In [1]:
```
import pandas as pd

import emission.core.get_database as edb
import emission.core.wrapper.entry as ecwe
import emission.storage.decorations.analysis_timeseries_queries as esda
import emission.storage.decorations.trip_queries as esdt
import emission.storage.decorations.timeline as esdl
import emission.storage.timeseries.abstract_timeseries as esta
import emission.storage.timeseries.timequery as estt
import scaffolding
from uuid import UUID

%matplotlib inline
```

```
URL not formatted, defaulting to "Stage_database"
Connecting to database URL db
```

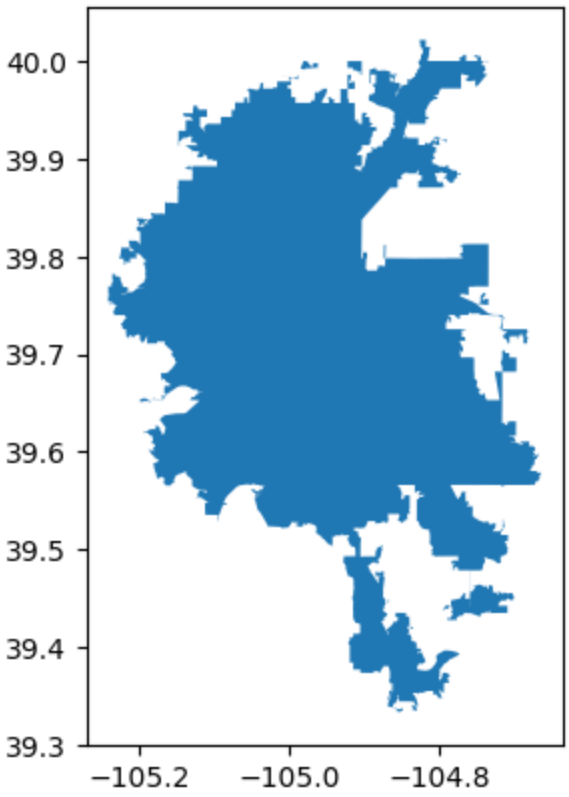
# Load the programs

In [2]:
```
# Split UUIDs by program
program_uuid_map = {}
for ue in edb.get_uuid_db().find():
    program = ue['user_email'].split("_")[0]
    if program in program_uuid_map.keys():
        program_uuid_map[program].append(str(ue['uuid']))
    else:
        print(f"Found new program {program}, creating new list")
        program_uuid_map[program] = []
        program_uuid_map[program].append(str(ue['uuid']))
```

```
Found new program 4c, creating new list
Found new program cc, creating new list
Found new program fc, creating new list
Found new program pc, creating new list
Found new program sc, creating new list
Found new program stage, creating new list
Found new program vail, creating new list
Found new program prepilot, creating new list
```

In [3]:
```
uuid_program_list = []
for ue in edb.get_uuid_db().find():
    program = ue['user_email'].split("_")[0]
    uuid_program_list.append({"program": program, "opcode": ue["user_email"]
```

In [4]:
```
uuid_program_df = pd.DataFrame.from_dict(uuid_program_list)
uuid_program_df.head()
```




In [15]:
```
denver_pixels = gpd.read_file("geom/denver_uza_grid.shp")
```

In [16]:
```
denver_pixels
```

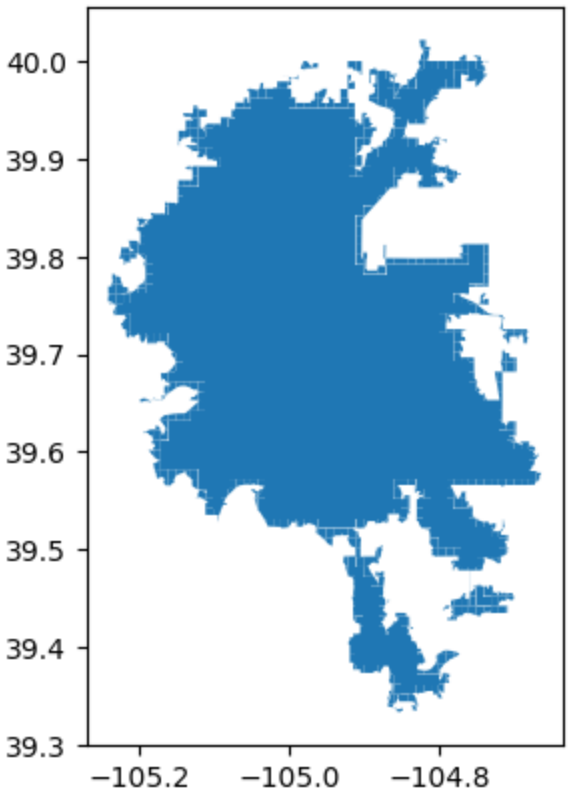
Out[16]:

| | id | lon | lat | geometry |
|---|---|---|---|---|
| **0** | 1 | -104.823128 | 40.017839 | POLYGON ((-104.82228 40.01201, -104.82276 40.0... |
| **1** | 2 | -104.818025 | 40.015812 | POLYGON ((-104.82098 40.02200, -104.81787 40.0... |
| **2** | 3 | -104.822497 | 40.005625 | POLYGON ((-104.82509 40.00201, -104.82454 40.0... |
| **3** | 4 | -104.817468 | 40.007237 | POLYGON ((-104.81300 40.01201, -104.81428 40.0... |
| **4** | 5 | -104.771824 | 40.003194 | POLYGON ((-104.77268 40.00201, -104.77268 40.0... |
| **...** | ... | ... | ... | ... |
| **2191** | 2192 | -104.864830 | 39.340298 | POLYGON ((-104.86098 39.34195, -104.86120 39.3... |
| **2192** | 2193 | -104.854509 | 39.335405 | MULTIPOLYGON (((-104.85878 39.34201, -104.8589... |
| **2193** | 2194 | -104.849958 | 39.335809 | POLYGON ((-104.85098 39.33716, -104.85032 39.3... |
| **2194** | 2195 | -104.833317 | 39.337796 | MULTIPOLYGON (((-104.83560 39.33528, -104.8350... |
| **2195** | 2196 | -104.830081 | 39.339239 | MULTIPOLYGON (((-104.83098 39.33796, -104.8302... |

2196 rows × 4 columns

In [17]:
```
denver_pixels.plot()
```

Out[17]:
```
<AxesSubplot:>
```




## Prepping dataframes for plotting

In [18]:
```
all_start_end_points = trip_program_df.start_loc.append(trip_program_df.end_
```

In [19]:
```
import shapely as shp
```

In [20]:
```
# I wanted to use shapely's from_geojson but it doesn't seem to be supporte
all_geo_start_end_points = gpd.GeoSeries(all_start_end_points.apply(lambda p
```

In [21]:
```
denver_boundary.geometry.iloc[0]
```

Out[21]:

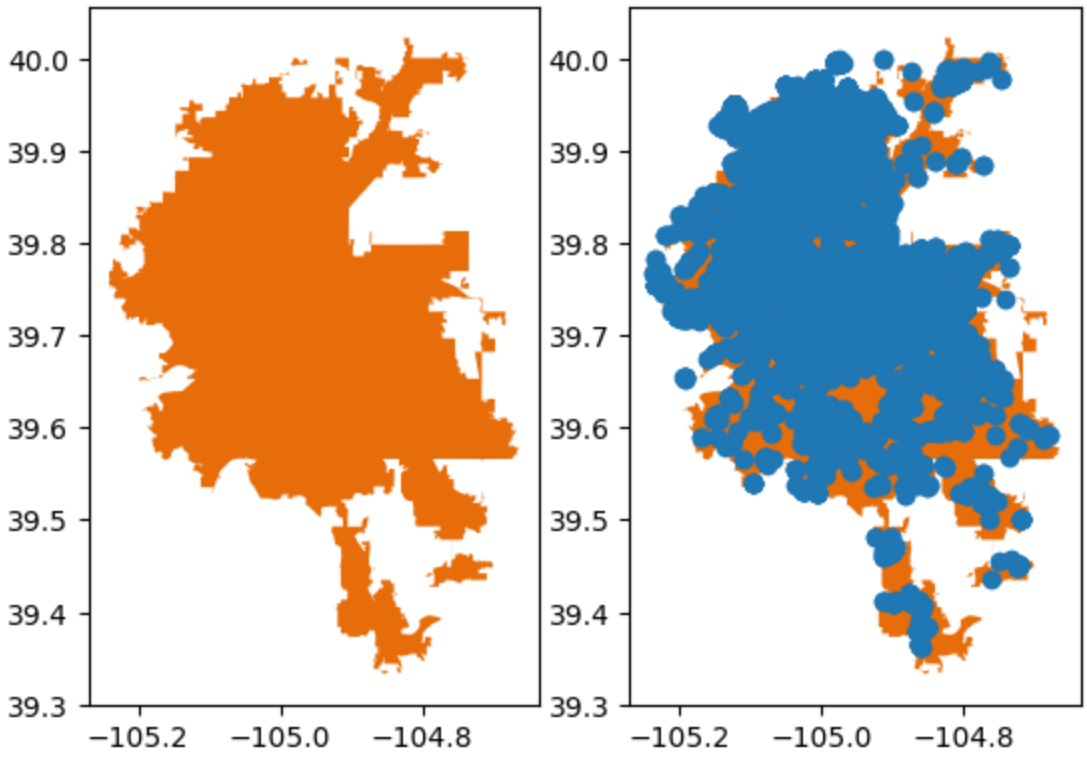
b'  '

In [22]:
```
all_start_or_end_within = all_geo_start_end_points[all_geo_start_end_points.
```

In [23]:
```
import matplotlib.pyplot as plt
```

In [24]:
```
fig, ax_arr = plt.subplots(nrows=1, ncols=2)
denver_boundary.plot(color="#E66D0A", ax=ax_arr[0])
denver_boundary.plot(color="#E66D0A", ax=ax_arr[1])
all_start_or_end_within.plot(ax=ax_arr[1])
```

Out[24]: <AxesSubplot:>




In [25]:
```
e_bike_trips = trip_program_df[trip_program_df.mode_confirm == 'pilot_ebike'
```

Out[25]: 15807

In [26]:
```
car_like_trips = trip_program_df.query('mode_confirm == "drove_alone" | mode
```

Out[26]: 28717

In [27]:
```
e_bike_start_end_points = e_bike_trips.start_loc.append(e_bike_trips.end_loc
```

Out[27]: 31614

In [28]:
```
e_bike_geo_start_end_points = gpd.GeoSeries(e_bike_start_end_points.apply(la
```

In [29]:
```
e_bike_start_or_end_within = e_bike_geo_start_end_points[e_bike_geo_start_en
```

In [30]:
```
car_like_start_end_points = car_like_trips.start_loc.append(car_like_trips.e
```

Out[30]: 57434

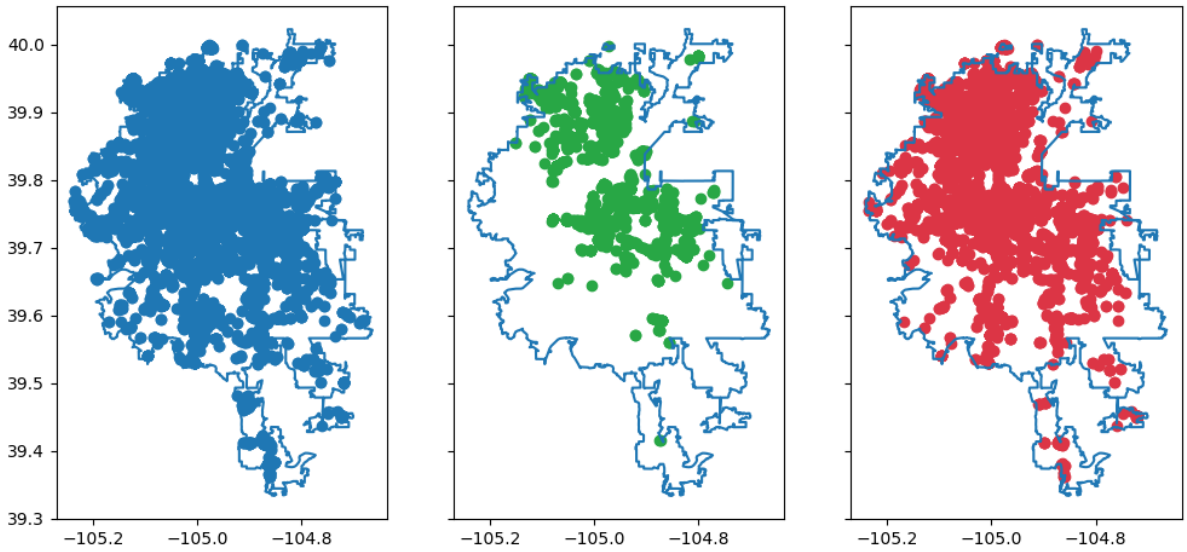
In [31]:
```
car_like_geo_start_end_points = gpd.GeoSeries(car_like_start_end_points.appl
```

In [32]:
```
car_like_start_or_end_within = car_like_geo_start_end_points[car_like_geo_st
```

## General plotting of start/end points

In [33]:
```
fig, ax_arr = plt.subplots(nrows=1, ncols=3, figsize=(12,6), sharex=True, sh
denver_boundary.boundary.plot(ax=ax_arr[0])
all_start_or_end_within.plot(ax=ax_arr[0])
denver_boundary.boundary.plot(ax=ax_arr[1])
e_bike_start_or_end_within.plot(color="#28a745", ax=ax_arr[1])
denver_boundary.boundary.plot(ax=ax_arr[2])
car_like_start_or_end_within.plot(color="#dc3545", ax=ax_arr[2])
```

Out[33]: <AxesSubplot:>




## Heatmap using folium

Example from: https://stackoverflow.com/a/65756840/4040267

In [34]:
```
import folium
import folium.plugins as fpl
```

In [35]:
```
denver_boundary.representative_point()
```

Out[35]:
```
0    POINT (-104.96168 39.67707)
dtype: geometry
```

In [36]:
```
import numpy as np
```

In [37]:
```
all_start_or_end_within.apply(lambda p: [p.y, p.x])
```

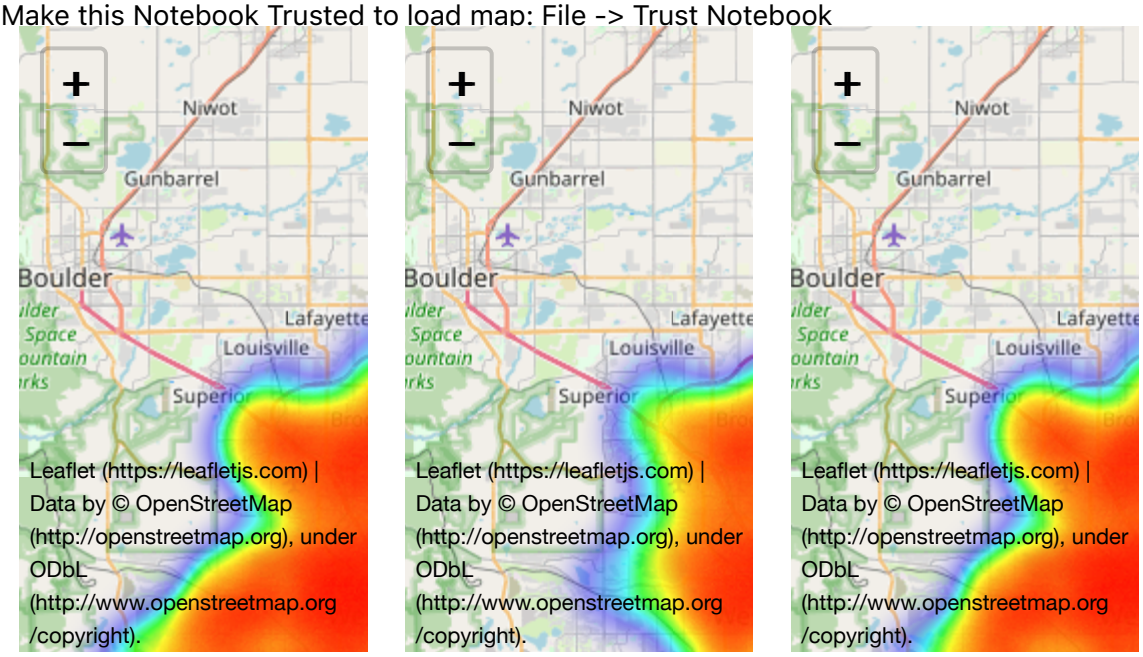
In [41]:
```
all_map = folium.Map(
    location=(39.67707, -104.96168),
    zoom_start=11
)
fpl.HeatMap(all_start_or_end_within.apply(lambda p: [p.y, p.x])).add_to(all_

e_bike_map = folium.Map(
    location=(39.67707, -104.96168),
    zoom_start=11
)
fpl.HeatMap(e_bike_start_or_end_within.apply(lambda p: [p.y, p.x])).add_to(e

car_like_map = folium.Map(
    location=(39.67707, -104.96168),
    zoom_start=11
)
fpl.HeatMap(car_like_start_or_end_within.apply(lambda p: [p.y, p.x])).add_to


fig = bre.Figure()
fig.add_subplot(1,3,1).add_child(all_map)
fig.add_subplot(1,3,2).add_child(e_bike_map)
fig.add_subplot(1,3,3).add_child(car_like_map)
```

Out[41]: Make this Notebook Trusted to load map: File -> Trust Notebook




## Heatmap using geopandas

Example from: https://gist.github.com/perrygeo/c426355e40037c452434

In [42]:
```
from scipy import ndimage
```

In [43]:
```
denver_bounds_extent = (denver_boundary.bounds)[["minx", "maxx", "maxy", "mi
```

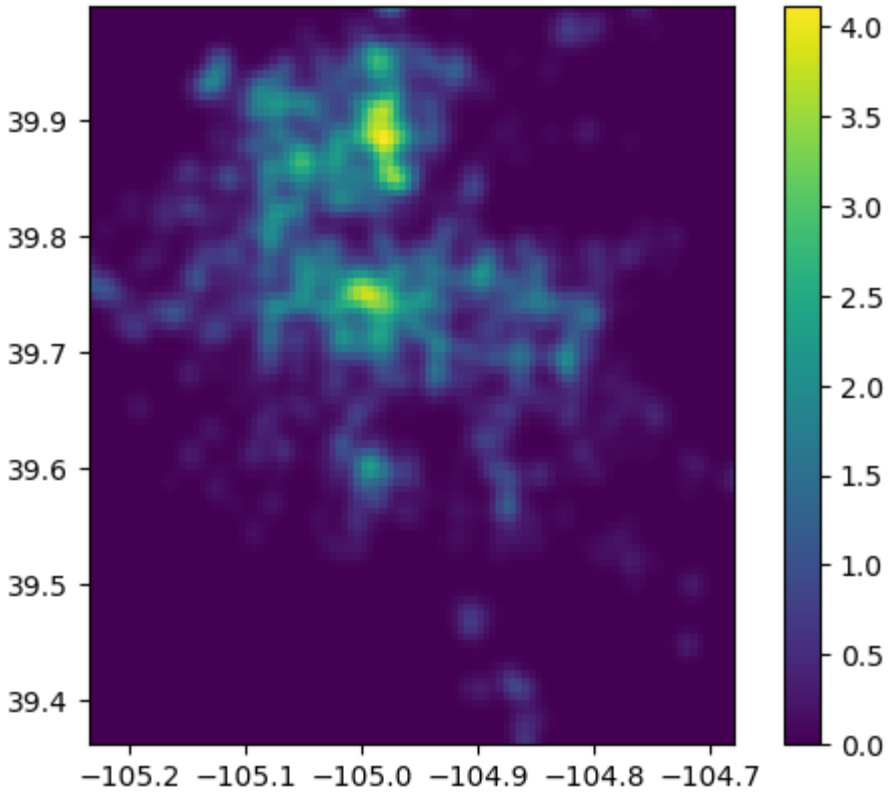
In [44]:
```
def heatmap(d, bins=(100,100), smoothing=1.3, cmap='viridis', ax=None, bound
    heatmap, xedges, yedges = np.histogram2d(d.y, d.x, bins=bins)
    if bounds is None:
        extent = [yedges[0], yedges[-1], xedges[-1], xedges[0]]
        print(extent)
    else:
        extent = bounds
        print(extent)

    logheatmap = np.log(heatmap)
    logheatmap[np.isneginf(logheatmap)] = 0
    logheatmap = ndimage.filters.gaussian_filter(logheatmap, smoothing, mode
    # heatmap = ndimage.filters.gaussian_filter(heatmap, smoothing, mode='ne

    if ax is None:
        plt.imshow(logheatmap, cmap=cmap, extent=extent)
        plt.colorbar()
        plt.gca().invert_yaxis()
        plt.show()
    else:
        ax.imshow(logheatmap, cmap=cmap, extent=extent)
        ax.invert_yaxis()
```

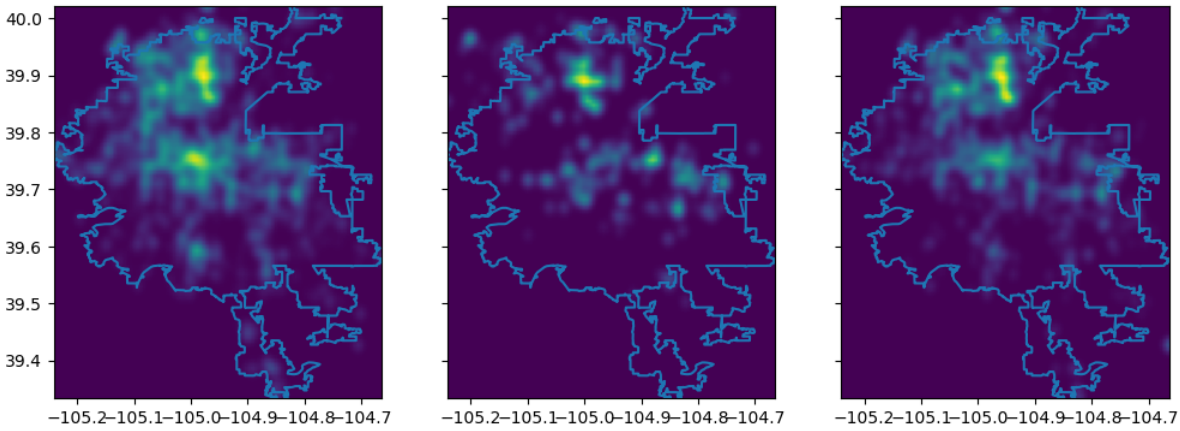
In [45]:
```
heatmap(all_start_or_end_within)
```

```
[-105.23569961329444, -104.680522, 39.9994063, 39.362273017939245]
/root/miniconda-4.12.0/envs/emission/lib/python3.7/site-packages/ipykernel_
launcher.py:10: RuntimeWarning: divide by zero encountered in log
  # Remove the CWD from sys.path while we load stuff.
```



In [46]:
```
fig, ax_arr = plt.subplots(nrows=1, ncols=3, sharex=True, sharey=True, figsi
denver_boundary.boundary.plot(ax=ax_arr[0])
heatmap(all_start_or_end_within, ax=ax_arr[0], bounds=denver_bounds_extent)
denver_boundary.boundary.plot(ax=ax_arr[1])
heatmap(e_bike_start_or_end_within, ax=ax_arr[1], bounds=denver_bounds_exten
denver_boundary.boundary.plot(ax=ax_arr[2])
heatmap(car_like_start_or_end_within, ax=ax_arr[2], bounds=denver_bounds_ext
```

```
[-105.24098, -104.664079, 40.022008, 39.333386]
[-105.24098, -104.664079, 40.022008, 39.333386]
/root/miniconda-4.12.0/envs/emission/lib/python3.7/site-packages/ipykernel_
launcher.py:10: RuntimeWarning: divide by zero encountered in log
  # Remove the CWD from sys.path while we load stuff.
/root/miniconda-4.12.0/envs/emission/lib/python3.7/site-packages/ipykernel_
launcher.py:10: RuntimeWarning: divide by zero encountered in log
  # Remove the CWD from sys.path while we load stuff.
[-105.24098, -104.664079, 40.022008, 39.333386]
/root/miniconda-4.12.0/envs/emission/lib/python3.7/site-packages/ipykernel_
launcher.py:10: RuntimeWarning: divide by zero encountered in log
  # Remove the CWD from sys.path while we load stuff.
```



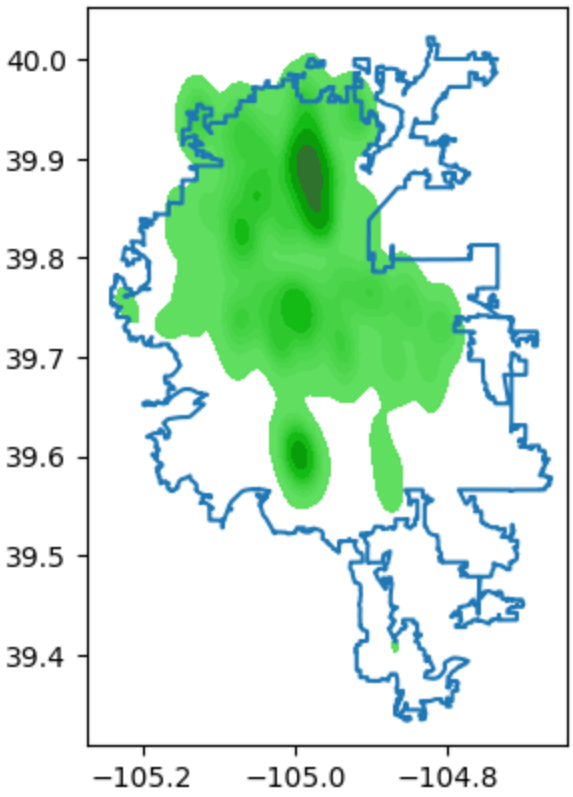

## Using seaborn

https://www.geeksforgeeks.org/kde-plot-visualization-with-pandas-and-seaborn/

In [47]:
```
import seaborn as sns
```

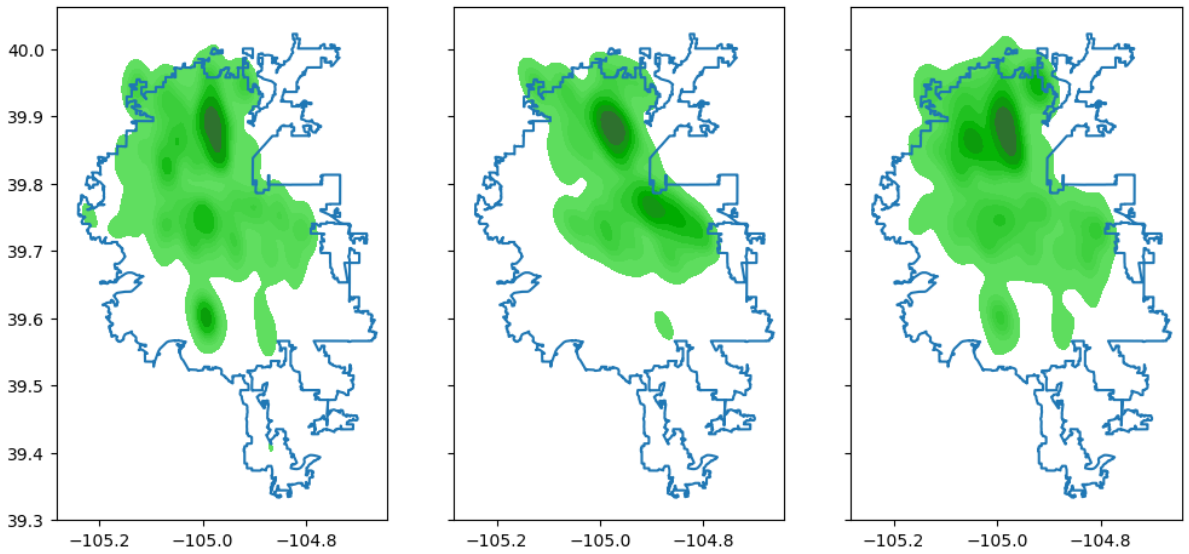
In [48]:
```
ax = sns.kdeplot(x=all_start_or_end_within.x, y=all_start_or_end_within.y, c
denver_boundary.boundary.plot(ax=ax)
```

Out[48]:
```
<AxesSubplot:>
```




In [49]:
```
fig, ax_arr = plt.subplots(nrows=1, ncols=3, sharex=True, sharey=True, figsi
denver_boundary.boundary.plot(ax=ax_arr[0])
sns.kdeplot(x=all_start_or_end_within.x, y=all_start_or_end_within.y, ax=ax_
denver_boundary.boundary.plot(ax=ax_arr[1])
sns.kdeplot(x=e_bike_start_or_end_within.x, y=e_bike_start_or_end_within.y,
denver_boundary.boundary.plot(ax=ax_arr[2])
sns.kdeplot(x=car_like_start_or_end_within.x, y=car_like_start_or_end_within
```

Out[49]: <AxesSubplot:>




In [50]:
```
all_geo_start_end_points.head()
```

Out[50]:
```
0    POINT (-107.85793 37.29581)
1    POINT (-107.79645 37.22962)
2    POINT (-107.86439 37.29887)
3    POINT (-107.86020 37.22296)
4    POINT (-107.87440 37.25015)
dtype: geometry
```

## Final, most complicated version using pixels, and computing the ratio of e-bike to car_like trips

In [51]:
```
# First, let's just make a dataframe with the three different counts: total,
def get_counts(pixel_polygon):
    all_trip_count = np.count_nonzero(all_geo_start_end_points.within(pixel_
    e_bike_trip_count = np.count_nonzero(e_bike_geo_start_end_points.within(
    car_like_trip_count = np.count_nonzero(car_like_geo_start_end_points.wit
    return pd.Series([all_trip_count, e_bike_trip_count, car_like_trip_count
```

### Test with a small dataset

In [52]:
```
test_pixels = denver_pixels.head(n=50).copy()
```

In [53]:
```
test_pixels
```

Out[53]:

| | id | lon | lat | geometry |
|---|---|---|---|---|
| **0** | 1 | -104.823128 | 40.017839 | POLYGON ((-104.82228 40.01201, -104.82276 40.0... |
| **1** | 2 | -104.818025 | 40.015812 | POLYGON ((-104.82098 40.02200, -104.81787 40.0... |
| **2** | 3 | -104.822497 | 40.005625 | POLYGON ((-104.82509 40.00201, -104.82454 40.0... |
| **3** | 4 | -104.817468 | 40.007237 | POLYGON ((-104.81300 40.01201, -104.81428 40.0... |
| **4** | 5 | -104.771824 | 40.003194 | POLYGON ((-104.77268 40.00201, -104.77268 40.0... |
| **5** | 6 | -104.767324 | 40.004074 | POLYGON ((-104.77098 40.00442, -104.77071 40.0... |
| **6** | 7 | -104.751980 | 40.005228 | POLYGON ((-104.75098 40.00350, -104.75328 40.0... |
| **7** | 8 | -104.747498 | 40.003922 | POLYGON ((-104.74908 40.00201, -104.74908 40.0... |
| **8** | 9 | -104.982229 | 39.994921 | POLYGON ((-104.98472 39.99291, -104.98121 39.9... |
| **9** | 10 | -104.976859 | 39.996055 | POLYGON ((-104.98098 40.00021, -104.98094 40.0... |
| **10** | 11 | -104.965638 | 39.995603 | POLYGON ((-104.97098 39.99499, -104.97057 39.9... |
| **11** | 12 | -104.921551 | 39.996931 | POLYGON ((-104.92215 40.00033, -104.92098 40.0... |
| **12** | 13 | -104.915754 | 39.996556 | POLYGON ((-104.92098 40.00033, -104.91098 40.0... |
| **13** | 14 | -104.907163 | 39.996168 | POLYGON ((-104.91098 40.00033, -104.90335 40.0... |
| **14** | 15 | -104.891552 | 39.992109 | POLYGON ((-104.89213 39.99201, -104.89210 39.9... |
| **15** | 16 | -104.886587 | 39.992474 | POLYGON ((-104.89098 39.99221, -104.88809 39.9... |
| **16** | 17 | -104.823920 | 39.996995 | POLYGON ((-104.82877 39.99201, -104.82877 39.9... |
| **17** | 18 | -104.816192 | 39.996684 | POLYGON ((-104.81526 40.00201, -104.81573 40.0... |
| **18** | 19 | -104.805979 | 39.996216 | POLYGON ((-104.81098 40.00042, -104.80098 40.0... |
| **19** | 20 | -104.795979 | 39.996219 | POLYGON ((-104.80098 40.00043, -104.79098 40.0... |
| **20** | 21 | -104.785979 | 39.996222 | POLYGON ((-104.79098 40.00043, -104.78098 40.0... |
| **21** | 22 | -104.775853 | 39.996377 | POLYGON ((-104.78098 40.00044, -104.77268 40.0... |
| **22** | 23 | -104.766166 | 39.996766 | POLYGON ((-104.76424 40.00201, -104.76478 40.0... |
| **23** | 24 | -104.755979 | 39.996241 | POLYGON ((-104.76098 40.00047, -104.75098 40.0... |
| **24** | 25 | -104.746415 | 39.996538 | POLYGON ((-104.75098 40.00048, -104.74907 40.0... |
| **25** | 26 | -104.739240 | 39.995667 | MULTIPOLYGON (((-104.74098 40.00050, -104.7352... |
| **26** | 27 | -105.004443 | 39.984592 | POLYGON ((-105.00679 39.98488, -105.00197 39.9... |
| **27** | 28 | -104.996341 | 39.982869 | MULTIPOLYGON (((-105.00098 39.98425, -104.9972... |
| **28** | 29 | -104.989236 | 39.983492 | MULTIPOLYGON (((-104.98098 39.99200, -104.9810... |
| **29** | 30 | -104.979189 | 39.991922 | POLYGON ((-104.97826 39.99201, -104.97826 39.9... |
| **30** | 31 | -104.921506 | 39.984189 | POLYGON ((-104.92204 39.98638, -104.92098 39.9... |
| **31** | 32 | -104.917921 | 39.985820 | MULTIPOLYGON (((-104.92098 39.98635, -104.9155... |
| **32** | 33 | -104.906229 | 39.990146 | POLYGON ((-104.90334 39.99201, -104.90333 39.9... |
| **33** | 34 | -104.892021 | 39.987019 | POLYGON ((-104.89098 39.98659, -104.89172 39.9... |

| | id | lon | lat | geometry |
|---|---|---|---|---|
| **34** | 35 | -104.885385 | 39.987995 | POLYGON ((-104.88447 39.98201, -104.88440 39.9... |
| **35** | 36 | -104.875936 | 39.986323 | POLYGON ((-104.88098 39.99051, -104.87969 39.9... |
| **36** | 37 | -104.866934 | 39.985790 | POLYGON ((-104.87098 39.99059, -104.86537 39.9... |
| **37** | 38 | -104.859280 | 39.985037 | POLYGON ((-104.86098 39.98675, -104.85622 39.9... |
| **38** | 39 | -104.832833 | 39.983777 | POLYGON ((-104.83719 39.98201, -104.83638 39.9... |
| **39** | 40 | -104.825706 | 39.986779 | POLYGON ((-104.83098 39.98796, -104.82918 39.9... |
| **40** | 41 | -104.815980 | 39.987008 | POLYGON ((-104.82098 39.99201, -104.81098 39.9... |
| **41** | 42 | -104.805980 | 39.987008 | POLYGON ((-104.81098 39.99201, -104.80098 39.9... |
| **42** | 43 | -104.795980 | 39.987008 | POLYGON ((-104.80098 39.99201, -104.79098 39.9... |
| **43** | 44 | -104.785980 | 39.987008 | POLYGON ((-104.79098 39.99201, -104.78098 39.9... |
| **44** | 45 | -104.776022 | 39.987029 | POLYGON ((-104.77098 39.98685, -104.77119 39.9... |
| **45** | 46 | -104.765503 | 39.988695 | POLYGON ((-104.76424 39.98201, -104.76425 39.9... |
| **46** | 47 | -104.755980 | 39.987008 | POLYGON ((-104.76098 39.99201, -104.75098 39.9... |
| **47** | 48 | -104.746042 | 39.987006 | POLYGON ((-104.74098 39.98934, -104.74133 39.9... |
| **48** | 49 | -104.739514 | 39.987208 | MULTIPOLYGON (((-104.73887 39.99201, -104.7384... |
| **49** | 50 | -105.052331 | 39.972034 | POLYGON ((-105.05381 39.97201, -105.05368 39.9... |

In [54]:
```
test_pixels[["all_trip_count", "e_bike_trip_count", "car_like_trip_count"]]
```

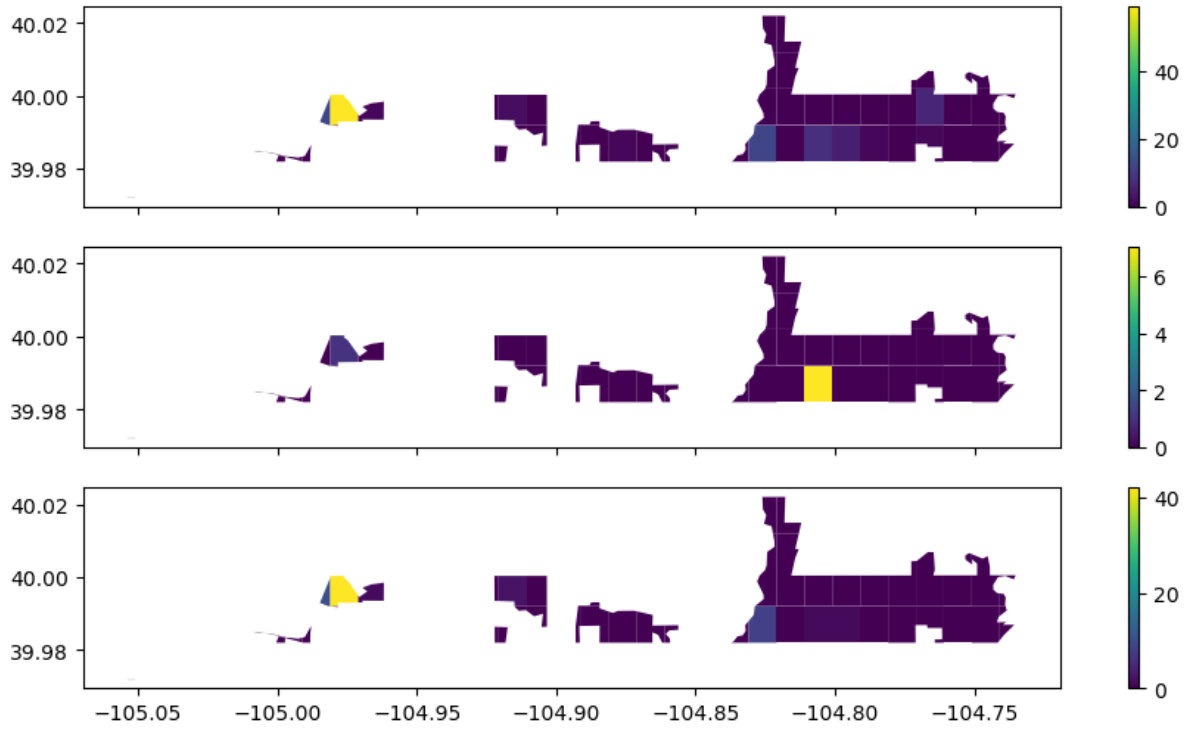
In [55]:
```
test_pixels.head()
```

Out[55]:

| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_like_trip_ |
|---|---|---|---|---|---|---|---|
| **0** | 1 | -104.823128 | 40.017839 | POLYGON ((-104.82228 40.01201, -104.82276 40.0... | 0 | 0 | |
| **1** | 2 | -104.818025 | 40.015812 | POLYGON ((-104.82098 40.02200, -104.81787 40.0... | 0 | 0 | |
| **2** | 3 | -104.822497 | 40.005625 | POLYGON ((-104.82509 40.00201, -104.82454 40.0... | 0 | 0 | |
| **3** | 4 | -104.817468 | 40.007237 | POLYGON ((-104.81300 40.01201, -104.81428 40.0... | 0 | 0 | |
| **4** | 5 | -104.771824 | 40.003194 | POLYGON ((-104.77268 40.00201, -104.77268 40.0... | 0 | 0 | |

In [56]:
```
fig, ax_arr = plt.subplots(nrows=3, ncols=1, figsize=(12,6), sharex=True, sh
test_pixels.plot(column="all_trip_count", legend=True, ax=ax_arr[0])
test_pixels.plot(column="e_bike_trip_count", legend=True, ax=ax_arr[1])
test_pixels.plot(column="car_like_trip_count", legend=True, ax=ax_arr[2])
```

Out[56]:
```
<AxesSubplot:>
```




In [57]:
```
test_pixels["e_bike_2_car_like"] = test_pixels.e_bike_trip_count / test_pixe
```

In [58]:
```
test_pixels.head()
```

Out[58]:

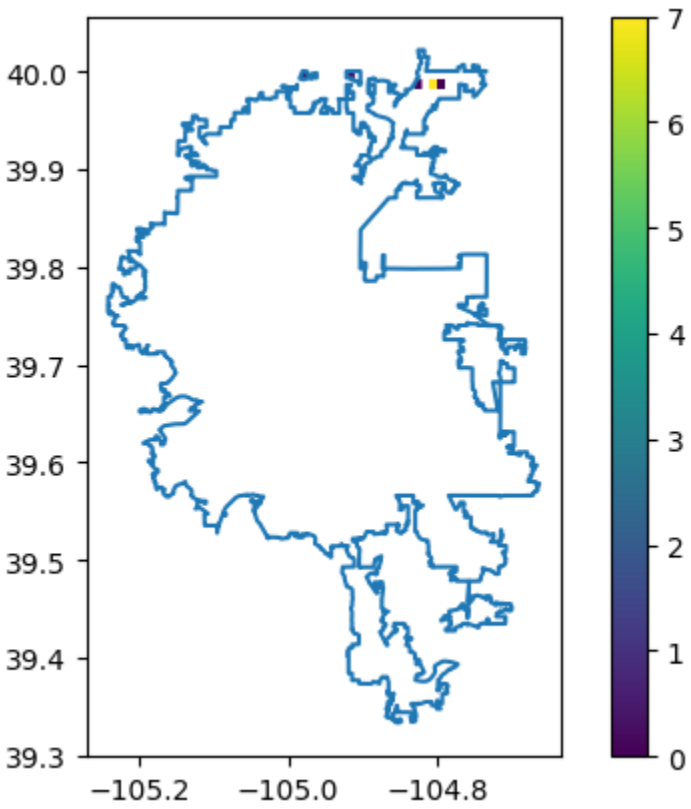
| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_like_trip_ |
|---|---|---|---|---|---|---|---|
| **0** | 1 | -104.823128 | 40.017839 | POLYGON ((-104.82228 40.01201, -104.82276 40.0... | 0 | 0 | |
| **1** | 2 | -104.818025 | 40.015812 | POLYGON ((-104.82098 40.02200, -104.81787 40.0... | 0 | 0 | |
| **2** | 3 | -104.822497 | 40.005625 | POLYGON ((-104.82509 40.00201, -104.82454 40.0... | 0 | 0 | |
| **3** | 4 | -104.817468 | 40.007237 | POLYGON ((-104.81300 40.01201, -104.81428 40.0... | 0 | 0 | |
| **4** | 5 | -104.771824 | 40.003194 | POLYGON ((-104.77268 40.00201, -104.77268 40.0... | 0 | 0 | |

In [59]:
```
test_pixels.e_bike_2_car_like.dropna()
```

Out[59]:
```
8     0.00000
9     0.02381
10    0.00000
12    0.00000
39    0.00000
41    7.00000
42    0.00000
Name: e_bike_2_car_like, dtype: float64
```

In [60]:
```
ax = denver_boundary.boundary.plot()
test_pixels.plot(column="e_bike_2_car_like", ax=ax, legend=True)
```

Out[60]: <AxesSubplot:>

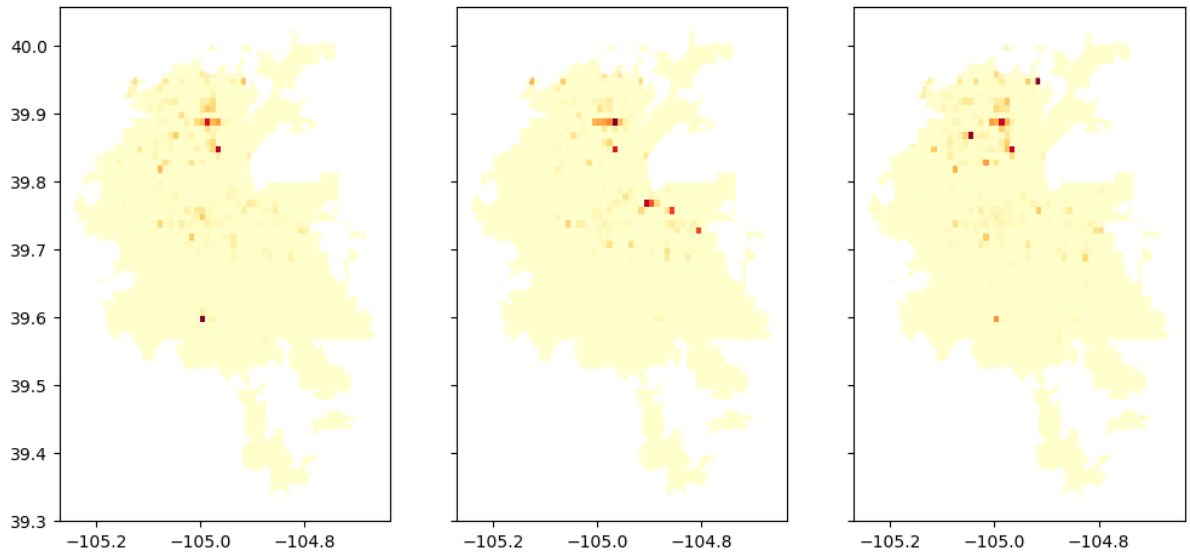



## And now for the final version

In [61]:
```
denver_pixels_all = denver_pixels.copy()
denver_pixels_all[["all_trip_count", "e_bike_trip_count", "car_like_trip_cou
```

In [62]:
```
fig, ax_arr = plt.subplots(nrows=1, ncols=3, figsize=(12,6), sharex=True, sh
denver_pixels_all.plot(column="all_trip_count", ax=ax_arr[0], cmap="YlOrRd")
denver_pixels_all.plot(column="e_bike_trip_count", ax=ax_arr[1], cmap="YlOrR
denver_pixels_all.plot(column="car_like_trip_count", ax=ax_arr[2], cmap="YlO
```

Out[62]: <AxesSubplot:>




In [63]:
```
denver_pixels_all["e_bike_2_car_like"] = denver_pixels_all.e_bike_trip_count
```

In [64]:
```
denver_pixels_all.replace(np.inf, denver_pixels_all.replace(np.inf, 0).e_bik
```

In [65]:
```
denver_pixels_all[denver_pixels_all.e_bike_2_car_like > 5]
```

Out[65]:

| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_ |
|---|---|---|---|---|---|---|---|
| **41** | 42 | -104.805980 | 39.987008 | POLYGON ((-104.81098 39.99201, -104.80098 39.9... | 8 | 7 | |
| **69** | 70 | -104.796013 | 39.977110 | POLYGON ((-104.79098 39.97232, -104.79775 39.9... | 2 | 2 | |
| **160** | 161 | -105.118677 | 39.936365 | MULTIPOLYGON (((-105.11857 39.94201, -105.1186... | 2 | 1 | |
| **161** | 162 | -105.105850 | 39.934014 | POLYGON ((-105.11098 39.93468, -105.10926 39.9... | 2 | 2 | |
| **167** | 168 | -105.045980 | 39.937008 | POLYGON ((-105.05098 39.94201, -105.04098 39.9... | 3 | 3 | |
| **181** | 182 | -104.905824 | 39.935694 | POLYGON ((-104.91098 39.93976, -104.91072 39.9... | 1 | 1 | |
| **194** | 195 | -105.105972 | 39.927009 | POLYGON ((-105.11098 39.92692, -105.10905 39.9... | 2 | 1 | |
| **277** | 278 | -104.955980 | 39.907008 | POLYGON ((-104.96098 39.91201, -104.95098 39.9... | 3 | 1 | |
| **311** | 312 | -104.955980 | 39.897008 | POLYGON ((-104.96098 39.90201, -104.95098 39.9... | 4 | 2 | |
| **313** | 314 | -104.935980 | 39.897008 | POLYGON ((-104.94098 39.90201, -104.93098 39.9... | 15 | 9 | |
| **347** | 348 | -104.955980 | 39.887008 | POLYGON ((-104.96098 39.89201, -104.95098 39.8... | 118 | 64 | |

| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_ |
|---|---|---|---|---|---|---|---|
| **361** | 362 | -104.815980 | 39.887008 | POLYGON ((-104.82098 39.89201, -104.81098 39.8... | 4 | 1 | |
| **385** | 386 | -104.955980 | 39.877008 | POLYGON ((-104.96098 39.88201, -104.95098 39.8... | 5 | 1 | |
| **386** | 387 | -104.945980 | 39.877008 | POLYGON ((-104.95098 39.88201, -104.94098 39.8... | 33 | 12 | |
| **387** | 388 | -104.935980 | 39.877008 | POLYGON ((-104.94098 39.88201, -104.93098 39.8... | 4 | 2 | |
| **423** | 424 | -104.945980 | 39.867008 | POLYGON ((-104.95098 39.87201, -104.94098 39.8... | 1 | 1 | |
| **496** | 497 | -104.896772 | 39.847908 | POLYGON ((-104.89098 39.84844, -104.89843 39.8... | 2 | 2 | |
| **526** | 527 | -104.925980 | 39.837008 | POLYGON ((-104.93098 39.84201, -104.92098 39.8... | 2 | 2 | |
| **559** | 560 | -104.915980 | 39.827008 | POLYGON ((-104.92098 39.83201, -104.91098 39.8... | 4 | 4 | |
| **663** | 664 | -104.975980 | 39.797008 | POLYGON ((-104.98098 39.80201, -104.97098 39.8... | 2 | 2 | |
| **667** | 668 | -104.935980 | 39.797008 | POLYGON ((-104.94098 39.80201, -104.93098 39.8... | 13 | 2 | |

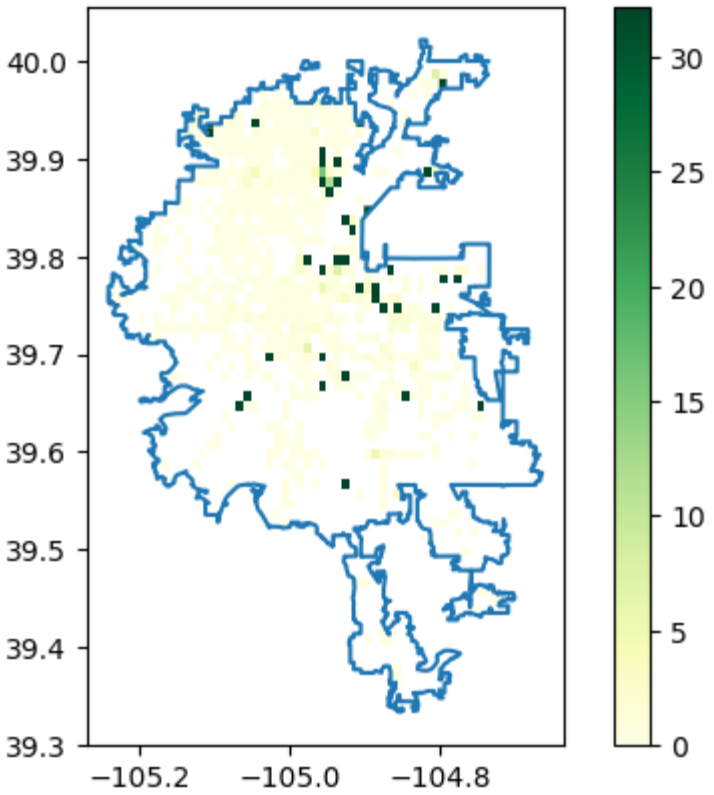
| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_ |
|---|---|---|---|---|---|---|---|
| **668** | 669 | -104.925980 | 39.797008 | POLYGON ((-104.93098 39.80201, -104.92098 39.8... | 7 | 1 | |
| **713** | 714 | -104.955980 | 39.787008 | POLYGON ((-104.96098 39.79201, -104.95098 39.7... | 13 | 11 | |
| **715** | 716 | -104.935980 | 39.787008 | POLYGON ((-104.94098 39.79201, -104.93098 39.7... | 88 | 15 | |
| **722** | 723 | -104.865980 | 39.787008 | POLYGON ((-104.87098 39.79201, -104.86098 39.7... | 17 | 10 | |
| **778** | 779 | -104.795980 | 39.777008 | POLYGON ((-104.80098 39.78201, -104.79098 39.7... | 5 | 1 | |
| **780** | 781 | -104.775980 | 39.777008 | POLYGON ((-104.78098 39.78201, -104.77098 39.7... | 6 | 2 | |
| **818** | 819 | -104.905980 | 39.767008 | POLYGON ((-104.91098 39.77201, -104.90098 39.7... | 236 | 161 | |
| **820** | 821 | -104.885980 | 39.767008 | POLYGON ((-104.89098 39.77201, -104.88098 39.7... | 84 | 44 | |
| **871** | 872 | -104.885980 | 39.757008 | POLYGON ((-104.89098 39.76201, -104.88098 39.7... | 11 | 1 | |
| **923** | 924 | -104.875980 | 39.747008 | POLYGON ((-104.88098 39.75201, -104.87098 39.7... | 7 | 3 | |

| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_ |
|---|---|---|---|---|---|---|---|
| **925** | 926 | -104.855980 | 39.747008 | POLYGON ((-104.86098 39.75201, -104.85098 39.7... | 2 | 2 | |
| **930** | 931 | -104.805980 | 39.747008 | POLYGON ((-104.81098 39.75201, -104.80098 39.7... | 6 | 4 | |
| **1114** | 1115 | -104.975980 | 39.707008 | POLYGON ((-104.98098 39.71201, -104.97098 39.7... | 59 | 37 | |
| **1152** | 1153 | -105.025980 | 39.697008 | POLYGON ((-105.03098 39.70201, -105.02098 39.7... | 20 | 5 | |
| **1159** | 1160 | -104.955980 | 39.697008 | POLYGON ((-104.96098 39.70201, -104.95098 39.7... | 1 | 1 | |
| **1252** | 1253 | -104.925980 | 39.677008 | POLYGON ((-104.93098 39.68201, -104.92098 39.6... | 6 | 2 | |
| **1294** | 1295 | -104.955980 | 39.667008 | POLYGON ((-104.96098 39.67201, -104.95098 39.6... | 2 | 2 | |
| **1331** | 1332 | -105.055980 | 39.657008 | POLYGON ((-105.06098 39.66201, -105.05098 39.6... | 14 | 2 | |
| **1352** | 1353 | -104.845980 | 39.657008 | POLYGON ((-104.85098 39.66201, -104.84098 39.6... | 7 | 1 | |
| **1375** | 1376 | -105.065980 | 39.647008 | POLYGON ((-105.07098 39.65201, -105.06098 39.6... | 2 | 1 | |

| | id | lon | lat | geometry | all_trip_count | e_bike_trip_count | car_ |
|---|---|---|---|---|---|---|---|
| **1407** | 1408 | -104.745980 | 39.647008 | POLYGON ((-104.75098 39.65201, -104.74098 39.6... | 2 | 1 | |
| **1790** | 1791 | -104.925980 | 39.567008 | POLYGON ((-104.93098 39.57201, -104.92098 39.5... | 4 | 2 | |

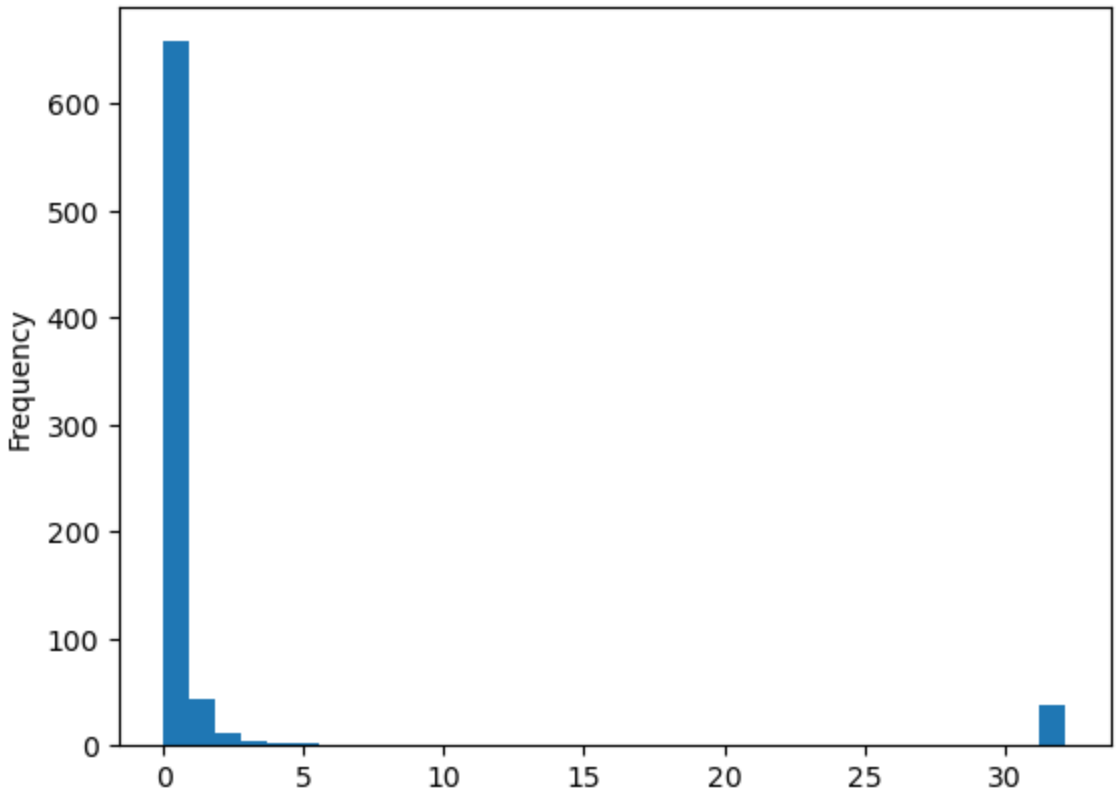
In [66]:
```
ax = denver_boundary.boundary.plot()
denver_pixels_all.plot(column="e_bike_2_car_like", ax=ax, cmap="YlGn", legen
```

Out[66]: <AxesSubplot:>




In [67]:
```
denver_pixels_all.e_bike_2_car_like.dropna().plot(kind="hist", bins=35)
```

Out[67]: <AxesSubplot:ylabel='Frequency'>




In [68]:
```
np.count_nonzero(denver_pixels_all.e_bike_2_car_like < 0.5), np.count_nonzer
```

Out[68]: (612, 154)

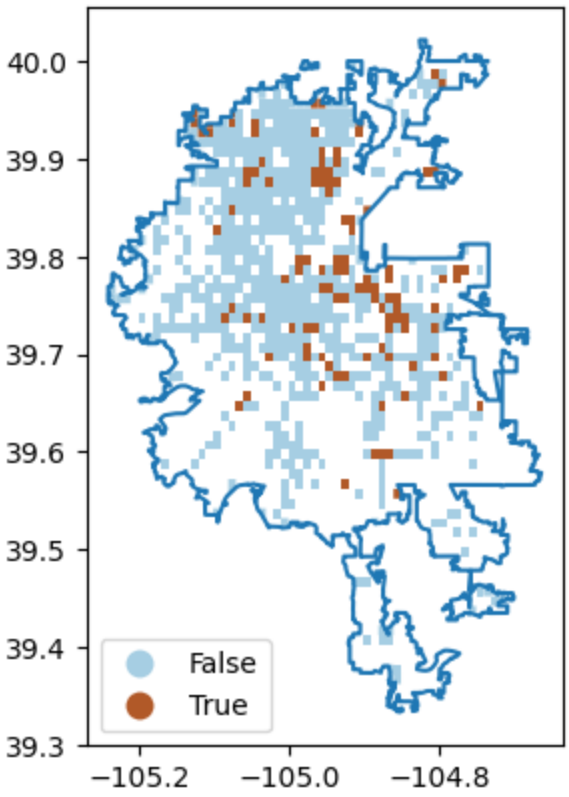
In [69]:
```
108 * 100 / 658
```

Out[69]: 16.41337386018237

In [70]:
```
denver_pixels_all["e_bike_better"] = pd.Categorical(denver_pixels_all.e_bike
```

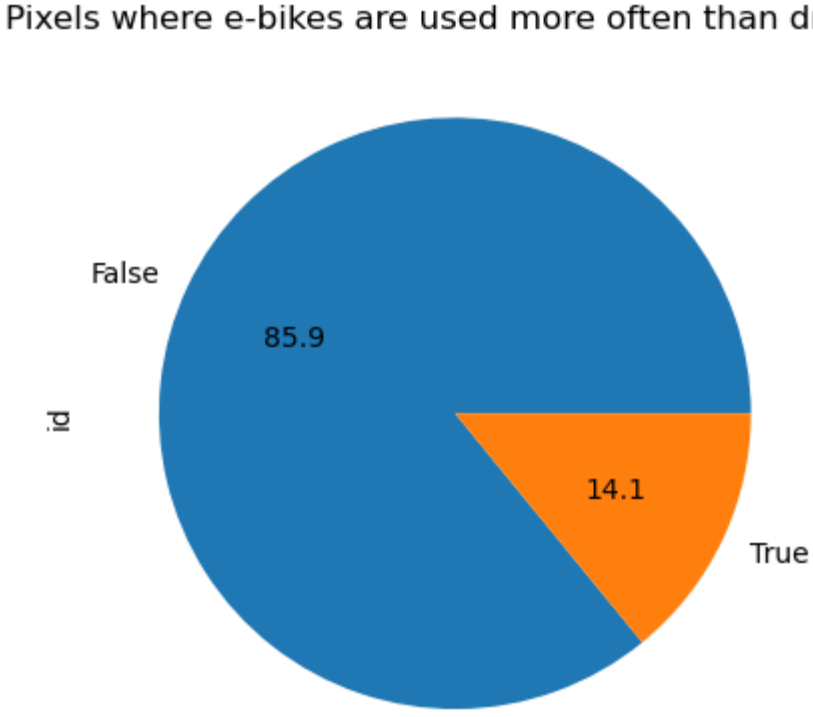
In [71]:
```
ax = denver_boundary.boundary.plot()
denver_pixels_all.dropna(axis='rows', how='any').plot(column = "e_bike_bette
```

Out[71]: <AxesSubplot:>




In [72]:
```
denver_pixels_all.dropna().groupby("e_bike_better").count().id.plot(kind="pi
```

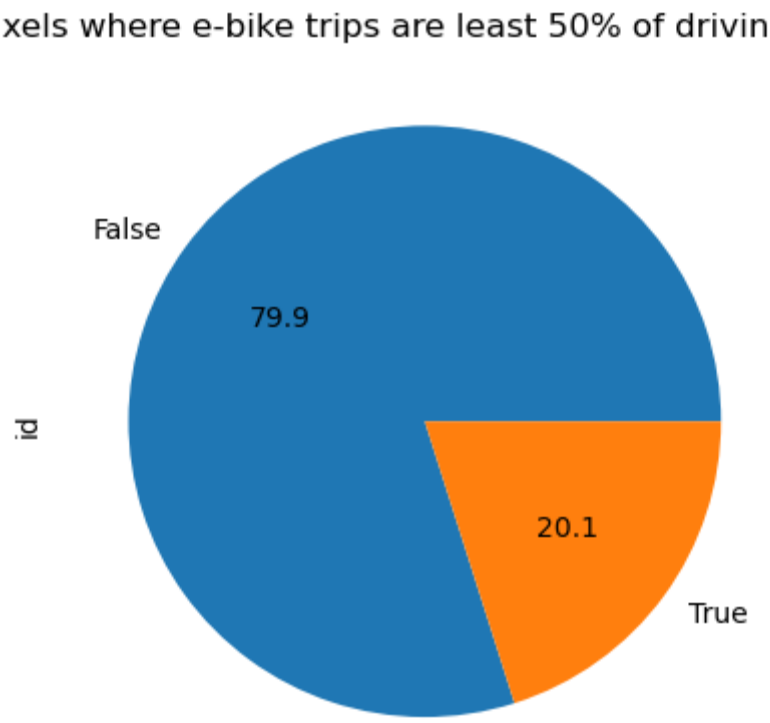
Out[72]:
```
<AxesSubplot:title={'center':'Pixels where e-bikes are used more often than
driving'}, ylabel='id'>
```




In [73]:
```
denver_pixels_all["e_bike_gt_50_pct"] = pd.Categorical(denver_pixels_all.e_b
```

In [74]:
```
denver_pixels_all.dropna().groupby("e_bike_gt_50_pct").count().id.plot(kind=
```

Out[74]:
```
<AxesSubplot:title={'center':'Pixels where e-bike trips are least 50% of dr
iving trips'}, ylabel='id'>
```




## Another complex check - does this differ across programs

In [75]:
```
def get_endpoints_within(trip_df):
    start_end_points = trip_df.start_loc.append(trip_df.end_loc)
    geo_start_end_points = gpd.GeoSeries(start_end_points.apply(lambda p: sh
    start_or_end_within = geo_start_end_points[geo_start_end_points.within(d
    return start_or_end_within
```

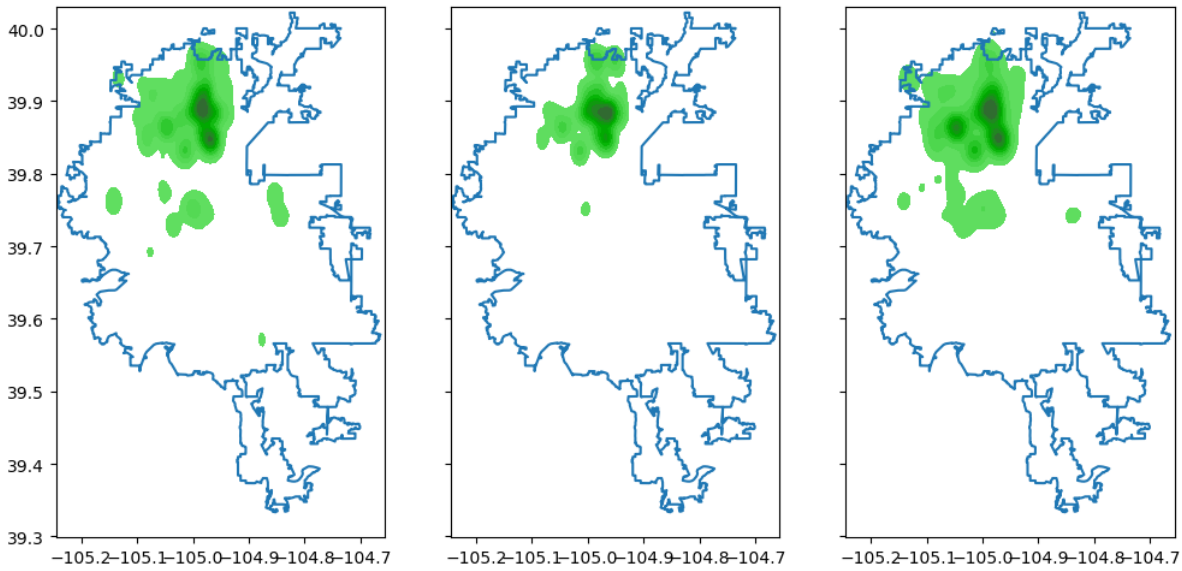
In [76]:
```
def get_all_e_bike_car_like(trip_df):
    return {"all": get_endpoints_within(trip_df),
            "e_bike": get_endpoints_within(trip_df.query('mode_confirm == "p
            "car_like": get_endpoints_within(trip_df.query('mode_confirm ==
           }
```

In [77]:
```
sc_trips_split = get_all_e_bike_car_like(trip_program_df.query("program == '
```

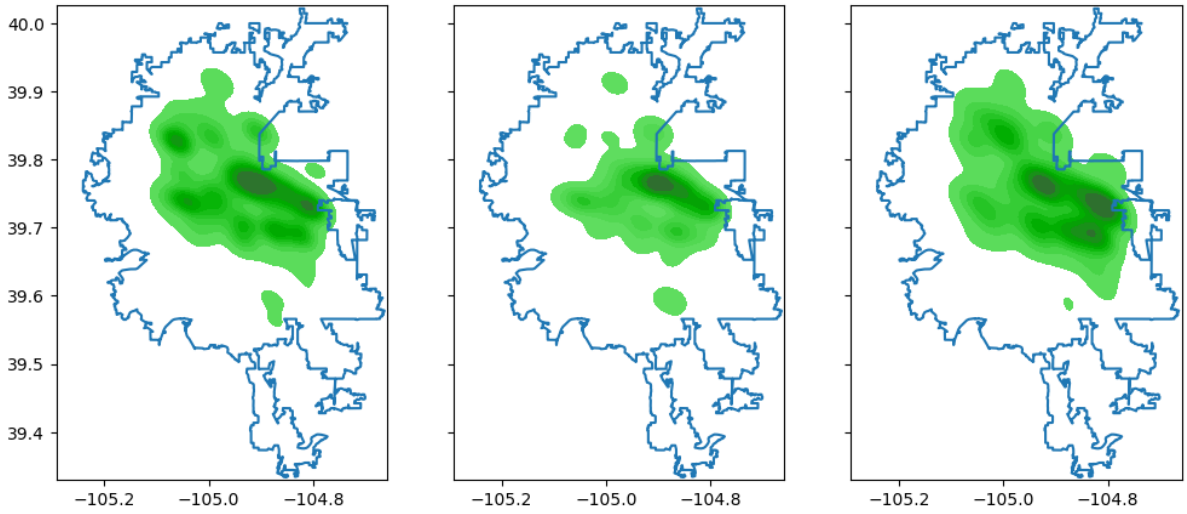
In [78]:
```
prepilot_trips_split = get_all_e_bike_car_like(trip_program_df.query("progra
```

In [79]:
```
def get_kde_plots(trips_split):
    fig, ax_arr = plt.subplots(nrows=1, ncols=3, sharex=True, sharey=True, f
    denver_boundary.boundary.plot(ax=ax_arr[0])
    sns.kdeplot(x=trips_split["all"].x, y=trips_split["all"].y, ax=ax_arr[0]
    denver_boundary.boundary.plot(ax=ax_arr[1])
    sns.kdeplot(x=trips_split["e_bike"].x, y=trips_split["e_bike"].y, ax=ax_
    denver_boundary.boundary.plot(ax=ax_arr[2])
    sns.kdeplot(x=trips_split["car_like"].x, y=trips_split["car_like"].y, ax
    # return fig
```

In [80]:
```
get_kde_plots(sc_trips_split)
```

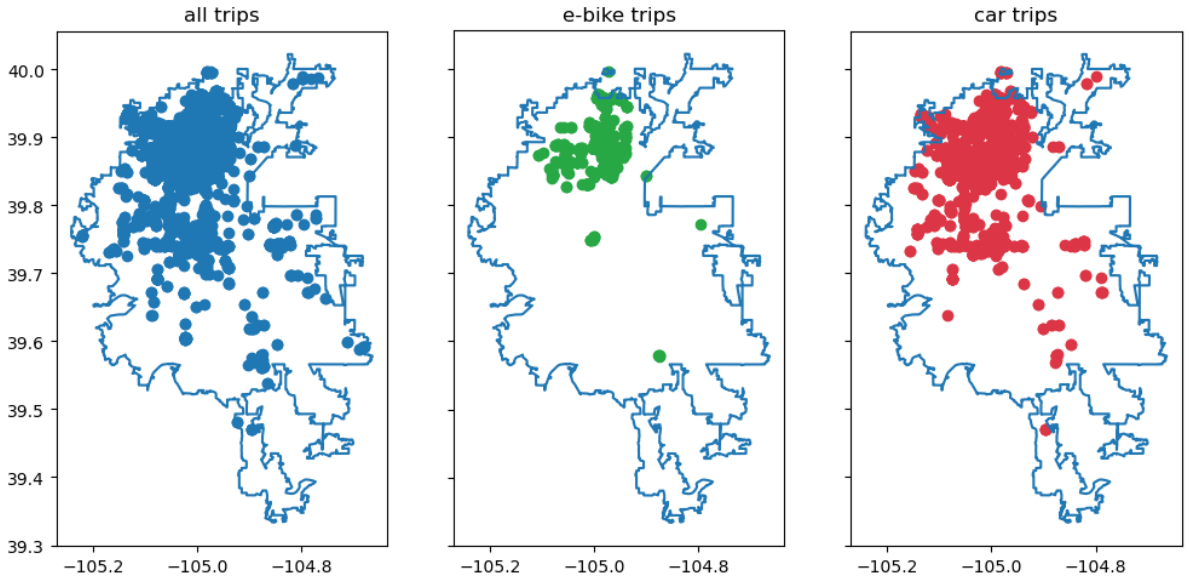



In [81]:
```
get_kde_plots(prepilot_trips_split)
```

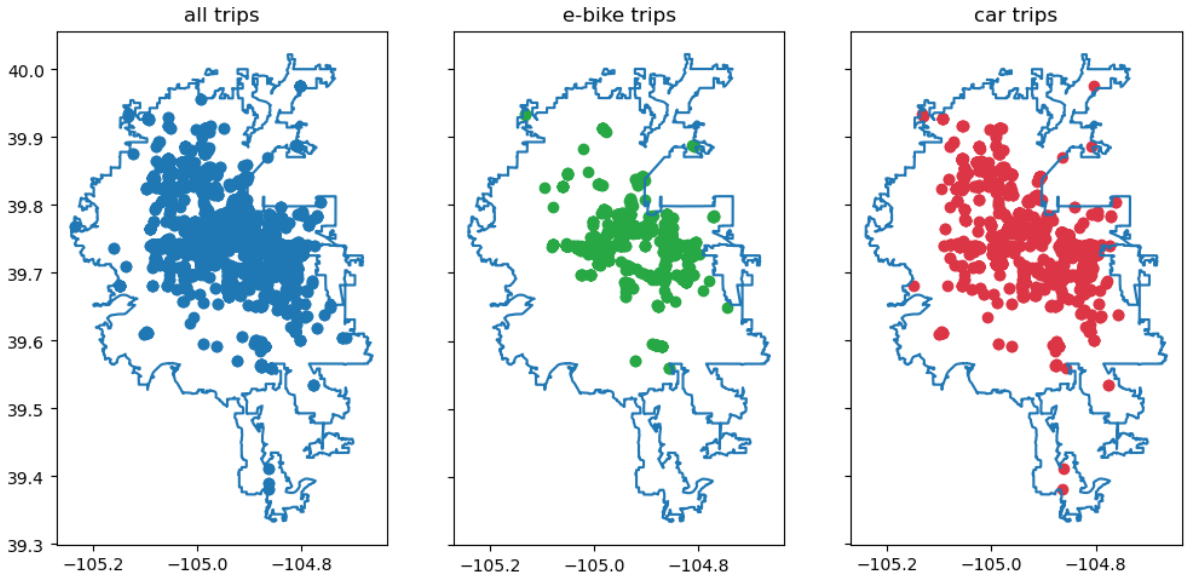


In [135…
```
def get_scatter_plots(trips_split):
    fig, ax_arr = plt.subplots(nrows=1, ncols=3, sharex=True, sharey=True, f
    denver_boundary.boundary.plot(ax=ax_arr[0])
    trips_split["all"].plot(ax=ax_arr[0])
    ax_arr[0].set_title("all trips")
    denver_boundary.boundary.plot(ax=ax_arr[1])
    trips_split["e_bike"].plot(color="#28a745", ax=ax_arr[1])
    ax_arr[1].set_title("e-bike trips")
    denver_boundary.boundary.plot(ax=ax_arr[2])
    trips_split["car_like"].plot(color="#dc3545", ax=ax_arr[2])
    ax_arr[2].set_title("car trips")
    # return fig
```

In [136…
```
get_scatter_plots(sc_trips_split)
```




In [137…
```
get_scatter_plots(prepilot_trips_split)
```



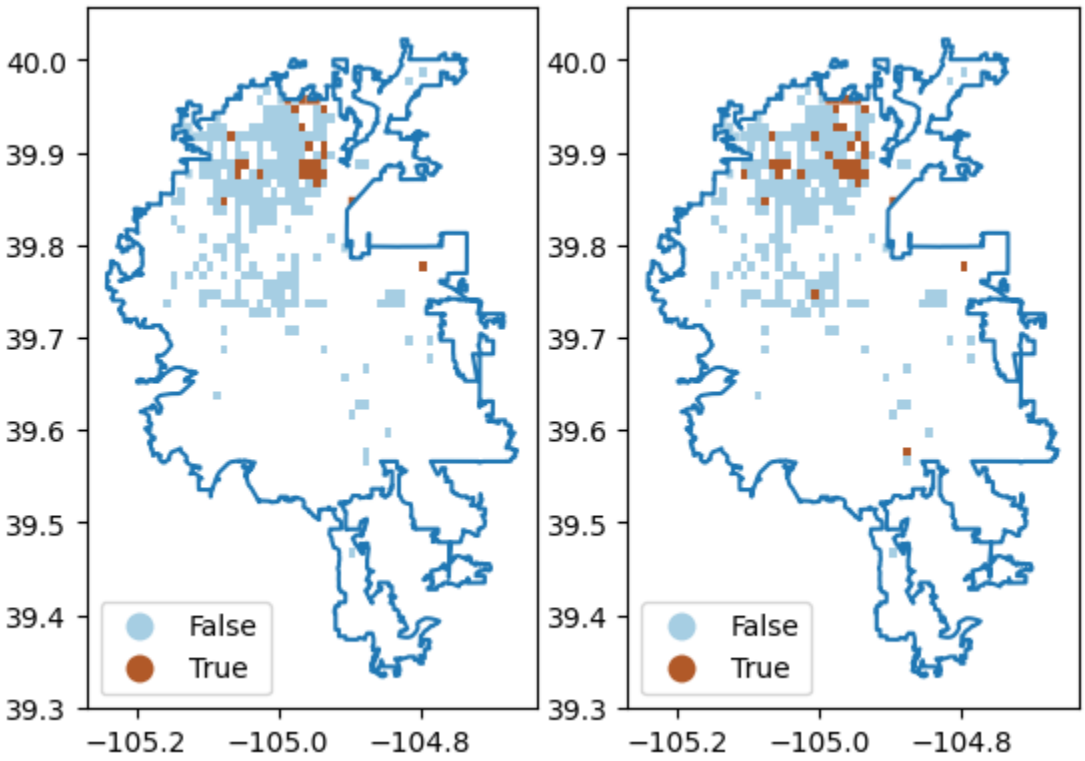
In [85]:
```
# First, let's just make a dataframe with the three different counts: total,
def get_counts_trips_split(trips_split, pixel_polygon):
    all_trip_count = np.count_nonzero(trips_split["all"].within(pixel_polygo
    e_bike_trip_count = np.count_nonzero(trips_split["e_bike"].within(pixel_
    car_like_trip_count = np.count_nonzero(trips_split["car_like"].within(pi
    return pd.Series([all_trip_count, e_bike_trip_count, car_like_trip_count
```

In [86]:
```
def get_pixel_stats(trips_split):
    curr_pixel_stats = denver_pixels.copy()
    curr_pixel_stats[["all_trip_count", "e_bike_trip_count", "car_like_trip_
    curr_pixel_stats["e_bike_2_car_like"] = curr_pixel_stats.e_bike_trip_cou
    curr_pixel_stats["e_bike_better"] = pd.Categorical(curr_pixel_stats.e_bi
    curr_pixel_stats["e_bike_gt_50_pct"] = pd.Categorical(curr_pixel_stats.e
    return curr_pixel_stats
```

In [87]:
```
sc_pixel_stats = get_pixel_stats(sc_trips_split)
```

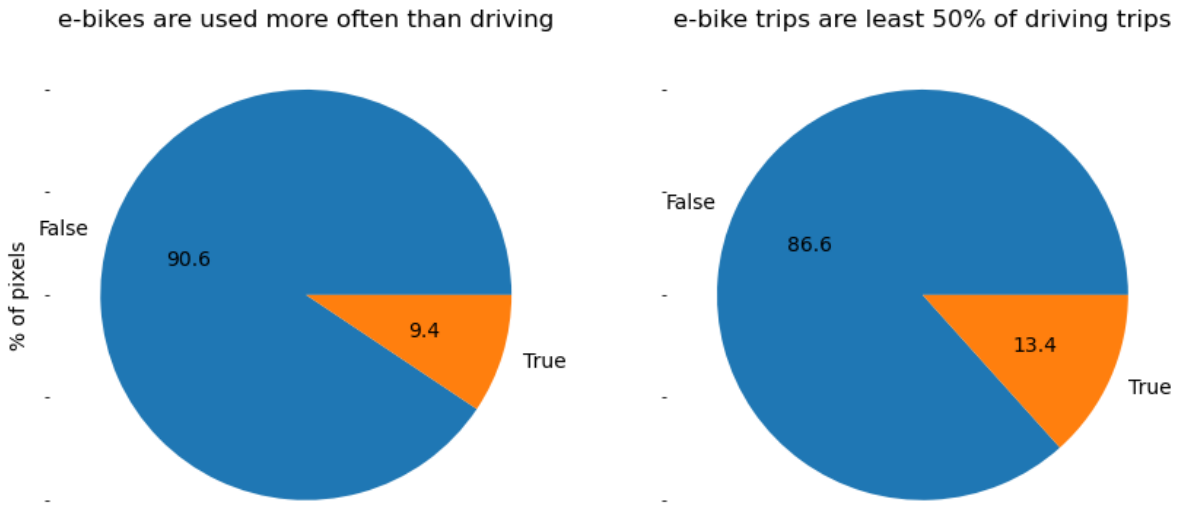
In [88]:
```
def show_pixel_stats(pixel_stats):
    fig, ax_arr = plt.subplots(nrows=1, ncols=2)
    denver_boundary.boundary.plot(ax=ax_arr[0])
    pixel_stats.dropna(axis='rows', how='any').plot(column = "e_bike_better"
    denver_boundary.boundary.plot(ax=ax_arr[1])
    pixel_stats.dropna(axis='rows', how='any').plot(column = "e_bike_gt_50_p
```

In [89]:
```
show_pixel_stats(sc_pixel_stats)
```



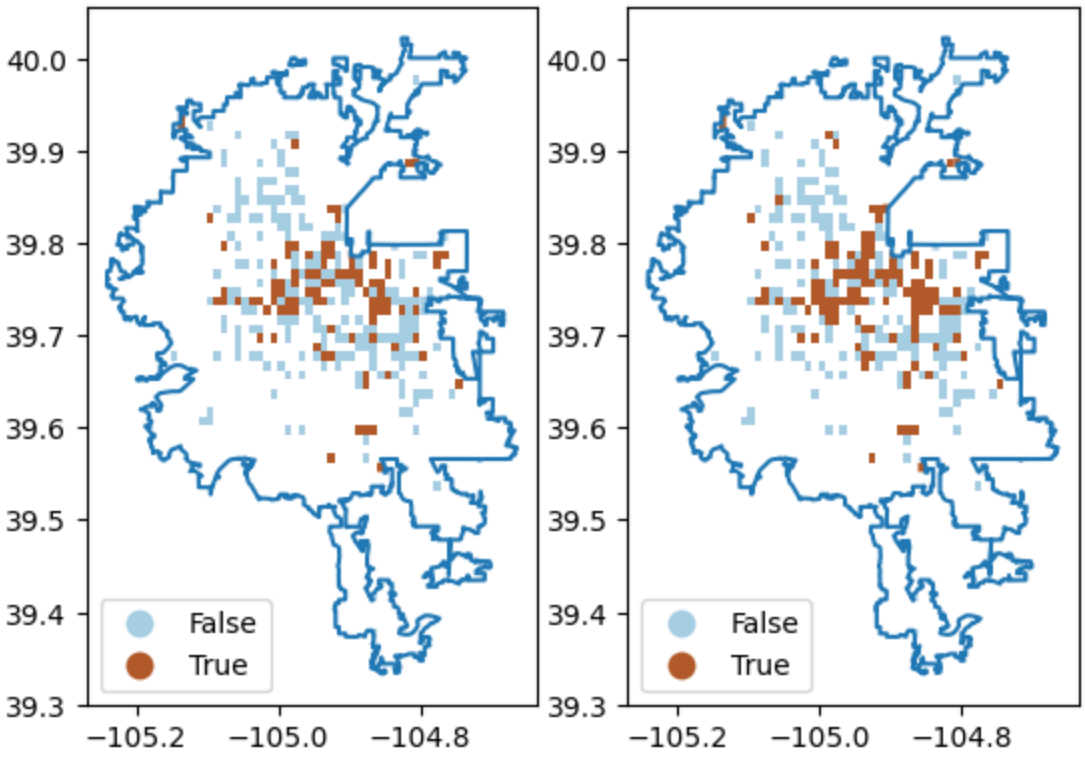
In [90]:
```
def show_pixel_percents(pixel_stats):
    fig, ax_arr = plt.subplots(nrows=1, ncols=2, sharex=True, sharey=True, f
    pixel_stats.dropna().groupby("e_bike_better").count().id.plot(kind="pie"
    pixel_stats.dropna().groupby("e_bike_gt_50_pct").count().id.plot(kind="p
    ax_arr[0].set_ylabel("% of pixels")
```

In [91]:
```
show_pixel_percents(sc_pixel_stats)
```

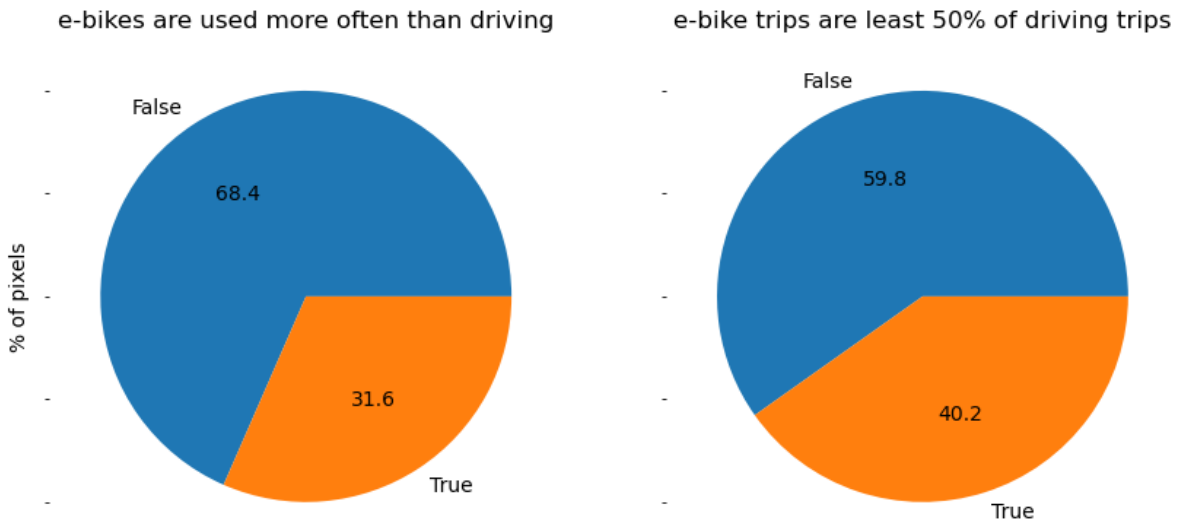



In [92]:
```
prepilot_pixel_stats = get_pixel_stats(prepilot_trips_split)
```

In [93]:
```
show_pixel_stats(prepilot_pixel_stats)
```

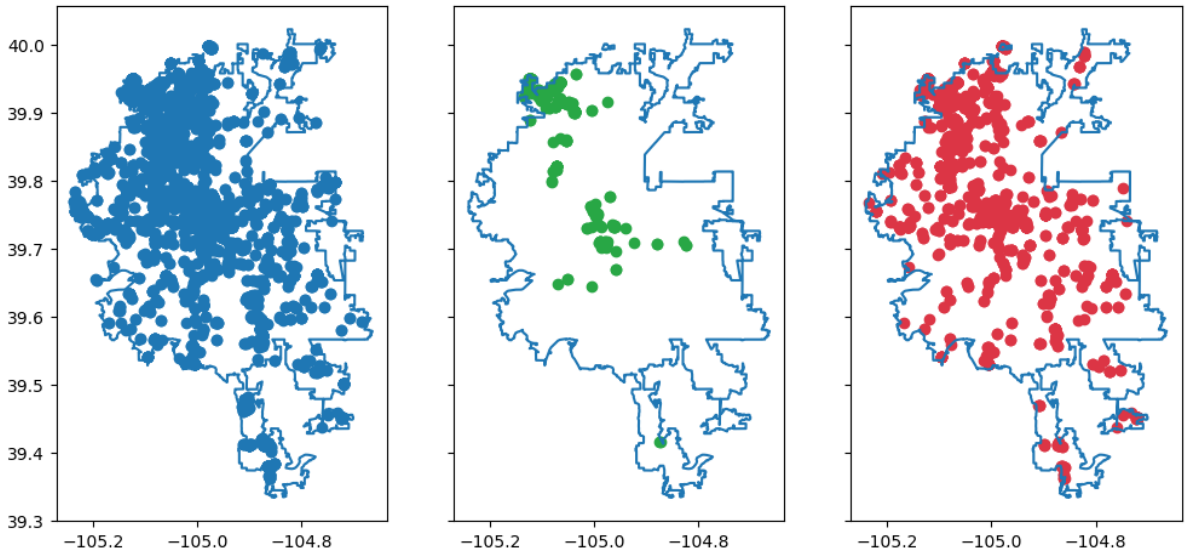



In [94]:
```
show_pixel_percents(prepilot_pixel_stats)
```



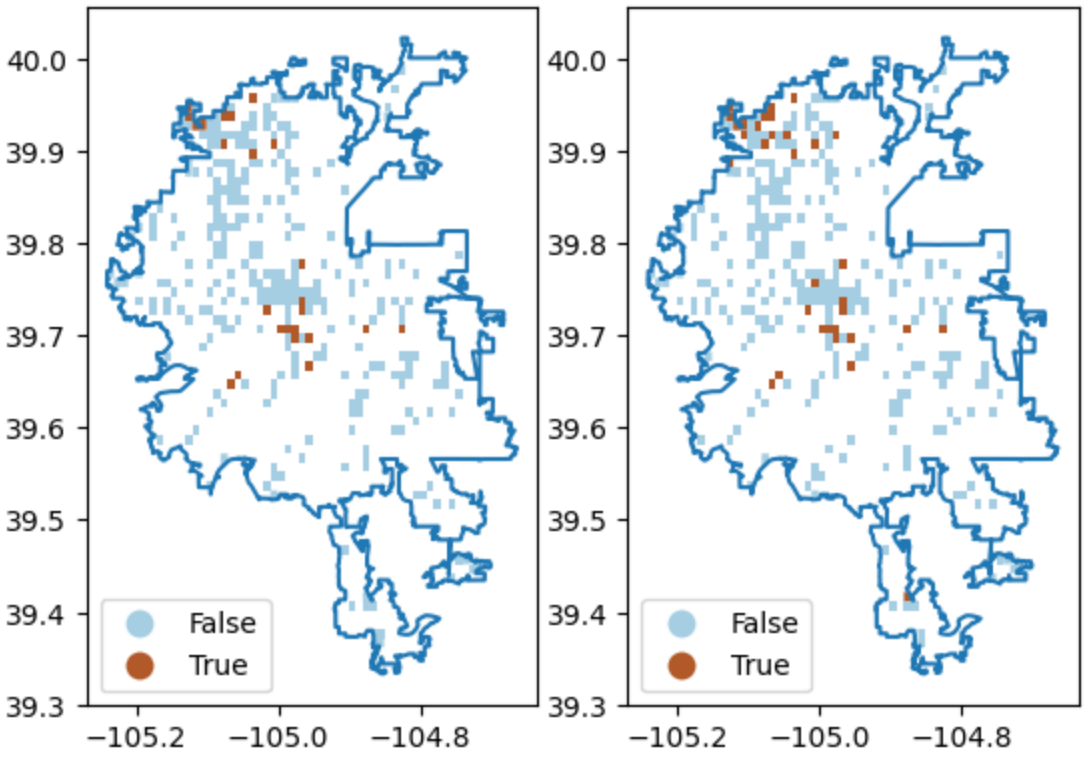

In [95]:
```
other_programs_split = get_all_e_bike_car_like(trip_program_df.query('progra
```

In [96]:
```
get_scatter_plots(other_programs_split)
```

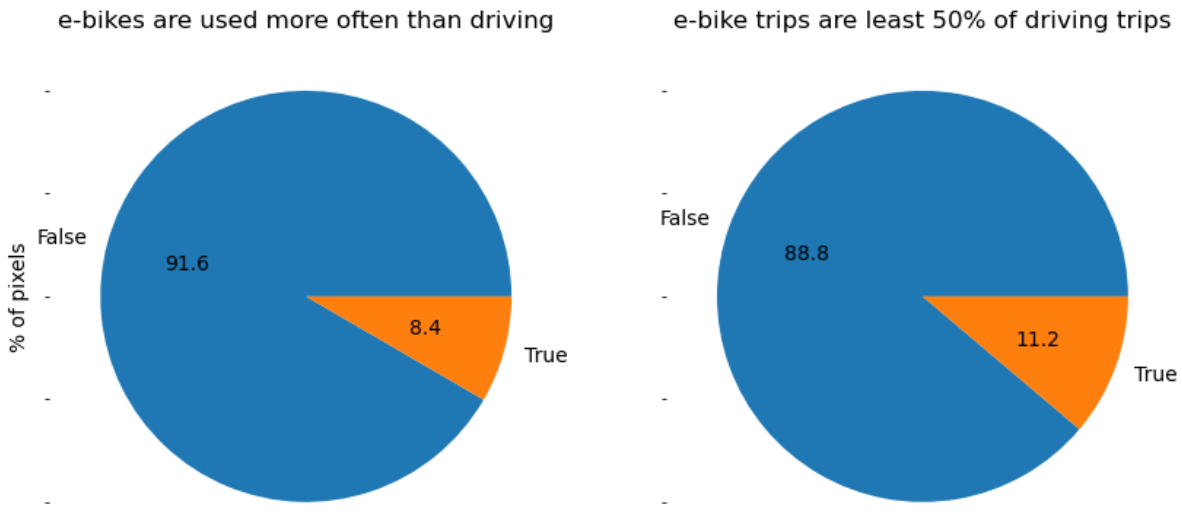



In [97]:
```
other_programs_pixel_stats = get_pixel_stats(other_programs_split)
```

In [98]:
```
show_pixel_stats(other_programs_pixel_stats)
```




In [99]:
```
show_pixel_percents(other_programs_pixel_stats)
```




## Experimenting with whether we can associate other fields with the start_end_loc points

In [111…
```
start_loc_df = trip_program_df[["program", "mode_confirm", "start_loc"]].cop
start_loc_df["type"] = ["start"] * len(start_loc_df)
start_loc_df.rename(columns = {"start_loc": "location"}, inplace=True)
start_loc_df.head()
```