A4 Part 2 - Line Graph

April 2, 2025

1 Part 2: Simple Line Graphs, Scatter Plots, & Violin Plots

1.1 Rule et al.'s Rules (Discussion in Part 3)

1. **Rule 1**: Tell a story for an audience
2. **Rule 2**: Document the process, not just the results
3. **Rule 3**: Use cell divisions to make steps clear
4. **Rule 8**: Share and explain your data
5. **Rule 9**: Design your notebooks to be read, run, and explored

1.1.1 Import Libraries & Load Data

[1]:
```
import os
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import pytz
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.nonparametric.smoothers_lowess import lowess   # I will be␣
 ↪explaining this when we get to the scatterplot

file_path = os.path.realpath("/home/jovyan/work/resources/course_assignments/
 ↪assets/strava.csv")
df = pd.read_csv(file_path)

df['timestamp'] = pd.to_datetime(df['timestamp'])
df['timestamp'] = pd.to_datetime(df['timestamp'], utc=True)
df['timestamp'] = df['timestamp'].dt.tz_convert('America/Detroit')
```

1.1.2 Smooth Line Graph

As an owner of a Fitbit and Apple Watch, one of the first visualizations I thought of when I saw the data was tracking heart rate.

Specifically, it would be interesting to see how Professor Brook's heart rate changed throughout his workouts. In this sense, it would make sense if: - The x-axis represents the length of the

workout, probably in minutes - This means we will need to somehow calculate the "delta"/length of the workout - We will also need to calculate how each minute passes - The y-axis represents the professor's heart rate

1.1.3 The Problem of the NaN Values

We saw from "Preparing the Data" that there were 64 recorded workouts. This might crowd our line graph, but let's play around with it and see what we can do to make it as visually appealing but also informative as possible.

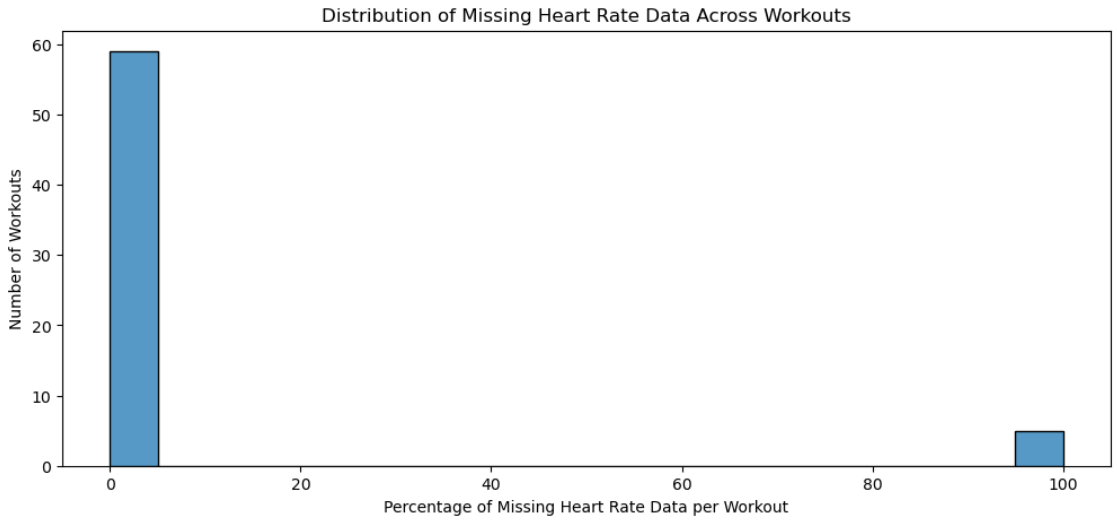
[2]:
```
print(df[df['heart_rate'].isna()].describe())
```

```
       Air Power  Cadence  Form Power  Ground Time  Leg Spring Stiffness  \
count        0.0      0.0         0.0          0.0                   0.0
mean         NaN      NaN         NaN          NaN                   NaN
std          NaN      NaN         NaN          NaN                   NaN
min          NaN      NaN         NaN          NaN                   NaN
25%          NaN      NaN         NaN          NaN                   NaN
50%          NaN      NaN         NaN          NaN                   NaN
75%          NaN      NaN         NaN          NaN                   NaN
max          NaN      NaN         NaN          NaN                   NaN

       Power  Vertical Oscillation  altitude      cadence      distance  \
count    0.0                   0.0       0.0  2283.000000   2294.000000
mean     NaN                   NaN       NaN    54.534385  11723.431072
std      NaN                   NaN       NaN    30.087879  11561.863993
min      NaN                   NaN       NaN     0.000000      0.000000
25%      NaN                   NaN       NaN    25.000000   2872.760000
50%      NaN                   NaN       NaN    71.000000   6313.670000
75%      NaN                   NaN       NaN    76.000000  18745.277500
max      NaN                   NaN       NaN   118.000000  39007.120000

       enhanced_altitude  enhanced_speed  fractional_cadence  heart_rate  \
count        2294.000000     2294.000000              2283.0         0.0
mean          259.221709        6.095188                 0.0         NaN
std            17.134105        2.680136                 0.0         NaN
min           219.600000        0.000000                 0.0         NaN
25%           246.000000        4.434250                 0.0         NaN
50%           257.200000        6.270000                 0.0         NaN
75%           272.200000        7.791000                 0.0         NaN
max           312.800000       15.349000                 0.0         NaN

       position_lat  position_long  speed  unknown_87  unknown_88  unknown_90
count  2.294000e+03   2.294000e+03    0.0      2283.0         0.0         0.0
mean   5.045897e+08  -9.994587e+08    NaN         0.0         NaN         NaN
std    3.284731e+05   6.002277e+05    NaN         0.0         NaN         NaN
min    5.039868e+08  -1.000741e+09    NaN         0.0         NaN         NaN
25%    5.043625e+08  -9.998499e+08    NaN         0.0         NaN         NaN
```

```
50%    5.045281e+08  -9.992678e+08    NaN         0.0         NaN         NaN
75%    5.048967e+08  -9.990651e+08    NaN         0.0         NaN         NaN
max    5.051599e+08  -9.984833e+08    NaN         0.0         NaN         NaN
```

