SVD Component Mortality Model

Reproducibility Materials: Data and Code

Samuel J. Clark

2015-2019, 2022

Contents

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 4.2.1 (2022-06-23)"
```

- 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 figures 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'. 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/fltper_1x1"
        value <- list(
          sap.code = 1,
          data.path = path
        )
      } else if (period == 5) {
        path <- "/lt_female/fltper_1x5"
        value <- list(
          sap.code = 2,
          data.path = path
        )
      } else if (period == 10) {
        path <- "/lt_female/fltper_1x10"
        value <- list(
          sap.code = 3,
          data.path = path
        )
      } else {
        value <- list(
          sap.code = 0,
          data.path = ""
        )
      }
    } else if (age == 5) {
```

```r
      if (period == 1) {
        path <- "/lt_female/fltper_5x1"
        value <- list(
          sap.code = 4,
          data.path = path
        )
      } else if (period == 5) {
        path <- "/lt_female/fltper_5x5"
        value <- list(
          sap.code = 5,
          data.path = path
        )
      } else if (period == 10) {
        path <- "/lt_female/fltper_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/mltper_1x1"
        value <- list(
          sap.code = 7,
          data.path = path
        )
      } else if (period == 5) {
        path <- "/lt_male/mltper_1x5"
        value <- list(
          sap.code = 8,
          data.path = path
        )
      } else if (period == 10) {
        path <- "/lt_male/mltper_1x10"
        value <- list(
          sap.code = 9,
          data.path = path
        )
      } else {
```

```r
      value <- list(
        sap.code = 0,
        data.path = ""
      )
    }
  } else if (age == 5) {
    if (period == 1) {
      path <- "/lt_male/mltper_5x1"
      value <- list(
        sap.code = 10,
        data.path = path
      )
    } else if (period == 5) {
      path <- "/lt_male/mltper_5x5"
      value <- list(
        sap.code = 11,
        data.path = path
      )
    } else if (period == 10) {
      path <- "/lt_male/mltper_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) {
```

```r
        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)

}
```

```r
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 the version of HMD from 2018 that was used in the *Demography* paper. 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 <- "September 20, 2022"
```

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.

```r
# 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*.

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

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

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

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

```
## List of 7
##  $ creation.date: chr "Tue Sep 20 18:36:09 2022"
##  $ 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!

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

```
## List of 7
##  $ creation.date: chr "Tue Sep 20 18:36:09 2022"
##  $ female       :List of 50
##   ..$ AUS    :List of 99
##   .. ..$ 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 180
##   .. ..$ 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 99
##   .. ..$ 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 180
##   .. ..$ 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 99
##   .. ..$ 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 180
##   .. ..$ 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 ...
```

```r
# 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* × *lifetable* matrices in the *RData* directory.

```r
# 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 1ax
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
```

```r
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
l5.f <- extract.lt.col(hmd.5x1.list,"female","lx")
save(file="../RData/l5.f.RData",compress=TRUE,list=c("l5.f"))
l5.m <- extract.lt.col(hmd.5x1.list,"male","lx")
save(file="../RData/l5.m.RData",compress=TRUE,list=c("l5.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".

```r
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] "l5.f.RData"    "l5.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 Balarus 1914–1918.

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             246             247             248
## female.BEL.1917 female.BEL.1918
##             249             250
```

```r
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
```

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
```

```
##             246             247             248
## female.BEL.1917 female.BEL.1918
##             249             250
```

```r
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
```

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             246             247             248
## female.BEL.1917 female.BEL.1918
##             249             250
```

```r
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
```

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             246             247             248
## female.BEL.1917 female.BEL.1918
##             249             250
```

```r
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
```

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           246           247           248
## male.BEL.1917 male.BEL.1918
##           249           250
```

```r
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
```

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           246           247           248
## male.BEL.1917 male.BEL.1918
##           249           250
```

```r
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
```

```r
which(is.na(e1.m[1,]))
```

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           246           247           248
## male.BEL.1917 male.BEL.1918
##           249           250
```

```r
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
```

```r
which(is.na(e5.m[1,]))
```

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           246           247           248
## male.BEL.1917 male.BEL.1918
##           249           250
```

```r
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
```

```r
# 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]
l5.m <- l5.m[,-remove]

# verify all is well now
```

```r
q1.f[1,which(is.na(q1.f[1,]))]
```

```
## numeric(0)
```

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

```
## numeric(0)
```

```r
e1.f[1,which(is.na(q1.f[1,]))]
```

```
## numeric(0)
```

```r
e5.f[1,which(is.na(q5.f[1,]))]
```

```
## numeric(0)
```

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

```
## numeric(0)
```

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

```
## numeric(0)
```

```r
e1.m[1,which(is.na(q1.m[1,]))]
```

```
## numeric(0)
```

```r
e5.m[1,which(is.na(q5.m[1,]))]
```

```
## numeric(0)
```

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

```
## [1]  110 4861
```

```r
dim(q1.m)
```

```
## [1]  110 4861
```

```r
dim(q5.f)
```

```
## [1]   23 4861
```

```r
dim(q5.m)
```

```
## [1]   23 4861
```

```r
dim(e1.f)
```

```
## [1]  111 4861
```

```r
dim(e1.m)
```

```
## [1]  111 4861
```

```r
dim(e5.f)
```

```
## [1]   24 4861
```

```r
dim(e5.m)
```

```
## [1]   24 4861
```

```r
dim(a1.f)
```

```
## [1]  111 4861
```

```r
dim(a1.m)
```

```
## [1]  111 4861
```

```r
dim(a5.f)
```

```
## [1]   24 4861
```

```r
dim(a5.m)
```

```
## [1]   24 4861
```

```r
dim(l1.f)
```

```
## [1]  111 4861
```

```r
dim(l1.m)
```

```
## [1]  111 4861
```

```r
dim(l5.f)
```

```
## [1]   24 4861
```

```r
dim(l5.m)
```

```
## [1]   24 4861
```

```r
# 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
```

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

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

```r
identical(str_sub(colnames(e1.f),8,),str_sub(colnames(e1.m),6,))
```

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

```r
identical(str_sub(colnames(e5.f),8,),str_sub(colnames(e5.m),6,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.f),8,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.m),6,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.f),8,))
```

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

```r
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"))
```

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
##               2876               3849
## female.NZL_MA.1956 female.NZL_MA.1959
##               3856               3859
```

```
# 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          0.10649            0.11516
## 81          0.10649            0.11516
## 82          0.10649            0.11516
## 83          0.10649            0.11516
## 84          0.10649            0.11516
## 85          0.10649            0.11516
## 86          0.10649            0.11516
## 87          0.10649            0.11516
## 88          0.10649            0.11516
## 89          0.10649            0.11516
## 90          0.10649            0.11516
## 91          0.10649            0.11516
## 92          0.10649            0.11516
## 93          0.10649            0.11516
## 94          0.10649            0.11516
## 95          0.10649            0.11516
## 96          0.10649            0.11516
```

```
## 97          0.10649           0.11516
## 98          0.10649           0.11516
## 99          0.10649           0.11516
## 100         0.10649           0.11516
## 101         0.10649           0.11516
## 102         0.10649           0.11516
## 103         0.10649           0.11516
## 104         0.10649           0.11516
## 105         0.10649           0.11516
## 106         0.10649           0.11516
## 107         0.10649           0.11516
## 108         0.10649           0.11516
## 109         0.10649           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
```

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

```
##    female.ISL.1852 female.NZL_MA.1949
##               2876               3849
## female.NZL_MA.1956 female.NZL_MA.1959
##               3856               3859
```

```
# 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           0.43050            0.45759
## 85-89           0.43050            0.45759
## 90-94           0.43050            0.45759
## 95-99           0.43050            0.45759
## 100-104         0.43050            0.45759
## 105-109         0.43050            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
```

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

```
## named integer(0)
```

```r
# 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
```

```r
# remove all flat LTs that were flat for either females or males
# from 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
```

```r
# remove them from both sexes ane verify
q1.f <- q1.f[,-remove.1]
which(q1.f[106,]==q1.f[110,])
```

```
## named integer(0)
```

```r
q1.m <- q1.m[,-remove.1]
which(q1.m[106,]==q1.m[110,])
```

```
## named integer(0)
```

```r
q5.f <- q5.f[,-remove.5]
which(q5.f[23,]==q5.f[19,])
```

```
## named integer(0)
```

```r
q5.m <- q5.m[,-remove.5]
which(q5.m[23,]==q5.m[19,])
```

```
## named integer(0)
```

```r
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 manes
dim(q1.f)
```

```
## [1]  110 4857
```

```r
dim(q1.m)
```

```
## [1]  110 4857
```

```r
dim(q5.f)
```

```
## [1]   23 4857
```

```r
dim(q5.m)
```

```
## [1]   23 4857
```

```r
dim(e1.f)
```

```
## [1]  111 4857
```

```r
dim(e1.m)
```

```
## [1]  111 4857
```

```r
dim(e5.f)
```

```
## [1]   24 4857
```

```r
dim(e5.m)
```

```
## [1]   24 4857
```

```r
dim(a1.f)
```

```
## [1]  111 4857
```

```r
dim(a1.m)
```

```
## [1]  111 4857
```

```r
dim(a5.f)
```

```
## [1]   24 4857
```

```r
dim(a5.m)
```

```
## [1]   24 4857
```

```r
dim(l1.f)
```

```
## [1]  111 4857
```

```r
dim(l1.m)
```

```
## [1]  111 4857
```

```r
dim(l5.f)
```

```
## [1]   24 4857
```

```r
dim(l5.m)
```

```
## [1]   24 4857
```

```r
# 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
```

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

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

```r
identical(str_sub(colnames(e1.f),8,),str_sub(colnames(e1.m),6,))
```

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

```r
identical(str_sub(colnames(e5.f),8,),str_sub(colnames(e5.m),6,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.f),8,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(e1.m),6,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(q5.f),8,))
```

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

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

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

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

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

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

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l5.f),8,))
```

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

```r
identical(str_sub(colnames(q1.f),8,),str_sub(colnames(l5.m),6,))
```

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

```r
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.

```r
length(q1.f) # values in q1.f
```

```
## [1] 534270
```

```r
length(q1.f[q1.f==0]) # zero cells in q1.f
```

```
## [1] 3071
```

```r
length(q5.f) # values in q5.f
```

```
## [1] 111711
```

```r
length(q5.f[q5.f==0]) # zero cells in q5.f
```

```
## [1] 93
```

```r
length(q1.m) # values in q1.m
```

```
## [1] 534270
```

```r
length(q1.m[q1.m==0]) # zero cells in q1.m
```

```
## [1] 1666
```

```r
length(q5.m) # values in q5.m
```

```
## [1] 111711
```

```r
length(q5.m[q5.m==0]) # zero cells in q5.m
```

```
## [1] 39
```

```r
# female
q1.f.nz <- q1.f
q1.f.nz[q1.f.nz==0] <- 0.000001
```

```r
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")
```

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

```
## [1] 534270
```

```r
length(q1.f.nz[q1.f.nz==0]) # zero cells in q1.f
```

```
## [1] 0
```

```r
length(q5.f.nz) # values in q5.f
```

```
## [1] 111711
```

```r
length(q5.f.nz[q5.f.nz==0]) # zero cells in q5.f
```

```
## [1] 0
```

```r
length(q1.m.nz) # values in q1.m
```

```
## [1] 534270
```

```r
length(q1.m.nz[q1.m.nz==0]) # zero cells in q1.m
```

```
## [1] 0
```

```r
length(q5.m.nz) # values in q5.m
```

```
## [1] 111711
```

```r
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.

```r
# 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
q1l.f <- log(q1.f.nz)
q1logit.f <- logit(q1.f.nz)
q5l.f <- log(q5.f.nz)
q5logit.f <- logit(q5.f.nz)
```

```r
# log and logit transform the male nqx
q1l.m <- log(q1.m.nz)
q1logit.m <- logit(q1.m.nz)
q5l.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.

```r
dim(q1l.f)
```

```
## [1]  110 4857
```

```r
dim(q1logit.f)
```

```
## [1]  110 4857
```

```r
dim(q5l.f)
```

```
## [1]   23 4857
```

```r
dim(q5logit.f)
```

```
## [1]   23 4857
```

```r
dim(q1l.m)
```

```
## [1]  110 4857
```

```r
dim(q1logit.m)
```

```
## [1]  110 4857
```

```r
dim(q5l.m)
```

```
## [1]   23 4857
```

```r
dim(q5logit.m)
```

```
## [1]   23 4857
```

2.3 Additional Indicator Calculation

We need child, ${}_{5}q_{0}$, and adult, ${}_{4}5q_{1}5$, mortality values for females and males. Calculate these from the 1×1 ${}_{n}q_{x}$ values and store in separate matrices, including the log and logit transformed values.

```r
# 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)
}
```

```r
# 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)
```

```r
# log and logit
Ql.f <- log(Q.f)
Qlogit.f <- logit(Q.f)

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

rownames(Q.f) <- c("Child Mortality","Adult Mortality")
rownames(Ql.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)
```

```r
# log and logit
Ql.m <- log(Q.m)
Qlogit.m <- logit(Q.m)

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

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

```r
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.

```r
# 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 *q0 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 predicitons 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(0) # sampled qls
    ret.ql.nsamp <- list(0) # out of sample qls
    ret.samp.names <- list(0) # sampled LT names
    ret.svd <- list(0) # svd of sampled qls
    ret.svd.sm <- list(0) # smooth svd of samlped qls
    ret.mods <- list(0) # models
    ret.recon.samp <- list(0) # sample reconstructions
    ret.error.samp <- list(0) # sample errors
    ret.recon.nsamp <- list(0) # out of sample reconstructions
    ret.error.nsamp <- list(0) # out of sample errors
    ret.errsum.samp <- list(0) # summary of sample errors
    ret.errsum.nsamp <- list(0) # 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 <- Ql[1,samp] # child mortality, logged
```

```r
cmls <- cml^2 # square of logged child mortality
cmlc <- cml^3 # cube of logged child mortality

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

# calcualte 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 + cmlc)

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

# predictions for all LTs, both sampled and out of sample
# start by transforming child mortality for all LTs
cml.p <- Ql[1,]
cm.p <- expit(cml.p)
cmls.p <- cml.p^2
cmlc.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,cmlc.p))
# names for predictors that match the variable in the original model
colnames(predictors.aml) <- c("cm","cml","cmls","cmlc")
# predictions for adult mortality
if (adult) {
    aml.p <- Ql[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)
amls.p <- aml.p^2
amlc.p <- aml.p^3
cmlaml.p <- cml.p*aml.p
# data frame of predictors
predictors.vs <- data.frame(cbind(
  cm.p,cml.p,cmls.p,cmlc.p,am.p,amls.p,amlc.p,cmlaml.p))
# names for predictors that match the variables in the original regressions
colnames(predictors.vs) <- c(
  "cm","cml","cmls","cmlc","am","amls","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 <- Ql[1,samp] # sample child mortality
    cmls <- cml^2 # square of sample child mortality
    q0.betas <- lm(as.numeric(ql[1,samp]) ~ cml + cmls) # q0 ~ cml + cmls
    # predictors for all LTs
    cml.p <- Ql[1,] # child mortality for all LTs
    cmls.p <- cml.p^2 # square of child mortality for all LTs
    predictors.q0 <- data.frame(cbind(cml.p,cmls.p))
    colnames(predictors.q0) <- c("cml","cmls")
    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(ql[,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(ql[,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, ${}_45q_{15}$

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

```

```r
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 resutls 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:l])
}

# 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 retured: 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 retured: 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: 1.09948301315308"
```

```
print(paste("Apply way:",stop.time.apply-start.time.apply))
```

```
## [1] "Apply way: 1.11148500442505"
```

```
# 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 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) {
```

```r
# 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 <- aperm(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]
```

```
}
q5l.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]
}
q5l.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[,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[,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
    ,l5.lq.f = l5.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
    ,l5.lq.m = l5.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 ${}_45q_{15}$ 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
#   motality 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        1521.297     0.01361815  0.330284
## lq          1587.174     0.01420786  0.396844
```

```
comps.child$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1757.213     0.01572999 0.3907967
## lq          1884.154     0.01686633 0.3792040
```

```
cat("\n")
```

```
comps.adult$female
```

```
##      total.abs.error mean.abs.error max.error
## comp        1356.626     0.01214407 0.2188245
## lq          1479.815     0.01324681 0.3865020
```

```
comps.adult$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1439.749     0.01288816 0.3960295
## lq          1556.590     0.01393408 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,Ql.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)
```

```r
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(q1logit.m,Ql.m,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.m <- matrix(data=0,ncol=2,nrow=N)
errsum.iqrs.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.iqrs.1.m[i,1] <-
      qlPred_0.1.m$errsum.samp[[i]][5] - qlPred_0.1.m$errsum.samp[[i]][2]
    errsum.iqrs.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.iqrs.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.iqrs.3.m[i,1] <-
      qlPred_0.3.m$errsum.samp[[i]][5] - qlPred_0.3.m$errsum.samp[[i]][2]
    errsum.iqrs.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.iqrs.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.iqrs.5.m[i,1] <-
      qlPred_0.5.m$errsum.samp[[i]][5] - qlPred_0.5.m$errsum.samp[[i]][2]
    errsum.iqrs.5.m[i,2] <-
      qlPred_0.5.m$errsum.nsamp[[i]][5] - qlPred_0.5.m$errsum.nsamp[[i]][2]
}
```

```r
# 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 × age relationships among ${}_5q_x$ for all ages and both sexes and save as (very large) PDF files.

```r
# 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
```

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

```r
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
```

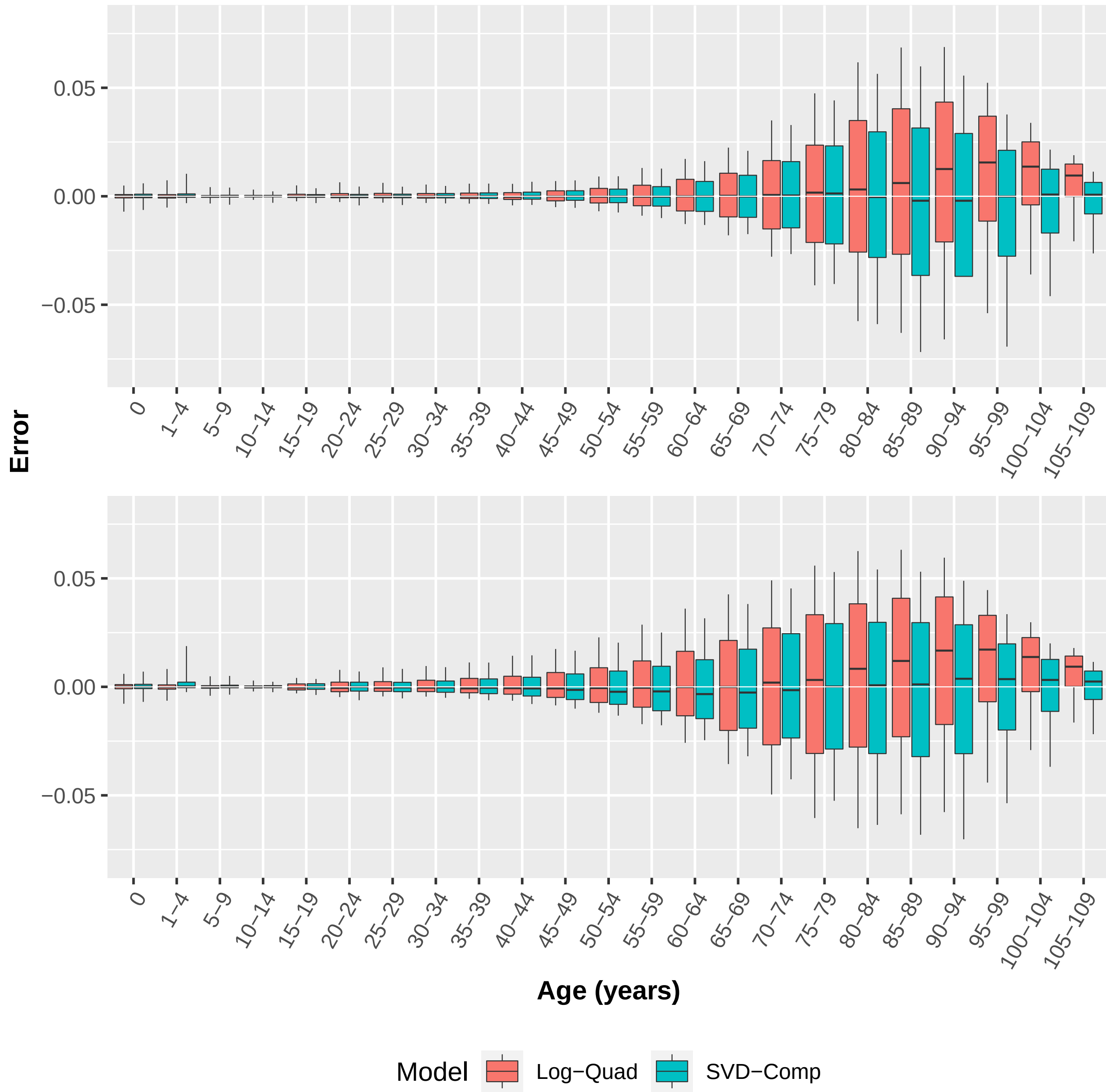
```r
rm(list=c("nr","i","j"))
```

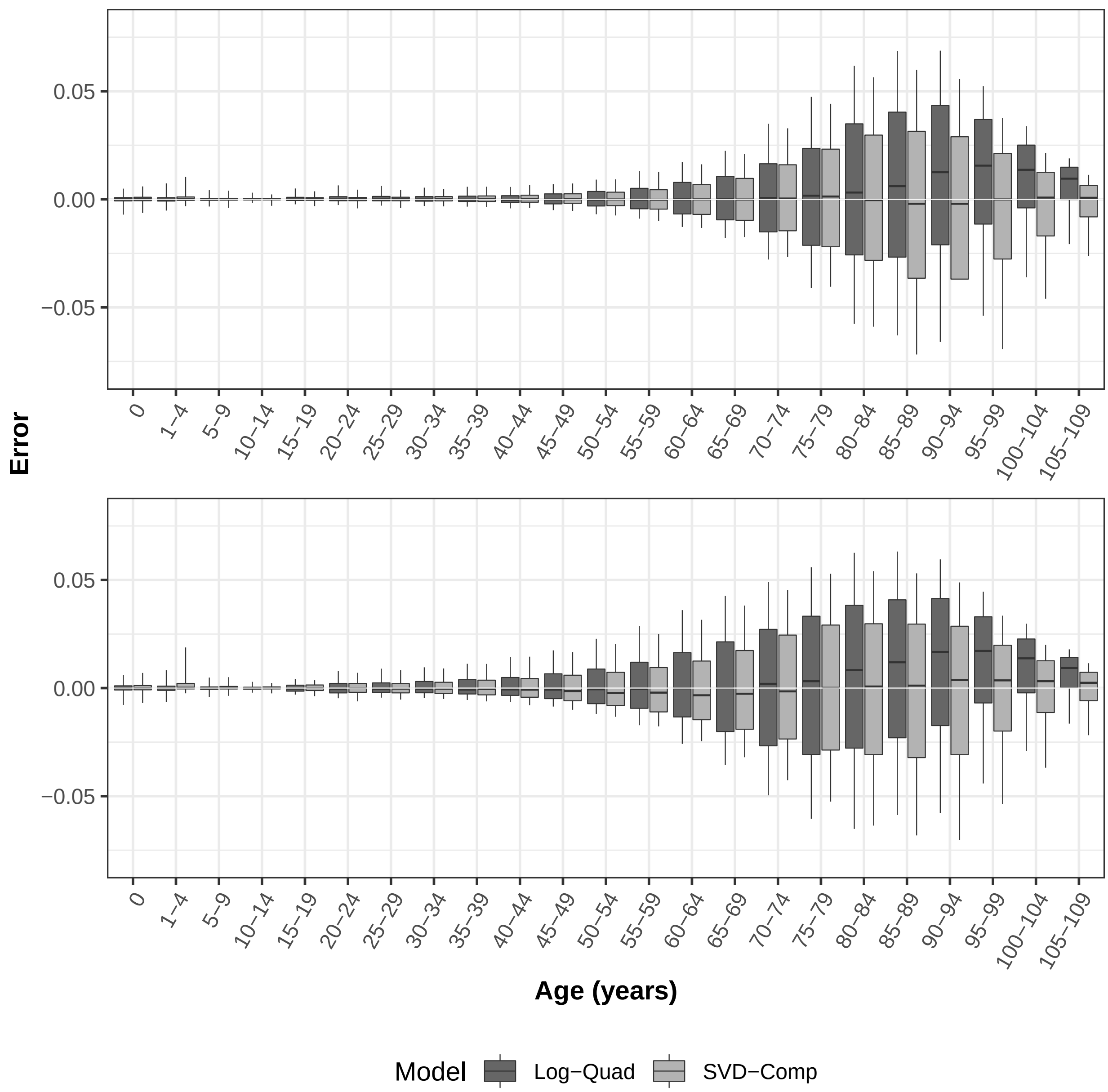
Plot the SVD-comp and Log Quad error distributions by sex and age using 25%, 50%, and 75% quantiles and whiskers to 10% and 90%.

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

```r
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",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",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
```

```r
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.

```r
# 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
##            2845
```

```r
best.m <- which(tot.abs.err.m==min(tot.abs.err.m))
best.m
```

```
## male.FRACNP.1970 male.FRATNP.1970
##             1647             1852
```

```r
# 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] 4394
```

```r
swe.m.1751 <- which(colnames(q1.m)=="male.SWE.1751")
swe.m.1751
```

```
## [1] 4394
```

```r
# 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
```

```r
# 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] 143
```

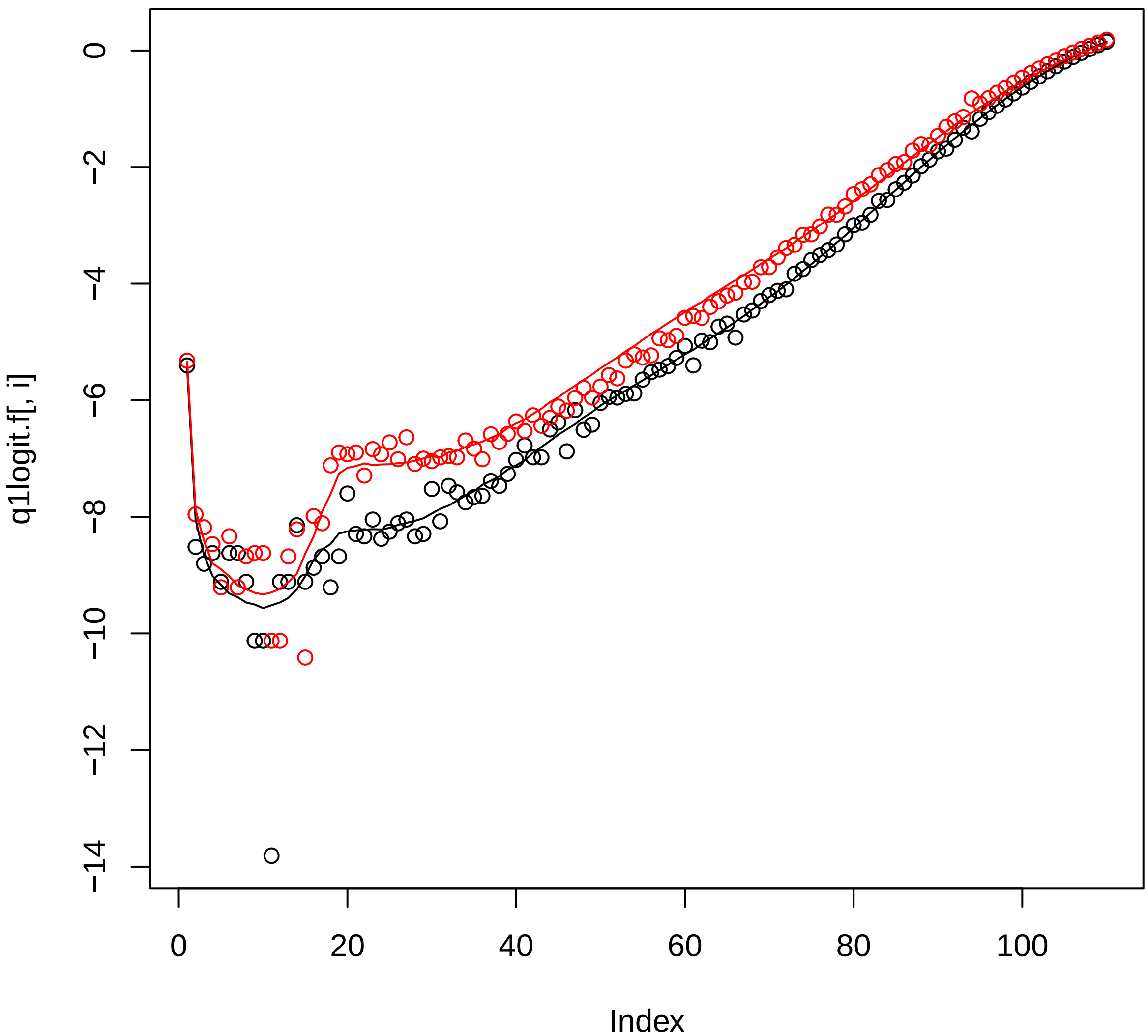
```r
aut.m.1990 <- which(colnames(q1.m)=="male.AUT.1990")
aut.m.1990
```

