Are 'Best' Recipes Really the Best?

Author: Aaron Rasin

Website Link: https://aaron-m-r.github.io/Recipes/ (https://aaron-m-r.github.io/Recipes/)

Import packages

In [1]:

```
import pandas as pd
import numpy as np
import os
import csv
import random
from pathlib import Path
from tqdm.notebook import tqdm

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

Introduction

Our research question is **Are recipes that claim to be the 'best' (or high quality) actually have higher ratings?**. However, along the way, we'd also like to take a look at a few other areas on interest; Are more nutritious recipes rated higher? Have recipes gotten more complicated (i.e. longer time/more steps) over time?

Cleaning and EDA

Data Wrangling

We first clean the data and extract the relevant information. This involves merging the recipe dataset with the rating dataset (left merge) so that no rating/comment is without a recipe. Then, we rename a few columns to clarify their meaning. Next, we have to deal with comments that give a rating of zero. This is actually not technically a rating, but just a comment. Ratings are on a scale of 1 to 5, so 0 represents absence of a rating, and thus shouldn't be counted as a rating of 0 out of 5, but should be counted as missing (NaN). Now we can calculate the average rating of each recipe. This operation creates duplicate columns, which we immediately remove. Let's take a look at our dataset so far.

Data Extraction

Now that we have our single DataFrame, let's make the features of interest easily accessible. First, notice that the dates in the date column are strings, which is tedious to perform operations on. Instead, we convert all of the dates from string to datetime which makes them much easier to deal with later. Second, we also notice that columns like tags, nutrition, steps and ingredients look like lists, but are actually strings. This is also quite annoying to use in data analysis, so we convert those strings to lists in order to access their items easier. Since we know we'll want to investigate nutrition later, we separate each item in the list of nutrition facts into its own column, where each value is a float. Let's look at what the nutrition columns look like.

Read in csv files as dataframes (recipes and interactionsa)

In [2]:

```
path1 = Path('Best Recipes?/food_data') / 'RAW_interactions.csv'
raw_reviews = pd.read_csv(path1)

path2 = Path('Best Recipes?/food_data') / 'RAW_recipes.csv'
raw_recipes = pd.read_csv(path2)
```

Merge recipe dataframe with review dataframe

In [3]:

```
recipes = raw_recipes.merge(raw_reviews, left_on='id', right_on='recipe_id', how='left')
```

Rename columns

```
In [4]: recipes = recipes.rename(columns = {'submitted': 'recipe_date', 'date': 'review_date'})
```

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

Out[5]:

	name	id	minutes	contributor_id	recipe_date	tags	nutrition	n_steps	steps	description	ingredients	n_ingredients	
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	['heat the oven to 350f and arrange the rack i...	these are the most; chocolatey, moist, rich, d...	['bittersweet chocolate', 'unsalted butter', '...	9	38
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	['pre-heat oven the 350 degrees f', 'in a mixi...	this is the recipe that we use at my school ca...	['white sugar', 'brown sugar', 'salt', 'margar...	11	42
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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	119
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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	76

Replace zeros with NaN

We do this because the only review options are 1-5, not 0. A rating of 0 indicates no rating at all, so we replace it with a null value.

```
In [6]: recipes = recipes.replace(0, np.nan)
```

Calculate each recipe's average rating

Using the comments, we can calculate the average rating per recipe.

In [7]:
```
recipes.head()
```

Out[7]:

	name	id	minutes	contributor_id	recipe_date	tags	nutrition	n_steps	steps	description	ingredients	n_ingredients	
0	1 brownies in the world best ever	333281	40.0	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	['heat the oven to 350f and arrange the rack i...	these are the most; chocolatey, moist, rich, d...	['bittersweet chocolate', 'unsalted butter', '...	9	38
1	1 in canada chocolate chip cookies	453467	45.0	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	['pre-heat oven the 350 degrees f', 'in a mixi...	this is the recipe that we use at my school ca...	['white sugar', 'brown sugar', 'salt', 'margar...	11	42
2	412 broccoli casserole	306168	40.0	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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	2
3	412 broccoli casserole	306168	40.0	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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	119
4	412 broccoli casserole	306168	40.0	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	['preheat oven to 350 degrees', 'spray a 2 qua...	since there are already 411 recipes for brocco...	['frozen broccoli cuts', 'cream of chicken sou...	9	76

In [8]:

```python
# Make a dataframe of recipe ratings, grouped by recipe
mean_rating = recipes.groupby('id').mean()[['rating']]

# Make a dataframe of everything except ratings, grouped by recipe
unique_recipes = recipes.groupby('id').first().drop(columns='rating')

# Merge ratings with everything else to get one row per recipe
recipes = unique_recipes.merge(mean_rating, left_index=True, right_index=True, how='inner')

recipes.head()
```

Out[8]:

	name	minutes	contributor_id	recipe_date	tags	nutrition	n_steps	steps	description	ingredients	n_ingredients	u
id												
275022	impossible macaroni and cheese pie	50.0	531768	2008-01-01	['60-minutes-or-less', 'time-to-make', 'course...	[386.1, 34.0, 7.0, 24.0, 41.0, 62.0, 8.0]	11	['heat oven to 400 degrees fahrenheit', 'greas...	one of my mom's favorite bisquick recipes. thi...	['cheddar cheese', 'macaroni', 'milk', 'eggs',...	7	24
275024	impossible rhubarb pie	55.0	531768	2008-01-01	['60-minutes-or-less', 'time-to-make', 'course...	[377.1, 18.0, 208.0, 13.0, 13.0, 30.0, 20.0]	6	['heat oven to 375 degrees', 'grease 10" pan ,...	a childhood favorite of mine. my mom loved it ...	['rhubarb', 'eggs', 'bisquick', 'butter', 'sal...	8	17
275026	impossible seafood pie	45.0	531768	2008-01-01	['60-minutes-or-less', 'time-to-make', 'course...	[326.6, 30.0, 12.0, 27.0, 37.0, 51.0, 5.0]	7	['preheat oven to 400f', 'lightly grease large...	this is an oldie but a goodie. mom's stand by ...	['frozen crabmeat', 'sharp cheddar cheese', 'c...	9	55
275030	paula deen s caramel apple cheesecake	45.0	666723	2008-01-01	['60-minutes-or-less', 'time-to-make', 'course...	[577.7, 53.0, 149.0, 19.0, 14.0, 67.0, 21.0]	11	['preheat oven to 350f', 'reserve 3 / 4 cup ap...	thank you paula deen! hubby just happened to ...	['apple pie filling', 'graham cracker crust', ...	9	15
275032	midori poached pears	25.0	307114	2008-01-01	['lactose', '30-minutes-or-less', 'time-to-mak...	[386.9, 0.0, 347.0, 0.0, 1.0, 0.0, 33.0]	8	['bring midori , sugar , spices , rinds and wa...	the green colour looks fabulous and the taste ...	['midori melon liqueur', 'water', 'caster suga...	9	30

Drop unnecessary columns

In [9]:

