

Handling Missing Values in SAS

SAS 9.4 & SAS Viya

Melodie Rush

Global Customer Success Principal Data Scientist

<https://www.linkedin.com/in/melodierush>

Copyright © SAS Institute Inc. All rights reserved.



Agenda

What are missing values



General Definition

SAS Definition



Why do missing values happen



Reasons

Options



How to manage missing values in SAS



Programming

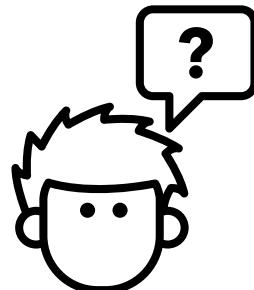
Products



What is Missing Data?

Definition

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data. - [Wikipedia](#)



What is Missing Data?

SAS

Missing Value

- is a value that indicates that no data value is stored for the variable in the current observation. There are three kinds of missing values:
- numeric
- character
- special numeric

By default, SAS prints a missing numeric value as a single period (.) and a missing character value as a blank space. See [Creating Special Missing Values](#) for more information about special numeric missing values.

A photograph of two people in an office setting. A man in a blue sweater is standing and looking thoughtfully upwards, with his hand near his chin. A woman in a purple cardigan is seated at a desk in the foreground, looking towards the man. The background shows office windows.

Why is the data missing?

Missing Completely At Random (MCAR)

The probability of missingness doesn't depend on anything.



Missing At Random (MAR)

The probability of missingness does not depend on the unobserved value of the missing variable, but it can depend on any of the other variables in your dataset



Not Missing at Random (NMAR)

The probability of missingness depends on the unobserved value of the missing variable itself





When should you be concerned?



Reporting

- May draw inaccurate conclusions or inference about the data

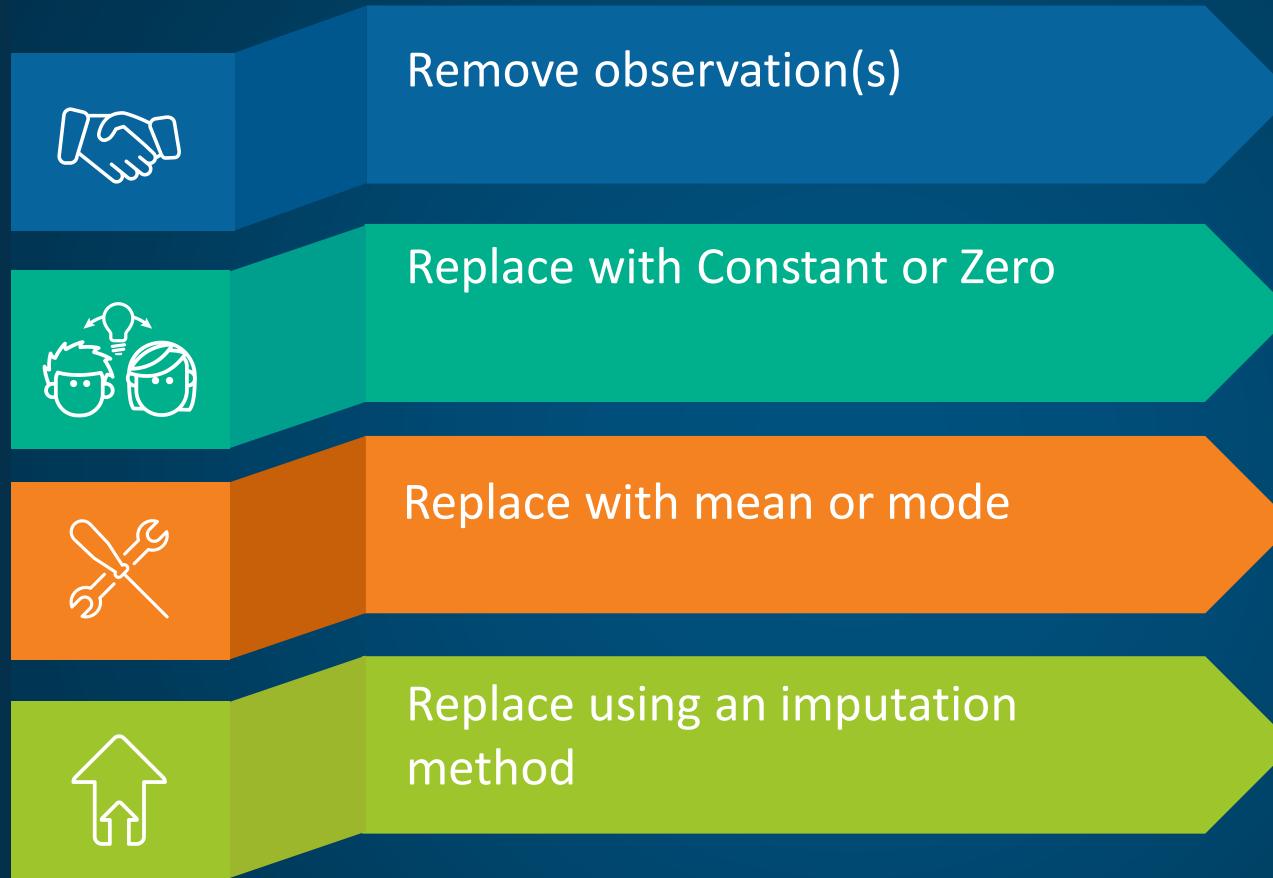


Predictive Modeling

- Bias in the estimation of parameters
- Significant effect on the conclusions



What should you do about missing values?



Functions

Useful Functions

Character

MISSING(*expression*)

- Returns a number that indicates whether the argument contains a missing value.
 - `missing_flag = missing(gender);`

CMISS(*argument-1 <, argument-2,...>*)

- Counts the number of missing arguments.
 - `char_miss = cmiss(BP_Status, Chol_Status, Smoking_Status, Weight_Status);`

COALESCEC(*expression[, ...expression]*)

- Returns the first non-null or nonmissing value from a list of character arguments.
 - `charvar = coalescec(charvar, "***NOT ANSWERED***");`

[Function Documentation](#)

Useful Functions

Numeric

MISSING(*expression*)

- Returns a number that indicates whether the argument contains a missing value.
 - `missing_flag = missing(income);`

NMISS(*argument-1 <... argument-n>*)

- Returns the number of missing numeric values.
 - `num_miss = nmiss(AgeAtDeath, AgeAtStart, AgeCHDdiag);`

COALESCE(*argument-1<..., argument-n>*)

- Returns the first nonmissing value from a list of numeric arguments.
 - `numvar = coalesce(numvar, 1000);`

Useful Functions

Numeric

Both return zero for missing values

- $y=\text{sum}(x,0);$
- $y=\text{coalesce}(x,0);$

In SQL use the mean or coalesce function

- case when missing(var1) then mean(var1) else var1 end as var1
- coalesce(var1, mean(var1)) as var1

Procedures



Remove observations



WHERE

In data steps, proc SQL, and procedures



IF

In data steps



CASE

proc SQL

Replace Value(s)

PROC STDIZE



Replacing Missing Values

PROC STDIZE

- Replace all numeric missing values with zero

```
PROC STDIZE data=table1 out=table2 reponly missing=0;  
run;
```

reponly – only replace; do not standardize

missing – can be any constant

- Replace all numeric missing values with the mean

```
PROC STDIZE data=table1 out=table2 reponly method=mean;  
var _numeric_;  
run;
```

method – includes MEDIAN, SUM and others for doing standardization activities

PROC STANDARD



Replacing Missing Values

PROC STANDARD

- Replace all numeric missing values with mean

```
PROC STANDARD data=table 1 out=table2 replace;  
run;
```



Use Imputation Method

PROC HPIMPUTE



Replacing Missing Values

PROC HPIMPUTE

```
proc hpimpute data=sampsio.hmeq out=out1;  
    input mortdue value clage debtinc;  
    impute mortdue / value = 70000;  
    impute value / method = mean;  
    impute clage / method = random;  
    impute debtinc / method = pmedian;  
run;
```

Imputation Results					
Variable	Imputation Indicator	Imputed Variable	N Missing	Type of Imputation	Imputation Value (Seed)
MORTDUE	M_MORTDUE	IM_MORTDUE	518	Given value	70000
VALUE	M_VALUE	IM_VALUE	112	Mean	101776
CLAGE	M_CLAGE	IM_CLAGE	308	Random	5.00000
DEBTINC	M_DEBTINC	IM_DEBTINC	1267	Pseudo Median	34.81696

MEAN, RANDOM, PMEDIAN or Constant Value

[HPIMPUTE Procedure Documentation](#)

