

SVD Component Mortality Model

Reproducibility Materials: Data and Code

Samuel J. Clark

2015-2019, 2022, 2024

Contents

1 Preliminaries	1
2 HMD Data	2
2.1 Access and parse HMD	2
2.2 Clean HMD	18
2.3 Additional Indicator Calculation	31
3 SVD Component Model of Mortality	33
3.1 <i>svdMod()</i> function	33
3.2 <i>ltPredict()</i> function	39
4 Validation	41
5 Plotting	53
6 Make Tables	96
7 Test on Other Countries	124
8 Make Compressed Models Object for Package	139
9 Wrap up	157

1 Preliminaries

This R Markdown document was created in RStudio using the Knit Directory = *Document Directory* setting. The code assumes it will execute in the document directory, and depending on how you run this, you may have to set that manually, just above.

This R Markdown document was prepared using

R version:

```
R.Version()$version.string
```

```
## [1] "R version 4.4.1 (2024-06-14)"  
• RStudio version 1.2.1335  
• Additional R packages from CRAN  
– knitr
```

- bookdown
- formatR
- devtools
- reshape2
- ggplot2
- plyr
- stargazer
- xtable
- readr
- stringr
- httr
- lubridate

This document is distributed in R Markdown .Rmd and PDF .pdf documents. First load the necessary packages.

2 HMD Data

2.1 Access and parse HMD

First clear everything out, including directories where results will be stored.

```
rm(list=ls()) # clear R environment
system("rm -r ../data/HMD/*") # remove contents of HMD data directory
system("rm -r ../RData/*") # remove contents of Rdata directory
system("rm -r ../figures/*") # remove contents of figures directory
system("rm -r ../tables/*") # remove contents of tables directory
```

The data come from the Human Mortality Database (HMD) and are available online at www.mortality.org – [click here](#). To get the latest HMD, go to the HMD web site, sign up for an account, and download the ‘All HMD statistics zip file’ (maybe at https://www.mortality.org/File/Download/hmd.v6/zip/all_hmd/hmd_statistics_20241010.zip). The code below is used to process the contents of that file to extract the life tables.

```
# code to unzip the HMD Statistics file and organize the life tables
#   into an easy-to-manipulate list
list.raw.lts <- function(hmd.dir,age,period) {

  lts <- list(
    female = read.raw.lt(hmd.dir,"female",age,period),
    male = read.raw.lt(hmd.dir,"male",age,period),
    both = read.raw.lt(hmd.dir,"both",age,period)
  )

  return(lts)
}

read.raw.lt <- function(hmd.dir,sex,age,period) {

  # <hmd.dir> must contain a string that specifies the location
  # of the 'hmd_statistics' directory created when the
  # HMD .zip file is unzipped.

  switch <- sex.per.age.switch(sex,age,period,hmd.dir)
```

```

data.dir <- paste(hmd.dir,switch$data.path,sep="")

lt <- list()

files <- Sys.glob(paste(eval(data.dir),"/*.txt",sep=""))
files.split <- strsplit(files,"\\.")

for (i in 1:length(files.split)) {
  country.name <- strsplit(basename(files[i]),"\\".')[[1]][1]
  lt[[country.name]] = parse.lt(files[i],age)
}

return(lt)

}

sex.per.age.switch <- function(sex,age,period,root.dir) {

  subtract <- 0
  for (i in 1:3) {
    if(str_sub(root.dir,1,1)=="." | str_sub(root.dir,1,1)=="/") {
      subtract <- subtract+1
    }
  }
  root.length <- str_length(root.dir)-subtract

  if (sex == "female") {

    if (age == 1) {
      if (period == 1) {
        path <- "/lt_female/fltpер_1x1"
        value <- list(
          sap.code = 1,
          data.path = path
        )
      } else if (period == 5) {
        path <- "/lt_female/fltpер_1x5"
        value <- list(
          sap.code = 2,
          data.path = path
        )
      } else if (period == 10) {
        path <- "/lt_female/fltpер_1x10"
        value <- list(
          sap.code = 3,
          data.path = path
        )
      }
    } else {
      value <- list(
        sap.code = 0,
        data.path = ""
      )
    }
  }
}

```

```

        }
    } else if (age == 5) {
        if (period == 1) {
            path <- "/lt_female/fltpers_5x1"
            value <- list(
                sap.code = 4,
                data.path = path
            )
        } else if (period == 5) {
            path <- "/lt_female/fltpers_5x5"
            value <- list(
                sap.code = 5,
                data.path = path
            )
        } else if (period == 10) {
            path <- "/lt_female/fltpers_5x10"
            value <- list(
                sap.code = 6,
                data.path = path
            )
        } else {
            value <- list(
                sap.code = 0,
                data.path = ""
            )
        }
    } else {
        value <- list(
            sap.code = 0,
            data.path = ""
        )
    }
}

} else if (sex == "male") {

    if (age == 1) {
        if (period == 1) {
            path <- "/lt_male/mltpers_1x1"
            value <- list(
                sap.code = 7,
                data.path = path
            )
        } else if (period == 5) {
            path <- "/lt_male/mltpers_1x5"
            value <- list(
                sap.code = 8,
                data.path = path
            )
        } else if (period == 10) {
            path <- "/lt_male/mltpers_1x10"
            value <- list(
                sap.code = 9,
                data.path = path
            )
        }
    }
}

```

```

        )
    } else {
        value <- list(
            sap.code = 0,
            data.path = ""
        )
    }
} else if (age == 5) {
    if (period == 1) {
        path <- "/lt_male/mltpers_5x1"
        value <- list(
            sap.code = 10,
            data.path = path
        )
    } else if (period == 5) {
        path <- "/lt_male/mltpers_5x5"
        value <- list(
            sap.code = 11,
            data.path = path
        )
    } else if (period == 10) {
        path <- "/lt_male/mltpers_5x10"
        value <- list(
            sap.code = 12,
            data.path = path
        )
    } else {
        value <- list(
            sap.code = 0,
            data.path = ""
        )
    }
} else {
    value <- list(
        sap.code = 0,
        data.path = ""
    )
}

} else if (sex == "both") {

    if (age == 1) {
        if (period == 1) {
            path <- "/lt_both/bltper_1x1"
            value <- list(
                sap.code = 7,
                data.path = path
            )
        } else if (period == 5) {
            path <- "/lt_both/bltper_1x5"
            value <- list(
                sap.code = 8,
                data.path = path
            )
        }
    }
}

```

```

        )
    } else if (period == 10) {
        path <- "/lt_both/bltper_1x10"
        value <- list(
            sap.code = 9,
            data.path = path
        )
    } else {
        value <- list(
            sap.code = 0,
            data.path = ""
        )
    }
} else if (age == 5) {
    if (period == 1) {
        path <- "/lt_both/bltper_5x1"
        value <- list(
            sap.code = 10,
            data.path = path
        )
    } else if (period == 5) {
        path <- "/lt_both/bltper_5x5"
        value <- list(
            sap.code = 11,
            data.path = path
        )
    } else if (period == 10) {
        path <- "/lt_both/bltper_5x10"
        value <- list(
            sap.code = 12,
            data.path = path
        )
    } else {
        value <- list(
            sap.code = 0,
            data.path = ""
        )
    }
} else {
    value <- list(
        sap.code = 0,
        data.path = ""
    )
}

} else {
    value <- list(
        sap.code = 0,
        data.path = ""
    )
}

return(value)

```

```

}

parse.lt <- function(file.name,age) {

  if (age == 1) {
    w <- c(9, 11, 11, 9, 6, 8, 8, 8, 9, NA)
  } else if (age == 5) {
    w <- c(9, 11, 11, 9, 6, 8, 8, 8, 9, NA)
  } else {
    w <- NA
  }

  return(
    read_fwf(
      file=file.name
      , na = c("", "NA")
      , skip = 3
      , col_types="ccnnnnnnnn"
      , fwf_widths(
        widths = w
        , col_names=c("period","age","mx","qx","ax","lx","dx","Lx","Tx","ex")
      )
    )
  )
}

list.lts <- function(hmd.dir,age,period) {

  list.raw.lts <- list.raw.lts(hmd.dir,age,period)

  if (age == 1) {
    ages <- 111
  } else if (age == 5) {
    ages <- 24
  } else {
    ages <- NA
  }

  lt.list <- list()
  lt.list[["creation.date"]] <- date()
  lt.list[["female"]] <- list()
  lt.list[["male"]] <- list()
  lt.list[["both"]] <- list()
  lt.list[["age"]] <- age
  lt.list[["age.groups"]] <- ages
  lt.list[["period"]] <- period

  for (sx in c("female","male","both")) {
    for (i in 1:length(list.raw.lts[[sx]])) {
      lt.list[[sx]][[eval(names(list.raw.lts[[sx]][i]))]] <- list()
    }
  }
}

```

```

    for (j in 1:(dim(list.raw.lts[[sx]][[i]])[1]/ages)) {
      pop <- eval(names(list.raw.lts[[sx]][i]))
      per <- unlist(list.raw.lts[[sx]][[i]][(ages*j-(ages-1)),1])
      per <- paste("P",str_replace_all(per, "[ - ]", "to"),sep="")
      lt.list[[sx]][[pop]][[per]] = list.raw.lts[[sx]][[i]][((ages*j-(ages-1)): (ages*j)),]
    }
  }
}

return(lt.list)
}

count.lts <- function (lt.list,sex) {

  lt.cumsum <- 0
  for (i in 1:length(lt.list[[sex]])) {
    lt.cumsum <- lt.cumsum + length(lt.list[[sex]][[i]])
  }

  return(lt.cumsum)
}

extract.lt.col <- function (lt.list,sex,col.name) {

  if (lt.list$age == 1) {
    ages <- 111
  } else if (lt.list$age == 5) {
    ages <- 24
  } else {
    ages <- NA
  }

  if (col.name == "period") {
    col <- 1
  } else if (col.name == "age") {
    col <- 2
  } else if (col.name == "mx") {
    col <- 3
  } else if (col.name == "qx") {
    col <- 4
  } else if (col.name == "ax") {
    col <- 5
  } else if (col.name == "lx") {
    col <- 6
  } else if (col.name == "dx") {
    col <- 7
  } else if (col.name == "Lx") {
    col <- 8
  } else if (col.name == "Tx") {

```

```

    col <- 9
} else if (col.name == "ex") {
  col <- 10
} else {
  col <- NA
}

lts.count <- count.lts(lt.list,"female")

if (col > 1) {
  lts.colmat <- matrix(data=rep(0,ages*lts.count),nrow=ages,ncol=lts.count)
} else if (col == 1) {
  lts.colmat <- matrix(data=rep("",ages*lts.count),nrow=ages,ncol=lts.count)
}

col.names <- rep("",lts.count)
col.index <- 1

for (i in 1:length(lt.list[[sex]])) {
  for (j in 1:length(lt.list[[sex]][[i]])) {
    if (col > 1) {
      lts.colmat[,col.index] <- as.numeric(unlist(lt.list[[sex]][[i]][[j]][,col]))
    } else if (col == 1) {
      lts.colmat[,col.index] <- as.character(unlist(lt.list[[sex]][[i]][[j]][,col]))
    }
    pop <- names(lt.list[[sex]])[i]
    per <- str_sub(names(lt.list[[sex]][[i]])[j],2,str_length(names(lt.list[[sex]][[i]])[j]))
    col.names[col.index] <- paste(eval(sex),".",pop,".",per,sep="")
    col.index <- col.index + 1
  }
}

colnames(lts.colmat) <- col.names
rownames(lts.colmat) <- unlist(lt.list$female[[1]][[1]][,2])

return(lts.colmat)
}

```

The following chunk unzips various versions of the HMD data starting with the version of HMD from 2018 that was used in the *Demography* paper. You need to comment appropriately to get the version you want. The data are unzipped into the *data/HMD/hmd_statistics* directory.

```

# unzip(zipfile="../data/HMD Archive/hmd_statistics_20181102.zip",exdir="../data/HMD/hmd_statistics")
# data.date <- "November 2, 2018"
# unzip(zipfile="../data/HMD Archive/hmd_statistics_20220915.zip",exdir="../data/HMD/hmd_statistics")
# data.date <- "October 18, 2024"
unzip(zipfile="../data/HMD Archive/hmd_statistics_20241010.zip",exdir="../data/HMD/hmd_statistics")
data.date <- "October 10, 2024"

```

HMD nomenclature describes *age*×*period* life tables. For example 1×1 are single calendar year by single year of age, and 5×5 are five-year age groups by five-year periods, with the first age group broken into 0 and 1–4 years. The following code creates an R list for each of various commonly used life tables. The resulting lists are saved in the *RData* directory in compressed form. Unless it has been fixed between when I write this and when you execute it, the following code will produce errors for some of the Belarus files – those files do

not contain data. Finally, if you use the HMD download functions on their own, make sure not to prepend ‘/’ or ‘./’ to the path for the *hmd_statistics* directory. Several errors will appear; these are due to several HMD life tables not having any values, at this time all from Belarus.

```
# 1-year age x 1-year period life tables
hmd.1x1.list <- list.lts("../data/HMD/hmd_statistics", 1, 1)
# arguments are
# path to 'hmd_statistics' directory
# age designator: either 1 or 5 year age groups
# period designator: either 1, 5, or 10 year age groups
save(file="../RData/hmd-1x1.RData", compress=TRUE, list=c("hmd.1x1.list"))

# hmd.1x5.list <- list.lts("data/HMD/hmd_statistics", 1, 5, download.result$headers$date)
# save(file="../RData/hmd-1x5.RData", compress=TRUE, list=c("hmd.1x5.list"))
# hmd.1x10.list <- list.lts("data/HMD/hmd_statistics", 1, 10, download.result$headers$date)
# save(file="../RData/hmd-1x10.RData", compress=TRUE, list=c("hmd.1x10.list"))

# 5-year age x 1-year life tables
hmd.5x1.list <- list.lts("../data/HMD/hmd_statistics", 5, 1)
save(file="../RData/hmd-5x1.RData", compress=TRUE, list=c("hmd.5x1.list"))

# hmd.5x5.list <- list.lts("data/HMD/hmd_statistics", 5, 5, download.result$headers$date)
# save(file="../RData/hmd-5x5.RData", compress=TRUE, list=c("hmd.5x5.list"))
# hmd.5x10.list <- list.lts("data/HMD/hmd_statistics", 5, 10, download.result$headers$date)
# save(file="../RData/hmd-5x10.RData", compress=TRUE, list=c("hmd.5x10.list"))

# rm(list=c("download.result"))
```

Have quick look at the lists saved in *Rdata*.

```
list.files("../RData/")
```

```
## [1] "hmd-1x1.RData" "hmd-5x1.RData"
```

Have a look at the top-level structure of the list.

```
str(hmd.5x1.list, max.level = 1)
```

```
## List of 7
## $ creation.date: chr "Tue Oct 22 14:30:49 2024"
## $ female      :List of 50
## $ male        :List of 50
## $ both        :List of 50
## $ age         : num 5
## $ age.groups  : num 24
## $ period      : num 1
```

Have a quick look at the full structure of the lists, vastly truncated!

```
str(hmd.5x1.list, list.len = 4, vec.len = 2)
```

```
## List of 7
## $ creation.date: chr "Tue Oct 22 14:30:49 2024"
## $ female      :List of 50
##   ..$ AUS    :List of 101
##     ...$ P1921: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
##       ... .$. period: chr [1:24] "1921" "1921" ...
##       ... .$. age   : chr [1:24] "0" "1-4" ...
##       ... .$. mx    : num [1:24] 0.05999 0.00602 ...
##       ... .$. qx    : num [1:24] 0.0575 0.0237 0.00952 0.00637 0.0102 ...
```

```

## ... . . . [list output truncated]
## ... $ P1922: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1922" "1922" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.04594 0.00449 ...
## ... . . . $ qx : num [1:24] 0.0444 0.0178 ...
## ... . . . [list output truncated]
## ... $ P1923: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1923" "1923" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.05565 0.00478 ...
## ... . . . $ qx : num [1:24] 0.0535 0.0189 ...
## ... . . . [list output truncated]
## ... $ P1924: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1924" "1924" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.05379 0.00485 ...
## ... . . . $ qx : num [1:24] 0.0517 0.0191 ...
## ... . . . [list output truncated]
## ... . . . [list output truncated]
## ... $ AUT :List of 73
## ... . . . $ P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1947" "1947" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.07981 0.00414 ...
## ... . . . $ qx : num [1:24] 0.0757 0.0164 ...
## ... . . . [list output truncated]
## ... $ P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1948" "1948" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.07327 0.00316 ...
## ... . . . $ qx : num [1:24] 0.0698 0.0125 ...
## ... . . . [list output truncated]
## ... $ P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1949" "1949" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.06717 0.00347 ...
## ... . . . $ qx : num [1:24] 0.0642 0.0138 ...
## ... . . . [list output truncated]
## ... $ P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1950" "1950" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.05953 0.00274 ...
## ... . . . $ qx : num [1:24] 0.0571 0.0109 ...
## ... . . . [list output truncated]
## ... . . . [list output truncated]
## ... $ BEL :List of 182
## ... . . . $ P1841: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $ period: chr [1:24] "1841" "1841" ...
## ... . . . $ age : chr [1:24] "0" "1-4" ...
## ... . . . $ mx : num [1:24] 0.1516 0.0411 ...
## ... . . . $ qx : num [1:24] 0.137 0.148 ...
## ... . . . [list output truncated]
## ... $ P1842: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)

```

```

## ... .$. period: chr [1:24] "1842" "1842" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1601 0.0463 ...
## ... .$. qx : num [1:24] 0.144 0.165 ...
## ... [list output truncated]
## ... $. P1843: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1843" "1843" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1482 0.0413 ...
## ... .$. qx : num [1:24] 0.135 0.149 ...
## ... [list output truncated]
## ... $. P1844: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1844" "1844" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1373 0.0353 ...
## ... .$. qx : num [1:24] 0.125 0.129 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. BGR :List of 75
## ... $. P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1947" "1947" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1356 0.0143 ...
## ... .$. qx : num [1:24] 0.1241 0.0551 ...
## ... [list output truncated]
## ... $. P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1948" "1948" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1224 0.0141 ...
## ... .$. qx : num [1:24] 0.1129 0.0542 ...
## ... [list output truncated]
## ... $. P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1949" "1949" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.11473 0.00887 ...
## ... .$. qx : num [1:24] 0.1064 0.0347 ...
## ... [list output truncated]
## ... $. P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1950" "1950" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.0931 0.0072 ...
## ... .$. qx : num [1:24] 0.0875 0.0282 ...
## ... [list output truncated]
## ... [list output truncated]
## ... [list output truncated]
## ... $. male :List of 50
## ... $. AUS :List of 101
## ... $. P1921: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1921" "1921" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.07653 0.00699 ...
## ... .$. qx : num [1:24] 0.0725 0.0275 ...
## ... [list output truncated]
## ... $. P1922: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)

```

```

## ... .$. period: chr [1:24] "1922" "1922" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06353 0.00527 ...
## ... .$. qx : num [1:24] 0.0606 0.0208 ...
## ... [list output truncated]
## ... $. P1923: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1923" "1923" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06939 0.00587 ...
## ... .$. qx : num [1:24] 0.066 0.0231 ...
## ... [list output truncated]
## ... $. P1924: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1924" "1924" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06487 0.00561 ...
## ... .$. qx : num [1:24] 0.0618 0.0221 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. AUT :List of 73
## ... $. P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1947" "1947" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.0994 0.00482 0.00153 0.00131 0.00228 ...
## ... .$. qx : num [1:24] 0.0929 0.0191 ...
## ... [list output truncated]
## ... $. P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1948" "1948" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.09421 0.00377 ...
## ... .$. qx : num [1:24] 0.0884 0.0149 ...
## ... [list output truncated]
## ... $. P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1949" "1949" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.08592 0.00371 ...
## ... .$. qx : num [1:24] 0.081 0.0147 ...
## ... [list output truncated]
## ... $. P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1950" "1950" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.07735 0.00302 ...
## ... .$. qx : num [1:24] 0.0733 0.012 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. BEL :List of 182
## ... $. P1841: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1841" "1841" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1865 0.0395 ...
## ... .$. qx : num [1:24] 0.165 0.143 ...
## ... [list output truncated]
## ... $. P1842: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1842" "1842" ...
## ... .$. age : chr [1:24] "0" "1-4" ...

```

```

## ... .$. mx : num [1:24] 0.1918 0.0439 ...
## ... .$. qx : num [1:24] 0.169 0.157 ...
## ... [list output truncated]
## ... $. P1843: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1843" "1843" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1815 0.0408 ...
## ... .$. qx : num [1:24] 0.161 0.147 ...
## ... [list output truncated]
## ... $. P1844: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1844" "1844" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1714 0.0348 ...
## ... .$. qx : num [1:24] 0.153 0.127 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. BGR :List of 75
## ... $. P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1947" "1947" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.156 0.014 ...
## ... .$. qx : num [1:24] 0.1408 0.0541 ...
## ... [list output truncated]
## ... $. P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1948" "1948" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1397 0.0131 ...
## ... .$. qx : num [1:24] 0.1273 0.0505 ...
## ... [list output truncated]
## ... $. P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1949" "1949" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.136 0.01 ...
## ... .$. qx : num [1:24] 0.1245 0.0389 ...
## ... [list output truncated]
## ... $. P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1950" "1950" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.11234 0.00779 ...
## ... .$. qx : num [1:24] 0.1041 0.0305 ...
## ... [list output truncated]
## ... [list output truncated]
## ... [list output truncated]
## ... $. both :List of 50
## ... $. AUS :List of 101
## ... $. P1921: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1921" "1921" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06844 0.00651 ...
## ... .$. qx : num [1:24] 0.0652 0.0256 ...
## ... [list output truncated]
## ... $. P1922: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1922" "1922" ...
## ... .$. age : chr [1:24] "0" "1-4" ...

```

```

## ... .$. mx : num [1:24] 0.05492 0.00489 ...
## ... .$. qx : num [1:24] 0.0527 0.0193 ...
## ... [list output truncated]
## ... $. P1923: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1923" "1923" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06265 0.00533 ...
## ... .$. qx : num [1:24] 0.0599 0.021 ...
## ... [list output truncated]
## ... $. P1924: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1924" "1924" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.05943 0.00523 ...
## ... .$. qx : num [1:24] 0.0569 0.0207 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. AUT :List of 73
## ... $. P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1947" "1947" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.08989 0.00449 ...
## ... .$. qx : num [1:24] 0.0846 0.0178 ...
## ... [list output truncated]
## ... $. P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1948" "1948" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.08402 0.00347 ...
## ... .$. qx : num [1:24] 0.0794 0.0138 ...
## ... [list output truncated]
## ... $. P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1949" "1949" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.07677 0.00359 ...
## ... .$. qx : num [1:24] 0.0729 0.0143 ...
## ... [list output truncated]
## ... $. P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1950" "1950" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.06864 0.00288 ...
## ... .$. qx : num [1:24] 0.0654 0.0114 ...
## ... [list output truncated]
## ... [list output truncated]
## ... $. BEL :List of 182
## ... $. P1841: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1841" "1841" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1693 0.0403 ...
## ... .$. qx : num [1:24] 0.152 0.146 ...
## ... [list output truncated]
## ... $. P1842: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... .$. period: chr [1:24] "1842" "1842" ...
## ... .$. age : chr [1:24] "0" "1-4" ...
## ... .$. mx : num [1:24] 0.1762 0.0451 ...
## ... .$. qx : num [1:24] 0.157 0.161 ...

```

```

## ... . . . [list output truncated]
## ... $. P1843: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1843" "1843" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.1652 0.0411 ...
## ... . . . $. qx : num [1:24] 0.148 0.148 ...
## ... . . . [list output truncated]
## ... $. P1844: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1844" "1844" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.155 0.035 ...
## ... . . . $. qx : num [1:24] 0.14 0.128 ...
## ... . . . [list output truncated]
## ... . . . [list output truncated]
## ... $. BGR :List of 75
## ... $. P1947: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1947" "1947" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.1462 0.0142 ...
## ... . . . $. qx : num [1:24] 0.1327 0.0546 ...
## ... . . . [list output truncated]
## ... $. P1948: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1948" "1948" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.1313 0.0136 ...
## ... . . . $. qx : num [1:24] 0.1203 0.0523 ...
## ... . . . [list output truncated]
## ... $. P1949: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1949" "1949" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.12588 0.00945 ...
## ... . . . $. qx : num [1:24] 0.1158 0.0368 ...
## ... . . . [list output truncated]
## ... $. P1950: tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## ... . . . $. period: chr [1:24] "1950" "1950" ...
## ... . . . $. age : chr [1:24] "0" "1-4" ...
## ... . . . $. mx : num [1:24] 0.103 0.0075 0.0016 0.0013 0.00245 ...
## ... . . . $. qx : num [1:24] 0.0961 0.0294 ...
## ... . . . [list output truncated]
## ... . . . [list output truncated]
## ... .. [list output truncated]
## ... [list output truncated]

```

Have look at the full structure of one life table.

```

str(hmd.5x1.list[[3]][[1]][[1]])

## # tibble [24 x 10] (S3: tbl_df/tbl/data.frame)
## $ period: chr [1:24] "1921" "1921" "1921" "1921" ...
## $ age : chr [1:24] "0" "1-4" "5-9" "10-14" ...
## $ mx : num [1:24] 0.07653 0.00699 0.002 0.00172 0.0022 ...
## $ qx : num [1:24] 0.07253 0.02745 0.00993 0.00857 0.01093 ...
## $ ax : num [1:24] 0.28 1.36 2.25 2.5 2.72 2.64 2.58 2.54 2.59 2.55 ...
## $ lx : num [1:24] 100000 92747 90201 89305 88540 ...
## $ dx : num [1:24] 7253 2546 896 765 967 ...

```

```

## $ Lx      : num [1:24] 94762 364272 448539 444614 440492 ...
## $ Tx      : num [1:24] 5910278 5815516 5451243 5002704 4558090 ...
## $ ex      : num [1:24] 59.1 62.7 60.4 56 51.5 ...
# or equivalently
str(hmd.5x1.list$female$AUS$P1921)

## tibble [24 x 10] (S3:tbl_df/tbl/data.frame)
## $ period: chr [1:24] "1921" "1921" "1921" "1921" ...
## $ age    : chr [1:24] "0" "1-4" "5-9" "10-14" ...
## $ mx     : num [1:24] 0.05999 0.00602 0.00192 0.00128 0.00205 ...
## $ qx     : num [1:24] 0.0575 0.0237 0.00952 0.00637 0.0102 ...
## $ ax     : num [1:24] 0.28 1.38 2.14 2.6 2.68 2.57 2.57 2.64 2.62 2.55 ...
## $ lx     : num [1:24] 100000 94250 92016 91140 90559 ...
## $ dx     : num [1:24] 5750 2234 876 580 924 ...
## $ Lx     : num [1:24] 95857 371152 457576 454303 450651 ...
## $ Tx     : num [1:24] 6317561 6221704 5850553 5392977 4938674 ...
## $ ex     : num [1:24] 63.2 66 63.6 59.2 54.5 ...

```

Extract single calendar year 1 and 5-year age group probabilities of dying and life expectancies and save them in *age* \times *lifetable* matrices in the *RData* directory.

```

# 1x1 nqx
q1.f <- extract.lt.col(hmd.1x1.list,"female","qx")
save(file="../RData/q1.f.RData",compress=TRUE,list=c("q1.f"))
q1.m <- extract.lt.col(hmd.1x1.list,"male","qx")
save(file="../RData/q1.m.RData",compress=TRUE,list=c("q1.m"))
# remove last row where nqx = 1
q1.f <- q1.f[1:(nrow(q1.f)-1),]
q1.m <- q1.m[1:(nrow(q1.m)-1),]

# 1x1 lax
a1.f <- extract.lt.col(hmd.1x1.list,"female","ax")
save(file="../RData/a1.f.RData",compress=TRUE,list=c("a1.f"))
a1.m <- extract.lt.col(hmd.1x1.list,"male","ax")
save(file="../RData/a1.m.RData",compress=TRUE,list=c("a1.m"))

# 1x1 lx
l1.f <- extract.lt.col(hmd.1x1.list,"female","lx")
save(file="../RData/l1.f.RData",compress=TRUE,list=c("l1.f"))
l1.m <- extract.lt.col(hmd.1x1.list,"male","lx")
save(file="../RData/l1.m.RData",compress=TRUE,list=c("l1.m"))

# 1x1 ex
e1.f <- extract.lt.col(hmd.1x1.list,"female","ex")
save(file="../RData/e1.f.RData",compress=TRUE,list=c("e1.f"))
e1.m <- extract.lt.col(hmd.1x1.list,"male","ex")
save(file="../RData/e1.m.RData",compress=TRUE,list=c("e1.m"))

# 5x1 nqx
q5.f <- extract.lt.col(hmd.5x1.list,"female","qx")
save(file="../RData/q5.f.RData",compress=TRUE,list=c("q5.f"))
q5.m <- extract.lt.col(hmd.5x1.list,"male","qx")
save(file="../RData/q5.m.RData",compress=TRUE,list=c("q5.m"))
# remove last row where nqx = 1
q5.f <- q5.f[1:(nrow(q5.f)-1),]

```

```

q5.m <- q5.m[1:(nrow(q5.m)-1),]

# 5x1 5ax
a5.f <- extract.lt.col(hmd.5x1.list,"female","ax")
save(file="../RData/a5.f.RData",compress=TRUE,list=c("a5.f"))
a5.m <- extract.lt.col(hmd.5x1.list,"male","ax")
save(file="../RData/a5.m.RData",compress=TRUE,list=c("a5.m"))

# 5x1 lx
15.f <- extract.lt.col(hmd.5x1.list,"female","lx")
save(file="../RData/15.f.RData",compress=TRUE,list=c("15.f"))
15.m <- extract.lt.col(hmd.5x1.list,"male","lx")
save(file="../RData/15.m.RData",compress=TRUE,list=c("15.m"))

# 5x1 ex
e5.f <- extract.lt.col(hmd.5x1.list,"female","ex")
save(file="../RData/e5.f.RData",compress=TRUE,list=c("e5.f"))
e5.m <- extract.lt.col(hmd.5x1.list,"male","ex")
save(file="../RData/e5.m.RData",compress=TRUE,list=c("e5.m"))

# rm(list=c("hmd.1x1.list","hmd.5x1.list"))

```

Have another quick look at the lists and now matrices saved in “Rdata”.

```

list.files("../RData/")

## [1] "a1.f.RData"      "a1.m.RData"      "a5.f.RData"
## [4] "a5.m.RData"      "e1.f.RData"      "e1.m.RData"
## [7] "e5.f.RData"      "e5.m.RData"      "hmd-1x1.RData"
## [10] "hmd-5x1.RData"   "l1.f.RData"      "l1.m.RData"
## [13] "15.f.RData"      "15.m.RData"      "q1.f.RData"
## [16] "q1.m.RData"      "q5.f.RData"      "q5.m.RData"

```

2.2 Clean HMD

There are two persistent problems with the HMD 1×1 life tables: 1. as mentioned just above, some of the Belarus life tables are empty, and 2. the $1q_x$ values for some life tables are ‘flat’ at older ages, i.e. are constant.

In both cases, these life tables need to be removed. We’ll get rid of the Belarus tables first. The strategy is general: identify life tables with ‘NA’ values and remove those. They turn out to be Belarus 1914–1918.

```

## females
which(is.na(q1.f[1,]))

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##           248           249           250
## female.BEL.1917 female.BEL.1918
##           251           252
q1.f[1,which(is.na(q1.f[1,]))]

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##           NA           NA           NA
## female.BEL.1917 female.BEL.1918
##           NA           NA

```

```

which(is.na(q5.f[1,]))

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          248          249          250
## female.BEL.1917 female.BEL.1918
##          251          252

q5.f[1,which(is.na(q5.f[1,]))]

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          NA          NA          NA
## female.BEL.1917 female.BEL.1918
##          NA          NA

which(is.na(e1.f[1,]))

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          248          249          250
## female.BEL.1917 female.BEL.1918
##          251          252

e1.f[1,which(is.na(e1.f[1,]))]

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          NA          NA          NA
## female.BEL.1917 female.BEL.1918
##          NA          NA

which(is.na(e5.f[1,]))

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          248          249          250
## female.BEL.1917 female.BEL.1918
##          251          252

e5.f[1,which(is.na(e5.f[1,]))]

## female.BEL.1914 female.BEL.1915 female.BEL.1916
##          NA          NA          NA
## female.BEL.1917 female.BEL.1918
##          NA          NA

## males
which(is.na(q1.m[1,]))

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          248          249          250
## male.BEL.1917 male.BEL.1918
##          251          252

q1.m[1,which(is.na(q1.m[1,]))]

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          NA          NA          NA
## male.BEL.1917 male.BEL.1918
##          NA          NA

which(is.na(q5.m[1,]))

## male.BEL.1914 male.BEL.1915 male.BEL.1916

```

```

##          248          249          250
## male.BEL.1917 male.BEL.1918
##          251          252
q5.m[1,which(is.na(q5.m[1,]))]

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          NA          NA          NA
## male.BEL.1917 male.BEL.1918
##          NA          NA
which(is.na(e1.m[1,]))

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          248          249          250
## male.BEL.1917 male.BEL.1918
##          251          252
e1.m[1,which(is.na(e1.m[1,]))]

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          NA          NA          NA
## male.BEL.1917 male.BEL.1918
##          NA          NA
which(is.na(e5.m[1,]))

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          248          249          250
## male.BEL.1917 male.BEL.1918
##          251          252
e5.m[1,which(is.na(e5.m[1,]))]

## male.BEL.1914 male.BEL.1915 male.BEL.1916
##          NA          NA          NA
## male.BEL.1917 male.BEL.1918
##          NA          NA
# in all matrices, empty columns are THE SAME, numbered 236-340
remove <- unique(c(which(is.na(q1.f[1,])),which(is.na(q1.m[1,]))))

q1.f <- q1.f[,-remove]
q5.f <- q5.f[,-remove]
e1.f <- e1.f[,-remove]
e5.f <- e5.f[,-remove]
a1.f <- a1.f[,-remove]
a5.f <- a5.f[,-remove]
l1.f <- l1.f[,-remove]
l5.f <- l5.f[,-remove]

q1.m <- q1.m[,-remove]
q5.m <- q5.m[,-remove]
e1.m <- e1.m[,-remove]
e5.m <- e5.m[,-remove]
a1.m <- a1.m[,-remove]
a5.m <- a5.m[,-remove]
l1.m <- l1.m[,-remove]

```

```

15.m <- 15.m[,-remove]

# verify all is well now
q1.f[1,which(is.na(q1.f[1,]))]

## numeric(0)
q5.f[1,which(is.na(q5.f[1,]))]

## numeric(0)
e1.f[1,which(is.na(e1.f[1,]))]

## numeric(0)
e5.f[1,which(is.na(e5.f[1,]))]

## numeric(0)
q1.m[1,which(is.na(q1.m[1,]))]

## numeric(0)
q5.m[1,which(is.na(q5.m[1,]))]

## numeric(0)
e1.m[1,which(is.na(e1.m[1,]))]

## numeric(0)
e5.m[1,which(is.na(e5.m[1,]))]

## numeric(0)
# make sure all matrices have same number of columns and same
# column names
dim(q1.f)

## [1] 110 4906
dim(q1.m)

## [1] 110 4906
dim(q5.f)

## [1] 23 4906
dim(q5.m)

## [1] 23 4906
dim(e1.f)

## [1] 111 4906
dim(e1.m)

## [1] 111 4906
dim(e5.f)

## [1] 24 4906

```

```

dim(e5.m)
## [1] 24 4906
dim(a1.f)

## [1] 111 4906
dim(a1.m)

## [1] 111 4906
dim(a5.f)

## [1] 24 4906
dim(a5.m)

## [1] 24 4906
dim(l1.f)

## [1] 111 4906
dim(l1.m)

## [1] 111 4906
dim(l5.f)

## [1] 24 4906
dim(l5.m)

## [1] 24 4906
# NB, last argument in str_sub intentionally empty, defaults to end of string
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q1.m),6,))

## [1] TRUE
identical(str_sub(colnames(q5.f),8,),str_sub(colnames(q5.m),6,))

## [1] TRUE
identical(str_sub(colnames(e1.f),8,),str_sub(colnames(e1.m),6,))

## [1] TRUE
identical(str_sub(colnames(e5.f),8,),str_sub(colnames(e5.m),6,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.f),8,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.m),6,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.f),8,))

## [1] TRUE

```

```

identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.m),6,))

## [1] TRUE

identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l1.f),8,))

## [1] TRUE

identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l1.m),6,))

## [1] TRUE

identical(str_sub(colnames(q1.f),8,),str_sub(colnames(15.f),8,))

## [1] TRUE

rm(list=c("remove"))

```

Now identify and remove the ‘flat’ life tables. These turn out to be Iceland 1852 and New Zealand Maori 1949, 1956, and 1959.

```

## females -- FOUR PROBLEM LIFE TABLES
# 1-year age
# identify flat female LTs
which(q1.f[106,]==q1.f[110,])

##      female.ISL.1852 female.NZL_MA.1949
##      2901           3887
## female.NZL_MA.1956 female.NZL_MA.1959
##      3894           3897

# verify that they are constant at roughly ages 80+
q1.f[75:110,which(q1.f[106,]==q1.f[110,])]

##      female.ISL.1852 female.NZL_MA.1949
## 74      0.06242       0.07424
## 75      0.07010       0.11634
## 76      0.08167       0.15404
## 77      0.09211       0.04880
## 78      0.09056       0.00000
## 79      0.09189       0.33434
## 80      0.10651       0.11516
## 81      0.10651       0.11516
## 82      0.10651       0.11516
## 83      0.10651       0.11516
## 84      0.10651       0.11516
## 85      0.10651       0.11516
## 86      0.10651       0.11516
## 87      0.10651       0.11516
## 88      0.10651       0.11516
## 89      0.10651       0.11516
## 90      0.10651       0.11516
## 91      0.10651       0.11516
## 92      0.10651       0.11516
## 93      0.10651       0.11516

```

```

## 94      0.10651    0.11516
## 95      0.10651    0.11516
## 96      0.10651    0.11516
## 97      0.10651    0.11516
## 98      0.10651    0.11516
## 99      0.10651    0.11516
## 100     0.10651    0.11516
## 101     0.10651    0.11516
## 102     0.10651    0.11516
## 103     0.10651    0.11516
## 104     0.10651    0.11516
## 105     0.10651    0.11516
## 106     0.10651    0.11516
## 107     0.10651    0.11516
## 108     0.10651    0.11516
## 109     0.10651    0.11516
## female.NZL_MA.1956 female.NZL_MA.1959
## 74      0.08003    0.10611
## 75      0.10351    0.23040
## 76      0.09529    0.05827
## 77      0.08515    0.07232
## 78      0.10131    0.20249
## 79      0.14296    0.08829
## 80      0.11708    0.11642
## 81      0.11708    0.11642
## 82      0.11708    0.11642
## 83      0.11708    0.11642
## 84      0.11708    0.11642
## 85      0.11708    0.11642
## 86      0.11708    0.11642
## 87      0.11708    0.11642
## 88      0.11708    0.11642
## 89      0.11708    0.11642
## 90      0.11708    0.11642
## 91      0.11708    0.11642
## 92      0.11708    0.11642
## 93      0.11708    0.11642
## 94      0.11708    0.11642
## 95      0.11708    0.11642
## 96      0.11708    0.11642
## 97      0.11708    0.11642
## 98      0.11708    0.11642
## 99      0.11708    0.11642
## 100     0.11708    0.11642
## 101     0.11708    0.11642
## 102     0.11708    0.11642
## 103     0.11708    0.11642
## 104     0.11708    0.11642
## 105     0.11708    0.11642
## 106     0.11708    0.11642
## 107     0.11708    0.11642
## 108     0.11708    0.11642
## 109     0.11708    0.11642

```

```

# 5-year age
# identify flat female LTs
which(q5.f[23,]==q5.f[19,])

##      female.ISL.1852 female.NZL_MA.1949
##          2901           3887
## female.NZL_MA.1956 female.NZL_MA.1959
##          3894           3897

# verify that they are constant at roughly ages 80+
q5.f[15:23,which(q5.f[23,]==q5.f[19,])]

##      female.ISL.1852 female.NZL_MA.1949
## 65-69        0.22643       0.27392
## 70-74        0.24125       0.30321
## 75-79        0.35970       0.52667
## 80-84        0.43058       0.45759
## 85-89        0.43058       0.45759
## 90-94        0.43058       0.45759
## 95-99        0.43058       0.45759
## 100-104      0.43058       0.45759
## 105-109      0.43058       0.45759
##      female.NZL_MA.1956 female.NZL_MA.1959
## 65-69        0.27451       0.22760
## 70-74        0.35737       0.32634
## 75-79        0.42851       0.51114
## 80-84        0.46347       0.46145
## 85-89        0.46347       0.46145
## 90-94        0.46347       0.46145
## 95-99        0.46347       0.46145
## 100-104      0.46347       0.46145
## 105-109      0.46347       0.46145

## males -- ALL OK
# 1-year age
# identify flat female LTs
which(q1.m[106,]==q1.m[110,])

## named integer(0)

# verify that they are constant at roughly ages 80+
q1.m[75:110,which(q1.m[106,]==q1.m[110,])]

## 
## 74
## 75
## 76
## 77
## 78
## 79
## 80
## 81
## 82
## 83
## 84
## 85

```

```

## 86
## 87
## 88
## 89
## 90
## 91
## 92
## 93
## 94
## 95
## 96
## 97
## 98
## 99
## 100
## 101
## 102
## 103
## 104
## 105
## 106
## 107
## 108
## 109

# 5-year age
# identify flat female LTs
which(q5.m[23,]==q5.m[19,])

## named integer(0)
# verify that they are constant at roughly ages 80+
q5.m[15:23,which(q5.m[23,]==q5.m[19,])]

## 
## 65-69
## 70-74
## 75-79
## 80-84
## 85-89
## 90-94
## 95-99
## 100-104
## 105-109

# remove all flat LTs that were flat for either females or males
# from both female and male collections.
remove.1 <- unique(c(which(q1.f[106,]==q1.f[110,]),which(q1.m[106,]==q1.m[110,])))
remove.5 <- unique(c(which(q5.f[23,]==q5.f[19,]),which(q5.m[23,]==q5.m[19,])))

# are the remove lists the same?
identical(remove.1,remove.5)

## [1] TRUE
# remove them from both sexes and verify
q1.f <- q1.f[,-remove.1]

```

```

which(q1.f[106,]==q1.f[110,])

## named integer(0)
q1.m <- q1.m[,-remove.1]
which(q1.m[106,]==q1.m[110,])

## named integer(0)
q5.f <- q5.f[,-remove.5]
which(q5.f[23,]==q5.f[19,])

## named integer(0)
q5.m <- q5.m[,-remove.5]
which(q5.m[23,]==q5.m[19,])

## named integer(0)
e1.f <- e1.f[,-remove.1]
e1.m <- e1.m[,-remove.1]

e5.f <- e5.f[,-remove.5]
e5.m <- e5.m[,-remove.5]

a1.f <- a1.f[,-remove.1]
a1.m <- a1.m[,-remove.1]

a5.f <- a5.f[,-remove.5]
a5.m <- a5.m[,-remove.5]

l1.f <- l1.f[,-remove.1]
l1.m <- l1.m[,-remove.1]

l5.f <- l5.f[,-remove.5]
l5.m <- l5.m[,-remove.5]

# make sure all matrices have same number of columns and same
# column names
dim(q1.f)

## [1] 110 4902
dim(q1.m)

## [1] 110 4902
dim(q5.f)

## [1] 23 4902
dim(q5.m)

## [1] 23 4902
dim(e1.f)

## [1] 111 4902

```

```

dim(e1.m)

## [1] 111 4902
dim(e5.f)

## [1] 24 4902
dim(e5.m)

## [1] 24 4902
dim(a1.f)

## [1] 111 4902
dim(a1.m)

## [1] 111 4902
dim(a5.f)

## [1] 24 4902
dim(a5.m)

## [1] 24 4902
dim(l1.f)

## [1] 111 4902
dim(l1.m)

## [1] 111 4902
dim(l5.f)

## [1] 24 4902
dim(l5.m)

## [1] 24 4902
# NB, last argument in str_sub intentionally empty, defaults to end of string
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q1.m),6,))

## [1] TRUE
identical(str_sub(colnames(q5.f),8,),str_sub(colnames(q5.m),6,))

## [1] TRUE
identical(str_sub(colnames(e1.f),8,),str_sub(colnames(e1.m),6,))

## [1] TRUE
identical(str_sub(colnames(e5.f),8,),str_sub(colnames(e5.m),6,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.f),8,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.m),6,))
```

```

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.f),8,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.m),6,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l1.f),8,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l1.m),6,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l5.f),8,))

## [1] TRUE
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l5.m),6,))

## [1] TRUE
rm(list=c("remove.1","remove.5"))

```

The last data cleaning step involves identifying nq_x values that are zero and replacing these with very small numbers. This is necessary so that we use the log function to transform these.

```

length(q1.f) # values in q1.f
## [1] 539220
length(q1.f[q1.f==0]) # zero cells in q1.f
## [1] 3190
length(q5.f) # values in q5.f
## [1] 112746
length(q5.f[q5.f==0]) # zero cells in q5.f
## [1] 99
length(q1.m) # values in q1.m
## [1] 539220
length(q1.m[q1.m==0]) # zero cells in q1.m
## [1] 1732
length(q5.m) # values in q5.m
## [1] 112746
length(q5.m[q5.m==0]) # zero cells in q5.m
## [1] 41
# female
q1.f.nz <- q1.f

```

```

q1.f.nz[q1.f.nz==0] <- 0.000001
q5.f.nz <- q5.f
q5.f.nz[q5.f.nz==0] <- 0.000001

# male
q1.m.nz <- q1.m
q1.m.nz[q1.m.nz==0] <- 0.000001
q5.m.nz <- q5.m
q5.m.nz[q5.m.nz==0] <- 0.000001

cat("\n")

length(q1.f.nz) # values in q1.f

## [1] 539220
length(q1.f.nz[q1.f.nz==0]) # zero cells in q1.f

## [1] 0
length(q5.f.nz) # values in q5.f

## [1] 112746
length(q5.f.nz[q5.f.nz==0]) # zero cells in q5.f

## [1] 0
length(q1.m.nz) # values in q1.m

## [1] 539220
length(q1.m.nz[q1.m.nz==0]) # zero cells in q1.m

## [1] 0
length(q5.m.nz) # values in q5.m

## [1] 112746
length(q5.m.nz[q5.m.nz==0]) # zero cells in q5.m

## [1] 0

```

Take the log and logit transforms of the nq_x values.

```

# function for logit transformation
logit <- function(x) {
  return(log(x/(1-x)))
}

# function for inverse logit transformation
expit <- function(x) {
  return(exp(x)/(1+exp(x)) )
}

# log and logit transform the female nqx
q11.f <- log(q1.f.nz)
q1logit.f <- logit(q1.f.nz)
q51.f <- log(q5.f.nz)

```

```

q5logit.f <- logit(q5.f.nz)

# log and logit transform the male nqx
q11.m <- log(q1.m.nz)
q1logit.m <- logit(q1.m.nz)
q51.m <- log(q5.m.nz)
q5logit.m <- logit(q5.m.nz)

```

Check how many life tables are left and be sure all the data objects have the same number of life tables and age groups.

```

dim(q11.f)

## [1] 110 4902
dim(q1logit.f)

## [1] 110 4902
dim(q51.f)

## [1] 23 4902
dim(q5logit.f)

## [1] 23 4902
dim(q11.m)

## [1] 110 4902
dim(q1logit.m)

## [1] 110 4902
dim(q51.m)

## [1] 23 4902
dim(q5logit.m)

## [1] 23 4902

```

2.3 Additional Indicator Calculation

We need child, $5q_0$, and adult, $45q_{15}$, mortality values for females and males. Calculate these from the 1×1 nq_x values and store in separate matrices, including the log and logit transformed values.

```

# function to generate 5q0 from a matrix of 1qx
convert1qxTo5q0 <- function(q1) {

  # q1 is an age by life table matrix of 1qx
  # q5 is 1 by life table matrix/vector of 5q0

  tmp.q <- rep(1,ncol(q1))
  for (i in 1:5) {
    tmp.q <- tmp.q * (1-q1[i,])
  }
  q5 <- as.matrix(1-tmp.q)
  return(q5)
}

```

```

}

# function to generate 45q15 from a matrix of 1qx
convert1qxTo45q15 <- function(q1) {

  # q1 is an age by life table matrix of 1qx
  # q5 is 1 by life table matrix/vector of 45q15

  tmp.q <- rep(1,ncol(q1))
  for (i in 16:60) {
    tmp.q <- tmp.q * (1-q1[i,])
  }
  q5 <- as.matrix(1-tmp.q)
  return(q5)
}

# now actually create the child and adult mortality indicators

# female

# make matrix with 5q0 in row 1 and 45q15 in row 2
Q.f <- rbind(t(convert1qxTo5q0(q1.f)),t(convert1qxTo45q15(q1.f)))
# check for zeroes
Q.f[Q.f==0]

## numeric(0)

# log and logit
Q1.f <- log(Q.f)
Qlogit.f <- logit(Q.f)

colnames(Q.f) <- colnames(q1.f)
colnames(Q1.f) <- colnames(q1.f)
colnames(Qlogit.f) <- colnames(q1.f)

rownames(Q.f) <- c("Child Mortality","Adult Mortality")
rownames(Q1.f) <- c("Child Mortality","Adult Mortality")
rownames(Qlogit.f) <- c("Child Mortality","Adult Mortality")

# male

# make matrix with 5q0 in row 1 and 45q15 in row 2
Q.m <- rbind(t(convert1qxTo5q0(q1.m)),t(convert1qxTo45q15(q1.m)))
# check for zeroes
Q.m[Q.m==0]

## numeric(0)

# log and logit
Q1.m <- log(Q.m)
Qlogit.m <- logit(Q.m)

colnames(Q.m) <- colnames(q1.m)
colnames(Q1.m) <- colnames(q1.m)
colnames(Qlogit.m) <- colnames(q1.m)

```

```

rownames(Q.m) <- c("Child Mortality", "Adult Mortality")
rownames(Ql.m) <- c("Child Mortality", "Adult Mortality")
rownames(Qlogit.m) <- c("Child Mortality", "Adult Mortality")

