Homework 2 Solution

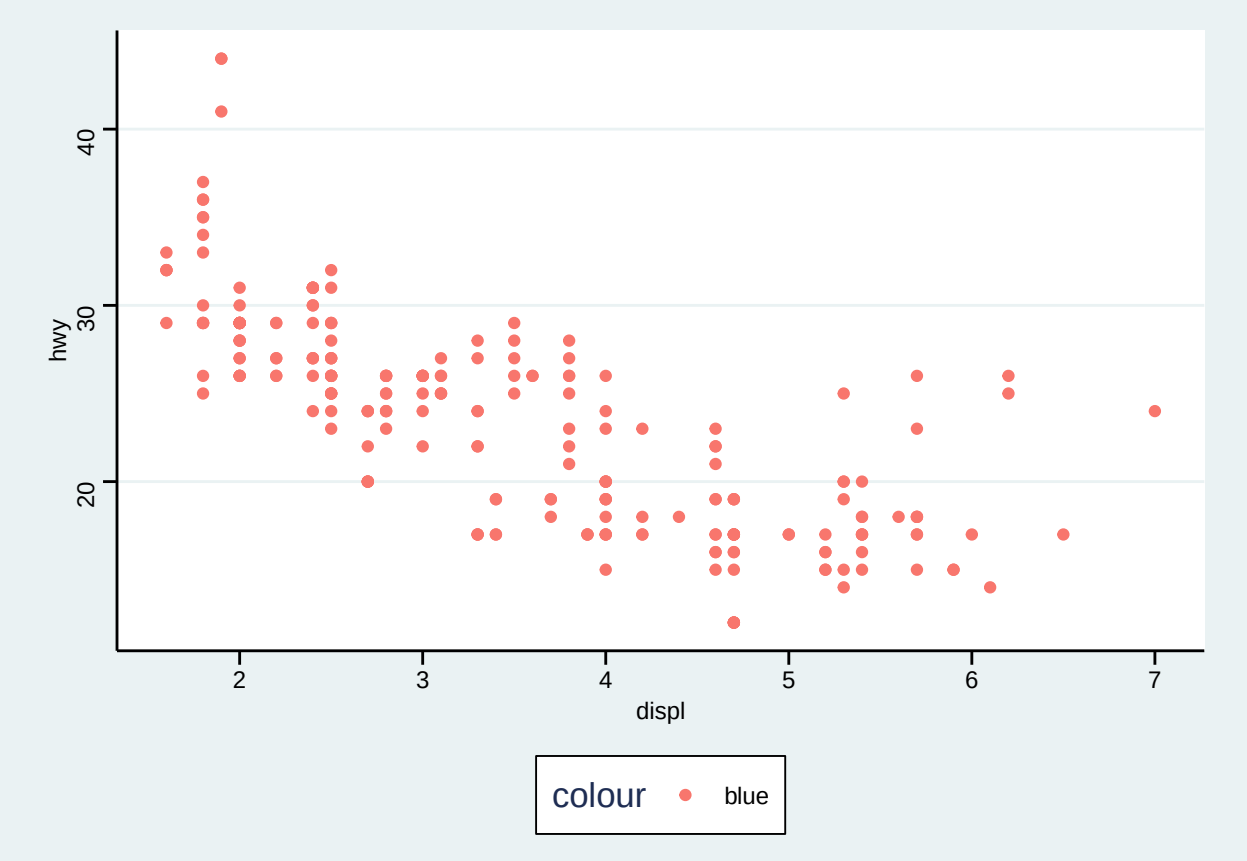
Disclaimer

This homework solution is for the sole benefit of students taking Stat 220 from Prof. Bastola during Spring term 2024. Dissemination of this solution to people who are not registered for this course is not permitted and will be considered grounds for Academic Dishonesty for the all individuals involved in the giving and receiving of the solution.

