

# Welltory COVID-19 and Wearables Dataset Exploratory Data Analysis

Importing necessary modules

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

## Loading Dataset

First we will read the dataset into memory. All dataframes are stored inside a dictionary "dfs".

Link to dataset: <https://github.com/Welltory/hrv-covid19>

```
1 FILENAMES = ['participants', 'blood_pressure', 'heart_rate', 'hrv_measurements', 's
2 URL = 'https://raw.githubusercontent.com/Welltory/hrv-covid19/master/data/'
3 EXTENSION = '.csv'
4
5 dfs = {}
6
7 for fn in FILENAMES:
8     dfs[fn] = pd.read_csv(URL + fn + EXTENSION)
```

## Participants

---

Let's start with the participants data. The participant data consists of the user unique ID, the gender, age range, city, country, height, weight, and the date when symptoms start to show (if available).

```
1 print(dfs['participants'].head())
```

	user_code	gender	age_range	city	country	height	weight	\
0	007b8190cf	m	25-34	Mandalay	Myanmar	170.18	96.162	
1	013f6d3e5b	f	18-24	São Paulo	Brazil	174.00	77.300	
2	01bad5a519	m	45-54	St Petersburg	Russia	178.00	92.000	
3	0210b20eea	f	25-34	Sochi	Russia	169.00	60.000	
4	024719e7da	f	45-54	St Petersburg	Russia	158.00	68.500	

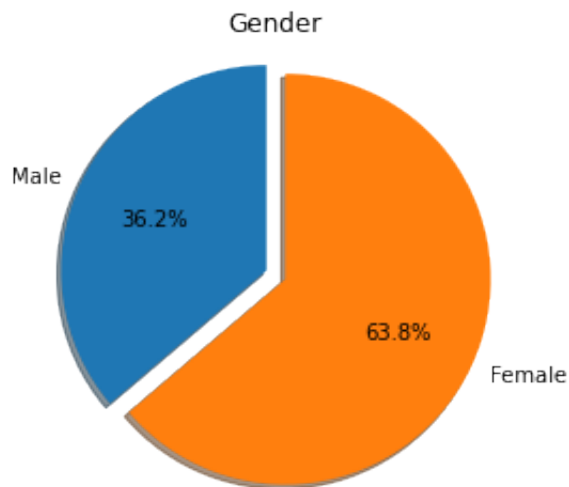
  

	symptoms_onset
0	NaN
1	5/15/2020
2	4/5/2020
3	5/6/2020
4	5/27/2020

Then, we will look at the participants demographics distribution.

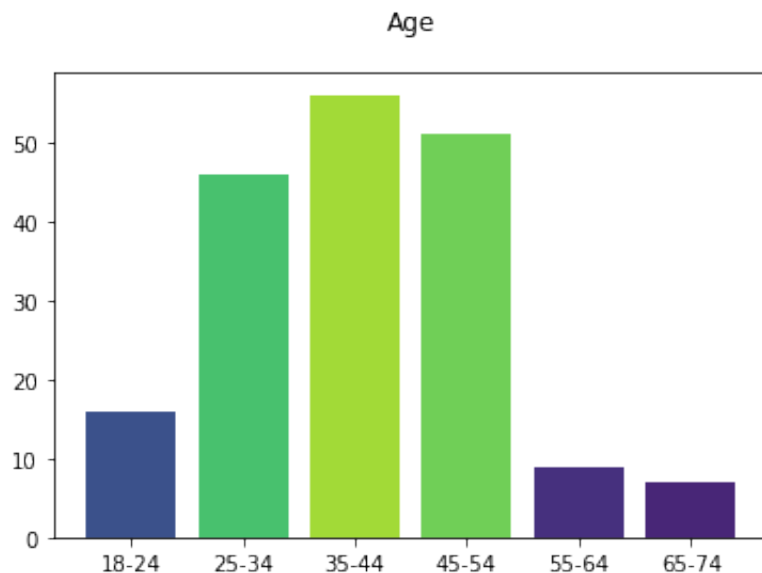
## ▼ Gender

```
1 # Draw pie chart (gender)
2 statGender = dfs['participants']['gender'].value_counts()
3 total = len(dfs['participants']['gender'])
4 figGender, axGender = plt.subplots()
5 labels = ['Male', 'Female']
6 sizes = [statGender['m'] / total, statGender['f'] / total]
7 explode = (0.1, 0)
8 axGender.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True)
9 axGender.axis('equal')
10 axGender.set_title('Gender')
11 plt.show()
```



