Joint Quantiles from Multiple Continuous Variables as a Categorical Variable with Linear Index

Fan Wang

2020-04-01

Contents

Joint Quantiles from Continuous

> Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools Package, R4Econ Repository (bookdown site), or Intro Stats with R Repository.

There are multiple or a single continuous variables. Find which quantile each observation belongs to for each of the variables. Then also generate a joint/interaction variable of all combinations of quantiles from different variables.

The program has these features:

1. Quantiles breaks are generated based on group_by characteristics, meaning quantiles for individual level characteristics when data is panel
2. Quantiles variables apply to full panel at within-group observation levels.
3. Robust to non-unique breaks for quantiles (non-unique grouped together)
4. Quantile categories have detailed labeling (specifying which non-unique groupings belong to quantile)

When joining multiple quantile variables together:

1. First check if only calculate quantiles at observations where all quantile base variables are not null
2. Calculate Quantiles for each variable, with different quantile levels for sub-groups of variables
3. Summary statistics by mulltiple quantile-categorical variables, summary

Build Program

```
# Quantiles for any variable
gen_quantiles <- function(var, df, prob=c(0.25, 0.50, 0.75)) {
    enframe(quantile(as.numeric(df[[var]]), prob, na.rm=TRUE), 'quant.perc', var)
}
# Support Functions for Variable Suffix
f_Q_suffix <- function(seq.quantiles) {
    quantile.suffix <- paste0('Qs', min(seq.quantiles),
                              'e', max(seq.quantiles),
                              'n', (length(seq.quantiles)-1))
}
# Support Functions for Quantile Labeling
f_Q_label <- function(arr.quantiles,
                      arr.sort.unique.quantile,
```

```
                        seq.quantiles) {
    paste0('(',
           paste0(which(arr.quantiles %in% arr.sort.unique.quantile), collapse=','),
           ') of ', f_Q_suffix(seq.quantiles))
}
# Generate New Variable Names with Quantile Suffix
f_var_rename <- function(name, seq.quantiles) {
    quantile.suffix <- paste0('_', f_Q_suffix(seq.quantiles))
    return(sub('_q', quantile.suffix, name))
}

# Check Are Values within Group By Unique? If not, STOP
f_check_distinct_ingroup <- function(df, vars.group_by, vars.values_in_group) {

    df.uniqus.in.group <- df %>% group_by(!!!syms(vars.group_by)) %>%
            mutate(quant_vars_paste = paste(!!!(syms(vars.values_in_group)), sep='-')) %>%
            mutate(unique_in_group = n_distinct(quant_vars_paste)) %>%
            slice(1L) %>%
            ungroup() %>%
            group_by(unique_in_group) %>%
            summarise(n=n())

    if (sum(df.uniqus.in.group$unique_in_group) > 1) {
        print(df.uniqus.in.group)
        print(paste('vars.values_in_group', vars.values_in_group, sep=':'))
        print(paste('vars.group_by', vars.group_by, sep=':'))
        stop("The variables for which quantiles are to be taken are not identical within the group varia
    }
}
```

Support Functions

Data Slicing and Quantile Generation

- Function 1: generate quantiles based on group-specific characteristics. the groups could be at the panel observation level as well.

```
# First Step, given groups, generate quantiles based on group characteristics
# vars.cts2quantile <- c('wealthIdx', 'hgt0', 'wgt0')
# seq.quantiles <- c(0, 0.3333, 0.6666, 1.0)
# vars.group_by <- c('indi.id')
# vars.arrange <- c('indi.id', 'svymthRound')
# vars.continuous <- c('wealthIdx', 'hgt0', 'wgt0')
df_sliced_quantiles <- function(df, vars.cts2quantile, seq.quantiles,
                                vars.group_by, vars.arrange) {

    # Slicing data
    df.grp.L1 <- df %>% group_by(!!!syms(vars.group_by)) %>% arrange(!!!syms(vars.arrange)) %>% slice(1

    # Quantiles based on sliced data
    df.sliced.quantiles <- lapply(vars.cts2quantile, gen_quantiles, df=df.grp.L1, prob=seq.quantiles) %

    return(list(df.sliced.quantiles=df.sliced.quantiles,
                df.grp.L1=df.grp.L1))
}
```

Data Cutting

- Function 2: cut groups for full panel dataframe based on group-specific characteristics quantiles.

```
# Cutting Function, Cut Continuous Variables into Quantiles with labeing
f_cut <- function(var, df.sliced.quantiles, seq.quantiles, include.lowest=TRUE, fan.labels=TRUE, print=

    # unparsed string variable name
    var.str <- substitute(var)

    # Breaks
    arr.quantiles <- df.sliced.quantiles[[var.str]]
    arr.sort.unique.quantiles <- sort(unique(arr.quantiles))
    if (print) {
        print(arr.sort.unique.quantiles)
    }

    # Regular cutting With Standard Labels
    # TRUE, means the lowest group has closed bracket left and right
    var.quantile <- cut(var, breaks=arr.sort.unique.quantiles, include.lowest=include.lowest)

    # Use my custom labels
    if (fan.labels) {
        levels.suffix <- lapply(arr.sort.unique.quantiles[1:(length(arr.sort.unique.quantiles)-1)],
                                f_Q_label,
                                arr.quantiles=arr.quantiles,
                                seq.quantiles=seq.quantiles)
        if (print) {
            print(levels.suffix)
        }
        levels(var.quantile) <- paste0(levels(var.quantile), '; ', levels.suffix)
    }

    # Return
    return(var.quantile)
}
```

```
# Combo Quantile Function
# vars.cts2quantile <- c('wealthIdx', 'hgt0', 'wgt0')
# seq.quantiles <- c(0, 0.3333, 0.6666, 1.0)
# vars.group_by <- c('indi.id')
# vars.arrange <- c('indi.id', 'svymthRound')
# vars.continuous <- c('wealthIdx', 'hgt0', 'wgt0')
df_cut_by_sliced_quantiles <- function(df, vars.cts2quantile, seq.quantiles,
                                       vars.group_by, vars.arrange) {

    # Check Are Values within Group By Unique? If not, STOP
    f_check_distinct_ingroup(df, vars.group_by, vars.values_in_group=vars.cts2quantile)

    # First Step Slicing
    df.sliced <- df_sliced_quantiles(df, vars.cts2quantile, seq.quantiles, vars.group_by, vars.arrange)

    # Second Step Generate Categorical Variables of Quantiles
    df.with.cut.quant <- df %>% mutate_at(vars.cts2quantile,
                                funs(q=f_cut(., df.sliced$df.sliced.quantiles,
```

```
                                            seq.quantiles=seq.quantiles,
                                            include.lowest=TRUE, fan.labels=TRUE)))

    if (length(vars.cts2quantile) > 1) {
        df.with.cut.quant <- df.with.cut.quant %>%
                             rename_at(vars(contains('_q')),
                                       funs(f_var_rename(., seq.quantiles=seq.quantiles)))
    } else {
        new.var.name <- paste0(vars.cts2quantile[1], '_', f_Q_suffix(seq.quantiles))
        df.with.cut.quant <- df.with.cut.quant %>% rename(!!new.var.name := q)
    }

    # Newly Generated Quantile-Cut Variables
    vars.quantile.cut <- df.with.cut.quant %>%
                select(matches(paste0(vars.cts2quantile, collapse='|'))) %>%
                select(matches(f_Q_suffix(seq.quantiles)))

    # Return
    return(list(df.with.cut.quant = df.with.cut.quant,
                df.sliced.quantiles=df.sliced$df.sliced.quantiles,
                df.grp.L1=df.sliced$df.grp.L1,
                vars.quantile.cut=vars.quantile.cut))

}
```