```python
recipes = recipes.drop(columns=['recipe_id', 'review'])
```

Convert dates from string to datetime

In [10]:

```python
recipes['recipe_date'] = pd.to_datetime(recipes['recipe_date'])
```

Convert any strings of lists to lists

In [11]:

```python
def str2list(lst):
    lst = lst.strip('][').split(', ')
    return [item.replace("'", "") for item in lst]
```

In [12]:

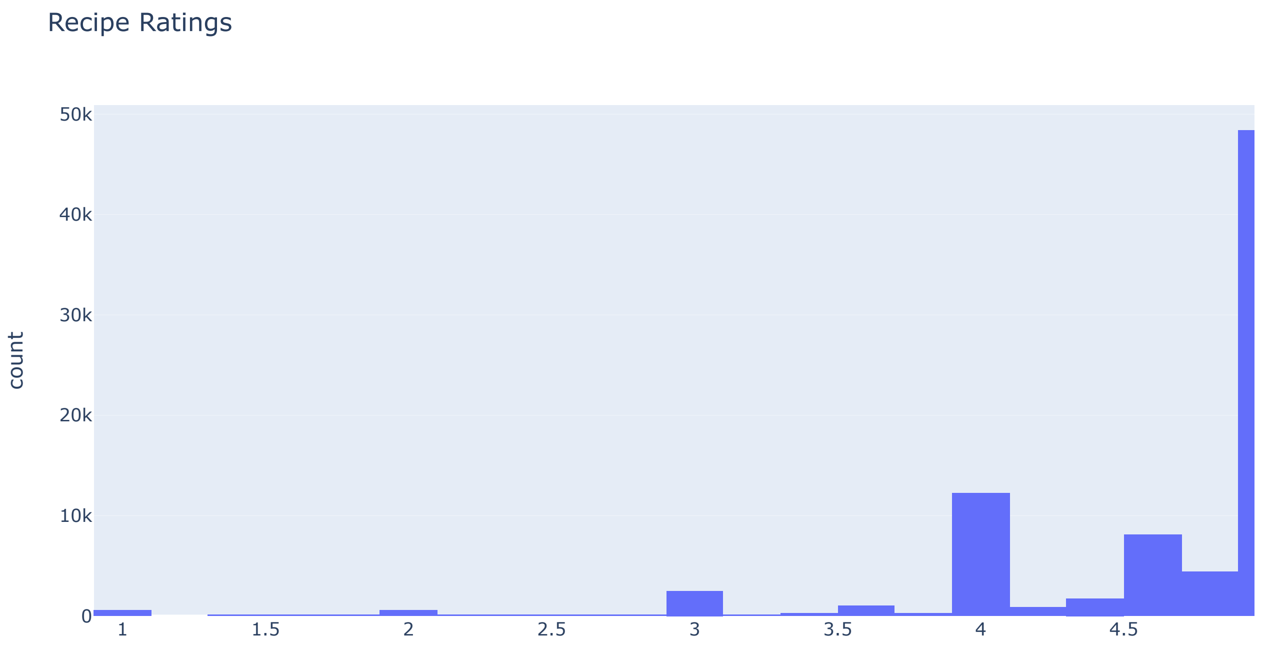
```python
for column in ['tags', 'nutrition', 'steps']:
    recipes[column] = recipes[column].apply(str2list)
```

```
In [13]: nutrition_facts = ['calories', 'total fat (PDV)', 'sugar (PDV)', 'sodium (PDV)', 'protein (PDV)', 'satur

         # Split nutrition list into 7 nutrition columns
         recipes[nutrition_facts] = pd.DataFrame(recipes.nutrition.tolist(), index= recipes.index)

         # Convert nutrition facts from strings to flaots
         for fact in nutrition_facts:
             recipes[fact] = recipes[fact].astype(float)
```

Univariate Analysis

Histogram of ratings

```
In [14]: fig = px.histogram(recipes, x="rating", nbins=30, title = 'Recipe Ratings')
         fig.write_html('ratings_histogram.html', include_plotlyjs='cdn')
         fig.show()
```

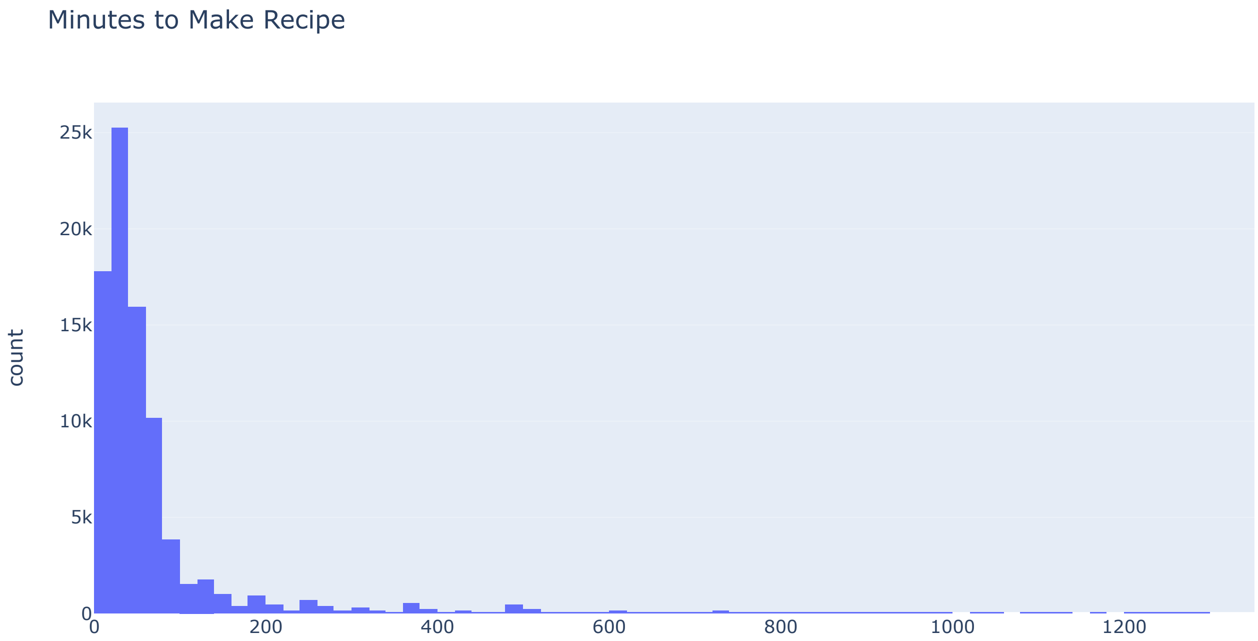


Histogram of minutes

