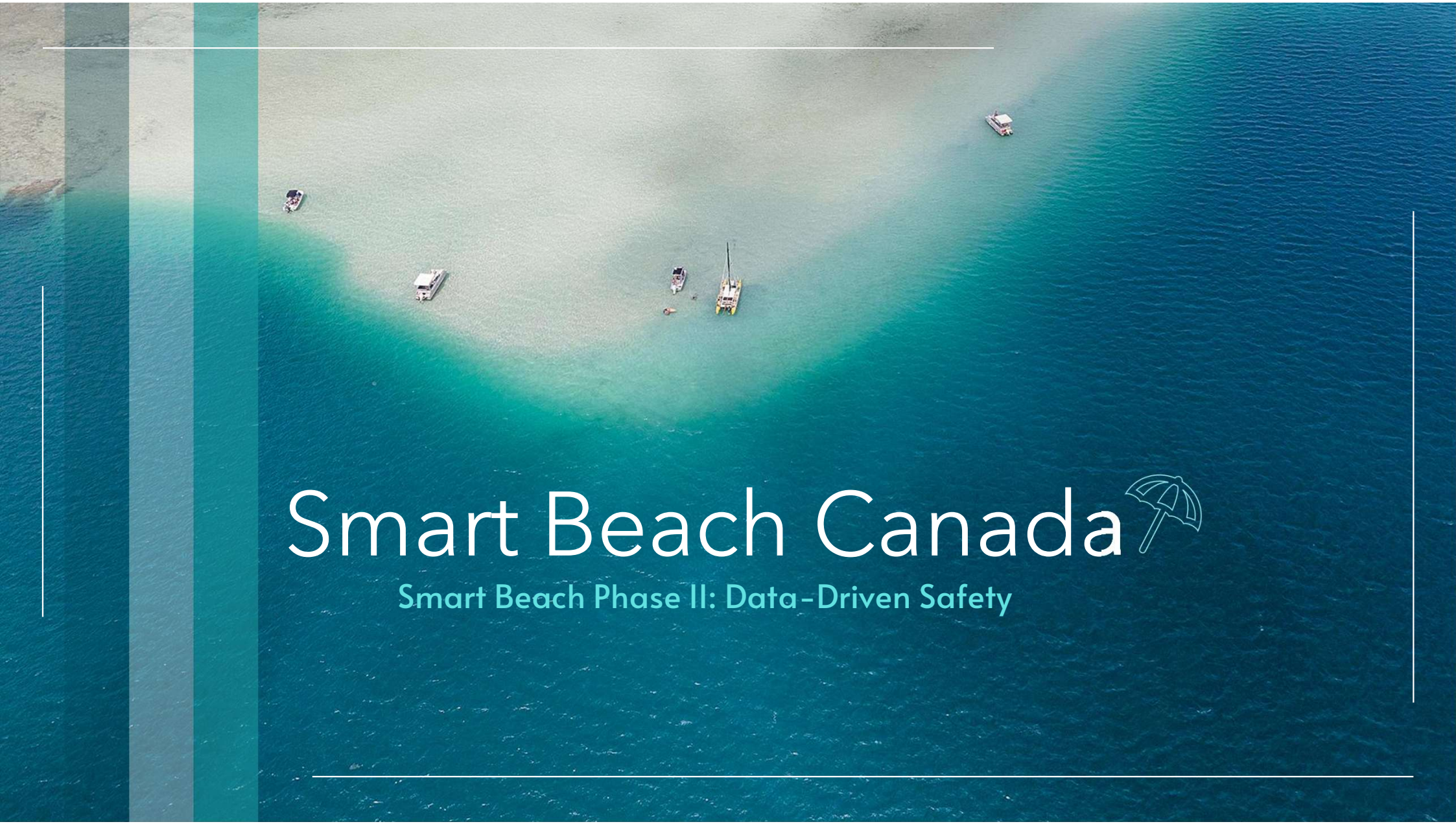


# Smart Beach Canada

## Smart Beach Phase II: Data-Driven Safety

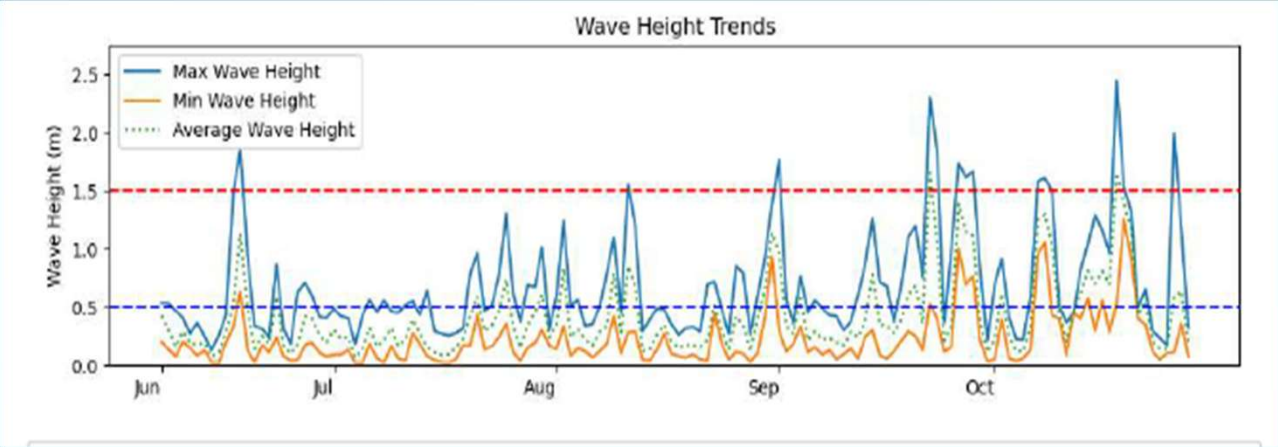
# Wave Height Exploration

## Wave Height Trends: June - October

```
In [43]: merged_df.groupby('conditions')[['maxwaveheight']].agg(['count','mean','median','min','max'])
```

Out[43]:

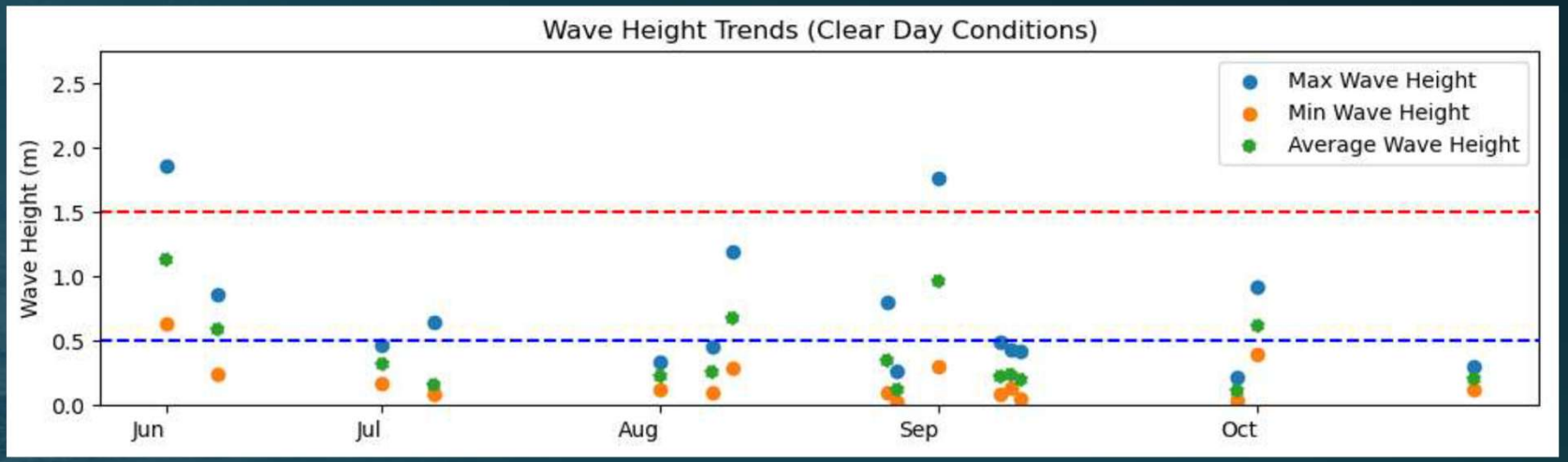
| | maxwaveheight | | | | |
|---|---|---|---|---|---|
| | count | mean | median | min | max |
| conditions | | | | | |
| clear-day | 16 | 0.713188 | 0.479 | 0.215 | 1.855 |
| partly-cloudy-day | 58 | 0.574448 | 0.455 | 0.137 | 1.826 |
| rain | 69 | 0.783015 | 0.589 | 0.261 | 2.447 |
| snow | 1 | 1.538000 | 1.538 | 1.538 | 1.538 |



