Final Project

Anish Shenoy, Jae Yoon Kim

11/25/2020

Effects of Global Warming Induced Flooding on Low Lying Areas

Import, Merge, and Clean data

First, import the First Street Foundation (FSF) Flood Risk Summary Statistics by zip found here: https://registry.opendata.aws/fsf-flood-risk/?fbclid=IwAR2JUyZWXcXiKdXuXMXON-1VmcUm6RpCnUVXMiydCrrDCaXHg41zdkD-iI8

```
zip_risk <- read_csv("./data/fsf/v1.1/Zip_level_risk_FEMA_FSF_v1.1.csv")
# zip_risk <- read_csv("/cloud/project/Final/Flood-Data-Analysis-master/data/FSF/v1.1/Zip_level_risk_FE
zip_risk
```

```
## # A tibble: 32,158 x 34
##    zcta5ce count_property count_fema_sfha pct_fema_sfha count_fs_risk_2~
##    <chr>            <dbl>           <dbl>         <dbl>            <dbl>
##  1 01001             4927             550          11.2              526
##  2 01002             6893              81           1.2              122
##  3 01003               76               0           0                  0
##  4 01005             3006               0           0                138
##  5 01007             6474              63           1                100
##  6 01008             1171              10           0.9               37
##  7 01009              373              16           4.3               30
##  8 01010             2317             115           5                138
##  9 01011             1071             170          15.9              125
## 10 01012              470              18           3.8               14
## # ... with 32,148 more rows, and 29 more variables: pct_fs_risk_2020_5 <dbl>,
## #   count_fs_risk_2050_5 <dbl>, pct_fs_risk_2050_5 <dbl>,
## #   count_fs_risk_2020_100 <dbl>, pct_fs_risk_2020_100 <dbl>,
## #   count_fs_risk_2050_100 <dbl>, pct_fs_risk_2050_100 <dbl>,
## #   count_fs_risk_2020_500 <dbl>, pct_fs_risk_2020_500 <dbl>,
## #   count_fs_risk_2050_500 <dbl>, pct_fs_risk_2050_500 <dbl>,
## #   count_fs_fema_difference_2020 <dbl>, pct_fs_fema_difference_2020 <dbl>,
## #   avg_risk_score_all <dbl>, avg_risk_score_2_10 <dbl>,
## #   avg_risk_fsf_2020_100 <dbl>, avg_risk_fsf_2020_500 <dbl>,
## #   avg_risk_score_sfha <dbl>, avg_risk_score_no_sfha <dbl>,
## #   count_floodfactor1 <dbl>, count_floodfactor2 <dbl>,
## #   count_floodfactor3 <dbl>, count_floodfactor4 <dbl>,
## #   count_floodfactor5 <dbl>, count_floodfactor6 <dbl>,
## #   count_floodfactor7 <dbl>, count_floodfactor8 <dbl>,
## #   count_floodfactor9 <dbl>, count_floodfactor10 <dbl>
```

Next, import data about each zip code in the US uing the tidycensus package

```
library(tidycensus)
census_api_key("0a473d1b1e161d8f87da3ad85e59af583f28f1d6")

# Variable codes found here:
# https://api.census.gov/data/2010/dec/sf1/variables.html
# P013001: MEDIAN AGE
# H006001: TOTAL ALL RACES IN HOUSEHOLD
# H006002: TOTAL HOUSEHOLDER WHITE ALONE
# H006003: TOTAL BLACK/AFRICAN-AMERICAN ALONE
zip_dem_data <- get_decennial(geography = "zcta",
                          variables = c(median_age = "P013001",
                                        total_pop = "H006001",
                                        total_white = "H006002",
                                        total_black = "H006003"),
                          year = 2010,
                          output = "wide")

zip_income_data <- get_acs(geography = "zcta",
                           variables = c(median_income = "B19013_001"),
                           year = 2018,
                           output = "wide")
```

Finally, merge the the data by zip code

```
merged <- inner_join(zip_dem_data, zip_risk, by = c("GEOID" = "zcta5ce")) %>%
  inner_join(zip_income_data, by = "GEOID") %>%
  select(-NAME.y) %>%
  mutate(prop_white = total_white/total_pop,
         prop_black = total_black/total_pop,
         prop_nonwhite = 1 - prop_white) %>%
  inner_join(zipcode, by=c('GEOID'='zip')) %>%
  # Get rid of counties with no houses at risk now or in the future
  # Make sure we don't have NA's in demographic info
  filter(count_fs_risk_2050_100 > 0,
         count_fs_risk_2020_100 > 0,
         prop_nonwhite >= 0,
         median_incomeE > 0,
         median_age > 0,
         total_pop > 0)

merged
```

```
## # A tibble: 29,666 x 48
##    GEOID NAME.x median_age total_pop total_white total_black count_property
##    <chr> <chr>       <dbl>     <dbl>       <dbl>       <dbl>          <dbl>
##  1 77477 ZCTA5~       31.6     12968        5486        3364          11109
##  2 77478 ZCTA5~       44.3      9005        5511         610          11770
##  3 77479 ZCTA5~       39       24076       13864        1663          32591
##  4 77480 ZCTA5~       39.6      2981        2419         374           5740
##  5 77482 ZCTA5~       40.3       831         660         125           1637
##  6 77484 ZCTA5~       37.7      3875        3046         339           7320
##  7 77485 ZCTA5~       42.2      1535        1218         161           3478
##  8 77486 ZCTA5~       40        2661        2186         282           5071
##  9 77488 ZCTA5~       38.1      5445        3428        1360           9635
```

```
## 10 77489 ZCTA5~       34.8     11516        1362        9101          14508
## # ... with 29,656 more rows, and 41 more variables: count_fema_sfha <dbl>,
## #   pct_fema_sfha <dbl>, count_fs_risk_2020_5 <dbl>, pct_fs_risk_2020_5 <dbl>,
## #   count_fs_risk_2050_5 <dbl>, pct_fs_risk_2050_5 <dbl>,
## #   count_fs_risk_2020_100 <dbl>, pct_fs_risk_2020_100 <dbl>,
## #   count_fs_risk_2050_100 <dbl>, pct_fs_risk_2050_100 <dbl>,
## #   count_fs_risk_2020_500 <dbl>, pct_fs_risk_2020_500 <dbl>,
## #   count_fs_risk_2050_500 <dbl>, pct_fs_risk_2050_500 <dbl>,
## #   count_fs_fema_difference_2020 <dbl>, pct_fs_fema_difference_2020 <dbl>,
## #   avg_risk_score_all <dbl>, avg_risk_score_2_10 <dbl>,
## #   avg_risk_fsf_2020_100 <dbl>, avg_risk_fsf_2020_500 <dbl>,
## #   avg_risk_score_sfha <dbl>, avg_risk_score_no_sfha <dbl>,
## #   count_floodfactor1 <dbl>, count_floodfactor2 <dbl>,
## #   count_floodfactor3 <dbl>, count_floodfactor4 <dbl>,
## #   count_floodfactor5 <dbl>, count_floodfactor6 <dbl>,
## #   count_floodfactor7 <dbl>, count_floodfactor8 <dbl>,
## #   count_floodfactor9 <dbl>, count_floodfactor10 <dbl>, median_incomeE <dbl>,
## #   median_incomeM <dbl>, prop_white <dbl>, prop_black <dbl>,
## #   prop_nonwhite <dbl>, city <chr>, state <chr>, latitude <dbl>,
## #   longitude <dbl>
```

