Solution NCGS exercise (R software)

Contents

Question 1: Data import **2**

Question 2: Scatterplot and correlation plot **2**

Question 3: Spaguetti plots **5**

Question 4: Display the mean values **6**

Question 5: ANOVA with unstructured correlation matrix **7**

Question 6: ANOVA with baseline adjustement **11**

Question 7: ANCOVA **13**

Appendix A: Alternative syntax for question 2 **17**

Appendix B: Fitting mixed models in R **20**

Packages 20

Arguments for functions from the nlme package 21

NOTE: This document proposes an R syntax giving the necessary outputs to answer to the questions of the exercise. The focus here is then on the implementation in R software and not on the interpretation or the validity of the results: we refer to the SAS solution for a discussion of the two latter points.

Because of the multiplicity of the packages in R, there are often several ways to perform a given operation (e.g. fitting a mixed model or converting a dataset from the long to the wide format). Some can be more efficient (in term of computation time or memory usage), other can be closer to the natural language or enabling a concise syntax. We don't claim to propose here the "best" R syntax but we tried to provide an readable syntax that could be re-used in other problems. In some cases an alternative syntax, usually more complex but more efficient/generalisable, is proposed in appendix.

First, load the necessary packages

```
> library(nlme)     # gls function
> library(lattice) # xyplot
> library(psych)     # pairs.panels
>
> # optional
> library(data.table)    # data.table
```

Question 1: Data import

You can read a ".txt" (or here a ".sas") file using the `read.table` function:

```
> df.data_ncgs <- read.table("http://publicifsv.sund.ku.dk/~jufo/courses/repeated15/ncgs.sas",
+                            header = FALSE,
+                            skip = 8, nrow = 111 - 8, na.strings = ".")
> names(df.data_ncgs) <- c("treat","id","y0","y1","y2","y3","y4")
```

As for many R functions there are many possible arguments, hopefully most of them have valid default values. Here it is important to specify `na.strings = "."` so that R recognise correctly the missing values (otherwise it will treat them as characters). Because the file is not in the correct format (it contains SAS commands at the begining) we have to skip the first lines and to rename the columns.

```
> str(df.data_ncgs)
```

```
'data.frame':   103 obs. of  7 variables:
 $ treat: int  1 1 1 1 1 1 1 1 1 1 ...
 $ id   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ y0   : int  178 254 185 219 205 182 310 191 245 229 ...
 $ y1   : int  246 260 232 268 232 213 334 204 270 200 ...
 $ y2   : int  295 278 215 241 265 173 290 227 209 238 ...
 $ y3   : int  228 245 220 260 242 200 286 228 255 259 ...
 $ y4   : int  274 340 292 320 230 193 248 196 213 221 ...
```

The dataset contains 103 observations and 7 variables

To have meaningful names for the treatment variable, we will first convert it from the numeric format (e.g. 1 or 2) to the factor format:

```
> df.data_ncgs$treat <- factor(df.data_ncgs$treat, levels = 1:2, labels = c("T", "C"))
```

The id variable will be convert to factor as it should not be treated as a continuous variable:

```
> df.data_ncgs$id <- factor(df.data_ncgs$id)
```

Question 2: Scatterplot and correlation plot

In this question we will look at the relation between the endpoint measured at different time. As a shortcut for the names indicating the relevant columns we define:

```
> y_columns <- c("y0","y1","y2","y3","y4")
```

or equivalently

```
> y_columns <- paste("y",0:4,sep = "")
```

Then we define two vectors, each containing the index of the observations relative to a treatment group:

```
> index_treatment <- which(df.data_ncgs$treat == "T")
> index_control <- which(df.data_ncgs$treat == "C")
```

We can then compute the variance/covariance matrix, the correlation matrix, and the mean response over time for each group:

```
> ### Treatment group
> Sigma.T <- var(df.data_ncgs[index_treatment,y_columns], na.rm = TRUE)
> Sigma.T
```

```
         y0       y1        y2        y3        y4
y0 1962.463 1302.197 1150.8540  952.3460 1009.2794
y1 1302.197 1715.216 1109.2159 1023.4270 1199.3746
y2 1150.854 1109.216 1553.9016  696.8556 1265.5746
y3  952.346 1023.427  696.8556 1147.6087  866.6111
y4 1009.279 1199.375 1265.5746  866.6111 2545.6921
```

```
> cor(df.data_ncgs[index_treatment,y_columns], use = "pairwise.complete.obs")
```

```
          y0        y1        y2        y3        y4
y0 1.0000000 0.7203380 0.6226680 0.5907770 0.4581925
y1 0.7203380 1.0000000 0.6695283 0.7153099 0.5832976
y2 0.6226680 0.6695283 1.0000000 0.5374287 0.6363163
y3 0.5907770 0.7153099 0.5374287 1.0000000 0.5140974
y4 0.4581925 0.5832976 0.6363163 0.5140974 1.0000000
```

```
> mean.T <- apply(df.data_ncgs[index_treatment,y_columns], # for a given dataset
+                 MARGIN = 2, # for each column (1 would be row, 2 indicates columns)
+                 FUN = mean, na.rm = TRUE) # apply the function mean with argument na.rm = T
> mean.T
```

```
      y0       y1       y2       y3       y4
226.0161 245.5323 252.0182 256.7955 254.5526
```

```
> # the previous formula is equivalent to
> mean.T <- c("y0" = mean(df.data_ncgs[index_treatment,"y0"], na.rm = TRUE),
+             "y1" = mean(df.data_ncgs[index_treatment,"y1"], na.rm = TRUE),
+             "y2" = mean(df.data_ncgs[index_treatment,"y2"], na.rm = TRUE),
+             "y3" = mean(df.data_ncgs[index_treatment,"y3"], na.rm = TRUE),
+             "y4" = mean(df.data_ncgs[index_treatment,"y4"], na.rm = TRUE)
+ )
>
> ### Control group
> Sigma.C <- var(df.data_ncgs[index_control,y_columns], na.rm = TRUE)
> Sigma.C
```

```
          y0       y1       y2       y3       y4
y0 3080.437 2342.718 2158.734 2404.831 2086.777
y1 2342.718 2755.492 2261.284 2392.099 2123.539
y2 2158.734 2261.284 2267.723 2184.922 1828.959
y3 2404.831 2392.099 2184.922 2666.957 2012.982
y4 2086.777 2123.539 1828.959 2012.982 2439.191
```

