QuidelCovidCorrelationAnalysis

August 21, 2020

[1]:
```
import os
from os.path import join
import matplotlib.pyplot as plt
from datetime import date, timedelta, datetime, date

import numpy as np
import pandas as pd
from scipy.stats import spearmanr

import covidcast
```

[2]:
```
# Constants
lag = 7
```

[9]:
```
# Combine output csv files for test_per_device
read_dir = "../covidcast-indicators/quidel_covidtest/receiving"
csv_df_all = pd.DataFrame(columns = ["time_value", "val", "geo_id", "geo_res",␣
 ↪"smoother"])

for fn in os.listdir(read_dir):
    if "test_per_device" not in fn:
        continue
    timestamp, geo_res, _, _, smoother, _, _, _ = fn.split("_")
    tempt_df = pd.read_csv(read_dir + '/' + fn, sep = ",").drop(["se",␣
 ↪"sample_size"], axis = 1)
    tempt_df["time_value"] = datetime.strptime(timestamp,'%Y%m%d')
    tempt_df["geo_res"] = geo_res
    tempt_df["smoother"] = smoother
    csv_df_all = csv_df_all.append(tempt_df, sort = True)
csv_df_all = csv_df_all.sort_values("time_value")
```

[10]:
```
csv_df_all.head()
```

[10]:

	geo_id	geo_res	smoother	time_value	val
24	39300	msa	smoothed	2020-05-26	4.714286
57	40137	county	smoothed	2020-05-26	4.040404

```
58  40139  county  smoothed 2020-05-26  16.285714
59  40153  county  smoothed 2020-05-26   3.555389
61  48005  county  smoothed 2020-05-26   3.677474
```

[11]:
```
def get_api_df(geo_res, start_date, end_date):
    """
    Get jhu-csse from Our COVIDcast API
    """
    return covidcast.signal("jhu-csse", "confirmed_7dav_incidence_prop",
                        start_date, end_date, geo_res).drop(
                            columns = ["direction", "issue", "lag", "stderr"],
                            axis = 1).rename(
                                {"geo_value":"geo_id", "value": "val"}, axis =␣
↪1)
```

[12]:
```
def calculate_corr_df(geo_res, smoother, start_date, end_date, csv_df_all, lag):
    """
    Calculate
    """
    api_df = get_api_df(geo_res, start_date + timedelta(days = lag), end_date +␣
↪timedelta(days = lag))
    csv_df = csv_df_all.loc[
        (csv_df_all["geo_res"] == geo_res) \
        & (csv_df_all["smoother"] == smoother) \
        & (csv_df_all["time_value"] <= end_date) \
        & (csv_df_all["time_value"] >= start_date)
    ]

    if geo_res != "state":
        api_df["geo_id"] = api_df["geo_id"].astype(int)
        csv_df["geo_id"] = csv_df["geo_id"].astype(int)
    geo_list = list(set(api_df["geo_id"]).intersection(set(csv_df["geo_id"])))

    api_df["time_value"] = [x - timedelta(days = lag) for x in␣
↪api_df["time_value"]]

    test_df = csv_df.merge(api_df, on=["time_value", "geo_id"], suffixes =␣
↪("_test", "_response"))

    # Get time series correlation
    corrs = []
    p_values = []
    for loc in geo_list:
        tempt_df = test_df[test_df["geo_id"] == loc]
        corr, p = spearmanr(tempt_df["val_test"], tempt_df["val_response"])
        corrs.append(corr)
        p_values.append(p)
```

```
    return pd.DataFrame({"geo_id": geo_list, "rank_corr": corrs, "p_value":␣
↪p_values}).sort_values("rank_corr")
```

