Imports

In [2]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import MinMaxScaler, RobustScaler
from scipy.stats.mstats import winsorize
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.metrics import root_mean_squared_error, r2_score, make_scorer
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor

np.random.seed(42) #Sets the random seed.
```

A place to put some functions used throughout the notebook.

In [4]:

```
# Function to print a summary from dataframes. Adds skewness and kurtosis values.
def see_summary (data_frame):
    summary = data_frame.describe(include='all').T
    summary['skewness'] = data_frame.skew()
    summary['kurtosis'] = data_frame.kurtosis()
    print(summary)

# Function to plot Distributions
def distribution (df_column, column_name):
    plt.hist(df_column,bins=25)
    plt.title(f"Histogram of {column_name}")
    plt.xlabel(column_name)
    plt.ylabel("Frequency")
    plt.show()
```

Loading in Data and Getting the Summary Statistics Dropping Time_Value and Smoothed_wtested_positive_14d columns

In [6]:

```
# print(os.getcwd())
df = pd.read_csv('covidcast_new-1.csv')
# print (df.head())

if "time_value" in df.columns:
    df = df.drop('time_value', axis=1)

# see_summary(df)

# Dropping rows w/o the target label.
df = df[df['smoothed_wtested_positive_14d'].notnull()]
```

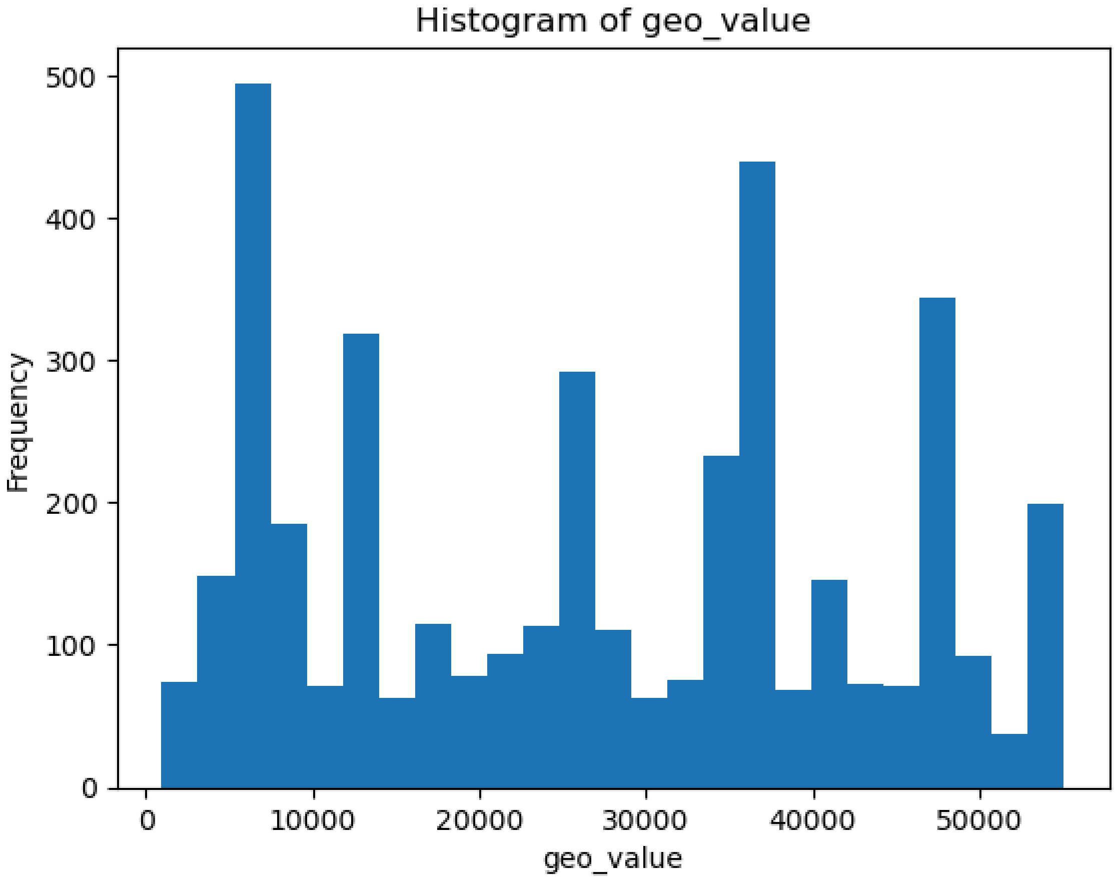
```python

see_summary(df)

# Checking for empty values.
empty = df.isnull().sum()
# print(empty)
percent_null = (empty / len(df)) * 100
null_summary = pd.DataFrame({'Null Count': empty, 'Percent Null': percent_null})
print(null_summary)
```

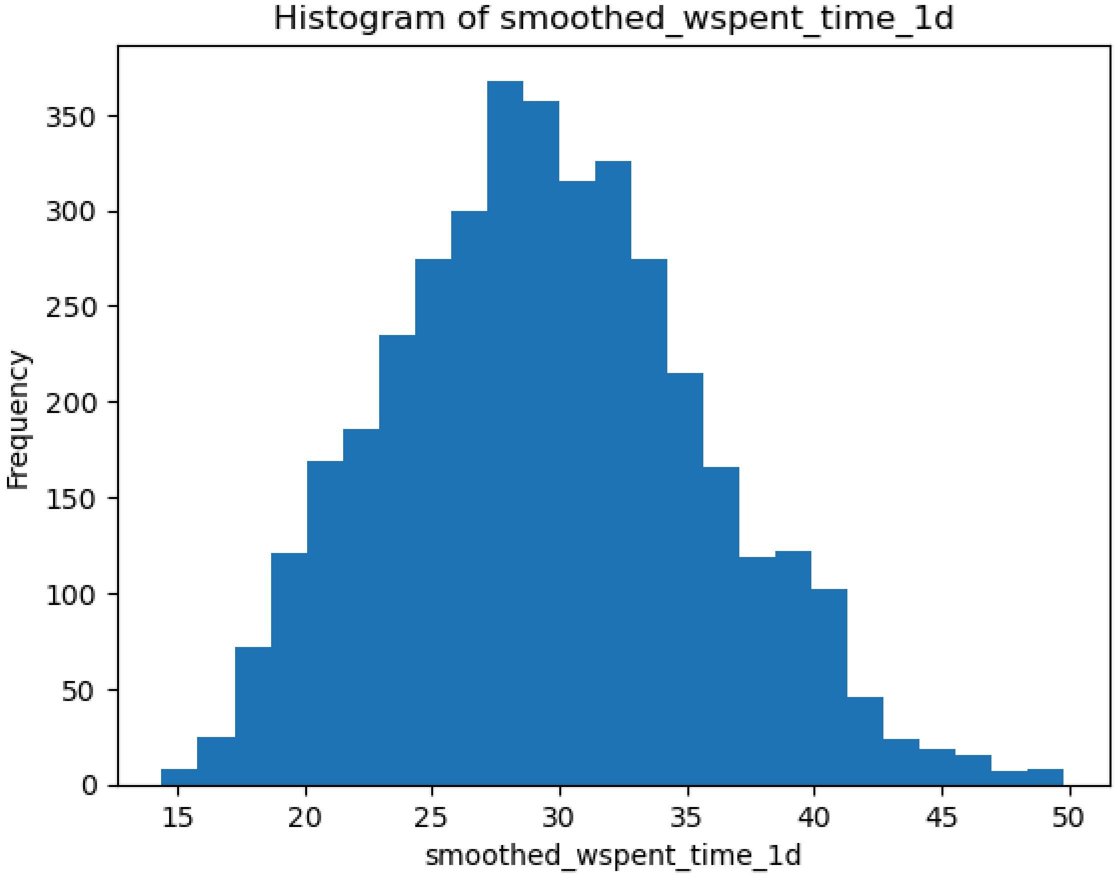
```
                                       count          mean           std  \
geo_value                             3994.0  26665.803205  15971.575875
smoothed_wspent_time_1d               3873.0     29.540826      6.165547
smoothed_wtested_14d                  3879.0     15.685206      5.090793
smoothed_wpublic_transit_1d           3873.0      3.955646      5.141994
smoothed_wworried_become_ill          3903.0     70.512220      5.402762
smoothed_wvaccine_likely_govt_health  3903.0     32.974395      7.484255
smoothed_wshop_1d                     3873.0     52.650528      4.830845
smoothed_wtested_positive_14d         3994.0     17.250770      7.419522
smoothed_wwork_outside_home_1d        3873.0     31.864066      5.510357
smoothed_wothers_masked               3899.0     82.046408     13.480812
smoothed_wcli                         3859.0      1.061285      0.508627
smoothed_wcovid_vaccinated            3894.0     12.434098      6.143912
smoothed_wvaccine_likely_friends      3903.0     35.650618      4.814160
smoothed_wrestaurant_1d               3873.0     13.280293      5.520454
smoothed_wvaccine_likely_politicians  3904.0     11.176533      3.619874
smoothed_wvaccine_likely_who          3906.0     37.847991      8.030457
smoothed_wwearing_mask                3899.0     92.475646      5.526132
smoothed_wlarge_event_1d              3873.0      7.279319      3.842110

                                              min           25%           50%  \
geo_value                             1000.000000  12000.000000  26125.000000
smoothed_wspent_time_1d                 14.410554     25.064621     29.237196
smoothed_wtested_14d                     5.437303     11.806790     14.637136
smoothed_wpublic_transit_1d              0.115497      1.874536      2.548976
smoothed_wworried_become_ill            52.405818     67.245161     70.981141
smoothed_wvaccine_likely_govt_health    17.410006     26.907812     32.732026
smoothed_wshop_1d                       39.237833     49.263607     52.036982
smoothed_wtested_positive_14d            1.548609     11.616797     16.530456
smoothed_wwork_outside_home_1d          14.558697     28.047715     31.672672
smoothed_wothers_masked                 42.950717     73.219360     87.896169
smoothed_wcli                            0.000000      0.691872      0.982991
smoothed_wcovid_vaccinated               0.891041      7.547537     11.543419
smoothed_wvaccine_likely_friends        22.622258     31.942427     35.381211
smoothed_wrestaurant_1d                  0.424278      8.810386     13.094899
smoothed_wvaccine_likely_politicians     2.123555      8.428762     10.711590
smoothed_wvaccine_likely_who            20.127120     31.260485     38.262343
smoothed_wwearing_mask                  74.543138     88.491468     94.900084
smoothed_wlarge_event_1d                 0.632336      4.112238      6.486758

                                               75%           max  skewness  \
geo_value                             39000.000000  55025.000000  0.066418
smoothed_wspent_time_1d                  33.608535     49.831174  0.252439
smoothed_wtested_14d                     18.775748     34.951317  0.762946
smoothed_wpublic_transit_1d               3.539643     36.015469  3.803156
smoothed_wworried_become_ill             74.379649     85.446476 -0.421239
smoothed_wvaccine_likely_govt_health     38.387293     53.992167  0.211561
smoothed_wshop_1d                        55.867318     67.597229  0.365532
smoothed_wtested_positive_14d            22.153012     46.644291  0.484910
smoothed_wwork_outside_home_1d           35.472144     49.848038  0.131769
smoothed_wothers_masked                  92.435684     98.935958 -1.103265
smoothed_wcli                             1.354964      3.356476  0.826955
smoothed_wcovid_vaccinated               16.622363     45.466215  0.830052
smoothed_wvaccine_likely_friends         39.094392     54.182794  0.310092
smoothed_wrestaurant_1d                  17.753224     28.340772  0.151007
smoothed_wvaccine_likely_politicians     13.525339     25.197001  0.564857
```

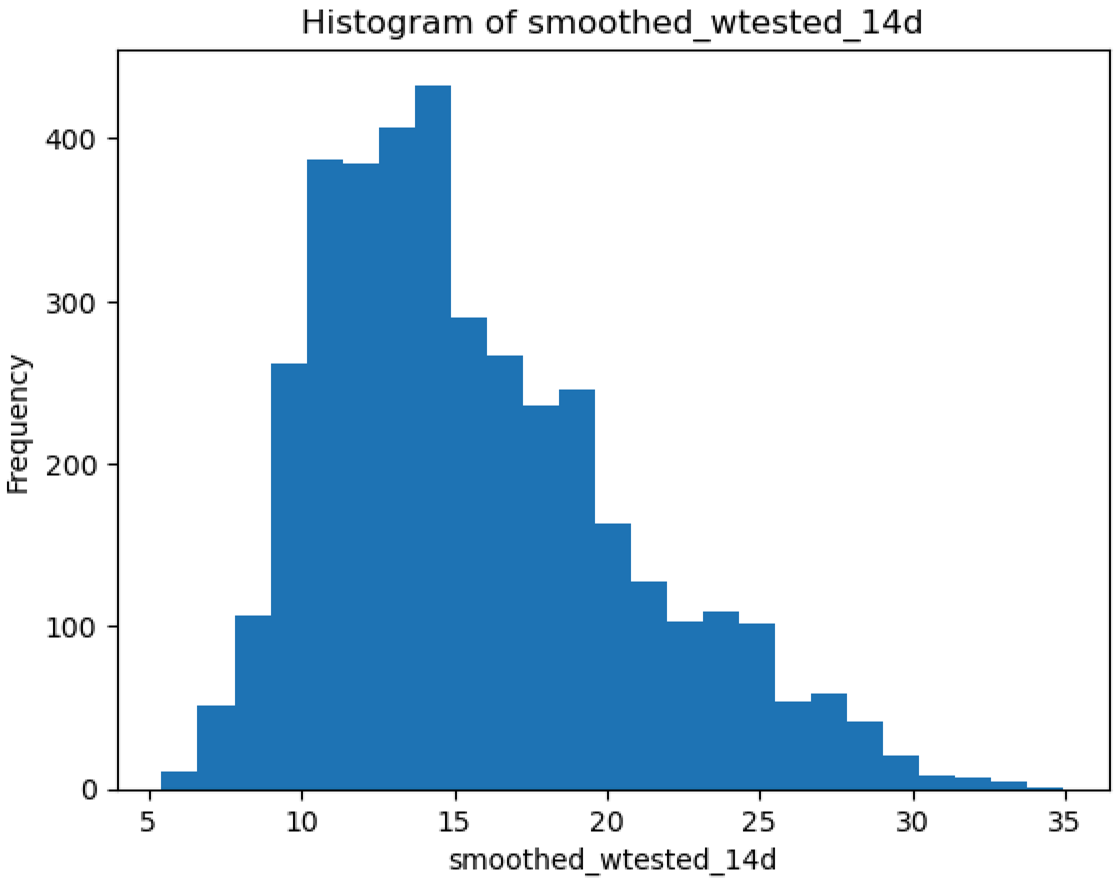
```
smoothed_wvaccine_likely_who             43.922948     58.742213 -0.009496
smoothed_wwearing_mask                   96.702689     99.732673 -1.051875
smoothed_wlarge_event_1d                  9.971153     22.619464  0.625615

