Algorithm Foundations of Data Science and Engineering

Lecture 3: Sampling

YANHAO WANG

DaSE @ ECNU
(for course related communications)
yhwang@dase.ecnu.edu.cn

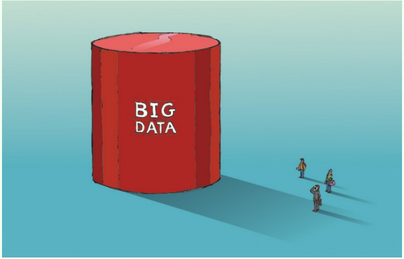
Sep. 10, 2021

Outline

Motivation of Sampling

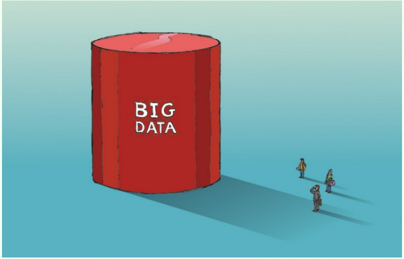
Sampling
- Simple Random Sampling
- Systematic Sampling
- Stratified Sampling
- Reservoir Sampling

Data volume in the world



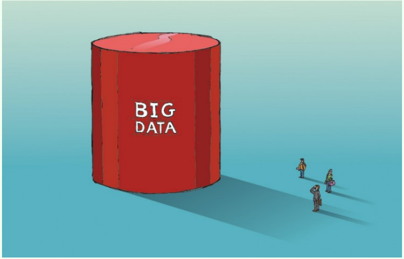
- 1TB (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- 1PB (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- 1EB (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- 1ZB (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- 1YB (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

Data volume in the world



- 1TB (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- 1PB (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- 1EB (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- 1ZB (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- 1YB (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

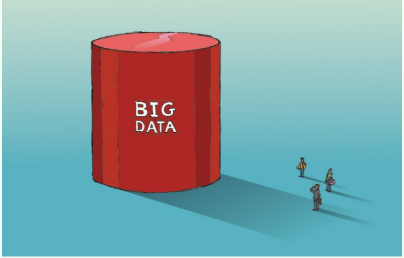
Data volume in the world



World's data volume:

- 1TB (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- 1PB (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- 1EB (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- 1ZB (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- 1YB (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

Data volume in the world

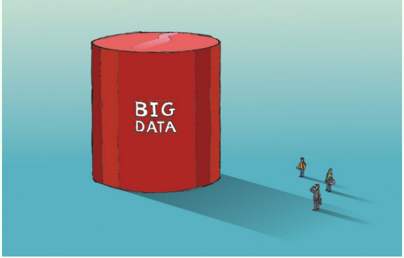


World's data volume:

- A full 90% of all the data in the world has been generated over the last two years;

- 1TB (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- 1PB (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- 1EB (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- 1ZB (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- 1YB (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

Data volume in the world

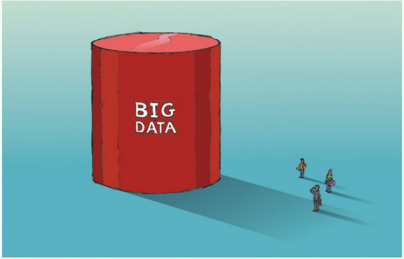


- $1TB$ (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- $1PB$ (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- $1EB$ (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- $1ZB$ (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- $1YB$ (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

World's data volume:

- A full 90% of all the data in the world has been generated over the last two years;
- The world's data volume is expected to grow 40 per cent per year;

Data volume in the world

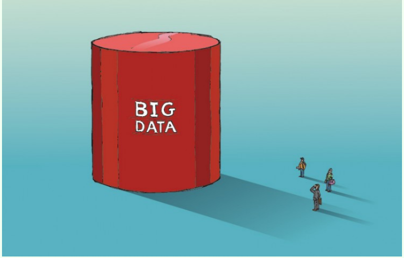


- 1TB (Terabyte) = 2^{10} GB = 2^{40} B;
- 1PB (Petabyte) = 2^{10} TB = 2^{50} B;
- 1EB (Exabyte) = 2^{10} PB = 2^{60} B;
- 1ZB (Zettabyte) = 2^{10} EB = 2^{70} B;
- 1YB (YottaByte) = 2^{10} ZB = 2^{80} B;

World's data volume:

- A full 90% of all the data in the world has been generated over the last two years;
- The world's data volume is expected to grow 40 per cent per year;
- Consumer IP internet traffic amounted to 100 EB per month in 2017;

Data volume in the world



- 1TB (Terabyte) $= 2^{10}$ GB $= 2^{40}$ B;
- 1PB (Petabyte) $= 2^{10}$ TB $= 2^{50}$ B;
- 1EB (Exabyte) $= 2^{10}$ PB $= 2^{60}$ B;
- 1ZB (Zettabyte) $= 2^{10}$ EB $= 2^{70}$ B;
- 1YB (YottaByte) $= 2^{10}$ ZB $= 2^{80}$ B;

World's data volume:

- A full 90% of all the data in the world has been generated over the last two years;
- The world's data volume is expected to grow 40 per cent per year;
- Consumer IP internet traffic amounted to 100 EB per month in 2017;
- Data volume in the world will amount to 40 ZB by 2020.

Hadoop: A brief history of big data processing

Hadoop was created by Doug Cutting, the creator of Apache Lucene. Hadoop has its origins in Apache Nutch, which is a part of the Lucene project.

Hadoop: A brief history of big data processing

Hadoop was created by Doug Cutting, the creator of Apache Lucene. Hadoop has its origins in Apache Nutch, which is a part of the Lucene project.

- Nutch was started in 2002. However, they realized that it wouldn't scale to the billions of pages on the Web;

Hadoop: A brief history of big data processing

Hadoop was created by Doug Cutting, the creator of Apache Lucene. Hadoop has its origins in Apache Nutch, which is a part of the Lucene project.

- Nutch was started in 2002. However, they realized that it wouldn't scale to the billions of pages on the Web;
- Google published papers in 2003 and 2004 that described GFS and MapReduce;

Hadoop: A brief history of big data processing

Hadoop was created by Doug Cutting, the creator of Apache Lucene. Hadoop has its origins in Apache Nutch, which is a part of the Lucene project.

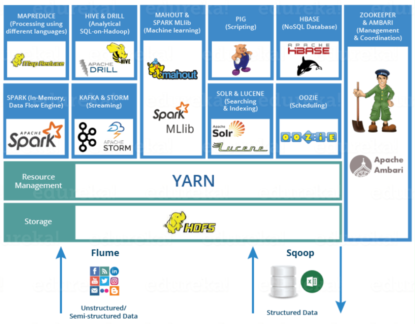
- Nutch was started in 2002. However, they realized that it wouldn't scale to the billions of pages on the Web;
- Google published papers in 2003 and 2004 that described GFS and MapReduce;
- In 2006, Doug Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale;

Hadoop: A brief history of big data processing

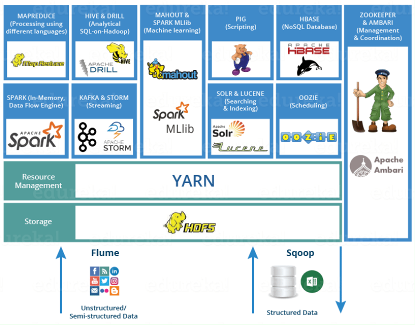
Hadoop was created by Doug Cutting, the creator of Apache Lucene. Hadoop has its origins in Apache Nutch, which is a part of the Lucene project.

