BookReview_EDA

December 5, 2020

[1]:
```
import gdown

url = 'https://drive.google.com/uc?id=1UPZiTughL3iDtPwreoUs_SX-LfVktrI3'
output = 'BX-CSV-Dump.zip'
gdown.download(url, output, quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=1UPZiTughL3iDtPwreoUs_SX-LfVktrI3
To: /content/BX-CSV-Dump.zip
26.1MB [00:00, 59.7MB/s]
```

[1]:
```
'BX-CSV-Dump.zip'
```

[2]:
```
! unzip BX-CSV-Dump.zip
```

```
Archive:  BX-CSV-Dump.zip
  inflating: BX-Book-Ratings.csv
  inflating: BX-Books.csv
  inflating: BX-Users.csv
```

1 Book Crossing EDA

[3]:
```
%matplotlib inline

import scipy
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler, MinMaxScaler

sns.set()
palette = sns.color_palette("icefire")

plt.style.use('ggplot')
```

```
sns.set_context("talk")
```

1.1 Loading, Cleaning and Merging the Dataset

Reading the csv files

[4]:
```
users = pd.read_csv(
    '/content/BX-Users.csv',
    names=['user_id', 'location', 'age'],
    sep=';',
    skiprows=1,
    encoding='ISO-8859-1',
    low_memory=False,
    error_bad_lines=False
)
users
```

[4]:
```
        user_id                               location   age
0             1                     nyc, new york, usa   NaN
1             2               stockton, california, usa  18.0
2             3      moscow, yukon territory, russia   NaN
3             4               porto, v.n.gaia, portugal  17.0
4             5  farnborough, hants, united kingdom   NaN
...         ...                                    ...   ...
278853   278854                portland, oregon, usa   NaN
278854   278855  tacoma, washington, united kingdom  50.0
278855   278856            brampton, ontario, canada   NaN
278856   278857            knoxville, tennessee, usa   NaN
278857   278858                 dublin, n/a, ireland   NaN

[278858 rows x 3 columns]
```

parse the datatypes properly

[5]:
```
users.dtypes
```

[5]:
```
user_id       int64
location     object
age         float64
dtype: object
```

[6]:
```
users.describe().T
```

[6]:
```
            count           mean           std  ...       50%        75%
max
user_id  278858.0  139429.500000  80499.515020  ...  139429.5  209143.75
278858.0
age      168096.0      34.751434     14.428097  ...      32.0      44.00
244.0
```

```
[2 rows x 8 columns]
```

age cannot be 244 ! so let's fix that

[7]:
```
users.loc[(users.age > 100) | (users.age < 5), 'age'] = np.nan
users.age = users.age.fillna(users.age.mean())
```

[8]:
```
users['age'] = users['age'].astype(np.uint8)
```

[9]:
```
users['age'].describe()
```

[9]:
```
count    278858.000000
mean         34.446733
std          10.551712
min           5.000000
25%          29.000000
50%          34.000000
75%          35.000000
max         100.000000
Name: age, dtype: float64
```

[10]:
```
users.isna().sum()
```

[10]:
```
user_id     0
location    0
age         0
dtype: int64
```

[11]:
```
books = pd.read_csv(
    '/content/BX-Books.csv',
    names=['isbn', 'book_title', 'book_author', 'year_of_publication',␣
 ↪'publisher', 'img_s', 'img_m', 'img_l'],
    sep=';',
    skiprows=1,
    encoding='ISO-8859-1',
    low_memory=False,
    error_bad_lines=False
)
books
```

[11]:
```
              isbn  ...                                              img_l
0       0195153448  ...  http://images.amazon.com/images/P/0195153448.0...
1       0002005018  ...  http://images.amazon.com/images/P/0002005018.0...
2       0060973129  ...  http://images.amazon.com/images/P/0060973129.0...
3       0374157065  ...  http://images.amazon.com/images/P/0374157065.0...
4       0393045218  ...  http://images.amazon.com/images/P/0393045218.0...
...            ...  ...                                                ...
271374  0440400988  ...  http://images.amazon.com/images/P/0440400988.0...
271375  0525447644  ...  http://images.amazon.com/images/P/0525447644.0...
271376  006008667X  ...  http://images.amazon.com/images/P/006008667X.0...
271377  0192126040  ...  http://images.amazon.com/images/P/0192126040.0...
271378  0767409752  ...  http://images.amazon.com/images/P/0767409752.0...
```

```
[271379 rows x 8 columns]
```

parse the data types properly

[12]:
```
books.dtypes
```

[12]:
```
isbn                 object
book_title           object
book_author          object
year_of_publication  object
publisher            object
img_s                object
img_m                object
img_l                object
dtype: object
```

drop ['img_s', 'img_m', 'img_l'] since they are not useful for us

[13]:
```
books = books.drop(['img_s', 'img_m', 'img_l'], axis=1)
```

year_of_publication should be a integer

[14]:
```
books['year_of_publication'] = pd.to_numeric(books['year_of_publication'],␣
↪errors='coerce')
```

[15]:
```
books.loc[(books['year_of_publication'] == 0) | (books['year_of_publication'] >␣
↪2008), 'year_of_publication' ] = np.nan
books['year_of_publication'] = books['year_of_publication'].
↪fillna(round(books['year_of_publication'].mean()))
books['year_of_publication'] = pd.to_numeric(books['year_of_publication'],␣
↪downcast='unsigned')
```

[16]:
```
books.isna().sum()
```

[16]:
```
isbn                 0
book_title           0
book_author          1
year_of_publication  0
publisher            2
dtype: int64
```

[17]:
```
books = books.dropna()
```

[18]:
```
books.describe().T
```

[18]:
```
                      count         mean       std  ...     50%     75%
max
year_of_publication  271376.0  1993.692427  8.248715  ...  1995.0  2000.0
2008.0

[1 rows x 8 columns]
```

[19]:
```
ratings = pd.read_csv(
    '/content/BX-Book-Ratings.csv',
    names=['user_id', 'isbn', 'book_rating'],
    sep=';',
    skiprows=1,
    encoding='ISO-8859-1',
    low_memory=False,
    error_bad_lines=False
)
ratings
```

[19]:

	user_id	isbn	book_rating
0	276725	034545104X	0
1	276726	0155061224	5
2	276727	0446520802	0
3	276729	052165615X	3
4	276729	0521795028	6
...	...	...	...
1149775	276704	1563526298	9
1149776	276706	0679447156	0
1149777	276709	0515107662	10
1149778	276721	0590442449	10
1149779	276723	05162443314	8

[1149780 rows x 3 columns]

[20]:
```
ratings['book_rating'] = ratings['book_rating'].astype(np.uint8)
```

[21]:
```
ratings.dtypes
```

[21]:
```
user_id         int64
isbn           object
book_rating     uint8
dtype: object
```

[22]:
```
ratings.isna().sum()
```

[22]:
```
user_id        0
isbn           0
book_rating    0
dtype: int64
```

[23]:
```
ratings.describe().T.astype(np.int32)
```

[23]:

	count	mean	std	min	25%	50%	75%	max
user_id	1149780	140386	80562	2	70345	141010	211028	278854
book_rating	1149780	2	3	0	0	0	7	10

