Running Workout Statistics and Graphs

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from datetime import datetime

plt.style.use('ggplot')

start = time.perf_counter()
```

```
df = pd.read_csv('HeartWatch-Workouts-20230718-to-20230822.csv')
```

```
df = df.drop(['Date', 'from', 'to', 'rpe', 'Load', 'bpm-lo', 'bpm-90%+-%', '90%+-mins',
              'bpm-80-90%-%', '80-90%-mins','bpm-70-80%-%', '70-80%-mins','bpm-60-70%-%',
              '60-70%-mins','bpm-50-60%-%', '50-60%-mins'], axis=1)

# Drop Run w/ Bear
df = df.drop(19)
```

```
# Fix Datetime Columns
df['ISO'] = pd.to_datetime(df['ISO'])
df['Duration'] = pd.to_timedelta(df['Duration'])
df['/km'] = pd.to_timedelta(df['/km'])

df['ISO'] = df['ISO'].dt.date

df.head()
```

	ISO	Duration	Type	bpm-Avg.	bpm-%	bpm-hi	Cals	Cals/h	km	km/h	/
0	2023-07-19	0 days 00:30:04	Running	155.9	81.4	164.0	289.8	578.3	3.81	7.6	0
1	2023-07-20	0 days 00:36:56	Running	157.3	82.1	170.0	399.0	648.0	5.06	8.2	0
2	2023-07-21	0 days 00:31:31	Running	152.8	79.7	181.0	302.5	575.7	4.08	7.8	0
3	2023-07-24	0 days 01:12:34	Cycling	135.6	70.8	188.0	277.1	229.1	17.31	14.3	0
4	2023-07-25	0 days 02:16:52	Cycling	143.6	74.9	182.0	661.4	290.0	35.16	15.4	0

```
dfRun = df[df['Type'] == 'Running']
dfBike = df[df['Type'] == 'Cycling']
dfOther = df[~df['Type'].isin(['Running', 'Cycling'])]

# print(dfRun.info())
# print(dfBike.info())
# print(dfOther.head())
```

```
dfRun.tail()
```

	ISO	Duration	Type	bpm-Avg.	bpm-%	bpm-hi	Cals	Cals/h	km	km/h
23	2023-08-17	0 days 00:41:24	Running	157.9	82.4	189.0	498.9	723.0	6.12	8.9
24	2023-08-18	0 days 00:49:30	Running	139.7	72.9	162.0	550.8	667.5	6.64	8.0
25	2023-08-18	0 days 00:49:51	Running	148.9	77.7	170.0	591.6	712.0	7.09	8.5
27	2023-08-21	0 days 00:57:06	Running	149.2	77.9	166.0	690.5	725.4	8.25	8.7
29	2023-08-22	0 days 01:21:26	Running	160.2	83.6	191.0	1008.3	742.9	12.61	9.3

```
# Pace Calculation
# Calculate the total seconds of Duration column
total_seconds = dfRun['Duration'].dt.total_seconds()

# Average Pace from M/S
mps = dfRun['km'].sum()*1000 / total_seconds.sum()
kph = mps * 3.6
mpk = 60 / kph

integer_part = int(mpk)
decimal_part = mpk - integer_part

# Convert decimal part to minutes by dividing by 60
decimal_minutes = round(decimal_part * 60,0)
```

```
# Weighted HR
dfRunWght = dfRun

# Convert the time delta to decimal hours and create a new column
dfRunWght["Duration"] = dfRunWght["Duration"].apply(lambda x: x.total_seconds() / 3600)
dfRunWght['Weighted HR'] = dfRunWght['Duration'] * dfRunWght['bpm-Avg.']

