SESUG 2024 Paper 93

SAS and Python can complement each other in statistical modeling

Valentina Grouverman, Lindzee Smith

RTI International

Abstract

Statistical modeling is a significant part of data scientists' work as well as many SAS programmers. In the past we used mostly SAS and relied on its standard procedures as regressions. Now Python can provide us additional functionality and flexibility. Its compatibility with most cloud platforms and continuously increasing number of libraries and packages available for data analysis, modeling and machine learning made Python an ideal choice for statistical modeling in cloud environments. This article illustrates how SAS and Python can complement each other in model's development when we can effortlessly switch between these languages and see benefits of this adaptable approach.

Introduction

When we work on statistical models, we always start with data analysis, learning our data, cleaning a sample, and identifying outliers. We apply techniques and tools to identify patterns, trends, and correlations. This is an important step before developing models based on sample data. SAS incorporates all essential procedures to run the descriptive statistics on the sample and is better than languages like R and Python for outputting variable frequencies and regression results. In contrast, data visualization is much easier with R and Python.

Nowadays, data collection occurs on a large scale, and we often work in big data environments where it’s more difficult to recognize trends and pattens by just running frequencies of variables. Data visualization helps to overcome this challenge by providing well-organized visual representations of sample data, making it easier to understand and analyze. Python has become a popular choice for data visualization because of its simplicity and extensive list of available libraries for this purpose.

While visualizations are also possible in SAS, they often require add-on packages to a SAS license (such as SAS/GRAPH), which can be expensive. If you are using SAS Viya, and its core SAS Visual Analytics, its browser-based tools allow your team to work on sample data visualizations. Even so, coding visualizations in SAS requires more time and effort than accomplishing the same graphs in Python, which is available at no cost.

Sample Data

This analysis utilized synthetic, public-use claims and beneficiary files, available on The Centers for Medicare & Medicaid Services (CMS) website CMS 2008-2010 Data Entrepreneurs’

Synthetic Public Use File (DE-SynPUF) | CMS. A sample of claims and beneficiaries enrolled in 2010 was used to produce these results.

Data Visualizations with Python

There are a few Python libraries (Plotly, Matplotlib, Seaborn, Boken etc.) that support various types of graphs, including line graphs, scatter and bubble plots, pie charts and bar charts, etc. Data visualizations in Python, even for big data require fewer statements than in SAS or R. This analysis used the Seaborn library to visualize the sample data. We were interested visualizing total expenditures by beneficiary age and sex, with mean, minimum and maximum range values to predict next year expenditures based on these same factors.

Figure 1 is a histogram developed using the Seaborn library. It reflects total expenditures in dollars by sex. The graph includes a density plot, showing the proportion of values in each range as well. The code to produce this graph is above the figure:

```
sns.histplot(x='total', data=df,  kde=True, hue='BENE_SEX_IDENT_CD')
plt.title(" Statistics of Total by sex")
plt.legend(title="Gender", loc="upper right")
plt.show()
```

