DTSC 5301 PROJECT

9/2/2021

Question of Interest

We want to know whether or not the stimulus checks sent out by the US government have had a positive impact on the economy (using consumer spending as a proxy for how healthy the economy is).

Why Should YOU Care How Healthy the US Economy Is?

Why should we care about consumer spending? It seems like a big picture idea that won't really effect any of us specifically, right? Wrong. When the economy is healthy, there are more jobs available. Those jobs also pay more. As graduate students, we all want to get good paying jobs as Data Scientists, and that will be more likely to happen more quickly if the economy is healthy.

This analysis will help us understand how stimulus checks effect the US economy, and therefore, indirectly, the analysis will also help us understand whether or not stimulus checks will help us get good-paying jobs quickly after graduation.

Data Source

All of our data was aggregated by Opportunity Insights at https://github.com/OpportunityInsights/EconomicTracker. In this analysis, we use spending data provided by Affinity Solutions, job postings data from Burning Glass Technologies, COVID data from the CDC, GPS mobility reports from Google, unemployment claims from the Department of Labor, and employment levels from Paychex, Intuit, Earnin and Kronos.

Primary Reference:

"The Economic Impacts of COVID-19: Evidence from a New Public Database Built Using Private Sector Data", by Raj Chetty, John Friedman, Nathaniel Hendren, Michael Stepner, and the Opportunity Insights Team. November 2020. Available at: https://opportunityinsights.org/wp-content/uploads/2020/05/tracker_paper.pdf

Read in Data from GitHub Repository

```
covid_daily_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/main/c

move_daily_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/main/da

affinity_daily_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/mai
```

```r
job_listings_weekly_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTrack

employment_daily_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/

ui_claims_weekly_df <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/

state_id <- read_csv("https://raw.githubusercontent.com/OpportunityInsights/EconomicTracker/main/data/G

#stimulus <- read_csv("")
```

Join the datasets we're interested in into one dataset.

Here we join the datasets of interest based on a shared date and state of measurements. We are joining weekly datasets (job listings and unemployment insurance claims) with the daily datasets. This will leave a bunch of **NA** values. We'll come back to fix that later.

```r
df <- left_join(affinity_daily_df, move_daily_df, by = c("year", "month", "day", "statefips"))

df <- left_join(df, covid_daily_df, by = c("year", "month", "day", "statefips"))

df <- left_join(df, employment_daily_df, by = c("year", "month", "day", "statefips"))

df <- full_join(df, job_listings_weekly_df, by = c("year", "month", "day" = "day_endofweek", "statefips

df <- full_join(df, ui_claims_weekly_df, by = c("year", "month", "day" = "day_endofweek", "statefips"))

df <- left_join(df, state_id, by = c("statefips"))
```

Combine "month", "day", and "year" columns into a "date" column

```r
# https://tidyr.tidyverse.org/reference/unite.html
df <- df %>% unite("date", day:month:year, remove = FALSE, sep = "-")
```

```
## Warning in x:y: numerical expression has 2 elements: only the first used
```

```r
# https://lubridate.tidyverse.org/reference/ymd.html
df$date <- dmy(df$date)

df <- df %>% mutate(week = week(date))
```

Data Selection and Cleaning

In this code chunk, we select the columns that we're interested in from the dataframe that we joined a couple of steps ago. We could use all of the data in the dataframe, but we choose not to, since not all of the features will be helpful in answering the question of whether or not the stimulus checks have boosted the economy.

```
df <- df %>%
  select(date, year, month, day, week, statename, stateabbrev, state_pop2019, initclaims_rate_regular, c
  mutate(
    spend_all = as.double(spend_all),
    gps_parks = as.double(gps_parks),
    new_case_count = as.double(new_case_count),
    new_death_count = as.double(new_death_count),
    case_count = as.double(case_count),
    death_count = as.double(death_count),
    gps_transit_stations = as.double(gps_transit_stations),
    emp = as.double(emp),
    contclaims_rate_combined = as.double(contclaims_rate_combined)
  )


glimpse(df)
```

```
## Rows: 31,110
## Columns: 24
## $ date                     <date> 2020-01-01, 2020-01-01, 2020-01-01, 2020-01~
## $ year                     <dbl> 2020, 2020, 2020, 2020, 2020, 2020, 2020, 20~
## $ month                    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ day                      <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ week                     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ statename                <chr> "Alabama", "Alaska", "Arizona", "Arkansas", ~
## $ stateabbrev              <chr> "AL", "AK", "AZ", "AR", "CA", "CO", "CT", "D~
## $ state_pop2019            <dbl> 4903185, 731545, 7278717, 3017804, 39512223,~
## $ initclaims_rate_regular  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ contclaims_rate_combined <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ bg_posts                 <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ emp                      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ spend_all                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_retail_and_recreation <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_grocery_and_pharmacy <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_parks                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_transit_stations     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_workplaces           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_residential          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ gps_away_from_home       <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ new_case_count           <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ new_death_count          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ case_count               <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ death_count              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

To fix the **NA** values found in our dataset, we replace them all with **0**. We could have handled them many other ways, like replacing them with the column mean, median, mode, or by training a regression model to fill them based on the rows that were not missing those values. For this dataset, though, we found that **NA**s are frequently used when the there was no interesting data to report (in other words, the value for the feature was zero). This can be seen particularly in columns like new_death_count and new_case_count from the CDC COVID dataset.

We also drop rows that have 0 spending data because this value is not realistic. Also, we need the spend_all column to be clean because we will be using it for plotting and training a regression model later on.

