Hospital Procedure Volume

November 30, 2021

1 I would like to thank Mattia Ciollaro for providing the data and skills necessary to continue my journey

Purpose of this analysis: To explore different types of procedures performed at a hospital.

Data Source: A clean (The data squeaks type of clean) data set of 1,463 hospitals and the number of surgical procedures they performed in 2016.

Types of Procedures Performed: Gastrointestinal (GI), Eye, Nervous System, Musculoskeletal, Skin (Derm), Genitourinary, Cardiovascular, Respiratory, and Other.

Note: The actual amount of procedures performed at a large health system (For Example: Mayo Clinic) is far more than what you might expect if you are coming from a non-healthcare background. To give perspective, Mayo Clinic general surgeons perform over 10,000 procedures annually: https://www.mayoclinic.org/departments-centers/general-surgery/overview

[1]:
```
from os import path
import matplotlib.lines as ln
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
```

1.1 Read data

[2]:
```
HospitalProcedures = pd.read_csv("hospital_procedures_volumes.csv")
```

Each row is a hospital.

The columns provide counts of procedures performed at the given hospital in the year 2016

[3]:
```
HospitalProcedures.head(10)
```

[3]:

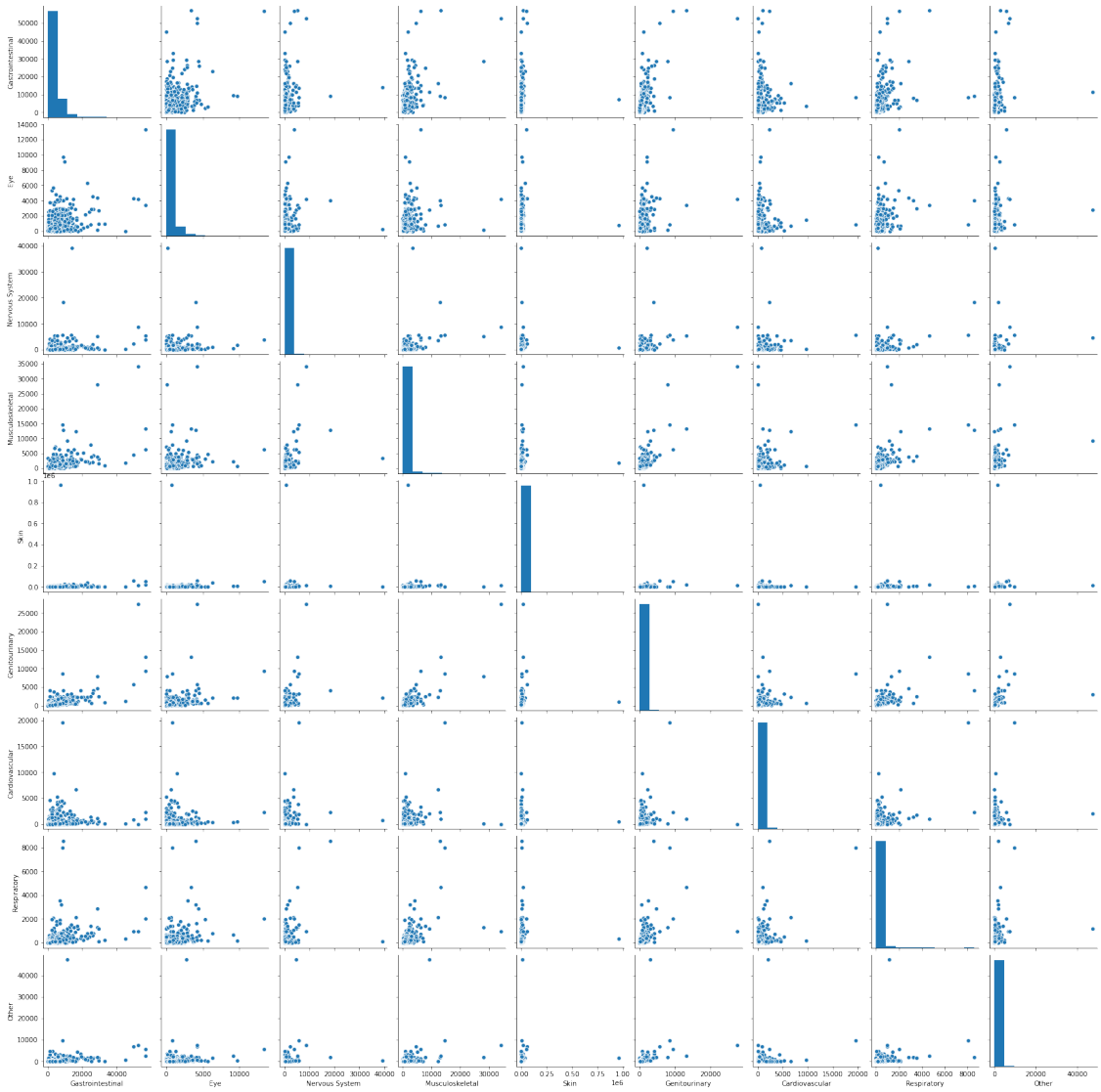
	Gastrointestinal	Eye	Nervous System	Musculoskeletal	Skin	\
0	3461.0	77.0	170.0	711.0	241.0	
1	2464.0	25.0	23.0	69.0	994.0	
2	1248.0	275.0	133.0	111.0	4936.0	