- Nutch was started in 2002. However, they realized that it wouldn't scale to the billions of pages on the Web;
- Google published papers in 2003 and 2004 that described GFS and MapReduce;
- In 2006, Doug Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale;
- In February 2008, Yahoo! announced that its production search index was being generated by a 10,000-core Hadoop cluster.

Ecosystem for big data processing

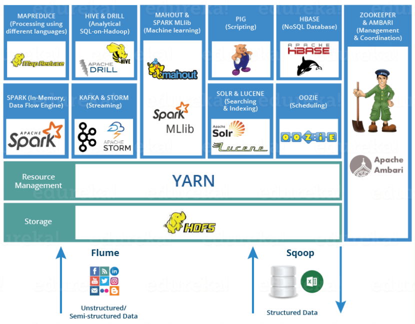


Ecosystem for big data processing



A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

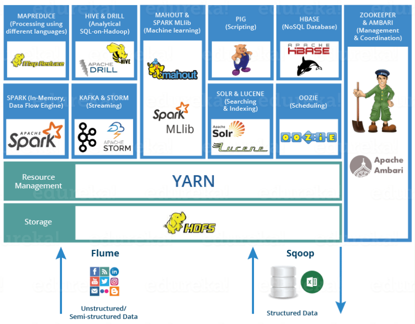
Ecosystem for big data processing



Infrastructural technologies are the core of the big data ecosystem.

A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

Ecosystem for big data processing

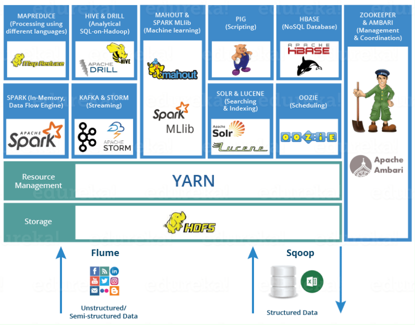


Infrastructural technologies are the core of the big data ecosystem.

- Hadoop;

A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

Ecosystem for big data processing

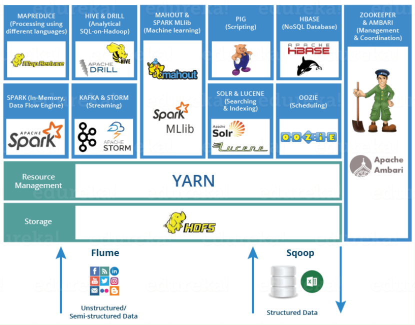


A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

Infrastructural technologies are the core of the big data ecosystem.

- Hadoop;
- NoSQL, like HBase;

Ecosystem for big data processing

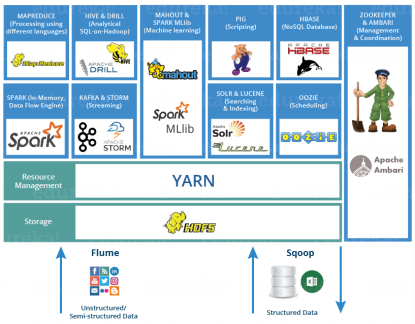


A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

Infrastructural technologies are the core of the big data ecosystem.

- Hadoop;
- NoSQL, like HBase;
- Spark;

Ecosystem for big data processing

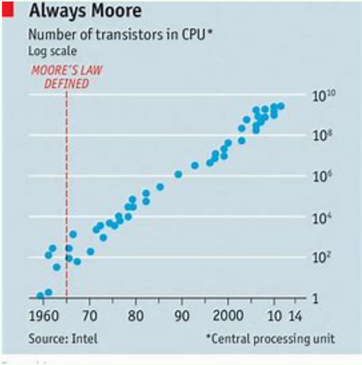


A number of large-scale data processing frameworks have thereby been developed, such as MapReduce, Spark, Storm, Flink, Dryad, Caffe, Tensorflow.

Infrastructural technologies are the core of the big data ecosystem.

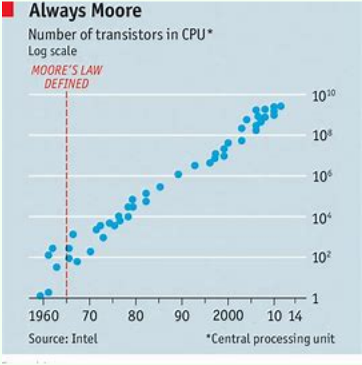
- Hadoop;
- NoSQL, like HBase;
- Spark;
- Storm;
- …;

Moore's Law is dead



Observing that the total number of components in these circuits had roughly doubled each year [Gordon E. Moore 1965].

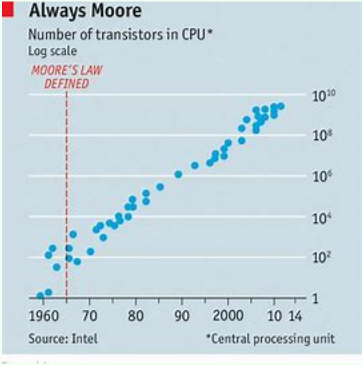
Moore's Law is dead



Observing that the total number of components in these circuits had roughly doubled each year [Gordon E. Moore 1965].

History proved Moore correct.

Moore's Law is dead

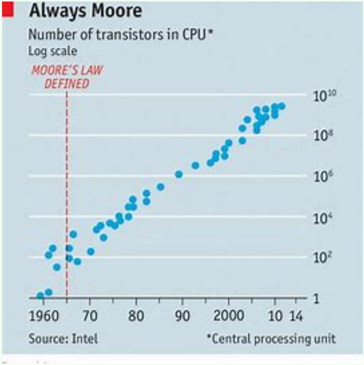


Observing that the total number of components in these circuits had roughly doubled each year [Gordon E. Moore 1965].

History proved Moore correct.

- Moore's lawthough its doubling period was lengthened to 18 months in the mid-1970s;

Moore's Law is dead

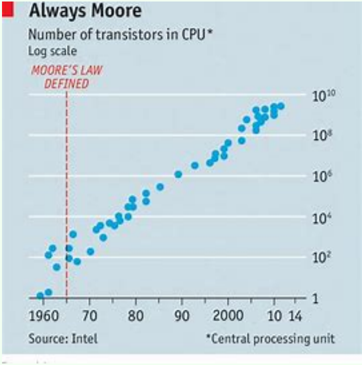


Observing that the total number of components in these circuits had roughly doubled each year [Gordon E. Moore 1965].

History proved Moore correct.

- Moore's lawthough its doubling period was lengthened to 18 months in the mid-1970s;
- It continued into the second decade of the 21st century with tens of nanometres in size;

Moore's Law is dead

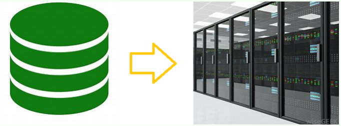


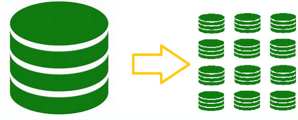
Observing that the total number of components in these circuits had roughly doubled each year [Gordon E. Moore 1965].