Problem 1: Spot the error

Explain why the following command does not color the data points blue, then write down the command that will turn the points blue.

```
> library(ggplot2)
> ggplot(data = mpg) +
+   geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



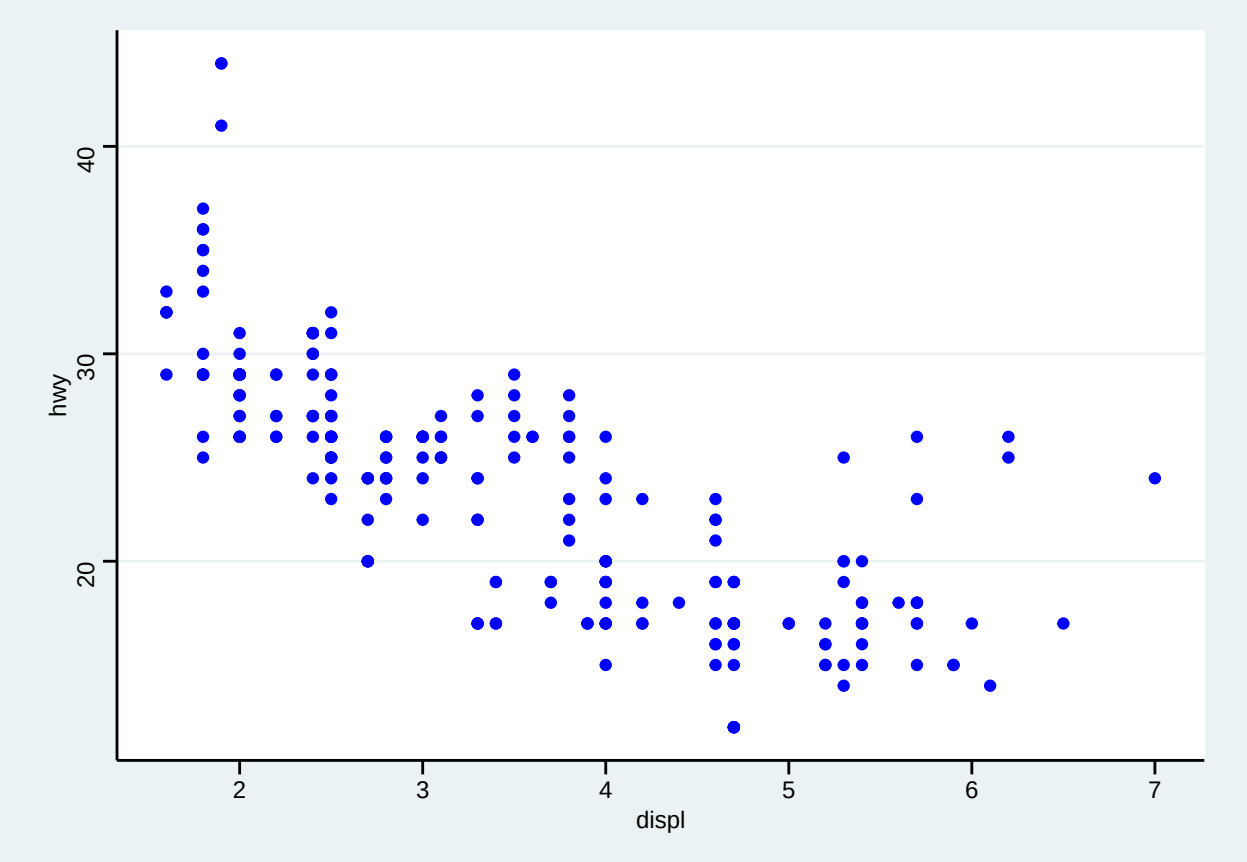
answer:

The `aes` aesthetic command defines how variables in the data set map to positions in the graph. Using `color` in the `aes` results in ggplot thinking that the character vector `"blue"` is a discrete variable value. It is

assigned to all data points and used as a "discrete" variable value when coloring the graph. The first color choice for the default `scale_color_discrete` palette is the red you see in the graph.

To correct the graph, move the color choice outside the `aes`:

```
> ggplot(data = mpg) +
+   geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



Problem 2: Penguins

Load the data `penguins` using the command

```
> # install.packages("palmerpenguins")  # run once then DELETE FROM RMD
> data(penguins, package = 'palmerpenguins')
```

