SVD Component Mortality Model

Reproducibility Materials: Data and Code

Samuel J. Clark

2015-2019

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

```
## [1] "R version 3.5.3 (2019-03-11)"
```

- RStudio version 1.2.1335
- Additional R packages from CRAN
    - knitr
    - bookdown
    - formatR
    - devtools
    - reshape2
    - ggplot2

  - plyr
  - stargazer
  - xtable
  - readr
  - stringr
  - httr

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

2 HMD Data

2.1 Download 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**. The file *getHMD.R* contains a set of functions that automate downloading and parsing all of the HMD life tables. This code is not commented, but you can get the general gist of what's going on by reading it. In a nutshell, the HMD .zip file is downloaded and unzipped (if the HMD web site isn't working, a pre-downloaded version of the file from 2018-11-02 is used), and then all life tables of a specified type are read into a big R list that is very fast and flexible. Functions are provided for extracting a specified column from all life tables in such a list and outputing the result as a matrix. The following chunk sources the functions for automating downloading and processing of the HMD life tables.

```
# source('../R/getHMD.R') # load the 'read HMD'
# functions

# uncommented code to download 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) {
            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 {
                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) {
                path <- "/lt_both/bltper_1x10"
                value <- list(sap.code = 9, data.path = path)
            } else {
```

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

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

    return(value)

}


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

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

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

}


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

    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[["download.date"]] <- download.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)

}


download.hmd <- function(output.file, unzip.dir, hmd.user,
    hmd.pass) {

    url <- "http://www.mortality.org/hmd/zip/all_hmd/hmd_statistics.zip"
    print("Downloading HMD ...")
    hmd.zip <- try(GET(url, authenticate(hmd.user, hmd.pass),
        write_disk(output.file, overwrite = TRUE), progress(),
        show.error.messages = FALSE))
    if (class(hmd.zip) == "try-error") {
        return(hmd.zip)
        stop(attributes(hmd.zip)$condition)
    } else if (http_status(hmd.zip)$category == "Client error") {
        if (http_status(hmd.zip)$message == "Client error: (401) Unauthorized") {
            return(hmd.zip)
            stop("Check user name and password and try again.")
        } else {
            return(hmd.zip)
            stop(http_status(hmd.zip)$message)
        }
    }
    unzip(zipfile = output.file, exdir = unzip.dir)
    return(hmd.zip)

}
```

The following chunk uses the download.hmd() function to downlaod the HMD life tables. To run you need to insert your own HMD user name and password. The data are unzipped into the *data/HMD/hmd_statistics* directory that needs to exist before downloading.

```
# try the download
download.result <- download.hmd(
  output.file="../data/HMD/hmd_statistics.zip",
  unzip.dir="../data/HMD/hmd_statistics",
#################################################################
##########################  IMPORTANT  ##########################
#################################################################
#   with following two lines commented, this script will use    #
#    archived HMD and replicate the article                     #
#   uncomment following two lines to download and analyse       #
```

```
#    current HMD                                                 #
##################################################################
  # hmd.user="sam@samclark.net",
  # hmd.pass="1241662754"
)
# if download.hmd() returns an error, then use the local cached copy of the HMD
if (class(download.result)=="try-error") {
  print(paste("Something went wrong -- usually this means the HMD site is not available.",
              " Using local cached version of HMD instead of a live download.",sep=""))
  unzip(zipfile="../data/HMD Archive/hmd_statistics.zip",exdir="../data/HMD/hmd_statistics")
}
```

The *download.result* object contains a description of what happened when the download was requested. You can have a look with this code.

```
if (class(download.result) != "try-error") {
    names(download.result)
    download.result
    download.result$date
    download.result$times
    download.result$headers
} else {
    print("Download did not happen, using local cached version of HMD.")
}
```

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.

```
# set the download date if the download was successful
if (class(download.result) != "try-error") {
    download.date <- download.result$headers$date
} else {
    download.date <- "Local cached HMD"
}

# 1-year age x 1-year period life tables
hmd.1x1.list <- list.lts("../data/HMD/hmd_statistics", 1,
    1, download.date)
# 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, download.date)
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.date','download.result'))
```

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

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

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

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

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

```
## List of 8
##  $ creation.date: chr "Fri Apr 12 19:31:50 2019"
##  $ download.date: chr "Local cached HMD"
##  $ female       :List of 49
##  $ male         :List of 49
##  $ both         :List of 49
##  $ age          : num 5
##  $ age.groups   : num 24
##  $ period       : num 1
```

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

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

```
## List of 8
##  $ creation.date: chr "Fri Apr 12 19:31:50 2019"
##  $ download.date: chr "Local cached HMD"
##  $ female       :List of 49
##   ..$ AUS    :List of 94
##   .. ..$ P1921:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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 71
##   .. ..$ P1947:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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 175
##   .. ..$ P1841:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1842:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1843:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1844:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. .. [list output truncated]
##   ..$ BGR    :List of 64
##   .. ..$ P1947:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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 49
##   ..$ AUS    :List of 94
##   .. ..$ P1921:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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 71
##   .. ..$ P1947:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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 175
##   .. ..$ P1841:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1842:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1843:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. ..$ P1844:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   .. .. ..- attr(*, "problems")=Classes 'tbl_df', 'tbl' and 'data.frame':    960 obs. of  5 variable
##   .. .. .. ..$ row     : int [1:960] 1753 1753 1753 1753 1753 ...
##   .. .. .. ..$ col     : chr [1:960] "mx" "qx" ...
##   .. .. .. ..$ expected: chr [1:960] "a number" "a number" ...
##   .. .. .. ..$ actual  : chr [1:960] "." "." ...
##   .. .. .. .. [list output truncated]
##   .. .. [list output truncated]
```

```
##   ..$ BGR    :List of 64
##   .. ..$ P1947:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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:Classes 'tbl_df', 'tbl' and 'data.frame': 24 obs. of  10 variables:
##   .. .. ..$ 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]
##   [list output truncated]
```

Have look at the full structure of one life table.

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

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

```r
# or equivalently
str(hmd.5x1.list$female$AUS$P1921)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    24 obs. of  10 variables:
##  $ period: chr  "1921" "1921" "1921" "1921" ...
##  $ age   : chr  "0" "1-4" "5-9" "10-14" ...
##  $ mx    : num  0.05999 0.00602 0.00192 0.00128 0.00205 ...
##  $ qx    : num  0.0575 0.0237 0.00952 0.00637 0.0102 ...
##  $ ax    : num  0.28 1.38 2.14 2.6 2.68 2.57 2.57 2.64 2.62 2.55 ...
##  $ lx    : num  100000 94250 92016 91140 90559 ...
##  $ dx    : num  5750 2234 876 580 924 ...
```

```
##  $ Lx    : num  95857 371152 457576 454303 450651 ...
##  $ Tx    : num  6317561 6221704 5850553 5392977 4938674 ...
##  $ ex    : num  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.

```
# 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
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”.

```
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.

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             239             240             241
## female.BEL.1917 female.BEL.1918
##             242             243
```

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

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

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             239             240             241
## female.BEL.1917 female.BEL.1918
##             242             243
```

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

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

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             239             240             241
## female.BEL.1917 female.BEL.1918
##             242             243
```

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

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

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

```
## female.BEL.1914 female.BEL.1915 female.BEL.1916
##             239             240             241
## female.BEL.1917 female.BEL.1918
##             242             243
```

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

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

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           239           240           241
## male.BEL.1917 male.BEL.1918
##           242           243
```

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

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

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           239           240           241
## male.BEL.1917 male.BEL.1918
##           242           243
```

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

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

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           239           240           241
## male.BEL.1917 male.BEL.1918
```

```
##            242           243
```

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

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

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

```
## male.BEL.1914 male.BEL.1915 male.BEL.1916
##           239           240           241
## male.BEL.1917 male.BEL.1918
##           242           243
```

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

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

```
# in all matrices, empty columns are THE SAME, numbered
# 236-340
remove <- unique(c(which(is.na(q1.f[1, ])), which(is.na(q1.m[1,
    ]))))

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
## [1]  110 4614
```

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

```
## [1]  110 4614
```

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

```
## [1]   23 4614
```

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

```
## [1]   23 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]   24 4614
```

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

```
## [1]   24 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]   24 4614
```

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

```
## [1]   24 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]  111 4614
```

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

```
## [1]   24 4614
```

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

```
## [1]   24 4614
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
rm(list = c("remove"))
```

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
##               2709               3643
## female.NZL_MA.1956 female.NZL_MA.1959
##               3650               3653
```

```
# 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
##               2709               3643
```

```
## female.NZL_MA.1956 female.NZL_MA.1959
##               3650               3653
```

```
# 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)
```

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

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

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

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

```
e1.f <- e1.f[, -remove.1]
e1.m <- e1.m[, -remove.1]

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

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

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

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

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

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

```
## [1]  110 4610
```

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

```
## [1]  110 4610
```

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

```
## [1]   23 4610
```

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

```
## [1]   23 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]   24 4610
```

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

```
## [1]   24 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]   24 4610
```

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

```
## [1]   24 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]  111 4610
```

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

```
## [1]   24 4610
```

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

```
## [1]   24 4610
```

```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
```

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

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

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

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

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

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

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

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

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

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

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

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

```
rm(list = c("remove.1", "remove.5"))
```

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

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

```
## [1] 507100
```

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

```
## [1] 2604
```

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

```
## [1] 106030
```

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

```
## [1] 75
```

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

```
## [1] 507100
```

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

```
## [1] 1380
```

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

```
## [1] 106030
```

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

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

```
# female
q1.f.nz <- q1.f
q1.f.nz[q1.f.nz == 0] <- 1e-06
q5.f.nz <- q5.f
q5.f.nz[q5.f.nz == 0] <- 1e-06

# male
q1.m.nz <- q1.m
q1.m.nz[q1.m.nz == 0] <- 1e-06
q5.m.nz <- q5.m
q5.m.nz[q5.m.nz == 0] <- 1e-06

cat("\n")
```

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

```
## [1] 507100
```

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

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

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

```
## [1] 106030
```

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

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

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

```
## [1] 507100
```

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

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

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

```
## [1] 106030
```

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

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

Take the log and logit transforms of the ${}_nq_x$ values.

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

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

```r
# 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)

# 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 4610
```

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

```
## [1]  110 4610
```

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

```
## [1]   23 4610
```

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

```
## [1]   23 4610
```

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

```
## [1]  110 4610
```

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

```
## [1]  110 4610
```

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

```
## [1]   23 4610
```

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

```
## [1]   23 4610
```

2.3 Additional Indicator Calculation

We need child, $_5q_0$, and adult, $_45q_15$, mortality values for females and males. Calculate these from the 1 × 1 $_nq_x$ values and store in separate matrices, including the log and logit transformed values.

```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)
}

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

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

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

# now actually create the child and adult mortality
# indicators

# female

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

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

```
# log and logit
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)
```

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

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

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

3 SVD Component Model of Mortality

3.1 *svdMod()* function

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

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

Inputs to the function:

- 'ql' are the logit-transformed input age-specific ${}_nq_x$ (life tables) arranged as age×lifetable
- 'Ql' are the logit-transformed summary mortality indicators: child mortality, ${}_5q_0$, and adult mortality, ${}_45q_{15}$, arranged in $2 \times n$ form where the first row is child mortality, the second adult mortality, and the columns correspond to life tables
- 'N' is the number of times to repeat sampling/validation
- 'S' is the fraction of life tables to include in the sample
- 'offset' is a number used to offset the the age-specific mortality rates from the origin before calculating the SVD; this is an easy way to give each age group approximately the same weight in the SVD calculation; it is added back when predictions are made
- 'retAll' is a switch indicating if all results should be returned or just summaries
- 'adult' is a switch indicating if adult mortality, ${}_45q_{15}$, is supplied and should be used directly as an input to the model when predictions are made; if not, then child mortality, ${}_5q_0$, is the only direct input, and adult mortality is used *indirectly* by predicting it from child mortality and then using it together with child mortality for the predictions for other ages
- 'q0Fix' is a switch indicating if the *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

```r
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
            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

    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)
}
```

```r
# 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: 2.04072999954224"
```

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

```
## [1] "Apply way: 1.88316297531128"
```

```
# 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) {

  # 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
```

```r
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
```

```r
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"
```

```r
# 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"
```

```r
# have a look
cat("\n")
```

```r
comps.child$female
```

```
##      total.abs.error mean.abs.error max.error
## comp        1445.741     0.01363521 0.3306283
## lq          1501.606     0.01416209 0.3968440
```

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

```
##      total.abs.error mean.abs.error max.error
## comp        1674.491     0.01579262 0.3773859
## lq          1777.257     0.01676183 0.3792040
```

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

```
comps.adult$female
```

```
##      total.abs.error mean.abs.error max.error
## comp        1297.786     0.01223980 0.2204735
## lq          1399.336     0.01319754 0.3865020
```

```
comps.adult$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1378.421     0.01300029 0.3866729
## lq          1472.245     0.01388518 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)
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,
        Q1.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]
}

# 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.

```
# age relationships

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

```
## pdf
##   2
```

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

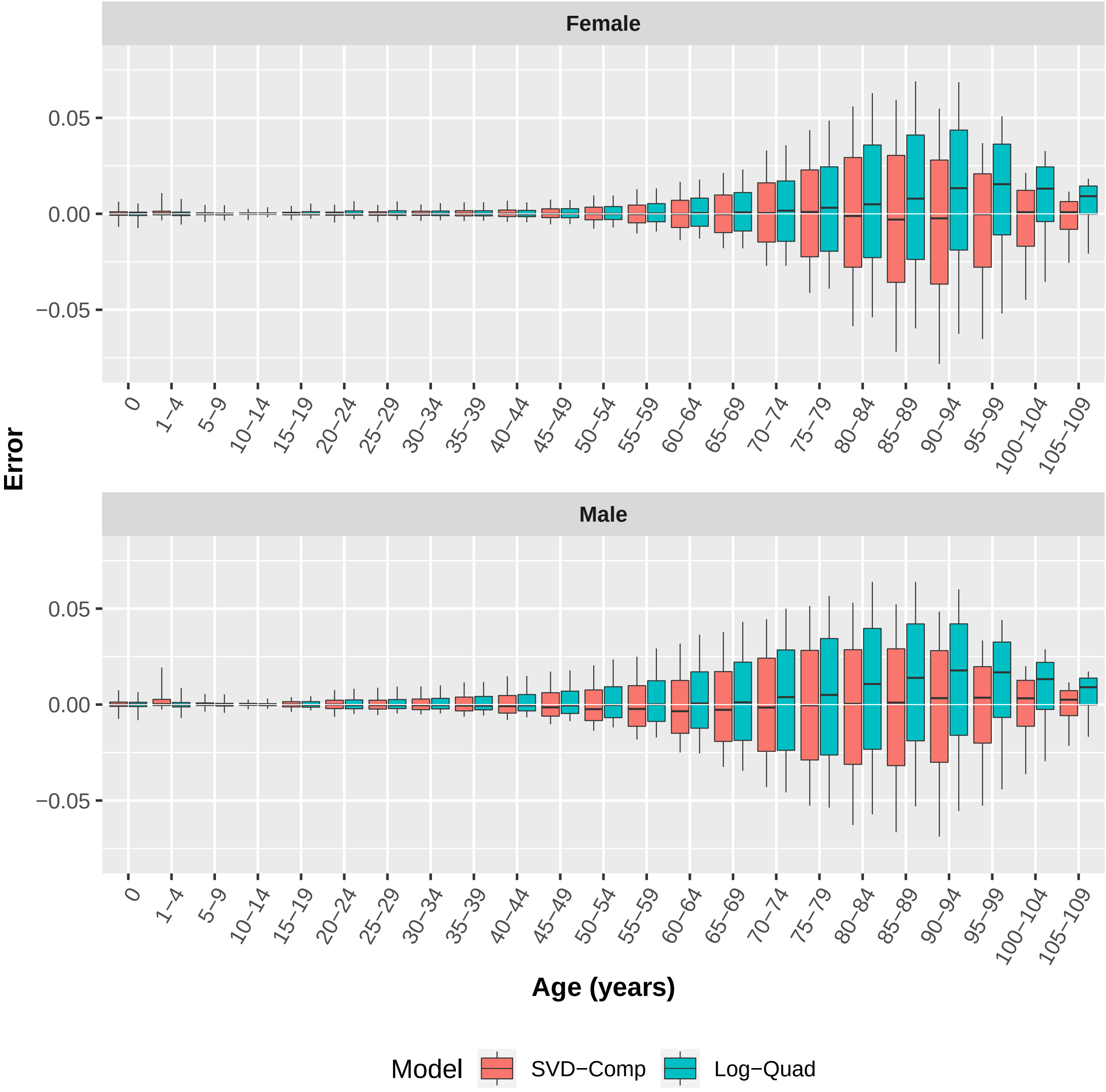
```
## pdf
##   2
```

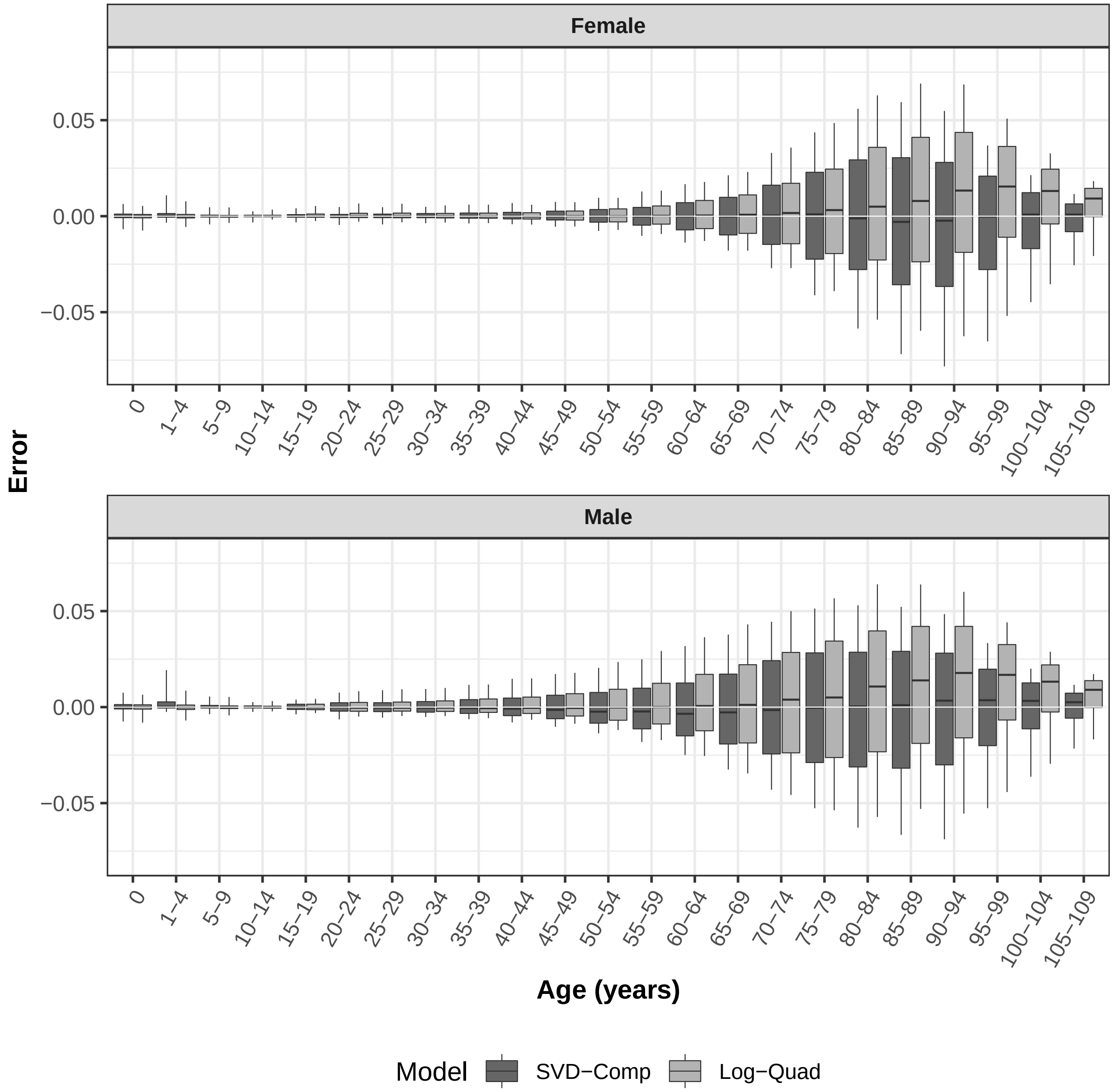
```
rm(list = c("nr", "i", "j"))
```

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

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

# reshape the data

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

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

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

e <- rbind(efn, emn)

e.sum <- ddply(e, .(Sex, `Age (years)`, Model), summarize,
    ymin = quantile(Error, 0.1), ymax = quantile(Error,
        0.9), 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 = 0.2) +
    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 = 0.2) + 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 = 0.7) + ggsave("../figures/fig3-BW.pdf",
    width = 6.5, height = 6.5, units = c("in"))

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


Female
Male
Error
0.05
0.00
−0.05
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
85−89
90−94
95−99
100−104
105−109
Age (years)
Model
SVD−Comp
Log−Quad



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.DEUTE.2006
##               824
```

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

```
## male.DNK.2010
##          1095
```

```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] 4161
```

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

```
## [1] 4161
```

```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.2012
## female.SVN.2013
## female.SVN.2014
## 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] 138
```

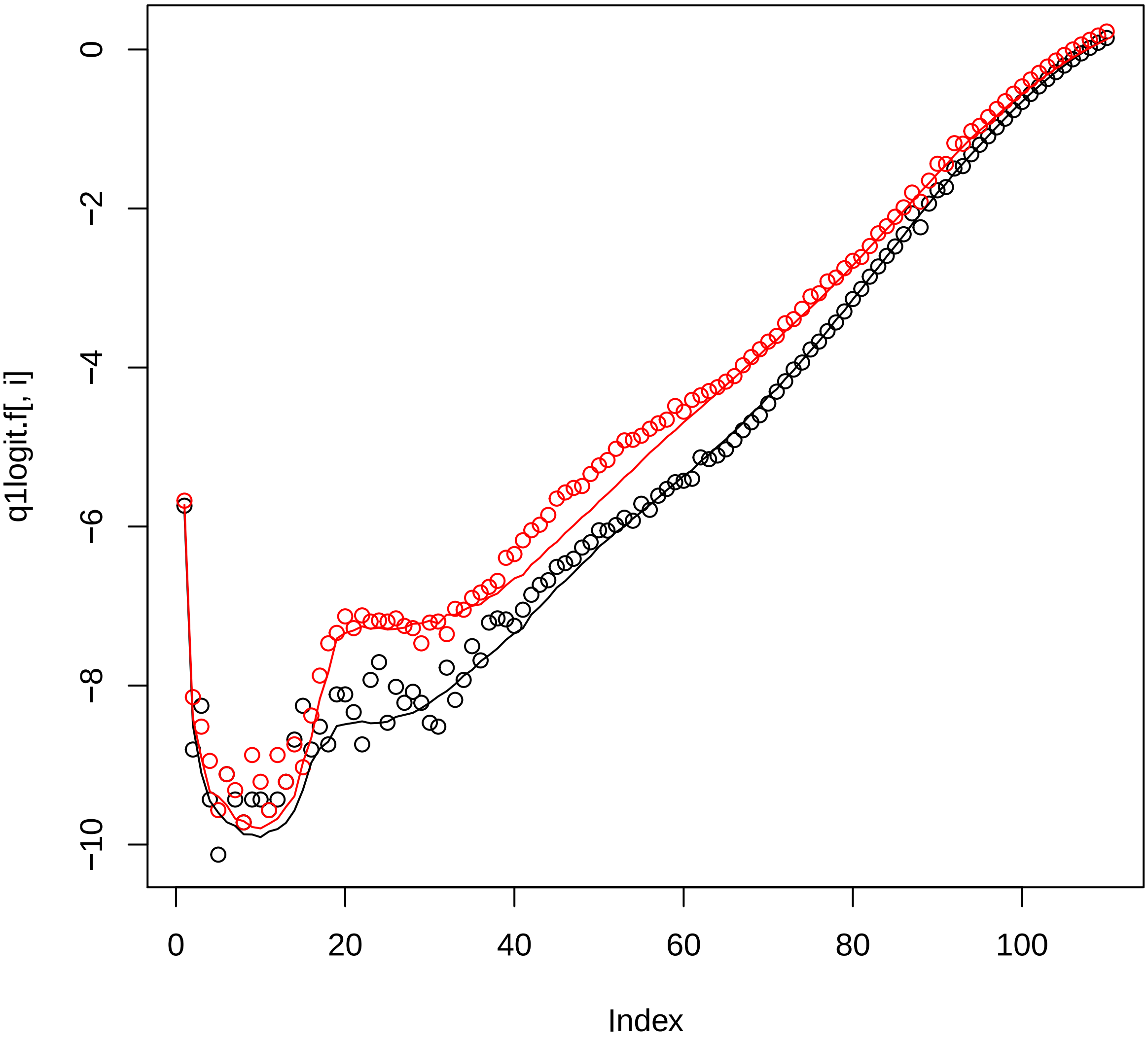
```r
aut.m.1990 <- which(colnames(q1.m) == "male.AUT.1990")
aut.m.1990
```

