Bayesian Subnational Estimation using Complex Survey Data

Introduction to R

Zehang Richard Li

Introduction to R

This document is an R Markdown file. It is a great way to combine data analysis with reports. To learn more about R Markdown, check out the online book at https://bookdown.org/yihui/rmarkdown/.

Vector

In R, we use `<-` or `=` to assign the value on the right hand side to an object on the left hand side. For example,

```
x <- "Hello"
y <- c(1, 2, 3)
```

The `c(...)` notation defines a vector, i.e., a collection of elements of the same type. For some special vectors, there are faster ways to define them. For example,

```
x <- 1:100
```

Matrix and array

Similarly, there are two-dimensional extension of vector (matrix), and higher-dimensional extensions of vector (array).

```
x <- matrix(1:24, nrow = 3)
y <- array(1:24, dim = c(2, 3, 4))
```

Many operators in R are vectorized, meaning that operations occur in parallel in certain R objects.

```
x <- x^2 + 1
y <- log(y)
```

Data frame

A tabular form of data where each column may contain different types of values (i.e., similar to a spreadsheet), is called data frame in R.

```
x <- data.frame(ID = 1:4, name = c("A", "B", "C", "D"), color = factor(c("red",
    "blue", "blue", "green")))
```

Notice that although factors look similar to characters, but they can behave very differently in action. I encourage new users of R to read the section on 'R Nuts and Bolts' of this online book R Programming for Data Science (if not the whole book) for more details.

List

Another commonly used data structure is 'list'. It creates a list of elements that are not necessarily of the same type, dimension, or have the same attributes. For example,

```
y <- list(apple = 1:20, orange = x)
```

Subsetting

To subset a vector or matrix, we can use element/column/row index or names, for example,

```
x[1:2, ]
```

```
##   ID name color
## 1  1    A   red
## 2  2    B  blue
```

```
x[1:2, 2:3]
```

```
##   name color
## 1    A   red
## 2    B  blue
```

```
x[, c("ID", "name")]
```

```
##   ID name
## 1  1    A
## 2  2    B
## 3  3    C
## 4  4    D
```

To subset a list is very similar,

```
y[[1]]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
y[["orange"]]
```

```
##   ID name color
## 1  1    A   red
## 2  2    B  blue
## 3  3    C  blue
## 4  4    D green
```

```
y$orange
```

```
##   ID name color
## 1  1    A   red
## 2  2    B  blue
## 3  3    C  blue
## 4  4    D green
```

R packages

Installation

To install packages that are hosted on CRAN, we only need to use the `install.packages` function.

```
install.packages("ggplot2", dep = TRUE)
install.packages("rgdal", dep = TRUE)
install.packages("survey", dep = TRUE)
install.packages("SUMMER", dep = TRUE)

install.packages("readstata13", dep = TRUE)
install.packages("gridExtra", dep = TRUE)
```

Some packages not hosted on CRAN. For example, the INLA package that we will be using for space-time smoothing is hosted on its own website.

```
install.packages("INLA", repos = c(getOption("repos"), INLA = "https://inla.r-inla-download.org/R/stable"))
```

Load R packages

Once a package is installed, you can load it into your workspace. Then you can use all the functions and data provided in that package.

```
library(SUMMER)
library(ggplot2)
```

You can check your loaded packages in the workspace by

```
sessionInfo()
```

Maps

We download the Kenya subnational boundaries (both province-level and county-level) from the DHS spatial data repository. shapefiles typically consist of at least these three files:

- The first file (*.shp) contains the geography of each shape.
- The second file (*.shx) is an index file which contains record offsets.
- The third file (*.dbf) contains feature attributes with one record per feature.

In this set of notes, I organize the files as the following directory structure. You may change to something different, and update the file directory path accordingly.

