



# Handling Missing Values in SAS

SAS 9 and 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

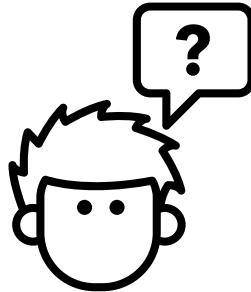
Products



# 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.

# 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?



Remove observation(s)



Replace with Constant or Zero



Replace with mean or mode



Replace using an imputation  
method



# 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=sum(x,0);`
- `y=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;
```

- Replace all numeric missing values with the mean

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

**reponly** – only replace; do not standardize

**missing** – can be any constant

**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

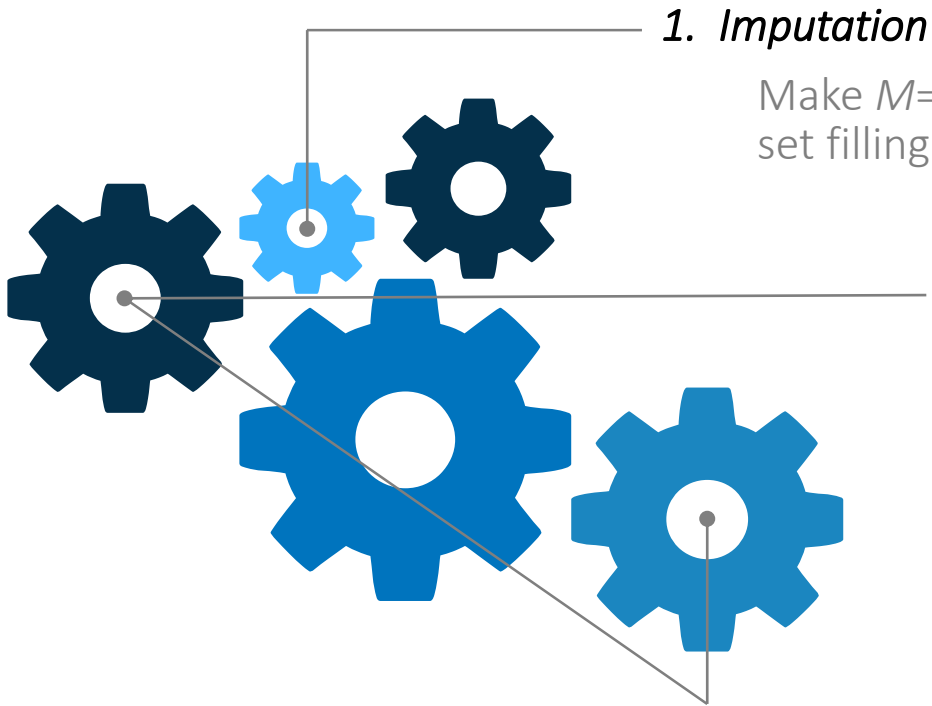


# Multiple Imputation

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



```
PROC REG data=outmi outest=outreg covout noprint;  
    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;
```