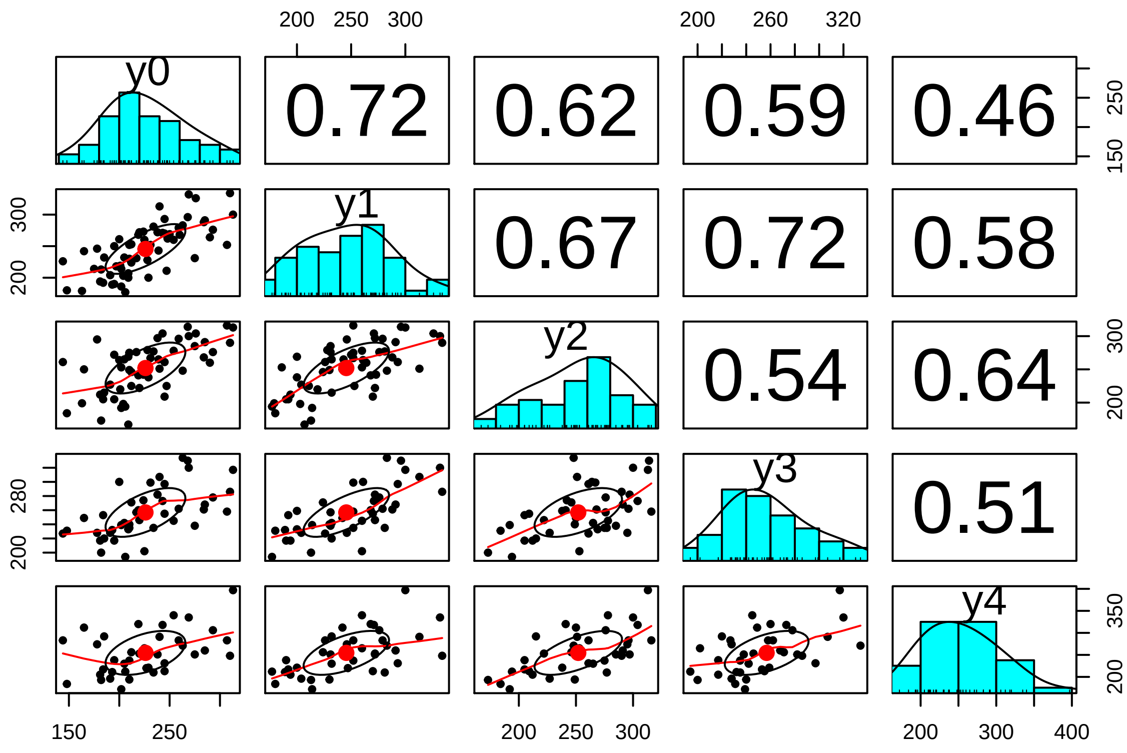
```
> cor(df.data_ncgs[index_control,y_columns], use = "pairwise.complete.obs")
```

```
          y0        y1        y2        y3        y4
y0 1.0000000 0.8161257 0.8323153 0.8442542 0.7612846
y1 0.8161257 1.0000000 0.8874039 0.8688476 0.8191013
y2 0.8323153 0.8874039 1.0000000 0.8779475 0.7776534
y3 0.8442542 0.8688476 0.8779475 1.0000000 0.7892396
y4 0.7612846 0.8191013 0.7776534 0.7892396 1.0000000
```

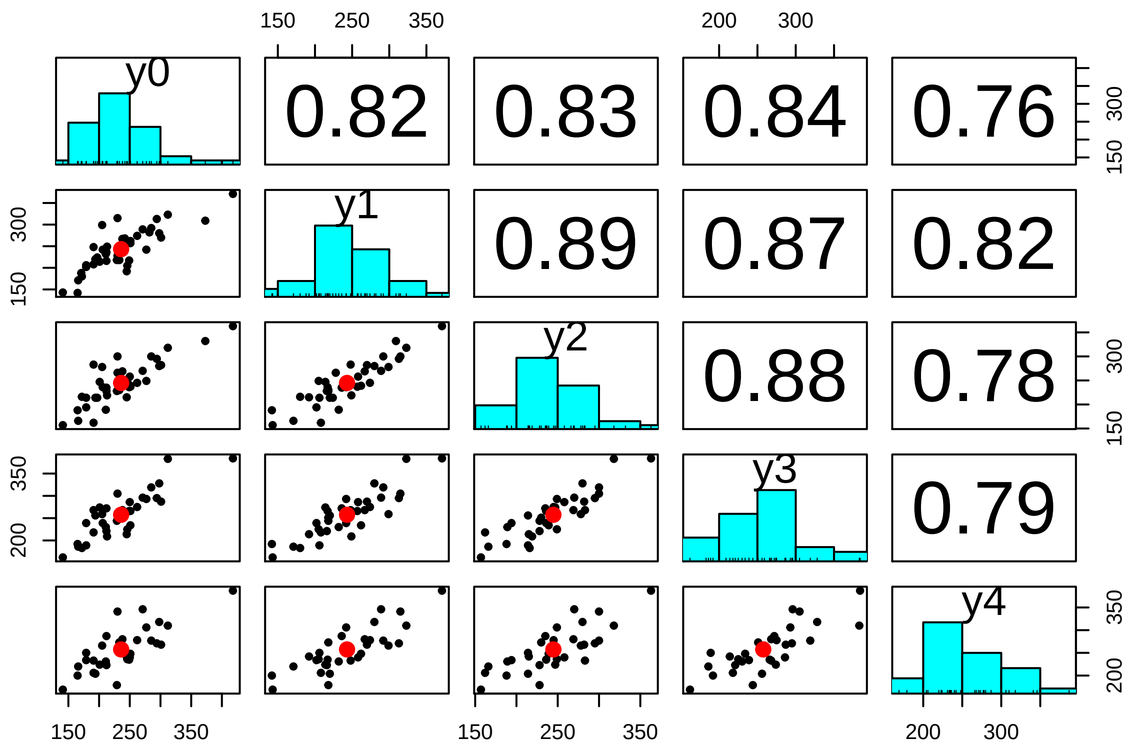
```
> mean.C <- apply(df.data_ncgs[index_control,y_columns], 2, mean, na.rm = TRUE)
```

and display some exploratory graphs:

```
> pairs.panels(df.data_ncgs[index_treatment,y_columns])
```



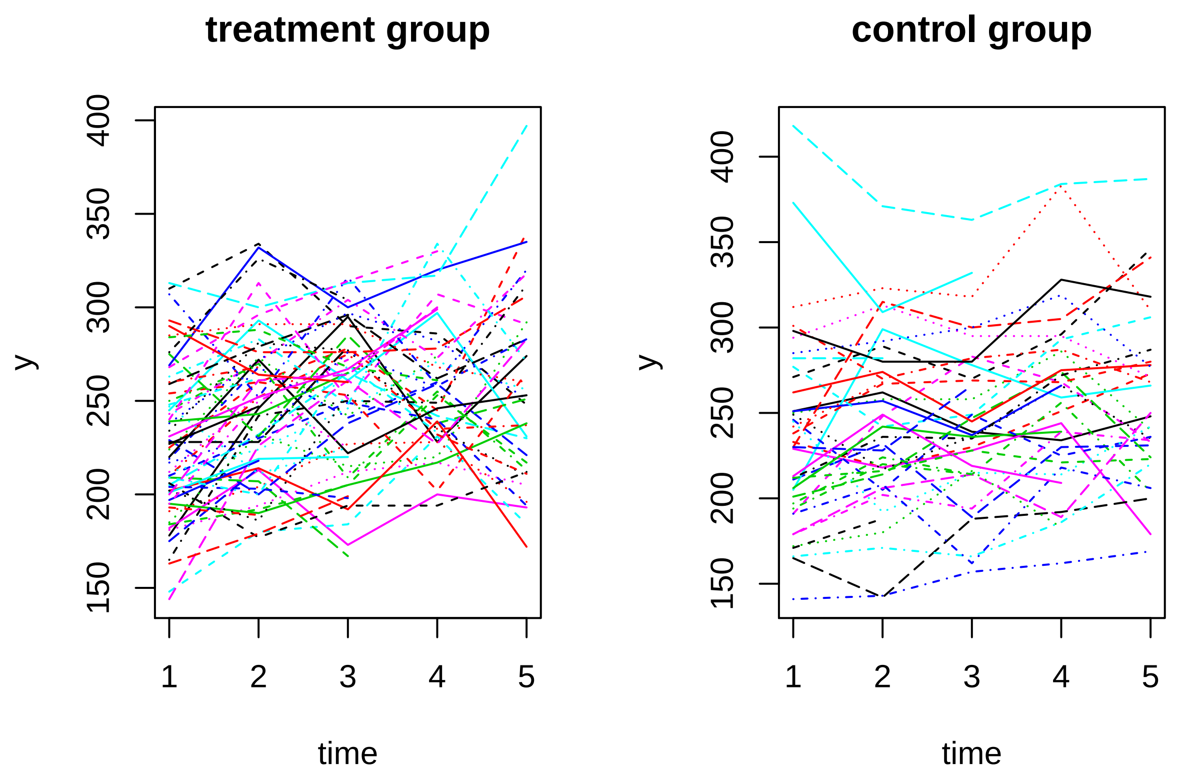
```
> pairs.panels(df.data_ncgs[index_control,y_columns],
+                 smooth = FALSE, ellipses = FALSE, density = FALSE)
```



```
> # See appendix A for alternative syntaxes to answer this question
```

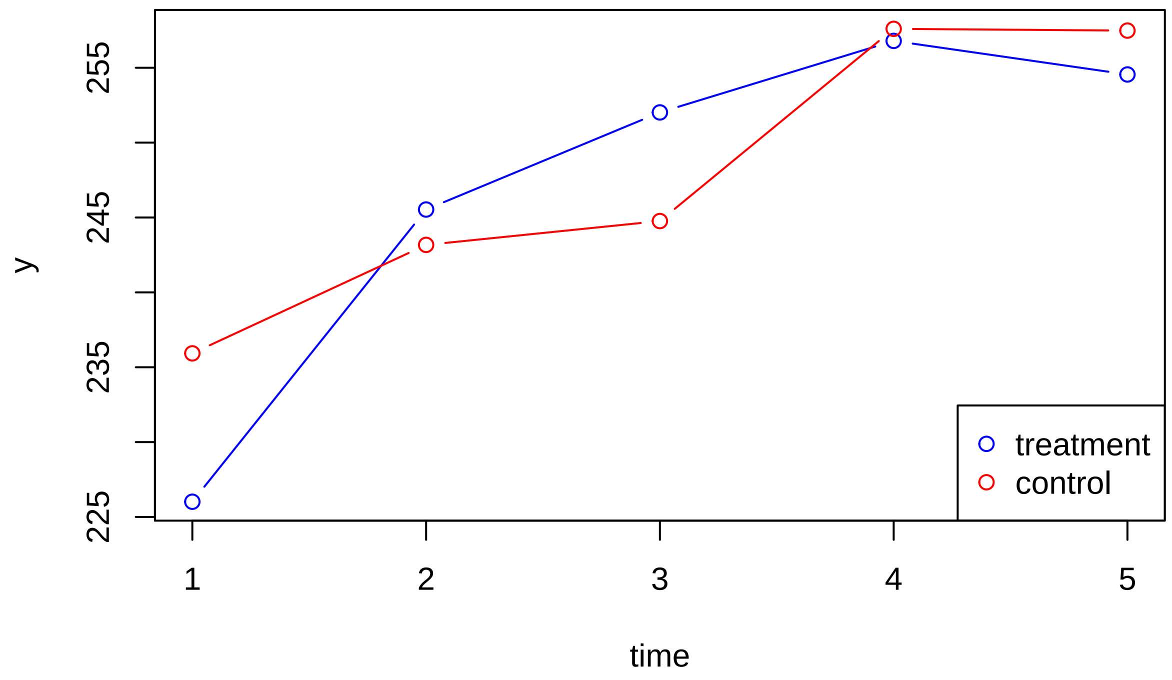
Question 3: Spaguetti plots

```
> par(mfrow = c(1,2))
>
> matplot(t(df.data_ncgs[index_treatment,y_columns]),
+         type = "l", ylab = "y", xlab = "time", main = "treatment group")
>
> matplot(t(df.data_ncgs[index_control,y_columns]),
+         type = "l", ylab = "y", xlab = "time", main = "control group")
```



Question 4: Display the mean values

```
> plot(mean.T, type = "b", xlab = "time", ylab = "y", col = "blue",
+      ylim = range(c(mean.C, mean.T)))
> points(mean.C, type = "b", col = "red")
> legend("bottomright", legend = c("treatment","control"), col = c("blue","red"), pch = 21)
```



Question 5: ANOVA with unstructured correlation matrix

First we need to convert the data from the wide format to the long format

```
> # duplicate the columns y0 in order keep the baseline measurement in the long format
> df.data_ncgs_baseline <- data.frame(df.data_ncgs,
+                                     baseline = df.data_ncgs$y0)
>
> dfLong.data_ncgs <- reshape(df.data_ncgs_baseline,
+             varying = y_columns, # time variables in the wide format ("y0", "y1", ...)
+             idvar = "id", # variable that indicates the clusters
+             direction  = "long", # from wide to long format
+             sep = "" # separator between the variable name ("y") and time ("0",...)
+ )
> dfLong.data_ncgs$time.factor <- as.factor(dfLong.data_ncgs$time)
>
> head(dfLong.data_ncgs)
```

```
    treat id baseline time   y time.factor
1.0     T  1      178    0 178           0
2.0     T  2      254    0 254           0
3.0     T  3      185    0 185           0
4.0     T  4      219    0 219           0
5.0     T  5      205    0 205           0
```

```
6.0     T  6      182    0 182           0
```

Then, we set the reference level for each categorical explanatory variable:

```
> dfLong.data_ncgs$treat <- relevel(dfLong.data_ncgs$treat, ref = "C")
> dfLong.data_ncgs$time.factor <- relevel(dfLong.data_ncgs$time.factor, ref = "0")
```

Then we use the **gls** function from the library nlme:

```
> # See appendix B for more details about available R packages
> # for modelling repeated measurements
> gls.UN <- gls(y ~  time.factor + treat + treat:time.factor,
+               data = dfLong.data_ncgs, na.action = "na.exclude",
+               correlation = corSymm(form = ~1 | id),
+               weights = varIdent(form = ~1 | time.factor)
+ )
> # same as gls(y ~ treat * time.factor, ...)
>
> ## summary of the fitted model
> summary(gls.UN)
```

```
Generalized least squares fit by REML
  Model: y ~ time.factor + treat + treat:time.factor
  Data: dfLong.data_ncgs
       AIC      BIC    logLik
  4314.588 4416.587 -2132.294

Correlation Structure: General
 Formula: ~1 | id
 Parameter estimate(s):
 Correlation:
  1     2     3     4
2 0.770
3 0.732 0.773
4 0.738 0.800 0.726
5 0.586 0.665 0.678 0.625
Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | time.factor
 Parameter estimates:
        0         1         2         3         4
1.0000000 0.9320567 0.8791668 0.8974016 1.0300809

Coefficients:
                        Value Std.Error  t-value p-value
(Intercept)         235.92683  7.305948 32.29243  0.0000
time.factor1          7.24390  4.805426  1.50744  0.1324
time.factor2          8.84620  5.207262  1.69882  0.0901
time.factor3         23.10333  5.292171  4.36557  0.0000
time.factor4         21.12230  7.398137  2.85508  0.0045
treatT               -9.67829  9.412956 -1.02819  0.3044
time.factor1:treatT  12.21751  6.193408  1.97266  0.0492
time.factor2:treatT  16.28893  6.738391  2.41733  0.0160
time.factor3:treatT   4.75670  6.973254  0.68213  0.4955
time.factor4:treatT   6.53598  9.763271  0.66945  0.5036

 Correlation: 
                    (Intr) tm.fc1 tm.fc2 tm.fc3 tm.fc4 treatT tm.1:T tm.2:T
time.factor1        -0.429                                                 
time.factor2        -0.500  0.581                                          
time.factor3        -0.466  0.606  0.526                                   
time.factor4        -0.392  0.476  0.522  0.438                            
treatT              -0.776  0.333  0.388  0.362  0.304                     
time.factor1:treatT  0.333 -0.776 -0.451 -0.470 -0.369 -0.429              
time.factor2:treatT  0.387 -0.449 -0.773 -0.407 -0.404 -0.497  0.578       
time.factor3:treatT  0.354 -0.460 -0.400 -0.759 -0.332 -0.456  0.592  0.513
time.factor4:treatT  0.297 -0.361 -0.396 -0.332 -0.758 -0.378  0.463  0.503
                    tm.3:T
time.factor1              
time.factor2              
time.factor3              
time.factor4              
treatT                    
time.factor1:treatT       
time.factor2:treatT       
time.factor3:treatT       
time.factor4:treatT  0.419

Standardized residuals:
        Min          Q1         Med          Q3         Max 
-2.32029931 -0.68866951 -0.02685014  0.60855781  3.89204119 

Residual standard error: 46.7809 
Degrees of freedom: 447 total; 437 residual
```

```
> ## extract the covariance matrix
> getVarCov(gls.UN)
```

```
Marginal variance covariance matrix
       [,1]   [,2]   [,3]   [,4]   [,5]
[1,] 2188.5 1513.0 1407.9 1449.2 1320.7
[2,] 1513.0 1762.4 1335.5 1409.3 1345.6
[3,] 1407.9 1335.5 1691.5 1254.4 1343.3
[4,] 1449.2 1409.3 1254.4 1762.4 1264.2
[5,] 1320.7 1345.6 1343.3 1264.2 2322.1
  Standard Deviations: 46.781 41.981 41.128 41.981 48.188 
```

Small differences compared to SAS output, e.g.:

```
                            R        SAS
-2 log-likelihood -4264.5882522 -4265.0800
sigma2_11          2188.4521465  2186.6100
sigma2_12          1512.9966849  1570.0500
betaT                -9.6782907    -9.9107
sd.betaT              9.4129562     9.4128
p_value.betaT         0.3044301     0.2949
```

Test the significance of the effects

```
> anova(gls.UN, type = "marginal")
```

```
Denom. DF: 437
                  numDF   F-value p-value
(Intercept)           1 1042.8011  <.0001
time.factor           4    5.9778  0.0001
treat                 1    1.0572  0.3044
time.factor:treat     4    1.9792  0.0967
```

```
> # see correction of exercise 2, appendix B, for explainations about the choice of the type argument
```

Here despite we reach the same conclusion that we obtained with SAS, the p.values show non neclectable differences.

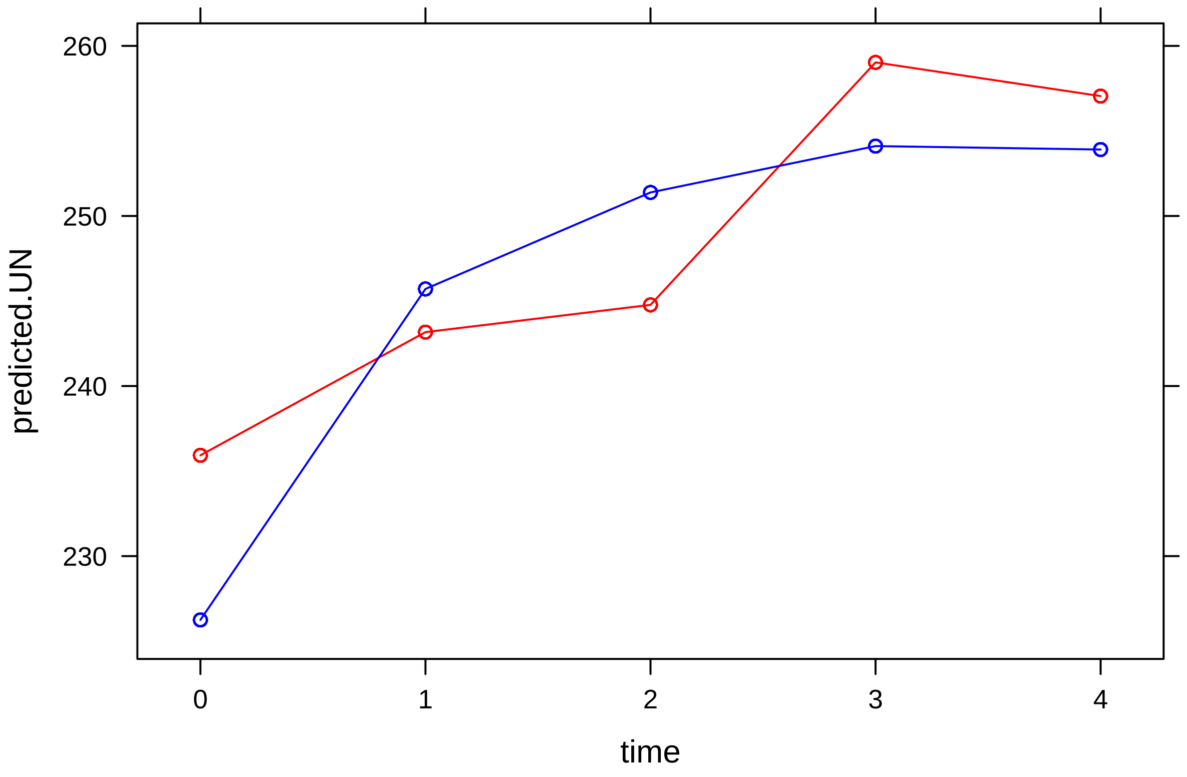
Get the confidence intervals

```
> intervals(gls.UN, which = "coef")[[1]][c("treatT","time.factor4","time.factor4:treatT"),]
```

```
                         lower       est.     upper
treatT              -28.178584 -9.678291  8.822002
time.factor4          6.581951 21.122304 35.662657
time.factor4:treatT -12.652821  6.535984 25.724788
```

Display the fitted mean value for each group:

```
> dfLong.data_ncgs_Pred <- data.frame(dfLong.data_ncgs,predicted.UN = predict(gls.UN))
>
> xyplot(predicted.UN ~ time, group = treat,
+        data = na.omit(dfLong.data_ncgs_Pred[order(dfLong.data_ncgs_Pred$time),]),
+        type = "b", col = c("red", "blue"))
```



```
> # see correction of exercice 1, appendix C, for a more general way to obtain the fitted values
```

Question 6: ANOVA with baseline adjustement

First re-define the treatment variable, considering all patient as placebo at baseline (i.e. constraining common baseline value between groups):

```
> dfLong.data_ncgs$treat_adj <- ifelse( (dfLong.data_ncgs$treat == "T") * (dfLong.data_ncgs$time != 0),
+                                        yes = "T", no = "C" )
> # for each observation if treat equals T and time is not 0 then set treat_adj to T otherwise to C
>
> dfLong.data_ncgs$treat_adj <- relevel(factor(dfLong.data_ncgs$treat_ad), ref = "C")
```

Define the possible interactions for the treated patients:

```
> dfLong.data_ncgs$time_X_treat <- factor((dfLong.data_ncgs$treat == "T")*(dfLong.data_ncgs$time))
```

Estimate the model:

```
> dfLong.data_ncgs$time.factor <- relevel(dfLong.data_ncgs$time.factor, ref = "0")
> dfLong.data_ncgs$time_X_treat <- relevel(dfLong.data_ncgs$time_X_treat, ref = "0")
>
> gls.UN_BaseConstrain <- gls(y ~ time.factor + time_X_treat,
+                              data = dfLong.data_ncgs, na.action = "na.exclude",
+                              correlation = corSymm(form = ~1 | id),
+                              weights = varIdent(form = ~1 | time.factor)
+ )
```