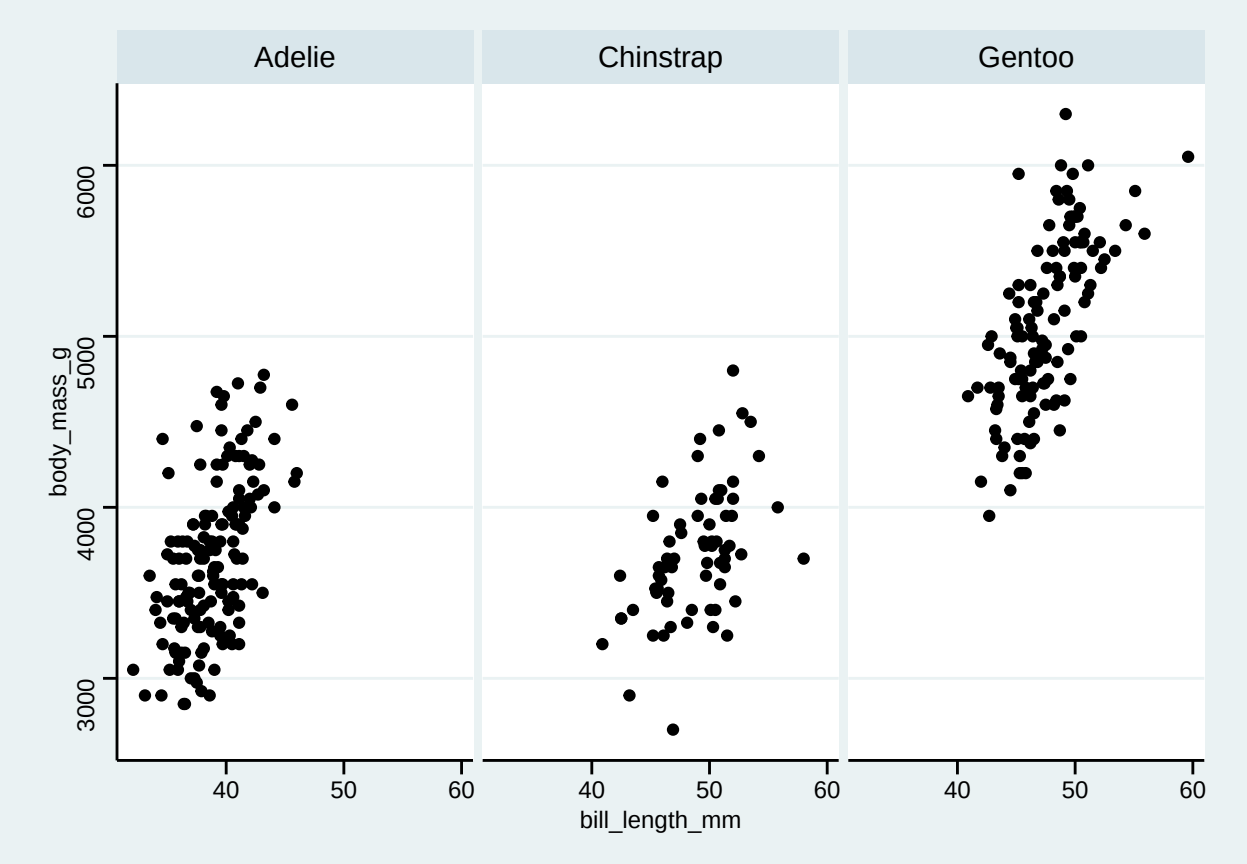
Look at help file for for variable info: `?palmerpenguins::penguins`.

a.

