Smoothing Revisted and Model Fitting

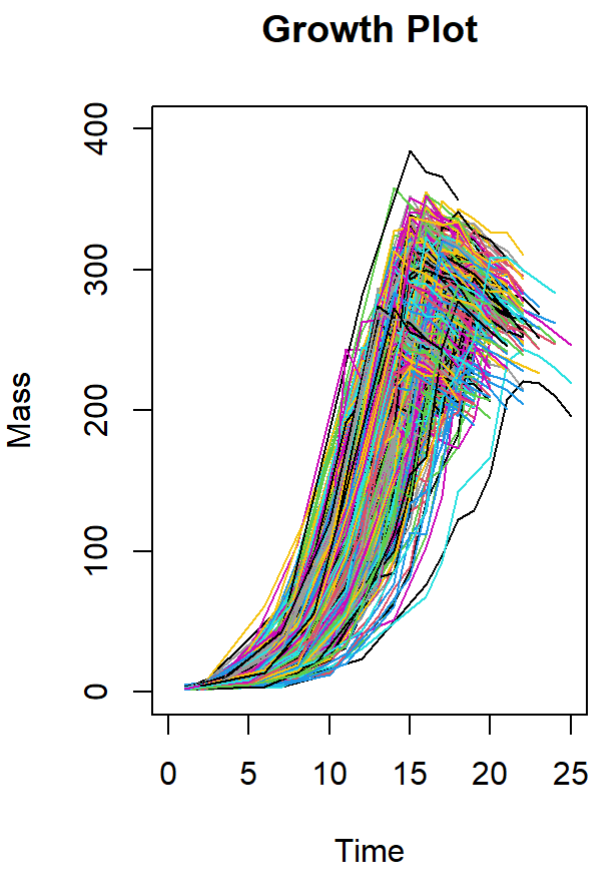
Import library

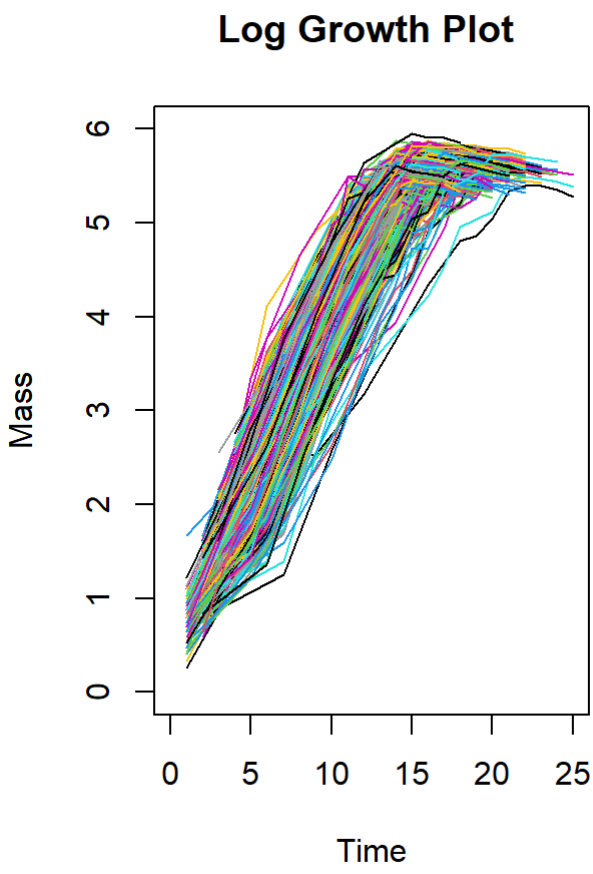
```
library(Matrix)
library(fdasrvf)
library(lme4)
library(pedigreemm)
library(npreg)
library(ggplot2)
library(plotly)
```

Import data and re-scale time to unit interval.

```
setwd("D:/KCL_2023-2027_PhD/Year 1/Genetics/FMEMs-quantitative-genetics/R_code")
TRFUN25PUP4 = read.delim("TRFUN25PUP4.DAT",header = FALSE)
names(TRFUN25PUP4)<-c("id","sire","dam","trait","x")
df <- data.frame(TRFUN25PUP4)

FirstUniqueIdPos <- which(duplicated(df$id) == FALSE)
N = length(FirstUniqueIdPos) # N = 873 subjects
n = length(df$id) # n = 6860 observations
age_list <- split(df$x,df$id)
trait_list <- split(df$trait,df$id)

age_list_new <- list()
for (i in 1:N){
  age_list_new[[i]] = (age_list[[i]]-min(age_list[[i]]))/(max(age_list[[i]])-min(age_list
[[i]]))
}

df$x_rescaled <- unsplit(age_list_new,df$id)
```

