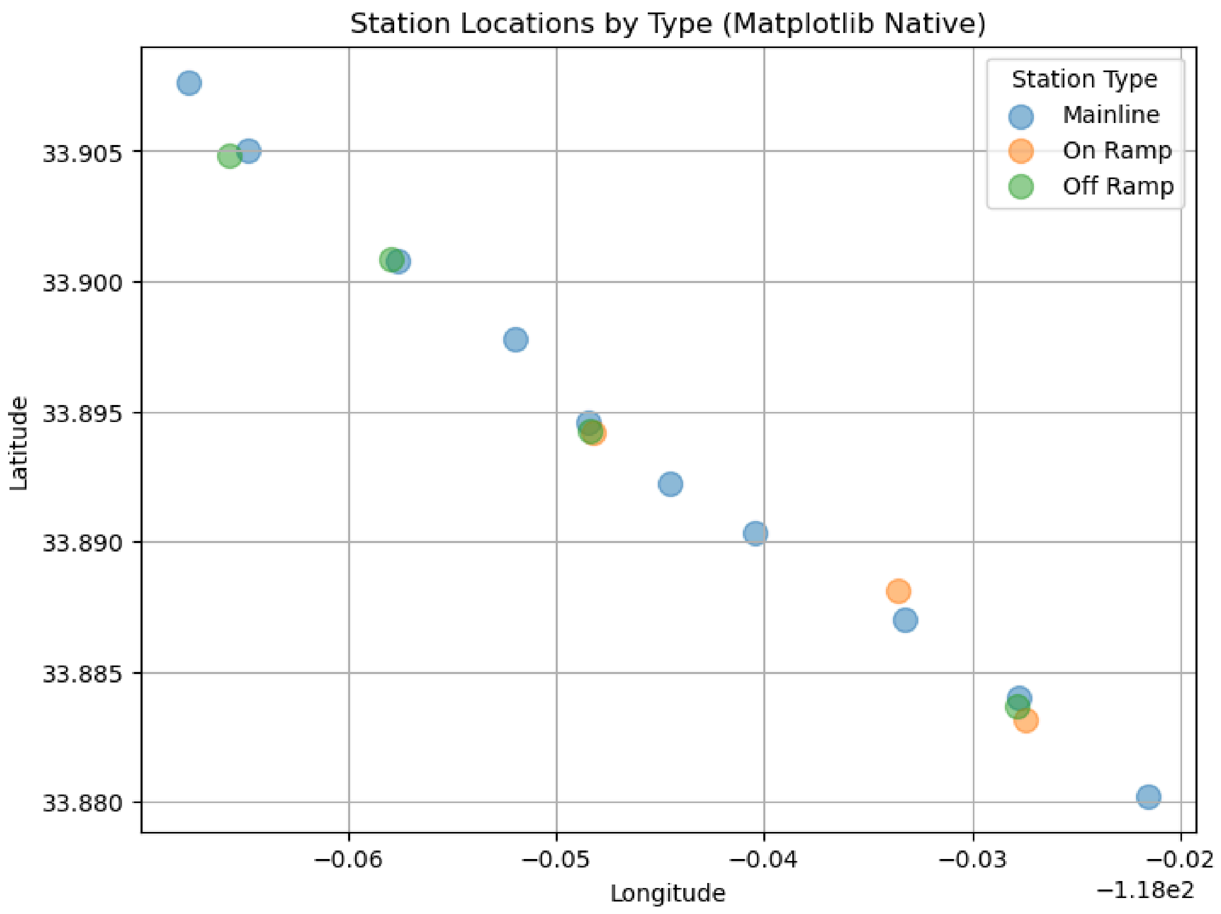
EDA on PeMS I5-N District 7 Station data

Authors: Drew Mortenson, ChatGPT

First I want to start by confirming that the latitude and longitude makes sense. If we plot all the stations on a graph of latitude/longitude, we expect to see a mostly straight line. There should not be significant curves, as the road (I5-N) passes diagonally northward through the LA area.

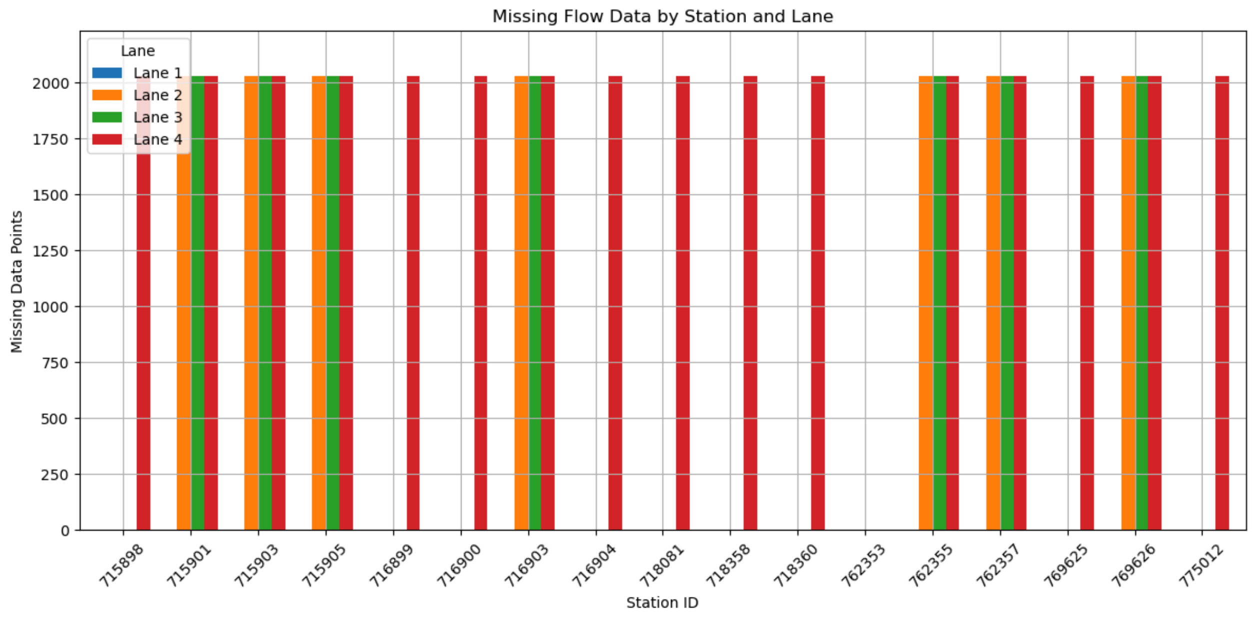
In [8]:
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

raw_data = pd.read_pickle("full_station_data.pkl")

station_df = (
    raw_data[["Station_ID", "Station_Type", "Latitude", "Longitude"]]
    .drop_duplicates(subset="Station_ID")
    .sort_values(by="Station_ID")
    .reset_index(drop=True)
)

# Fetch the station IDs, use that to form a new table with just station_id, latitude, longitude
station_ids = raw_data["Station_ID"].unique()

# Also fetch the station_types for iterating through
station_types = station_df["Station_Type"].unique()

# Arbitrary color selection
colors = plt.cm.tab10(range(len(station_types)))

plt.figure(figsize=(8, 6))
# Plot each group with a given color
for station_type, color in zip(station_types, colors):
    subset = station_df[station_df["Station_Type"] == station_type]
    x = subset["Longitude"]
    y = subset["Latitude"]

