SASPy – die Stärken von SAS in Python nutzen

Bernhard Müller
HMS Analytical Software GmbH
Rohrbacher Str. 26
D-69115 Heidelberg
bernhard.mueller@analytical-software.de

Zusammenfassung

SASPy ist ein Open-Source-Projekt von SAS und stellt Python-Schnittstellen zum SAS-System zur Verfügung. Mit Hilfe von SASPy kann eine SAS Sitzung in Python gestartet und Analysen mit den praxisbewährten SAS-Prozeduren durchgeführt werden.
Der Vorteil: Ein Python-Programmierer kann in seiner gewohnten Umgebung arbeiten und gleichzeitig die mächtigen Algorithmen und Modelle seiner SAS-Kollegen nutzen.
Dieser Beitrag erörtert was SASPy ist und wie man SASPy installiert und bedient. Es wird erklärt wie Daten zwischen SAS und Python transferiert werden, sowie welche SAS-Funktionalität zur Verfügung steht.

Schlüsselwörter: SASPy, Python

1 Einleitung

SASPy ist ein neues Open Source Projekt der Firma SAS um die Stärken von SAS in Python Skripte zu bringen. Python-Entwickler können auf SAS Daten und Methoden mit bekannten Objekten und Syntax zugreifen.
Es handelt sich hier um ein ganz normales Python Paket, das man mit gewohnten Python Mitteln verwalten kann. Visierte Python-Entwickler werden die Module und die Zusammenhänge ohne weiteres einsehen und verstehen können.
Das Projekt wird in GitHub verwaltet und ist ausführlich dokumentiert. Wie sonst üblich werden auch alle Issues und Bugs in GitHub erfasst und getrackt. Weil das Paket von SAS stammt, hat man auch die Möglichkeit sich mit Problemen direkt an die SAS Hotline zu wenden.
SASPy ist Open Source - jeder darf sich an dem Projekt beteiligen. Bekanntermaßen leben Open Source Projekte von der regen Beteiligung der Mitwirkenden. SASPy ist aber auch für SAS-Entwickler interessant, die ab und zu mit Python arbeiten dürfen oder müssen. Es gibt die Möglichkeit direkt auf SAS-Quellen zuzugreifen, bewährte SAS-Funktionalität anzuwenden und sogar SAS-Code direkt in Python auszuführen.

Aber besonders in dem Bereich Analytics kann man das volle Potential von Python mit den bewährten Modellierungsmethoden von SAS optimal kombinieren.

2 Installation

2.1 Voraussetzungen

Die aktuellen Voraussetzungen und Abhängigkeiten sind:

- Python 3.x
- SAS 9.4 oder SAS Viya 3.1
- Für eine Verbindung über IOM (Integrated Object Method)
  - Java auf dem Client
  - Die passenden JAR Files aus der SAS Installation

Es kann mit SAS auf jeder unterstützter Plattform verbunden werden.

2.2 SASPy installieren

SASPy ist ein ganz normales Python Paket und wird am einfachsten mit dem Paketverwaltungsprogramm `pip`[1] für Python installiert.
Zum Beispiel: Für die Installation in Microsoft Windows muss man einfach ein Kommando-Fenster öffnen und das Kommando wie folgt eingeben:

```
C:\>pip install saspy
```

Die aktuellste Version von SASPy wird standardmäßig in das Python Root-Verzeichnis installiert. Die Module befinden sich dann im Unterverzeichnis
`\Lib\site-packages\saspy`.
Auf Linux oder UNIX gibt man das Kommando auf der Kommandozeile eines Terminal-Fensters ein.
Unter Umständen will man doch eine spezifische Version von SASPy installieren. In diesem Fall kann man die gewünschte Version direkt aus dem git-Repository des SASPy-Projektes wie folgt mit `pip` holen:

```
C:\>pip install http://github.com/sassoftware/saspy/releases/saspy-
X.X.X.tar.gz
```

Es ist aber zu beachten, dass Open Source Projekte stetig im Wandel sind. Es empfiehlt sich daher immer die aktuellste offizielle Version zu verwenden.

3 Konfiguration

SASPy ist in der Lage verschiedene SAS Sitzungen auf unterschiedlichen Plattformen (Windows, Linux, Mainframe,…) zu starten. Verbinden kann man sich sowohl mit einer lokalen als auch Remote-Sitzung.

[1] Ab Python 3.4 ist dieses Modul in der Standardinstallation von Python enthalten

3.1 Access Methods

Um die verschiedenen Verbindungsarten zu ermöglichen, stellt SAS einige Access Methods zur Verfügung.