```
# https://stackoverflow.com/questions/45576805/how-to-replace-all-na-in-a-dataframe-using-tidyrreplace-

#length(df$date)

#colSums(is.na(df))

df <- df %>% replace(is.na(.), 0)
df <- df %>% filter(spend_all != 0)
```

Combing Weekly and Daily Data

In this code chunk, we aggregate daily data into weekly data. For most columns, the appropriate aggregate function is **mean()**, with the exception of the *new_case_count*, *new_death_count*, *case_count*, *death_count*, and *date columns*.

We also add in additional data about the states, at this point (like state name, abbreviation, and population).

```
df_weekly <- df %>%
  group_by(year, week, stateabbrev) %>%
  summarize(spend_all = mean(spend_all), contclaims_rate_combined = mean(contclaims_rate_combined), bg_
```

```
## `summarise()` has grouped output by 'year', 'week'. You can override using the `.groups` argument.
```

```
df_weekly <- left_join(df_weekly, state_id, by = c("stateabbrev"))
```

Adding Stimulus Check Data

Here we add in the data for the COVID stimulus checks. We do this by creating a feature encoding for each check, where the value for that check is **0** before the check is sent out, and **1** after the check is sent out. This process created three new features, which we creatively named *first_check*, *second_check*, and *third_check*

```
df_weekly <- df_weekly %>% mutate(first_check = (if (date < ymd("2020-04-15")) 0
                                                 else 1),
                                  second_check = (if (date < ymd("2021-01-04")) 0
                                                  else 1),
                                  third_check = (if (date < ymd("2021-03-18")) 0
                                                 else 1))
```

Double Check that Data is Clean

We already took care of **NA** values a few steps ago. Now we need to ensure that there are no infinite values, as well.

```
sum(sapply(df_weekly, is.infinite))
```

```
## [1] 0
```

As you can see from the results above, the number of infinite values in our dataset currently is 0.

Training a Linear Model on Our Data

In this code chunk, we fit a linear model to the *gps_retail_and_recreation*, *emp*, *first_check*, *second_check*, and *third_check* features, with the goal of predicting the *spend_all* variable. The working assumption here is that *spend_all* is a dependent variable, and the others are independent variables.

```
lm <- lm(spend_all ~ gps_retail_and_recreation + emp + first_check +
           second_check + third_check, df_weekly)
```

