Test Case: Hurricane Data

Brooke Anderson (BA), Rachel Severson (RS)

Main task

Here are some recent tropical storms that were very severe with severe impacts to humans:

Storm	Landfall time	Landfall location	Some places affected
Hurricane Andrew	0905 UTC 24 August 1992	near Homestead AFB FL	Miami, Key Largo, Nassau (Bahamas)
Labor Day Storm	late on September 2, 1935	Florida Keys	Florida Keys, St. Petersburg, Florida
Cyclone Tracy	very early December 25, 1974	Darwin, Australia	Darwin
Tropical Storm Bilis	12:50 p.m. local time July 14, 2006	near Fuzhou, China	Guangdong, Hunan, Fujian

You will be trying to collect relevant data for one of these storms.

Relevant NOAA weather products

In later sections, you will have specific goals to try to achieve, in terms of what data to get. However, as you work on this project, please keep a running list here of any of the NOAA data products that you think might have data that could be used to explore the storm you're working on. Include the name of the dataset, a short description of what you might be able to get from it, and a link to any websites that give more information.

- BA: The IBTracS dataset gives data on tropical storm tracks. It can be pulled through the `rnoaa` function `storm_data`.
- BA: The GHCN-Daily dataset has daily measurements of precipitation and, for some monitors, temperature and wind speed, internationally. It sounds like coverage is best in the US, Australia, and Canada, but there are monitors worldwide. Several `rnoaa` functions work with this data: `meteo_nearby_stations` (find all the monitors within a certain radius of a lat-lon, or find the [x] closest monitors to a point), `meteo_pull_monitors`, and `ghcnd_search`. The function `weather_fips` in `countyweather` (a package Rachel's developing on GitHub) can pull and aggregate data from this source from all monitors in a county based on the county's FIPS code.

BA: Note: For many of the `rnoaa` functions, you will need an API key from NOAA. Here's how you should save that on your computer so that you can get this document to compile without including your API key (these instructions are adapted from here:

1. Ask NOAA for an API key: http://www.ncdc.noaa.gov/cdo-web/token
2. Open a new text file. Put the following in it (note: this might not show up here, but there should be a second blank line):

```
noaakey == [your NOAA API key]
```

3. Save this text file as `.Renviron` in your home directory. Your computer might be upset that the file name starts with a dot. That's okay– it should; ignore the warning and save like this.
4. Restart R.
5. Now you can pull your key using `Sys.getenv("noaakey")`

Now you can gear up to use functions that require the API key:

```
options("noaakey" = Sys.getenv("noaakey"))
```

Relevant other data sources

As you work on this project, also keep a running list here of any data from sources other than NOAA that you think might have data that could be used to explore the storm you're working on. Include the name of the dataset, a short description of what you might be able to get from it, and a link to any websites that give more information and / or the data itself.

- BA: The USGS data on streamflow could be relevant for assessing flooding. This is available through the `waterData` package in R.

Specific tasks

As you work on these, include the code that you try. Include things that worked and things that you thought would work but that did not. Also write down what parts were easy to figure out how to do, and which you had to search around a lot to figure out what to do.

Winds at landfall

Get any data you can that gives measurements of winds when the storm made landfall (use the landfall listed for the storm in the table– often, there are more than one for a storm for different locations, but just try for the listed one).

Try to get:

- A measure of how strong winds were over land around where the storm made landfall
- A measure of how strong winds were over water around where the storm made landfall
- Wind directions at different locations (on land or over water) near landfall
- An estimate of how many of the stations near the landfall that were operating the week before the storm that were still operational and recording data at landfall

A measure of how strong winds were over land around where the storm made landfall

BA: It may work, for daily wind speeds, to use the Global Historical Climatology Network Daily data. If you have monitor ids, you can use the `meteo_pull_monitors` function from `rnoaa` to pull data from this source. If you have the FIPS code for a county, you can use the function `fips_stations` in `countyweather` (a package Rachel is developing) to get the station IDs for all relevant stations in a county.

