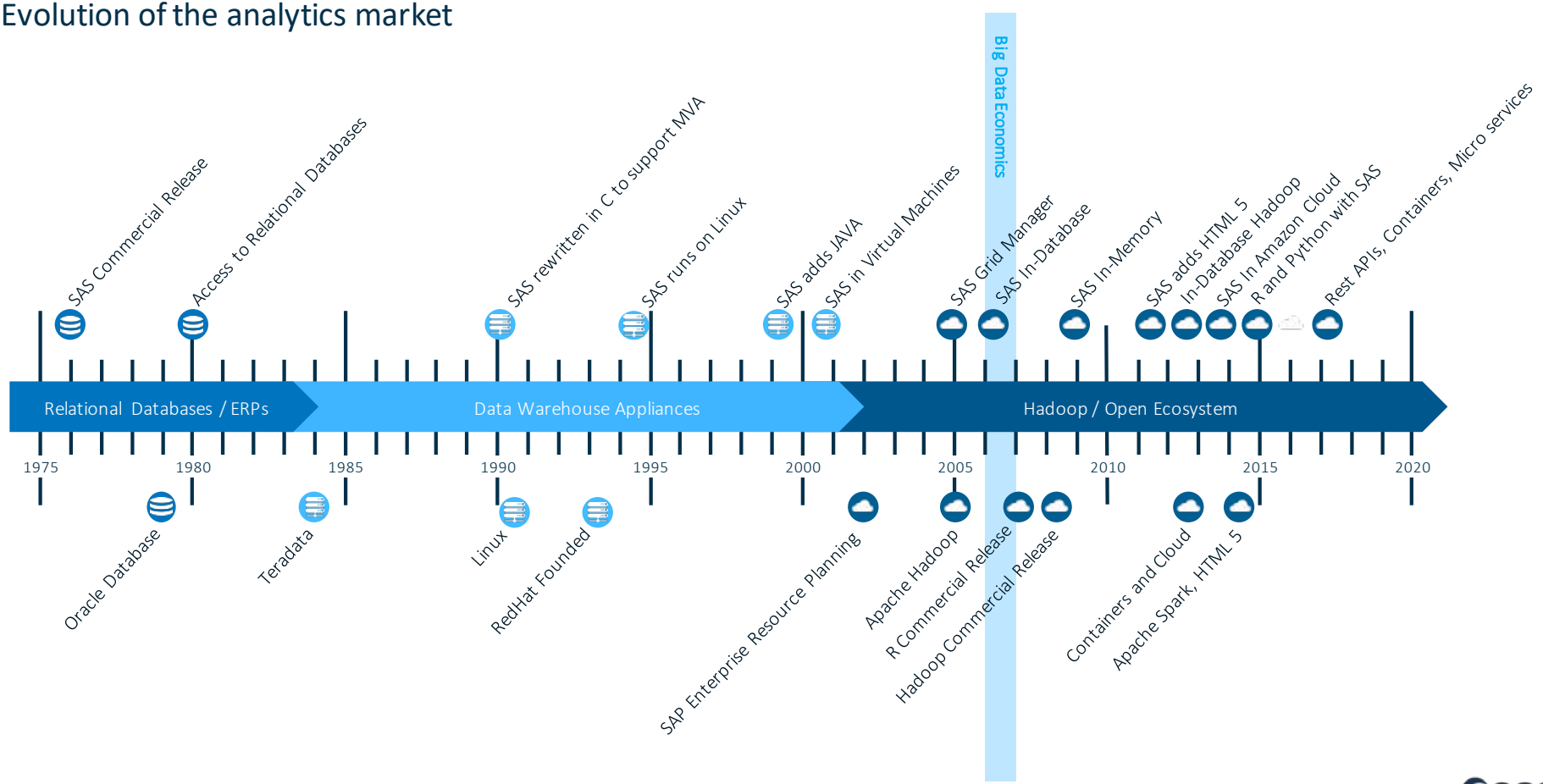
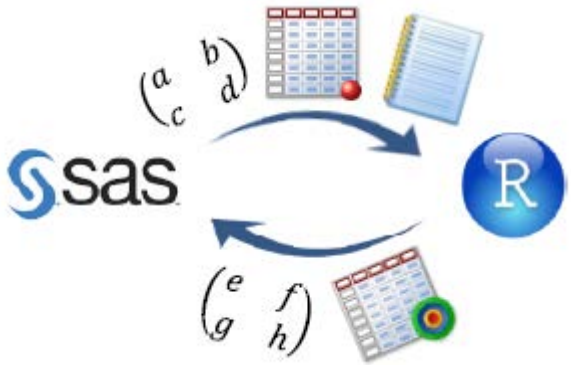
The New Languages of SAS 9.4 and SAS Viya:
A SAS Programmer's Primer

SAS OPEN TIMELINE
Evolution of the analytics market



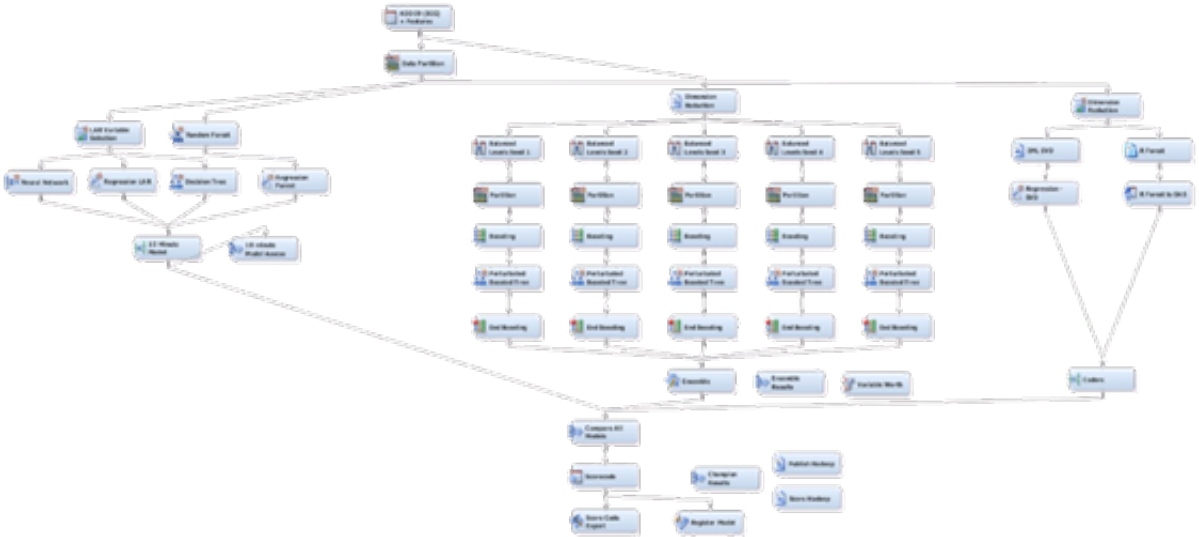
Using SAS/IML to Interact with R

- 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

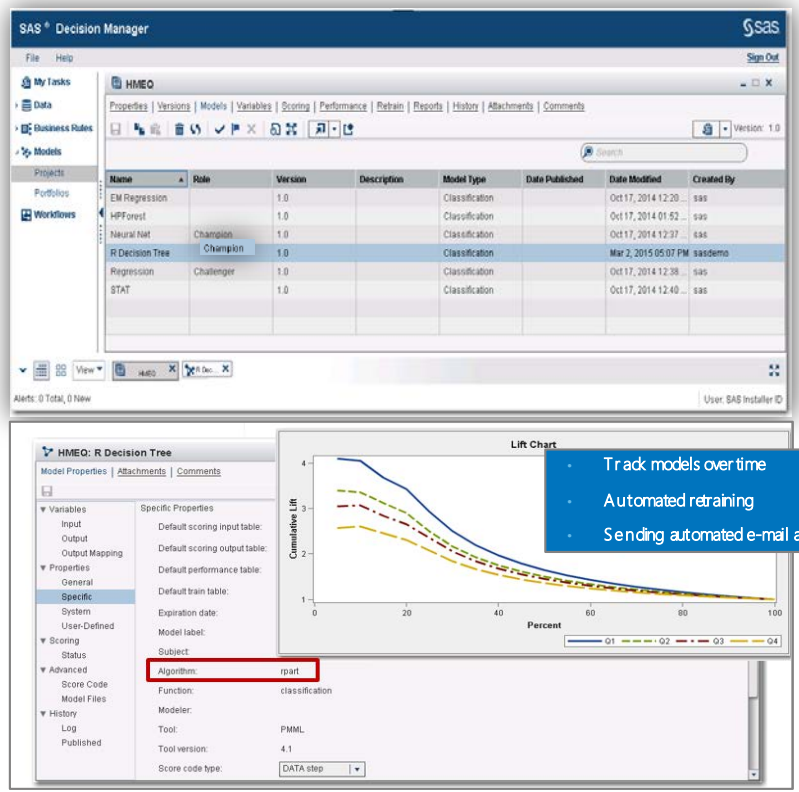
- Leverage SAS advanced analytics in a easy-to-use GUI environment
- Leverage R & Python modeling packages from a SAS environment
- Compare & Ensemble R, Python, and SAS models in one interface



