Uncertainty Quantification (ACM41000)
Assignemt 3

Ian Towey

04128591

May 4, 2018

Contents

1 ODE with constant coefficients

1.1 Estimate α_0 and α_1 and 95%CI

Estimate of differetial equation parameters

$$\alpha_0 = 0.3290305$$
$$\alpha_1 = 0.1203271$$

With 95% confidence intervals

$$0.3268669 < \alpha_0 < 0.3311941$$
$$0.1188871 < \alpha_1 < 0.1217670$$

1.2 Estimate $\hat{f}$ and 95%CI

$\hat{f}$ estimates and 95% CI can be seen in Figure 4

1.3 Accuracy of fit analysis

The $SSE = 0.001773525$, this is close to zero so the model should have a small random error somponent, this makes the model more useful for predictive purposes.

The $ISE = 0.000002091086e$ is also very small with indictes the curve fits the unknown density f very well.

1.4 Model comparison

- Nelson-Siegel soultion fits the majority of the data well, but does not fit the data less than 5 years very well(Figure 1)
- Svensson solution almost seems to interpolate the data and fits very well, but the resulting curve does not seem to be continous with discontinuities at the second and third data point(Figure 2)
- Data2LD model fits the data well and is better than the Nelson-Siegel model fitting the points less than 5 years, but not as well as Svensson(Figure 3)