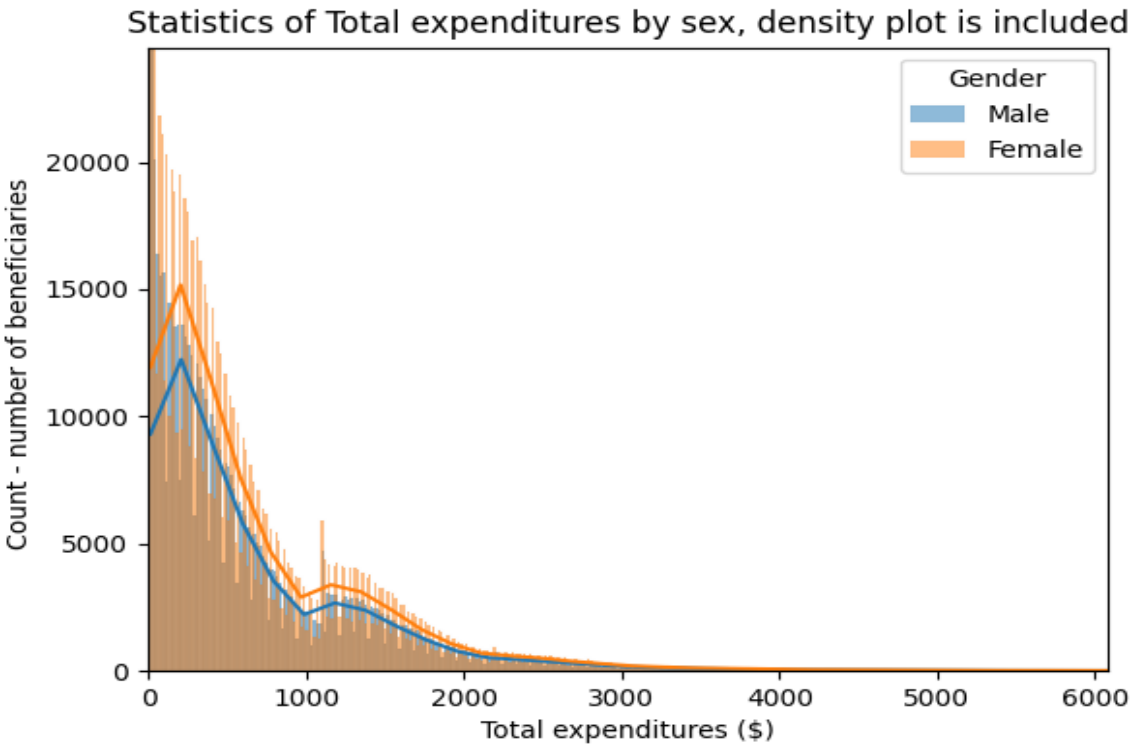

Fig.1 Total expenditures by sex with a density plot line.

Figures 2 and 3 are line plots of total expenditures by beneficiary age and sex, developed using the Seaborn library. Programming statements to produce each figure, along with the figures themselves, are below:

```
sns.lineplot(x='BENE_AGE', y='total', data=df)
plt.xlabel("Beneficiary age")
plt.ylabel("Total expenditures ($)")
plt.title(" Statistics of Total expenditures by beneficiary age, whole sample ")
plt.show()
```

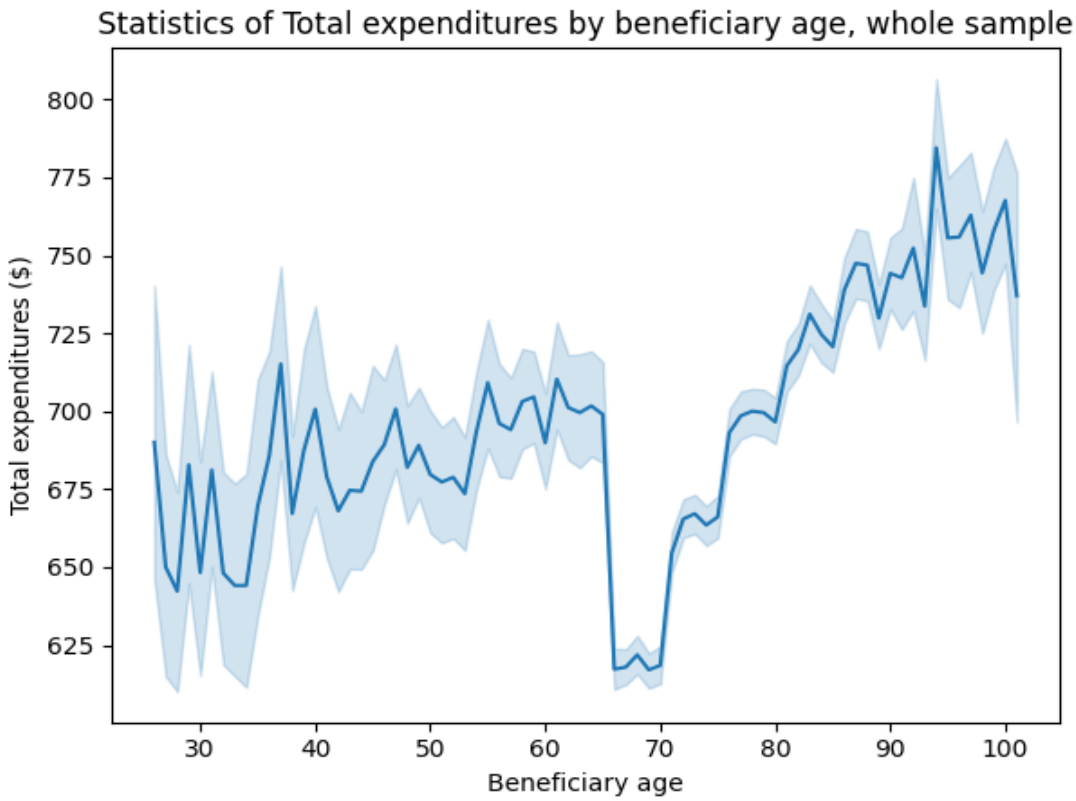

Fig.2 Total expenditures by age

```
df_gt65_f=df_gt65[df_gt65['BENE_SEX'] ==2 ]
df_gt65_m=df_gt65[df_gt65['BENE_SEX'] ==1 ]
sns.lineplot(x='BENE_AGE', y='total', data=df_gt65_f, label='Female')
sns.lineplot(x='BENE_AGE', y='total', data=df_gt65_m, label='Male')
plt.xlabel("Beneficiary age")
plt.ylabel("Total expenditures ($)")
plt.title("Statistics of Total expenditures by beneficiary age")
plt.show()
```

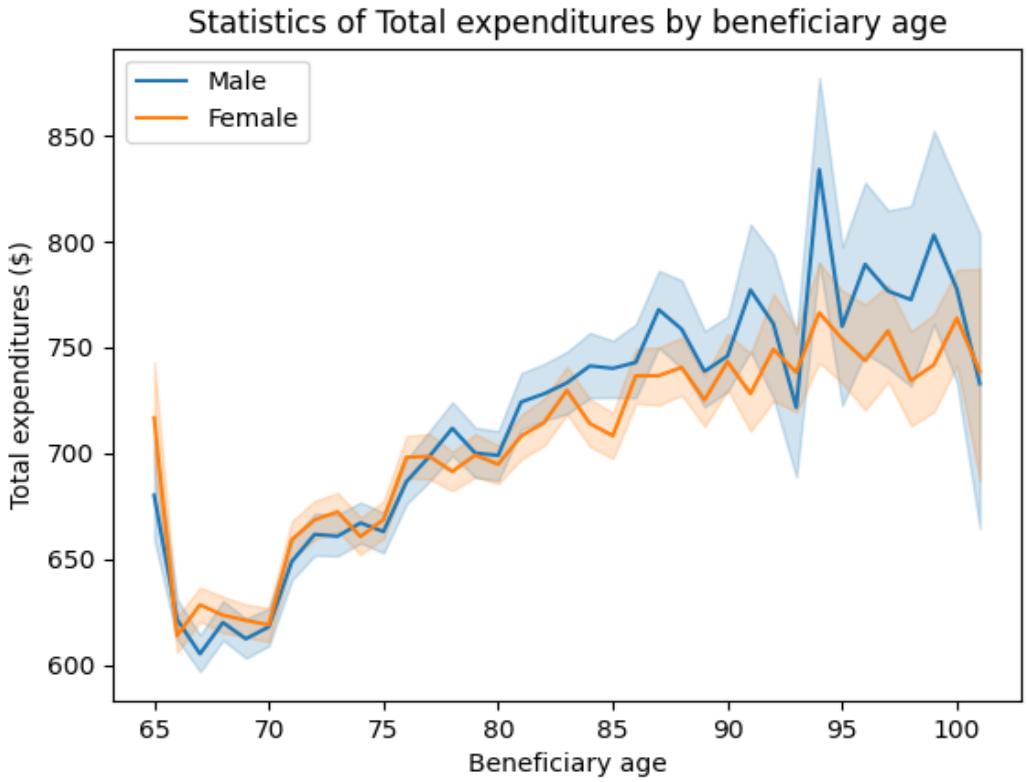

Fig. 3 Total expenditures by age and sex

Based on the relationship between total expenditures and beneficiary age, we created a model for beneficiaries older than 65 years old. We also classified each model by beneficiary sex.

```
proc reg data=sample tableout outest=temp;
    by BENE_SEX_IDENT_CD;
        model  total =  BENE_AGE
                       SP_DEPRESSN
                       SP_DIABETES
                       SP_ISCHMCHT
                       SP_OSTEOPRS
                     / noint;

    output out= sample_pred (keep=DESYNPUF_ID BENE_AGE Predicted) P=Predicted;
run;
```

With the output generated from the code above, Python libraries again could be used to visualize our model results. Figure 4 represents predicted expenditures by beneficiary age and sex, developed using SAS regression for beneficiaries older than 65 years:

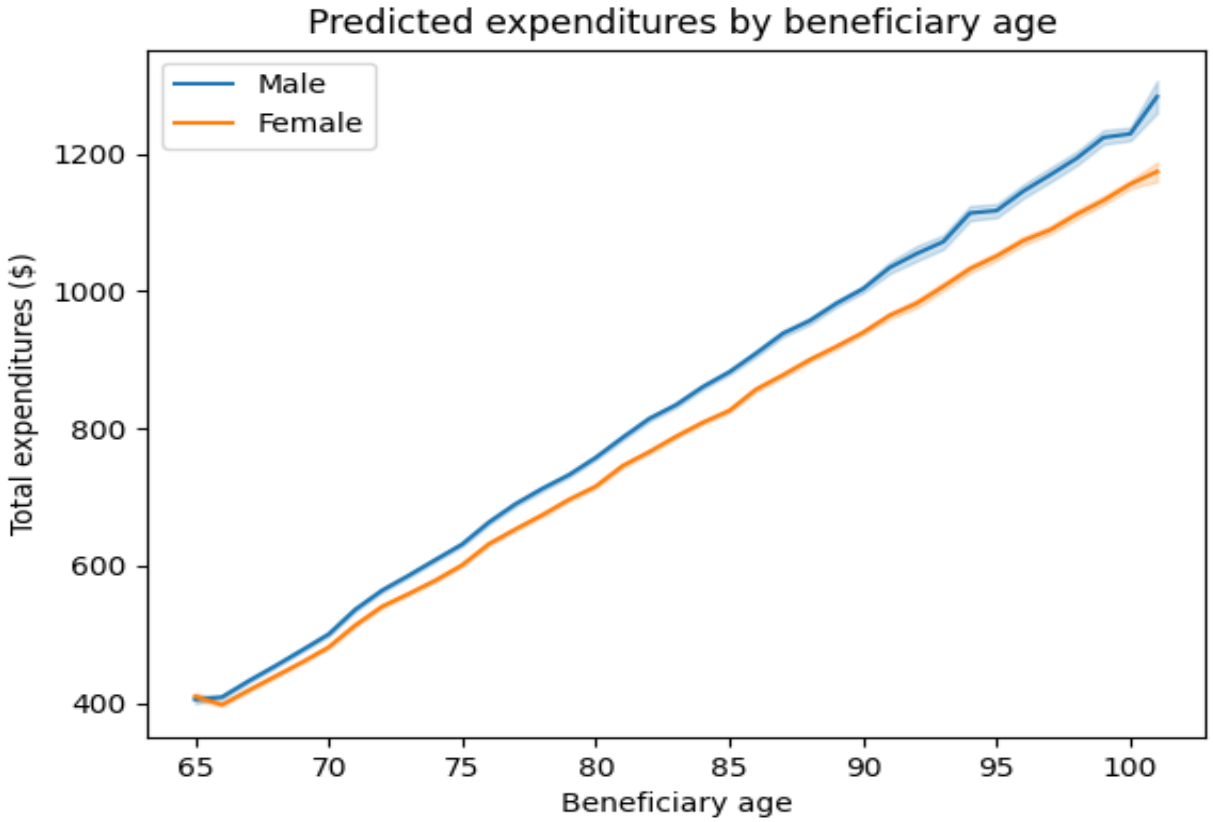

Fig. 4 Predicted total expenditures by beneficiary age and sex

Conclusion

For several decades SAS was widely used for data analysis and statistical modeling. We relied on its data manipulation capabilities with comprehensive tools and procedures to manage highly optimized complex computations and develop statistical models. Nowadays, its lack of continuous development of built-in functions and procedures, and its limited integrations with other programming languages, can be compensated for by using open-source languages like Python.

In contrast, Python is compatible with most cloud platforms, and it includes a growing number of libraries and packages suitable for data visualization and modeling. We can switch between SAS and Python programming languages to use advantage of both in our work.