Midterm Examination

Biology 697: Introduction to Computational Data Analysis

10/22/2012

Welcome to your mid-term! I hope you enjoy. Note, in all of the questions below, there are easy not so code intensive ways of doing it, and there are longer more involved, yet still workable ways to answer them. I would suggest that before you dive into analyses, you do the following. First, breathe. Second, think about the steps you need to execute to get answer the question. Write them down. Third, for those parts of problems that require code, put those steps, in sequence, in comments in your script file. Use those as signposts to step-by-step walk through the things you need to do. Fourth, go over these steps, and see if there are any that could be easily abstracted into functions, could be vectorized, or otherwise done so that you can expend the minimum amount of effort on the problem to get the correct answer.

1 Sampling Your System

Each of you has a study system your work in and a question of interest. Give an example of one variable that you would sample in order to get a sense of its variation in nature. Describe, in detail, how you would sample for the population of that variable in order to understand its distribution. Questions to consider include, but are not limited to: Just what is your sample versus your population? What would your sampling design be? Why would you design it that particular way? What are potential confounding influences of both sampling technique and sample design that you need to be careful to avoid? What statistical distribution might the variable take, and why?

```
# Things that must be included for full points:
# 1) Clear description of what is being measured
# 2) Definition of population of what is being measured
# 3) Clear differentiation between sample and population
# 4) Understanding of how the population versus sample of
# population is distributed
# 5) Description of what factors that must be accounted for
# in sampling
# 6) Discussion of sampling design - is it random, stratified,
# stratified random?
```

```
# 7) Discussion of biological factors that will shape variable's
# distribution
```

2 *Phragmites*

At the Plum Island Ecosystems LTER (`http://ecosystems.mbl.edu/PIE/`), they have been collecting data on the rapidly expanding and potentially harmful *Phragmites australis* along transects in a salt marsh (Argulla Rd.) restoration site and a reference control site (Rough Meadows) since 1997. For the sampling, they took measurements of the two tallest *Phragmites* stems per five meter interval along transects. The data is available here - `http://ecosystems.mbl.edu/PIE/data/LTE/LTE-EX-ARGILLA-RM-PHRAGHEIGHTS.html`.

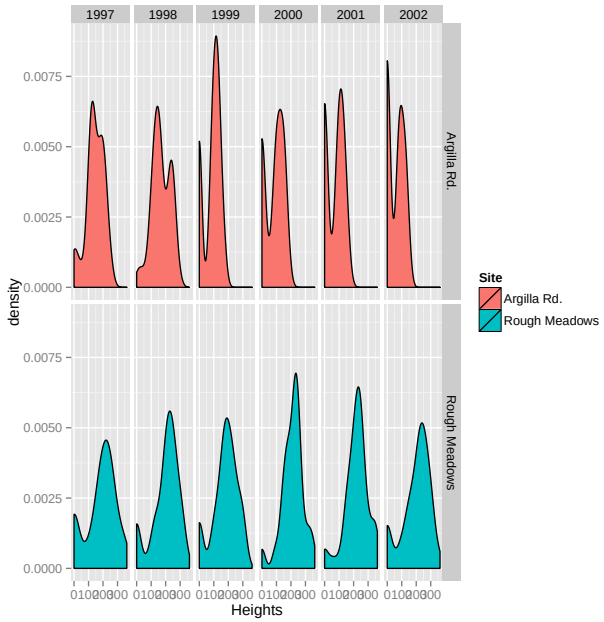
1. Before you even look at the data (no peeking!) would you expect the population that is being sampled from at each site in each year to be normal? Why or why not?

```
While the population of Phragmited may be distributed normally,
the sample is only measuring extreme values - i.e. the tail end
of a normal distribution.  This distribution is likely not to be
normal.  With a little extra digging, we can find that there are
a family of extreme value distributions, of which the Gumbel
Distribution seems a likely candidate, as it is a distribution of
maximum values. http://en.wikipedia.org/wiki/Gumbel_distribution
```

2. Visualize the data in an informative way to see differences in the population of sampled *Phragmites* across space and time.

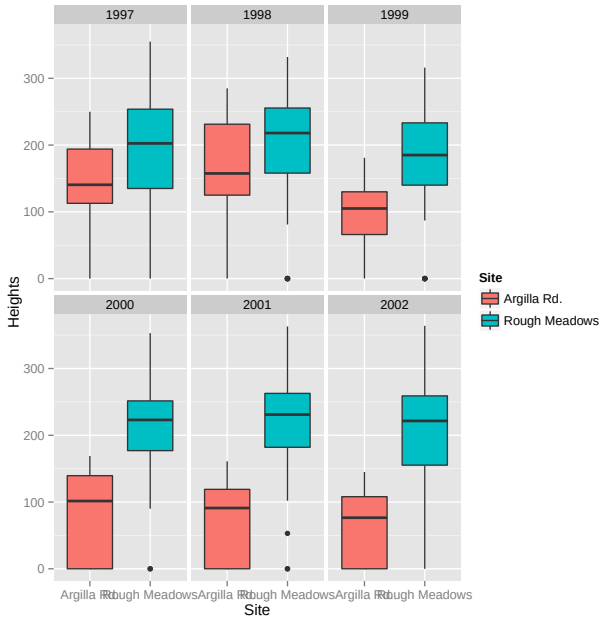
```
To visualize the distributions, we will ll use the density geom in
ggplot2 to see the probability densities.
```

```
library(ggplot2)
phrag <- read.csv("./data/LTE-EX-ARGILLA-RM-PHRAGHEIGHTS.dat")
qplot(Heights, geom="density",  group=Year, fill=Site, data=phrag) +
    facet_grid(Site ~ Year)
```