2 Two ODE models one with estimated forcing function and one time-varing parameter

3 Dynamical Systems

Error :: singular matrix running Data2LD line 103 Q3.R

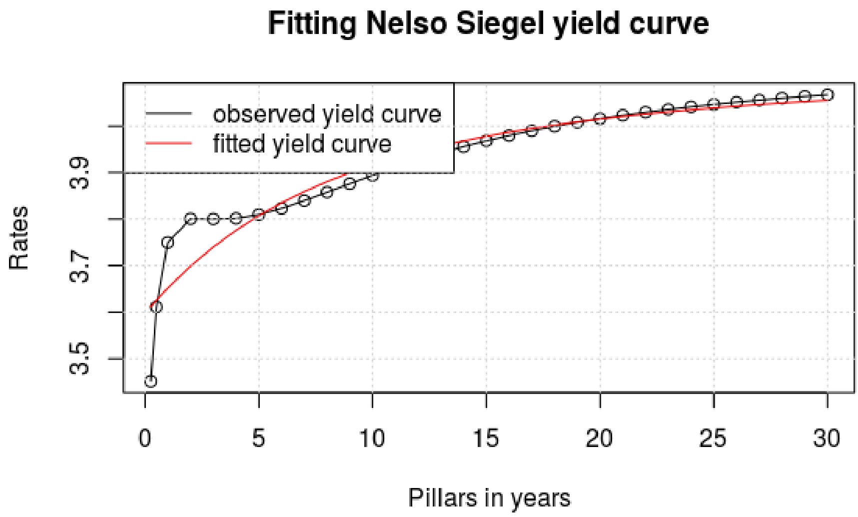

Figure 1: Nelson Seigel

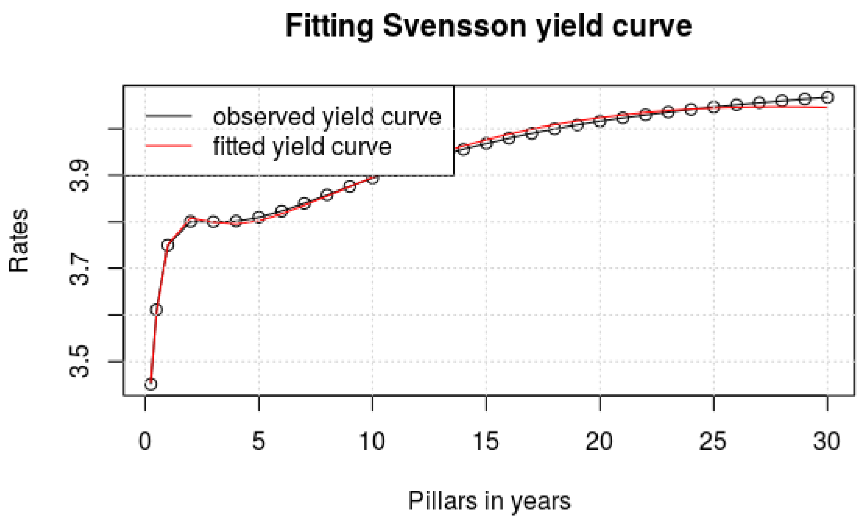

Figure 2: Svensson.png

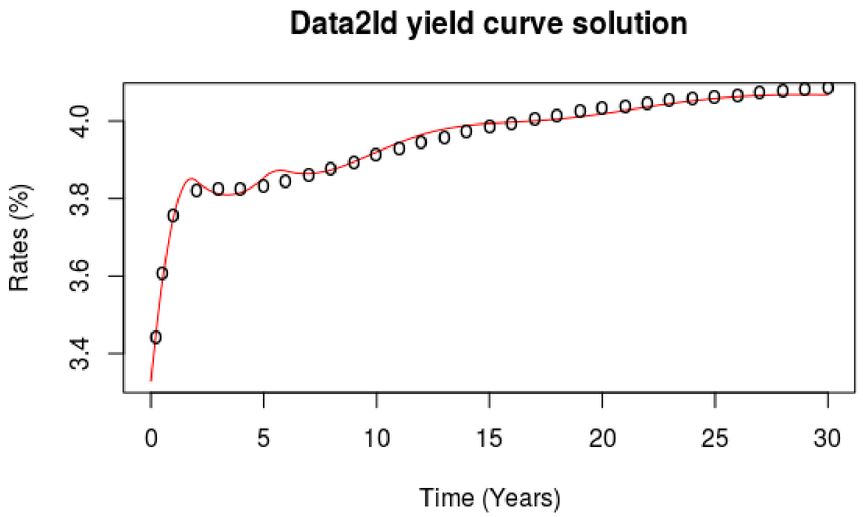

Figure 3: Data2ld

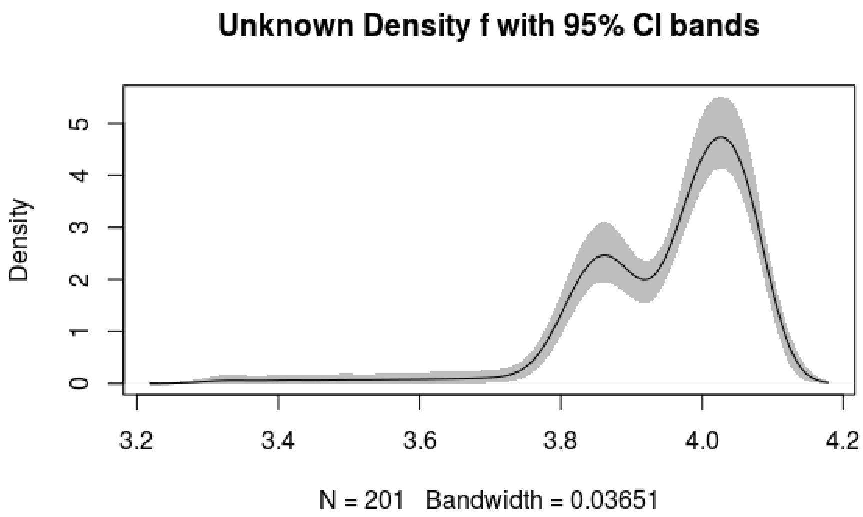

Figure 4: Density

1

A Q1 Code

```
#(part a)
#define data
library(YieldCurve)
data(ECBYieldCurve)
rate.ECB <- as.vector(first(ECBYieldCurve,'1 day'))
length(rate.ECB)
maturity.ECB <- c(0.25,0.5,seq(1,30,by=1))
length(maturity.ECB)
YCrng <- c(0,max(maturity.ECB))
#define model
# Set up the constant basis
conbasis <- create.constant.basis(YCrng)
# Define the two constant coefficient functions
coef1 <- make.coef(fun=conbasis, parvec=0.25, estimate=TRUE,
    coeftype="beta")
coef2 <- make.coef(fun=conbasis, parvec=0.25,  estimate=TRUE,
    coeftype="beta")
#  Set up the coefficient list
coefList  <- vector("list",2)
coefList[[1]] <- coef1
coefList[[2]] <- coef2

#  check coefficient list
coefCheckList <- coefCheck(coefList)
coefList      <- coefCheckList$coefList
ntheta        <- coefCheckList$ntheta
print(paste("ntheta = ",ntheta))
# Define the two terms in the homogeneous part of the equation
Xterm0 <- make.Xterm(variable=1, derivative=1, ncoef=1, factor
    =-1)
Xterm1 <- make.Xterm(variable=1, derivative=2, ncoef=2, factor
    =-1)
# Set up the XList vector of length two
XList <- vector("list",2)
XList[[1]] <- Xterm0
XList[[2]] <- Xterm1

#  Define the single differential equation in the model
YCVariable <- make.variable(order=3, XList=XList)
YCVariableList=vector("list",1)
YCVariableList[[1]] <- YCVariable

#check model
YCVariableList <- modelCheck(YCVariableList, coefList)

#  Define the List array containing the data
yList1 = list(argvals=maturity.ECB, y=rate.ECB)
yList <- vector("list",1)
yList[[1]] <- yList1

plot(maturity.ECB, rate.ECB,main="Data21d yield curve solution",
    xlab=c("Pillars in years"), ylab=c("Rates"),type="o")

knots     <- 1:21
knots     <- c(0,1,1,2,2,3,3,4,seq(5,30,len=8))
length(knots)
norder    <- 6
nbasis    <- 24
YCBasis <- create.bspline.basis(YCrng,nbasis,norder,knots)
```

```
YCTimefine <- seq(0,30,len=201)
YCBasisfine <- eval.basis(YCTimefine, YCBasis)