                                       kurtosis
geo_value                             -1.298427
smoothed_wspent_time_1d               -0.233199
smoothed_wtested_14d                   0.100584
smoothed_wpublic_transit_1d           14.693356
smoothed_wworried_become_ill           0.061129
smoothed_wvaccine_likely_govt_health  -0.745169
smoothed_wshop_1d                     -0.288753
smoothed_wtested_positive_14d         -0.059485
smoothed_wwork_outside_home_1d        -0.027514
smoothed_wothers_masked                0.230422
smoothed_wcli                          1.016754
smoothed_wcovid_vaccinated             1.091027
smoothed_wvaccine_likely_friends      -0.217339
smoothed_wrestaurant_1d               -0.899545
smoothed_wvaccine_likely_politicians   0.053907
smoothed_wvaccine_likely_who          -0.914625
smoothed_wwearing_mask                 0.138972
smoothed_wlarge_event_1d              -0.358648
                                      Null Count  Percent Null
geo_value                                      0      0.000000
smoothed_wspent_time_1d                      121      3.029544
smoothed_wtested_14d                         115      2.879319
smoothed_wpublic_transit_1d                  121      3.029544
smoothed_wworried_become_ill                  91      2.278418
smoothed_wvaccine_likely_govt_health          91      2.278418
smoothed_wshop_1d                            121      3.029544
smoothed_wtested_positive_14d                  0      0.000000
smoothed_wwork_outside_home_1d               121      3.029544
smoothed_wothers_masked                       95      2.378568
smoothed_wcli                                135      3.380070
smoothed_wcovid_vaccinated                   100      2.503756
smoothed_wvaccine_likely_friends              91      2.278418
smoothed_wrestaurant_1d                      121      3.029544
smoothed_wvaccine_likely_politicians          90      2.253380
smoothed_wvaccine_likely_who                  88      2.203305
smoothed_wwearing_mask                        95      2.378568
smoothed_wlarge_event_1d                     121      3.029544
```

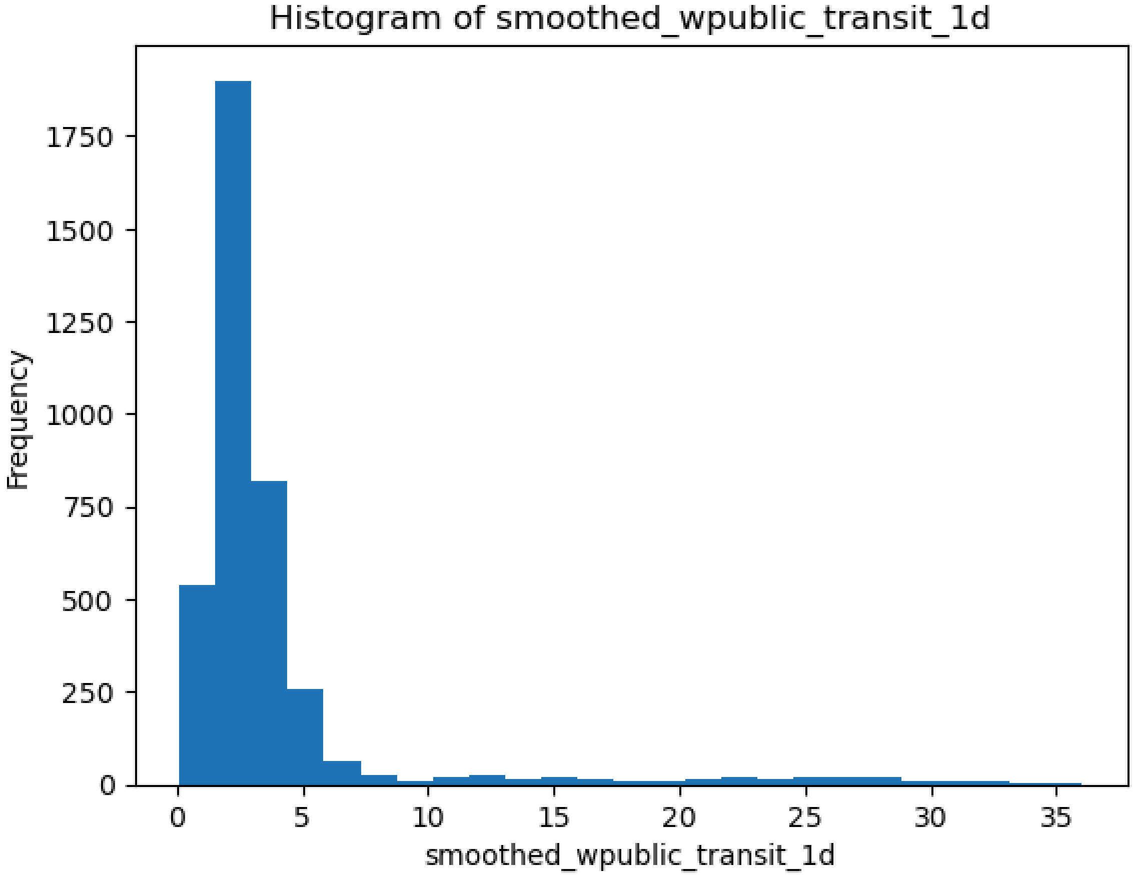
Making Histograms for each column in the data frame.

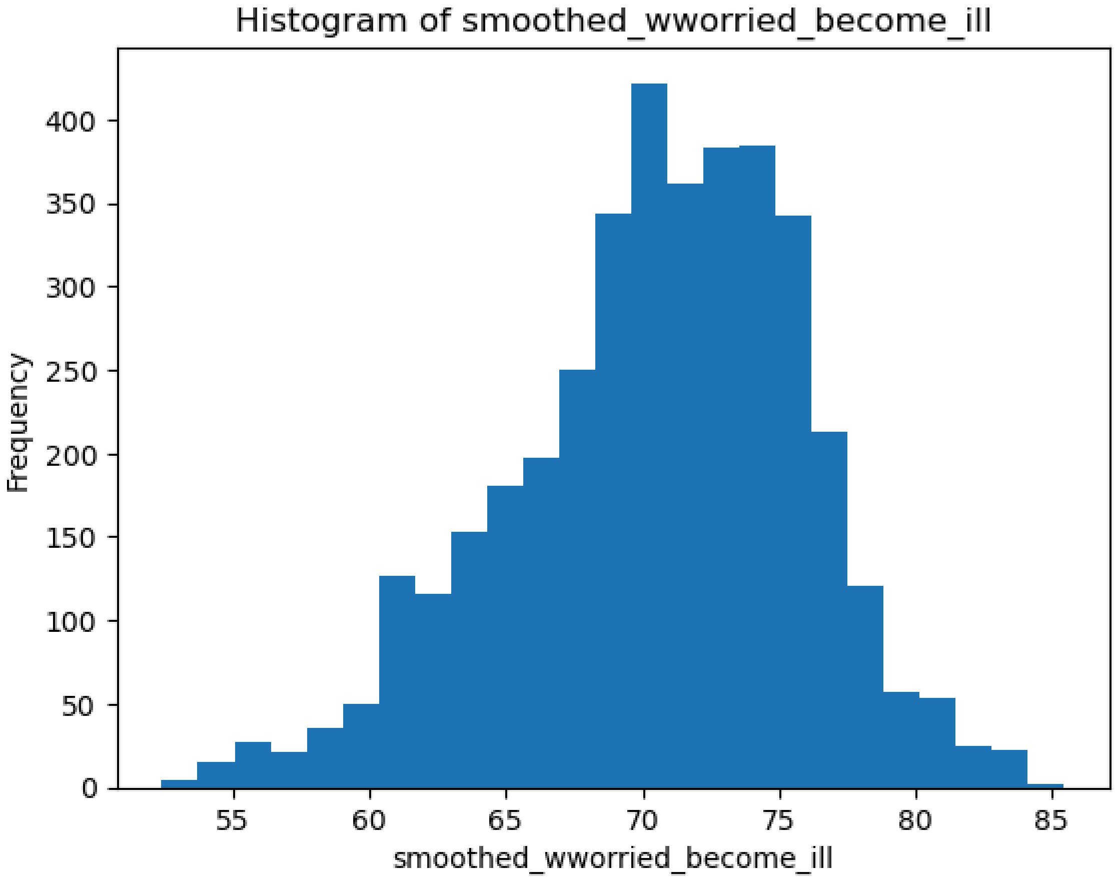
```
In [8]: for col in df.columns:
            distribution(df[col], col)
```

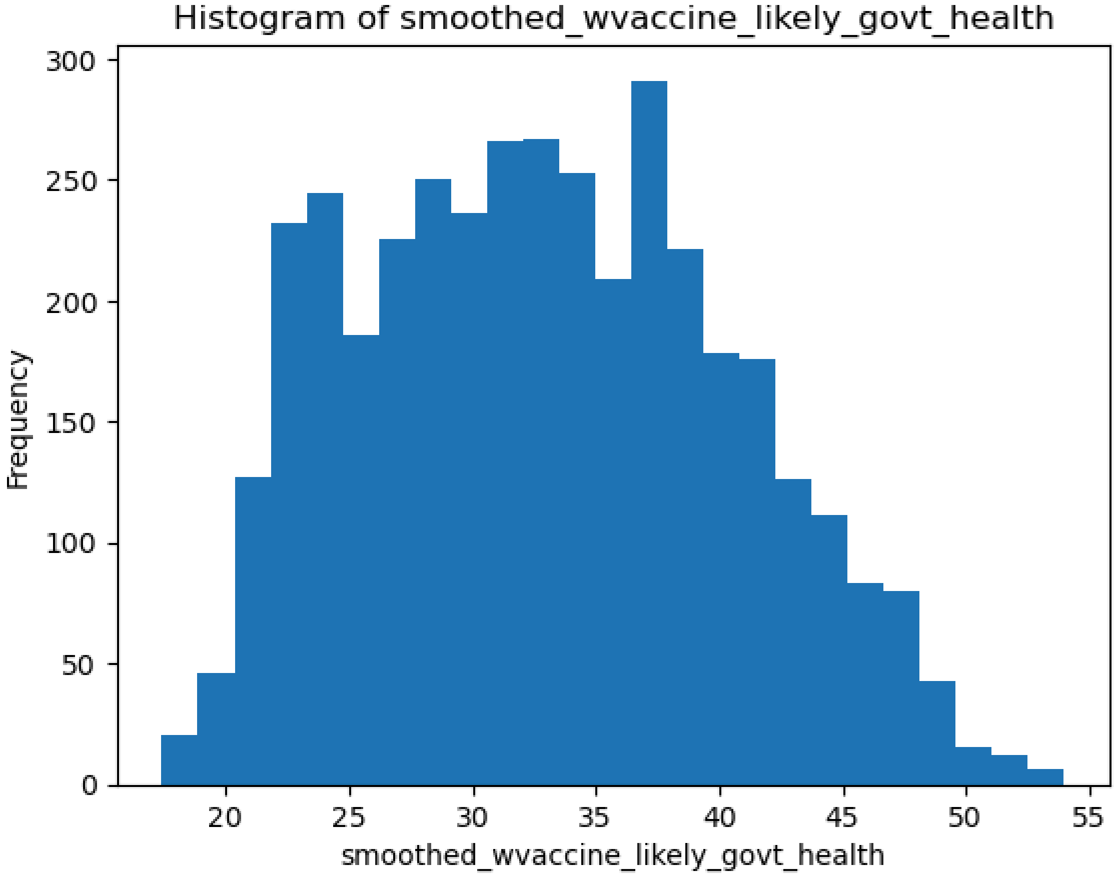

Histogram of geo_value
Frequency
0
100
200
300
400
500
0
10000
20000
30000
40000
50000
geo_value

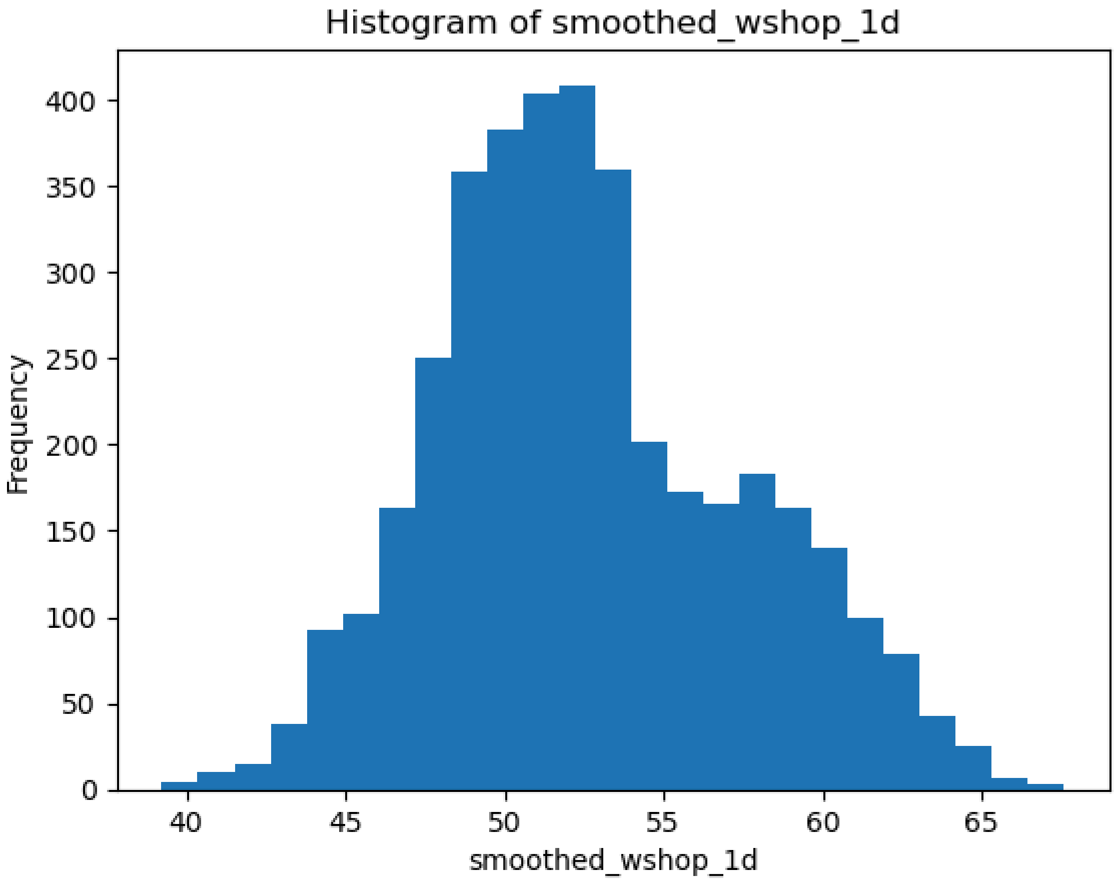

Histogram of smoothed_wspent_time_1d
Frequency
0
50
100
150
200
250
300
350
15
20
25
30
35
40
45
50
smoothed_wspent_time_1d


Histogram of smoothed_wtested_14d
Frequency
0
100
200
300
400
5
10
15
20
25
30
35
smoothed_wtested_14d

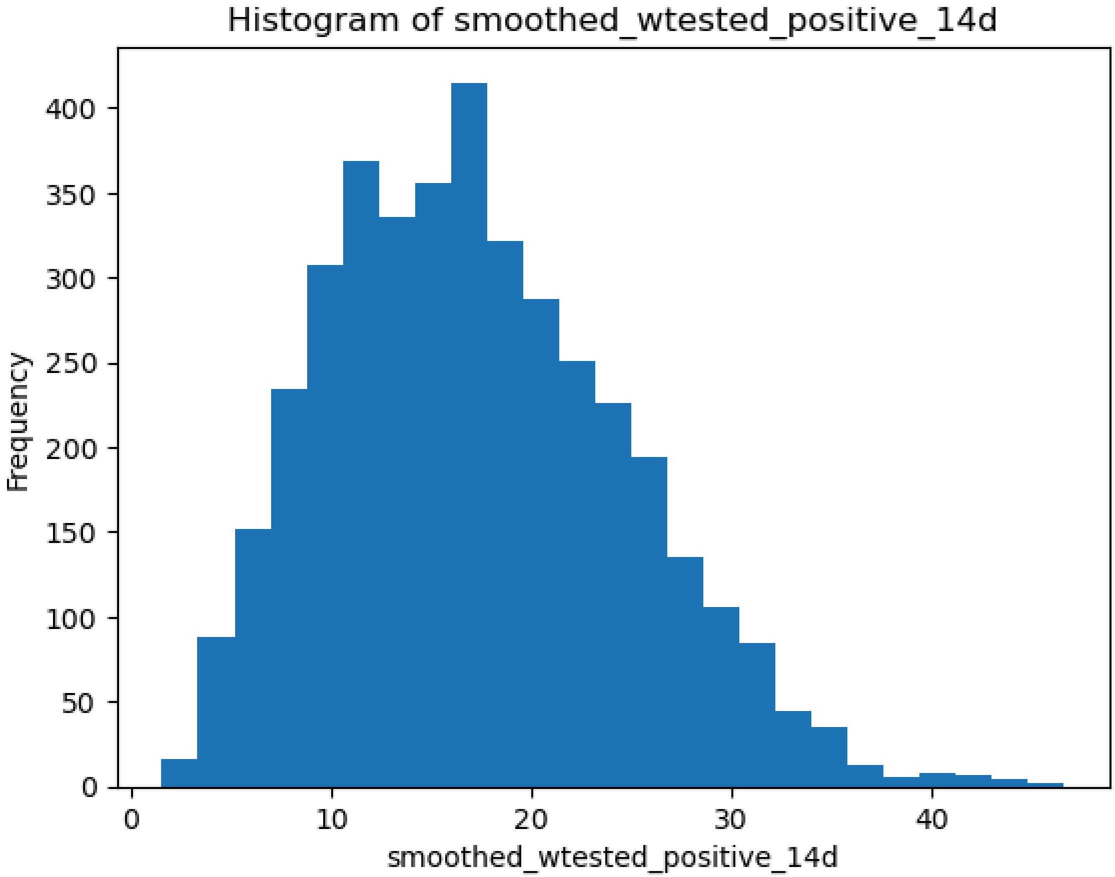

Histogram of smoothed_wpublic_transit_1d
Frequency
0
250
500
750
1000
1250
1500
1750
0
5
10
15
20
25
30
35
smoothed_wpublic_transit_1d


Histogram of smoothed_wworried_become_ill
Frequency
0
50
100
150
200
250
300
350
400
55
60
65
70
75
80
85
smoothed_wworried_become_ill

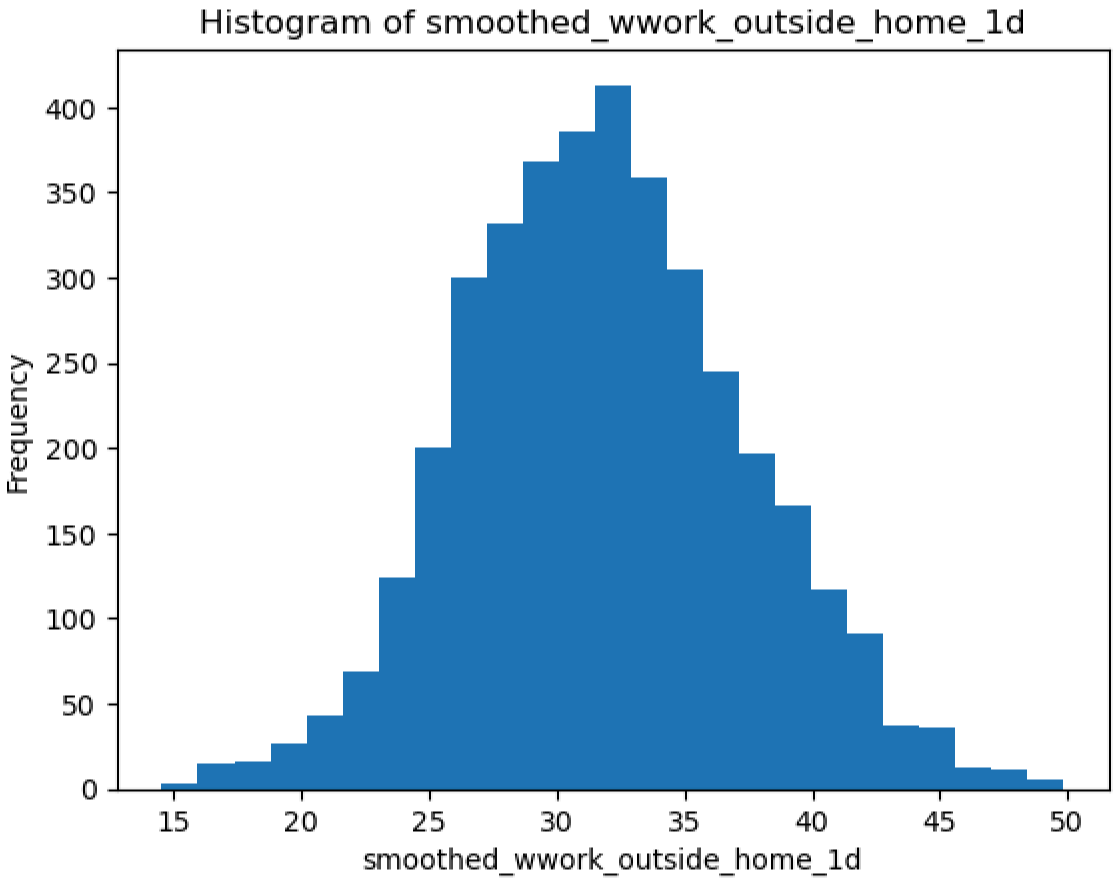

Histogram of smoothed_wvaccine_likely_govt_health
Frequency
0
50
100
150
200
250
300
20
25
30
35
40
45
50
55
smoothed_wvaccine_likely_govt_health

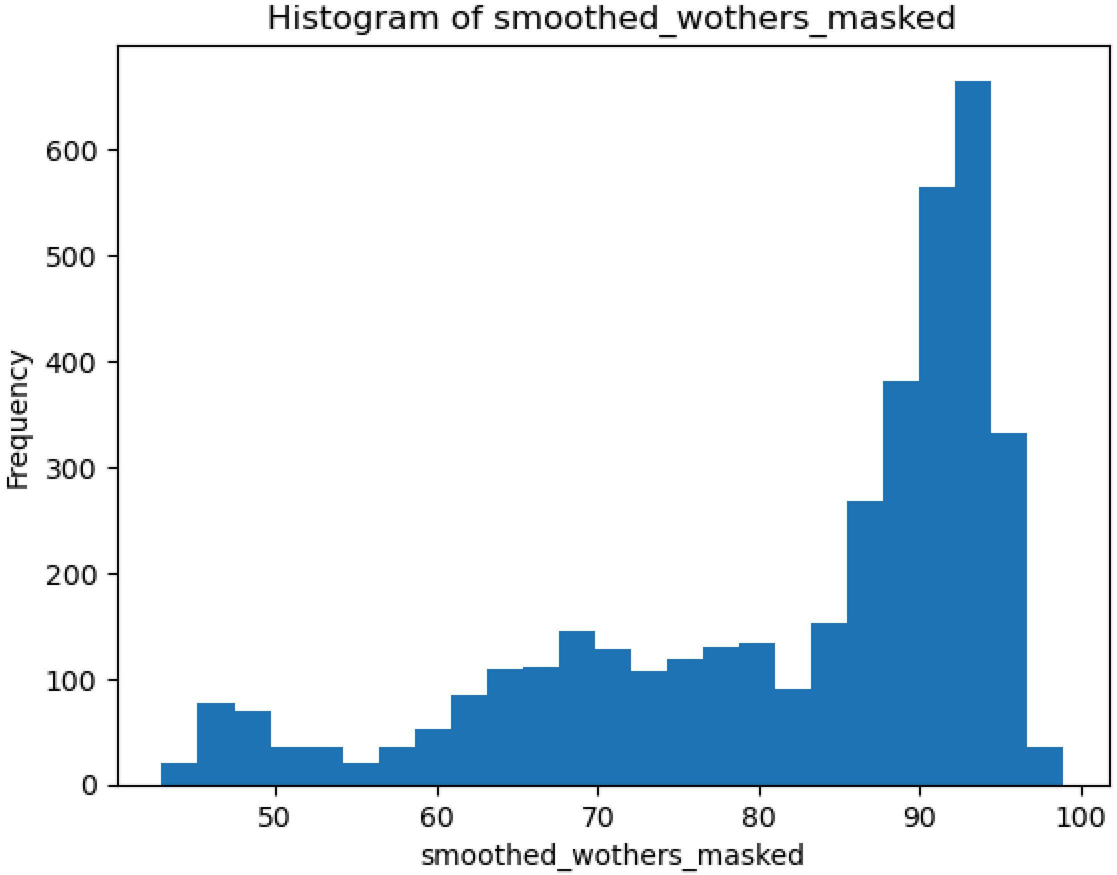


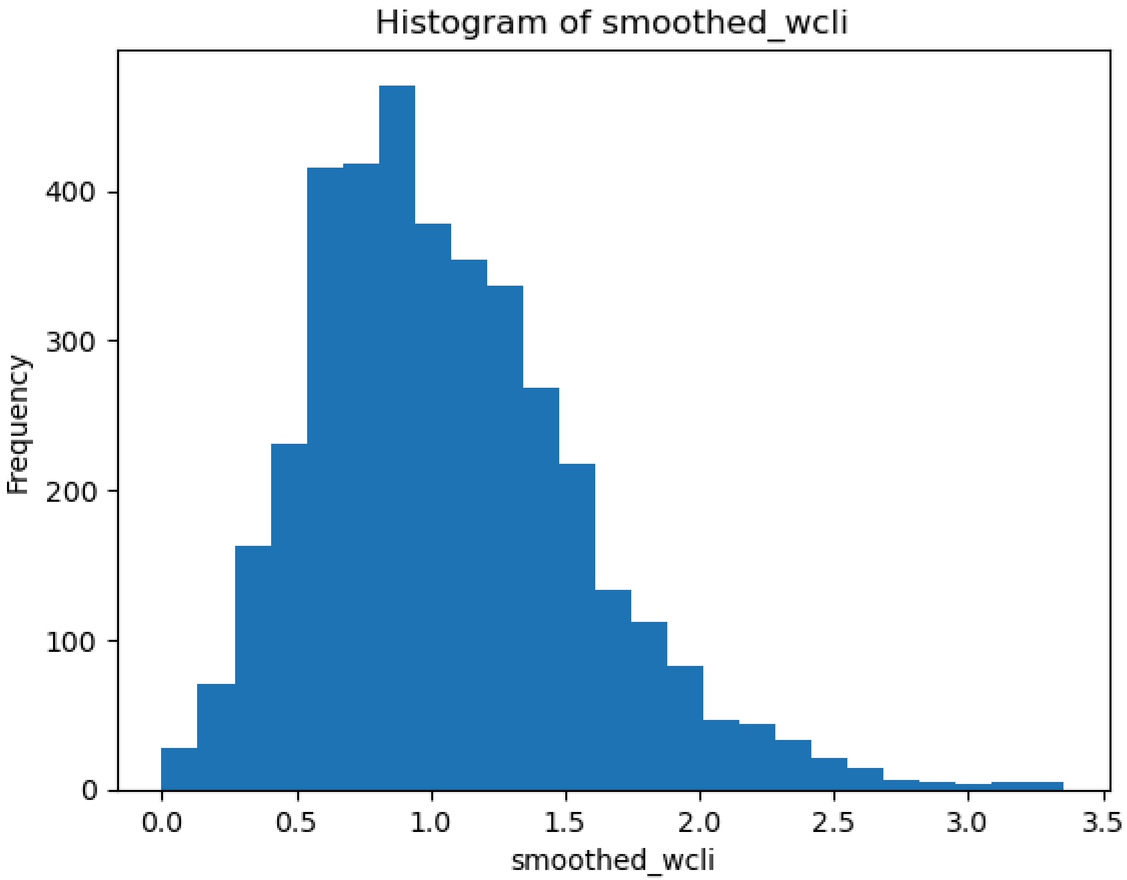
Histogram of smoothed_wshop_1d

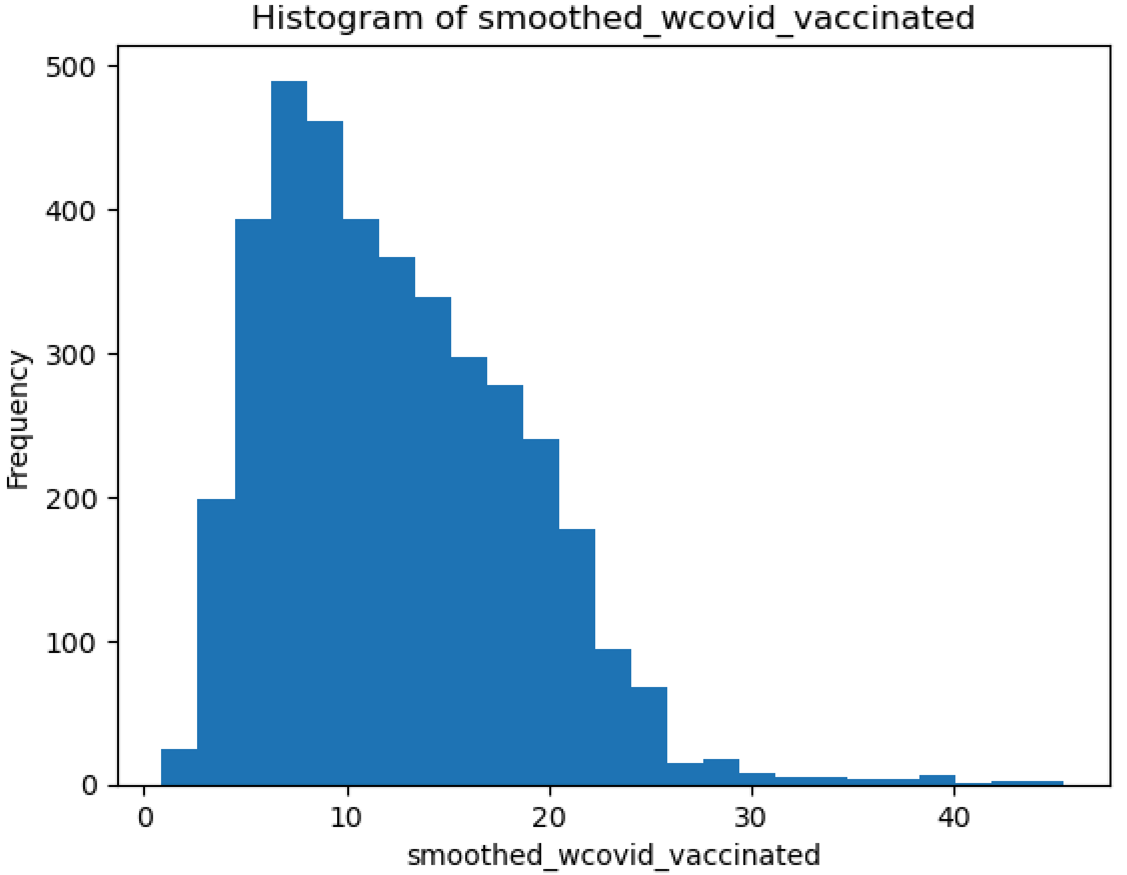


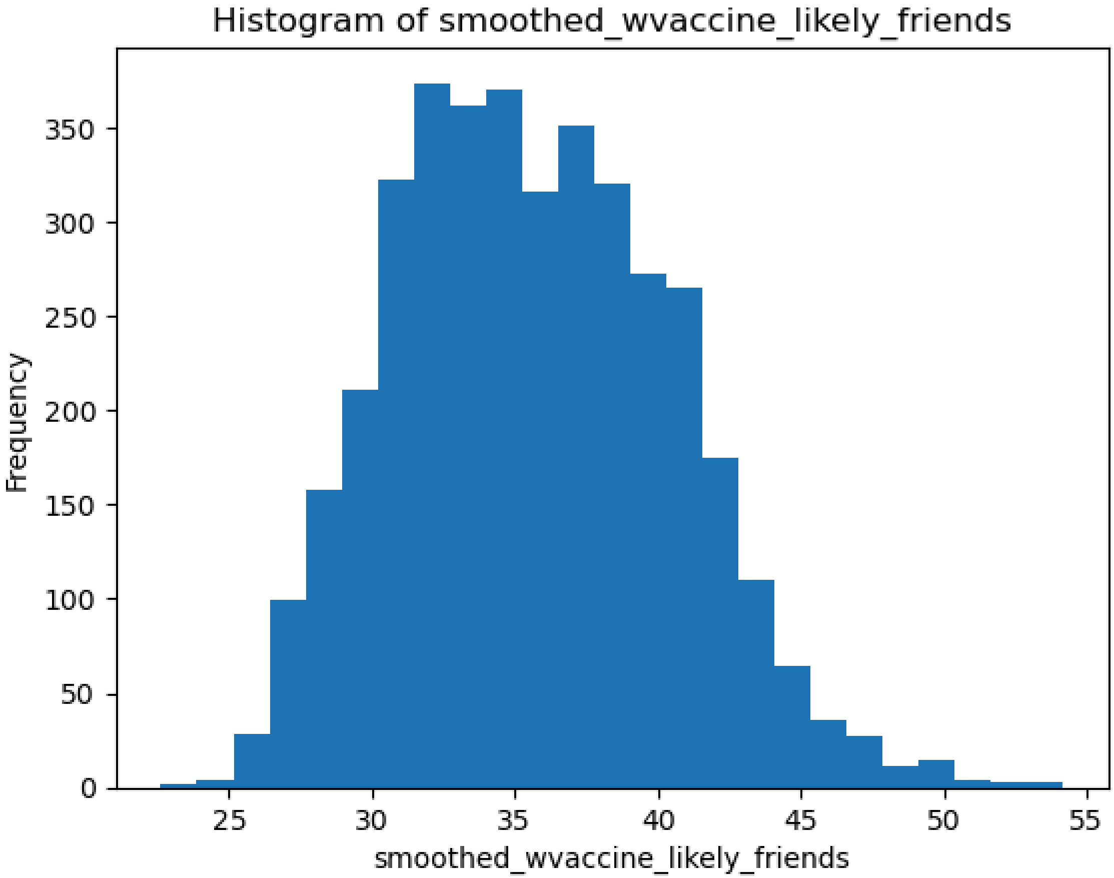
Histogram of smoothed_wtested_positive_14d

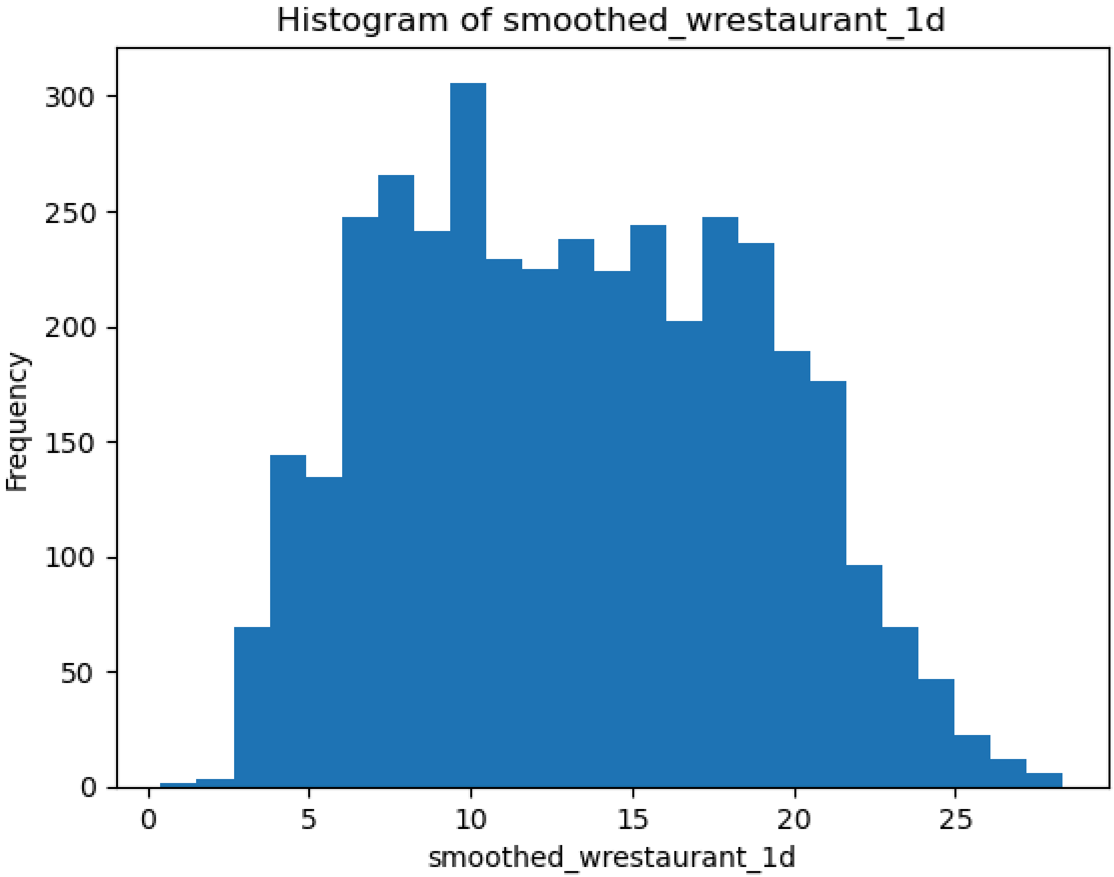

Histogram of smoothed_wwork_outside_home_1d
Frequency
0
50
100
150
200
250
300
350
400
15
20
25
30
35
40
45
50
smoothed_wwork_outside_home_1d

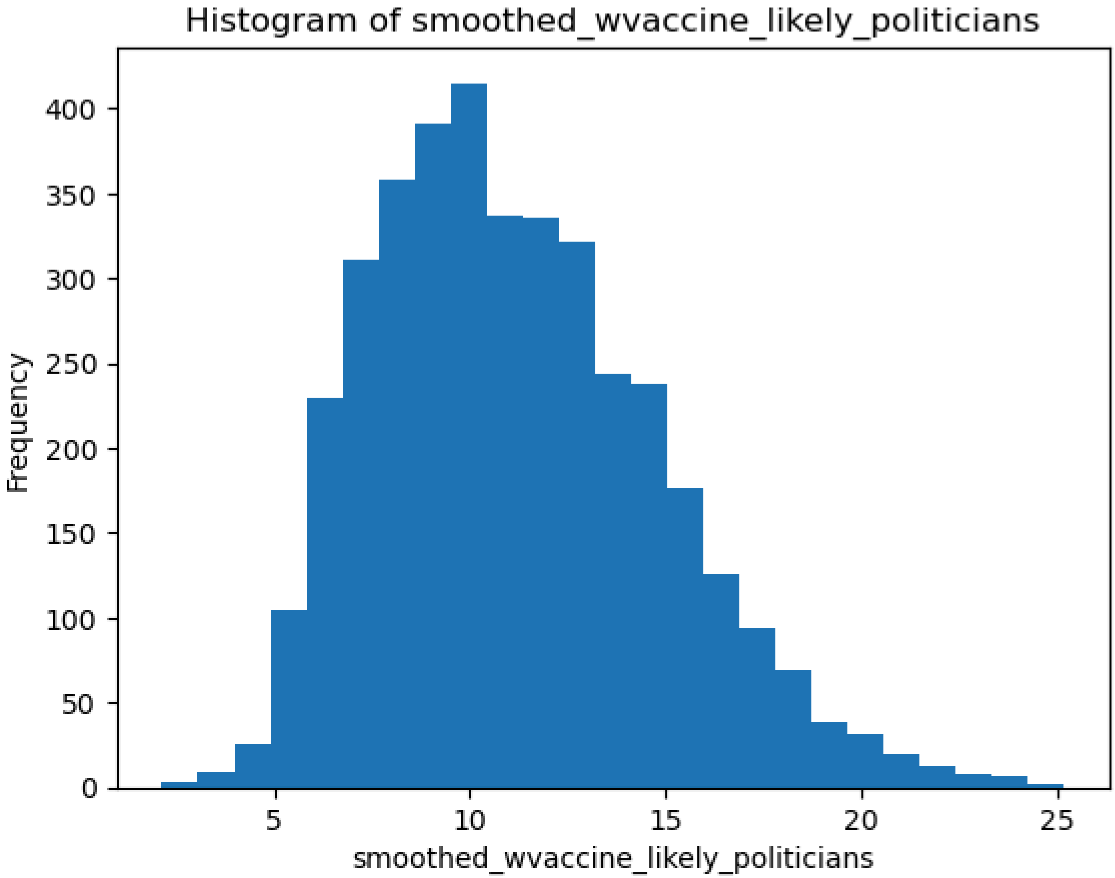

Histogram of smoothed_wothers_masked
Frequency
0
100
200
300
400
500
600
50
60
70
80
90
100
smoothed_wothers_masked


Histogram of smoothed_wcli
Frequency
0
100
200
300
400
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
smoothed_wcli

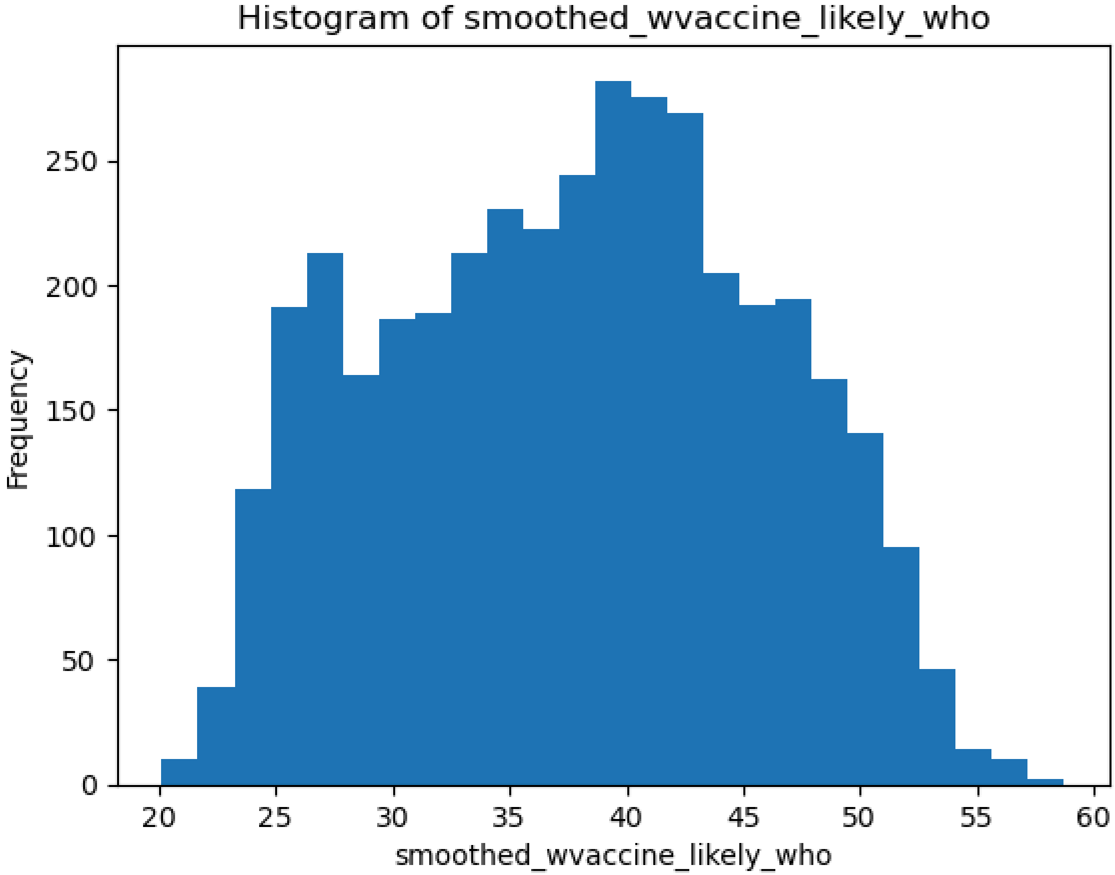

Histogram of smoothed_wcovid_vaccinated
Frequency
0
100
200
300
400
500
0
10
20
30
40
smoothed_wcovid_vaccinated


Histogram of smoothed_wvaccine_likely_friends
Frequency
0
50
100
150
200
250
300
350
25
30
35
40
45
50
55
smoothed_wvaccine_likely_friends

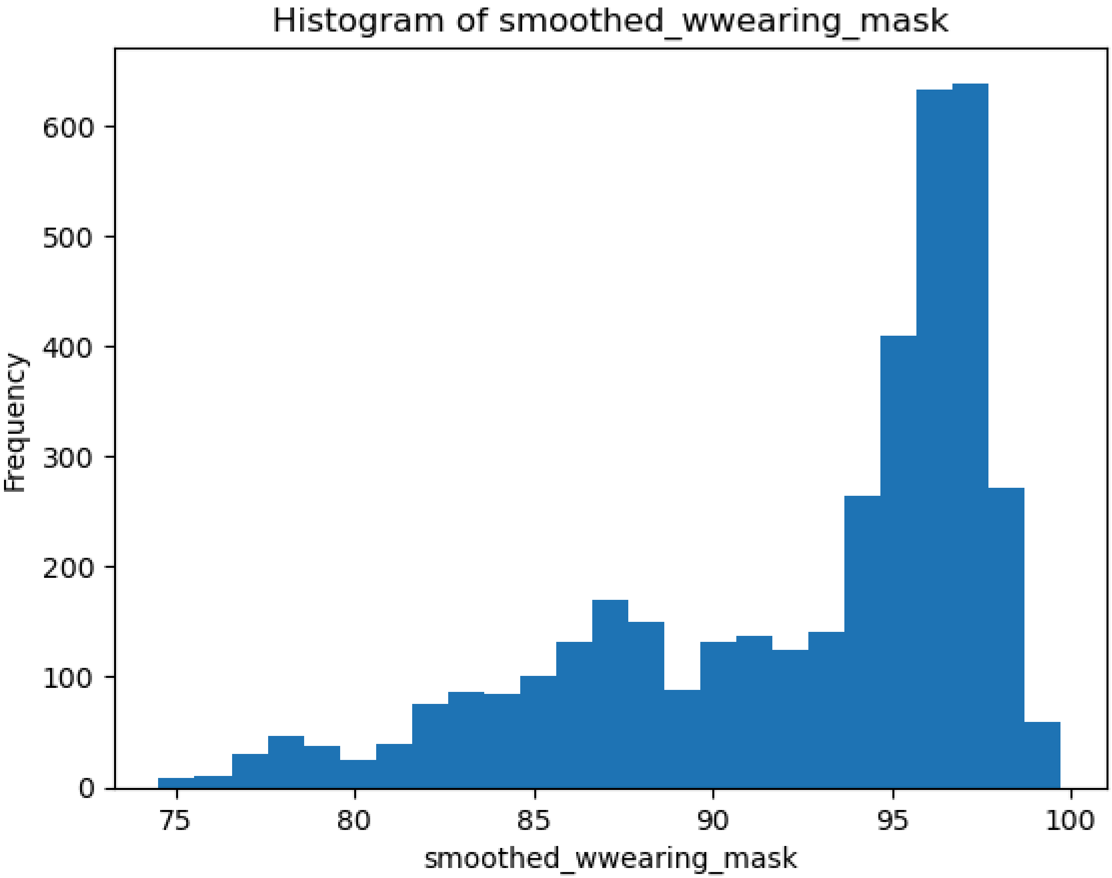

Histogram of smoothed_wrestaurant_1d
Frequency
0
50
100
150
200
250
300
0
5
10
15
20
25
smoothed_wrestaurant_1d

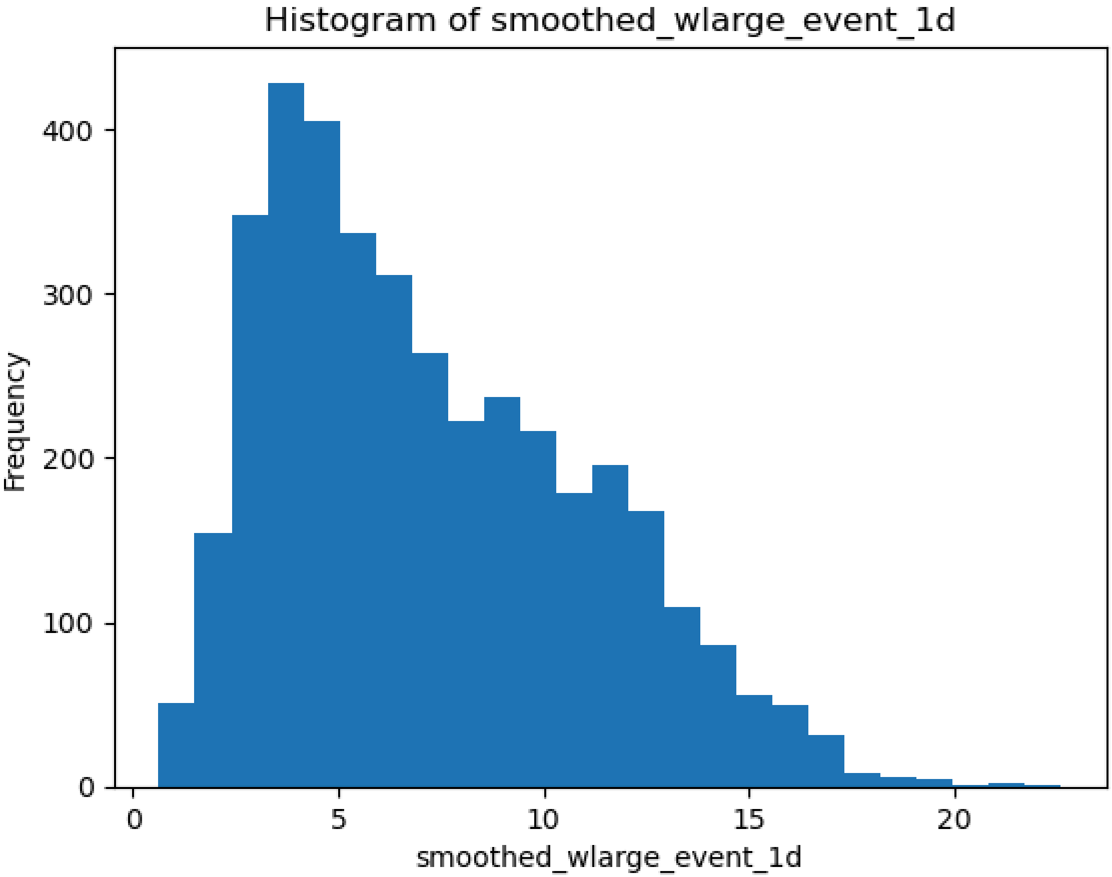


Histogram of smoothed_wvaccine_likely_politicians



Histogram of smoothed_wvaccine_likely_who





Removing Geo_Value from the data frame.

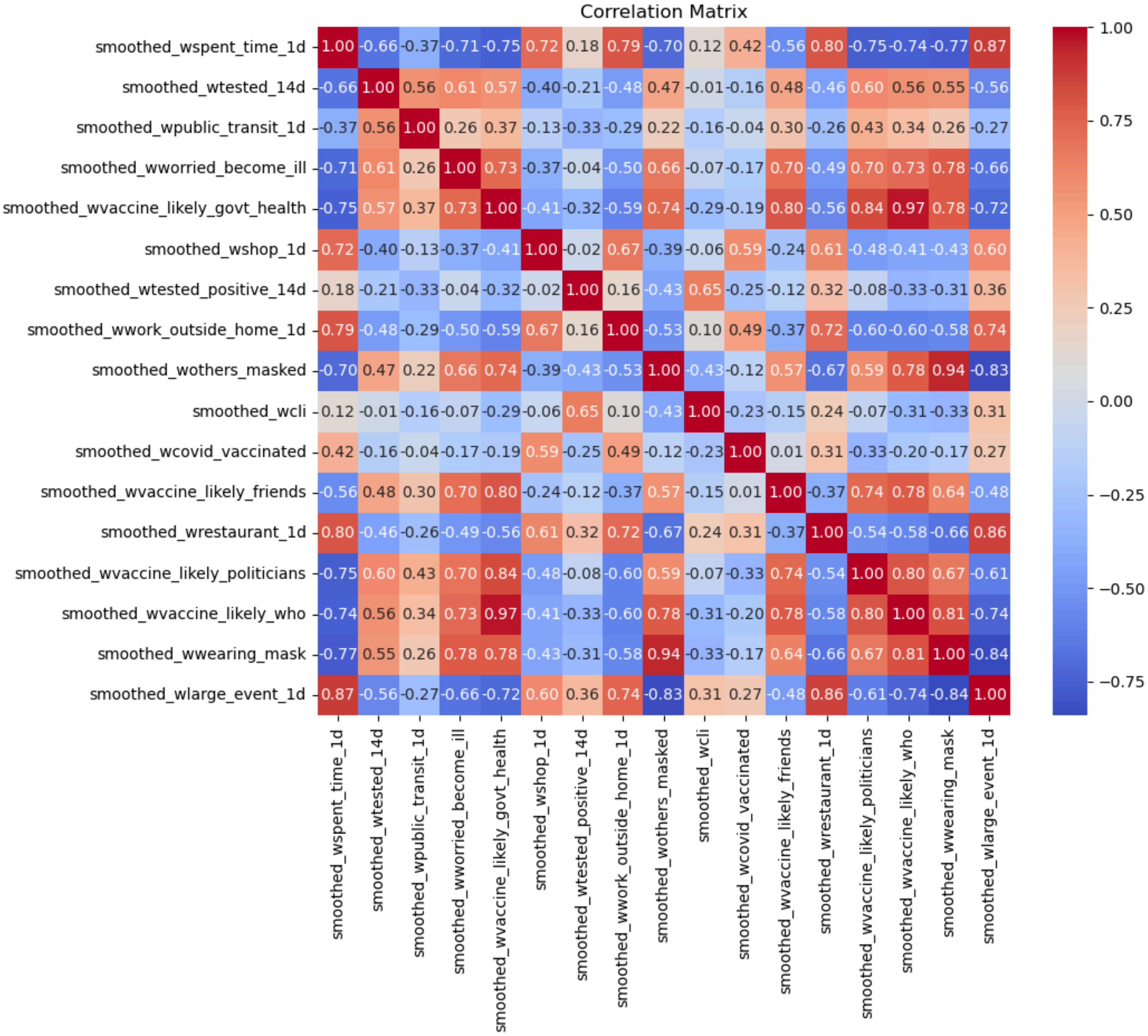
In [10]:

```
# Drop Geo_Value
if "geo_value" in df.columns:
    df = df.drop('geo_value', axis=1)

# print (df.columns)
```

Visualize a correlation matrix

In [12]:

```
correlation_matrix = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



Now, looking at dropping unreliable instances in the dataset. Set the threshold to 51% So anything less would be dropped.

In [14]:

```
# Testing the threshold feature here.
threshold = df.shape[1] * 0.51
print (threshold)
```

```
print(len(df))
df_test = df.dropna(thresh=threshold)
print(len(df_test))
```

```
8.67
3994
3881
```

In [15]:

```
df= df.dropna(thresh=threshold)

see_summary(df)
```

```
                                       count       mean        std        min  \
smoothed_wspent_time_1d               3873.0  29.540826   6.165547  14.410554
smoothed_wtested_14d                  3878.0  15.683821   5.090718   5.437303
smoothed_wpublic_transit_1d           3873.0   3.955646   5.141994   0.115497
smoothed_wworried_become_ill          3878.0  70.490606   5.395952  52.405818
smoothed_wvaccine_likely_govt_health  3877.0  32.970862   7.493678  17.410006
smoothed_wshop_1d                     3873.0  52.650528   4.830845  39.237833
smoothed_wtested_positive_14d         3881.0  17.417078   7.426650   1.548609
smoothed_wwork_outside_home_1d        3873.0  31.864066   5.510357  14.558697
smoothed_wothers_masked               3877.0  81.978574  13.484213  42.950717
smoothed_wcli                         3858.0   1.061560   0.508406   0.000000
smoothed_wcovid_vaccinated            3879.0  12.459743   6.141449   0.891041
smoothed_wvaccine_likely_friends      3878.0  35.636582   4.805895  22.622258
smoothed_wrestaurant_1d               3873.0  13.280293   5.520454   0.424278
smoothed_wvaccine_likely_politicians  3878.0  11.179896   3.627547   2.123555
smoothed_wvaccine_likely_who          3878.0  37.814370   8.040078  20.127120
smoothed_wwearing_mask                3878.0  92.455172   5.531985  74.543138
smoothed_wlarge_event_1d              3873.0   7.279319   3.842110   0.632336

                                            25%        50%        75%  \
smoothed_wspent_time_1d               25.064621  29.237196  33.608535
smoothed_wtested_14d                  11.804559  14.634947  18.766986
smoothed_wpublic_transit_1d            1.874536   2.548976   3.539643
smoothed_wworried_become_ill          67.226083  70.940729  74.369506
smoothed_wvaccine_likely_govt_health  26.898734  32.732026  38.387519
smoothed_wshop_1d                     49.263607  52.036982  55.867318
smoothed_wtested_positive_14d         11.797376  16.722884  22.346990
smoothed_wwork_outside_home_1d        28.047715  31.672672  35.472144
smoothed_wothers_masked               73.160600  87.850911  92.358267
smoothed_wcli                          0.692047   0.983577   1.354976
smoothed_wcovid_vaccinated             7.586523  11.557308  16.638127
smoothed_wvaccine_likely_friends      31.940948  35.328402  39.072705
smoothed_wrestaurant_1d                8.810386  13.094899  17.753224
smoothed_wvaccine_likely_politicians   8.426404  10.711590  13.530993
smoothed_wvaccine_likely_who          31.230199  38.211052  43.898167
smoothed_wwearing_mask                88.446697  94.878346  96.699905
smoothed_wlarge_event_1d               4.112238   6.486758   9.971153

                                            max  skewness   kurtosis
smoothed_wspent_time_1d               49.831174  0.252439  -0.233199
smoothed_wtested_14d                  34.951317  0.763690   0.102078
smoothed_wpublic_transit_1d           36.015469  3.803156  14.693356
smoothed_wworried_become_ill          85.446476 -0.423295   0.066312
smoothed_wvaccine_likely_govt_health  53.992167  0.212274  -0.746578
smoothed_wshop_1d                     67.597229  0.365532  -0.288753
smoothed_wtested_positive_14d         46.644291  0.461610  -0.072193
smoothed_wwork_outside_home_1d        49.848038  0.131769  -0.027514
smoothed_wothers_masked               98.935958 -1.099047   0.217337
smoothed_wcli                          3.356476  0.828985   1.018072
smoothed_wcovid_vaccinated            45.466215  0.827044   1.093374
smoothed_wvaccine_likely_friends      54.182794  0.309652  -0.216484
smoothed_wrestaurant_1d               28.340772  0.151007  -0.899545
smoothed_wvaccine_likely_politicians  25.197001  0.563509   0.044135
smoothed_wvaccine_likely_who          58.742213 -0.002269  -0.916316
smoothed_wwearing_mask                99.732673 -1.046291   0.123753
smoothed_wlarge_event_1d              22.619464  0.625615  -0.358648
```

Filling in missing values using a normal distribution if we can assume normality when examining kurtosis and skewness. If not using the median value instead of the mean to fill the value.

In [17]:

```
for col in df.columns:
    if df[col].skew() > 1 or df[col].skew() < -1 or df[col].kurtosis() > 3 or df[co
        if df[col].isnull().any():
            median = df[col].median()
            std = df[col].std()
            num_missing = df[col].isnull().sum()
            random_values = np.random.normal(loc=median, scale=std, size=num_missin
            df.loc[df[col].isnull(), col] = random_values
    else:
        if df[col].isnull().any():
            mean = df[col].mean()
            std = df[col].std()
            num_missing = df[col].isnull().sum()
            random_values = np.random.normal(loc=mean, scale=std, size=num_missing)
            df.loc[df[col].isnull(), col] = random_values

# print(df.isnull().sum()) # A check to ensure there are no missing values in the d
```

In [18]:

```
see_summary(df)
```

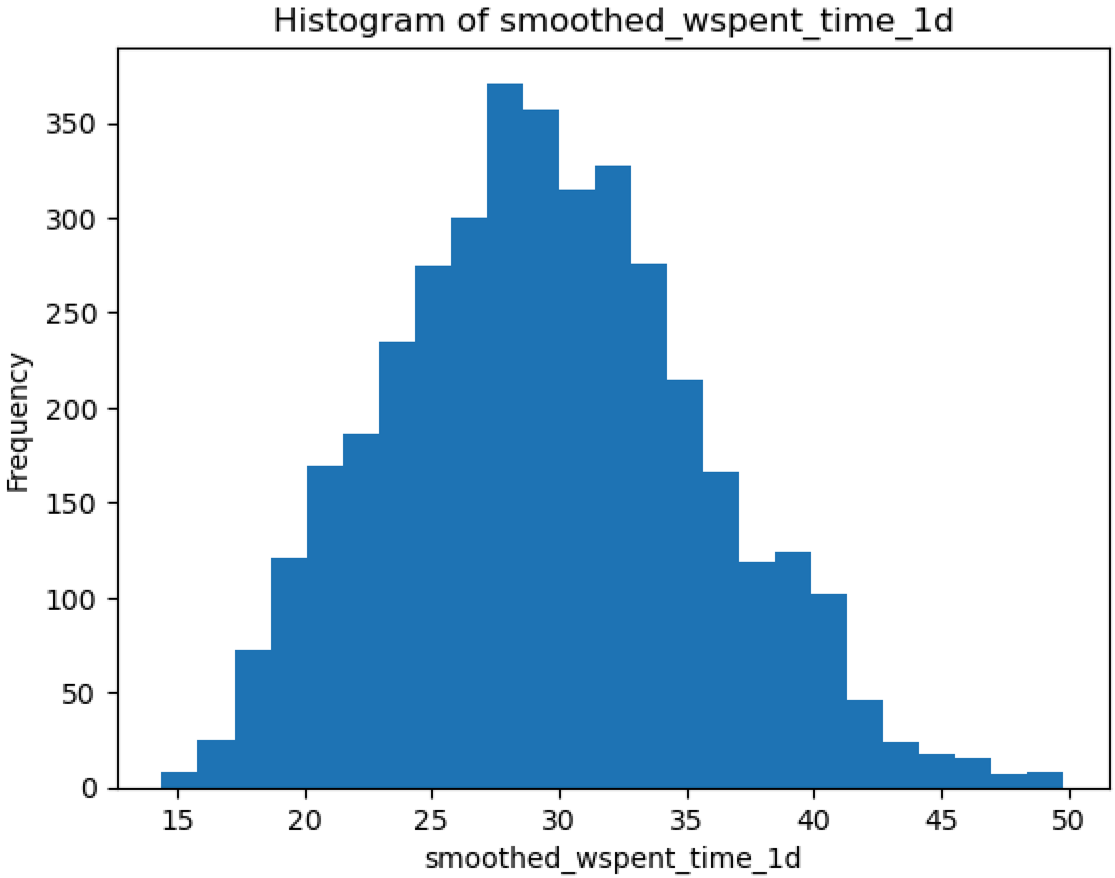
```
                                       count       mean        std        min  \
smoothed_wspent_time_1d               3881.0  29.547202   6.163782  14.410554   
smoothed_wtested_14d                  3881.0  15.683309   5.089229   5.437303   
smoothed_wpublic_transit_1d           3881.0   3.944756   5.145345  -7.289100   
smoothed_wworried_become_ill          3881.0  70.490366   5.396782  52.405818   
smoothed_wvaccine_likely_govt_health  3881.0  32.967158   7.492192  17.410006   
smoothed_wshop_1d                     3881.0  52.647410   4.830428  39.237833   
smoothed_wtested_positive_14d         3881.0  17.417078   7.426650   1.548609   
smoothed_wwork_outside_home_1d        3881.0  31.860306   5.510784  14.558697   
smoothed_wothers_masked               3881.0  81.975541  13.479502  42.950717   
smoothed_wcli                         3881.0   1.061778   0.507946   0.000000   
smoothed_wcovid_vaccinated            3881.0  12.459294   6.140299   0.891041   
smoothed_wvaccine_likely_friends      3881.0  35.640142   4.806781  22.622258   
smoothed_wrestaurant_1d               3881.0  13.274551   5.523319  -1.367283   
smoothed_wvaccine_likely_politicians  3881.0  11.180037   3.627597   2.123555   
smoothed_wvaccine_likely_who          3881.0  37.816214   8.038293  20.127120   
smoothed_wwearing_mask                3881.0  92.457160   5.530702  74.543138   
smoothed_wlarge_event_1d              3881.0   7.276345   3.840568   0.632336   

                                            25%        50%        75%  \
smoothed_wspent_time_1d               25.083013  29.239432  33.609422   
smoothed_wtested_14d                  11.811252  14.632757  18.766185   
smoothed_wpublic_transit_1d            1.867921   2.547657   3.539643   
smoothed_wworried_become_ill          67.225151  70.937099  74.370409   
smoothed_wvaccine_likely_govt_health  26.898734  32.732026  38.387066   
smoothed_wshop_1d                     49.256378  52.026334  55.867049   
smoothed_wtested_positive_14d         11.797376  16.722884  22.346990   
smoothed_wwork_outside_home_1d        28.044042  31.677526  35.472144   
smoothed_wothers_masked               73.160600  87.849436  92.358184   
smoothed_wcli                          0.693081   0.984630   1.355507   
smoothed_wcovid_vaccinated             7.587347  11.557308  16.635955   
smoothed_wvaccine_likely_friends      31.942055  35.329483  39.078546   
smoothed_wrestaurant_1d                8.810386  13.094899  17.739917   
smoothed_wvaccine_likely_politicians   8.426382  10.711300  13.531053   
smoothed_wvaccine_likely_who          31.231207  38.213084  43.898699   
smoothed_wwearing_mask                88.451826  94.879620  96.699987   
smoothed_wlarge_event_1d               4.112238   6.486758   9.968190   

                                            max  skewness   kurtosis  
smoothed_wspent_time_1d               49.831174  0.250941  -0.234041  
smoothed_wtested_14d                  34.951317  0.764061   0.103664  
smoothed_wpublic_transit_1d           36.015469  3.788282  14.654352  
smoothed_wworried_become_ill          85.446476 -0.422557   0.064193  
smoothed_wvaccine_likely_govt_health  53.992167  0.212818  -0.745504  
smoothed_wshop_1d                     67.597229  0.366952  -0.288830  
smoothed_wtested_positive_14d         46.644291  0.461610  -0.072193  
smoothed_wwork_outside_home_1d        49.848038  0.130526  -0.028440  
smoothed_wothers_masked               98.935958 -1.098680   0.217835  
smoothed_wcli                          3.356476  0.824678   1.013254  
smoothed_wcovid_vaccinated            45.466215  0.827245   1.094620  
smoothed_wvaccine_likely_friends      54.182794  0.308714  -0.219087  
smoothed_wrestaurant_1d               28.340772  0.146768  -0.890456  
smoothed_wvaccine_likely_politicians  25.197001  0.563594   0.042883  
smoothed_wvaccine_likely_who          58.742213 -0.002736  -0.915836  
smoothed_wwearing_mask                99.732673 -1.047027   0.125975  
smoothed_wlarge_event_1d              22.619464  0.626024  -0.355598  
```

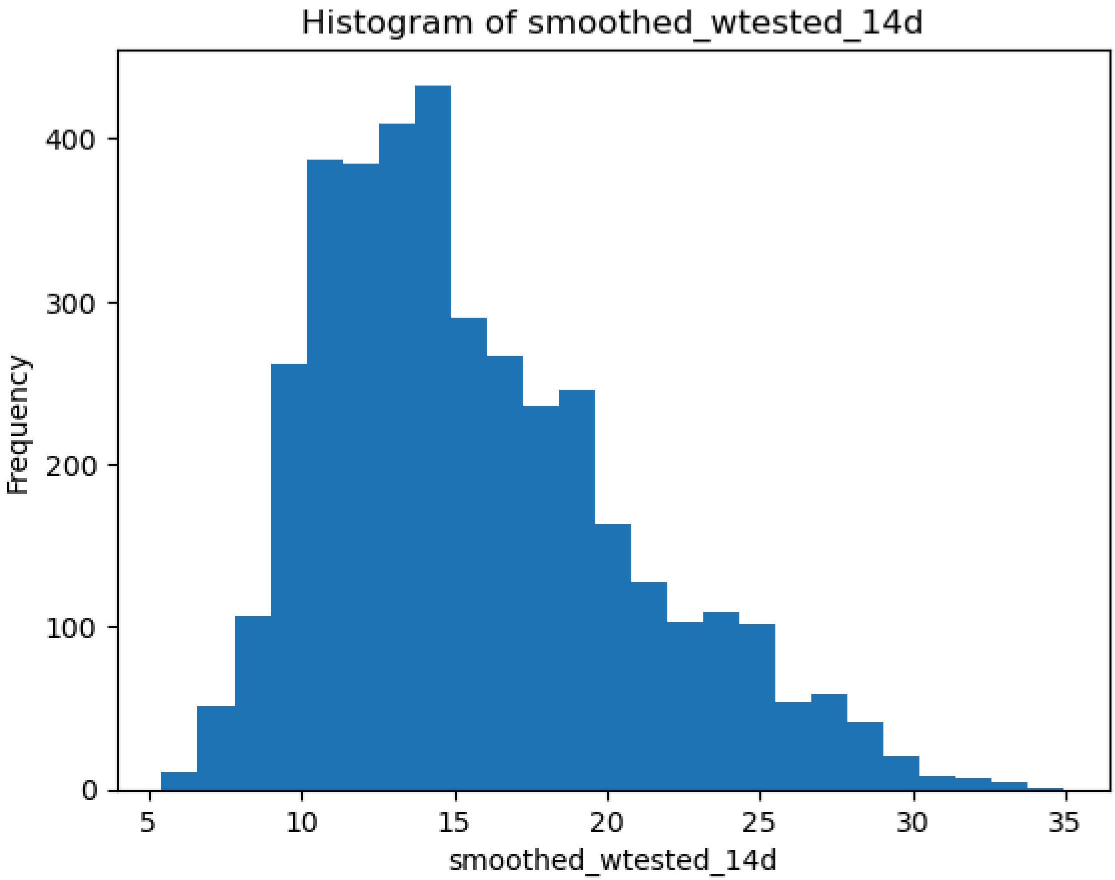
Now, transforming the columns to make them approximately normal for our models.

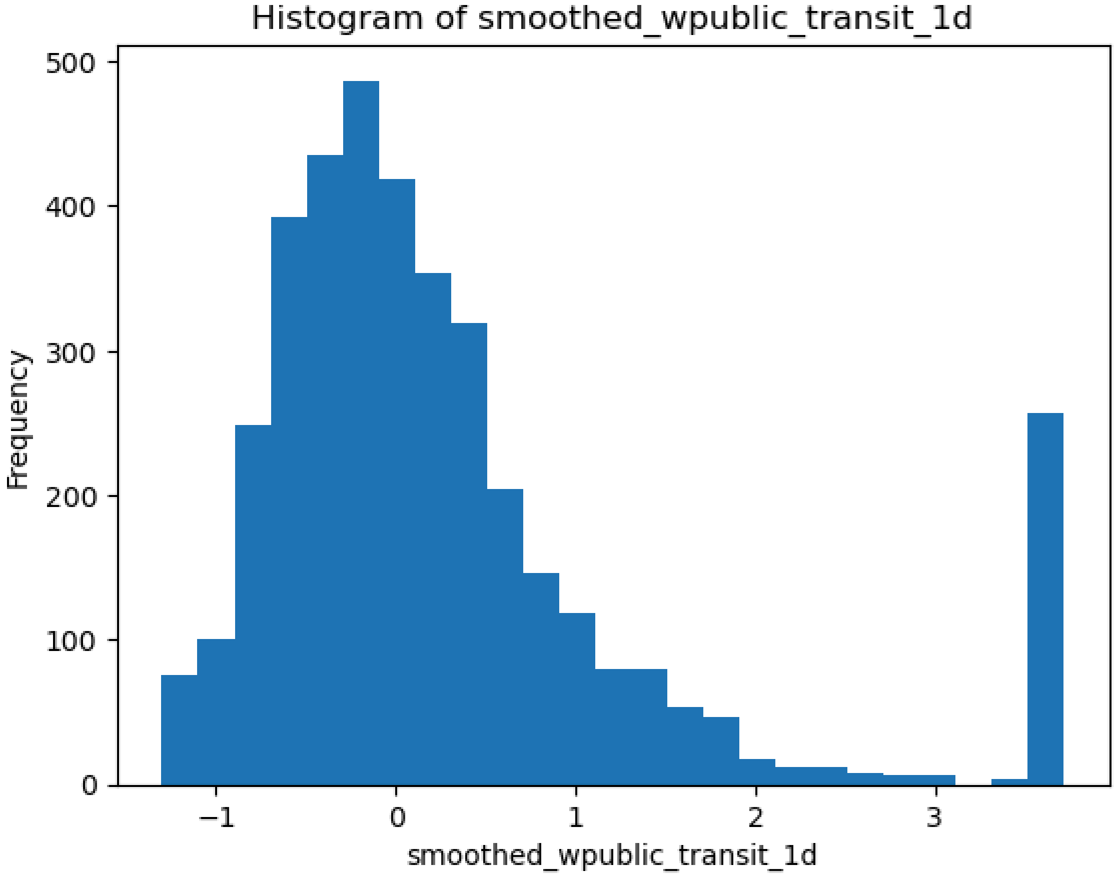
In [20]:
```
scalar = MinMaxScaler()
robust = RobustScaler()

for col in df.columns:
    if col == 'smoothed_wpublic_transit_1d':
        min_val = df['smoothed_wpublic_transit_1d'].min()
        shift_amount = abs(min_val) + 0.1 # Making the values non-negative before u
        df['smoothed_wpublic_transit_1d'] = np.log1p(df['smoothed_wpublic_transit_1
        df['smoothed_wpublic_transit_1d'] = winsorize(df['smoothed_wpublic_transit_
        df[['smoothed_wpublic_transit_1d']] = robust.fit_transform(df[['smoothed_wp
    elif df[col].skew() > 1 or df[col].skew() < -1:
        if df[col].kurtosis() > 3 or df[col].kurtosis() < 0:
            df[col] = np.log1p(df[col] - df[col].min() + 1)
        df[[col]] = scalar.fit_transform(df[[col]])

see_summary(df)
```

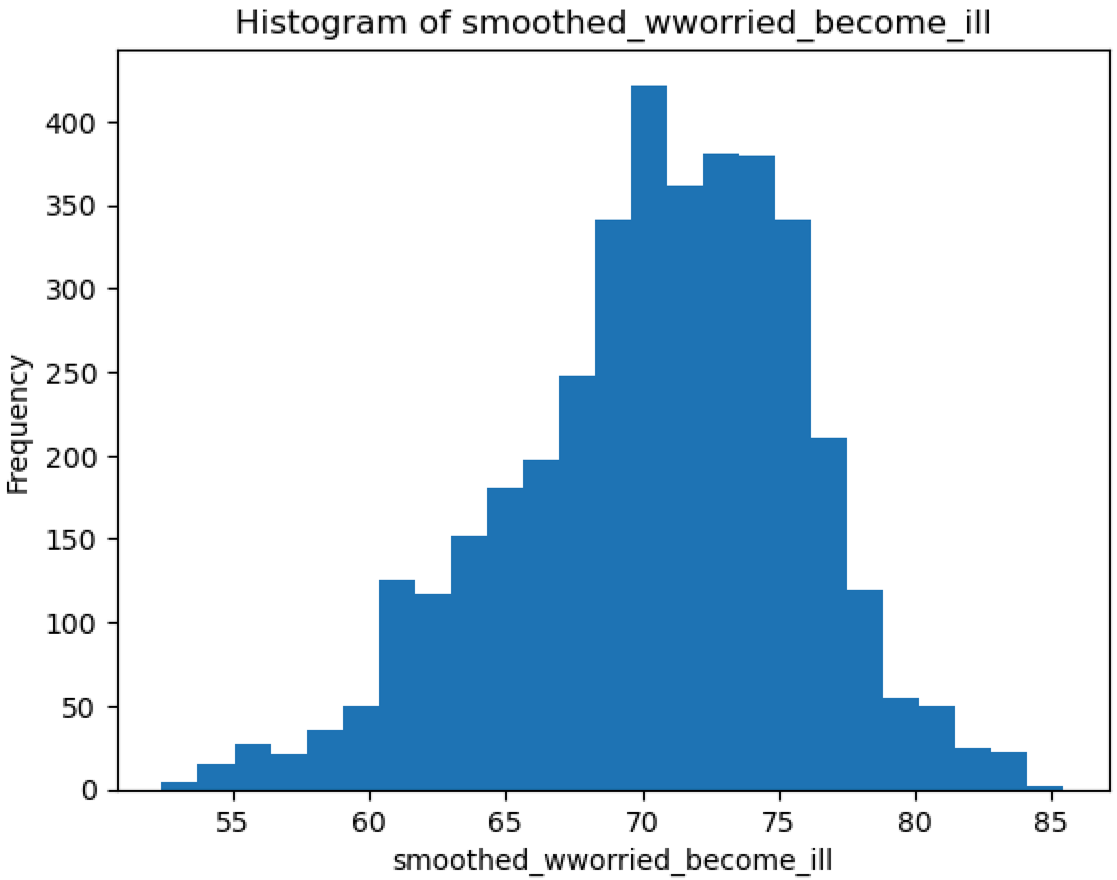
```
                                     count       mean       std        min  \
smoothed_wspent_time_1d             3881.0  29.547202  6.163782  14.410554   
smoothed_wtested_14d                3881.0  15.683309  5.089229   5.437303   
smoothed_wpublic_transit_1d         3881.0   0.300209  1.147643  -1.299343   
smoothed_wworried_become_ill        3881.0  70.490366  5.396782  52.405818   
smoothed_wvaccine_likely_govt_health  3881.0  32.967158  7.492192  17.410006   
smoothed_wshop_1d                   3881.0  52.647410  4.830428  39.237833   
smoothed_wtested_positive_14d       3881.0  17.417078  7.426650   1.548609   
smoothed_wwork_outside_home_1d      3881.0  31.860306  5.510784  14.558697   
smoothed_wothers_masked             3881.0   0.697056  0.240769   0.000000   
smoothed_wcli                       3881.0   1.061778  0.507946   0.000000   
smoothed_wcovid_vaccinated          3881.0  12.459294  6.140299   0.891041   
smoothed_wvaccine_likely_friends    3881.0  35.640142  4.806781  22.622258   
smoothed_wrestaurant_1d             3881.0  13.274551  5.523319  -1.367283   
smoothed_wvaccine_likely_politicians  3881.0  11.180037  3.627597   2.123555   
smoothed_wvaccine_likely_who        3881.0  37.816214  8.038293  20.127120   
smoothed_wwearing_mask              3881.0   0.711169  0.219563   0.000000   
smoothed_wlarge_event_1d            3881.0   7.276345  3.840568   0.632336   

                                          25%        50%        75%  \
smoothed_wspent_time_1d             25.083013  29.239432  33.609422   
smoothed_wtested_14d                11.811252  14.632757  18.766185   
smoothed_wpublic_transit_1d         -0.424979   0.000000   0.575021   
smoothed_wworried_become_ill        67.225151  70.937099  74.370409   
smoothed_wvaccine_likely_govt_health  26.898734  32.732026  38.387066   
smoothed_wshop_1d                   49.256378  52.026334  55.867049   
smoothed_wtested_positive_14d       11.797376  16.722884  22.346990   
smoothed_wwork_outside_home_1d      28.044042  31.677526  35.472144   
smoothed_wothers_masked              0.539604   0.801974   0.882509   
smoothed_wcli                        0.693081   0.984630   1.355507   
smoothed_wcovid_vaccinated           7.587347  11.557308  16.635955   
smoothed_wvaccine_likely_friends    31.942055  35.329483  39.078546   
smoothed_wrestaurant_1d              8.810386  13.094899  17.739917   
smoothed_wvaccine_likely_politicians   8.426382  10.711300  13.531053   
smoothed_wvaccine_likely_who        31.231207  38.213084  43.898699   
smoothed_wwearing_mask               0.552161   0.807339   0.879605   
smoothed_wlarge_event_1d             4.112238   6.486758   9.968190   

                                          max  skewness  kurtosis  
smoothed_wspent_time_1d             49.831174  0.250941 -0.234041  
smoothed_wtested_14d                34.951317  0.764061  0.103664  
smoothed_wpublic_transit_1d          3.718327  1.730247  2.763155  
smoothed_wworried_become_ill        85.446476 -0.422557  0.064193  
smoothed_wvaccine_likely_govt_health  53.992167  0.212818 -0.745504  
smoothed_wshop_1d                   67.597229  0.366952 -0.288830  
smoothed_wtested_positive_14d       46.644291  0.461610 -0.072193  
smoothed_wwork_outside_home_1d      49.848038  0.130526 -0.028440  
smoothed_wothers_masked              1.000000 -1.098680  0.217835  
smoothed_wcli                        3.356476  0.824678  1.013254  
smoothed_wcovid_vaccinated          45.466215  0.827245  1.094620  
smoothed_wvaccine_likely_friends    54.182794  0.308714 -0.219087  
smoothed_wrestaurant_1d             28.340772  0.146768 -0.890456  
smoothed_wvaccine_likely_politicians  25.197001  0.563594  0.042883  
smoothed_wvaccine_likely_who        58.742213 -0.002736 -0.915836  
smoothed_wwearing_mask               1.000000 -1.047027  0.125975  
smoothed_wlarge_event_1d            22.619464  0.626024 -0.355598  
```

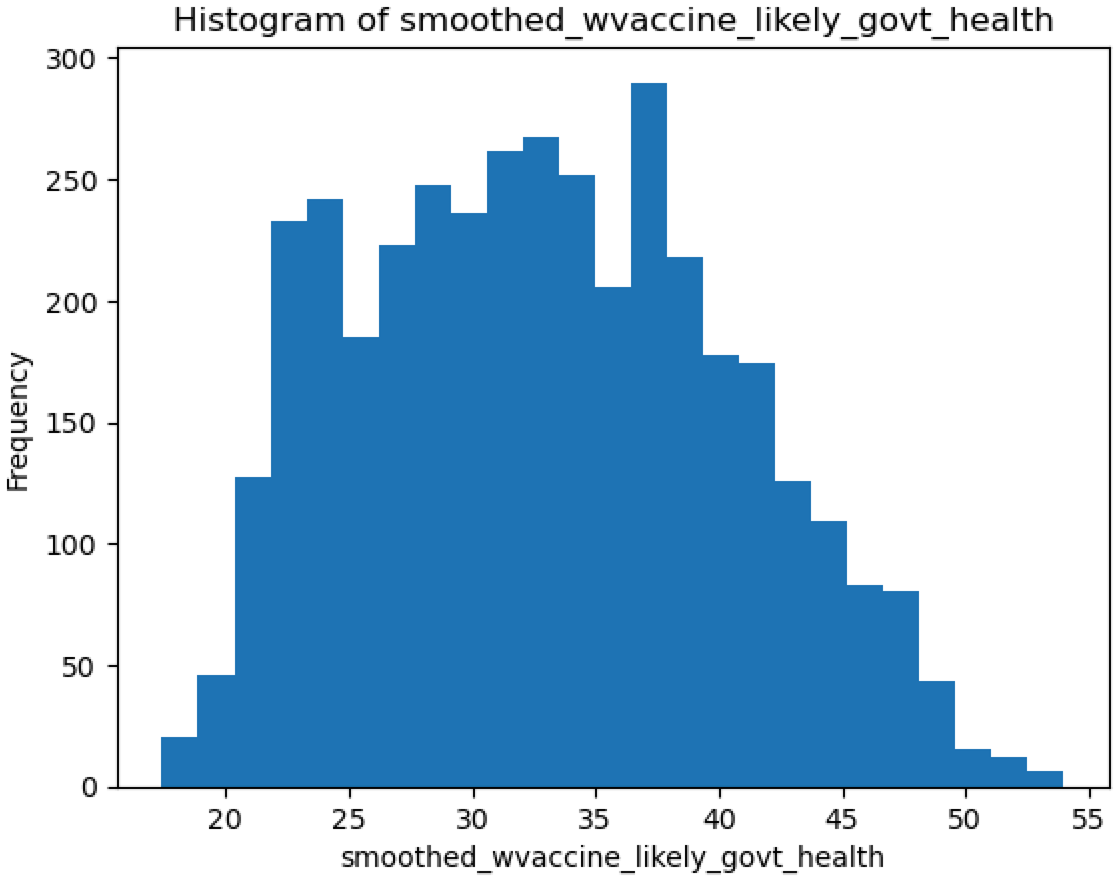
In [21]:
```
for col in df.columns:
    distribution(df[col], col)
```

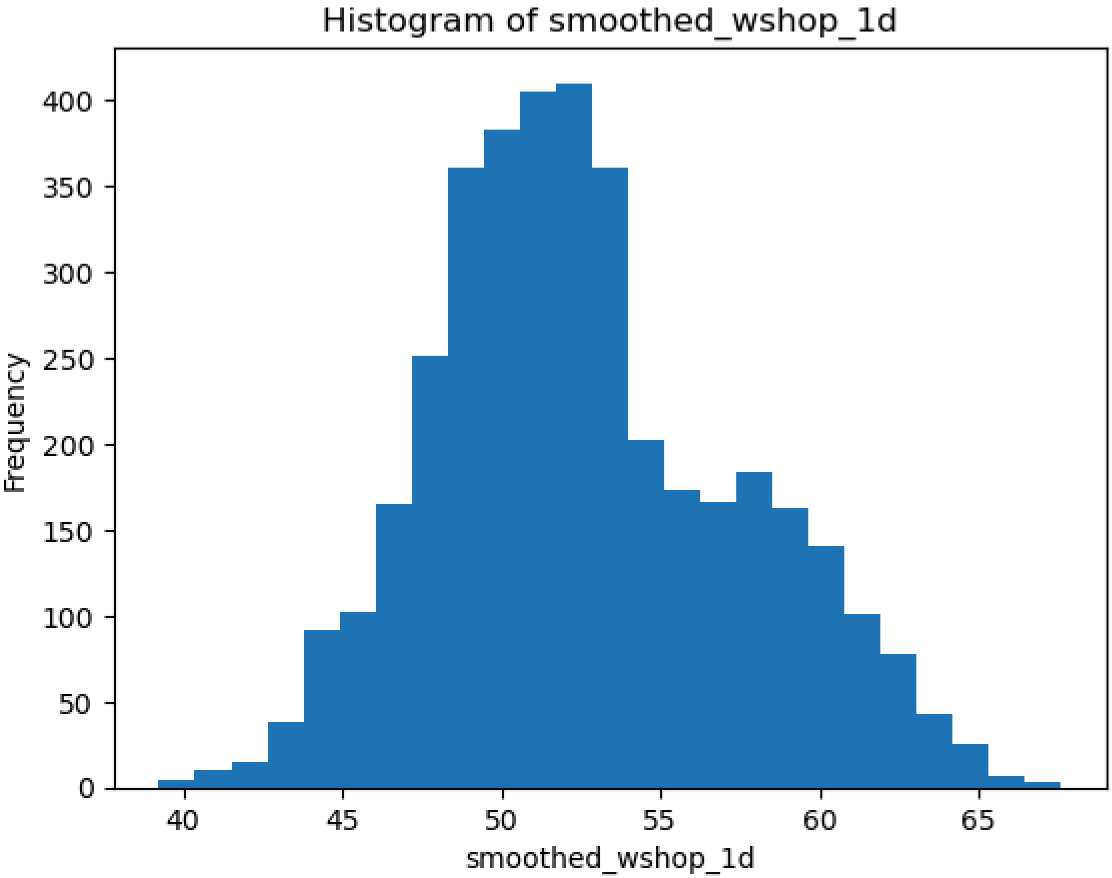


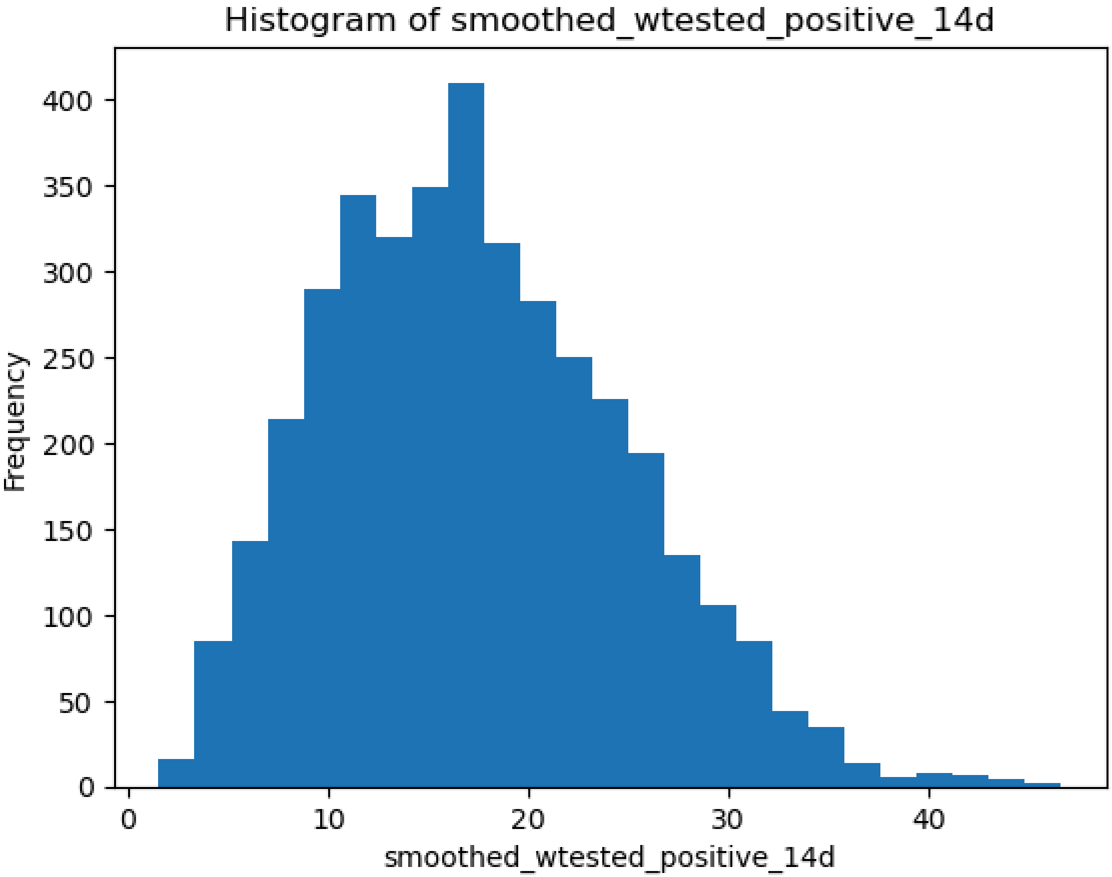

Histogram of smoothed_wtested_14d
Frequency
0
100
200
300
400
5
10
15
20
25
30
35
smoothed_wtested_14d

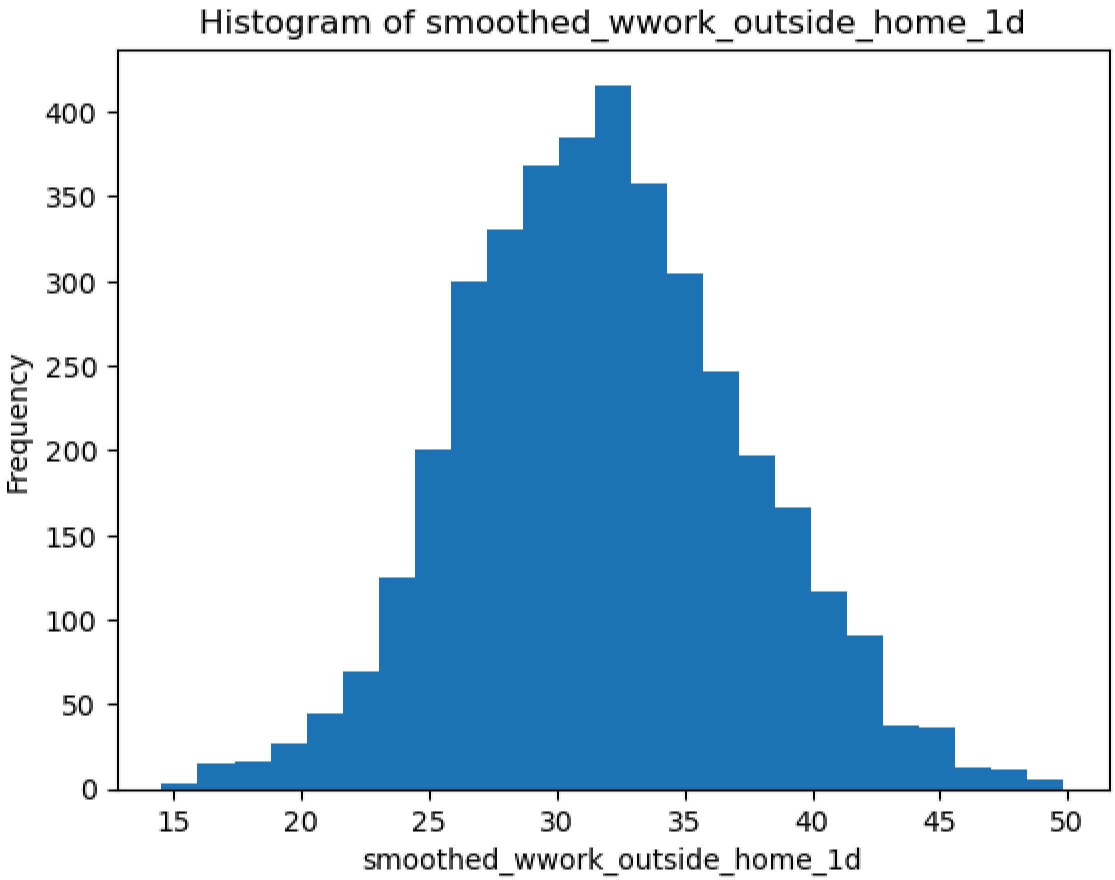

Histogram of smoothed_wpublic_transit_1d
Frequency
0
100
200
300
400
500
−1
0
1
2
3
smoothed_wpublic_transit_1d

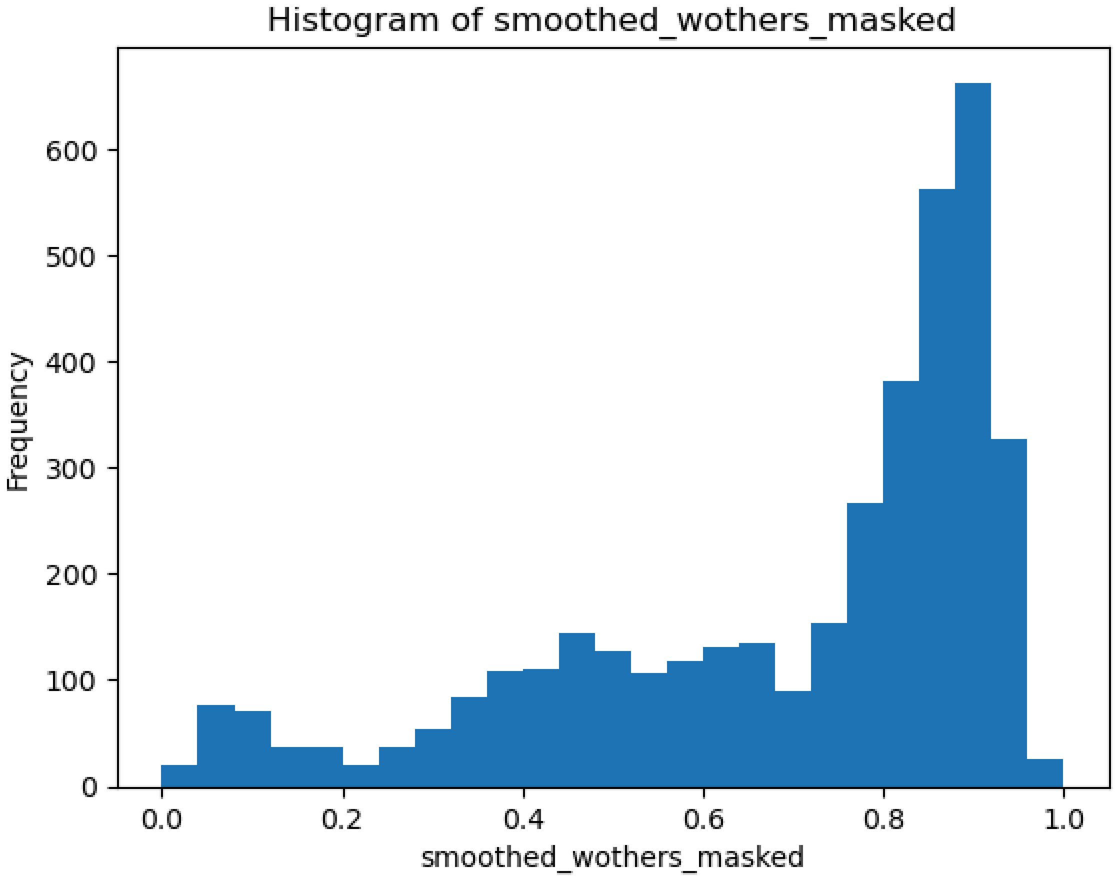

Histogram of smoothed_wworried_become_ill
Frequency
400
350
300
250
200
150
100
50
0
55
60
65
70
75
80
85
smoothed_wworried_become_ill

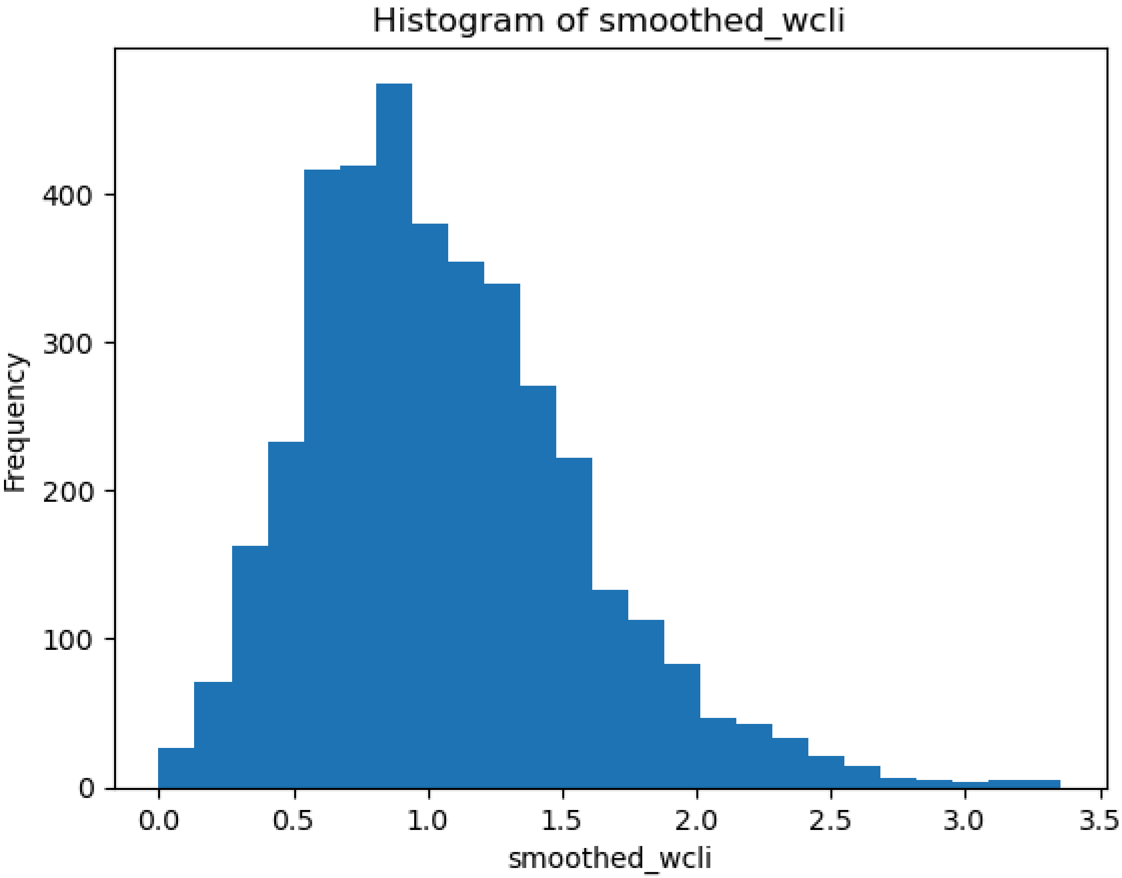

Histogram of smoothed_wvaccine_likely_govt_health
Frequency
300
250
200
150
100
50
0
20
25
30
35
40
45
50
55
smoothed_wvaccine_likely_govt_health

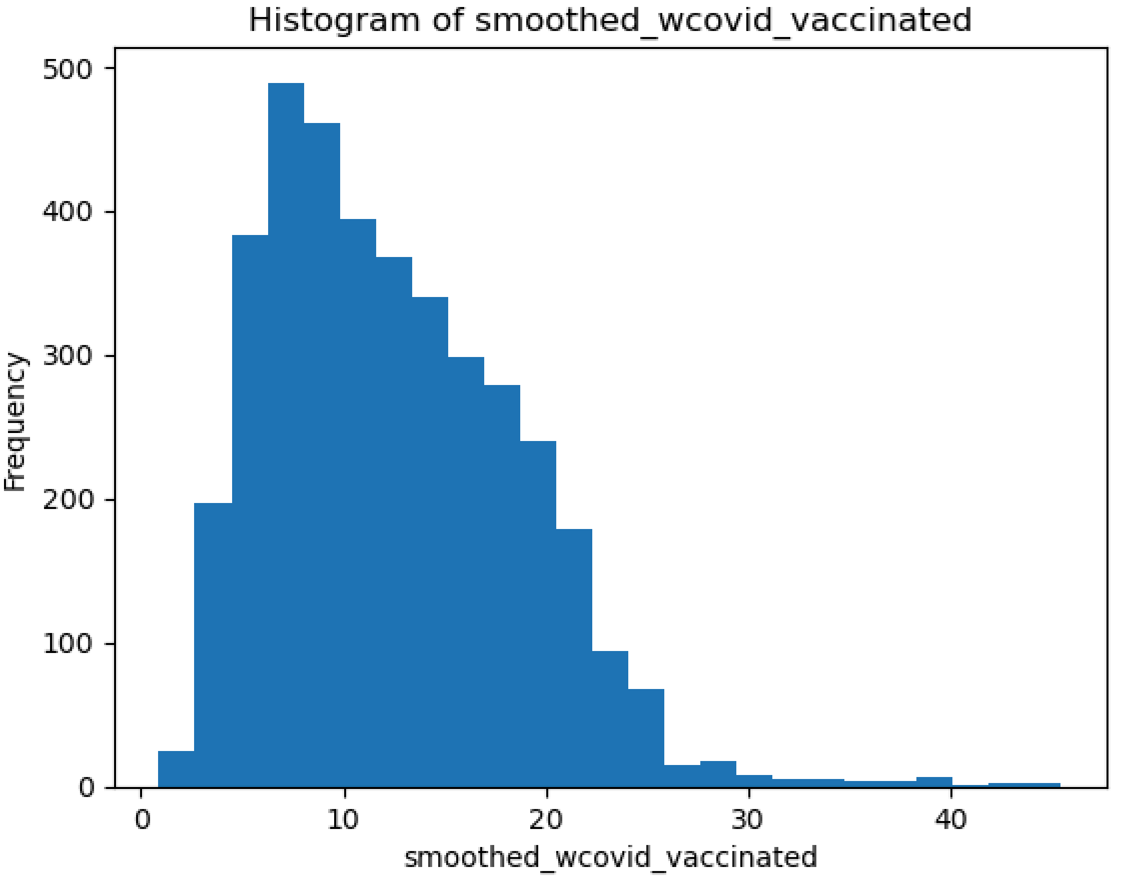

Histogram of smoothed_wshop_1d
Frequency
0
50
100
150
200
250
300
350
400
40
45
50
55
60
65
smoothed_wshop_1d

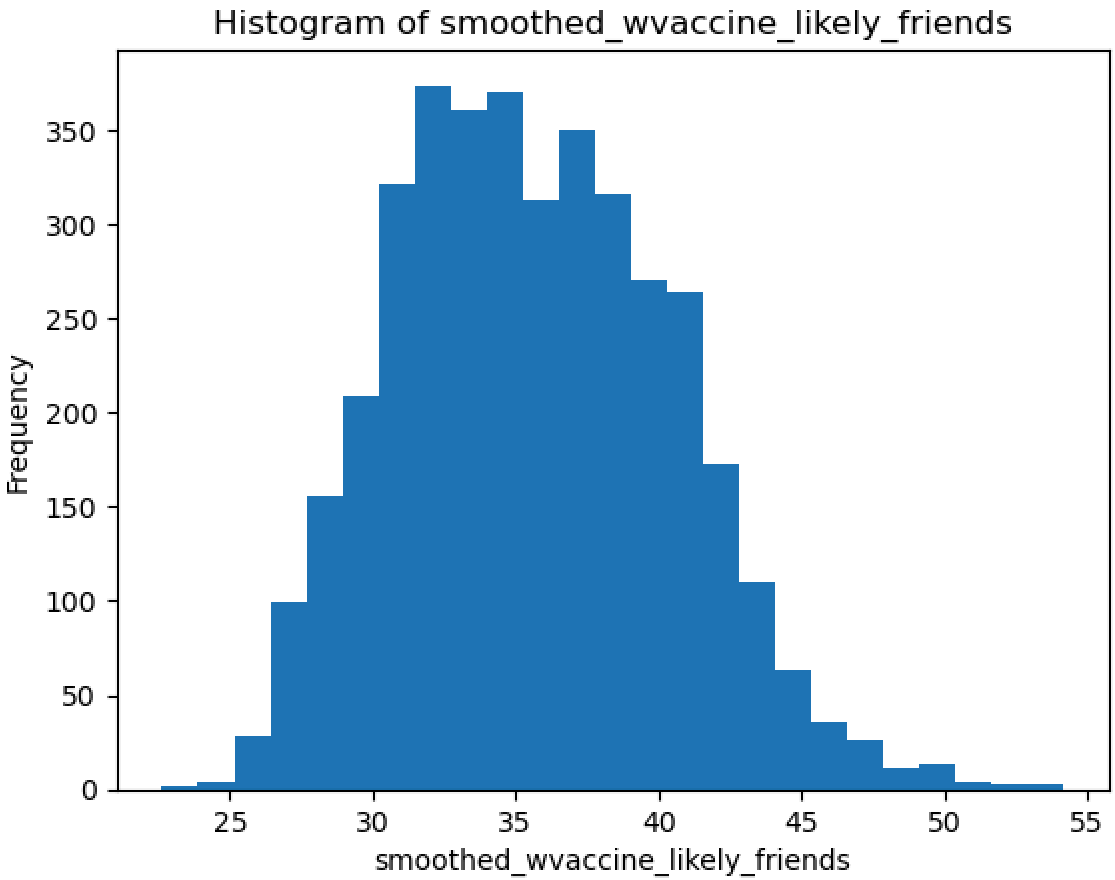

Histogram of smoothed_wtested_positive_14d
Frequency
0
50
100
150
200
250
300
350
400
0
10
20
30
40
smoothed_wtested_positive_14d

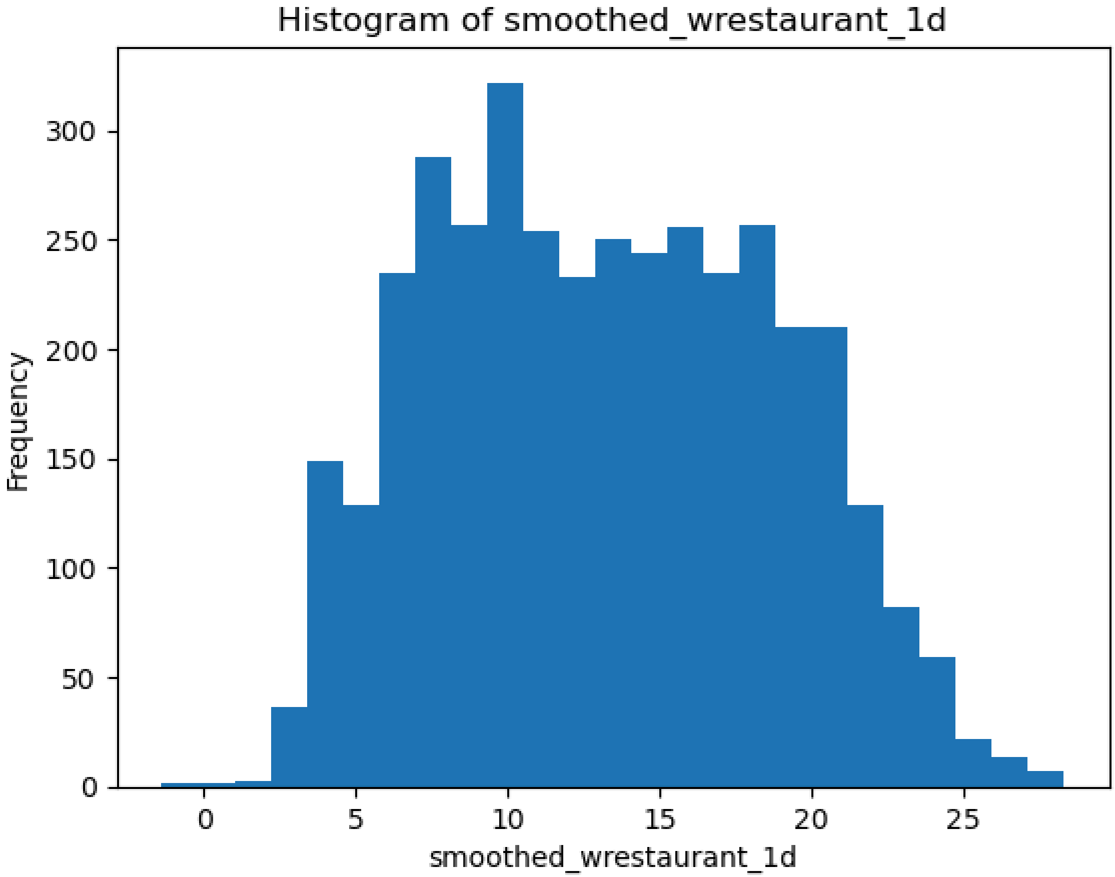

Histogram of smoothed_wwork_outside_home_1d
Frequency
0
50
100
150
200
250
300
350
400
15
20
25
30
35
40
45
50
smoothed_wwork_outside_home_1d

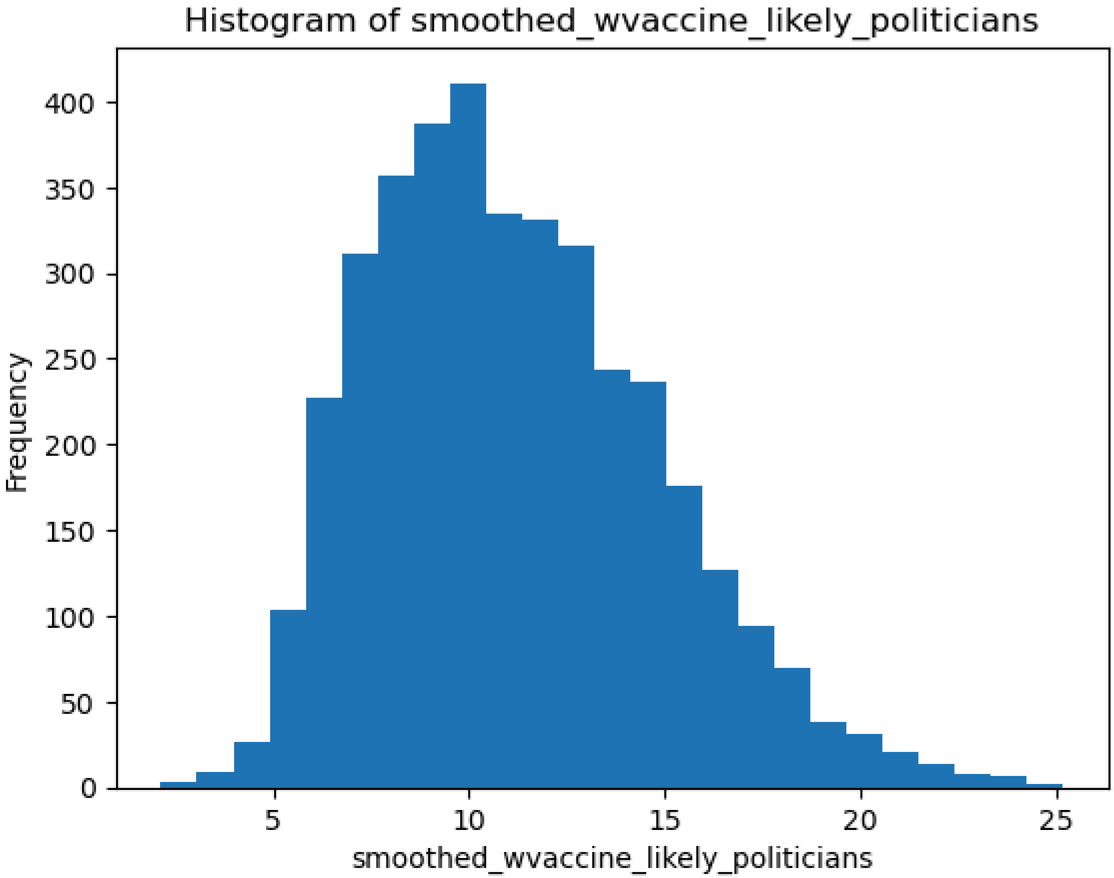

Histogram of smoothed_wothers_masked
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
smoothed_wothers_masked

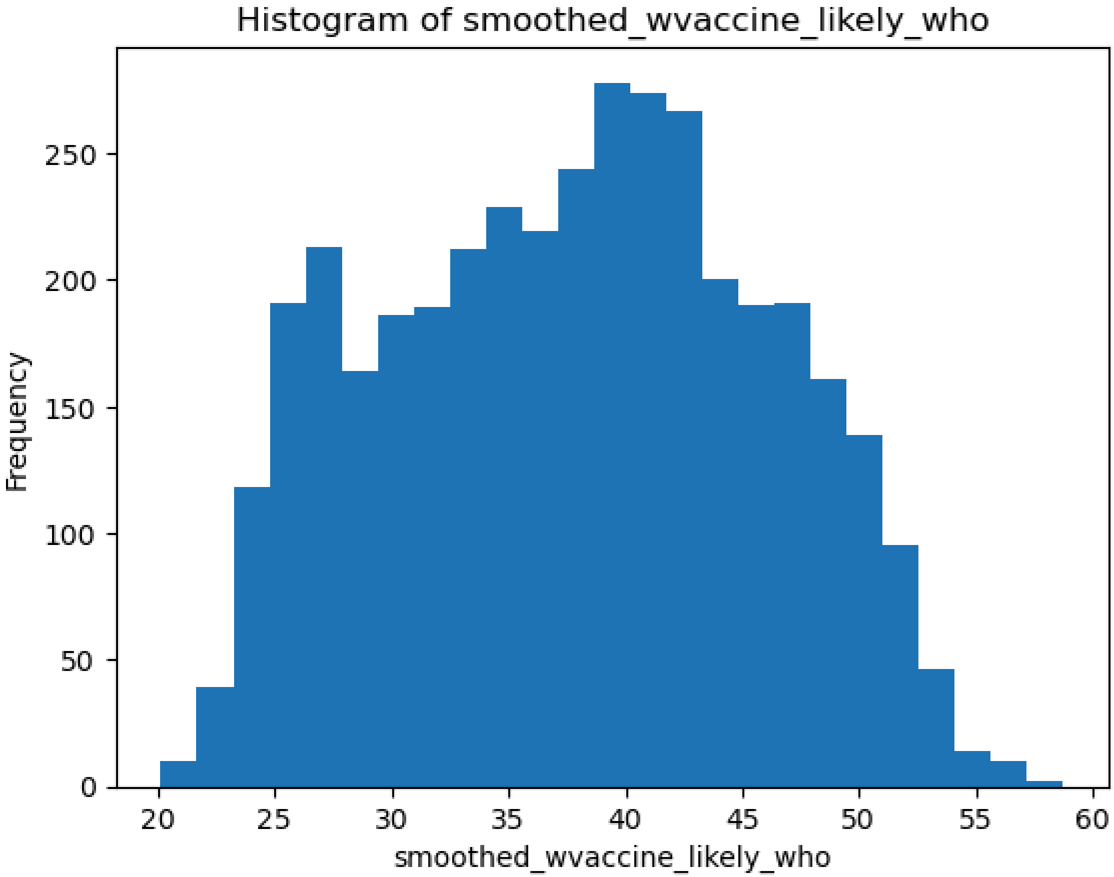

Histogram of smoothed_wcli
Frequency
0
100
200
300
400
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
smoothed_wcli

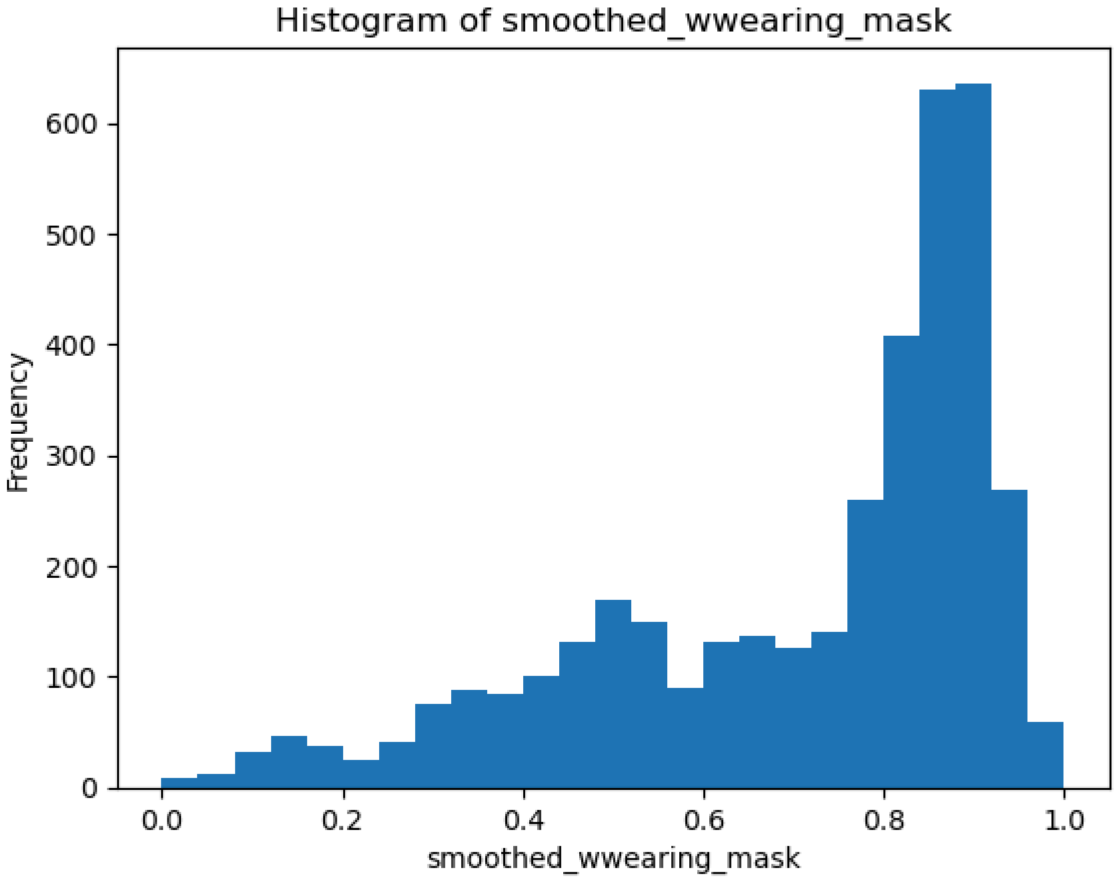

Histogram of smoothed_wcovid_vaccinated
Frequency
0
100
200
300
400
500
0
10
20
30
40
smoothed_wcovid_vaccinated

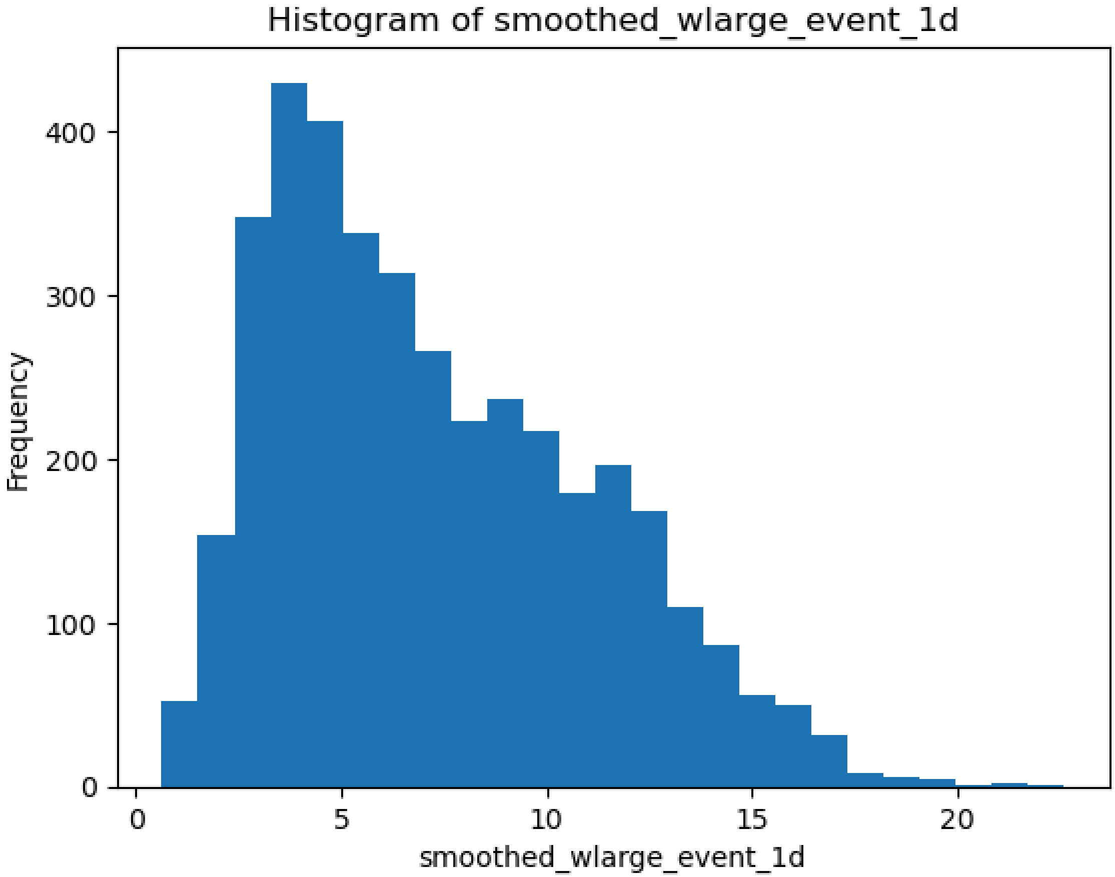

Histogram of smoothed_wvaccine_likely_friends
Frequency
0
50
100
150
200
250
300
350
25
30
35
40
45
50
55
smoothed_wvaccine_likely_friends


Histogram of smoothed_wrestaurant_1d
Frequency
0
50
100
150
200
250
300
0
5
10
15
20
25
smoothed_wrestaurant_1d


Histogram of smoothed_wvaccine_likely_politicians
Frequency
0
50
100
150
200
250
300
350
400
5
10
15
20
25
smoothed_wvaccine_likely_politicians

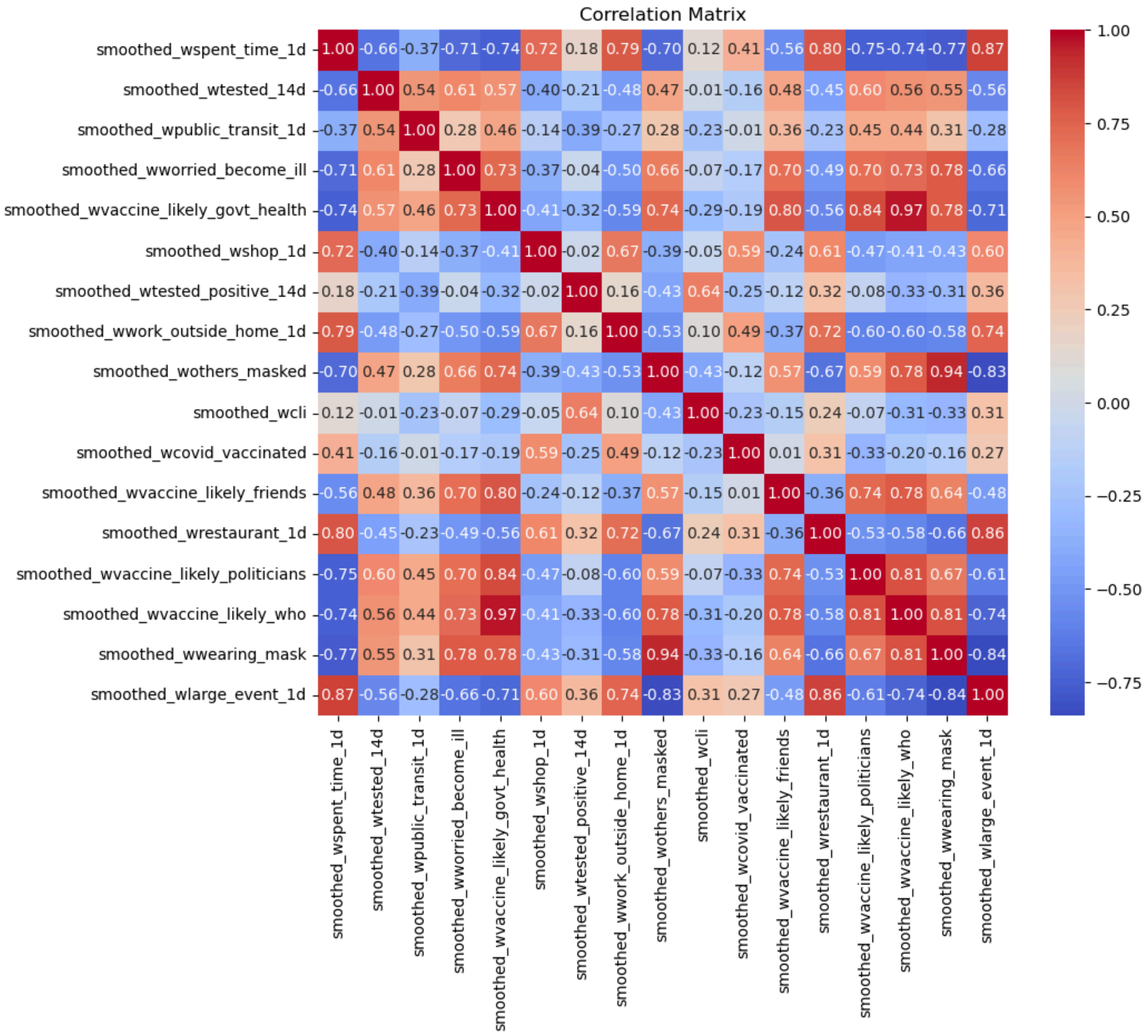

Histogram of smoothed_wvaccine_likely_who
Frequency
0
50
100
150
200
250
20
25
30
35
40
45
50
55
60
smoothed_wvaccine_likely_who


Histogram of smoothed_wwearing_mask
Frequency
0
100
200
300
400
500
600
0.0
0.2
0.4
0.6
0.8
1.0
smoothed_wwearing_mask


Histogram of smoothed_wlarge_event_1d
Frequency
0
100
200
300
400
0
5
10
15
20
smoothed_wlarge_event_1d

In [22]:

```
correlation_matrix = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



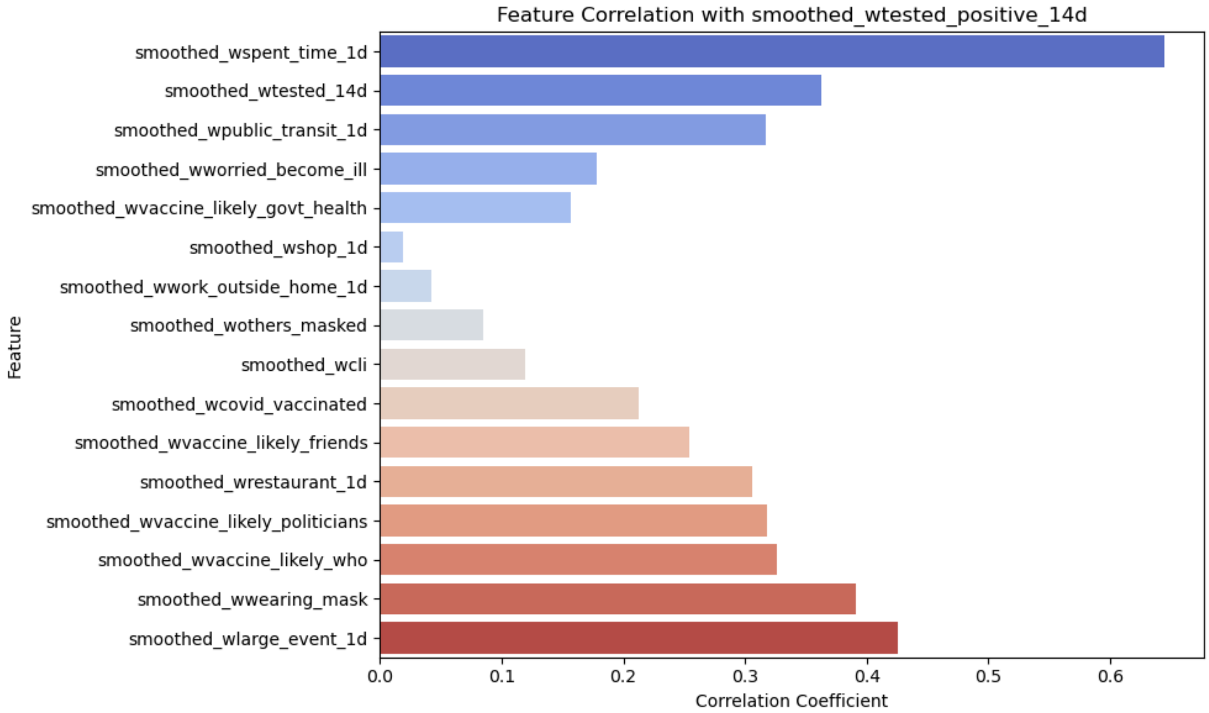
In [23]:

```
# Calculate VIF for each feature
vif = pd.DataFrame({
    "Feature": df.columns,
    "VIF": [variance_inflation_factor(df.values, i) for i in range(df.shape[1])]
})

print("\nVariance Inflation Factor (VIF):")
print(vif.sort_values(by="VIF", ascending=False))

# Calculate the Correlation for each Target Variable.
print('\nCorrelation against first target variable\n')
correlations_q1 = correlation_matrix['smoothed_wtested_positive_14d'].drop('smoothe

correlations_q1_sorted = correlations_q1.sort_values(ascending=False)
print(correlations_q1_sorted)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=correlations_q1_sorted.abs().values, y=correlations_q1.index, palette
plt.title('Feature Correlation with smoothed_wtested_positive_14d')
plt.xlabel('Correlation Coefficient')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

print('\nCorrelation against second target variable\n')
correlations_q2 = correlation_matrix['smoothed_wcovid_vaccinated'].drop('smoothed_w

correlations_q2_sorted = correlations_q2.sort_values(ascending=False)
print(correlations_q2_sorted)

plt.figure(figsize=(10, 6))
sns.barplot(x=correlations_q2_sorted.abs().values, y=correlations_q2.index, palette
plt.title('Feature Correlation with smoothed_wtested_positive_14d')
plt.xlabel('Correlation Coefficient')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