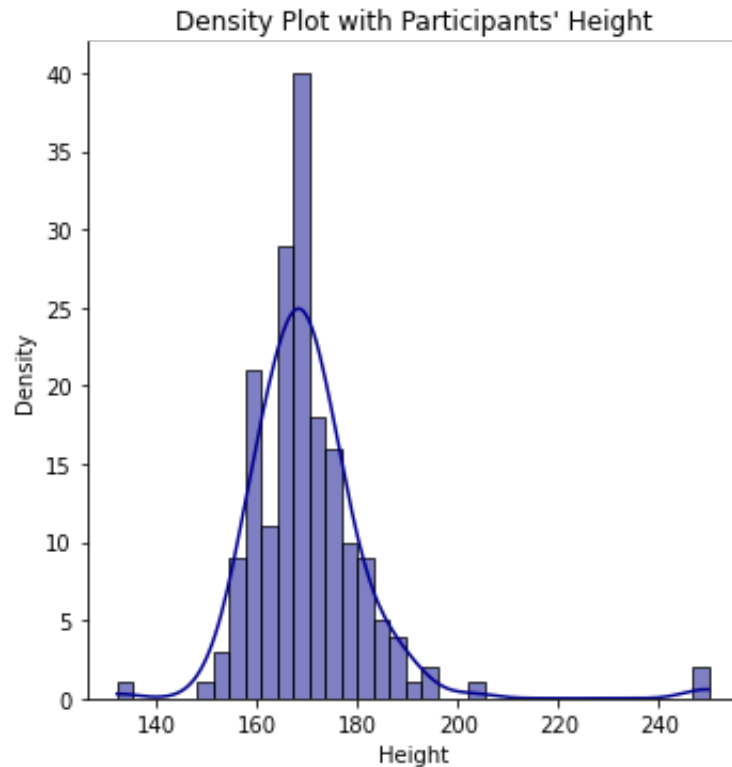
## ▼ Age

```
1 # Age
2 statAge = dfs['participants']['age_range'].value_counts()
3 figAge, axAge = plt.subplots()
4
5 # Extract data from dataframe
6 labels = ['18-24', '25-34', '35-44', '45-54', '55-64', '65-74']
7 values = [statAge['18-24'], statAge['25-34'], statAge['35-44'], statAge['45-54'],
8           statAge['55-64'], statAge['65-74']]
9 # set colormap
10 cmap = plt.get_cmap("viridis")
11 norm = plt.Normalize(vmin=0, vmax=65)
12
13 # plot bar chart
14 axAge.bar(labels, values, color=cmap(norm(values)))
15 axAge.set_title('Age', pad=20)
16 plt.show()
```



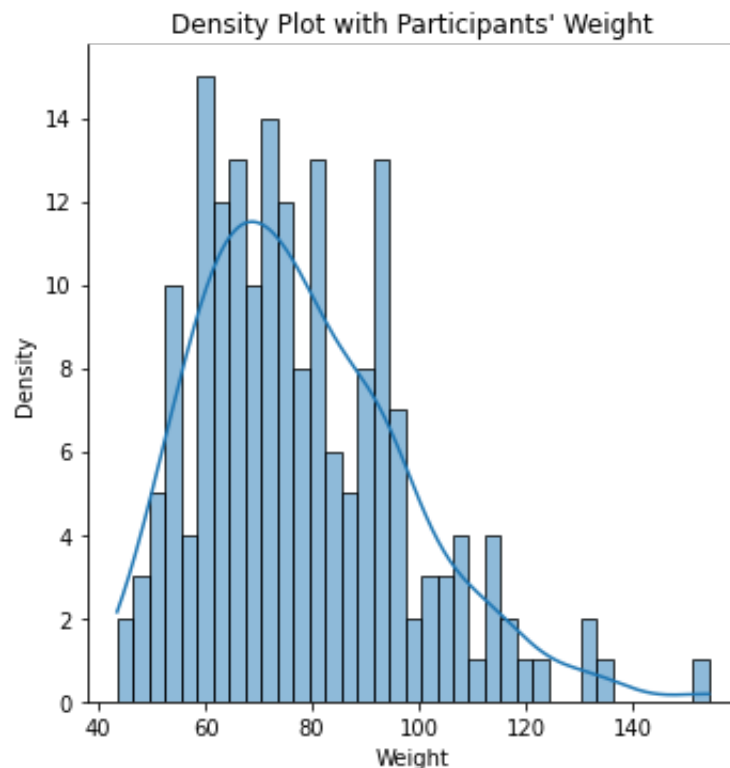
## ▼ Height

```
1 # Height
2 sns.displot(dfs['participants']['height'], kde=True,
3             bins=int(len(dfs['participants'])/5), color = 'darkblue')
4 plt.title('Density Plot with Participants\' Height')
5 plt.xlabel('Height')
6 plt.ylabel('Density')
7 plt.show()
```



## ▼ Weight

```
1 # Weight
2 sns.displot(dfs['participants']['weight'], kde=True,
3             bins=int(len(dfs['participants'])/5))
4 plt.title('Density Plot with Participants\' Weight')
5 plt.xlabel('Weight')
6 plt.ylabel('Density')
7 plt.show()
```