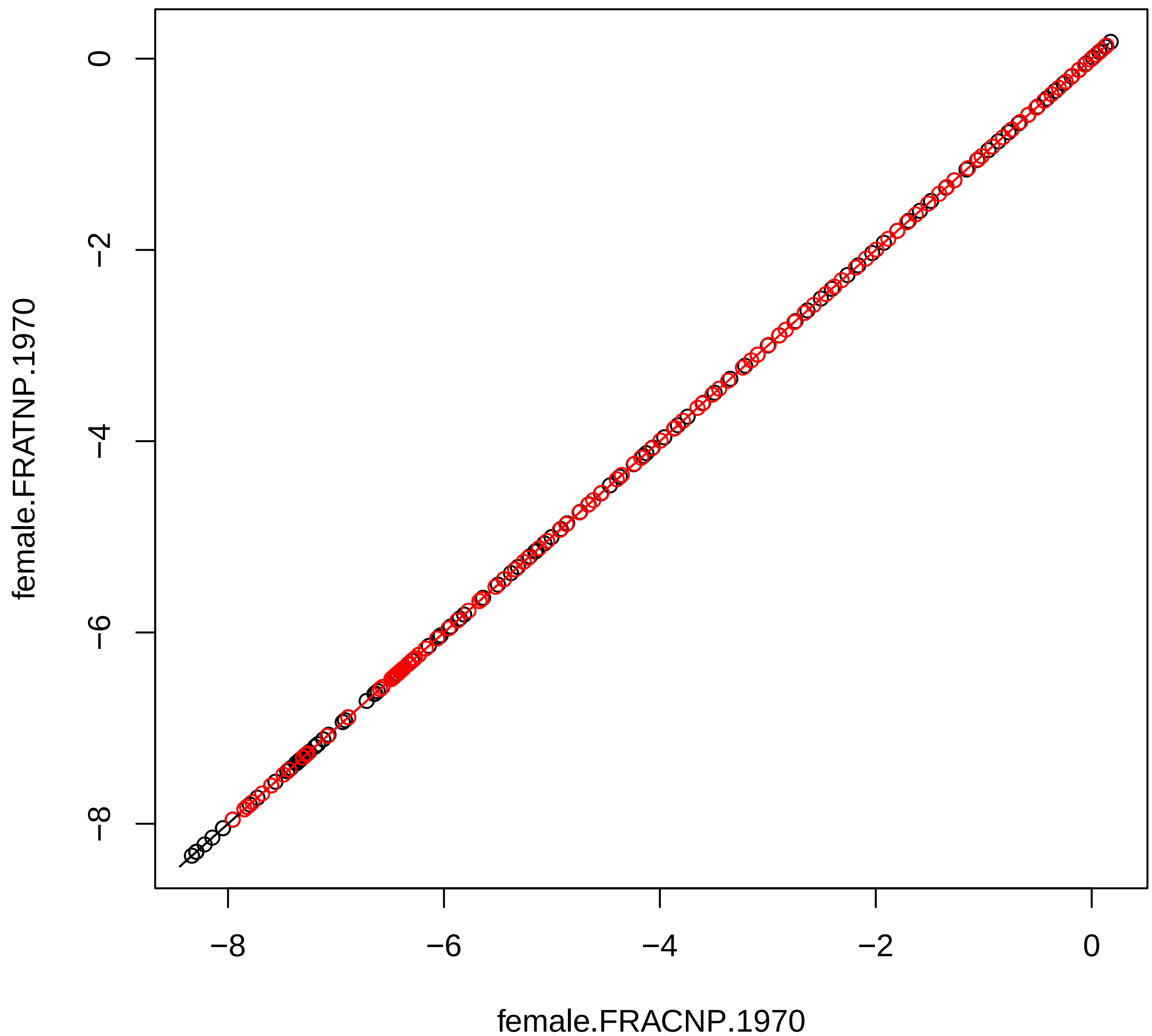
```
## [1] 143
```

```r
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:
```

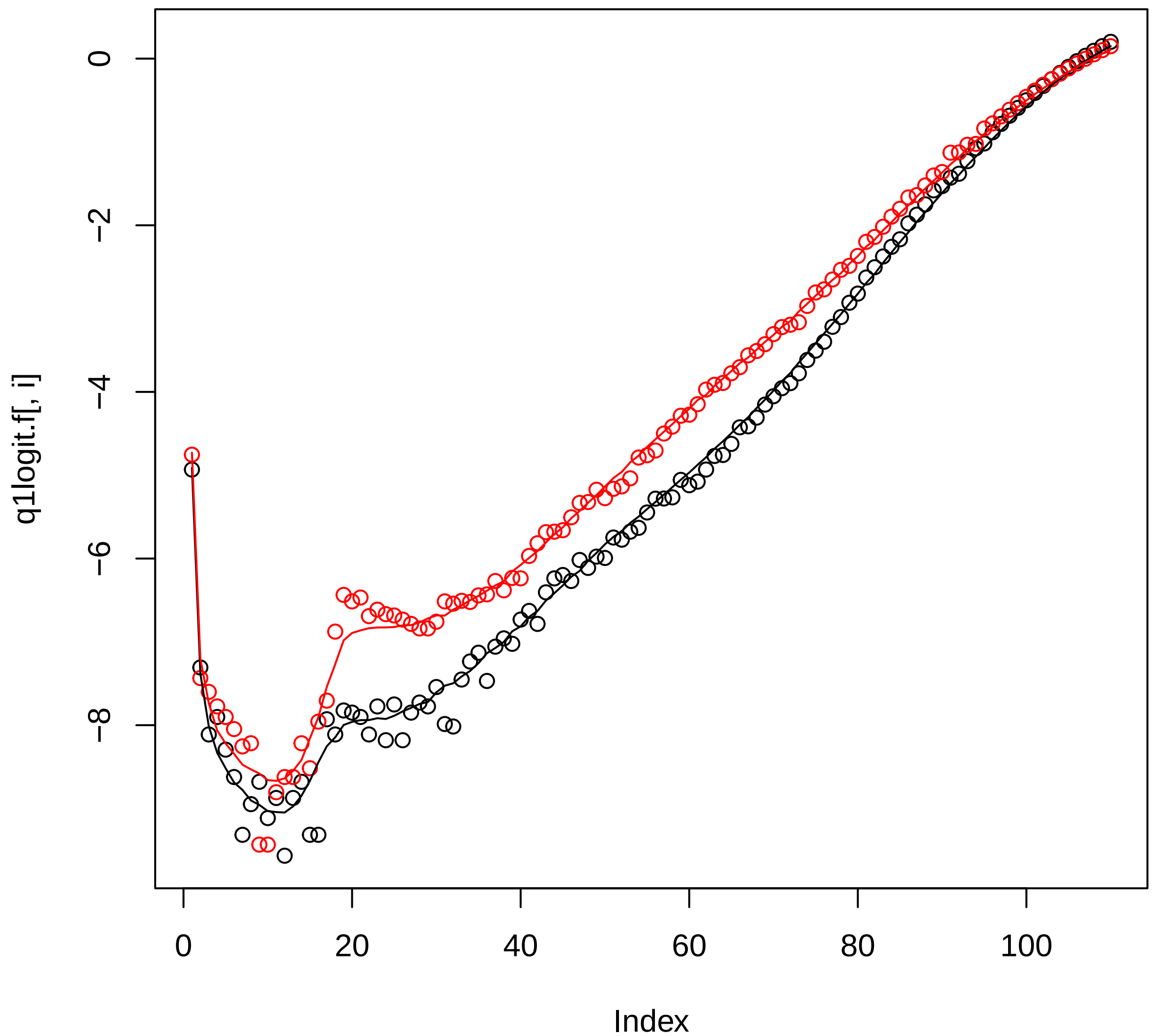
```
i.low <- aut.f.1990
i.high <- swe.f.1751
tmp.1.m <- cbind(rep("Male",110),rownames(q1logit.m)
                 ,rep("Sweden, 1751",110),"Data",q1logit.m[,i.high])
tmp.1.f <- cbind(rep("Female",110),rownames(q1logit.m)
                 ,rep("Sweden, 1751",110),"Data",q1logit.f[,i.high])
tmp.2.m <-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

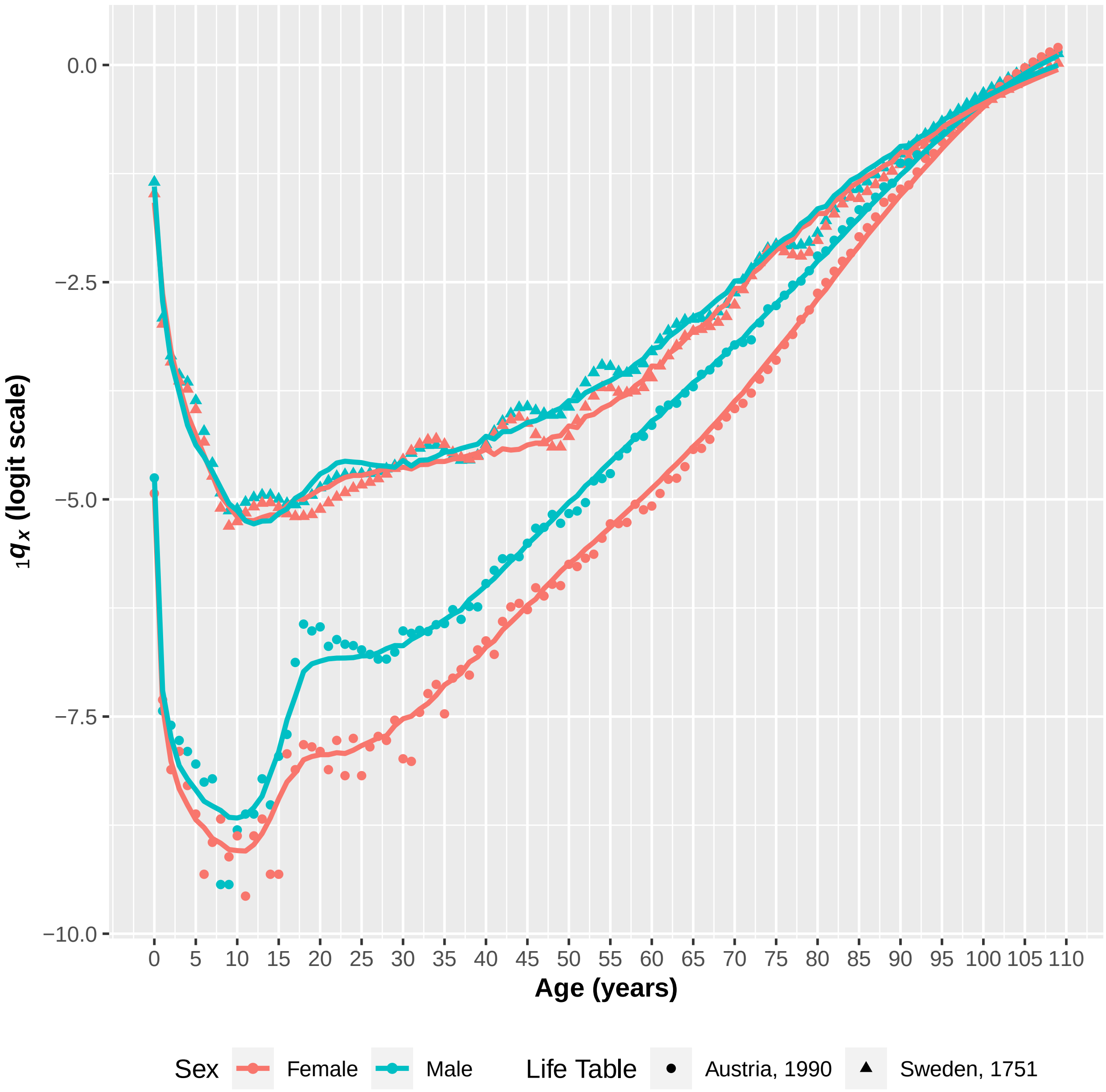
```r
    # theme(legend.justification=c(1,0), legend.position=c(0.98,0.02)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
    scale_shape_discrete(name = "Life Table")
ggsave("../figures/fig1.pdf",width=6.5,height=6.5,units=c("in"))

# 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"))
```


0
−2
−4
−6
−8
−10
−12
−14
q1logit.f[, i]
0
20
40
60
80
100
Index

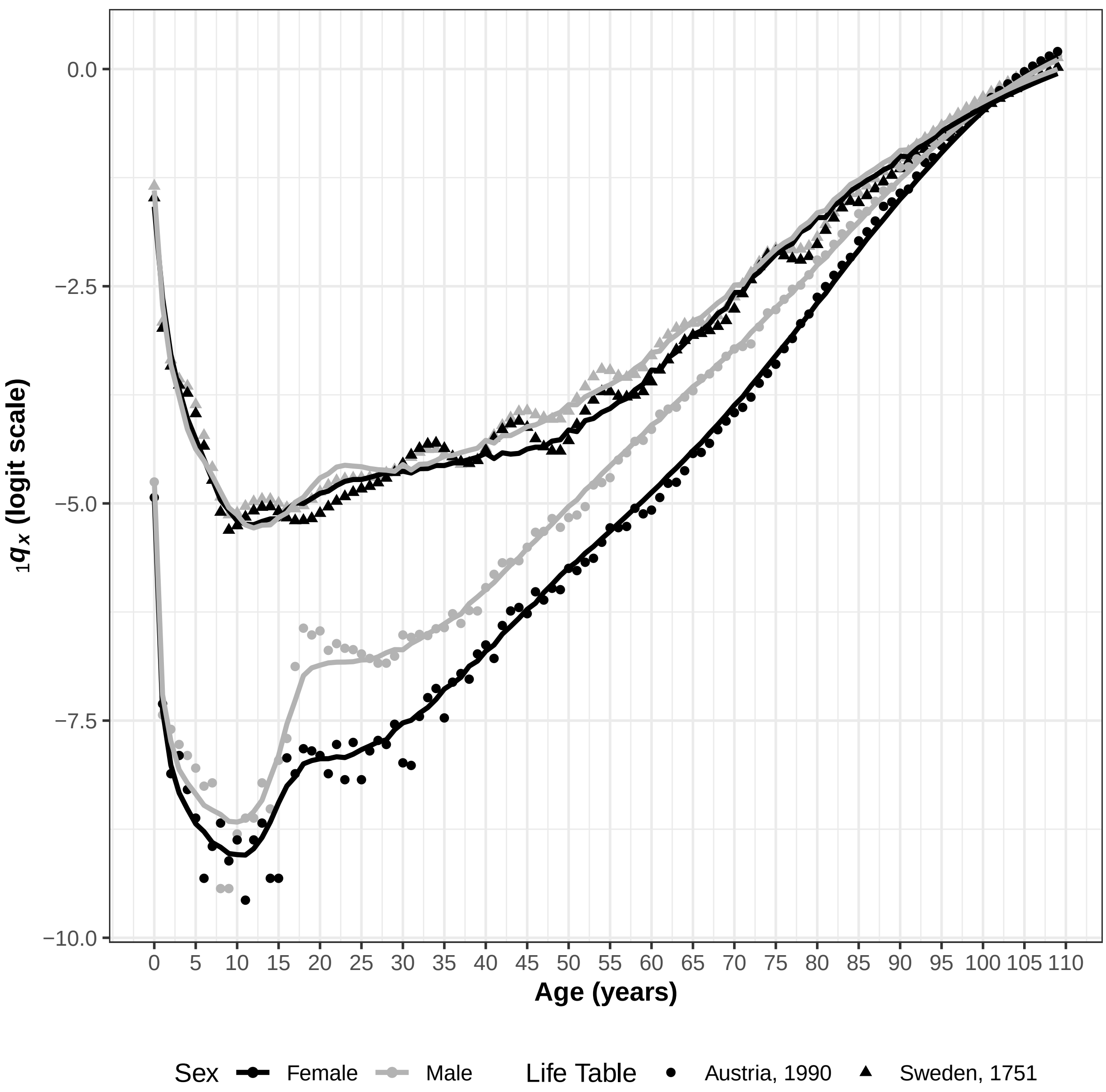


0
−2
−4
−6
−8
female.FRATNP.1970
−8
−6
−4
−2
0
female.FRACNP.1970


q1logit.f[, i]
0
−2
−4
−6
−8
0
20
40
60
80
100
Index



$_1q_x$ (logit scale)
0.0
−2.5
−5.0
−7.5
−10.0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110
Age (years)
Sex
Female
Male
Life Table
Austria, 1990
Sweden, 1751



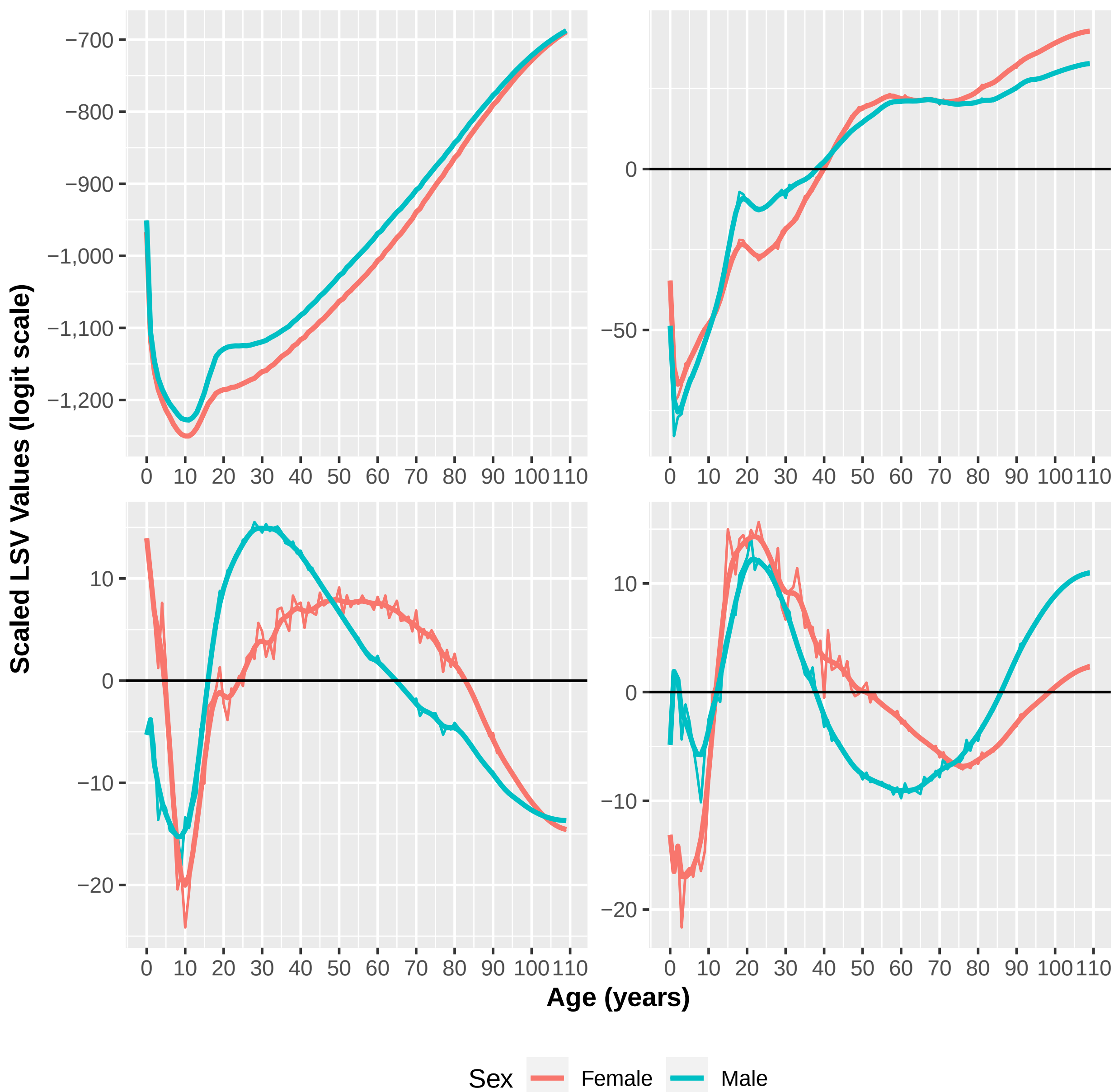
Plot the scaled left singular vectors of the SVD decompositions of logit-transformed $_1q_x$.

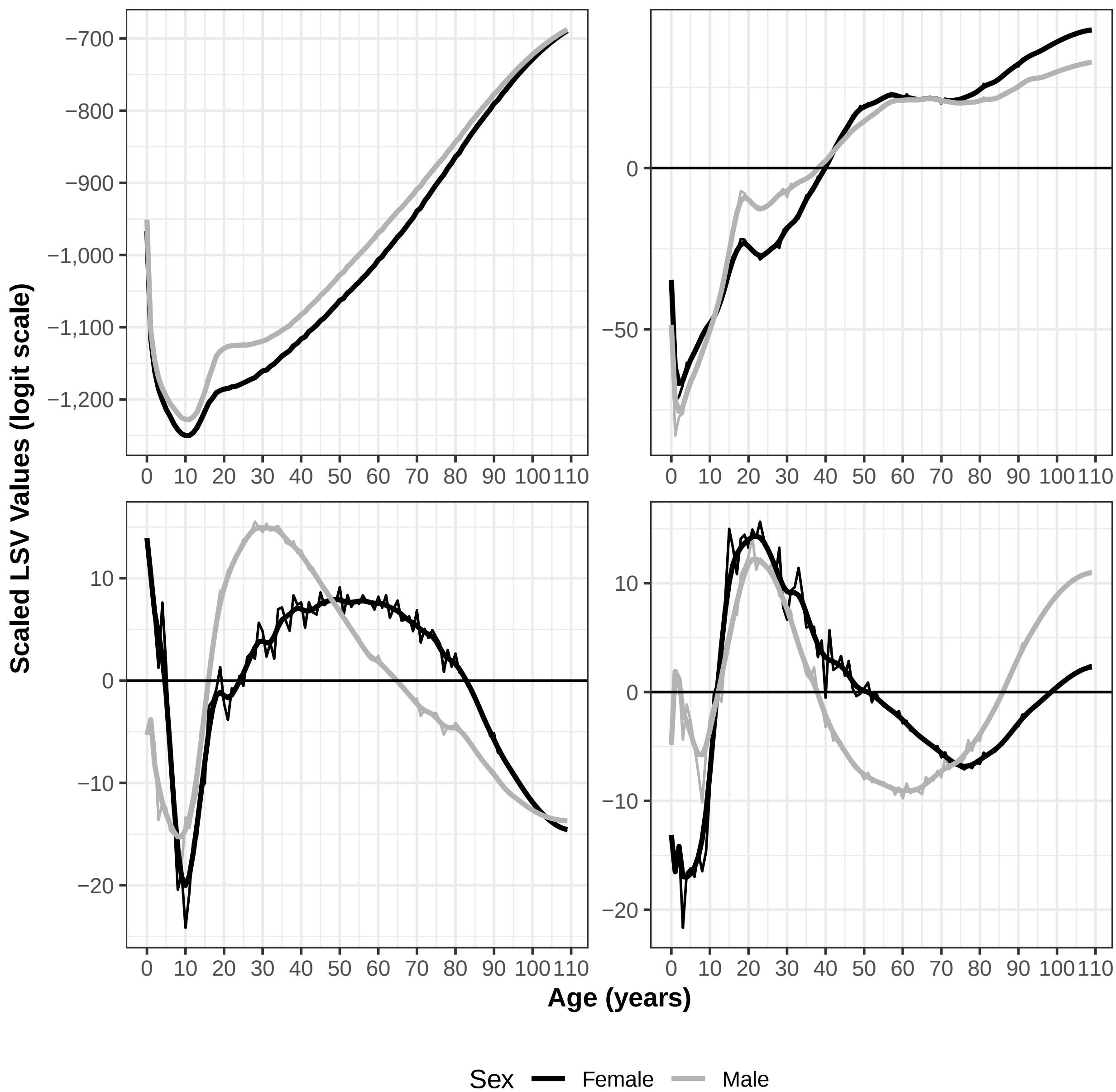
```
# svds
svd.m <- mod.1_0.m$svd$s1
svd.f <- mod.1_0.f$svd$s1

