How to Select a Random Sample in SAS: A Step-by-Step Guide

Authored by

stats writer

December 1, 2025

The ability to select a representative subset from a larger collection of data is fundamental in statistical analysis and data science. In the world of statistical programming, specifically using the **Statistical Analysis System (SAS)**, selecting a random sample is a streamlined process crucial for tasks such as quality control, large-scale survey analysis, or machine learning model training. A random sample ensures that every observation in the source data has an equal chance of being included, minimizing selection bias and allowing for robust inferential statistics.

SAS provides several powerful procedures for complex sampling designs, but the most common and robust tool for selecting a simple random sample is the **PROC SURVEYSELECT** procedure. This procedure is designed specifically for probability sampling methods, offering flexibility through various parameters to meet sophisticated research needs, whether you are sampling from a single dataset, managing multiple sources, or implementing complex stratified sampling techniques.

Understanding how to effectively utilize **PROC SURVEYSELECT** is essential for any SAS user involved in serious statistical work. This guide focuses on the two primary methods for generating a random sample: specifying a fixed sample size (n) or defining a proportion of the total observations (sampling rate). We will explore these methods through practical, reproducible examples.

Understanding PROC SURVEYSELECT for Sampling

The **PROC SURVEYSELECT** statement is the cornerstone of sampling within the SAS environment. It is designed to handle intricate sampling scenarios, but for basic simple random sampling (SRS), its syntax is straightforward. The procedure requires specifying the input dataset, the output dataset name, the sampling method, and the criteria for selection (size or rate).

Crucially, we must use the `METHOD=SRS` option to ensure that the selection process follows simple random sampling--meaning observations are selected independently and without replacement. Furthermore, using the `SEED=` option is a professional best practice. A seed is a starting value for the random number generator, ensuring that if you or a colleague rerun the code, the exact same random sample will be generated, guaranteeing reproducibility.

Below are the two most common and effective ways to select rows randomly from a SAS dataset, categorized by the selection criterion:

Method 1: Selecting a Random Sample Using Fixed Sample Size (SAMPSIZE)

This approach is utilized when the researcher needs a specific, fixed number of observations (n) regardless of the total size of the original dataset. This is specified using the `SAMPSIZE=` parameter within the **PROC SURVEYSELECT** statement. This method is ideal when computational constraints or resource limitations dictate the maximum number of observations that can be

processed. In the following syntax, we specify that we want to retrieve exactly 3 observations.

```
proc surveyselect data=original_data
out=random_sample
method=srs /*specify simple random sampling as sampling method*/
sampsize=3 /*select 3 observations randomly*/
seed=123; /*set seed to make this example reproducible*/
run;
```

Method 2: Selecting a Random Sample Using a Proportion of Observations (SAMPRATE)

Alternatively, researchers often prefer to define the sample as a percentage or proportion of the total population size. This ensures the sample size scales automatically if the source dataset changes, maintaining a consistent representation rate. This is achieved by using the `SAMPRATE=` parameter, where the value is provided as a decimal between 0 and 1 (e.g., 0.2 represents 20%). This method is typically preferred when maintaining a consistent sampling fraction is more important than achieving a specific count, especially in quality assurance or monitoring systems.

```
proc surveyselect data=original_data
out=random_sample
method=srs /*specify simple random sampling as sampling method*/
samprate=0.2 /*select 20% of all observations randomly*/
seed=123; /*set seed to make this example reproducible*/
run;
```

Creating the Sample Data Structure in SAS

To illustrate these two methods clearly, we will first create a small, representative dataset named `original_data`. This dataset simulates observations collected on different basketball teams, containing variables for team name (character), points scored (numeric), and rebounds collected (numeric). Although this is a small sample, the principles demonstrated apply identically to massive production datasets.

The following SAS code block first uses the `DATA` and `DATALINES` statements to construct the source data. We then verify the creation using the `PROC PRINT` statement, a simple procedure used to display the contents of a SAS table in the output window.

```
/*create dataset*/
```

```
data original_data;
input team $ points rebounds;
datalines;
Warriors 25 8
Wizards 18 12
Rockets 22 6
Celtics 24 11
Thunder 27 14
Spurs 33 19
Nets 31 20
Mavericks 34 10
Kings 22 11
Pelicans 39 23
;
run;

/*view dataset*/
proc print data=original_data;
```

Upon execution, the `PROC PRINT` command generates the following table, confirming that we have 10 observations in our initial population from which to draw our samples.

Obs	team	points	rebounds
1	Warriors	25	8
2	Wizards	18	12
3	Rockets	22	6
4	Celtics	24	11
5	Thunder	27	14
6	Spurs	33	19
7	Nets	31	20
8	Maverick	34	10
9	Kings	22	11
10	Pelicans	39	23

Executing Random Sampling Based on a Fixed Sample Size

In this first practical illustration, our objective is to extract exactly 3 observations from the

`original_data` dataset. We accomplish this by setting the `SAMPSIZE=3` option. By defining the `SEED=123`, we ensure that the selection process is repeatable. The output dataset, `random_sample`, will contain only these three observations, chosen randomly and without any inherent bias toward specific teams or statistical values.

This technique is particularly useful in research phases where a pilot study requires a manageable subset of data for initial testing, or when performing cross-validation where training and validation subsets must adhere to strict size limits.

The following code block executes the sampling process and then uses `PROC PRINT` to display the resulting subset.

```
/*select random sample*/
proc surveyselect data=original_data
out=random_sample
method=srs
sampsize=3
seed=123;
run;

/*view random sample*/
proc print data=random_sample;
```

The execution results in a new dataset, showing only the three rows selected by the procedure. Note that the specific rows selected depend entirely on the seed value, confirming that the process of simple random sampling was successfully executed.

Obs	team	points	rebounds
1	Warriors	25	8
2	Thunder	27	14
3	Pelicans	39	23

As demonstrated by the output image, exactly three rows were randomly selected from the original ten observations, successfully fulfilling the requirement set by `SAMPSIZE=3`.

Implementing Random Sampling Based on a Sampling Proportion

For scenarios where proportional representation is key, we use the `SAMPRATE=` option. In this

example, we aim to select a random sample that constitutes 20% of the total observations available in `original_data`. Since the original dataset has 10 observations, a 20% sample rate (0.2) will yield 2 observations (10 * 0.2 = 2). **PROC SURVEYSELECT** handles the calculation and selection automatically.

This methodology is particularly advantageous when dealing with dynamic datasets that change in size frequently. By using `SAMPRATE`, the sample size adjusts automatically while maintaining the defined sampling fraction, thereby eliminating the need for manual recalculation of the required sample size (n).

We keep the sampling method as `METHOD=SRS` and reuse the same seed (`SEED=123`) to ensure that the randomness is controlled and the result is reproducible across runs. The following code executes this proportional selection.

```
/*select random sample*/
proc surveyselect data=original_data
out=random_sample
method=srs
samprate=0.2
seed=123;
run;

/*view random sample*/
proc print data=random_sample;
```

The resulting output confirms that two rows were randomly selected, corresponding precisely to the 20% sample rate applied to the 10 total observations in the original population.

Obs	team	points	rebounds
1	Warriors	25	8
2	Spurs	33	19

Best Practices for Reproducible Sampling

While the examples above focus on simple random sampling, advanced statistical work often demands more complexity. SAS's **PROC SURVEYSELECT** procedure supports various methods beyond SRS, including stratified sampling, cluster sampling, and systematic sampling. Researchers should always consult statistical theory to determine the most appropriate method for

their specific population and research goals.

Regardless of the method chosen, consistency and reproducibility are paramount in data analysis. We strongly recommend always including the `SEED=` option. Without a defined seed, rerunning the code will produce a different sample every time, making verification and auditing nearly impossible. The seed value itself is arbitrary; consistency in its application is what matters.

Key parameters to master include:

DATA=: Specifies the input population dataset.
OUT=: Names the output dataset containing the selected sample.
METHOD=: Defines the selection technique (e.g., SRS, STRATA, CLUSTER).
SAMPSIZE= or SAMPRATE=: Determines the size criteria for the sample.

Further SAS Sampling Applications

Generating a random sample is often just the first step in a larger analysis workflow. The output dataset created by **PROC SURVEYSELECT** can immediately be passed to other analytical procedures, such as `PROC GLM`, `PROC REG`, or `PROC LOGISTIC`, to conduct statistical tests or build predictive models based on the representative subset.

For more detailed information on complex sampling designs, refer to the extensive official documentation for **PROC SURVEYSELECT**, which provides specifications for multi-stage sampling and incorporating sampling weights.

If you are interested in exploring other common SAS tasks beyond basic sampling, the following resources may be helpful:

The following tutorials explain how to perform other common tasks in SAS:

How to Calculate Summary Statistics in SAS
Performing Linear Regression Analysis in SAS
Merging Multiple Datasets Using PROC SQL