So you’ve run an ANOVA and found that your residuals are neither normally distributed, nor homogeneous, or that you are in violation of any other assumptions. *Naturally* you want to run some a-parametric analysis… but how?

In this post I will demonstrate how to run a permutation test ANOVA (easy!) and how to run bootstrap follow-up analysis (a bit more challenging) in a mixed design (both within- and between-subject factors) ANOVA. I’ve chosen to use a mixed design model in this demonstration for two reasons:

1. I’ve never seen this done before.
2. You can easily modify this code (change / skip some of these steps) to accommodate purely within- or purely between-subject designs.

**Permutation ANOVA**

Running a permutation test for your ANOVA in R is as easy as… running an ANOVA in R, but substituting aov with aovperm from the permuco package.

library(permuco)  
data(obk.long, package = "afex") # data from the afex package   
  
# permutation anova  
fit\_mixed\_p <-  
 aovperm(value ~ treatment \* gender \* phase \* hour + Error(id / (phase \* hour)),  
 data = obk.long)

fit\_mixed\_p

##   
## Permutation test using Rd\_kheradPajouh\_renaud to handle nuisance variables and 5000 permutations.  
## SSn dfn SSd dfd MSEn MSEd F  
## treatment 179.730 2 228.06 10 89.8652 22.806 3.94049  
## gender 83.448 1 228.06 10 83.4483 22.806 3.65912  
## treatment:gender 130.241 2 228.06 10 65.1206 22.806 2.85547  
## phase 129.511 2 80.28 20 64.7557 4.014 16.13292  
## treatment:phase 77.885 4 80.28 20 19.4713 4.014 4.85098  
## gender:phase 2.270 2 80.28 20 1.1351 4.014 0.28278  
## treatment:gender:phase 10.221 4 80.28 20 2.5553 4.014 0.63660  
## hour 104.285 4 62.50 40 26.0714 1.563 16.68567  
## treatment:hour 1.167 8 62.50 40 0.1458 1.563 0.09333  
## gender:hour 2.814 4 62.50 40 0.7035 1.562 0.45027  
## treatment:gender:hour 7.755 8 62.50 40 0.9694 1.562 0.62044  
## phase:hour 11.347 8 96.17 80 1.4183 1.202 1.17990  
## treatment:phase:hour 6.641 16 96.17 80 0.4151 1.202 0.34529  
## gender:phase:hour 8.956 8 96.17 80 1.1195 1.202 0.93129  
## treatment:gender:phase:hour 14.155 16 96.17 80 0.8847 1.202 0.73594  
## parametric P(>F) permutation P(>F)  
## treatment 5.471e-02 0.0586  
## gender 8.480e-02 0.0916  
## treatment:gender 1.045e-01 0.1084  
## phase 6.732e-05 0.0002  
## treatment:phase 6.723e-03 0.0056  
## gender:phase 7.566e-01 0.7644  
## treatment:gender:phase 6.424e-01 0.6480  
## hour 4.027e-08 0.0002  
## treatment:hour 9.992e-01 0.9996  
## gender:hour 7.716e-01 0.7638  
## treatment:gender:hour 7.555e-01 0.7614  
## phase:hour 3.216e-01 0.3260  
## treatment:phase:hour 9.901e-01 0.9910  
## gender:phase:hour 4.956e-01 0.5100  
## treatment:gender:phase:hour 7.496e-01 0.7590

The results of the permutation test suggest an interaction between Treatment (a between subject factor) and Phase (a within-subject factor). To fully understand this interaction, we would like to conduct some sort of follow-up analysis (planned comparisons or post hoc tests). Usually I would pass the model to emmeans for any follow-ups, but here, due to our assumption violations, we need some sort of bootstrapping method.

**Bootstrapping with car and emmeans**

For bootstrapping we will be using the Boot function from the carpackage. In the case of within-subject factors, this function requires that they be specified in a multivariate data structure. Let’s see how this is done.

**1. Make your data WIIIIIIIIIIDEEEEEEEE**

library(dplyr)

library(tidyr)

obk\_mixed\_wide <- obk.long %>%  
 unite("cond", phase, hour) %>%  
 spread(cond, value)  
  
head(obk\_mixed\_wide)

## id treatment gender age fup\_1 fup\_2 fup\_3 fup\_4 fup\_5 post\_1 post\_2  
## 1 1 control M -4.75 2 3 2 4 4 3 2  
## 2 2 control M -2.75 4 5 6 4 1 2 2  
## 3 3 control M 1.25 7 6 9 7 6 4 5  
## 4 4 control F 7.25 4 4 5 3 4 2 2  
## 5 5 control F -5.75 4 3 6 4 3 6 7  
## 6 6 A M 7.25 9 10 11 9 6 9 9  
## post\_3 post\_4 post\_5 pre\_1 pre\_2 pre\_3 pre\_4 pre\_5  
## 1 5 3 2 1 2 4 2 1  
## 2 3 5 3 4 4 5 3 4  
## 3 7 5 4 5 6 5 7 7  
## 4 3 5 3 5 4 7 5 4  
## 5 8 6 3 3 4 6 4 3  
## 6 10 8 9 7 8 7 9 9

This is not enough, as we *also* need our new columns (representing the different levels of the within subject factors) to be in a matrix column.

obk\_mixed\_matrixDV <- obk\_mixed\_wide %>%  
 select(id, age, treatment, gender)  
obk\_mixed\_matrixDV$M <- obk\_mixed\_wide %>%  
 select(-id, -age, -treatment, -gender) %>%  
 as.matrix()  
  
str(obk\_mixed\_matrixDV)

## 'data.frame': 16 obs. of 5 variables:  
## $ id : Factor w/ 16 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...  
## $ age : num -4.75 -2.75 1.25 7.25 -5.75 7.25 8.25 2.25 2.25 -7.75 ...  
## $ treatment: Factor w/ 3 levels "control","A",..: 1 1 1 1 1 2 2 2 2 3 ...  
## $ gender : Factor w/ 2 levels "F","M": 2 2 2 1 1 2 2 1 1 2 ...  
## $ M : num [1:16, 1:15] 2 4 7 4 4 9 8 6 5 8 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : NULL  
## .. ..$ : chr "fup\_1" "fup\_2" "fup\_3" "fup\_4" ...

**2. Fit your *regular* model**

fit\_mixed <- aov(M ~ treatment \* gender, obk\_mixed\_matrixDV)

Note that the left-hand-side of the formula (the M) is a matrix representing all the within-subject factors and their levels, and the right-hand-side of the formula (treatment \* gender) includes only the between-subject factors.

**3. Define the contrast(s) of interest**

For this step we will be using emmeans. But first, we need to create a list of the within-subject factors and their levels (I did say this was difficult – bear with me!). This list needs to correspond to the order of the multi-variate column in our data, such that if there is more than one factor, the combinations of factor levels are used in expand.grid order. In our case:

colnames(obk\_mixed\_matrixDV$M)

## [1] "fup\_1" "fup\_2" "fup\_3" "fup\_4" "fup\_5" "post\_1" "post\_2"  
## [8] "post\_3" "post\_4" "post\_5" "pre\_1" "pre\_2" "pre\_3" "pre\_4"   
## [15] "pre\_5"

rm\_levels <- list(hour = c("1", "2", "3", "4", "5"),  
 phase = c("fup", "post", "pre"))

Make sure you get the order of the variables and their levels correct! This will affect your results!

Let’s use emmeans to get the estimates of the pairwise differences between the treatment groups within each phase of the study:

library(emmeans)  
# get the correct reference grid with the correct ultivariate levels!  
rg <- ref\_grid(fit\_mixed, mult.levs = rm\_levels)  
rg

## 'emmGrid' object with variables:  
## treatment = control, A, B  
## gender = F, M  
## hour = multivariate response levels: 1, 2, 3, 4, 5  
## phase = multivariate response levels: fup, post, pre

# get the expected means:  
em\_ <- emmeans(rg, ~ phase \* treatment)  
em\_

## phase treatment emmean SE df lower.CL upper.CL  
## fup control 4.33 0.551 10 3.11 5.56  
## post control 4.08 0.628 10 2.68 5.48  
## pre control 4.25 0.766 10 2.54 5.96  
## fup A 7.25 0.604 10 5.90 8.60  
## post A 6.50 0.688 10 4.97 8.03  
## pre A 5.00 0.839 10 3.13 6.87  
## fup B 7.29 0.461 10 6.26 8.32  
## post B 6.62 0.525 10 5.45 7.80  
## pre B 4.17 0.641 10 2.74 5.59  
##   
## Results are averaged over the levels of: gender, hour   
## Confidence level used: 0.95

# run pairwise tests between the treatment groups within each phase  
c\_ <- contrast(em\_, "pairwise", by = 'phase')  
c\_

## phase = fup:  
## contrast estimate SE df t.ratio p.value  
## control - A -2.9167 0.818 10 -3.568 0.0129   
## control - B -2.9583 0.719 10 -4.116 0.0054   
## A - B -0.0417 0.760 10 -0.055 0.9983   
##   
## phase = post:  
## contrast estimate SE df t.ratio p.value  
## control - A -2.4167 0.931 10 -2.595 0.0634   
## control - B -2.5417 0.819 10 -3.105 0.0275   
## A - B -0.1250 0.865 10 -0.144 0.9886   
##   
## phase = pre:  
## contrast estimate SE df t.ratio p.value  
## control - A -0.7500 1.136 10 -0.660 0.7911   
## control - B 0.0833 0.999 10 0.083 0.9962   
## A - B 0.8333 1.056 10 0.789 0.7177   
##   
## Results are averaged over the levels of: gender, hour   
## P value adjustment: tukey method for comparing a family of 3 estimates

# extract the estimates  
est\_names <- c("fup: control - A", "fup: control - B", "fup: A - B",  
 "post: control - A", "post: control - B", "post: A - B",  
 "post: control - A", "post: control - B", "post: A - B")  
est\_values <- summary(c\_)$estimate  
names(est\_values) <- est\_names  
est\_values

## fup: control - A fup: control - B fup: A - B post: control - A   
## -2.91666667 -2.95833333 -0.04166667 -2.41666667   
## post: control - B post: A - B post: control - A post: control - B   
## -2.54166667 -0.12500000 -0.75000000 0.08333333   
## post: A - B   
## 0.83333333

**4. Running the bootstrap**

Now let’s wrap this all in a function that accepts the fitted model as an argument:

treatment\_phase\_contrasts <- function(mod){  
 rg <- ref\_grid(mod, mult.levs = rm\_levels)  
  
 # get the expected means:  
 em\_ <- emmeans(rg, ~ phase \* treatment)  
  
 # run pairwise tests between the treatment groups within each phase  
 c\_ <- contrast(em\_, "pairwise", by = 'phase')  
  
 # extract the estimates  
 est\_names <- c("fup: control - A", "fup: control - B", "fup: A - B",  
 "post: control - A", "post: control - B", "post: A - B",  
 "post: control - A", "post: control - B", "post: A - B")  
 est\_values <- summary(c\_)$estimate  
 names(est\_values) <- est\_names  
 est\_values  
}  
  
# test it  
treatment\_phase\_contrasts(fit\_mixed)

## fup: control - A fup: control - B fup: A - B post: control - A   
## -2.91666667 -2.95833333 -0.04166667 -2.41666667   
## post: control - B post: A - B post: control - A post: control - B   
## -2.54166667 -0.12500000 -0.75000000 0.08333333   
## post: A - B   
## 0.83333333

Finally, we will use car::Boot to get the bootstrapped estimates!

library(car)

treatment\_phase\_results <-  
 Boot(fit\_mixed, treatment\_phase\_contrasts, R = 50) # R = 599 at least

summary(treatment\_phase\_results) # original vs. bootstrapped estimate (bootMed)

##   
## Number of bootstrap replications R = 27   
## original bootBias bootSE bootMed  
## fup..control...A -2.916667 0.0342593 0.58002 -3.05000  
## fup..control...B -2.958333 -0.0246914 0.73110 -2.96667  
## fup..A...B -0.041667 -0.0589506 0.35745 -0.16667  
## post..control...A -2.416667 -0.1728395 0.65088 -2.75000  
## post..control...B -2.541667 -0.1425926 0.77952 -2.66667  
## post..A...B -0.125000 0.0302469 0.58006 -0.11667  
## post..control...A.1 -0.750000 0.0067901 0.83692 -0.56667  
## post..control...B.1 0.083333 -0.0169753 0.89481 0.33333  
## post..A...B.1 0.833333 -0.0237654 0.73113 1.08333

confint(treatment\_phase\_results, type = "perc") # does include zero?

## Bootstrap percent confidence intervals  
##   
## 2.5 % 97.5 %  
## fup..control...A -4.000000 -1.750000  
## fup..control...B -4.300000 -1.500000  
## fup..A...B -0.750000 0.750000  
## post..control...A -3.500000 -1.333333  
## post..control...B -4.250000 -1.333333  
## post..A...B -1.416667 0.875000  
## post..control...A.1 -2.600000 0.700000  
## post..control...B.1 -2.000000 1.500000  
## post..A...B.1 -0.600000 1.833333

Results indicate that the Control group is lower than both treatment groups in the post and fup (follow -up) phases.

If we wanted p-values, we could use this little function:

boot\_pvalues <- function(x, side = c(0, -1, 1)) {  
 side <- side[1]  
 x <- as.data.frame(x$t)  
  
 ps <- sapply(x, function(.x) {  
 s <- na.omit(.x)  
 s0 <- 0  
 N <- length(s)  
  
 if (side == 0) {  
 min((1 + sum(s >= s0)) / (N + 1),  
 (1 + sum(s <= s0)) / (N + 1)) \* 2  
 } else if (side < 0) {  
 (1 + sum(s <= s0)) / (N + 1)  
 } else if (side > 0) {  
 (1 + sum(s >= s0)) / (N + 1)  
 }  
 })  
  
 setNames(ps,colnames(x))  
}  
  
boot\_pvalues(treatment\_phase\_results)

## fup: control - A fup: control - B fup: A - B post: control - A   
## 0.07142857 0.07142857 0.71428571 0.07142857   
## post: control - B post: A - B post: control - A post: control - B   
## 0.07142857 0.85714286 0.42857143 0.85714286   
## post: A - B   
## 0.35714286

These p-values can then be passed to p.adjust() for the p-value adjustment method of your choosing.

**Summary**

I’ve demonstrated how to run permutation tests on main effects / interactions, with follow-up analysis using the bootstrap method. Using this code as a basis for any analysis you might have in mind gives you all the flexibility of emmeans, which supports many (many) models!

[](https://i0.wp.com/2.bp.blogspot.com/-K9U-zJEOtlo/XNRObvcih7I/AAAAAAAA7E8/CS2KVjO3I0gofb5BsyXXjfH9Ye99LL0VgCLcBGAs/s1600/Picture1.png?ssl=1)