ANOVA

author: Maria Paniw date: 09.02.2016

What is ANOVA

The **analysis of variance**, or ANOVA, is a technique to partition the sum of squares (or the variance) of our models similar to linear regression. The main difference to linear regression is that ANOVA assumes that the predictor is categorical and the response is continuous.

I will not go into detail on variance decomposition in ANOVA as we did that already with linear regression.

To see how ANOVA works in R, lets get right to an example:

Let's assume we measure the mean weight of 20 birds in 4 populations. The mean weights are 2.5, 4.6, 2.9, 2.8 kg. We can simulate the data as follows:

```
npop=4 # number of populations
nbirds=20 # samples per populations
sigma=1.5 # residual standaed deviation

n = npop*nbirds #total number of samples

set.seed(20)
eps= rnorm(n,0,sigma) # random variation

X=rep(c("A","B","C","D"),each=nbirds) # indicator for pop
```

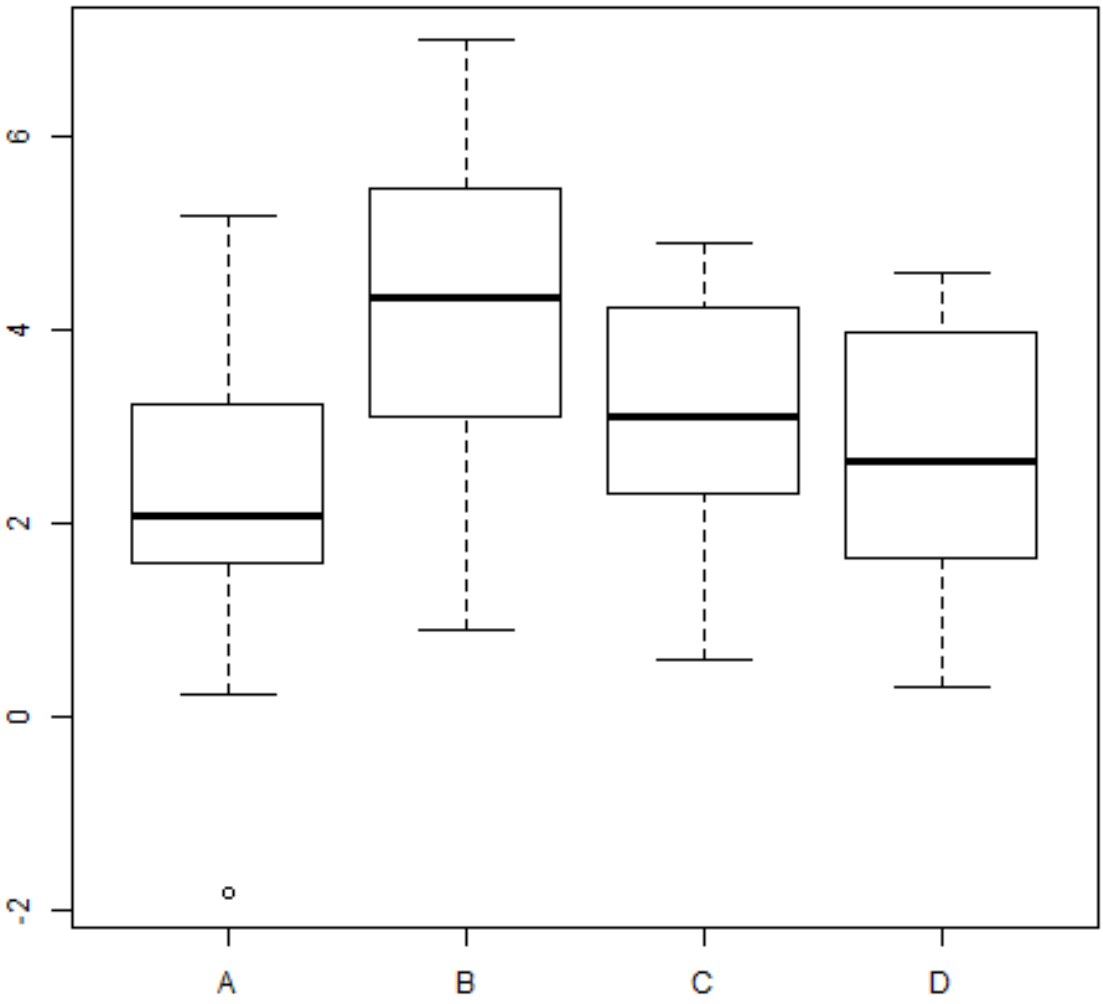
```
# Factors in R are used to group other variables (in our case, Y)
# based on their levels
X=as.factor(X) # factors are critical for statistical analyses in R!

means=rep(c(2.5,4.6,2.9,2.8),each=nbirds)

Y=means+eps

data=data.frame(pop=X,weight=Y)
```

```
boxplot(weight~pop,data)
```



ANOVA in R

How do we analyze our data (weight as a function of population) in R?

There are two ways of parameterizing this model. Option 1 (default in R) is the *effects* parameterization.

$$weight_i = \alpha + \beta_{j(i)} * pop_i + \epsilon_i$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

This parameterization is very similar to linear regression. We have a mean, α, which represents the mean of

population A. The indicators j represent the differences of the means of populations B-D to the mean of A. The choice of reference is arbitrary (in R the fist level in alphabetical order) and has no effect on the result.

In R, we create the following model matrix:

```
model.matrix(~pop,data)[15:22,] # the first 20 entries define pop A,
```

```
   (Intercept) popB popC popD
15           1    0    0    0
16           1    0    0    0
17           1    0    0    0
18           1    0    0    0
19           1    0    0    0
20           1    0    0    0
21           1    1    0    0
22           1    1    0    0
```

```
# entries 21-40 define the difference between A and B,
# entries 41-60 define the difference between A and C, etc.
```

```
#Fitting the lm, we get the following response:

mod=lm(weight~pop,data)

summary(mod)
```

```
Call:
lm(formula = weight ~ pop, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0529 -0.8448 -0.0085  1.1047  2.9598

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.2184     0.3328   6.666 3.71e-09 ***
popB          2.1239     0.4706   4.513 2.29e-05 ***
popC          0.9102     0.4706   1.934   0.0568 .
popD          0.4135     0.4706   0.879   0.3823

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.488 on 76 degrees of freedom
Multiple R-squared:  0.2318,    Adjusted R-squared:  0.2014
F-statistic: 7.643 on 3 and 76 DF,  p-value: 0.0001568
```

Pairwise comparisons

You can of course test significant pairwise differences between treatment levels:

```
pairwise.t.test(data$weight, data$pop, p.adjust = "bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$weight and data$pop

  A       B       C
B 0.00014 -       -
C 0.34085 0.07106 -
D 1.00000 0.00303 1.00000

P value adjustment method: bonferroni
```

The `pairwise.t.test()` is a simple method. We will go over mode complicated pairwise comparisons when we talk about glm.

Look at the residual standard error - it is what we defined as the residual sd when simulating the data. The mean weight in pop A is 2.2. Our simulation said 2.5 - the prediction is pretty close, given the small sample size.

The mean weight at pop B is $2.2 + 2.1 = 4.3$; again, pretty close.

Of course, we can once more use matrix algebra:

```
X=as.matrix(model.matrix(~pop,data))
beta=solve(crossprod(X))%*%crossprod(X,Y)

beta # Magic!!! (not really)
```

```
                [,1]
(Intercept) 2.2183542
popB        2.1238751
popC        0.9102465
popD        0.4135438
```

Option 2

The other option to parameterize an ANOVA model is the *means* parameterization. It looks like this:

$$weight_i = \alpha_{j(i)} * pop_i + \epsilon_i$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

The parameter α_j now represents the mean weight in each of the four populations.

In R, we create the following model matrix:

```
model.matrix(~pop-1,data)[15:22,] # the first 20 entries define pop A,
```

```
   popA popB popC popD
15    1    0    0    0
16    1    0    0    0
17    1    0    0    0
18    1    0    0    0
19    1    0    0    0
20    1    0    0    0
21    0    1    0    0
22    0    1    0    0
```

```
# entries 21-40 define mean weight in pop B, etc.
```

```
#Fitting the lm, we get the following response:

mod=lm(weight~pop-1,data) # include the -1 to fit the means model

summary(mod)# The significance has no meaning here!!!!
```

```
Call:
lm(formula = weight ~ pop - 1, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0529 -0.8448 -0.0085  1.1047  2.9598

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
popA   2.2184     0.3328   6.666 3.71e-09 ***
popB   4.3422     0.3328  13.049  < 2e-16 ***
popC   3.1286     0.3328   9.402 2.28e-14 ***
popD   2.6319     0.3328   7.909 1.64e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.488 on 76 degrees of freedom
Multiple R-squared:  0.8279,    Adjusted R-squared:  0.8189
F-statistic: 91.42 on 4 and 76 DF,  p-value: < 2.2e-16
```

Estimating the parameters using matrix algebra works here as well, of course.