

# R Compute Gini Coefficient for Discrete Samples

Fan Wang

2020-04-14

## Contents

<b>1 Gini Discrete Sample</b>	<b>1</b>
1.1 Gini Formula for Discrete Sample . . . . .	1
1.2 Implement GINI Formula . . . . .	2

## 1 Gini Discrete Sample

Go to the [RMD](#), [R](#), [PDF](#), or [HTML](#) version of this file. Go back to [fan's REconTools Package](#), [R Code Examples Repository \(bookdown site\)](#), or [Intro Stats with R Repository \(bookdown site\)](#).

This works out how the `ff_dist_gini_vector_pos` function works from [Fan's REconTools Package](#).

### 1.1 Gini Formula for Discrete Sample

There is an vector values (all positive). This could be height information for N individuals. It could also be income information for N individuals. Calculate the [GINI](#) coefficient treating the given vector as population. This is not an estimation exercise where we want to estimate population gini based on a sample. The given array is the population. The population is discrete, and only has these N individuals in the length n vector.

Note that when the sample size is small, there is a limit to inequality using the formula defined below given each  $N$ . So for small  $N$ , can not really compare inequality across arrays with different  $N$ , can only compare arrays with the same  $N$ .

The GINI formula used here is:

$$GINI = 1 - \frac{2}{N+1} \cdot \left( \sum_{i=1}^N \sum_{j=1}^i x_j \right) \cdot \left( \sum_{i=1}^N x_i \right)^{-1}$$

Derive the formula in the steps below.

*Step 1 Area Formula*

$$\Gamma = \sum_{i=1}^N \frac{1}{N} \cdot \left( \sum_{j=1}^i \left( \frac{x_j}{\sum_{j=1}^N x_j} \right) \right)$$

*Step 2 Total Area Given Perfect equality*

With perfect equality  $x_i = a$  for all  $i$ , so need to divide by that.

$$\Gamma^{\text{equal}} = \sum_{i=1}^N \frac{1}{N} \cdot \left( \sum_{j=1}^i \left( \frac{a}{\sum_{j=1}^N a} \right) \right) = \frac{N+1}{N} \cdot \frac{1}{2}$$

As the number of elements of the vecotr increases:

$$\lim_{N \rightarrow \infty} \Gamma^{\text{equal}} = \lim_{N \rightarrow \infty} \frac{N+1}{N} \cdot \frac{1}{2} = \frac{1}{2}$$

### Step 3 Arriving at Finite Vector Gini Formula

Given what we have from above, we obtain the gini formula, divide by total area below 45 degree line.

$$GINI = 1 - \left( \sum_{i=1}^N \sum_{j=1}^i x_j \right) \cdot \left( N \cdot \sum_{i=1}^N x_i \right)^{-1} \cdot \left( \frac{N+1}{N} \cdot \frac{1}{2} \right)^{-1} = 1 - \frac{2}{N+1} \cdot \left( \sum_{i=1}^N \sum_{j=1}^i x_j \right) \cdot \left( \sum_{i=1}^N x_i \right)^{-1}$$

### Step 4 Maximum Inequality given N

Suppose  $x_i = 0$  for all  $i < N$ , then:

$$GINI^{x_i=0 \text{ except } i=N} = 1 - \frac{2}{N+1} \cdot X_N \cdot (X_N)^{-1} = 1 - \frac{2}{N+1}$$

$$\lim_{N \rightarrow \infty} GINI^{x_i=0 \text{ except } i=N} = 1 - \lim_{N \rightarrow \infty} \frac{2}{N+1} = 1$$

Note that for small N, for example if  $N = 10$ , even when one person holds all income, all others have 0 income, the formula will not produce gini is zero, but that gini is equal to  $\frac{2}{11} \approx 0.1818$ . If  $N = 2$ , inequality is at most,  $\frac{2}{3} \approx 0.667$ .

$$\text{MostUnequalGINI}(N) = 1 - \frac{2}{N+1} = \frac{N-1}{N+1}$$

## 1.2 Implement GINI Formula

The **GINI** formula just derived is trivial to compute.

1. scalar:  $\frac{2}{N+1}$
2. cumsum:  $\sum_{j=1}^i x_j$
3. sum of cumsum:  $\left( \sum_{i=1}^N \sum_{j=1}^i x_j \right)$
4. sum:  $\sum_{i=1}^N X_i$

There are no package dependencies. Define the formula here:

```
# Formula, directly implement the GINI formula Following Step 4 above
fv_dist_gini_vector_pos_test <- function(ar_pos) {
  # Check length and given warning
  it_n <- length(ar_pos)
  if (it_n <= 100) warning('Data vector has n=', it_n, ', max-inequality/max-gini=', (it_n-1)/(it_n + 1))
  # Sort
  ar_pos <- sort(ar_pos)
  # formula implement
  fl_gini <- 1 - ((2/(it_n+1)) * sum(cumsum(ar_pos)) * (sum(ar_pos)) ^ (-1))
  return(fl_gini)
}
```

Generate a number of examples Arrays for testing

```

# Example Arrays of data
ar_equal_n1 = c(1)
ar_ineql_n1 = c(100)

ar_equal_n2 = c(1,1)
ar_ineql_alittle_n2 = c(1,2)
ar_ineql_somewht_n2 = c(1,2^3)
ar_ineql_alotine_n2 = c(1,2^5)
ar_ineql_veryvry_n2 = c(1,2^8)
ar_ineql_mostmst_n2 = c(1,2^13)

ar_equal_n10 = c(2,2,2,2,2,2, 2, 2, 2, 2)
ar_ineql_some_n10 = c(1,2,3,5,8,13,21,34,55,89)
ar_ineql_very_n10 = c(1,2^2,3^2,5^2,8^2,13^2,21^2,34^2,55^2,89^2)
ar_ineql_extr_n10 = c(1,2^2,3^3,5^4,8^5,13^6,21^7,34^8,55^9,89^10)

```

Now test the example arrays above using the function based no our formula:

```

##
## Small N=1 Hard-Code
## ar_equal_n1: 0
## ar_ineql_n1: 0
##
## Small N=2 Hard-Code, converge to 1/3, see formula above
## ar_ineql_alittle_n2: 0.1111111
## ar_ineql_somewht_n2: 0.2592593
## ar_ineql_alotine_n2: 0.3131313
## ar_ineql_veryvry_n2: 0.3307393
##
## Small N=10 Hard-Code, convege to 9/11=0.8181, see formula above
## ar_equal_n10: 0
## ar_ineql_some_n10: 0.5395514
## ar_ineql_very_n10: 0.7059554
## ar_ineql_extr_n10: 0.8181549

```