```
In [15]: # Exclude recipes that take over 24 hours to make
         under_12hrs = recipes[recipes['minutes']<24*60]
```

```
In [16]: # Proportion of recipes not included
         1 - under_12hrs.shape[0]/recipes.shape[0]

Out[16]: 0.0074717719796615345
```

In [17]:
```
fig = px.histogram(under_12hrs, x="minutes", nbins=100, title='Minutes to Make Recipe')
fig.write_html('minutes_histogram.html', include_plotlyjs='cdn')
fig.show()
```

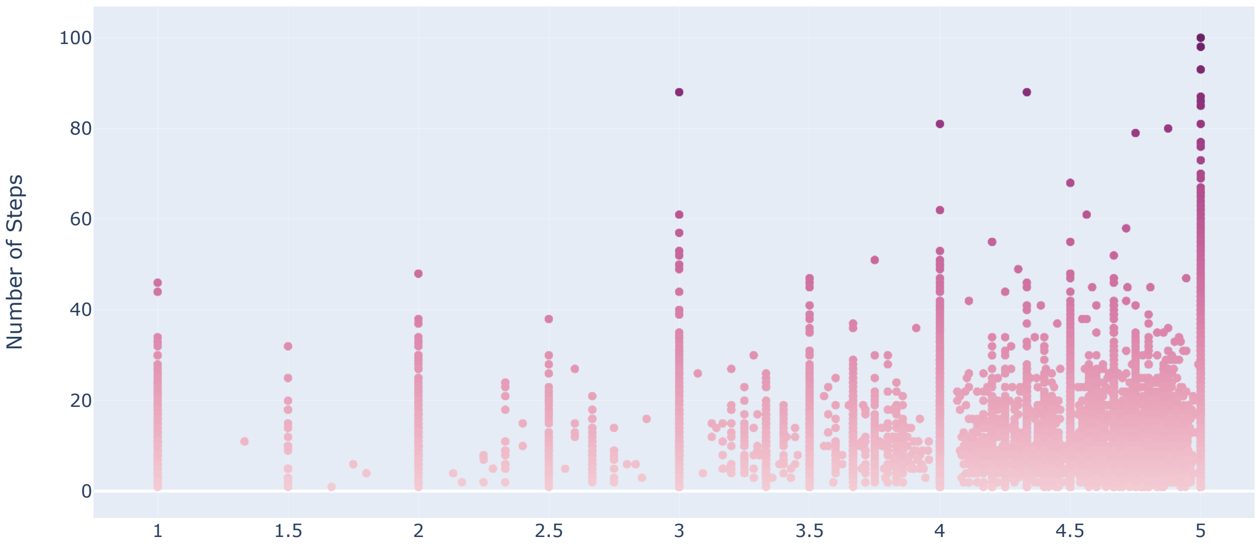


Bivariate Analysis

Number of steps and average rating

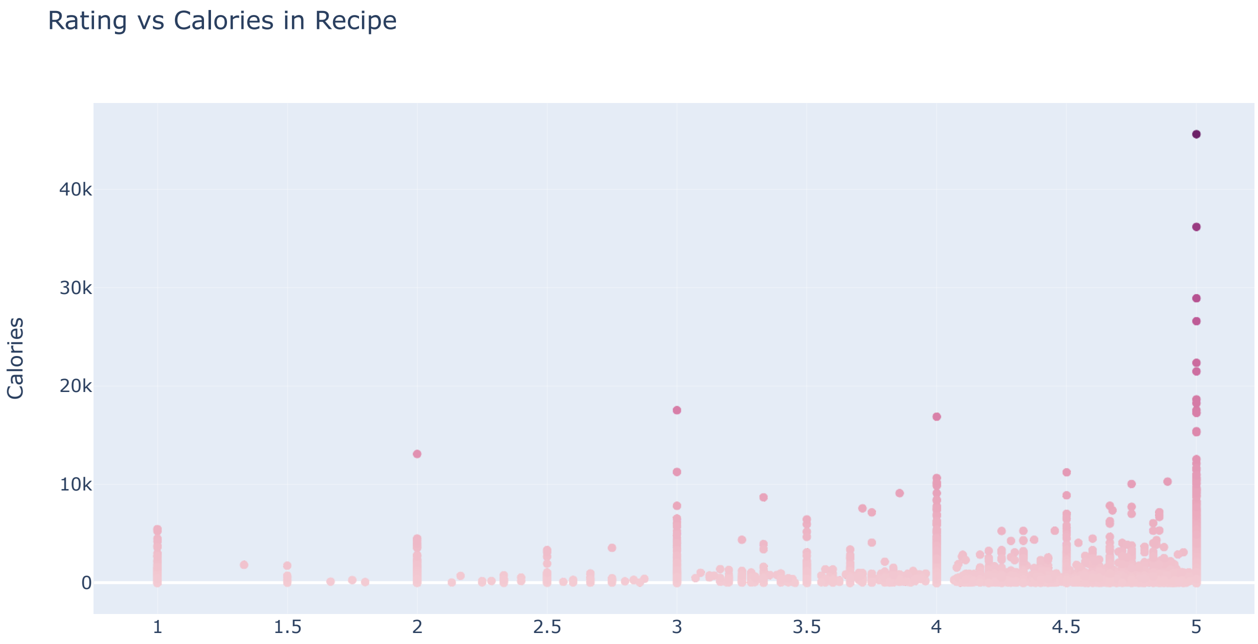
In [18]:
```
fig = px.scatter(y=recipes['n_steps'], x=recipes['rating'], title='Rating vs Number of Steps in Recipe'
                 color=recipes['n_steps'], color_continuous_scale='magenta',
                 labels = {'x': 'Rating', 'y': 'Number of Steps', 'color': 'Steps'})
fig.write_html('scatter_nsteps_rating.html', include_plotlyjs='cdn')
fig.show()
```

Rating vs Number of Steps in Recipe



Calories and average rating

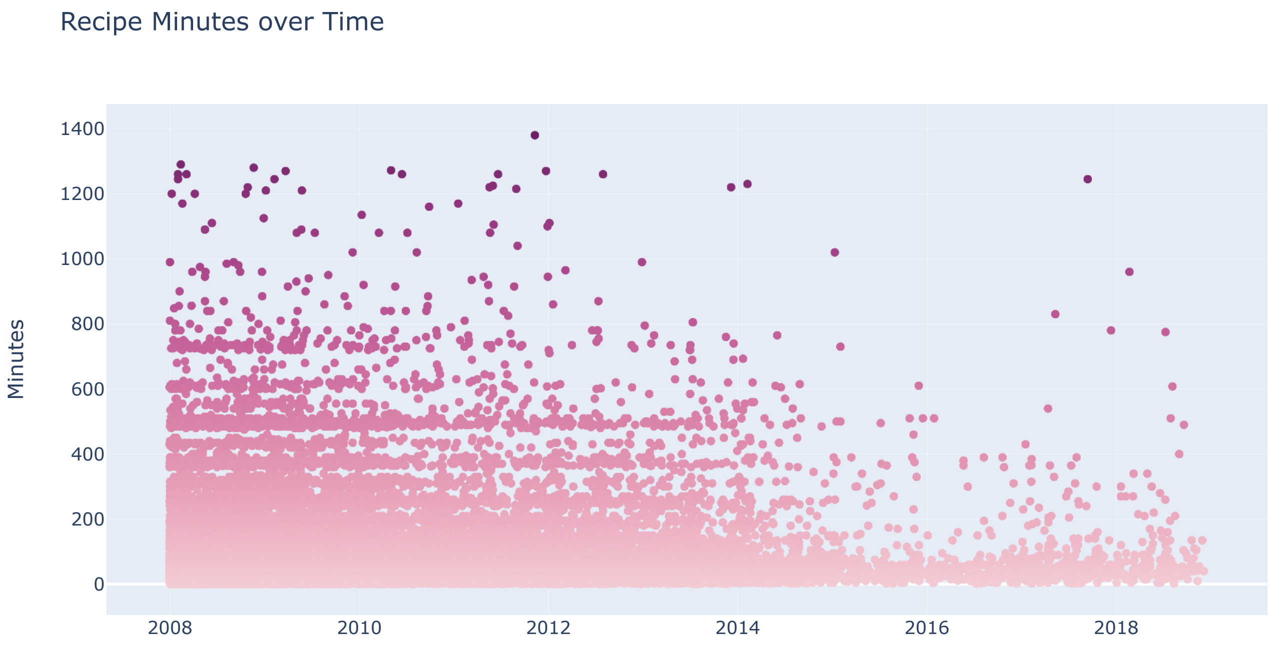
In [19]:
