Lab 6: Independent Contrasts

Today we're going to use *R* to derive and analyze independent contrasts for continuous data. First we're going to learn about some statistical models in *R* by analyzing the raw data from our last lab. Then we'll generate some phylogenetically independent contrasts and see how they affect our analyses. Finally, in the last section you will be given a second data set to analyze and explain.

Several other programs can do PIC, including *CAIC, PDAP* and the PDAP:PDTree module in *Mesquite.* We will stick with *R* for now, but you may want to explore some other options on your own.

A word to the wary: I will use two abbreviations in this lab ***PC (Principle Component)*** and ***PIC (Phylogenetically independent Contrast)***. These are very different things and you should be sure to keep them straight.

Statistical Analysis in R

First open your workspace from the last lab. Type **"ls()"** to see all your objects.

Statistical Distributions

R has a number of statistical distributions automatically embedded in the software. These can be very useful for exploring and analyzing data. To see a list of distributions look at the Intro to R documentation on line. I will use an example of the *normal* distribution, which we are all familiar with. Later we will use the *bivariate normal* distribution and the *binomial* distribution.

For any distribution you can add a prefix before the name of the distribution to create different functions:

r---(n,parameters) will return *n* random draw from a distribution.
q---(p,parameters) will return the value for the distribution which has the given *p-value.*
p---(q,parameters) will return a p-value for a given value from the distribution.
d—(x,parameters) will return the probability density for a given value.

Let's compare our snout-vent length data to a normal distribution. One assumption of linear regression is that all the parameters are normally distributed.

```
hist(Anole.ordered[,1])
mn<-mean(Anole.ordered[,1])
stand<-sd(Anole.ordered[,1])
```

That looks pretty crappy. It looks like it has tons of positive skew. Let's see what the p-values look like for some of our biggest numbers:

quant<-pnorm(sort(Anole.ordered[,1]),mn,stand)
quant[28:30]

Considering that we only have 30 values here, the fact that 3 of them are in the top 98.7% percentile is more than a little suspicious. (What percentile would you expect them to be in?) The p-values of our data could be considered the expected percentage of data points that should be less than that value. Furthermore the cumulative probability of the sampled data points represent the percentage of data points that are less than that value for the actual distribution. If our data is in fact normal, then our p-values should match up with the cumulative probability of our data points, which are distributed evenly between 0 and 1. Let's plot our p-values against some evenly distributed fractions and draw a straight line with slope equals 1 to see if they are the same:

plot((1:30)/31,quant)
abline(0,1)

No, definitely not.
On the flip side let's see what values we expect at the 90^{th} percentile:

qnorm(0.90,mn,stand)

While we actually got:

sort(Anole.ordered[,1],decreasing=TRUE)[3]

For the normal distribution you can actually make plots of these comparisons automatically using *qqnorm*, but you could make these plots yourself anyway for any distribution.
Finally let's look at what this distribution would look like if it had the same parameters, but was normal:

hist(rnorm(30,mn,stand))
hist(rnorm(3000,mn,stand))

Parametric Correlation of Variables

Now let's plot all our data against each other to see what it looks like:

pairs(Anole.ordered)

As you can see all these variables seam to be positively correlated. This is not surprising, since they are all measures of body size. For our initial analysis we are going to look at *mass* as a function of *snout-vent length*, so plot them against each other:

plot(Anole.ordered[,1],Anole.ordered[,2])

To test for this correlation we will use the *cor.test* function based on *Pearson's product moment correlation.*

.

cor.test(Anole.ordered[,1],Anole.ordered[,2],alternative= "g")

alternative="g" makes the test one tailed for a positive correlation between the variables. We can make this test one tailed, because we can *a priori* assume a positive relationship between measures of size. If you could not make this assumption, then you should keep the test two-tailed.

A bunch of information will appear on the screen. At the bottom you are given the calculated correlation (the covariance divided by the total variance) and 95% confidence intervals. Above that you see statistics testing the hypothesis that there is no correlation between these data. The t-statistic is used with 28 degrees of freedom and the null hypothesis is completely rejected.