# Wave Height Trends

## Clear Day Conditions

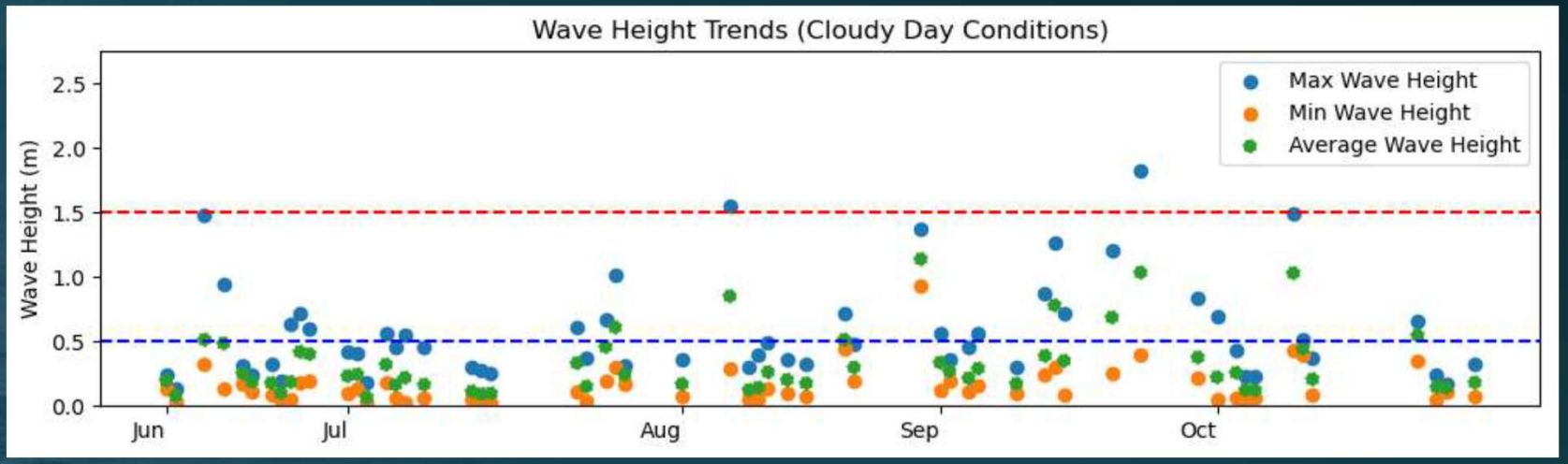
| conditions | maxwaveheight count | mean | median | min | max |
|---|---|---|---|---|---|
| clear-day | 16 | 0.713188 | 0.479 | 0.215 | 1.855 |
| partly-cloudy-day | 58 | 0.574448 | 0.455 | 0.137 | 1.826 |
| rain | 69 | 0.783015 | 0.589 | 0.261 | 2.447 |
| snow | 1 | 1.538000 | 1.538 | 1.538 | 1.538 |



# Wave Height Trends

Partly Cloudy Days

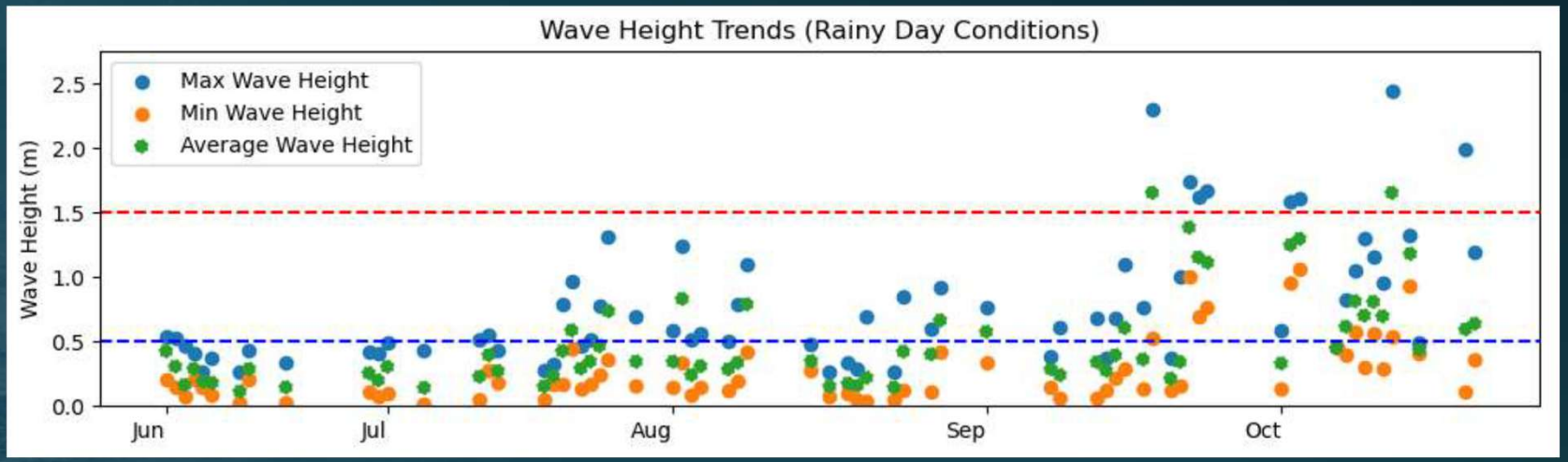
| | maxwaveheight | | | | |
|---|---|---|---|---|---|
| | count | mean | median | min | max |
| conditions | | | | | |
| clear-day | 16 | 0.713188 | 0.479 | 0.215 | 1.855 |
| partly-cloudy-day | 58 | 0.574448 | 0.455 | 0.137 | 1.826 |
| rain | 69 | 0.783015 | 0.589 | 0.261 | 2.447 |
| snow | 1 | 1.538000 | 1.538 | 1.538 | 1.538 |



# Wave Height Trends

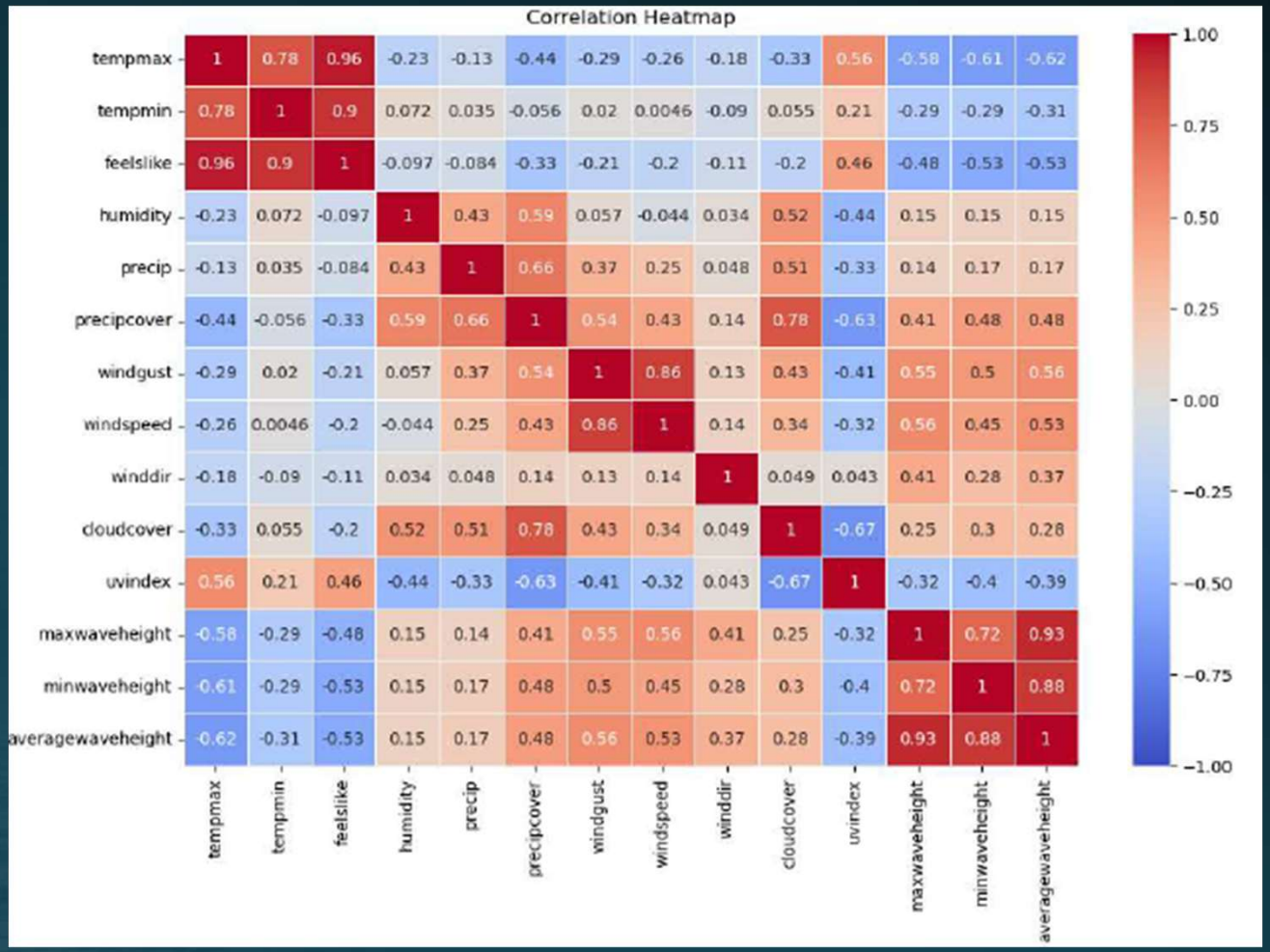
Rainy Days

| | maxwaveheight | | | | |
|---|---|---|---|---|---|
| | count | mean | median | min | max |
| conditions | | | | | |
| clear-day | 16 | 0.713188 | 0.479 | 0.215 | 1.855 |
| partly-cloudy-day | 58 | 0.574448 | 0.455 | 0.137 | 1.826 |
| rain | 69 | 0.783015 | 0.589 | 0.261 | 2.447 |
| snow | 1 | 1.538000 | 1.538 | 1.538 | 1.538 |