```
summary(lm)
```

```
##
## Call:
## lm(formula = spend_all ~ gps_retail_and_recreation + emp + first_check +
##     second_check + third_check, data = df_weekly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34163 -0.04494  0.00693  0.04757  0.41405
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -0.041870   0.003174 -13.190  < 2e-16 ***
## gps_retail_and_recreation  0.308678   0.012020  25.681  < 2e-16 ***
## emp                        0.185988   0.025534   7.284 3.81e-13 ***
## first_check                0.044965   0.003733  12.045  < 2e-16 ***
## second_check               0.133771   0.004405  30.367  < 2e-16 ***
## third_check                0.013451   0.004700   2.862  0.00423 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08172 on 4482 degrees of freedom
## Multiple R-squared:  0.6325, Adjusted R-squared:  0.6321
## F-statistic:  1543 on 5 and 4482 DF,  p-value: < 2.2e-16
```

Summarize the Model

Based on the information displayed above, all of the variables we are regressing on are statistically significant for predicting the overall spending.

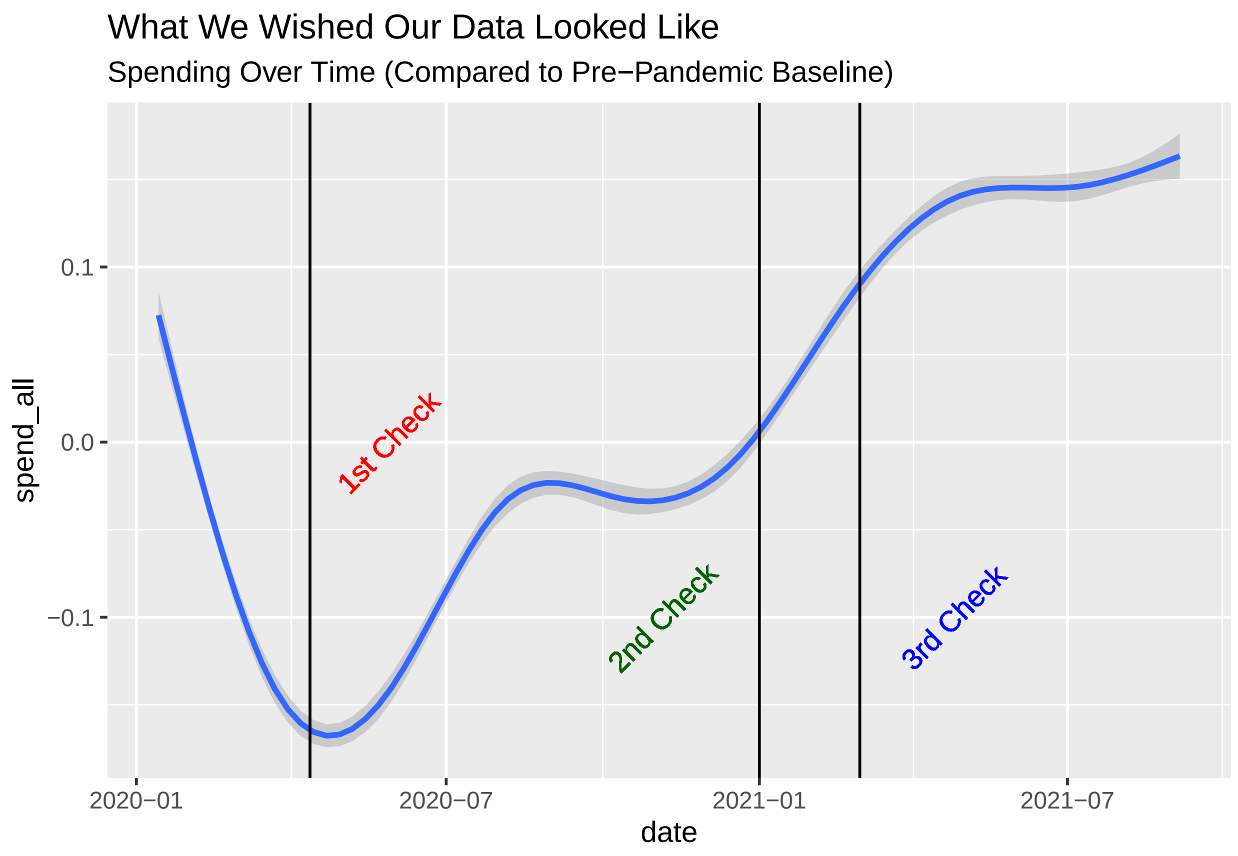
Plotting spending over time for all states and categories

The dates for the stimulus checks were approximated from this article.

```
# https://stackoverflow.com/questions/38815996/r-adding-geom-vline-labels-to-geom-histogram-labels