Assumptions of Pearson's Correlation

That looks great, right? Low p-values high R^2? No, it sucks ass. The problem is that the data does not match the assumptions of the model, in particular the data are not from a bivariate normal distribution. That means not only is each data set normally distributed, but for every value of one parameter the other is normally distributed. First let's test the assumption that each variable is normally distributed.

qqnorm(Anole.ordered[,1])
qqline(Anole.ordered[,1])

This is a plot of the quantiles of your data against the expected quantiles of a normal distribution. q*qline* puts a line through the first and third quartiles. If this distribution was close to normal, then all the dots would be close to the line. As you can see they are not, the terminal values fall far from the line. Repeat this for your *mass* as well.

Since neither data set is normally distributed, you know that together they do not fit a bivariate normal distribution, but let's check anyway just for fun. This is a little tricky, so I wrote a function that will plot the cumulative probability of our data against their p-values from the multivariate normal distribution. The values from these distributions should match exactly, so we'll add a line of slope 1 through the origin for comparison. Load the ***mnormt*** package, open *test.mvn.R*, run it, and then:

qqmvn(Anole.ordered[,1:2])
abline(0,1)

Wow, that really sucks.

Transforming Data

Luckily we do have some recourse, transforming the data. Transforming data is a science and an art, we are not going to spend any time on it here. We will just rely on the simple log-log

transform that we already did. Not only is this justified in that you may expect this type of data to be log normal, it also makes sense because these data should pass through the origin, so a log log transform will show a linear relationship between any two variables that have a relationship $y=bx^a$, which is a more general form of the normal linear relationship passing through the origin.

Let's check all our assumptions for the log transformed data:

qqnorm(Anole.log[,1])
qqline(Anole.log[,1])

qqnorm(Anole.log[,2])
qqline(Anole.log[,2])

qqmvn(Anole.log[,1:2])
abline(0,1)

Still not perfect, but over all pretty good, and a lot better than what we saw before. Let's test for this correlation:

cor.test(Anole.log[,1],Anole.log[,2],alternative= "g")

That looks great, still highly correlated.

Nonparametric Tests

So what do you do, if you can't get your data to fit the assumptions of your model? You use a nonparametric test, which does not rely on any assumptions. These tests are more ***robust*** than parametric tests, but less ***powerful***. There are an infinite number of nonparametric tests that depend on what hypothesis you're testing. In this case we want to see if one parameter is larger when the other parameter is larger. We're going to try two test statistics, *Kendall tau* and *Spearman's rho*. Both tests look to see if higher ranks of one variable are associated with higher or lower ranks of another.

For the *Kendall* test:

cor.test(Anole.log[,1],Anole.log[,2],alternative= "g",method="k")

This gives you the test statistic, *z*, an estimate p-value and *Tau*. The p-value is one-tailed as before. As you can see, the results are still highly significant, although the p-value is a little larger. Check out this test for the data set that has not been log transformed:

cor.test(Anole.ordered[,1],Anole.ordered[,2],alternative= "g",method="k")

The results are identical, because taking the log of a data set does not change the ranks of the data points. If $y>x$, then $ln(y)>ln(x)$.

We can also do the Spearman test, which is very similar:

cor.test(Anole.log[,1],Anole.log[,2],alternative= "g",method="s")

Obviously these two variables are highly correlated.