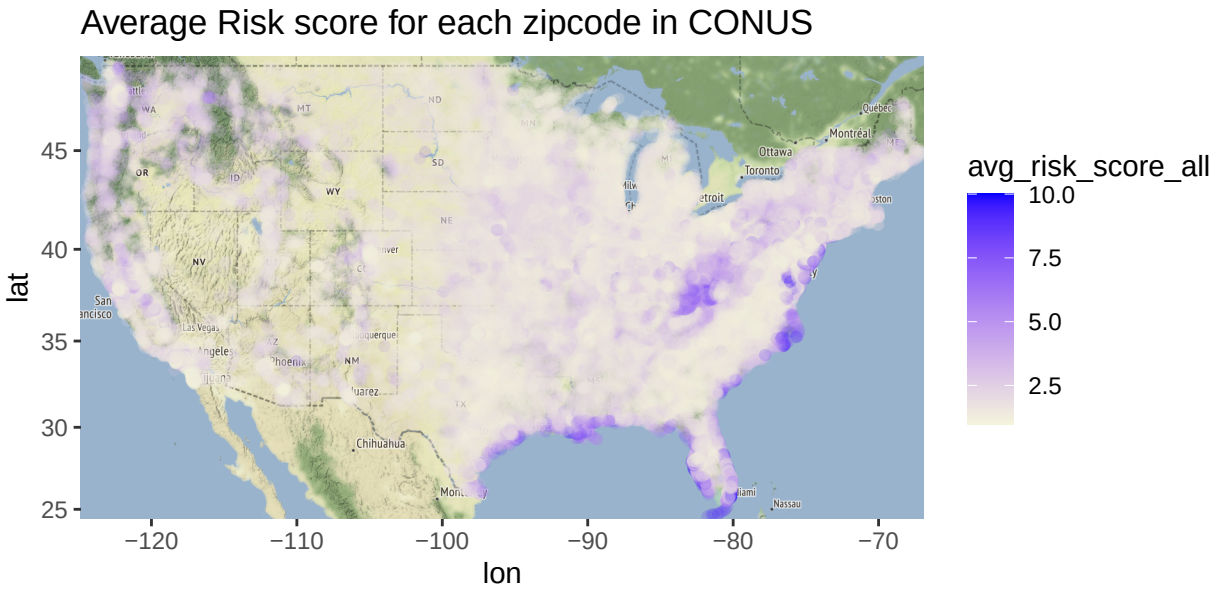
Preliminary Data Exploration

Graph of FEMA average risk scores per zip code overlayed across entire Continental USA.

```
# Download a map of the CONUS
conus_map<- get_map(location=c(-124.848974, 24.396308, -66.885444, 49.384358),
                    zoom=5, maptype = 'terrain',
                    source='osm',color='color')

ggmap(conus_map) +
  geom_point(aes(x=longitude,
                 y=latitude,
                 show_guide = TRUE,
                 colour=avg_risk_score_all),
             data=merged, alpha=.2, na.rm = T)  +
  scale_color_gradient(low="beige", high="blue") +
  labs(title='Average Risk score for each zipcode in CONUS')
```

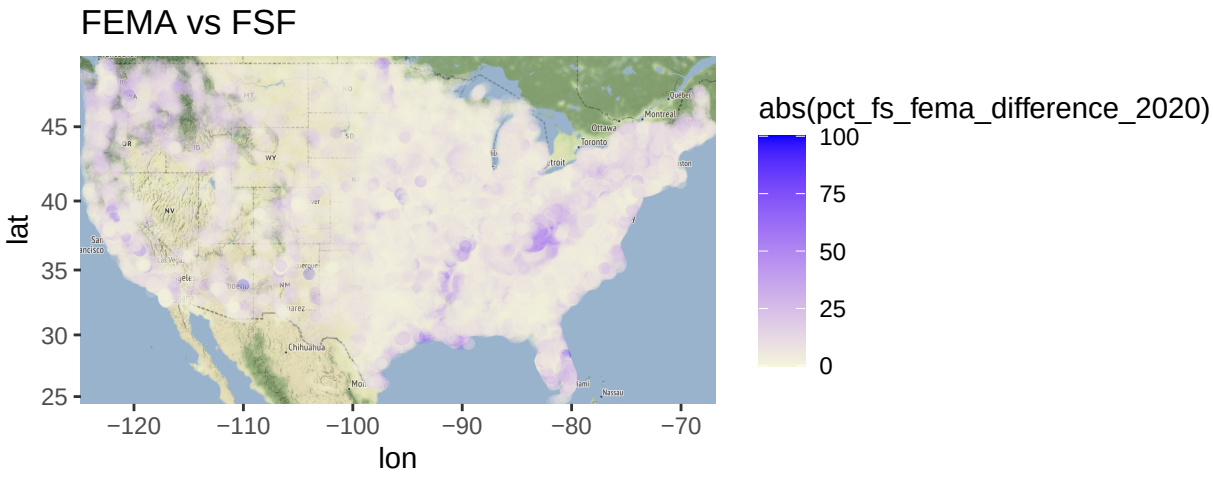
```
## Warning: Ignoring unknown aesthetics: show_guide
```



This map shows where scores from FEMA disagrees with the FSF data

```
# Where did FEMA get it wrong
ggmap(conus_map) +
  geom_point(aes(x=longitude,
                 y=latitude,
                 show_guide = TRUE,
                 colour = abs(pct_fs_fema_difference_2020)),
             data=merged, alpha=.2, na.rm = T)  +
  scale_color_gradient(low="beige", high="blue") +
  labs(title='FEMA vs FSF')
```

```
## Warning: Ignoring unknown aesthetics: show_guide
```


FEMA vs FSF
lat
45
40
35
30
25
−120
−110
−100
−90
−80
−70
lon
abs(pct_fs_fema_difference_2020)
100
75
50
25
0

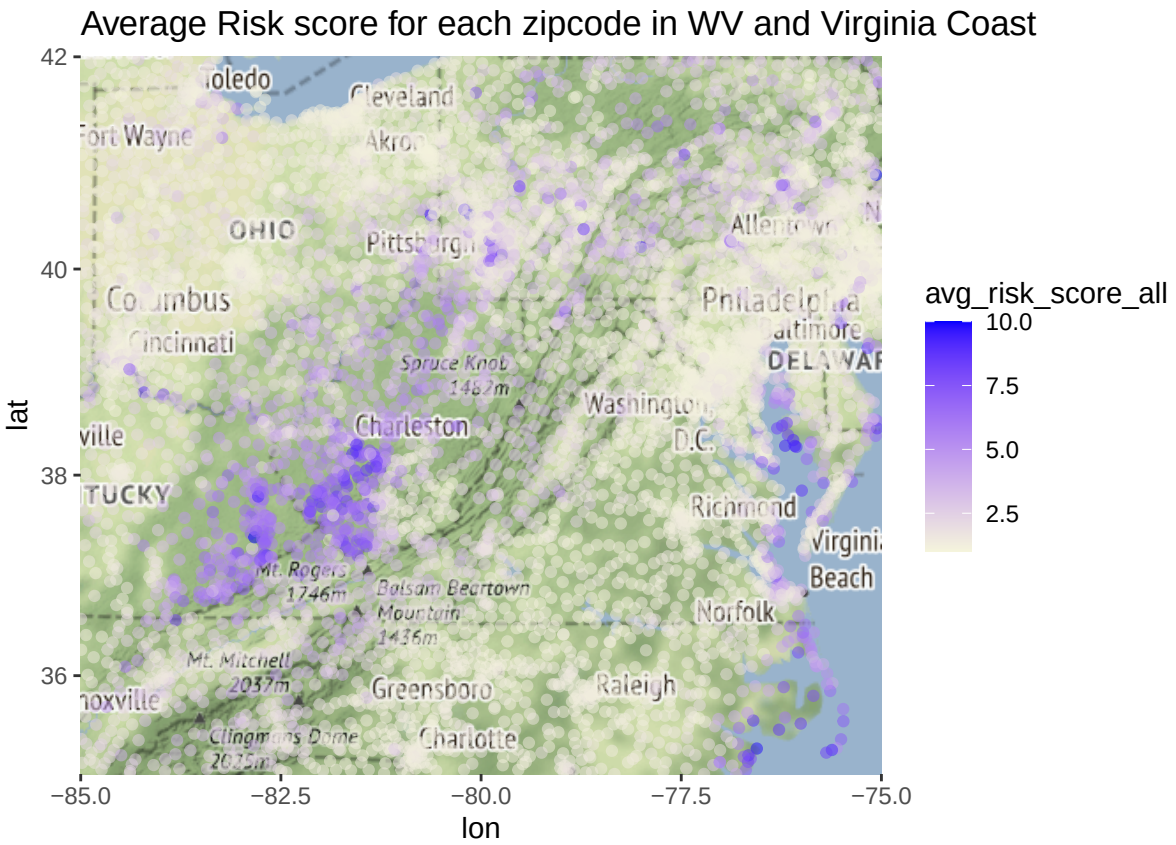
Three regions jump out: Kentucky/West Virginia, Coast of Virginia, and the Gulf Coast/panhandle. We will look at each in more detail.

```
# Download a terrain map of the WV to understand why it's flooding so much
wv_bbox <- c(-85, 35, -75, 42)
wv_map<-get_map(location=wv_bbox, zoom=6, maptype = 'terrain',
             source='osm',color='color', size = c(800, 800))

wv_merged <- filter(.data=merged, latitude >= wv_bbox[2], latitude <= wv_bbox[4],
                    longitude >= wv_bbox[1], longitude <= wv_bbox[3])

# Overlay the risk scores to understand
ggmap(wv_map)+ geom_point(
        aes(x=longitude, y=latitude, show_guide = TRUE, colour=avg_risk_score_all),
        data=wv_merged, alpha=.5, na.rm = T)  +
        scale_color_gradient(low="beige", high="blue") +
        labs(title='Average Risk score for each zipcode in WV and Virginia Coast')
```

```
## Warning: Ignoring unknown aesthetics: show_guide
```



