Relation Between a Recipe's Cooking Time and Average Rating

Name(s): Gloria Kao

Website Link: https://gkao25.github.io/cooking_time_avg_rating/

Code

In [1]:
```
import pandas as pd
import numpy as np
import os

import plotly.express as px
pd.options.plotting.backend = 'plotly'
```

In [2]:
```
# %pip install statsmodels
# %pip install tabulate
```

In [3]:
```
# load data
recipes = pd.read_csv('food_data/RAW_recipes.csv')
ratings = pd.read_csv('food_data/RAW_interactions.csv')
```

In [4]:
```
recipes.shape
```

Out[4]:
```
(83782, 12)
```

In [5]:
```
recipes.head()
```

Out[5]:

	name	id	minutes	contributor_id	submitted	tags	nutrition	n_steps
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	['60-minutes-or-less', 'time-to-make', 'course...	[138.4, 10.0, 50.0, 3.0, 3.0, 19.0, 6.0]	10
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	['60-minutes-or-less', 'time-to-make', 'cuisin...	[595.1, 46.0, 211.0, 22.0, 13.0, 51.0, 26.0]	12
2	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6
3	millionaire pound cake	286009	120	461724	2008-02-12	['time-to-make', 'course', 'cuisine', 'prepara...	[878.3, 63.0, 326.0, 13.0, 20.0, 123.0, 39.0]	7
4	2000 meatloaf	475785	90	2202916	2012-03-06	['time-to-make', 'course', 'main-ingredient', ...	[267.0, 30.0, 12.0, 12.0, 29.0, 48.0, 2.0]	17

In [6]:
```
# print(recipes.head().to_markdown(index=False))
```

In [7]:
```
ratings.shape
```

Out[7]: (731927, 5)

In [8]:
```
ratings.head()
```

Out[8]:

	user_id	recipe_id	date	rating	review
0	1293707	40893	2011-12-21	5	So simple, so delicious! Great for chilly fall...
1	126440	85009	2010-02-27	5	I made the Mexican topping and took it to bunk...
2	57222	85009	2011-10-01	5	Made the cheddar bacon topping, adding a sprin...
3	124416	120345	2011-08-06	0	Just an observation, so I will not rate. I fo...
4	2000192946	120345	2015-05-10	2	This recipe was OVERLY too sweet. I would sta...

In [9]:

```
# print(ratings.head().to_markdown(index=False))
```

Cleaning and EDA

In [10]:

```
# merging the two datasets together
data = recipes.merge(ratings, left_on='id', right_on='recipe_id')
data.head()
```

Out[10]:

	name	id	minutes	contributor_id	submitted	tags	nutrition	n_steps	s
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	['60-minutes-or-less', 'time-to-make', 'course...	[138.4, 10.0, 50.0, 3.0, 3.0, 19.0, 6.0]	10	the tc ar the
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	['60-minutes-or-less', 'time-to-make', 'cuisin...	[595.1, 46.0, 211.0, 22.0, 13.0, 51.0, 26.0]	12	ove de f'
2	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6	['pr ov deg 'sp 2
3	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6	['pr ov deg 'sp 2
4	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6	['pr ov deg 'sp 2

In [11]:

```
# print(data.head().to_markdown(index=False))
```

In [12]:

```
data[data['rating'] == 0].loc[11, 'review']
```

Out[12]:

```
'We tried it last weekend and it the entire family loved it!  Spicy but not too sp
icy.  It is the best chili I&#039;ve ever tried.  Great Crock Pot recipe and the h
ouse smelled great while cooking!  Awesome!  In the process of cooking our second
batch to take to a Halloween party tonight.'
```

Looking at the example above, some people wrote positive reviews but gave a rating of 0, meaning that the reviewer did not give the receipe a number rating (perhaps because they forgot), so 0 should be replaced with `np.nan` and we can assess its missingness later.

In [13]:
```python
# fill all ratings of 0 with np.nan
data['rating'] = data['rating'].replace(to_replace={0: np.NaN})
```

In [14]:
```python
# find the average rating per recipe as Series and add it to the dataset
avg_rating = data.groupby('recipe_id').mean()['rating']
data = data.merge(avg_rating, left_on='id', right_index=True, suffixes=('', '_avg')
```

In [15]:
```python
# drop the duplicated recipe_id column and rename id and average rating columns
data = data.drop(columns=['recipe_id'])
data = data.rename(columns={'id': 'recipe_id', 'rating_avg': 'avg_rating'})
data.head()
```

Out[15]:

	name	recipe_id	minutes	contributor_id	submitted	tags	nutrition	n_steps
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	['60-minutes-or-less', 'time-to-make', 'course...	[138.4, 10.0, 50.0, 3.0, 3.0, 19.0, 6.0]	10
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	['60-minutes-or-less', 'time-to-make', 'cuisin...	[595.1, 46.0, 211.0, 22.0, 13.0, 51.0, 26.0]	12
2	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6
3	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6
4	412 broccoli casserole	306168	40	50969	2008-05-30	['60-minutes-or-less', 'time-to-make', 'course...	[194.8, 20.0, 6.0, 32.0, 22.0, 36.0, 3.0]	6

In [16]:
```python
print(pd.DataFrame(data.dtypes).to_markdown())
```

	0
name	object
recipe_id	int64
minutes	int64
contributor_id	int64
submitted	object
tags	object
nutrition	object
n_steps	int64
steps	object
description	object
ingredients	object
n_ingredients	int64
user_id	int64
date	object
rating	float64
review	object
avg_rating	float64