# 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", 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", 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"))
```



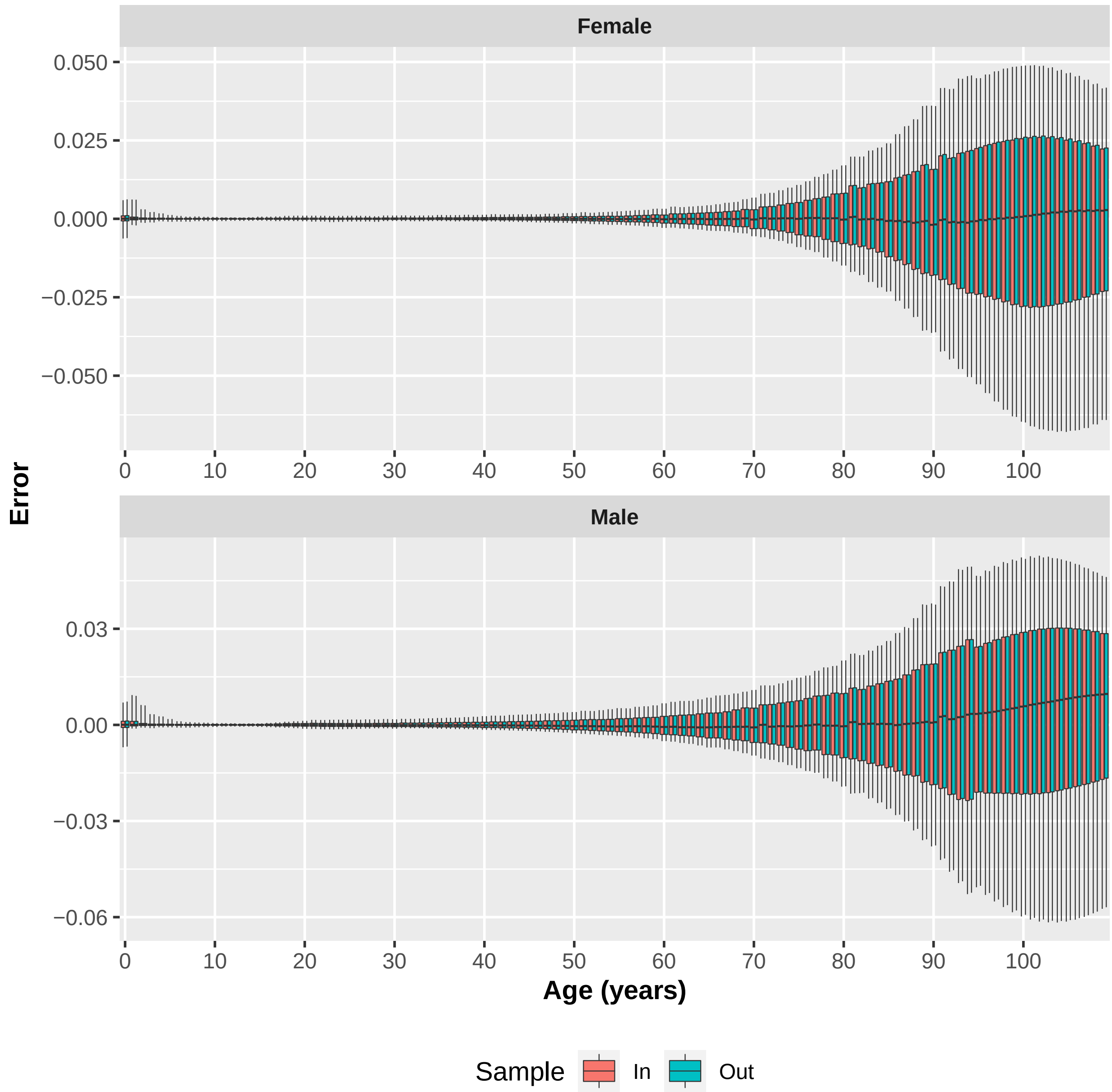


Plot the single-year prediction error distributions from 50 50% samples.

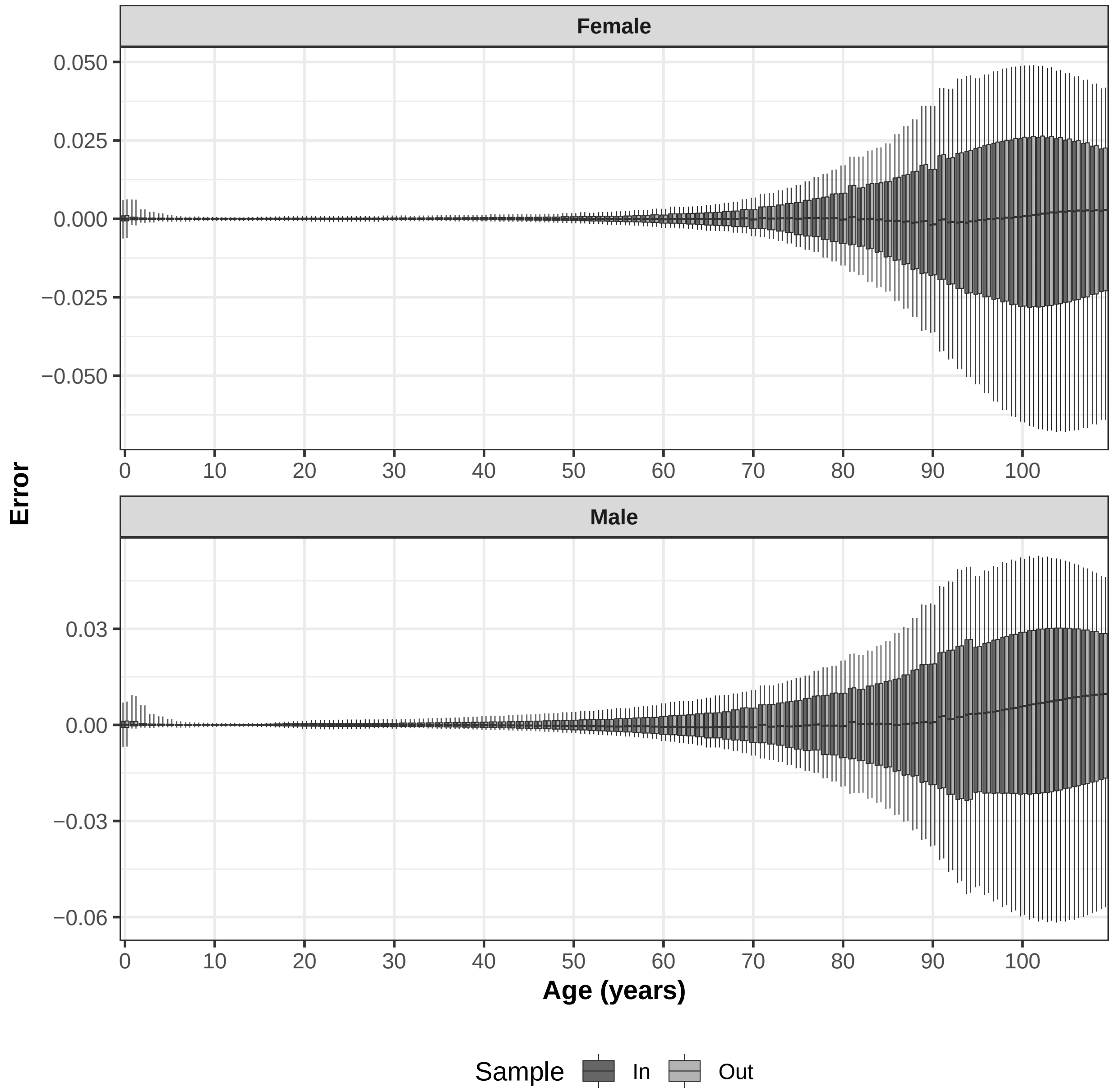
```
# female
esf <- melt(error.age.f)
esf <- cbind(esf[,c(1,3)],"In","Female")
colnames(esf) <- c("Age","Error","Sample","Sex")

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"))
```

```r
# 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,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])
}

# 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"))
```



Female
0.050
0.025
0.000
−0.025
−0.050
0
10
20
30
40
50
60
70
80
90
100
Error
Male
0.03
0.00
−0.03
−0.06
Age (years)
Sample
In
Out



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]),
```

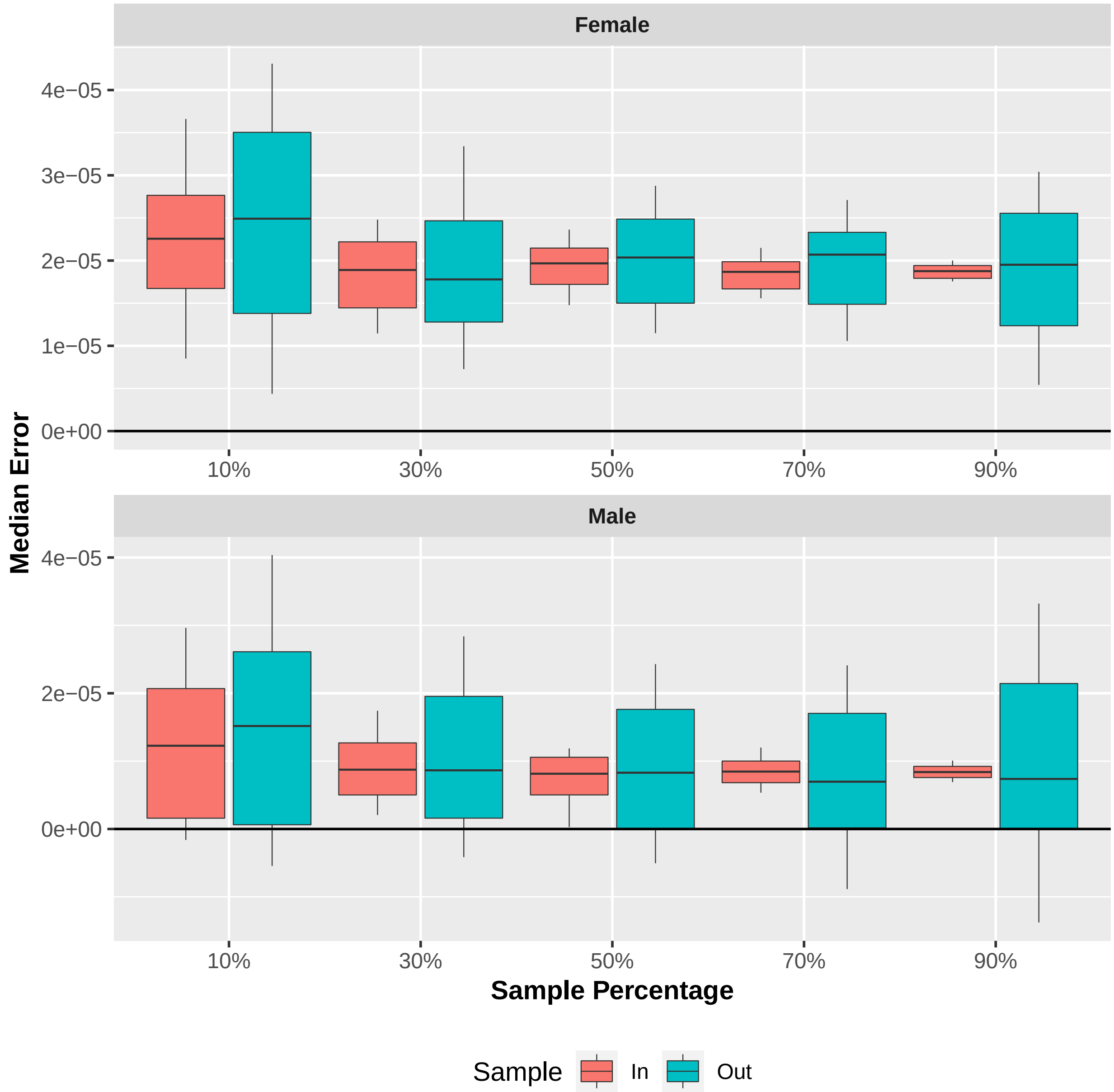
```r
    cbind("Female","Out","50%",errsum.meds.5.f[,2]),
    cbind("Female","Out","70%",errsum.meds.7.f[,2]),
    cbind("Female","Out","90%",errsum.meds.9.f[,2])
)

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),
```

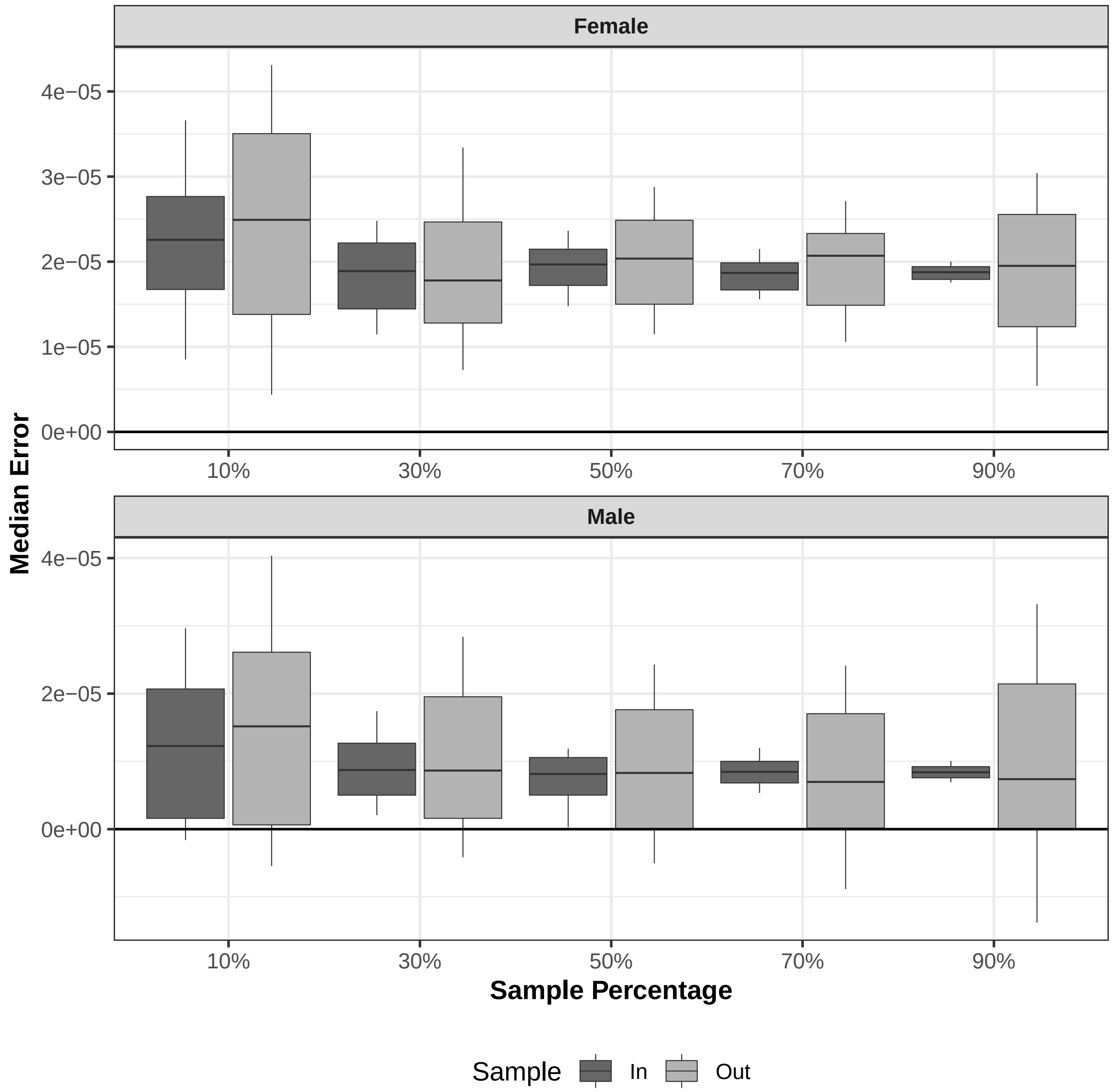
```r
    middle = median(Median),
    lower = quantile(Median,0.25),
    upper = quantile(Median,0.75)
)

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
```

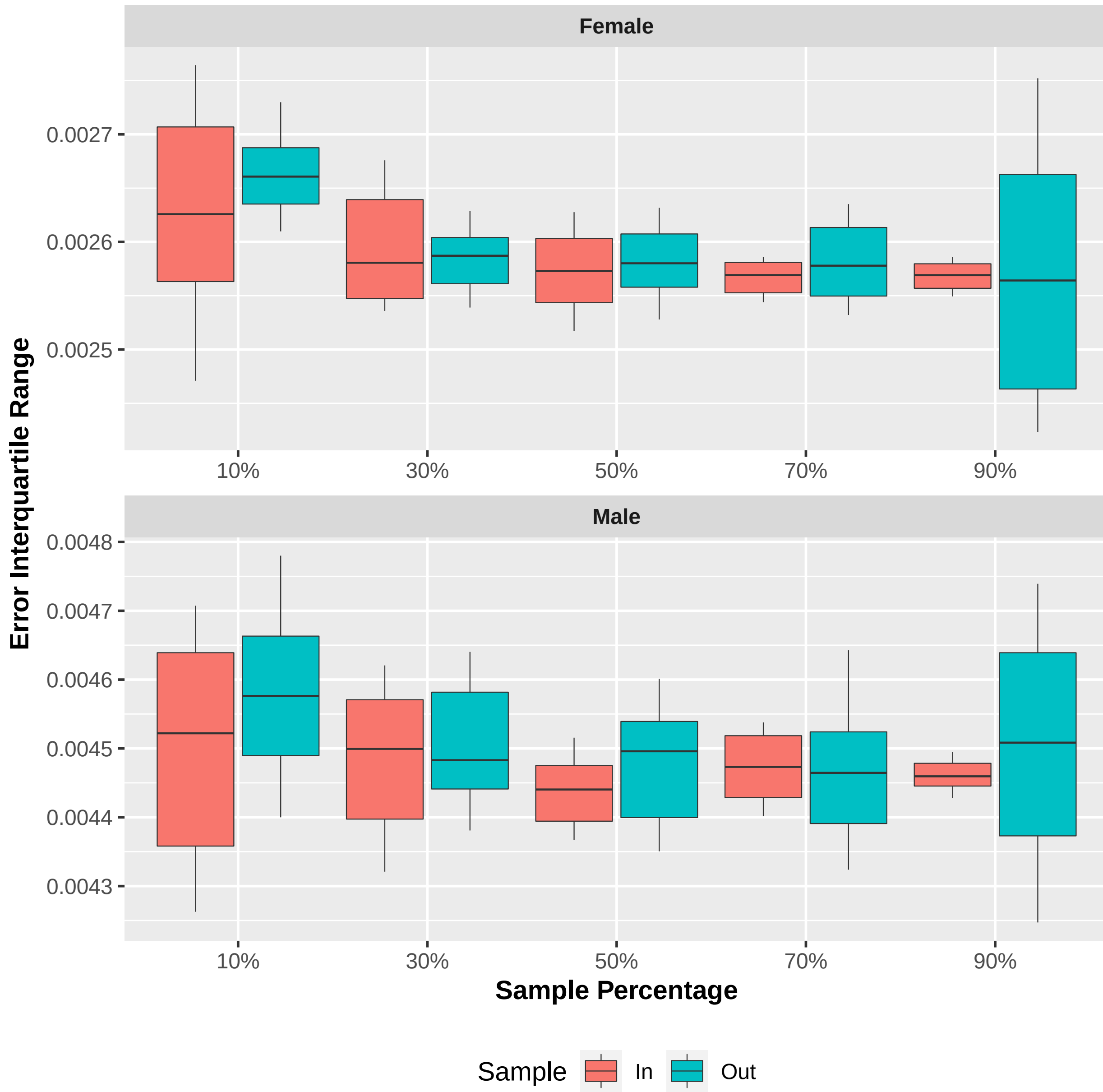
```
# female
es.s.f <- rbind(
    cbind("Female","In","10%",errsum.iqrs.1.f[,1]),
    cbind("Female","In","30%",errsum.iqrs.3.f[,1]),
    cbind("Female","In","50%",errsum.iqrs.5.f[,1]),
    cbind("Female","In","70%",errsum.iqrs.7.f[,1]),
    cbind("Female","In","90%",errsum.iqrs.9.f[,1])
)

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)
```

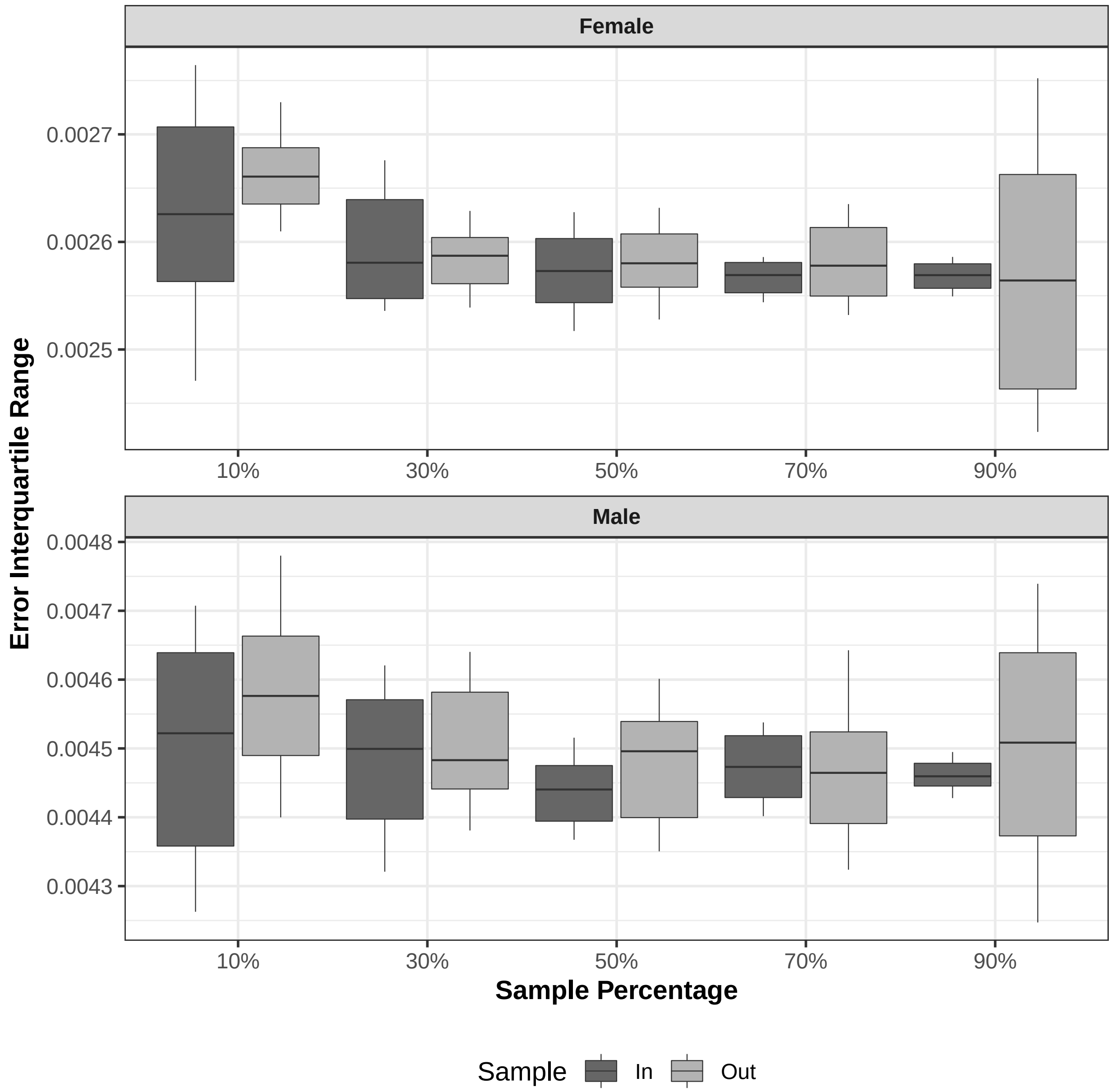
```r
es.df <- rbind(es.m.df,es.f.df)

# 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"))
```




Female
Male
4e−05
3e−05
2e−05
1e−05
0e+00
10%
30%
50%
70%
90%
Median Error
Sample Percentage
Sample
In
Out


Female
Male
Error Interquartile Range
0.0027
0.0026
0.0025
0.0048
0.0047
0.0046
0.0045
0.0044
0.0043
10%
30%
50%
70%
90%
Sample Percentage
Sample
In
Out



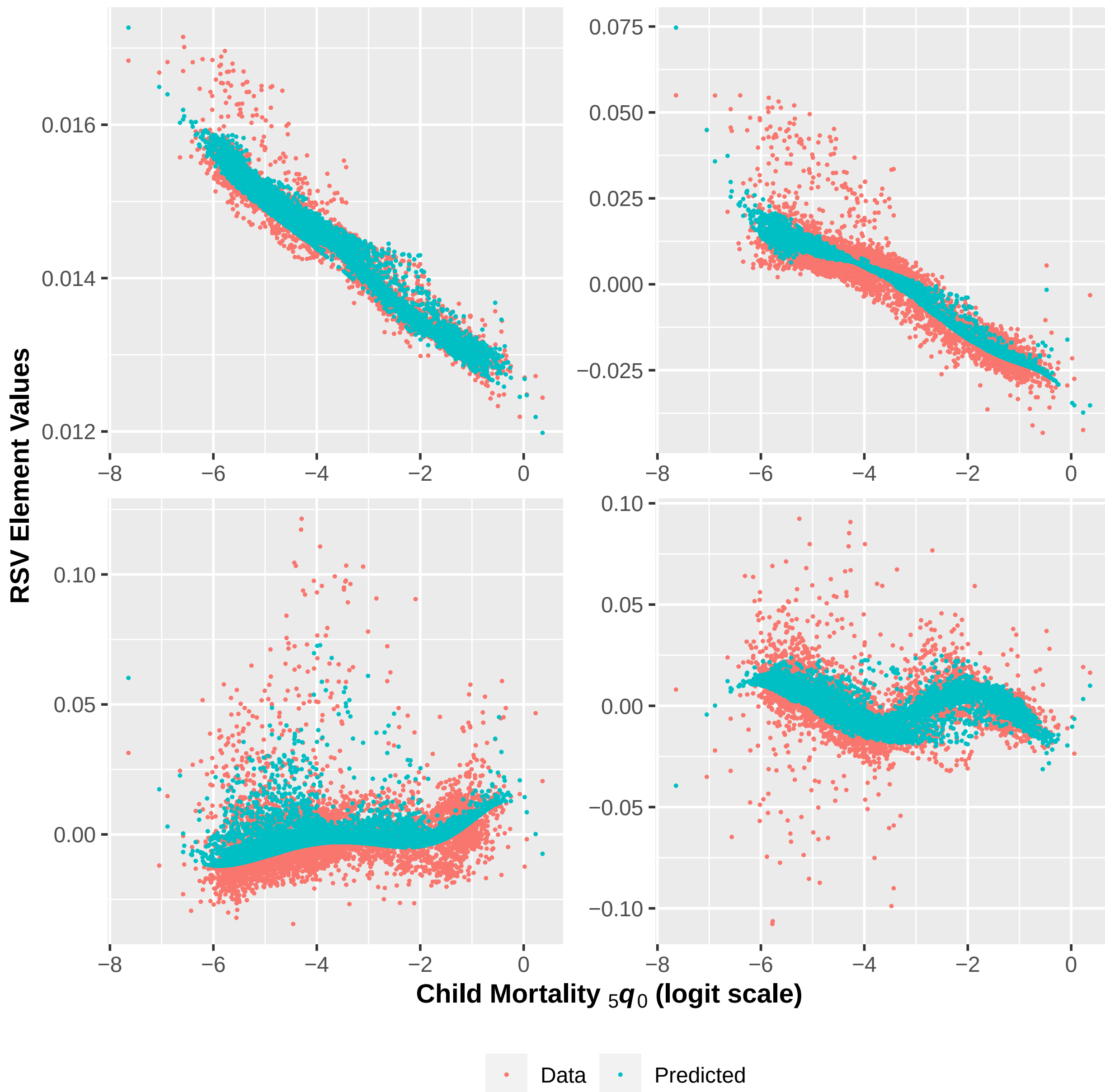
Plot right singular vectors versus child mortality, ${}_5q_0$.

