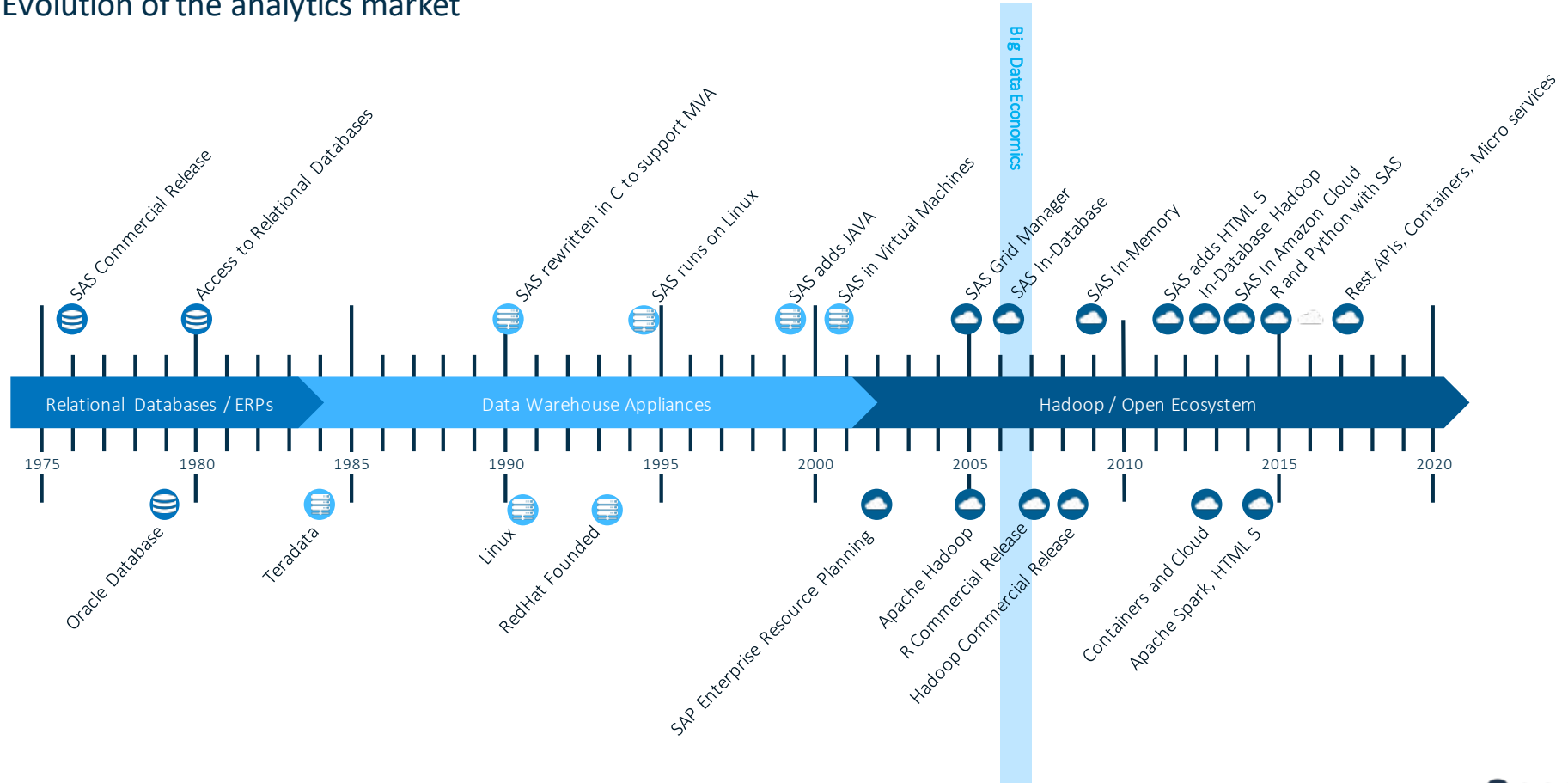




The New Languages of SAS 9.4 and SAS Viya: A SAS Programmer's Primer

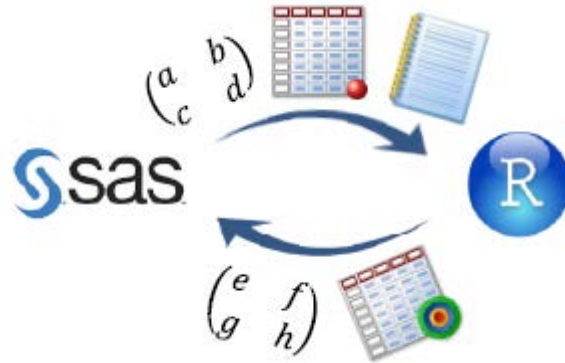
SAS OPEN TIMELINE

Evolution of the analytics market



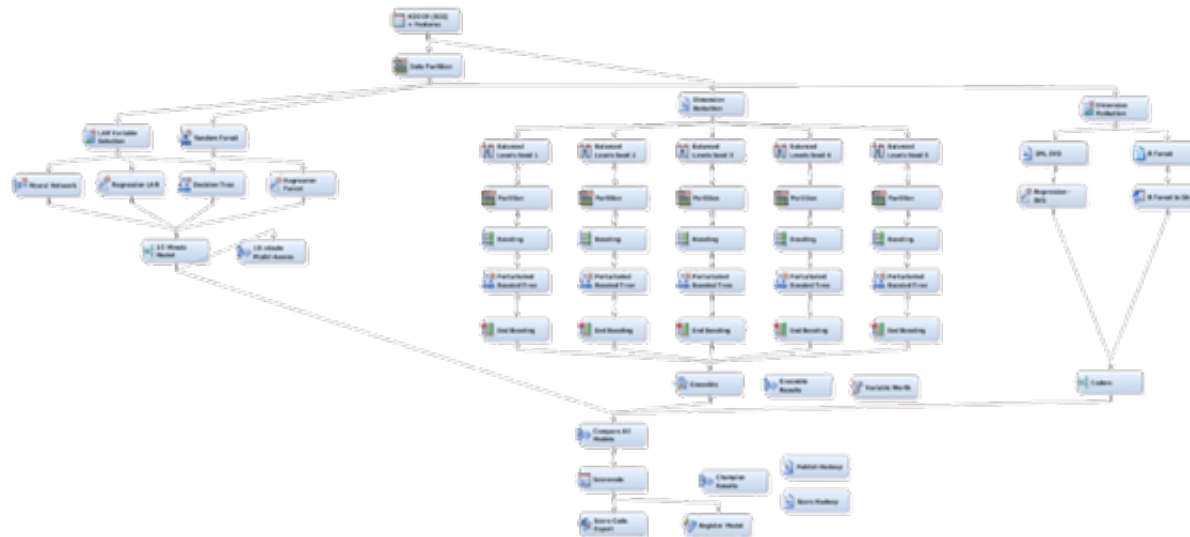
Using SAS/IML to Interact with R

- Send IML matrices and SAS data sets to R
- Submit R code in the IML script
- Return R results as IML matrices or SAS data sets



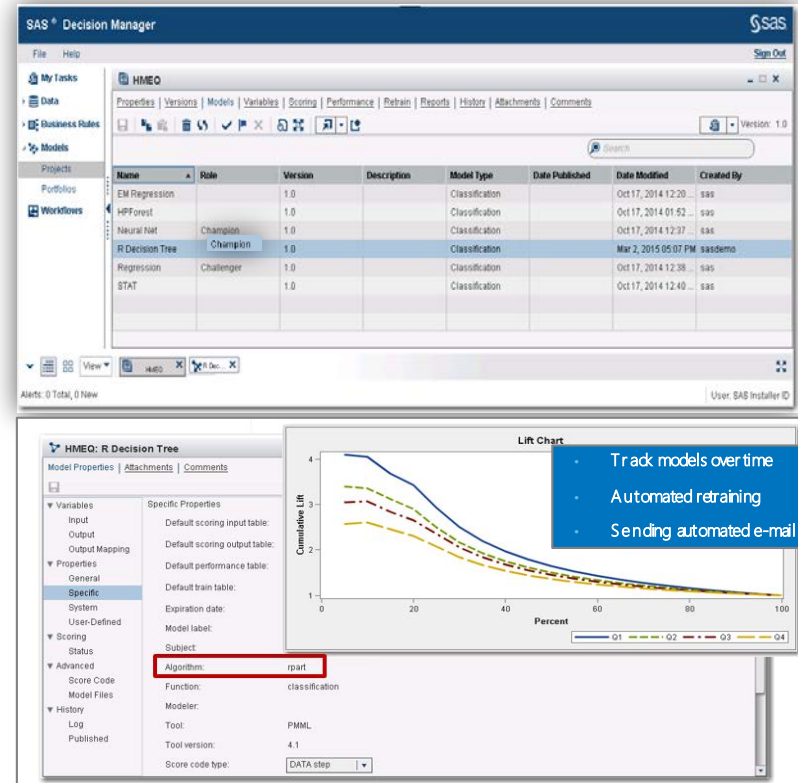
Using SAS Enterprise Miner with Python and R

- Leverage SAS advanced analytics in a easy-to-use GUI environment
- Leverage R & Python modeling packages from a SAS environment
- Compare & Ensemble R, Python, and SAS models in one interface



Using SAS Model Manager to Organize R and Python

- SAS Model Manager streamlines the steps of creating, managing, deploying, monitoring, and operationalizing analytic models
- Can accept R & Python models



