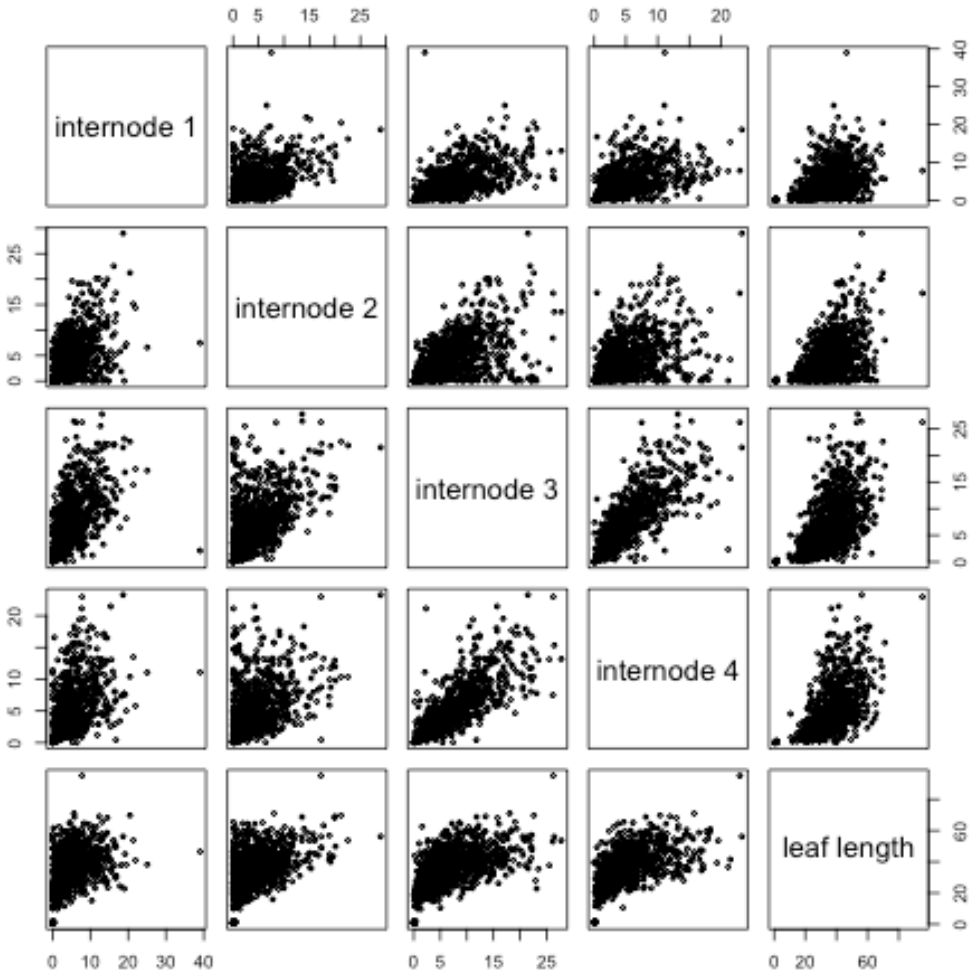
LAB # 3
Introduction to Statistical analysis with R

Julin Maloof
Department of Plant Biology
University of California, Davis
One Shields Ave
Davis, CA 95616
http://malooflab.openwetware.org/
+1 (530) 752 8077
jnmaloof@ucdavis.edu



Background

The need for statistics in biology

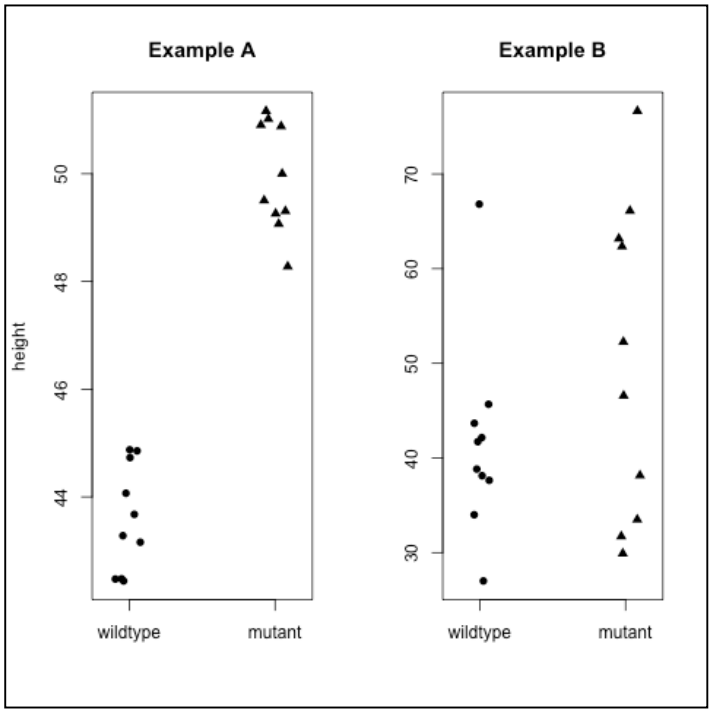
When I was a graduate student some professors were known to quip that if you needed to use statistics that the experiment wasn't worth doing. The days for such thinking are now long past us. Some fields of biology have a long history of using statistics (e.g. ecology, quantitative genetics) whereas others have been slower to embrace statistical methods (e.g. molecular biology). However all aspects of biology research are becoming more quantitative. Whether comparing a mutant plant population to wild-type, assessing different environmental treatments on plant physiology, or performing microarrays or other genomics experiments you undoubtedly will be faced with the need for statistics in your work.

