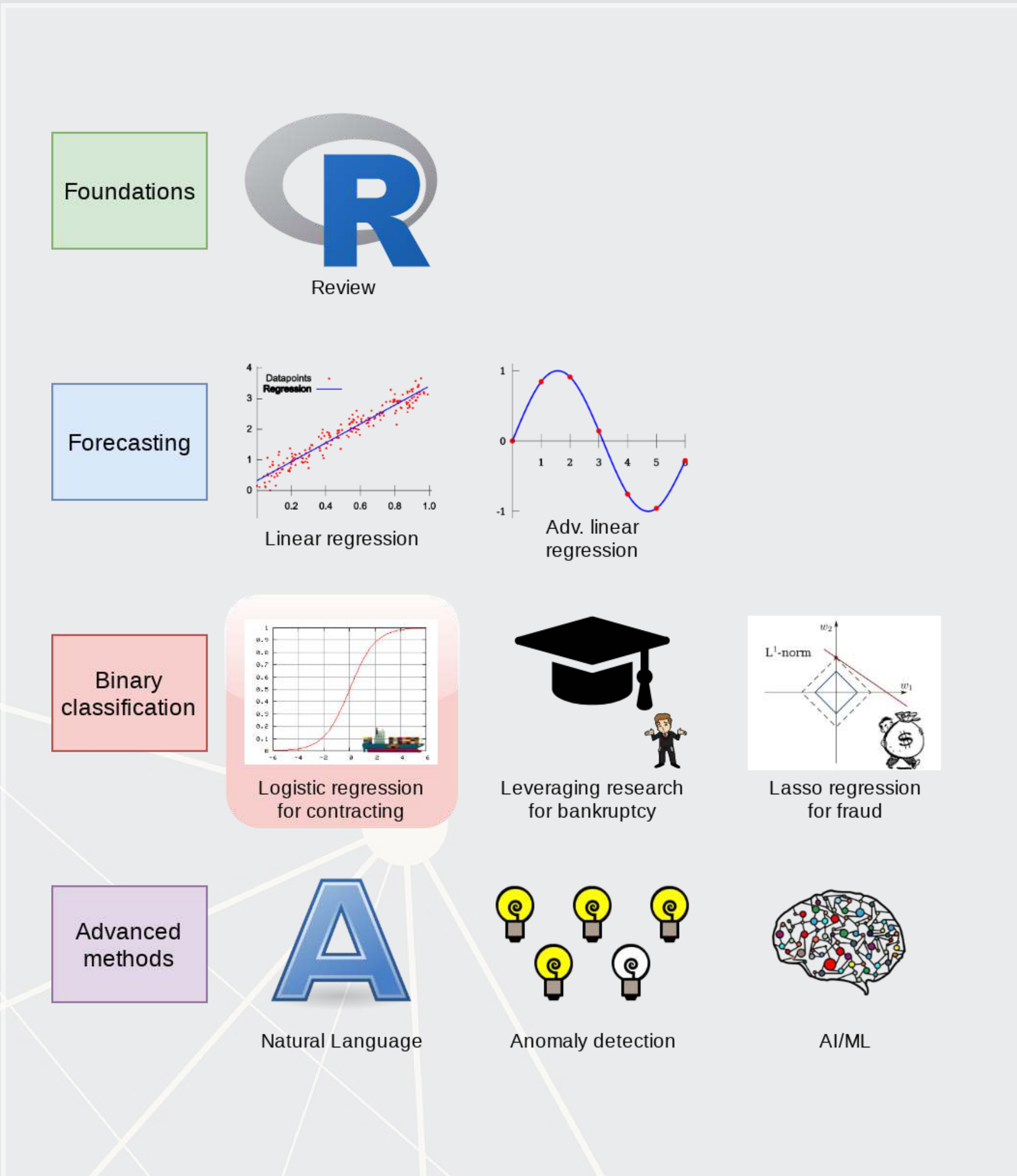
# ACCT 420: Logistic Regression

## Session 4

Dr. Richard M. Crowley

# Front matter

# Learning objectives




- **Theory:**
  - Understanding binary problems
- **Application:**
  - Detecting shipping delays caused by typhoons
- **Methodology:**
  - Logistic regression
  - Spatial visualization

# Datacamp

- Explore on your own
- No specific required class this week

# Assignment 2

- Looking at Singaporean retail firms
  - Mostly focused on time and cyclicality
  - Some visualization
  - A little of what we cover today
- Optional:
  - You can work in *pairs* on this assignment
    - If you choose to do this, please only make 1 submission and include both your names on the submission

# Binary outcomes

# What are binary outcomes?

- Thus far we have talked about events with continuous outcomes
  - Revenue: Some positive number
  - Earnings: Some number
  - ROA: Some percentage
- Binary outcomes only have two possible outcomes
  - Did something happen, *yes* or *no*?
  - Is a statement *true* or *false*?

# Accounting examples of binary outcomes

- Financial accounting:
  - Will the company's earnings meet analysts' expectations?
  - Will the company have positive earnings?
- Managerial accounting:
  - Will we have ___ problem with our supply chain?
  - Will our customer go bankrupt?
- Audit:
  - Is the company committing fraud?
- Taxation:
  - Is the company too aggressive in their tax positions?

> We can assign a probability to any of these

# Regression approach: Logistic regression

- When modeling a binary outcome, we use logistic regression
  - A.k.a. logit model
- The *logit* function is $logit(x) = \log\left(\frac{x}{1-x}\right)$
  - Also called *log odds*

$$\log\left(\frac{\text{Prob}(y=1|X)}{1-\text{Prob}(y=1|X)}\right) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \varepsilon$$

> There are other ways to model this though, such as probit

# Implementation: Logistic regression

- The logistic model is related to our previous linear models as such:

> Both linear and logit models are under the class of General Linear Models (GLMs)

- To regress a GLM, we use the `glm()` command.
  - In fact, the `lm()` command we have been using is actually `glm()` when you specify the option `family=gaussian`
- To run a logit regression:

```
mod <- glm(y ~ x1 + x2 + x3 + ..., data=df, family=binomial)

summary(mod)
```

> `family=binomial` is what sets the model to be a logit

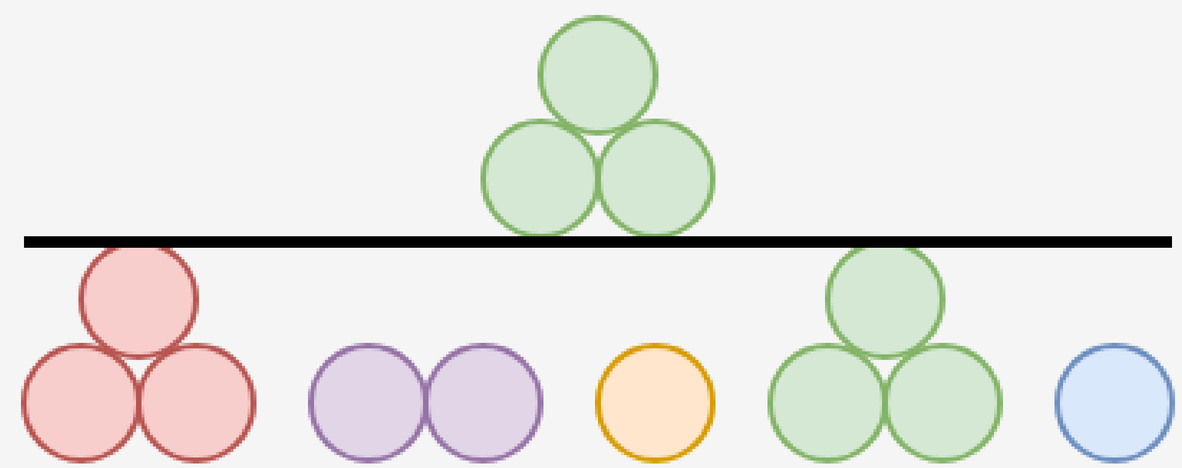
# Interpreting logit values

- The **sign** of the coefficients means the same as before
  - +: *increases* the likelihood of $y$ occurring
  - -: *decreases* the likelihood of $y$ occurring
- The level of a coefficient is different
  - The relationship isn't linear between $x_i$ and $y$ now
  - Instead, coefficients are in log odds
    - Thus, $e^{\beta_i}$ gives you the *odds*, $o$
- You can interpret the odds for a coefficient
  - Increased by $[o - 1]\%$
- You need to sum all relevant log odds before converting to a probability!