- Projct_main_folder
  - `vignettes` folder (my working directory)
    - some_R_script.R
  - `data` folder
    - `shapefiles`: map shapefiles
      - .shp, .shx, .dbf, …
    - `ZZBR62DT`: DHS model dataset
    - …

```
library(rgdal)
geo <- readOGR("../data/shapefiles/sdr_subnational_boundaries.shp", verbose = FALSE)
```

A simple visualization of the map can be called directly by

```
plot(geo)
```

Design-based estimates of U5MR

Full birth history data

We now download the DHS model datasets from . We will be using the Stata dataset (.dta) format of the `Births Recode` file. For illustration, we only consider the following 4 regions.

```
library(readstata13)
data <- read.dta13("../data/ZZBR62DT/ZZBR62FL.DTA")
data <- subset(data, v139 %in% c("region 1", "region 2", "region 3", "region 4"))
head(data[, 1:10])
```

```
##            caseid bidx v000 v001 v002 v003 v004    v005 v006 v007
## 1         1  1  2    1  ZZ6    1    1    2    1 1057703    6 2015
## 2         1  1  2    2  ZZ6    1    1    2    1 1057703    6 2015
## 3         1  1  2    3  ZZ6    1    1    2    1 1057703    6 2015
## 4         1  1  2    4  ZZ6    1    1    2    1 1057703    6 2015
## 5         1  1  2    5  ZZ6    1    1    2    1 1057703    6 2015
## 6         1  1  2    6  ZZ6    1    1    2    1 1057703    6 2015
```

A few key variables of interest are

- Survey designs:
  - Strata: `v023`
  - Cluster ID: `v001`
  - Household ID: `v002`
  - Survey weight: `v005`
- Date of interview `v008`
- Date of child's birth: `b3`
- Indicator for death of child: `b5`
- Age of death of child in months: `b7`

Notice that dates are represented using century month codes (CMC) format in DHS data, i.e., the number of the month since the beginning of 1900. A small number of DHS did not use the Gregorian calendar (the calendar used in most of the world). For example, the Ethiopian calendar is 92 months behind the Gregorian calendar in general. Then we need to transform dates to the Gregorian calendar, either by hand, or in the `getBirths` function that we will discuss later with the `cmc.adjust` argument (See SUMMERR package document for details).

Most of DHS uses a **stratified cluster sampling design** in which the country is first partitioned into a set of strata (e.g., province by urban/rural). Within each strata (`v023`), clusters (`v001`) are sampled. Within each cluster, households (`v002`) are sampled. Within each household, individuals are selected for interview.

Person-month format

In order to calculate the design-based estimates of the U5MR using discrete survival models, we first split the full birth history data into person-month format. We will use the `getBirths` function in the SUMMER package. We can see from `range(data$v008)` that the CMC code for the date of interview corresponds to the year `1900 + floor(1386/12)= 2015`. In practice, however, the survey year is usually known. The survey year variable allows the mis-recorded data. Dates after the `surveyyear` will be removed. Thus for survey taking place over multiple years, the later year is suggested to be used as `surveyyear`. If set to NA then no checking will be performed.

```
births <- getBirths(data = data, surveyyear = 2015, strata = c("v023"), dob = "b3",
    alive = "b5", age = "b7", date.interview = "v008", variables = c("v001",
        "v002", "v004", "v005", "v021", "v022", "v023", "v024", "v025", "v139"),
    month.cut = c(1, 12, 24, 36, 48, 60), year.cut = seq(1980, 2020, by = 5))
head(births)
```

```
##     dob survey_year died id.new v001 v002 v004    v005 v021 v022
## 6  1349        2015    0      2    1    1    1 1057703    1   26
## 7  1349        2015    0      2    1    1    1 1057703    1   26
## 8  1349        2015    0      2    1    1    1 1057703    1   26
## 9  1349        2015    0      2    1    1    1 1057703    1   26
## 10 1349        2015    0      2    1    1    1 1057703    1   26
## 11 1349        2015    0      2    1    1    1 1057703    1   26
##                v023     v024  v025     v139 agemonth obsStart obsStop
## 6  region 2 - rural region 2 rural region 2        0     1349    1350
## 7  region 2 - rural region 2 rural region 2        1     1350    1351
## 8  region 2 - rural region 2 rural region 2        2     1351    1352
## 9  region 2 - rural region 2 rural region 2        3     1352    1353
## 10 region 2 - rural region 2 rural region 2        4     1353    1354
## 11 region 2 - rural region 2 rural region 2        5     1354    1355
##    obsmonth year  age  time           strata
## 6      1349  112    0 10-14 region 2 - rural
## 7      1350  112 1-11 10-14 region 2 - rural
## 8      1351  112 1-11 10-14 region 2 - rural
## 9      1352  112 1-11 10-14 region 2 - rural
## 10     1353  112 1-11 10-14 region 2 - rural
## 11     1354  112 1-11 10-14 region 2 - rural
```

Now each observation of child is split into up to 60 rows of data, and marked with the corresponding age group in [0,1), [1,12), [12,24), [24, 36), [26, 48), and [48, 60) months and the 5-year periods starting from 80-84.