3.1.1 STDIO

Diese Methode steht nur auf Linux und UNIX zur Verfügung. SAS und Python müssen auf demselben Host installiert sein. Die relevante Konfiguration erkennt man an der Tatsache, dass die Schlüsselwörter `java` und `ssh` in der Connection Definition nicht vorkommen.

```
default  = {'saspath': '/opt/sas94/SASHome/SASFoundation/9.4/bin/sas_u8',
            'options' : ["-fullstimer"]
            }
```

3.1.2 STDIO over SSH

Diese Methode steht nur auf Linux und UNIX zur Verfügung. Hiermit kann man mit SAS auf einem Remote-Host verbinden, vorausgesetzt der Benutzerkonto ist für „passwordless“ SSH eingerichtet. Das Schlüsselwort `ssh` wird in dieser Methode als Trigger verwendet.

```
ssh      = {'saspath' : '/opt/sas94/SASHome/SASFoundation/9.4/bin/sas_u8',
            'ssh'     : '/usr/bin/ssh',
            'host'    : 'remote.linux.host',
            'options' : ["-fullstimer"]
           }
```

3.1.3 IOM

IOM (Integrated Object Method) unterstützt die Verbindung mit SAS auf allen unterstützten Plattformen. Es ist auch die bevorzugte Methode für die Verbindung mit einer lokalen SAS Sitzung auf Windows. Das Schlüsselwort für diese Methode ist `java`.

Diese Methode braucht folgendes:

- SAS Java IOM Client (die betroffenen jar-Files können aus der SAS Installation auf dem Client kopiert werden und sind Plattformunabhängig, d.h. sie funktionieren sowohl in Windows als auch auf UNIX und Linux).
- Die Einstellung CLASSPATH mit Einbindung des SAS Java IOM Clients, des Files `saspyiom.jar`, und optional der clientseitigen Verschlüsselungsfiles.

3.2 Die Konfigurationsdatei

Trotz verschiedenen Verbindungsmethoden braucht man nur eine einzelne Konfigurationsdatei. Die Konfigurationsdatei `sascfg.py` aus dem Repository wird Default mäßig mitinstalliert. Wo sie abgelegt wird kann wie folgt in Python ermittelt werden:

```
>>>import saspy
>>>saspy.SAScfg
```

Es ist aber besondere Vorsicht mit dieser Konfigurationsdatei geboten. Weil sie aus dem git-Repository stammt, wird sie bei jedem Upgrade oder bei jeder Neuinstallation überschrieben. Anpassungen zu dieser Datei sind daher nicht fortdauernd.

Um Ihre eigene Konfiguration permanent und geschützt abzulegen, empfiehlt SAS die Erstellung einer Kopie der Beispielkonfiguration `sascfg.py`. Bitte darauf achten, dass diese Kopie einen ganz bestimmten Name haben muss: `sascfg_personal.py`. Der vorgegebene Name ist deshalb sehr wichtig, weil SAS zunächst nach dieser sucht und erst wenn nicht vorhanden, nach der Konfigurationsdatei `sascfg.py`. Die „persönliche“ Konfigurationsdatei kann in einem beliebigen Verzeichnis abgelegt werden, der Pfad muss aber dann in der Umgebungsvariable PYTHONPATH gesetzt sein.

Falls eine `sascfg_personal.py` vorhanden und korrekt eingebunden ist, wird sie beim Aufruf `saspy.SAScfg` angezeigt.

3.3 Zusammenstellung der Konfigurationsdatei

Die Konfigurationsdatei besteht aus drei Hauptteilen:

- `SAS_config_names`
  Eine Liste der Connection Definitionen, die überhaupt verwendet werden dürfen. Ein Administrator kann auf diesem Weg bestimmte Connections einschränken, obwohl sie alle in der Konfigurationsdatei definiert sind.
- `SAS_config_options`
  Aktuell gibt es nur die eine Option `lock_down`, die steuert ob man die Einstellungen zur Laufzeit aufheben kann.
- Connections
  Hier werden die verschiedenen Connections definiert. Sie müssen nicht alle aktiv sein, da man über `SAS_config_names` steuert welche überhaupt verwendet werden dürfen.

3.4 Weitere Konfigurationen

Für die Verbindung mit einer lokalen SAS-Sitzung ist noch einiges zu beachten:

- Die Datei `sspiauth.dll` muss in der System PATH Umgebungsvariable stehen. Diese Datei ist Teil der SAS-Installation und unter `SASHome\SASFoundation\9.4\core\sasext` zu finden. In der PATH Umgebungsvariable soll lediglich der Pfad angegeben werden ohne Angabe der Datei selbst.
- Unter Umständen wird in die Windows-Registry nach einem eingetragenen SAS-Aufruf nachgeschaut. Falls die SAS-Installation nicht ganz sauber gelaufen ist und die Registry-Einträge fehlen oder falsch angelegt wurden, kann es zu einem Fehler beim Starten von SAS kommen. Es empfiehlt sich SAS dann neu zu installieren.

4 Die SAS Funktionalität

Eine SAS-Sitzung wird wie folgt gestartet:

```
>>>import saspy
>>>sas = saspy.SASsession(cfgname='winlocal')
```

Die erfolgreiche Verbindung wird im Log bestätigt:

```
Using SAS Config named: winlocal
SAS Connection established. Subprocess id is 5161
```

Für SAS-Entwickler besteht die Möglichkeit SAS-Code direkt in Python auszuführen:

```
>>>from IPython.display import HTML
>>>sascode = sas.submit("""
proc means data=sashelp.class nway;
    class sex;
    var weight height;
run;
""")
>>>HTML(sascode['LST'])
```

Die Darstellung des Ergebnisses wird in diesem Fall mit dem Display-Werkzeug von IPython realisiert.

The SAS System

The MEANS Procedure

Sex	N Obs	Variable	N	Mean	Std Dev	Minimum	Maximum
F	9	Weight	9	90.1111111	19.3839137	50.5000000	112.5000000
		Height	9	60.5888889	5.0183275	51.3000000	66.5000000
M	10	Weight	10	108.9500000	22.7271864	83.0000000	150.0000000
		Height	10	63.9100000	4.9379370	57.3000000	72.0000000

SASPy stellt einige Methoden bereit für die Arbeit mit SAS-Daten, die sehr ausführlich in der API-Referenz dokumentiert sind.

- Referenzieren einer bestehenden SAS-Tabelle:

```
>>>cl = sas.sasdata('class', 'sashelp')
```

- Abruf der Attribute einer SAS-Tabelle:

```
>>>cl.columnInfo()
```

	Member	Num	Variable	Type	Len	Pos
0	SASHELP.CLASS	3	Age	Num	8	0
1	SASHELP.CLASS	4	Height	Num	8	8
2	SASHELP.CLASS	1	Name	Char	8	24
3	SASHELP.CLASS	2	Sex	Char	1	32
4	SASHELP.CLASS	5	Weight	Num	8	16

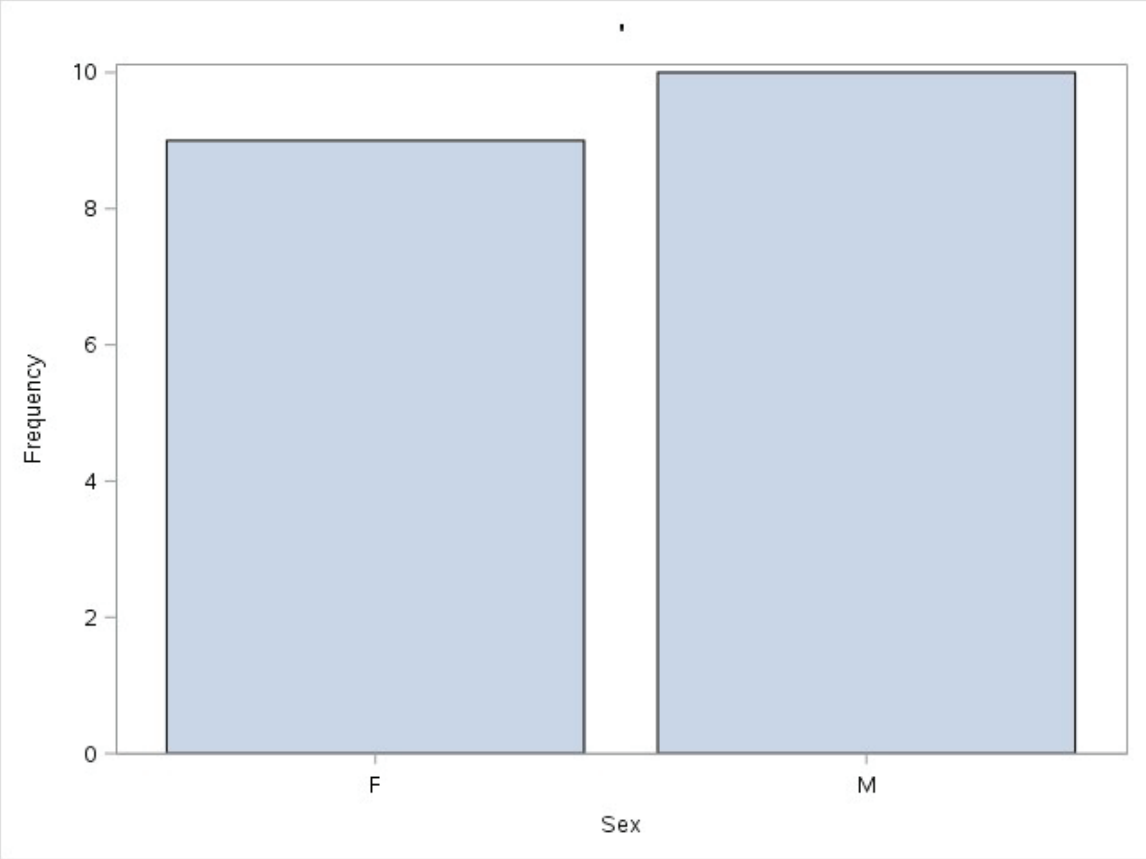
- Berechnung einfacher Statistiken:

```
>>>cl.means()
```

	Variable	N	NMiss	Median	Mean	StdDev	Min	P25	P50	P75	Max
0	Age	19	0	13.0	13.315789	1.492672	11.0	12.0	13.0	15.0	16
1	Height	19	0	62.8	62.336842	5.127075	51.3	57.5	62.8	66.5	72
2	Weight	19	0	99.5	100.026316	22.773933	50.5	84.0	99.5	112.5	150

- Erstellung einfacher Grafiken:

```
>>>cl.bar('sex')
```



5 Daten zwischen SAS und Python austauschen

Daten werden zwischen SAS und Python über Pandas Dataframes ausgetauscht. Mit der Methode `in_pd = pd.read_csv('myCSVFile')` werden die externen Daten zunächst in einen Dataframe eingelesen. Die Methode `in_sas = sas.df2sd(in_pd)` wandelt den Dataframe in eine SAS-Tabelle um. Es gibt selbstverständlich auch eine Methode für die Umwandlung einer SAS-Tabelle in einen Dataframe (`sas.sd2df`). Es werden hier jeweils die Kurzform der Methoden verwendet: `df2sd (dataframe2sasdata)` und `sd2df (sasdata2dataframe)`.

6 Analytics

Ein wichtiger Einsatz von SASPy ist für die analytische Modellierung. SAS bietet von Haus aus diese Möglichkeit für diverse Bereiche wie Statistik, Machine Learning, usw. an. Die Funktionalitäten sind ähnlich organisiert wie in SAS. Bitte beachten Sie aber, dass nur ein lizenziertes Paket von SAS hier zur Verfügung stehen kann.

- STAT (SAS/STAT)
- ETS (SAS/ETS)
- Machine learning (SAS Enterprise Miner)
- QC (SAS/QC)
- UTIL (SAS Base procedures)

SAS stellt hochwertige Methoden für die analytischen Funktionen bereit, die sehr ähnlich wie die SAS-Prozeduren aufgebaut sind. Mit der Funktion `dir()` kann man alle Methoden eines Produkts auflisten. Nicht alle sind gleich mit einer Prozedur zu setzen, die meisten aber schon.

Beispiel: STAT

```
>>>stat = sas.sasstat()
>>>dir(stat)

 'factor',
 'glm',
 'hplogistic',
 'hpreg',
 'hpsplit',
 'logger',
 'logistic',
 'mixed',
 'phreg',
 'reg',
 'sas',
 'sasproduct',
 'tpspline',
 'ttest'
```

Beispiel: ETS

```
>>>stat = sas.sasstat()
>>>dir(stat)

 'arima',
 'esm',
 'logger',
 'sas',
 'sasproduct',
 'timedata',
 'timeid',
 'timeseries',
 'ucm'
```

6 Fazit

Das Projekt SASPy ist gelungen und wird stetig verbessert. Die beiden Welten SAS und Python haben sich damit sehr viel angenähert. Die Zusammenarbeit zwischen SAS- und Python-Teams im analytischen Bereich wurden hierdurch gestärkt.

Literatur

[1] SASPy offizielle Dokumentation: https://sassoftware.github.io/saspy/index.html

[2] SASPy API-Referenz: https://sassoftware.github.io/saspy/api.html

[3] SASPy Projekt-Repository: https://github.com/sassoftware/saspy