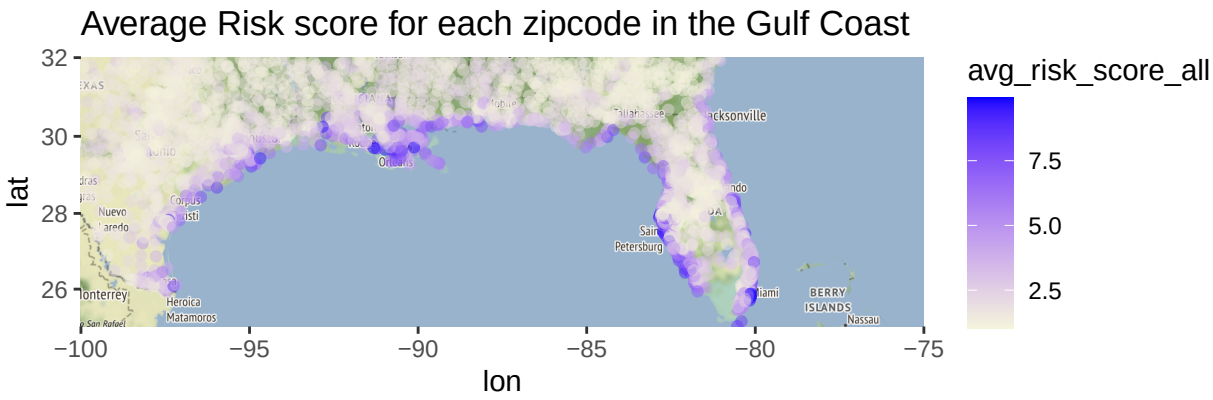
The majority of risk in West Virginia comes from around Charleston along the river.

```
# Download a terrain map of the gulf coast to understand why it's flooding so much
fl_bbox <- c(-100, 25, -75, 32)
fl_map<-get_map(location=fl_bbox, zoom=6, maptype = 'terrain',
             source='osm',color='color')

fl_merged <- filter(.data=merged, latitude >= fl_bbox[2], latitude <= fl_bbox[4],
                    longitude >= fl_bbox[1], longitude <= fl_bbox[3])
```

```
# Overlay the risk scores to understand
ggmap(fl_map)+ geom_point(
        aes(x=longitude, y=latitude, show_guide = TRUE, colour=avg_risk_score_all),
        data=fl_merged, alpha=.5, na.rm = T)  +
        scale_color_gradient(low="beige", high="blue") +
        labs(title='Average Risk score for each zipcode in the Gulf Coast')
```

```
## Warning: Ignoring unknown aesthetics: show_guide
```



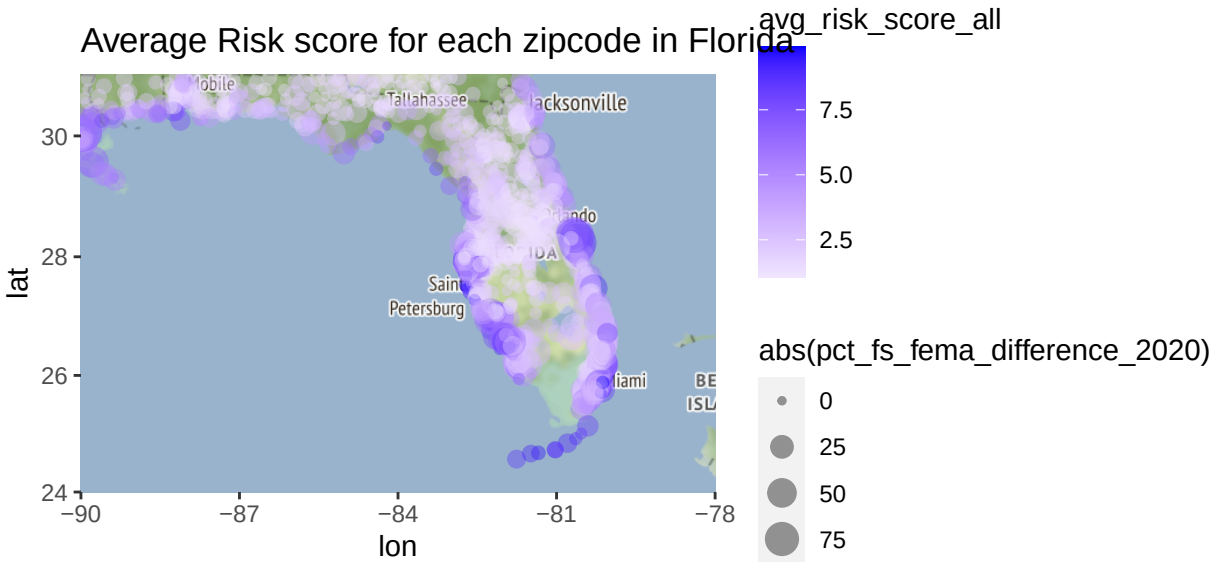
We would like to look at Florida in more depth. Florida is in a unique position where there are large portions of the state where many people live with high risk scores.

```
# Filter data for florida only
fl <- filter(.data=merged, state == 'FL')

fl_only_bbox <- c(-90, 24, -78, 31)
fl_only_merged <- filter(.data=merged, latitude >= fl_only_bbox[2],
                         latitude <= fl_only_bbox[4],
                         longitude >= fl_only_bbox[1],
                         longitude <= fl_only_bbox[3])
fl_only_map<- get_map(location=fl_only_bbox, zoom=6, maptype = 'terrain',
                      source='osm',color='color')

# Overlay the risk scores to understand
# Make the absolute value of the percent diff between FSF and FEMA the size.
# We care about the magnitude of their error,
# not necessarily the size for a broad overview.
ggmap(fl_only_map) +
  geom_point(aes(x=longitude, y=latitude,
                 show_guide = TRUE,
                 colour=avg_risk_score_all,
                 size=abs(pct_fs_fema_difference_2020)),
             data=fl_only_merged, alpha=.4, na.rm = T)  +
  scale_color_gradient2(low="red", mid="white", high="blue") +
  labs(title='Average Risk score for each zipcode in Florida')
```