Plot the raw data



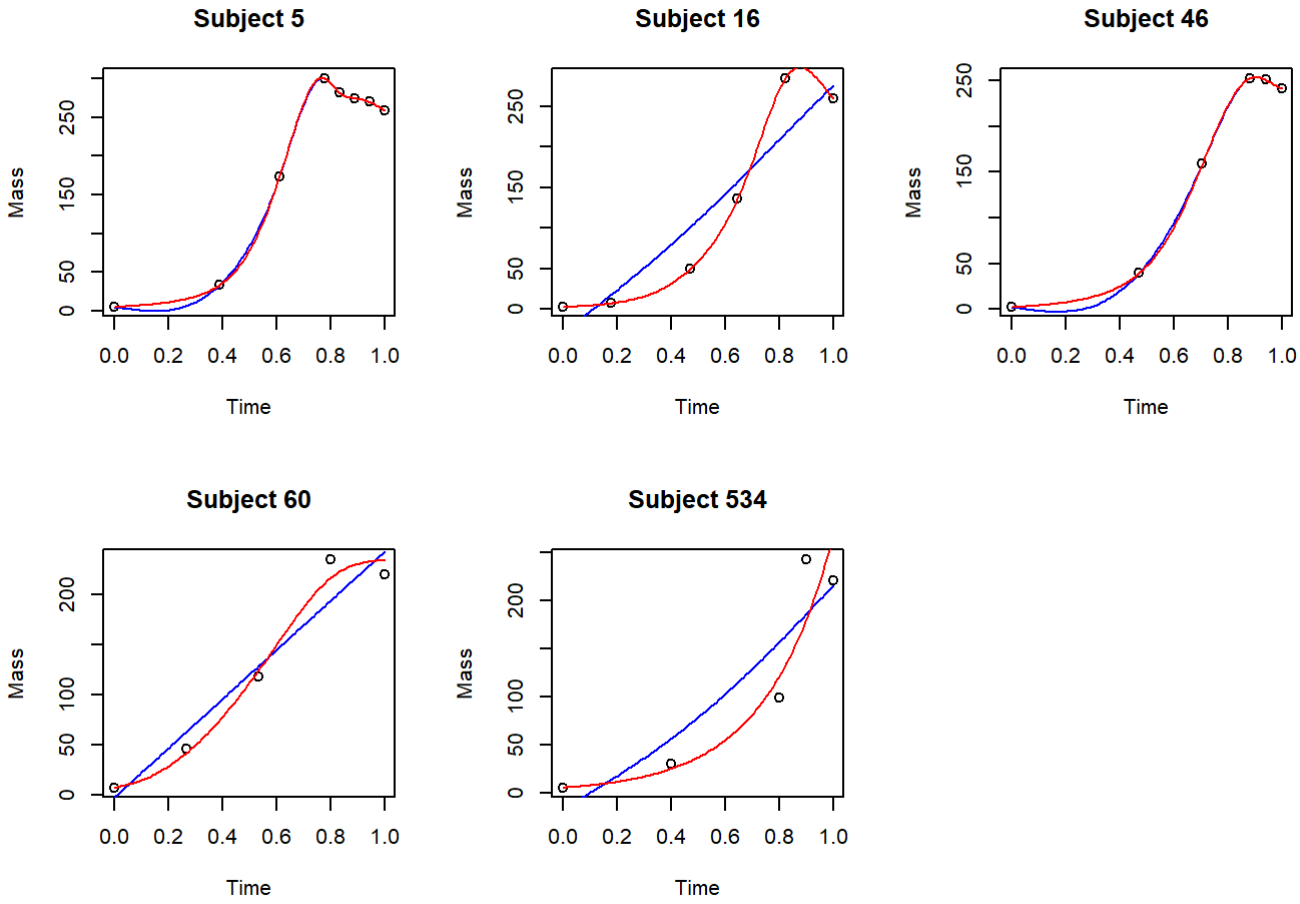


Data smoothing

Use penalised smoothing spline as basis functions. Here we take two ways to smooth the data. First, smooth body mass on its original. Second, smooth growth curves on the logarithmic scale and then take exponential to recover body mass. This imposes positive smoothing (same methodology as `smooth.pos()` in the **fda** package). We also report the smoothing parameter λ_s selected by GCV.

```
agefine <- seq(0,1,length=100) # create a fine time grid
mass_hat <- matrix(0,100,N) # store smooth growth curves on original scale
logmass_hat <- matrix(0,100,N) # store smooth log growth curves
pred_mass <- matrix(0,100,N) # store the smoothed mass recovered by taking exponential
lambda_mass <- rep(0,N) # smoothing parameter used for each growth curve
lambda_logmass <- rep(0,N) # smoothing parameter used for each log growth curve
for (i in 1:N){
  ss_mass <- smooth.spline(age_list_new[[i]], trait_list[[i]], cv=FALSE,
                           all.knots=TRUE)
  ss_logmass <- smooth.spline(age_list_new[[i]], log(trait_list[[i]]), cv=FALSE,
                              all.knots=TRUE) # all distinct points as knots
  mass_hat[,i] <- predict(ss_mass,agefine)$y
  logmass_hat[,i] <- predict(ss_logmass, agefine)$y
  pred_mass[,i] <- exp(predict(ss_logmass,agefine)$y)
  lambda_mass[i] <- ss_mass$lambda
  lambda_logmass[i] <- ss_logmass$lambda
}
```