Create a scatterplot of `body_mass_g` (y) against `bill_length_mm` (x) and separate the plot into facets by `species`.

answer:

```
> g <- ggplot(penguins, aes(x = bill_length_mm, y = body_mass_g))
> g + geom_point() +
+   facet_wrap(~species)
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```

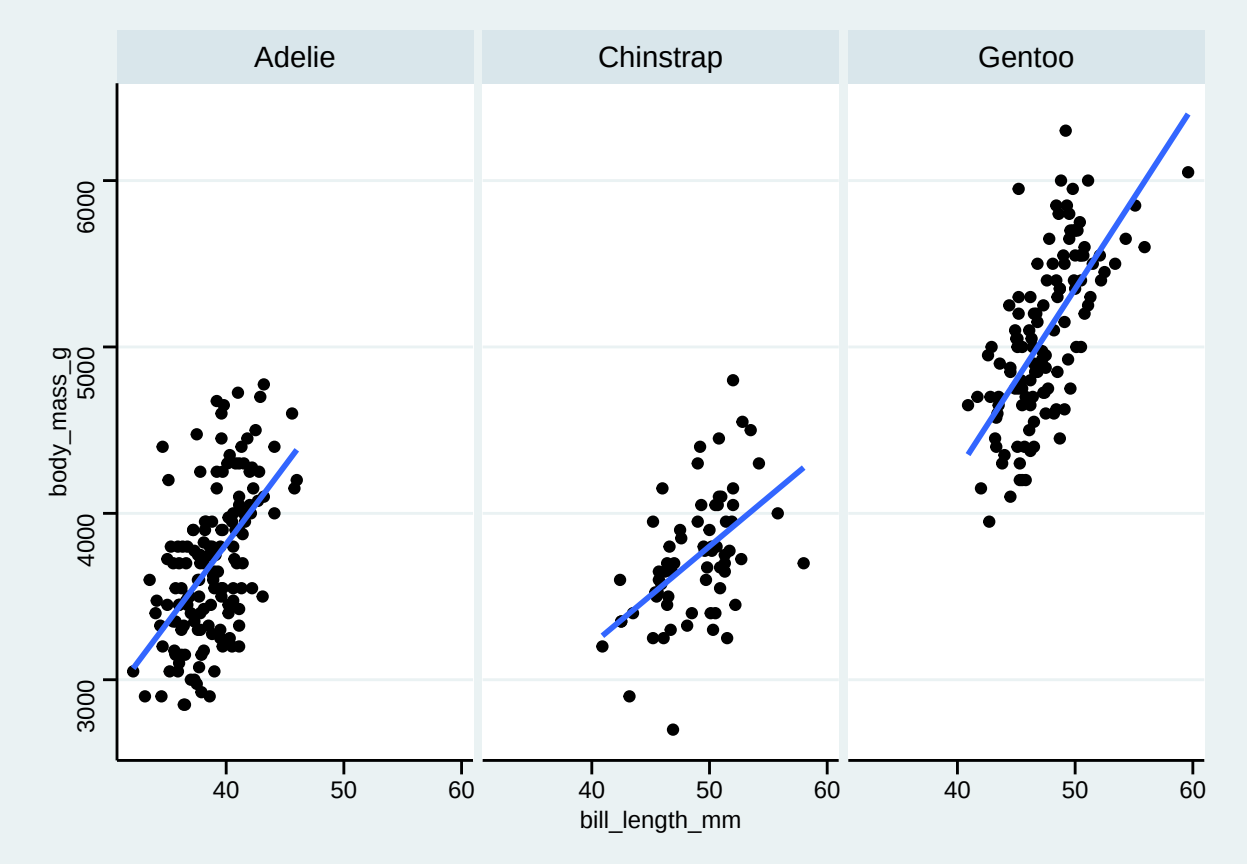


b.

Add regression lines to the plot in (a) for each species with no standard error shading. See `?geom_smooth` to see how to add a regression (`lm`) line instead of a wigglely smoother.

answer:

```
> g + geom_point() +
+   facet_wrap(~species) +
+   geom_smooth(method = lm, se = FALSE)
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```