Small differences compared to SAS output, e.g.:

```
                                  R       SAS
-2 log-likelihood   -4271.9672568 -4272.5000
sigma2_11            2190.5335591  2188.9400
sigma2_12            1514.3970398  1571.7200
betaT_time4             2.7335330     3.0033
sd.betaT_time4          9.0417703     9.2511
p_value.betaT_time4     0.7625495     0.7463
```

```
> anova(gls.UN_BaseConstrain, type = "marginal")
```

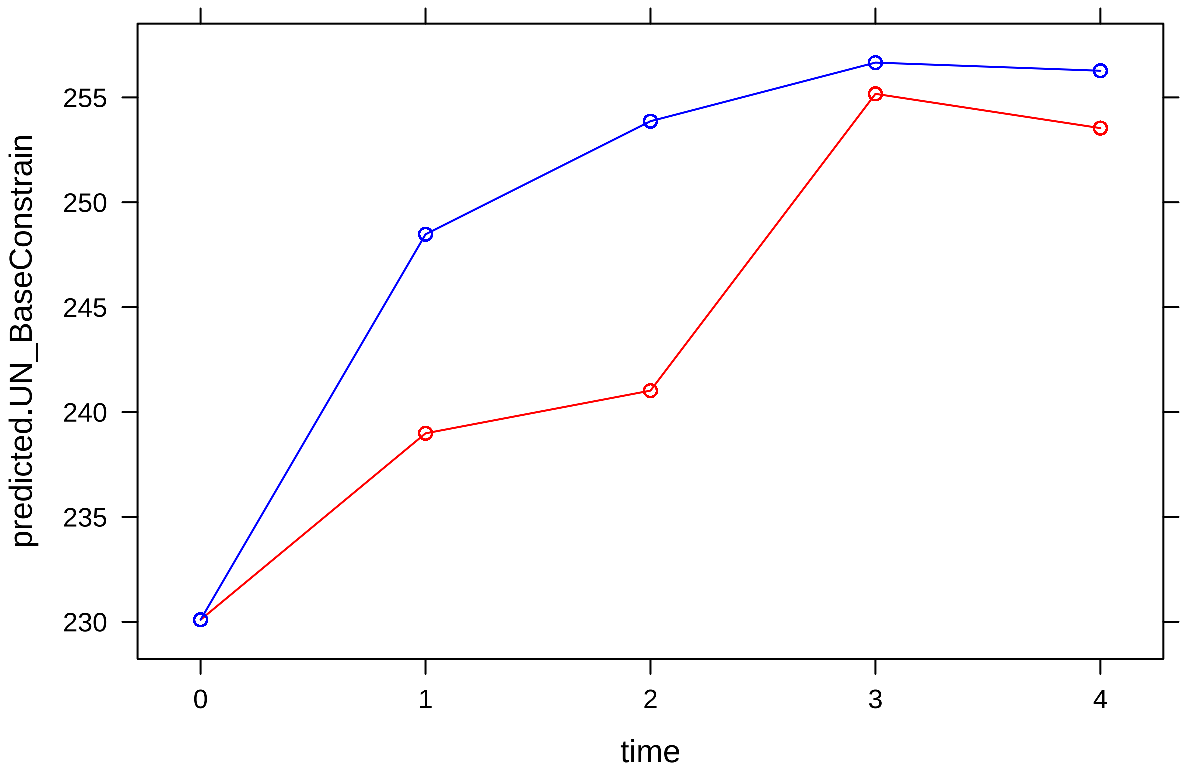
```
Denom. DF: 438
             numDF   F-value p-value
(Intercept)      1 2492.5335  <.0001
time.factor      4    7.6350  <.0001
time_X_treat     4    1.7165  0.1452
```

```
> intervals(gls.UN_BaseConstrain, which = "coef")[[1]][c("time.factor4","time_X_treat4"),]
```

```
                   lower      est.    upper
time.factor4    9.582319 23.439584 37.29685
time_X_treat4 -15.037116  2.733533 20.50418
```

Display the fitted mean value for each group:

```
> dfLong.data_ncgs_Pred <- data.frame(dfLong.data_ncgs_Pred,
+                                     predicted.UN_BaseConstrain = predict(gls.UN_BaseConstrain))
>
>
> xyplot(predicted.UN_BaseConstrain ~ time, group = treat,
+        data = na.omit(dfLong.data_ncgs_Pred[order(dfLong.data_ncgs_Pred$time),]),
+        type = "b", col = c("red", "blue"))
```



Question 7: ANCOVA

```
> # define a new dataset excluding baseline
> dfLong.data_ncgs_ANCOVA <- dfLong.data_ncgs[dfLong.data_ncgs$time > 0,]
> dfLong.data_ncgs_ANCOVA$time.factor <- factor(dfLong.data_ncgs_ANCOVA$time.factor)
>
> # define the reference for each qualitative variable
> dfLong.data_ncgs_ANCOVA$treat <- relevel(dfLong.data_ncgs_ANCOVA$treat, ref = "C")
> dfLong.data_ncgs_ANCOVA$time.factor <- relevel(dfLong.data_ncgs_ANCOVA$time.factor, ref = "4")
>
> gls.UN_ANCOVA <- gls(y ~ time.factor + treat + baseline:time.factor + time.factor:treat,
+                     data = dfLong.data_ncgs_ANCOVA, na.action = "na.exclude",
+                            correlation = corSymm(form = ~1 | id),
+                            weights = varIdent(form = ~1 | time.factor)
+ )
>
> summary(gls.UN_ANCOVA)
```

```
Generalized least squares fit by REML
  Model: y ~ time.factor + treat + baseline:time.factor + time.factor:treat
  Data: dfLong.data_ncgs_ANCOVA
       AIC      BIC    logLik
  3252.519 3336.232 -1604.259
```