3	1785.0	49.0	109.0	617.0	155.0
4	8052.0	1473.0	410.0	1871.0	375.0
5	1953.0	1308.0	1777.0	838.0	1663.0
6	1548.0	140.0	186.0	130.0	564.0
7	1962.0	116.0	7.0	333.0	817.0
8	13964.0	205.0	39244.0	3318.0	1220.0
9	3460.0	10.0	151.0	532.0	2892.0

	Genitourinary	Cardiovascular	Respiratory	Other
0	322.0	1.0	146.0	56.0
1	63.0	24.0	55.0	27.0
2	100.0	170.0	57.0	474.0
3	549.0	155.0	33.0	209.0
4	732.0	212.0	281.0	116.0
5	352.0	102.0	86.0	3.0
6	146.0	309.0	18.0	9.0
7	28.0	9.0	28.0	3.0
8	2007.0	779.0	127.0	329.0
9	982.0	152.0	397.0	280.0

1.2 Let's Examine The Data

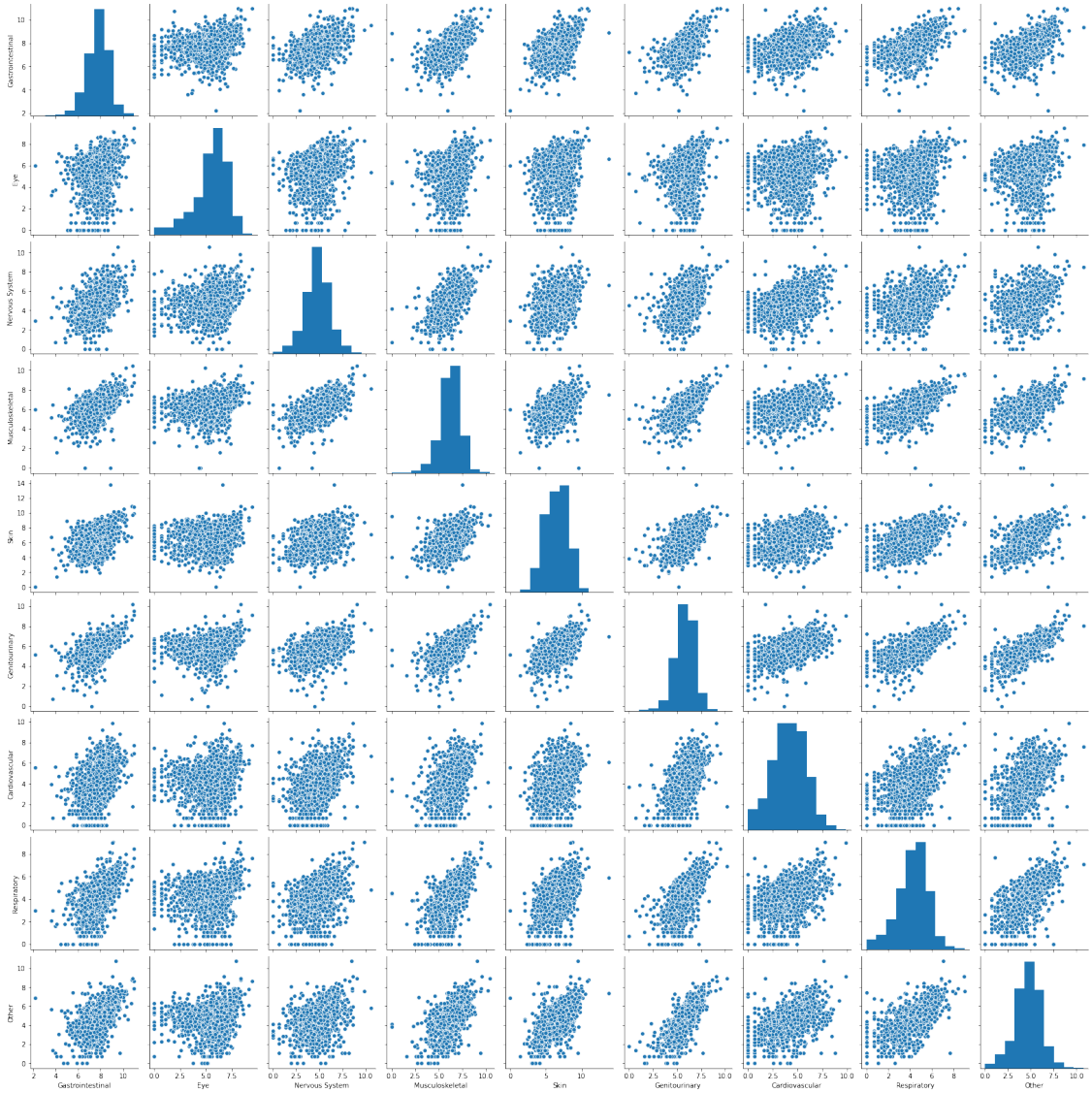
[4]:
```
_plt = sns.pairplot(HospitalProcedures)
```



