Code Dump

Rena Cohen

12/9/2020

```
# Here is the initial NYT dataset

masks <- read_csv("raw_data_masks/mask-use-by-county.csv") %>%

  # changing column names to lowercase

  janitor::clean_names() %>%
  mutate(countyfp = as.character(countyfp))

# Here is population data

pop <- read_csv("raw_data_masks/PopulationEstimates(1).csv") %>%

  # Relevant variables are county code, state, rural urban continuum, and population

  select("FIPStxt", "State", "Rural-urban_Continuum Code_2013",
         "POP_ESTIMATE_2019") %>%
  rename("countyfp" = "FIPStxt", "state" = "State", "ru_continuum" =
           "Rural-urban_Continuum Code_2013",
         "pop_2019" = "POP_ESTIMATE_2019") %>%
  mutate(countyfp = ifelse(nchar(countyfp) == 4, paste0("0", countyfp),
                           countyfp)) %>%
  mutate(countyfp = as.character(countyfp))



# Moving on to education data

education <- read_csv("raw_data_masks/Education.csv") %>%
  select("FIPS Code", "Percent of adults with less than a high school diploma, 2014-18",
         "Percent of adults with a high school diploma only, 2014-18",
         "Percent of adults completing some college or associate's degree, 2014-18",
         "Percent of adults with a bachelor's degree or higher, 2014-18") %>%
  rename("countyfp" = "FIPS Code",
         "pct_less_than_hs" = "Percent of adults with less than a high school diploma, 2014-18",
         "pct_hs" = "Percent of adults with a high school diploma only, 2014-18",
         "pct_some_college" = "Percent of adults completing some college or associate's degree, 2014-18
         "pct_college" = "Percent of adults with a bachelor's degree or higher, 2014-18")  %>%
  mutate(countyfp = as.character(countyfp)) %>%
  mutate(countyfp = ifelse(nchar(countyfp) == 4, paste0("0", countyfp), countyfp))
```

```r
# 2020 Presidential elections

voting <- read_csv("raw_data_masks/2020_US_County_Level_Presidential_Results.csv") %>%
  select(county_fips, county_name, per_gop, state_name) %>%
  rename("countyfp" = "county_fips", "percent_gop" = "per_gop") %>%
  mutate(countyfp = as.character(countyfp))

# Mask mandates

mask_mandates <- read_excel("raw_data_masks/earliestpolicy_08042020.xlsx") %>%
  select("1067", "...5") %>%
  rename("county" = "1067", "mandate" = "...5") %>%

  # adding leading 0's to county codes without them, since excel got rid of them

  mutate(countyfp = ifelse(nchar(county) == 4, paste0("0", county), county)) %>%

  # creating a dummy variable coded 1 if there is a county mandate

  mutate(county_mandate = ifelse(is.na(mandate), 0, 1)) %>%
  select(county_mandate, countyfp) %>%
  mutate(countyfp = as.character(countyfp))

# NYT Covid data

covid <- read_csv("raw_data_masks/us-counties.csv") %>%

# I'm going to select the row with the number of cases and deaths at the beginning of
# the NYT mask survey, which was July 2nd

  filter(date == "2020-07-02") %>%
  select(fips, cases, deaths, county) %>%
  rename("countyfp" = fips) %>%
  mutate(countyfp = as.character(countyfp))

# Poverty rate data

poverty <- read_csv("raw_data_masks/PovertyEstimates.csv") %>%
  select("FIPStxt", "PCTPOVALL_2018") %>%
  rename("countyfp" = "FIPStxt", "pct_poverty" = "PCTPOVALL_2018") %>%
  mutate(countyfp = as.character(countyfp)) %>%
  mutate(countyfp = ifelse(nchar(countyfp) == 4, paste0("0", countyfp), countyfp))

# Population raw data

older_people <-  read_csv("raw_data_masks/cc-est2019-agesex-02.csv") %>%
  janitor::clean_names() %>%

  # Feature engineering to get the FIPS code by combining state and county

  unite("countyfp", state:county, sep = "") %>%