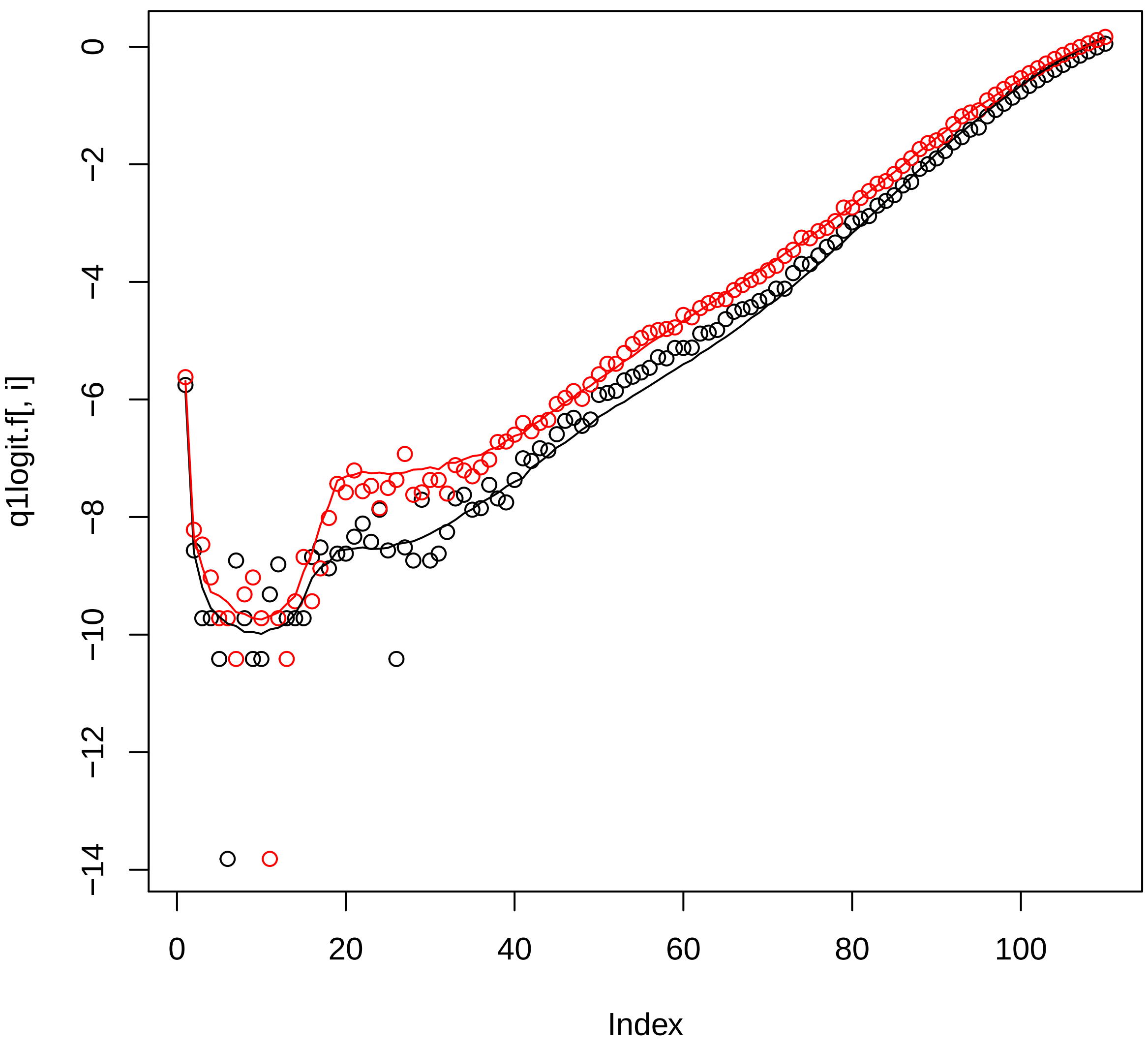
```
## [1] 138
```

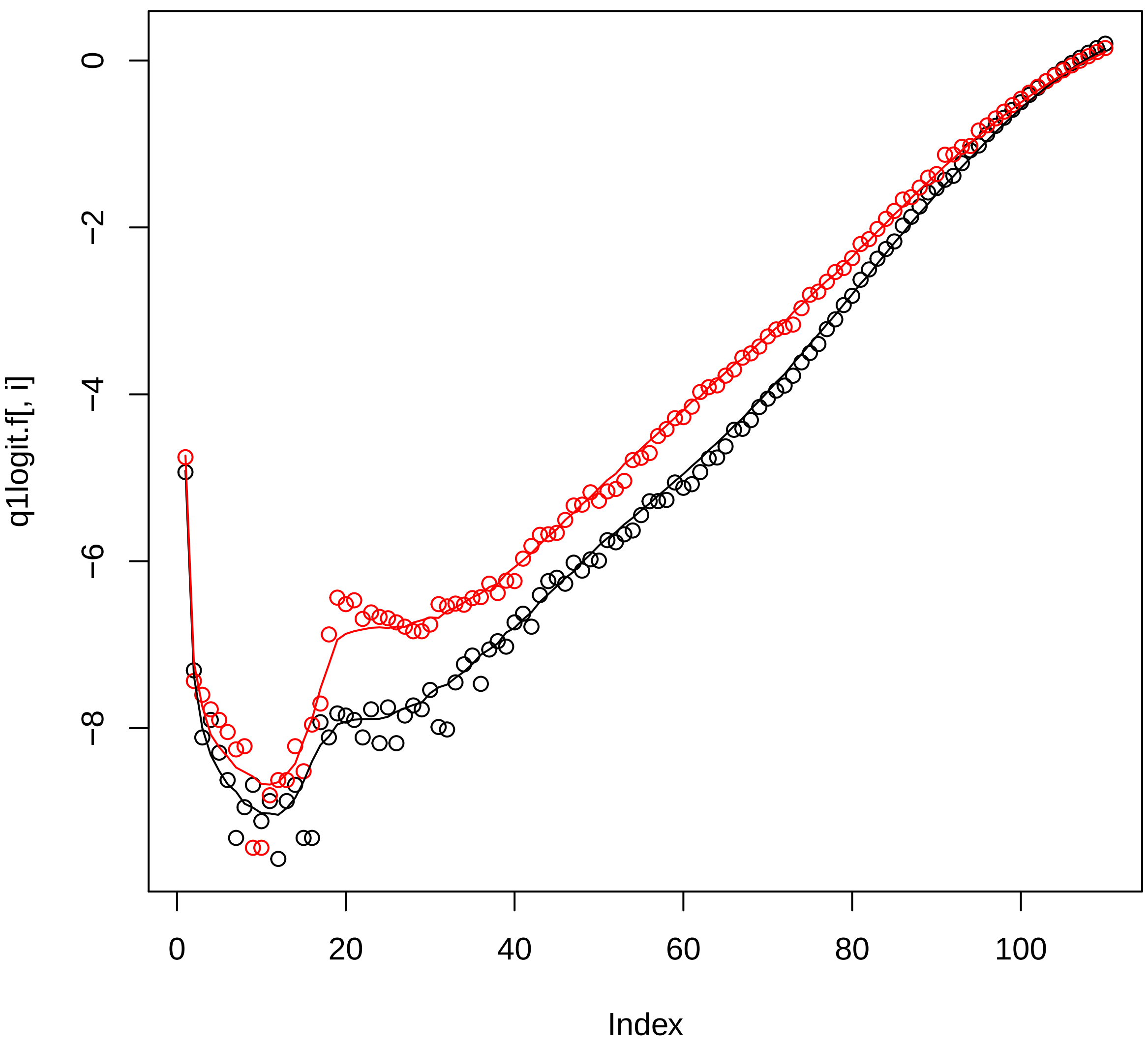
```r
i <- aut.f.1990
plot(q1logit.f[, i], )
points(q1logit.m[, i], col = "red")
```

```r
points(mod.1_0.f$recon.samp$s1[, i], type = "l")
points(mod.1_0.m$recon.samp$s1[, i], type = "l", col = "red")

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

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

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

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

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

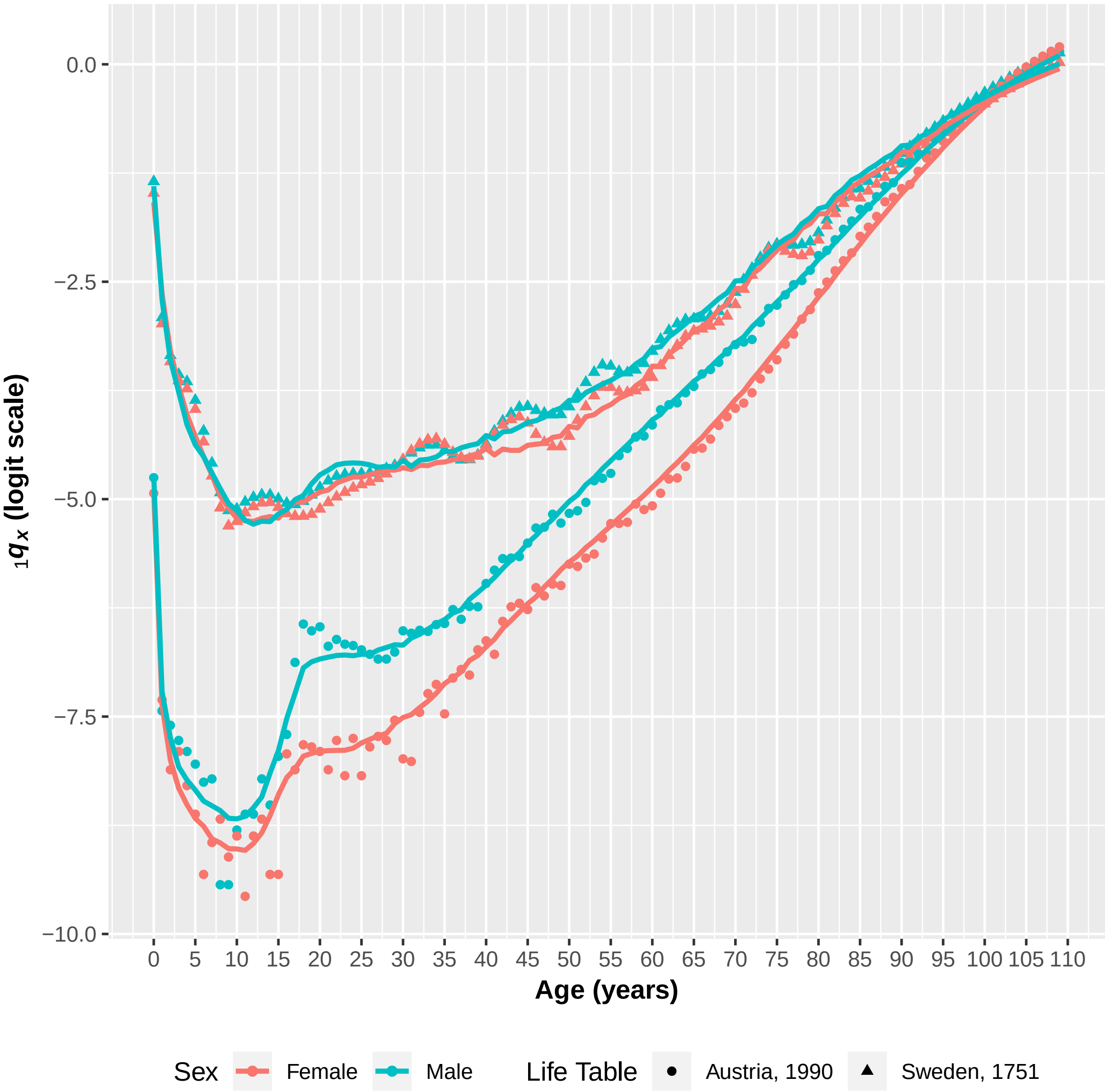
```
# grayscale
ggplot(data = data.fig1.df, aes(x = Age, y = Value, group = interaction(Sex,
    LT), colour = Sex, shape = LT)) + geom_point(size = 1.5) +
    geom_line(data = pred.fig1.df, aes(x = Age, y = Value,
        group = interaction(Sex, LT), colour = Sex), size = 1) +
    scale_x_continuous(breaks = seq(0, 110, 5)) + labs(y = expression(""[bold(1)] *
    bolditalic("q")[bolditalic(x)] * bold(" (logit scale)")),
    x = expression(bold("Age (years)"))) + # theme(legend.justification=c(1,0),
# legend.position=c(0.98,0.02)) +
theme_bw() + theme(legend.position = "bottom", legend.box = "horizontal") +
    scale_shape_discrete(name = "Life Table") + scale_colour_grey(start = 0,
    end = 0.7) + 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"))
```



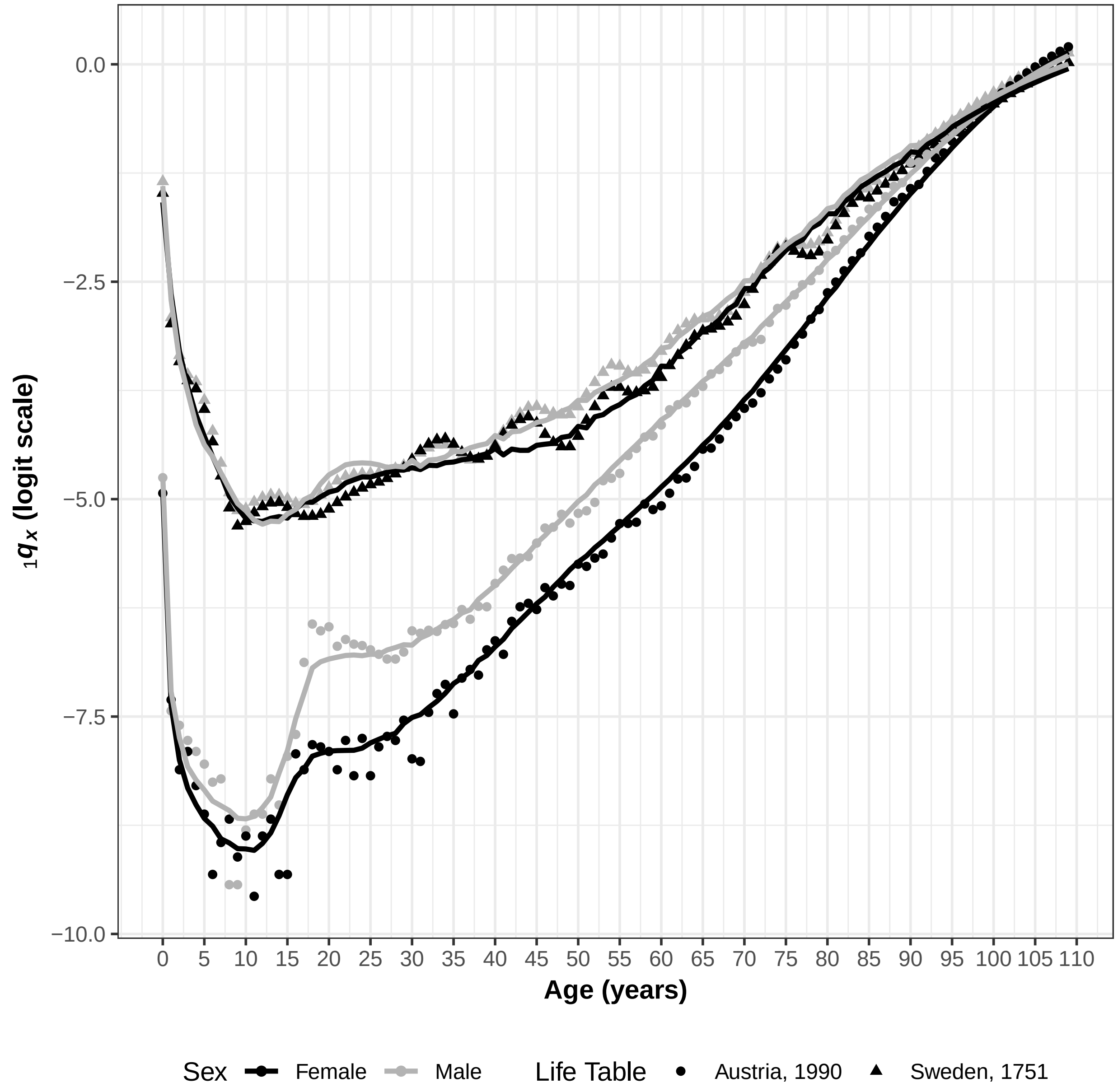

0
−2
−4
−6
−8
−10
−12
−14
q1logit.f[, i]
0
20
40
60
80
100
Index



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



0.0
−2.5
−5.0
−7.5
−10.0
${}_1q_x$ (logit scale)
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

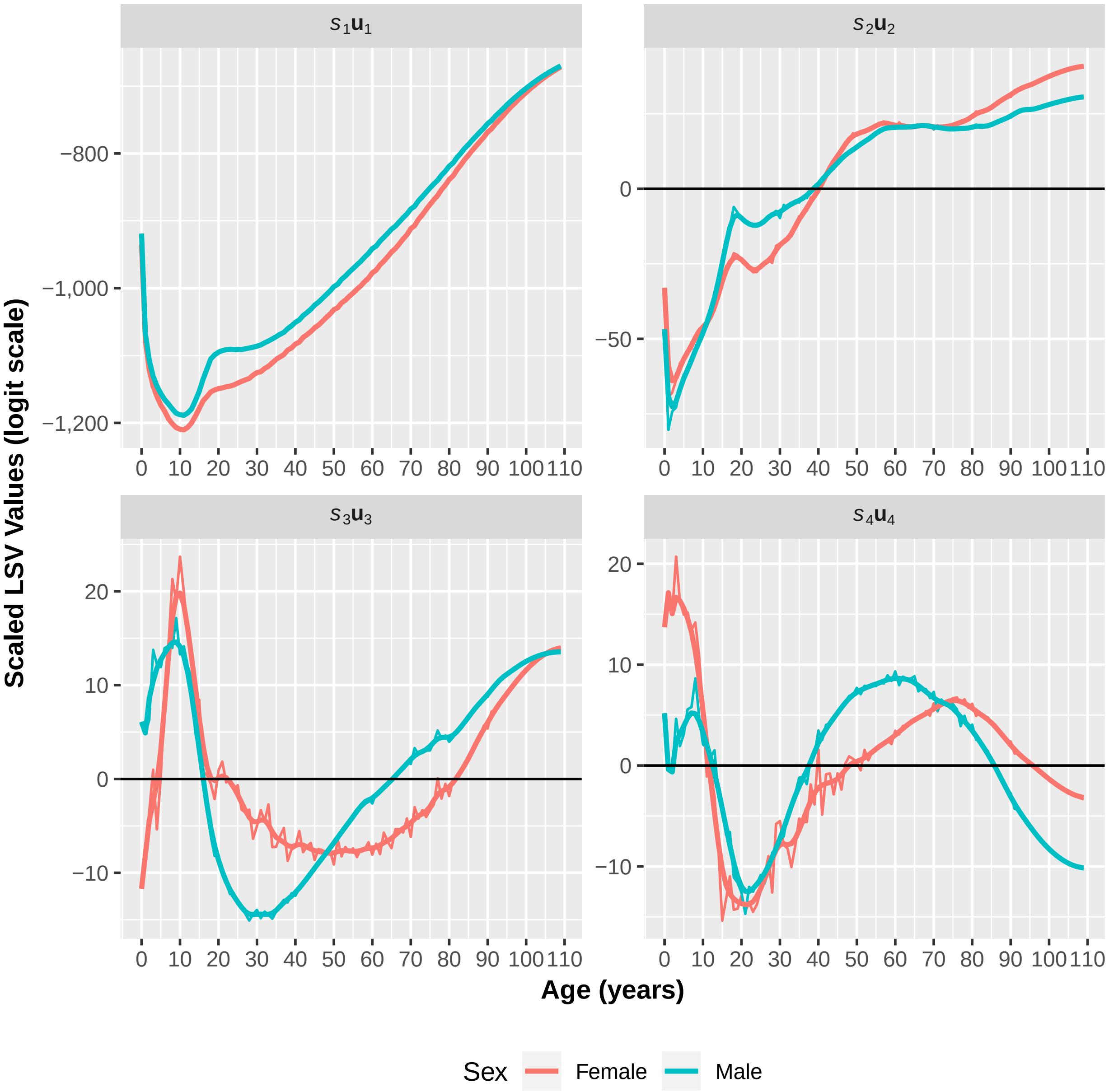


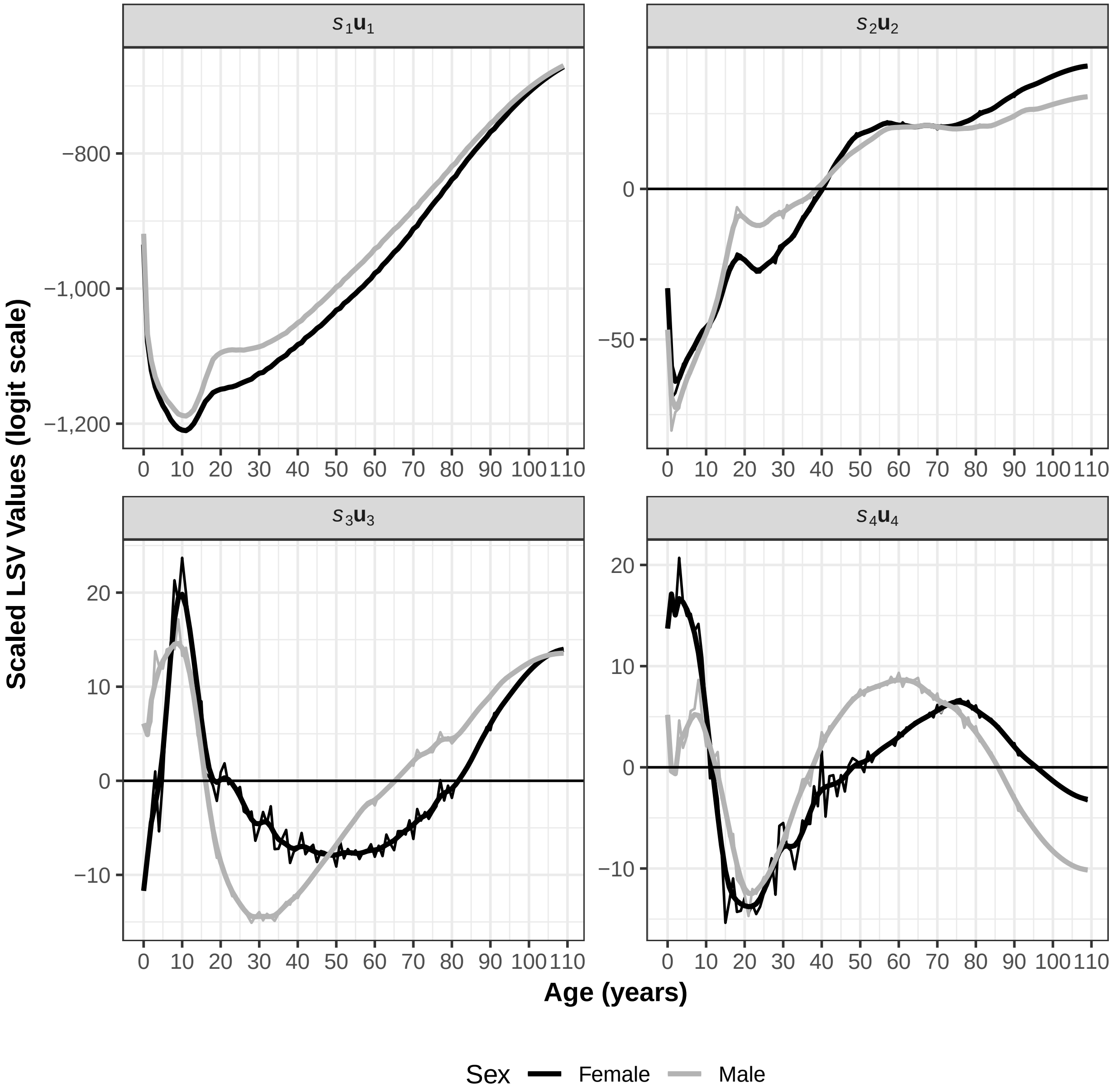
Plot the scaled left singular vectors of the SVD decompositions of logit-transformed $_1q_x$.

```
# 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])
```

```r
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 = 0.7) + 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"))
```


$s_1\mathbf{u}_1$
$s_2\mathbf{u}_2$
$s_3\mathbf{u}_3$
$s_4\mathbf{u}_4$
−800
−1,000
−1,200
0
−50
20
10
0
−10
20
10
0
−10
0 10 20 30 40 50 60 70 80 90 100 110
Scaled LSV Values (logit scale)
Age (years)
Sex
Female
Male

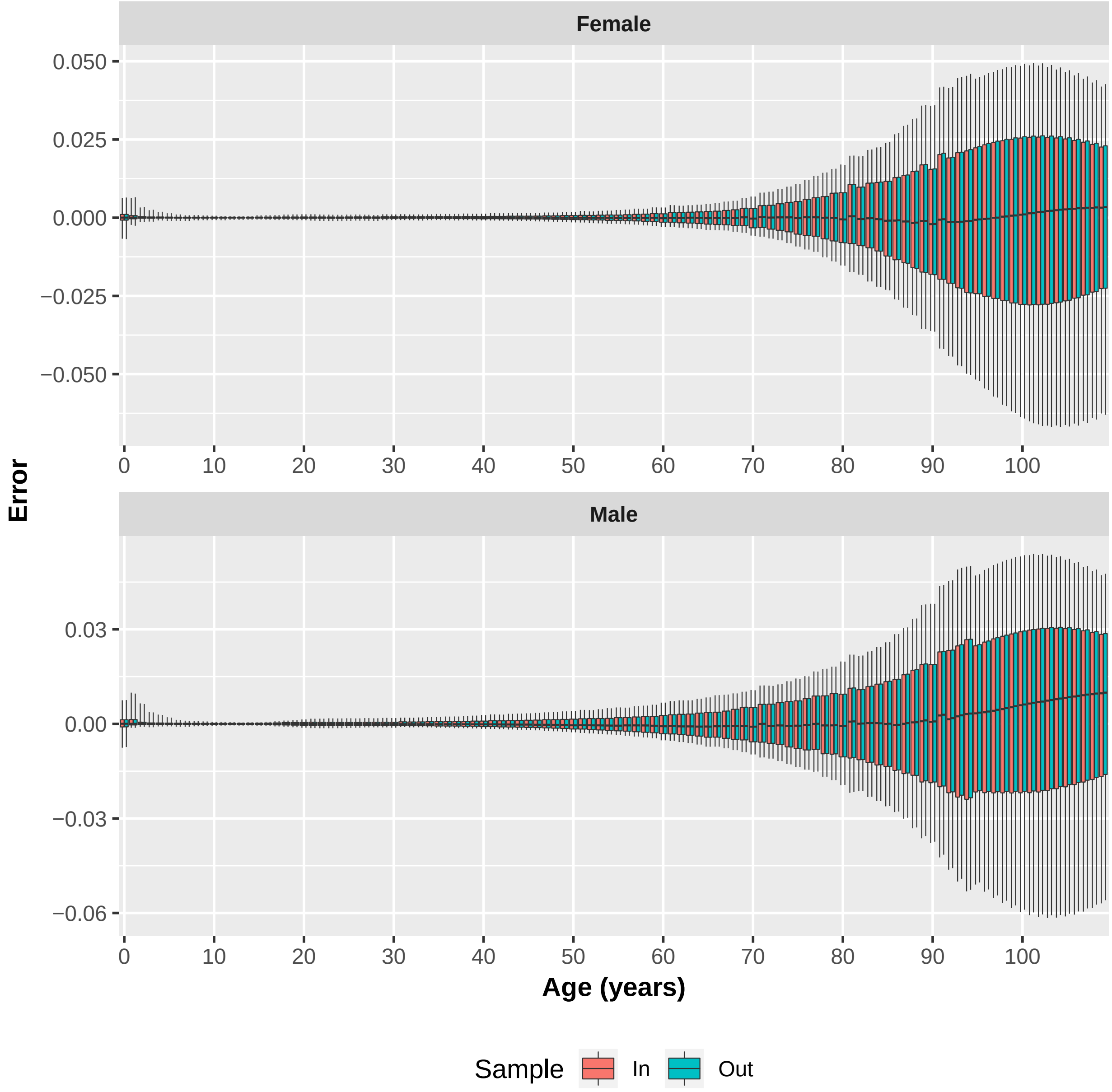


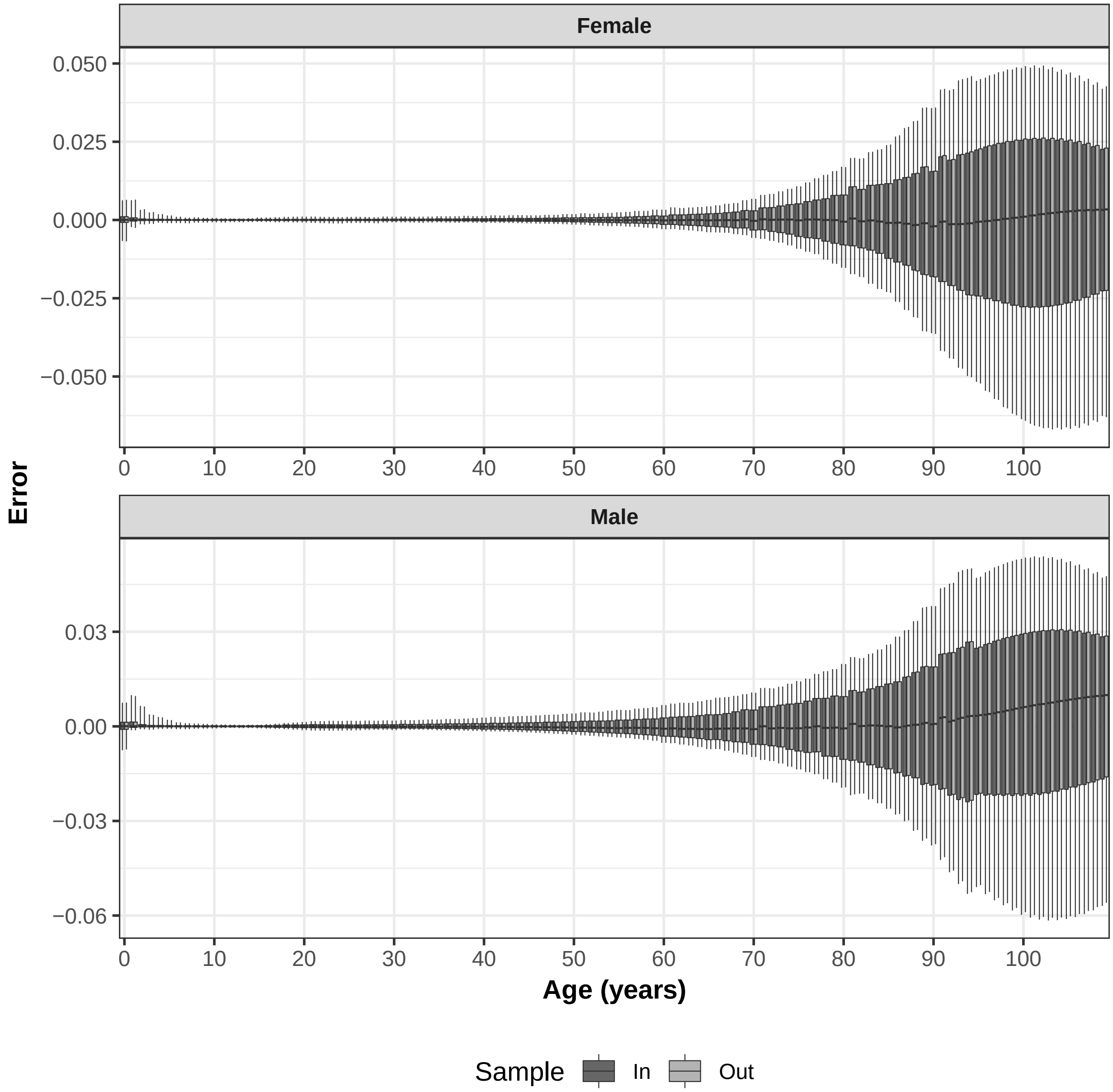
Plot the single-year prediction error distributions from 50 50% samples.

