R Notebook for Chapter 6: Mortality surface modeling

Companion Code to Gaussian Process Models for Quantitative Finance

Mike Ludkovski, Jimmy Risk

10/29/2024

This RMarkdown file presents an illustrative use of Gaussian Process surrogates for mortality surface modeling, linking to **Section 6.4** of the book.

Accessing Mortality Data

We make use of the following packages * `DiceKriging`: kriging methods for single-output GP (SOGP). * `rgenoud`: genetic optimization solver used for Maximum Likelihood Estimation of GP hyperparameters. * `data.table`: extension of `data.frame` to enhance data manipulation.

The script in the `createMortData.R` loads the csv files containing the Deaths and Exposures for the respective population.

```
library(dplyr)
library(rgenoud)
library(DiceKriging)
library(data.table)
library(plot3D)
source("createMortData.R")
```

The input data is an `R` dataframe with factors **age**, **year**, **deaths** and **exposures**. The corresponding log mortality rates are computed as

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

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

For the illustrations below we work with the dataset for Denmark Males.

```
mortData = createMortData(year_start=1990, year_end=2016,
age_start=60, age_end=80, sex="m", sex_cat="no")  # M=males
mortData = mortData[country %in% c("Denmark")]
head(mortData)
```

```
##    age year country       rate         y
## 1:  60 1990 Denmark 0.01561052 -4.159810
## 2:  60 1991 Denmark 0.01602873 -4.133373
## 3:  60 1992 Denmark 0.01762344 -4.038525
## 4:  60 1993 Denmark 0.01486386 -4.208822
## 5:  60 1994 Denmark 0.01553909 -4.164396
## 6:  60 1995 Denmark 0.01599507 -4.135475
```

Fitting the baseline GP model

For our Gaussian Process Regression models, 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) and the following parameters:

- *formula* determines the mean function $\mu(x)$; `formula = ~x.age` corresponds to a prior mean function that is linear in Age $\mu(x) = \beta_0 + \beta_{ag}^1 x_{ag}$
- *covtype* refers to kernel type, taken to be Matern-5/2, separable in Age and Year.
- *nugget.estim=TRUE* tells km() to infer the intrinsic (homoskedastic, input-independent) noise variance σ_ϵ^2 as part of the model.
- *optim.method="gen"* tells km() to utilize a genetic optimization algorithm from library **rgenoud**, which produces more reliable hyperparameter estimates and is recommended by **DiceKriging** authors.
- *control=...* are internal (recommended) parameters of the above optimization algorithm
- See package manual https://cran.r-project.org/web/packages/DiceKriging/DiceKriging.pdf for a more detailed explanation of **km()** options.

```
xMort <- data.frame(age = mortData$age, year = mortData$year)
yMort <- mortData$y
mortModel_nug <- km(formula = ~x.age,
                    design = data.frame(x = xMort), response = yMort,
                    nugget.estim=TRUE,
                    covtype="matern5_2",
                    optim.method="gen",
                    # the "control" parameters below handle speed versus risk of
                    # converging to local minima.  See "rgenoud" package for details
                    control=list(max.generations=100,pop.size=100,wait.generations=8,
                                 solution.tolerance=1e-5,trace=F))
```

```
mortModel_nug
```

```
##
## Call:
## km(formula = ~x.age, design = data.frame(x = xMort), response = yMort,
##     covtype = "matern5_2", nugget.estim = TRUE, optim.method = "gen",
##     control = list(max.generations = 100, pop.size = 100, wait.generations = 8,
##         solution.tolerance = 1e-05, trace = F))
##
## Trend  coeff.:
##                Estimate
##  (Intercept)    -9.8521
##        x.age     0.0919
##
## Covar. type  : matern5_2
## Covar. coeff.:
##                 Estimate
##  theta(x.age)    24.2764
## theta(x.year)     9.8826
##
## Variance estimate: 0.03688447
##
## Nugget effect estimate: 0.001787395
```

The above code output shows estimates of (in order) $\beta_0, \beta_{ag}^1, \theta_{ag}, \theta_{yr}, \eta^2$ and σ_ϵ^2. 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**:

```
nug <- mortModel_nug@covariance@nugget
mortModel <- km(formula = ~x.age, 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)
```