ggplot(df_weekly, aes(x = date, y = spend_all)) +
  geom_smooth() +
  geom_vline(xintercept = as.Date("2020-04-12")) +
  geom_vline(xintercept = as.Date("2021-01-01")) +
  geom_vline(xintercept = as.Date("2021-03-01")) +
  geom_text(aes(x = as.Date("2020-05-28"), label = "1st Check"), color = "red", angle = 45, y = 0) +
  geom_text(aes(x = as.Date("2020-11-05"), label = "2nd Check"), color = "dark green", angle = 45, y =
  geom_text(aes(x = as.Date("2021-04-25"), label = "3rd Check"), color = "blue", angle = 45, y = -.1) +
  labs(title = "What We Wished Our Data Looked Like", subtitle = "Spending Over Time (Compared to Pre-Pa
```

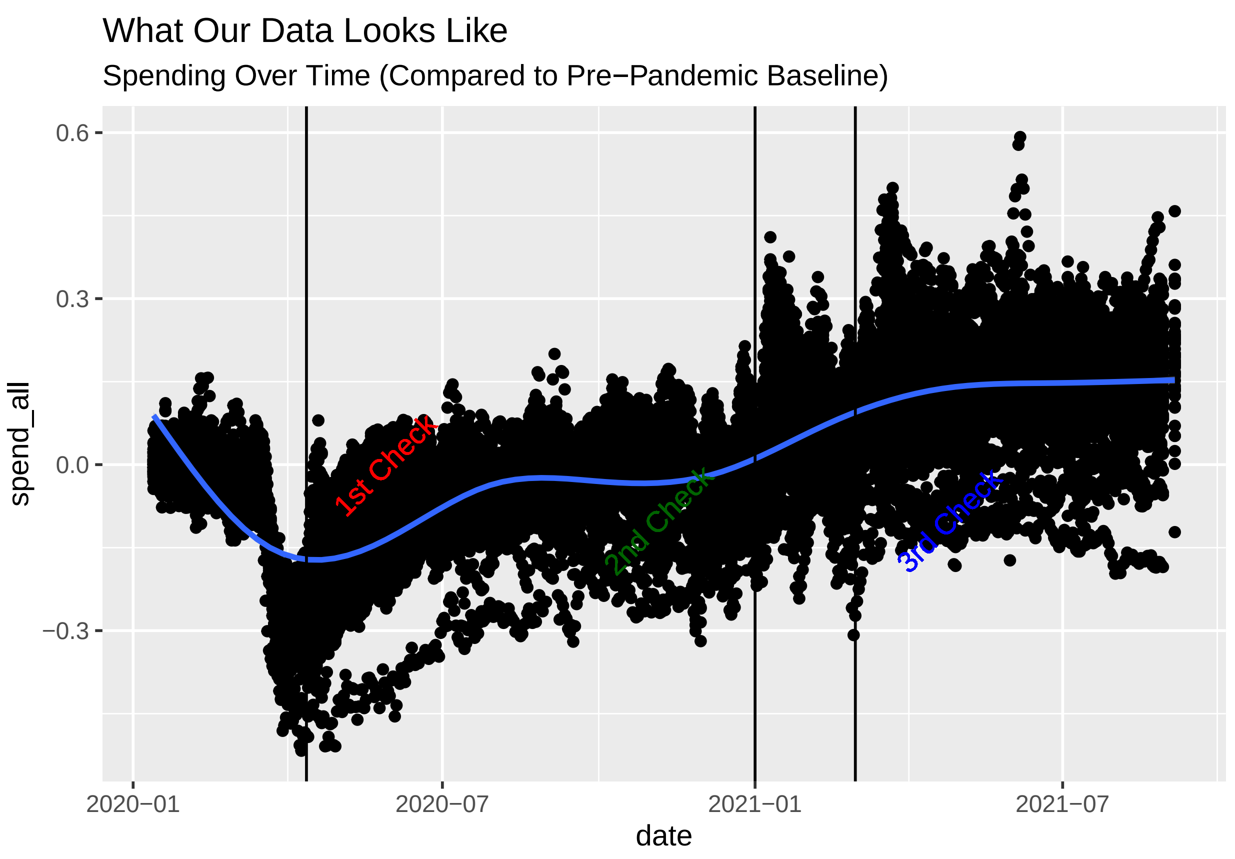
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



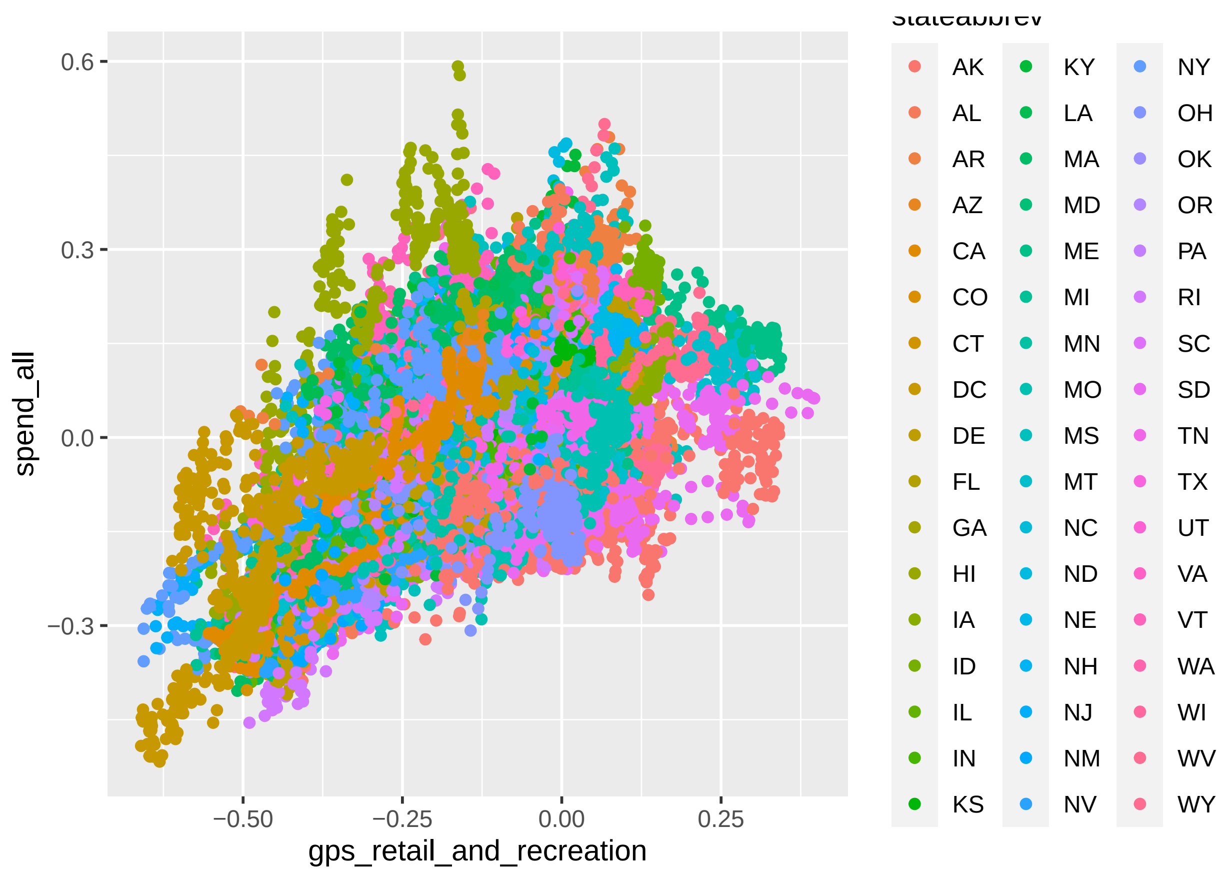
```
# https://stackoverflow.com/questions/38815996/r-adding-geom-vline-labels-to-geom-histogram-labels