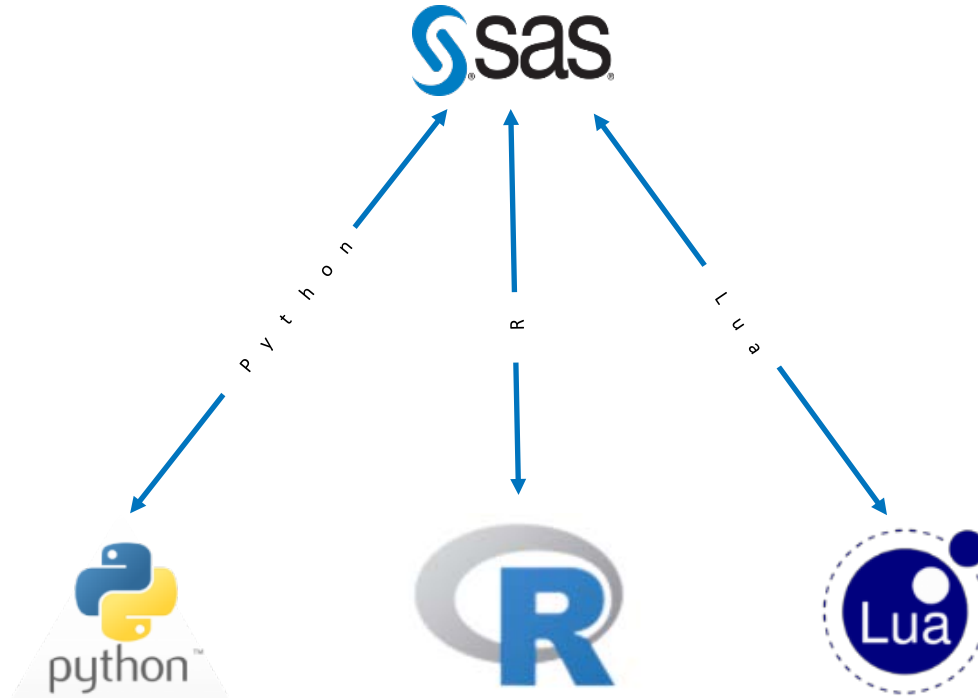
PROC LUA

- PROC LUA lets you submit Lua code from within a SAS program

```
proc lua;  
  submit;  
    local fish=sas.read_ds("sashelp.fish")  
    print("fish=", table.tostring(fish))  
  endsubmit;  
run;
```

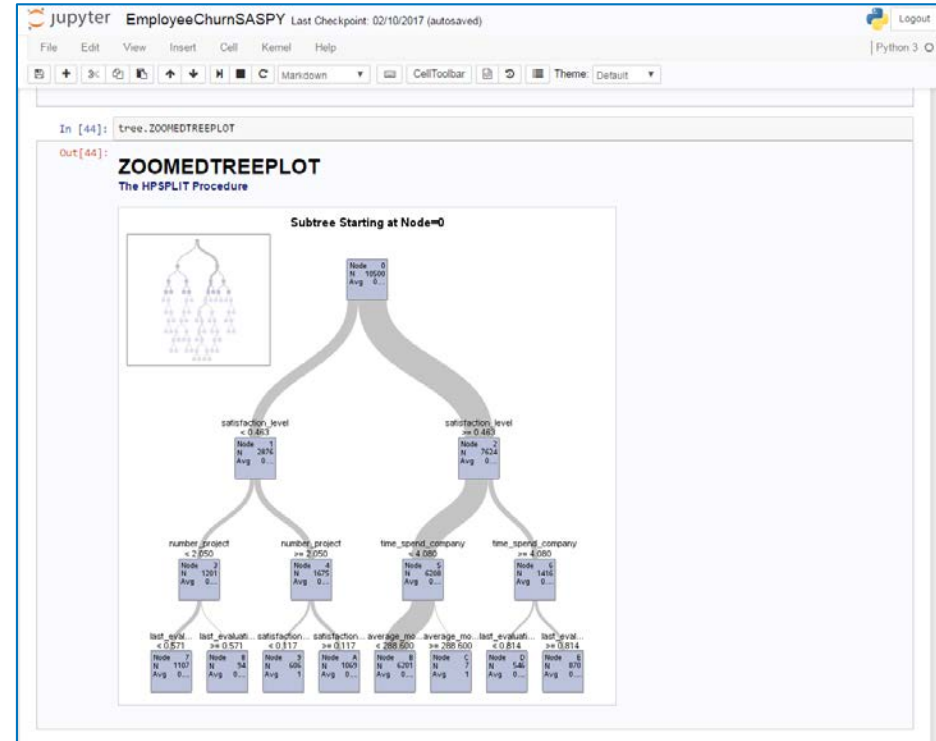


Calling Code and Exchanging Data and Results



SASPy

- Provides Python APIs to the SAS system
- Run SAS code from Python
- Run analytics from Python using object-oriented methods
- Can transfer between SAS data sets and Pandas data frames



SASPy

```
In [13]: #import SAS package and start SAS Session  
from saspy import *
```

```
In [14]: #move the pandas data to a SAS data object  
sas_dta=sas.dataframe2sasdata(dta, 'dta')
```

```
In [15]: sas_dta.head()
```

Out[15]:

SAS Output

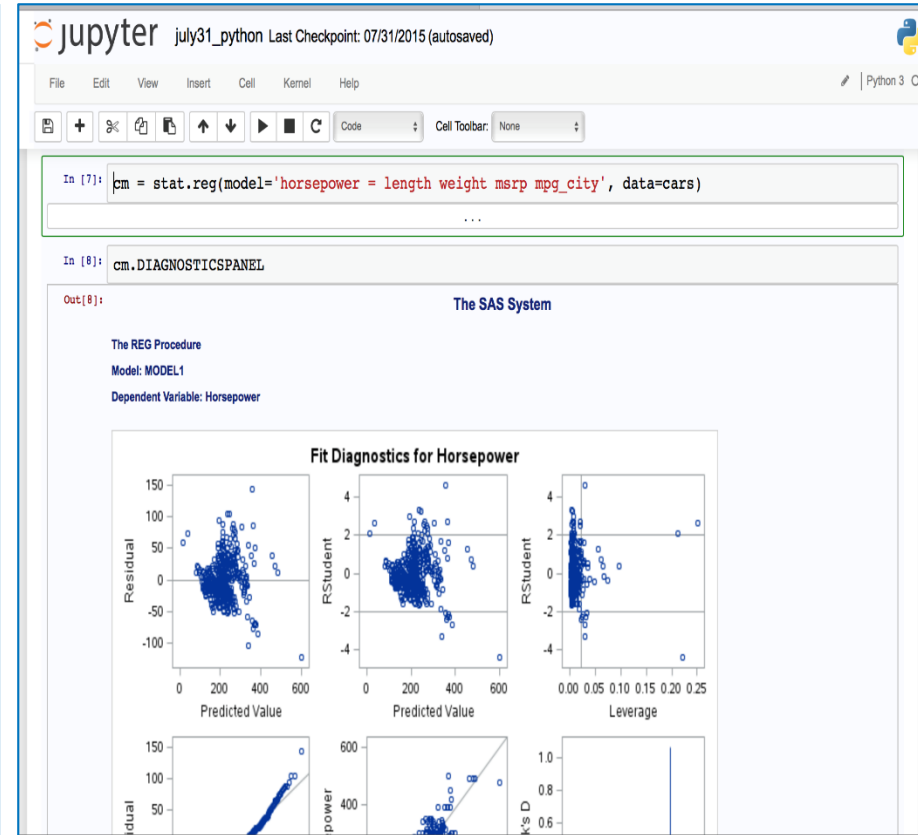
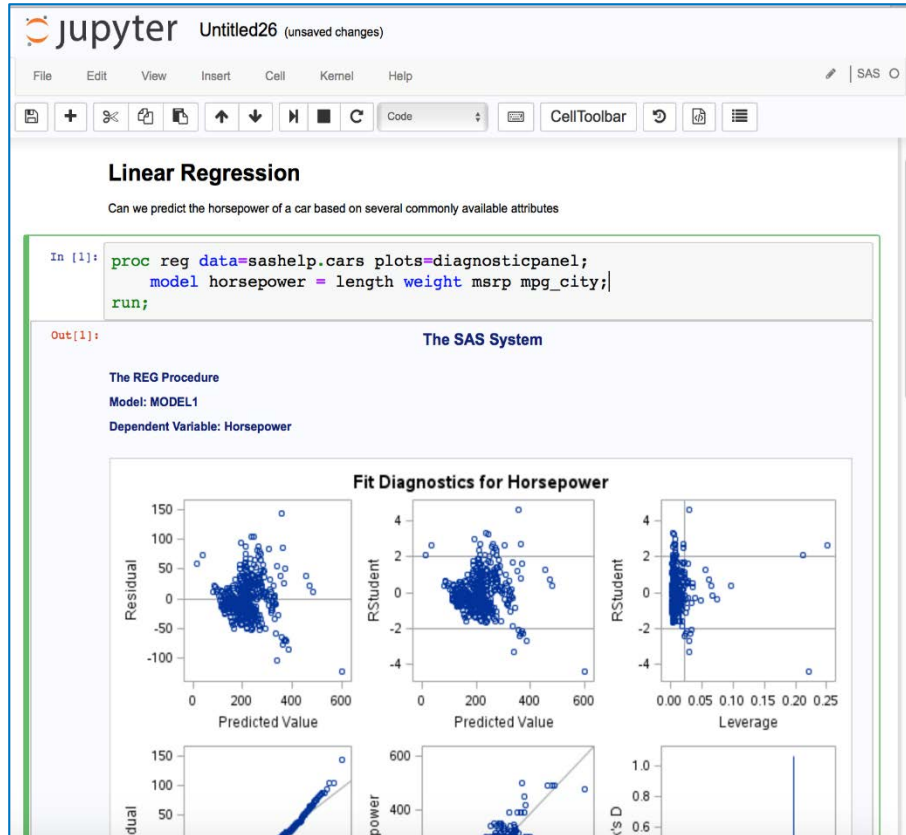
The SAS System

The PRINT Procedure

Data Set WORK.DTA

Obs	rate_marriage	age	yrs_married	children	religious	educ	occupation	occupation_husb	affairs	affair
1	3	32	9.0	3	3	17	2	5	0.11111	1
2	3	27	13.0	3	1	14	3	4	3.23077	1
3	4	22	2.5	0	1	16	3	5	1.40000	1
4	4	37	16.5	4	3	16	5	5	0.72727	1
5	5	27	9.0	1	1	14	3	4	4.66667	1