For example, the FIPS for Miami-Dade is 12086. Based on the README file for this data, some of the windspeed variables have the abbreviations: "AWND", "WSFG", "WSF1". You can therefore run:

```
# library(devtools)
# install_github("ropenscilabs/rnoaa") # if you need to install the package
# install_github("leighseverson/countyweather") # if you need to install the package
library(rnoaa)
library(countyweather)
miami_stations <- fips_stations("12086", date_min = "1992-08-01",
                                date_max = "1992-09-01")
miami_stations
```

```
## character(0)
```

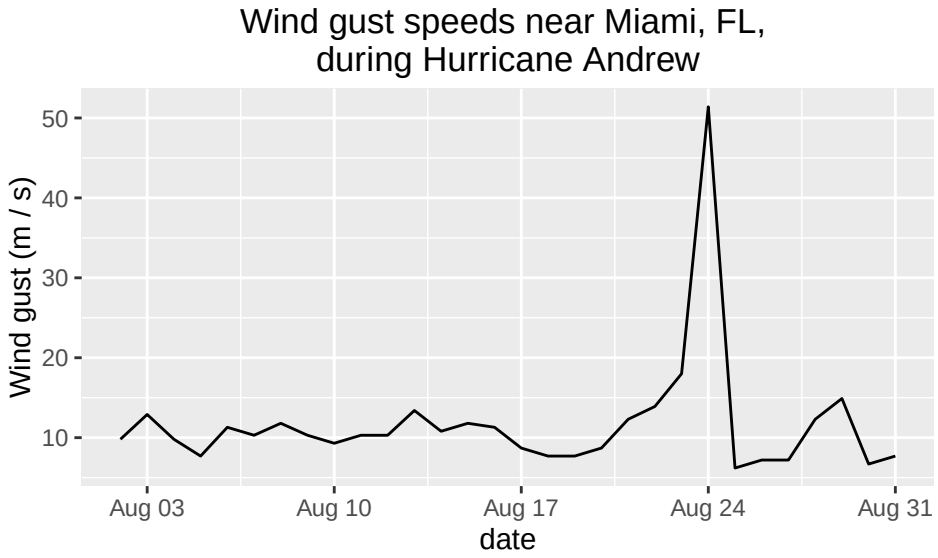
This does not pull any weather stations. However, I know that the station "USW00012839" should exist for this time– it's the Miami Intl Airport station. I wonder why this isn't picked up by `fips_stations`?

Once you know the station number, you can run:

```
miami_wind <- meteo_pull_monitors("USW00012839", date_min = "1992-08-01",
                                  date_max = "1992-09-01",
                                  var = c("AWND", "WSFG", "WSF1")) %>%
  mutate(wsfg = wsfg / 10) # Convert to m / s-- see the README for this data from link
head(miami_wind)
```

```
## Source: local data frame [6 x 5]
##
##            id       date  awnd  wsf1  wsfg
##         (chr)     (date) (int) (int) (dbl)
## 1 USW00012839 1992-08-02    38    67   9.8
## 2 USW00012839 1992-08-03    27    94  12.9
## 3 USW00012839 1992-08-04    34    76   9.8
## 4 USW00012839 1992-08-05    33    54   7.7
## 5 USW00012839 1992-08-06    33    98  11.3
## 6 USW00012839 1992-08-07    48    80  10.3
```

```
ggplot(miami_wind, aes(x = date, y = wsfg)) +
  geom_line() + ggtitle("Wind gust speeds near Miami, FL, \nduring Hurricane Andrew") +
  ylab("Wind gust (m / s)")
```



It seems like we should be able to get wind data from some of the other NOAA data sources. In particular, it seems like we should be able to get hourly, or another fine temporal resolution, of wind data. Check the "Data sources" section of the file at the bottom of the page here.