Testing Multiple Correlations at Once

We are not limited to doing the tests for correlations between variables 2 at a time. p*icante* has a function *cor.table*, which can do every pairwise comparison at once. Load *picante*, then type:

cor.table(Anole.log)

You will get two tables. The first table shows the Pearson's correlation coefficient for each of the pairwise comparisons; the second shows the p-values for these correlations. Unlike *cor.test, cor.table* can only do two-tailed tests, since our test is one tailed, we can divide all these p-values by 2. As you can see, all these correlations are highly significant. However, do they match the assumptions of multivariate normality?

qqbvn(Anole.log)

This is another little function I wrote to test these assumptions. The little plots along the diagonal are comparisons of the distribution of each variable to the normal distribution; the other little plots to the top right are comparisons of each pair of variables to their expectation under a bivariate normal distribution, just like the plot produced by *qqmvn*. The big plot on the bottom left is a comparison of all the data together to their expectation under a multivariate normal distribution. The numbers represent p-values for the *Shapiro-Wilks* test of normality; significant values indicate rejection of the normal distribution. Simulating data showed that both these tests may be a little harsh, but these results clearly indicate that very little of this data really fits a normal distribution.

Luckily *cor.table* can do the same nonparametric tests as *cor.test* can.

cor.table(Anole.log,cor.method="k")

and

cor.table(Anole.log,cor.method="s")

Obviously, all these results are significant, no matter what test you use.

Principle Components Analysis

When we originally looked at the data it was clear that all the characters were highly correlated with size. What if we wanted to look at changes in theses quantitative characters independently of that one big factor? Well, we can look at the **principle components** of this same set of characters. Principle components are the same data points but on a different set of axes. The originally set of axes that our characters were measured in are rotated, so that the total variance is minimized. This new set of axes separates out the parts of the individual characters that are linearly correlated with each other into different components. The idea is that each of

the new axes explain how the data varies together, rather than leaving all the data separated as they were measured.

Why would you want to do this? Let's take as an example a data set with only two columns hind leg length and fore leg length. One thing you could learn from a PCA is how correlated these characters are. If they are totally uncorrelated (that would be a crazy looking animal), then each of the PCs would explain 50% of the variation. One PC would be fore leg length and the other hind leg. On the other hand, if they are highly correlated, then the first PC would explain most of the variation in both fore and hind limbs, and would be called something like limb length. The second PC would explain only a little of the variance and would correspond to something like fore leg to hind leg ratio. The idea is that measurements along the PCs are somehow more informative than those along the original axes. You learn more talking about limb length and fore leg/hind leg ratio than you do by talking about fore leg and hind leg length independently. For data sets with a number of variables PCA can be quite useful for picking out highly informative groupings of variables.

PCs<-prcomp(Anole.log)

First let's look at some basic information about our new axes:

summary(PCs)
screeplot(PCs)

This will show three rows describing the amount of variance described by each PC. Look at the proportion of variance rows. The plot shows the same thing. As you can see the first component explains 96.7% of the variance. That's a lot, and we can assume that this component is essentially size. The standard is to say that all the components that cumulatively explain 95% of the variance are significant, and the rest are just a bunch of noise. In this case that is just the first component. We could just as well set it at 99% or whatever other arbitrarily chosen number.

Now let's look at the vectors these components define:

PCs$rotation

The first component shows positive slopes for all of our values, this is what we would expect for size. The slope is particularly biased to *mass,* as we might expect, since mass should scale to the cube of length in a species of uniform shape. On the other hand the *lamellae* are relatively unaffected. The second component is characterized mostly by an increase in tail length with a slight increase in the hind legs, and interestingly a decrease in mass and in the fore legs. Thus this represents some shift in overall body shape. To see how the different taxa line up on these new axes type:

biplot(PCs)

The x-axis is PC1, so taxa to your left are smaller. The y-axis is PC2, so divergence along this axis shows divergence in this first shape parameter; taxa to the top have relatively long tails and hind legs, and relatively low mass and short fore legs for their size. The vectors in the

middle show how much each of our original variables contributes to each component. As you can see, these two components are completely uncorrelated, that is a consequence of PCA, and tells you nothing.

To compare other PCs (like say the 2nd and 3rd) use the *choices* command:

biplot(PCs,choices=c(2,3))

If for some reason you want values of each of your taxa for each of the PCs, you can find them in *PCs$x*.