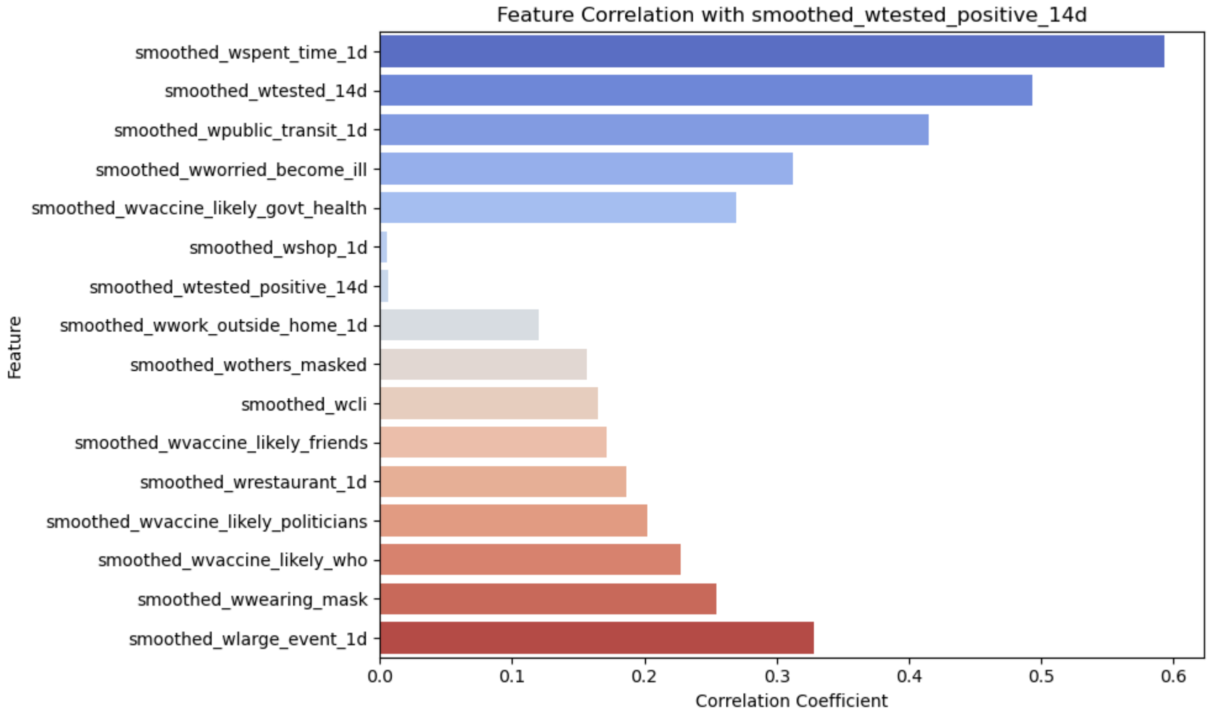
```
Variance Inflation Factor (VIF):
                                 Feature         VIF
14          smoothed_wvaccine_likely_who  487.872251
4   smoothed_wvaccine_likely_govt_health  459.513086
3           smoothed_wworried_become_ill  451.506093
5                      smoothed_wshop_1d  332.463854
11      smoothed_wvaccine_likely_friends  224.904295
0                smoothed_wspent_time_1d  213.990810
15                smoothed_wwearing_mask  171.901531
7         smoothed_wwork_outside_home_1d  116.478312
8                smoothed_wothers_masked  116.189655
13  smoothed_wvaccine_likely_politicians   54.870675
16              smoothed_wlarge_event_1d   52.252252
12               smoothed_wrestaurant_1d   30.352951
1                   smoothed_wtested_14d   27.270011
6          smoothed_wtested_positive_14d   17.708139
9                          smoothed_wcli   11.124711
10            smoothed_wcovid_vaccinated   10.456908
2            smoothed_wpublic_transit_1d    2.071089

Correlation against first target variable

smoothed_wcli                           0.644866
smoothed_wlarge_event_1d                0.362287
smoothed_wrestaurant_1d                 0.316785
smoothed_wspent_time_1d                 0.178562
smoothed_wwork_outside_home_1d          0.157035
smoothed_wshop_1d                      -0.019185
smoothed_wworried_become_ill           -0.042193
smoothed_wvaccine_likely_politicians   -0.084647
smoothed_wvaccine_likely_friends       -0.119244
smoothed_wtested_14d                   -0.212765
smoothed_wcovid_vaccinated             -0.254523
smoothed_wwearing_mask                 -0.305986
smoothed_wvaccine_likely_govt_health   -0.317712
smoothed_wvaccine_likely_who           -0.326254
smoothed_wpublic_transit_1d            -0.391650
smoothed_wothers_masked                -0.425991
Name: smoothed_wtested_positive_14d, dtype: float64
```

```
C:\Users\devon\AppData\Local\Temp\ipykernel_5492\347997800.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=correlations_q1_sorted.abs().values, y=correlations_q1.index, palett
e='coolwarm')
```



```
Correlation against second target variable

smoothed_wshop_1d                       0.593360
smoothed_wwork_outside_home_1d          0.493420
smoothed_wspent_time_1d                 0.414895
smoothed_wrestaurant_1d                 0.312001
smoothed_wlarge_event_1d                0.269740
smoothed_wvaccine_likely_friends        0.005683
smoothed_wpublic_transit_1d            -0.006584
smoothed_wothers_masked                -0.120322
smoothed_wtested_14d                   -0.156850
smoothed_wwearing_mask                 -0.164796
smoothed_wworried_become_ill           -0.171622
smoothed_wvaccine_likely_govt_health   -0.186620
smoothed_wvaccine_likely_who           -0.201994
smoothed_wcli                          -0.227072
smoothed_wtested_positive_14d          -0.254523
smoothed_wvaccine_likely_politicians   -0.327793
Name: smoothed_wcovid_vaccinated, dtype: float64
```

```
C:\Users\devon\AppData\Local\Temp\ipykernel_5492\347997800.py:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=correlations_q2_sorted.abs().values, y=correlations_q2.index, palett
e='coolwarm')
```



The Data is preprocessed to a point where most appears normally distributed. You can see from the correlation matrix and VIF at the end we have some multicollinear relationships. I did not remove them because some of our models would do it themselves.

In [25]:
```
# df.to_csv("Devonte_Transformed_dataset.csv", index=False)

#For Question 1:
# Drop smoothed wtested positive 14d
x1 = df.drop('smoothed_wtested_positive_14d', axis=1)

# dataframe with only the target label.
y1 = df['smoothed_wtested_positive_14d']

x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=0.2, rand

#For Question 2:
# Drop smoothed wcovid vaccinated
x2 = df.drop('smoothed_wcovid_vaccinated', axis=1)

# dataframe with only the target label.
y2 = df['smoothed_wcovid_vaccinated']

x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=0.2, rand
```

Functions for Learning

In [27]:
```
def linear_model(x_train, x_test, y_train, y_test):
    lr_model = LinearRegression()
    lr_model.fit(x_train, y_train)

    y_pred_lr = lr_model.predict(x_test)
    print("Linear Regression R²:", r2_score(y_test, y_pred_lr))
    print("Linear Regression RMSE:", root_mean_squared_error(y_test, y_pred_lr))

# Want to see if there is much difference in the baseline when using cross-validati
def cross_validated_lm(x, y, split): # x: feature variables, y: target variables, s
    lm = LinearRegression()
    kf = KFold(n_splits=split, shuffle=True, random_state=42)

    r2_scores = cross_val_score(lm, x, y, cv=kf, scoring="r2")
    rmse_scores = -cross_val_score(lm, x, y, cv=kf, scoring="neg_root_mean_squared_

    print(f"Mean R²: {r2_scores.mean():.4f}")
    print(f"Mean RMSE: {rmse_scores.mean():.4f}")

def cross_validated_Regularization(model_name, x, y, alpha, split): # model_name: L

    if model_name == ElasticNet:
        model = model_name(alpha=alpha, l1_ratio=0.5)
    else:
        model = model_name(alpha=alpha)

    kf = KFold(n_splits=split, shuffle=True, random_state=42)

    r2_scores = cross_val_score(model, x, y, cv=kf, scoring="r2")
    rmse_scores = -cross_val_score(model, x, y, cv=kf, scoring="neg_root_mean_squar

    return (model_name, alpha, r2_scores.mean(), rmse_scores.mean())

def best_model_description (model_name, x_train, x_test, y_train, y_test, alpha):
    if model_name == ElasticNet:
        model = model_name(alpha=alpha, l1_ratio=0.5)
    else:
        model = model_name(alpha=alpha)

    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    r2 = r2_score(y_test, y_pred)
    rmse = root_mean_squared_error(y_test, y_pred)

    coefficients = model.coef_
    feature_weights = pd.Series(coefficients, index=x_train.columns)

    print('Evaluation Data:\nR^2: ' + str(r2) + '\nRMSE: ' + str(rmse) + '\n')

    print(feature_weights.sort_values(ascending=False))

def best_model_with_pca(model_name, x_train, x_test, y_train, y_test, alpha, pca_mo
    if model_name == ElasticNet:
        model = model_name(alpha=alpha, l1_ratio=0.5)
    else:
        model = model_name(alpha=alpha)

    # Fit model and predict
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    # Metrics
    r2 = r2_score(y_test, y_pred)
    rmse = root_mean_squared_error(y_test, y_pred)

    print(f"R^2 Score: {r2:.4f}")
    print(f"RMSE: {rmse:.4f}")

    # Feature importance on PCA components
    coefficients = model.coef_
    pc_names = [f"PC{i+1}" for i in range(len(coefficients))]
    feature_weights = pd.Series(coefficients, index=pc_names)

    print("\nFeature importances (sorted by weight):\n")
    print(feature_weights.sort_values(ascending=False))

    return model, feature_weights
```

Baseline Models:

In [29]:

```
print ("Baseline 1 Target (smoothed wtested positive 14d)")
linear_model(x1_train, x1_test, y1_train, y1_test)

print ("Baseline 2 Target (smoothed wcovid vaccinated)")
linear_model(x2_train, x2_test, y2_train, y2_test)
```

```
Baseline 1 Target (smoothed wtested positive 14d)
Linear Regression R²: 0.6117156036694975
Linear Regression RMSE: 4.612358910291211
Baseline 2 Target (smoothed wcovid vaccinated)
Linear Regression R²: 0.582592164522707
Linear Regression RMSE: 3.8560399591679344
```

Baseline Model using Cross-Validation

In [31]:

```
print ("Baseline 1 Target (smoothed wtested positive 14d)")
cross_validated_lm(x1, y1, 10)
print ("Baseline 2 Target (smoothed wcovid vaccinated)")
cross_validated_lm(x2, y2, 10)
```

```
Baseline 1 Target (smoothed wtested positive 14d)
Mean R²: 0.6286
Mean RMSE: 4.5189
Baseline 2 Target (smoothed wcovid vaccinated)
Mean R²: 0.5377
Mean RMSE: 4.1680
```

Ridge and Lasso Regression Models

In [33]:

```
alphas = [0.001, 0.01, 1, 10, 100] # Tuning the penalty for these models.
models = [Lasso, Ridge, ElasticNet] # Determing which regression to use.

best_model = None
best_alpha = None
best_r2 = None
best_rmse = float('inf')
```

```
print ("Target (smoothed wtested positive 14d)")
for a in alphas:
    for m in models:
        model, alpha, r2, rmse = cross_validated_Regularization(m, x1_train, y1_tra
        if rmse < best_rmse:
            best_model = model
            best_alpha = alpha
            best_r2 = r2
            best_rmse = rmse

print(f"The best model is: {best_model. __name__} with an alpha of {best_alpha}. \n

print('\nThe coefficients for this model are:\n')

best_model_description (best_model , x1_train, x1_test, y1_train, y1_test, best_alp

best_model_2 = None
best_alpha_2 = None
best_r2_2 = None
best_rmse_2 = float('inf')

print("\nTarget (smoothed wcovid vaccinated)")
for a in alphas:
    for m in models:
        model, alpha, r2, rmse = cross_validated_Regularization(m, x2_train, y2_tra
        if rmse < best_rmse_2:
            best_model_2 = model
            best_alpha_2 = alpha
            best_r2_2 = r2
            best_rmse_2 = rmse

print(f"The best model is: {best_model_2. __name__} with an alpha of {best_alpha_2}

print('\nThe coefficients for this model are:\n')

best_model_description (best_model_2 , x2_train, x2_test, y2_train, y2_test, best_a
Target (smoothed wtested positive 14d)
The best model is: Ridge with an alpha of 0.01.
R^2: 0.6321
RMSE: 4.4946

The coefficients for this model are:

Evaluation Data:
R^2: 0.6117074321264773
RMSE: 4.612407444163958

smoothed_wwearing_mask                  10.351066
smoothed_wcli                            4.960262
smoothed_wvaccine_likely_politicians     0.533502
smoothed_wlarge_event_1d                 0.373462
smoothed_wworried_become_ill             0.321111
smoothed_wvaccine_likely_friends         0.260996
smoothed_wrestaurant_1d                  0.213972
smoothed_wwork_outside_home_1d           0.034709
smoothed_wvaccine_likely_who             0.033909
smoothed_wshop_1d                       -0.090049
smoothed_wspent_time_1d                 -0.179334
smoothed_wcovid_vaccinated              -0.211614
smoothed_wtested_14d                    -0.232205
smoothed_wvaccine_likely_govt_health    -0.450208
smoothed_wpublic_transit_1d             -1.489093
smoothed_wothers_masked                -11.800190
dtype: float64

Target (smoothed wcovid vaccinated)
The best model is: Ridge with an alpha of 1.
R^2: 0.5226
RMSE: 4.2511

The coefficients for this model are:

Evaluation Data:
R^2: 0.5824368608257332
RMSE: 3.856757245137484

smoothed_wwearing_mask                  2.261368
smoothed_wvaccine_likely_friends        0.586713
smoothed_wshop_1d                       0.508858
smoothed_wwork_outside_home_1d          0.299391
smoothed_wvaccine_likely_govt_health    0.157940
smoothed_wtested_14d                    0.112766
smoothed_wspent_time_1d                 0.086840
smoothed_wpublic_transit_1d             0.032471
smoothed_wworried_become_ill           -0.016942
smoothed_wrestaurant_1d                -0.024989
smoothed_wtested_positive_14d          -0.188045
smoothed_wvaccine_likely_who           -0.347515
smoothed_wlarge_event_1d               -0.385616
smoothed_wvaccine_likely_politicians   -0.440882
smoothed_wcli                          -0.838416
```

```
smoothed_wothers_masked                -2.953220
dtype: float64
```

Feature Selection

In [35]:

```
scaler_x1 = StandardScaler()
scaler_x2 = StandardScaler()

x1_train_scaled = scaler_x1.fit_transform(x1_train)
x2_train_scaled = scaler_x2.fit_transform(x2_train)


pca = PCA(n_components=0.95)  # Keep 95% of variance
x1_pca = pca.fit_transform(x1_train_scaled)
x2_pca = pca.fit_transform(x2_train_scaled)

x1_test_scaled = scaler_x1.transform(x1_test)
x2_test_scaled = scaler_x2.transform(x2_test)

pca = PCA(n_components=0.95)  # Keep 95% of variance
x1_test_pca = pca.fit_transform(x1_test_scaled)
x2_test_pca = pca.fit_transform(x2_test_scaled)

print(f"Reduced to {x1_pca.shape[1]} principal components for x1")
print(f"Reduced to {x2_pca.shape[1]} principal components for x2")

print('\nTarget Question 1')
model_q1, weights_q1 = best_model_with_pca(best_model, x1_pca, x1_test_pca, y1_trai

print('\nTarget Question 2')
model_q2, weights_q2 = best_model_with_pca(best_model_2, x2_pca, x2_test_pca, y2_tr
Reduced to 9 principal components for x1
Reduced to 9 principal components for x2

Target Question 1
R^2 Score: 0.5777
RMSE: 4.8101

Feature importances (sorted by weight):

PC3    2.678210
PC1    0.712773
PC8    0.185343
PC5   -0.108141
PC6   -0.272794
PC7   -0.409414
PC9   -1.647846
PC2   -2.101883
PC4   -3.280550
dtype: float64

Target Question 2
R^2 Score: 0.3894
RMSE: 4.6639

Feature importances (sorted by weight):

PC5    1.689813
PC3    1.317748
PC6    1.228069
PC8    0.801693
PC7    0.774012
PC1    0.565668
PC4   -0.684124
PC2   -2.227426
PC9   -2.228972
dtype: float64
```

In [36]:

```
# x1_pca_df = pd.DataFrame(x1_pca, columns=[f'PC{i+1}' for i in range(x1_pca.shape[
# x2_pca_df = pd.DataFrame(x2_pca, columns=[f'PC{i+1}' for i in range(x2_pca.shape[

# x1_test_pca_df = pd.DataFrame(x1_test_pca, columns=[f'PC{i+1}' for i in range(x1_
# x2_test_pca_df = pd.DataFrame(x2_test_pca, columns=[f'PC{i+1}' for i in range(x2_

# # Export to CSV
# x1_pca_df.to_csv("Devonte_x1_pca_train.csv", index=False)
# x2_pca_df.to_csv("Devonte_x2_pca_train.csv", index=False)

# x1_test_pca_df.to_csv("x1_pca_test.csv", index=False)
# x2_test_pca_df.to_csv("x2_pca_test.csv", index=False)
```

In [37]:

```
print('What comprises each pca for x1\n')
loadings_1 = pd.DataFrame(pca.components_.T,
                          columns=[f'PC{i+1}' for i in range(pca.n_components_)],
                          index=x1_train.columns)

print(loadings_1)

print('\nWhat comprises each pca for x2\n')
loadings_2 = pd.DataFrame(pca.components_.T,
                        columns=[f'PC{i+1}' for i in range(pca.n_components_)],
                        index=x2_train.columns)

print(loadings_2)
```

```
What comprises each pca for x1

                                        PC1       PC2       PC3       PC4  \
smoothed_wspent_time_1d            0.309295 -0.132544  0.185345 -0.079630
smoothed_wtested_14d              -0.234913  0.092286  0.035079  0.403042
smoothed_wpublic_transit_1d       -0.168067 -0.185971  0.189871  0.741812
smoothed_wworried_become_ill      -0.261984  0.167716  0.217196 -0.172941
smoothed_wvaccine_likely_govt_health -0.302042 -0.034083  0.249285 -0.069099
smoothed_wshop_1d                  0.194084 -0.272275  0.383356 -0.109202
smoothed_wwork_outside_home_1d     0.118197  0.599011  0.103408 -0.150121
smoothed_wothers_masked            0.255290 -0.148330  0.337864 -0.072696
smoothed_wcli                     -0.263603 -0.173234 -0.031783 -0.274283
smoothed_wcovid_vaccinated         0.082267  0.617886  0.064108  0.131922
smoothed_wvaccine_likely_friends  -0.242076  0.046925  0.462414 -0.137426
smoothed_wrestaurant_1d            0.257057  0.019503  0.388645  0.035885
smoothed_wvaccine_likely_politicians -0.289264  0.179294  0.239471  0.094974
smoothed_wvaccine_likely_who      -0.302467 -0.044905  0.216095 -0.106577
smoothed_wwearing_mask            -0.285369 -0.060391  0.015359 -0.241766
smoothed_wlarge_event_1d           0.294449  0.053896  0.270133  0.121918

                                        PC5       PC6       PC7       PC8  \
smoothed_wspent_time_1d            0.041447  0.088329  0.033222  0.076813
smoothed_wtested_14d               0.548542  0.226004  0.440007  0.039190
smoothed_wpublic_transit_1d       -0.043635 -0.438561 -0.305543 -0.020657
smoothed_wworried_become_ill       0.348763 -0.232160  0.142133  0.222127
smoothed_wvaccine_likely_govt_health -0.232796  0.141911  0.004082  0.061647
smoothed_wshop_1d                  0.322001  0.130615 -0.493467  0.481401
smoothed_wwork_outside_home_1d    -0.052983 -0.590710 -0.061245  0.131623
smoothed_wothers_masked            0.231695 -0.141972 -0.013742 -0.728010
smoothed_wcli                      0.214363 -0.112771 -0.053732 -0.185292
smoothed_wcovid_vaccinated         0.241001  0.422354 -0.392917 -0.174980
smoothed_wvaccine_likely_friends  -0.187216  0.110401 -0.019771 -0.224785
smoothed_wrestaurant_1d           -0.024848 -0.055354  0.502193  0.183167
smoothed_wvaccine_likely_politicians -0.285805  0.152671 -0.021885 -0.027859
smoothed_wvaccine_likely_who      -0.183835  0.122172 -0.003929  0.082677
smoothed_wwearing_mask             0.284640 -0.204204 -0.036173 -0.057305
smoothed_wlarge_event_1d          -0.152565  0.048031  0.180566  0.001028

                                        PC9
smoothed_wspent_time_1d           -0.146906
smoothed_wtested_14d               0.247255
smoothed_wpublic_transit_1d       -0.180835
smoothed_wworried_become_ill       0.136763
smoothed_wvaccine_likely_govt_health -0.090205
smoothed_wshop_1d                  0.179435
smoothed_wwork_outside_home_1d     0.144723
smoothed_wothers_masked            0.139407
smoothed_wcli                     -0.372176
smoothed_wcovid_vaccinated        -0.364184
smoothed_wvaccine_likely_friends   0.351807
smoothed_wrestaurant_1d           -0.475728
smoothed_wvaccine_likely_politicians  0.038771
smoothed_wvaccine_likely_who      -0.196203
smoothed_wwearing_mask            -0.348748
smoothed_wlarge_event_1d          -0.048064
```

```
What comprises each pca for x2

                                          PC1       PC2       PC3       PC4  \
smoothed_wspent_time_1d              0.309295 -0.132544  0.185345 -0.079630
smoothed_wtested_14d                -0.234913  0.092286  0.035079  0.403042
smoothed_wpublic_transit_1d         -0.168067 -0.185971  0.189871  0.741812
smoothed_wworried_become_ill        -0.261984  0.167716  0.217196 -0.172941
smoothed_wvaccine_likely_govt_health -0.302042 -0.034083  0.249285 -0.069099
smoothed_wshop_1d                    0.194084 -0.272275  0.383356 -0.109202
smoothed_wtested_positive_14d        0.118197  0.599011  0.103408 -0.150121
smoothed_wwork_outside_home_1d       0.255290 -0.148330  0.337864 -0.072696
smoothed_wothers_masked             -0.263603 -0.173234 -0.031783 -0.274283
smoothed_wcli                        0.082267  0.617886  0.064108  0.131922
smoothed_wvaccine_likely_friends    -0.242076  0.046925  0.462414 -0.137426
smoothed_wrestaurant_1d              0.257057  0.019503  0.388645  0.035885
smoothed_wvaccine_likely_politicians -0.289264  0.179294  0.239471  0.094974
smoothed_wvaccine_likely_who        -0.302467 -0.044905  0.216095 -0.106577
smoothed_wwearing_mask              -0.285369 -0.060391  0.015359 -0.241766
smoothed_wlarge_event_1d             0.294449  0.053896  0.270133  0.121918

                                          PC5       PC6       PC7       PC8  \
smoothed_wspent_time_1d              0.041447  0.088329  0.033222  0.076813
smoothed_wtested_14d                 0.548542  0.226004  0.440007  0.039190
smoothed_wpublic_transit_1d         -0.043635 -0.438561 -0.305543 -0.020657
smoothed_wworried_become_ill         0.348763 -0.232160  0.142133  0.222127
smoothed_wvaccine_likely_govt_health -0.232796  0.141911  0.004082  0.061647
smoothed_wshop_1d                    0.322001  0.130615 -0.493467  0.481401
smoothed_wtested_positive_14d       -0.052983 -0.590710 -0.061245  0.131623
smoothed_wwork_outside_home_1d       0.231695 -0.141972 -0.013742 -0.728010
smoothed_wothers_masked              0.214363 -0.112771 -0.053732 -0.185292
smoothed_wcli                        0.241001  0.422354 -0.392917 -0.174980
smoothed_wvaccine_likely_friends    -0.187216  0.110401 -0.019771 -0.224785
smoothed_wrestaurant_1d             -0.024848 -0.055354  0.502193  0.183167
smoothed_wvaccine_likely_politicians -0.285805  0.152671 -0.021885 -0.027859
smoothed_wvaccine_likely_who        -0.183835  0.122172 -0.003929  0.082677
smoothed_wwearing_mask               0.284640 -0.204204 -0.036173 -0.057305
smoothed_wlarge_event_1d            -0.152565  0.048031  0.180566  0.001028

                                          PC9
smoothed_wspent_time_1d             -0.146906
smoothed_wtested_14d                 0.247255
smoothed_wpublic_transit_1d         -0.180835
smoothed_wworried_become_ill         0.136763
smoothed_wvaccine_likely_govt_health -0.090205
smoothed_wshop_1d                    0.179435
smoothed_wtested_positive_14d        0.144723
smoothed_wwork_outside_home_1d       0.139407
smoothed_wothers_masked             -0.372176
smoothed_wcli                       -0.364184
smoothed_wvaccine_likely_friends     0.351807
smoothed_wrestaurant_1d             -0.475728
smoothed_wvaccine_likely_politicians  0.038771
smoothed_wvaccine_likely_who        -0.196203
smoothed_wwearing_mask              -0.348748
smoothed_wlarge_event_1d            -0.048064
```

Going to try RandomForest Regressor Model Below

In [39]:

```
def rf_model(x_train, y_train, estimate, depth, feature):

    regr = RandomForestRegressor(n_estimators=estimate, max_depth=depth, max_featur
    regr = regr.fit(x_train, y_train)
    kf = KFold(n_splits=5, shuffle=True, random_state=42)

    r2 = cross_val_score(regr, x_train, y_train, cv=kf, scoring='r2')
    rmse = -cross_val_score(regr, x_train, y_train, cv=kf, scoring='neg_mean_square

    return (estimate, depth, feature), r2.mean(), rmse.mean()

def rf_eval(parameters, x_train, x_test, y_train, y_test):
    regr = RandomForestRegressor(n_estimators=parameters[0], max_depth=parameters[1
    regr = regr.fit(x_train, y_train)
    regr_pred = regr.predict(x_test)
    r2 = r2_score(y_test, regr_pred)
    rmse = root_mean_squared_error(y_test, regr_pred)

    print('Evaluation Data:\nR^2: ' + str(r2) + '\nRMSE: ' + str(rmse))
```

In [40]:

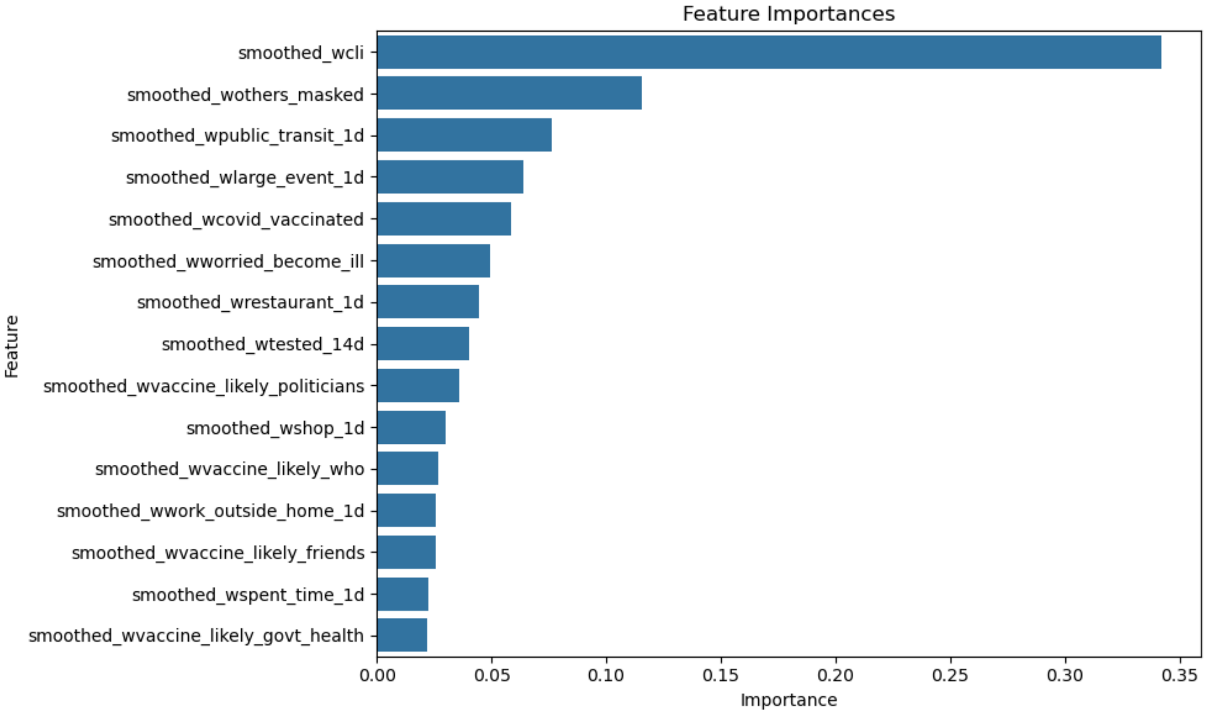
```
# Took my laptop about 20 mins to run.
estimators = [50, 100, 150] #Number of trees in the forest.
depths = [5,10,15,20] # Determining how deep to make the model.
features = [6, 8, 10, 12] #How many features to consider when splitting.

best_parameters = None
best_r2 = None
best_rmse = float('inf')

print("Covid Vaccine Target")
for estimate in estimators:
    for d in depths:
        for f in features:
            parameters, r2, rmse = rf_model(x1_train, y1_train, estimate, d, f) #To
        if rmse < best_rmse:
            best_parameters = parameters
            best_r2 = r2
            best_rmse = rmse


print('Best Parameters:' + str(best_parameters) + '\nR^2: ' + str(best_r2) + '\nRMS

rf_eval(best_parameters, x1_train, x1_test,y1_train, y1_test)

best_parameters_2 = None
best_r2_2 = None
best_rmse_2 = float('inf')

print("Positive Cases Target")
for estimate in estimators:
    for d in depths:
        for f in features:
            parameters, r2, rmse = rf_model(x2_train, y2_train, estimate, d, f)
        if rmse < best_rmse_2:
            best_parameters_2 = parameters
            best_r2_2 = r2
            best_rmse_2 = rmse


print('Best Parameters:' + str(best_parameters_2) + '\nR^2: ' + str(best_r2_2) + '\

rf_eval(best_parameters, x2_train, x2_test,y2_train, y2_test)
```

```
Covid Vaccine Target
Best Parameters:(150, 20, 12)
R^2: 0.8118981980616496
RMSE: 10.360617227155643
Evaluation Data:
R^2: 0.8240825704967376
RMSE: 3.104577893018349
Positive Cases Target
Best Parameters:(150, 20, 12)
R^2: 0.789178208952738
RMSE: 8.021496570012955
Evaluation Data:
R^2: 0.8241278119378252
RMSE: 2.502994011151542
```

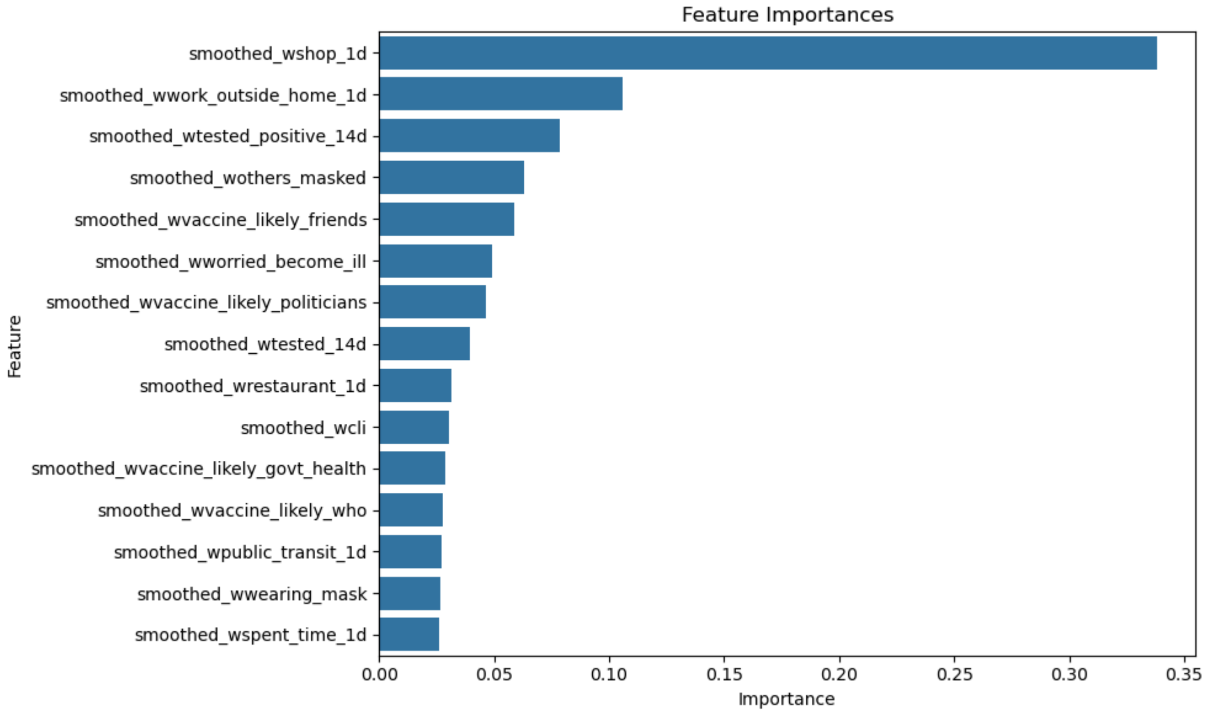
In [41]:

```
regr = RandomForestRegressor(n_estimators=120, max_depth=20, max_features=12, rando
regr.fit(x1_train, y1_train)

importances = regr.feature_importances_

# Combine with column names
feature_names = x1_train.columns
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(feature_importance_df.head(10))

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(15))
plt.title("Feature Importances")
plt.tight_layout()
plt.show()


regr = RandomForestRegressor(n_estimators=120, max_depth=20, max_features=12, rando
regr.fit(x2_train, y2_train)

importances_2 = regr.feature_importances_

# Combine with column names
feature_names_2 = x2_train.columns
feature_importance_df_2 = pd.DataFrame({
    'Feature': feature_names_2,
    'Importance': importances_2
}).sort_values(by='Importance', ascending=False)

print(feature_importance_df_2.head(10))

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df_2.head(15))
plt.title("Feature Importances")
plt.tight_layout()
plt.show()
```

```
                                Feature  Importance
8                         smoothed_wcli    0.342219
7                smoothed_wothers_masked    0.115471
2            smoothed_wpublic_transit_1d    0.076371
15             smoothed_wlarge_event_1d    0.063721
9              smoothed_wcovid_vaccinated    0.058395
3             smoothed_wworried_become_ill    0.049601
11              smoothed_wrestaurant_1d    0.044621
1                  smoothed_wtested_14d    0.040417
12  smoothed_wvaccine_likely_politicians    0.035978
5                      smoothed_wshop_1d    0.030148
```



```
                                Feature  Importance
5                      smoothed_wshop_1d    0.338071
7         smoothed_wwork_outside_home_1d    0.105886
6           smoothed_wtested_positive_14d    0.078717
8                smoothed_wothers_masked    0.062890
10      smoothed_wvaccine_likely_friends    0.058795
3             smoothed_wworried_become_ill    0.049311
12  smoothed_wvaccine_likely_politicians    0.046582
1                  smoothed_wtested_14d    0.039673
11              smoothed_wrestaurant_1d    0.031441
9                         smoothed_wcli    0.030546
```



In [42]:

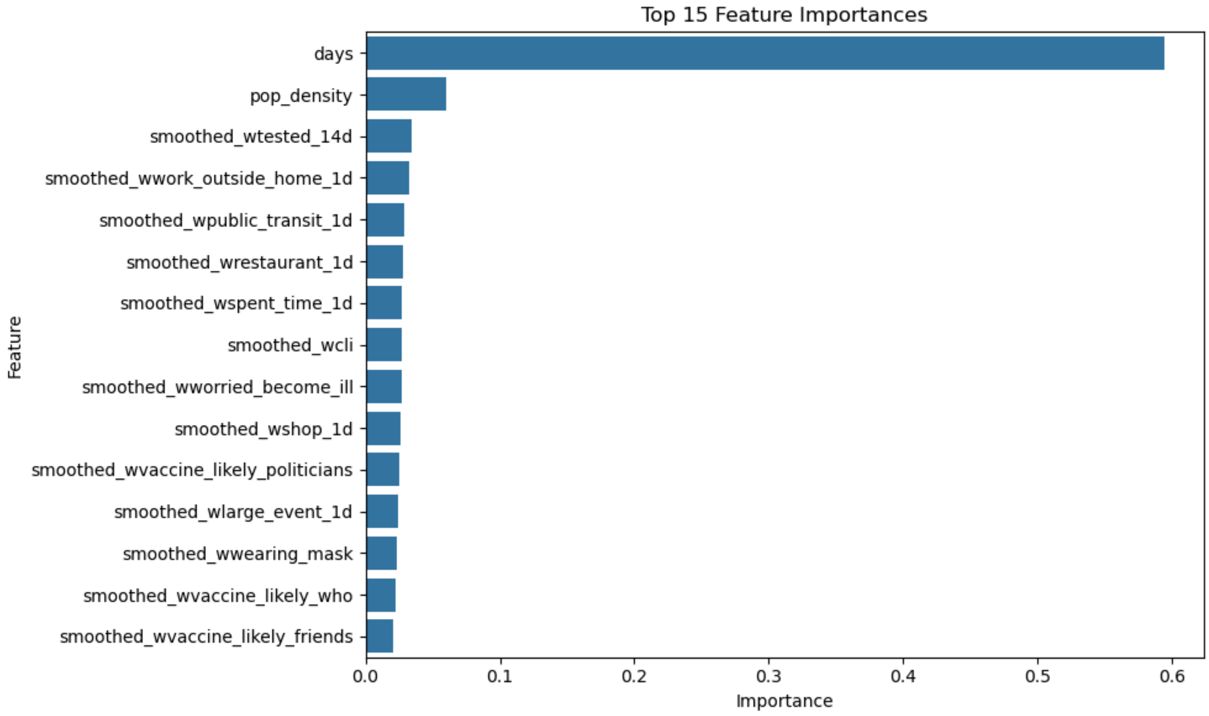
```python
# Read and Clean in Justin's EDA.

def read_clean_csv(filepath):
    df = pd.read_csv(filepath)
    # Drop first column if it's unnamed (usually an index column)
    if df.columns[0].startswith('Unnamed'):
        df = df.iloc[:, 1:]
    return df

cv_test = read_clean_csv('cv_test.csv')
cv_train = read_clean_csv('cv_train.csv')
tp_test = read_clean_csv('tp_original_test.csv')
tp_train = read_clean_csv('tp_original_train.csv')
tp_imputed_test = read_clean_csv('tp_wimputed_val.csv')
tp_imputed_train = read_clean_csv('tp_wimputed_train.csv')

#For Covid Vaccine:
cv_train_x = cv_train.drop('smoothed_wcovid_vaccinated', axis=1)
cv_test_x = cv_test.drop('smoothed_wcovid_vaccinated', axis=1)

cv_train_y = cv_train['smoothed_wcovid_vaccinated']
cv_test_y = cv_test['smoothed_wcovid_vaccinated']

#For Test_Positive Original:
tp_train_x = tp_train.drop('tested_pos', axis=1)
tp_test_x = tp_test.drop('tested_pos', axis=1)

tp_train_y = tp_train['tested_pos']
tp_test_y = tp_test['tested_pos']

#For Test_Positive Imputed:
tp_imputed_train_x = tp_imputed_train.drop('tested_pos', axis=1)
tp_imputed_test_x = tp_imputed_test.drop('tested_pos', axis=1)
```

```
tp_imputed_train_y = tp_imputed_train['tested_pos']
tp_imputed_test_y = tp_imputed_test['tested_pos']
```

In [43]:

```
# For covid vaccine

regr = RandomForestRegressor(n_estimators=120, max_depth=20,random_state = 42)
regr.fit(cv_train_x, cv_train_y)

kf = KFold(n_splits=5, shuffle=True, random_state=42)
r2 = cross_val_score(regr, cv_train_x, cv_train_y, cv=kf, scoring='r2')
rmse = -cross_val_score(regr, cv_train_x, cv_train_y, cv=kf, scoring='neg_mean_squa

print('\nR^2: ' + str(r2.mean()) + '\nRMSE: ' + str(rmse.mean()))

regr_pred = regr.predict(cv_test_x)
r2 = r2_score(cv_test_y, regr_pred)
rmse = root_mean_squared_error(cv_test_y, regr_pred)

print('Evaluation Data:\nR^2: ' + str(r2) + '\nRMSE: ' + str(rmse))

importances = regr.feature_importances_

feature_names = cv_train_x.columns
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(feature_importance_df.head(10))

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(15))
plt.title("Top 15 Feature Importances")
plt.tight_layout()
plt.show()
```

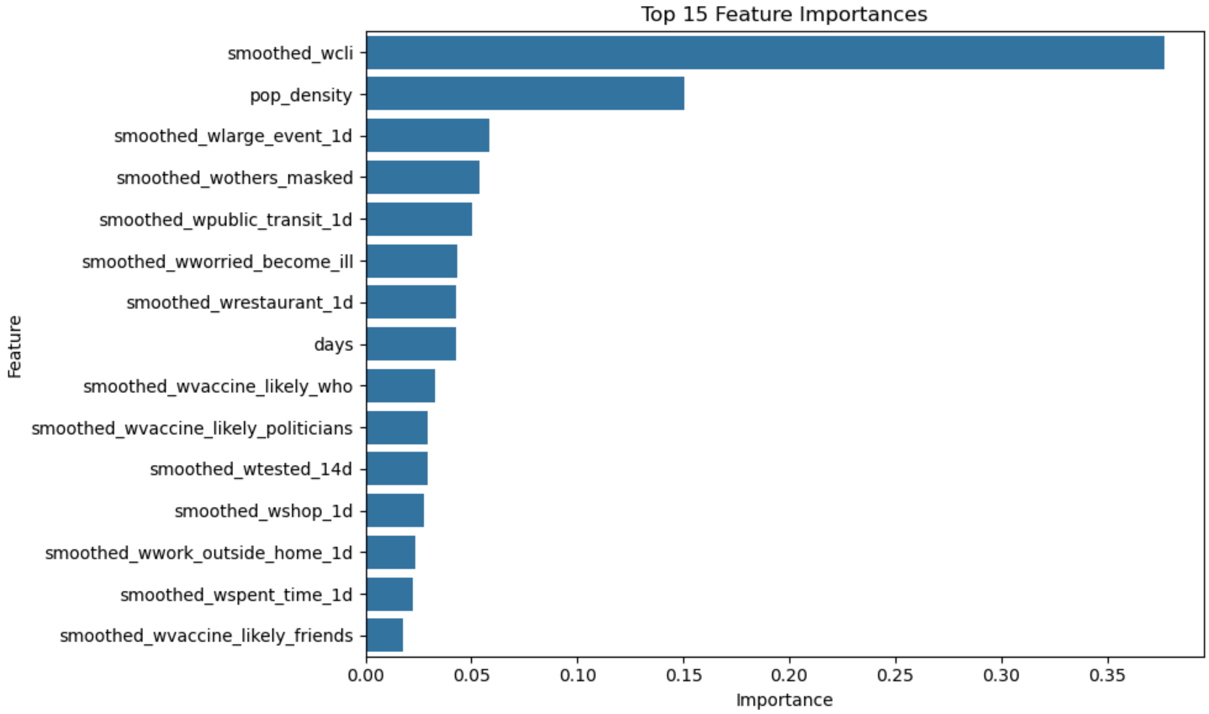
```
R^2: 0.7309944762592161
RMSE: 11.943156584966662
Evaluation Data:
R^2: 0.7646942208429746
RMSE: 3.214171847436728
                           Feature  Importance
14                            days    0.594082
13                     pop_density    0.059637
1               smoothed_wtested_14d    0.034740
5     smoothed_wwork_outside_home_1d    0.032475
2        smoothed_wpublic_transit_1d    0.029104
8            smoothed_wrestaurant_1d    0.027773
0            smoothed_wspent_time_1d    0.027179
6                      smoothed_wcli    0.027138
3       smoothed_wworried_become_ill    0.026696
4                  smoothed_wshop_1d    0.025745
```



In [44]:

```
# For Positive Test Positive using original

regr = RandomForestRegressor(n_estimators=120, max_depth=20,random_state = 42)
regr.fit(tp_train_x, tp_train_y)

kf = KFold(n_splits=5, shuffle=True, random_state=42)
r2 = cross_val_score(regr, tp_train_x, tp_train_y, cv=kf, scoring='r2')
rmse = -cross_val_score(regr, tp_train_x, tp_train_y, cv=kf, scoring='neg_mean_squa

print('\nR^2: ' + str(r2.mean()) + '\nRMSE: ' + str(rmse.mean()))

regr_pred = regr.predict(tp_test_x)
r2 = r2_score(tp_test_y, regr_pred)
rmse = root_mean_squared_error(tp_test_y, regr_pred)

print('Evaluation Data:\nR^2: ' + str(r2) + '\nRMSE: ' + str(rmse))

importances = regr.feature_importances_

feature_names = tp_train_x.columns
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(feature_importance_df.head(10))

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(15))
plt.title("Top 15 Feature Importances")
plt.tight_layout()
plt.show()
```

```
R^2: 0.8293547531519712
RMSE: 9.151638600152694
Evaluation Data:
R^2: 0.854278849091203
RMSE: 2.9581236748404653
                                  Feature  Importance
7                            smoothed_wcli    0.376836
13                             pop_density    0.150315
12               smoothed_wlarge_event_1d    0.058321
6                  smoothed_wothers_masked    0.054008
2               smoothed_wpublic_transit_1d    0.050079
3               smoothed_wworried_become_ill    0.043242
9                  smoothed_wrestaurant_1d    0.042551
14                                    days    0.042508
11              smoothed_wvaccine_likely_who    0.032609
10  smoothed_wvaccine_likely_politicians    0.029376
```

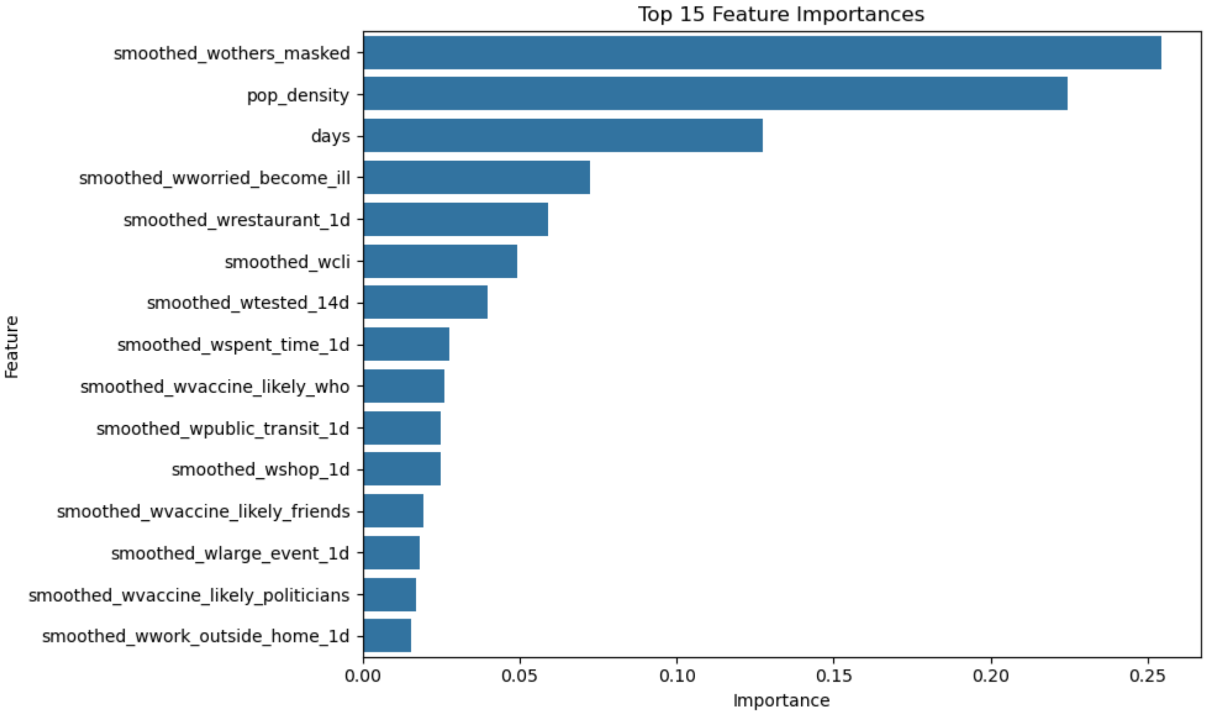


In [45]:

```python
# For Positive Test Positive using imputed

regr = RandomForestRegressor(n_estimators=120, max_depth=20,random_state = 42)
regr.fit(tp_imputed_train_x, tp_imputed_train_y)

kf = KFold(n_splits=5, shuffle=True, random_state=42)
r2 = cross_val_score(regr, tp_imputed_train_x, tp_imputed_train_y, cv=kf, scoring='
rmse = -cross_val_score(regr, tp_imputed_train_x, tp_imputed_train_y, cv=kf, scorin

print('\nR^2: ' + str(r2.mean()) + '\nRMSE: ' + str(rmse.mean()))

regr_pred = regr.predict(tp_imputed_test_x)
r2 = r2_score(tp_imputed_test_y, regr_pred)
rmse = root_mean_squared_error(tp_imputed_test_y, regr_pred)

print('Evaluation Data:\nR^2: ' + str(r2) + '\nRMSE: ' + str(rmse))
```

```
importances = regr.feature_importances_

feature_names = tp_imputed_train_x.columns
feature_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print(feature_importance_df.head(10))

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df.head(15))
plt.title("Top 15 Feature Importances")
plt.tight_layout()
plt.show()
```

```
R^2: 0.8850073568288301
RMSE: 3.196820894510201
Evaluation Data:
R^2: 0.9016865988408448
RMSE: 1.6371571912483383
                         Feature  Importance
6         smoothed_wothers_masked    0.254220
13                    pop_density    0.224479
14                           days    0.127433
3   smoothed_wworried_become_ill    0.072400
9         smoothed_wrestaurant_1d    0.059138
7                   smoothed_wcli    0.049365
1            smoothed_wtested_14d    0.039744
0         smoothed_wspent_time_1d    0.027634
11  smoothed_wvaccine_likely_who    0.025879
2     smoothed_wpublic_transit_1d    0.024912
```


Top 15 Feature Importances
Feature
smoothed_wothers_masked
pop_density
days
smoothed_wworried_become_ill
smoothed_wrestaurant_1d
smoothed_wcli
smoothed_wtested_14d
smoothed_wspent_time_1d
smoothed_wvaccine_likely_who
smoothed_wpublic_transit_1d
smoothed_wshop_1d
smoothed_wvaccine_likely_friends
smoothed_wlarge_event_1d
smoothed_wvaccine_likely_politicians
smoothed_wwork_outside_home_1d
Importance