```

We now have everything we need to get going. Clean up or clear stuff we don't need and save everything.

```

# rm(list=c("download.result", "hmd.1x1.list", "hmd.5x1.list", "remove", "remove.1"
#       , "remove.5", "i", "tmp.q", "count.lts", "download.hmd", "extract.lt.col", "list.lts"
#       , "list.raw.lts", "parse.lt", "read.raw.lt", "sex.per.age.switch"))
save.image("../RData/hmd.qs.RData")
# load("../RData/hmd.qs.RData")

```

3 SVD Component Model of Mortality

3.1 *svdMod()* function

svdMod() is a function that wraps up most of the operations needed to calculate and validate SVD-Comp models. This function does a lot and can be used in a variety of ways:

- Calculate/estimate an SVD-component mortality model using a set of age-specific nq_x as inputs
- Calculate/estimate a smoothed SVD-component mortality model using a set of age-specific nq_x as inputs
- Randomly sample a set of age-specific nq_x , calculate an SVD-component model of mortality (smoothed or not), predict nq_x for the not-sampled age-specific nq_x , and summarize the prediction errors
- All of this can be repeated a specified number of times
- The return object contains very detailed results for everything that was requested

Inputs to the function:

- ‘ql’ are the logit-transformed input age-specific nq_x (life tables) arranged as age×lifetable
- ‘Ql’ are the logit-transformed summary mortality indicators: child mortality, $5q_0$, and adult mortality, $45q_{15}$, arranged in $2 \times n$ form where the first row is child mortality, the second adult mortality, and the columns correspond to life tables
- ‘N’ is the number of times to repeat sampling/validation
- ‘S’ is the fraction of life tables to include in the sample
- ‘offset’ is a number used to offset the the age-specific mortality rates from the origin before calculating the SVD; this is an easy way to give each age group approximately the same weight in the SVD calculation; it is added back when predictions are made
- ‘retAll’ is a switch indicating if all results should be returned or just summaries
- ‘adult’ is a switch indicating if adult mortality, $45q_{15}$, is supplied and should be used directly as an input to the model when predictions are made; if not, then child mortality, $5q_0$, is the only direct input, and adult mortality is used *indirectly* by predicting it from child mortality and then using it together with child mortality for the predictions for other ages
- ‘q0Fix’ is a switch indicating if the q_0 fix should be executed during predition
- ‘smooth’ is a switch indicating if the SVD-comp model should be smoothed
- ‘C’ specifies the number of components to include in the SVD-component model

The return object is a large list that contains:

- ‘ql.samp’ - a list of the sampled age-specific nq_x , i.e. life tables, (one for each sample)
- ‘ql.nsamp’ - a list of the not sampled age-specific nq_x (one for each sample)
- ‘names’ - the names of the sampled life tables
- ‘svd’ - a list of the SVD decompositions (one for each sample) of the sampled life tables
- ‘svd.sm’ - a list of the smoothed SVD decompositions (one for each sample) of the sampled life tables
- ‘mods’ - a list of the the regression return objects (one for each sample) from the regression models for each SVD component weight in the model, the model for adult mortality, and the model for the q0 fix

- ‘recon.samp’ - a list of the predicted life tables (one for each sample) for life tables in the sample
- ‘error.samp’ - a list of the prediction errors (one for each sample) for the sampled life tables
- ‘recon.nsamp’ - a list of the predicted life tables (one for each sample) for life tables not in the sample
- ‘error.nsamp’ - a list of the prediction errors (one for each sample) for the not sampled life tables
- ‘errsum.samp’ - a list of summaries (one for each sample) of in-sample errors, uses R’s *summary()* function
- ‘errsum.nsamp’ - a list of summaries (one for each sample) of out-of-sample errors, uses R’s *summary()* function
- ‘offset’ - the *offset* value used when the function was called
- ‘retAll’ - the *retAll* value used when the function was called
- ‘adult’ - the *adult* value used when the function was called
- ‘q0fix’ - the *q0fix* value used when the function was called
- ‘smooth’ - the *smooth* value used when the function was called
- ‘C’ - the *C* value used when the function was called

A couple notes:

- The input age-specific nq_x must all be logit-transformed, the function assumes this and uses an *expit* transformation to do predictions on the natural scale
- The ‘mods’ return object is very useful for doing predictions and building additional modeling features using the return object of this function
- the ‘retAll’ option is included because full results can be *very* large, and returning the summaries is a much more compact way to do things if you need to run many times and don’t need the detailed results each time

```
svdMod <- function(ql,Ql,N,S,offset,retAll,adult,q0Fix,smooth,C=4,printS=FALSE) {

  # ql is input qs
  # Ql is input summary indicators (child and adult ql)
  # N is number of samples
  # S is sample fraction
  # offset is the SVD offset
  # retAll is switch to return 'all' or just error summaries
  # adult is a switch indicating whether to include adult mortality in the model
  # q0Fix is a switch to execute the q0 fix
  # smooth is a switch to smooth left singular vectors
  # C is number of components, default C=4
  # printS is a switch to turn off printing the sample number at each iteration

  ret.ql.samp <- list() # sampled qls
  ret.ql.nsamp <- list() # out of sample qls
  ret.samp.names <- list() # sampled LT names
  ret.svd <- list() # svd of sampled qls
  ret.svd.sm <- list() # smooth svd of sampled qls
  ret.mods <- list() # models
  ret.recon.samp <- list() # sample reconstructions
  ret.error.samp <- list() # sample errors
  ret.recon.nsamp <- list() # out of sample reconstructions
  ret.error.nsamp <- list() # out of sample errors
  ret.errsum.samp <- list() # summary of sample errors
  ret.errsum.nsamp <- list() # summary of out of sample errors

  # Ensure C is in reasonable range: 1-4
  if (C < 1 | C > 4) {
    C = 4
  }
}
```

```

    print("Setting C=4")
}

cat("\n")
print(paste("Adult mortality is direct input to predictions:",adult))
print(paste("SVD model is smoothed:",smooth))
print(paste(N,"iterations"))
print(paste(round(S*100,0),"% sample fraction",sep=""))
print(paste(C,"components"))

if (S > 0) {

  for (i in 1:N) {

    if (printS) {print(paste(" Sample:",i))}

    # pick the sample
    if (S == 1) {
      samp <- colnames(ql) # the sample is all LTs
      nsamp <- NA # nothing in the out of sample
    } else {
      # identify sample
      samp <- sample(colnames(ql),ncol(ql)*S)
      # identify out of sample
      nsamp <- colnames(ql)[-which(colnames(ql) %in% samp)]
    }
    name <- paste("s",i,sep="") # give this sample a name

    # store the sample
    ret.samp.names[[i]] <- samp # store sampled LT names to return list
    names(ret.samp.names)[i] <- name # name the sampled LT names

    # store the sampled qls
    ret ql.samp[[i]] <- ql[,samp] # store sampled qls in return list
    names(ret ql.samp)[i] <- name # name the sampled qls

    # store the out of sample qls
    if (S == 1) {
      ret ql.nsamp[[i]] <- NA # no out of sample LTs
    } else {
      ret ql.nsamp[[i]] <- ql[,nsamp] # store out of sample qls in return list
    }
    names(ret ql.nsamp)[i] <- name # name out of sample qls

    # calculate the svd of the sampled qls
    svd <- svd(ql[,samp] - offset) # subtract offset before calculating svd

    # store the SVD
    ret.svd[[i]] <- svd # store svd in return list
    names(ret.svd)[i] <- name # name the svd

    # calculate transformations of *sampled* child mortality: input is logged
    cm <- expit(QL[1,samp]) # child mortality, natural scale
  }
}

```

```

cml <- Q1[1,samp] # child mortality, logged
cmls <- cml^2 # square of logged child mortality
cmcl <- cml^3 # cube of logged child mortality

# calculate transformations of *sampled* adult mortality: input is logged
am <- expit(Q1[2,samp]) # adult mortality, natural scale
aml <- Q1[2,samp] # adult mortality, logged
ams <- aml^2 # square of logged adult mortality
amlc <- aml^3 # cube of logged adult mortality

# calculate one-way interaction of *sampled* child and adult mortality
cmlaml <- cml*aml

# model *sampled* adult mortality ~ child mortality
aml.betas <- lm(aml ~ cm + cml + cmls + cmcl)

# model *sampled* first four vs ~ child mortality and adult mortality
v1.betas <- lm(svd$v[,1] ~ cm + cml + cmls + cmcl + am + ams + amlc + cmlaml)
v2.betas <- lm(svd$v[,2] ~ cm + cml + cmls + cmcl + am + ams + amlc + cmlaml)
v3.betas <- lm(svd$v[,3] ~ cm + cml + cmls + cmcl + am + ams + amlc + cmlaml)
v4.betas <- lm(svd$v[,4] ~ cm + cml + cmls + cmcl + am + ams + amlc + cmlaml)

# predictions for all LTs, both sampled and out of sample
# start by transforming child mortality for all LTs
cml.p <- Q1[1,]
cm.p <- expit(cml.p)
cmls.p <- cml.p^2
cmcl.p <- cml.p^3

# predict the adult mortality that goes with this child mortality
# data frame of predictors
predictors.aml <- data.frame(cbind(cm.p,cml.p,cmls.p,cmcl.p))
# names for predictors that match the variable in the original model
colnames(predictors.aml) <- c("cm","cml","cmls","cmcl")
# predictions for adult mortality
if (adult) {
  aml.p <- Q1[2,]
} else {
  aml.p <- predict.lm(aml.betas,newdata=predictors.aml)
}

# predict vs using child mortality and (predicted) adult mortality
# transform predicted adult mortality
am.p <- expit(aml.p)
ams.p <- aml.p^2
amlc.p <- aml.p^3
cmlaml.p <- cmcl.p*aml.p
# data frame of predictors
predictors.vs <- data.frame(cbind(
  cm.p,cml.p,cmls.p,cmcl.p,am.p,ams.p,amlc.p,cmlaml.p))
# names for predictors that match the variables in the original regressions
colnames(predictors.vs) <- c(
  "cm","cml","cmls","cmcl","am","ams","amlc","cmlaml")

```

```

# predictions for each v
v1.p <- predict.lm(v1.betas,newdata=predictors.vs)
v2.p <- predict.lm(v2.betas,newdata=predictors.vs)
v3.p <- predict.lm(v3.betas,newdata=predictors.vs)
v4.p <- predict.lm(v4.betas,newdata=predictors.vs)

# smooth left singular vectors
if (smooth) {
  for (k in 2:6) {
    t <- ksmooth(seq(1,dim(svd$u)[1],1),svd$u[,k],kernel = "normal", bandwidth = (k+1))
    t$y[1:(k-1)] <- svd$u[,k][1:(k-1)]
    svd$u[,k] <- t$y
  }
  # store the smooth SVD
  ret.svd.sm[[i]] <- svd # store the smooth svd in return list
  names(ret.svd.sm)[i] <- name # name the smooth svd
} else {
  ret.svd.sm[[i]] <- NA
  names(ret.svd.sm)[i] <- name # name the smooth svd
}

# reconstruct the predicted LTs
v <- cbind(v1.p,v2.p,v3.p,v4.p)      # matrix of new predicted vs
r.p <- ql - ql # data frame for reconstructed values with names
for (z in 1:C) { # loop over C components
  # svd reconstruction; sum of rank-1 matrices, one for each v
  r.p <- r.p + svd$d[z] * svd$u[,z] %*% t(v[,z])
}
r.p <- r.p + offset # add the offset back in

if (q0Fix) {
  # fix up q0 prediction
  # child mortality for sample LTs
  cml <- Q1[1,samp] # sample child mortality
  cmcls <- cml^2 # square of sample child mortality
  q0.betas <- lm(as.numeric(q1[1,samp]) ~ cml + cmcls) # q0 ~ cml + cmcls
  # predictors for all LTs
  cml.p <- Q1[1,] # child mortality for all LTs
  cmcls.p <- cml.p^2 # square of child mortality for all LTs
  predictors.q0 <- data.frame(cbind(cml.p,cmcls.p))
  colnames(predictors.q0) <- c("cml","cmcls")
  q0.p <- predict.lm(q0.betas,newdata=predictors.q0)
  # replace the predicted values for q0 with those from model above
  r.p[1,] <- q0.p
} else {
  q0.betas <- NA
}

# store the models
ret.mods[[i]] <- list(
  aml=aml.betas
  ,v1=v1.betas
  ,v2=v2.betas
)

```

```

,v3=v3.betas
,v4=v4.betas
,q0=q0.betas)
names(ret.mods)[i] <- name

# results: sampled LTs
# store the reconstructed sampled LTs
ret.recon.samp[[i]] <- r.p[,samp]
names(ret.recon.samp)[i] <- name

# store the errors in the reconstructed sampled LTs
ret.error.samp[[i]] <- expit(q1[,samp]) - expit(r.p[,samp])
names(ret.error.samp)[i] <- name

# store summaries of errors in sampled LTs
ret.errsum.samp[[i]] <-
summary(as.vector(as.matrix(ret.error.samp[[i]])))
names(ret.errsum.samp)[i] <- name

if (S == 1) {
  # results: out of sample LTs
  # store the reconstructed out of sample LTs
  ret.recon.nsamp[[i]] <- NA
  names(ret.recon.nsamp)[i] <- name

  # store the errors in the reconstructed out of sample LTs
  ret.error.nsamp[[i]] <- NA
  names(ret.error.nsamp)[i] <- name

  # store the summaries of errors in out of sample LTs
  ret.errsum.nsamp[[i]] <- NA
  names(ret.errsum.nsamp)[i] <- name
} else {
  # results: out of sample LTs
  # store the reconstructed out of sample LTs
  ret.recon.nsamp[[i]] <- r.p[,nsamp]
  names(ret.recon.nsamp)[i] <- name

  # store the errors in the reconstructed out of sample LTs
  ret.error.nsamp[[i]] <- expit(q1[,nsamp]) - expit(r.p[,nsamp])
  names(ret.error.nsamp)[i] <- name

  # store the summaries of errors in out of sample LTs
  ret.errsum.nsamp[[i]] <-
summary(as.vector(as.matrix(ret.error.nsamp[[i]])))
  names(ret.errsum.nsamp)[i] <- name
}

# put all the return lists together into one big list
if (retAll == TRUE) {
  return(list(

```

```

        ql.samp = ret ql.samp # sampled qls
        ,ql.nsamp = ret ql.nsamp # out of sample qls
        ,names = ret samp.names # sampled LT names
        ,svd = ret svd # svd of sampled qls
        ,svd.sm = ret svd.sm # smooth svd of sampled qls
        ,mods = ret mods # models
        ,recon.samp = ret recon.samp # sample reconstructions
        ,error.samp = ret error.samp # sample errors
        ,recon.nsamp = ret recon.nsamp # out of sample reconstructions
        ,error.nsamp = ret error.nsamp # out of sample errors
        ,errsum.samp = ret errsum.samp # summary of sample errors
        ,errsum.nsamp = ret errsum.nsamp # summary of out of sample errors
        ,offset = offset # the offset necessary to reconstruct
        ,retAll = retAll
        ,adult = adult
        ,q0fix = q0Fix
        ,smooth = smooth
        ,C=C
    ))
} else {
    return(list(
        errsum.samp = ret errsum.samp # summary of sample errors
        ,errsum.nsamp = ret errsum.nsamp # summary of out of sample errors
    ))
}
} else {
    print("S must be larger than 0.0")
    return()
}

print("Done")
}

```

3.2 *ltPredict()* function

ltPredict() is a function that uses a return object from *svdMod()* to predict new life tables.

Inputs to the function:

- ‘mods’ is an output object from *svdMod()*
- ‘smooth’ is a switch indicating if the smoothed left singular vectors should be used for the prediction
- ‘cml’ is a value/vector for the input level(s) of logit-scale child mortality, ${}_5q_0$
- ‘aml’ is a value/vector for the input level(s) of logit-scale adult mortality, ${}_{45}q_{15}$

The return object is:

- ‘r.p’ – a dataframe containing the predicted life table(s)

```

ltPredict <- function(mods,smooth,cml,aml) {

    # mods is an output object from svdMod()
    # cml is a vector of logit child mortality rates
    # aml is a vector of logit adult mortality rates
    # if aml not supplied, then aml predicted from cml
    # smooth is a switch to use smoothed SVD if it's available

```

```

cm <- expit(cml)
cmls <- cml^2
cmlc <- cml^3
preds.aml <- data.frame(
  cm = as.numeric(cm),
  cml = as.numeric(cml),
  cmls = as.numeric(cmls),
  cmlc = as.numeric(cmlc)
)
if(missing(aml)) {
  # predict adult mortality
  aml <- predict(mods$mods$s1$aml,newdata=preds.aml)
}

# predict vs
am <- expit(aml)
amls <- aml^2
amlc <- aml^3
cmlaml <- cml*aml
preds.vs <- data.frame(
  cm = as.numeric(cm),
  cml = as.numeric(cml),
  cmls = as.numeric(cmls),
  cmlc = as.numeric(cmlc),
  am = as.numeric(am),
  amls = as.numeric(amls),
  amlc = as.numeric(amlc),
  cmlaml = as.numeric(cmlaml)
)
v1 <- predict(mods$mods$s1$v1,newdata=preds.vs)
v2 <- predict(mods$mods$s1$v2,newdata=preds.vs)
v3 <- predict(mods$mods$s1$v3,newdata=preds.vs)
v4 <- predict(mods$mods$s1$v4,newdata=preds.vs)

# if smoothed SVD available
if (smooth & mods$smooth) {
  svd <- mods$svd.sm$s1
} else {
  svd <- mods$svd$s1
}

# construct LTs
v <- cbind(v1,v2,v3,v4)
r.p <- matrix(data=0,ncol=length(cml),nrow=length(svd$u[,1]))
for (z in 1:4) {
  r.p <- r.p + svd$d[z] * svd$u[,z] %*% t(v[,z])
}
r.p <- r.p + mods$offset

# fix q0
if (mods$q0fix) {
  cmls <- cml^2
  preds.q0 <- data.frame(

```

```

    cml = as.numeric(cml),
    cmls = as.numeric(cmls)
)
r.p[1,] <- predict(mods$mods$s1$q0,newdata=preds.q0)
}

# fix up r.p
r.p <- data.frame(r.p)
colnames(r.p) <- paste("cml.",cml,sep="")
rownames(r.p) <- rownames(mods$ql.samp$s1)

# returns the matrix of predicted values
return(r.p)
}

```

4 Validation

First, run the *svdComp()* function with one iteration and a 100% sample in both ‘base’ and ‘smoothed’ form using only child mortality as a direct input, i.e. ‘adult’ is set to FALSE, and specifying two components. This will yield SVD-Comp models calibrated on the entire HMD data set. The *svdComp()* function provides a little feedback, here indicating that two-component models were run on one sample of 100% with the child mortality-only model, either base or smoothed.

```

# specify some key parameters, just to be a bit more readable!
adult <- FALSE
smooth <- FALSE
N <- 1
S <- 1
C <- 4
offset <- 10
# base model
mod.1_0.m <- svdMod(q1logit.m,Qlogit.m,N,S,offset,TRUE,adult,TRUE,smooth,C)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

mod.1_0.f <- svdMod(q1logit.f,Qlogit.f,N,S,offset,TRUE,adult,TRUE,smooth,C)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

# smooth now
smooth <- TRUE
mod.1_0.sm.m <- svdMod(q1logit.m,Qlogit.m,N,S,offset,TRUE,adult,TRUE,smooth,C)

##

```

```

## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: TRUE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"
mod.1_0.sm.f <- svdMod(q1logit.f,Qlogit.f,N,S,offset,TRUE,adult,TRUE,smooth,C)

```

```

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: TRUE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

```

To compare the predicted results from the SVD-comp model calibrated with the entire HMD to results produced by Wilmoth et al.'s Log Quad model, we must calculate five-year age group probabilities of dying, tq_x , because Log Quad operates only with five-year age groups. The following code uses the single-year age group predictions from SVD-comp to calculate five-year age group probabilities of dying.

```

# fast functions to calculate a vector of 5qx
#   for any five-year age group indexed from 0 by 1

# ages 1-4
oneToFourYear <- function(oneYear) {
  return(1-prod(sapply(2:5, function(x,y) (1-y[x]), y=oneYear)))
}

# five-year age groups
fiveYear <- function(start,oneYear) {
  return(1-prod(sapply((start*5+1):(start*5+5), function(x,y) (1-y[x]), y=oneYear)))
}

# fast function to convert full schedule of 1qx into full schedule of 5qx
fiveYearQ <- function(oneYear) {
  sapply(0:(trunc(length(oneYear)/5-1)), function(x,y) fiveYear(x,y), y=oneYear)
}

# fast function to calculate 45q15 from 1qx
adultQ <- function(oneYear) {
  return(1-prod(sapply(16:60, function(x,y) (1-y[x]), y=oneYear)))
}

# fast function to calculate 5q0 from standard 5qx
childQ5 <- function(fiveYear) {
  return(1-prod(sapply(1:2, function(x,y) (1-y[x]), y=fiveYear)))
}

# fast function to calculate 45q15 from 5qx
adultQ5 <- function(fiveYear) {
  return(1-prod(sapply(5:13, function(x,y) (1-y[x]), y=fiveYear)))
}

# fast function to calculate a full standard 5-year age schedule
standardFiveYear <- function(oneYear) {
  l <- trunc(length(oneYear)/5)

```

```

  c(oneYear[1],oneToFourYear(oneYear),fiveYearQ(oneYear)[2:1])
}

# examples
# using single age schedule of 1qx
# adultQ(<data>)
# fiveYearQ(<data>)
# standardFiveYear(<data>)
# using a matrix of age schedules of 1qx
# apply(data,2,<function:adultQ or fiveYearQ or standardFiveYear>)

# function to calculate a matrix of 5qx from a matrix of 1qx
convert1qxTo5qxApply <- function(q1) {

  # q1 contains values of 1qx and is an age by life table matrix
  # with at least two columns

  # q5 is returned: an age by life table matrix with 5qx

  q5 <- apply(q1,2,standardFiveYear)
  colnames(q5) <- colnames(q1)
  rNames <- c("0","1-4")
  for (i in seq(1,(trunc(nrow(q1)/5)-1),1)) {
    rNames <- c(rNames,paste(i*5,(i*5+4),sep="-"))
  }
  rownames(q5) <- rNames
  return(q5)
}

# simpler and more readable approach which turns out to be roughly as fast
# or faster in this markdown document!

# function to convert 1qx to standard age group 5qx
convert1qxTo5qx <- function(q1) {

  # q1 contains values of 1qx and is an age by life table matrix
  # q5 is returned: an age by life table matrix with 5qx

  q5 <- matrix(data=0,ncol=ncol(q1),nrow=23)
  rNames <- rep("",23)
  # age 0
  q5[1,] <- as.matrix(q1[1,])
  rNames[1] <- "0"
  # ages 1-4
  tmp.q <- rep(1,ncol(q1))
  for (i in 2:5) {
    tmp.q <- tmp.q * (1-q1[i,])
  }
  q5[2,] <- as.matrix(1-tmp.q)
  rNames[2] <- "1-4"
  # five-year age groups for ages 5-105 (ending 110)
  for (j in 1:21) {
    tmp.q <- rep(1,ncol(q1))
}

```

```

    for (i in (j*5+1):(j*5+5)) {
      tmp.q <- tmp.q * (1-q1[i,])
    }
  q5[(j+2),] <- as.matrix(1-tmp.q)
  rNames[(j+2)] <- paste((j*5),"-",(j*5+4),sep="")
}
rownames(q5) <- rNames
colnames(q5) <- colnames(q1)
return(q5)
}

# compare the speed of the two approaches
start.time.apply <- Sys.time()
tmp.apply <- convert1qxTo5qxApply(expit(mod.1_0.f$recon.samp$s1))
stop.time.apply <- Sys.time()
start.time.loop <- Sys.time()
tmp.loop <- convert1qxTo5qxApply(expit(mod.1_0.f$recon.samp$s1))
stop.time.loop <- Sys.time()
# results!
print(paste("Loop way:",stop.time.loop-start.time.loop))

## [1] "Loop way: 1.10127782821655"
print(paste("Apply way:",stop.time.apply-start.time.apply))

## [1] "Apply way: 1.10848879814148"
# apply way usually a tiny bit faster

# check to be sure they produce same answer
all.equal(tmp.loop,tmp.apply)

## [1] TRUE
# looks like it!
rm(list=c("tmp.loop","tmp.apply"))

# Now actually calculate the 5qx schedules from the predicted 1qx

# female
q5p.f <- convert1qxTo5qxApply(expit(mod.1_0.f$recon.samp$s1))

# male
q5p.m <- convert1qxTo5qxApply(expit(mod.1_0.m$recon.samp$s1))

```

R code supplied by Wilmoth et al. is used to calculate the predicted five-year age group probabilities of dying using the Log Quad model using the same inputs as those used by SVD-Comp: the $5q_0$ and $45q_{15}$ values stored in the ‘Q.f’ and ‘Q.m’ matrices – logit-transformed (‘Qlogit.f’ and ‘Qlogit.m’) for SVD-Comp. For more information on the Log Quad model code download here (www.demog.berkeley.edu/~jrw/LogQuad). First create a function to do the comparisons.

```

# function to conduct comparison of predicted 5qx from SVD-Comp
# and Log-Quad
doComparison <- function(q1logit.f,Qlogit.f,q1logit.m,Qlogit.m
                          ,q5.f,q5.m,N,S,offset,adult,smooth,C) {

```

```

# q1logit.f - female logit 1qx
# Qlogit.f - female child and adult mortality levels
# q1logit.m - male logit 1qx
# Qlogit.m - male child and adult mortality levels
# q5.f - female 5qx values from HMD site
# q5.m - male 5qx values from HMD site
# N - number of samples taken
# S - sample fraction
# offset - size of offset
# adult - include adult mortality directly
# smooth - use smoothing
# C - number of components to use

# rerun models setting using adult mortality directly
mod.1_0.m <- svdMod(q1logit.m,Qlogit.m,N,S,10,TRUE,adult,TRUE,smooth,C)
mod.1_0.f <- svdMod(q1logit.f,Qlogit.f,N,S,10,TRUE,adult,TRUE,smooth,C)

# store the predicted values from the model in five-year age groups
q5p.f <- convert1qxTo5qxApply(expit(mod.1_0.f$recon.samp$s1))
q5p.m <- convert1qxTo5qxApply(expit(mod.1_0.m$recon.samp$s1))

# create Log-Quad predictions
# fit the log-quad using child mortality only

# Source functions file
source("../R/logQuad/DataProgramsExamples/R/functions.R")

# Create labels for age vectors
ages.5x1 <- c("0","1-4",paste(seq(5,105,5),seq(9,109,5),sep="-"),"110+")
sexes <- c("Female","Male","Total")

# Import matrix of model coefficients
tmp1 <- read.csv("../R/logQuad/DataProgramsExamples/Data/coefs.logquad.HMD719.csv")
tmp2 <- array(c(as.matrix(tmp1[, 3:6]))
              , dim=c(24, 3, 4)
              , dimnames=list(ages.5x1, sexes, c("ax", "bx", "cx", "vx")))
coefs <- apem(tmp2, c(1,3,2))

# female
q5.lq.f <- q5.f - q5.f
e5.lq.f <- rbind(q5.f - q5.f,rep(0,ncol(q5.f)))
a5.lq.f <- rbind(q5.f - q5.f,rep(0,ncol(q5.f)))
l5.lq.f <- rbind(q5.f - q5.f,rep(0,ncol(q5.f)))
for (i in 1:ncol(q5.f)) {
  if (adult) {
    lqfit <- lthat.any2.logquad(coefs,"Female",Q5=Q.f[1,i],QQa=Q.f[2,i]) # with adult
  } else {
    lqfit <- lthat.any2.logquad(coefs,"Female",Q5=Q.f[1,i],k=0) # without adult
  }
  q5.lq.f[,i] <- lqfit$lt[1:23,2]
  a5.lq.f[,i] <- lqfit$lt[1:24,3]
  e5.lq.f[,i] <- lqfit$lt[1:24,8]
  l5.lq.f[,i] <- lqfit$lt[1:24,4]
}

```

```

}

q51.lq.f <- log(q5.lq.f)
q5logit.lq.f <- logit(q5.lq.f)

# male
q5.lq.m <- q5.m - q5.m
e5.lq.m <- rbind(q5.m - q5.m, rep(0, ncol(q5.m)))
a5.lq.m <- rbind(q5.m - q5.m, rep(0, ncol(q5.m)))
l5.lq.m <- rbind(q5.m - q5.m, rep(0, ncol(q5.m)))
for (i in 1:ncol(q5.m)) {
  if (adult) {
    lqfit <- lthat.any2.logquad(coefs, "Male", Q5=Q.m[1,i], QQa=Q.m[2,i]) # with adult
  } else {
    lqfit <- lthat.any2.logquad(coefs, "Male", Q5=Q.m[1,i], k=0) # without adult
  }
  q5.lq.m[,i] <- lqfit$lt[1:23,2]
  a5.lq.m[,i] <- lqfit$lt[1:24,3]
  e5.lq.m[,i] <- lqfit$lt[1:24,8]
  l5.lq.m[,i] <- lqfit$lt[1:24,4]
}
q51.lq.m <- log(q5.lq.m)
q5logit.lq.m <- logit(q5.lq.m)

# compare the fits using the 5qx values obtained
# directly from the HMD web site, q5.f and q5.m
# construct a vector of comparison descriptors

# females
comps.f <- matrix(data = 0, nrow = 2, ncol = 3)
colnames(comps.f) <- c("total.abs.error", "mean.abs.error", "max.error")
rownames(comps.f) <- c("comp", "lq")
comps.f[1,1] <- sum(abs(q5.f - q5p.f))
comps.f[2,1] <- sum(abs(q5.f - q5.lq.f))
comps.f[1,2] <- comps.f[,1]/(ncol(q5p.f)*nrow(q5p.f))
comps.f[1,3] <- max(q5.f - q5p.f)
comps.f[2,3] <- max(q5.f - q5.lq.f)

# males
comps.m <- matrix(data = 0, nrow = 2, ncol = 3)
colnames(comps.m) <- c("total.abs.error", "mean.abs.error", "max.error")
rownames(comps.m) <- c("comp", "lq")
comps.m[1,1] <- sum(abs(q5.m - q5p.m))
comps.m[2,1] <- sum(abs(q5.m - q5.lq.m))
comps.m[1,2] <- comps.m[,1]/(ncol(q5p.m)*nrow(q5p.m))
comps.m[1,3] <- max(q5.m - q5p.m)
comps.m[2,3] <- max(q5.m - q5.lq.m)

comps <- list(
  female = comps.f,
  male = comps.m

, q5p.f = q5p.f
, q5p.m = q5p.m

```

```

,q5.lq.f = q5.lq.f
,e5.lq.f = e5.lq.f
,a5.lq.f = a5.lq.f
,15.lq.f = 15.lq.f
,q5l.lq.f = q5l.lq.f
,q5logit.lq.f = q5logit.lq.f

,q5.lq.m = q5.lq.m
,e5.lq.m = e5.lq.m
,a5.lq.m = a5.lq.m
,15.lq.m = 15.lq.m
,q5l.lq.m = q5l.lq.m
,q5logit.lq.m = q5logit.lq.m
)

return(comps)
}

```

Now compare the models first using only child mortality ${}_5q_0$ to predict and then using both child ${}_5q_0$ and adult ${}_{45}q_{15}$ mortality to predict.

```

# set basic model parameters
smooth <- FALSE
N <- 1
S <- 1
C <- 4
offset <- 10

# do comparison between SVD-Comp and Log-Quad using only child mortality
# as an input
comps.child <- doComparison(q1logit.f,Qlogit.f,q1logit.m,Qlogit.m,q5.f
                           ,q5.m,N,S,offset,adult=FALSE,smooth,C)

## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

# do comparison between SVD-Comp and Log-Quad using both child and adult
# mortality as inputs
comps.adult <- doComparison(q1logit.f,Qlogit.f,q1logit.m,Qlogit.m,q5.f
                            ,q5.m,N,S,offset,adult=TRUE,smooth,C)

##
## [1] "Adult mortality is direct input to predictions: TRUE"
## [1] "SVD model is smoothed: FALSE"

```

```

## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: TRUE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"
# have a look
cat("\n")

comps.child$female

##      total.abs.error mean.abs.error max.error
## comp      1533.818    0.01360419 0.3303233
## lq       1601.868    0.01420776 0.3968440

comps.child$male

##      total.abs.error mean.abs.error max.error
## comp      1771.413    0.01571154 0.3993513
## lq       1905.930    0.01690463 0.3792040
cat("\n")

comps.adult$female

##      total.abs.error mean.abs.error max.error
## comp      1368.131    0.01213463 0.2185885
## lq       1493.637    0.01324780 0.3865020

comps.adult$male

##      total.abs.error mean.abs.error max.error
## comp      1451.754    0.01287632 0.4037572
## lq       1574.161    0.01396201 0.3532550

```

Run 50 50% samples to summarize one-year age group prediction errors.

```

# 50 runs with 50% sampling proportion
adult <- FALSE
smooth <- FALSE
N <- 50
S <- 0.5
mod.0_5.50.m <- svdMod(q1logit.m,Qlogit.m,N,S,10,TRUE,adult,TRUE,smooth,C)

```

```

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "50% sample fraction"
## [1] "4 components"

mod.0_5.50.f <- svdMod(q1logit.f,Qlogit.f,N,S,10,TRUE,adult,TRUE,smooth,C)

```

```

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"

```

```

## [1] "50 iterations"
## [1] "50% sample fraction"
## [1] "4 components"

# female

# sampled errors
# aggregate errors by age
error.age.f <- matrix(data=0, ncol=0, nrow=length(mod.0_5.50.f$error.samp$s1[,1]))
for (i in 1:N) {
  #print(i)
  error.age.f <- cbind(error.age.f, as.matrix(mod.0_5.50.f$error.samp[[i]]))
}

# out of sample errors
#aggregate errors by age
error.age.nsamp.f <- matrix(data=0, ncol=0, nrow=length(mod.0_5.50.f$error.nsamp$s1[,1]))
for (i in 1:N) {
  #print(i)
  error.age.nsamp.f <- cbind(error.age.nsamp.f, as.matrix(mod.0_5.50.f$error.nsamp[[i]]))
}

# male

# sampled errors
# aggregate errors by age
error.age.m <- matrix(data=0, ncol=0, nrow=length(mod.0_5.50.m$error.samp$s1[,1]))
for (i in 1:N) {
  #print(i)
  error.age.m <- cbind(error.age.m, as.matrix(mod.0_5.50.m$error.samp[[i]]))
}

# out of sample errors
#aggregate errors by age
error.age.nsamp.m <- matrix(data=0, ncol=0, nrow=length(mod.0_5.50.m$error.nsamp$s1[,1]))
for (i in 1:N) {
  #print(i)
  error.age.nsamp.m <- cbind(error.age.nsamp.m, as.matrix(mod.0_5.50.m$error.nsamp[[i]]))
}

```

Run 50 samples with sampling fractions 10%, 30%, 50%, 70%, and 90% to summarize and characterize prediction errors as the sample fration varies.

```

# female
adult <- FALSE
smooth <- FALSE
N <- 50
for (S in seq(0.1,0.9,0.2)) {
  assign(paste("qlPred_", S, ".f", sep=""),
         , svdMod(q1logit.f, Q1.f, N, S, 10, FALSE, adult, TRUE, smooth, C))
}

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"

```

```

## [1] "10% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "30% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "50% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "70% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "90% sample fraction"
## [1] "4 components"

# ... this could be more elegant as a list or by usign 'assign' like above, but ...
# summary errors
errsum.meds.1.f <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.1.f <- matrix(data=0,ncol=2,nrow=N)
for (i in 1:N) {
  errsum.meds.1.f[i,1] <- qlPred_0.1.f$errsum.samp[[i]][3]
  errsum.meds.1.f[i,2] <- qlPred_0.1.f$errsum.nsamp[[i]][3]
  errsum.iqrs.1.f[i,1] <-
    qlPred_0.1.f$errsum.samp[[i]][5] - qlPred_0.1.f$errsum.samp[[i]][2]
  errsum.iqrs.1.f[i,2] <-
    qlPred_0.1.f$errsum.nsamp[[i]][5] - qlPred_0.1.f$errsum.nsamp[[i]][2]
}
# summary errors
errsum.meds.3.f <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.3.f <- matrix(data=0,ncol=2,nrow=N)
for (i in 1:N) {
  errsum.meds.3.f[i,1] <- qlPred_0.3.f$errsum.samp[[i]][3]
  errsum.meds.3.f[i,2] <- qlPred_0.3.f$errsum.nsamp[[i]][3]
  errsum.iqrs.3.f[i,1] <-
    qlPred_0.3.f$errsum.samp[[i]][5] - qlPred_0.3.f$errsum.samp[[i]][2]
  errsum.iqrs.3.f[i,2] <-
    qlPred_0.3.f$errsum.nsamp[[i]][5] - qlPred_0.3.f$errsum.nsamp[[i]][2]
}

# summary errors
errsum.meds.5.f <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.5.f <- matrix(data=0,ncol=2,nrow=N)

```

```

for (i in 1:N) {
  errsum.meds.5.f[i,1] <- qlPred_0.5.f$errsum.samp[[i]][3]
  errsum.meds.5.f[i,2] <- qlPred_0.5.f$errsum.nsamp[[i]][3]
  errsum.iqrs.5.f[i,1] <-
    qlPred_0.5.f$errsum.samp[[i]][5] - qlPred_0.5.f$errsum.samp[[i]][2]
  errsum.iqrs.5.f[i,2] <-
    qlPred_0.5.f$errsum.nsamp[[i]][5] - qlPred_0.5.f$errsum.nsamp[[i]][2]
}

# summary errors
errsum.meds.7.f <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.7.f <- matrix(data=0,ncol=2,nrow=N)
for (i in 1:N) {
  errsum.meds.7.f[i,1] <- qlPred_0.7.f$errsum.samp[[i]][3]
  errsum.meds.7.f[i,2] <- qlPred_0.7.f$errsum.nsamp[[i]][3]
  errsum.iqrs.7.f[i,1] <-
    qlPred_0.7.f$errsum.samp[[i]][5] - qlPred_0.7.f$errsum.samp[[i]][2]
  errsum.iqrs.7.f[i,2] <-
    qlPred_0.7.f$errsum.nsamp[[i]][5] - qlPred_0.7.f$errsum.nsamp[[i]][2]
}

# summary errors
errsum.meds.9.f <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.9.f <- matrix(data=0,ncol=2,nrow=N)
for (i in 1:N) {
  errsum.meds.9.f[i,1] <- qlPred_0.9.f$errsum.samp[[i]][3]
  errsum.meds.9.f[i,2] <- qlPred_0.9.f$errsum.nsamp[[i]][3]
  errsum.iqrs.9.f[i,1] <-
    qlPred_0.9.f$errsum.samp[[i]][5] - qlPred_0.9.f$errsum.samp[[i]][2]
  errsum.iqrs.9.f[i,2] <-
    qlPred_0.9.f$errsum.nsamp[[i]][5] - qlPred_0.9.f$errsum.nsamp[[i]][2]
}

# male
adult <- FALSE
smooth <- FALSE
N <- 50
for (S in seq(0.1,0.9,0.2)) {
  assign(paste("qlPred_",S,".m",sep=""))
  ,svdMod(qilogit.m,Q1.m,N,S,10,TRUE,adult,smooth,C))
}

## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "10% sample fraction"
## [1] "4 components"
## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "30% sample fraction"
## [1] "4 components"

```

```

## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "50% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "70% sample fraction"
## [1] "4 components"
##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "50 iterations"
## [1] "90% sample fraction"
## [1] "4 components"

# ... this could be more elegant as a list or by usign 'assign' like above, but ...
# summary errors
errsum.meds.1.m <- matrix(data=0, ncol=2, nrow=N)
errsum.iqrss.1.m <- matrix(data=0, ncol=2, nrow=N)
for (i in 1:N) {
  errsum.meds.1.m[i,1] <- qlPred_0.1.m$errsum.samp[[i]][3]
  errsum.meds.1.m[i,2] <- qlPred_0.1.m$errsum.nsamp[[i]][3]
  errsum.iqrss.1.m[i,1] <-
    qlPred_0.1.m$errsum.samp[[i]][5] - qlPred_0.1.m$errsum.samp[[i]][2]
  errsum.iqrss.1.m[i,2] <-
    qlPred_0.1.m$errsum.nsamp[[i]][5] - qlPred_0.1.m$errsum.nsamp[[i]][2]
}
# summary errors
errsum.meds.3.m <- matrix(data=0, ncol=2, nrow=N)
errsum.iqrss.3.m <- matrix(data=0, ncol=2, nrow=N)
for (i in 1:N) {
  errsum.meds.3.m[i,1] <- qlPred_0.3.m$errsum.samp[[i]][3]
  errsum.meds.3.m[i,2] <- qlPred_0.3.m$errsum.nsamp[[i]][3]
  errsum.iqrss.3.m[i,1] <-
    qlPred_0.3.m$errsum.samp[[i]][5] - qlPred_0.3.m$errsum.samp[[i]][2]
  errsum.iqrss.3.m[i,2] <-
    qlPred_0.3.m$errsum.nsamp[[i]][5] - qlPred_0.3.m$errsum.nsamp[[i]][2]
}

# summary errors
errsum.meds.5.m <- matrix(data=0, ncol=2, nrow=N)
errsum.iqrss.5.m <- matrix(data=0, ncol=2, nrow=N)
for (i in 1:N) {
  errsum.meds.5.m[i,1] <- qlPred_0.5.m$errsum.samp[[i]][3]
  errsum.meds.5.m[i,2] <- qlPred_0.5.m$errsum.nsamp[[i]][3]
  errsum.iqrss.5.m[i,1] <-
    qlPred_0.5.m$errsum.samp[[i]][5] - qlPred_0.5.m$errsum.samp[[i]][2]
  errsum.iqrss.5.m[i,2] <-
    qlPred_0.5.m$errsum.nsamp[[i]][5] - qlPred_0.5.m$errsum.nsamp[[i]][2]
}

```

```

# summary errors
errsum.meds.7.m <- matrix(data=0, ncol=2, nrow=N)
errsum.iqrs.7.m <- matrix(data=0, ncol=2, nrow=N)
for (i in 1:N) {
  errsum.meds.7.m[i,1] <- qlPred_0.7.m$errsum.samp[[i]][3]
  errsum.meds.7.m[i,2] <- qlPred_0.7.m$errsum.nsamp[[i]][3]
  errsum.iqrs.7.m[i,1] <-
    qlPred_0.7.m$errsum.samp[[i]][5] - qlPred_0.7.m$errsum.samp[[i]][2]
  errsum.iqrs.7.m[i,2] <-
    qlPred_0.7.m$errsum.nsamp[[i]][5] - qlPred_0.7.m$errsum.nsamp[[i]][2]
}

# summary errors
errsum.meds.9.m <- matrix(data=0, ncol=2, nrow=N)
errsum.iqrs.9.m <- matrix(data=0, ncol=2, nrow=N)
for (i in 1:N) {
  errsum.meds.9.m[i,1] <- qlPred_0.9.m$errsum.samp[[i]][3]
  errsum.meds.9.m[i,2] <- qlPred_0.9.m$errsum.nsamp[[i]][3]
  errsum.iqrs.9.m[i,1] <-
    qlPred_0.9.m$errsum.samp[[i]][5] - qlPred_0.9.m$errsum.samp[[i]][2]
  errsum.iqrs.9.m[i,2] <-
    qlPred_0.9.m$errsum.nsamp[[i]][5] - qlPred_0.9.m$errsum.nsamp[[i]][2]
}

```

5 Plotting

Most plotting is done using *ggplot*. First load the necessary packages. The plots are not generated in the same order of appearance as the paper.