Different Vars Different Probabilities Joint Quantiles

- Accomondate multiple continuousv ariables
- Different percentiles
- list of lists
- generate joint categorical variables
- keep only values that exist for all quantile base vars

```
# Function to handle list inputs with different quantiles vars and probabilities
df_cut_by_sliced_quantiles_grps <- function(quantile.grp.list, df, vars.group_by, vars.arrange) {
   vars.cts2quantile <- quantile.grp.list$vars
   seq.quantiles <- quantile.grp.list$prob
   return(df_cut_by_sliced_quantiles(df, vars.cts2quantile, seq.quantiles, vars.group_by, vars.arrange))
}
# Show Results
df_cut_by_sliced_quantiles_joint_results_grped <- function(df.with.cut.quant.all, vars.cts2quantile, va
                                                           vars.quantile.cut.all, var.qjnt.grp.idx) {
    # Show ALL
    df.group.panel.cnt.mean <- df.with.cut.quant.all %>% group_by(!!!syms(vars.quantile.cut.all), !!sym
            summarise_at(vars.cts2quantile, funs(mean, n()))

    # Show Based on SLicing first
    df.group.slice1.cnt.mean <- df.with.cut.quant.all %>% group_by(!!!syms(vars.group_by)) %>% arrange(
            group_by(!!!syms(vars.quantile.cut.all), !!sym(var.qjnt.grp.idx)) %>%
            summarise_at(vars.cts2quantile, funs(mean, n()))

    return(list(df.group.panel.cnt.mean=df.group.panel.cnt.mean,
                df.group.slice1.cnt.mean=df.group.slice1.cnt.mean))
}
```

```r
# # Joint Quantile Group Name
# var.qjnt.grp.idx <- 'group.index'
# # Generate Categorical Variables of Quantiles
# vars.group_by <- c('indi.id')
# vars.arrange <- c('indi.id', 'svymthRound')
# # Quantile Variables and Quantiles
# vars.cts2quantile.wealth <- c('wealthIdx')
# seq.quantiles.wealth <- c(0, .5, 1.0)
# vars.cts2quantile.wgthgt <- c('hgt0', 'wgt0')
# seq.quantiles.wgthgt <- c(0, .3333, 0.6666, 1.0)
# drop.any.quantile.na <- TRUE
# # collect to list
# list.cts2quantile <- list(list(vars=vars.cts2quantile.wealth,
#                                prob=seq.quantiles.wealth),
#                           list(vars=vars.cts2quantile.wgthgt,
#                                prob=seq.quantiles.wgthgt))

df_cut_by_sliced_quantiles_joint <- function(df, var.qjnt.grp.idx,
                                             list.cts2quantile,
                                             vars.group_by, vars.arrange,
                                             drop.any.quantile.na = TRUE,
                                             toprint = TRUE) {

  #  Original dimensions
  if(toprint) {
   print(dim(df))
  }

  # All Continuous Variables from lists
  vars.cts2quantile <- unlist(lapply(list.cts2quantile, function(elist) elist$vars))
  vars.cts2quantile

  # Keep only if not NA for all Quantile variables
  if (drop.any.quantile.na) {
   df.select <- df %>% drop_na(c(vars.group_by, vars.arrange, vars.cts2quantile))
  } else {
   df.select <- df
  }

  if(toprint) {
   print(dim(df.select))
  }

  # Apply qunatile function to all elements of list of list
  df.cut.list <- lapply(list.cts2quantile, df_cut_by_sliced_quantiles_grps,
                        df=df.select, vars.group_by=vars.group_by, vars.arrange=vars.arrange)

  # Reduce Resulting Core Panel Matrix Together
  df.with.cut.quant.all <- lapply(df.cut.list, function(elist) elist$df.with.cut.quant) %>% reduce(left_
  df.sliced.quantiles.all <- lapply(df.cut.list, function(elist) elist$df.sliced.quantiles)

  if(toprint) {
    print(dim(df.with.cut.quant.all))
```

```
  }

  # Obrain Newly Created Quantile Group Variables
  vars.quantile.cut.all <- unlist(lapply(df.cut.list, function(elist) names(elist$vars.quantile.cut)))
  if(toprint) {
    print(vars.quantile.cut.all)
    print(summary(df.with.cut.quant.all %>% select(one_of(vars.quantile.cut.all))))
  }

  # Generate Joint Quantile Index Variable
  df.with.cut.quant.all <- df.with.cut.quant.all %>% mutate(!!var.qjnt.grp.idx := group_indices(., !!!sy

  # Quantile Groups
  arr.group.idx <- t(sort(unique(df.with.cut.quant.all[[var.qjnt.grp.idx]])))

  # Results Display
  df.group.print <- df_cut_by_sliced_quantiles_joint_results_grped(df.with.cut.quant.all, vars.cts2quan
                                                   vars.group_by, vars.arrange,
                                                   vars.quantile.cut.all, var.qjnt.grp.idx)

  # list to Return
  # These returns are the same as returns earlier: df_cut_by_sliced_quantiles
  # Except that they are combined together
  return(list(df.with.cut.quant = df.with.cut.quant.all,
              df.sliced.quantiles = df.sliced.quantiles.all,
              df.grp.L1 = (df.cut.list[[1]])$df.grp.L1,
              vars.quantile.cut = vars.quantile.cut.all,
              df.group.panel.cnt.mean = df.group.print$df.group.panel.cnt.mean,
              df.group.slice1.cnt.mean = df.group.print$df.group.slice1.cnt.mean))

}
```

Program Testing Load Data

```
# Library
library(tidyverse)

# Load Sample Data
setwd('C:/Users/fan/R4Econ/_data/')
df <- read_csv('height_weight.csv')
```

```
## Parsed with column specification:
## cols(
##   S.country = col_character(),
##   vil.id = col_double(),
##   indi.id = col_double(),
##   sex = col_character(),
##   svymthRound = col_double(),
##   momEdu = col_double(),
##   wealthIdx = col_double(),
##   hgt = col_double(),
##   wgt = col_double(),
##   hgt0 = col_double(),
##   wgt0 = col_double(),
```

```
##   prot = col_double(),
##   cal = col_double(),
##   p.A.prot = col_double(),
##   p.A.nProt = col_double()
## )
```

```
# Joint Quantile Group Name
var.qjnt.grp.idx <- 'group.index'
list.cts2quantile <- list(list(vars=c('hgt0'), prob=c(0, .3333, 0.6666, 1.0)))
results <- df_cut_by_sliced_quantiles_joint(df, var.qjnt.grp.idx, list.cts2quantile,
                                            vars.group_by = c('indi.id'), vars.arrange = c('indi.id', '
                                            drop.any.quantile.na = TRUE, toprint = FALSE)
# Show Results
results$df.group.slice1.cnt.mean
```

Hgt0 3 Groups

```
## # A tibble: 3 x 4
## # Groups:   hgt0_Qs0e1n3 [3]
##   hgt0_Qs0e1n3                group.index  mean     n
##   <fct>                             <int> <dbl> <int>
## 1 [40.6,48.5]; (1) of Qs0e1n3           1  47.0   580
## 2 (48.5,50.2]; (2) of Qs0e1n3           2  49.4   561
## 3 (50.2,58]; (3) of Qs0e1n3             3  51.7   568
```

```
# Joint Quantile Group Name
var.qjnt.grp.idx <- 'wltQuintle.index'
list.cts2quantile <- list(list(vars=c('wealthIdx'), prob=seq(0, 1.0, 0.20)))
results <- df_cut_by_sliced_quantiles_joint((df %>% filter(S.country == 'Guatemala')),
                                            var.qjnt.grp.idx, list.cts2quantile,
                                            vars.group_by = c('indi.id'), vars.arrange = c('indi.id', '
                                            drop.any.quantile.na = TRUE, toprint = FALSE)
# Show Results
results$df.group.slice1.cnt.mean
```

Wealth 5 Groups Guatemala

```
## # A tibble: 5 x 4
## # Groups:   wealthIdx_Qs0e1n5 [5]
##   wealthIdx_Qs0e1n5         wltQuintle.index  mean     n
##   <fct>                                <int> <dbl> <int>
## 1 [1,1.6]; (1) of Qs0e1n5                  1  1.25   151
## 2 (1.6,2.1]; (2) of Qs0e1n5                2  1.82   139
## 3 (2.1,2.3]; (3) of Qs0e1n5                3  2.25   139
## 4 (2.3,2.9]; (4) of Qs0e1n5                4  2.70   134
## 5 (2.9,6.6]; (5) of Qs0e1n5                5  3.77   111
```

```
# Joint Quantile Group Name
var.qjnt.grp.idx <- 'group.index'
list.cts2quantile <- list(list(vars=c('hgt0', 'wgt0'), prob=c(0, .5, 1.0)))
results <- df_cut_by_sliced_quantiles_joint(df, var.qjnt.grp.idx, list.cts2quantile,
```

```
                                          vars.group_by = c('indi.id'), vars.arrange = c('indi.id', '
                                          drop.any.quantile.na = TRUE, toprint = FALSE)
```

Hgt0 2 groups, Wgt0 2 groups too

```
## Joining, by = "quant.perc"
```

```
# Show Results
results$df.group.slice1.cnt.mean
```

```
## # A tibble: 4 x 7
## # Groups:   hgt0_Qs0e1n2, wgt0_Qs0e1n2 [4]
##   hgt0_Qs0e1n2                wgt0_Qs0e1n2                          group.index hgt0_mean wgt0_mean hg
##   <fct>                       <fct>                                       <int>     <dbl>     <dbl>  <
## 1 [40.6,49.4]; (1) of Qs0e1n2 [1.4e+03,3.01e+03]; (1) of Qs0e1n2              1      47.4     2650.
## 2 [40.6,49.4]; (1) of Qs0e1n2 (3.01e+03,5.49e+03]; (2) of Qs0e1n2           2      48.5     3244.
## 3 (49.4,58]; (2) of Qs0e1n2   [1.4e+03,3.01e+03]; (1) of Qs0e1n2            3      50.4     2829.
## 4 (49.4,58]; (2) of Qs0e1n2   (3.01e+03,5.49e+03]; (2) of Qs0e1n2           4      51.3     3483.
```

```
# Joint Quantile Group Name
var.qjnt.grp.idx <- 'group.index'
list.cts2quantile <- list(list(vars=c('wealthIdx'), prob=c(0, .5, 1.0)), list(vars=c('hgt0'), prob=c(0,
results <- df_cut_by_sliced_quantiles_joint((df %>% filter(S.country == 'Cebu')),
                                          var.qjnt.grp.idx, list.cts2quantile,
                                          vars.group_by = c('indi.id'), vars.arrange = c('indi.id',
                                          drop.any.quantile.na = TRUE, toprint = FALSE)
```

Hgt0 2 groups, Wealth 2 groups, Cebu Only

```
## Joining, by = c("S.country", "vil.id", "indi.id", "sex", "svymthRound", "momEdu", "wealthIdx", "hgt"
## "p.A.nProt")
```

```
# Show Results
results$df.group.slice1.cnt.mean
```

```
## # A tibble: 6 x 7
## # Groups:   wealthIdx_Qs0e1n2, hgt0_Qs0e1n3 [6]
##   wealthIdx_Qs0e1n2          hgt0_Qs0e1n3                group.index wealthIdx_mean hgt0_mean wealthI
##   <fct>                      <fct>                             <int>          <dbl>     <dbl>
## 1 [5.2,8.3]; (1) of Qs0e1n2  [41.1,48.4]; (1) of Qs0e1n3           1           7.15      46.9
## 2 [5.2,8.3]; (1) of Qs0e1n2  (48.4,50.1]; (2) of Qs0e1n3           2           7.18      49.2
## 3 [5.2,8.3]; (1) of Qs0e1n2  (50.1,58]; (3) of Qs0e1n3             3           7.13      51.3
## 4 (8.3,19.3]; (2) of Qs0e1n2 [41.1,48.4]; (1) of Qs0e1n3           4          11.1       47.2
## 5 (8.3,19.3]; (2) of Qs0e1n2 (48.4,50.1]; (2) of Qs0e1n3           5          11.2       49.3
## 6 (8.3,19.3]; (2) of Qs0e1n2 (50.1,58]; (3) of Qs0e1n3             6          11.6       51.7
```

Results of income + Wgt0 + Hgt0 joint Gruops in Cebu Weight at month 0 below and above median, height at month zero into three terciles.

```
# Joint Quantile Group Name
var.qjnt.grp.idx <- 'wltHgt0Wgt0.index'
list.cts2quantile <- list(list(vars=c('wealthIdx'), prob=c(0, .5, 1.0)), list(vars=c('hgt0', 'wgt0'), pr
results <- df_cut_by_sliced_quantiles_joint((df %>% filter(S.country == 'Cebu')),
                                          var.qjnt.grp.idx, list.cts2quantile,
                                          vars.group_by = c('indi.id'), vars.arrange = c('indi.id', '
                                          drop.any.quantile.na = TRUE, toprint = FALSE)
```

```
## Joining, by = "quant.perc"Joining, by = c("S.country", "vil.id", "indi.id", "sex", "svymthRound", "m
## "prot", "cal", "p.A.prot", "p.A.nProt")
```

```
# Show Results
results$df.group.slice1.cnt.mean
```

```
## # A tibble: 8 x 10
## # Groups:   wealthIdx_Qs0e1n2, hgt0_Qs0e1n2, wgt0_Qs0e1n2 [8]
##   wealthIdx_Qs0e1n2       hgt0_Qs0e1n2            wgt0_Qs0e1n2                  wltHgt0Wgt0.ind~ weal
##   <fct>                   <fct>                   <fct>                                    <int>
## 1 [5.2,8.3]; (1) of Qs0e~ [41.1,49.2]; (1) of Qs~ [1.4e+03,2.98e+03]; (1) of ~                1
## 2 [5.2,8.3]; (1) of Qs0e~ [41.1,49.2]; (1) of Qs~ (2.98e+03,5.49e+03]; (2) of~                2
## 3 [5.2,8.3]; (1) of Qs0e~ (49.2,58]; (2) of Qs0e~ [1.4e+03,2.98e+03]; (1) of ~                3
## 4 [5.2,8.3]; (1) of Qs0e~ (49.2,58]; (2) of Qs0e~ (2.98e+03,5.49e+03]; (2) of~                4
## 5 (8.3,19.3]; (2) of Qs0~ [41.1,49.2]; (1) of Qs~ [1.4e+03,2.98e+03]; (1) of ~                5
## 6 (8.3,19.3]; (2) of Qs0~ [41.1,49.2]; (1) of Qs~ (2.98e+03,5.49e+03]; (2) of~                6
## 7 (8.3,19.3]; (2) of Qs0~ (49.2,58]; (2) of Qs0e~ [1.4e+03,2.98e+03]; (1) of ~                7
## 8 (8.3,19.3]; (2) of Qs0~ (49.2,58]; (2) of Qs0e~ (2.98e+03,5.49e+03]; (2) of~                8
```

Line by Line–Quantiles Var by Var The idea of the function is to generate quantiles levels first, and then use those to generate the categories based on quantiles. Rather than doing this in one step. These are done in two steps, to increase clarity in the quantiles used for quantile category generation. And a dataframe with these quantiles are saved as a separate output of the function.

Dataframe of Variables' Group-by Level Quantiles Quantiles from Different Variables. Note that these variables are specific to the individual, not individual/month. So we need to first slick the data, so that we only get the first rows.

Do this in several steps to clarify group_by level. No speed loss.

```
# Selected Variables, many Percentiles
vars.group_by <- c('indi.id')
vars.arrange <- c('indi.id', 'svymthRound')
vars.cts2quantile <- c('wealthIdx', 'hgt0', 'wgt0')
seq.quantiles <- c(0, 0.3333, 0.6666, 1.0)
df.sliced <- df_sliced_quantiles(df, vars.cts2quantile, seq.quantiles, vars.group_by, vars.arrange)
```

```
## Joining, by = "quant.perc"Joining, by = "quant.perc"
```

```
df.sliced.quantiles <- df.sliced$df.sliced.quantiles
df.grp.L1 <- df.sliced$df.grp.L1
```

```
df.sliced.quantiles
```

```
## # A tibble: 4 x 4
##   quant.perc wealthIdx  hgt0  wgt0
##   <chr>          <dbl> <dbl> <dbl>
## 1 0%               1    40.6 1402.
## 2 33.33%           5.2  48.5 2843.
## 3 66.66%           8.3  50.2 3209.
## 4 100%            19.3  58   5494.
```

```
# Quantiles all Variables
suppressMessages(lapply(names(df), gen_quantiles, df=df.grp.L1, prob=seq(0.1,0.9,0.10)) %>% reduce(full_
```

```
## Warning in quantile(as.numeric(df[[var]]), prob, na.rm = TRUE): NAs introduced by coercion
```

```
## Warning in quantile(as.numeric(df[[var]]), prob, na.rm = TRUE): NAs introduced by coercion
```

```
## # A tibble: 9 x 16
##   quant.perc S.country vil.id indi.id   sex svymthRound momEdu wealthIdx   hgt   wgt  hgt0  wgt0  pr
##   <chr>          <dbl>  <dbl>   <dbl> <dbl>       <dbl>  <dbl>     <dbl> <dbl> <dbl> <dbl> <dbl> <db
## 1 10%               NA      3    203.    NA           0    5.7       1.7  46.3 1397.  46.6 2500.    0
## 2 20%               NA      4    405.    NA           0    6.9       2.3  47.3 1840.  47.7 2686.    0
## 3 30%               NA      6    608.    NA           0    7.7       3.3  48   2272.  48.3 2804.    0
## 4 40%               NA      8    810.    NA           0    8.6       6.3  48.7 2669.  48.8 2910.    0
## 5 50%               NA      9   1012     NA           0    9.3       7.3  49.4 3050.  49.4 3013     0
## 6 60%               NA     13   1214.    NA           0   10.4       8.3  49.9 3440.  49.9 3126.    0
## 7 70%               NA     14   1416.    NA           0   11.4       8.3  50.5 3857.  50.4 3250.    0
## 8 80%               NA     17   1619.    NA           0   12.7       9.3  51.2 4258.  51.0 3418.    1
## 9 90%               NA     26   1821.    NA           0   14.6      11.3  52.3 4704.  52   3683.    1
```

Cut Quantile Categorical Variables Using the Quantiles we have generate, cut the continuous variables to generate categorical quantile variables in the full dataframe.

Note that we can only cut based on unique breaks, but sometimes quantile break-points are the same if some values are often observed, and also if there are too few observations with respect to quantile groups.

To resolve this issue, we only look at unique quantiles.

We need several support Functions: 1. support functions to generate suffix for quantile variables based on quantile cuts 2. support for labeling variables of resulting quantiles beyond bracketing

```
# Function Testing
arr.quantiles <- df.sliced.quantiles[[substitute('wealthIdx')]]
arr.quantiles
```

```
## [1]  1.0  5.2  8.3 19.3
```

```
arr.sort.unique.quantiles <- sort(unique(df.sliced.quantiles[[substitute('wealthIdx')]]))
arr.sort.unique.quantiles
```

```
## [1]  1.0  5.2  8.3 19.3
```

```
f_Q_label(arr.quantiles, arr.sort.unique.quantiles[1], seq.quantiles)
```

```
## [1] "(1) of Qs0e1n3"
```

```
f_Q_label(arr.quantiles, arr.sort.unique.quantiles[2], seq.quantiles)
```

```
## [1] "(2) of Qs0e1n3"
```

```
lapply(arr.sort.unique.quantiles[1:(length(arr.sort.unique.quantiles)-1)],
       f_Q_label,
       arr.quantiles=arr.quantiles,
       seq.quantiles=seq.quantiles)
```

```
## [[1]]
## [1] "(1) of Qs0e1n3"
##
## [[2]]
## [1] "(2) of Qs0e1n3"
##
## [[3]]
## [1] "(3) of Qs0e1n3"
```

```
# Generate Categorical Variables of Quantiles
vars.group_by <- c('indi.id')
```

```
vars.arrange <- c('indi.id', 'svymthRound')
vars.cts2quantile <- c('wealthIdx', 'hgt0', 'wgt0')
seq.quantiles <- c(0, 0.3333, 0.6666, 1.0)
df.cut <- df_cut_by_sliced_quantiles(df, vars.cts2quantile, seq.quantiles, vars.group_by, vars.arrange)
```

```
## Joining, by = "quant.perc"Joining, by = "quant.perc"
```

```
vars.quantile.cut <- df.cut$vars.quantile.cut
df.with.cut.quant <- df.cut$df.with.cut.quant
df.grp.L1 <- df.cut$df.grp.L1
```

```
# Cut Variables Generated
names(vars.quantile.cut)
```

```
## [1] "wealthIdx_Qs0e1n3" "hgt0_Qs0e1n3"      "wgt0_Qs0e1n3"
```

```
summary(vars.quantile.cut)
```

```
##                   wealthIdx_Qs0e1n3                       hgt0_Qs0e1n3
##  [1,5.2]; (1) of Qs0e1n3   :10958   [40.6,48.5]; (1) of Qs0e1n3:10232   [1.4e+03,2.84e+03]; (1) of Qs
##  (5.2,8.3]; (2) of Qs0e1n3 :13812   (48.5,50.2]; (2) of Qs0e1n3: 9895   (2.84e+03,3.21e+03]; (2) of (
##  (8.3,19.3]; (3) of Qs0e1n3:10295   (50.2,58]; (3) of Qs0e1n3  : 9908   (3.21e+03,5.49e+03]; (3) of (
##                                     NA's                      : 5030   NA's
```

```
# options(repr.matrix.max.rows=50, repr.matrix.max.cols=20)
# df.with.cut.quant
```

```
# Group By Results
f.count <- function(df, var.cts, seq.quantiles) {
    df %>% select(S.country, indi.id, svymthRound, matches(paste0(var.cts, collapse='|'))) %>%
        group_by(!!sym(f_var_rename(paste0(var.cts,'_q'), seq.quantiles))) %>%
        summarise_all(funs(n=n()))
}
```

```
# Full Panel Results
lapply(vars.cts2quantile, f.count, df=df.with.cut.quant, seq.quantiles=seq.quantiles)
```

Individual Variables' Quantile Cuts Review Results

```
## Warning: Factor `hgt0_Qs0e1n3` contains implicit NA, consider using `forcats::fct_explicit_na`
```

```
## Warning: Factor `wgt0_Qs0e1n3` contains implicit NA, consider using `forcats::fct_explicit_na`
```

```
## [[1]]
## # A tibble: 3 x 5
##   wealthIdx_Qs0e1n3          S.country_n indi.id_n svymthRound_n wealthIdx_n
##   <fct>                            <int>     <int>         <int>       <int>
## 1 [1,5.2]; (1) of Qs0e1n3          10958     10958         10958       10958
## 2 (5.2,8.3]; (2) of Qs0e1n3        13812     13812         13812       13812
## 3 (8.3,19.3]; (3) of Qs0e1n3       10295     10295         10295       10295
##
## [[2]]
## # A tibble: 4 x 5
##   hgt0_Qs0e1n3               S.country_n indi.id_n svymthRound_n hgt0_n
##   <fct>                            <int>     <int>         <int>  <int>
## 1 [40.6,48.5]; (1) of Qs0e1n3      10232     10232         10232  10232
```

```
## 2 (48.5,50.2]; (2) of Qs0e1n3        9895      9895          9895   9895
## 3 (50.2,58]; (3) of Qs0e1n3          9908      9908          9908   9908
## 4 <NA>                               5030      5030          5030   5030
##
## [[3]]
## # A tibble: 4 x 5
##   wgt0_Qs0e1n3                        S.country_n indi.id_n svymthRound_n wgt0_n
##   <fct>                                     <int>     <int>         <int>  <int>
## 1 [1.4e+03,2.84e+03]; (1) of Qs0e1n3        10105     10105         10105  10105
## 2 (2.84e+03,3.21e+03]; (2) of Qs0e1n3       10056     10056         10056  10056
## 3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3        9858      9858          9858   9858
## 4 <NA>                                       5046      5046          5046   5046
```

```
# Results Individual Slice
lapply(vars.cts2quantile, f.count,
       df=(df.with.cut.quant %>% group_by(!!!syms(vars.group_by)) %>% arrange(!!!syms(vars.arrange)) %>%
       seq.quantiles = seq.quantiles)
```

```
## Warning: Factor `hgt0_Qs0e1n3` contains implicit NA, consider using `forcats::fct_explicit_na`
```

```
## Warning: Factor `wgt0_Qs0e1n3` contains implicit NA, consider using `forcats::fct_explicit_na`
```

```
## [[1]]
## # A tibble: 3 x 5
##   wealthIdx_Qs0e1n3          S.country_n indi.id_n svymthRound_n wealthIdx_n
##   <fct>                            <int>     <int>         <int>       <int>
## 1 [1,5.2]; (1) of Qs0e1n3            683       683           683         683
## 2 (5.2,8.3]; (2) of Qs0e1n3          768       768           768         768
## 3 (8.3,19.3]; (3) of Qs0e1n3         572       572           572         572
##
## [[2]]
## # A tibble: 4 x 5
##   hgt0_Qs0e1n3                S.country_n indi.id_n svymthRound_n hgt0_n
##   <fct>                             <int>     <int>         <int>  <int>
## 1 [40.6,48.5]; (1) of Qs0e1n3         580       580           580    580
## 2 (48.5,50.2]; (2) of Qs0e1n3         561       561           561    561
## 3 (50.2,58]; (3) of Qs0e1n3           568       568           568    568
## 4 <NA>                                314       314           314    314
##
## [[3]]
## # A tibble: 4 x 5
##   wgt0_Qs0e1n3                        S.country_n indi.id_n svymthRound_n wgt0_n
##   <fct>                                     <int>     <int>         <int>  <int>
## 1 [1.4e+03,2.84e+03]; (1) of Qs0e1n3          569       569           569    569
## 2 (2.84e+03,3.21e+03]; (2) of Qs0e1n3         569       569           569    569
## 3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3         570       570           570    570
## 4 <NA>                                        315       315           315    315
```

Differential Quantiles for Different Variables Then Combine to Form New Groups Collect together different quantile base variables and their percentile cuttings quantile rules. Input Parameters.

```
# Generate Categorical Variables of Quantiles
vars.group_by <- c('indi.id')
vars.arrange <- c('indi.id', 'svymthRound')
```

```
# Quantile Variables and Quantiles
vars.cts2quantile.wealth <- c('wealthIdx')
seq.quantiles.wealth <- c(0, .5, 1.0)
vars.cts2quantile.wgthgt <- c('hgt0', 'wgt0')
seq.quantiles.wgthgt <- c(0, .3333, 0.6666, 1.0)
drop.any.quantile.na <- TRUE
# collect to list
list.cts2quantile <- list(list(vars=vars.cts2quantile.wealth,
                               prob=seq.quantiles.wealth),
                          list(vars=vars.cts2quantile.wgthgt,
                               prob=seq.quantiles.wgthgt))
```

Check if Within Group Variables Are The Same Need to make sure quantile variables are unique within groups

```
vars.cts2quantile <- unlist(lapply(list.cts2quantile, function(elist) elist$vars))
f_check_distinct_ingroup(df, vars.group_by, vars.values_in_group=vars.cts2quantile)
```

```
# Original dimensions
dim(df)
```

Keep only non-NA for all Quantile Variables

```
## [1] 35065    15
```

```
# All Continuous Variables from lists
vars.cts2quantile <- unlist(lapply(list.cts2quantile, function(elist) elist$vars))
vars.cts2quantile
```

```
## [1] "wealthIdx" "hgt0"      "wgt0"
```

```
# Keep only if not NA for all Quantile variables
if (drop.any.quantile.na) {
    df.select <- df %>% drop_na(c(vars.group_by, vars.arrange, vars.cts2quantile))
}
dim(df.select)
```

```
## [1] 30019    15
```

```
# Dealing with a list of quantile variables
df.cut.wealth <- df_cut_by_sliced_quantiles(df.select, vars.cts2quantile.wealth, seq.quantiles.wealth, v
summary(df.cut.wealth$vars.quantile.cut)
```

Apply Quantiles for Each Quantile Variable

```
##                   wealthIdx_Qs0e1n2
##  [1,7.3]; (1) of Qs0e1n2   :14936
##  (7.3,19.3]; (2) of Qs0e1n2:15083
```

```
# summary((df.cut.wealth$df.with.cut.quant)[['wealthIdx_Qs0e1n2']])
# df.cut.wealth$df.with.cut.quant %>% filter(is.na(wealthIdx_Qs0e1n2))
# df.cut.wealth$df.with.cut.quant %>% filter(indi.id == 500)
```

```
df.cut.wgthgt <- df_cut_by_sliced_quantiles(df.select, vars.cts2quantile.wgthgt, seq.quantiles.wgthgt, v
```

```
## Joining, by = "quant.perc"
```

```
summary(df.cut.wgthgt$vars.quantile.cut)
```

```
##                        hgt0_Qs0e1n3                                    wgt0_Qs0e1n3
##  [40.6,48.5]; (1) of Qs0e1n3:10216   [1.4e+03,2.84e+03]; (1) of Qs0e1n3 :10105
##  (48.5,50.2]; (2) of Qs0e1n3: 9895   (2.84e+03,3.21e+03]; (2) of Qs0e1n3:10056
##  (50.2,58]; (3) of Qs0e1n3  : 9908   (3.21e+03,5.49e+03]; (3) of Qs0e1n3: 9858
```

```
# Function to handle list inputs with different quantiles vars and probabilities
df_cut_by_sliced_quantiles_grps <- function(quantile.grp.list, df, vars.group_by, vars.arrange) {
    vars.cts2quantile <- quantile.grp.list$vars
    seq.quantiles <- quantile.grp.list$prob
    return(df_cut_by_sliced_quantiles(df, vars.cts2quantile, seq.quantiles, vars.group_by, vars.arrange
}
```

```
# Apply function
df.cut.list <- lapply(list.cts2quantile, df_cut_by_sliced_quantiles_grps,
                      df=df.select, vars.group_by=vars.group_by, vars.arrange=vars.arrange)
```

Apply Quantiles Functionally

```
## Joining, by = "quant.perc"
```

```
# Reduce Resulting Matrixes Together
df.with.cut.quant.all <- lapply(df.cut.list, function(elist) elist$df.with.cut.quant) %>% reduce(left_j
```

```
## Joining, by = c("S.country", "vil.id", "indi.id", "sex", "svymthRound", "momEdu", "wealthIdx", "hgt"
## "p.A.nProt")
```

```
dim(df.with.cut.quant.all)
```

```
## [1] 30019    18
```

```
# Obrain Newly Created Quantile Group Variables
vars.quantile.cut.all <- unlist(lapply(df.cut.list, function(elist) names(elist$vars.quantile.cut)))
vars.quantile.cut.all
```

```
## [1] "wealthIdx_Qs0e1n2" "hgt0_Qs0e1n3"      "wgt0_Qs0e1n3"
```

Summarize by Groups Summarize by all groups.

```
summary(df.with.cut.quant.all %>% select(one_of(vars.quantile.cut.all)))
```

```
##                   wealthIdx_Qs0e1n2                       hgt0_Qs0e1n3
##  [1,7.3]; (1) of Qs0e1n2   :14936   [40.6,48.5]; (1) of Qs0e1n3:10216   [1.4e+03,2.84e+03]; (1) of Qs
##  (7.3,19.3]; (2) of Qs0e1n2:15083   (48.5,50.2]; (2) of Qs0e1n3: 9895   (2.84e+03,3.21e+03]; (2) of
##                                     (50.2,58]; (3) of Qs0e1n3  : 9908   (3.21e+03,5.49e+03]; (3) of
```

```
# df.with.cut.quant.all %>%
#     group_by(!!!syms(vars.quantile.cut.all)) %>%
#     summarise_at(vars.cts2quantile, funs(mean, n()))
```

```
# Generate Joint Quantile Index Variable
var.qjnt.grp.idx <- 'group.index'
df.with.cut.quant.all <- df.with.cut.quant.all %>% mutate(!!var.qjnt.grp.idx := group_indices(., !!!sym
```

```
arr.group.idx <- t(sort(unique(df.with.cut.quant.all[[var.qjnt.grp.idx]])))
arr.group.idx
```

Generate Joint Quantile Vars Unique Groups

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,
## [1,]    1    2    3    4    5    6    7    8    9    10    11    12    13    14    15    16    17
```

```
df.with.cut.quant.all %>% group_by(!!!syms(vars.quantile.cut.all), !!sym(var.qjnt.grp.idx)) %>%
        summarise_at(vars.cts2quantile, funs(mean, n()))
```

```
## # A tibble: 18 x 10
## # Groups:   wealthIdx_Qs0e1n2, hgt0_Qs0e1n3, wgt0_Qs0e1n3 [18]
##    wealthIdx_Qs0e1n2        hgt0_Qs0e1n3            wgt0_Qs0e1n3                  group.index weal
##    <fct>                    <fct>                   <fct>                               <int>
##  1 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~           1
##  2 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~           2
##  3 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~           3
##  4 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~           4
##  5 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~           5
##  6 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~           6
##  7 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ [1.4e+03,2.84e+03]; (1) of Qs~           7
##  8 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ (2.84e+03,3.21e+03]; (2) of Q~           8
##  9 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ (3.21e+03,5.49e+03]; (3) of Q~           9
## 10 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~          10
## 11 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~          11
## 12 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~          12
## 13 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~          13
## 14 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~          14
## 15 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~          15
## 16 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ [1.4e+03,2.84e+03]; (1) of Qs~          16
## 17 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ (2.84e+03,3.21e+03]; (2) of Q~          17
## 18 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ (3.21e+03,5.49e+03]; (3) of Q~          18
```

```
df.with.cut.quant.all  %>% group_by(!!!syms(vars.group_by)) %>% arrange(!!!syms(vars.arrange)) %>% slic
        group_by(!!!syms(vars.quantile.cut.all), !!sym(var.qjnt.grp.idx)) %>%
        summarise_at(vars.cts2quantile, funs(mean, n()))
```