ggplot(df, aes(x = date, y = spend_all)) +
  geom_point() +
  geom_smooth() +
  geom_vline(xintercept = as.Date("2020-04-12")) +
  geom_vline(xintercept = as.Date("2021-01-01")) +
  geom_vline(xintercept = as.Date("2021-03-01")) +
  geom_text(aes(x = as.Date("2020-05-28"), label = "1st Check"), color = "red", angle = 45, y = 0) +
  geom_text(aes(x = as.Date("2020-11-05"), label = "2nd Check"), color = "dark green", angle = 45, y =
  geom_text(aes(x = as.Date("2021-04-25"), label = "3rd Check"), color = "blue", angle = 45, y = -.1) +
  labs(title = "What Our Data Looks Like", subtitle = "Spending Over Time (Compared to Pre-Pandemic Bas
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```r
ggplot(df, aes(x = gps_retail_and_recreation, y = spend_all, color = stateabbrev)) +
  geom_point()
```


stateabbrev
AK
AL
AR
AZ
CA
CO
CT
DC
DE
FL
GA
HI
IA
ID
IL
IN
KS
KY
LA
MA
MD
ME
MI
MN
MO
MS
MT
NC
ND
NE
NH
NJ
NM
NV
NY
OH
OK
OR
PA
RI
SC
SD
TN
TX
UT
VA
VT
WA
WI
WV
WY
spend_all
0.6
0.3
0.0
−0.3
−0.50
−0.25
0.00
0.25
gps_retail_and_recreation