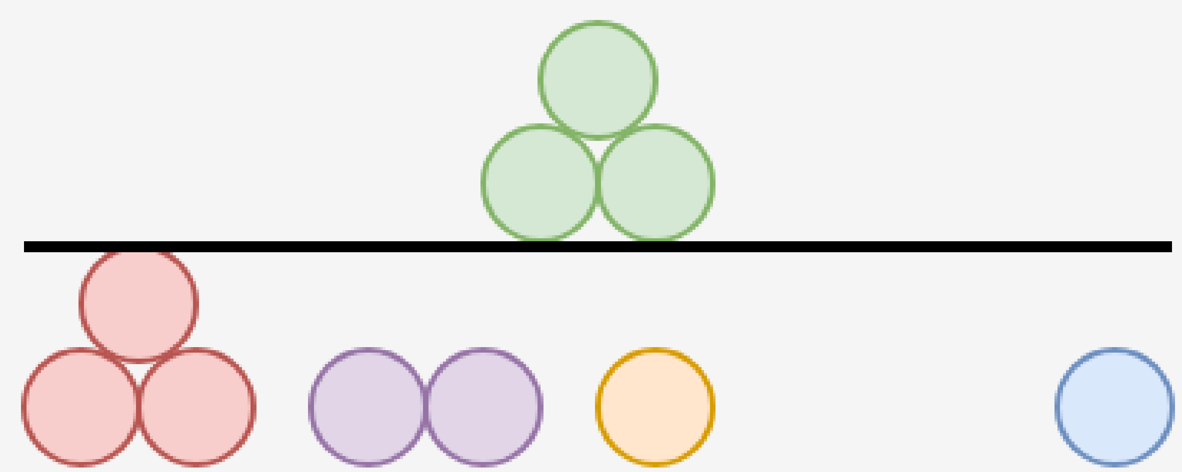
# Odds vs probability

We have the following 10 objects:

The **probability** of green is:  3/10



The **odds** of green is:  3 to 7



# Example logit regression

> Do holidays increase the likelihood that a department more than doubles its store's average weekly sales across departments?

```r
# Create the binary variable from Walmart sales data
df$double <- ifelse(df$Weekly_Sales > df$store_avg*2,1,0)
fit <- glm(double ~ IsHoliday, data=df, family=binomial)
tidy(fit)
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     -3.45    0.00924    -373.  0.
## 2 IsHolidayTRUE    0.539   0.0278       19.4 1.09e-83
```

> Holidays increase the odds… but by how much?

# Logistic regression interpretation

# A simple interpretation

- The model we just saw the following model:

$$logodds(Double\ sales) = -3.44 + 0.54 IsHoliday$$

- There are two ways to interpret this:
  1. Coefficient by coefficient
  2. In total

# Interpretting specific coefficients

$$logodds(Double\ sales) = -3.44 + 0.54 IsHoliday$$

- Interpreting specific coefficients is easiest done manually
- Odds for the $IsHoliday$ coefficient are `exp(0.54) = 1.72`
  - This means that having a holiday modifies the baseline (i.e., non-Holiday) odds by 1.72 to 1
    - Where 1 to 1 is considered no change
  - Baseline is 0.032 to 1

```
# Automating the above:
exp(coef(fit))
```

```
##   (Intercept) IsHolidayTRUE
##    0.03184725    1.71367497
```

# Interpretting in total

- It is important to note that log odds are additive
  - So, calculate a new log odd by plugging in values for variables and adding it all up
    - Holiday: $-3.44 + 0.54 * 1 = -2.9$
    - No holiday: $-3.44 + 0.54 * 0 = -3.44$
- Then calculate odds and log odds like before
  - With holiday: `exp(-2.9) = 0.055`
  - Without holiday: `exp(-3.44) = 0.032`
  - Ratio of holiday to without: 1.72!
    - This is the individual log odds for holiday

> We need to specify values to calculate log odds in total

# Converting to probabilities

- We can calculate a probability at any given point using the log odds

$$Probability = \frac{log\ odds}{log\ odds + 1}$$

  - Probability of double sales…
    - With a holiday: `0.055 / (0.55 + 1) = 0.052`
    - Without a holiday: `0.032 / (0.032 + 1) = 0.031`

> These are easier to interpret, but require specifying values

# Using predict() to simplify it

- `predict()` can calculate log odds and probabilities for us with minimal effort
  - Specify `type="response"` to get probabilities

```
test_data <- as.data.frame(IsHoliday = c(0,1))
predict(model, test_data)   # log odds
```

```
## [1] -3.44 -2.90
```

```
predict(model, test_data, type="response")  #probabilities
```

```
## [1] 0.03106848 0.05215356
```

  - Here, we see the baseline probability is 3.1%
  - The probability of doubling sales on a holiday is higher, at 5.2%

# R practice: Logit

- A continuation of last week's practices answering:
  - Is Walmart more likely to see a year over year decrease in quarterly revenue during a recession?
- Practice using `mutate()` and `glm()`
- Do exercises 1 and 2 in today's practice file
  - R Practice
  - Shortlink: rmc.link/420r4

# Logistic regression interpretation redux

# What about more complex models?

- Continuous inputs in the model
  - What values do we pick to determine probabilities?
- Multiple inputs?
  - We can scale up what we did, but things get messy
    - Mathematically, the inputs get interacted within the inner workings of logit…
      - So the impact of each input depends on the values of the others!

# Consider this model

```
model2 <- glm(double ~ IsHoliday + Temperature + Fuel_Price, data=df, family=binom
summary(model2)
```

```
##
## Call:
## glm(formula = double ~ IsHoliday + Temperature + Fuel_Price,
##     family = binomial, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.4113  -0.2738  -0.2464  -0.2213   2.8562
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.7764917  0.0673246  -26.39   <2e-16 ***
## IsHolidayTRUE  0.3704298  0.0284395   13.03   <2e-16 ***
## Temperature   -0.0108268  0.0004698  -23.04   <2e-16 ***
## Fuel_Price    -0.3091950  0.0196234  -15.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 120370  on 421569  degrees of freedom
```

# Odds and probabilities

```
# Odds
exp(coef(model2))
```

```
##    (Intercept) IsHolidayTRUE   Temperature    Fuel_Price
##      0.1692308     1.4483570     0.9892316     0.7340376
```

```
# Typical September days
hday_sep    <- mean(predict(model2, filter(df, IsHoliday, month==9), type="respons
no_hday_sep <-  mean(predict(model2, filter(df, !IsHoliday, month==9), type="respo
# Typical December days
hday_dec    <- mean(predict(model2, filter(df, IsHoliday, month==12), type="respon
no_hday_dec <-  mean(predict(model2, filter(df, !IsHoliday, month==12), type="resp

html_df(data.frame(Month=c(9,9,12,12),
                   IsHoliday=c(FALSE,TRUE,FALSE,TRUE),
                   Probability=c(no_hday_sep, hday_sep, no_hday_dec, hday_dec)))
```

| Month | IsHoliday | Probability |
|---|---|---|
| 9 | FALSE | 0.0266789 |
| 9 | TRUE | 0.0374761 |
| 12 | FALSE | 0.0398377 |
| 12 | TRUE | 0.0586483 |

# A bit easier: Marginal effects

> Marginal effects tell us the average change in our output for a change of 1 to an input

- The above definition is very similar to how we interpret linear regression coefficients
  - The only difference is the word *average* – the effect changes a bit depending on the input data
- Using `margins`, we can calculate marginal effects
- There are a few types that we could calculate:
  - An *Average Marginal Effect* tells us what the average effect of an input is across all values in our data
    - This is the default method in the package
  - We can also specify a specific value to calculate marginal effects at (like with our probabilities last slides)

# Marginal effects in action

```
# Calculate AME marginal effects
library(margins)
m <- margins(model2)
m
```

```
##   Temperature Fuel_Price IsHoliday
##    -0.0003377  -0.009644   0.01334
```

- A holiday increase the probability of doubling by a flat 1.33%
  - Not too bad when you consider that the probability of doubling is 3.23%
- If the temperature goes up by 1°F (0.55°C), the probability of doubling changes by -0.03%
- If the fuel price increases by 1 USD for 1 gallon of gas, the probability of doubling changes by -0.96%

# `margins` niceties

- We can get some extra information about our marginal effects through `summary()`:

```
summary(m) %>%
  html_df()
```

| factor | AME | SE | z | p | lower | upper |
|---|---|---|---|---|---|---|
| Fuel_Price | -0.0096438 | 0.0006163 | -15.64800 | 0 | -0.0108517 | -0.0084359 |
| IsHoliday | 0.0133450 | 0.0011754 | 11.35372 | 0 | 0.0110413 | 0.0156487 |
| Temperature | -0.0003377 | 0.0000149 | -22.71255 | 0 | -0.0003668 | -0.0003085 |

  - Those p-values work just like with our linear models
  - We also get a confidence interval
    - Which we can plot!