```
modeffects 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 (FHDI), 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;
```

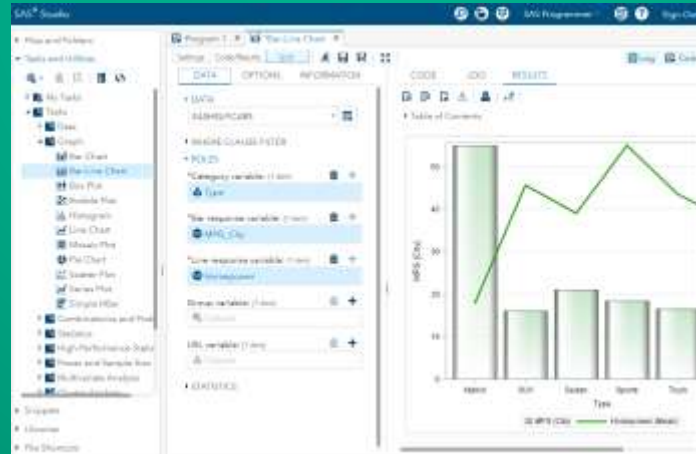
[PROC SURVEYIMPUTE Documentation](#)





# 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		
BAD	Frequency	Percent
Non-missing	5960	100.00

Amount of current loan request		
LOAN	Frequency	Percent
Non-missing	5960	100.00

Amount due on existing mortgage		
MORTDUE	Frequency	Percent
.	518	8.69
Non-missing	5442	91.31

Home improvement or Debt Consolidation		
REASON	Frequency	Percent
.	252	4.23
Non-missing	5708	95.77

Prof/Exec/Office/Self/Other		
JOB	Frequency	Percent
.	279	4.68
Non-missing	5681	95.32

Years on current job		
YOJ	Frequency	Percent
.	515	8.64
Non-missing	5445	91.36

# 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	0.0336
Non-missing	Non-missing	.	.	.	.	.	.	.	.	.	.	Non-missing	8	0.1007
Non-missing	Non-missing	.	.	.	.	Non-missing	Non-missing	Non-missing	.	Non-missing	Non-missing	.	1	0.0188
Non-missing	Non-missing	.	.	.	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	1	0.0188
Non-missing	Non-missing	.	.	Non-missing	.	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	.	2	0.0336
Non-missing	Non-missing	.	.	Non-missing	.	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	Non-missing	1	0.0188
Non-missing	Non-missing	.	.	Non-missing	Non-missing	.	.	.	.	.	.	.	2	0.0336
Non-missing	Non-missing	.	.	Non-missing	Non-missing	Non-missing	.	.	.	.	.	.	2	0.0336

# Standardize Data Task

## Select Missing Values Method & How to replace missing values

Tasks

Filter

- 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.sas Standardize Data.ctl

Submit Cancel History Add