SASPy



Viewing the SAS Code Behind SASPy

```
In [12]: sas.teach_me_SAS(True)
sas_dta.hist('educ',title='Histogram of Education', label='Education Level')
sas.teach_me_SAS(False)
```

Lastly executed on Fri, Oct 30 2015 at 11:35 AM in 0 s

```
proc sgplot data=work.dta;
  histogram educ / scale=count legendlabel='Education Level';
  title "Histogram of Education";
  density educ;
run;
title "";
```

SASPy Available on GitHub

 [Features](#) [Business](#) [Explore](#) [Marketplace](#) [Pricing](#)

This repository

[Sign in](#) or [Sign up](#)

 [sassoftware](#) / [saspy](#)

[Watch](#) 19 [Star](#) 45 [Fork](#) 25

[Code](#) [Issues](#) 6 [Pull requests](#) 0 [Projects](#) 0 [Insights](#)

A Python interface module to the SAS System. It works with Linux, Windows, and mainframe SAS. It supports the `sas_kernel` project (a Jupyter Notebook kernel for SAS) or can be used on its own. <https://sassoftware.github.io/saspy>

617 commits

2 branches

3 releases

4 contributors

Branch: **master** [New pull request](#) [Find file](#) [Clone or download](#)

 **tomweber-sas** clean up trailing blank in type column Latest commit `a049918` 10 days ago

 saspy	clean up trailing blank in type column	10 days ago
 .gitignore	add gitignore back in	3 months ago
 CONTRIBUTING.md	Add contributing file after learning a little markup	11 months ago
 ContributorAgreement.txt	This got lost somewhere along the way	11 days ago
 MANIFEST.in	fix manifest	3 months ago
 README.md	remove saspy from doc per legal	29 days ago
 __init__.py	update setup.py for win and osx installs plus version numbers	3 months ago
 saspy_example_github.ipynb	example saspy Jupyter notebook	2 months ago
 setup.py	fix typo	25 days ago



Copyright © SAS Institute Inc. All rights reserved.

MULTIPLE INTERFACES, SINGLE CODE BASE

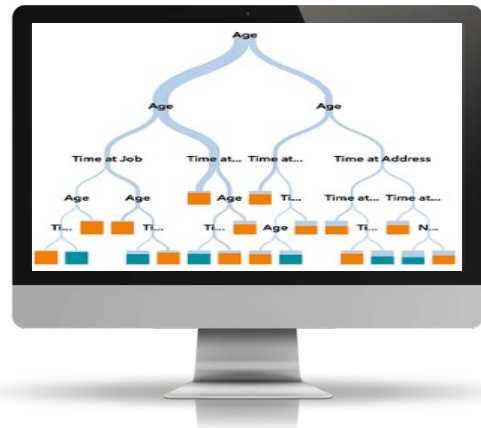
Visual Interfaces



Programming Interfaces



API Interfaces



Source-based Engines

In-Stream



In-Hadoop



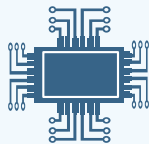
In-Database



Parallel & Serial, Pub / Sub,
Web Services, MQs

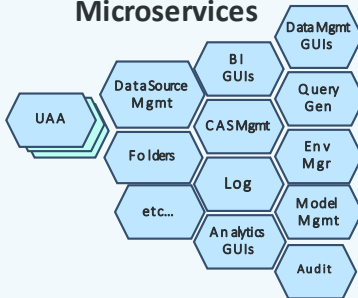
SAS Viya™

In-Memory Runtime Engine



Cloud Analytics Services (CAS)