  # We want only the most recent data (year = 12)
  filter(year == 12) %>%

  # Creating a variable that's the percent seniors in a county

  mutate(pct_senior = age65plus_tot/popestimate) %>%
  mutate(countyfp = as.character(countyfp))

total_pop <- read_csv("raw_data_masks/cc-est2019-alldata-02.csv") %>%
  janitor::clean_names() %>%

  # Feature engineering to get the FIPS code by combining state and county

  unite("countyfp", state:county, sep = "") %>%

  # We want only the most recent data (year = 12)

  filter(year == 12) %>%

  group_by(countyfp) %>%

  # Age group 0 represents characteristics of the population at large

  filter(agegrp == 0) %>%

# Making race and gender into percentages rather than raw counts
  mutate(pct_female = tot_female/tot_pop) %>%
  mutate(pct_black = (ba_male + ba_female)/tot_pop) %>%
  mutate(pct_native = (ia_male + ia_female)/tot_pop) %>%
  mutate(pct_asian = (aa_male + aa_female)/tot_pop) %>%
  mutate(pct_hispanic = (h_male + h_female)/tot_pop) %>%
  select(countyfp, pct_female, pct_black, tot_pop, pct_native, pct_asian, pct_hispanic) %>%
  mutate(countyfp = as.character(countyfp))

clean_data_2 <- readRDS("clean_data_2.rds")

# Merging in the NYT data from recent days
recent_covid <- read_csv("raw_data_masks/recent_covid.csv") %>%
  select(fips, cases, deaths, state, county) %>%
  rename(countyfp = fips)


# Note that this is Data from December 5th

clean_data_complete <- left_join(clean_data_2, recent_covid,
                                 by = "countyfp")  %>%

  # getting rid of variables that are no longer useful

  select(-c("always", "frequently", "sometimes", "rarely", "never", "cases_27",
            "deaths_27", "case_growth_1", "case_growth_2", "pct_less_than_hs",
            "pct_hs", "pct_trump_2016")) %>%
  # creating a county party indicator
  mutate(vote_dem = ifelse(pct_trump_2020 < 50, 1, 0)) %>%

  # creating a variable for some college

  mutate(pct_anycollege = pct_college + pct_some_college)%>%

  rename(cases_current = cases) %>%
  select(-c("pct_college", "pct_some_college")) %>%

  # Making case rates rather than raw numbers
  # For July, I took the average of the number of cases at the beginning of
  # the survey and at the end. I also multiplied these by 100,000 so that it
  # could be interpreted as number of cases per 100,000 people (this will make
  # it so that we no longer have negatives on our log scale later)

  mutate(case_rate_july = 100000*(cases_14+cases_02)/(2*pop_2019))%>%
  mutate(death_rate_july = 100000*(deaths_14+deaths_02)/(2*pop_2019)) %>%
  mutate(case_rate_december = 100000*cases_current/pop_2019) %>%
  mutate(death_rate_december = 100000*deaths/pop_2019) %>%

  select(-c("cases_14", "cases_02", "cases_current", "deaths_14",
          "deaths_02", "deaths")) %>%

  # Transforming that which needs transforming based on Chloe's suggestions

  mutate(log_pct_poverty = log(pct_poverty)) %>%
  mutate(log_pct_seniors = log(pct_seniors)) %>%
  mutate(log_pct_minority = log(pct_native + pct_black + pct_asian +
                                pct_hispanic)) %>%
  mutate(log_density = log(density)) %>%
  mutate(log_case_rate_july = log(case_rate_july + 1)) %>%
  mutate(log_death_rate_july = log(death_rate_july + 1)) %>%
  mutate(log_death_rate_december = log(death_rate_december + 1)) %>%
  mutate(log_case_rate_december = log(case_rate_december + 1)) %>%

  # deselecting variables we no longer need

  select(-c("pct_poverty", "pct_black", "pct_native", "pct_hispanic",
          "pct_asian", "pct_seniors", "case_rate_july", "death_rate_july",
          "case_rate_december", "death_rate_december"))

