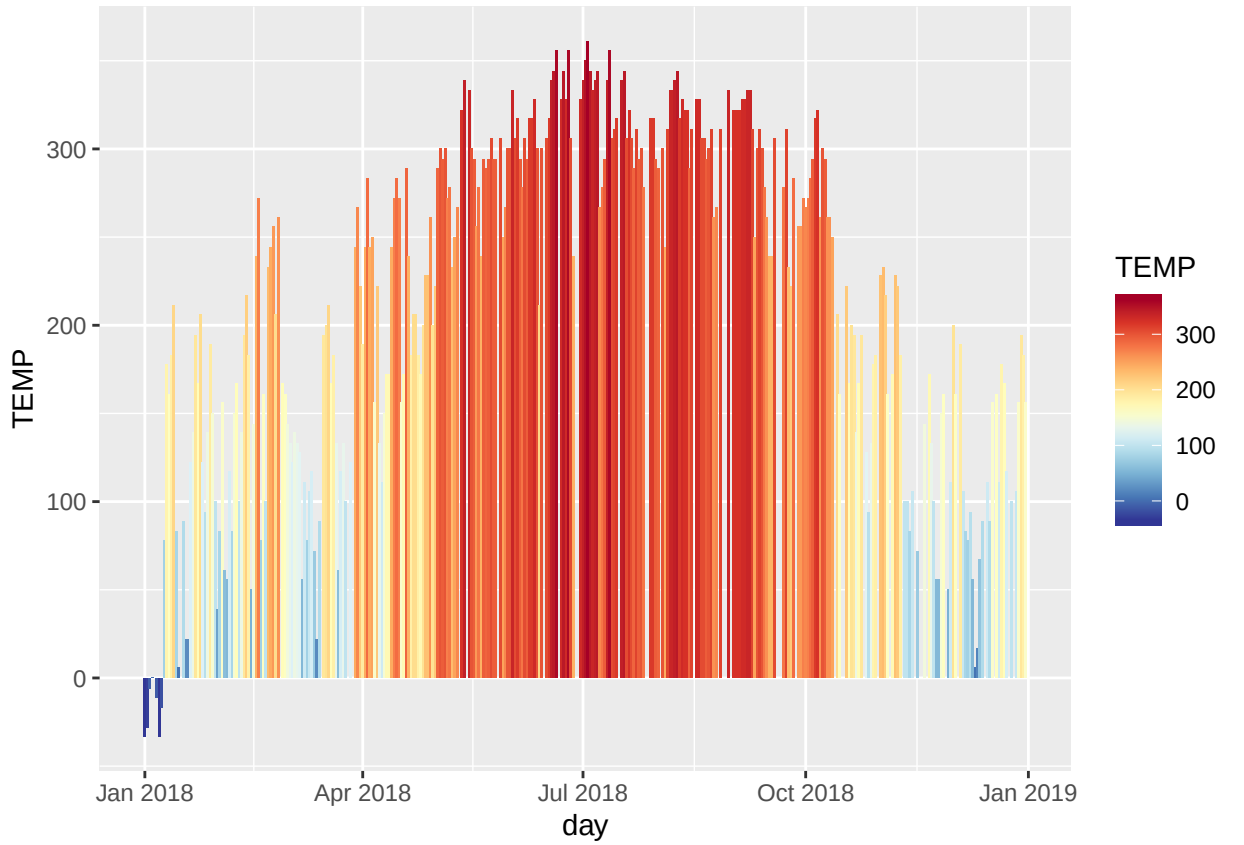
Now look at the vector if we use **rev.**

```
rev(x)
```

```
## [1] "e" "d" "c" "b" "a"
```

