MCCT Dataset Development

Nicholus Tint Zaw

2022-06-30

Purpose of the document

This document organized the record of the step-by-step process involved in the development of the MCCT baseline dataset for the respective module, which includes the interested outcome variables for analysis.

Data Cleaning

Perform the data cleaning based on the `30-clean_hh.R` function package. As the data collection used two different surveys from HH and anthro, the data cleaning was performed separately for each respective raw data. Please note that the function did not work perfectly, and I am not sure why. Therefore, the individual syntax applied in the function package is used to perform data cleaning instead of the function in finally.

HH data

```
# load datasets
load('hh.rda')
load('checks.rda')

# load function
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/30-clean_hh.R")

# data cleaning
# hh <- clean_hh(df = hh, checks = checks)
# function is not working perfectly. run individual commend instead of function

hh <- hh[row.names(hh) != 350, ]
hh <- hh[hh$geo_villward != "", ]
hh[hh$geo_vill == "168652" & !is.na(hh$geo_vill), "geo_rural"] <- "2"


for(i in checks$id) {
    hh[checks$index[checks$id == i],
       checks$variable[checks$id == i]] <- checks$newvalue[checks$id == i]
}
```

Anthro data

```
load('anthroDF.rda')
load('anthroChecks.rda')

# data cleaning
# data cleaning - run by individual commend line from function package
anthroDF[anthroDF$geo_vill == "168652" & !is.na(anthroDF$geo_vill), "geo_rural"] <- "2"

for(i in anthroChecks$id) {
    anthroDF[anthroChecks$index[anthroChecks$id == i],
             anthroChecks$variable[anthroChecks$id == i]] <- anthroChecks$newvalue[anthroChecks$id == i]
}
```

Outcome indicators Calculation

In this session, I calculated reported indicators from the MCCT baseline analysis based on our secondary data analysis's individual (interested) modules.

The process was simple. First, load the necessary raw `rda` files and then import the `r-function` script file to the `r` global environment. Then, apply the respective function to get the newly calculated outcome indicators variable. Please note that the function application for each module was based on the example session from the respective `r-function` file. For example, the calculation workflow for the child immunization and child health-seeking behavior were applied based on the example function usage session from the `06-recode_chealth.R` file. After calculating the respective module, each newly calculated dataset was saved as STATA `dta` file.

Child Vaccinations & Child health

Load the datasets

```
load('childHealth.rda')
load('hhMembers.rda')
```

Load the functions

```
# run the 06 r script file
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/06-recode_chealth.R")
```

Variables calculation

```
# creat the dataset with eligable children
chealth <- create_chealth(df = childHealth, x = hh, y = hhMembers)

# cimbined all child health module indicators
child_health <- recode_chealth(df = chealth)
```

```
child_health <- janitor::clean_names(child_health)
chealth <- janitor::clean_names(chealth)

write_dta(chealth, file.path(getwd(), "stata_dta", "chealth.dta"))
write_dta(child_health, file.path(getwd(), "stata_dta", "child_health_all.dta"))
```

Child Anthro

Load the datasets

```
load('childAnthro.rda')
```

Load the functions

```
# run the 06 r script file
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/03-recode_anthro.R")
```

Variables calculation

```
# created anthro indicator datset
childanthro <- recode_anthro(df = create_canthro(df = childAnthro, x = anthroDF))
```

```
## ================================================================================
## ================================================================================
## ================================================================================
```

```
childanthro <- janitor::clean_names(childanthro)

write_dta(childanthro, file.path(getwd(), "stata_dta", "child_anthro_all.dta"))
```

Maternal health (pregnancy)

load the datasets

```
load('anc1.rda')
load('anc2.rda')
load('hhMembers.rda')
```

load the functions

```
# run the 06 r script file
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/05-recode_anc.R")
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/14-recode_delivery.R")
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/17-recode_pnc.R")
```

Variables calculation

```
# Recode anc indicators for currently pregnant women
x_current <- create_anc(df = anc1, x = hh, y = hhMembers, status = "current")
anc_current <- recode_anc(df = x_current, status = "current")

# Recode anc indicators for non-pregnant women
x_past <- create_anc(df = anc2, x = hh, y = hhMembers, status = "past")
anc_past <- recode_anc(df = x_past, status = "past")

# Recode birth/delivery indicators
delivery <- recode_birth(df = x_past)

# Recode postnatal care indicators
pnc <- recode_pnc(df = x_past)

anc_current <- janitor::clean_names(anc_current)
anc_past <- janitor::clean_names(anc_past)
delivery <- janitor::clean_names(delivery)
pnc <- janitor::clean_names(pnc)

write_dta(anc_current, file.path(getwd(), "stata_dta", "anc_current.dta"))
write_dta(anc_past, file.path(getwd(), "stata_dta", "anc_past.dta"))
write_dta(delivery, file.path(getwd(), "stata_dta", "delivery.dta"))
write_dta(pnc, file.path(getwd(), "stata_dta", "pnc.dta"))

anc1 <- janitor::clean_names(anc1)
anc2 <- janitor::clean_names(anc2)

write_dta(anc1, file.path(getwd(), "stata_dta", "anc1.dta"))
write_dta(anc2, file.path(getwd(), "stata_dta", "anc2.dta"))
```

Coping Strategies

load the datasets

This module will use the existing loaded `hh` module and not be required to load other datasets.

Load the functions

```
# run the 06 r script file
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/10-recode_csi.R")
```

Variables calculation

```
# consumption-based Coping Strategies Index
rcsi <- recode_csi_consumption(df = hh)
```

```r
# livelihoods-based Coping Strategies Index
lcsi <- recode_csi_livelihoods(df = hh)

rcsi <- janitor::clean_names(rcsi)
lcsi <- janitor::clean_names(lcsi)

write_dta(rcsi, file.path(getwd(), "stata_dta", "rcsi.dta"))
write_dta(lcsi, file.path(getwd(), "stata_dta", "lcsi.dta"))
```

PPI and Split into Quintiles

Load the datasets

This module will use the existing loaded `hh` module and not be required to load other datasets.

Load the functions

```r
# run the 06 r script file
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/13-recode_ppi.R")
source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/25-split_to_quintiles.R")
```

Variables calculation

```r
# Recode poverty probability index indicators
ppiDF <- recode_ppi(df = hh)
ppiDF <- split_to_quintiles(df = ppiDF)

ppiDF <- janitor::clean_names(ppiDF)

write_dta(ppiDF, file.path(getwd(), "stata_dta", "ppiDF.dta"))
```

Weight Calculation

The `r` file called `23-calculate_weight.R` did not include the calculation of weight applied in the MCCT baseline analysis. It just described the function. We need to calculate the input parameters required to use that function. I found the `weight.R` file from the `data-raw` folder to calculate those required function input parameters. The syntax from that file calculates the required information for each study stratum to calculate the weight. The blow was the detailed syntax execution.

```r
options(stringsAsFactors = FALSE)

pop <- read.csv("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/data-raw/pop/popMyanmar.csv

x <- pop[ , c("geo_ward_vt_eho", "population")]
y <- hh[hh$sample_component == 1, c("geo_state", "geo_rural", "geo_villward")]
hhWeight <- hh[hh$sample_component == 1, ]
z <- merge(x, y, by.x = "geo_ward_vt_eho", by.y = "geo_villward", all.y = TRUE)
```

```r
hhWeight <- merge(x, hhWeight, by.x = "geo_ward_vt_eho", by.y = "geo_villward", all.y = TRUE)

### get median population size of clusters in a state

medianPop <- tapply(X = z$population, INDEX = z$geo_state, FUN = median, na.rm = TRUE)

z$population[is.na(z$population) & z$geo_state == "MMR002"] <- medianPop[1]
z$population[is.na(z$population) & z$geo_state == "MMR003"] <- medianPop[2]

### get weights for MMR002 and geo_rural == 0 (Rural)

z1 <- z[z$geo_state == "MMR002" & z$geo_rural == "0", ]

z1 <- aggregate(x = z1[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z1$geo_ward_vt_eho), FUN = "unique")

z1$totalPop <- sum(z1$population)

### get weights for MMR002 and geo_rural == 1 (Urban)

z2 <- z[z$geo_state == "MMR002" & z$geo_rural == "1", ]

z2 <- aggregate(x = z2[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z2$geo_ward_vt_eho), FUN = "unique")

z2$totalPop <- sum(z2$population)


### get weights for MMR002 and geo_rural == 2 (EHO)

z3 <- z[z$geo_state == "MMR002" & z$geo_rural == "2", ]

z3 <- aggregate(x = z3[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z3$geo_ward_vt_eho), FUN = "unique")

z3$totalPop <- sum(z3$population)


### get weights for MMR003 and geo_rural == 0 (Rural)

z4 <- z[z$geo_state == "MMR003" & z$geo_rural == "0", ]

z4 <- aggregate(x = z4[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z4$geo_ward_vt_eho), FUN = "unique")

z4$totalPop <- sum(z4$population)

### get weights for MMR003 and geo_rural == 1 (Urban)

z5 <- z[z$geo_state == "MMR003" & z$geo_rural == "1", ]

z5 <- aggregate(x = z5[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z5$geo_ward_vt_eho), FUN = "unique")
```

```r
z5$totalPop <- sum(z5$population)


### get weights for MMR003 and geo_rural == 2 (EHO)

z6 <- z[z$geo_state == "MMR003" & z$geo_rural == "2", ]

z6 <- aggregate(x = z6[ , c("geo_state", "geo_rural", "geo_ward_vt_eho", "population")],
                by = list(z6$geo_ward_vt_eho), FUN = "unique")

z6$totalPop <- sum(z6$population)


zz <- data.frame(rbind(z1, z2, z3, z4, z5, z6))

### Kayin pop - 1055359; Kayah pop - 286627

nClusters <- vector(mode = "numeric", length = nrow(zz))
nClusters[zz$geo_state == "MMR002" & zz$geo_rural == "0"] <- 24
nClusters[zz$geo_state == "MMR002" & zz$geo_rural == "1"] <- 24
nClusters[zz$geo_state == "MMR002" & zz$geo_rural == "2"] <- 26
nClusters[zz$geo_state == "MMR003" & zz$geo_rural == "0"] <- 27
nClusters[zz$geo_state == "MMR003" & zz$geo_rural == "1"] <- 17
nClusters[zz$geo_state == "MMR003" & zz$geo_rural == "2"] <- 27

zz$nClusters <- nClusters

clusterSize <- data.frame(table(z$geo_ward_vt_eho))
names(clusterSize) <- c("geo_villward", "size")

source("C:/Users/Nicholus Tint Zaw/Documents/GitHub/myanmarMCCTdata/R/23-calculate_weights.R")

zz <- merge(zz, clusterSize,
            by.x = "geo_ward_vt_eho", by.y = "geo_villward",
            all.x = TRUE)

zz$weights <- get_weights(n = zz$population,
                          N = zz$totalPop,
                          m = zz$nClusters,
                          c = zz$size)

zz <- janitor::clean_names(zz)

write_dta(zz, file.path(getwd(), "stata_dta", "svy_weight.dta"))
```

If you want to merge that weight dataset with the hh or anthro dataset, please use `geo_ward_vt_eho` as a merge key variable. Please note that the whole weight calculation that VI did was based on the `component - 1` sample, not including the `component - 2` sample, which was applied for RDD analysis. (you can use the `sample_component` variable to identify the sample component in the dataset) Therefore, before merging, please drop the`component - 2` observation from the dataset. In this case, you might encounter one issue in the anthro dataset because there was no variable to identify which observation baseline to what sample component. In this case, merge the `hh` dataset with the `anthro` dataset first to get the sample component information. However, you will not get to merge all anthro observations with the `hh` dataset.

Some observations from the anthro dataset could not match the `hh` observation because the field team failed to record the correct household information in the `anthro survey data`.

The cleaned household dataset was saved in the STATA format using the code below. Before converting into STATA format, drop the very long name variable, which was not accepted in STATA format.

```
hh <- dplyr::select(hh, -c("SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.support
                           "SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.suppo
                           "SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.suppo
                           "SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.suppo
                           "SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.suppo
                           "SET.OF.consent.hh_grp.hh_income_grp.hh_reg_income_grp.reg_income_grp.suppo

hh <- janitor::clean_names(hh)

write_dta(hh, file.path(getwd(), "stata_dta", "hh.dta"))
```

Dataset Merging

We still need to perform dataset merging to create to combine the respective module `master data` and the `newly created dataset`. Here, I refer to `master data` as the original variables from the respective module dataset and the `newly created dataset` as the output data frame, which contains only the reported outcome variable. Therefore, we need to merge those two datasets to get one complete dataset for each module. Then, merge again with `hh` dataset to get the household information. However, those dataset merging will be performed with STATA dofile as it is easy to trace the result of merging.