analyze_loseit_challenge_data

Analyze r/loseit Challenge Data¶

Now we can read in the already cleaned file. If you don't have the cleaned data, you will need to run Find and Clean Loseit Data and Inspect Loseit Data.

We begin by loading in the dataset and look at the counts.

Partipicant Stats¶

Username¶

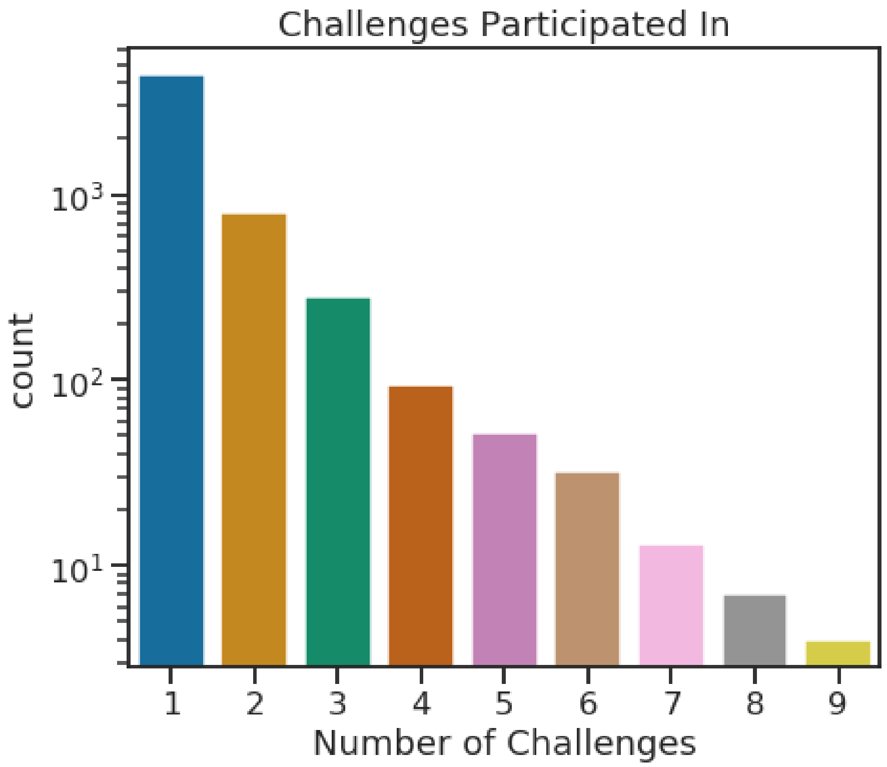
From partipicating in a few different challenges, there are many people who have done multiple challenges. I want to begin by looking at how many have done more than one challenge, and who has partipicated in the most challenges.

```
multiple challenges: 1288 and single challenge: 4397
```

So we can see that there have been 1288 people who have participated in more than one challenge (and using the same username) and 4397 people who have only participated in a single challenge so far – again this doesn't account for people who sign-up using multiple accounts.

```
bugs_bunny01           9
axecutable             9
Lovellama              9
BlackAnemones          9
kmrbriscoe             8
unreuly                8
shd244                 8
soahtree               8
Radiant_Indignation    8
MrBad-Example          8
BethLynn85             8
kej9311                7
angiev70               7
thosethighstho         7
```