In [17]:

```python
# turning tags, nutrition, steps, and ingredients columns into lists

def string_to_list(full_s):
    full_s = full_s.strip('[]')
    l = full_s.split(', ')
    l2 = []
    for s in l:
        s = s.strip("\'")
        l2.append(s)
    return l2

to_change = ['tags', 'nutrition', 'steps', 'ingredients']
for i in to_change:
    data[i] = data[i].apply(string_to_list)
```

In [18]:

```python
# splitting the nutrition column into multiple numerical columns
nutrition_cols = ['calories (#)', 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)',
                     'protein (PDV)', 'saturated fat (PDV)', 'carbohydrates (PDV)']
data[nutrition_cols] = pd.DataFrame(data['nutrition'].tolist())

# drop the old nutrition column
data = data.drop(columns=['nutrition'])
data = data.loc[:, [
    'name', 'recipe_id', 'minutes', 'contributor_id', 'submitted', 'tags',
    'calories (#)', 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)',
    'protein (PDV)', 'saturated fat (PDV)', 'carbohydrates (PDV)',
    'n_steps', 'steps', 'description', 'user_id', 'date', 'rating', 'review', 'avg_
]]
data.head()
```

Out[18]:

	name	recipe_id	minutes	contributor_id	submitted	tags	calories (#)	total fat (PDV)	sug (PD
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	[60-minutes-or-less, time-to-make, course, mai...	138.4	10.0	50
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	[60-minutes-or-less, time-to-make, cuisine, pr...	595.1	46.0	211
2	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6
3	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6
4	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6

5 rows × 21 columns

In [19]:

```
# print(data.head().to_markdown(index=False))
```

In [20]:

```
# changing column data types
data = data.astype({'calories (#)': float, 'total fat (PDV)': float, 'sugar (PDV)':
                    'sodium (PDV)': float, 'protein (PDV)': float, 'saturated fat (
```

```python
                     'carbohydrates (PDV)': float})
print(pd.DataFrame(data.dtypes).to_markdown())
```

```
|                     | 0       |
|:--------------------|:--------|
| name                | object  |
| recipe_id           | int64   |
| minutes             | int64   |
| contributor_id      | int64   |
| submitted           | object  |
| tags                | object  |
| calories (#)        | float64 |
| total fat (PDV)     | float64 |
| sugar (PDV)         | float64 |
| sodium (PDV)        | float64 |
| protein (PDV)       | float64 |
| saturated fat (PDV) | float64 |
| carbohydrates (PDV) | float64 |
| n_steps             | int64   |
| steps               | object  |
| description         | object  |
| user_id             | int64   |
| date                | object  |
| rating              | float64 |
| review              | object  |
| avg_rating          | float64 |
```

In [21]:
```python
tags = pd.DataFrame(data['tags'].to_list(), index=data['recipe_id'])
tags.head()
```

Out[21]:

	0	1	2	3	4	5	6	7
recipe_id								
333281	60-minutes-or-less	time-to-make	course	main-ingredient	preparation	for-large-groups	desserts	lunch
453467	60-minutes-or-less	time-to-make	cuisine	preparation	north-american	for-large-groups	canadian	british-columbian
306168	60-minutes-or-less	time-to-make	course	main-ingredient	preparation	side-dishes	vegetables	easy
306168	60-minutes-or-less	time-to-make	course	main-ingredient	preparation	side-dishes	vegetables	easy
306168	60-minutes-or-less	time-to-make	course	main-ingredient	preparation	side-dishes	vegetables	easy

5 rows × 49 columns

Univariate Analysis

In [22]:
```
data.describe()
```

Out[22]:

	recipe_id	minutes	contributor_id	calories (#)	total fat (PDV)	sugar
count	234428.000000	2.344280e+05	2.344280e+05	234428.000000	234428.000000	234428.(
mean	373164.497406	1.067899e+02	1.239259e+07	419.526876	31.919830	63.8
std	67801.078378	3.285982e+03	1.484920e+08	583.224035	55.392112	210.4
min	275022.000000	0.000000e+00	1.533000e+03	0.000000	0.000000	0.(
25%	314272.000000	2.000000e+01	2.166250e+05	170.700000	8.000000	8.(
50%	363144.500000	3.500000e+01	4.471990e+05	301.100000	20.000000	22.(
75%	424517.750000	6.000000e+01	7.774530e+05	491.100000	39.000000	58.(
max	537716.000000	1.051200e+06	2.002290e+09	45609.000000	3464.000000	30260.(

In [23]:
```
# print(data.describe().to_markdown())
```

In [24]:
```
# some recipes have multiple reviews
# group by recipe id so each value don't get counted multiple times
recipe_grouped = data.groupby('recipe_id').mean().reset_index()
recipe_grouped.head()
```

Out[24]:

	recipe_id	minutes	contributor_id	calories (#)	total fat (PDV)	sugar (PDV)	sodium (PDV)	protein (PDV)	saturated fat (PDV)
0	275022	50.0	531768.0	386.1	34.0	7.0	24.0	41.0	62.0
1	275024	55.0	531768.0	377.1	18.0	208.0	13.0	13.0	30.0
2	275026	45.0	531768.0	326.6	30.0	12.0	27.0	37.0	51.0
3	275030	45.0	666723.0	577.7	53.0	149.0	19.0	14.0	67.0
4	275032	25.0	307114.0	386.9	0.0	347.0	0.0	1.0	0.0

In [25]:
```
# we can see the difference between this table and data.describe
recipe_grouped.describe()
```

Out[25]:

