R Markdown Supplement to Gaussian Process Models for Mortality Rates and Improvement Factors

Jimmy Risk, Mike Ludkovski and Howard Zail

January 2018

The following notebook is an electronic supplement to the article *Gaussian Process Models for Mortality Rates and Improvement Factors* by M. Ludkovski, J. Risk and H. Zail, henceforth **LRZ**.

In particular, most of the plots and figures appearing in the manuscript can be obtained by executing the code below. By modifying the input datasets to include other countries/genders/sub-populations, similar analysis can be extended by the readers.

Mortality Data

```
library(dplyr)
library(rgenoud)
library(DiceKriging)
library(StMoMo)
library(fields)
```

The input data should be an R data frame with factors **age**, **year**, **deaths** and **exposures**. The corresponding log mortality rates are computed as

$$y^n = \log(D^n/L^n)$$

where D^n and L^n are, respectively, the number of deaths and midyear count of lives for the nth age/year pair $x^n = (x_{ag}^n, x_{yr}^n)$.

```
mortData <- read.csv("cdcMale.csv",header=T)
mortData$rate <- mortData$D / mortData$L
mortData$y <- log(mortData$rate)
head(mortData)
```

```
##   X age year     D       L        rate         y
## 1 1  50 1999  9775 1847555 0.005290776 -5.241790
## 2 2  51 1999 10470 1762492 0.005940452 -5.125970
## 3 3  52 1999 11509 1900702 0.006055131 -5.106849
## 4 4  53 1999  9885 1355175 0.007294261 -4.920667
## 5 5  54 1999 10717 1413117 0.007583944 -4.881722
## 6 6  55 1999 11728 1390616 0.008433673 -4.775523
```

Fitting the baseline GP model

As a start we use US male dataset based on CDC data.

For the baseline Gaussian Process model, we utilize the package **DiceKriging** available on CRAN.

The function **km()** is used to fit a GP model based on input-output set (x, y). Regarding the parameters supplied:

- *formula* determines the mean function $m(x)$ [here a constant, see further linear and quadratic-mean models below].

- *covtype* refers to kernel type, taken to be squared-exponential, aka Gaussian.

- *nugget.estim=TRUE* tells km() to infer the intrinsic (homoskedastic, cell-independent) noise variance σ^2 as part of the model.

- *optim.method="gen"* tells km() to utilize a genetic optimization algorithm from library **rgenoud()**, which produces more reliable parameter estimates and is recommended by DiceKriging() authors.

- *control=...* are internal (recommended) parameters of the above optimization algorithm

- See https://cran.r-project.org/web/packages/DiceKriging/DiceKriging.pdf for a more detailed explanation of **km()** options.

```r
xMort <- data.frame(age = mortData$age, year = mortData$year)
yMort <- mortData$y
mortModel_nug <- km(formula = ~1,
                    design = data.frame(x = xMort), response = yMort,
                    nugget.estim=TRUE,
                    covtype="gauss",
                    optim.method="gen",
                    control=list(max.generations=100,pop.size=100,wait.generations=8,
                                 solution.tolerance=1e-5))
```

```r
mortModel_nug
```

```
##
## Call:
## km(formula = ~1, design = data.frame(x = xMort), response = yMort,
##     covtype = "gauss", nugget.estim = TRUE, optim.method = "gen",
##     control = list(max.generations = 100, pop.size = 100, wait.generations = 8,
##         solution.tolerance = 1e-05))
##
## Trend  coeff.:
##               Estimate
##   (Intercept)   -3.8711
##
## Covar. type  : gauss
## Covar. coeff.:
##               Estimate
##   theta(x.age)   15.7528
##  theta(x.year)   15.5641
##
## Variance estimate: 1.822485
##
## Nugget effect estimate: 0.0002808419
```

The above code output shows estimates of (in order) $\beta_0, \theta_{ag}, \theta_{yr}, \eta^2$ and σ^2, cf Table 3 in the article. Despite the assumption of a nugget effect, **km()** by default treats the observations as without having noise variance. To be consistent with our setting, we re-enter the model using the above fitted parameters along with inputting **noise.var**:

```r
nug <- mortModel_nug@covariance@nugget
mortModel <- km(formula = ~1, design = mortModel_nug@X, response = mortModel_nug@y,
                noise.var = rep(nug,mortModel_nug@n),
                coef.trend = mortModel_nug@trend.coef,  # re-use obtained hyperparameters
                coef.cov = mortModel_nug@covariance@range.val,
                coef.var = mortModel_nug@covariance@sd2,
                covtype = mortModel_nug@covariance@name)
```

Predicting using the GP model

After constructing a **km** object, the main workhorse is the **predict()** command. Among its outputs are the posterior mean, posterior standard deviation, and 95% credible bands. The **predict()** command allows for *any* test set with the same number of covariates as the training set. In our case, we may get a 2-d posterior mortality surface for any ages and years desired to forecast upon.
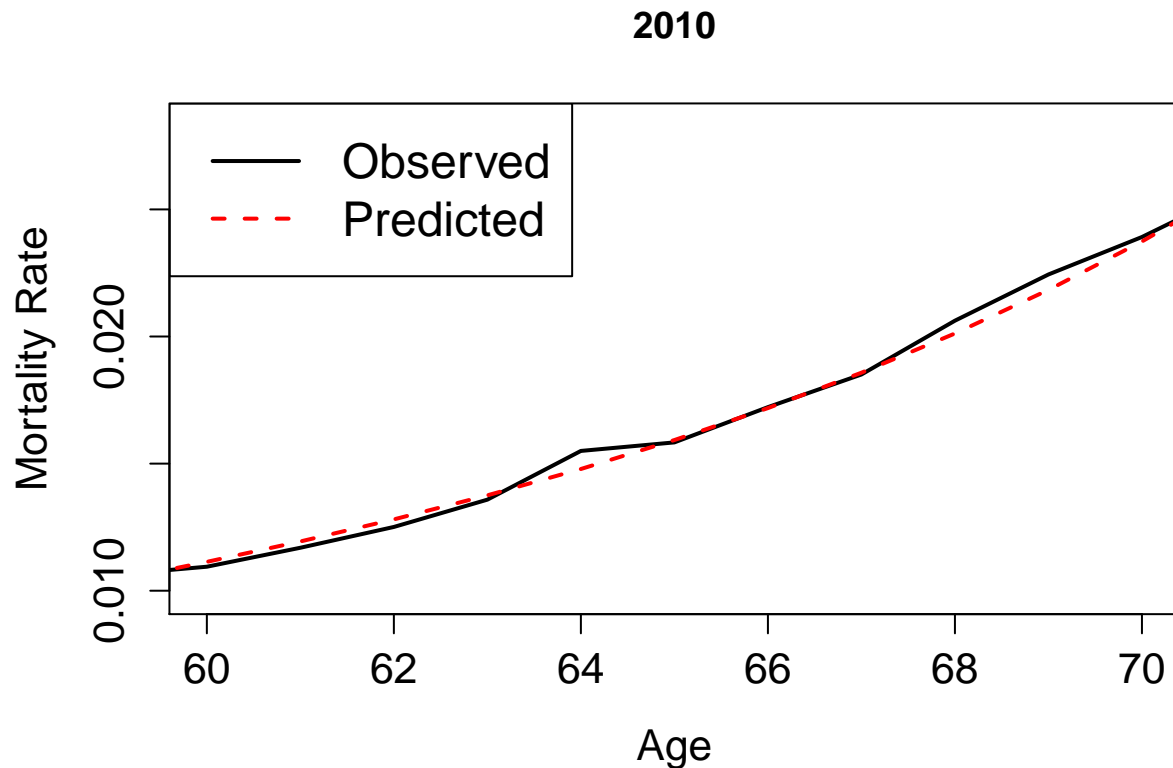
In-sample Smoothing

We beging with producing plots of estimated (smoothed) mortality rates for calendar year 2010 and ages 58–72. We use the library **dplyr** for easier data handling.

```r
# cf Figure 1 in LRZ
agesForecast <- seq(58,72,1)
yearsForecast <- 2010

rateForecastTrue <- dplyr::filter(mortData,age %in% agesForecast, year %in% yearsForecast)$rate
# build data frame for desired forecasts, then call predict
xPred <- data.frame(age=agesForecast,year=yearsForecast)
mortPred <- predict(mortModel, newdata=data.frame(x=xPred),
                    cov.compute=TRUE,
                    se.compute=TRUE,type="UK")

# plot mortality as a function of age
plot(agesForecast,rateForecastTrue,type="l",lwd=2, main="2010",
     xlab="Age",ylab="Mortality Rate",cex.axis=1.3,cex.lab=1.3,cex=1.3,
     xlim=c(60,70))
lines(agesForecast,exp(mortPred$mean),col=2,lty=2,lwd=2)
legend("topleft",c("Observed","Predicted"),lwd=c(2,2),lty=c(1,2),col=c(1,2),cex=1.5)
```

2010

We can predict on a larger data set to get a plot showing several years of smoothed data. Below we show in-sample smoothing for years 2010–2014, inclusive (cf right panel of Figure 1 in LRZ)

```
# Figure 1 in LRZ

agesForecast <- seq(58,72,1)
yearsForecast <- 2010:2014

xPred <- select(dplyr::filter(mortData, age %in% agesForecast, year %in% yearsForecast), age,year)
mortPred <- predict(mortModel, newdata=data.frame(x=xPred),cov.compute=TRUE,
            se.compute=TRUE,type="UK")

xPred$m <- mortPred$mean
rateForecast <- dplyr::filter(xPred,age %in% agesForecast, year %in% 2014)$m
plot(agesForecast,exp(rateForecast),
     type="l",lwd=2, main="Smoothed Mortality",
     xlab="Age",ylab="Mortality Rate",cex.axis=1.3,cex.lab=1.3,cex=1.3, xlim=c(60,70))

for(yrForecast in 2010:2013){
  rateForecast <- dplyr::filter(xPred,age %in% agesForecast, year %in% yrForecast)$m
  lines(agesForecast,exp(rateForecast),
        type="l",lwd=2, lty=yrForecast-2008,col=yrForecast-2008)
}
legend("topleft",c("2010","2011","2012","2013","2014"),col=c(2,3,4,5,1),
       lty=c(2,3,4,5,1),lwd=rep(2,5),cex=1.3)
```