The densities are funky, and in some cases even bimodal at Argila Road. The typical measures of normal distributions may not be good here. To examine further, let us look at a boxplot.

```
qplot(Site, Heights, geom="boxplot",  group=Site, fill=Site, data=phrag,
      position="dodge") +
  facet_wrap(~Year)
```



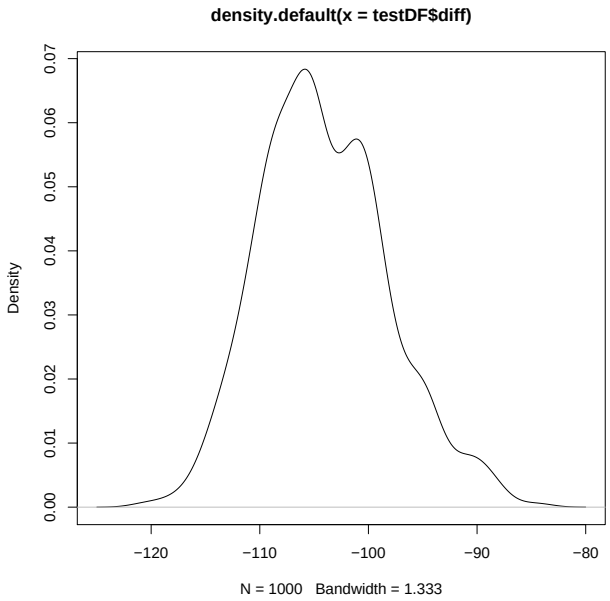
3. Based on your observations of the data, what property or properties of the sampled populations would you want to compare between restoration and reference site to determine effectiveness of restoration?

```
Well, we can see some interesting patterns in the plots above.
The difference seems to get bigger as time goes by, and Argila
Road has its median decrease from the control at Rough Meadows.
The median looks like it would be an interesting measure.  The
number of  transects with 0 tall plants might also be interesting.
```

4. One way to ask if two samples differ in an arbitrary property is to calculate the bootstrapped confidence interval of the difference between them - i.e. calculate a property of a resampled replicate of population a, do the same for population b, take their difference, then rinse and repeat to get an confidence interval on the difference. Write a function to do this, and apply it to one of your properties, pooling across all years. What is your null hypothesis, and what does the result tell you with regards to your null? Use 1000 simulated draws.

```
library(bootstrap)

#a function that, given the name of a site, some set of years, a
#a number of bootstrap draws will get a median from the data
bootMedianPhrag <- function(SiteName, Years=unique(phrag$Year), n.boot=10){
```

```
  #get heights from a subset of the data
  vec <- with(phrag, Heights[which(Site==SiteName & Year %in% Years)])

  #now get the boostrapped median from that vector
  bootstrap(vec, n.boot, theta=median)$thetastar
}

#now generate a data frame with 1000 bootstrap medians from both sites
n.boot <- 1000
testDF <- data.frame(A = bootMedianPhrag("Argilla Rd.", n.boot=n.boot),
                     R = bootMedianPhrag("Rough Meadows", n.boot=n.boot))

#what is the difference?
testDF$diff <- testDF$A - testDF$R

#visualize
plot(density(testDF$diff))
```




```
# As this looks roughly normal, we could do a t-test.  But
# let's also just see the 95% CI of the whole thing
quantile(testDF$diff)

##   0%  25%  50%  75% 100%
```

```
## -121 -108 -105 -100  -84
```

```
t.test(testDF$diff)
```

```
##
##  One Sample t-test
##
## data:  testDF$diff
## t = -558.1, df = 999, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -104.4 -103.7
## sample estimates:
## mean of x
##    -104.1
```

```
Both the 95% CI and the t-test seem to indicate
that this distribution of differences overlaps 0.
The medians between sites do not appear to be different
pooled across all years
```

5. Now look at whether the difference between control and reference site changes across years. How would you interpret this analysis? Feel free to look at additional properties if you think it will help you describe the differences between reference and control.

```
# iterate over all years using sapply with unique(years)
# and look at the difference between the bootstrapped
# medians at each site.  Then get the 95% CI of the
# difference and see how that changes over time
medianDiff <-sapply(unique(phrag$Year), function(ayear){
  A <- bootMedianPhrag("Argilla Rd.", Years=ayear, n.boot=n.boot)
  R <- bootMedianPhrag("Rough Meadows", Years=ayear, n.boot=n.boot)

  #the difference
  diff <- A-R

  #the return value which concatinated year
  #so we get useful output
  c(year=ayear, quantile(diff, c(0.025, 0.975)))

})
```

```
# Transpose the return matrix for easier
# reading
t(medianDiff)
```

```
##      year    2.5%   97.5%
## [1,] 1997  -94.50  -21.99
## [2,] 1998  -85.01  -25.95
## [3,] 1999 -113.51  -58.50
## [4,] 2000 -146.00  -95.00
## [5,] 2001 -219.00 -112.00
## [6,] 2002 -231.50 -114.98
```

```
We can see from the above returned table that Argilla always
has a shorter median maximum Phragmites height.  However,
the median shifts lower and lower over time.  The 95% CI of
the difference in the final year doesn ot even overlap the 95% CI
of the first year.  It might be informative to look at the number
of tranects with no Phrag
```

```
# a function to count number of 0 measurements
n0 <- function(vec) length(which(vec==0))

# a function to get the number of zeroes from a
# bootstrapped sample
boot0 <- function(SiteName, Years=unique(phrag$Year), n.boot=10){

  # get heights from a subset of the data
  vec <- with(phrag, Heights[which(Site==SiteName & Year %in% Years)])

  # now get the boostrapped number of 0s from that vector
  bootstrap(vec, n.boot, theta=n0)$thetastar
}

# like the median by year analysis above, let's iterate
# over all years and look at the difference in number of
# transects with 0 plants
zeroDiff <-sapply(unique(phrag$Year), function(ayear){
  A <- boot0("Argilla Rd.", Years=ayear, n.boot=n.boot)
  R <- boot0("Rough Meadows", Years=ayear, n.boot=n.boot)

  diff <- A-R

  c(year=ayear, quantile(diff, c(0.025, 0.975)))
```

```
})

t(zeroDiff)

##      year 2.5% 97.5%
## [1,] 1997  -10     2
## [2,] 1998  -10    -1
## [3,] 1999    1    17
## [4,] 2000   11    26
## [5,] 2001   13    29
## [6,] 2002   11    29
```

```
In the first year, the 95% CI of number of zeroes
overlaps 0, hence, the two sites may not be different.
This changes as time goes by, and Argilla Rd. accumulates
more and more transects with no Phragmites, and we can see
the 95% CI shift up and stay roughly between 11 and 28.
```