Goals

One lab period, of course, does not provide nearly enough time for me to teach statistics. The goals for this lab are to briefly review (or introduce) some of the most common statistical tests and to introduce R, a powerful open source software package for statistical analysis and programming. If you have not already done so I strongly encourage you to take an introductory statistics class, or to work through an introductory book. Several good books are mentioned in the resources section.

Mean and variance: key concepts in statistics

In basic statistics, one common procedure is to determine how likely it is that an observed phenomenon could have occurred by random chance. Consider a scenario where you have a mutant and wild-type population of plants. You want to know whether your mutation affects plant growth, so you measure the height of 10 wild-type plants and find that their mean height is 43cm while the mean of 10 mutant plants is 50cm. How likely is it that you would have observed this difference of 7cm if the mutant truly has no effect on plant height? Stated another way: how likely is it that if you had measured 10 wild-type plants, and then measured a second set of 10 wild-type plants that you would have observed a difference of 7cm? Looking at example A in the figure to the left we intuitively feel that that wildtype is different from mutant, whereas in example B we are less certain. Why? Because



there is greater *variance* in the samples of example B. Several of the most common statistical tests formalize these concepts and compare the variance between group means to the variance within groups and we will cover those tests in this module.

The R statistical environment

We will use the open source software R for this lab. There are several reasons to use R. Since R is free you can take what you learn in this class and use it at your home institution without additional cost. R runs on Windows, Mac, and linux. R provides a robust, flexible, and extensible platform for analysis. R is a platform used by many statisticians and bio-statisticians to develop and release their analysis packages, so you have access to the latest developments. For example, see http://cran.r-project.org/web/views/Genetics.html and http://cran.r-project.org/web/views/Phylogenetics.html
We will cover R for genomics data and Bioconductor in a later lab.

RStudio

Although R comes with a built-in editor and graphical user interface, we will use the 3rd party application Rstudio: http://www.rstudio.org/ . Rstudio provides a nicer interface to R and provides a more uniform feel across platforms (Mac, Windows, Linux).

Online Resources

Rice Virtual Lab in Statistics: http://onlinestatbook.com/rvls/ provides an excellent, understandable online tutorial and resource in statsistics.

The R Project: http://www.r-project.org/

Downloads for R: http://cran.r-project.org/

Online R documentation: http://cran.r-project.org/other-docs.html

Recommended books

Michael J. Crawley. Statistics: An Introduction using R. Wiley, 2005. ISBN 0-470-02297-3

Peter Dalgaard. Introductory Statistics with R. Springer, 2nd edition, 2008. ISBN 978-0-387-79053-4

Larry Gonick and Woollcott Smith. The Cartoon Guide to Statistics. Collins Reference, 1993. ISBN 0062731025

Stanton Glantz. Primer of Biostatistics. McGraw-Hill Medical, 2005. ISBN 0071435093

Lab Work

Formatting conventions

Commands that you are to type into R will be indented, begin with the R prompt ">" and in **`bold courier`** font. The resulting output will be in normal (not bold) `courier` font. In many cases the output is not included in this manual. Do not type the leading ">".

```
> print("this illustrates the text used for R input and output")
[1] "this illustrates the text used for R input and output"
```

Many Windows keyboard shortcuts that begin by holding down the control (ctrl) key are instead initiated on the Mac with the command (cmd) key. Because some people may be using Macs, I will indicate such keys as ctrl/cmd-. For example, if I was explaining how to copy items I would write "first select the items and then press ctrl/cmd-C". This would indicate that Windows users would press ctrl-C and Mac users would press cmd-C

I. Introduction to R

R uses a command line interface. We will spend a few minutes to become familiar with basic R usage.

It is helpful to create a folder for each project that you do in R. Before starting R use Windows Explorer to create a new folder to contain the documents from today's lab. Find the "Maloof Stats_Lab" folder on the ftp server. Copy the entire contents into the folder that you just created on your computer.

Start RStudio by selecting RStudio from the Windows start menu (you may need to look under "All Programs"). The lower right pane of RStudio contains a file browser. Use that to navigate to the folder you created for today's lab. If you do not like the Rstudio file navigator you can also click on the "…" three period on the right-hand side to get a standard browser. Try both options. Once you have navigated to the correct folder for today, click on the "More" pulldown menu and select "Set as Working Directory". Now RStudio will by default read and write files to this directory.

