Transition to R Classes 4 and 5: Basic Statistics – Regression, ANOVA, t-test

Tools for basic statistics in the base R package.

Goals:

1. Fitting Models
   1A. Model objects and Extractor functions p2
   1B. attach() and detach() p3
   1C. How to write formulas for models in R p4
2. Linear regression using lm() p5
   2A. Simple Linear Regression p5
   2B. Accessing individual named components of lm model objects p7
   2C. Using predict() to get fitted lines and confidence intervals p9
   2D. Multiple Linear Regression and Stepwise Model Selection p10
3. ANOVA using aov() or lm() p12
   3A. Independent 2-Sample t-test, test of equal variance, Wilcoxon Rank test p12
   3B. Paired sample and one-sample t-tests p13
   3C. Simple one-way ANOVA using lm and aov p14
   3D. oneway.test, and Kruskal-Wallis anova packages p18
   3E. Factorial design ANOVA p19
   3F. Blocked or split-plot design ANOVA p20
   3G. Nested ANOVA p20
   3H. ANCOVA p21
4. Testing for homogeneity of variance among groups p22
5. Type I and Type III sums of squares p22
6. Summary cheat sheet p23

Sample data sets:
The examples here draw on five data frames (rd, three, two, fb, nes), available from the course web page by running the following code in R.

#run this block of code to get all five data frames
rd<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/RegressionDataset.csv",sep="",header=TRUE)
three<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/ThreeTreatmentDataset.csv",sep="",header=TRUE)
two<-three[-which(three$treatment=="potassium"),]
fb<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/factorialdata.csv",sep="",header=TRUE)
nes<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/NestedAOVdata.csv",sep="",header=TRUE)
#end of code to get data frames
(1) **Model Objects and Extractor Functions:** When you run an analytical function in R, you usually get back a very sparse result, not enough for what you need. That is because the output of analytical functions like linear regression (lm) and analysis of variance (aov), are *model objects*, usually of type List and most of the model object are hidden from view. These model objects often have a large number of important embedded components that you can access using a variety of *extractor functions* (e.g., summary(), anova()) that package the information into useful structures and present the results.

*Always assign the output of a model you run to an object and then extract it; don't just run the model.*

*Do this:*  
\[ \text{lmut} <- \text{lm(rd$temp ~ rd$precip)} \]  
*Not this:*  
\[ \text{lm(rd$temp ~ rd$precip)} \]

To see a "table of contents" of the model object, use the `str()` function.  
\[ \text{str(lmut)} \]  
This shows that `lmut` is a list with 12 elements

- You can access some components of the model object by name (e.g., `coefficients(lmut)`)  
- You can use the `$` to access named components (e.g., `lmut$coefficients`)  
- You can call elements by the position in the list; e.g., since fitted.values are the 5th `$` element in the model object list, so you can call  
\[ \text{lmut[[5]]} \]  
#the `[[ ]]` is used as position indicator in lists

Some elements of the list have sub-elements; e.g., "lmut$qr" (element 7) has five $ sub-elements, the third of which is "pivot". There are several ways to see the values. Each of the following gives the same result:  
\[ \text{lmut$qr[3]} \quad \text{lmut[[7]][3]} \quad \text{lmut[[7]]$pivot} \quad \text{lmut$qr$pivot} \]

Doing statistics in R is a matter of knowing which analytical functions are appropriate, what the structure of the resulting model object is, and which extractor functions and names are useful for getting what you want.

**Generic extractor functions**

There are a number of generic extractor functions that package the model output in useful ways.

<table>
<thead>
<tr>
<th>Generic extractor</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>summary()</code></td>
<td>shows parameter estimates, and statistical values such as F, $R^2$, P, df</td>
</tr>
<tr>
<td><code>anova()</code></td>
<td>gives the ANOVA table with SS, MS, F, P, df</td>
</tr>
</tbody>
</table>
| `plot()`          | produces 4 diagnostic plots: Residual vs. Fitted; Normal QQ; Scale Location; Residual vs. leverage. Hit return in console to produce each in turn; scroll through with command-arrows. Or use this to see all at once:  
\[ \text{par(mfrow=c(2,2)); plot(myModel); par(mfrow=c(1,1))} \] |
| `coef()`          | shows the estimated parameters from the fit model |
| `fitted()`        | shows the fitted valued as predicted by the model for the independent variables included in the model |
| `resid()`         | shows the residuals; measured minus predicted values of the dependent variable |
| `predict()`       | produces a smooth function based on the fitted model to plot |
| `methods(class=function)` | shows all the extractor functions available for a function (usually) |
1B. A note on attach() and detach()

Attach(): a useful and dangerous friend to avoid typing the data frame name repeatedly
If you are only going to be working with one data frame for a while, and you don't want to have to type the dataframe$column1 each time, and instead just call column1, you can use the function attach.

Compare:
```r
rdout<-lm(rd$temp~rd$precip) #specify the data frame for each variable
plot(rd$temp~rd$precip)
```

with:
```r
attach(rd) #attach the data frame
rdout<-lm(temp~precip) #call directly to the variables
plot(temp~precip)
detach(rd) #detach the data frame to work on a different data frame
```

Be careful with attach()! If you have a variables called "temp" and "precip" in data frame "rd" (rd$temp) and attach the data frame "rd", you then just call lm(temp~precip). BUT, if you happen to have another object in your workspace called simply "temp", things can get ugly very quickly. Use attach with caution, and be sure to detach as soon as appropriate.
1C. How to write formulas for models in R

Models in R take the form: response variable ~ explanatory variables

R formulas have several special symbols:

* interaction, e.g., A*B is A, B, and the interactions between A and B
^ interaction, e.g., (A+B)^2 is A, B, and the interactions between A and B
: interaction, e.g., A:B is the interaction between A and B (compare to *)
/ nested, e.g., A/B is A + B nested in A, note that left to right is largest to smallest plot
I() The "as is" function. Because *, /, and ^ are both operators (multiply, divide, to the power) AND they mean something different in formulas (interaction, nested, interaction), the I() allows what is inside the () to be treated as an operator.  y~x + I(x^2) + I(1/z) says to fit y = x + x^2 + 1/z