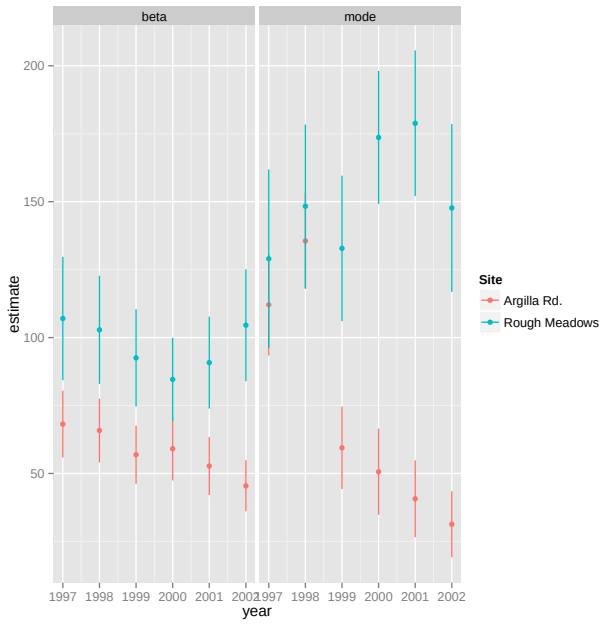
6. Extra Credit: There is a particular distribution that may describe this data well. Using likelihood, fit the distribution to each site x year's data. Visually examine change in the parameter values over time at the control versus restoration site.

```
The Gumbell distribution takes two parameters,
the mode of the distribution and a shape
parameter related to the spread.
From http://en.wikipedia.org/wiki/Gumbel_distribution
m = mode
median = m - beta * ln (ln(2))
sd = beta*pi/sqrt(6)
Density = 1/beta * e^(-z^-e^(-z))
where z = (x-m)/beta
Or there are other generalized extreme value
(GEV) distributions
```

```
# To begin with, we nee da density function for the
# Gumbel distribution

dgumbel <- function(x, mode, beta){
  z <- (x-mode)/beta
  1/beta *  exp(-z - exp(-z))
}
```

```r
#now a negative log-likelihood function
llGumbel <- function(Heights, mode, beta)
  -sum(log(dgumbel(Heights, mode, beta)))

library(bbmle)
#start for beta, with mode = 200
(median(phrag$Heights) - 200) / (log(log(2)))

## [1] 150.1


#let's fit the whole thing
phragGumbel <- mle2(llGumbel, data=phrag, start=list(mode=200, beta=150))

#which CI is better?
confint(phragGumbel)

##      2.5 % 97.5 %
## mode 91.99 106.16
## beta 82.58  92.71

confint(phragGumbel, method="quad")

##      2.5 % 97.5 %
## mode 91.91 106.05
## beta 82.34  92.45


#need a function to get a mode
getMode <- function(x){
  z <- density(x)
  z$x[which(z$y == max(z$y))]

}
#a function to get a startvalue of beta
getBeta <- function(vec) (median(vec) - getMode(vec))/(log(log(2)))

#very close, so, we can use Wald SE for the following...
params <- lapply(unique(phrag$Year), function(ayear){
  subdata<-subset(phrag, phrag$Year==ayear)
  adata <- subdata[which(subdata$Site=="Argilla Rd."),]
  rdata <- subdata[which(subdata$Site!="Argilla Rd."),]

  #used startvalue functions as was having convergence problems
```

```
  a <- mle2(llGumbel, data=adata,
            start=list(mode=getMode(adata$Heights), beta=100), optimizer="nlm")
  r <- mle2(llGumbel, data=rdata,
            start=list(mode=getMode(rdata$Heights), beta=100), optimizer="nlm")

  #combine the coefficients and CIs
  ret <- rbind(confint(a, method="quad"), confint(r, method="quad"))
  ret <- data.frame(year=rep(ayear, nrow(ret)),
                    Site = c("Argilla Rd.", "Argilla Rd.",
                             "Rough Meadows", "Rough Meadows"),
                    names=rownames(ret),
                    estimate = c(coef(a), coef(r)), ret)

  ret
})

#and the output is a list, so, turn it into a data frame
params <- ldply(params)

ggplot(data=params,
       mapping=aes(x=year, y=estimate,
                   ymin=X2.5.., ymax=X97.5.., colour=Site)) +
  geom_pointrange() +
  facet_grid(~names)
```



```
The mode and variability of the maximum values at Argila Rd.
drops over time.
```