Join the three datasets based on `user_id` and `isbn`

[24]:
```
temp = pd.merge(users, ratings, on='user_id')
temp = pd.merge(temp, books, on='isbn')
```

```
dataset = temp.copy()
```

[25]:
```
dataset
```

[25]:
```
         user_id  ...                 publisher
0              2  ...  Oxford University Press
1              8  ...    HarperFlamingo Canada
2          11400  ...    HarperFlamingo Canada
3          11676  ...    HarperFlamingo Canada
4          41385  ...    HarperFlamingo Canada
...          ...  ...                       ...
1031167   278851  ...     Simon &amp;amp; Schuster
1031168   278851  ...           Broadway Books
1031169   278851  ...          Lone Star Books
1031170   278851  ...               Kqed Books
1031171   278851  ...  American Map Corporation

[1031172 rows x 9 columns]
```

Split the location into `city`, `state` and `country` and replacing missing location details with just `n/a`

[26]:
```
location = dataset['location'].str.split(', ', n=2, expand=True)
location.columns = ['city', 'state', 'country']
location = location.fillna('n/a')
```

[27]:
```
dataset['city'] = location['city'] ; dataset['state'] = location['state'] ;␣
↪dataset['country'] = location['country']
```

[28]:
```
dataset = dataset.drop(['location'], axis=1)
```

[29]:
```
dataset.describe().T.astype(np.int32)
```

[29]:
```
                      count    mean    std  ...     50%     75%     max
user_id             1031172  140594  80524  ...  141210  211426  278854
age                 1031172      36     10  ...      34      41     100
book_rating         1031172       2      3  ...       0       7      10
year_of_publication 1031172    1995      7  ...    1997    2001    2008

[4 rows x 8 columns]
```

[30]:
```
dataset.isna().sum()
```

[30]:
```
user_id               0
age                   0
isbn                  0
book_rating           0
book_title            0
book_author           0
year_of_publication   0
publisher             0
city                  0
```

```
state                0
country              0
dtype: int64
```

[31]:
```
dataset.shape
```

[31]:
```
(1031172, 11)
```

[32]:
```
dataset.dtypes
```

[32]:
```
user_id               int64
age                   uint8
isbn                 object
book_rating           uint8
book_title           object
book_author          object
year_of_publication  uint16
publisher            object
city                 object
state                object
country              object
dtype: object
```

This will be the final dataset we will be working with !

[33]:
```
dataset.head(5)
```

[33]:
```
   user_id  age        isbn  ...      city       state country
0        2   18  0195153448  ...  stockton  california     usa
1        8   34  0002005018  ...   timmins     ontario  canada
2    11400   49  0002005018  ...    ottawa     ontario  canada
3    11676   34  0002005018  ...       n/a         n/a     n/a
4    41385   34  0002005018  ...   sudbury     ontario  canada

[5 rows x 11 columns]
```

[34]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1031172 entries, 0 to 1031171
Data columns (total 11 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   user_id              1031172 non-null  int64
 1   age                  1031172 non-null  uint8
 2   isbn                 1031172 non-null  object
 3   book_rating          1031172 non-null  uint8
 4   book_title           1031172 non-null  object
 5   book_author          1031172 non-null  object
 6   year_of_publication  1031172 non-null  uint16
 7   publisher            1031172 non-null  object
 8   city                 1031172 non-null  object
```

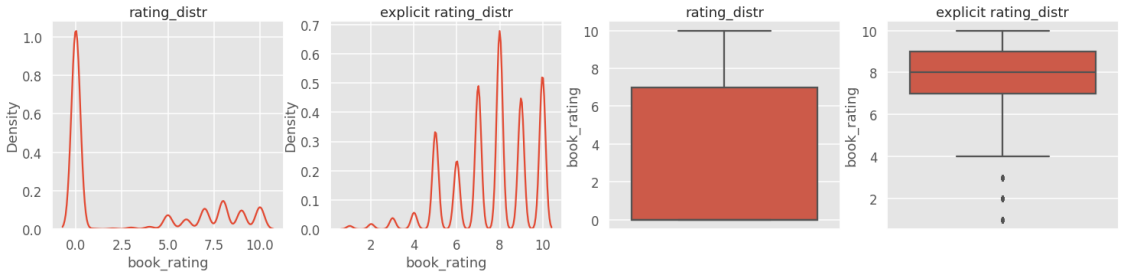
```
 9   state                1031172 non-null  object
 10  country              1031172 non-null  object
dtypes: int64(1), object(7), uint16(1), uint8(2)
memory usage: 74.7+ MB
```

[35]:
```
cleaned_data = dataset.copy()
```

[53]:
```
# dataset = cleaned_data.copy()
```

1.2 Analyzing the Feature Space

[55]:
```
f, axes = plt.subplots(ncols = 4, figsize=(25, 5))
sns.kdeplot(x="book_rating", data=dataset, ax=axes[0]).set_title('rating_distr')
sns.kdeplot(x="book_rating", data=dataset[dataset['book_rating'] != 0],␣
 ↪ax=axes[1]).set_title('explicit rating_distr')
sns.boxplot(y="book_rating", data=dataset, orient='v', ax=axes[2]).
 ↪set_title('rating_distr')
sns.boxplot(y="book_rating", data=dataset[dataset['book_rating'] != 0],␣
 ↪orient='v', ax=axes[3]).set_title('explicit rating_distr')
plt.show()
```



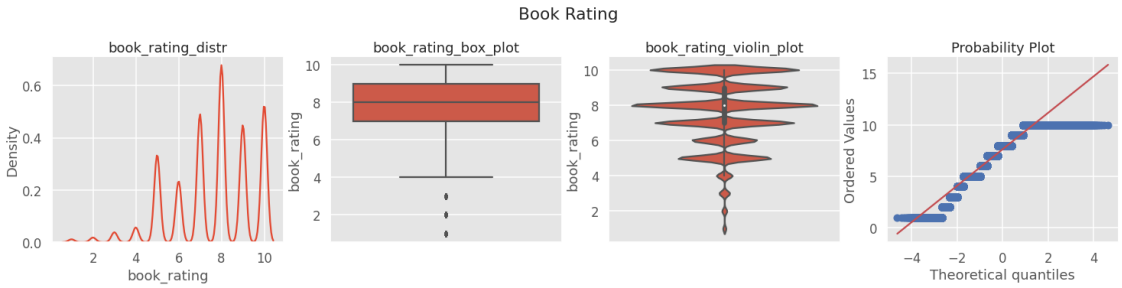
We can remove 0 ratings, since these are unrated, and why would someone rate a book as 0 ?

[56]:
```
dataset = dataset[dataset['book_rating'] != 0]
```

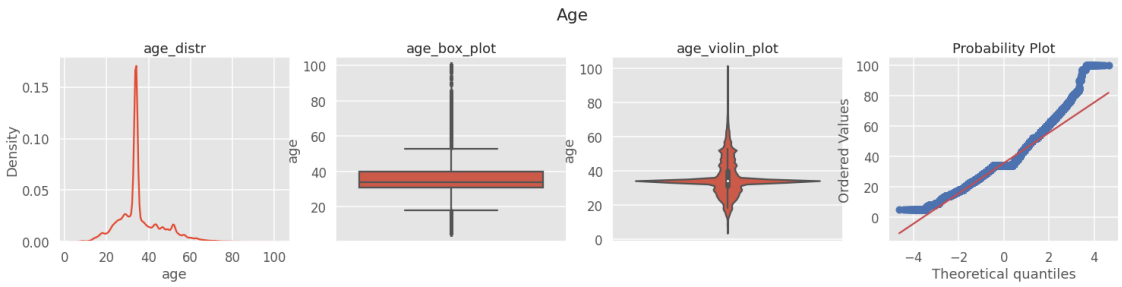
[57]:
```
def plot_univariate(dataset, column_name, suptitle = None, kde_only = True):
    f, axes = plt.subplots(ncols = 4, figsize=(25, 5))
    if kde_only:
        sns.kdeplot(x=column_name, data=dataset, ax=axes[0]).
 ↪set_title(f'{column_name}_distr')
    else:
        sns.histplot(x=column_name, data=dataset, kde=True, ax=axes[0]).
 ↪set_title(f'{column_name}_distr')
    sns.boxplot(y=column_name, data=dataset, orient='v', ax=axes[1]).
 ↪set_title(f'{column_name}_box_plot')
    sns.violinplot(y=column_name, data=dataset, orient='v', ax=axes[2]).
 ↪set_title(f'{column_name}_violin_plot')
```