```
fig = px.scatter(y=recipes['calories'], x=recipes['rating'], title='Rating vs Calories in Recipe',
                  color=recipes['calories'], color_continuous_scale='magenta',
                 labels = {'x': 'Rating', 'y': 'Calories', 'color': 'Calories'})
fig.write_html('scatter_calories_rating.html', include_plotlyjs='cdn')
fig.show()
```


Rating vs Calories in Recipe

Aggregate Analysis

In [20]:
```
# Remove recipe time (minutes) outliers
no_minute_outliers = recipes[recipes['minutes'] < 24*60]
```

In [21]:
```
fig = px.scatter(y=no_minute_outliers['minutes'], x=no_minute_outliers['recipe_date'],
                 title='Recipe Minutes over Time', color=no_minute_outliers['minutes'], color_continuou
                labels = {'x': 'Date', 'y': 'Minutes', 'color': 'Minutes'})
fig.write_html('scatter_minutes_date.html', include_plotlyjs='cdn')
fig.show()
```

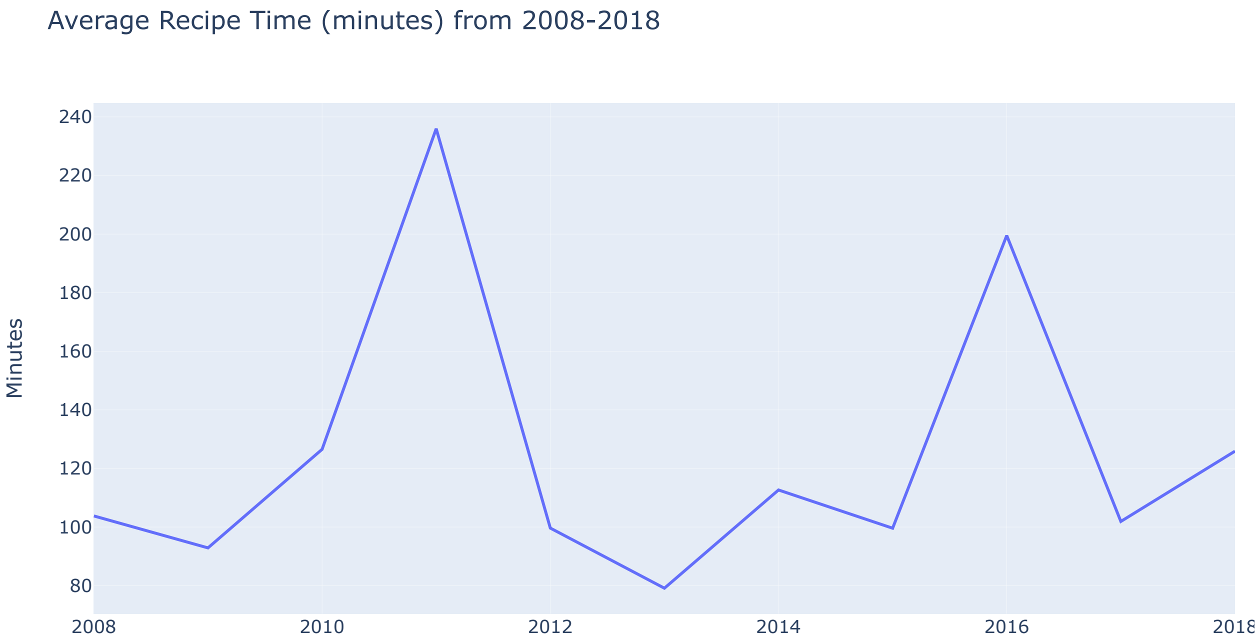


In [22]:
```
# Add 'year' column to recipes dataframe
with_year = recipes.assign(year = recipes['recipe_date'].apply(lambda x: x.year))