```
# 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,
    0.1), ymax = quantile(Error, 0.9), 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 = 0.7) + ggsave("../figures/fig4-BW.pdf",
    width = 6.5, height = 6.5, units = c("in"))

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





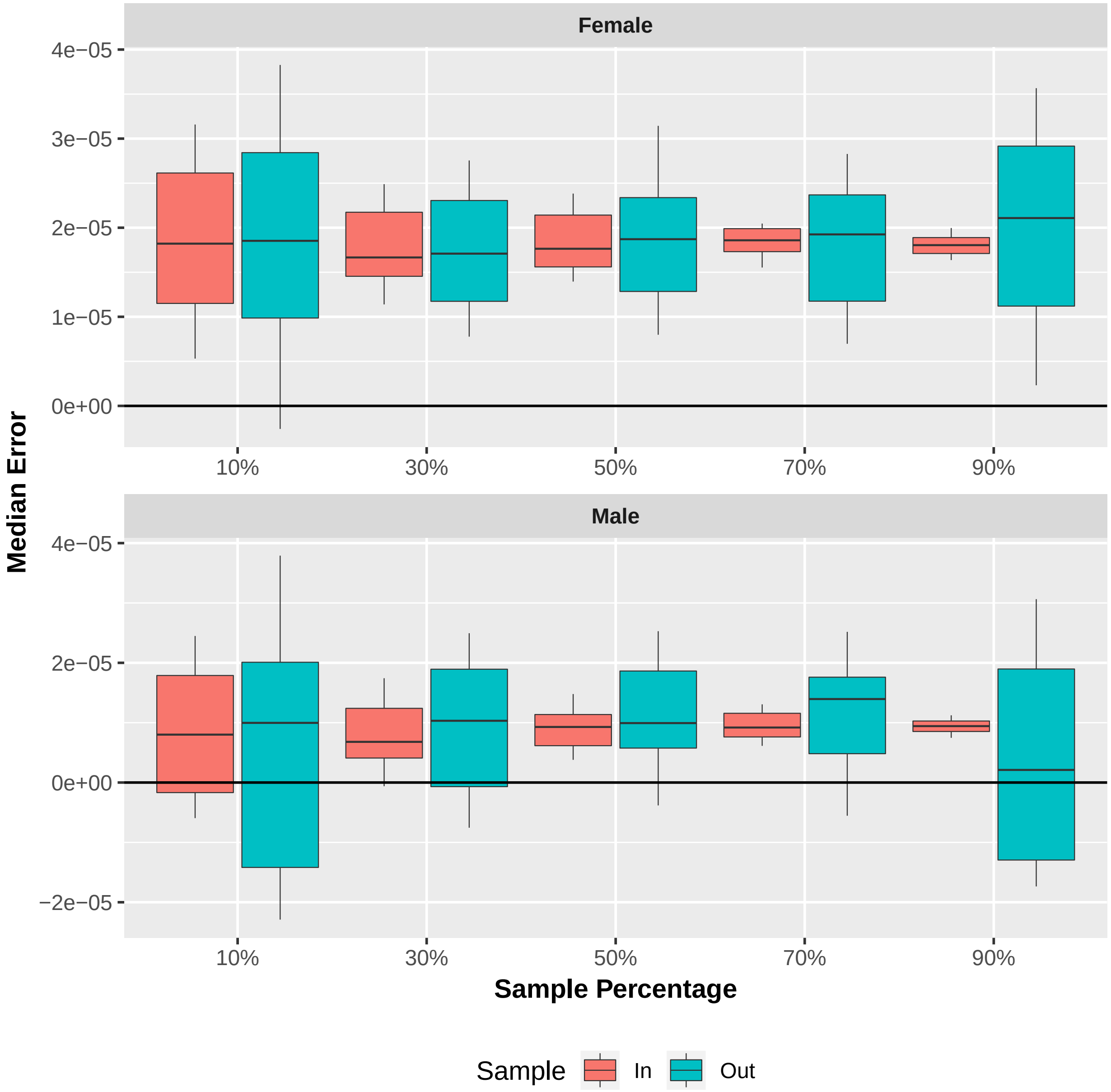
Plot the prediction error distributions by sample fraction for 50 50% samples

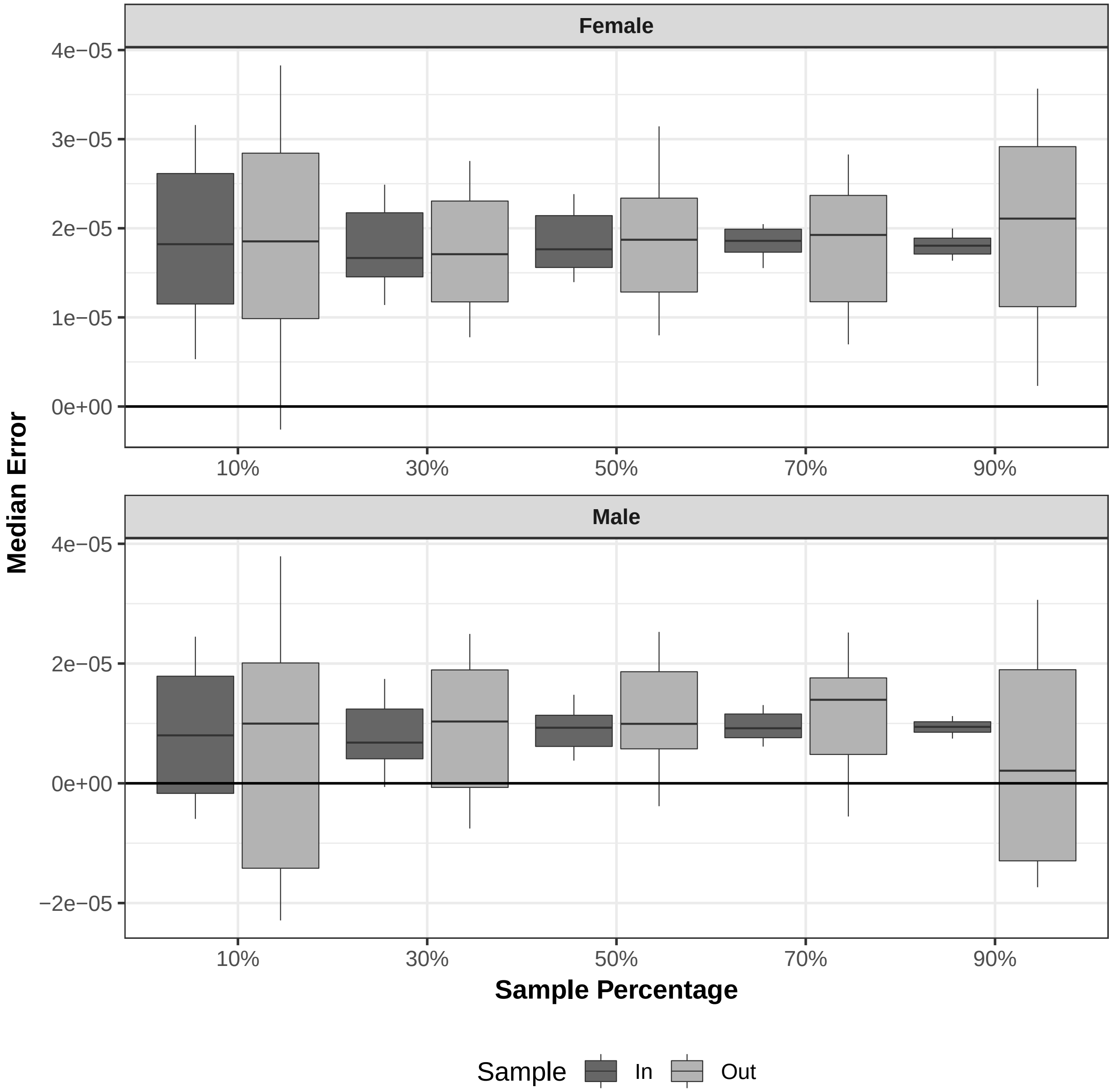
```
# medians female
es.s.f <- rbind(cbind("Female", "In", "10%", errsum.meds.1.f[,
    1]), cbind("Female", "In", "30%", errsum.meds.3.f[,
    1]), cbind("Female", "In", "50%", errsum.meds.5.f[,
    1]), cbind("Female", "In", "70%", errsum.meds.7.f[,
    1]), cbind("Female", "In", "90%", errsum.meds.9.f[,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

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

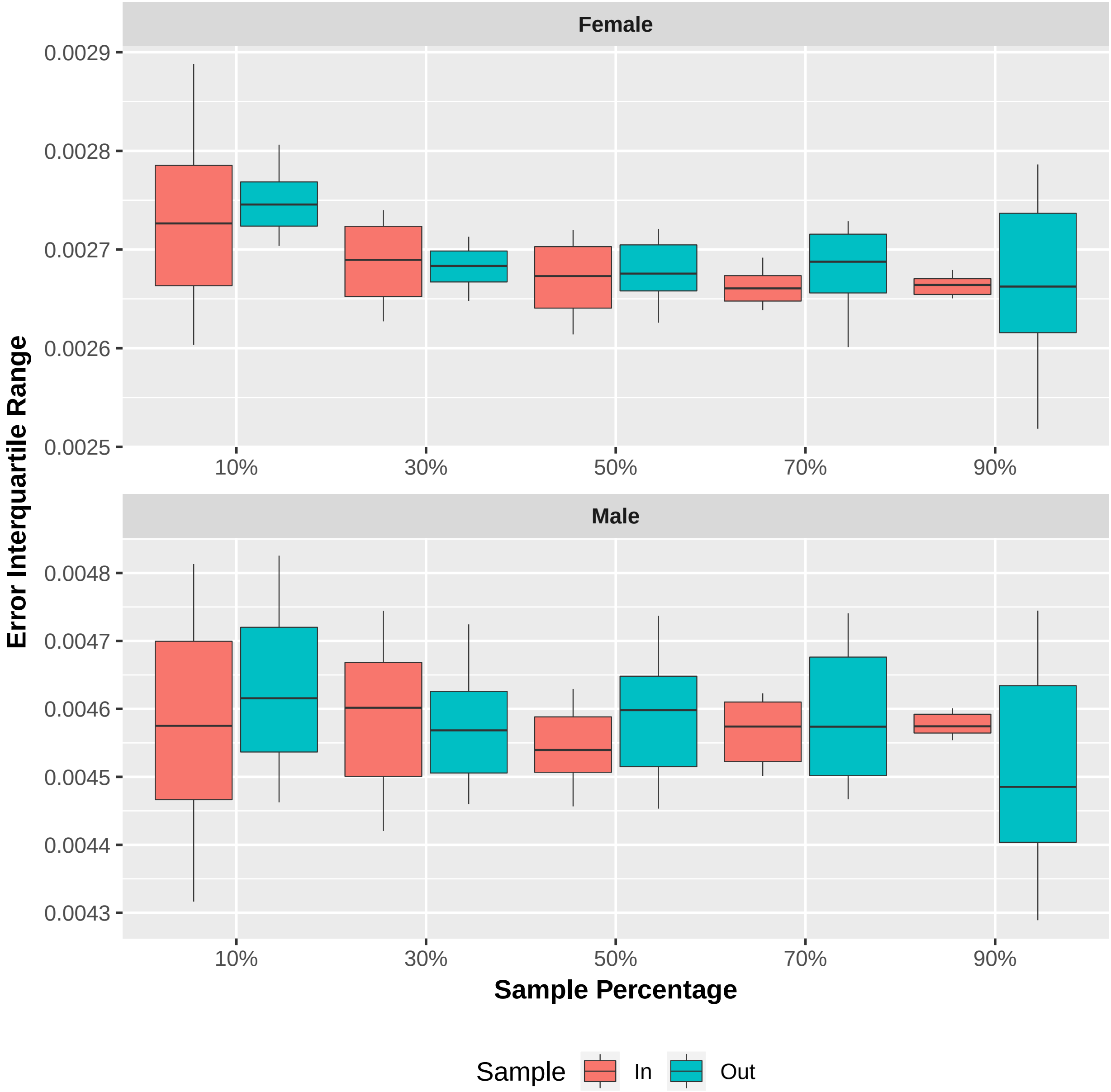
```
es.f <- rbind(es.s.f, es.ns.f)

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

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

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

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

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

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

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

e.sum <- ddply(es.df, .(Sex, Sample, Fraction), summarize,
    ymin = quantile(Median, 0.1), ymax = quantile(Median,
        0.9), middle = median(Median), lower = quantile(Median,
        0.25), upper = quantile(Median, 0.75))

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

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

# Plot
ggplot(data = e.sum, aes(x = Fraction)) + geom_boxplot(aes(fill = Sample,
    ymin = ymin, ymax = ymax, middle = middle, upper = upper,
    lower = lower), stat = "identity", size = 0.2) + geom_hline(yintercept = 0) +
    # scale_y_continuous(limits = c(-2e-05,4.5e-05)) +
# theme(legend.justification=c(1,0),
# legend.position=c(0.85,.56)) +
```

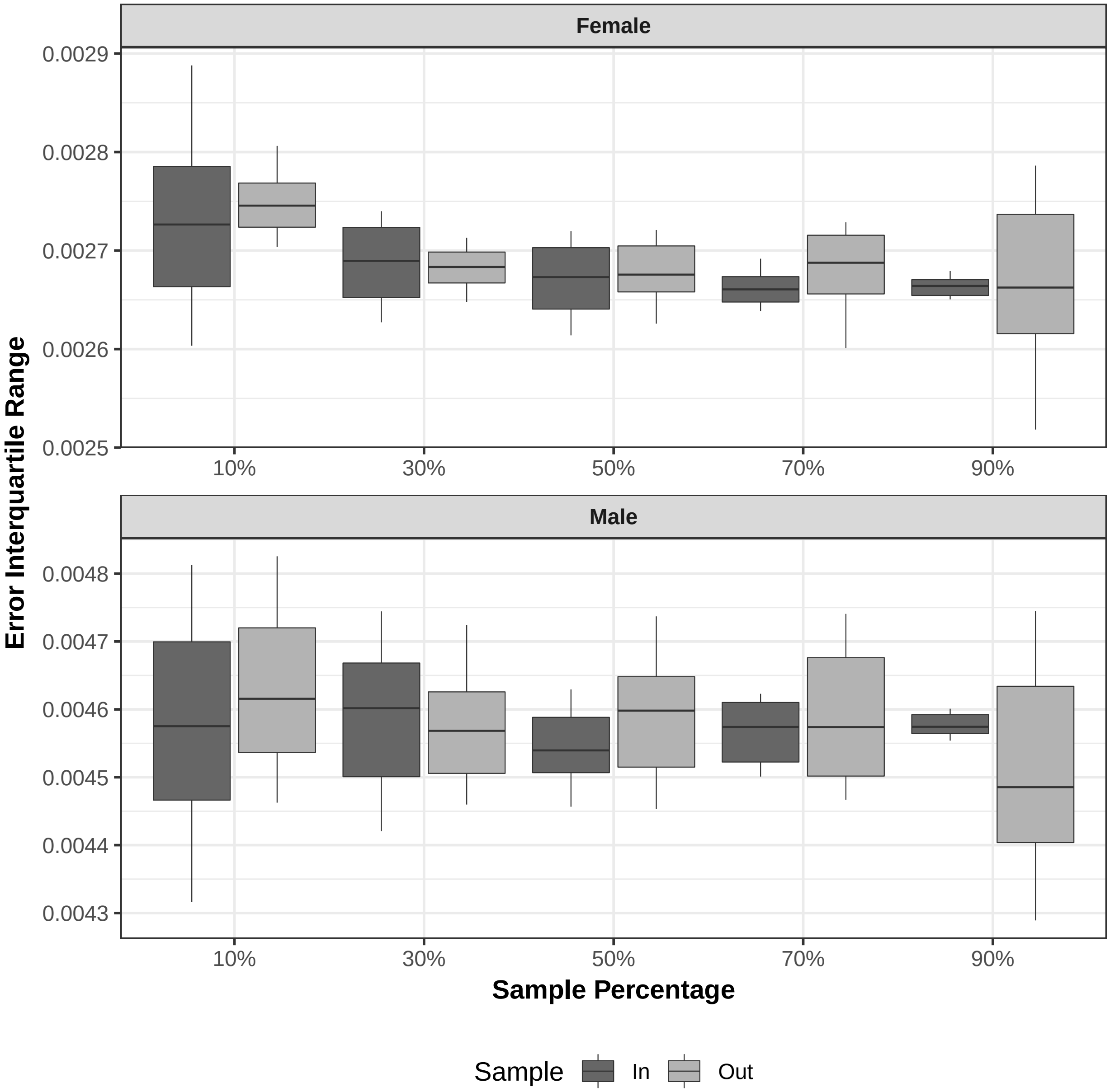
```r
theme(legend.position = "bottom", legend.box = "horizontal") +
    labs(y = expression(bold("Median Error")), x = expression(bold("Sample Percentage"))) +
    facet_wrap(~Sex, ncol = 1, scales = "free", labeller = s.labeller) +
    ggsave("../figures/fig5a.pdf", width = 6.5, height = 6.5,
        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 = 0.7) + ggsave("../figures/fig5a-BW.pdf",
    width = 6.5, height = 6.5, units = c("in"))


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

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

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

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

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

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

```
        "Out", "90%", errsum.iqrs.9.m[, 2]))

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

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

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

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

e.sum <- ddply(es.df, .(Sex, Sample, Fraction), summarize,
    ymin = quantile(Median, 0.1), ymax = quantile(Median,
        0.9), 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(li
# 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(li
# 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 = 0.7) + 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
4e−05
3e−05
2e−05
1e−05
0e+00
10%
30%
50%
70%
90%
Male
4e−05
2e−05
0e+00
−2e−05
10%
30%
50%
70%
90%
Median Error
Sample Percentage
Sample
In
Out


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


Female
Male
Error Interquartile Range
0.0029
0.0028
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

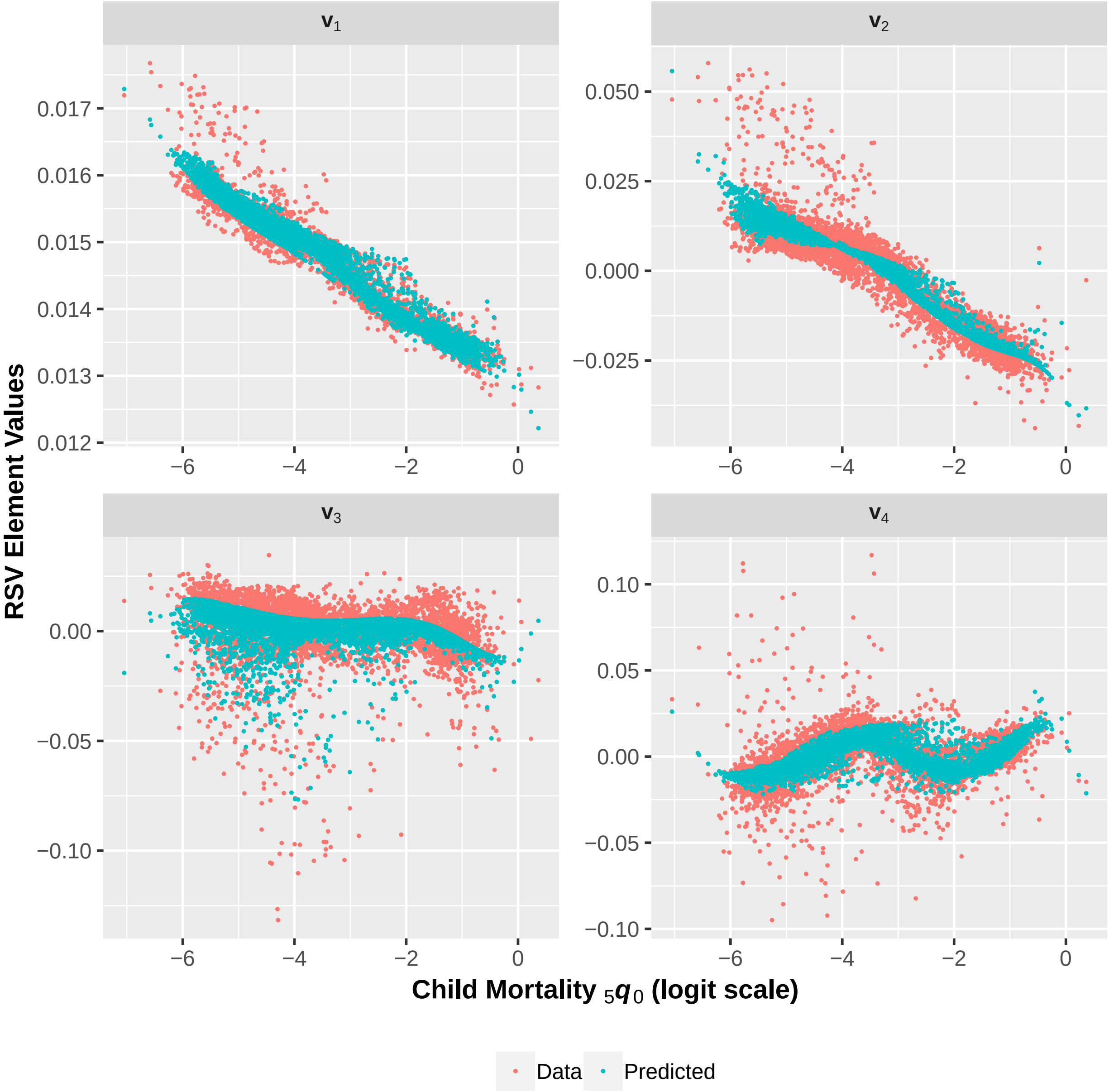


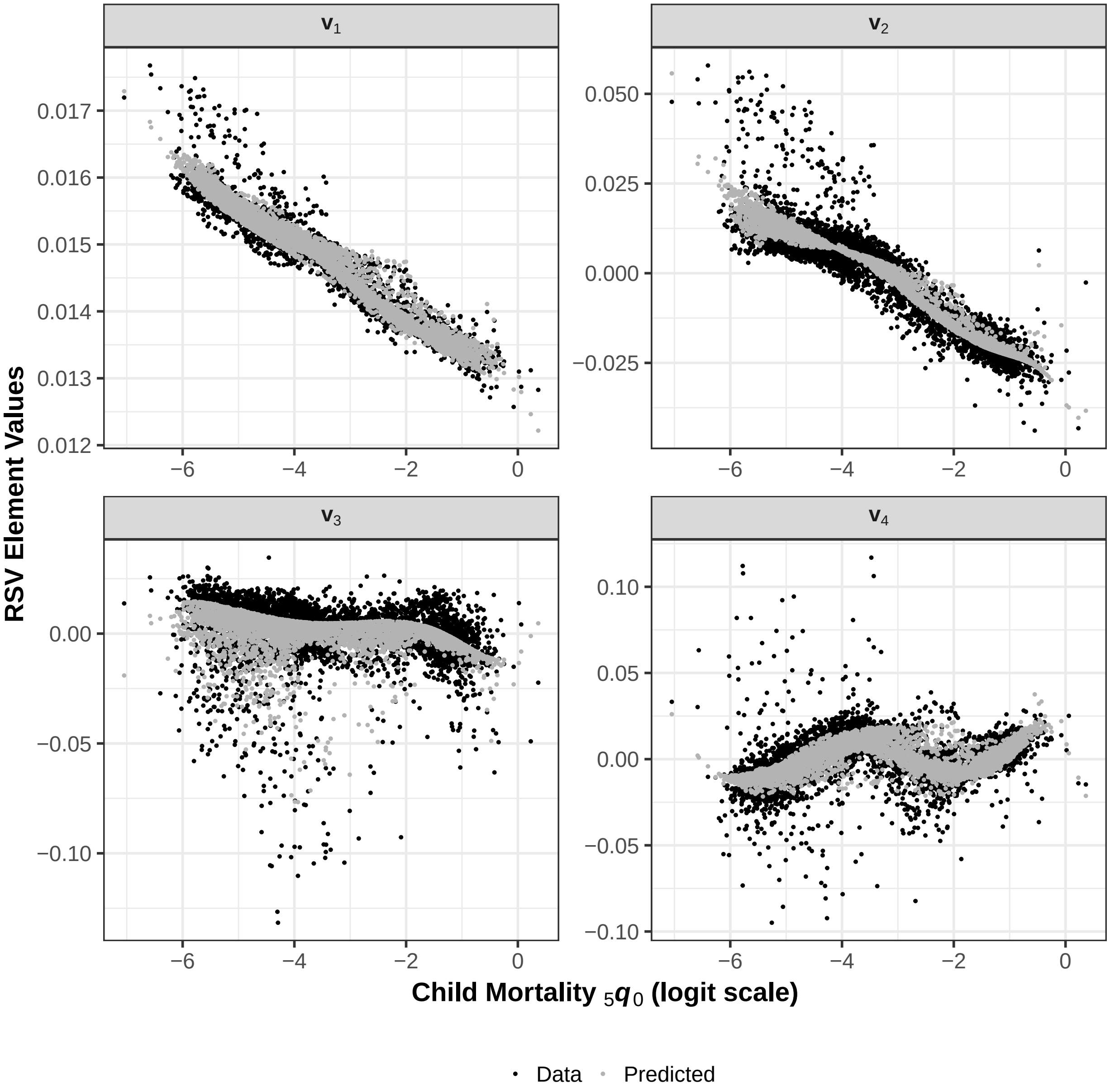
Plot right singular vectors versus child mortality, ${}_5q_0$.