```
    scipy.stats.probplot(dataset[column_name], dist="norm", plot=axes[3])
    if suptitle:
        plt.suptitle(suptitle)
        plt.subplots_adjust(top=0.80)
    plt.show()
```

[58]:
```
plot_univariate(dataset=dataset, column_name='book_rating', suptitle='Book␣
↪Rating')
```



[59]:
```
plot_univariate(dataset=dataset, column_name='age', suptitle='Age')
```



1.3 Data Transformation

1.3.1 Min-Max Normalization

$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$

[82]:
```
scaler = MinMaxScaler()
scaled = scaler.fit_transform(dataset['book_rating'].values.reshape(-1, 1)).
↪reshape(-1)
```

[83]:
```
scaled_rating = pd.DataFrame(data=scaled, columns=['book_rating'])
scaled_rating
```

[83]:

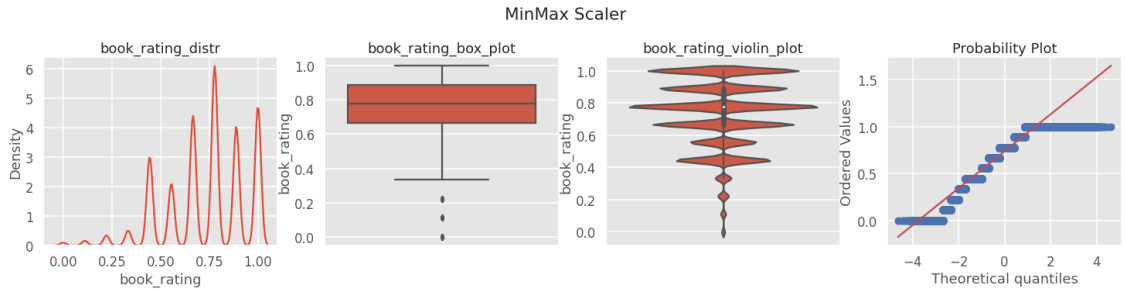
	book_rating
0	0.444444
1	0.777778

```
2          0.777778
3          0.888889
4          0.888889
...             ...
383844     0.666667
383845     0.444444
383846     0.666667
383847     0.666667
383848     1.000000

[383849 rows x 1 columns]
```

[84]:
```
scaled_rating.describe().T
```

[84]:
```
              count     mean       std  min       25%       50%       75%  max
book_rating  383849.0  0.7363  0.204593  0.0  0.666667  0.777778  0.888889  1.0
```

[85]:
```
plot_univariate(dataset=scaled_rating, column_name='book_rating',␣
 ↪suptitle='MinMax Scaler')
```



1.3.2 Z-Score Standardization

$z = \frac{x-\mu}{\sigma}$

[86]:
```
scaler = StandardScaler()
scaled = scaler.fit_transform(dataset['book_rating'].values.reshape(-1, 1)).
 ↪reshape(-1)
```

[87]:
```
scaled_rating = pd.DataFrame(data=scaled, columns=['book_rating'])
scaled_rating
```

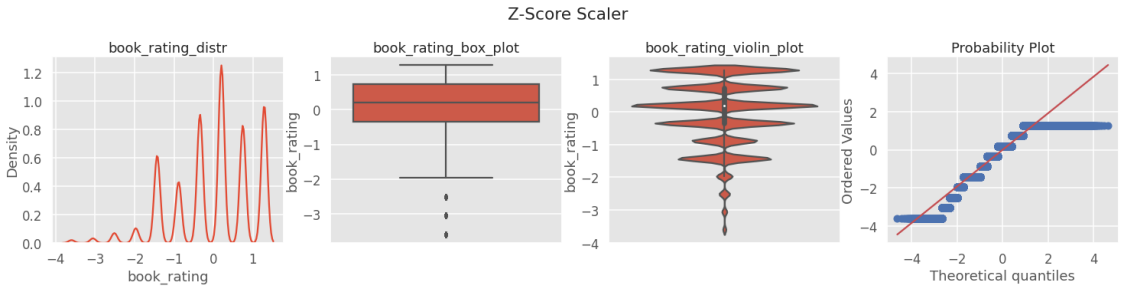
[87]:
```
        book_rating
0         -1.426522
1          0.202732
2          0.202732
3          0.745817
4          0.745817
...             ...
383844    -0.340353
```

```
383845    -1.426522
383846    -0.340353
383847    -0.340353
383848     1.288902

[383849 rows x 1 columns]
```

[88]:
```
scaled_rating.describe().T
```

[88]:
```
                count          mean       std  ...       50%       75%       max
book_rating  383849.0  2.953229e-14  1.000001  ...  0.202732  0.745817  1.288902