# Plotting marginal effects

```
plot(m, which=summary(m)$factor)
```


> Note: The `which...` part is absolutely necessary at the moment due to a bug in the package

# Marginal effects at a specified value

```r
margins(model2, at = list(IsHoliday = c(TRUE, FALSE)),
        variables = c("Temperature", "Fuel_Price")) %>%
  summary() %>%
  html_df()
```

| factor | IsHoliday | AME | SE | z | p | lower | upper |
|---|---|---|---|---|---|---|---|
| Fuel_Price | FALSE | -0.0093401 | 0.0005989 | -15.59617 | 0 | -0.0105139 | -0.0081664 |
| Fuel_Price | TRUE | -0.0131335 | 0.0008717 | -15.06650 | 0 | -0.0148420 | -0.0114250 |
| Temperature | FALSE | -0.0003271 | 0.0000146 | -22.46024 | 0 | -0.0003556 | -0.0002985 |
| Temperature | TRUE | -0.0004599 | 0.0000210 | -21.92927 | 0 | -0.0005010 | -0.0004188 |

```r
margins(model2, at = list(Temperature = c(0, 20, 40, 60, 80, 100)),
        variables = c("IsHoliday")) %>%
  summary() %>%
  html_df()
```

| factor | Temperature | AME | SE | z | p | lower | upper |
|---|---|---|---|---|---|---|---|
| IsHoliday | 0 | 0.0234484 | 0.0020168 | 11.62643 | 0 | 0.0194955 | 0.0274012 |
| IsHoliday | 20 | 0.0194072 | 0.0016710 | 11.61387 | 0 | 0.0161320 | 0.0226824 |
| IsHoliday | 40 | 0.0159819 | 0.0013885 | 11.51001 | 0 | 0.0132604 | 0.0187033 |
| IsHoliday | 60 | 0.0131066 | 0.0011592 | 11.30623 | 0 | 0.0108345 | 0.0153786 |
| IsHoliday | 80 | 0.0107120 | 0.0009732 | 11.00749 | 0 | 0.0088046 | 0.0126193 |
| IsHoliday | 100 | 0.0087305 | 0.0008213 | 10.62977 | 0 | 0.0071207 | 0.0103402 |

# Today's Application: Shipping delays

# The question

> Can we leverage global weather data to predict shipping delays?

# A bit about shipping data

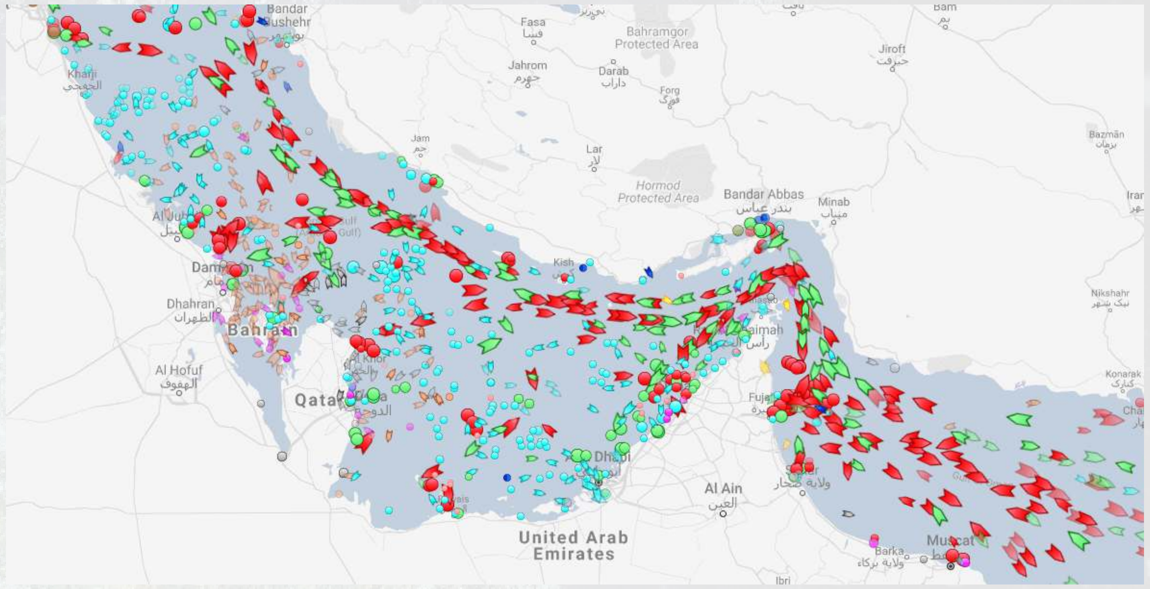
- WRDS doesn’t have shipping data
- There are, however, vendors for shipping data, such as:

- They pretty much have any data you could need:
  - Over 650,000 ships tracked using ground and satellite based AIS
    - AIS: Automatic Identification System
  - Live mapping
  - Weather data
  - Fleet tracking
  - Port congestion
  - Inmarsat support for ship operators