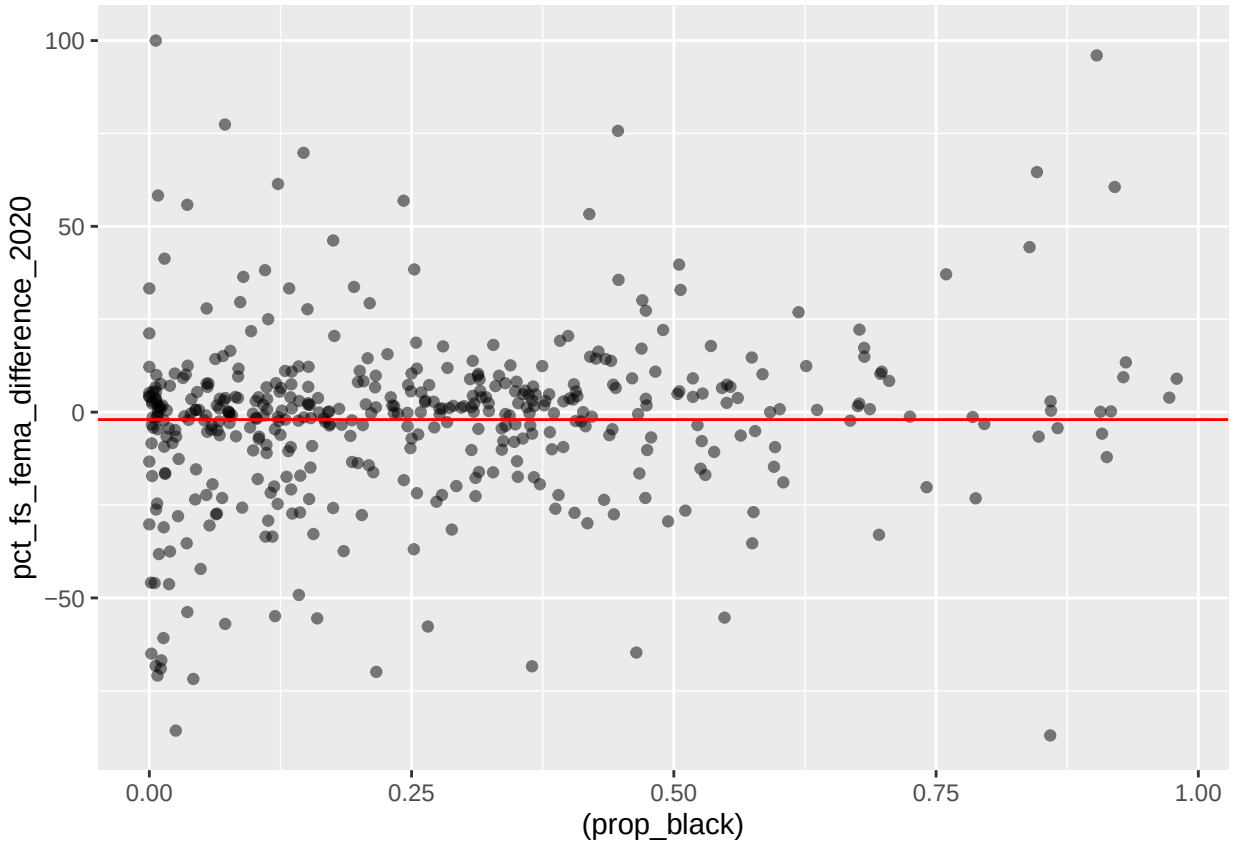
```
## Warning: Ignoring unknown aesthetics: show_guide
```


Average Risk score for each zipcode in Florida
Tallahassee
Jacksonville
Saint
Petersburg
Miami
lat
lon
30
28
26
24
−90
−87
−84
−81
−78
avg_risk_score_all
7.5
5.0
2.5
abs(pct_fs_fema_difference_2020)
0
25
50
75

In September of 2005, Hurricane Katrina, a Category 5 storm, destroyed New Orleans and was at the time the costliest tropical cyclone on record. There was much criticism of the government response, especially when people began to observe mass negligence and mismanagment, and were even more enraged when they believed it was fueled by race or class. One of the more memorable moments was when rapper, producer, fashion designer, and future presidential nominee Kanye West criticized the George Bush administration, calling them out because he thought "George Bush doesn't care about Black people".

Since we have the percent difference between FEMA projections and projections by First Street Foundations, we can use them to see if we can find a relationship between it and black/minority percentage in that zipcode.

```
LA_m <- filter(merged, state == 'LA')
yint <- mean(LA_m$pct_fs_fema_difference_2020)
merged %>%
  filter(state == 'LA') %>%
  ggplot(mapping = aes(x = (prop_black),
                       y = pct_fs_fema_difference_2020,
                       )) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = yint,color='red')
```



Fit a regression to this data

```
LA_fit <- stan_glm(data = LA_m, pct_fs_fema_difference_2020 ~ prop_black +
                     median_incomeE + prop_black:median_incomeE, refresh=0)
LA_fit
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      pct_fs_fema_difference_2020 ~ prop_black + median_incomeE + prop_black:median_incomeE
##  observations: 443
```

```
##  predictors:   4
## ------
##                            Median MAD_SD
## (Intercept)                -12.4    5.2
## prop_black                  18.4   13.0
## median_incomeE               0.0    0.0
## prop_black:median_incomeE    0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 22.9    0.8
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

We find the regression to be pct_difference = -12.4 + 18.6(prop_black). However the standard error is incredibly high and its 95% confidence interval includes 0.

Housing Changes

To get a picture of how the real estate market would change as a result of flooding, let's first try to understand the real estate market. Here, we'll be taking a look at both residential rent prices. Residential home prices are coming from the Housing and Urban Development's Small area fair market rents, which are used to give vouchers for people on section 8 housing. These should be relatively accurate in reflecting current fair market rents.

We would like to limit it to the state of florida since the way that rent prices are determined varies widly from region to region, much less state to state. Out of 1469 zip codes in Florida, we have residential rental data on 1431. Let's unpack the dataset and merge it with our existing.