[1 rows x 8 columns]
```

[89]:
```
plot_univariate(dataset=scaled_rating, column_name='book_rating',␣
 ↪suptitle='Z-Score Scaler')
```



1.3.3 Decimal Scaling

$v_i' = \frac{v_i}{10^j}$

[90]:
```
p = dataset['book_rating'].max()
q = len(str(abs(p)))
scaled = dataset['book_rating'].values / 10 ** q
```

[91]:
```
scaled_rating = pd.DataFrame(data=scaled, columns=['book_rating'])
scaled_rating
```

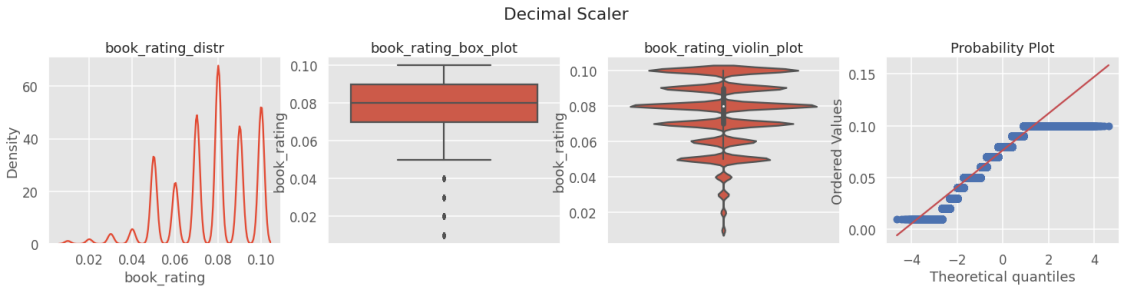
[91]:
```
        book_rating
0              0.05
1              0.08
2              0.08
3              0.09
4              0.09
...             ...
383844         0.07
383845         0.05
383846         0.07
383847         0.07
```

```
383848         0.10

[383849 rows x 1 columns]
```

[92]:
```
scaled_rating.describe().T
```

[92]:
```
                count      mean       std   min   25%   50%   75%  max
book_rating  383849.0  0.076267  0.018413  0.01  0.07  0.08  0.09  0.1
```

[93]:
```
plot_univariate(dataset=scaled_rating, column_name='book_rating',␣
↪suptitle='Decimal Scaler')
```



1.4 Data Normality

1.4.1 Natural Log Transform

[162]:
```
transformed = np.log(dataset['book_rating'].astype(np.float32))
```

[163]:
```
trans_rating = pd.DataFrame(data=transformed, columns=['book_rating'])
trans_rating
```

[163]:
```
         book_rating
1           1.609438
3           2.079442
5           2.079442
8           2.197225
9           2.197225
...              ...
1031166     1.945910
1031168     1.609438
1031169     1.945910
1031170     1.945910
1031171     2.302585

[383849 rows x 1 columns]
```

[164]:
```
trans_rating.describe().T
```

[164]:
```
                count      mean       std  ...       50%       75%       max
book_rating  383849.0  1.994574  0.301864  ...  2.079442  2.197225  2.302585

[1 rows x 8 columns]
```

[165]:
```
plot_univariate(dataset=trans_rating, column_name='book_rating',␣
 ↪suptitle='Natural Log Transform')
```

1.4.2 Square Root Transform

[166]:
```
transformed = np.sqrt(dataset['book_rating'].astype(np.float32))
```

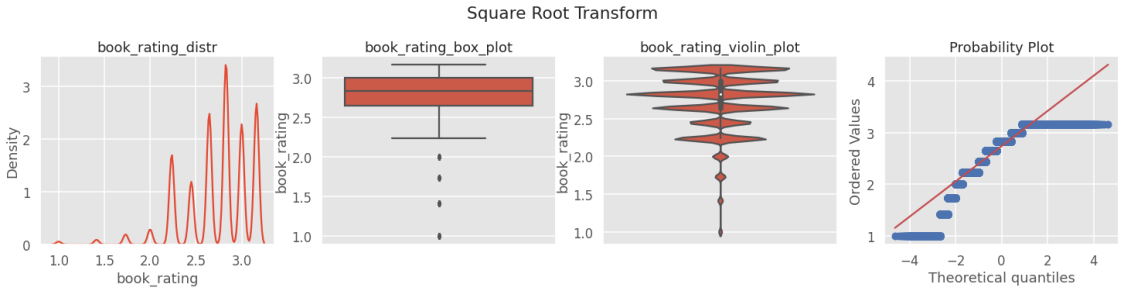
[167]:
```
trans_rating = pd.DataFrame(data=transformed, columns=['book_rating'])
trans_rating
```

[167]:
```
         book_rating
1           2.236068
3           2.828427
5           2.828427
8           3.000000
9           3.000000
...              ...
1031166     2.645751
1031168     2.236068
1031169     2.645751
1031170     2.645751
1031171     3.162278

[383849 rows x 1 columns]
```

[168]:
```
trans_rating.describe().T
```

[168]:
```
                count      mean       std  ...       50%  75%       max
book_rating  383849.0  2.738279  0.362097  ...  2.828427  3.0  3.162278

[1 rows x 8 columns]
```

[169]:
```
plot_univariate(dataset=trans_rating, column_name='book_rating',␣
↪suptitle='Square Root Transform')
```



1.4.3 Inverse Square Root Transformation

[170]:
```
transformed = np.power(dataset['book_rating'].astype(np.float32), -1/2)
```

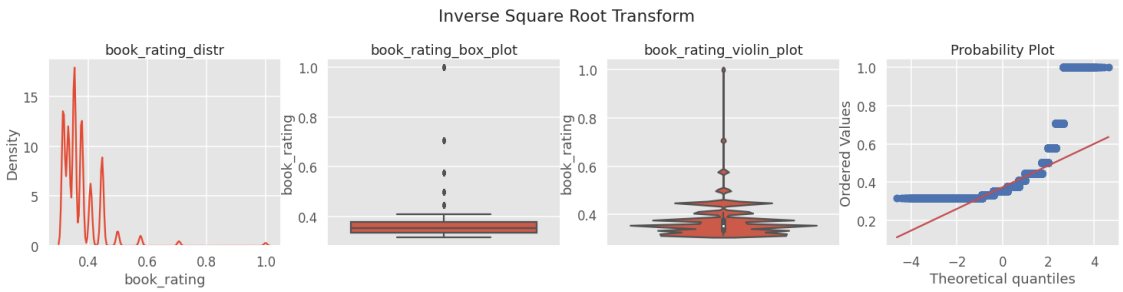
[171]:
```
trans_rating = pd.DataFrame(data=transformed, columns=['book_rating'])
trans_rating
```

[171]:
```
         book_rating
1           0.447214
3           0.353553
5           0.353553
8           0.333333
9           0.333333
...              ...
1031166     0.377964
1031168     0.447214
1031169     0.377964
1031170     0.377964
1031171     0.316228

[383849 rows x 1 columns]
```

[172]:
```
trans_rating.describe().T
```

[172]:
```
                count      mean       std  ...       50%       75%  max
book_rating  383849.0  0.373803   0.06913  ...  0.353553  0.377964  1.0