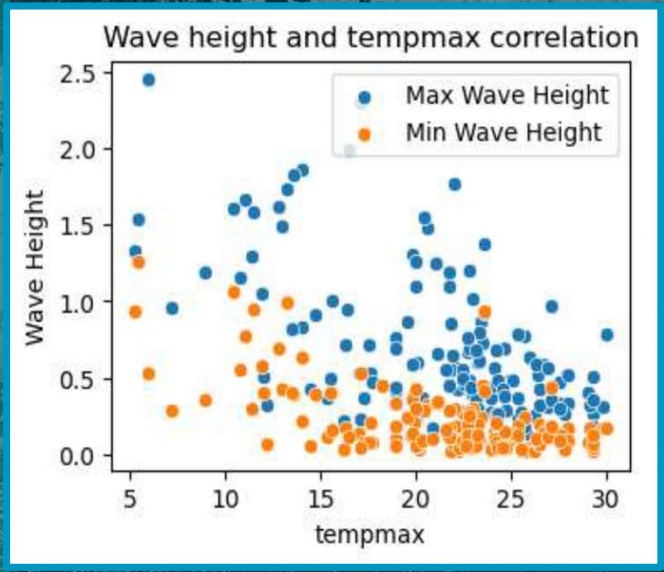


# Correlation Heatmap

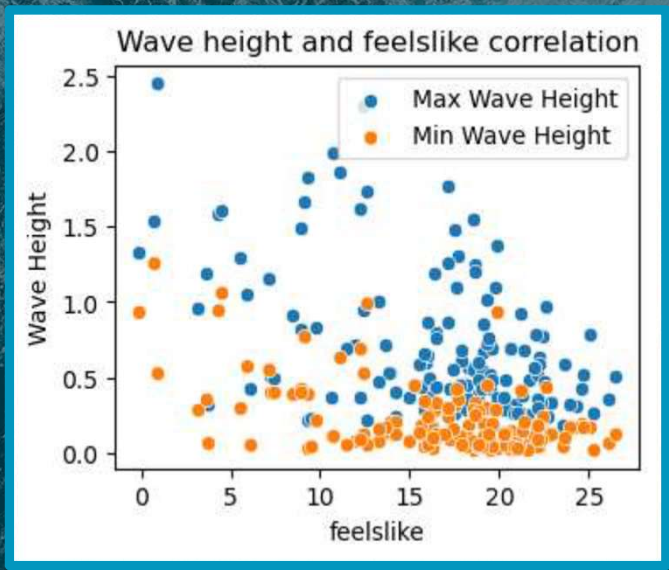
Wave Height vs Weather Variables



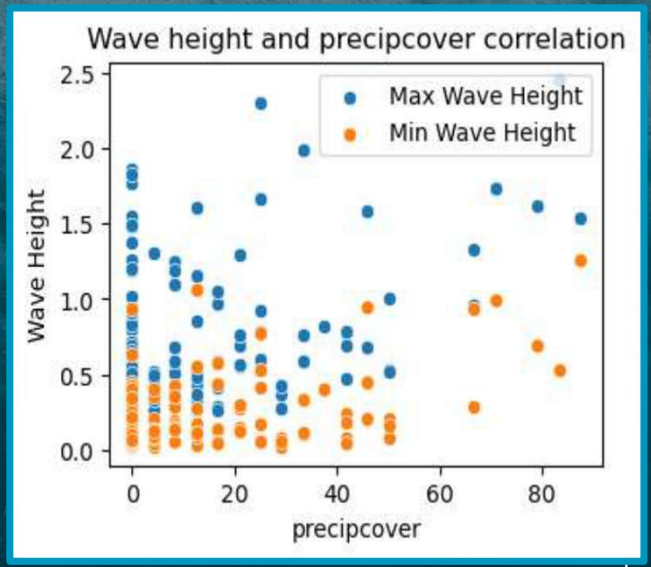
# Correlation of wave height and other Variables




| | |
|---|---|
| maxwaveheight | -0.58 |
| maxwaveheight | -0.61 |

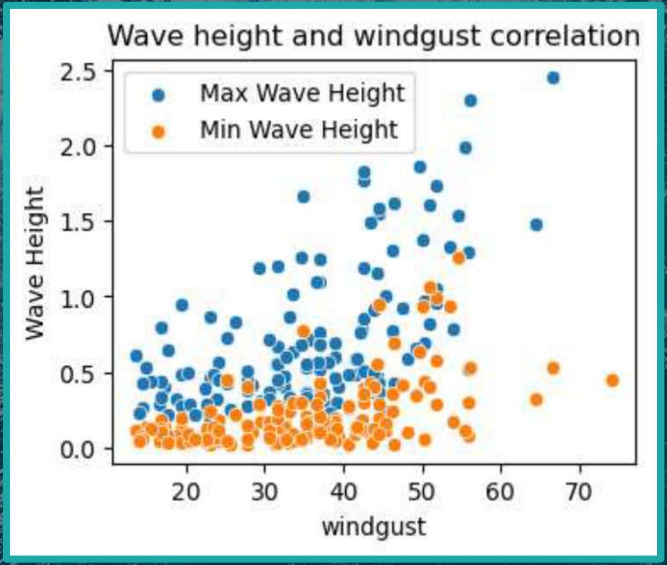



| | |
|---|---|
| maxwaveheight | -0.48 |
| minwaveheight | -0.53 |

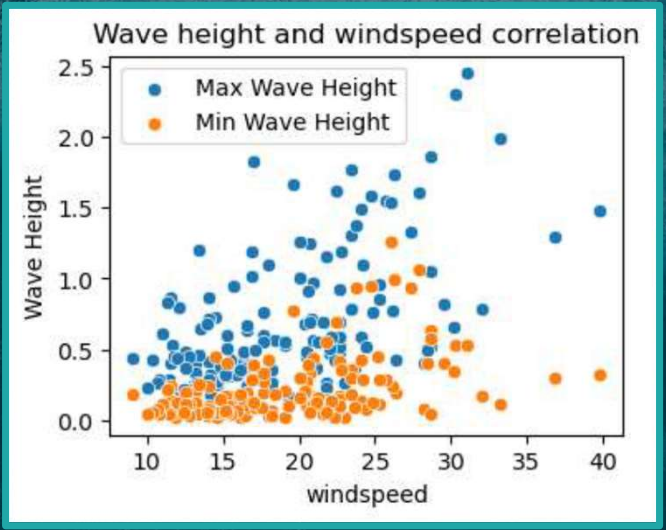



| | |
|---|---|
| maxwaveheight | 0.41 |
| minwaveheight | 0.48 |

# Correlation of wave height and other Variables




| | |
|---|---|
| maxwaveheight | 0.55 |
| maxwaveheight | 0.5 |




| | |
|---|---|
| maxwaveheight | 0.56 |
| minwaveheight | 0.45 |

# Wind Direction Conversion: Degrees to Direction

```
def wind_direction(x):
    if 0 <= x < 45:
        return 'N-NE'
    elif 45 <= x < 90:
        return 'NE-E'
    elif 90 <= x < 135:
        return 'E-SE'
    elif 135 <= x < 180:
        return 'SE-S'
    elif 180 <= x < 225:
        return 'S-SW'
    elif 225 <= x < 270:
        return 'SW-W'
    elif 270 <= x < 315:
        return 'W-NW'
    elif 315 <= x < 360:
        return 'NW-N'
    else:
        return 'Unknown'

merged_df['windir_cat'] = merged_df['winddir'].apply(wind_direction)

print(merged_df)
```

# Maximum Wave Height with Wind Direction

| windir_cat | count | mean | median | min | max |
|---|---|---|---|---|---|
| W-NW | 28 | 0.942071 | 0.6635 | 0.228 | 2.447 |
| NW-N | 20 | 1.082350 | 1.0240 | 0.276 | 2.305 |
| N-NE | 12 | 0.576667 | 0.4910 | 0.215 | 1.665 |
| SW-W | 21 | 0.752429 | 0.6740 | 0.261 | 1.611 |
| S-SW | 36 | 0.519389 | 0.4630 | 0.189 | 1.297 |
| NE-E | 8 | 0.462125 | 0.4125 | 0.137 | 0.915 |
| E-SE | 4 | 0.526500 | 0.5190 | 0.271 | 0.797 |
| SE-S | 15 | 0.336533 | 0.3050 | 0.173 | 0.606 |

# Exploration of People Data

```
In [ ]: #People count

In [190]: PCdf = pd.read_csv('image_counts_all.csv')

In [191]: PCdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   File_name        312 non-null    object
 1   Year             312 non-null    int64 
 2   Total People     312 non-null    int64 
 3   People in Water  312 non-null    int64 
 4   People on Sand   312 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 12.3+ KB
```

```
In [195]: PCdf.head()
```

Out[195]:

| | File_name | Year | Total People | People in Water | People on Sand |
|---|---|---|---|---|---|
| 0 | 2022-08-25_15-35-39_jpg.rf.f29c550c1dfa9e50ded... | 2022 | 14 | 1 | 13 |
| 1 | 2022-08-25_15-35-40_jpg.rf.5bfb5ece1d5e39664fd... | 2022 | 16 | 2 | 14 |
| 2 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 |
| 3 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 |
| 4 | 2022-08-25_15-40-27_jpg.rf.ceb8e5145f8ec9240ed... | 2022 | 15 | 3 | 12 |

