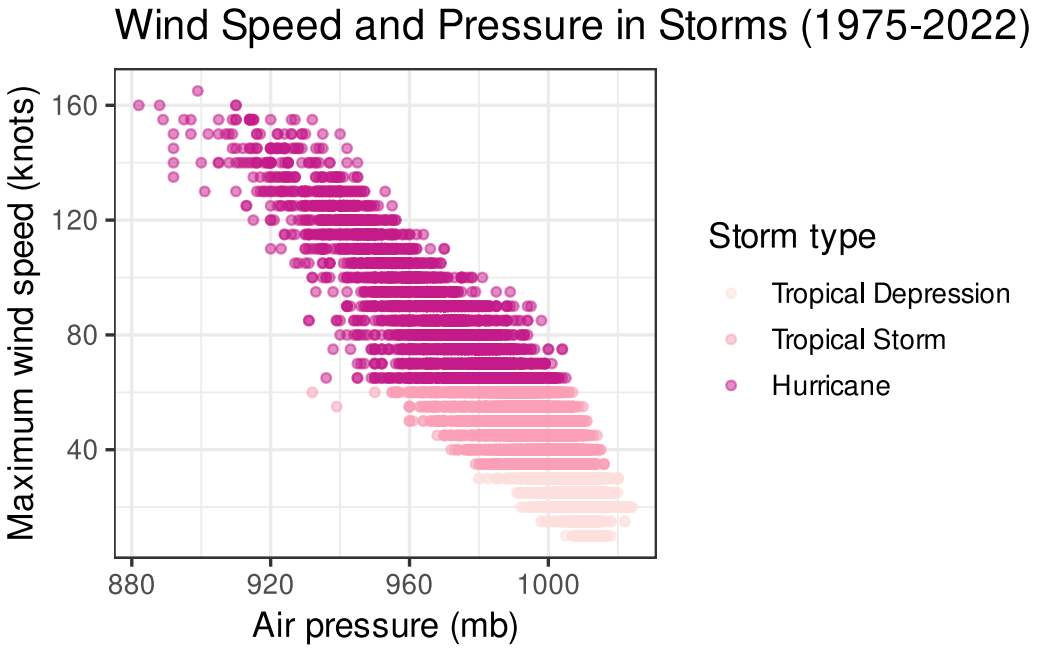
Explanatory Data Analysis

All of the plots in this problem set are meant to focus on a particular story to tell your a particular audience. As you craft them, think carefully about the labels, themes, aesthetic settings, and the manner in which it facilitates comparisons.

1. Provide the code that will create this plot from the `storms` data set inside the `dplyr` package. Here, each dot is a single storm. A few notes:

- You will need to create a new column indicating storm type. For that you'll need to learn how that determination is made and then add it to the data frame. The `case_when()` function is a helpful shortcut similar to `ifelse()`.
- The color palette belows to a set of palettes called ColorBrewer. You can read more about them here: https://ggplot2.tidyverse.org/reference/scale_brewer.html.



2. Are there any changes that you'd make to the plot above to make it easier to read or clarify what it is showing?

3. Remake your plot using a very specific visual theme: the botanical paintings of the US Department of Agriculture. They have been wrapped up into a ggplot2 theme inside the `ggpomological` package: https://github.com/gadenbuie/ggpomological. Read about how to use the package and write below the additional layers that you added to your plot above (for a challenge, see if you can get the handwriting font to show up).

4. The `present` data frame stores data from the US Census about the number of boys and girls born each year between 1940 and 2013. It is stored in the `stat20data` package, which can be installed from GitHub using the `devtools` package.

```
# install.packages("devtools")
# devtools::install_github("https://github.com/stat20/stat20data")
library(stat20data)
slice(present, 1:5)
```

   Make a plot to visualize the trend over time in the proportion of births that are girls. Put the annual variability into context by ensuring the viewer can see the full range of possible values that the proportion could take. Allow the comparison to a baseline of .5 by drawing a horizontal line at that value (see `geom_hline()`).

5. (Challenge) Some of the most effective plots on the web don't come from R or Python, but from dedicated visualization libraries written in JavaScript. That is the case for the EV registration plot made by the Department of Energy (https://afdc.energy.gov/data/10962), which uses a library called Highcharts.

   While you could craft the plot directly using JavaScript, a member of the R community has written a package that allows you to make them directly from R: you write R code, which the package turns into Javascript, which Highcharts then renders as a plot. Read how to use the package on its documention site: https://github.com/jbkunst/highcharter/.

   Your task is to recreate the EV Highcharts plot exactly using the data found in an excel spreadsheet on the DoE website. You can export the excel spreadsheet as a csv file, as we did in class, or you can experiment with the `readxl` package to read it into R directly.

Joins

```
# A tibble: 4 × 3
  student_id name    major
       <dbl> <chr>   <chr>
1          1 Ada     Stat
2          2 Bruno   DS
3          3 Chandra Stat
4          4 Diego   Econ
```

```
# A tibble: 4 × 2
  student_id quiz_score
       <dbl>      <dbl>
1          2         88
2          3         94
3          4         77
4          5         91
```

These data sets illustrate a roster of students registered for a particular lab section (`roster`) and the record of grades on a quiz (`grades`). For each of the following, write appropriate the `dplyr` join command and sketch the resulting data frame that contains those records.