    # Jitter the points slightly, otherwise several stations overlap
    x_jitter = x + np.random.normal(0, 0.0005, len(x))
    y_jitter = y + np.random.normal(0, 0.0005, len(y))

    plt.scatter(
        x_jitter,
        y_jitter,
        alpha=0.5,
        label=station_type,
        color=color,
        s=100
    )

plt.title("Station Locations by Type (Matplotlib Native)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend(title="Station Type")
plt.grid(True)
plt.show()
```



These results look great! Nice and linear relationship between longitude and latitude, we can actually overlay this on google maps and see they align nicely. A small amount of jitter has been added to the datapoints to ensure they do not visually overlap despite being at the same geographic coordinates.

Now let's plot a grouped histogram to examine how data is missing. Only one station had a 4th lane, so we've imputed NA values for lane 4 in all stations except that one. Similarly, the on/off ramps all only have a single lane, so they contain NA values in lanes 2, 3, and 4. Plotting this grouped by station and sub-grouped by lane will enable us to explore precisely how much missing or bad data there is, as well as highlighting how we may need to proceed to ensure all this data works well together.

In [14]:
```
'''
This code was initially written by ChatGPT, later extended and modified by Drew
'''

import matplotlib.pyplot as plt
import pandas as pd

# Load
df = pd.read_pickle("full_station_data.pkl")

# Lanes to check
lanes = [1, 2, 3, 4]
col_template = "Lane {} Flow (Veh/5 Minutes)"

records = []
for station_id, group in df.groupby("Station_ID"):
    total = len(group)
    for i in lanes:
        col = col_template.format(i)
        if col in group.columns:
            # count cells that are either NaN or blank strings
            val = group[col]
            missing = ((val.isna()) | (val.astype(str).str.strip() == "")).sum()
        else:
            missing = 0
        records.append({
            "Station_ID": station_id,
            "Lane": f"Lane {i}",
            "Missing": missing
        })

missing_df = pd.DataFrame(records)

# Pivot into wide form
pivot = missing_df.pivot(index="Station_ID", columns="Lane", values="Missing").fillna(0)

# Plot
ax = pivot.plot(kind="bar", figsize=(12, 6), width=0.8)
ax.set_title("Missing Flow Data by Station and Lane")
ax.set_xlabel("Station ID")
ax.set_ylabel("Missing Data Points")
ax.set_xticklabels(pivot.index, rotation=45)
ax.legend(title="Lane")
ax.grid(True)

# Only force y≥0 if there is nonzero missing data
ymax = pivot.values.max()
if ymax > 0:
    ax.set_ylim(0, ymax * 1.1)

plt.tight_layout()
plt.show()
```

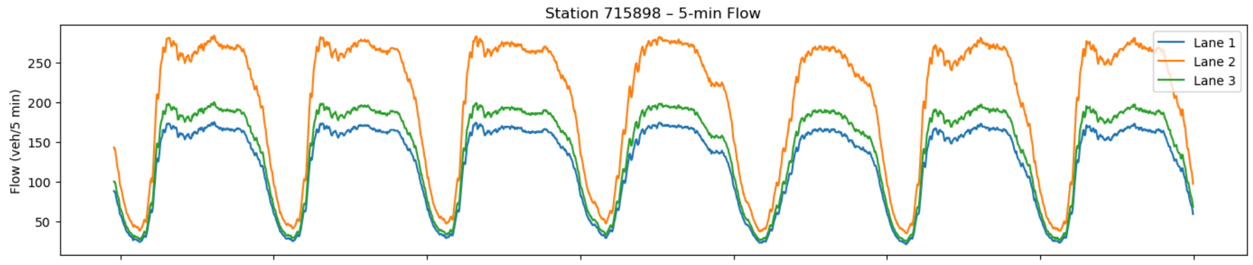


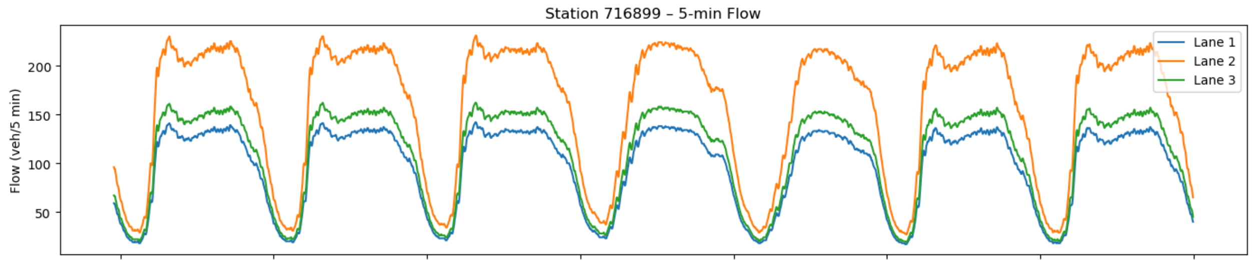
This output is a little confusing, so let's break it down, because there are 3 visual types of missingness.

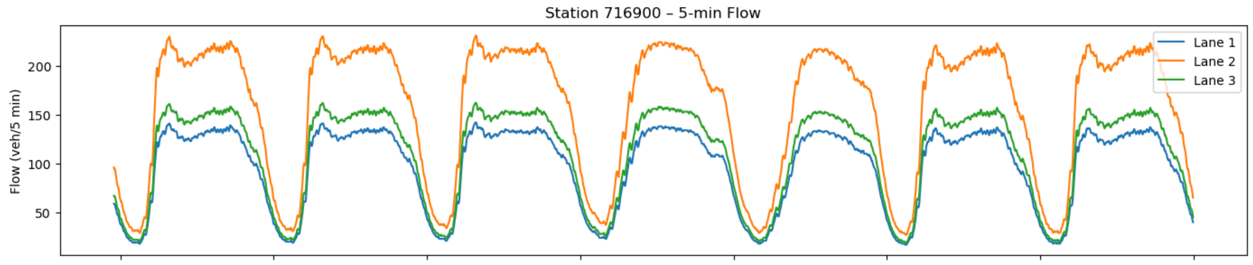
1. 762353 is the only station (Mainline or otherwise, but 762353 is Mainline) which does not report missing lane 4, which is accurate to what I saw while downloading and preprocessing the data.
2. The other stations with only a red bar are Mainline stations, they have lanes 1, 2, and 3, but not lane 4, and thus are missing all data for it.
3. The stations with 3 bars, orange, green, and red, are on/off ramp stations, and they only have lane 1, hence the other 3 lanes are missing.

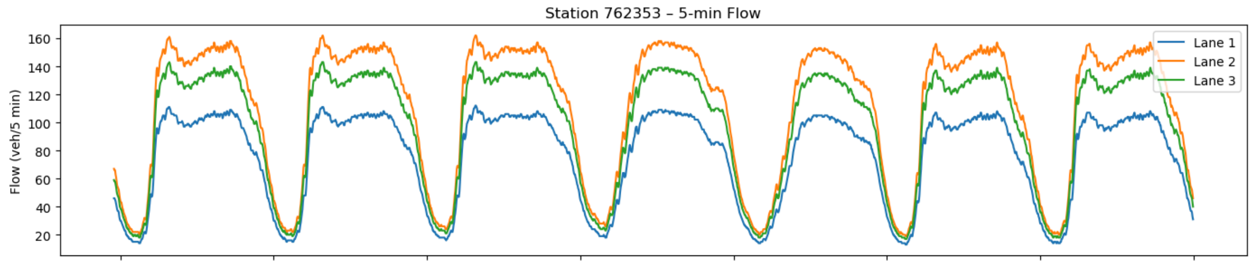
All in all, this visual output is precisely what we expect to see, and it confirms that there are 0 missing data points outside of entirely missing lanes. Now that we know what to expect in terms of missingness, let's examine the flow over time at each station and lane.

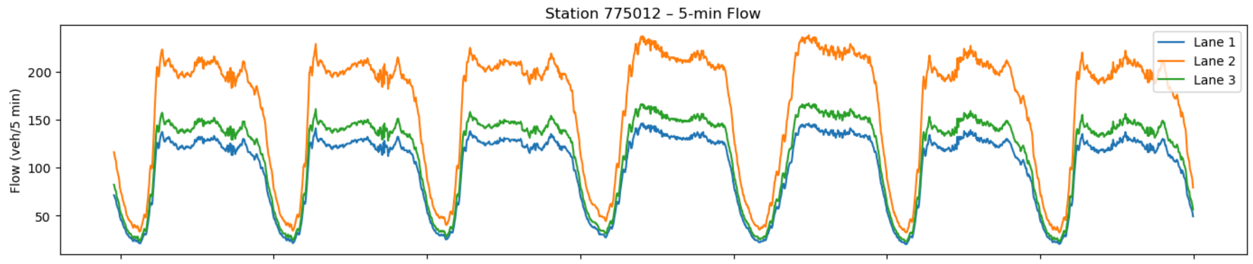
In [ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Load and prep
df = pd.read_pickle("full_station_data.pkl")
df["5 Minutes"] = pd.to_datetime(df["5 Minutes"])

# Keep only mainline stations
main = df[df["Station_Type"] == "Mainline"]

stations = main["Station_ID"].unique()[:10]

# Set up subplots
n = len(stations)
fig, axes = plt.subplots(n, 1, figsize=(14, 3*n), sharex=True)

for ax, sid in zip(axes, stations):
    sub = main[main["Station_ID"] == sid].sort_values("5 Minutes")
    # Plot lanes 1–3
    for lane in [1, 2, 3]:
        col = f"Lane {lane} Flow (Veh/5 Minutes)"
        if col in sub.columns:
            ax.plot(sub["5 Minutes"], sub[col], label=f"Lane {lane}")
    ax.set_title(f"Station {sid} – 5-min Flow")
    ax.set_ylabel("Flow (veh/5 min)")
    ax.legend(loc="upper right")

# Final touches
axes[-1].set_xlabel("Time")
plt.tight_layout()
plt.show()
```

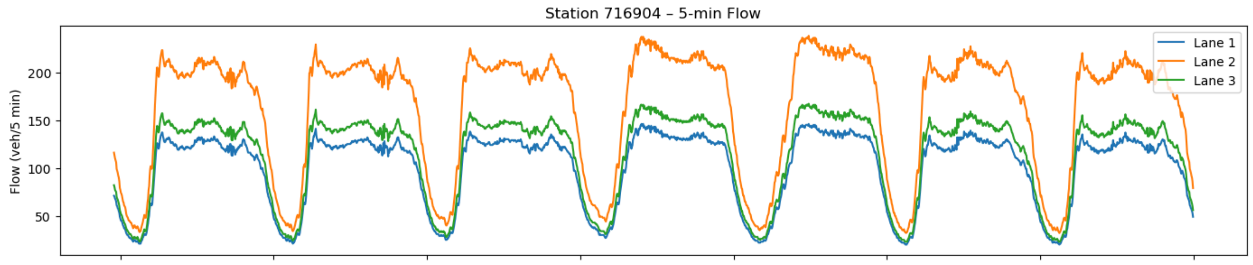

Station 715898 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3

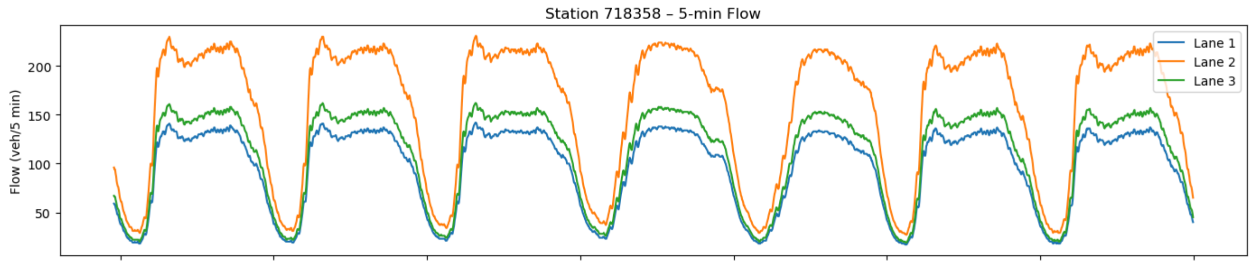

Station 716899 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3


Station 716900 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3

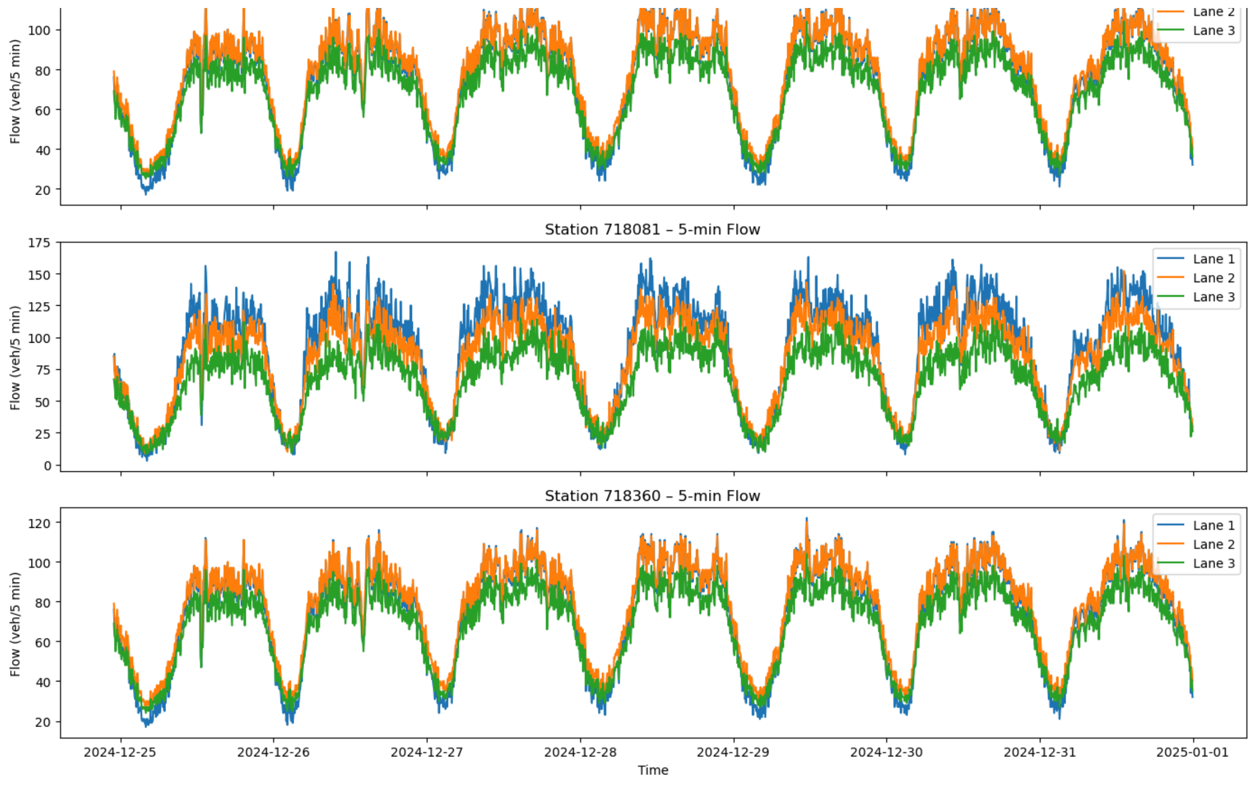

Station 762353 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3


Station 775012 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3


Station 716904 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3


Station 718358 – 5-min Flow
Flow (veh/5 min)
Lane 1
Lane 2
Lane 3

Station 769625 – 5-min Flow
120
Lane 1



Analysis

In general we see clear and strongly-defined rush-hour patterns, with visual traffic minimums around 1-2 AM and maximums stretching from 12 PM to 10-11:59 PM. We should definitely find min and max rows of flow to confirm this analysis. We will further explore the minimum and maximum values of each stations flow, reporting the hours at which these occur, to confirm our visual analysis.

We find unusual flow, especially so, for lane 2 at stations 718360, 718081, and 769625

- Large amount of variation in flow quantities between measurements (more jagged and up/down compared to other stations)
- Minimal to non-existant gap between flow in lanes, especially compared to other stations
- Lane 1 is actually the highest flow, and lane 3 the lowest flow at 718081, which is unusual compared to other stations, which report lane 1 with lowest flow and lane 3 with highest flow. This trend is consistent across the day.
  - Perhaps people are trying to get off the highway at this station? Large quantity of traffic merging into lane 1 and exiting?
  - 769625, 718360 both reflect similar trends during peak-flow hours, lane 2 and lane 1 seem to share a roughly equivalent flow, while lane 3 has a slightly lower flow (a trend which is reversed during minimum flow hours, with lane 2 being max, lane 3 being average, and lane 1 being low). This trend is remarkably consistent between the two.
- Examining this further, 718360 is the last station in the set (south-north), 718081 is the second to last, and 769625 is the third to last. THESE STATIONS OCCUR AROUND A MAJOR INTERCHANGE/AREA WITH MANY OFF/ON RAMPS!!!!!
  - In particular, 769625 occurs right before (or precisely at, hard to say) the first off-ramp of the intersection/interchange thingy, 718081 occurs right at the end of the subsequent on-ramp (or precisely at, hard to say), and 718360 occurs a little ways down the road (about 1200 ft down the road).

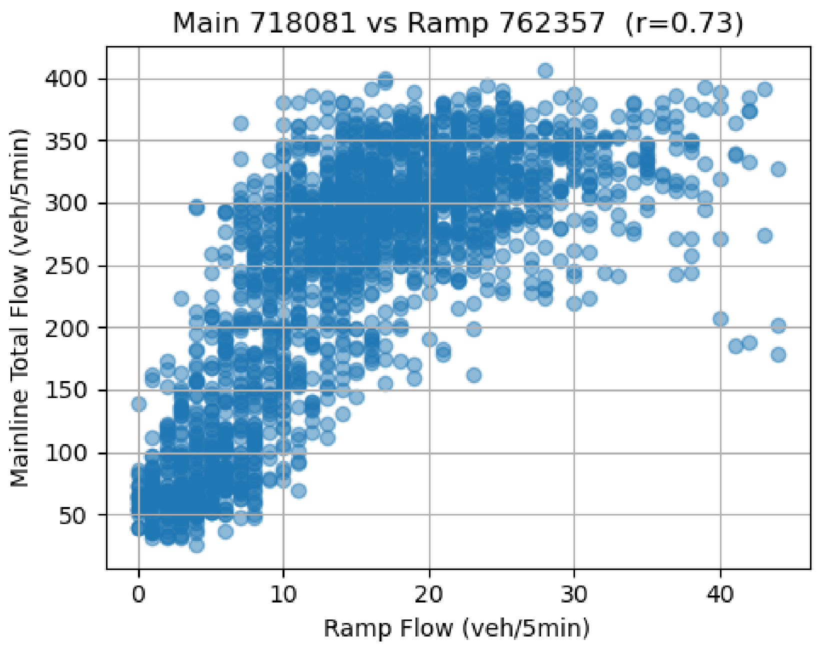
So it seems this oddity is related to the on/off ramps, perhaps this area relates to a major intersection more than other areas with off ramps. The related stations for on/off ramps in this stretch are:

- 769626 is the first off ramp, and is associated with the 769625 station (reporting Mainline flow), reporting off-ramp flow at that station
- 762357 is listed as an "off ramp" on PeMS, but based on the fact that it is recorded for I5-N, and is in the same location as 769625, this is not physically possible. **I'm going to infer that this was mislabeled, and should be considered an on ramp. I'm relabeling within data_preprocessing and saving to a new file. The old file has been copied and saved as "full_station_data_bad_label.csv/pkl".**

762353 - Of minor interest, but not as much as the concerning/highly different output from the 3 aforementioned stations. Relative to other stations, 762353 records a more equitable distribution across lanes 1, 2, and 3, and the gap between them is visually constant.

In [17]:
```python
'''
After I wrote the above analysis and provided it to ChatGPT, I had it help me write this section to e
This code is entirely written by ChatGPT.
'''

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# --- Load & prep ------------------------------------------------------------
df = pd.read_pickle("full_station_data.pkl")
df["5 Minutes"] = pd.to_datetime(df["5 Minutes"])

# --- A) Rush-hour & trough hours (Mainline only, exclude hour=0) ------------
# Filter to mainline stations
mainline = df[df["Station_Type"] == "Mainline"].copy()
mainline["hour"] = mainline["5 Minutes"].dt.hour

# Global mean Lane-1 flow by hour
hourly = mainline.groupby("hour")["Lane 1 Flow (Veh/5 Minutes)"]\
                 .mean()\
                 .sort_index()
valid = hourly.loc[hourly.index != 0]

print("Overall min-flow hour:", hourly.idxmin(), "→", hourly.min())
print("Overall max-flow hour:", valid.idxmax(), "→", valid.max())

# Per-station mean by (Station_ID, hour)
station_hourly = (
    mainline
    .groupby(["Station_ID", "hour"])["Lane 1 Flow (Veh/5 Minutes)"]
    .mean()
    .reset_index(name="mean_flow")
)

# Exclude hour 0 for peak/trough
sh = station_hourly[station_hourly["hour"] != 0]

# Peak hour per station
peak = (
    sh.loc[sh.groupby("Station_ID")["mean_flow"].idxmax()]
    .set_index("Station_ID")["hour"]
)

# Low hour per station
low = (
    sh.loc[sh.groupby("Station_ID")["mean_flow"].idxmin()]
    .set_index("Station_ID")["hour"]
)

print("\nPer-station peak hours (1-23):")
print(peak.to_frame("Peak_Hour").T)
print("\nPer-station low hours (1–23):")
print(low.to_frame("Low_Hour").T)

# --- C+D) Merge & correlate with actual ramps -----------------------------
# Build ramp_df from on/off-ramps
ramp_df = df[df["Station_Type"].isin(["On Ramp", "Off Ramp"])][
    ["5 Minutes", "Station_ID", "Lane 1 Flow (Veh/5 Minutes)"]
].rename(columns={"Lane 1 Flow (Veh/5 Minutes)": "RampFlow"})

print("\nRamp station IDs found:", ramp_df["Station_ID"].unique())

# Define mainline↔ramp pairs to test
pairs = {
    769625: 769626,  # Mainline 769625 ↔ its off-ramp 769626
    718081: 762357,  # Mainline 718081 ↔ (re-labeled) on-ramp 762357
}

for main_id, ramp_id in pairs.items():
    print(f"\n=== Main {main_id} vs Ramp {ramp_id} ===")
    # Mainline lanes 1–3
    cols_main = [f"Lane {i} Flow (Veh/5 Minutes)" for i in (1,2,3)]
    m = df.loc[df.Station_ID == main_id, ["5 Minutes"] + cols_main]
    # Ramp lane-1 flow
    r = ramp_df.loc[ramp_df.Station_ID == ramp_id, ["5 Minutes", "RampFlow"]]

    # Align on timestamp
    merged = pd.merge(m, r, on="5 Minutes", how="inner")