# Removing Extra Copies

```
In [199]: PCdf.File_name.str.contains('Copy')

Out[199]: 0      False
          1      False
          2       True
          3      False
          4       True
                 ...
          307    False
          308    False
          309    False
          310    False
          311    False
          Name: File_name, Length: 312, dtype: bool
```

```
print(PeopleCountdf)
```

| | File_name | Year | Total People | People in Water | People on Sand |
|---|---|---|---|---|---|
| 0 | 2022-08-25_15-35-39_jpg.rf.f29c550c1dfa9e50ded... | 2022 | 14 | 1 | 13 |
| 1 | 2022-08-25_15-35-40_jpg.rf.5bfb5ece1d5e39664fd... | 2022 | 16 | 2 | 14 |
| 3 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 |
| 5 | 2022-08-25_15-40-27_jpg.rf.ceb8e5145f8ec9240ed... | 2022 | 15 | 3 | 12 |
| 7 | 2022-08-25_15-41-33_jpg.rf.5d69e86774bd8db8810... | 2022 | 18 | 3 | 15 |
| .. | ... | ... | ... | ... | ... |
| 307 | 2023-08-17_06-40-17_jpg.rf.1a45cf9a03b13987afb... | 2023 | 0 | 0 | 0 |
| 308 | 2023-08-21_15-55-19_jpg.rf.545cc1b11449621bb2b... | 2023 | 19 | 3 | 16 |
| 309 | 2023-08-21_17-35-20_jpg.rf.3abe7a112ec79a468be... | 2023 | 20 | 0 | 20 |
| 310 | 2023-08-22_06-20-17_jpg.rf.e0d9e001700c794b191... | 2023 | 0 | 0 | 0 |
| 311 | 2023-08-22_08-10-18_jpg.rf.4f2622d52d4785cfa9c... | 2023 | 2 | 0 | 2 |

[261 rows x 5 columns]

# Extracting Data and Time from File_name

```
In [215]: PeopleCountdf = PeopleCountdf.copy()

          PeopleCountdf['DateTime'] = PeopleCountdf['File_name'].str.extract(r'(\d{4}-\d{2}-\d{2}_\d{2}-\d{2}-\d{2})')
          PeopleCountdf['DateTime'] = pd.to_datetime(PeopleCountdf['DateTime'], format='%Y-%m-%d_%H-%M-%S', errors='coerce')
```

```
In [216]: PeopleCountdf.head()
```

Out[216]:

| | File_name | Year | Total People | People in Water | People on Sand | DateTime |
|---|---|---|---|---|---|---|
| 0 | 2022-08-25_15-35-39_jpg.rf.f29c550c1dfa9e50ded... | 2022 | 14 | 1 | 13 | 2022-08-25 15:35:39 |
| 1 | 2022-08-25_15-35-40_jpg.rf.5bfb5ece1d5e39664fd... | 2022 | 16 | 2 | 14 | 2022-08-25 15:35:40 |
| 3 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 | 2022-08-25 15:38:19 |
| 5 | 2022-08-25_15-40-27_jpg.rf.ceb8e5145f8ec9240ed... | 2022 | 15 | 3 | 12 | 2022-08-25 15:40:27 |
| 7 | 2022-08-25_15-41-33_jpg.rf.5d69e86774bd8db8810... | 2022 | 18 | 3 | 15 | 2022-08-25 15:41:33 |

```
PeopleCount2022['DateTime'].value_counts()
```

```
2022-08-27 17:20:16    2
2022-08-27 17:00:16    2
2022-08-25 15:35:39    1
2022-09-25 07:10:16    1
2022-09-17 16:10:16    1
                      ..
2022-08-27 16:00:16    1
2022-08-27 16:10:15    1
2022-08-27 16:40:16    1
2022-08-27 17:10:15    1
2022-10-27 08:50:16    1
Name: DateTime, Length: 218, dtype: int64
```

# Limitations Due to No Consistency

```
In [245]: date_counts = PeopleCount2022['DateTime'].dt.date.value_counts().reset_index()
          date_counts.columns = ['Date', 'Count']
          print(date_counts)
```

| | Date | Count |
|---|---|---|
| 0 | 2022-08-27 | 44 |
| 1 | 2022-08-26 | 27 |
| 2 | 2022-08-25 | 15 |
| 3 | 2022-08-28 | 14 |
| 4 | 2022-08-29 | 7 |
| 5 | 2022-09-30 | 7 |
| 6 | 2022-10-18 | 6 |
| 7 | 2022-09-10 | 5 |
| 8 | 2022-09-13 | 5 |
| 9 | 2022-10-14 | 5 |
| 10 | 2022-10-09 | 4 |
| 11 | 2022-09-29 | 4 |
| 12 | 2022-10-01 | 4 |
| 13 | 2022-10-07 | 4 |
| 14 | 2022-09-02 | 4 |
| 15 | 2022-10-19 | 3 |
| 16 | 2022-09-14 | 3 |
| 17 | 2022-09-11 | 3 |
| 18 | 2022-10-08 | 3 |
| 19 | 2022-10-16 | 3 |
| 20 | 2022-09-19 | 3 |
| 21 | 2022-10-10 | 2 |
| 22 | 2022-08-31 | 2 |
| 23 | 2022-10-06 | 2 |
| 24 | 2022-10-20 | 2 |
| 25 | 2022-10-02 | 2 |
| 26 | 2022-10-13 | 2 |
| 27 | 2022-09-07 | 2 |
| 28 | 2022-09-04 | 2 |
| 29 | 2022-09-08 | 2 |
| 30 | 2022-09-25 | 2 |
| 31 | 2022-09-24 | 2 |
| 32 | 2022-09-22 | 2 |
| 33 | 2022-10-21 | 2 |
| 34 | 2022-09-17 | 2 |
| 35 | 2022-09-09 | 2 |
| 36 | 2022-09-27 | 2 |
| 37 | 2022-10-22 | 1 |
| 38 | 2022-10-25 | 1 |
| 39 | 2022-10-23 | 1 |
| 40 | 2022-10-24 | 1 |
| 41 | 2022-09-28 | 1 |
| 42 | 2022-10-15 | 1 |
| 43 | 2022-10-12 | 1 |
| 44 | 2022-10-11 | 1 |
| 45 | 2022-10-05 | 1 |
| 46 | 2022-09-20 | 1 |
| 47 | 2022-09-15 | 1 |
| 48 | 2022-09-12 | 1 |
| 49 | 2022-09-06 | 1 |
| 50 | 2022-09-05 | 1 |
| 51 | 2022-10-27 | 1 |

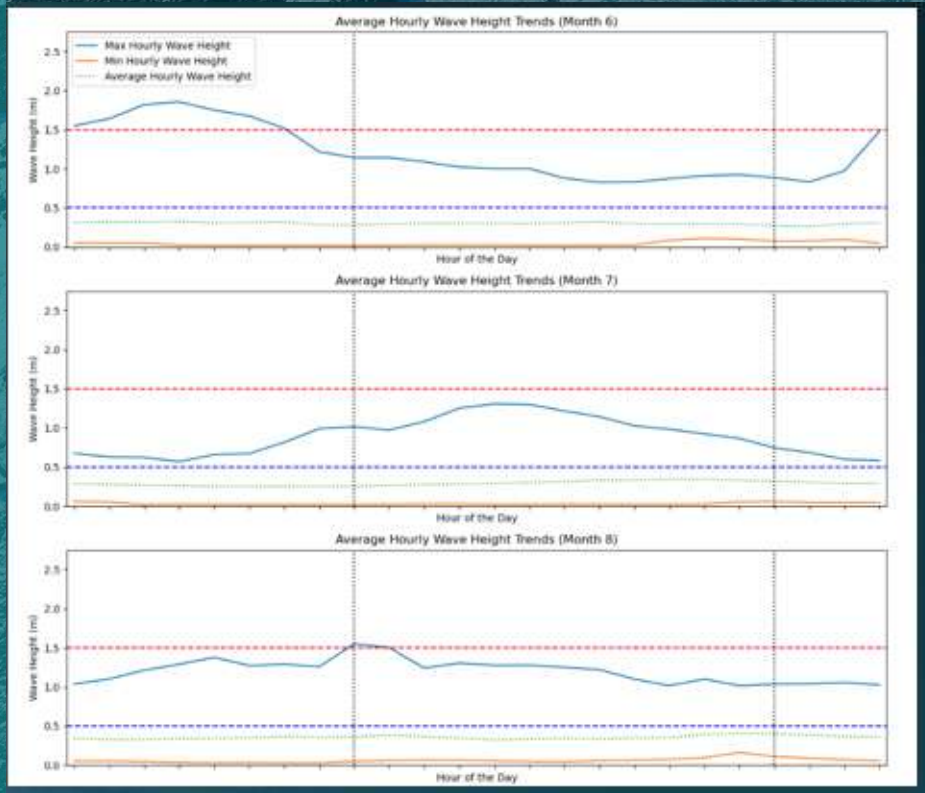
| Year | Total People | People in Water | People on Sand | DateTime |
|---|---|---|---|---|
| 2022 | 14 | 1 | 13 | 2022-08-25 15:35:39 |
| 2022 | 16 | 2 | 14 | 2022-08-25 15:35:40 |
| 2022 | 18 | 5 | 13 | 2022-08-25 15:38:19 |
| 2022 | 15 | 3 | 12 | 2022-08-25 15:40:27 |
| 2022 | 18 | 3 | 15 | 2022-08-25 15:41:33 |

Some days have too many readings while some days do not have enough.