A measure of how strong winds were over water around where the storm made landfall

BA: For this, it seems like buoy data might work. The `buoy` function in `rnoaa` might work. I'm not sure how you could get station IDs through `rnoaa` for the buoy stations. I used the website to find a station close to Miami ("fwyf1"). Once you have a buoy id, you can do this kind of call:

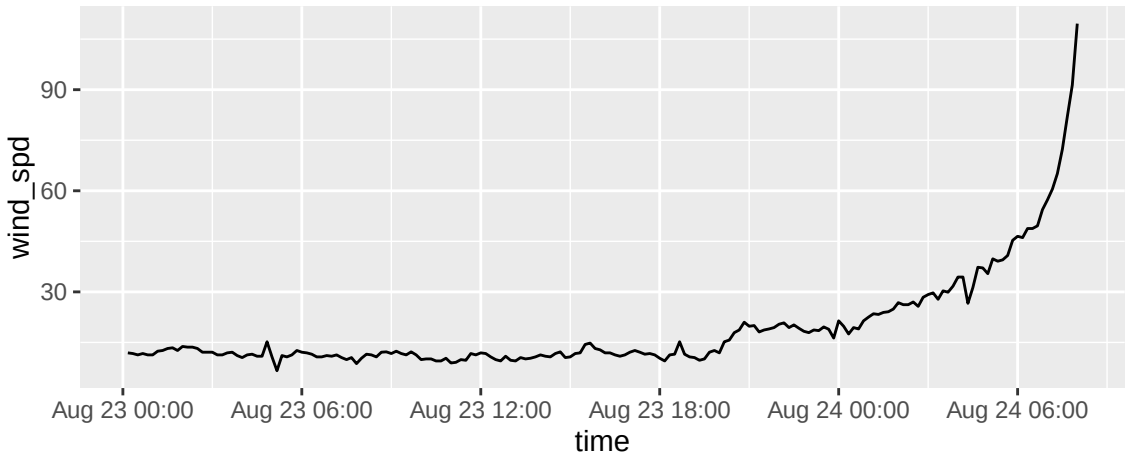
```
ex <- buoy(dataset = 'cwind', buoyid = "fwyf1", year = 1992)
```

```
## Using c1992.nc
```

```
head(ex$data)
```

```
##                   time   lat   lon wind_dir wind_spd
## 1 1991-12-31T23:10:00Z 25.59 -80.1       19     10.3
## 2 1991-12-31T23:20:00Z 25.59 -80.1       19     10.3
## 3 1991-12-31T23:30:00Z 25.59 -80.1       12      9.3
## 4 1991-12-31T23:40:00Z 25.59 -80.1       12     10.1
## 5 1991-12-31T23:50:00Z 25.59 -80.1        6      9.9
## 6 1992-01-01T00:00:00Z 25.59 -80.1        7      9.3
```

```
fl_buoy <- ex$data %>% mutate(time = ymd_hms(time)) %>%
  filter(time > ymd_hms("1992-08-23 00:00:00") &
           time < ymd_hms("1992-08-26 00:00:00"))
ggplot(fl_buoy, aes(x = time, y = wind_spd)) + geom_line()
```



It looks like this buoy was actually lost during the hurricane. It stops recording during the worst of the storm, and it doesn't look like it got back online in 1992 after that.

It looks like you need to look through the historical station maps to find a station, rather than the current ones. If there's somewhere we could get a list of all buoy ids by latitude and longitude, we could identify the ones closest to a location.

Wind directions at different locations (on land or over water) near landfall

An estimate of how many of the stations near the landfall that were operating the week before the storm that were still operational and recording data at landfall

Precipitation at affected cities

For each of the affected cities, estimate the precipitation during the storm and on neighboring days. Include:

- Daily and hourly estimates of rainfall
- How many stations you used to get each of those values
- A map of stations you used to get those values, with some measure on the map of the maximum daily or hourly rainfall measured at that station

Daily and hourly estimates of rainfall

RS: The countyweather package has some functions for pulling and aggreagating weather data by FIPS code. For the US storms, this might be helpful for this task. If you have devtools installed, you can install countyweather using the following code:

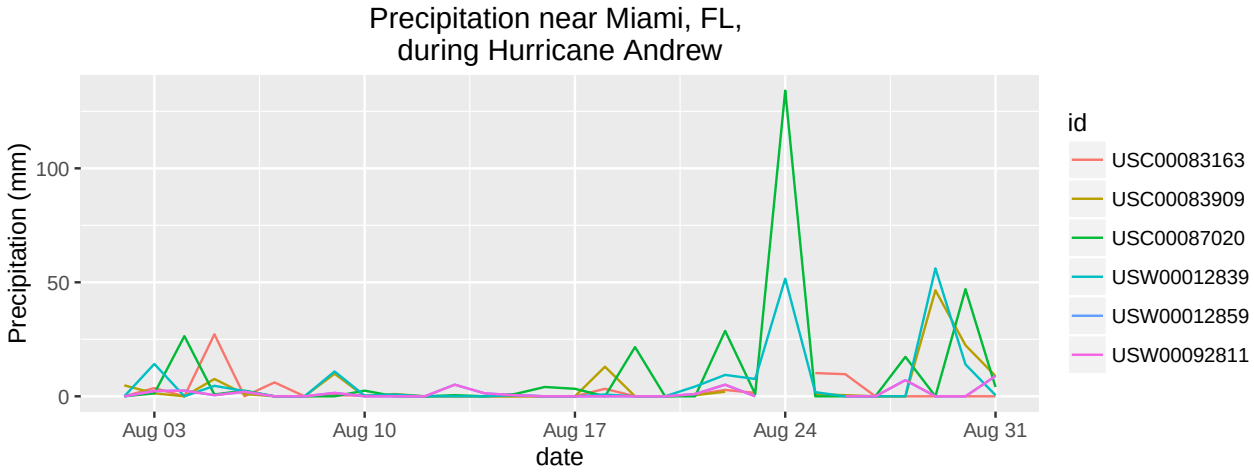
```
# library(devtools)
# install_github("ropenscilabs/rnoaa") # if you need to install the package
# install_github("leighseverson/countyweather") # if you need to install the package
library(rnoaa)
library(countyweather)
miami_stations <- fips_stations(fips = "12086")
miami_rain <- meteo_pull_monitors(miami_stations, var = "PRCP")
range(miami_rain$date)
```

BG: For some reason, NCDC seems to only be identifying Miami weather monitors that operated from 2007 on.

You can instead use the `meteo_nearby_stations` function to find all precipitation monitors in a certain radius (parts of this take a long time to run, so I won't evaluate):

```
station_data <- ghcnd_stations()[[1]] # Takes a while to run
miami <- data.frame(id = "miami", latitude = 25.7617, longitude = -80.1918)
miami_monitors <- meteo_nearby_stations(lat_lon_df = miami,
                                        station_data = station_data,
                                        radius = 50,
                                        var = c("PRCP"),
                                        year_min = 1992, year_max = 1992)
miami_monitors <- miami_monitors[[1]]$id
```

```
miami_weather <- meteo_pull_monitors(miami_monitors, date_min = "1992-08-01",
                                     date_max = "1992-09-01",
                                     var = c("PRCP"))
ggplot(miami_weather, aes(x = date, y = prcp, color = id)) +
  geom_line() + ggtitle("Precipitation near Miami, FL, \nduring Hurricane Andrew") +
  ylab("Precipitation (mm)")
```



How many stations you used to get each of those values

A map of stations you used to get those values, with some measure on the map of the maximum daily or hourly rainfall measured at that station

Flooding related to the hurricane

Was there severe flooding related to the storm? For locations along the storm path, try to figure out:

- What areas were along the storm path?
- Can you identify streams and rivers that were affected by the storm?
- What cities had problems with flooding during the storm? Get some measure to identify if and to what degree the location flooded.
- How closely were rainfall and flooding severity linked in locations?

What areas were along the storm path?

BG: For this, you'll need to have the storm track for each storm. I guess then match with distance to lat-lon coordinates for city or county centers to identify locations near the storm track?