Plot the basic age \times age relationships among $5q_x$ for all ages and both sexes and save as (very large) PDF files.

```

# age relationships

# female
pdf(file="../figures/femaleLogit(q)AgeScatterplots.pdf")
qln.f <- as.matrix(q1logit.f)
nr <- nrow(qln.f)
for (i in seq(0,100,5)) {
  for (j in seq(i,100,5)) {
    plot(qln.f[(j+1),] ~ qln.f[(i+1),]
        ,cex=0.1,xlab=paste("Age ",i,sep="")
        ,ylab=paste("Age ",j,sep="")
        ,xlim=c(-12,0),ylim=c(-12,0)
        ,main="Logit(q) by Logit(q) in 5-year Age Groups")
  }
}
dev.off()

## pdf
## 2
# male
pdf(file="../figures/maleLogit(q)AgeScatterplots.pdf")

```

```

qln.m <- as.matrix(q1logit.m)
nr <- nrow(qln.m)
for (i in seq(0,100,5)) {
  for (j in seq(i,100,5)) {
    plot(qln.m[(j+1),] ~ qln.m[(i+1),]
        ,cex=0.1,xlab=paste("Age ",i,sep="")
        ,ylab=paste("Age ",j,sep="")
        ,xlim=c(-12,0),ylim=c(-12,0)
        ,main="Logit(q) by Logit(q) in 5-year Age Groups")
  }
}
dev.off()

## pdf
## 2
rm(list=c("nr","i","j"))

```

Plot the SVD-comp and Log Quad error distributions by sex and age using 25%, 50%, and 75% quantiles and whiskers to 10% and 90%.

```

# the predicted values from both models are stored in comps.child and comps.adult;
# we are making comparisons using the child-only predictions
# the q5.x values are straight from HMD
# errors on natural scale
# female
errors.comp.f <- q5.f - comps.child$q5p.f
errors.lq.f <- q5.f - comps.child$q5.lq.f
# male
errors.comp.m <- q5.m - comps.child$q5p.m
errors.lq.m <- q5.m - comps.child$q5.lq.m

# reshape the data

# female
ecf <- melt(as.matrix(errors.comp.f))
elf <- melt(as.matrix(errors.lq.f))
ecf <- cbind(ecf[,c(1,3)],rep("SVD-Comp",nrow(ecf)))
colnames(ecf) <- c("Age (years)","Error","Model")
elf <- cbind(elf[,c(1,3)],rep("Log-Quad",nrow(elf)))
colnames(elf) <- c("Age (years)","Error","Model")
ef <- rbind(ecf,elf)

# male
ecm <- melt(as.matrix(errors.comp.m))
elm <- melt(as.matrix(errors.lq.m))
ecm <- cbind(ecm[,c(1,3)],rep("SVD-Comp",nrow(ecm)))
colnames(ecm) <- c("Age (years)","Error","Model")
elm <- cbind(elm[,c(1,3)],rep("Log-Quad",nrow(elm)))
colnames(elm) <- c("Age (years)","Error","Model")
em <- rbind(ecm,elm)

efn <- cbind(em,"Female")
colnames(em) <- c("Age (years)","Error","Model","Sex")
emn <- cbind(em,"Male")

```

```

colnames(emn) <- c("Age (years)", "Error", "Model", "Sex")

e <- rbind(efn,emn)

e.sum <- ddply(e,.(Sex, `Age (years)`), Model),
  summarize,
  ymin = quantile(Error,.1),
  ymax = quantile(Error,.9),
  middle = median(Error),
  lower = quantile(Error,0.25),
  upper = quantile(Error,0.75)
)

s.names <- list(
  'S#1' = expression(bold("Female")),
  'S#2' = expression(bold("Male"))
)
# s.names

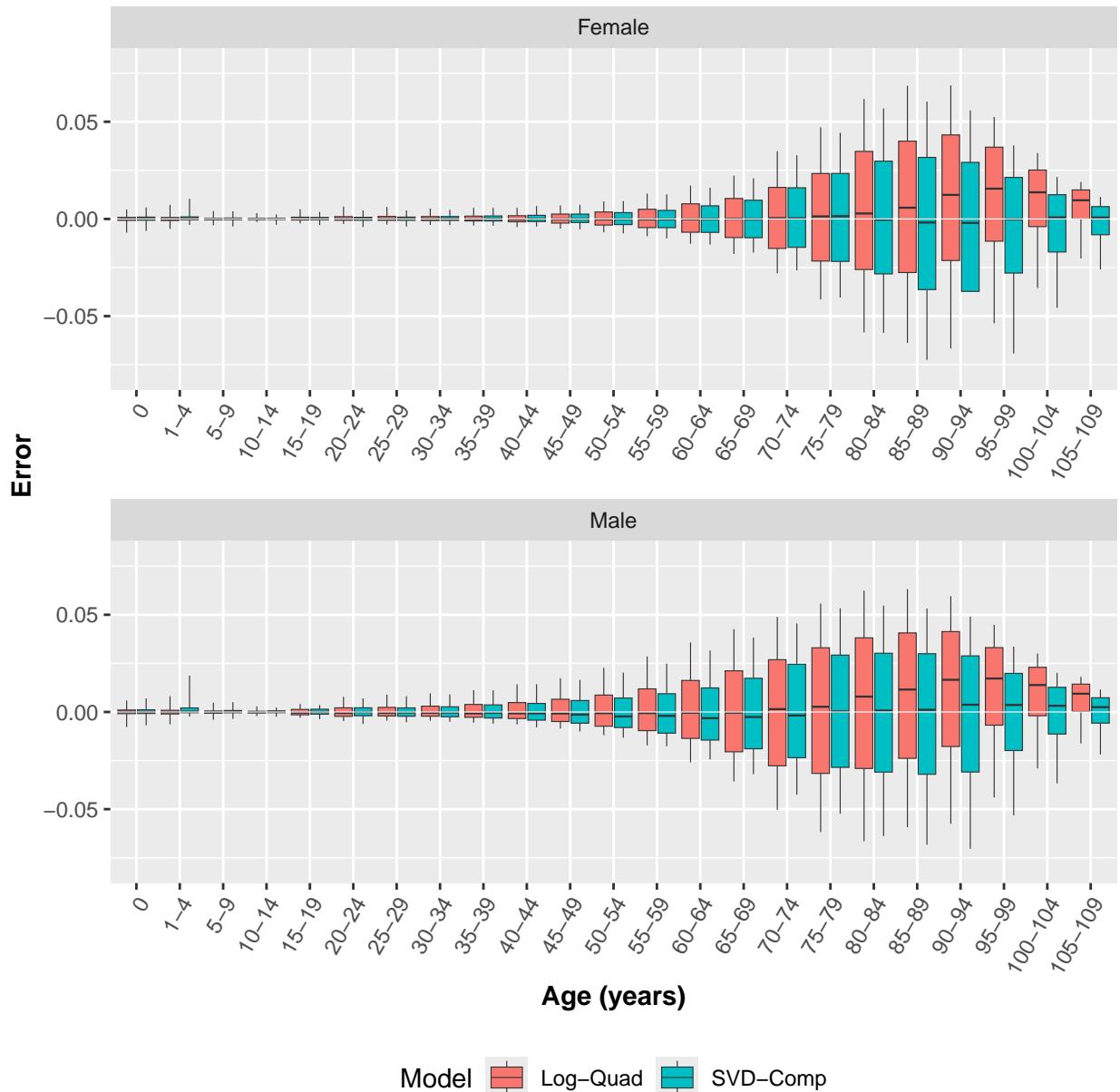
s.labeller <- function(variable,value){
  return(s.names[value])
}

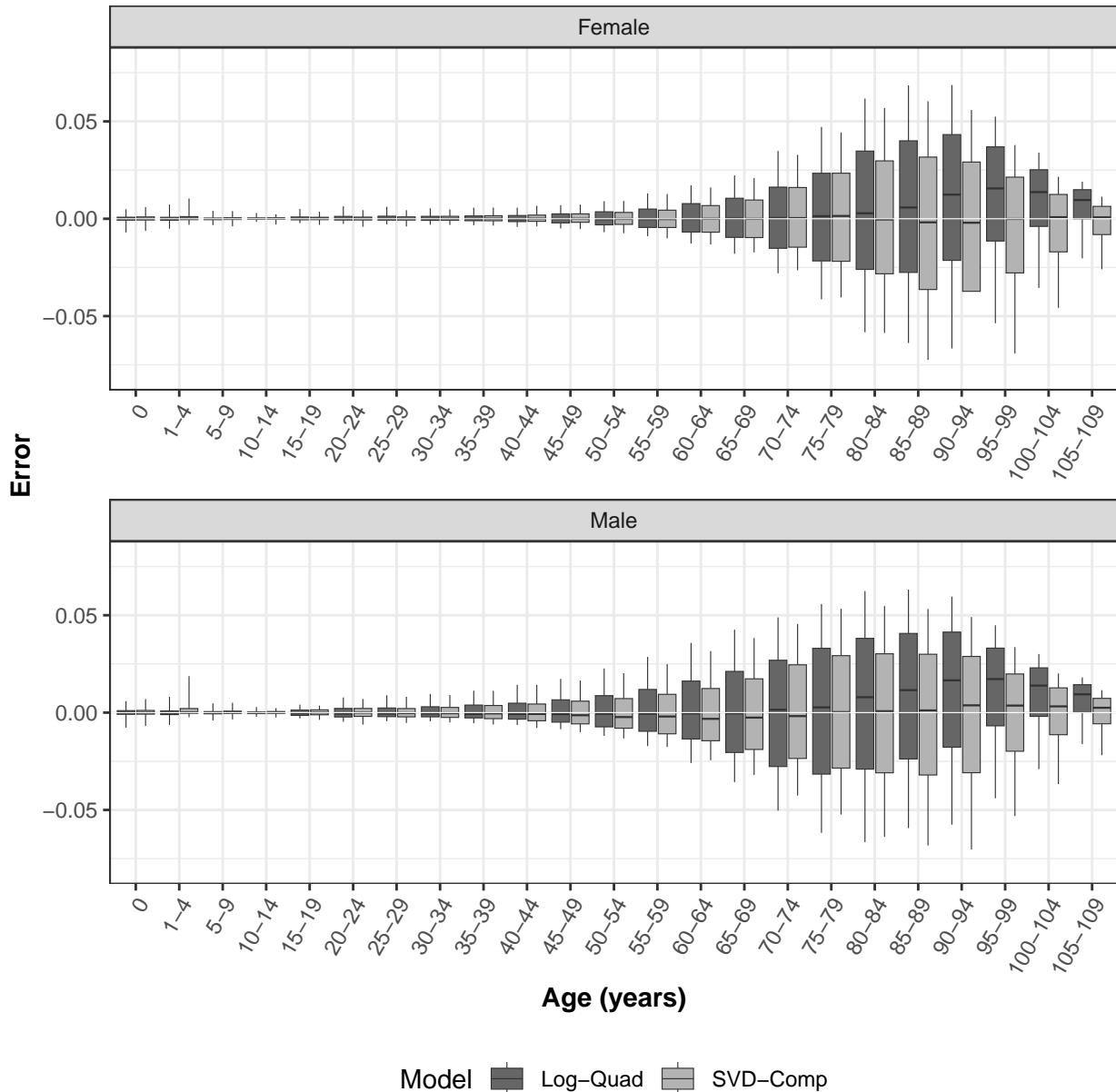
ggplot(data = e.sum, aes(x=`Age (years)`)) +
  geom_boxplot(aes(fill=Model,ymin = ymin,ymax = ymax
                  ,middle = middle,upper = upper
                  ,lower= lower),stat='identity',size=0.2) +
  scale_y_continuous(limits = c(-0.08,0.08)) +
  # theme(legend.justification=c(1,0), legend.position=c(.22,0.02)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  geom_hline(yintercept=0,colour="white",lwd=.2) +
  labs(y = expression(bold("Error")), x = expression(bold("Age (years)"))) +
  facet_wrap(~Sex,ncol=1,scales="free")
  # facet_wrap(~Sex,ncol=1,scales="free", labeller=s.labeller)
ggsave("../figures/fig3.pdf",width=6.5,height=6.5,units=c("in"))

# grayscale
ggplot(data = e.sum, aes(x=`Age (years)`)) +
  geom_boxplot(aes(fill=Model,ymin = ymin,ymax = ymax
                  ,middle = middle,upper = upper
                  ,lower= lower),stat='identity',size=0.2) +
  scale_y_continuous(limits = c(-0.08,0.08)) +
  # theme(legend.justification=c(1,0), legend.position=c(.22,0.02)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),) +
  geom_hline(yintercept=0,colour="white",lwd=.2) +
  labs(y = expression(bold("Error")), x = expression(bold("Age (years)"))) +
  facet_wrap(~Sex,ncol=1,scales="free") +
  # facet_wrap(~Sex,ncol=1,scales="free", labeller=s.labeller) +
  scale_fill_grey(start = 0.4, end = .7)
ggsave("../figures/fig3-BW.pdf",width=6.5,height=6.5,units=c("in"))

```

```
# clean up
rm(list=c("e.sum", "e", "efn", "emn", "em", "elm", "ecm", "ef", "elf", "ecf"))
```





Plot the example mortality schedules with data and predictions from SVD-Comp. Use Austria 1990 as a low mortality example and Sweden 1751 as a high mortality example.

```
# calculate the total absolute error per life table
tot.abs.err.f <- colSums(abs(errors.comp.f))
tot.abs.err.m <- colSums(abs(errors.comp.m))
# using that metric, look for best fitting female and male LTs
best.f <- which(tot.abs.err.f==min(tot.abs.err.f))
best.f

## female.IRL.2004
##          2870
best.m <- which(tot.abs.err.m==min(tot.abs.err.m))
best.m

## male.FRACNP.1970 male.FRATNP.1970
```

```

##          1662          1869
# looks like East Germany 2006 for females and Denmark 2010 for males

# find the earliest Swedish LT
# cat(colnames(q1.f),sep="\n") # lists all the female LTs; use with caution - very long
swe.f.1751 <- which(colnames(q1.f)=="female.SWE.1751")
swe.f.1751

## [1] 4433
swe.m.1751 <- which(colnames(q1.m)=="male.SWE.1751")
swe.m.1751

## [1] 4433
# looks like Sweden 1751 is number 4,140; double check
cat(colnames(q1.f)[(swe.f.1751-3):(swe.f.1751+3)],sep="\n")

## female.SVN.2017
## female.SVN.2018
## female.SVN.2019
## female.SWE.1751
## female.SWE.1752
## female.SWE.1753
## female.SWE.1754

# have a quick look at both the 'best' fitting LTs and choose one
i <- best.f
plot(q1logit.f[,i],)
points(q1logit.m[,i],col="red")
points(mod.1_0.f$recon.samp$s1[,i],type="l")
points(mod.1_0.m$recon.samp$s1[,i],type="l",col="red")
i <- best.m
plot(q1logit.f[,i],)
points(q1logit.m[,i],col="red")
points(mod.1_0.f$recon.samp$s1[,i],type="l")
points(mod.1_0.m$recon.samp$s1[,i],type="l",col="red")
# don't like either of them much as pretty examples

# Austria 1990 is nice example of low mortality - will use that for low mortality example
aut.f.1990 <- which(colnames(q1.f)=="female.AUT.1990")
aut.f.1990

## [1] 145
aut.m.1990 <- which(colnames(q1.m)=="male.AUT.1990")
aut.m.1990

## [1] 145
i <- aut.f.1990
plot(q1logit.f[,i],)
points(q1logit.m[,i],col="red")
points(mod.1_0.f$recon.samp$s1[,i],type="l")
points(mod.1_0.m$recon.samp$s1[,i],type="l",col="red")

# data - use Sweden 1751 and Austria 1990:
i.low <- aut.f.1990

```

```

i.high <- swe.f.1751
tmp.1.m <- cbind(rep("Male",110),rownames(q1logit.m)
                  ,rep("Sweden, 1751",110),"Data",q1logit.m[,i.high])
tmp.1.f <- cbind(rep("Female",110),rownames(q1logit.m)
                  ,rep("Sweden, 1751",110),"Data",q1logit.f[,i.high])
tmp.2.m <- cbind(rep("Male",110),rownames(q1logit.m)
                  ,rep("Austria, 1990",110),"Data",q1logit.m[,i.low])
tmp.2.f <- cbind(rep("Female",110),rownames(q1logit.m)
                  ,rep("Austria, 1990",110),"Data",q1logit.f[,i.low])
data.fig1 <- rbind(tmp.1.m,tmp.1.f,tmp.2.m,tmp.2.f)

data.fig1.df <- data.frame(
  Sex = as.character(data.fig1[,1]),
  Age = as.numeric(data.fig1[,2]),
  LT = as.character(data.fig1[,3]),
  Type = as.character(data.fig1[,4]),
  Value = as.numeric(data.fig1[,5])
)

# predictions
m.1.p <- mod.1_0.m$recon.samp$s1[,i.high]
f.1.p <- mod.1_0.f$recon.samp$s1[,i.high]
m.2.p <- mod.1_0.m$recon.samp$s1[,i.low]
f.2.p <- mod.1_0.f$recon.samp$s1[,i.low]
tmp.1.m <- cbind(rep("Male",110),rownames(q1logit.m)
                  ,rep("Sweden, 1751",110),"Predicted",m.1.p)
tmp.1.f <- cbind(rep("Female",110),rownames(q1logit.m)
                  ,rep("Sweden, 1751",110),"Predicted",f.1.p)
tmp.2.m <- cbind(rep("Male",110),rownames(q1logit.m)
                  ,rep("Austria, 1990",110),"Predicted",m.2.p)
tmp.2.f <- cbind(rep("Female",110),rownames(q1logit.m)
                  ,rep("Austria, 1990",110),"Predicted",f.2.p)
pred.fig1 <- rbind(tmp.1.m,tmp.1.f,tmp.2.m,tmp.2.f)

pred.fig1.df <- data.frame(
  Sex = as.character(pred.fig1[,1]),
  Age = as.numeric(pred.fig1[,2]),
  LT = as.character(pred.fig1[,3]),
  Type = as.character(pred.fig1[,4]),
  Value = as.numeric(pred.fig1[,5])
)

# Plot
ggplot(data = data.fig1.df, aes(x=Age, y=Value
                                , group=interaction(Sex,LT), colour=Sex, shape=LT)) +
  geom_point(size=1.5) +
  geom_line(data = pred.fig1.df, aes(x=Age
                                      , y=Value, group=interaction(Sex,LT)
                                      , colour=Sex), size=1) +
  scale_x_continuous(breaks=seq(0,110,5)) +
  labs(y = expression(''[bold(1)]*bolditalic('q')[bolditalic(x)]*bold(' (logit scale)'))
       , x = expression(bold("Age (years)"))) +
  # theme(legend.justification=c(1,0), legend.position=c(0.98,0.02)) +

```

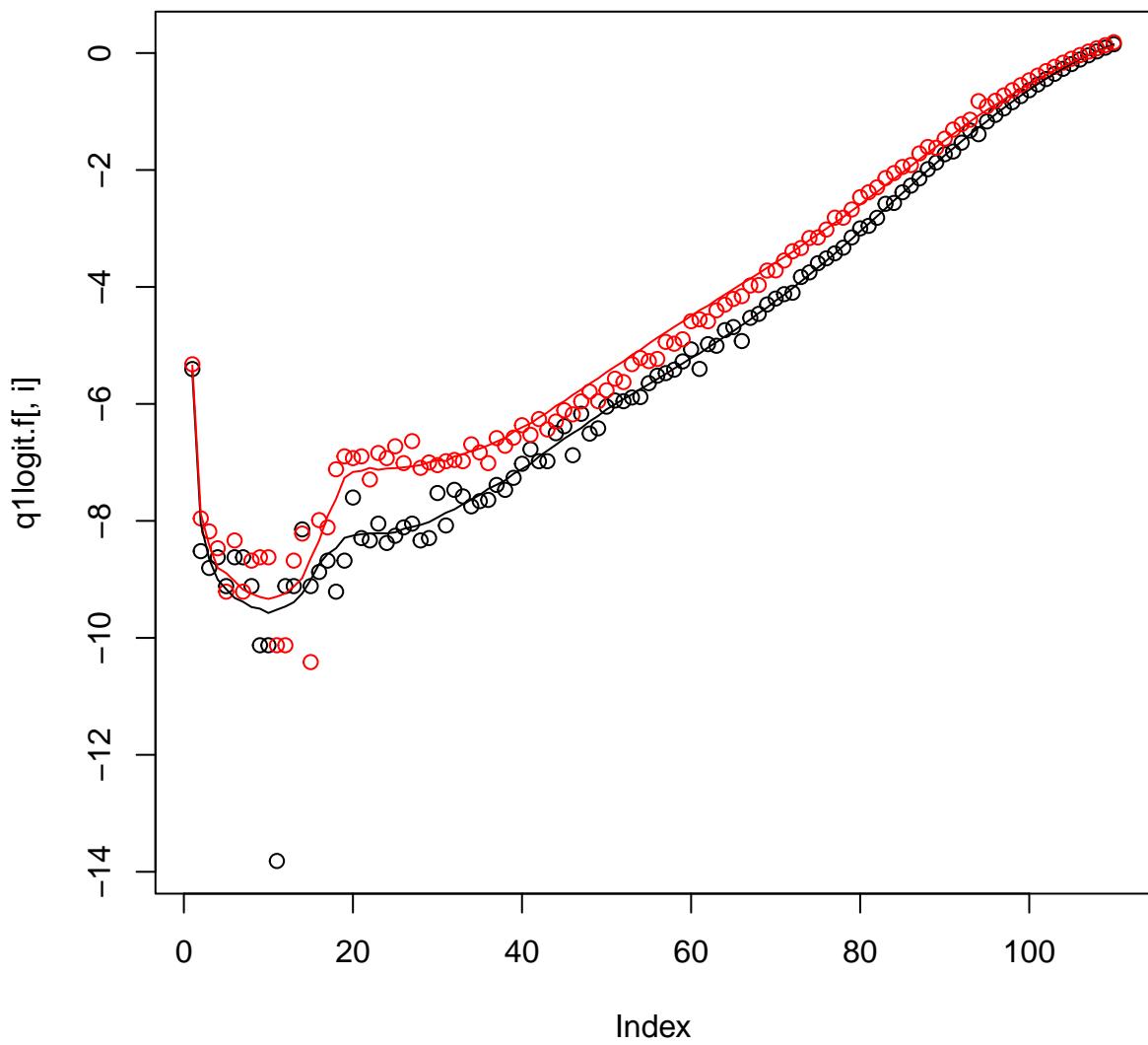
```

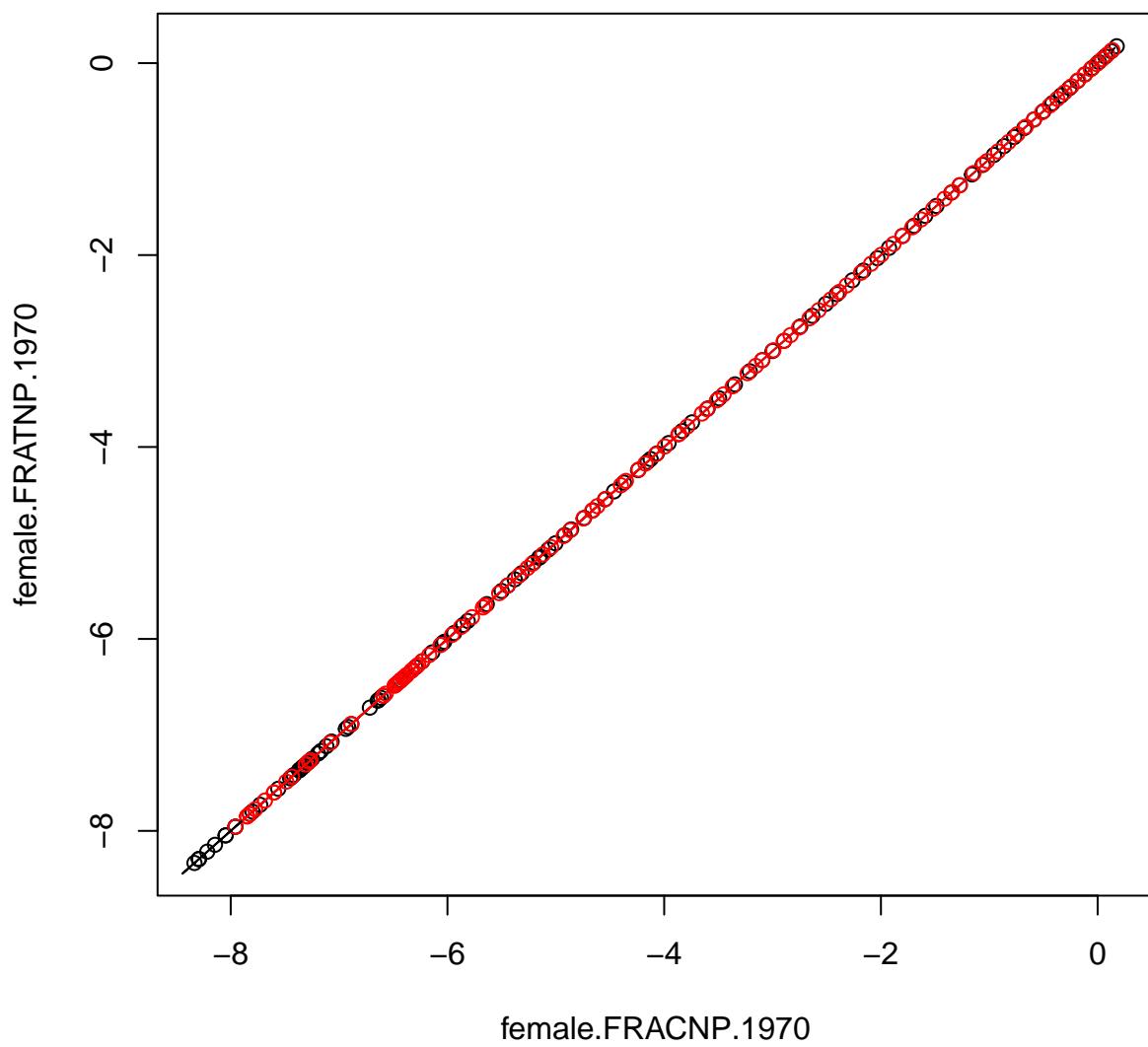
theme(legend.position="bottom", legend.box = "horizontal") +
  scale_shape_discrete(name = "Life Table")
ggsave("../figures/fig1.pdf",width=6.5,height=6.5,units=c("in"))

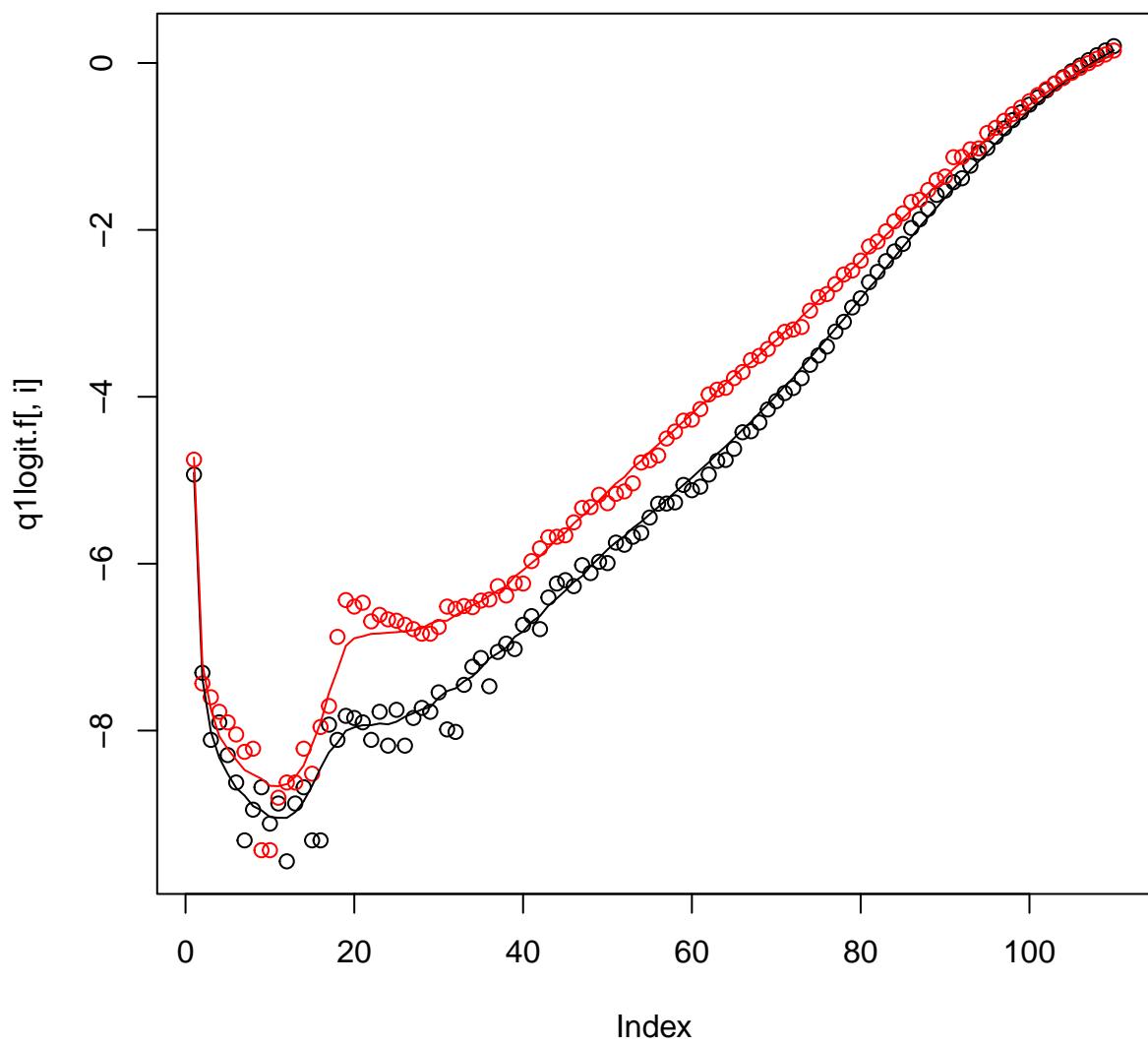
# grayscale
ggplot(data = data.fig1.df, aes(x=Age, y=Value
                                 , group=interaction(Sex,LT), colour=Sex, shape=LT)) +
  geom_point(size=1.5) +
  geom_line(data = pred.fig1.df, aes(x=Age
                                       , y=Value, group=interaction(Sex,LT)
                                       , colour=Sex), size=1) +
  scale_x_continuous(breaks=seq(0,110,5)) +
  labs(y = expression('' [bold(1)]*bolditalic('q') [bolditalic(x)]*bold(' (logit scale)'))
       , x = expression(bold("Age (years)"))) +
  # theme(legend.justification=c(1,0), legend.position=c(0.98,0.02)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  scale_shape_discrete(name = "Life Table") +
  scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig1-BW.pdf",width=6.5,height=6.5,units=c("in"))

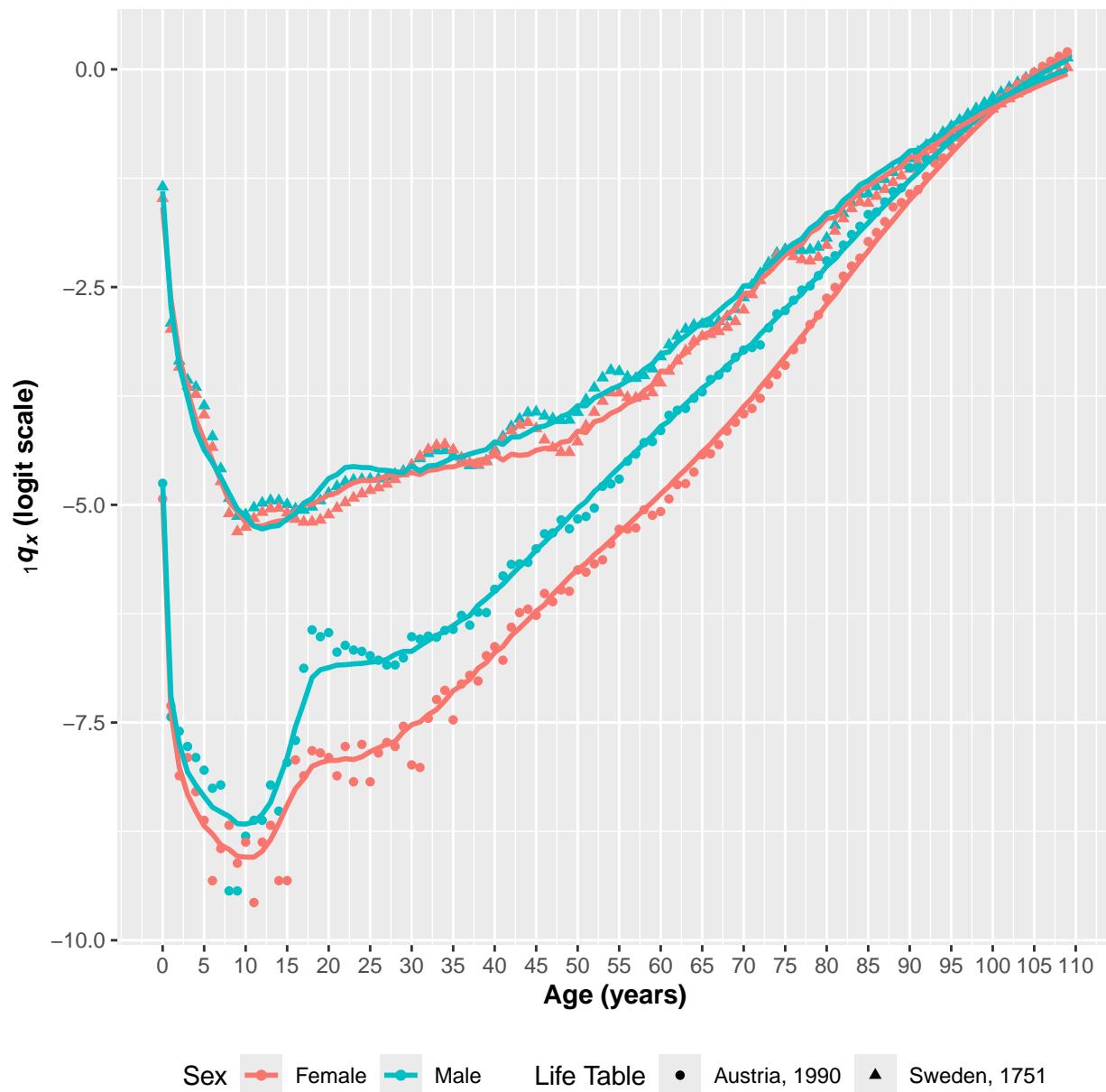
# clean up
rm(list=c("pred.fig1.df","pred.fig1","tmp.2.f","tmp.2.m","tmp.1.f","tmp.1.m"
         ,"f.2.p","m.2.p","f.1.p","m.1.p","data.fig1.df","data.fig1","i.low"
         ,"i.high","tot.abs.err.f","tot.abs.err.m"))

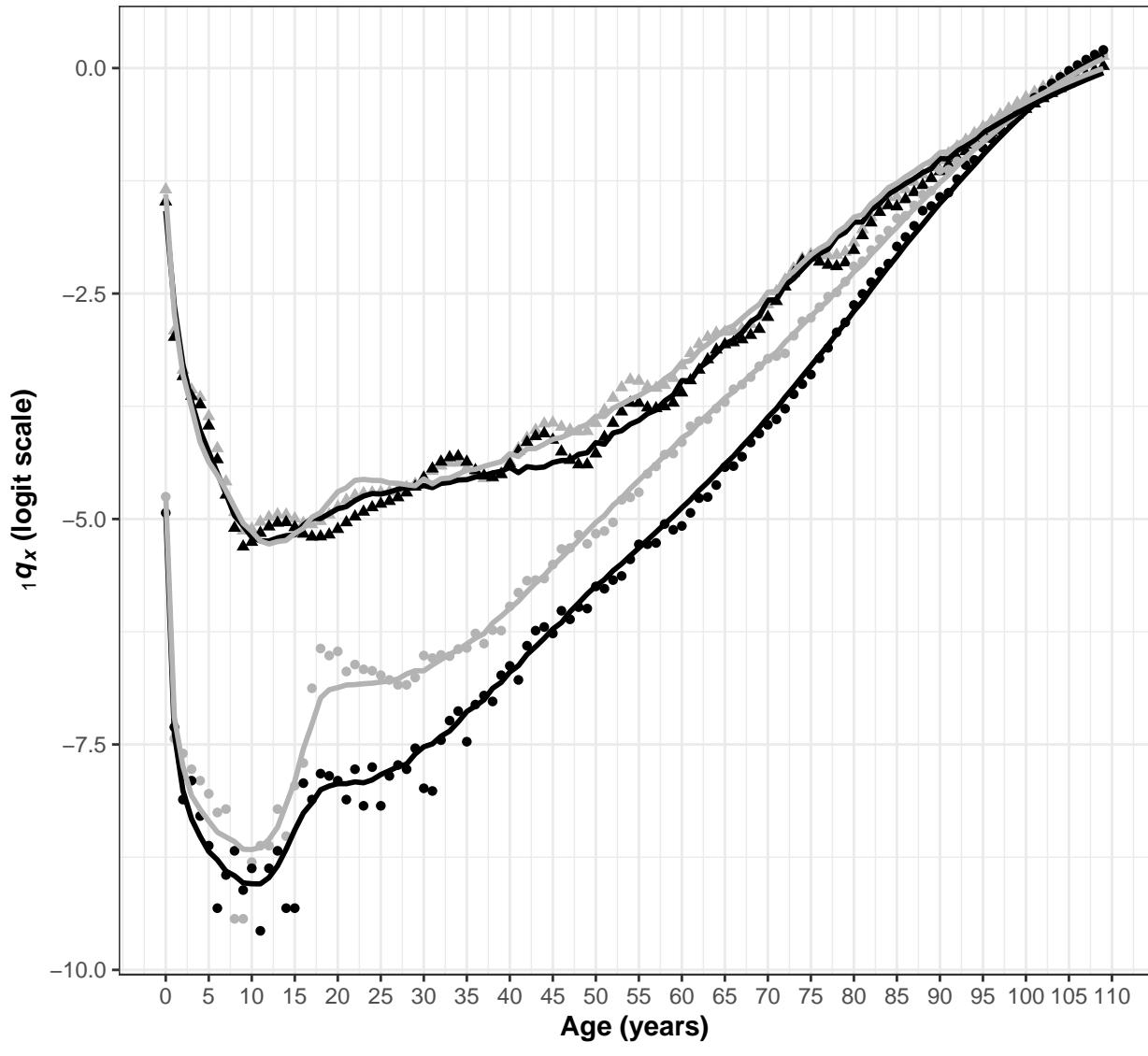
```











Sex ● Female ● Male Life Table ● Austria, 1990 ▲ Sweden, 1751

Plot the scaled left singular vectors of the SVD decompositions of logit-transformed $1q_x$.

```
# svds
svd.m <- mod.1_0.m$svd$s1
svd.f <- mod.1_0.f$svd$s1

# scaled us
u1.m <- cbind(rep("Male",110),rownames(q1logit.m)
               ,rep("u1",110),svd.m$d[1]*svd.m$u[,1])
u2.m <- cbind(rep("Male",110),rownames(q1logit.m)
               ,rep("u2",110),svd.m$d[2]*svd.m$u[,2])
u3.m <- cbind(rep("Male",110),rownames(q1logit.m)
               ,rep("u3",110),-1*svd.m$d[3]*svd.m$u[,3])
u4.m <- cbind(rep("Male",110),rownames(q1logit.m)
               ,rep("u4",110),svd.m$d[4]*svd.m$u[,4])
```

```

u1.f <- cbind(rep("Female",110),rownames(q1logit.m)
               ,rep("u1",110),svd.f$d[1]*svd.f$u[,1])
u2.f <- cbind(rep("Female",110),rownames(q1logit.m)
               ,rep("u2",110),svd.f$d[2]*svd.f$u[,2])
u3.f <- cbind(rep("Female",110),rownames(q1logit.m)
               ,rep("u3",110),svd.f$d[3]*svd.f$u[,3])
u4.f <- cbind(rep("Female",110),rownames(q1logit.m)
               ,rep("u4",110),svd.f$d[4]*svd.f$u[,4])

us <- rbind(u1.m,u2.m,u3.m,u4.m,u1.f,u2.f,u3.f,u4.f)

us.df <- data.frame(
  Sex = as.character(us[,1]),
  Age = as.numeric(us[,2]),
  U = as.character(us[,3]),
  Value = as.numeric(us[,4])
)
save(file="../RData/us.RData",compress=TRUE,list=c("us.df"))
# write.csv(us.df,file="../tables/us.csv")
# str(us.df)

# smooth svds
# svds
svd.sm.m <- mod.1_0.sm.m$svd.sm$s1
svd.sm.f <- mod.1_0.sm.f$svd.sm$s1

# us
u1.sm.m <- cbind(rep("Male",110),rownames(q1logit.m)
                   ,rep("u1",110),svd.sm.m$d[1]*svd.sm.m$u[,1])
u2.sm.m <- cbind(rep("Male",110),rownames(q1logit.m)
                   ,rep("u2",110),svd.sm.m$d[2]*svd.sm.m$u[,2])
u3.sm.m <- cbind(rep("Male",110),rownames(q1logit.m)
                   ,rep("u3",110),-1*svd.sm.m$d[3]*svd.sm.m$u[,3])
u4.sm.m <- cbind(rep("Male",110),rownames(q1logit.m)
                   ,rep("u4",110),svd.sm.m$d[4]*svd.sm.m$u[,4])

u1.sm.f <- cbind(rep("Female",110),rownames(q1logit.m)
                   ,rep("u1",110),svd.sm.f$d[1]*svd.sm.f$u[,1])
u2.sm.f <- cbind(rep("Female",110),rownames(q1logit.m)
                   ,rep("u2",110),svd.sm.f$d[2]*svd.sm.f$u[,2])
u3.sm.f <- cbind(rep("Female",110),rownames(q1logit.m)
                   ,rep("u3",110),svd.sm.f$d[3]*svd.sm.f$u[,3])
u4.sm.f <- cbind(rep("Female",110),rownames(q1logit.m)
                   ,rep("u4",110),svd.sm.f$d[4]*svd.sm.f$u[,4])

us.sm <- rbind(u1.sm.m,u2.sm.m,u3.sm.m
                 ,u4.sm.m,u1.sm.f,u2.sm.f,u3.sm.f,u4.sm.f)

us.sm.df <- data.frame(
  Sex = as.character(us.sm[,1]),
  Age = as.numeric(us.sm[,2]),
  U = as.character(us.sm[,3]),
  Value = as.numeric(us.sm[,4])
)

```

```

)
save(file="../RData/us-smooth.RData",compress=TRUE,list=c("us.sm.df"))
# write.csv(us.sm.df,file="../tables/us-smooth.csv")
# str(us.sm.df)

# Plot

# data for horizontal lines at 0
hlines <- data.frame(
  U = as.character(c("u1","u2","u3","u4")),
  y = as.numeric(c(NA,0,0,0))
)

u.names <- list(
  'U#1' = expression(italic('s')[1]*bold('u')[1]),
  'U#2' = expression(italic('s')[2]*bold('u')[2]),
  'U#3' = expression(italic('s')[3]*bold('u')[3]),
  'U#4' = expression(italic('s')[4]*bold('u')[4])
)
# u.names

u.labeller <- function(variable,value){
  return(u.names[value])
}

# Plot
ggplot(data = us.df, aes(x=Age, y=Value, group=Sex, colour=Sex)) +
  geom_line() +
  geom_line(data = us.sm.df, aes(x=Age, y=Value), size=1) +
  geom_hline(data = hlines, aes(yintercept = y)) +
  scale_x_continuous(breaks=seq(0,110,10)) +
  scale_y_continuous(labels=function(x) format(x, big.mark = ",",
                                              decimal.mark = ".", scientific = FALSE)) +
  facet_wrap(~U, scales="free") +
  # facet_wrap(~U, scales="free", labeller=u.labeller) +
  labs(y = expression(bold("Scaled LSV Values (logit scale)")),
       x = expression(bold("Age (years)"))) +
  # theme(legend.justification=c(1,0), legend.position=c(0.99,0.56))
  theme(legend.position="bottom", legend.box = "horizontal")
ggsave("../figures/fig2.pdf",width=6.5,height=6.5,units=c("in"))

# grayscale
ggplot(data = us.df, aes(x=Age, y=Value, group=Sex, colour=Sex)) +
  geom_line() +
  geom_line(data = us.sm.df, aes(x=Age, y=Value), size=1) +
  geom_hline(data = hlines, aes(yintercept = y)) +
  scale_x_continuous(breaks=seq(0,110,10)) +
  scale_y_continuous(labels=function(x) format(x, big.mark = ",",
                                              decimal.mark = ".", scientific = FALSE)) +
  facet_wrap(~U, scales="free") +
  # facet_wrap(~U, scales="free", labeller=u.labeller) +
  labs(y = expression(bold("Scaled LSV Values (logit scale)")),
       x = expression(bold("Age (years)")))

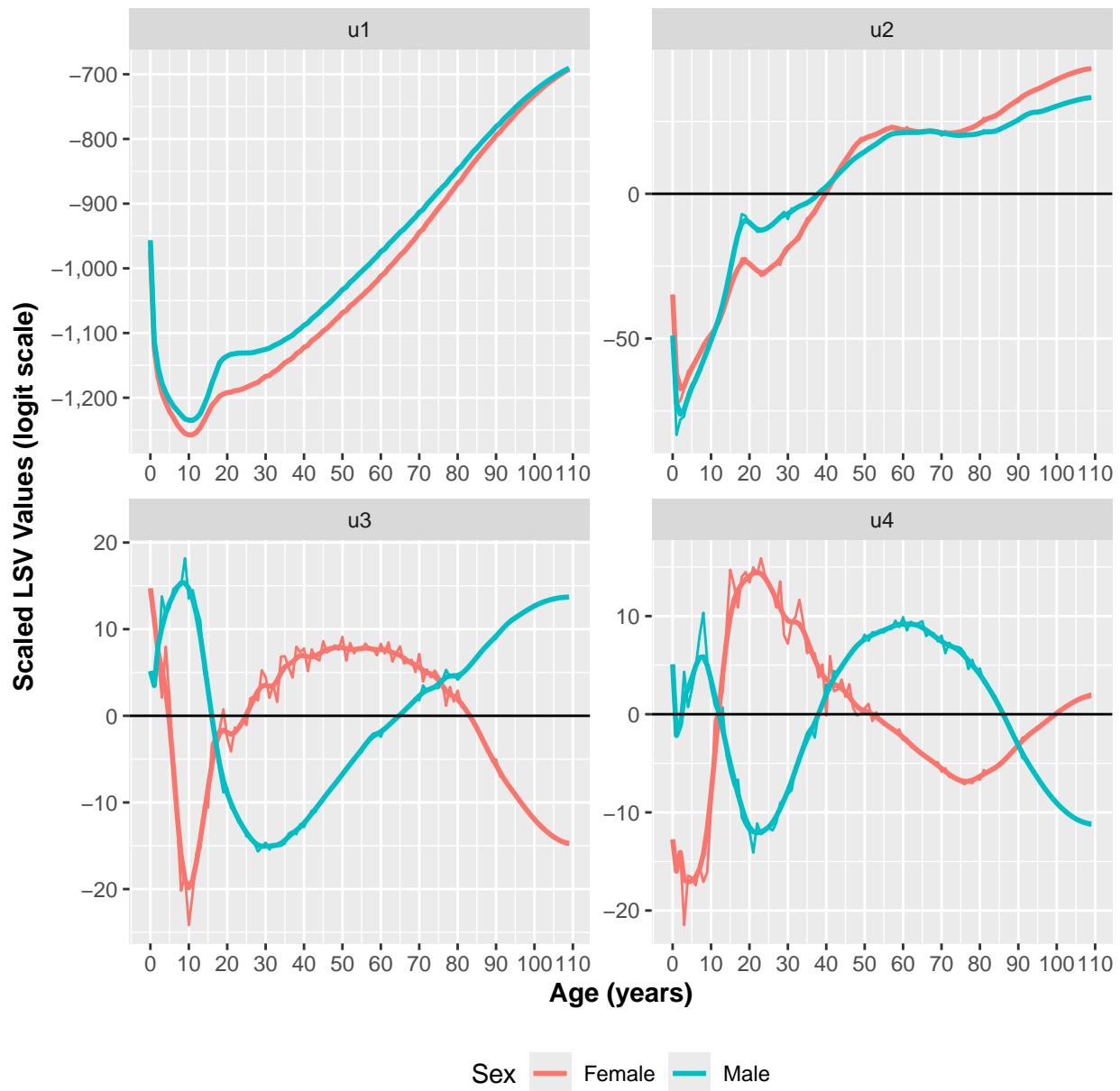
```

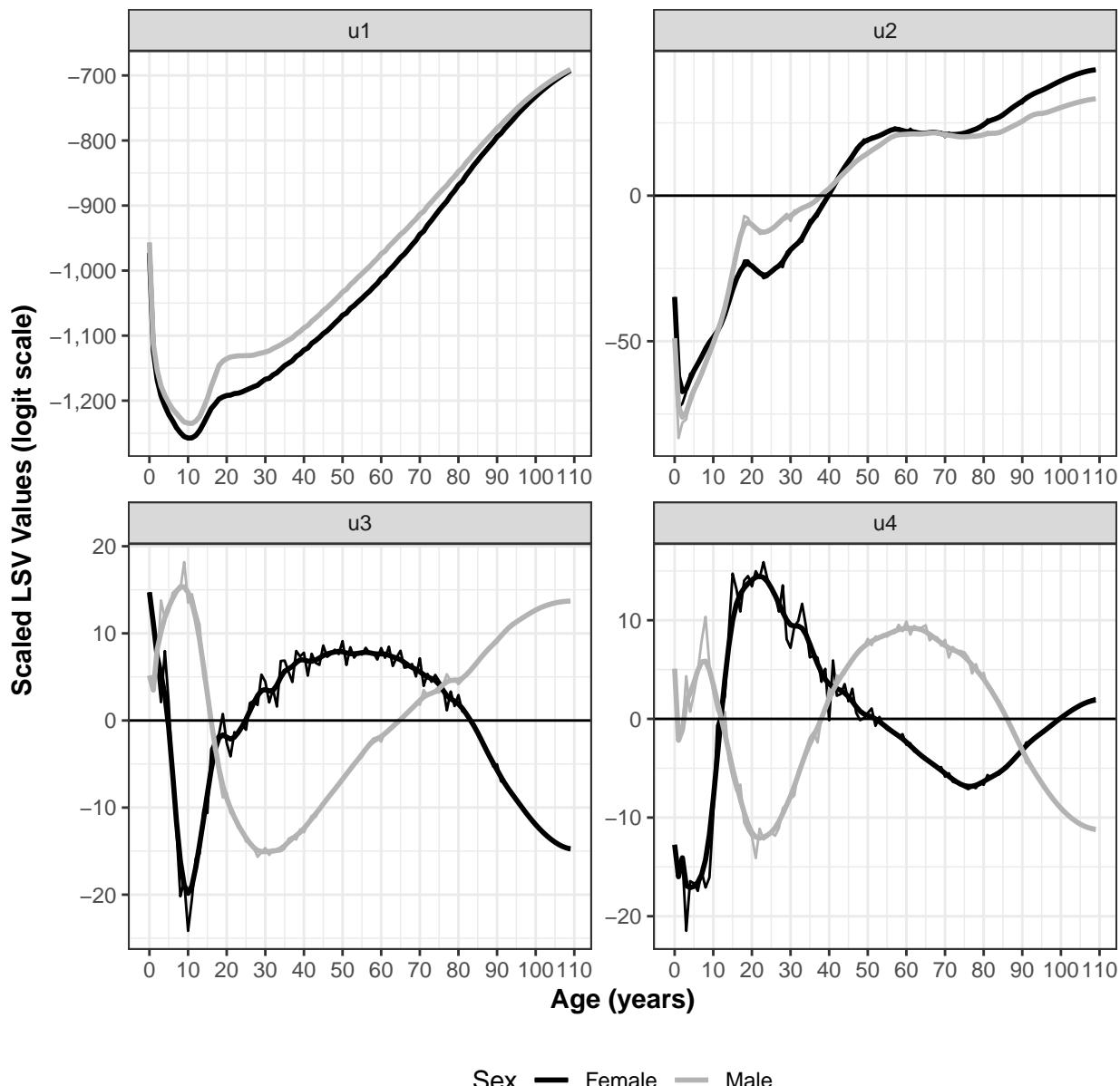
```

# theme(legend.justification=c(1,0), legend.position=c(0.99,0.56))
theme_bw() +
theme(legend.position="bottom", legend.box = "horizontal") +
scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig2-BW.pdf", width=6.5, height=6.5, units=c("in"))

# clean up
rm(list=c("u.labeller", "u.names", "hlines", "us.sm.df", "us.sm", "u4.sm.f",
         "u3.sm.f", "u2.sm.f", "u1.sm.f", "u4.sm.m", "u3.sm.m", "u2.sm.m", "u1.sm.m",
         "svd.sm.f", "svd.sm.m", "us.df", "us", "u4.f", "u3.f", "u2.f", "u1.f",
         "u4.m", "u3.m", "u2.m", "u1.m", "svd.f", "svd.m"))

```





Plot the single-year prediction error distributions from 50 50% samples.

```
# female
esf <- melt(error.age.f)
esf <- cbind(esf[,c(1,3)], "In", "Female")
colnames(esf) <- c("Age", "Error", "Sample", "Sex")

ensf <- melt(error.age.nsamp.f)
ensf <- cbind(ensf[,c(1,3)], "Out", "Female")
colnames(ensf) <- c("Age", "Error", "Sample", "Sex")
# rm(list=c("error.age.f", "error.age.nsamp.f"))

ef <- rbind(esf,ensf)
ef$Age <- factor(ef$Age)
rm(list=c("esf", "ensf"))
```

```

# male
esm <- melt(error.age.m)
esm <- cbind(esm[,c(1,3)], "In", "Male")
colnames(esm) <- c("Age", "Error", "Sample", "Sex")

ensm <- melt(error.age.nsamp.m)
ensm <- cbind(ensm[,c(1,3)], "Out", "Male")
colnames(ensm) <- c("Age", "Error", "Sample", "Sex")
# rm(list=c("error.age.m", "error.age.nsamp.m"))

em <- rbind(esm,ensm)
em$Age <- factor(em$Age)
rm(list=c("esm", "ensm"))

# combine male and female
eb <- rbind(em,ef)
rm(list=c("em", "ef"))

# order female first
eb[,4] <- factor(eb[,4], levels=c("Female", "Male"))
# str(eb)

e.sum <- ddply(eb,.(Sex, Age, Sample),
  summarize,
  ymin = quantile(Error,.1),
  ymax = quantile(Error,.9),
  middle = median>Error),
  lower = quantile>Error,.25),
  upper = quantile>Error,.75)
)

s.names <- list(
  'S#1' = expression(bold("Female")),
  'S#2' = expression(bold("Male"))
)
# s.names

s.labeller <- function(variable,value){
  return(s.names[value])
}

# Plot
ggplot(data = e.sum, aes(x=Age)) +
  geom_boxplot(aes(fill=Sample
                  ,ymin = ymin,ymax = ymax
                  ,middle = middle
                  ,upper = upper
                  ,lower=lower)
               ,stat='identity',size=0.2) +
  # scale_y_continuous(limits = c(-0.03,0.03)) +
  scale_x_discrete(breaks=seq(0,110,10)) +
  # theme(legend.justification=c(1,0), legend.position=c(.15,0.02)) +
  theme(legend.position="bottom", legend.box = "horizontal") +