[1 rows x 8 columns]
```

[173]:
```
plot_univariate(dataset=trans_rating, column_name='book_rating',␣
↪suptitle='Inverse Square Root Transform')
```

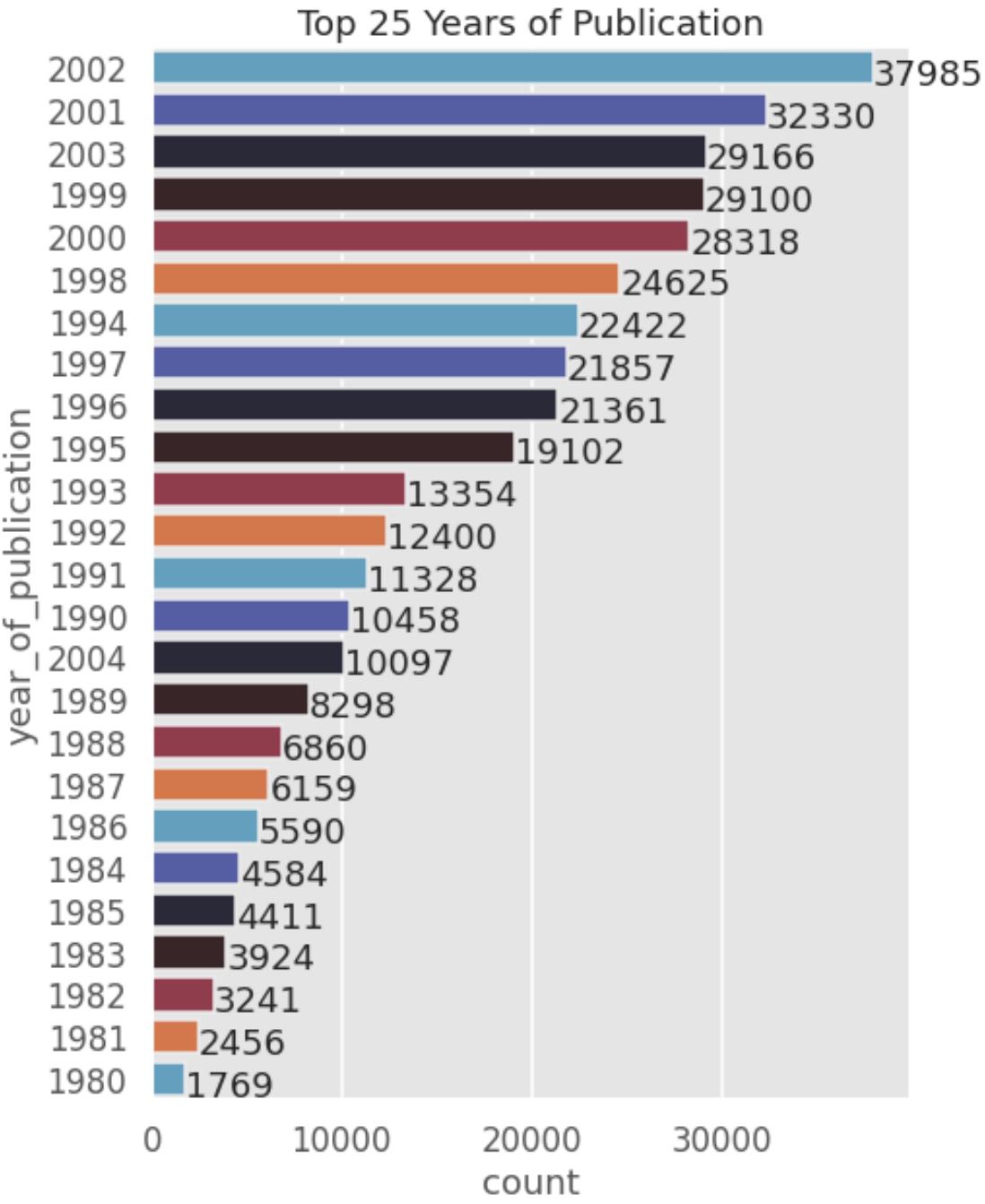


1.5 Exploratory Data Analysis

[186]:
```
# to plot values in barplot, https://stackoverflow.com/a/56780852
def show_values_on_bars(axs, h_v="v", space=0.4):
    def _show_on_single_plot(ax):
        if h_v == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                _y = p.get_y() + p.get_height()
                value = int(p.get_height())
                ax.text(_x, _y, value, ha="center")
        elif h_v == "h":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() + float(space)
                _y = p.get_y() + p.get_height()
                value = int(p.get_width())
                ax.text(_x, _y, value, ha="left")

    if isinstance(axs, np.ndarray):
        for idx, ax in np.ndenumerate(axs):
            _show_on_single_plot(ax)
    else:
        _show_on_single_plot(axs)
```

[256]:
```
eda = dataset['year_of_publication'].copy().value_counts().head(25).
 ↪reset_index()
eda.columns = ['year_of_publication', 'count']
eda = eda.sort_values(by=['count'], ascending=False)
```

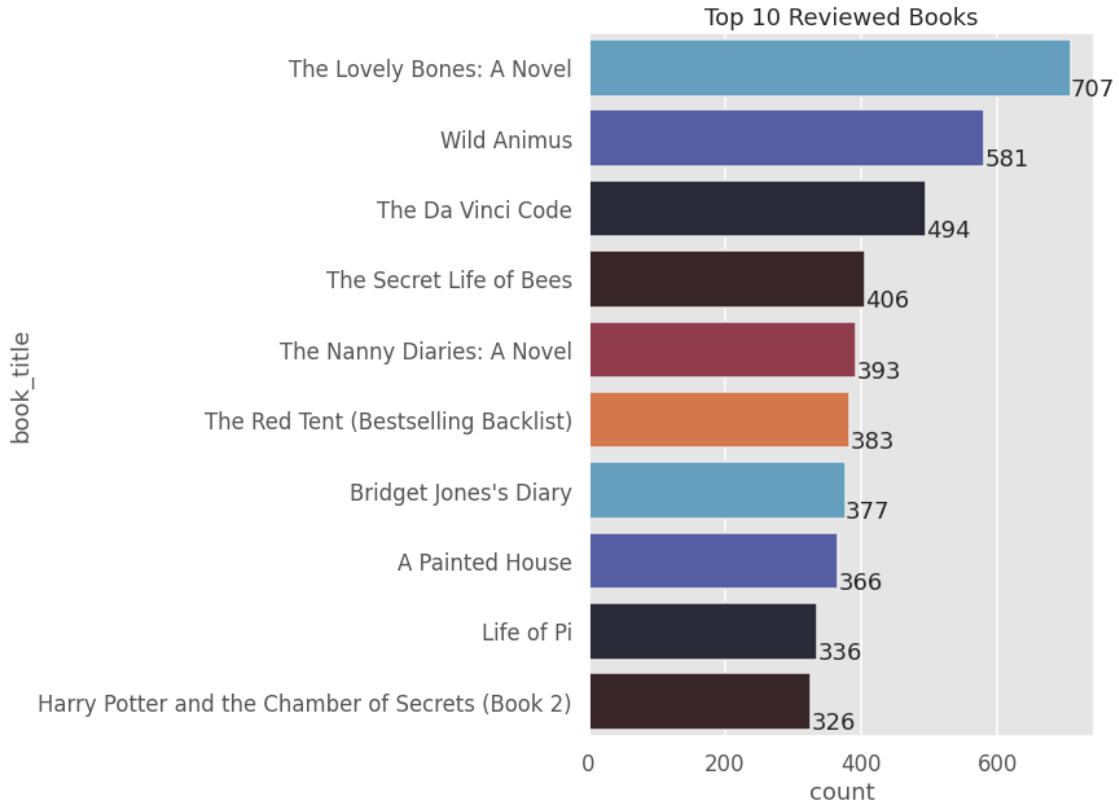
[257]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='count', y='year_of_publication', data=eda,␣
 ↪order=eda['year_of_publication'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
plt.title('Top 25 Years of Publication')
plt.show()
```



[258]:
```
eda = dataset['book_title'].value_counts().head(10).reset_index()
eda.columns = ['book_title', 'count']
```

[259]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='count', y='book_title', data=eda,␣
 ↪order=eda['book_title'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
```

```
plt.title('Top 10 Reviewed Books')
plt.show()
```



[280]:
```
rating_count = dataset['book_title'].value_counts().head(25).reset_index().
 ↪sort_values(by='book_title').reset_index(drop=True)
rating_count.columns = ['book_title', 'rating_count']
rating_count.head(5)
```

[280]:

	book_title	rating_count
0	The Girls' Guide to Hunting and Fishing	259
1	The Testament	261
2	Timeline	263
3	The Catcher in the Rye	265
4	To Kill a Mockingbird	267

[281]:
```
rating_sum = dataset[dataset['book_title'].isin(rating_count['book_title'])].
 ↪groupby(['book_title'])['book_rating'].sum().reset_index().
 ↪sort_values(by=['book_title'])
rating_sum.columns = ['book_title', 'rating_sum']
rating_sum.head(5)
```

[281]:

	book_title	rating_sum
0	A Painted House	2708.0

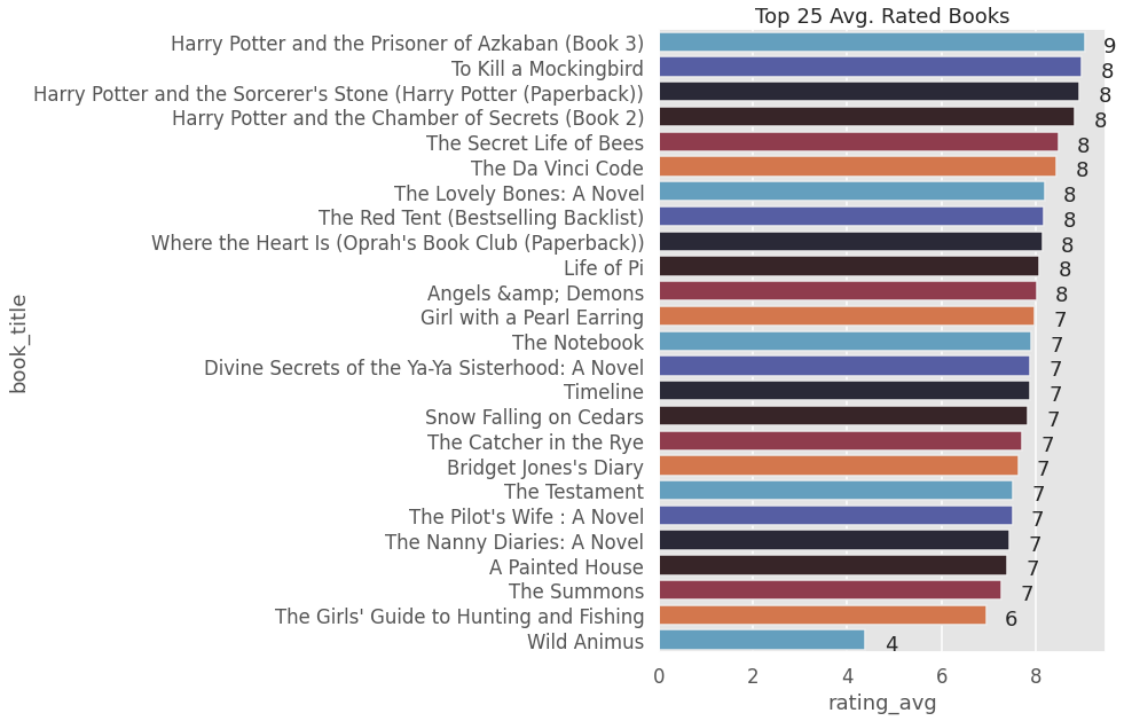
```
1                              Angels &amp;amp; Demons      2485.0
2                            Bridget Jones's Diary      2875.0
3  Divine Secrets of the Ya-Ya Sisterhood: A Novel      2544.0
4                         Girl with a Pearl Earring      2219.0
```

[282]:
```
avg_rating = pd.merge(rating_count, rating_sum, on='book_title')
avg_rating['rating_avg'] = avg_rating['rating_sum'] / avg_rating['rating_count']
avg_rating = avg_rating.sort_values(by='rating_avg', ascending=False).
 ↪reset_index(drop=True)
```

[283]:
```
avg_rating.head(5)
```

[283]:
```
                                          book_title  ...  rating_avg
0  Harry Potter and the Prisoner of Azkaban (Book 3)  ...    9.043321
1                              To Kill a Mockingbird  ...    8.977528
2  Harry Potter and the Sorcerer's Stone (Harry P...  ...    8.936508
3   Harry Potter and the Chamber of Secrets (Book 2)  ...    8.840491
4                            The Secret Life of Bees  ...    8.477833

[5 rows x 4 columns]
```

[284]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='rating_avg', y='book_title', data=avg_rating,␣
 ↪order=avg_rating['book_title'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
plt.title('Top 25 Avg. Rated Books')
plt.show()
```

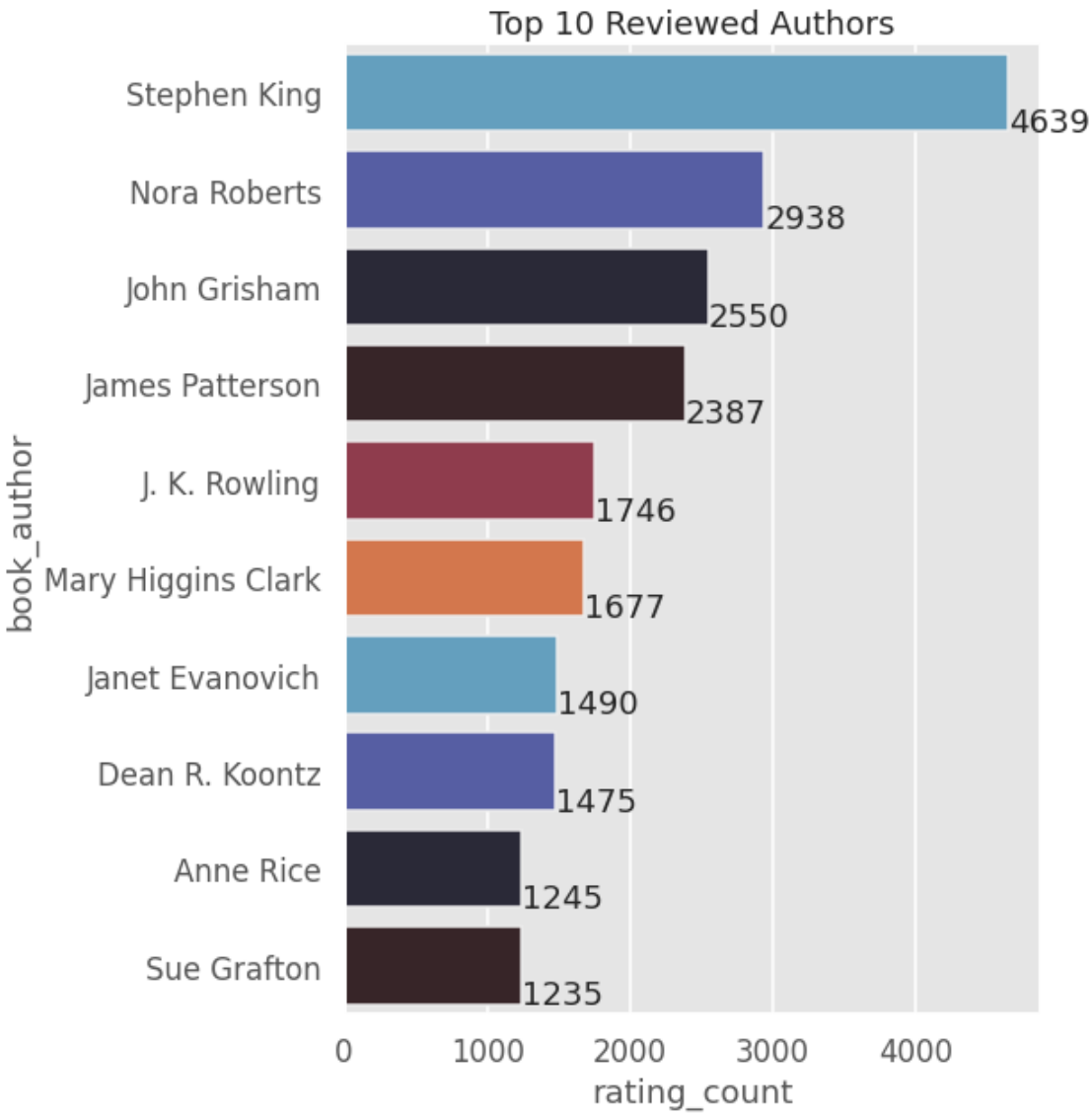


[285]:
```
author_count = dataset['book_author'].value_counts().head(10).reset_index()
author_count.columns = ['book_author', 'rating_count']
author_count.head(5)
```

[285]:

	book_author	rating_count
0	Stephen King	4639
1	Nora Roberts	2938
2	John Grisham	2550
3	James Patterson	2387
4	J. K. Rowling	1746

[286]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='rating_count', y='book_author', data=author_count,␣
↪order=author_count['book_author'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
plt.title('Top 10 Reviewed Authors')
plt.show()
```

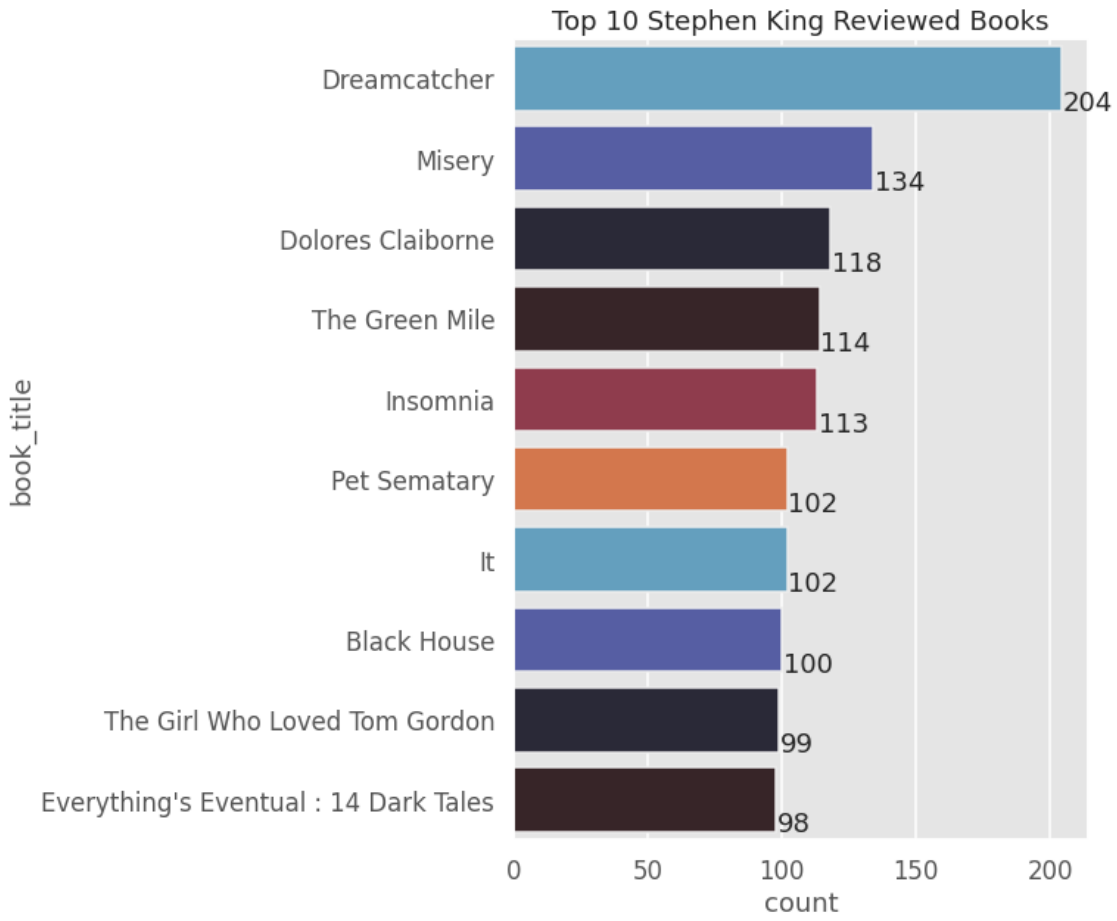


1.5.1 It seems Stephen King is a very popular author !

[288]:
```
stephen_king = dataset[dataset['book_author'] == 'Stephen King']
```

[289]:
```
eda = stephen_king['book_title'].value_counts().head(10).reset_index()
eda.columns = ['book_title', 'count']
```

[290]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='count', y='book_title', data=eda,␣
 ↪order=eda['book_title'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
plt.title('Top 10 Stephen King Reviewed Books')
plt.show()
```