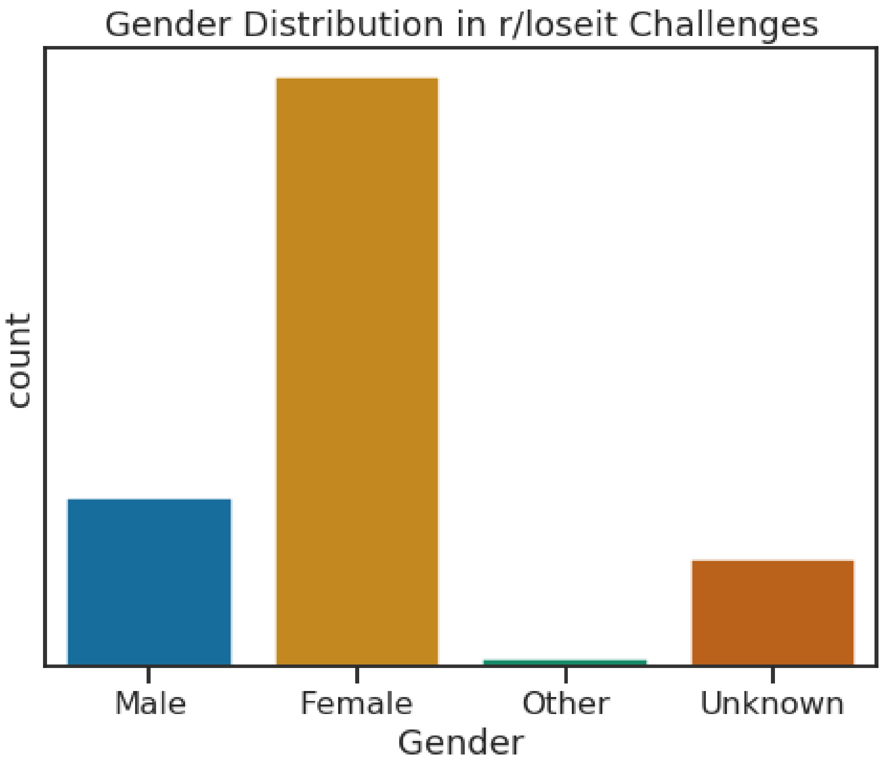
```
Radioactive_Kitten     7
Name: Username, dtype: int64
```



Comment about the number of challenges that people partipicate in.

Gender¶

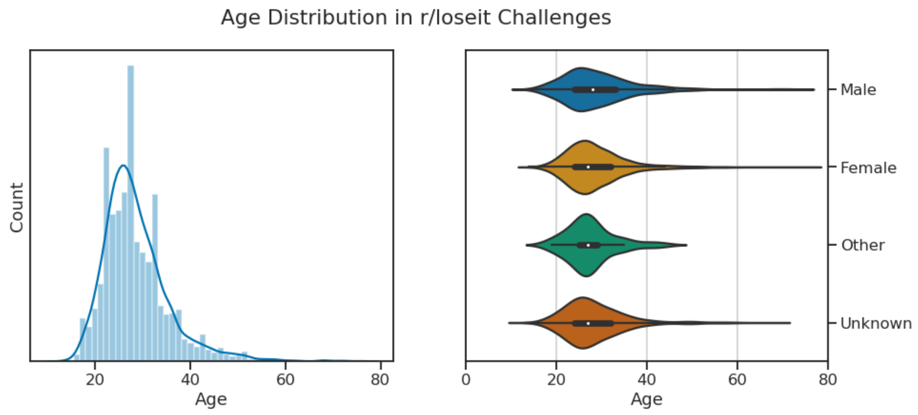
Let's start by looking at the gender distribution of r/loseit partipicants.



```
Males: 19.33% , Females: 67.44%, Other: 0.87%, Unknown: 12.37%
```

Looking that this plot it is pretty clear that loseit challenges have a very large gender imbalance. Around 67% of the partipicants in the challenges are women, 19% are male, 12% are unknown, and almost 1% of partipicants identify as other. Because there is such a large imbalance, for most of my analysis I will try and look at how the numbers vary between gender – if they do at all.

Age¶



```
most common: [(26.0, 647)], average: 28.6, yongest: 13.0, oldest: 76.0
```

Looking at the age distributions, we find that there is a large peak at the most common age 26. The oldest participant is 76 and the youngest has been 13. Looking the distributions seperated by gender, we see that there us not a huge difference between them.