Direct estimates

We can calculate design-based direct estimates of U5MR using the transformed data with the `getDirect` function in the SUMMER package.

```
years <- levels(births$time)
direct <- getDirect(births = births, years = years, regionVar = "v139", timeVar = "time",
    clusterVar = "~v001 + v002", ageVar = "age", weightsVar = "v005")
```

The `direct` object now contains design-based U5MR estimates at both national and subnational levels. 95% confidence intervals, logit transformed U5MR and the associated variance and precisions are also reported. Notice that since we only have one survey, the survey index is NA. In the next R session we will use the similar function `getDirectList` which process multiple person-month files.

```
head(direct)
```

```
##   region years mean lower upper logit.est var.est region_num survey
## 1    All 80-84 0.29  0.20  0.42     -0.88  0.0748          0     NA
## 2    All 85-89 0.26  0.21  0.31     -1.06  0.0186          0     NA
## 3    All 90-94 0.24  0.21  0.28     -1.15  0.0113          0     NA
## 4    All 95-99 0.22  0.19  0.24     -1.29  0.0070          0     NA
## 5    All 00-04 0.22  0.20  0.25     -1.25  0.0045          0     NA
## 6    All 05-09 0.19  0.18  0.21     -1.44  0.0029          0     NA
##   logit.prec
## 1         13
## 2         54
## 3         89
## 4        144
## 5        224
## 6        350
```

```
tail(direct)
```

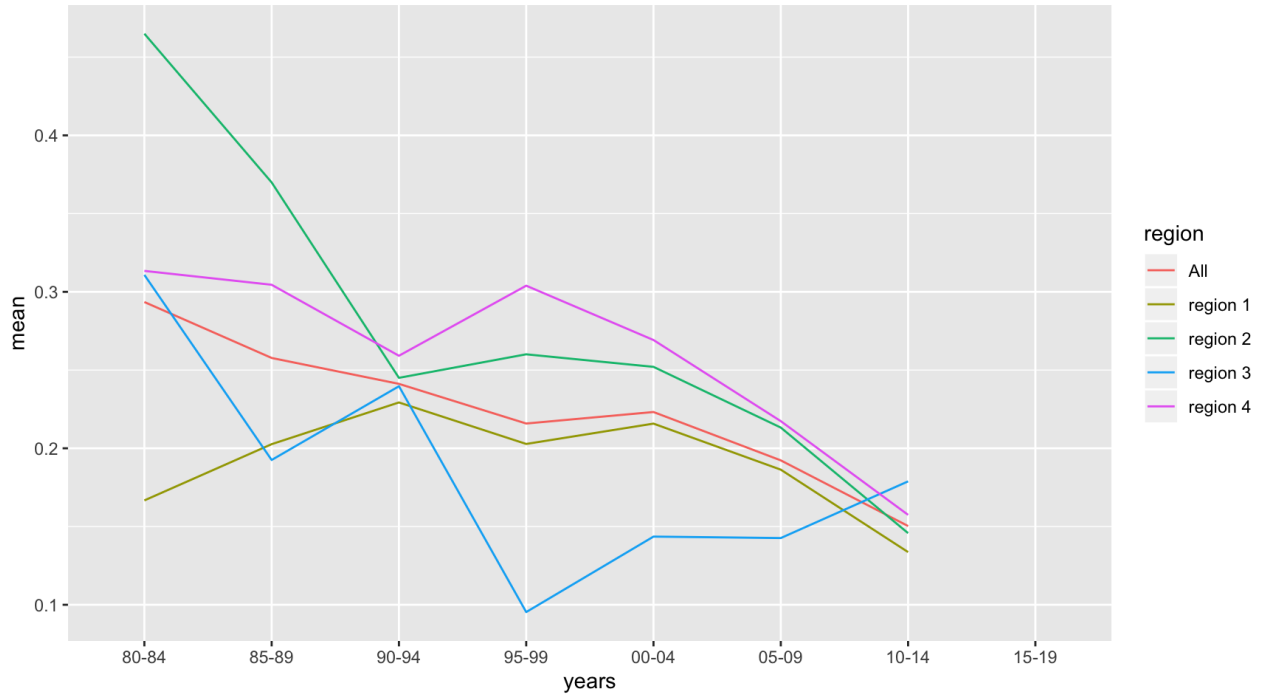
```
##       region years mean lower upper logit.est var.est region_num survey
## 35 region 4 90-94 0.26  0.20  0.32     -1.05  0.0262          4     NA
## 36 region 4 95-99 0.30  0.26  0.36     -0.83  0.0150          4     NA
## 37 region 4 00-04 0.27  0.23  0.32     -1.00  0.0130          4     NA
## 38 region 4 05-09 0.22  0.18  0.27     -1.28  0.0186          4     NA
## 39 region 4 10-14 0.16  0.13  0.18     -1.68  0.0085          4     NA
## 40 region 4 15-19   NA    NA    NA        NA      NA          4     NA
##    logit.prec
## 35         38
## 36         67
## 37         77
## 38         54
## 39        117
## 40         NA
```