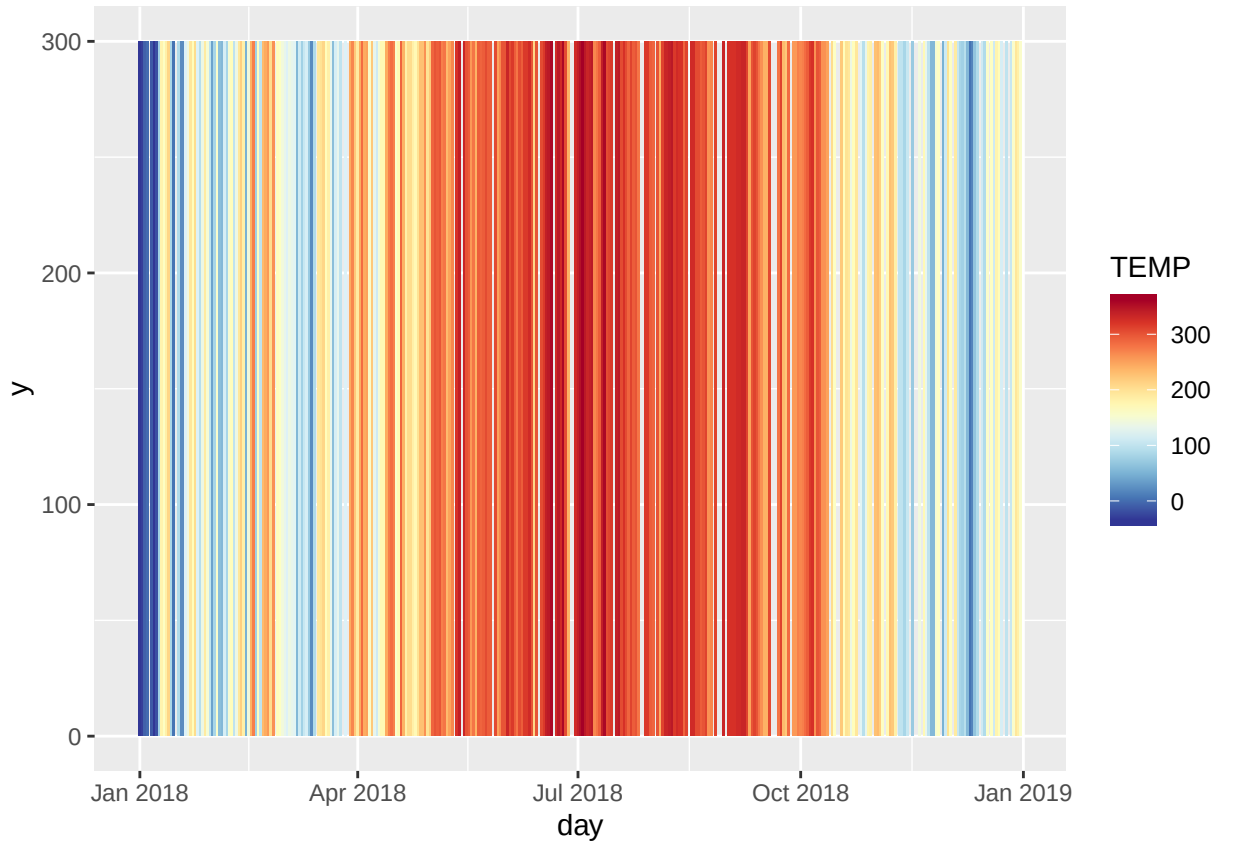
Hence we can use the same function to invert the color ramp.

```
p <- p + scale_fill_gradientn(colours = colorRampPalette(rev(brewer.pal(11,"RdYlBu")))(11))
p
```