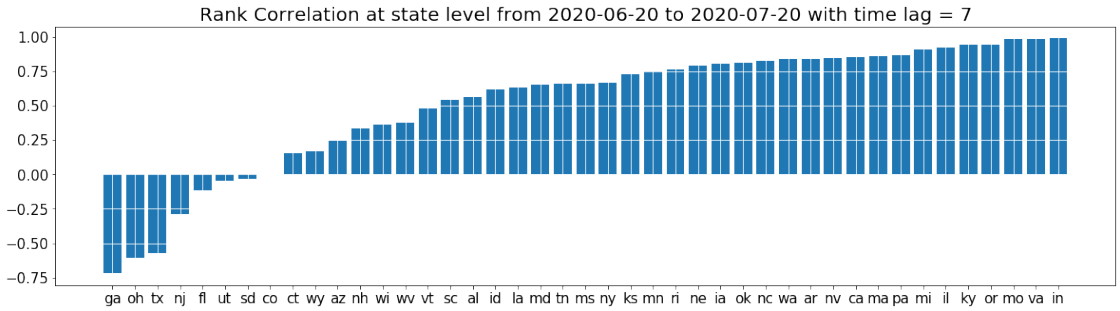
0.0.1 State Level

[13]:
```
geo_res = "state"
smoother = "smoothed"
start_date = datetime(2020, 6, 20)
end_date = datetime(2020, 7, 20)
```

[14]:
```
lag = 7
state_result_df = calculate_corr_df(geo_res, smoother, start_date, end_date,␣
↪csv_df_all, lag)
print("%d out of %d %ss have rank correlations larger than 0.5.
↪"%(sum(state_result_df["rank_corr"] > 0.5), state_result_df["rank_corr"].
↪unique().shape[0], geo_res))
plt.figure(figsize = (20, 5))
plt.grid(color='w', linestyle='solid')
plt.bar(state_result_df["geo_id"], state_result_df["rank_corr"])
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.title("Rank Correlation at %s level from %s to %s with time lag =␣
↪%d"%(geo_res, start_date.date(), end_date.date(), lag), fontsize = 20)
```

```
28 out of 43 states have rank correlations larger than 0.5.
```

[14]:
```
Text(0.5, 1.0, 'Rank Correlation at state level from 2020-06-20 to 2020-07-20
with time lag = 7')
```

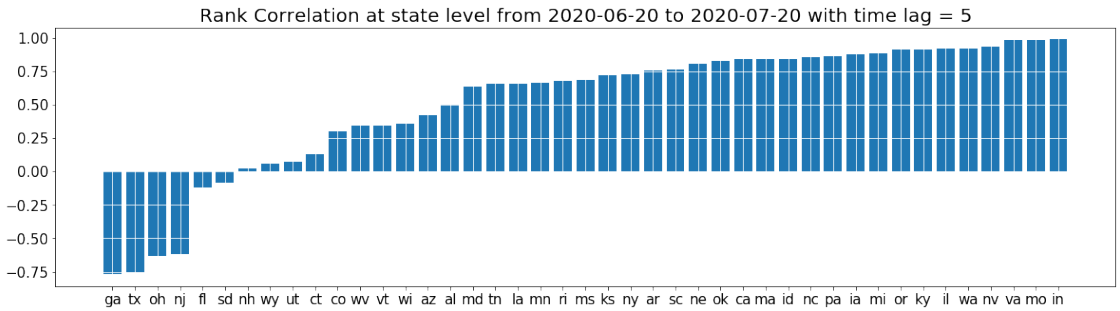


[15]:
```
lag = 5
state_result_df = calculate_corr_df(geo_res, smoother, start_date, end_date,␣
↪csv_df_all, lag)
print("%d out of %d %ss have rank correlations larger than 0.5.
↪"%(sum(state_result_df["rank_corr"] > 0.5), state_result_df["rank_corr"].
↪unique().shape[0], geo_res))
```

```
plt.figure(figsize = (20, 5))
plt.grid(color='w', linestyle='solid')
plt.bar(state_result_df["geo_id"], state_result_df["rank_corr"])
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.title("Rank Correlation at %s level from %s to %s with time lag =␣
 ↪%d"%(geo_res, start_date.date(), end_date.date(), lag), fontsize = 20)
```

```
28 out of 43 states have rank correlations larger than 0.5.
```