  # removing missing values: there aren't too many and I can investigate later,
# but I'm just going to remove any rows that have missing values for now.

sapply(clean_data_complete, function(x) sum(is.na(x)))

# Getting a dataset with just the incomplete observations... easier to
# edit this data in excel

write.csv(clean_data_complete, "clean_data_complete.csv")

# Adding back in with the NYC numbers
```

```r
clean_data_complete_ny <- read_csv("raw_data_masks/clean_data_complete.csv")

lm <- lm(pct_mask~log_case_rate_july, data = clean_data_complete)
summary(lm)
data_complete = na.omit(clean_data_complete_ny)

# Finally, splitting data into test and train to save as RDS

set.seed(139)

data_complete <- read_csv("raw_data_masks/clean_data_complete.csv")
nrow(data_complete)

# Carrying out the 80-20 split

ids <- sample(1:nrow(data_complete), round(0.8*nrow(data_complete)), replace = F)
data_train <- data_complete[ids,]
data_test <- data_complete[-ids,]


# Saving everything as an RDS

saveRDS(data_complete, "data_complete.RDS")
saveRDS(data_train, "data_train.RDS")
saveRDS(data_test, "data_test.RDS")
```

```r
# Removing NY and AK from the dataset of just rows with missing data
# values

clean_data_complete_ny <- read_csv("raw_data_masks/clean_data_complete.csv")

data_complete <- na.omit(clean_data_complete_ny)

just_na <- anti_join(clean_data_complete_ny, data_complete)

just_na <- just_na %>%
  filter(state != "NY" &
          state != "AK")

# Creating a datset with only rows without missing data

full = na.omit(data_complete)

# Conducting a series of two sample t-tests comparing characteristics of
# rows that were missing data with rows that were not

t.test(just_na$pct_mask, full$pct_mask)
t.test(just_na$density, full$density)
t.test(just_na$pct_female, full$pct_female)
t.test(just_na$pct_trump_2020, full$pct_trump_2020)
t.test(just_na$log_pct_seniors, full$log_pct_seniors)
t.test(just_na$pct_anycollege, full$pct_anycollege)
t.test(just_na$log_pct_minority, full$log_pct_minority)
t.test(just_na$log_pct_poverty, full$log_pct_poverty)
```

```
hist(clean_data_2$pct_mask,
     main = "Histogram of Mask-Wearing Score",
     xlab = "Mask-Wearing Score",
     ylab = "Number of Counties",
     col = "cornflowerblue")

hist(clean_data_2$density,
     main = "Histogram of Density (untransformed)",
     xlab = "Density",
     ylab = "Number of Counties",
     col = "cornflowerblue")

hist(log(clean_data_2$density),
     main = "Histogram of log(Density)",
     xlab = "log(Density)",
     ylab = "Number of Counties",
     col = "cornflowerblue")

plot(pct_mask ~ log(density), data = clean_data_2,
        main = "Mask Wearing by log(Density)",
        xlab = "log(Density)",
        ylab = "Mask-Wearing Score",
        col = "cornflowerblue")

plot(pct_mask ~ log_density, data = mask_train,
        main = "Mask Wearing by Density",
        xlab = "log(Density)",
        ylab = "Mask-Wearing Score")
points(7.291, 90.025, pch = 23, bg = "#EE9988", cex = 2)
points(5.64, 91.1, pch = 23, bg = "#77AADD", cex = 2)
legend("bottomright", legend = c("Chloe's County", "Rena's County"), pt.bg = c("#EE9988", "#77AADD"), p

plot(pct_mask ~ pct_trump_2020, data = mask_train,
        main = "Mask Wearing by Percent for Trump",
        xlab = "Percent of Votes for Trump",
        ylab = "Mask-Wearing Score")
points(35.97, 90.025, pch = 23, bg = "#EE9988", cex = 2)
points(35.64, 91.1, pch = 23, bg = "#77AADD", cex = 2)
legend("bottomleft", legend = c("Chloe's County", "Rena's County"), pt.bg = c("#EE9988", "#77AADD"), pc


plot(pct_mask ~ log_pct_seniors, data = mask_train,
        main = "Mask Wearing by % Seniors",
        xlab = "log(percent Seniors)",