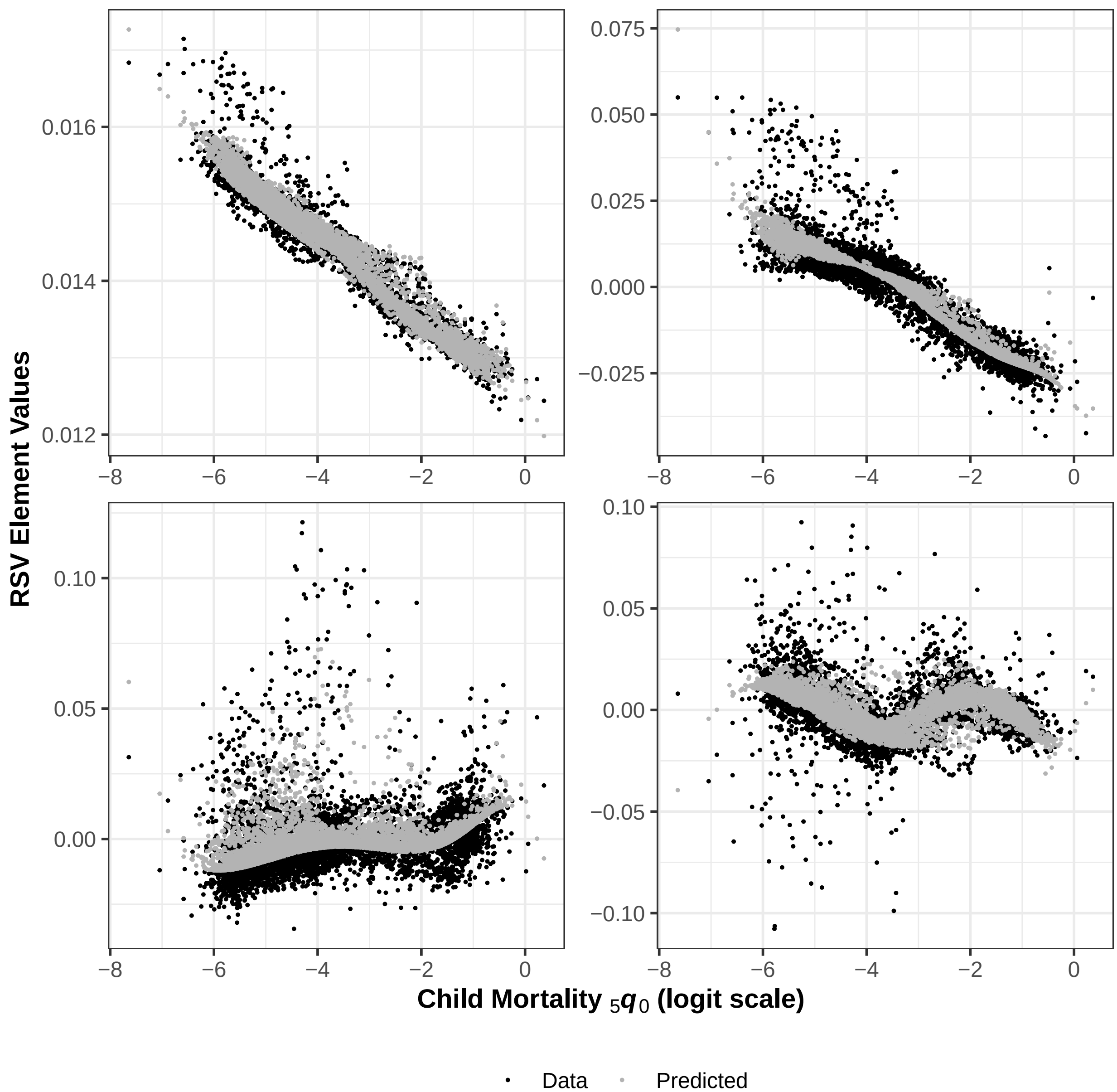
```
# svds
svd.m <- mod.1_0.m$svd$s1
svd.f <- mod.1_0.f$svd$s1

# 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,]),
```

```r
    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,])
)

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])
)

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"))
```

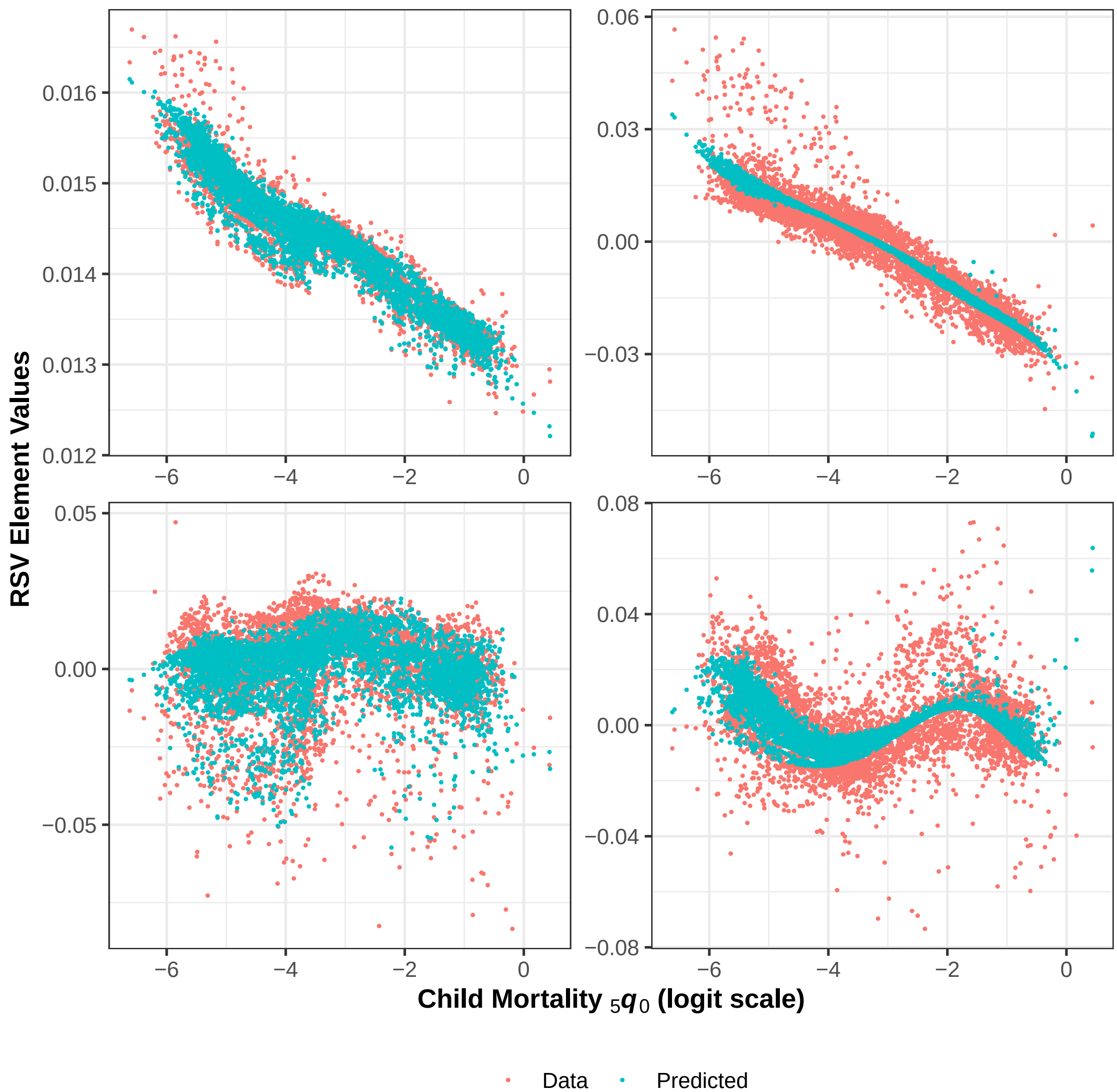
```
         , x = expression(bold('Child Mortality ')
                          [bold(5)]*bolditalic('q')[bold(0)]*bold(' (logit scale)'))) +
    # theme(legend.justification=c(1,0), legend.position=c(0.99,.88)) +
  theme(legend.position="bottom", legend.box = "horizontal") +
    theme(legend.title=element_blank()) +
    facet_wrap(~V, scale="free", 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", 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", 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", 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"
```

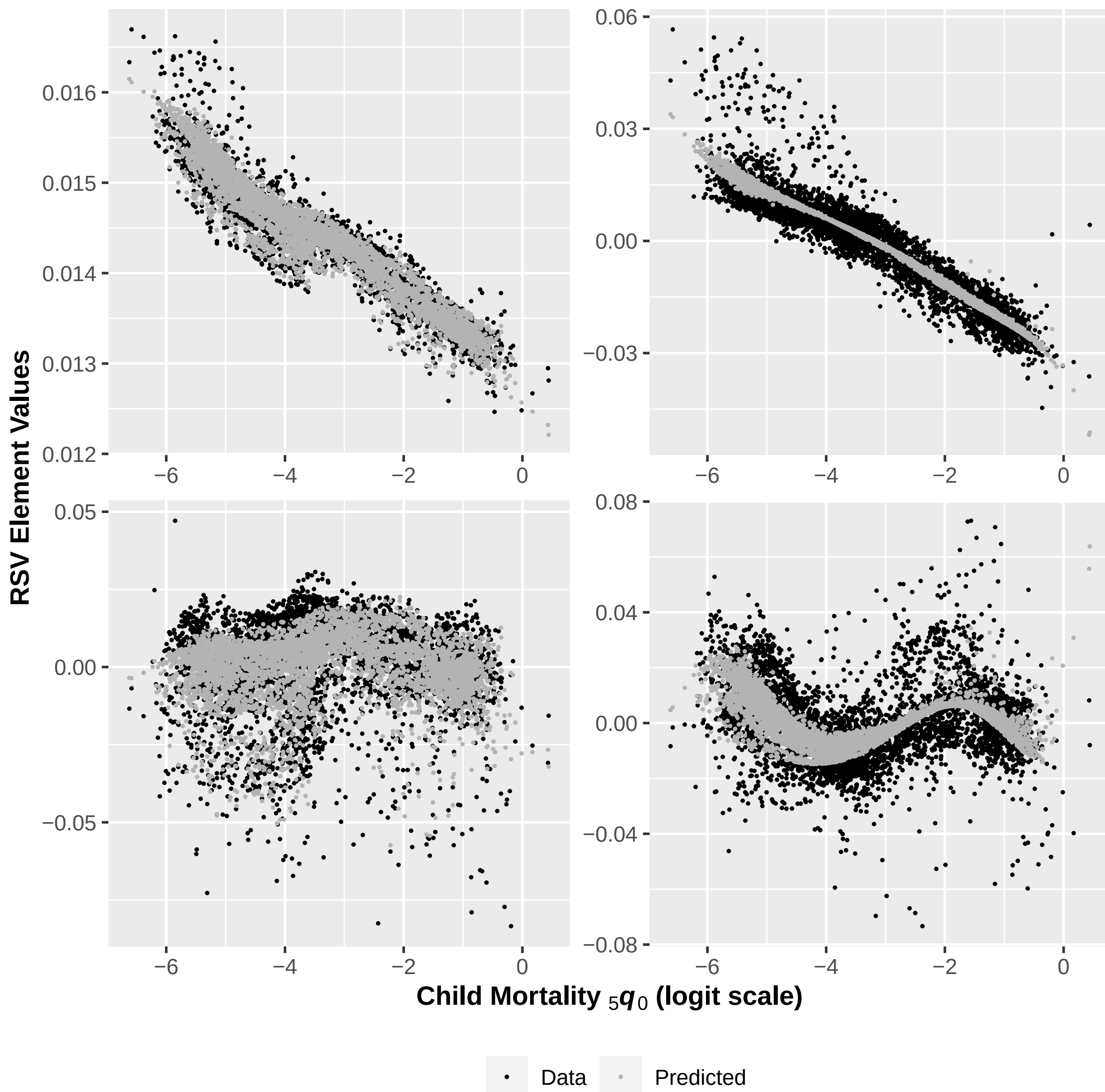
```
        ,"vs.cm.p.df","vs.cm.p","vs.cm.df","vs.cm","svd.m","svd.f"))
```





RSV Element Values
Child Mortality $_5q_0$ (logit scale)
0.016
0.014
0.012
0.075
0.050
0.025
0.000
−0.025
0.10
0.05
0.00
−0.05
−0.10
−8
−6
−4
−2
0
Data
Predicted


RSV Element Values
Child Mortality $_5q_0$ (logit scale)
Data
Predicted
0.016
0.015
0.014
0.013
0.012
0.06
0.03
0.00
−0.03
0.05
0.00
−0.05
0.08
0.04
0.00
−0.04
−0.08
−6
−4
−2
0

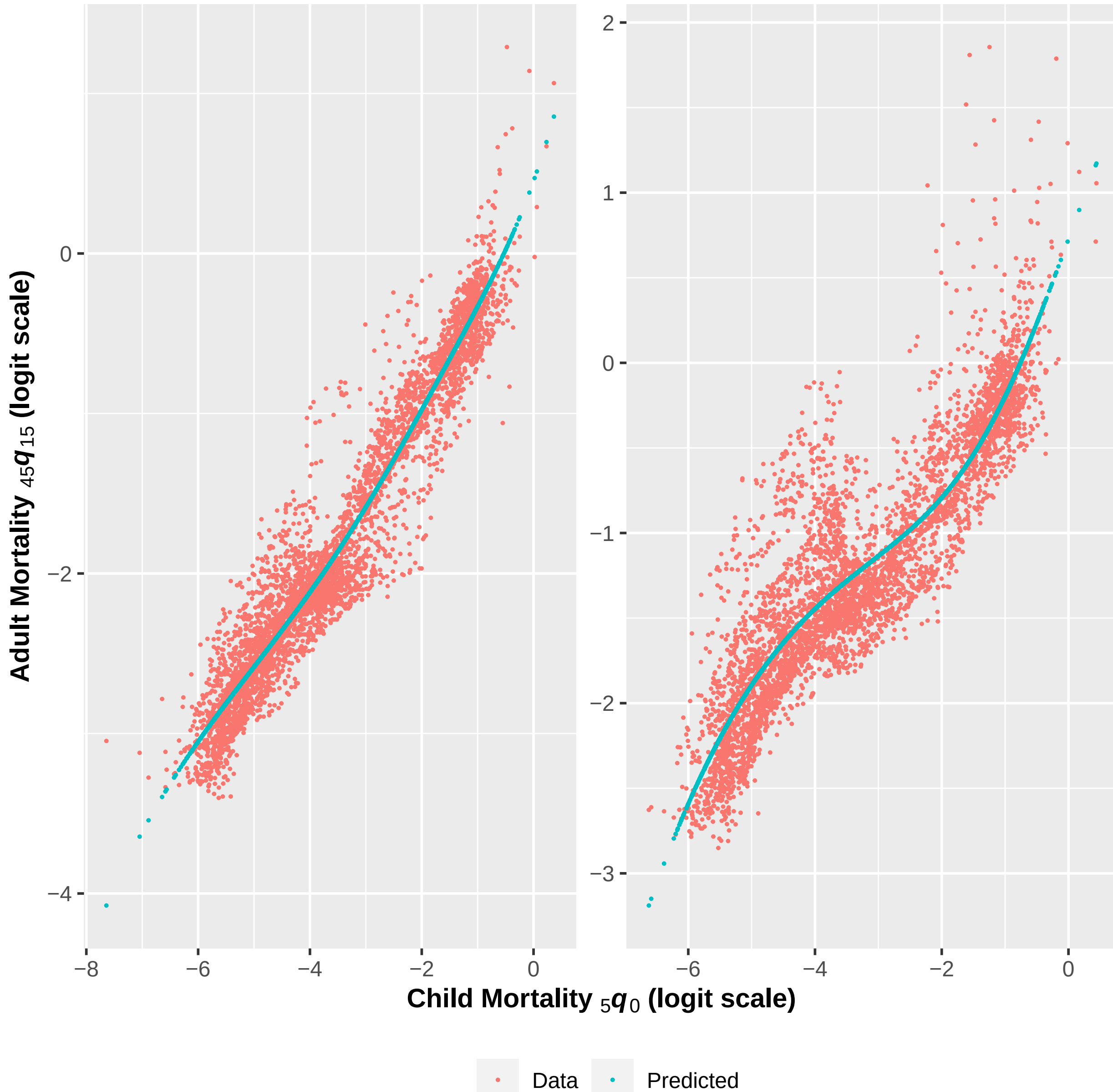


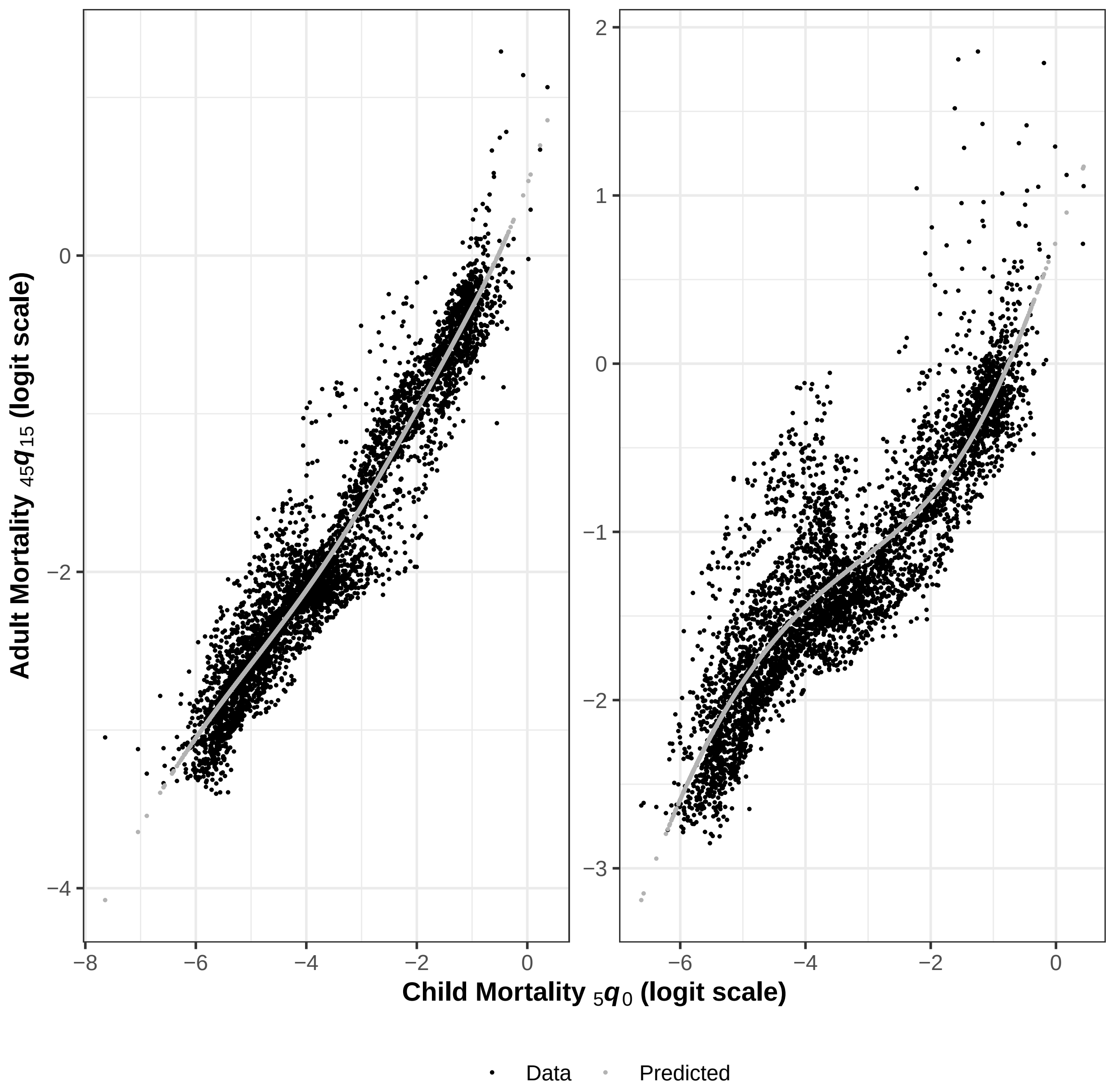
Plot adult mortality, $_45q_{15}$, by child mortality, $_5q_0$.

```
# 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)
```

```r
# 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", 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", 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"))
```



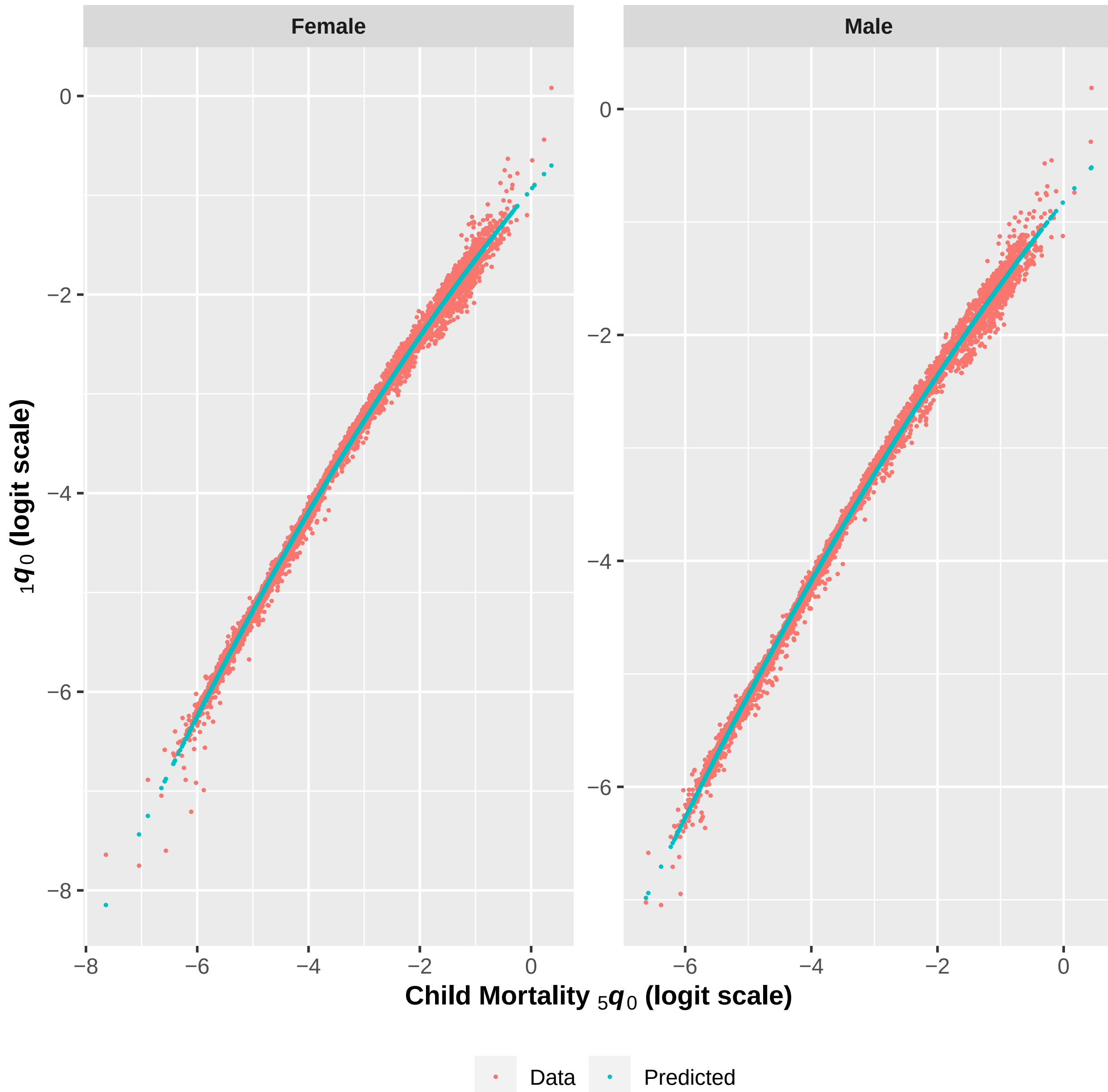


Plot probability of dying in the first year of life, ${}_1q_0$, versus child mortality, ${}_5q_0$.

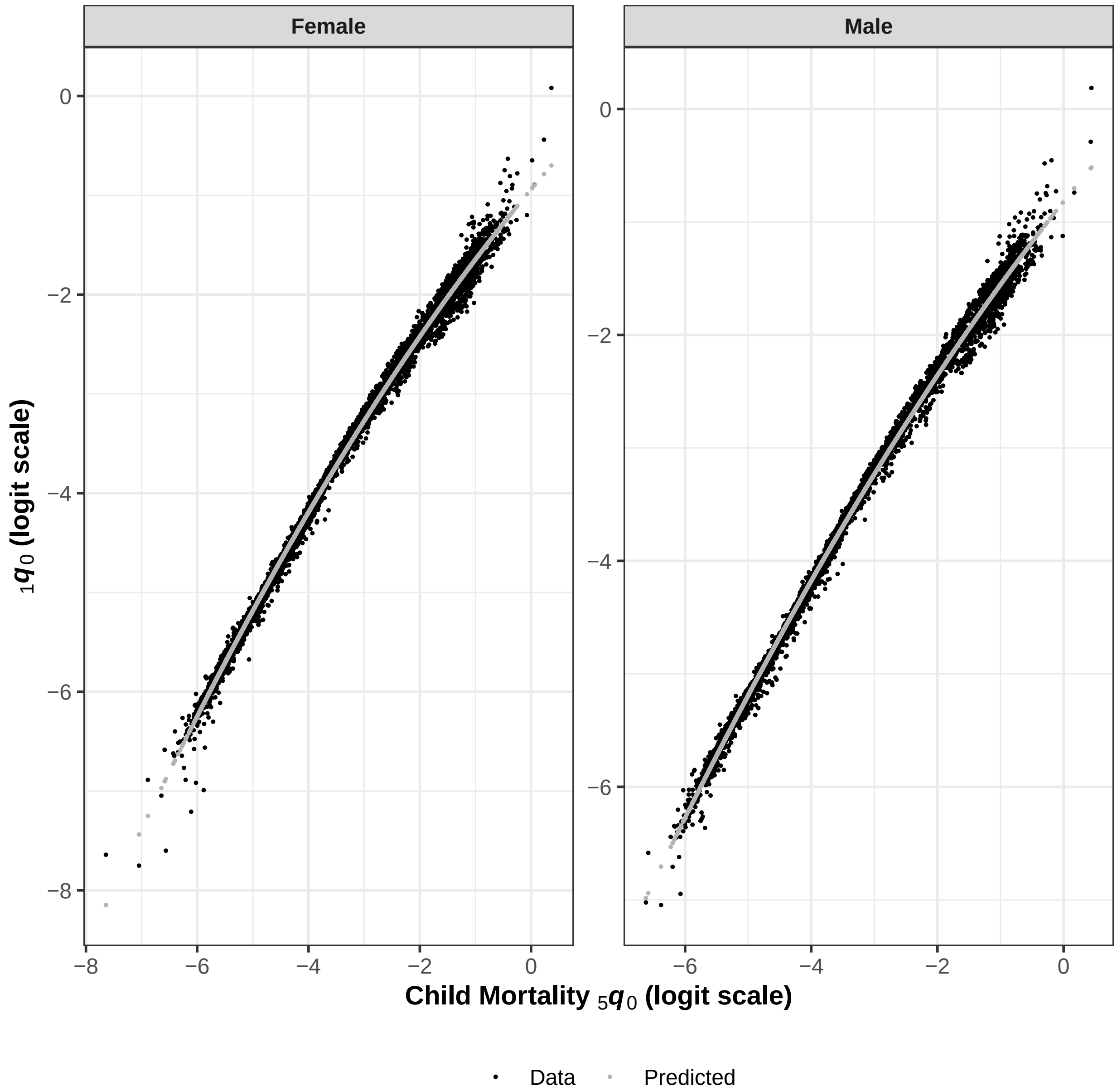
```
# data
q0.cm.m <- data.frame(
    Sex = as.character("Male"),
    CM = as.numeric(Qlogit.m[1,]),
    q0 = as.numeric(q1logit.m[1,])
)
# str(q0.cm.m)

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

