Standard Statistical Tests

Basic Standard Statistics: Comparing Groups

We'll use the `frogs` dataset (a larger version of the examp data we used yesterday) to run through the functions R provides for classical statistics tests.

Variance Ratio Test (Fisher's F test)

We can easily compare the variance between two groups using an F-test. This would not be hard to do by 'hand', but there is a built-in function for it called `var.test()`.

```
groupA <- frog$svl[frog$trtmt == "A"] # create vector of treatment A svl
                                                        data

groupB <- frog$svl[frog$trtmt == "B"] # create vector of treatment B svl
                                                        data

var.test(groupA,groupB) # test for difference in variance
```

t-test

There is a function called `t.test()`. It has the form:

`t.test (v1, v2)` where `v1` and `v2` are vectors of data to be compared; y is optional – if you submit only x, than a one-sample t-test is done comparing the mean of the sample to zero (by default) or to a value you specify with the argument `mu=value`. For a paired t-test, use the argument `paired = TRUE`.

An alternative (and perhaps more intuitive) form for the function is:

`t.test(y ~ x)` where `y` is a numerical response variable and `x` is a grouping variable. We can optionally include the argument `data =` to specify a data.frame containing x and y.

Thus, for frogs data, we can test for an effect of treatment (ignoring species effects), using the code:

```
t.test(mass ~ trtmt, data=frogs)
```

If you want a non-parametric Mann-Whitney U test, the code is nearly identical, but using the function `wilcox.test(y ~ x)`

Correlation and Regression

`cor(x,y)` is the function for a simply correlation. Passing a matrix to `cor()` instead of two vectors will give you a matrix of all pairwise correlation coefficients

`lm()` is probably one of the most commonly used functions in R; it stands for linear model and accepts a formula for analyzing continuous response variables.

Given a single x- and y-variable, `lm()` will perform a simple linear regression and output the coefficients *a* and *b*, the intercept and slope of the line of best fit (least-squares estimate).

Using the frogs dataset again, we might perform a regression of mass on svl in the data:

```
mod.lm <- lm(mass ~ svl, data=frogs) # note that we save the results to an
                                        object
summary(mod.lm)  # To get complete output of the regression results,
                  including coefficient estimates and R² value
anova(mod.lm)    # To get the anova table output of the regression
```

ANOVA

The command for doing an ANOVA is `aov()`; `lm()` will also work since the `aov()` function in fact calls `lm()`.

`aov(formula, data)`

formula syntax:

```
y ~ A  # a one-way ANOVA
y ~ A + B # a two-way ANOVA testing main effects only
y ~ A*B # a two-way ANOVA testing main effects and all possible
     interactions
y ~ A + B + A:B # a two-way ANOVA testing main effects A, B and interaction A:B
```

Be Sure to Know what your Analysis is Actually Doing!

Note that `aov()` performs a sequential ANOVA (in SAS terms, a Type I SS); this means that if your data are unbalanced, the order in which you enter the terms matters! The package car will calculate Type II and Type III SS ANOVA tables for model objects produced by `aov()`, `lm()`, glm(), `lmer()` and others.

Let's continue on with analyzing our frogsle data set and analyze the species effect.

```
mod1 <- aov(svl ~ spp + trtmt, data=frogs) # perform anova and save to object
                                            'mod1'
```

First, we need to see if the model is appropriate by looking at the diagnostic plots.

```
plot(mod1) # to get default 4 diagnostic plots

plot(mod1, which = c(1,2)) # get just the residuals & qqplot
```

The car package provides a variety of diagnostic tools, including a qqplot with 95% confidence envelope for the normal line:

`library(car)`

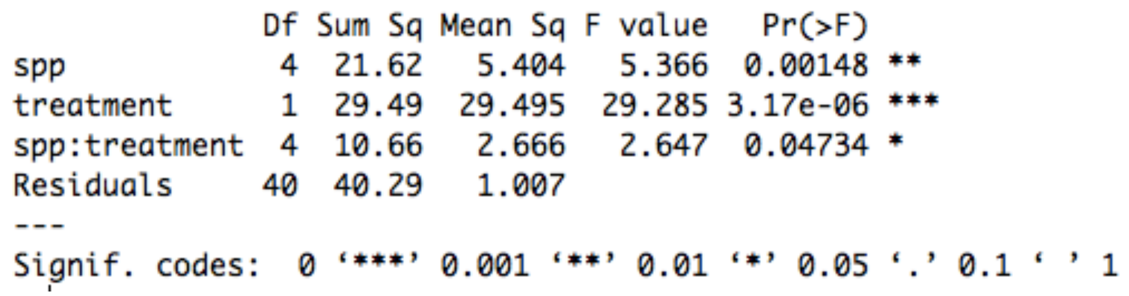
`qqPlot(lm(svl~spp*treatment, data=frogs))` # qqPlot needs output from lm(); you'll be working with lm() in the next session, the model results are identical using aov() and lm().

Another car package function is `outlierTest(mod1)` – this will indicate whether there are any significant outliers in your data.

Given the model seems appropriate, let's now look at the results. The `summary()` function gives us the standard ANOVA table output.

Here's

a screenshot of `summary(mod1)`:



```
              Df Sum Sq Mean Sq F value   Pr(>F)    
spp            4  21.62   5.404   5.366  0.00148 ** 
treatment      1  29.49  29.495  29.285 3.17e-06 ***
spp:treatment  4  10.66   2.666   2.647  0.04734 *  
Residuals     40  40.29   1.007                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we have a significant interaction, we likely want to start interpreting the data by making an interaction plot.

```
with (frogs, interaction.plot(spp, treatment, svl, ylab = "Snout-vent
   Length", xlab = "Species", trace.label = "Treatment"))
```

Seeing these significant results, the next step is to look at how the factors are affecting the data by looking at effect sizes. There are (again) many ways we can do this.

`model.tables(mod1, se=TRUE)` provides the effect sizes for each factor level, as well as standard errors.

`model.tables(mod1, "mean", se=TRUE)` provides the means for each level in each factor.

`plot.design(svl ~ spp:treatment, data=frogs)` provides a plot summarizing effect sizes.

`summary.lm(mod1)` summarizes the model in terms of model coefficients.

The effect sizes given depend on the contrasts option. The default in R is treatment contrasts (contr.treatment) for unordered factors and orthogonal polynomial contrasts (contr.poly) for ordered factors. In treatment contrasts each level of the factor is compared to the first level (i.e. the first level in the factor is used as the reference; not that unless you order your factor levels, this will be the first in alphabetical order). You can change the contrasts using the argument `contrasts=` in the function call, or in your session using the `options(contrasts=)` command.

Multiple comparisons for significant differences among treatment level combinations can be generated with the command: `TukeyHSD(mod1)`.

If you want to do planned contrasts you can use the contrasts() function. You use this function to set the contrast matrices that R should use for a particular factor. You will need to know how to set up planned contrasts by coding the matrix values.

For example, if we wanted to compare toads with the other 4 species (regardless of treatment) we can change the contrast matrix for the factor spp by using:

```
contrasts(frogs$spp) <- cbind("toads vs everything else"= c(4,-1,-1,-1,-1))
```

If we want to compare treatmentA in bullfrog & wood_frog with treatment A in grays_treefrog & spring_peeper we need to first create a factor that represents our interaction.

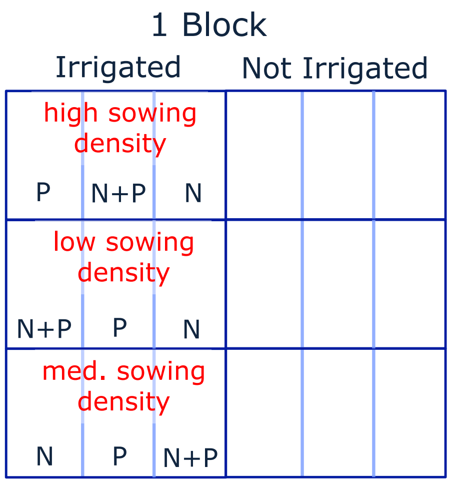
```
frogs$intvar <- frogs$treatment:frogs$spp  # create factor that tests interaction

contrasts(frogs$intvar) <- cbind("A Ranids vs A Hylids" = c(0,1,1,-1,-
1,0,0,0,0,0))  # Now set coding for the interaction factor

mod.cont <- aov(svl ~ spp+treatment+intvar, data=frogs)  # run this model

summary(mod.cont, split = list(intvar = list('A Ranids vs A Hylids' = 1), spp
= list('toads vs. rest' = 1)))  # modify our summary output to include the new
contrasts we've created as components of the appropriate factors
```

```
                               Df Sum Sq Mean Sq F value   Pr(>F)    
spp                             4  21.62   5.404   5.366  0.00148 ** 
  spp: toads vs. rest           1   2.68   2.679   2.660  0.11076    
treatment                       1  29.49  29.495  29.285 3.17e-06 ***
intvar                          4  10.66   2.666   2.647  0.04734 *  
  intvar: A Ranids vs A Hylids  1   0.03   0.027   0.027  0.87010    
Residuals                      40  40.29   1.007                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Nested ANOVA

When our factors are nested within one another, the appropriate error term for the F-ratio changes. Split-plot designs are a common experimental design where fixed factors are nested.

Consider data from on experiment on the effects of irrigation and sowing density on crop yields (this data set comes from Crawley 2007). The study design use four fields (block), each of which was divided in half, with one side irrigated and the other side not. The irrigation plots were then divided into 3 levels of sowing density and finally each of these was divided into 3 levels of fertilizer treatment. An example of the layout is shown below; 4 such blocks were run, with all treatments assigned at random within each level of the next higher factor.



We indicate the nested nature of the data by using the Error term in our formula statement:

```
mod.yields <- aov(yield ~
    irrigation*density*fertilizer + Error(block/irrigation/density),
    data=yields)
```

```
summary(mod.yields)
```

Look carefully at the output and you will notice that there are 4 ANOVA tables, starting with the largest level, block (there is no test for block because we have no replication at this level). The irrigation factor is next and it is tested using block:irrigation as the error term; density and irrigation:density effects are both tested using error = block:irrigation:density. Finally, fertilizer (and associated interaction) effects are tested using the within factor error term.

Multivariate Analysis of Variance

If we have multiple response variables we need to create a matrix of the response variables and use that matrix in the formula we send to aov(). (Below I use the trick as before of creating a matrix of y variables right inside the formula.) Using the function `manova()` provides us the appropriate summary method.

We'll try this out using the frogs data set again. Recall that those data had measures of mass and svl (snout-vent length) for each individual. Since we have measured both mass and svl on the same individuals (hypothetically), we clearly cannot consider them independent and it would not be appropriate to test the two responses as independent.

```
mult.mod <- manova(cbind(frogs$svl, frogs$mass) ~ spp*treatment, data=frogs)
```

```
summary(mult.mod)
```

`summary.aov(mult.mod)` #this will give you univariate tests

`library(car)` # the function `Anova()` in the car package provides more information from the analysis

```
summary(Anova(mult.mod))
```

The consistent results of all four estimators of significant effects in the MANOVA are encouraging in suggesting that the model result is robust, we should nonetheless look at some diagnostics.

Residual plot: to do this, we pull out the residuals and fitted values from mult.mod and plot them (the plot() function for a MANOVA model does not yet exist).

```
plot(y=mult.mod$resid, x=mult.mod$fitted)
```

Multivariate normality: the package mvnormtest provides an extension of the Shapiro test to assess multivariate normality

```
library("mvnormtest")

ymat <- t(as.matrix(cbind(frogs$svl, frogs$mass))) # t() is transpose

mshapiro.test(ymat)
```

Given the assumptions the data violate, we can consider using the function adonis() in the vegan package to do a robust MANOVA. This function uses permutation tests and thus does not depend on the data matching a multivariate normal distribution.

```
library("vegan")

mod.rob <- adonis(cbind(frogs$svl, frogs$mass) ~ spp*treatment, data=frogs)
            # adonis() requires the response variables be in matrix format

mod.rob

mod.rob$model.matrix

mod.rob$coefficients
```

Generalized Linear Models

These are the models we need if we have categorical (or otherwise nowhere near normally distributed) response variables. This occurs commonly with biological data, where the response is often in the form of presence/absence, alive/dead, brood size, etc.

To see how these models work, we will return to the turtle data we've been using. We'd like to model nest predation as a function of the season in which the nest is laid. One would imagine that this might also vary with year, so we will include year in the model also.

Logistic Regression

Nest depredation is binary data; nests were scored either as depredated (1) or not (0). This type of data can be modelled with a logistic regression, transforming the response using logit(y) and a binomial error distribution.

In R, generalized linear models are run with the function glm(). [Current or former SAS users beware! In SAS, PROC GLM is a general linear model (the equivalent of R's lm) and the switched notation is bound to drive you nuts for a while.] Here's the syntax for the turtle model described above:

```
glm(predation ~ season*year, family=”binomial”, data=turtles)

summary(mod.full)

anova(mod.full)

anova(mod.full, test='Chisq')
```

Binomial Response Variables

For the binomial model you have several options for the form of your response (y) variable. It can be a numerical variable of zeroes and ones (as we have here). You can also use a factor with two levels (success, failure). Finally it can be a matrix of number of successes and total number.

The appropriate way to test whether a particular variable included in the model is significant is to run a series of nested models. In this way you can remove one factor at a time and then compare the models to determine whether model fit is significantly improved with the additional factor. If it is now, we conclude that that factor was not a significant term in the model. We continue to simplify our model in this way until we find that removing a term does generate a model significantly different from the previous model. Significant difference in nested model can be assessed using the anova() function.

```
mod1 <- glm(predation~ season + year, family=binomial, data=turtles)

anova(mod1,mod.full, test='Chi')


mod2 <- glm(predation~ year, family=binomial, data=turtles)

anova(mod2, mod1, test='Chisq')
```

```
mod.min <- glm(predation ~ 1, family=binomial, data=turtles)
anova(mod.min, mod2, test='Chisq')
```

<u>Poisson Regression</u>

When we have count data, we use the Poisson distribution and the log link to model data. This is done identically to the logistic regression – we need only change the family option.

```
pr.mod <- glm(clutch.size ~ year*season*sw.veg, family = poisson,
data=turtles)
summary(pr.mod)
anova(pr.mod, test='Chisq')
```

```
pr.mod1 <- glm(clutch.size ~ year*season, family=poisson, data=turtles)
anova(pr.mod1,pr.mod, test='Chisq')
```

<u>Generalized Linear Mixed Models</u>

If we have a non-normal response variable and both random and fixed independent factors then we use a generalized linear mixed model. As with general linear mixed models, these are fit in R with the function `lmer()` from the package lme4.

Longitudinal Data (Repeated Measures) Example

(from Douglas Bates, author of package lme4)

```
library(lme4)
library(lattice)
```

## plot data (xyplot requires package lattice – should be already loaded as part of base R)

```
print(xyplot(Reaction ~ Days | Subject, sleepstudy, aspect = "xy", layout =
  c(6,3), type = c("g", "p", "r"), index.cond = function(x,y) coef(lm(y ~
  x))[1], xlab = "Days of sleep deprivation", ylab = "Average reaction time
  (ms)"))
xtabs(~ Subject + Days, sleepstudy) # check balance of design
```

Use lmer to estimate model that has days as fixed effect and subject as random effect

```
(fit1 <- lmer(Reaction ~ 1 + Days + (1 + Days|Subject), sleepstudy, REML =
0)) # this is the full model with intercept and slope estimate for each
subject; 'Days | Subject' indicates that days are repeated measures
```

We might now ask, is it true that the change in response time with sleep deprivation differs among individuals in slope, or is it that the change is similar among all subjects and it is only that subjects differ in intercept (their starting reaction time). To do this we want to fit a model that does not allow slope to vary among subjects. Non-intuitively we do this by adding subject as random effect on its own to estimate subject intercepts and then adding 0 to our 'Days | Subject' no intercept is estimated here (we cannot estimate the intercept twice!)

```
(fit2 <- lmer(Reaction ~ 1 + Days + (1|Subject) + (0+Days|Subject),
sleepstudy, REML = 0))


anova(fit2, fit1)
```