        ylab = "Mask-Wearing Score")

plot(pct_mask ~ log_pct_minority, data = mask_train,
        main = "Mask Wearing by % Minority",
        xlab = "log(percent Minority)",
        ylab = "Mask-Wearing Score")

plot(pct_mask ~ log_pct_poverty, data = mask_train,
```

```r
        main = "Mask Wearing by % Poverty",
        xlab = "log(percent Poverty)",
        ylab = "Mask-Wearing Score")

plot(pct_mask ~ pct_anycollege, data = mask_train,
        main = "Mask Wearing by College",
        xlab = "% who attended college",
        ylab = "Mask-Wearing Score")

boxplot(pct_mask ~ ru_continuum, data = mask_train,
        main = "Mask Wearing by Rural-Urban Continuum Score",
        xlab = "RU Score",
        ylab = "Mask-Wearing Score")

boxplot(pct_mask ~ dem_governor, data = mask_train,
        main = "Mask Wearing by Democratic Governor",
        xlab = "Democratic Governor",
        ylab = "Mask-Wearing Score")

hist(clean_data_2$pct_trump_2016,
     main = "Histogram of Percent Voted for Trump 2016 (untransformed)",
     xlab = "Percent Trump 2016",
     ylab = "Number of Counties",
     col = "cornflowerblue")

hist(log(100-clean_data_2$pct_trump_2016),
     main = "Histogram of log(100-Trump2016)",
     xlab = "log(100-trump2016)",
     ylab = "Number of Counties",
     col = "cornflowerblue")

plot(pct_mask ~ log(100-clean_data_2$pct_trump_2016), data = clean_data_2,
        main = "Mask Wearing by log(100-Trump2016)",
        xlab = "log(100-trump2016)",
        ylab = "Mask-Wearing Score",
        col = "cornflowerblue")
```

```r
# Fitting a model with just the intercept

interceptmodel = lm(pct_mask ~ 1, data = mask_train)

# Fitting a model with all main effects

fullmodel = lm(pct_mask ~ pop_2019 + log_pct_seniors +
                 ru_continuum + pct_trump_2020 +
                 log_pct_minority + dem_governor + pct_anycollege +
                 log_density + pct_female + log_pct_poverty,
               data = mask_train)

# Fitting a model with main effects and interaction terms

interactionmodel = lm(pct_mask ~ (pop_2019 + log_pct_seniors +
                                  ru_continuum + pct_trump_2020 +
                                  log_pct_minority + dem_governor +
                                    pct_anycollege + log_density + pct_female + log_pct_poverty)^2,
               data = mask_train)

# Backwards forwards step from main effects to interaction terms

selectedmodel = step(fullmodel, scope = list(lower = formula(interceptmodel),
                                             upper = formula(interactionmodel)),
     direction = "both", trace = 0)

# Checking assumptions

plot(fullmodel, which = 1)
plot(fullmodel, which = 2)

# Fitting a random forest model

rf1 = randomForest(pct_mask ~ pop_2019 + log_pct_seniors + ru_continuum + pct_trump_2020 + log_pct_mino
               data = mask_clean, mtry = 6, maxnodes=10)

# Creating variance important part

varImpPlot(rf1, main = "Random Forest Variable Importance")

dt2= rpart(pct_mask ~ pct_trump_2020 + dem_governor,
               data = mask_clean, control=list(maxdepth = 10))

# Creating another random forest model with apolitical predictors

rf3 = randomForest(pct_mask ~ pop_2019 + log_pct_seniors + ru_continuum +
                     log_pct_minority + pct_anycollege + log_density + pct_female + log_pct_poverty,
               data = mask_clean, mtry = 6, maxnodes=10)

# Building an RMSE function

RMSE = function(y,yhat){
  SSE = sum((y-yhat)^2)
  return(sqrt(SSE/length(y)))
}

# Calculating RMSE on test and train sets for all models fit

mask_test = readRDS("data_test.rds")
test_clean = na.omit(mask_test)
train_clean = na.omit(mask_train)