Phylogentically Independent Contrasts

Finally, we're going to get into the meat of this lab. We've already discussed the theory behind PICs in lecture, so let's just get right down to it. To calculate PICs for the logs of our traits on our tree:

pic(Anole.log[,1],Anole.ultra,var.contrasts=TRUE)

This will return a matrix of 29 rows and 2 columns. Each row refers to one internal node, and the name of the row gives the name of that node. Column 1 shows the contrast across that node and column 2 shows it's expected variance. Let's make an array containing the contrasts for all our characters and all their variances:

Anole.PIC<-array(dim=c(29,2,6))
for(i in 1:6) Anole.PIC[,,i]<-pic(Anole.log[,i],Anole.ultra,var.contrasts=TRUE)

array will create an array with the dimensions given. *for(i in 1:6)* will repeat the functions given in the command for every element in the vector 1:6 being the value of *i*. I bet you can figure out the rest.

Confirming Assumptions of PIC

So, we have our contrasts, but just like with our linear correlations, we need to make sure that our data fit the assumptions of the model. Luckily David Ackerly has provided us with a function that makes this all very easy.

First load the ***picante*** package. Open the R script *diagnostics.R* and run it; if it doesn't work try *diagnostics2.R* . Then type:

diagnostics(Anole.log[,1],Anole.ultra)

Six plots will appear. The two on the left will test whether the positivized PICs fit a half-normal distribution centered at 0. The top left is just a histogram of the PICs and it should look like just half of a normal distribution. The plot on the bottom right is a QQ-plot of the PICs against a half normal. It should fit approximately a line. Both those plots look good.

You want the next three plots to show no significant relationship. The top middle is the most important; it plots the standardized contrasts against their branch lengths. If there is a

significant negative relationship here, then that implies your branch lengths are bad. Because contrasts are standardized by dividing by branch lengths, long arbitrary branch lengths can make the contrasts too small. This problem can be alleviated by replacing all your branch lengths with their logs. There is a negative slope here; it's not significant, so we wouldn't normally worry about it, but just for fun let's log convert the branches:

```
Anole.tree.log<-Anole.ultra
Anole.tree.log$edge.length<-log(Anole.ultra$edge.length+1)
diagnostics(Anole.log[,1],Anole.tree.log)
```

The p-value is now even higher for that relationship. The plot on the top right shows the relationship between the contrasts and the age of the node; 0 is the time of the root and the present can be found to the right. A positive relationship indicates extremely low signal and a negative extremely high. The plot in the bottom middle shows the relationship between the reconstructed value at a node and the contrasts; a positive relationship implies that the data should be log transformed, before the contrast is taken. To see an example of this look at the untransformed *mass* contrasts to the transformed ones:

```
diagnostics(Anole.ordered[,2],Anole.ultra)
diagnostics(Anole.log[,2],Anole.ultra)
```

See, more justification for a log transformation. The plot on the bottom right is of course the plots of *k* that we learned about during lecture on Thursday. Values of *k* close to 1 show the expected amount of signal under Brownian Motion. If you have a lot of taxa, you may want to suppress the calculation of *k* with *"calcK=FALSE"*, as it can be cumbersome.

Investigate the contrasts for our other traits. Are you satisfied that they fit the assumptions? Great then let's move on.

Parametric Significance of PICs

The first thing that we want to do is test for a significant relationship between the PICs. It is important to recognize the difference between the relationship between the raw data and the relationship between the PICs. In the first case you are asking whether taxa with greater length have larger mass; in the second your question is do evolutionary increases in length correspond with increases in mass.

As always let's start by plotting all our factors against each other:

```
pairs(Anole.PIC[,1,])
```

They all look very correlated. First we'll make sure that our contrasts match the assumptions of Pearson's test. We've already concluded that our data is normal from the half-normal plots in the tests of PIC assumptions. So all we need to test is that our data fits a bivariate normal distribution. Remember that PICs must have a mean of 0, so we're going to have to have to force the fit through the origins by setting the means ourselves:

```
qqbvn(Anole.PIC[,1,],means=rep(0,6))
```

That looks pretty good. Why would you expect the PICs to fit a multivariate normal distribution better than the raw data? The p-values for the Shapiro-Wilks test are not displayed, because its assumptions are not met by PICs, so we're going to have to go off of fit alone. I might be concerned about the fits of all the comparisons with the lamellae or the tail lengths, but the rest look great.