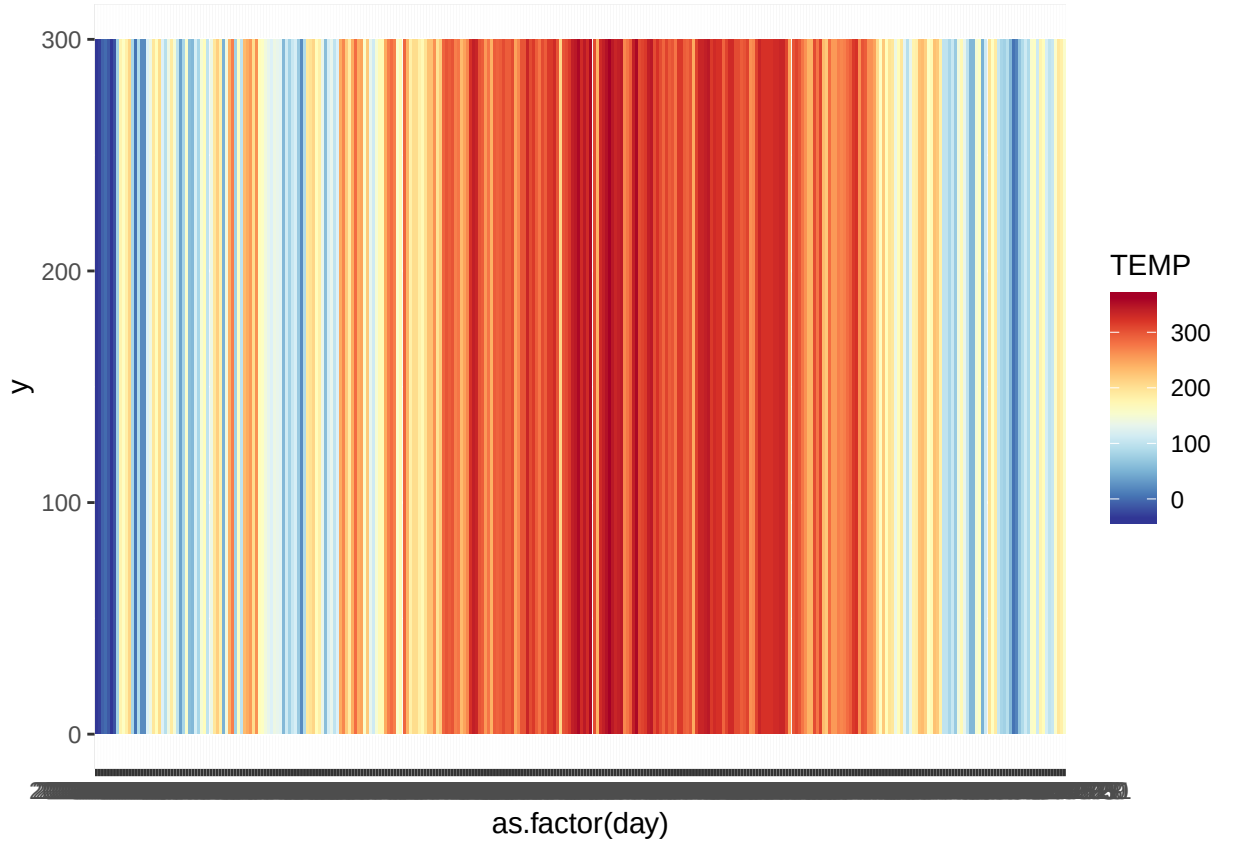
Now this looks much better. This figure reminded me of warming stripes by Ed Hawkins http://blogs.reading.ac.uk/climate-lab-book/files/2018/12/wmo_stripes.png In warming stripes Hawkins compares temperature for multiple years at a location, and only uses relative color to show the patterns of temperature change. To mimic this, I will assign the same value of y to all the bars, keeping the color coding the bars based on TEMP.

```
p + aes(y = 300)
```