[3]:
```python
# We know there's approximately 5% missing values, but let's dive in further
missing_hr = df['heart_rate'].isna().sum()
total_rows = len(df)
print(f"Missing `heart_rate` values: {missing_hr} ({missing_hr / total_rows *␣
 ↪100:.2f}%)")

# Are specific workouts missing heart rate data? Or is the missing heart rate␣
 ↪data randomly distributed across all workouts?
# Let's determine this by grouping our data by `datafile` (i.e., workout)
missing_per_workout = df.groupby('datafile')['heart_rate'].apply(lambda x: x.
 ↪isna().mean() * 100)
print(missing_per_workout.sort_values(ascending=False))  # Show % missing per␣
 ↪session

plt.figure(figsize=(12, 5))
sns.histplot(missing_per_workout, bins=20)
plt.xlabel("Percentage of Missing Heart Rate Data per Workout")
plt.ylabel("Number of Workouts")
plt.title("Distribution of Missing Heart Rate Data Across Workouts")
plt.show()
```

```
Missing `heart_rate` values: 2294 (5.64%)
datafile
activities/2875543445.fit.gz    100.0
activities/2917827233.fit.gz    100.0
activities/2903861013.fit.gz    100.0
activities/2903714629.fit.gz    100.0
activities/2875620974.fit.gz    100.0
                                …
activities/2737527405.fit.gz      0.0
activities/2740274713.fit.gz      0.0
activities/2742761748.fit.gz      0.0
activities/2746000812.fit.gz      0.0
activities/2925939753.fit.gz      0.0
Name: heart_rate, Length: 64, dtype: float64
```



This is fantastic! This bimodal graph tells us is that: - Almost 60 workouts have no missing heart rate data - Approximately 5 workouts have 100% missing heart rate data - It's possible that the heart rate sensor in the fitness tracker malfunctioned or "dropped transmission" during those 5 workouts, which led to the absence of data.

In this case, I would drop the workouts that have missing heart rate values. If the missing heart rate values were randomly distributed across the data, we might have had to impute our data with certain values (e.g., median, mean, interpolate) to complete it. However, because it appears that only specific workouts didn't have heart rate values, we can safely [and in good conscience] drop those workouts.

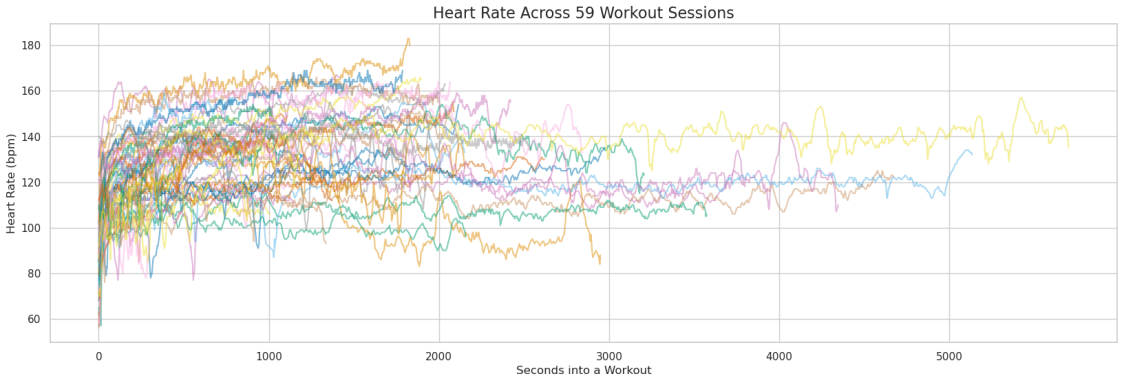
[4]:
```
# Preparing Our Data for the Line Graph
df = df.dropna(subset=['heart_rate'])

# Group our Data by Workout & Calculate Time per Workout (End - Start)
df['time_delta'] = df.groupby('datafile')['timestamp'].transform(lambda x: x.
 ↪max() - x.min())

# Convert All Time to Seconds (I noticed that we can't convert to minutes,␣
 ↪because the first few lines in strata.csv shows that his heart rate surges␣
 ↪in the first few seconds)
df['workout_duration_seconds'] = df['time_delta'].dt.total_seconds()

# We need to correspond the 'standard time' of the workout to the 'seconds into␣
 ↪the workout'
df['seconds_in_workout'] = (df['timestamp'] - df.
 ↪groupby('datafile')['timestamp'].transform('min')).dt.total_seconds()

df[['datafile', 'seconds_in_workout', 'heart_rate']].tail()
```

[4]:

	datafile	seconds_in_workout	heart_rate
40644	activities/2925939753.fit.gz	5691.0	143.0
40645	activities/2925939753.fit.gz	5693.0	142.0
40646	activities/2925939753.fit.gz	5694.0	142.0
40647	activities/2925939753.fit.gz	5699.0	138.0
40648	activities/2925939753.fit.gz	5702.0	135.0

Note that I had to change my `x-axis` units to seconds as opposed to minutes. The reason being that, with a quick glance of the CSV file, we can see that the professor's heart rate increased within the first few seconds of his first workout.