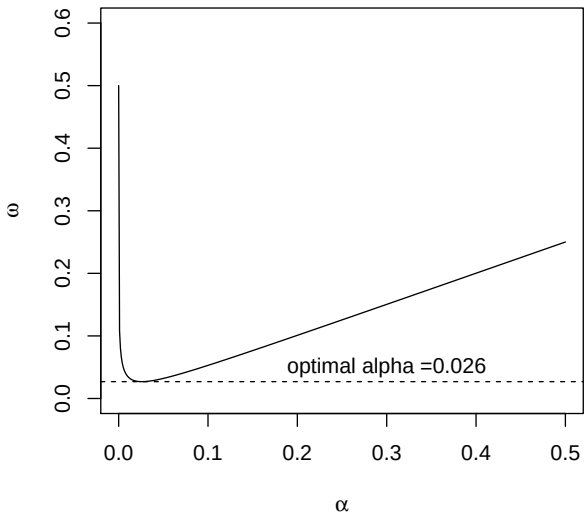
3 Power and α

In their 2012 paper, *S*etting an Optimal α That Minimizes Errors in Null Hypothesis Significance Tests (`http://dx.doi.org/10.1371/journal.pone.0032734`), Mudge et al outline a procedure where one uses both the type I and type II error rate to calculate a third quantity, ω. For any data set, we can calculate β given α, a sample size, a measure of effect size for an estimated parameter that we deem critical, and variation as measured in our data. Once we have obtained α and β, we can calculate ω as

$$\omega = \frac{\alpha + \beta}{2}$$

and then plot a curve of the relationship between α and ω. The value of α at the minimum value of ω is the 'optimal α' that balances type I and type II error against one another. For example, here's a plot of α versus omega for one particular test with a dashed line at the minimum value of ω to highlight the optimal value of alpha.

```
n<-100
set.seed(698)
z<-rnorm(n, 2, 5)

alpha <- seq(0,0.5, 0.001)
beta <- sapply(alpha, function(x) 1-power.t.test(n, 2, sd(z),
                                                 sig.level=x,
                                                 alternative="two.sided",
                                                 type="one.sample")$power)
omega = (alpha+beta)/2

plot(omega ~ alpha, type="l",
     xlab=expression(alpha), ylab=expression(omega),
     ylim=c(0,0.6))

#the baseline
abline(h=min(omega), lty=2)
text(0.3, 0.05, paste("optimal alpha =",
                      alpha[which(omega==min(omega))], sep=""))
```



The other great property of this is that we can calculate this optimal alpha after sampling our data. We can use the variation observed in our data in the calculation of power. Only the effect size, sample size, and α levels need to be specified *a* priori.

Let's assume you're interested in testing whether the observed temperature anomoly (the difference from the long-term average) around the globe is different from 0. To appease critics, you're assessment of a critical effect size is 1.5 degrees C. You know from looking at all of your observed temperatures that the standard deviation from temperatures across the globe is 5 degrees C. Using simulation to calculate β, what is your optimal alpha for 100 samples? How does your optimal alpha change with sample sizes from 10 to 1000? How does this relationship change if the standard deviation across all of the temperature sensors was 10 degrees C? Note, using functions to help you avoid heavy lifting are going to be pretty key here.

```
First, let we will make a function that calculates beta for a t test
with a given effect size, sample size, and SD using 1000 simulations.
```

```
# a function to calculate beta given a number of alphas

# a function to run a t test and extract it's p value
pFromT <- function(...) t.test(...)$p.value
```

```r
# a function that, given a vector of
# p values and an alpha, gets the type II error rate
beta <- function(pvec, alpha=0.05) sum(pvec>alpha)/length(pvec)

# a function that will get beta  for a
# t-test given a critical effect size, sd,
# sample size, and alpha using simulation
beta.t.test.sim <- function (effect.size = 1.5, temp.sd = 5,
                       n=100, n.sims=1000, alpha=0.05) {
  #get beta from running n.sims of distributions
  # and getting p from a t test using the preset alpha
  beta(sapply(1:n.sims, function(x) pFromT( rnorm(n,
                                                  mean=effect.size,
                                                  sd = temp.sd ),
                                            alternative="two.sided")), alpha=alpha)
}

#the baseline power
beta.t.test.sim()
```

```
## [1] 0.147
```

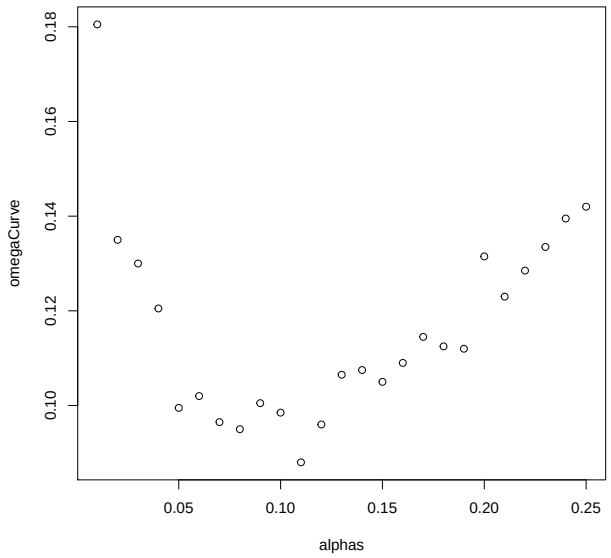
```r
#compare it to the equation version, just for confirmation
1-power.t.test(100, 1.5, sd=5, alternative="two.sided", type="one.sample")$power
```

```
## [1] 0.1561
```