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

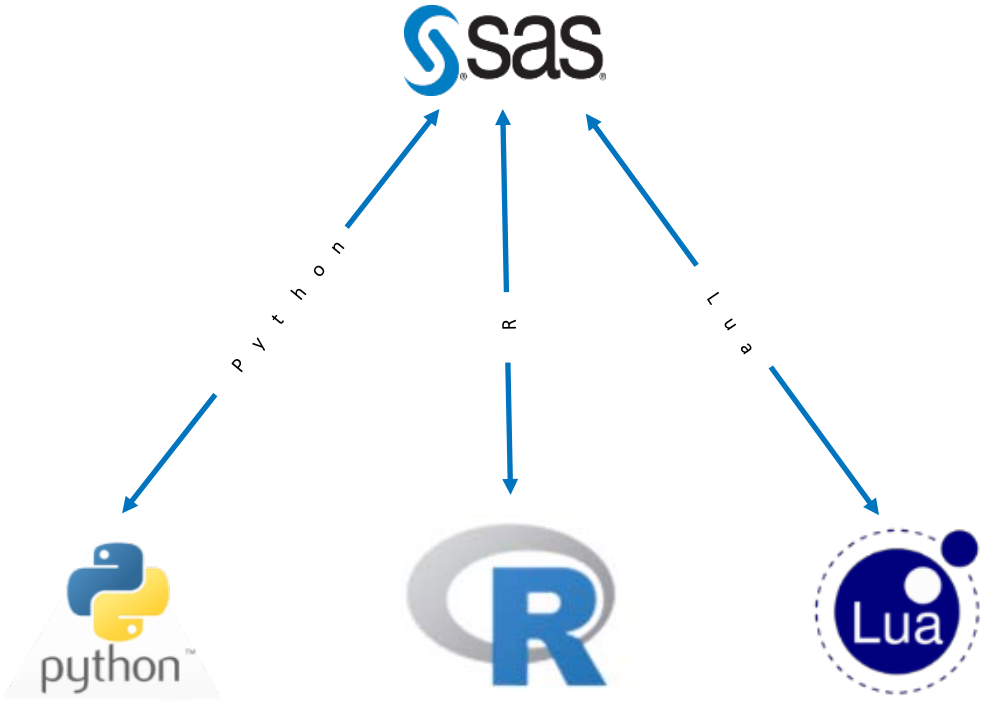


PROC LUA

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

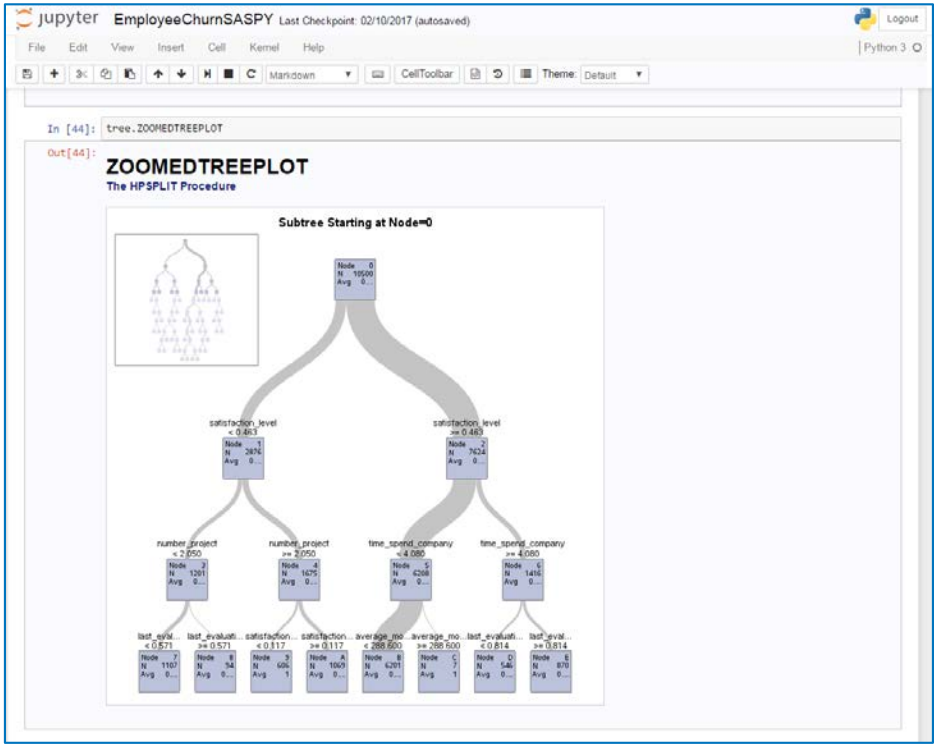
```
proc lua;
   submit;
      local fish=sas.read_ds("sashelp.fish")
      print("fish=", table.tostring(fish))
   endsubmit;
run;
```

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 dta 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

jupyter Untitled26 (unsaved changes)

File Edit View Insert Cell Kernel Help | SAS

Code CellToolbar

Linear Regression

Can we predict the horsepower of a car based on several commonly available attributes

In [1]:
```
proc reg data=sashelp.cars plots=diagnosticpanel;
    model horsepower = length weight msrp mpg_city;
run;
```

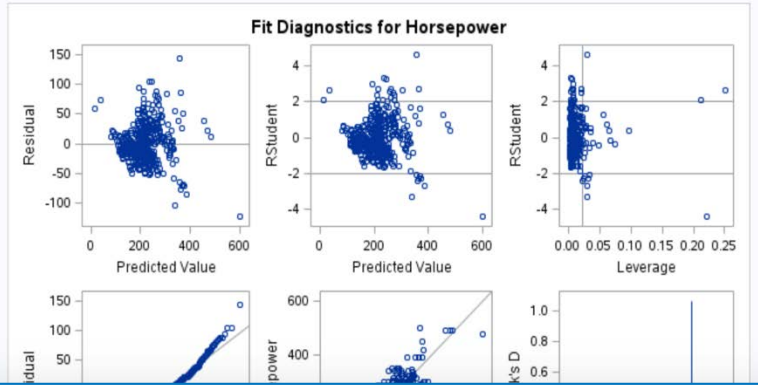
Out[1]:

The SAS System

The REG Procedure

Model: MODEL1

Dependent Variable: Horsepower



jupyter july31_python Last Checkpoint: 07/31/2015 (autosaved)

File Edit View Insert Cell Kernel Help | Python 3

Code Cell Toolbar: None

In [7]:
```
cm = stat.reg(model='horsepower = length weight msrp mpg_city', data=cars)
```

In [8]:
```
cm.DIAGNOSTICSPANEL
```

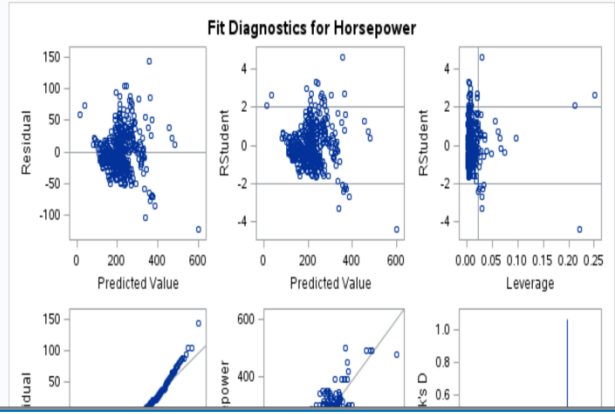
Out[8]:

The SAS System

The REG Procedure

Model: MODEL1

Dependent Variable: Horsepower



Viewing the SAS Code Behind SASPy

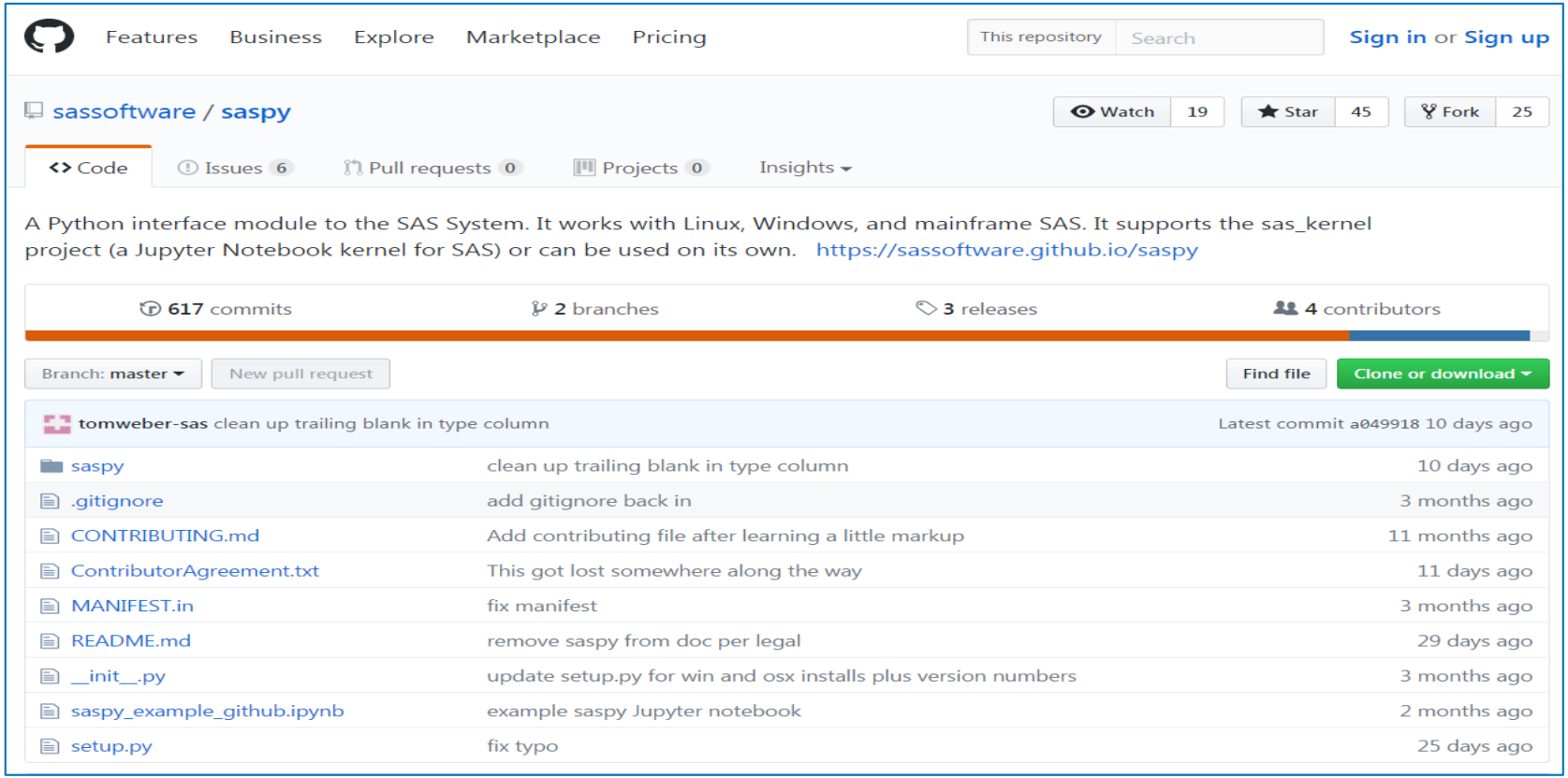
In [12]:

```
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



SAS Viya

MULTIPLE INTERFACES, SINGLE CODE BASE

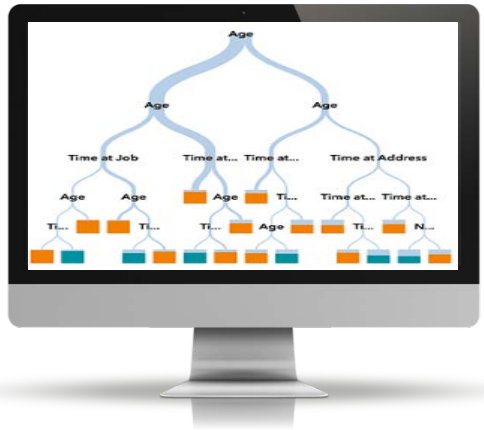
Visual Interfaces

Programming Interfaces

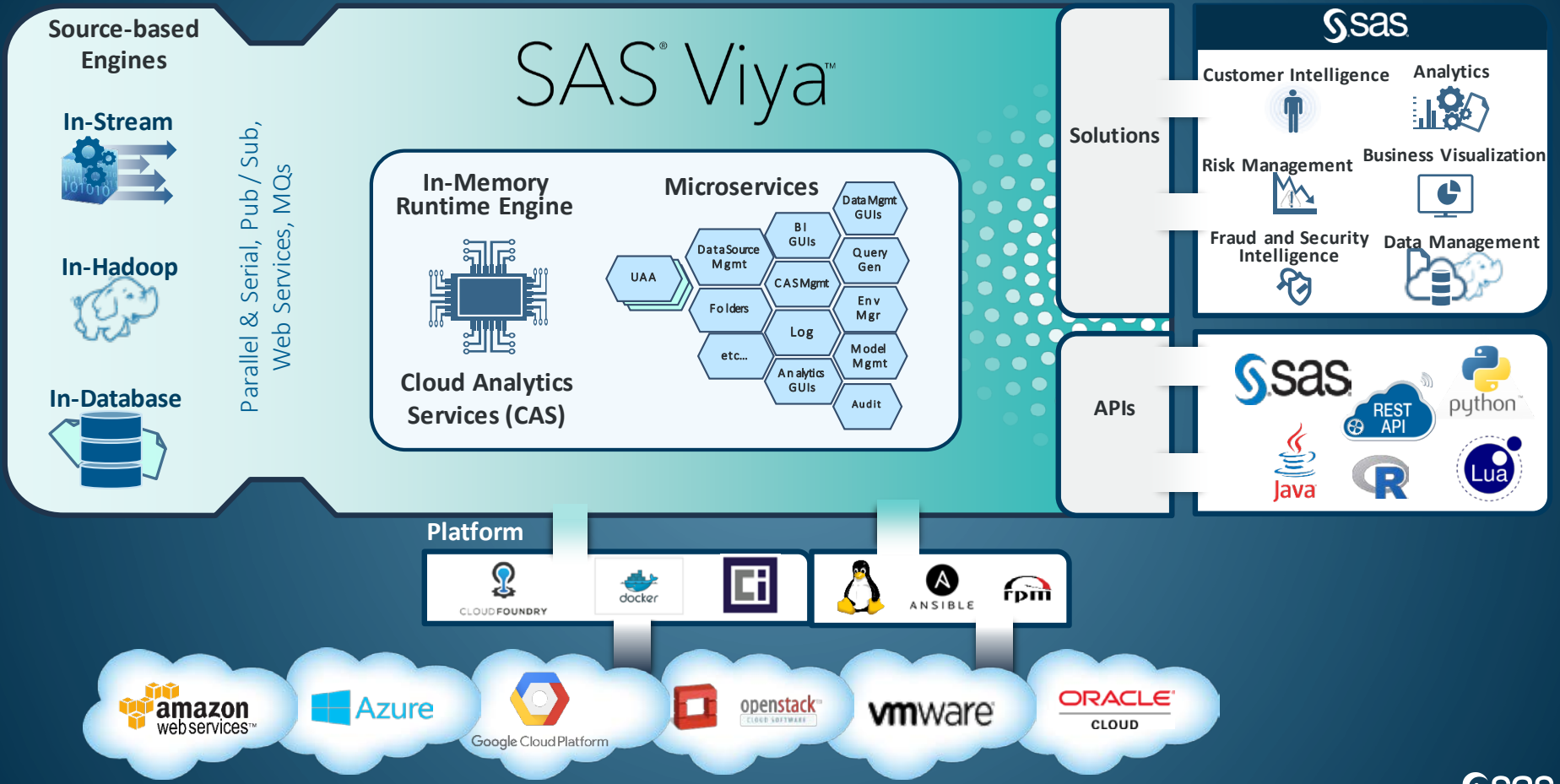
python sas Java R

API Interfaces

REST API

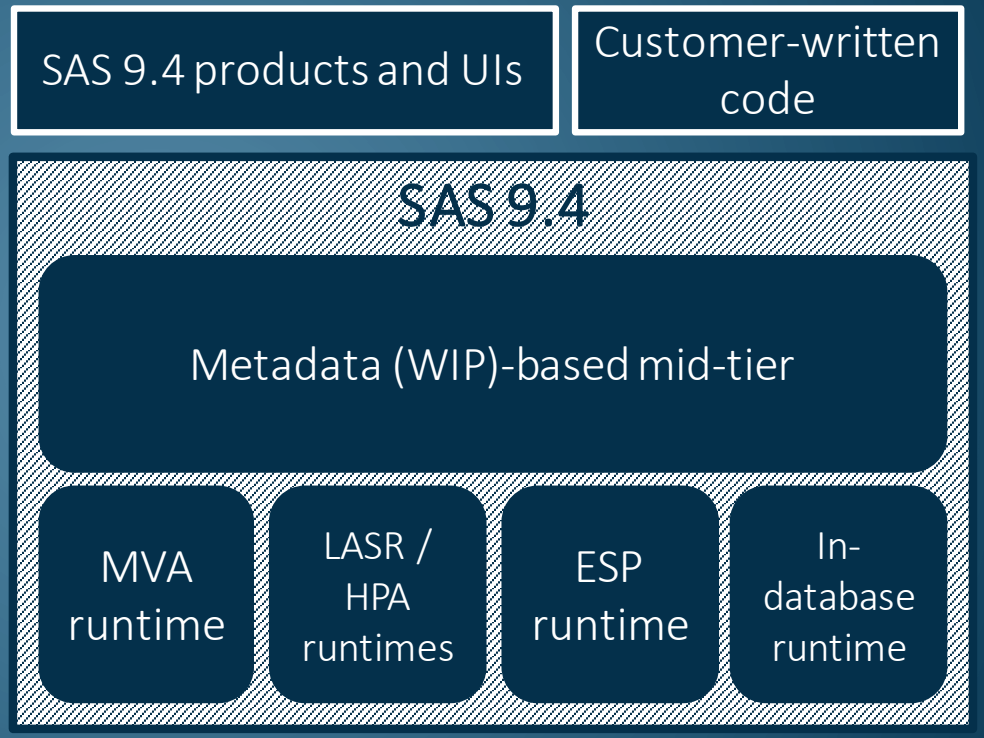


sas

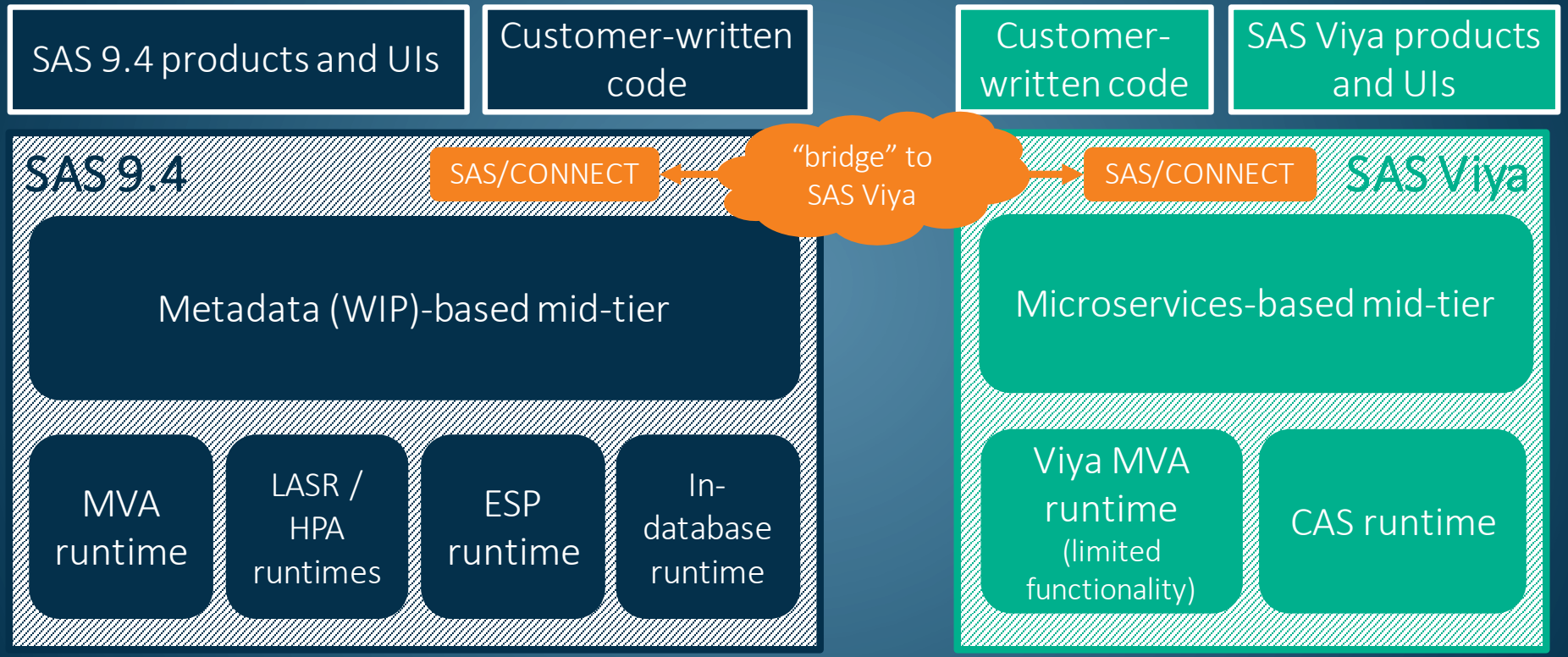