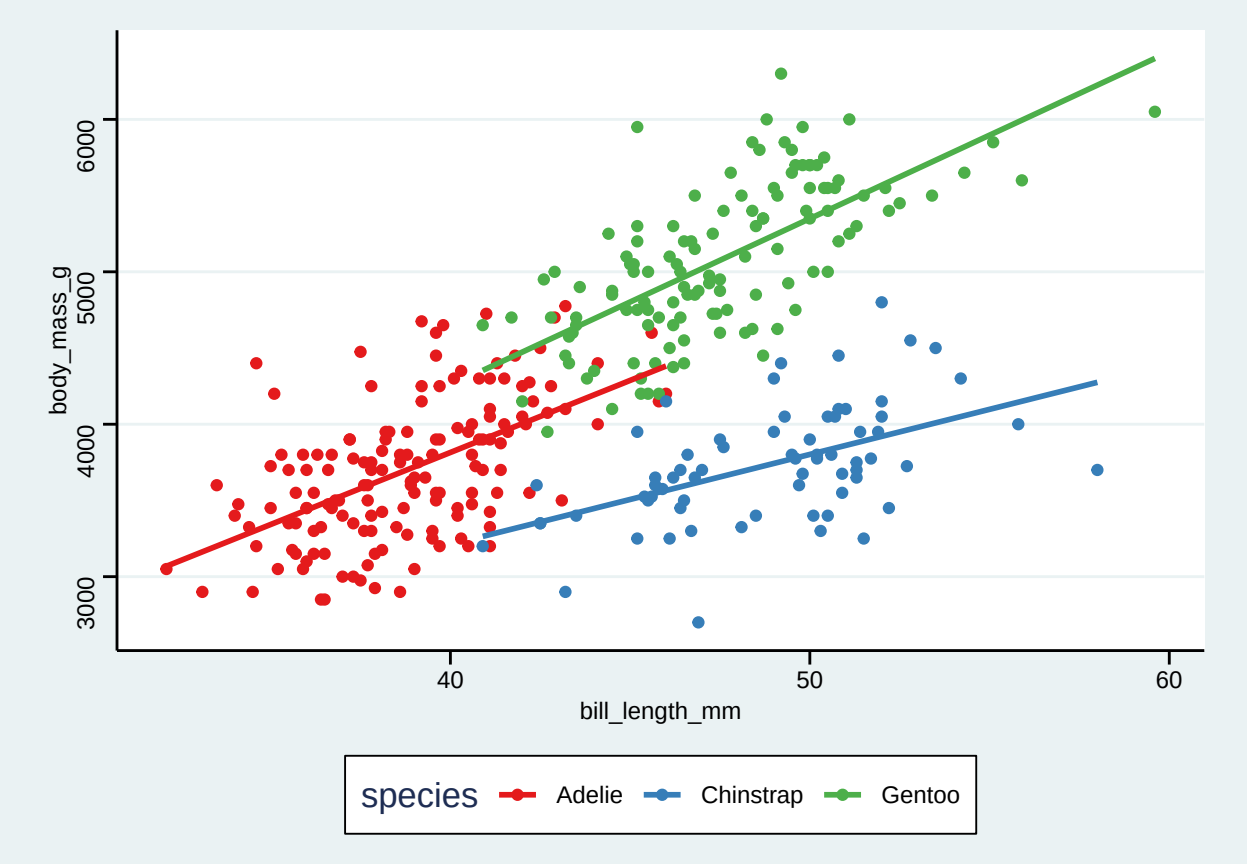


c.

Instead of faceting by `species`, use `species` to color the points and regression lines. Use a color scheme for the points that is not the default coloring.

answer:

```
> g + geom_point(aes(color = species)) +
+   geom_smooth(aes(color = species), method = lm, se = FALSE) +
+   scale_color_brewer(palette = "Set1")
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_smooth()`).
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_point()`).
```



d.

Which visual, (b) or (c), makes it easiest to compare the **slopes** of the lines for each species? Explain.

answer:

The plot in (c) is easier to directly compare slopes on the same x/y scales.

Problem 3: Storm paths by year

Install the package `nasaweather`, if needed, and load the `storms` data.

```
> # install.packages("nasaweather")
> data(storms, package = "nasaweather")
```

a.