```r
# 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"))
```



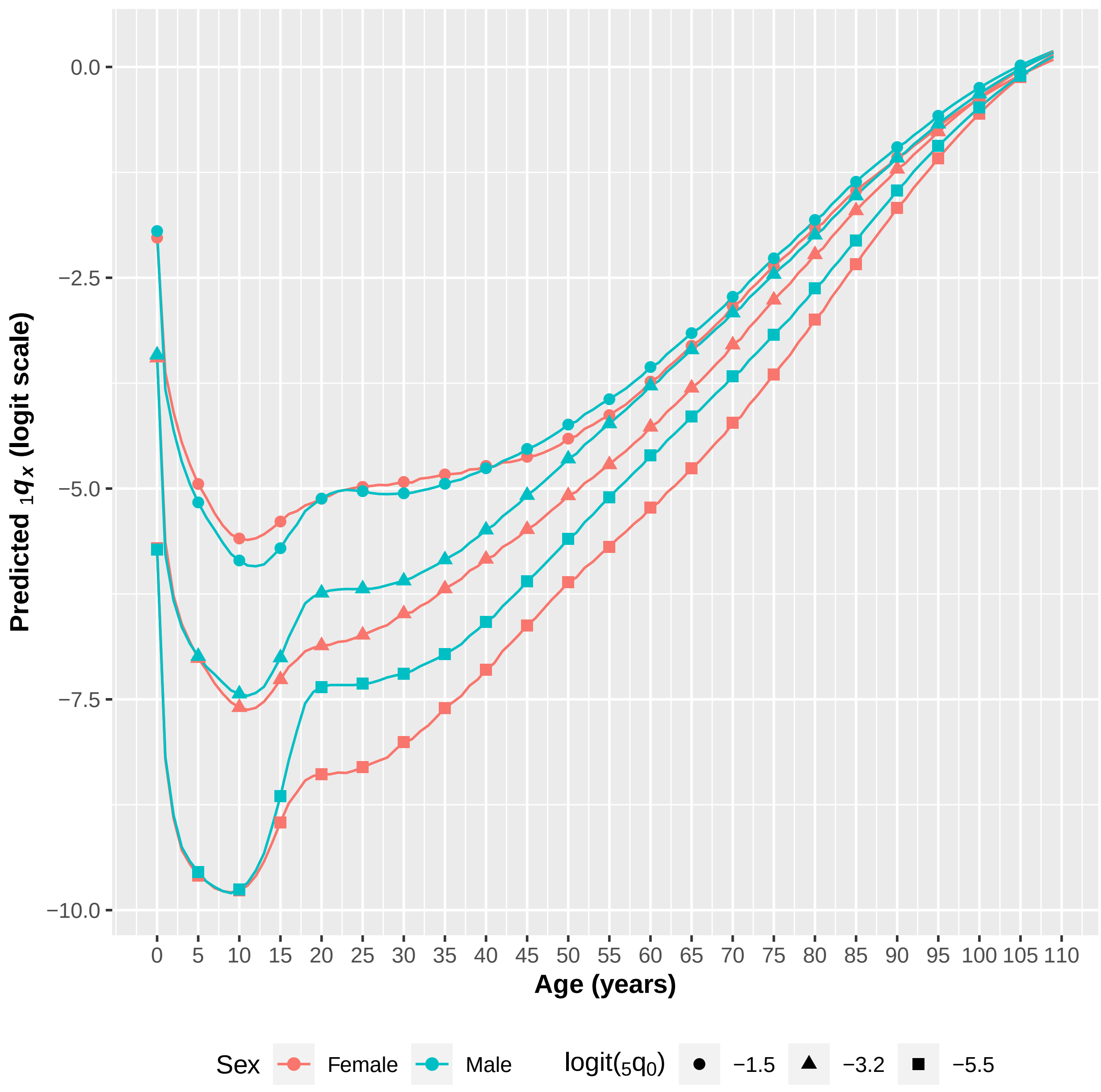
Female
Male
$_1q_0$ (logit scale)
Child Mortality $_5q_0$ (logit scale)
0
−2
−4
−6
−8
Data
Predicted



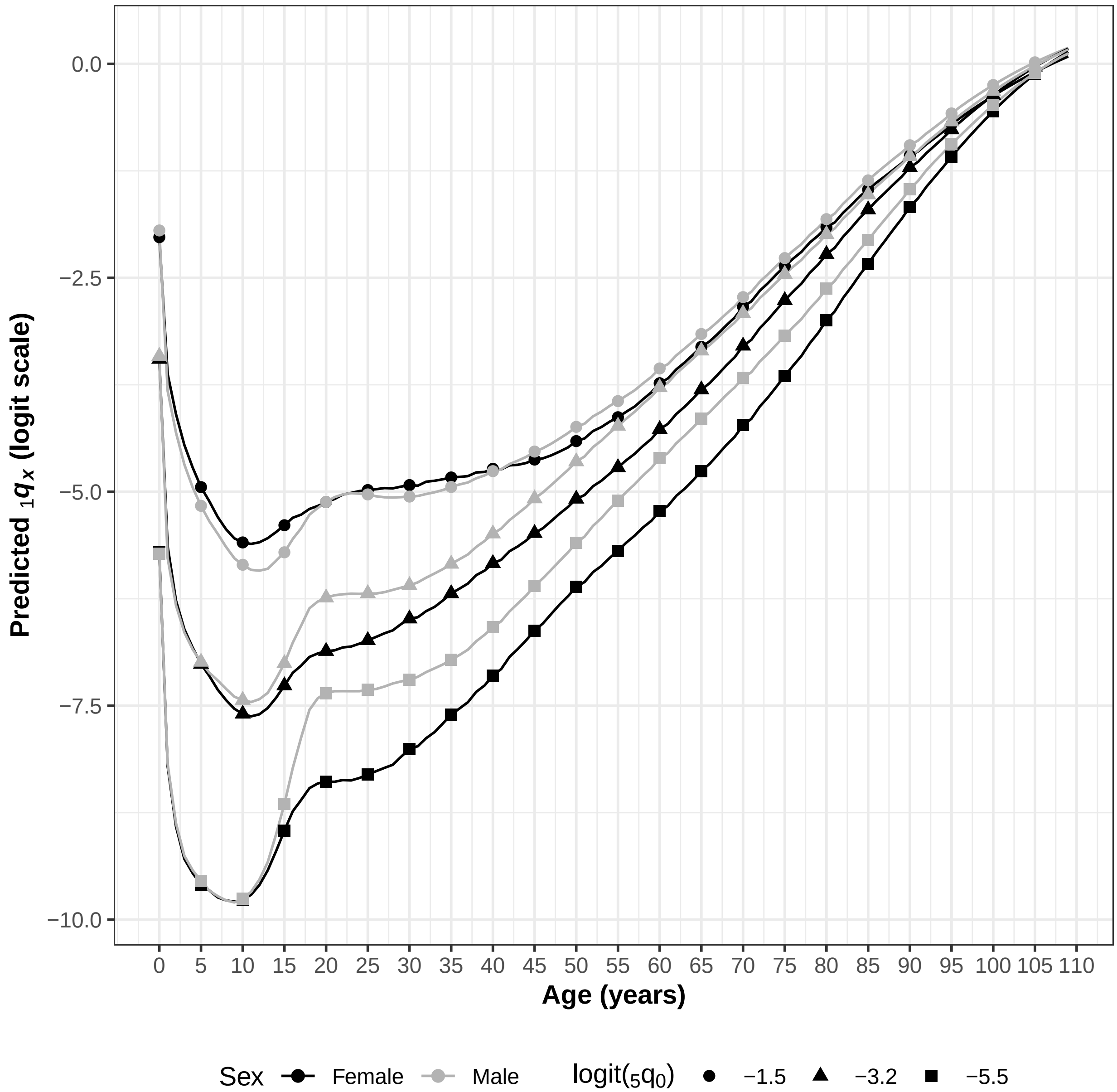
Plot heuristic predictions or life tables at three levels of child mortality from very low to very high.

```
# 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]),
```

```r
    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"))
```



Predicted $_1q_x$ (logit scale)
0.0
−2.5
−5.0
−7.5
−10.0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
Age (years)
Sex
Female
Male
logit($_5q_0$)
−1.5
−3.2
−5.5



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
```

```r
# 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/fltper_5x1/*")
# 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
```

```r
# # 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 -- 2019 & 99  \\
## AUT & Austria & 1947 -- 2019 & 73  \\
## BEL & Belgium & 1841 -- 1913 & 73  \\
## BEL & Belgium & 1919 -- 2020 & 102  \\
## BGR & Bulgaria & 1947 -- 2021 & 75  \\
## BLR & Belarus & 1959 -- 2018 & 60  \\
## CAN & Canada & 1921 -- 2019 & 99  \\
## CHE & Switzerland & 1876 -- 2021 & 146  \\
## CHL & Chile & 1992 -- 2020 & 29  \\
## CZE & Czechia & 1950 -- 2019 & 70  \\
## DEUTE & East Germany & 1956 -- 2020 & 65  \\
## DEUTNP & Germany & 1990 -- 2020 & 31  \\
## DEUTW & West Germany & 1956 -- 2020 & 65  \\
## DNK & Denmark & 1835 -- 2021 & 187  \\
## ESP & Spain & 1908 -- 2020 & 113  \\
## EST & Estonia & 1959 -- 2019 & 61  \\
## FIN & Finland & 1878 -- 2021 & 144  \\
```

```
## FRACNP & France -- Civilian Population & 1816 -- 2020 & 205  \\
## FRATNP & France -- Total Population & 1816 -- 2020 & 205  \\
## GBRCENW & England and Wales -- Civilian National Population & 1841 -- 2020 & 180  \\
## GBRTENW & England and Wales -- Total Population & 1841 -- 2020 & 180  \\
## GBR_NIR & Northern Ireland & 1922 -- 2020 & 99  \\
## GBR_NP & United Kingdom & 1922 -- 2020 & 99  \\
## GBR_SCO & Scotland & 1855 -- 2020 & 166  \\
## GRC & Greece & 1981 -- 2019 & 39  \\
## HKG & Hong Kong & 1986 -- 2019 & 34  \\
## HRV & Croatia & 2001 -- 2020 & 20  \\
## HUN & Hungary & 1950 -- 2020 & 71  \\
## IRL & Ireland & 1950 -- 2020 & 71  \\
## ISL & Iceland & 1838 -- 1851 & 14  \\
## ISL & Iceland & 1853 -- 2020 & 168  \\
## ISR & Israel & 1983 -- 2016 & 34  \\
## ITA & Italy  & 1872 -- 2019 & 148  \\
## JPN & Japan & 1947 -- 2020 & 74  \\
## KOR & Republic of Korea & 2003 -- 2020 & 18  \\
## LTU & Lithuania & 1959 -- 2020 & 62  \\
## LUX & Luxembourg & 1960 -- 2020 & 61  \\
## LVA & Latvia & 1959 -- 2019 & 61  \\
## NLD & Netherlands & 1850 -- 2019 & 170  \\
## NOR & Norway & 1846 -- 2020 & 175  \\
## 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 -- 2021 & 82  \\
## RUS & Russia & 1959 -- 2014 & 56  \\
## SVK & Slovakia  & 1950 -- 2019 & 70  \\
## SVN & Slovenia & 1983 -- 2019 & 37  \\
## SWE & Sweden & 1751 -- 2021 & 271  \\
## TWN & Taiwan & 1970 -- 2019 & 50  \\
## UKR & Ukraine & 1959 -- 2013 & 55  \\
## USA & The United States of America & 1933 -- 2020 & 88  \\
```

```
# 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
```

```r
      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] ""
```

```r
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): 4857"
```

```r
print(paste("Check, totalling up country counts of life tables:",ltGrandTot))
```

```
## [1] "Check, totalling up country counts of life tables: 4857"
```

```r
print("")
```

```
## [1] ""
```

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

```
## [1] "Four 'by-hand' corrections"
```

```r
print("GBRTENW: England and Wales - Total Population")
```

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

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

```
## [1] "GBRCENW: England and Wales - Civilian National Population"
```

```r
print("FRACNP: France - Civilian Population")
```

```
## [1] "FRACNP: France - Civilian Population"
```

```r
print("FRATNP: France - Total Population")
```

```
## [1] "FRATNP: France - Total Population"
```

```r
rm(list=c("country","year","files","i","j","index"))
```

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

```r
# 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)
```

```r
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.

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

```
##      total.abs.error mean.abs.error max.error
## comp        1521.297     0.01361815  0.330284
## lq          1587.174     0.01420786  0.396844
```

```r
comps.child$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1757.213     0.01572999 0.3907967
## lq          1884.154     0.01686633 0.3792040
```

```r
cat("\n")
```

```r
comps.adult$female
```

```
##      total.abs.error mean.abs.error max.error
## comp        1356.626     0.01214407 0.2188245
## lq          1479.815     0.01324681 0.3865020
```

```r
comps.adult$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1439.749     0.01288816 0.3960295
```

```
## lq          1556.590     0.01393408 0.3532550
```

```r
# 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"),
  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: $\\logit(\\qoz)_{z \\ell} = f(\\qf_{\\, z \\ell})$",
    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: $v_{\\ell i} = f_{i}(\\qf_{\\, \\ell},\\qff_{\\, \\ell})$",
```

```
    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.m$mods$s1$v1,
    mod.1_0.m$mods$s1$v2,
    mod.1_0.m$mods$s1$v3,
    mod.1_0.m$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 predicions, 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)
```

```r
  # 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$l5.f)/sum(rowSums(l$l5.f))
# sum(weight.f)
# male
weights.m <- rowSums(l$l5.m)/sum(rowSums(l$l5.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(
```

```r
  l5.f = l5.f
 ,l5.m = l5.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         1.276397041  1.299075559 -0.0226785177
## 1-4       1.408414640  1.290327243  0.1180873966
## 5-9       0.767699838  0.736912327  0.0307875109
## 10-14     0.502958388  0.487051658  0.0159067306
## 15-19     0.625435289  0.704798886 -0.0793635978
## 20-24     0.768475956  0.856923997 -0.0884480406
## 25-29     0.763100146  0.847788329 -0.0846881829
## 30-34     0.755313144  0.803990005 -0.0486768613
## 35-39     0.836285415  0.842893415 -0.0066080001
## 40-44     0.957739804  0.934247321  0.0234924839
## 45-49     1.122183897  1.115201309  0.0069825875
## 50-54     1.474448083  1.483264194 -0.0088161107
## 55-59     1.938532883  1.968277258 -0.0297443748
## 60-64     2.500755628  2.600315934 -0.0995603067
## 65-69     3.090331871  3.231895979 -0.1415641083
## 70-74     4.048727135  4.228522119 -0.1797949838
## 75-79     4.784486488  4.861158780 -0.0766722919
## 80-84     4.665956185  4.817339861 -0.1513836757
## 85-89     3.421536917  3.433315881 -0.0117789638
## 90-94     1.708400939  1.741393783 -0.0329928438
## 95-99     0.451925274  0.482189042 -0.0302637684
## 100-104   0.057295467  0.065067305 -0.0077718371
## 105-109   0.003584785  0.004230546 -0.0006457608
##          37.929985215 38.836180732 -0.9061955166
##          tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0        1.511894e+00  1.528508492 -0.0166144353
## 1-4      2.006323e+00  1.537837965  0.4684849839
## 5-9      8.704992e-01  0.863474847  0.0070243634
## 10-14    5.157093e-01  0.486676709  0.0290326166
## 15-19    8.907541e-01  0.857298234  0.0334558425
## 20-24    1.688862e+00  1.639896988  0.0489647990
## 25-29    1.567353e+00  1.520258070  0.0470944637
## 30-34    1.507634e+00  1.476153019  0.0314811126
## 35-39    1.688835e+00  1.649389101  0.0394458805
## 40-44    1.983133e+00  1.944609759  0.0385230917
## 45-49    2.408708e+00  2.394912804  0.0137947008
```

```
## 50-54   2.987014e+00  3.043303050 -0.0562894572
## 55-59   3.579720e+00  3.765964229 -0.1862438722
## 60-64   4.312860e+00  4.696451870 -0.3835916987
## 65-69   4.806364e+00  5.280026515 -0.4736620825
## 70-74   5.037189e+00  5.622853908 -0.5856648027
## 75-79   4.683567e+00  5.104966989 -0.4214003454
## 80-84   3.450290e+00  3.741253262 -0.2909632785
## 85-89   1.954104e+00  2.032708522 -0.0786040986
## 90-94   7.889422e-01  0.830952346 -0.0420101742
## 95-99   1.574758e-01  0.177269739 -0.0197938914
## 100-104 1.669422e-02  0.019790770 -0.0030965460
## 105-109 9.858686e-04  0.001187835 -0.0002019664
##         4.841491e+01 50.215745023 -1.8008347944
##
## $tae.e
##         tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.316375e+02 4.435639e+02 -1.192646e+01
## 1-4     4.712269e+02 4.875814e+02 -1.635449e+01
## 5-9     4.359237e+02 4.538331e+02 -1.790943e+01
## 10-14   4.148063e+02 4.324512e+02 -1.764497e+01
## 15-19   4.011535e+02 4.169110e+02 -1.575757e+01
## 20-24   3.816970e+02 3.946570e+02 -1.295996e+01
## 25-29   3.603127e+02 3.710708e+02 -1.075806e+01
## 30-34   3.413208e+02 3.512874e+02 -9.966535e+00
## 35-39   3.251453e+02 3.358103e+02 -1.066504e+01
## 40-44   3.087366e+02 3.206638e+02 -1.192719e+01
## 45-49   2.904298e+02 3.024484e+02 -1.201864e+01
## 50-54   2.685858e+02 2.800086e+02 -1.142274e+01
## 55-59   2.430104e+02 2.531206e+02 -1.011028e+01
## 60-64   2.135252e+02 2.225416e+02 -9.016454e+00
## 65-69   1.811755e+02 1.875033e+02 -6.327839e+00
## 70-74   1.450620e+02 1.495430e+02 -4.480983e+00
## 75-79   1.053975e+02 1.084342e+02 -3.036668e+00
## 80-84   6.726290e+01 6.910133e+01 -1.838436e+00
## 85-89   3.553078e+01 3.588474e+01 -3.539649e-01
## 90-94   1.449795e+01 1.451744e+01 -1.949645e-02
## 95-99   3.859405e+00 3.847913e+00  1.149282e-02
## 100-104 5.657454e-01 5.640760e-01  1.669427e-03
## 105-109 4.351718e-02 4.323725e-02  2.799255e-04
## 110+    6.847563e-03 2.873634e-03  3.973928e-03
##         5.440913e+03 5.635391e+03 -1.944778e+02
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.471387e+02 6.797748e+02 -3.263616e+01
## 1-4     6.855123e+02 7.085808e+02 -2.306845e+01
## 5-9     6.369454e+02 6.811779e+02 -4.423243e+01
## 10-14   6.196616e+02 6.622809e+02 -4.261928e+01
## 15-19   6.094582e+02 6.516896e+02 -4.223147e+01
## 20-24   5.872440e+02 6.282499e+02 -4.100591e+01
## 25-29   5.478639e+02 5.894658e+02 -4.160190e+01
## 30-34   5.124671e+02 5.561032e+02 -4.363613e+01
## 35-39   4.781096e+02 5.230636e+02 -4.495395e+01
## 40-44   4.414516e+02 4.861256e+02 -4.467402e+01
## 45-49   4.001936e+02 4.421840e+02 -4.199045e+01
## 50-54   3.525941e+02 3.908551e+02 -3.826096e+01
```

```
## 55-59   2.988042e+02 3.319281e+02 -3.312388e+01
## 60-64   2.407061e+02 2.680165e+02 -2.731035e+01
## 65-69   1.807868e+02 2.008049e+02 -2.001819e+01
## 70-74   1.248854e+02 1.376655e+02 -1.278012e+01
## 75-79   7.662827e+01 8.274340e+01 -6.115137e+00
## 80-84   4.091614e+01 4.282835e+01 -1.912204e+00
## 85-89   1.834887e+01 1.853039e+01 -1.815181e-01
## 90-94   6.526819e+00 6.469250e+00  5.756872e-02
## 95-99   1.406234e+00 1.399643e+00  6.590657e-03
## 100-104 1.742800e-01 1.752970e-01 -1.016996e-03
## 105-109 1.254183e-02 1.249790e-02  4.393426e-05
## 110+    3.083579e-03 1.065312e-03  2.018268e-03
##         7.507839e+03 8.090126e+03 -5.822873e+02
##
## $tot.tae.e0
##             [,1]    [,2]      [,3]
## Female 6515.920 6695.96 -180.0396
## Male   9092.473 9551.02 -458.5469
```

```
# 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           1.2763970  1.299075559  -0.02267852
## 1-4         6.5655548  1.290327243   5.27522754
## 5-9         1.9712095  0.736912327   1.23429717
## 10-14       1.0574502  0.487051658   0.57039850
## 15-19       1.1119954  0.704798886   0.40719646
## 20-24       1.3761875  0.856923997   0.51926355
```

```
## 25-29      1.4218237  0.847788329   0.57403539
## 30-34      1.2711688  0.803990005   0.46717879
## 35-39      1.0033507  0.842893415   0.16045724
## 40-44      0.9860545  0.934247321   0.05180722
## 45-49      1.9602618  1.115201309   0.84506050
## 50-54      3.2137956  1.483264194   1.73053137
## 55-59      4.7471309  1.968277258   2.77885360
## 60-64      6.1785240  2.600315934   3.57820802
## 65-69      8.2155184  3.231895979   4.98362242
## 70-74     10.1262286  4.228522119   5.89770653
## 75-79     11.8440615  4.861158780   6.98290267
## 80-84     11.6584911  4.817339861   6.84115127
## 85-89      8.5119605  3.433315881   5.07864458
## 90-94      4.2609760  1.741393783   2.51958221
## 95-99      1.1866399  0.482189042   0.70445085
## 100-104    0.1633992  0.065067305   0.09833186
## 105-109    0.0108397  0.004230546   0.00660915
##           90.1190191 38.836180732  51.28283840
##          tae.comp.q.m   tae.lq.q.m tae.diff.q.m
## 0         1.511894057  1.528508492 -0.016614435
## 1-4       7.676854777  1.537837965  6.139016812
## 5-9       2.245951139  0.863474847  1.382476292
## 10-14     1.050603058  0.486676709  0.563926349
## 15-19     1.018552387  0.857298234  0.161254154
## 20-24     1.867845046  1.639896988  0.227948058
## 25-29     1.735519184  1.520258070  0.215261113
## 30-34     1.633987344  1.476153019  0.157834325
## 35-39     1.705271193  1.649389101  0.055882092
## 40-44     2.018405138  1.944609759  0.073795379
## 45-49     2.893961587  2.394912804  0.499048784
## 50-54     4.456199719  3.043303050  1.412896669
## 55-59     6.673701147  3.765964229  2.907736918
## 60-64     8.808687832  4.696451870  4.112235962
## 65-69    10.634923020  5.280026515  5.354896506
## 70-74    10.968741667  5.622853908  5.345887758
## 75-79    10.009131311  5.104966989  4.904164322
## 80-84     7.418279797  3.741253262  3.677026535
## 85-89     4.222187569  2.032708522  2.189479047
## 90-94     1.710207397  0.830952346  0.879255051
## 95-99     0.357483275  0.177269739  0.180213536
## 100-104   0.039302325  0.019790770  0.019511555
## 105-109   0.002363182  0.001187835  0.001175347
##          90.660053153 50.215745023 40.444308129
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f tae.diff.e.f
## 0        7.815686e+02 4.435639e+02 3.380047e+02
## 1-4      8.263645e+02 4.875814e+02 3.387831e+02
## 5-9      5.724389e+02 4.538331e+02 1.186058e+02
## 10-14    5.389082e+02 4.324512e+02 1.064570e+02
## 15-19    5.399802e+02 4.169110e+02 1.230691e+02
## 20-24    5.501101e+02 3.946570e+02 1.554531e+02
## 25-29    5.765874e+02 3.710708e+02 2.055166e+02
## 30-34    6.102629e+02 3.512874e+02 2.589755e+02
```

```
## 35-39   6.391148e+02 3.358103e+02 3.033045e+02
## 40-44   6.539417e+02 3.206638e+02 3.332780e+02
## 45-49   6.469975e+02 3.024484e+02 3.445490e+02
## 50-54   6.114286e+02 2.800086e+02 3.314200e+02
## 55-59   5.578564e+02 2.531206e+02 3.047358e+02
## 60-64   4.888000e+02 2.225416e+02 2.662583e+02
## 65-69   4.150160e+02 1.875033e+02 2.275126e+02
## 70-74   3.339551e+02 1.495430e+02 1.844121e+02
## 75-79   2.520183e+02 1.084342e+02 1.435841e+02
## 80-84   1.698738e+02 6.910133e+01 1.007724e+02
## 85-89   9.432465e+01 3.588474e+01 5.843990e+01
## 90-94   3.942179e+01 1.451744e+01 2.490435e+01
## 95-99   1.059582e+01 3.847913e+00 6.747912e+00
## 100-104 1.600958e+00 5.640760e-01 1.036882e+00
## 105-109 1.255631e-01 4.323725e-02 8.232585e-02
## 110+    6.847563e-03 2.873634e-03 3.973928e-03
##         9.911298e+03 5.635391e+03 4.275907e+03
##         tae.comp.e.m   tae.lq.e.m tae.diff.e.m
## 0       9.374851e+02 6.797748e+02 2.577103e+02
## 1-4     9.854030e+02 7.085808e+02 2.768222e+02
## 5-9     7.327646e+02 6.811779e+02 5.158677e+01
## 10-14   7.370021e+02 6.622809e+02 7.472123e+01
## 15-19   7.518149e+02 6.516896e+02 1.001253e+02
## 20-24   7.492382e+02 6.282499e+02 1.209883e+02
## 25-29   7.431149e+02 5.894658e+02 1.536490e+02
## 30-34   7.396981e+02 5.561032e+02 1.835949e+02
## 35-39   7.352659e+02 5.230636e+02 2.122023e+02
## 40-44   7.250725e+02 4.861256e+02 2.389469e+02
## 45-49   7.009293e+02 4.421840e+02 2.587452e+02
## 50-54   6.528874e+02 3.908551e+02 2.620323e+02
## 55-59   5.802800e+02 3.319281e+02 2.483520e+02
## 60-64   4.826484e+02 2.680165e+02 2.146319e+02
## 65-69   3.734726e+02 2.008049e+02 1.726676e+02
## 70-74   2.626272e+02 1.376655e+02 1.249617e+02
## 75-79   1.670380e+02 8.274340e+01 8.429456e+01
## 80-84   9.309148e+01 4.282835e+01 5.026313e+01
## 85-89   4.281594e+01 1.853039e+01 2.428556e+01
## 90-94   1.503269e+01 6.469250e+00 8.563437e+00
## 95-99   3.213132e+00 1.399643e+00 1.813490e+00
## 100-104 3.970862e-01 1.752970e-01 2.217892e-01
## 105-109 2.830138e-02 1.249790e-02 1.580348e-02
## 110+    3.083579e-03 1.065312e-03 2.018268e-03
##         1.121132e+04 8.090126e+03 3.121198e+03
##
## $tot.tae.e0
##            [,1]    [,2]     [,3]
## Female 11798.42 6695.96 5102.457
## Male   13171.92 9551.02 3620.899
```

```
# 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
```

```r
             ,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"
```

```r
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"
```

```r
# 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")
```

```r
age.comps.2
```

```
## $tae.q
##          tae.comp.q.f   tae.lq.q.f  tae.diff.q.f
## 0         1.276397041  1.299075559 -0.0226785177
## 1-4       1.461898382  1.290327243  0.1715711391
## 5-9       0.793781128  0.736912327  0.0568688008
## 10-14     0.528131666  0.487051658  0.0410800081
## 15-19     0.653884062  0.704798886 -0.0509148240
## 20-24     0.802745157  0.856923997 -0.0541788403
## 25-29     0.787198500  0.847788329 -0.0605898297
## 30-34     0.769116684  0.803990005 -0.0348733216
## 35-39     0.857035983  0.842893415  0.0141425685
## 40-44     0.987742451  0.934247321  0.0534951301
## 45-49     1.150698047  1.115201309  0.0354967376
## 50-54     1.511329506  1.483264194  0.0280653121
## 55-59     1.975077039  1.968277258  0.0067997814
## 60-64     2.619154070  2.600315934  0.0188381355
## 65-69     3.255378863  3.231895979  0.0234828843
## 70-74     4.370234871  4.228522119  0.1417127516
## 75-79     5.218852039  4.861158780  0.3576932593
## 80-84     4.983594870  4.817339861  0.1662550088
```