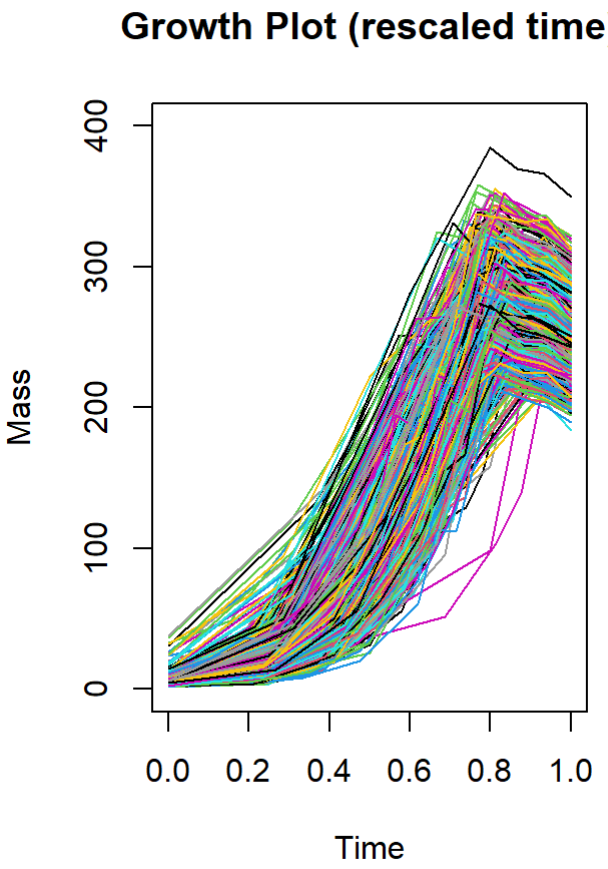
Some smoothed curves have negative values near $t = 0$, let us plot them and the corresponding original data points. There are 77 questionable smoothed growth curves, and here we show 5 of them. On each plot, black points represent the data measurements; blue curve represents smoothing on the original scale; red curve represents smoothing by imposing positive constraint.

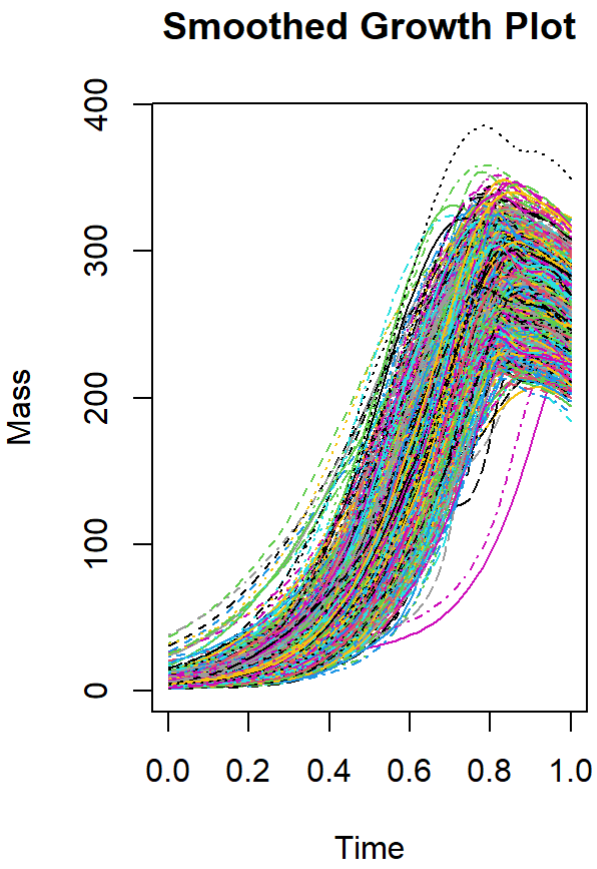


Sampling points are not taken at fixed times as they vary in number and location. The largest number of measurements per individual is 14 and the smallest is 5. Mass is measured more frequently in the last days of the time period. The irregular sampling points lead to two problems: for subjects with fewer measurements, GCV will select very large λ, and fitted curves approach to standard linear regression to the data (e.g. Subject 16, Subject 60); for subjects with more sparse measurements around the starting period, the fitted curves have negative values near $t = 0$ (e.g. Subject 5, Subject 46).

If we compare the two methods to smooth the growth curves, we find that positive smoothing performs better in general. So we continue with positive smoothing and manually adjust the smoothing parameter λ for subjects where the initial smoothing results are unsatisfactory, i.e. Subject 534 with $\lambda_{534} = 93480.4$.

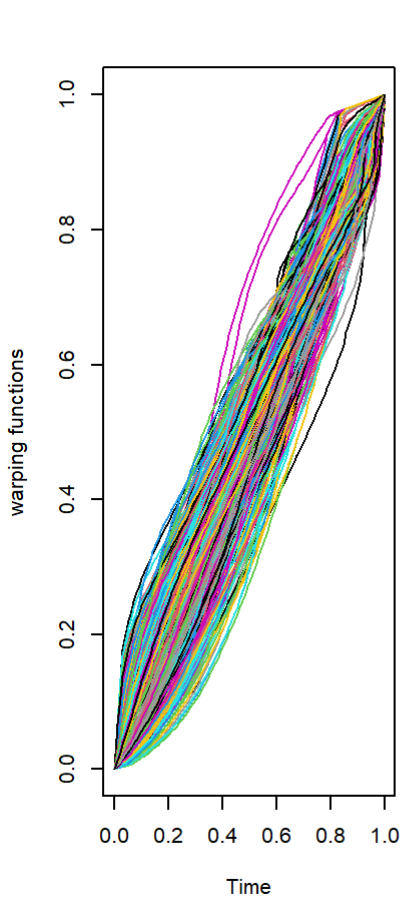
The most smoothing parameter λs selected by GCV are quite small. Let us restrict $\lambda \leq 10^{-4}$ and re-smooth growth curves. The adjusted smooth curves are store in the matrix `trait_hat`.

