PCB 6468 - Methods in Experimental Ecology Spring 2018 Semester
Dr. Pedro F. Quintana-Ascencio

R Demonstration – Frameworks for Statistical Analysis

Objective: The purpose of this session is to use R to compare the density of ant nests in two different habitats using a frequentist approach (via a parametric analysis of variance, or ANOVA), a Monte Carlo approach (via a simple randomization test), and a Bayesian approach.

Part I. Loading the Data Into R

The tab-delimited text file `Ant_Nest_Data.txt` contains the ant nest density dataset shown in **Table 5.1** of Gotelli & Ellison (2004, p. 108). Browse to the course website and right-click on this file to save it to the `PCB6466` folder on your Desktop.

Now start R and type the following commands to load the ant nest dataset and attach it:

```
> file_name <- "Ant_Nest_Data.txt"
> nest_data <- read.table(file_name, header=T)
> attach(nest_data)
```

You can calculate the mean nest density for each habitat using the following code to filter each observation by its habitat type, either “Forest” or “Field”:

```
> mean(NestsPerQuadrat[Habitat=="Forest"])
[1] 7
> mean(NestsPerQuadrat[Habitat=="Field"])
[1] 10.75
```

The mean density of ant nests in the sample from the “Field” habitat was 10.75 and the mean of the sample from the “Forest” habitat was 7. In the next three parts of this lesson, we will use frequentist, Bayesian, and Monte Carlo approaches to determine whether these mean nest densities are significantly different from each other.

Part II. Parametric Approach: Analysis of Variance (Frequentitst Approach)

Analysis of variance (ANOVA) is a common parametric test used when your predictor variable (or variables) is categorical and your response variable is continuous. In Methods I you already learned about ANOVA in detail. You can also review Chapter 10 of the Gotelli & Ellison (2004) textbook. While we will gloss over most of the details in the following ANOVA demonstration, you should recognize that ANOVA is a reasonable statistical method to test for differences in the group means of our ant nest data.

ANOVA belongs to a whole family of statistical tests known as *linear models*, which also includes regression and analysis of covariance (ANCOVA). The R function `lm` is used to fit linear models, and we will be working with this function extensively starting with our discussion of the topic of regression (Chapter 9 in Gotelli & Ellison). For now, however, we will limit our use of the `lm` function to specifying a simple model for our ANOVA:

```
> model <- lm(NestsPerQuadrat ~ Habitat)
```

This command creates a simple linear model that relates our response variable (`NestsPerQuadrat`) to our categorical predictor variable (`Habitat`) and stores the result in a new variable named `model`. The tilde character (~) is a special R operator used in the specification of models. Once we have defined our model, it is a simple matter to use the built-in `anova` function to perform an analysis of variance on our data:

```
> anova(model)
Analysis of Variance Table

Response: NestsPerQuadrat
          Df Sum Sq Mean Sq F value  Pr(>F)
Habitat    1 33.750  33.750  8.7805 0.01806 *
Residuals  8 30.750   3.844
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

You should be familiar with details of this output (which is known as an *ANOVA table*) after you previous course. For now, it is important to note that the F-ratio of approximately 8.78 is the same as that reported in Gotelli & Ellison (2004, p. 119), as is the P-value of approximately 0.018 (see output highlighted in red above).

<u>Part III. Monte Carlo Approach: Randomization Test</u>

As mentioned in class, one of the major disadvantages of ANOVA and many other parametric analyses is that these tests assume that the data are sampled from normal distributions. Yet, an examination of the box plots for the nest density data (grouped by habitat type) suggest that these data may not be normally distributed. Verify this for yourself using the `boxplot` function.

One of the major advantages of the Monte Carlo approach to statistical analysis is that it doesn’t require you to assume that the data are sampled from a normal distribution (or any other specific probability distribution, for that matter). It only assumes that the data are drawn from random, independent samples (Gotelli & Ellison 2004, p. 115).

Another assumption of the Monte Carlo method is that the test statistic adequately represents the pattern we are interested in testing. For our ant nest density example, we define our test statistic as the absolute difference between the means of the “Forest” and “Field” samples. We can calculate and display the observed value of this test statistic with the following lines of code:

```
> mean_forest <- mean(NestsPerQuadrat[Habitat=="Forest"])
> mean_field <- mean(NestsPerQuadrat[Habitat=="Field"])
> diff_obs <- abs(mean_forest - mean_field)
> diff_obs
[1] 3.75
```

Our observed test statistic value of 3.75 matches that reported for $\texttt{DIF}_{\texttt{obs}}$ on p. 111 of Gotelli & Ellison (2004).

Now we need to construct a “null distribution” of our test statistic by reshuffling the habitat labels (“Field” or “Forest”) and then randomly reassigning them to the nest density observations. The following line of code uses the *`sample`* function (without replacement) to randomly re-order the habitat labels:

```
> Habitat_random <- sample(Habitat, length(Habitat))
```

Next, the following pair of commands will create a new data frame object, `random_data`, that contains the original nest density observations, but with the randomly shuffled habitat labels assigned to them:

```
> Nests_random <- NestsPerQuadrat
> random_data <- data.frame(Habitat_random, Nests_random)
```

We can compare this to the original data in **Table 5.1** (Gotelli & Ellison, p. 108) and verify that the labels have indeed been randomized (NOTE: Due to random sampling, your data will most likely be different than that shown here. There is also a very small chance that your data may have the same labels as the original data in Table 5.1):

```
> random_data
   Habitat_random Nests_random
1          Forest            9
2          Forest            6
3          Forest            4
4          Forest            6
5           Field            7
6          Forest           10
7           Field           12
8           Field            9
9          Forest           12
10          Field           10
```

Next, we will calculate our test statistic again by taking the absolute value of the difference between the mean density in the “Forest” and “Field” habitats (the value you will obtain may differ from the realization in the example):

```
> mean_forest <- mean(Nests_random[Habitat_random=="Forest"])
> mean_field <- mean(Nests_random[Habitat_random=="Field"])
> abs(mean_forest - mean_field)
[1] 1.666667
```

To build our null distribution, however, we need to repeat this process many times, usually 5000 or more. Doing this by hand would be extremely time-consuming and tedious, but we can use a *`for`* loop inside an R script to do all the hard work for us. Download the `Ant_Nest_Density.R` script from the course website and save it to your `PCB6466` folder. Open the script and examine the block of code at lines after "## Part I Monte Carlo Analysis":

```
## create an empty array to hold the test statistic for each iteration
iterations <- 5000
diffs <- numeric(iterations)

## for each iteration, randomize the data and compute new test statistic
for (i in 1:iterations) {

       ## randomize the nest data
       Habitat_random <- sample(Habitat, length(Habitat))
       Nests_random <- NestsPerQuadrat
       random_data <- data.frame(Habitat_random, Nests_random)

       ## compute the group means for the randomized data
       mean_forest <-
       mean(Nests_random[Habitat_random=="Forest"]) mean_field <-
       mean(Nests_random[Habitat_random=="Field"])

       ## compute and save the test statistic
       diffs[i] <- abs(mean_forest - mean_field)
}
```

This block of code creates a null distribution for our test statistic of size `iterations` (which is defined earlier in the script and is initially set to a value of 5000) and stores it in the vector `diffs`. The following lines of code will display the distribution of our test statistic as a histogram and compute and display the tail probability, `P_Mc`:

```
## show a histogram of the test statistic
hist(diffs, xlab="DIF")
abline(v=diff_obs, col="red")

## calculate the tail probability
P_Mc <- length(diffs[diffs >= diff_obs])/iterations
P_Mc
```

If you run the script several times (e.g., by choosing *`Edit◊Run all`* from the R menu), you will get different values reported for `P_Mc`, the tail probability. As noted in the lecture, one of the main objections to the Monte Carlo approach is that different analyses of the same dataset can lead to slightly different statistical results. But as the number of iterations approaches infinity, the tail probability will converge on a single number. Verify this yourself by, for example, changing the `iterations` variable to a value of 10000 and then running the script several times. What happens to the mean and variance of the computed tail probability values?

Part IV Bayesian Analysis.

Bayesian analysis allows us to quantify the probability of a hypothesis. We want to determine the probability of the hypothesis given the data. The hypothesis needs to be specific and needs to be quantitative. For example, we can evaluate the probability of the difference in the number of nests being larger than 3.

P(difference > 3 | estimated difference from the data)

We start by modifying the variables to enter them in jags. The number of nest per quadrat is in the proper format, but we need to change the format of the label of the forest types. We will change “forest” = 0 and “field” = 1. We also define the total sample size “n”.

```
y <- NestsPerQuadrat
x <- c(0,0,0,0,0,0,1,1,1,1)
n <- 10
```

To reach the interface of jags you will need to manually load the packages `coda` and `R2OpenBUGS` and call the packages using the function “library”.

```
library(rjags)
library(coda)
```

We will define the Bayesian model in `jags` as described below (see files: ants_jags1.R and ants.jags2.R. Please inspect the definition of the priors for the parameters of the model. We will also need to define the likelihood function for the model. When x = 0, the model calculates the distribution of the data for forest. We use one derived quantity to estimate the distribution for the field (when x = 1). Do the math and confirm that these arguments make sense.

```
model {


  #Priors
  mu1 ~ dnorm(0,0.001) #1/1000
  delta ~ dnorm(0,0.001)
  #delta ~ dnorm(3,1/(2^2)) #remember 1/variance
  tau <- 1/(sigma*sigma)
  sigma ~ dunif(0,10)
  #Likelihood
  for (i in 1:n)
  {
    y[i]~ dnorm(mu[i],tau)
    mu[i] <- mu1 + delta*x[i]
    residual[i] <- y[i]-mu[i]
  }
  # Derived quantities
  mu2 <- mu1 + delta
}
```

We decide if we want diffuse or informed priors. Initially be sure to use option = 1

```
option <- c("diffuse","informed")[1]
if (option =="diffuse") {model =  "ants_jags1.R"}
if (option =="informed") {model =  "ants_jags2.R"}
```

In the next steps we “read” the data and define the initial values of the parameters that are going to be used to start the simulation.

```
# Inits function
inits=list(
  list(mu1=rnorm(1), delta=rnorm(1), sigma=rlnorm(1)),
  list(mu1=rnorm(1), delta=rnorm(1), sigma=rlnorm(1)),
  list(mu1=rnorm(1), delta=rnorm(1), sigma=rlnorm(1))
)


# Bundle data
mean.data=list(
  y=y,
  x=x,
  n=n
)
```

We indicate the parameters that are going to be estimated. We also define the settings of the model. In this case we will use 3 chains, each with 10000 samples, and we will discard the first 100 values as burn in.

```
#Parameters to estimate
params<-c("mu1", "mu2", "delta", "sigma")

#MCMC settings

nc <- 3       # Number of chains
ni <- 10000   # Number of draws from posterior for each chain
nb <- 100     # Number of draws to discard as burn-in
nt <- 10      # Thinning rate
```

Finally, we will use the function “jags” to run the program. Notice that it includes all the arguments that we defined above.

```
# MCMC settings, start Gibbs sampler and plot results and diagnostics
jm=jags.model(model,data=mean.data,inits=inits,n.chains=3,n.adapt=5000)
update(jm, n.iter=10000)
zc=coda.samples(jm,variable.names=params,n.iter=10000)

plot(zc)
summary(zc)
R11<- summary(zc)
obs

mu1.est <-zc[[1]][,1]
```

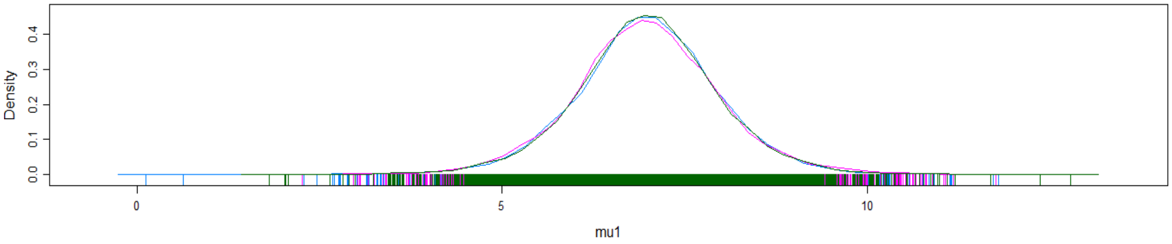
The function `coda.samples` will return two data tables of your parameters, one showing mean and standard deviation for each variable; the other showing quantiles for each variable (remember individual results may be slightly different):

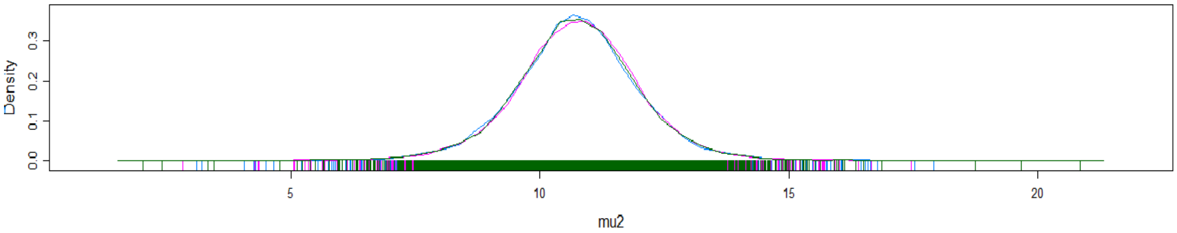
```
Iterations = 15001:25000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean     SD Naive SE Time-series SE
delta  3.732 1.5509 0.008954       0.013575
mu1    7.011 0.9876 0.005702       0.008818
mu2   10.742 1.2105 0.006989       0.006877
sigma  2.344 0.7386 0.004264       0.008934

