caseMatch for Panel Data

Dataset and code can be found at GitHub: https://github.com/m-note/Replication/tree/master/caseMatch (https://github.com/m-note/Replication/tree/master/caseMatch)

1. How it works

caseMatchPD calculates the Mahalanobis distance of two vectors. For a given panel data that does not have any missing variables (the first table in Figure 1), caseMatchPD firstly makes vectors for each id (that is a country in Figure 1). Since panel data has timeseries for each id, caseMatchPD merges all years of a id into a single vector (the second table in Figure 1). Then, the distance between a pair of these new vectors are calculated in turn.

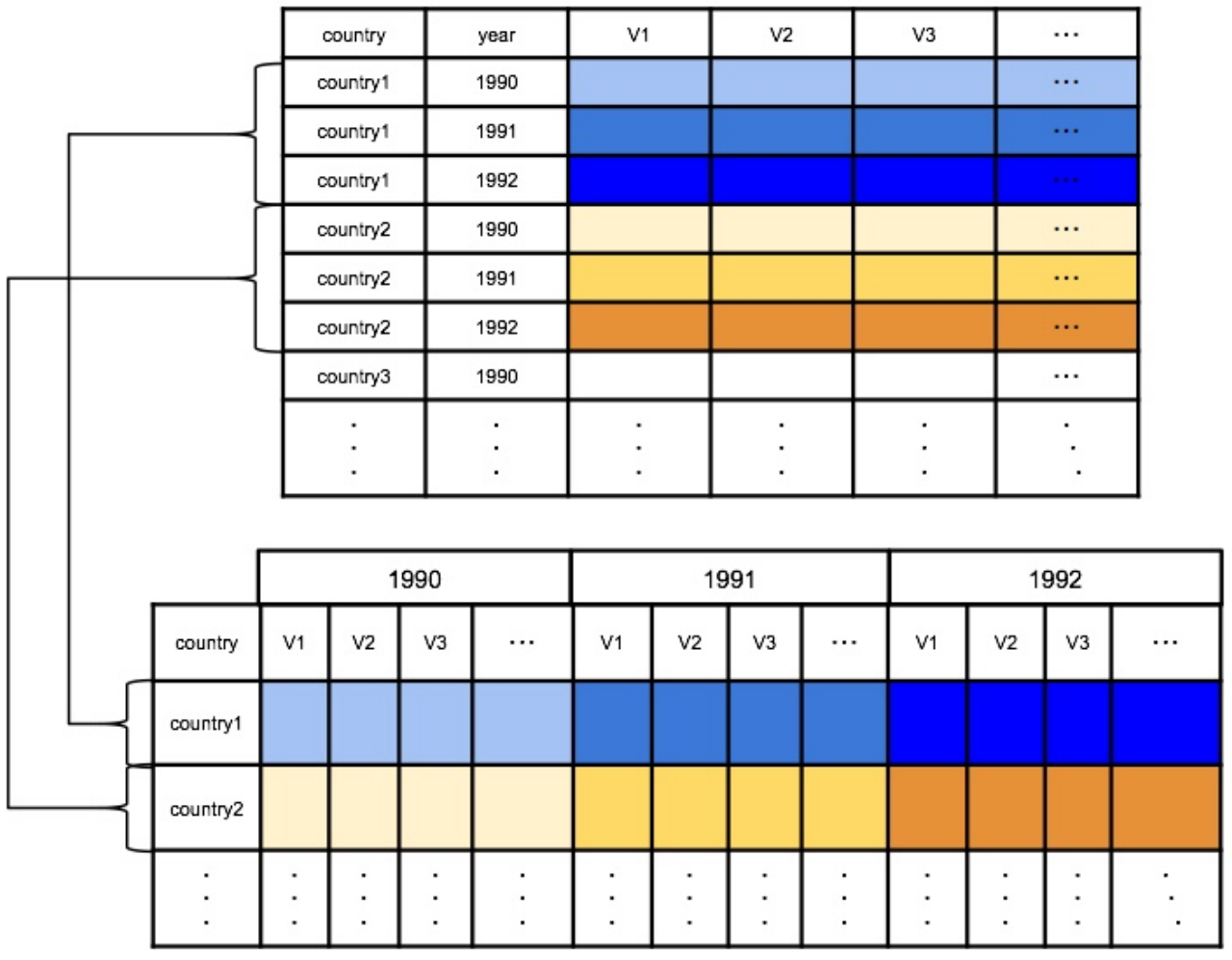

Figure 1: How caseMatchPD makes vectors

Is it possible to think as shown above for panel data?