Smoothed Mortality
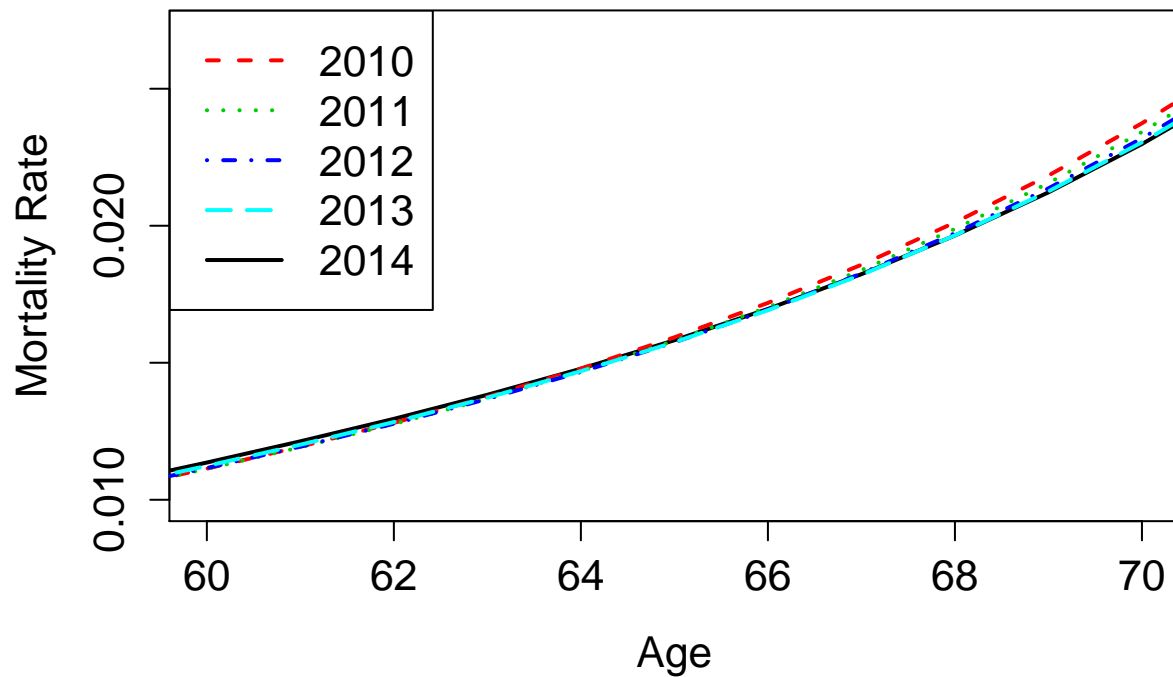
Mortality over time

We can also produce a plot with year on the x-axis. Below, we additionally illustrate use of 95% credible intervals that quantify the (statistical) credibility of the smoothing. Our implementation of building a model with noise variance equal to the nugget causes **predict()** to return credible bands for the true surface f and not the observation process y. Thus we need to add σ^2 to achieve the bands for the observed mortality y that would be relevant for coverage tests.

```
# cf Figure 2 in LRZ
agesForecast <- c(60,70,84)
yearsForecast <- 1998:2015
yearsObserved <- 1999:2014

nYr <- length(yearsForecast)
nAg <- length(agesForecast)
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsForecast,nAg))

mortPred <- predict(mortModel, newdata=data.frame(x=xPred),cov.compute=TRUE,
             se.compute=TRUE,type="UK")

xPred$m <- mortPred$mean
xPred$lower95 <- mortPred$lower95 - 1.96*sqrt(nug)  # predictive CI for y
xPred$upper95 <- mortPred$upper95 + 1.96*sqrt(nug)

for(ageObs in agesForecast){
```

```r
    rateObserved <- dplyr::filter(mortData, age == ageObs, year %in% yearsObserved)$rate
    dataPred <- dplyr::filter(xPred, age == ageObs, year %in% yearsForecast)

    # exponentiate to show actual mortality rates (not logged)
    ratePred <- exp(dataPred$m);
    upper95Pred <- exp(dataPred$upper95); lower95Pred <- exp(dataPred$lower95)

    plot(yearsObserved,rateObserved, type="l",
        lwd=2, main = paste("Age", ageObs),
        xlab="Year", ylab="Mortality Rate", cex.axis=1.5, cex.lab=1.5, cex=1.5,
        xlim=c(1999,2014), ylim=c(min(lower95Pred),max(upper95Pred)))
    lines(yearsForecast,ratePred, col=2, lwd=2)
    lines(yearsForecast,upper95Pred, col=2, lwd=2, lty=2)
    lines(yearsForecast,lower95Pred, col=2, lwd=2, lty=2)
    legend("topright",c("Observed","Predicted","95% Credible Band"),
        lwd=c(2,2,2),lty=c(1,1,2),col=c(1,2,2),cex=1)
}
```
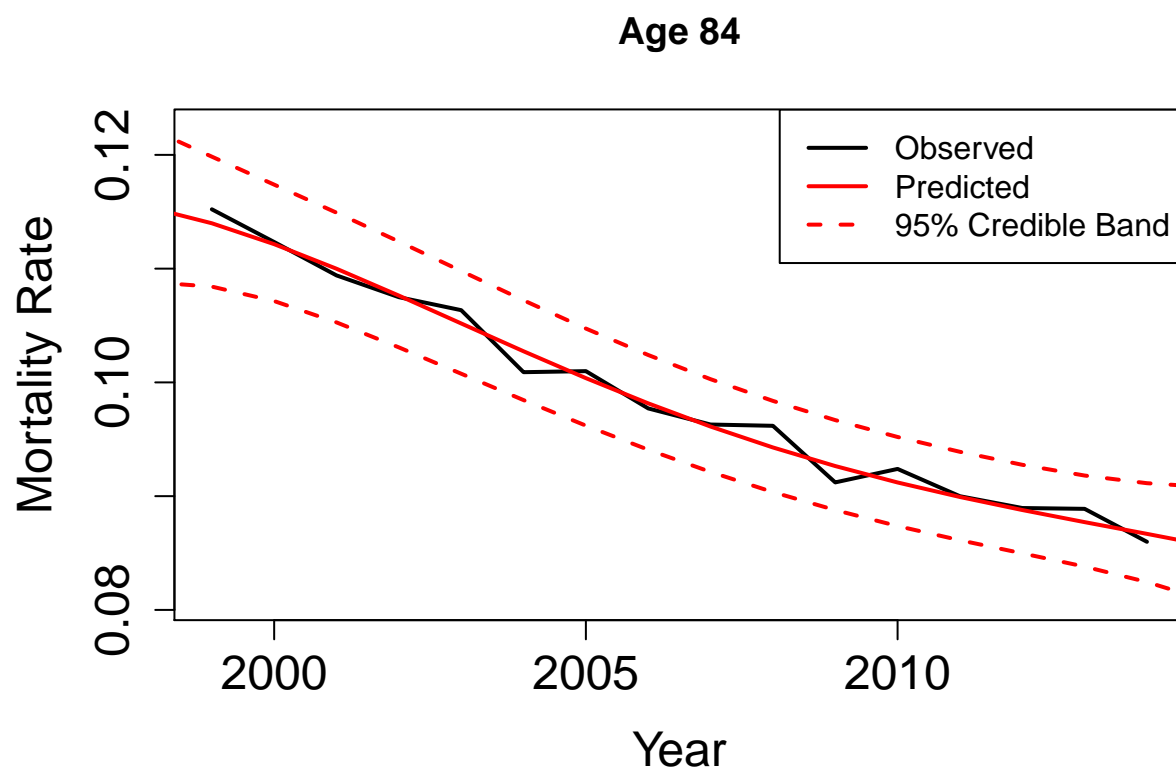
Age 70

Age 84
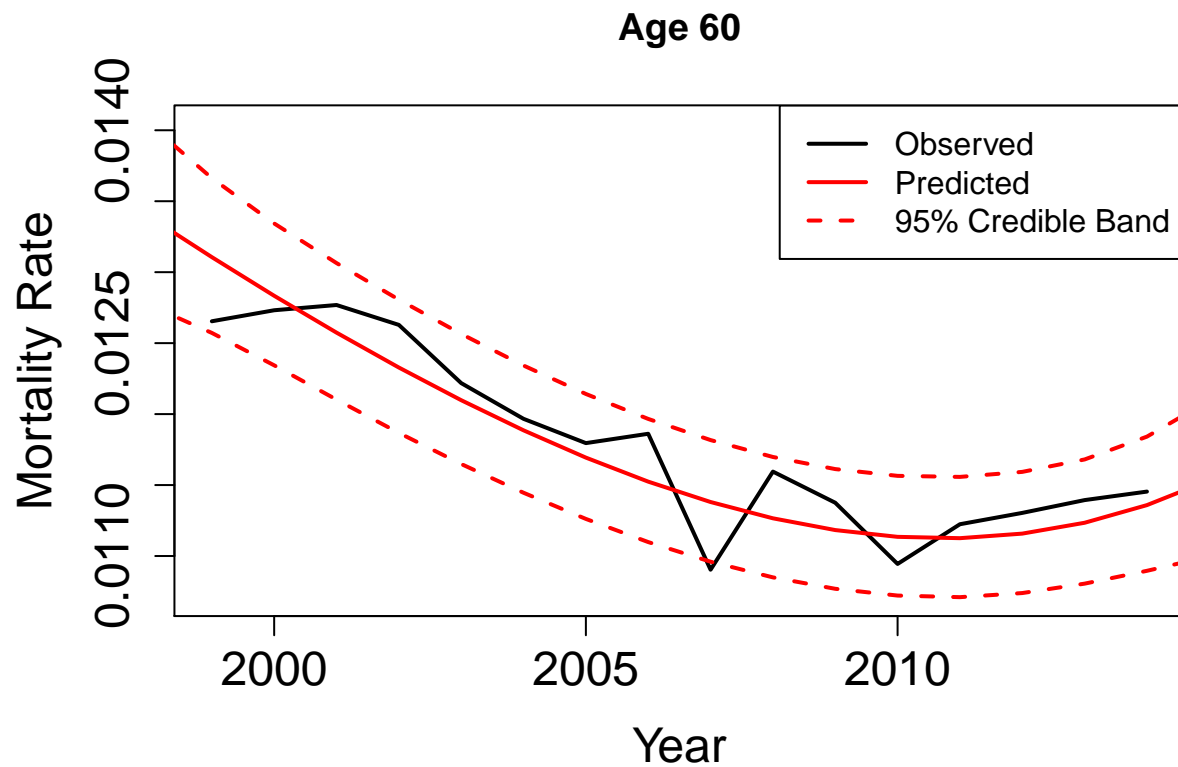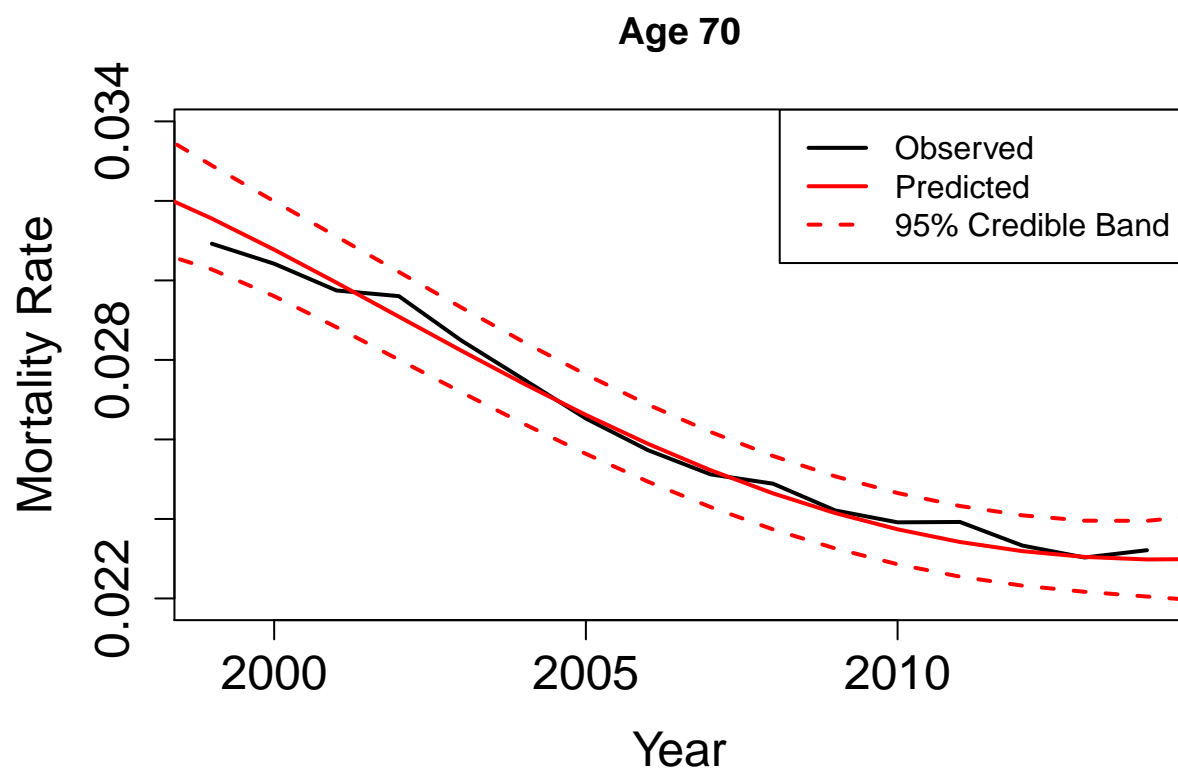
Mortality Improvement

For YoY mortality improvement rates, we calculate the finite difference expression. For comparison we also plot the improvement rates reported from SOA MP-2015 tables. See Figure 3 in LRZ.

```r
# compare to MP-2015 tables
mp2015 <- read.csv("mp2015.csv")

agesForecast <- 48:86
agesObserved <- 50:84
yearsForecast <- c(2000,2014)
yearsPred <- c(1999,2000,2013,2014) # need to add one year back for improvement

nYr <- length(yearsPred)
nAg <- length(agesForecast)

# predict
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsPred,nAg))
mortPred <- predict(mortModel, newdata=data.frame(x=xPred),cov.compute=TRUE,
          se.compute=TRUE,type="UK")

xPred$m <- mortPred$mean
```

```r
for(yr in yearsForecast){
  forwardObs <- dplyr::filter(mortData, age %in% agesObserved, year == yr)$rate
  backwardObs <- dplyr::filter(mortData, age %in% agesObserved, year == yr-1)$rate
  MIbackobs <- 1-forwardObs/backwardObs    # raw observed improvement rates

  forwardPred <- filter(xPred, age %in% agesForecast, year == yr)$m
  backwardPred <- filter(xPred, age %in% agesForecast, year == yr-1)$m
  # smoothed improvement rates using the GP model
  mibackgp <- 1-exp(forwardPred)/exp(backwardPred)

  plot(agesObserved,MIbackobs, type="l", lwd=2, main = yr, ylim=c(-0.05, 0.1), xlab="age",
       ylab="YoY Mortality Improvement Rate", cex.axis=1.5,cex.lab=1.5,cex=1.5, col=2)
  lines(agesForecast,mibackgp, col=4, lwd=2, lty=3)
  lines(c(0,100),c(0,0))

  # MP-2015 rates
  mpRate <- dplyr::filter(mp2015, gender=="male", age %in% agesForecast, year == yr)$improvement
  lines(agesForecast, mpRate, col=3, lwd=2, lty=2)

  legend("topleft",c("Observed","GP Smoothed","MP-2015"), col=c(2,4,3),lwd=rep(2,3),
         lty=c(1,3,2),cex=1.3, bty="n")

}
```
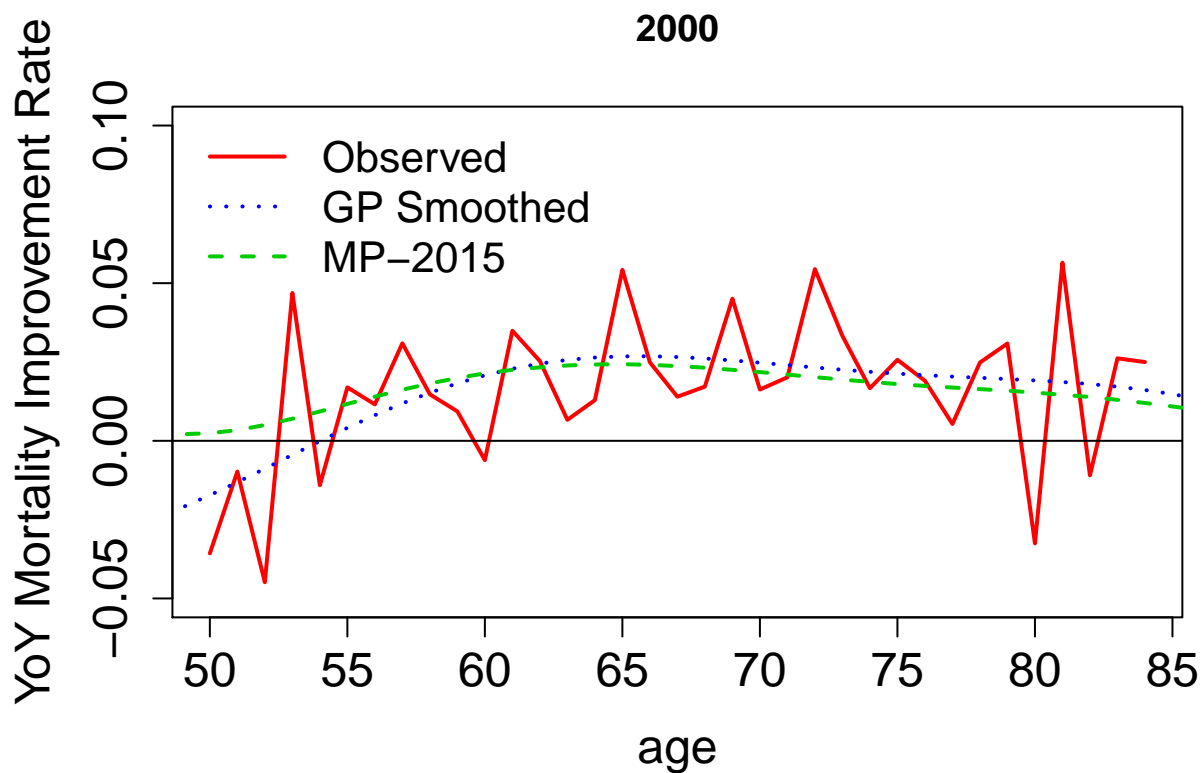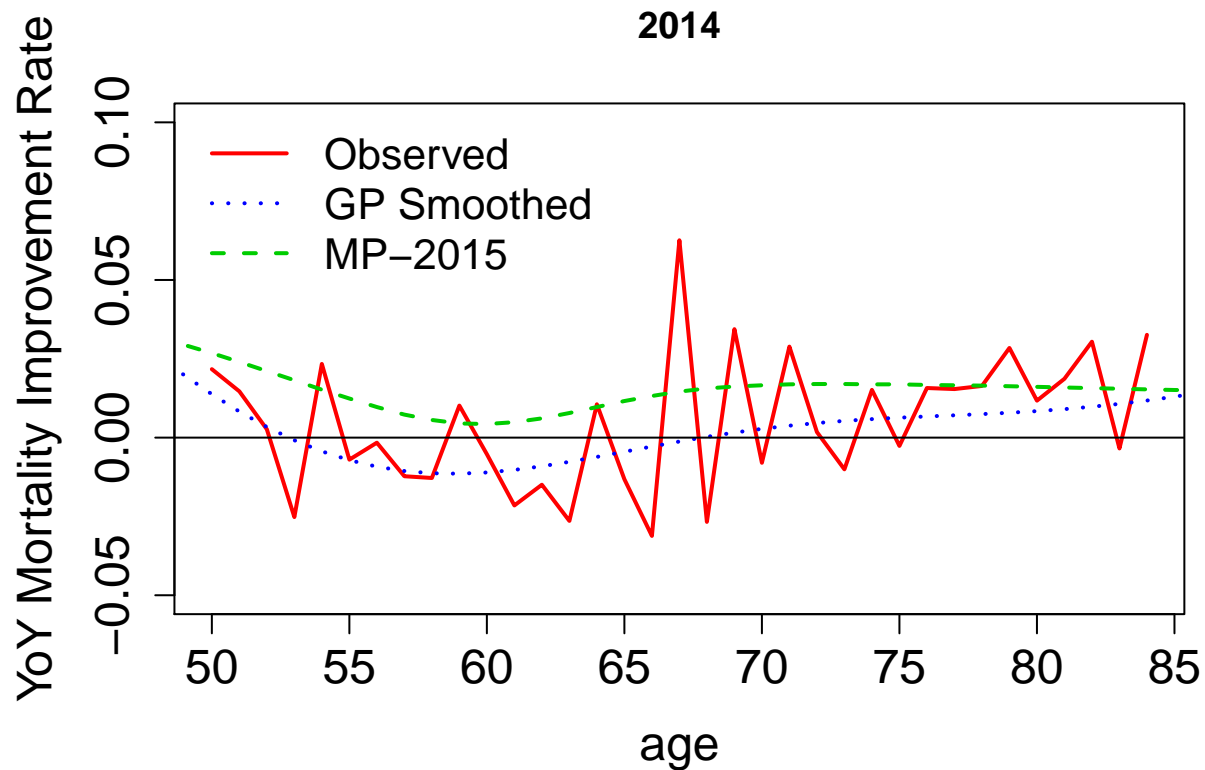
The following code produces improvement rates for every even year and plots them together, cf Fig 4 in LRZ

```
agesForecast <- 48:86
yearsForecast <- 1999:2014


nYr <- length(yearsForecast)
nAg <- length(agesForecast)
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsForecast,nAg))
mortPred <- predict(mortModel, newdata=data.frame(x=xPred),cov.compute=TRUE,
            se.compute=TRUE,type="UK")

xPred$m <- mortPred$mean
forwardPred <- dplyr::filter(xPred, age %in% agesForecast, year %in% 2000:2014)$m
backwardPred <- dplyr::filter(xPred, age %in% agesForecast, year %in% 1999:2013)$m
mibackgp <- 1-exp(forwardPred)/exp(backwardPred)   # GP-smoothed improvement rate
xPred$mibackgp <- NA
xPred$mibackgp[which(xPred$year > 1999)] <- mibackgp

miPlot <- dplyr::filter(xPred, age %in% agesForecast, year == 2014)$mibackgp
plot(agesForecast,miPlot, type="l", lwd=2, xlab="Age", xaxs="i",
        ylab="YoY Mortality Improvement Rate", cex.axis=1.5,cex.lab=1.5,cex=1.5,
      ylim=c(-0.0275, 0.0275), xlim=c(50,85))
lines(c(0,100),c(0,0))
for(yr in seq(2000,2012,2)){
  miPlot <- dplyr::filter(xPred, age %in% agesForecast, year == yr)$mibackgp
  lines(agesForecast,miPlot,col=(yr-1996)/2,lty=(yr-1996)/2,lwd=2)
```
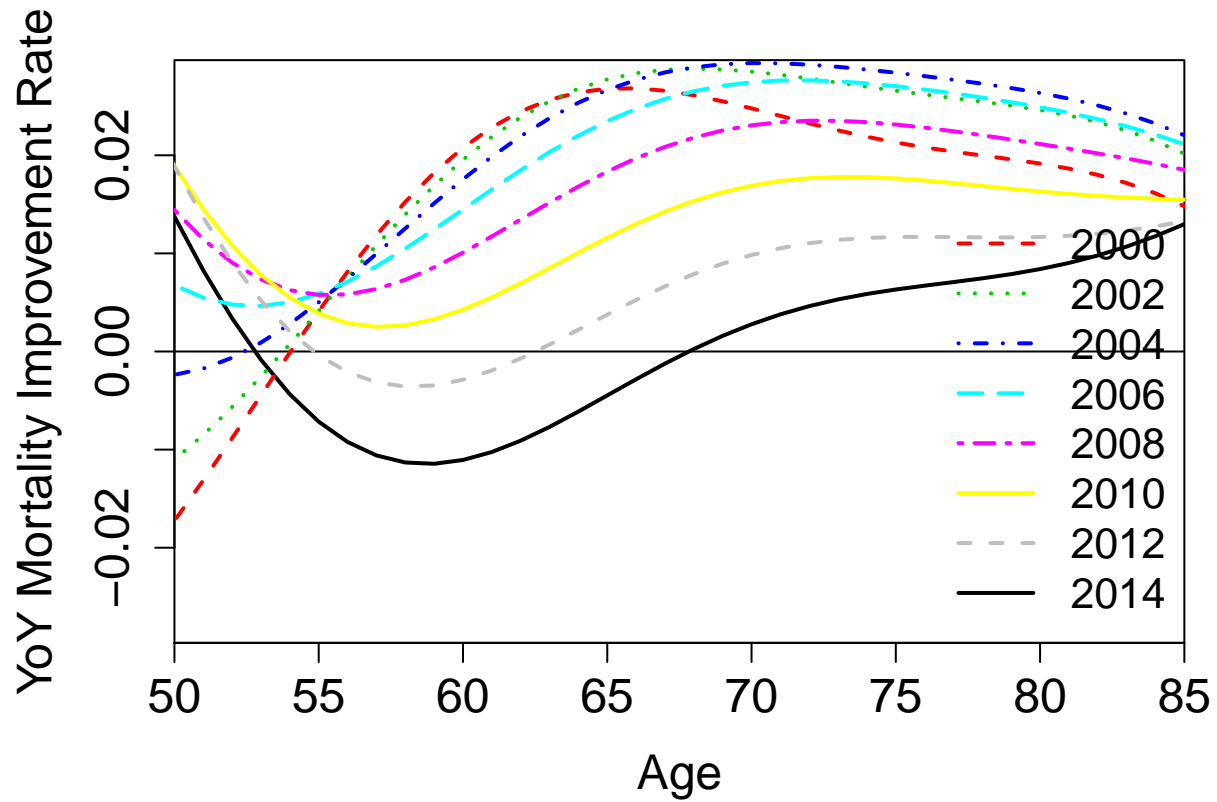
```
}
legend("bottomright",legend=seq(2000,2014,by=2), col=c(2:8,1),lwd=rep(2,8),
        lty=c(2:8,1),cex=1.3,bty='n')
```
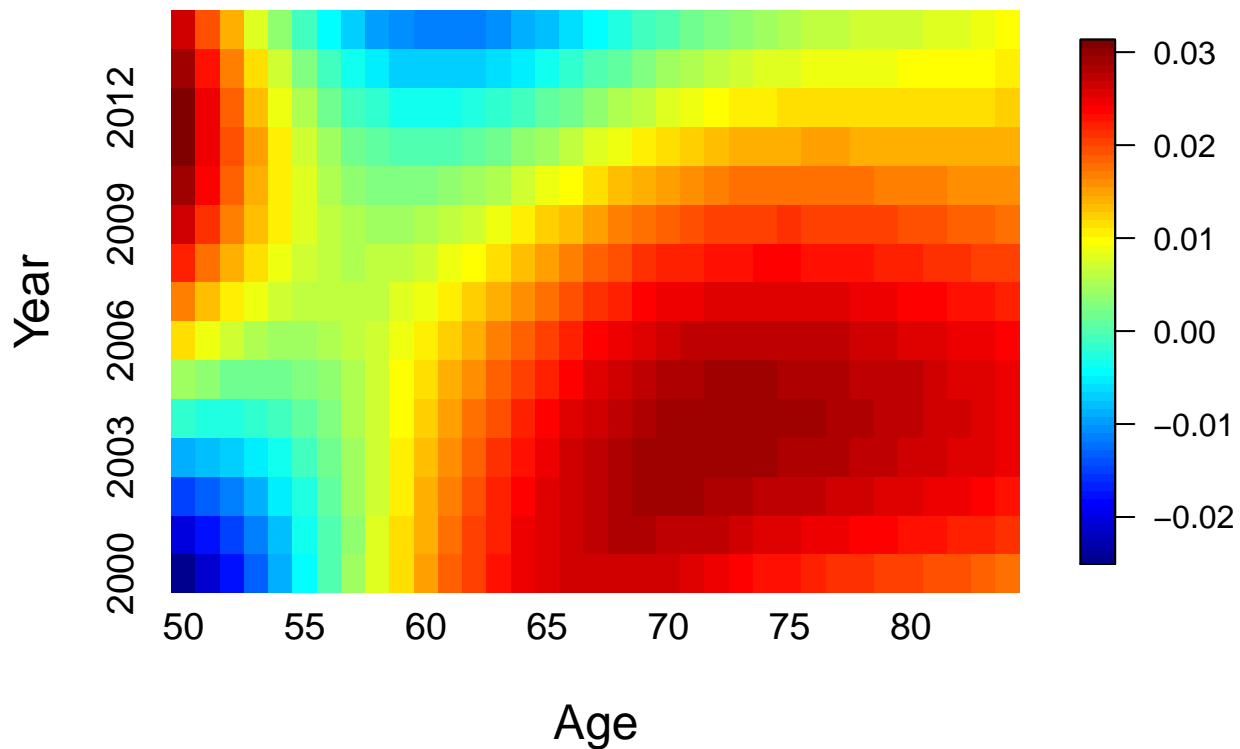


```
image.plot(t(matrix(mibackgp,15,35)),xaxs="i",yaxs="i",xlab="Age",ylab="Year",
        main="", axes="F", cex.axis=1.5,cex.lab=1.5)
```

```
## Warning in matrix(mibackgp, 15, 35): data length [585] is not a sub-
## multiple or multiple of the number of columns [35]
```

```
title(main="")
axis(2, at=seq(0,1,length=15), labels=2000:2014, tick=FALSE, cex.axis=1.2,line=-0.2)
axis(1, at=seq(0,1,by=5/34), labels=seq(50,84,by=5), tick=FALSE,, cex.axis=1.2,line=-0.6)
```

Comparison of Mean-function specifications

km() takes in an object of class **formula** to specify a trend function, just like with linear lm() and generalized linear model glm() fitting. The following code produces models with linear and quadratic trend. We also change the training set to Subset III.

```r
# create mortality Subset III as in Section 3.2 of LRZ
mortData3 <- dplyr::filter(mortData, age %in% 50:70, year %in% 1999:2010)
xMort3 <- data.frame(age = mortData3$age, year = mortData3$year)
yMort3 <- mortData3$y

# fit intercept only model
mortModel3int_nug <- km(formula = ~1,
                    design = data.frame(x = xMort3), response = yMort3,
                    nugget.estim=TRUE,
                    covtype="gauss",
                    optim.method="gen",
                    control=list(max.generations=100,pop.size=100,
                                 wait.generations=8,solution.tolerance=1e-5))

nug_int <- mortModel3int_nug@covariance@nugget
```

```r
mortModelint <- km(formula = ~1,
                   design = mortModel3int_nug@X, response = mortModel3int_nug@y,
                   noise.var = rep(nug_int,mortModel3int_nug@n),
                   coef.trend = mortModel3int_nug@trend.coef,
                   coef.cov = mortModel3int_nug@covariance@range.val,
                   coef.var = mortModel3int_nug@covariance@sd2,
                   covtype = mortModel3int_nug@covariance@name)


# fit linear trend model
mortModel3lin_nug <- km(formula = ~x.age+x.year,   # linear in age, linear in year
                   design = data.frame(x = xMort3), response = yMort3,
                   nugget.estim=TRUE,
                   covtype="gauss",
                   optim.method="gen",
                   control=list(max.generations=100,pop.size=100,
                                wait.generations=8,solution.tolerance=1e-5))


nug_lin <- mortModel3lin_nug@covariance@nugget
mortModellin <- km(formula = ~x.age+x.year,
                   design = mortModel3lin_nug@X, response = mortModel3lin_nug@y,
                   noise.var = rep(nug_lin,mortModel3lin_nug@n),
                   coef.trend = mortModel3lin_nug@trend.coef,
                   coef.cov = mortModel3lin_nug@covariance@range.val,
                   coef.var = mortModel3lin_nug@covariance@sd2,
                   covtype = mortModel3lin_nug@covariance@name)


# fit quadratic-age trend model
mortModel3quad_nug <- km(formula = ~x.age+I(x.age^2)+x.year,  # quadratic in age, linear in year
                       design = data.frame(x = xMort3), response = yMort3,
                       nugget.estim=TRUE,
                       covtype="gauss",
                       optim.method="gen",
                       control=list(max.generations=100,pop.size=100,
                                    wait.generations=8,solution.tolerance=1e-5))


nug_quad <- mortModel3quad_nug@covariance@nugget
mortModelquad <- km(formula = ~x.age+I(x.age^2)+x.year,
                   design = mortModel3quad_nug@X, response = mortModel3quad_nug@y,
                   noise.var = rep(nug_quad,mortModel3quad_nug@n),
                   coef.trend = mortModel3quad_nug@trend.coef,
                   coef.cov = mortModel3quad_nug@covariance@range.val,
                   coef.var = mortModel3quad_nug@covariance@sd2,
                   covtype = mortModel3quad_nug@covariance@name)
```

Using **show()** we can compare the fitted coefficients of the trends, as well as the GP hyperparameters, such as the lengthscales θ, see Table 5 in LRZ.

```r
# cf Table 5 in LRZ
show(mortModelint)


##
## Call:
## km(formula = ~1, design = mortModel3int_nug@X, response = mortModel3int_nug@y,
##     covtype = mortModel3int_nug@covariance@name, coef.trend = mortModel3int_nug@trend.coef,
##     coef.cov = mortModel3int_nug@covariance@range.val, coef.var = mortModel3int_nug@covariance@sd2,
```

```
##      noise.var = rep(nug_int, mortModel3int_nug@n))
##
## Trend  coeff.:
##
##   (Intercept)     -4.5005
##
## Covar. type  : gauss
## Covar. coeff.:
##
##  theta(x.age)      7.1164
## theta(x.year)     11.4884
##
## Variance: 0.3780334
```

show(mortModellin)

```
##
## Call:
## km(formula = ~x.age + x.year, design = mortModel3lin_nug@X, response = mortModel3lin_nug@y,
##      covtype = mortModel3lin_nug@covariance@name, coef.trend = mortModel3lin_nug@trend.coef,
##      coef.cov = mortModel3lin_nug@covariance@range.val, coef.var = mortModel3lin_nug@covariance@sd2,
##      noise.var = rep(nug_lin, mortModel3lin_nug@n))
##
## Trend  coeff.:
##
##   (Intercept)     19.0244
##         x.age      0.0811
##        x.year     -0.0141
##
## Covar. type  : gauss
## Covar. coeff.:
##
##  theta(x.age)      3.6149
## theta(x.year)     3.5492
##
## Variance: 0.001456947
```

show(mortModelquad)

```
##
## Call:
## km(formula = ~x.age + I(x.age^2) + x.year, design = mortModel3quad_nug@X,
##      response = mortModel3quad_nug@y, covtype = mortModel3quad_nug@covariance@name,
##      coef.trend = mortModel3quad_nug@trend.coef, coef.cov = mortModel3quad_nug@covariance@range.val,
##      coef.var = mortModel3quad_nug@covariance@sd2, noise.var = rep(nug_quad,
##          mortModel3quad_nug@n))
##
## Trend  coeff.:
##
##   (Intercept)     19.6413
##         x.age      0.0636
##   I(x.age^2)      0.0001
##        x.year     -0.0142
##
## Covar. type  : gauss
```

```
## Covar. coeff.:
##
##  theta(x.age)     3.6293
## theta(x.year)     3.4746
##
## Variance: 0.001403191
```