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

**Default location measure**

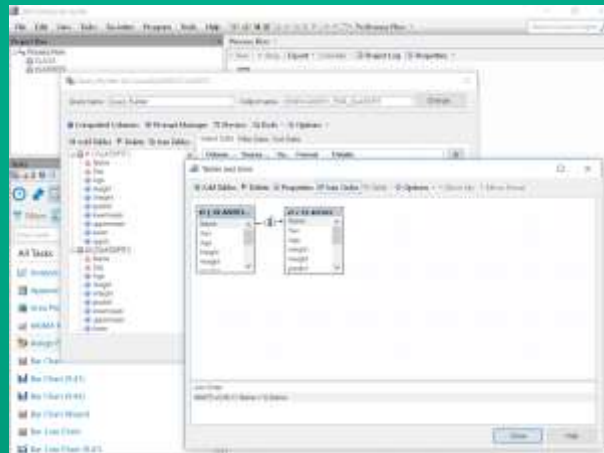
Mean

Median

Minimum

Specify custom value

# SAS Enterprise Guide





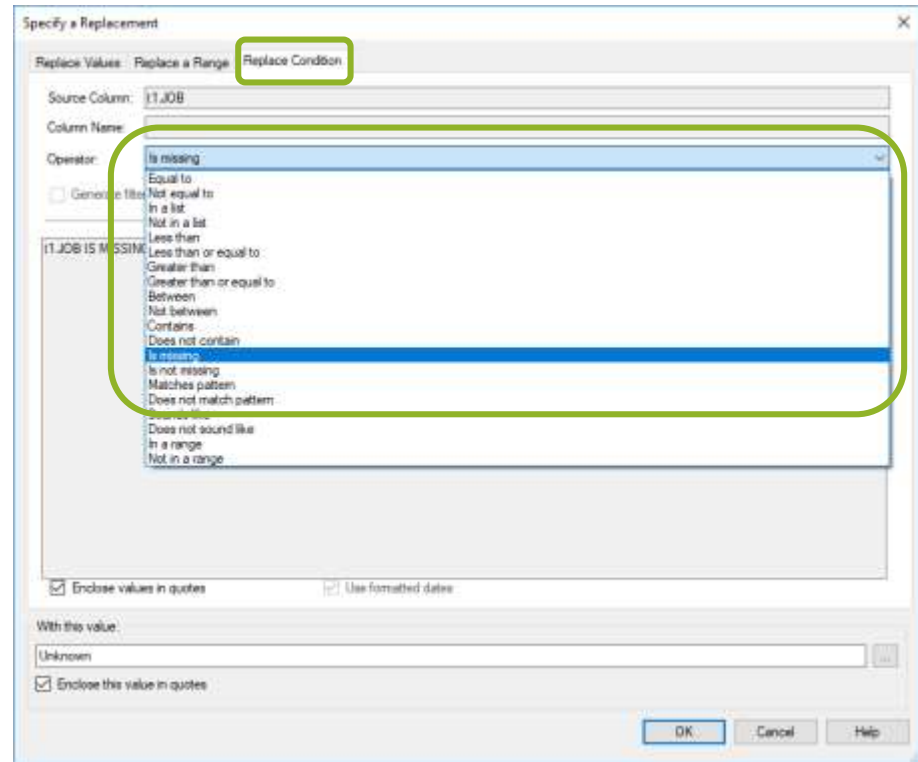
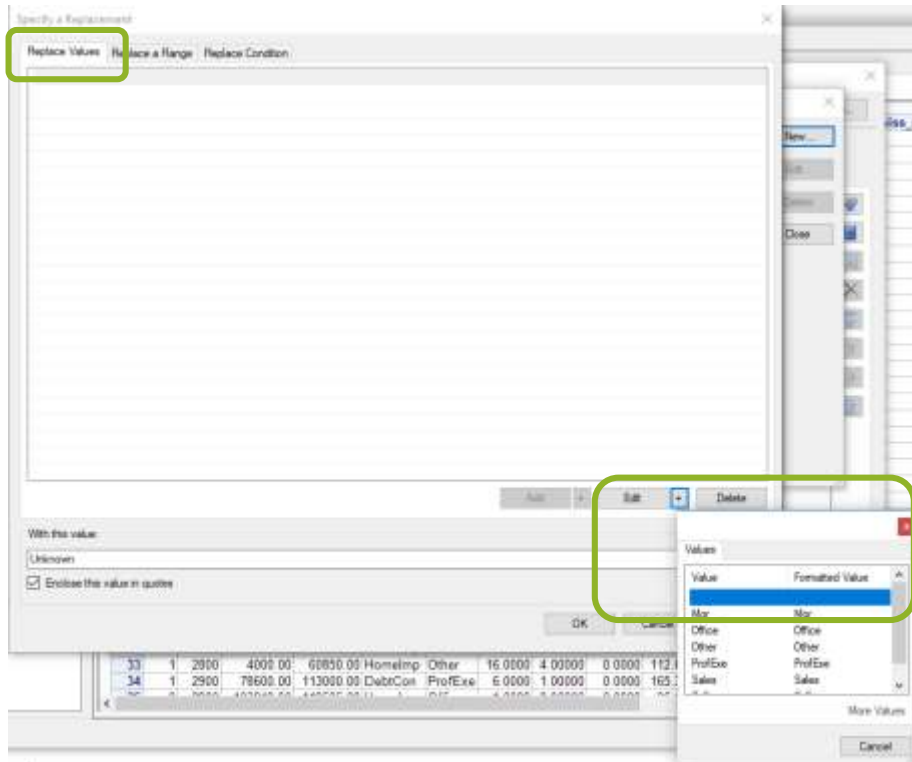
## 3 Ways

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

Specify a Replacement

Replace Values | Replace a Range | Replace Condition

With this value:  
0

Enclose this value in quotes

Values

Value	Formatted Value
2063	2063
2838	2838
2800	2800
3372	3372
4000	4000

OK Cancel

Specify a Replacement

Replace Values | Replace a Range | Replace Condition

Source Column: I1.MORTDUE

Column Name:

Operator: Is missing

Generate filter

I1.MORTDUE IS

Enclose values in quotes

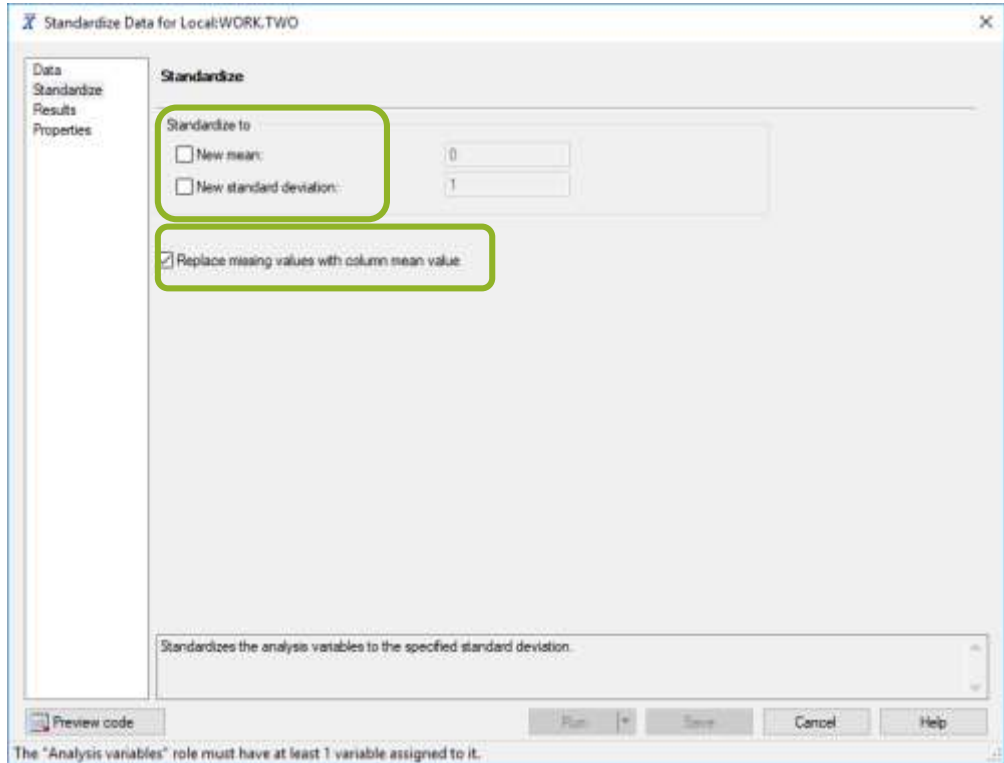
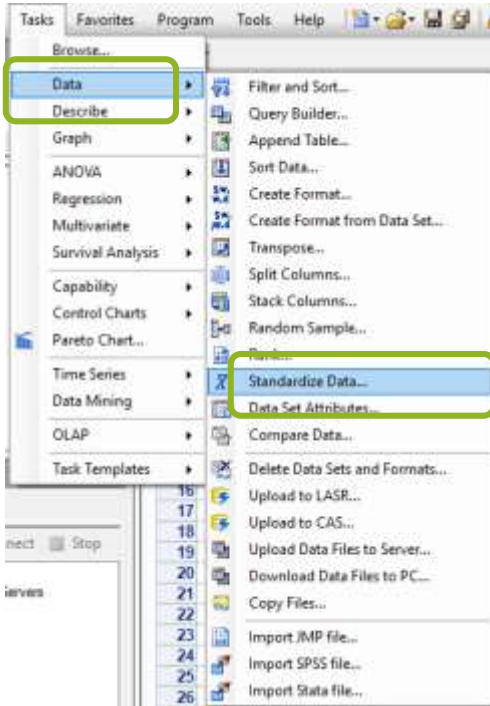
With this value:  
0

Enclose this value in quotes

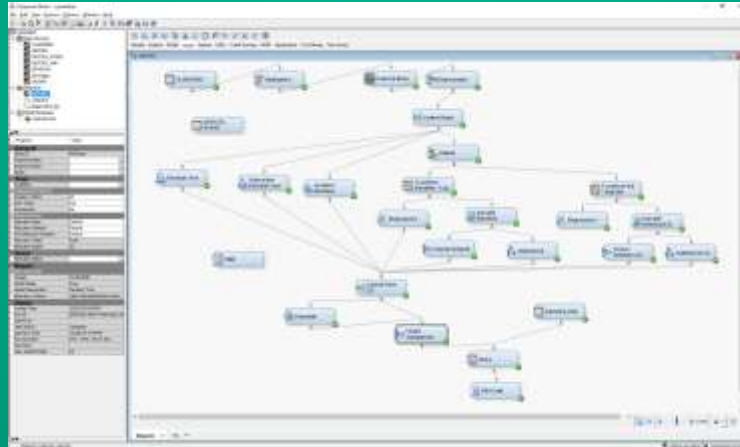
OK Cancel Help

# 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

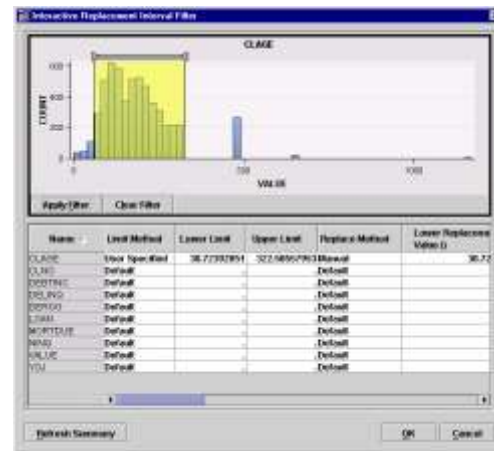
- PROC STDIZE - [documentation](#)
- PROC STANDARD - [documentation](#)
- PROC HPIMPUTE - [documentation](#)
- SAS/STAT PROC MI - [documentation](#)

# 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

A dialog box titled "Replacement Editor" showing a table of replacement rules for various variables. The table has columns for Variable, Level, Frequency, Type, Char Raw value, Num Raw Value, and Replacement Value. At the bottom, there are "Reset", "OK", and "Cancel" buttons.

Variable	Level	Frequency	Type	Char Raw value	Num Raw Value	Replacement Value
BAD	0	4771	N		0.0	
BAD	1	1189	N		1.0	
BAD	_UNKNOWN_		N			DEFAULT_
JOB	Other	2388	C	Other		
JOB	ProfExe	1276	C	ProfExe		
