Test Case: Hurricane Data

Brooke Anderson (BA), Rachel Severson (RS), Casey Patrizio (CP), Mike Lyons (ML), Joshua Ferreri (JF), Enqun Wang (EW), Holly Kessler (HK)

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:

```r
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.

CP: I had issues running `install_github("ropenscilabs/rnoaa")` in order to use the function `meteo_pull_monitors`, so instead try `load_all()`. Note that this will only work if you have devtools installed and you have forked the rnoaa respository from Github.
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", dependencies=TRUE) # 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
```

```
## [1] "USC00083909" "USC00084095" "USC00087020" "USC00087760" "USC00088780"
## [6] "USR0000FTEN" "USW00012839" "USW00012859" "USW00092811"
```

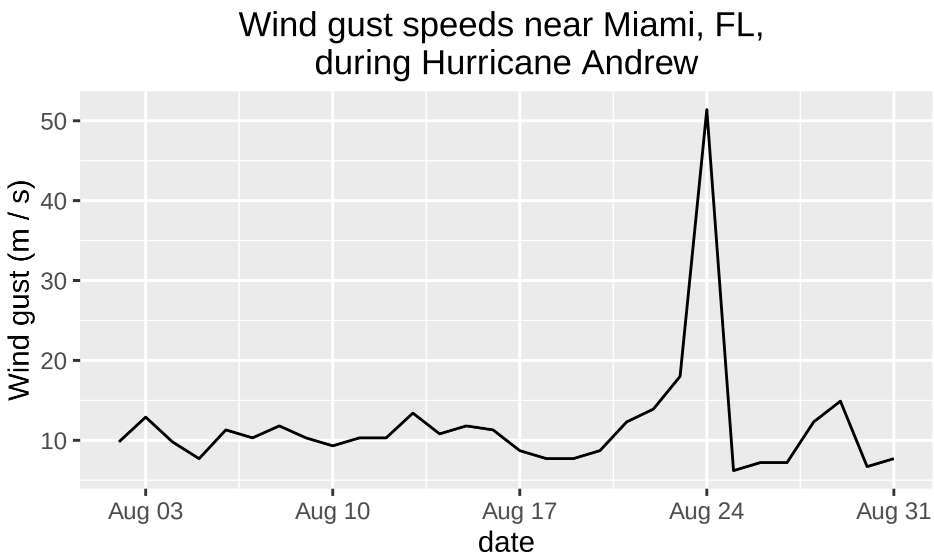
This does not pull any weather stations. However, I know that the station "USW00012839" should exist for this time– it's the Miami Intl Airport station. I wonder why this isn't picked up by `fips_stations`?

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.

RS: The `isd()` function from the rnoaa package looks like it's getting hourly data. Here's an example with the station at Miami International Airport.

CP: A current issue is that the `isd()` function takes USAF and WBAN station codes as input, whereas we would like to use FIPS codes (county codes) as input. My initial thought for a way around this is to use the `isd_stations_search` from the rnoaa package, which takes lat/lon and radius (km) as input. I can use census data from here to find population weighted lat/lon associated with a given FIPS code, and then use the lat/lon as input to `isd_stations_search` to get the stations associated with that FIPS code. Note that we would like to produce output similar to `weather_fips`, a function that takes a county code as input and then gives weather data in the surrounding area. `weather_fips`, except for hourly data. The `weather_fips` code can be forked from here.

```
##Cyclone Tracy
station_data <- ghcnd_stations()[[1]] # Takes a while to run
darwin_ll <- geocode("Darwin, Australia")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Darwin,%20Australia&s
```

```
darwin <- data.frame(id = "darwin",
                     latitude = darwin_ll$lat,
                     longitude = darwin_ll$lon)
darwin_monitors <- meteo_nearby_stations(lat_lon_df = darwin,
                                         lat_colname = "latitude",
                                         lon_colname = "longitude",
                                         station_data = station_data,
                                         limit = 10,
                                         var = c("AWDR", "AWND", "WSFG", "WSF1",
                                                 "FMTM", "MDWM", "WDF1", "WDF2",
                                                 "WDF5", "WDFG", "WDFI", "WDFM",
                                                 "WDMV", "WSF2", "WSF5", "WSFI",
                                                 "WSFM", "WT11"),
                                         year_min = 1974, year_max = 1974)
```

- JF: Tapping into the GHCND, there were no stations at this time in Australia that have recorded wind speed values (of any sort). The closest station was over 2000 miles away.

```
keys_stations <- fips_stations("12087", date_min = "1935-08-15",
                               date_max = "1935-09-15")
keys_monitors <- ncdc_stations(datasetid = "GHCND", locationid = "FIPS:12087",
                            limit = 1000)
keys_monitors <- data.frame(id = keys_monitors$data$id,
                           mindate = keys_monitors$data$mindate,
                           maxdate = keys_monitors$data$maxdate)
keys_monitors$id <- gsub("GHCND:", "", keys_monitors$id)
keys_monitors$mindate <- as.Date(keys_monitors$mindate)
keys_monitors$maxdate <- as.Date(keys_monitors$maxdate)
keys_monitors <- subset(keys_monitors,
                        keys_monitors$mindate <= as.Date("1935-08-01") &
                          keys_monitors$maxdate >= as.Date("1935-09-01")) #still nothing
keys_monitors
```

```
## [1] id      mindate maxdate
## <0 rows> (or 0-length row.names)
```

```
keys <- geocode("Florida Keys, FL")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Florida%20Keys,%20FL&
```

```
keys <- data.frame(id = "keys", latitude = keys$lat, longitude = keys$lon)
keys_monitors <- meteo_nearby_stations(lat_lon_df = keys,
                                       lat_colname = "latitude",
                                       lon_colname = "longitude",
                                       station_data = station_data,
                                       limit = 10,
                                       var = c("AWDR", "AWND", "WSFG", "WSF1",
                                               "FMTM", "MDWM", "WDF1", "WDF2",
                                               "WDF5", "WDFG", "WDFI", "WDFM",
                                               "WDMV", "WSF2", "WSF5", "WSFI",
                                               "WSFM", "WT11"),
                                       year_min = 1935, year_max = 1935)
keys_monitors
```

```
## $keys
##             id                     name latitude longitude distance
## 1  USC00087395           FL PUNTA GORDA  26.9333  -82.0500 265.7253
## 2  USC00080228              FL ARCADIA  27.2181  -81.8739 296.1820
## 3  USC00088786                FL TAMPA  27.9500  -82.4500 383.2446
## 4  USC00085643       FL MERRITT ISLAND  28.3500  -80.7000 435.4439
## 5  USW00012841 FL ORLANDO EXECUTIVE AP  28.5453  -81.3331 445.8657
## 6  USC00086414                FL OCALA  29.1639  -82.0778 513.2394
## 7  USC00085705           FL MIDDLEBURG  30.0603  -81.8475 612.1131
## 8  USC00087440 FL RAIFORD STATE PRISON  30.0678  -82.1928 614.2498
## 9  USC00083470      FL GLEN ST MARY 1 W  30.2717  -82.1856 636.8276
## 10 USC00085099             FL LIVE OAK  30.2889  -82.9650 648.0760
```

```r
pete <- geocode("St. Petersburg, FL")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=St.%20Petersburg,%20F
```

```r
pete <- data.frame(id = "pete", latitude = pete$lat, longitude = pete$lon)
pete_monitors <- meteo_nearby_stations(lat_lon_df = pete,
                                       lat_colname = "latitude",
                                       lon_colname = "longitude",
                                       station_data = station_data,
                                       limit = 10,
                                       var = c("AWDR", "AWND", "WSFG", "WSF1",
                                               "FMTM", "MDWM", "WDF1", "WDF2",
                                               "WDF5", "WDFG", "WDFI", "WDFM",
                                               "WDMV", "WSF2", "WSF5", "WSFI",
                                               "WSFM", "WT11"),
                                       year_min = 1935, year_max = 1935)
pete_monitors
```

```
## $pete
##             id                    name latitude longitude  distance
## 1  USC00088786                FL TAMPA  27.9500  -82.4500  28.06212
## 2  USC00080228              FL ARCADIA  27.2181  -81.8739  95.06357
## 3  USC00087395          FL PUNTA GORDA  26.9333  -82.0500 107.37257
## 4  USW00012841 FL ORLANDO EXECUTIVE AP  28.5453  -81.3331 154.50052
## 5  USC00086414                FL OCALA  29.1639  -82.0778 165.93144
## 6  USC00085643       FL MERRITT ISLAND  28.3500  -80.7000 200.43065
## 7  USC00087440 FL RAIFORD STATE PRISON  30.0678  -82.1928 260.96431
## 8  USC00085705          FL MIDDLEBURG  30.0603  -81.8475 267.66009
## 9  USC00083470     FL GLEN ST MARY 1 W  30.2717  -82.1856 283.46057
## 10 USC00085099             FL LIVE OAK  30.2889  -82.9650 284.01978
```

```r
pete_monitors <- pete_monitors[[1]]$id
pete_monitors <- pete_monitors[1]
pete_weather <- meteo_pull_monitors(pete_monitors, date_min = "1935-08-15",
                                     date_max = "1935-09-15",
                                     var = c("AWDR", "AWND", "WSFG", "WSF1",
                                             "FMTM", "MDWM", "WDF1", "WDF2",
                                             "WDF5", "WDFG", "WDFI", "WDFM",
                                             "WDMV", "WSF2", "WSF5", "WSFI",
                                             "WSFM", "WT11"))
```

-JF: using `fips_stations`, was unable to find any weather station for the Florida Keys for 1935. Using `meteo_nearby_stations`, found the closest station to have recorded any sort of wind data to be USC00088786 in Tampa, FL. After pulling data for this station in September 1935, recieved data for `wt11` only, which seems to be potentially binary where 1 indicated high or damaging winds and NA is recorded for days where such winds are not observed.

