# R Notebook

*Jiangshan Lin & Jinyu Wang (based on Jingmin Sun's imputation)*

*10/01/2019*

## Contents

**Notebook User Guide** **1**

**Start of imputation part:** **1**

## Notebook User Guide

Executing this R notebook requires some subset of the following packages:

- `ggplot2`
- `tidyverse`
- `glue`
- `VIM`
- `DMwR`
- `Amelia`
- `randomForest`
- `nnet`
- `urbnmapr`
- `viridis`
- `matrixStats`

These will be installed and loaded as necessary.

## Start of imputation part:

We will be using following functions:

- `cdc.mort.mat`:
- for CDC data if state not declared, tmp is state data, else use all data,
- drop all trash data (county fips is zero)
- use only county_fips, death_rate and period
- change data fram from long to wide for calculation:

```
func_demo_long <- data.frame(
  County = c("Autauga", "Escambia", "Adair"),
  county_fips = c("01001", "12033", "40001"),
  death_rate = c("34.97237","50.95881","50.16618"),
  period = c("2000-2002","2006-2008","2003-2005")
)

kable(func_demo_long) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

| County | county_fips | death_rate | period |
|---|---|---|---|
| Autauga | 01001 | 34.97237 | 2000-2002 |
| Escambia | 12033 | 50.95881 | 2006-2008 |
| Adair | 40001 | 50.16618 | 2003-2005 |

```
func_demo_wide <- data.frame(
  County = c("Autauga", "Escambia", "Adair"),
  county_fips = c("01001", "12033", "40001"),
  `2000-2002` = c("34.97237","...","..."),
  `2003-2005` = c("...","...","50.16618"),
  `2006-2008` = c("...","50.95881","...")
)

kable(func_demo_wide) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

| County | county_fips | X2000.2002 | X2003.2005 | X2006.2008 |
|---|---|---|---|---|
| Autauga | 01001 | 34.97237 | ... | ... |
| Escambia | 12033 | ... | ... | 50.95881 |
| Adair | 40001 | ... | 50.16618 | ... |

```
#   county_fips, death_rate, period                 county_fips period1 period2
# a                                           a
# b                                  =====>   b
# c                                           c
```

```
cdc.mort.mat <- function(cdc.data.long, state.abbr, death.cause = "Despair") {

  tmp <- cdc.data.long
  if (state.abbr != "ALL") {
    tmp <- dplyr::filter(cdc.data.long, state_abbr == state.abbr)
  }

  dplyr::filter(tmp, death_cause == death.cause) %>%
    tidyr::drop_na(county_fips) %>%
    dplyr::select(county_fips, death_rate, period) %>%
    tidyr::spread(key = period, value = death_rate) %>%
    dplyr::arrange(county_fips)
}
```

cdc.mort.state.mat: same as cdc.mort.mat

```
cdc.mort.state.mat <- function(cdc.data.state.long, state.abbr, death.cause = "Despair") {

  tmp <- cdc.data.state.long
  if (state.abbr != "ALL") {
    tmp <- dplyr::filter(cdc.data.state.long, state_abbr == state.abbr)
  }

  dplyr::filter(tmp, death_cause == death.cause) %>%
    tidyr::drop_na(state_fips) %>%
    dplyr::select(state_fips, state_abbr, death_rate, period) %>%
    tidyr::spread(key = period, value = death_rate) %>%
    dplyr::arrange(state_fips)
```

```
}
```

cdc.pop.mat: In order to get population

```
cdc.pop.mat <- function(cdc.data.long, state.abbr, death.cause = "Despair") {
  tmp <- cdc.data.long
  if (state.abbr != "ALL") {
    tmp <- dplyr::filter(cdc.data.long, state_abbr == state.abbr)
  }
}
```

We will impute the determinant matrix directly, using the package `Amelia` (Explanation in Appendix), which can help us to set the bounds of each determinant. However, we don't know the bounds for each determinant exactly, so the imputed determinants matrix may not be useful. But we impute them anyway and set the bound of each determinant to their existing range. We do the Amelia fitting, and produce 2 imputed output get mean as final result. If both imputation fails, replace it by state mean or random rate that is closer to other non-missing mortality rates in this county. If one imputation fails, use the success one. The function will finally return a calculated R data with the same format as the input data.