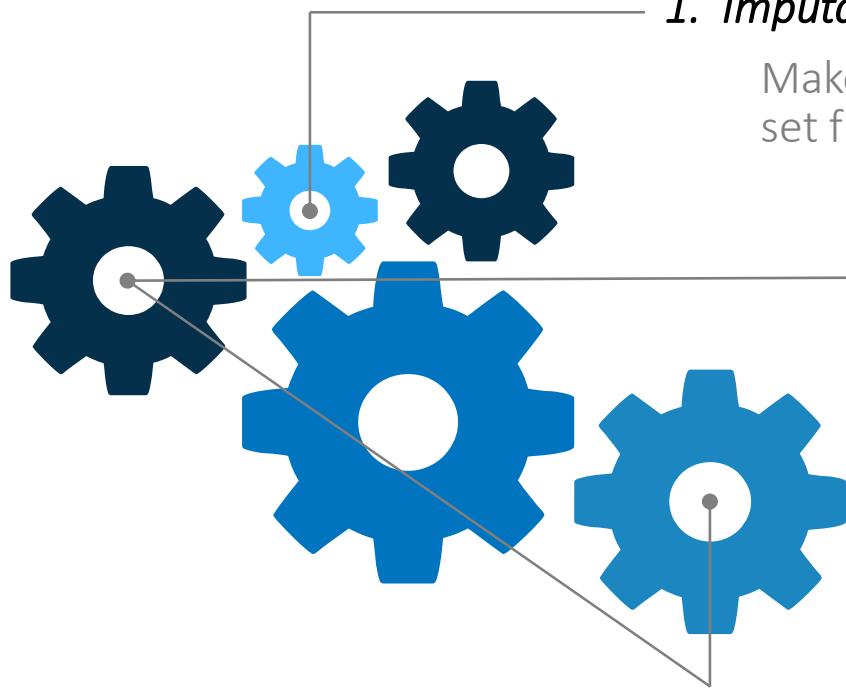
Multiple Imputation

PROC MI & PROC MIANALYZE



Multiple Imputation In 3 Steps

Using PROC MI & PROC MIANALYZE



1. *Imputation*

Make $M=5$ to $>25^*$ copies of incomplete data set filling in with conditionally random values

2. *Analysis*

Of each data set separately

3. *Pooling*

- *Point estimates.* Average across M analyses
- *Standard errors.* Combine variances

[PROC MI Documentation](#)

Multiple Imputation

Step 1 Proc MI Example: Imputation

Oxygen	RunTime	RunPulse
44.609	11.37	178
54.297	8.65	156
49.874	9.22	.
.	11.95	176
39.442	13.08	174
50.541	.	.
44.754	11.12	176
51.855	10.33	166
40.836	10.95	168
46.774	10.25	.
39.407	12.63	174
45.441	9.63	164

```
PROC MI data=mi_example out=outmi  
    seed=1234;  
    var Oxygen RunTime RunPulse;  
    run;
```

Multiple Imputation

Step 1:
Imputation

Step 2:
Analysis

Step 3:
Pooling

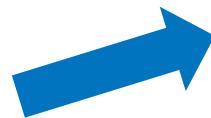
Oxygen	RunTime	RunPulse
44.609	11.37	178
54.297	8.65	156
49.874	9.22	.
.	11.95	176
39.442	13.08	174
50.541	.	.
44.754	11.12	176
51.855	10.33	166
40.836	10.95	168
46.774	10.25	.
39.407	12.63	174
45.441	9.63	164

```
PROC MI data=mi_example out=outmi  
    seed=1234;  
    var Oxygen RunTime RunPulse;  
    run;
```

Multiple Imputation

Step 1 Results: Imputation

Oxygen	RunTime	RunPulse
44.609	11.37	178
54.297	8.65	156
49.874	9.22	
.	11.95	176
39.442	13.08	174
50.541	.	
44.754	11.12	176
51.855	10.33	166
40.836	10.95	168
46.774	10.25	
39.407	12.63	174
45.441	9.63	164



Multiple imputed
datasets created

Oxygen	RunTime	RunPulse
44.609	11.37	178
54.297	8.65	156
49.874	9.22	169.856
48.987	11.95	176
39.442	13.08	174
50.541	10.932	178.697
44.754	11.12	176
51.855	10.33	166
40.836	10.95	168
46.774	10.25	157.241
39.407	12.63	174
45.441	9.63	164



Oxygen	RunTime	RunPulse
44.609	11.37	178
54.297	8.65	156
49.874	9.22	173.309
50.095	11.95	176
39.442	13.08	174
50.541	11.769	158.932
44.754	11.12	176
51.855	10.33	166
40.836	10.95	168
46.774	10.25	161.803
39.407	12.63	174
45.441	9.63	164

Multiple Imputation

Step 1:
Imputation

Step 2:
Analysis

Step 3:
Pooling

```
PROC REG data=outmi outest=outreg covout noint;  
    model Oxygen = RunTime RunPulse;  
    by _Imputation_;  
run;
```

Other model options: Reg, Logistic, Genmod, Mixed, GLM

Multiple Imputation

Step 2 Results: Parameter Estimates & Covariance Matrices

```
PROC PRINT data=outreg (obs=8);  
  var _Imputation_ _Type_ _Name_ Intercept RunTime RunPulse;  
run;
```

Obs	_Imputation_	_TYPE_	_NAME_	Intercept	RunTime	RunPulse
1	1	PARMS		82.9694	-2.44422	-0.06121
2	1	COV	Intercept	65.1698	0.26463	-0.39518
3	1	COV	RunTime	0.2646	0.14005	-0.0101
4	1	COV	RunPulse	-0.3952	-0.0101	0.00293
5	2	PARMS		85.1831	-3.0485	-0.03452
6	2	COV	Intercept	85.3406	-0.44671	-0.46786
7	2	COV	RunTime	-0.4467	0.13629	-0.00581
8	2	COV	RunPulse	-0.4679	-0.00581	0.00308

Multiple Imputation



```
PROC MIANALYZE data=outreg;  
  modeleffects Intercept RunTime RunPulse;  
run;
```

Replaces var Notice the dependent variable is
not included here

Multiple Imputation Parameter Estimates								
Parameter	Estimate	Std Error	95% Confidence Limits		DF	Minimum	Maximum	Pr > t
Intercept	92.696519	12.780914	65.35758	120.0355	14.412	82.969385	101.288118	<.0001
RunTime	-2.915452	0.48346	-3.90873	-1.9222	26.264	-3.146336	-2.444217	<.0001
RunPulse	-0.086795	0.070425	-0.23209	0.0585	24.163	-0.13547	-0.034519	0.2296

PROC SURVEYIMPUTE

PROC SURVEYIMPUTE

Brand new in SAS/Stat 14.1

Impute missing values – PROC SURVEYIMPUTE

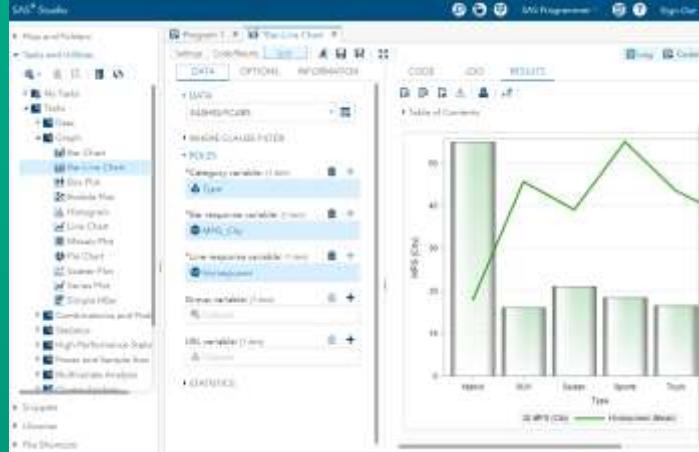
The SURVEYIMPUTE procedure imputes missing values of an item in a sample survey by replacing them with observed values from the same item. Imputation methods include single and multiple hot-deck Imputation, Approximate Bayesian bootstrap (ABB) imputation, Fractional hotdeck imputation (FHD), and fully efficient fractional imputation (FEFI)

```
/* Joint imputation for Department and Response*/
proc surveyimpute data=SIS_Survey_Sub method=fefi varmethod=jackknife;
    class Department Response;
    var Department Response;
    strata State NewUser;
    cluster School;
    weight SamplingWeight;
    output out=SIS_Survey_Imputed outjkcoefs=SIS_JKCoefs;
run;
```

[PROC SURVEYIMPUTE Documentation](#)

Products

SAS Studio



Replacing Missing Values

SAS Studio



3 Ways

1. **Task – Data → Describe Missing Data**
2. **Task – Data → Standardize Data**
 - Omit row
 - Replace Value
3. **SAS Code**
 - PROC STDIZE - [documentation](#)
 - PROC STANDARD - [documentation](#)
 - PROC HPIMPUTE - [documentation](#)
 - SAS/STAT PROC MI - [documentation](#)

Describe Missing Data Task Frequencies

Tasks

Data

List Table Attributes

Characterize Data

Describe Missing Data

List Data

Transpose Data

Stack/Split Columns

Filter Data

Select Random Sample

Partition Data

Sort Data

Rank Data

Transform Data

Standardize Data

Missing Data Frequencies

Legend: ., A, B, etc = Missing

Default or seriously delinquent

REASON	Frequency	Percent
Non-missing	5960	100.00
	252	4.23

Amount of current loan request

LOAN	Frequency	Percent
Non-missing	5960	100.00
	279	4.68

Amount due on existing mortgage

MORTDUE	Frequency	Percent
.	518	8.69
Non-missing	5442	91.31

Home improvement or Debt Consolidation

REASON	Frequency	Percent
Non-missing	5708	95.77
	252	4.23

Prof/Exec/Office/Self/Other

JOB	Frequency	Percent
Non-missing	5681	95.32
	279	4.68

Years on current job

YOJ	Frequency	Percent
Non-missing	5445	91.36
.	515	8.64

Describe Missing Data Task

Missing Data Pattern

Missing Data Patterns across Variables

Legend: ., A, B, etc = Missing

Default or seriously delinquent	Amount of current loan request	Amount due on existing mortgage	Value of current property	Home improvement or Debt Consolidation	Prof/Exec/Office/Self/Other	Years on current job	No. of major derogatory reports	No. of delinquent credit lines	Age of oldest credit line in months	No. of recent credit inquiries	No. of trade credit lines	Debt to income ratio	Frequency	Percent
Non-missing	Non-missing	.	-			.	.	-	2	0.0338
Non-missing	Non-missing	.	-			.	.	-	.	.	.	Non-missing	6	0.1007
Non-missing	Non-missing	.	-			Non-missing	Non-missing	Non-missing	.	Non-missing	Non-missing	.	1	0.0168
Non-missing	Non-missing	.	-		Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	1	0.0168
Non-missing	Non-missing	.	-	Non-missing		Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	2	0.0338
Non-missing	Non-missing	.	-	Non-missing		Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	1	0.0168
Non-missing	Non-missing	.	-	Non-missing	Non-missing	.	.	-	2	0.0338
Non-missing	Non-missing	.	-	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	2	0.0338

Standardize Data Task

Select Missing Values Method & How to replace missing values

The screenshot shows the SAS Studio interface with the 'Standardize Data' task selected in the left sidebar. The main window displays the task configuration with two tabs: 'OPTIONS' (selected) and 'DATA'. The 'OPTIONS' tab includes sections for 'METHODS' (with 'Center data only' checked) and 'TREATMENT OF MISSING VALUES'. The 'TREATMENT OF MISSING VALUES' section has a dropdown for 'Missing values method' set to 'Replace missing value'. A secondary dropdown for 'Replace missing values with:' is open, showing options: 'Default location measure' (selected), 'Mean', 'Median', 'Minimum', and 'Specify custom value'. The 'Default location measure' option is highlighted with a green box.

Tasks

- My Tasks
- SAS Tasks
- Data
- List Table Attributes
- Characterize Data
- Describe Missing Data
- List Data
- Transpose Data
- Stack/Split Columns
- Filter Data
- Select Random Sample
- Partition Data
- Sort Data
- Rank Data
- Transform Data
- Standardize Data
- Recode Values
- Recode Ranges

Program: Standardize Data.ctk

Submit Cancel History Add a

DATA OPTIONS OUTPUT INFORMATION

METHODS

Center data only

Standardization method:

TREATMENT OF MISSING VALUES

Missing values method:

- Omit observations with missing values
- Omit observations with missing values**
- Replace missing value

TREATMENT OF MISSING VALUES

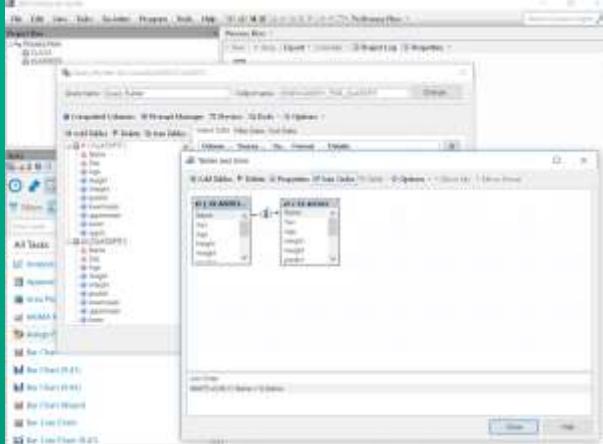
Missing values method:

Replace missing value

Replace missing values with:

- Default location measure**
- Mean
- Median
- Minimum
- Specify custom value

SAS Enterprise Guide



Replacing Missing Values

SAS Enterprise Guide

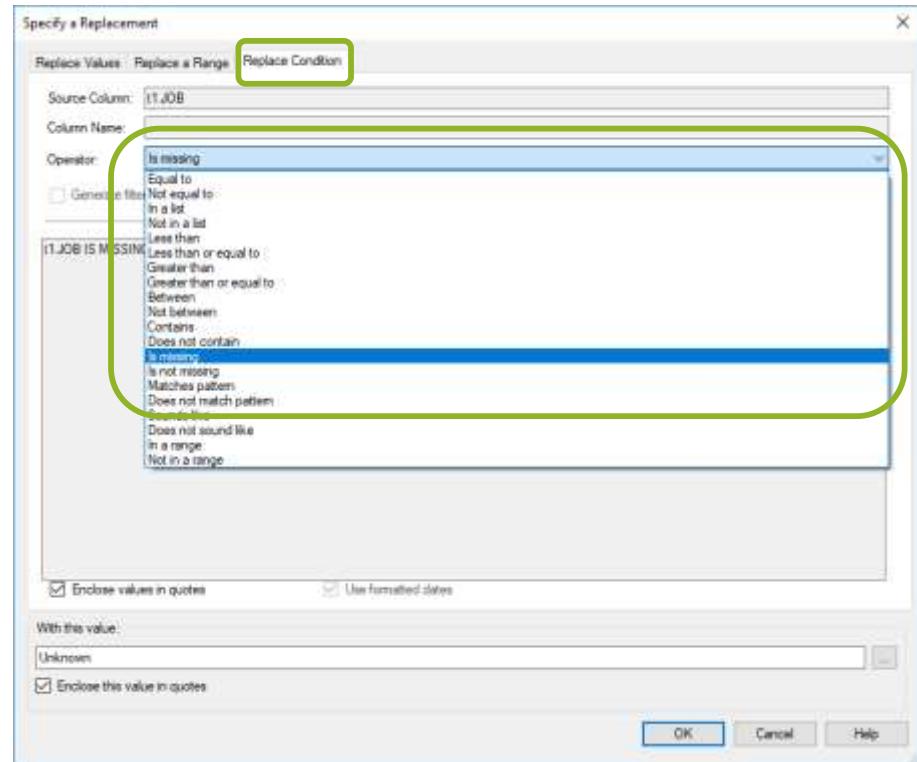
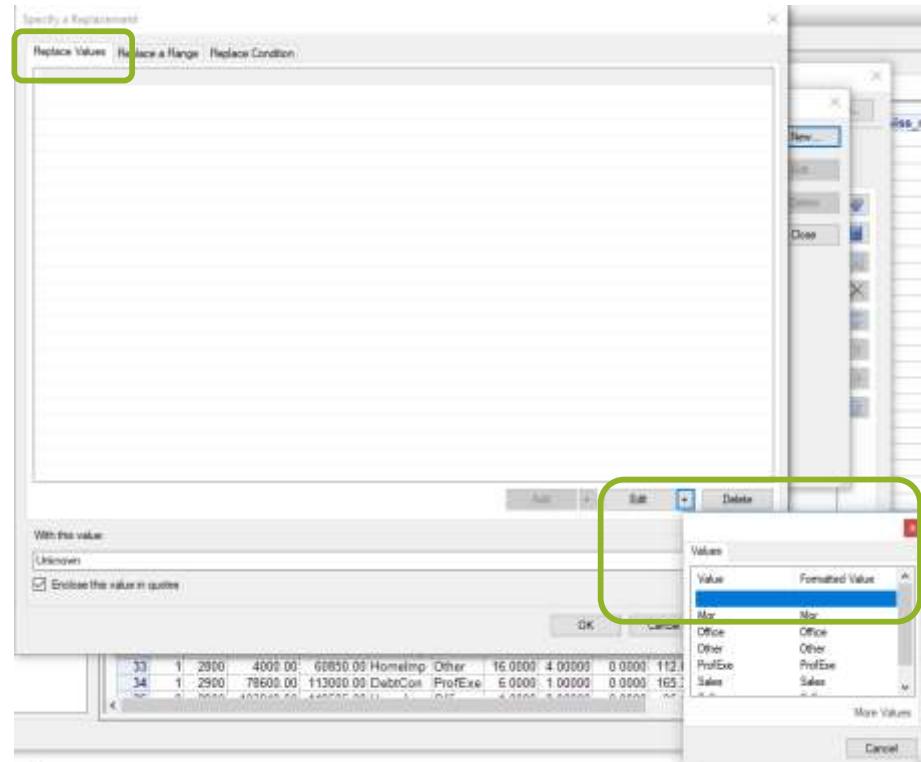


3 Ways

1. **Task -** Query Builder using Computed Column → Replace Values (*numeric & character*)
2. **Task –** Data → Standardize Data (*numeric only - replaces with mean*)
3. **SAS Code**
 - PROC STDIZE - [documentation](#)
 - PROC STANDARD - [documentation](#)
 - PROC HPIMPUTE - [documentation](#)
 - SAS/STAT PROC MI - [documentation](#)

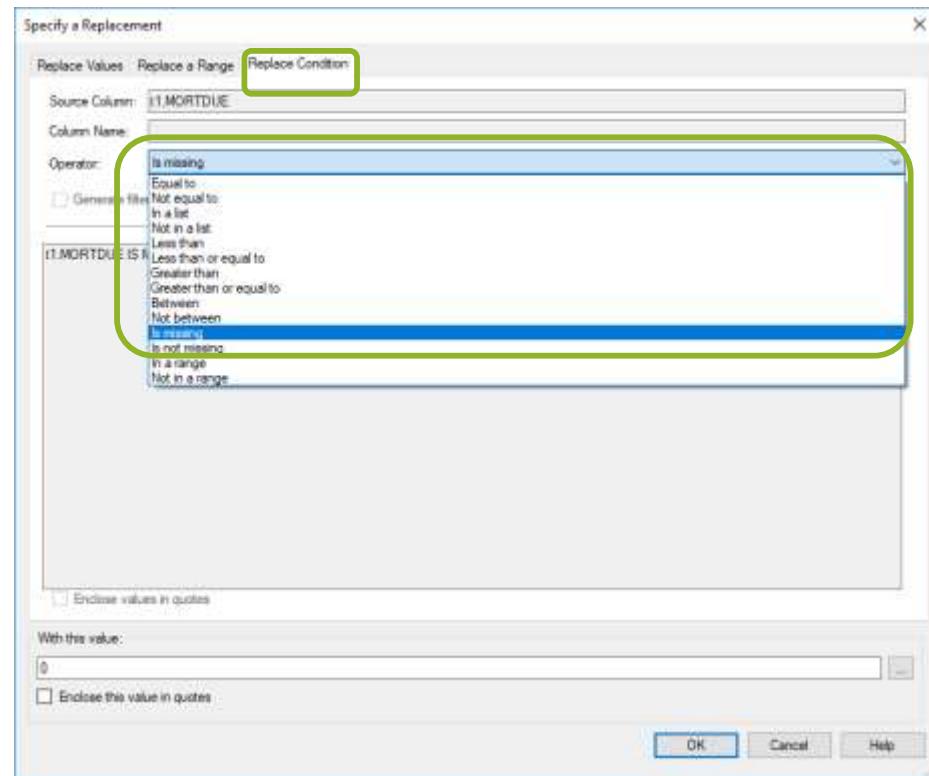
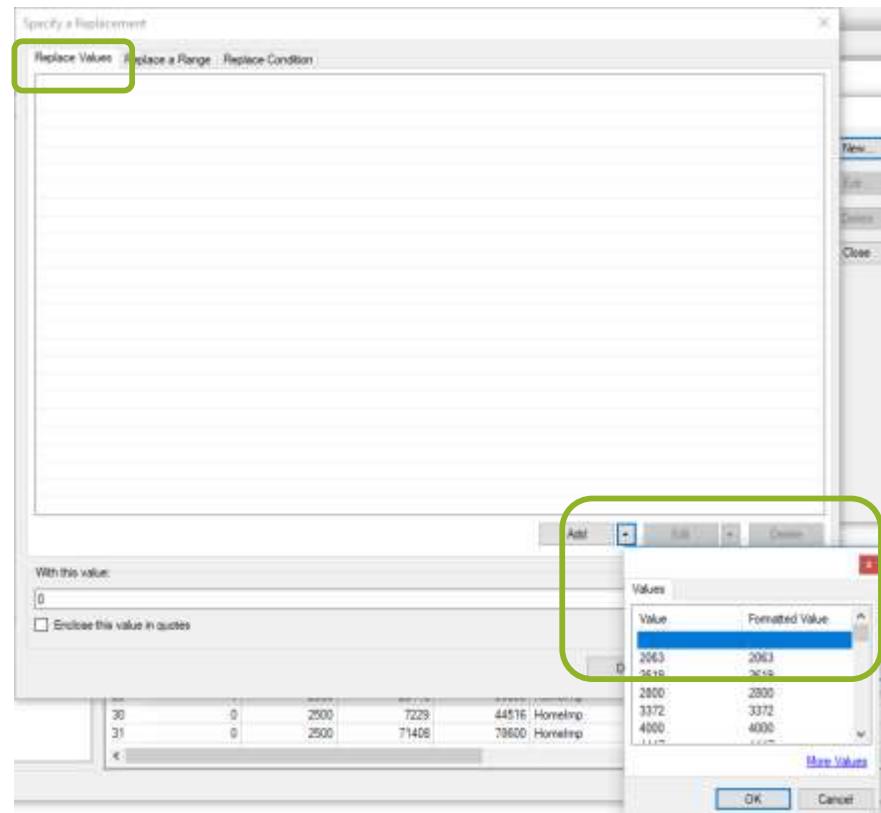
Query Builder Task

Replace Value or Replace Condition for Character Variable



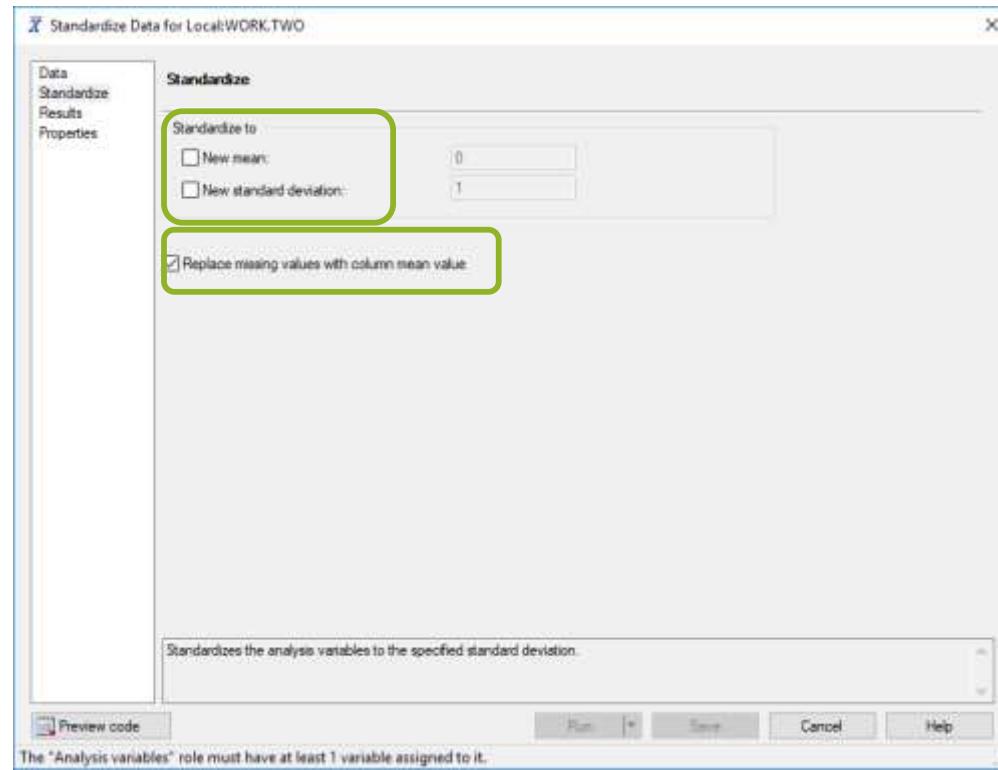
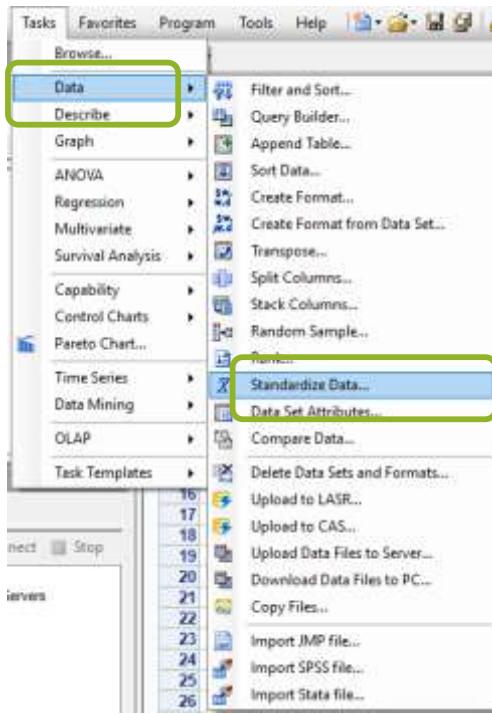
Query Builder Task

Replace Value or Replace Condition for Numeric Variable

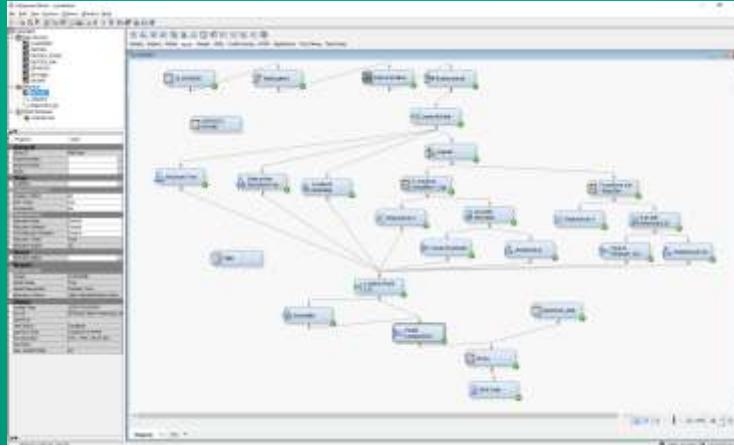


Standardize Data Task

Uncheck New mean & New standard deviation

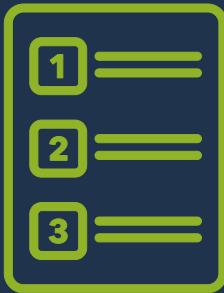


SAS Enterprise Miner



Replacing Missing Values

SAS Enterprise Miner



3 Ways

1. Replacement Node

- Missing values with constants

2. Impute Node

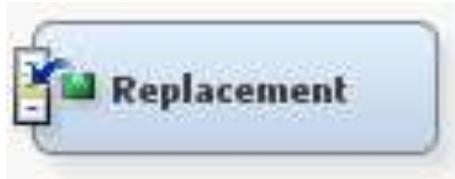
- **Class variables** – count, default constant value, distribution, tree, tree surrogate
- **Target variables** – count, default constant value, distribution
- **Interval variables** – mean, median, midrange, distribution, tree, tree surrogate, mid-minimum spacing, Tukey's Biweight, Huber, Andrew's Wave, default constant

3. SAS Code Node

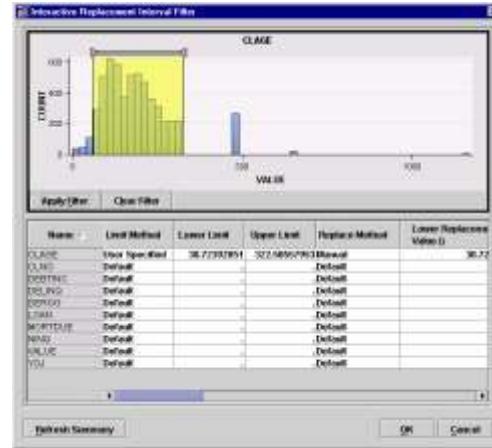
- PROC STDIZE - [documentation](#)
- PROC STANDARD - [documentation](#)
- PROC HPIMPUTE - [documentation](#)
- SAS/STAT PROC MI - [documentation](#)

Replacing Missing Values

Replacement Node



- Used to interactively specify replacement values for class and interval levels
 - Trim outliers
 - Replace Missing
- Use to generate score code to process unknown levels when scoring



Variable	Level	Frequency	Type	Char Raw Value	Num Raw Value	Replacement Value
CLAGE	0	4771	N		0.0	
CLAGE	1	1189	N		1.0	
CLAGE	UNKNOWN		N			DEFAULT
CLM	Other	2380	C	Other		
CLM	ProfExe	1278	C	ProfExe		
CLM	Office	948	C	Office		
CLM	Mgr	787	C	Mgr		
CLM		279	C			
CLM	Self	193	C	Self		
CLM	Sales	109	C	Sales		
CLM	UNKNOWN		C			DEFAULT
DEBTINC	DebtCon	3920	C	DebtCon		
DEBTINC	HomeImp	1780	C	HomeImp		
DEBTINC		252	C			DEFAULT
DEBTINC	UNKNOWN		C			DEFAULT

Replacing Missing Values

Impute Node



- Used to replace missing values
- Many modeling techniques will drop rows of data that have any missing values
- Creates imputation indicator variables

The screenshot shows the 'Train' tab of the 'Impute' node configuration window. It includes sections for 'Variables', 'Non Missing Variables' (set to 'No'), 'Missing Cutoff' (set to '50.0'), and two main groups: 'Class Variables' and 'Interval Variables'. Under 'Class Variables', 'Count' is selected. Under 'Interval Variables', 'Mean' is selected. Other options like 'Default Input Method' (Maximum, Minimum, Median, Midrange, Distribution, Tree), 'Default Target Method' (Tree, Tree Surrogate), and 'Method Options' (Random Seed, Tuning Parameters, Tree Imputation) are also visible. The 'Score' tab at the bottom has settings for 'Indicator Variables' (Type: None, Source: Imputed Variables, Role: Rejected).

- Class Variables
- Interval Variables

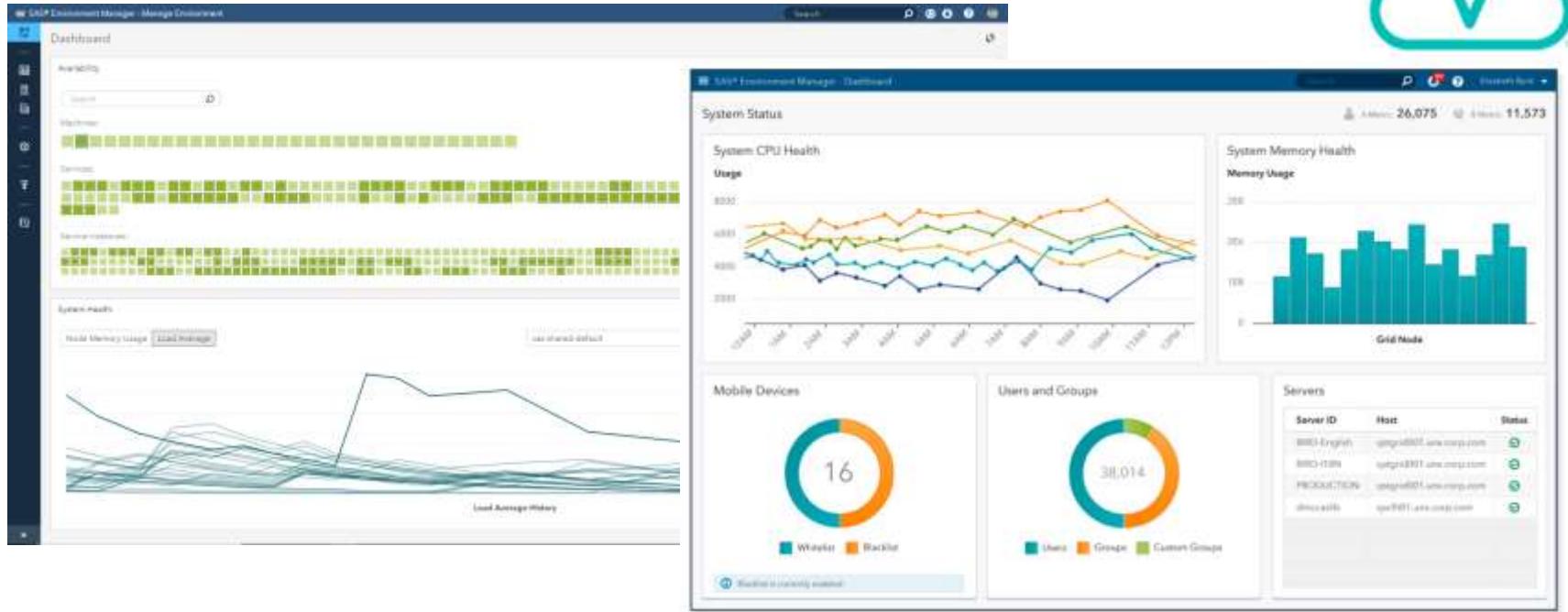


SAS Viya

Visual Statistics, Visual Data Mining and Machine Learning

What is SAS Viya?

SAS Viya is a cloud-enabled, in-memory analytics engine that provides quick, accurate and reliable analytical insights.



SAS Viya Products

SAS Viya takes advantage of a cloud-enabled, open platform. Most offerings include both a coding interface as well a visual interface.

- SAS Visual Analytics
- SAS Visual Statistics
- SAS Visual Data Mining and Machine Learning
- SAS Visual Forecasting
- SAS Visual Text Analytics
- SAS Optimization
- SAS Econometrics
- SAS Model Manager
- SAS Data Preparation
- SAS Visual Investigator
- SAS Business Analytics
- SAS Intelligent Decisioning
- SAS Cybersecurity
- SAS Detection and Investigation
- SAS Event Stream Processing
- And more...





MULTIPLE INTERFACES, SINGLE CODE BASE



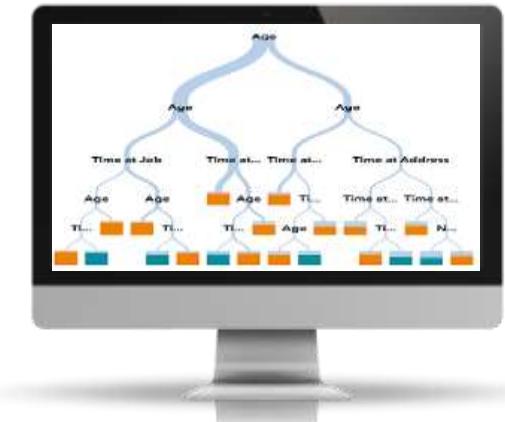
Visual Interfaces



Programming Interfaces



API Interfaces



Visual Interfaces



Visual Interfaces

Explore and Visualize Data

SAS® Visual Analytics - Explore and Visualize Data

Demo VDDML Start

Logistic Regression Default or seriously delinquent (event=1) Validation Misclassification Rate (Event) 0.1063 Observations Used 5,960

Create pipeline

Fit Summary

Variable	p-value
No. of delinquent credit lines	<0.00001
Value of current property_mean	<0.00001
Debt to income ratio	<0.00001
Age of oldest credit line in months	<0.00001
Debt to income ratio_mean	<0.00001
No. of major derogatory reports	<0.00001
Prof/Exec/Office/Self/Other	<0.00001
No. of major derogatory reports_mean	<0.00001
No. of recent credit inquiries	<0.00001
No. of trade credit lines_mean	<0.00001
Value of current property_mean	<0.00001
No. of delinquent credit lines_mean	<0.00001
Age of oldest credit line in months_mean	<0.00001
Years on current job	<0.00001
No. of trade credit lines	<0.00001

Residual Plot

Lift

Options

Logistic Regression

- General
- Event level: 1
- Informative missingness
- Variable selection method: None
- Link function: Logit

Convergence

Assessment

Model Display

Visual Interfaces

Explore and Visualize Data

The screenshot shows the SAS Visual Analytics interface with a green box highlighting a tooltip and a green arrow pointing to a specific option in the Options panel.

Left Panel (Toolbox): Shows categories like Date, Objects, Fit, Agg, Freq, Not Null, and Null Agg.

Top Bar: Shows the title "SAS® Visual Analytics - Explore and Visualize Data", a search bar, and user information "sesdemo".

Central Area: A box plot visualization for a variable named "Log". The box plot shows a median of 1, a Q1 of 0, and a Q3 of 735. Below the box plot is a descriptive text box:

Extends the model to include observations with missing values. A continuous effect is imputed with the observed mean, and an indicator variable that denotes missingness is created. A classification effect treats missing values as a distinct level.

Bottom Area: A histogram of p-values. The x-axis ranges from 0.1 to <0.00001. The y-axis shows the count of observations. A legend indicates "Model" (green) and "Best" (orange).

Right Panel (Options): The "Logistic Regression" section contains the following settings:

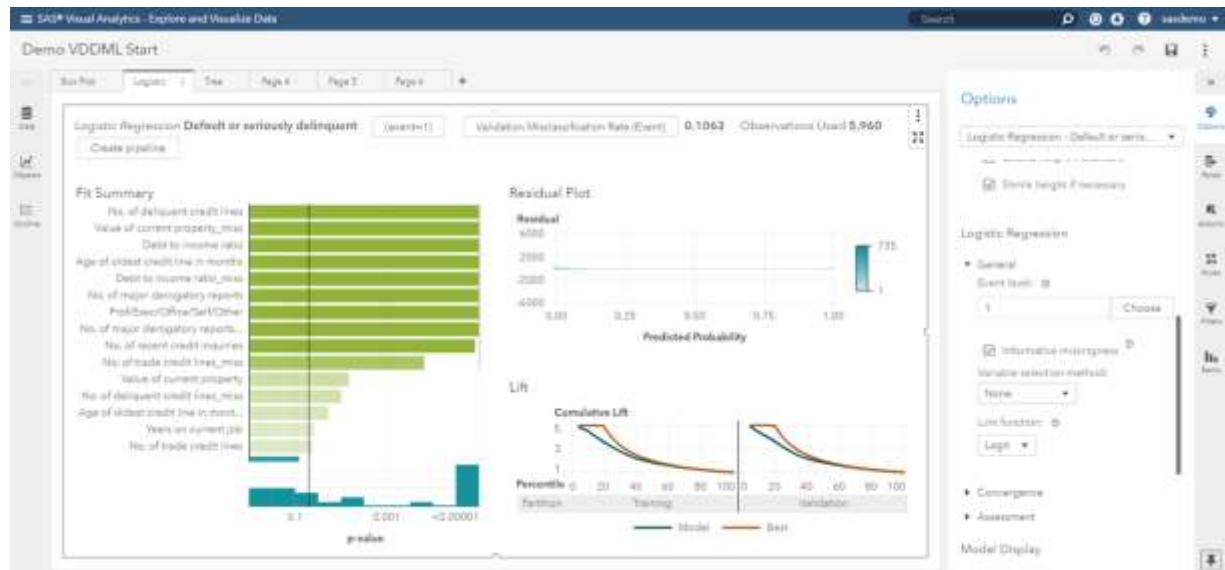
- General:** Event level: ①
- Missing Data:** Shrink height if necessary (checkbox)
- Informative missingness:** (checkbox, highlighted by a green box and arrow)
- Variable selection method:** None
- Link function:** Logit
- Convergence:**
- Assessment:**
- Model Display:**

Visual Interfaces

Explore and Visualize Data

Available for

- Logistic Regression
LOGSELECT
- Linear Regression
REGSELECT
- Generalized Linear Model
GENSELECT
- Neural Network
NNET
- Support Vector Machines
SVMACHINE



Visual Interfaces

Explore and Visualize Data

SAS® Visual Analytics - Explore and Visualize Data

Demo VDDML

Data

HMEQ

Filter

New data item

Category

- Default or seriously delinquent - 2
- Home improvement or Deb... - 3
- Partition - 2
- Prof/Exec/Office/Self/Other - 7

Measure

- Age of oldest credit line in months
- Amount due on existing mortgage
- Amount of current loan request
- Debt to income ratio
- Frequency

Forrest GB Co

ously delinquent (event)

Add data source...

New data source join...

Save data view...

Data views...

Join data to HMEQ

Remove data source

Change data source...

Refresh data source

Apply data source filter...

Map data sources...

Set unique identifier data item...

View measure details...

New aggregated data source...

SAS® Visual Analytics - Explore and Visualize Data

Demo VDDML

Data

HMEQ

Filter

New data item

Outline

Hierarchy...

Custom category...

Calculated item...

Geography item...

Parameter...

Interaction effect...

Spline effect...

Partition...

delinqu... - 2

or Deb... - 3

f/Other - 7

line in months

ting mortgage

an request

Available

- Filter out missing values
- Replace with constant

Visual Interfaces

Prepare Data

Use Code or Calculated Column

- Replace with Constant or Zero
- Code for imputation

The screenshot shows the SAS Data Studio interface for 'Prepare Data'. On the left, a sidebar menu under 'Transforms' lists several options: 'Add Transform' (selected), 'Column Transforms' (including Change case, Convert column, Remove, Rename, Split, Trim whitespace), 'Custom Transforms' (Calculated column, selected and highlighted with a green box), and 'Data Quality Transforms' (Casing, Field extraction, Gender analysis, Identification analysis, Match and cluster, Matchcodes, Parsing). In the center, a code editor titled 'Code - Step 1 of 1' contains the following SAS code:

```
1 /* BEGIN data step with the output table data */
2 data {{_dp_outputTable}} (caslib={{_dp_outputCaslib}} promote="no");
3 /* Set the input set */
4 set {{_dp_inputTable}} (caslib={{_dp_inputCaslib}} );
5 IF MORTDUE <= . THEN MORTDUE=0;
6 /* END data step run */
7 run;
```

Below the code editor is a data preview table titled 'HMEQ(session)'. The table has 12 columns: BAD, LOAN, MORTDUE, VALUE, REASON, JOB, YOJ, DEROG, DELINQ, CLAGE. The first five rows of data are:

BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE
1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.36666...
1	1300	70053	68400	HomeImp	Other	7	0	2	121.8333...
1	1500	13500	16700	HomeImp	Other	4	0	0	149.4466...
1	1500	0	-	-	-	-	-	-	-

Visual Interfaces

Prepare Data

The screenshot shows the SAS Data Studio interface for 'Prepare Data'. On the left, a sidebar lists various transform options under 'Transforms': Add Transform, Column Transforms (Change case, Convert column, Remove, Rename, Split, Trim whitespace), Custom Transforms (Calculated column, Code), and Data Quality Transforms (Casing, Field extraction, Gender analysis, Identification analysis, Match and cluster, Matchcodes, Parsing). The 'Code' section is currently selected. In the main workspace, 'Plan 1+' is active. The 'Code' tab displays a DATA step with the following code:

```
1 /* BEGIN data step with the output, cause data */
2 data ({_dp_outputTable}) (caslib={_dp_outputCaslib}) promote="no";
3 /* Set the input set */
4 set {_dp_inputTable} (caslib={_dp_inputCaslib});
5 if MORTDUE = . then MORTDUE=0;
6 /* END data step run */
7 run;
```

The 'HMEQ (session)' data preview shows the first few rows of a dataset with columns: BAD, LOAN, MORTDUE, VALUE, REASON, JOB, YOJ, DEROG, DELINQ, CLAGE. The data is as follows:

BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE
1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.36666...
1	1300	70053	68400	HomeImp	Other	7	0	2	121.8333...
1	1500	13500	16700	HomeImp	Other	4	0	0	149.4666...
1	1500	0	-			-	-	-	-
0	1700	97800	112000	HomeImp	Office	3	0	0	93.33333...

Visual Interfaces

Prepare Data

The screenshot shows the SAS Data Studio interface for 'Prepare Data'. On the left, a sidebar lists various transform types: Transforms, Add Transforms, Split, Trim whitespace, Custom Transforms (Calculated column, Code selected), Data Quality Transforms, Casing, Field extraction, Gender analysis, Identification analysis, Match and cluster, Matchcodes, Parsing, and Standardize.

The main area displays a code editor titled 'Code - Step 1 of 1' with the following content:

```
1 /* Create a copy of the input table to the output table. */
2 /* This statement should be replaced by the actual code you intend to run. */
3 table.partition
4 table=
```

A dropdown menu above the code editor is set to 'CASL'. A green box highlights this dropdown and the code editor area.

Below the code editor is a data preview for the 'HMEQ' dataset. The preview shows three rows of data with the following columns:

BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ
1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.3666...	1
1	1300	20053	48400	HomeImp	Other	7	0	2	121.833...	0
1	1500	13500	16700	HomeImp	Other	4	0	0	149.466...	1

At the bottom right of the preview, it says 'Result rows: 100'.

Visual Interfaces

Build Models - Pipelines

Model Studio - Build Models

Demo VDDML

Nodes

- Data Mining Preprocessing
 - Anomaly Detection
 - Clustering
 - Feature Extraction
 - Filtering
 - Imputation
 - Manage Variables
 - Replacement
- Text Mining
- Transformations
- Variable Clustering
- Variable Selection
- Supervised Learning
- Postprocessing
- Miscellaneous
 - Data Exploration
 - Open Source Code
 - SAS Code
 - Save Data

Pipelines

Pipeline Comparison

Insights

Interactive-Model Pipeline Open Source Example Advanced with Autotuning Batch EM Code Example

Run Pipeline

```
graph TD; Data[Data] --> Imputation[Imputation]; Imputation --> VariableSel[Variable Selection]; VariableSel --> StepwiseL[Stepwise Logistic]; VariableSel --> ForwardL[Forward Logistic]; VariableSel --> DecisionT[Decision Tree]; VariableSel --> Forest[Forest]; StepwiseL --> ModelComp[Model Comparison]; ForwardL --> ModelComp; DecisionT --> ModelComp; Forest --> ModelComp;
```

The diagram illustrates a data mining pipeline. It starts with a 'Data' node at the top, which branches down to an 'Imputation' node. From 'Imputation', the flow splits into four parallel paths: 'Variable Selection', 'Stepwise Logistic', 'Forward Logistic', and 'Decision Tree'. Finally, all four paths converge at a 'Model Comparison' node at the bottom.

Visual Interfaces

Build Models – Pipelines – Imputation Node

The screenshot shows the configuration interface for the 'Imputation' node in the SAS Visual Data Pipeline. The interface is divided into several sections:

- Left Panel (Imputation Settings):**
 - Description:** "Imputes missing values for class and interval inputs using the specified methods."
 - Impute non-missing variables**
 - Missing percentage cutoff:** 50
 - Reject original variables**
 - Summary statistics**
 - Class Inputs:** Default method is set to **Count**. Other options include (none), Cluster count, Constant value, and Distribution.
- Center Panel (Interval Inputs):**
 - Default method:** Set to **Mean**.
 - Other options in the dropdown menu:
 - (none)
 - Cluster mean
 - Constant value
 - Maximum
 - Mean
 - Median
 - Midrange
 - Minimum
 - Single indicator** (disabled)
- Right Panel (Constant Values):**
 - Constant character value:** (empty input field)
 - Constant number value:** 0
 - Indicators:**
 - Single indicator**
 - Unique indicators**
 - Indicator subject:** Imputed variables
 - Indicator role:** Rejected

Visual Interfaces

Build Models – Pipelines – Replacement Node

The screenshot shows the SAS Visual Data Mining and Machine Learning interface. A node titled "Replacement" is selected. The "Description" panel states: "Replaces data values such as outliers and unknown class levels with specified values." The "Replacement value for unknown class levels:" dropdown is set to "Missing value". Under "Interval Inputs", the "Default limits method:" dropdown is set to "Standard deviation from the m...". The "Standard deviations:" input field contains the value "3". The "Replacement value:" dropdown is also set to "Missing value".

The **Replacement** node is a Data Mining Preprocessing node. It is used to generate score code to replace outliers and unknown class levels with specified values. In some cases, you might want to reassign specified nonmissing values (trim your variable's distribution) before performing imputation calculations for the missing values. This is a typical task for the **Replacement** node.

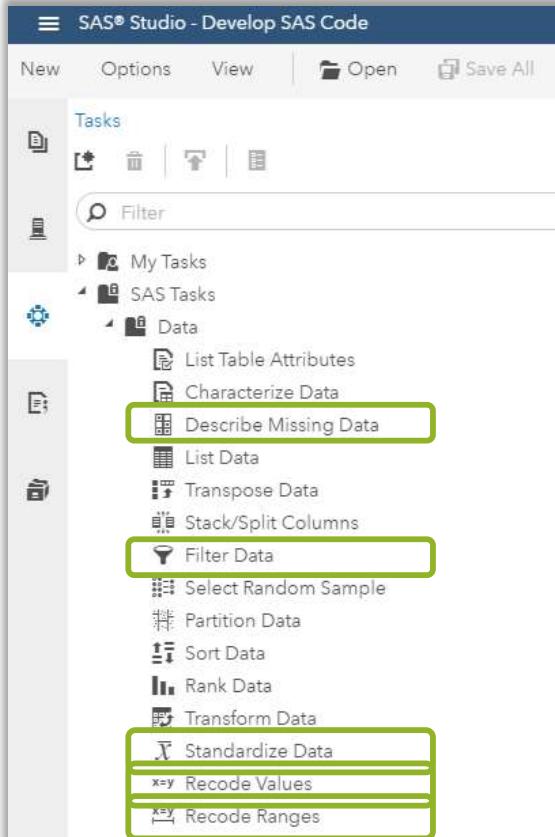


Programming Interfaces

SAS Studio and Open Source

Programming Interfaces

SAS Studio – Develop SAS Code



Same options as
described for SAS 9

PLUS

Programming Interfaces

SAS Studio – Develop SAS Code

SAS Viya Prepare and Explore

- Summary
- Transform Data
- Variable Selection
- Sampling
- Partitioning
- Binning
- Imputation

- Replace with
- mean
 - median
 - random number
 - mode

DATA
PUBLIC.HMEQ
Filter: (none)

ROLES

Interval Variables

Replace missing values with the mean:

- # LOAN
- # MORTDUE

Replace missing values with the median:

- # VALUE
- # DEROG
- # DEBTINC

Replace missing values with a random nu...

- # CLAGE
- # NINQ

Random number seed

Nominal Variables

Replace missing levels with the mode:

- # YOJ

Programming Interfaces

SAS Studio – Develop SAS Code

DATA **OUTPUT** INFORMATION

▼ OUTPUT DATA

The following table must use a CAS engine libref:

- Save imputed data

Specify a CAS table: *

Overwrite data

PUBLIC.HMEQ2



Include variables from the input CAS table:

- All variables
- Variables used in the analysis
- No variables
- Selected variables

```
proc varimpute data=PUBLIC.HMEQ;  
  input LOAN MORTDUE / ctech=mean;  
  input VALUE DEROG DEBTINC / ctech=median;  
  input CLAGE NINQ / ctech=random;  
  input YOJ / ntech=mode;  
  output out=PUBLIC.HMEQ2;  
run;
```

Programming Interfaces

SAS Studio – Develop SAS Code

Imputation Method	Number of Variables	Seed
Mean	2	
Random	2	1879166545
Median	3	
Mode	1	

Imputed Values for Interval Variables						
Variable	Label	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
LOAN	Amount of current loan request	Mean	IM_LOAN	5960	0	18608
MORTDUE	Amount due on existing mortgage	Mean	IM_MORTDUE	5442	518	73760.8
VALUE	Value of current property	Median	IM_VALUE	5848	112	89231
DEROG	No. of major derogatory reports	Median	IM_DEROG	5252	708	0
DEBTINC	Debt to income ratio	Median	IM_DEBTINC	4693	1267	34.8183
CLAGE	Age of oldest credit line in months	Random	IM_CLAGE	5652	308	
NINQ	No. of recent credit inquiries	Random	IM_NINQ	5450	510	

Imputed Values for Nominal Variables						
Variable	Label	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
YOJ	Years on current job	Mode	IM_YOJ	5445	515	0

Programming Interfaces

Develop Code using CAS Actions

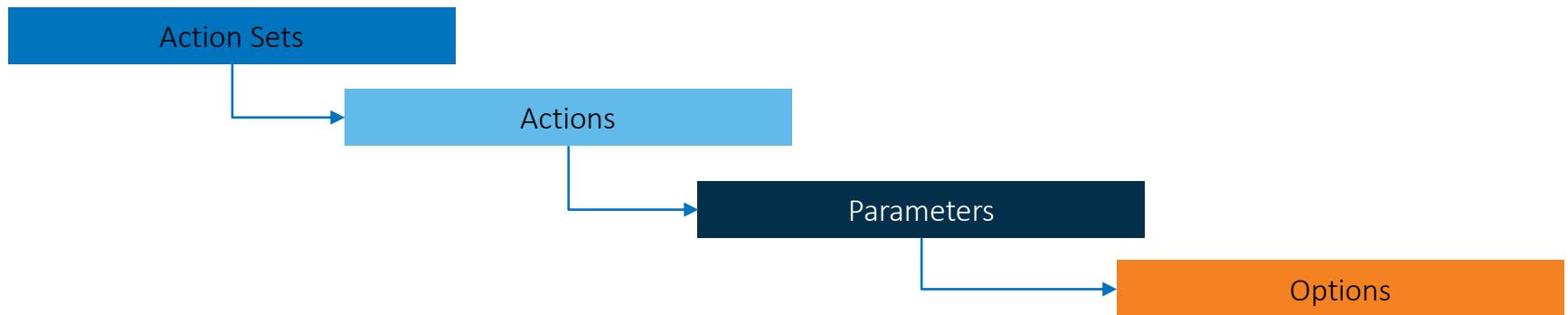
- CAS – Cloud Analytic Server
- CAS actions are the tools used to interact with data on the CAS server.
- CAS actions are wrappers for parallel processing algorithms.
- CAS actions can load data, transform data, compute statistics, perform analytics, and create output.

Python Functions ≡ SAS 9.4 Procedures ≡ CAS Actions

CASL – Cloud Analytic Server Language

Programming Interfaces

CAS Actions Hierarchies



```
table.attribute <result =results> <status=rc> /  
  attributes={{  
    column="string",  
    * key="string",  
    value="string" | 64-bit-integer | integer | double | binary-large-object  
  }, {...}}
```

Programming Interfaces

Develop Code using CAS Actions

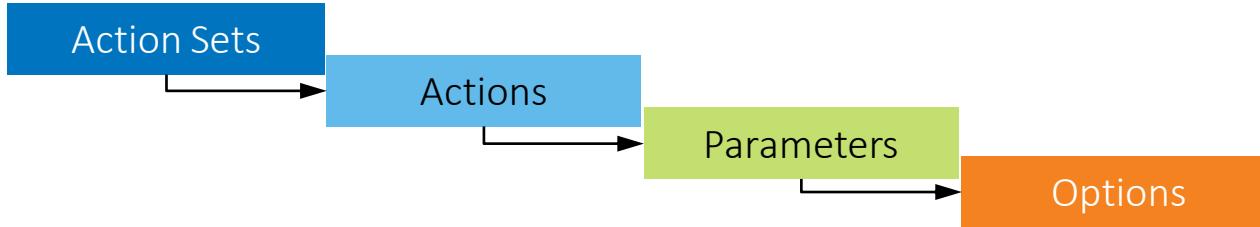
DataPreprocess Action Set

- Actions
 - impute
 - transform

Programming Interfaces

CAS Actions Hierarchies

- The functionality mimics the look and feel of Python syntax, making it easy for Python users to take advantage of CAS.



```
sas.datapreprocess.impute(  
    table = dict(),  
    inputs = value_list,  
    methodContinuous = "median",  
    methodNominal="mode",  
    casOut = dict()  
    replace=TRUE  
)
```

Programming Interfaces

Develop Code using CAS Actions

```
PROC CAS;
```

```
datapreprocess.impute /  
  table={name="carsInfo"}  
  inputs={"msrp", "invoice", "make"}  
  methodContinuous="median"  
  methodNominal="mode"  
  casout={name = "outImpute" replace=True};  
run;
```

The SAS System						
Results from dataPreprocess.impute						
Imputation Information for CARSINFO						
Variable	Imputation Method	Result Variable	N	N Miss	Continuous Imputed Value	Nominal Imputed Value
MSRP	Median	IMP_MSRP	428	0	27635	
Invoice	Median	IMP_Invoice	428	0	25294.5	
Make	Mode	IMP_Make	428	0	.	Toyota

methodContinuous="MAX" | "MEAN" | "MEDIAN" | "MIDRANGE" | "MIN" | "MODE" | "RANDOM" | "VALUE"

methodNominal="MAX" | "MEAN" | "MEDIAN" | "MIDRANGE" | "MIN" | "MODE" | "RANDOM" | "VALUE"

Programming Interfaces

Develop Code using CAS Actions

Jupyter Notebook

```
In [18]: r=sas.dataPreprocess.transform(  
    table=hmeq,  
    casOut={"name":"hmeq_prep", "replace":True},  
    copyAllVars=True,  
    outVarsNameGlobalPrefix="IM",  
    requestPackages=[  
        {"impute":{"method":"MEAN"}, "inputs":{"clage"}},  
        {"impute":{"method":"MEDIAN"}, "inputs":{"delinq"}},  
        {"impute":{"method":"VALUE", "valuesInterval":{2}}, "inputs":{"ninq"}},  
        {"impute":{"method":"VALUE", "valuesInterval":{35.0, 7, 2}}, "inputs":{"debtinc", "yoj"}}  
    ]  
)  
render_html(r)
```

Variable Transformation Information for HMEQ

Variable	Transformation Name	Result Variable	Number of Observations	Number of Missing	Imputed Value
CLAGE	IM	IM_CLAGE	5652	308	179.77
DEBTINC	IM	IM_DEBTINC	4693	1267	35.0000
DELINQ	IM	IM_DELINQ	5380	580	0
NINQ	IM	IM_NINQ	5450	510	2.0000
YOJ	IM	IM_YOJ	5445	515	2.0000

[Using SWAT
available
from GitHub](#)



Resources

Where to learn more

Where to learn more?

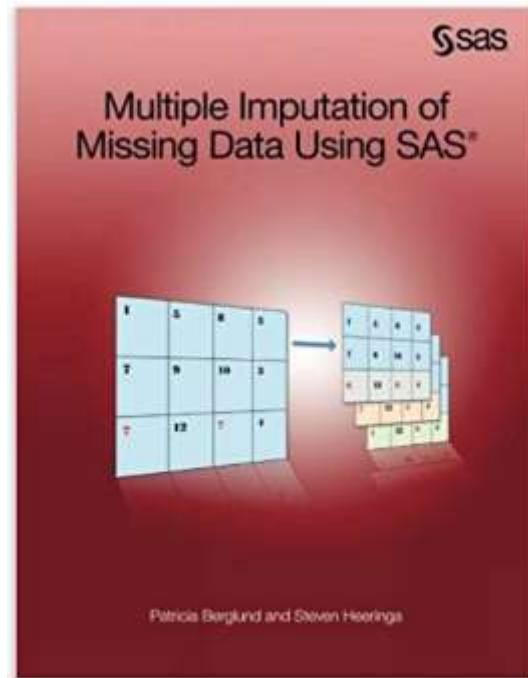
SAS Documentation

- [Working with Missing Data in SAS](#)
- [Proc HPIMPUTE Documentation](#)
- [SAS Enterprise Miner Impute Missing Values](#)
- [Proc MI Documentation](#)
- [Proc MIANALYZE Documentation](#)

Where to learn more?

Book

[Multiple Imputation of Missing Data Using SAS](#)



Where to learn more?

Videos

- [Getting Started with SAS Enterprise Miner: Exploring Input Data and Replacing Missing Values](#)
- [SAS Enterprise Miner Tip: Imputing Missing Values](#)
- [Handling Missing Values in Survey Data](#)
- [SAS Missing Data](#)
- [Missing Values in SAS Data Step](#)

Where to learn more?

Papers

- Managing Missing Data Using SAS® Enterprise Guide®
<http://support.sas.com/resources/papers/proceedings14/SAS257-2014.pdf>
- Hot-Deck Imputation: A Simple DATA Step Approach
<https://analytics.ncsu.edu/sesug/1999/075.pdf>
- Imputing Dose Levels for Adverse Events
<https://www.lexjansen.com/pharmasug/2013/HO/PharmaSUG-2013-HO03.pdf>
- Identifying and Overcoming Common Data Mining Mistakes
<http://www2.sas.com/proceedings/forum2007/073-2007.pdf>
- A SAS® Macro for Single Imputation
<https://www.lexjansen.com/pharmasug/2008/sp/SP10.pdf>

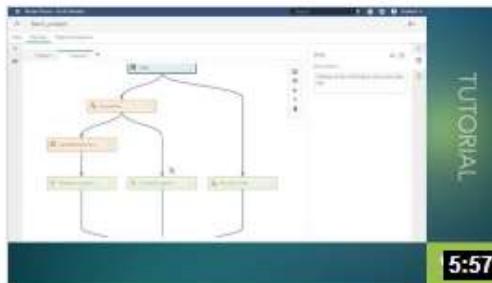
SAS® Viya Resources

Videos

- 6 minutes getting started [video](#)
- 5 minutes getting started [video](#)
- 8 minute demo [video](#)
- Feature Engineering [video](#)



Using the Automated Analysis Feature in
SAS® Visual Analytics in SAS® Viya®



Getting Started with Data Mining and
Machine Learning Pipelines on SAS®
Viya®



Building and Using Pipelines in SAS®
Visual Forecasting

SAS® Viya Resources

SAS Visual Statistics User's Guide

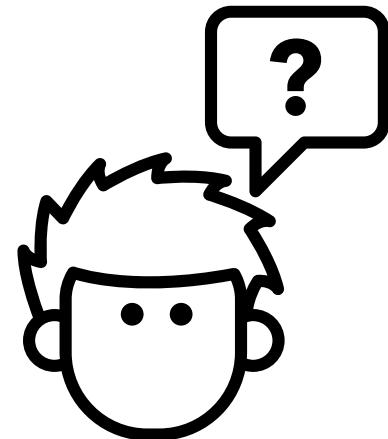
<http://support.sas.com/software/products/visual-statistics/index.html#s1=2>

SAS Visual Data Mining and Machine Learning User's Guide

<http://support.sas.com/software/products/visual-data-mining-machine-learning/index.html#s1=1>

Overview, Training, Samples and Tips

- [SAS Viya Overview](#)
- [SAS Viya Training](#)
- [A Beginner's Guide to Programming in the SAS® Cloud Analytics Services Environment](#)



Resources

Programming

- [SAS Studio](#)
- [CAS actions documentation](#)
- [SAS Github page for SWAT-Python](#)
- [SAS Github page for SWAT-R](#)
- [More example scripts for using SWAT-R & SWAT-Python](#)



Where to learn more?

Training

- <https://www.statistics.com/missing-data/>
- https://uisug.org.uiowa.edu/sites/uisug.org.uiowa.edu/files/wysiwyg_uploads/Handling%20Missing%20Values%20with%20SAS.pdf
- <https://statisticalhorizons.com/sensitivity-analysis>
- https://stats.idre.ucla.edu/wp-content/uploads/2017/01/Missing-Data-Techniques_UCLA.pdf



Questions?

Thank you for your time and attention!

Connect with me:

LinkedIn: <https://www.linkedin.com/in/melodierush>

Twitter: @Melodie_Rush

sas.com