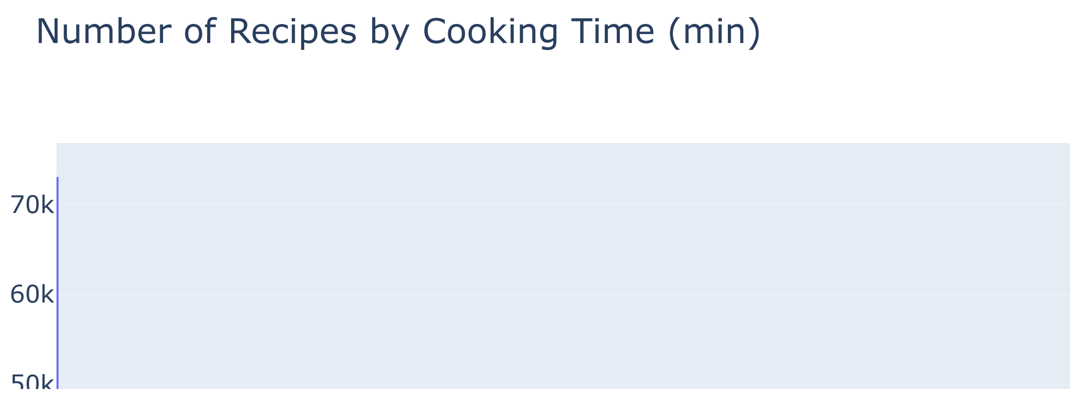
	recipe_id	minutes	contributor_id	calories (#)	total fat (PDV)	sugar (PD
count	83781.000000	8.378100e+04	8.378100e+04	83781.000000	83781.000000	83781.000
mean	381431.198816	1.150318e+02	1.504101e+07	429.921535	32.624712	68.665
std	68715.509257	3.990895e+03	1.655032e+08	636.630335	60.148826	247.239
min	275022.000000	0.000000e+00	1.533000e+03	0.000000	0.000000	0.000
25%	321550.000000	2.000000e+01	2.254260e+05	171.300000	8.000000	9.000
50%	374473.000000	3.500000e+01	4.612830e+05	305.400000	20.000000	23.000
75%	436201.000000	6.500000e+01	8.274370e+05	498.700000	39.000000	61.000
max	537716.000000	1.051200e+06	2.002290e+09	45609.000000	3464.000000	30260.000

In [26]:

```
# print(recipe_grouped.describe().to_markdown())
```

In [27]:

```
# using the grouped data
# graph cooking time
fig = px.histogram(recipe_grouped, x='minutes', title='Number of Recipes by Cooking
fig.show()
```



In [28]:
```
# fig.write_html('cooking_time_outlier_hist.html', include_plotlyjs='cdn')
```

In [29]:
```
# checking for outliers in minutes
data.sort_values('minutes')['minutes']
```

Out[29]:
```
223951          0
223949          0
223950          0
54905           1
25349           1
            ...
107395     129600
107394     259205
106700     288000
109931    1051200
109932    1051200
Name: minutes, Length: 234428, dtype: int64
```

There are some recipes that require more than 10000 minutes, which is over 1 week. We will drop these values from the main dataset and store them in a separate dataframe in case we need them later.

In [30]:
```
long_cook_time = data[data['minutes'] >= 10000]
data = data[data['minutes'] < 10000]
```

In [ ]:
```
# average rating
fig = px.histogram(data, x='avg_rating', title='Average Rating of Recipes')
fig.show()
```

In [32]:
```
# fig.write_html('average_rating_of_recipes_hist.html', include_plotlyjs='cdn')
```

In [33]:
```
data.describe()
```

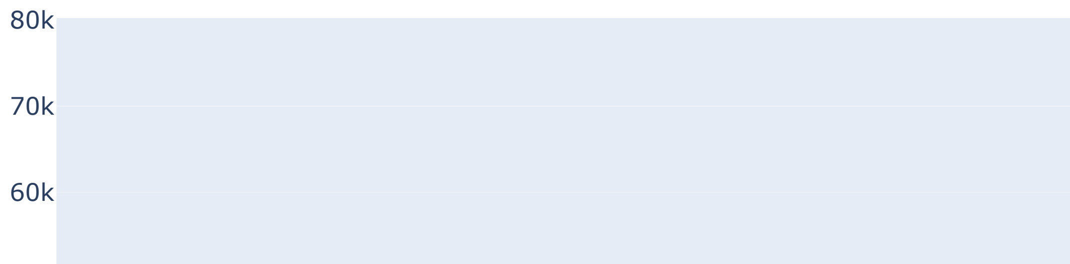
Out[33]:

	recipe_id	minutes	contributor_id	calories (#)	total fat (PDV)	sugai
count	234206.000000	234206.000000	2.342060e+05	234206.000000	234206.000000	234206.
mean	373176.652631	74.357348	1.237835e+07	419.396919	31.940450	63.
std	67816.166289	225.729159	1.483913e+08	583.368761	55.410724	210.
min	275022.000000	0.000000	1.533000e+03	0.000000	0.000000	0.
25%	314244.000000	20.000000	2.158290e+05	170.700000	8.000000	8.
50%	363184.000000	35.000000	4.474870e+05	300.900000	20.000000	22.
75%	424536.000000	60.000000	7.782900e+05	490.900000	39.000000	58.
max	537716.000000	9740.000000	2.002290e+09	45609.000000	3464.000000	30260.