RS: Thanks to CP's contributions above, here's a first stab at a function putting all of this together (sadly unfinished as of 5pm 4/21 but I'll plan to keep working on it!) The goal is to get averaged hourly data across multiple stations for a given FIPS, year, and desired weather variables.

```r
library(stringr)

## want to aggregate all stations for a particular FIPS !

# 1. get station list for a particular fips
# probably want to use ggmaps geocodes for this instead
isd_fips_stations <- function(fips){
  census_data <- read.csv('http://www2.census.gov/geo/docs/reference/cenpop2010/county/CenPop2010_Mean_(
  state <- census_data$STATEFP
  county <- census_data$COUNTYFP

  state[str_length(state) == 1] <- paste0(0, state[str_length(state) == 1])
  county[str_length(county) == 1] <- paste0(00, county[str_length(county) == 1])
  county[str_length(county) == 2] <- paste0(0, county[str_length(county) == 2])

  FIPS <- paste0(state,county)
  census_data$FIPS <- FIPS

  lat <- census_data$LATITUDE
  lon <- census_data$LONGITUDE

  your_fips <- fips
  row_num <- which(grepl(your_fips, census_data$FIPS))

  lat_FIPS <- lat[row_num]
  lon_FIPS <- lon[row_num]

  stations <- isd_stations_search(lat = lat_FIPS, lon = lon_FIPS,
                                  radius = 50)
  return(stations)
}


# 2. get tidy full dataset for all monitors
# let user specify weather variables? for example, var = c("wind_speed",
# "temperature")

# function to get dataset for a single monitor

int_surface_data <- function(usaf_code, wban_code, year, var = "all"){
  isd_df <- rnoaa::isd(usaf = usaf_code, wban = wban_code, year = year)$data
  # add date time (suggested by one of the rnoaa package vignette examples for isd())
  isd_df$date_time <- ymd_hm(sprintf("%s %s", as.character(isd_df$date), isd_df$time))
  # select variables
  cols <- c("usaf_station", "wban_station", "date_time", "latitude", "longitude")
  subset_vars <- append(var, cols)
  isd_df <- dplyr::select_(isd_df, .dots = subset_vars)
  # change misisng weather data values to NA - it looks like non-signed items are filled
  # with 9 (quality codes), 999 or 9999; signed items are positive filled (+9999 or +99999)
  # ftp://ftp.ncdc.noaa.gov/pub/data/noaa/ish-format-document.pdf
  #for(i in 1:length(var)){
  #  isd_df <- filter(isd_df, var[i] < 900)
  #}
```

```r
  # great the above code failed miserably...not sure how to do this.

  # also want to add a check for percent missing for each weather variable
  # and add option to filter
  return(isd_df)
}

# 3. pull data for multiple monitors

isd_monitors_data <- function(fips, year, var = "all"){
  ids <- isd_fips_stations(fips)
  safe_int <- purrr::safely(int_surface_data)

  mult_stations <- lapply(int_surface_data, usaf_code = ids$usaf, wban_code =
                          ids$wban, year = year, var = var)

  test <- mapply(safe_int, usaf_code = ids$usaf, wban_code =
                 ids$wban, year = year, var = var)
  check_station <- sapply( )
  # ^ reference meteo_pull_monitors
}

# 4. average across stations

ave_hourly <- function(all_stations){
  averaged <- gather(all_stations, key, value, -)
}
```

RS: We should be able to filter the res dataframe to the particular month we're interested in to get hourly wind data. We found the USAF and WBAN codes for Miami by searching this text file of stations - there should be a better way to find USAF and WBAN codes for a particular location or station.

CP: I found a function to go from location (lat/lon) to USAF and WBAN codes `isd_stations_search`, and use census data to go from FIPS to lat/lon. See above code.

RS: More info about this data can be found by going to NOAA's land-based station data site, then following the link for Integrated Surface Hourly Data base (3505) at the bottom of the page. For example, the readme.txt and ish-format-document.pdf files could be helpful.

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:

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

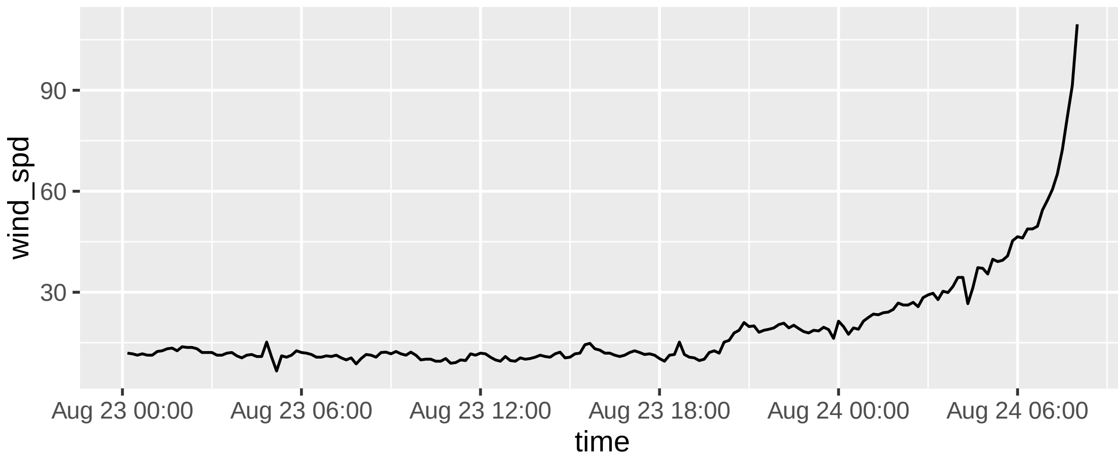
```
## Using c1992.nc
```

```r
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:

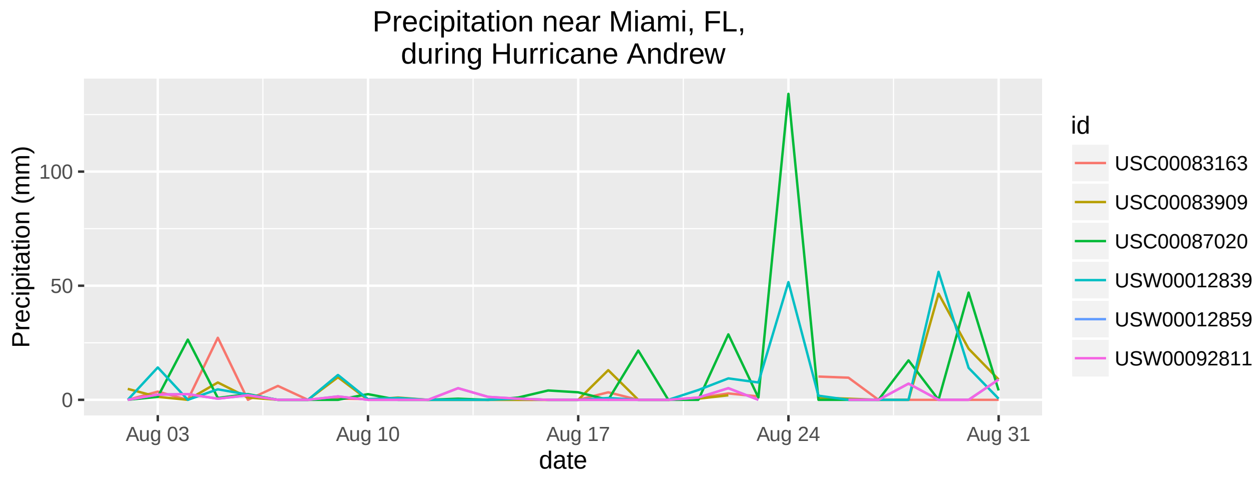
```r
# library(devtools)
# install_github("ropenscilabs/rnoaa") # if you need to install the package
# install_github("leighseverson/countyweather") # if you need to install the package
library(rnoaa)
library(countyweather)
miami_stations <- fips_stations(fips = "12086")
miami_rain <- meteo_pull_monitors(miami_stations,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

```r
# 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
```

```r
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)")
```




```r
##Cyclone Tracy
library(ggmap)
darwin_ll <- geocode("Darwin, Australia")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Darwin,%20Australia&s
```

```
darwin <- data.frame(id = "darwin",
                     latitude = darwin_ll$lat,
                     longitude = darwin_ll$lon)
darwin_monitors <- meteo_nearby_stations(lat_lon_df = darwin,
                                         lat_colname = "latitude",
                                         lon_colname = "longitude",
                                         station_data = station_data,
                                         radius = 50,
                                         var = "PRCP",
                                         year_min = 1974, year_max = 1974)
darwin_monitors
```

```
## $darwin
##             id                        name latitude longitude   distance
## 1  ASN00014167                 STOKES HILL -12.4648  130.8495  0.8667207
## 2  ASN00014170                  LARRAKEYAH -12.4582  130.8283  1.5511151
## 3  ASN00014018           DARWIN LOCO WORKS -12.4394  130.8419  2.6050088
## 4  ASN00014163      DARWIN BOTANIC GARDENS -12.4396  130.8376  2.6222584
## 5  ASN00014218                       PARAP -12.4331  130.8409  3.3068747
## 6  ASN00014040 DARWIN AIRPORT COMPARISON -12.4227  130.8844  6.4287085
## 7  ASN00014017 DARWIN QUARANTINE STATION -12.4867  130.9000  6.8559917
## 8  ASN00014015              DARWIN AIRPORT -12.4239  130.8925  7.0049875
## 9  ASN00014182              WINNELLIE PARK -12.4333  130.9000  7.1236270
## 10 ASN00014162               COCONUT GROVE -12.3990  130.8508  7.1645586
## 11 ASN00014169             COONAWARRA NAVY -12.4333  130.9050  7.6095447
## 12 ASN00014164                     JINGILI -12.3931  130.8714  8.3941018
## 13 ASN00014270      NIGHTCLIFF SPORTS CLUB -12.3833  130.8500  8.8879774
## 14 ASN00014112             NIGHTCLIFF POOL -12.3786  130.8469  9.3821330
## 15 ASN00014116      BERRIMAH RESEARCH FARM -12.4429  130.9325 10.0967194
## 16 ASN00014009              CHANNEL ISLAND -12.5547  130.8677 10.5962975
## 17 ASN00014175              LEE POINT RAAF -12.3600  130.8883 12.5003079
## 18 ASN00014172            ELEVEN MILE RAAF -12.4509  130.9575 12.6346621
## 19 ASN00014129                     BELYUEN -12.5380  130.6985 17.6578037
## 20 ASN00014149              HOWARD SPRINGS -12.4558  131.0505 22.6757999
## 21 ASN00014110     NA RAILWAY 22 MILE SITE -12.5617  131.0717 27.2731591
## 22 ASN00014215       BERRY SPRINGS RANGERS -12.6997  131.0000 31.4419001
## 23 ASN00014080                    NOONAMAH -12.6367  131.0733 31.7059127
## 24 ASN00014032                  KOOLPINYAH -12.3889  131.1767 37.2867538
## 25 ASN00014183            DARWIN RIVER DAM -12.8324  130.9713 43.4309727
```