# What can we see from naval data?

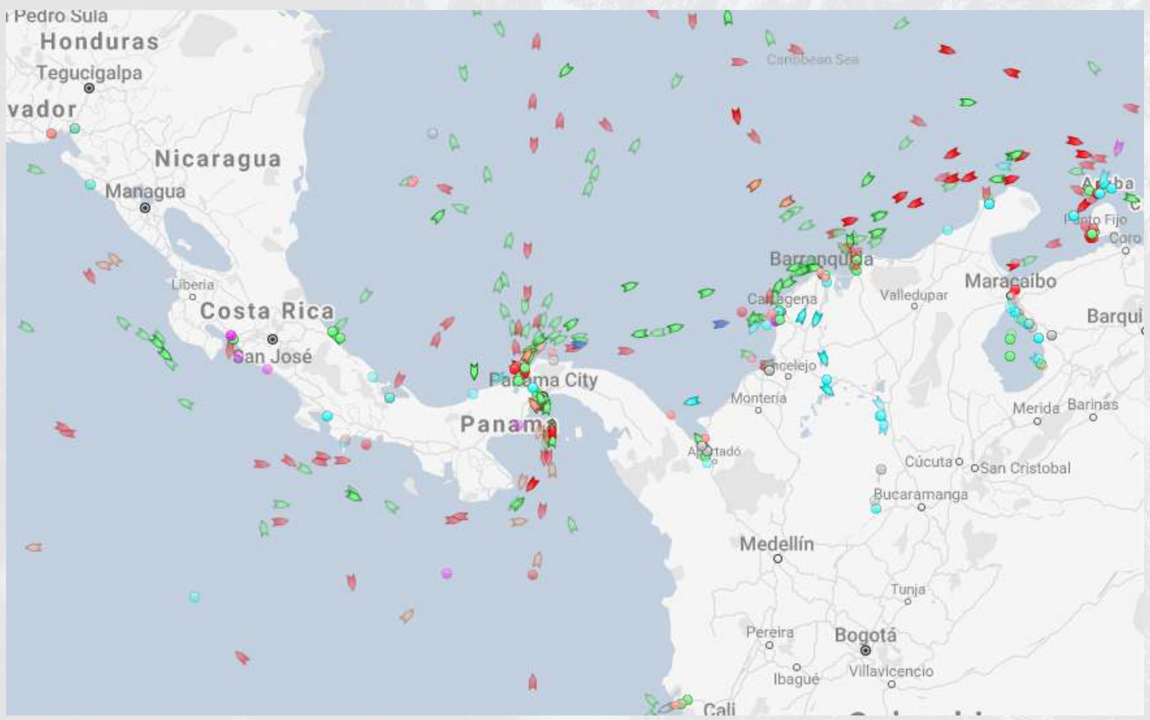
Yachts in the Mediterranean

Oil tankers in the Persian gulf

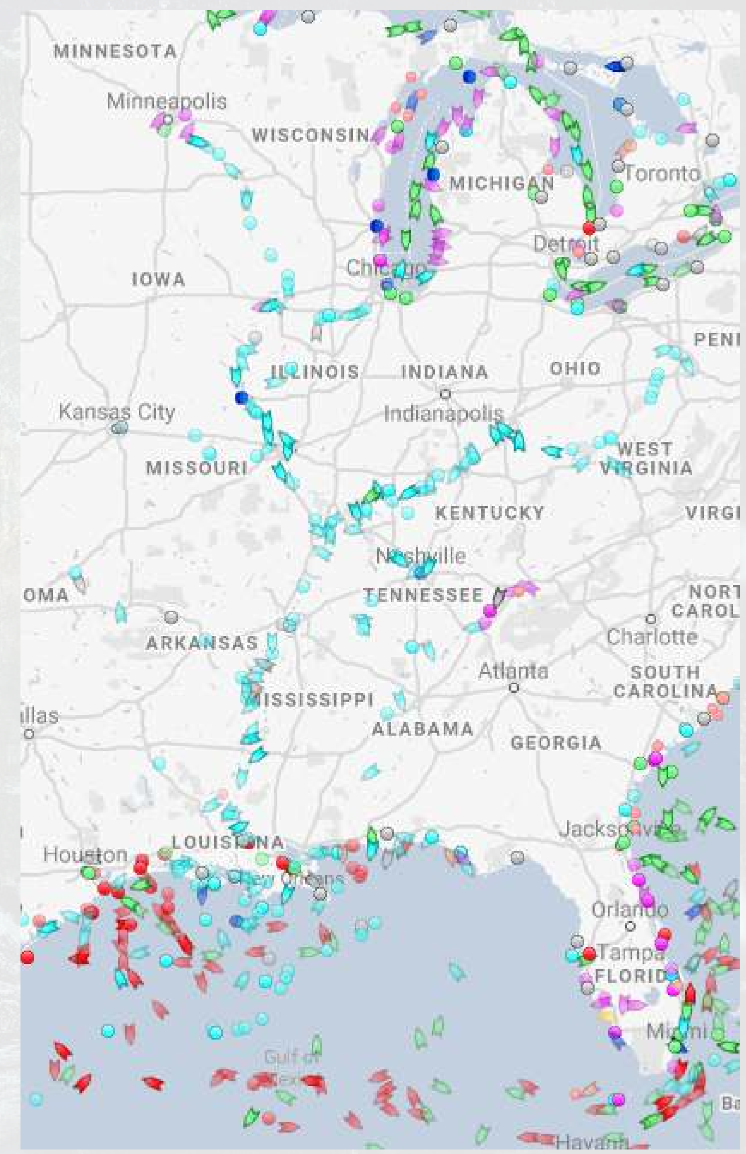


# What can we see from naval data?

Shipping route via the Panama canal

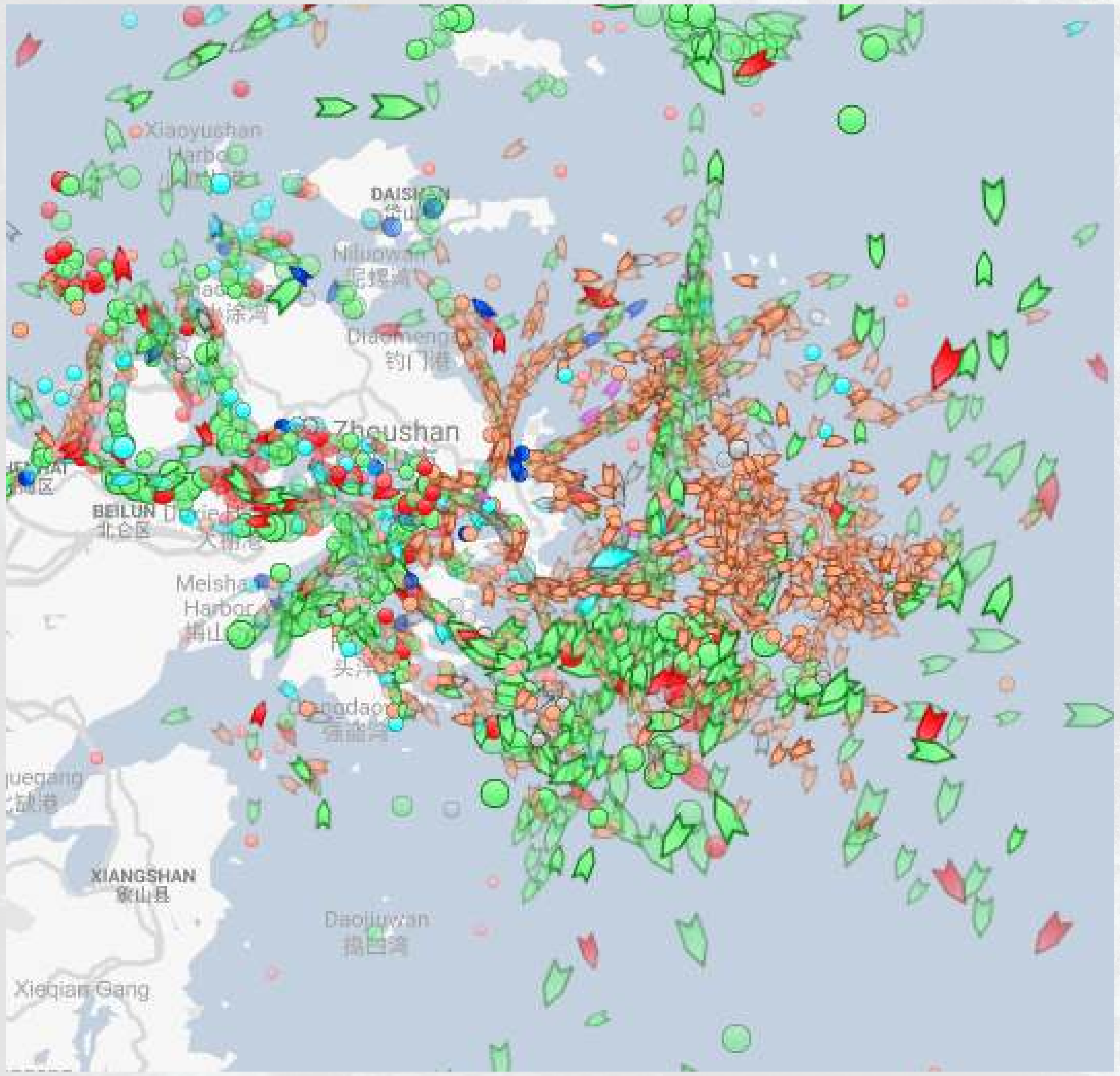



River shipping on the Mississippi river, USA

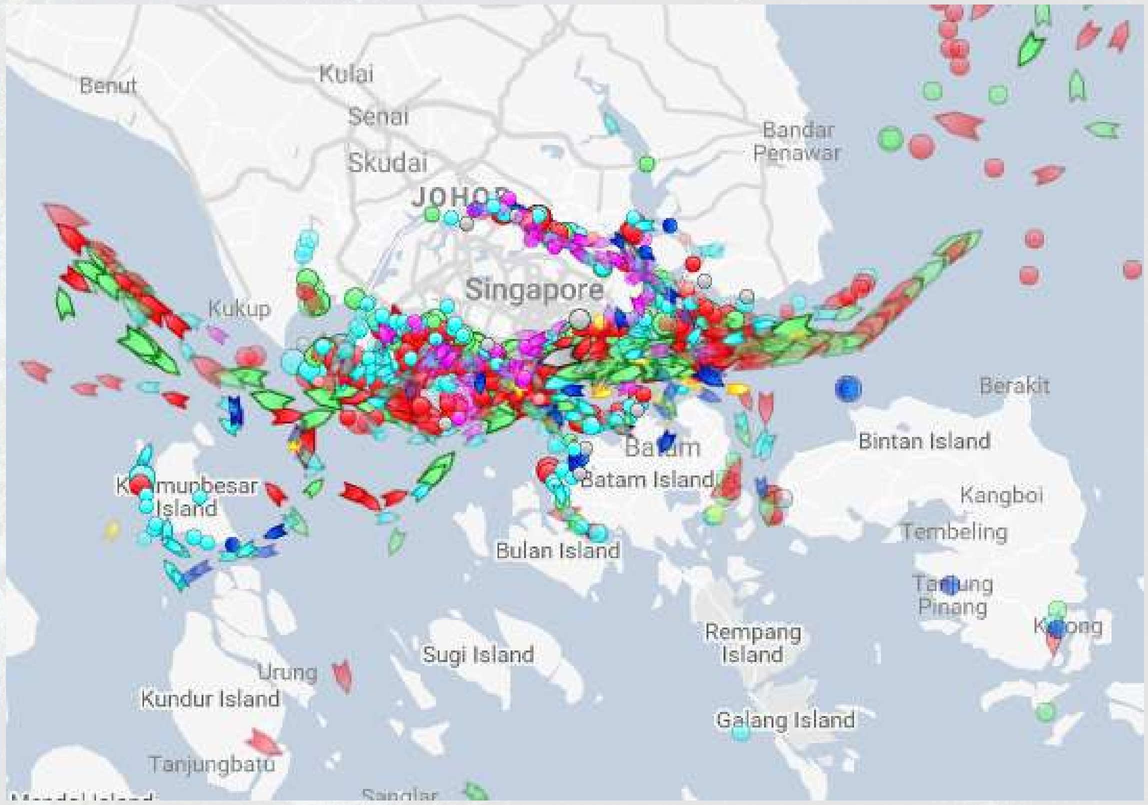


# What can we see from naval data?

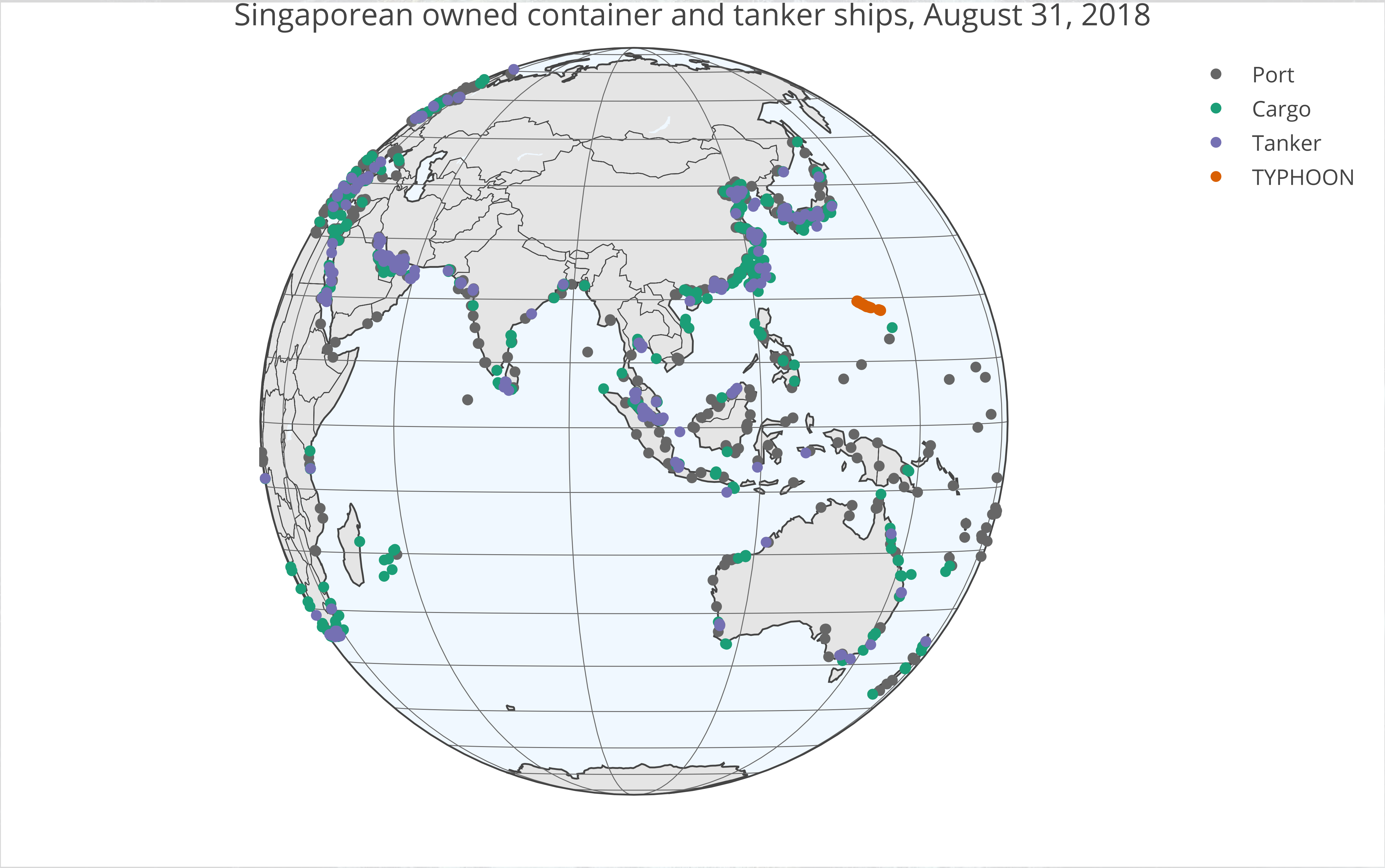
Busiest ports by containers and tons (Shanghai & Ningbo-Zhoushan, China)




Busiest port for transshipment (Singapore)



# Examining Singaporean owned ships



# Code for last slide's map

```
library(plotly)  # for plotting
library(RColorBrewer)  # for colors
# plot with boats, ports, and typhoons
# Note: geo is defined in the appendix -- it controls layout
palette = brewer.pal(8, "Dark2")[c(1,8,3,2)]
p <- plot_geo(colors=palette) %>%
  add_markers(data=df_ports, x = ~port_lon, y = ~port_lat, color = "Port") %>%
  add_markers(data=df_Aug31, x = ~lon, y = ~lat, color = ~ship_type,
              text=~paste('Ship name',shipname)) %>%
  add_markers(data=typhoon_Aug31, x = ~lon, y = ~lat, color="TYPHOON",
              text=~paste("Name", typhoon_name)) %>%
   layout(showlegend = TRUE, geo = geo,
    title = 'Singaporean owned container and tanker ships, August 31, 2018')
p
```

- `plot_geo()` is from `plotly`
- `add_markers()` adds points to the map
- `layout()` adjusts the layout
- Within geo, a list, the following makes the map a globe
  - `projection=list(type="orthographic")`