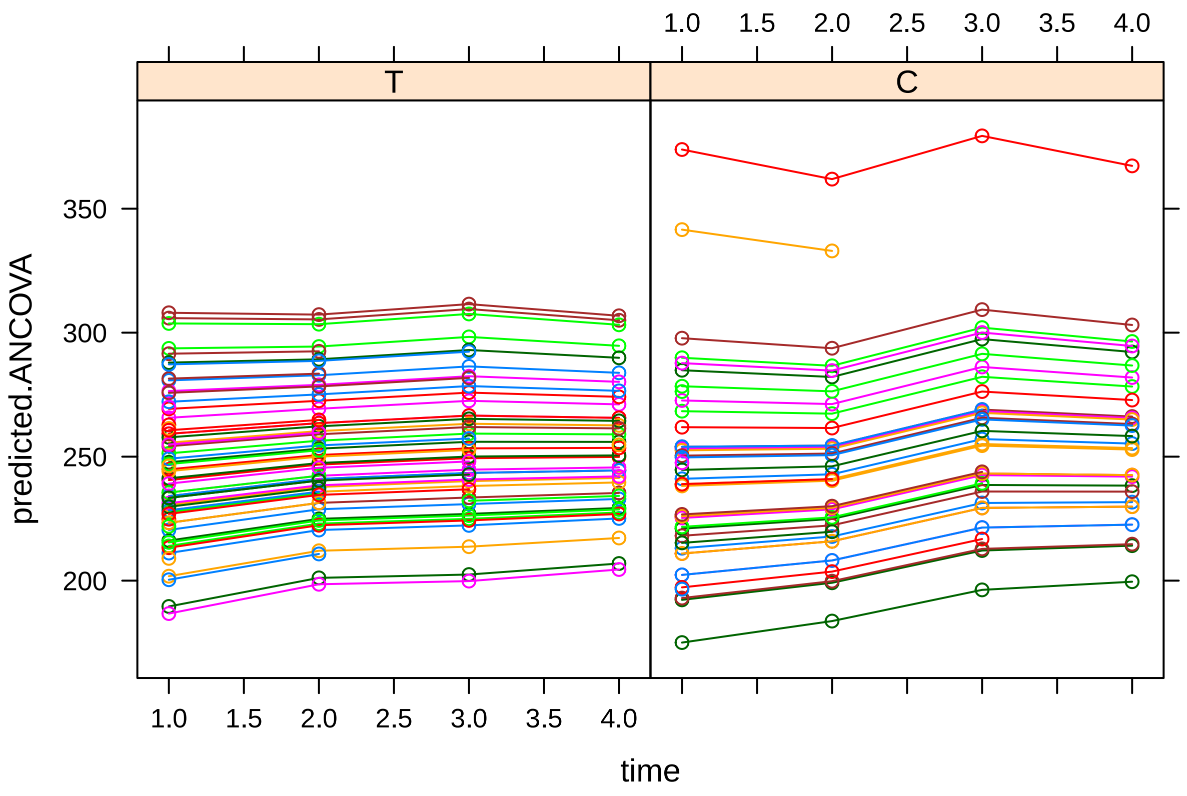
```
Correlation Structure: General
 Formula: ~1 | id
 Parameter estimate(s):
 Correlation:
  1     2     3
2 0.484
3 0.538 0.405
4 0.413 0.450 0.351
Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | time.factor
 Parameter estimates:
       1        2        3        4
1.000000 1.008466 1.021427 1.402180

Coefficients:
                          Value Std.Error   t-value p-value
(Intercept)           114.20579 23.282670  4.905184  0.0000
time.factor1          -40.38810 22.531613 -1.792508  0.0740
time.factor2          -21.24251 22.112801 -0.960643  0.3374
time.factor3          -11.12551 23.937110 -0.464781  0.6424
treatT                  3.10515  9.175648  0.338412  0.7353
time.factor4:baseline   0.60544  0.094115  6.432980  0.0000
time.factor1:baseline   0.71782  0.059465 12.071346  0.0000
time.factor2:baseline   0.64346  0.061397 10.480390  0.0000
time.factor3:baseline   0.66099  0.066151  9.992257  0.0000
time.factor1:treatT     6.43175  8.887752  0.723664  0.4698
time.factor2:treatT     9.81010  8.755954  1.120392  0.2634
time.factor3:treatT    -1.55640  9.437972 -0.164909  0.8691

 Correlation:
                      (Intr) tm.fc1 tm.fc2 tm.fc3 treatT tm.f4: tm.f1: tm.f2:
time.factor1          -0.795
time.factor2          -0.777  0.763
time.factor3          -0.765  0.778  0.711
treatT                -0.323  0.257  0.250  0.250
time.factor4:baseline -0.956  0.760  0.743  0.731  0.109
time.factor1:baseline -0.349 -0.262  0.058  0.018  0.036  0.366
time.factor2:baseline -0.383  0.102 -0.252  0.131  0.042  0.401  0.473
time.factor3:baseline -0.292 -0.006  0.064 -0.364  0.028  0.306  0.493  0.371
time.factor1:treatT    0.257 -0.324 -0.246 -0.255 -0.804 -0.089  0.029 -0.012
time.factor2:treatT    0.249 -0.245 -0.319 -0.231 -0.784 -0.086 -0.005  0.025
time.factor3:treatT    0.250 -0.255 -0.232 -0.313 -0.773 -0.087 -0.001 -0.014
                      tm.f3: tm.1:T tm.2:T
time.factor1
time.factor2
time.factor3
treatT
time.factor4:baseline
time.factor1:baseline
time.factor2:baseline
time.factor3:baseline
time.factor1:treatT    0.004
time.factor2:treatT   -0.003  0.771
time.factor3:treatT    0.025  0.786  0.720

Standardized residuals:
        Min          Q1         Med          Q3         Max
-2.60299664 -0.66011574 -0.05365112  0.70347683  2.79202320

Residual standard error: 27.94718
Degrees of freedom: 344 total; 332 residual
```

Small differences compared to SAS output, e.g.:

```
                             R        SAS
-2 log-likelihood -3208.5186764 -3208.7690
sigma2_11           781.0447801   780.6600
sigma2_12           529.7942658   383.6900
betaT                 3.1051486     3.0028
sd.betaT              9.1756484     9.2572
p_value.betaT         0.7352667     0.7465
```

```
Denom. DF: 332
                     numDF  F-value p-value
(Intercept)              1 24.06083  <.0001
time.factor              3  1.81636  0.1439
treat                    1  0.11452  0.7353
time.factor:baseline     4 47.26183  <.0001
time.factor:treat        3  1.19168  0.3129
```

```
           lower     est.    upper
treatT -14.94459 3.105149 21.15489
```


T
C
predicted.ANCOVA
350
300
250
200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
time

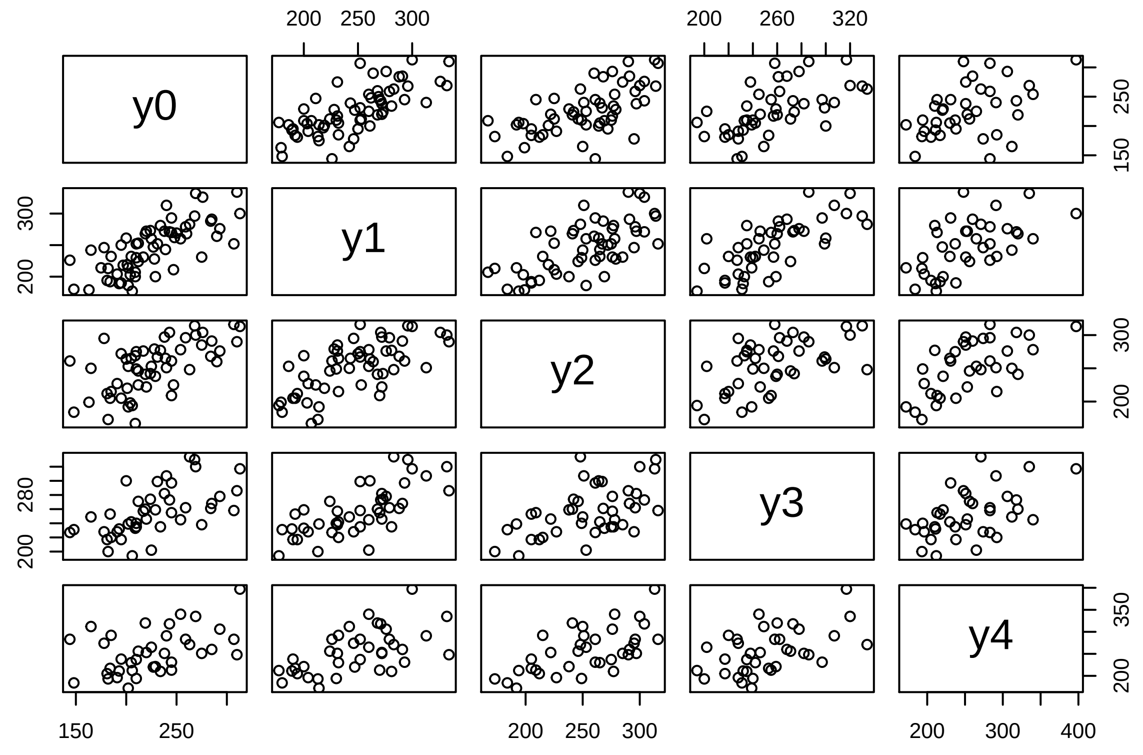
Appendix A: Alternative syntax for question 2