The `rnoaa` package has some functions to tap into the IBTracS dataset, which has tropical storm tracks. It looks like you might be able to use the `storm_data` function to pull in data on a storm. You can pull for just one storm if you have the storm serial number, but it's not entirely clear to me how you would figure out what this number is for a storm you want to look for.

You can pull by year or basin, but it looks like you can't pull by both at the same time. I'll put just for 1992:

```
# install_github("ropenscilabs/rnoaa") # if you need to install the package
library(rnoaa)
hurrs_1992 <- storm_data(year = 1992)
```

```
## <path>~/.rnoaa/storms/year/Year.1992.ibtracs_all.v03r06.csv
```

It looks like that pulls a list object, with the element `data` that has the actual data:

```
names(hurrs_1992)
```

```
## [1] "data"
```

```
colnames(hurrs_1992$data)[1:10]
```

```
##  [1] "serial_num" "season"     "num"        "basin"      "sub_basin"
##  [6] "name"       "iso_time"   "nature"     "latitude"   "longitude"
```

Maybe we could use the `name` column to look for Andrew?

```
andrew <- hurrs_1992$data[which(hurrs_1992$data$name == "ANDREW"), ]
andrew[1:5, 1:10]
```

```
##         serial_num season num basin sub_basin   name            iso_time
## 2417 1992230N11325   1992   4    NA        MM ANDREW 1992-08-16 18:00:00
## 2418 1992230N11325   1992   4    NA        MM ANDREW 1992-08-17 00:00:00
## 2419 1992230N11325   1992   4    NA        MM ANDREW 1992-08-17 06:00:00
## 2420 1992230N11325   1992   4    NA        MM ANDREW 1992-08-17 12:00:00
## 2421 1992230N11325   1992   4    NA        MM ANDREW 1992-08-17 18:00:00
##      nature latitude longitude
## 2417     TS     10.8     -35.5
## 2418     TS     11.2     -37.4
## 2419     TS     11.7     -39.6
## 2420     TS     12.3     -42.0
## 2421     TS     13.1     -44.2
```

This looks about right...

Note: after digging some more, it looks like you can use the `storm_meta` function with the argument `what = "storm_names"` from `storm_data` to get the serial number for a storm if you know its name:

```
storm_names <- storm_meta(what = "storm_names")
head(storm_names, 3)
```

```
##              id              name
## 1 1842298N11080 NOT NAMED(td9636)
## 2 1845336N10074 NOT NAMED(td9636)
## 3 1848011S09079 NOT NAMED(td9636)
```

```
grep("ANDREW", storm_names$name, value = TRUE)
```

```
## [1] "ANDREW(hurdat_atl) - bal011986(atcf)"
## [2] "ANDREW(hurdat_atl) - bal041992(atcf)"
```

```
storm_names[grep("ANDREW", storm_names$name), ]
```

```
##                 id                                  name
## 9276 1986156N26284 ANDREW(hurdat_atl) - bal011986(atcf)
## 9963 1992230N11325 ANDREW(hurdat_atl) - bal041992(atcf)
```

```
andrew_id <- storm_names[grep("ANDREW", storm_names$name), 1][2]
andrew_2 <- storm_data(storm = andrew_id)
```

```
## <path>~/.rnoaa/storms/storm/Storm.1992230N11325.ibtracs_all.v03r06.csv
```

```
head(andrew_2$data[ , c("iso_time", "latitude", "longitude", "wind.wmo.")], 3)
```