Now we can look for a linear correlation. We also have to force our fit through the origin here. c*or.test* can't do that, but *cor.table* can, if we use the *cor.type="c"* (for contrast) command:

cor.table(Anole.PIC[,1,],cor.type="c")

As you can see all these results are very significant. Keep in mind that we made 15 separate comparisons; you would expect to find one comparison with a p-value less than 0.05 more than 1 in 20 times, when you look at 15 comparisons. As a general rule you should multiply your p-values by the number of comparisons you make. The number of comparisons can easily be calculated as *n(n-1)/2*, where *n* is the number of variables that we have. Let's do that to see the effects on our results.

results.cor<-cor.table(Anole.PIC[,1,],cor.type="c")
results.cor$P*15/2

We divided by two, because these p-values are for a two tailed-test and our test is one tailed. Of course our results are still all significant. They were so significant before, that we wouldn't expect multiplying by 7.5 to have much of an effect. For a quick view of which comparisons are significant try:

results.cor$P*7.5<0.05

Nonparametric Significance of PICs

As with raw data, PICs may not always fit the assumptions of linear regression. In that case you need to do a nonparametric test. The first and most basic test is to ask whether one character increases, when the other increases. This is a very straight forward test. The first thing to do is create a logical vector that is true when both PICs are positive or negative, false when one is negative and the other positive, and excludes contrasts for which either character shows no change.

ref<-Anole.PIC[,1,1]!=0 && Anole.PIC[,1,2]!=0
both.pos<-(Anole.PIC[ref,1,1]*Anole.PIC[ref,1,2])>0
length(both.pos)
sum(both.pos)

For 26 out of 29 comparisons between contrasts either both show an increase or a decrease. We can test the significance of this result by comparing it to the *binomial* distribution. The binomial distribution is a distribution that takes one of two values. The probability of

getting one result or the other is one of the parameters in the model. We will assume that there is a 50% chance of getting each result. In that way we will try to reject the null hypothesis that it is equally likely for the contrasts to change in the same direction as it is for them to change in the opposite.

pbinom(3,29,0.5)

We used 29-26=3, instead of 26, because this formula evaluates the probability of getting that many results or fewer, so for a one tailed test of positive correlation, we want to know the probability of getting as many negative results or fewer. As you can see this relationship is highly significant.

cor.table also has been programmed for you to do non-parametric rank correlation tests for independent contrasts. As with parametric tests, you must use the *cor.type="c"* command:

cor.table(Anole.PIC[,1,],cor.method="k",cor.type="c")

and

cor.table(Anole.PIC[,1,],cor.method="s",cor.type="c")

Are all those results significant? Don't forget to multiply by 7.5. (Would you like to make a chart of all the possible comparisons between PICs using the binomial distribution? If so see the appendix.)

PCA on PICs

You can also generate PCs from PICs. Why would you want to do that? Why is that any better than doing it on the raw data? The idea is that the PCs of the PICs represent the principle axes of evolutionary change. In other words, the ways that things actually change when they evolve. This can obviously be a very interesting bit of information.

We have to do one little trick to do PCA on PICs to force them through the origin. We are going to create a mirror image of the PICs and stick that together with the other PICs. This will double the amount of data, but since PCA does not do any statistical tests, degrees of freedom don't matter, so we'll be fine.

PIC.mirror<-rbind(Anole.PIC[,1,],Anole.PIC[,1,]*(-1))
PCs.PIC<-prcomp(PIC.mirror)

Now let's check them out:

summary(PCs.PIC)

As you can see, the first PC still describes the vast majority of the variance, and it is probably change in size. Let's look at the vectors:

PCs.PIC$rotation

These look very much like the original PCs that we generated. For some data sets there can be large differences. When would you expect to see such a difference?