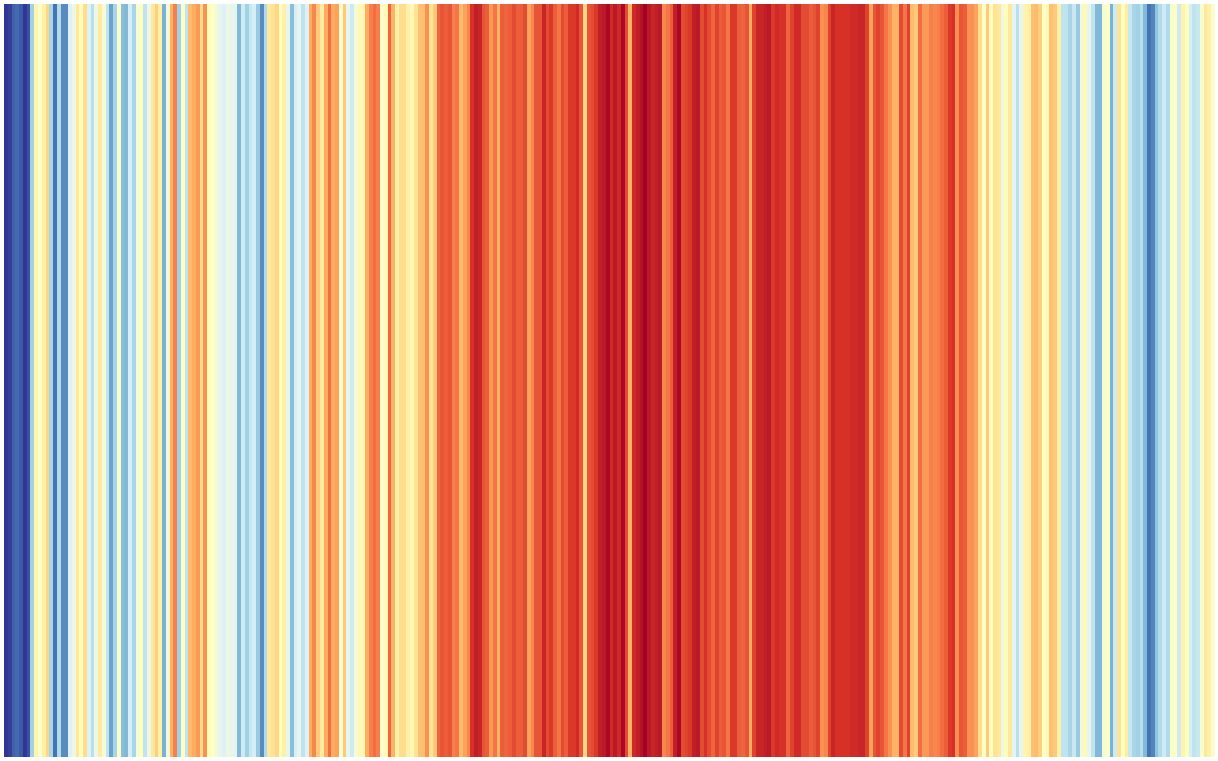
I don't like the missing days. So I am going to make days a factor.

```
p + aes(x = as.factor(day), y = 300)
```



Also, let me eliminate a bunch of stuff that I don't want to see.