```
#rent_price <- read_csv("/cloud/project/Final/Flood-Data-Analysis-master/data/fy2021-safmrs.csv")
rent_price <- read_csv("data/fy2021-safmrs.csv")

# rename columns
names(rent_price)[names(rent_price) == 'ZIP\nCode'] <- 'zip'
names(rent_price)[names(rent_price) == 'HUD Metro Fair Market Rent Area Name'] <- 'RegionName'
names(rent_price)[names(rent_price) == 'SAFMR\n2BR -\n110%\nPayment\nStandard'] <- 'BR2_rent'

# Merge things in
rent_price <- filter(rent_price, grepl("FL", RegionName))
merged_rent <- inner_join(rent_price, merged, by = c("zip" = "GEOID"))
merged_rent <- merged_rent %>%
  filter(!is.na(BR2_rent)) %>%
  transmute(zip, BR2_rent=parse_number(BR2_rent), total_pop, prop_black,
            count_property, median_incomeE, pct_fs_fema_difference_2020,
            avg_risk_score_all, state, city, longitude, latitude)
rent_mean <- mean(merged_rent$BR2_rent)
rent_sd <- sd(merged_rent$BR2_rent)
# Z score the rent, log median income and property count and total_population
merged_rent <- mutate(merged_rent, BR2_z = (BR2_rent - rent_mean) / rent_sd,
                      log_incomeE = log(median_incomeE),
                      log_property = log(count_property), log_pop=log(total_pop))

merged_rent <- drop_na(merged_rent)
merged_rent
```

```
## # A tibble: 934 x 16
##    zip   BR2_rent total_pop prop_black count_property median_incomeE
##    <chr>    <dbl>     <dbl>      <dbl>          <dbl>          <dbl>
##  1 33901     1177      8476     0.195            6275          35758
##  2 33903     1177     10490     0.0149           9622          39783
##  3 33904     1298     14295     0.0162          15874          51558
##  4 33905     1177     10974     0.0973          18724          47430
##  5 33907     1298     10367     0.0803           4960          42566
##  6 33908     1518     18920     0.0166          18308          56593
##  7 33909     1419      8113     0.0622          24264          54280
##  8 33912     1529      7253     0.0119           7257          73271
##  9 33913     1738      6486     0.0399          19260          80713
## 10 33914     1474     14455     0.0225          25000          64484
## # ... with 924 more rows, and 10 more variables:
## #   pct_fs_fema_difference_2020 <dbl>, avg_risk_score_all <dbl>, state <chr>,
## #   city <chr>, longitude <dbl>, latitude <dbl>, BR2_z <dbl>,
## #   log_incomeE <dbl>, log_property <dbl>, log_pop <dbl>
```

Fit a regression to predict rent change.

```
rent_fit <- stan_glm(data=merged_rent,
                     BR2_z ~ log_pop+prop_black+log_property+log_incomeE,
                     refresh=0)
rent_fit
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      BR2_z ~ log_pop + prop_black + log_property + log_incomeE
##  observations: 934
##  predictors:   5
## ------
##              Median MAD_SD
## (Intercept)  -22.8    0.9
## log_pop        0.5    0.0
## prop_black     0.7    0.2
## log_property  -0.5    0.0
## log_incomeE    2.1    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.7    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
coef(rent_fit)
```

```
##  (Intercept)      log_pop   prop_black log_property  log_incomeE
##  -22.8272244    0.5243480    0.6746736   -0.4972045    2.0778937
```

We try to compare different fits by using different predictors and interactors

```
rent_fit_a <- stan_glm(data=merged_rent, BR2_z ~ log_pop+prop_black+
                       log_property+log_incomeE, refresh=0)
rent_fit_a
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      BR2_z ~ log_pop + prop_black + log_property + log_incomeE
##  observations: 934
##  predictors:   5
## ------
##              Median MAD_SD
## (Intercept)  -22.8    0.8
## log_pop        0.5    0.0
## prop_black     0.7    0.2
## log_property  -0.5    0.0
## log_incomeE    2.1    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.7    0.0
##
## ------
```

```
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
rent_fit_b <- stan_glm(data=merged_rent, BR2_z ~ log_pop+prop_black+
                         log_property+log_incomeE+
                         log_incomeE:log_property, refresh=0)
rent_fit_b
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      BR2_z ~ log_pop + prop_black + log_property + log_incomeE + log_incomeE:log_property
##  observations: 934
##  predictors:   6
## ------
##                           Median MAD_SD
## (Intercept)               -7.8    5.4
## log_pop                    0.5    0.0
## prop_black                 0.7    0.2
## log_property              -2.2    0.6
## log_incomeE                0.7    0.5
## log_property:log_incomeE  0.2    0.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.7    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
rent_fit_c <- stan_glm(data=merged_rent, BR2_z ~ log_pop+prop_black+
                         log_property+log_incomeE+
                         log_property:log_pop, refresh=0)
rent_fit_c
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      BR2_z ~ log_pop + prop_black + log_property + log_incomeE + log_property:log_pop
##  observations: 934
##  predictors:   6
## ------
##                        Median MAD_SD
## (Intercept)           -23.8    1.3
## log_pop                 0.7    0.1
## prop_black              0.7    0.2
## log_property           -0.4    0.1
## log_incomeE             2.1    0.1
## log_pop:log_property    0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.7    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
```

```
## * For info on the priors used see ?prior_summary.stanreg
```

```
# we compare these with loo_cv
a_loo <- loo(rent_fit_a, k_threshold = 0.7)
```

```
## All pareto_k estimates below user-specified threshold of 0.7.
## Returning loo object.
```

```
b_loo <- loo(rent_fit_b, k_threshold = 0.7)
```

```
## All pareto_k estimates below user-specified threshold of 0.7.
## Returning loo object.
```

```
c_loo <- loo(rent_fit_c, k_threshold = 0.7)
```

```
## All pareto_k estimates below user-specified threshold of 0.7.
## Returning loo object.
```

```
loo_compare(a_loo, b_loo, c_loo)
```

```
##            elpd_diff se_diff
## rent_fit_b  0.0       0.0
## rent_fit_a -2.9       3.6
## rent_fit_c -3.8       4.5
```

The three regression models after going through leave-one-out cross validation we can see their predictive accuracy. We find that c_fit has the highest predictive accuracy when looking at the elpd_diff relative to the standard error when compared with a_fit and b_fit

```
rent_fit <- rent_fit_c
```

Simulate a disaster where we lose a certain number of properties

```
# simulate a disaster where we lose a certain number of properties
# take a normal random and multiplied against avg_risk_score which
# ranges from 0 to 10 and that percentage is lost.
post_rent <- mutate(.data=merged_rent,
                    pct_left = 1-((abs(rnorm(1, mean=0, sd=(0.1))))
                                  * avg_risk_score_all),
                    log_property = log(1+pct_left * count_property))
post_rent <- drop_na(post_rent)
# take 10 draws for each and take median
n_draws <- 10
preds <- posterior_predict(rent_fit, newdata=post_rent, draws=n_draws)
preds <- apply(preds, 2, median, na.rm=TRUE)
```

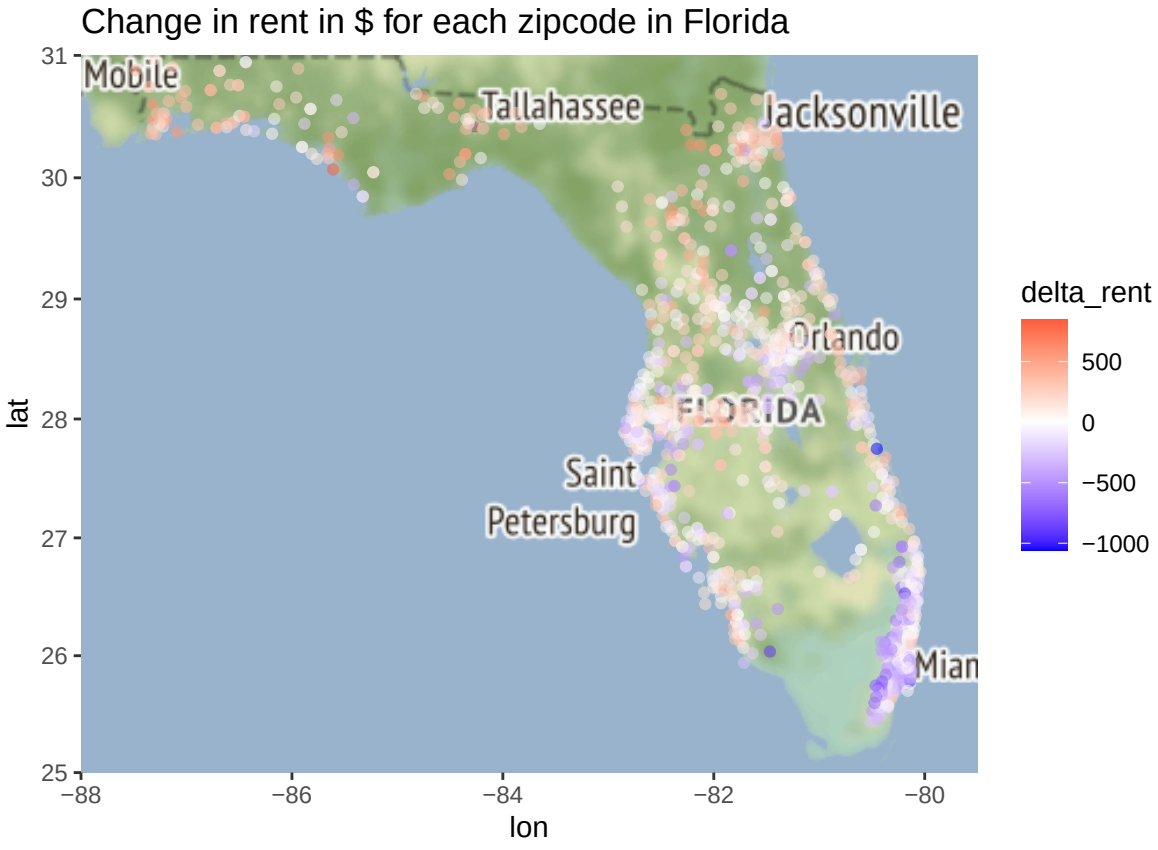
```
# subtract the original rent prices from predictions to find
# estimated nominal rent difference
merged_rent$delta_rent <- preds * rent_sd - merged_rent$BR2_z * rent_sd
merged_rent
```