Finally, let's plot them:

biplot(PCs.PIC)

or

biplot(PCs.PIC,choices=c(2,3))

You will see your contrasts spread out over the PCs. This tells you how much change of each type happened along each contrast. Remember you mirrored your contrasts, so each one will appear twice on opposite sides of the plot.

Your Assignment

Here is your assignment for this week. I'm really going to grade this one, not just mark off that you did it, so do a good job. Turn it in to me on paper in class next Tuesday, the 24th of February. **Please try to be concise.**

You should already have downloaded the nexus file *PIC.assignment* from the web site. This file has a tree with 49 taxa and a data matrix with 3 continuous characters. This is a modified version of a file from the *Mesquite* examples.

1. Are these characters correlated without considering the phylogeny? What are the p-value of the correlations? Is the correlation positive or negative? What statistical test did you use? Why did you choose that test? Was the test two-tailed or one-tailed? Why?

2. Are these characters correlated, when you do consider the phylogeny? Does the data fit the assumptions of independent contrasts? What are the p-value of the correlations? Is the correlation positive or negative? What statistical test did you use? Why did you choose that test? Was the test two-tailed or one-tailed?

3 Why is there a difference in result between these two tests for significance? Be specific; you should both look over the contrast plots and plot reconstructions of all these characters on a tree (in R or Mesquite). Does this say anything interesting about the evolution of these taxa?

You should always ask yourself these questions for any analysis involving PICs.

Appendix

A Brief Digression on Writing Functions

Up until now I've avoided the subject of writing functions. Let's write a function that will produce a table of results from a two-tailed sign test for every possible comparison between traits. I just need to explain a few things first.

In *R* functions are treated as objects, just like variables. When you write a function, you write "function" followed by parentheses, which have the possible input parameters. You can define defaults for those parameters at this time. You also assign that function to a named object like any other object in *R*. The function will then appear in your list of objects.

*"for (**num** in **vector**)"* will run the commands that follow it for every element in the vector. Replacing ***num*** with that element.

"if" is followed by some conditional statement in parentheses. The subsequent commands are only carried out if the conditional statement is true. If it is not true the commands following *"else"* are executed.

All of these execute a series of commands that appear after them; by after I mean either on the same line, or surrounded by "{}".

```
sign.test<-function(x) { ##Create a function object named sign.test
                         ##with one input paramater, x
        n<-dim(x)[2]                    ##How many columns in x
        counts<-ps<-array(0,dim=c(n,n))      ##Create two n by n matrices
        not0<-x!=0                      ##logical vector indicating which changes are 0
        for (i in 1:(n-1)){             ##start a loop that will run variable i from 1 to n-1
                for (j in (i+1):n) {
                        ref<-not0[,i] && not0[,j] ##Neither should be 0
                        both.pos<-(x[ref,i]*x[ref,j])>0
                        pos<-sum(both.pos)      ##The number of positive results
                        counts[i,j]<-pos        ##Add those to above the diagonal
                        neg<-length(both.pos)-pos   ##The number of negative results
                        counts[j,i]<-neg        ##Add those to below the diagonal
                        times<-min(pos,neg)   ##For 2-tailed want pos or neg correlation
                        ps[i,j]<-2*pbinom(times,pos+neg,0.5) ##we do the test
                                                        ##times 2 cause two-tailed
                        ps[j,i]<-ps[i,j]   ##Below diahonal matches the top
                }
        }
        list(counts=counts,ps=ps)   ##return your results as a list
}
```

Now we can run the test on all our data:

sign.test(Anole.PIC[,1,])

There, don't you feel accomplished. The first table shows the number of positive results above the diagonal and the number of negative results below. The second shows the p-values. As you can see many of our correlations are significant, but not all. Don't forget to multiply by 7.5. To see exactly which are significant.

sign.test(Anole.PIC[,1,])$ps*7.5<0.05

How does that compare to the other non-parametric tests?