JOB	Office	848	C	Office		
JOB	Mgr	787	C	Mgr		
JOB		279	C			
JOB	Self	193	C	Self		
JOB	Sales	109	C	Sales		
JOB	_UNKNOWN_		C			DEFAULT_
REASON	DebtCon	3928	C	DebtCon		
REASON	HomeImp	1780	C	HomeImp		
REASON		252	C			
REASON	_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

Train	
Variables	
Non Missing Variables	No
Missing Cutoff	50.0
Class Variables	Count
Default Input Method	Default Constant Value
Default Target Method	Distribution
Normalize Values	Tree
Interval Variables	Mean
Default Input Method	Maximum
Default Target Method	Minimum
Default Constant Value	Median
Default Character Value	Midrange
Default Number Value	Distribution
Method Options	Tree
Random Seed	Tree Surrogate
Tuning Parameters	
Tree Imputation	
Score	
Hide Original Variables	Yes
Indicator Variables	
Type	None
Source	Imputed Variables
Role	Rejected

> Class Variables

> Interval Variables



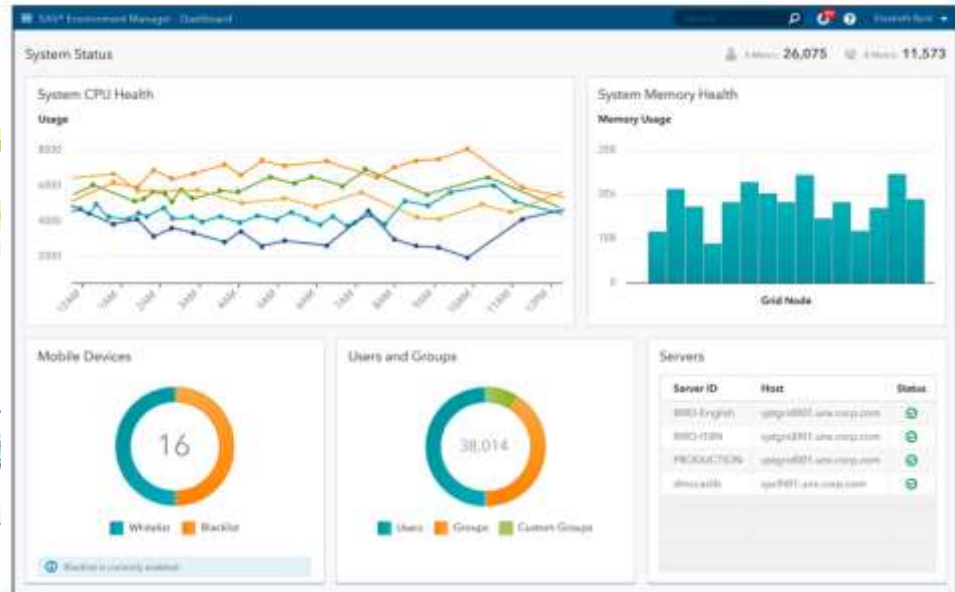
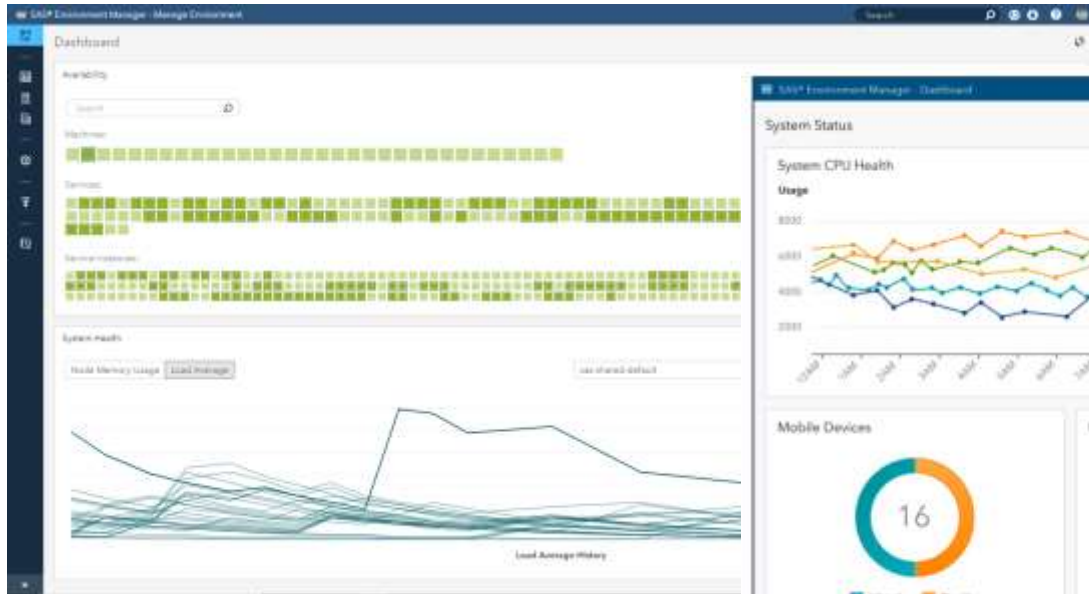
# SAS Viya

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 is an underlying foundation for additional products that will take 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 Mining
  - SAS Optimization
  - SAS Econometrics
  - SAS Visual Investigator



## MULTIPLE INTERFACES, SINGLE CODE BASE

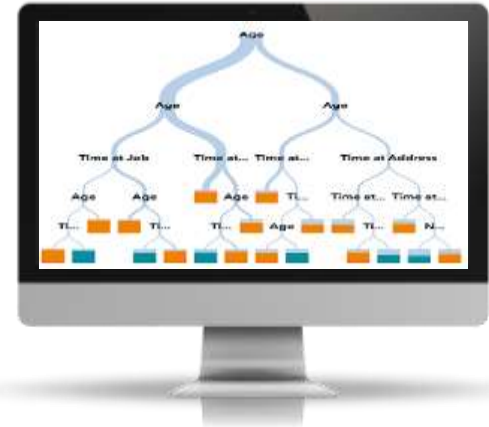
Visual Interfaces



Programming Interfaces



API Interfaces

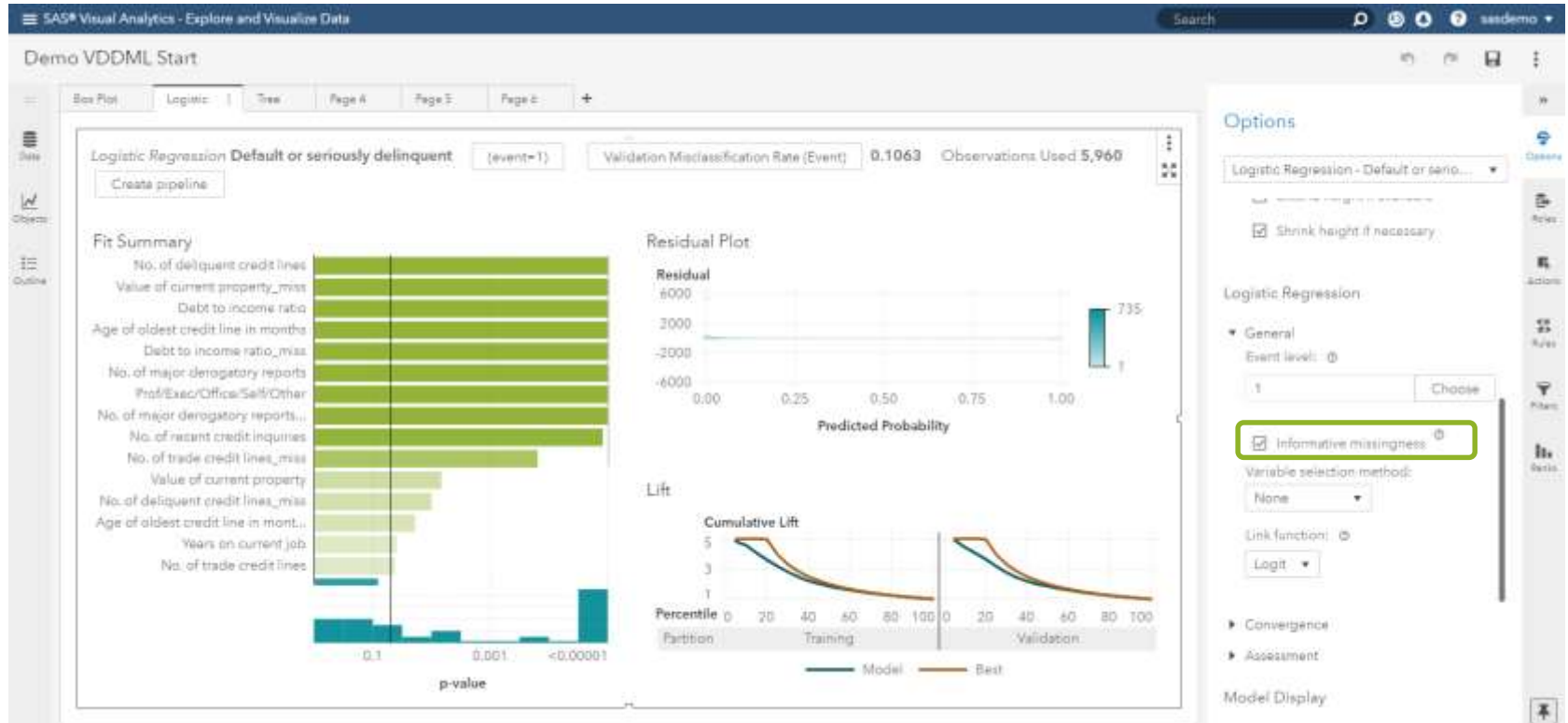




# Visual Interfaces

# Visual Interfaces

## Explore and Visualize Data



# Visual Interfaces

## Explore and Visualize Data

The screenshot displays the SAS Visual Analytics interface for configuring a Logistic Regression model. The main workspace shows a box plot for the 'Log' variable with a mean of 735 and a range from 1 to 735. Below it is a histogram of p-values with categories 0.1, 0.001, and <0.00001. The 'Options' panel on the right is titled 'Logistic Regression - Default or perso...' and includes a 'General' section with 'Event level: 0' and a 'Choose' button. A green arrow points to the 'Informative missingness' checkbox, which is checked. Below this, the 'Variable selection method' is set to 'None' and the 'Link function' is 'Logit'. The 'Model Display' section is partially visible at the bottom.

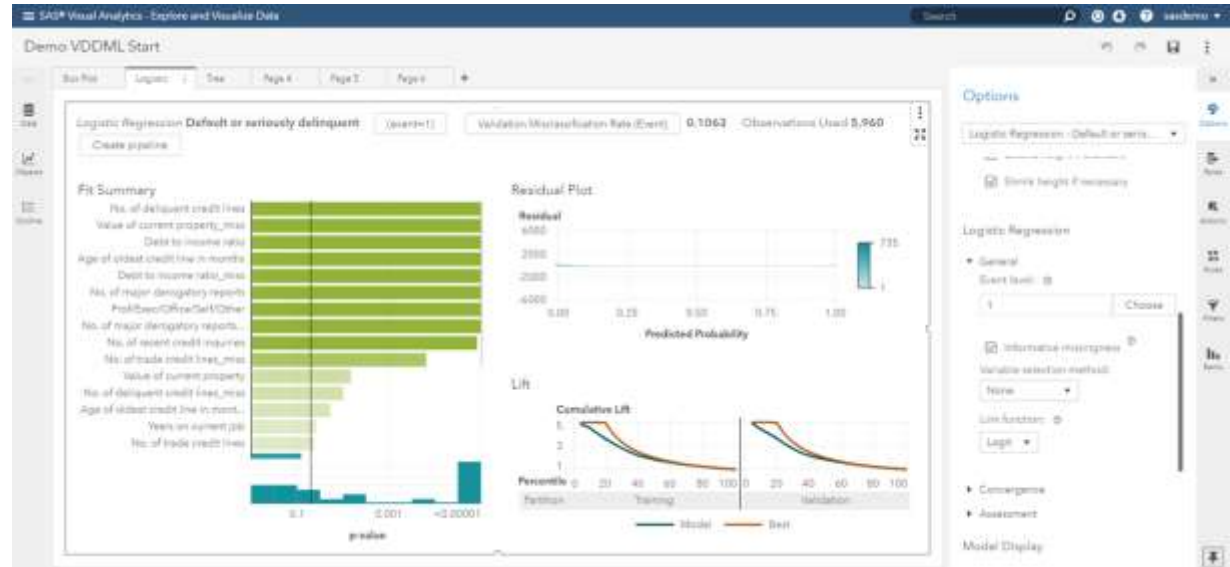
Extends the model to include observations with missing values. A continuous effect is imputed with the observed mean, and an indicator variable that denotes missingness is created. A classification effect treats missing values as a distinct level.

# 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

Forrest GB Co

Data

HMEQ

Filter

+ New data item

Category

- Default or seriously delinqu... - 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

Apply data source filter...

Available

- Filter out missing values
- Replace with constant

SAS® Visual Analytics - Explore and Visualize Data

Demo VDDML

Data

HMEQ

Filter

+ New data item

Hierarchy...

Custom category...

Calculated item...

Geography item...

Parameter...

Interaction effect...

Spline effect...

Partition...



# Visual Interfaces

## Prepare Data

### Use Code or Calculated Column

- Replace with Constant or Zero
- Code for imputation

The screenshot shows the SAS Data Studio 'Prepare Data' interface. On the left, the 'Transforms' panel is visible, with 'Code' selected under 'Custom Transforms'. The main workspace displays a 'Code - Step 1 of 1' step with the following SAS code:

```
1 /* Write data step with the output table data */
2 data [_<_outputTable_] (<<SAS-[_<_outputTable_]>> _name_='W');
3 /* Set the input set */
4 set [_<_inputTable_] (<<SAS-[_<_inputTable_]>> );
5 IF N(DEL) = . THEN N(DEL) = 0;
6 /* END data step run */
7 run;
```

Below the code editor, the 'HMEQ (session)' table is displayed with columns: @ BAD, @ LOAN, @ MORTDUE, @ VALUE, @ SEASON, @ JOB, @ Y0J, @ DEROG, @ DELNG, @ CLAGE. The table shows 4 rows of data.

@ BAD	@ LOAN	@ MORTDUE	@ VALUE	@ SEASON	@ JOB	@ Y0J	@ DEROG	@ DELNG	@ CLAGE
1	1000	2580	2825	Homebnp	Other	10.5	0	0	91.3666...
1	1800	7000	4400	Homebnp	Other	7	0	2	52.1433...
1	1500	1300	1670	Homebnp	Other	4	0	0	74.9066...
1	1900	0							
2	1100	7700	11200	Homebnp	Office	2	0	0	93.3333...

# Visual Interfaces

## Prepare Data

The screenshot displays the SAS Data Studio interface for data preparation. On the left, a sidebar lists various transform categories: Column Transforms, Custom Transforms, and Data Quality Transforms. The 'Code' option under Custom Transforms is selected. The main workspace shows a code editor with a SAS DATA step script. A green box highlights the 'DATA step' dropdown menu in the toolbar above the code editor. Below the code editor, a data table titled 'HMEQ (session)' is displayed, showing columns for various financial and demographic variables. The table has 10 columns and 5 rows of data. The 'Result rows' control is set to 100.

```
Code - Step 1 of 1
```

Code

DATA step

```
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;
```

HMEQ (session) Table Profile Metadata

Result rows: 100

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 displays the SAS Data Studio 'Prepare Data' interface. On the left, a 'Transforms' sidebar lists various data manipulation options, with 'Code' selected. The main workspace is titled 'Plan 1' and shows a 'Code - Step 1 of 1' editor. A green box highlights the 'CASL' dropdown menu in the code editor's toolbar. The code editor contains the following SAS code:

```
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=
```

Below the code editor, the 'HMEQ' table is displayed. The interface includes tabs for 'Table', 'Profile', and 'Metadata'. The 'Table' tab is active, showing a data table with columns: BAD, LOAN, MORTDUE, VALUE, REASON, JOB, YOJ, DEROG, DELINQ, CLAGE, NINQ. The 'Result rows' are set to 100. The table contains three rows of data:

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	70053	68400	HomeImp	Other	7	0	2	121.833...	0
1	1500	13500	16700	HomeImp	Other	4	0	0	149.466...	1

# Visual Interfaces

## Build Models - Pipelines

The screenshot displays the SAS Model Studio - Build Models interface. On the left, a sidebar titled "Nodes" lists various data mining tasks. The "Data Mining Preprocessing" category is expanded, showing tasks like Imputation, Variable Selection, and Replacement, which are highlighted with green boxes. Other categories include Supervised Learning, Postprocessing, and Miscellaneous. The main workspace shows a pipeline diagram starting with a "Data" node, which branches into "Imputation", "Variable Selection", "Stepwise Logistic Regression", "Forward Logistic Regression", "Decision Tree", and "Forest". All these nodes converge into a final "Model Comparison" node. A "Run Pipeline" button is visible in the top right corner of the workspace.

```
graph TD; Data[Data] --> Imputation[Imputation]; Data --> VS[Variable Selection]; Data --> SLR[Stepwise Logistic Regression]; Data --> FLR[Forward Logistic Regression]; Data --> DT[Decision Tree]; Data --> Forest[Forest]; Imputation --> SLR; Imputation --> FLR; VS --> SLR; VS --> FLR; SLR --> MC[Model Comparison]; FLR --> MC; DT --> MC; Forest --> MC;
```

# Visual Interfaces

## Build Models – Pipelines – Imputation Node

>>

### Imputation

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:  
Count

Count  
Distribution

▼ Interval Inputs

Default method:  
Mean

(none)  
Cluster mean  
Constant value  
Maximum  
Mean  
Median  
Midrange  
Minimum

Single indicator

▼ Constant Values

Constant character value:

Constant number value:  
0

▼ Indicators

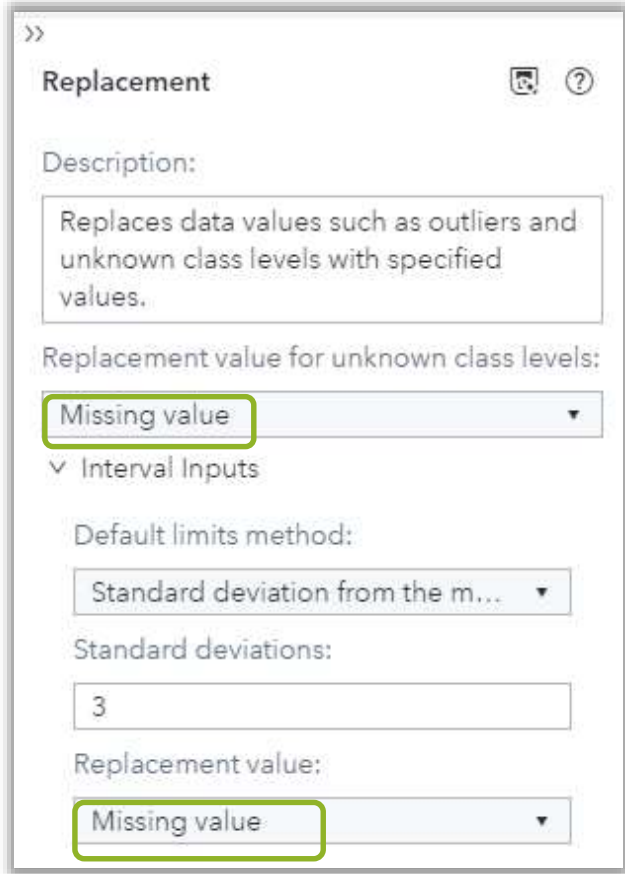
Single indicator  
 Unique indicators

Indicator subject:  
Imputed variables

Indicator role:  
Rejected

# Visual Interfaces

## Build Models – Pipelines – Replacement Node



The screenshot shows the configuration window for the Replacement node. The title bar reads 'Replacement' with a help icon and a close icon. Below the title bar, there is a 'Description:' section with a text box containing the text: 'Replaces data values such as outliers and unknown class levels with specified values.' Below this is a dropdown menu for 'Replacement value for unknown class levels:' with 'Missing value' selected. Underneath is a collapsed section for 'Interval Inputs'. Below that is a 'Default limits method:' dropdown menu with 'Standard deviation from the m...' selected. Below this is a text box for 'Standard deviations:' containing the number '3'. Finally, there is a 'Replacement value:' dropdown menu with 'Missing value' selected. The 'Missing value' options in both dropdowns are highlighted with a green border.

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.

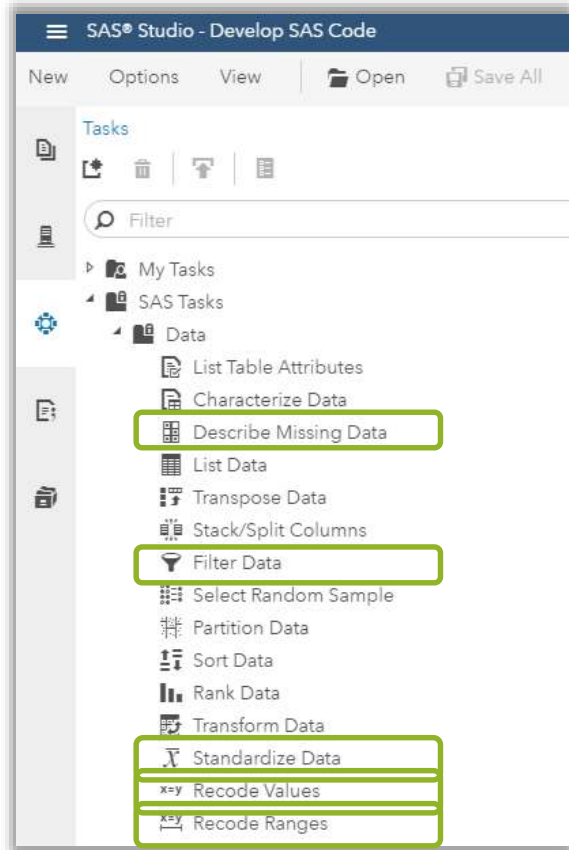


# Programming Interfaces

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: ↑ ↓ 🗑️ +

- YOU

# 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  $\equiv$  SAS 9.4 Procedures  $\equiv$  CAS Actions

CASL – Cloud Analytic Server Language

# Programming Interfaces

## Develop Code using CAS Actions

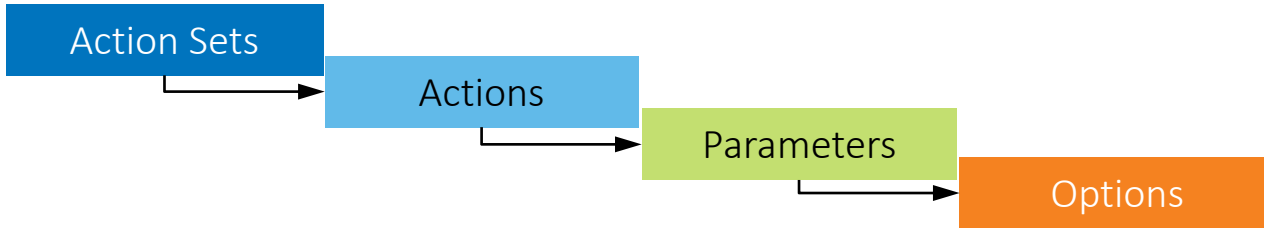
### DataPreprocess Action Set

- 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_prepped", "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?

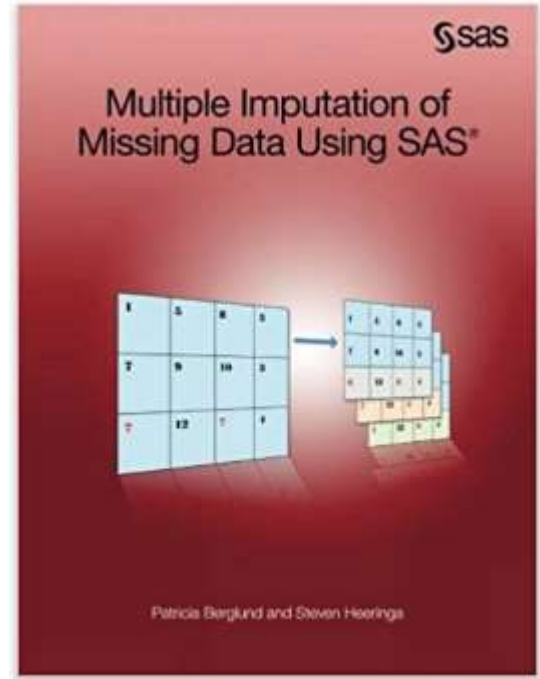
## SAS Documentation

- [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>

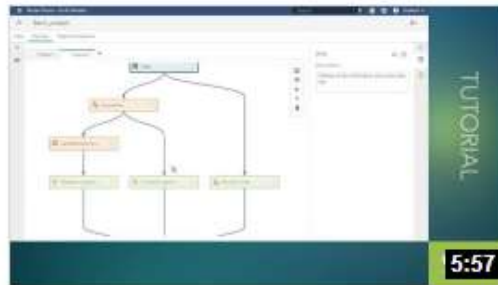
# SAS® Viya Resources

## 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

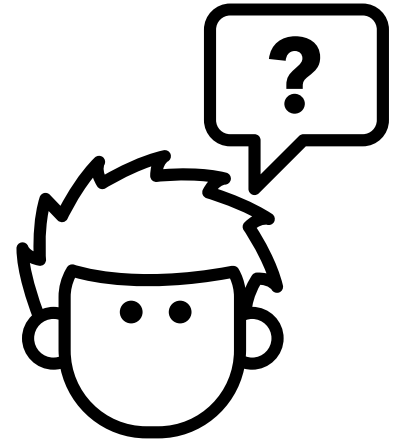
<http://support.sas.com/software/products/visual-statistics/index.html#s1=2>

## 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

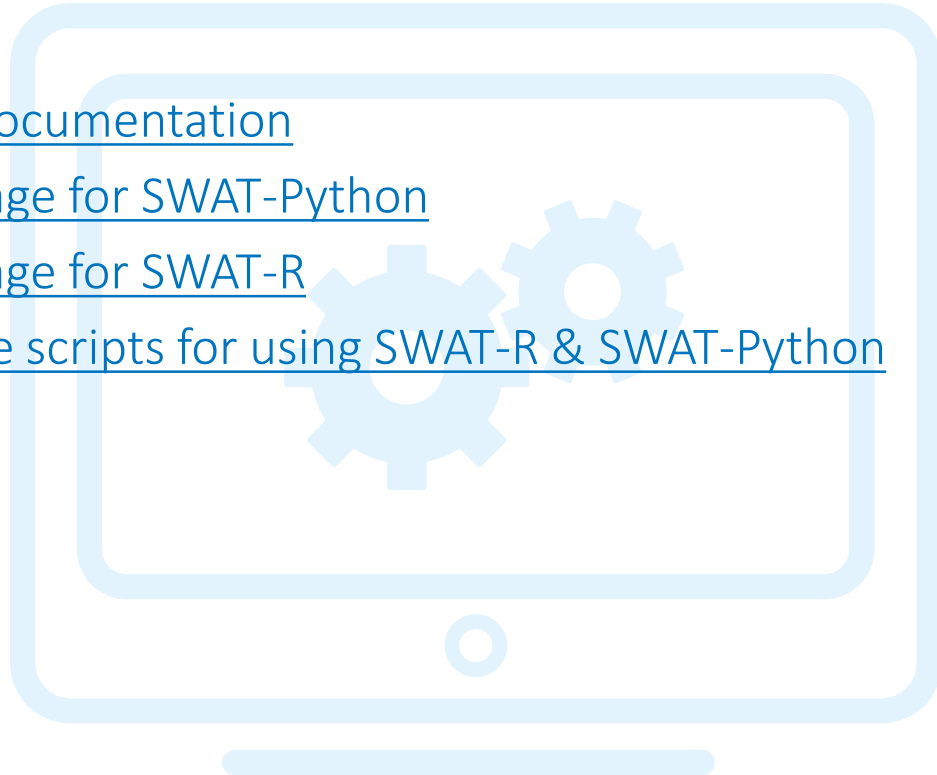
- [SAS Viya Overview](#)
- [SAS Viya Training](#)
- [A Beginner's Guide to Programming in the SAS® Cloud Analytics Services Environment](#)



# 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](#)







# Questions?

Thank you for your time and attention!

Connect with me:

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

Twitter: @Melodie\_Rush

sas.com