```

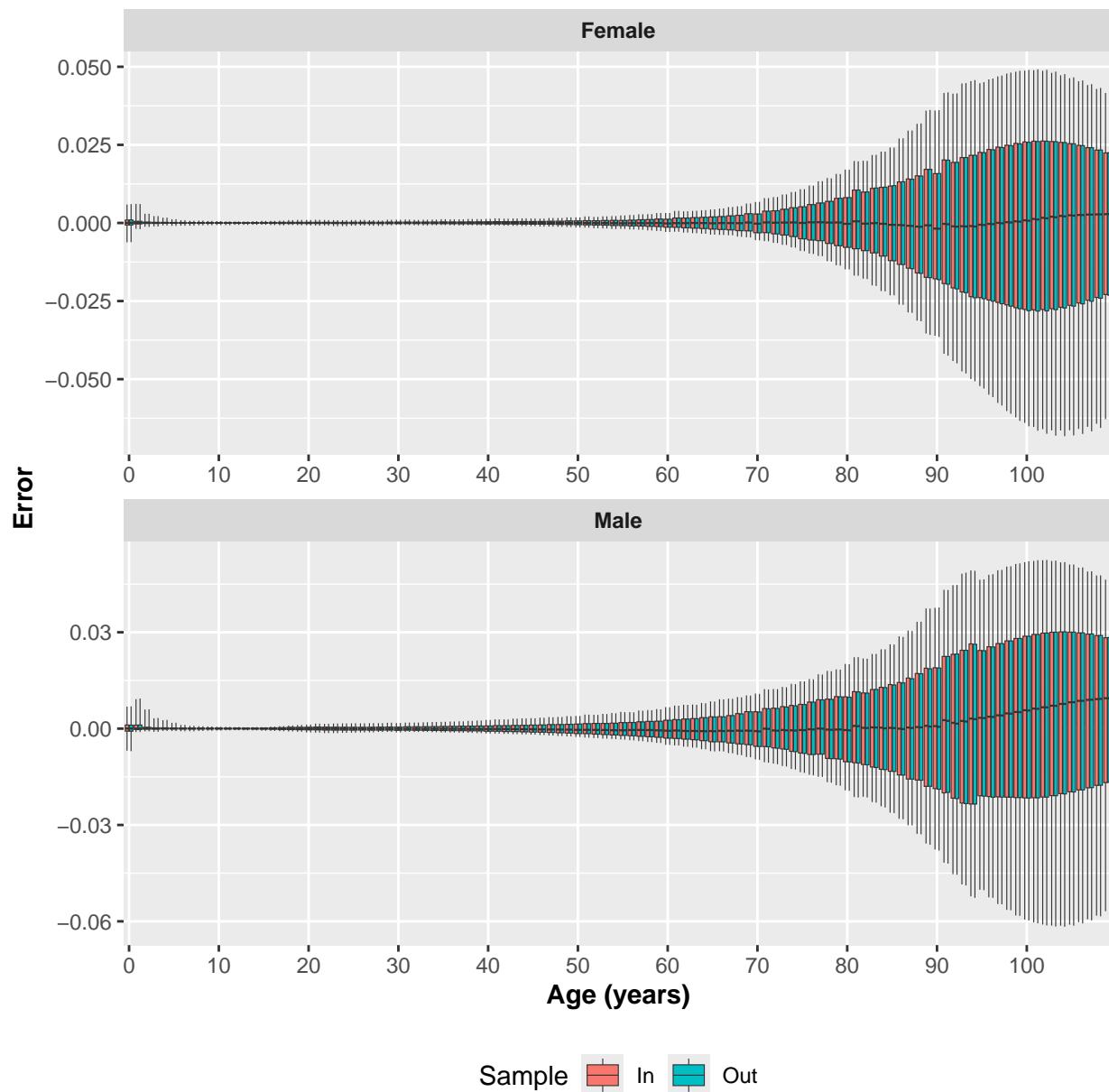
```

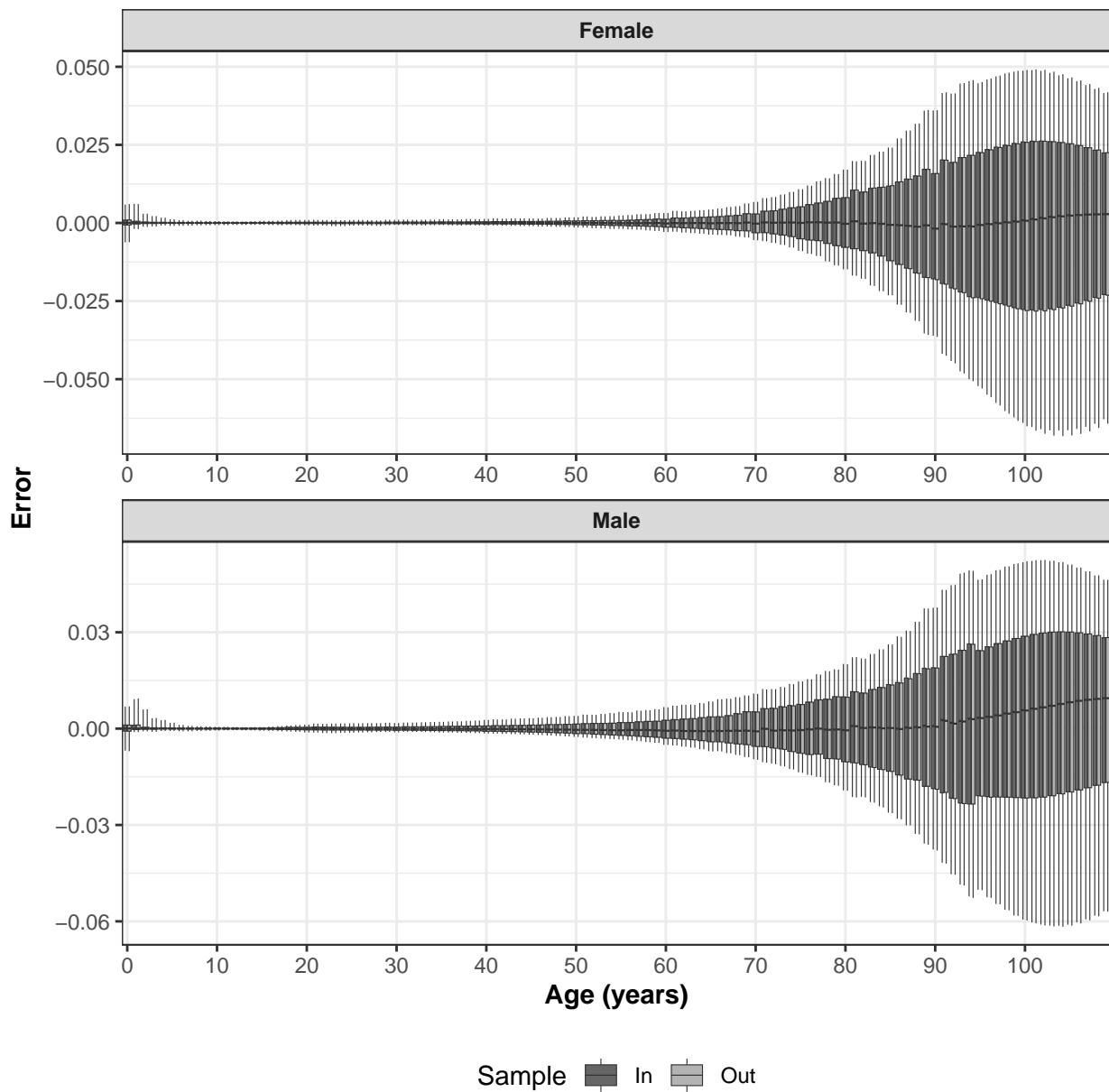
# facet_wrap(~Sex, ncol=1) +
facet_wrap(~Sex, ncol=1, scales="free", labeller=s.labeller) +
labs(x=expression(bold("Age (years)")), y=expression(bold("Error")))
ggsave("../figures/fig4.pdf", width=6.5, height=6.5, units=c("in"))

# grayscale
ggplot(data = e.sum, aes(x=Age)) +
  geom_boxplot(aes(fill=Sample
                  ,ymin = ymin, ymax = ymax
                  ,middle = middle
                  ,upper = upper
                  ,lower=lower)
                ,stat='identity', size=0.2) +
  # scale_y_continuous(limits = c(-0.03,0.03)) +
  scale_x_discrete(breaks=seq(0,110,10)) +
  # theme(legend.justification=c(1,0), legend.position=c(.15,0.02)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  # facet_wrap(~Sex, ncol=1) +
  facet_wrap(~Sex, ncol=1, scales="free", labeller=s.labeller) +
  labs(x=expression(bold("Age (years)")), y=expression(bold("Error"))) +
  scale_fill_grey(start = 0.4, end = .7)
ggsave("../figures/fig4-BW.pdf", width=6.5, height=6.5, units=c("in"))

# clean up
rm(list=c("e.sum", "eb", "mod.0_5.50.f", "mod.0_5.50.m"))

```





Plot the prediction error distributions by sample fraction for 50 50% samples

```
# medians
# female
es.s.f <- rbind(
  cbind("Female", "In", "10%", errsum.meds.1.f[,1]),
  cbind("Female", "In", "30%", errsum.meds.3.f[,1]),
  cbind("Female", "In", "50%", errsum.meds.5.f[,1]),
  cbind("Female", "In", "70%", errsum.meds.7.f[,1]),
  cbind("Female", "In", "90%", errsum.meds.9.f[,1])
)

es.ns.f <- rbind(
  cbind("Female", "Out", "10%", errsum.meds.1.f[,2]),
  cbind("Female", "Out", "30%", errsum.meds.3.f[,2]),
  cbind("Female", "Out", "50%", errsum.meds.5.f[,2]),
```

```

    cbind("Female","Out","70%",errsum.meds.7.f[,2]),
    cbind("Female","Out","90%",errsum.meds.9.f[,2])
)

es.f <- rbind(es.s.f,es.ns.f)

es.f.df <- data.frame(
  Sex = as.character(es.f[,1]),
  Sample = as.character(es.f[,2]),
  Fraction = as.character(es.f[,3]),
  Median = as.numeric(es.f[,4])
)
# str(es.f.df)

# male
es.s.m <- rbind(
  cbind("Male","In","10%",errsum.meds.1.m[,1]),
  cbind("Male","In","30%",errsum.meds.3.m[,1]),
  cbind("Male","In","50%",errsum.meds.5.m[,1]),
  cbind("Male","In","70%",errsum.meds.7.m[,1]),
  cbind("Male","In","90%",errsum.meds.9.m[,1])
)

es.ns.m <- rbind(
  cbind("Male","Out","10%",errsum.meds.1.m[,2]),
  cbind("Male","Out","30%",errsum.meds.3.m[,2]),
  cbind("Male","Out","50%",errsum.meds.5.m[,2]),
  cbind("Male","Out","70%",errsum.meds.7.m[,2]),
  cbind("Male","Out","90%",errsum.meds.9.m[,2])
)

es.m <- rbind(es.s.m,es.ns.m)

es.m.df <- data.frame(
  Sex = as.character(es.m[,1]),
  Sample = as.character(es.m[,2]),
  Fraction = as.character(es.m[,3]),
  Median = as.numeric(es.m[,4])
)
# str(es.m.df)

es.df <- rbind(es.f.df,es.m.df)
# str(es.df)

# order female first
es.df[,1] <- factor(es.df[,1], levels=c("Female","Male"))
# str(es.df)

e.sum <- ddply(es.df,.(Sex, Sample, Fraction),
  summarize,
  ymin = quantile(Median,.1),
  ymax = quantile(Median,.9),
  middle = median(Median),

```

```

    lower = quantile(Median,0.25),
    upper = quantile(Median,0.75)
)

s.names <- list(
  'S#1' = expression(bold("Female")),
  'S#2' = expression(bold("Male"))
)
# s.names

s.labeller <- function(variable,value){
  return(s.names[value])
}

# Plot
ggplot(data = e.sum, aes(x=Fraction)) +
  geom_boxplot(aes(fill=Sample
                    ,ymin = ymin
                    ,ymax = ymax
                    ,middle = middle
                    ,upper = upper
                    ,lower=lower),stat='identity',size=0.2) +
  geom_hline(yintercept=0) +
  # scale_y_continuous(limits = c(-2e-05,4.5e-05)) +
  # theme(legend.justification=c(1,0), legend.position=c(0.85,.56)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  labs(y = expression(bold("Median Error")))
  , x = expression(bold("Sample Percentage"))) +
  facet_wrap(~Sex,ncol=1,scales="free",labeller=s.labeller)
ggsave("../figures/fig5a.pdf",width=6.5,height=6.5,units=c("in"))

# grayscale
ggplot(data = e.sum, aes(x=Fraction)) +
  geom_boxplot(aes(fill=Sample
                    ,ymin = ymin
                    ,ymax = ymax
                    ,middle = middle
                    ,upper = upper
                    ,lower=lower),stat='identity',size=0.2) +
  geom_hline(yintercept=0) +
  # scale_y_continuous(limits = c(-2e-05,4.5e-05)) +
  # theme(legend.justification=c(1,0), legend.position=c(0.85,.56)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  labs(y = expression(bold("Median Error")))
  , x = expression(bold("Sample Percentage"))) +
  facet_wrap(~Sex,ncol=1,scales="free",labeller=s.labeller) +
  scale_fill_grey(start = 0.4, end = .7)
ggsave("../figures/fig5a-BW.pdf",width=6.5,height=6.5,units=c("in"))

# iqrs
# female

```

```

es.s.f <- rbind(
  cbind("Female","In","10%",errsum.iqrs.1.f[,1]),
  cbind("Female","In","30%",errsum.iqrs.3.f[,1]),
  cbind("Female","In","50%",errsum.iqrs.5.f[,1]),
  cbind("Female","In","70%",errsum.iqrs.7.f[,1]),
  cbind("Female","In","90%",errsum.iqrs.9.f[,1])
)

es.ns.f <- rbind(
  cbind("Female","Out","10%",errsum.iqrs.1.f[,2]),
  cbind("Female","Out","30%",errsum.iqrs.3.f[,2]),
  cbind("Female","Out","50%",errsum.iqrs.5.f[,2]),
  cbind("Female","Out","70%",errsum.iqrs.7.f[,2]),
  cbind("Female","Out","90%",errsum.iqrs.9.f[,2])
)

es.f <- rbind(es.s.f,es.ns.f)

es.f.df <- data.frame(
  Sex = as.character(es.f[,1]),
  Sample = as.character(es.f[,2]),
  Fraction = as.character(es.f[,3]),
  Median = as.numeric(es.f[,4])
)
# str(es.f.df)

# male
es.s.m <- rbind(
  cbind("Male","In","10%",errsum.iqrs.1.m[,1]),
  cbind("Male","In","30%",errsum.iqrs.3.m[,1]),
  cbind("Male","In","50%",errsum.iqrs.5.m[,1]),
  cbind("Male","In","70%",errsum.iqrs.7.m[,1]),
  cbind("Male","In","90%",errsum.iqrs.9.m[,1])
)

es.ns.m <- rbind(
  cbind("Male","Out","10%",errsum.iqrs.1.m[,2]),
  cbind("Male","Out","30%",errsum.iqrs.3.m[,2]),
  cbind("Male","Out","50%",errsum.iqrs.5.m[,2]),
  cbind("Male","Out","70%",errsum.iqrs.7.m[,2]),
  cbind("Male","Out","90%",errsum.iqrs.9.m[,2])
)

es.m <- rbind(es.s.m,es.ns.m)

es.m.df <- data.frame(
  Sex = as.character(es.m[,1]),
  Sample = as.character(es.m[,2]),
  Fraction = as.character(es.m[,3]),
  Median = as.numeric(es.m[,4])
)
# str(es.m.df)

```

```

es.df <- rbind(es.m.df,es.f.df)

# order female first
es.df[,1] <- factor(es.df[,1], levels=c("Female","Male"))
# str(es.df)

e.sum <- ddply(es.df,.(Sex, Sample, Fraction),
  summarize,
  ymin = quantile(Median,.1),
  ymax = quantile(Median,.9),
  middle = median(Median),
  lower = quantile(Median,0.25),
  upper = quantile(Median,0.75)
)

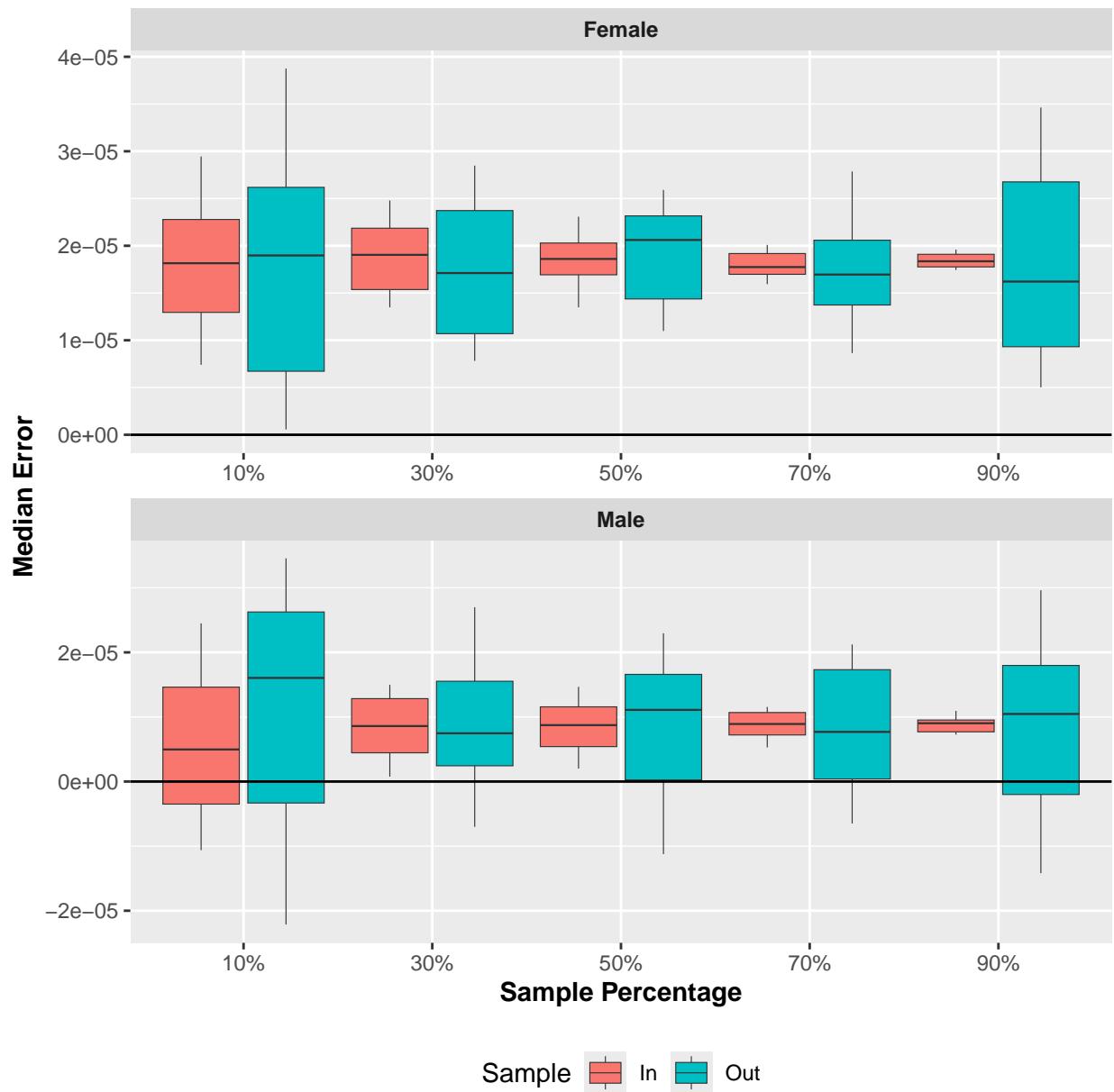
# Plot
ggplot(data = e.sum, aes(x=Fraction)) +
  geom_boxplot(aes(fill=Sample
                  ,ymin = ymin
                  ,ymax = ymax
                  ,middle = middle
                  ,upper = upper
                  ,lower=lower),stat='identity',size=0.2) +
  # geom_hline(yintercept=0) +
  # scale_y_continuous(limits = c(0.0025,0.005)) +
  # theme(legend.justification=c(1,0), legend.position=c(0.85,.84)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  labs(y = expression(bold("Error Interquartile Range")))
  , x = expression(bold("Sample Percentage"))) +
  facet_wrap(~Sex,ncol=1,scales="free",labeller=s.labeller)
ggsave("../figures/fig5b.pdf",width=6.5,height=6.5,units=c("in"))

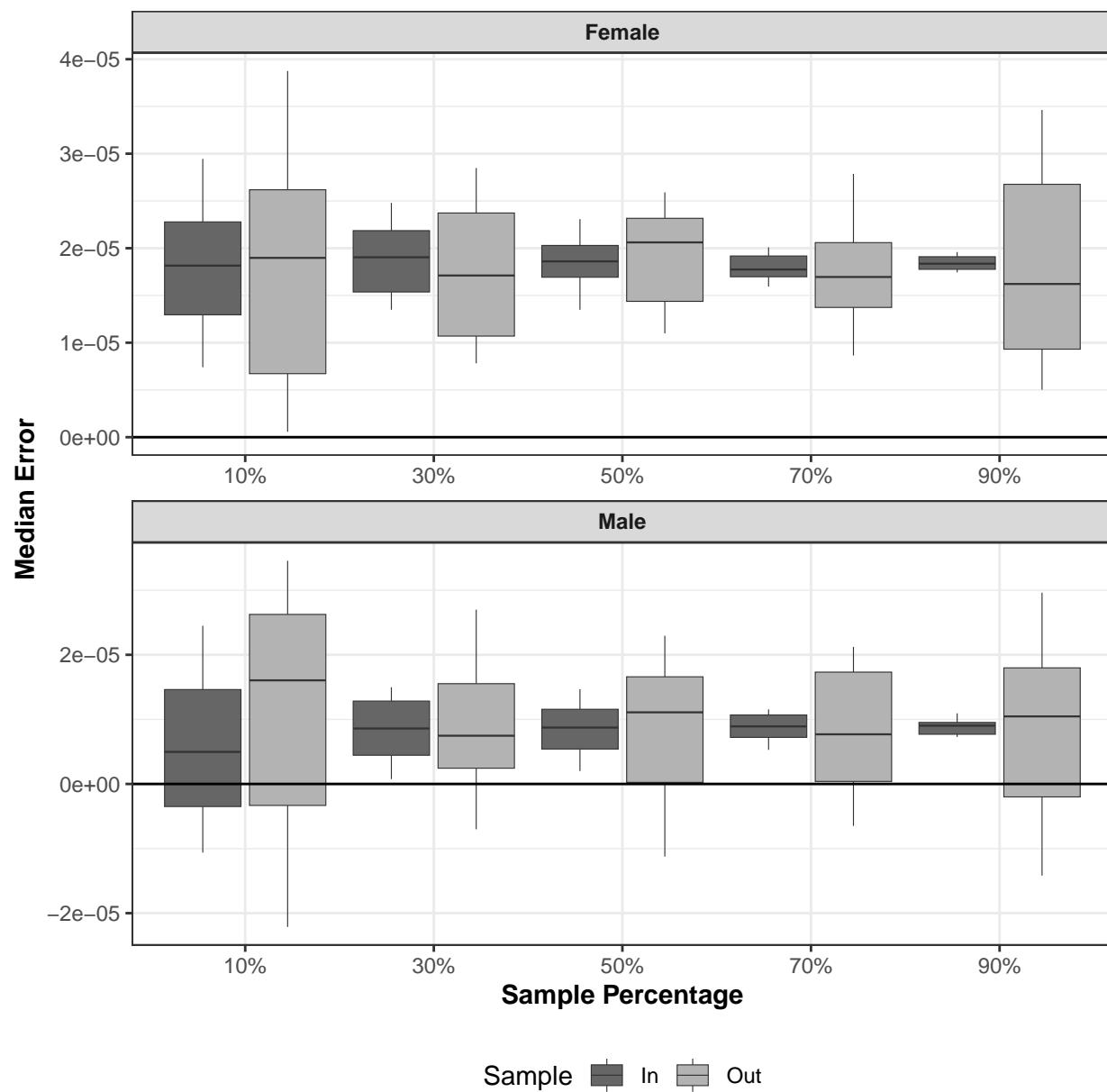
# grayscale
ggplot(data = e.sum, aes(x=Fraction)) +
  geom_boxplot(aes(fill=Sample
                  ,ymin = ymin
                  ,ymax = ymax
                  ,middle = middle
                  ,upper = upper
                  ,lower=lower),stat='identity',size=0.2) +
  # geom_hline(yintercept=0) +
  # scale_y_continuous(limits = c(0.0025,0.005)) +
  # theme(legend.justification=c(1,0), legend.position=c(0.85,.84)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  labs(y = expression(bold("Error Interquartile Range")))
  , x = expression(bold("Sample Percentage"))) +
  facet_wrap(~Sex,ncol=1,scales="free",labeller=s.labeller) +
  scale_fill_grey(start = 0.4, end = .7)
ggsave("../figures/fig5b-BW.pdf",width=6.5,height=6.5,units=c("in"))

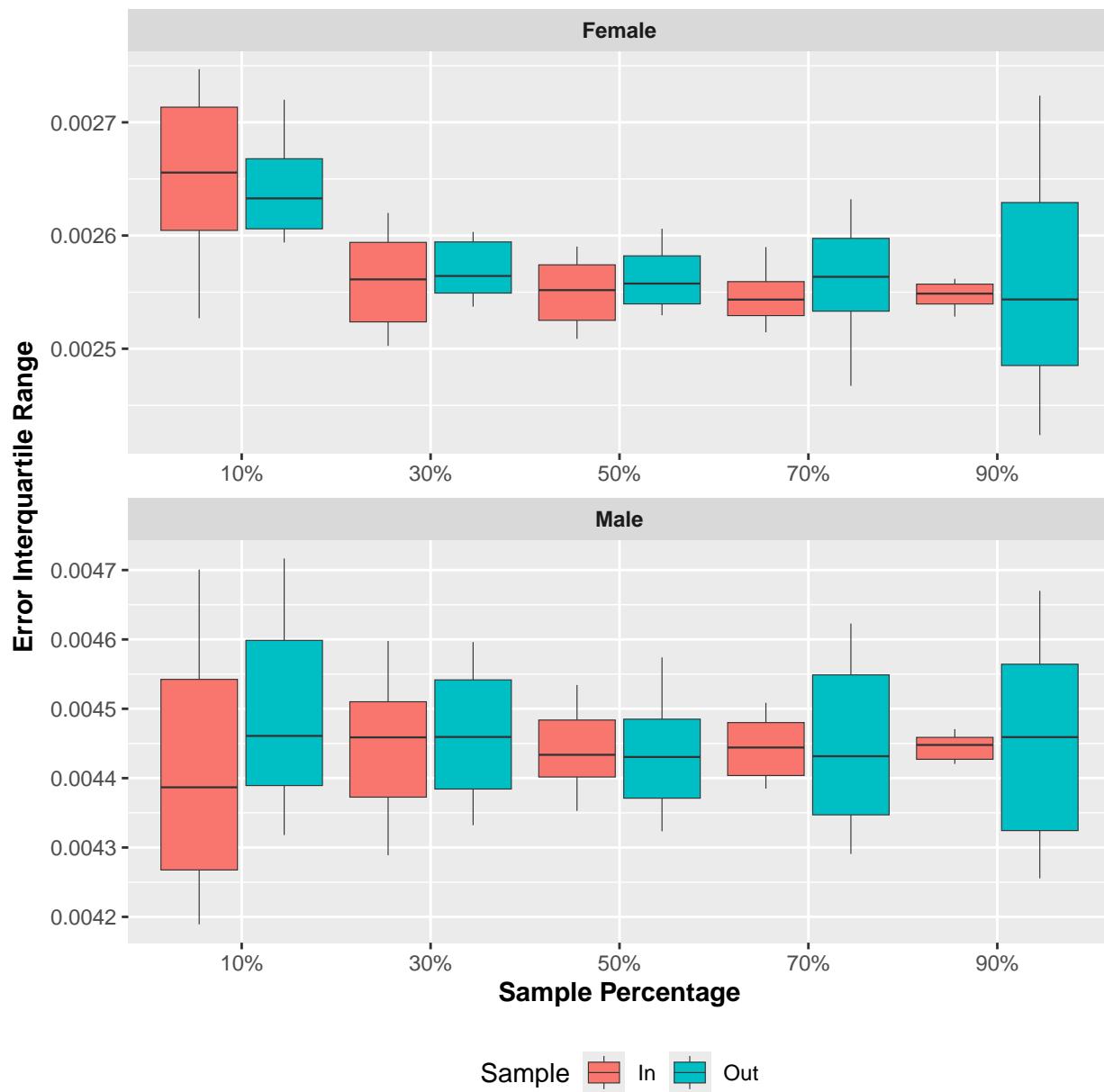
# clean up
rm(list=c("e.sum","es.df","es.m.df","es.m","es.ns.m")

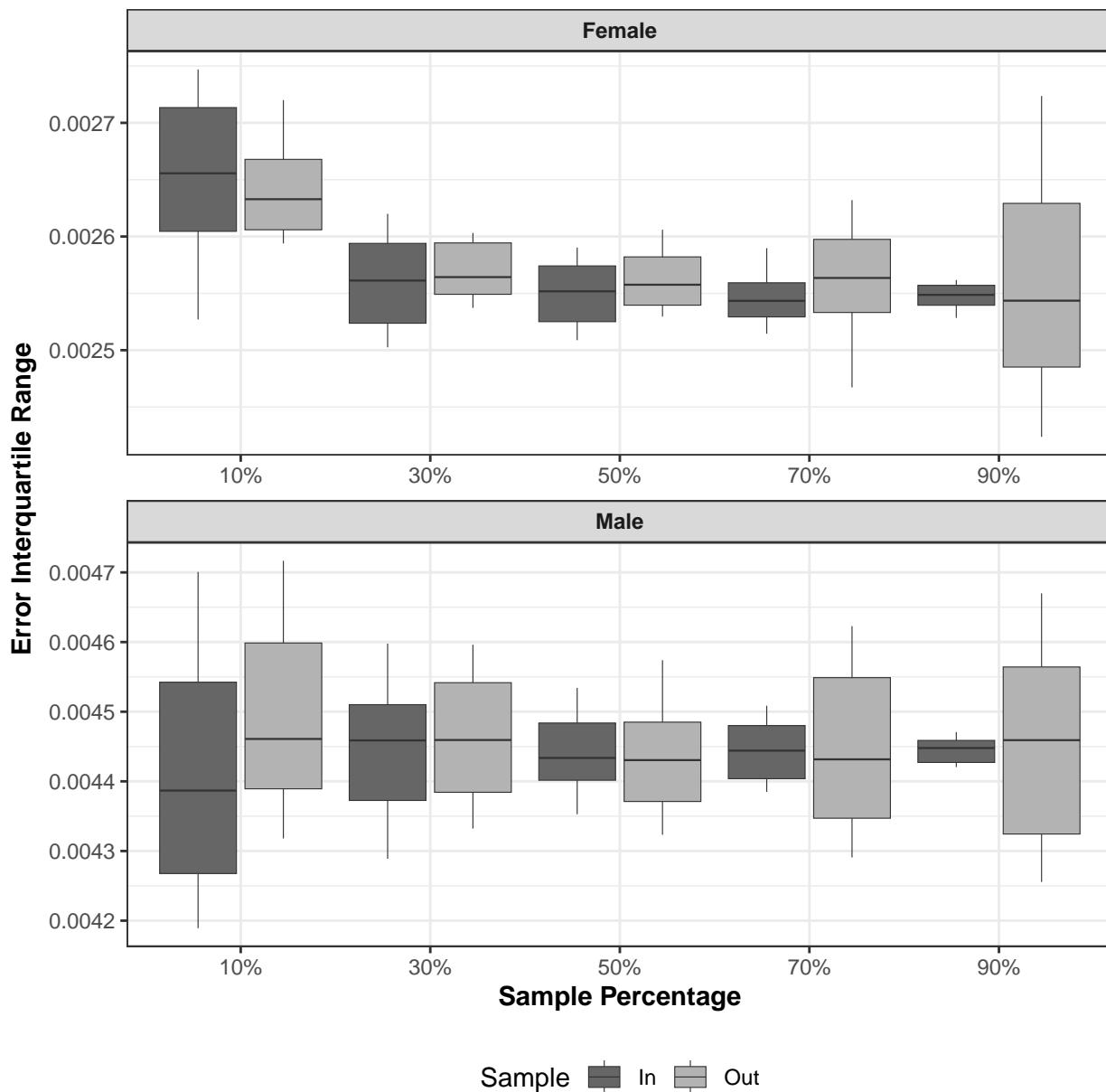
```

```
, "es.s.m", "es.f.df", "es.f", "es.ns.f", "es.s.f"))
```









Plot right singular vectors versus child mortality, $5q_0$.

```
# svds
svd.m <- mod.1_0.m$svd$s1
svd.f <- mod.1_0.f$svd$s1

# right singular vectors
vs.cm <- rbind(
  cbind("Female", "v1", svd.f$v[, 1], Qlogit.f[1,]),
  cbind("Female", "v2", svd.f$v[, 2], Qlogit.f[1,]),
  cbind("Female", "v3", svd.f$v[, 3], Qlogit.f[1,]),
  cbind("Female", "v4", svd.f$v[, 4], Qlogit.f[1,]),
  cbind("Male", "v1", svd.m$v[, 1], Qlogit.m[1,]),
  cbind("Male", "v2", svd.m$v[, 2], Qlogit.m[1,]),
  cbind("Male", "v3", svd.m$v[, 3], Qlogit.m[1,]),
  cbind("Male", "v4", svd.m$v[, 4], Qlogit.m[1,])
```

```

)

vs.cm.df <- data.frame(
  Sex = as.character(vs.cm[,1]),
  V = as.character(vs.cm[,2]),
  Value = as.numeric(vs.cm[,3]),
  CM = as.numeric(vs.cm[,4])
)
# str(vs.cm.df)

# predicted right singular vectors
vs.cm.p <- rbind(
  cbind("Female","v1",predict(mod.1_0.f$mods$s1$v1),Qlogit.f[1,]),
  cbind("Female","v2",predict(mod.1_0.f$mods$s1$v2),Qlogit.f[1,]),
  cbind("Female","v3",predict(mod.1_0.f$mods$s1$v3),Qlogit.f[1,]),
  cbind("Female","v4",predict(mod.1_0.f$mods$s1$v4),Qlogit.f[1,]),
  cbind("Male","v1",predict(mod.1_0.m$mods$s1$v1),Qlogit.m[1,]),
  cbind("Male","v2",predict(mod.1_0.m$mods$s1$v2),Qlogit.m[1,]),
  cbind("Male","v3",predict(mod.1_0.m$mods$s1$v3),Qlogit.m[1,]),
  cbind("Male","v4",predict(mod.1_0.m$mods$s1$v4),Qlogit.m[1,])
)
# str(vs.cm.p)

vs.cm.p.df <- data.frame(
  Sex = as.character(vs.cm.p[,1]),
  V = as.character(vs.cm.p[,2]),
  Value = as.numeric(vs.cm.p[,3]),
  CM = as.numeric(vs.cm.p[,4])
)
# str(vs.cm.p.df)

v.names <- list(
  'V#1' = expression(bold('v')[1]),
  'V#2' = expression(bold('v')[2]),
  'V#3' = expression(bold('v')[3]),
  'V#4' = expression(bold('v')[4])
)
# str(v.names)

v.labeller <- function(variable,value){
  return(v.names[value])
}

vs.cm <- rbind(
  cbind(vs.cm.df,Type="Data"),
  cbind(vs.cm.p.df,Type="Predicted")
)
# str(vs.cm)

# Plot female
ggplot(data = vs.cm[which(vs.cm[,1]=="Female"),]
  , aes(x=CM, y=Value, group=Type, colour=Type)) +
  geom_point(size=0.2) +
  labs(y = expression(bold("RSV Element Values")))
  , x = expression(bold('Child Mortality '))

```

```

        [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')))) +
# theme(legend.justification=c(1,0), legend.position=c(0.99,.88)) +
theme(legend.position="bottom", legend.box = "horizontal") +
theme(legend.title=element_blank()) +
facet_wrap(~V, scale="free")
# facet_wrap(~V, scale="free", labeller=v.labeller)
ggsave("../figures/fig2-1f.pdf",width=6.5,height=6.5,units=c("in"))

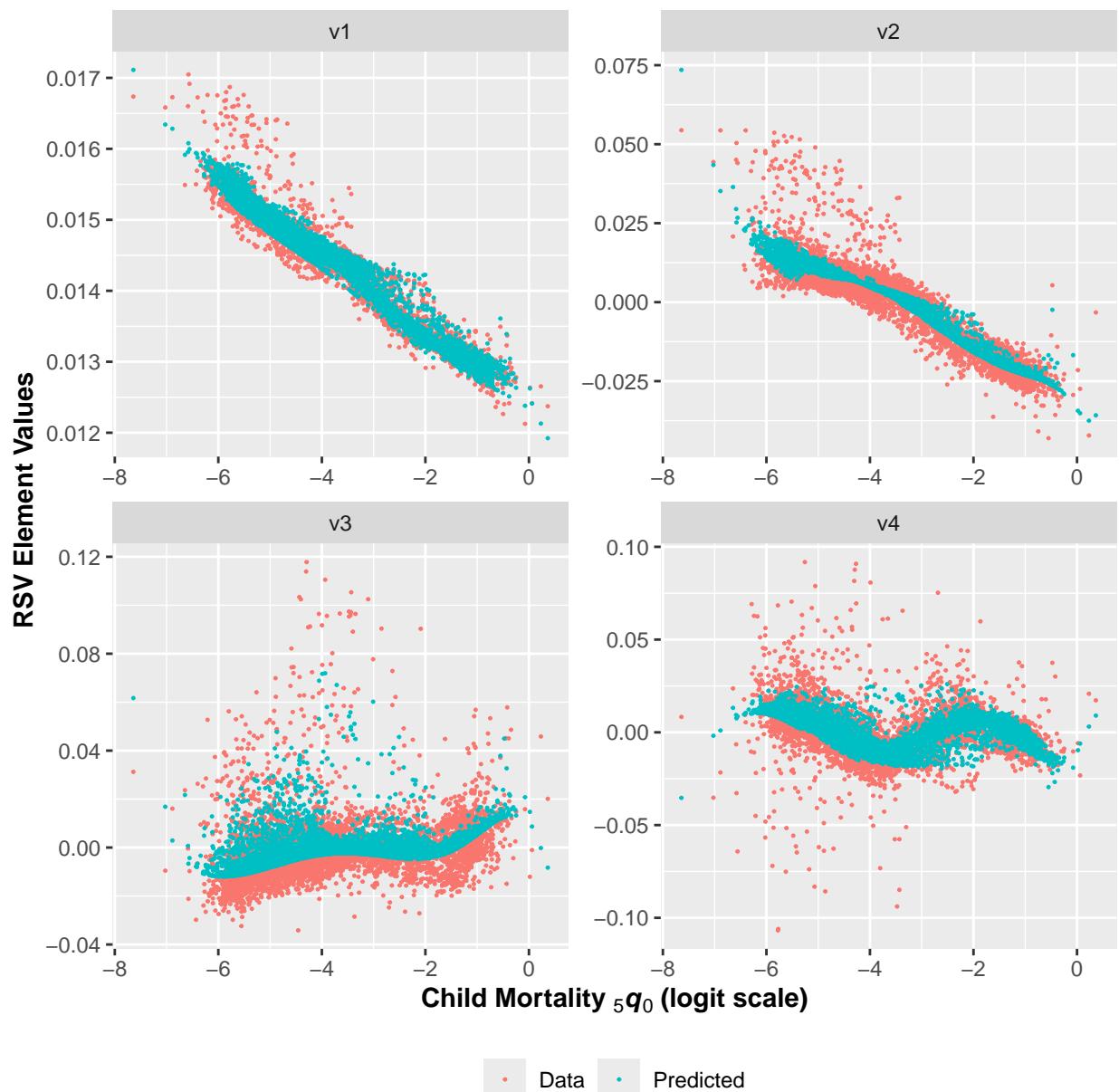
# grayscale
ggplot(data = vs.cm[which(vs.cm[,1]=="Female"),]
      , aes(x=CM, y=Value, group=Type, colour=Type)) +
geom_point(size=0.2) +
labs(y = expression(bold("RSV Element Values"))
     , x = expression(bold('Child Mortality '))
        [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')))) +
# theme(legend.justification=c(1,0), legend.position=c(0.99,.88)) +
theme_bw() +
theme(legend.position="bottom", legend.box = "horizontal") +
theme(legend.title=element_blank()) +
facet_wrap(~V, scale="free") +
# facet_wrap(~V, scale="free", labeller=v.labeller) +
scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig2-1f-BW.pdf",width=6.5,height=6.5,units=c("in"))

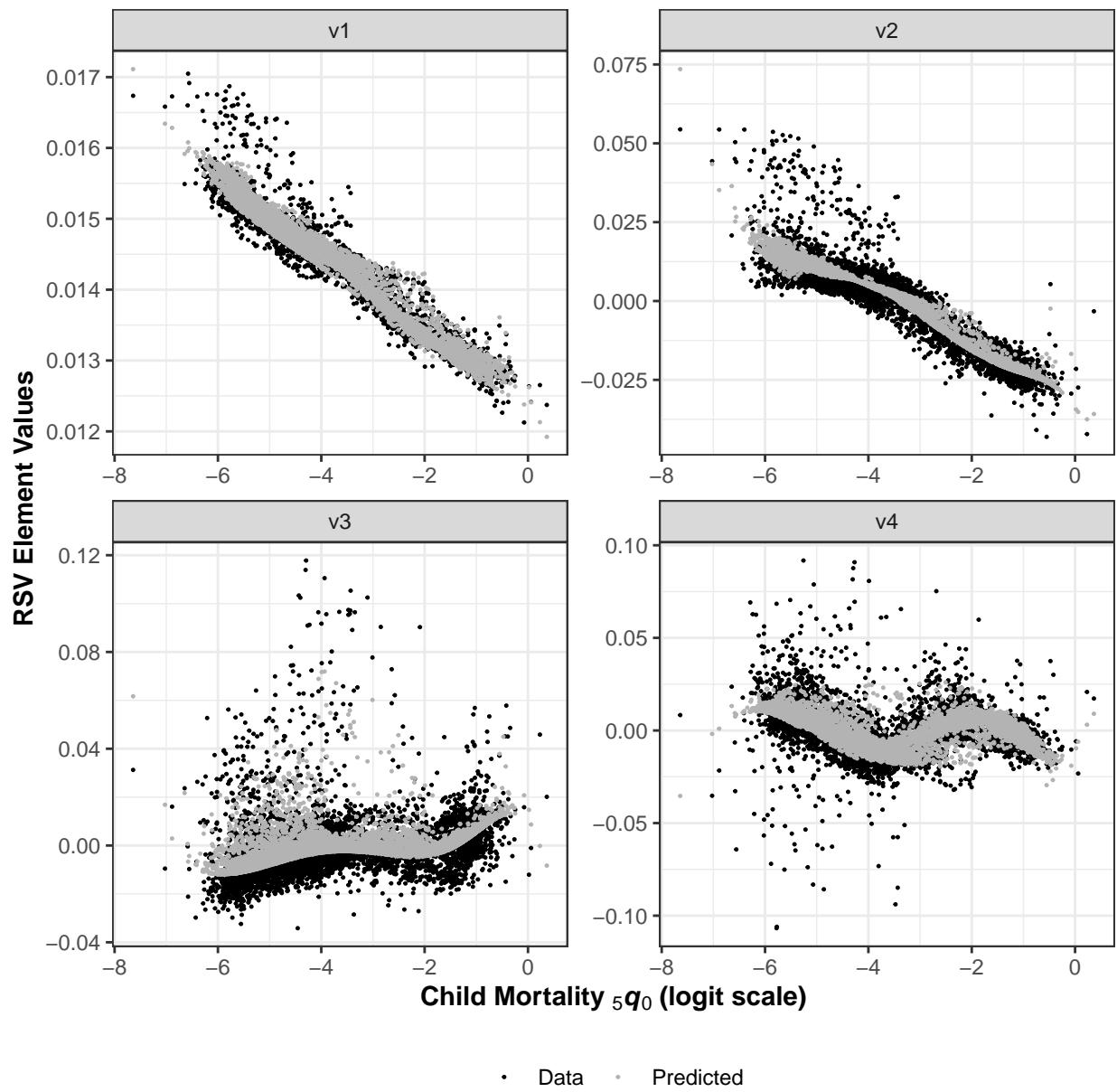
# Plot male
ggplot(data = vs.cm[which(vs.cm[,1]=="Male"),]
      , aes(x=CM, y=Value, group=Type, colour=Type)) +
geom_point(size=0.2) +
labs(y = expression(bold("RSV Element Values"))
     , x = expression(bold('Child Mortality '))
        [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')))) +
# theme(legend.justification=c(1,0), legend.position=c(0.99,.88)) +
theme_bw() +
theme(legend.position="bottom", legend.box = "horizontal") +
theme(legend.title=element_blank()) +
facet_wrap(~V, scale="free")
# facet_wrap(~V, scale="free", labeller=v.labeller)
ggsave("../figures/fig2-1m.pdf",width=6.5,height=6.5,units=c("in"))

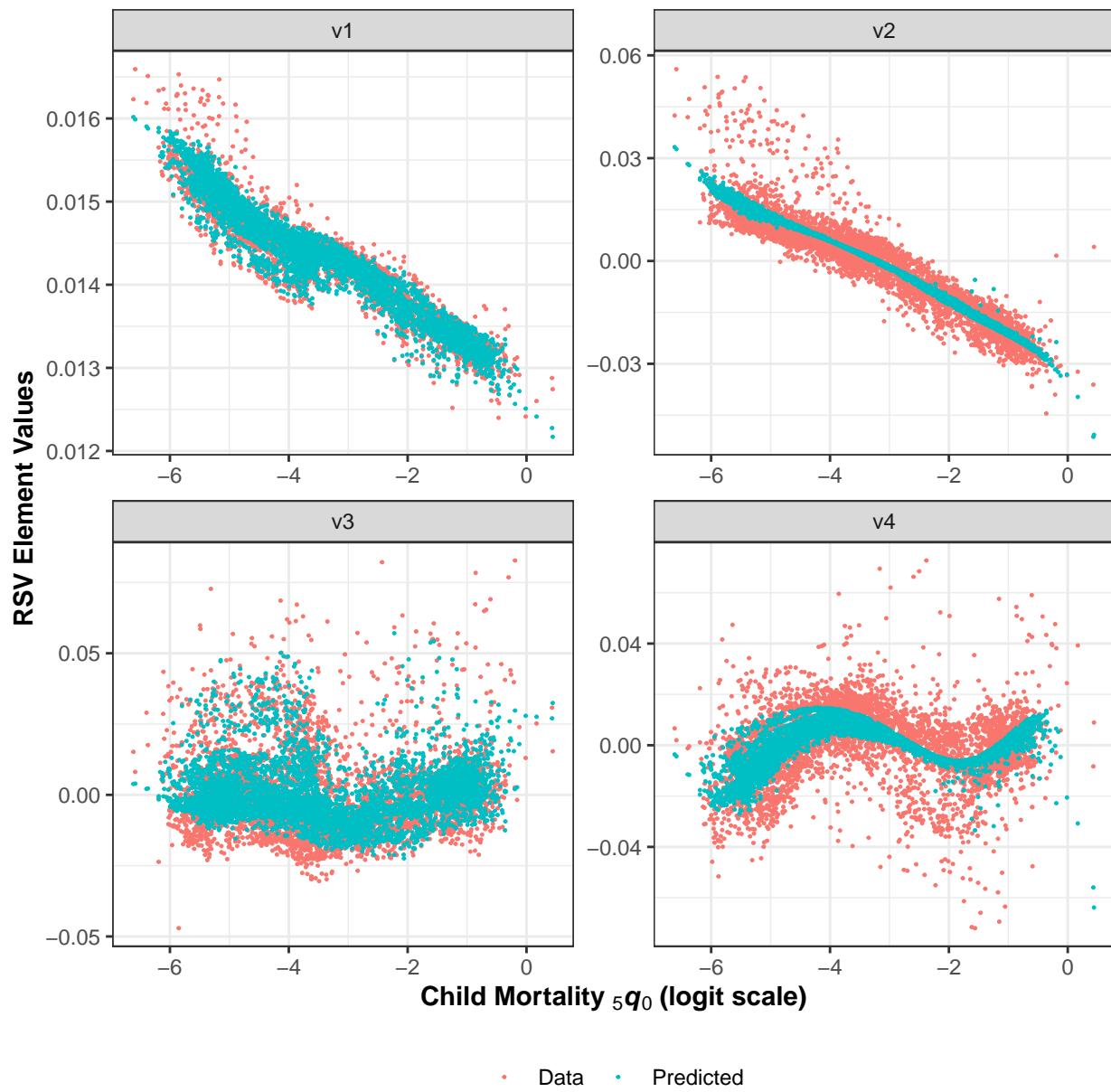
# grayscale
ggplot(data = vs.cm[which(vs.cm[,1]=="Male"),]
      , aes(x=CM, y=Value, group=Type, colour=Type)) +
geom_point(size=0.2) +
labs(y = expression(bold("RSV Element Values"))
     , x = expression(bold('Child Mortality '))
        [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')))) +
# theme(legend.justification=c(1,0), legend.position=c(0.99,.88)) +
theme(legend.position="bottom", legend.box = "horizontal") +
theme(legend.title=element_blank()) +
facet_wrap(~V, scale="free") +
# facet_wrap(~V, scale="free", labeller=v.labeller) +
scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig2-1m-BW.pdf",width=6.5,height=6.5,units=c("in"))

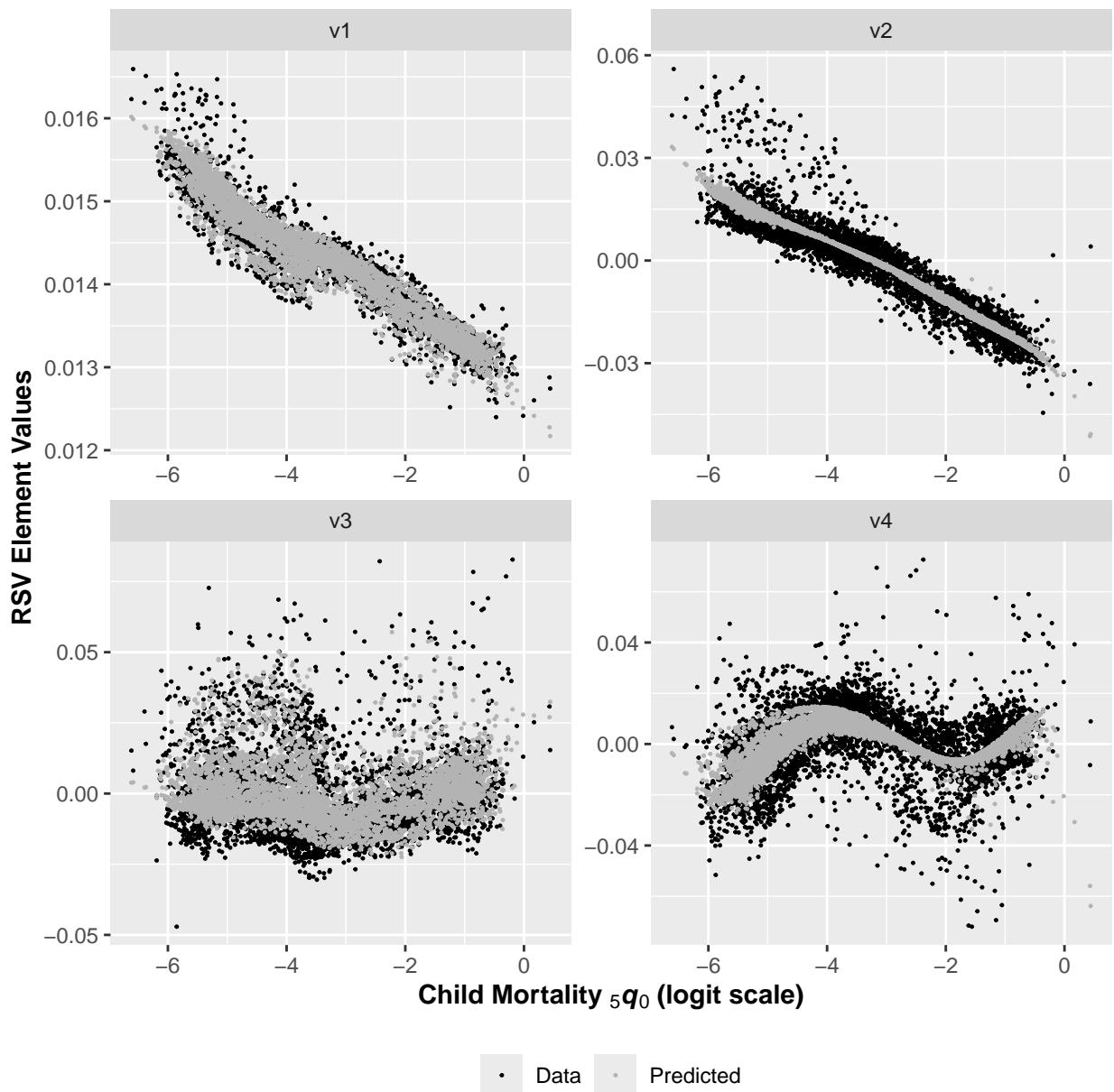
```

```
# clean up
rm(list=c("vs.cm","v.labeller","v.names",
,"vs.cm.p.pdf","vs.cm.p","vs.cm.df","vs.cm","svd.m","svd.f"))
```