[5]:
```
plt.figure(figsize=(20, 6))
sns.set(style="whitegrid")

sns.lineplot(data=df,
             x='seconds_in_workout',
             y='heart_rate', hue='datafile',
             alpha=0.5,       # slightly transparent to improve readability due␣
 ↪to number of lines
             palette='colorblind',
             legend=None)

plt.title(f'Heart Rate Across {len(df["datafile"].unique())} Workout Sessions',␣
 ↪fontsize=16)
plt.xlabel('Seconds into a Workout', fontsize=12)
plt.ylabel('Heart Rate (bpm)', fontsize=12)

plt.show()
```



Let's see if we can use `Plotly` to create a more interactive chart. I'd like to be able to zoom in to the left side of the chart since the data is more densely packed in that area.

[19]:
```
fig = px.line(df,
              x='seconds_in_workout',
```

```
              y='heart_rate',
              color='datafile',
              labels={'seconds_in_workout': 'Seconds into a Workout',
                      'heart_rate': 'Heart Rate (bpm)'},
              title=f'Heart Rate Across {len(df["datafile"].unique())} Workout␣
 ↪Sessions')

fig.show()
```

Because of Plotly's native interactivity, we can easily zoom into the beginning of the line graph and see the details. The hover feature also gives us exact values.

However, I still find that the "seconds" is a bit hard to use because we're not exactly attuned to use seconds in everyday life. Minutes, on the other hand, feel much more intuitive; and I wonder if a chart made with minute would be easier to understand.

To do this, we would have to convert a lot of data into decimal form (e.g., 00:01:15 is now 1.25 minutes), which is not ideal, but it's something worth exploring and evaluating.

[7]:
```
# Calculate the workout duration (in minutes instead of seconds this time)
# To do this, we convert ALL time to seconds, then divide all time by 60 to get␣
 ↪the number of minutes
df['workout_duration_minutes'] = df['time_delta'].dt.total_seconds() / 60

# Create a new column for the number of minutes INTO a workout
# To do this, we calculate the number of seconds from the initial timestamp,␣
 ↪then divide by 60
df['minute_into_workout'] = (df['timestamp'] - df.
 ↪groupby('datafile')['timestamp'].transform('min')).dt.total_seconds() / 60

# Check the updated structure
df[['datafile', 'minute_into_workout', 'heart_rate']].head()
```

[7]:

	datafile	minute_into_workout	heart_rate
0	activities/2675855419.fit.gz	0.000000	68.0
1	activities/2675855419.fit.gz	0.016667	68.0
2	activities/2675855419.fit.gz	0.066667	71.0
3	activities/2675855419.fit.gz	0.183333	77.0
4	activities/2675855419.fit.gz	0.200000	80.0

[8]:
```
fig = px.line(df,
              x='minute_into_workout',
              y='heart_rate',
              color='datafile',
              labels={'minute_into_workout': 'Minutes into a Workout',
                      'heart_rate': 'Heart Rate (bpm)'},
              title=f'Heart Rate Across {len(df["datafile"].unique())} Workout␣
 ↪Sessions')
```

```
fig.show()
```

We can double-click on any of the data files in the legend to isolate a specific workout, which is a great feature that `Plotly` offers us.

The next thing I'd like to tackle is the legend. It's horrendous and not human-friendly. Instead of the datafile name, let's rename the workouts to something more meaningful and easily understood by humans, like the date of a workout.

[9]:
```
# We'll start by creating a new column, `workout_date`, from the existing␣
 ↪column, `timestamp`.
# Since each workout has multiple timestamps that span the entire workout,␣
 ↪we'll use the "minimum" value to represent the start date.
df['workout_date'] = df.groupby('datafile')['timestamp'].transform('min').dt.
 ↪date
df['workout_date'] = df['workout_date'].astype(str)

fig = px.line(df,
              x='minute_into_workout',
              y='heart_rate',
              color='workout_date',
              labels={'minute_into_workout': 'Minutes into a Workout',
                      'heart_rate': 'Heart Rate (bpm)'},
              title=f'Heart Rate Across {len(df["datafile"].unique())} Workout␣
 ↪Sessions')

fig.update_layout(legend_title_text='Workout Date')

fig.show()
```

1.2 This is BAD!

Something happened here and I clicked around to see what had happened. It appears that, amid a few other dates, July 31, 2019 showed 4 graphs!

My *incorrect* original premise was that Professor Brooks only worked out once a day. However, this was a misconception. If I were to print the number of unique datafiles and workout dates, I would see that there are fewer workout_dates than datafiles. Thus, my conclusion is that I should be more specific in naming my legend. Instead of naming each workout by only its date, I should also include its start time.

[10]:
```
print(len(df["datafile"].unique()))
print(len(df["workout_date"].unique()))
```

```
59
42
```

[11]:
```
# Same grouping process as before
df['workout_start_time'] = df.groupby('datafile')['timestamp'].transform('min')

# This time, let's include the hour and minute of the starting time of the␣
 ↪workout.
# This should improve our graph
df['workout_label'] = df['workout_start_time'].dt.strftime('%Y-%m-%d %H:%M')

# Continue as before
fig = px.line(df,
              x='minute_into_workout',
              y='heart_rate',
              color='workout_label',  # Now the legend includes date and start␣
 ↪time
              labels={'minute_into_workout': 'Minutes into a Workout',
                      'heart_rate': 'Heart Rate (bpm)',
                      'workout_label': 'Workout Start Time'},
              title=f'Heart Rate Across {len(df["workout_label"].unique())}␣
 ↪Workout Sessions')  # Confirmed that we have 59 uniquely named workouts for␣
 ↪our legend!

fig.show()
```

1.3 But What Now?

This is good information, but we can't really draw any useful conclusions or actionable findings from this *yet*. However, we might be able to use this *in conjuction* with other charts to inform our findings. Let's see if we can supplement this data-dense line chart with a few other graphs.