2. Preparation

dplyr is required for sorting

```
library(dplyr)
```

Load data

This dataset shows how much aid each recipient recieved in each year, and some relating variables to foreign aid.

```
data <- read.csv("PanelData_930.csv")
```

Data consists of 13 variables:
- recipient: name of recipient country (128 countries)
- year: 1991-2010 (20 years)
- un_region: United Nations regional code
- Aid_All: Total aid amount (logged)
- p4_polity2: Polity IV score
- fh_inverse_pr / fh_inverse_cl: freedom house score
- Giniall: Gini coefficient scaled 0-100
- Corruption: Corruption Perceptions Index
- Rents: Total natural resources rents of GDP
- Population: Total population (logged)
- GDP: GDP per capita (constant 2005 US dollars, logged)
- eu: EU membership in 2013

```
head(data)
```

```
##     recipient year un_region  Aid_All p4_polity2 fh_inverse_pr
## 1 Afghanistan 1991        34 18.49972         -8             1
## 2 Afghanistan 1992        34 18.56790          0             2
## 3 Afghanistan 1993        34 18.21110          0             1
## 4 Afghanistan 1994        34 17.89176          0             1
## 5 Afghanistan 1995        34 18.56311          0             1
## 6 Afghanistan 1996        34 18.64389         -7             1
##   fh_inverse_cl  Giniall Corruption Rents Population      GDP eu
## 1             1 46.84820   3.985369     0   16.35016 6.395312  0
## 2             2 60.51358   3.722769     0   16.44104 6.050286  0
## 3             1 51.08042   3.884185     0   16.53518 7.119950  0
## 4             1 40.63805   1.731650     0   16.61796 4.944424  0
## 5             1 42.58897   4.150223     0   16.68262 7.278733  0
## 6             1 35.53721   2.855289     0   16.72869 6.959101  0
```

3. Function

```
caseMatchPD <- function(data, id, year=1, dropvars=NA, returnNum=NA, method="mahal
anobis", treatment=NULL){
  # data: dataset
  # id: id (ex. country name)
  # year: how many years are there for each country
  #       If the dataset is not panel data, year=1
  # dropvars: columns that are not used
  # returnNum: return number

  # drop dropvars
  if(is.na(dropvars[1])){}else{data <- data[, !(colnames(data) %in% dropvars)]}

  # unique id vector
  id_unique <- unique(data[,id])

  # If there is a treatment, divide dataset into one with treatment and one withou
t
```

```
  if (!is.null(treatment)){
    ## Dataset with treatment
    data_T <- data[data[, treatment]==1, ]
    data_T <- data_T[ , !(colnames(data_T) %in% treatment)]
    data_NT <- data[data[, treatment]==0, ]
    data_NT <- data_NT[ , !(colnames(data_NT) %in% treatment)]
  }

  # creating vectors
    for_time = 1
    result_list <- as.list(NULL) # initialize results list
    for (i in seq(1, nrow(data), year)){
      data_temp <- data[data[id]==as.character(id_unique[for_time]), # id matches
                        !(colnames(data) %in% c(id, treatment))]  # id is not need
ed for vector
      result_list <- append(result_list, list(as.vector(t(data_temp))))
      for_time = for_time + 1
    }
  if (!is.null(treatment)){
    ## Dataset with treatment
      for_time = 1
      id_unique_T <- unique(data_T[, id])
      result_list_T <- as.list(NULL) # initialize results list
      for (i in seq(1, nrow(data_T), year)){
        data_temp <- data_T[data_T[id]==as.character(id_unique_T[for_time]), # id
matches
                            !(colnames(data_T) %in% id)]  # id is not needed for vec
tor
        result_list_T <- append(result_list_T, list(as.vector(t(data_temp))))
        for_time = for_time + 1
      }
      ## Dataset without treatment
      for_time = 1
      id_unique_NT <- unique(data_NT[, id])
      result_list_NT <- as.list(NULL) # initialize results list
      for (i in seq(1, nrow(data_NT), year)){
        data_temp <- data_NT[data_NT[id]==as.character(id_unique_NT[for_time]), #
id matches
                             !(colnames(data_NT) %in% id)]  # id is not needed for ve
ctor
        result_list_NT <- append(result_list_NT, list(as.vector(t(data_temp))))
        for_time = for_time + 1
      }
  }


  # for dataset without treatment / make combinations at first
  if (is.null(treatment)){
    combination <- as.data.frame(t(combn(seq(1, length(id_unique), 1), 2)))
    print(paste("Number of combinations:", nrow(combination)))
    combi_length <- nrow(combination)
  }

  # calculation
```

```
  if (method=="cos"){
    # cos similarity
    result <- data.frame("id1" = NA, "id2" = NA, "similarity"=NA)
    for (s in 1:nrow(combination)){
      country1_num <- combination[s, 1]
      country2_num <- combination[s, 2]

      country1_vec <- unlist(result_list[country1_num])
      country2_vec <- unlist(result_list[country2_num])

      cos <- country1_vec %*% country2_vec / sqrt(country1_vec%*%country1_vec * co
untry2_vec%*%country2_vec)

      result <- rbind.data.frame(result,
            data.frame("id1" = id_unique[country1_num], "id2" = id_unique[country2
_num], "similarity"=cos))
    }
    result <- result[2:nrow(result),]
    # sort by dplyr
    result <- result %>%
    dplyr::arrange(desc(similarity))
  }

  if(method=="euclid"){
    # euclidean distance
    result <- data.frame("id1" = NA, "id2" = NA, "distance"=NA)
    for (s in 1:nrow(combination)){
      id1_num <- combination[s, 1]
      id2_num <- combination[s, 2]

      id1_vec <- unlist(result_list[id1_num])
      id2_vec <- unlist(result_list[id2_num])

      euc <- sqrt(sum((id1_vec - id2_vec) ^ 2))
      result <- rbind.data.frame(result,
            data.frame("id1" = id_unique[id1_num], "id2" = id_unique[id2_num], "di
stance"=euc))
    }
    result <- result[2:nrow(result),] # delete NA attached in initialization
    # sort by dplyr
    result <- result %>%
    dplyr::arrange(distance)
  }

  if(method=="mahalanobis" & is.null(treatment)){ # Datast without treatment
    result <- data.frame("id1" = NA, "id2" = NA, "distance"=NA)
    # the way to make variance-covariance matrix is different from year>1 and year
=1
    if (year > 1){
      data_temp <- data[, !(colnames(data) %in% c(id, dropvars))]
      data_temp <- t(data.frame(result_list))
      rownames(data_temp) <- NULL
      covData <- cov(data_temp)
    }else{covData <- cov(data[, !(colnames(data) %in% c(id, dropvars))])}
```

```
    t<-proc.time() # check time
    for (s in 1:nrow(combination)){
      id1_num <- combination[s, 1]
      id2_num <- combination[s, 2]

      id1_vec <- unlist(result_list[id1_num])
      id2_vec <- unlist(result_list[id2_num])

      #print(paste("Processing", s, "out of", combi_length, "/", id_unique[id1_num
], id_unique[id2_num]))
      maha <- mahalanobis(id1_vec, id2_vec, covData, tol=1e-50) # set tolalence ht
tp://goo.gl/hmmia0

      result <- rbind.data.frame(result,
            data.frame("id1" = id_unique[id1_num], "id2" = id_unique[id2_num], "di
stance"=maha))

      if (maha < 0){
        print("Warning: Mahalanobis distance is negative value")
        break
      }
    }
    print(proc.time()-t) # time
    result <- result[2:nrow(result),] # delete NA attached in initialization
    # sort by dplyr
    result <- result %>%
      dplyr::arrange(distance)
  }

  if(method=="mahalanobis" & !is.null(treatment)){ # Datast with treatment
    result <- data.frame("id1" = NA, "id2" = NA, "distance"=NA)
    # the way to make variance-covariance matrix is different from year>1 and year
=1
    if (year > 1){
      data_temp <- t(data.frame(result_list))
      data_temp_T <- t(data.frame(result_list_T))
      data_temp_NT <- t(data.frame(result_list_NT))
      rownames(data_temp) <- NULL
      covData <- cov(data_temp)

      t<-proc.time() # check time
       for (i in 1:length(result_list_T)){
        treatment_vec <- unlist(result_list_T[i])

        for (s in 1:length(result_list_NT)){
          not_treatment_vec <- unlist(result_list_NT[s])

          maha <- mahalanobis(treatment_vec, not_treatment_vec, covData, tol=1e-50
)

          result <- rbind.data.frame(result,
                data.frame("id1" = id_unique_T[i], "id2" = id_unique_NT[s], "dista
nce"=maha))
```

```
    }
  }
  print(proc.time()-t) # time
  result <- result[2:nrow(result),]
  # sort by dplyr
  result <- result %>%
  dplyr::arrange(distance)
  }else{ # year = 1
    covData <- cov(data[, !(colnames(data) %in% c(id, treatment))])

    t<-proc.time() # check time
    for (i in 1:length(result_list_T)){
    treatment_vec <- unlist(result_list_T[i])

      for (s in 1:length(result_list_NT)){
        not_treatment_vec <- unlist(result_list_NT[s])

        maha <- mahalanobis(treatment_vec, not_treatment_vec, covData, tol=1e-50
)

        result <- rbind.data.frame(result,
              data.frame("id1" = id_unique_T[i], "id2" = id_unique_NT[s], "dista
nce"=maha))
      }
    }
  result <- result[2:nrow(result),]
  print(proc.time()-t) # time
    # sort by dplyr
      result <- result %>%
      dplyr::arrange(distance)
    }
  }

  if(is.na(returnNum)){
    return (result)
  }else{return (result[1:returnNum,])}
}
```

4. Analysis

4.1 Analysis 1

This new function, caseMatchPD can return the same results as caseMatch.
caseMatch:

```
library(caseMatch)
data(EU)
```

```
mvars <- c("socialist","rgdpc","FHc","FHp","trade")
dropvars <- c("countryname","population")
out <- case.match(data=EU, id.var="countryname", leaveout.vars=dropvars,
        distance="mahalanobis", case.N=2,
        number.of.matches.to.return=10,
        treatment.var="eu", max.variance=TRUE)
```

caseMatchPD:

```
dropvars <- c("population")
out2 <- caseMatchPD(data=EU, id="countryname", year=1, dropvars=dropvars, method="
mahalanobis", treatment="eu")
```

Results are the same as caseMatch

```
out$cases
```

```
##     distances           unit id     unit id treat.variance
## 1  0.01770143         Australia     Austria            0.5
## 2  0.03716947            Canada Netherlands            0.5
## 3  0.08294586           Denmark      Norway            0.5
## 4  0.08573355         Australia      Sweden            0.5
## 5  0.13915500         Australia     Denmark            0.5
## 6  0.14219836           Belgium      Canada            0.5
## 7  0.18550358           Hungary  Kazakhstan            0.5
## 8  0.19608721           Finland      Israel            0.5
## 9  0.29438953           Austria      Norway            0.5
## 10 0.30398838 Antigua & Barbuda      Greece            0.5
```

```
out2[1:10,]
```

```
##            id1               id2   distance
## 1      Austria         Australia 0.01770143
## 2  Netherlands            Canada 0.03716947
## 3      Denmark            Norway 0.08294586
## 4       Sweden         Australia 0.08573355
## 5      Denmark         Australia 0.13915500
## 6      Belgium            Canada 0.14219836
## 7      Hungary        Kazakhstan 0.18550358
## 8      Finland            Israel 0.19608721
## 9      Austria            Norway 0.29438953
## 10      Greece Antigua & Barbuda 0.30398838
```

4.2 Analysis 2

In this analysis, treatment (eu) is not considered. So caseMatchPD simply finds the most similar cases.

Set parameters

```
dropvars <- c("year", "Aid_All",
              "un_region",
              "eu",
              "GDP",
              "Population"
              )
id <- "recipient"
```

Run caseMatchPD

```
outcome <- caseMatchPD(data, id, year=20, dropvars=dropvars, method="mahalanobis")
```

```
outcome[1:15,]
```

```
##            id1           id2 distance
## 1  Netherlands United States 142.2775
## 2     Honduras United States 166.0666
## 3      Namibia United States 171.4529
## 4      Hungary United States 171.8009
## 5   Kazakhstan United States 176.4491
## 6      Ecuador United States 177.6402
## 7    Australia United States 178.0851
## 8      Hungary        Turkey 178.9377
## 9       Turkey United States 178.9668
## 10    Bulgaria   Netherlands 180.7274
## 11   Australia       Tunisia 181.4500
## 12    Bulgaria       Uruguay 181.6222
## 13      Canada   Netherlands 182.3255
## 14    Slovenia United States 182.3778
## 15     Myanmar United States 183.3322
```

4.3 Analysis 3

Treatment (whether joined eu or not) is now considered.

Set parameters

```
dropvars_T <- c("year", "Aid_All",
              "un_region",
              "GDP",
              "Population"
              )
id <- "recipient"
```

Run caseMatchPD

```
outcome_T1 <- caseMatchPD(data, id, year=20, dropvars=dropvars_T, method="mahalano
bis", treatment="eu")
```

```
outcome_T1[1:15,]
```

```
##            id1           id2 distance
## 1  Netherlands United States 142.2775
## 2      Hungary United States 171.8009
## 3      Hungary        Turkey 178.9377
## 4     Bulgaria       Uruguay 181.6222
## 5  Netherlands        Canada 182.3255
## 6     Slovenia United States 182.3778
## 7  Netherlands      Botswana 184.9887
## 8      Hungary         China 185.1215
## 9       Poland       Uruguay 185.6218
## 10 Netherlands       Myanmar 187.1292
## 11 Netherlands     Australia 187.5382
## 12     Austria United States 191.2780
## 13     Hungary   El Salvador 192.0224
## 14      Poland      Botswana 192.1417
## 15 Netherlands        Panama 192.2674
```

This setting returns negative Mahalanovis distance

```
dropvars_T <- c("year", "Aid_All",
                "un_region",
                "GDP"
                )
id <- "recipient"
```

```
outcome_T2 <- caseMatchPD(data, id, year=20, dropvars=dropvars_T, method="mahalano
bis", treatment="eu")
```

```
outcome_T2[1:15,]
```

```
##                id1             id2   distance
## 1          Estonia           India -936.39535
## 2           Latvia           China -506.34436
## 3          Estonia   United States -393.01513
## 4         Slovenia   United States -237.58417
## 5  Slovak Republic           China -116.82655
## 6          Estonia            Iran -109.66254
## 7          Romania            Oman  -86.78132
## 8  Slovak Republic   United States  -86.14110
## 9          Romania           Gabon  -81.55229
## 10         Romania Solomon Islands  -67.60329
## 11         Estonia       Argentina  -63.69049
## 12         Estonia          Russia  -38.80663
## 13         Estonia     Philippines  -18.71867
## 14        Slovenia        Thailand  -18.21322
## 15     Netherlands Solomon Islands  -17.55908
```

Is there any way to avoid negative values? Or is it possible to just ignore them and take positive values as results?