# Singaporean ship movement

Link to ship movement animation

# Code for last slide's map

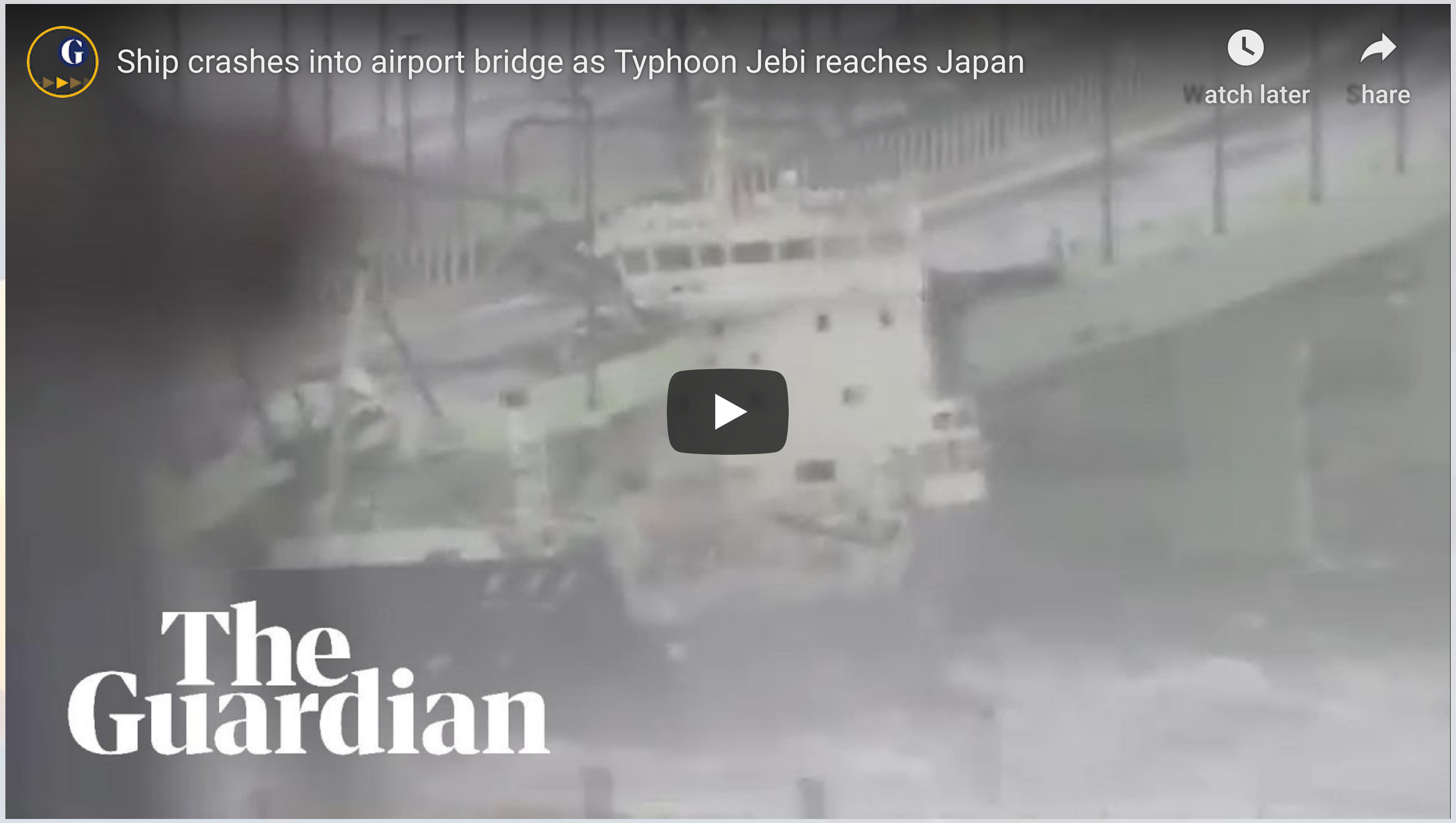
```
library(sf)  # Note: very difficult to install except on Windows
library(maps)
# Requires separately installing "maptools" and "rgeos" as well
# This graph requires ~7GB of RAM to render
world1 <- sf::st_as_sf(map('world', plot = FALSE, fill = TRUE))

df_all <- df_all %>% arrange(run, imo)

p <- ggplot(data = world1) +
  geom_sf() +
  geom_point(data = df_all, aes(x = lon, y = lat, frame=frame,
                                text=paste("name:",shipname)))
ggplotly(p) %>%
   animation_opts(
    1000, easing = "linear", redraw = FALSE)
```

- `world1` contains the map data
- `geom_sf()` plots map data passed to `ggplot()`
- `geom_point()` plots ship locations as longitude and latitude
- `ggplotly()` converts the graph to html and animates it
  - Animation follows the `frame` aesthetic

# What might matter for shipping?

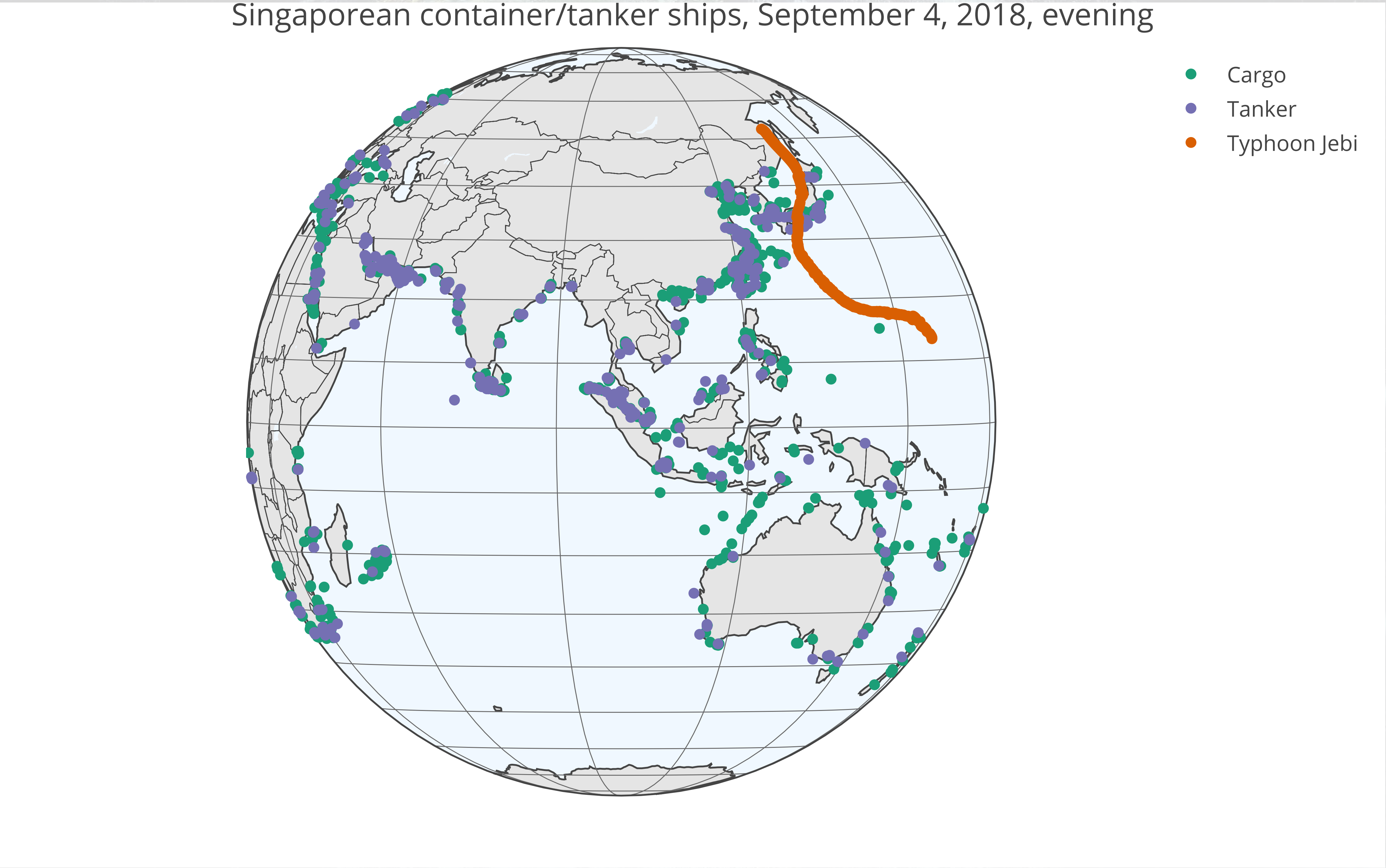
> What observable events or data might provide insight as to whether a naval shipment will be delayed or not?