cdc.impute: arguments: 1 cdc.data.long: R data get from Loader_CDC.R 2 cdc.data.state.long: R data get from Loader_CDC.R 3 state.abbr: "All" or abbreviate of a state 4 death.cause

Amelia: low bound: 0 high_bound: random(0~9)/(population*10^5) if population is NA set it to Inf calculate times: 2

```
cdc.impute <- function(cdc.data.long, cdc.data.state.long, state.abbr, death.cause = "Despair") {

  # > cdc.data.wide <- cdc.mort.mat(cdc.data, "ALL", "Despair")
  # > cdc.supp.wide <- cdc.pop.mat(cdc.data, "ALL", "Despair")
  # > cdc.data.state.wide <- cdc.mort.state.mat(cdc.data.despair.state, "ALL", "Despair")

  cdc.data.wide <- cdc.mort.mat(cdc.data.long, state.abbr, death.cause)
  cdc.data.wide[is.na(cdc.data.wide)] <- NaN

  cdc.supp.wide <- cdc.pop.mat(cdc.data.long, state.abbr, death.cause)

  cdc.data.state.wide <- cdc.mort.state.mat(cdc.data.state.long, state.abbr, death.cause)

  county.data <- unique( cdc.data %>%
    dplyr::select(state_name, state_abbr, county_name, county_fips, urban_2013)
    )

  rownames(cdc.data.state.wide) <- cdc.data.state.wide$state_abbr

  hi_bound <- as.data.frame(matrix(Inf, nrow=nrow(cdc.data.wide), ncol=6))   # fips as row ,period as c

  colnames(hi_bound) <- colnames(cdc.data.wide)[-1]
  rownames(hi_bound) <- cdc.data.wide$county_fips

  random1 <- hi_bound
  random2 <- hi_bound

  set.seed(123)

  miss_index <- which(is.na(cdc.data.wide[,-1]),arr.ind=TRUE)
  get_pop_index <- t(matrix(c(0,8), nrow=ncol(miss_index), ncol=nrow(miss_index)))
```

```r
hi_bound[miss_index] <-
  dplyr::if_else( is.na(cdc.supp.wide[(miss_index+get_pop_index)]), Inf,
                  9 / as.numeric( cdc.supp.wide[(miss_index+get_pop_index)] ) * 10^5)

random1[miss_index] <-
  dplyr::if_else( is.na(cdc.supp.wide[(miss_index+get_pop_index)]), Inf,
                  sample(c(0:9), 1, replace=TRUE) /
                    as.numeric( cdc.supp.wide[(miss_index+get_pop_index)] ) * 10^5)

random2[miss_index] <-
  dplyr::if_else( is.na(cdc.supp.wide[(miss_index+get_pop_index)]), Inf,
                  sample(c(0:9), 1, replace=TRUE) /
                    as.numeric( cdc.supp.wide[(miss_index+get_pop_index)] ) * 10^5)

# imputation

cdc.states.split <- split(cdc.data.wide, cdc.supp.wide$state_abbr) #splite by state

complt.df <- data.frame()

# for each state
for (i in 1:length(cdc.states.split)) {

  state_name <- names(cdc.states.split[i])
  state <- cdc.states.split[[i]]
  rownames(state) <- state$county_fips

  # category a) complete informations

  complt <- state[complete.cases(state), -1]

  # category b) have missing value and have more than 4 valid value

  part <- state[ !complete.cases(state) & rowSums(is.na(state)) <= 4 , -1]

  state_mean <- cdc.data.state.wide[state_name,c(-1,-2)]

  #if (nrow(complt) == nrow(state)) {
  #  complt.df <- rbind(complt.df, complt)
  #  next
  #}

  if (nrow(part) > 0) {
    for (j in 1:nrow(part)) {
      county <- part[j,]
      complt <- rbind(complt, county)

      miss_i <- which(is.na(county))
      lo_bound <- rep(0, length(miss_i))
      up_bound <- as.numeric(hi_bound[rownames(county), miss_i])

      bound <- cbind(miss_i, lo_bound, up_bound)
```

```
      county_nonmiss_mean <- rowMeans(county[,-miss_i])

      amelia_impute <- Amelia::amelia(complt, m=2, parallel="multicore", bounds=bound, p2s=0)
      amelia_out1 <- amelia_impute$imputations$imp1
      amelia_out2 <- amelia_impute$imputations$imp2

      amelia_list <- list(amelia_out1, amelia_out2)

      amelia_result <- Reduce("+", amelia_list) / length(amelia_list)

      # If both imputation fails, replace it by state mean or random rate that is closer
      # to other non-missing mortality rates in this county
      if (length(amelia_result) == 0) {

        less_i <- which( is.na(complt[rownames(county),]) &
                         (state_mean <= hi_bound[rownames(county),]) )
        random_i <- which( is.na(complt[rownames(county),]) &
                           (state_mean > hi_bound[rownames(county),]) )

        complt[rownames(county),less_i] <- state_mean[,less_i]

        complt[rownames(county),random_i] <-
          sapply(rbind(random1[rownames(county),], random2[rownames(county),]),
                 function(x) {
                   x[ which( abs(x-county_nonmiss_mean) ==
                               min(abs(x-county_nonmiss_mean) ) ) ]
                 }
          )[random_i]
      }
      # If one of the imputations failed, use the successful one
      else if (xor(length(amelia_out1) == 1, length(amelia_out2) == 1)) {
        if (length(amelia_out1) == 1){
          amelia_result <- amelia_out2
        }
        else{
          amelia_result <- amelia_out1
        }
        complete <- amelia_result
      }
      # If both imputations succeed, use the mean output
      else {
        complt <- amelia_result
      }
    }
  }

  # category c)

  miss_more <- state[ rowSums(is.na(state)) > 4 , -1]

  if (nrow(miss_more) > 0) {
    for (k in 1:nrow(miss_more)) {
      county <- miss_more[k,]
```

```
      less_i <- which( is.na(county) &
                         (state_mean <= hi_bound[rownames(county),]) )
      random_i <- which( is.na(county) &
                           (state_mean > hi_bound[rownames(county),]) )
      county[,less_i] <- state_mean[,less_i]
      county[,random_i] <-
        colMeans(rbind(random1[rownames(county),],
                       random2[rownames(county),])
        )[random_i]
      complt <- rbind(complt, county)
    }
  }

  complt.df <- rbind(complt.df, complt)
  print(i)
}

final.df <-

  # combine all info for counties (fips, state, death_num, population, etc.)
  dplyr::mutate(complt.df,county_fips = rownames(complt.df)) %>%
  dplyr::inner_join(dplyr::select(cdc.supp.wide, -state_abbr), by="county_fips") %>%
  dplyr::inner_join(county.data, by="county_fips") %>%

  # before gather, combine all death info, easier to structure
  tidyr::unite(`2000-2002`, `2000-2002`, `2000-2002.death_num`, `2000-2002.population`, sep=",") %>%
  tidyr::unite(`2003-2005`, `2003-2005`, `2003-2005.death_num`, `2003-2005.population`, sep=",") %>%
  tidyr::unite(`2006-2008`, `2006-2008`, `2006-2008.death_num`, `2006-2008.population`, sep=",") %>%
  tidyr::unite(`2009-2011`, `2009-2011`, `2009-2011.death_num`, `2009-2011.population`, sep=",") %>%
  tidyr::unite(`2012-2014`, `2012-2014`, `2012-2014.death_num`, `2012-2014.population`, sep=",") %>%
  tidyr::unite(`2015-2017`, `2015-2017`, `2015-2017.death_num`, `2015-2017.population`, sep=",") %>%

  # gather we convert data from wide to long
  tidyr::gather(key="period", value="death_info", 1:6) %>%

  # seperate the death info so we have three columns
  tidyr::separate("death_info", c("death_rate","death_num","population"), sep=",") %>%

  # make int death_num
  dplyr::mutate( death_rate = as.numeric(death_rate),
                 death_num = as.numeric(death_num),
                 population = as.numeric(population) ) %>%

  # convert the death_rate to the nearest integer num/pop rate
  dplyr::mutate( death_num = round(death_rate*population/10^5) ) %>%
  dplyr::mutate(
    death_rate = dplyr::if_else(is.na(population), death_rate, death_num/population*10^5),
    death_cause = death.cause ) %>%

  # Rearrangement
  dplyr::arrange(county_fips) %>%
  dplyr::select(
    # Rearrangement
```

```
      period, state_name, state_abbr, county_name,
      county_fips, urban_2013, population, death_num, death_rate, death_cause
    ) %>%
    as.data.frame()

}
```