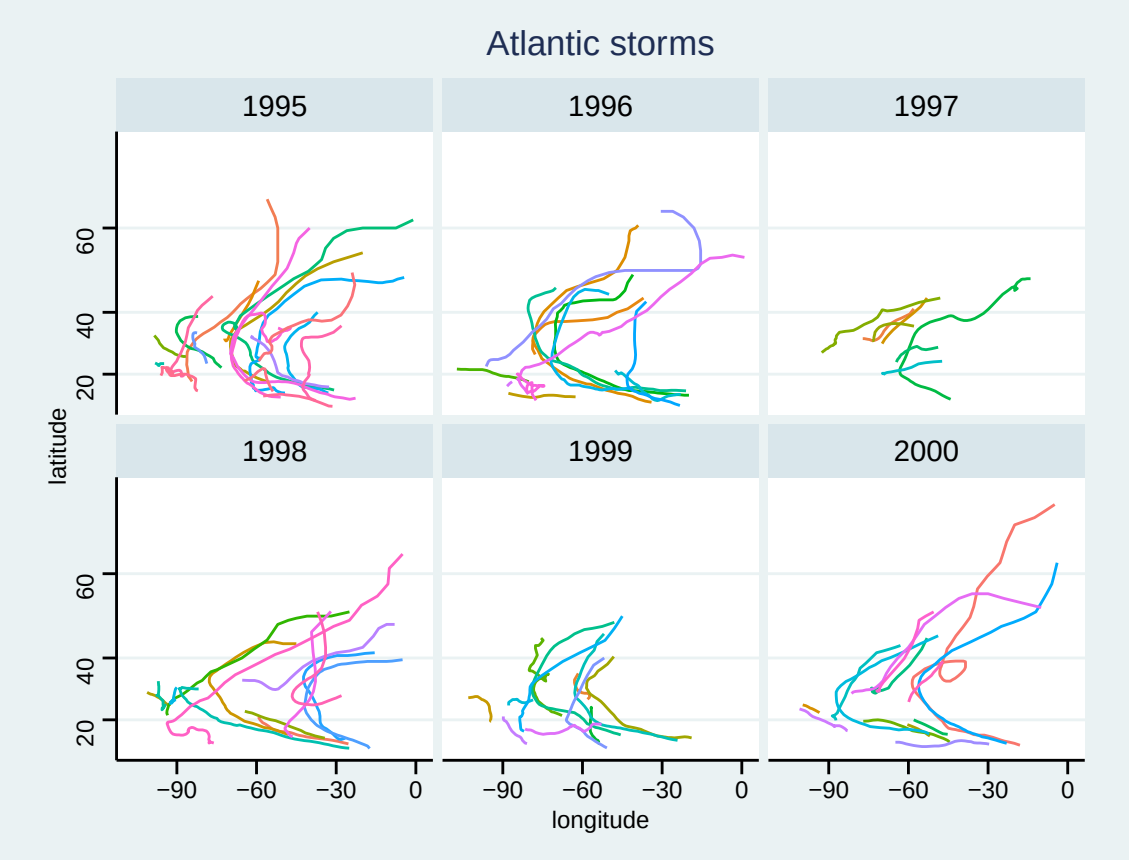
Use `geom_path()` to plot the path of each tropical storm in this data set where you

- use color to distinguish the storms from one another
- use faceting to plot each `year` on its own panel
- change your coordinate system to a map

answer: The `group` argument is needed so that one path is drawn for each named storm.

```
> dplyr::glimpse(storms)
Rows: 2,747
Columns: 11
```

```
$ name     <chr> "Allison", "Allison", "Allison", "Allison", "Allison", "Allis~
$ year     <int> 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1995, 1~
$ month    <int> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6~
$ day      <int> 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8~
$ hour     <int> 0, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, 18, 0, 6, 12, 18, 0, 6,~
$ lat      <dbl> 17.4, 18.3, 19.3, 20.6, 22.0, 23.3, 24.7, 26.2, 27.6, 28.5, 2~
$ long     <dbl> -84.3, -84.9, -85.7, -85.8, -86.0, -86.3, -86.2, -86.2, -86.1~
$ pressure <int> 1005, 1004, 1003, 1001, 997, 995, 987, 988, 988, 990, 990, 99~
$ wind     <int> 30, 30, 35, 40, 50, 60, 65, 65, 65, 60, 60, 45, 30, 35, 35, 4~
$ type     <chr> "Tropical Depression", "Tropical Depression", "Tropical Storm~
$ seasday  <int> 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8~
> storm_plot <-  ggplot(storms, aes(x = long, y = lat)) +
+   geom_path(aes(group = name, color = name)) +
+   facet_wrap(~year) +
+   scale_color_discrete(guide = "none") +
+   labs(
+     x = "longitude",
+     y = "latitude",
+     title = "Atlantic storms"
+     )
> storm_plot + coord_map()
```



b.