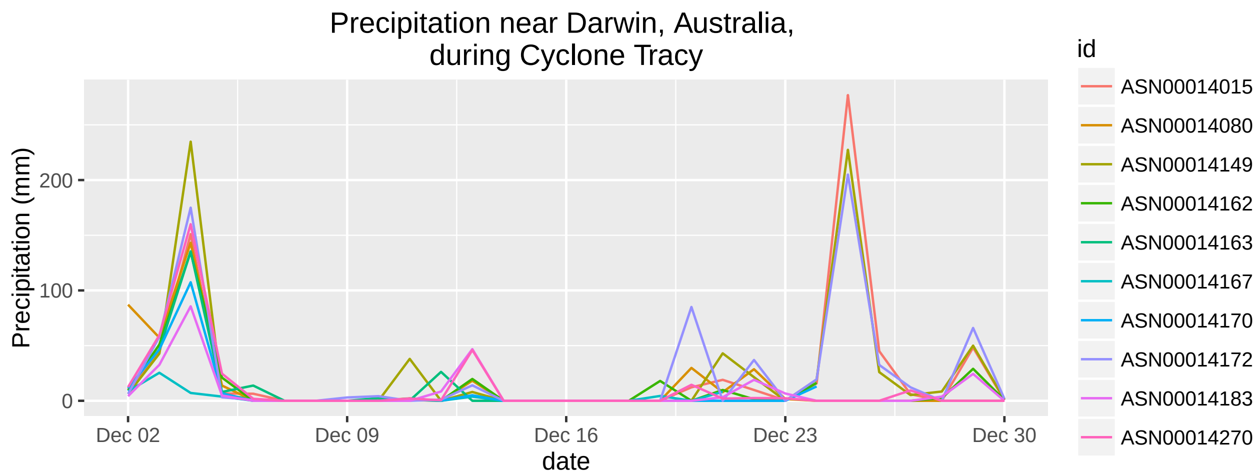
```
darwin_monitors <- darwin_monitors[[1]]$id
```

```
darwin_weather <- meteo_pull_monitors(darwin_monitors, date_min = "1974-12-01",
                                      date_max = "1974-12-31",
                                      var = "PRCP")
```

-JF: `darwin_monitors` includes 25 possible monitors for the year 1974. However, when using the function `meteo_pull_monitors`, data was pulled for only 10 of these possible 25 monitors. This may be due to the date restriction used to pull data for this storm, and 15 monitors simply do not have available data from this time frame. No error message was given.

```r
ggplot(darwin_weather, aes(x = date, y = prcp, color = id)) +
  geom_line() + ggtitle("Precipitation near Darwin, Australia, \nduring Cyclone Tracy") +
  ylab("Precipitation (mm)")
```

```
## Warning: Removed 23 rows containing missing values (geom_path).
```




```r
##Labor Day Storm
keys <- geocode("Florida Keys, St. Petersburg, FL")
```

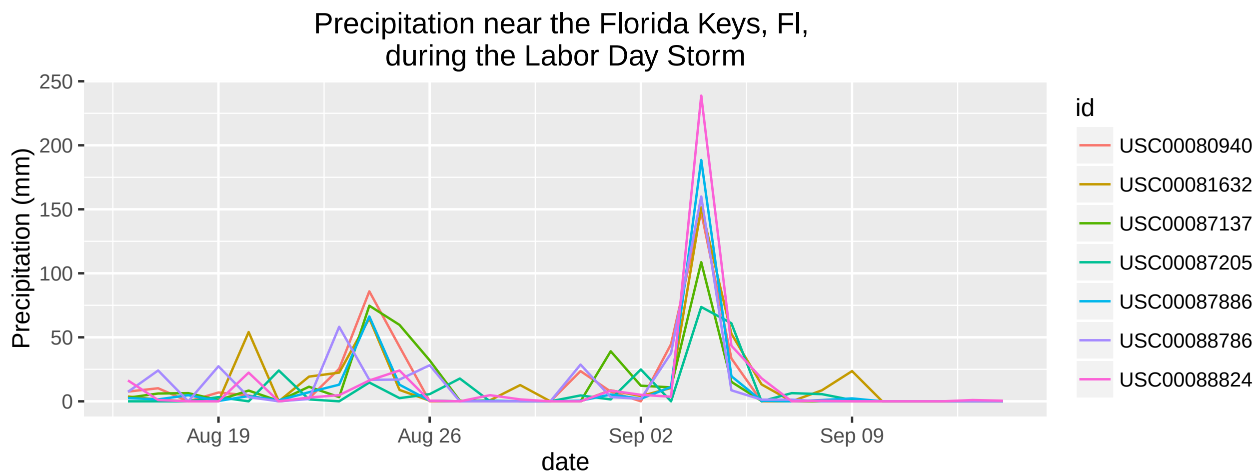
```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Florida%20Keys,%20St
```

```r
keys <- data.frame(id = "keys", latitude = keys$lat, longitude = keys$lon)
keys_monitors <- meteo_nearby_stations(lat_lon_df = keys,
                                       lat_colname = "latitude",
                                       lon_colname = "longitude",
                                       station_data = station_data,
                                       radius = 50,
                                       var = "PRCP",
                                       year_min = 1935, year_max = 1935)
keys_monitors
```

```
## $keys
##            id                     name latitude longitude distance
## 1 USC00088786                 FL TAMPA  27.9500  -82.4500 11.44555
## 2 USC00087886         FL ST PETERSBURG  27.7628  -82.6261 16.89103
## 3 USC00087137         FL PINELLAS PARK  27.8333  -82.7000 20.99748
## 4 USC00081632            FL CLEARWATER  27.9667  -82.7667 30.22285
## 5 USC00087205            FL PLANT CITY  28.0236  -82.1422 38.89319
## 6 USC00080940      FL BRADENTON EXP STN  27.4833  -82.5500 41.51840
## 7 USC00088824 FL TARPON SPGS SEWAGE PL  28.1522  -82.7539 42.35373
```

```r
keys_monitors <- keys_monitors[[1]]$id
```

```
keys_weather <- meteo_pull_monitors(keys_monitors, date_min = "1935-08-15",
                                    date_max = "1935-09-15",
                                    var = "PRCP")
ggplot(keys_weather, aes(x = date, y = prcp, color = id)) +
  geom_line() + ggtitle("Precipitation near the Florida Keys, Fl, \nduring the Labor Day Storm") +
  ylab("Precipitation (mm)")
```




```
fuzhou <- geocode("Fuzhou, China")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Fuzhou,%20China&sense
```

```
fuzhou <- data.frame(id = "fuzhou", latitude = fuzhou$lat, longitude = fuzhou$lon)
fuzhou_monitors <- meteo_nearby_stations(lat_lon_df = fuzhou,
                                         lat_colname = "latitude",
                                         lon_colname = "longitude",
                                         station_data = station_data,
                                         radius = 50,
                                         var = "PRCP",
                                         year_min = 2006, year_max = 2006)
fuzhou_monitors
```

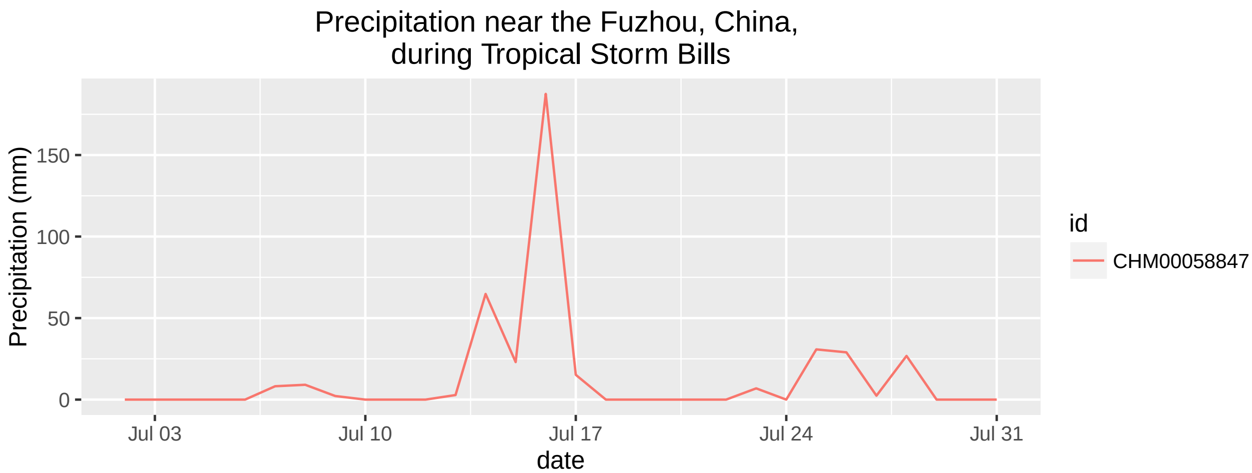
```
## $fuzhou
##            id   name latitude longitude distance
## 1 CHM00058847 FUZHOU   26.083   119.283 1.645583
```

```
fuzhou_monitors <- fuzhou_monitors[[1]]$id
```

- JF: Looks like there is only one station in/near Fuzhou, China, within a 50 mile radius.