rf1_train = round(RMSE(train_clean$pct_mask, predict(rf1, new=train_clean)), 3)
rf1_test = round(RMSE(test_clean$pct_mask, predict(rf1, new=test_clean)), 3)

dt2_train = round(RMSE(train_clean$pct_mask, predict(dt2, new=train_clean)), 3)
dt2_test = round(RMSE(test_clean$pct_mask, predict(dt2, new=test_clean)), 3)

rf3_train = round(RMSE(train_clean$pct_mask, predict(rf3, new=train_clean)), 3)
rf3_test = round(RMSE(test_clean$pct_mask, predict(rf3, new=test_clean)), 3)
```

```r
political_train = round(RMSE(train_clean$pct_mask,
                             predict(political, new=train_clean)), 3)
political_test = round(RMSE(test_clean$pct_mask,
                            predict(political, new=test_clean)), 3)

apolitical_train = round(RMSE(train_clean$pct_mask,
                              predict(apolitical, new=train_clean)), 3)
apolitical_test = round(RMSE(test_clean$pct_mask,
                             predict(apolitical, new=test_clean)), 3)

fullmodel_train = round(RMSE(train_clean$pct_mask,
                             predict(fullmodel, new=train_clean)), 3)
fullmodel_test = round(RMSE(test_clean$pct_mask,
                            predict(fullmodel, new=test_clean)), 3)

interactionmodel_train = round(RMSE(train_clean$pct_mask,
                                    predict(interactionmodel, new=train_clean)), 3)
interactionmodel_test = round(RMSE(test_clean$pct_mask,
                                   predict(interactionmodel, new=test_clean)), 3)

selectedmodel_train = round(RMSE(train_clean$pct_mask, predict(selectedmodel, new=train_clean)), 3)
selectedmodel_test = round(RMSE(test_clean$pct_mask, predict(selectedmodel, new=test_clean)), 3)
```

```r
# Running logistic regression

data_train = readRDS("data_train.RDS")
logit_1 = glm(county_mandate~pct_trump_2020, data = data_train,
              family = "binomial")
dummy.pct = seq(0,100,1)
yhat = predict(logit_1, new = data.frame(pct_trump_2020 = dummy.pct))
phat = exp(yhat)/(1+exp(yhat))


# Making a plot

plot(county_mandate~pct_trump_2020, data = data_train, cex = 0.5, pch = 16, col = rgb(0.2,0.2,0.2,0.1),
     xlab = "Percent Trump Votes, 2020",
     main = "Mandate Odds vs. Partisanship")
lines(phat~dummy.pct, col = "dodgerblue", lwd = 2)

# Limiting to rows we have data on for mask mandates

has_man <- data_train %>%
  na.omit(county_mandate)

# Fitting quadratic model

lm_trump_quad <- lm(pct_mask~pct_trump_2020 + I(pct_trump_2020^2),
                    data = has_man)

# Fitting quadratic model with county mandate

lm_trump_quad_mandate <- lm(pct_mask~pct_trump_2020 + I(pct_trump_2020^2)
                             + county_mandate, data = has_man)

# Mandate term is significant

anova(lm_trump_quad, lm_trump_quad_mandate)

# Adding an interaction term

lm_trump_quad_mandate_interact <- lm(pct_mask~pct_trump_2020 +
                                       I(pct_trump_2020^2) + county_mandate +
                       pct_trump_2020*county_mandate +
                       I(pct_trump_2020^2)*county_mandate,
           data = has_man)

# Interaction term is not significant

anova(lm_trump_quad_mandate, lm_trump_quad_mandate_interact)

# Adding a term for state mandates

lm_trump_state <- lm(pct_mask~ pct_trump_2020 + I(pct_trump_2020^2) + state_mandate +  county_mandate, 
summary(lm_trump_state)

# State coefficient is significant

anova(lm_trump_quad_mandate, lm_trump_state)

# Adding an interaction between state and county

lm_trump_state_interact <- lm(pct_mask~ pct_trump_2020 +
                                I(pct_trump_2020^2) + state_mandate +
             county_mandate + state_mandate:county_mandate,
           data = has_man)


# Interaction term is significant

anova(lm_trump_state, lm_trump_state_interact)

