Practical 1

18/02/2022

Longitudinal and Time-To-Event data analysis

Practical 1

Question 1

We will analyse data on serum cholesterol from the National Cooperative Gallstone Study (NCGS).

Data description

Data from the National Cooperative Gallstone Study (NCGS)

Aim of the study: the assessment of the safety of the drug chenodiol for the treatment of cholesterol gallstones In this study, patients were randomly assigned to high-dose (750 mg per day), low-dose (375 mg per day), or placebo.

This dataset consists of a subset of data on patients who had floating gallstones and who were assigned to the high-dose and placebo groups.

In the NCGS it was suggested that chenodiol would dissolve gallstones but in doing so might increase levels of serum cholesterol.

As a result, serum cholesterol (mg/dL) was measured at baseline and at 6, 12, 20, and 24 months of follow-up.

Many cholesterol measurements are missing because of missed visits, laboratory specimens were lost or inadequate, or patient follow-up was terminated.

Variable List:

Treatment Group (1=High dose chenodiol, 2-Placebo), Subject ID, Response at Baseline, Response at Month 6, Response at Month 12, Response at Month 20, Response at Month 24.

Note: Response = Serum cholesterol (. denotes missing value).

We start with analysing the first three observations (baseline, month 6, month 12) and use only subjects with complete data for these time points. The data for this subset is in the file dataex1.txt. The data set is in wide format.

```
#clean the R environment
rm(list=ls())
#libraries to be installed
#ggplot2: producing nicer plots
if(!require(ggplot2)) {
  install.packages("ggplot2"); require(ggplot2)}
#tidyr: allows for easier data munipulation
if(!require(tidyr)) {
  install.packages("tidyr"); require(tidyr)}
#mvtnorm: allows to simulate from multivariate normal
if(!require(mvtnorm)) {
  install.packages("mvtnorm"); require(mvtnorm)}
#knitr: allows to produce nicer tables as summary
```

```
if(!require(knitr)) {
  install.packages("mvtnorm"); require(knitr)}
#set your working directory
#use setwd()
#or do it manually
#Read the data set (wide format)
data<-read.table("dataex1.txt",header=T)
```

(a) Compute the correlations between observations at baseline, month 6 and month 12 for the placebo group and for the month 12 group. Comment on the results.

Solution We need to compute the correlation matrices among the responses at the three different time points (baseline, month 6, month 12) for both the placebo and the treatment group. In order to compute the matrices, we need to create two distinct data sets, one of each group. Then, we can compute the correlation matrices.

```
#Create data set for the treatment group
treat<-subset(data,data$grp==1)
#Create data set for the placebo group
plac<-subset(data,data$grp==2)
#correlations for the treatment group
corr_treat<-cor(treat[,3:5])
print(corr_treat)
```

```
##           t1        t2        t3
## t1 1.0000000 0.7059545 0.6226680
## t2 0.7059545 1.0000000 0.6695283
## t3 0.6226680 0.6695283 1.0000000
```

```
#correlations for the placebo group
corr_plac<-cor(plac[,3:5])
print(corr_plac)
```

```
##           t1        t2        t3
## t1 1.0000000 0.8081594 0.8323153
## t2 0.8081594 1.0000000 0.8874039
## t3 0.8323153 0.8874039 1.0000000
```

We can observe a positive correlation among the responses at the different time points (baseline, month 6, month 12) both in the placebo group and the treatment group. However, in the placebo group, the correlations between the responses are slightly higher. This result is due to the fact that repeated observations are usually (positively) correlated.

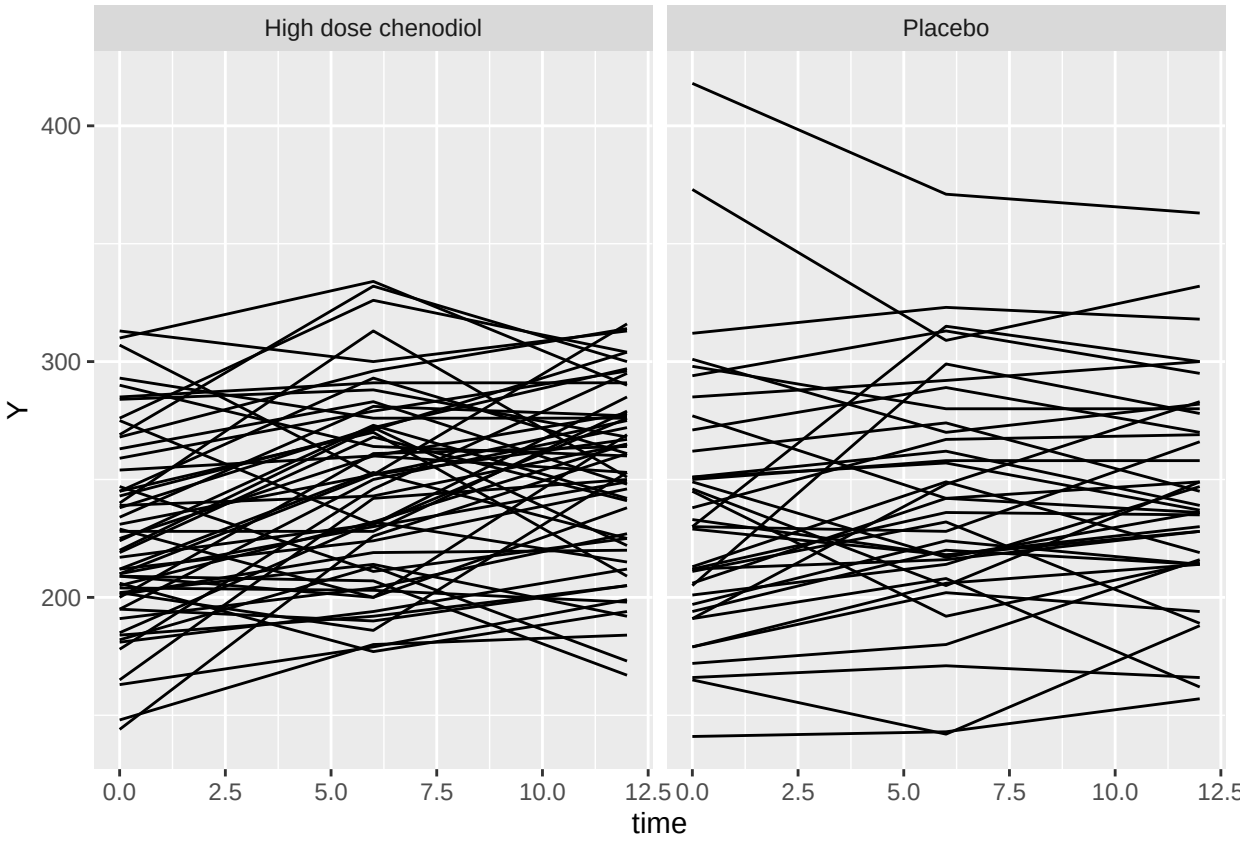
(b) Reshape the file in long format and make spaghetti plots of the individual trajectories in the two groups. (Either use two colours or make two plots)

Solution In order to generate a spaghetti plot, we need to write the data sets we have just created according to a long format.

```
#reshape the data set for the treatment group
treatlong<-reshape(treat,varying=c("t1","t2","t3"),v.names="Y",
                   timevar="time",time=c(0,6,12),direction="long")
#reshape the data set for the placebo group
placlong<-reshape(plac,varying=c("t1","t2","t3"),v.names="Y",
                  timevar="time",time=c(0,6,12),direction="long")
#generate two distinct spaghetti plot
#spaghetti plot for the treatment group
```

```
p<-ggplot(treatlong,aes(x=time,y=Y,group=id))+geom_line()
#spaghetti plot for the placebo group
p<-ggplot(placlong,aes(x = time, y = Y,group = id))+geom_line()

#generate a grid of 2 spaghetti plots
group_names<-c('1'="High dose chenodiol",'2'="Placebo")
p<-ggplot(data=rbind(placlong,treatlong),aes(x=time,y=Y,group=id))+
  facet_grid(.~grp,labeller=as_labeller(group_names))
p+geom_line()
```

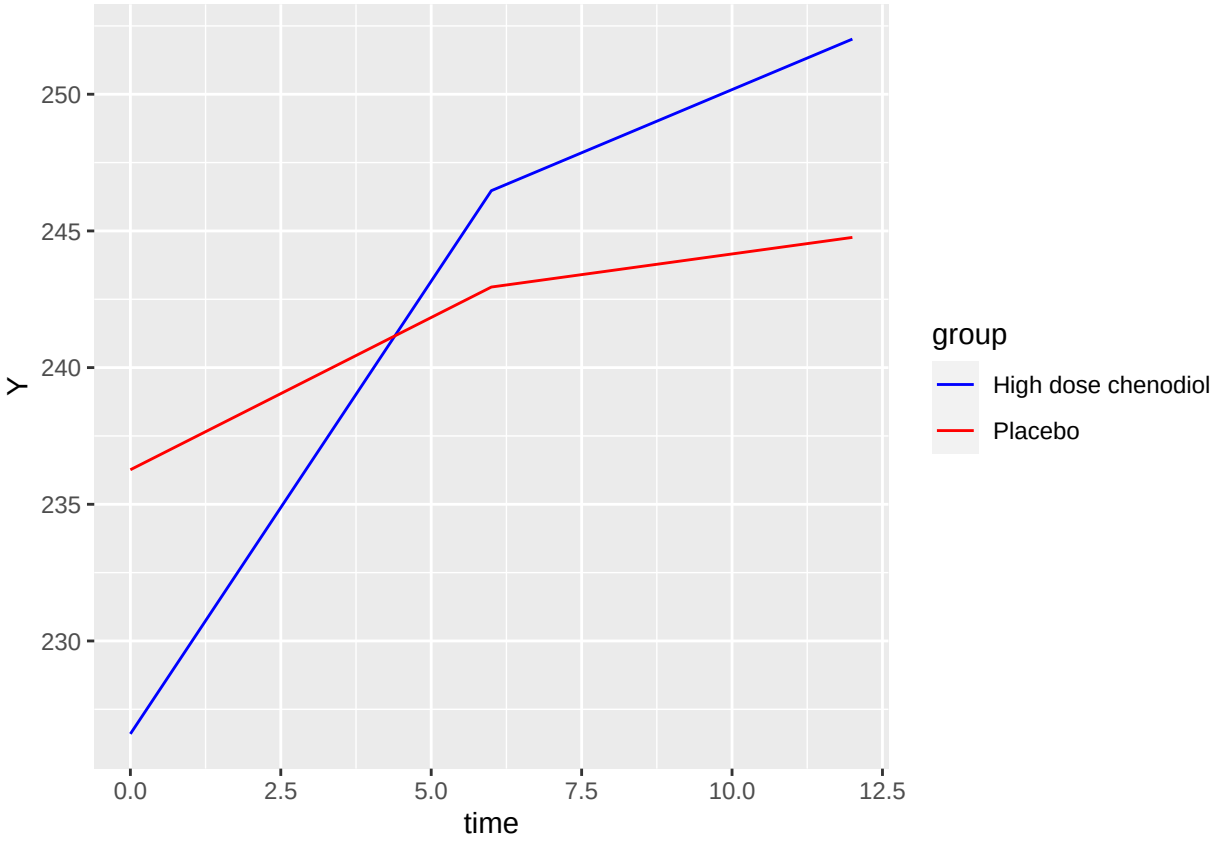


(c)Compute for each group and for each time point the mean and plot these for the two groups. Describe what you observe. Comment on the results.

Solution In order to compute the means for each time points, we need to calculate the averages for different time points (i.e. 0,6,12) using the data sets in the long format referring to the placebo group and the treatment group. This calculation can be performed using the *aggragate* function. The latter allows to calculate summary measures such as mean for a given variable of interest (in this example, serum cholesterol level) for different categories of interest (in this example, the 3 distinct time points). The results of the calculation are stored in a data frame object in which the first column denotes the measurement time and the second indicates the average. Since this calculation is performed for the treatment and placebo groups separately, a third column must be added to indicate the group membership. Then, the two data sets are merged into a unique data set.

```
#Creation of the data set with average responses for the placebo group
placmean<-aggregate(placlong$Y,list(placlong$time),FUN=mean)
#Placebo group=1
placmean<-cbind(placmean,rep(1,3))
colnames(placmean)[3]<-"grp"
```

```
#Creation of the data set with average responses for the treatment group
treatmean<-aggregate(treatlong$Y,list(treatlong$time),FUN=mean)
#Treatment group=0
treatmean<-cbind(treatmean,rep(0,3))
colnames(treatmean)[3]<-"grp"
#Join the rows of the two data sets
meandata<-rbind(treatmean,placmean)
#Rename the first two columns of the new data set
colnames(meandata)[1:2]<-c("time","Y")
#Create a ggplot object to plot the average treatments
#across the different measurement times in the two groups
#grp is treated as factor to ensure that we get a discrete legend
mean_plot<-ggplot(meandata,aes(x=time,y=Y,group=as.factor(grp),color=as.factor(grp)))+
  scale_color_manual(labels = c("High dose chenodiol", "Placebo"),
                     values = c("blue", "red"))+guides(color=guide_legend("group"))
#Show the actual plot
mean_plot+geom_line()
```



At baseline the average serum cholesterol level is much higher among patients that have been assigned to the placebo with respect to the subjects assigned to high dose chenodiol. However, after the treatment assignment at baseline, a rapid increase in the mean of serum cholesterol level is observed among patients that received high dose chenodiol. As matter of fact, after 6 months, the average response among patients assigned to the high dose chenodiol is 246.5 whereas it was 226.6 at baseline. On the other, the increase in the average level of serum cholesterol level among subjects, who received the placebo, is smaller (236.3 at baseline and 242.9 at 6 months). Finally, after 12 months, there is a further increase in the average responses, especially, in the group of patients that received the high dose chenodiol.

Question 2

We will perform a simulation study to compare the performance of the OLS estimator and the GLS estimator for correlated data.

Model 1: $Y = \beta_1 x + \beta_2 t + e$

Y is the response

X is a binary covariate

t is a continous regressor

e is the error term that is assumed to be normally distributed with zero mean and variance σ^2

Model 2: $Y = \beta_1 x + \beta_2 t + \beta_3 x \times t + e$

Model 2 is obtained from Model 1 by adding an interaction term between the two covariates.

(a)We start with the cross sectional setting. Choose values for β_1, β_2 and σ^2 and generate a dataset with observations of 100 individuals. Half of the subjects have $x = 0$ and half of the subjects have $x = 1$. The covariate t has values $0, 1, 2, 3, 4$, each value with the same proportion. Then, randomly generate the errors e and then compute Y . Estimate the model parameters by using OLS and check whether the estimates agree with the true values.

Solution We need to specify the true values of the model parameters and the sample size. Then, we randomly generate 100 values for the error term from a normal distribution and assign the values of the covariates to the patients. Subsequently, we can compute the responses and derive the estimates. We need to recall that the estimation of the model parameters require to specify a design matrix, in which we must specify a column of 1s, a column for the binary variable x, a column for the continuous covariate t and a column for the interaction between the two previously mentioned covariates.

```
#make your results reproducible
set.seed(1)
#Choose the true values for the model paramters
beta_1<-2
beta_2<-0.8
sigma<-1.5
#Fix the sample size
n<-100
#Randomly generate values for the error term from a normal distribution
ee<-rnorm(n,0,sigma)
#Assign the values of the covariate x to the patients
x<-c(rep(0,50),rep(1,50))
#Assign the values of the covariate t to the patients
t<-c(rep(c(0:4),20))
#Compute the design matrix where the a column of 1s must be added
xx<-as.matrix(cbind(rep(1,n),x,t,x*t))
#Compute the responses
y<-beta_1*x+beta_2*t+ee
#Compute the estimated regression coefficients
beta_est<-solve(t(xx)%*%xx)%*%t(xx)%*%y
beta_real<-c(beta_1,beta_2)
#Compute an estimate for the variance parameter sigma^2
sigma_est<-1/(n-2)*sum((y-xx%*%beta_est)^2)
#Compare the results
#print(cbind(beta_est[2:3,1],beta_real))
#print(c(sigma,sigma_est))
#it allows to produce tables in the same style as latex
```

```
#options(knitr.table.format = "latex")
#table for the comparison of the estimated vs.real betas
kable(cbind(beta_est[2:3,1],beta_real),col.names = c("est $\\beta$", "$\\beta$"),
      digits=2,escape = FALSE,booktabs = TRUE)
```

	est β	β
x	2.22	2.0
t	0.82	0.8

```
#table for the comparison of the estimated vs.real sigma^2
kable(t(c(sigma,sigma_est)),col.names = c("$\\sigma^2$","est $\\sigma^2$"),
      digits=2,escape = FALSE)
```

σ^2	est σ^2
1.5	1.83

```
#Alternatevely use the print() function to show your estimates
```

The estimates of the model coeffiecients of t and x are quite close to the true ones. On the other hand, the estimated value for σ^2 is a bit higher than the true one (1.83 vs. 1.50).

(b) Now generate multiple observations per subject by using the function rmvnorm of the package mtvnorm. Let's assume 5 observations for each subject. The correlation structure for the random variable e is as follows

$$\begin{bmatrix} 1 & 0.7 & 0.7 & 0.7 & 0.7^2 \\ 0.7 & 1 & 0.7 & 0.7 & 0.7 \\ 0.7 & 0.7 & 1 & 0.7 & 0.7 \\ 0.7 & 0.7 & 0.7 & 1 & 0.7 \\ 0.7^2 & 0.7 & 0.7 & 0.7 & 1 \end{bmatrix}$$

Now generate multivariate e and then create Y. For example first use the wide format to create a data set with an id variable, an x variable as above and five Y variables. Then you can use reshape to obtain the long format. Fit the model using the OLS and the GLS estimators. For GLS you can use the true correlation structure. (see lecture for code) Do you obtain the same estimates for the two estimators. What about the standard errors?

Solution First, we need to specify the correlation matrix in R. Then, we can simulate the data set. In this case, since we have multiple responses for each patient, we need to include an ID variable that uniquely identifies each subject. Furthermore, in this simulation, we must specify the error term to be distributed according to a multivariate normal with a variance-covariance matrix that is obtained by multiplying the parameter σ^2 by the specified correlation structure. After generating values for the error term, we can use Model 1 to estimate the responses. The generated data set is in wide format as the responses are stored in different columns, one for each measurement time. Therefore, in order to fit Model 2 and perform the estimation of the parameters, we need to write the data set in a long format, in which we have multiple rows for each subject. The parameters of Model 2 are estimated both using OLS and GLS.

```
#Specify the correlation matrix
cory<-matrix(0.7,nrow=5,ncol=5)
cory[5,1]<-0.7^2
cory[1,5]<-0.7^2
```

```
diag(cory)<-1
coryinv<-solve(cory)
#Variance-Covariance matrix of the error term
sigma<-1.5*cory
#Patient's id
id<-c(1:100)
#Assign the covariates to the patients
xbin<-c(rep(0,50),rep(1,50))
#true values of the model paramters
beta1<-2
beta2<-0.8
#Generate the values of the standard error (one column for each time points)
ee<-rmvnorm(n=100, mean = rep(0, 5), sigma = sigma)
#Generate the responses
y<-beta1*xbin+beta2*matrix(c(0,1,2,3,4), ncol=5, nrow=100, byrow=T)+ee
#Store the data in a data frame
#we have five different columns for the response (one for each time point)
data<-as.data.frame(cbind(id,xbin,y))
#rename the columns
colnames(data)<-c("id","xbin","t1","t2","t3","t4","t5")
#Rewrite the data using a long format
datalong <- reshape(data, idvar="id", varying=c("t1","t2","t3","t4","t5"),
                    v.names=c("Y"), timevar="time",time=c(0,1,2,3,4), direction="long")

#OLS estimation
#create the design matrix
xx<-cbind(rep(1,nrow(datalong)),datalong$xbin,datalong$time,datalong$xbin*datalong$time)
prod1<-crossprod(xx,xx)
prod2<-crossprod(xx,datalong$Y)
#take the inverse of prod1
prod1inv<-solve(prod1)
#generate the vector of the estimated coefficients
betaOLS<-prod1inv%*%prod2
print(betaOLS)
```

```
##            [,1]
## [1,] 0.11381632
## [2,] 1.71821206
## [3,] 0.76293579
## [4,] 0.07903165
```

```
#genearate the std. errors of the estimated coefficients
sOLS<- as.matrix(sqrt(diag(prod1inv)))


#GLS estimation
#initialize prod1 to 0
prod1<-0
#initialize prod2 to 0
prod2<-0
#update prod1 and prod2 for each patient
for (i in unique(datalong$id)){
x<-xx[(datalong$id==i),]
y<-datalong$Y[(datalong$id==i)]
```

```
prod1<-prod1+crossprod(x,coryinv%*%x)
prod2<-prod2+crossprod(x,coryinv%*%y)}
#take the inverse of prod1
prod1inv<-solve(prod1)
#calculate the estimated coefficients of the model
betaGLS<-prod1inv%*%prod2
print(betaGLS)
```

```
##           [,1]
## [1,] 0.1521835
## [2,] 1.7282131
## [3,] 0.7554827
## [4,] 0.0687144
```

```
#calculate the estimated value of sigma^2
ss<-var(datalong$Y-xx%*%betaGLS)
#calculate the std. errors of the estimated coefficients
sGLS<-as.matrix(sqrt(c(ss)*diag(prod1inv)))
#store the output in a numeric object
output<-numeric()
#create a data set to include the results
output<-rbind(output,cbind(t(betaOLS),t(betaGLS),t(sOLS),t(sGLS)))

#Bias for the estimated beta1 by OLS
mean((output[,2]-2))
```

```
## [1] -0.2817879
```

```
#Bias for the estimated beta1 by GLS
mean((output[,6]-2))
```

```
## [1] -0.2717869
```

```
#Bias for the estimated beta2 by OLS
mean((output[,3]-0.8))
```

```
## [1] -0.03706421
```

```
#Bias for the estimated beta2 by GLS
mean((output[,7]-0.8))
```

```
## [1] -0.04451731
```

```
#Standard error of the estimated beta1 by OLS
mean(output[,10])
```

```
## [1] 0.1549193
```

```
#Standard error of the estimated beta1 by GLS
mean(output[,14])
```

```
## [1] 0.2326179
```

```
#Standard error of the estimated beta2 by OLS
mean(output[,11])
```

```
## [1] 0.04472136
```

```
#Standard error of the estimated beta2 by GLS
mean(output[,15])
```

```
## [1] 0.03647073
```

```
#Show the estimates for beta1 and beta2
results<-cbind(c("OLS","GLS"),output[,c(2,6)],output[,c(3,7)])
colnames(results)<-c("method","t","x")
print(results)
```

```
##      method t                  x
## [1,] "OLS"  "1.71821206350614" "0.762935791318383"
## [2,] "GLS"  "1.72821305995781" "0.755482686630324"
```

```
#Show the estimates for the std. errors of beta1 and beta2
results<-cbind(c("OLS","GLS"),output[,c(10,14)],output[,c(11,15)])
colnames(results)<-c("method","t","x")
print(results)
```

```
##      method t                   x
## [1,] "OLS"  "0.154919333848297" "0.0447213595499958"
## [2,] "GLS"  "0.232617945339513" "0.0364707348947889"
```

The estimated regression coefficients related to the covariates x and t seem to be quite similar in both estimation procedures. The GLS provides a larger estimated standard errors of the effect of covariate x with respect to OLS. On the other hand, the two methods provide standard errors for the effect of the covariate t that are almost identical.

(c)For your generated dataset from (b) make spaghetti plots of a subset of individuals. Generate now a dataset with less correlation for example 0.2 instead of 0.7. Make again spaghetti plots. Do you understand the difference?

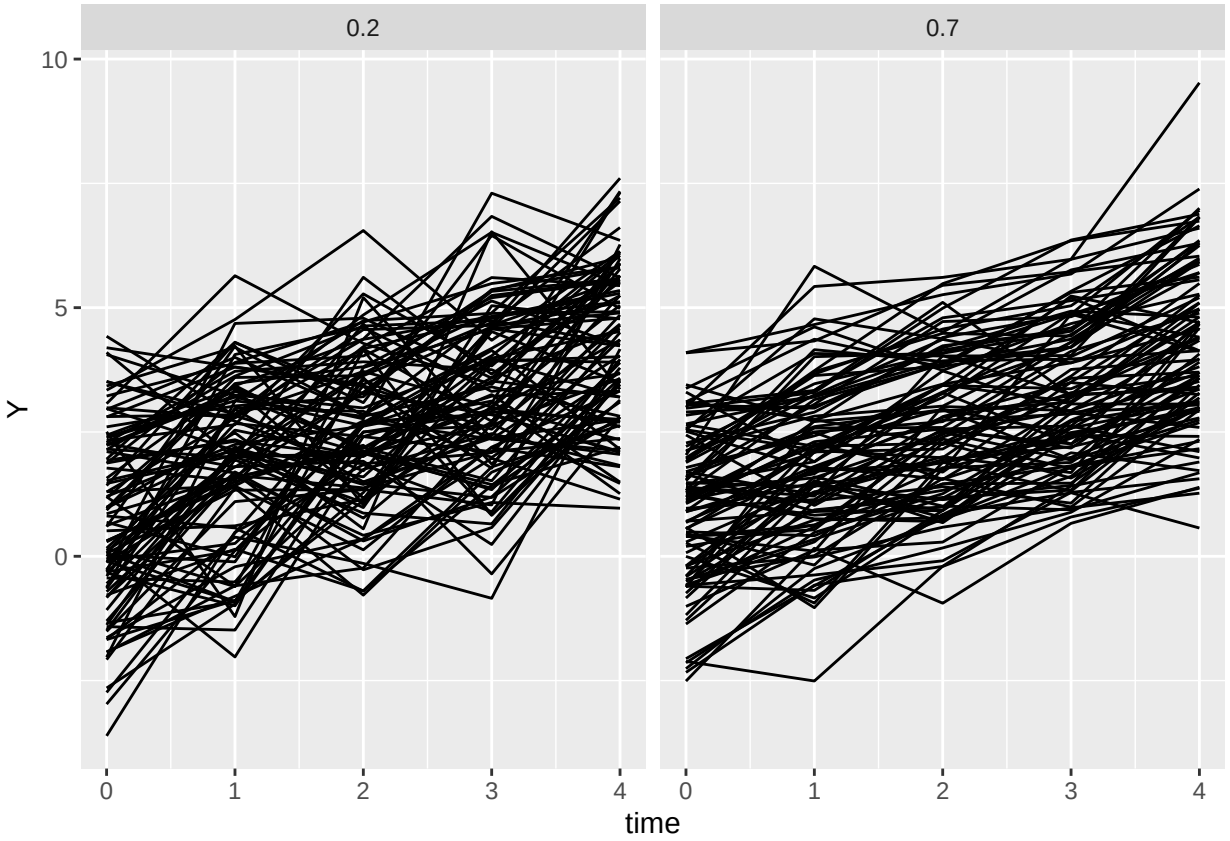
Solution We can use the same code as in the previous question and simply change the value of the correlation parameter.

```
#Specify the correlation matrix
cory_1<-matrix(0.2,nrow=5,ncol=5)
cory_1[5,1]<-0.2^2
cory_1[1,5]<-0.2^2
diag(cory_1)<-1
coryinv_1<-solve(cory_1)
#Variance-Covariance matrix of the error term
sigma<-1.5*cory_1
#Patient's id
id_1<-c(1:100)
#Assign the covariates to the patients
xbin_1<-c(rep(0,50),rep(1,50))
#true values of the model paramters
beta1_1<-2
beta2_1<-0.8
#Generate the values of the standard error (one column for each time points)
ee_1<-rmvnorm(n=100, mean = rep(0, 5), sigma = sigma)
y_1<-beta1_1*xbin_1+beta2_1*matrix(c(0,1,2,3,4), ncol=5, nrow=100, byrow=T)+ee_1
#Store the data in a data frame
#we have five different columns for the response (one for each time point)
data_1<-as.data.frame(cbind(id_1,xbin_1,y_1))
#rename the columns
colnames(data_1)<-c("id","xbin","t1","t2","t3","t4","t5")
#Rewrite the data using a long format
datalong_1 <- reshape(data_1, idvar="id", varying=c("t1","t2","t3","t4","t5"),
```

```
                    v.names=c("Y"), timevar="time",time=c(0,1,2,3,4), direction="long")

#create the spaghetti plot
#simulated data set for 0.7
datalong<-cbind(datalong,c(rep(0.7,500)))
colnames(datalong)[ncol(datalong)]<-"correlation"
#simulated data set for 0.2
datalong_1<-cbind(datalong_1,c(rep(0.2,500)))
colnames(datalong_1)[ncol(datalong_1)]<-"correlation"

p_1<-ggplot(rbind(datalong,datalong_1),aes(x=time,y=Y,group=id))+
  facet_grid(.~correlation)
#show the spaghetti plot
p_1+geom_line()
```



A higher variability in the patients' trajectories is observed if we choose a lower correlation. This is because a lower correlation implies that there is a lower association among the observed values of the responses at the different time points. Hence, a higher diversity in the patient's responses is observed at the different measurement times.