# HR/Speed Decimal
dfRun['HR/Speed'] = (dfRun['bpm-Avg.'] / dfRun['km/h'])
```

```
/var/folders/y_/8cmvsg791ys9qqbzj_yxrlj40000gn/T/ipykernel_28788/3208722300.py:5: SettingWitl
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  dfRunWght["Duration"] = dfRunWght["Duration"].apply(lambda x: x.total_seconds() / 3600)
/var/folders/y_/8cmvsg791ys9qqbzj_yxrlj40000gn/T/ipykernel_28788/3208722300.py:6: SettingWitl
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  dfRunWght['Weighted HR'] = dfRunWght['Duration'] * dfRunWght['bpm-Avg.']
/var/folders/y_/8cmvsg791ys9qqbzj_yxrlj40000gn/T/ipykernel_28788/3208722300.py:9: SettingWitl
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  dfRun['HR/Speed'] = (dfRun['bpm-Avg.'] / dfRun['km/h'])
```

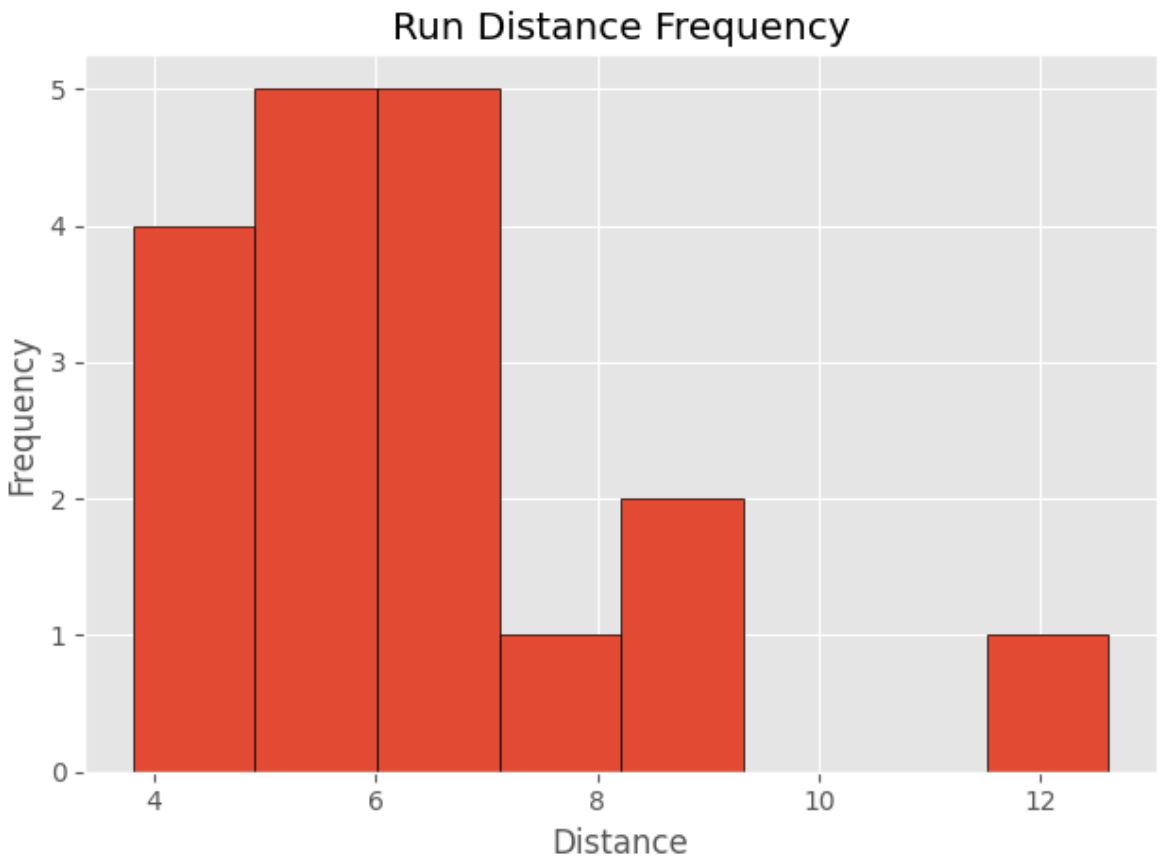
Graphs

Running Distance Frequency

```
rdf_bins = int(dfRun['km'].max() - dfRun['km'].min())

plt.hist(dfRun['km'], edgecolor='black', bins=rdf_bins)
plt.title('Run Distance Frequency')
plt.xlabel('Distance')
plt.ylabel('Frequency')
# plt.xticks(rotation=45)
```