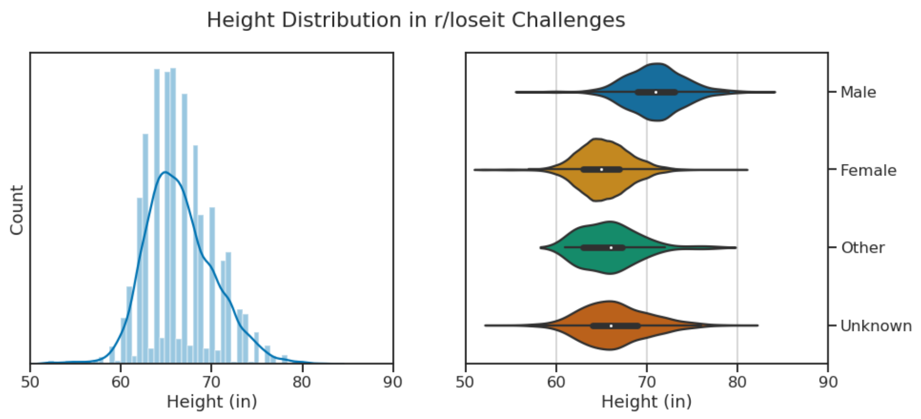
One thing of note, is that there are many people who have done multiple challenes. These results might change slightly if we were only look at the distributions for unique usernames.

```
most common: [(26.0, 470)], average: 28.4, yongest: 13.0, oldest: 76.0
```

So we find that even with removing duplicate usernames, the most common age of partipicants is still 26.

Height¶

The next piece of information that I want to look at is the height distribution.

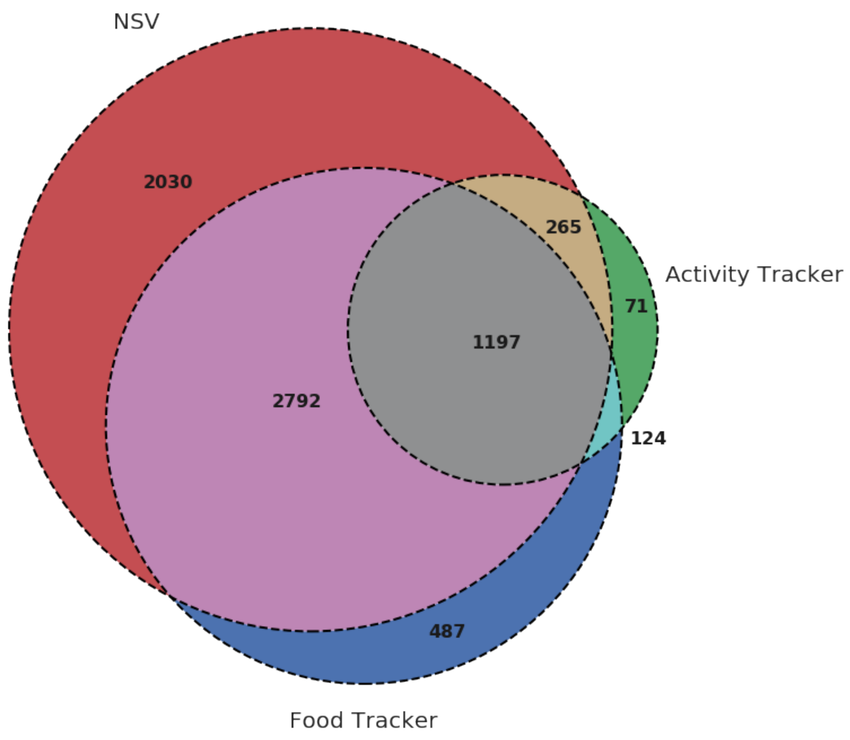


```
average: 66.5 in, shortest: 52.0 in, tallest: 82.6 in
```

NSV and Tracking¶

Next, I want to look at the proportion of those who give links to either fitness or food trackers and provide a NSV.