Some readings are from night hours while some are from day

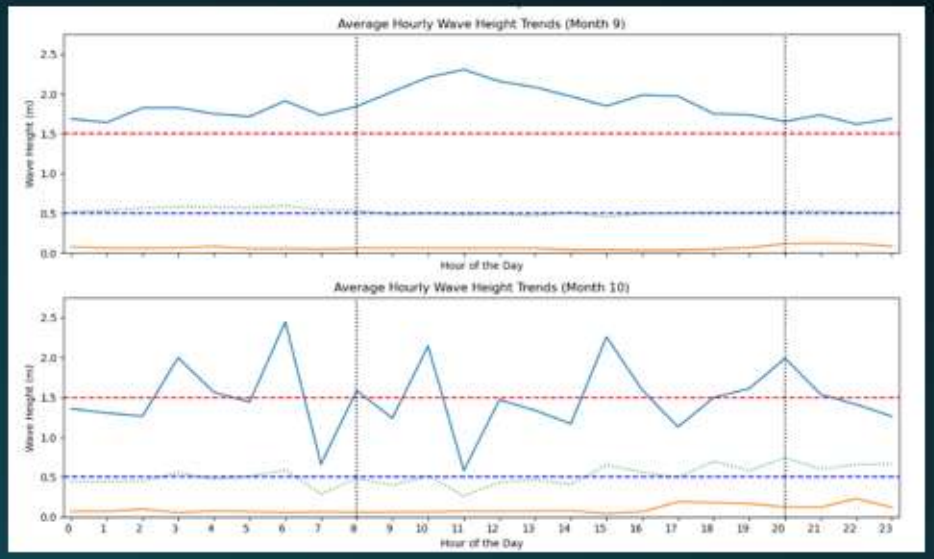
Data inconsistency and limited availability can affect the output negatively

# Why Time Matters when Collecting Data




June - August

Max, Min and Average Wave height trends
Showcasing hourly variations with separators at 8AM and 8PM as a high beach usage band

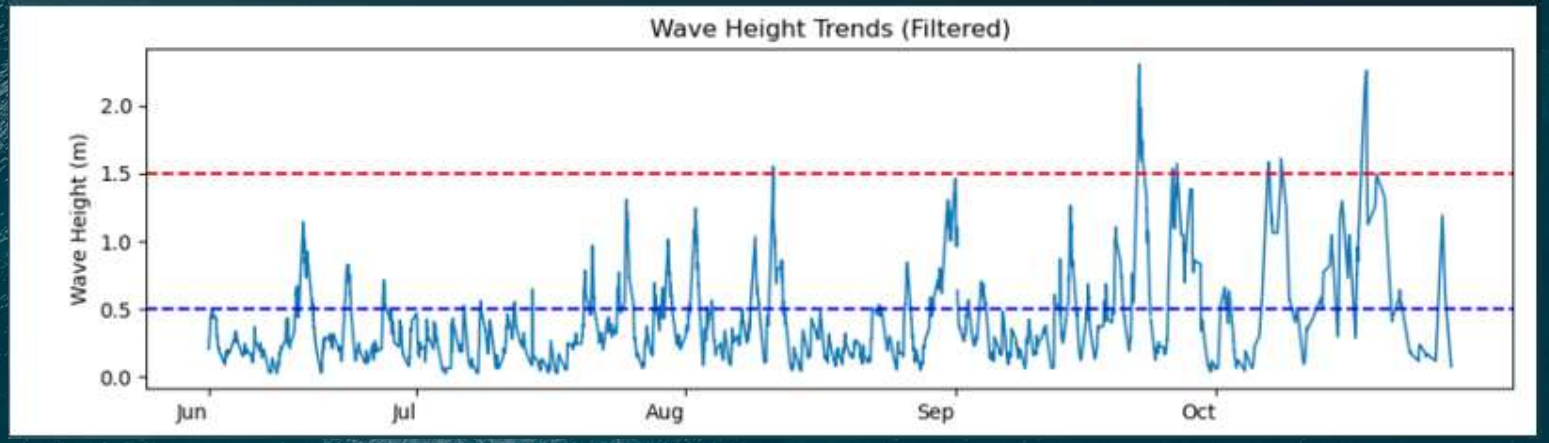



September - October

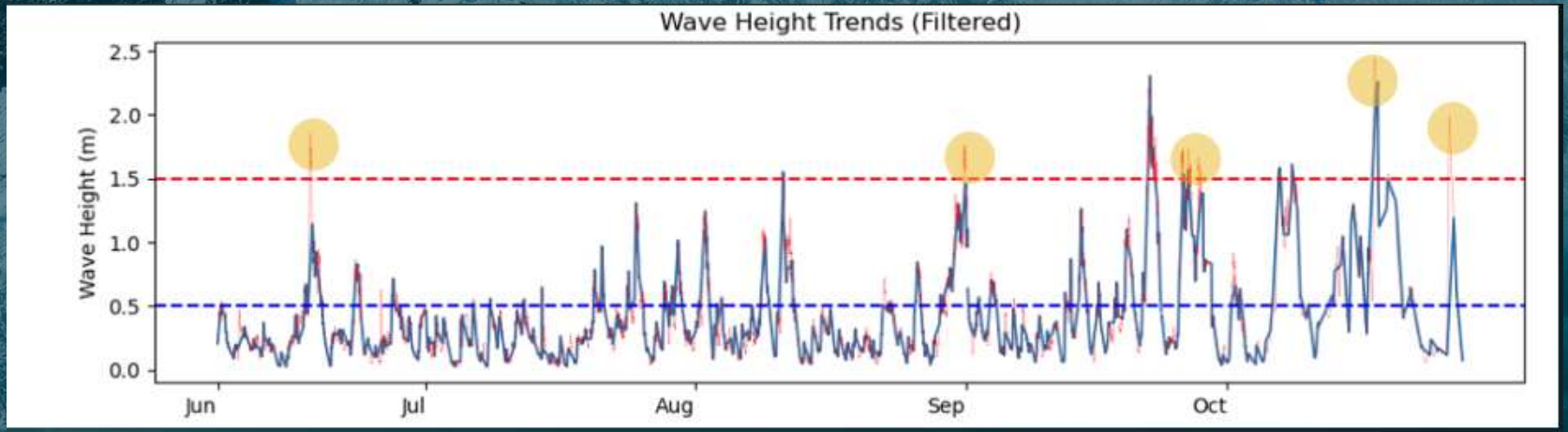
# Maximum Wave heights: 8:00AM – 8:00PM

```
In [141]: buoy_data['time'] = pd.to_datetime(buoy_data['time (UTC)'])

          buoy_data.set_index('time', inplace=True)

          filtered_data = buoy_data.between_time('08:00:00', '20:00:00')
          filtered_data.reset_index(inplace=True)
```

```
In [142]: filtered_data.info()  #From 8000 rows we went to 4000

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 4207 entries, 0 to 4206
          Data columns (total 41 columns):
           #   Column                                Non-Null Count  Dtype
          ---  ------                                --------------  -----
           0   time                                  4207 non-null   datetime64[ns, UTC]
           1   time (UTC)                            4207 non-null   object
           2   air_pressure_at_mean_sea_level (Pa)   4207 non-null   float64
```



# Wave Data In terms of High Wave Days




```
In [172]: count_wave_height = (buoy_data['sea_surface_wave_significant_height (m)'] > 1.5).sum()
          print(f"Wave height > 1.5m count: {count_wave_height}")

          Wave height > 1.5m count: 171

In [173]: count_wave_height = (filtered_data['sea_surface_wave_significant_height (m)'] > 1.5).sum()
          print(f"Daytime Wave height > 1.5m count: {count_wave_height}")

          Daytime Wave height > 1.5m count: 53
```

```
['Saturday, 2022-06-18' 'Thursday, 2022-08-11' 'Thursday, 2022-09-01'
 'Thursday, 2022-09-22' 'Friday, 2022-09-23' 'Monday, 2022-09-26'
 'Tuesday, 2022-09-27' 'Wednesday, 2022-09-28' 'Friday, 2022-10-07'
 'Saturday, 2022-10-08' 'Tuesday, 2022-10-18' 'Wednesday, 2022-10-19'
 'Wednesday, 2022-10-26']
```

More than a 100 High wave occurrences were during night times and 53 were during daytime. In terms of total High wave days, we had 6/13 days with High waves during daytime

# Daytime High Wave Days

```
['Thursday, 2022-08-11' 'Thursday, 2022-09-22' 'Monday, 2022-09-26'
 'Friday, 2022-10-07' 'Saturday, 2022-10-08' 'Tuesday, 2022-10-18']
```

| | time | weekday | waveheight |
|---|---|---|---|
| 2276 | 2022-08-11 08:20:00+00:00 | Thursday | 1.513 |
| 2277 | 2022-08-11 08:40:00+00:00 | Thursday | 1.551 |
| 2278 | 2022-08-11 09:00:00+00:00 | Thursday | 1.503 |
| 3717 | 2022-09-22 08:00:00+00:00 | Thursday | 1.722 |
| 3718 | 2022-09-22 08:20:00+00:00 | Thursday | 1.733 |
| 3719 | 2022-09-22 08:40:00+00:00 | Thursday | 1.843 |
| 3720 | 2022-09-22 09:00:00+00:00 | Thursday | 1.843 |
| 3721 | 2022-09-22 09:20:00+00:00 | Thursday | 2.028 |
| 3722 | 2022-09-22 09:40:00+00:00 | Thursday | 2.027 |
| 3723 | 2022-09-22 10:00:00+00:00 | Thursday | 2.024 |
| 3724 | 2022-09-22 10:20:00+00:00 | Thursday | 2.076 |
| 3725 | 2022-09-22 10:40:00+00:00 | Thursday | 2.208 |
| 3726 | 2022-09-22 11:00:00+00:00 | Thursday | 2.175 |
| 3727 | 2022-09-22 11:20:00+00:00 | Thursday | 2.129 |
| 3728 | 2022-09-22 11:40:00+00:00 | Thursday | 2.305 |
| 3729 | 2022-09-22 12:00:00+00:00 | Thursday | 2.158 |
| 3730 | 2022-09-22 12:20:00+00:00 | Thursday | 2.080 |
| 3731 | 2022-09-22 12:40:00+00:00 | Thursday | 2.084 |
| 3732 | 2022-09-22 13:00:00+00:00 | Thursday | 2.084 |
| 3733 | 2022-09-22 13:20:00+00:00 | Thursday | 1.752 |
| 3734 | 2022-09-22 13:40:00+00:00 | Thursday | 1.793 |
| 3735 | 2022-09-22 14:00:00+00:00 | Thursday | 1.928 |
| 3736 | 2022-09-22 14:20:00+00:00 | Thursday | 1.904 |
| 3737 | 2022-09-22 14:40:00+00:00 | Thursday | 1.969 |
| 3738 | 2022-09-22 15:00:00+00:00 | Thursday | 1.848 |
| 3739 | 2022-09-22 15:20:00+00:00 | Thursday | 1.752 |
| 3740 | 2022-09-22 15:40:00+00:00 | Thursday | 1.773 |
| 3741 | 2022-09-22 16:00:00+00:00 | Thursday | 1.807 |
| 3742 | 2022-09-22 16:20:00+00:00 | Thursday | 1.769 |
| 3743 | 2022-09-22 16:40:00+00:00 | Thursday | 1.986 |
| 3744 | 2022-09-22 17:00:00+00:00 | Thursday | 1.972 |
| 3745 | 2022-09-22 17:20:00+00:00 | Thursday | 1.630 |
| 3746 | 2022-09-22 17:40:00+00:00 | Thursday | 1.730 |
| 3747 | 2022-09-22 18:00:00+00:00 | Thursday | 1.717 |
| 3748 | 2022-09-22 18:20:00+00:00 | Thursday | 1.752 |
| 3749 | 2022-09-22 18:40:00+00:00 | Thursday | 1.592 |
| 3750 | 2022-09-22 19:00:00+00:00 | Thursday | 1.736 |
| 3751 | 2022-09-22 19:20:00+00:00 | Thursday | 1.716 |
| 3752 | 2022-09-22 19:40:00+00:00 | Thursday | 1.645 |
| 3753 | 2022-09-22 20:00:00+00:00 | Thursday | 1.651 |
| 3865 | 2022-09-26 08:00:00+00:00 | Monday | 1.531 |
| 3866 | 2022-09-26 08:20:00+00:00 | Monday | 1.536 |
| 3867 | 2022-09-26 08:40:00+00:00 | Monday | 1.511 |
| 3898 | 2022-09-26 19:20:00+00:00 | Monday | 1.540 |
| 3899 | 2022-09-26 19:40:00+00:00 | Monday | 1.540 |
| 3900 | 2022-09-26 20:00:00+00:00 | Monday | 1.572 |
| 4106 | 2022-10-07 08:00:00+00:00 | Friday | 1.585 |
| 4107 | 2022-10-07 10:00:00+00:00 | Friday | 1.543 |
| 4115 | 2022-10-08 19:00:00+00:00 | Saturday | 1.611 |
| 4160 | 2022-10-18 10:20:00+00:00 | Tuesday | 2.145 |
| 4161 | 2022-10-18 15:20:00+00:00 | Tuesday | 2.260 |
| 4162 | 2022-10-18 15:40:00+00:00 | Tuesday | 2.005 |
| 4163 | 2022-10-18 16:00:00+00:00 | Tuesday | 1.591 |

August 8, Thursday
- 8AM – 9AM

September 22, Thursday
- 8AM – 8PM

September 26, Monday
- 8AM – 9AM
- 7PM – 8PM

October 7, Friday
- 8AM, 10AM

October 8, Saturday
- 7PM

October 18, Tuesday
- 10AM
- 3PM – 4PM

# Why is Data Consistency Needed?

## High Wave Days:

August 8, Thursday(3)
- 8AM – 9AM

September 22, Thursday(37)
- 8AM – 8PM

September 26, Monday(6)
- 8AM – 9AM
- 7PM – 8PM

October 7, Friday(2)
- 8AM, 10AM

October 8, Saturday(1)
- 7PM

October 18, Tuesday(4)
- 10AM
- 3PM – 4PM

## People Count Data:

August 8, Thursday(0)
- No data

September 22, Thursday(2)
- 1:23PM, 6:54PM

September 26, Monday(0)
- No data

October 7, Friday(3)
- 9:10AM, 10:10AM, 10:37AM

October 8, Saturday(3)
- 12:30PM, 2:10PM, 3:10PM

October 18, Tuesday(6)
- 8:20AM, 11:20AM, 12:20PM,
- 1:30PM, 2:10PM, 3:40PM

# Machine Learning Challenges:

Determining Maximum Number of People on any given day for predicting based on weather variables:

- Summation of all counts per day is not valid as the frequency of readings is not consistent and may provide misleading and skewed information.

- Average of all counts per day is not valid for the same reason because the time when readings were taken is not consistent, some days have readings taken at unusual times like early morning or late night when no one is at the beach while some days have more readings during day times.

- Max count among all readings for a given date would result in a similar problem as some dates/times which could potentially have higher traffic due to favorable weather might not have been recorded.

# Proposed Solution:

We can take max count for each day; however, we will study the frequency of reading times for each day and limit the selection to a timeframe that had a significant number of readings taken every day within that frame.

Note: The predicted data may or may not provide correct results due to limitations from the input data. Although we will create a machine learning model for the scope of this project even if more data is not available, retraining the model with a better dataset is highly advised.

# Selecting Timeframes with most records

```
In [23]: PeopleCount2022['date'] = PeopleCount2022['DateTime'].dt.date
         PeopleCount2022['month'] = PeopleCount2022['DateTime'].dt.month
         PeopleCount2022['hour'] = PeopleCount2022['DateTime'].dt.hour
         PeopleCount2022
```

Out[23]:

| | File_name | Year | Total People | People in Water | People on Sand | DateTime | date | month | hour |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2022-08-25_15-35-39_jpg.rf.f29c550c1dfa9e50ded... | 2022 | 14 | 1 | 13 | 2022-08-25 15:35:39 | 2022-08-25 | 8 | 15 |
| 1 | 2022-08-25_15-35-40_jpg.rf.5bfb5ece1d5e39664fd... | 2022 | 16 | 2 | 14 | 2022-08-25 15:35:40 | 2022-08-25 | 8 | 15 |
| 2 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 | 2022-08-25 15:38:19 | 2022-08-25 | 8 | 15 |
| 3 | 2022-08-25_15-40-27_jpg.rf.ceb8e5145f8ec9240ed... | 2022 | 15 | 3 | 12 | 2022-08-25 15:40:27 | 2022-08-25 | 8 | 15 |
| 4 | 2022-08-25_15-41-33_jpg.rf.5d69e86774bd8db8810... | 2022 | 18 | 3 | 15 | 2022-08-25 15:41:33 | 2022-08-25 | 8 | 15 |
| ... | ... | ... | ... | ... | ... | ... | | | |
| 215 | 2022-10-22_15-43-52_jpg.rf.f0f727fb618ad2bf751... | 2022 | 2 | 0 | 2 | 2022-10-22 1 | | | |
| 216 | 2022-10-23_17-20-16_jpg.rf.c9fbc83271b930be8bd... | 2022 | 4 | 0 | 4 | 2022-10-23 1 | | | |
| 217 | 2022-10-24_17-30-16_jpg.rf.359aea533c623c69372... | 2022 | 7 | 0 | 7 | 2022-10-24 1 | | | |
| 218 | 2022-10-25_18-33-52_jpg.rf.1ee485d7e774f4d046c... | 2022 | 2 | 0 | 2 | 2022-10-25 1 | | | |
| 219 | 2022-10-27_08-50-16_jpg.rf.b579b95f7074c1aa942... | 2022 | 2 | 0 | 2 | 2022-10-27 0 | | | |

220 rows × 9 columns

(a) Extracting hours from Datetime to check hourly records

```
In [30]: PeopleCount2022['hour'].value_counts().sort_index()
```

Out[30]:

```
6     2
7    12
8    10
9     3
10    9
11   16
12   15
13   20
14   25
15   34
16   27
17   28
18   19
Name: hour, dtype: int64
```

```
In [28]: hourly_entries_per_day = PeopleCount2022.groupby(['date', 'hour']).size().reset_index(name='Count')
         print(hourly_entries_per_day)
```

```
          date  hour  Count
0   2022-08-25    15      7
1   2022-08-25    16      6
2   2022-08-25    17      2
3   2022-08-26    11      2
4   2022-08-26    13      1
..         ...   ...    ...
128 2022-10-22    15      1
129 2022-10-23    17      1
130 2022-10-24    17      1
131 2022-10-25    18      1
132 2022-10-27     8      1

[133 rows x 3 columns]
```

(b) Grouping Data based on hours and checking for duplicates

# Hourly records without duplicates and filtering the timeframe with most records

In [34]:
```
total_distinct_hourly_counts = PeopleCount2022.groupby('hour')['date'].nunique().reset_index(name='Count')
print(total_distinct_hourly_counts)
```

```
    hour  Count
0      6      2
1      7     10
2      8     10
3      9      3
4     10      8
5     11      9
6     12     10
7     13     12
8     14     14
9     15     17
10    16     12
11    17     12
12    18     14
```

In [35]:
```
filtered_data = PeopleCount2022[(PeopleCount2022['hour'] >= 12) & (PeopleCount2022['hour'] <= 17)]
```