Because Professor Brooks was working out July through October, we can see if the professor's health improved **over time**.

I did a some research and found that: - "Individuals who are more physically fit tend to have lower resting heart rates and a more controlled increase in heart rate during exercise." [https://www.heartscope.com.au/why-does-the-heart-rate-increase-during-exercise/#:~:text=or%20light%20jogging.-,Fitness%20Level,cardiovascular%20systems%20are%20more%20efficient.] - " For a 40-year-old man, a target heart rate zone for cardio exercise is generally between 90 and 153 beats per minute (bpm), with an estimated maximum heart rate (MHR) of 180 bpm." [https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates] - Please note that I have decided to utilize the assumption from Deepti Wilson's comment shared on Slack on March 27th at 4:17 PM. The comment stated that it was acceptable to assume Professor Brooks was an **average 40-year-old male**. I confirmed this assumption again Erik Lang during his office hours on Saturday, March 29th. - There are 3 exercise zones we can consider: [https://www.webmd.com/fitness-exercise/what-to-know-heart-rate-fat-burning-cardio] - Lower-Intensity Zone: 50-60% MHR (approx. 90-106 bpm) - Fat-Burning Zone: 60-70% MHR (approx. 106-126 bpm) - Aerobic/Cardio Zone: 70% + MHR (approx. 126+) - "A lower resting heart rate generally indicates better cardiovascular health and fitness, while a higher

resting heart rate can be an indicator of increased risk for cardiovascular disease and mortality." [https://www.health.harvard.edu/blog/your-resting-heart-rate-can-reflect-your-current-and-future-health-201606172482#:~:text=%22In%20certain%20cases%2C%20a%20lower%20resting%20heart,cardiac%

With this in mind, I'd like to see if I can "clean up" the line graph so it's not as **visually overwhelming**. The purpose of the original line graph was to see whether there was a general shape that we can identify in Professor Brook's workouts. Ideally, I'd like to see how the professor warms up, gets going, and cools down for his workouts.

One way that I can do this is to group the workouts by `minute_into_workout`, and then calculate the average `heart_rate` for the given minute.

[12]:
```
# Group workouts by minute, then average the HR for each minute
avg_hr_by_minute = df.groupby('minute_into_workout')['heart_rate'].mean().
 ↪reset_index()
fig = px.line(avg_hr_by_minute, x='minute_into_workout', y='heart_rate',
              labels={'minute_into_workout': 'Minutes into Workout',␣
 ↪'heart_rate': 'Average HR (bpm)'},
              title='Average Heart Rate Profile Across All Workouts')
fig.show()
```

WOW! I really love this graph! A lot more than I expected :) :)

So let's discuss this graph. Due to the vast amount of data (i.e., decimals of minutes), the graph appears to go straight up and down - this is incorrect. The graph's "static" look reflects the **shape and fluctuation** of Professor Brooks' average HR values over time. - Skinny (Smoother, Less "Jumpy"): A flatter or more gradual curve suggests his average HR is **more consistent** and **fluctuates less** across workouts at those minutes. There are fewer wild "swings" in the underlying data. - Wide (Jagged, "Static"): A taller, more erratic line (sort of "static" on an old TV) indicates bigger ups and downs in the averages, which indicates **greater variability/fluctuations** in the professor's HR across workouts at those points.

If I were to "section" the graph into meaningful chunks, I would do so in the following way: - **0–15 Minutes: Consistent Warm-Ups (Low Variance, Consistent Warmup)** - What It Means: The average HR starts low (approximately 85-140 bpm) and rises steadily as workouts begin. The "skinniness" suggests most workouts follow a similar warmup pattern - HR increases predictably across the 59 sessions. - Professor Brooks: Your warmups are rock-solid consistent! Your HR typically climbs smoothly during the first 15 minutes of your workouts. - **15–35 Minutes: Peak Effort Zone (Moderate Variance, Aerobic/Cardio Zone with Max HR)** - The line "widens" as HR rises to the maximum and/or plateaus (around the 140–150 bpm range), but with greater fluctuation than the preceeding 15 minutes. The high-effort phase—HR hits its max as workouts reach full intensity, with some sessions pushing to 150 bpm, while others hovering lower. This could mean workouts start diverging; some workouts ramp up faster, others plateau, depending on intensity or type. - Professor Brooks: Here's your peak zone, 15–35 minutes, where your HR tops out around 140–150 bpm (putting you in the aerobic/cardio zone), and the line "widens" a bit. You're cranking it up, but with some diversity in how hard each session hits. Your workouts appear to start to split; some push harder, others ease off, showing variety in how you build effort. - **35–45 Minutes: Slow Decrease (High Variance, Slow Decrease)** - A very slight decrease in HR begins as workouts start easing off, but with more variability as some drop faster than others. - Professor Brooks: From 35–45 minutes, your HR slides from approximately 135 to 120

bpm, and a slow wind-down from aerobic/cardio to the fat-burning zone kicks in. This transition period showing greater variance, showing some workouts cool off quicker than others while you continue to push. - **45–65 Minutes: Plateau (High Variance)** - A steady average hides huge swings: some workouts dip to 103 bpm, others spike to 150 bpm, creating that "tall static" look. - Professor Brooks: As the data plateaus around 120 bpm, but the range is wild — 103 to 150 bpm! The tall static indicates the great variance in workout intensity at the 45-65-minute mark. This is a great low-intensity/fat-burning zone to plateau in. - **65–85 Minutes: Late-Game Surge (High Variance, Increase)** - A surprising late surge! Workouts that make it to 65-85 minutes appear to ramp up again, with big variability (that "static" persists). The 75–85 stretch narrows slightly as HR trends upward more consistently. - Professor Brooks: From 65–85 minutes, it's a comeback! Your HR climbs from around 110 to 140 bpm. Could this be where you find your second wind? The variance continues as some long workouts surge later than others, though it narrows a bit toward the end. The "narrowing" may be due to fewer workouts making it to this point. - **85+ Minutes: Very Narrow (Extremely Low Variance, Very Rare Long Workouts)** - Only a few workouts reach this far, and their HRs cluster tightly, with very little room for variation due to so few data points. - Professor Brooks: Past 85 minutes, we see very little variance in your workouts as only your longest sessions reach this point.