History proved Moore correct.

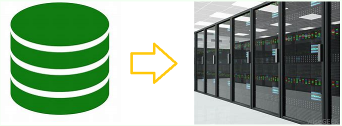
- Moore's lawthough its doubling period was lengthened to 18 months in the mid-1970s;
- It continued into the second decade of the 21st century with tens of nanometres in size;
- The processing power of computers increases exponentially every couple of years has hit its limit, according to Jensen Huang, CEO of Nvidia.

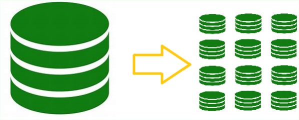
Choices of big data processing





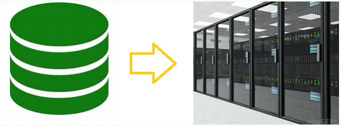
Choices of big data processing



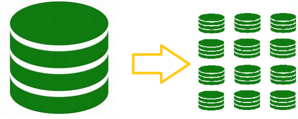


Given the characteristics of the unprecedented amount of data, the speed of data production, and the multiple of the structure of data, large-scale data processing is essential to analyzing and mining such big data timely. However, it is not the only choice.

Choices of big data processing

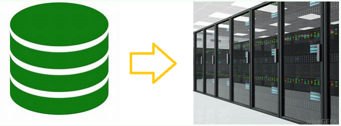


Given the characteristics of the unprecedented amount of data, the speed of data production, and the multiple of the structure of data, large-scale data processing is essential to analyzing and mining such big data timely. However, it is not the only choice.

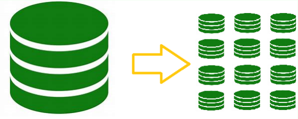


We don't need to poll every single American of voting age whenever these events occur. We may need only polling 1,100 people randomly selected.

Choices of big data processing



Given the characteristics of the unprecedented amount of data, the speed of data production, and the multiple of the structure of data, large-scale data processing is essential to analyzing and mining such big data timely. However, it is not the only choice.



We don't need to poll every single American of voting age whenever these events occur. We may need only polling 1,100 people randomly selected. With that (relatively) small set of data, they are extrapolating how the rest of the public feels and is likely to vote.

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;
  - Systematic sampling;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;
  - Systematic sampling;
  - Stratified sampling;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;
  - Systematic sampling;
  - Stratified sampling;
  - Cluster sampling;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;
  - Systematic sampling;
  - Stratified sampling;
  - Cluster sampling;
  - Multi-stage sampling;

Sampling

Sample is a subset of the population on which observations are taken for obtaining information about the population.

- Since studying a sample we wish to draw valid conclusions about the population, sample should desirably be "representative" of the target population;
- The most common sampling techniques used for official surveys are:
  - Simple random sampling;
  - Systematic sampling;
  - Stratified sampling;
  - Cluster sampling;
  - Multi-stage sampling;
- All are examples of probability sampling.

Outline

Simple random sampling

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$Pr[X_i = 1] = \frac{1}{N}.$$

Simple random sampling

The procedure of sampling in which the units are selected with probability proportional to a given measure of size.

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$Pr[X_i = 1] = \frac{1}{N}.$$

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$Pr[X_i = 1] = \frac{1}{N}.$$

The procedure of sampling in which the units are selected with probability proportional to a given measure of size.

- Simple random sampling is simplest method of probability sampling;

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$Pr[X_i = 1] = \frac{1}{N}.$$

The procedure of sampling in which the units are selected with probability proportional to a given measure of size.

- Simple random sampling is simplest method of probability sampling;
- Simple random sampling is special type of equal probability selection method;

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$Pr[X_i = 1] = \frac{1}{N}.$$

The procedure of sampling in which the units are selected with probability proportional to a given measure of size.

- Simple random sampling is simplest method of probability sampling;
- Simple random sampling is special type of equal probability selection method;
  - With replacement;

Simple random sampling

Let N be the volume of population, and X_i be a r.v.

$$X_i = \begin{cases} 1, & \text{the i-th unit is selected;} \\ 0, & \text{otherwise.} \end{cases}$$

Then

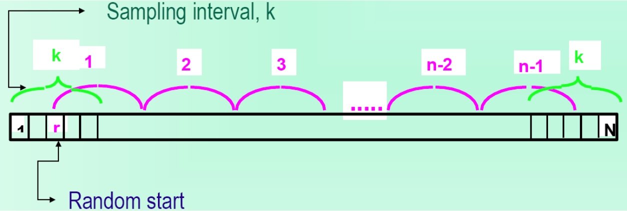
$$Pr[X_i = 1] = \frac{1}{N}.$$

The procedure of sampling in which the units are selected with probability proportional to a given measure of size.

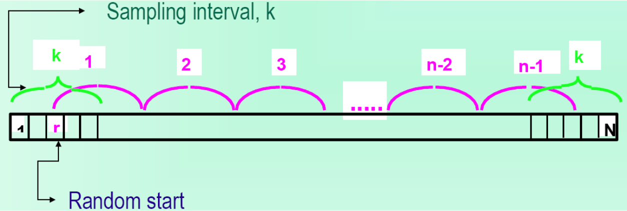
- Simple random sampling is simplest method of probability sampling;
- Simple random sampling is special type of equal probability selection method;
    - With replacement;
    - Without replacement.

Outline

Systematic sampling

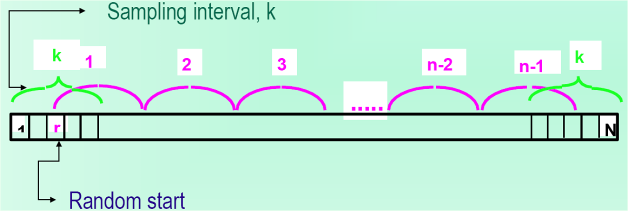


Systematic sampling



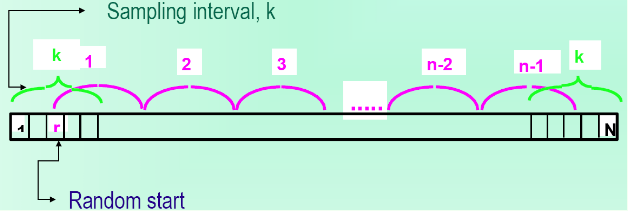
- Systematic sampling involves selecting n sample units from a population of N units;

Systematic sampling



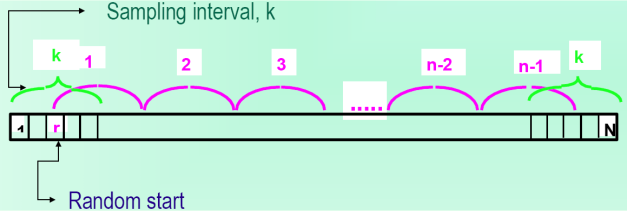
- Systematic sampling involves selecting n sample units from a population of N units;
- Instead of randomly choosing the n units in the sample, a skip pattern is run through a list (frame) of the N units to select the sample;

Systematic sampling



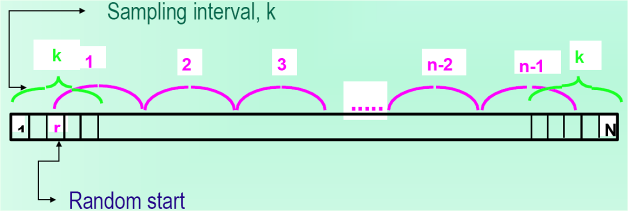
- Systematic sampling involves selecting n sample units from a population of N units;
- Instead of randomly choosing the n units in the sample, a skip pattern is run through a list (frame) of the N units to select the sample;
  - Decide on a sample size n and compute the skip, $k = \frac{N}{n}$;

Systematic sampling



- Systematic sampling involves selecting n sample units from a population of N units;
- Instead of randomly choosing the n units in the sample, a skip pattern is run through a list (frame) of the N units to select the sample;
  - Decide on a sample size n and compute the skip, $k = \frac{N}{n}$;
  - Choose a random start r between 1 and k (inclusive);

Systematic sampling



- Systematic sampling involves selecting n sample units from a population of N units;
- Instead of randomly choosing the n units in the sample, a skip pattern is run through a list (frame) of the N units to select the sample;
  - Decide on a sample size n and compute the skip, $k = \frac{N}{n}$;
  - Choose a random start r between 1 and k (inclusive);
  - Add "k" to selected random number to select the next unit repeatedly.

Problem of linear systematic sampling

- If N is a multiple of n, then the number of units in each of the k possible systematic samples is n;

Problem of linear systematic sampling

- If N is a multiple of n, then the number of units in each of the k possible systematic samples is n;
  - In this case systematic sampling amounts to grouping the N units into k samples of exactly n units each in a systematic manner and selecting one of them with probability $\frac{1}{k}$

Problem of linear systematic sampling

- If N is a multiple of n, then the number of units in each of the k possible systematic samples is n;
  - In this case systematic sampling amounts to grouping the N units into k samples of exactly n units each in a systematic manner and selecting one of them with probability $\frac{1}{k}$
  - In this case, the sampling scheme is equal probability selection method (epsem);

Problem of linear systematic sampling

- If N is a multiple of n, then the number of units in each of the k possible systematic samples is n;
  - In this case systematic sampling amounts to grouping the N units into k samples of exactly n units each in a systematic manner and selecting one of them with probability $\frac{1}{k}$
  - In this case, the sampling scheme is equal probability selection method (epsem);
- But, if $\frac{N}{n}$ is not an integer, then the number of units selected systematically with the sampling interval

$$k \approx \text{ nearest integer to } \frac{N}{n}$$

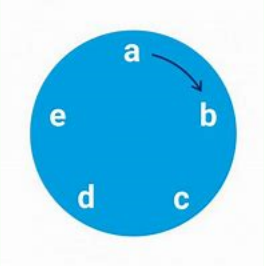
Problem of linear systematic sampling

- If N is a multiple of n, then the number of units in each of the k possible systematic samples is n;
  - In this case systematic sampling amounts to grouping the N units into k samples of exactly n units each in a systematic manner and selecting one of them with probability $\frac{1}{k}$
  - In this case, the sampling scheme is equal probability selection method (epsem);
- But, if $\frac{N}{n}$ is not an integer, then the number of units selected systematically with the sampling interval

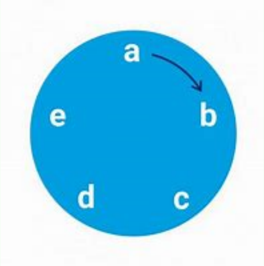
$$k \approx \text{ nearest integer to } \frac{N}{n}$$

- This problem may be overcome by adopting a device, known as circular systematic sampling;

Circular systematic sampling

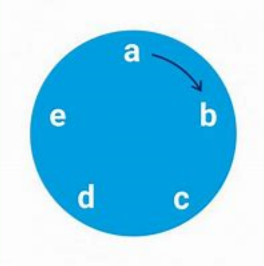


Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

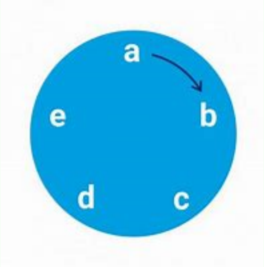
Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

- Determine the interval k C rounding down to the integer nearest to $\frac{N}{n}$, e.g., If $N = 15$ and $n = 4$, then k is taken as 3 and not 4;

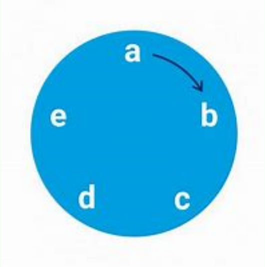
Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

- Determine the interval k C rounding down to the integer nearest to $\frac{N}{n}$, e.g., If $N = 15$ and $n = 4$, then k is taken as 3 and not 4;
- Take a random start between 1 and N;

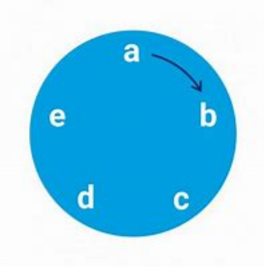
Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

- Determine the interval k C rounding down to the integer nearest to $\frac{N}{n}$, e.g., If $N = 15$ and $n = 4$, then k is taken as 3 and not 4;
- Take a random start between 1 and N;
- Skip through the circle by k units each time to select the next unit until n units are selected;

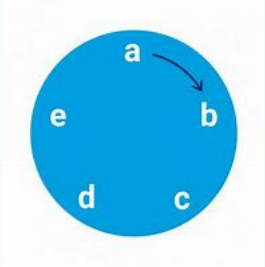
Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

- Determine the interval k C rounding down to the integer nearest to $\frac{N}{n}$, e.g., If $N = 15$ and $n = 4$, then k is taken as 3 and not 4;
- Take a random start between 1 and N;
- Skip through the circle by k units each time to select the next unit until n units are selected;
- Thus there could be N possible distinct samples instead of k;

Circular systematic sampling



Useful when $\frac{N}{n}$ is not integer.

- Determine the interval k C rounding down to the integer nearest to $\frac{N}{n}$, e.g., If $N = 15$ and $n = 4$, then k is taken as 3 and not 4;
- Take a random start between 1 and N;
- Skip through the circle by k units each time to select the next unit until n units are selected;
- Thus there could be N possible distinct samples instead of k;

Ensures each unit equal chance of being selected into sample.

Pros and cons of circular systematic sampling

- Pros

Pros and cons of circular systematic sampling

- Pros
  - Operationally convenient - easier to draw a sample;

Pros and cons of circular systematic sampling

- Pros
  - Operationally convenient - easier to draw a sample;
  - It distributes the sample more evenly over the population.

Pros and cons of circular systematic sampling

- Pros
  - Operationally convenient - easier to draw a sample;
  - It distributes the sample more evenly over the population. Thus it is likely to be more efficient than SRSWOR, particularly when the ordering of the units in the list is related to characteristics of the variable of interest;

Pros and cons of circular systematic sampling

- Pros
  - Operationally convenient - easier to draw a sample;
  - It distributes the sample more evenly over the population. Thus it is likely to be more efficient than SRSWOR, particularly when the ordering of the units in the list is related to characteristics of the variable of interest;
- Cons

Pros and cons of circular systematic sampling

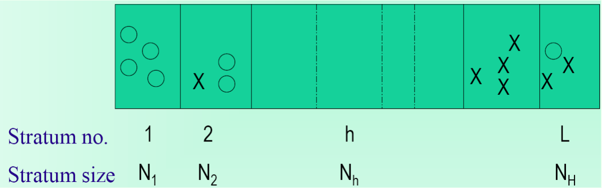
- Pros
  - Operationally convenient - easier to draw a sample;
  - It distributes the sample more evenly over the population. Thus it is likely to be more efficient than SRSWOR, particularly when the ordering of the units in the list is related to characteristics of the variable of interest;
- Cons
  - Requires complete list of the population;

Pros and cons of circular systematic sampling

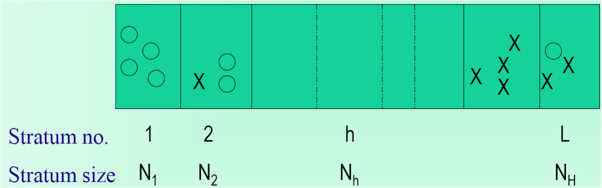
- Pros
  - Operationally convenient - easier to draw a sample;
  - It distributes the sample more evenly over the population. Thus it is likely to be more efficient than SRSWOR, particularly when the ordering of the units in the list is related to characteristics of the variable of interest;
- Cons
  - Requires complete list of the population;
  - A bad arrangement of the units may produce a very inefficient sample.

Outline

Stratified sampling

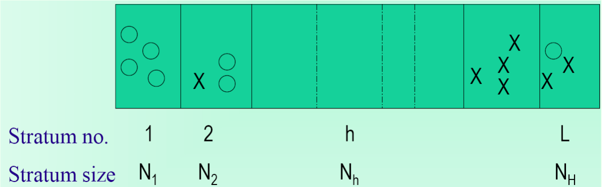


Stratified sampling



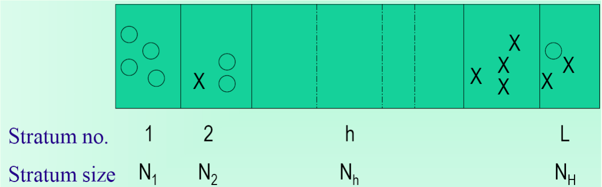
- Divides the population into a number of distinct groups (strata) based on auxiliary information;

Stratified sampling



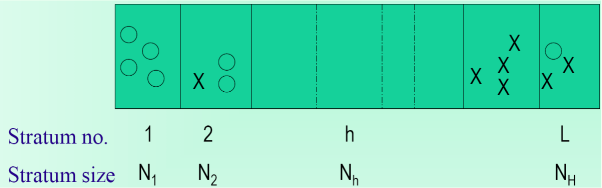
- Divides the population into a number of distinct groups (strata) based on auxiliary information;
- Each stratum is composed of units that satisfy the condition set by the values of the stratifying variable;

Stratified sampling



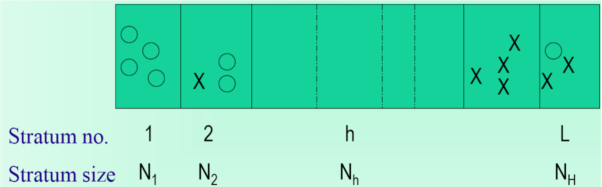
- Divides the population into a number of distinct groups (strata) based on auxiliary information;
- Each stratum is composed of units that satisfy the condition set by the values of the stratifying variable;
  - Reduce the standard error of the estimates;

Stratified sampling



- Divides the population into a number of distinct groups (strata) based on auxiliary information;
- Each stratum is composed of units that satisfy the condition set by the values of the stratifying variable;
  - Reduce the standard error of the estimates;
  - Provide separate estimates required for each sub-division of the population - “domain” estimates;

Stratified sampling



- Divides the population into a number of distinct groups (strata) based on auxiliary information;
- Each stratum is composed of units that satisfy the condition set by the values of the stratifying variable;
  - Reduce the standard error of the estimates;
  - Provide separate estimates required for each sub-division of the population - "domain" estimates;
  - Using different sampling procedures for different sub-population to increase efficiency of the estimates.

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;
- Sampling in each stratum is carried out independently;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;
- Sampling in each stratum is carried out independently;
  - Sampling fractions may differ;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;
- Sampling in each stratum is carried out independently;
  - Sampling fractions may differ;
  - Selection procedures may also be different;

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;
- Sampling in each stratum is carried out independently;
  - Sampling fractions may differ;
  - Selection procedures may also be different;
- The total sample size is distributed over all the strata - allocation

Stratified sampling Cont'd

- Division or stratification of the population into homogeneous (similar) groups called strata;
- Selecting the sample using a selection procedure:
  - Like SRS or systematic sampling within each stratum;
  - Independent of the other strata;
- Sampling in each stratum is carried out independently;
  - Sampling fractions may differ;
  - Selection procedures may also be different;
- The total sample size is distributed over all the strata - allocation
- At the end of the survey, the stratum results are combined to provide an estimate for entire population

Clustering and stratification

Defining Strata

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;
  - Depends on availability of stratifying information in sampling frame: less information, fewer strata;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;
  - Depends on availability of stratifying information in sampling frame: less information, fewer strata;
  - At least two sampling units per stratum to be able to compute sampling error;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;
  - Depends on availability of stratifying information in sampling frame: less information, fewer strata;
  - At least two sampling units per stratum to be able to compute sampling error;
- Sample allocation to strata

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;
  - Depends on availability of stratifying information in sampling frame: less information, fewer strata;
  - At least two sampling units per stratum to be able to compute sampling error;
- Sample allocation to strata
  - Proportionate allocation: an uniform sampling fraction is applied to each strata; that is, the sample size selected from each stratum is made proportionate to the population size of the stratum;

Clustering and stratification

Defining Strata

- Choice of stratification variables (location, degree, etc):
  - Homogeneous within strata; Heterogeneous across strata;
  - Highly correlated with study variables;
- Number of strata;
  - Depends on availability of stratifying information in sampling frame: less information, fewer strata;
  - At least two sampling units per stratum to be able to compute sampling error;
- Sample allocation to strata
  - Proportionate allocation: an uniform sampling fraction is applied to each strata; that is, the sample size selected from each stratum is made proportionate to the population size of the stratum;
  - Disproportionate allocation: different sampling rates are used deliberately in different strata.

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):
  - This implies that the overall sampling fraction is

$$\frac{n_h}{N_h} = \frac{n}{N};$$

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):
  - This implies that the overall sampling fraction is

$$\frac{n_h}{N_h} = \frac{n}{N};$$

  - The number of elements taken from the h-th stratum is

$$n_h = N_h \cdot \frac{n}{N}.$$

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):
  - This implies that the overall sampling fraction is

$$\frac{n_h}{N_h} = \frac{n}{N};$$

  - The number of elements taken from the h-th stratum is

$$n_h = N_h \cdot \frac{n}{N}.$$

- For a given total variability in the population, the gain is greater if:

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):
  - This implies that the overall sampling fraction is

$$\frac{n_h}{N_h} = \frac{n}{N};$$

  - The number of elements taken from the h-th stratum is

$$n_h = N_h \cdot \frac{n}{N}.$$

- For a given total variability in the population, the gain is greater if:
  - The strata mean are more heterogeneous (more unequal strata mean);

Proportionate allocation

In proportionate stratification, $\frac{n_h}{N_h}$, is specified to be the same for each stratum.

- Choice of stratification variables (location, degree, etc):
  - This implies that the overall sampling fraction is

$$\frac{n_h}{N_h} = \frac{n}{N};$$

  - The number of elements taken from the h-th stratum is

$$n_h = N_h \cdot \frac{n}{N}.$$

- For a given total variability in the population, the gain is greater if:
  - The strata mean are more heterogeneous (more unequal strata mean);
  - The element values within the strata are more homogeneous.

Optimum allocation

Uses widely different sampling rates for the various strata.

Optimum allocation

Uses widely different sampling rates for the various strata.

- Objective: to achieve the least variance for the overall mean for the given sample size;

Optimum allocation

Uses widely different sampling rates for the various strata.

- Objective: to achieve the least variance for the overall mean for the given sample size;
- Without cost consideration, the allocation is

$$n_h = n \cdot \frac{N_h \sigma_h}{\sum N_h \sigma_h}.$$

Optimum allocation

Uses widely different sampling rates for the various strata.

- Objective: to achieve the least variance for the overall mean for the given sample size;
- Without cost consideration, the allocation is

$$n_h = n \cdot \frac{N_h \sigma_h}{\sum N_h \sigma_h}.$$

- This gives better efficiency as compared to proportionate allocation:

$$V_{SRS} \geq V_{prop} \geq V_{opt}.$$

Optimum allocation

Uses widely different sampling rates for the various strata.

- Objective: to achieve the least variance for the overall mean for the given sample size;
- Without cost consideration, the allocation is

$$n_h = n \cdot \frac{N_h \sigma_h}{\sum N_h \sigma_h}.$$

- This gives better efficiency as compared to proportionate allocation:

$$V_{SRS} \geq V_{prop} \geq V_{opt}.$$

- In practice, σ_h is unknown.

Optimum allocation

Uses widely different sampling rates for the various strata.

- Objective: to achieve the least variance for the overall mean for the given sample size;
- Without cost consideration, the allocation is

$$n_h = n \cdot \frac{N_h \sigma_h}{\sum N_h \sigma_h}.$$

- This gives better efficiency as compared to proportionate allocation:

$$V_{SRS} \geq V_{prop} \geq V_{opt}.$$

- In practice, σ_h is unknown.
- This refers to a systematic sampling with the units arranged in a certain order

Outline

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

- Queries on Google searches in May

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

- Queries on Google searches in May
- Products bought at Walmart during the Christmas season

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

- Queries on Google searches in May
- Products bought at Walmart during the Christmas season

Your goal is to efficiently return a random sample of 1,000 elements evenly distributed from the original stream. How would you do it?

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

- Queries on Google searches in May
- Products bought at Walmart during the Christmas season

Your goal is to efficiently return a random sample of 1,000 elements evenly distributed from the original stream. How would you do it?

The right answer is generating random integers between 0 and $N - 1$, then retrieving the elements at those indices and you have your answer. If you need to be generate unique elements, then just throw away indices you have already generated.

Introduction

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements, for example:

- Queries on Google searches in May
- Products bought at Walmart during the Christmas season

Your goal is to efficiently return a random sample of 1,000 elements evenly distributed from the original stream. How would you do it?

The right answer is generating random integers between 0 and $N-1$, then retrieving the elements at those indices and you have your answer. If you need to be generate unique elements, then just throw away indices you have already generated.

So, let me make the problem harder. You do not know N (the size of the stream) in advance. How would you do it?

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.
This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.
This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.
In many streaming cases, the data cannot be stored in our system.

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.

This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.

In many streaming cases, the data cannot be stored in our system.

There are several factors that have to be taken into account:

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.

This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.

In many streaming cases, the data cannot be stored in our system.

There are several factors that have to be taken into account:

- The size of the stream is unknown;

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.

This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.

In many streaming cases, the data cannot be stored in our system.

There are several factors that have to be taken into account:

- The size of the stream is unknown;
- You only access the data single pass;

Simple solution

A relatively easy and correct solution is to assign a random number to every element as you see it in the stream, and then always keep the top 1,000 numbered elements at all times.

This is similar to how a SQL Query with **ORDER BY RAND()** works. This strategy works well, and only requires additionally storing the randomly generated number for each element.

In many streaming cases, the data cannot be stored in our system.

There are several factors that have to be taken into account:

- The size of the stream is unknown;
- You only access the data single pass;
- All items have the same probability to be selected.

Reservoir Sampling

Another, more complex option is reservoir sampling.

Reservoir Sampling

Another, more complex option is reservoir sampling.

- you want to make a reservoir (array) of 1,000 elements and fill it with the first 1,000 elements in your stream. That way if you have exactly 1,000 elements, the algorithm works. This is the base case.

Reservoir Sampling

Another, more complex option is reservoir sampling.

- you want to make a reservoir (array) of 1,000 elements and fill it with the first 1,000 elements in your stream. That way if you have exactly 1,000 elements, the algorithm works. This is the base case.
- you want to process the i-th element (starting with $i = 1001$) such that at the end of processing that step, the 1,000 elements in your reservoir are randomly sampled amongst the elements you have seen so far.

Reservoir Sampling

Another, more complex option is reservoir sampling.

- you want to make a reservoir (array) of 1,000 elements and fill it with the first 1,000 elements in your stream. That way if you have exactly 1,000 elements, the algorithm works. This is the base case.
- you want to process the i-th element (starting with $i = 1001$) such that at the end of processing that step, the 1,000 elements in your reservoir are randomly sampled amongst the elements you have seen so far.

How can you do this?

Reservoir Sampling

Another, more complex option is reservoir sampling.

- you want to make a reservoir (array) of 1,000 elements and fill it with the first 1,000 elements in your stream. That way if you have exactly 1,000 elements, the algorithm works. This is the base case.
- you want to process the i-th element (starting with $i = 1001$) such that at the end of processing that step, the 1,000 elements in your reservoir are randomly sampled amongst the elements you have seen so far.

How can you do this?
Start with $i = 1001$. What probability after the 1001'th step should element 1,001 (or any element for that matter) be in the set of 1,000 elements?

Reservoir Sampling

Another, more complex option is reservoir sampling.

- you want to make a reservoir (array) of 1,000 elements and fill it with the first 1,000 elements in your stream. That way if you have exactly 1,000 elements, the algorithm works. This is the base case.
- you want to process the i-th element (starting with $i = 1001$) such that at the end of processing that step, the 1,000 elements in your reservoir are randomly sampled amongst the elements you have seen so far.

How can you do this?
Start with $i = 1001$. What probability after the 1001'th step should element 1,001 (or any element for that matter) be in the set of 1,000 elements? How to determine whether the 1001'th item is in the set of sample?

Reservoir Sampling

algorithm `reservoir`(k, S)
/* take k random samples from the dataset S */

initialize an array *samples* of size k
for $i = 1$ to $n = |S|$
    $o =$ the i-th item
    if $i \leq k$ **then**
        $samples[i] = o$
    else
        generate a random integer from 1 to x
        if $x \leq k$ **then**
            $samples[i] = o$

The reservoir algorithm is very efficient: it spends $O(1)$ time per item. Next, we will show that the algorithm is correct, namely:

Reservoir Sampling

algorithm reservoir(k, S)
/* take k random samples from the dataset S */

initialize an array *samples* of size k
for $i = 1$ to $n = |S|$
    $o =$ the i-th item
    if $i \le k$ **then**
        $samples[i] = o$
    else
        generate a random integer from 1 to x
        if $x \le k$ **then**
            $samples[i] = o$

The reservoir algorithm is very efficient: it spends $O(1)$ time per item. Next, we will show that the algorithm is correct, namely:

- Every item of S has the same probability of being sampled

Reservoir Sampling

```
algorithm reservoir(k, S)
/* take k random samples from the dataset S */
initialize an array samples of size k
for i = 1 to n = |S|
    o = the i-th item
    if i ≤ k then
        samples[i] = o
    else
        generate a random integer from 1 to x
        if x ≤ k then
            samples[i] = o
```

The reservoir algorithm is very efficient: it spends $O(1)$ time per item. Next, we will show that the algorithm is correct, namely:

- Every item of S has the same probability of being sampled
- For any two items o_1 and o_2, the events they are sampled are independent from each other.

Example

Let $S = \{59, 100, 2, 30, 63, \cdots\}$, and $k = 3$.

Example

Let $S = \{59, 100, 2, 30, 63, \cdots\}$, and $k = 3$.

- The first k items are directly added to the sample set. So samples $= (59, 100, 2)$.

Example

Let $S = \{59, 100, 2, 30, 63, \cdots\}$, and $k = 3$.

- The first k items are directly added to the sample set. So samples $= (59, 100, 2)$.
- Given the 4th item, the algorithm generates a random integer x from 1 to 4. Assume that the generated $x = 4$. As $x > k$, the item is ignored.

Example

Let $S = \{59, 100, 2, 30, 63, \cdots\}$, and $k = 3$.

- The first k items are directly added to the sample set. So samples $= (59, 100, 2)$.
- Given the 4th item, the algorithm generates a random integer x from 1 to 4. Assume that the generated $x = 4$. As $x > k$, the item is ignored.
- Given the 5th item, again, the algorithm generates x randomly, but now from 1 to 5. Assume that $x = 2$ this time. Hence, the item is added to samples, and replaces the 2nd value there. Hence, samples becomes $(59, 63, 2)$.

Example

Let $S = \{59, 100, 2, 30, 63, \cdots\}$, and $k = 3$.

- The first k items are directly added to the sample set. So samples $= (59, 100, 2)$.
- Given the 4th item, the algorithm generates a random integer x from 1 to 4. Assume that the generated $x = 4$. As $x > k$, the item is ignored.
- Given the 5th item, again, the algorithm generates x randomly, but now from 1 to 5. Assume that $x = 2$ this time. Hence, the item is added to samples, and replaces the 2nd value there. Hence, samples becomes $(59, 63, 2)$.
- The remaining items are processed in the same manner.

Proof

Theorem

After $n \geq k$ items in S have been processed, each of those items is sampled with probability $\frac{k}{n}$.

Proof.

Proof

Theorem

After $n \geq k$ items in S have been processed, each of those items is sampled with probability $\frac{k}{n}$.

Proof.

We prove the theorem by induction.

Proof

Theorem

After $n \geq k$ items in S have been processed, each of those items is sampled with probability $\frac{k}{n}$.

Proof.

We prove the theorem by induction.
Basic step: for $n = k$ the statement is obviously correct.

Proof

Theorem

After $n \geq k$ items in S have been processed, each of those items is sampled with probability $\frac{k}{n}$.

Proof.

We prove the theorem by induction.

Basic step: for $n = k$ the statement is obviously correct.

Inductive step: assuming the correctness for $n = m$, next we show that the statement is also correct for $n = m + 1$.

Proof

Theorem

After $n \geq k$ items in S have been processed, each of those items is sampled with probability $\frac{k}{n}$.

Proof.

We prove the theorem by induction.
Basic step: for $n = k$ the statement is obviously correct.
Inductive step: assuming the correctness for $n = m$, next we show that the statement is also correct for $n = m + 1$.

- The $(m+1)-$th object o is sampled if and only if the random number x generated for o falls in the range from 1 to k. Hence, o is sampled with probability $\frac{s}{m+1}$.

□

Proof Cont'd

Theorem

- o' is sampled (after processing o) if and only if (i) it was sampled after processing the first m items, and (ii) the random number x generated for o is not equivalent to the index value of o in the array samples.

By our inductive assumption,

Proof Cont'd

Theorem

- o' is sampled (after processing o) if and only if (i) it was sampled after processing the first m items, and (ii) the random number x generated for o is not equivalent to the index value of o in the array samples.

By our inductive assumption,

- (i) happens with probability $\frac{k}{m}$;

Proof Cont'd

Theorem

- o' is sampled (after processing o) if and only if (i) it was sampled after processing the first m items, and (ii) the random number x generated for o is not equivalent to the index value of o in the array samples.

By our inductive assumption,

- (i) happens with probability $\frac{k}{m}$;
- (ii) occurs with probability $\frac{m}{m+1}$.

Proof Cont'd

Theorem

- o' is sampled (after processing o) if and only if (i) it was sampled after processing the first m items, and (ii) the random number x generated for o is not equivalent to the index value of o in the array samples.

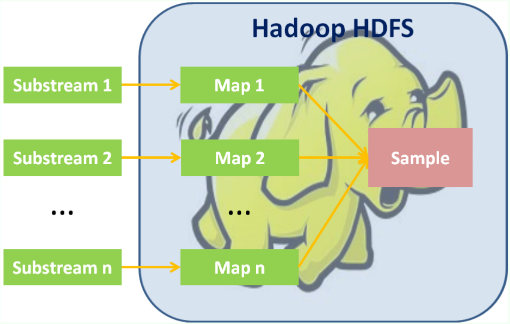
By our inductive assumption,

- (i) happens with probability $\frac{k}{m}$;
- (ii) occurs with probability $\frac{m}{m+1}$.

As the two events are independent, the probability that they happen simultaneously equals

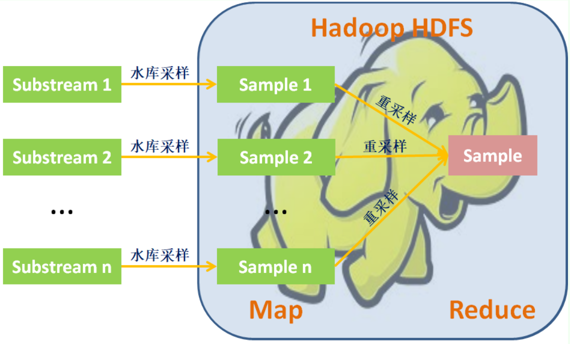
$$\frac{k}{m} \cdot \frac{m}{m+1} = \frac{k}{m+1}.$$

Application

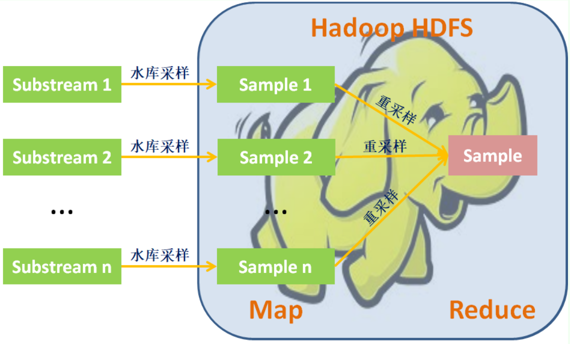


If I want to break break up the problem on say 10 machines and solve it close to 10 times faster, how can I do that?

Distributed reservoir sampling

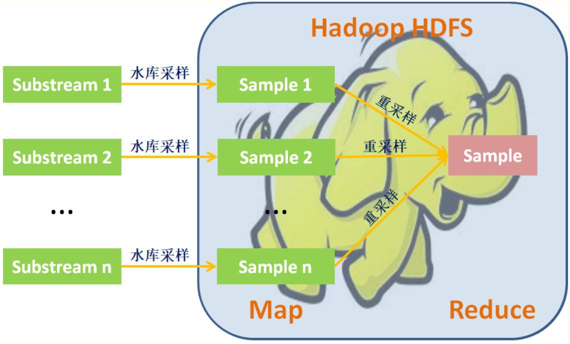


Distributed reservoir sampling



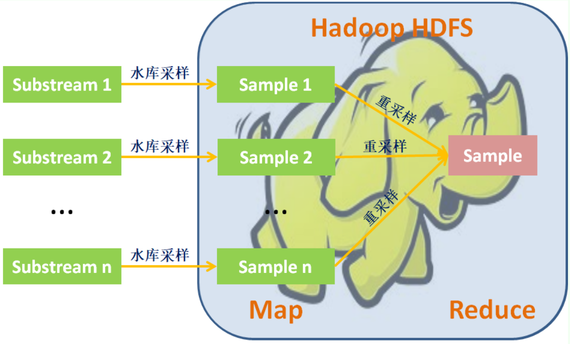
The answer is to have each of the 10 machines take roughly 1/10th of the input to process and generate their own reservoir sample from their subset of the data using the weighted variation above.

Distributed reservoir sampling



The answer is to have each of the 10 machines take roughly 1/10th of the input to process and generate their own reservoir sample from their subset of the data using the weighted variation above. Then, a final process must take the 10 output reservoirs and merge them.

Distributed reservoir sampling



The answer is to have each of the 10 machines take roughly 1/10th of the input to process and generate their own reservoir sample from their subset of the data using the weighted variation above. Then, a final process must take the 10 output reservoirs and merge them. Why?

Distributed reservoir sampling algorithm

Algorithm: Distributed reservoir sampling algorithm

Input : # Maps is n
Output: Sample H of size k

for *ith Map for* $1 \leq i \leq n$ **do**
    $F_i \leftarrow$ sample of k size in ith Map;
    $N_i \leftarrow$ the number of items in ith Map;
Initialize reservoir H;
for $1 \leq j \leq k$ **do**
    $p \leftarrow random(0, 1)$;
    Determine m s.t., $\sum_{i=1}^{m-1} N_i < p \sum_{i=1}^{n} N_i \leq \sum_{i=1}^{m} N_i$;
    Move an item from F_m into H;
return H;

Distributed reservoir sampling algorithm

Algorithm: Distributed reservoir sampling algorithm

Input : # Maps is n
Output: Sample H of size k

for ith Map for $1 \leq i \leq n$ **do**
  $F_i \leftarrow$ sample of k size in ith Map;
  $N_i \leftarrow$ the number of items in ith Map;
Initialize reservoir H;
for $1 \leq j \leq k$ **do**
  $p \leftarrow random(0, 1)$;
  Determine m s.t., $\sum_{i=1}^{m-1} N_i < p \sum_{i=1}^{n} N_i \leq \sum_{i=1}^{m} N_i$;
  Move an item from F_m into H;
return H;

The distributed reservoir sampling algorithm is very efficient: it spends $O(1)$ time per item.

Distributed reservoir sampling algorithm

Algorithm: Distributed reservoir sampling algorithm

Input : # Maps is n
Output: Sample H of size k

for *ith Map for* $1 \leq i \leq n$ **do**
    $F_i \leftarrow$ sample of k size in ith Map;
    $N_i \leftarrow$ the number of items in ith Map;
Initialize reservoir H;
for $1 \leq j \leq k$ **do**
    $p \leftarrow random(0, 1)$;
    Determine m s.t., $\sum_{i=1}^{m-1} N_i < p \sum_{i=1}^{n} N_i \leq \sum_{i=1}^{m} N_i$;
    Move an item from F_m into H;
return H;

The distributed reservoir sampling algorithm is very efficient: it spends $O(1)$ time per item. The algorithm is correct, namely:

Distributed reservoir sampling algorithm

Algorithm: Distributed reservoir sampling algorithm

Input : # Maps is n
Output: Sample H of size k

for *ith Map for* $1 \leq i \leq n$ **do**
  $F_i \leftarrow$ sample of k size in ith Map;
  $N_i \leftarrow$ the number of items in ith Map;
Initialize reservoir H;
for $1 \leq j \leq k$ **do**
  $p \leftarrow random(0, 1)$;
  Determine m s.t., $\sum_{i=1}^{m-1} N_i < p \sum_{i=1}^{n} N_i \leq \sum_{i=1}^{m} N_i$;
  Move an item from F_m into H;
return H;

The distributed reservoir sampling algorithm is very efficient: it spends $O(1)$ time per item. The algorithm is correct, namely:

- Every item of S has the same probability of being sampled

Distributed reservoir sampling algorithm

Algorithm: Distributed reservoir sampling algorithm

Input : # Maps is n
Output: Sample H of size k

for *ith Map for* $1 \le i \le n$ **do**
  $F_i \leftarrow$ sample of k size in ith Map;
  $N_i \leftarrow$ the number of items in ith Map;
Initialize reservoir H;
for $1 \le j \le k$ **do**
  $p \leftarrow random(0, 1)$;
  Determine m s.t., $\sum_{i=1}^{m-1} N_i < p \sum_{i=1}^{n} N_i \le \sum_{i=1}^{m} N_i$;
  Move an item from F_m into H;
return H;

The distributed reservoir sampling algorithm is very efficient: it spends $O(1)$ time per item. The algorithm is correct, namely:

- Every item of S has the same probability of being sampled
- For any two items o_1 and o_2, the events they are sampled are independent from each other.

Take-home messages

- Motivation of sampling
- Sampling
  - Simple Random Sampling
  - Systematic Sampling
  - Stratified Sampling
  - Reservoir Sampling