6. All registered students who took the quiz.

7. The quiz scores for all registered students.

8. The names and majors or all students who took the quiz.

9. All records for students either registered for the section or who took the quiz or both.

10. The name and major of the registered students who didn't take the quiz.

11. Calculate the maximum wind speed of each storm, then attach that column of data to the original `storms` data frame.

12. Create a small data frame that you can then use to filter the `storms` data so that it only retains records of storms that are "Hurricanes", "Tropical Storms", and "Tropical Depressions". (note: don't use `filter`!)

13. Create a small data frame that you can use to add on a new column to `storms` called `type` that has levels of "Hurricanes", "Tropical Storms", and "Tropical Depressions" for storms with that `status` and "Other" for all other storms. (note: don't use `mutate()`!)

14. Remake the state-by-state EV registration chart but instead of plotting the distribution of registrations, display, for every state, either: a) the proportion of vehicle registrations that are EV b) the ratio of EV registrations to residents. To do this you will need to track down additional data online and join it with the EV data registration data.

15. In the space below, sketch out four Venn diagrams, one for each of the mutating joins. Each diagram should have two circles representing tables X and Y, showing some overlap. Shade in the parts of the diagram where those rows exist in the joined data frame.

...and Pivots

The `tidyr` package contains the following data set called `fish_encounters` collected by aquatic scientist (and Cal grad!) Myfanwy Johnson.

```
library(tidyr)
fish_encounters
```

```
# A tibble: 114 × 3
   fish  station  seen
   <fct> <fct>   <int>
 1 4842  Release     1
 2 4842  I80_1       1
 3 4842  Lisbon      1
 4 4842  Rstr        1
 5 4842  Base_TD     1
 6 4842  BCE         1
 7 4842  BCW         1
 8 4842  BCE2        1
 9 4842  BCW2        1
10 4842  MAE         1
# ℹ 104 more rows
```

16. What is the unit of observation? What values are found in the `seen` column? What do they indicate? What values are missing? (see `?fish_encounters`)

17. Some forms of analysis require a format where every row is a unique fish and every column is a monitoring station. Write the code to perform this transformation and sketch the resulting data frame, illustrating the column names, the number of rows, and a sample of the values that are seen.

18. R uses `NA` to indicate a measurement that was intended to be taken but that is missing for whatever reason. In this case, that absence means that in fact the fish was not detected, so the value should really be a `0`. Revisit your `pivot_()` command and add an additional argument to fill those values with `0` instead of `NA` (see the help file for pivot).

19. Reshape your new data frame back to the shape of the original `fish_encounters`. What are the dimensions and how does it compare to the original data frame?

The package also contains a data frame called `billboard`, which records the billboard rankings of different songs in 2000.

```
library(tidyr)
billboard
```

```
# A tibble: 317 × 79
   artist     track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
   <chr>      <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2 Pac      Baby… 2000-02-26      87    82    72    77    87    94    99    NA
 2 2Ge+her    The … 2000-09-02      91    87    92    NA    NA    NA    NA    NA
 3 3 Doors D… Kryp… 2000-04-08      81    70    68    67    66    57    54    53
 4 3 Doors D… Loser 2000-10-21      76    76    72    69    67    65    55    59
 5 504 Boyz   Wobb… 2000-04-15      57    34    25    17    17    31    36    49
 6 98^0       Give… 2000-08-19      51    39    34    26    26    19     2     2
 7 A*Teens    Danc… 2000-07-08      97    97    96    95   100    NA    NA    NA
 8 Aaliyah    I Do… 2000-01-29      84    62    51    41    38    35    35    38
 9 Aaliyah    Try … 2000-03-18      59    53    38    28    21    18    16    14
10 Adams, Yo… Open… 2000-08-26      76    76    74    69    68    67    61    58
# ℹ 307 more rows
# ℹ 68 more variables: wk9 <dbl>, wk10 <dbl>, wk11 <dbl>, wk12 <dbl>,
#   wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>, wk18 <dbl>,
#   wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>, wk24 <dbl>,
#   wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>, wk30 <dbl>,
#   wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>, wk36 <dbl>,
#   wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, wk42 <dbl>, …
```

20. What are the dimensions of the data frame? What are the values stored in the `wk` columns? Why does it have `wk76` column? Weren't there only 52 weeks in 2000? (see `?billboard`)

21. This data frame possess a strange quality: the week data is stored across column names instead of in the values of a column called `week`. Pivot `billboard` to create this alternative (more useful) representation[1]. Make a sketch of the data frame, indicate its dimensions and the columns that it contains.

22. Remove all rows from the new data frame that are unranked in a given week. This can either be done using `drop_na()` or more elegantly back in your `pivot_()` function by adding another argument that takes care of it (see the help file for ideas). Provide the code that you used here.

23. Using this reshaped data frame, make a line plot[2] that shows how each track moves through ranks from one week to the next. Provide the code used to construct your plot as well as a sketch of the result.

To really polish your graphic:

- Add transparency to the lines to prevent overplotting
- Add useful labels
- Modify the theme if you wish
- *Challenge*: Invert the y-axis so that the top ranks (lowest numbers) are at the top of the y-axis.
- *Challenge*: Similar to the COVID plot seen in lecture, pick one line to highlight in a bright color and color the rest of the lines a more subtle color. This could be your favorite track, an example of a "slow burn" track that took a long time to reach the top, a "flash in the pan" track that gained fast and faded fast, or the track that stayed at number 1 for the longest.

[1] Hint: there is a handy shortcut that saves you from typing all of the column names: `starts_with()` inside the `tidyselect` package

[2] Note that `geom_line()` permits the use of a `group` aesthetic mapping to make one line for each group.