1.3.1 Future Notes

One thing I do notice that might benefit from further data analysis is how the professor cools down. The line graph above shows us some great data for his warm-ups; but it may also be just as important to understand how the professor cools down from his workouts, as cool downs gradually help us return our body to a resting state. This also prevents injury, and promotes recovery by gradually lowering heart rate, blood pressure, and body temperature, while also aiding in muscle recovery and relaxation [https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/exercise/art-20045517].

1.4 Did Professor Brook's Heart Health or Exercise Routine Improve?

1.4.1 A Scatter Plot with a Linear Trend Line

Now that we have a better idea of what the professor's workouts typically look like, we can investigate further to get a better idea of the *improvement* the professor may have experienced during these months. Thus, my first follow-up chart is to address whether Professor Brooks' heart health improved over time.

To do this, we can see how the professor's **average heart rate** during workouts changes over time (across all 59 workouts with heart rate readings). This could indicate improvements in fitness (lower average HR over time) or changes in workout intensity).

Although it makes sense to generate another line graph for a time-based graph, I don't believe this is the best way to approach our new graph. The reason being that we deduced earlier that Professor Brooks sometimes completes more than 1 workout in a day. This means that a standard line graph connecting points chronologically might create misleading jumps or trends (especially noticeable on days when the professor does a variety of low-intensity and strenuous workouts). Thus, the connection of the points could obscure the true progression over time and confuse the readers (especially with perfectly vertical lines in our graph).

I sought to address this issue by, instead, generating a scatterplot with a trend line. A **linear Ordinary Least Squares (OLS) regression** is one of the most basic trend lines we can fit to

our data. However, a quick search on `Plotly` produced another type of trend line called a **Locally WEighted Scatterplot Smoothing (LOWESS) trendline**. The LOWESS trend line fits a **smooth curve** to scatterplots, and allows us to capture non-linear trends. I thought that this would be a useful addition to our data, so I followed Plotly's page to add the LOWESS trend line [https://plotly.com/python/linear-fits/].

[13]:
```
# Calculate average HR per workout
avg_hr_per_workout = df.groupby('datafile').agg({
    'heart_rate': 'mean',
    'timestamp': 'min'  # Like before, the minimum time [of a datafile] corresponds to the start time of a workout
}).reset_index()

# Sort by Timestamp for Chronological Order
avg_hr_per_workout = avg_hr_per_workout.sort_values('timestamp')

# Generate Scatterplot
fig = px.scatter(avg_hr_per_workout,
                 x='timestamp',
                 y='heart_rate',
                 trendline='ols',  # Let's add a trendline. (ols = least squares regression line)
                                   # Reference: https://plotly.com/python/linear-fits/
                 trendline_color_override='orange',   # Used the following link to determine color: https://i.sstatic.net/xRwWi.png
                 labels={'timestamp': 'Workout Date', 'heart_rate': 'Average Heart Rate (bpm)'},
                 title=f'Average Heart Rate per Workout Across {len(avg_hr_per_workout["datafile"].unique())} Workouts with OLS Trendline')

fig.show()
```

1.4.2 A Scatter Plot with a Non-Linear "LOWESS" Trend Line

Let's repeat the process, but replace `ols` with `lowess` to see a non-linear trend line for the professor's data.

[14]:
```
# Calculate average HR per workout
avg_hr_per_workout = df.groupby('datafile').agg({
    'heart_rate': 'mean',
    'timestamp': 'min'  # Like before, the minimum time [of a datafile] corresponds to the start time of a workout
}).reset_index()

# Sort by Timestamp for Chronological Order
avg_hr_per_workout = avg_hr_per_workout.sort_values('timestamp')
```

```
# Generate Scatterplot
fig = px.scatter(avg_hr_per_workout,
                 x='timestamp',
                 y='heart_rate',
                 trendline='lowess',  # Let's try the LOWESS trendline here
                                      # Reference: https://plotly.com/python/
 ↪linear-fits/
                 trendline_color_override='limegreen',   # Used the following␣
 ↪link to determine color: https://i.sstatic.net/xRwWi.png
                 labels={'timestamp': 'Workout Date', 'heart_rate': 'Average␣
 ↪Heart Rate (bpm)'},
                 title=f'Average Heart Rate per Workout Across␣
 ↪{len(avg_hr_per_workout["datafile"].unique())} Workouts with LOWESS␣
 ↪Trendline')

# I added a labeled shaded region for the discussion below
fig.add_vrect(
    x0="2019-07-26", x1="2019-08-20",
    fillcolor="gray", opacity=0.2,
    layer="below", line_width=0,
    annotation_text="Mid-Summer Spike", annotation_position="top left"
)

fig.show()
```

1.4.3 Takeaways from Scatterplots with Accompanying Trend Lines

What a cool new trend line to experiment with! It's clear that the LOWESS line adapts to the data's natural shape rather than forcing a straight line. The LOWESS line appears to make a "backwards-Z" sort of trend line, with a noticeable increase mid-summer (approximately July 26 through August 20, seen in the above shaded region). However, the OLS regression line makes it immediately clear that there was an **overall increase** in the professor's average heart rate from July through October.