Next, we use these models to forecast beyond age 70, to see how well they can extrapolate. We do it for years 2011 and 2014 (both out of sample). Compare to Figure 5 in LRZ

```r
# Figure 5 in LRZ
agesForecast <- 48:86
yearsForecast <- c(2011,2014)
nYr <- length(yearsForecast)
nAg <- length(agesForecast)
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsForecast,nAg))

agesObserved <- 50:84
rateObserved <- dplyr::filter(mortData,age %in% agesObserved, year %in% yearsForecast)$rate

# predict using the 3 models above
mortPredInt <- predict(mortModelint, newdata=data.frame(x=xPred),cov.compute=TRUE,
             se.compute=TRUE,type="UK")
mortPredLin <- predict(mortModellin, newdata=data.frame(x=xPred),cov.compute=TRUE,
             se.compute=TRUE,type="UK")
mortPredQuad <- predict(mortModelquad, newdata=data.frame(x=xPred),cov.compute=TRUE,
             se.compute=TRUE,type="UK")

xPred$mInt <- exp(mortPredInt$mean)
xPred$mLin <- exp(mortPredLin$mean)
xPred$mQuad <- exp(mortPredQuad$mean)


for(yr in yearsForecast){
  rateObserved <- dplyr::filter(mortData,age %in% agesObserved, year == yr)$rate
  plot(agesObserved,rateObserved,
      lwd=2,cex.axis=1.5,cex.lab=1.5,main=yr,cex=1.5,
      xlab="Age", ylab="Mortality Rate",type="l")

  xPredyr <- dplyr::filter(xPred,year==yr)
  lines(agesForecast, xPredyr$mInt, col=2, lty=2, lwd=2)
  lines(agesForecast, xPredyr$mLin, col=3, lty=3, lwd=2)
  lines(agesForecast, xPredyr$mQuad, col=4, lty=4, lwd=2)
  lines(c(70,70),c(-100,100),lty=2)
  legend("topleft",legend=c("Observed","Intercept Only","Linear Trend","Quadratic Trend"),
      lwd=rep(2,4),col=1:4,lty=1:4,cex=1.3, bty="n")
}
```
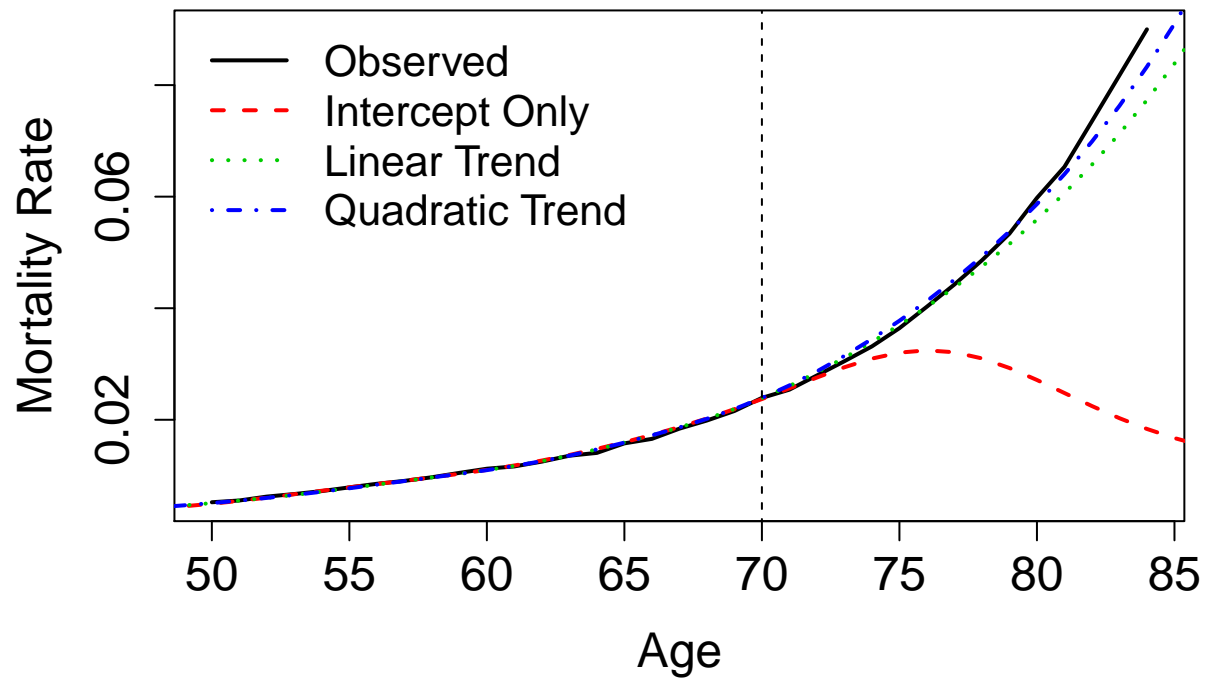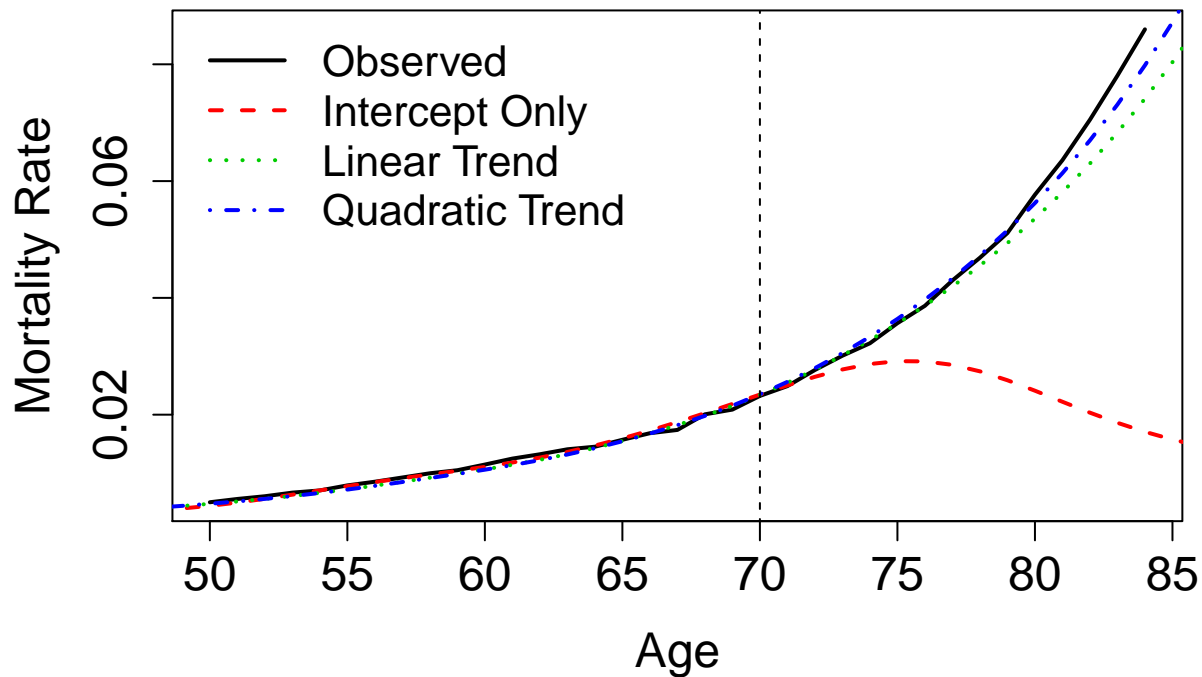
2011

2014

Mortality Scenarios using Conditional Sampling

The **simulate()** function allows to generate conditional samples from a fitted GP model. It can be used to produce stochastic scenarios of the mortality surface conditional on the observed data.

The next plot illustrates these using Subset II data, cf Figure 6 in LRZ. Also shown are the credible intervals for y along with those for f (which in the GP framework are the same except for $\pm 1.96 * \sigma$).

```r
# create mortality subset II as in Section 3.3 of LRZ
mortData2 <- dplyr::filter(mortData, year <= 2009 | age <= 70)
xMort2 <- data.frame(age = mortData2$age, year = mortData2$year)
yMort2 <- mortData2$y

# fit intercept only model
mortModel2int_nug <- km(formula = ~1,
                design = data.frame(x = xMort2), response = yMort2,
                nugget.estim=TRUE,
                covtype="gauss",
                optim.method="gen",
                control=list(max.generations=100,pop.size=100,
                        wait.generations=8,solution.tolerance=1e-5))

nug_2 <- mortModel2int_nug@covariance@nugget
mortModel2int <- km(formula = ~1,
                design = mortModel2int_nug@X, response = mortModel2int_nug@y,
```

```
                    noise.var = rep(nug_2,mortModel2int_nug@n),
                    coef.trend = mortModel2int_nug@trend.coef,
                    coef.cov = mortModel2int_nug@covariance@range.val,
                    coef.var = mortModel2int_nug@covariance@sd2,
                    covtype = mortModel2int_nug@covariance@name)
```

simulate() takes in "nugget.sim" to ensure numeric stability. The value can be quite small, here we take the nugget from the model and divide by 100. In some cases "nugget.sim" can be left out. Usually if the data is nonstationary and there is no trend to compensate, a nugget effect is needed for simulation.

```
# Figure 6 in LRZ

agesForecast <- 68:86
yearsForecast <- c(2011,2014)   # look at 2011 and 2014
nYr <- length(yearsForecast)
nAg <- length(agesForecast)
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsForecast,nAg))

# predict() to obtain predictive mean + CI
mortPred2 <- predict(mortModel2int, newdata=data.frame(x=xPred),
                     cov.compute=TRUE,  se.compute=TRUE,type="UK")

xPred$m <- exp(mortPred2$mean)
xPred$fupper95 <- exp(mortPred2$upper95)
xPred$flower95 <- exp(mortPred2$lower95)
xPred$yupper95 <- exp(mortPred2$upper95+1.96*sqrt(nug))
xPred$ylower95 <- exp(mortPred2$lower95-1.96*sqrt(nug))


agesObserved <- 50:84

# generate 5 cond simulations using simulate()
for(yr in yearsForecast){
  rateObserved <- dplyr::filter(mortData,age %in% agesObserved, year %in% yr)$rate
    xSim <- select(dplyr::filter(xPred,year==yr),age,year)
  nsim <- 5
  sim <- simulate(mortModel2int, nsim=nsim, newdata = data.frame(x=xSim), cond=TRUE,
                  nugget.sim=nug/100)

  xPredyr <- dplyr::filter(xPred,year==yr)

  plot(NULL,cex.axis=1.5,cex.lab=1.5,
       xlab="Age",ylab="Log-Mortality Rate", main=paste(yr,"(Intercept Only)"),
       xlim=c(70,84),ylim=c(min(log(xPredyr$ylower95)),max(log(xPredyr$yupper95))))

  for(i in 1:nsim){
    lines(agesForecast,sim[i,],col="grey57")
  }

  lines(agesForecast,log(xPredyr$m),col=2,lty=2,lwd=2)
  lines(agesForecast,log(xPredyr$flower95),col=3,lty=3,lwd=2)
  lines(agesForecast,log(xPredyr$fupper95),col=3,lty=3,lwd=2)

  # error bars for predictive interval of Y
```

```
    segments(agesForecast, log(xPredyr$ylower95),
             agesForecast, log(xPredyr$yupper95),lwd=2,col=1)
    epsilon = 0.075
    segments(agesForecast-epsilon,log(xPredyr$ylower95),
             agesForecast+epsilon,log(xPredyr$ylower95),lwd=c(2,2),col=1)
    segments(agesForecast-epsilon,log(xPredyr$yupper95),
             agesForecast+epsilon,log(xPredyr$yupper95),lwd=c(2,2),col=1)

    points(agesObserved,log(rateObserved),pch=16,col=4,cex=1.5)

    legend("topleft", c("Actual","Predicted","95% Credible Band for f","Sampled Trajectories of f",
                        "95% PI for Y"), cex=1.3,
           col=c("blue",2,3,"grey57",1), pch=c(16,-1,-1,-1,-1),lty=c(0,5,2,1,1),lwd=c(1,2,2,1,2))
}
```
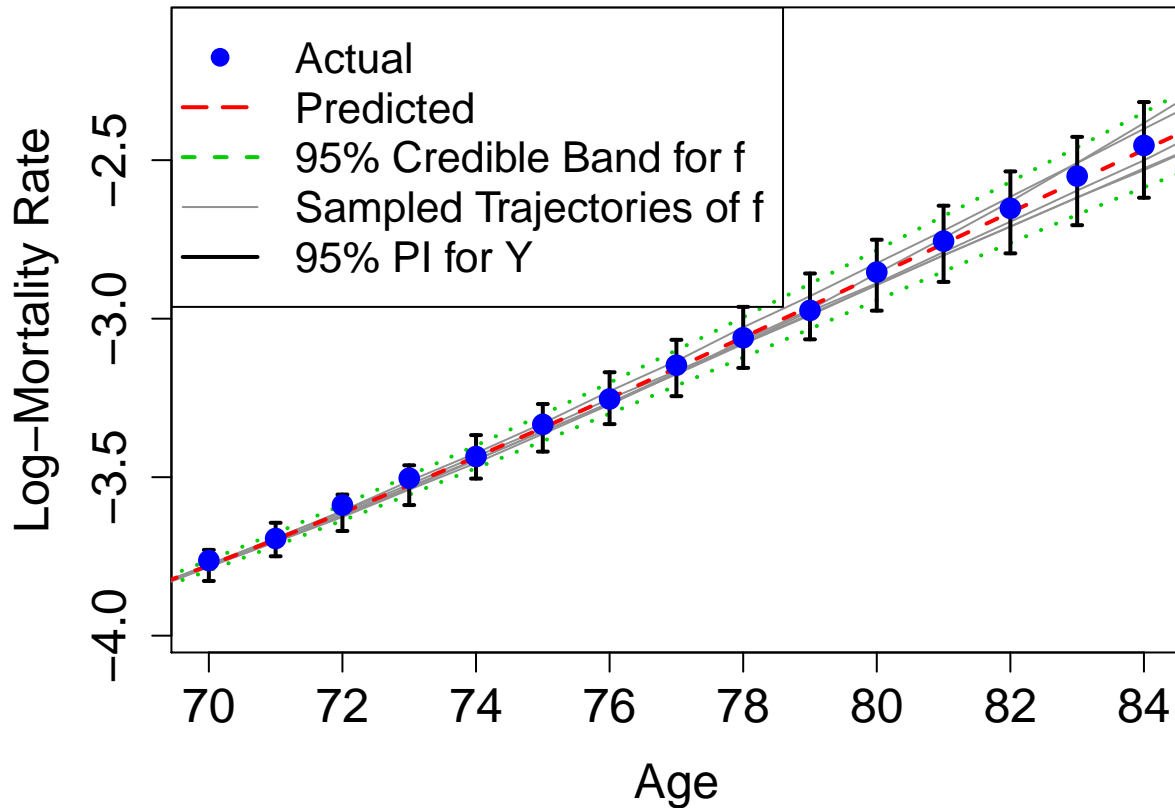

2011 (Intercept Only)
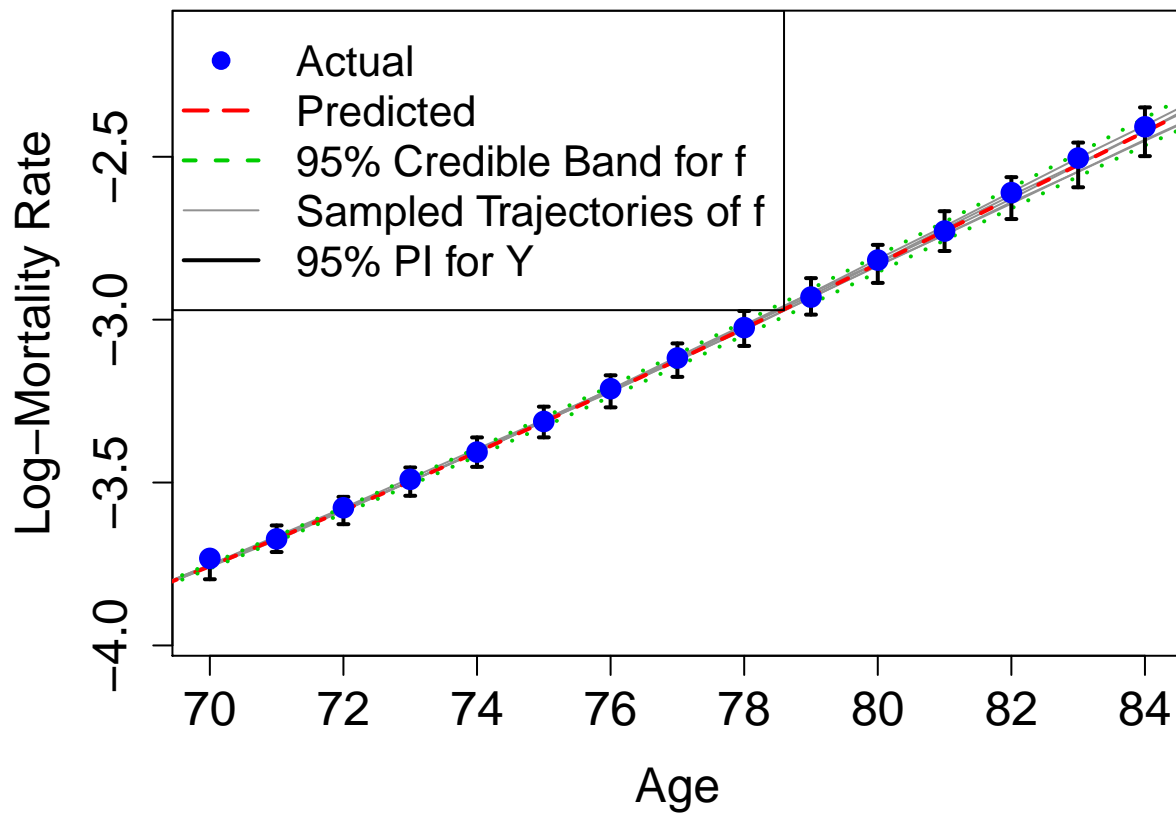
2014 (Intercept Only)

GP Spatial Derivatives for Instantaneous Improvement Factors

DiceKriging does not directly deal with GP derivatives. Here, we create a function that takes in a **km** object and returns the posterior mean and variance for the derivative $\frac{\partial}{\partial x_k} f(x)$ where $x = (x_1, \ldots, x_k, \ldots, x_n)$. This code is only for the case where the kernel family is squared-exponential (Gaussian), as it directly applies formulas for the respective spatial derivative. See Section 3.4 in LRZ for more details.

```
gpDerivative <- function(fit,xstar,i=2,DeltaInv=0){
  # fit is a km object
  #   - requires Gaussian kernel
  #   - requires no nugget (noise variance should be)
  #     supplied through noise.var
  # xstar is the test set
  # i indicates which dimension the derivative should be taken in
  #   by default i=2 (implies year derivative for this example)
  # DeltaInv is the inverse of the covariance matrix.  It can be given
  #   to reduce computational cost, if it has already been
  #   inverted.  Otherwise, it pulls Delta from the km object
  #   and inverts it.
  # Returns a posterior mean and posterior covariance for the
```

```r
  #  distribution evaluated at xstar.

  if(fit@covariance@name != "gauss"){
    stop("Covariance kernel is not Gaussian.")
  }
  if(length(fit@covariance@nugget) != 0){
    stop("Requires no nugget.  Refit model using nugget as noise variance.")
  }

  xstar <- as.matrix(xstar)
  y <- fit@y
  x <- fit@X
  eta2 <- fit@covariance@sd2
  theta <- fit@covariance@range.val[i]
  c.1 <- covMatrix(fit@covariance,xstar)$C - covMatrix(fit@covariance,xstar)$vn
  c.2 <- covMat1Mat2(fit@covariance,xstar,x, nugget.flag=FALSE)
  if(DeltaInv == 0){
    # T is the Choleski decomposition of the covariance matrix
    T <- fit@T
    Delta <- t(T) %*% T

    # Computing Delta through T is equivalent to the
    #   commented code below.  Included so that the user
    #   can see exactly what T is doing.
    # Delta <- covMatrix(fit@covariance,x, noise.var=0)$C
    # C <- covMatrix(fit@covariance,x, noise.var=0)$C
    # Sigma <- diag(fit@noise.var)
    # Delta <- C+Sigma

    DeltaInv <- solve(Delta)
  }

  # derivative of the squared-exponential kernel
  dcxx <- -1/theta^2 * outer(xstar[,i],xstar[,i],'-') * c.1
  d2c <- 1/theta^2*(-c.1 + outer(xstar[,i],xstar[,i],'-')*dcxx)
  dcxX <- -1/theta^2 * outer(xstar[,i],x[,i],'-') * c.2
  dcXx <- -1/theta^2 * outer(x[,i],xstar[,i],'-') * t(c.2)

  m <- dcxX %*% DeltaInv %*% y
  covMat <- -(d2c - dcxX %*% DeltaInv %*% dcXx)

  return(list(m=m,covmat=covMat))

}
```

The following code plots mortality improvement wrt calendar time using the instantaneous GP derivative. Also plotted are the GP backward difference and the MP-2015 improvement factors. Producing confidence bands for the backward difference requires knowing the distribution of $f(\cdot, yr) - f(\cdot, yr - 1)$. We can use the properties of GPs to easily compute this, and also the mean, variance and quantiles of $1 - \exp(f(\cdot, yr) - f(\cdot, yr - 1))$ through the lognormal distribution. See Figure 7 in LRZ.

```r
mortFiniteDiff <- function(fit,year=2015,ages=50:84,h=1,type="back",cl=0.95,
                           nsim=1e5,log=0){
  # fit is a km object
```

```r
#    - requires no nugget (noise variance should be)
#      supplied through noise.var
# year is the year at which the difference is initiated
# ages is the age range for the test set
# h is the time difference (h=1 for year over year improvement rate)
# type is the type of difference ("back", "centered")
# cl is the confidence level returned for the quantiles
# nsim is the number of simulations to estimate the quantiles
# Returns a posterior mean, standard deviation, and 80% bands
#   for the finite difference.
# If log==1, returns the log finite difference posterior f(yr)-f(yr-1)

if(fit@covariance@name != "gauss"){
  stop("Covariance kernel is not Gaussian.")
}

if(type == "back"){ # centered difference
  t0 <- year-h
  t1 <- year
} else{
  if(type == "centered"){
    t0 <- year-h/2
    t1 <- year+h/2
  } else{ stop("Need backward or centered difference") }
}
nAg <- length(ages)
xPred <- data.frame(age=rep(ages, 2),year=c(rep(t0,nAg),rep(t1,nAg)))
pred <- predict(fit, newdata = data.frame(x=xPred),cov.compute=TRUE, type="UK")
predictedlogMortDiff <- data.frame(age = ages, year = rep(year,nAg))
predictedlogMortDiff$m <- (pred$m[(nAg+1):(2*nAg)] - pred$m[1:nAg])/h
predictedlogMortDiff$sd2 <- (pred$sd[1:nAg]^2 + pred$sd[(nAg+1):(2*nAg)]^2 -
                              2*diag(pred$cov[1:nAg,(nAg+1):(2*nAg)]))/h^2
pu <- 1-(1-cl)/2
pl <- (1-cl)/2
predictedlogMortDiff$lower <- qnorm(pl, mean=predictedlogMortDiff$m,
                                      sd=sqrt(predictedlogMortDiff$sd2))
predictedlogMortDiff$upper <- qnorm(pu, mean=predictedlogMortDiff$m,
                                      sd=sqrt(predictedlogMortDiff$sd2))

if(log == 1){
  return(predictedlogMortDiff)
}

# use lognormal distribution.  X ~ N(mu, sd2)
# implies 1-exp(X) has mean 1-exp(mu+sd2/2)
# and variance exp(2mu+sd2)(exp(sd2)-1)
# and the quantiles can be taken as a 1-to-1 function
mu <- predictedlogMortDiff$m; sd2 <- predictedlogMortDiff$sd2
predictedMortDiff <- data.frame(age = ages, year = rep(year,nAg))
predictedMortDiff$m <- 1-exp(mu+sd2/2)
predictedMortDiff$sd2 <- exp(2*mu+sd2)*(exp(sd2)-1)
predictedMortDiff$lower <- 1-exp(predictedlogMortDiff$lower)
predictedMortDiff$upper <- 1-exp(predictedlogMortDiff$upper)
```

```
    return(predictedMortDiff)


}
```

Finally, we can produce mortality improvement plots for the two methods, including quantiles. Here it is for years 2000 and 2014, see Figure 7 in LRZ

```
# Fig 7 in LRZ

agesForecast <- 48:86
agePlotRange <- c(50,84)
yearsForecast <- c(2000,2014)
cl <- 0.8 # confidence level: show 10-90% quantiles

nYr <- length(yearsForecast)
nAg <- length(agesForecast)


for(yr in yearsForecast){
  xPred <- data.frame(age=agesForecast,year=rep(yr,nAg))
  fprime <- gpDerivative(mortModel,xPred,2)
  # instantaneous
  miDiff <- fprime; miDiff$m <- -miDiff$m # sign change from definition

  # YoY backward-looking
  miBack <- mortFiniteDiff(mortModel,year=yr,ages=agesForecast,cl=cl)
  miDiffsd <- sqrt(diag(miDiff$covmat))
  miDiffLower <- qnorm( (1-cl)/2, miDiff$m, miDiffsd)
  miDiffUpper <- qnorm( (1+cl)/2, miDiff$m, miDiffsd)

  # MP-2015 comparator
  mpRate <- filter(mp2015,gender=="male", age %in% agesForecast, year == yr)$improvement

  ylower <- min(miDiffLower,miBack$lower,mpRate); yupper <- max(miDiffUpper,miBack$upper,
                                                               mpRate)

  plot(NULL,xlim=agePlotRange,ylim=c(-0.02,0.06),main=yr, xaxs="i",
       ylab="Mortality Improvement Rate", xlab="age",cex.axis=1.5,cex.lab=1.5,
       cex.main=1.5)

  # shade the CI's of the GP estimates
  polygon(c(agesForecast,rev(agesForecast)),c(miBack$upper,rev(miBack$lower)),
          col="skyblue")
  polygon(c(agesForecast,rev(agesForecast)),c(miDiffUpper,rev(miDiffLower)),
          col="orange",density=25)

  lines(agesForecast,miBack$lower,col="blue")
  lines(agesForecast,miBack$upper,col="blue")
  lines(agesForecast,miDiffLower,col="red")
  lines(agesForecast,miDiffUpper,col="red")

  lines(agesForecast,miBack$m,lwd=2,col="blue")
  lines(agesForecast,miDiff$m,col="red",lty=2,lwd=2)
```
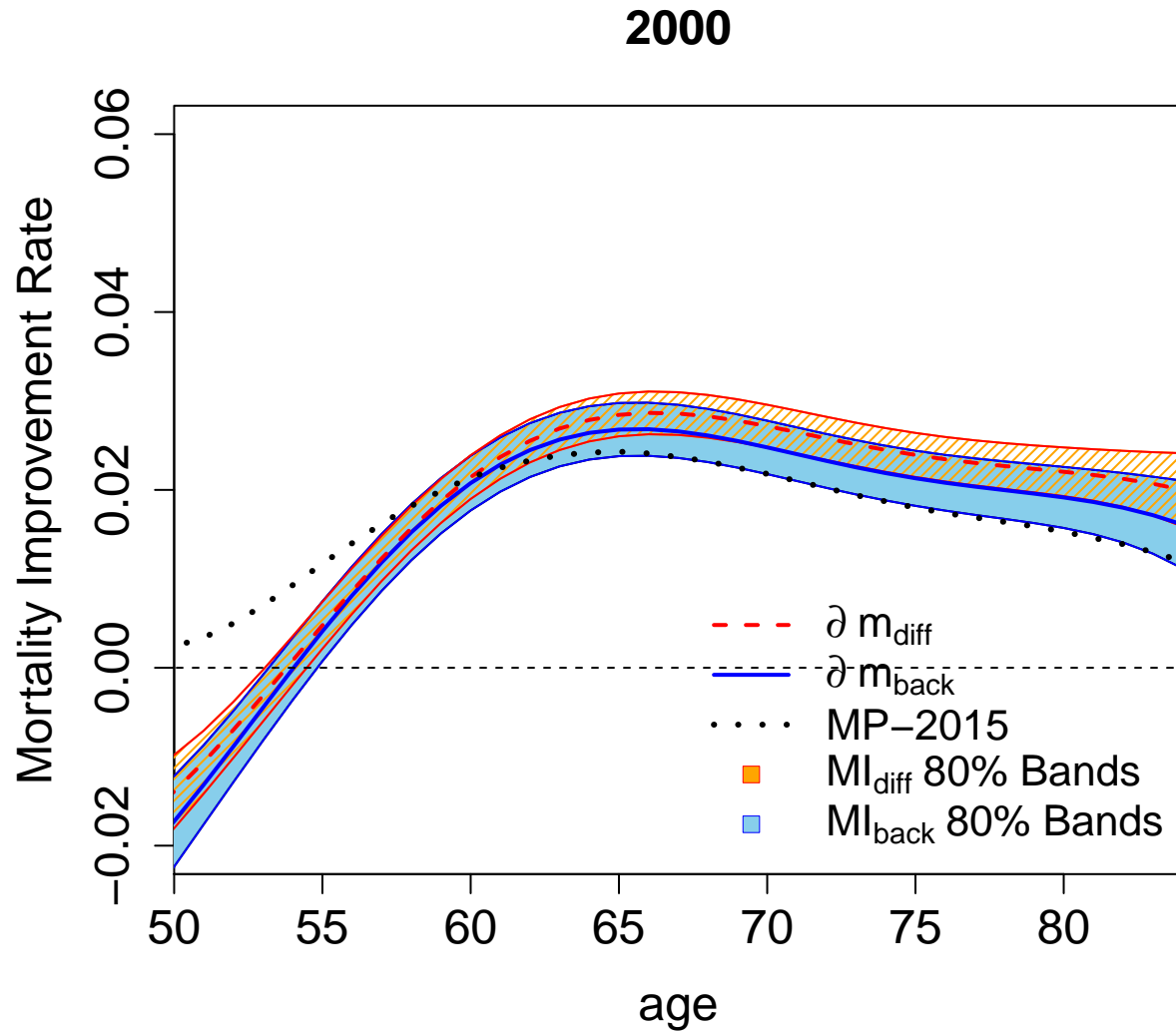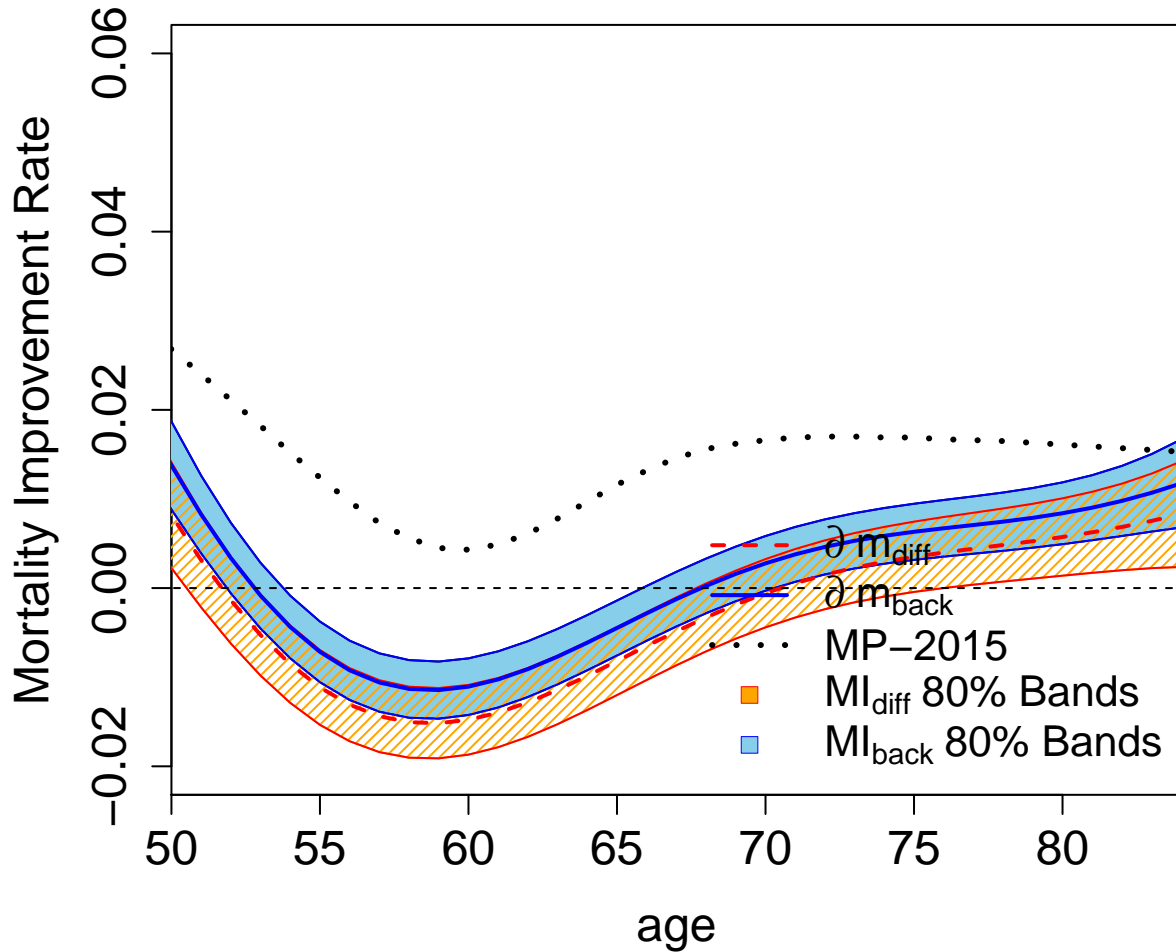
```
lines(agesForecast,mpRate,lty=3,lwd=3)

lines(c(-100,100),c(0,0),lty=2)
legend("bottomright", c(expression(partialdiff~m[diff]),expression(partialdiff~m[back]),"MP-2015",
                        expression(MI[diff]~"80% Bands"),expression(MI[back]~"80% Bands")),
       cex=1.3, col=c("red","blue",1), lty=c(2,1,3,0,0),lwd=c(2,2,3,0,0),
       pch=c(NA,NA,NA,22,22),pt.bg=c(NA,NA,NA,"orange","skyblue"),bty="n")
}
```

2000

2014

Long-range forecasts and comparison with APC models

Next, we produce plots showing long range forecasts including backtesting the latest five years of data.

To obtain the same amount of years in fitting, we input mortality experience from 1994 to 1998, and then use 1994 to 2010 as the test set. The public CDC database does not contain this data, so instead we use Human Mortality Database (HMD) data (for US males). For comparison, we include a Lee-Carter Age-Period-Cohort model, fitted using the package **StMoMo**. See Appendix B.2 in LRZ and corresponding figure 15.

```
# import the HMD data
mortHMD <- read.csv("hmd.csv",header=T)
mortHMD$rate <- mortHMD$D / mortHMD$L
mortHMD$y <- log(mortHMD$rate)
head(mortHMD)
```

```
##   X age year        L         D       rate          y
```

```
## 1 1   0 1933  997735.1 68438.11 0.068593465 -2.679558
## 2 2   1 1933 1028926.8 10329.16 0.010038770 -4.601301
## 3 3   2 1933 1100519.8  5140.05 0.004670566 -5.366475
## 4 4   3 1933 1128155.4  3759.88 0.003332768 -5.703952
## 5 5   4 1933 1156106.9  2932.59 0.002536608 -5.976928
## 6 6   5 1933 1212990.4  2537.53 0.002091962 -6.169653
```

Next, fit the Lee-Carter and GP models

```
agesFit <- 50:84; yearsFit <- 1994:2009
nAg <- length(agesFit); nYr <- length(yearsFit)

mortHMDfit <- dplyr::filter(mortHMD, age %in% agesFit, year %in% yearsFit)
Dmatrix <- matrix(mortHMDfit$D,nAg,nYr)
Lmatrix <- matrix(mortHMDfit$L,nAg,nYr)
LCfit <- StMoMo::fit(apc(), Dxt = Dmatrix, Ext = Lmatrix,
            ages = agesFit, years = yearsFit)

xHMD <- data.frame(age = mortHMDfit$age, year = mortHMDfit$year)
yHMD <- mortHMDfit$y
# quadratic age-trend
hmdModel_nug <- km(formula = ~x.age+I(x.age^2)+x.year,
               design = data.frame(x = xHMD),
               response = yHMD,
               nugget.estim=TRUE,
               covtype="gauss",
               optim.method="gen",
             control=list(max.generations=100,pop.size=100,
                          wait.generations=8,  solution.tolerance=1e-5))

nug_hmd <- hmdModel_nug@covariance@nugget
hmdModel <- km(formula = ~x.age+I(x.age^2)+x.year,
               design = hmdModel_nug@X, response = hmdModel_nug@y,
               noise.var = rep(nug_hmd,hmdModel_nug@n),
               coef.trend = hmdModel_nug@trend.coef,
               coef.cov = hmdModel_nug@covariance@range.val,
               coef.var = hmdModel_nug@covariance@sd2,
               covtype = hmdModel_nug@covariance@name)
```

