R Compute Gini Coefficient for Discrete Samples or Discrete Random Variable

Fan Wang

2021-03-18

Contents

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

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

Step 2 Total Area Given Perfect equality

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

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

As the number of elements of the vecotr increases:

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

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:

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

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

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, $\frac{2}{3} \approx 0.667$.

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

1.1.1 Implement GINI Formula for Discrete Sample

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
ffi_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: 0.1111

## ar_ineql_somewht_n2: 0.2593

## ar_ineql_alotine_n2: 0.3131

## ar_ineql_veryvry_n2: 0.3307

##
## Small N=10 Hard-Code, convege to 9/11=0.8181, see formula above

## ar_equal_n10: 0

## ar_ineql_some_n10: 0.5396

## ar_ineql_very_n10: 0.706

## ar_ineql_extr_n10: 0.8182
```

2 Gini Formula for Discrete Random Varialbe

For a discrete random variable, we are two arrays, an array of x values, and an array of $f(x)$ probability mass at each x value. Suppose the x values are unique/non-repeating. This is also Implemented in MEconTools with the ff_disc_rand_var_gini function.

Generate two arrays for x and $f(x)$, we will use the binomial distribution:

```
ar_choice_unique_sorted <- seq(0, 100, by=1)
ar_choice_prob <- dbinom(ar_choice_unique_sorted, 100, 0.01)
```

Generate mean and cumulative mean at each point:

```
# 1. to normalize, get mean (binomial so mean is p*N=50)
fl_mean <- sum(ar_choice_unique_sorted*ar_choice_prob);
# 2. get cumulative mean at each point
ar_mean_cumsum <- cumsum(ar_choice_unique_sorted*ar_choice_prob);
```

Normalizing and area calculation, following the same principle as above:

```
# 3. Share of wealth (income etc) accounted for up to this sorted type
ar_height <- ar_mean_cumsum/fl_mean;
# 4. The total area, is the each height times each width summed up
fl_area_drm <- sum(ar_choice_prob*ar_height);
```

Finally GINI coefficient:

```
# 5. area below 45 degree line might not be 1/2, depends on discretness
fl_area_below45 <- sum(ar_choice_prob*(cumsum(ar_choice_prob)/sum(ar_choice_prob)))

# 6. Gini is the distance between
fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45
print(paste0('fl_gini_index=', fl_gini_index))
```

```
## [1] "fl_gini_index=0.468573066002754"
```

2.1 Discrete Random Variable as Function

Organizing the code above as a function, and testing results out with the binomial distribution as an example.

For the binomial distribution, if the probability of success is very close to zero, that means nearly all mass is at lose all or nearly losing all. There will be non-zero but very small mass at higher levels of wins. Hence this should mean extreme inequality. GINI index should be close to 1. Alternatively, GINI index should be close to 0 when we have near 100 percent chance of success, then all mass is at winning all, perfect equality.

```
# Combining the code from above
ffi_dist_gini_random_var_pos_test <- function(ar_x_sorted, ar_prob_of_x) {
  # Check length and given warning

  # 1. to normalize, get mean (binomial so mean is p*N=50)
  fl_mean <- sum(ar_x_sorted*ar_prob_of_x);
  # 2. get cumulative mean at each point
  ar_mean_cumsum <- cumsum(ar_x_sorted*ar_prob_of_x);
  # 3. Share of wealth (income etc) accounted for up to this sorted type
  ar_height <- ar_mean_cumsum/fl_mean;
  # 4. The total area, is the each height times each width summed up
  fl_area_drm <- sum(ar_prob_of_x*ar_height);
  # 5. area below 45 degree line might not be 1/2, depends on discretness
  fl_area_below45 <- sum(ar_prob_of_x*(cumsum(ar_prob_of_x)/sum(ar_prob_of_x)))
  # 6. Gini is the distance between
  fl_gini_index <- (fl_area_below45-fl_area_drm)/fl_area_below45

  return(fl_gini_index)
}
```

Testing the function with the Binomial Distribution:

```
for (fl_binom_success_prob in seq(0.0001,0.9999,length.out=10)) {
  ar_x_sorted <- seq(0, 100, by=1)
  ar_prob_of_x <- dbinom(ar_x_sorted, 100, fl_binom_success_prob)
  fl_gini_index <- ffi_dist_gini_random_var_pos_test(ar_x_sorted, ar_prob_of_x)
  st_print <- paste0('binom p(success)=', fl_binom_success_prob ,
                     ', the fl_gini_index=', fl_gini_index)
  print(st_print)
}
```

```
## [1] "binom p(success)=0.0001, the fl_gini_index=0.990048846889393"
## [1] "binom p(success)=0.111188888888889, the fl_gini_index=0.147061101509638"
## [1] "binom p(success)=0.222277777777778, the fl_gini_index=0.0989942681255604"
## [1] "binom p(success)=0.333366666666667, the fl_gini_index=0.0753059593394789"
## [1] "binom p(success)=0.444455555555556, the fl_gini_index=0.0596495299535176"
## [1] "binom p(success)=0.555544444444444, the fl_gini_index=0.0476678493269222"
## [1] "binom p(success)=0.666633333333333, the fl_gini_index=0.0375168214334586"
## [1] "binom p(success)=0.777722222222222, the fl_gini_index=0.0280646430085938"
## [1] "binom p(success)=0.888811111111111, the fl_gini_index=0.0180312757603542"
## [1] "binom p(success)=0.9999, the fl_gini_index=0.000000990197372312816"
```

2.2 Compare Discrete Sample and Discrete Random Variable Functions for GINI

ff_dist_gini_random_var provides the GINI implementation for a discrete random variable. The procedure is the same as prior, except now each element of the "x" array has element specific weights associated with it. The function can handle unsorted array with non-unique values.

Test and compare ff_dist_gini_random_var provides the GINI implementation for a discrete random variable and ff_dist_gini_vector_pos.

There is a vector of values from 1 to 100, in ascending order. What is the equal-weighted gini, the gini result when smaller numbers have higher weights, and when larger numbers have higher weights?

First, generate the relevant values.

```
# array
ar_x <- seq(1, 100, length.out = 30)
# prob array
ar_prob_x_unif <- rep.int(1, length(ar_x))/sum(rep.int(1, length(ar_x)))
# prob higher at lower values
ar_prob_x_lowval_highwgt <- rev(cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# prob higher at lower values
ar_prob_x_highval_highwgt <- (cumsum(ar_prob_x_unif))/sum(cumsum(ar_prob_x_unif))
# show
kable(cbind(ar_x, ar_prob_x_unif, ar_prob_x_lowval_highwgt, ar_prob_x_highval_highwgt)) %>%
  kable_styling_fc()
```

Second, generate GINI values. What should happen?

1. The ff_dist_gini_random_var and ff_dist_gini_vector_pos results should be the same when the uniform distribution is used.
2. GINI should be higher, more inequality, if there is higher weights on the lower values.
3. GINI should be lower, more equality, if there is higher weight on the higher values.

```
ff_dist_gini_vector_pos(ar_x)
```

ar_x	ar_prob_x_unif	ar_prob_x_lowval_highwgt	ar_prob_x_highval_highwgt
1.000	0.0333	0.0645	0.0022
4.414	0.0333	0.0624	0.0043
7.828	0.0333	0.0602	0.0065
11.241	0.0333	0.0581	0.0086
14.655	0.0333	0.0559	0.0108
18.069	0.0333	0.0538	0.0129
21.483	0.0333	0.0516	0.0151
24.897	0.0333	0.0495	0.0172
28.310	0.0333	0.0473	0.0194
31.724	0.0333	0.0452	0.0215
35.138	0.0333	0.0430	0.0237
38.552	0.0333	0.0409	0.0258
41.965	0.0333	0.0387	0.0280
45.379	0.0333	0.0366	0.0301
48.793	0.0333	0.0344	0.0323
52.207	0.0333	0.0323	0.0344
55.621	0.0333	0.0301	0.0366
59.035	0.0333	0.0280	0.0387
62.448	0.0333	0.0258	0.0409
65.862	0.0333	0.0237	0.0430
69.276	0.0333	0.0215	0.0452
72.690	0.0333	0.0194	0.0473
76.103	0.0333	0.0172	0.0495
79.517	0.0333	0.0151	0.0516
82.931	0.0333	0.0129	0.0538
86.345	0.0333	0.0108	0.0559
89.759	0.0333	0.0086	0.0581
93.172	0.0333	0.0065	0.0602
96.586	0.0333	0.0043	0.0624
100.000	0.0333	0.0022	0.0645

```
## [1] 0.3267
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_unif)
```

```
## [1] 0.3267
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_lowval_highwgt)
```

```
## [1] 0.401
```

```
ff_dist_gini_random_var(ar_x, ar_prob_x_highval_highwgt)
```

```
## [1] 0.1927
```