Professor Brooks, the rise in your average heart rate of your workouts shows that your workouts likely got **tougher** over time! The overall rise in average heart rate suggests you've been pushing harder overall [from July-October], possibly increasing intensity or duration as the months progressed. This increase could mean your building strength or endurance; but it may also hint at other confounding variables (e.g., less recovery or external factors (e.g., increase in summer heat (July-August is the hottest time of the year in Michigan) or increases in stress/cortisol – but these would have had to continue throughout July-October).

1.5 Heart Rate Zone Distributions

To summarize what we have analyzed so far, we've compiled average heart rate "profiles" of Professor Brook's workouts with respect to: 1) Minutes into a Workout [*via line charts*], and 2) Workout Date (July-Oct) [*via scatter plots with trend lines*]

Perhaps the next logical step would be to better understand *how* Professor Brooks is exercising; specifically, how are the professor's workouts typically distributed in terms of lower-intensity, fat-burning, and aerobic/cardio zones?

Given that Professor Brook's maximum heart rate (MHR) is approximately 180 bpm (for an average 40-year-old man; calculated by `MHR = 220 - age`), we can categorize his workouts into three exercise zones we took into consideration: | Zone | Percentage of MHR | Approximate HR | | ——— | ——— | ——— | | Lower-Intensity Zone | 50-60% MHR | 90-106 bpm | | Fat-Burning Zone | 60-70% MHR | 106-126 bpm | | Aerobic/Cardio Zone | 70% + MHR | 126+ |

[15]:
```
# Let's first write a function that defines & categorizes the professor's heart␣
 ↪rate for us
def hr_zone(hr):
    if hr < 90: return 'Resting (<90 bpm)'  # Catch any sub-90 bpm data
    elif hr <= 106: return 'Lower-Intensity (90-106 bpm)'
    elif hr <= 126: return 'Fat-Burning (106-126 bpm)'
    else: return 'Aerobic/Cardio (126+ bpm)'

# Now we apply the HR Zone function to all HR data
df['hr_zone'] = df['heart_rate'].apply(hr_zone)

# Sort by datafile and timestamp to ensure chronological order
df = df.sort_values(['datafile', 'timestamp'])

# Calculate time differences in seconds within each workout
df['time_diff'] = df.groupby('datafile')['timestamp'].diff().dt.total_seconds().
 ↪fillna(0)

# Shift time_diff up and assign it to the previous HR (since the difference␣
 ↪applies to the interval ending at the current timestamp)
df['duration_seconds'] = df['time_diff'].shift(-1).fillna(0)

# Group by HR zone and add durations
zone_durations = df.groupby('hr_zone')['duration_seconds'].sum().reset_index()
zone_durations.columns = ['Zone', 'Seconds']

# Plot bar chart
fig = px.bar(zone_durations,
             x='Zone',
             y='Seconds',
             labels={'Seconds': 'Total Time (seconds)', 'Zone': 'Heart Rate␣
 ↪Zone'},
             title=f'Time Spent in Heart Rate Zones Across {len(df["datafile"].
 ↪unique())} Workouts',
             color='Zone',
             color_discrete_map={                                # Plotly's␣
 ↪Discrete Colors: https://plotly.com/python/discrete-color/
```

```
                'Resting (<90 bpm)': '#d62728',             # D3 Coloblind:␣
 ↪Red (increase visibility due to short height)
                'Lower-Intensity (90-106 bpm)': '#1f77b4',  # D3 Coloblind:␣
 ↪Blue
                'Fat-Burning (106-126 bpm)': '#ff7f0e',     # D3 Coloblind:␣
 ↪Orange
                'Aerobic/Cardio (126+ bpm)': '#2ca02c'      # D3 Coloblind:␣
 ↪Green
            })

fig.update_layout(
    xaxis_title="Heart Rate Zone",
    yaxis_title="Total Time (seconds)",
    showlegend=False,
)

fig.show()
```

Hmm.

This chart, while exactly what I had in mind, is surprisingly plain and wasn't as informative as I originally thought it would. But let's see what information we can derive from this histogram before creating a more informative chart.

We can tell from this chart that, throughout July-October, Professor Brooks spent the most time in the Aerobic/Cardio Zone (approx. 58.728k seconds = 16.313 hours). Next comes the Fat-Burning Zone, in which the professor spent approximately 41.256k (11.46 hrs). - Aerobic/cardio exercise is great for improving cardiovascular health; it strengthens ones heart and lungs, improving their efficiency in pumping blood and delivering oxygen [https://www.healthline.com/health/cardio-everyday#about-cardio]. The aerobic/cardio zone also burns more calories overall in a shorter amount of time compared to the fat-burning zone, even though a smaller percentage of calories comes from fat [https://www.webmd.com/fitness-exercise/what-to-know-heart-rate-fat-burning-cardio]. - On the other hand, the fat-burning zone is relatively straight-forward; this zone is effective for burning fat as fuel during the workout, especially if one can sustain the activity for longer durations.

In summary, from the histogram, Professor Brooks spent the most time in the Aerobic/Cardio Zone (16.31 hours), far exceeding the Fat-Burning Zone (11.46 hours) from July to October. This suggests a cardio-focused routine, which is great for heart health and calorie burn—aligned with benefits like improved oxygen delivery and stamina. However, the Fat-Burning Zone's lower total (11.46 hours) indicates less of an emphasis on sustained fat utilization, which could be key for weight loss goals [if that aligns with the professor's goals].

1.5.1 Suggestions from the Histogram

If the professor is interested in maximizing cardio benefits, he could try adding interval training to increase endurance. Or, if he has weight loss goals, he could add a couple workout sessions weekly that focus on keeping his heart rate in the Fat-Burning Zone to balance his workouts out.