Microservices



Solutions

Customer Intelligence



Analytics



Risk Management



Business Visualization



Fraud and Security Intelligence



Data Management



APIs



Platform



CLOUDFOUNDRY



docker



ANSIBLE



rpm



Google Cloud Platform



openstack™
CLOUD SOFTWARE

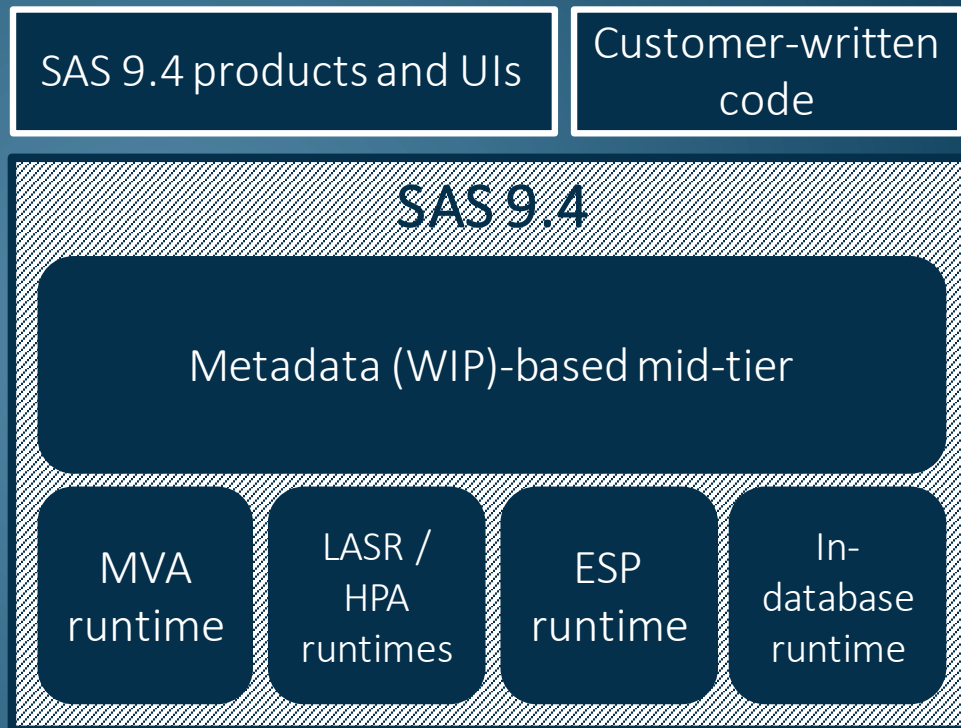


CLOUD

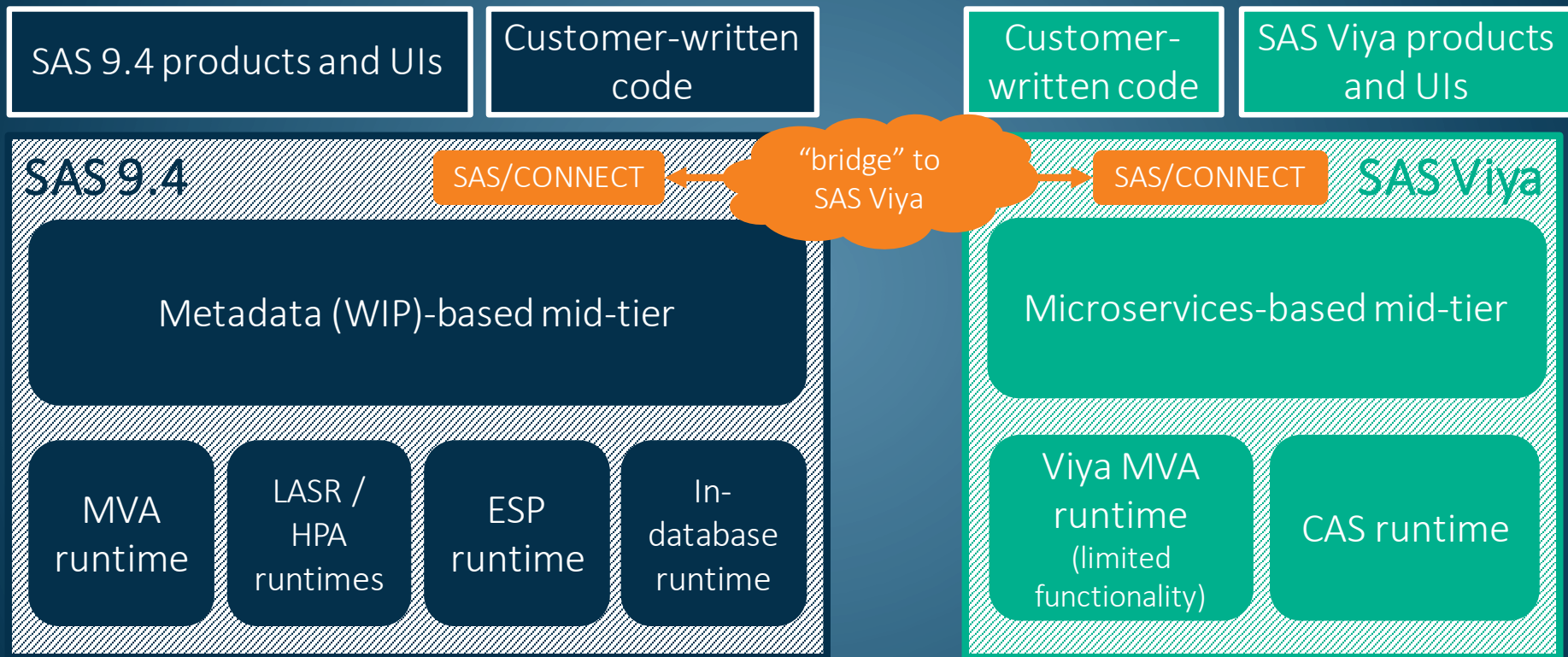


The SAS 9.4 platform

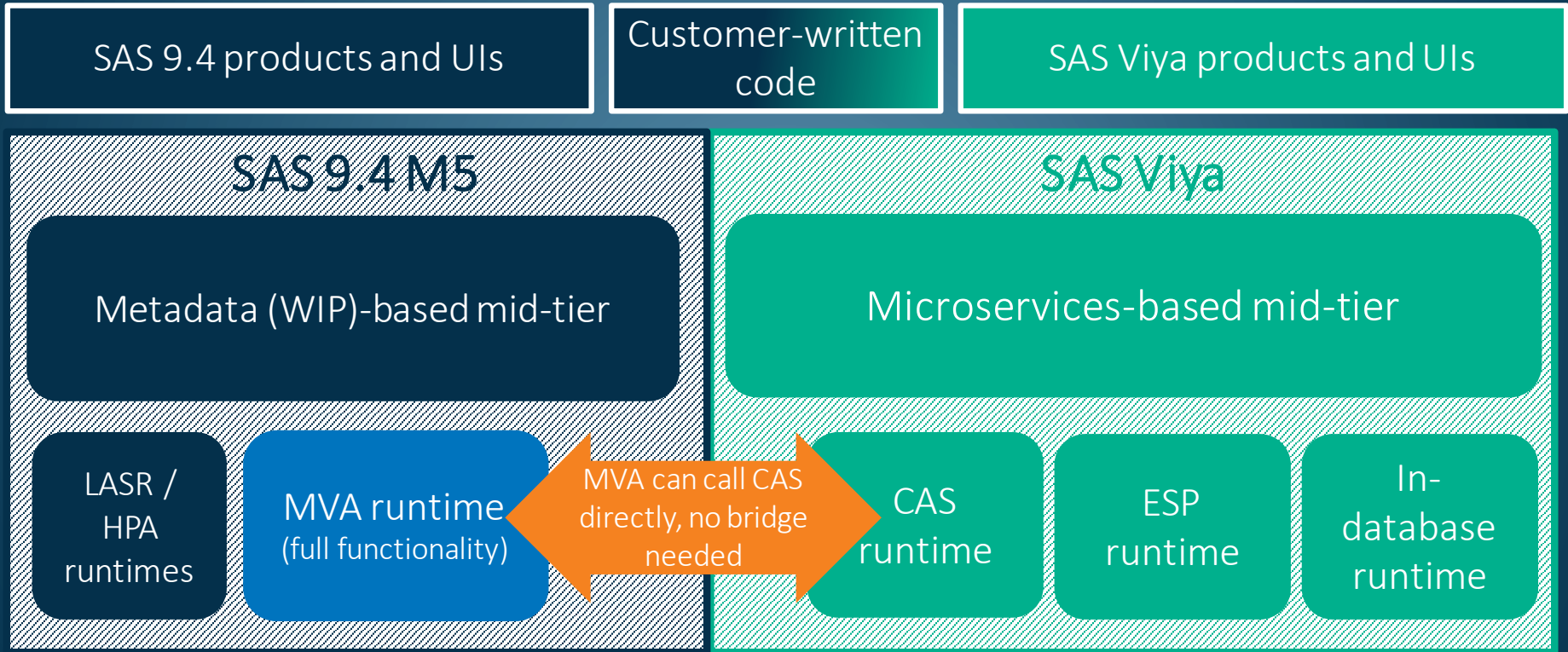
- Support for any analytics use case
- Broad and extensive customer base in production
- Modernization needed to adapt to changing technology, business trends



Prior to 9.4M5 architecture

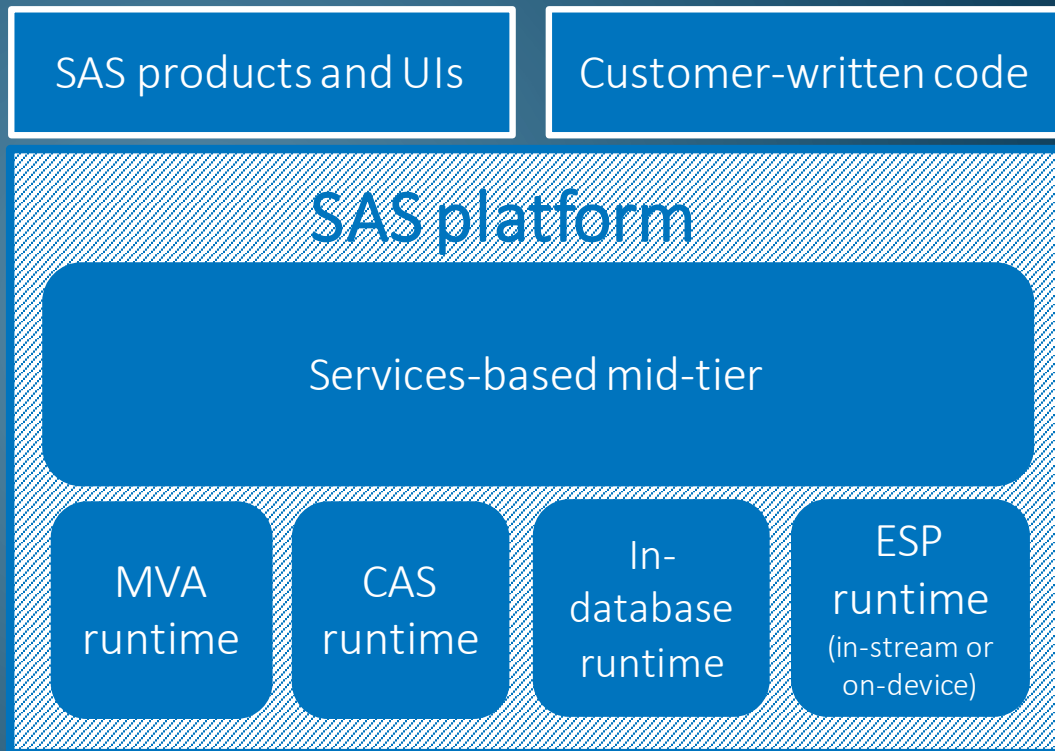


Today's architecture

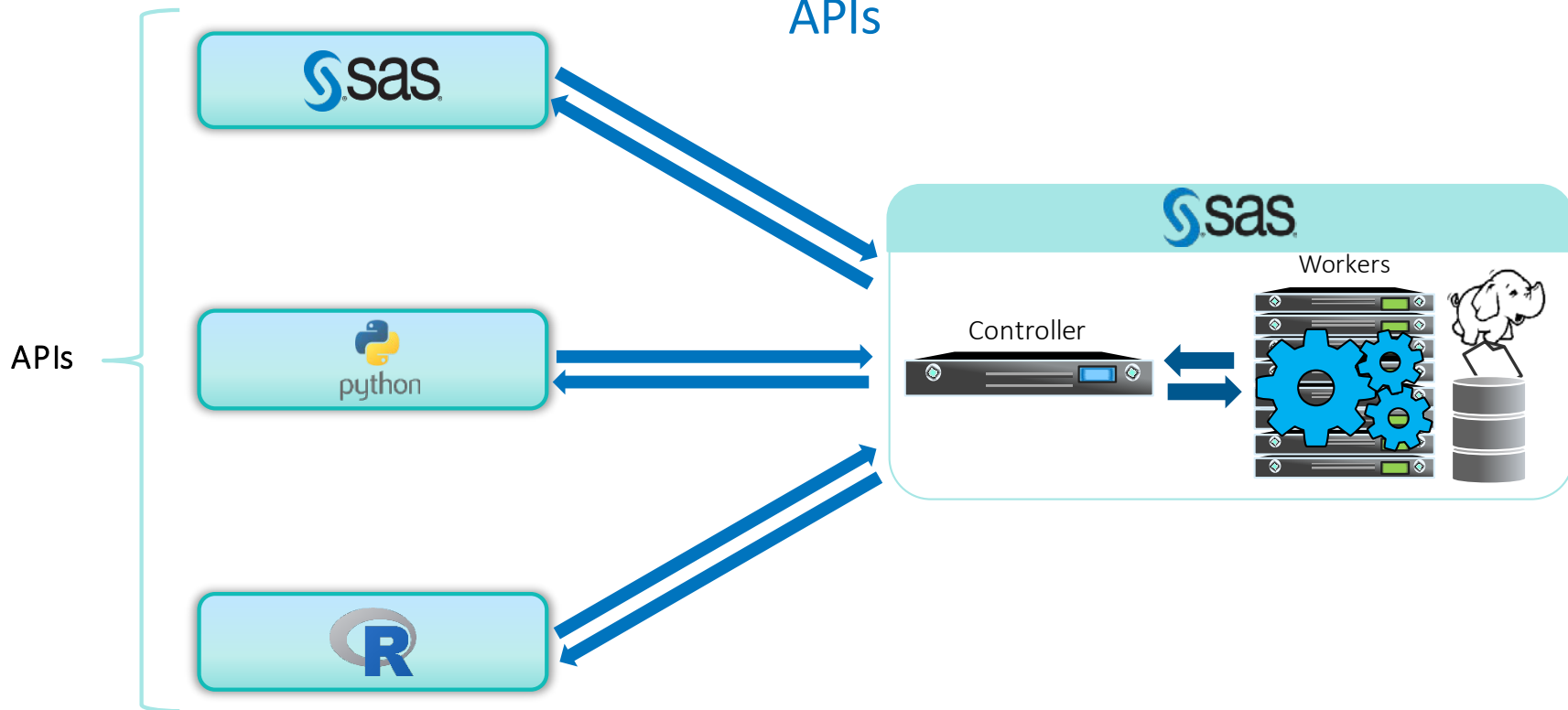


The SAS Platform – Projected Architecture

- Single platform blending v9, Viya features
- Support for pre-existing 9.4, Viya-based code
- Support for single, multi-user scenarios (multi-tenant)
- Simplified, updatable deployment
- Elastic, fault-tolerant operation

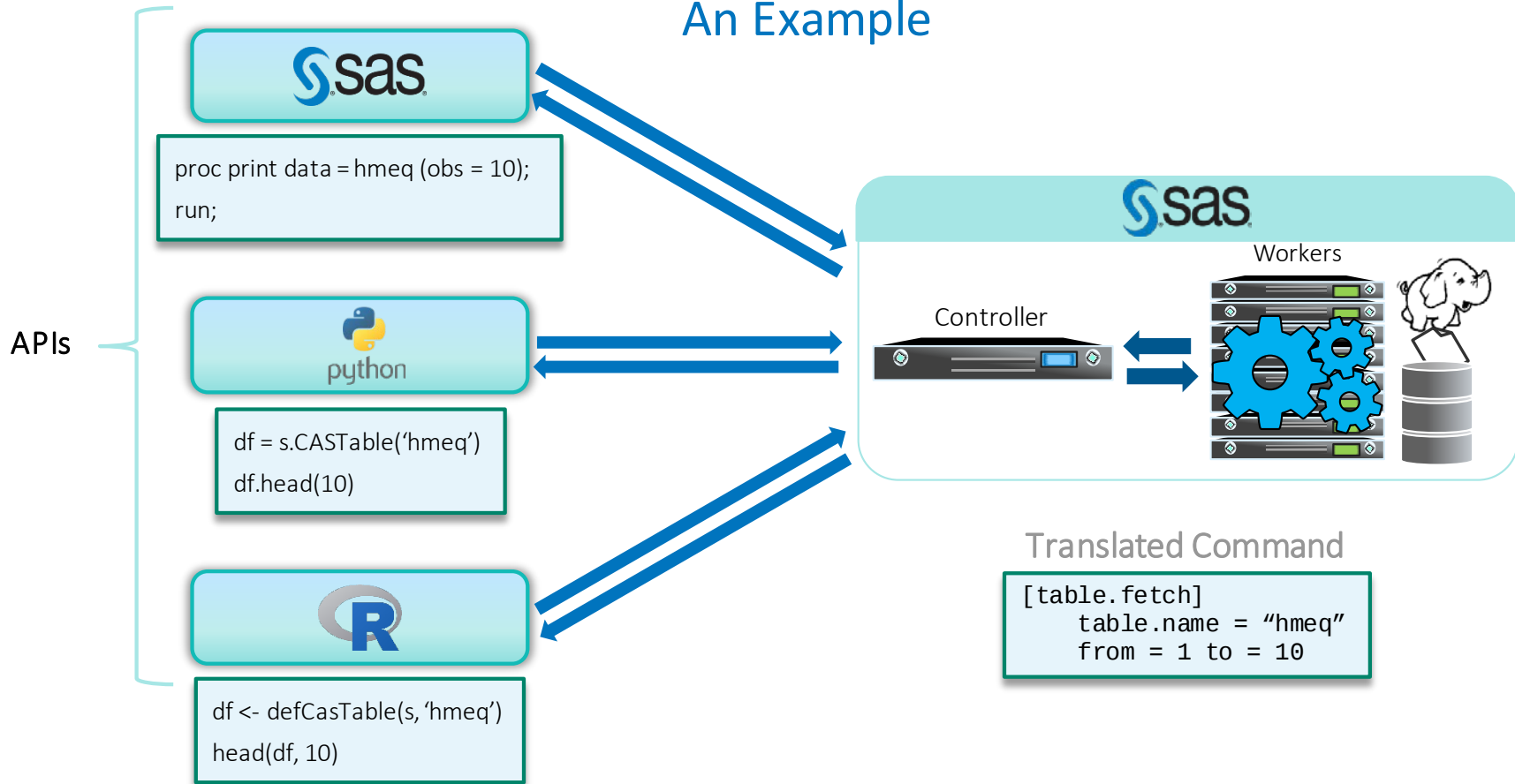


SAS Viya APIs



SAS Viya

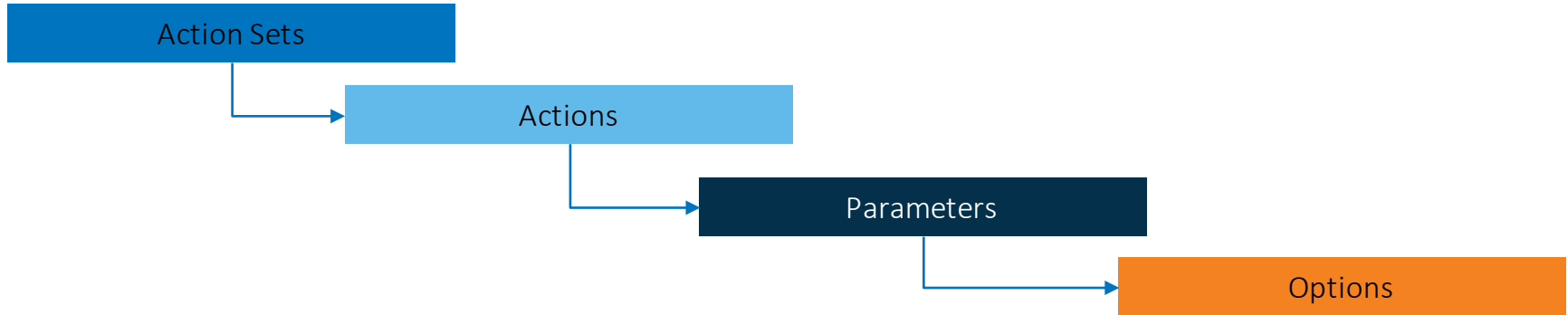
An Example



Simple CAS Actions Example

```
loadtable/path="datasources/cars.csv"  
  importOptions={fileType="csv"},  
  caslib="CASTestTmp";  
run;  
  
histogram result=hist_result/nBins=1  
  table={name="datasources.CARS"  
    caslib="CASTestTmp"  
    varList={{name="sales"}}};  
run;
```

CAS Actions Hierarchies



```
table.attribute <result =results> <status=rc> /
```

```
attributes={{
```

```
  column="string",
```

```
  * key="string",
```

```
  value="string" | 64-bit-integer | integer | double | binary-large-object
```

```
}, {...}}
```

PROC versus CAS Action

```
proc factmac data=mycas.movlens nfactors=10 learnstep=0.15
    maxiter=20 outmodel=mycas.factors;
  input userid itemid /level=nominal;
  target rating /level=interval;
  output out=mycas.out1 copyvars=(userid itemid rating);
run;
```

```
proc cas;
  action factmac result=R / table={name="movlens"},
  outModel={name="factors_out", replace=true},
  inputs={"userid", "itemid"},
  nominals={"userid", "itemid"},
  target="rating",
  maxIter=20, nFactors=10, learnStep=0.15,
  output={casout={name="score_out", replace="TRUE"},
  copyvars={"userid","itemid","rating"}};
run;
```

Cloud Analytic Services Language (CASL)

- CASL is the language specification that enables you to access the CAS server.
- Designed to offer easy access to SAS Viya functionality
- Modeled after DATA step and IML
- Available via PROC CAS

The screenshot shows the SAS Viya 3.2 Action Programming documentation interface. The top navigation bar includes the SAS logo, a home icon, the title 'SAS® Viya™ 3.2 Action Programming / Getting Started with CASL', and links for PDF, EPUB, search, and help. The left sidebar contains a table of contents with sections like 'Getting Started with SAS Viya for R', 'Getting Started with CASL' (expanded), 'Requirements', 'Connect and Start a Session', 'About Your Connection and Server', 'Example: Train Gradient Boosted Trees with k-fold Cross Validation', 'CAS Procedure', and 'CAS Procedure' (expanded) with sub-items 'PROC CAS Statement', 'ACTION Statement', and 'ASSIGNMENT Statement'. The main content area is titled 'Getting Started with the CASL Programming Language 3.2' and lists the same topics as the sidebar. At the bottom of the main content area, it states 'Copyright © SAS Institute Inc. All Rights Reserved. Last updated: March 15, 2017'.

CASL Example – Expression Parser

```
print "MRSP[1] (expression ) = "  
    findtable( fetch{table={name="carssashelp"}})[1,"MSRP"];
```

CASL Example – Broken Down

```
proc cas;
  fetch result=f / table={name="carssashelp"};
  table1 = findtable(f);
  print ;
  print "MRSP[1] (raw data) = " table1[1,"MSRP"];
  print "MRSP[1] (sorted) = " sort(table1,"MSRP")[1,"MSRP"];
  print "MRSP[1] (expression ) = "
    findtable( fetch{table={name="carssashelp"}})[1,"MSRP"];
  row = table1[1];
  print;
  print "print the row";
  print;
  print row;
run;
```

```
MRSP[1] (raw data)      = 36945
MRSP[1] (sorted)        = 11939
MRSP[1] (expression )  = 36945

print row

{ _Index_=1, Make=Acura      , Model= MDX                      , Type=SUV      , Origin=Asia
, DriveTrain=All    , MSRP=36945,
Invoice=33337, EngineSize=3.5, Cylinders=6, Horsepower=265, MPG_City=17, MPG_Highway=23, Weight=4451
, Wheelbase=106, Length=189}
```

CASL Example – Broken Down

```
proc cas;
```

```
  fetch result=f / table={name="carssashelp"};  
  table1 = findtable(f);
```

Get the 1st 20 rows of
carsashelp

```
  print ;
```

```
  print "MRSP[1] (raw data) = " table1[1,"MSRP"];
```

```
  print "MRSP[1] (sorted) = " sort(table1,"MSRP")[1,"MSRP"];
```

```
  print "MRSP[1] (expression ) = "
```

```
    findtable( fetch{table={name="carssashelp"}})[1,"MSRP"];
```

```
  row = table1[1];
```

```
  print;
```

```
  print "print the row";
```

```
  print;
```

```
  print row;
```

```
run;
```

```
MRSP[1] (raw data)      = 36945  
MRSP[1] (sorted)       = 11939  
MRSP[1] (expression )  = 36945
```

```
print row
```

```
{_Index_=1,Make=Acura      ,Model= MDX                      ,Type=SUV      ,Origin=Asia  
,DriveTrain=All  ,MSRP=36945,  
Invoice=33337,EngineSize=3.5,Cylinders=6,Horsepower=265,MPG_City=17,MPG_Highway=23,Weight=4451  
,Wheelbase=106,Length=189}
```

CASL Example – Broken Down

```
proc cas;  
  fetch result=f / table={name="carssashelp"};  
  table1 = findtable(f);  
  print ;  
  print "MRSP[1] (raw data) = " table1[1,"MSRP"];  
  print "MRSP[1] (sorted) = " sort(table1,"MSRP")[1,"MSRP"];  
  print "MRSP[1] (expression ) = "  
    findtable( fetch{table={name="carssashelp"}})[1,"MSRP"];  
  row = table1[1];  
  print;  
  print "print the row";  
  print;  
  print row;  
run;
```

Print the value at row
1, column MSRP

```
MRSP[1] (raw data)      = 36945
```

```
MRSP[1] (sorted)       = 11939
```

```
MRSP[1] (expression )  = 36945
```

```
print row
```

```
{_Index_=1,Make=Acura      ,Model= MDX                      ,Type=SUV      ,Origin=Asia  
,DriveTrain=All  ,MSRP=36945,  
Invoice=33337,EngineSize=3.5,Cylinders=6,Horsepower=265,MPG_City=17,MPG_Highway=23,Weight=4451  
,Wheelbase=106,Length=189}
```

CASL Example – Broken Down

```
proc cas;  
  fetch result=f / table={name="carssashelp"};  
  table1 = findtable(f);  
  print ;  
  print "MRSP[1] (raw data) = " table1[1,"MSRP"];  
  print "MRSP[1] (sorted) = " sort(table1,"MSRP")[1,"MSRP"];  
  print "MRSP[1] (expression) = "  
    findtable( fetch{table={name="carssashelp"}})[1,"MSRP"];  
  row = table1[1];  
  print;  
  print "print the row";  
  print;  
  print row;  
run;
```

Nest the action and function calls to get the same result

```
MRSP[1] (raw data)      = 36945  
MRSP[1] (sorted)       = 11939  
MRSP[1] (expression ) = 36945
```

```
print row
```

```
{_Index_=1,Make=Acura      ,Model= MDX                      ,Type=SUV      ,Origin=Asia  
,DriveTrain=All  ,MSRP=36945,  
Invoice=33337,EngineSize=3.5,Cylinders=6,Horsepower=265,MPG_City=17,MPG_Highway=23,Weight=4451  
,Wheelbase=106,Length=189}
```

CASL Example – DO Loop

```
/* list files in caslib path */
table.fileinfo result=fileresult / caslib("&mycaslib" ;
run;
describe(fileresult);
filelist=findtable(fileresult);

/* loop files in caslib path and load them into CAS */
do cvalue over filelist;
  print (cvalue.name);
  /* drop the table if it is in memory */
  table.droptable / caslib("&mycaslib" name=scan(cvalue.name,1,".")
quiet=true;
  table.loadtable / caslib("&mycaslib" path=cvalue.name
casout={name=scan(cvalue.name,1,".")}
        promote=true;
end;
```

CASL Example – DO Loop

```
/* list files in caslib path */  
table.fileinfo result=fileresult / caslib="&mycaslib" ;  
run;  
describe(fileresult);  
filelist=findtable(fileresult);  
  
/* loop files in caslib path and load them into CAS */  
do cvalue over filelist;  
  print (cvalue.name);  
  /* drop the table if it is in memory */  
  table.droptable / caslib="&mycaslib" name=scan(cvalue.name,1,".")  
quiet=true;  
  table.loadtable / caslib="&mycaslib" path=cvalue.name  
casout={name=scan(cvalue.name,1,".")}  
        promote=true;  
end;
```

Get a list of all files in the path

CASL Example – DO Loop

```
/* list files in caslib path */  
table.fileinfo result=fileresult / caslib="&mycaslib" ;  
run;  
describe(fileresult);  
filelist=findtable(fileresult);
```

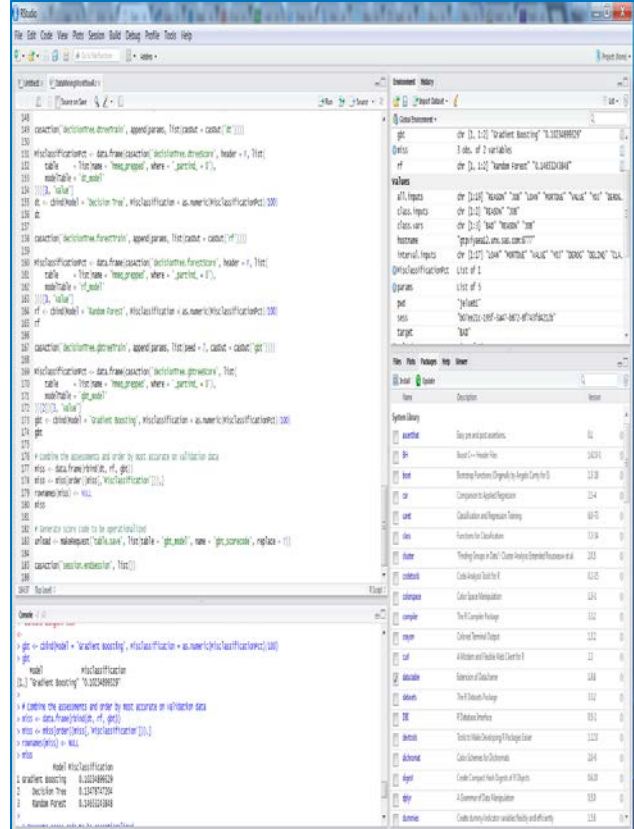
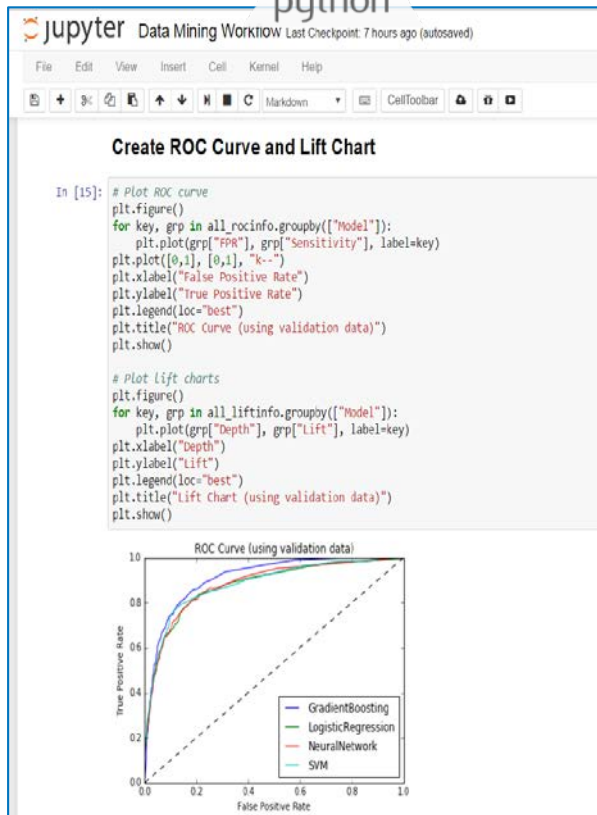
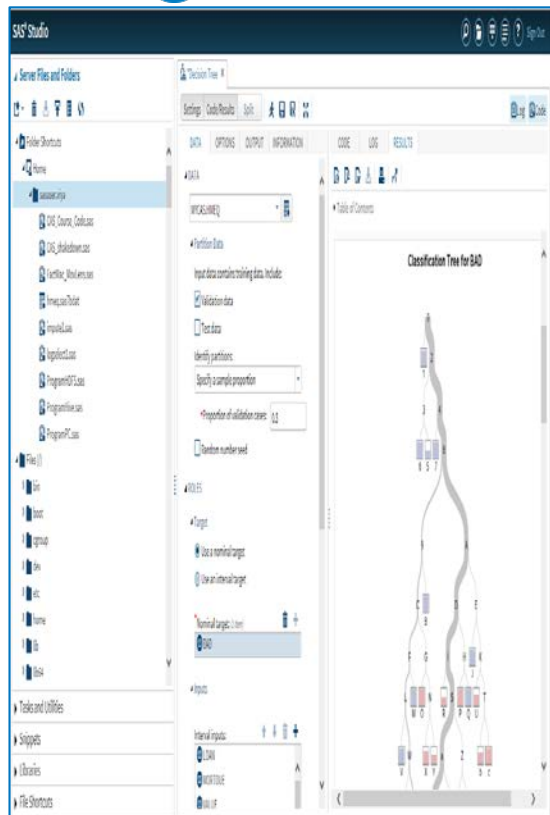
Dynamically load to
CAS all the tables



```
/* loop files in caslib path and load them into CAS */  
do cvalue over filelist;  
  print (cvalue.name);  
  /* drop the table if it is in memory */  
  table.droptable / caslib="&mycaslib" name=scan(cvalue.name,1,".")  
quiet=true;  
  table.loadtable / caslib="&mycaslib" path=cvalue.name  
casout={name=scan(cvalue.name,1,".")}  
        promote=true;  
end;
```



SAS Viya



SAS Scripting Wrapper for Analytics Transfer (SWAT)

- SWAT packages are available for Python, R and Lua free on GitHub or developer.sas.com.
- Download and install SWAT, connect to a CAS server, then write code to drive CAS actions.
- The SWAT package mimics much of the APIs of the native packages making it an easy addition for programmers familiar these languages.



SAS code

```
libname locallib '/opt/sasinside/DemoData';
```

```
libname mycaslib cas caslib=casuser;
```

```
%if not %sysfunc(exist(mycaslib.bank_raw)) %then %do;
```

```
    data mycaslib.bank raw;  
        set locallib.bank_raw;  
    run;
```

```
%end;
```

```
proc cardinality data=mycaslib.bank_raw  
    outcard=mycaslib.varsummarytemp  
    out=mycaslib.leveldetailtemp;  
run;
```

SAS code

Partial Output

Obs	Variable name	Width of the variable formatted value	Type of the raw values	Recommended level for analytics	Have more unreported levels	Number of levels	Number of missing values	Minimum numeric value	Maximum numeric value	Mean	Standard deviation
1	b_tgt	12	N	CLASS	N	2	0	0	1	0.1995296395	0.3996468866
2	cat_input1	5	C	CLASS	N	3	0	.	.	.	.
3	cat_input2	1	C	CLASS	N	5	0	.	.	.	.
4	cnt_tgt	12	N	CLASS	N	7	1	0	6	0.3117560991	0.6396821399
5	demog_age	12	N	INTERVAL	Y	20	266661	-1	89	58.716370999	16.850767422
6	demog_ho	12	N	CLASS	N	2	0	0	1	0.5502604624	0.4974677069
7	demog_pr	12	N	INTERVAL	Y	20	0	0	101	30.589372041	11.52600819
8	int_tgt	10	N	INTERVAL	Y	20	848529	0	500000	11235.867221	8491.8033469
9	r_demog_homeval	12	N	INTERVAL	Y	20	12133	7436	800067	107332.04879	93123.181215
10	r_demog_inc	12	N	INTERVAL	Y	20	253253	2493	200007	53040.586201	18976.611646

Obs	Variable name	Level index	Level frequency	Percentage of the frequency of current level	Percentage of non missing values	Raw numeric value	Raw character value	Formatted value of a variable
1	b_tgt	1	848529	80.04703605	80.04703605	0		0
2	b_tgt	2	211509	19.95296395	19.95296395	1		1
3	cat_input1	1	831371	78.42841483	78.42841483	.	X	X
4	cat_input1	2	77947	7.3437933357	7.3437933357	.	Y	Y
5	cat_input1	3	150820	14.227791834	14.227791834	.	Z	Z
6	cat_input2	1	188398	17.77275909	17.77275909	.	A	A
7	cat_input2	2	192382	18.148594673	18.148594673	.	B	B
8	cat_input2	3	169550	15.994709524	15.994709524	.	C	C
9	cat_input2	4	122282	11.535624195	11.535624195	.	D	D
10	cat_input2	5	387425	36.548312419	36.548312419	.	E	E
11	cnt_tgt	1	848529	80.04703605	80.047111553	0		0

An Example Flow for Python SWAT Interface to CAS

Import Packages

CAS Session and CAS Data
Load

Import CAS Action Sets

Use CAS Action Sets for
Printing and Cardinality

Use Python to Plot
Resulting Tables

SWAT Python

```
In [1]: from swat import *  
  
import pandas as pd  
  
import numpy as np  
  
from sklearn import linear_model
```

```
In [23]: sess = CAS(cashost, casport, authinfo="~/ .authinfo", caslib="casuser")
```

SWAT Python

```
sess.actionsetinfo(all=True)
```

Partial Output

	actionset	label
0	access	
1	accessControl	Access Controls
2	aggregation	
3	astore	
4	autotune	
5	boolRule	
6	builtins	System
7	cardinality	
8	clustering	
9	configuration	Server Properties
10	dataPreprocess	Data Preprocess
11	dataStep	DATA Step
12	decisionTree	

```
sess.loadactionset(actionset="cardinality")
```

NOTE: Added action set 'cardinality'.

SWAT Python

```
indata_dir="/opt/sasinside/DemoData"  
indata="bank_raw"
```

```
if not sess.table.tableExists(table=indata).exists:  
    tbl = sess.upload_file(indata_dir+"/"+indata+".sas7bdat", casout={"name":indata})
```

NOTE: Cloud Analytic Services made the uploaded file available as table BANK_RAW in caslib CASUSER(viyauser).
NOTE: The table BANK_RAW has been created in caslib CASUSER(viyauser) from binary data uploaded to Cloud Analytic Services.

```
tbl.head(10)
```

Partial Output:

	b_tgt	cat_input1	cat_input2	cnt_tgt	demog_age	demog_ho	demog_pr	int_tgt	r_demog_homeval	r_demog_inc
0	1.0	X	A	NaN	NaN	0.0	24.0	7000.0	57600.0	52106.0
1	1.0	X	A	2.0	NaN	0.0	24.0	7000.0	57587.0	52106.0
2	1.0	X	A	2.0	NaN	0.0	0.0	15000.0	44167.0	42422.0

SWAT Python

```
tbl_data_card=sess.CASTable('data_card',replace=True)
tbl.cardinality.summarize(cardinality=tbl_data_card)

tbl_data_card.vars=['_VARNAME_', '_TYPE_', '_NMISS_', '_NOBS_', '_MEAN_', '_MIN_', '_MAX_', '_STDDEV_']
df_data_card=tbl_data_card

df_data_card.head(25)
```

NOTE: Writing cardinality.

NOTE: status = 0.

NOTE: The Cloud Analytic Services server processed the request in 0.001606426 seconds.

Partial Output

	VARNAME	_TYPE_	_NMISS_	_NOBS_	_MEAN_	_MIN_	_MAX_	_STDDEV_
0	b_tgt	N	0.0	1060038.0	0.199530	0.00	1.00	0.399647
1	cat_input1	C	0.0	1060038.0	NaN	NaN	NaN	NaN
2	cat_input2	C	0.0	1060038.0	NaN	NaN	NaN	NaN
3	cnt_tgt	N	1.0	1060038.0	0.311756	0.00	6.00	0.699682
4	demog_age	N	266861.0	1060038.0	58.716371	-1.00	89.00	16.850767

SWAT Python

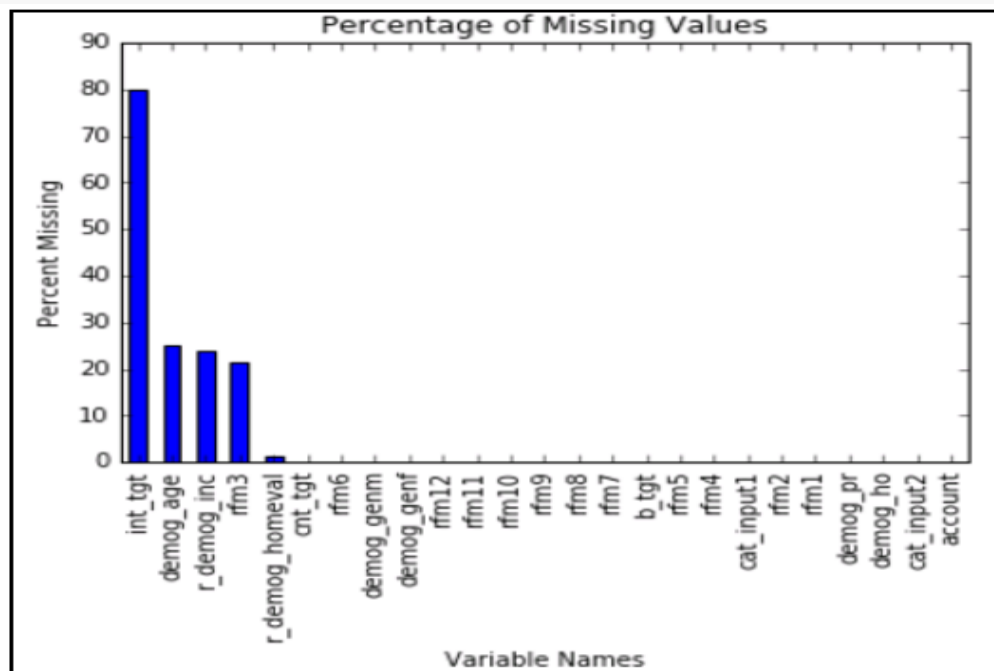
```
df_data_card=tbl_data_card.to_frame()
df_data_card['PERCENT_MISSING']=(df_data_card['_NMISS_']/df_data_card['_NOBS_'])*100

df_data_card=df_data_card.sort_values(by=['PERCENT_MISSING'], ascending=False)

tbl_forplot=pd.Series(list(df_data_card['PERCENT_MISSING']), index=list(df_data_card['_VARNAME_']))

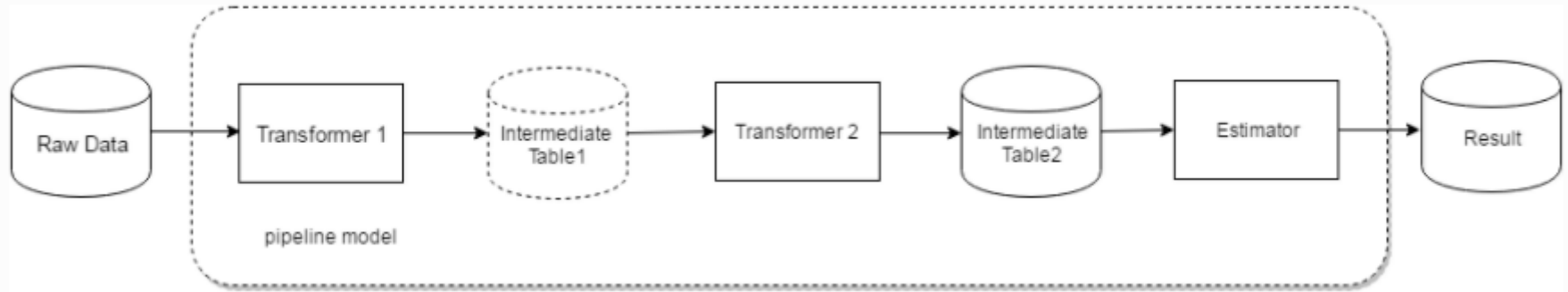
ax=tbl_forplot.plot(
    kind='bar',
    title='Percentage of Missing Values'
)

ax.set_ylabel('Percent Missing')
ax.set_xlabel('Variable Names')
```



Pipefitter

- Python API for developing pipelines for data transformation and model fitting
- Works with SAS 9.4 or SAS Viya
- Uses Python SWAT and SASpy under the covers



Pipefitter Available on GitHub



Features Business Explore Marketplace Pricing

This repository

Search

[Sign in](#) or [Sign up](#)

sassoftware / python-pipefitter

Watch

10

Star

6

Fork

2

Code

Issues 0

Pull requests 0

Projects 0

Insights ▾

The SAS pipefitter package provides a Python API for developing pipelines for data transformation and model fitting as stages of a repeatable machine learning workflow in either SAS v9 or SAS Viya.

17 commits

2 branches

1 release

2 contributors

Apache-2.0

Branch: master ▾

[New pull request](#)

[Find file](#)

[Clone or download ▾](#)



kesmit13 Remove xrange for Python 3 compatibility

Latest commit f9f1a9f 19 days ago

doc

Update line endings

2 months ago

examples

remove redundant lines

25 days ago

pipefitter

Remove xrange for Python 3 compatibility

19 days ago

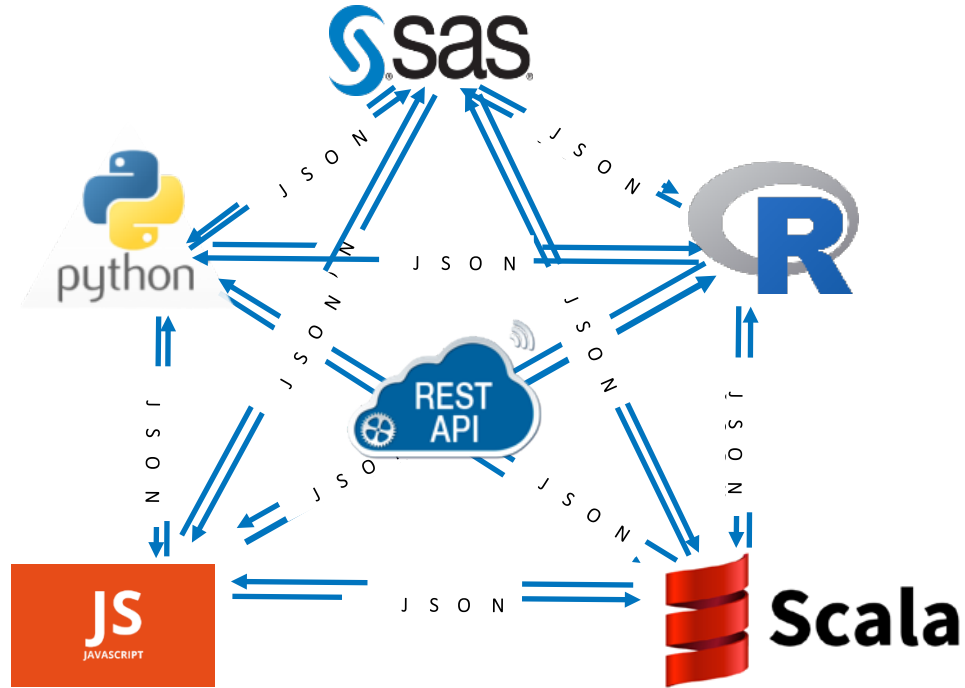
.gitattributes

Add git config files

2 months ago



REST APIs

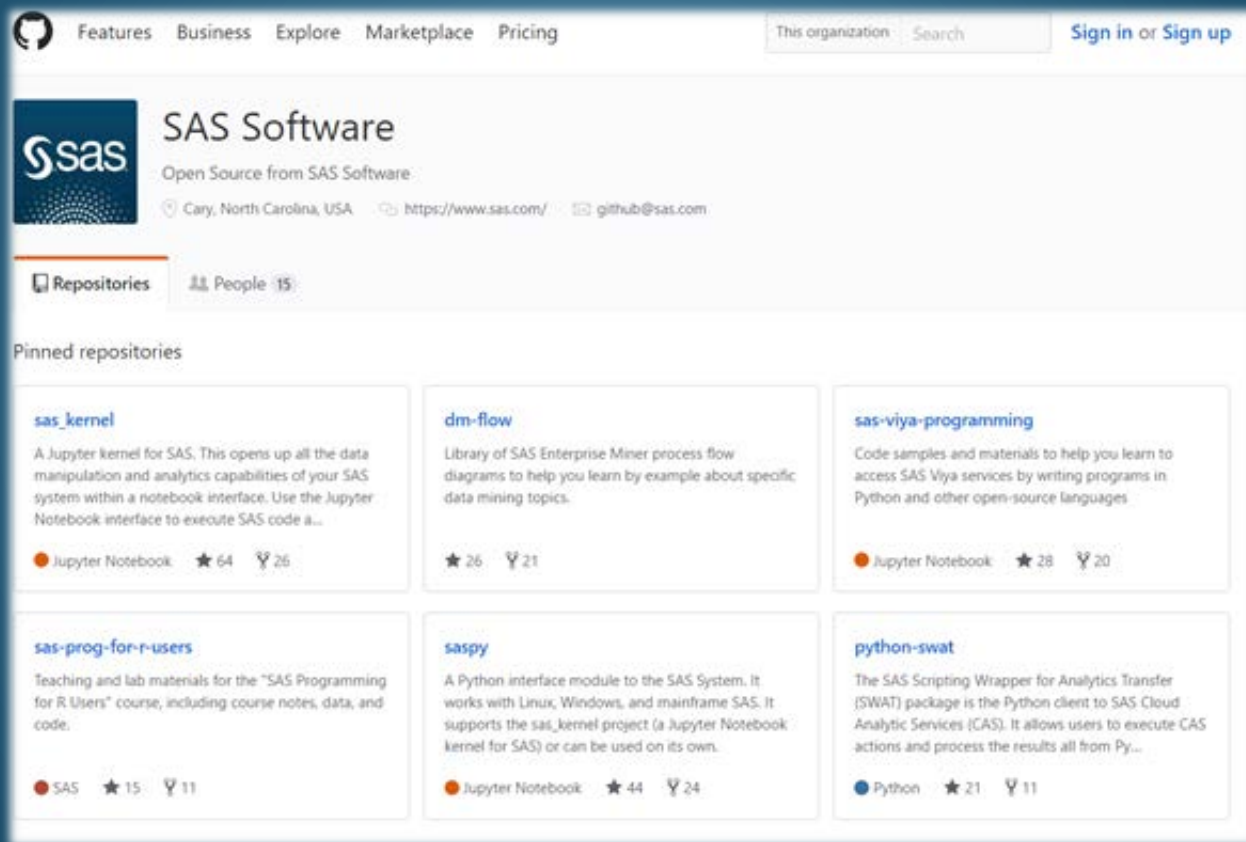


Communities



The image shows two overlapping browser windows. The top window is the SAS Software GitHub page, displaying navigation links like 'Personal', 'Open source', 'Business', 'Explore', 'Pricing', 'Blog', and 'Support', along with a search bar and 'Sign in'/'Sign up' buttons. The bottom window is the SAS Support Communities website. It features the SAS logo, navigation links for 'Products & Solutions', 'Industries', 'Support', 'Learn SAS', 'Partners', 'Community', and 'About SAS'. A prominent banner reads 'Welcome to SAS Support Communities' with the text 'Stuck on a problem? Ask the community for help.' and 'While you're here, get a SAS tip and share what you know. This community of SAS experts is here to help you to succeed!'. A 'Join Now' button is visible. Below the banner is a search bar and a 'Find a Community' button. The 'Latest Activity' section shows a post titled 'MDX for authorization in a cube' with 0 New, 6 Replies, and 0 Likes. A sidebar on the left lists 'Pinned repositories' including 'sas_kernel' and 'sas-prog-for-r-users'. A statistics box on the right shows 115,545 MEMBERS, 1,576 ONLINE, and 309,408 POSTS. A 'Post a Question' button is at the bottom right. The SAS logo is in the bottom right corner of the image.

<https://github.com/sassoftware>



The screenshot shows the GitHub organization page for SAS Software. The header includes navigation links: Features, Business, Explore, Marketplace, Pricing, and a search bar. The organization's name, logo, and location (Cary, North Carolina, USA) are displayed. Below this, there are tabs for Repositories and People. The 'Pinned repositories' section lists six repositories with their descriptions, languages, and statistics (stars and forks).

Repository Name	Description	Language	Stars	Forks
sas_kernel	A Jupyter kernel for SAS. This opens up all the data manipulation and analytics capabilities of your SAS system within a notebook interface. Use the Jupyter Notebook interface to execute SAS code a...	Jupyter Notebook	64	26
dm-flow	Library of SAS Enterprise Miner process flow diagrams to help you learn by example about specific data mining topics.		26	21
sas-viya-programming	Code samples and materials to help you learn to access SAS Viya services by writing programs in Python and other open-source languages	Jupyter Notebook	28	20
sas-prog-for-r-users	Teaching and lab materials for the "SAS Programming for R Users" course; including course notes, data, and code.	SAS	15	11
saspy	A Python interface module to the SAS System. It works with Linux, Windows, and mainframe SAS. It supports the sas_kernel project (a Jupyter Notebook kernel for SAS) or can be used on its own.	Jupyter Notebook	44	24
python-swath	The SAS Scripting Wrapper for Analytics Transfer (SWAT) package is the Python client to SAS Cloud Analytic Services (CAS). It allows users to execute CAS actions and process the results all from Py...	Python	21	11

developer.sas.com

 THE POWER TO KNOW.

[Home](#) [SAS](#) [REST](#) [Python](#) [Java](#) [Lua](#) [R](#)

SAS[®] for Developers [Give us your feedback](#)


Use the power of SAS[®] Viya[™] in your applications.

SAS Viya is an open analytics platform providing developers access to SAS services.
One underlying code base, programmable in the language of your choice.


SAS Viya uses PROC CAS to run CAS actions in SAS Cloud Analytic Services.


REST APIs for any client language to access SAS analytics, data and services.


Python APIs for using SAS Viya CAS actions.


Java APIs for using SAS Viya CAS actions.


Lua APIs for using SAS Viya CAS actions.


R APIs for using SAS Viya CAS actions.