```
plt.tight_layout()
plt.savefig('Graphs/Distance Frequency.png', dpi=300)
plt.show()
```

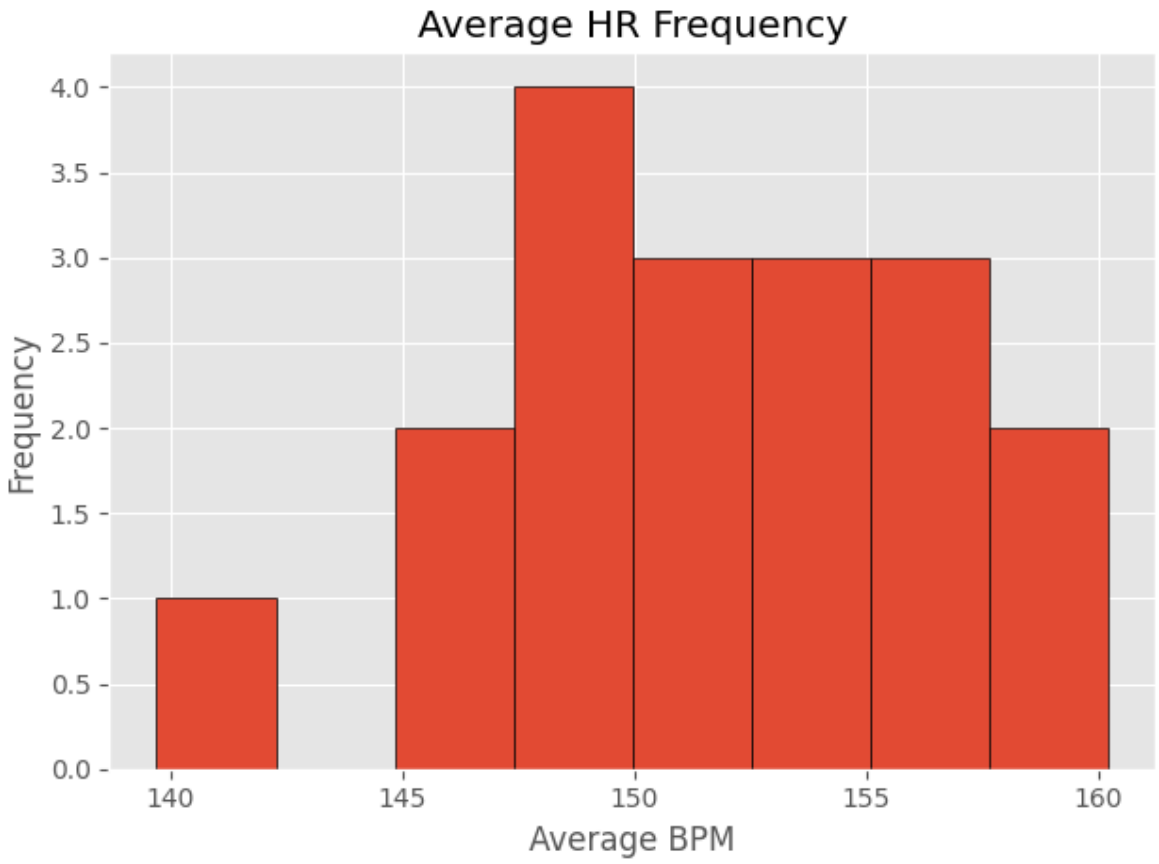


Average Heart Rate Frequency