```
# 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$v[, 3], Qlogit.m[1, ]), cbind("Male",
        "v4", svd.m$v[, 4], Qlogit.m[1, ]))
```

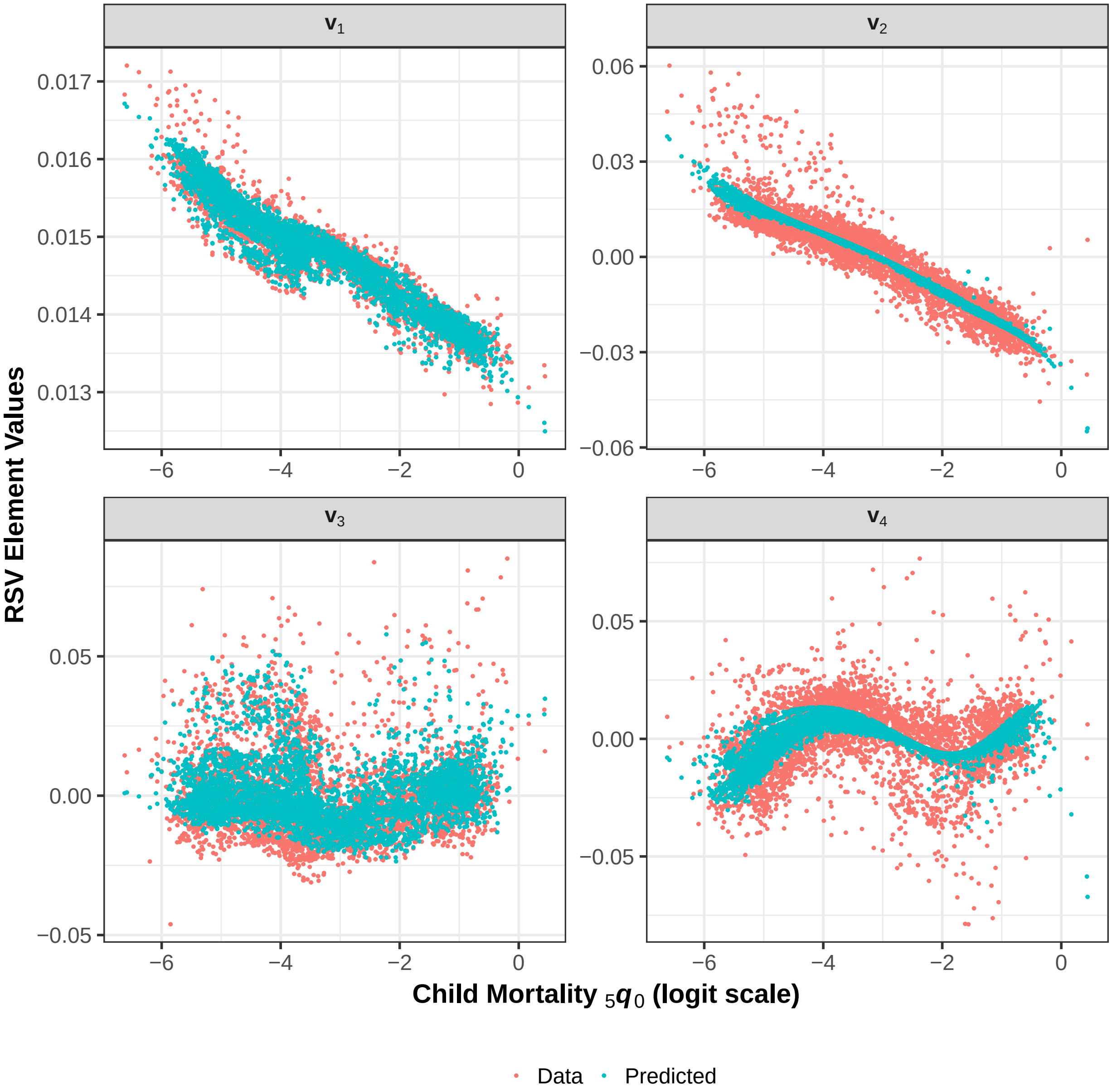
```
vs.cm.df <- data.frame(Sex = as.character(vs.cm[, 1]), V = as.character(vs.cm[,
    2]), Value = as.numeric(vs.cm[, 3]), CM = as.numeric(vs.cm[,
    4]))
# str(vs.cm.df)

# 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 = 0.7) + 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 = 0.7) + 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"))
```

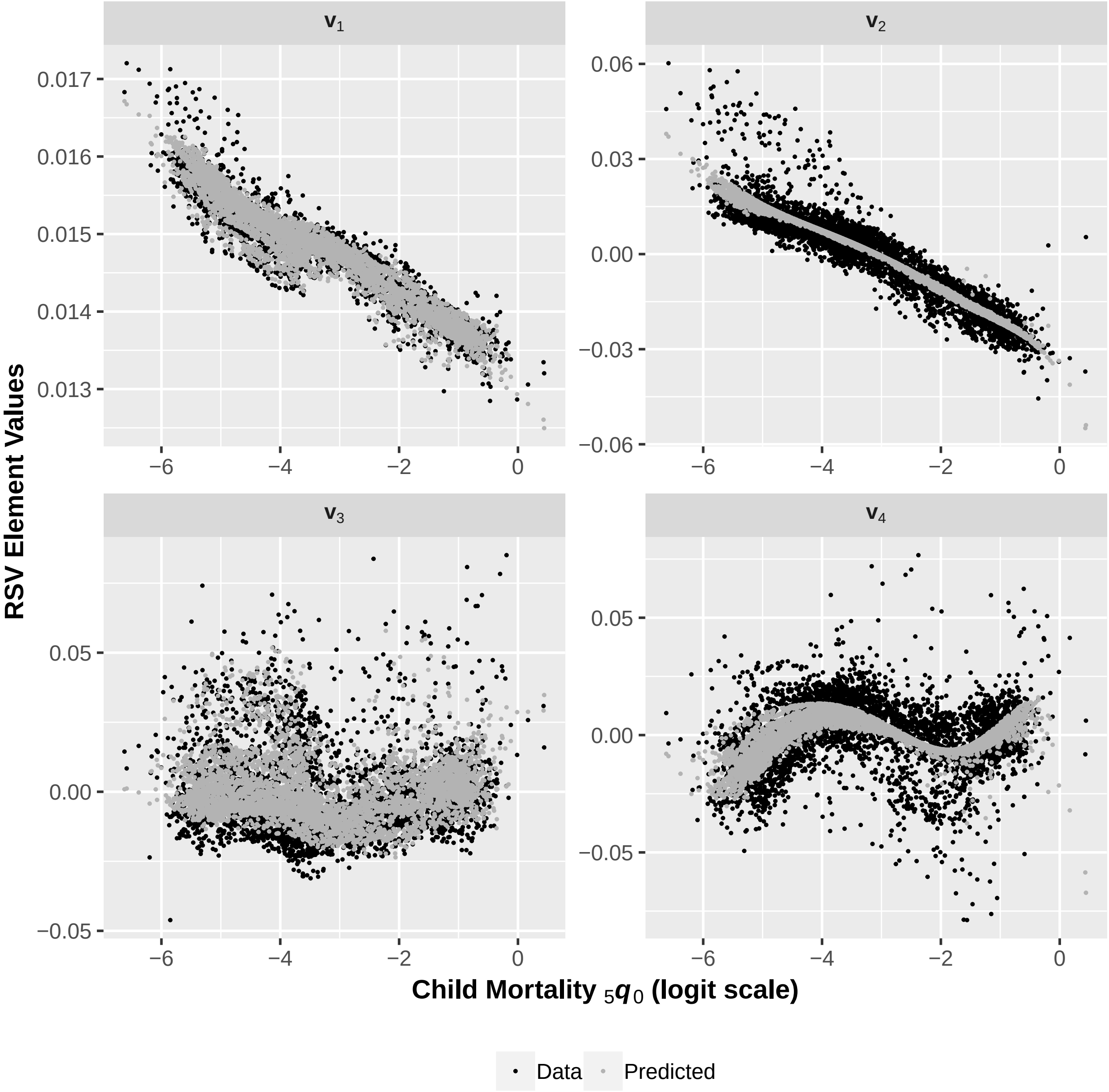

v1
v2
v3
v4
RSV Element Values
0.017
0.016
0.015
0.014
0.013
0.012
0.050
0.025
0.000
−0.025
0.00
−0.05
−0.10
0.10
0.05
0.00
−0.05
−0.10
−6
−4
−2
0
Child Mortality $_5q_0$ (logit scale)
Data
Predicted



$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
$\mathbf{v}_4$
RSV Element Values
0.017
0.016
0.015
0.014
0.013
0.012
0.050
0.025
0.000
−0.025
0.00
−0.05
−0.10
0.10
0.05
0.00
−0.05
−0.10
−6
−4
−2
0
Child Mortality $_5q_0$ (logit scale)
● Data
● Predicted



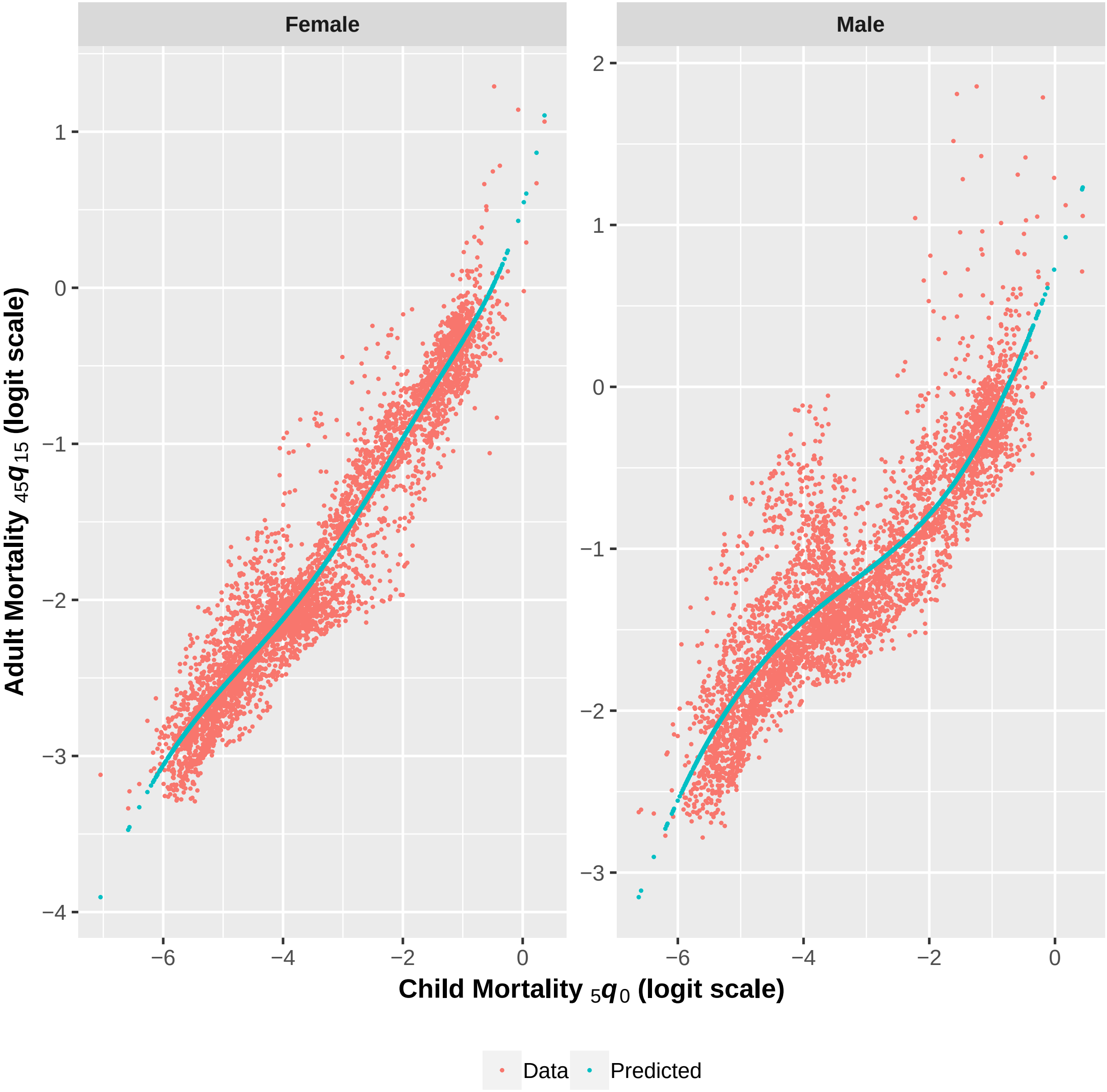
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$
$\mathbf{v}_4$
RSV Element Values
0.017
0.016
0.015
0.014
0.013
0.06
0.03
0.00
−0.03
−0.06
0.05
0.00
−0.05
0.05
0.00
−0.05
−6
−4
−2
0
Child Mortality $_5q_0$ (logit scale)
● Data
● Predicted

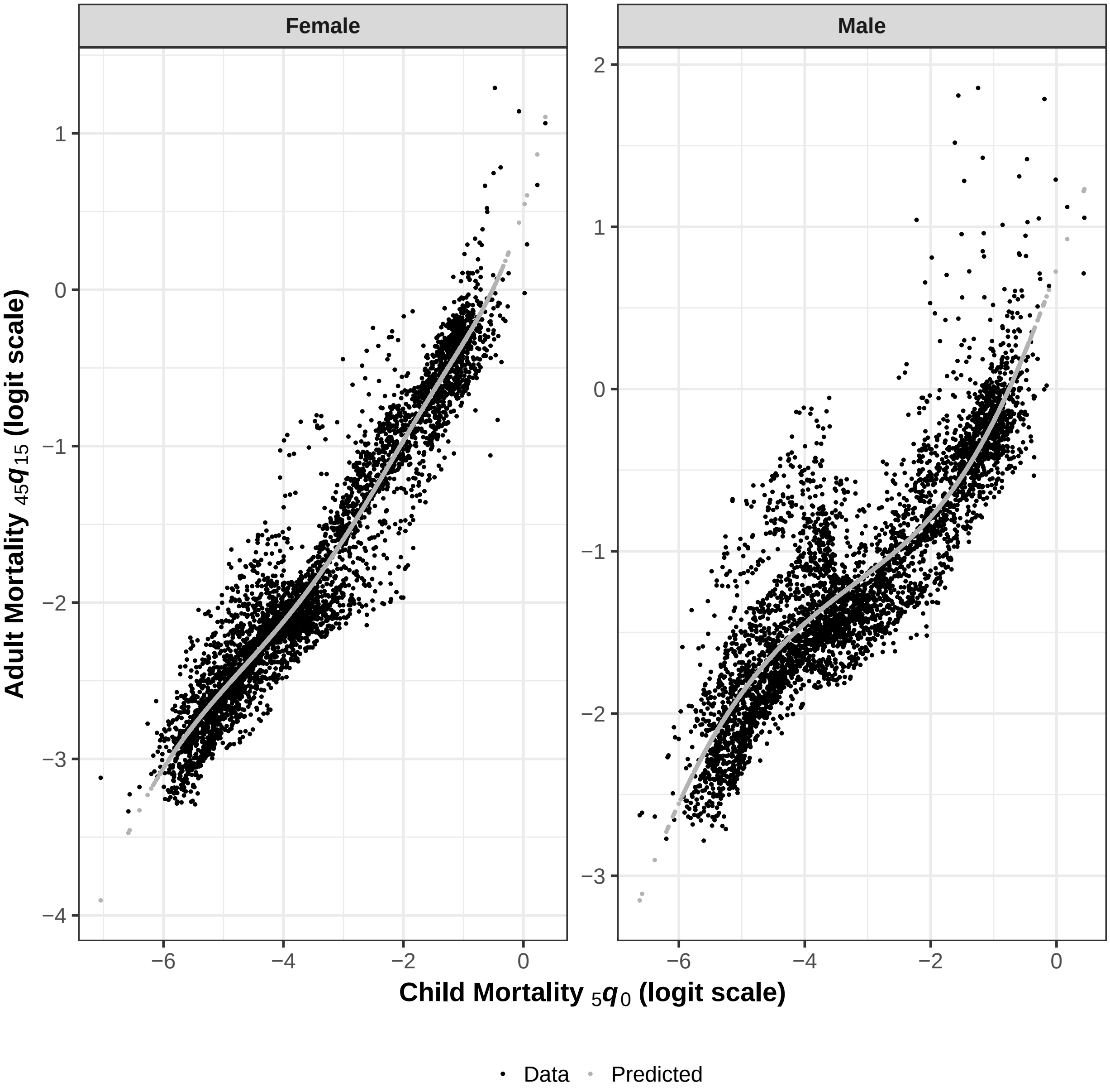


Plot adult mortality, $_{4}5q_{15}$, by child mortality, $_5q_0$.

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

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

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

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

```
        3]))
# str(am.cm.p.df)

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

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

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

# Plot
ggplot(data = am.cm, aes(x = CM, y = AM, group = Type, colour = Type)) +
    geom_point(size = 0.2) + labs(y = expression(bold("Adult Mortality ")[bold(45)] *
    bolditalic("q")[bold(15)] * bold(" (logit scale)")),
    x = expression(bold("Child Mortality ")[bold(5)] * bolditalic("q")[bold(0)] *
        bold(" (logit scale)"))) + # theme(legend.justification=c(1,0),
# legend.position=c(0.99,0.02)) +
theme(legend.position = "bottom", legend.box = "horizontal") +
    theme(legend.title = element_blank()) + facet_wrap(~Sex,
    scale = "free", 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 = 0.7) + 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"))
```

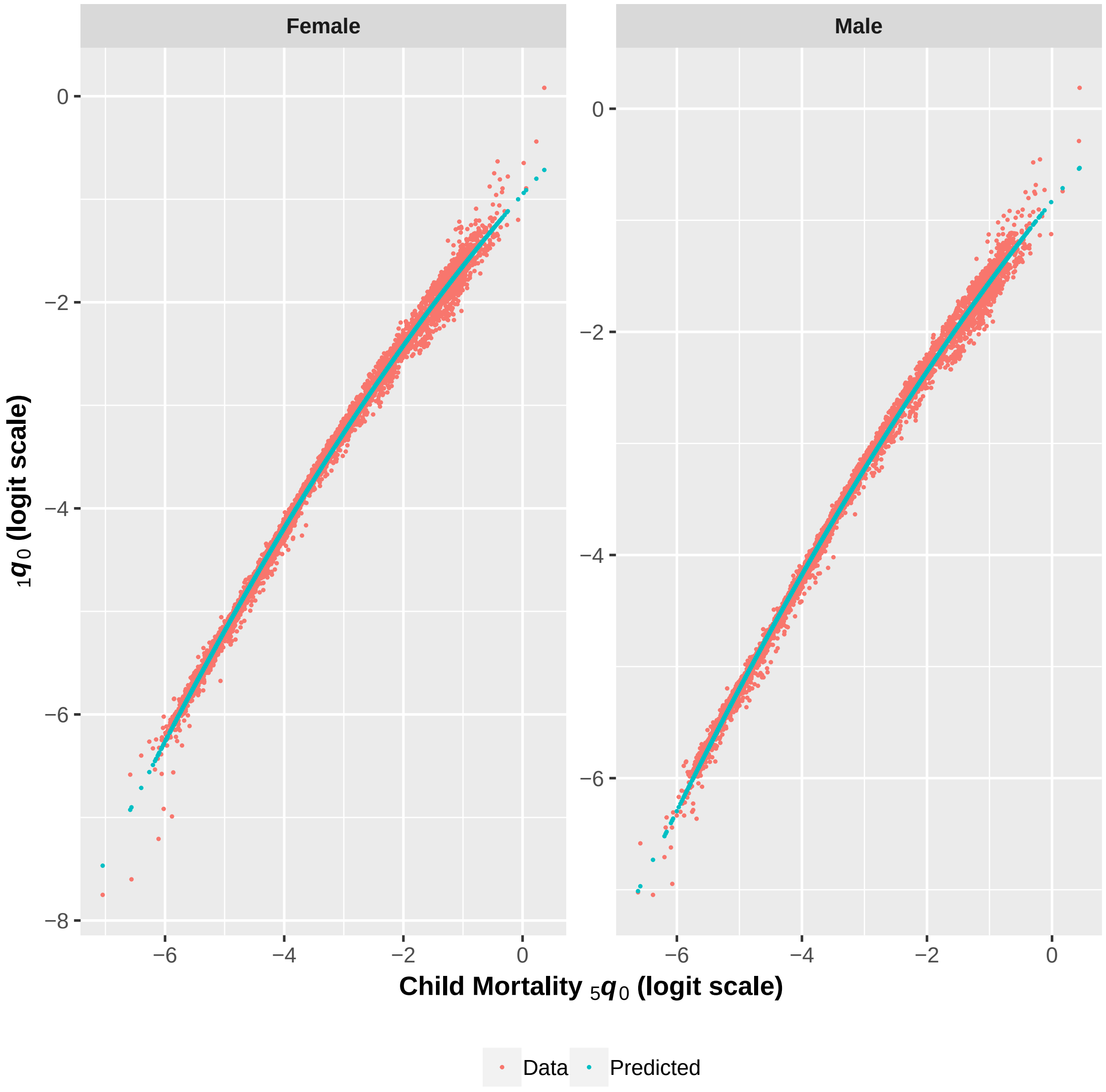

Female
Male
Adult Mortality $_{45}q_{15}$ (logit scale)
Child Mortality $_{5}q_{0}$ (logit scale)
1
0
−1
−2
−3
−4
2
1
0
−1
−2
−3
−6
−4
−2
0
−6
−4
−2
0
● Data
● Predicted

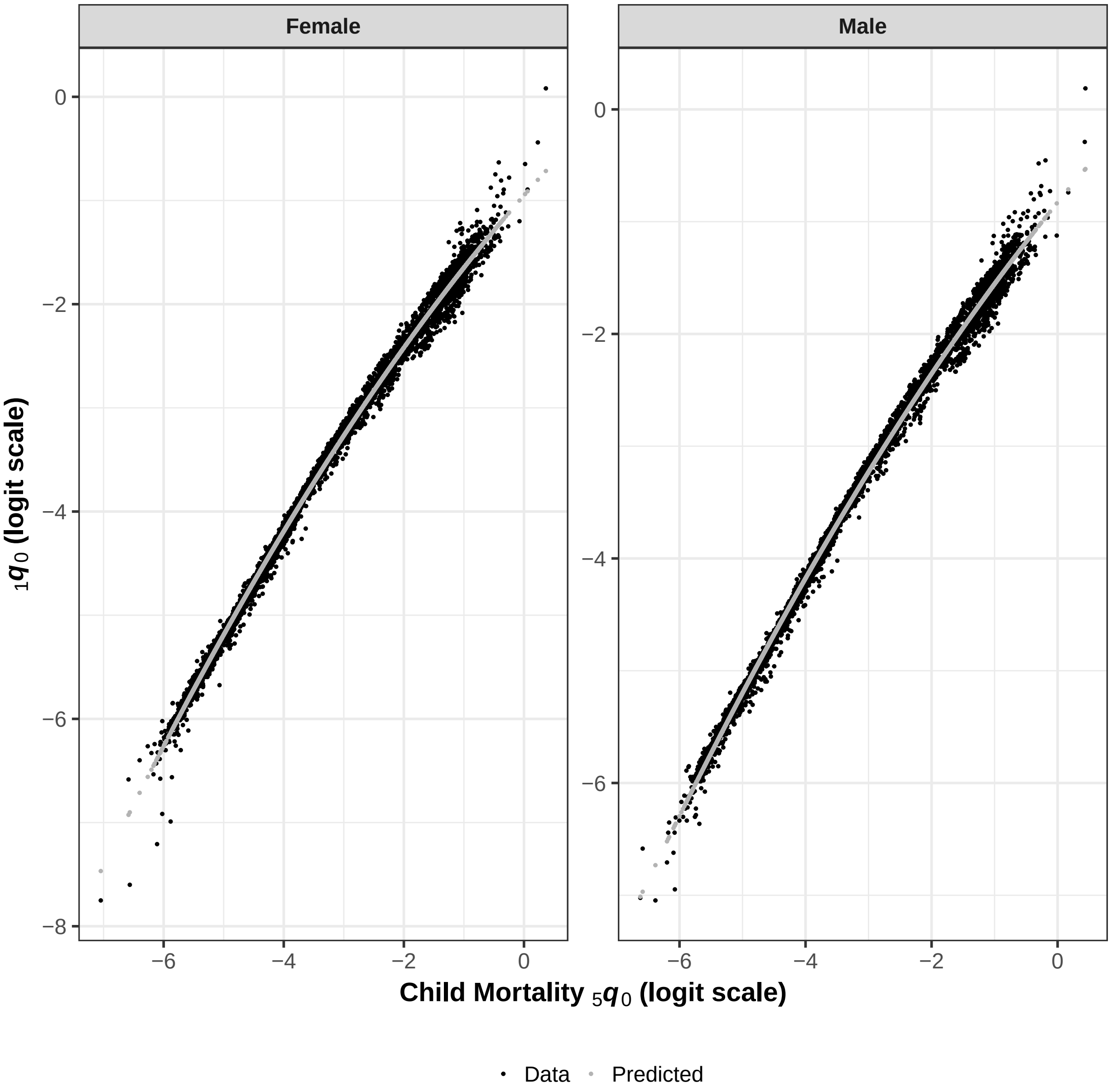


Plot probability of dying in the first year of life, $_1q_0$, versus child mortality, $_5q_0$.

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

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

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

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

```
# str(q0.cm.m.p)

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

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

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

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

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

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

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

# grayscale
ggplot(data = q0, aes(x = CM, y = q0, group = Type, colour = Type)) +
    geom_point(size = 0.2) + labs(y = expression(""[bold(1)] *
    bolditalic("q")[0] * bold(" (logit scale)")), x = expression(bold("Child Mortality ")[bold(5)] *
    bolditalic("q")[bold(0)] * bold(" (logit scale)"))) +
    # theme(legend.justification=c(1,0),
# legend.position=c(0.99,0.02)) +
theme_bw() + theme(legend.position = "bottom", legend.box = "horizontal") +
    theme(legend.title = element_blank()) + facet_wrap(~Sex,
    scale = "free", labeller = s.labeller) + scale_colour_grey(start = 0,
    end = 0.7) + 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"))
```

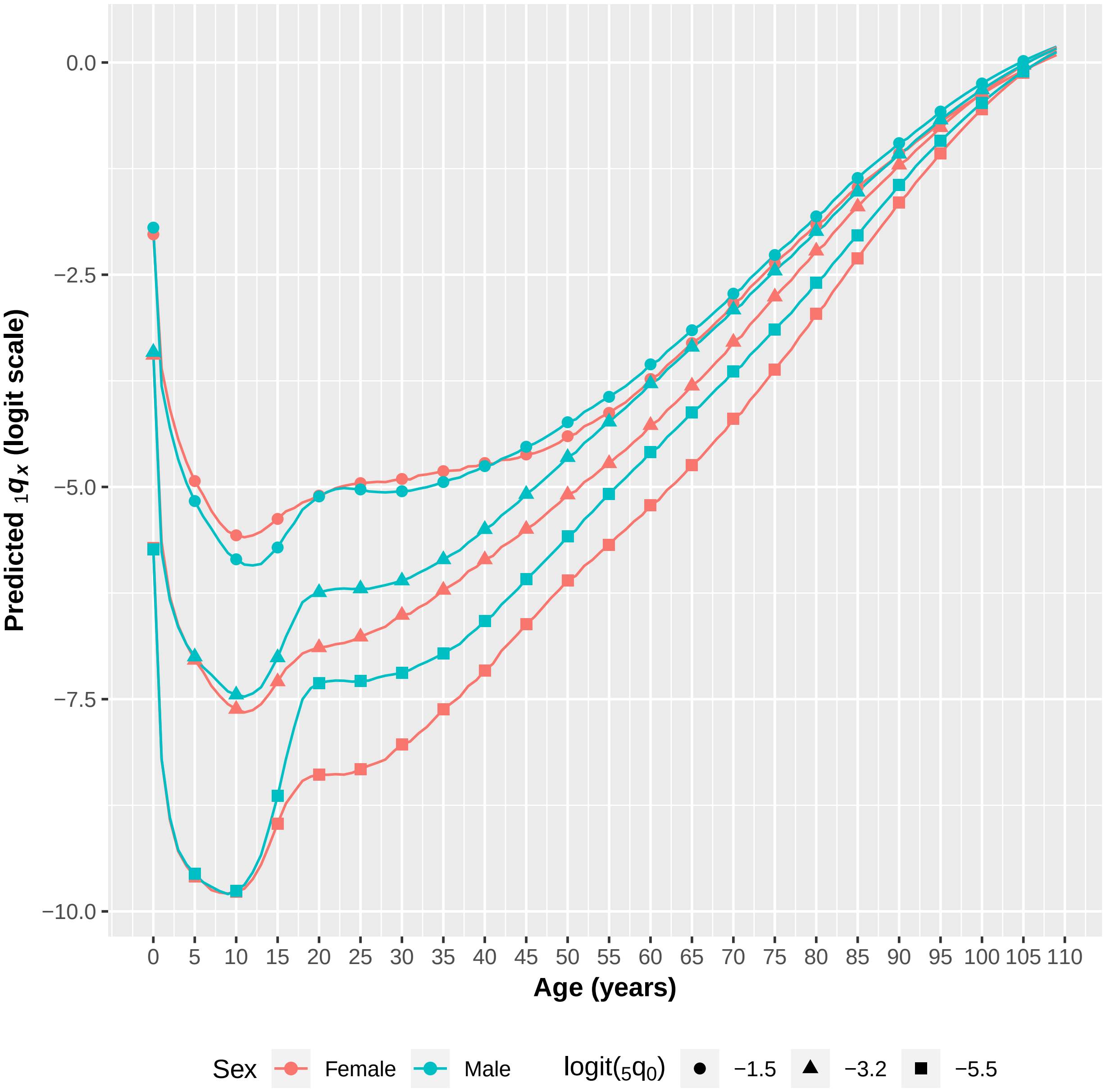

Female
Male
0
−2
−4
−6
−8
−6
−4
−2
0
$_1q_0$ (logit scale)
Child Mortality $_5q_0$ (logit scale)
Data
Predicted

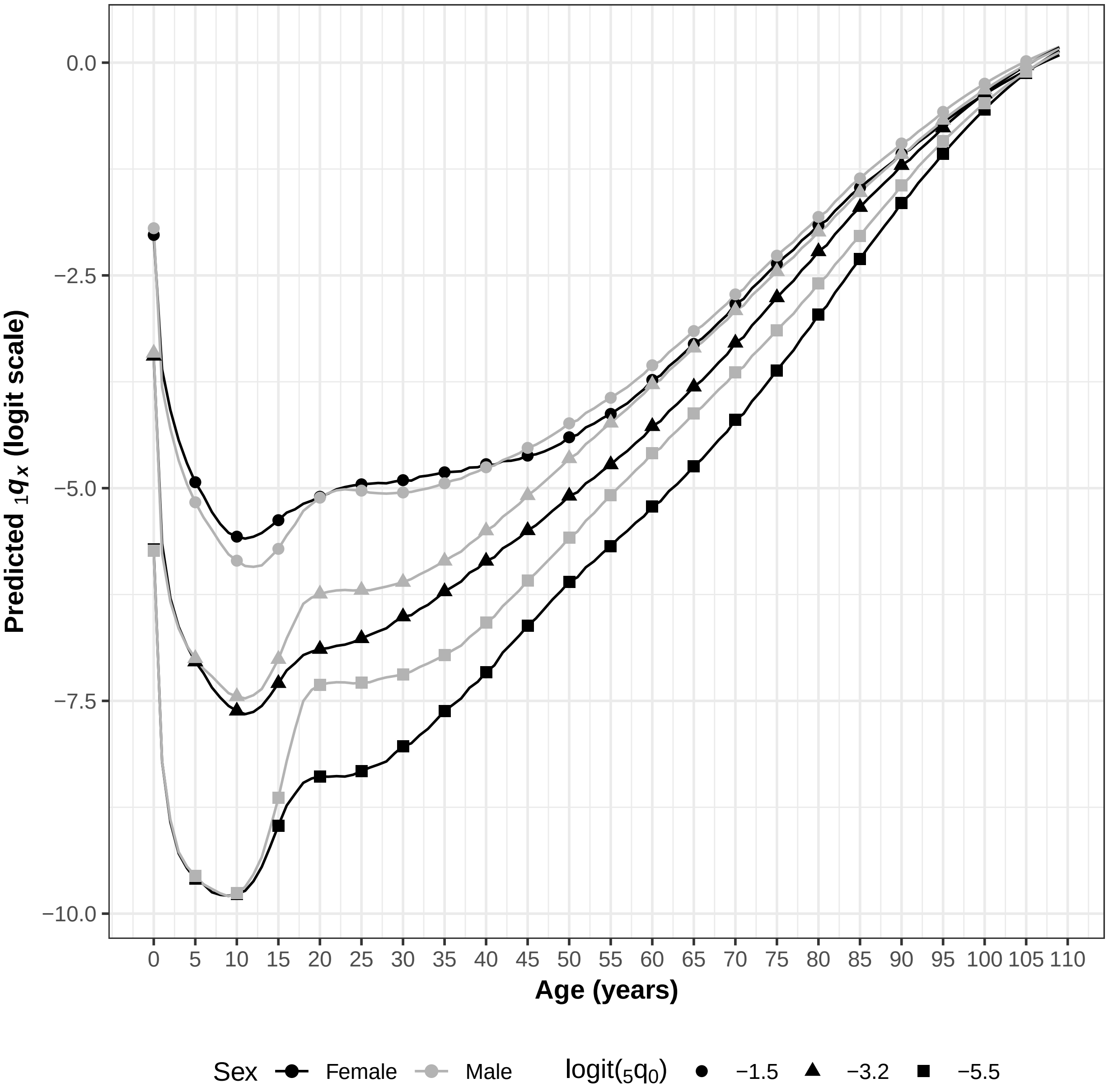


Plot heuristic predictions or life tables at three levels of child mortality from very low to very high.

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

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

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