Error() Allows specification of error terms to use in models when there are multiple error terms, such as in split-plot designs
- remove this factor from the model

If:
y is a continuous dependent variable
x and z are continuous independent variables
A, B, and C are categorical factors

<table>
<thead>
<tr>
<th>Model formulation</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>y ~ 1</td>
<td>#the null model of intercept only</td>
</tr>
<tr>
<td>y ~ x</td>
<td># y is a function of x</td>
</tr>
<tr>
<td>y ~ x+z</td>
<td>#multiple regression with two independent variables</td>
</tr>
<tr>
<td>y ~ x*z</td>
<td>#multiple regression with interaction as y ~ x+z+x:z</td>
</tr>
<tr>
<td>y ~ (x+z)^2</td>
<td># multiple regression with interaction as y ~ x+z+x:z</td>
</tr>
<tr>
<td>y ~ x + I(x^2) + z</td>
<td># fits the model y = x + x^2 + z</td>
</tr>
<tr>
<td>y ~ poly(x,2) + z</td>
<td># fits the model y = x + x^2 + z</td>
</tr>
<tr>
<td>y ~ x-1</td>
<td>#y as a function of x with no intercept (force through zero)</td>
</tr>
<tr>
<td>log(y) ~ I(1/x) + sqrt(z)</td>
<td>#fits transformed model  ln(y) = 1/x + √z</td>
</tr>
<tr>
<td>y ~ A</td>
<td>#one-way ANOVA</td>
</tr>
<tr>
<td>y ~ A+B</td>
<td>#two-way ANOVA</td>
</tr>
<tr>
<td>y ~ A*B</td>
<td>#2-way factorial ANOVA</td>
</tr>
<tr>
<td>y ~ A+B+A:B</td>
<td>#explicit form of 2-way factorial ANOVA</td>
</tr>
<tr>
<td>y ~ A<em>B</em>C-A:B:C</td>
<td>#3-way factorial ANOVA, but do not fit the 3-way interaction term</td>
</tr>
<tr>
<td>y ~ x+A</td>
<td># analysis of covariance, with one slope, two intercepts (covariate first)</td>
</tr>
<tr>
<td>y ~ x*A</td>
<td># analysis of covariance, with two slopes and two intercepts</td>
</tr>
<tr>
<td>y ~ A/B/C</td>
<td>#Factor C nested in B nested in A, left to right is largest to smallest</td>
</tr>
<tr>
<td>y ~ A + B %in% A</td>
<td>#A plus B nested in A, the equivalent of y ~ A/B</td>
</tr>
<tr>
<td>y ~ A<em>B</em>C+Error(A/B/C)</td>
<td>#Split-plot factorial with different error variances for each of 3 plot sizes</td>
</tr>
</tbody>
</table>

The update function. update() allows you to test reduced variations of a full model without re-writing all the terms each time. The term "~." (tilde period) means "the model as it is."

FullModel <- lm(y~A*B)  # fit the full model of y = A + B + A:B, and save as FullModel
NoIntModel<- update(FullModel,~. -A:B) #based on FullModel, fit the reduced model y = A + B.
(2) Linear regression using \texttt{lm()}

2A. Simple linear regression

Data frame \texttt{rd} includes data on mean temperature, mean precipitation, and the number of species (temp, precip, num\_spp) in 30 plots.

Use this code to get it from the course website

\begin{verbatim}
rd<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/RegressionDataset.csv",sep="",header=TRUE)
\end{verbatim}

\begin{verbatim}
head(rd,4)
temp   precip num\_spp
1   15    7.55427       5
2    8   18.25095       3
3    6  27.53882       1
4   16  48.95995      11
\end{verbatim}

Here are three ways to do the same thing: fit a simple linear regression of num\_spp on precip; (that is, num\_spp = B0 + B1(precip)) and save the model object to \texttt{slrout}

\begin{enumerate}
\item specify the data frame for each variable

\begin{verbatim}
slrout<-lm(rd$num\_spp ~ rd$precip)
\end{verbatim}

\item specify the data frame once, then just call variables with simple names

\begin{verbatim}
slrout<-lm(data=rd, num\_spp ~ precip)
\end{verbatim}

\item attach the data frame, then call the variables by simple names

\begin{verbatim}
attach(rd)
slrout<-lm(num\_spp ~ precip)
detach(rd)  #do not forget to detach when you are done!
\end{verbatim}
\end{enumerate}

Remember, you have now created a model object called "slrout" with lots of component elements. You have to use extractor functions to see and use the different elements.

\begin{verbatim}
str(slrout)  #shows the table of contents of the model object slrout
\end{verbatim}

In abbreviated form, the main components are:

\begin{verbatim}
$ coefficients : Named num [1:2] 4.8781 0.0964
$ residuals    : Named num [1:30] -0.607 -3.638 -6.534 1.401 2.496 ... 
$ effects      : Named num [1:30] -109 46.09 -5.81 2 3.04 ...
$ rank         : int 2
$ fitted.values: Named num [1:30] 5.61 6.64 7.53 9.6 10.5 ... 
$ assign       : int [1:2] 0 1
$ qr           :List of 5 (qr, qraux, pivot, tol, rank)
$ df.residual  : int 28
$ xlevels      : list()
$ call         : language lm(formula = num\_spp ~ precip)
$ terms        :Classes 'terms', 'formula' length 3 num\_spp ~ precip
$ model        :'data.frame': 30 obs. of 2 variables:
    ..$ num\_spp: int [1:30] 5 3 11 13 14 15 6 19 ...
    ..$ precip : num [1:30] 7.55 18.25 27.54 48.96 58.35 ... 
\end{verbatim}
The following six lines will show you the most of what you need for simple linear regression.
slrout  #shows the model
summary(slrout) #this shows parameter estimates and errors, plus significance
anova(slrout) #gives the ANOVA table
par(mfrow=c(2,2)); plot(slrout); par(mfrow=c(1,1)) #4 diagnostic plots on 1 page
plot(rd$num_spp~rd$precip) #plot of the original data
abline(slrout)  #takes the fitted model, and draws that line through the data

HERE IS THE OUTPUT

> slrout  #shows the model
Call:
  lm(formula = num_spp ~ precip)
Coefficients:
(Intercept)       precip
     4.8781      0.09643

> summary(slrout) #this shows parameter estimates and errors, plus significance
Call:
  lm(formula = num_spp ~ precip)
Residuals:
     Min      1Q  Median      3Q     Max
-8.7030 -4.4400  1.7850  3.1000  5.5530
Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             4.8781     1.7383  2.8060  0.00902 **
precip                   0.0964     0.0097  9.9050 1.19e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.653 on 28 degrees of freedom
Multiple R-squared: 0.7783,  Adjusted R-squared:  0.7691
F-statistic: 98.11 on 1 and 28 DF,  p-value: 1.187e-10

> anova(slrout) #gives the ANOVA table
Analysis of Variance Table

Response: num_spp
          Df Sum Sq Mean Sq  F value    Pr(F)
precip     1 2124.42 2124.42  98.113    1.19e-10 ***
Residuals 28  606.28   21.65
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> par(mfrow=c(2,2)); plot(slrout); par(mfrow=c(1,1)) #diagnostic plots
> #note that the par statements just put the four plots on a single page

See next page for the graphs that appear in the quartz window (command-3)
Use command-left arrow to scroll back through graphs
Use command-right arrow to scroll forward through graphs
Figure 1. Number of species increases with greater mean precipitation.
2B. Accessing individual named components of \textit{lm} model objects

\begin{verbatim}
str(slrout) \#see the TOC of the 12 components embedded in the \textit{lm} model object
In abbreviated form, the main components are:
$ coefficients : Named num [1:2] 4.8781 0.0964
$ residuals   : Named num [1:30] -0.607 -3.638 -6.534 1.401 2.496 ...
$ effects     : Named num [1:30] -109 46.09 -5.81 2 3.04 ...
$ rank        : int 2
$ fitted.values: Named num [1:30] 5.61 6.64 7.53 9.6 10.5 ...
$ assign      : int [1:2] 0 1
$ qr          : List of 5 (qr, graux, pivot, tol, rank)
$ df.residual : int 28
$ xlevels     : list()
$ call        : language lm(formula = num_spp ~ precip)
$ terms       : Classes 'terms', 'formula' length 3 num_spp ~ precip
$ model       : 'data.frame': 30 obs. of 2 variables:
..$ num_spp: int [1:30] 5 3 1 11 13 13 14 15 6 19 ...
..$ precip : num [1:30] 7.55 18.25 27.54 48.96 58.35 ...

methods(class=lm) \#shows the available extractor functions
[1] add1.lm*  alias.lm*  anova.lm*  case.names.lm*
[5] confint.lm* cooks.distance.lm* deviance.lm* dfbeta.lm*
[9] dfbetas.lm* dummy.coef.lm* effects.lm*
[13] extractAIC.lm* family.lm* formula.lm* hatvalues.lm
[17] influence.lm* kappa.lm* labels.lm* logLik.lm*
[21] model.frame.lm model.matrix.lm plot.lm* predict.lm
[25] print.lm   proj.lm*  residuals.lm rstandard.lm
[29] rstudent.lm simulate.lm* summary.lm variable.names.lm*
[33] vcov.lm*
Non-visible functions are asterisked \#THAT MEANS THEY DON’T RETURN ANYTHING

You can extract individual elements by calling these methods, or by addressing the elements of the model object list, depending on what you need. See page 2.

Try these, as examples:
formula(slrout)
effectAIC(slrout)
fitted(slrout)
coefficients(slrout)
slrout$coefficients
slrout$[1]
slrout$model
slrout$call
slrout$df

You can extract data, statistics, and descriptors from the model to use in other analyses, to adorn graphics, or to create tables of values. For example, try this:
plot(rd$num_spp~rd$precip)
abline(slrout)
text(150,5,paste("num_spp=" , round(slrout$coefficients["(Intercept)"],3) , "+",
round(slrout$coefficients["precip"],3),"precip"),pos=4)
text(150,3,expression(R[adj]^2~"=",pos=4)
text(150,3,paste("          ",round(summary(slrout)$r.squared,3)),pos=4)
\end{verbatim}
2C. Using `predict()` to get fitted lines and confidence intervals

The model object produced by `lm()` contains all the information to generate fitted values and confidence intervals for a line. The function `predict()` gives you those values for each of the values of the independent variable used in creating the original model. `predict()` takes the general form:

\[
predict(modelObject, interval = c("none", "confidence", "prediction"), level = 0.95)
\]

Interval can take one of three values:
- "none" or blank: the predicted line itself
- "confidence" or "c": CI that reflect uncertainty around the line itself; if level=0.95, traditional 95% CI
- "prediction" or "p": PI reflect uncertainty about future observations

`predict()` takes the model object from one set of data, and applies it across a desired range of x values in a separate data frame, but with the same variable name as used in the original model. The output includes three columns: the model fit, the lower CI, and the upper CI.

By default, `predict()` uses the values for the dependent variable used to fit the original model to generate the fitted data or confidence intervals. If you want to predict over a different range of dependent variables, you can specify `newdata`. This takes the general form:

\[
predict(modelObject, newdata=data.frame(originalDepvarName=yourNewValues))
\]

#start with the linear regression of number of species on precipitation, as in 1A.

```r
slrout<-lm(num_spp~precip, data=rd) #produces an lm model object called slrout
predfit<-predict(slrout) #generates the fitted values for each of the original x values from precip
#look at output where open are original data and closed are predicted
plot(num_spp~precip, data=rd); points(predfit~rd$precip,pch=19)
```

#If you want to fit data over a particular range

```r
predfit2<-predict(slrout,newdata=data.frame(precip=seq(50,150,10)))
#uses predict to generate fitted values over the range of 50 to 150 by step 10
#look at output where open are original data and closed are predicted
plot(num_spp~precip, data=rd); points(predfit2~seq(50,150,10),pch=19)
or showing predicted values as a line
plot(num_spp~precip, data=rd); points(predfit2~seq(50,150,10),type="l")
```

#You can also use `predict` to give confidence intervals

```r
predCI95<-predict(slrout,interval="confidence", level=0.95) #95% Confidence Int.
predPI95<-predict(slrout,interval="p", level=0.95) #95% Prediction Int.
```

#Confidence interval is where mean of future observations are most likely to occur
#Prediction interval is where future observations are most likely to occur.

#Put the fitted line and both kinds of confidence intervals on one graph

```r
plot(rd$precip,rd$num_spp)
lines(predfit2~seq(50,150,10),lwd=5, lty=2) #fitted line over particular range
matlines(rd$precip, predCI95, lty=2,col="black") #draws confidence bands
matlines(rd$precip, predPI95, lty=3,col="blue") #draws prediction bands
matlines(rd$precip, predfit, lty=1, col="red", lwd=1) #draws regression line
```
2D. Multiple Linear Regression and Stepwise Model Selection
Use data frame "rd" from 2A, check if precip and temp are both predictive of number of species. First, take a look at pairwise correlations among the three variables.

```r
pairs(rd,panel=panel.smooth)  #graphs
cor(rd)  #correlation matrix
```

```
temp  precip  num_spp
temp 1.0000000 0.1177029 0.1902117
precip 0.1177029 1.0000000 0.8820303
num_spp 0.1902117 0.8820303 1.0000000
```

Since there is no strong correlation between temp and precip, you could go ahead and include both on the right side of the lm model statement

```r
mlrout<- lm(num_spp~precip+temp,data=rd)
summary(mlrout)
```

```
Call:  lm(formula = num_spp ~ precip + temp, data = rd)

Residuals:     Min       1Q   Median     3Q      Max

Coefficients:             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.567401   2.937668   0.874    0.390
precip      0.095302   0.009812  9.713 2.64e-10  # precip is important
temp        0.166867   0.170939   0.976    0.338  # temp is not important

Residual standard error: 4.657 on 27 degrees of freedom
Multiple R-squared: 0.7855,   Adjusted R-squared: 0.7697
F-statistic: 49.45 on 2 and 27 DF,  p-value: 9.397e-10
```

You can use AIC (Akaike Information Criterion) to compare different models. First, by hand:

```r
AIC(lm(num_spp~precip+temp+precip*temp,data=rd))  #full model with interaction
[1] 183.6184
AIC(lm(num_spp~precip+temp,data=rd))  #no interaction term
[1] 182.2800
AIC(update(mlrout,~. -temp))  #precipitation only, using the update function
[1] 181.3205
AIC(lm(num_spp~temp,data=rd))  #temperature only
[1] 225.3643
```

*Rule of thumb:* $\Delta$AIC (model i– best model) < 2 suggests substantial support for the reduced model; $\Delta$AIC between 3 and 7 suggests much less support, $\Delta$AIC suggests little support for the model. Removing the interaction term or temp and the interaction term has a small, negative $\Delta$AIC (181.3-183.6) suggesting that the reduced model (num_spp~precip) has a lot of support. Removing precip instead (leaving num_spp~temp) has a huge $\Delta$AIC =225.4-183.6, suggesting there is no support for that model. Together, this suggests that temp contributes little, and the best model is numspp~precip.
R has an automated **stepwise selection function `step`**, that uses AIC scores to do stepwise selection of model terms, beginning with the full model. Automatically keeps the best model, given its cutoffs.

```r
sd<-step(lm(num_spp~precip*temp,data=rd),direction="backward", trace=1)
summary(sd)
```

```
> sd<-step(lm(num_spp~precip*temp,data=rd),direction="backward", trace=1)

Start:  AIC=96.48
num_spp ~ precip * temp

              Df Sum of Sq    RSS    AIC
precip:temp  1    12.773 585.61 95.144   #removes the interaction term
<none>                     572.84 96.482

Step:  AIC=95.14
num_spp ~ precip + temp

              Df Sum of Sq     RSS     AIC
temp          1     20.67  606.28  94.184   #removes the temperature term
<none>                 585.61  95.144   #AIC with precip + temp in model
precip        1  2046.29 2631.90 138.228   #removes the precipitation term

Step:  AIC=94.18
num_spp ~ precip

              Df Sum of Sq     RSS     AIC
<none>                 606.28  94.184   #AIC with just precipitation term
precip        1  2124.4  2730.70 137.333   #AIC with intercept only

> summary(sd)   #gives summary of best model

Call:
lm(formula = num_spp ~ precip, data = rd)   #structure of the best model

Residuals:
     Min      1Q  Median       3Q      Max
-8.703  -4.449  1.785  3.100  5.553

Coefficients:
             Estimate Std. Error   t value  Pr(>|t|)
(Intercept)   4.87809  1.738302  2.80635  0.009020
precip       0.09643  0.009735  9.90512  1.19e-10

(Intercept) **
precip      ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.653 on 28 degrees of freedom
Multiple R-squared: 0.778, Adjusted R-squared: 0.77
F-statistic: 98.11 on 1 and 28 DF,  p-value: 1.187e-10

Notes on optional parameters:
Direction: can take the values "backward", "forward", "both"
trace: if positive, gives information on each step. Larger values give more info.
k: allows you to change the number of degrees of freedom for the penalty
(3) Means comparisons: ANOVA & t-test
Get data frames three and two
three<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/ThreeTreatmentDataset.csv",sep="","header=TRUE)
two<-three[-which(three$treatment=="potassium")],]

3A. Independent, 2-sample t-test, test of equal variance, and wilcoxon rank test
First, the simplest ANOVA, an independent-sample t-test on data frame "two"
Two treatments (control and nitrogen), with 10 reps each; dependent variable plant_mass
#independent-sample t-test with unequal variances
t.test(plant_mass~treatment, data=two)
  Welch Two Sample t-test

data:  plant_mass by treatment
t = -16.3829, df = 15.237, p-value = 4.359e-11
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -2.446290  1.883710
sample estimates:
mean in group control mean in group nitrogen
  2.479                  4.644

#Note - t.test() default is to assume unequal variances.
#to assume equal variances and pool them,
t.test(plant_mass~treatment, data=two, var.equal=T)

#Test whether the variances are different or not
var.test(plant_mass~treatment, data=two)
  F test to compare two variances

data:  plant_mass by treatment
F = 2.4831, num df = 9, denom df = 9, p-value = 0.1916
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
  0.6167757  9.9970885
sample estimates:
  ratio of variances
  2.483135
  # p=0.19 so we cannot reject that the variances are equal

#non-parametric rank-test comparison of two independent samples
wilcox.test(plant_mass~treatment, data=two)
  Wilcoxon rank sum test with continuity correction

data:  plant_mass by treatment
W = 0, p-value = 0.0001806
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x = c(2.2, 2.64, 2.64, 2, 2.9, 2.28, 2.82, :
  cannot compute exact p-value with ties
3B. Paired-sample and one-sample tests
Rearrange the data in "two" so that they are "paired" data (pretend they were collected that way)

```r
cnpair <- data.frame(cbind(two[1:10,2], two[11:20,2]))
names(cnpair) <- c("control", "nitrogen")
cnpair
```

```r
count nitrogen
1  2.20 4.68
2  2.64 4.84
3  2.64 4.20
4  2.00 4.77
5  2.90 4.38
6  2.28 4.79
7  2.82 4.77
8  2.36 4.83
9  2.93 4.43
10 2.02 4.75
```

t.test(cnpair$control, cnpair$nitrogen, paired=TRUE) #paired t-test

```
        Paired t-test

data:  cnpair$control and cnpair$nitrogen

  t = -13.4803, df = 9, p-value = 2.842e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.528313  -1.801687
sample estimates:
mean of the differences
  -2.165
```

wilcox.test(cnpair$control, cnpair$nitrogen, paired=TRUE) #non-parametric test

```
 Wilcoxon signed rank test
data:  control and nitrogen

  V = 0, p-value = 0.001953
alternative hypothesis: true location shift is not equal to 0
```

t.test(cnpair$control, mu=2.0) #1-sample t-test, is control mean ≠ 2.0?

```
        One Sample t-test
data:  cnpair$control

  t = 4.2929, df = 9, p-value = 0.002011
alternative hypothesis: true mean is not equal to 2
95 percent confidence interval:
 2.226591 2.731409
sample estimates:
mean of x
  2.479
```
3C. Simple one-way ANOVA (using dataframe "three")

There are a variety of ways to do Analysis of Variance in R. Here we will look at the functions lm, aov, oneway.test, and the Kruskal-Wallace Test.

• lm and aov give identical results, but the output is slightly different.
• If you have multiple error terms, you must use aov.
• Some extractor functions, like Tukey HSD and model.tables are not compatible with lm.
• Overall, aov is probably more generally useful for ANOVAS.
• oneway.test is a variant that does not assume equal variances across groups.
• kruskal.test is a non-parametric comparison among groups.
• If you want to do mixed-models and specify fixed and random effects, use lmer in lme4 package.

First, take a quick look at data means and variance measures using box-and-whisker plots.

```
plot(three$plant_mass~three$treatment)
```

**How to read a box-and-whisker plot:** The thick line is the median; The upper and lower part of the box are the 25% and 75% percentiles (1st and 3rd quantiles); The whiskers show either the maximum and minimum values OR 1.5X the interquartile range (~2 standard deviations), whichever is smaller. If the latter, outlier points are shown individually.
Using the "lm" approach to ANOVA

attach(three)
a3<lm(plant_mass~treatment)  #run an ANOVA using lm and put in a3
anova(a3); summary(a3)  #show basic info from model object
pairwise.t.test(plant_mass,treatment)  #LSD pairwise post-hoc
meansa3<tapply(plant_mass,treatment, mean)  #get the means for each treatment
sda3<tapply(plant_mass,treatment, sd)  #get the sd for each treatment
as.table(cbind(meansa3,sda3))  #put means and sd into pretty table
detach(three)

#NOTE: SEE THE NEXT PAGE FOR SOME HELP IN INTERPRETATION OF THE OUTPUT

Analysis of Variance Table

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>2</td>
<td>23.589</td>
<td>11.7946</td>
<td>121.15</td>
<td>3.274e-14 ***</td>
</tr>
<tr>
<td>Residuals</td>
<td>27</td>
<td>2.6285</td>
<td>0.0974</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Call:

lm(formula = plant_mass ~ treatment)

Residuals:

Min 1Q Median 3Q Max
-0.4790 -0.2630  0.0730  0.1797  0.5200

Coefficients:

|                  | Estimate | Std. Error | t value | Pr(>|t|)       |
|------------------|----------|------------|---------|---------------|
| (Intercept)      | 2.47900  | 0.09867    | 25.125  | < 2e-16 ***   |
| treatmentnitrogen| 2.16500  | 0.13954    | 15.516  | 5.66e-15 ***  |
| treatmentpotassium| 0.93100  | 0.13954    | 6.672   | 3.68e-07 ***  |
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.312 on 27 degrees of freedom
Multiple R-squared: 0.8997,    Adjusted R-squared: 0.8923
F-statistic: 121.2 on 2 and 27 DF,  p-value: 3.274e-14

> pairwise.t.test(plant_mass,treatment)  #LSD pairwise post-hoc

Pairwise comparisons using t tests with pooled SD
data:  plant_mass and treatment

  control nitrogen
nitrogen  1.7e-14 -
potassium 3.7e-07 3.7e-09

P value adjustment method: holm

> meansa3<tapply(plant_mass,treatment, mean)  #get the means for each treatment
> sda3<tapply(plant_mass,treatment, sd)  #get the sd for each treatment
> as.table(cbind(meansa3,sda3))  #put means and sd into pretty table

  meansa3  sda3
control  2.4790000 0.3528440
nitrogen 4.6440000 0.2239147
potassium 3.4100000 0.3426693
Interpreting `summary()` and `anova()` estimates from `lm()` ANOVA models in R

In the data frame three used in the example above, we are looking at an experiment with one treatment factor that has three levels: control, nitrogen, and potassium. Above we created the model object called “a3” from `lm(plant_mass~treatment)`. In many statistical packages, we get a measure of the significance of the overall effect of treatment, including all the levels. This kind of output is extracted using `anova()` or `summary.aov()`

```r
> anova(a3)
            Df Sum Sq Mean Sq F value    Pr(>F)
 treatment   2 23.589 11.7946 121.150 3.274e-14 ***
 Residuals 27  2.628  0.0974
```

However, when we used `lm()` to create a ANOVA model object the `summary(a3)` output looks like this (in part):

```
Coefficients:            Estimate Std. Error t value Pr(>|t|)
(Intercept)              2.47900    0.09867  25.125  < 2e-16 ***
treatmentnitrogen        2.16500    0.13954  15.516 5.66e-15 ***
treatmentpotassium       0.93100    0.13954   6.672 3.68e-07 ***
```

In a regression, the Coefficient estimates are easy to interpret – they are the coefficients for the regression equation, starting with the intercept, and then a coefficient for each term in the model (num_spp ~ temp + precip would give three coefficient estimates – one intercept and one each for temp and precip). For this ANOVA, we use the simple model `lm(plant_mass~treatment)` but get three coefficients. That is because R defaults to treatment contrasts in presenting output from `lm()` anova models. Recall that the underlying model fit by `lm(plant_mass~treatment)` is really:

\[
\text{plant_mass} = a + b*\text{treatment}_1 + c*\text{treatment}_2 + d* \text{treatment}_3
\]

By convention, the mean for whatever treatment level comes first in your data frame (in this case, control) becomes the coefficient estimate for the Intercept. The other estimates are the differences between this mean and the other means of treatment levels.

So: to get the mean value of nitrogen, it would be 2.479+2.165 = 4.644.

The mean value of potassium would be 2.479+0.931 = 3.41

Recall from above that when we calculated group means using `tapply`, we got:

```
means        sd        control   2.4790000 0.3528440
            nitrogen  4.6440000 0.2239147
            potassium 3.4100000 0.3426693
```

You can control which is the reference level using "relevel"

```r
fb$treatment<-relevel(fb$treatment,ref="potassium")
summary(lm(fb$plant_mass~fb$cultivar*fb$treatment))
```
**Using the "aov" approach to ANOVA**

```r
attach(three)
a4 <- aov(plant_mass ~ treatment)
a4; summary.lm(a4);
TukeyHSD(a4) # does not work for lm
model.tables(a4, "means", se = T, n = T) # does not work for lm
meansa4 <- tapply(plant_mass, treatment, mean)
sda4 <- tapply(plant_mass, treatment, sd);
as.table(cbind(meansa4, sda4))
detach(three)
```

We'll just look at a couple things here, rather than show the full output. When using `aov()` for `anova`, the `summary()` and `anova()` extractors function identically, but differently from what `summary()` give you for an object created using `lm()`. The extractor function `summary.lm()` gives the treatment-contrast approach discussed above.

`anova()` provides a test of the overall significance of treatment (all the levels included)

```
anova(a4)
```

Analysis of Variance Table

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>2</td>
<td>23.5891</td>
<td>11.7946</td>
<td>121.15</td>
<td>3.274e-14 ***</td>
</tr>
<tr>
<td>Residuals</td>
<td>27</td>
<td>2.6285</td>
<td>0.0974</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

`summary.lm()` provides information about the significance of the effect of each treatment level, as described in the previous section.

```
summary.lm(a4)
```

Coefficients:

|              | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------|----------|------------|---------|----------|
| (Intercept)  | 2.47900  | 0.09867    | 25.125  | < 2e-16  *** |
| treatmentnitrogen | 2.16500  | 0.13954    | 15.516  | 5.66e-15 *** |
| treatmentpotassium | 0.93100  | 0.13954    | 6.672   | 3.68e-07 *** |

Residual standard error: 0.312 on 27 degrees of freedom
Multiple R-squared: 0.8997,  Adjusted R-squared: 0.8923
F-statistic: 121.2 on 2 and 27 DF,  p-value: 3.274e-14

Note that several other useful extractor functions, like `TukeyHSD()` and `model.tables()` work for model objects created using `aov()` but not for `lm()`.

```
TukeyHSD(a4)  # provides a conservative post-hoc comparison of treatment means
Tukey multiple comparisons of means
$treatment$
```

```
diff     lwr      upr  p adj
nitrogen-control  2.165  1.8190293  2.5109707  0.0e+00
potassium-control  0.931  0.5850293  1.2769707  1.1e-06
potassium-nitrogen -1.234 -1.5799707 -0.8880293  0.0e+00
```

Overall, doing ANOVAS using `aov()` is probably a bit more generally useful because there are more analytical options.
3D. Using the "oneway.test" approach to ANOVA
This is a one-way analysis of variance that does not assume equal variances.

```r
a5 <- oneway.test(plant_mass ~ treatment, data=three)
> a5
One-way analysis of means (not assuming equal variances)

data:  plant_mass and treatment
F = 140.9555, num df = 2.000, denom df = 17.123, p-value = 2.319e-11

#note – this is pretty much all that is available. There is not much to extract.
str(a5)
List of 5
$ statistic: Named num 141
  ... attr(*, "names")= chr "F"
$ parameter: Named num [1:2] 2 17.1
  ... attr(*, "names")= chr [1:2] "num df" "denom df"
$ p.value : num 2.32e-11
$ method : chr "One-way analysis of means (not assuming equal variances)"
$ data.name: chr "plant_mass and treatment"  
  - attr(*, "class")= chr"htest"
```

Using the non-parametric "kruskal.test" approach to ANOVA
Here is the non-parametric Kruskal-Wallis test for differences among groups.

```r
a6 <- kruskal.test(plant_mass ~ treatment, data=three)
a6
Kruskal-Wallis rank sum test

data:  plant_mass by treatment
Kruskal-Wallis chi-squared = 25.8179, df = 2, p-value = 2.476e-06

#As for oneway.test, that is pretty much it. Not much to extract.
str(a6)
List of 5
$ statistic: Named num 25.8
  ... attr(*, "names")= chr "Kruskal-Wallis chi-squared"
$ parameter: Named num 2
  ... attr(*, "names")= chr "df"
$ p.value : num 2.48e-06
$ method : chr "Kruskal-Wallis rank sum test"
$ data.name: chr "plant_mass by treatment"  
  - attr(*, "class")= chr"htest"
```
3E. Factorial ANOVAS

```r
fb<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/factorialdata.csv",sep=""," ,header=TRUE)  # experiment with two cultivars and three treatments, 10 reps of each combination.
For this part, imagine it is a completely randomized design (i.e., ignore "block")

attach(fb)
a6<-aov(plant_mass~cultivar*treatment) #set up as a factorial using *
a6; anova(a6); summary.lm(a6)
model.tables(a6,"means",se=TRUE)
TukeyHSD(a6)
interaction.plot(cultivar,treatment,plant_mass)
detach(fb)

In this case, anova(a6) shows that overall, there is only a marginally significant effect of the
interaction term, but that both cultivar and treatment have main significant main effects.
```

Response: plant_mass

```
Df  Sum Sq Mean Sq  F value   Pr(>F)
cultivar            1 129.067 129.067 75.1196 8.39e-12 ***
treatment           2  72.817  36.409 21.1906 1.61e-07 **
cultivar:treatment  2   8.945   4.473  2.6032   0.0833 .
Residuals          54  92.780   1.718

summary.lm(a6) breaks it down into treatment contrasts, and shows that the effect of treatment is
due entirely to nitrogen.
```

```
Estimate Std. Error t value Pr(>|t|)
(Intercept)                           4.9400     0.4145  11.918  < 2e-16 ***
cultivarwildtype                      2.6800     0.5862   4.572 2.86e-05 ***
treatmentnitrogen                     2.9600     0.5862   5.049 5.38e-06 ***
treatmentpotassium                    0.8800     0.5862   1.501    0.139
cultivarwildtype:treatmentnitrogen   -0.5400     0.8290 -0.651    0.518
cultivarwildtype:treatmentpotassium  1.3000     0.8290  1.568    0.123

model.tables(a6) gives you a quick summary of means, standard errors, and n
```

Tables of means

```

Grand mean 7.686667
cultivar
correct  GM  wildtype
6.220 9.153
treatment
control  nitrogen  potassium
6.28 8.97 7.81
cultivar:treatment
correct  treatment
cultivar  GM  control  nitrogen  potassium
4.94 7.90 5.82
wildtype 7.62 10.04 9.80

Standard errors for differences of means

cultivar treatment cultivar:treatment
0.3384 0.4145 0.5862
replic. 30 20 10
```
3F. Blocked or split-plot ANOVAS

fb<-read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/factorialdata.csv",sep"," ,header=TRUE)

Experiment with two cultivars and three treatments, arranged in 10 complete blocks with a split-plot design. Each block is split in two, with one half sown to GM and half to wildtype. Each cultivar is then split in three, and receives nitrogen, potassium, or control.

Thus, each block has all combinations of 2x3 within it.
The largest plot size (block) is split with cultivar treatments; each cultivar is then split with one of three fertilizer treatments.

The treatments (cultivar and treatment) are coded as a factorial, either as cultivar*treatment or (cultivar+treatment)^2. The blocked, split-plot design requires specifying the error terms explicitly, start from large to small, but not including the smallest unit.

Thus: `aov(plant_mass~(cultivar + treatment)^2 + Error(block/cultivar))`

Note that not all the usual extractor functions work when the Error terms are specified.

```
attach(fb)
a7<-aov(plant_mass~(cultivar + treatment)^2 + Error(block/cultivar))
a7; summary(a7) #note: anova() and summary.lm() don’t work for this kind of model
interaction.plot(cultivar,treatment,plant_mass)
detach(fb)
```

3G. Nested ANOVA

nes<-

read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/NestedAOVdata.csv",sep"," ,header=TRUE)

Comparison of native vs. exotic species of clover, with three species nested within each origin; 12 individuals of each species; dependent variable is percent survival.

General format for nesting is y~A/B/C, going from largest to smallest left to right.

For nes, where species is nested within origin:

```
nes<-nes[order(nes$origin,nes$species),]  #sort data frame by origin and species
a8<-lm(survival~origin/species, data=nes)
a8; anova(a8); summary(a8)
```
3H. ANCOVA (Analysis of Covariance)

Three treatments (control, nitrogen, potassium), with plants each. Measured the initial size (the covariate) and the plant_mass at the end of the experiment. Treatment is a categorical factor, and init_size is a continuous variable.

First take a look at how the dependent variable plant_mass is associated with the covariate init_size. Use pch to give each treatment a different symbol.

```r
plot(plant_mass~init_size, pch=as.numeric(treatment), data=three)
```

It looks like plants that started off larger ended up larger in the end. It is important to test whether any effect of treatment might be a simple product of initial size, or whether there is an interaction between treatment and initial size.

Analysis of covariance to test whether there is an effect of treatment on biomass, after removing the effect of initial size.

We will use AIC to test whether the interaction and covariate are important in the model.

To start with the full model, use the * to include interactions, and then fit each of the other reduced models. Note that the order of the terms in the model matters – the covariate comes first, and then the factors. Then use AIC to select the best model.

```r
Lout1<-lm(plant_mass~init_size*treatment, data=three)
Lout2<-lm(plant_mass~init_size + treatment, data=three)
Lout3<-lm(plant_mass~treatment, data=three)
Lout4<-lm(plant_mass~init_size, data=three)
AIC(Lout1); AIC(Lout2); AIC(Lout3); AIC(Lout4)
```

The resulting AIC values are 20.39, 20.35, 20.09, and 80.12 respectively. Models 1, 2, and 3 are not different from each other; the simplest model, then is Lout3, which only includes treatment. You can get to the same place using step(Lout1). You can also compare models using ANOVA.