```
## 85-89    3.501424501  3.433315881  0.0681086202
## 90-94    1.699064123  1.741393783 -0.0423296601
## 95-99    0.454266993  0.482189042 -0.0279220494
## 100-104  0.059606293  0.065067305 -0.0054610114
## 105-109  0.003835616  0.004230546 -0.0003949303
##         39.720447885 38.836180732  0.8842671530
##          tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0        1.511894057  1.528508492 -0.0166144353
## 1-4      1.980148013  1.537837965  0.4423100474
## 5-9      0.925969122  0.863474847  0.0624942749
## 10-14    0.546134660  0.486676709  0.0594579501
## 15-19    0.877824876  0.857298234  0.0205266425
## 20-24    1.663425842  1.639896988  0.0235288541
## 25-29    1.559813803  1.520258070  0.0395557325
## 30-34    1.522212113  1.476153019  0.0460590938
## 35-39    1.692561842  1.649389101  0.0431727415
## 40-44    2.007021568  1.944609759  0.0624118094
## 45-49    2.414783685  2.394912804  0.0198708819
## 50-54    2.978528411  3.043303050 -0.0647746389
## 55-59    3.649882170  3.765964229 -0.1160820588
## 60-64    4.521637593  4.696451870 -0.1748142769
## 65-69    5.132340645  5.280026515 -0.1476858694
## 70-74    5.404170996  5.622853908 -0.2186829124
## 75-79    5.012603645  5.104966989 -0.0923633444
## 80-84    3.578192519  3.741253262 -0.1630607426
## 85-89    1.995947366  2.032708522 -0.0367611558
## 90-94    0.809317939  0.830952346 -0.0216344072
## 95-99    0.166303427  0.177269739 -0.0109663118
## 100-104  0.018068625  0.019790770 -0.0017221454
## 105-109  0.001069694  0.001187835 -0.0001181405
##         49.969852612 50.215745023 -0.2458924112
##
## $tae.e
##         tae.comp.e.f   tae.lq.e.f tae.diff.e.f
## 0       4.453513e+02 4.435639e+02  1.787383770
## 1-4     4.829266e+02 4.875814e+02 -4.654788657
## 5-9     4.467077e+02 4.538331e+02 -7.125364200
## 10-14   4.260670e+02 4.324512e+02 -6.384249044
## 15-19   4.128040e+02 4.169110e+02 -4.107069955
## 20-24   3.940927e+02 3.946570e+02 -0.564264939
## 25-29   3.743738e+02 3.710708e+02  3.302995858
## 30-34   3.575737e+02 3.512874e+02  6.286368776
## 35-39   3.425977e+02 3.358103e+02  6.787402845
## 40-44   3.267807e+02 3.206638e+02  6.116894230
## 45-49   3.081567e+02 3.024484e+02  5.708261686
## 50-54   2.860352e+02 2.800086e+02  6.026657584
## 55-59   2.599877e+02 2.531206e+02  6.867011524
## 60-64   2.303181e+02 2.225416e+02  7.776441774
## 65-69   1.956378e+02 1.875033e+02  8.134481417
## 70-74   1.563862e+02 1.495430e+02  6.843229503
## 75-79   1.122975e+02 1.084342e+02  3.863352282
## 80-84   6.974214e+01 6.910133e+01  0.640804512
## 85-89   3.577204e+01 3.588474e+01 -0.112701920
## 90-94   1.442372e+01 1.451744e+01 -0.093724322
```

```
## 95-99   3.875126e+00 3.847913e+00  0.027213378
## 100-104 5.836155e-01 5.640760e-01  0.019539523
## 105-109 4.664290e-02 4.323725e-02  0.003405652
## 110+    6.847563e-03 2.873634e-03  0.003973928
##         5.682544e+03 5.635391e+03 47.153255205
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.588253e+02 6.797748e+02 -2.094952e+01
## 1-4     6.962974e+02 7.085808e+02 -1.228334e+01
## 5-9     6.478938e+02 6.811779e+02 -3.328407e+01
## 10-14   6.289193e+02 6.622809e+02 -3.336163e+01
## 15-19   6.183616e+02 6.516896e+02 -3.332801e+01
## 20-24   5.963131e+02 6.282499e+02 -3.193683e+01
## 25-29   5.584084e+02 5.894658e+02 -3.105736e+01
## 30-34   5.261488e+02 5.561032e+02 -2.995445e+01
## 35-39   4.945403e+02 5.230636e+02 -2.852322e+01
## 40-44   4.595098e+02 4.861256e+02 -2.661580e+01
## 45-49   4.191640e+02 4.421840e+02 -2.302008e+01
## 50-54   3.719995e+02 3.908551e+02 -1.885558e+01
## 55-59   3.177582e+02 3.319281e+02 -1.416987e+01
## 60-64   2.570724e+02 2.680165e+02 -1.094409e+01
## 65-69   1.929331e+02 2.008049e+02 -7.871884e+00
## 70-74   1.323643e+02 1.376655e+02 -5.301179e+00
## 75-79   8.030907e+01 8.274340e+01 -2.434337e+00
## 80-84   4.194186e+01 4.282835e+01 -8.864856e-01
## 85-89   1.869845e+01 1.853039e+01  1.680661e-01
## 90-94   6.702232e+00 6.469250e+00  2.329822e-01
## 95-99   1.476505e+00 1.399643e+00  7.686220e-02
## 100-104 1.870634e-01 1.752970e-01  1.176642e-02
## 105-109 1.355195e-02 1.249790e-02  1.054051e-03
## 110+    3.083579e-03 1.065312e-03  2.018268e-03
##         7.725841e+03 8.090126e+03 -3.642850e+02
##
## $tot.tae.e0
##            [,1]    [,2]       [,3]
## Female 6722.942 6695.96   26.98202
## Male   9256.674 9551.02 -294.34643
```

```
# 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         1.276397041  1.299075559 -0.0226785177
## 1-4       1.378210873  1.290327243  0.0878836295
## 5-9       0.814472399  0.736912327  0.0775600715
## 10-14     0.502307381  0.487051658  0.0152557230
## 15-19     0.648045776  0.704798886 -0.0567531103
## 20-24     0.797524746  0.856923997 -0.0593992506
## 25-29     0.798049667  0.847788329 -0.0497386619
## 30-34     0.772951511  0.803990005 -0.0310384942
## 35-39     0.850672970  0.842893415  0.0077795554
## 40-44     0.963223182  0.934247321  0.0289758616
## 45-49     1.124387098  1.115201309  0.0091857889
## 50-54     1.474237225  1.483264194 -0.0090269687
## 55-59     1.934482976  1.968277258 -0.0337942818
## 60-64     2.522522767  2.600315934 -0.0777931676
## 65-69     3.175383498  3.231895979 -0.0565124814
## 70-74     4.297338871  4.228522119  0.0688167526
## 75-79     5.183592416  4.861158780  0.3224336361
## 80-84     4.973027802  4.817339861  0.1556879413
## 85-89     3.560499384  3.433315881  0.1271835027
## 90-94     1.724943201  1.741393783 -0.0164505820
## 95-99     0.452181912  0.482189042 -0.0300071299
## 100-104   0.057368534  0.065067305 -0.0076987710
## 105-109   0.003605604  0.004230546 -0.0006249422
##          39.285426835 38.836180732  0.4492461033
##          tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0         1.511894057  1.528508492 -0.0166144353
## 1-4       2.005815301  1.537837965  0.4679773357
## 5-9       0.895549846  0.863474847  0.0320749986
## 10-14     0.516005715  0.486676709  0.0293290056
## 15-19     0.885465486  0.857298234  0.0281672522
```

```
## 20-24    1.678759198  1.639896988  0.0388622100
## 25-29    1.573988701  1.520258070  0.0537306303
## 30-34    1.520782559  1.476153019  0.0446295402
## 35-39    1.690964211  1.649389101  0.0415751105
## 40-44    1.968039653  1.944609759  0.0234298940
## 45-49    2.372751500  2.394912804 -0.0221613041
## 50-54    2.956432640  3.043303050 -0.0868704092
## 55-59    3.657104259  3.765964229 -0.1088599705
## 60-64    4.529926956  4.696451870 -0.1665249141
## 65-69    5.115138523  5.280026515 -0.1648879917
## 70-74    5.351126329  5.622853908 -0.2717275798
## 75-79    4.914138189  5.104966989 -0.1908288005
## 80-84    3.512151355  3.741253262 -0.2291019067
## 85-89    1.947518315  2.032708522 -0.0851902072
## 90-94    0.794538689  0.830952346 -0.0364136568
## 95-99    0.165251191  0.177269739 -0.0120185480
## 100-104  0.018125482  0.019790770 -0.0016652876
## 105-109  0.001081406  0.001187835 -0.0001064291
##         49.582549560 50.215745023 -0.6331954633
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0        4.398449e+02 4.435639e+02 -3.719058e+00
## 1-4      4.775343e+02 4.875814e+02 -1.004709e+01
## 5-9      4.438715e+02 4.538331e+02 -9.961607e+00
## 10-14    4.232910e+02 4.324512e+02 -9.160232e+00
## 15-19    4.099467e+02 4.169110e+02 -6.964379e+00
## 20-24    3.906772e+02 3.946570e+02 -3.979774e+00
## 25-29    3.695119e+02 3.710708e+02 -1.558879e+00
## 30-34    3.509635e+02 3.512874e+02 -3.238274e-01
## 35-39    3.353114e+02 3.358103e+02 -4.988813e-01
## 40-44    3.196599e+02 3.206638e+02 -1.003851e+00
## 45-49    3.022288e+02 3.024484e+02 -2.196167e-01
## 50-54    2.812972e+02 2.800086e+02  1.288632e+00
## 55-59    2.563450e+02 2.531206e+02  3.224376e+00
## 60-64    2.276557e+02 2.225416e+02  5.114044e+00
## 65-69    1.946985e+02 1.875033e+02  7.195186e+00
## 70-74    1.564731e+02 1.495430e+02  6.930139e+00
## 75-79    1.131969e+02 1.084342e+02  4.762715e+00
## 80-84    7.093245e+01 6.910133e+01  1.831114e+00
## 85-89    3.663421e+01 3.588474e+01  7.494662e-01
## 90-94    1.461076e+01 1.451744e+01  9.331961e-02
## 95-99    3.861994e+00 3.847913e+00  1.408156e-02
## 100-104  5.661715e-01 5.640760e-01  2.095539e-03
## 105-109  4.375917e-02 4.323725e-02  5.219247e-04
## 110+     6.847563e-03 2.873634e-03  3.973928e-03
##          5.619164e+03 5.635391e+03 -1.622753e+01
##          tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0        6.563619e+02 6.797748e+02 -2.341292e+01
## 1-4      6.936967e+02 7.085808e+02 -1.488409e+01
## 5-9      6.462847e+02 6.811779e+02 -3.489313e+01
## 10-14    6.290579e+02 6.622809e+02 -3.322299e+01
## 15-19    6.187187e+02 6.516896e+02 -3.297098e+01
## 20-24    5.963859e+02 6.282499e+02 -3.186401e+01
```

```
## 25-29   5.586382e+02 5.894658e+02 -3.082760e+01
## 30-34   5.263918e+02 5.561032e+02 -2.971140e+01
## 35-39   4.945639e+02 5.230636e+02 -2.849963e+01
## 40-44   4.593807e+02 4.861256e+02 -2.674488e+01
## 45-49   4.188463e+02 4.421840e+02 -2.333773e+01
## 50-54   3.711141e+02 3.908551e+02 -1.974097e+01
## 55-59   3.161097e+02 3.319281e+02 -1.581837e+01
## 60-64   2.548272e+02 2.680165e+02 -1.318931e+01
## 65-69   1.903630e+02 2.008049e+02 -1.044191e+01
## 70-74   1.301011e+02 1.376655e+02 -7.564386e+00
## 75-79   7.857828e+01 8.274340e+01 -4.165122e+00
## 80-84   4.109206e+01 4.282835e+01 -1.736285e+00
## 85-89   1.828358e+01 1.853039e+01 -2.468099e-01
## 90-94   6.591300e+00 6.469250e+00  1.220497e-01
## 95-99   1.465330e+00 1.399643e+00  6.568758e-02
## 100-104 1.871356e-01 1.752970e-01  1.183868e-02
## 105-109 1.364385e-02 1.249790e-02  1.145953e-03
## 110+    3.083579e-03 1.065312e-03  2.018268e-03
##         7.707056e+03 8.090126e+03 -3.830698e+02
##
## $tot.tae.e0
##             [,1]    [,2]       [,3]
## Female 6639.818 6695.96  -56.14222
## Male   9222.062 9551.02 -328.95790
```

```r
# 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"
```

```r
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"
```

```r
# 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")
```

```r
age.comps.4
```

```
## $tae.q
##          tae.comp.q.f   tae.lq.q.f  tae.diff.q.f
## 0         1.276397041  1.299075559 -0.0226785177
## 1-4       1.408414640  1.290327243  0.1180873966
## 5-9       0.767699838  0.736912327  0.0307875109
## 10-14     0.502958388  0.487051658  0.0159067306
## 15-19     0.625435289  0.704798886 -0.0793635978
## 20-24     0.768475956  0.856923997 -0.0884480406
## 25-29     0.763100146  0.847788329 -0.0846881829
## 30-34     0.755313144  0.803990005 -0.0486768613
## 35-39     0.836285415  0.842893415 -0.0066080001
## 40-44     0.957739804  0.934247321  0.0234924839
## 45-49     1.122183897  1.115201309  0.0069825875
## 50-54     1.474448083  1.483264194 -0.0088161107
## 55-59     1.938532883  1.968277258 -0.0297443748
## 60-64     2.500755628  2.600315934 -0.0995603067
## 65-69     3.090331871  3.231895979 -0.1415641083
## 70-74     4.048727135  4.228522119 -0.1797949838
## 75-79     4.784486488  4.861158780 -0.0766722919
## 80-84     4.665956185  4.817339861 -0.1513836757
## 85-89     3.421536917  3.433315881 -0.0117789638
## 90-94     1.708400939  1.741393783 -0.0329928438
## 95-99     0.451925274  0.482189042 -0.0302637684
## 100-104   0.057295467  0.065067305 -0.0077718371
## 105-109   0.003584785  0.004230546 -0.0006457608
##          37.929985215 38.836180732 -0.9061955166
##          tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0        1.511894e+00  1.528508492 -0.0166144353
## 1-4      2.006323e+00  1.537837965  0.4684849839
## 5-9      8.704992e-01  0.863474847  0.0070243634
## 10-14    5.157093e-01  0.486676709  0.0290326166
## 15-19    8.907541e-01  0.857298234  0.0334558425
## 20-24    1.688862e+00  1.639896988  0.0489647990
## 25-29    1.567353e+00  1.520258070  0.0470944637
## 30-34    1.507634e+00  1.476153019  0.0314811126
## 35-39    1.688835e+00  1.649389101  0.0394458805
## 40-44    1.983133e+00  1.944609759  0.0385230917
## 45-49    2.408708e+00  2.394912804  0.0137947008
## 50-54    2.987014e+00  3.043303050 -0.0562894572
## 55-59    3.579720e+00  3.765964229 -0.1862438722
## 60-64    4.312860e+00  4.696451870 -0.3835916987
## 65-69    4.806364e+00  5.280026515 -0.4736620825
## 70-74    5.037189e+00  5.622853908 -0.5856648027
## 75-79    4.683567e+00  5.104966989 -0.4214003454
```

```
## 80-84   3.450290e+00  3.741253262 -0.2909632785
## 85-89   1.954104e+00  2.032708522 -0.0786040986
## 90-94   7.889422e-01  0.830952346 -0.0420101742
## 95-99   1.574758e-01  0.177269739 -0.0197938914
## 100-104 1.669422e-02  0.019790770 -0.0030965460
## 105-109 9.858686e-04  0.001187835 -0.0002019664
##         4.841491e+01 50.215745023 -1.8008347944
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.316375e+02 4.435639e+02 -1.192646e+01
## 1-4     4.712269e+02 4.875814e+02 -1.635449e+01
## 5-9     4.359237e+02 4.538331e+02 -1.790943e+01
## 10-14   4.148063e+02 4.324512e+02 -1.764497e+01
## 15-19   4.011535e+02 4.169110e+02 -1.575757e+01
## 20-24   3.816970e+02 3.946570e+02 -1.295996e+01
## 25-29   3.603127e+02 3.710708e+02 -1.075806e+01
## 30-34   3.413208e+02 3.512874e+02 -9.966535e+00
## 35-39   3.251453e+02 3.358103e+02 -1.066504e+01
## 40-44   3.087366e+02 3.206638e+02 -1.192719e+01
## 45-49   2.904298e+02 3.024484e+02 -1.201864e+01
## 50-54   2.685858e+02 2.800086e+02 -1.142274e+01
## 55-59   2.430104e+02 2.531206e+02 -1.011028e+01
## 60-64   2.135252e+02 2.225416e+02 -9.016454e+00
## 65-69   1.811755e+02 1.875033e+02 -6.327839e+00
## 70-74   1.450620e+02 1.495430e+02 -4.480983e+00
## 75-79   1.053975e+02 1.084342e+02 -3.036668e+00
## 80-84   6.726290e+01 6.910133e+01 -1.838436e+00
## 85-89   3.553078e+01 3.588474e+01 -3.539649e-01
## 90-94   1.449795e+01 1.451744e+01 -1.949645e-02
## 95-99   3.859405e+00 3.847913e+00  1.149282e-02
## 100-104 5.657454e-01 5.640760e-01  1.669427e-03
## 105-109 4.351718e-02 4.323725e-02  2.799255e-04
## 110+    6.847563e-03 2.873634e-03  3.973928e-03
##         5.440913e+03 5.635391e+03 -1.944778e+02
##          tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.471387e+02 6.797748e+02 -3.263616e+01
## 1-4     6.855123e+02 7.085808e+02 -2.306845e+01
## 5-9     6.369454e+02 6.811779e+02 -4.423243e+01
## 10-14   6.196616e+02 6.622809e+02 -4.261928e+01
## 15-19   6.094582e+02 6.516896e+02 -4.223147e+01
## 20-24   5.872440e+02 6.282499e+02 -4.100591e+01
## 25-29   5.478639e+02 5.894658e+02 -4.160190e+01
## 30-34   5.124671e+02 5.561032e+02 -4.363613e+01
## 35-39   4.781096e+02 5.230636e+02 -4.495395e+01
## 40-44   4.414516e+02 4.861256e+02 -4.467402e+01
## 45-49   4.001936e+02 4.421840e+02 -4.199045e+01
## 50-54   3.525941e+02 3.908551e+02 -3.826096e+01
## 55-59   2.988042e+02 3.319281e+02 -3.312388e+01
## 60-64   2.407061e+02 2.680165e+02 -2.731035e+01
## 65-69   1.807868e+02 2.008049e+02 -2.001819e+01
## 70-74   1.248854e+02 1.376655e+02 -1.278012e+01
## 75-79   7.662827e+01 8.274340e+01 -6.115137e+00
## 80-84   4.091614e+01 4.282835e+01 -1.912204e+00
```

```
## 85-89   1.834887e+01 1.853039e+01 -1.815181e-01
## 90-94   6.526819e+00 6.469250e+00  5.756872e-02
## 95-99   1.406234e+00 1.399643e+00  6.590657e-03
## 100-104 1.742800e-01 1.752970e-01 -1.016996e-03
## 105-109 1.254183e-02 1.249790e-02  4.393426e-05
## 110+    3.083579e-03 1.065312e-03  2.018268e-03
##         7.507839e+03 8.090126e+03 -5.822873e+02
##
## $tot.tae.e0
##             [,1]    [,2]      [,3]
## Female 6515.920 6695.96 -180.0396
## Male   9092.473 9551.02 -458.5469
```

```
# 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 accross 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
```

```r
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
```

```r
mex.m.Q
```

```
## [1] 0.06264442 0.23507692
```

```r
rsa.f.Q
```

```
## [1] 0.0688960 0.4660984
```

```r
rsa.m.Q
```

```
## [1] 0.0815180 0.5427579
```

```r
# 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"
```

```r
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
```

```r
# female child and adult only
plot(mex.f.ql)
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.ql)
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.ql)
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.ql)
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.ql,"Mexico","Female","Data")
  ,cbind(c(0,1,seq(5,80,5))
         ,mex.m.ql,"Mexico","Male","Data")
  ,cbind(c(0,1,seq(5,80,5))
         ,rsa.f.ql,"South Africa","Female","Data")
  ,cbind(c(0,1,seq(5,80,5))
         ,rsa.m.ql,"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 -3.16197355       Mexico Female
## 2     1 -4.74918895       Mexico Female
## 3     5 -5.53223013       Mexico Female
## 4    10 -5.88681609       Mexico Female
## 5    15 -5.50014556       Mexico Female
## 6    20 -5.30190385       Mexico Female
## 7    25 -5.12504259       Mexico Female
## 8    30 -4.86506728       Mexico Female
## 9    35 -4.52446585       Mexico Female
## 10   40 -4.16919155       Mexico Female
## 11   45 -3.79822725       Mexico Female
## 12   50 -3.39669152       Mexico Female
## 13   55 -2.98892386       Mexico Female
## 14   60 -2.50987889       Mexico Female
## 15   65 -2.00234977       Mexico Female
## 16   70 -1.42004684       Mexico Female
## 17   75 -0.80867158       Mexico Female
## 18   80 -0.16627089       Mexico Female
## 19    0 -2.96689523       Mexico   Male
## 20    1 -4.44350581       Mexico   Male
## 21    5 -5.14372825       Mexico   Male
## 22   10 -5.38795840       Mexico   Male
## 23   15 -4.71910152       Mexico   Male
## 24   20 -4.38857115       Mexico   Male
## 25   25 -4.38163845       Mexico   Male
## 26   30 -4.26868196       Mexico   Male
## 27   35 -4.02166063       Mexico   Male
## 28   40 -3.68211981       Mexico   Male
## 29   45 -3.29282126       Mexico   Male
## 30   50 -2.87288971       Mexico   Male
## 31   55 -2.44918528       Mexico   Male
## 32   60 -1.98995125       Mexico   Male
## 33   65 -1.52430199       Mexico   Male
## 34   70 -1.00769816       Mexico   Male
## 35   75 -0.45638471       Mexico   Male
## 36   80  0.15471179       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 -2.92641236 South Africa Female
## 74    1 -3.95528058 South Africa Female
## 75    5 -5.74053282 South Africa Female
## 76   10 -5.90654915 South Africa Female
## 77   15 -4.37335825 South Africa Female
## 78   20 -3.98166950 South Africa Female
## 79   25 -3.70379524 South Africa Female
## 80   30 -3.46072093 South Africa Female
## 81   35 -3.15940978 South Africa Female
## 82   40 -2.92515995 South Africa Female
## 83   45 -2.61821512 South Africa Female
## 84   50 -2.29864078 South Africa Female
## 85   55 -1.96510385 South Africa Female
## 86   60 -1.58037741 South Africa Female
## 87   65 -1.21584386 South Africa Female
## 88   70 -0.78761272 South Africa Female
## 89   75 -0.35447644 South Africa Female
## 90   80  0.14085660 South Africa Female
## 91    0 -2.71764120 South Africa   Male
## 92    1 -3.65240958 South Africa   Male
## 93    5 -4.70438113 South Africa   Male
## 94   10 -4.82348060 South Africa   Male
## 95   15 -3.61985690 South Africa   Male
## 96   20 -2.96657870 South Africa   Male
```

