MA50259: Statistical Design of Investigations

Lab sheet 1: Completely randomised design

In this practical you will learn how to simulate and analyse your own completely randomised experimental design. Take the time to run each of the following commands and analyse the displayed results to understand what the code is doing.

1. Install and load the `tidyverse` package

```r
install.packages('tidyverse')
```

```r
library('tidyverse')
```

2. Set the number of treatment levels, `t` to 6 and the number of replicates per treatment level `r` (balanced design) to 10. This determines the number of experimental units `n`, that is, 60.

```r
r<-10
t<-6
n<-t*r
```

3. Create a character vector with the names of the treatment levels

```r
levels<-c("level 1","level 2","level 3","level 4", "level 5","level 6")
levels
class(levels)
```

4. Create a vector of length `n` with `r` replicates of each of the `t` treatment levels

```r
v<-rep(levels,each = r)
v
class(v)
```

5. Convert the vector `v` into a factor that R can manage for statistical analysis

```r
f <- factor(v)
f
class(f)
```

6. Two alternative ways to create the vector `f` are as follows

```
# alternative code 1
 f <- factor(rep(levels,each = r))
# Nesting the 2 commands avoids the unnecessary intermediate vector v

# alternative code 2
 f <- rep(levels,each = r) %>% factor()
#  Pipe %>% is equivalent the nested commands above but makes clear what is evaluated first
```

7. The following commands perform the randomisation of the experimental units to treatement levels as a random re-shuffling of the factor f. You may want to have a look at the help file of the R command `sample`

```
set.seed(5678)
1:n # row labels
rows<-sample(1:n,n) # random re-shuffling of rows
rows
fac <- f[rows]
fac
# set.seed is used to control randomness if we do not set the seed then we obtain different results
# everytime we run the program!!! You can test that yourself!
```

8. An alternative code for creating the factor `fac` is as follows (as it is presented in the lecture slides):

```
set.seed(5678)
fac <- sample(f,size=n)
# fac <- f %>% sample(size=n) alternative code
```

9. Now, create a data.frame (a dataset, basically) with two variables: the (randomised) factor `fac` and a vector of simple labels for the units, namely, from 1 to n

```
units <- 1:n # unit labels
crd <- tibble(units=units, treatment=fac ) # creates data.frame (called tibble in the tidyverse)
crd
```

```
## # A tibble: 60 x 2
##    units treatment
##    <int> <fct>
##  1     1 level 4
##  2     2 level 5
##  3     3 level 6
##  4     4 level 3
##  5     5 level 3
##  6     6 level 6
##  7     7 level 1
##  8     8 level 3
##  9     9 level 3
## 10    10 level 6
## # i 50 more rows
```

```
glimpse(crd)
```

```
## Rows: 60
## Columns: 2
## $ units     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ treatment <fct> level 4, level 5, level 6, level 3, level 3, level 6, level ~
```

10. Re-arrange the rows of the data.frame `crd` so that experimental units with the same treatment levels are together and the treatments levels are in their original order: `level 1`, `level 2`, etc.

```
crd<-arrange(crd,treatment)  # arrange by treatement level
# alternative code
# crd<- crd %>% arrange(treatment)
glimpse(crd)
```

11. Note we did the labeling of experimental units after the randomisation! This might not be appropriate if (for any reason) we need to identify some experimental units with their original labels. The following code performs this and creates a new data.frame called `crd1`

```
crd1 <- tibble(units=units, treatment=f )
crd1<-crd1[rows,]
crd1
crd1<- crd1 %>% arrange(treatment)
crd1
```

Note the two data.frames `crd` and `crd1` are different since the experimental units are labelled differently.

12. Now we simulate the response values using a simple model. We set the mean response for each treatment levels as follows: first set the overall mean response μ and also set the effects τ_i of each treatment level. The mean response values for each treament level is given by $\mu + \tau_i$

```
mu<-500 # overall mean response
tau<-c(-20,50,0,-30,-10,100) # treatment effects over mu
means<-mu+tau # mean response for each treament level
means
```

13. Create a vector of length `n` with `r` replicates of each of the `t` different mean response values

```
means<-mu+tau %>% rep(each=r) # vector of means
# means<-rep(mu+tau,each=r) # alternative code
means
```

14. Set the standard deviation σ and simulate the response values for all the experimental units by simulating Gaussian (Normal) random numbers with means given by `means` and common standard deviation σ

```
sd<-10 # common standard deviation
y<-rnorm(n,mean=means,sd=sd) # note the units should be arranged by treatment level
y
```

15. Append the generated vector of response values to the `crd1` data.frame

```
crd1$response<-y
glimpse(crd1)
```

16. Plot the data in `crd1` using `ggplot`

```
ggplot(crd1,aes(treatment,response))+geom_point()
ggplot(crd1,aes(treatment,response))+geom_boxplot()
```

17. Analyse the data using a linear model via the `lm` command in R

```
mod.crd1<-lm(response~treatment,data=crd1)
coefs<-coef(mod.crd1)
coefs
```

The estimates given, correspond to estimates of μ, τ_2, τ_3, τ_4, τ_5 and τ_6 so you can compare them with the actual population values.

```
c(mu,tau[-1])
```

The estimate of τ_1 has been arbitrarily set to zero!

18. The estimates of the mean values for each level of the treatment effect are given as follows

```
taus<-c(0,coefs[-1])
means2<-coefs[1]+taus
means2
```

You can compare these with the actual population values

```
mu+tau
```

19. It turns out that the above mean estimates are exactly the same as the sample response means for each level of the treatment factor.

```
by_group <- group_by(crd1, treatment) #groups the data frame
by_group
summaries.crd<-summarize(by_group, means = mean(response))
# applies the mean function for each group defined by the levels of the treatment factor
glimpse(summaries.crd)
```