R dplyr Evaluate Function Do Anything Group Stack Results

Fan Wang

2020-04-01

Contents

1 MxQ to MxP Rows **1**
1.1 MxQ to Mx1 Rows: Within Group Gini 1
1.1.1 Large Dataframe 1
1.1.2 Compute Group specific gini, NORMAL 2

1 MxQ to MxP Rows

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).

1.1 MxQ to Mx1 Rows: Within Group Gini

There is a Panel with M individuals and each individual has Q records/rows. A function generate an individual specific outcome given the Q individual specific inputs, along with shared parameters and arrays across the M individuals.

For example, suppose we have a dataframe of individual wage information from different countries, each row is an individual from one country. We want to generate country specific gini based on the individual data for each country in the dataframe. But additionally, perhaps the gini formula requires not just individual income but some additional parameters or shared dataframes as inputs.

Given the within m income observations, we can compute gini statistics that are individual specific based on the observed distribution of incomes. For this, we will use the ff_dist_gini_vector_pos.html function from REconTools.

To make this more interesting, we will generate large dataframe with more M and more Q each m.

1.1.1 Large Dataframe

There are up to ten thousand income observation per person. And there are ten people.

```
# Parameter Setups
it_M <- 10
it_Q_max <- 10000
fl_rnorm_mu <- 1
ar_rnorm_sd <- seq(0.01, 0.2, length.out=it_M)
ar_it_q <- sample.int(it_Q_max, it_M, replace=TRUE)

# N by Q varying parameters
mt_data = cbind(ar_it_q, ar_rnorm_sd)
tb_M <- as_tibble(mt_data) %>% rowid_to_column(var = "ID") %>%
```

```
                rename(sd = ar_rnorm_sd, Q = ar_it_q) %>%
                mutate(mean = fl_rnorm_mu)
```

1.1.2 Compute Group specific gini, NORMAL

There is only one input for the gini function *ar_pos*. Note that the gini are not very large even with large SD, because these are normal distributions. By Construction, most peple are in the middle. So with almost zero standard deviation, we have perfect equality, as standard deviation increases, inequality increases, but still pretty equal overall, there is no fat upper tail.

Note that there are three ways of referring to variable names with dot, which are all shown below:

1. We can explicitly refer to names
2. We can use the dollar dot structure to use string variable names in do anything.
3. We can use dot bracket, this is the only option that works with string variable names

```
# A. Normal Draw Expansion, Explicitly Name
set.seed('123')
tb_income_norm_dot_dollar <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.$Q,
                    mean=.$mean,
                    sd=.$sd)) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot dollar differently with string variable name
set.seed('123')
tb_income_norm_dollar_dot <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(`$`(., 'Q'),
                    mean = `$`(., 'mean'),
                    sd = `$`(., 'sd'))) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# Normal Draw Expansion again, dot double bracket
set.seed('123')
svr_mean <- 'mean'
svr_sd <- 'sd'
svr_Q <- 'Q'
tb_income_norm_dot_bracket_db <- tb_M %>% group_by(ID) %>%
  do(income = rnorm(.[[svr_Q]],
                    mean = .[[svr_mean]],
                    sd = .[[svr_sd]])) %>%
  unnest(c(income)) %>%
  left_join(tb_M, by="ID")

# display
sum(sum(tb_income_norm_dollar_dot - tb_income_norm_dot_dollar - tb_income_norm_dot_bracket_db))
```

```
## [1] -493416029
```

```
# display
head(tb_income_norm_dot_dollar, 20)
```

```
# Gini by Group
tb_gini_norm <- tb_income_norm_dollar_dot %>% group_by(ID) %>%
```

```r
  do(inc_gini_norm = ff_dist_gini_vector_pos(.$income)) %>%
  unnest(c(inc_gini_norm)) %>%
  left_join(tb_M, by="ID")

# display
kable(tb_gini_norm) %>%
  kable_styling_fc()
```

ID	inc_gini_norm	Q	sd	mean
1	0.0055921	5603	0.0100000	1
2	0.0176426	9693	0.0311111	1
3	0.0294306	4576	0.0522222	1
4	0.0415049	3783	0.0733333	1
5	0.0532419	7831	0.0944444	1
6	0.0651498	5967	0.1155556	1
7	0.0776071	9301	0.1366667	1
8	0.0887634	7816	0.1577778	1
9	0.1018649	9267	0.1788889	1
10	0.1134874	1386	0.2000000	1