1.6 A More Informative Graph?

1.6.1 Exploring Violin Plots

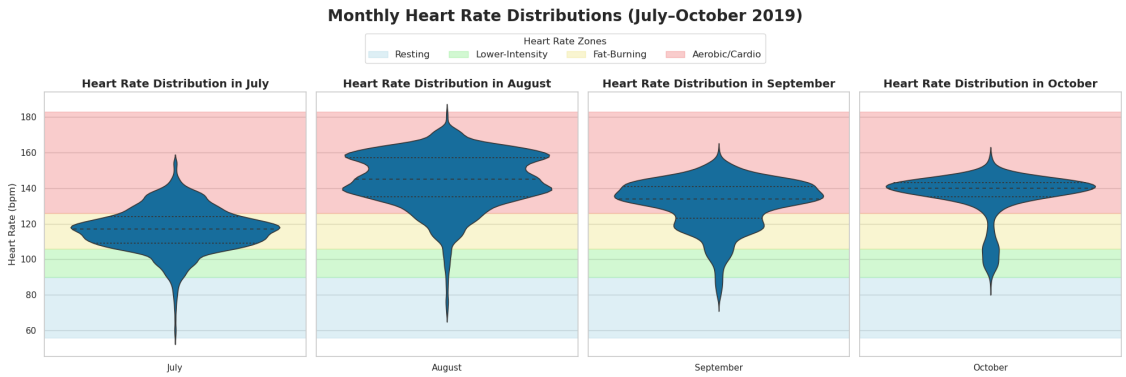
Although the histogram helped us gain a better understanding of the distribution of his heart rate zones across his workouts, it still feels like something is missing.

Perhaps seeing his heart rate distributions on a month-by-month basis would help us see how the professor has been focusing his workouts over the summer.

In this sense, let us try creating a violin plot. It would be great if we could do a side-by-side comparison of the 4 months (i.e., July, August, September, and October), all using the same scale for comparison (i.e., faceted plots). To make things more informative, we can also try shading in the heart rate zones so that we can easily tell in which zones each distribution lie.

[16]:
```
# Extract Month names for grouping
df['month'] = df['timestamp'].dt.strftime('%B')             # Extract full␣
 ↪month names

# Defining the Order of Months (ensuring chronological order)
month_order = ['July', 'August', 'September', 'October']

# Set up a 1-row × 4-column Subplot
fig, axes = plt.subplots(1, 4, figsize=(20, 6), sharey=True)

# Define HR Zones & Colors
zones = [
    (None, 90, "Resting", "lightblue"),
    (90, 106, "Lower-Intensity", "lightgreen"),
    (106, 126, "Fat-Burning", "khaki"),
    (126, None, "Aerobic/Cardio", "lightcoral"),
]

# Loop through each month, creating a violin plot with shaded HR zones
for i, month in enumerate(month_order):
    ax = axes[i]
    month_data = df[df['month'] == month]                   # Filter data for␣
 ↪the current month

    # Shade-in heart rate zones
    for lower, upper, label, color in zones:
        ax.axhspan(lower if lower else df['heart_rate'].min(),
                   upper if upper else df['heart_rate'].max(),
                   color=color, alpha=0.4, label=label if i == 0 else "")

    # Create Violin Plots
    sns.violinplot(data=month_data, x='month', y='heart_rate', ax=ax,
                   inner='quartile', palette='colorblind', linewidth=1.2)
```

```
    # Plot Titles
    ax.set_title(f"Heart Rate Distribution in {month}", fontsize=14,␣
 ↪fontweight='bold')
    ax.set_xlabel("")                                          # Remove x-axis label
    ax.set_ylabel("Heart Rate (bpm)" if i == 0 else "")     # Only 1 label for␣
 ↪y-axis on first plot needed

# Adding a Legend (only needed for the first subplot)
handles, labels = axes[0].get_legend_handles_labels()

# Tweaking Legend Features
fig.legend(handles, labels, loc="upper center",
           bbox_to_anchor=(0.5, 1.04),                         # Position legend␣
 ↪above the plots
           title="Heart Rate Zones", fontsize=12, ncol=4)  # Use 4 columns for␣
 ↪horizontal layout

# Add Chart Title
plt.suptitle("Monthly Heart Rate Distributions (July-October 2019)",␣
 ↪fontsize=20, fontweight='bold', y=1.1)
plt.tight_layout()
plt.show()
```



This is a great start! I did have to play around with moving the legend and title around to make sure nothing was covered, but this is definitely on the right track. However, since all our previous plots were generated using `Plotly`, it was strange having a static chart that had no interactivity (e.g., hovertext, zoom). In this sense, it would be nice to have a `Plotly` version of the violin plots for more native interactivity capabilities.

[17]:
```
# Extract Full Month Names
df['month'] = df['timestamp'].dt.strftime('%B')

# Define Months in Order
```

```
month_order = ['July', 'August', 'September', 'October']

# Create Figure
fig = go.Figure()

# Loop each month to generate a violin plot for each
for i, month in enumerate(month_order):
    month_data = df[df['month'] == month]
    fig.add_trace(go.Violin(
        x=[i] * len(month_data),
        y=month_data['heart_rate'],
        name=month,       # Still name it for clarity, but won't show in legend
        line_color='steelblue',
        box_visible=True,
        meanline_visible=True,
        showlegend=False  # Hide from legend
    ))

# Define HR Zones & Colors
zones = [
    (0, 90, "Resting", "cornflowerblue"),
    (90, 106, "Lower-Intensity", "green"),
    (106, 126, "Fat-Burning", "gold"),
    (126, 180, "Aerobic/Cardio", "red"),
]

# Add Shaded HR Zones
for lower, upper, label, color in zones:
    fig.add_hrect(
        y0=lower, y1=upper,
        fillcolor=color, opacity=0.2, layer="below", line_width=0
    )

# Add Dummy Traces for Legend (so that only HR Zones show)
for lower, upper, label, color in zones:
    fig.add_trace(go.Scatter(
        x=[None], y=[None], mode='markers',
        marker=dict(color=color, opacity=0.4, size=15),
        name=label,
        showlegend=True  # Show only zones in legend
    ))

# Update layout
fig.update_layout(
    title=dict(
        text="Monthly Heart Rate Distributions (July-October 2019)",
        font=dict(size=20),
```

```
        y=0.95,
        x=0.5,
        xanchor="center"
    ),
    xaxis=dict(
        tickmode='array',
        tickvals=[0, 1, 2, 3],
        ticktext=month_order,
        title="Month"
    ),
    yaxis_title="Heart Rate (bpm)",
    yaxis_range=[0, 200],
    plot_bgcolor='white',
    paper_bgcolor='white',
    legend=dict(
        title="Heart Rate Zones",
        orientation="h",
        yanchor="top",
        y=1.1,
        xanchor="center",
        x=0.5,
        font=dict(size=12)
    )
)

fig.show()
```