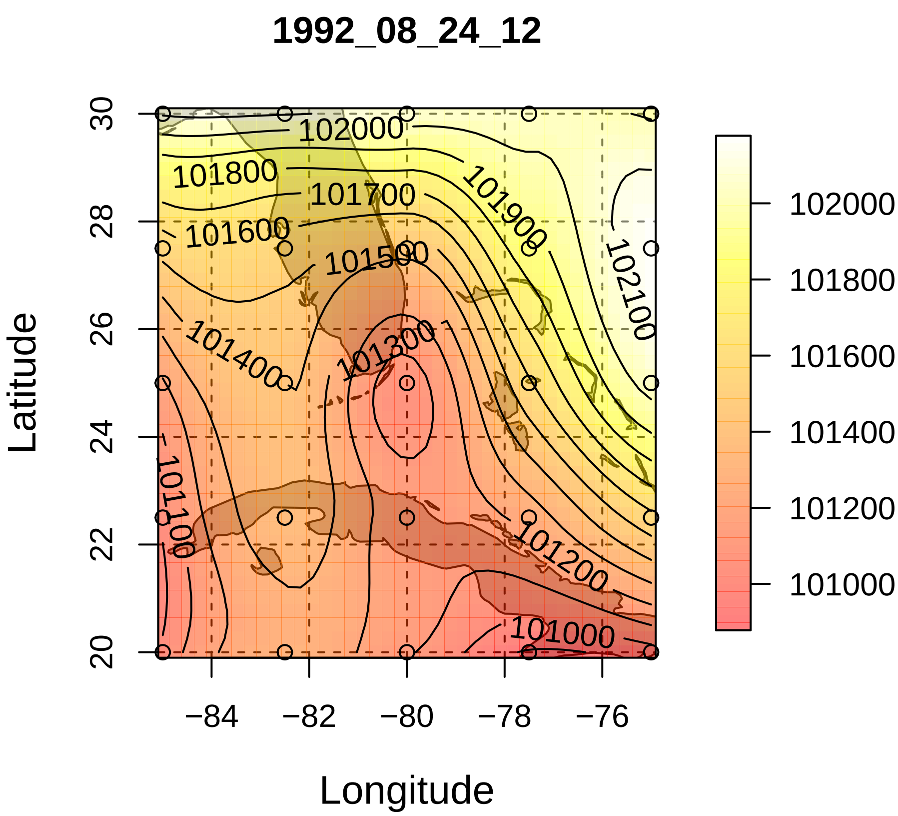
```
fuzhou_weather <- meteo_pull_monitors(fuzhou_monitors, date_min = "2006-07-01",
                                    date_max = "2006-08-01",
                                    var = "PRCP")
```

```r
ggplot(fuzhou_weather, aes(x = date, y = prcp, color = id)) +
  geom_line() + ggtitle("Precipitation near the Fuzhou, China, \nduring Tropical Storm Bills") +
  ylab("Precipitation (mm)")
```



HK: Here is the pressure data during Andrew using the package RNCEP We aren't sure how accurate the model is compared to the true data. Regarding the Lon and Lat values, I picked a random larger box around the Miami station

```
## [1] Units of variable 'pres.sfc' are Pascals
```



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)
```

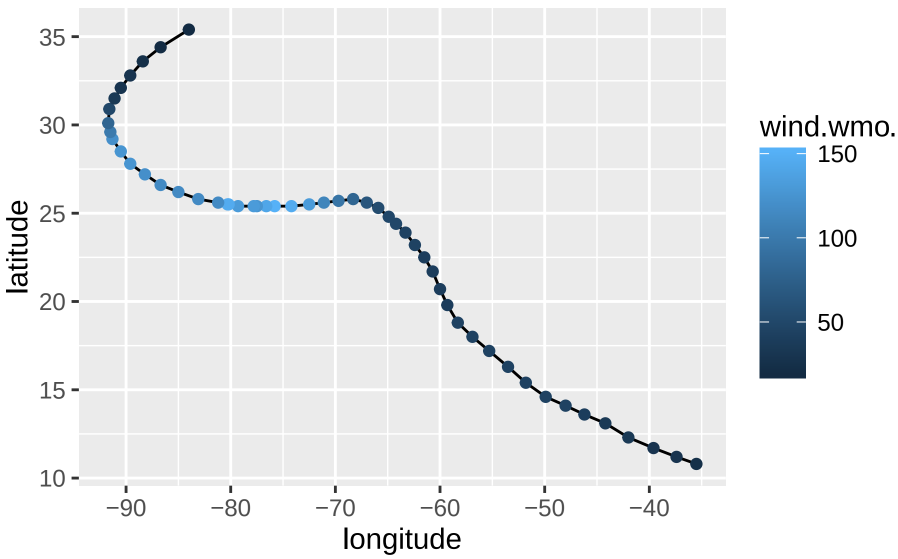
```
##               iso_time latitude longitude wind.wmo.
## 1 1992-08-16 18:00:00     10.8     -35.5        25
## 2 1992-08-17 00:00:00     11.2     -37.4        30
## 3 1992-08-17 06:00:00     11.7     -39.6        30
```

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)
```

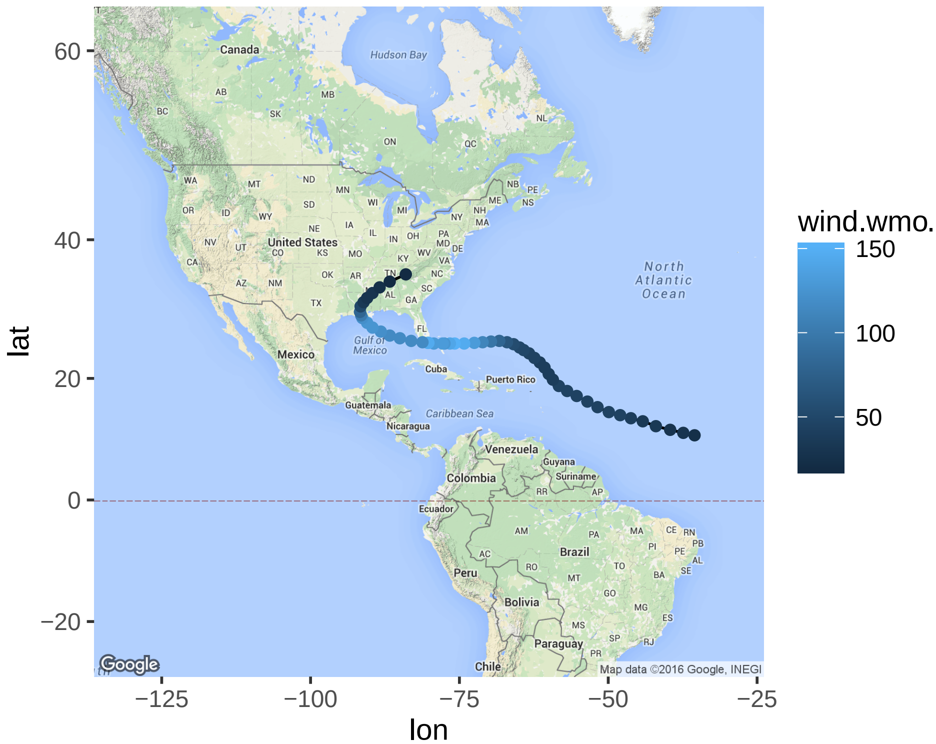
```
##                    iso_time latitude longitude wind.wmo.
## 2417 1992-08-16 18:00:00     10.8     -35.5        25
## 2418 1992-08-17 00:00:00     11.2     -37.4        30
## 2419 1992-08-17 06:00:00     11.7     -39.6        30
```

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



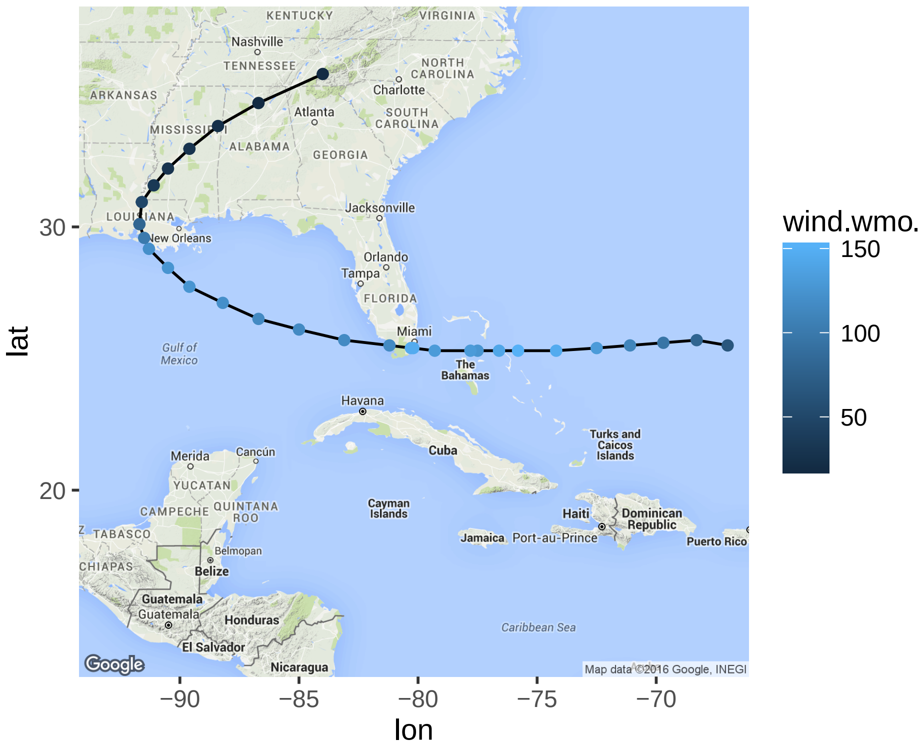
Let me add to a map:

```
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.

Cyclone Tracy example: (note: need to pull from 1975, even though it was in 1974)