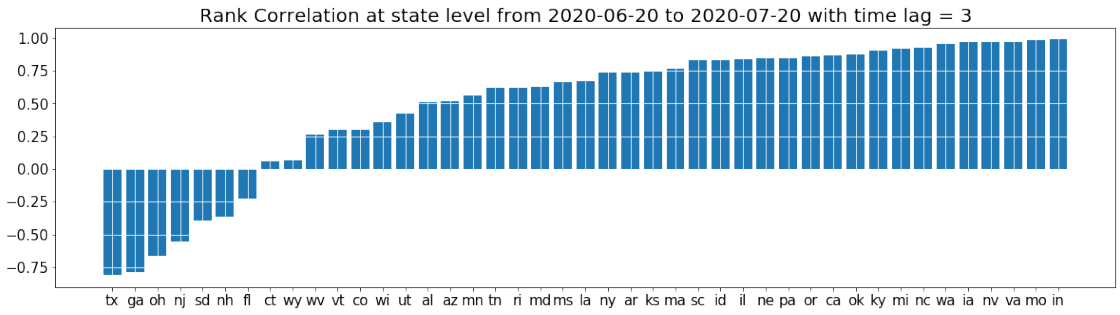
[15]:
```
Text(0.5, 1.0, 'Rank Correlation at state level from 2020-06-20 to 2020-07-20
with time lag = 5')
```



[16]:
```
lag = 3
state_result_df = calculate_corr_df(geo_res, smoother, start_date, end_date,␣
 ↪csv_df_all, lag)
print("%d out of %d %ss have rank correlations larger than 0.5.
 ↪"%(sum(state_result_df["rank_corr"] > 0.5), state_result_df["rank_corr"].
 ↪unique().shape[0], geo_res))
plt.figure(figsize = (20, 5))
plt.grid(color='w', linestyle='solid')
plt.bar(state_result_df["geo_id"], state_result_df["rank_corr"])
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
plt.title("Rank Correlation at %s level from %s to %s with time lag =␣
 ↪%d"%(geo_res, start_date.date(), end_date.date(), lag), fontsize = 20)
```

```
29 out of 43 states have rank correlations larger than 0.5.
```

[16]:
```
Text(0.5, 1.0, 'Rank Correlation at state level from 2020-06-20 to 2020-07-20
with time lag = 3')
```



0.0.2 MSA Level

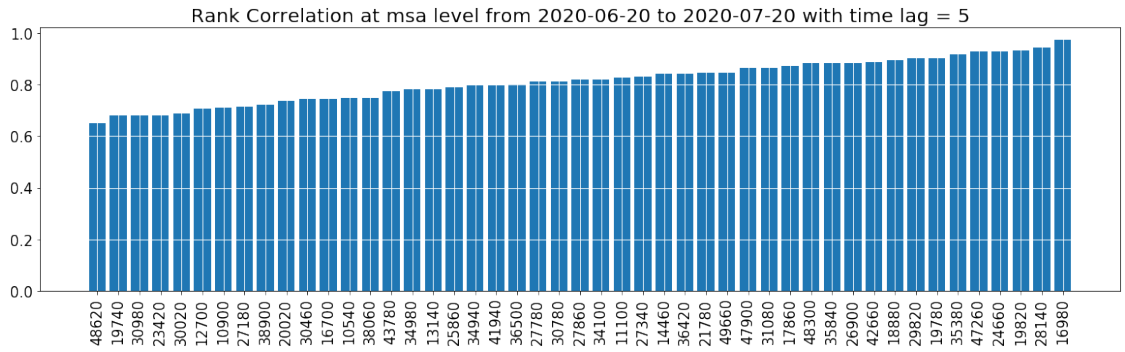
[20]:
```
geo_res = "msa"
smoother = "smoothed"
start_date = datetime(2020, 6, 20)
end_date = datetime(2020, 7, 20)
```