    if merged.empty:
        print("  → No overlapping timestamps, skipping.")
        continue

    # Require at least some nonzero ramp readings
    nz = merged["RampFlow"] != 0
    if nz.sum() < 10:
        print(f"  → RampFlow nonzero <10 readings ({nz.sum()}), skipping.")
        continue

    # Sum mainline lanes into TotalMain
    merged["TotalMain"] = merged[cols_main].sum(axis=1)

    # Pearson correlation
    r_val = merged["RampFlow"].corr(merged["TotalMain"])
    print(f"  Pearson r = {r_val:.3f}")

    # Cross-correlation at lags −3…+3
    xcorr = {
        lag: merged["RampFlow"].corr(merged["TotalMain"].shift(lag))
        for lag in range(-3, 4)
    }
    print("  Cross-corr lags −3…+3:", xcorr)

    # Scatter plot
    plt.figure(figsize=(5,4))
    plt.scatter(merged["RampFlow"], merged["TotalMain"], alpha=0.5)
    plt.title(f"Main {main_id} vs Ramp {ramp_id}  (r={r_val:.2f})")
    plt.xlabel("Ramp Flow (veh/5min)")
    plt.ylabel("Mainline Total Flow (veh/5min)")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
Overall min-flow hour: 3 → 24.188095238095237
Overall max-flow hour: 14 → 125.35952380952381

Per-station peak hours (1–23):
Station_ID  715898  716899  716900  716904  718081  718358  718360  762353  \
Peak_Hour       14      14      14      12      16      14      16      14

Station_ID  769625  775012
Peak_Hour       16      12

Per-station low hours (1–23):
Station_ID  715898  716899  716900  716904  718081  718358  718360  762353  \
Low_Hour         3       2       2       3       3       2       3       2

Station_ID  769625  775012
Low_Hour         3       3

Ramp station IDs found: [715901 762355 715903 716903 715905 769626 762357]

=== Main 769625 vs Ramp 769626 ===
  → RampFlow nonzero <10 readings (0), skipping.

=== Main 718081 vs Ramp 762357 ===
  Pearson r = 0.735
  Cross-corr lags −3…+3: {-3: 0.7194850749582926, -2: 0.7233300808676292, -1: 0.7237650357421967, 0:
0.7346393324410804, 1: 0.7287726762750604, 2: 0.7319039079142422, 3: 0.7266731292547952}
```



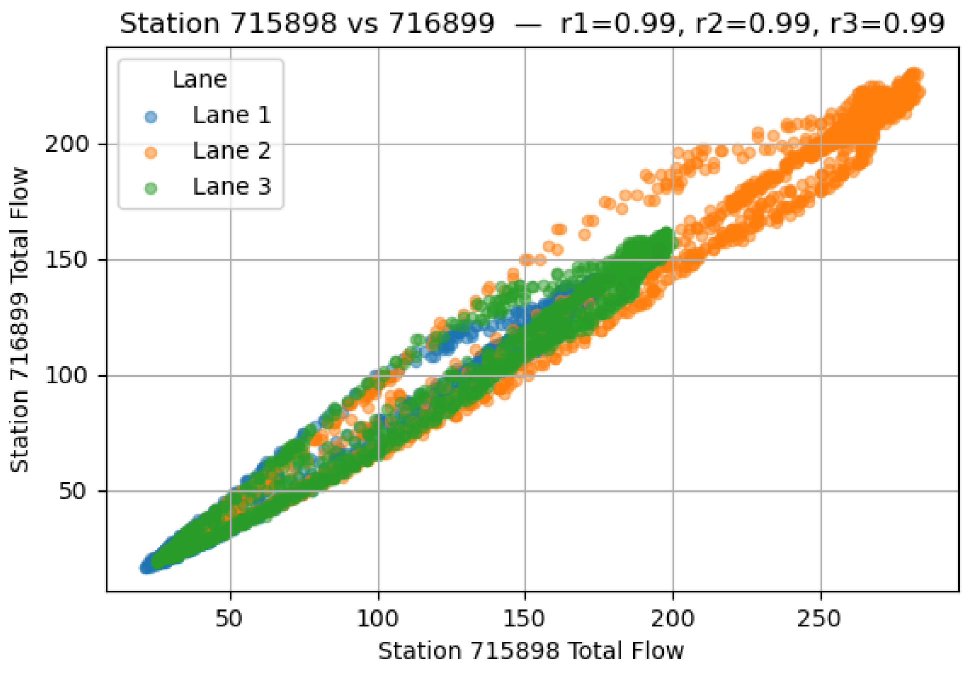
769626 (off ramp at the same location as 769625 Mainline sensor) is missing all of it's data, presumably it was failing during the time series that I pulled from PeMS so we will ignore it moving forward.

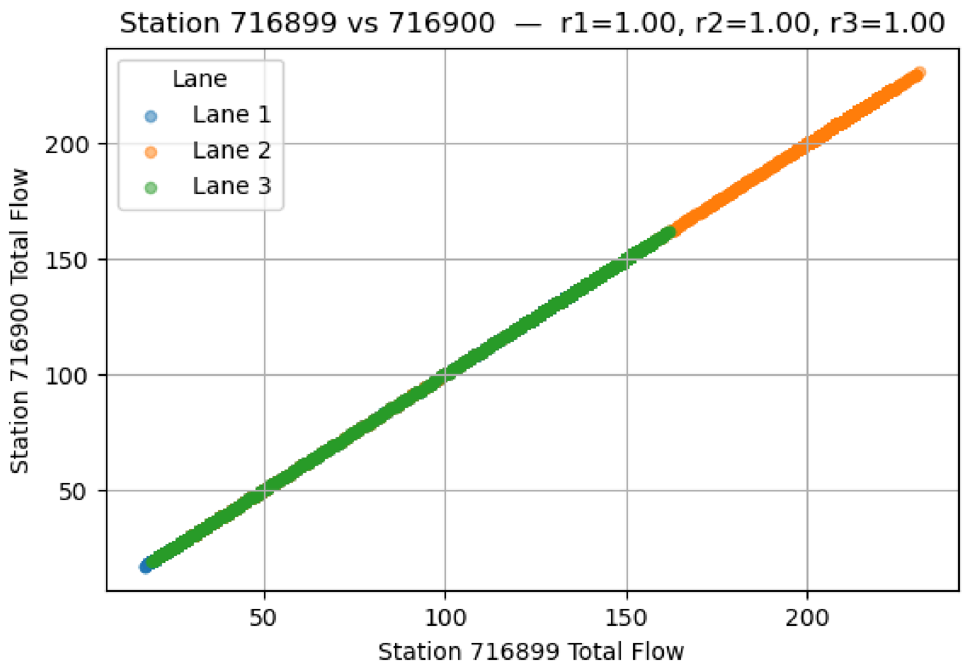
By contrast, we find that the on ramp sensor 762357 can explain about $(0.735^2) = .54$ 54% of the variance in traffic flow at the 718081 Mainline sensor which occurs in the same geographic location, just after the on-ramp itself. This effect remains consistently strong when lagged by 5-15 minutes in either direction.