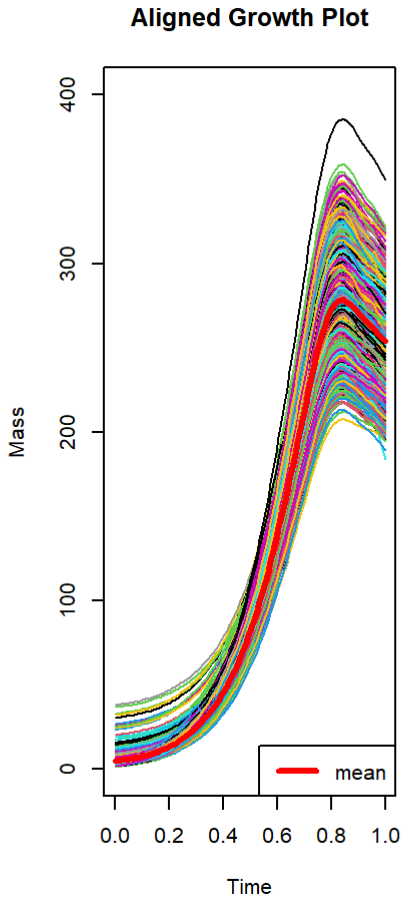


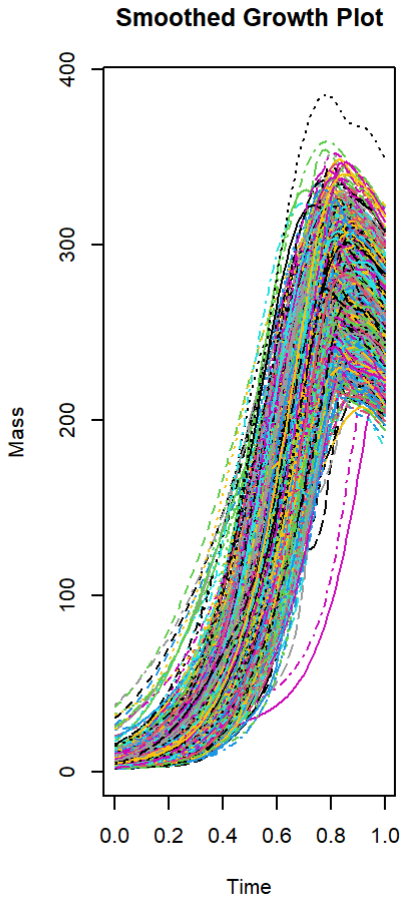


Curve registration

```
aligned_mass_process <- time_warping(f=trait_hat, time=agefine)
aligned_mass_curve <- aligned_mass_process$fn
aligned_mean <- aligned_mass_process$fmean
warping_funs <- aligned_mass_process$warping_functions


par(mfrow=c(1,3))
plot(c(0,1), c(0,1), type = 'n',xlab = 'Time',
     ylab = 'warping functions')
for (i in 1:N){
  lines(agefine, warping_funs[,i], type="l" , col=i)
}


plot(c(0,1), c(0,400), type = 'n',xlab = 'Time',
     ylab = 'Mass', main = 'Aligned Growth Plot')
for (i in 1:N){
  lines(agefine, aligned_mass_curve[,i], type="l",col=i)
}
lines(agefine, aligned_mean, lwd = 3.0, col="red")
legend("bottomright", legend="mean", lwd=3.0, col="red")

matplot(agefine, trait_hat, col=1:N, type = "l", xlab="Time", ylab="Mass", main="Smoothed Growt
h Plot")
```