# Typhoon Jebi




- link

# Typhoons in the data

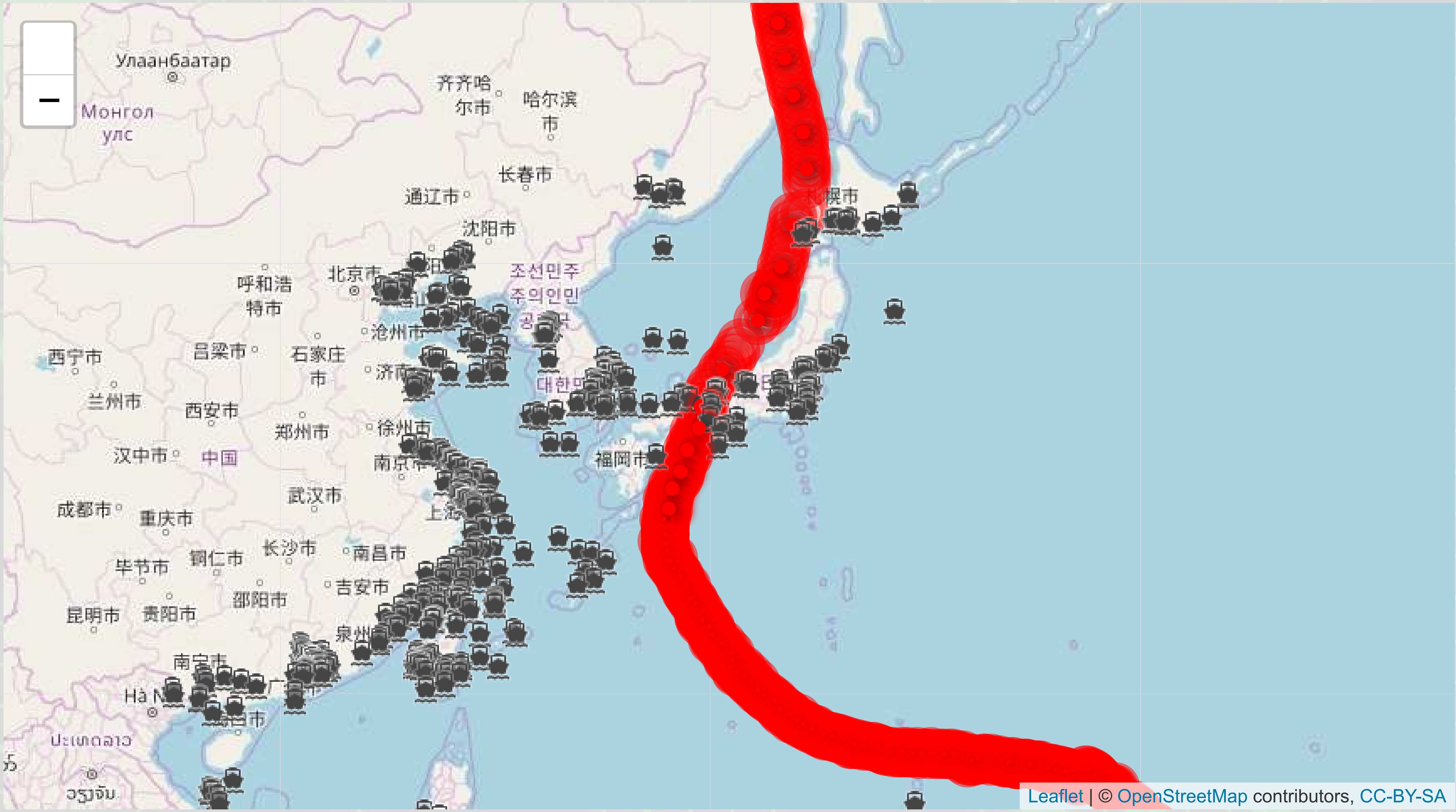


# Code for last slide’s map

```
# plot with boats and typhoons
palette = brewer.pal(8, "Dark2")[c(1,3,2)]
p <- plot_geo(colors=palette) %>%
  add_markers(data=df_all[df_all$frame == 14,], x = ~lon, y = ~lat,
              color = ~ship_type, text=~paste('Ship name',shipname)) %>%
  add_markers(data=typhoon_Jebi, x = ~lon,
              y = ~lat, color="Typhoon Jebi",
              text=~paste("Name", typhoon_name, "</br>Time: ", date)) %>%
   layout(showlegend = TRUE, geo = geo,
    title = 'Singaporean container/tanker ships, September 4, 2018, evening')
p
```

- This map is made the same way as the first map

# Typhoons in the data using leaflet



# Code for last slide’s map

```
library(leaflet)
library(leaflet.extras)

# typhoon icons
icons <- pulseIcons(color='red',
  heartbeat = ifelse(typhoon_Jebi$intensity_vmax > 150/1.852, 0.8,
    ifelse(typhoon$intensity_vmax < 118/1.852, 1.6, 1.2)),
  iconSize=ifelse(typhoon_Jebi$intensity_vmax > 150/1.852, 5,
    ifelse(typhoon_Jebi$intensity_vmax < 118/1.852, 2, 3)))

# ship icons
shipicons <- iconList(
  ship = makeIcon("../Figures/ship.png", NULL, 18, 18)
)
leaflet() %>%
  addTiles() %>%
  setView(lng = 136, lat = 34, zoom=4) %>%
  addPulseMarkers(data=typhoon_Jebi[seq(1,nrow(typhoon_Jebi),5),], lng=~lon,
                  lat=~lat, label=~date, icon=icons) %>%
  addCircleMarkers(data=typhoon_Jebi[typhoon_Jebi$intensity_vmax > 150/1.852,],
    lng=~lon, lat=~lat,stroke = TRUE, radius=3, color="red", label=~date) %>%
```

# R Practice on mapping

- Practice mapping typhoon data
  - 1 map using `plotly`
  - 1 map using `leaflet`
- Practice using `plotly` and `leaflet`
  - No practice using `ggplot2` as `sf` is missing on DataCamp light
    - And `sf` can be tough to install for anyone on a Mac
- Do exercises 3 and 4 in today's practice file
  - R Practice
  - Shortlink: rmc.link/420r4

# Predicting delays due to typhoons

# Data

- If the ship will report a delay of at least 3 hours some time in the next 12-24 hours
- What we have:
  - Ship location
  - Typhoon location
  - Typhoon wind speed

> We need to calculate distance between ships and typhoons

# Distance for geo

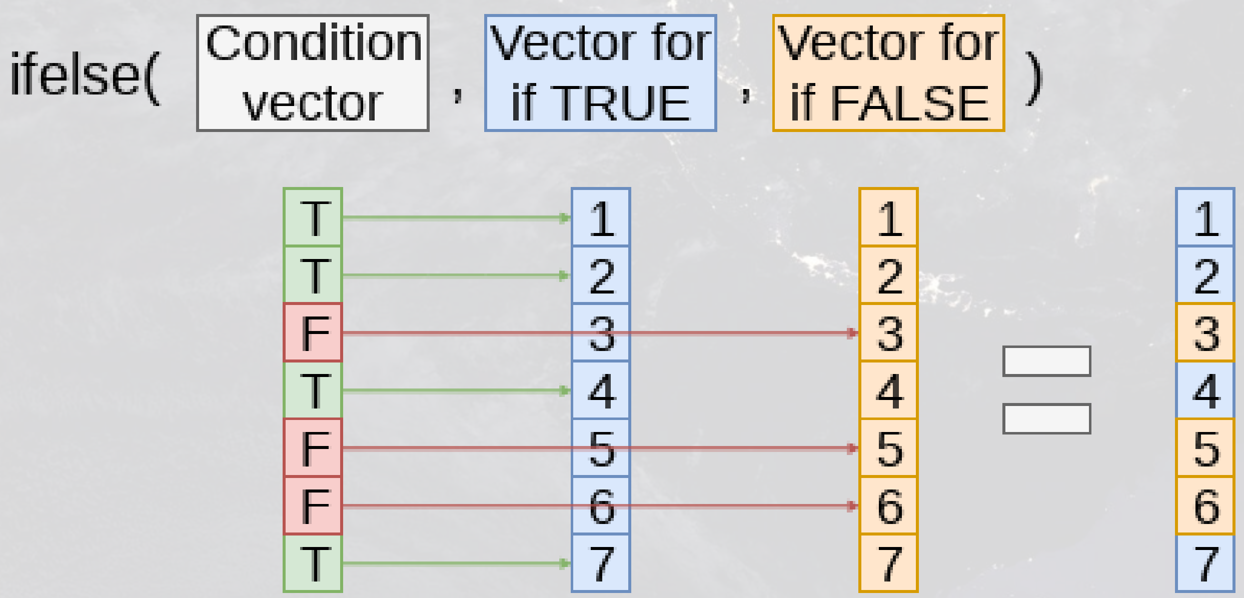
- There are a number of formulas for this
  - *Haversine* for a simple calculation
  - *Vincenty's formulae* for a complex, incredibly accurate calculation
    - Accurate within **0.5mm**
- Use `distVincentyEllipsoid()` from `geosphere` to get a reasonably quick and accurate calculation
  - Calculates distance between two sets of points, x and y, structured as matrices
  - Matrices must have longitude in the first column and latitude in the second column
  - Provides distance in meters by default

```
library(geosphere)
x <- as.matrix(df3[,c("lon","lat")])  # ship location
y <- as.matrix(df3[,c("ty_lon","ty_lat")]) # typhoon location

df3$dist_typhoon <- distVincentyEllipsoid(x, y) / 1000
```