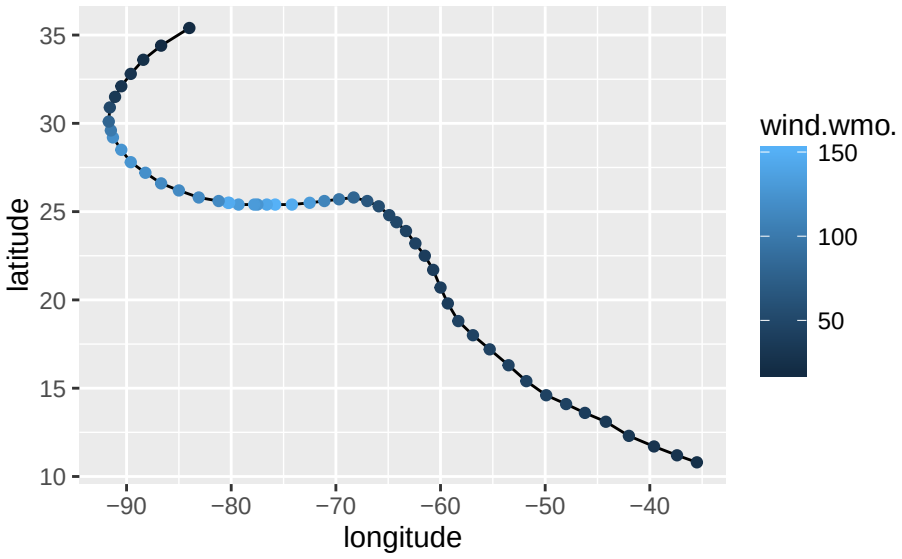
```
##              iso_time latitude longitude wind.wmo.
## 1 1992-08-16 18:00:00     10.8     -35.5        25
## 2 1992-08-17 00:00:00     11.2     -37.4        30
## 3 1992-08-17 06:00:00     11.7     -39.6        30
```

Let's see what happens if we plot the locations of this:

```
andrew_loc <- select(andrew, iso_time, latitude, longitude, wind.wmo.)
head(andrew_loc, 3)
```

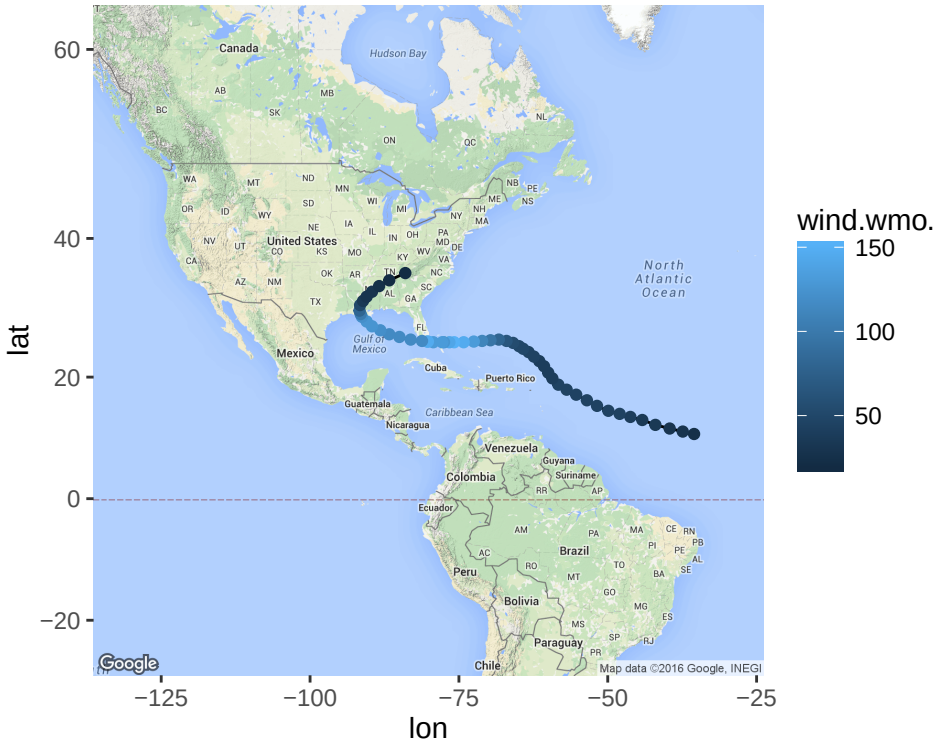
```
##                 iso_time latitude longitude wind.wmo.
## 2417 1992-08-16 18:00:00     10.8     -35.5        25
## 2418 1992-08-17 00:00:00     11.2     -37.4        30
## 2419 1992-08-17 06:00:00     11.7     -39.6        30
```

```
ggplot(andrew_loc, aes(x = longitude, y = latitude)) +
  geom_path() + geom_point(aes(color = wind.wmo.))
```