(d)Finally carry out the simulation study. Generate 100 data sets with 100 subjects with 5 observations each. Check whether the OLS and GLS estimators for β_1 are biased. Compute the empirical standard errors for the estimators of β_1 and compare these with the theoretical ones. What is your conclusion?

Solution In order to generate the 100 data sets we run a loop, in which a data set is simulated at each iteration based on the normal model with multiple responses. At the end of each iteration, we estimate the model coefficients according to both OLS and GLS. Then, in order to get the final estimates of such

parameters, we average the results across the different iterations. After the computation of these estimates, we can compare them with their true values. If the estimated values are quite close to the real ones, then the estimated parameters are unbiased.

```
#specify the correlation structure
cory<-matrix(0.7,nrow=5,ncol=5)
cory[5,1]<-0.7^2
cory[1,5]<-0.7^2
diag(cory)<-1
coryinv<-solve(cory)
#Create an object to store the results of the simulation
output<-numeric()
for (k in 1:100){
#Randomly generate 100 values from a multivariate normal distribution
ee<-rmvnorm(n=100, mean = rep(0, 5), sigma = sigma)
#Calculate the responses at the different measurement times
y<-beta1*xbin+beta2*matrix(c(0,1,2,3,4), ncol=5, nrow=100, byrow=T)+ee
#Store the results in a data frame
data<-as.data.frame(cbind(id,xbin,y))
#rename the columns
colnames(data)<-c("id","xbin","t1","t2","t3","t4","t5")
#Rewrite the data using a long format
datalong <- reshape(data, idvar="id",
                    varying=c("t1","t2","t3","t4","t5"), v.names=c("Y"), timevar="time",
                    time=c(0,1,2,3,4), direction="long")

#OLS estimation
xx<-cbind(rep(1,nrow(datalong)),datalong$xbin,datalong$time,datalong$xbin*datalong$time)
prod1<-crossprod(xx,xx)
prod2<-crossprod(xx,datalong$Y)
prod1inv<-solve(prod1)
betaOLS<-prod1inv%*%prod2
betaOLS
sOLS<- as.matrix(sqrt(diag(prod1inv)))

#GLS
#initialize prod1 to 0
prod1<-0
#initialize prod2 to 0
prod2<-0
for (i in unique(datalong$id)){
x<-xx[(datalong$id==i),]
y<-datalong$Y[(datalong$id==i)]
prod1<-prod1+crossprod(x,coryinv%*%x)
prod2<-prod2+crossprod(x,coryinv%*%y)}
prod1inv<-solve(prod1)
betaGLS<-prod1inv%*%prod2
ss<-var(datalong$Y-xx%*%betaGLS)
betaGLS
sGLS<-as.matrix(sqrt(c(ss)*diag(prod1inv)))
output<-rbind(output,cbind(t(betaOLS),t(betaGLS),t(sOLS),t(sGLS)))}
#Bias of the OLS estimator for the treatment effect
mean((output[,2]-2))
```

```
## [1] -0.01124591
```

```r
#Bias of the GLS estimator for the treatment effect
mean((output[,6]-2))
```

```
## [1] -0.01281608
```

```r
#Bias of the OLS estimator for the effect of time
mean((output[,3]-0.8))
```

```
## [1] -0.0009914558
```

```r
#Bias of the OLS estimator for the effect of time
mean((output[,7]-0.8))
```

```
## [1] -0.0003979617
```

```r
#estimate for the standard error of the treatment effect when using OLS
sqrt(var(output[,2]))
```

```
## [1] 0.2018718
```

```r
#estimate for the standard error of the treatment effect when using GLS
sqrt(var(output[,6]))
```

```
## [1] 0.2122736
```

```r
#estimates for the sample conditional variance when using OLS
mean(output[,10])
```

```
## [1] 0.1549193
```

```r
#estimates for the sample conditional variance when using GLS
mean(output[,14])
```

```
## [1] 0.2336091
```

The estimates for β_1 are unbiased both when we use OLS and GLS. As matter of fact, after performing 100 simulations and averaging the result of each simulation, we obtain an average estimate for the treatment effect that is very close to the real one. Furthermore, if we increase the number of simulations, we will obtain estimates that are closer and closer to the real ones.