# Group by year to get average statistics
by_year = with_year.groupby('year').mean()

# Group by year to see how many recipes per year
recipes_per_year = with_year.groupby('year').count()[['minutes']].rename(columns={'minutes': 'recipes_p

# Combine number of recipes and average recipe time
aggregated = by_year[['minutes']].merge(recipes_per_year, left_index=True, right_index=True)

aggregated
```

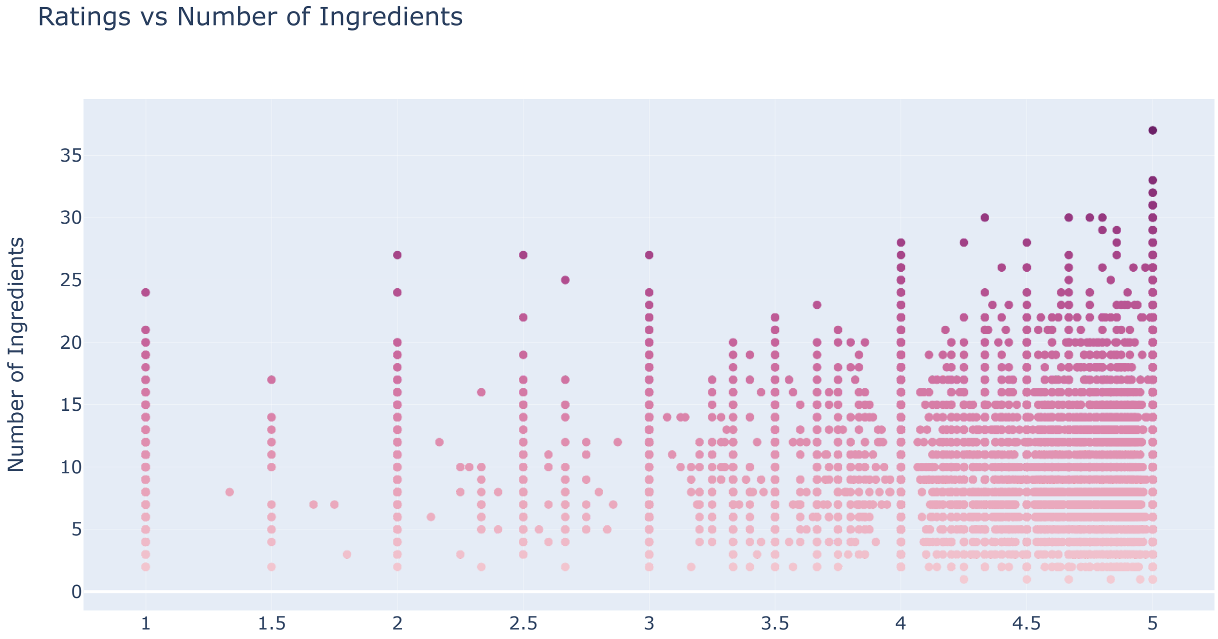
Out[22]:

	minutes	recipes_per_year
year		
2008	103.788707	30744
2009	92.882512	22547
2010	126.525290	11902
2011	235.983098	7573
2012	99.650858	5187
2013	79.122890	3792
2014	112.650143	1049
2015	99.601307	306
2016	199.509804	204
2017	101.913194	288
2018	125.894180	189

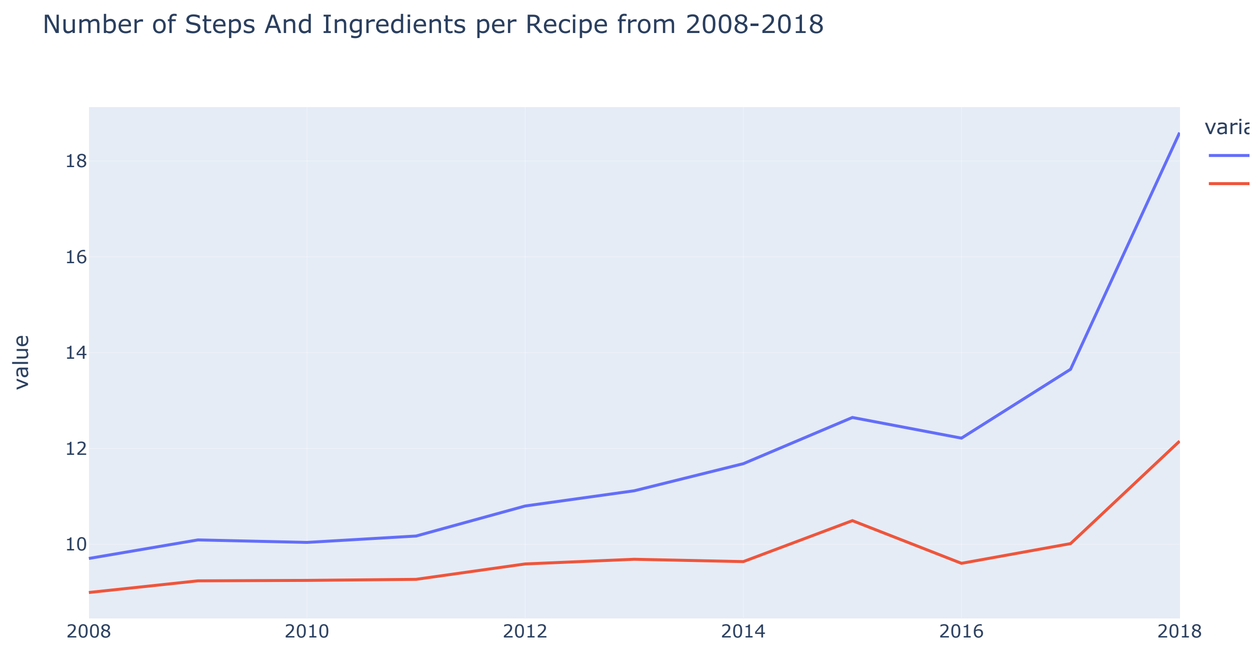
```
In [23]: fig = px.line(by_year, x=aggregated.index, y=['minutes'], title='Average Recipe Time (minutes) from 200
                      labels = {'value': 'Minutes'})
         fig.write_html('line_minutes_year.html', include_plotlyjs='cdn')
         fig.show()
```



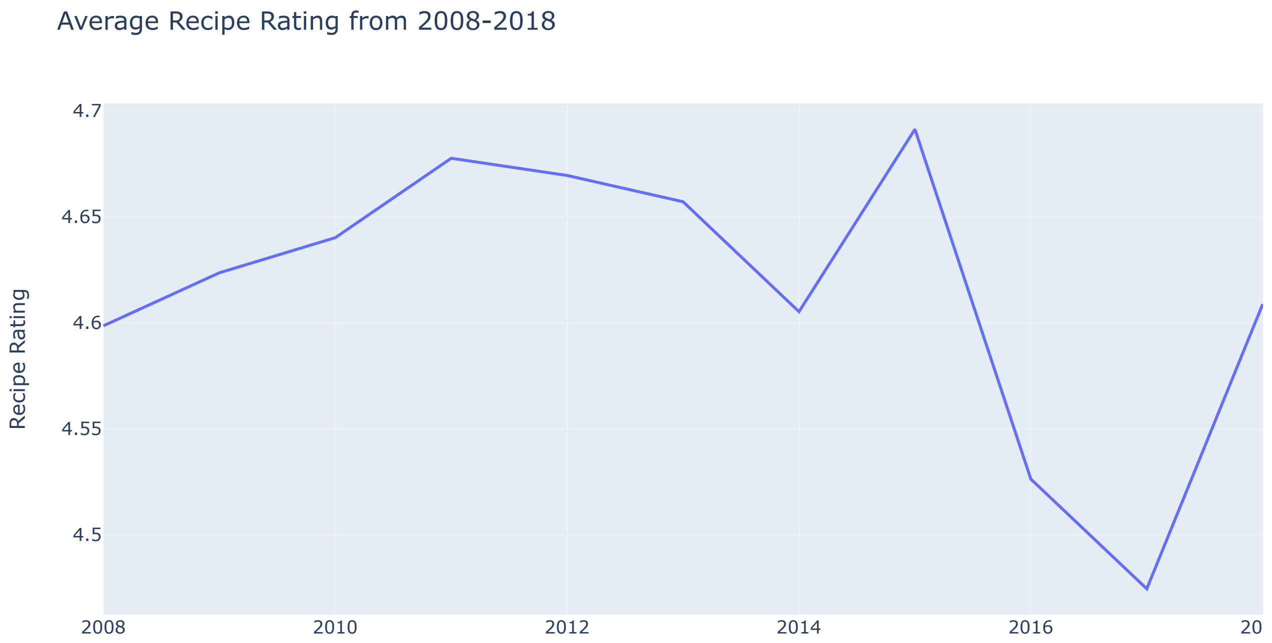
```
In [24]: fig = px.scatter(y=recipes['n_ingredients'], x=recipes['rating'], title='Ratings vs Number of Ingredien
                          color=recipes['n_ingredients'], color_continuous_scale='magenta',
                         labels={'x': 'Recipe Rating', 'y':'Number of Ingredients', 'color': 'Ingredients'})
         fig.write_html('scatter_ningrendients_rating.html', include_plotlyjs='cdn')
         fig.show()
```