Out[53]:

| | DateTime | File_name | Year | Total People | People in Water | People on Sand | month | hour |
|---|---|---|---|---|---|---|---|---|
| **0** | 2022-08-25 15:35:39 | 2022-08-25_15-35-39_jpg.rf.f29c550c1dfa9e50ded... | 2022 | 14 | 1 | 13 | 8 | 15 |
| **1** | 2022-08-25 15:35:40 | 2022-08-25_15-35-40_jpg.rf.5bfb5ece1d5e39664fd... | 2022 | 16 | 2 | 14 | 8 | 15 |
| **2** | 2022-08-25 15:38:19 | 2022-08-25_15-38-19_jpg.rf.592c8ceddf472ad8562... | 2022 | 18 | 5 | 13 | 8 | 15 |
| **3** | 2022-08-25 15:40:27 | 2022-08-25_15-40-27_jpg.rf.ceb8e5145f8ec9240ed... | 2022 | 15 | 3 | 12 | 8 | 15 |
| **4** | 2022-08-25 15:41:33 | 2022-08-25_15-41-33_jpg.rf.5d69e86774bd8db8810... | 2022 | 18 | 3 | 15 | 8 | 15 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **144** | 2022-10-18 15:40:15 | 2022-10-18_15-40-15_jpg.rf.d3ff33ba3fe8db1d542... | 2022 | 5 | 2 | 3 | 10 | 15 |
| **145** | 2022-10-21 13:30:16 | 2022-10-21_13-30-16_jpg.rf.fc5f4afa75e1c39ffcd... | 2022 | 2 | 0 | 2 | 10 | 13 |
| **146** | 2022-10-22 15:43:52 | 2022-10-22_15-43-52_jpg.rf.f0f727fb618ad2bf751... | 2022 | 2 | 0 | 2 | 10 | 15 |
| **147** | 2022-10-23 17:20:16 | 2022-10-23_17-20-16_jpg.rf.c9fbc83271b930be8bd... | 2022 | 4 | 0 | 4 | 10 | 17 |
| **148** | 2022-10-24 17:30:16 | 2022-10-24_17-30-16_jpg.rf.359aea533c623c69372... | 2022 | 7 | 0 | 7 | 10 | 17 |

149 rows × 8 columns

# Taking Max People count for each unique day within the 12PM – 5PM timeframe

In [61]:
```
MaxPeopleCount = filtered_data.groupby(filtered_data['DateTime'].dt.date)['Total People'].max().reset_index()
MaxPeopleCount.columns = ['Date', 'Daily Max People']
MaxPeopleCount
```

Out[61]:

| | Date | Max Total People | Month | Day | Day_Name |
|---|---|---|---|---|---|
| 0 | 2022-08-25 | 22 | 8 | 25 | Thursday |
| 1 | 2022-08-26 | 25 | 8 | 26 | Friday |
| 2 | 2022-08-27 | 32 | 8 | 27 | Saturday |
| 3 | 2022-08-28 | 29 | 8 | 28 | Sunday |
| 4 | 2022-08-29 | 28 | 8 | 29 | Monday |
| 5 | 2022-09-02 | 23 | 9 | 2 | Friday |
| 6 | 2022-09-04 | 21 | 9 | 4 | Sunday |
| 7 | 2022-09-05 | 19 | 9 | 5 | Monday |
| 8 | 2022-09-06 | 13 | 9 | 6 | Tuesday |
| 9 | 2022-09-07 | 16 | 9 | 7 | Wednesday |
| 10 | 2022-09-09 | 1 | 9 | 9 | Friday |
| 11 | 2022-09-10 | 15 | 9 | 10 | Saturday |
| 12 | 2022-09-11 | 14 | 9 | 11 | Sunday |
| 13 | 2022-09-12 | 2 | 9 | 12 | Monday |
| 14 | 2022-09-13 | 3 | 9 | 13 | Tuesday |
| 15 | 2022-09-14 | 10 | 9 | 14 | Wednesday |
| 16 | 2022-09-15 | 3 | 9 | 15 | Thursday |
| 17 | 2022-09-17 | 21 | 9 | 17 | Saturday |
| 18 | 2022-09-19 | 10 | 9 | 19 | Monday |
| 19 | 2022-09-22 | 7 | 9 | 22 | Thursday |
| 20 | 2022-09-25 | 0 | 9 | 25 | Sunday |
| 21 | 2022-09-27 | 0 | 9 | 27 | Tuesday |
| 22 | 2022-09-28 | 0 | 9 | 28 | Wednesday |
| 23 | 2022-09-29 | 2 | 9 | 29 | Thursday |
| 24 | 2022-09-30 | 3 | 9 | 30 | Friday |
| 25 | 2022-10-01 | 4 | 10 | 1 | Saturday |
| 26 | 2022-10-08 | 7 | 10 | 8 | Saturday |
| 27 | 2022-10-09 | 7 | 10 | 9 | Sunday |
| 28 | 2022-10-12 | 0 | 10 | 12 | Wednesday |
| 29 | 2022-10-13 | 3 | 10 | 13 | Thursday |
| 30 | 2022-10-14 | 4 | 10 | 14 | Friday |
| 31 | 2022-10-16 | 3 | 10 | 16 | Sunday |
| 32 | 2022-10-18 | 5 | 10 | 18 | Tuesday |
| 33 | 2022-10-21 | 2 | 10 | 21 | Friday |
| 34 | 2022-10-22 | 2 | 10 | 22 | Saturday |
| 35 | 2022-10-23 | 4 | 10 | 23 | Sunday |
| 36 | 2022-10-24 | 7 | 10 | 24 | Monday |

# Selecting Timeframes with most records

Checking the hourly data within the timeframe on daily level
– First figure shows the record with the maximum people for any given hour
– Second figure shows the number of records for any given hour

| Max People | Dates | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Hours | 2022-08-25 | 2022-08-26 | 2022-08-27 | 2022-08-28 | 2022-08-29 | 2022-09-02 | 2022-09-04 | 2022-09-05 | 2022-09-06 |
| 12.00 | | | 24 | 26 | | | | | |
| 13.00 | | 15 | 29 | 12 | 28 | 17 | | | |
| 14.00 | | 22 | 29 | 21 | | | 4 | | |
| 15.00 | 20 | 25 | 32 | 17 | 8 | 23 | 21 | | 13 |
| 16.00 | 22 | 22 | 29 | 29 | | 6 | | | |
| 17.00 | 16 | 19 | 28 | 10 | 22 | | | 19 | |
| **Max Value** | **22** | **25** | **32** | **29** | **28** | **23** | **21** | **19** | **13** |

| Max People Count | Dates | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Hours | 2022-08-25 | 2022-08-26 | 2022-08-27 | 2022-08-28 | 2022-08-29 | 2022-09-02 | 2022-09-04 | 2022-09-05 | 2022-09-06 |
| 12.00 | | | 6 | 1 | | | | | |
| 13.00 | | 1 | 8 | 1 | 1 | 1 | | | |
| 14.00 | | 5 | 8 | 1 | | | 1 | | |
| 15.00 | 7 | 5 | 5 | 2 | 1 | 2 | 1 | | 1 |
| 16.00 | 6 | 4 | 3 | 4 | | 1 | | | |
| 17.00 | 2 | 6 | 8 | 1 | 3 | | | 1 | |
| **Grand Total** | **15** | **21** | **38** | **10** | **5** | **4** | **2** | **1** | **1** |

# Selecting Timeframes with most records

Merging the selected People Count data to other weather and wave variables

| | Date | Max Total People | Month | Day | Day_Name | tempmax | tempmin | temp | feelslikemax | feelslikemin | ... | winddir | cloudcover | visibility | uvindex | conditions | maxwaveheight | minwaveheight | averagewaveheight | month_name | windir_cat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2022-08-25 | 22 | 8 | 25 | Thursday | 24.3 | 17.3 | 20.8 | 24.3 | 17.3 | ... | 132.9 | 63.5 | 24.0 | 7 | rain | 0.271 | 0.052 | 0.145986 | Aug | E-SE |
| 1 | 2022-08-26 | 25 | 8 | 26 | Friday | 21.8 | 12.7 | 19.1 | 21.8 | 12.7 | ... | 347.5 | 75.3 | 20.3 | 5 | rain | 0.854 | 0.121 | 0.423131 | Aug | NW-N |
| 2 | 2022-08-27 | 32 | 8 | 27 | Saturday | 23.3 | 9.2 | 16.5 | 23.3 | 8.9 | ... | 122.3 | 9.6 | 24.9 | 9 | clear-day | 0.797 | 0.102 | 0.349016 | Aug | E-SE |
| 3 | 2022-08-28 | 29 | 8 | 28 | Sunday | 27.8 | 13.8 | 20.9 | 28.4 | 13.8 | ... | 174.9 | 19.3 | 24.0 | 9 | clear-day | 0.264 | 0.030 | 0.121794 | Aug | SE-S |
| 4 | 2022-08-29 | 28 | 8 | 29 | Monday | 26.3 | 21.1 | 23.6 | 26.3 | 21.1 | ... | 191.3 | 74.6 | 24.0 | 6 | rain | 0.594 | 0.105 | 0.401742 | Aug | S-SW |
| 5 | 2022-09-02 | 23 | 9 | 2 | Friday | 26.9 | 17.2 | 21.8 | 27.4 | 17.2 | ... | 185.3 | 29.4 | 24.0 | 9 | partly-cloudy-day | 0.567 | 0.121 | 0.337455 | Sep | S-SW |
| 6 | 2022-09-04 | 21 | 9 | 4 | Sunday | 18.9 | 15.8 | 16.5 | 18.9 | 15.8 | ... | 57.6 | 74.5 | 24.0 | 1 | rain | 0.768 | 0.336 | 0.577739 | Sep | NE-E |
| 7 | 2022-09-05 | 19 | 9 | 5 | Monday | 21.7 | 15.4 | 17.8 | 21.7 | 15.4 | ... | 63.7 | 69.0 | 24.0 | 8 | partly-cloudy-day | 0.456 | 0.114 | 0.215522 | Sep | NE-E |
| 8 | 2022-09-06 | 13 | 9 | 6 | Tuesday | 21.6 | 12.7 | 17.4 | 21.6 | 12.7 | ... | 34.8 | 21.8 | 24.0 | 9 | partly-cloudy-day | 0.558 | 0.158 | 0.291101 | Sep | N-NE |
| 9 | 2022-09-07 | 16 | 9 | 7 | Wednesday | 22.4 | 9.9 | 16.1 | 22.4 | 9.5 | ... | 9.6 | 15.8 | 24.0 | 9 | clear-day | 0.492 | 0.092 | 0.224268 | Sep | N-NE |
| 10 | 2022-09-09 | 1 | 9 | 9 | Friday | 24.5 | 12.6 | 19.0 | 24.5 | 12.6 | ... | 183.2 | 13.0 | 24.0 | 9 | clear-day | 0.423 | 0.052 | 0.199431 | Sep | S-SW |
| 30 | 2022-10-14 | 4 | 10 | 14 | Friday | 11.9 | 5.9 | 8.5 | 11.9 | 2.3 | ... | 194.7 | 44.6 | 23.0 | 7 | rain | 1.046 | 0.577 | 0.814500 | Oct | S-SW |
| 31 | 2022-10-16 | 3 | 10 | 16 | Sunday | 10.8 | 6.7 | 8.7 | 10.8 | 4.0 | ... | 230.8 | 40.5 | 22.5 | 6 | rain | 1.160 | 0.558 | 0.809615 | Oct | SW-W |
| 32 | 2022-10-18 | 5 | 10 | 18 | Tuesday | 6.0 | 2.1 | 4.6 | 2.3 | -0.8 | ... | 286.6 | 83.1 | 22.6 | 3 | rain | 2.447 | 0.534 | 1.655364 | Oct | W-NW |
| 33 | 2022-10-21 | 2 | 10 | 21 | Friday | 15.5 | 2.0 | 9.1 | 15.5 | -1.6 | ... | 180.8 | 30.5 | 23.0 | 6 | rain | 0.497 | 0.405 | 0.436667 | Oct | S-SW |
| 34 | 2022-10-22 | 2 | 10 | 22 | Saturday | 21.1 | 11.9 | 15.8 | 21.1 | 11.9 | ... | 184.6 | 28.8 | 24.0 | 7 | partly-cloudy-day | 0.656 | 0.350 | 0.548857 | Oct | S-SW |
| 35 | 2022-10-23 | 4 | 10 | 23 | Sunday | 23.2 | 12.2 | 16.9 | 23.2 | 12.2 | ... | 153.8 | 18.2 | 24.0 | 6 | clear-day | 0.296 | 0.122 | 0.206455 | Oct | SE-S |
| 36 | 2022-10-24 | 7 | 10 | 24 | Monday | 22.6 | 10.8 | 16.0 | 22.6 | 10.8 | ... | 147.8 | 36.1 | 24.0 | 6 | partly-cloudy-day | 0.237 | 0.053 | 0.146818 | Oct | SE-S |