Next, produce the forecasting plots for 2010:2040 (extrapolate 30 years out).

```
# Figure 15

agesForecast <- c(50,60,70,84)
lengthForecast <- 30
yearsForecast <- 1994:(2009+lengthForecast)
yearsLCsim <- 2010:(2009+lengthForecast)
yearsObserved <- 1994:2014
nYr <- length(yearsForecast)
nAg <- length(agesForecast)
# forecasting data.frame
xPred <- data.frame(age=rep(agesForecast,each=nYr),year=rep(yearsForecast,nAg))

# GP predictions
hmdPred <- predict(hmdModel, newdata=data.frame(x=xPred),cov.compute=TRUE,
            se.compute=TRUE,type="UK")
```

```r
nug_hmd <- hmdModel@noise.var[1]

xPred$m <- hmdPred$mean
xPred$lower95 <- hmdPred$lower95-1.96*sqrt(nug_hmd)
xPred$upper95 <- hmdPred$upper95+1.96*sqrt(nug_hmd)

#Lee-Carter predictions
LCratesHistoric <- predict(LCfit,
                           years=LCfit$years,
                           kt=c(LCfit$kt),
                           gc=c(LCfit$gc),
                           type="rates")
LCForecast <- forecast(LCfit,h=lengthForecast)
LCRates <- cbind(LCratesHistoric,LCForecast$rates)
LCsim <- simulate(LCfit,h=lengthForecast)    # simulate to build a CI

for(ag in agesForecast){
  ageStr <- toString(ag)
  LCm <- LCRates[ageStr,]
  LCquantile <- apply(LCsim$rates[ageStr,,],1,quantile, probs=c(0.05,0.95))

  xPredAg <- dplyr::filter(xPred,age == ag)
  # GP credible interval
  GPm <- exp(xPredAg$m)
  GPlower95 <- exp(xPredAg$lower95)
  GPupper95 <- exp(xPredAg$upper95)

  rateObserved <- dplyr::filter(mortHMD,age == ag, year %in% yearsObserved)$rate

  yLower <- min(rateObserved,GPlower95,LCm)
  yUpper <- max(rateObserved,GPupper95,LCm)

  plot(NULL,xlab="Year", main=paste("Age",ageStr),xlim=c(min(yearsForecast),
                                                          max(yearsForecast)),
       ylab="Mortality Rate",type="l",ylim=c(yLower,yUpper), xaxs="i",
       cex.lab=1.5,cex.axis=1.5, cex.main=1.5)

  # shade the GP and LC CI's
  polygon(c(yearsForecast,rev(yearsForecast)),c(GPupper95,rev(GPlower95)),col="skyblue")
  polygon(c(yearsLCsim,rev(yearsLCsim)),c(LCquantile[2,],rev(LCquantile[1,])),
          col="orange",density=25)

  lines(yearsForecast,LCm,col="red",lwd=2)
  lines(yearsLCsim,LCquantile[1,],col="red")
  lines(yearsLCsim,LCquantile[2,],col="red")

  lines(yearsForecast,GPm,col=4,lwd=2,lty=1)
  lines(yearsForecast,GPlower95,col=4,lty=1)
  lines(yearsForecast,GPupper95,col=4,lty=1)

  points(yearsObserved,rateObserved,pch=16,cex=1.5)

  legend("bottomleft", c("Lee-Carter", "GP", "Observed", "Lee-Carter 95% PI",
```
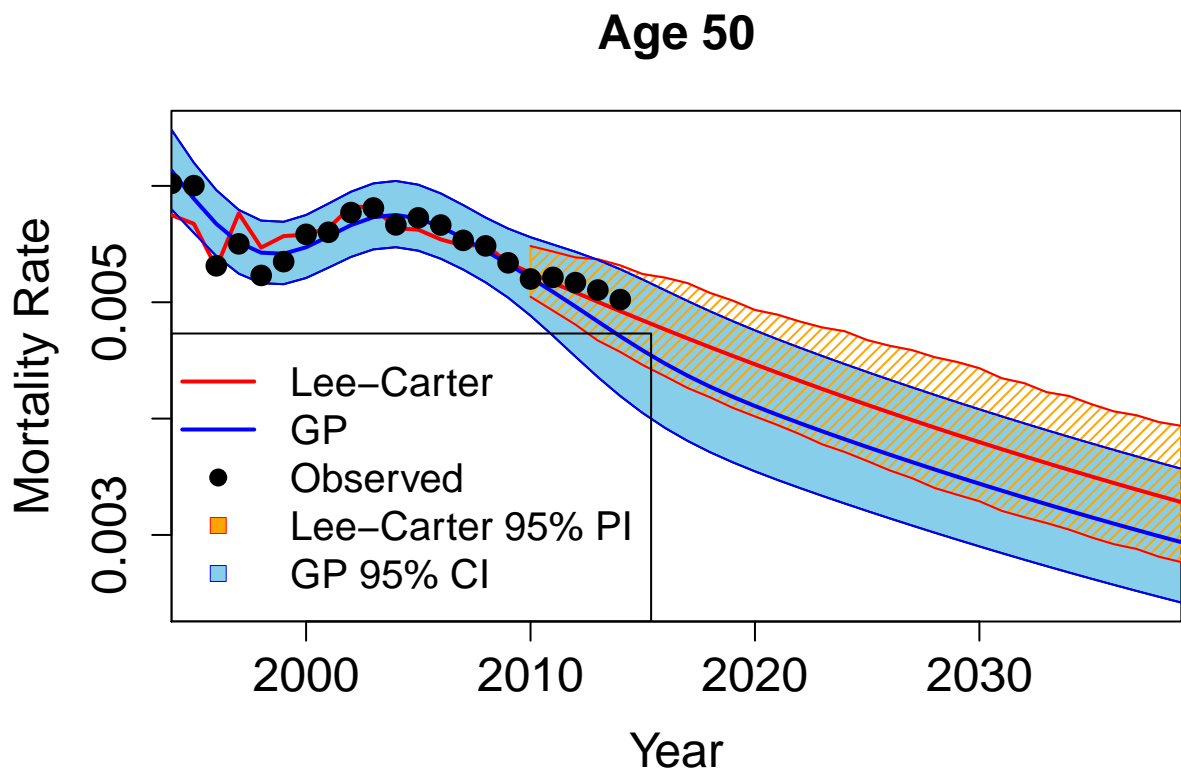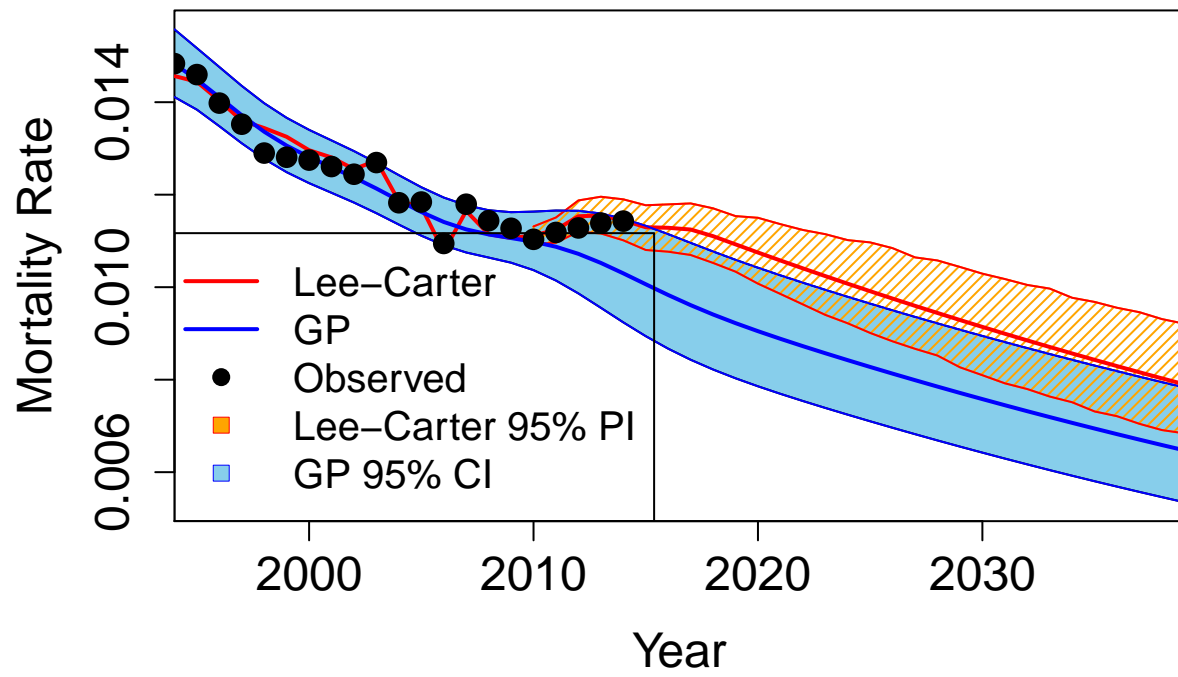
```
                              "GP 95% CI"),
       cex=1.25, col=c("red","blue",1), lty=c(1,1,0,0,0),lwd=c(2,2,NA,0,0),
       pch=c(NA,NA,16,22,22),pt.bg=c(NA,NA,NA,"orange","skyblue"))

}
```
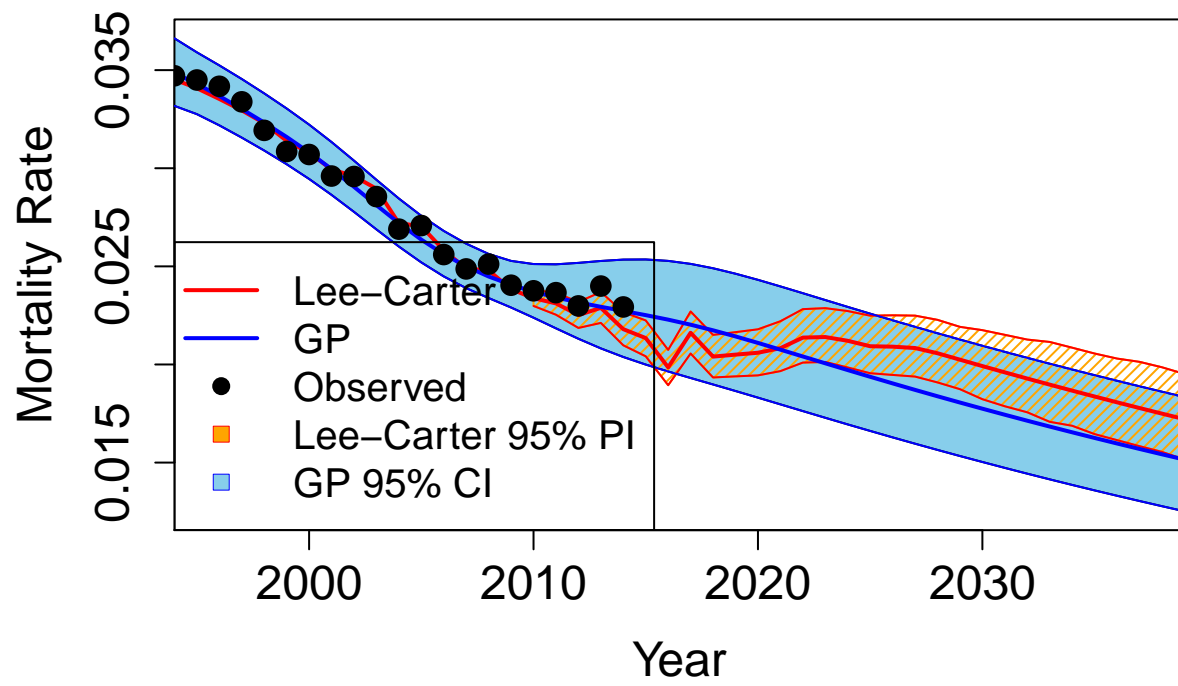
Age 50

Age 60

- Lee−Carter
- GP
- ● Observed
- ■ Lee−Carter 95% PI
- ■ GP 95% CI

Age 70

Age 84