Now that we have the above infrastructure, we can write a loop to calculate omega over a range of alpha values

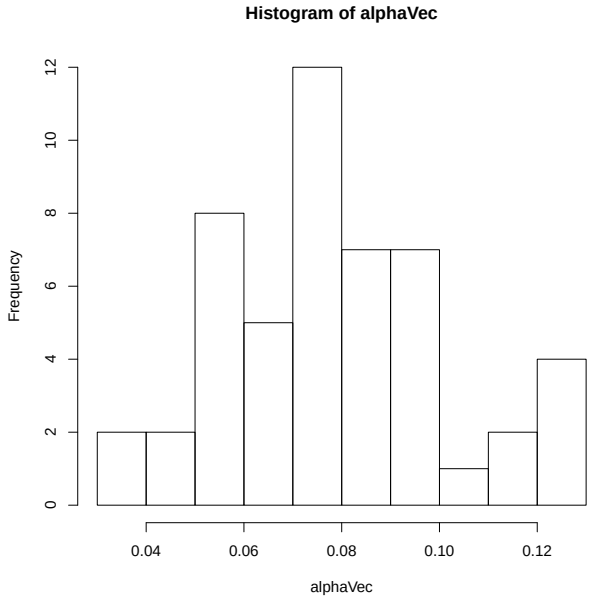
```r
# a function to calculate omega
omega <- function(alpha, beta) (alpha+beta)/2

#a vector of alphas to iterate over
alphas<-seq(0.01, 0.25, .01)

# use sapply to iterate over a bunch of alphas
# then use the simulation function to obtain beta
# and calculate alpha
omegaCurve <- sapply(alphas, function(a)
            omega(a, beta.t.test.sim(n.sims=1000, alpha=a)))

#plot the result
```

```r
plot(omegaCurve ~ alphas, type="p")
```



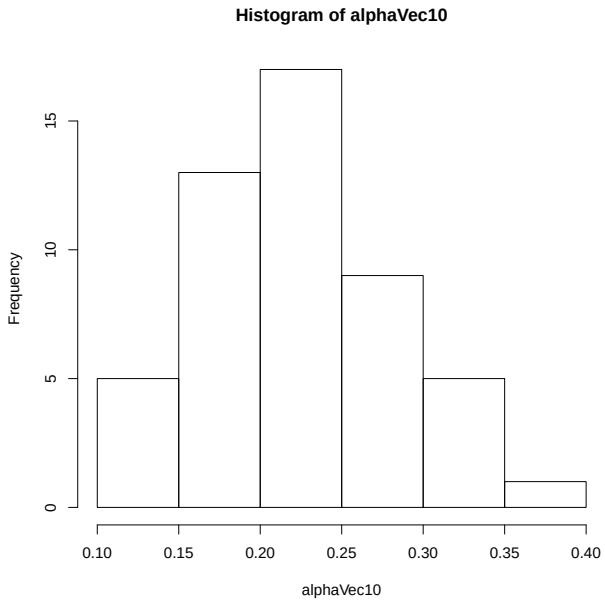

```r
#not smooth, but, let's see what the answer is...
alphas[which(omegaCurve == min(omegaCurve))]

## [1] 0.11


# one solution is to do this a number of time,
# get a dist for alpha, and use that as our optimal alpha
getOptimalAlpha <- function(alphas = seq(0.035, 0.15, .005),
                            n.sims=1000, n=100, sd=5){

  #use the omega vector method again
  omegaVec <- sapply(alphas, function(a)
    omega(a, beta.t.test.sim(n.sims=n.sims, n=n, alpha=a, temp.sd=sd)))

  #use omega to get the optimal alpha
  ret <- alphas[which(omegaVec == min(omegaVec))]

  #using min, as sometimes multiple values are returned.
  #we'll be generous, and take the smallest value
  return(min(ret))
}
```

```r
#so, what is our average optimal alpha? Let's get 5 to see...
set.seed(697)
alphaVec <- sapply(1:50, function(x) getOptimalAlpha(n.sims=100), simplify=T)

hist(alphaVec)
```




```r
mean(alphaVec)

## [1] 0.0805

median(alphaVec)

## [1] 0.0775
```