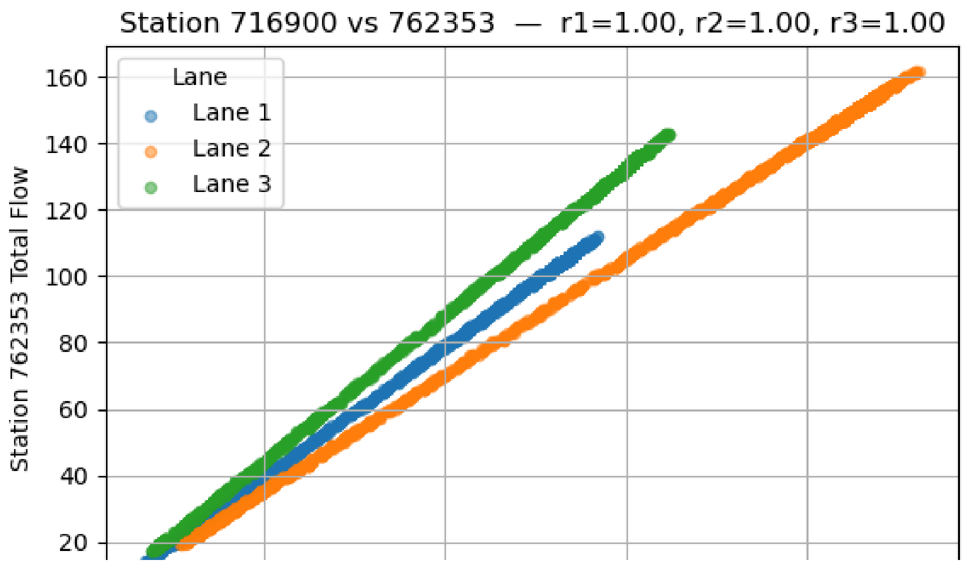
Averaging across all of the mainline sensors:

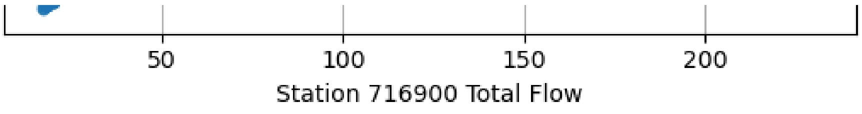
- Average hour of minimum flow: 3 AM
  - Average flow at 3 AM: 24.188 vehicles per hour
- Average hour of maximum flow: 2 PM
  - Average flow at 2 PM: 125.359

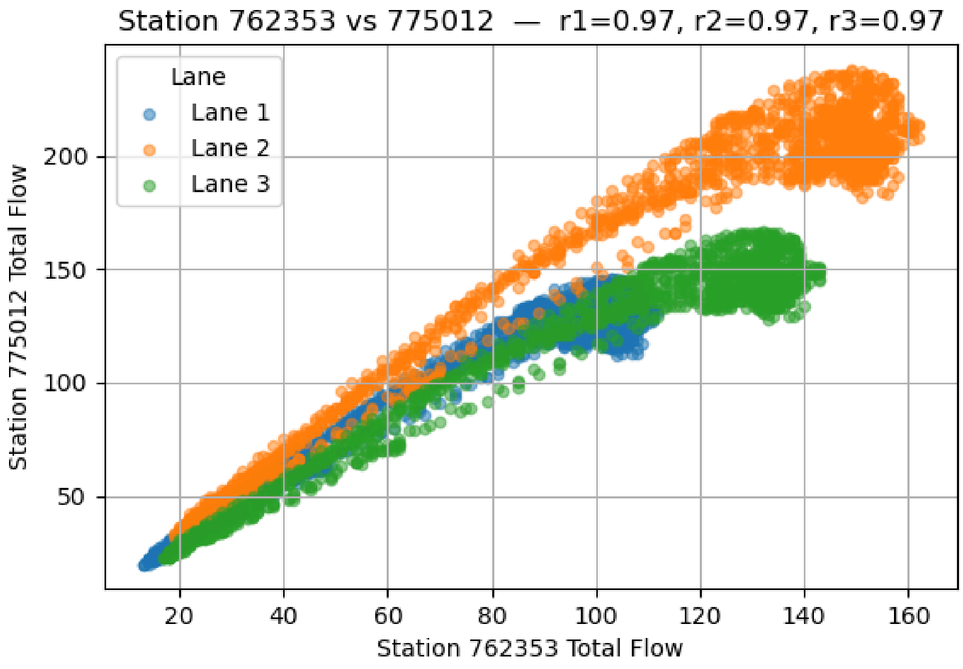
In [19]:
```python
'''
This is code I wrote prior to getting a little lost in the sauce with matplotlib.
  Ended up turning to chatgpt and having it rework my code into something that plots.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load data
df = pd.read_pickle("full_station_data.pkl")

# Drop non-mainline entries
df = df[df["Station_Type"] == "Mainline"]

coordinate_order = df["Latitude"].unique().sort()

# (0, 1), (1, 2) (2, 3) (3, 4), (4, 5), (5, 6), (6, 7),  (7, 8), (8, 9)
for i in [0, 1, 2, 3, 4, 5, 6, 7, 8]:
  x1 = df[df["Latitude"] == coordinate_order[i]]
  x2 = df[df["Latitude"] == coordinate_order[i + 1]]'''

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Load & filter
df = pd.read_pickle("full_station_data.pkl")
df["5 Minutes"] = pd.to_datetime(df["5 Minutes"])
main = df[df["Station_Type"] == "Mainline"]

# Order stations by latitude
station_coords = (
    main[["Station_ID","Latitude"]]
    .drop_duplicates()
    .sort_values("Latitude")
)
station_order = station_coords["Station_ID"].tolist()

# Prepare subplots for each adjacent pair
n_pairs = len(station_order) - 1
fig, axes = plt.subplots(n_pairs, 1, figsize=(6, 4 * n_pairs), sharex=False)

for ax, i in zip(axes, range(n_pairs)):
    s1 = station_order[i]
    s2 = station_order[i+1]

    # Extract lane flows for both stations
    cols = [f"Lane {j} Flow (Veh/5 Minutes)" for j in (1,2,3)]
    d1 = main.loc[main.Station_ID==s1, ["5 Minutes"] + cols]
    d2 = main.loc[main.Station_ID==s2, ["5 Minutes"] + cols]

    # Align on time
    merged = pd.merge(d1, d2, on="5 Minutes", suffixes=(f"_{s1}", f"_{s2}"))

    # Plot each lane
    for lane, color in zip((1,2,3), plt.rcParams['axes.prop_cycle'].by_key()['color']):
        c1 = f"Lane {lane} Flow (Veh/5 Minutes)_{s1}"
        c2 = f"Lane {lane} Flow (Veh/5 Minutes)_{s2}"
        ax.scatter(
            merged[c1],
            merged[c2],
            alpha=0.5,
            label=f"Lane {lane}",
            color=color,
            s=20
        )

    # Compute per-lane Pearson r’s
    r_vals = {
        lane: np.corrcoef(
            merged[f"Lane {lane} Flow (Veh/5 Minutes)_{s1}"],
            merged[f"Lane {lane} Flow (Veh/5 Minutes)_{s2}"]
        )[0,1]
        for lane in (1,2,3)
    }

    ax.set_title(
        f"Station {s1} vs {s2}  —  " +
        ", ".join(f"r{lane}={r:.2f}" for lane,r in r_vals.items())
    )
    ax.set_xlabel(f"Station {s1} Total Flow")
    ax.set_ylabel(f"Station {s2} Total Flow")
    ax.legend(title="Lane")
    ax.grid(True)

plt.tight_layout()
plt.show()
```

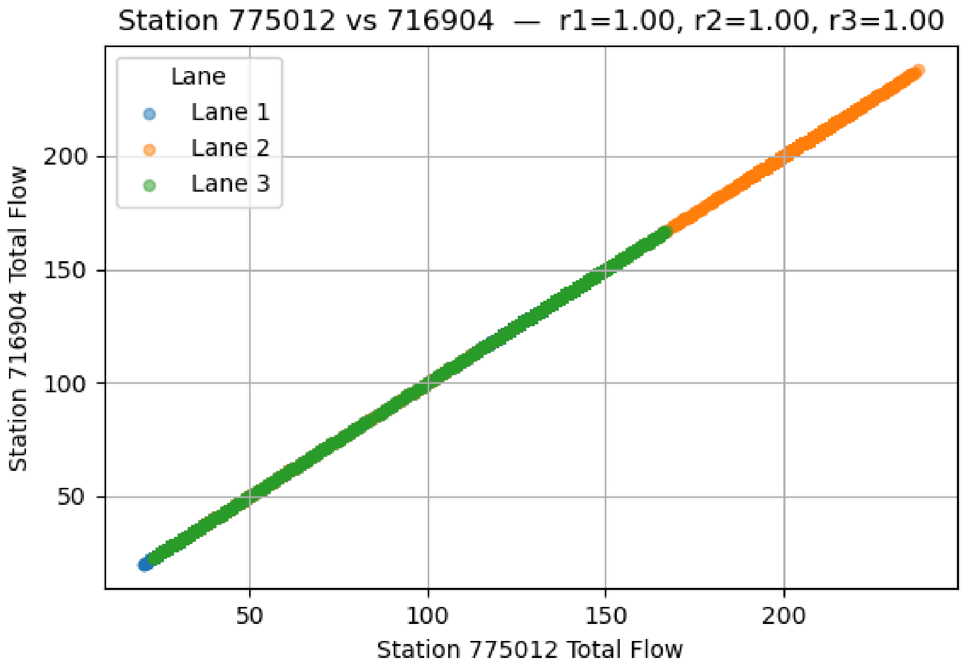

Station 715898 vs 716899 — r1=0.99, r2=0.99, r3=0.99
Lane
Lane 1
Lane 2
Lane 3
Station 716899 Total Flow
Station 715898 Total Flow
50
100
150
200
250

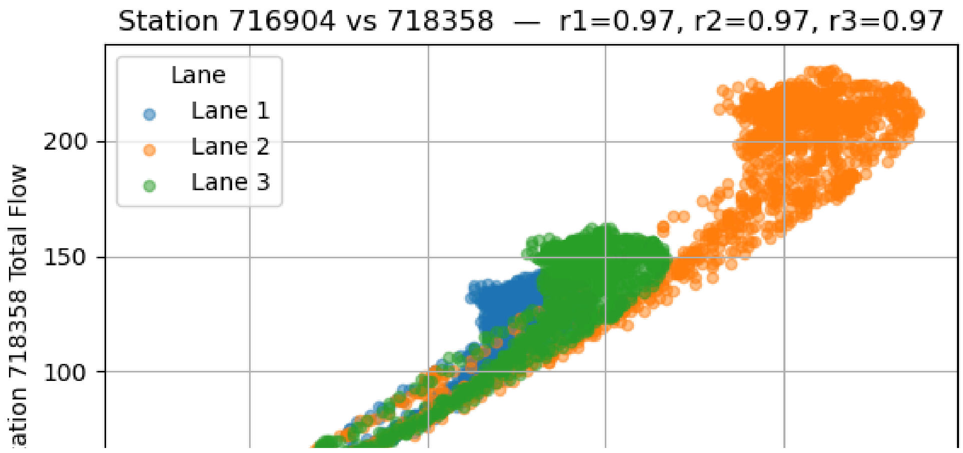

Station 716899 vs 716900 — r1=1.00, r2=1.00, r3=1.00
Lane
Lane 1
Lane 2
Lane 3
Station 716900 Total Flow
Station 716899 Total Flow
50
100
150
200

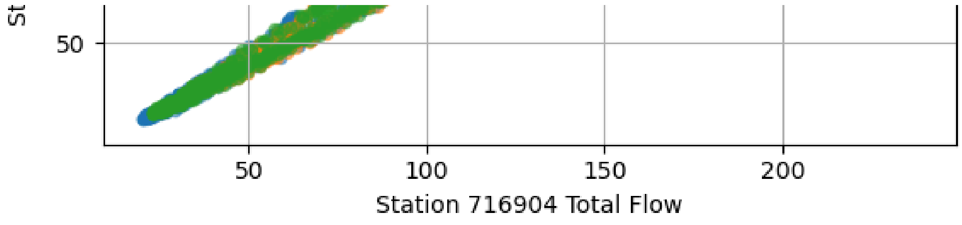

Station 716900 vs 762353 — r1=1.00, r2=1.00, r3=1.00
Lane
Lane 1
Lane 2
Lane 3
Station 762353 Total Flow
20
40
60
80
100
120
140
160