[25]:
```
lag = 5
result_df = calculate_corr_df(geo_res, smoother, start_date, end_date,␣
 ↪csv_df_all, lag)
print("%d out of %d %ss have rank correlations larger than 0.5.
 ↪"%(sum(result_df["rank_corr"] > 0.5), result_df["rank_corr"].unique().
 ↪shape[0], geo_res))
print("%d out of %d %ss have rank correlations larger than 0.
 ↪"%(sum(result_df["rank_corr"] > 0), result_df["rank_corr"].unique().
 ↪shape[0], geo_res))


result_df = result_df.loc[result_df.index[-50:]].dropna()
plt.figure(figsize = (20, 5))
plt.grid(color='w', linestyle='solid')
plt.bar(result_df["geo_id"].astype(str), result_df["rank_corr"])
plt.xticks(fontsize = 15, rotation = 90)
plt.yticks(fontsize = 15)
plt.title("Rank Correlation at %s level from %s to %s with time lag =␣
 ↪%d"%(geo_res, start_date.date(), end_date.date(), lag), fontsize = 20)
```

```
/Users/jingjingtang/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  from ipykernel import kernelapp as app
```

```
73 out of 190 msas have rank correlations larger than 0.5.
117 out of 190 msas have rank correlations larger than 0.
```

[25]:
```
Text(0.5, 1.0, 'Rank Correlation at msa level from 2020-06-20 to 2020-07-20 with
time lag = 5')
```

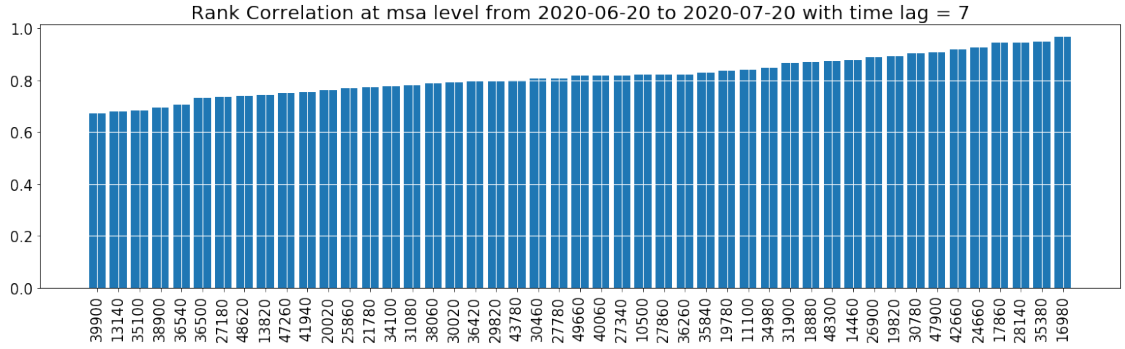


[26]:
```
lag = 7
result_df = calculate_corr_df(geo_res, smoother, start_date, end_date,␣
 ↪csv_df_all, lag)
print("%d out of %d %ss have rank correlations larger than 0.5.
 ↪"%(sum(result_df["rank_corr"] > 0.5), result_df["rank_corr"].unique().
 ↪shape[0], geo_res))
print("%d out of %d %ss have rank correlations larger than 0.
 ↪"%(sum(result_df["rank_corr"] > 0), result_df["rank_corr"].unique().
 ↪shape[0], geo_res))



result_df = result_df.loc[result_df.index[-50:]].dropna()
plt.figure(figsize = (20, 5))
plt.grid(color='w', linestyle='solid')
plt.bar(result_df["geo_id"].astype(str), result_df["rank_corr"])
plt.xticks(fontsize = 15, rotation = 90)
plt.yticks(fontsize = 15)
plt.title("Rank Correlation at %s level from %s to %s with time lag =␣
 ↪%d"%(geo_res, start_date.date(), end_date.date(), lag), fontsize = 20)
```

```
/Users/jingjingtang/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  from ipykernel import kernelapp as app
```

```
75 out of 190 msas have rank correlations larger than 0.5.
125 out of 190 msas have rank correlations larger than 0.
```

[26]:
```
Text(0.5, 1.0, 'Rank Correlation at msa level from 2020-06-20 to 2020-07-20 with
time lag = 7')
```



[ ]: