Using the 'TABUS' C++ Reshaper

Christian "Small-Plate" Testa

```
library(MITUS)
devtools::load_all()
```

```
## Loading tabus
```

```
model_load('US')
load(system.file('US/US_parAll10_2018-12-10.rda', package='MITUS')) # input parameters
out <- OutputsZint(1,
        ParMatrix=ParMatrix,
        startyr=1950,
        endyr=2050,
        Int1=0,
        Int2=0,
        Int3=0,
        Int4=0,
        Int5=0,
        Scen1=0,
        Scen2=0,
        Scen3=0)
```

Define the original `format_as_restab` for double-checking our results.

Generate ResTab (4D) objects

```
ResTabC <- format_as_restab(out)
```

Generate res_tab2 dataframe (without data yet)

```
make_empty_res_tab2 <- function() {
    # Specify the levels of each dimension to the data
    CatList <- list()
    CatList[[1]] <- c(
        "ltbi_000s",
        "pct_ltbi",
        "tb_incidence_000s",
        "tb_incidence_per_mil",
        "tb_mortality_000s",
        "tb_mortality_per_mil")
    CatList[[2]] <- c('base_case')
    CatList[[3]] <- c("all_populations","usb_population","fb_population")
    CatList[[4]] <- c("0-4",paste(0:8*10+5,1:9*10+4,sep="-"),"95+")
    CatList[[5]] <- c("absolute_value","pct_basecase_same_year","pct_basecase_2016")
    CatList[[6]] <- 2018:2049
    # CatList[[7]] <- c("mean","ci_low","ci_high")

    # Make the specified levels integer-leveled factors
    CatList_factors <- lapply(CatList, function(x) {
        factor(x=1:length(x), levels=1:length(x), labels=x) })

    # Turn the integer-factors into all possible combinations in a dataframe
    # with an extra column of NA values for a 'value' column
```

```
    res_tab2 <-  cbind(expand.grid(CatList_factors),NA)

    # Name the columns
    colnames(res_tab2) <- c(
        "outcome",
        "scenario",
        "population",
        "age_group",
        "comparator",
        "year",
        "value")

    return(res_tab2)
}
```

Specify the original reshaper function in R

```
original_reshaper <- function(ResTabC) {
    res_tab2 <- make_empty_res_tab2()

  for (it in 1:nrow(res_tab2)) {
    i1 <- as.integer(res_tab2[it,'outcome'])
    i2 <- as.integer(res_tab2[it, 'scenario'])
    i3 <- as.integer(res_tab2[it, 'population'])
    i4 <- as.integer(res_tab2[it, 'age_group'])
    i5 <- as.integer(res_tab2[it, 'comparator'])
    i6 <- as.integer(res_tab2[it, 'year'])
    i7 <- as.integer(res_tab2[it, 'statistic'])

    res_tab2[it, 'value'] <-
      switch(
        as.character(res_tab2[it, 'comparator']),
        'absolute_value' = {
          ResTabC[[i3]][i2, i6, i1 + 1, i4]
        },
        'pct_basecase_same_year' = {
          ResTabC[[i3]][i2, i6, i1 + 1, i4] /
            ResTabC[[i3]][1, i6, i1 + 1, i4] * 100
        },
        'pct_basecase_2016' = {
          ResTabC[[i3]][i2, i6, i1 + 1, i4] /
            ResTabC[[i3]][1, i6, 1, i4] * 100
        }
      )
  }
    return(res_tab2)
}
```

Now let's test to see if we get the same outcomes from each of our reshapers.

```
res_tab2_1 <- original_reshaper(ResTabC)

# Make an empty res_tab2 for comparison to res_tab2_1
library(dplyr)
res_tab2_2 <- make_empty_res_tab2()
```

```
# convert to a matrix with integer values where levels would be
res_tab2_2 %<>% mutate_if(is.factor, as.integer) %>% as.matrix

# Import the C++ Reshaper
library(inline)
# We use readr to ensure that the UTF8 encoding of the .cpp file is preserved,
# newlines are interpreted properly, etc. Other approaches, such as using readLines
# don't automatically respect newlines and tabs properly.
cpp_reshaper <- cxxfunction(
    signature(ResTab='numeric', ResTabus='numeric', ResTabfb='numeric', res_tab2 = 'numeric'),
    plugin='Rcpp',
    body=readr::read_file(
        system.file('inline_cpp/format_restab2.cpp', package='tabus')))

res_tab2_2 <- cpp_reshaper(ResTabC$ResTab, ResTabC$ResTabus, ResTabC$ResTabfb, res_tab2_2)
```

Now we do our final test! Do we get exactly the same results? 0 indicates yes!

```
if(max(res_tab2_1[,'value'] - res_tab2_2[,'value']) == 0) {
    print('SUCCESS!')
} else print(':(')
```

```
## [1] "SUCCESS!"
```

If one wants to format the `res_tab2_2` object to be filled with characters instead of integers, we just have to re-factor it.

```
    # Specify the levels of each dimension to the data
    CatList <- list()
    CatList[[1]] <- c(
        "ltbi_000s",
        "pct_ltbi",
        "tb_incidence_000s",
        "tb_incidence_per_mil",
        "tb_mortality_000s",
        "tb_mortality_per_mil")
    CatList[[2]] <- c('base_case')
    CatList[[3]] <- c("all_populations","usb_population","fb_population")
    CatList[[4]] <- c("0-4",paste(0:8*10+5,1:9*10+4,sep="-"),"95+")
    CatList[[5]] <- c("absolute_value","pct_basecase_same_year","pct_basecase_2016")
    CatList[[6]] <- 2018:2049


# Format as a dataframe
res_tab2_2 <- as.data.frame(res_tab2_2)

# 'Factorize' each column
for (i in 1:6) {
    res_tab2_2[,i] <- factor(res_tab2_2[,i], labels = CatList[[i]])
}

# Print for validation and comfort of mind
head(res_tab2_2)
```

```
##                outcome  scenario      population age_group     comparator
## 1            ltbi_000s base_case all_populations       0-4 absolute_value
```

```
## 2             pct_ltbi base_case all_populations       0-4 absolute_value
## 3    tb_incidence_000s base_case all_populations       0-4 absolute_value
## 4 tb_incidence_per_mil base_case all_populations       0-4 absolute_value
## 5    tb_mortality_000s base_case all_populations       0-4 absolute_value
## 6 tb_mortality_per_mil base_case all_populations       0-4 absolute_value
##   year        value
## 1 2018 1.605267e+02
## 2 2018 7.214754e-01
## 3 2018 6.517660e+00
## 4 2018 2.929314e-02
## 5 2018 1.628483e-01
## 6 2018 7.319098e-04
```