```r
anova(Lout1,Lout2,Lout3,Lout4)
```

Analysis of Variance Table

<table>
<thead>
<tr>
<th>Model 1: plant_mass ~ init_size * treatment</th>
<th>RSS</th>
<th>Df</th>
<th>Sum of Sq</th>
<th>F</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.Df</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>2.1740</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>2.4799</td>
<td>-0.3058</td>
<td>1.6880</td>
<td>0.2061</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>2.6285</td>
<td>-0.1487</td>
<td>1.6411</td>
<td>0.2124</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>20.7803</td>
<td>-18.1517</td>
<td>200.3833</td>
<td>3.807e-13***</td>
</tr>
</tbody>
</table>

Use anova() and summary() to extract the most useful information from the model object.
(4) Testing for homogeneity of variance among groups

Testing for homogeneity of variance among groups for ANOVAS

There are two common tests for homogeneity of variance, Bartlett and Fligner-Killen

\[
\text{attach(three)} \\
\text{bartlett.test(plant_mass~treatment)} \\
\text{fligner.test(plant_mass~treatment)} \\
\text{detach(three)}
\]

\[
> \text{bartlett.test(plant_mass~treatment)} \\
\text{Bartlett test of homogeneity of variances} \\
\text{data: plant_mass by treatment} \\
\text{Bartlett's K-squared = 1.9741, df = 2, p-value = 0.3727}
\]

\[
> \text{fligner.test(plant_mass~treatment)} \\
\text{Fligner-Killeen test of homogeneity of variances} \\
\text{data: plant_mass by treatment} \\
\text{Fligner-Killeen:med chi-squared = 3.6421, df = 2, p-value = 0.1619}
\]

# Both tests indicate no significant difference in variance among groups

(5) Type I and III Sums of Squares

There is a raging debate over the appropriate uses of Type I, Type II, or Type III sums of squares. Type III SS have become quite common in Ecology, largely as a legacy of the way SAS handles things. By default, though, R uses Type I (sequential) sums of squares. This is an issue for studies with unbalanced designs – in balanced designs, Type I and III are the same. We are not going to discuss the relative merits in this class. The Car package in R is designed to handle different types of SS easily – check it out. There is a lot of discussion on the web on different approaches. However, here is how you can do it in R base package.

# Create an unbalanced version of the fb (factorial) data frame from above

\[
\text{fbu<-fb[-c(6,16,18,33,34,59),]} \\
\]

# To get ANOVA table with Type I Sums of Squares

\[
\text{fbout<-lm(plant_mass~cultivar*treatment, data=fbu)} \\
\text{anova(fbout)}
\]

# to get Type III SS:

\[
\text{options(contrasts=c("contr.sum","contr.poly"))} \\
\text{fbout2<-lm(plant_mass~cultivar*treatment, data=fbu)} \\
\text{drop1(fbout2, .~., test="F")}
\]

# Note: if instead you run these two versions using fb data frame, the ANOVA tables are identical
Operators
* interaction : interaction / nested ^ interaction I() as is ~. the model as it is Error() specify error terms

Linear regression models
Lout<-lm(DepVar~IndepVar, data=Mydata) #simple linear regression
Lout; anova(Lout); summary(Lout); plot(Lout) #extract model object
predict(Lout,interval = "confidence, level = 0.95) #95%CI

ANOVA models
Aout<-aov(DepVar~Factor, data=Mydata) #one-way ANOVA
Aout<-aov(DepVar~FactorA*FactorB, data=Mydata) #two-way factorial ANOVA
Aout<-aov(DepVar~(FactorA+FactorB)^2, data=Mydata) #two-way factorial ANOVA
Aout<-aov(DepVar~Factor + Block, data=Mydata) #Randomized block ANOVA
Aout; anova(Aout); summary.lm(Aout);TukeyHSD(Aout) #extract model object

t-tests
t.test(DepVar~Factor, data=Mydata) #independent sample t-test
t.test(DepVar~Factor, data=Mydata, var.equal=T) #ind-sample t-test, pooled variance
t.test(DepVar, mu=2.0) #one-sample t-test that mean is different from 2.0
t.test(DepVar1,DepVar2, data=Mydata, paired=TRUE) #paired sample t-test
wilcox.test(DepVar~Factor, data=Mydata) #wilcoxon rank-sum test

Testing assumptions
var.test(DepVar~Factor, data=Mydata) # test if two variances are equal
bartlett.test(DepVar~Factor,data=Mydata) #Bartlett test of homogeneity of variances
fligner.test(DepVar~Factor,data=Mydata) #Fligner-Killeen test homogeneity of variance
cor(Mydata[col:col], method="pearson") #pairwise correlation coefficients (method = pearson, spearman, or kendall)
cor.test(var1, var2, method="spearman") #test of correlation between two variables

ANCOVA models
Lout<-lm(DepVar~Covar*Factor, data=Mydata) #Analysis of Covariance ANCOVA

Stepwise regression
step(lm(DepVar~IVar1+IVar2+IVar1*Ivar2, data=Mydata),direction="backward", trace=1)
AIC(modelobject) #gives AIC values

Plotting
boxplot(DepVar~Factor, data=Mydata) #Boxplot of means across groups
coplot(DepVar~CoVar|Factor, data=Mydata) #visualize covariance across factors
plot(DepVar~IndepVar); abline(lm(DepVar~IndepVar, data=Mydata)); CI95<-predict(Lout,int="c",level=0.95); matlines(IndepVar, CI95, lty=2) #plot w 95%CI
pairs(Mydata[col:col]) #pair-wise scatterplots of variables in a data frame

Post-hoc tests
TukeyHSD(Aout) #post-hoc HSD comparison of means from model object
pairwise.t.test(DepVar,Factor) #post-hoc LSD comparison of means