2. Quantiles for each variable:

        2.5%    25%    50%    75%  97.5%
delta 0.5913  2.793  3.733  4.682  6.794
mu1   5.0581  6.406  7.010  7.607  8.996
mu2   8.3183 10.010 10.744 11.482 13.158
sigma 1.3896  1.836  2.193  2.671  4.167
```

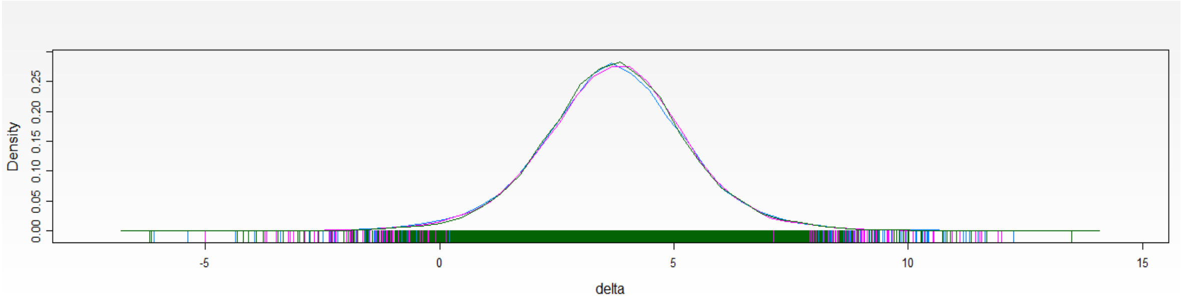
Notice the value of `delta` is the same or very close as when we did the MC analysis, but this time we assume that it is a distribution; therefore it also has an SD associated with it. `Deviance` (-2 times the log-likelihood ratio of the reduced model minus the likelihood of the full model) is used to compare two models, here the reference model (full model) has the data fitted exactly (http://en.wikipedia.org/wiki/Deviance_statistics). The means for forest (mu1) and field (mu2) are virtually the same as when we calculated them with a frequentist approach, but again, now they are distributions and not simple values.

We can use the credibility percentiles and given the Bayesian estimate of mean difference of 3.75, P(diff >2.785 | 3.75) is 0.75. We get that from looking at the value highlighted in red, which shows that 25% of the probability density distribution of delta is less than 2.79. In other words there is a probability of 0.75 that ant nest densities between the two habitats are different by at least 2.79 nests. The actual probability for > 3 nests can be calculated but for now, this is good.

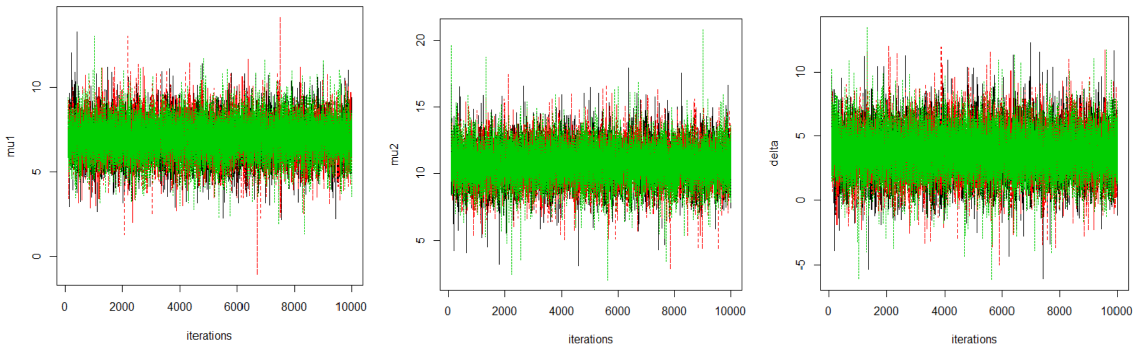
We can also generate several diagnostic plots for our Bayesian analysis. Below, we create density plots for `delta`, mu1, mu2, and delta as examples. The function `plot(zc)` will show the probability density of the each of the three parameters.







We can also use the function traceplot to examine the history of each chain.



In the plot below, you can observe the distribution of the diffuse prior (flat line) and the posterior (in red). The mean of this distribution (vertical line in red) overlaps with that from the frequentist analysis (vertical line in blue).

Now change the option for the informed prior (option = 2). What happen?
Is the normally distributed prior a proper prior?

Do not forget to detach the data

```
detach(nest_data)
```