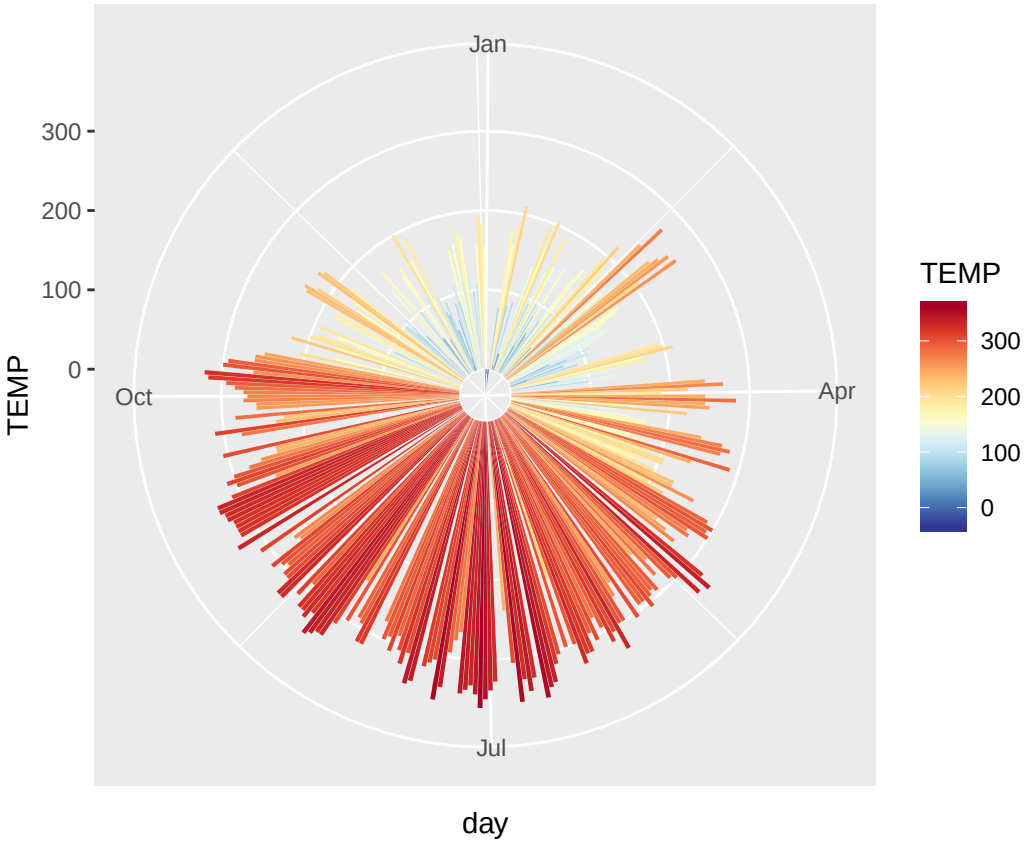
```
p + aes(x = as.factor(day), y = 300) +
  theme_minimal() +
   theme(
     axis.text = element_blank(),
     axis.title = element_blank(),
     panel.grid.major = element_blank(),
     panel.grid.minor = element_blank(),
     legend.position='none')
```



Pretty cool!

But let's make this graph even more fun. While we can see that the x axis is time, it does not gives us a sense of a cycle. We could change the image to invoke the metaphor of time from a clock. (I was also inspired by Andrieko et. al.'s figure on cyclical processes).

```
p + coord_polar(theta = "x")
```



What patterns can you see now that were not as obvious when seeing the data horizontally?
What changes could be made to this visualization to make it more elegant and informative?

Now let's do another version of a plot which has become very popular to visualize change in time. The ridgeline plot (and also know as the joy plot... but that is another story) was orginally from a radioastronomer looking at pulsars. The plot shows brightness as a function of time for a periodic process. If you look for #joyplots in twitter https://twitter.com/hashtag/joyplot you will see many expamples.

One of the main beauties of R is that people create libraries with very easy to read documentation. To make a ridgline plot first we download a library and prepare the data.

```
#install.packages("ggridges")

library(ggridges) # open the ggridges library. It needs ggplot2 and a recent version of R to work
```

```
##
## Attaching package: 'ggridges'
```

```
## The following object is masked from 'package:ggplot2':
##
##     scale_discrete_manual
```