Seaborn (Denoted by the sns designation) pairplots allow you to read the data from left to right and check for associtions between the procedures.

The first set of pairplots has some skewness, so applying a log transformation should "correct" it.

[5]:
```
_plt = sns.pairplot(np.log(HospitalProcedures))
```

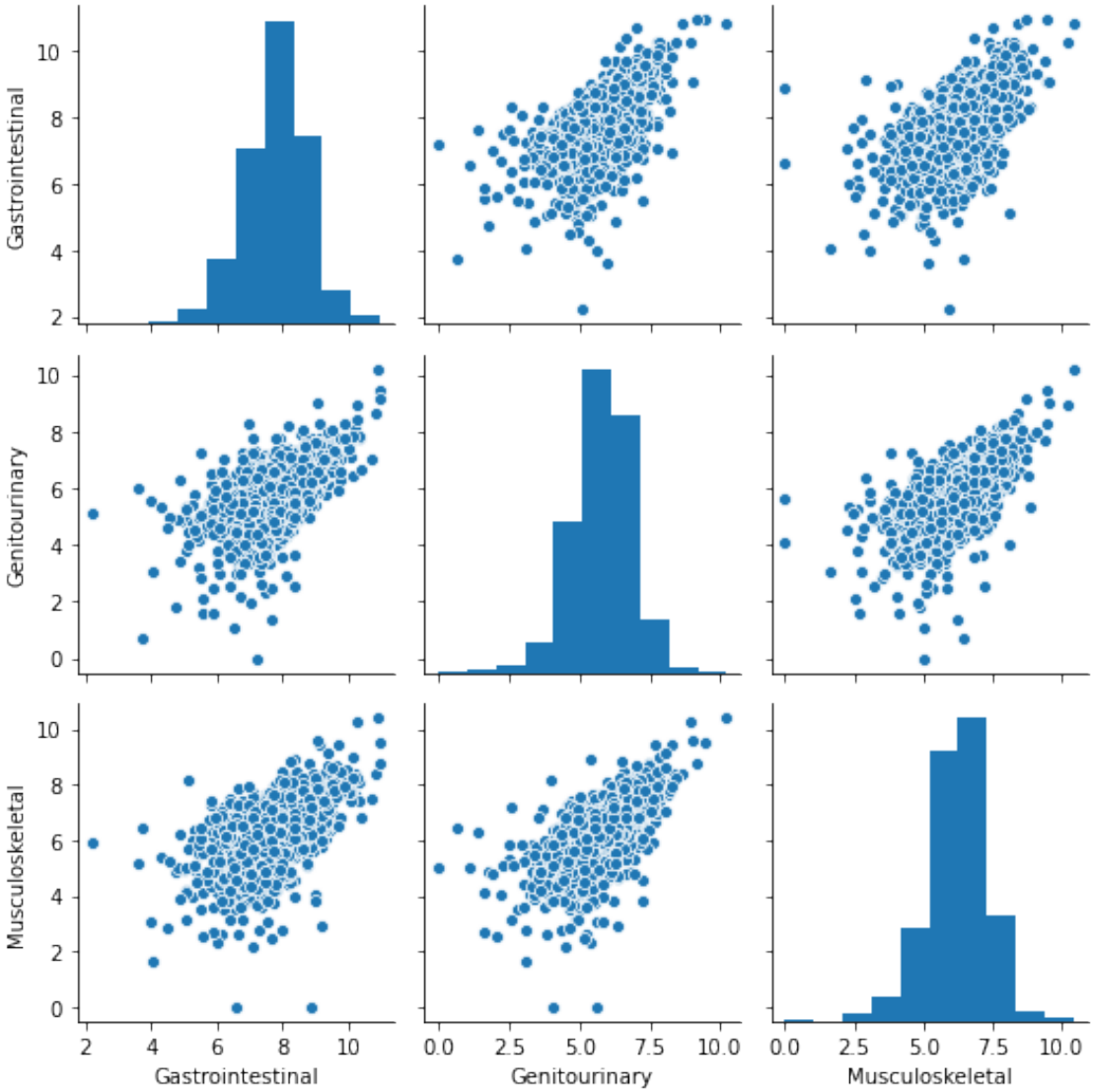


Looking at the pair plots after the log transformation, its interesting to see that a potential relationship exists between the volume of Gastrointestinal, Genitourinary, and Musculoskeletal procedures.

Let's examine that a bit further in the next pair plot

Seaborn Pair Plot Documentation: https://seaborn.pydata.org/generated/seaborn.pairplot.html

[6]:
```
_plt = sns.pairplot(
    np.log(HospitalProcedures[
        ["Gastrointestinal", "Genitourinary", "Musculoskeletal"]
    ])
)
plt.tight_layout()
```



We can see that the counts of the above procedures are positively correlated. Let's take advantage of these correlations to build a predictive model.

In case you were wondering, here is the full correlation matrix:

[7]:
```
np.log(HospitalProcedures).corr()
```

[7]:

	Gastrointestinal	Eye	Nervous System	Musculoskeletal	\
Gastrointestinal	1.000000	0.273242	0.488188	0.563213	
Eye	0.273242	1.000000	0.270040	0.233043	
Nervous System	0.488188	0.270040	1.000000	0.639709	
Musculoskeletal	0.563213	0.233043	0.639709	1.000000	
Skin	0.469180	0.126126	0.339109	0.469409	
Genitourinary	0.593920	0.176226	0.447444	0.633061	

```
Cardiovascular            0.384172  0.097935        0.415486         0.433881
Respiratory               0.551406  0.190709        0.420631         0.590303
Other                     0.521969  0.128830        0.364640         0.593747

                     Skin  Genitourinary  Cardiovascular  Respiratory  \
Gastrointestinal  0.469180       0.593920        0.384172     0.551406
Eye               0.126126       0.176226        0.097935     0.190709
Nervous System    0.339109       0.447444        0.415486     0.420631
Musculoskeletal   0.469409       0.633061        0.433881     0.590303
Skin              1.000000       0.515370        0.406639     0.484982
Genitourinary     0.515370       1.000000        0.552054     0.628251
Cardiovascular    0.406639       0.552054        1.000000     0.458652
Respiratory       0.484982       0.628251        0.458652     1.000000
Other             0.570308       0.748685        0.538454     0.643636

                     Other
Gastrointestinal  0.521969
Eye               0.128830
Nervous System    0.364640
Musculoskeletal   0.593747
Skin              0.570308
Genitourinary     0.748685
Cardiovascular    0.538454
Respiratory       0.643636
Other             1.000000
```

1.2.1 Make a version of the 'logged data' for future use

[8]:
```
LoggedHospitalProcedures = np.log(HospitalProcedures)
```

We will use the log-counts of the other medical procedures to develop a model that can help us predict the count of the Gastrointestinal medical procedures at a given hospital.

[9]:
```
X = sm.add_constant(LoggedHospitalProcedures.drop("Gastrointestinal", axis=1))
```

[10]:
```
Y = HospitalProcedures["Gastrointestinal"]
```

[11]:
```
log_Y = LoggedHospitalProcedures["Gastrointestinal"]
```

1.3 Goal

Devlop a regression model to predict the number of yearly Gastrointestinal procedures at a given hospital given the number of procedures performed in a different medical specializations.

Two ways to go about performing Poisson Regression in Python: Method 1: The way shown below with statsmodels. Statsmodels Documentation: https://www.statsmodels.org/stable/index.html Method 2: Via Scikit-Learn. Documentation for Poisson Regression in Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PoissonRegressor.html

1.4 Poisson Regression Model

Pick the `Poisson` regression model from the `statsmodels` module.

[12]:
```
poisson_regression = sm.Poisson(Y, X)
```

[13]:
```
poisson_regression_fit = poisson_regression.fit()
```

```
Optimization terminated successfully.
         Current function value: 685.234747
         Iterations 6
```

[14]:
```
poisson_regression_pred = poisson_regression_fit.predict(X)
```

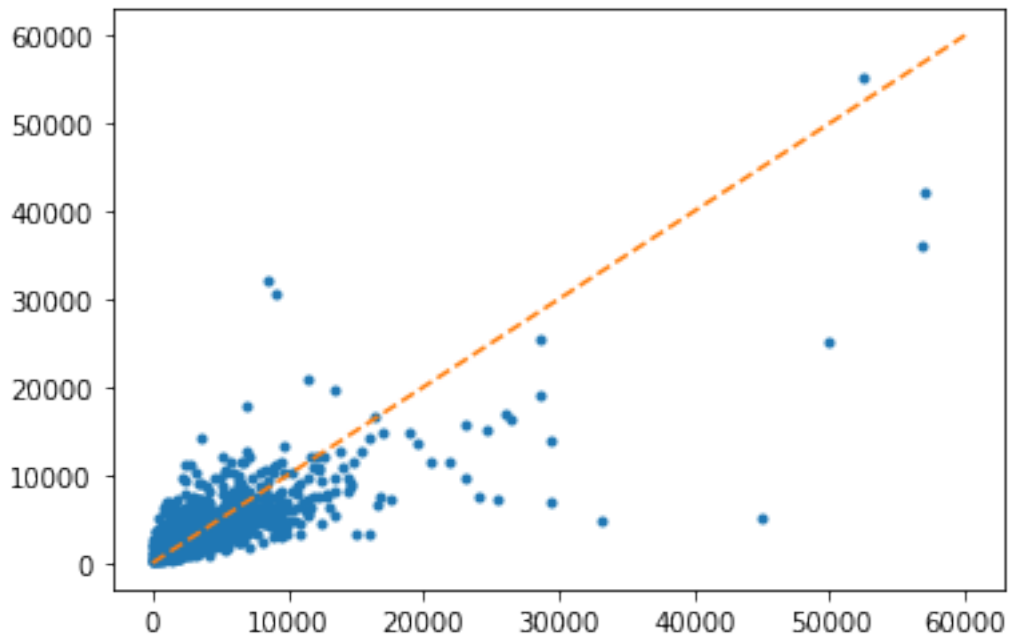
So we created a prediction. Let's plot it and see how it performed.

Note: Great references for plots: https://jakevdp.github.io/PythonDataScienceHandbook/index.html
Example Plots with Code: https://matplotlib.org/stable/gallery/index.html#pyplot

[15]:
```
_plt = plt.plot(Y, poisson_regression_pred, ".")

# a little utility to add the bisector line
def add_bisector(xmin, xmax, ymin, ymax, **kwargs):
    _bisector = ln.Line2D(
        [xmin, xmax], [ymin, ymax],
        **kwargs
    )
    ax = plt.gca()
    _ = ax.add_line(_bisector)

add_bisector(0, 60000, 0, 60000, linestyle="--", color=sns.color_palette()[1])
```



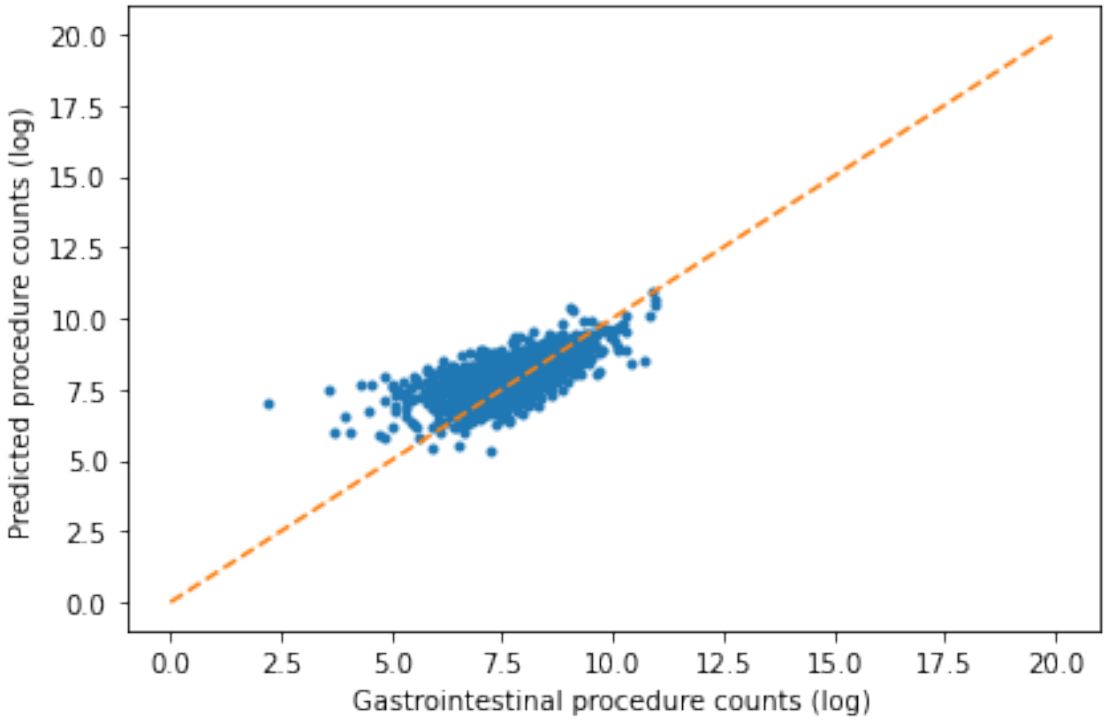
Another skewed distribution. To better visualize the predictions, let's utilize a log-log coordinates system.

[16]:
```
_plt = plt.plot(log_Y, np.log(poisson_regression_pred), ".")

# add the bisector line
add_bisector(0, 20, 0, 20, linestyle="--", color=sns.color_palette()[1])

# add titles and axis labels
ax = plt.gca()
ax.set_xlabel("Gastrointestinal procedure counts (log)")
ax.set_ylabel("Predicted procedure counts (log)")

plt.tight_layout()
```



Let's now compute the RMSE for our model. RMSE stands for Root Mean Square Error (RMSE): https://stats.stackexchange.com/questions/56302/what-are-good-rmse-values

[17]:
```
def compute_rmse(y, y_pred):
    squared_residuals = np.square(y - y_pred)
    return np.sqrt(np.mean(squared_residuals))
```

[18]:
```
poisson_regression_rmse = compute_rmse(
    HospitalProcedures["Gastrointestinal"], poisson_regression_pred
)
```

[19]:
```
poisson_regression_rmse
```

[19]:
```
3052.0095605946335
```

For comparison, what is the RMSE associated to the dummy model that only uses the average "Gastrointestinal" counts and none of the other available predictors?

[20]:
```
# Note that this is simply the variance of the response variable
dummy_rmse = np.std(HospitalProcedures["Gastrointestinal"])
```

[21]:
```
dummy_rmse
```

[21]:
```
4665.03848056645
```

We are doing better than the dummy model, but how much better?

[22]:
```
poisson_regression_rmse / dummy_rmse
```

[22]:
```
0.6542303077045668
```

It looks like the model has a RMSE that is about 35% lower that the RMSE corresponding to the dummy model corresponding to the mean of the response variable.

Interpretation of the Model: Credit for the equations and text block goes to Mattia Ciollaro

1.4.1 Interpretation of the model

We have fitted a model for $E(Y \mid X)$ by assuming that:

- Y is Poisson distributed with expected value equal to $E(Y \mid X)$
- $E(Y \mid X) = \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)$

How can we interpret, e.g., β_1?

Well, by doing some little algebra, we see that

$$E(Y \mid X_1 = x + 1, X_2, \ldots, X_p) \quad / \quad E(Y \mid X_1 = x, X_2, \ldots, X_p) = \exp(\beta_1)$$

which means that $\exp(\beta_1)$ measures the relative variation of the expected value of Y corresponding to a unit increase in X_1 (assuming that all other predictors are kept constant).

[23]:
```
poisson_regression_fit.params
```

[23]:
```
const              3.881848
Eye                0.054298
Nervous System     0.079036
Musculoskeletal    0.094404
Skin               0.071910
```

```
Genitourinary      0.328292
Cardiovascular    -0.011764
Respiratory        0.107771
Other              0.012123
dtype: float64
```

[24]:
```
np.exp(poisson_regression_fit.params["Eye"])
```

[24]:
```
1.0557987394578274
```

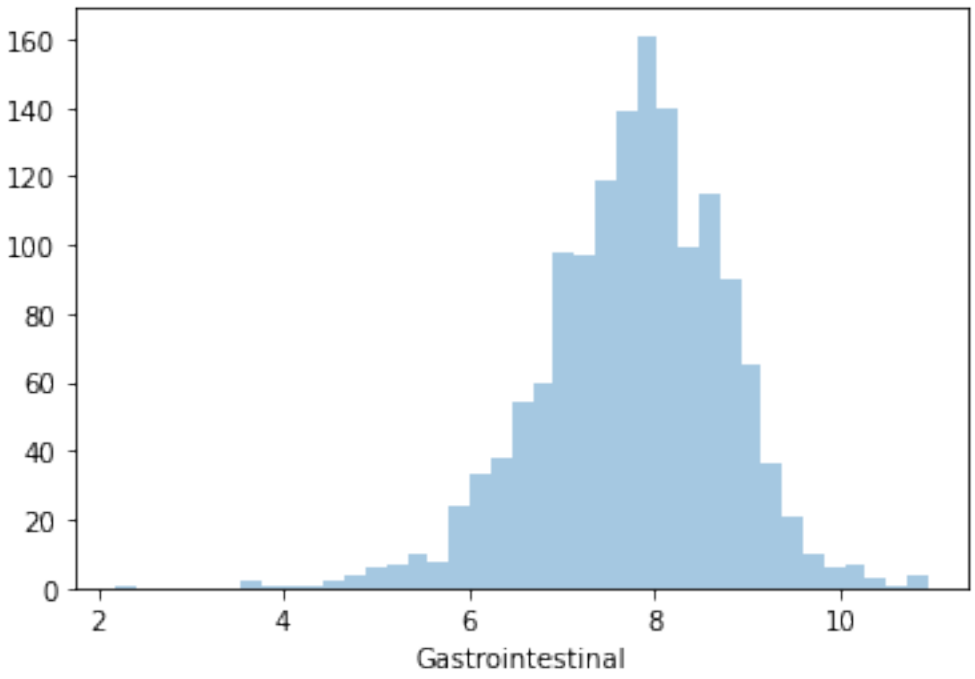
What the expression above is telling us is as follows: Each unit increase in the log of the Eye procedure count is associated to an estimated 5% increaes in the number of Gastrointestinal procedures.

1.5 What We Already Know

Another approach that we could use is to perform a multiple linear regression of the log-counts of the "Gastrointestinal" procedures on the log-counts of the other procedures.

In fact, it looks like the log-transformed "Gastrointestinal" counts are *approximately* Normally distributed.

[25]:
```
_ = sns.distplot(log_Y, kde=False)
```



[26]:
```
linear_regression = sm.OLS(log_Y, X)
```

[27]:
```
linear_regression_fit = linear_regression.fit()
```

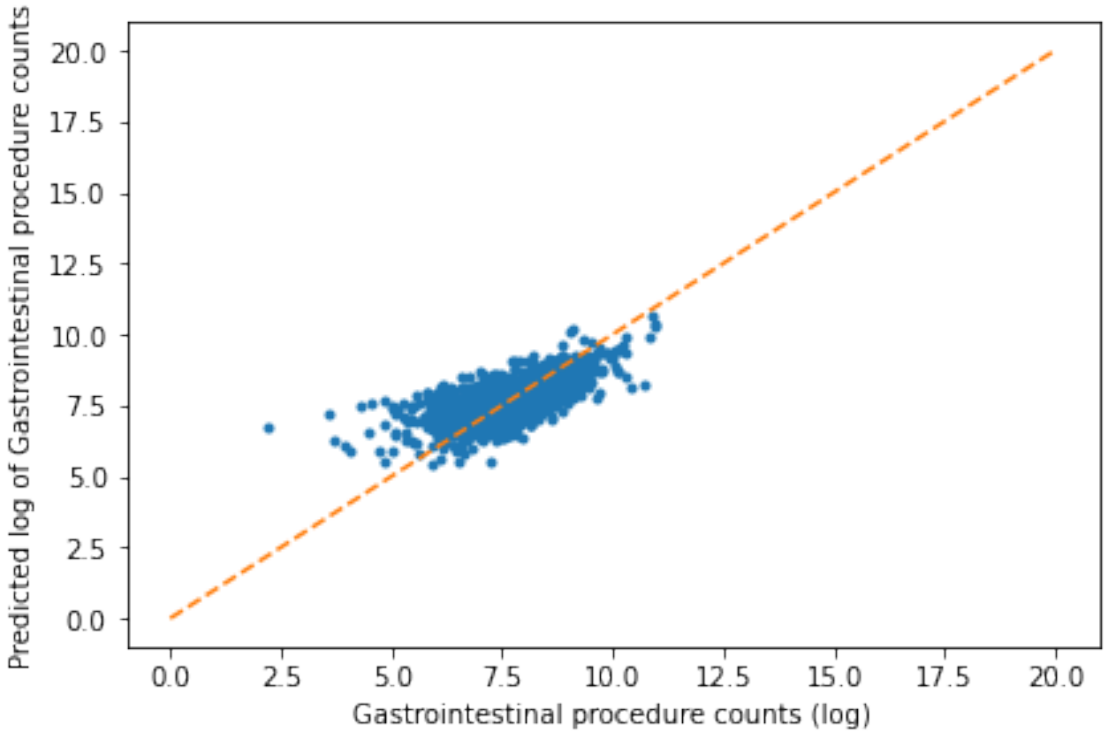
[28]:
```
linear_regression_pred = linear_regression_fit.predict(X)
```

In this case, don't forget that the model is trained to predict $\log Y$ and not Y!

[29]:
```
_plt = plt.plot(log_Y, linear_regression_pred, ".")

add_bisector(0, 20, 0, 20, linestyle="--", color=sns.color_palette()[1])

ax = plt.gca()
ax.set_xlabel("Gastrointestinal procedure counts (log)")
ax.set_ylabel("Predicted log of Gastrointestinal procedure counts")
plt.tight_layout()
```



What is the RMSE of this linear regression model?

[30]:
```
linear_regression_rmse = compute_rmse(
    HospitalProcedures["Gastrointestinal"], np.exp(linear_regression_pred)
)
```

[31]:
```
linear_regression_rmse
```

[31]:
```
3257.5571006968353
```

The linear regression model seems to be outperformed by the Poisson model by 6%-7%.

[32]:
```
linear_regression_rmse / poisson_regression_rmse
```

[32]:
```
1.067348262192912
```

1.5.1 Interpretation of the model

This time we have fitted a model for $E(\log Y \mid X)$.

Note that this is not a Generalized Linear Model for the response variable Y!!!

However, it is a linear regression for the response variable $Z = \log Y$ and therefore a Generalized Linear Model for Z (since linear regression models are a subset of the family of GLMs).

This time, we assumed that:

- $\log Y$ is approximately Normally distributed with expected value equal to $E(\log Y \mid X)$
- $E(\log Y \mid X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$

So, this time, how do we interpret, e.g., β_1?

Again, after some little algebra, we see that

$E(\log Y \mid X_1 = x + 1, X_2, \ldots, X_p) - E(\log Y \mid X_1 = x, X_2, \ldots, X_p) = \beta_1$

which means that β_1 measures the absolute variation of the expected value of $\log Y$ corresponding to a unit increase in X_1 (assuming that all other predictors are kept constant).

[33]:
```
linear_regression_fit.params
```

[33]:
```
const              3.778556
Eye                0.065907
Nervous System     0.116710
Musculoskeletal    0.097207
Skin               0.083978
Genitourinary      0.242366
Cardiovascular    -0.015102
Respiratory        0.121340
Other              0.013898
dtype: float64
```

[34]:
```
linear_regression_fit.params["Eye"]
```

[34]:
```
0.06590675799068697
```

In this example, the Poisson model provided a better fit and an easier interpretation of the coefficients.