Modern regression

#1 – Errors again

This section will again use the data on abundance of the dickcissel vs. various environmental variables used before (http://128.196.231.204/614/dickcissel.csv). For sections #1 and #2, we will use only log10(abund) and clDD (climate degree days), so create two variables:

```
lab=log10(?$abund+0.1) #? represents your dataframe
dd=?$clDD
```

Start with some simple exploration. Plot the data.

```
plot(lab~dd)
```

Kind of a noisy mess. Probably not linear. But lets start in the linear domain and compare a robust vs. an OLS (GLM) regression. Run a GLM regression of lab on dd using the lm command. Now run the same model using robust regression (rlm) from library MASS, saving the result into a different object. Now plot the data. Use abline(m1,lty=1) to plot the OLS line (m1= model object from lm command). Use abline(m2,lty=2) to plot the robust regression. Not a big difference in this case is there? Let's add two more lines. The default for rlm is to do a Huber M-estimation. Let's add a bisquare M-estimation and LTS regression:

```
mbsq=rlm(lab~dd,method='MM')
abline(mbsq,lty=3) #bisquare M-estimation
mlts=ltsreg(lab~dd)
abline(mlts,lty=4) #LTS robust regression
```

4) Report the slope and intercept for the four regressions (OLS, Huber M-estimate, Bisquare M-estimate, LTS), no confidence intervals.

What about quantile regression. Do slopes vary with quantile? In particular there is a well-developed theory (covered in class) why the outer envelope might be the most relevant for abundance vs. one variable (captures those cases where that variable is limiting). Load the library quantreg (you will have to download it from CRAN first – remember choose "Install packages" from the packages menu).

Get three model objects representing the 10%, 50%, and 90% lines. Use the "rq" command which works like all the others. Plot the data and the lines:

```
plot(lab~dd)
abline(mq10) # mq10 is output from rq with tau=0.10
etc…
```

5) Report the slopes for these three lines.

This data sure looks nonlinear try plotting nonlinear quantiles using spline quantile regression (rqss):

```
plot(lab~dd) # basic plot
#fit a 90% quantile use qss to indicate a spline on dd
#constraint=N means no contraint (e.g. strictly increasing)
```

```
#lambda=1000 =smooth across a window 1000 wide
fit=rqss(lab~qss(dd,constraint="N",lambda=1000),tau=0.9)
#normally plot blows away what is underneath, but not for rqss
plot(fit,col="blue",lwd=2,lty=2,add=T)
```

6) Report approximately what degree day value has the maximum abundance according this graph (for DD<7000) using the 90% quantile as a reasonable upper envelope. Try using different quantiles (tau=?) and smoothings (lambda=?) to see how they work. **7) Report a one sentence result of something interesting you found while playing with this.**

#2 - Explicit non-linear hypothesis

The nonlinear quantile sure looks a lot more reasonable. Lets try a non-linear approach. Let's try fitting the Michaelis-Menton (MM) function "a*dd/(half+dd)" where a is the y-value of the asymptote and half is the x-value (here=dd) where ½ of the asymptote is reached. The MM function doesn't handle x<0 or y<0, but currently with we have lab<0, so we will add +1 to lab. Now issue an NLS command:

```
mnl=nls(lab+1~a*dd/(half+dd),start=list(a=?,half=?))
```

From the plot guestimate a reasonable value for a and half and put it in place of the ?'s above. If your guesses are really bad you will get an error message about singular gradient matrices – try different values for a, half. Analyze the output. **8) Report the asymptote and weight at which half the asymptote is reached (caution – don't forget to backtransform by subtracting 1 where needed).** Try plotting this (we have to use the predict command just as in the logistic):

```
x=seq(1000,8500,length=101)
lines(x,predict(mnl,data.frame(dd=x)),col="blue", lwd=2)
```

Not a great fit. Lets try a piecewise linear regression with one break point. Recall this is in library "segmented" and starts by fitting an lm model (you might still having it hanging around from the first part of this section) then passing the lm model to the segmented command to get a new model (see class notes or helpfiles for details on syntax). Run it once with an initial breakpoint of 3000 (then plot with one line type (lty=). Run it again with an initial guess breakpoint of 4000 (and plot it with a different line type). **9) Does the model depend on initial guesses?** Now fit a model with two breakpoints. Try a couple. **10) Can you find any initial breakpoints that work?** Try 4500/6500. Compare both of the one breakpoint and the two breakpoint models using the command AIC. **11) Which is superior?**

#3 – Machine Learning - univariate

We will now extend into techniques that commonly use multiple explanatory variables and do what is known as habitat modelling.

The nonlinear and piecewise fits look a little more reasonable, but what if we just want to predict and don't care about the function form? Smooth, local or nonparametric regression is ideal for this. Create a plot that has a LOWESS, Kernel and Local Polynomial lines on it:

```
#setup for plotting
plot(lab~clDD,data=?)
x=seq(1000,9000,length=101)
# loess plot
```

```
mlo=loess(lab~clDD,span=0.75,data=?)
lines(x,predict(mlo,data.frame(dd=x)),lty=1,col="green")
# spline plot
library(splines)
msp=lm(lab~ns(clDD,df=5),data=?)
lines(x,predict(msp,data.frame(dd=x)),lty=2,col="blue")
#kernel smoothing – note no object produce, output is two
# columns which go straight into lines as the x & y inputs
lines(ksmooth(?$clDD,?$lab,"normal",bandwidth=1000),lty=3,col="r
ed")
```

Try changing the span/df/bandwidth parameters and see what happens as you "tune" the smoothing parameter. **12) What does your eyeball estimate tell you is the best smoothing parameter for loess? bandwidth for ksmooth? 13) Report a (non-eyeball) prediction of the lab for dd=5000 using the loess with span=0.9.** Hint: you used the predict command above. How would you modify it to get a numerical prediction out rather than plot it and how would you input just one value.

Part 3 – Machine learning - multivariate

We will continue to use the dickcissel data set for habitat modeling, but throw in all possible explanatory variables.

lm

Although most of the relationships are non-linear, let's try a simple linear-regression:

```
dc.lm<-lm(log(abund+0.1)~.-Present,data=dc)
```

Note: We're using the formula of predict log(abund+0.1) (left side of "~") as a function of everything ("."), using the dataframe "d" (or whatever you called it) as a context. The +0.1 is to avoid log(0) and 0.1 is slightly smaller than the lowest observed abundances of 0.2.

We are going to want to compare the fit of various models. To this end, we will create our own function to calculate r2 on various types of objects, type:

```
"r2<-function(y,obj) {cor(y,predict(obj))^2}"
```

Try it on the linear model "`r2(log(?$abund+0.1),dc.lm)`". How does this compare with the calculated r2 (hint, use the summary function)?

GAM

Finally time to try a new model – General Additive Model (GAM). The GAM is accessed using the "gam" function which works just like the "lm" function. You must issue a "library(mgcv)" first to load the relevant library. From here on out, we will look at 4 variables: clTma, clP, NDVI, grass. Now try:

dc.gam<-gam(log(abund+0.1)~s(var1)+s(var2)+…,data=dc)

Where var1, etc are the variables you are interested in. Note that these variables are wrapped with a "s()" this tells the model to use a spline fit for the var. You can replace one or more of the "s()" with "lo()" to use loesss instead or if you think a relationship is linear, you can try without either s() or lo().

14) Report the r^2 of the GAM. The plot function on a GAM model object gives a nice set of plots. From these plots, **15) which variable has the least effect (plot most like a horizontal line).**

Neural nets

Try predicting using a neural net. In this case, predict presence/absence ("Present~clTma+clP+NDVI+grass"), don't forget to use r2(?$Present,dc.nn) as well. To load neural nets, use "library(nnet)". The function is "nnet" which works just like lm & gam, but requires one additional parameter "size=3". The summary of nnet is not very useful, is it? Predict the probability of being present:

```
predict(dc.nn,data.frame(clTma=30,NDVI=-0.1,
    clP=1000,grass=0.2))
```

and **16) report this probability**.

CART

Now it is time for CART:

```
library(rpart)
#rpart doesn't handle the ~.-Present notation
d2=d[,-2] #create a new dataframe without Present in
names(d2) #make sure Present is gone
dc.rp<-rpart(log10(abund+0.1)~.,data=d2)
```

The two most useful commands for a CART are print( ) and plot(). After plot() use text(dc.rp,digits=2). **17) Report what the top three nodes in the tree are, how many cases are found in each split, and what the predicted abundance is.**

One problem with CART is that the tree can change greatly with slightly different data. To test this, we're going to run CART on two subsamples. Split your data in half:

```
samp<-sample(1:646,323)
```

Now create two objects, one for the sample, and one for the complement of the sample. Use the "subset=samp" or "subset= -samp" parameters to do this. **18) Are the trees the same? Do they have the same biology? Report briefly.**

Briefly explore the issue of overfitting. Run "plotcp(dc.rp)". This plots error in the tree vs. size of tree (cp is a cutoff level and directly relates to # of nodes which is plotted across the top). A dotted horizontal line plots the lowest error (for the biggest tree) +1 standard error (the top of the errorbar for the last dot) vs cp. Find the value of cp and n (# nodes) where the graph crosses the line. **19) Report the cp and n value.**Prune the tree accordingly: dc.rp.prune=prune(dc.rp,cp=?). Run "print(dc.rp.prune)" – did it shrink the tree?