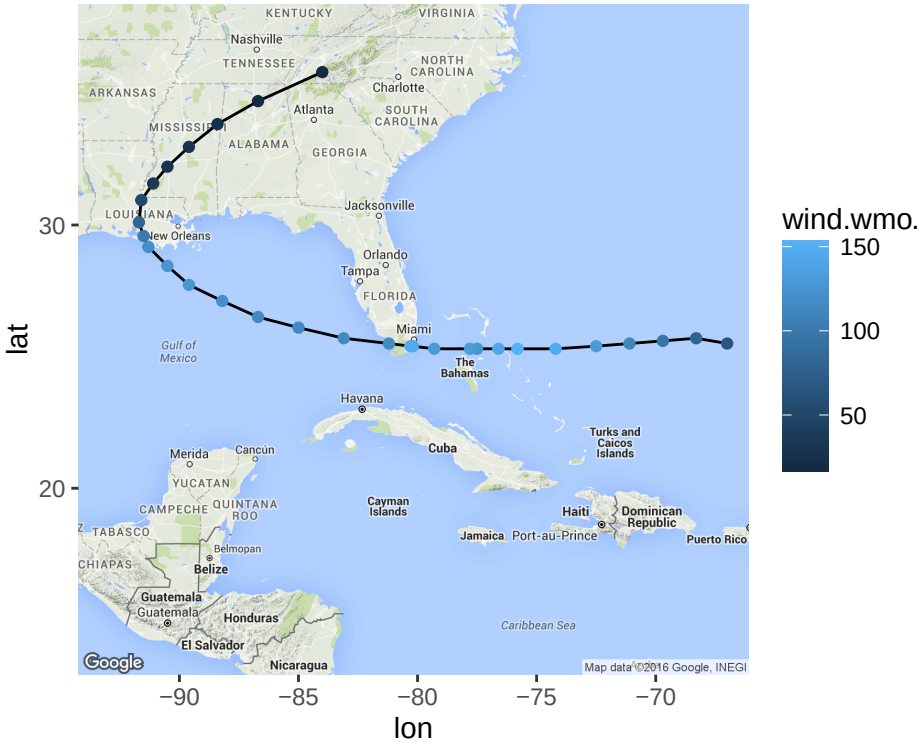
Let me add to a map:

```
library(ggmap)
miami_map <- get_map(location = 'Miami', zoom = 3)
ggmap(miami_map) +
  geom_path(data = andrew_loc, aes(x = longitude, y = latitude)) +
  geom_point(data = andrew_loc, aes(x = longitude, y = latitude,
                                    color = wind.wmo.))
```



And more locally:

```
miami_map <- get_map(location = 'Miami', zoom = 5)
ggmap(miami_map) +
  geom_path(data = andrew_loc, aes(x = longitude, y = latitude)) +
  geom_point(data = andrew_loc, aes(x = longitude, y = latitude,
                                    color = wind.wmo.))
```



Next, we could measure distance from US county centers and pick out the counties that were within [x] kilometers of the 6-hour storm location measures.

Can you identify streams and rivers that were affected by the storm?

BG: No idea where to start here... Are there any databases where you can identify rivers or streams that are close to a certain location? Does streamgage meta-data have some information saying what river or stream it's on? If so, we might be able to pull meta-data on all streamgages through USGS, filter to just ones within the right date range, and then pull the latitude-longitude and the stream / river name, then calculate distances to the central path of the storm from storm tracks.

What cities had problems with flooding during the storm? Get some measure to identify if and to what degree the location flooded.