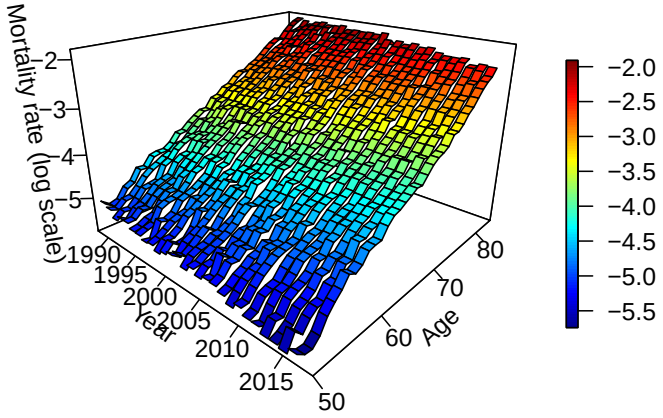
Figure 6.2: raw and predicted mortality surfaces

We use the **plot3D** library to visualize the raw log-mortality surface (the ribbons emphasize that data is collected annually) for Males in Denmark.

```
agesObserved <- 50:84
yearsObserved <- 1990:2016
mortData_ext = createMortData(year_start=min(yearsObserved), year_end=max(yearsObserved),
age_start=min(agesObserved), age_end=max(agesObserved), sex="m", sex_cat="no")
mortData_ext = mortData_ext[country %in% c("Denmark")]

rateObserved <- dplyr::filter(mortData_ext,age %in% agesObserved, year %in% yearsObserved)$rate
rateObserved <- matrix(log(rateObserved),nrow=length(yearsObserved));

ribbon3D(yearsObserved,agesObserved, z=rateObserved, along="y", expand=0.65,
                    border="black", zlim=c(-5.8,-1.8), phi=22, theta=42, xlim=c(1988,2018),
                    xlab="Year",ylab="Age",zlab="Mortality rate (log scale)", space=0.1,
                    colkey = list(length = 0.68, width = 0.4, dist=-0.1),
                    ticktype="detailed")
```



We then show the smooth surface based on GP projections from the previous **mortModel**. The predictive surface extends beyond the range of observed data: covering Ages $[55, 85]$ in the plot (beyond the training range of $[60, 80]$) and covering the future years of 2017–2024 relative to the training range of 1990–2016.

```
agesForecast <- 55:85
yearsForecast <- 1990:2024

# build data frame for desired forecasts, then call predict
xPredImage <- data.frame(expand.grid(agesForecast, yearsForecast))
colnames(xPredImage) <- c("age","year")

mortPredImage <- predict(mortModel, newdata=data.frame(x=xPredImage),
                    cov.compute=TRUE,
                    se.compute=TRUE,type="UK")
mortPredImage <- matrix(mortPredImage$m,nrow=length(yearsForecast),byrow=TRUE)

persp3D(yearsForecast,agesForecast, z=mortPredImage,border="black",
        xlab="Year", ylab="Age", zlab="Mortality rate (log scale)", zlim=c(-5.9,-1.8),
        colkey = list(length = 0.7, width = 0.5, dist=-0.1),
        ticktype="detailed", phi=22, theta=42, expand=0.65 )
```

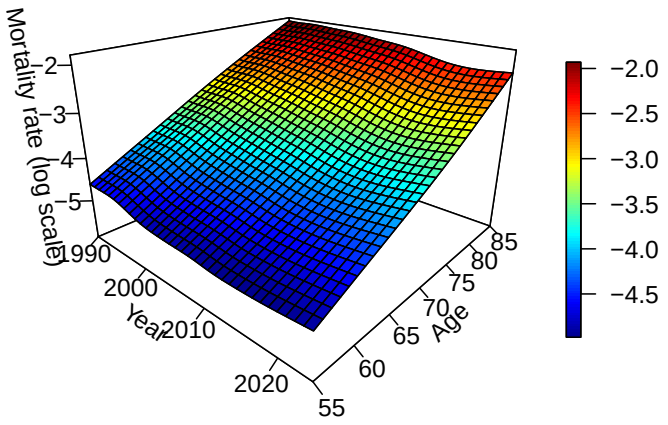


Figure 6.3 left: Mortality over time

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. Our implementation of building a model with noise variance equal to the nugget causes **predict()** to return credible bands for the latent mortality surface $f(\cdot)$ and not the observation process $Y(\cdot)$. Thus we need to add σ^2_ϵ to achieve the bands for the observed mortality y that would be relevant for coverage tests.

As illustration, we show evolution of mortality across time for a fixed Age. Decreasing trend corresponds to mortality improvement. Below, we additionally show the 95% credible intervals that quantify the (statistical) credibility of the smoothing. We use the library **dplyr** for easier data handling.

```
agesForecast <- 70
yearsForecast <- 2000:2026
yearsObserved <- 1999:2016
```

```r
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
y_sd = sqrt((mortPred$sd ^2) + nug)
xPred$lower95  = mortPred$mean - 1.96 * y_sd # predictive CI for y at 95% using z_0.975=1.96
xPred$upper95  = mortPred$mean + 1.96 * y_sd

for(ageObs in agesForecast){
  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,
       pch=18, main = paste("Age", ageObs),
       xlab="Year", ylab="Mortality Rate", cex.axis=1.5, cex.lab=1.5, cex=1.5,
       xlim=c(2000.5,2024), ylim=c(min(lower95Pred),max(upper95Pred)), bty="n", col="darkblue")
  lines(yearsForecast,ratePred, col=2, lwd=3)
  polygon(c(yearsForecast,rev(yearsForecast)),c(lower95Pred,rev(upper95Pred)),
          col="skyblue", density=25,lwd=1)

  legend("topright",c("Observed","Predicted","95% Credible Band"),
       lwd=c(2,2,2), lty=c(0,1,2), pch=c(18,NA,NA),
       col=c("darkblue",2,"skyblue"), cex=1.3, bty="n")
}
```

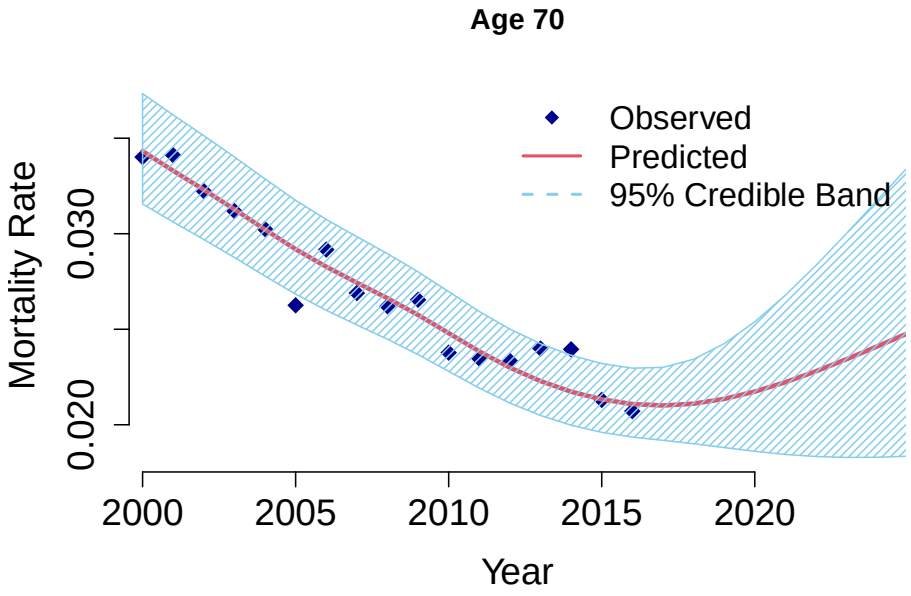


Figure 6.3 right: Mortality Improvement

It is common to interpret a mortality surface via the (annual) mortality *improvement factors* which measure longevity changes year-over-year. The raw annual percentage mortality improvement is:

$$MI^{raw}(\mathbf{x}) = 1 - \frac{\exp\left(y(x_{ag}, x_{yr})\right)}{\exp\left(y(x_{ag}, x_{yr}-1)\right)}.$$

The smoothed improvement factors are obtained by replacing y's by the GP model posterior $m(\mathbf{x})$'s:

$$MI^{GP}(\mathbf{x}) := 1 - \frac{\exp\left(m_*(x_{ag}, x_{yr})\right)}{\exp\left(m_*(x_{ag}, x_{yr}-1)\right)}.$$

The following code produces improvement rates $MI^{GP}(\mathbf{x})$ for every odd calendar year and plots them.

```
agesForecast <- 60:80
yStart = 2008; yEnd = 2018;
yearsForecast <- yStart:yEnd


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% (yStart+1):yEnd)$m
backwardPred <- dplyr::filter(xPred, age %in% agesForecast, year %in% yStart:(yEnd-1))$m
mibackgp <- 1-exp(forwardPred)/exp(backwardPred)  # GP-smoothed improvement rate
```

```
xPred$mibackgp <- NA
xPred$mibackgp[which(xPred$year > yStart)] <- mibackgp

miPlot <- dplyr::filter(xPred, age %in% agesForecast, year == yEnd)$mibackgp
plot(c(0,100), c(0,0), type="l", lwd=1, xlab="Age", xaxs="i",
     ylab="YoY Mortality Improvement Rate", cex.axis=1.5, cex.lab=1.5, cex=1.5,
     ylim=c(-0.01, 0.04), xlim=c(60,80),bty='n')
for(yr in seq(yStart+1,yEnd-1,2)){
  # plot the MI for that year
  miPlot <- dplyr::filter(xPred, age %in% agesForecast, year == yr)$mibackgp
  lines(agesForecast, miPlot, col=(yr-2007)/2, lty=(yr-2007)/2, lwd=2)

}
legend("bottomright", legend=seq(yStart+1,yEnd-1,by=2), col=c(1:5,1),
       lwd=rep(2,5), lty=c(1:5,1), cex=1.5, bty='n')
```

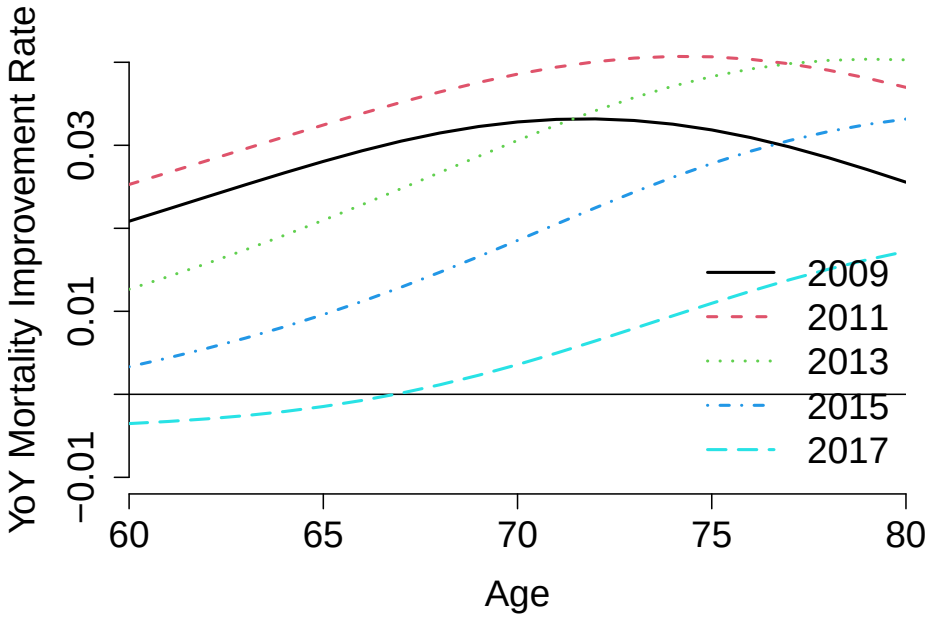


Figure 6.4 left: Life Expectancy Boxplot

To generate scenarios for period life expectancy, we train a model on a wider range of Ages. In the Danish dataset, observations are reliable up to age 103, so we use that for training the new `mortDataE`. Due to the larger training set (of size 27 x 44 = 1188 training inputs), this takes a couple of minutes.

```
mortDataE = createMortData(year_start=1990, year_end=2016,
                           age_start=60, age_end=103, sex="m", sex_cat="no")
mortDataE = mortDataE[country %in% c("Denmark")]
control.params = list(max.generations=100,pop.size=100,
                      wait.generations=8,solution.tolerance=1e-5,trace=F)

# fit linear age-year trend model
xMortE <- data.frame(age = mortDataE$age, year = mortDataE$year)
yMortE <- mortDataE$y
```

```
mortModelE_nug <- km(formula = ~x.age+x.year,  #linear in age, linear in year
                         design = data.frame(x = xMortE), response = yMortE,
                         nugget.estim=TRUE,
                         covtype="matern5_2",
                         optim.method="gen",
                         control=control.params)

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

We next generate 200 posterior samples (i.e., scenarios $\omega_1, \ldots, \omega_n$) based on the GP posterior mortality curves in Age for the period Life Expectancy (LE) e_{65} at age 65 from the point of view of Year 2017 through 2026. After aggregating across samples, we visualize the resulting boxplots of simulated future e_{65}, computed as

$$e_{65}(t;\omega) = \sum_{x=66}^{103} \exp(f_*(x,t;\omega)) \cdot \left(\prod_{u=66}^{x-1} (1 - e^{f_*(u,t;\omega)}) \right) \cdot (x - 65).$$

We find that the GP model projects an average gain in LE of about 1 year between 2016 and 2026, although the range is from 0 (no gain) to 2+ years of gain.

```
nsim <- 200
maxAge <- 105
agesForecast <- seq(65,maxAge, 1)
lifeExp65 <- array(0, dim=c(10,nsim))  # period life expectancy for each scenario
for (j in 1:10) {
  yearsForecast <- 2016+j #:2026

  xPredE <- data.frame(age=agesForecast,year=yearsForecast)
  mortPredE <- predict(mortModelE, newdata=data.frame(x=xPredE),
                       cov.compute=TRUE,
                       se.compute=TRUE,type="UK")

  sim <- simulate(mortModelE, nsim=nsim, newdata = data.frame(x=xPredE), cond=TRUE,
                  nugget.sim=nug_E/100)

  # first two terms: dying in the first year, dying in exactly the second year
  lifeExp65[j,] = exp(sim[,1]) *1 + exp(sim[,2])*(1-exp(sim[,1])) *2
  for (ag in 3:(maxAge-65+1))  # add probability of dying at age 'ag'
      lifeExp65[j,] = lifeExp65[j,] + exp(sim[,ag])*apply( 1-exp(sim[,1:(ag-1)]),1,prod) * ag
}

# Display a boxplot for the distribution of future e_{65}
par(bty="n")
boxplot(t(lifeExp65), pch=18, main="", names=2017:2026, xlab="Year",
        ylab="Life Expectancy at 65", cex.axis=1.3, cex.lab=1.3, cex=1.3,col="lightblue3")
```

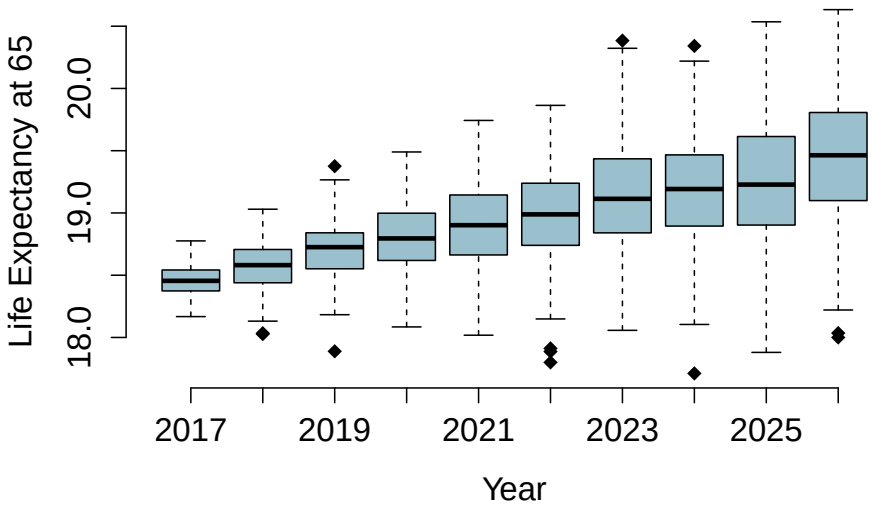


Figure 6.4 right: Comparison of Mean-function specifications

km() takes in an object of class **formula** to specify the prior mean function $\mu(\cdot)$, just like with linear OLS model `lm()` and generalized linear model `glm()` fitting. The following code produces models with constant, linear and quadratic prior means.