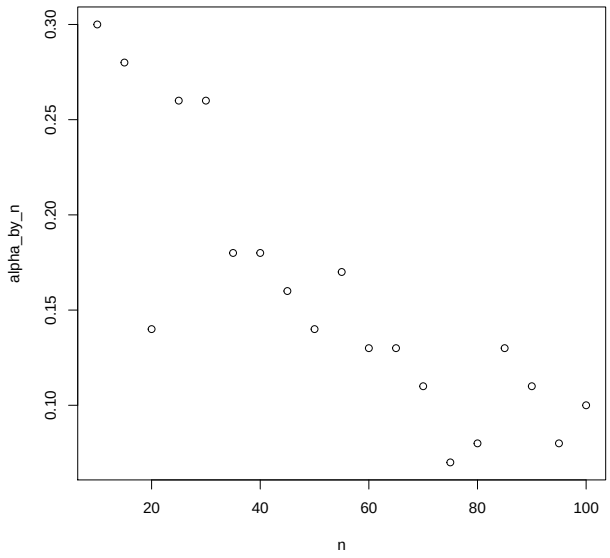
```r
#What if the SD was 10?
set.seed(697)
alphaVec10 <- sapply(1:50, function(x)
  getOptimalAlpha(alphas = seq(0.01, 0.4, .01), n.sims=100, sd=10), simplify=T)

#plot results
hist(alphaVec10)
```




```
mean(alphaVec10)

## [1] 0.229

median(alphaVec10)

## [1] 0.225
```

```
In both of the above examples, the mean and median
largely agreed, so, we will use them.  We can see that
the optimal alpha moves up as the SD increases.
```

```
# How would it change with sample size 10:100?
n <- seq(10,100,5)
set.seed(697)

# We'll use sapply with a single set of alphas, although,
# we could have done this and gotten an average of, say,
# 5 samples per alpha to get a smoother result
alpha_by_n <- sapply(n, function(x)
  getOptimalAlpha(alphas = seq(0.01, 0.4, .01), n.sims=200, n=x))

# Visualize the effect
plot(alpha_by_n ~ n)
```




```
#analyze the simulations
summary(lm(alpha_by_n ~ n))

##
## Call:
## lm(formula = alpha_by_n ~ n)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.09223 -0.02005 -0.00733  0.03566  0.04886
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.274404   0.019533   14.05  8.7e-11 ***
## n           -0.002109   0.000318   -6.63  4.2e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.038 on 17 degrees of freedom
## Multiple R-squared: 0.721,Adjusted R-squared: 0.705
## F-statistic:   44 on 1 and 17 DF,  p-value: 4.23e-06
```

```
As n goes up, the optimal alpha goes down.
```

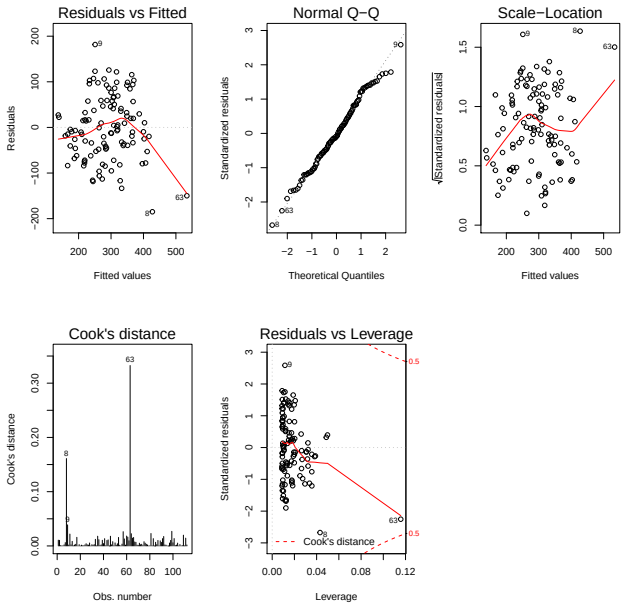
4 Regression and Simulation

In Maestre and Reynold's 2006 paper, they examine the effect of species diversity on the root:shoot ratio of biomass in plants. As good scientists, they deposited their data at Dryad at `http://datadryad.org/resource/doi:10.5061/dryad.3488j`. Is there a general relationship between aboveground and belowground biomass in their data set? Evaluate this relationship. Visualize the fit and prediction confidence intervals. Next, visualize the fit and prediction confidence intervals using simulation - i.e., for each simulated line, draw values for each coefficient using a normal distribution with the coefficients' means and SEs. You may want to use separate figures to show simulations just incorporating fit error versus prediction error. Also, make sure to overlay the best fit line on top so that we can tell what is our fit line versus what are the simulations used to show error. You may need many simulated draws to accurately show error.

Note, to get simulated residual standard error values, you need to use an inverse chisqaure distribution. So, here's an example where n is the sample size and est.se is the residual standard error, extracted from the summary of the linear model (you'll need to do that a bit here, or use `vcov` on the lm object to get the parameter variance), to get one random draw of a residual standard error -

```
df <- n-1
X <- rchisq(1, df=df)
ses <- est.se * sqrt(df/X)
```

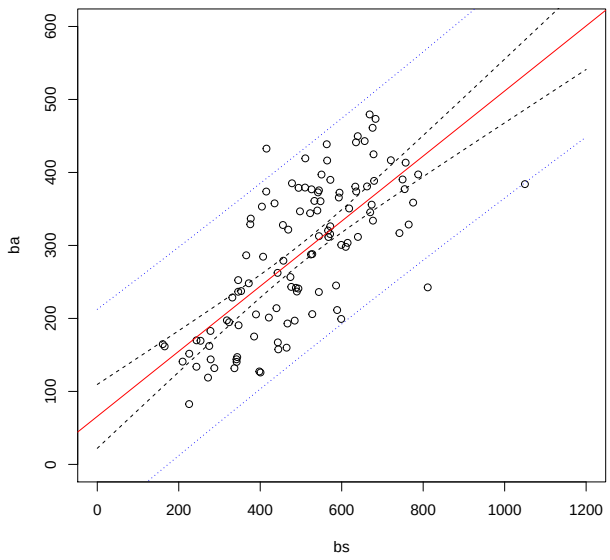
```
#First, a linear fit
#load in the data
biodiv <- read.delim("./data/Maestre_Oecol151.txt")

#fit a linear model to it
biomassLM <- lm(ba ~ bs, data=biodiv)

#check our assumptions
par(mfrow=c(2,3))
plot(biomassLM, which=1:5)
```