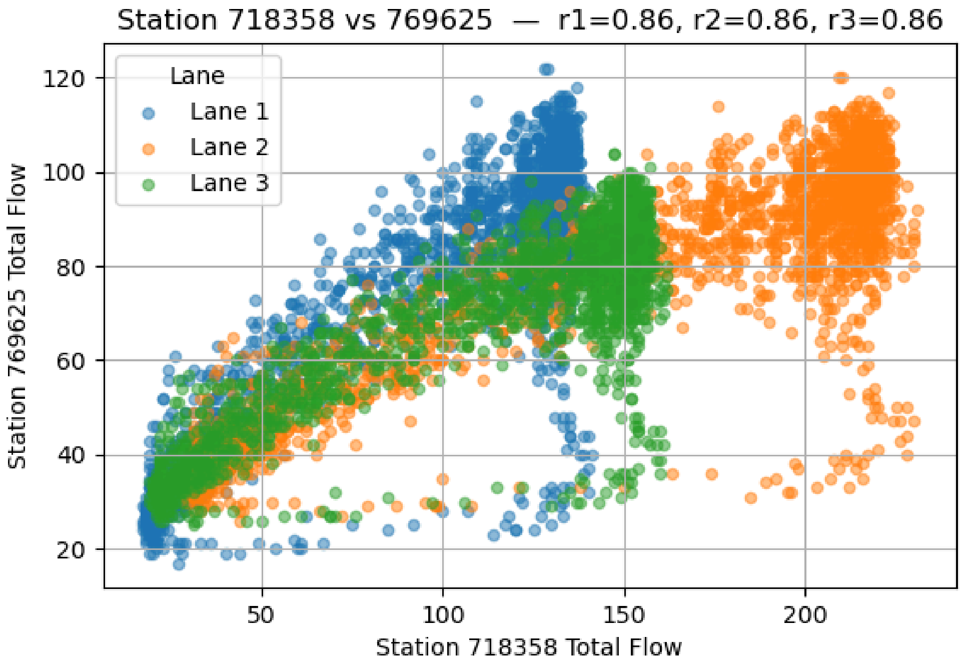

50
100
150
200
Station 716900 Total Flow

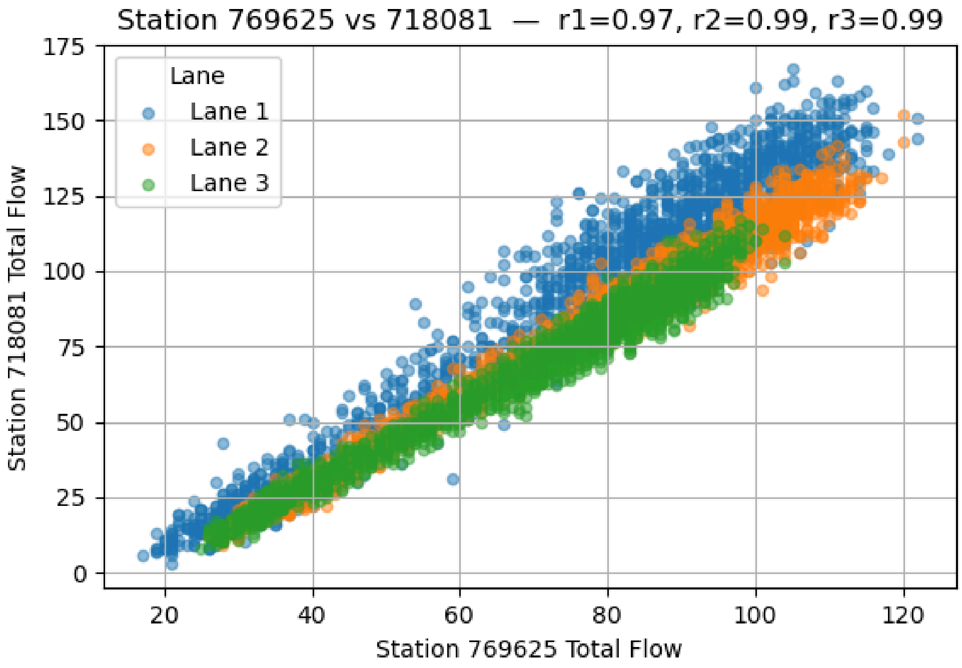

Station 762353 vs 775012 — r1=0.97, r2=0.97, r3=0.97
Lane
Lane 1
Lane 2
Lane 3
Station 775012 Total Flow
Station 762353 Total Flow

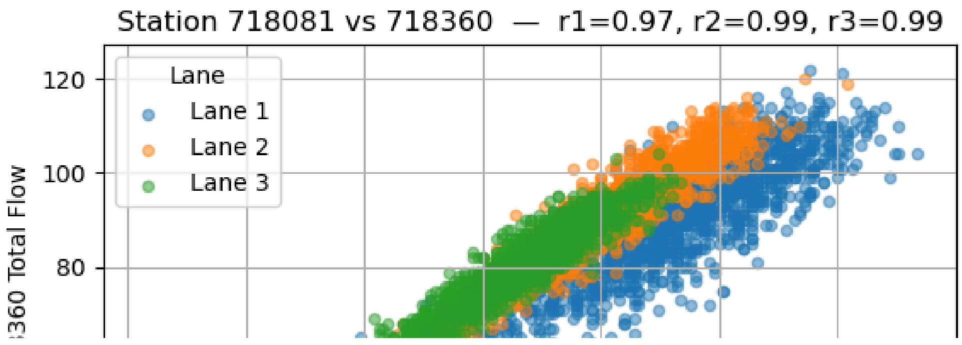

Station 775012 vs 716904 — r1=1.00, r2=1.00, r3=1.00
Lane
Lane 1
Lane 2
Lane 3
Station 716904 Total Flow
Station 775012 Total Flow

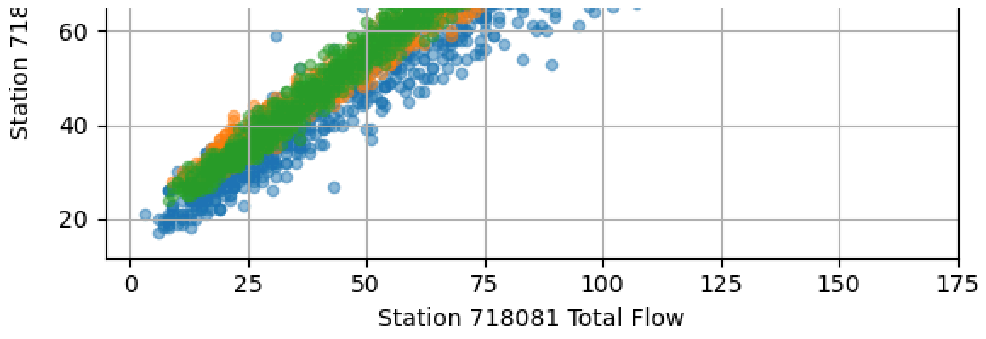

Station 716904 vs 718358 — r1=0.97, r2=0.97, r3=0.97
Lane
Lane 1
Lane 2
Lane 3
ation 718358 Total Flow


St
50
50
100
150
200
Station 716904 Total Flow


Station 718358 vs 769625 — r1=0.86, r2=0.86, r3=0.86
Lane
Lane 1
Lane 2
Lane 3
Station 769625 Total Flow
Station 718358 Total Flow
20
40
60
80
100
120
50
100
150
200


Station 769625 vs 718081 — r1=0.97, r2=0.99, r3=0.99
Lane
Lane 1
Lane 2
Lane 3
Station 718081 Total Flow
Station 769625 Total Flow
0
25
50
75
100
125
150
175
20
40
60
80
100
120


Station 718081 vs 718360 — r1=0.97, r2=0.99, r3=0.99
Lane
Lane 1
Lane 2
Lane 3
360 Total Flow
80
100
120



The results from comparing adjacent stations along the mainline corridor shows that, in general, the flow at an immediately upstream station is very strongly correlated with the flow at an immediately downstream station (r > 0.95). One notable exception: 718358 to 769625 have r = 0.86 for all three lanes, which is a stark drop in spatial continuity.

Station 769625 sits in the same geographic coordinate location as an off-ramp (station 769626), which did not record any data during our sample timeframe, but which likely represents a significant point of exit for traffic. I hypothesize that heavy exiting and de-merging at that location may weaken or disrupt the one-to-one relation between down/upstream neighboring stations, lowering the Pearson correlation.

Since station 769626 was dead or did not record data during the sampled time period, we are unable to utilize on/off ramp traffic flow as a predictive measure in our model. Instead, to remedy and explore the merging/de-merging hypothesis, we will introduce two new binary features into our model, exit_zone and merge_zone.

- exit_zone is 1 if an off ramp station occurs between two mainline stations, or at either of the related mainline stations, and 0 otherwise
- merge_zone is 1 if an on ramp station occurs between two mainline stations, or at either of the related mainline stations, and 0 otherwise.

We can construct one model with these additional features and another without them to compare and contrast how well our model is able to account for the noise in the presence or absence of a derived feature predicted to be relevant to the noise.