I had to be clever about a few things (especially the legend!), but I am much happier with how this turned out on `Plotly` due to the library's native interactivity. Now onto the discussion!

1. **July: Fat-Burning Dominance** (Max: 155, Q3: 124, Median: 117, Mean: 116, Q1: 109, Min: 56)
    - July's HR distribution is compact and centered in the Fat-Burning Zone, suggesting a consistent intensity level across recorded running or cycling sessions. The narrow spread (15 bpm between Q1 and Q3) indicates low variability. The max HR (155 bpm) dipping into the Aerobic/Cardio Zone suggests occasional bursts, possibly brief accelerations or inclines, but these are outliers in what appears to be a consistent demonstration of moderate-intensity exercise.
2. **August: Aerobic/Cardio Dominance** (Max: 183, Q3: 157, Median: 145, Mean: 144, Q1: 135, Min: 69)
    - August marked a clear shift to higher-intensity efforts in intense cardio. The wider Q1-to-Q3 spread and overall higher HRs suggest more variability, possibly mixing tough workouts with lower-intensity periods (min: 69 bpm). August is also one of the hottest months of the year. If we were to consider that a significant portion of these exercises took place outside (which we will begin to explore in [Part 3] of this project), then the additional heat-related stress could also be increasing the professor's overall HR. Outside of the heat, the aerobic/cardio dominance could also indicate a push in fitness level, a change in routine, or perhaps the presence of workouts that don't incorporate as much

cooldown times.

3. **September: Aerobic/Cardio with Some Fat-Burning Mix** (mMx: 160, Q3: 141, Median: 134, Mean: 131, Q1:123, Min: 76)
    - September continues with the aerobic/cardio focus but softens a bit; it is overall less extreme than August (lower max HR), with some moderate activity creeping in. This might suggest a balance between pushing cardio limits and sustained efforts at a lower HR, or a natural taper after August's intensity [, perhaps due to the cooling weather].
4. **October: Aerobic/Cardio-Heavy (Limited Data)** (Max: 157, Q3: 143, Median: 140, Mean: 136, q1: 135, Min: 86)
    - October's single day (Oct. 3) leans hard into an intense, aerobic/cardio-centered workout. The tight distribution around 140 bpm suggests a focused, intense session. It's important to note that the sample size limits broader conclusion about the month of October.

1.7 Last Remarks from Simple Graphs

Professor Brooks, the 59 workouts we worked with weave a cohesive and exciting story across our three simple graphs, showcasing your summer 2019 fitness journey with clarity and depth.

The line graph profiled what your typical workout sessions look like: - Warm-ups take around 20 minutes to ramp your heart rate to 140 bpm—a steady build. - Your peak heart rate (~150 bpm) hits between 20–30 minutes, showing a strong push. - Between 30–45 minutes, your heart rate eases to ~125 bpm, a controlled descent. - You catch a second wind at 45 minutes, steadily climbing until the end

The scatter plot with linear OLS regression tracked your average HR rising from 120 to 145 bpm from July to October, signaling workouts growing in intensity over these months. The LOWESS trendline spiked mid-summer, peaking in August —- likely the summer heat amping up cardiac stress, pushing your average HR higher. This visual layered regression to spotlight that mid-summer intensity surge.

The violin plots (upgraded from the simple histogram) deconstructed your heart rate distributions by month, revealing where your efforts landed across zones: - July's exercises clustered in the fat-burning zone (106–126 bpm) - August shifted hard into the aerobic/cardio zone (126+ bpm); this is your peak effort month - September dialed back slightly but stayed mostly in the aerobic/cardio zone - October ramps up again, though with only one workout, monthly generalizations shouldn't be made

1.8 Suggestions for Your Fitness Goals

1. If you would like to focus on heart health: > We can see from the data that you're already a cardio champ, especially in August! To keep boosting your cardiovascular endurance, you can continue to lean into those longer workout sessions (where your HR often hits 140+ bpm), sustaining 30–45 minutes in the aerobic/cardio zone (126–150 bpm). Your "second wind pattern" shows that you have late game stamina; knowing that can help you push through the plateau and back into your aerobic/cardio zone by around Minute 70.

2. If you would like to focus on fat reduction: > July's fat-burning focus (106–126 bpm) was spot-on! You could revisit that with more of the exercises that you completed during that month. Keeping your HR steady at 110–120 bpm would be crucial for this. You can mix in some August-style intensity (short cardio bursts) to keep things interesting, but prioritize

duration in the fat-burning zone over the aerobic/cardio zone. The data shows you have great stamina in the aerobic/cardio zone (which is very difficult to do!), so keeping your HR steady with a lower intensity workout will likely be not too challenging for you.