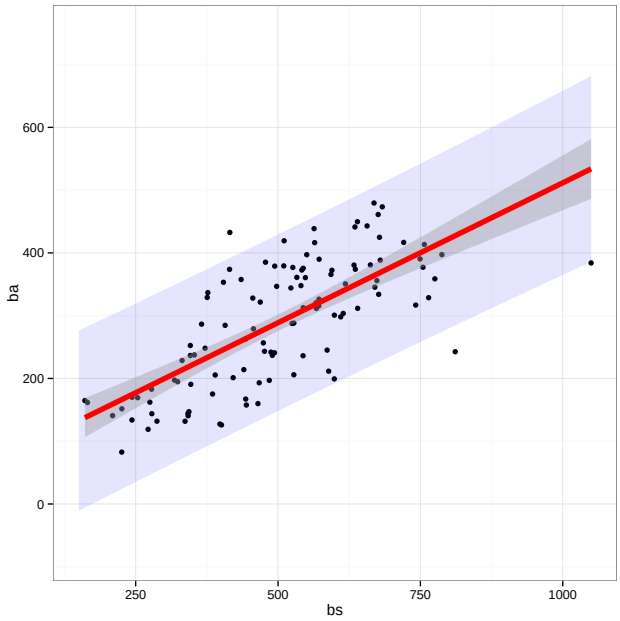

```
par(mfrow=c(1,1))

#it looks good, so let's test it all with an ANOVA
#as well as look at the coefficients
anova(biomassLM)

## Analysis of Variance Table
##
## Response: ba
##            Df Sum Sq Mean Sq F value Pr(>F)
## bs          1 561521  561521     112 <2e-16 ***
## Residuals 110 549207    4993
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary(biomassLM)

##
## Call:
## lm(formula = ba ~ bs, data = biodiv)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -184.9  -46.4   -5.9   52.0  181.7
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   65.797     22.076    2.98   0.0035 **
## bs             0.446      0.042   10.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.7 on 110 degrees of freedom
## Multiple R-squared: 0.506,Adjusted R-squared: 0.501
## F-statistic:  112 on 1 and 110 DF,  p-value: <2e-16
```

```
#So, there's a relationship.  Let's visualize it
#### Fit and Prediction via Direct Calculation

#first, the data points and fit line
plot(ba ~ bs, data=biodiv,  xlim=c(0, 1200), ylim=c(0,600))
abline(biomassLM, col="red")

# Get predicted fits over a wide range
# along with the CI of the fits
x<-seq(0,1200)
predFit <- predict(biomassLM, data.frame(bs = x), interval="confidence")
matlines(x, predFit[,2:3], col="black", lty=2)

# Now get the confidence interval of the prediction
# over a wide range and add it to the plot
predConf <- predict(biomassLM, data.frame(bs = x), interval="prediction")
matlines(x, predConf[,2:3], col="blue", lty=3)
```

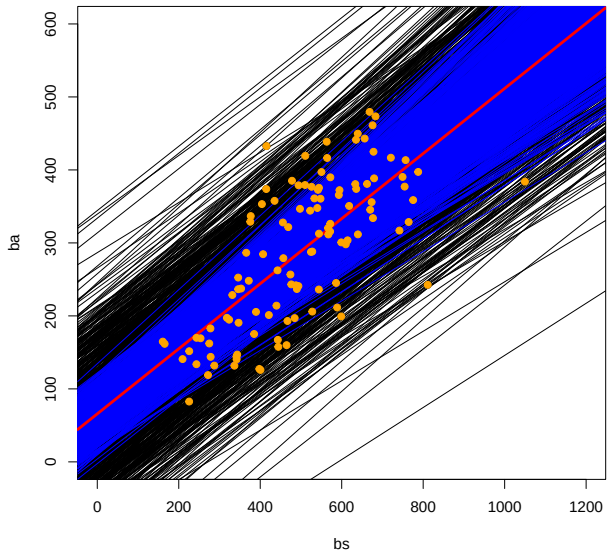



```
# OR, do the same with ggplot2
library(ggplot2)
predConf <- as.data.frame(predConf)
predConf$bs <- x

qplot(bs, ba, data=biodiv) +
    geom_ribbon(data=data.frame(predConf),
                mapping=aes(x=bs, y=fit, ymin=lwr, ymax=upr),
                alpha=0.1, fill="blue")+
    theme_bw() +
    stat_smooth(method="lm", color="red", lwd=2) +
    xlim(c(150,1050))
```




```
####SIMULATED LINES
n.sims <- 1000

#get coefficient draws for the slope and intercept
int <- rnorm(n.sims, coef(biomassLM)[1], sqrt(vcov(biomassLM)[1,1]))
slope <- rnorm(n.sims, coef(biomassLM)[2], sqrt(vcov(biomassLM)[2,2]))

#get draws of the population residual standard deviation
df <- nrow(biodiv)-1
est.se <- summary(biomassLM)$sigma
  X <- rchisq(n.sims, df=df)
ses <- est.se * sqrt(df/X)
e<-rnorm(n.sims, 0, ses)

#Setup the plot
plot(ba ~ bs, data=biodiv,  xlim=c(0, 1200), ylim=c(0,600))

#plot the prediction CI
for(i in 1:n.sims) abline(a=int[i]+e[i], b=slope[i], col="black")

#plot the fit CI over top
for(i in 1:n.sims) abline(a=int[i], b=slope[i], col="blue")

#add the fit line
```

```r
abline(biomassLM, lwd=3, col="red")

#and lastly the points
points(biodiv$bs, biodiv$ba, pch=19, col="orange")
```

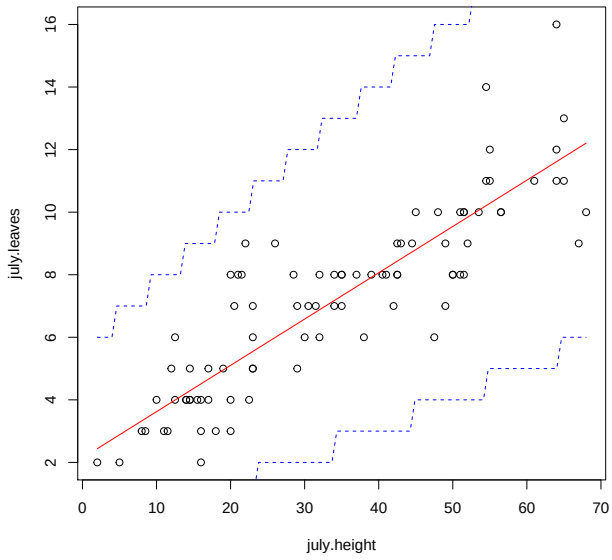



```r
#It' pretty, isn't it?
```

5 Mutualism and Likelihood

In their 2011 paper, Stanton-Geddes and Anderson assessed the role of a facultative mutualism between a legume and soil rhizobia in limiting the plant's range. After publishing, they deposited their data at Dryad `http://datadryad.org/resource/doi:10.5061/dryad.8566`. As part of their field experiment, they looked at a variety of plant properties in the field. One of interest to us is the relationship between plant height and number of leaves in July. Examine the relationship using likelihood with your choice of error distribution and a linear function. Why did you chose this error distribution? Plot the fitted curve along with fit and prediction error. Is there another distribution you could have used? Why? How do results from your model fit compare to those of another error distribution? What does this tell you about the relationship between plant height and number of leaves?

```r
#read in the data
mutual<-read.csv("./data/field_inoculation_expt_2009.csv", na.strings=".")

#clean it up so we have no NAs, as this will be a problem later
mutual<-with(mutual, data.frame(july.leaves = july.leaves, july.height = july.height))
mutual <- na.omit(mutual)

#let's just plot the data
plot(july.leaves ~ july.height, data=mutual)

# There are several distributions that could describe this linear fit.
# But, as this is count data (# of leaves) let's start with a Poisson
# distribution.

library(bbmle)
pois_fit<-mle2(july.leaves ~ dpois(lambda = a*july.height + b),
               start=list(a=1, b=5), data=mutual)

#add a fit to the plot based on the poisson
pline <- function(x) coef(pois_fit)[1]*x + coef(pois_fit)[2]
curve(pline, add=T, col="red")

#add the prediction error
pred_line <- function(x, q) qpois(q, coef(pois_fit)[1]*x + coef(pois_fit)[2])
curve(pred_line(x,0.975), add=T, col="blue", lty=2)
curve(pred_line(x,0.025), add=T, col="blue", lty=2)
```




```
The linear poisson fit above may be the best fit, but,
1) Maybe the whole thing is overdispersed and a Negative
Binomial error may be better or 2) Actually...the variance may
not increase with the mean.  Let us fit the other two models
and then compare them with a LR Chis-Squre test.
```

```
#the normal fit
norm_fit <- mle2(july.leaves ~ dnorm(mean = a*july.height + b, sd=sd),
                 start=list(a=1, b=5, sd=5), data=mutual)

# The negative binomial fit.  Note, this was
# not well behaved, so, I used SANN in the end
nb_fit<-mle2(july.leaves ~ dnbinom(mu = a*july.height + b, size=z),
             start=list(a=1, b=5, z=2), data=mutual, method="SANN")


#first comparison - Negative binomial v. Poisson
anova(nb_fit, pois_fit)

## Likelihood Ratio Tests
## Model 1: nb_fit, july.leaves~dnbinom(mu=a*july.height+
##           b,size=z)
## Model 2: pois_fit, july.leaves~dpois(lambda=a*july.height+
##           b)
```

```
##   Tot Df Deviance Chisq Df Pr(>Chisq)
## 1      3      377
## 2      2      355  22.2  1    2.5e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


#Poisson is better, so, normal v. Poisson
anova(norm_fit, pois_fit)

## Likelihood Ratio Tests
## Model 1: norm_fit, july.leaves~dnorm(mean=a*july.height+
##           b,sd=sd)
## Model 2: pois_fit, july.leaves~dpois(lambda=a*july.height+
##           b)
##   Tot Df Deviance Chisq Df Pr(>Chisq)
## 1      3      313
## 2      2      355  42.3  1    7.9e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


# The normal fit is still the better one, despite the
# data being count data.  The normal fit has a much lower
# deviance and produces a better fit.  Thus, while it may
# be count data, at this range, a Normal error may be a
# better approximation.
```

6 Extra Credit

Hubway, the Boston based bike rental company, is releasing all of their trip data. The data set is huge - about 60 MB. They're also providing lat and long information for all stations. They are hosting a data visualization challenge at `http://hubwaydatachallenge.org`. For your extra credit, find and visualize something interesting in the data. Note, ggplot and it's map geom might come in handy (or not). If you also want to play with breaking down and analyzing data using different groupings, you may want to look into the plyr library at `http://plyr.had.co.nz/` and available on CRAN. We'll be using plyr later in the course, but, it might be useful for exploring the data.

Extra points for each interesting or surprising thing you find. And, heck, if you get into this, enter the challenge!