Learning Cluster Sampling with R: A Practical Guide

Authored by

Mohammed looti

November 7, 2025

Introduction to Probability Sampling and Cluster Methodology

In the field of statistical analysis and research, it is often impractical or impossible to collect data from every single member of a population. Consequently, researchers rely on meticulously designed sampling methods to select a representative subset. This selected subset, or sample, allows analysts to draw meaningful inferences and conclusions about the larger group with quantifiable levels of certainty. Choosing the correct methodology is paramount to ensuring the validity and generalizability of the final results.

Among the various approaches to probability sampling, **cluster sampling** stands out as a powerful and cost-effective technique, particularly when dealing with geographically dispersed or naturally grouped populations. Unlike simple random sampling, where individuals are selected independently, cluster sampling involves dividing the target population into discrete, non-overlapping subgroups known as clusters. These clusters are often naturally occurring units, such as schools, city blocks, hospitals, or, as we will explore in our example, pre-defined tour groups.

The defining characteristic of **cluster sampling** is that the researcher randomly selects a subset of these clusters, and then every single element within the chosen clusters is included in the final sample. This approach simplifies the logistical challenges associated with data collection. This tutorial will focus specifically on demonstrating how to execute this specialized sampling procedure efficiently within the R environment, providing a clear, reproducible workflow for statisticians and data scientists.

Advantages and Practical Applications of Cluster Sampling

The primary appeal of **cluster sampling** is its logistical efficiency. When the population is spread across a wide area, collecting data from randomly selected individuals using methods like simple random sampling can incur significant travel time and cost. By contrast, if a researcher selects only a few geographic clusters (e.g., specific cities or districts), all data collection can be concentrated in those localized areas, drastically reducing operational expenses and simplifying fieldwork.

Furthermore, this method is especially valuable when a complete list of all individual members of the population is unavailable or difficult to compile, but a list of the clusters themselves exists. For instance, a government agency might not have a perfect database of every resident, but they will certainly have a list of all defined electoral precincts (clusters). The trade-off for this convenience is often a slight decrease in statistical precision compared to other methods, as individuals within the same cluster tend to share more characteristics than those drawn randomly from the entire population, leading to increased sampling error.

Our practical demonstration below will illustrate a common application: a tourism company wishing to gauge customer satisfaction. Instead of randomly interrupting individual customers across all

tours (which would require complex scheduling and tracking), the company treats each tour as a cluster. By randomly selecting a few tours and surveying everyone on those selected tours, they achieve a representative sample efficiently. This scenario perfectly highlights the balance between methodological rigor and real-world feasibility that **cluster sampling** provides, particularly when implemented using statistical software like R.

Setting the Stage: Preparing Data for Cluster Sampling in R

To effectively demonstrate **cluster sampling**, we will simulate a practical scenario involving a company that offers ten distinct city tours on a given day. The company aims to collect feedback on customer experience, measured on a scale of 1 to 10. For the purpose of this example, we assume that each of the ten tours hosts exactly 20 customers, resulting in a total population size of 200 customers (10 clusters * 20 members each). The objective is to randomly select four of these ten tours (clusters) and survey all 80 customers belonging to those four selected groups.

We begin by creating a synthetic **data frame** in R that mirrors this structure. A **data frame** is the fundamental structure for storing tabular data in R, where each column can contain a different mode of data. We must ensure our data clearly defines the cluster identifier (the 'tour' number) and the variable of interest (the 'experience' rating). The use of the `set.seed(1)` function is a standard best practice in R programming; it guarantees that the subsequent generation of random numbers will be identical every time the code is executed, ensuring the reproducibility of our results for verification.

The following code block constructs our simulated population of 200 customer records. The `tour` variable uses the `rep(1:10, each=20)` command to assign 20 sequential records to each of the 10 tours. The `experience` variable is generated using `rnorm(200, mean=7, sd=1)`, which draws 200 random values from a normal distribution centered around a mean rating of 7, simulating realistic customer feedback.

 # Ensure the simulation is reproducible
set.seed(1)

# Create the population data frame (200 customers across 10 tours)
df <- data.frame(tour = rep(1:10, each=20),
experience = rnorm(200, mean=7, sd=1))

# Inspect the initial structure of the data frame
head(df)

tour experience
1 1 6.373546

2 1 7.183643
3 1 6.164371
4 1 8.595281
5 1 7.329508
6 1 6.179532

Implementing Single-Stage Cluster Sampling using R

Once the population data structure is established, the next critical step is the execution of the **cluster sampling** process itself. This requires a two-step approach in R: first, randomly selecting the cluster identifiers (the tour numbers), and second, filtering the original population data to include only those records associated with the selected identifiers.

The selection of the clusters is achieved using the powerful `sample()` function. We first identify all unique cluster IDs using `unique(df$tour)`, which returns the numbers 1 through 10. We then pass this vector of IDs to `sample()`, specifying `size=4`, indicating that we want to randomly select four clusters. Crucially, we set `replace=F` (or `FALSE`), which ensures that once a cluster is selected, it cannot be selected again, aligning with the standards of simple random sampling of clusters.

The final step involves subsetting the main data frame (`df`) to create our final cluster sample (`cluster_sample`). We use logical indexing combined with the special R operator `%in%`. The `%in%` operator is invaluable for this task as it checks whether the values in the `df$tour` column are present within the vector of randomly selected cluster IDs (stored in the `clusters` variable). This highly efficient operation pulls all 20 records corresponding to each of the four chosen tours simultaneously, completing the single-stage cluster sample extraction.

 # Randomly choose 4 tour groups out of the 10 available clusters
clusters <- sample(unique(df$tour), size=4, replace=F)

# Define the sample as all members who belong to one of the 4 tour groups (using %in% operator)
cluster_sample <- df

# View how many customers were included from each selected tour
table(cluster_sample$tour)

2 7 8 10
20 20 20 20

Analyzing and Interpreting the Cluster Sample Results

The successful execution of the R code yields two immediate results. First, the `clusters` variable contains the specific identifiers of the four tours that were randomly selected. Due to the initial use of `set.seed(1)` for reproducibility, the output consistently shows that tours 2, 7, 8, and 10 were chosen by the `sample()` function in this particular run. Second, the `table()` function applied to the resulting `cluster_sample$tour` confirms the structure of our final sample.

The output from the `table()` command clearly summarizes the composition of the acquired sample, verifying that all members of the selected clusters were included, which is the core principle of single-stage **cluster sampling**. The resulting distribution confirms our methodology was executed correctly:

20 customers from tour group #2 were successfully included in the sample.

20 customers from tour group #7 were successfully included in the sample.

20 customers from tour group #8 were successfully included in the sample.

20 customers from tour group #10 were successfully included in the sample.

The final sample is thus perfectly composed of 80 total customers (4 tours * 20 customers/tour) drawn from the four randomly chosen tour groups. This structured approach, facilitated by the data manipulation capabilities of R, provides the necessary data structure for subsequent statistical analysis, such as calculating the mean customer experience rating for the entire sample and using that mean to estimate the population mean.

Related: For specific details on the R operator used in the filtering step, refer to: How to Use %in% Operator in R

Additional Resources for Sampling Techniques

Explore these related tutorials to expand your expertise in statistical sampling methodology:

Understanding Different Types of Sampling Methods
Stratified Sampling in R
Systematic Sampling in R