```
    cml.input[1], lt.m[, 1]), cbind("Male", as.numeric(rownames(lt.m)),
    cml.input[2], lt.m[, 2]), cbind("Male", as.numeric(rownames(lt.m)),
    cml.input[3], lt.m[, 3]))

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

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

# Plot
ggplot(data = lt.p.df, aes(x = Age, y = ql, group = interaction(Sex,
    cml), colour = Sex, shape = cml)) + geom_line() + geom_point(data = lt.p.df[seq(1,
    nrow(lt.p.df), 5), ], size = 2) + scale_x_continuous(breaks = c(seq(0,
    110, 5))) + labs(y = expression(bold("Predicted ")[bold(1)] *
    bolditalic("q")[bolditalic(x)] * bold(" (logit scale)")),
    x = expression(bold("Age (years)"))) + # theme(legend.justification=c(1,0),
# legend.position=c(0.95,0.05)) +
theme_bw() + theme(legend.position = "bottom", legend.box = "horizontal") +
    scale_shape_discrete(name = expression("logit("[5] *
        "q"[0] * ")")) + scale_colour_grey(start = 0, end = 0.7) +
    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
```

```
# 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 # 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 -- 2014 & 94  \\
## AUT & Austria & 1947 -- 2017 & 71  \\
## BEL & Belgium & 1841 -- 1913 & 73  \\
## BEL & Belgium & 1919 -- 2015 & 97  \\
## BGR & Bulgaria & 1947 -- 2010 & 64  \\
## BLR & Belarus & 1959 -- 2016 & 58  \\
## CAN & Canada & 1921 -- 2011 & 91  \\
## CHE & Switzerland & 1876 -- 2016 & 141  \\
## CHL & Chile & 1992 -- 2008 & 17  \\
## CZE & Czechia & 1950 -- 2016 & 67  \\
## DEUTE & East Germany & 1956 -- 2015 & 60  \\
## DEUTNP & Germany & 1990 -- 2015 & 26  \\
## DEUTW & West Germany & 1956 -- 2015 & 60  \\
## DNK & Denmark & 1835 -- 2016 & 182  \\
## ESP & Spain & 1908 -- 2016 & 109  \\
## EST & Estonia & 1959 -- 2017 & 59  \\
```

```
## FIN & Finland & 1878 -- 2015 & 138  \\
## FRACNP & France -- Civilian Population & 1816 -- 2016 & 201  \\
## FRATNP & France -- Total Population & 1816 -- 2016 & 201  \\
## GBRCENW & England and Wales -- Civilian National Population & 1841 -- 2016 & 176  \\
## GBRTENW & England and Wales -- Total Population & 1841 -- 2016 & 176  \\
## GBR_NIR & Northern Ireland & 1922 -- 2016 & 95  \\
## GBR_NP & United Kingdom & 1922 -- 2016 & 95  \\
## GBR_SCO & Scotland & 1855 -- 2016 & 162  \\
## GRC & Greece & 1981 -- 2013 & 33  \\
## HRV & Croatia & 2002 -- 2016 & 15  \\
## HUN & Hungary & 1950 -- 2017 & 68  \\
## IRL & Ireland & 1950 -- 2014 & 65  \\
## ISL & Iceland & 1838 -- 1851 & 14  \\
## ISL & Iceland & 1853 -- 2016 & 164  \\
## ISR & Israel & 1983 -- 2014 & 32  \\
## ITA & Italy & 1872 -- 2014 & 143  \\
## JPN & Japan & 1947 -- 2016 & 70  \\
## KOR & Korea & 2003 -- 2016 & 14  \\
## LTU & Lithuania & 1959 -- 2017 & 59  \\
## LUX & Luxembourg & 1960 -- 2014 & 55  \\
## LVA & Latvia & 1959 -- 2017 & 59  \\
## NLD & Netherlands & 1850 -- 2016 & 167  \\
## NOR & Norway & 1846 -- 2014 & 169  \\
## 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 -- 2013 & 66  \\
## POL & Poland & 1958 -- 2016 & 59  \\
## PRT & Portugal & 1940 -- 2015 & 76  \\
## RUS & Russia & 1959 -- 2014 & 56  \\
## SVK & Slovakia & 1950 -- 2014 & 65  \\
## SVN & Slovenia & 1983 -- 2014 & 32  \\
## SWE & Sweden & 1751 -- 2016 & 266  \\
## TWN & Taiwan & 1970 -- 2014 & 45  \\
## UKR & Ukraine & 1959 -- 2013 & 55  \\
## USA & The United States of America & 1933 -- 2016 & 84  \\
```

```r
# create a dataframe with the country names,
# abbreviations, and number of consecutive life tables
ltGrandTot <- 0
life.tables <- data.frame(country = character(length(ltList)),
    abbreviation = character(length(ltList)), startYear = numeric(length(ltList)),
    stopYear = numeric(length(ltList)), tables = numeric(length(ltList)),
    stringsAsFactors = FALSE)
for (i in 1:length(ltList)) {
    for (j in 1:length(country.names)) {
        if (ltList[[i]][1] == str_to_upper(country.names[[j]][1])) {
            ltTot <- as.numeric(ltList[[i]][3]) - as.numeric(ltList[[i]][2]) +
                1
            ltGrandTot <- ltGrandTot + ltTot
            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
capture.output(file = "../tables/LTSummaries.txt", print.xtable(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
if (download.date == "Local cached HMD") {
    data.date <- "November 2, 2018"
} else {
    data.date <- download.date
}
capture.output(file = "../tables/HMDdate.txt", cat(data.date))

print("")
```

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

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

```
## [1] "Total number of life tables in raw data (q1.f): 4610"
```

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

```
## [1] "Check, totalling up country counts of life tables: 4610"
```

```
print("")
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Make table of summary comparison results.

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

```
##      total.abs.error mean.abs.error max.error
## comp        1445.741     0.01363521 0.3306283
## lq          1501.606     0.01416209 0.3968440
```

```
comps.child$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1674.491     0.01579262 0.3773859
## lq          1777.257     0.01676183 0.3792040
```

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

```
comps.adult$female
```

```
##      total.abs.error mean.abs.error max.error
## comp        1297.786     0.01223980 0.2204735
## lq          1399.336     0.01319754 0.3865020
```

```
comps.adult$male
```

```
##      total.abs.error mean.abs.error max.error
## comp        1378.421     0.01300029 0.3866729
## lq          1472.245     0.01388518 0.3532550
```

```
# make lines for the table

# female make code more readable ...  a, c b, d
a.f <- comps.child$female[1, 1]  # child-only SVD-Comp
```

```r
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(a.f, big.mark = ",", digits = 0), "&", format(c.f,
        big.mark = ",", digits = 0), "&", format(c.f - a.f,
        big.mark = ",", digits = 0), " \\\\", sep = " "),
    "\n"), cat(paste("R2 & Log-Quad &", format(b.f, big.mark = ",",
    digits = 0), "&", format(d.f, big.mark = ",", digits = 0),
    "&", format(d.f - b.f, big.mark = ",", digits = 0),
    " \\\\", sep = " "), "\n"), cat(paste("R3 & R2-R1 &",
    format(round(b.f - a.f, 0), nsmall = 0), "&", format(round(d.f -
        c.f, 0), nsmall = 0), "&", format(round((d.f - b.f) -
        (c.f - a.f), 0), nsmall = 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(a.m, big.mark = ",", digits = 0), "&", format(c.m,
        big.mark = ",", digits = 0), "&", format(c.m - a.m,
        big.mark = ",", digits = 0), " \\\\", sep = " "),
    "\n"), cat(paste("R2 & Log-Quad &", format(b.m, big.mark = ",",
    digits = 0), "&", format(d.m, big.mark = ",", digits = 0),
    "&", format(d.m - b.m, big.mark = ",", digits = 0),
    " \\\\", sep = " "), "\n"), cat(paste("R3 & R2-R1 &",
    format(round(b.m - a.m, 0), nsmall = 0), "&", format(round(d.m -
        c.m, 0), nsmall = 0), "&", format(round((d.m - b.m) -
        (c.m - a.m), 0), nsmall = 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.

```r
# 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: $\\logit(\\qff)_{z \\ell} = f(\\qf_{\\, z \
    label = "tab:appA:adultMxMod", dep.var.labels.include = FALSE,
    dep.var.caption = "$\\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: $\\logit(\\qoz)_{z \\ell} = f(\\qf_{\\, z \\
    label = "tab:appA:infantMxMod", dep.var.labels.include = FALSE,
    dep.var.caption = "$\\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
capture.output(file = "../tables/vsFemale.txt", stargazer(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,
    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
capture.output(file = "../tables/vsMale.txt", stargazer(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,
    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, ] <- 1e+05  # 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)

    # 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(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        1.285808191  1.309623169 -0.0238149773
## 1-4      1.405605466  1.297656974  0.1079484913
## 5-9      0.774404676  0.739540137  0.0348645388
## 10-14    0.509533972  0.488058117  0.0214758555
## 15-19    0.631768929  0.704215885 -0.0724469560
## 20-24    0.776761164  0.856790241 -0.0800290770
## 25-29    0.767838972  0.847677326 -0.0798383535
## 30-34    0.759797072  0.801390372 -0.0415933000
## 35-39    0.835094853  0.835957359 -0.0008625060
## 40-44    0.949378962  0.918270079  0.0311088829
## 45-49    1.104900947  1.088777855  0.0161230918
## 50-54    1.446209914  1.442450719  0.0037591953
## 55-59    1.890110030  1.908891692 -0.0187816618
## 60-64    2.426462671  2.509440007 -0.0829773354
## 65-69    2.975484183  3.094838412 -0.1193542288
## 70-74    3.859023820  4.004770837 -0.1457470169
## 75-79    4.452347552  4.511095621 -0.0587480699
## 80-84    4.256483071  4.364023689 -0.1075406180
## 85-89    3.032195850  3.032863378 -0.0006675275
## 90-94    1.466339018  1.497975829 -0.0316368113
## 95-99    0.373584075  0.398302792 -0.0247187170
## 100-104  0.045990476  0.051727293 -0.0057368173
## 105-109  0.002843693  0.003311803 -0.0004681095
##         36.027967560 36.707649587 -0.6796820277
##         tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0       1.522463e+00 1.541778e+00 -0.0193146311
## 1-4     2.006411e+00 1.546716e+00  0.4596952560
## 5-9     8.794502e-01 8.656519e-01  0.0137982698
## 10-14   5.178406e-01 4.864013e-01  0.0314393559
## 15-19   8.854434e-01 8.426238e-01  0.0428195914
## 20-24   1.691406e+00 1.619755e+00  0.0716517282
## 25-29   1.567782e+00 1.504629e+00  0.0631531799
## 30-34   1.502594e+00 1.457767e+00  0.0448277045
## 35-39   1.674806e+00 1.621177e+00  0.0536296841
## 40-44   1.951835e+00 1.895655e+00  0.0561803576
## 45-49   2.351121e+00 2.316191e+00  0.0349304345
## 50-54   2.893663e+00 2.923593e+00 -0.0299296940
## 55-59   3.446941e+00 3.593629e+00 -0.1466873652
## 60-64   4.120316e+00 4.439985e+00 -0.3196690361
## 65-69   4.537801e+00 4.918807e+00 -0.3810053609
## 70-74   4.686842e+00 5.135102e+00 -0.4482603087
## 75-79   4.272008e+00 4.563438e+00 -0.2914298592
## 80-84   3.078585e+00 3.282934e+00 -0.2043487757
## 85-89   1.700815e+00 1.766515e+00 -0.0657000210
## 90-94   6.657873e-01 7.064783e-01 -0.0406910103
## 95-99   1.285520e-01 1.440494e-01 -0.0154973660
## 100-104 1.334220e-02 1.559056e-02 -0.0022483617
## 105-109 8.105134e-04 9.597548e-04 -0.0001492414
##         4.609662e+01 4.718942e+01 -1.0928054694
##
```

```
## $tae.e
##         tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.108094e+02 4.147746e+02 -3.965212e+00
## 1-4     4.502406e+02 4.594468e+02 -9.206167e+00
## 5-9     4.164232e+02 4.257798e+02 -9.356524e+00
## 10-14   3.954913e+02 4.044524e+02 -8.961155e+00
## 15-19   3.815325e+02 3.889475e+02 -7.414935e+00
## 20-24   3.618813e+02 3.668358e+02 -4.954584e+00
## 25-29   3.401536e+02 3.433747e+02 -3.221157e+00
## 30-34   3.209064e+02 3.236770e+02 -2.770591e+00
## 35-39   3.042722e+02 3.084361e+02 -4.163924e+00
## 40-44   2.879651e+02 2.936671e+02 -5.702007e+00
## 45-49   2.700467e+02 2.762443e+02 -6.197647e+00
## 50-54   2.487578e+02 2.548594e+02 -6.101545e+00
## 55-59   2.240441e+02 2.294080e+02 -5.363838e+00
## 60-64   1.956326e+02 2.005822e+02 -4.949684e+00
## 65-69   1.648870e+02 1.678420e+02 -2.955035e+00
## 70-74   1.308905e+02 1.327151e+02 -1.824629e+00
## 75-79   9.421929e+01 9.527098e+01 -1.051684e+00
## 80-84   5.958329e+01 6.014278e+01 -5.594867e-01
## 85-89   3.100424e+01 3.091626e+01  8.798583e-02
## 90-94   1.240199e+01 1.235616e+01  4.582874e-02
## 95-99   3.209910e+00 3.197597e+00  1.231316e-02
## 100-104 4.565116e-01 4.550265e-01  1.485138e-03
## 105-109 3.451248e-02 3.425943e-02  2.530459e-04
## 110+    5.822578e-03 2.470463e-03  3.352115e-03
##         5.104850e+03 5.193418e+03 -8.856859e+01
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.189620e+02 6.306460e+02 -1.168393e+01
## 1-4     6.577537e+02 6.605873e+02 -2.833600e+00
## 5-9     6.085742e+02 6.336571e+02 -2.508288e+01
## 10-14   5.910655e+02 6.149495e+02 -2.388406e+01
## 15-19   5.808574e+02 6.045474e+02 -2.368999e+01
## 20-24   5.590702e+02 5.820842e+02 -2.301396e+01
## 25-29   5.200264e+02 5.445539e+02 -2.452747e+01
## 30-34   4.849374e+02 5.120736e+02 -2.713621e+01
## 35-39   4.506845e+02 4.799306e+02 -2.924610e+01
## 40-44   4.143479e+02 4.441654e+02 -2.981748e+01
## 45-49   3.740160e+02 4.020256e+02 -2.800958e+01
## 50-54   3.277359e+02 3.532945e+02 -2.555868e+01
## 55-59   2.759549e+02 2.979205e+02 -2.196555e+01
## 60-64   2.203616e+02 2.383392e+02 -1.797765e+01
## 65-69   1.637741e+02 1.764317e+02 -1.265764e+01
## 70-74   1.119345e+02 1.193255e+02 -7.391022e+00
## 75-79   6.791521e+01 7.092950e+01 -3.014291e+00
## 80-84   3.590373e+01 3.653155e+01 -6.278165e-01
## 85-89   1.588130e+01 1.578827e+01  9.303009e-02
## 90-94   5.518531e+00 5.459889e+00  5.864214e-02
## 95-99   1.154423e+00 1.150808e+00  3.615189e-03
## 100-104 1.399408e-01 1.404794e-01 -5.385160e-04
## 105-109 1.030230e-02 1.022130e-02  8.100027e-05
## 110+    2.868997e-03 1.001968e-03  1.867030e-03
##         7.086583e+03 7.424544e+03 -3.379612e+02
##
```

```
## $tot.tae.e0
##            [,1]    [,2]       [,3]
## Female 6134.350 6193.56  -59.20994
## Male   8595.722 8757.98 -162.25837
```

```
# create lines for tables
capture.output(file = "../tables/ageCompQ-1.txt", print.xtable(xtable(age.comps$tae.q[(1:nrow(age.comps
    1), ], booktabs = TRUE, digits = 4), only.contents = TRUE,
    include.rownames = TRUE, include.colnames = FALSE, hline.after = NULL,
    format.args = list(big.mark = ",")))
capture.output(file = "../tables/ageCompQ-2.txt", cat(paste("0-109 & ",
    paste(format(round(age.comps$tae.q[nrow(age.comps$tae.q),
        ], 4), format = "d", big.mark = ","), collapse = " & "),
    " \\\\", sep = "")))

capture.output(file = "../tables/ageCompE-1.txt", print.xtable(xtable(age.comps$tae.e[(1:nrow(age.comps
    1), ], booktabs = TRUE, digits = 2), only.contents = TRUE,
    include.rownames = TRUE, include.colnames = FALSE, hline.after = NULL,
    format.args = list(big.mark = ",")))
capture.output(file = "../tables/ageCompE-2.txt", cat(paste("0+ & ",
    paste(format(round(age.comps$tae.e[nrow(age.comps$tae.e),
        ], 2), format = "d", big.mark = ","), collapse = " & "),
    " \\\\", sep = "")))

capture.output(file = "../tables/ageCompTot.txt", print.xtable(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
capture.output(file = "../tables/us.txt", print.xtable(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"
```

```r
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"
```

```r
# 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")
```

```r
age.comps.1
```

```
## $tae.q
##         tae.comp.q.f   tae.lq.q.f tae.diff.q.f
## 0         1.28580819  1.309623169 -0.023814977
## 1-4       6.50578917  1.297656974  5.208132196
## 5-9       1.94209962  0.739540137  1.202559488
## 10-14     1.04369833  0.488058117  0.555640218
## 15-19     1.09731300  0.704215885  0.393097114
## 20-24     1.36293794  0.856790241  0.506147700
## 25-29     1.41027018  0.847677326  0.562592856
## 30-34     1.26783812  0.801390372  0.466447749
## 35-39     1.00444260  0.835957359  0.168485238
## 40-44     0.97341055  0.918270079  0.055140469
## 45-49     1.88886195  1.088777855  0.800084093
## 50-54     3.08802264  1.442450719  1.645571926
## 55-59     4.55643503  1.908891692  2.647543341
## 60-64     5.94447983  2.509440007  3.435039824
## 65-69     7.94402675  3.094838412  4.849188338
## 70-74     9.81464282  4.004770837  5.809871980
## 75-79    11.41790149  4.511095621  6.906805870
## 80-84    11.04869359  4.364023689  6.684669896
## 85-89     7.83804137  3.032863378  4.805177991
## 90-94     3.74919759  1.497975829  2.251221765
## 95-99     0.99245394  0.398302792  0.594151143
## 100-104   0.13069329  0.051727293  0.078965993
## 105-109   0.00845995  0.003311803  0.005148147
##          86.31551794 36.707649587 49.607868357
##         tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0        1.522463392 1.541778e+00 -0.0193146311
## 1-4      7.641210211 1.546716e+00  6.0944942571
## 5-9      2.212398818 8.656519e-01  1.3467468819
```

```
## 10-14     1.033069197 4.864013e-01  0.5466679154
## 15-19     1.002918310 8.426238e-01  0.1602944674
## 20-24     1.854624869 1.619755e+00  0.2348701542
## 25-29     1.729233067 1.504629e+00  0.2246041962
## 30-34     1.627440364 1.457767e+00  0.1696735963
## 35-39     1.692017226 1.621177e+00  0.0708404335
## 40-44     1.974881556 1.895655e+00  0.0792269319
## 45-49     2.784572526 2.316191e+00  0.4683815895
## 50-54     4.259596141 2.923593e+00  1.3360029713
## 55-59     6.378550406 3.593629e+00  2.7849217656
## 60-64     8.410207743 4.439985e+00  3.9702222533
## 65-69    10.144636897 4.918807e+00  5.2258302135
## 70-74    10.420683920 5.135102e+00  5.2855819548
## 75-79     9.414432550 4.563438e+00  4.8509942026
## 80-84     6.814196421 3.282934e+00  3.5312628209
## 85-89     3.739018307 1.766515e+00  1.9725034807
## 90-94     1.433720148 7.064783e-01  0.7272418454
## 95-99     0.283417233 1.440494e-01  0.1393678438
## 100-104   0.030178140 1.559056e-02  0.0145875788
## 105-109   0.001855555 9.597548e-04  0.0008958003
##          86.405322997 4.718942e+01 39.2158985231
##
## $tae.e
##          tae.comp.e.f   tae.lq.e.f tae.diff.e.f
## 0        7.627039e+02 4.147746e+02 3.479293e+02
## 1-4      8.068862e+02 4.594468e+02 3.474393e+02
## 5-9      5.575243e+02 4.257798e+02 1.317445e+02
## 10-14    5.231429e+02 4.044524e+02 1.186905e+02
## 15-19    5.223195e+02 3.889475e+02 1.333721e+02
## 20-24    5.302867e+02 3.668358e+02 1.634509e+02
## 25-29    5.549217e+02 3.433747e+02 2.115469e+02
## 30-34    5.869899e+02 3.236770e+02 2.633130e+02
## 35-39    6.148764e+02 3.084361e+02 3.064403e+02
## 40-44    6.296619e+02 2.936671e+02 3.359948e+02
## 45-49    6.230954e+02 2.762443e+02 3.468510e+02
## 50-54    5.885393e+02 2.548594e+02 3.336799e+02
## 55-59    5.365671e+02 2.294080e+02 3.071591e+02
## 60-64    4.696153e+02 2.005822e+02 2.690330e+02
## 65-69    3.980527e+02 1.678420e+02 2.302107e+02
## 70-74    3.190440e+02 1.327151e+02 1.863289e+02
## 75-79    2.387816e+02 9.527098e+01 1.435106e+02
## 80-84    1.583294e+02 6.014278e+01 9.818659e+01
## 85-89    8.548431e+01 3.091626e+01 5.456806e+01
## 90-94    3.437061e+01 1.235616e+01 2.201445e+01
## 95-99    8.855386e+00 3.197597e+00 5.657789e+00
## 100-104  1.283821e+00 4.550265e-01 8.287947e-01
## 105-109  9.815337e-02 3.425943e-02 6.389394e-02
## 110+     5.822578e-03 2.470463e-03 3.352115e-03
##          9.551436e+03 5.193418e+03 4.358018e+03
##          tae.comp.e.m   tae.lq.e.m tae.diff.e.m
## 0        9.072944e+02 6.306460e+02 2.766485e+02
## 1-4      9.540774e+02 6.605873e+02 2.934902e+02
## 5-9      7.019789e+02 6.336571e+02 6.832173e+01
## 10-14    7.029391e+02 6.149495e+02 8.798963e+01
```

```
## 15-19   7.164021e+02 6.045474e+02 1.118547e+02
## 20-24   7.127801e+02 5.820842e+02 1.306959e+02
## 25-29   7.056740e+02 5.445539e+02 1.611201e+02
## 30-34   7.016987e+02 5.120736e+02 1.896251e+02
## 35-39   6.973163e+02 4.799306e+02 2.173857e+02
## 40-44   6.876381e+02 4.441654e+02 2.434727e+02
## 45-49   6.649454e+02 4.020256e+02 2.629197e+02
## 50-54   6.189124e+02 3.532945e+02 2.656179e+02
## 55-59   5.493874e+02 2.979205e+02 2.514669e+02
## 60-64   4.557993e+02 2.383392e+02 2.174600e+02
## 65-69   3.511330e+02 1.764317e+02 1.747013e+02
## 70-74   2.450164e+02 1.193255e+02 1.256909e+02
## 75-79   1.539049e+02 7.092950e+01 8.297539e+01
## 80-84   8.383266e+01 3.653155e+01 4.730111e+01
## 85-89   3.726925e+01 1.578827e+01 2.148098e+01
## 90-94   1.249236e+01 5.459889e+00 7.032468e+00
## 95-99   2.551372e+00 1.150808e+00 1.400564e+00
## 100-104 3.066342e-01 1.404794e-01 1.661549e-01
## 105-109 2.235596e-02 1.022130e-02 1.213466e-02
## 110+    2.868997e-03 1.001968e-03 1.867030e-03
##         1.066338e+04 7.424544e+03 3.238832e+03
##
## $tot.tae.e0
##            [,1]    [,2]     [,3]
## Female 11388.96 6193.56 5195.403
## Male   12599.88 8757.98 3841.905
```

```
# create a table with results for 1 components
capture.output(file = "../tables/ageCompTotC-1.txt", print.xtable(xtable(age.comps.1$tot.tae.e0,
    booktabs = TRUE, digits = 2), only.contents = TRUE,
    include.rownames = TRUE, include.colnames = FALSE, hline.after = NULL,
    format.args = list(big.mark = ",")))