```
## # A tibble: 934 x 17
##    zip   BR2_rent total_pop prop_black count_property median_incomeE
##    <chr>    <dbl>     <dbl>      <dbl>          <dbl>          <dbl>
##  1 33901     1177      8476     0.195           6275          35758
##  2 33903     1177     10490     0.0149          9622          39783
##  3 33904     1298     14295     0.0162         15874          51558
##  4 33905     1177     10974     0.0973         18724          47430
##  5 33907     1298     10367     0.0803          4960          42566
##  6 33908     1518     18920     0.0166         18308          56593
```

```
##  7 33909     1419      8113     0.0622          24264          54280
##  8 33912     1529      7253     0.0119           7257          73271
##  9 33913     1738      6486     0.0399          19260          80713
## 10 33914     1474     14455     0.0225          25000          64484
## # ... with 924 more rows, and 11 more variables:
## #   pct_fs_fema_difference_2020 <dbl>, avg_risk_score_all <dbl>, state <chr>,
## #   city <chr>, longitude <dbl>, latitude <dbl>, BR2_z <dbl>,
## #   log_incomeE <dbl>, log_property <dbl>, log_pop <dbl>, delta_rent <dbl>
```

```
fl_only_bbox <- c(-88, 25, -79.5, 31)
fl_only_map<-get_map(location=fl_only_bbox, zoom=6, maptype = 'terrain',
            source='osm',color='color')
# Overlay the risk scores to understand
ggmap(fl_only_map)+ geom_point(
        aes(x=longitude, y=latitude, show_guide = TRUE, colour=delta_rent),
        data=merged_rent, alpha=.5, na.rm = T)  +
        scale_color_gradient2(low="blue", mid="white", high="red") +
        labs(title='Change in rent in $ for each zipcode in Florida')
```

```
## Warning: Ignoring unknown aesthetics: show_guide
```



Demographic analysis

Here, we try to analyze whether there is a relationship between the demographics of a community and their predicted flood risk Set aside some of the data as test data to evaluate our models.

```
sims <- sample(nrow(merged), 20000)
train <- slice(merged, sims)
test <- slice(merged, -1*sims)
```

Fit a regression on percentage at risk in 2050 (within 100km) of the county using median age, total population, median income, proportion black, proportion non-white, and percentage at risk in 2020 as predictors. We log transform total_pop, count_property, and median_incomeE to account for their skewness.

```
# Fit #0
fit0 <- stan_glm(pct_fs_risk_2050_100 ~ prop_nonwhite + log(median_incomeE) + median_age + log(total_po
                 data = train,
                 refresh = 0)
summary(fit0, digits=2)
```

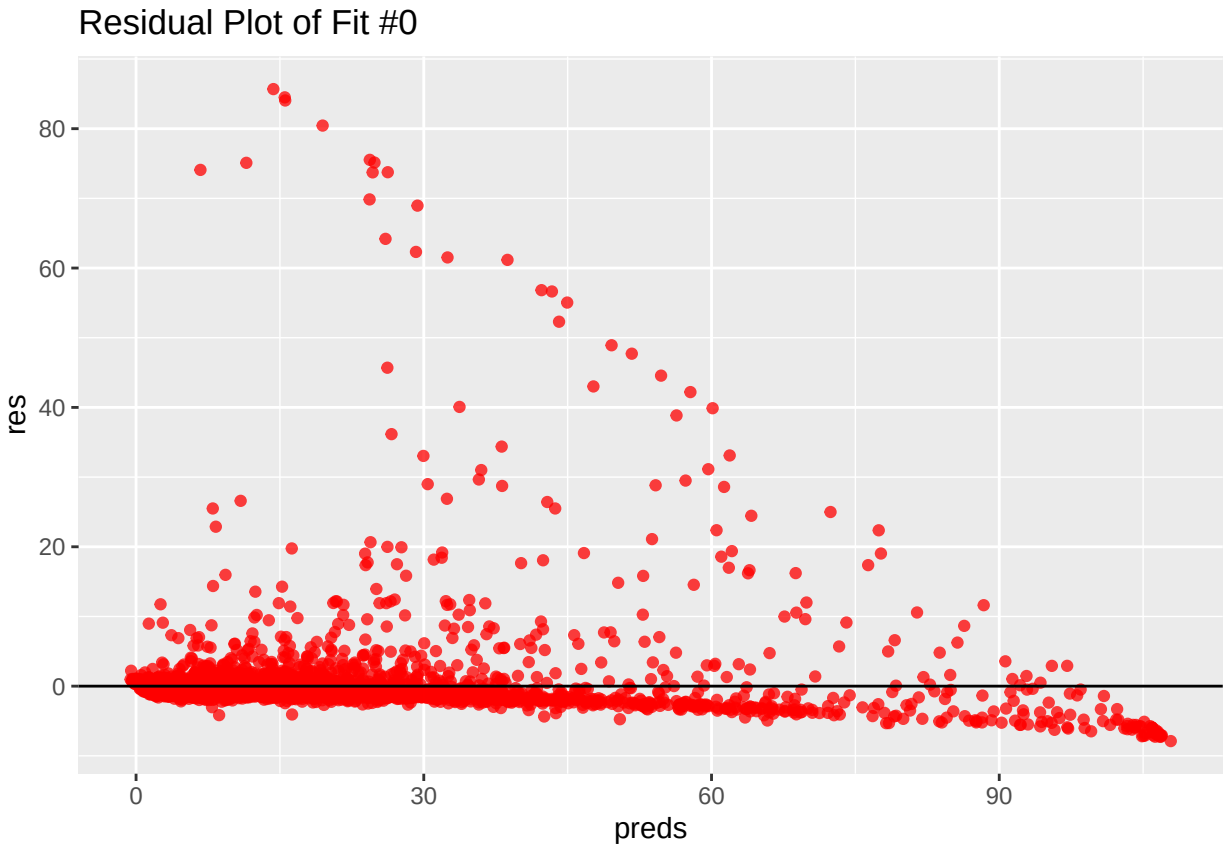
```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      pct_fs_risk_2050_100 ~ prop_nonwhite + log(median_incomeE) +
##     median_age + log(total_pop) + pct_fs_risk_2020_100
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 20000
##  predictors:   6
##
## Estimates:
##                        mean   sd    10%   50%   90%
## (Intercept)          -7.28   0.81 -8.31 -7.30 -6.24
## prop_nonwhite         1.79   0.16  1.58  1.80  2.00
## log(median_incomeE)   0.48   0.07  0.38  0.48  0.57
## median_age            0.01   0.00  0.01  0.01  0.02
## log(total_pop)        0.18   0.02  0.16  0.18  0.21
## pct_fs_risk_2020_100  1.07   0.00  1.07  1.07  1.07
## sigma                 3.64   0.02  3.62  3.64  3.67
##
## Fit Diagnostics:
##           mean   sd    10%   50%   90%
## mean_PPD 12.86   0.04 12.82 12.86 12.91
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                       mcse Rhat n_eff
## (Intercept)           0.01 1.00 5042
## prop_nonwhite         0.00 1.00 4663
## log(median_incomeE)   0.00 1.00 4797
## median_age            0.00 1.00 4948
## log(total_pop)        0.00 1.00 4791
## pct_fs_risk_2020_100 0.00 1.00 6144
## sigma                 0.00 1.00 1361
```

```
## mean_PPD             0.00 1.00 2059
## log-posterior        0.05 1.00 1374
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

As expected, the coefficient on the percentage of properties at risk in 2020 is close to 1 since it is basically an offset. We find that the coefficient on proportion nonwhite is 1.92 with a 95% confidence interval of (1.6, 2.24). Since this does not include 0, we can say that there is a positive correlation between a county's risk score and the proportion that is black. Specifically, when the other predictors are held constant, every percentage increase in black population leads to a 2.57% increase in 2050 flood risk.

Check residual plot for fit #0 using test data.

```
preds <- predict(fit0, newdata = test)
res <- test$pct_fs_risk_2050_100 - preds
test %>%
  ggplot(mapping = aes(x = preds,
                       y = res)) +
  geom_point(color = "red",
             alpha = 0.75) +
  geom_hline(yintercept = 0) +
  ggtitle("Residual Plot of Fit #0")
```



The fit is not very good. This is because a linear regression (though easy to interpret) is probably not the best fit here since the outcome variable is a proportion that is bounded between 0 and 1.

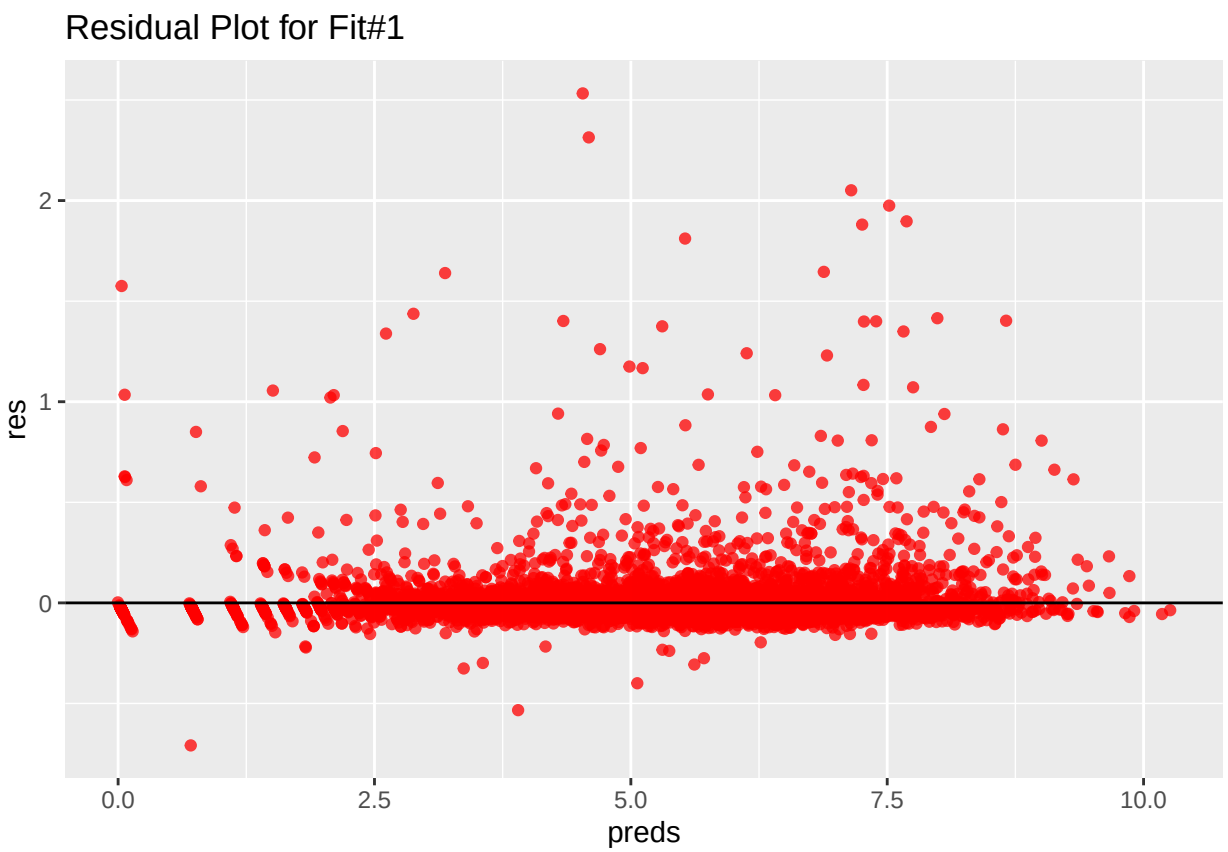
Instead, let's fit a linear regression that predicts the number of properties that will be at risk in 2050. The counts will be log transformed since they are skewed.

```r
# Fit #1
fit1 <- stan_glm(log(count_fs_risk_2050_100) ~ prop_nonwhite +
                   log(median_incomeE) + median_age + log(total_pop) +
                   log(count_fs_risk_2020_100),
                 data = train,
                 refresh = 0)
summary(fit1, digits=2)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      log(count_fs_risk_2050_100) ~ prop_nonwhite + log(median_incomeE) +
##     median_age + log(total_pop) + log(count_fs_risk_2020_100)
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 20000
##  predictors:   6
##
## Estimates:
##                               mean   sd    10%   50%   90%
## (Intercept)                  -0.23   0.03 -0.26 -0.23 -0.19
## prop_nonwhite                 0.09   0.01  0.08  0.09  0.09
## log(median_incomeE)           0.01   0.00  0.01  0.01  0.02
## median_age                    0.00   0.00  0.00  0.00  0.00
## log(total_pop)                0.01   0.00  0.01  0.01  0.02
## log(count_fs_risk_2020_100)   0.99   0.00  0.99  0.99  1.00
## sigma                         0.13   0.00  0.13  0.13  0.13
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 5.31   0.00 5.31  5.31  5.31
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                               mcse Rhat n_eff
## (Intercept)                   0.00 1.00 4843
## prop_nonwhite                 0.00 1.00 4170
## log(median_incomeE)           0.00 1.00 4693
## median_age                    0.00 1.00 4275
## log(total_pop)                0.00 1.00 3197
## log(count_fs_risk_2020_100)   0.00 1.00 3460
## sigma                         0.00 1.01  484
## mean_PPD                      0.00 1.00 2525
## log-posterior                 0.05 1.00 1312
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

As expected, the coefficient on the number of properties at risk in 2020 is 1 since it is basically an offset. The coefficient on prop_nonwhite is again noticeably high with a tight confidence interval. At 0.09, it tells us that when all else is equal, every percentage increase in nonwhite population causes the log of the number of properties at risk in 2050 to go up by 0.09. Check the residual plot for fit #1

```
preds <- predict(fit1, newdata = test)
res <- log(test$count_fs_risk_2050_100) - preds
test %>%
  ggplot(mapping = aes(x = preds,
                       y = res)) +
  geom_point(color = "red",
             alpha = 0.75) +
  geom_hline(yintercept = 0) +
  ggtitle("Residual Plot for Fit#1")
```



This residual plot is much better as the points seem to be spread more evenly.

Finally, since the number of houses at risk is a count, we can try to fit a poisson regression.