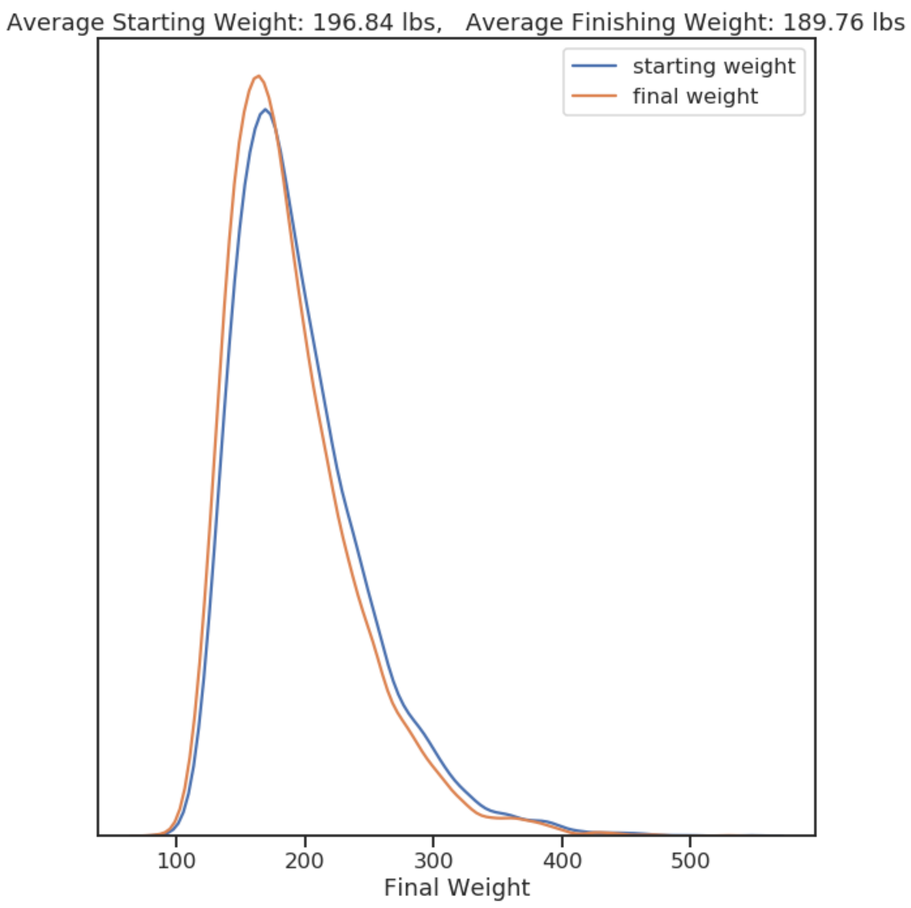
```
Has NSV: 79.96
Gives activity tracker: 21.08
Gives food tracker: 58.53
```



```
number of people who don't give any info: 893, 11.36%
```

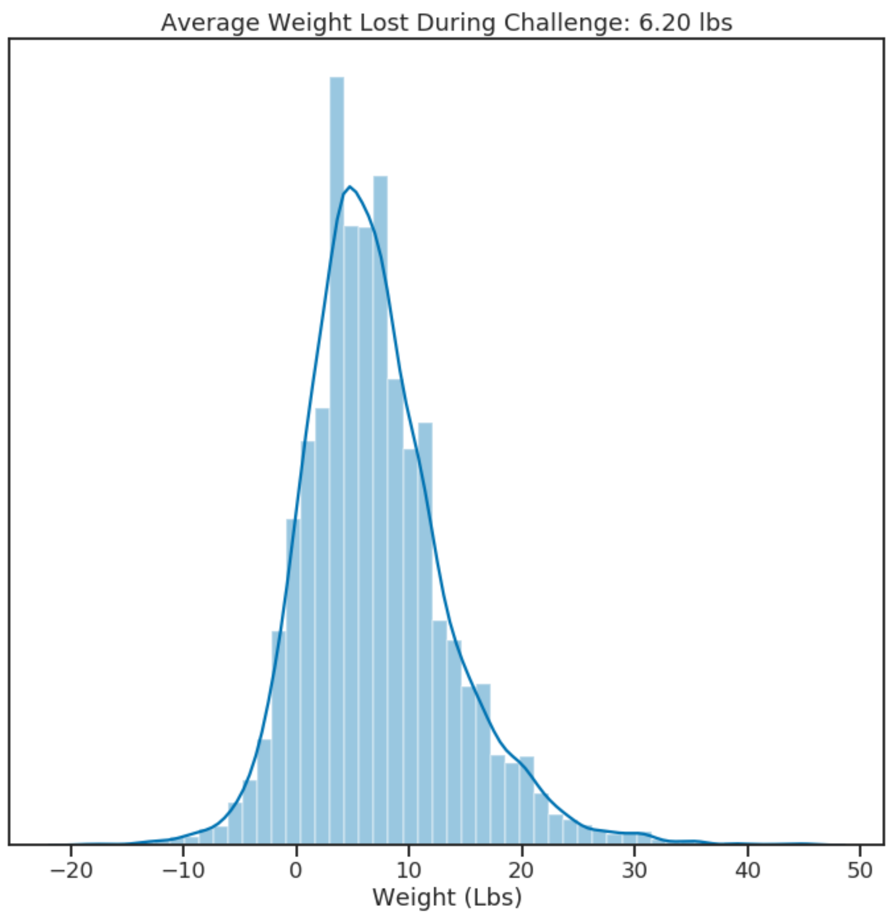
In [81]:

```
plt.figure(figsize=(10, 10))
ax = sns.distplot(loss_df['Starting Weight'],
                  hist=False, label='starting weight')
ax = sns.distplot(loss_df['Final Weight'], hist=False, label='final weight')
ax.set_yticks([])
ax.set_title(
    f'Average Starting Weight: {loss_df["Starting Weight"].mean():.2f} lbs,   Average Finish
ax.legend()
plt.tight_layout()
plt.savefig('./figures/starting_final_weight.png', dpi=400)
```



In [138]:

```
plt.figure(figsize=(10, 10))
ax = sns.distplot(loss_df['Total Challenge Loss'])
ax.set_xlabel('Weight (Lbs)')
ax.set_title(
    f'Average Weight Lost During Challenge: {loss_df["Total Challenge Loss"].median():.2f} l
ax.set_yticks([])
plt.tight_layout()
plt.savefig('./figures/average_weight_lost.png', dpi=400)
```

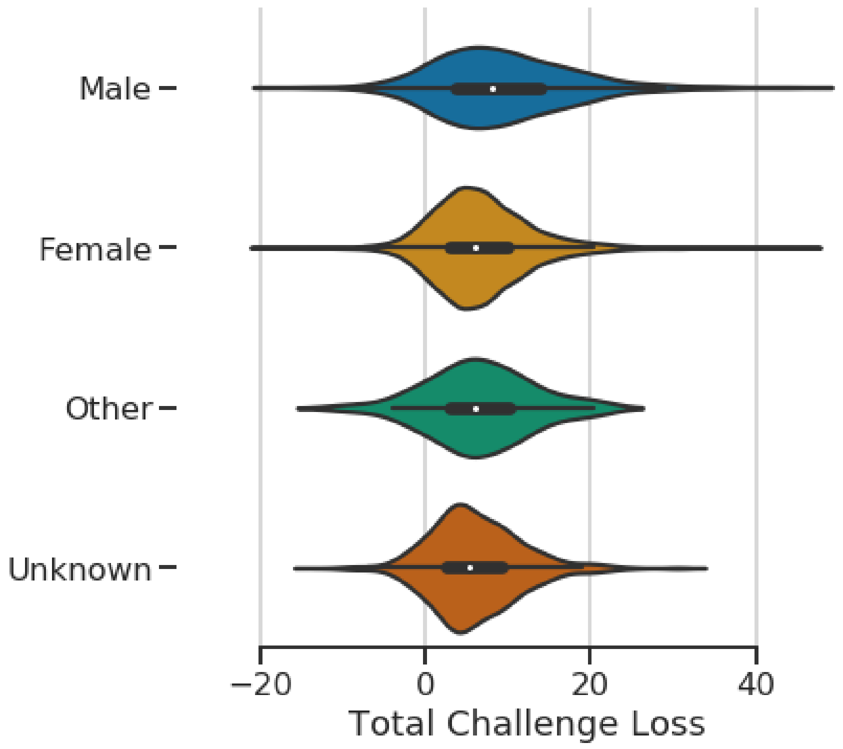


In [90]:

```
f, ax = plt.subplots(figsize=(7, 6))

sns.violinplot(x="Total Challenge Loss", y="Gender", data=loss_df)

ax.xaxis.grid(True)
ax.set(ylabel="")
ax.set_xlim([-30, 55])
sns.despine(trim=True, left=True)
f.tight_layout()
f.savefig('./figures/violin_gender_weight_lost.png', dpi=400)
```



In [43]:

```
combined_nsv = ''
for idx in range(len(big_df)):
    combined_nsv += f'{big_df["NSV Text"].values[idx]}. '
```

In [24]:

```
Counter(big_df['NSV Text'].str.lower()).most_common(50)
```

Out[24]:

```
[('', 1575),
 ('fit into old clothes', 43),
 ('run a 5k', 34),
 ('run 5k', 23),
 ('finish c25k', 14),
 ('run 10k', 13),
 ('run a 10k', 10),
 ('go down a pant size', 10),
 ('run a mile', 9),
 ('fit into old jeans', 8),
 ('complete c25k', 7),
 ('fit into my old jeans', 6),
 ('drop a pant size', 6),
 ('start c25k', 6),
 ('fit into old clothes ', 5),
 ('lose a pant size', 5),
 ('fit into an old dress', 5),
 ('run faster', 5),
 ('feel better', 5),
 ('finish c25k!', 5),
 ('clothes fitting better', 5),
 ('consistency', 5),
 ('visible abs', 5),
 ('go down a pants size', 5),
 ('fit into clothes better', 5),
 ('200', 5),
 ('run a 5k without stopping', 5),
 ('have more energy', 4),
 ('fit into old clothes.', 4),
 ('size 12 jeans!', 4),
 ('run 5k without stopping', 4),
 ('run for longer distances', 4),
 ('start running again', 4),
 ('run a 5k ', 4),
 ('fitting into my old clothes', 4),
 ('log every day', 4),
 ('sub 30 minute 5k', 4),
 ('size 8 jeans', 4),
 ('feel more confident', 4),
 ('run 5km', 4),
 ('clothes fit better', 4),
 ('size 12', 4),
 ('145', 4),
 ('150', 4),
 ('run a 5k!', 4),
 ('170', 4),
 ('fit into pre-pregnancy jeans', 3),
 ('look better in clothes', 3),
 ('fit into smaller clothes', 3),
 ('fit into my shorts', 3)]
```

In [ ]:

```
doc = nlp(combined_nsv)
# all tokens that arent stop words or punctuations
words = [token.text for token in doc.doc if token.is_stop !=
         True and token.is_punct != True]
# noun tokens that arent stop words or punctuations
```

```
nouns = [token.text for token in doc.doc if token.is_stop !=
         True and token.is_punct != True and token.pos_ == "NOUN"]
```

In [ ]:

```
# five most common tokens
word_freq = Counter(words)
common_words = word_freq.most_common(5)

# five most common noun tokens
noun_freq = Counter(nouns)
common_nouns = noun_freq.most_common(5)
```

In [ ]:

```
word_freq.most_common(10)
```

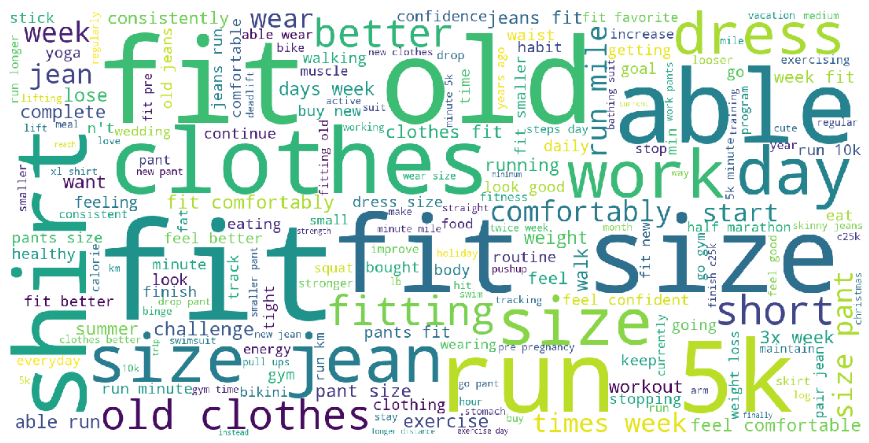
In [ ]:

```
noun_freq.most_common(10)
```

In [56]:

```
# Create and generate a word cloud image:
num_nsv = 300
text = " ".join(review for review in words)
text = text.lower()
wordcloud = WordCloud(background_color="white",
                      width=1600, height=800).generate(text)

# Display the generated image:
plt.figure(figsize=(20, 10), facecolor='w')
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig(
    f'./figures/spacy_nsv_{num_nsv}_wordcloud.png', facecolor='w', bbox_inches='tight')
```



In [54]:

```
text = " ".join(review for review in words)
```

In [93]:

```
# list of text documents
text = ["The quick brown fox jumped over the lazy dog."]
# create the transform
vectorizer = CountVectorizer()
# tokenize and build vocab
vectorizer.fit(big_df['NSV Text'])
# summarize
print(list(vectorizer.vocabulary_.items())[:10])
print("\n vocabulary size {}".format(len(vectorizer.vocabulary_)))
all_words = vectorizer.get_feature_names()
hot_words = vectorizer.fit_transform(big_df['NSV Text']).toarray()
```

```
[('finish', 1209), ('the', 2767), ('10k', 20), ('under', 2905), ('44mins', 215), ('fit', 12

 vocabulary size 3100
```

In [94]:

```
freq = hot_words.sum(axis=0)
```

In [95]:

```
words_df = pd.DataFrame(freq)
```

In [96]:

```
words_df = words_df.T
```

In [97]:

```
words_df.columns = all_words
```

In [98]:

```
words_df[words_df.columns[words_df.loc[0].argsort()][::-1]].T.head(20)
```

Out[98]:

	0
to	1593
my	1446
fit	1334
into	1208
in	865
size	752
the	685
run	617
and	581
clothes	466
jeans	461
of	453
week	444
be	431
for	398
pants	353
5k	319
able	308
old	306
on	302

In [104]:

```
# Create and generate a word cloud image:
num_nsv = 300
text = " ".join(review for review in big_df['NSV Text'].head(num_nsv))
text = text.lower()
wordcloud = WordCloud(background_color="white",
                      width=1600, height=800).generate(text)

# Display the generated image:
plt.figure(figsize=(20, 10), facecolor='w')
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig(f'./figures/nsv_{num_nsv}_wordcloud.png',
            facecolor='w', bbox_inches='tight')
```

In [102]:

```
texts = ["dog cat fish", "dog cat cat", "fish bird", 'bird']
cv = CountVectorizer()
cv_fit = cv.fit_transform(texts)

print(cv.get_feature_names())
print(cv_fit.toarray())
#['bird', 'cat', 'dog', 'fish']
# [[0 1 1 1]
# [0 2 1 0]
# [1 0 0 1]
# [1 0 0 0]]
print(cv_fit.toarray().sum(axis=0))
# [2 3 2 2]
```

```
['bird', 'cat', 'dog', 'fish']
[[0 1 1 1]
 [0 2 1 0]
 [1 0 0 1]
 [1 0 0 0]]
[2 3 2 2]
```

In [103]:

```
# check these out
stop_words = set(stopwords.words('english'))
wordnet_lemmatizer = WordNetLemmatizer()


def tokenize_normalize(tweet):
    only_letters = re.sub("[^a-zA-Z]", " ", tweet)
    tokens = nltk.word_tokenize(only_letters)[2:]
    lower_case = [l.lower() for l in tokens]
    filtered_result = list(filter(lambda l: l not in stop_words, lower_case))
    lemmas = [wordnet_lemmatizer.lemmatize(t) for t in filtered_result]
    return lemmas
```

In [104]:

```
# the sklearn vectorizer scans our corpus, build the vocabulary, and changes text into vecto
vectorizer = CountVectorizer(
    strip_accents='unicode',
    lowercase=True,
    tokenizer=tokenize_normalize,
    min_df=2,
    # you may want to try 2 grams. The vocab will get very large though
    ngram_range=(1, 4),
)
# first learn the vocabulary
vectorizer.fit(big_df['NSV Text'])
```

Out[104]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=2,
        ngram_range=(1, 4), preprocessor=None, stop_words=None,
        strip_accents='unicode', token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=<function tokenize_normalize at 0x7fdb0b4dc730>,
        vocabulary=None)
```

In [105]:

```
# take a peek at the vocabulary learnt. We have the terms and the counts
print(list(vectorizer.vocabulary_.items())[:10])
print("\n vocabulary size {}".format(len(vectorizer.vocabulary_)))
```

```
[('k', 1120), ('min', 1443), ('k min', 1131), ('next', 1534), ('size', 1987), ('jean', 1077)

 vocabulary size 2624
```