Plot adult mortality, $45q_{15}$, by child mortality, $5q_0$.

```
# data
am.cm <- rbind(
  cbind("Male",Qlogit.m[1],Qlogit.m[2]),
  cbind("Female",Qlogit.f[1],Qlogit.f[2]))
)

am.cm.df <- data.frame(
  Sex = as.character(am.cm[,1]),
  CM = as.numeric(am.cm[,2]),
  AM = as.numeric(am.cm[,3]))
#
# str(am.cm.df)

# predicted
```

```

am.cm.p <- rbind(
  cbind("Male",Qlogit.m[1,],predict(mod.1_0.m$mods$s1$aml)),
  cbind("Female",Qlogit.f[1,],predict(mod.1_0.f$mods$s1$aml))
)

am.cm.p.df <- data.frame(
  Sex = as.character(am.cm.p[,1]),
  CM = as.numeric(am.cm.p[,2]),
  AM = as.numeric(am.cm.p[,3])
)
# str(am.cm.p.df)

am.cm <- rbind(
  cbind(am.cm.df,Type="Data"),
  cbind(am.cm.p.df,Type="Predicted")
)
# str(am.cm)

s.names <- list(
  'S#1' = expression(bold("Female")),
  'S#2' = expression(bold("Male"))
)
# s.names

s.labeller <- function(variable,value){
  return(s.names[value])
}

# Plot
ggplot(data = am.cm, aes(x=CM, y=AM, group=Type, colour=Type)) +
  geom_point(size=0.2) +
  labs(y = expression(bold('Adult Mortality ') [bold(45)]*bolditalic('q')[bold(15)]*bold(' (logit scale)')),
       x = expression(bold('Child Mortality ') [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)'))) +
    # theme(legend.justification=c(1,0), legend.position=c(0.99,0.02)) +
    theme(legend.position="bottom", legend.box = "horizontal") +
    theme(legend.title=element_blank()) +
    facet_wrap(~Sex, scale="free")
    # facet_wrap(~Sex, scale="free", labeller=s.labeller)
ggsave("../figures/fig2-2.pdf",width=6.5,height=6.5,units=c("in"))

# grayscale
ggplot(data = am.cm, aes(x=CM, y=AM, group=Type, colour=Type)) +
  geom_point(size=0.2) +
  labs(y = expression(bold('Adult Mortality ') [bold(45)]*bolditalic('q')[bold(15)]*bold(' (logit scale)')),
       x = expression(bold('Child Mortality ') [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)'))) +
    # theme(legend.justification=c(1,0), legend.position=c(0.99,0.02)) +
    theme_bw() +
    theme(legend.position="bottom", legend.box = "horizontal") +
    theme(legend.title=element_blank())

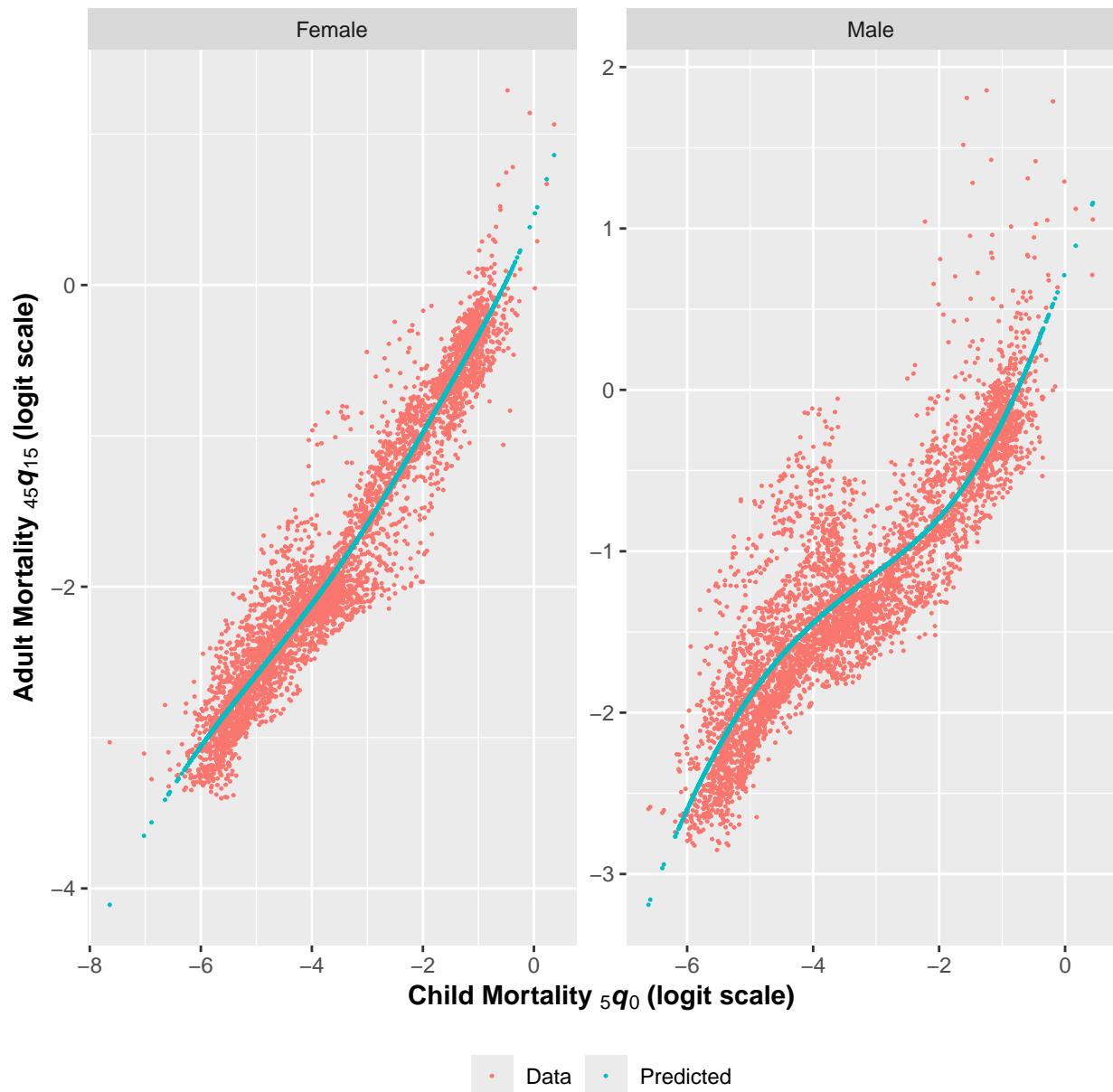
```

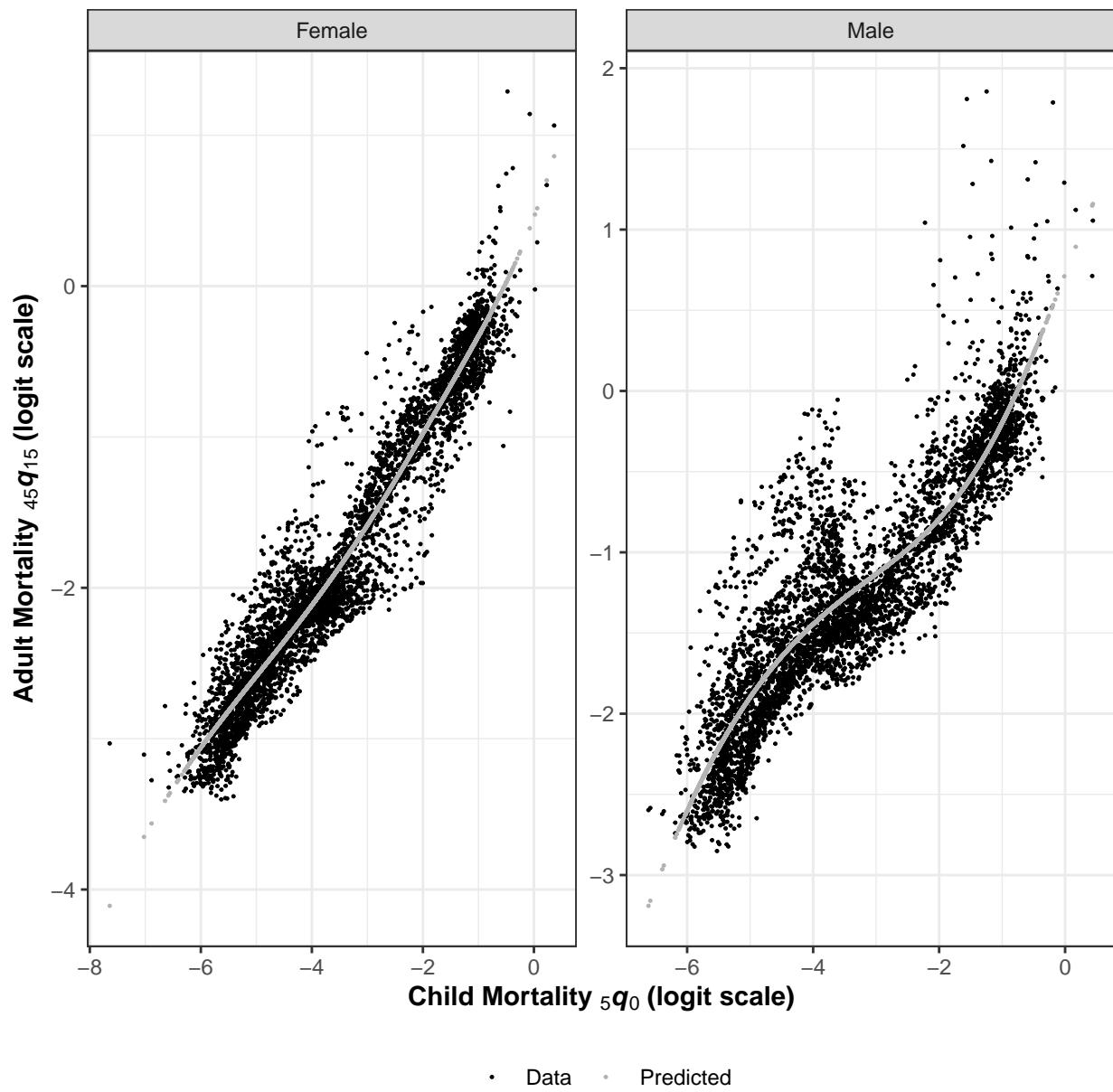
```

facet_wrap(~Sex, scale="free") +
# facet_wrap(~Sex, scale="free", labeller=s.labeller) +
scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig2-2-BW.pdf", width=6.5, height=6.5, units=c("in"))

# clean up
rm(list=c("am.cm", "am.cm.p.df", "am.cm.p", "am.cm.df"))

```





Plot probability of dying in the first year of life, ${}_1q_0$, versus child mortality, ${}_5q_0$.

```
# data
q0.cm.m <- data.frame(
  Sex = as.character("Male"),
  CM = as.numeric(Qlogit.m[1,]),
  q0 = as.numeric(q1logit.m[1,])
)
# str(q0.cm.m)

q0.cm.f <- data.frame(
  Sex = as.character("Female"),
  CM = as.numeric(Qlogit.f[1,]),
  q0 = as.numeric(q1logit.f[1,])
)
# str(q0.cm.f)
```

```

q0.cm.df <- rbind(q0.cm.m, q0.cm.f)

# predicted
q0.cm.m.p <- data.frame(
  Sex = as.character("Male"),
  CM = as.numeric(Qlogit.m[1,]),
  q0 = as.numeric(predict(mod.1_0.m$mods$s1$q0))
)
# str(q0.cm.m.p)

q0.cm.f.p <- data.frame(
  Sex = as.character("Female"),
  CM = as.numeric(Qlogit.f[1,]),
  q0 = as.numeric(predict(mod.1_0.f$mods$s1$q0))
)
# str(q0.cm.f.p)

q0.cm.p.df <- rbind(q0.cm.m.p, q0.cm.f.p)

q0 <- rbind(
  cbind(q0.cm.df, Type="Data"),
  cbind(q0.cm.p.df, Type="Predicted")
)

# order female first
q0[,1] <- factor(q0[,1], levels=c("Female", "Male"))
# str(q0)

s.names <- list(
  'S#1' = expression(bold("Female")),
  'S#2' = expression(bold("Male"))
)
# s.names

s.labeller <- function(variable,value){
  return(s.names[value])
}

# Plot
ggplot(data = q0, aes(x=CM, y=q0, group=Type, colour=Type)) +
  geom_point(size=0.2) +
  labs(y = expression(' [bold(1)]*bolditalic('q')[0]*bold(' (logit scale)')),
       x = expression(bold('Child Mortality ')))
  [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')) +
  # theme(legend.justification=c(1,0), legend.position=c(0.99,0.02)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  theme(legend.title=element_blank()) +
  facet_wrap(~Sex, scale="free", labeller=s.labeller)
ggsave("../figures/fig2-3.pdf", width=6.5, height=6.5, units=c("in"))

# grayscale
ggplot(data = q0, aes(x=CM, y=q0, group=Type, colour=Type)) +
  geom_point(size=0.2) +

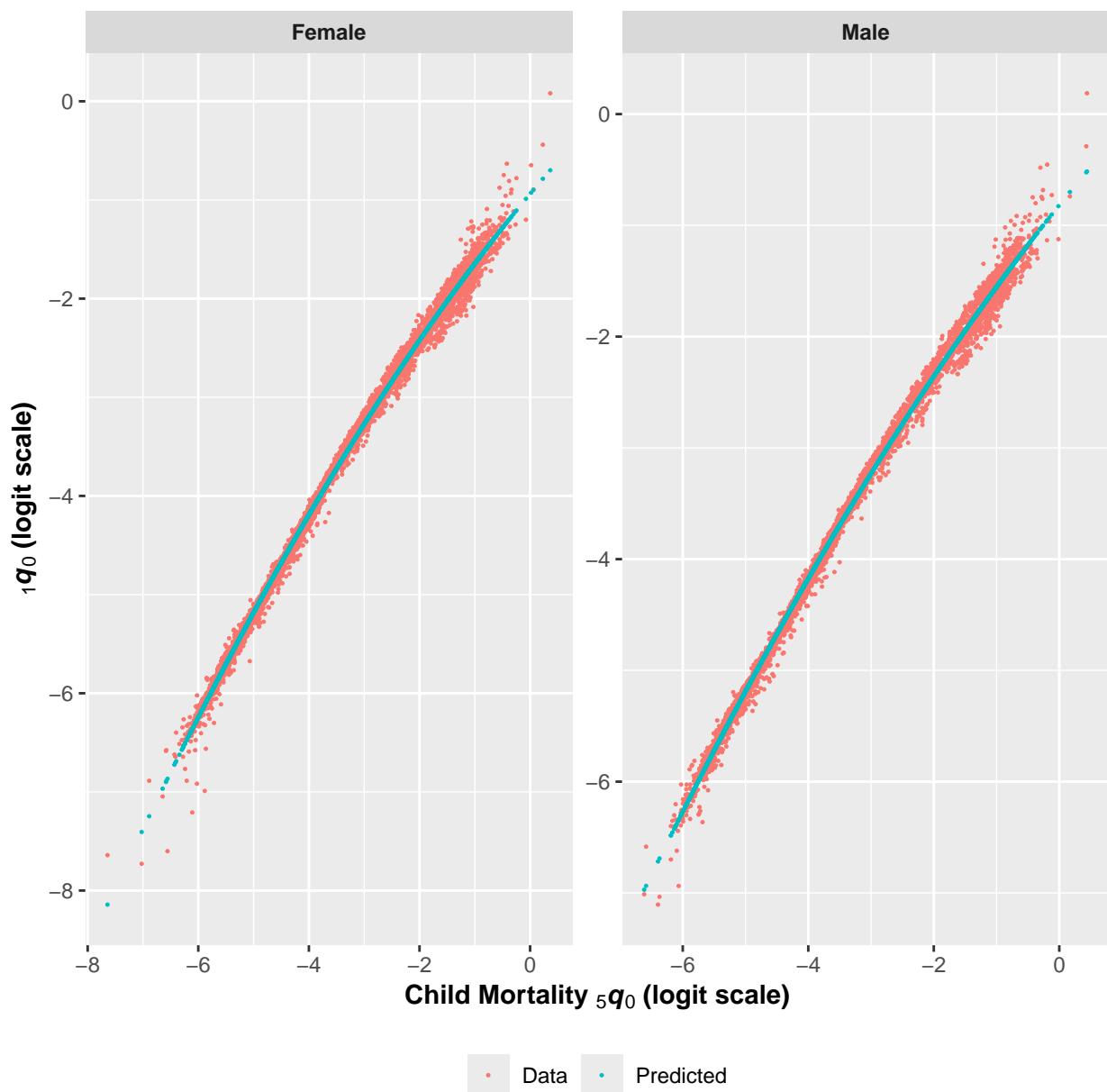
```

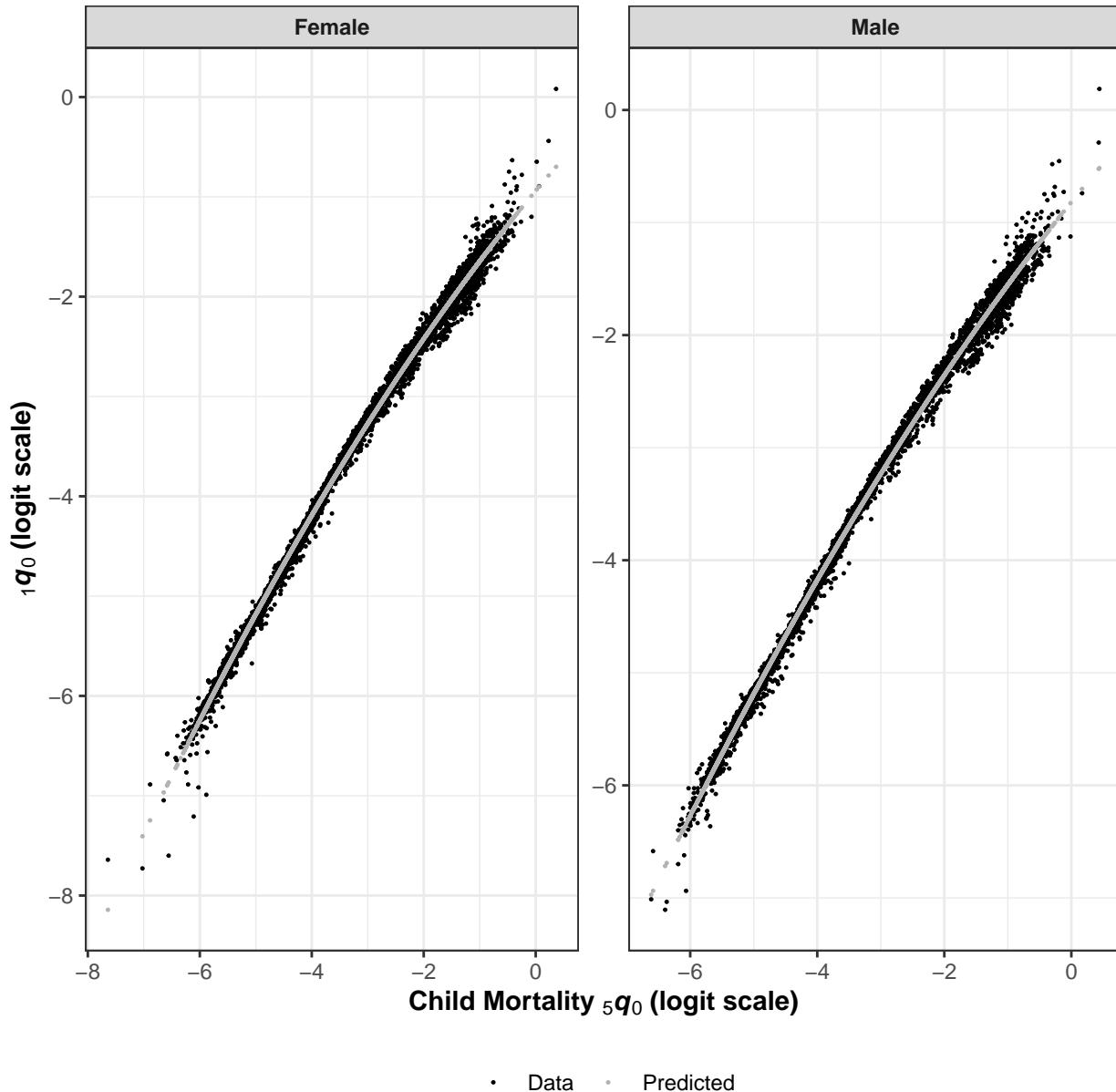
```

  labs(y = expression(" [bold(1)]*bolditalic('q')[0]*bold(' (logit scale)')")
       , x = expression(bold('Child Mortality '))
                  [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)')) + 
  # theme(legend.justification=c(1,0), legend.position=c(0.99,0.02)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  theme(legend.title=element_blank()) +
  facet_wrap(~Sex, scale="free", labeller=s.labeller) +
  scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig2-3-BW.pdf", width=6.5, height=6.5, units=c("in"))

# clean up
rm(list=c("q0","q0.cm.p.df","q0.cm.f.p"
         ,"q0.cm.m.p","q0.cm.df","q0.cm.f","q0.cm.m"))

```





Plot heuristic predictions or life tables at three levels of child mortality from very low to very high.

```
# some values for logit-scale 5q0
cml.input <- c(-5.5, -3.2, -1.5)

# predict life tables using the basic models we fit earlier
lt.f <- ltPredict(mod.1_0.sm.f, smooth=TRUE, cml.input)
# str(lt.f)
lt.m <- ltPredict(mod.1_0.sm.m, smooth=TRUE, cml.input)
# str(lt.m)

lt.p <- rbind(
  cbind("Female", as.numeric(rownames(lt.f)), cml.input[1], lt.f[,1]),
  cbind("Female", as.numeric(rownames(lt.f)), cml.input[2], lt.f[,2]),
  cbind("Female", as.numeric(rownames(lt.f)), cml.input[3], lt.f[,3]),
  cbind("Male", as.numeric(rownames(lt.m)), cml.input[1], lt.m[,1]),
```

```

    cbind("Male",as.numeric(rownames(lt.m)),cml.input[2],lt.m[,2]),
    cbind("Male",as.numeric(rownames(lt.m)),cml.input[3],lt.m[,3])
)

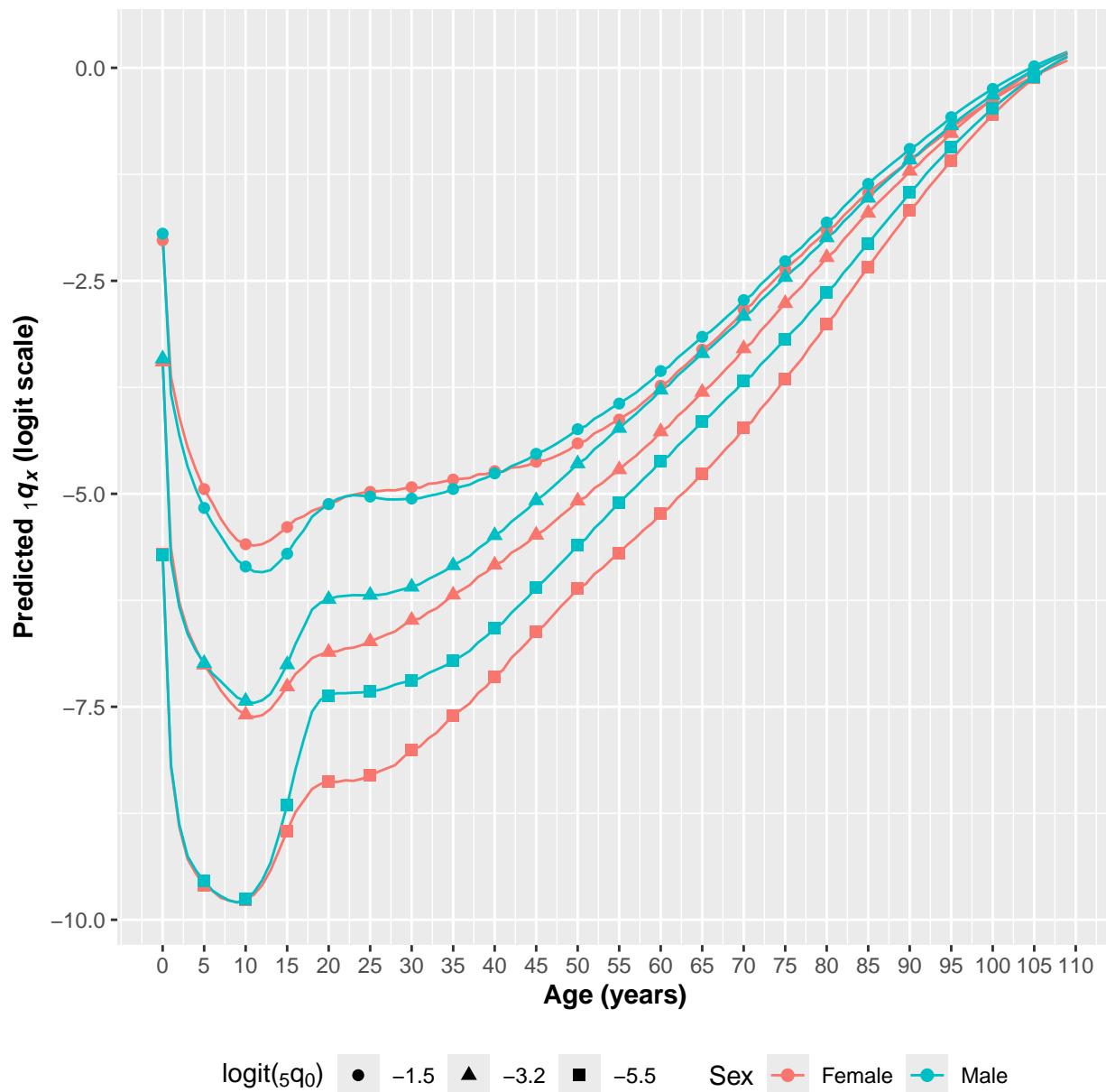
lt.p.df <- data.frame(
  Sex = as.character(lt.p[,1]),
  Age = as.numeric(lt.p[,2]),
  cml = as.character(lt.p[,3]),
  ql = as.numeric(lt.p[,4])
)
# str(lt.p.df)

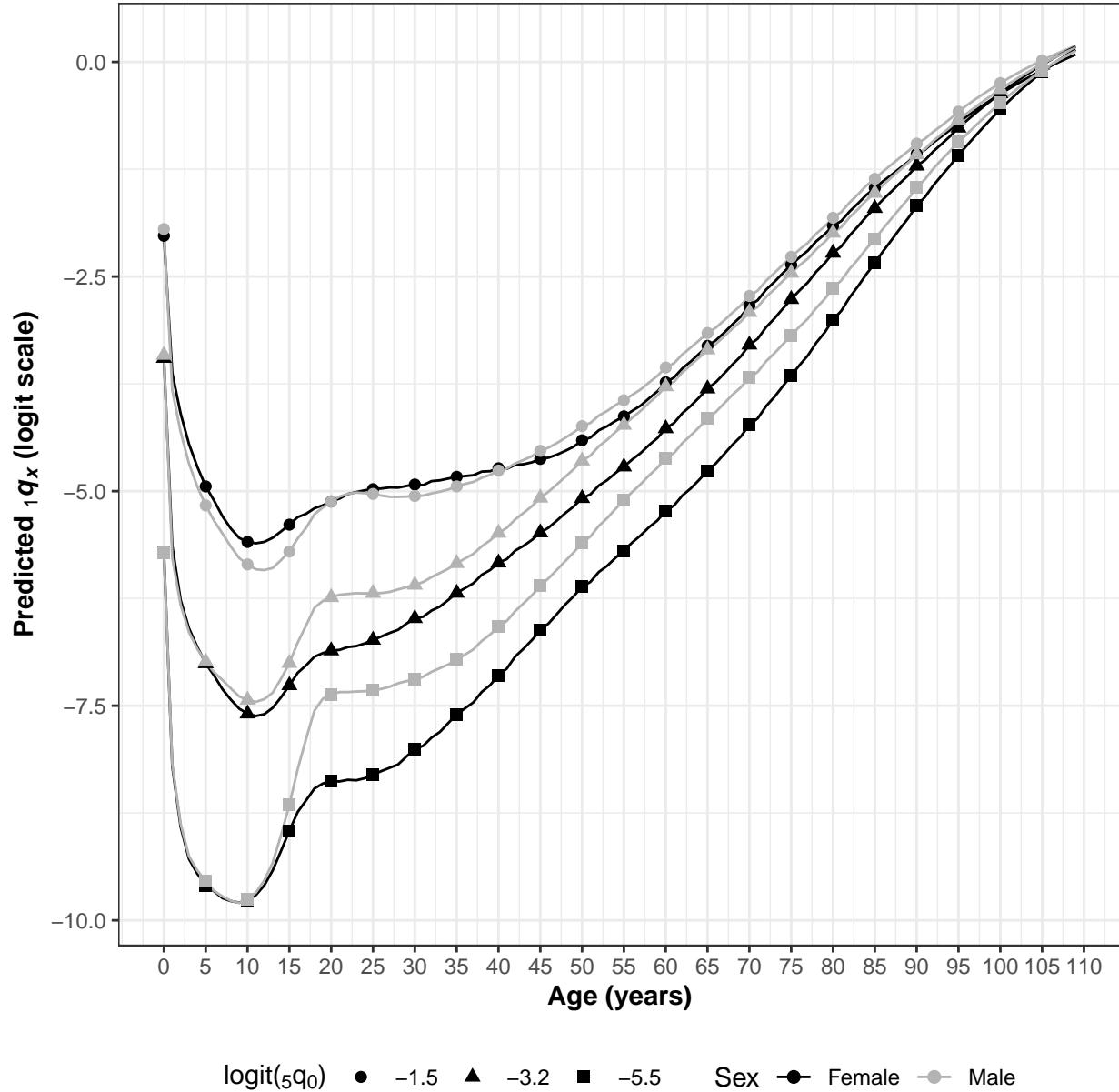
# Plot
ggplot(data = lt.p.df, aes(x=Age, y=ql
                            , group=interaction(Sex,cml), colour=Sex, shape=cml)) +
  geom_line() +
  geom_point(data = lt.p.df[seq(1, nrow(lt.p.df), 5),],size=2) +
  scale_x_continuous(breaks=c(seq(0,110,5))) +
  labs(y = expression(bold('Predicted '))
       [bold(1)]*bolditalic('q')[bolditalic(x)]*bold(' (logit scale)'))
       , x = expression(bold("Age (years)"))) +
  # theme(legend.justification=c(1,0), legend.position=c(0.95,0.05)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  scale_shape_discrete(name = expression('logit('[5]*'q'[0]*')'))
ggsave("../figures/fig6.pdf",width=6.5,height=6.5,units=c("in"))

# Plot
ggplot(data = lt.p.df, aes(x=Age, y=ql
                            , group=interaction(Sex,cml), colour=Sex, shape=cml)) +
  geom_line() +
  geom_point(data = lt.p.df[seq(1, nrow(lt.p.df), 5),],size=2) +
  scale_x_continuous(breaks=c(seq(0,110,5))) +
  labs(y = expression(bold('Predicted '))
       [bold(1)]*bolditalic('q')[bolditalic(x)]*bold(' (logit scale)'))
       , x = expression(bold("Age (years)"))) +
  # theme(legend.justification=c(1,0), legend.position=c(0.95,0.05)) +
  theme_bw() +
  theme(legend.position="bottom", legend.box = "horizontal") +
  scale_shape_discrete(name = expression('logit('[5]*'q'[0]*')')) +
  scale_colour_grey(start = 0, end = .7)
ggsave("../figures/fig6-BW.pdf",width=6.5,height=6.5,units=c("in"))

# clean up
rm(list=c("lt.p.df","lt.p","lt.m","lt.f","cml.input"))

```





6 Make Tables

Load the Stargazer and xtable packages for making nice LaTeX tables from regression output. The code uses `capture.output()` to redirect output to text files where either complete LaTeX tables are stored, or the rows of LaTeX tables are stored. The rows-only tables slot nicely into headers and footers that are nicely formatted in the manuscript, and the whole tables (Stargazer output) are completely ready to go and slot straight into the LaTeX.

Create table describing which HMD life tables are included in the analysis.

```
# recall that the same life tables are used for females and males
all.equal(colnames(q1.f), paste("fe", colnames(q1.m), sep=""))
```

```
## [1] TRUE
```

```

# data frame with the life table names with years
allLifetables <- read.table(text = colnames(q1.f)
                           , sep = ".", colClasses = "character")
colnames(allLifetables) <- c("sex","country","year")
allLifetables$year <- as.numeric(allLifetables$year)

# summarize life tables
country <- allLifetables[1,2]
year <- allLifetables[1,3]
ltList <- list()
ltList[[1]] <- c(country,year,"")
index <- 1
for (i in 2:nrow(allLifetables)) {
  if ((allLifetables[i,2] != country) |
      (allLifetables[i,3] != (allLifetables[(i-1),3]+1))) {
    ltList[[index]][3] <- allLifetables[(i-1),3]
    country <- allLifetables[i,2]
    year <- allLifetables[i,3]
    index <- index + 1
    ltList[[index]] <- c(country,year,"")
  }
  if(i==nrow(allLifetables)) {
    ltList[[index]][3] <- allLifetables[i,3]
  }
}
# have a look at the list of life countries with their life tables
# ltList

# create LaTeX for life table summaries

# function to parse out full country names from first line of
# HMD life table text file
parse.countries <- function(file.name) {

  con <- file(file.name,"r")
  first.line <- readLines(con,n=1)
  close(con)

  return(str_split(first.line,",")[[1]][1])
}

# read and split the list of file names from HMD
files <- Sys.glob("../data/HMD/hmd_statistics/lt_female/fltpersonal/*")
# files

# # make a list of country abbreviations and their full names
# country.names <- list()
# for (i in 1:length(files)) {
#   country.names[[i]] <- c(strsplit(basename(files[i]),"\\".')[[1]][1]
#                         ,parse.countries(files[[i]]))
# }
# # have a look at the list of full country names

```

```

# # country.names

# make a list of country abbreviations and their full names
country.names <- list()
for (i in 1:length(files)) {
  country.names[[i]] <- c(strsplit(basename(files[i])
    , "\\\\"[[1]][1],parse.countries(files[[i]])))

  # 'by hand' correction to full country names
  if (country.names[[i]][1] == "GBRTENW") {
    country.names[[i]][2] <- "England and Wales -- Total Population"
  }
  if (country.names[[i]][1] == "GBRCENW") {
    country.names[[i]][2] <- "England and Wales -- Civilian National Population"
  }
  if (country.names[[i]][1] == "FRACNP") {
    country.names[[i]][2] <- "France -- Civilian Population"
  }
  if (country.names[[i]][1] == "FRATNP") {
    country.names[[i]][2] <- "France -- Total Population"
  }
}
# have a look at the list of full country names
# country.names

# print life table summaries to local output
ltGrandTot <- 0
for (i in 1:length(ltList)) {
  for(j in 1:length(country.names)) {
    if(ltList[[i]][1] == str_to_upper(country.names[[j]][1])) {
      ltTot <- as.numeric(ltList[[i]][3])-as.numeric(ltList[[i]][2])+1
      ltGrandTot <- ltGrandTot + ltTot
      cat(country.names[[j]][1],"&",country.names[[j]][2],"&",ltList[[i]][2]
        ,"--",ltList[[i]][3],"&",ltTot," \\\\",\"\n")
    }
  }
}

## AUS & Australia & 1921 -- 2021 & 101 \\
## AUT & Austria & 1947 -- 2019 & 73 \\
## BEL & Belgium & 1841 -- 1913 & 73 \\
## BEL & Belgium & 1919 -- 2022 & 104 \\
## BGR & Bulgaria & 1947 -- 2021 & 75 \\
## BLR & Belarus & 1959 -- 2018 & 60 \\
## CAN & Canada & 1921 -- 2022 & 102 \\
## CHE & Switzerland & 1876 -- 2022 & 147 \\
## CHL & Chile & 1992 -- 2020 & 29 \\
## CZE & Czechia & 1950 -- 2021 & 72 \\
## DEUTE & East Germany & 1956 -- 2020 & 65 \\
## DEUTNP & Germany & 1990 -- 2020 & 31 \\
## DEUTW & West Germany & 1956 -- 2020 & 65 \\
## DNK & Denmark & 1835 -- 2023 & 189 \\
## ESP & Spain & 1908 -- 2021 & 114 \\
## EST & Estonia & 1959 -- 2019 & 61 \\
## FIN & Finland & 1878 -- 2023 & 146 \\

```

```

## FRACNP & France -- Civilian Population & 1816 -- 2022 & 207 \\
## FRATNP & France -- Total Population & 1816 -- 2022 & 207 \\
## GBRCEW & England and Wales -- Civilian National Population & 1841 -- 2021 & 181 \\
## GBRTEW & England and Wales -- Total Population & 1841 -- 2021 & 181 \\
## GBR_NIR & Northern Ireland & 1922 -- 2021 & 100 \\
## GBR_NP & United Kingdom & 1922 -- 2021 & 100 \\
## GBR_SCO & Scotland & 1855 -- 2021 & 167 \\
## GRC & Greece & 1981 -- 2019 & 39 \\
## HKG & Hong Kong & 1986 -- 2020 & 35 \\
## HRV & Croatia & 2001 -- 2020 & 20 \\
## HUN & Hungary & 1950 -- 2020 & 71 \\
## IRL & Ireland & 1950 -- 2020 & 71 \\
## ISL & Iceland & 1838 -- 1851 & 14 \\
## ISL & Iceland & 1853 -- 2022 & 170 \\
## ISR & Israel & 1983 -- 2016 & 34 \\
## ITA & Italy & 1872 -- 2021 & 150 \\
## JPN & Japan & 1947 -- 2022 & 76 \\
## KOR & Republic of Korea & 2003 -- 2020 & 18 \\
## LTU & Lithuania & 1959 -- 2020 & 62 \\
## LUX & Luxembourg & 1960 -- 2022 & 63 \\
## LVA & Latvia & 1959 -- 2019 & 61 \\
## NLD & Netherlands & 1850 -- 2021 & 172 \\
## NOR & Norway & 1846 -- 2023 & 178 \\
## NZL_MA & New Zealand -- Maori & 1948 -- 1948 & 1 \\
## NZL_MA & New Zealand -- Maori & 1950 -- 1955 & 6 \\
## NZL_MA & New Zealand -- Maori & 1957 -- 1958 & 2 \\
## NZL_MA & New Zealand -- Maori & 1960 -- 2008 & 49 \\
## NZL_NM & New Zealand -- Non-Maori & 1901 -- 2008 & 108 \\
## NZL_NP & New Zealand & 1948 -- 2021 & 74 \\
## POL & Poland & 1958 -- 2019 & 62 \\
## PRT & Portugal & 1940 -- 2022 & 83 \\
## RUS & Russia & 1959 -- 2014 & 56 \\
## SVK & Slovakia & 1950 -- 2019 & 70 \\
## SVN & Slovenia & 1983 -- 2019 & 37 \\
## SWE & Sweden & 1751 -- 2023 & 273 \\
## TWN & Taiwan & 1970 -- 2021 & 52 \\
## UKR & Ukraine & 1959 -- 2013 & 55 \\
## USA & The United States of America & 1933 -- 2022 & 90 \\
# create a dataframe with the country names, abbreviations, and number of
# consecutive life tables
ltGrandTot <- 0
life.tables <- data.frame(
  "country" = character(length(ltList)),
  "abbreviation" = character(length(ltList)),
  "startYear" = numeric(length(ltList)),
  "stopYear" = numeric(length(ltList)),
  "tables" = numeric(length(ltList)),
  ,stringsAsFactors = FALSE
)
for (i in 1:length(ltList)) {
  for(j in 1:length(country.names)) {
    if(ltList[[i]][1] == str_to_upper(country.names[[j]][1])) {
      ltTot <- as.numeric(ltList[[i]][3])-as.numeric(ltList[[i]][2])+1
      ltGrandTot <- ltGrandTot+ltTot
    }
  }
}

```

```

        ltGrandTot <- ltGrandTot + ltTot
        life.tables$abbreviation[i] <- as.character(country.names[[j]][1])
        life.tables$country[i] <- as.character(country.names[[j]][2])
        life.tables$startYear[i] <- as.numeric(ltList[[i]][2])
        life.tables$stopYear[i] <- as.numeric(ltList[[i]][3])
        life.tables$tables[i] <- as.numeric(ltTot)
    }
}
}

lts.print <- cbind(life.tables[,c(1,2)]
    ,paste(life.tables[,3],"--",life.tables[,4],sep="")
    ,life.tables[,5])

# save rows of table summarizing life tables in a text file
print(file="../tables/LTSummaries.txt"
    ,xtable(lts.print
        ,booktabs=TRUE,digits=0)
    ,only.contents = TRUE,include.rownames = FALSE
    ,include.colnames = FALSE,hline.after=NULL)

# save the number of life tables for each sex
capture.output(file="../tables/LTtot.txt"
    ,cat(format(ltGrandTot, nsmall=0, big.mark=",")))

# save the number of life tables for both sexes
capture.output(file="../tables/LTtotBoth.txt"
    ,cat(format(2*ltGrandTot, nsmall=0, big.mark=",")))

# save the download date
capture.output(file="../tables/HMDdate.txt",cat(data.date))

print("")

## [1] ""

print(paste("Total number of life tables in raw data (q1.f):"
    ,length(colnames(q1.f)))))

## [1] "Total number of life tables in raw data (q1.f): 4902"
print(paste("Check, totalling up country counts of life tables:",ltGrandTot))

## [1] "Check, totalling up country counts of life tables: 4902"
print("")

## [1] ""

print("Four 'by-hand' corrections")

## [1] "Four 'by-hand' corrections"
print("GBRTENW: England and Wales - Total Population")

## [1] "GBRTENW: England and Wales - Total Population"

```

```

print("GBRCENW: England and Wales - Civilian National Population")

## [1] "GBRCENW: England and Wales - Civilian National Population"
print("FRACNP: France - Civilian Population")

## [1] "FRACNP: France - Civilian Population"
print("FRATNP: France - Total Population")

## [1] "FRATNP: France - Total Population"
rm(list=c("country","year","files","i","j","index"))

```

Make lines that describe the sum of squares explained by each SVD component.

```

# sum of squares explained by each component is the square of the
# corresponding singular value

# calculate the ss for females
ss.1.f <- (mod.1_0.f$svd$s1$d^2)[1]/sum(mod.1_0.f$svd$s1$d^2)
ss.2.f <- (mod.1_0.f$svd$s1$d^2)[2]/sum(mod.1_0.f$svd$s1$d^2)
ss.3.f <- (mod.1_0.f$svd$s1$d^2)[3]/sum(mod.1_0.f$svd$s1$d^2)
ss.4.f <- (mod.1_0.f$svd$s1$d^2)[4]/sum(mod.1_0.f$svd$s1$d^2)
ss.1to4.f <- format(round(sum((mod.1_0.f$svd$s1$d^2)[1:4])
                         /sum(mod.1_0.f$svd$s1$d^2),6),format="d")
ss.f <- format(round(c(ss.1.f,ss.2.f,ss.3.f,ss.4.f),6),format="d")
# write the line for males
capture.output(file="../tables/ssFullFemale.txt",
cat(paste(paste(ss.f[1:3],collapse=", "),sep=""),", and ",ss.f[4],sep=""))
)
capture.output(file="../tables/ssFullFemaleTotal.txt",
cat(ss.1to4.f)
)

# calculate the ss for males
ss.1.m <- (mod.1_0.m$svd$s1$d^2)[1]/sum(mod.1_0.m$svd$s1$d^2)
ss.2.m <- (mod.1_0.m$svd$s1$d^2)[2]/sum(mod.1_0.m$svd$s1$d^2)
ss.3.m <- (mod.1_0.m$svd$s1$d^2)[3]/sum(mod.1_0.m$svd$s1$d^2)
ss.4.m <- (mod.1_0.m$svd$s1$d^2)[4]/sum(mod.1_0.m$svd$s1$d^2)
ss.1to4.m <- format(round(sum((mod.1_0.m$svd$s1$d^2)[1:4])
                           /sum(mod.1_0.m$svd$s1$d^2),6),format="d")
ss.m <- format(round(c(ss.1.m,ss.2.m,ss.3.m,ss.4.m),6),format="d")
# write the line for males
capture.output(file="../tables/ssFullMale.txt",
cat(paste(paste(ss.m[1:3],collapse=", "),sep=""),", and ",ss.m[4],sep=""))
)
capture.output(file="../tables/ssFullMaleTotal.txt",
cat(ss.1to4.m)
)

# calculate fractions of 2+ component ss explained by
# components 2-4

# female
ss.2plus.tot.f <- sum(mod.1_0.f$svd$s1$d[2:length(mod.1_0.f$svd$s1$d)]^2)

```

```

ss.2plus.2.f <- (mod.1_0.f$svd$s1$d^2)[2]/ss.2plus.tot.f
ss.2plus.3.f <- (mod.1_0.f$svd$s1$d^2)[3]/ss.2plus.tot.f
ss.2plus.4.f <- (mod.1_0.f$svd$s1$d^2)[4]/ss.2plus.tot.f
ss.2plus.f <- c(ss.2plus.2.f,ss.2plus.3.f,ss.2plus.4.f)
ss.2plus.format.f <- format(round(ss.2plus.f,6),format="d")
ss.2plus.tot.f <- format(round(sum(ss.2plus.f),6),format="d")
capture.output(file="../tables/ssFemale.txt",
cat(paste(paste(ss.2plus.format.f[1:2],collapse=", "),
           ,sep=""),", and ",ss.2plus.format.f[3],sep=""))
)
capture.output(file="../tables/ssFemaleTotal.txt",
cat(ss.2plus.tot.f)
)
# male
ss.2plus.tot.m <- sum(mod.1_0.m$svd$s1$d[2:length(mod.1_0.m$svd$s1$d)]^2)
ss.2plus.2.m <- (mod.1_0.m$svd$s1$d^2)[2]/ss.2plus.tot.m
ss.2plus.3.m <- (mod.1_0.m$svd$s1$d^2)[3]/ss.2plus.tot.m
ss.2plus.4.m <- (mod.1_0.m$svd$s1$d^2)[4]/ss.2plus.tot.m
ss.2plus.m <- c(ss.2plus.2.m,ss.2plus.3.m,ss.2plus.4.m)
ss.2plus.format.m <- format(round(ss.2plus.m,6),format="d")
ss.2plus.tot.m <- format(round(sum(ss.2plus.m),6),format="d")
# write the line for males
capture.output(file="../tables/ssMale.txt",
cat(paste(paste(ss.2plus.format.m[1:2],collapse=", "),
           ,sep=""),", and ",ss.2plus.format.m[3],sep=""))
)
capture.output(file="../tables/ssMaleTotal.txt",
cat(ss.2plus.tot.m)
)

```

Make table of summary comparison results.

```

# the summary comparisons are stored in comps. ... objects
comps.child$female

##      total.abs.error mean.abs.error max.error
## comp        1533.818     0.01360419 0.3303233
## lq         1601.868     0.01420776 0.3968440

comps.child$male

##      total.abs.error mean.abs.error max.error
## comp        1771.413     0.01571154 0.3993513
## lq         1905.930     0.01690463 0.3792040

cat("\n")

comps.adult$female

##      total.abs.error mean.abs.error max.error
## comp        1368.131     0.01213463 0.2185885
## lq         1493.637     0.01324780 0.3865020

comps.adult$male

##      total.abs.error mean.abs.error max.error
## comp        1451.754     0.01287632 0.4037572