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

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

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

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

```
## [1] "1 iterations"
## [1] "100% sample fraction"
## [1] "2 components"
```

```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         1.285808191  1.309623169 -0.0238149773
## 1-4       1.511146077  1.297656974  0.2134891026
## 5-9       0.808544141  0.739540137  0.0690040039
## 10-14     0.526631577  0.488058117  0.0385734600
## 15-19     0.651270698  0.704215885 -0.0529451868
## 20-24     0.804282231  0.856790241 -0.0525080104
## 25-29     0.785431675  0.847677326 -0.0622456506
## 30-34     0.764372635  0.801390372 -0.0370177376
## 35-39     0.848389503  0.835957359  0.0124321432
## 40-44     0.976435675  0.918270079  0.0581655961
## 45-49     1.132657678  1.088777855  0.0438798237
## 50-54     1.483978917  1.442450719  0.0415281982
## 55-59     1.930368991  1.908891692  0.0214772991
## 60-64     2.551170467  2.509440007  0.0417304598
## 65-69     3.135437380  3.094838412  0.0405989679
## 70-74     4.139780124  4.004770837  0.1350092866
## 75-79     4.810746714  4.511095621  0.2996510921
## 80-84     4.502584955  4.364023689  0.1385612657
## 85-89     3.085540322  3.032863378  0.0526769446
## 90-94     1.460562863  1.497975829 -0.0374129663
## 95-99     0.376024852  0.398302792 -0.0222779403
## 100-104   0.047565413  0.051727293 -0.0041618803
## 105-109   0.003015721  0.003311803 -0.0002960819
##          37.621746799 36.707649587  0.9140972120
##          tae.comp.q.m   tae.lq.q.m   tae.diff.q.m
## 0         1.522463392 1.541778e+00 -1.931463e-02
## 1-4       1.994024064 1.546716e+00  4.473081e-01
## 5-9       0.934275058 8.656519e-01  6.862312e-02
## 10-14     0.548839260 4.864013e-01  6.243798e-02
## 15-19     0.874673949 8.426238e-01  3.205011e-02
## 20-24     1.662009166 1.619755e+00  4.225445e-02
## 25-29     1.551894271 1.504629e+00  4.726540e-02
## 30-34     1.506337223 1.457767e+00  4.857046e-02
## 35-39     1.669623278 1.621177e+00  4.844649e-02
## 40-44     1.971951950 1.895655e+00  7.629733e-02
## 45-49     2.360615129 2.316191e+00  4.442419e-02
## 50-54     2.889954239 2.923593e+00 -3.363893e-02
## 55-59     3.510055132 3.593629e+00 -8.357351e-02
## 60-64     4.298700276 4.439985e+00 -1.412852e-01
## 65-69     4.805026438 4.918807e+00 -1.137802e-01
```

```
## 70-74    4.973187843 5.135102e+00 -1.619141e-01
## 75-79    4.517732719 4.563438e+00 -4.570563e-02
## 80-84    3.169823587 3.282934e+00 -1.131100e-01
## 85-89    1.742350857 1.766515e+00 -2.416397e-02
## 90-94    0.685338361 7.064783e-01 -2.113994e-02
## 95-99    0.135349685 1.440494e-01 -8.699704e-03
## 100-104  0.014268151 1.559056e-02 -1.322411e-03
## 105-109  0.000868547 9.597548e-04 -9.120786e-05
##         47.339362576 4.718942e+01  1.499381e-01
## 
## $tae.e
##         tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.255861e+02 4.147746e+02  10.811450934
## 1-4     4.628845e+02 4.594468e+02   3.437701494
## 5-9     4.247085e+02 4.257798e+02  -1.071290624
## 10-14   4.035180e+02 4.044524e+02  -0.934434396
## 15-19   3.901472e+02 3.889475e+02   1.199746458
## 20-24   3.715156e+02 3.668358e+02   4.679756175
## 25-29   3.517933e+02 3.433747e+02   8.418537502
## 30-34   3.350725e+02 3.236770e+02  11.395510908
## 35-39   3.196560e+02 3.084361e+02  11.219856223
## 40-44   3.038892e+02 2.936671e+02  10.222005694
## 45-49   2.855039e+02 2.762443e+02   9.259510559
## 50-54   2.639106e+02 2.548594e+02   9.051173440
## 55-59   2.385241e+02 2.294080e+02   9.116143171
## 60-64   2.099611e+02 2.005822e+02   9.378879900
## 65-69   1.769739e+02 1.678420e+02   9.131848597
## 70-74   1.402972e+02 1.327151e+02   7.582166138
## 75-79   9.969742e+01 9.527098e+01   4.426444631
## 80-84   6.147056e+01 6.014278e+01   1.327783839
## 85-89   3.116552e+01 3.091626e+01   0.249262028
## 90-94   1.235848e+01 1.235616e+01   0.002314198
## 95-99   3.228173e+00 3.197597e+00   0.030576126
## 100-104 4.687723e-01 4.550265e-01   0.013745838
## 105-109 3.667028e-02 3.425943e-02   0.002410852
## 110+    5.822578e-03 2.470463e-03   0.003352115
##         5.312373e+03 5.193418e+03 118.954451800
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.288541e+02 6.306460e+02 -1.791864e+00
## 1-4     6.667792e+02 6.605873e+02  6.191908e+00
## 5-9     6.180367e+02 6.336571e+02 -1.562046e+01
## 10-14   5.989305e+02 6.149495e+02 -1.601905e+01
## 15-19   5.883242e+02 6.045474e+02 -1.622323e+01
## 20-24   5.663719e+02 5.820842e+02 -1.571230e+01
## 25-29   5.284402e+02 5.445539e+02 -1.611370e+01
## 30-34   4.961633e+02 5.120736e+02 -1.591030e+01
## 35-39   4.645619e+02 4.799306e+02 -1.536864e+01
## 40-44   4.296944e+02 4.441654e+02 -1.447105e+01
## 45-49   3.898204e+02 4.020256e+02 -1.220522e+01
## 50-54   3.437115e+02 3.532945e+02 -9.583031e+00
## 55-59   2.912965e+02 2.979205e+02 -6.623917e+00
## 60-64   2.335753e+02 2.383392e+02 -4.763917e+00
## 65-69   1.734086e+02 1.764317e+02 -3.023166e+00
## 70-74   1.177275e+02 1.193255e+02 -1.597942e+00
```

```
## 75-79   7.069097e+01 7.092950e+01 -2.385335e-01
## 80-84   3.673754e+01 3.653155e+01  2.059942e-01
## 85-89   1.625377e+01 1.578827e+01  4.654948e-01
## 90-94   5.683582e+00 5.459889e+00  2.236930e-01
## 95-99   1.209757e+00 1.150808e+00  5.894916e-02
## 100-104 1.486950e-01 1.404794e-01  8.215649e-03
## 105-109 1.100558e-02 1.022130e-02  7.842851e-04
## 110+    2.868997e-03 1.001968e-03  1.867030e-03
##         7.266434e+03 7.424544e+03 -1.581094e+02
##
## $tot.tae.e0
##            [,1]    [,2]      [,3]
## Female 6355.000 6193.56 161.44038
## Male   8733.096 8757.98 -24.88418
```

```
# create a table with results for 2 components
capture.output(file = "../tables/ageCompTotC-2.txt", print.xtable(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        1.285808191  1.309623169 -0.0238149773
## 1-4      1.448076298  1.297656974  0.1504193237
## 5-9      0.833974494  0.739540137  0.0944343568
## 10-14    0.507410728  0.488058117  0.0193526114
## 15-19    0.649044412  0.704215885 -0.0551714728
## 20-24    0.803617792  0.856790241 -0.0531724492
## 25-29    0.798313035  0.847677326 -0.0493642910
## 30-34    0.768152161  0.801390372 -0.0332382111
## 35-39    0.841069874  0.835957359  0.0051125147
## 40-44    0.949653313  0.918270079  0.0313832338
## 45-49    1.105073480  1.088777855  0.0162956252
## 50-54    1.446536064  1.442450719  0.0040853452
## 55-59    1.888410119  1.908891692 -0.0204815725
## 60-64    2.464142167  2.509440007 -0.0452978397
## 65-69    3.070790072  3.094838412 -0.0240483395
## 70-74    4.089763507  4.004770837  0.0849926694
## 75-79    4.794356682  4.511095621  0.2832610601
## 80-84    4.506306943  4.364023689  0.1422832532
## 85-89    3.129507401  3.032863378  0.0966440228
## 90-94    1.474359746  1.497975829 -0.0236160834
## 95-99    0.373297609  0.398302792 -0.0250051834
## 100-104  0.046076748  0.051727293 -0.0056505451
## 105-109  0.002866161  0.003311803 -0.0004456413
##         37.276606997 36.707649587  0.5689574097
##         tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0        1.522463392 1.541778e+00 -1.931463e-02
## 1-4      2.007524189 1.546716e+00  4.608082e-01
## 5-9      0.905395356 8.656519e-01  3.974342e-02
## 10-14    0.519360559 4.864013e-01  3.295928e-02
## 15-19    0.880295096 8.426238e-01  3.767125e-02
## 20-24    1.678246787 1.619755e+00  5.849207e-02
## 25-29    1.572030317 1.504629e+00  6.740145e-02
## 30-34    1.512880998 1.457767e+00  5.511423e-02
## 35-39    1.675786492 1.621177e+00  5.460970e-02
## 40-44    1.939066821 1.895655e+00  4.341220e-02
## 45-49    2.321138426 2.316191e+00  4.947489e-03
## 50-54    2.867337843 2.923593e+00 -5.625533e-02
## 55-59    3.516199893 3.593629e+00 -7.742875e-02
## 60-64    4.306662489 4.439985e+00 -1.333230e-01
## 65-69    4.793216499 4.918807e+00 -1.255902e-01
## 70-74    4.934318236 5.135102e+00 -2.007837e-01
## 75-79    4.439414651 4.563438e+00 -1.240237e-01
## 80-84    3.117173782 3.282934e+00 -1.657598e-01
## 85-89    1.697061581 1.766515e+00 -6.945325e-02
## 90-94    0.669393339 7.064783e-01 -3.708496e-02
## 95-99    0.133539462 1.440494e-01 -1.050993e-02
## 100-104  0.014235445 1.559056e-02 -1.355116e-03
## 105-109  0.000874022 9.597548e-04 -8.573278e-05
##         47.023615674 4.718942e+01 -1.658088e-01
##
```

```
## $tae.e
##         tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.214281e+02 4.147746e+02  6.6534501569
## 1-4     4.588743e+02 4.594468e+02 -0.5724678570
## 5-9     4.230239e+02 4.257798e+02 -2.7558128299
## 10-14   4.018217e+02 4.044524e+02 -2.6307177179
## 15-19   3.883738e+02 3.889475e+02 -0.5736875035
## 20-24   3.693614e+02 3.668358e+02  2.5255434316
## 25-29   3.481562e+02 3.433747e+02  4.7814898259
## 30-34   3.296990e+02 3.236770e+02  6.0220860254
## 35-39   3.136621e+02 3.084361e+02  5.2259297440
## 40-44   2.981869e+02 2.936671e+02  4.5197854368
## 45-49   2.808038e+02 2.762443e+02  4.5594915240
## 50-54   2.602204e+02 2.548594e+02  5.3609591087
## 55-59   2.359192e+02 2.294080e+02  6.5111921459
## 60-64   2.081710e+02 2.005822e+02  7.5887648590
## 65-69   1.764887e+02 1.678420e+02  8.6466497382
## 70-74   1.406060e+02 1.327151e+02  7.8908802064
## 75-79   1.005492e+02 9.527098e+01  5.2781959467
## 80-84   6.241651e+01 6.014278e+01  2.2737379848
## 85-89   3.173310e+01 3.091626e+01  0.8168444363
## 90-94   1.245448e+01 1.235616e+01  0.0983177656
## 95-99   3.207747e+00 3.197597e+00  0.0101498818
## 100-104 4.571742e-01 4.550265e-01  0.0021477532
## 105-109 3.478501e-02 3.425943e-02  0.0005255724
## 110+    5.822578e-03 2.470463e-03  0.0033521155
##         5.265655e+03 5.193418e+03 72.2368077507
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.263207e+02 6.306460e+02 -4.325224e+00
## 1-4     6.641657e+02 6.605873e+02  3.578452e+00
## 5-9     6.160276e+02 6.336571e+02 -1.762948e+01
## 10-14   5.984997e+02 6.149495e+02 -1.644984e+01
## 15-19   5.881790e+02 6.045474e+02 -1.636841e+01
## 20-24   5.662205e+02 5.820842e+02 -1.586372e+01
## 25-29   5.284677e+02 5.445539e+02 -1.608621e+01
## 30-34   4.962605e+02 5.120736e+02 -1.581308e+01
## 35-39   4.644070e+02 4.799306e+02 -1.552354e+01
## 40-44   4.293127e+02 4.441654e+02 -1.485272e+01
## 45-49   3.892972e+02 4.020256e+02 -1.272843e+01
## 50-54   3.427845e+02 3.532945e+02 -1.051002e+01
## 55-59   2.897293e+02 2.979205e+02 -8.191171e+00
## 60-64   2.315641e+02 2.383392e+02 -6.775124e+00
## 65-69   1.711955e+02 1.764317e+02 -5.236269e+00
## 70-74   1.157736e+02 1.193255e+02 -3.551848e+00
## 75-79   6.919474e+01 7.092950e+01 -1.734759e+00
## 80-84   3.599701e+01 3.653155e+01 -5.345370e-01
## 85-89   1.584599e+01 1.578827e+01  5.772126e-02
## 90-94   5.561485e+00 5.459889e+00  1.015960e-01
## 95-99   1.192985e+00 1.150808e+00  4.217678e-02
## 100-104 1.478994e-01 1.404794e-01  7.420066e-03
## 105-109 1.102810e-02 1.022130e-02  8.067996e-04
## 110+    2.868997e-03 1.001968e-03  1.867030e-03
##         7.246159e+03 7.424544e+03 -1.783844e+02
##
```

```
## $tot.tae.e0
##            [,1]    [,2]      [,3]
## Female 6292.912 6193.56  99.35165
## Male   8697.914 8757.98 -60.06576
```

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

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

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

```
mod.1_0.f.4 <- svdMod(q1logit.f, Qlogit.f, N.4, S.4, 10,
    TRUE, adult.4, TRUE, smooth.4, C.4)
```

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

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

```
age.comps.4
```

```
## $tae.q
##         tae.comp.q.f   tae.lq.q.f   tae.diff.q.f
## 0        1.285808191  1.309623169 -0.0238149773
## 1-4      1.405605466  1.297656974  0.1079484913
## 5-9      0.774404676  0.739540137  0.0348645388
## 10-14    0.509533972  0.488058117  0.0214758555
## 15-19    0.631768929  0.704215885 -0.0724469560
## 20-24    0.776761164  0.856790241 -0.0800290770
## 25-29    0.767838972  0.847677326 -0.0798383535
## 30-34    0.759797072  0.801390372 -0.0415933000
```

```
## 35-39    0.835094853  0.835957359 -0.0008625060
## 40-44    0.949378962  0.918270079  0.0311088829
## 45-49    1.104900947  1.088777855  0.0161230918
## 50-54    1.446209914  1.442450719  0.0037591953
## 55-59    1.890110030  1.908891692 -0.0187816618
## 60-64    2.426462671  2.509440007 -0.0829773354
## 65-69    2.975484183  3.094838412 -0.1193542288
## 70-74    3.859023820  4.004770837 -0.1457470169
## 75-79    4.452347552  4.511095621 -0.0587480699
## 80-84    4.256483071  4.364023689 -0.1075406180
## 85-89    3.032195850  3.032863378 -0.0006675275
## 90-94    1.466339018  1.497975829 -0.0316368113
## 95-99    0.373584075  0.398302792 -0.0247187170
## 100-104  0.045990476  0.051727293 -0.0057368173
## 105-109  0.002843693  0.003311803 -0.0004681095
##         36.027967560 36.707649587 -0.6796820277
##         tae.comp.q.m   tae.lq.q.m  tae.diff.q.m
## 0       1.522463e+00 1.541778e+00 -0.0193146311
## 1-4     2.006411e+00 1.546716e+00  0.4596952560
## 5-9     8.794502e-01 8.656519e-01  0.0137982698
## 10-14   5.178406e-01 4.864013e-01  0.0314393559
## 15-19   8.854434e-01 8.426238e-01  0.0428195914
## 20-24   1.691406e+00 1.619755e+00  0.0716517282
## 25-29   1.567782e+00 1.504629e+00  0.0631531799
## 30-34   1.502594e+00 1.457767e+00  0.0448277045
## 35-39   1.674806e+00 1.621177e+00  0.0536296841
## 40-44   1.951835e+00 1.895655e+00  0.0561803576
## 45-49   2.351121e+00 2.316191e+00  0.0349304345
## 50-54   2.893663e+00 2.923593e+00 -0.0299296940
## 55-59   3.446941e+00 3.593629e+00 -0.1466873652
## 60-64   4.120316e+00 4.439985e+00 -0.3196690361
## 65-69   4.537801e+00 4.918807e+00 -0.3810053609
## 70-74   4.686842e+00 5.135102e+00 -0.4482603087
## 75-79   4.272008e+00 4.563438e+00 -0.2914298592
## 80-84   3.078585e+00 3.282934e+00 -0.2043487757
## 85-89   1.700815e+00 1.766515e+00 -0.0657000210
## 90-94   6.657873e-01 7.064783e-01 -0.0406910103
## 95-99   1.285520e-01 1.440494e-01 -0.0154973660
## 100-104 1.334220e-02 1.559056e-02 -0.0022483617
## 105-109 8.105134e-04 9.597548e-04 -0.0001492414
##         4.609662e+01 4.718942e+01 -1.0928054694
##
## $tae.e
##         tae.comp.e.f   tae.lq.e.f  tae.diff.e.f
## 0       4.108094e+02 4.147746e+02 -3.965212e+00
## 1-4     4.502406e+02 4.594468e+02 -9.206167e+00
## 5-9     4.164232e+02 4.257798e+02 -9.356524e+00
## 10-14   3.954913e+02 4.044524e+02 -8.961155e+00
## 15-19   3.815325e+02 3.889475e+02 -7.414935e+00
## 20-24   3.618813e+02 3.668358e+02 -4.954584e+00
## 25-29   3.401536e+02 3.433747e+02 -3.221157e+00
## 30-34   3.209064e+02 3.236770e+02 -2.770591e+00
## 35-39   3.042722e+02 3.084361e+02 -4.163924e+00
## 40-44   2.879651e+02 2.936671e+02 -5.702007e+00
```

```
## 45-49   2.700467e+02 2.762443e+02 -6.197647e+00
## 50-54   2.487578e+02 2.548594e+02 -6.101545e+00
## 55-59   2.240441e+02 2.294080e+02 -5.363838e+00
## 60-64   1.956326e+02 2.005822e+02 -4.949684e+00
## 65-69   1.648870e+02 1.678420e+02 -2.955035e+00
## 70-74   1.308905e+02 1.327151e+02 -1.824629e+00
## 75-79   9.421929e+01 9.527098e+01 -1.051684e+00
## 80-84   5.958329e+01 6.014278e+01 -5.594867e-01
## 85-89   3.100424e+01 3.091626e+01  8.798583e-02
## 90-94   1.240199e+01 1.235616e+01  4.582874e-02
## 95-99   3.209910e+00 3.197597e+00  1.231316e-02
## 100-104 4.565116e-01 4.550265e-01  1.485138e-03
## 105-109 3.451248e-02 3.425943e-02  2.530459e-04
## 110+    5.822578e-03 2.470463e-03  3.352115e-03
##         5.104850e+03 5.193418e+03 -8.856859e+01
##         tae.comp.e.m   tae.lq.e.m  tae.diff.e.m
## 0       6.189620e+02 6.306460e+02 -1.168393e+01
## 1-4     6.577537e+02 6.605873e+02 -2.833600e+00
## 5-9     6.085742e+02 6.336571e+02 -2.508288e+01
## 10-14   5.910655e+02 6.149495e+02 -2.388406e+01
## 15-19   5.808574e+02 6.045474e+02 -2.368999e+01
## 20-24   5.590702e+02 5.820842e+02 -2.301396e+01
## 25-29   5.200264e+02 5.445539e+02 -2.452747e+01
## 30-34   4.849374e+02 5.120736e+02 -2.713621e+01
## 35-39   4.506845e+02 4.799306e+02 -2.924610e+01
## 40-44   4.143479e+02 4.441654e+02 -2.981748e+01
## 45-49   3.740160e+02 4.020256e+02 -2.800958e+01
## 50-54   3.277359e+02 3.532945e+02 -2.555868e+01
## 55-59   2.759549e+02 2.979205e+02 -2.196555e+01
## 60-64   2.203616e+02 2.383392e+02 -1.797765e+01
## 65-69   1.637741e+02 1.764317e+02 -1.265764e+01
## 70-74   1.119345e+02 1.193255e+02 -7.391022e+00
## 75-79   6.791521e+01 7.092950e+01 -3.014291e+00
## 80-84   3.590373e+01 3.653155e+01 -6.278165e-01
## 85-89   1.588130e+01 1.578827e+01  9.303009e-02
## 90-94   5.518531e+00 5.459889e+00  5.864214e-02
## 95-99   1.154423e+00 1.150808e+00  3.615189e-03
## 100-104 1.399408e-01 1.404794e-01 -5.385160e-04
## 105-109 1.030230e-02 1.022130e-02  8.100027e-05
## 110+    2.868997e-03 1.001968e-03  1.867030e-03
##         7.086583e+03 7.424544e+03 -3.379612e+02
##
## $tot.tae.e0
##             [,1]    [,2]       [,3]
## Female 6134.350 6193.56  -59.20994
## Male   8595.722 8757.98 -162.25837
```

```
# 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]
```

```r
mex.m.q <- standardFiveYear(mex.m.q)[1:18]

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

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

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

# child and adult Mx

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

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

# have a look
mex.f.Q
```

```
## [1] 0.05336201 0.13518150
```

```
mex.m.Q
```

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

```
rsa.f.Q
```

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

```
rsa.m.Q
```

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

```
# Predictions

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

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

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

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

```
### predictions with child and adult

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

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

```r
rsa.m.p5.ca <- standardFiveYear(expit(rsa.m.p.ca[, 1]))

# predictions with Log-Quad

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

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

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

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

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

### Plots, logit scale

### Mexico female child and adult only
plot(mex.f.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 -3.15840049       Mexico Female
## 2     1 -4.75010506       Mexico Female
## 3     5 -5.54514497       Mexico Female
## 4    10 -5.90047153       Mexico Female
## 5    15 -5.51757062       Mexico Female
## 6    20 -5.31771387       Mexico Female
## 7    25 -5.13573474       Mexico Female
## 8    30 -4.87492640       Mexico Female
## 9    35 -4.53510615       Mexico Female
## 10   40 -4.17305212       Mexico Female
## 11   45 -3.80190195       Mexico Female
## 12   50 -3.39906683       Mexico Female
```

```
## 13   55 -2.98999854       Mexico Female
## 14   60 -2.50951800       Mexico Female
## 15   65 -1.99926277       Mexico Female
## 16   70 -1.41427648       Mexico Female
## 17   75 -0.80067082       Mexico Female
## 18   80 -0.15839475       Mexico Female
## 19    0 -2.96380393       Mexico   Male
## 20    1 -4.43705975       Mexico   Male
## 21    5 -5.14818928       Mexico   Male
## 22   10 -5.39399881       Mexico   Male
## 23   15 -4.71576987       Mexico   Male
## 24   20 -4.38881740       Mexico   Male
## 25   25 -4.38598993       Mexico   Male
## 26   30 -4.27276668       Mexico   Male
## 27   35 -4.02682889       Mexico   Male
## 28   40 -3.68549466       Mexico   Male
## 29   45 -3.29474684       Mexico   Male
## 30   50 -2.87371372       Mexico   Male
## 31   55 -2.44874386       Mexico   Male
## 32   60 -1.98870195       Mexico   Male
## 33   65 -1.52112879       Mexico   Male
## 34   70 -1.00228620       Mexico   Male
## 35   75 -0.44950413       Mexico   Male
## 36   80  0.16057489       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 -2.92286464 South Africa Female
## 74    1 -3.94001820 South Africa Female
## 75    5 -5.68994613 South Africa Female
## 76   10 -5.86930272 South Africa Female
## 77   15 -4.34277224 South Africa Female
## 78   20 -3.97842681 South Africa Female
## 79   25 -3.70866357 South Africa Female
## 80   30 -3.47328726 South Africa Female
## 81   35 -3.16479440 South Africa Female
## 82   40 -2.92672816 South Africa Female
## 83   45 -2.61782732 South Africa Female
## 84   50 -2.29711356 South Africa Female
## 85   55 -1.96394228 South Africa Female
## 86   60 -1.58089976 South Africa Female
## 87   65 -1.22238041 South Africa Female
## 88   70 -0.79991350 South Africa Female
## 89   75 -0.37225634 South Africa Female
## 90   80  0.12045668 South Africa Female
## 91    0 -2.71449321 South Africa   Male
## 92    1 -3.64257827 South Africa   Male
## 93    5 -4.66587492 South Africa   Male
## 94   10 -4.80731817 South Africa   Male
## 95   15 -3.60968229 South Africa   Male
## 96   20 -2.94164554 South Africa   Male
## 97   25 -2.82535790 South Africa   Male
## 98   30 -2.75596246 South Africa   Male
## 99   35 -2.63361974 South Africa   Male
## 100  40 -2.46573490 South Africa   Male
## 101  45 -2.26323428 South Africa   Male
## 102  50 -2.01570814 South Africa   Male
## 103  55 -1.74905626 South Africa   Male
## 104  60 -1.38535229 South Africa   Male
## 105  65 -1.01668881 South Africa   Male
## 106  70 -0.57988938 South Africa   Male
## 107  75 -0.08694849 South Africa   Male
## 108  80  0.48876482 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
```

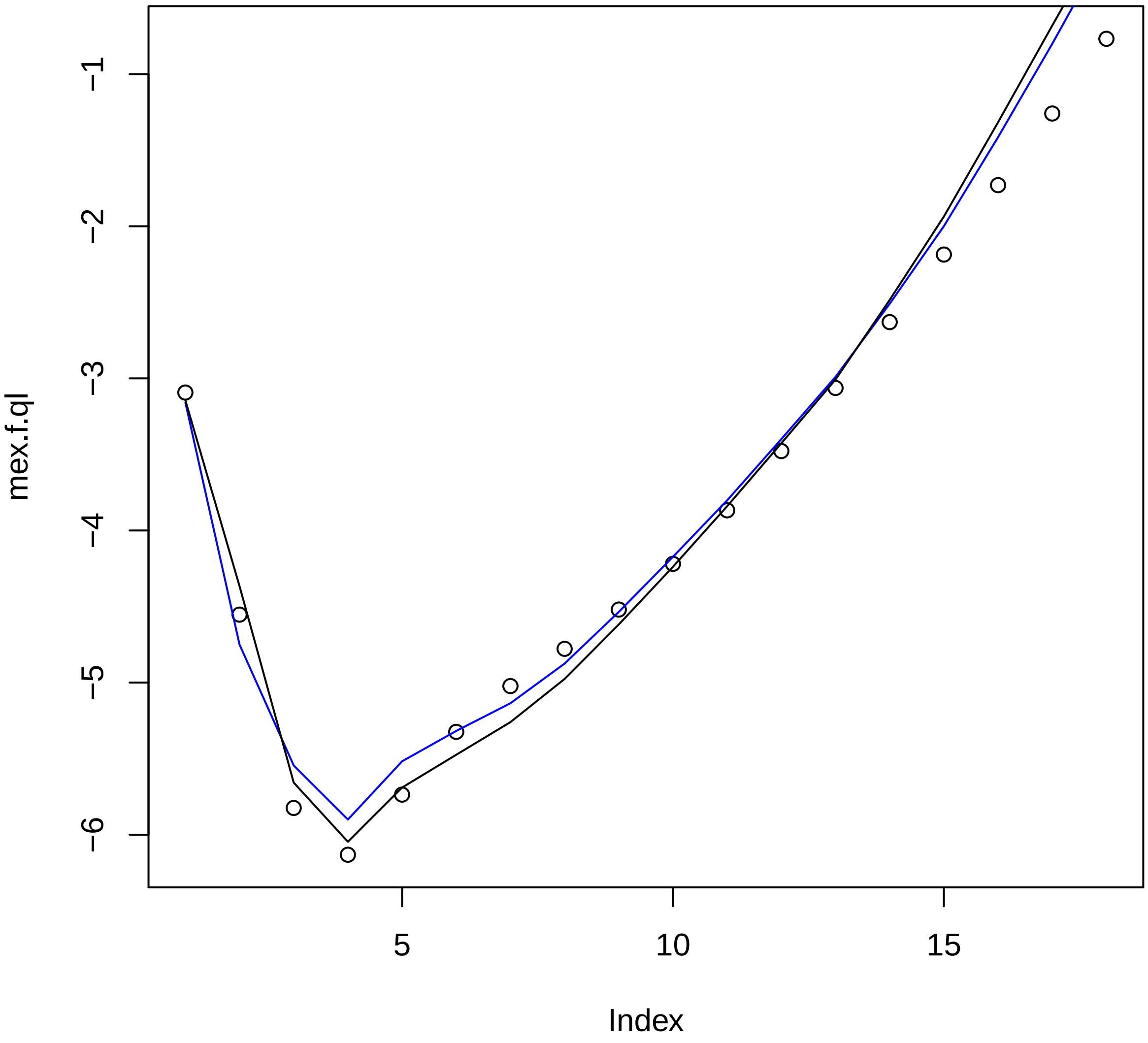
```
## 121  55 -2.17063112 South Africa Female
## 122  60 -1.87705073 South Africa Female
## 123  65 -1.47428624 South Africa Female
## 124  70 -1.02804138 South Africa Female
## 125  75 -0.51873534 South Africa Female
## 126  80  0.04001734 South Africa Female
## 127   0 -2.69594596 South Africa   Male
## 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
```