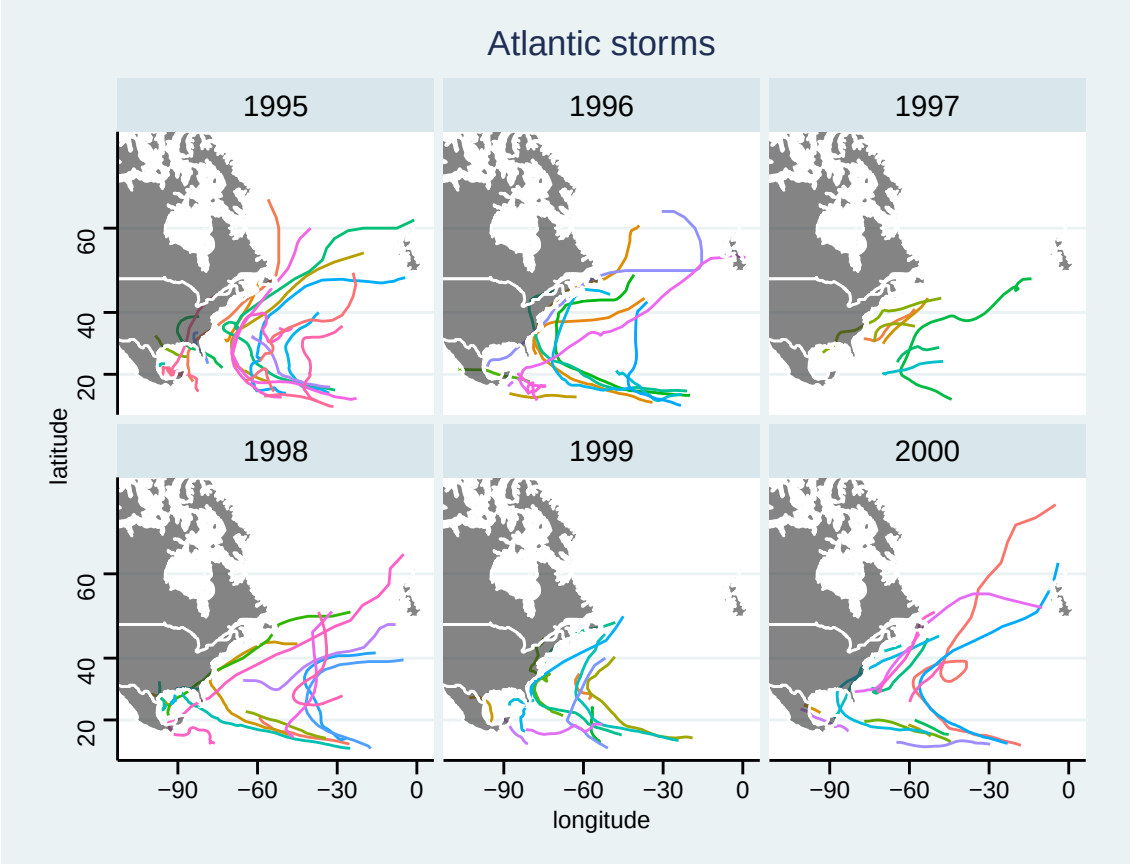
Add a map of the US, Mexico, Canada and the UK to your map in (a) so you can get a better idea where these storms are traveling, truncating the longitude as suggested in the hints.