```
In [25]: fig = px.line(by_year, x=by_year.index, y=['n_steps', 'n_ingredients'], title='Number of Steps And Ingr
         fig.write_html('line_steps_ingredients', include_plotlyjs='cdn')
         fig.show()
```



```
In [26]: fig = px.line(by_year, x=by_year.index, y=['rating'], title='Average Recipe Rating from 2008-2018',
                       labels = {'value': 'Recipe Rating'})
         fig.show()
```



Assessment of Missingness

In [27]:

```
recipes.isnull().sum()
```

Out[27]:

```
name                      1
minutes                   1
contributor_id            0
recipe_date               0
tags                      0
nutrition                 0
n_steps                   0
steps                     0
description              70
ingredients               0
n_ingredients             0
user_id                   1
review_date               1
rating                 2609
calories                  0
total fat (PDV)           0
sugar (PDV)               0
sodium (PDV)              0
protein (PDV)             0
saturated fat (PDV)       0
carbohydrates (PDV)       0
dtype: int64
```

Permutation Testing

In order to test our theories, we run permutations test with 10,000 trials and a p-value of .05. If less than 5% of the trials in the permutation test contain test statistics equal to or greater than the observed test statistic, we can reject the null hypothesis.

In [28]:

```
# Add column for rating missingness
recipes['rating_missing'] = recipes['rating'].isna()
```

In [29]:

```
# Function for calculating observed statistic (difference of means)
def observe(df, independent, dependent, absolute=True):
    grouped = df.groupby(dependent).mean()
    true = grouped.loc[True][independent]
    false = grouped.loc[False][independent]
    if absolute:
        return abs(true-false)
    return true-false
```

In [30]:

```
# Function for running a permutation test on two features
def permutation_test(df, independent, dependent, repetitions, absolute=True):

    # Test statistic observed from original data
    observed = observe(df, independent, dependent, absolute=True)

    # Empty array for storing results
    results = np.array([])

    # Make a copy of dataframe in order to keep original dataframe unshuffled
    copy = df.copy()

    # Run trials (n=repetitions)
    for _ in tqdm(np.arange(repetitions)):

        # Shuffle groups of recipes
        copy[independent] = np.random.permutation(copy[independent])

        # Calculate and record test statistic
        results = np.append(results, observe(copy, independent, dependent, absolute=True))