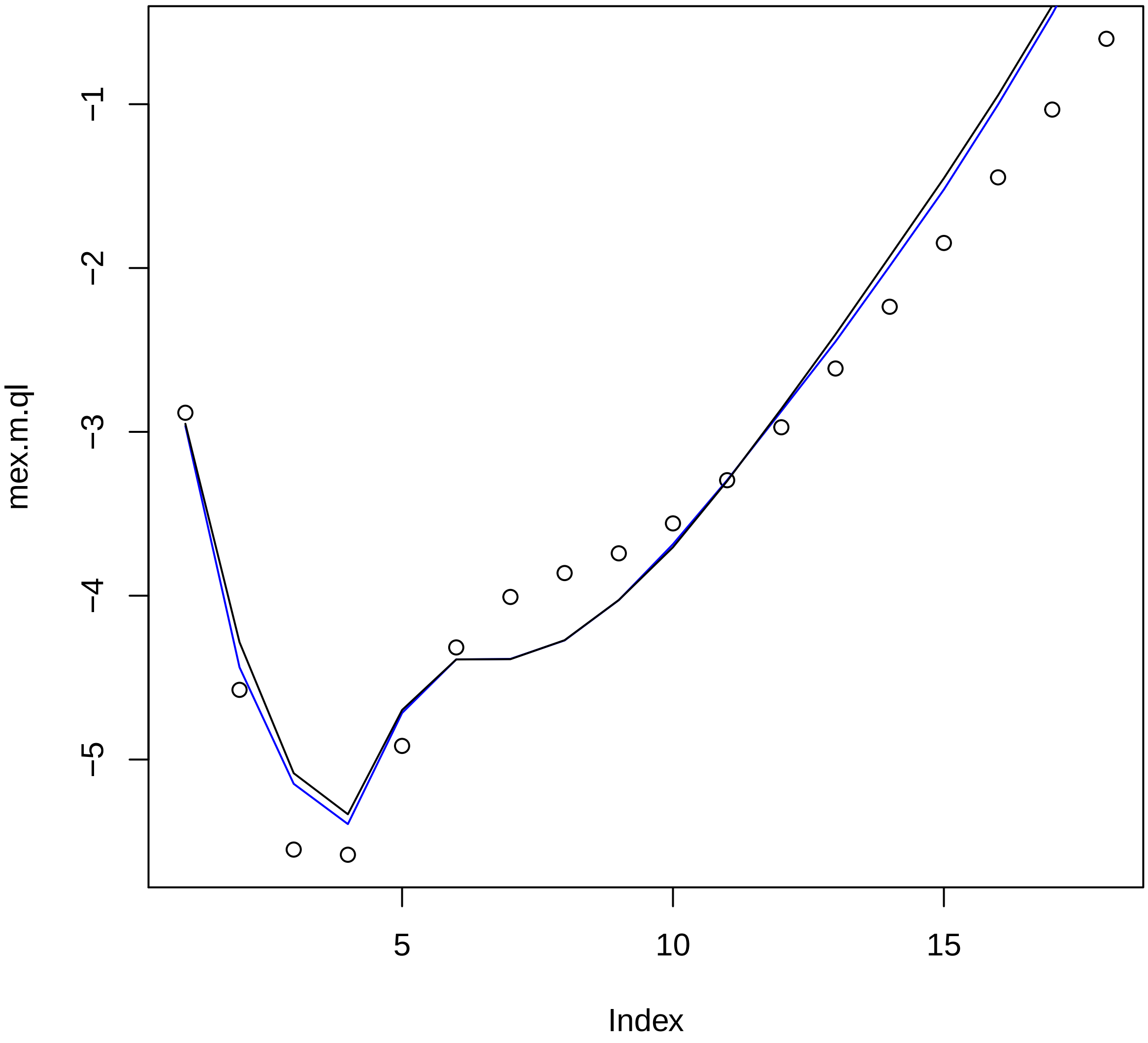
```
## 84  Predicted by SVD-Comp
## 85  Predicted by SVD-Comp
## 86  Predicted by SVD-Comp
## 87  Predicted by SVD-Comp
## 88  Predicted by SVD-Comp
## 89  Predicted by SVD-Comp
## 90  Predicted by SVD-Comp
## 91  Predicted by SVD-Comp
## 92  Predicted by SVD-Comp
## 93  Predicted by SVD-Comp
## 94  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
```

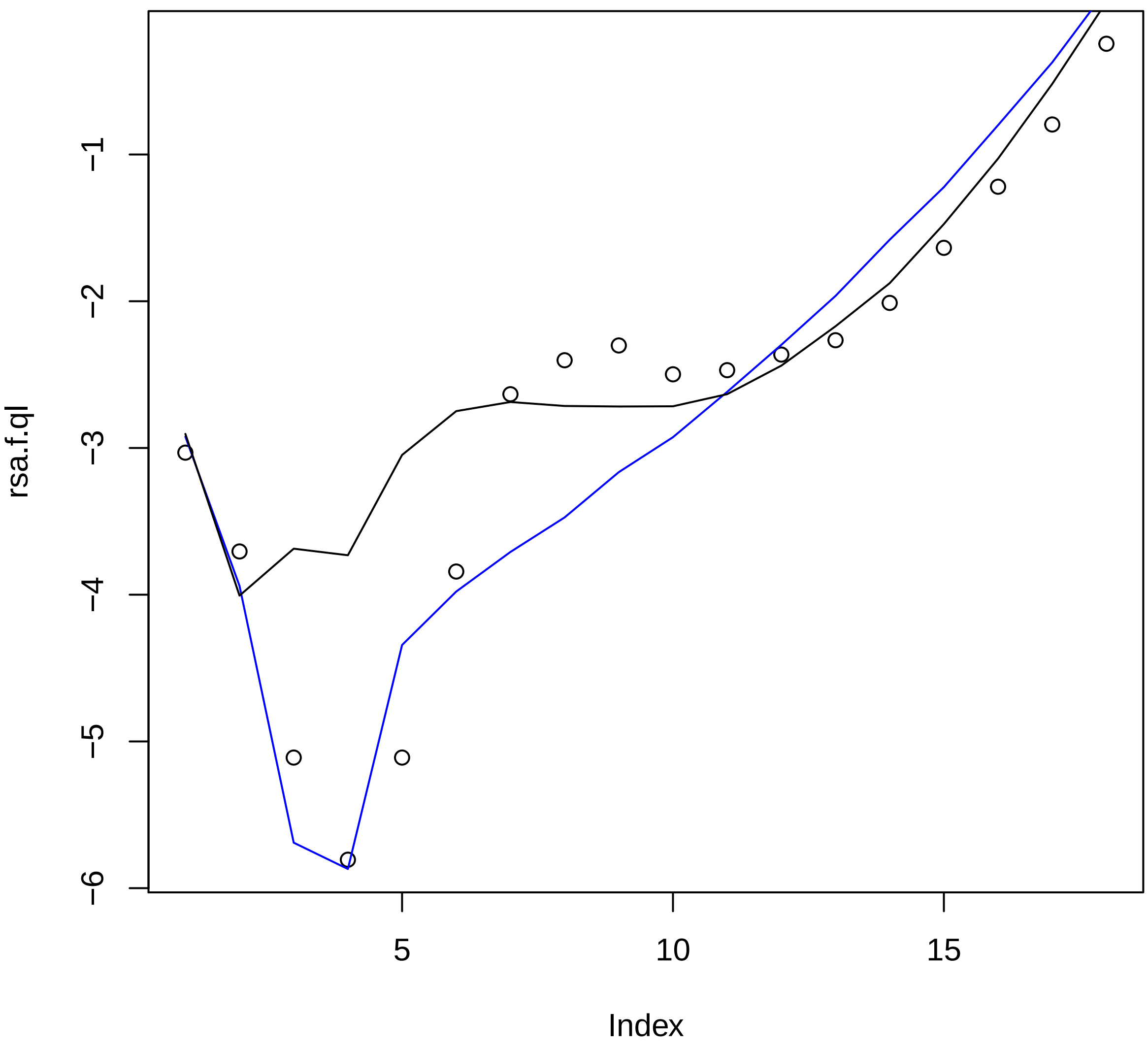
```
## 138 Predicted by Log-Quad
## 139 Predicted by Log-Quad
## 140 Predicted by Log-Quad
## 141 Predicted by Log-Quad
## 142 Predicted by Log-Quad
## 143 Predicted by Log-Quad
## 144 Predicted by Log-Quad
```

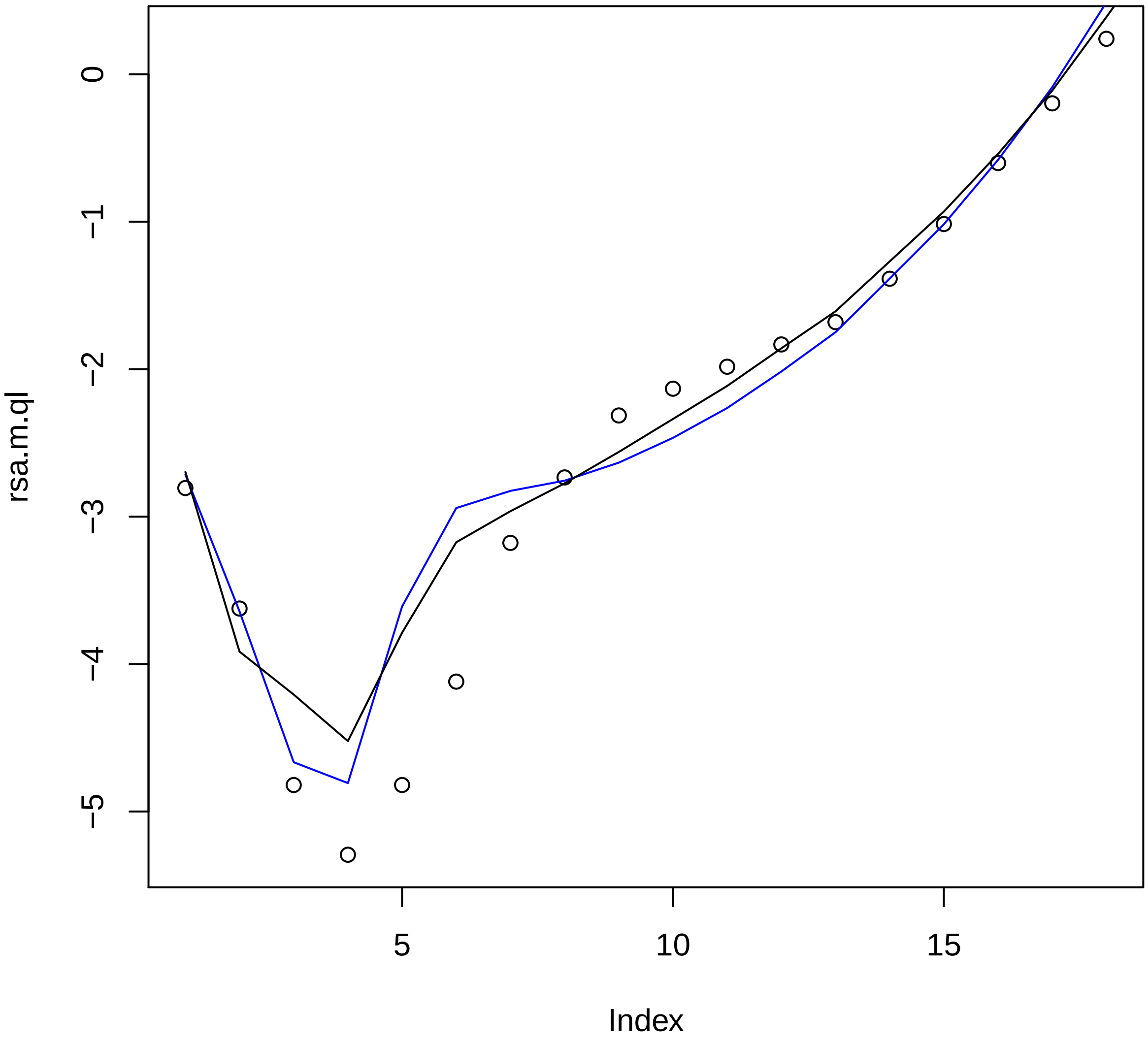
```
# plot data and predictions
ggplot(data = q.logit.data, aes(x = Age, y = Value, colour = Source)) +
    geom_line(data = q.logit.pred, aes(x = Age, y = Value,
        colour = Source), size = 1) + scale_x_continuous(breaks = c(0,
    1, seq(5, 80, 5)), labels = c("0,1-4", "", paste(seq(5,
    80, 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(0.15,
    0.91)) + 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(0.15,
    0.91)) + theme(legend.position = "bottom", legend.box = "horizontal") +
    theme(strip.text = element_text(face = "bold")) + scale_colour_grey(start = 0,
    end = 0.8) + ggsave("../figures/fig7-BW.pdf", width = 6.5,
    height = 6.5, units = c("in"))
```

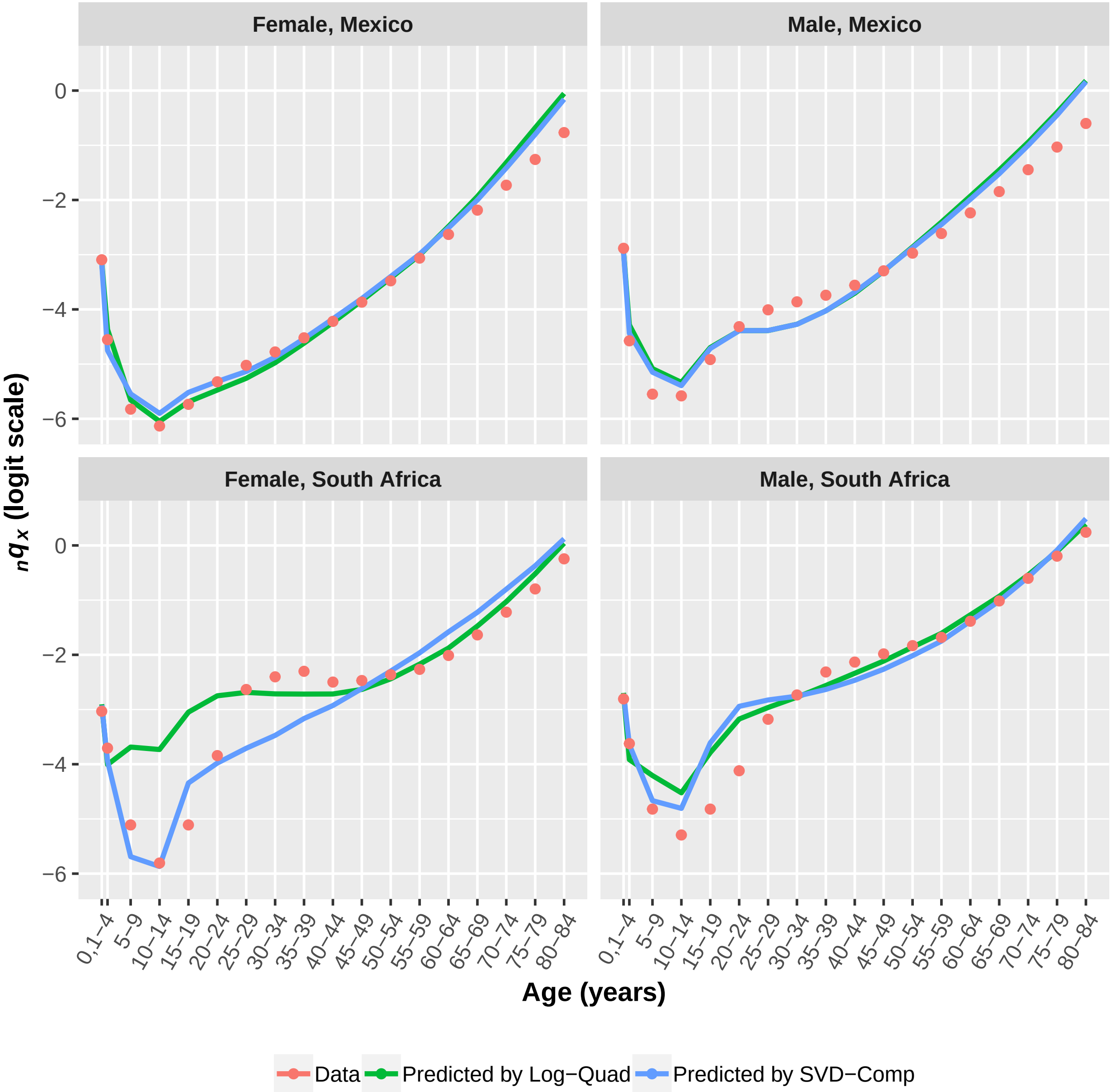


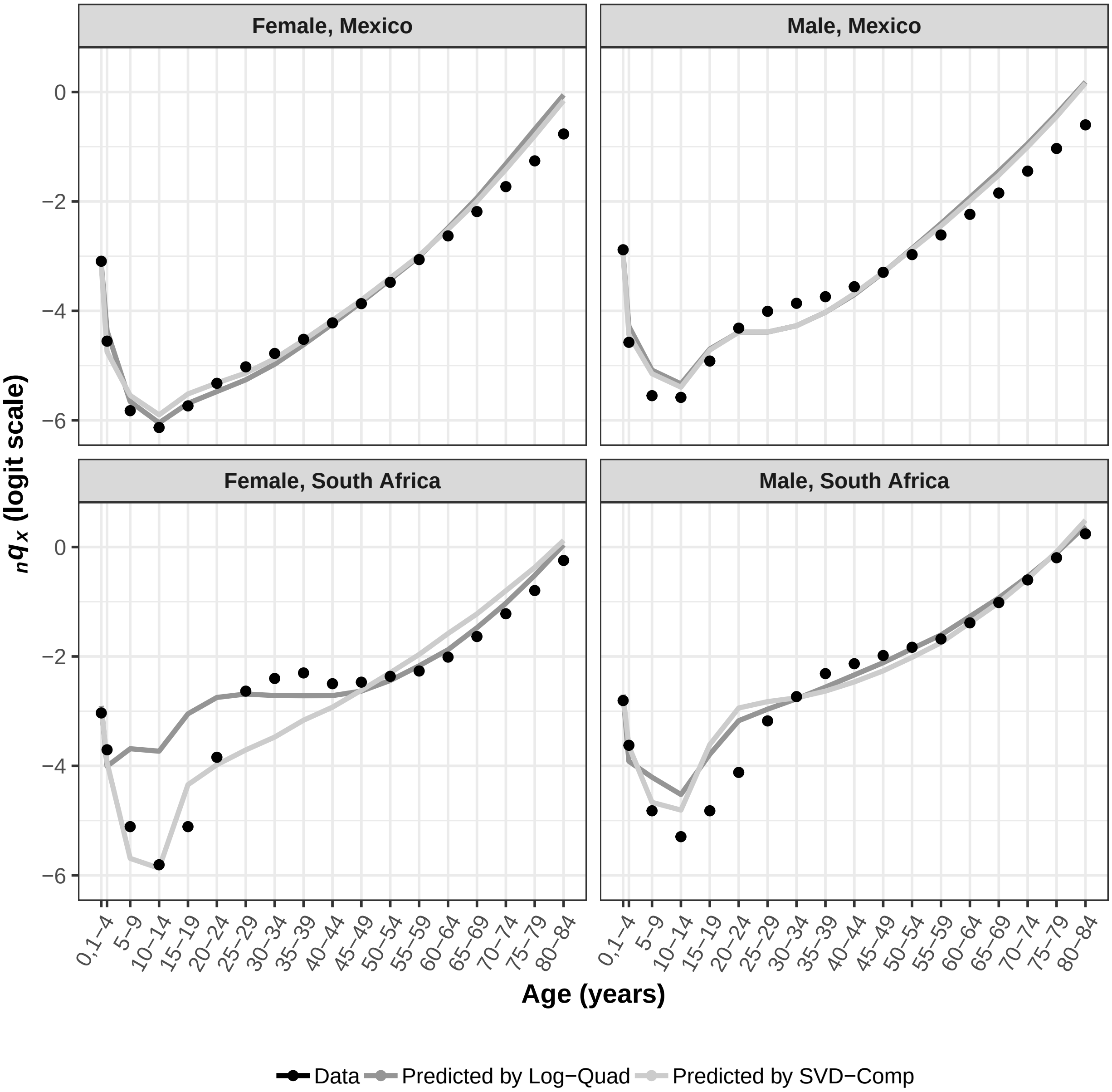
mex.f.ql
−1
−2
−3
−4
−5
−6
5
10
15
Index


mex.m.ql
−1
−2
−3
−4
−5
5
10
15
Index


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


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


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")

# 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 -934.81217 -1078.100301 -1121.21604 -1145.2916297
## 2  -32.99025   -69.490925   -67.86207   -63.7837818
## 3  -11.69773    -8.068989    -3.52183     0.9974696
## 4   13.71181    17.132887    15.05006    20.7077312
##              4             5            6           7
## 1 -1160.685642 -1173.2862553 -1182.517755 -1193.76699
## 2   -58.178780   -56.6680603   -53.996840   -53.36262
## 3    -5.378499     0.2667628     8.425027    14.22613
## 4    16.321638    14.9561594    15.176928    13.44546
##             8           9           10           11
## 1 -1201.25817 -1206.91829 -1209.423345 -1210.303262
## 2   -48.84598   -46.73611   -45.755033   -46.033797
## 3    21.30969    19.06144    23.700518    19.916801
## 4    14.18258    10.83815     5.474541    -1.091755
##              12           13           14           15
## 1 -1206.6571539 -1200.033331 -1189.916024 -1178.670393
## 2   -42.3177904   -40.445241   -35.589607   -30.443038
## 3    15.4238493    13.901605     8.618961     8.440697
## 4    -0.4554301    -3.078721    -8.145599   -15.368778
##              16           17            18
## 1 -1167.2730792 -1160.905888 -1154.0006870
## 2   -25.7805487   -24.877492   -21.4763153
## 3     0.8121733     0.419798    -0.5214893
## 4   -13.2525635   -10.980000   -14.2932031
##             19            20           21
## 1 -1151.148324 -1148.9962600 -1148.103284
## 2   -22.068752   -23.5686739   -24.816738
## 3    -2.141245     0.8982334     1.858938
## 4   -14.187194   -12.8942770   -14.460371
##              22            23           24
## 1 -1146.3264382 -1145.4395426 -1143.612718
## 2   -26.4021749   -27.8075888   -27.735071
## 3    -0.3666085    -0.0724522    -1.069955
## 4   -13.5375389   -14.5051234   -13.787198
##              25           26           27           28
```

```
## 1 -1140.9366149 -1138.436012 -1136.115358 -1134.004769
## 2   -25.7864157   -24.480313   -23.619273   -24.710907
## 3    -0.6622387    -3.263604    -3.563435    -3.267511
## 4   -12.4497726   -11.002969    -8.983937   -12.587832
##             29           30           31           32
## 1 -1129.114003 -1125.269436 -1124.115213 -1119.206619
## 2   -18.996073   -18.274580   -17.806027   -16.925832
## 3    -6.374302    -5.008698    -3.315137    -4.515220
## 4    -5.784267    -5.499891    -7.849819    -8.234091
##             33           34           35           36
## 1 -1115.835515 -1110.861879 -1105.619430 -1102.105419
## 2   -15.818446   -12.585437    -9.143346    -8.588348
## 3    -2.707198    -7.266547    -7.227161    -6.115746
## 4   -10.073368    -7.927899    -5.249192    -5.615115
##             37           38           39            40
## 1 -1098.488362 -1091.854493 -1088.599365 -1083.0097720
## 2    -7.286434    -2.879493    -2.964686    -0.7508958
## 3    -5.223041    -8.734254    -7.505259    -7.2698574
## 4    -5.595850    -1.871233    -3.847943     1.5920128
##             41            42            43
## 1 -1080.116203 -1073.0626654 -1069.1299698
## 2     1.694397     4.6252393     7.3838375
## 3    -5.540805    -7.8055573    -7.2161259
## 4    -4.871465    -0.8718879    -0.7767959
##             44            45           46
## 1 -1064.541365 -1058.8514420 -1054.889348
## 2     9.283018    11.1644946    12.109123
## 3    -6.800455    -8.6333907    -7.465745
## 4    -2.855260    -0.7720526    -2.393644
##              47            48            49
## 1 -1049.4542090 -1043.6547412 -1038.4110186
## 2    15.5908287    16.3279039    18.5009792
## 3    -7.5838034    -8.0044147    -7.6368042
## 4     0.1816466     0.9119338     0.6607888
##              50           51           52
## 1 -1031.9139423 -1029.035524 -1021.971725
## 2    17.7204241    19.212011    18.838770
## 3    -9.1163164    -6.450327    -8.249205
## 4     0.2965171    -0.476073     1.553808
##              53           54           55           56
## 1 -1017.7797202 -1012.146580 -1007.318725 -1001.531402
## 2    19.6590513    20.243928    21.085839    21.654670
## 3    -7.2491736    -7.832601    -7.392123    -8.324445
## 4     0.5107027     1.356850     1.789300     2.007953
##            57          58          59          60
## 1 -996.652526 -990.375377 -985.025744 -977.255251
## 2   22.451681   21.817291   21.749096   20.038136
## 3   -7.515985   -7.510561   -6.747312   -8.076162
## 4    2.270948    2.482399    2.128034    3.456917
##            61          62          63          64
## 1 -973.208784 -965.410193 -959.818653 -953.261444
## 2   22.113236   20.869301   20.506518   20.535917
## 3   -6.899889   -8.005847   -5.715749   -6.667228
## 4    3.077265    3.937888    3.805540    4.441357
```

```
##            65          66          67          68
## 1 -946.294415 -941.186624 -934.309492 -927.107105
## 2   20.510389   20.985521   21.504532   20.829420
## 3   -7.376701   -5.357044   -5.349904   -5.715573
## 4    4.646478    4.700423    4.802689    5.381330
##            69          70          71          72
## 1 -920.772636 -911.668892 -907.156339 -897.890510
## 2   21.359732   19.736063   21.176998   20.171163
## 3   -4.202347   -6.183550   -2.999161   -4.232993
## 4    4.931949    6.168962    5.366502    6.342689
##            73          74          75          76
## 1 -891.034010 -883.263194 -875.739129 -868.738524
## 2   20.725812   20.991569   20.879042   21.699420
## 3   -3.327413   -4.051487   -3.342029   -2.697316
## 4    6.281325    6.467125    6.676577    6.746863
##              77          78           79          80
## 1 -862.35064847 -853.669232 -846.9606410 -838.549113
## 2   22.38557812   22.297605   23.2402296   23.682562
## 3    0.07064408   -2.099756   -0.5147411   -1.819591
## 4    6.19439775    6.567227    5.7278639    6.121445
##              81            82           83          84
## 1 -833.41544021 -824.38599294 -817.1344828 -809.213014
## 2   25.78987663   25.26542519   25.9917358   26.078457
## 3   -0.09811858   -0.02852641    0.9519132    1.226091
## 4    4.91597936    5.26318351    4.8370870    4.791055
##            85          86          87          88
## 1 -802.531725 -795.342441 -788.901990 -782.192583
## 2   27.023658   28.099464   29.077736   29.707654
## 3    2.120207    2.952962    3.933364    4.559833
## 4    4.273231    4.053887    3.331666    2.951848
##            89          90          91          92
## 1 -775.803406 -768.100896 -763.489981 -756.266707
## 2   31.065477   30.577440   32.806112   32.865273
## 3    5.703415    5.383927    7.183681    7.389361
## 4    2.334745    2.385219    1.189876    1.191874
##             93           94           95            96
## 1 -750.2708398 -744.0903099 -737.3014974 -731.38674180
## 2   33.7329875   34.2424604   34.4189738   35.07435303
## 3    8.0981488    8.5103744    9.0876925    9.65858311
## 4    0.7817512    0.5256476    0.3003499   -0.05316081
##             97           98          99         100
## 1 -725.6289774 -720.0377564 -714.622373 -709.391720
## 2   35.7125796   36.3304793   36.924762   37.492161
## 3   10.2023140   10.7170435   11.200832   11.652157
## 4   -0.3979676   -0.7325309   -1.055445   -1.365091
##           101         102         103         104
## 1 -704.353888 -699.516069 -694.884620 -690.464524
## 2   38.029442   38.533610   39.001871   39.431837
## 3   12.069480   12.451542   12.797393   13.106330
## 4   -1.660054   -1.938819   -2.200237   -2.443102
##           105         106         107         108
## 1 -686.259436 -682.271699 -678.501955 -674.949544
## 2   39.821570   40.169652   40.475220   40.738110
## 3   13.378040   13.612565   13.810361   13.972177
```