```
# Poisson regression
fit2 <- stan_glm(count_fs_risk_2050_100 ~ prop_nonwhite + log(median_incomeE) +
                   median_age + log(total_pop),
                 offset = log(count_fs_risk_2020_100),
                 data = train,
                 family = poisson,
                 refresh = 0)
summary(fit2, digits=4)
```

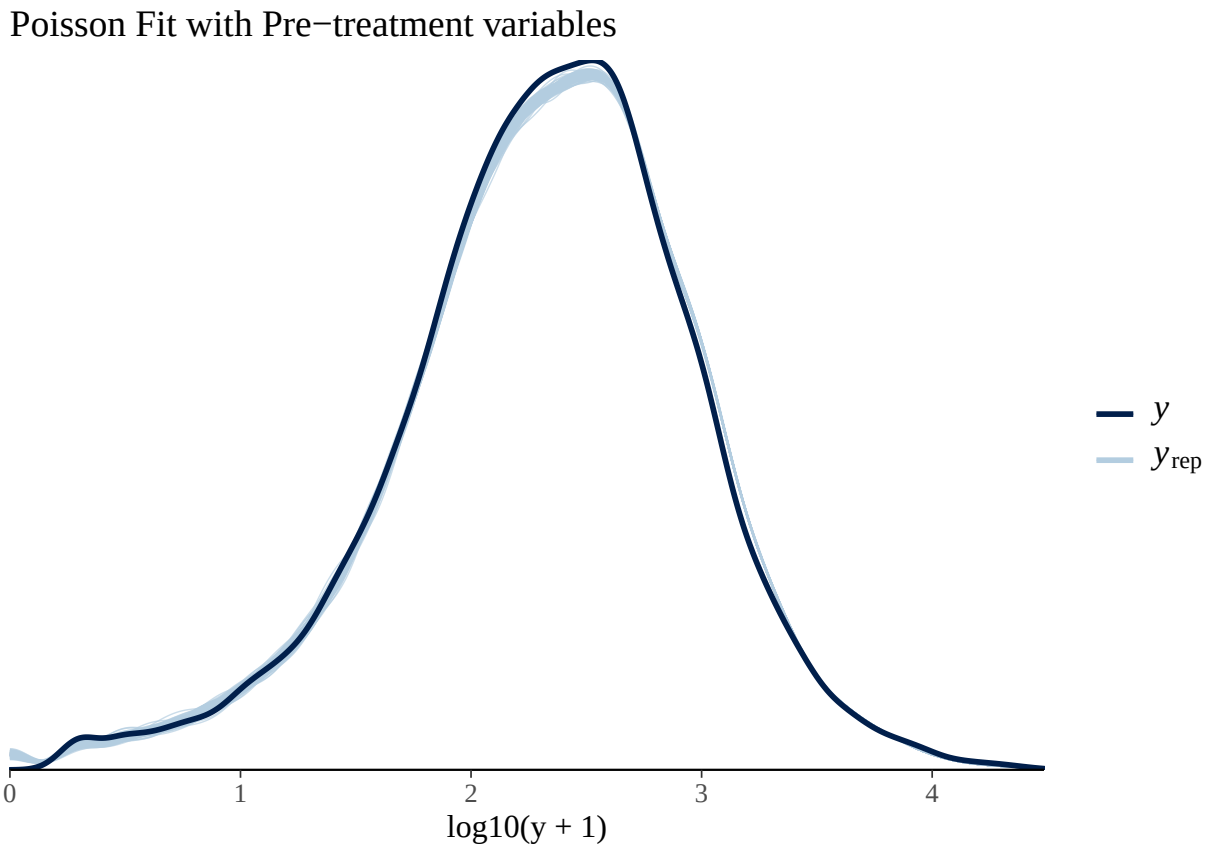
```
##
## Model Info:
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      count_fs_risk_2050_100 ~ prop_nonwhite + log(median_incomeE) +
##     median_age + log(total_pop)
```

```
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 20000
##  predictors:   5
##
## Estimates:
##                       mean    sd      10%     50%     90%
## (Intercept)         -0.4211  0.0099 -0.4338 -0.4209 -0.4084
## prop_nonwhite        0.1568  0.0019  0.1544  0.1568  0.1592
## log(median_incomeE)  0.0224  0.0009  0.0212  0.0224  0.0235
## median_age           0.0015  0.0001  0.0014  0.0015  0.0015
## log(total_pop)       0.0221  0.0003  0.0217  0.0221  0.0224
##
## Fit Diagnostics:
##            mean     sd       10%      50%      90%
## mean_PPD 542.2320   0.2310 541.9386 542.2350 542.5319
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                      mcse   Rhat   n_eff
## (Intercept)         0.0002 0.9995 4239
## prop_nonwhite       0.0000 0.9996 3635
## log(median_incomeE) 0.0000 0.9993 4122
## median_age          0.0000 0.9996 4070
## log(total_pop)      0.0000 1.0011 1233
## mean_PPD            0.0044 1.0007 2779
## log-posterior       0.0393 1.0032 1598
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

Again, we see prop_nonwhite having a very large impact with the highest coefficient of all the predictors. In addition, it has a very small standard error. Evaluate this fit by comparing the distribution of the model with the true distribution of the test data.

```
# Plot the fit
# Generate 1000 draws
yrep_4 <- posterior_predict(fit2, newdata = test,
                            offset = log(test$count_fs_risk_2020_100),
                            draws=1000)
n_sims <- nrow(yrep_4)

sims_display <- sample(n_sims, 100)
ppc_dens_overlay(log10(test$count_fs_risk_2050_100 + 1) , log10(yrep_4[sims_display, ] + 1)) +
  xlab('log10(y + 1)') +
  theme(axis.line.y = element_blank()) +
  ggtitle("Poisson Fit with Pre-treatment variables")
```



The model fits fairly well. Now, compare the 3 models using RMSE. All predictions will be converted so that they output the number of properties at risk in 2050.

```
# Have every model predict counts
fit0_count_preds <- predict(fit0, newdata = test) * test$count_property
fit1_count_preds <- exp(predict(fit1, newdata = test))
fit2_count_preds <- colMeans(yrep_4)
actual_counts <- test$count_fs_risk_2050_100

fit0_RMSE <- sqrt(mean((fit0_count_preds - actual_counts)^2))
fit1_RMSE <- sqrt(mean((fit1_count_preds - actual_counts)^2))
fit2_RMSE <- sqrt(mean((fit2_count_preds - actual_counts)^2))

paste("Fit #0 RMSE: ", fit0_RMSE)
```

```
## [1] "Fit #0 RMSE:  124241.71345232"
```

```
paste("Fit #1 RMSE: ", fit1_RMSE)
```

```
## [1] "Fit #1 RMSE:  406.710544965477"
```

```
paste("Fit #2 RMSE: ", fit2_RMSE)
```

```
## [1] "Fit #2 RMSE:  397.388095867277"
```

The Poisson model fits best, though the second linear regression comes very close. In both of these models, the coefficients signal that the nonwhite proportion of the population strongly correlates with the number of properties that will be at risk.

Conclusion