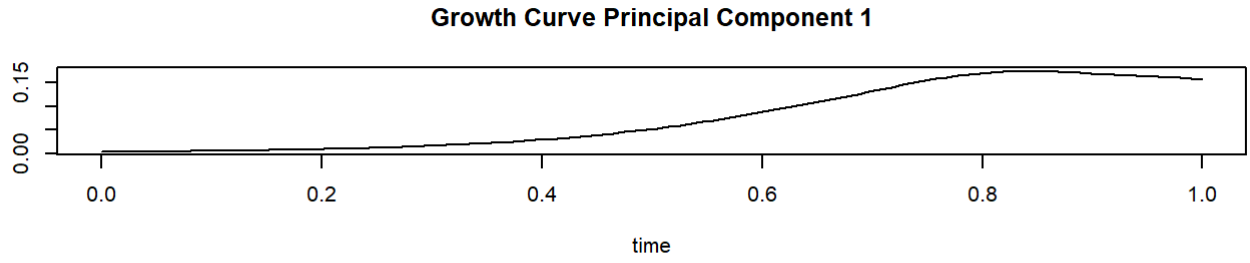


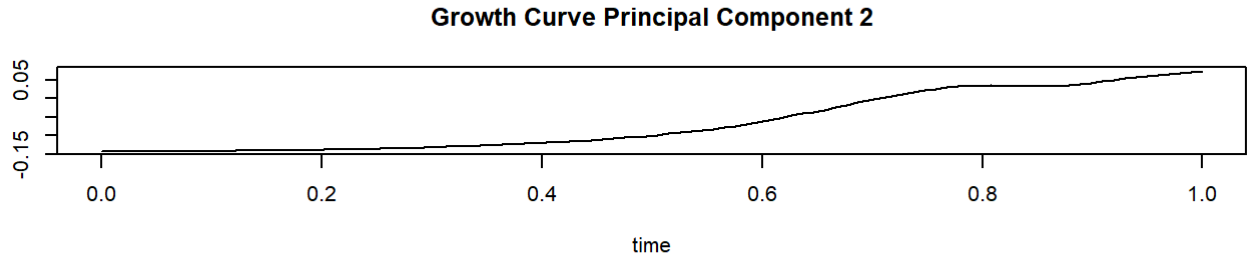


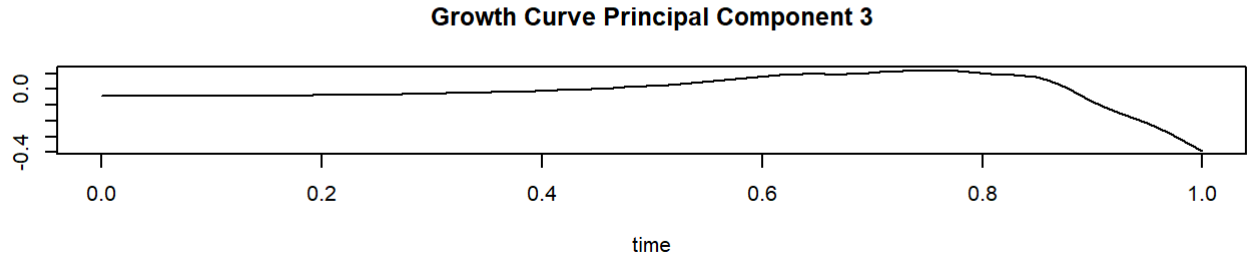


Functional Principal Component Analysis

```
fpcaobj_mass <- prcomp(x=t(aligned_mass_curve), retx = TRUE, center = TRUE, rank. = 3)
eigen_mass <- fpcaobj_mass$rotation # eigen vectors

par(mfrow=c(3,1))
for (i in 1:3) {
  plot(agefine, eigen_mass[, i], type = "l",
       xlab = "time", ylab = "",
       main = paste("Growth Curve Principal Component", i))
}
```








```
## Test for orthogonality
eigen_mass[,1] %*% eigen_mass[,2]
```

```
##              [,1]
## [1,] 2.341877e-16
```

```
eigen_mass[,1] %*% eigen_mass[,3]
```

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

```
eigen_mass[,2] %*% eigen_mass[,3]
```

```
##              [,1]
## [1,] -5.20417e-17
```

Fit Genetic Functional Mixed-Effect Model

Step 1: Combine the subject ids, aligned mass and principal components into a data frame which will be used to fit the mixed-effect model.

```
subjectID <- rep(unique(df$id), each=100)
trait_pred <- c(aligned_mass_curve)
basis1 <- rep(eigen_mass[,1], times = 873)
basis2 <- rep(eigen_mass[,2], times = 873)
basis3 <- rep(eigen_mass[,3], times = 873)
new_df <- data.frame(subjectID, trait_pred, basis1, basis2, basis3)
names(new_df) <- c("subjectID", "trait_hat", "basis1", "basis2", "basis3")
```

Step 2: Calculate the pedigree and the additive genetic relationship matrix A.

```
pos = df$id[FirstUniqueIdPos] # extract ids for all subjects
sire_id = df$sire[FirstUniqueIdPos] # extract ids for sire
dam_id = df$dam[FirstUniqueIdPos] # extract ids for dam

pede <- editPed(sire = sire_id, dam = dam_id, label = pos)
ped<- with(pede, pedigree(label=label, sire=sire, dam=dam))
A <- getA(ped)[163:1035,163:1035]
```

Step 3: Fit both fixed and random effects using the same basis.

```
fmeFormula <- trait_hat ~ new_df$basis1 + new_df$basis2 + new_df$basis3 +
  (-1 + new_df$basis1 + new_df$basis2 + new_df$basis3 | new_df$subjectID) +
  (-1 + new_df$basis1 + new_df$basis2 + new_df$basis3 | new_df$subjectID) # LME model formula
system.time(
  fit_sameBasis <- fit_genetic_fmm(formula= fmeFormula, data=new_df, A = A, phi = eigen_mass)
) # user  system  elapsed
```

```
##    用户    系统    流逝
##  821.45   69.21 1188.19
```

