Weather & Health Project Initial Brief

Data Source

Main Dataset: Chronic Illness: Symptoms, Treatments, and Triggers

Jupyter Notebook on Github: Initial Data Exploration and Cleaning

Description: Exploring how treatments and environmental stressors impact user reported symptoms from the Flaredown app. Weather is recorded automatically at time of reporting, so geographical location and weather condition factors can be examined in addition to self reported triggers.

Reason for choosing:

I suffer from chronic illness and understanding the factors that can affect my health is a practical concern. In addition to my own experience, my past in biomedical research makes the data of interest to me.

Variables included:

user_id object
age float64
sex object
country object
checkin_date object
trackable_id object
trackable_type object
trackable_name object
trackable_value object

Possible Questions

- Does, and if so, how does X treatment affect Y symptom? What symptoms and treatments are correlated?
- Are there subsets of triggers and effects that could more accurately represent symptoms and predict known effective treatments?
- Is it possible to reliably predict flare triggers for a given user or condition?
- Could we develop a way to recommend treatments more effectively based on similarity of users? (an algorithm for recommendations for treatments or things for certain people to avoid)
- Can we guess a condition based on a user’s symptoms and triggers?

Limitations & ethical considerations

- **Subjectivity and Bias:**
  - Users self report everything, so there's not a standard measure for symptoms and triggers or effectiveness of treatments.
  - The user base is very largely slanted toward female, so any results may not be as transferable to males.
- **Correlation and Causation**:
  - The nature of the data, and the analysis that can be done with it, leaves any insight up to coming from correlation. To figure out any causation, the users would need to be selected to meet certain uniform sets of criteria, even if there are different subset of people and triggers applied repeatedly to see if the same results happen after each trial. This is a good start though to finding relevant correlations.

Ethical considerations:

- **Privacy**: The user IDs are not identifiable and none of the other data should be, but I'll be on the lookout for ways that personally identifiable info could creep into the self entered field
- **Health advice**: Any conclusions from correlated symptoms and triggers should not be taken as health advice.

Summary from the Data Card:

Introduction

- *Flaredown is an app that helps patients of chronic autoimmune and invisible illnesses improve their symptoms by avoiding triggers and evaluating their treatments. Each day, patients track their symptom severity, treatments and doses, and any potential environmental triggers (foods, stress, allergens, etc) they encounter.*

About the data

- *Instead of coupling symptoms to a particular illness, Flaredown asks users to create their unique set of conditions, symptoms and treatments ("trackables"). They can then "check-in" each day and record the severity of symptoms and conditions, the doses of treatments, and "tag" the day with any unexpected environmental factors.*

  User: *includes an ID, age, sex, and country.*

  Condition: *an illness or diagnosis, for example Rheumatoid Arthritis, rated on a scale of 0 (not active) to 4 (extremely active).*

  Symptom: *self-explanatory, also rated on a 0–4 scale.*

  Treatment: *anything a patient uses to improve their symptoms, along with an optional dose, which is a string that describes how much they took during the day. For instance "3 x 5mg".*

  Tag: *a string representing an environmental factor that does not occur every day, for example "ate dairy" or "rainy day".*

  Food: *food items were seeded from the publicly-available USDA food database. Users have also added many food items manually.*

  Weather: *weather is pulled automatically for the user's postal code from the Dark Sky API. Weather parameters include a description, precipitation intensity, humidity, pressure, and min/max temperatures for the day.*

  HBI: *the Harvey Bradshaw Index is a standardized metric to gauge the severity of Crohn's disease specifically, often used in evaluation of therapies. Patients with Crohn's disease who scored 3 or less on the HBI are very likely to be in remission according to the CDAI. Patients with a score of 8 to 9 or higher are considered to have severe disease.*

  If users do not see a symptom, treatment, tag, or food in our database (for instance "Abdominal Pain" as a symptom) they may add it by simply naming it. This means that the data requires some cleaning, but it is patient-centered and indicates their primary concerns.

Data description and characterization

I worked with these in the Jupyter notebook and included the lists and charts here:

```
df_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7976223 entries, 0 to 7976222
Data columns (total 9 columns):
 #   Column           Dtype
---  ------           -----
 0   user_id          object
 1   age              float64
 2   sex              object
 3   country          object
 4   checkin_date     object
 5   trackable_id     object
 6   trackable_type   object
 7   trackable_name   object
 8   trackable_value  object
dtypes: float64(1), object(8)
memory usage: 547.7+ MB
```

```
df_original.nunique()
```

```
user_id             42283
age                   100
sex                     4
country               164
checkin_date         1675
trackable_id       222465
trackable_type          7
trackable_name     117214
trackable_value     15960
dtype: int64
```

```
df_original.head()
```

	user_id	age	sex	country	checkin_date	trackable_id	trackable_type	trackable_name	trackable_value
0	QEVuQwEABlEzkh7fsBBjEe26RylVcg==	NaN	NaN	NaN	2015-11-26	1069	Condition	Ulcerative colitis	0
1	QEVuQwEAWRNGnuTRqXG2996KSkTIEw==	32.0	male	US	2015-11-26	1069	Condition	Ulcerative colitis	0
2	QEVuQwEA+WkNxtp/qkHvN2YmTBBDqg==	2.0	female	CA	2017-04-28	3168	Condition	pain in left upper arm felt like i was getting...	4
3	QEVuQwEA+WkNxtp/qkHvN2YmTBBDqg==	2.0	female	CA	2017-04-28	3169	Condition	hip pain when gettin up	3
4	QEVuQwEA+WkNxtp/qkHvN2YmTBBDqg==	2.0	female	CA	2017-04-28	3170	Condition	pain in hand joints	4

'user_id' is alphanumerical and unique for distinct people, so let's use unique integer user ids to save memory

```
df_original['user_id'] = pd.Categorical(df_original['user_id'])
df_original['user_id']= df_original.user_id.cat.codes
```

```
df_original.head()
```

	user_id	age	sex	country	checkin_date	trackable_id	trackable_type	trackable_name	trackable_value
0	9070	NaN	NaN	NaN	2015-11-26	1069	Condition	Ulcerative colitis	0
1	22737	32.0	male	US	2015-11-26	1069	Condition	Ulcerative colitis	0
2	376	2.0	female	CA	2017-04-28	3168	Condition	pain in left upper arm felt like i was getting...	4
3	376	2.0	female	CA	2017-04-28	3169	Condition	hip pain when gettin up	3
4	376	2.0	female	CA	2017-04-28	3170	Condition	pain in hand joints	4

We can see above that there are many values in age column which are 0.0 which can bias our inference. So I am replacing these by NaN for consistency.

Cleaning up data consistency

```
df_original["age"] = df_original.age.replace(0.0,np.nan)
```

```
df_original.head()
```

	user_id	age	sex	country	checkin_date	trackable_id	trackable_type	trackable_name	trackable_value
0	9070	NaN	NaN	NaN	2015-11-26	1069	Condition	Ulcerative colitis	0
1	22737	32.0	male	US	2015-11-26	1069	Condition	Ulcerative colitis	0
2	376	2.0	female	CA	2017-04-28	3168	Condition	pain in left upper arm felt like i was getting...	4
3	376	2.0	female	CA	2017-04-28	3169	Condition	hip pain when gettin up	3
4	376	2.0	female	CA	2017-04-28	3170	Condition	pain in hand joints	4

Age data before cleaning

```
df_original.age.describe()
```

```
count     7.666965e+06
mean      3.506981e+01
std       1.437929e+02
min      -1.966910e+05
25%       2.600000e+01
50%       3.400000e+01
75%       4.300000e+01
max       2.018000e+03
Name: age, dtype: float64
```

Here , minimum and maximum age are not valid, we need to clean it more .

Since negative age and above 117 is not practically possible so replacing them by NaN.¶

```
df_original[(df_original['age'] > 117) | (df_original['age'] < 0) ].shape  # number of columns to be replaced by NaN
```

```
(478, 9)
```

```
df_original = df_original.assign(age = lambda x: x.age.where(x.age.ge(0)))    # All negative ages replaced by NaN for consistency
```

```
df_original = df_original.assign(age = lambda x: x.age.where(x.age.le(117)))  # All ages greater than 117 are replaced by NaN
```

```
df_original[(df_original['age'] > 117) | (df_original['age'] < 0) ].shape  # as we can see they are replced
```

```
(0, 9)
```

```
df_original.age.describe()   # now age statistics makes more sense
```

```
count    7.666487e+06
mean     3.508054e+01
std      1.171827e+01
min      1.000000e+00
25%      2.600000e+01
50%      3.400000e+01
75%      4.300000e+01
max      9.900000e+01
```

^Corrected age data

Categorizing users on the basis of gender:

```
df_original.sex.value_counts()  # Total number of check-ins of differet sex categories
```

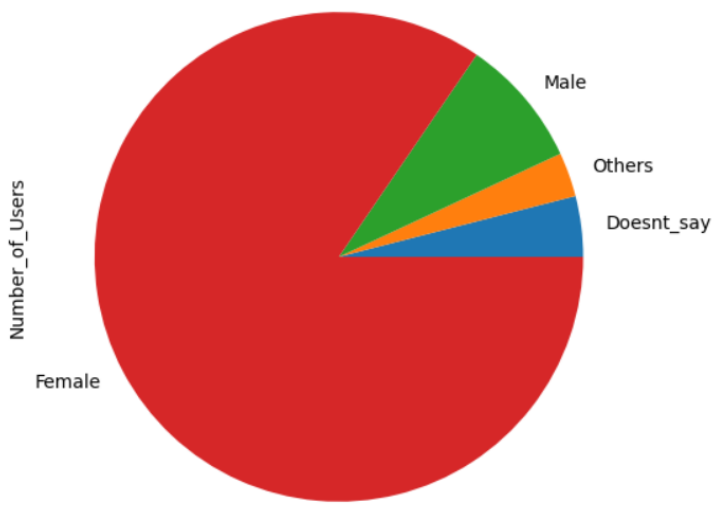
```
sex
female        6478402
male           574907
other          428312
doesnt_say     362467
Name: count, dtype: int64
```

```
df_sex_unique = pd.DataFrame([{'Number_of_Users' : df_original[df_original.sex=="doesnt_say"].user_id.nunique()}
                              ,{'Number_of_Users' : df_original[df_original.sex=="other"].user_id.nunique()}
                              ,{'Number_of_Users' : df_original[df_original.sex=="male"].user_id.nunique()}
                              ,{'Number_of_Users' : df_original[df_original.sex=="female"].user_id.nunique()}
                              ], index=['Doesnt_say', 'Others', 'Male','Female'])
df_sex_unique.head()
```

	Number_of_Users
Doesnt_say	1640
Others	1200
Male	3497
Female	34659

```
plt.figure(figsize=(10,6))
df_sex_unique.Number_of_Users.plot(kind='pie')
```

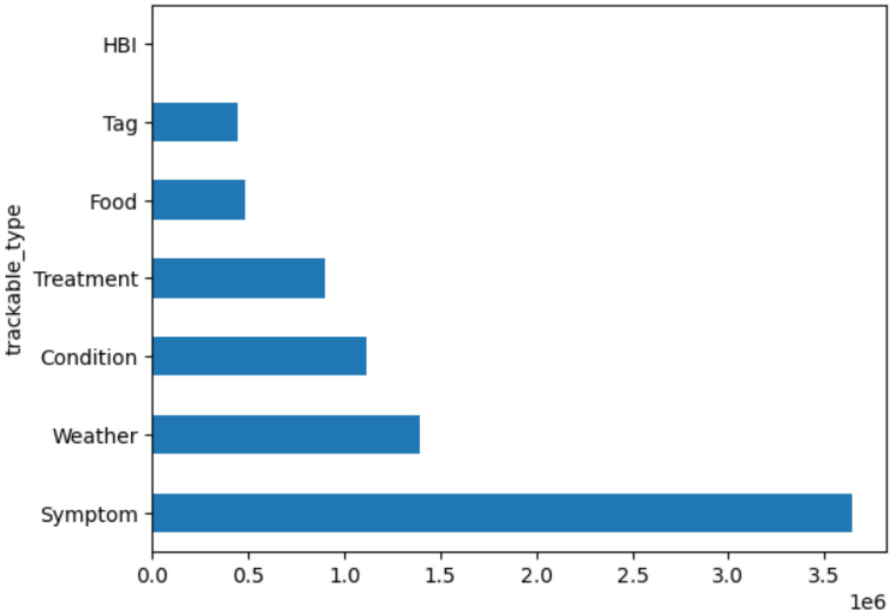
```
<Axes: ylabel='Number_of_Users'>
```



Categorizing entries on the basis of trackable type:

```
df_original.trackable_type.value_counts()
```

```
trackable_type
Symptom      3642279
Weather      1393806
Condition    1111517
Treatment     901820
Food          480971
Tag           445669
HBI              161
Name: count, dtype: int64
```




```
df_original[df_original.trackable_type=="Symptom"].trackable_name.value_counts().head(10)  # Top 10 different symptoms traced
```

```
trackable_name
Headache                 108550
Fatigue                  107512
Nausea                    89520
Brain fog                 73175
Joint pain                64936
Fatigue and tiredness     63395
Anxiety                   61545
Diarrhea                  52418
Dizziness                 50150
Depression                43370
Name: count, dtype: int64
```

Exploring weather effects

```
print("There are a total of ",df_original[df_original.trackable_type=="Weather"].trackable_name.nunique()," unique weather conditions")
```

```
There are a total of  6  unique weather conditions
```

```
df_original[df_original.trackable_type=="Weather"].trackable_name.value_counts()
```

```
trackable_name
icon               232301
temperature_min    232301
temperature_max    232301
precip_intensity   232301
pressure           232301
humidity           232301
Name: count, dtype: int64
```

```python
s_max = df_original[df_original.trackable_name=="temperature_max"].trackable_value
s_min = df_original[df_original.trackable_name=="temperature_min"].trackable_value
```

```python
max_temp = pd.to_numeric(s_max, errors='coerce')
min_temp = pd.to_numeric(s_min, errors='coerce')
```

```python
max_temp.describe()
```

```
count    232301.000000
mean         64.854288
std          18.754588
min         -21.000000
25%          51.000000
50%          67.000000
75%          80.000000
max         119.000000
Name: trackable_value, dtype: float64
```

```python
#Pressure description
pd.to_numeric(df_original[df_original.trackable_name=="pressure"].trackable_value, errors='coerce').describe()
```

```
count    232301.000000
mean       1016.427583
std           7.600215
min         938.000000
25%        1012.000000
50%        1016.000000
75%        1021.000000
max        1051.000000
Name: trackable_value, dtype: float64
```

Humidity

```
#Humidity description
pd.to_numeric(df_original[df_original.trackable_name=="humidity"].trackable_value, errors='coerce').describe()
```

```
count    232334.000000
mean         70.817625
std          15.379442
min           1.000000
25%          63.000000
50%          73.000000
75%          82.000000
max         100.000000
Name: trackable_value, dtype: float64
```

```
#Precipitation Intensity
pd.to_numeric(df_original[df_original.trackable_name=="precip_intensity"].trackable_value, errors='coerce').describe()
```

```
count    232096.000000
mean          0.004591
std           0.013581
min           0.000000
25%           0.000100
50%           0.000500
75%           0.003700
max           1.098300
Name: trackable_value, dtype: float64
```

```
print("There are a total of ",df_original[df_original.trackable_type=="Condition"].trackable_name.nunique()," unique conditions")
```

```
There are a total of  9443  unique conditions
```

```
df_original[df_original.trackable_type=="Condition"].trackable_name.value_counts().head(10)
```

```
trackable_name
Fibromyalgia               55255
Depression                 50109
Anxiety                    46968
Chronic fatigue syndrome   28259
Migraine                   26082
IBS                        17324
Fatigue                    14920
Asthma                     14218
Endometriosis              13873
Ehlers-Danlos syndrome     13677
Name: count, dtype: int64
```

Types of treatments used

```
print("There are a total of ",df_original[df_original.trackable_type=="Treatment"].trackable_name.nunique()," unique treatments")
```

```
There are a total of  8154  unique treatments
```

```
df_original[df_original.trackable_type=="Treatment"].trackable_name.value_counts().head(10)
```

```
trackable_name
Ibuprofen      21484
Magnesium      11417
Paracetamol    11253
Vitamin D3     11168
Vitamin d       9867
Gabapentin      9627
Tramadol        9278
Prednisone      8309
Naproxen        8211
Omeprazole      7552
Name: count, dtype: int64
```

Wordcloud for Common Tags