

Combining the Power of RShiny and SAS Viya

David Weik, SAS



Who am I

- Worked as a SAS-Datawarehouse Admin/ETL-Dev/RPA-Dev
- Joined SAS in 2020 in Pre-Sales with a focus on Open-Source integration
- Joined SAS R&D in July 2024 to build and deliver ready to use models



David Weik



Agenda



Quick View on SAS Viya & SAS Visual Analytics



Quick View on RShiny



How do they integrate?



Is that it?



Demo



Summary

Quick View on SAS Viya & SAS Visual Analytics

What is SAS Viya?

End-to-End Analytical Life Cycle Platform

ANALYTICS LIFE CYCLE

Discover Information Assets

Manage Data

Explore and Visualize

Build Models

Manage Models

Build Decisions

Share and Collaborate

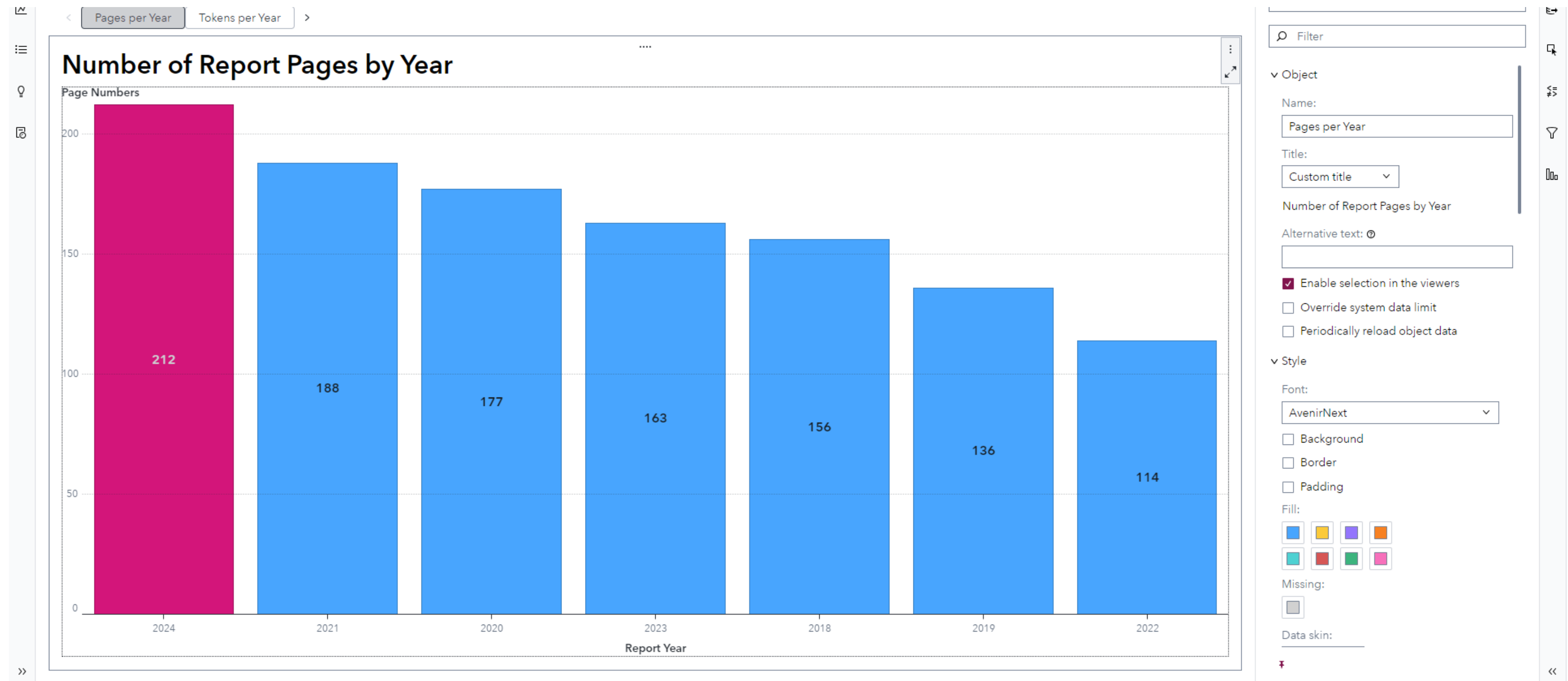
Develop Code and Flows

The screenshot displays the SAS Viya user interface, which is a multi-tabbed web application. The top navigation bar includes the following tabs: SAS® Information Catalog - Discover Information Assets, SAS® Data Explorer - Manage Data, SAS® Visual Analytics - Explore and Visualize, Model Studio - Build Models, and SAS® Studio - Develop Code and Flows. The main workspace is divided into several sections:

- Left Panel:** A 'Steps' panel with a search bar and a list of categories under 'SAS Steps' and 'Shared'. The categories include: Data (Input and Output), Develop, Transform Data, Integrate, Statistics, Data Quality, Econometrics, Visualize Data, Enrichment, Machine Learning, Text Analytics, Manage Models, Optimization and Network Analysis, Examine Data, Prepare Data, and Statistical Process Control.
- Center Panel:** A 'Test-Flow.flw' workspace with a toolbar containing 'Run', 'Cancel', and other icons. Below the toolbar, there are tabs for 'Flow', 'Generated Code', and 'Submitted Code and Results'. The 'Flow' tab is active, showing three swimlanes:
 - Swimlane 1:** A sequence of steps: CARS (Data) → Query (SQL) → SAS Program (Code) → Python Program (Code).
 - Swimlane 2:** A sequence of steps: CARS (Data) → Bar-Line Chart (Visualization).
 - Swimlane 3:** A sequence of steps: CARS (Data) → Automated Feature... (Modeling).
- Bottom Right:** A 'Seed' input field with the value '12,345'.

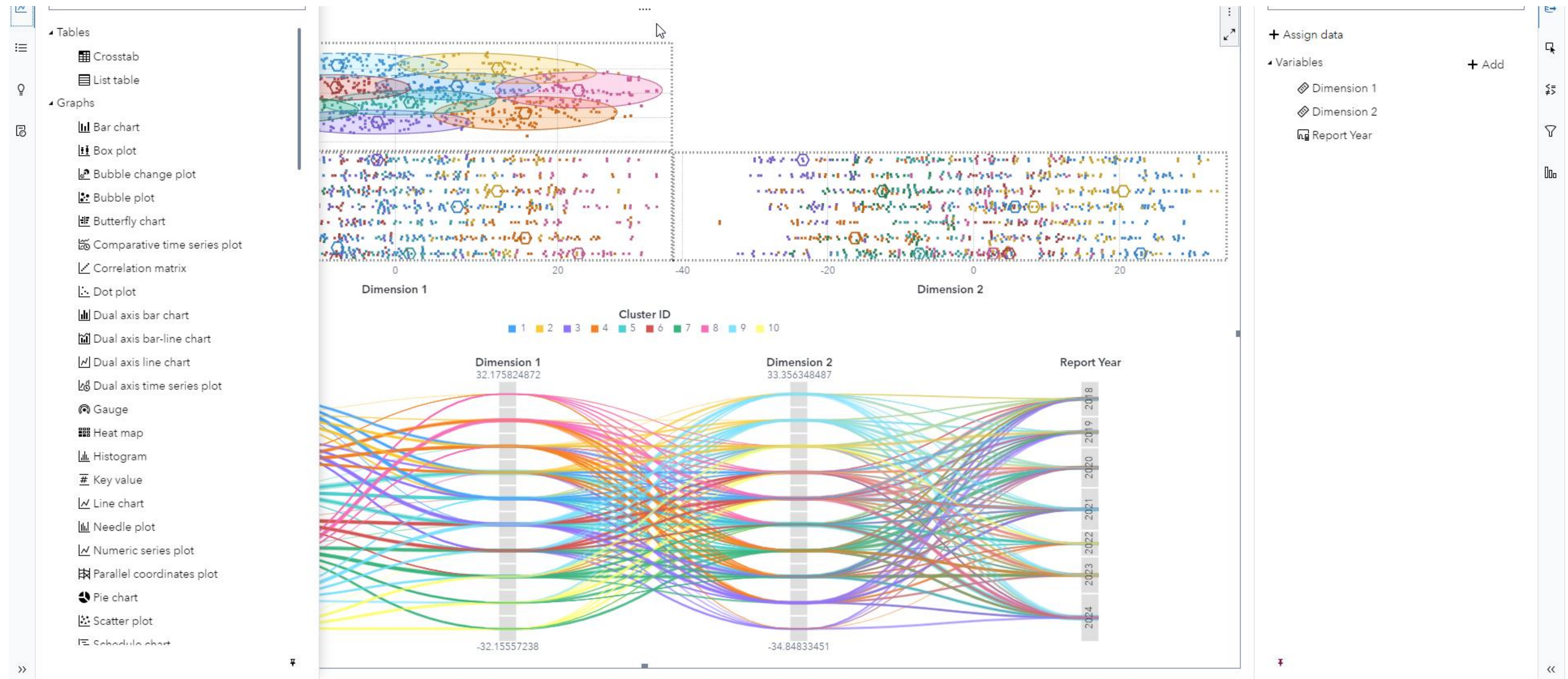
What is SAS Visual Analytics?

Dashboarding



What is SAS Visual Analytics?

Dashboarding and more



What about Data Access?

What can I connect to this?

Select a Data Source

Filter

All SAS Data Stores Storage Locations Cloud Providers Online Services Generic

Path/DNFS	PostgreSQL	Salesforce
SAP HANA	SAP IQ	SAS Federation Server
SAS LASR	SAS SPDE	SingleStore
Snowflake	Spark	Teradata
Vertica	Viya with SingleStore	Yellowbrick

Close

How about access rights?

And how to monitor usage?

		Read/Write	Select	Limite	Create	DropT	Delete	Insert	Updat	Delete	AlterT.	Manag
Authenticated Users	Write 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Guest	Custom	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
gerdaw	Write 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
sas Unix Service Acct	Full Control 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Quick View on RShiny

Quick look at our target R Shiny Application

What we want to do

- Visualizing Data and making use of interactive features in R Shiny to trigger custom calculations
- Authenticate users in order to be able to access the applications
- Access to the data needs to be limited on a row level
- Enable users to consume up to data

Different Components in RShiny

What will we be using?

- Input & Output UI Components
- Layout Components
- Reactive, ggplot2, plotly, renderDataTable, renderText
- Dplyr
- Jsonlite
- Inline scripting

How do they integrate?

Integration with SAS Visual Analytics

Data Driven Content Object

The screenshot displays the SAS Visual Analytics interface. The main window shows a data table with columns: Country, Region, Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), Generosity, and Dystopia Residual. The table contains 15 rows of data. A modal window titled 'Add Data Items' is open, showing a list of data items to be added to the report. The items include Document ID, Document Name, Page Number, Page Text, Report Year, and various topics related to AI, environment, and machine learning. A 'Variables' dialog box is also visible, showing 'Add' and 'Close' buttons. On the right side, the 'Options' panel is open, showing settings for the data-driven content object, including Name, Title, Alternative text, and checkboxes for 'Enable selection in the viewers', 'Extend width if available', and 'Shrink width if necessary'.

Country	Region	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
Denmark	Western Europe	1.44178	0.79504	0.57941	0.44453	0.36171	2.73939
Switzerland	Western Europe	1.52733	0.86303	0.58557	0.41203	0.28083	2.69463
Iceland	Western Europe	1.42666	0.86733	0.56624	0.14975	0.47678	2.83137
Norway	Western Europe	1.57744	0.79579	0.59609	0.35776	0.37895	2.66465
Finland	Western Europe	1.40598	0.81091	0.57104	0.41004	0.25492	2.82596
Canada	North America	1.44015	0.8276	0.5737	0.31329	0.44834	2.70485
Netherlands	Western Europe	1.46468	0.81231	0.55211	0.29927	0.47416	2.70749
New Zealand	Australia and Zealand	1.36066	0.83096	0.58147	0.41904	0.49401	2.47553
Australia	Australia and Zealand		0.8512	0.56837	0.32331	0.47407	2.5465
Sweden	Western Europe		0.83121	0.58218	0.40867	0.38254	2.54734
Israel	Middle East Northern Africa		0.84917	0.36432	0.08728	0.32288	3.31029
Austria	Western Europe		0.80565	0.4355	0.21348	0.32865	2.69343
United States	North America	1.50796	0.779	0.48163	0.14868	0.41077	2.72782
Costa Rica	Latin America Caribbean	1.06879	0.76146	0.55225	0.10547	0.22553	3.35168
Puerto Rico	Latin America Caribbean	1.35943	0.77758	0.46823	0.12275	0.22202	3.0076

Receiving the data in RShiny

JSON data

```
▼ {version: '1', resultName: 'dd85', rowCount: 8, availableRowCount: 8, data: Array(8), ...} ⓘ  
  availableRowCount: 8  
  ▼ columns: Array(2) 1  
    ▶ 0: {name: 'bi101', label: 'Year', type: 'number', usage: 'categorical', format: {...}}  
    ▶ 1: {name: 'bi100', label: 'Value', type: 'number', usage: 'categorical', format: {...}}  
      length: 2  
    ▶ [[Prototype]]: Array(0)  
  ▼ data: Array(8) 2  
    ▶ 0: (2) [2019, 5388]  
    ▶ 1: (2) [2020, 6453]  
    ▶ 2: (2) [2021, -4345]  
    ▶ 3: (2) [2022, 7345]  
    ▶ 4: (2) [2023, 8345]  
    ▶ 5: (2) [2024, 5345]  
    ▶ 6: (2) [2025, 6345]  
    ▶ 7: (2) [2026, 3345]  
    length: 8
```

Sending Interactions back

Reacting to the reaction

```
function sendMessage(message) {
  var url =
    window.location != window.parent.location
      ? document.referrer
      : document.location.href;
  window.parent.postMessage(message, url);
}

// Send instructional message
function sendDataMessage(resultName) {
  let textMessage =
    'This Application requires two variables as input. The first is the
value for the x-axis (any type) and second value for the y-axis (numeric).';
  document.getElementById('graph').innerText = textMessage;

  var message = {
    resultName: resultName,
    message: textMessage,
  };
  sendMessage(message);
}
```


Is that it?

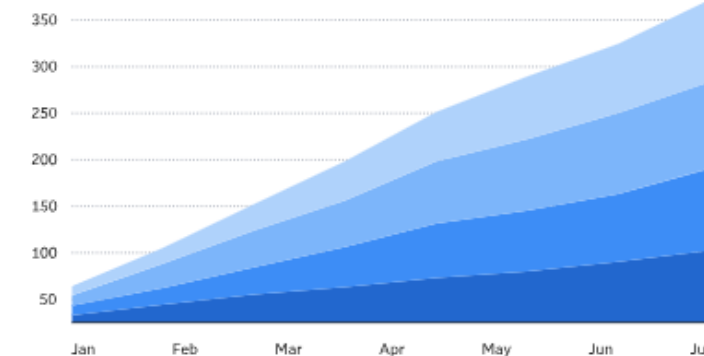
APIs, APIs, APIs

Building on top of services - <https://developers.sas.com/>

The power of SAS analytics in open source


Learn how to build applications that integrate the SAS AI and analytics platform with open source.

Data Insights powered by SAS



Explore REST APIs by Category

All APIs

 **Auto Machine Learning**
Enables CRUD operations on automation ...


 **CAS**
Cloud Analytic Services action and action...


 **Compute and Jobs**
Perform CAS Management, Jobs, and ...

 **Data Management**
Manage all aspects of data quality, data ...

 **Fraud and Compliance**
The SAS Detection API enables you to ...

 **Health and Life Sciences**
CRUD operations for Health and Life ...

 **Models and Decisions**
Manage and publish models; decision flo...

 **Platform Administration**
High level tasks for authorization, files an...

 **Visual Investigator**
Manage data and workflows and search ...

SWAT

Analytical Transfer – Distributed Computation Engine

SAS Scripting Wrapper for Analytics Transfer

API Reference

- Utility Functions
- CAS
- CASResults
- SASDataFrame
- SASFormatter
- CASTable
- CASColumn
- CASTableGroupBy
- CASResponse
- Data Message Handlers
- Date and Time Functions

CASTable

- Constructor
- CAS Connections
- Setters and Getters
- Attributes and Underlying Data
- Indexing, Iteration
- GroupBy
- Computations / Descriptive Stats
- Reindexing / Selection / Label manipulation
- Sorting
- Combining / Merging
- Plotting**

- swat.cas.table.CASTable.plot
- swat.cas.table.CASTable.plot.area
- swat.cas.table.CASTable.plot.bar
- swat.cas.table.CASTable.plot.barh
- swat.cas.table.CASTable.plot.box
- swat.cas.table.CASTable.plot.density
- swat.cas.table.CASTable.plot.hexbin
- swat.cas.table.CASTable.plot.hist
- swat.cas.table.CASTable.plot.kde
- swat.cas.table.CASTable.plot.line
- swat.cas.table.CASTable.plot.pie
- swat.cas.table.CASTable.plot.scatter
- swat.cas.table.CASTable.boxplot
- swat.cas.table.CASTable.hist

Run R Code in SAS

Yes, that is possible

R Runner

Run Cancel | Refresh | Refresh

Parameters Configuration About

Provide an input dataset:
SASHELP.HEART **SAS table**

Note: If you have attached input data to the "inputtable" input port of this custom step, ensure you refer to the same as r_input_table within your R script.

Provide your R script location:

Run an R snippet:
`new_df=aggregate(x=r_input_table$MSRP,by=list(r_input_table$Origin),FUN=mean)` **R Script (type R code or select your R file)**

Provide name of R data frame you wish to output:
new_df

Note: If you are using this step, ensure that the R dataframe you refer does exist in the R session.

Provide output dataset for R environment variables:
WORK.R_ENV **SAS Dataset containing R variables**

Provide desired output table:
WORK.RDF **An R data frame exported as SAS table**

CAS - Submit Python and R Code x +

Run Cancel | Refresh | Refresh

Definition Options About

Select a Python or R script, connect the required in- and output tables for that script and then click run.

The Python and R scripts can be stored both in SAS Content and on the SAS Server, but please ensure that Python scripts end in .py and R scripts end in .R.

Select Python or R script: *

Input table(s)

Connect required input tables, up to a maximum of 10 tables. Input tables will be passed to the open-source program as part of the gateway arguments list (gateway.args for Python, gw\$args for R) as inputtable1-10. These arguments can be used inside the gateway.read_table or other methods. The inputTableCounter can go from 0 to 10 and can be used to keep track of how many input tables are connected.

Python example:
`df = gateway.read_table(gateway.args['inputtable1'])`

R example:
`tbl <- read_table(gw$args[['inputtable']])`

First input table:

Output table(s)

Connect required output tables, up to a maximum of 10 tables. Output table names will be passed to the open-source program as part of the gateway arguments list (gateway.args for Python, gw\$args for R) as outputtable1-10. These arguments can be used inside the gateway.write_table or other methods. The outputTableCounter can go from 0 to 10 and can be used to keep track of how many input tables are connected.

Python example:
`gateway.write_table(df, (gateway.args['outputtable1']))`

R example:
`tbl <- write_table(gw$args[['outputtable']])`

First output table:

Demo

Summary

In Summary

“The Best of Both Worlds” – Hannah Montana

- Code interactive UIs in R + RShiny
- Leverage the authentication and authorization of SAS Viya
- Build on top of row level security to ensure appropriate data access at every level
- Go deeper by moving complex computation from R to SAS (even R code)
- Using a big set of APIs to build full fledged applications

Questions

David.Weik@sas.com



David Weik