matplot(YCTimefine, YCBasisfine, main="Data2ld yield curve
    solution",type="l", xlim=c(0,30), ylim=c(0,1), xlab="Time t"
    , ylab="Basis functions phi(t)")

XbasisList <- vector("list",1)
XbasisList[[1]] <- YCBasis
# An evaluation of the criterion at the initial values
# Optimization of the criterion
#  algorithm constants
dbglev   <-  1    #  debugging level
iterlim  <- 50    #  maximum number of iterations
convrg  <- c(1e-8, 1e-4)  #  convergence criterion
rhoVec <- 0.995  #  light smoothing

Data2LDResult <- Data2LD(yList, XbasisList, YCVariableList,
    coefList, rhoVec)
Data2LDResultOpt <- Data2LD.opt(yList, XbasisList,
    YCVariableList, coefList,  rhoVec, convrg, iterlim, dbglev)

stderr  <- sqrt(diag( Data2LDResult$Var.theta))

Data2LDResultOpt$theta
thetaDn <- Data2LDResultOpt$theta - 2*stderr
thetaUp <- Data2LDResultOpt$theta + 2*stderr

cbind(Data2LDResultOpt$theta,thetaDn,thetaUp)

YCfd     <- Data2LDResult$XfdParList[[1]]$fd
YCfine   <- eval.fd(YCTimefine, YCfd)

plot(YCTimefine, YCfine, main="Data2ld yield curve solution",
    type="l",col="red", xlim=c(0,30),xlab="Time (Years)", ylab="
    Rates (%)")
points(maturity.ECB, rate.ECB, pch="o")

Data2LDResultOpt$theta
MSE         <- Data2LDResult$MSE        # Mean squared error for
    fit to data
DpMSE       <- Data2LDResult$DpMSE      #  gradient with respect
    to parameter values
D2ppMSE     <- Data2LDResult$D2ppMSE    #  Hessian matrix
XfdParList <- Data2LDResult$XfdParList #  List of fdPar objects
    for variable values
df          <- Data2LDResult$df         #  Degrees of freedom for
     fit
gcv         <- Data2LDResult$gcv        #  Generalized cross-
    validation coefficient
ISE         <- Data2LDResult$ISE        #  Size of second term,
    integrated squared error
SSE         <- MSE*df                   #  SSE
Var.theta   <- Data2LDResult$Var.theta  #  Estimate sampling
    variance for parameters


#(part b)
f <-YCfine
fit1 <- density(f)
fit2 <- replicate(10000,{
  x <- sample(xx, replace=TRUE);
  density(x, from=min(fit1$x), to=max(fit1$x))$y
```

```
}
)
fit3 <- apply(fit2, 1, quantile, c(0.025,0.975) )
plot(fit1, ylim=range(fit3), main="Unknown Density f with 95% CI
     bands")
polygon( c(fit1$x, rev(fit1$x)), c(fit3[1,], rev(fit3[2,])), col
    ='grey', border=F)
lines(fit1)
```

B Q2 Code

```
data <- read.csv(file='/home/ian/Desktop/ACM41000-Uncertainty-
    Quantification/assignement3/GMSL.csv', header=TRUE)
dim(data)
head(data, n=30)
plot(data, type = 'l')
rng<- c(0,nrow(data))

data$idx <- rownames(data)
GMSL.min <- -min(data$GMSL)
data = data %>% mutate(year = floor(Date), rescale.GMSL = GMSL +
     GMSL.min)
data.max.tmp.by.year =  data %>% group_by(year) %>% summarise(
    max.rescale.GMSL = max(rescale.GMSL))

pulse.idx = c(0,as.integer((data %>% inner_join(data.max.tmp.by.
    year, by = "year") %>% filter(rescale.GMSL == max.rescale.
    GMSL) %>%select(idx))$idx))


which.max(data[data$Date < 1994,]$GMSL)
which.max(data[data$Date >= 1994 & data$Date < 1995 ,]$GMSL)

conbasis <- create.constant.basis(HeadImpactRng)
# Define the three constant coefficient functions
coef1 <- make.coef(fun=conbasis, parvec=0.1, estimate=TRUE,
    coeftype="beta")
coef2 <- make.coef(fun=conbasis, parvec=0.1,  estimate=TRUE,
    coeftype="alpha")

coefList  <- vector("list",2)
coefList[[1]] <- coef1
coefList[[2]] <- coef2

coefCheckList <- coefCheck(coefList)
coefList      <- coefCheckList$coefList
ntheta        <- coefCheckList$ntheta
print(paste("ntheta = ",ntheta))
# Define the two terms in the homogeneous part of the equation
Xterm0 <- make.Xterm(variable=1, derivative=0, ncoef=1, factor
    =-1)
# Set up the XList vector of length two
XList <- vector("list",1)
XList[[1]] <- Xterm0

Pulsebasis <- create.bspline.basis(rng, 27, 1, pulse.idx)
Pulsefd    <- fd(matrix(c(0,1,0),27,1),Pulsebasis)

Fterm      <- make.Fterm(ncoef=2, Ufd=Pulsefd, factor=1)
# Set up the forcing term list of length one
FList      <- vector("list",1)
FList[[1]] <- Fterm

#  Define the single differential equation in the model
GMSL.ODE <- make.variable(order=1, XList=XList, FList=FList)

GMSL.ODEList=vector("list",1)
GMSL.ODEList[[1]] <- GMSL.ODE
#  check the object for internal consistency
GMSL.ODEList <- modelCheck(GMSL.ODEList, coefList)
```

```r
yList1 = list(argvals=data$Date, y=data$GMSL)
yList <- vector("list",1)
yList[[1]] <- yList1

knots      <- 0:nrow(data)
norder     <- 6
nbasis     <- 899
GMSL.Basis <- create.bspline.basis(rng,nbasis,norder,knots)

GMSLTimefine <- seq(0,nrow(data),len=1001)
GMSLTBasisfine <- eval.basis(GMSLTimefine, GMSL.Basis)
matplot(GMSLTimefine, GMSLTBasisfine, type="l", xlim=c(0,nrow(
    data)), ylim=c(0,1),
        xlab="Time t", ylab="Basis functions phi(t)")
lines(c(14,14), c(0,1), lty=2)
lines(c(15,15), c(0,1), lty=2)

XbasisList <- vector("list",1)
XbasisList[[1]] <- GMSL.Basis


dbglev   <-  1     #  debugging level
iterlim  <- 50     #  maximum number of iterations
convrg   <- c(1e-8, 1e-4)  #  convergence criterion
rhoVec <- 0.995  #  light smoothing

Data2LDResult <- Data2LD(yList, XbasisList, GMSL.ODEList,
    coefList, rhoVec)
Data2LDResultOpt <- Data2LD.opt(yList, XbasisList, GMSL.ODEList,
     coefList,  rhoVec, convrg, iterlim, dbglev)
```

C Q3 Code

```
library(stringr)
library(dplyr)
library(R.utils)
library(R.cache)
library(Matrix)
library(fda)

#init env
rm(list=ls())
source("/home/ian/Desktop/ACM41000-Uncertainty-Quantification/
    assignement3/Data2LD_Fix/Data2LD_Fix/R/make.coef.R")
source("/home/ian/Desktop/ACM41000-Uncertainty-Quantification/
    assignement3/Data2LD_Fix/Data2LD_Fix/R/coefCheck.R")

#import data
measles.data <- read.csv(file='/home/ian/Desktop/ACM41000-
    Uncertainty-Quantification/assignement3/Measles.csv', header
    =TRUE, na.strings = c("*"), colClasses = rep("integer", 10))
num.years = measles.data %>% distinct(YY) %>% nrow
measles.data[is.na(measles.data)] <- 0

#sort data
measles.data = measles.data %>% mutate(dt=as.integer(paste(YY,
    str_pad(MM,2,pad="0"),str_pad(X.DD,2,pad="0"), sep=""))) %>%
     select(-c(1,2,3)) %>% arrange(-desc(dt)) %>% mutate(wk_num=
    row_number()) %>% select(-c(8))
measles.data <- measles.data[seq(1,316,1),]
measles.rng<- c(0,nrow(measles.data))
plot(measles.data$wk_num, measles.data$London, type = 'l')
plot(measles.data$wk_num, measles.data$Newcastle, type = 'l')

#set up vars
city.population <- list(London=8171000,Bristol=436000,Liverpool
    =747500, Manchester=661000, Newcastle=269400, Birmingham
    =1105000, Sheffield=494000)
conbasis <- create.constant.basis(measles.rng)

#create the coefficients
coefList  <- vector("list",length(names(city.population))^2)
i <- 1
for (city in names(city.population)){
  for (c in names(city.population)){
    coefList[[i]] <-make.coef(fun=conbasis, parvec=0.1, estimate
        =TRUE, coeftype="beta")
    i <- i + 1
  }
}
coefCheckList <- coefCheck(coefList)

# Define the terms in the homogeneous part of the equation
XListList = list()
i = 1
for (c in names(city.population)){
  j = 1
  XList <- vector("list",length(city.population))
  for (city in names(city.population)){
    XList[[j]] <- make.Xterm(variable=1, derivative=0, ncoef=i,
        factor=-1)
    i <- i + 1
    j <- j + 1
```

```
  }
  XListList[[c]] <- XList
}

#  Define the system of differential equation in the model
measles.ODEList=vector("list",length(city.population))
idx = 1
for (city in names(city.population)){
  measles.ODEList[[idx]] <- make.variable(name=city,order=1,
      XList=XListList[[city]])
  idx <- idx + 1
}
#  check the object for internal consistency
measles.ODEList <- modelCheck(measles.ODEList, coefList)

#setup data
yList <- vector("list",7)
idx = 1
for (city in names(city.population)){
  yList[[idx]] <- list(argvals=seq(1,nrow(measles.data)), y=
      measles.data[[city]])
  idx <- idx + 1
}

#define basis functions
norder    <- 7
break.array <- seq(25,nrow(measles.data),52)
nbreaks   <- length(break.array)
nbasis    <- norder + nbreaks - 2
measles.Basis  <- create.bspline.basis(measles.rng, nbasis)
XbasisList <- vector("list",7)
XbasisList[[1]] <- measles.Basis
XbasisList[[2]] <- measles.Basis
XbasisList[[3]] <- measles.Basis
XbasisList[[4]] <- measles.Basis
XbasisList[[5]] <- measles.Basis
XbasisList[[6]] <- measles.Basis
XbasisList[[7]] <- measles.Basis

GMSLTimefine <- seq(0,nrow(measles.data),len=1001)
GMSLTBasisfine <- eval.basis(GMSLTimefine, measles.Basis)
matplot(GMSLTimefine, GMSLTBasisfine, type="l", xlim=c(0,nrow(
    measles.data)), ylim=c(0,1),
        xlab="Time t", ylab="Basis functions phi(t)")

dbglev   <-  1    #  debugging level
iterlim  <- 50    #  maximum number of iterations
convrg   <- c(1e-8, 1e-4)  #  convergence criterion
rhoVec <- rep(0.5,7)  #  light smoothing

length(yList[[1]]$y)
length(XbasisList)
length(measles.ODEList)
length(coefList)
length(rhoVec)

Data2LDResult <- Data2LD(yList, XbasisList, measles.ODEList,
    coefList, rhoVec)
Data2LDResultOpt <- Data2LD.opt(yList, XbasisList, measles.
    ODEList, coefList,  rhoVec, convrg, iterlim, dbglev)
```