```
## 97   25 -2.83975529 South Africa   Male
## 98   30 -2.75786976 South Africa   Male
## 99   35 -2.63033108 South Africa   Male
## 100  40 -2.46092946 South Africa   Male
## 101  45 -2.25655659 South Africa   Male
## 102  50 -2.00630632 South Africa   Male
## 103  55 -1.73809207 South Africa   Male
## 104  60 -1.37411482 South Africa   Male
## 105  65 -1.00615718 South Africa   Male
## 106  70 -0.57086267 South Africa   Male
## 107  75 -0.07831436 South Africa   Male
## 108  80  0.49564343 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
```

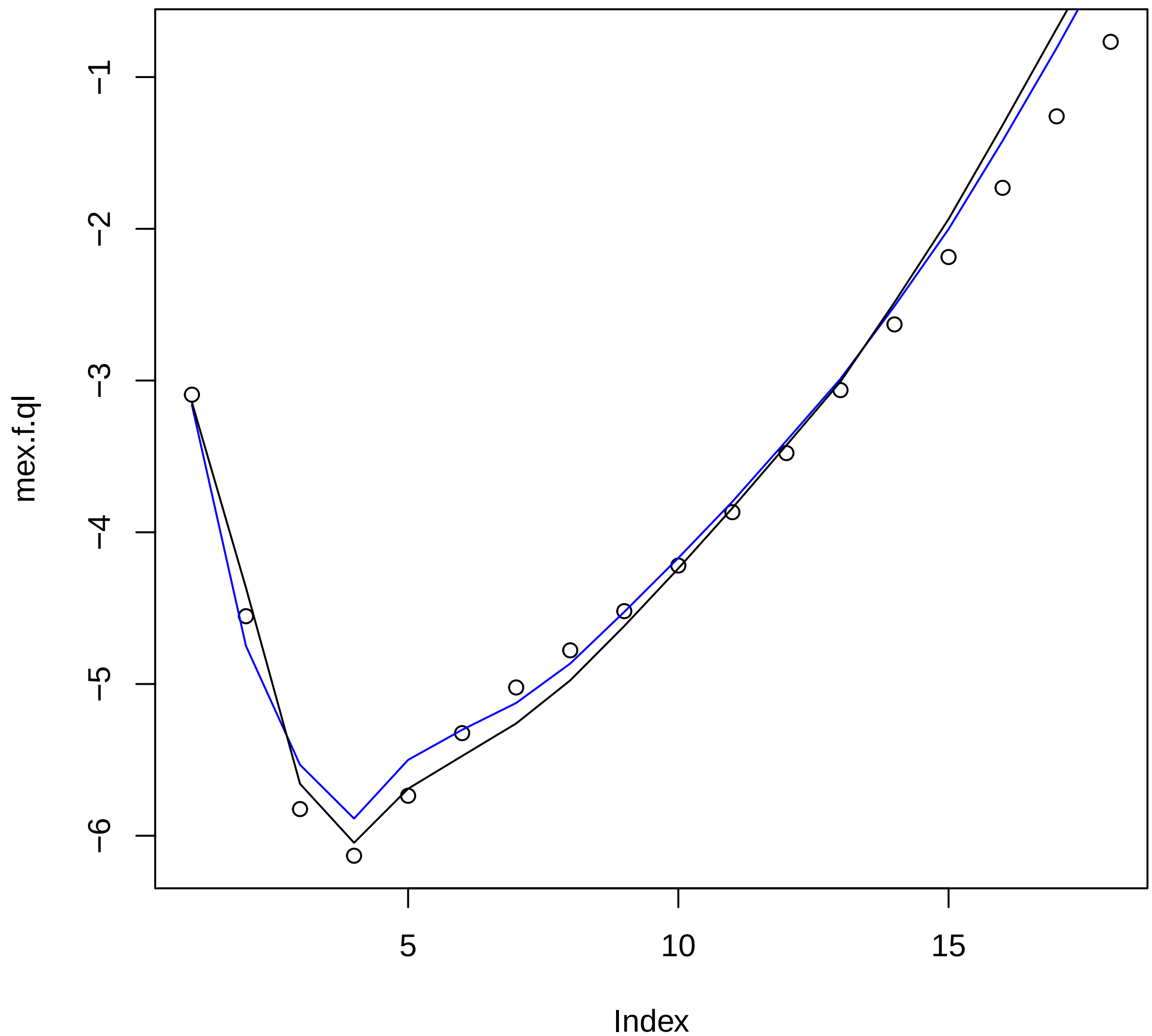
```
## 6   Predicted by SVD-Comp
## 7   Predicted by SVD-Comp
## 8   Predicted by SVD-Comp
## 9   Predicted by SVD-Comp
## 10  Predicted by SVD-Comp
## 11  Predicted by SVD-Comp
## 12  Predicted by SVD-Comp
## 13  Predicted by SVD-Comp
## 14  Predicted by SVD-Comp
## 15  Predicted by SVD-Comp
## 16  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
```

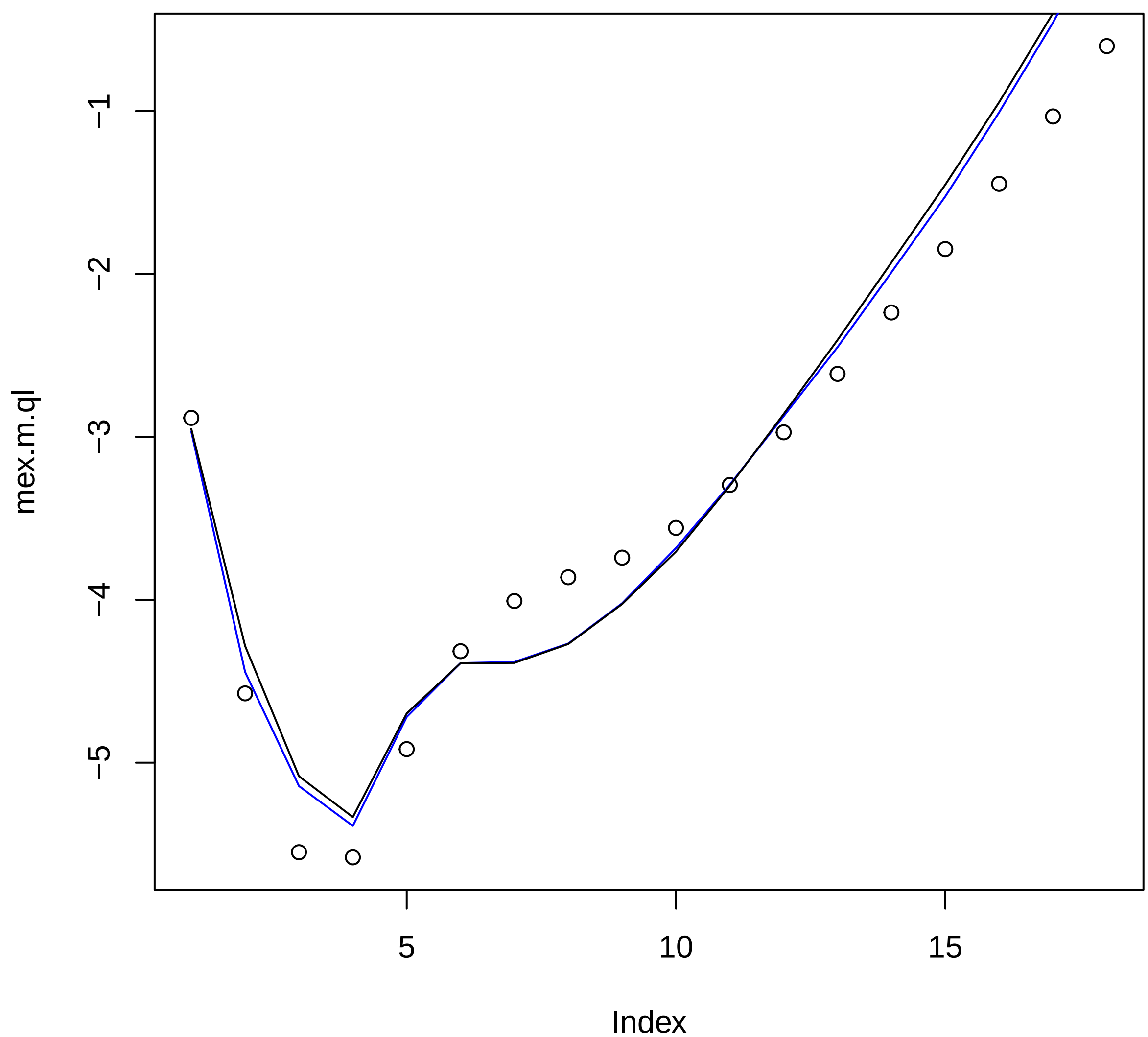
```
## 60  Predicted by Log-Quad
## 61  Predicted by Log-Quad
## 62  Predicted by Log-Quad
## 63  Predicted by Log-Quad
## 64  Predicted by Log-Quad
## 65  Predicted by Log-Quad
## 66  Predicted by Log-Quad
## 67  Predicted by Log-Quad
## 68  Predicted by Log-Quad
## 69  Predicted by Log-Quad
## 70  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
```

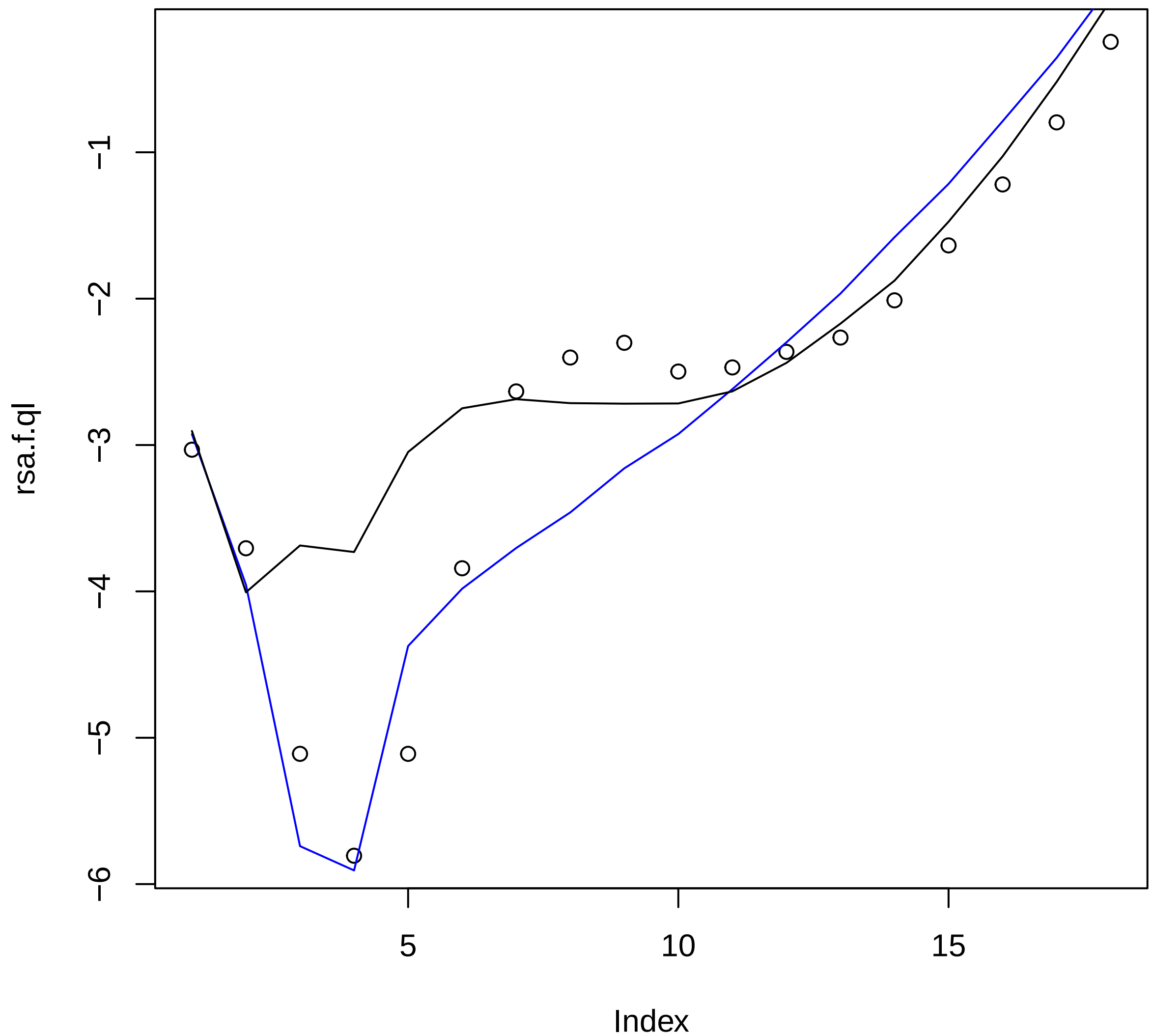
```
# plot data and predictions
ggplot(data = q.logit.data, aes(x=Age, y=Value, colour=Source)) +
  geom_line(data = q.logit.pred, aes(x=Age, y=Value, colour=Source), size=1) +
  scale_x_continuous(breaks=c(0,1,seq(5,80,5))
                     ,labels=c("0,1-4","",paste(seq(5,80,5),c(seq(9,84,5)),sep="-"))
                     ,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"))
```

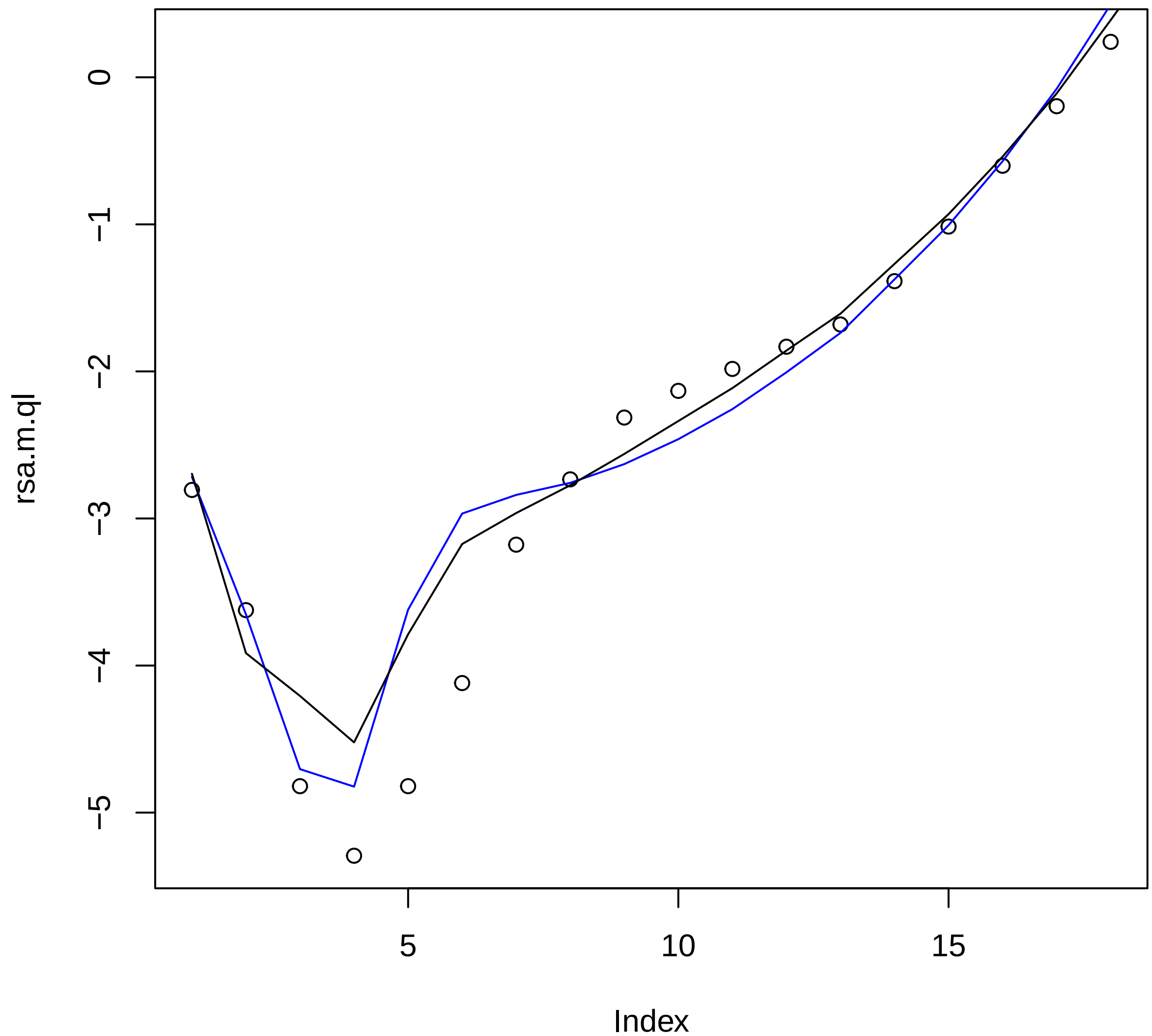

mex.f.ql
−1
−2
−3
−4
−5
−6
5
10
15
Index



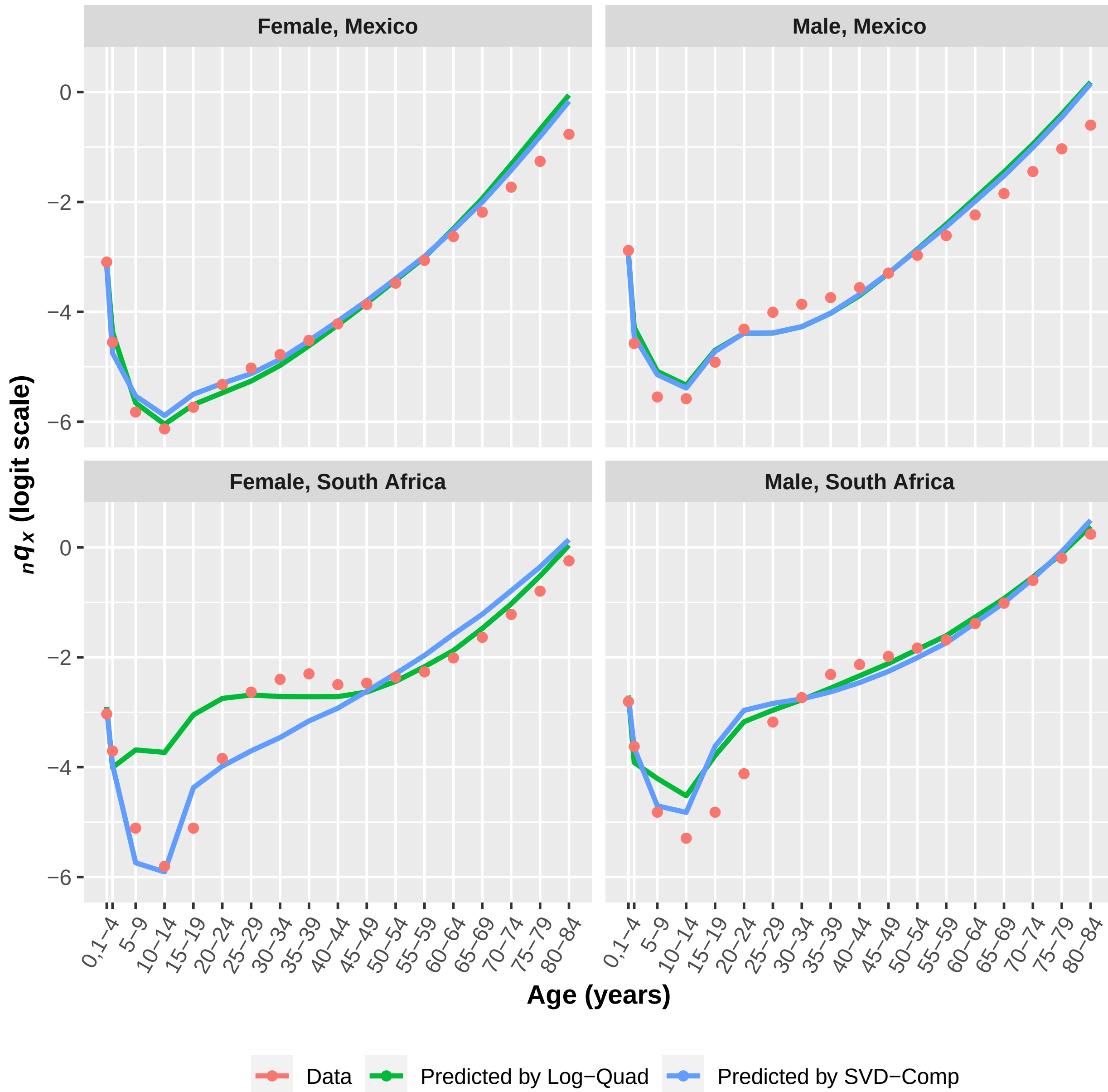
mex.m.ql
Index
−1
−2
−3
−4
−5
5
10
15

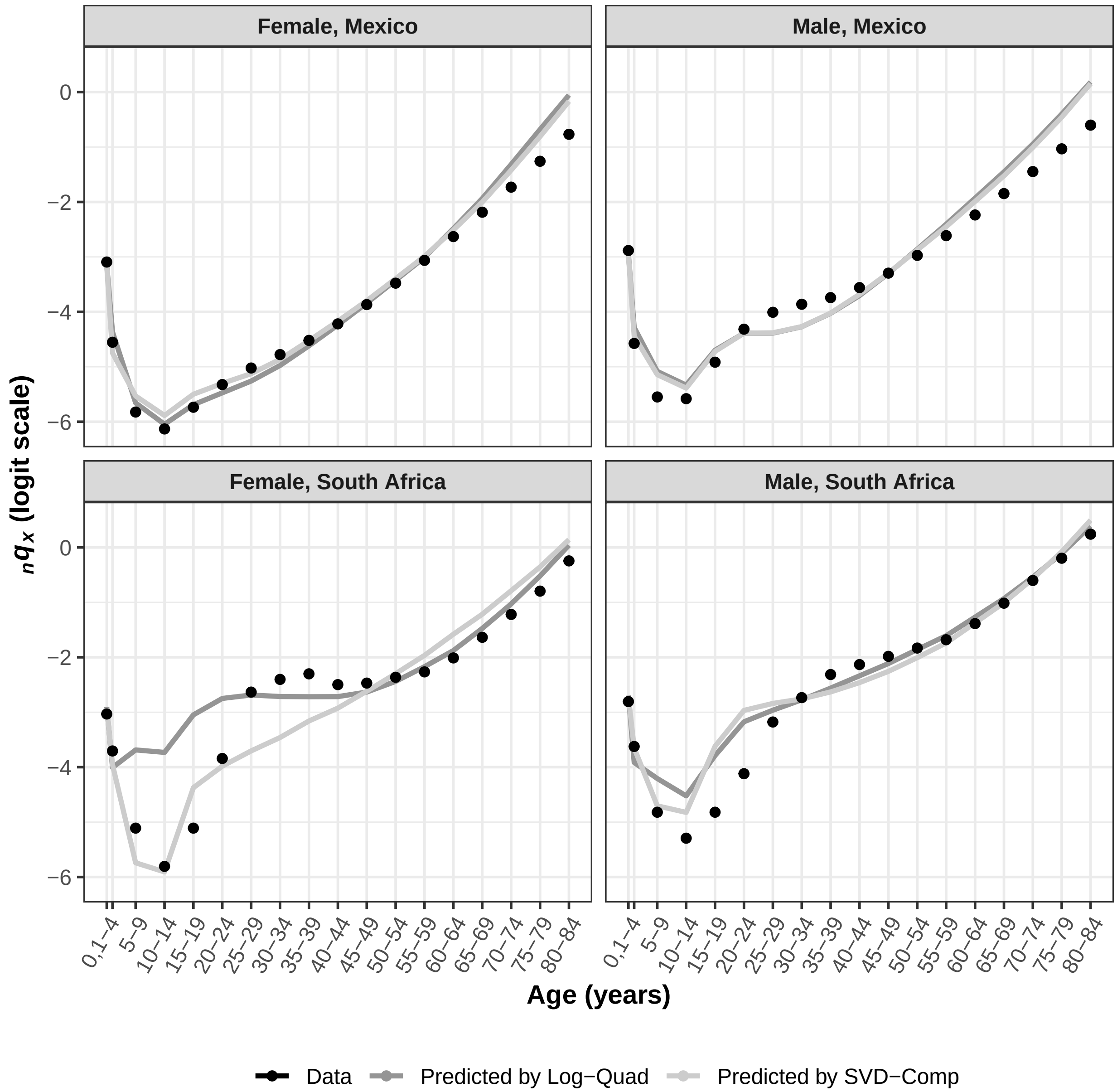


−6
−5
−4
−3
−2
−1
rsa.f.ql
5
10
15
Index



rsa.m.ql
Index
0
−1
−2
−3
−4
−5
5
10
15


Female, Mexico
Male, Mexico
Female, South Africa
Male, South Africa
$_nq_x$ (logit scale)
0
−2
−4
−6
0,1−4
5−9
10−14
15−19
20−24
25−29
30−34
35−39
40−44
45−49
50−54
55−59
60−64
65−69
70−74
75−79
80−84
Age (years)
Data
Predicted by Log−Quad
Predicted by SVD−Comp



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 informatin necessary to do predicdtions
#   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(
    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")
```

```r
  # library(plyr)
  # 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)

}

# create and same the models object for the package
mods <- 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
mods
```

```
## $female
## $female$components
##            0           1           2           3
## 1 -967.06359 -1116.42675 -1160.67713 -1185.541965
## 2  -34.62381   -72.42237   -70.75661   -67.133683
## 3   13.93553    10.33163     5.79337     1.243374
## 4  -13.12383   -16.51958   -14.17303   -21.655968
##             4            5            6           7
## 1 -1200.59141 -1213.848761 -1223.347096 -1234.05645
## 2   -60.41674   -59.597766   -57.046361   -55.29970
## 3     7.62036     1.294989    -7.214563   -13.06925
## 4   -16.60079   -16.194220   -16.980807   -14.76249
##             8           9           10            11
## 1 -1241.83256 -1247.63604 -1250.070705 -1250.0322918
## 2   -51.26389   -49.19489   -48.232620   -47.3139911
## 3   -20.43383   -18.83042   -24.150974   -20.7207516
## 4   -16.46759   -14.64097    -8.764605    -0.2288842
##              12           13           14          15
## 1 -1246.1822248 -1238.882198 -1228.469720 -1216.73747
## 2   -43.7695119   -41.526638   -36.808136   -31.64788
## 3   -15.7194647   -15.253874   -10.074954   -10.06274
## 4     0.8750684     2.545957     8.693657    14.99165
##             16           17            18           19
## 1 -1205.039182 -1198.404870 -1190.7440147 -1187.622618
## 2   -27.068895   -26.160447   -21.9817380   -22.184143
## 3    -2.535439    -1.983989    -0.8463695     1.314851
## 4    13.202427    10.831706    14.0841622    14.456911
##             20           21            22
## 1 -1185.666593 -1184.991000 -1182.6104478
## 2   -24.400945   -25.910659   -26.2873324
## 3    -2.263732    -3.847778    -0.7704208
## 4    13.246068    14.932044    14.1814834
##              23            24            25
## 1 -1181.9524585 -1179.7464010 -1177.2609100
## 2   -28.3384139   -27.3000553   -26.0251271
## 3    -0.9584123     0.4630131    -0.5259946
## 4    15.6530701    14.0395728    13.1335888
```

```
##            26           27           28           29
## 1 -1174.665096 -1172.011108 -1169.919034 -1164.874386
## 2   -24.667903   -23.831555   -24.748574   -19.342220
## 3     2.278062     2.717301     2.125829     5.650919
## 4    12.432587    10.609123    13.260753     7.778443
##            30           31           32           33
## 1 -1160.596930 -1159.339793 -1154.198379 -1150.729519
## 2   -17.998662   -17.637535   -16.619874   -15.605433
## 3     4.807705     2.335622     3.679062     2.127739
## 4     6.662782     9.314282     9.639564    11.408764
##            34           35           36           37
## 1 -1145.500549 -1139.907991 -1136.272332 -1132.446158
## 2   -12.279987    -8.462273    -8.054408    -6.824041
## 3     6.973417     7.144934     5.839328     4.857799
## 4     8.956947     5.925578     6.065168     5.988968
##            38           39            40           41
## 1 -1125.740216 -1122.151814 -1116.1933834 -1113.199320
## 2    -2.655093    -2.394760     0.1845252     2.417559
## 3     8.325669     7.410671     7.6374592     5.174530
## 4     3.202456     4.730808    -0.5246675     5.686984
##            42           43           44           45
## 1 -1105.998903 -1101.905142 -1096.987065 -1091.000766
## 2     5.046735     7.635766     9.708755    11.884754
## 3     7.636616     6.673712     6.446823     8.613008
## 4     2.030274     2.290401     3.320610     1.500237
##            46            47            48
## 1 -1086.926157 -1081.1253010 -1075.1392198
## 2    12.589947    16.3837960    17.0330743
## 3     7.367336     7.6638085     8.0723392
## 4     2.854477     0.3350641    -0.3798282
##             49            50            51
## 1 -1069.6909048 -1062.9816331 -1059.8271223
## 2    19.2101054    18.4535749    20.1028655
## 3     7.6229809     9.1228369     6.4103371
## 4    -0.1630624     0.3398623     0.8686003
##             52            53            54
## 1 -1052.6329486 -1.048269e+03 -1042.4438701
## 2    19.6028334  2.046463e+01    20.9897291
## 3     8.3535879  7.205368e+00     7.8265477
## 4    -0.9427321  2.565629e-03    -0.7986301
##            55           56           57           58
## 1 -1037.403611 -1031.520924 -1026.368722 -1019.969688
## 2    21.954840    22.320705    23.283527    22.622362
## 3     7.509202     8.321401     7.656460     7.666306
## 4    -1.294563    -1.428385    -1.694812    -2.026934
##            59           60           61          62
## 1 -1014.379453 -1006.626419 -1002.259704 -994.331302
## 2    22.651523    20.621303    22.882050   21.616408
## 3     6.941118     8.200623     7.113694    8.337416
## 4    -1.740999    -2.878266    -2.644821   -3.539225
##           63          64          65          66
## 1 -988.593297 -981.892512 -974.792835 -969.486954
## 2   21.256245   21.184119   21.080291   21.555145
## 3    6.138293    7.047931    7.819833    5.862483
```

```
## 4   -3.551409   -4.084213   -4.300689   -4.480041
##           67          68          69          70
## 1 -962.450087 -955.122869 -948.579350 -939.377422
## 2   22.062275   21.253836   21.802704   20.018264
## 3    5.939898    6.263290    4.823202    6.867731
## 4   -4.707119   -5.197629   -4.953301   -6.007324
##           71          72          73          74
## 1 -934.574380 -925.177173 -918.121756 -910.148373
## 2   21.591041   20.464167   20.999066   21.214247
## 3    3.717287    5.052876    4.159848    4.935873
## 4   -5.538665   -6.440958   -6.490337   -6.672901
##           75          76           77          78
## 1 -902.415302 -895.211161 -888.5366830 -879.625484
## 2   21.089770   21.827967   22.5895294   22.447188
## 3    4.237915    3.586668    0.8598131    2.999239
## 4   -6.937072   -7.106386   -6.7965676   -6.997127
##           79          80           81           82
## 1 -872.622870 -863.941345 -858.4127293 -849.1456295
## 2   23.465056   23.886859   26.1314468   25.5971050
## 3    1.354600    2.634924    0.7926711    0.7381374
## 4   -6.314681   -6.614000   -5.5627775   -5.8837373
##            83           84          85          86
## 1 -841.5661407 -833.3600631 -826.362564 -818.822075
## 2   26.3955879   26.5201375   27.533205   28.690462
## 3   -0.2833676   -0.5933839   -1.547169   -2.421228
## 4   -5.5127557   -5.4616734   -5.014329   -4.795563
##           87          88          89          90
## 1 -812.045683 -805.007022 -798.301465 -790.280434
## 2   29.763003   30.479808   31.899681   31.498954
## 3   -3.500167   -4.198494   -5.417725   -5.141433
## 4   -4.150733   -3.753928   -3.227601   -3.140871
##          91          92          93          94
## 1 -785.30725 -777.807561 -771.495208 -765.035356
## 2   33.86244   33.967326   34.932921   35.505241
## 3   -7.04729   -7.270859   -8.034129   -8.483344
## 4   -2.07547   -2.037961   -1.629650   -1.371734
##           95           96           97          98
## 1 -757.968450 -751.7664105 -745.7283853 -739.864734
## 2   35.739319   36.4675661   37.1767650   37.863291
## 3   -9.098187   -9.7244769  -10.3229093  -10.891338
## 4   -1.159667   -0.8118539   -0.4712219   -0.139356
##            99          100          101        102
## 1 -734.1856833 -728.7008911 -723.4193376 -718.34897
## 2   38.5233884   39.1533831   39.7495764   40.30856
## 3  -11.4275139  -11.9295953  -12.3957329  -12.82433
## 4    0.1822079    0.4917077    0.7875542    1.06810
##          103         104         105         106
## 1 -713.49683 -708.868520 -704.468162 -700.298395
## 2   40.82717   41.302733   41.733030   42.116555
## 3  -13.21415  -13.564224  -13.874042  -14.143443
## 4    1.33196    1.577768    1.804497    2.011377
##           107         108         109
## 1 -696.360104 -692.652554 -689.173254
## 2   42.452342   42.740350   42.980957
```

```
## 3  -14.372758  -14.562711  -14.714567
## 4    2.197873    2.363894    2.509534
##
## $female$components.sm
##            0           1            2            3
## 1 -967.06359 -1116.42675 -1160.677127 -1185.541965
## 2  -34.62381   -61.59745   -66.828173   -65.824139
## 3   13.93553    10.33163     6.641742     4.629219
## 4  -13.12383   -16.51958   -14.173034   -16.966043
##             4           5            6           7
## 1 -1200.591409 -1213.84876 -1223.347096 -1234.05645
## 2   -62.370573   -59.44137   -57.091161   -54.60673
## 3     2.284858    -1.42379    -6.604188   -12.06516
## 4   -17.012593   -16.69381   -16.093112   -15.13474
##            8           9           10           11
## 1 -1241.83256 -1247.63604 -1250.070705 -1250.032292
## 2   -51.97290   -49.78493   -48.143107   -46.341131
## 3   -16.48078   -19.18402   -20.009431   -19.048684
## 4   -13.51122   -10.92441    -7.424961    -3.446617
##             12           13           14           15
## 1 -1246.1822248 -1238.882198 -1228.469720 -1216.737466
## 2   -43.8351392   -40.565305   -36.501409   -32.126216
## 3   -16.8304688   -14.063646   -11.097858    -8.015854
## 4     0.5371768     4.267587     7.550862    10.103340
##            16           17           18           19
## 1 -1205.039182 -1198.404870 -1190.744015 -1187.622618
## 2   -28.376939   -25.577105   -23.703827   -23.308137
## 3    -5.081881    -2.785424    -1.465923    -1.146218
## 4    11.760113    12.695453    13.258295    13.678102
##            20           21           22           23
## 1 -1185.666593 -1184.991000 -1182.610448 -1181.952459
## 2   -24.250490   -25.523181   -26.581652   -27.183573
## 3    -1.453918    -1.678645    -1.394545    -0.767532
## 4    14.014580    14.253309    14.339671    14.192787
##             24            25           26
## 1 -1.179746e+03 -1177.2609100 -1174.665096
## 2 -2.693517e+01   -26.0006058   -24.967314
## 3 -6.415651e-02     0.6987692     1.553506
## 4  1.377196e+01    13.1255928    12.343399
##            27           28           29           30
## 1 -1172.011108 -1169.919034 -1164.874386 -1160.596930
## 2   -24.064771   -22.690180   -20.539429   -18.649151
## 3     2.426116     3.226926     3.779673     3.874639
## 4    11.474509    10.556011     9.732644     9.230671
##            31           32           33           34
## 1 -1159.339793 -1154.198379 -1150.729519 -1145.500549
## 2   -17.471891   -16.358417   -14.636862   -12.167152
## 3     3.683904     3.731120     4.337085     5.250434
## 4     9.112657     9.125898     8.889999     8.226376
##            35           36           37           38
## 1 -1139.907991 -1136.272332 -1132.446158 -1125.740216
## 2    -9.694464    -7.791922    -5.906852    -3.781192
## 3     5.923716     6.220328     6.490476     6.870077
## 4     7.256373     6.210086     5.224487     4.349493
```

```
##            39           40           41           42
## 1 -1122.151814 -1116.1933834 -1113.199320 -1105.998903
## 2    -1.793231     0.2504646     2.562338     5.017098
## 3     7.081472     6.9845926     6.813361     6.800112
## 4     3.649478     3.1901664     2.943633     2.789999
##            43           44           45           46
## 1 -1101.905142 -1096.987065 -1091.000766 -1086.926157
## 2     7.418969     9.599637    11.533255    13.455465
## 3     6.951748     7.208403     7.484222     7.670800
## 4     2.618388     2.369529     2.005285     1.519131
##             47            48           49
## 1 -1081.1253010 -1075.1392198 -1069.690905
## 2    15.4655760    17.1615204    18.307981
## 3     7.7794008     7.8753372     7.933537
## 4     0.9846837     0.5369035     0.260222
##             50            51            52
## 1 -1062.9816331 -1.059827e+03 -1052.6329486
## 2    19.0079381  1.952881e+01    19.9643836
## 3     7.8818561  7.752102e+00     7.6589082
## 4     0.1115508 -2.499636e-02    -0.2253186
##             53            54           55
## 1 -1048.2691963 -1042.4438701 -1037.403611
## 2    20.4637380    21.1038666    21.783338
## 3     7.6442015     7.6879951     7.759494
## 4    -0.4921269    -0.7957322    -1.105625
##            56           57           58           59
## 1 -1031.520924 -1026.368722 -1019.969688 -1014.379453
## 2    22.373553    22.706613    22.618448    22.205764
## 3     7.789956     7.726705     7.617005     7.558489
## 4    -1.399005    -1.670910    -1.939756    -2.233052
##            60           61          62          63
## 1 -1006.626419 -1002.259704 -994.331302 -988.593297
## 2    21.894361    21.886303   21.736853   21.426469
## 3     7.555718     7.510357    7.357590    7.160431
## 4    -2.563884    -2.921992   -3.284419   -3.630726
##           64          65          66          67
## 1 -981.892512 -974.792835 -969.486954 -962.450087
## 2   21.257709   21.309366   21.513714   21.630644
## 3    6.993939    6.786109    6.467797    6.133121
## 4   -3.949400   -4.239467   -4.509357   -4.772319
##           68          69          70          71
## 1 -955.122869 -948.579350 -939.377422 -934.574380
## 2   21.513409   21.231610   20.971855   20.910797
## 3    5.867894    5.634083    5.343360    4.998749
## 4   -5.040037   -5.318771   -5.607579   -5.898710
##           72          73          74          75
## 1 -925.177173 -918.121756 -910.148373 -902.415302
## 2   20.903434   20.971000   21.139742   21.405125
## 3    4.717958    4.534541    4.299040    3.835631
## 4   -6.178490   -6.429330   -6.632457   -6.769729
##           76          77          78          79
## 1 -895.211161 -888.536683 -879.625484 -872.622870
## 2   21.841956   22.329009   22.806390   23.454771
## 3    3.167824    2.552454    2.174362    1.933562
```

```
## 4   -6.825044   -6.790003   -6.668706   -6.477795
##            80          81           82           83
## 1 -863.941345 -858.412729 -849.1456295 -841.5661407
## 2   24.354318   25.262802   25.8583389   26.2946145
## 3    1.617996    1.132264    0.5240037   -0.1358115
## 4   -6.244055   -5.998575   -5.7611406   -5.5261060
##             84          85          86          87
## 1 -833.3600631 -826.362564 -818.822075 -812.045683
## 2   26.8493061   27.656640   28.651518   29.657400
## 3   -0.8432843   -1.627748   -2.487684   -3.372676
## 4   -5.2676204   -4.960709   -4.597433   -4.188802
##            88          89          90          91
## 1 -805.007022 -798.301465 -790.280434 -785.307248
## 2   30.597183   31.439861   32.280663   33.240479
## 3   -4.222789   -5.015733   -5.785973   -6.555585
## 4   -3.754421   -3.310231   -2.867640   -2.442621
##            92          93          94          95
## 1 -777.807561 -771.495208 -765.035356 -757.968450
## 2   34.103860   34.801618   35.382635   35.906385
## 3   -7.281285   -7.930190   -8.529260   -9.117802
## 4   -2.054009   -1.708789   -1.395284   -1.093020
##            96           97           98           99
## 1 -751.766410 -745.7283853 -739.8647339 -734.1856833
## 2   36.498868   37.1679294   37.8472525   38.5047905
## 3   -9.709137  -10.2933436  -10.8566131  -11.3902166
## 4   -0.786448   -0.4713708   -0.1525152    0.1626929
##            100          101         102         103
## 1 -728.7008911 -723.4193376 -718.348973 -713.496834
## 2   39.1325345   39.7266231   40.283680   40.800640
## 3  -11.8903064  -12.3547394  -12.781972  -13.170783
## 4    0.4685314    0.7615351    1.039635    1.301097
##           104         105        106         107
## 1 -708.868520 -704.468162 -700.29839 -696.360104
## 2   41.274859   41.704213   42.08657   42.415221
## 3  -13.520283  -13.829048  -14.09376  -14.306809
## 4    1.543929    1.764582    1.95705    2.114626
##           108         109
## 1 -692.652554 -689.173254
## 2   42.668475   42.827764
## 3  -14.461635  -14.562620
## 4    2.234469    2.320031
##
## $female$aml
##
## Call:
## lm(formula = aml ~ cm + cml + cmls + cmlc)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##     2.57631     -4.24789      2.04423      0.29814
##        cmlc
##     0.01891
##
##
```