```
mortData3 <- dplyr::filter(mortData, age %in% 60:80, year %in% 1999:2016)
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="matern5_2",
                    optim.method="gen",
                    control=control.params)

nug_int <- mortModel3int_nug@covariance@nugget
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 in both age and year
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="matern5_2",
                    optim.method="gen",
                    control=control.params)

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="matern5_2",
                         optim.method="gen",
                         control=control.params)

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_{ag}, θ_{yr}.

```
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)    -3.5476
##
## Covar. type  : matern5_2
## Covar. coeff.:
##
##  theta(x.age)    32.6871
## theta(x.year)    34.0000
##
```

```
## Variance: 1.262178
```

```
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)    40.8070
##        x.age     0.0965
##       x.year    -0.0255
##
## Covar. type  : matern5_2
## Covar. coeff.:
##
##  theta(x.age)     5.1058
## theta(x.year)     4.4595
##
## Variance: 0.00174879
```

```
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)    44.2132
##        x.age     0.0292
##   I(x.age^2)     0.0005
##       x.year    -0.0260
##
## Covar. type  : matern5_2
## Covar. coeff.:
##
##  theta(x.age)     4.9115
## theta(x.year)     3.2769
##
## Variance: 0.001201928
```

We can also print out the respective log-likelihood scores that can be converted into BIC scores and ultimately into Bayes Factors for comparing goodness-of-fit across models. As expected, the quadratic prior mean has the highest log-likelihood because it has the most degrees of freedom. However, after applying the BIC penalties, the linear-mean model ends up with the highest BIC.

```
c(mortModel3int_nug@logLik,mortModel3lin_nug@logLik,mortModel3quad_nug@logLik)
```

```
## [1] 616.2822 633.9895 635.7808
```

Next, we use these three GP models to forecast temporal mortality trends at ages 60, 70, 80, to see how well they can extrapolate. In the plots below we compare their Age-specific forecasts for years 2005–2030, where 2017 and onward is an extrapolation compared to the training data.

Note: we plot all the GP predictions on the same plot since we have the natural ordering that (log-)mortality is much lower at Age 60 compared to Age 70 compared to Age 80.

```
# Setup the test data.frame to be displayed
agesForecast <- c(60,70,80)
yearsForecast <- 2005:2030
nYr <- length(yearsForecast)
nAg <- length(agesForecast)
xPred <- data.frame(age=rep(agesForecast,each=nYr), year=rep(yearsForecast,nAg))

# Setup the raw training data that was used
agesObserved <- 60:80
yearsObserved <- 1999:2016
rateObserved <- dplyr::filter(mortData,age %in% agesObserved, year %in% yearsForecast)$rate

# predict using the 3 GP models above on the same test set
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$mInt <- (mortPredInt$mean)
xPred$mLin <- (mortPredLin$mean)
xPred$mQuad <- (mortPredQuad$mean)


xPredag <- dplyr::filter(xPred,age==80)
rateObserved <- log(dplyr::filter(mortData, year %in% yearsObserved, age == 80)$rate)
# set-up the overall plot layout
plot(yearsObserved, rateObserved,
     bty='n', main="", pch=19, cex.axis=1.3, cex.lab=1.3, cex=1.3,
     xlab="Year", ylab="Log Mortality Rate", ylim=c(-5, -2), xlim=range(yearsForecast))


for(ag in agesForecast){
  rateObserved <- log(dplyr::filter(mortData, year %in% yearsObserved, age == ag)$rate)
  # add points representing training data of observed log-mortality
  points(yearsObserved,rateObserved,col="darkblue",
       pch=19,cex=1.3)
  text(2008, rateObserved[2]+0.1*(ag != 70), paste("Age ",ag), cex=1.15)

  # add lines representing GP posterior mean
  xPredag <- dplyr::filter(xPred,age==ag)
  lines(yearsForecast, xPredag$mInt, col=2, lty=3, lwd=3)
  lines(yearsForecast, xPredag$mLin, col=3, lty=2, lwd=3)
```

```
  lines(yearsForecast, xPredag$mQuad, col=4, lty=1, lwd=3)

}
legend("topright",legend=c("Intercept Only","Linear Trend","Quadratic Trend"),
       lwd=rep(3,3), col=2:4, lty=3:1, cex=1.2, bty="n")
```