    # Return results array if desired (used for plotting)
    return results, observed
```

In [31]:
```
missing = recipes.groupby('rating_missing').mean()
missing
```

Out[31]:

	minutes	contributor_id	n_steps	n_ingredients	user_id	rating	calories	total fat (PDV)	sugar (PDV)	sodium (PDV)	
rating_missing											
False	111.378234	1.322609e+07	10.058948	9.206103	7.245279e+07	4.625363	427.191020	32.399566	67.814261	28.952595	33
True	228.719049	7.150255e+07	11.551936	9.460330	5.070442e+08	NaN	515.050326	39.651974	95.113837	28.602146	34

Rating NMAR Test on Protein Content

Null hypothesis: the distribution of ingredients per recipe without a rating is the **same** as the distribution of the protein per recipe with a rating

Alternative hypothesis: the distribution of ingredients per recipe without a rating is the **different** from the distribution of the protein per recipe with a rating

Test statistic: the absolute difference between protein content per recipe including rating and fat content per recipe missing rating.

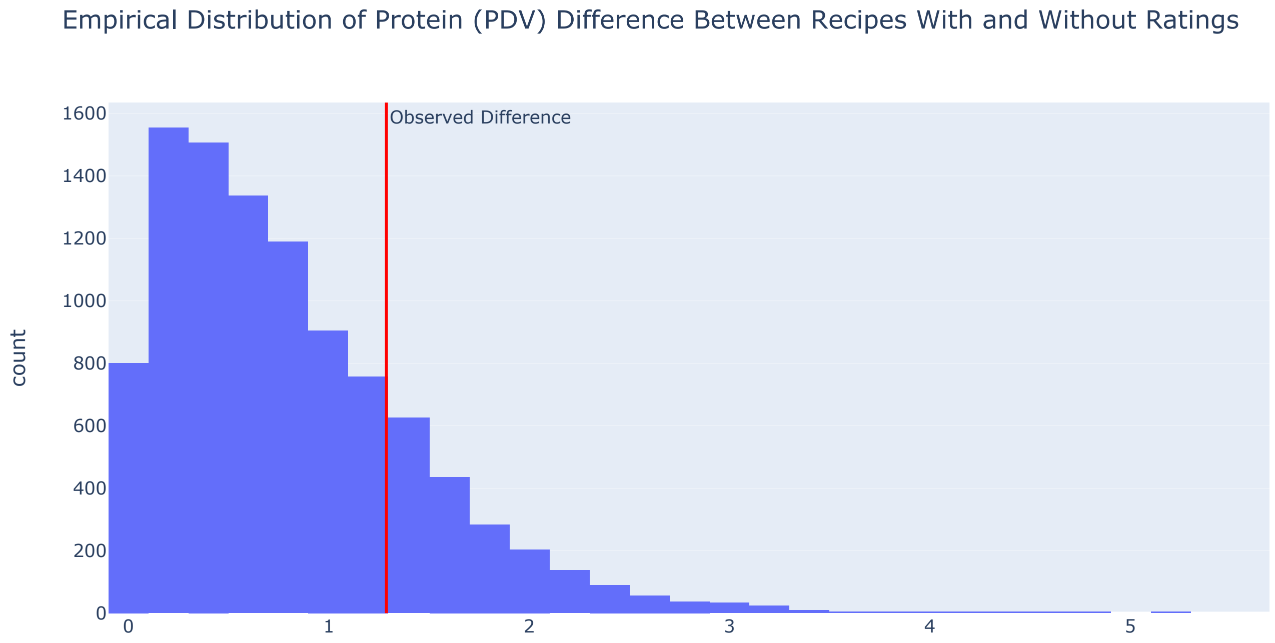
In [32]:
```
# Run permutation test on protein content and rating missingness
protein_res, protein_obs = permutation_test(recipes, 'protein (PDV)', 'rating_missing', 10_000)
```

```
100%                                        10000/10000 [04:13<00:00, 41.46it/s]
```

In [33]:
```
# Calculate p value
p_value = (protein_res>=protein_obs).mean()
p_value
```

Out[33]:
```
0.1988
```

In [34]:
```
fig = px.histogram(protein_res, x=0, nbins=50,
                   title='Empirical Distribution of Protein (PDV) Difference Between Recipes With and W
                   labels={'0': 'Difference in Recipe Protein Content (PDV)'})
fig.add_vline(x=protein_obs, line_color='red', annotation_text='Observed Difference')
fig.write_html('protein_missing_histogram.html', include_plotlyjs='cdn')
fig.show()
```



Rating NMAR Test on Total Fat Content

Null hypothesis: the distribution of fat per recipe without a rating is the **same** as the distribution of the fat per recipe with a rating

Alternative hypothesis: the distribution of fat per recipe without a rating is the **different** from the distribution of the fat per recipe with a rating

Test statistic: the absolute difference between average fat per recipe including rating and fat per recipe missing rating.

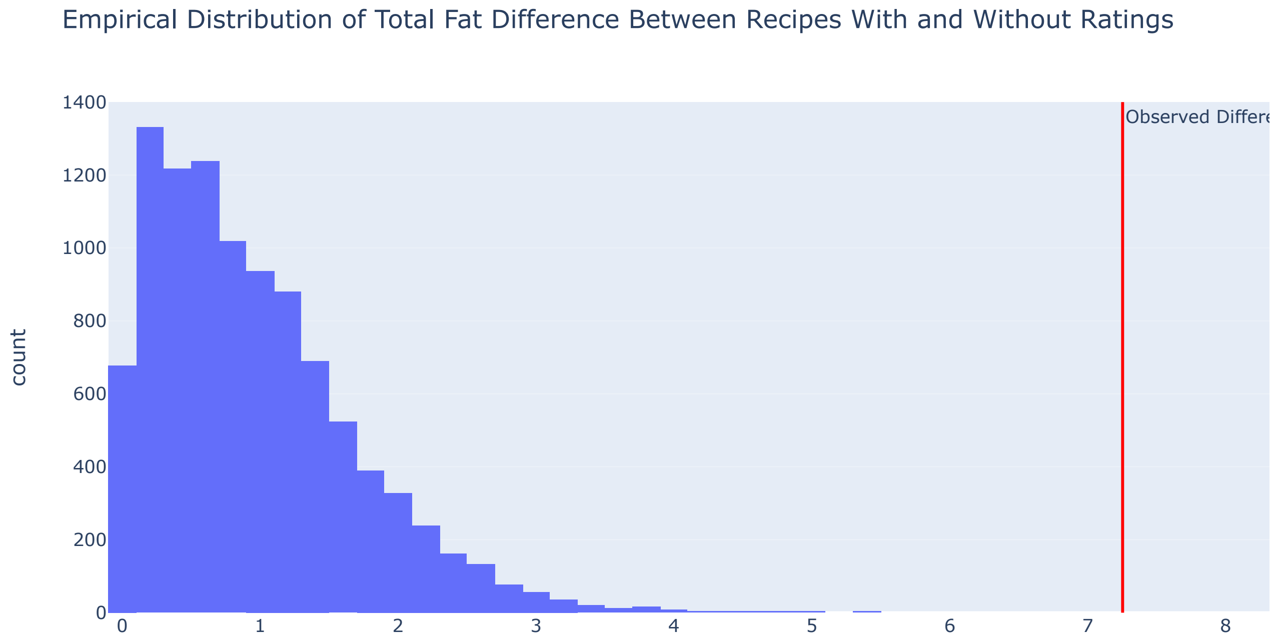
In [35]:
```
# Run permutation test on fat content and rating missingness
fat_res, fat_obs = permutation_test(recipes, 'total fat (PDV)', 'rating_missing', 10_000)
```