```
## $female$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   7.846e-03    1.243e-02   -3.554e-03   -5.499e-04
##        cmlc           am         amls         amlc
##  -4.315e-05   -2.764e-03    3.488e-04   -2.026e-05
##      cmlaml
##  -3.680e-04
##
##
## $female$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   -0.220526     0.379283    -0.118930    -0.021275
##        cmlc           am         amls         amlc
##   -0.001612    -0.006436     0.011503     0.001648
##      cmlaml
##   -0.006461
##
##
## $female$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   -0.380109     0.873734    -0.222922    -0.027430
##        cmlc           am         amls         amlc
##   -0.002306    -0.085602     0.022832    -0.003087
##      cmlaml
##   -0.042369
##
##
## $female$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##    0.724349    -1.510471     0.413895     0.082559
```

```
##        cmlc           am         amls         amlc
##    0.005291     0.040090    -0.010462    -0.002211
##      cmlaml
##   -0.001176
##
##
## $female$offset
## [1] 10
##
## $female$q0
##
## Call:
## lm(formula = as.numeric(ql[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml         cmls
##    -0.93982      0.66978     -0.03579
##
##
## $female$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 -950.863257 -1105.645300 -1145.830829 -1170.045016
## 2  -48.660300   -82.916090   -77.040650   -76.044167
## 3    5.304070     3.819371     6.214911    13.624958
## 4   -4.844289     1.906063     1.176176    -4.356452
##              4            5            6            7
## 1 -1184.838051 -1195.993962 -1205.571607 -1212.430334
## 2   -68.488530   -64.970151   -64.761794   -61.034974
## 3    11.940271    12.454002    14.659338    14.997479
## 4    -1.143279    -2.686364    -5.034853    -7.415561
##             8            9           10           11
## 1 -1219.50861 -1225.501927 -1227.436601 -1227.873577
## 2   -56.65388   -53.858028   -50.748271   -46.510385
## 3    14.89790    18.097414    13.371990    14.425385
## 4   -10.14122    -5.502163    -2.466852    -1.101847
##              12            13           14
```

```
## 1 -1224.2421390 -1217.5655311 -1204.305408
## 2   -42.5960518   -39.7153428   -31.649701
## 3    12.4099472    10.9572615     4.735743
## 4    -0.2896488    -0.9083504     4.107056
##             15           16           17           18
## 1 -1189.842527 -1171.098517 -1155.596690 -1139.847587
## 2   -27.030830   -17.917753   -13.688541    -7.099474
## 3     3.536273    -0.620052    -3.385913    -6.286673
## 4     5.642321     7.190544     7.101265    10.759794
##             19           20          21          22
## 1 -1133.162589 -1129.186456 -1126.66871 -1125.53390
## 2    -7.839506   -10.262136   -10.67072   -12.68911
## 3    -8.787657    -8.598965   -10.78834   -11.18347
## 4    11.514924    12.427635    14.32252    11.24353
##            23          24          25          26
## 1 -1124.93833 -1125.00040 -1124.57809 -1124.72970
## 2   -13.09728   -12.66191   -11.64926   -10.86549
## 3   -12.29240   -12.40905   -13.78979   -13.89663
## 4    12.25808    11.93006    11.22632    11.74608
##             27           28           29           30
## 1 -1123.767298 -1122.194187 -1120.744562 -1119.320582
## 2    -9.495757    -7.771659    -6.482400    -9.001799
## 3   -14.477711   -15.525585   -14.996245   -14.499726
## 4    10.763199     8.874942     8.489359     7.894918
##             31          32           33           34
## 1 -1117.331833 -1114.03053 -1111.166479 -1108.128856
## 2    -4.934678    -5.65809    -4.070336    -3.762962
## 3   -15.329862   -14.63863   -14.956882   -15.089880
## 4     7.389457     5.04400     4.329481     3.126856
##             35           36           37            38
## 1 -1104.414423 -1101.029853 -1097.680485 -1092.0520256
## 2    -3.667185    -2.402252    -2.106815     0.4665319
## 3   -14.553488   -13.479919   -13.283589   -13.6131906
## 4     1.641931     1.180550     2.269181    -0.4243616
##             39           40           41           42
## 1 -1087.754421 -1082.390455 -1078.725991 -1072.274254
## 2     1.674519     1.256575     4.336764     4.417585
## 3   -12.470153   -12.724072   -11.780970   -10.861907
## 4    -1.004769    -3.217685    -2.588940    -4.469861
##             43           44           45           46
## 1 -1067.375169 -1062.339984 -1055.995066 -1051.321006
## 2     7.109949     7.770366     8.959874    10.682912
## 3   -11.042079   -10.140877    -9.457318    -8.926584
## 4    -4.179554    -4.629467    -5.583978    -6.060575
##             47           48           49           50
## 1 -1045.768389 -1039.903719 -1034.101221 -1027.496412
## 2    11.986969    12.524503    13.998323    14.022496
## 3    -8.374570    -7.927354    -7.241141    -6.881568
## 4    -6.707329    -7.100805    -7.286729    -8.032024
##             51           52           53           54
## 1 -1023.561891 -1016.245133 -1011.161578 -1004.987709
## 2    15.922963    16.249062    16.900627    18.116859
## 3    -5.975928    -5.758346    -4.823868    -4.452465
## 4    -7.413003    -8.280532    -8.162680    -8.468091
```

```
##           55          56          57          58
## 1 -999.580196 -993.879586 -988.493227 -982.123429
## 2   19.318894   19.890178   20.907516   20.818118
## 3   -4.176508   -3.141196   -2.538588   -2.082766
## 4   -8.260869   -8.702443   -8.605966   -9.424347
##           59          60          61          62
## 1 -976.309095 -968.982318 -965.134572 -957.627091
## 2   21.275452   20.590599   21.552710   20.997094
## 3   -1.941337   -2.428316   -1.339085   -1.212979
## 4   -8.776613   -9.764840   -8.405350   -9.285591
##           63           64            65           66
## 1 -951.795347 -945.6780170 -9.393305e+02 -934.6490139
## 2   21.105912   21.1357210  2.104615e+01   21.6802024
## 3   -0.557739   -0.4139152  5.375161e-03    0.5502371
## 4   -9.022800   -9.1312151 -9.381004e+00   -7.8177728
##            67          68          69          70
## 1 -928.3750850 -921.941408 -916.137130 -908.673170
## 2   21.5969024   21.547883   21.586459   20.147452
## 3    0.9890884    1.277918    2.005082    1.757946
## 4   -8.1902881   -8.105078   -7.246373   -7.837652
##           71          72          73          74
## 1 -904.400080 -896.158742 -889.888352 -883.124952
## 2   21.438725   20.257067   20.360080   20.103136
## 3    3.466558    2.807345    3.048863    3.188181
## 4   -6.016203   -7.096097   -6.674785   -6.707961
##           75          76          77          78
## 1 -876.535632 -870.293339 -864.612319 -856.896197
## 2   20.014276   20.237892   20.573149   20.157213
## 3    3.178357    4.062651    5.284223    4.434957
## 4   -6.569859   -6.065644   -4.406236   -5.386502
##           79          80          81          82
## 1 -850.688462 -843.089066 -838.005187 -829.784778
## 2   20.509879   20.560600   21.883046   21.120380
## 3    4.781067    4.128123    4.622928    5.153977
## 4   -4.167972   -4.493385   -3.009982   -2.826041
##           83          84           85           86
## 1 -823.141872 -815.674235 -809.6706738 -802.77854967
## 2   21.384572   21.091125   22.0978991   22.64913879
## 3    5.667236    6.216076    6.6238652    7.40325770
## 4   -1.993051   -1.699781   -0.6893336   -0.09097252
##          87          88          89         90
## 1 -796.46842 -790.134455 -784.059612 -777.11646
## 2   23.40800   23.813666   24.766145   24.74456
## 3    7.88393    8.272794    8.809715    8.90082
## 4    0.85021    1.570566    2.601095    2.87672
##           91          92          93          94
## 1 -772.251501 -765.477027 -759.744272 -754.005538
## 2   26.644474   26.873583   27.722898   28.202255
## 3    9.782357   10.204049   10.708470   11.146618
## 4    4.429564    4.584059    5.279258    5.718869
##          95          96          97         98
## 1 -747.93822 -742.455154 -737.129547 -731.96894
## 2   27.51279   28.003038   28.486265   28.96011
## 3   11.22845   11.569905   11.887585   12.18060
```

```
## 4    6.46651    7.031989    7.565452    8.06514
##           99         100         101         102
## 1 -726.980395 -722.170296 -717.54423 -713.107095
## 2   29.422052   29.869605   30.30025   30.711551
## 3   12.448184   12.689932   12.90545   13.094625
## 4    8.529603    8.957495    9.34787    9.700069
##          103        104        105        106
## 1 -708.86257 -704.81341 -700.96128 -697.30651
## 2   31.10132   31.46749   31.80840   32.12263
## 3   13.25770   13.39495   13.50704   13.59470
## 4   10.01373   10.28890   10.52595   10.72568
##          107        108        109
## 1 -693.84843 -690.58519 -687.51378
## 2   32.40927   32.66762   32.89761
## 3   13.65907   13.70121   13.72263
## 4   10.88923   11.01803   11.11390
##
## $male$components.sm
##            0            1            2           3
## 1 -950.863257 -1105.645300 -1145.830829 -1170.04502
## 2  -48.660300   -71.766788   -75.443637   -73.80052
## 3    5.304070     3.819371     8.179914    10.21659
## 4   -4.844289     1.906063     1.176176    -1.87098
##             4            5            6            7
## 1 -1184.838051 -1195.993962 -1205.571607 -1212.430334
## 2   -69.846833   -66.310036   -63.610063   -60.621137
## 3    11.889481    13.060381    13.994934    14.777123
## 4    -2.707908    -3.787355    -4.922819    -5.723532
##             8            9           10           11
## 1 -1219.508606 -1225.501927 -1227.436601 -1227.873577
## 2   -57.171765   -53.765654   -50.308375   -46.579985
## 3    15.269386    15.239624    14.582710    13.382939
## 4    -5.761845    -4.920103    -3.493227    -1.889811
##              12           13           14           15
## 1 -1224.2421390 -1217.565531 -1204.305408 -1189.842527
## 2   -42.5982179   -37.946568   -32.244508   -25.866891
## 3    11.6060998     9.125311     6.100968     2.911231
## 4    -0.2866745     1.378195     3.155058     4.954824
##              16           17           18           19
## 1 -1171.0985169 -1155.596690 -1139.847587 -1133.162589
## 2   -19.4144797   -13.806587   -10.087785    -9.014506
## 3    -0.1970359    -3.092689    -5.599124    -7.570684
## 4     6.6637287     8.250337     9.700474    10.920692
##            20          21          22          23
## 1 -1129.186456 -1126.66871 -1125.53390 -1124.93833
## 2    -9.792105   -11.01471   -12.06924   -12.56684
## 3    -9.049644   -10.21226   -11.17702   -11.99452
## 4    11.770107    12.17732    12.21059    12.02621
##           24          25          26           27
## 1 -1125.00040 -1124.57809 -1124.72970 -1123.767298
## 2   -12.34510   -11.62369   -10.61254    -9.374351
## 3   -12.71814   -13.38198   -13.97963   -14.476037
## 4    11.74368    11.37715    10.87439    10.204842
##            28           29           30           31
```

```
## 1 -1122.194187 -1120.744562 -1119.320582 -1117.331833
## 2    -8.194900    -7.533769    -7.055412    -6.147933
## 3   -14.800971   -14.921537   -14.924744   -14.911126
## 4     9.405308     8.535719     7.607002     6.587781
##            32           33           34           35
## 1 -1114.030527 -1111.166479 -1108.128856 -1104.414423
## 2    -5.196161    -4.440752    -3.847663    -3.282406
## 3   -14.895037   -14.834422   -14.651561   -14.303786
## 4     5.479733     4.351674     3.300660     2.384659
##            36            37            38
## 1 -1101.029853 -1097.6804848 -1.092052e+03
## 2    -2.504133    -1.3847097 -1.100005e-02
## 3   -13.876058   -13.4893370 -1.314561e+01
## 4     1.570168     0.7445275 -1.825218e-01
##            39           40           41           42
## 1 -1087.754421 -1082.390455 -1078.725991 -1072.274254
## 2     1.193020     2.288012     3.621337     5.074808
## 3   -12.767186   -12.308817   -11.780555   -11.233968
## 4    -1.186885    -2.159332    -3.008514    -3.716946
##            43           44           45           46
## 1 -1067.375169 -1062.339984 -1055.995066 -1051.321006
## 2     6.535885     7.872123     9.162433    10.502307
## 3   -10.694302   -10.131799    -9.541503    -8.959401
## 4    -4.325958    -4.893564    -5.454437    -6.001989
##            47           48           49           50
## 1 -1045.768389 -1039.903719 -1034.101221 -1027.496412
## 2    11.710051    12.722101    13.633814    14.531991
## 3    -8.402823    -7.856758    -7.302606    -6.733666
## 4    -6.505760    -6.939509    -7.294502    -7.578263
##            51           52           53           54
## 1 -1023.561891 -1016.245133 -1011.161578 -1004.987709
## 2    15.466927    16.309131    17.140039    18.111585
## 3    -6.154897    -5.577715    -5.013549    -4.459451
## 4    -7.809539    -8.006414    -8.177884    -8.330621
##           55          56          57          58
## 1 -999.580196 -993.879586 -988.493227 -982.123429
## 2   19.093208   19.920061   20.524470   20.858182
## 3   -3.886412   -3.292526   -2.754869   -2.360891
## 4   -8.479296   -8.638064   -8.802157   -8.944533
##           59          60          61          62
## 1 -976.309095 -968.982318 -965.134572 -957.627091
## 2   20.978643   21.037847   21.126263   21.140492
## 3   -2.098806   -1.850329   -1.518738   -1.120510
## 4   -9.035515   -9.068471   -9.064973   -9.047481
##            63           64            65          66
## 1 -951.7953472 -945.6780170 -939.33047206 -934.649014
## 2   21.1154644   21.1442251   21.26772386   21.452733
## 3   -0.7160729   -0.3218424    0.08375592    0.510953
## 4   -9.0055353   -8.8986937   -8.70102358   -8.433010
##            67          68          69          70
## 1 -928.3750850 -921.941408 -916.137130 -908.673170
## 2   21.5468001   21.471893   21.210255   20.925187
## 3    0.9446219    1.375673    1.813928    2.258296
## 4   -8.1393628   -7.844207   -7.548039   -7.261083
```

```
##            71          72          73          74
## 1 -904.400080 -896.158742 -889.888352 -883.124952
## 2   20.768062   20.556869   20.323695   20.181002
## 3    2.647993    2.911682    3.084682    3.291261
## 4   -7.014848   -6.826665   -6.663637   -6.455029
##            75          76          77          78
## 1 -876.535632 -870.293339 -864.612319 -856.896197
## 2   20.158193   20.253195   20.345056   20.391131
## 3    3.627860    4.061842    4.421784    4.576632
## 4   -6.145808   -5.740310   -5.291531   -4.841437
##            79          80          81          82
## 1 -850.688462 -843.089066 -838.005187 -829.784778
## 2   20.543221   20.884244   21.226213   21.328107
## 3    4.587066    4.627900    4.830638    5.205629
## 4   -4.380847   -3.874975   -3.311615   -2.707039
##            83          84           85           86
## 1 -823.141872 -815.674235 -809.6706738 -802.77854967
## 2   21.342089   21.545955   22.0396188   22.67919319
## 3    5.682864    6.202823    6.7437290    7.28767654
## 4   -2.076160   -1.414323   -0.7095637    0.03951229
##             87          88          89          90
## 1 -796.4684204 -790.134455 -784.059612 -777.116461
## 2   23.3182815   23.938665   24.570001   25.306170
## 3    7.8027924    8.272484    8.716537    9.176228
## 4    0.8204479    1.615555    2.408225    3.181495
##            91          92          93          94
## 1 -772.251501 -765.477027 -759.744272 -754.005538
## 2   26.174152   26.950215   27.515951   27.800125
## 3    9.669953   10.165279   10.611421   10.982808
## 4    3.914596    4.592305    5.219478    5.816621
##            95          96          97          98
## 1 -747.938225 -742.455154 -737.129547 -731.968940
## 2   27.887709   28.094800   28.492279   28.953415
## 3   11.294281   11.583548   11.872180   12.155061
## 4    6.398433    6.963033    7.500092    8.002028
##            99         100         101         102
## 1 -726.980395 -722.170296 -717.544233 -713.107095
## 2   29.413159   29.859171   30.288331   30.698270
## 3   12.420172   12.661300   12.876715   13.066079
## 4    8.466445    8.893355    9.282759    9.634429
##           103        104        105        106
## 1 -708.862575 -704.81341 -700.96128 -697.30651
## 2   31.086791   31.45192   31.79198   32.10507
## 3   13.229534   13.36745   13.48018   13.56797
## 4    9.947941   10.22274   10.45730   10.64896
##          107        108        109
## 1 -693.84843 -690.58519 -687.51378
## 2   32.38464   32.60847   32.75375
## 3   13.63099   13.67141   13.69467
## 4   10.79606   10.90135   10.97255
##
## $male$aml
##
## Call:
```

```
## lm(formula = aml ~ cm + cml + cmls + cmlc)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##    -1.45473      4.36103     -0.07466      0.01962
##        cmlc
##     0.01066
##
##
## $male$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   9.683e-03    8.548e-03   -2.721e-03   -5.100e-04
##        cmlc           am         amls         amlc
##  -3.649e-05   -1.950e-03    9.816e-05   -1.404e-05
##      cmlaml
##  -3.438e-05
##
##
## $male$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##  -0.1703351    0.2778716   -0.0954026   -0.0179618
##        cmlc           am         amls         amlc
##  -0.0012921   -0.0096727    0.0024388    0.0006146
##      cmlaml
##  -0.0011772
##
##
## $male$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   -0.137819     0.378645    -0.101802    -0.022975
##        cmlc           am         amls         amlc
##   -0.001435    -0.108138     0.001963     0.001026
##      cmlaml
##    0.003612
##
##
```

```
## $male$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   0.8550781   -1.7611681    0.4939292    0.0936897
##        cmlc           am         amls         amlc
##   0.0060390    0.0516517    0.0036622    0.0001595
##      cmlaml
##   0.0024201
##
##
## $male$offset
## [1] 10
##
## $male$q0
##
## Call:
## lm(formula = as.numeric(ql[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml         cmls
##    -0.81911      0.69912     -0.03501
##
##
## $male$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/mods.RData",compress=TRUE,list=c("mods"))
# 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())
```

```r
# rm(list=ls())

# record when this stops
write(Sys.time(),file="../Rmd/stopped.txt")
# how long did this take?
started <- scan(file="../Rmd/started.txt")
stopped <- Sys.time()
# start, stop, duration
paste("Duration:",seconds_to_period(stopped-started))
```

```
## [1] "Duration: 6M 41.5753779411316S"
```