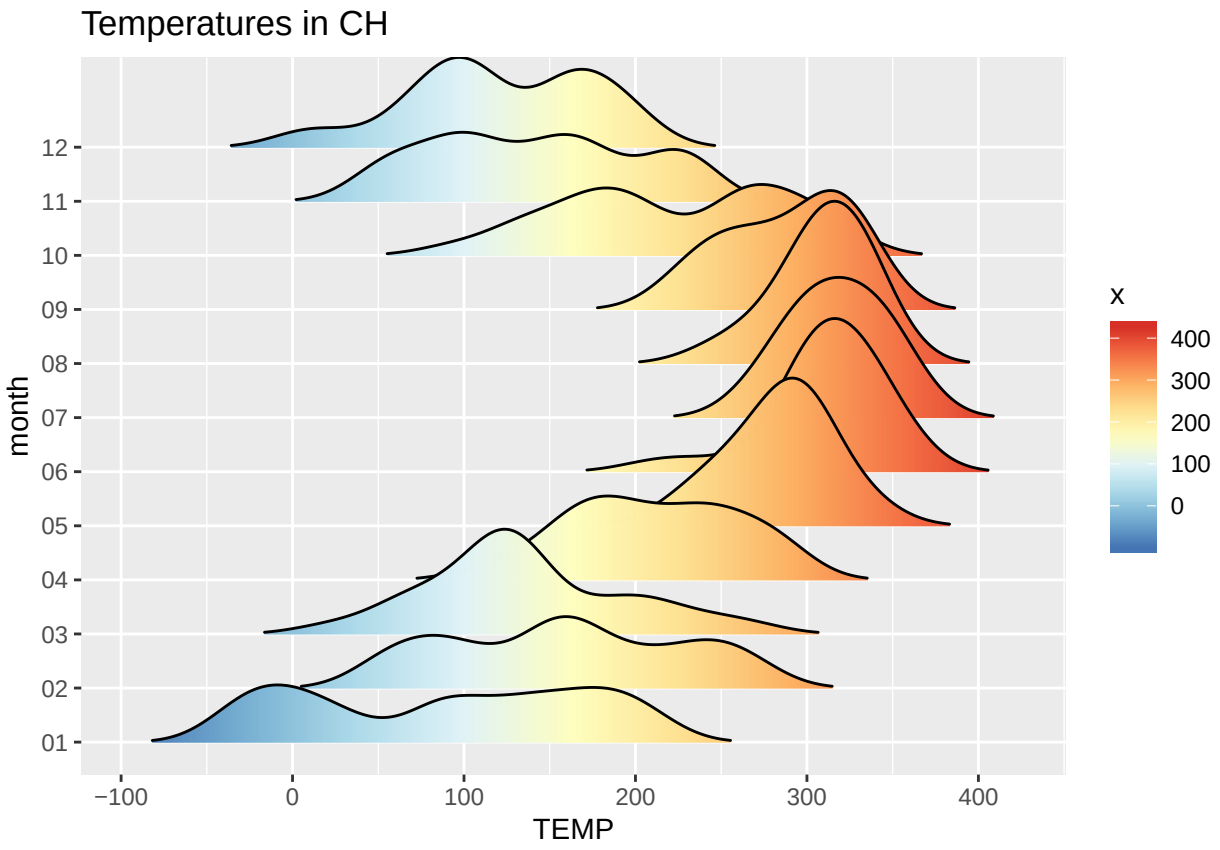
```
df$month <- format(df$day,"%m") ### from the column day create another column named month that has the
head(df)
```

```
##        N        name     date PRCP TEMP NA NA NA  NA        day month
## 1  21062 USC00311677 20180101 TMAX  -33 NA NA  7 800 2018-01-01    01
## 2 116823 USC00311677 20180102 TMAX  -28 NA NA  7 800 2018-01-02    01
## 3 213478 USC00311677 20180103 TMAX   -6 NA NA  7 800 2018-01-03    01
## 4 310682 USC00311677 20180104 TMAX    0 NA NA  7 800 2018-01-04    01
## 5 505407 USC00311677 20180106 TMAX  -11 NA NA  7 800 2018-01-06    01
## 6 600882 USC00311677 20180107 TMAX  -33 NA NA  7 800 2018-01-07    01
```

I made the column month because I am going to aggregate the data by month.

```
ggplot(df, aes(x = TEMP, y = month, fill = ..x..)) +
  geom_density_ridges_gradient(scale = 3, rel_min_height = 0.01) +
  labs(title = 'Temperatures in CH') +
  scale_fill_gradientn(colours = colorRampPalette(rev(brewer.pal(9,"RdYlBu")))(100))
```

```
## Picking joint bandwidth of 21.4
```



What do you see from this graph that you couldn't see using the bar graph or the circular graph? What changes could be made to this visualization to make it more elegant and informative?

You can download the r script from Sakai: https://sakai.unc.edu/x/rzyciF You can download the data from Sakai: https://sakai.unc.edu/x/fBC00s