Compute the mean over time by group using data.table:

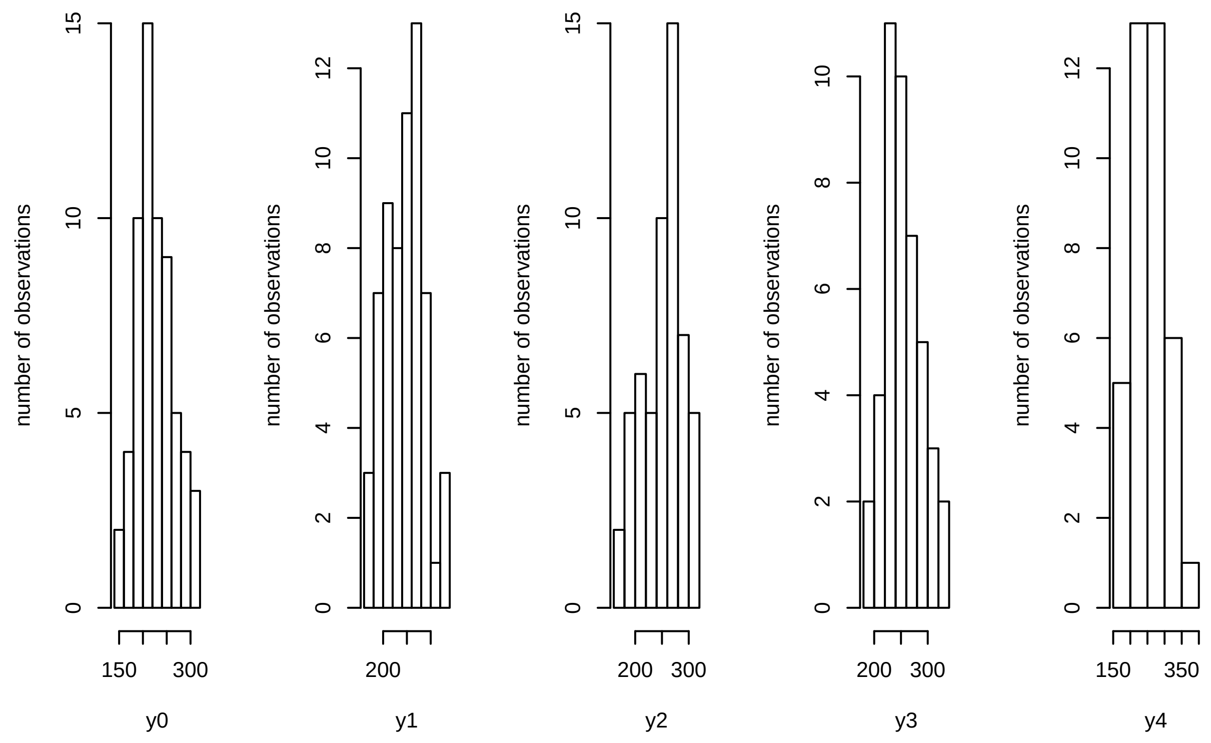
```
> dt.data_ncgs <- data.table(df.data_ncgs)
>
> dt.data_ncgs[, .(time = 0:4, colMeans(.SD, na.rm = TRUE)),
+              by = treat, .SDcols = y_columns]
>
> dt.data_ncgs[, .(time = 0:4, apply(.SD, 2, sd, na.rm = TRUE)),
+                   by = treat, .SDcols = y_columns]
```

Display exploratory graphs using plot and hist:

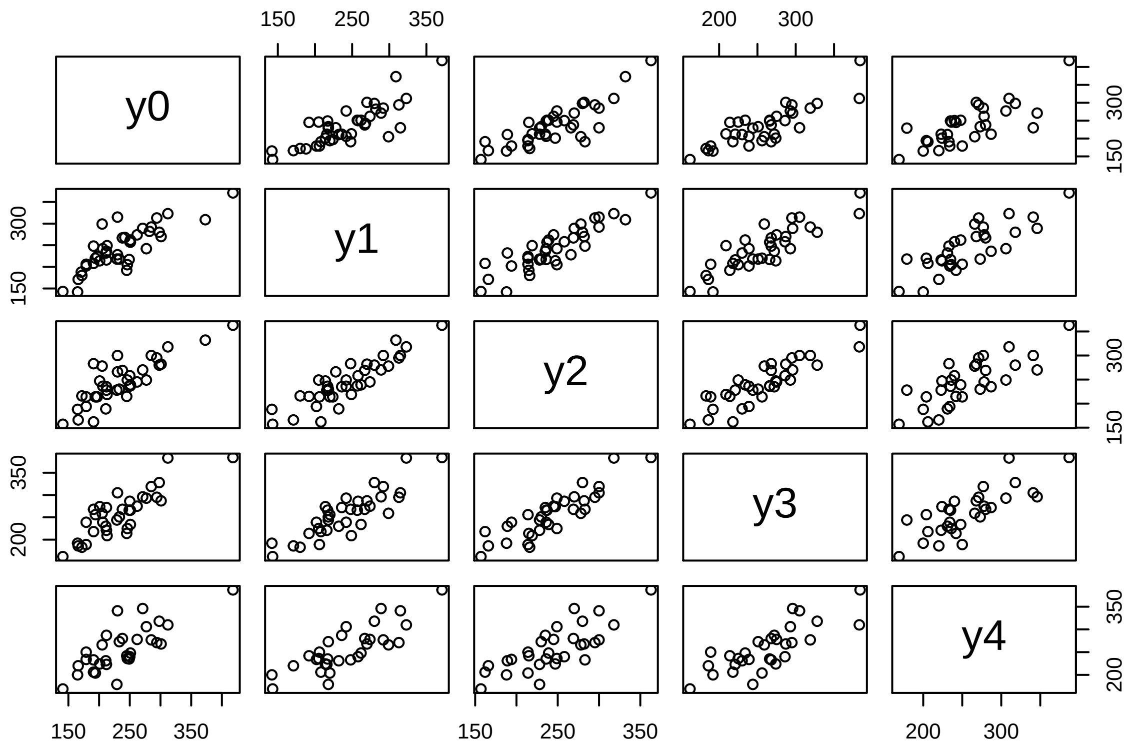
```
> ## treatment group
> plot(df.data_ncgs[index_treatment,y_columns])
```



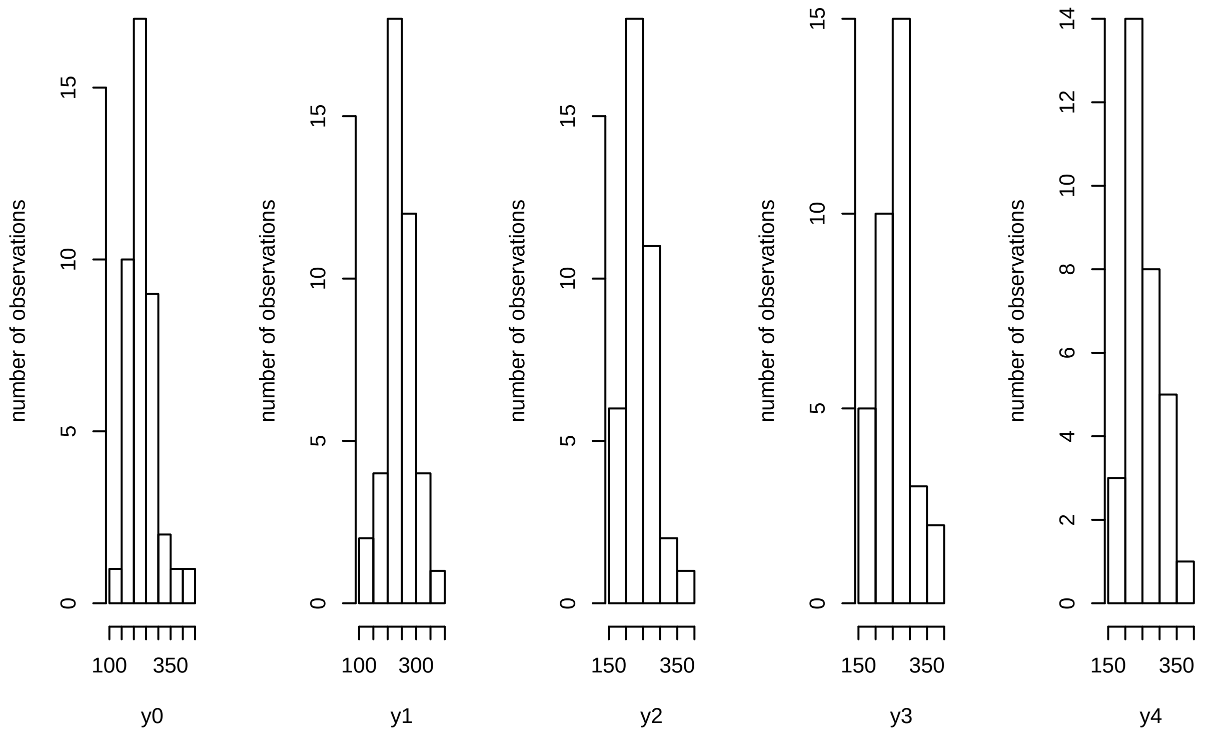
```
> par(mfrow = c(1,5)) # divide the windows into 5 five columns (and one row)
> hist(df.data_ncgs[index_treatment,"y0"], ylab = "number of observations", xlab = "y0", main = "")
> hist(df.data_ncgs[index_treatment,"y1"], ylab = "number of observations", xlab = "y1", main = "")
> hist(df.data_ncgs[index_treatment,"y2"], ylab = "number of observations", xlab = "y2", main = "")
> hist(df.data_ncgs[index_treatment,"y3"], ylab = "number of observations", xlab = "y3", main = "")
> hist(df.data_ncgs[index_treatment,"y4"], ylab = "number of observations", xlab = "y4", main = "")
```



```
> # equivalent syntax
> # res <- sapply(y_columns, function(x){
> #   hist(df.data_ncgs[index_treatment,x], ylab = "number of observations", xlab= x, main = "")
> #   })
>
> ## control group
> plot(df.data_ncgs[index_control,y_columns])
```



```
> par(mfrow = c(1,5))
> res <- sapply(y_columns, function(x){
+   hist(df.data_ncgs[index_control,x], ylab = "number of observations", xlab = x, main = "")
+ })
```



Appendix B: Fitting mixed models in R

Packages

There are several packages to fit mixed models in R. We only mention three of them here:

- *nlme package*: it enables to specify the form of the correlation structure between residuals, to model a potential heteroscedasticity and to consider random effects. It is limited to gaussian variables but can handle non-linear relationships (e.g. $Y \sim exp(\beta x)$). Main fonctions:
  - gls
  - lme
  - nlme

Reference book: Pinheiro, J.C., and Bates, D.M. (2000) “Mixed-Effects Models in S and S-PLUS”, Springer.

- *lme4 package*: it is a numerically more efficient alternative to nlme which is recommanded for large datasets or when several random effects are considered. Contrary to nlme, the correlation structure between residuals can only be model through random effects. No option for dealing with heteroscedasticity. However lme4 enable to model non-gaussian dependant variables (e.g. binary, poisson, . . .). Main functions:
  - lmer
  - glmer

  - nlmer

Reference book: Douglas M. Bates, lme4: Mixed-effects modeling with R, http://lme4.r-forge.r-project.org/lMMwR/lrgprt.pdf

- *MCMCglmm*: estimation of generalised linear mixed models using MCMC techniques. May be slower compared to lme4 but enables to consider non-gaussian responses and to specify the correlation structure between residuals using, for example, an unstructured matrix. Main functions:
  - MCMCglmm

See http://glmm.wikidot.com/pkg-comparison for a broader overview.

Arguments for functions from the nlme package

`gls`, `lme` and `nlme` take similar arguments:

- argument **correlation**: specifies the form of the covariance matrix (e.g. `correlation = corCompSymm(form = ~1 | animal)`). It is composed of three parts:
  - the structure of the correlation matrix (e.g. `corCompSymm`). See `?corClasses` for a list of the available structures.
  - the postion variable `form = ~1`: usually only an intercept but one may specify explanatory variables here if, for instance, the correlation between observation times is assumed be different regarding the age or the geographical position.
  - the grouping variable `| animal`. Here we indicates that observations are correlated within the same animal.
- argument **weight**: can be used to model a potential heteroscedasticity, e.g. variance dependant on some variables. As the correlation argument, it is composed of three parts:
  - the structure of the heteroschedasticity (e.g. `varIdent`). See `?varClasses` for a list of the available structures.
  - the postion variable `form = ~1`: usually only an intercept but one may specify explanatory variables here if, for instance, the variance is assumed to depend of a given variable.
  - the grouping variable `| time`. Here we indicates that observations observed at the same time have common variance.
- argument **na.action**: defines how to handle missing value.
  - `na.fail` (default option) leads to an error in presence of missing values.
  - `na.omit` deals with missing values by removing the corresponding lines in the dataset.
  - `na.exclude` ignores the missing values but, compared to `na.omit`, it enables to have outputs with the same number of observations compared to the original dataset.
  - `na.pass` will continue the execution of the function without any change. If the function cannot manage missing values, it will lead to an error.

Illustration:

```
> data(Orthodont, package = "nlme")
> dataset.NA <- rbind(NA, Orthodont)
> NROW(dataset.NA) # 109 observations
```

```
[1] 109
```

```
> #### na.fail
> attributes(try(
+   lme.fail <- lme(distance ~ age, data = dataset.NA)
+   ,silent = TRUE
+ ))$condition
```

```
<simpleError in na.fail.default(structure(list(age = c(NA, 8, 10, 12, 14, 8, 10, 12, 14, 8, 10, 12, 14,
```

```
> # same as:  fm1 <- lme(distance ~ age, data = dataset.NA, na.action = na.fail)
>
> #### na.omit
> lme.omit <- lme(distance ~ age, data = rbind(NA, Orthodont),
+                 na.action = na.omit)
> NROW(predict(lme.omit)) # 108 fitted values
```

```
[1] 108
```

```
> #### na.exclude
> lme.exclude <- lme(distance ~ age, data = rbind(NA, Orthodont),
+                    na.action = na.exclude)
> NROW(predict(lme.exclude)) # 109 fitted values (1 NA + 108 others)
```

```
[1] 109
```

```
> predict(lme.exclude)[1]
```

```
<NA>
  NA
```

```
> #### na.pass
> attributes(try(
+   lme.pass <- lme(distance ~ age, data = rbind(NA,Orthodont),
+                   na.action = na.pass)
+   ,silent = TRUE
+ ))$condition
```

```
<simpleError in if (max(tmpDims$ZXlen[[1L]]) < tmpDims$qvec[1L]) {    warning(gettextf("fewer observati
```