```
Tracy_1975<-storm_data(year=1975)
names(Tracy_1975)
```

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

```
colnames(Tracy_1975$data)[1:10]
```

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

```
tracy <- Tracy_1975$data[which(Tracy_1975$data$name == "07S:TRACY"), ]
as.vector(unique(Tracy_1975$data$name))
```

```
##   [1] "MARCIA"                           "NORAH"
##   [3] "PENNY"                            "01P"
##   [5] "02P:HSK0475"                      "03S:SELMA"
##   [7] "04P:HSK0675"                      "05P:HSK0775"
##   [9] "06P:HSK0875"                      "07S:TRACY"
##  [11] "ADELE"                            "08S"
##  [13] "09S:BLANDINE"                     "01B"
##  [15] "10S:CAMILLE"                      "12P"
##  [17] "11P:12P:FLORA"                    "14S:DEBORAH:ROBYN:ROBYN/DEBORA"
##  [19] "13P"                              "13P:GLORIA"
##  [21] "LOLA"                             "15P:HSK1675"
```

```
##  [23] "NOT NAMED"                       "16P:VAL"
##  [25] "17S"                             "ELSA"
##  [27] "18P:HSK1975"                     "19S"
##  [29] "FERNANDE"                        "20S:GERVAISE"
##  [31] "SHIRLEY"                         "21S:TRIXIE"
##  [33] "22S:HELOSE"                      "23P:ALISON"
##  [35] "24S:26S:INES"                    "25S:WILMA"
##  [37] "26S"                             "27S:VIDA"
##  [39] "28S:BEVERLEY"                    "29P:BETTY:HSK3075"
##  [41] "30P:AMELIA"                      "31S:JUNON"
##  [43] "CLARA"                           "32S:CLARA"
##  [45] "02W:TD 02"                       "02A"
##  [47] "03B"                             "DENISE"
##  [49] "04A"                             "AGATHA"
##  [51] "05B"                             "TD0616"
##  [53] "06B"                             "07A"
##  [55] "UNNAMED"                         "08B"
##  [57] "AMY"                             "BRIDGET"
##  [59] "CARLOTTA"                        "ELEANOR"
##  [61] "09B"                             "MAMIE"
##  [63] "BLANCHE"                         "FRANCENE"
##  [65] "NINA"                            "10B"
##  [67] "05W:TD 05"                       "(NAMELESS)(-)1:RITA:TD0809"
##  [69] "ORA"                             "PHYLLIS"
##  [71] "GEORGETTE"                       "HILARY"
##  [73] "11B"                             "RITA"
##  [75] "ILSA"                            "JEWEL"
##  [77] "CAROLINE"                        "SUSAN"
##  [79] "KATRINA"                         "DORIS"
##  [81] "TESS"                            "VIOLA"
##  [83] "WINNIE"                          "12B"
##  [85] "ELOISE"                          "ALICE"
##  [87] "BETTY"                           "LILY"
##  [89] "FAYE"                            "GLADYS"
##  [91] "13B"                             "MONICA"
##  [93] "NANETTE"                         "CORA"
##  [95] "14A"                             "ELSIE"
##  [97] "18W:FLOSSIE:TD 18"               "15B"
##  [99] "16A"                             "OLIVIA"
## [101] "GRACE"                           "HALLIE"
## [103] "17B"                             "HELEN:HELLEN"
## [105] "18B"                             "PRISCILLA"
## [107] "IDA"                             "19B"
## [109] "JUNE"                            "20B"
## [111] "SUBTROP:UNNAMED"                 "25W:TD 25:TD1226"
## [113] "24W:TD 24"
```

```
tracy[1:5, 1:10]
```

```
##        serial_num season num basin sub_basin      name            iso_time
## 346 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 11:30:00
## 347 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 12:00:00
## 348 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 18:00:00
## 349 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 23:30:00
```

```
## 350 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-22 00:00:00
##     nature latitude longitude
## 346     NR     -9.2     132.1
## 347     TS   -999.0    -999.0
## 348     NR     -9.6     131.5
## 349     NR     -9.9     131.2
## 350     TS   -999.0    -999.0
```

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

```
##               id                 name
## 1 1842298N11080   NOT NAMED(td9636)
## 2 1845336N10074   NOT NAMED(td9636)
## 3 1848011S09079   NOT NAMED(td9636)
## 4 1848011S09080 XXXX848003(reunion)
## 5 1848011S15057 XXXX848002(reunion)
## 6 1848011S16057   NOT NAMED(td9636)
```

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

```
## [1] "bsh071975(jtwc_sh) - TRACY(bom) - NOT NAMED(td9636) - TRACY(neumann) - TRACY(ds824_au) - 07S(ds8
```

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

```
##                  id
## 7945 1974356S09132
##
## 7945 bsh071975(jtwc_sh) - TRACY(bom) - NOT NAMED(td9636) - TRACY(neumann) - TRACY(ds824_au) - 07S(ds8
```

```
tracy_id <- storm_names[grep("TRACY", storm_names$name), 1][1]
tracy_2 <- storm_data(storm = tracy_id)
head(tracy_2$data[ , c("iso_time", "latitude", "longitude", "wind.wmo.")], 3)
```

```
##               iso_time latitude longitude wind.wmo.
## 1 1974-12-21 11:30:00     -9.2     132.1        -1
## 2 1974-12-21 12:00:00   -999.0    -999.0      -999
## 3 1974-12-21 18:00:00     -9.6     131.5        -1
```

```
tracy_loc <- select(tracy, iso_time, latitude, longitude, wind.wmo.)
tracy_loc$latitude[tracy_loc$latitude < -300]<-NA
tracy_loc$longitude[tracy_loc$longitude < -300]<-NA
tracy_loc$wind.wmo.[tracy_loc$wind.wmo. < -300]<-NA
head(tracy, 3)
```

```
##        serial_num season num basin sub_basin      name            iso_time
## 346 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 11:30:00
## 347 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 12:00:00
## 348 1974356S09132   1975   5    SI        WA 07S:TRACY 1974-12-21 18:00:00
##     nature latitude longitude wind.wmo. pres.wmo. center
```