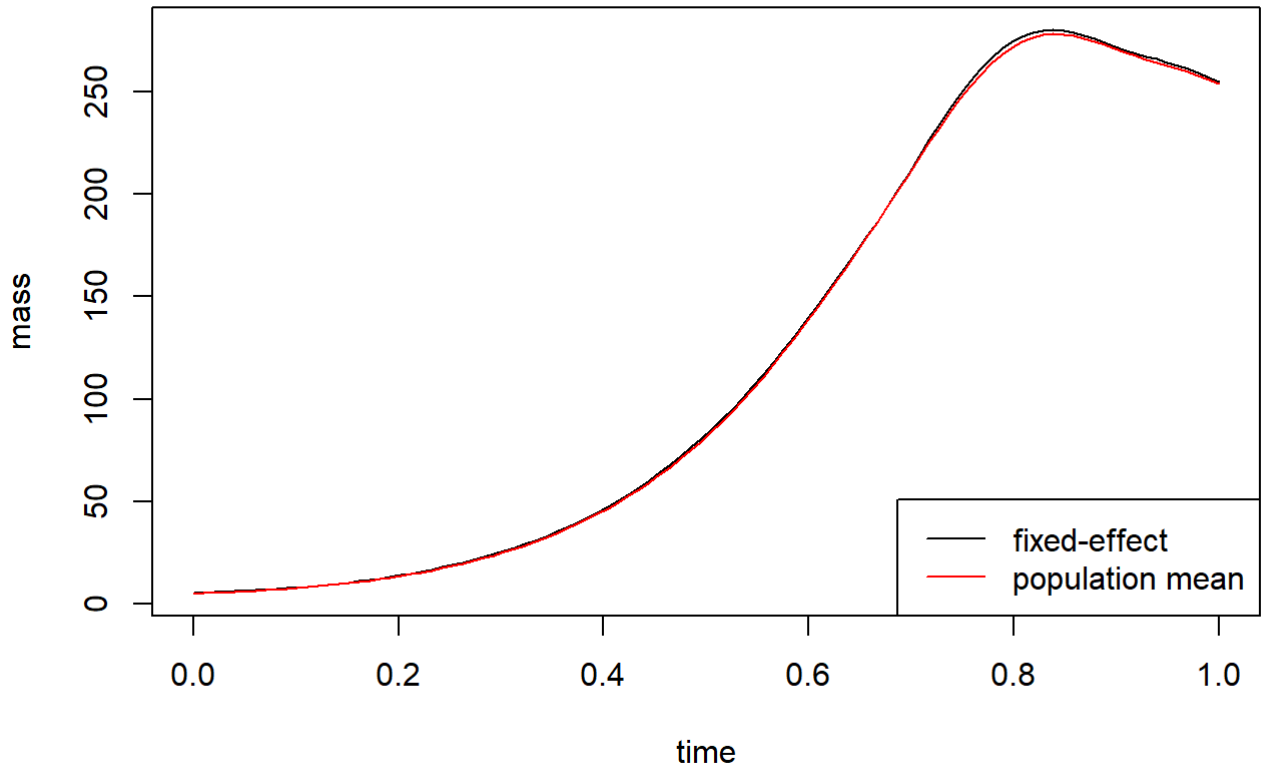
```
summary(fit_sameBasis)
```

```
## Linear mixed model fit by REML ['lmerMod']
##
## REML criterion at convergence: 292682.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -15.0483  -0.4289  -0.0130   0.4347  20.6842
##
## Random effects:
##  Groups             Name          Variance  Std.Dev. Corr
##  new_df.subjectID   new_df$basis1 22950.555 151.494
##                     new_df$basis2   503.572  22.440   0.04
##                     new_df$basis3   222.096  14.903   0.16 -0.15
##  new_df.subjectID.1 new_df$basis1 10209.379 101.041
##                     new_df$basis2   201.876  14.208  -0.11
##                     new_df$basis3    68.893   8.300  -0.44  0.48
##  Residual                             1.349   1.161
## Number of obs: 87300, groups:  new_df$subjectID, 873
##
## Fixed effects:
##                Estimate Std. Error t value
## (Intercept)     12.6225     0.1382  91.311
## new_df$basis1 1513.0756     8.2734 182.885
## new_df$basis2   89.9552     1.4913  60.319
## new_df$basis3    6.7518     0.7429   9.089
##
## Correlation of Fixed Effects:
##             (Intr) nw_d$1 nw_d$2
## new_df$bss1 -0.126
## new_df$bss2  0.604 -0.073
## new_df$bss3  0.124 -0.105  0.161
```

Step 4: Visualise the results

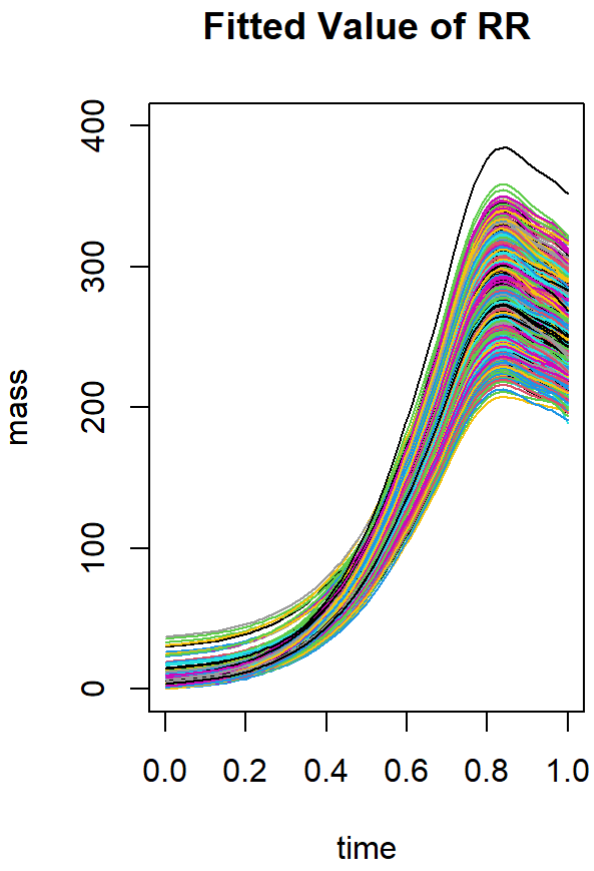
1. Extract the fixed effect and compare with the population mean calculated from the aligned data.

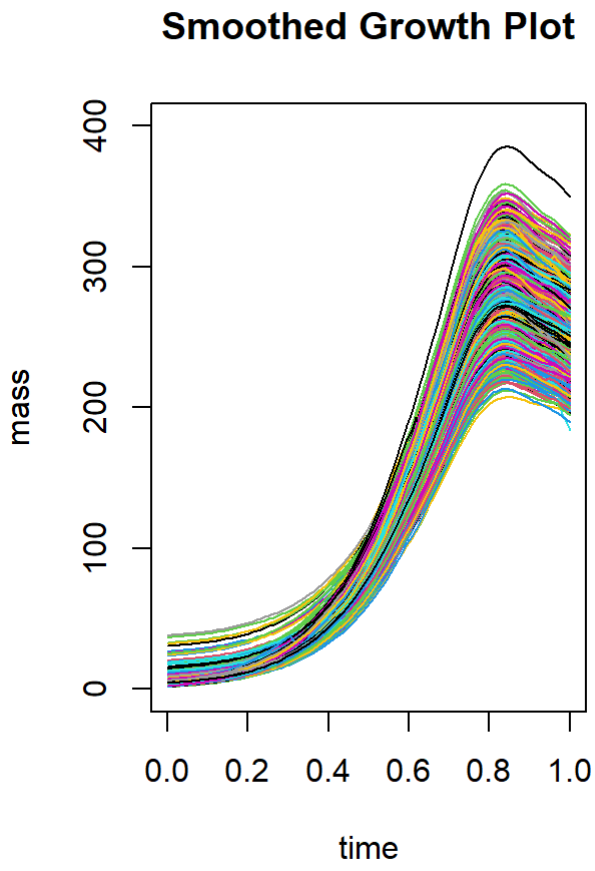
```
fe_coefs <- fixef(fit_sameBasis) # extract the fixed-effect coefs
fixef <- eigen_mass%*%fe_coefs[2:4] + fe_coefs[1]

par(mfrow=c(1,1))
plot(agefine, fixef, type = "l", xlab="time", ylab="mass")
lines(agefine, aligned_mean, type="l", col="red")
legend("bottomright", legend=c("fixed-effect", "population mean"), col=c("black", "red"), lwd=
1.0)
```



2. Plot the fitted curves and compare with the original data.

```
fitted_list <- split(fitted(fit_sameBasis), new_df$subjectID) # fitted value
par(mfrow=c(1,2))
plot(c(0,1), c(0,400), type="n", xlab="time", ylab="mass", main="Fitted Value of RR")
for (i in 1:N){
  lines(agefine, fitted_list[[i]], type="l", col=i)
}
plot(c(0,1), c(0,400), type="n", xlab="time", ylab="mass", main="Smoothed Growth Plot")
for (i in 1:N){
  lines(agefine, aligned_mass_curve[,i], type="l", col=i)
}
```

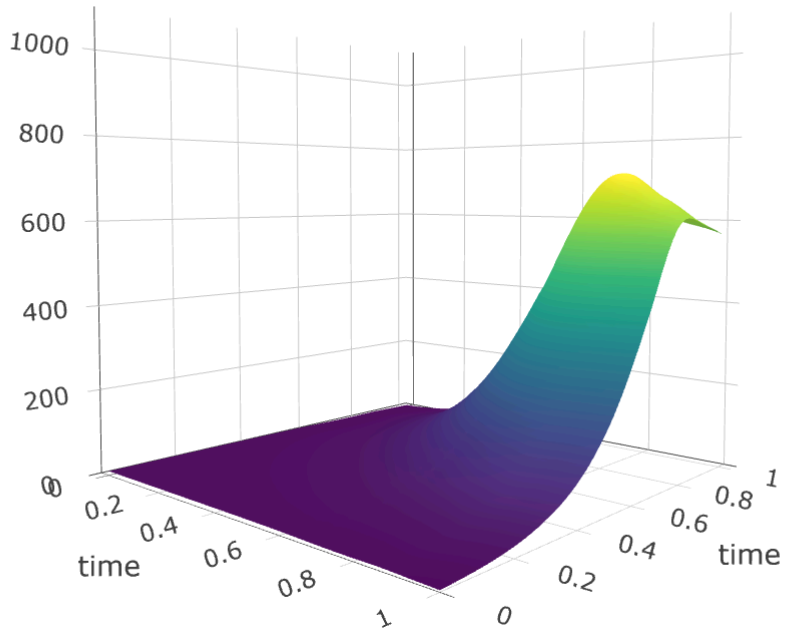




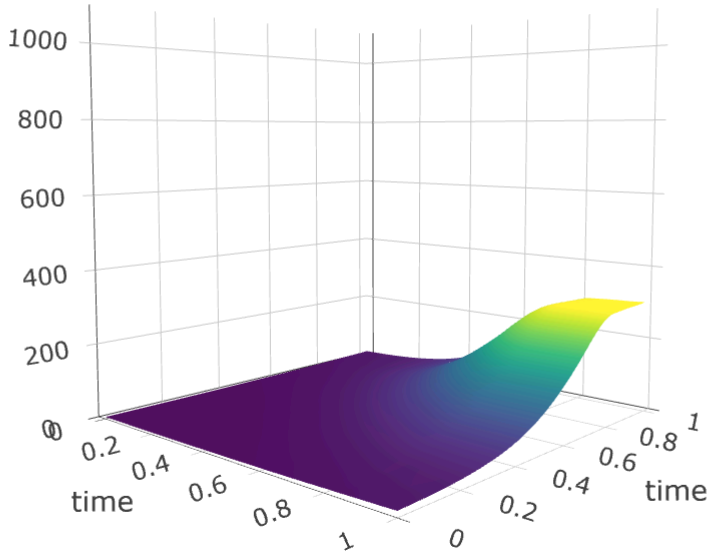
3. Extract the genetic and environmental covariance matrices and convert them to functions.

```
VC <- as.matrix(as.data.frame(VarCorr(fit_sameBasis))["vcov"])

CG <- matrix(0, 3,3) # genetic covariance matrix
CG[1:3, 1:3] <- VC[1:3]
CG[1,2] <- VC[4]
CG[1,3] <- VC[5]
CG[2,3] <- VC[6]
CG[2,1] <- CG[1,2]
CG[3,2] <- CG[2,3]
CG[3,1] <- CG[1,3]


CE <- matrix(0, 3,3) # environment covariance matrix
CE[1:3, 1:3] <- VC[7:9]
CE[1,2] <- VC[10]
CE[1,3] <- VC[11]
CE[2,3] <- VC[12]
CE[2,1] <- CE[1,2]
CE[3,2] <- CE[2,3]
CE[3,1] <- CE[1,3]

### Convert to genetic covariance function
CG_fun <- eigen_mass %*% CG %*% t(eigen_mass)
### environmental covariance function
CE_fun <- eigen_mass %*% CE %*% t(eigen_mass)
### Phenotypic covariance function
P_fun <- CG_fun + CE_fun
```

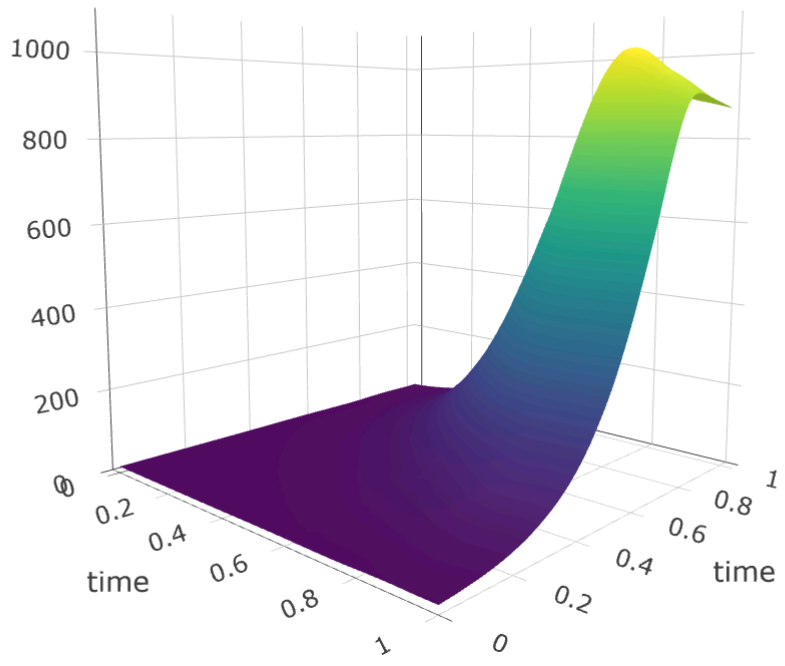
Genetic Variance Function



Environment Variance Function



Phenotypic Variance Function


1000
800
600
400
200
0
0
0.2
0.4
0.6
0.8
1
time
0
0.2
0.4
0.6
0.8
1
time