```
## # A tibble: 18 x 10
## # Groups:   wealthIdx_Qs0e1n2, hgt0_Qs0e1n3, wgt0_Qs0e1n3 [18]
##    wealthIdx_Qs0e1n2        hgt0_Qs0e1n3            wgt0_Qs0e1n3                  group.index weal
##    <fct>                    <fct>                   <fct>                               <int>
##  1 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~           1
##  2 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~           2
##  3 [1,7.3]; (1) of Qs0e1n2  [40.6,48.5]; (1) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~           3
##  4 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~           4
##  5 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~           5
##  6 [1,7.3]; (1) of Qs0e1n2  (48.5,50.2]; (2) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~           6
##  7 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ [1.4e+03,2.84e+03]; (1) of Qs~           7
##  8 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ (2.84e+03,3.21e+03]; (2) of Q~           8
##  9 [1,7.3]; (1) of Qs0e1n2  (50.2,58]; (3) of Qs0e1~ (3.21e+03,5.49e+03]; (3) of Q~           9
## 10 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~          10
## 11 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~          11
## 12 (7.3,19.3]; (2) of Qs0e~ [40.6,48.5]; (1) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~          12
## 13 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ [1.4e+03,2.84e+03]; (1) of Qs~          13
```

```
## 14 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ (2.84e+03,3.21e+03]; (2) of Q~          14
## 15 (7.3,19.3]; (2) of Qs0e~ (48.5,50.2]; (2) of Qs0~ (3.21e+03,5.49e+03]; (3) of Q~          15
## 16 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ [1.4e+03,2.84e+03]; (1) of Qs~          16
## 17 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ (2.84e+03,3.21e+03]; (2) of Q~          17
## 18 (7.3,19.3]; (2) of Qs0e~ (50.2,58]; (3) of Qs0e1~ (3.21e+03,5.49e+03]; (3) of Q~          18
```

Change values Based on Index Index from 1 to 18, change input values based on index

```
# arr.group.idx.subsidy <- arr.group.idx*2 - ((arr.group.idx)^2)*0.01
arr.group.idx.subsidy <- arr.group.idx*2
df.with.cut.quant.all %>%
        mutate(more_prot = prot + arr.group.idx.subsidy[!!sym(var.qjnt.grp.idx)]) %>%
        group_by(!!!syms(vars.quantile.cut.all), !!sym(var.qjnt.grp.idx))  %>%
        summarise_at(c('more_prot', 'prot'), funs(mean(., na.rm=TRUE)))
```

```
## # A tibble: 18 x 6
## # Groups:   wealthIdx_Qs0e1n2, hgt0_Qs0e1n3, wgt0_Qs0e1n3 [18]
##    wealthIdx_Qs0e1n2          hgt0_Qs0e1n3                wgt0_Qs0e1n3                          group.
##    <fct>                      <fct>                       <fct>                                 <
##  1 [1,7.3]; (1) of Qs0e1n2    [40.6,48.5]; (1) of Qs0e1n3 [1.4e+03,2.84e+03]; (1) of Qs0e1n3
##  2 [1,7.3]; (1) of Qs0e1n2    [40.6,48.5]; (1) of Qs0e1n3 (2.84e+03,3.21e+03]; (2) of Qs0e1n3
##  3 [1,7.3]; (1) of Qs0e1n2    [40.6,48.5]; (1) of Qs0e1n3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3
##  4 [1,7.3]; (1) of Qs0e1n2    (48.5,50.2]; (2) of Qs0e1n3 [1.4e+03,2.84e+03]; (1) of Qs0e1n3
##  5 [1,7.3]; (1) of Qs0e1n2    (48.5,50.2]; (2) of Qs0e1n3 (2.84e+03,3.21e+03]; (2) of Qs0e1n3
##  6 [1,7.3]; (1) of Qs0e1n2    (48.5,50.2]; (2) of Qs0e1n3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3
##  7 [1,7.3]; (1) of Qs0e1n2    (50.2,58]; (3) of Qs0e1n3   [1.4e+03,2.84e+03]; (1) of Qs0e1n3
##  8 [1,7.3]; (1) of Qs0e1n2    (50.2,58]; (3) of Qs0e1n3   (2.84e+03,3.21e+03]; (2) of Qs0e1n3
##  9 [1,7.3]; (1) of Qs0e1n2    (50.2,58]; (3) of Qs0e1n3   (3.21e+03,5.49e+03]; (3) of Qs0e1n3
## 10 (7.3,19.3]; (2) of Qs0e1n2 [40.6,48.5]; (1) of Qs0e1n3 [1.4e+03,2.84e+03]; (1) of Qs0e1n3
## 11 (7.3,19.3]; (2) of Qs0e1n2 [40.6,48.5]; (1) of Qs0e1n3 (2.84e+03,3.21e+03]; (2) of Qs0e1n3
## 12 (7.3,19.3]; (2) of Qs0e1n2 [40.6,48.5]; (1) of Qs0e1n3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3
## 13 (7.3,19.3]; (2) of Qs0e1n2 (48.5,50.2]; (2) of Qs0e1n3 [1.4e+03,2.84e+03]; (1) of Qs0e1n3
## 14 (7.3,19.3]; (2) of Qs0e1n2 (48.5,50.2]; (2) of Qs0e1n3 (2.84e+03,3.21e+03]; (2) of Qs0e1n3
## 15 (7.3,19.3]; (2) of Qs0e1n2 (48.5,50.2]; (2) of Qs0e1n3 (3.21e+03,5.49e+03]; (3) of Qs0e1n3
## 16 (7.3,19.3]; (2) of Qs0e1n2 (50.2,58]; (3) of Qs0e1n3   [1.4e+03,2.84e+03]; (1) of Qs0e1n3
## 17 (7.3,19.3]; (2) of Qs0e1n2 (50.2,58]; (3) of Qs0e1n3   (2.84e+03,3.21e+03]; (2) of Qs0e1n3
## 18 (7.3,19.3]; (2) of Qs0e1n2 (50.2,58]; (3) of Qs0e1n3   (3.21e+03,5.49e+03]; (3) of Qs0e1n3
```