Visualization

Line plot

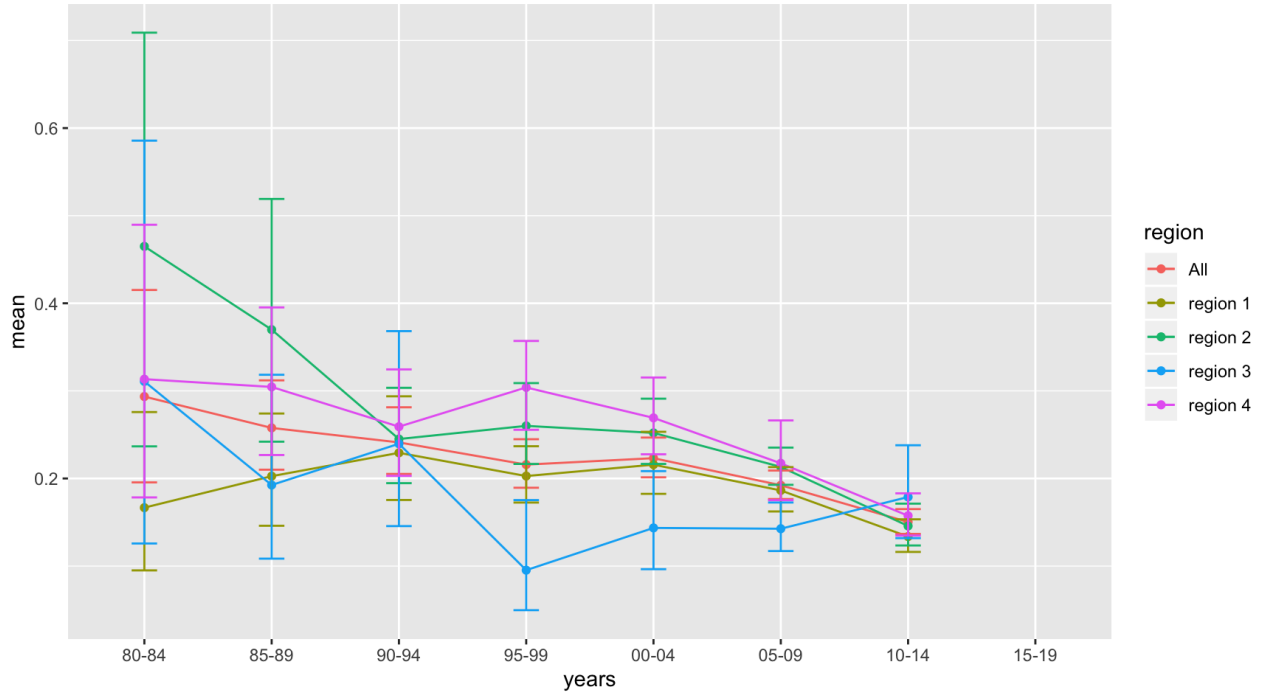
Let us first look at a simple line plot, we will use `ggplot2` for all the visualizations in this section.

```
direct$years <- factor(direct$years, levels = years)
g <- ggplot(data = direct, aes(x = years, y = mean, color = region, group = region)) +
    geom_line()
g
```

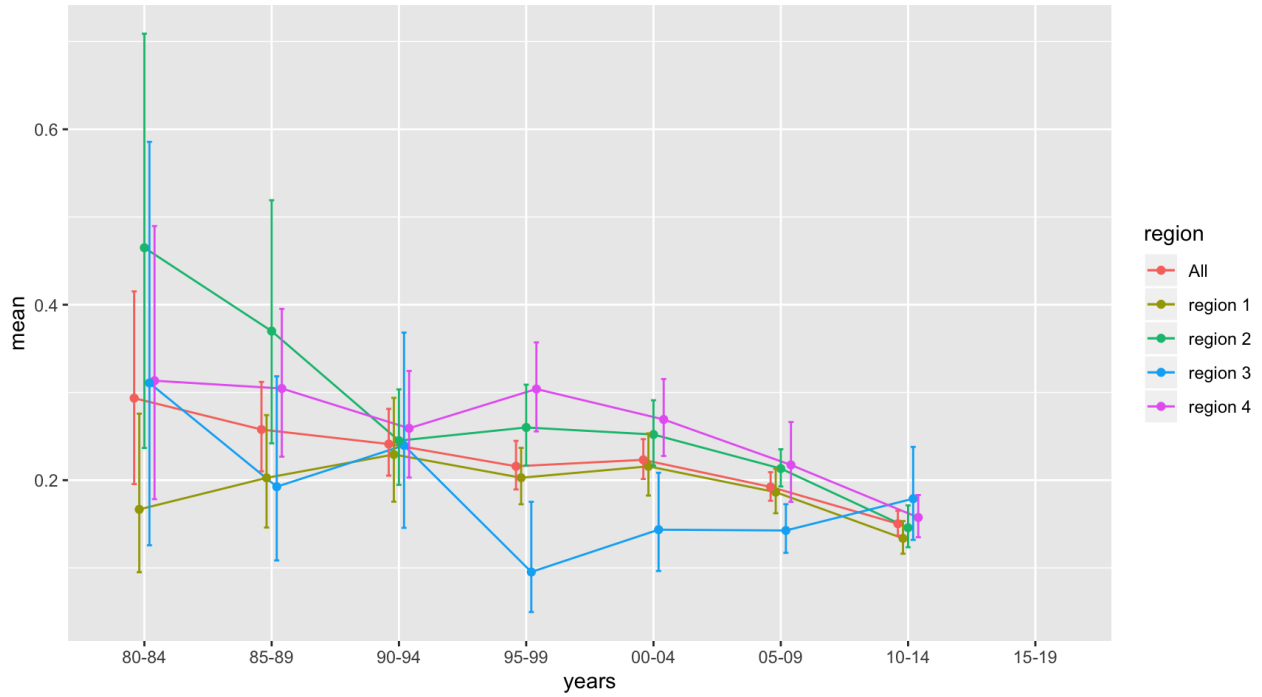


This figure looks good, but not very informative or pretty. Notice we only specified the minimal elements to make a graph: the data, the aesthetic mapping (x, y, color), and the geometric object (line). We can add more elements on top of the object `g` here.

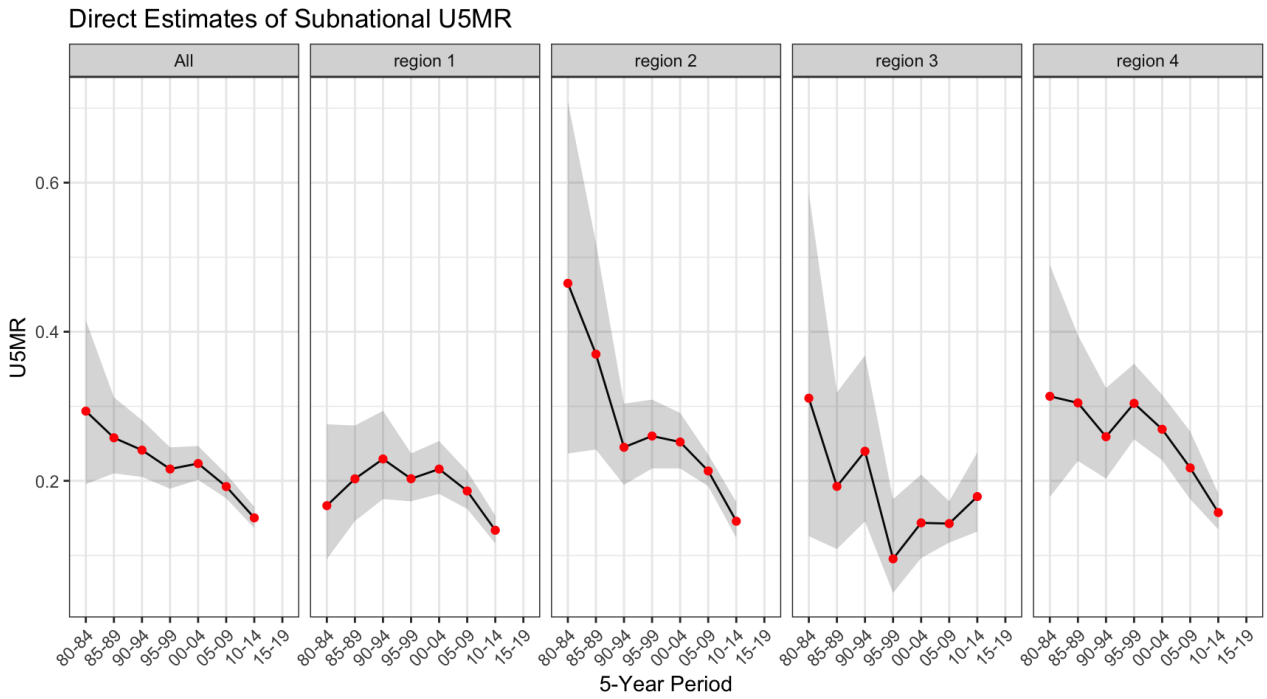
```
g + geom_point() + geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.2)
```




```
pos <- position_dodge(width = 0.2)
g <- ggplot(data = direct, aes(x = years, y = mean, color = region, group = region)) +
    geom_line(position = pos) + geom_point(position = pos) + geom_errorbar(aes(ymin = lower,
    ymax = upper), width = 0.2, position = pos)
g
```




```
g <- ggplot(data = direct, aes(x = years, y = mean, group = region)) + geom_line() +
    geom_ribbon(aes(ymin = lower, ymax = upper), color = NA, alpha = 0.2) +
    facet_wrap(~region, ncol = 5) + geom_point(color = "red") + theme_bw() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) + xlab("5-Year Period") +
    ylab("U5MR") + ggtitle("Direct Estimates of Subnational U5MR")
g
```



This is still a relative simple plot, but there are many other ways to further improve the presentation. The ggplot2 package makes it very flexible to add components to the plots. Interested readers may check out the ggplot2 website for more extensions and tutorials.

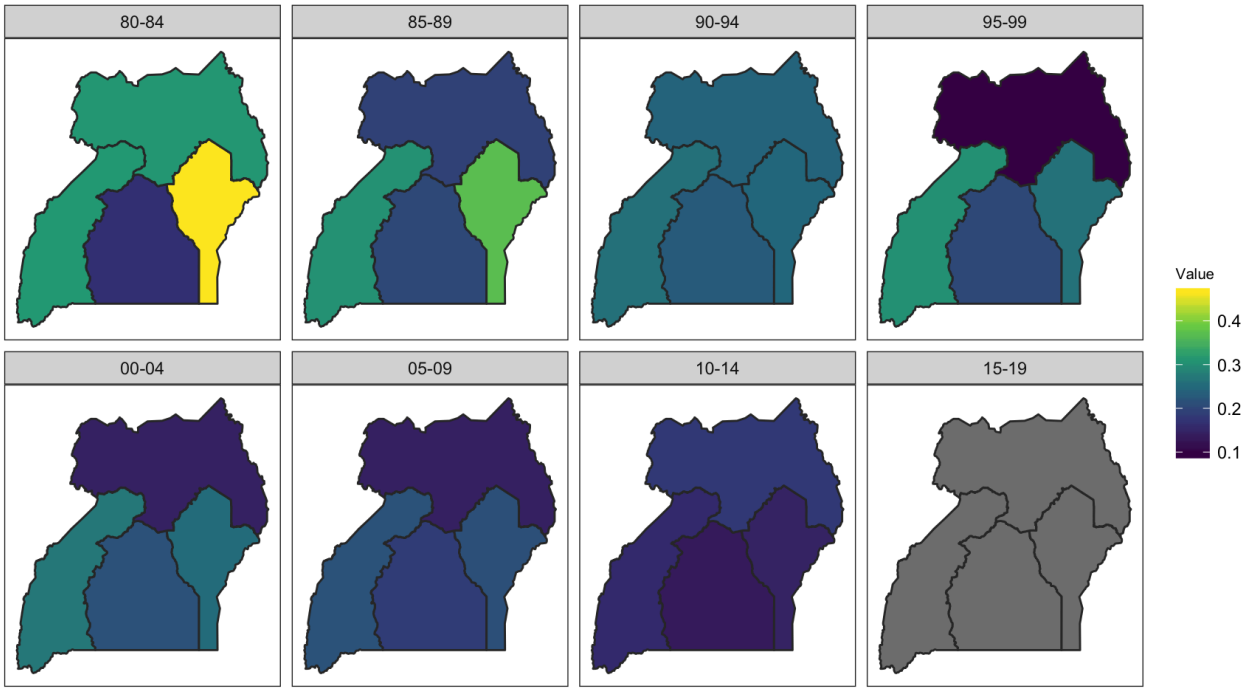
Plot on maps

Now let us visualize the results on a map using functions in the SUMMER package. The DHS model datasets are not associated with any particular real maps, so we will need to make up a map that have the same region names. We use the Uganda map built in the SUMMER package and change the names of the regions to the same as in the model dataset.

```
data(DemoMap)
geo <- DemoMap$geo
geo$REGNAME <- paste("region", 1:4)
```

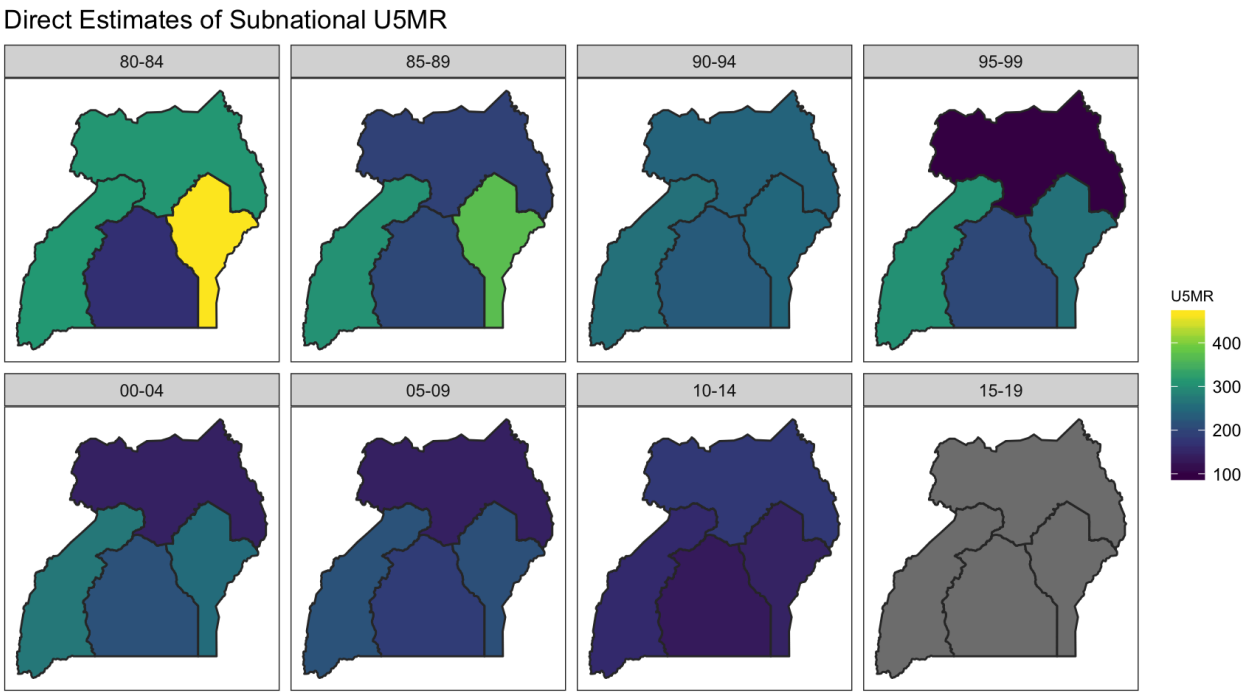
Now we use the `mapPlot` function in SUMMER to combine the data and map.

```
mapPlot(data = direct, geo = geo, by.data = "region", by.geo = "REGNAME", variables = "years",
    values = "mean", is.long = TRUE, ncol = 4)
```



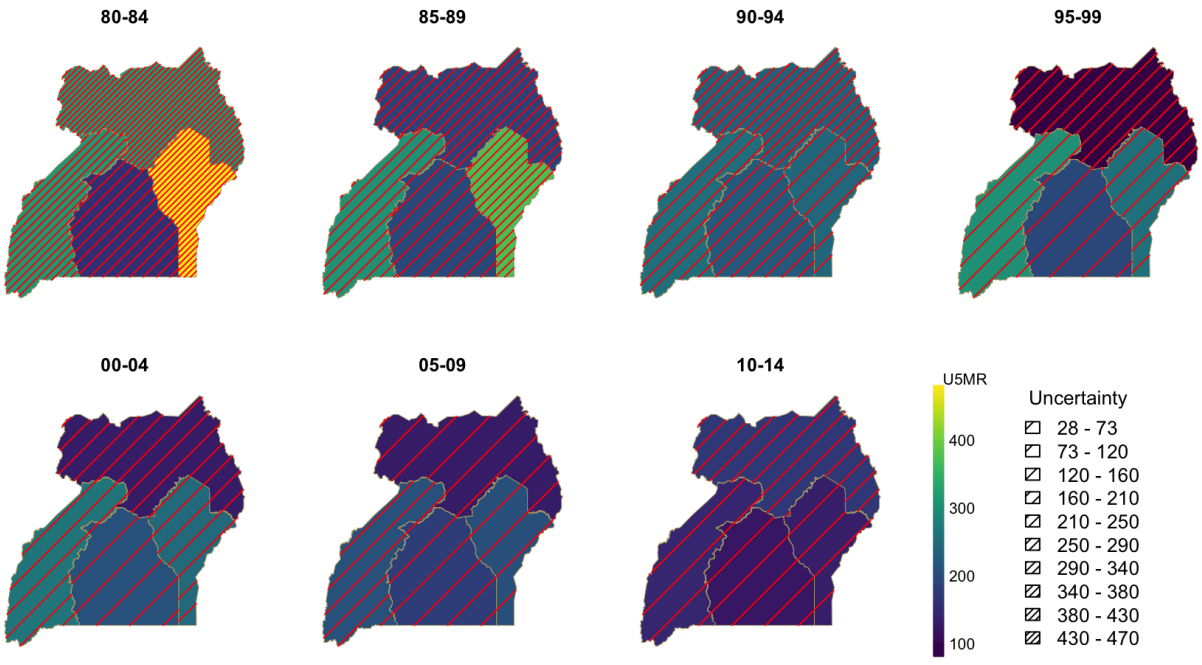
We can also change the scales to the more commonly used representation of number of deaths per 1000 live births.

```
mapPlot(data = direct, geo = geo, by.data = "region", by.geo = "REGNAME", variables = "years",
    values = "mean", is.long = TRUE, ncol = 4, per1000 = TRUE, size = 0.5, legend.label = "U5MR")
+
    ggtitle("Direct Estimates of Subnational U5MR")
```



We can also use hatching to indicate the width of the 95% CI on the map.

```
hatchPlot(data = subset(direct, years != "15-19"), geo = geo, by.data = "region",
    by.geo = "REGNAME", variables = "years", values = "mean", lower = "lower",
    upper = "upper", is.long = TRUE, ncol = 4, per1000 = TRUE, size = 0.5, legend.label = "U5MR",
    hatch = "red")
```



Getting help

Whenever you have question about how to use a particular function provided in some R package, the best source of information is usually the package document. Take `hatchPlot` function for example, you can access the help file using `?hatchPlot` when SUMMER package is loaded.