```
## 346     NR     -9.2     132.1        -1       990    bom
## 347     TS   -999.0    -999.0      -999      -999    N/A
## 348     NR     -9.6     131.5        -1       990    bom
##     wind.wmo..percentile pres.wmo..percentile track_type
## 346                 -100               50.387       main
## 347                 -999             -999.000       main
## 348                 -100               50.387       main
##     latitude_for_mapping longitude_for_mapping current.basin
## 346                -9.20                132.10            SI
## 347                -9.33                132.03            SI
## 348                -9.65                131.43            SI
##     hurdat_atl_lat hurdat_atl_lon hurdat_atl_grade hurdat_atl_wind
## 346           -999           -999             -999            -999
## 347           -999           -999             -999            -999
## 348           -999           -999             -999            -999
##     hurdat_atl_pres td9636_lat td9636_lon td9636_grade td9636_wind
## 346            -999     -999.0     -999.0         -999        -999
## 347            -999       -9.4      132.0         -999          10
## 348            -999       -9.7      131.4         -999          10
##     td9636_pres reunion_lat reunion_lon reunion_grade reunion_wind
## 346        -999        -999        -999          -999         -999
## 347        -999        -999        -999          -999         -999
## 348        -999        -999        -999          -999         -999
##     reunion_pres atcf_lat atcf_lon atcf_grade atcf_wind atcf_pres
## 346         -999     -999     -999       -999      -999      -999
## 347         -999     -999     -999       -999      -999      -999
## 348         -999     -999     -999       -999      -999      -999
##     ds824_sh_lat ds824_sh_lon ds824_sh_grade ds824_sh_wind ds824_sh_pres
## 346       -999.0       -999.0           -999          -999          -999
## 347         -9.4        132.0           -999            10          -999
## 348         -9.7        131.3           -999            25          -999
##     ds824_ni_lat ds824_ni_lon ds824_ni_grade ds824_ni_wind ds824_ni_pres
## 346         -999         -999           -999          -999          -999
## 347         -999         -999           -999          -999          -999
## 348         -999         -999           -999          -999          -999
##     bom_lat bom_lon bom_grade bom_wind bom_pres ds824_au_lat ds824_au_lon
## 346  -999.0  -999.0      -999     -999     -999       -999.0       -999.0
## 347    -9.2   132.0      -999     -999      990         -9.2        132.1
## 348    -9.6   131.5      -999     -999      990         -9.6        131.5
##     ds824_au_grade ds824_au_wind ds824_au_pres jtwc_sh_lat jtwc_sh_lon
## 346           -999          -999          -999      -999.0      -999.0
## 347           -999            45           990        -9.4       132.0
## 348           -999            45           990        -9.7       131.3
##     jtwc_sh_grade jtwc_sh_wind jtwc_sh_pres jtwc_wp_lat jtwc_wp_lon
## 346          -999         -999         -999        -999        -999
## 347          -999           10         -999        -999        -999
## 348          -999           25         -999        -999        -999
##     jtwc_wp_grade jtwc_wp_wind jtwc_wp_pres td9635_lat td9635_lon
## 346          -999         -999         -999       -999       -999
## 347          -999         -999         -999       -999       -999
## 348          -999         -999         -999       -999       -999
##     td9635_grade td9635_wind td9635_pres ds824_wp_lat ds824_wp_lon
## 346         -999        -999        -999         -999         -999
## 347         -999        -999        -999         -999         -999
```

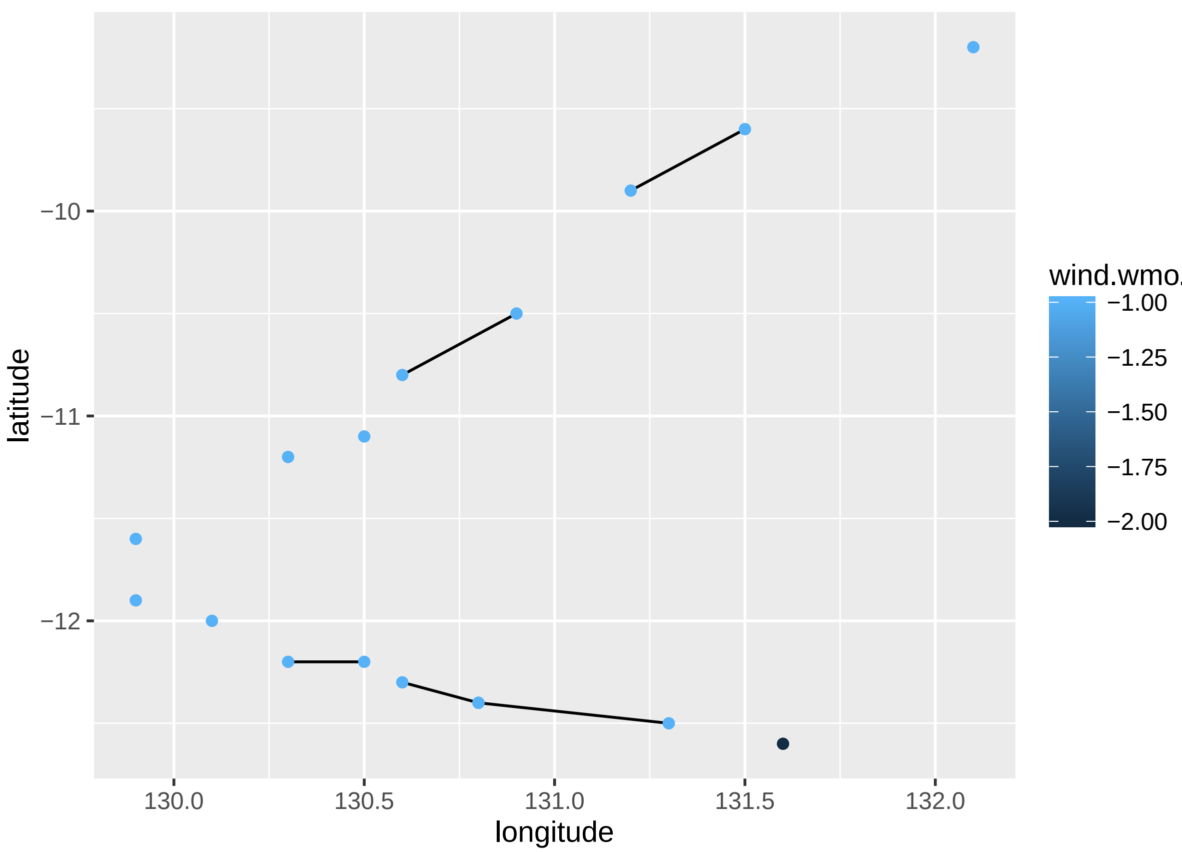
```
## 348          -999         -999         -999          -999          -999
##     ds824_wp_grade ds824_wp_wind ds824_wp_pres jtwc_io_lat jtwc_io_lon
## 346           -999          -999          -999        -999        -999
## 347           -999          -999          -999        -999        -999
## 348           -999          -999          -999        -999        -999
##     jtwc_io_grade jtwc_io_wind jtwc_io_pres cma_lat cma_lon cma_grade
## 346          -999         -999         -999    -999    -999      -999
## 347          -999         -999         -999    -999    -999      -999
## 348          -999         -999         -999    -999    -999      -999
##     cma_wind cma_pres hurdat_epa_lat hurdat_epa_lon hurdat_epa_grade
## 346     -999     -999           -999           -999             -999
## 347     -999     -999           -999           -999             -999
## 348     -999     -999           -999           -999             -999
##     hurdat_epa_wind hurdat_epa_pres jtwc_ep_lat jtwc_ep_lon jtwc_ep_grade
## 346            -999            -999        -999        -999          -999
## 347            -999            -999        -999        -999          -999
## 348            -999            -999        -999        -999          -999
##     jtwc_ep_wind jtwc_ep_pres ds824_ep_lat ds824_ep_lon ds824_ep_grade
## 346         -999         -999         -999         -999           -999
## 347         -999         -999         -999         -999           -999
## 348         -999         -999         -999         -999           -999
##     ds824_ep_wind ds824_ep_pres jtwc_cp_lat jtwc_cp_lon jtwc_cp_grade
## 346          -999          -999        -999        -999          -999
## 347          -999          -999        -999        -999          -999
## 348          -999          -999        -999        -999          -999
##     jtwc_cp_wind jtwc_cp_pres tokyo_lat tokyo_lon tokyo_grade tokyo_wind
## 346         -999         -999      -999      -999        -999       -999
## 347         -999         -999      -999      -999        -999       -999
## 348         -999         -999      -999      -999        -999       -999
##     tokyo_pres neumann_lat neumann_lon neumann_grade neumann_wind
## 346       -999      -999.0      -999.0          -999         -999
## 347       -999      -999.0      -999.0          -999         -999
## 348       -999        -9.5       131.8          -999           45
##     neumann_pres hko_lat hko_lon hko_grade hko_wind hko_pres cphc_lat
## 346         -999    -999    -999      -999     -999     -999     -999
## 347         -999    -999    -999      -999     -999     -999     -999
## 348          990    -999    -999      -999     -999     -999     -999
##     cphc_lon cphc_grade cphc_wind cphc_pres wellington_lat wellington_lon
## 346     -999       -999      -999      -999           -999           -999
## 347     -999       -999      -999      -999           -999           -999
## 348     -999       -999      -999      -999           -999           -999
##     wellington_grade wellington_wind wellington_pres newdelhi_lat
## 346             -999            -999            -999         -999
## 347             -999            -999            -999         -999
## 348             -999            -999            -999         -999
##     newdelhi_lon newdelhi_grade newdelhi_wind newdelhi_pres nadi_lat
## 346         -999           -999          -999          -999     -999
## 347         -999           -999          -999          -999     -999
## 348         -999           -999          -999          -999     -999
##     nadi_lon nadi_grade nadi_wind nadi_pres reunion_rmw
## 346     -999       -999      -999      -999        -999
## 347     -999       -999      -999      -999        -999
## 348     -999       -999      -999      -999        -999
##     reunion_wind_radii_1_ne reunion_wind_radii_1_se
```

```
## 346                  -999                     -999
## 347                  -999                     -999
## 348                  -999                     -999
##     reunion_wind_radii_1_sw reunion_wind_radii_1_nw
## 346                  -999                     -999
## 347                  -999                     -999
## 348                  -999                     -999
##     reunion_wind_radii_2_ne reunion_wind_radii_2_se
## 346                  -999                     -999
## 347                  -999                     -999
## 348                  -999                     -999
##     reunion_wind_radii_2_sw reunion_wind_radii_2_nw bom_mn_hurr_xtnt
## 346                  -999                     -999             -999
## 347                  -999                     -999             -999
## 348                  -999                     -999             -999
##     bom_mn_gale_xtnt bom_mn_eye_diam bom_roci atcf_rmw atcf_poci atcf_roci
## 346             -999            -999     -999     -999      -999      -999
## 347             -999            -999       -1     -999      -999      -999
## 348             -999            -999       -1     -999      -999      -999
##     atcf_eye atcf_wrad34_rad1 atcf_wrad34_rad2 atcf_wrad34_rad3
## 346     -999             -999             -999             -999
## 347     -999             -999             -999             -999
## 348     -999             -999             -999             -999
##     atcf_wrad34_rad4 atcf_wrad50_rad1 atcf_wrad50_rad2 atcf_wrad50_rad3
## 346             -999             -999             -999             -999
## 347             -999             -999             -999             -999
## 348             -999             -999             -999             -999
##     atcf_wrad50_rad4 atcf_wrad64_rad1 atcf_wrad64_rad2 atcf_wrad64_rad3
## 346             -999             -999             -999             -999
## 347             -999             -999             -999             -999
## 348             -999             -999             -999             -999
##     atcf_wrad64_rad4 tokyo_dir50 tokyo_long50 tokyo_short50 tokyo_dir30
## 346             -999        -999         -999          -999        -999
## 347             -999        -999         -999          -999        -999
## 348             -999        -999         -999          -999        -999
##     tokyo_long30 tokyo_short30 jtwc_..._rmw jtwc_..._poci jtwc_..._roci
## 346         -999          -999         -999          -999          -999
## 347         -999          -999            0             0             0
## 348         -999          -999            0             0             0
##     jtwc_..._eye jtwc_..._wrad34_rad1 jtwc_..._wrad34_rad2
## 346         -999                 -999                 -999
## 347            0                   -1                   -1
## 348            0                   -1                   -1
##     jtwc_..._wrad34_rad3 jtwc_..._wrad34_rad4 jtwc_..._wrad50_rad1
## 346                 -999                 -999                 -999
## 347                   -1                   -1                   -1
## 348                   -1                   -1                   -1
##     jtwc_..._wrad50_rad2 jtwc_..._wrad50_rad3 jtwc_..._wrad50_rad4
## 346                 -999                 -999                 -999
## 347                   -1                   -1                   -1
## 348                   -1                   -1                   -1
##     jtwc_..._wrad64_rad1 jtwc_..._wrad64_rad2 jtwc_..._wrad64_rad3
## 346                 -999                 -999                 -999
## 347                   -1                   -1                   -1
```