```
## 4   -2.666487   -2.869801   -3.052633   -3.214914
##           109
## 1 -671.612112
## 2   40.958623
## 3   14.099196
## 4   -3.356866
##
## $female$components.sm
##            0            1            2           3
## 1 -934.81217 -1078.100301 -1121.216041 -1145.29163
## 2  -32.99025   -58.988543   -63.952014   -62.92926
## 3  -11.69773    -8.068989    -4.420903    -2.49848
## 4   13.71181    17.132887    15.050060    16.67386
##               4           5            6           7
## 1 -1160.6856424 -1173.28626 -1182.517755 -1193.76699
## 2   -59.5927582   -56.69477   -54.442565   -52.17201
## 3    -0.3463088     3.08090     7.950616    13.09219
## 4    16.3274600    15.60045    14.591220    13.23684
##             8            9           10           11
## 1 -1201.25817 -1206.918292 -1209.423345 -1210.303262
## 2   -49.61767   -47.456878   -46.052637   -44.652947
## 3    17.13726    19.409608    19.819670    18.526995
## 4    11.30154     8.633641     5.395408     1.958207
##              12           13           14           15
## 1 -1206.657154 -1200.033331 -1189.916024 -1178.670393
## 2   -42.449133   -39.323477   -35.295935   -30.906156
## 3    16.027441    12.975990     9.754004     6.510828
## 4    -1.423945    -4.698681    -7.738761   -10.204250
##              16           17            18
## 1 -1167.273079 -1160.905888 -1154.0006870
## 2   -27.183741   -24.572774   -23.0144380
## 3     3.539144     1.313592     0.1047249
## 4   -11.835304   -12.732258   -13.2098132
##               19            20            21
## 1 -1151.1483238 -1148.9962600 -1148.1032840
## 2   -22.7652227   -23.6169447   -24.9114030
## 3    -0.1670923     0.1023427     0.3151709
## 4   -13.4989272   -13.6821016   -13.7728009
##               22            23            24
## 1 -1146.3264382 -1145.4395426 -1143.6127180
## 2   -26.2018134   -27.0201361   -26.8890087
## 3     0.1386775    -0.3237634    -0.9224932
## 4   -13.7366688   -13.4918093   -12.9700040
##              25           26           27           28
## 1 -1140.936615 -1138.436012 -1136.115358 -1134.004769
## 2   -25.921240   -24.812255   -23.899539   -22.548987
## 3    -1.663064    -2.517414    -3.361038    -4.086176
## 4   -12.192964   -11.261069   -10.260475    -9.250140
##              29           30           31           32
## 1 -1129.114003 -1125.269436 -1124.115213 -1119.206619
## 2   -20.470269   -18.724776   -17.664982   -16.604355
## 3    -4.536586    -4.567794    -4.368567    -4.383002
## 4    -8.373666    -7.854489    -7.759797    -7.845213
##              33           34           35           36
```

```
## 1 -1115.835515 -1110.861879 -1105.619430 -1102.105419
## 2   -14.927856   -12.556837   -10.193297    -8.303077
## 3    -4.875847    -5.643484    -6.220978    -6.494627
## 4    -7.733662    -7.228478    -6.403372    -5.433533
##             37           38           39            40
## 1 -1098.488362 -1091.854493 -1088.599365 -1083.0097720
## 2    -6.355594    -4.227115    -2.369339    -0.4480196
## 3    -6.759788    -7.088604    -7.215446    -7.0831098
## 4    -4.435841    -3.491861    -2.717247    -2.2050497
##             41           42           43           44
## 1 -1080.116203 -1073.062665 -1069.129970 -1064.541365
## 2     1.912682     4.522952     7.008868     9.134674
## 3    -6.967912    -7.046700    -7.242703    -7.460153
## 4    -1.931044    -1.785256    -1.674898    -1.536339
##             45            46            47
## 1 -1058.851442 -1054.8893478 -1049.4542090
## 2    10.971973    12.8304163    14.7837874
## 3    -7.642565    -7.7363009    -7.7843457
## 4    -1.294883    -0.9048838    -0.4255002
##              48            49            50
## 1 -1.043655e+03 -1038.4110186 -1031.9139423
## 2  1.644659e+01    17.5767207    18.2414562
## 3 -7.857219e+00    -7.9175378    -7.8723615
## 4 -3.264720e-04     0.2703773     0.4182407
##              51            52           53
## 1 -1029.0355237 -1021.9717255 -1017.779720
## 2    18.7282635    19.1631351    19.673866
## 3    -7.7418830    -7.6444817    -7.625339
## 4     0.5552834     0.7574517     1.026850
##             54           55           56          57
## 1 -1012.146580 -1007.318725 -1001.531402 -996.652526
## 2    20.316425    21.001157    21.600090   21.916418
## 3    -7.658135    -7.709217    -7.712358   -7.616769
## 4     1.332252     1.641254     1.928165    2.186065
##            58          59          60          61
## 1 -990.375377 -985.025744 -977.255251 -973.208784
## 2   21.809150   21.421424   21.158835   21.159859
## 3   -7.476133   -7.392309   -7.359304   -7.265059
## 4    2.435812    2.708919    3.017273    3.345914
##            62          63          64          65
## 1 -965.410193 -959.818653 -953.261444 -946.294415
## 2   21.002902   20.716409   20.602375   20.708508
## 3   -7.049742   -6.794782   -6.581802   -6.330000
## 4    3.672025    3.980574    4.261861    4.510371
##            66          67          68          69
## 1 -941.186624 -934.309492 -927.107105 -920.772636
## 2   20.951169   21.111003   21.051087   20.825711
## 3   -5.965430   -5.587186   -5.282056   -5.004971
## 4    4.731291    4.940974    5.155349    5.380049
##            70          71          72          73
## 1 -911.668892 -907.156339 -897.890510 -891.034010
## 2   20.606057   20.567007   20.591332   20.700780
## 3   -4.663557   -4.264071   -3.930728   -3.703345
## 4    5.610771    5.839908    6.057889    6.249161
```

```
##            74          75          76          77
## 1 -883.263194 -875.739129 -868.738524 -862.350648
## 2   20.907890   21.206925   21.665073   22.153868
## 3   -3.435541   -2.951175   -2.272745   -1.659273
## 4    6.391252    6.461607    6.446782    6.347949
##            78          79           80           81
## 1 -853.669232 -846.960641 -838.5491133 -833.4154402
## 2   22.620847   23.245862   24.1050952   24.9611950
## 3   -1.300201   -1.095876   -0.8276495   -0.3898574
## 4    6.176367    5.946876    5.6793921    5.4006610
##             82          83          84          85
## 1 -824.3859929 -817.134483 -809.213014 -802.531725
## 2   25.5075344   25.895956   26.396231   27.137971
## 3    0.1765126    0.797772    1.462737    2.196057
## 4    5.1303203    4.863447    4.573529    4.234737
##            86          87          88          89
## 1 -795.342441 -788.901990 -782.192583 -775.803406
## 2   28.054617   28.975826   29.833036   30.594145
## 3    2.994833    3.810599    4.588651    5.311471
## 4    3.839939    3.402902    2.946287    2.487028
##            90          91          92           93
## 1 -768.100896 -763.489981 -756.266707 -750.2708398
## 2   31.345151   32.211863   32.991217   33.6125821
## 3    6.015441    6.724472    7.397274    7.9997300
## 4    2.034632    1.602465    1.208274    0.8587744
##             94           95           96           97
## 1 -744.0903099 -737.3014974 -731.3867418 -725.6289774
## 2   34.1233626   34.5807549   35.1053549   35.7050205
## 3    8.5547933    9.0979080    9.6403985   10.1729172
## 4    0.5414416    0.2349479   -0.0761416   -0.3952438
##             98          99         100         101
## 1 -720.0377564 -714.622373 -709.391720 -704.353888
## 2   36.3161628   36.908158   37.473582   38.009013
## 3   10.6835970   11.165305   11.615002   12.030968
## 4   -0.7170226   -1.033864   -1.340139   -1.632542
##           102         103         104         105
## 1 -699.516069 -694.884620 -690.464524 -686.259436
## 2   38.511479   38.978272   39.407050   39.795913
## 3   12.411983   12.757105   13.065700   13.336686
## 4   -1.909176   -2.168477   -2.408626   -2.626278
##           106         107         108         109
## 1 -682.271699 -678.501955 -674.949544 -671.612112
## 2   40.142924   40.441899   40.672841   40.818386
## 3   13.567416   13.751737   13.884708   13.970867
## 4   -2.815687   -2.970451   -3.087959   -3.171742
##
## $female$aml
##
## Call:
## lm(formula = aml ~ cm + cml + cmls + cmlc)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##     5.92112    -10.79701      4.00517      0.69463
```

```
##        cmlc
##     0.04597
##
##
## $female$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   6.260e-03    1.633e-02   -4.751e-03   -7.956e-04
##        cmlc           am         amls         amlc
##  -6.122e-05   -2.820e-03    3.749e-04   -1.615e-05
##      cmlaml
##  -3.819e-04
##
##
## $female$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   -0.285395     0.500757    -0.156724    -0.029283
##        cmlc           am         amls         amlc
##   -0.002192    -0.002448     0.012689     0.002095
##      cmlaml
##   -0.006205
##
##
## $female$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##    0.345769    -0.796891     0.202197     0.022961
##        cmlc           am         amls         amlc
##    0.002058     0.079428    -0.023815     0.002820
##      cmlaml
##    0.043416
##
##
## $female$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
```

```
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   -0.933533     1.934863    -0.537323    -0.106343
##        cmlc           am         amls         amlc
##   -0.007025    -0.049005     0.014147     0.002104
##      cmlaml
##   -0.003531
##
##
## $female$offset
## [1] 10
##
## $female$q0
##
## Call:
## lm(formula = as.numeric(ql[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml         cmls
##     -0.9508       0.6592      -0.0377
##
##
## $female$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 -918.907400 -1067.7577497 -1106.9976489 -1130.482833
## 2  -46.762420   -80.3257730   -74.1620063   -72.859535
## 3   -6.093825    -4.9014561    -6.3117837   -13.775142
## 4    5.187333    -0.3900189    -0.6369774     4.633521
##              4            5            6            7
## 1 -1145.467156 -1156.215917 -1165.304184 -1171.972581
## 2   -65.772801   -61.863400   -61.348047   -57.246245
## 3   -12.212158   -11.915365   -13.959189   -14.145176
## 4     1.920031     3.085376     5.568228     5.782408
##             8            9          10           11
## 1 -1179.20131 -1185.634456 -1187.85386 -1188.719031
```

```
## 2   -53.18097   -51.176119   -48.51613   -44.674823
## 3   -13.96148   -17.160869   -13.28159   -14.143095
## 4     8.64218     4.862689     2.09220     1.748089
##              12           13           14           15
## 1 -1185.3539556 -1179.367834 -1166.757982 -1153.069064
## 2   -40.4969505   -38.072395   -30.082990   -25.843765
## 3   -12.0386341   -10.708389    -4.838728    -3.742612
## 4     0.8798751     1.515544    -3.065599    -4.221248
##              16           17           18           19
## 1 -1.135088e+03 -1119.994406 -1104.976629 -1098.944408
## 2 -1.732023e+01   -12.582271    -6.059524    -7.945558
## 3 -4.035991e-02     3.040498     6.075331     8.197556
## 4 -6.865471e+00    -6.585074   -11.082816   -11.609534
##             20          21          22          23
## 1 -1094.974023 -1092.67707 -1091.18860 -1090.64323
## 2   -10.166749   -11.03025   -12.04597   -12.26401
## 3     8.094635    10.25060    11.11026    12.27288
## 4   -12.580588   -14.69842   -12.02623   -12.50604
##            24          25          26           27
## 1 -1091.05006 -1090.80533 -1091.11963 -1089.911264
## 2   -12.58721   -11.78948   -11.36607    -8.961508
## 3    12.20967    13.35735    13.50989    14.381489
## 4   -11.74958   -10.84258   -11.71841   -10.365331
##             28           29           30           31
## 1 -1088.835823 -1087.567047 -1086.160889 -1084.208020
## 2    -8.400243    -7.364397    -9.740522    -5.375643
## 3    15.094168    14.411176    13.953099    14.842239
## 4    -8.635942    -8.237436    -7.350837    -7.041318
##             32           33           34           35
## 1 -1081.093880 -1078.435798 -1075.317085 -1071.972152
## 2    -6.379372    -4.961203    -3.978003    -4.598939
## 3    14.136720    14.472116    14.890262    14.141483
## 4    -4.735706    -3.974359    -2.777733    -1.195722
##             36           37            38
## 1 -1068.508786 -1065.560257 -1060.0771915
## 2    -2.624597    -3.093011    -0.5091694
## 3    13.421643    12.879395    13.1689338
## 4    -1.119241    -1.850958     0.7122004
##              39            40           41
## 1 -1055.8759034 -1050.6962716 -1047.156414
## 2     0.8938587     0.4514354     3.789717
## 3    12.1851889    12.4535174    11.640551
## 4     1.0732119     3.4734569     2.537027
##             42           43           44           45
## 1 -1040.777161 -1036.222318 -1031.371167 -1025.179953
## 2     4.134943     6.420122     7.153389     8.515586
## 3    10.944900    10.832794     9.998354     9.460780
## 4     4.040377     4.103012     4.584914     5.313658
##             46           47           48           49
## 1 -1020.743318 -1015.329017 -1009.804937 -1004.188594
## 2    10.145274    11.621507    11.826108    13.383243
## 3     8.916410     8.485985     7.902708     7.268519
## 4     5.728307     6.331918     6.893515     6.999801
##            50          51          52          53
```

```
## 1 -997.777654 -994.097103 -986.903491 -982.049441
## 2   13.437250   15.258691   15.740898   16.290281
## 3    6.983844    6.078865    5.851411    4.904395
## 4    7.701205    7.068229    7.901648    7.794469
##            54          55          56          57
## 1 -976.097212 -970.896291 -965.387386 -960.176928
## 2   17.535915   18.683344   19.276501   20.344859
## 3    4.567656    4.263172    3.289643    2.724659
## 4    8.105849    7.864041    8.292673    8.130057
##            58          59          60          61
## 1 -953.978910 -948.379380 -941.200060 -937.591799
## 2   20.275096   20.654850   20.048406   20.898099
## 3    2.291617    2.080595    2.618374    1.509016
## 4    8.946455    8.390194    9.337832    7.958720
##            62           63           64           65
## 1 -930.198506 -924.5725418 -918.5734711 -912.3624195
## 2   20.486635   20.5268522   20.6563287   20.6182449
## 3    1.416116    0.7290468    0.6334766    0.2099518
## 4    8.818968    8.5918179    8.6173659    8.8468299
##             66           67          68          69
## 1 -907.9145748 -901.7491004 -895.487712 -889.834177
## 2   21.1385531   21.1837263   21.110886   21.204106
## 3   -0.4054377   -0.7973464   -1.107968   -1.833113
## 4    7.3534470    7.6648716    7.563915    6.682618
##            70          71          72          73
## 1 -882.486582 -878.424221 -870.293626 -864.214496
## 2   19.837564   21.088000   20.026403   20.126150
## 3   -1.605840   -3.294001   -2.623511   -2.877187
## 4    7.300382    5.420135    6.543078    6.130091
##            74         75          76          77
## 1 -857.626694 -851.23161 -845.156607 -839.724954
## 2   19.851225   19.76794   20.049416   20.278227
## 3   -3.051375   -3.03863   -3.922084   -5.167379
## 4    6.203302    6.08607    5.587756    3.901878
##            78          79          80          81
## 1 -832.180196 -826.211813 -818.898210 -814.112678
## 2   19.936025   20.243795   20.234399   21.444922
## 3   -4.292642   -4.633387   -3.963119   -4.437799
## 4    4.935100    3.721218    4.060713    2.573278
##            82          83          84          85
## 1 -806.105990 -799.711084 -792.476835 -786.782192
## 2   20.670597   20.880805   20.559330   21.455172
## 3   -4.991629   -5.509210   -6.076071   -6.469185
## 4    2.477312    1.679605    1.477025    0.504247
##            86           87          88          89
## 1 -780.179766 -774.1536971 -768.109007 -762.318754
## 2   21.935225   22.6184392   22.925923   23.799227
## 3   -7.257010   -7.7368331   -8.113589   -8.662216
## 4   -0.039927   -0.9238375   -1.565932   -2.582696
##            90          91          92          93
## 1 -755.640320 -751.111137 -744.570987 -739.123298
## 2   23.702633   25.464396   25.658109   26.410372
## 3   -8.751635   -9.636846  -10.071138  -10.579257
## 4   -2.739763   -4.283337   -4.391203   -5.028195
```

```
##            94          95         96         97
## 1 -733.670476 -727.775823 -722.53691 -717.44895
## 2   26.777044   26.067273   26.48468   26.89721
## 3  -11.032325  -11.104530  -11.44645  -11.76423
## 4   -5.390992   -6.081449   -6.59490   -7.07833
##            98          99         100         101
## 1 -712.518841 -707.753005 -703.157225 -698.736500
## 2   27.302846   27.699483   28.085049   28.457353
## 3  -12.057035  -12.324199  -12.565218  -12.779860
## 4   -7.530303   -7.949676   -8.335431   -8.686902
##          102         103         104         105
## 1 -694.49518 -690.436551 -686.562931 -682.875751
## 2   28.81432   29.154043   29.474681   29.774755
## 3  -12.96803  -13.129963  -13.266024  -13.376832
## 4   -9.00369   -9.285696   -9.533101   -9.746394
##           106        107        108        109
## 1 -679.375195 -676.06048 -672.92983 -669.98046
## 2   30.052932   30.30832   30.54016   30.74823
## 3  -13.463232  -13.52632  -13.56719  -13.58726
## 4   -9.926345  -10.07408  -10.19096  -10.27854
##
## $male$components.sm
##             0             1             2            3
## 1 -918.907400 -1067.7577497 -1106.9976489 -1130.482833
## 2  -46.762420   -69.2341449   -72.6404749   -70.859043
## 3   -6.093825    -4.9014561    -8.5777244   -10.392577
## 4    5.187333    -0.3900189    -0.6369774     2.428802
##              4            5            6            7
## 1 -1145.467156 -1156.215917 -1165.304184 -1171.972581
## 2   -66.855021   -63.168542   -60.235519   -57.151344
## 3   -11.820047   -12.720042   -13.409787   -14.042210
## 4     3.097681     3.917469     4.716861     5.215068
##              8            9           10           11
## 1 -1179.201313 -1185.634456 -1187.853858 -1188.719031
## 2   -53.925228   -50.982433   -47.975248   -44.526205
## 3   -14.518931   -14.602081   -14.112633   -13.046000
## 4     5.131929     4.413756     3.289659     2.040126
##              12            13           14
## 1 -1185.3539556 -1179.3678340 -1166.757982
## 2   -40.7030447   -36.2354082   -30.783267
## 3   -11.3658852    -9.0075460    -6.153619
## 4     0.7419566    -0.7128786    -2.396808
##             15            16           17           18
## 1 -1153.069064 -1135.0880529 -1119.994406 -1104.976629
## 2   -24.696539   -18.4497216   -12.924260    -9.416816
## 3    -3.137867    -0.1460607     2.703693     5.188621
## 4    -4.236941    -6.1078215    -7.923543    -9.601648
##             19           20          21          22
## 1 -1098.944408 -1094.974023 -1092.67707 -1091.18860
## 2    -8.714735    -9.730916   -10.90407   -11.71497
## 3     7.135411     8.621973     9.85696    10.92147
## 4   -11.006613   -11.985796   -12.45870   -12.47518
##            23          24          25          26
## 1 -1090.64323 -1091.05006 -1090.80533 -1091.11963
```

```
## 2   -12.13514   -12.14085   -11.67920   -10.73773
## 3    11.78932    12.48805    13.10099    13.66756
## 4   -12.19109   -11.76598   -11.26945   -10.67723
##             27           28           29           30
## 1 -1089.911264 -1088.835823 -1087.567047 -1086.160889
## 2    -9.531846    -8.597743    -8.183067    -7.737077
## 3    14.134485    14.400310    14.453439    14.421459
## 4    -9.951616    -9.111299    -8.207944    -7.255760
##             31           32           33           34
## 1 -1084.208020 -1081.093880 -1078.435798 -1075.317085
## 2    -6.790102    -5.859306    -5.092387    -4.442810
## 3    14.413286    14.432099    14.427295    14.306123
## 4    -6.227857    -5.123182    -4.004588    -2.967544
##             35           36            37
## 1 -1071.972152 -1068.508786 -1065.5602574
## 2    -3.871715    -3.144514    -2.1577145
## 3    14.002229    13.582870    13.1761818
## 4    -2.071150    -1.281237    -0.4864325
##              38            39           40
## 1 -1060.0771915 -1055.8759034 -1050.696272
## 2    -0.8714882     0.3717158     1.575393
## 3    12.8220408    12.4744101    12.082568
## 4     0.3962924     1.3350227     2.221640
##             41           42           43           44
## 1 -1047.156414 -1040.777161 -1036.222318 -1031.371167
## 2     3.044126     4.563624     5.989119     7.311376
## 3    11.625727    11.123324    10.598970    10.053880
## 4     2.979836     3.613430     4.172191     4.702744
##             45           46           47           48
## 1 -1025.179953 -1020.743318 -1015.329017 -1009.804937
## 2     8.641552    10.009110    11.195475    12.145740
## 3     9.500172     8.958133     8.426695     7.893087
## 4     5.226928     5.740519     6.221817     6.643789
##             49          50          51          52
## 1 -1004.188594 -997.777654 -994.097103 -986.903491
## 2    13.022028   13.919975   14.866942   15.726080
## 3     7.353287    6.804032    6.240838    5.670760
## 4     6.988316    7.256865    7.470056    7.650322
##            53          54          55          56
## 1 -982.049441 -976.097212 -970.896291 -965.387386
## 2   16.555797   17.512781   18.485018   19.320786
## 3    5.110794    4.564851    4.007604    3.435937
## 4    7.807636    7.946138    8.078363    8.220175
##            57          58          59          60
## 1 -960.176928 -953.978910 -948.379380 -941.200060
## 2   19.944928   20.284986   20.398214   20.451616
## 3    2.918663    2.534908    2.274712    2.027688
## 4    8.371592    8.508498    8.598607    8.629542
##            61          62           63           64
## 1 -937.591799 -930.198506 -924.5725418 -918.5734711
## 2   20.543765   20.578746   20.5828469   20.6456150
## 3    1.700397    1.307874    0.9088496    0.5169059
## 4    8.619302    8.590433    8.5343504    8.4130432
##             65           66          67          68
```

```
## 1 -912.3624195 -907.9145748 -901.749100 -895.487712
## 2   20.7895793   20.9822574   21.098595   21.056273
## 3    0.1073272   -0.3277693   -0.767889   -1.203158
## 4    8.2025186    7.9230007    7.617468    7.309775
##           69          70          71          72
## 1 -889.83418 -882.486582 -878.424221 -870.293626
## 2   20.83221   20.584407   20.462067   20.287854
## 3   -1.64466   -2.090234   -2.479046   -2.743807
## 4    7.00245    6.708487    6.461757    6.280783
##            73          74          75          76
## 1 -864.214496 -857.626694 -851.231613 -845.156607
## 2   20.076642   19.939408   19.922118   20.017274
## 3   -2.923416   -3.140114   -3.485652   -3.924883
## 4    6.131462    5.939390    5.645681    5.253323
##            77          78          79          80
## 1 -839.724954 -832.180196 -826.211813 -818.898210
## 2   20.098854   20.133965   20.258074   20.542900
## 3   -4.285235   -4.434818   -4.435844   -4.467339
## 4    4.816835    4.380538    3.937194    3.454402
##            81          82          83          84
## 1 -814.112678 -806.105990 -799.711084 -792.476835
## 2   20.820798   20.871042   20.838985   20.985553
## 3   -4.665530   -5.042544   -5.525531   -6.050571
## 4    2.922995    2.360753    1.780777    1.174461
##            85           86           87          88
## 1 -786.782192 -780.1797662 -774.1536971 -768.109007
## 2   21.405292   21.9643634   22.5211466   23.056443
## 3   -6.593869   -7.1380894   -7.6525235   -8.121856
## 4    0.526544   -0.1655933   -0.8894405   -1.627376
##            89          90          91          92
## 1 -762.318754 -755.640320 -751.111137 -744.570987
## 2   23.601563   24.247543   25.024939   25.717589
## 3   -8.567092   -9.029921   -9.528676  -10.030396
## 4   -2.363510   -3.081786   -3.761365   -4.385396
##            93          94          95          96
## 1 -739.123298 -733.670476 -727.775823 -722.536912
## 2   26.200292   26.406184   26.429030   26.573736
## 3  -10.482818  -10.858492  -11.171536  -11.460680
## 4   -4.956984   -5.496836   -6.021736   -6.531891
##            97          98          99         100
## 1 -717.448947 -712.518841 -707.753005 -703.157225
## 2   26.904249   27.297915   27.692631   28.076869
## 3  -11.748829  -12.031292  -12.295922  -12.536393
## 4   -7.017838   -7.471945   -7.891601   -8.276815
##           101         102         103         104
## 1 -698.736500 -694.495183 -690.436551 -686.562931
## 2   28.447899   28.803681   29.142296   29.462002
## 3  -12.750973  -12.939355  -13.101715  -13.238452
## 4   -8.627748   -8.944388   -9.226527   -9.473817
##           105         106         107        108
## 1 -682.875751 -679.375195 -676.060481 -672.92983
## 2   29.761280   30.038382   30.287309   30.48773
## 3  -13.349962  -13.436524  -13.498416  -13.53790
## 4   -9.684991   -9.857712   -9.990453  -10.08562
```

```
##          109
## 1 -669.98046
## 2   30.61837
## 3  -13.56048
## 4  -10.15008
##
## $male$aml
##
## Call:
## lm(formula = aml ~ cm + cml + cmls + cmlc)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##    -0.70346      2.88318      0.35930      0.10413
##        cmlc
##     0.01598
##
##
## $male$v1
##
## Call:
## lm(formula = svd$v[, 1] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##   9.171e-03    1.034e-02   -3.288e-03   -6.297e-04
##        cmlc           am         amls         amlc
##  -4.522e-05   -1.981e-03    9.949e-05   -1.180e-05
##      cmlaml
##  -3.034e-05
##
##
## $male$v2
##
## Call:
## lm(formula = svd$v[, 2] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##  -0.1937817    0.3200685   -0.1093803   -0.0209625
##        cmlc           am         amls         amlc
##  -0.0015087   -0.0079042    0.0023952    0.0008903
##      cmlaml
##  -0.0006750
##
##
## $male$v3
##
## Call:
## lm(formula = svd$v[, 3] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
```

```
## Coefficients:
## (Intercept)           cm          cml         cmls
##   0.1606100   -0.4248549    0.1161023    0.0261142
##        cmlc           am         amls         amlc
##   0.0016607    0.1096817   -0.0021355   -0.0009072
##      cmlaml
##  -0.0035825
##
##
## $male$v4
##
## Call:
## lm(formula = svd$v[, 4] ~ cm + cml + cmls + cmlc + am + amls +
##     amlc + cmlaml)
##
## Coefficients:
## (Intercept)           cm          cml         cmls
##  -9.107e-01    1.880e+00   -5.268e-01   -1.002e-01
##        cmlc           am         amls         amlc
##  -6.459e-03   -5.700e-02   -3.956e-03   -3.731e-05
##      cmlaml
##  -2.277e-03
##
##
## $male$offset
## [1] 10
##
## $male$q0
##
## Call:
## lm(formula = as.numeric(ql[1, samp]) ~ cml + cmls)
##
## Coefficients:
## (Intercept)          cml         cmls
##    -0.82806      0.68947     -0.03693
##
##
## $male$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())
rm(list = ls())
```