# Clean up

- Some indicators to cleanly capture how far away the typhoon is

```
df3$typhoon_500 = ifelse(df3$dist_typhoon < 500 &
                         df3$dist_typhoon >= 0, 1, 0)
df3$typhoon_1000 = ifelse(df3$dist_typhoon < 1000 &
                          df3$dist_typhoon >= 500, 1, 0)
df3$typhoon_2000 = ifelse(df3$dist_typhoon < 2000 &
                          df3$dist_typhoon >= 1000, 1, 0)
```



# Do typhoons delay shipments?

```r
fit1 <- glm(delayed ~ typhoon_500 + typhoon_1000 + typhoon_2000, data=df3,
            family=binomial)
summary(fit1)
```

```
##
## Call:
## glm(formula = delayed ~ typhoon_500 + typhoon_1000 + typhoon_2000,
##     family = binomial, data = df3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.2502  -0.2261  -0.2261  -0.2261   2.7127
##
## Coefficients:
##              Estimate Std. Error  z value Pr(>|z|)
## (Intercept)  -3.65377    0.02934 -124.547   <2e-16 ***
## typhoon_500   0.14073    0.16311    0.863   0.3883
## typhoon_1000  0.20539    0.12575    1.633   0.1024
## typhoon_2000  0.16059    0.07106    2.260   0.0238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 14329  on 59184  degrees of freedom
```

It appears so!

# Interpretation of coefficients

```r
odds1 <- exp(coef(fit1))
odds1
```

```
##   (Intercept)  typhoon_500 typhoon_1000 typhoon_2000
##    0.02589334   1.15111673   1.22800815   1.17420736
```

- Ships 1,000 to 2,000 km from a typhoon have a 17% increased odds of having a delay

```r
m1 <- margins(fit1)
summary(m1)
```

```
##        factor    AME     SE      z      p   lower  upper
##  typhoon_1000 0.0052 0.0032 1.6322 0.1026 -0.0010 0.0115
##  typhoon_2000 0.0041 0.0018 2.2570 0.0240  0.0005 0.0076
##   typhoon_500 0.0036 0.0042 0.8626 0.3883 -0.0046 0.0117
```

- Ships 1,000 to 2,000 km from a typhoon have an extra 0.41% chance of having a delay (baseline of 2.61%)

# What about typhoon intensity?

- Hong Kong's typhoon classification: Official source
  1. 41-62 km/h: Tropical depression
  2. 63-87 km/h: Tropical storm
  3. 88-117 km/h: Severe tropical storm
  4. 118-149 km/h: **Typhoon**
  5. 150-184 km/h: **Severe typhoon**
  6. 185+km/h: **Super typhoon**

```
# Cut makes a categorical variable out of a numerical variable using specified bin
df3$Super <- ifelse(df3$intensity_vmax * 1.852 > 185, 1, 0)
df3$Moderate <- ifelse(df3$intensity_vmax * 1.852 >= 88 &
                         df3$intensity_vmax * 1.852 < 185, 1, 0)
df3$Weak <- ifelse(df3$intensity_vmax * 1.852 >= 41 &
                     df3$intensity_vmax * 1.852 < 88, 1, 0)
df3$HK_intensity <- cut(df3$intensity_vmax * 1.852 ,c(-1,41, 62, 87, 117, 149, 999
table(df3$HK_intensity)
```

```
##
##   (-1,41]   (41,62]   (62,87]  (87,117] (117,149] (149,999]
##      3398     12039     12615     11527      2255     21141
```

# Typhoon intensity and delays

```r
fit2 <- glm(delayed ~ (typhoon_500 + typhoon_1000 + typhoon_2000) :
              (Weak + Moderate + Super), data=df3,
            family=binomial)
tidy(fit2)
```

```
## # A tibble: 10 x 5
##    term                  estimate std.error statistic p.value
##    <chr>                    <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)           -3.65       0.0290 -126.     0      
##  2 typhoon_500:Weak      -0.00879    0.213    -0.0413 0.967  
##  3 typhoon_500:Moderate   0.715      0.251     2.86   0.00430
##  4 typhoon_500:Super     -8.91     123.       -0.0726 0.942  
##  5 typhoon_1000:Weak      0.250      0.161     1.55   0.121  
##  6 typhoon_1000:Moderate  0.123      0.273     0.451  0.652  
##  7 typhoon_1000:Super    -0.0269     0.414    -0.0648 0.948  
##  8 typhoon_2000:Weak      0.182      0.101     1.80   0.0723 
##  9 typhoon_2000:Moderate  0.0253     0.134     0.189  0.850  
## 10 typhoon_2000:Super     0.311      0.136     2.29   0.0217
```

> Moderate storms predict delays when within 500km

> Super typhoons predict delays when 1,000 to 2,000km away

# Interpretation of coefficients

```
odds2 <- exp(coef(fit2))
odds2[c(1, 3, 8, 10)]
```

```
##           (Intercept) typhoon_500:Moderate    typhoon_2000:Weak
##            0.02589637           2.04505487           1.19923751
##    typhoon_2000:Super
##            1.36507575
```

- Ships within 500km of a moderately strong storm have 105% higher odds of a delay
- Ships 1,000 to 2,000km from a weak typhoon have 20% higher odds of a delay
- Ships 1,000 to 2,000km from a super typhoon have 37% higher odds of a delay

# Marginal effects

```r
m2 <- margins(fit2)
summary(m2) %>%
  html_df()
```

| factor | AME | SE | z | p | lower | upper |
|---|---|---|---|---|---|---|
| Moderate | 0.0007378 | 0.0006713 | 1.0990530 | 0.2717449 | -0.0005779 | 0.0020535 |
| Super | -0.0050241 | 0.0860163 | -0.0584087 | 0.9534231 | -0.1736129 | 0.1635647 |
| typhoon_1000 | 0.0035473 | 0.0036186 | 0.9802921 | 0.3269420 | -0.0035450 | 0.0106396 |
| typhoon_2000 | 0.0039224 | 0.0017841 | 2.1985908 | 0.0279070 | 0.0004257 | 0.0074191 |
| typhoon_500 | -0.0440484 | 0.6803640 | -0.0647424 | 0.9483791 | -1.3775373 | 1.2894405 |
| Weak | 0.0009975 | 0.0005154 | 1.9353011 | 0.0529534 | -0.0000127 | 0.0020077 |

- Delays appear to be driven mostly by 2 factors:
  1. A typhoon 1,000 to 2,000 km away from the ship
  2. Weak typhoons

# What might matter for shipping?

> What other observable events or data might provide insight as to whether a naval shipment will be delayed or not?

- What is the reason that this event or data would be useful in predicting delays?
  - I.e., how does it fit into your mental model?

# End matter

# For next week

- For next week:
  - Second individual assignment
    - Finish by the end of *next* Thursday
    - Submit on eLearn
  - Think about who you want to work with for the project

# Packages used for these slides

- `broom`
- `geosphere`
- `kableExtra`
- `knitr`
- `leaflet`
- `leaflet.extras`
- `lubridate`
- `magrittr`
- `margins`
- `maps`
- `maptools`
- `plotly`
- `revealjs`
- `rgeos`
- `sf`
- `tidyverse`

# Custom code

```r
# styling for plotly maps
geo <- list(
  showland = TRUE,
  showlakes = TRUE,
  showcountries = TRUE,
  showocean = TRUE,
  countrywidth = 0.5,
  landcolor = toRGB("grey90"),
  lakecolor = toRGB("aliceblue"),
  oceancolor = toRGB("aliceblue"),
  projection = list(
    type = 'orthographic',  # detailed at https://plot.ly/r/reference/#layout-geo-projection
    rotation = list(
      lon = 100,
      lat = 1,
      roll = 0
    )
  ),
  lonaxis = list(
    showgrid = TRUE,
    gridcolor = toRGB("gray40"),
    gridwidth = 0.5
  ),
  lataxis = list(
    showgrid = TRUE,
    gridcolor = toRGB("gray40"),
    gridwidth = 0.5
  )
)
```