```
## 348                  -1                    -1                    -1
##     jtwc_..._wrad64_rad4
## 346                -999
## 347                  -1
## 348                  -1
```

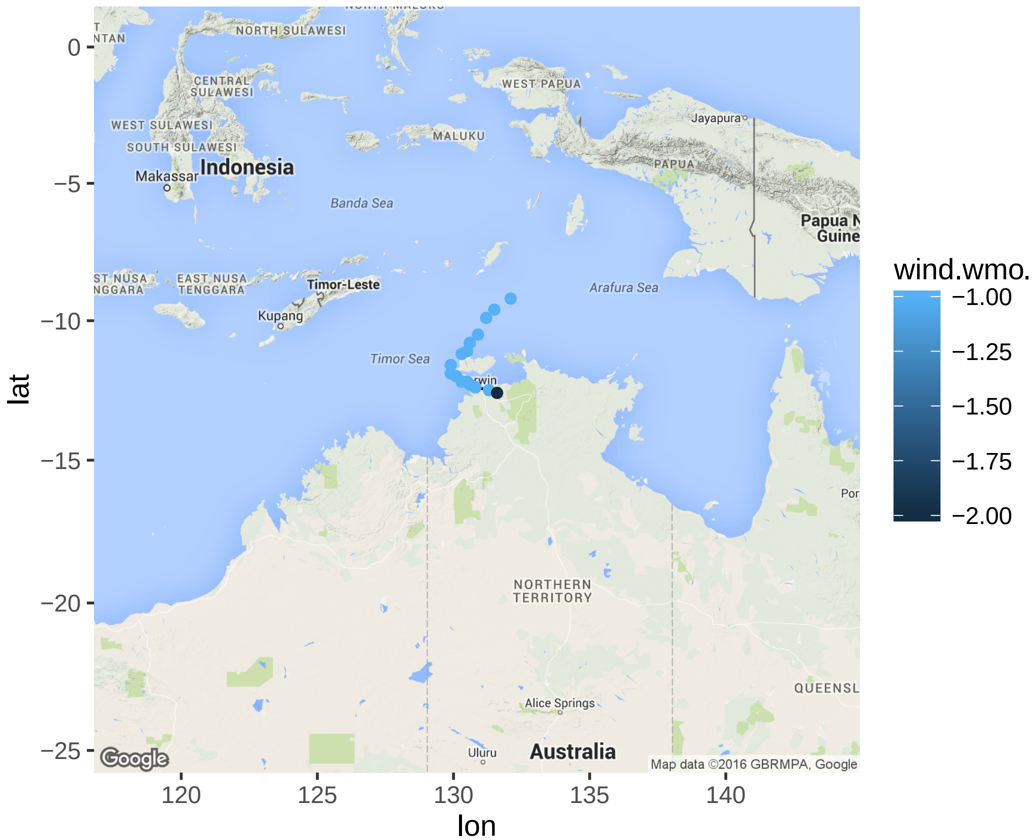
```r
ggplot(tracy_loc, aes(x = longitude, y = latitude)) +
  geom_path() + geom_point(aes(color = wind.wmo.))
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```




```r
library(ggmap)
darwin_map <- get_map(location = 'Darwin', zoom = 5)
ggmap(darwin_map) +
  geom_path(data = tracy_loc, aes(x = longitude, y = latitude)) +
  geom_point(data = tracy_loc, aes(x = longitude, y = latitude,
                                   color = wind.wmo.))
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```



Storm Path Function

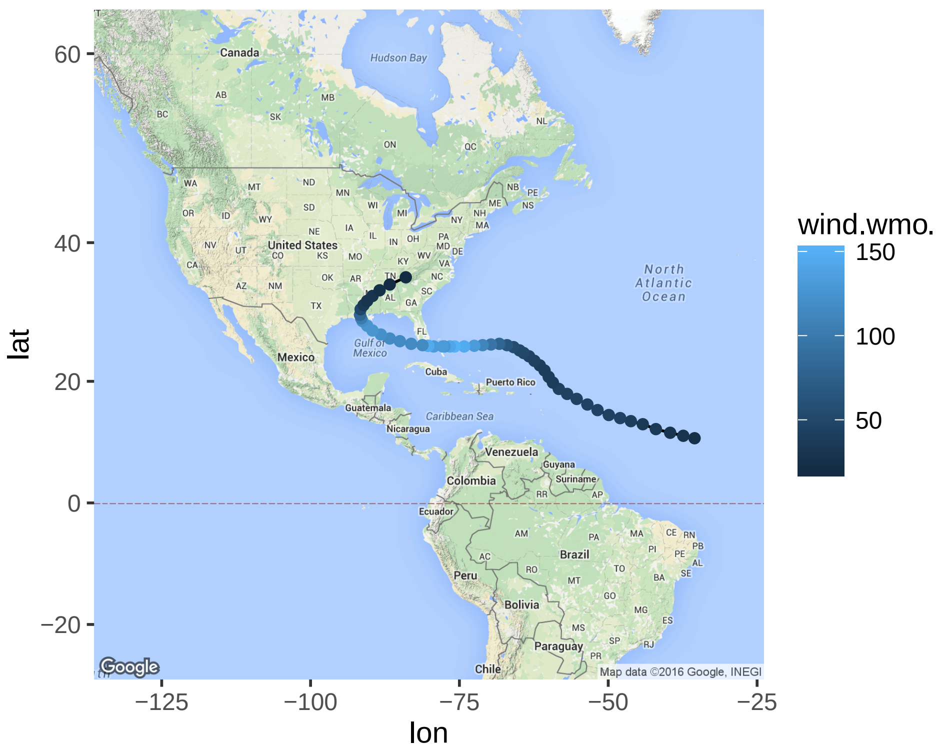
EW: This is a function for plotting the storm path. You need to know the `name`, `year`, and `location` of the storm you would like to analyze. The function provides two types of map: whole map or local map, which can be set via `local`.

```
require(rnoaa)
require(dplyr)
require(ggplot2)
require(ggmap)

storm_path <- function(name, year, location, local = TRUE){
  # name: name of the storm
  # year: year of the storm
  # location: location of the storm
  # local: if true, plot the local map; else, plot the whole plot

  hurrs_year <- storm_data(year = year)
  storm <- hurrs_year$data[which(hurrs_year$data$name == name), ]
  if(sum(storm$latitude == -999)){
    storm <- storm[-c(which(storm$latitude == -999)),]
  }
  storm_loc <- select(storm, iso_time, latitude, longitude, wind.wmo.)

  if(local){
    map <- get_map(location = location, zoom = 5)
  }else{
```

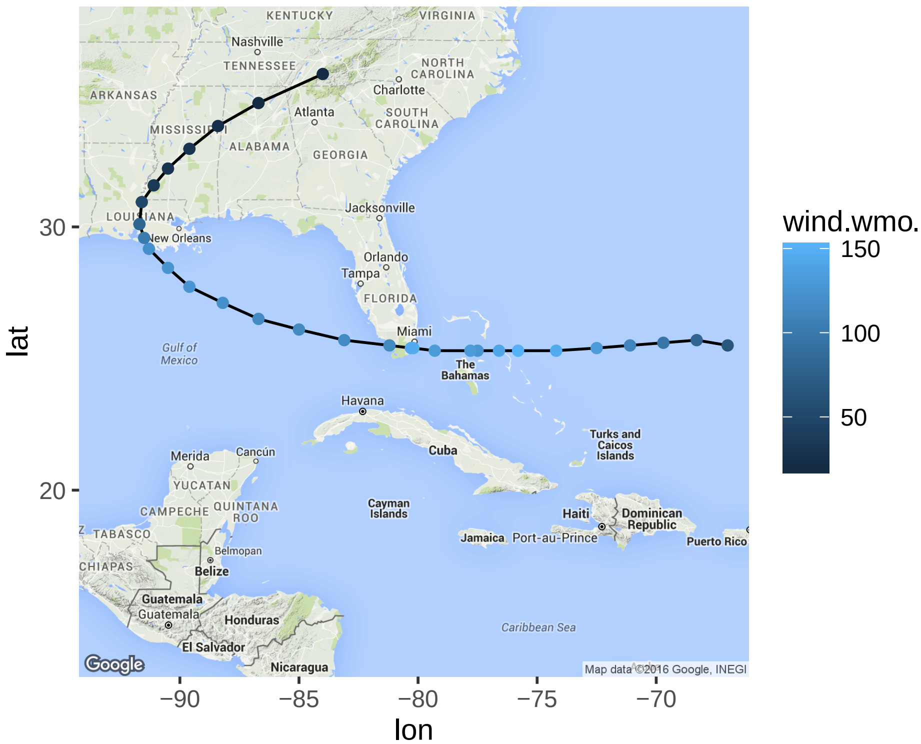
```
    map <- get_map(location = location, zoom = 3, source='google')
  }
  ggmap(map) +
    geom_path(data = storm_loc, aes(x = longitude, y = latitude)) +
    geom_point(data = storm_loc, aes(x = longitude, y = latitude,
                                     color = wind.wmo.))
}
```

Hurricane Andrew:

```
storm_path(name = "ANDREW", year = 1992, location = "Miami", local = F)
```



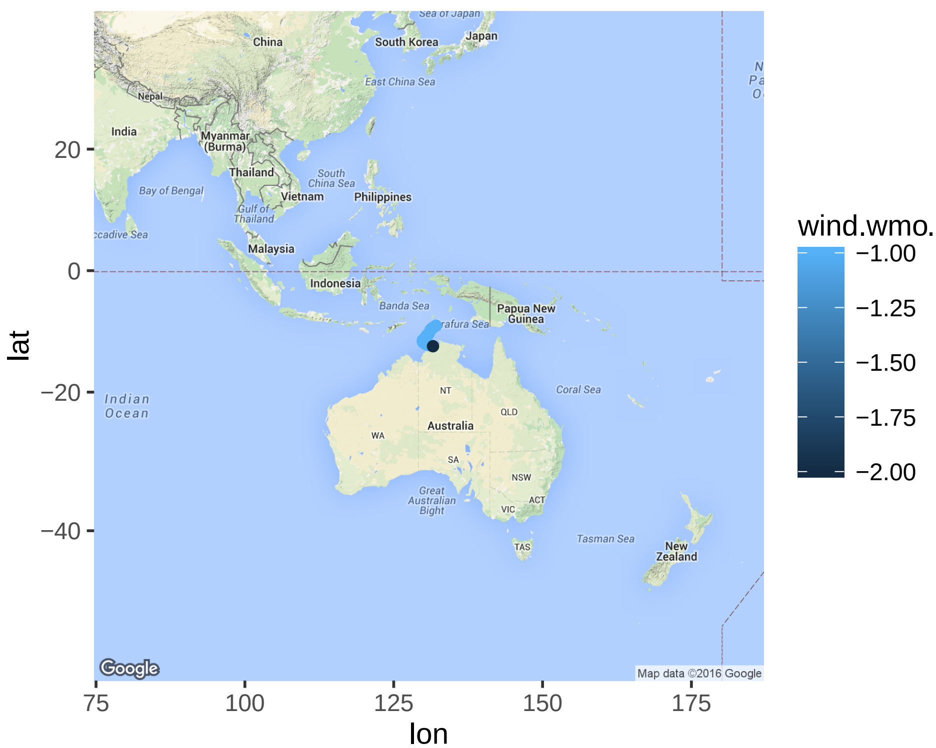