Hints:

- Look at the help file for `?map_data`. To create map data with only the counties around the storm paths, use data from the the `world` map from `map_data` with the following `region`s specified: `usa`, `mexico`, `canada` and `uk`.
- Add the map data for the regions of interest using `geom_polygon`. (Hint: make sure your paths map doesn't have `color` as a global `aes`.)
- Modify your `coord_map` from part (a) to add the arguments `xlim` and `ylim` to set the min and max limits of your graph to the min/max latitude and longitude of the `storms`.

answer: First pull the map data for the countries of interest

```
> ctry <- map_data("world",
+                  region = c(
+                    "usa",
+                    "mexico",
+                    "canada",
+                    "uk"
+                  ))
> dplyr::glimpse(ctry)
Rows: 19,331
Columns: 6
$ long      <dbl> -59.78760, -59.92227, -60.03775, -60.11426, -60.11748, -59.9~
$ lat       <dbl> 43.93960, 43.90391, 43.90664, 43.93911, 43.95337, 43.93960, ~
$ group     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, ~
$ order     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 18, 19, 2~
$ region    <chr> "Canada", "Canada", "Canada", "Canada", "Canada", "Canada", ~
$ subregion <chr> "Sable Island", "Sable Island", "Sable Island", "Sable Islan~
> storm_plot +
+   geom_polygon(data = ctry,
+                aes(x = long,
+                    y = lat,
+                    group = group), alpha = 0.6, color = "white") +
+   coord_map(xlim  = c(min(storms$long), max(storms$long)),
+             ylim  = c(min(storms$lat), max(storms$lat)))
```



Problem 4: explain command (no R needed)

Consider the data set shown in the table below. Each row represents a class and `classType` of `S` denotes a statistics class and `C` denotes a CS class. The variable `m` counts the number of mac users in the class and `w` counts the number of windows users.

classType	m	w
C	10	4
C	3	1
C	7	3
S	2	7
S	7	10

What data set will be produced by the following commands? Describe (**in words**) how the original data set is being modified and show what it looks like using an R Markdown table (like the one used above) to display the new data set. Assume the original data set is named `mydata`. (No credit will be given creating the fake data frame and just running the code chunks.)

a.

```
> mydata %>%
+   filter(classType == "C") %>%
+   select(m, w)
```

answer: This data set only looks at the number of mac/windows users in CS classes.

m	w
10	4
3	1
7	3

b.

```
> mydata %>%
+   mutate(ratioW = w/sum(w))
```

answer: The `sum(w)` denominator equals 25, so the ratios are 4/25, 1/25, etc. The value of 25 gives the total number of windows users in all classes so `ratioW` records the proportion of all windows users who are in a given class.

classType	m	w	ratioW
C	10	4	0.16
C	3	1	0.04
C	7	3	0.12
S	2	7	0.28
S	7	10	0.40

c.

```
> mydata %>%
+   group_by(classType) %>%
+   mutate(ratioW = w/sum(w))
```

answer: The `sum(w)` denominator equals the sum of `w` for each `classType` group. The total for `C` classes is 8 windows users in all CS classes listed and for `S` classes is 17 windows users in stats classes. `ratioW` records the proportion of all windows users in a given class type (CS or stats) who are in a specific class.

classType	m	w	ratioW
C	10	4	0.50
C	3	1	0.125
C	7	3	0.375
S	2	7	0.412
S	7	10	0.588

d.

```
> mydata %>%
+   group_by(classType) %>%
+   summarize(Y = sum(w+m))
```

answer: We are grouping by `classType` then summarizing the class totals `w+m` with the `sum` function. This will produce a small data frame that gives the total number of CS and stats students:

classType	Y
C	28

classType	Y
S	26

e.

```
> mydata %>%
+   group_by(classType) %>%
+   mutate(X = w+m, Y = sum(w+m))
```

answer:

We are grouping by `classType` which will only effect the output for any command that summarizes variables like `sum`. The `mutate` command preserves the original data frame and adds the variable `X` that records the total number of mac and windows users in a given class (row). The `Y` variable sums the values of `X` for all entries in each `classType` (grouping variable). These values are repeated (28 CS students and 26 stats students) for all classes in each `classType`.

classType	m	w	X	Y
C	10	4	14	28
C	3	1	4	28
C	7	3	10	28
S	2	7	9	26
S	7	10	17	26