37 rows × 28 columns

# Check for correlations and Patterns

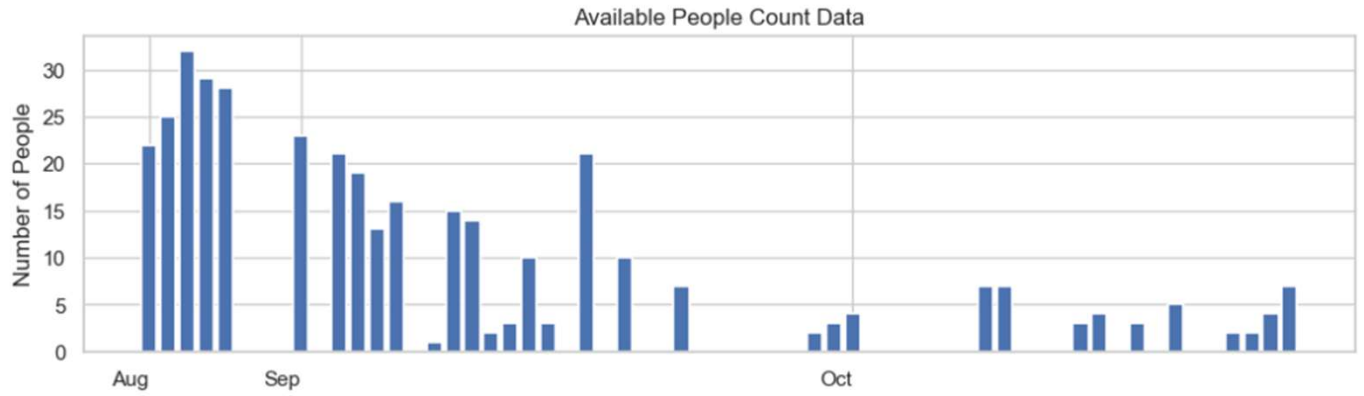
Checking records available based on weather conditions and monthly outlook.

In [72]:
```
allVals.groupby('conditions')[['Max Total People']].agg(['count','mean','median','min','max'])
```

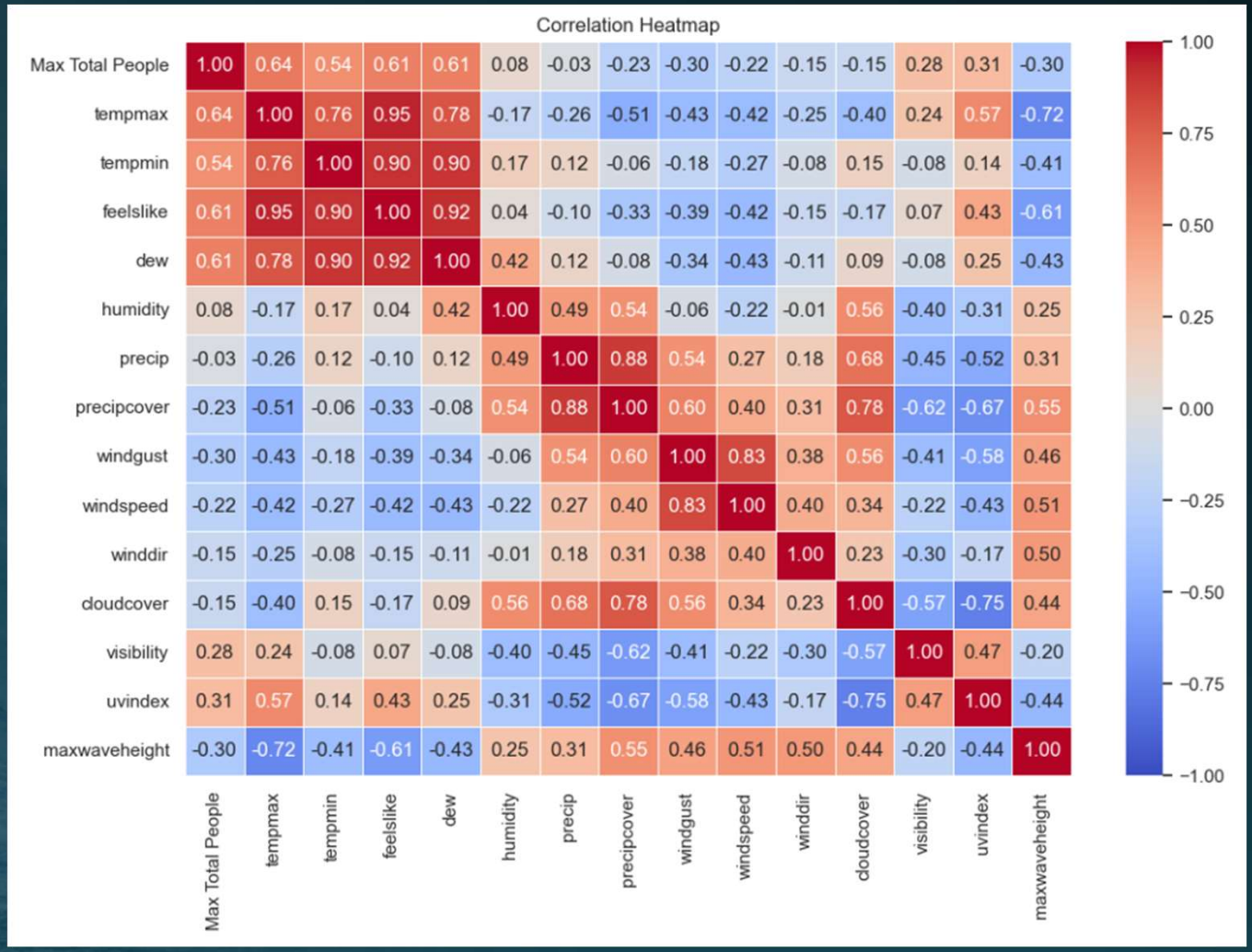
Out[72]:

| | Max Total People | | | | |
|---|---|---|---|---|---|
| | count | mean | median | min | max |
| conditions | | | | | |
| clear-day | 6 | 14.166667 | 10.0 | 1 | 32 |
| partly-cloudy-day | 12 | 9.000000 | 7.0 | 2 | 23 |
| rain | 19 | 9.157895 | 5.0 | 0 | 28 |

In [91]:
```
plt.figure(figsize=(12, 3))

plt.bar(allVals['Date'], allVals['Max Total People'], linewidth=1.2)

plt.title('Available People Count Data')
plt.ylabel('Number of People')
month_index = allVals.groupby('Month').head(1).index
plt.xticks(allVals.loc[month_index, 'Date'], allVals.loc[month_index, 'month_name'], rotation=0, ha='right')

plt.show()
```



# Check for correlations and Patterns




People Count Correlation Heatmap

# Check for correlations and Patterns

Scatter Plot for Max People vs Max temperature.

```
In [100]: plt.figure(figsize=(6, 3))
          sns.scatterplot(x='tempmax', y='Max Total People', data=allVals)
          plt.title('Number of People vs Max temp')
          plt.show()
```