```
storm_path(name = "ANDREW", year = 1992, location = "Miami", local = T)
```



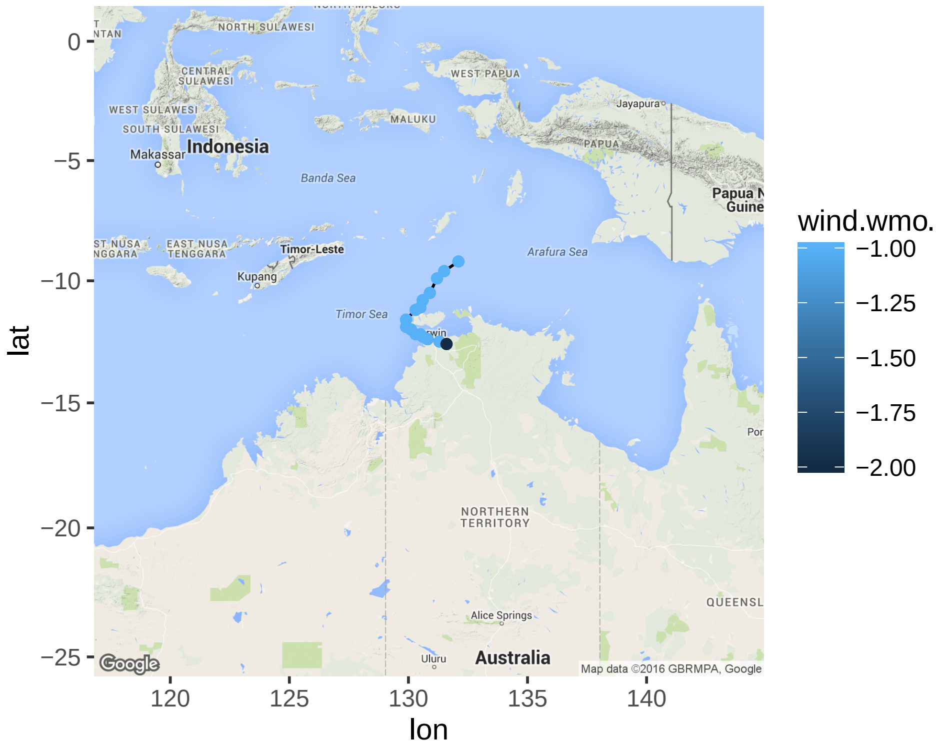
Labor Day Storm: cannot find the data.

Cyclone Tracy:

```
storm_path(name = "07S:TRACY", year = 1975, location = "Darwin", local = F)
```

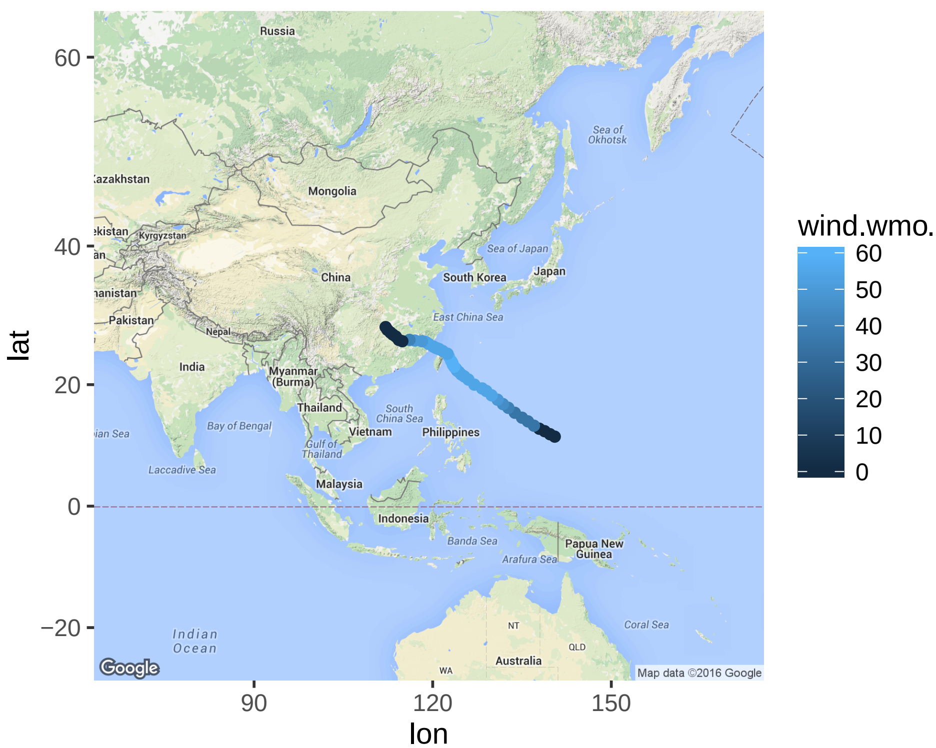


```r
storm_path(name = "07S:TRACY", year = 1975, location = "Darwin", local = T)
```

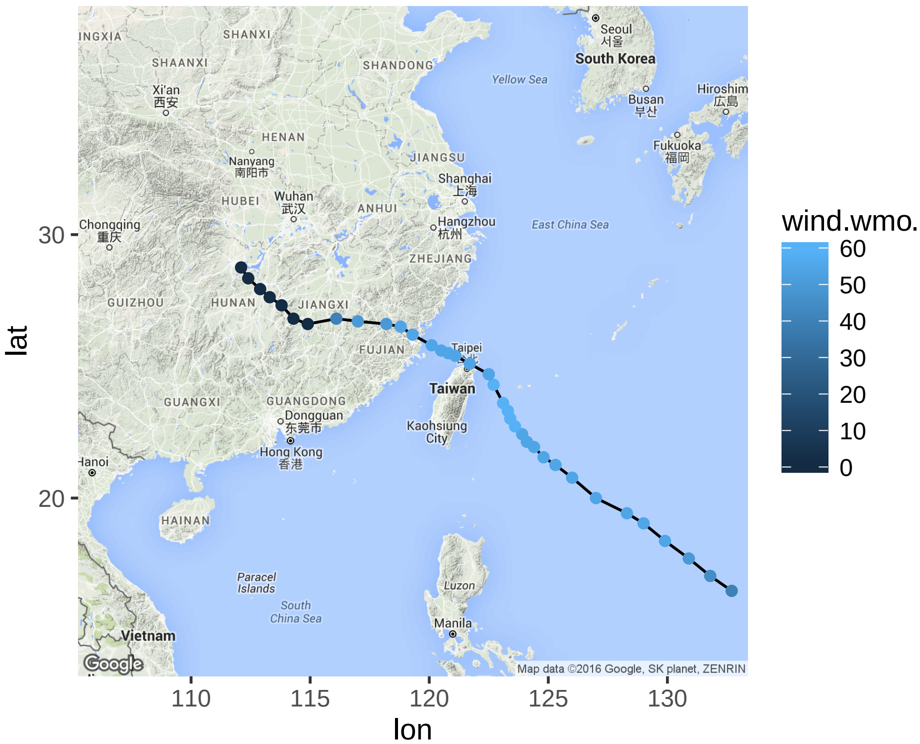


Tropical Storm Bilis:

```r
storm_path(name = "BILIS", year = 2006, location = "Fuzhou", local = F)
```



```r
storm_path(name = "BILIS", year = 2006, location = "Fuzhou", local = T)
```



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:

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

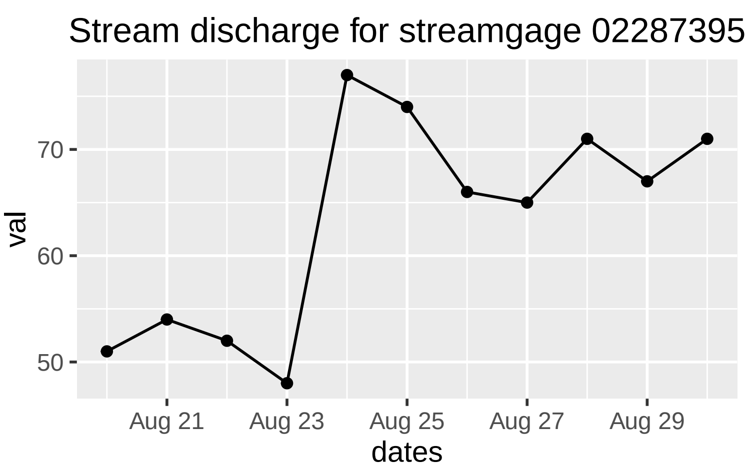
```
##  [1] "02287395"        "02287497"        "02288600"
##  [4] "02289019"        "02289040"        "02289041"
##  [7] "02289050"        "02289060"        "02289500"
## [10] "02290710"        "022907647"       "02290765"
## [13] "02290769"        "252523080352500" "254543080405401"
## [16] "254543080491101"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

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:

```r
# 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
```

```r
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.