```
hrf_bins = int((dfRun['bpm-Avg.'].max() - dfRun['bpm-Avg.'].min())/2.5)

plt.hist(dfRun['bpm-Avg.'], edgecolor='black', bins=hrf_bins)
plt.title('Average HR Frequency')
plt.xlabel('Average BPM')
plt.ylabel('Frequency')
# plt.xticks(rotation=45)
plt.tight_layout()
```

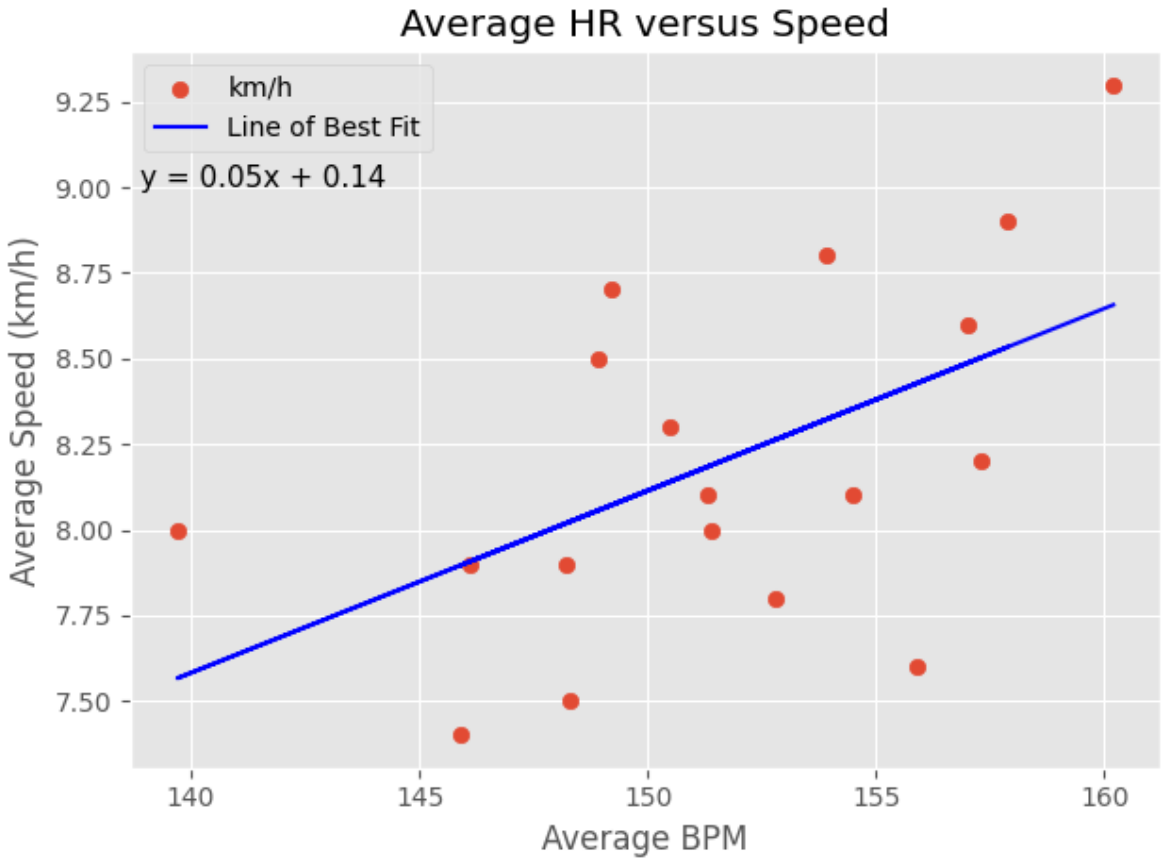
```
plt.savefig('Graphs/HR Frequency.png', dpi=300)
plt.show()
```



Average Heart Rate versus Average Speed (km/h)

```
# Line of Best Fit
# Fit a linear regression line to the data
degree = 1
coefficients = np.polyfit(dfRun['bpm-Avg.'], dfRun['km/h'], degree)
slope = coefficients[0]
intercept = coefficients[1]
# Calculate the predicted y-values using the line equation
predicted_y = slope * dfRun['bpm-Avg.'] + intercept
equation = f'y = {slope:.2f}x + {intercept:.2f}'
```

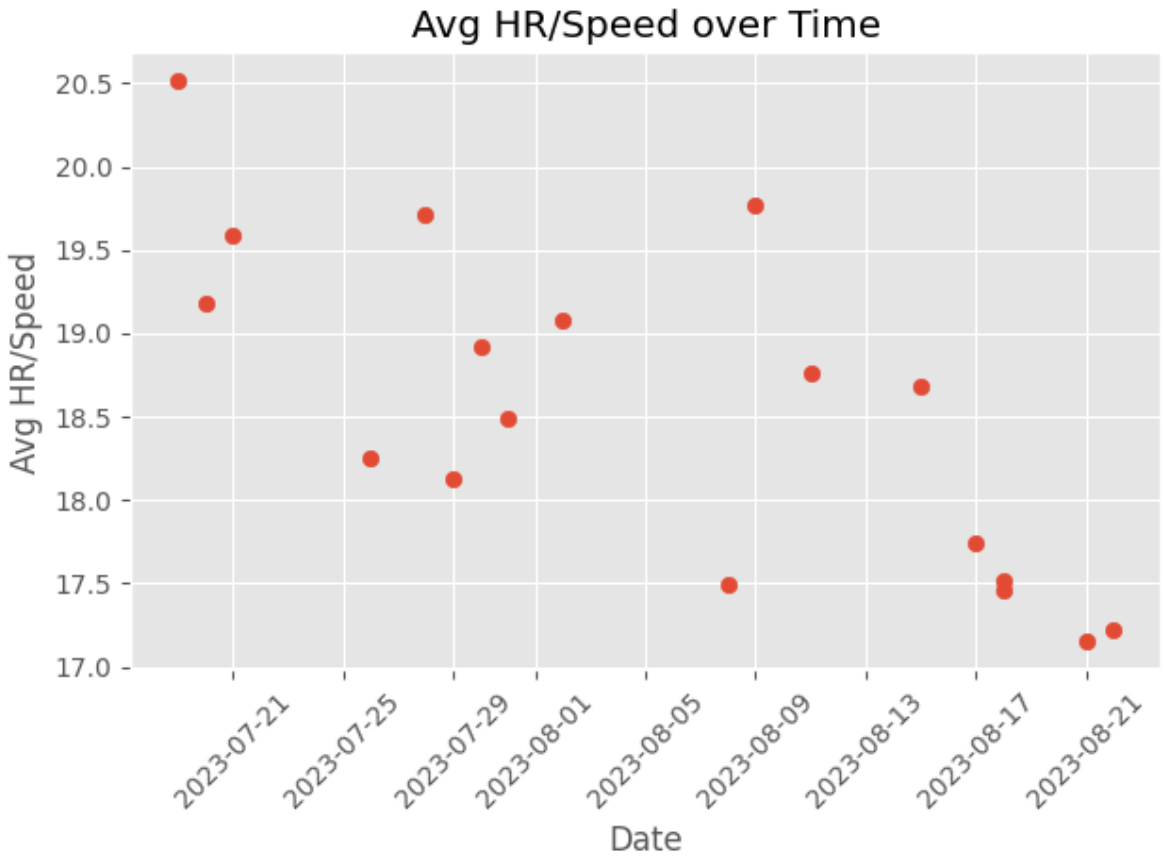
```
# Plot
plt.scatter('bpm-Avg.', 'km/h', data=dfRun)
plt.plot(dfRun['bpm-Avg.'], predicted_y, color='blue', label='Line of Best Fit')
plt.xlabel('Average BPM')
plt.ylabel('Average Speed (km/h)')
plt.title('Average HR versus Speed')
plt.text(0.01, 0.81, equation, fontsize=11, transform=plt.gca().transAxes)
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig('Graphs/SpeedvsHR.png', dpi=300)
plt.show()
```



Heart Rate / Speed over Time

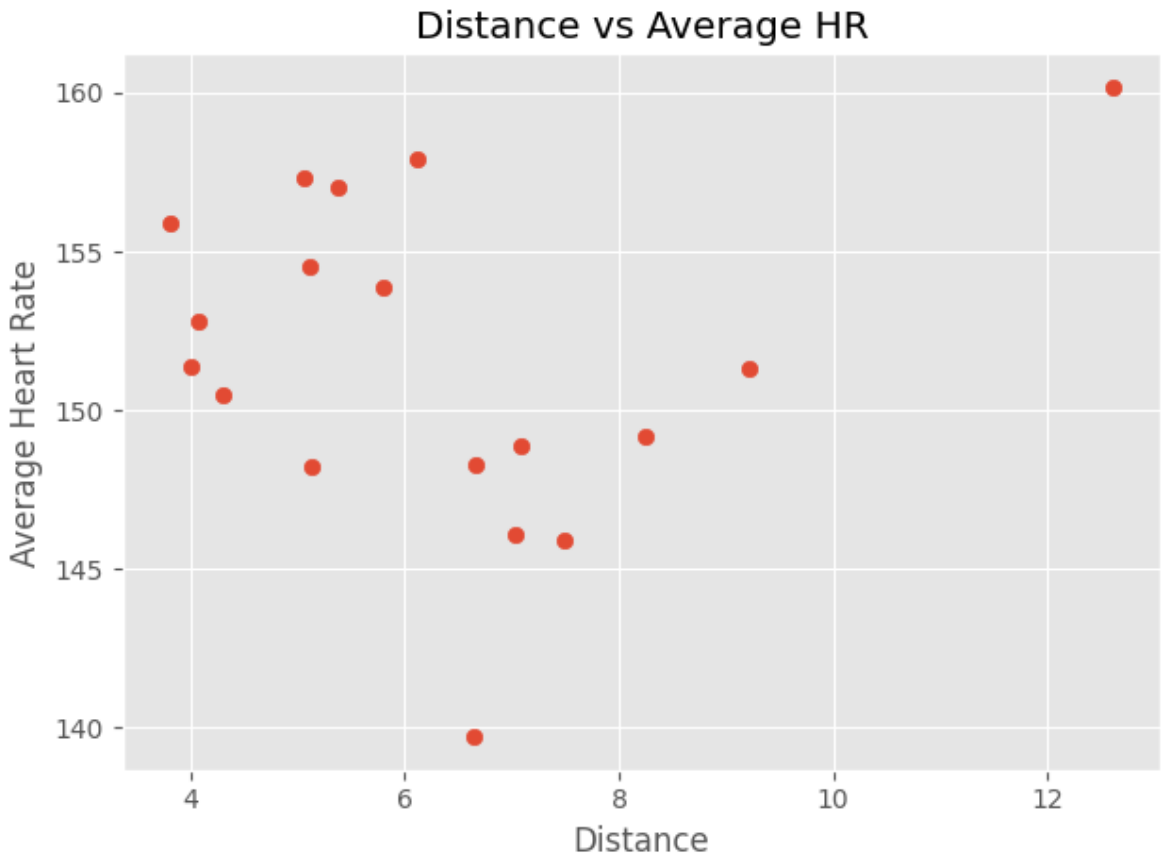
Lower is better

```
plt.scatter('ISO', 'HR/Speed', data=dfRun)
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.ylabel('Avg HR/Speed')
plt.title('Avg HR/Speed over Time')
plt.grid(True)
plt.tight_layout()
plt.savefig('Graphs/HR-Speed over Time.png', dpi=300)
plt.show()
```



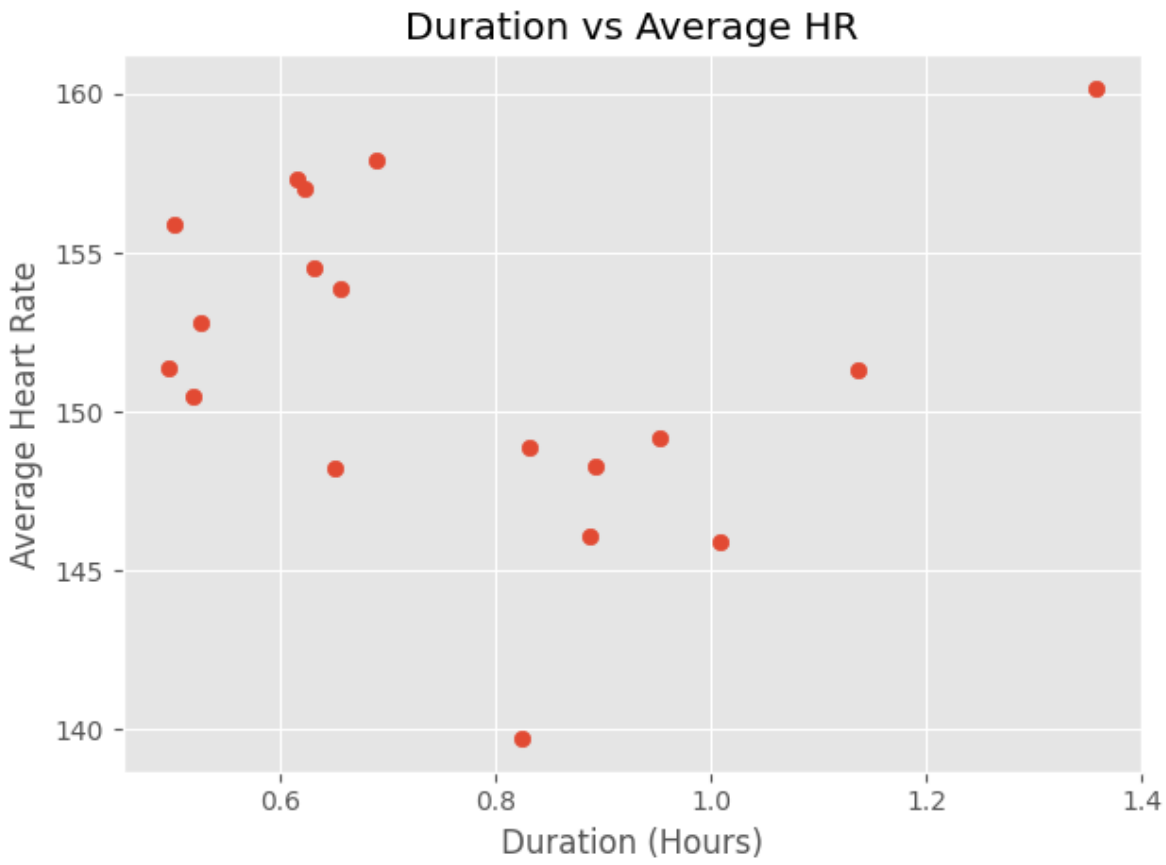
Distance vs Average Heart Rate

```
plt.scatter('km', 'bpm-Avg.', data=dfRun)
plt.xlabel('Distance')
plt.ylabel('Average Heart Rate')
plt.title('Distance vs Average HR')
plt.grid(True)
plt.tight_layout()
plt.savefig('Graphs/Distance vs Avg HR.png', dpi=300)
```



Duration vs Average Heart Rate

```
plt.scatter('Duration', 'bpm-Avg.', data=dfRun)
plt.xlabel('Duration (Hours)')
plt.ylabel('Average Heart Rate')
plt.title('Duration vs Average HR')
plt.grid(True)
plt.tight_layout()
plt.savefig('Graphs/Duration vs Avg HR.png', dpi=300)
```




```
# Average Duration, Distance, Average HR, Average Max HR Average Calories
avg_dist = round(dfRun['km'].mean(),2)
avg_hr = round(dfRun['bpm-Avg.'].mean(),2)
avg_wght_hr = round(dfRunWght['Weighted HR'].sum() / dfRunWght['Duration'].sum(),2)
avg_maxhr = round(dfRun['bpm-hi'].mean(),2)
```

```
avg_cals = round(dfRun['Cals'].mean(),2)
avg_dur = dfRun['Duration'].mean()

# Count Runs
num_runs = dfRun['km'].count()
ovr_5k = dfRun[dfRun['km'] >=5].count()['km']
povr_5k = round(ovr_5k / num_runs *100,2)
ovr_10k = dfRun[dfRun['km'] >=10].count()['km']
povr_10k = round(ovr_10k / num_runs *100,2)

# Maximums
max_dur = dfRun['Duration'].max()
max_dist = dfRun['km'].max()
max_avghr = dfRun['bpm-Avg.'].max()
max_maxhr = dfRun['bpm-hi'].max()
max_cals = dfRun['Cals'].max()

# Totals
tot_dist = round(dfRun['km'].sum(),2)
tot_dur = dfRun['Duration'].sum()
tot_cals = round(dfRun['Cals'].sum(),2)

# Medians
med_dist = round(dfRun['km'].median(),2)
med_avg_hr = round(dfRun['bpm-Avg.'].median(),2)
med_max_hr = round(dfRun['bpm-hi'].median(),2)
med_cals = round(dfRun['Cals'].median(),2)

# Durations to Time Format
avg_dur_h = int(avg_dur)
max_dur_h = int(max_dur)
tot_dur_h = int(tot_dur)

avg_dur_m_dec = (avg_dur - avg_dur_h)*60
max_dur_m_dec = (max_dur - max_dur_h)*60
tot_dur_m_dec = (tot_dur - tot_dur_h)*60

avg_dur_m = int(avg_dur_m_dec)
max_dur_m = int(max_dur_m_dec)
tot_dur_m = int(tot_dur_m_dec)
```

```
avg_dur_s = int((avg_dur_m_dec - avg_dur_m)*60)
max_dur_s = int((max_dur_m_dec - max_dur_m)*60)
tot_dur_s = int((tot_dur_m_dec - tot_dur_m)*60)

avg_dur_f = str(avg_dur_h) + ':' + str(avg_dur_m) + ':' + str(avg_dur_s)
max_dur_f = str(max_dur_h) + ':' + str(max_dur_m) + ':' + str(max_dur_s)
tot_dur_f = str(tot_dur_h) + ':' + str(tot_dur_m) + ':' + str(tot_dur_s)
```

Final Running Stats

```
print(f'Runs: {num_runs}')
print(f'Runs over 5k(%): {ovr_5k} ({povr_5k}%)')
print(f'Runs over 10k(%): {ovr_10k} ({povr_10k}%)')

print('----------------------------')
print(f'Average Duration: {avg_dur_f}')
print(f'Average Distance: {avg_dist}')
print(f"Average Pace: {integer_part}:{decimal_minutes}")
print(f'Average Weighted HR: {avg_wght_hr}')
print(f'Average HR: {avg_hr}')
print(f'Average Max HR: {avg_maxhr}')
print(f'Average Calories: {avg_cals}')

print('----------------------------')
print(f'Max Duration: {max_dur_f}')
print(f'Max Distance: {max_dist}')
print(f'Max Average HR: {max_avghr}')
print(f'Max Max HR: {max_maxhr}')
print(f'Max Calories: {max_cals}')

print('----------------------------')
print(f'Median Distance: {med_dist}')
print(f'Median Avg HR: {med_avg_hr}')
print(f'Median Max HR: {med_max_hr}')
print(f'Median Calories: {med_cals}')

print('----------------------------')
print(f'Total Duration: {tot_dur_f}')
print(f'Total Distance: {tot_dist}')
print(f'Total Calories Burn: {tot_cals}')
```

```
print('----------------------------')
print(f'Runtime: {round(time.perf_counter() - start,2)}s')
```

```
Runs: 18
Runs over 5k(%): 14 (77.78%)
Runs over 10k(%): 1 (5.56%)
----------------------------
Average Duration: 0:46:0
Average Distance: 6.32
Average Pace: 7:17.0
Average Weighted HR: 151.45
Average HR: 151.61
Average Max HR: 170.11
Average Calories: 517.15
----------------------------
Max Duration: 1:21:26
Max Distance: 12.61
Max Average HR: 160.2
Max Max HR: 191.0
Max Calories: 1008.3
----------------------------
Median Distance: 5.96
Median Avg HR: 151.35
Median Max HR: 167.0
Median Calories: 489.65
----------------------------
Total Duration: 13:48:7
Total Distance: 113.82
Total Calories Burn: 9308.7
----------------------------
Runtime: 1.61s
```