Module 3: R

3.1 Initial explorative analysis

The data set `planks` is imported as described in **R** Module 1. Assume that the data set is called `planks` in **R**.

The plots in **figure 3.2** in Module 3 are produced using the function `interaction.plot` which requires three arguments: first the factor that is to be on the x-axis, then the factor that separates the data into distinct graphs and finally the response variable. An optional parameter `legend` which takes either FALSE (F) or TRUE (T) specifies whether or not a legend should be added (relating the graphs to the factor levels)

```
with(planks, interaction.plot(width,plank,humidity,legend=F))
with(planks, interaction.plot(depth,plank,humidity,legend=F))
with(planks, interaction.plot(width,depth,humidity,legend=F))
with(planks, interaction.plot(depth,width,humidity,legend=F))
```

Notice that the `with{ ... }` function around the `interaction.plot` statements results in evaluation of the statements within a frame where the data set `planks` is available. This approach avoids having to attach data sets.

To obtain all four plots in a two-by-two setup exactly like in **figure 3.2**, the statement `par(mfrow=c(2,2))` should be issued prior to the above `with` statements. As already mentioned in the **R** Module 1, the function `par` is used to set a variety of graphical parameters (try typing `?par` for details). The parameter `mfrow` is a vector of length two where the first component is the number of rows on the graphical device and the second component is the number of columns on the graphical device. To return to the default use `par(mfrow=c(1,1))`.

3.2 Test of overall effects/model reduction

In the previous section we did not need to define factors (Module 2)to use `interaction.plot`, but now we do. Configure the three variables depth, plank and width as factors

```
planks$plank<-factor(planks$plank)
planks$depth<-factor(planks$depth)
planks$width<-factor(planks$width)
```

Analysis of models including random effects can be done using the `lme` function in the package `nlme`. The general model with fixed-effects structure consisting of the interaction between two factors and random effects assigned to the plank is specified as follows

```
model1<-lme(humidity~depth+width+depth:width,random=~1|plank,data=planks)
```

Notice that the fixed-effects structure is specified as "depth+width+depth:width" ("depth*width" would give the same result). The relevant tests of the fixed-effects structure are obtained applying `anova(model1)`

```
> anova(model1)

            numDF denDF  F-value p-value
(Intercept)     1   266 593.6487  <.0001
depth           4   266  78.2592  <.0001
width           2   266  29.6463  <.0001
depth:width     8   266 1.0840  0.3745
```

The interaction is not significant and a reduced model can be formulated

```
model2<-lme(humidity~depth+width,random=~1|plank,data=planks)
```

The tests of the two remaining terms are

```
> anova(model2)

            numDF denDF  F-value p-value
(Intercept)     1   274 593.6487  <.0001
depth           4   274  78.0676  <.0001
width           2   274  29.5738  <.0001
```

Both factors are highly significant and no further reduction is possible.

3.3 Post hoc analysis and summarizing the results

As in **R** Module 1 we can use the `estimable` function in the package `gregmisc` to compute the estimated mean levels.
In order to compute the estimated mean levels for each level of depth, we need to construct the 5×7-matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1/3 & 1/3 \\ 1 & 1 & 0 & 0 & 0 & 1/3 & 1/3 \\ 1 & 0 & 1 & 0 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 1 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 & 1 & 1/3 & 1/3 \end{pmatrix}$$

We do it in two steps: 1) A matrix with 0's in all entries is constructed. 2) The matrix is filled row by row.

```
conMat1<-matrix(0, 5, 7)

conMat1[1,]<-c(1,0,0,0,0,1/3,1/3)
conMat1[2,]<-c(1,1,0,0,0,1/3,1/3)
conMat1[3,]<-c(1,0,1,0,0,1/3,1/3)
conMat1[4,]<-c(1,0,0,1,0,1/3,1/3)
conMat1[5,]<-c(1,0,0,0,1,1/3,1/3)
```

Before the contrast matrix is ready to be supplied to the function `estimable`, we need to set the column and row names of the matrix. The row names are the factor levels (of depth) which we choose to estimate, whereas the column names are the names of all estimates as they are named in `model2`.

```
rownames(conMat1) <- c("9","1","3","5","7")
colnames(conMat1) <- names(coef(model2))
```

Now we can use the function `estimable` to obtain a 6×7-matrix containing estimates with their estimated standard deviations and 95%-confidence intervals

```
estMat1<-estimable(model2,conMat1,conf.int=0.95)
```

giving the output (selected columns only!)

```
> estMat1[, c(1, 2, 6, 7)]

  Estimate Std. Error Lower.CI Upper.CI
```

```
9 4.653333  0.2360734 4.184426 5.122241
1 4.715000  0.2360734 4.246092 5.183908
3 5.905000  0.2360734 5.436092 6.373908
5 6.195000  0.2360734 5.726092 6.663908
7 5.863333  0.2360734 5.394426 6.332241
```

Notice that the function issues a warning which is related to the degrees of freedom (p. 18 in **R** Module 1). The degrees of freedom used are different from the ones in SAS (DDFM=SATTERTH) and therefore the confidence intervals are not exactly the same as in SAS.

Similarly the estimated mean levels for each level of width can be computed. The explicit construction of the matrix `conMat1` illustrates how SAS computes LSMEANS as particular averages of estimates.

To obtain estimated contrasts between mean levels for depth, we need to construct the 10×7-matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \end{pmatrix}$$

Notice that the last two columns are not used at all as they pertain to the mean levels of width. The matrix is again constructed row-wise in **R**

```
conMat2<-matrix(0, 10, 7)

rownames(conMat2) <- c("1-9","3-9","5-9","7-9","1-3","1-5","1-7","3-5","3-7","5-7")
colnames(conMat2) <- names(coef(model2))

conMat2[1,]<-c(0,1,0,0,0,0,0)
conMat2[2,]<-c(0,0,1,0,0,0,0)
conMat2[3,]<-c(0,0,0,1,0,0,0)
conMat2[4,]<-c(0,0,0,0,1,0,0)
conMat2[5,]<-c(0,1,-1,0,0,0,0)
conMat2[6,]<-c(0,1,0,-1,0,0,0)
conMat2[7,]<-c(0,1,0,0,-1,0,0)
conMat2[8,]<-c(0,0,1,-1,0,0,0)
conMat2[9,]<-c(0,0,1,0,-1,0,0)
conMat2[10,]<-c(0,0,0,1,-1,0,0)
```

The `estimable` function can be used again

```
estMat2<-estimable(model2,conMat2,conf.int=0.95)
```

The resulting output is (selected columns only)

```
> estMat2[, c(1, 2, 6, 7, 5)]

       Estimate Std. Error   Lower.CI    Upper.CI    Pr(>|t|)
1-9  0.06166667  0.1161519 -0.1669970  0.29033030 0.595908591
3-9  1.25166667  0.1161519  1.0230030  1.48033030 0.000000000
5-9  1.54166667  0.1161519  1.3130030  1.77033030 0.000000000
7-9  1.21000000  0.1161519  0.9813364  1.43866364 0.000000000
1-3 -1.19000000  0.1161519 -1.4186636 -0.96133636 0.000000000
1-5 -1.48000000  0.1161519 -1.7086636 -1.25133636 0.000000000
1-7 -1.14833333  0.1161519 -1.3769970 -0.91966970 0.000000000
3-5 -0.29000000  0.1161519 -0.5186636 -0.06133636 0.013121976
3-7  0.04166667  0.1161519 -0.1869970  0.27033030 0.720076675
5-7  0.33166667  0.1161519  0.1030030  0.56033030 0.004626231
```

The same approach can be used to obtain estimated contrasts for width.

Bonferroni correction of the p-values is obtained using the function `p.adjust` which takes a vector of p-values to be corrected as the first argument and the type of correction as the second argument (Bonferroni is just one out of several). So we can adjust the above p-values (in the fifth column of the matrix `estMat2`) by writing

```
> p.adjust(estMat2[, c(5)], method = c("bon"))

 [1] 1.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
 [7] 0.00000000 0.13121976 1.00000000 0.04626231
```

The Tukey-Kramer correction of p-values and confidence limits can be obtained using the function `tk.adjust` which requires as argument a matrix with 3 columns. The first column should contain the contrast estimates, the second the estimated standard deviation (of the estimated contrast) and the third the degrees of freedom. We simply use the relevant three columns in `estMat2`

```
> tk.adjust(estMat2[, c(1, 2, 4)])

[1] 9.841102e-01 3.023137e-13 2.666756e-13 3.114176e-13 3.144152e-13
 [6] 2.666756e-13 3.151923e-13 9.433259e-02 9.964432e-01 3.702260e-02
```

The function `tk.adjust` is not a part of any package, but is available here:

```
tk.adjust<-function(estMat,conf.int)
{
noCI<-F

if (missing(conf.int)) {noCI<-T}
```

```
n1<-nrow(estMat)
n<-(1+sqrt(1+8*n1))/2

confWidth<-rep(0,n1)
pValue<-rep(0,n1)
for (i in 1:n1)
{
dev<-estMat[i,2]/sqrt(2)
if (!noCI) {confWidth[i]<-qtukey(conf.int,n,estMat[i,3])*dev}
pValue[i]<-ptukey(abs(estMat[i,1])/dev,n,estMat[i,3],lower.tail=F)
}

dimNames<-attr(estMat,"dimnames")
dimNames[[2]]<-c("Adj Pr(>|t|)","Adj Lower CI","Adj Upper CI")

pdiffMat<-matrix(0,n1,3,dimnames=dimNames)
pdiffMat[,2]<-estMat[,1]-confWidth
pdiffMat[,3]<-estMat[,1]+confWidth
pdiffMat[,1]<-pValue

if (noCI) {return(pdiffMat[,1])}
else {return(pdiffMat)}
}
```

Adjusted confidence limits can also be obtained specifying the option `conf.int` as in the following example (a 95% confidence interval). The estimates and their estimated standard deviations and the adjusted confidence limits and p-values are bound together using the function `cbind`.

```
> cbind(estMat2[, c(1, 2)], tk.adjust(estMat2[, c(1, 2, 4)],conf.int = 0.95))

       Estimate Std. Error Adj Pr(>|t|) Adj Lower CI Adj Upper CI
1-9  0.06166667  0.1161519 9.841102e-01  -0.25729003   0.38062336
3-9  1.25166667  0.1161519 3.023137e-13   0.93270997   1.57062336
5-9  1.54166667  0.1161519 2.666756e-13   1.22270997   1.86062336
7-9  1.21000000  0.1161519 3.114176e-13   0.89104330   1.52895670
1-3 -1.19000000  0.1161519 3.144152e-13  -1.50895670  -0.87104330
1-5 -1.48000000  0.1161519 2.666756e-13  -1.79895670  -1.16104330
1-7 -1.14833333  0.1161519 3.151923e-13  -1.46729003  -0.82937664
3-5 -0.29000000  0.1161519 9.433259e-02  -0.60895670   0.02895670
3-7  0.04166667  0.1161519 9.964432e-01  -0.27729003   0.36062336
5-7  0.33166667  0.1161519 3.702260e-02   0.01270997   0.65062336
```

This output is identical to the SAS output using the option ADJUST=TUKEY.