# Checking assumptions for mandate models

plot(lm_trump_state_interact)

# Here is the bootstrapped confidence interval for the added bonus of county

nsims = 1000
difs = rep(NA, nsims)
for(i in 1:nsims){
  # sample some data
  boot = data_train[sample(1:nrow(has_man), size = nrow(has_man),
                                                         replace = T),]
  # fit the model and recalculate betas
  lm.boot = lm(formula(lm_trump_state_interact), data = boot)
```

```r
  # pull off the two relevant coefficients

  difs[i] = coef(lm.boot)[5]+coef(lm.boot)[6]


}

ci.boot <- quantile(difs, c(0.025, 0.975))
ci.boot
```

```r
# Making a scatterplot between pct_mask nad log_case_Rate_december

ggplot(data_train, aes(x = pct_mask, y = log_case_rate_december)) +
         geom_point(na.rm = T, alpha = 0.5) +
         labs(x = "Percent Mask Wearing in July",
              y = "Log Case Rate, December",
              title = "Predicting Case Rates in December from
              from Mask Wearing in July") +
  theme_bw()

# Making side by side boxplots to compare densities

# Showing shift from rural to urban counties
ggplot(data = data_train, aes(x = as.factor(ru_continuum),
                              y = log_case_rate_december)) +
  geom_boxplot(na.rm = T) +
  labs(title = "Case Distribution by Community Type, December",
       x = "Rural Urban Continuum Score \n(1 = most urban)",
       y = "Log Case Rate in December") +
  theme_bw()

ggplot(data = data_train, aes(x = as.factor(ru_continuum),
                              y = log_case_rate_july)) +
  geom_boxplot(na.rm = T) +
  labs(title = "Case Distribution by Community Type, July",
       x = "Rural Urban Continuum Score \n(1 = most urban)",
       y = "Log Case Rate in July")  +
  theme_bw()

# Calculating the various mixed effects models
lmer_state <- lmer(log_case_rate_december~
                     (1|state),
                   data = data_train, REML = F)
summary(lmer_state)

# This model was bad... didn't add anything

lmer_state_density <- lmer(log_case_rate_december~
                     (1|state) + log_density,
                   data = data_train, REML = F)

# Adding in a fixed effect of state
```

```r
lmer_state_mask <- lmer(log_case_rate_december~pct_mask + (1|state),
                        data = data_train, REML = F)
summary(lmer_state_mask)

# Conducing a chi-squared likelihoood ratio test

anova(lmer_state_mask, lmer_state)

# Assumption Checking for final mixed effects

# Checking linearity and constant variance
plot(residuals(lmer_state_mask)~predict(lmer_state_mask),
     main = "Residuals vs. Fitted, lmer_state_mask",
     xlab = "Fitted Values for Log_case_rate_December",
     ylab = "Residuals")

# checking normality of residuals
qqnorm(residuals(lmer_state_mask),
       main = "Q-Q Plot for Residuals")
qqline(residuals(lmer_state_mask))

# Checking normality of random intercepts
qqnorm(coef(lmer_state_mask)$state[['(Intercept)']],
       main = "Q-Q Plot for Random Intercepts")
qqline(coef(lmer_state_mask)$state[['(Intercept)']])
hist(coef(lmer_state_mask)$state[['(Intercept)']],
     main = "Histogram of Random Intercepts",
     xlab = "Random Intercept for State")

# Calculating RMSE for the two mixed effects models on the test set

RMSE = function(model, newdata, y){
  yhat = predict(model, newdata = newdata)
  RMSE = sqrt(sum((y-yhat)^2/nrow(newdata)))
  return(RMSE)
}

# The addition of the fixed effect does improve things slightly

data_test <- readRDS("~/Desktop/STAT 139/stat139_project/data_test.RDS")
RMSE(lmer_state_mask, data_test, data_test$log_case_rate_december)
RMSE(lmer_state, data_test, data_test$log_case_rate_december)
```