## ▼ Blood Pressure Data

---

Next, we will look at the blood pressure data. It consists of the date and time of the measurement taken by a particular participant, and the corresponding diastolic and systolic reading. Some measurements also include indices such as the functional changes index, circulatory efficiency, kerdo vegetation index, and robinson index.

```
1 print(dfs['blood_pressure'].head())
```

	user_code	measurement_datetime	diastolic	systolic	\
0	01bad5a519	2020-04-29 22:33:33	100	150	
1	01bad5a519	2020-04-30 01:33:33	100	150	
2	01bad5a519	2020-04-30 09:16:38	95	140	
3	01bad5a519	2020-04-30 12:16:38	95	140	
4	01bad5a519	2020-05-01 06:58:06	80	130	

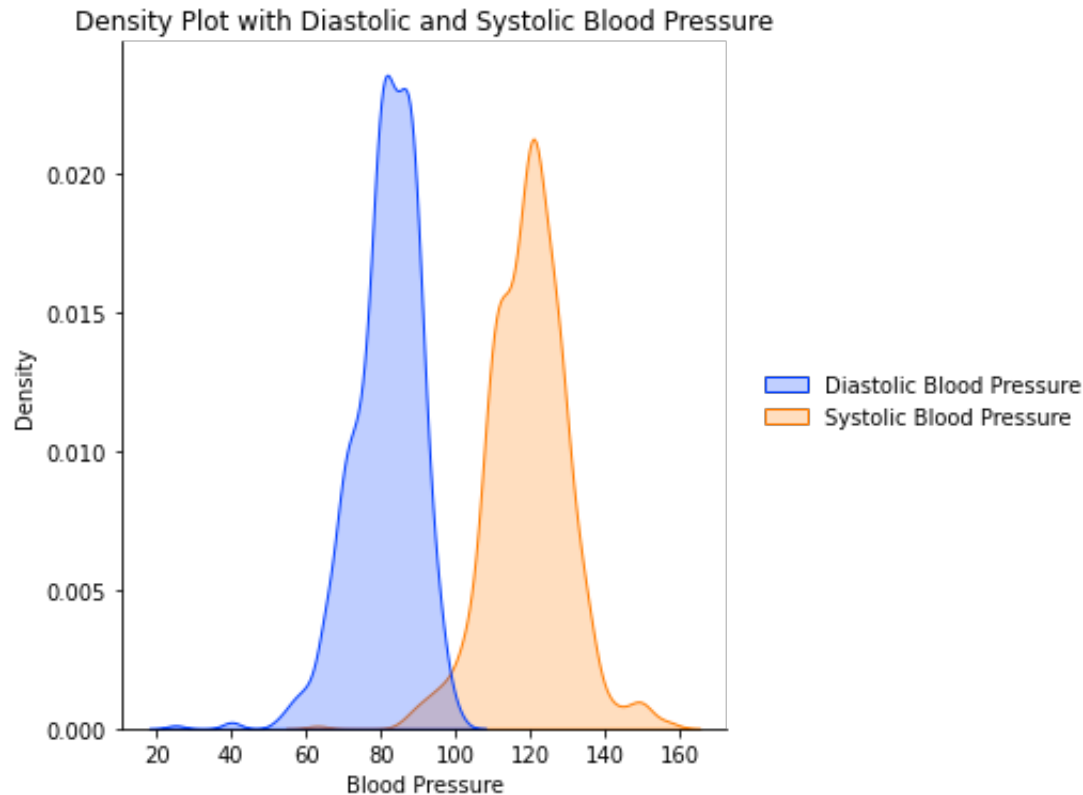
	functional_changes_index	circulatory_efficiency	kerdo_vegetation_index
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	3.38	4545.0	6.0
3	NaN	NaN	NaN
4	2.89	4000.0	NaN

	robinson_index
0	NaN
1	NaN
2	141.4
3	NaN
4	104.0

Here, we plot the distribution of the Diastolic Blood Pressure and the Systolic Blood Pressure.

```
1 # Blood Pressure
2 data = pd.DataFrame(data=np.c_[dfs['blood_pressure']['diastolic'], dfs['blood_p
3 sns.displot(data=data, kind='kde', palette=sns.color_palette('bright')[:2], fil
4 plt.title('Density Plot with Diastolic and Systolic Blood Pressure')
5 plt.xlabel('Blood Pressure')
6 plt.ylabel('Density')
7 plt.show()
```



## ▼ Heart Rate Data

In the heart rate data, it has the heart rate of each participant at a specific date and time, and an indicator of whether the participant is at rest.