BG: The `waterData` package allows you to pull daily hydrologic time series from the USGS. You need to have the station IDs to do that first, though.

The package `countyweather` (this is Rachel's package) has a function to pull all the station IDs for a US county, based on the county FIPS code. This function is still in development (so feel free to fork the package and make your own changes!), but for right now you can use it like this– to get the station IDs for Miami-Dade, which has a FIPS code of 12086:

```
# library(devtools)
# install_github("leighseverson/countyweather") # install the package if you need to
library(countyweather)
miami_ids <- countyweather:::streamflow("12086",
                                        date_min = "1992-08-20",
                                        date_max = "1992-08-30")
miami_ids
```

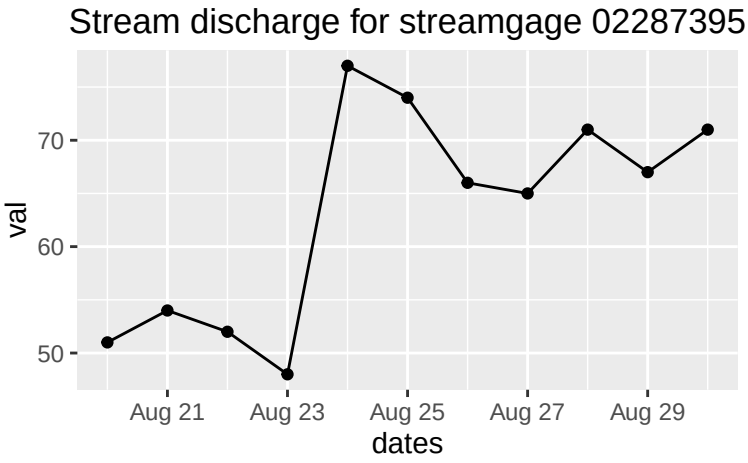
```
##  [1] "02287395"        "02287497"        "02288600"
##  [4] "02289019"        "02289040"        "02289041"
##  [7] "02289050"        "02289060"        "02289500"
## [10] "02290710"        "022907647"       "02290765"
## [13] "02290769"        "252523080352500" "254543080405401"
## [16] "254543080491101"
```

Then you can use these IDs with `importDVs` from the `waterData` package to import daily hydrological time series data. This function will, I think, only pull for one gauge at a time:

```
# install.packages(waterData) # if you need to install the package
library(waterData)
miami_1 <- importDVs(miami_ids[1], sdate = "1992-08-20", edate = "1992-08-30")
head(miami_1)
```

```
##      staid val      dates qualcode
## 1 02287395  51 1992-08-20        A
## 2 02287395  54 1992-08-21        A
## 3 02287395  52 1992-08-22        A
## 4 02287395  48 1992-08-23        A
## 5 02287395  77 1992-08-24        A
## 6 02287395  74 1992-08-25        A
```

```
ggplot(miami_1, aes(x = dates, y = val)) + geom_line() +
  geom_point() + ggtitle("Stream discharge for streamgage 02287395")
```



The default is for `importDVs` to pull in the mean daily value. You can use the `stat` option to change what you get in, although for this, you'd want to look up what all the USGS statistics codes are. You can also

change the USGS parameter code using the `code` argument. The default is "00060", which is the discharge in cubic feet per second. The help file for `importDvs` has web links to find the right codes for these.

So, I think next steps would be to use something from the `apply` family of functions or a loop to go through and get data from all of the Miami gauges, and then plot the discharge for them all. Are there any problems with pulling data from any of the gauges that we get back from the `streamflow` function?

How closely were rainfall and flooding severity linked in locations?

BG: One way to do this would be to try to get county-level aggregated daily estimates of rainfall and flooding intensity. You could do all sorts of things with this. For example, you could take the county maximum of both values for days near the storm and see how well correlated those are across different counties. You could also try to measure how many days are typical between the maximum rain fall and maximum stream flow.

Plotting the storm

Plot the full track of the storm. Show its intensity as it progressed along the track, as well as the affected cities.