```
100%                                        10000/10000 [04:23<00:00, 31.19it/s]
```

In [36]:
```
# Calculate p value
p_value = (fat_res>=fat_obs).mean()
p_value
```

Out[36]:
```
0.0
```

In [37]:
```
fig = px.histogram(fat_res, x=0, nbins=50,
                   title='Empirical Distribution of Total Fat Difference Between Recipes With and Witho
                   labels={'0': 'Difference in Total Fat (PDV)'})
fig.add_vline(x=fat_obs, line_color='red', annotation_text='Observed Difference')
fig.write_html('fat_missing_histogram.html', include_plotlyjs='cdn')
fig.show()
```



Hypothesis Testing

Now, we'd like to return to investigate our initial question. Basically, does a recipe bragging how good it is actually tell you anything about the quality of the recipe itself? We'd like to know if recipes with words related to quality or authenticity have higher ratings. First, we have to put the recipe titles back into the dataframe since we lost them during the groupby. Then we add a boolean column 'best' with True for recipes with some superlative or indication of high quality (IHQ) in the title.

In [38]:
```
# Remove any recipes without a title
ihq = recipes.copy()[recipes['name'].isna()==False]

# Add 'best' column
ihq['best'] = ihq['name'].apply(lambda x: 'best' in x  or
                                           'most' in x or
                                           'amazing' in x or
                                           'delicious' in x or
                                           'yummy' in x or
                                           'perfect' in x or
                                           'ultimate' in x or
                                            'good' in x or
                                           'love' in x or
                                           'authentic' in x)
```

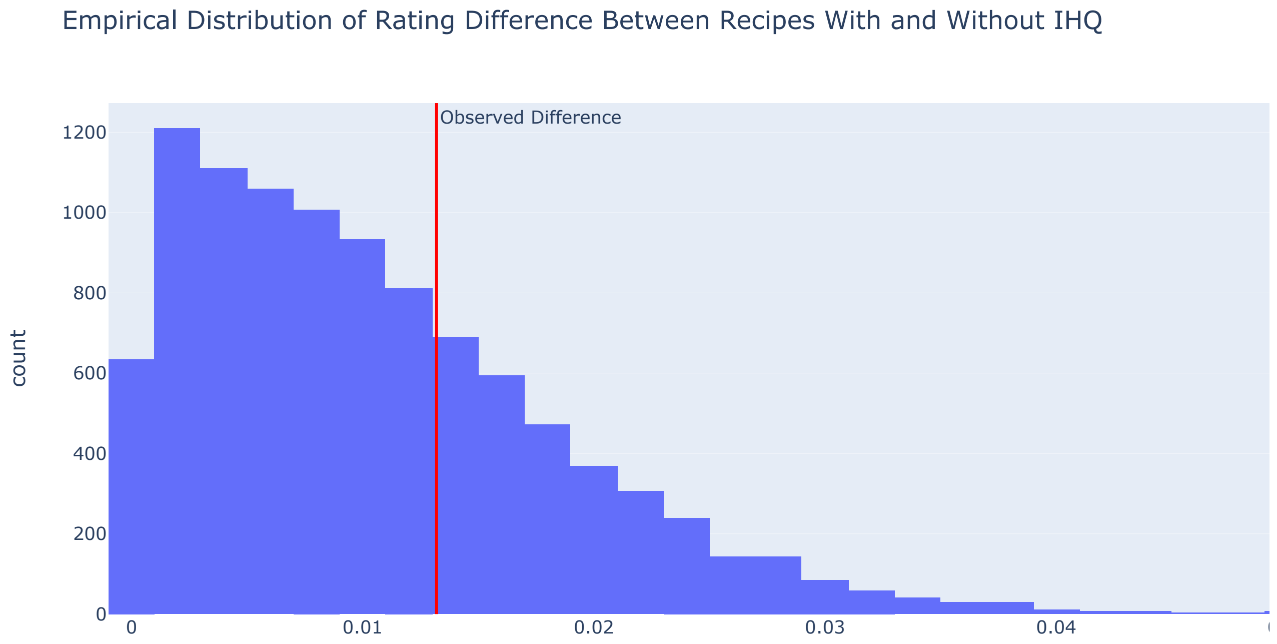
In [39]:
```
# Permutation test to see if IHQ recipes and no IHQ recipes come from same rating distribution (observe
best_res, best_obs = permutation_test(ihq, 'rating', 'best', 10_000, absolute=False)
```

```
100%                                        10000/10000 [04:38<00:00, 40.79it/s]
```

In [40]:
```
# Calculate p value
p_value = (best_res>=best_obs).mean()
p_value
```

Out[40]:
```
0.316
```

In [41]:
```
fig = px.histogram(best_res, x=0, nbins=50,
                   title='Empirical Distribution of Rating Difference Between Recipes With and Without
                   labels={'0': 'Difference in Rating'})
fig.add_vline(x=best_obs, line_color='red', annotation_text='Observed Difference')
fig.write_html('best_rating_histogram.html', include_plotlyjs='cdn')
fig.show()
```


Empirical Distribution of Rating Difference Between Recipes With and Without IHQ

In [ ]: