Lab 8: More ANOVA Models

October 18, 2018

In this lab, we explore ANOVA designs that are a bit more complicated than a one-way ANOVA. We are going to explore three different designs: factorial design, block design, and repeated measures design. There are many others, but this should get you comfortable with their analysis so that you can learn others by yourself as needed.

The learning goals of the lab are to:

- Recognize different study designs, when and how to use them
- Understand how to adapt your analysis for each design
- Reinforce your ability to conduct ANOVA, *post-hoc* tests, and interpret the results
- Practice writing about the results of your analyses.

At the end of the lab, there are a few problems to answer. *Submit your answers to the class Sakai site under the Assignments folder before midnight on either Mon., Oct. 22 (Section 03) or Wed., Oct. 25 (Sections 01 and 02).*

More functions in R

In this lab, we introduce a few new R commands.

- `aov()` - fits an ANOVA model by a call to `lm` for each stratum
- `with()` - tells a function the data to use, possibly modifying the original data; avoids the need to use `attach()`
- `lm()` - fits a linear model to the data; used for regression (independent variable = continuous) and ANOVA (independent variable = categorical)
- `interaction.plot()` - plots the mean of the response for the two-way combinations of factors, thereby illustrating possible interactions
- `bartlett.test()` - provides a parametric k-sample test of the equality of variances
- `points()` - draws a sequence of points at user-specified coordinates
- `*` - symbol used to define a full first order model (all main effects and interactions): $y \sim \beta_0 + \beta_1 A + \beta_2 B + \beta_3 AB$
- `:` - symbol used to define an interaction in a model
- `mtext()` - writes text in one of the margins of a plot, where `side` specifies which side of the figure to write on and `line` specifies on which margin line to write
- `text()` - writes text within the plotting area of a figure, using x, y coordinates to specify the position
- `axis()` - adds an axis to the plot, allowing for custom axes (usually the axis on the original plot is first suppressed)
- `paste()` - "pastes" together vectors after converting them to characters
- `residuals()` - returns the residuals from a linear model.

Factorial Design

A factorial design has two or more factors, each with two or more levels. The test subjects for a factorial design are assigned to treatment levels of every factor combination at random. This means that we can investigate statistical interactions, in which the response to one factor depends on the level of another factor. Let's use an example of animal diets from Crawley (2005). In this dataset (download `Growth.csv`), the

response variable is weight gain of domestic animals after 6 weeks. There are two factors, diet and supplement, and each level of diet is crossed with each level of supplement making it a factorial design.

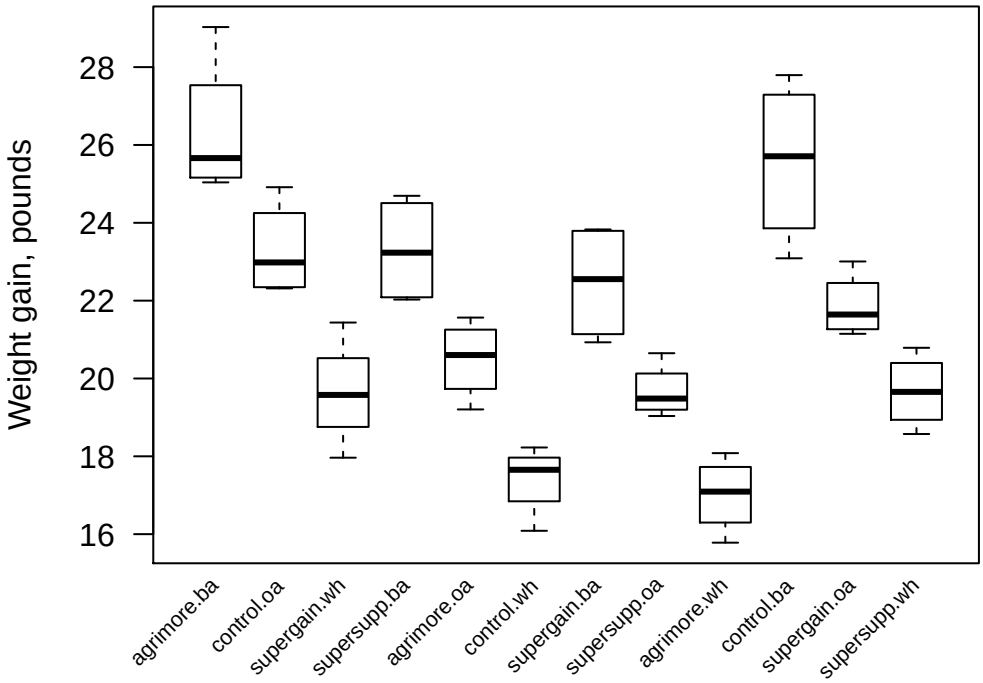
```
feed <- read.csv("Growth.csv", header = T)
```

How many levels are there of each factor? Use `levels()` to check it out. Before we get started, also check to make sure diet and supplement are defined as factors.

Getting a feel for the data

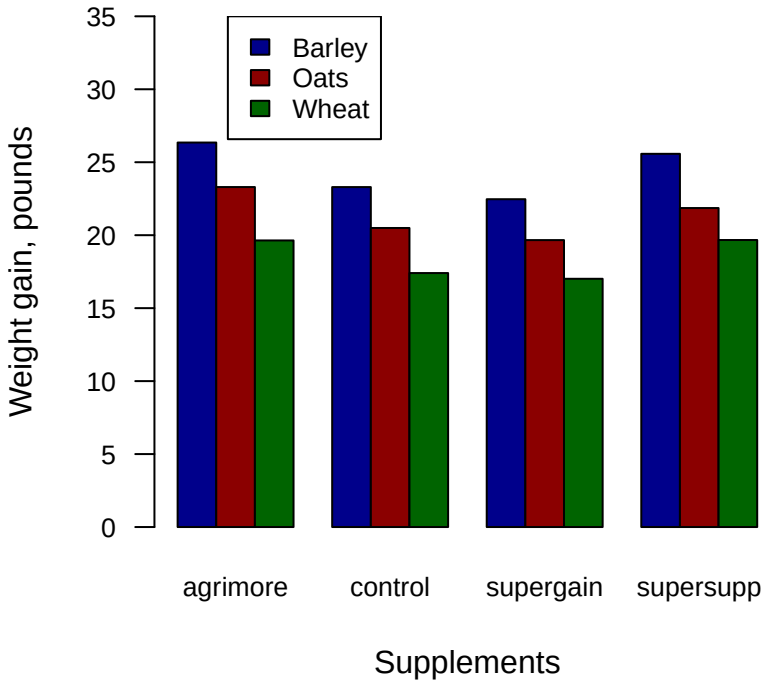
We first want to get a picture of the data. Note that we use the function `with()` to tell R which data to use for the boxplot. This function can be useful when you do not want to attach data permanently to the R search path. In the below code, we also paste together the names of the supplement and diet levels to create labels for the boxplot. In the `substring` line of code, we abbreviate the diet names to just two letters so that we can get everything on the graph. Note the use of `axis`, `text`, and `mtext`.

```
par(mar = c(5, 4, 0.5, 2))
 with(feed, boxplot(gain ~ diet + supplement, las = 1,
                    ylab = c("Weight gain, pounds"), xaxt = "n"))

  axis(1, labels = FALSE, tick = F)
   lab1 <- rep(levels(feed$supplement), 3)
    lab2 <- rep(levels(feed$diet), 4)
   lab2 <- substring(lab2, 1, 2)
  labels <- paste(lab1, lab2, sep = ".")

 text(x =  seq_along(labels), y = par("usr")[3] - 0.5, srt = 45, adj = 1,
      labels = labels, xpd = TRUE, cex = 0.7)
  mtext("Supplements", adj = 0.5, line = -27)
```



We could also view the data as a barplot.

```
cols <- c("darkblue", "darkred", "darkgreen")
with(feed, barplot(tapply(gain, list(diet, supplement), mean),
                   ylim =c(0,35), beside = T, col = cols, las=1,
                   ylab = c("Weight gain, pounds"), cex.axis = 0.8,
                   xlab = c("Supplements"), cex.names = 0.8))
labs <- c("Barley", "Oats", "Wheat")
legend(2.3, 35, labs, fill = cols, cex = 0.8)
```



Running the model

There are a couple ways to run a factorial ANOVA in R, let's start with `aov()`.

```
mod0 <- aov(gain ~ diet*supplement, data = feed)
```

Here the * specifies that we want estimates for the main effects of each level of diet and supplement and the interaction between diet and supplement. We could also write out each of the main effects and the interaction effect. Note that we do this below using two different functions: `aov()` and `lm()`, using `anova()` to get the ANOVA table. There are often multiple ways to do things in R, but the below lines of code should give the same results.

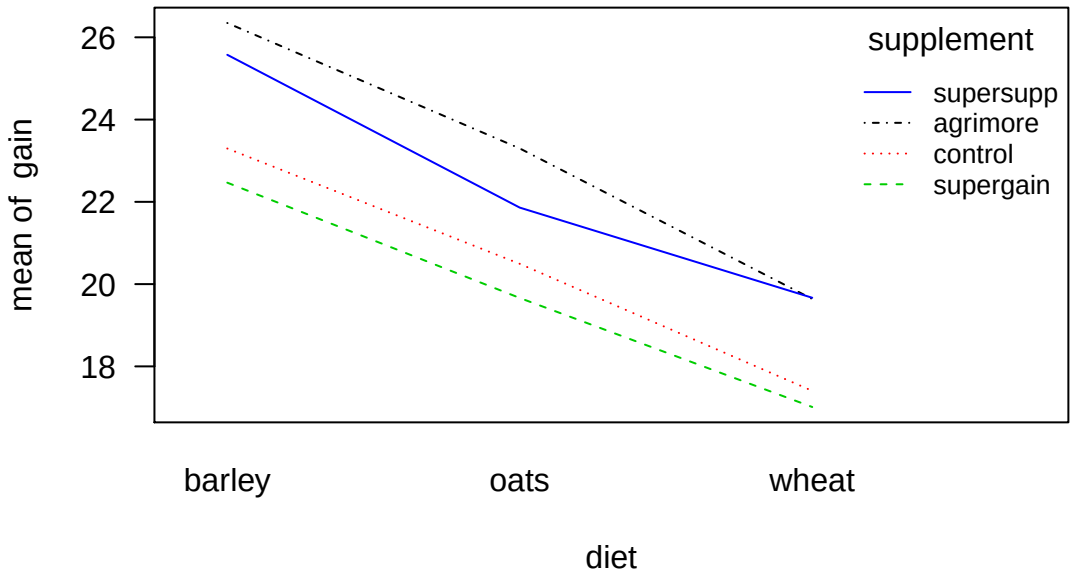
```
mod0 <- aov(gain ~ diet + supplement + diet:supplement, data = feed)
 summary(mod0)

mod0a <- lm(gain ~ diet + supplement + diet:supplement, data = feed)
 anova(mod0a)
```

The results provide no support of a significant interaction between diet and supplement. We can use an interaction plot to get a better idea of what is going on. An interaction plot displays the dependent variable on the y-axis, the levels of one factor on the x-axis, and separate lines for the means of each level of the

second variable.

These types of plots can be used to determine whether an interaction term should be included in our ANOVA model. (The `interaction.plot` function is a bit annoying because it isn't very flexible (e.g., there is no way to change the position of the legend without hacking the code)).

```
with(feed, interaction.plot(diet, supplement, gain, col=c(1,2,3,4), las = 1, cex = 0.9))
```



The consistent decline in weight gain from barley to oats to wheat demonstrates the significance of the main effect of diet. Similarly, the separation of the lines for supplement also suggests a difference along that main effect. The lines are parallel to each other, indicating no interaction.

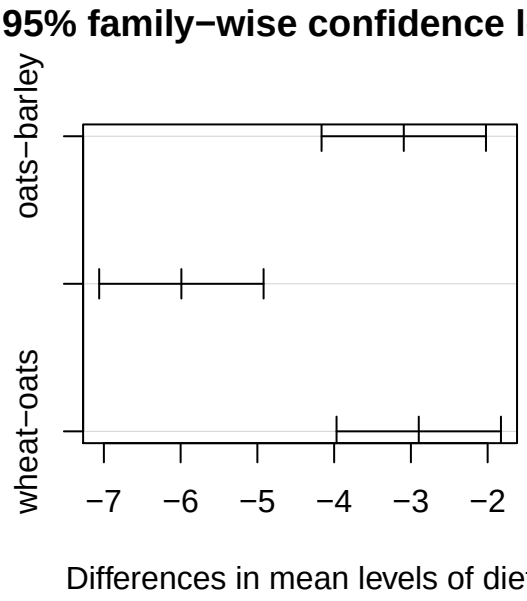
Refining the model

In many cases, the goal of analysis is to look for the simplest (most parsimonious) model for our data. Therefore, we can refine our model taking out non-important, here interpreted as non-significant, variables. Let's simplify our ANOVA model by taking out the interaction term.

```
mod1 <- aov(gain ~ diet + supplement, data = feed)
 summary(mod1)
```

Like the previous model, the ANOVA results tell us that diet and supplement are statistically significant main effects, but where do the differences in weight gain lie?

```
require(graphics)
 TukeyHSD(mod1, "diet", ordered = TRUE)
   plot(TukeyHSD(mod1, "diet"))
```



The Tukey test tells us that there are significant differences between the means of each of the diets. In other words, the mean weight gain is greatest on barley, then oats, and then wheat, with significant differences in mean weight gain between each pair of diets at the 0.05 level.

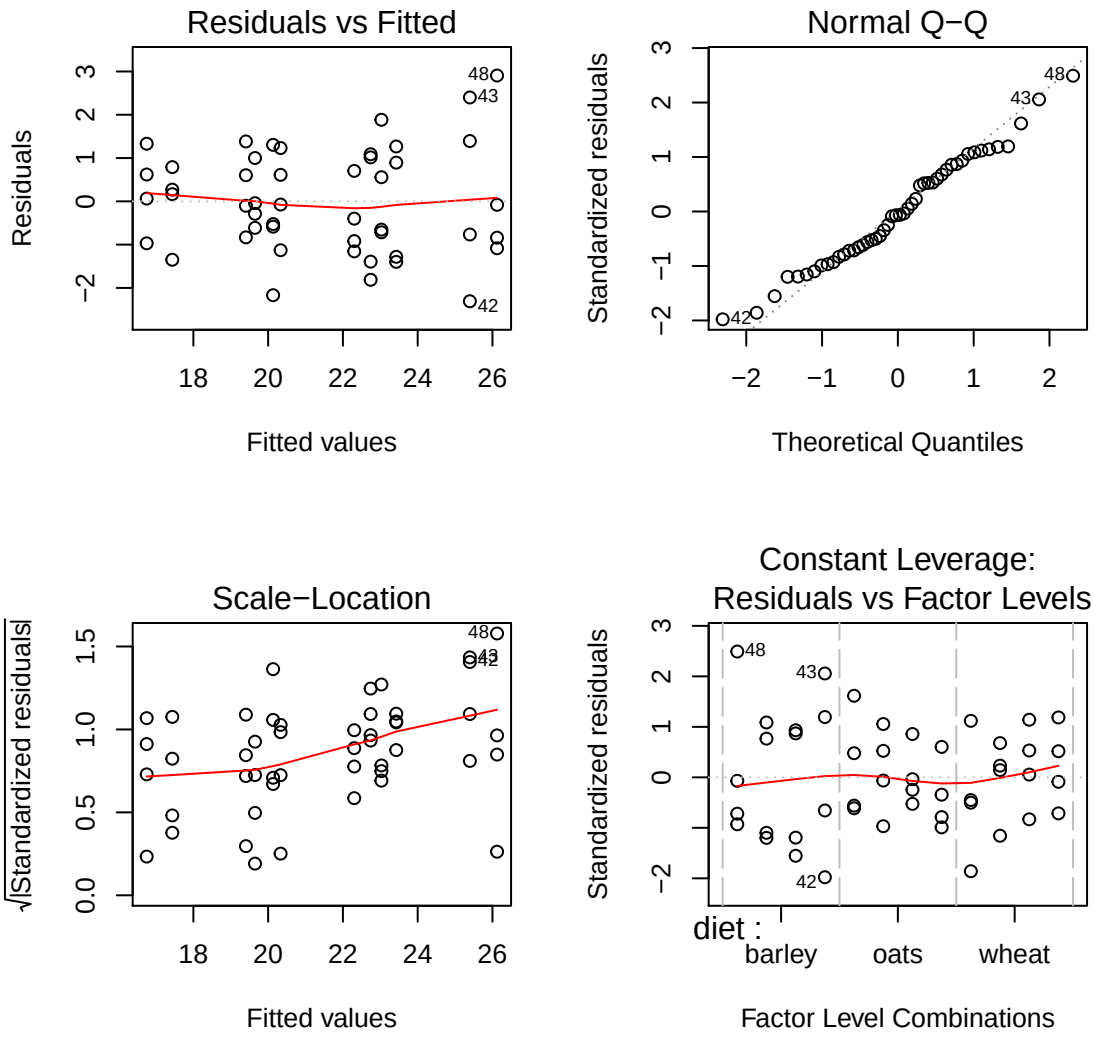
To Do

Modify the above code to examine where the *post-hoc* differences lie for different levels of supplement.

Checking ANOVA assumptions

When conducting any statistical analysis it is important to evaluate how well the model fits the data and that the data meet the assumptions of the model. There are numerous ways to do this and a variety of statistical tests to evaluate deviations from model assumptions. Generally statisticians examine multiple diagnostic plots after running regression models. Here is an introduction to diagnostics, which we will be talking about during our discussion of regression.

```
par(mfrow=c(2,2))
 plot(mod1)
```



Plotting of the model provides four diagnostic plots.

Residual plot

The first plot is plot of the residuals (distance of the data points from the expected values (treatment means)) versus the fitted data. Residuals can be thought of as elements of variation unexplained by the fitted model. We expect them to be (roughly) normal and (approximately) independently distributed with a mean of 0 and constant variance. In the plot, points should be randomly scattered around the centerline. Any pattern indicates either a violation of linearity or homoscedasticity. Note that because we are dealing with ANOVA and categorical independent variables the residuals form swarms around their treatment means. In this plot there might be some evidence of heteroscedasticity because the variance seems to increase with the treatment means. Remedies to this would be to square root or log transform the dependent variable or add a higher order term to the model (e.g. an interaction).

To Do

Log-transform the dependent variable, re-run the model, and evaluate the residual plot. Does the log-transformation help?

```
mod1a <- aov(log(gain) ~ diet + supplement, data = feed)
 summary(mod1a)
```

QQ plot

The second plot is a qq plot, which we have already used to evaluate the normality of data. Significant departures from the line suggest violations of normality. If the pattern were S-shaped or banana shaped, we would need a different model. We could also perform a Shapiro-Wilk test of normality on the residuals with `shapiro.test()` (keeping in mind our caveats towards this test).

```
shapiro.test(residuals(mod1))
```

Or, we could look at the distribution of residuals.

```
hist(residuals(mod1))
```

Standardized residual plot

The third plot is a plot of standardized residuals versus the fitted values. It repeats the first plot, but on a different scale. It shows the square root of the standardized residuals (where all the residuals are positive). If there was a problem, the points would be distributed inside a triangular shape, with the scatter of the residuals increasing as the fitted values increase.

Leverage plot

The fourth plot is a residuals-leverage plot that shows Cook's distance for each of the observed values of the factor levels. Cook's distance measures relative change in the coefficients as each replicate is deleted. So the point is to highlight those y (response) values that have the biggest effect on parameter estimates. The idea is to verify that no single data point is so influential that leaving it out changes the structure of the model. In this plot, there do not seem to be any outliers, although observation 48 has more leverage than other observations.

Like the F-test, `var.test()`, the `bartlett.test()` function provides a parametric test of the equality of variances, but for more samples. The null hypothesis of the test is that the variances are all equal.

```
bartlett.test(gain~diet, data=feed)
```

To Do

Run the Bartlett test for supplement. What do the results of both Bartlett tests seem to indicate?

Block Design

The basic idea behind blocking is to group the experimental units into blocks of similar units and carry out the assignment of treatments separately within each block. With every treatment included at least once in every block, the design is called a complete block design. For example, say we want to compare the effectiveness of three fertilizers for increasing tomato growth. We would apply each of the fertilizers (treatments) to randomly chosen tomato plants in several different gardens (block). We are interested in

which fertilizer has the greatest affect on growth, but take into account variation across gardens because there may be unmeasured factors (soil fertility, exposure to the sun) that led plants in some gardens to have more or less growth than other gardens.

Blocking removes as much variability as possible from the random error so that the differences among the groups are more evident. The focus of the analysis is on the difference among groups (or treatments), not the blocks.

In the completely randomized design (e.g. one-way ANOVA), the total variation (SS_T) is subdivided into variation due to differences among the treatment groups (SS_A) and variation within the groups (SS_E). Within-group variation is considered to be random variation, and among-group variation is due to differences from group-to-group and random variation. To remove the effects of the blocking from the random variation component in the randomized block design, the within-group variation (SS_E) is subdivided into variation due to differences among the blocks (SS_{BL}) and random variation (SS_ϵ).

In this example, we want to test whether four different antibiotics result in different levels of antibodies in the blood. Sixteen people are randomly assigned one of the four antibiotics and samples of their blood are taken for analysis. To process the blood samples as quickly as possible, the samples are sent to four different laboratories – each lab receives one blood sample treated with one of the four antibiotics.

```
lab <- c(rep(1:4, each = 4))
 antibiotic <- rep(c(1:4), 4)
  results <- (c(9.3, 9.4, 9.6, 10, 9.4, 9.3, 9.8, 9.9, 9.2,
                9.4, 9.5, 9.7, 9.7, 9.6, 10, 10.2))
  dlabs <- data.frame(lab = factor(lab),
 antibiotic = factor(antibiotic), results)
```

Each laboratory has its own instruments and personnel that might cause variation in the results across laboratories. Our variable of interest is the level of antibodies in the blood samples; laboratories are blocks whose uncontrolled effects we want to separate from the main effect.

First, let's do the analysis without the block effect. What is your interpretation?

```
mod2 <- aov(results ~ antibiotic, data = dlabs)
```

Now, let's include the block effect. How does this change your interpretation? We could do this using either the `lm()` function or the `aov()` function.

```
mod3 <- lm(results ~ factor(antibiotic) + factor(lab), data = dlabs)
 anova(mod3)

mod3 <- aov(results ~ factor(antibiotic) + factor(lab), data = dlabs)
 summary(mod3)
```

The `TukeyHSD` function we have been using for *post hoc* tests only works for `aov` models. If you want to conduct the *post hoc* test on an `lm` model, first install and load the `mosaic` package. Then you can use `TukeyHSD` for the test.

So, up to now, this should all seem pretty straightforward. `mod3` looks pretty much like a factorial ANOVA without the interaction, and we are treating the effect of lab as a nuisance. If it were only so easy...

Recall that there are two types of ANOVA. Fixed effects ANOVA applies when the treatments have been specifically chosen. For example, when you are interested in the effects of the particular antibiotics above. In other words, you want to know how antibiotics 1, 2, 3 and 4 specifically impact the results.

H_0: There is no difference in level of antibodies among antibiotics 1, 2, 3, and 4

Random effects ANOVA, applies to hypotheses that are more general. Instead of examining the effects of four specific antibiotics, your null hypothesis might be:

H_0: There is no difference in level of antibodies among all antibiotics.

Therefore, the antibiotics chosen are merely representatives of a wider range of antibiotics, even though your random selection might be antibiotics 1, 2, 3, and 4.

If we treat the blocking variable as a fixed effect, then the inference will only apply to those particular blocks (or samples). If we treat the blocking variable as a random effect, then inference can be made to the population of all possible blocks. The second option is what we are after; however, the rule of thumb is that you need at least six subjects (six samples) to estimate a random effect (i.e. variance) or the precision on the estimate cannot be estimated. Note that this also depends upon the assumption that the blocks are chosen randomly from a normal distribution of blocks.

The take-away message is that it is preferable to treat the blocking factor as a random effect, but may not always be possible. Let's see what happens when we set up the model for the Model II ANOVA. Note that the term `Error` below classifies `lab` as a random effect, rather than a fixed effect.

```
mod4 <- aov(results ~ antibiotic + Error(lab), data = dlabs)
 summary(mod4)
```

It looks like we got lucky and the results do not seem to have changed even though we only had four samples (blocks). However, the `TukeyHSD` function does not work on models with random effects. Let's re-run the same model, but using the `lmer` function from the `lme4` package. `lmer` conducts mixed effects models, which include both fixed and random effects and is more generally more useful than `aov`. To get the ANOVA table, wrap `mod5` in `anova`, and to get the rest of the results wrap it in `summary`.

```
require(lme4)
 mod5 <- lmer(results ~ antibiotic + (1|lab), data = dlabs)
  anova(mod5)
   summary(mod5)
```

The other advantage of `lmer` is that it allows us to run *post hoc* tests. Install and load the packages `lmerTest` and `emmeans`. The `glht` function provides general linear hypotheses and multiple comparisons for parametric models. The `linfct` argument specifiesd the linear hypothesis to be tested.

```
 require(emmeans)
  require(lmerTest)
   require(multcomp)
   summary(glht(mod5, linfct = mcp(antibiotic = "Tukey")))
```

Incidentally, the above ANOVA model could be used for situations where you have a fixed effect and a random effect, where the random effect is not necessarily a block, but rather some other nuisance variable.

Repeated Measures Design

A repeated measure ANOVA is used when all members of a random sample are measured under a number of different conditions. As the sample is exposed to each condition in turn, the measurement of the dependent variable is repeated. Using a standard ANOVA would not be appropriate because it fails to model the correlation between the repeated measures: the data violate the assumption of independence.

Let's look at an example of two advertising campaigns in ten different cities. For each city, the sales before, during and after the advertising campaign are measured. We want to know if the campaign has a significant effect on sales.

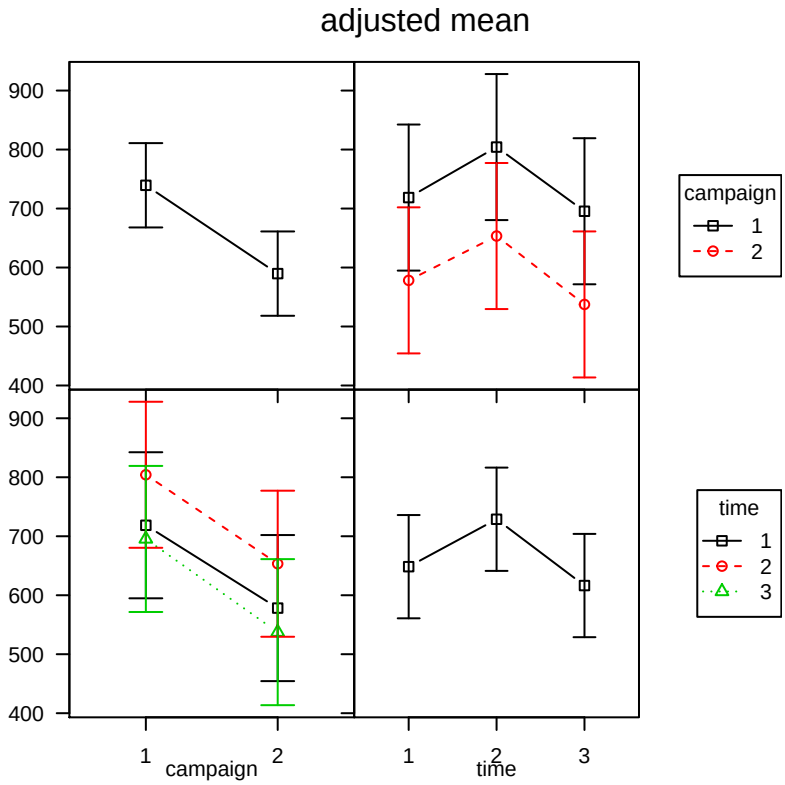
```
ad <- read.csv("Sales.csv", header=T)
 ad <- with(ad, data.frame(sales, city=factor(city),
            campaign = factor(campaign), time = factor(time)))
```

Let's first proceed without including repeated measures. It looks like there is not a significant effect of campaign, time, or their interaction.

```
mod6 <- lm(sales ~ campaign * time, data = ad)
anova(mod6)
```

The package `phia` offers another way to plot interactions. Download it and add it to your working library.

```
require(phia)
 plot(interactionMeans(mod6), las = 1)
  with(ad, tapply(sales, list(time), mean))
 with(ad, tapply(sales, list(campaign), mean))
```



Now let's add in the repeated measures of city (i.e. the sales measurements were taken at the same city before, during, and after the ad campaign). Does this change your interpretation?

```
mod7 <- aov(sales ~ campaign * time + Error(city/campaign), data = ad)
 summary(mod7)

mod8 <- lmer(sales ~ campaign * time + (1|city/campaign), data = ad)
 anova(mod8)
  summary(mod8)
```

Note that we can run this model using `aov`. I produces a warning message, which we can ignore as the model

works fine. The campaign results are listed under the city error (`Error: city`). This is because city is nested within campaign. Because there is not a signficant interaction between time and campaign, we could simplify the model by taking out the interaction.

A better way to run this model is with `lmer`.

```
mod8 <- lmer(sales ~ campaign * time + (1|city/campaign), data = ad)
 anova(mod8)
  summary(mod8)
```

This is written in the more general linear model form. We'll keep going with this model...

We can conclude that there is no interaction between campaign and time ($F_{2,16} = 0.547$, $p = 0.589$) and no main effect of campaign ($F_{1,8} = 0.734$, $p = 0.417$), but there is an effect of time in sales ($F_{2,16} = 93.686$, $p < 0.0001$).

Problems

Analyze the data for the two examples below. Make sure to use the appropriate ANOVA design for each study. For each problem, write a 1-page description of your analysis and the results. Please include your code at the bottom of your write-up in R Markdown. Each write-up should include the following information:

- Null and alternative hypotheses of your tests
- Justification for your choice of ANOVA model
- A description of how you checked the assumptions of your statistical test
- Results of your statistical test, interpreting your test in 2-3 sentences that include the appropriate reporting of the statistics
- An interpretation of any necessary *post-hoc* tests
- An interpretation of diagnostic figures

If you employ a model with a random effect use `lmer()` as the typical post-ANOVA diagnostics (e.g. `plot(mod)`) will not work with `aov`.

Problem 1

Evaluate the effect of salt on plant biomass growth in 24 experimental vegetation plots (download `salt.csv`). The assigned treatment (one of 6 levels of salt addition: 10, 15, 20, 25, 30, 35 g m^{-2}) is applied to the soil of a plot and at the end of the experiment the biomass of plants in each plot is measured. The experimental units are grouped into four blocks of six plots each, based on geographic proximity, and the treatments are assigned completely at random within each block. Thus, each treatment occurs exactly once in each block.

Problem 2

Pangolins are scaly anteaters that inhabit the tropical forests of Asia and Africa and are hunted for their meat and their scales, which are made of hair. A researcher wants to evaluate the effect of diet on the thickness of pangolin scales, with the idea that thicker scales would better protect pangolins from predators. She rears pangolins and provides them with identical diets, but different doses (0.5, 1, 2 milligrams) of supplements (Vitamin B and Zinc). Download the `ScaleThickness.csv` file from Sakai and analyze the data to determine the potential effects of `doses` and `supplements` and whether there is an interaction between the two of them.