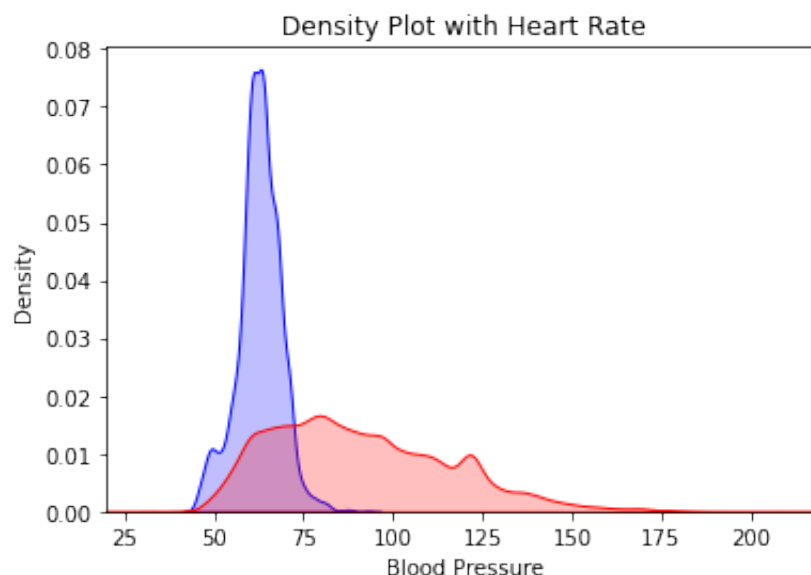


```
1 print(dfs['heart_rate'].head())
```

	user_code	datetime	heart_rate	is_resting
0	007b8190cf	2020-04-26 04:49:25	70	0
1	01bad5a519	2020-04-23 06:21:03	74	0
2	01bad5a519	2020-04-23 09:46:01	82	0
3	01bad5a519	2020-04-23 14:05:06	90	0
4	01bad5a519	2020-04-24 03:41:18	72	0

We can observe that the heart rate is significantly higher when the user is not resting.

```
1 # Extract Heart Rate for both is_resting states
2 df = dfs['heart_rate']
3 resting = df.loc[df['is_resting'] == 1]['heart_rate']
4 activity = df.loc[df['is_resting'] == 0]['heart_rate']
5
6 # Plot the heart rates onto the same plot
7 sns.kdeplot(resting, shade=True, color="b", label='resting')
8 sns.kdeplot(activity, shade=True, color="r", label='active')
9 plt.title('Density Plot with Heart Rate')
10 plt.xlim([20, 220])
11 plt.xlabel('Blood Pressure')
12 plt.ylabel('Density')
13 plt.show()
```



## ▼ HRV Measurements Data

TODO

```
1 print(dfs['hrv_measurements'].head())
```

	user_code	rr_code	measurement_datetime	time_of_day	bpm	meanrr	\
0	007b8190cf	10489a6aea	2020-04-21 21:23:08	morning	75	795.90	
1	007b8190cf	9610d4d4dc	2020-04-26 11:19:25	morning	70	858.00	
2	013f6d3e5b	f3de056155	2020-05-15 04:14:21	night	83	724.10	
3	013f6d3e5b	b04489e32f	2020-05-19 03:06:02	night	75	802.64	
4	01bad5a519	ac52c706c6	2019-12-31 09:07:43	morning	78	768.07	

	mxdmn	snnn	rmssd	pnn50	...	lf	hf	vlf	lfhf	\
0	0.12	45.802	54.174	15.15	...	508.0	1076.0	267.0	0.472	
1	0.11	32.889	33.022	16.16	...	409.0	310.0	176.0	1.319	
2	0.17	54.811	65.987	17.17	...	432.0	881.0	194.0	0.490	
3	0.20	72.223	70.039	22.22	...	814.0	1487.0	1719.0	0.547	
4	0.10	29.650	21.196	4.04	...	489.0	128.0	96.0	3.820	

	total_power	how_feel	how_mood	how_sleep	\
0	1851.0	0	-1	NaN	
1	895.0	0	0	0.0	
2	1507.0	-1	-2	NaN	
3	4020.0	0	0	NaN	
4	713.0	0	0	0.0	

	tags	\
0	COVID-19; Workout; Sex; Hobby; Studying; Sleep...	
1	NaN	
2	COVID-19; Fast/Diet; Hungry; Tired; Fever; I c...	
3	NaN	
4	NaN	

	rr_data
0	819,1008,831,847,785,778,866,839,801,793,846,8...
1	888,775,811,883,890,894,894,899,893,889,890,83...
2	694,832,642,801,751,716,737,742,773,760,701,73...
3	821,817,771,805,833,788,747,724,792,825,775,75...
4	741,740,734,737,740,731,751,747,745,728,747,76...

[5 rows x 22 columns]

## ▼ Sleep Data

TODO

```
1 print(dfs['sleep'].head())
```

	user_code	day	sleep_begin	sleep_end
0	0d297d2410	2019-12-31	2019-12-31 07:50:32	2019-12-31 08:45:22
1	0d297d2410	2020-01-01	2020-01-01 04:13:41	2020-01-01 09:45:02
2	0d297d2410	2020-01-02	2020-01-02 02:14:52	2020-01-02 08:06:00
3	0d297d2410	2020-01-03	2020-01-03 00:10:00	2020-01-03 08:45:10
4	0d297d2410	2020-01-04	2020-01-04 01:27:25	2020-01-04 08:52:20

	sleep_duration	sleep_awake_duration	sleep_rem_duration
0	3290.0	NaN	NaN
1	19881.0	NaN	NaN
2	21068.0	NaN	NaN
3	30910.0	NaN	NaN
4	26695.0	NaN	NaN

	sleep_light_duration	sleep_deep_duration	pulse_min	pulse_max
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	21480.0	NaN	55.0	95.0

	pulse_average
0	NaN
1	NaN
2	NaN
3	NaN
4	72.5

## ▼ Survey Data

In the survey data, the participants describe their severity level of COVID-19 related symptoms and some other health conditions.

```
1 print(dfs['surveys']['scale'].value_counts())
```

S_COVID_FATIGUE	209
S_COVID_COUGH	207

S_COVID_COUGH	207
S_COVID_OVERALL	207
S_COVID_TROUBLE	206
S_COVID_BREATH	205
S_COVID_CONFUSION	204
S_COVID_PAIN	204
S_COVID_BLUISH	203
S_COVID_FEVER	203
S_CORONA	86
S_HRA_ALC	51
S_HRA_VIT	41
S_HRA_SLEEP	23
S_HRA_ANX	21
S_HEART	15
S_HRA_ALLERG	13
S_HRA_BONE	12
S_HRA_D	12
S_HRA_DEP	11
S_HRA_PANIC	9
S_HRA_LUNG	8
S_HRA_ASTHMA	7
S_HRA_NECK	7
S_HRA_ARR	6
S_HRA_VARI	6
S_HRA_COLDS	6
S_HRA_LIVER	5
S_HRA_CHOL	5
S_HRA_IRR	4
S_HRA_ANEMIA	4
S_HRA_POST	4
S_HRA_HEAD	4
S_COVID_SYMPTOMS	4
S_HRA_JOINT	4
S_HRA_SUGAR	4
S_HEART_1	3
S_DIAB_REASON3	3
S_HEART_3	3
S_HRA_LBP	3
S_DIABETES	3
S_HRA_PERPAIN	2
S_HRA_HEAVY	2
S_DIAB_REASON5	2
S_HRA_DBT	2
S_HRA_OVARY	2
S_HRA_FIBRO	2
S_HEART_5	1
S_HRA_HORM	1
S_DIAB_REASON6	1
S_HEART_4	1
S_HRA_OCD	1
S_DIAB_REASON4	1

```

S_HRA_ENDO          1
S_HRA_EPILEPSY      1
S_DIAB_REASON2      1
S_DIAB_REASON1      1
S_HRA_HBP            1
S_HRA_EDEMA          1
Name: scale, dtype: int64

```

The descriptions of each scale can be found at 'scales\_description.csv'

```
1 print(dfs['scales_description'].head())
```

	Scale	Description	Value
0	S_COVID_SYMPTOMS	How long the user has been experiencing symptoms	1
1	S_COVID_SYMPTOMS	How long the user has been experiencing symptoms	2
2	S_COVID_SYMPTOMS	How long the user has been experiencing symptoms	3
3	S_COVID_SYMPTOMS	How long the user has been experiencing symptoms	4
4	S_COVID_COUGH	Symptom intensity: Coughing	1

	Meaning
0	Less than 3 days
1	3 to 6 days
2	7 to 14 days
3	More than 14 days
4	User isn't experiencing symptom

Here we plot the sum of the scores with respect to the 9 questions related to COVID-19 symptoms.

```

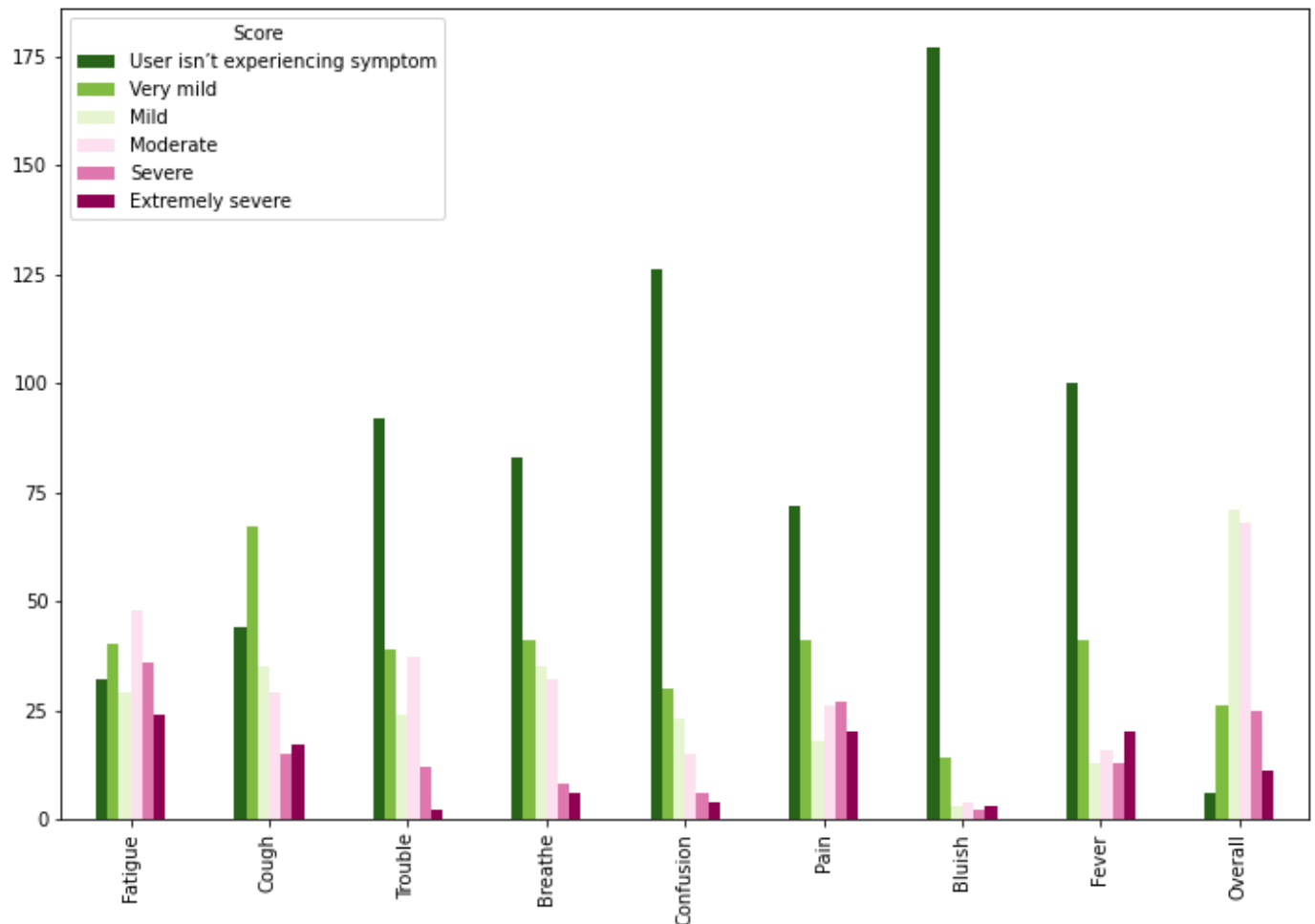
1 df = dfs['surveys']
2
3 # Set labels of each item
4 labels = ['Fatigue', 'Cough', 'Trouble', 'Breathe', 'Confusion', 'Pain', 'Bluish', 'Fe
5
6 # Retrieve statistic of the survey
7 covidFatigue = df.loc[df['scale'] == 'S_COVID_FATIGUE'].groupby(['value']).size()
8 covidCough = df.loc[df['scale'] == 'S_COVID_COUGH'].groupby(['value']).size()
9 covidTrouble = df.loc[df['scale'] == 'S_COVID_TROUBLE'].groupby(['value']).size()
10 covidBreathe = df.loc[df['scale'] == 'S_COVID_BREATH'].groupby(['value']).size()
11 covidConfusion = df.loc[df['scale'] == 'S_COVID_CONFUSION'].groupby(['value']).size()
12 covidPain = df.loc[df['scale'] == 'S_COVID_PAIN'].groupby(['value']).size()
13 covidBluish = df.loc[df['scale'] == 'S_COVID_BLUISH'].groupby(['value']).size()
14 covidFever = df.loc[df['scale'] == 'S_COVID_FEVER'].groupby(['value']).size()

```

```

15 covidOverall = df.loc[df['scale'] == 'S_COVID_OVERALL'].groupby(['value']).size
16
17 # Plot grouped bar graph
18 dfAll = pd.concat([covidFatigue, covidCough, covidTrouble, covidBreathe, covidC
19 dfAll = dfAll.rename_axis('Score', axis='columns')
20 dfAll = dfAll.rename(columns={1: 'User isn't experiencing symptom', 2: 'Very mi
21 dfAll = dfAll.rename(index={0: 'Fatigue', 1: 'Cough', 2: 'Trouble', 3: 'Breathe
22 dfAll.plot(kind='bar', figsize=(12, 8), colormap='PiYG_r')
23 plt.show()

```



## ▼ Wearables Data

TODO

```
1 print(dfs['wearables'].head())
```

	user_code	day	resting_pulse	pulse_average	pulse_min	pulse_max
0	007b8190cf	2020-04-26	NaN	70.0	70.0	70.0
1	01bad5a519	2020-02-12	NaN	NaN	NaN	NaN
2	01bad5a519	2020-02-13	NaN	NaN	NaN	NaN
3	01bad5a519	2020-02-15	NaN	NaN	NaN	NaN
4	01bad5a519	2020-02-16	NaN	NaN	NaN	NaN

	average_spo2_value	body_temperature_avg	stand_hours_total	steps_count
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	8574.0
2	NaN	NaN	NaN	7462.0
3	NaN	NaN	NaN	2507.0
4	NaN	NaN	NaN	10131.0

	distance	steps_speed	total_number_of_flights_climbed	\
0	NaN	NaN	NaN	
1	NaN	57.90	NaN	
2	NaN	59.10	NaN	
3	NaN	60.97	NaN	
4	NaN	49.10	NaN	

	active_calories_burned	basal_calories_burned	total_calories_burned	\
0	NaN	2859.0	2859.0	
1	NaN	2624.0	2624.0	
2	NaN	2624.0	2624.0	
3	NaN	2624.0	2624.0	
4	NaN	2624.0	2624.0	

	average_headphone_exposure	average_environment_exposure
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

## ▼ Weather Data

The weather data describes the environment where the readings are measured. This includes the date, average temperature (celcius), atmospheric pressure, precipitation intensity, humidity, and clouds.

```
1 print(dfs['weather'].head())
```

	user_code	day	avg_temperature_C	atmospheric_pressure	\
0	013f6d3e5b	2020-05-22	18.0667	1017.6	
1	01bad5a519	2020-01-11	-1.2111	1016.4	
2	01bad5a519	2020-01-30	0.5056	1004.7	
3	01bad5a519	2020-04-02	-0.2444	994.4	
4	01bad5a519	2020-04-12	5.1778	1016.1	

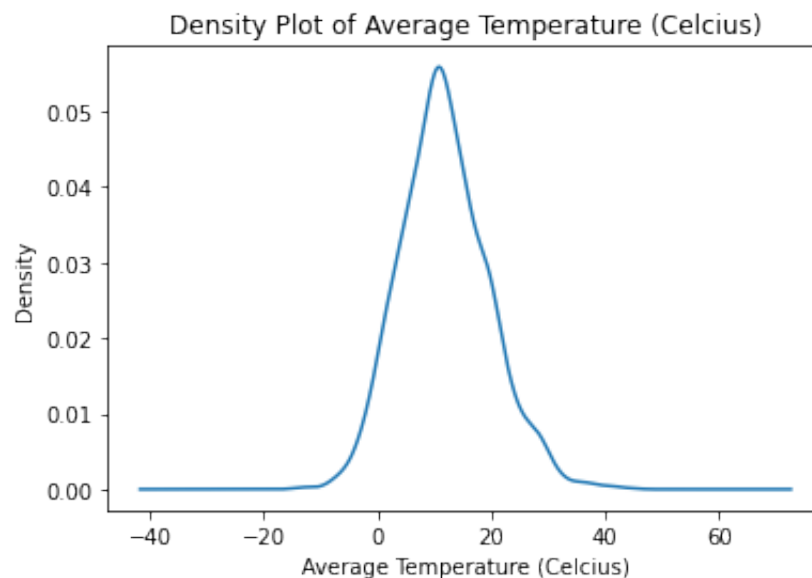
  

	precip_intensity	humidity	clouds
0	0.0002	70.0	67.0
1	0.0002	92.0	6.0
2	0.0009	85.0	100.0
3	0.0025	91.0	87.0
4	0.0000	61.0	91.0

The density plot for each item is plotted below.

### ▼ Average Temperature (Celcius)

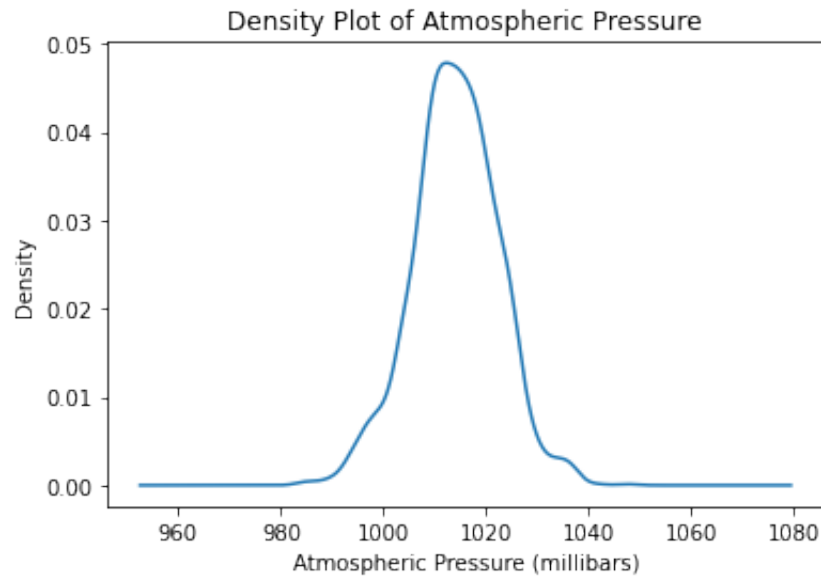
```
1 ax = dfs['weather']['avg_temperature_C'].plot(kind='kde', title='Density Plot o
2 ax.set_xlabel('Average Temperature (Celcius)')
3 plt.show()
```





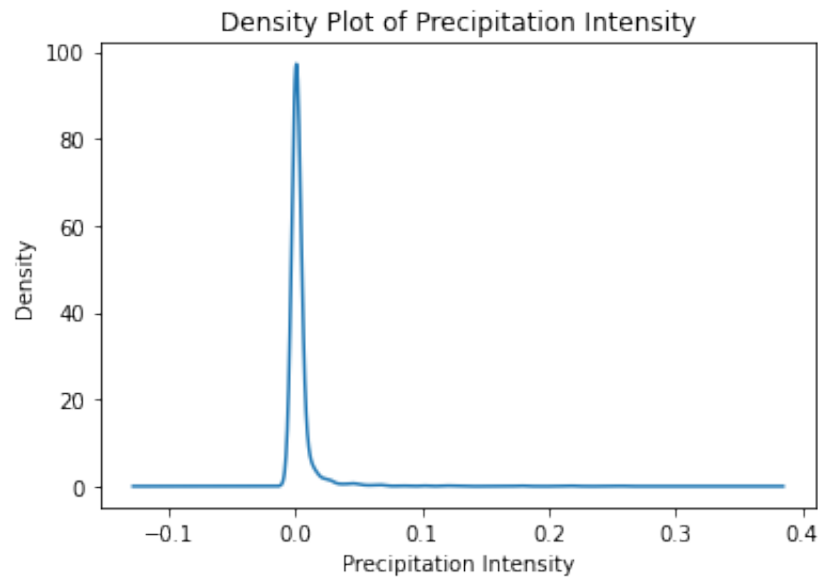
## ▼ Atmospheric Pressure

```
1 ax = dfs['weather']['atmospheric_pressure'].plot(kind='kde', title='Density Plot of Atmospheric Pressure')
2 ax.set_xlabel('Atmospheric Pressure (millibars)')
3 plt.show()
```



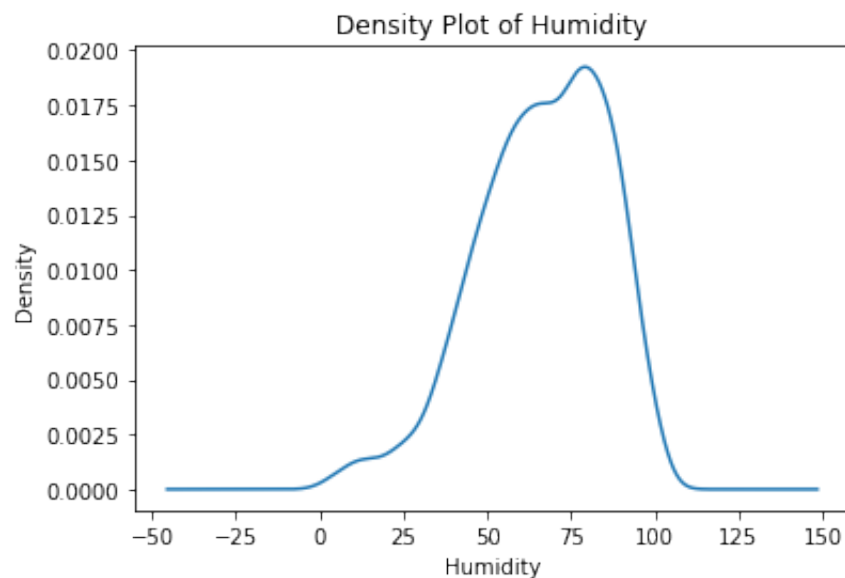
## ▼ Precipitation Intensity

```
1 ax = dfs['weather']['precip_intensity'].plot(kind='kde', title='Density Plot of  
2 ax.set_xlabel('Precipitation Intensity')  
3 plt.show()
```



## ▼ Humidity

```
1 ax = dfs['weather']['humidity'].plot(kind='kde', title='Density Plot of Humidit  
2 ax.set_xlabel('Humidity')  
3 plt.show()
```



## ▼ Clouds

```
1 ax = dfs['weather']['clouds'].plot(kind='kde', title='Density Plot of Clouds')
2 ax.set_xlabel('Clouds')
3 plt.show()
```