```

```

## lq          1574.161    0.01396201 0.3532550
# make lines for the table

# female
# make code more readable ...
# a, c
# b, d
a.f <- comps.child$female[1,1] # child-only SVD-Comp
b.f <- comps.child$female[2,1] # child-only Log-Quad
c.f <- comps.adult$female[1,1] # child/adult SVD-Comp
d.f <- comps.adult$female[2,1] # child/adult Log-Quad
# write the few lines
capture.output(file= "../tables/compsFemale.txt",
  cat(paste("R1 & SVD-Comp &", format(round(a.f,0),big.mark=",",), "&" ,
    format(round(c.f,0),big.mark=",",), "&" ,
    format(round(c.f - a.f,0), big.mark=",",)
    , " \\\\", sep=" "), "\n"),
  cat(paste("R2 & Log-Quad &", format(round(b.f,0),big.mark=",",), "&" ,
    format(round(d.f,0),big.mark=",",), "&" ,
    format(round(d.f - b.f,0), big.mark=",",)
    , " \\\\", sep=" "), "\n"),
  cat(paste("R3 & R2-R1 &", format(round(
    b.f - a.f,0)), "&", format(round(d.f - c.f,0)), "&" ,
    format(round((d.f - b.f) - (c.f - a.f),0))
    , " \\\\", sep=" "), "\n"),
  cat(paste("R4 & R3/R1 (\%) &", format(round(100*
    (b.f - a.f)/a.f,1),nsmall=1), "&" ,
    format(round(100*(d.f - c.f)/ c.f,1),nsmall=1), "&" ,
    format(round(100*((d.f - b.f) - (c.f - a.f)) / (c.f - a.f),1),nsmall=1)
    , " \\\\", sep=" "), "\n")
)

# male
# make code more readable ...
# a, c
# b, d
a.m <- comps.child$male[1,1] # child-only SVD-Comp
b.m <- comps.child$male[2,1] # child-only Log-Quad
c.m <- comps.adult$male[1,1] # child/adult SVD-Comp
d.m <- comps.adult$male[2,1] # child/adult Log-Quad
# write the few lines
capture.output(file= "../tables/compsMale.txt",
  cat(paste("R1 & SVD-Comp &", format(round(a.m,0),big.mark=",",), "&" ,
    format(round(c.m,0),big.mark=",",), "&" ,
    format(round(c.m - a.m,0), big.mark=",",)
    , " \\\\", sep=" "), "\n"),
  cat(paste("R2 & Log-Quad &", format(round(b.m,0),big.mark=",",), "&" ,
    format(round(d.m,0),big.mark=",",), "&" ,
    format(round(d.m - b.m,0), big.mark=",",)
    , " \\\\", sep=" "), "\n"),
  cat(paste("R3 & R2-R1 &", format(round(
    b.m - a.m,0)), "&", format(round(d.m - c.m,0)), "&" ,
    format(round((d.m - b.m) - (c.m - a.m),0))
    , " \\\\", sep=" "), "\n")
)

```

```

    , " \\\\", sep=" "), "\n"),
cat(paste("R4 & R3/R1 (\%\% ) &", format(round(100*
(b.m - a.m)/a.m,1),nsmall=1), "&"
, format(round(100*(d.m - c.m)/ c.m,1),nsmall=1), "&"
, format(round(100*((d.m - b.m) - (c.m - a.m)) / (c.m - a.m),1),nsmall=1)
, " \\\\", sep=" "), "\n")
)

```

Make nice LaTeX tables from the regression models that are part of SVD-Comp. Don't worry about the warnings *Stargazer* raises.

```

# adult mortality model
capture.output(file="../tables/adultMortality.txt",
stargazer(
  mod.1_0.f$mods$s1$aml,
  mod.1_0.m$mods$s1$aml,
  title="Adult Mortality Models: $\\logit(\\qff_z \\ell) = f(\\qf_{\\}, z \\ell)$",
  label="tab:appA:adultMxMod",
  dep.var.labels.include = FALSE,
  dep.var.caption = "$\\logit(\\qff)$",
  model.numbers = FALSE,
  column.labels=c("Female","Male"),
  covariate.labels=c("$\\qf$","$\\mbox{logit}(\\qf)$"
                    ,"$\\mbox{logit}(\\qf)^2$","$\\mbox{logit}(\\qf)^3$"),
  omit.stat=c("LL","ser"),
  single.row = TRUE
))

# infant mortality
capture.output(file="../tables/infantMortality.txt",
stargazer(
  mod.1_0.f$mods$s1$q0,
  mod.1_0.m$mods$s1$q0,
  title="Infant Mortality Models: $\\logit(\\qoz_z \\ell) = f(\\qf_z, z \\ell)$",
  label="tab:appA:infantMxMod",
  dep.var.labels.include = FALSE,
  dep.var.caption = "$\\logit(\\qoz)$",
  model.numbers = FALSE,
  column.labels=c("Female","Male"),
  covariate.labels=c("$\\mbox{logit}(\\qf)$","$\\mbox{logit}(\\qf)^2$"),
  omit.stat=c("LL","ser"),
  single.row = TRUE
))

# vs - female
mods <- list(
  mod.1_0.f$mods$s1$v1,
  mod.1_0.f$mods$s1$v2,
  mod.1_0.f$mods$s1$v3,
  mod.1_0.f$mods$s1$v4
)
capture.output(file="../tables/vsFemale.txt",
stargazer(mods,
  title="Female RSV Models: $v_{\\ell i} = f_{i}(\\qf_{\\}, \\ell), \\qff_{\\ell i}$",

```

```

label="tab:appA:femaleRSVMods",
dep.var.labels.include = FALSE,
dep.var.caption = "Right Singular Vector Elements",
model.numbers = FALSE,
column.labels = c("$\\mbf{v}_1$","$\\mbf{v}_2$",
                 "$\\mbf{v}_3$","$\\mbf{v}_4$"),
covariate.labels=c(
  "$\\qf$",
  "$\\mbox{logit}(\\qf)$",
  "$\\mbox{logit}(\\qf)^2$",
  "$\\mbox{logit}(\\qf)^3$",
  "$\\qff$",
  "$\\mbox{logit}(\\qff)^2$",
  "$\\mbox{logit}(\\qff)^3$",
  "$\\qf \\times \\qff$"
),
omit.stat=c("LL","ser")
))

# vs - male
mods <- list(
  mod.1_0$mods$s1$v1,
  mod.1_0$mods$s1$v2,
  mod.1_0$mods$s1$v3,
  mod.1_0$mods$s1$v4
)
capture.output(file="..../tables/vsMale.txt",
stargazer(mods,
  title="Male RSV Models: $v_{\\ell_i} = f_i(\\qf_{\\ell}, \\qff_{\\ell})$",
  label="tab:appA:maleRSVMods",
  dep.var.labels.include = FALSE,
  dep.var.caption = "Right Singular Vector Elements",
  model.numbers = FALSE,
  column.labels = c("$\\mbf{v}_1$","$\\mbf{v}_2$",
                 "$\\mbf{v}_3$","$\\mbf{v}_4$"),
  covariate.labels=c(
    "$\\qf$",
    "$\\mbox{logit}(\\qf)$",
    "$\\mbox{logit}(\\qf)^2$",
    "$\\mbox{logit}(\\qf)^3$",
    "$\\qff$",
    "$\\mbox{logit}(\\qff)^2$",
    "$\\mbox{logit}(\\qff)^3$",
    "$\\qf \\times \\qff$"
),
  omit.stat=c("LL","ser")
))

```

Create lines for age-specific error tables. These compare the l_x -weighted age-specific total absolute error (tae) in prediction between SVD-Comp and Log Quad. The coding strategy is to write a couple functions to automate this set of calculations so that it can be repeated several times later using different numbers of components in the SVD-Comp predictions. The first function *lthat()* creates full life tables from the SVD-Comp predictions – so that we can get age-specific expectations of life, e_x . The second function *ageSpecificErrorComparisons()* used the first and actually calculates the age-weighted prediction errors and

their differences and organizes them into a nice return object.

```
# function to calculate life table from matrix of 1qx
lthat <- function(q,sex,a1.f,a1.m) {
  # calculate lx
  zeroes <- matrix(0,nrow=(nrow(q)+1),ncol=ncol(q))
  l <- zeroes
  l[1,] <- 100000 # l0 = 100000
  # loop through ages and calculate lx
  for (i in 2:nrow(l)) {
    l[i,] <- l[(i-1),]*(1-q[(i-1),])
  }
  # calculate Lx
  L <- zeroes
  # loop through ages and calculate Lx
  for (i in 1:(nrow(l)-1)) {
    L[i,] <- l[(i+1),] + ifelse(str_to_lower(sex)=="female",
                                   a1.f[i,],a1.m[i,]) * (l[i,]-l[(i+1),])
  }
  L[nrow(L),] <- ifelse(str_to_lower(sex)=="female",
                         a1.f[nrow(L),],a1.m[nrow(L),]) * l[nrow(L),]
  # calculate Tx
  T <- zeroes
  for (i in 1:(nrow(l)-1)) {
    T[i,] <- colSums(L[(i:nrow(T)),])
  }
  T[nrow(T),] <- L[nrow(T),]
  # calculate ex
  e <- T/l
  lt <- list(qx=q,lx=l,Lx=L,Tx=T,ex=e)
  return(lt)
}

# function to conduct age-specific comparisons
#   of prediction errors between SVD-Comp and Log Quad
ageSpecificErrorComparisons <- function(mod.f,mod.m,lt.lq,q,l,e,a1.f,a1.m) {

  # mod.f is svdMod() return object for females
  # mod.m is svdMod() return object for males
  # lt.q is object with q, e, l columns
  #   from Log Quad predictions, five-year age groups
  # q is input HMD life table 5qx columns
  # l is input HMD life table lx columns, five-year age groups
  # e is input HMD life table e columns, five-year age groups

  # create five-year age groups of predicted values
  # female

  # female
  qp.f <- expit(mod.f$recon.samp$s1)
  q5p.f <- convert1qxTo5qxApply(qp.f)
  # male
  qp.m <- expit(mod.m$recon.samp$s1)
  q5p.m <- convert1qxTo5qxApply(qp.m)
```

```

# predicted life tables at five-year age group start ages
# female
lt.comp.f <- lthat(qp.f,"Female",a1.f,a1.m) # female life tables
lt.comp.f.qx <- q5p.f
lt.comp.f.lx <- lt.comp.f$lx[c(1,2,seq(6,111,5)),]
lt.comp.f.ex <- lt.comp.f$ex[c(1,2,seq(6,111,5)),]

# male
lt.comp.m <- lthat(qp.m,"Male",a1.f,a1.m) # male life tables
lt.comp.m.qx <- q5p.m
lt.comp.m.lx <- lt.comp.m$lx[c(1,2,seq(6,111,5)),]
lt.comp.m.ex <- lt.comp.m$ex[c(1,2,seq(6,111,5)),]

# log quad predicted life tables
# female
lt.lq.f.qx <- lt.lq$q5.lq.f
lt.lq.f.lx <- lt.lq$l5.lq.f
lt.lq.f.ex <- lt.lq$e5.lq.f
# male
lt.lq.m.qx <- lt.lq$q5.lq.m
lt.lq.m.lx <- lt.lq$l5.lq.m
lt.lq.m.ex <- lt.lq$e5.lq.m

# age-schedule of weights based on HMD lx values
# female
weights.f <- rowSums(l$15.f)/sum(rowSums(l$15.f))
# sum(weight.f)
# male
weights.m <- rowSums(l$15.m)/sum(rowSums(l$15.m))
# sum(weight.m)

# sum age-specific absolute errors in 5qx

# female
tae.comp.q.f <- rowSums(abs(lt.comp.f.qx-q$q5.f)) * weights.f[1:23]
tae.lq.q.f <- rowSums(abs(lt.lq.f.qx-q$q5.f)) * weights.f[1:23]
tae.diff.q.f <- tae.comp.q.f-tae.lq.q.f
# male
tae.comp.q.m <- rowSums(abs(lt.comp.m.qx-q$q5.m)) * weights.m[1:23]
tae.lq.q.m <- rowSums(abs(lt.lq.m.qx-q$q5.m)) * weights.m[1:23]
tae.diff.q.m <- tae.comp.q.m-tae.lq.q.m

# store it all
tae.q <- cbind(tae.comp.q.f,tae.lq.q.f
                 ,tae.diff.q.f,tae.comp.q.m,tae.lq.q.m,tae.diff.q.m)
tae.q <- rbind(tae.q,colSums(tae.q))

# sum age-specific absolute errors in ex

# female
tae.comp.e.f <- rowSums(abs(lt.comp.f.ex-e$e5.f)) * weights.f
tae.lq.e.f <- rowSums(abs(lt.lq.f.ex-e$e5.f)) * weights.f
tae.diff.e.f <- tae.comp.e.f-tae.lq.e.f
# male

```

```

tae.comp.e.m <- rowSums(abs(lt.comp.m.ex-e$e5.m)) * weights.m
tae.lq.e.m <- rowSums(abs(lt.lq.m.ex-e$e5.m)) * weights.m
tae.diff.e.m <- tae.comp.e.m-tae.lq.e.m

# store it all
tae.e <- cbind(tae.comp.e.f,tae.lq.e.f
                 ,tae.diff.e.f,tae.comp.e.m,tae.lq.e.m,tae.diff.e.m)
tae.e <- rbind(tae.e,colSums(tae.e))

# total absolute error in e0
# female
tot.tae.comp.e0.f <- sum(abs(lt.comp.f.ex[1,]-e$e5.f[1,]))
tot.tae.lq.e0.f <- sum(abs(lt.lq.f.ex[1,]-e$e5.f[1,]))
tot.tae.diff.e0.f <- tot.tae.comp.e0.f-tot.tae.lq.e0.f
# male
tot.tae.comp.e0.m <- sum(abs(lt.comp.m.ex[1,]-e$e5.m[1,]))
tot.tae.lq.e0.m <- sum(abs(lt.lq.m.ex[1,]-e$e5.m[1,]))
tot.tae.diff.e0.m <- tot.tae.comp.e0.m-tot.tae.lq.e0.m

# have a look at it all
tot.tae.e0 <- rbind(c(tot.tae.comp.e0.f
                       ,tot.tae.lq.e0.f,tot.tae.diff.e0.f)
                     ,c(tot.tae.comp.e0.m,tot.tae.lq.e0.m
                        ,tot.tae.diff.e0.m))
rownames(tot.tae.e0) <- c("Female","Male")

return(list(
  tae.q = tae.q
  ,tae.e = tae.e
  ,tot.tae.e0 = tot.tae.e0
))
}

# create list of five-year age group q, e, and l columns
# from Log Quad predictions conducted earlier
lt.lq <- list(
  q5.lq.f = comps.child$q5.lq.f
  ,e5.lq.f = comps.child$e5.lq.f
  ,l5.lq.f = comps.child$l5.lq.f
  ,q5.lq.m = comps.child$q5.lq.m
  ,e5.lq.m = comps.child$e5.lq.m
  ,l5.lq.m = comps.child$l5.lq.m
)

# create list of 5qx (five-year age groups) from HMD life tables
q <- list(
  q5.f = q5.f
  ,q5.m = q5.m
)

# create list of lx (five-year age groups) from HMD life tables
l <- list(

```

```

    15.f = 15.f
    ,15.m = 15.m
)

# create list of ex (five-year age groups) from HMD life tables
e <- list(
  e5.f = e5.f
  ,e5.m = e5.m
)

# calculate age-specific comparisons in prediction errors
age.comps <- ageSpecificErrorComparisons(
  mod.1_0.f,mod.1_0.m,lt.lq,q,l,e,a1.f,a1.m)
# have a look
age.comps

## $tae.q
##           tae.comp.q.f   tae.lq.q.f   tae.diff.q.f
## 0        1.274576735  1.297233240 -0.0226565048
## 1-4      1.408642198  1.289036012  0.1196061863
## 5-9      0.768009199  0.736492559  0.0315166396
## 10-14     0.502821047  0.486823749  0.0159972985
## 15-19     0.626049559  0.704876573 -0.0788270136
## 20-24     0.768516703  0.857060707 -0.0885440042
## 25-29     0.763305793  0.848098118 -0.0847923259
## 30-34     0.756275811  0.804742161 -0.0484663507
## 35-39     0.836650885  0.844481623 -0.0078307374
## 40-44     0.961101183  0.936995930  0.0241052531
## 45-49     1.124972727  1.120028613  0.0049441141
## 50-54     1.478516483  1.490918543 -0.0124020603
## 55-59     1.945855361  1.979596056 -0.0337406952
## 60-64     2.512083891  2.616680093 -0.1045962025
## 65-69     3.108742498  3.254985725 -0.1462432268
## 70-74     4.081456235  4.271187273 -0.1897310380
## 75-79     4.841454736  4.926832431 -0.0853776953
## 80-84     4.740783508  4.904941882 -0.1641583735
## 85-89     3.495277867  3.511496470 -0.0162186029
## 90-94     1.754908944  1.787401665 -0.0324927218
## 95-99     0.466763887  0.498031026 -0.0312671395
## 100-104    0.059340397  0.067499575 -0.0081591779
## 105-109    0.003696479  0.004375082 -0.0006786026
##          38.279802127 39.239815108 -0.9600129814
##           tae.comp.q.m   tae.lq.q.m   tae.diff.q.m
## 0        1.509600915  1.526154490 -0.0165535749
## 1-4      2.002035933  1.536224082  0.4658118505
## 5-9      0.867411985  0.863027711  0.0043842741
## 10-14     0.515046975  0.486713551  0.0283334247
## 15-19     0.890692991  0.860015651  0.0306773406
## 20-24     1.688429365  1.643764536  0.0446648286
## 25-29     1.568435038  1.523449996  0.0449850427
## 30-34     1.509058761  1.479942826  0.0291159355
## 35-39     1.691430711  1.654222685  0.0372080269
## 40-44     1.987269079  1.952716291  0.0345527880
## 45-49     2.416185314  2.408272935  0.0079123789

```

```

## 50-54    2.998635037  3.064041949 -0.0654069116
## 55-59    3.594312689  3.794220593 -0.1999079037
## 60-64    4.334740268  4.740872963 -0.4061326950
## 65-69    4.843664238  5.348903581 -0.5052393425
## 70-74    5.100618529  5.727486555 -0.6268680259
## 75-79    4.764436021  5.222380347 -0.4579443256
## 80-84    3.528247334  3.843713007 -0.3154656731
## 85-89    2.007799919  2.090101282 -0.0823013636
## 90-94    0.814367488  0.856392989 -0.0420255011
## 95-99    0.163226183  0.184202976 -0.0209767935
## 100-104   0.017300338  0.020634919 -0.0033345806
## 105-109   0.001013241  0.001229015 -0.0002157742
##          48.813958352  50.828684927 -2.0147265750
##
## $tae.e
##           tae.comp.e.f   tae.lq.e.f   tae.diff.e.f
## 0          4.351343e+02  4.488274e+02 -1.369309e+01
## 1-4        4.746635e+02  4.927074e+02 -1.804391e+01
## 5-9        4.393281e+02  4.589434e+02 -1.961530e+01
## 10-14      4.182012e+02  4.375600e+02 -1.935875e+01
## 15-19      4.045099e+02  4.220134e+02 -1.750353e+01
## 20-24      3.850596e+02  3.997308e+02 -1.467120e+01
## 25-29      3.636798e+02  3.761462e+02 -1.246641e+01
## 30-34      3.446863e+02  3.563449e+02 -1.165858e+01
## 35-39      3.285306e+02  3.408473e+02 -1.231670e+01
## 40-44      3.121311e+02  3.256429e+02 -1.351182e+01
## 45-49      2.937892e+02  3.073443e+02 -1.355516e+01
## 50-54      2.718690e+02  2.847086e+02 -1.283961e+01
## 55-59      2.461976e+02  2.575757e+02 -1.137808e+01
## 60-64      2.165749e+02  2.266830e+02 -1.010808e+01
## 65-69      1.840224e+02  1.912519e+02 -7.229439e+00
## 70-74      1.475940e+02  1.528070e+02 -5.212995e+00
## 75-79      1.074493e+02  1.109866e+02 -3.537286e+00
## 80-84      6.869363e+01  7.084205e+01 -2.148418e+00
## 85-89      3.637768e+01  3.683845e+01 -4.607739e-01
## 90-94      1.489748e+01  1.493034e+01 -3.286166e-02
## 95-99      3.982213e+00  3.971052e+00  1.116088e-02
## 100-104    5.855478e-01  5.839672e-01  1.580673e-03
## 105-109    4.490054e-02  4.462715e-02  2.733874e-04
## 110+       6.971239e-03  2.913201e-03  4.058038e-03
##          5.498009e+03  5.717334e+03 -2.193249e+02
##           tae.comp.e.m   tae.lq.e.m   tae.diff.e.m
## 0          6.510078e+02  6.895783e+02 -3.857054e+01
## 1-4        6.893637e+02  7.181268e+02 -2.876309e+01
## 5-9        6.410171e+02  6.906145e+02 -4.959741e+01
## 10-14      6.238369e+02  6.716764e+02 -4.783954e+01
## 15-19      6.136538e+02  6.610412e+02 -4.738741e+01
## 20-24      5.913498e+02  6.374057e+02 -4.605597e+01
## 25-29      5.519158e+02  5.983426e+02 -4.642676e+01
## 30-34      5.164862e+02  5.648200e+02 -4.833383e+01
## 35-39      4.821291e+02  5.316529e+02 -4.952381e+01
## 40-44      4.454383e+02  4.945496e+02 -4.911132e+01
## 45-49      4.041177e+02  4.503434e+02 -4.622569e+01
## 50-54      3.564159e+02  3.985604e+02 -4.214441e+01

```

```

## 55-59  3.025450e+02 3.390018e+02 -3.645682e+01
## 60-64  2.442199e+02 2.743171e+02 -3.009726e+01
## 65-69  1.839504e+02 2.060942e+02 -2.214377e+01
## 70-74  1.274482e+02 1.417105e+02 -1.426227e+01
## 75-79  7.842901e+01 8.535009e+01 -6.921080e+00
## 80-84  4.196632e+01 4.420907e+01 -2.242746e+00
## 85-89  1.885919e+01 1.910333e+01 -2.441372e-01
## 90-94  6.734159e+00 6.675621e+00  5.853751e-02
## 95-99  1.456355e+00 1.449960e+00  6.394490e-03
## 100-104 1.805199e-01 1.819738e-01 -1.453961e-03
## 105-109 1.289934e-02 1.288794e-02  1.140421e-05
## 110+    3.122697e-03 1.076756e-03  2.045942e-03
##           7.572537e+03 8.224819e+03 -6.522823e+02
##
## $tot.tae.e0
##          [,1]      [,2]      [,3]
## Female  6581.451 6788.56 -207.1093
## Male    9166.612 9709.71 -543.0982
# create lines for tables

print(file="../tables/ageCompQ-1.txt"
      ,xtable(age.comps$tae.q[(1:nrow(age.comps$tae.q)-1),]
              ,booktabs=TRUE,digits=4)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL
      ,format.args=list(big.mark = ","))

capture.output(file="../tables/ageCompQ-2.txt",
cat(paste("0-109 & ",paste(format(round(age.comps$tae.q[nrow(age.comps$tae.q),],4)
                                         ,format="d",big.mark=",",collapse=" & "),"\\""\",sep=""))
)

print(file="../tables/ageCompE-1.txt"
      ,xtable(age.comps$tae.e[(1:nrow(age.comps$tae.e)-1),]
              ,booktabs=TRUE,digits=2)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL
      ,format.args=list(big.mark = ","))

capture.output(file="../tables/ageCompE-2.txt",
cat(paste("0+ & ",paste(format(round(age.comps$tae.e[nrow(age.comps$tae.e),],2)
                                         ,format="d",big.mark=",",collapse=" & "),"\\""\",sep=""))
)

print(file="../tables/ageCompTot.txt"
      ,xtable(age.comps$tot.tae.e0
              ,booktabs=TRUE,digits=2)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL
      ,format.args=list(big.mark = ",")))

```

Create lines for tables with scaled component values.

```

# calculate the scaled components
su.f <- mod.1_0.f$svd$s1$u %*% diag(mod.1_0.f$svd$s1$d)
su.m <- mod.1_0.m$svd$s1$u %*% diag(mod.1_0.m$svd$s1$d)
# first 4 components of both
su <- cbind(seq(0,109,1),su.f[,1:4],su.m[,1:4])
# make the table rows
print(file="../tables/us.txt"
      ,xtable(su,booktabs=TRUE,digits=c(0,0,2,2,2,2,2,2,2,2))
      ,only.contents = TRUE,include.rownames = FALSE
      ,include.colnames = FALSE,hline.after=NULL)

```

Conduct age-specific error comparison using 1–4 components. To do this, rerun the models with *svdMod()* asking for 1–4 components, and then recalculate the error comparisons for each of those models. These results are for discussion in text only, no tables produced.

```

# 1 component
# re-run models with 1 component
adult.1 <- FALSE
smooth.1 <- FALSE
N.1 <- 1
S.1 <- 1
C.1 <- 1
# base model
mod.1_0.m.1 <- svdMod(q1logit.m,Qlogit.m,N.1,S.1,10,TRUE,adult.1,TRUE,smooth.1,C.1)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "1 components"
mod.1_0.f.1 <- svdMod(q1logit.f,Qlogit.f,N.1,S.1,10,TRUE,adult.1,TRUE,smooth.1,C.1)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "1 components"

# calculate age-specific comparisons in prediction errors
# use the lt.q,q,l, and e from above
age.comps.1 <- ageSpecificErrorComparisons(mod.1_0.f.1,mod.1_0.m.1,lt.lq,q,l,e,a1.f,a1.m)
# have a look
cat("\n\n")

age.comps.1

## $tae.q
##          tae.comp.q.f   tae.lq.q.f tae.diff.q.f
## 0        1.27457674  1.297233240 -0.022656505
## 1-4      6.57562339  1.289036012  5.286587382
## 5-9      1.97720149  0.736492559  1.240708926
## 10-14    1.05962735  0.486823749  0.572803605
## 15-19    1.11442380  0.704876573  0.409547226
## 20-24    1.37684482  0.857060707  0.519784111

```

```

## 25-29      1.42291534  0.848098118  0.574817219
## 30-34      1.27048128  0.804742161  0.465739123
## 35-39      1.00421493  0.844481623  0.159733312
## 40-44      0.99072115  0.936995930  0.053725223
## 45-49      1.97435053  1.120028613  0.854321919
## 50-54      3.23673689  1.490918543  1.745818345
## 55-59      4.78067234  1.979596056  2.801076289
## 60-64      6.22139777  2.616680093  3.604717672
## 65-69      8.26914971  3.254985725  5.014163980
## 70-74      10.18460697 4.271187273  5.913419693
## 75-79      11.92475110 4.926832431  6.997918665
## 80-84      11.76543755  4.904941882  6.860495668
## 85-89      8.63265816  3.511496470  5.121161688
## 90-94      4.36016253  1.787401665  2.572760861
## 95-99      1.22569815  0.498031026  0.727667128
## 100-104    0.16981994  0.067499575  0.102320363
## 105-109    0.01123931  0.004375082  0.006864227
##          90.82331123 39.239815108 51.583496120
##          tae.comp.q.m   tae.lq.q.m   tae.diff.q.m
## 0          1.509600915 1.526154490 -0.016553575
## 1-4        7.682259011 1.536224082 6.146034929
## 5-9        2.251099316 0.863027711 1.388071606
## 10-14      1.054126013 0.486713551 0.567412463
## 15-19      1.019991121 0.860015651 0.159975471
## 20-24      1.869221906 1.643764536 0.225457370
## 25-29      1.736270799 1.523449996 0.212820803
## 30-34      1.634670281 1.479942826 0.154727455
## 35-39      1.707247752 1.654222685 0.053025067
## 40-44      2.024979275 1.952716291 0.072262984
## 45-49      2.909251012 2.408272935 0.500978077
## 50-54      4.481427641 3.064041949 1.417385692
## 55-59      6.711311578 3.794220593 2.917090985
## 60-64      8.862630934 4.740872963 4.121757970
## 65-69      10.705087162 5.348903581 5.356183581
## 70-74      11.059713378 5.727486555 5.332226823
## 75-79      10.121355260 5.222380347 4.898974913
## 80-84      7.536165909 3.843713007 3.692452902
## 85-89      4.321789485 2.090101282 2.231688203
## 90-94      1.771725497 0.856392989 0.915332508
## 95-99      0.373822926 0.184202976 0.189619949
## 100-104    0.041178708 0.020634919 0.020543789
## 105-109    0.002457824 0.001229015 0.001228809
##          91.387383702 50.828684927 40.558698775
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f   tae.diff.e.f
## 0          7.840091e+02 4.488274e+02 3.351817e+02
## 1-4        8.290106e+02 4.927074e+02 3.363032e+02
## 5-9        5.748015e+02 4.589434e+02 1.158581e+02
## 10-14      5.415854e+02 4.375600e+02 1.040254e+02
## 15-19      5.430361e+02 4.220134e+02 1.210227e+02
## 20-24      5.536301e+02 3.997308e+02 1.538993e+02
## 25-29      5.804364e+02 3.761462e+02 2.042902e+02
## 30-34      6.143625e+02 3.563449e+02 2.580176e+02

```

```

## 35-39  6.432603e+02 3.408473e+02 3.024130e+02
## 40-44  6.580821e+02 3.256429e+02 3.324392e+02
## 45-49  6.509939e+02 3.073443e+02 3.436495e+02
## 50-54  6.152632e+02 2.847086e+02 3.305545e+02
## 55-59  5.614365e+02 2.575757e+02 3.038608e+02
## 60-64  4.920718e+02 2.266830e+02 2.653888e+02
## 65-69  4.179485e+02 1.912519e+02 2.266967e+02
## 70-74  3.364915e+02 1.528070e+02 1.836845e+02
## 75-79  2.542928e+02 1.109866e+02 1.433063e+02
## 80-84  1.718868e+02 7.084205e+01 1.010447e+02
## 85-89  9.594199e+01 3.683845e+01 5.910354e+01
## 90-94  4.041127e+01 1.493034e+01 2.548092e+01
## 95-99  1.094676e+01 3.971052e+00 6.975703e+00
## 100-104 1.663149e+00 5.839672e-01 1.079182e+00
## 105-109 1.301837e-01 4.462715e-02 8.555655e-02
## 110+   6.971239e-03 2.913201e-03 4.058038e-03
##          9.971699e+03 5.717334e+03 4.254365e+03
##          tae.comp.e.m  tae.lq.e.m  tae.diff.e.m
## 0        9.425970e+02 6.895783e+02 2.530187e+02
## 1-4      9.906270e+02 7.181268e+02 2.725002e+02
## 5-9      7.377945e+02 6.906145e+02 4.718001e+01
## 10-14    7.426336e+02 6.716764e+02 7.095719e+01
## 15-19    7.576876e+02 6.610412e+02 9.664641e+01
## 20-24    7.552469e+02 6.374057e+02 1.178412e+02
## 25-29    7.492093e+02 5.983426e+02 1.508667e+02
## 30-34    7.458124e+02 5.648200e+02 1.809923e+02
## 35-39    7.413157e+02 5.316529e+02 2.096628e+02
## 40-44    7.309395e+02 4.945496e+02 2.363898e+02
## 45-49    7.065773e+02 4.503434e+02 2.562339e+02
## 50-54    6.582527e+02 3.985604e+02 2.596924e+02
## 55-59    5.852584e+02 3.390018e+02 2.462566e+02
## 60-64    4.872041e+02 2.743171e+02 2.128870e+02
## 65-69    3.774717e+02 2.060942e+02 1.713775e+02
## 70-74    2.659910e+02 1.417105e+02 1.242805e+02
## 75-79    1.696396e+02 8.535009e+01 8.428948e+01
## 80-84    9.495974e+01 4.420907e+01 5.075067e+01
## 85-89    4.401718e+01 1.910333e+01 2.491386e+01
## 90-94    1.560098e+01 6.675621e+00 8.925354e+00
## 95-99    3.358679e+00 1.449960e+00 1.908718e+00
## 100-104  4.156157e-01 1.819738e-01 2.336418e-01
## 105-109  2.940539e-02 1.288794e-02 1.651745e-02
## 110+     3.122697e-03 1.076756e-03 2.045942e-03
##          1.130264e+04 8.224819e+03 3.077823e+03
##
## $tot.tae.e0
##          [,1]      [,2]      [,3]
## Female  11858.22 6788.56 5069.658
## Male    13272.38 9709.71 3562.667
# create a table with results for 1 components
print(file="../tables/ageCompTotC-1.txt"
      ,xtable(age.comps.1$tot.tae.e0,booktabs=TRUE,digits=2)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL)

```

```

,format.args=list(big.mark = ","))

# 2 components
# re-run models with 1 component
adult.2 <- FALSE
smooth.2 <- FALSE
N.2 <- 1
S.2 <- 1
C.2 <- 2
# base model
mod.1_0.m.2 <- svdMod(q1logit.m,Qlogit.m,N.2,S.2,10,TRUE,adult.2,TRUE,smooth.2,C.2)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "2 components"

mod.1_0.f.2 <- svdMod(q1logit.f,Qlogit.f,N.2,S.2,10,TRUE,adult.2,TRUE,smooth.2,C.2)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "2 components"

# calculate age-specific comparisons in prediction errors
# use the lt.q,q,l, and e from above
age.comps.2 <- ageSpecificErrorComparisons(mod.1_0.f.2,mod.1_0.m.2,lt.lq,q,l,e,a1.f,a1.m)
# have a look
cat("\n\n")

age.comps.2

## $tae.q
##          tae.comp.q.f   tae.lq.q.f   tae.diff.q.f
## 0        1.274576735  1.297233240 -0.0226565048
## 1-4      1.464136106  1.289036012  0.1751000938
## 5-9      0.794294528  0.736492559  0.0578019694
## 10-14    0.527633241  0.486823749  0.0408094920
## 15-19    0.654494947  0.704876573 -0.0503816256
## 20-24    0.802437950  0.857060707 -0.0546227572
## 25-29    0.787098123  0.848098118 -0.06099999953
## 30-34    0.771265784  0.804742161 -0.0334763779
## 35-39    0.858084645  0.844481623  0.0136030220
## 40-44    0.993076822  0.936995930  0.0560808912
## 45-49    1.153967148  1.120028613  0.0339385353
## 50-54    1.515299827  1.490918543  0.0243812842
## 55-59    1.982078009  1.979596056  0.0024819523
## 60-64    2.630576772  2.616680093  0.0138966788
## 65-69    3.277056679  3.254985725  0.0220709537
## 70-74    4.416161047  4.271187273  0.1449737741
## 75-79    5.298630816  4.926832431  0.3717983844
## 80-84    5.081064556  4.904941882  0.1761226743

```

```

## 85-89    3.583673446  3.511496470  0.0721769761
## 90-94    1.745529420  1.787401665  -0.0418722450
## 95-99    0.469609106  0.498031026  -0.0284219198
## 100-104   0.061834359  0.067499575  -0.0056652156
## 105-109   0.003966453  0.004375082  -0.0004086285
##          40.146546520  39.239815108  0.9067314120
##          tae.comp.q.m   tae.lq.q.m   tae.diff.q.m
## 0         1.509600915  1.526154490  -0.0165535749
## 1-4       1.975789923  1.536224082  0.4395658404
## 5-9       0.922196411  0.863027711  0.0591687000
## 10-14     0.546111345  0.486713551  0.0593977943
## 15-19     0.878192868  0.860015651  0.0181772174
## 20-24     1.663774653  1.643764536  0.0200101165
## 25-29     1.562006172  1.523449996  0.0385561762
## 30-34     1.525948754  1.479942826  0.0460059279
## 35-39     1.696987610  1.654222685  0.0427649251
## 40-44     2.011436727  1.952716291  0.0587204358
## 45-49     2.420656073  2.408272935  0.0123831376
## 50-54     2.988187437  3.064041949  -0.0758545122
## 55-59     3.667524455  3.794220593  -0.1266961376
## 60-64     4.555168517  4.740872963  -0.1857044463
## 65-69     5.190325852  5.348903581  -0.1585777283
## 70-74     5.492155417  5.727486555  -0.2353311372
## 75-79     5.116874712  5.222380347  -0.1055056350
## 80-84     3.667812104  3.843713007  -0.1759009029
## 85-89     2.050333330  2.090101282  -0.0397679519
## 90-94     0.835821288  0.856392989  -0.0205717009
## 95-99     0.172943644  0.184202976  -0.0112593325
## 100-104   0.018822848  0.020634919  -0.0018120702
## 105-109   0.001105483  0.001229015  -0.0001235321
##          50.469776536  50.828684927  -0.3589083909
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f tae.diff.e.f
## 0         4.494214e+02  4.488274e+02  0.594069885
## 1-4       4.869487e+02  4.927074e+02  -5.758724139
## 5-9       4.506641e+02  4.589434e+02  -8.279266588
## 10-14     4.300532e+02  4.375600e+02  -7.506758857
## 15-19     4.167974e+02  4.220134e+02  -5.215987381
## 20-24     3.981357e+02  3.997308e+02  -1.595054686
## 25-29     3.784827e+02  3.761462e+02  2.336535758
## 30-34     3.616948e+02  3.563449e+02  5.349996173
## 35-39     3.467243e+02  3.408473e+02  5.876933907
## 40-44     3.309285e+02  3.256429e+02  5.285597180
## 45-49     3.122929e+02  3.073443e+02  4.948613927
## 50-54     2.901023e+02  2.847086e+02  5.393615889
## 55-59     2.639688e+02  2.575757e+02  6.393124000
## 60-64     2.341441e+02  2.266830e+02  7.461111523
## 65-69     1.992108e+02  1.912519e+02  7.958990469
## 70-74     1.595599e+02  1.528070e+02  6.752897277
## 75-79     1.147498e+02  1.109866e+02  3.763228531
## 80-84     7.136196e+01  7.084205e+01  0.519912041
## 85-89     3.666040e+01  3.683845e+01  -0.178048408
## 90-94     1.482136e+01  1.493034e+01  -0.108984650

```

```

## 95-99    4.001519e+00 3.971052e+00  0.030466372
## 100-104  6.048429e-01 5.839672e-01  0.020875732
## 105-109  4.824255e-02 4.462715e-02  0.003615400
## 110+     6.971239e-03 2.913201e-03  0.004058038
##           5.751385e+03 5.717334e+03 34.050817392
##           tae.comp.e.m   tae.lq.e.m   tae.diff.e.m
## 0        6.634582e+02 6.895783e+02 -2.612005e+01
## 1-4      7.008593e+02 7.181268e+02 -1.726749e+01
## 5-9      6.525928e+02 6.906145e+02 -3.802173e+01
## 10-14    6.336385e+02 6.716764e+02 -3.803795e+01
## 15-19    6.231187e+02 6.610412e+02 -3.792251e+01
## 20-24    6.010082e+02 6.374057e+02 -3.639753e+01
## 25-29    5.631908e+02 5.983426e+02 -3.515183e+01
## 30-34    5.309889e+02 5.648200e+02 -3.383109e+01
## 35-39    4.994487e+02 5.316529e+02 -3.220416e+01
## 40-44    4.644951e+02 4.945496e+02 -3.005457e+01
## 45-49    4.242677e+02 4.503434e+02 -2.607569e+01
## 50-54    3.771518e+02 3.985604e+02 -2.140853e+01
## 55-59    3.227600e+02 3.390018e+02 -1.624183e+01
## 60-64    2.617384e+02 2.743171e+02 -1.257871e+01
## 65-69    1.969666e+02 2.060942e+02 -9.127579e+00
## 70-74    1.354991e+02 1.417105e+02 -6.211397e+00
## 75-79    8.238145e+01 8.535009e+01 -2.968636e+00
## 80-84    4.306296e+01 4.420907e+01 -1.146108e+00
## 85-89    1.921855e+01 1.910333e+01  1.152261e-01
## 90-94    6.920328e+00 6.675621e+00  2.447065e-01
## 95-99    1.533381e+00 1.449960e+00  8.342029e-02
## 100-104  1.946498e-01 1.819738e-01  1.267596e-02
## 105-109  1.400746e-02 1.288794e-02  1.119519e-03
## 110+     3.122697e-03 1.076756e-03  2.045942e-03
##           7.804511e+03 8.224819e+03 -4.203082e+02
##
## $tot.tae.e0
##           [,1]      [,2]      [,3]
## Female  6797.545 6788.56  8.985368
## Male    9341.923 9709.71 -367.787308

# create a table with results for 2 components
print(file="../tables/ageCompTotC-2.txt"
      ,xtable(age.comps.2$tot.tae.e0
              ,booktabs=TRUE,digits=2)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL
      ,format.args=list(big.mark = ","))

# 3 components
# re-run models with 1 component
adult.3 <- FALSE
smooth.3 <- FALSE
N.3 <- 1
S.3 <- 1
C.3 <- 3
# base model
mod.1_0.m.3 <- svdMod(q1logit.m,Qlogit.m,N.3,S.3,10,TRUE,adult.3,TRUE,smooth.3,C.3)

```

```

## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "3 components"

mod.1_0.f.3 <- svdMod(q1logit.f,Qlogit.f,N.3,S.3,10,TRUE,adult.3,TRUE,smooth.3,C.3)

## 
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "3 components"

# calculate age-specific comparisons in prediction errors
# use the lt.q,q,l, and e from above
age.comps.3 <- ageSpecificErrorComparisons(mod.1_0.f.3,mod.1_0.m.3,lt.lq,q,l,e,a1.f,a1.m)
# have a look
cat("\n\n")

age.comps.3

## $tae.q
##          tae.comp.q.f   tae.lq.q.f   tae.diff.q.f
## 0        1.274576735  1.297233240 -0.0226565048
## 1-4      1.370985429  1.289036012  0.0819494174
## 5-9      0.814100156  0.736492559  0.0776075973
## 10-14    0.502514458  0.486823749  0.0156907097
## 15-19    0.647035746  0.704876573 -0.0578408272
## 20-24    0.795107672  0.857060707 -0.0619530347
## 25-29    0.796199838  0.848098118 -0.0518982799
## 30-34    0.774349813  0.804742161 -0.0303923488
## 35-39    0.852665594  0.844481623  0.0081839711
## 40-44    0.968065747  0.936995930  0.0310698168
## 45-49    1.128203563  1.120028613  0.0081749504
## 50-54    1.478593841  1.490918543 -0.0123247022
## 55-59    1.941891772  1.979596056 -0.0377042838
## 60-64    2.529876783  2.616680093 -0.0868033107
## 65-69    3.189131700  3.254985725 -0.0658540255
## 70-74    4.330323096  4.271187273  0.0591358222
## 75-79    5.252009241  4.926832431  0.3251768099
## 80-84    5.062779049  4.904941882  0.1578371668
## 85-89    3.646417549  3.511496470  0.1349210786
## 90-94    1.774335813  1.787401665 -0.0130658525
## 95-99    0.467327107  0.498031026 -0.0307039189
## 100-104  0.059391361  0.067499575 -0.0081082144
## 105-109  0.003713478  0.004375082 -0.0006616039
##          39.659595541 39.239815108  0.4197804328
##          tae.comp.q.m   tae.lq.q.m   tae.diff.q.m
## 0        1.509600915  1.526154490 -0.0165535749
## 1-4      2.001558785  1.536224082  0.4653347030
## 5-9      0.891912574  0.863027711  0.0288848639
## 10-14    0.515133005  0.486713551  0.0284194547
## 15-19    0.885871237  0.860015651  0.0258555858

```

```

## 20-24    1.679336982  1.643764536  0.0355724458
## 25-29    1.575677214  1.523449996  0.0522272180
## 30-34    1.523167929  1.479942826  0.0432251030
## 35-39    1.694315814  1.654222685  0.0400931299
## 40-44    1.971726902  1.952716291  0.0190106107
## 45-49    2.378611109  2.408272935  -0.0296618257
## 50-54    2.967259087  3.064041949  -0.0967828620
## 55-59    3.675248467  3.794220593  -0.1189721259
## 60-64    4.563537255  4.740872963  -0.1773357085
## 65-69    5.172040609  5.348903581  -0.1768629715
## 70-74    5.436927133  5.727486555  -0.2905594213
## 75-79    5.0159115941 5.222380347  -0.2064644052
## 80-84    3.599325202  3.843713007  -0.2443878048
## 85-89    2.000742330  2.090101282  -0.0893589526
## 90-94    0.820532754  0.856392989  -0.0358602352
## 95-99    0.171853209  0.184202976  -0.0123497673
## 100-104  0.018875272  0.020634919  -0.0017596464
## 105-109  0.001117301  0.001229015  -0.0001117146
##           50.070287026 50.828684927  -0.7583979011
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f   tae.diff.e.f
## 0        4.429805e+02  4.488274e+02  -5.846906e+00
## 1-4      4.806365e+02  4.927074e+02  -1.207088e+01
## 5-9      4.471529e+02  4.589434e+02  -1.179053e+01
## 10-14    4.265818e+02  4.375600e+02  -1.097815e+01
## 15-19    4.132533e+02  4.220134e+02  -8.760122e+00
## 20-24    3.939629e+02  3.997308e+02  -5.767859e+00
## 25-29    3.728293e+02  3.761462e+02  -3.316880e+00
## 30-34    3.542997e+02  3.563449e+02  -2.045199e+00
## 35-39    3.386502e+02  3.408473e+02  -2.197108e+00
## 40-44    3.229880e+02  3.256429e+02  -2.654896e+00
## 45-49    3.055910e+02  3.073443e+02  -1.753346e+00
## 50-54    2.846426e+02  2.847086e+02  -6.607124e-02
## 55-59    2.596789e+02  2.575757e+02  2.103225e+00
## 60-64    2.309079e+02  2.266830e+02  4.224895e+00
## 65-69    1.978511e+02  1.912519e+02  6.599227e+00
## 70-74    1.594190e+02  1.528070e+02  6.612062e+00
## 75-79    1.155943e+02  1.109866e+02  4.607703e+00
## 80-84    7.258101e+01  7.084205e+01  1.738954e+00
## 85-89    3.759467e+01  3.683845e+01  7.562181e-01
## 90-94    1.503203e+01  1.493034e+01  1.016881e-01
## 95-99    3.987555e+00  3.971052e+00  1.650281e-02
## 100-104  5.858353e-01  5.839672e-01  1.868190e-03
## 105-109  4.509144e-02  4.462715e-02  4.642854e-04
## 110+     6.971239e-03  2.913201e-03  4.058038e-03
##           5.676853e+03  5.717334e+03  -4.048107e+01
##          tae.comp.e.m   tae.lq.e.m   tae.diff.e.m
## 0        6.610383e+02  6.895783e+02  -2.853996e+01
## 1-4      6.983313e+02  7.181268e+02  -1.979544e+01
## 5-9      6.511176e+02  6.906145e+02  -3.949694e+01
## 10-14    6.340052e+02  6.716764e+02  -3.767115e+01
## 15-19    6.236703e+02  6.610412e+02  -3.737089e+01
## 20-24    6.012310e+02  6.374057e+02  -3.617478e+01

```

```

## 25-29  5.635217e+02 5.983426e+02 -3.482087e+01
## 30-34  5.312999e+02 5.648200e+02 -3.352018e+01
## 35-39  4.995151e+02 5.316529e+02 -3.213781e+01
## 40-44  4.644366e+02 4.945496e+02 -3.011302e+01
## 45-49  4.239960e+02 4.503434e+02 -2.634735e+01
## 50-54  3.762851e+02 3.985604e+02 -2.227521e+01
## 55-59  3.210986e+02 3.390018e+02 -1.790322e+01
## 60-64  2.594474e+02 2.743171e+02 -1.486972e+01
## 65-69  1.943406e+02 2.060942e+02 -1.175363e+01
## 70-74  1.331900e+02 1.417105e+02 -8.520460e+00
## 75-79  8.060154e+01 8.535009e+01 -4.748550e+00
## 80-84  4.218301e+01 4.420907e+01 -2.026050e+00
## 85-89  1.879316e+01 1.910333e+01 -3.101724e-01
## 90-94  6.805076e+00 6.675621e+00  1.294544e-01
## 95-99  1.521595e+00 1.449960e+00  7.163438e-02
## 100-104 1.947094e-01 1.819738e-01  1.273551e-02
## 105-109 1.410490e-02 1.288794e-02  1.216961e-03
## 110+    3.122697e-03 1.076756e-03  2.045942e-03
##          7.786641e+03 8.224819e+03 -4.381783e+02
##
## $tot.tae.e0
##           [,1]      [,2]      [,3]
## Female  6700.125 6788.56 -88.43505
## Male    9307.849 9709.71 -401.86118

# create a table with results for 3 components
print(file="../tables/ageCompTotC-3.txt"
      ,xtable(age.comps.3$tot.tae.e0,booktabs=TRUE,digits=2)
      ,only.contents = TRUE,include.rownames = TRUE
      ,include.colnames = FALSE,hline.after=NULL
      ,format.args=list(big.mark = ","))

# 4 components
# re-run models with 1 component
adult.4 <- FALSE
smooth.4 <- FALSE
N.4 <- 1
S.4 <- 1
C.4 <- 4
# base model
mod.1_0.m.4 <- svdMod(q1logit.m,Qlogit.m,N.4,S.4,10,TRUE,adult.4,TRUE,smooth.4,C.4)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"
mod.1_0.f.4 <- svdMod(q1logit.f,Qlogit.f,N.4,S.4,10,TRUE,adult.4,TRUE,smooth.4,C.4)

##
## [1] "Adult mortality is direct input to predictions: FALSE"
## [1] "SVD model is smoothed: FALSE"
## [1] "1 iterations"

```

```

## [1] "100% sample fraction"
## [1] "4 components"
# calculate age-specific comparisons in prediction errors
# use the lt.q,q,l, and e from above
age.comps.4 <- ageSpecificErrorComparisons(mod.1_0.f.4,mod.1_0.m.4,lt.lq,q,l,e,a1.f,a1.m)
# have a look
cat("\n\n")

age.comps.4

## $tae.q
##          tae.comp.q.f    tae.lq.q.f    tae.diff.q.f
## 0      1.274576735  1.297233240 -0.0226565048
## 1-4    1.408642198  1.289036012  0.1196061863
## 5-9    0.768009199  0.736492559  0.0315166396
## 10-14   0.502821047  0.486823749  0.0159972985
## 15-19   0.626049559  0.704876573 -0.0788270136
## 20-24   0.768516703  0.857060707 -0.0885440042
## 25-29   0.763305793  0.848098118 -0.0847923259
## 30-34   0.756275811  0.804742161 -0.0484663507
## 35-39   0.836650885  0.844481623 -0.0078307374
## 40-44   0.961101183  0.936995930  0.0241052531
## 45-49   1.124972727  1.120028613  0.0049441141
## 50-54   1.478516483  1.490918543 -0.0124020603
## 55-59   1.945855361  1.979596056 -0.0337406952
## 60-64   2.512083891  2.616680093 -0.1045962025
## 65-69   3.108742498  3.254985725 -0.1462432268
## 70-74   4.081456235  4.271187273 -0.1897310380
## 75-79   4.841454736  4.926832431 -0.0853776953
## 80-84   4.740783508  4.904941882 -0.1641583735
## 85-89   3.495277867  3.511496470 -0.0162186029
## 90-94   1.754908944  1.787401665 -0.0324927218
## 95-99   0.466763887  0.498031026 -0.0312671395
## 100-104 0.059340397  0.067499575 -0.0081591779
## 105-109 0.003696479  0.004375082 -0.0006786026
##          38.279802127 39.239815108 -0.9600129814
##          tae.comp.q.m    tae.lq.q.m    tae.diff.q.m
## 0      1.509600915  1.526154490 -0.0165535749
## 1-4    2.002035933  1.536224082  0.4658118505
## 5-9    0.867411985  0.863027711  0.0043842741
## 10-14   0.515046975  0.486713551  0.0283334247
## 15-19   0.890692991  0.860015651  0.0306773406
## 20-24   1.688429365  1.643764536  0.0446648286
## 25-29   1.568435038  1.523449996  0.0449850427
## 30-34   1.509058761  1.479942826  0.0291159355
## 35-39   1.691430711  1.654222685  0.0372080269
## 40-44   1.987269079  1.952716291  0.0345527880
## 45-49   2.416185314  2.408272935  0.0079123789
## 50-54   2.998635037  3.064041949 -0.0654069116
## 55-59   3.594312689  3.794220593 -0.1999079037
## 60-64   4.334740268  4.740872963 -0.4061326950
## 65-69   4.843664238  5.348903581 -0.5052393425
## 70-74   5.100618529  5.727486555 -0.6268680259
## 75-79   4.764436021  5.222380347 -0.4579443256

```

```

## 80-84    3.528247334  3.843713007 -0.3154656731
## 85-89    2.007799919  2.090101282 -0.0823013636
## 90-94    0.814367488  0.856392989 -0.0420255011
## 95-99    0.163226183  0.184202976 -0.0209767935
## 100-104   0.017300338  0.020634919 -0.0033345806
## 105-109   0.001013241  0.001229015 -0.0002157742
##          48.813958352  50.828684927 -2.0147265750
##
## $tae.e
##           tae.comp.e.f   tae.lq.e.f   tae.diff.e.f
## 0          4.351343e+02  4.488274e+02 -1.369309e+01
## 1-4        4.746635e+02  4.927074e+02 -1.804391e+01
## 5-9        4.393281e+02  4.589434e+02 -1.961530e+01
## 10-14      4.182012e+02  4.375600e+02 -1.935875e+01
## 15-19      4.045099e+02  4.220134e+02 -1.750353e+01
## 20-24      3.850596e+02  3.997308e+02 -1.467120e+01
## 25-29      3.636798e+02  3.761462e+02 -1.246641e+01
## 30-34      3.446863e+02  3.563449e+02 -1.165858e+01
## 35-39      3.285306e+02  3.408473e+02 -1.231670e+01
## 40-44      3.121311e+02  3.256429e+02 -1.351182e+01
## 45-49      2.937892e+02  3.073443e+02 -1.355516e+01
## 50-54      2.718690e+02  2.847086e+02 -1.283961e+01
## 55-59      2.461976e+02  2.575757e+02 -1.137808e+01
## 60-64      2.165749e+02  2.266830e+02 -1.010808e+01
## 65-69      1.840224e+02  1.912519e+02 -7.229439e+00
## 70-74      1.475940e+02  1.528070e+02 -5.212995e+00
## 75-79      1.074493e+02  1.109866e+02 -3.537286e+00
## 80-84      6.869363e+01  7.084205e+01 -2.148418e+00
## 85-89      3.637768e+01  3.683845e+01 -4.607739e-01
## 90-94      1.489748e+01  1.493034e+01 -3.286166e-02
## 95-99      3.982213e+00  3.971052e+00  1.116088e-02
## 100-104    5.855478e-01  5.839672e-01  1.580673e-03
## 105-109    4.490054e-02  4.462715e-02  2.733874e-04
## 110+       6.971239e-03  2.913201e-03  4.058038e-03
##           tae.comp.e.m   tae.lq.e.m   tae.diff.e.m
## 0          6.510078e+02  6.895783e+02 -3.857054e+01
## 1-4        6.893637e+02  7.181268e+02 -2.876309e+01
## 5-9        6.410171e+02  6.906145e+02 -4.959741e+01
## 10-14      6.238369e+02  6.716764e+02 -4.783954e+01
## 15-19      6.136538e+02  6.610412e+02 -4.738741e+01
## 20-24      5.913498e+02  6.374057e+02 -4.605597e+01
## 25-29      5.519158e+02  5.983426e+02 -4.642676e+01
## 30-34      5.164862e+02  5.648200e+02 -4.833383e+01
## 35-39      4.821291e+02  5.316529e+02 -4.952381e+01
## 40-44      4.454383e+02  4.945496e+02 -4.911132e+01
## 45-49      4.041177e+02  4.503434e+02 -4.622569e+01
## 50-54      3.564159e+02  3.985604e+02 -4.214441e+01
## 55-59      3.025450e+02  3.390018e+02 -3.645682e+01
## 60-64      2.442199e+02  2.743171e+02 -3.009726e+01
## 65-69      1.839504e+02  2.060942e+02 -2.214377e+01
## 70-74      1.274482e+02  1.417105e+02 -1.426227e+01
## 75-79      7.842901e+01  8.535009e+01 -6.921080e+00
## 80-84      4.196632e+01  4.420907e+01 -2.242746e+00

```

```

## 85-89  1.885919e+01 1.910333e+01 -2.441372e-01
## 90-94  6.734159e+00 6.675621e+00  5.853751e-02
## 95-99  1.456355e+00 1.449960e+00  6.394490e-03
## 100-104 1.805199e-01 1.819738e-01 -1.453961e-03
## 105-109 1.289934e-02 1.288794e-02  1.140421e-05
## 110+    3.122697e-03 1.076756e-03  2.045942e-03
##          7.572537e+03 8.224819e+03 -6.522823e+02
##
## $tot.tae.e0
##      [,1]      [,2]      [,3]
## Female 6581.451 6788.56 -207.1093
## Male   9166.612 9709.71 -543.0982

# create lines for table with e0 total errors for log-quad and
# SVD-Comp with components 1-4
# start with log-quad
capture.output(file="../tables/ageCompLQ.txt",
cat(paste("Log-Quad & ",paste(format(round(age.comps.1$tot.tae.e0[,2],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\""))
)

# SVD-Comp components 1-4
capture.output(file="../tables/ageCompSVD-Comp.txt",
cat(paste("SVD-Comp, C=1 & ",paste(format(round(age.comps.1$tot.tae.e0[,1],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=2 & ",paste(format(round(age.comps.2$tot.tae.e0[,1],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=3 & ",paste(format(round(age.comps.3$tot.tae.e0[,1],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=4 & ",paste(format(round(age.comps.4$tot.tae.e0[,1],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\""))
)

# differences between SVD-Comp components 1-4 and log-quad
capture.output(file="../tables/ageCompSVD-CompLogQuadDiffs.txt",
cat(paste("SVD-Comp, C=1 - Log-Quad & ",paste(format(round(age.comps.1$tot.tae.e0[,1]
-age.comps.1$tot.tae.e0[,2],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=2 - Log-Quad & ",paste(format(round(age.comps.2$tot.tae.e0[,1]
-age.comps.1$tot.tae.e0[,2],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=3 - Log-Quad & ",paste(format(round(age.comps.3$tot.tae.e0[,1]
-age.comps.1$tot.tae.e0[,2],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\"\\n",
paste("SVD-Comp, C=4 - Log-Quad & ",paste(format(round(age.comps.4$tot.tae.e0[,1]
-age.comps.1$tot.tae.e0[,2],0)
,big.mark = ",",nsmall=0,trim=TRUE),collapse=" & "
,sep="")), " \\\\""))
)

```

```
)
```

7 Test on Other Countries

SVD-Comp is tested on two different countries that are not part of the HMD and are not developed countries but for which reasonable data exist: Mexico and South Africa. Example life tables for Mexico (1983–1985) from the Human Life Table Database (www.lifetable.de) [<https://www.lifetable.de/data/hld.zip>] and South Africa (2005) come from the WHO's Global Health Observatory [<http://apps.who.int/gho/data/?theme=main&vid=61540>]. The life tables are converted to standard five-year age groups ending at ages 80–84, the oldest second-to-last age group that is common across both examples. Predictions are made with both SVD-Comp and Log Quad using both child and adult mortality as predictors, and both the data and those predictions are plotted.

```
# Mexico
# read 1983-1985 Mexican life tables from Human Life Table Database
mex <- read.csv("../data/non-HMD life tables/Mexico1983-1985.csv", header=TRUE)
# female
mex.f.q <- mex[,15][97:191]
mex.f.q <- standardFiveYear(mex.f.q)[1:18]
# male
mex.m.q <- mex[,15][1:95]
mex.m.q <- standardFiveYear(mex.m.q)[1:18]

# South Africa
# read 2005 life tables for South Africa from the WHO Global Health Observatory
rsa <- read.csv("../data/non-HMD life tables/SouthAfrica2005.csv", header=TRUE)
# female
rsa.f.q <- rsa[,3][1:18]
# male
rsa.m.q <- rsa[,2][1:18]

# Now have standard 5qx through ages 80-84, i.e. not including 1.0 at age 85

# logits
mex.f ql <- logit(mex.f.q)
mex.m ql <- logit(mex.m.q)
rsa.f ql <- logit(rsa.f.q)
rsa.m ql <- logit(rsa.m.q)

# child and adult Mx

# Mexico
# female
mex.f.Q <- rep(0,2)
# child mx
mex.f.Q[1] <- childQ5(mex.f.q)
# adult mx
mex.f.Q[2] <- adultQ5(mex.f.q)
# mmale
mex.m.Q <- rep(0,2)
# child mx
mex.m.Q[1] <- childQ5(mex.m.q)
# adult mx
```

```

mex.m.Q[2] <- adultQ5(mex.m.q)

# RSA
# female
rsa.f.Q <- rep(0,2)
# child mx
rsa.f.Q[1] <- childQ5(rsa.f.q)
# adult mx
rsa.f.Q[2] <- adultQ5(rsa.f.q)
# mmale
rsa.m.Q <- rep(0,2)
# child mx
rsa.m.Q[1] <- childQ5(rsa.m.q)
# adult mx
rsa.m.Q[2] <- adultQ5(rsa.m.q)

# have a look
mex.f.Q

## [1] 0.05336201 0.13518150
mex.m.Q

## [1] 0.06264442 0.23507692
rsa.f.Q

## [1] 0.0688960 0.4660984
rsa.m.Q

## [1] 0.0815180 0.5427579
# Predictions

# models
adult <- TRUE
smooth <- TRUE
N <- 1
S <- 1
C <- 4
mod.1_0.sm.m <- svdMod(q1logit.m,Qlogit.m,N,S,10,TRUE,adult,TRUE,TRUE,C)

##
## [1] "Adult mortality is direct input to predictions: TRUE"
## [1] "SVD model is smoothed: TRUE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

mod.1_0.sm.f <- svdMod(q1logit.f,Qlogit.f,N,S,10,TRUE,adult,TRUE,TRUE,C)

##
## [1] "Adult mortality is direct input to predictions: TRUE"
## [1] "SVD model is smoothed: TRUE"
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "4 components"

```

```

### predictions with child and adult

### Mexico
# female
mex.f.p.ca <- ltPredict(mod.1_0.sm.f,TRUE,logit(mex.f.Q[1]),logit(mex.f.Q[2]))
mex.f.p5.ca <- standardFiveYear(expit(mex.f.p.ca[,1]))
# male
mex.m.p.ca <- ltPredict(mod.1_0.sm.m,TRUE,logit(mex.m.Q[1]),logit(mex.m.Q[2]))
mex.m.p5.ca <- standardFiveYear(expit(mex.m.p.ca[,1]))

## RSA
# female
rsa.f.p.ca <- ltPredict(mod.1_0.sm.f,TRUE,logit(rsa.f.Q[1]),logit(rsa.f.Q[2]))
rsa.f.p5.ca <- standardFiveYear(expit(rsa.f.p.ca[,1]))
# male
rsa.m.p.ca <- ltPredict(mod.1_0.sm.m,TRUE,logit(rsa.m.Q[1]),logit(rsa.m.Q[2]))
rsa.m.p5.ca <- standardFiveYear(expit(rsa.m.p.ca[,1]))

# predictions with Log-Quad

# Source functions file
source("../R/logQuad/DataProgramsExamples/R/functions.R")

# Create labels for age vectors
ages.5x1 <- c("0","1-4",paste(seq(5,105,5),seq(9,109,5),sep="-"),"110+")
sexes <- c("Female","Male","Total")

# Import matrix of model coefficients
tmp1 <- read.csv("../R/logQuad/DataProgramsExamples/Data/coefs.logquad.HMD719.csv")
tmp2 <- array(c(as.matrix(tmp1[, 3:6]))
              , dim=c(24, 3, 4)
              , dimnames=list(ages.5x1, sexes, c("ax", "bx", "cx", "vx")))
coefs <- aperm(tmp2, c(1,3,2))

### Mexico
# female
mex.f.LQp.ca <- lthat.any2.logquad(
  coefs,"Female",Q5=mex.f.Q[1],QQa=mex.f.Q[2])$lt[1:23,2] # with adult
# male
mex.m.LQp.ca <- lthat.any2.logquad(
  coefs,"Male",Q5=mex.m.Q[1],QQa=mex.m.Q[2])$lt[1:23,2] # with adult

### RSA
# female
rsa.f.LQp.ca <- lthat.any2.logquad(
  coefs,"Female",Q5=rsa.f.Q[1],QQa=rsa.f.Q[2])$lt[1:23,2] # with adult
# male
rsa.m.LQp.ca <- lthat.any2.logquad(
  coefs,"Male",Q5=rsa.m.Q[1],QQa=rsa.m.Q[2])$lt[1:23,2] # with adult

### Plots, logit scale

### Mexico

```

```

# female child and adult only
plot(mex.f.q1)
points(logit(mex.f.p5.ca),type="l",col="blue")
points(logit(mex.f.LQp.ca),type="l")
# male child and adult only
plot(mex.m.q1)
points(logit(mex.m.p5.ca),type="l",col="blue")
points(logit(mex.m.LQp.ca),type="l")

### RSA
# female child and adult only
plot(rsa.f.q1)
points(logit(rsa.f.p5.ca),type="l",col="blue")
points(logit(rsa.f.LQp.ca),type="l")
# male child and adult only
plot(rsa.m.q1)
points(logit(rsa.m.p5.ca),type="l",col="blue")
points(logit(rsa.m.LQp.ca),type="l")

### ggplot

# data

ages <- ages.5x1[1:18]
# data only
q.logit.data <- data.frame(rbind(
  cbind(c(0,1,seq(5,80,5))
    ,mex.f.q1,"Mexico","Female","Data")
  ,cbind(c(0,1,seq(5,80,5))
    ,mex.m.q1,"Mexico","Male","Data")
  ,cbind(c(0,1,seq(5,80,5))
    ,rsa.f.q1,"South Africa","Female","Data")
  ,cbind(c(0,1,seq(5,80,5))
    ,rsa.m.q1,"South Africa","Male","Data"))
))
colnames(q.logit.data) <- c("Age","Value","Country","Sex","Source")
rownames(q.logit.data) <- seq(1,18*4,1)
q.logit.data$Age <- as.numeric(as.character(q.logit.data$Age))
q.logit.data$Value <- as.numeric(as.character(q.logit.data$Value))
# predicted values
q.logit.pred <- data.frame(rbind(
  cbind(c(0,1,seq(5,80,5))
    ,logit(mex.f.p5.ca)[1:18],"Mexico","Female","Predicted by SVD-Comp")
  ,cbind(c(0,1,seq(5,80,5))
    ,logit(mex.m.p5.ca)[1:18],"Mexico","Male","Predicted by SVD-Comp")
  ,cbind(c(0,1,seq(5,80,5))
    ,logit(mex.f.LQp.ca)[1:18],"Mexico","Female","Predicted by Log-Quad")
  ,cbind(c(0,1,seq(5,80,5))
    ,logit(mex.m.LQp.ca)[1:18],"Mexico","Male","Predicted by Log-Quad")
  ,cbind(c(0,1,seq(5,80,5))
    ,logit(rsa.f.p5.ca)[1:18],"South Africa","Female","Predicted by SVD-Comp")
  ,cbind(c(0,1,seq(5,80,5))
    ,logit(rsa.m.p5.ca)[1:18],"South Africa","Male","Predicted by SVD-Comp"))
)

```

```

,cbind(c(0,1,seq(5,80,5))
      ,logit(rsa.f.LQp.ca)[1:18],"South Africa","Female","Predicted by Log-Quad")
,cbind(c(0,1,seq(5,80,5))
      ,logit(rsa.m.LQp.ca)[1:18],"South Africa","Male","Predicted by Log-Quad")
))
colnames(q.logit.pred) <- c("Age","Value","Country","Sex","Source")
rownames(q.logit.pred) <- seq(1,18*8,1)
q.logit.pred$Age <- as.numeric(as.character(q.logit.pred$Age))
q.logit.pred$Value <- as.numeric(as.character(q.logit.pred$Value))
q.logit.pred

```

	Age	Value	Country	Sex
## 1	0	-3.16229313	Mexico	Female
## 2	1	-4.75009499	Mexico	Female
## 3	5	-5.53012219	Mexico	Female
## 4	10	-5.88708037	Mexico	Female
## 5	15	-5.50020932	Mexico	Female
## 6	20	-5.30128721	Mexico	Female
## 7	25	-5.12443893	Mexico	Female
## 8	30	-4.86455240	Mexico	Female
## 9	35	-4.52458850	Mexico	Female
## 10	40	-4.16886332	Mexico	Female
## 11	45	-3.79788596	Mexico	Female
## 12	50	-3.39630325	Mexico	Female
## 13	55	-2.98850515	Mexico	Female
## 14	60	-2.50961680	Mexico	Female
## 15	65	-2.00227945	Mexico	Female
## 16	70	-1.42026301	Mexico	Female
## 17	75	-0.80905797	Mexico	Female
## 18	80	-0.16651585	Mexico	Female
## 19	0	-2.96713683	Mexico	Male
## 20	1	-4.44583543	Mexico	Male
## 21	5	-5.14337389	Mexico	Male
## 22	10	-5.38710972	Mexico	Male
## 23	15	-4.71864216	Mexico	Male
## 24	20	-4.38907217	Mexico	Male
## 25	25	-4.38261119	Mexico	Male
## 26	30	-4.26896209	Mexico	Male
## 27	35	-4.02106831	Mexico	Male
## 28	40	-3.68155998	Mexico	Male
## 29	45	-3.29233530	Mexico	Male
## 30	50	-2.87248318	Mexico	Male
## 31	55	-2.44894568	Mexico	Male
## 32	60	-1.98962824	Mexico	Male
## 33	65	-1.52420279	Mexico	Male
## 34	70	-1.00803730	Mexico	Male
## 35	75	-0.45704580	Mexico	Male
## 36	80	0.15380435	Mexico	Male
## 37	0	-3.14378162	Mexico	Female
## 38	1	-4.36588948	Mexico	Female
## 39	5	-5.65752532	Mexico	Female
## 40	10	-6.04630623	Mexico	Female
## 41	15	-5.69006250	Mexico	Female
## 42	20	-5.47445718	Mexico	Female

## 43	25	-5.25989363	Mexico	Female
## 44	30	-4.97604526	Mexico	Female
## 45	35	-4.61769092	Mexico	Female
## 46	40	-4.23856979	Mexico	Female
## 47	45	-3.84040530	Mexico	Female
## 48	50	-3.42644951	Mexico	Female
## 49	55	-3.00770496	Mexico	Female
## 50	60	-2.48413340	Mexico	Female
## 51	65	-1.93595423	Mexico	Female
## 52	70	-1.31650115	Mexico	Female
## 53	75	-0.68048081	Mexico	Female
## 54	80	-0.05337267	Mexico	Female
## 55	0	-2.95041291	Mexico	Male
## 56	1	-4.28376855	Mexico	Male
## 57	5	-5.08349945	Mexico	Male
## 58	10	-5.33390921	Mexico	Male
## 59	15	-4.69779670	Mexico	Male
## 60	20	-4.38907757	Mexico	Male
## 61	25	-4.38742733	Mexico	Male
## 62	30	-4.27169186	Mexico	Male
## 63	35	-4.02638839	Mexico	Male
## 64	40	-3.70442732	Mexico	Male
## 65	45	-3.30003467	Mexico	Male
## 66	50	-2.85954531	Mexico	Male
## 67	55	-2.40497650	Mexico	Male
## 68	60	-1.92954455	Mexico	Male
## 69	65	-1.45194116	Mexico	Male
## 70	70	-0.94569706	Mexico	Male
## 71	75	-0.39864474	Mexico	Male
## 72	80	0.18129494	Mexico	Male
## 73	0	-2.92674668	South Africa	Female
## 74	1	-3.97361502	South Africa	Female
## 75	5	-5.77846890	South Africa	Female
## 76	10	-5.93109048	South Africa	Female
## 77	15	-4.39913852	South Africa	Female
## 78	20	-4.00296459	South Africa	Female
## 79	25	-3.72254448	South Africa	Female
## 80	30	-3.47168798	South Africa	Female
## 81	35	-3.17025603	South Africa	Female
## 82	40	-2.92538339	South Africa	Female
## 83	45	-2.61922038	South Africa	Female
## 84	50	-2.29899497	South Africa	Female
## 85	55	-1.96426787	South Africa	Female
## 86	60	-1.57797872	South Africa	Female
## 87	65	-1.21154606	South Africa	Female
## 88	70	-0.78338895	South Africa	Female
## 89	75	-0.34923993	South Africa	Female
## 90	80	0.14639367	South Africa	Female
## 91	0	-2.71791199	South Africa	Male
## 92	1	-3.64823732	South Africa	Male
## 93	5	-4.70568274	South Africa	Male
## 94	10	-4.82546662	South Africa	Male
## 95	15	-3.62432145	South Africa	Male
## 96	20	-2.97126379	South Africa	Male

```

## 97 25 -2.83743015 South Africa Male
## 98 30 -2.75531200 South Africa Male
## 99 35 -2.62781359 South Africa Male
## 100 40 -2.45813257 South Africa Male
## 101 45 -2.25428802 South Africa Male
## 102 50 -2.00431564 South Africa Male
## 103 55 -1.73569724 South Africa Male
## 104 60 -1.37239190 South Africa Male
## 105 65 -1.00482848 South Africa Male
## 106 70 -0.56986695 South Africa Male
## 107 75 -0.07658767 South Africa Male
## 108 80 0.49789527 South Africa Male
## 109 0 -2.90412553 South Africa Female
## 110 1 -4.00631550 South Africa Female
## 111 5 -3.68628735 South Africa Female
## 112 10 -3.73116611 South Africa Female
## 113 15 -3.04774334 South Africa Female
## 114 20 -2.74891422 South Africa Female
## 115 25 -2.68677932 South Africa Female
## 116 30 -2.71393627 South Africa Female
## 117 35 -2.71744123 South Africa Female
## 118 40 -2.71604827 South Africa Female
## 119 45 -2.63324876 South Africa Female
## 120 50 -2.43892804 South Africa Female
## 121 55 -2.17063112 South Africa Female
## 122 60 -1.87705073 South Africa Female
## 123 65 -1.47428624 South Africa Female
## 124 70 -1.02804138 South Africa Female
## 125 75 -0.51873534 South Africa Female
## 126 80 0.04001734 South Africa Female
## 127 0 -2.69594596 South Africa Male
## 128 1 -3.91555820 South Africa Male
## 129 5 -4.20696450 South Africa Male
## 130 10 -4.52232347 South Africa Male
## 131 15 -3.78742645 South Africa Male
## 132 20 -3.17384325 South Africa Male
## 133 25 -2.96299314 South Africa Male
## 134 30 -2.77415274 South Africa Male
## 135 35 -2.56119479 South Africa Male
## 136 40 -2.33738148 South Africa Male
## 137 45 -2.11351664 South Africa Male
## 138 50 -1.85815711 South Africa Male
## 139 55 -1.60725550 South Africa Male
## 140 60 -1.26893860 South Africa Male
## 141 65 -0.93059509 South Africa Male
## 142 70 -0.53962853 South Africa Male
## 143 75 -0.10991051 South Africa Male
## 144 80 0.38891300 South Africa Male

## Source
## 1 Predicted by SVD-Comp
## 2 Predicted by SVD-Comp
## 3 Predicted by SVD-Comp
## 4 Predicted by SVD-Comp
## 5 Predicted by SVD-Comp

```

```
## 6 Predicted by SVD-Comp
## 7 Predicted by SVD-Comp
## 8 Predicted by SVD-Comp
## 9 Predicted by SVD-Comp
## 10 Predicted by SVD-Comp
## 11 Predicted by SVD-Comp
## 12 Predicted by SVD-Comp
## 13 Predicted by SVD-Comp
## 14 Predicted by SVD-Comp
## 15 Predicted by SVD-Comp
## 16 Predicted by SVD-Comp
## 17 Predicted by SVD-Comp
## 18 Predicted by SVD-Comp
## 19 Predicted by SVD-Comp
## 20 Predicted by SVD-Comp
## 21 Predicted by SVD-Comp
## 22 Predicted by SVD-Comp
## 23 Predicted by SVD-Comp
## 24 Predicted by SVD-Comp
## 25 Predicted by SVD-Comp
## 26 Predicted by SVD-Comp
## 27 Predicted by SVD-Comp
## 28 Predicted by SVD-Comp
## 29 Predicted by SVD-Comp
## 30 Predicted by SVD-Comp
## 31 Predicted by SVD-Comp
## 32 Predicted by SVD-Comp
## 33 Predicted by SVD-Comp
## 34 Predicted by SVD-Comp
## 35 Predicted by SVD-Comp
## 36 Predicted by SVD-Comp
## 37 Predicted by Log-Quad
## 38 Predicted by Log-Quad
## 39 Predicted by Log-Quad
## 40 Predicted by Log-Quad
## 41 Predicted by Log-Quad
## 42 Predicted by Log-Quad
## 43 Predicted by Log-Quad
## 44 Predicted by Log-Quad
## 45 Predicted by Log-Quad
## 46 Predicted by Log-Quad
## 47 Predicted by Log-Quad
## 48 Predicted by Log-Quad
## 49 Predicted by Log-Quad
## 50 Predicted by Log-Quad
## 51 Predicted by Log-Quad
## 52 Predicted by Log-Quad
## 53 Predicted by Log-Quad
## 54 Predicted by Log-Quad
## 55 Predicted by Log-Quad
## 56 Predicted by Log-Quad
## 57 Predicted by Log-Quad
## 58 Predicted by Log-Quad
## 59 Predicted by Log-Quad
```

```
## 60 Predicted by Log-Quad
## 61 Predicted by Log-Quad
## 62 Predicted by Log-Quad
## 63 Predicted by Log-Quad
## 64 Predicted by Log-Quad
## 65 Predicted by Log-Quad
## 66 Predicted by Log-Quad
## 67 Predicted by Log-Quad
## 68 Predicted by Log-Quad
## 69 Predicted by Log-Quad
## 70 Predicted by Log-Quad
## 71 Predicted by Log-Quad
## 72 Predicted by Log-Quad
## 73 Predicted by SVD-Comp
## 74 Predicted by SVD-Comp
## 75 Predicted by SVD-Comp
## 76 Predicted by SVD-Comp
## 77 Predicted by SVD-Comp
## 78 Predicted by SVD-Comp
## 79 Predicted by SVD-Comp
## 80 Predicted by SVD-Comp
## 81 Predicted by SVD-Comp
## 82 Predicted by SVD-Comp
## 83 Predicted by SVD-Comp
## 84 Predicted by SVD-Comp
## 85 Predicted by SVD-Comp
## 86 Predicted by SVD-Comp
## 87 Predicted by SVD-Comp
## 88 Predicted by SVD-Comp
## 89 Predicted by SVD-Comp
## 90 Predicted by SVD-Comp
## 91 Predicted by SVD-Comp
## 92 Predicted by SVD-Comp
## 93 Predicted by SVD-Comp
## 94 Predicted by SVD-Comp
## 95 Predicted by SVD-Comp
## 96 Predicted by SVD-Comp
## 97 Predicted by SVD-Comp
## 98 Predicted by SVD-Comp
## 99 Predicted by SVD-Comp
## 100 Predicted by SVD-Comp
## 101 Predicted by SVD-Comp
## 102 Predicted by SVD-Comp
## 103 Predicted by SVD-Comp
## 104 Predicted by SVD-Comp
## 105 Predicted by SVD-Comp
## 106 Predicted by SVD-Comp
## 107 Predicted by SVD-Comp
## 108 Predicted by SVD-Comp
## 109 Predicted by Log-Quad
## 110 Predicted by Log-Quad
## 111 Predicted by Log-Quad
## 112 Predicted by Log-Quad
## 113 Predicted by Log-Quad
```

```

## 114 Predicted by Log-Quad
## 115 Predicted by Log-Quad
## 116 Predicted by Log-Quad
## 117 Predicted by Log-Quad
## 118 Predicted by Log-Quad
## 119 Predicted by Log-Quad
## 120 Predicted by Log-Quad
## 121 Predicted by Log-Quad
## 122 Predicted by Log-Quad
## 123 Predicted by Log-Quad
## 124 Predicted by Log-Quad
## 125 Predicted by Log-Quad
## 126 Predicted by Log-Quad
## 127 Predicted by Log-Quad
## 128 Predicted by Log-Quad
## 129 Predicted by Log-Quad
## 130 Predicted by Log-Quad
## 131 Predicted by Log-Quad
## 132 Predicted by Log-Quad
## 133 Predicted by Log-Quad
## 134 Predicted by Log-Quad
## 135 Predicted by Log-Quad
## 136 Predicted by Log-Quad
## 137 Predicted by Log-Quad
## 138 Predicted by Log-Quad
## 139 Predicted by Log-Quad
## 140 Predicted by Log-Quad
## 141 Predicted by Log-Quad
## 142 Predicted by Log-Quad
## 143 Predicted by Log-Quad
## 144 Predicted by Log-Quad

# plot data and predictions
ggplot(data = q.logit.data, aes(x=Age, y=Value, colour=Source)) +
  geom_line(data = q.logit.pred, aes(x=Age, y=Value, colour=Source), size=1) +
  scale_x_continuous(breaks=c(0,1,seq(5,80,5))
    ,labels=c("0,1-4","",paste(seq(5,80,5),c(seq(9,84,5)),sep="-"))
    ,minor_breaks = c()) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  geom_point(size=1.5) +
  # geom_line(aes(x=Age, y=Value, colour=Source), size=0.5) +
  labs(y = expression(''[bolditalic(n)]*bolditalic('q')[bolditalic(x)]*bold(' (logit scale)'))
    , x = expression(bold("Age (years)"))) +
  facet_wrap(~interaction(Sex,Country,sep=" ", ncol=2) +
  # facet_wrap(~Sex + Country,ncol=2) +
  theme(legend.title=element_blank(),legend.position=c(.15,.91)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
  theme(strip.text = element_text(face="bold")))
ggsave("../figures/fig7.pdf",width=6.5,height=6.5,units=c("in"))

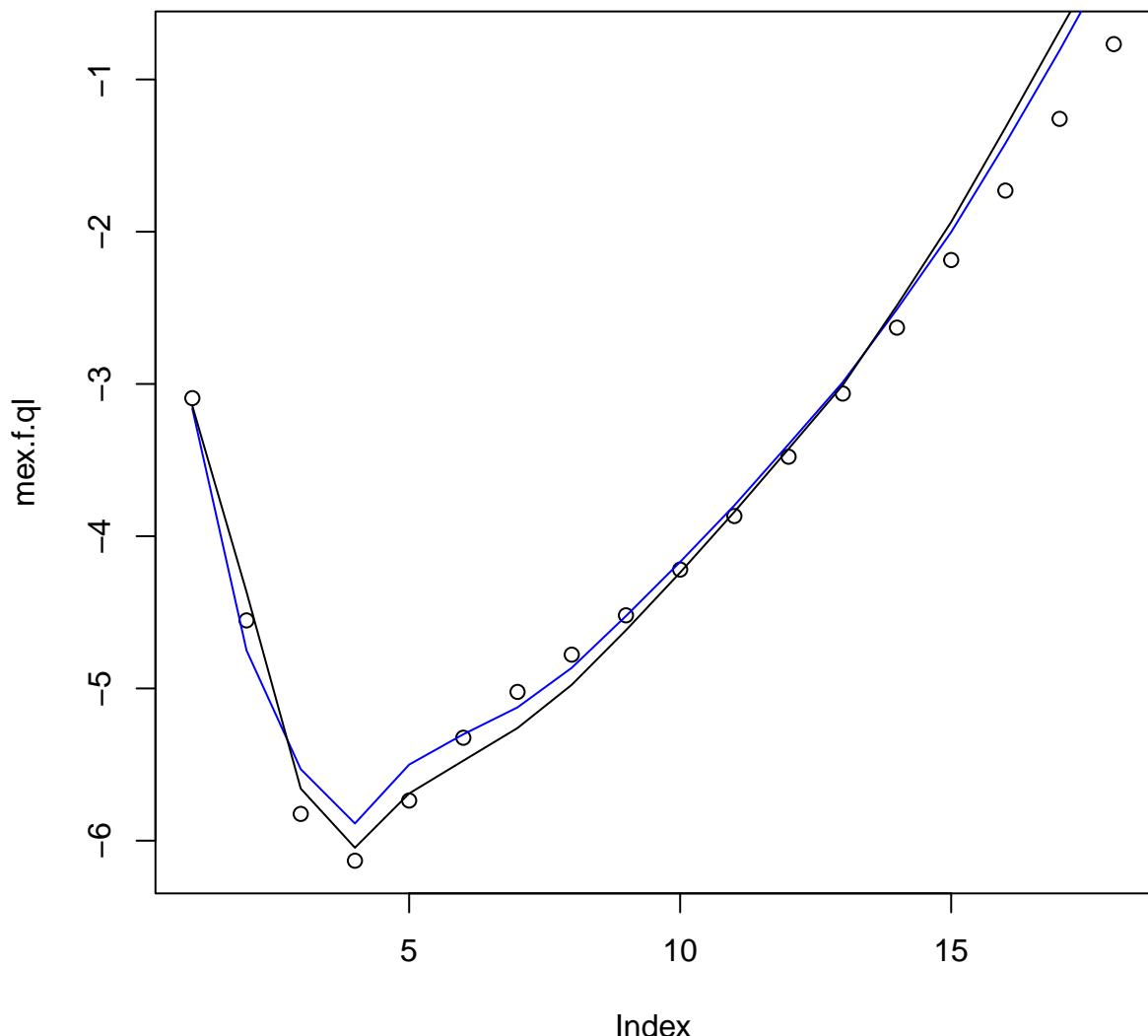
# grayscale
ggplot(data = q.logit.data, aes(x=Age, y=Value, colour=Source)) +
  geom_line(data = q.logit.pred, aes(x=Age, y=Value, colour=Source), size=1) +
  scale_x_continuous(breaks=c(0,1,seq(5,80,5)))

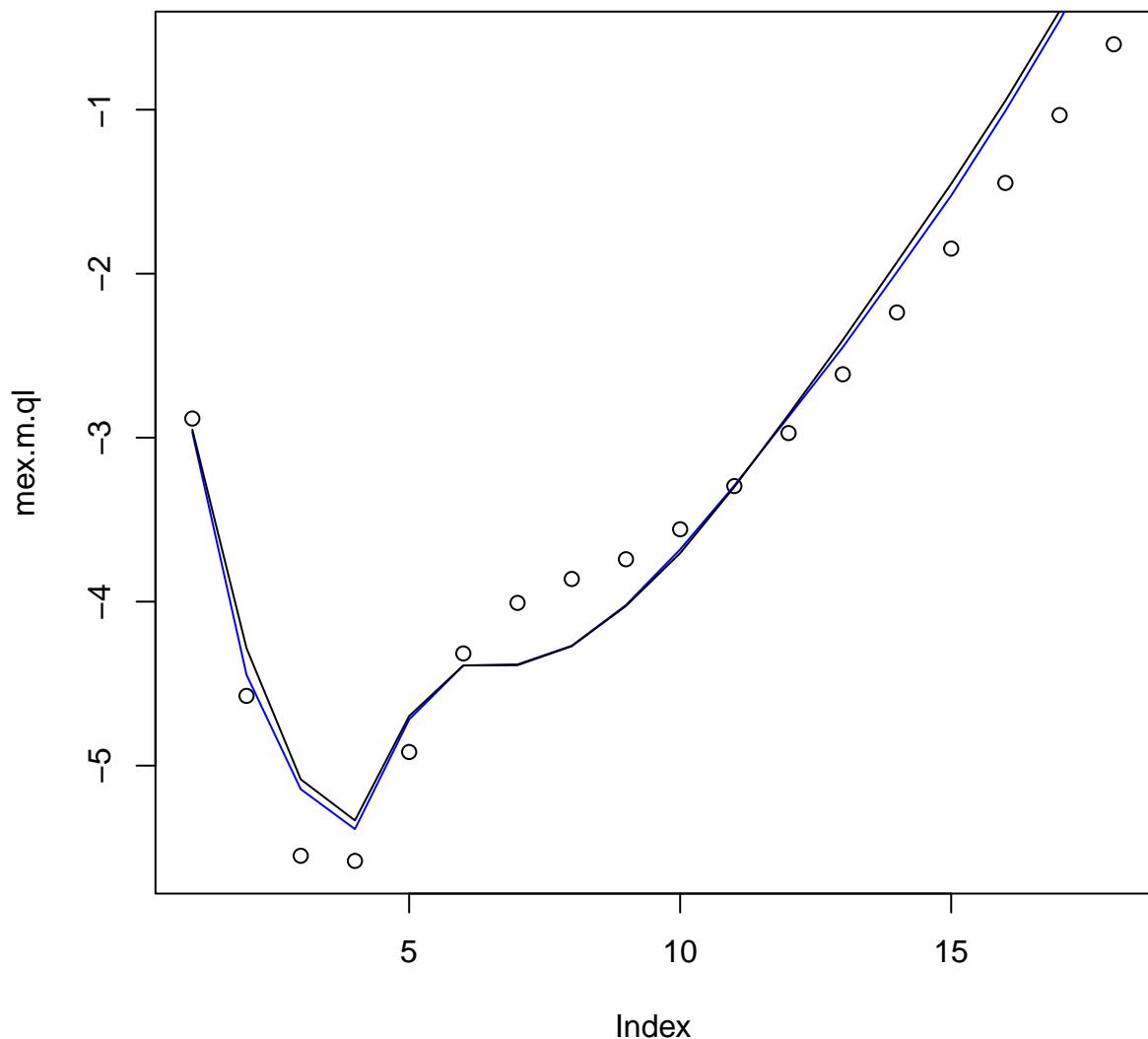
```

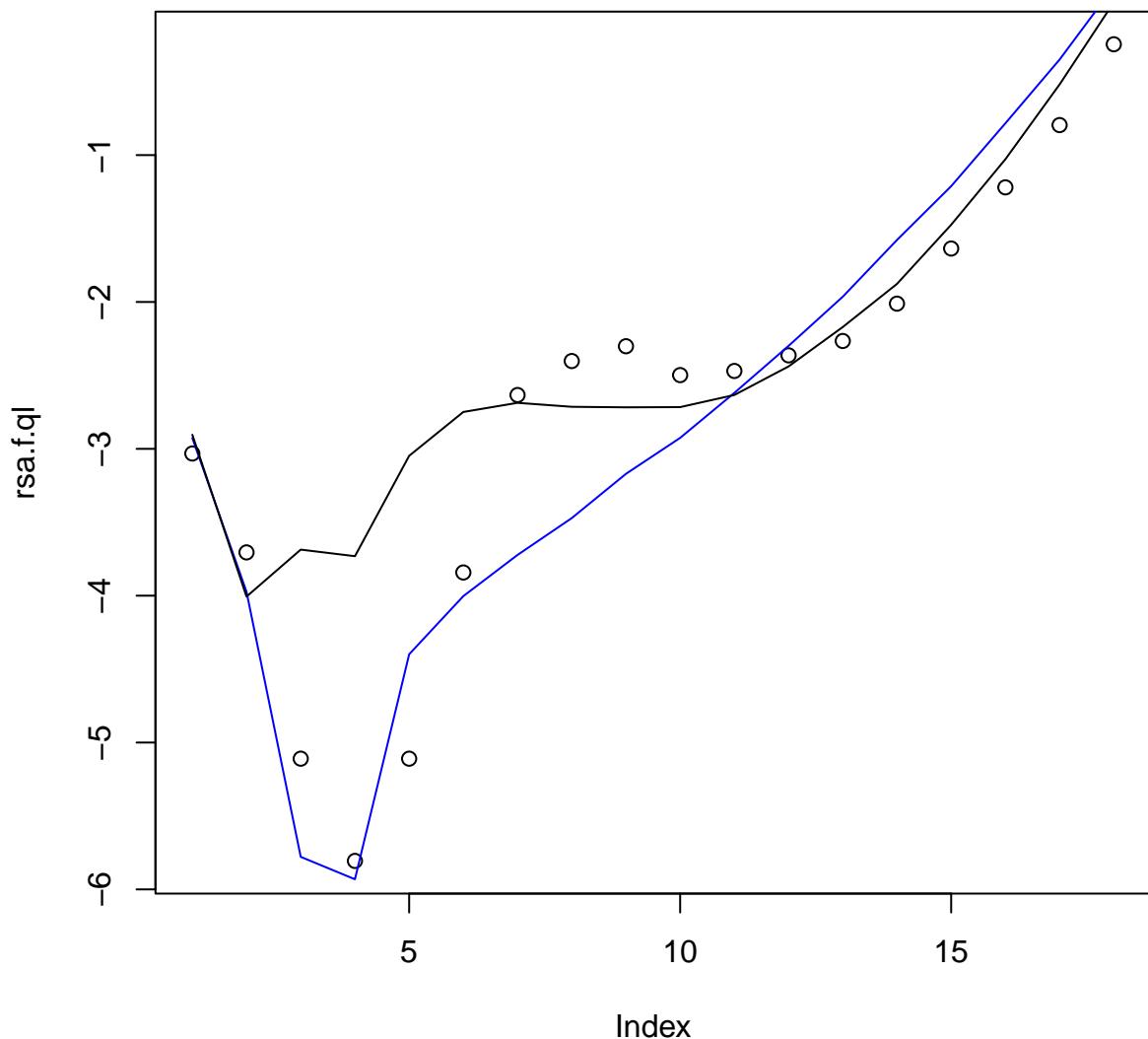
```

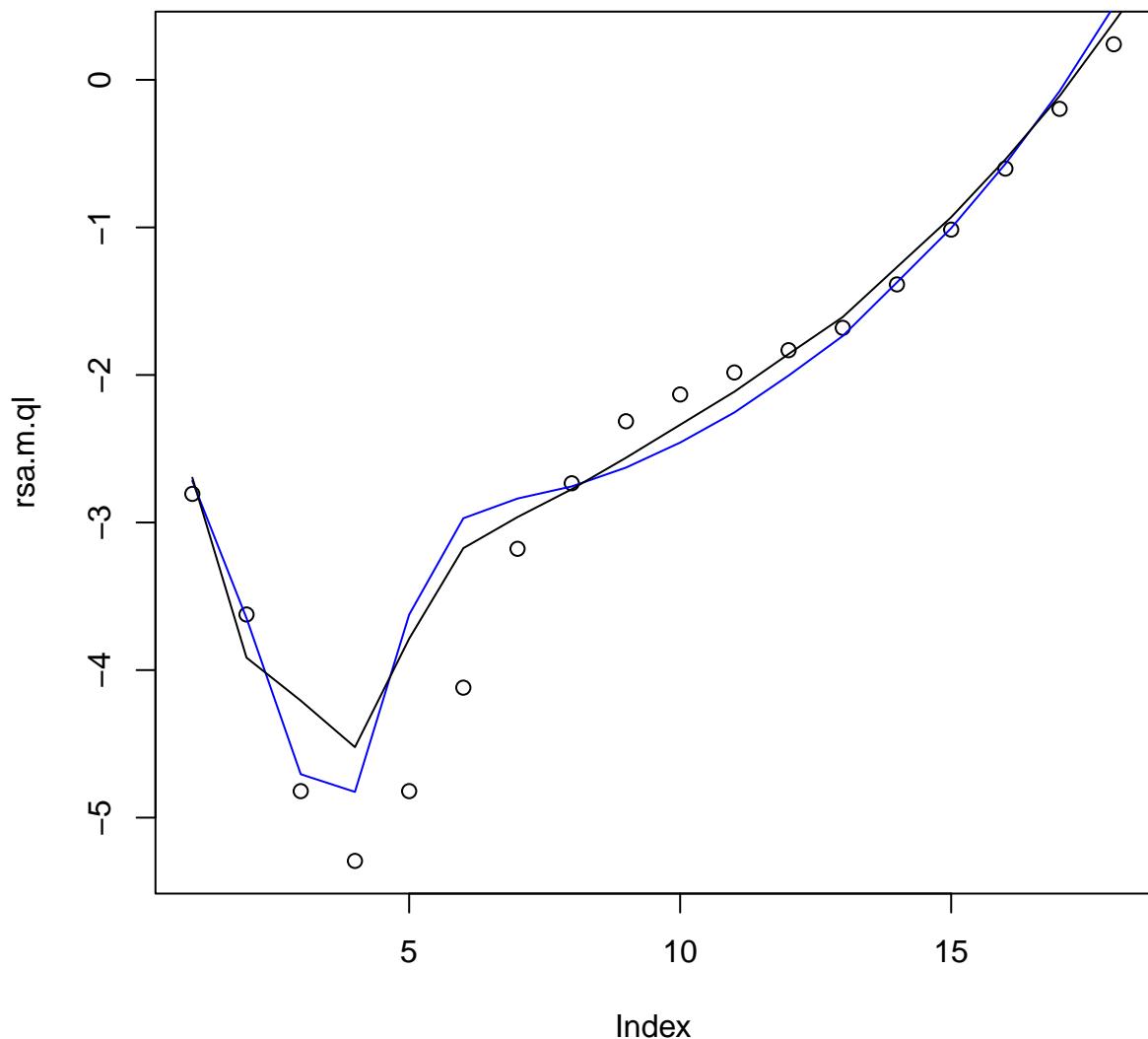
        ,labels=c("0,1-4","",paste(seq(5,80,5),c(seq(9,84,5)),sep="-"))
        ,minor_breaks = c() +
theme_bw() +
theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
geom_point(size=1.5) +
# geom_line(aes(x=Age, y=Value, colour=Source), size=0.5) +
labs(y = expression(''[bolditalic(n)]*bolditalic('q')[bolditalic(x)]*bold(' (logit scale)'))
     , x = expression(bold("Age (years)"))) +
facet_wrap(~interaction(Sex,Country,sep=" ", ncol=2) +
# facet_wrap(~Sex + Country,ncol=2) +
theme(legend.title=element_blank(),legend.position=c(.15,.91)) +
theme(legend.position="bottom", legend.box = "horizontal") +
theme(strip.text = element_text(face="bold")) +
scale_colour_grey(start = 0, end = .8)
ggsave("../figures/fig7-BW.pdf",width=6.5,height=6.5,units=c("in"))

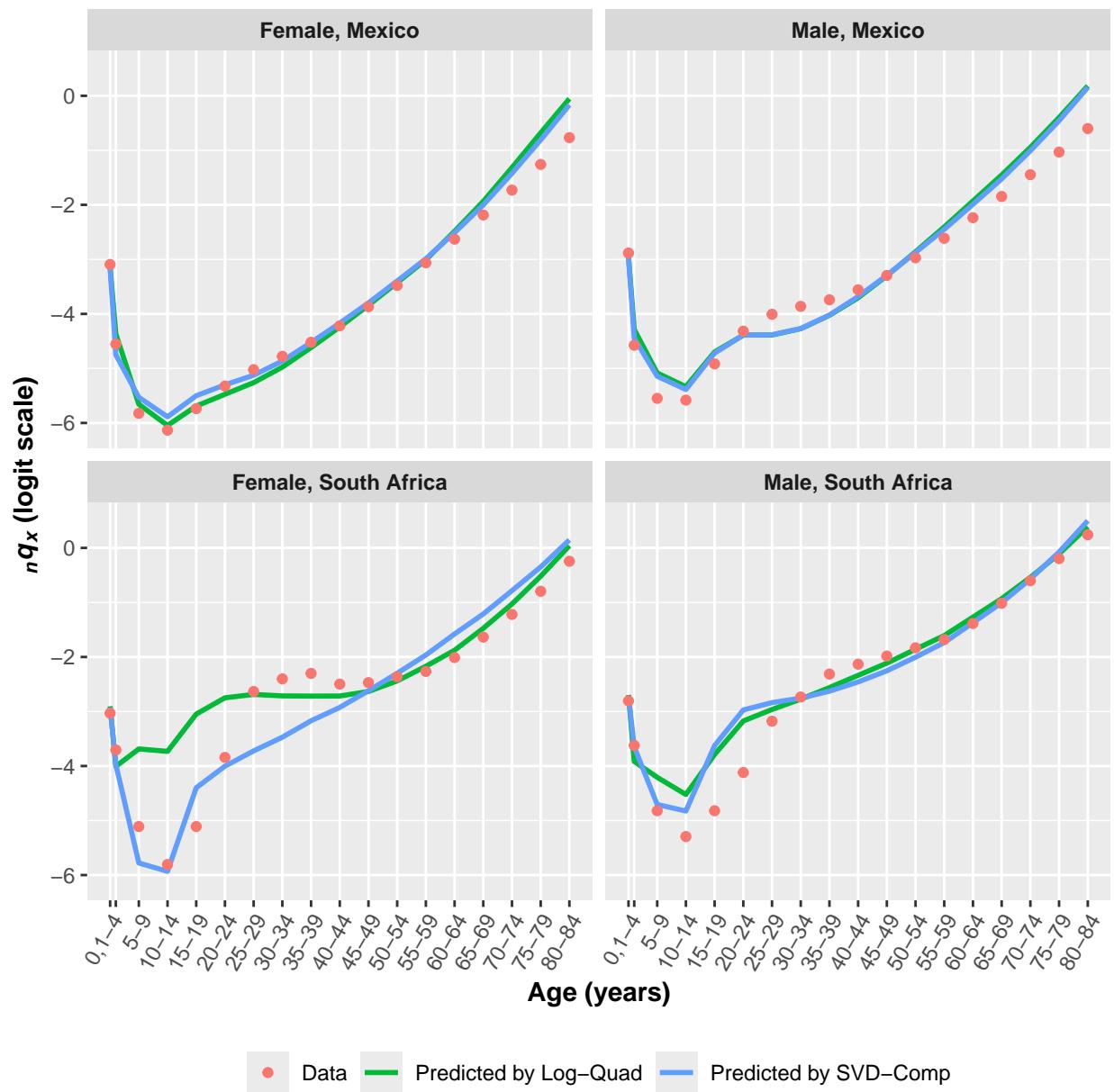
```

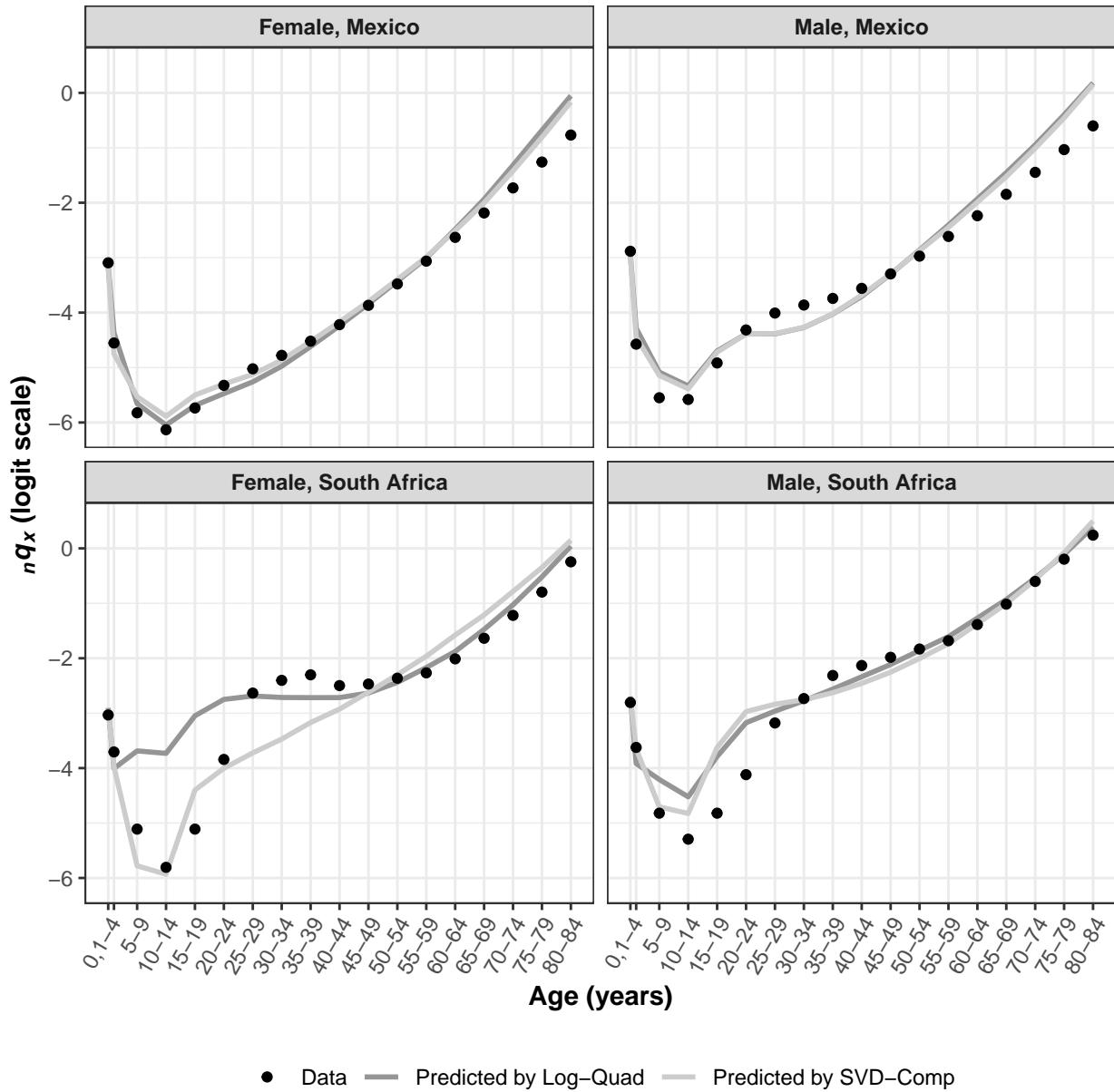












8 Make Compressed Models Object for Package

The SVD-Comp package predicts life tables using child or (child,adult) mortality as inputs. To do this calibrated by HMD, it needs all the component values and models calculated above. This piece of code wraps all that into a nice list and saves it in very compact form.

```
# function to make a list with all the information necessary to do predictions
#   using the HMD-calibrated SVD-Comp
make.models <- function(allModels.f,allModels.sm.f,allModels.m,allModels.sm.m) {

  # initialize array to hold components
  components <- array(
    data=rep(0,880),
    dim=c(2,4,110),
    dimnames=list(
      # ... (omitted for brevity)
    )
  )

  # calculate components
  components[1] <- calculateComponents(allModels.f,allModels.sm.f,allModels.m,allModels.sm.m)

  # calculate other components
  components[2] <- calculateOtherComponents(allModels.f,allModels.sm.f,allModels.m,allModels.sm.m)

  # ... (omitted for brevity)
}
```

```

    c("female","male"),
    c("1","2","3","4"),
    c(paste(seq(0,109,1),sep=' ',"))
)
)

# initialize array to hold smoothed components
components.sm <- array(
  data=rep(0,880),
  dim=c(2,4,110),
  dimnames=list(
    c("female","male"),
    c("1","2","3","4"),
    c(paste(seq(0,109,1),sep=' ',")))
)
)

# grab the component values
for (s in 1:2) {
  for (v in 1:4) {
    ifelse(
      s==1,
      components[s,v,] <- allModels.f$svd$s1$d[v]*allModels.f$svd$s1$u[,v],
      components[s,v,] <- allModels.m$svd$s1$d[v]*allModels.m$svd$s1$u[,v]
    )
  }
}

# store female and male components separately
components.f <- components[1,,]
components.m <- components[2,,]

# same for smooth components
for (s in 1:2) {
  for (v in 1:4) {
    ifelse(
      s==1,
      components.sm[s,v,] <- allModels.sm.f$svd$s1$d[v]*allModels.sm.f$svd.sm$s1$u[,v],
      components.sm[s,v,] <- allModels.sm.m$svd$s1$d[v]*allModels.sm.m$svd.sm$s1$u[,v]
    )
  }
}

components.sm.f <- components.sm[1,,]
components.sm.m <- components.sm[2,,]

# # plot the components to be sure you have the right ones
# par(mfrow=c(4,2))
# pdf(file="../figures/components.pdf")
# {
#   plot(components[1,1,])
#   points(components[2,1,],col="red")
#   points(components.sm[1,1,],type="l")
}

```

```

# points(components.sm[2,1,],type="l",col="red")
#
# plot(components[1,2,])
# points(components[2,2,],col="red")
# points(components.sm[1,2,],type="l")
# points(components.sm[2,2,],type="l",col="red")
#
# plot(components[1,3,])
# points(components[2,3,],col="red")
# points(components.sm[1,3,],type="l")
# points(components.sm[2,3,],type="l",col="red")
#
# plot(components[1,4,])
# points(components[2,4,],col="red")
# points(components.sm[1,4,],type="l")
# points(components.sm[2,4,],type="l",col="red")
#
# plot(components.sm[1,1,],type="l",ylim=c(-1250,75))
# abline(h=0,lwd=0.5)
# points(components.sm[1,2,],type="l",col="red")
# points(components.sm[1,3,],type="l",col="green")
# points(components.sm[1,4,],type="l",col="blue")
#
# plot(components.sm[1,2,],type="l",col="red")
# abline(h=0,lwd=0.5)
# points(components.sm[1,3,],type="l",col="green")
# points(components.sm[1,4,],type="l",col="blue")
#
# plot(components.sm[2,1,],type="l",ylim=c(-1250,75))
# abline(h=0,lwd=0.5)
# points(components.sm[2,2,],type="l",col="red")
# points(components.sm[2,3,],type="l",col="green")
# points(components.sm[2,4,],type="l",col="blue")
#
# plot(components.sm[2,2,],type="l",col="red")
# abline(h=0,lwd=0.5)
# points(components.sm[2,3,],type="l",col="green")
# points(components.sm[2,4,],type="l",col="blue")
# } # test that I got the right SVD stuff
# dev.off()

# strip unnecessary stuff from an object
cleanModel = function(cm) {
  cm$y = c()
  cm$model = c()
  cm$residuals = c()
  cm$fitted.values = c()
  cm$effects = c()
  cm$qr$qr = c()
  cm$linear.predictors = c()
  cm$weights = c()
  cm$prior.weights = c()
  cm$data = c()
}

```

```

attr(cm$terms,".Environment") = c()
attr(cm$formula,".Environment") = c()
return(cm)
}

# # have a look at the massive compression
# object.size(allModels.m$mods$s1$v1)
# object.size(cleanModel(allModels.m$mods$s1$v1))
# as.numeric(object.size(cleanModel(allModels.m$mods$s1$v1))) / as.numeric(object.size(allModels.m$mo

# create the female models list
mods.f <- list(
  components = components.f, # components
  components.sm = components.sm.f, # smooth components
  aml = cleanModel(allModels.f$mods$s1$aml), # adult mx model
  v1 = cleanModel(allModels.f$mods$s1$v1), # v1 model
  v2 = cleanModel(allModels.f$mods$s1$v2), # v2 model
  v3 = cleanModel(allModels.f$mods$s1$v3), # v3 model
  v4 = cleanModel(allModels.f$mods$s1$v4), # v4 model
  offset = allModels.f$offset, # offset
  q0 = cleanModel(allModels.f$mods$s1$q0), # 1q0 model
  rownames = rownames(allModels.f$ql.samp$s1) # row names = age groups
)
# mods.f
# object.size(mods.f)

# make male models list
mods.m <- list(
  components = components.m,
  components.sm = components.sm.m,
  aml = cleanModel(allModels.m$mods$s1$aml),
  v1 = cleanModel(allModels.m$mods$s1$v1),
  v2 = cleanModel(allModels.m$mods$s1$v2),
  v3 = cleanModel(allModels.m$mods$s1$v3),
  v4 = cleanModel(allModels.m$mods$s1$v4),
  offset = allModels.m$offset,
  q0 = cleanModel(allModels.m$mods$s1$q0),
  rownames = rownames(allModels.m$ql.samp$s1)
)
# mods.m
object.size(mods.m)

# make a list of both female and male model lists
mods <- list(
  female = mods.f,
  male = mods.m
)
# mods

# have a look at the size of the finished models list
# format(object.size(mods),units="Mb")

# library(dplyr)

```

```

# d <- ldply(names(mods), function(v) {
#   v.size <- format(object.size(mods[[v]]), unit="Mb")
#   data.frame(variable=v, size=v.size)
# })
# # have a look at the sizes of the components of the models list
# d[order(as.numeric(d$size), decreasing=TRUE),]

return(mods)

}

# REMEMBER TO RENAME THIS WITH EACH UPDATE OF HMD
# create and save the models object for the package
mods2024 <- make.models(mod.1_0.f,mod.1_0.sm.f,mod.1_0.m,mod.1_0.sm.m)
# have a look at it
mods2024

## $female
## $female$components
##          0           1           2           3
## 1 -972.91349 -1123.43646 -1167.910961 -1192.712003
## 2  -34.81028    -72.94632    -71.395415   -67.432650
## 3   14.71944     11.19250     6.455879    2.106179
## 4  -12.75612    -15.99700    -14.076739   -21.483708
##          4           5           6           7
## 1 -1208.005233 -1221.271744 -1230.818299 -1241.54286
## 2   -61.244152    -60.188047    -57.742453   -56.00178
## 3    7.972546     1.894367    -6.498427   -12.55972
## 4  -16.442942    -16.668050    -17.438678   -15.50472
##          8           9          10          11
## 1 -1249.20994 -1255.25990 -1257.528532 -1257.2126061
## 2   -51.84514    -49.97878    -48.829405   -47.4679172
## 3   -20.18971    -18.45742    -24.156110   -20.4946739
## 4  -17.09387    -16.07411    -9.669938   -0.6949281
##          12          13          14          15
## 1 -1253.4577490 -1246.00315 -1235.247641 -1223.65088
## 2   -44.2644633    -41.75949    -36.595524   -31.87461
## 3   -15.8346402    -15.31339    -10.260870   -10.62327
## 4    0.6162533     2.25183     8.685016   14.72471
##          16          17          18          19
## 1 -1211.997320 -1205.117017 -1197.466951 -1194.2138088
## 2   -27.661249    -26.271255    -22.321822   -22.2651489
## 3   -3.295143    -2.300484    -1.590282    0.7468359
## 4   13.142611    10.880073    14.016514   14.4791157
##          20          21          22          23
## 1 -1192.065617 -1191.438685 -1189.138021 -1188.484383
## 2   -24.183772    -25.629856    -26.252417   -28.386555
## 3   -2.594435    -4.127336    -1.342834   -1.560177
## 4   13.431211    14.974219    14.266309   15.887050
##          24          25          26          27
## 1 -1186.4995977 -1183.794970 -1181.17792 -1178.536186
## 2   -27.8725062    -26.109626    -24.57630   -24.009458
## 3   -0.4619104    -1.073254    2.00312    2.264274
## 4   14.2171511    13.229259    12.50655   10.875827

```

```

##          28          29          30          31
## 1 -1176.298294 -1171.335236 -1167.018340 -1165.654293
## 2 -24.552705 -19.441678 -18.085164 -17.451315
## 3  1.771149  5.264095  4.446515  2.073362
## 4 13.533763  8.073832  7.195991  9.392158
##          32          33          34          35
## 1 -1160.376117 -1157.083219 -1151.614304 -1146.125815
## 2 -16.220679 -15.767664 -11.921374 -8.424125
## 3  3.572999  1.596068  6.829267  6.873290
## 4  9.920636 11.668651  9.376474  6.232228
##          36          37          38          39
## 1 -1142.463185 -1138.709610 -1131.942416 -1128.132642
## 2 -8.077629 -7.089690 -2.951217 -2.102633
## 3  5.620087  4.404774  7.980759  7.358577
## 4  6.401035  6.209539  3.756289  5.075188
##          40          41          42          43
## 1 -1122.1437923 -1119.092895 -1111.885433 -1107.791341
## 2  0.4644841  2.746790  5.288577  7.781581
## 3  7.7793326  5.147231  7.659171  6.623792
## 4 -0.1392554  5.923456  2.346856  2.517957
##          44          45          46          47
## 1 -1102.842559 -1096.816105 -1092.715734 -1086.8778428
## 2  9.862106 12.025890 12.736976 16.4447468
## 3  6.383241  8.604181  7.304073  7.6134340
## 4  3.530487  1.780868  3.061289  0.6093993
##          48          49          50
## 1 -1080.8130717 -1.075356e+03 -1068.6165556
## 2  17.2425085  1.932955e+01 18.5670222
## 3  8.1337630  7.618321e+00 9.1108738
## 4 -0.1306609  7.216159e-02 0.6016079
##          51          52          53
## 1 -1065.419542 -1058.1768103 -1053.7790042
## 2  20.209862 19.7867058 20.6289677
## 3  6.369617  8.4123784 7.2115352
## 4  1.059093 -0.7040014 0.2024709
##          54          55          56          57
## 1 -1047.9603336 -1042.849299 -1036.956375 -1031.757225
## 2  21.0523232 22.121147 22.435804 23.439826
## 3  7.7982877  7.562139  8.342947  7.709589
## 4 -0.5792483 -1.080430 -1.194020 -1.476486
##          58          59          60          61
## 1 -1025.340452 -1019.705269 -1011.925999 -1007.528994
## 2  22.762224 22.798111 20.782821 23.018289
## 3  7.736643  6.993970  8.315999  7.198134
## 4 -1.831611 -1.544383 -2.627240 -2.462454
##          62          63          64          65
## 1 -999.578882 -993.816622 -987.054611 -979.930160
## 2  21.760930 21.359686 21.375022 21.251951
## 3  8.474786  6.253066  7.238997  8.007061
## 4 -3.310906 -3.384239 -3.892839 -4.088647
##          66          67          68          69
## 1 -974.589978 -967.529550 -960.189409 -953.612671
## 2  21.726131 22.208663 21.364963 21.917324
## 3  6.059498  6.155349  6.478501  5.047575

```

```

## 4 -4.320950 -4.567565 -5.062520 -4.877193
##      70      71      72      73
## 1 -944.389448 -939.546708 -930.123067 -923.048076
## 2  20.118113  21.691703  20.561613  21.057189
## 3   7.130693   3.965492   5.347559   4.442763
## 4  -5.859300  -5.503449  -6.377358  -6.478775
##      74      75      76      77
## 1 -915.019091 -907.266447 -900.000286 -893.284216
## 2  21.287388  21.117699  21.910287  22.647300
## 3   5.231373   4.518156   3.914395   1.161830
## 4  -6.605901  -6.912590  -7.122604  -6.886699
##      78      79      80      81
## 1 -884.327017 -877.279200 -868.554261 -862.954172
## 2  22.522164  23.531807  23.933364  26.206811
## 3   3.324128   1.646214   2.918571   1.037466
## 4  -7.019513  -6.411154  -6.685907  -5.681086
##      82      83      84      85
## 1 -853.6460016 -846.0091574 -837.7479952 -830.694471
## 2  25.6678251  26.4761537  26.6089944  27.632842
## 3   0.9937386  -0.0417946  -0.3594744  -1.331794
## 4  -6.0064269  -5.6678264  -5.6227134  -5.211594
##      86      87      88      89
## 1 -823.090145 -816.250818 -809.147180 -802.375953
## 2  28.802767  29.893255  30.632603  32.069325
## 3  -2.219276  -3.331298  -4.045424  -5.287420
## 4  -5.015759  -4.404557  -4.010685  -3.516515
##      90      91      92      93
## 1 -794.294685 -789.250968 -781.692159 -775.313754
## 2  31.684770  34.070853  34.195215  35.181671
## 3  -5.026972  -6.980752  -7.211167  -7.992459
## 4  -3.397687  -2.398520  -2.330963  -1.935585
##      94      95      96      97
## 1 -768.801147 -761.685907 -755.428445 -749.336397
## 2  35.767313  36.008683  36.751813  37.475439
## 3  -8.459966  -9.082524  -9.727391  -10.344121
## 4  -1.677911  -1.488586  -1.150994  -0.819747
##      98      99      100     101
## 1 -743.4202914 -737.6904787 -732.1568610 -726.8285549
## 2  38.1758672  38.8492647  39.4918334  40.0997678
## 3 -10.9304916 -11.4841834 -12.0032173 -12.4856705
## 4  -0.4965047  -0.1827301   0.1197687   0.4094079
##      102     103     104     105
## 1 -721.7137035 -716.8194612 -712.151561 -707.714246
## 2  40.6695933  41.1980853  41.682466  42.120490
## 3 -12.9298175 -13.3343368 -13.698191 -14.020772
## 4   0.6845059   0.9436628   1.185508   1.408924
##      106     107     108     109
## 1 -703.510233 -699.540392 -695.804031 -692.298604
## 2  42.510601  42.851866  43.144240  43.388166
## 3 -14.301866 -14.541771 -14.741189 -14.901354
## 4   1.613119   1.797549   1.962056   2.106707
##
## $female$components.sm
##      0      1      2      3

```

```

## 1 -972.91349 -1123.43646 -1167.91096 -1192.71200
## 2 -34.81028 -62.04911 -67.33761 -66.37081
## 3 14.71944 11.19250 7.37010 5.29843
## 4 -12.75612 -15.99700 -14.07674 -16.90008
##          4      5      6      7
## 1 -1208.005233 -1221.271744 -1230.818299 -1241.54286
## 2 -62.987362 -60.101431 -57.761520 -55.27597
## 3 2.898234 -0.843851 -6.060561 -11.59658
## 4 -17.106262 -16.974042 -16.564199 -15.77578
##          8      9     10     11
## 1 -1249.20994 -1255.25990 -1257.528532 -1257.212606
## 2 -52.63703 -50.42087 -48.667466 -46.734132
## 3 -16.11777 -18.92786 -19.849000 -18.978046
## 4 -14.27207 -11.72025 -8.153749 -4.032733
##          12     13     14     15
## 1 -1253.4577490 -1246.003148 -1235.247641 -1223.650882
## 2 -44.1428548 -40.755251 -36.615579 -32.330952
## 3 -16.8564419 -14.207073 -11.384213 -8.442729
## 4 0.1123678 3.976483 7.353014 9.969568
##          16     17     18     19
## 1 -1211.997320 -1205.117017 -1197.46695 -1194.21381
## 2 -28.682476 -25.857960 -23.89783 -23.35994
## 3 -5.600088 -3.336175 -2.00903 -1.64954
## 4 11.676509 12.656308 13.25998 13.71710
##          20     21     22     23
## 1 -1192.06562 -1191.438685 -1189.138021 -1188.484383
## 2 -24.14860 -25.379971 -26.540399 -27.312819
## 3 -1.91205 -2.139249 -1.920216 -1.369308
## 4 14.08645 14.352967 14.461953 14.331307
##          24     25     26
## 1 -1186.4995977 -1183.7949704 -1181.177920
## 2 -27.1621105 -26.1598748 -25.025503
## 3 -0.6823664 0.1424415 1.085004
## 4 13.9205225 13.2852808 12.525903
##          27     28     29     30
## 1 -1178.536186 -1176.298294 -1171.335236 -1167.018340
## 2 -24.078992 -22.685093 -20.543845 -18.620012
## 3 2.016612 2.846066 3.422771 3.550450
## 4 11.693857 10.815529 10.019852 9.525419
##          31     32     33     34
## 1 -1165.654293 -1160.376117 -1157.083219 -1151.614304
## 2 -17.343683 -16.190561 -14.498879 -12.042398
## 3 3.390741 3.449869 4.055775 4.973945
## 4 9.402780 9.415108 9.189310 8.540888
##          35     36     37     38
## 1 -1146.125815 -1142.463185 -1138.709610 -1131.942416
## 2 -9.629685 -7.848679 -6.049012 -3.859027
## 3 5.647935 5.932698 6.208437 6.648812
## 4 7.583988 6.549458 5.576845 4.710945
##          39     40     41     42
## 1 -1128.132642 -1122.1437923 -1119.092895 -1111.885433
## 2 -1.685602 0.4947735 2.835163 5.252459
## 3 6.962591 6.9511010 6.812496 6.792650
## 4 4.006724 3.5274980 3.252666 3.071895

```

```

##          43          44          45          46
## 1 -1107.791341 -1102.842559 -1096.816105 -1092.715734
## 2    7.603614     9.754995   11.675038   13.585126
## 3    6.927353     7.172974   7.446598   7.638966
## 4    2.880061     2.619471   2.250749   1.764355
##          47          48          49
## 1 -1086.877843 -1080.8130717 -1075.3558673
## 2   15.594001    17.3009237  18.4433916
## 3    7.760797     7.8701246   7.9327587
## 4    1.230529     0.7815904   0.5008143
##          50          51          52
## 1 -1068.6165556 -1065.4195424 -1.058177e+03
## 2   19.1330966    19.6612784  2.011073e+01
## 3    7.8797902    7.7524769   7.665232e+00
## 4    0.3456537    0.2020626  -3.855293e-03
##          53          54          55
## 1 -1053.7790042 -1047.9603336 -1042.8492991
## 2   20.6034755    21.2293985  21.9115690
## 3    7.6545256    7.7026517   7.7833921
## 4   -0.2737296   -0.5781451  -0.8880089
##          56          57          58          59
## 1 -1036.956375 -1031.757225 -1025.340452 -1019.705269
## 2   22.509629    22.848291   22.763810   22.353942
## 3    7.826373     7.776008   7.679143   7.634284
## 4   -1.182169   -1.456091  -1.727069  -2.021749
##          60          61          62          63
## 1 -1011.925999 -1007.528994 -999.578882 -993.816622
## 2   22.042894    22.029561   21.874354   21.569111
## 3    7.646259     7.616405   7.480942   7.304170
## 4   -2.354158   -2.715609  -3.083361  -3.436068
##          64          65          66          67
## 1 -987.054611 -979.930160 -974.589978 -967.529550
## 2   21.417347    21.477940   21.674907   21.773949
## 3    7.159337     6.970245   6.666075   6.342801
## 4   -3.762274   -4.062622  -4.346771  -4.626994
##          68          69          70          71
## 1 -960.189409 -953.612671 -944.389448 -939.546708
## 2   21.637781    21.343494   21.076137   21.008284
## 3    6.088858     5.868058   5.591903   5.261777
## 4   -4.912797   -5.209341  -5.516033  -5.825049
##          72          73          74          75
## 1 -930.123067 -923.048076 -915.019091 -907.266447
## 2   20.990182    21.043666   21.200975   21.462337
## 3    4.993468     4.819188   4.590467   4.133866
## 4   -6.121446   -6.387295  -6.606223  -6.762962
##          76          77          78          79
## 1 -900.000286 -893.284216 -884.327017 -877.279200
## 2   21.903811    22.395341   22.873330   23.518788
## 3    3.472556     2.859946   2.477908   2.226177
## 4   -6.840948   -6.828244  -6.726417  -6.552752
##          80          81          82          83
## 1 -868.554261 -862.954172 -853.6460016 -846.0091574
## 2   24.416801    25.330028   25.9323565  26.3757933
## 3    1.895523     1.395065   0.7748969  0.1045569

```

```

## 4 -6.336102 -6.108816 -5.8908881 -5.6766055
##          84          85          86          87
## 1 -837.7479952 -830.694471 -823.090145 -816.250818
## 2  26.9394119  27.757805  28.766559  29.789527
## 3 -0.6149608 -1.414668 -2.293217 -3.199102
## 4 -5.4401626 -5.156260 -4.815748 -4.427861
##          88          89          90          91
## 1 -809.147180 -802.375953 -794.294685 -789.250968
## 2  30.748112  31.609313  32.468732  33.448228
## 3 -4.071051 -4.886667 -5.680802 -6.474096
## 4 -4.010971 -3.581002 -3.150101 -2.734804
##          92          93          94          95
## 1 -781.692159 -775.313754 -768.801147 -761.685907
## 2  34.331350  35.046883  35.642233  36.178200
## 3 -7.220822 -7.887511 -8.502352 -9.106162
## 4 -2.354377 -2.016812 -1.711758 -1.419231
##          96          97          98          99
## 1 -755.428445 -749.3363972 -743.4202914 -737.6904787
## 2  36.783501  37.4663587  38.1594561  38.8302202
## 3 -9.713461 -10.3146418 -10.8953707 -11.4462919
## 4 -1.123011 -0.8178912 -0.5080417 -0.2008162
##          100         101         102         103
## 1 -732.15686097 -726.8285549 -721.7137035 -716.8194612
## 2  39.47047502  40.0762596  40.6441156  41.1709052
## 3 -11.96321904 -12.4438504 -12.8865346 -13.2899578
## 4  0.09794944  0.3846991  0.6573134  0.9140294
##          104         105         106         107
## 1 -712.151561 -707.714246 -703.510233 -699.540392
## 2  41.653917  42.090977  42.479922  42.813956
## 3 -13.653150 -13.974571 -14.250679 -14.473367
## 4  1.152831  1.370158  1.560002  1.715635
##          108         109
## 1 -695.804031 -692.29860
## 2  43.071146  43.23280
## 3 -14.635530 -14.74149
## 4  1.834136  1.91882
##
## $female$aml
##
## Call:
## lm(formula = aml ~ cm + cml + cmcls + cmclc)
##
## Coefficients:
## (Intercept)          cm          cml          cmcls
## 2.57256      -4.23366      2.04570      0.29981
## cmclc
## 0.01917
##
## $female$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmcls + cmclc + am + amls +
##     amlc + cmclaml)

```

```

## 
## Coefficients:
## (Intercept)          cm          cml          cmls
## 7.935e-03    1.214e-02 -3.458e-03 -5.278e-04
##   cmlc          am          amls          amlc
## -4.148e-05   -2.774e-03  3.464e-04 -2.214e-05
##   cmlaml
## -3.719e-04
##
##
## $female$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)          cm          cml          cmls
## -0.220585    0.381119 -0.119156 -0.021188
##   cmlc          am          amls          amlc
## -0.001602   -0.007855  0.011313  0.001510
##   cmlaml
## -0.006717
##
##
## $female$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)          cm          cml          cmls
## -0.401665    0.919834 -0.235462 -0.029920
##   cmlc          am          amls          amlc
## -0.002460   -0.087807  0.022911 -0.003114
##   cmlaml
## -0.042306
##
##
## $female$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)          cm          cml          cmls
## 0.700751   -1.457776  0.399897  0.080301
##   cmlc          am          amls          amlc
## 0.005121    0.035974 -0.009747 -0.002357
##   cmlaml
## -0.002643
##

```

```

##
## $female$offset
## [1] 10
##
## $female$q0
##
## Call:
## lm(formula = as.numeric(q1[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml          cmls
## -0.93877    0.67084   -0.03559
##
## $female$rownames
## [1] "0"   "1"   "2"   "3"   "4"   "5"   "6"   "7"
## [9] "8"   "9"   "10"  "11"  "12"  "13"  "14"  "15"
## [17] "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"
## [25] "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"
## [33] "32"  "33"  "34"  "35"  "36"  "37"  "38"  "39"
## [41] "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"
## [49] "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
## [57] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"
## [65] "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"
## [73] "72"  "73"  "74"  "75"  "76"  "77"  "78"  "79"
## [81] "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"
## [89] "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"
## [97] "96"  "97"  "98"  "99"  "100" "101" "102" "103"
## [105] "104" "105" "106" "107" "108" "109"
##
##
## $male
## $male$components
##          0           1           2           3
## 1 -956.683028 -1112.530322 -1153.077797 -1177.336137
## 2 -48.973365  -83.284877  -77.962563  -76.865301
## 3 -5.157522   -3.482452   -6.335255  -13.780415
## 4  5.084461   -2.171409   -1.135138   4.329985
##          4           5           6           7
## 1 -1192.0186411 -1203.353181 -1212.77562 -1219.720254
## 2 -68.8646781   -65.775828  -65.07063  -61.375585
## 3 -11.6326299   -12.512683  -14.64239  -15.036231
## 4  0.7244526    2.544674   4.78138   7.935642
##          8           9          10          11
## 1 -1226.73418 -1232.815191 -1234.675393 -1235.0685594
## 2 -57.04964   -54.396818  -50.928084  -47.0679761
## 3 -15.04103   -18.178683  -13.518356  -14.5084494
## 4  10.33869    5.413494   2.707134   0.9061641
##          12          13          14
## 1 -1.231447e+03 -1224.5199410 -1211.283163
## 2 -4.323666e+01  -40.0151408  -32.306008
## 3 -1.254985e+01  -10.9595426  -4.997027
## 4  2.804063e-02   0.6962256  -4.118847
##          15          16          17          18

```

```

## 1 -1196.521973 -1177.7109034 -1162.120267 -1146.047507
## 2 -27.280384 -18.3145515 -14.086153 -6.980748
## 3 -3.614796 0.6012524 3.098067 6.364498
## 4 -5.943127 -7.3170809 -6.658601 -10.504674
## 19 20 21 22
## 1 -1139.26607 -1135.513535 -1132.86754 -1131.85493
## 2 -7.65885 -10.635073 -10.77221 -13.12394
## 3 8.84674 8.295672 10.63405 11.07388
## 4 -11.32606 -11.940190 -14.10377 -11.12592
## 23 24 25 26
## 1 -1131.04244 -1131.05179 -1130.73586 -1130.72881
## 2 -12.96892 -12.43570 -11.74041 -10.62887
## 3 12.35584 12.50606 13.92002 13.94918
## 4 -12.21879 -11.94613 -11.53347 -11.84821
## 27 28 29 30
## 1 -1129.940607 -1128.172517 -1126.700947 -1125.264571
## 2 -9.718192 -7.534058 -6.237771 -8.740707
## 3 14.611862 15.657450 15.143741 14.643513
## 4 -11.051913 -9.050074 -8.627944 -8.106670
## 31 32 33 34
## 1 -1123.386165 -1119.891781 -1117.084494 -1114.122933
## 2 -5.056862 -5.317181 -3.888981 -3.820726
## 3 15.444603 14.793862 15.051548 15.007530
## 4 -7.755216 -5.262800 -4.427907 -3.045756
## 35 36 37 38
## 1 -1110.274408 -1106.861198 -1103.469870 -1097.8260010
## 2 -3.459792 -2.173112 -1.871729 0.6381298
## 3 14.719956 13.591889 13.348181 13.6735678
## 4 -1.901663 -1.341920 -2.394314 0.2698918
## 39 40 41 42
## 1 -1093.5378702 -1088.105391 -1084.416036 -1077.954750
## 2 1.7865751 1.488276 4.503435 4.571797
## 3 12.5087346 12.849615 11.853921 10.938876
## 4 0.8942627 3.046146 2.442911 4.406748
## 43 44 45 46
## 1 -1073.040049 -1067.988933 -1061.595089 -1056.891957
## 2 7.215081 7.816829 9.055133 10.782362
## 3 11.097237 10.150313 9.514459 8.964206
## 4 4.140852 4.608158 5.549215 6.098956
## 47 48 49 50
## 1 -1051.292986 -1045.417405 -1039.568475 -1032.958880
## 2 12.107628 12.571774 14.091355 14.016277
## 3 8.446764 7.928164 7.294804 6.861816
## 4 6.706820 7.158142 7.313293 8.109793
## 51 52 53 54
## 1 -1028.956538 -1021.605682 -1016.494746 -1010.288016
## 2 15.984762 16.322277 16.960918 18.171623
## 3 5.964880 5.781441 4.839662 4.457371
## 4 7.468141 8.356141 8.232695 8.562713
## 55 56 57 58
## 1 -1004.834650 -999.094214 -993.682160 -987.290700
## 2 19.396645 19.972118 20.964341 20.867236
## 3 4.198922 3.163065 2.541789 2.072549
## 4 8.335206 8.771672 8.700063 9.541514

```

```

##          59          60          61          62
## 1 -981.438502 -974.075511 -970.202059 -962.661666
## 2  21.317274  20.655561  21.602137  21.045153
## 3   1.931582   2.435186   1.320386   1.183893
## 4   8.878093   9.886551   8.535697   9.424880
##          63          64          65          66
## 1 -956.7873033 -950.6380973 -944.27411290 -939.5504350
## 2  21.1674019  21.2051289  21.07473867  21.7298040
## 3   0.5374691   0.3841729  -0.02618765  -0.5826234
## 4   9.1392741   9.2662895   9.48770271   7.9609967
##          67          68          69          70
## 1 -933.244941 -926.786910 -920.969716 -913.476692
## 2  21.649715  21.597411  21.583412  20.162071
## 3  -1.023380  -1.294976  -2.064466  -1.791464
## 4   8.333647   8.252935   7.425469   8.011817
##          71          72          73          74
## 1 -909.159224 -900.908614 -894.592517 -887.795978
## 2  21.475232  20.238847  20.377545  20.113802
## 3  -3.503345  -2.856335  -3.079839  -3.215672
## 4   6.205964   7.283192   6.863824   6.887553
##          75          76          77          78
## 1 -881.170274 -874.874839 -869.158616 -861.399807
## 2  20.022587  20.279888  20.612890  20.194523
## 3  -3.209937  -4.076569  -5.307226  -4.461321
## 4   6.740595   6.227993   4.596656   5.552574
##          79          80          81          82
## 1 -855.143515 -847.504063 -842.365397 -834.101293
## 2  20.580401  20.610517  21.955339  21.200051
## 3  -4.796680  -4.154607  -4.649873  -5.178356
## 4   4.345664   4.661154   3.166188   2.976475
##          83          84          85          86
## 1 -827.408074 -819.888159 -813.8269191 -806.8709568
## 2  21.480141  21.201664  22.2304394  22.8066093
## 3  -5.689034  -6.238346  -6.6452412  -7.4236907
## 4   2.131509   1.817417   0.7955961   0.1684787
##          87          88          89          90
## 1 -800.5068076 -794.113654 -787.972323 -780.972474
## 2  23.5812789  24.013655  24.994528  24.994489
## 3  -7.9076012  -8.299625  -8.838186  -8.921504
## 4  -0.7776146  -1.523922  -2.582752  -2.877834
##          91          92          93          94
## 1 -776.051646 -769.222733 -763.430113 -757.629111
## 2  26.913013  27.160196  28.036158  28.545374
## 3  -9.805954  -10.221756  -10.724827  -11.170773
## 4  -4.440636  -4.618029  -5.332289  -5.799025
##          95          96          97          98
## 1 -751.531057 -745.998595 -740.625130 -735.418412
## 2  27.860705  28.369197  28.869887  29.360245
## 3  -11.246231 -11.587350 -11.904693 -12.197393
## 4  -6.552194  -7.134007  -7.683178  -8.197783
##          99         100         101         102
## 1 -730.385601 -725.533248 -720.867057 -716.392054
## 2  29.837700  30.299639  30.743473  31.166646
## 3 -12.464671 -12.706112 -12.921308 -13.110159

```

```

## 4 -8.676323 -9.117316 -9.519754 -9.882849
##          103      104      105      106
## 1 -712.11201 -708.02974 -704.14693 -700.46396
## 2  31.56694   31.94226   32.29090   32.61145
## 3 -13.27290 -13.40979 -13.52153 -13.60882
## 4 -10.20623 -10.48986 -10.73414 -10.93982
##          107      108      109
## 1 -696.98012 -693.69351 -690.60107
## 2  32.90308   33.16509   33.39752
## 3 -13.67279 -13.71456 -13.73558
## 4 -11.10810 -11.24047 -11.33879
##
## $male$components.sm
##          0      1      2      3
## 1 -956.683028 -1112.530322 -1153.077797 -1177.336137
## 2 -48.973365  -72.300579  -76.122933  -74.495981
## 3 -5.157522   -3.482452  -8.123748  -10.182584
## 4  5.084461   -2.171409  -1.135138   1.753942
##          4      5      6      7
## 1 -1192.018641 -1203.353181 -1212.775625 -1219.720254
## 2  -70.466010   -66.858732  -64.063716  -61.013759
## 3  -11.853452  -13.034978  -14.003314  -14.826085
## 4   2.583744    3.699854   4.908597   5.780592
##          8      9     10     11
## 1 -1226.734184 -1232.815191 -1234.675393 -1235.068559
## 2  -57.572522  -54.170278  -50.715754  -47.046422
## 3  -15.351540  -15.341414  -14.691272  -13.490593
## 4   5.847793    4.981345   3.493593   1.821091
##          12     13     14     15
## 1 -1231.4472771 -1224.519941 -1211.283163 -1196.521973
## 2  -43.1007441  -38.428082  -32.694400  -26.263361
## 3  -11.7132060  -9.237841  -6.219193  -3.026575
## 4   0.1662591   -1.517014  -3.273533  -5.013804
##          16     17     18     19
## 1 -1.177711e+03 -1162.120267 -1146.047507 -1139.266071
## 2  -1.975176e+01  -14.030655  -10.155840  -9.053989
## 3   9.218528e-02   3.004820   5.525478   7.485633
## 4  -6.631394e+00  -8.117766  -9.488547  -10.669110
##          20     21     22     23
## 1 -1135.513535 -1132.86754 -1131.85493 -1131.04244
## 2  -9.932086  -11.21957  -12.22822  -12.58576
## 3   8.936974   10.10049   11.11307  11.99914
## 4  -11.523143  -11.97546  -12.08377  -11.98845
##          24     25     26     27
## 1 -1131.05179 -1130.73586 -1130.72881 -1129.940607
## 2  -12.27226  -11.55405  -10.56630  -9.325086
## 3   12.77786   13.47108   14.08278  14.591430
## 4  -11.79229  -11.49556  -11.03851  -10.392607
##          28     29     30     31
## 1 -1128.172517 -1126.700947 -1125.264571 -1123.386165
## 2  -8.070607  -7.345624  -6.891527  -6.015615
## 3   14.928546   15.056111   15.058771  15.036337
## 4  -9.604395  -8.743288  -7.820167  -6.794119
##          32     33     34     35

```

```

## 1 -1119.891781 -1117.084494 -1114.122933 -1110.274408
## 2 -5.040687 -4.296933 -3.733423 -3.132495
## 3 15.001173 14.916549 14.722171 14.381011
## 4 -5.661825 -4.503269 -3.433636 -2.516588
## 36 37 38
## 1 -1106.861198 -1103.4698702 -1097.8260010
## 2 -2.304784 -1.1798374 0.1700703
## 3 13.960129 13.5687797 13.2181295
## 4 -1.708909 -0.8872396 0.0410758
## 39 40 41 42
## 1 -1093.537870 -1088.105391 -1084.416036 -1077.954750
## 2 1.362177 2.464434 3.791241 5.217124
## 3 12.840391 12.386616 11.857179 11.300913
## 4 1.050678 2.033606 2.900936 3.634050
## 43 44 45 46
## 1 -1073.040049 -1067.988933 -1061.595089 -1056.891957
## 2 6.643550 7.957694 9.249664 10.598311
## 3 10.748271 10.176848 9.584461 9.002603
## 4 4.269439 4.860853 5.441969 6.006991
## 47 48 49 50
## 1 -1051.292986 -1045.417405 -1039.56847 -1032.958880
## 2 11.802348 12.798412 13.69172 14.577201
## 3 8.442540 7.887788 7.32225 6.743505
## 4 6.526072 6.973191 7.33975 7.633310
## 51 52 53 54
## 1 -1028.956538 -1021.605682 -1016.494746 -1010.288016
## 2 15.517126 16.369770 17.203044 18.176472
## 3 6.161087 5.586212 5.025500 4.473390
## 4 7.872633 8.075944 8.252403 8.408976
## 55 56 57 58
## 1 -1004.834650 -999.094214 -993.682160 -987.290700
## 2 19.163384 19.990117 20.585940 20.911138
## 3 3.900531 3.303721 2.759983 2.359804
## 4 8.561346 8.725199 8.896457 9.047227
## 59 60 61 62
## 1 -981.438502 -974.075511 -970.202059 -962.661666
## 2 21.029829 21.091120 21.179577 21.194576
## 3 2.093526 1.841172 1.504167 1.100458
## 4 9.146455 9.186499 9.188166 9.173462
## 63 64 65 66
## 1 -956.7873033 -950.638097 -944.2741129 -939.5504350
## 2 21.1723388 21.198675 21.3155589 21.4990720
## 3 0.6925032 0.296395 -0.1107664 -0.5398126
## 4 9.1325456 9.026792 8.8322516 8.5703502
## 67 68 69 70
## 1 -933.2449415 -926.786910 -920.969716 -913.476692
## 2 21.5921495 21.506793 21.231200 20.941369
## 3 -0.9758112 -1.410090 -1.851698 -2.297827
## 4 8.2854036 8.000328 7.714124 7.435533
## 71 72 73 74
## 1 -909.159224 -900.908614 -894.592517 -887.795978
## 2 20.781926 20.565155 20.331897 20.193441
## 3 -2.687426 -2.949434 -3.119091 -3.321455
## 4 7.194920 7.009087 6.845410 6.634308

```

```

##          75          76          77          78
## 1 -881.170274 -874.874839 -869.158616 -861.399807
## 2  20.178096  20.284030  20.384536  20.438272
## 3  -3.654128  -4.085334  -4.444075  -4.598832
## 4   6.322411   5.915158   5.465288   5.013549
##          79          80          81          82
## 1 -855.143515 -847.504063 -842.365397 -834.101293
## 2  20.598589  20.946037  21.296492  21.410329
## 3  -4.609762  -4.651509  -4.854882  -5.229580
## 4   4.549621   4.038166   3.466848   2.851938
##          83          84          85          86
## 1 -827.408074 -819.888159 -813.8269191 -806.87095678
## 2  21.438593  21.659568  22.1732861  22.83425959
## 3  -5.705922  -6.225102  -6.7657762  -7.31026312
## 4   2.208378   1.531712   0.8103956   0.04344561
##          87          88          89          90
## 1 -800.506808 -794.113654 -787.97232 -780.972474
## 2  23.495336  24.139595  24.79556  25.554328
## 3  -7.826547  -8.297217  -8.74114  -9.199523
## 4  -0.756502  -1.571546  -2.38448  -3.177639
##          91          92          93          94
## 1 -776.051646 -769.222733 -763.430113 -757.629111
## 2  26.442936  27.240316  27.829157  28.134520
## 3  -9.691490 -10.185424 -10.630967 -11.002156
## 4  -3.930055  -4.626718  -5.272427  -5.887202
##          95          96          97          98
## 1 -751.531057 -745.998595 -740.625130 -735.418412
## 2  28.238828  28.461510  28.875462  29.352934
## 3  -11.313126 -11.601575 -11.889496 -12.171903
## 4  -6.485409  -7.065378  -7.617198  -8.133449
##          99         100         101         102
## 1 -730.385601 -725.533248 -720.867057 -716.392054
## 2  29.828116  30.288461  30.730748  31.152532
## 3  -12.436664 -12.677471 -12.892567 -13.081602
## 4  -8.611587  -9.051396  -9.452706  -9.815182
##          103         104         105         106
## 1 -712.11201 -708.02974 -704.14693 -700.46396
## 2  31.55156  31.92582  32.27361  32.59305
## 3  -13.24472 -13.38228 -13.49466 -13.58208
## 4  -10.13833 -10.42154 -10.66321 -10.86061
##          107         108         109
## 1 -696.98012 -693.69351 -690.60107
## 2  32.87755  33.10479  33.25199
## 3  -13.64477 -13.68491 -13.70796
## 4  -11.01204 -11.12038 -11.19360
##
## $male$aml
##
## Call:
## lm(formula = aml ~ cm + cml + cmcls + cmclc)
##
## Coefficients:
## (Intercept)           cm           cml          cmcls
## -1.6301080  4.7076206 -0.1751449  0.0003132

```

```

##          cmlc
##  0.0094663
##
##
## $male$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmls + cmlc + am + amls +
##      amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm            cml            cmls
##  9.827e-03   8.125e-03  -2.588e-03  -4.824e-04
##          cmlc           am            amls            amlc
##  -3.450e-05  -1.940e-03   9.829e-05  -1.351e-05
##          cmlaml
##  -3.385e-05
##
##
## $male$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##      amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm            cml            cmls
## -0.1662716   0.2704571  -0.0929704  -0.0174491
##          cmlc           am            amls            amlc
##  -0.0012552  -0.0098686   0.0024560   0.0005735
##          cmlaml
##  -0.0012481
##
##
## $male$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##      amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm            cml            cmls
##  0.140969   -0.384067   0.103570   0.023322
##          cmlc           am            amls            amlc
##  0.001459    0.107615  -0.001932  -0.001004
##          cmlaml
##  -0.003587
##
##
## $male$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##      amlc + cmlaml)

```

```

## 
## Coefficients:
## (Intercept)          cm          cml          cmls
## -0.8530559    1.7550057   -0.4927060   -0.0934993
##      cmlc          am          amls          amlc
## -0.0060271   -0.0504524   -0.0035495   -0.0001218
##      cmlaml
## -0.0023309
##
##
## $male$offset
## [1] 10
##
## $male$q0
##
## Call:
## lm(formula = as.numeric(ql[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml          cmls
## -0.81835     0.70002   -0.03481
##
##
## $male$rownames
## [1] "0"   "1"   "2"   "3"   "4"   "5"   "6"   "7"
## [9] "8"   "9"   "10"  "11"  "12"  "13"  "14"  "15"
## [17] "16"  "17"  "18"  "19"  "20"  "21"  "22"  "23"
## [25] "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"
## [33] "32"  "33"  "34"  "35"  "36"  "37"  "38"  "39"
## [41] "40"  "41"  "42"  "43"  "44"  "45"  "46"  "47"
## [49] "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
## [57] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"
## [65] "64"  "65"  "66"  "67"  "68"  "69"  "70"  "71"
## [73] "72"  "73"  "74"  "75"  "76"  "77"  "78"  "79"
## [81] "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"
## [89] "88"  "89"  "90"  "91"  "92"  "93"  "94"  "95"
## [97] "96"  "97"  "98"  "99"  "100" "101" "102" "103"
## [105] "104" "105" "106" "107" "108" "109"

save(file="../RData/mods2024.RData",compress=TRUE,list=c("mods2024"))
# the saved file should be around 15KB !!

```

9 Wrap up

Save the workspace and clear everything

```

save(file=paste("../Rdata/All-SVD-Comp_"
               ,format(Sys.time(), "%Y-%m-%d")
               ," .RData",sep=""),compress=TRUE,list=ls())
rm(list=ls())

# record when this stops
write(as.numeric(format(Sys.time(), "%s")),file="../Rmd/stopped.txt")
# how long did this take?

```

```
started <- scan(file="../Rmd/started.txt")
stopped <- as.numeric(format(Sys.time(), "%s"))
# start, stop, duration
paste("Duration:", seconds_to_period(stopped-started))

## [1] "Duration: 7M 30S"
```