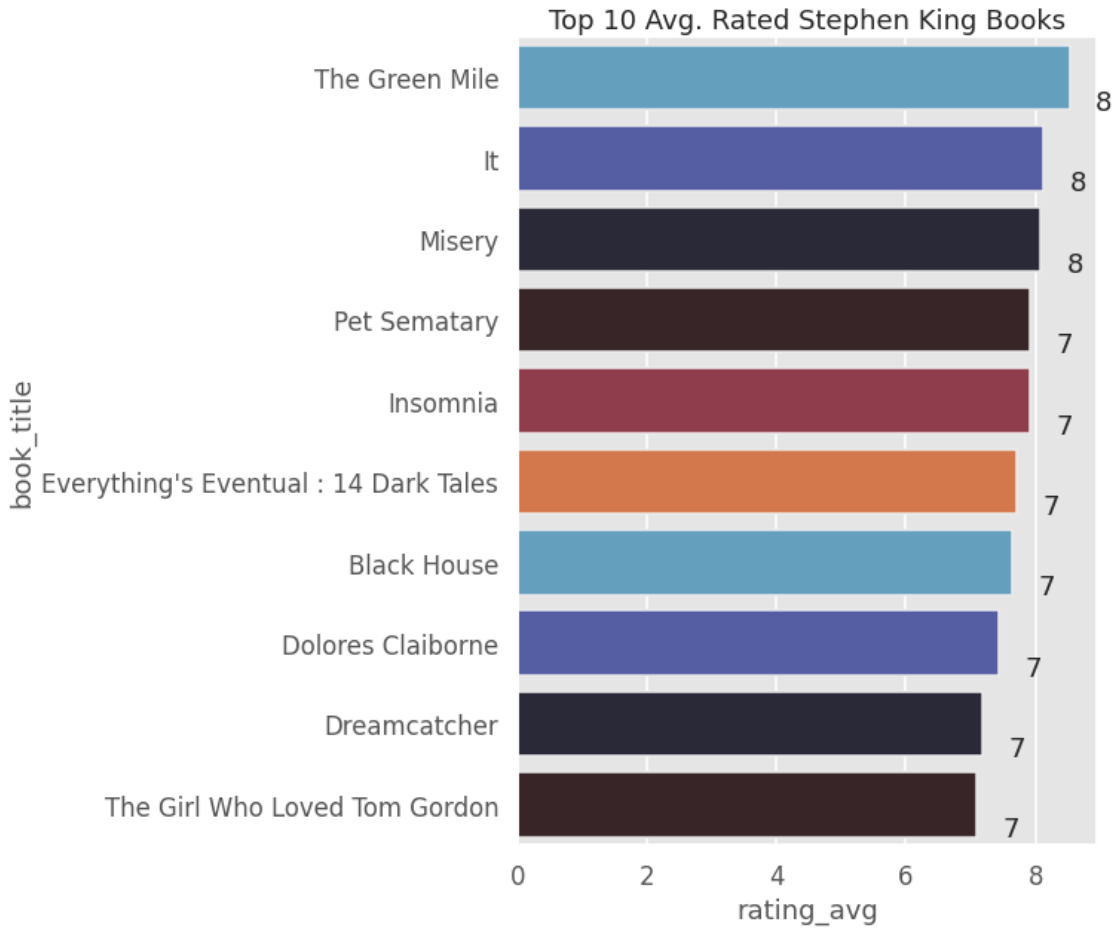
[291]:
```
rating_count = stephen_king['book_title'].value_counts().head(10).reset_index().
 ↪sort_values(by='book_title').reset_index(drop=True)
rating_count.columns = ['book_title', 'rating_count']
rating_sum = stephen_king[stephen_king['book_title'].
 ↪isin(rating_count['book_title'])].groupby(['book_title'])['book_rating'].
 ↪sum().reset_index().sort_values(by=['book_title'])
rating_sum.columns = ['book_title', 'rating_sum']
avg_rating = pd.merge(rating_count, rating_sum, on='book_title')
avg_rating['rating_avg'] = avg_rating['rating_sum'] / avg_rating['rating_count']
avg_rating = avg_rating.sort_values(by='rating_avg', ascending=False).
 ↪reset_index(drop=True)
```

[292]:
```
avg_rating.head(5)
```

[292]:

	book_title	rating_count	rating_sum	rating_avg
0	The Green Mile	114	972.0	8.526316
1	It	102	829.0	8.127451
2	Misery	134	1082.0	8.074627
3	Pet Sematary	102	808.0	7.921569
4	Insomnia	113	895.0	7.920354

[293]:
```
plt.figure(figsize=(7, 10))
splot = sns.barplot(x='rating_avg', y='book_title', data=avg_rating,␣
 ↪order=avg_rating['book_title'], orient='h', palette=palette)
show_values_on_bars(splot, h_v="h")
plt.title('Top 10 Avg. Rated Stephen King Books')
plt.show()
```



1.6 Convert this Notebook to PDF

[181]:
```
%%capture
! apt update
! apt install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-generic-recommended
```

[182]:
```
import subprocess
import shlex
```

Convert to PDF

[185]:
```
s = subprocess.Popen(shlex.split(
    f'jupyter nbconvert /content/BookReview_EDA.ipynb --to pdf'
    ), shell = False, stdout = subprocess.PIPE, stderr = subprocess.PIPE)
s.wait()
s.stdout.read()
```

[185]:
```
b''
```

Convert to LaTeX

[186]:
```
s = subprocess.Popen(shlex.split(
    f'jupyter nbconvert /content/BookReview_EDA.ipynb --to latex'
    ), shell = False, stdout = subprocess.PIPE, stderr = subprocess.PIPE)
s.wait()
s.stdout.read()
```

[186]:
```
b''
```

[187]:
```
! zip -r BookReview_EDA_latex.zip BookReview_EDA_files BookReview_EDA.tex
```

```
  adding: BookReview_EDA_files/ (stored 0%)
  adding: BookReview_EDA_files/BookReview_EDA_53_0.png (deflated 7%)
  adding: BookReview_EDA_files/BookReview_EDA_99_0.png (deflated 12%)
  adding: BookReview_EDA_files/BookReview_EDA_105_0.png (deflated 13%)
  adding: BookReview_EDA_files/BookReview_EDA_83_0.png (deflated 6%)
  adding: BookReview_EDA_files/BookReview_EDA_101_0.png (deflated 13%)
  adding: BookReview_EDA_files/BookReview_EDA_78_0.png (deflated 6%)
  adding: BookReview_EDA_files/BookReview_EDA_92_0.png (deflated 13%)
  adding: BookReview_EDA_files/BookReview_EDA_94_0.png (deflated 14%)
  adding: BookReview_EDA_files/BookReview_EDA_108_0.png (deflated 14%)
  adding: BookReview_EDA_files/BookReview_EDA_52_0.png (deflated 5%)
  adding: BookReview_EDA_files/BookReview_EDA_66_0.png (deflated 5%)
  adding: BookReview_EDA_files/BookReview_EDA_48_0.png (deflated 6%)
  adding: BookReview_EDA_files/BookReview_EDA_88_0.png (deflated 8%)
  adding: BookReview_EDA_files/BookReview_EDA_60_0.png (deflated 5%)
  adding: BookReview_EDA_files/BookReview_EDA_72_0.png (deflated 5%)
  adding: BookReview_EDA.tex (deflated 87%)
```

[ ]:
```

```