In [34]:
```
# print(data.describe().to_markdown())
```

In [35]:
```
# 75% of the recipes are finished in 60 minutes
minutes_60 = data[data['minutes'] <= 60]
fig = px.histogram(minutes_60, x='avg_rating', title='Average rating for recipes th
fig.show()
```

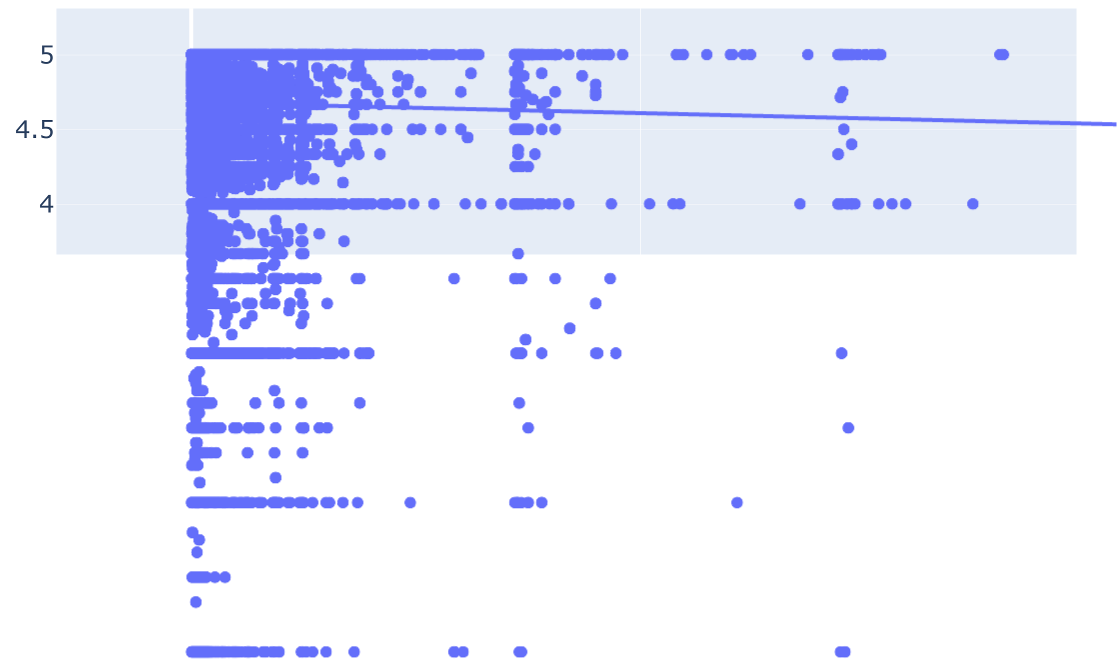
Average rating for recipes that take less than 1hr



Bivariate Analysis

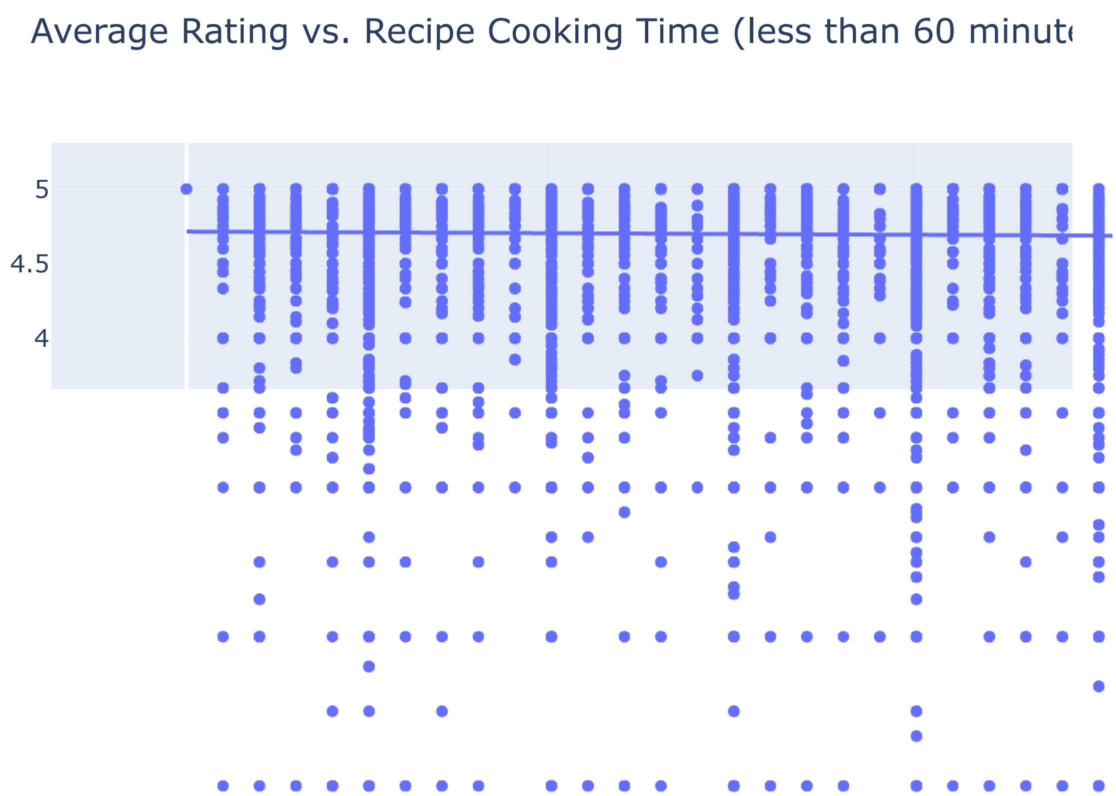
In [36]:
```
fig = px.scatter(data, x='minutes', y='avg_rating', trendline='ols',
                 title='Average Rating vs. Recipe Cooking Time')
fig.show()
```

Average Rating vs. Recipe Cooking Time




```
In [37]: # fig.write_html('avg_rating_cooking_time_scatter.html', include_plotlyjs='cdn')
```

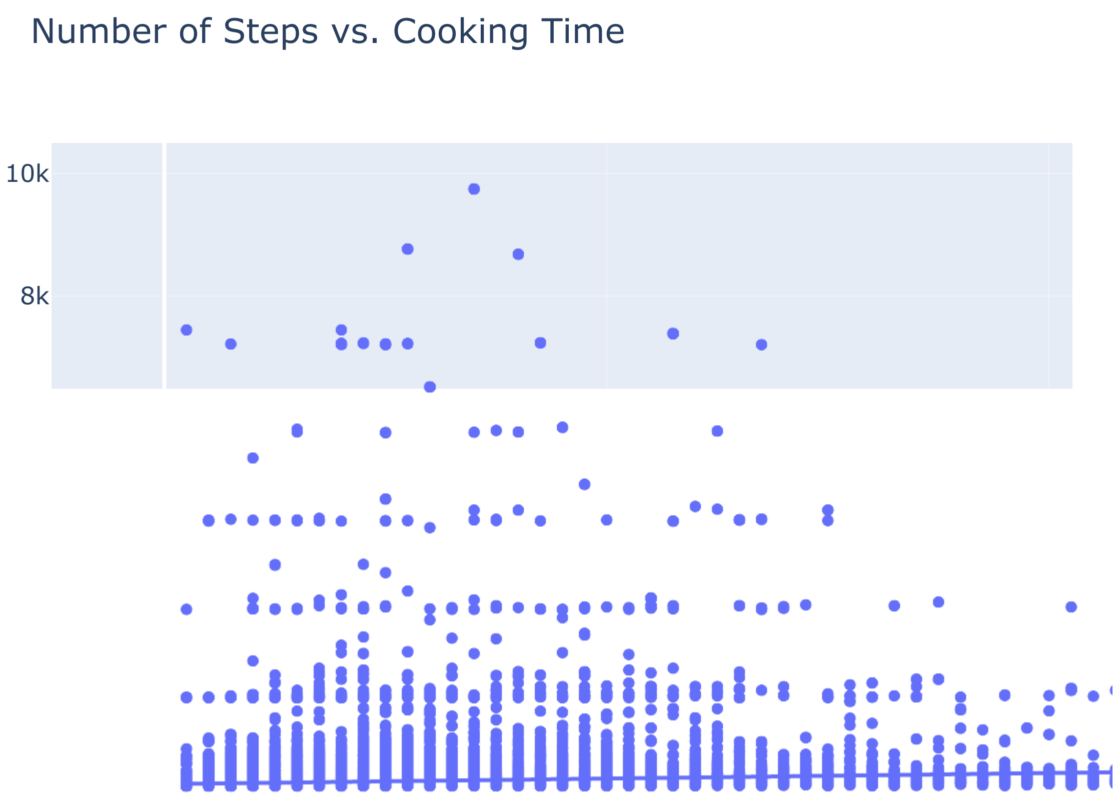
```
In [38]: fig = px.scatter(minutes_60, x='minutes', y='avg_rating', trendline='ols',
                          title='Average Rating vs. Recipe Cooking Time (less than 60 minute
         fig.show()
```



In [39]:
```
# fig.write_html('avg_rating_cooking_time_1h_scatter.html', include_plotlyjs='cdn')
```

There doesn't seem to be any clear correlation.

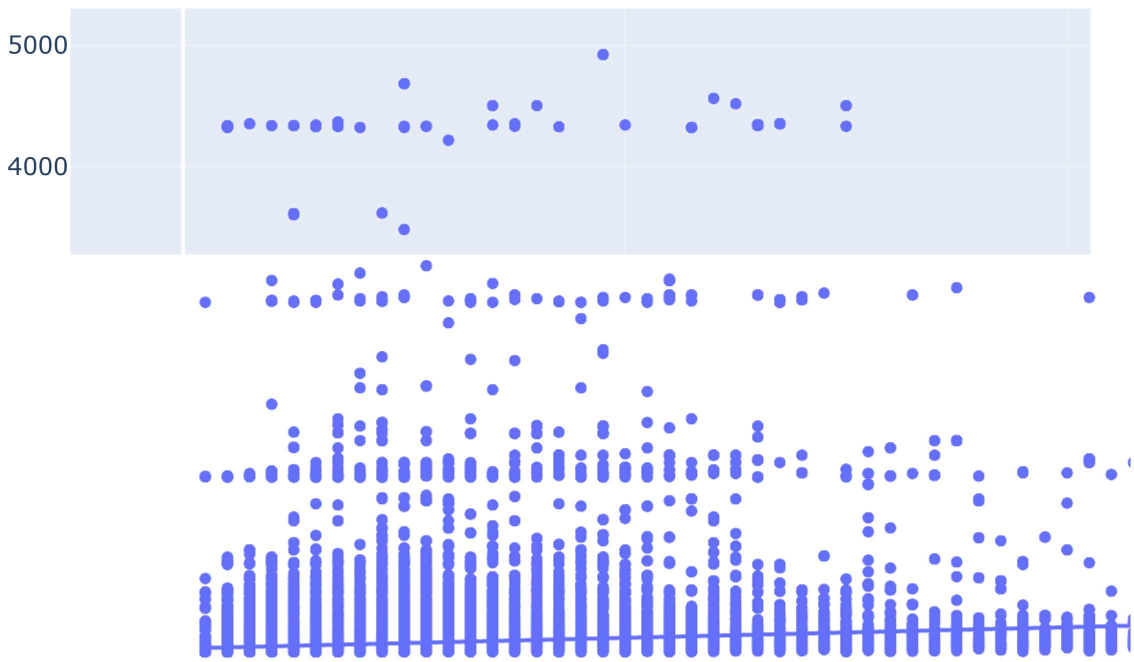
In [40]:
```
# number of steps vs. cooking time
fig = px.scatter(data, x='n_steps', y='minutes', trendline='ols',
                 title='Number of Steps vs. Cooking Time')
fig.show()
```



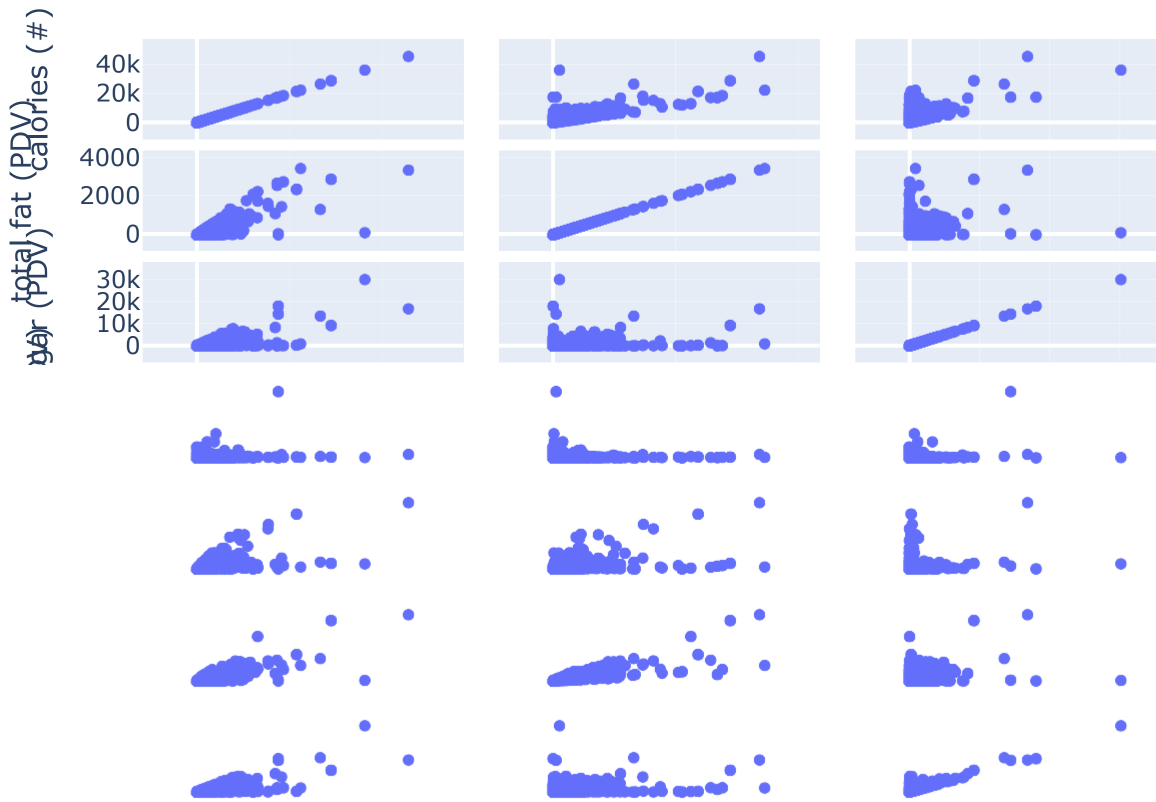
A very slight positive correlation. We want to remove the outliers in cooking time again and see if there is a stronger association.

In [41]:
```
fig = px.scatter(data[data['minutes'] <= 5000], x='n_steps', y='minutes', trendline
                 title='Number of Steps vs. Cooking Time (less than 5000 min)')
fig.show()
```

Number of Steps vs. Cooking Time (less than 5000 min)



In [42]:
```
# looking at other columns that may have associations
# nutrition columns
nutritions = data.iloc[:, range(6, 13)]
fig = px.scatter_matrix(nutritions)
fig.show()
```



In [43]:
```
# fig.write_html('nutritions_scatter_matrix.html', include_plotlyjs='cdn')
```

There seem to be positive correlations between calories/total fat, calories/sugar, calories/protein, calories/saturated fat, calories/carbohydrates, carbohydrates/sugar, saturated fat/total fat, protein/total fat.

Interesting Aggregates

In [44]:
```
data.head()
```

Out[44]:

	name	recipe_id	minutes	contributor_id	submitted	tags	calories (#)	total fat (PDV)	sug (PD
0	1 brownies in the world best ever	333281	40	985201	2008-10-27	[60-minutes-or-less, time-to-make, course, mai...	138.4	10.0	50
1	1 in canada chocolate chip cookies	453467	45	1848091	2011-04-11	[60-minutes-or-less, time-to-make, cuisine, pr...	595.1	46.0	211
2	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6
3	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6
4	412 broccoli casserole	306168	40	50969	2008-05-30	[60-minutes-or-less, time-to-make, course, mai...	194.8	20.0	6

5 rows × 21 columns

In [45]:
```
data.groupby('recipe_id').mean()
```

Out[45]:

recipe_id	minutes	contributor_id	calories (#)	total fat (PDV)	sugar (PDV)	sodium (PDV)	protein (PDV)	saturated fat (PDV)	car
275022	50.0	5.317680e+05	386.1	34.0	7.0	24.0	41.0	62.0	
275024	55.0	5.317680e+05	377.1	18.0	208.0	13.0	13.0	30.0	
275026	45.0	5.317680e+05	326.6	30.0	12.0	27.0	37.0	51.0	
275030	45.0	6.667230e+05	577.7	53.0	149.0	19.0	14.0	67.0	
275032	25.0	3.071140e+05	386.9	0.0	347.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	...	...	
537459	10.0	4.007080e+05	220.7	15.0	49.0	2.0	3.0	30.0	
537485	45.0	2.000379e+09	52.8	3.0	0.0	4.0	1.0	1.0	
537543	55.0	2.001202e+09	1617.0	104.0	213.0	8.0	40.0	203.0	
537671	135.0	2.002199e+09	207.9	12.0	93.0	10.0	6.0	8.0	
537716	40.0	2.001976e+09	407.9	34.0	21.0	49.0	28.0	64.0	

83714 rows × 13 columns

In [46]:

```
# are ther users that makes multiple of reviews?
data['user_id'].value_counts()
```

Out[46]:

```
424680        4934
383346        2522
169430        2336
37449         2261
128473        1979
              ...
2001438558       1
1438886          1
993709           1
421202           1
1803287907       1
Name: user_id, Length: 67207, dtype: int64
```

In [47]:

```
print(pd.DataFrame(data['user_id'].value_counts()).head().to_markdown())
```

```
|        |   user_id |
|-------:|----------:|
| 424680 |      4934 |
| 383346 |      2522 |
| 169430 |      2336 |
|  37449 |      2261 |
| 128473 |      1979 |
```

In [48]:
```
# do these user tend to give higher or lower ratings?
# this table shows us the different ratings a user has given
pivoted = data.pivot_table(values='recipe_id', columns='rating', index='user_id', a
pivoted
```

Out[48]:

rating	1.0	2.0	3.0	4.0	5.0
user_id					
1535	NaN	NaN	3.0	29.0	47.0
1581	NaN	NaN	NaN	NaN	1.0
1634	NaN	NaN	NaN	NaN	2.0
1676	NaN	NaN	NaN	NaN	2.0
1792	NaN	NaN	NaN	NaN	1.0
...	...	...	...	...	...
2002371341	NaN	NaN	NaN	NaN	1.0
2002371755	NaN	NaN	NaN	NaN	1.0
2002371792	NaN	NaN	NaN	1.0	NaN
2002371843	NaN	NaN	NaN	NaN	1.0
2002372464	NaN	NaN	NaN	1.0	NaN

57095 rows × 5 columns

In [49]:
```
# print(pivoted.head().to_markdown())
```

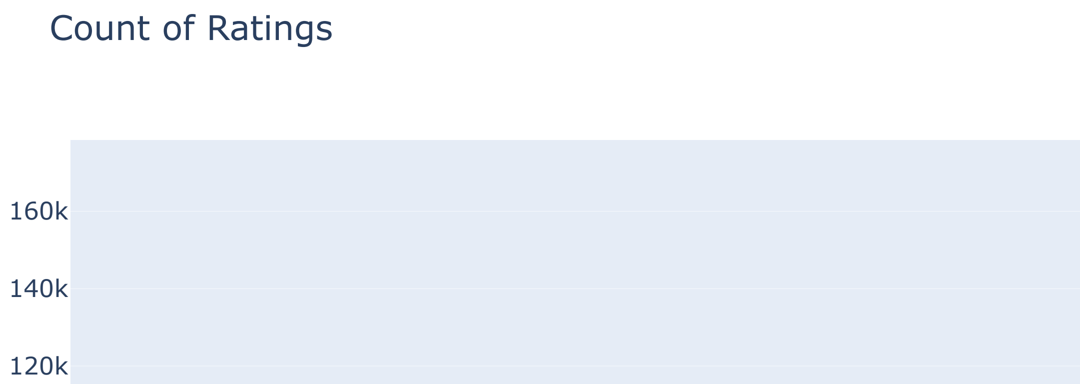
In [50]:
```
# example: the number of ratings user 424680 has given
pivoted.loc[424680]
```

Out[50]:
```
rating
1.0       NaN
2.0       NaN
3.0       2.0
4.0     127.0
5.0    4801.0
Name: 424680, dtype: float64
```

In [51]:
```
# the number of ratings in total
pivoted.sum()
```

Out[51]:
```
rating
1.0      2869.0
2.0      2367.0
3.0      7169.0
4.0     37288.0
5.0    169503.0
dtype: float64
```

In [52]:
```
fig = px.bar(pivoted.sum(), title='Count of Ratings')
fig.show()
```



In [53]:
```
# fig.write_html('rating_count_bar.html', include_plotlyjs='cdn')
```

Assessment of Missingness

NMAR Anlysis:

In [54]:
```
# columns with null values
data.isna().sum()
```

Out[54]:
```
name                          1
recipe_id                     0
minutes                       0
contributor_id                0
submitted                     0
tags                          0
calories (#)                  0
total fat (PDV)               0
sugar (PDV)                   0
sodium (PDV)                  0
protein (PDV)                 0
saturated fat (PDV)           0
carbohydrates (PDV)           0
n_steps                       0
steps                         0
description                 114
user_id                       0
date                          0
rating                    15010
review                       57
avg_rating                 2766
dtype: int64
```

In [55]:
```python
# print(pd.DataFrame(data.isna().sum()).to_markdown())
```

Only one recipe does not have `name`, which I believe is an outlier and therefore MAR. The `description` column has some null values, meaning that some contributors did not write a description for their receipes. Perhaps these values are missing because the recipe is very simple and self-explanatory by its name, therefore needing no further description. The missing `description` are depedent on `name`, so the column is MAR. The `rating` column has many missing values, and the `avg_rating` column as well, which is calculated from `rating`. The null values in `rating` were previously 0, so I believe these values are MAR because if we look at their corresponding `review` they could be positive, and the user might have simply forgotten to give a number rating. In conclusion, I believe there is no column in my dataset that is NMAR.

Missingness Dependency

In [56]:
```python
# analyzing the missingness of avg_rating depending on rating
# their means are very close so we will use KS statistics
data['avg_rating'].mean(), data['rating'].mean()
```

Out[56]:
```
(4.676391936612404, 4.67972499498166)
```

In [57]:
```python
from scipy.stats import ks_2samp
ks_2samp(data.loc[data['rating'].isna(), 'avg_rating'], data.loc[data['rating'].not
```

Out[57]:
```
KstestResult(statistic=0.18427714856762156, pvalue=0.0, statistic_location=5.0, st
atistic_sign=-1)
```

At the confidence level of 5%, we reject the null hypothesis that the two distributions are the same with a p-value of 0.0, and that `avg_rating` is dependent on `rating`.

In [58]:
```
# analyzing the missingness of rating depending on cooking time
# both are numerical columns so we will look at difference in group means
data['rating'].mean(), data['minutes'].mean()
```

Out[58]:
```
(4.67972499498166, 74.35734780492388)
```

In [59]:
```
shuffled = data.copy()
diff_mean = []
for i in range(1000):
    shuffled['rating'] = np.random.permutation(shuffled['rating'])
    missing_mean = shuffled[shuffled['rating'].isna()]['minutes'].mean()
    not_missing_mean = shuffled[shuffled['rating'].notna()]['minutes'].mean()
    diff_mean.append(abs(missing_mean - not_missing_mean))
```

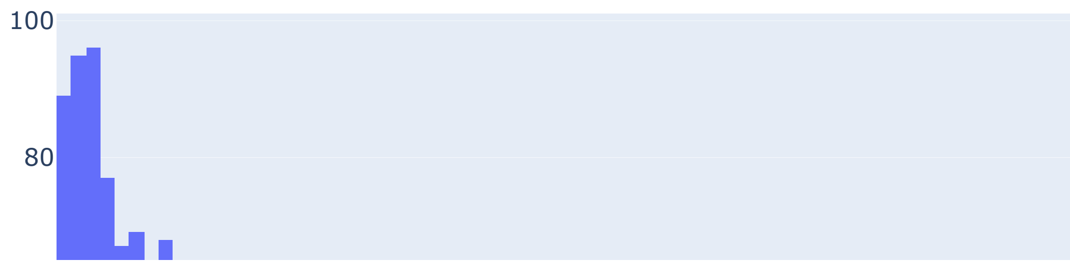
In [60]:
```
observed = abs(data[data['rating'].isna()]['minutes'].mean() - data[data['rating'].
p_value = np.mean(diff_mean >= observed)
p_value
```

Out[60]:
```
0.0
```

The probability that the means of ratings and cooking time (minutes) are more extreme than the observed statistic is 0.0, so we would reject the null hypothesis that the two columns `rating` and `minutes` are dependent on each other.

In [61]:
```
fig = px.histogram(diff_mean, title='Distribution of the Mean Differences')
fig.add_vline(x=observed, line_color='red')
fig.show()
```

Distribution of the Mean Differences



In [62]:
```
# fig.write_html('mean_diff_hist.html', include_plotlyjs='cdn')
```

In [63]:
```
data_copy = data.copy()
data_copy = data_copy[data_copy['minutes'] <= 120]
data_copy['rating missing'] = data_copy['rating'].isna()
px.histogram(data_copy, x='minutes', color='rating missing', histnorm='probability'
```

Hypothesis Testing

Questions: What is the relationship between the cooking time and average rating of recipes?

Null Hypothesis: There is no relationship between cooking time and average rating of recipes.

Alternative Hypothesis: The average rating of recipes is dependent on the cooking time.

In [64]:
```
# permutation test
shuffled = data.copy()
diff_mean = []
for i in range(1000):
    shuffled['avg_rating'] = np.random.permutation(shuffled['avg_rating'])
    missing_mean = shuffled[shuffled['avg_rating'].isna()]['minutes'].mean()
    not_missing_mean = shuffled[shuffled['avg_rating'].notna()]['minutes'].mean()
    diff_mean.append(abs(missing_mean - not_missing_mean))
```

In [65]:
```
observed = abs(data[data['avg_rating'].isna()]['minutes'].mean() - data[data['avg_r
p_value = np.mean(diff_mean >= observed)
```

```
p_value
```

Out[65]: 0.0

In [66]:

```
fig = px.histogram(diff_mean, title='Distribution of the Mean Differences')
fig.add_vline(x=observed, line_color='red')
fig.show()
```

Distribution of the Mean Differences

In [67]:

```
# fig.write_html('hypo_test_mean_diff_hist.html', include_plotlyjs='cdn')
```

The probability that the means of ratings and cooking time (minutes) are more extreme than the observed statistic is 0.0, so we would reject the null hypothesis that there are no relationships between the `avg_rating` and `minutes` columns.

In [ ]: