Advanced Lab 4: More Linear Modeling and ANOVA

Introduction: Multiple predictors

We will now continue the project the previous lab, and start considering more complex models, with more than one predictor. We will start with a model including all the predictors:

```
fitAll <- lm(Black~., data=targetingRaceDiff)
summary(fitAll)
```

Let's take a closer look at this model. We have:

- The intercept
- A term for the subject ID. We should probably remove this.
- The coefficient for iat.
- A factor level for male gender. Presumably the female case is the base case.
- Factor levels for age. Notice that because age is treated as an ordered variable, R chooses to code it using *polynomial contrasts*, which in effect assume that the age values are equidistant and in order. This might not be an accurate assumption in our case.
- A factor level for the weapon value being set to unarmed, with base case being armed.
- A factor level for action incorrect, with base case being the correct action.
- The coefficient for White.

Calling anova on the model will tell us about each factor as a whole, and whether its contribution is statistically significant:

```
anova(fitAll)
```

This table identifies for us the contribution towards the overall variability in the sense of "sum of squared residuals" that comes from each of the predictors. The F-values compare the full model to the model without the corresponding predictor.

We see that the predictor with the largest P-value, and hence smaller contribution, is the subject ID. This is good, there is no reason whatsoever that the subject ID should have anything to do with the data. Let's remove it and reconsider:

```
fitNoSubj <- lm(Black~.-subject, data=targetingRaceDiff)
summary(fitNoSubj)
anova(fitNoSubj)
```

We can see that the residual standard error remained essentially the same with the removal of the subject variable. And we also see a number of other variables that are potentially not influential. We now want to consider dropping other variables from the model.

One powerful utility that R offers is the ability to remove one variable at a time from a model, and consider all the resulting smaller models. The drop1 method is one way to proceed for that. It will compare the current model with all the models resulting from dropping (in turn) each of the current predictors.

```
drop1(fitNoSubj)
drop1(fitNoSubj, test="F")     # If you want to include F-tests
```

What we see is the effect of removing each of the variables, in terms of how much the RSS will change. The AIC column portrays the **Akaike Information Criterion**. The AIC is a number that takes into consideration the RSS but also penalizes models based on the number of parameters (p) used. It is technically given by the formula:

$$-2\log(\text{likelihood}) + 2p$$

Here the likelihood is essentially the probability of observing the data that was observed under the assumptions in our model. Our model-fitting procedure chooses the model coefficients so as to maximize the likelihood.

A larger model will fit the data better, and so will have a higher log(likelihood) and therefore a lower value for $-2\log(\text{likelihood})$. But it will also have more parameters, so $2p$ will be higher. The formula aims to balance the two: The *lower* the AIC the more you have gained by the bigger model, while also accounting for how much bigger the model was. A model with *lower* AIC is fitting the data "more economically". There is often an indeterminate additive constant involved in AIC computations, so the AIC values are only appropriate for comparisons between models.

The first line, marked <none>, contains the values for the full model, while the remaining lines correspond to the models with the respective predictors removed.

In this case we see that the biggest reduction in AIC can be obtained by dropping the age variable. This would make sense since that variable, coded as a factor variable, uses 5 parameters. If you carried out the line with test="F", you will notice that age is not the predictor with the largest P-value. This is because the F-test does not penalize models as much for how many parameters they use.

Let's remove age and consider the model again. Here we use the update function, which allows us to update an existing model and change one aspect of it. The formula .~.-age in the update function tells it that it should use the model as is but remove the age variable.

```
fitNoSubjNoAge <- fitNoSubj %>% update(formula=.~.-age)
summary(fitNoSubjNoAge)
anova(fitNoSubjNoAge)
drop1(fitNoSubjNoAge)
```

We now see iat as a next possible variable to remove.

```
fitNoSubjNoAgeNoIat <- fitNoSubjNoAge %>% update(formula=.~.-iat)
summary(fitNoSubjNoAgeNoIat)
anova(fitNoSubjNoAgeNoIat)
drop1(fitNoSubjNoAgeNoIat)
```

Removing gender seems to be the next step:

```
fitNoSubjNoAgeNoIatNoGender <- fitNoSubjNoAgeNoIat %>% update(formula=.~.-gender)
summary(fitNoSubjNoAgeNoIatNoGender)
anova(fitNoSubjNoAgeNoIatNoGender)
drop1(fitNoSubjNoAgeNoIatNoGender)
```

We now have a model where removing a variable will not provide an AIC improvement.

We can also consider adding predictors in, one at a time (the scope= formula specifies which predictors to try to add; here we've provided three such predictors):

```
add1(fitNoSubjNoAgeNoIatNoGender, scope=.~.+gender+age+poly(iat, 2))
```

We see that none of them is an improvement.

It is clear however that the three most important effects are due to the reaction time against white targets, whether the target was armed or unarmed, and whether the action was correct or incorrect. Let's review some graphs that break the cases down by these three variables:

```
p <- ggplot(targetingRaceDiff) +
    aes(y=Black) +
    facet_grid(weapon~action)

p + aes(x=White) + geom_point() + geom_smooth()
p + aes(x=White) + geom_point() + geom_lm()
```

Interaction Terms

Before moving on to more complex modeling techniques, let's try adding interaction terms to our model. Our initial model was fitAll , but to keep it simple let's at least remove the age and subject:

```
fitStandard <- fitAll %>% update(formula=.~.-age-subject)
summary(fitStandard)
```

We would like to consider adding some interaction terms now. For example let's add an interaction term that accounts for the effect of weapon status on the white reaction time contribution:

```
fitInter1 <- fitStandard %>% update(formula=.~.+White:weapon)
summary(fitInter1)
anova(fitStandard, fitInter1)
```

From the P-value, we see that this interaction term does not add anything significant. Let's add an interaction term for weapon and action:

```
fitInter2 <- fitStandard %>% update(formula=.~.+action:weapon)
summary(fitInter2)
anova(fitStandard, fitInter2)
```

This appears to be significant! Let's add an interaction term between white reaction time and action:

```
fitInter3 <- fitInter2 %>% update(formula=.~.+White:action)
summary(fitInter3)
anova(fitInter2, fitInter3)
```

That also appears to be significant!

Before we move on, let's use the step method to tell R to perform this step-wise model selection that we performed manually earlier:

```
step(fitInter3)
```

Take a minute to study the result of this call to step.

We see that R successively removed the iat and gender factors from our model, using the AIC numbers as guides. You can look at the documentation for step to learn more about its use.

```
modelFinal <- step(fitInter3)
summary(modelFinal)
```

Mixed Effects Modeling

(This section is a work in progress)

For this section, we return to analyzing targetingFinal .

The above techniques have not accounted at all for the nature of the distinct participants. It is reasonable to assume that each subject has a different baseline reaction time, and we have to account for that effect. The most appropriate way to model that is likely via a *random effect*, i.e. assuming that there is a constant base reaction time component to each subject's reaction time, that comes from a distribution whose parameters we would need to determine. So our model for the reaction time may end up looking something like this:

$$\text{time} = \mu + \beta_1 \times \text{iat} + \beta_2 \times \text{weapon} + \cdots + \text{tsubj} + \epsilon$$

where tsubj is a base reaction time for each subject and is drawn from a distribution $N(0, \sigma_s)$, while ϵ is still the standard error term from a $N(0, \sigma)$ distribution, and β_1, β_2 etc are still the *fixed effects* from the various factors and predictors.

We will start with a simple case. We need a new package for this, called lme4, as mixed effects modeling goes beyond the standard linear modeling techniques. You will need to install this package (using the *Install* menu in the *Packages* pane).

The method lmer from that package is our main workhorse. Here is one example call:

```
library(lme4)
```

Now we will consider a mixed effects model where in addition to our standard fixed terms, it also includes a random intercept addition for each subject. this is indicated by the term (1| subject) in the formula:

```
mixedFit1 <- lmer(time ~ race + weapon + action:weapon +
                    (1|subject), data=targetingFinal)
summary(mixedFit1)

mixedFit2 <- lmer(time ~ iat + race + weapon + action:weapon +
                    (1|subject), data=targetingFinal)
anova(mixedFit1, mixedFit2)

mixedFit3 <- lmer(time ~ iat + weapon + action:weapon +
                    (1|subject), data=targetingFinal)
anova(mixedFit3, mixedFit2)

mixedFit4 <- lmer(time ~ weapon + action:weapon +
                    (1|subject), data=targetingFinal)
anova(mixedFit4, mixedFit3)
```

TODO