The SAS 9.4 platform

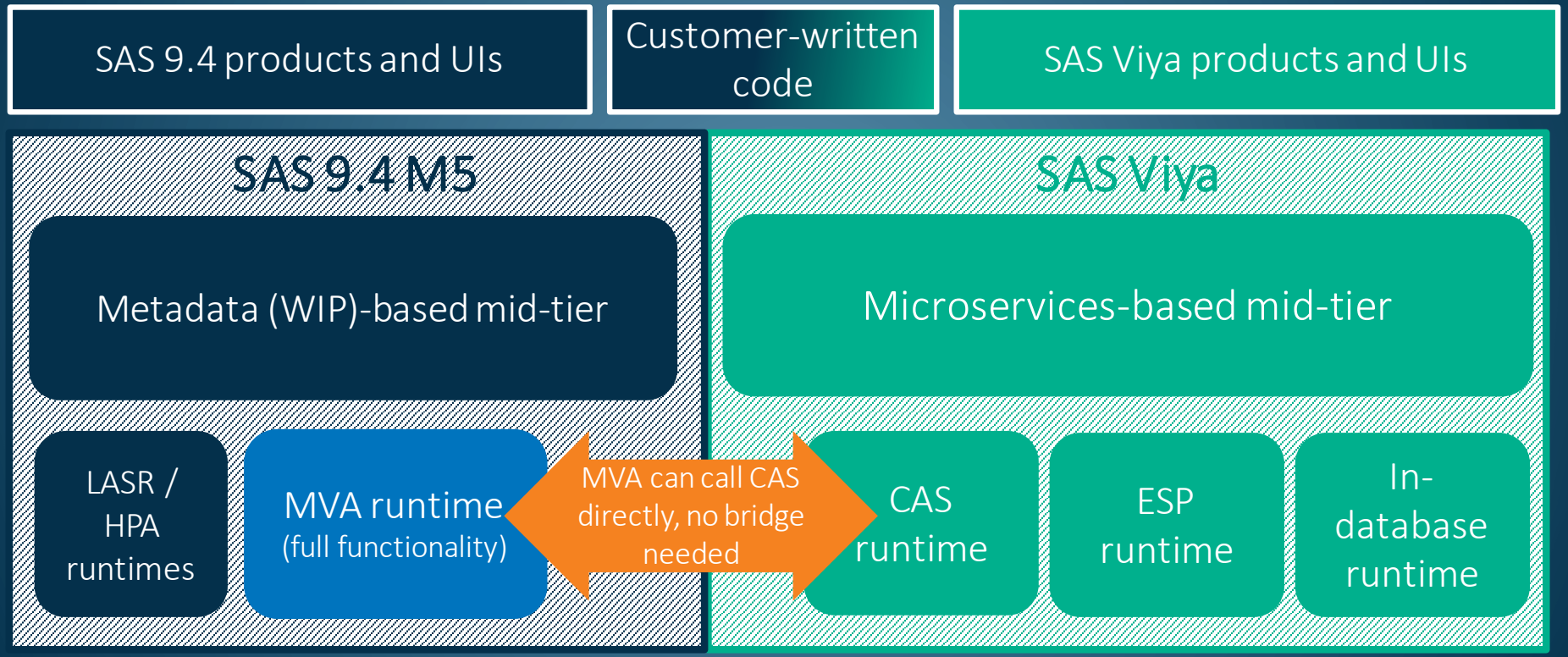
- Support for any analytics use case
- Broad and extensive customer base in production
- Modernization needed to adapt to changing technology, business trends



Prior to 9.4M5 architecture

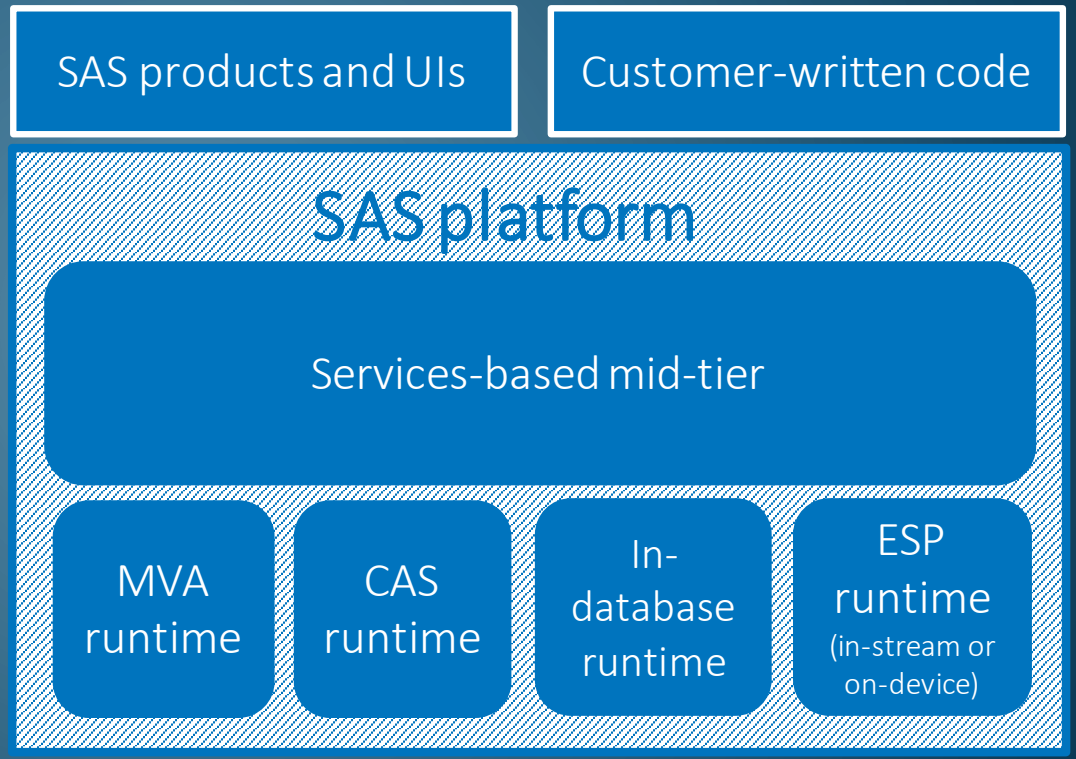


Today's architecture



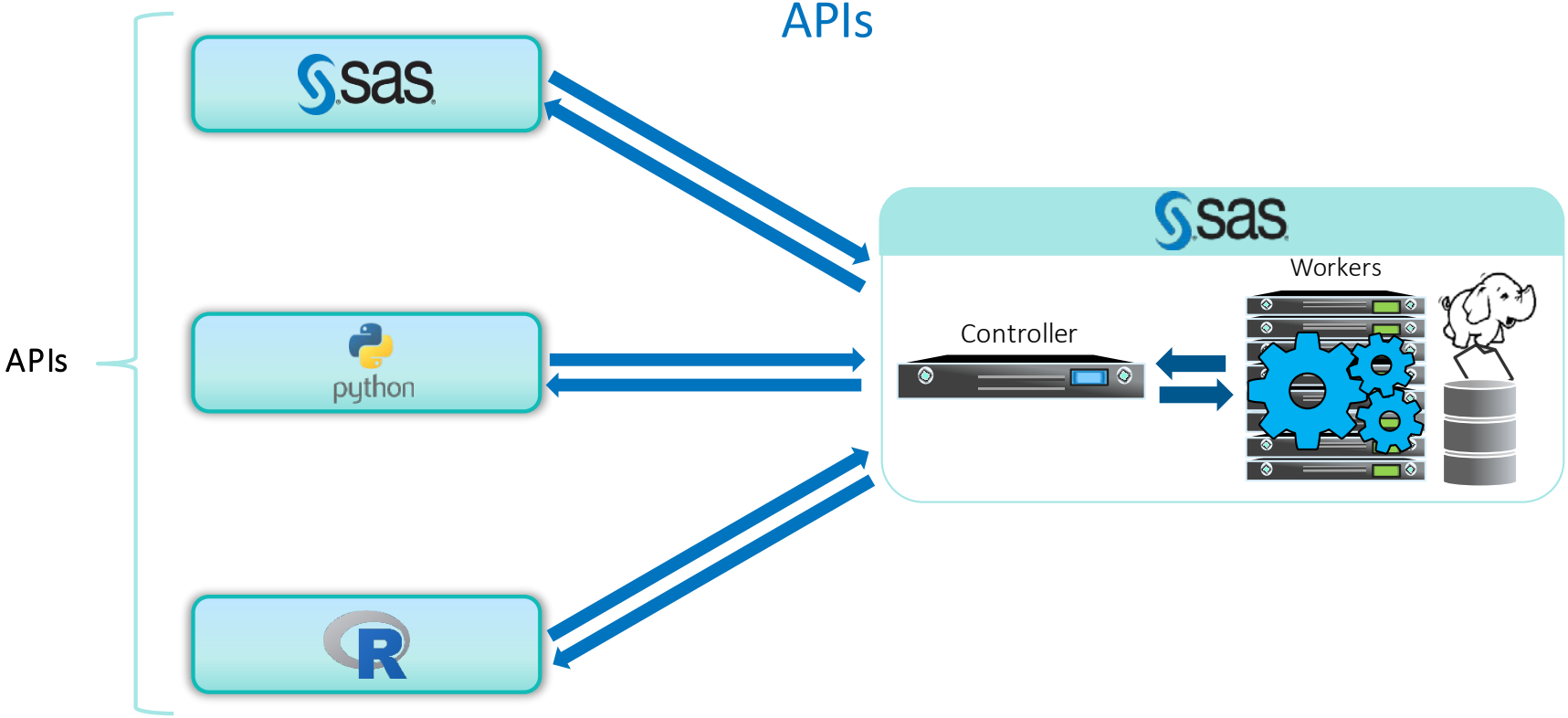
The SAS Platform – Projected Architecture

- 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

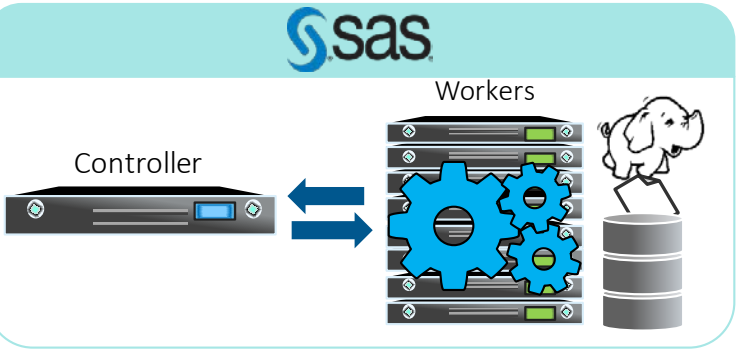
An Example

APIs

```
proc print data = hmeq (obs = 10);
run;
```

```
df = s.CASTable('hmeq')
df.head(10)
```

```
df <- defCasTable(s, 'hmeq')
head(df, 10)
```



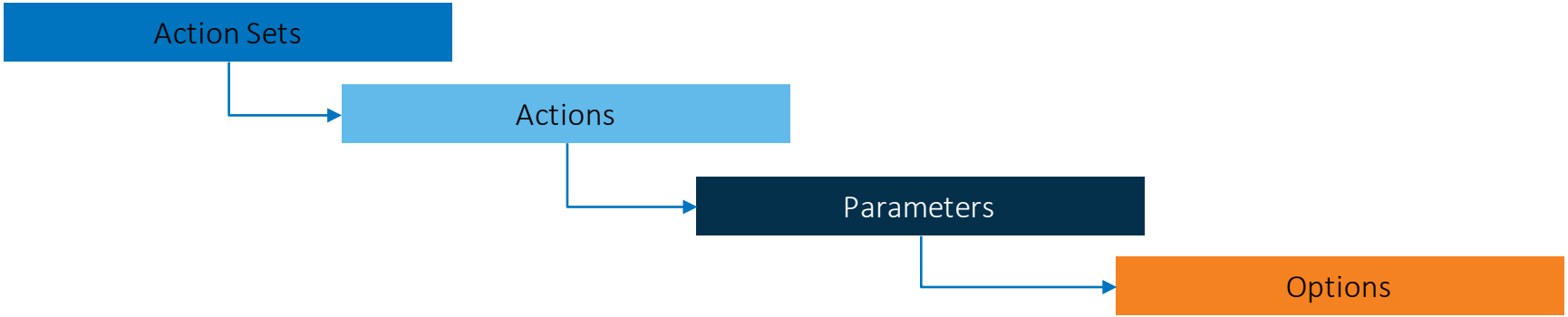
Translated Command

```
[table.fetch]
    table.name = "hmeq"
    from = 1 to = 10
```

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
    }, {...}}
```

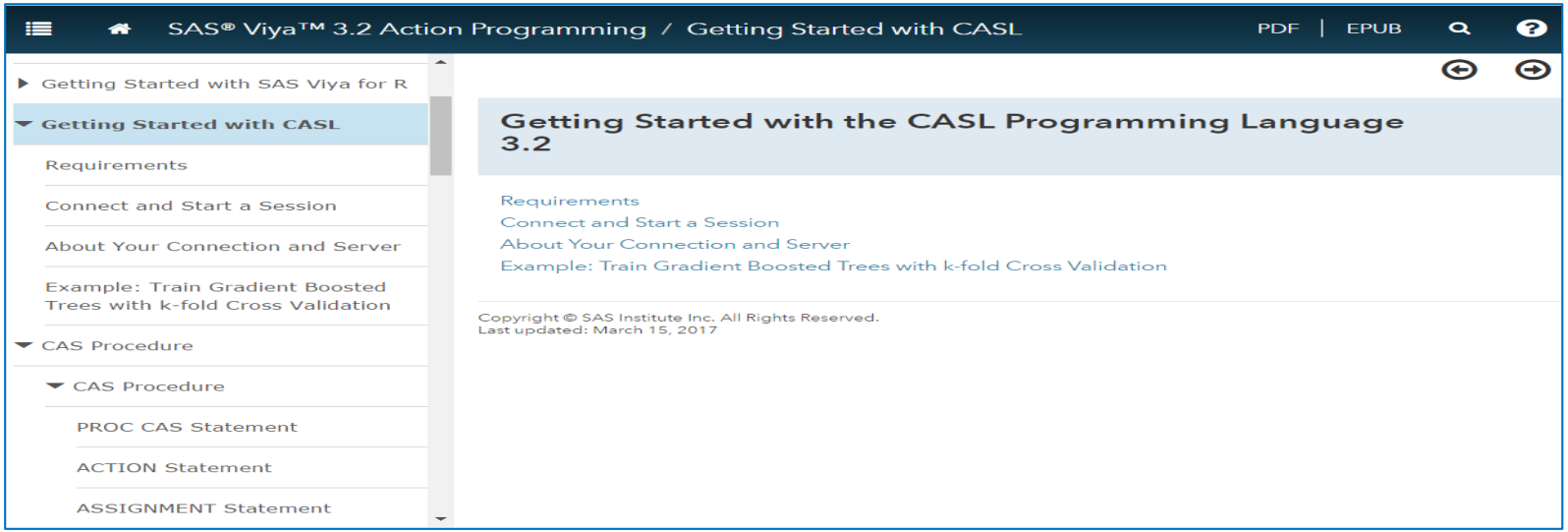
PROC versus CAS Action

```
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



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);
   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;
```

Get the 1st 20 rows of carsashelp

```
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;
```

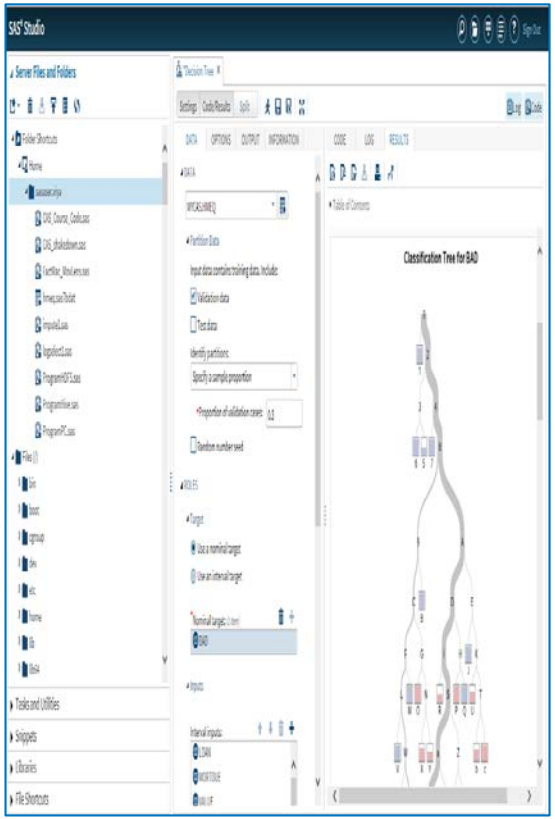
Get a list of all files in the path

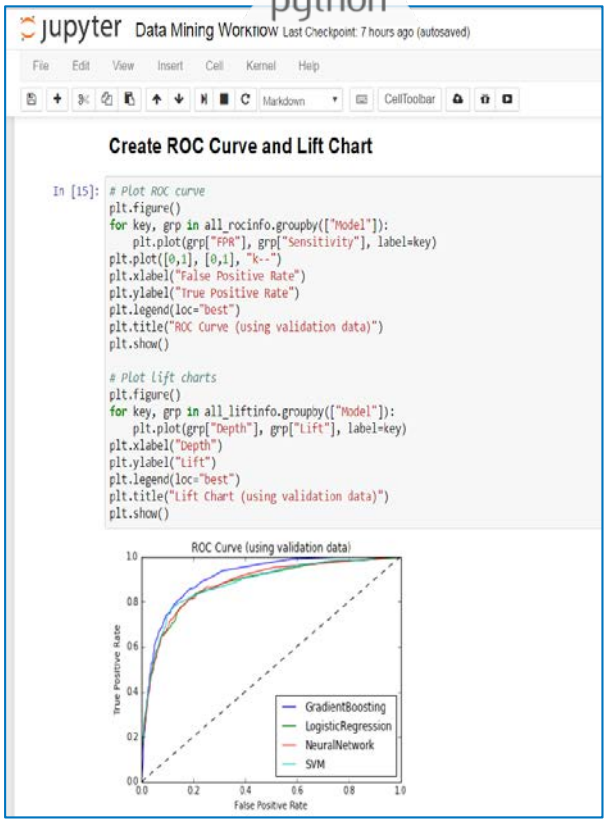
CASL Example – DO Loop

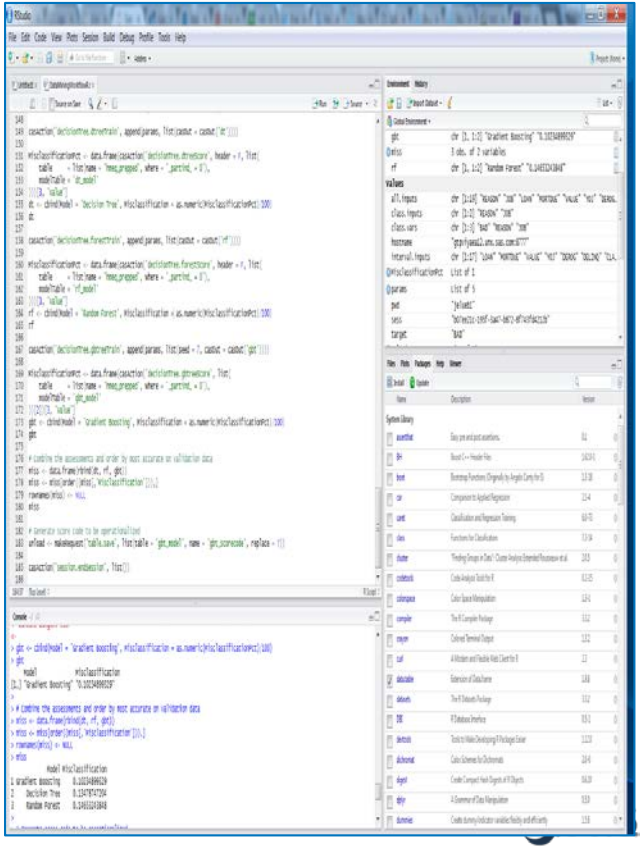
```
/* list files in caslib path */
table.fileinfo result=fileresult / caslib="&mycaslib" ;
run;
describe(fileresult);
filelist=findtable(fileresult);

/* 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;
```

Dynamically load to CAS all the tables

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;
```

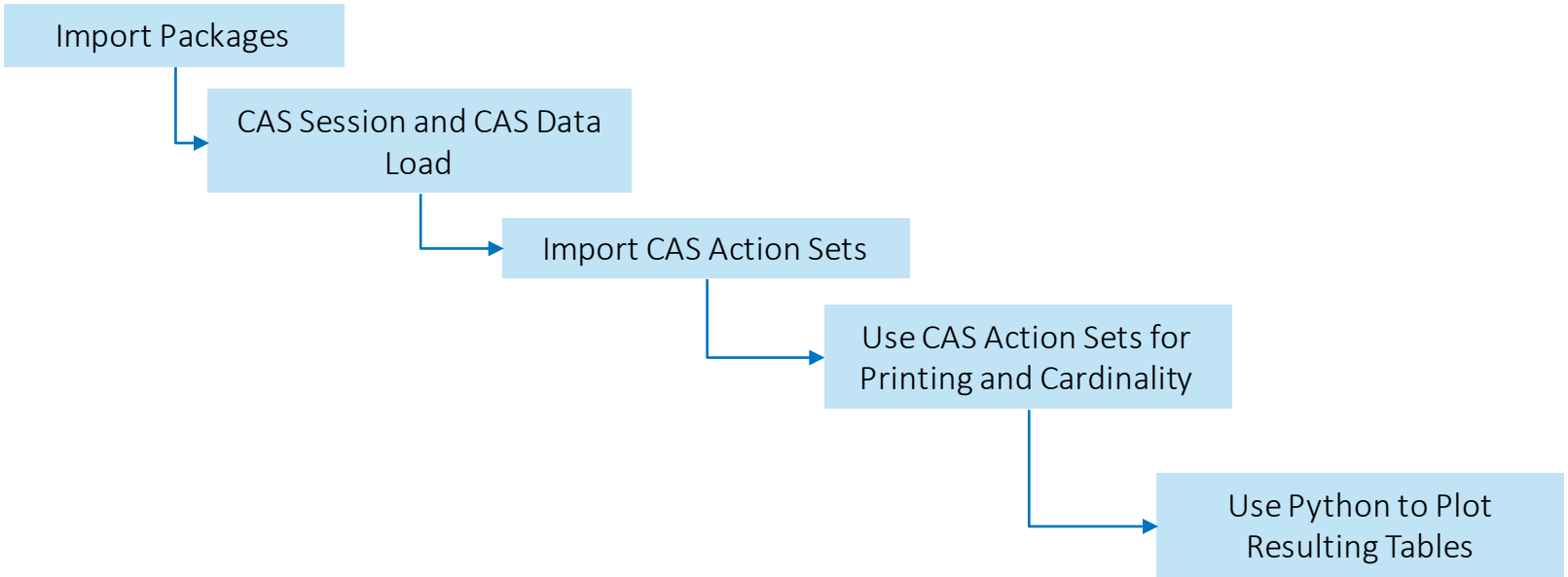
SAS code

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	0.3996469866
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	0.3117560891	0.6996821399
5	demog_age	12	N	INTERVAL	Y	20	266861	-1	89	58.716370999	16.950767422
6	demog_ho	12	N	CLASS	N	2	0	0	1	0.5502604624	0.4974677089
7	demog_pr	12	N	INTERVAL	Y	20	0	0	101	30.569372041	11.52600819
8	int_tgt	10	N	INTERVAL	Y	20	848529	0	500000	11235.857221	8491.8033469
9	r_demog_homeval	12	N	INTERVAL	Y	20	12133	7436	600067	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	80.04703805	80.04703805	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	77847	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	80.04703805	80.047111563	0		0

An Example Flow for Python SWAT Interface to CAS



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 Analyti
c 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)
```

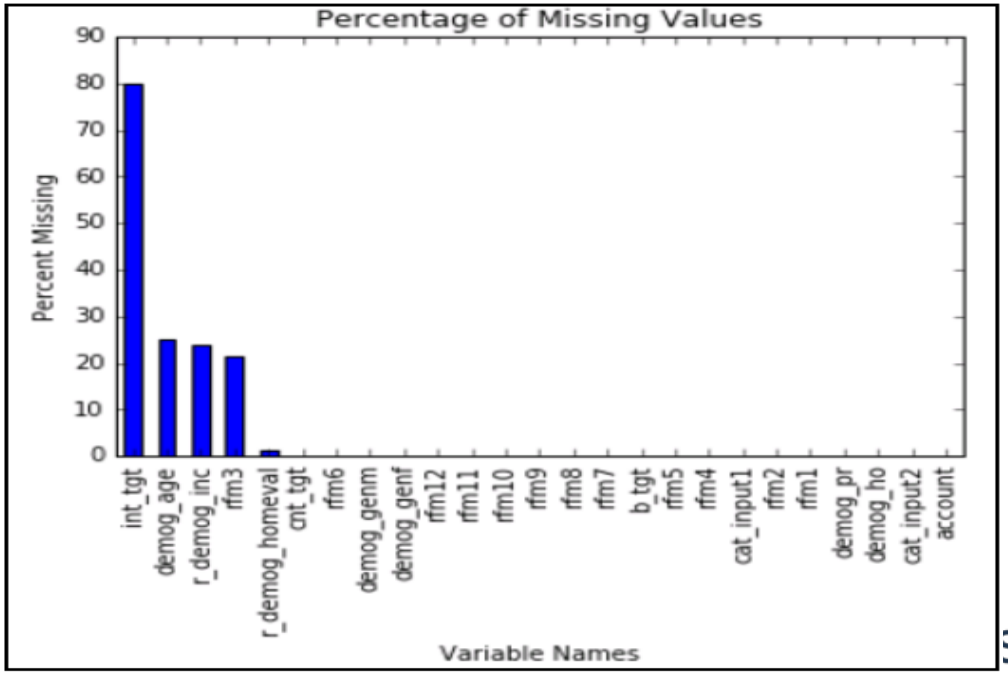
```
NOTE: Writing cardinality.
NOTE: status = 0.
NOTE: The Cloud Analytic Services server processed the request in 0.001606426 seconds.
```

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

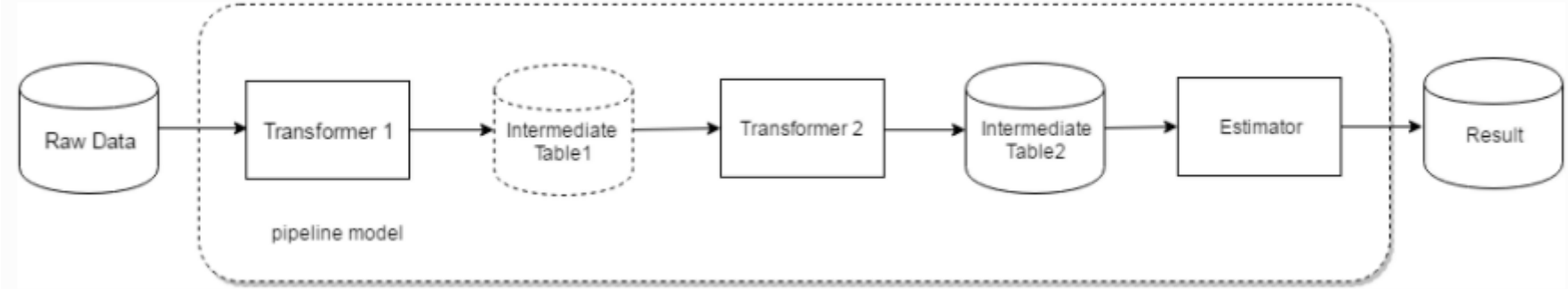
SWAT Python

```
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.

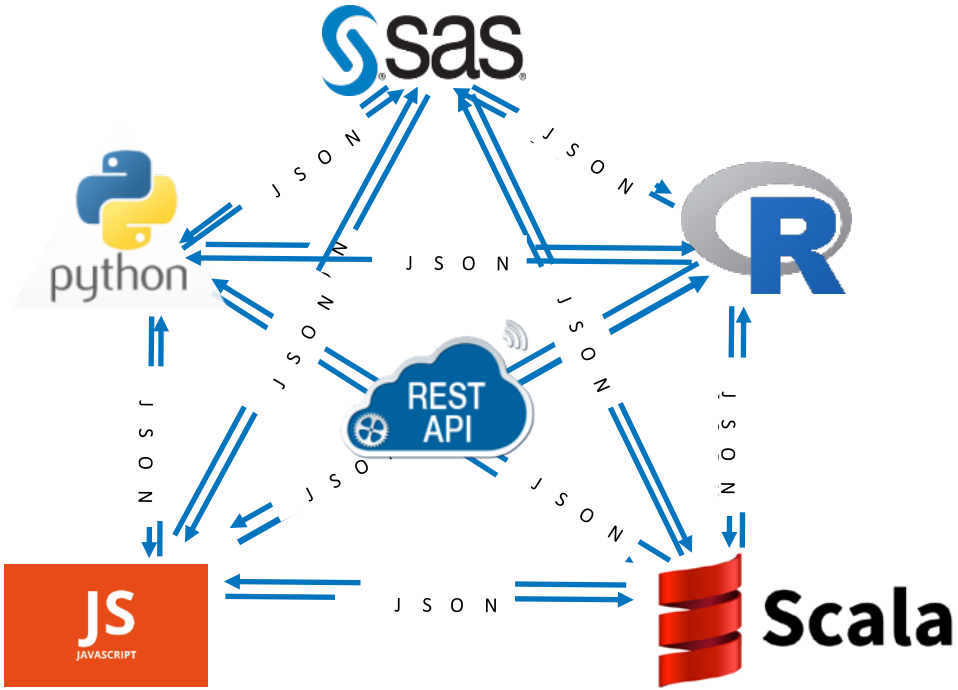
17 commits 2 branches 1 release 2 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

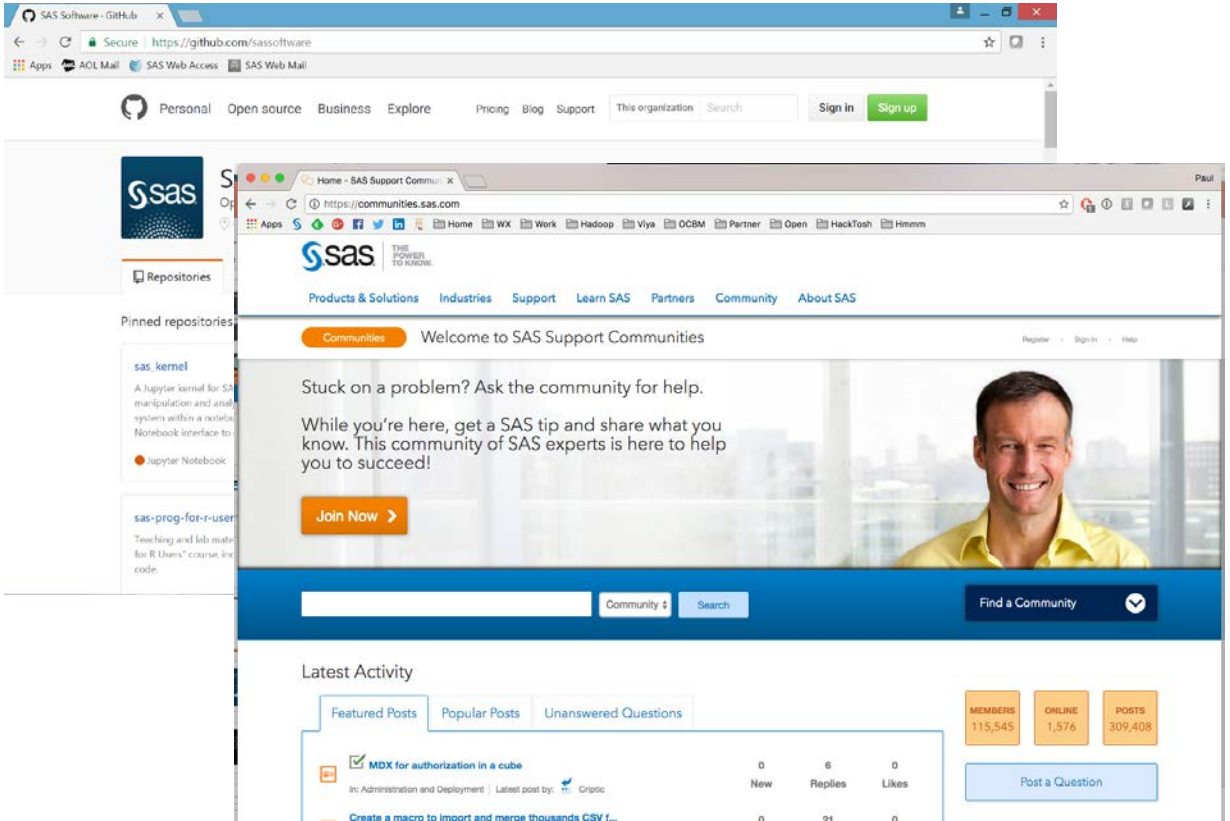
kesmit13 Remove xrange for Python 3 compatibility Latest commit f9f1a9f 19 days ago

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

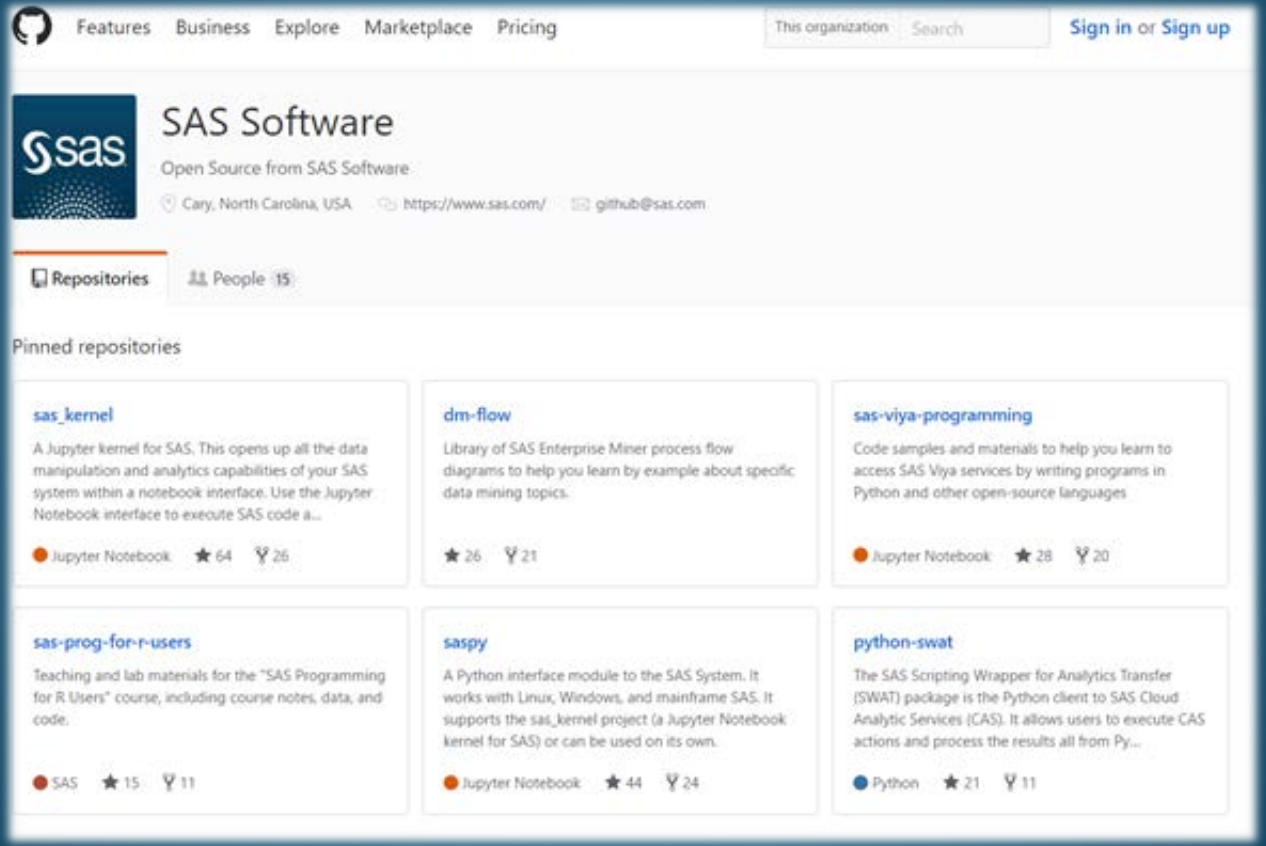
REST APIs



Communities



https://github.com/sassoftware



developer.sas.com