When RStudio started, a "Console" window opened. It is possible to type commands directly into this window. You can try this by typing some simple arithmetic into the console window:

```
> 4 + 4
[1] 8
```

It is much better to keep a record of the commands that you have given R. This way you know what you have done and you can re-run analyses as needed. You can do this by creating "**script**" files using the RStudio editor. To start a script file choose "New> R Script" from the "File" pull-down menu or by typing

“ctrl/cmd-shift-N”. A new pane opens in the top left corner. Now you can type commands into the script editor. To send them to the console to be executed type ctrl/cmd-enter (aka ctrl/cmd-return). Ctrl/cmd-enter will send the current line or the current selection to the console. Retype "4 + 4" into the script editor and the press ctrl-R (mac users: command-enter).

As you saw, you can type arithmetic commands directly into R as if it were a calculator. More complex operations use **functions**. You can use functions to perform simple math as well. For example the sum function will add a series of numbers. The "#" symbol denotes a comment. Anything after # on a line is ignored. To get in the habit of using the script editor, be sure to type the following into the script editor and then use ctrl-R to send it to the console.

```
> sum(4,4) # same as 4 + 4
[1] 8
```

To save you from typing, all of the commands in this lab manual are in the script file "CSHL.R". Open this file in RStudio. Now instead of typing the commands you can just select them and use ctrl-R to execute them.

The information given between the parentheses when you called the sum function are the **arguments** to the function…the information that the function will operate on. Even if you are not passing any information to a function you still must include the parentheses for R to know that you want the function to be executed.

A colon “:” can be used to define a simple series.

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

EXERCISE 1: What is the sum of the numbers from 2000 to 20000?

Objects in R are used to store data, results, and even functions. To assign a value to an object use “<-“. To see the contents of an object, type its name.

```
> a <- 5
> b <- 2:20
> a
[1] 5
> b
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

You can list the objects in your workspace by using the ls() command. Objects can be removed using the rm() command.

```
> d <- "I just want to be deleted"
> d
[1] "I just want to be deleted"
> ls()
[1] "a" "b" "d"
> rm(d)
> ls()
[1] "a" "b"
> d
Error: object 'd' not found
```

EXERCISE 2: Add the contents of a and b together and place the results in a new object. Examine the result. Do you get different results using "sum" and "+"? If so, why?

The "c" command (for concatenate) can be used to make a list of values.

```
> numbers <- c(5,76,23)
> numbers
[1]  5 76 23
>
> animals <- c("dogs","cats","people")
> animals
[1] "dogs"   "cats"   "people"
```

Note that objects can hold single values single values or multiple values. Square brackets "[]" are used to extract subsets of data from an object.

```
> animals[2]
[1] "cats"
```

EXERCISE 3: What is the sum of the 5^{th} through 10^{th} element of the object b?

Objects can also be arranged in multiple dimensions. The matrix command makes 2-dimensional objects. When extracting from a multi-dimensional object you must specify both dimensions within the square brackets.

```
> m <- matrix(data=1:25,ncol=5,byrow=T)
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

```
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
[5,]   21   22   23   24   25
> m[5,5]
[1] 25
```

EXERCISE 4: When extracting from a 2-dimensional object, which number specifies rows and which specifies columns? What does the command m[3,] do? How can you extract the 3rd, 4th and 5th columns of m together as one object?

EXERCISE 5: Type "**`cbind(m,101:105)`**" . What does the cbind command do? Create a new object "n" where the ***first*** row is a new row of numbers (your choice) and the following rows are the followed by the m matrix. Extra credit: do the same but reverse the order of the rows from m.

II. Importing data into R

The easiest way to import data into R is to import a spreadsheet that has been saved in comma-separated values (.csv) format. .csv is available in Excel and other common spreadsheet applications. Tab or text delimited spreadsheets are also easily imported.

We will work with data in the file "TomatoR2CSHL.csv" which contains data from a pilot experiment measuring the growth of various wild tomato accessions under different light conditions.

Explanation of column headings:
-shelf, flat, col, row: information about where each plant was grown
-acs: the accession number for that strain
-trt: "H" is light with a high red:far-red ratio; simulated sun. "L" is light with a low red:far-red ratio; simulated plant shade.
-days: days since germination
-date: date of measurement
-hyp: hypocotyl length
-int1-int4, intleng: individual and total internode lengths
-totleng: length of main shoot
-leafnum: which leaf was measured
-petleng: petiole length
-leafleng: leaf length
-leafwid: leaf width
-ndvi: a measure of plant vegetation density

-lat: latitude of origin
-long: longitude of origin
-alt: altitude of origin
-species: species of plant
-who: who measured the plants

import the data into R using the read.csv() function:

```
> data <- read.csv("TomatoR2CSHL.csv")
```

You can get a quick summary of an object using the summary() function. The first few lines of the object can be examined using head() and the last few lines using tail(). How would you look at the entire object?

```
> summary(data)
> head(data)
> tail(data)
```

You should **always** look at your data after importing to make sure that it imported correctly.

III. Exploratory data analysis
Before performing statistical tests you will likely want to examine your data. We can use both graphical methods and summary statistics. The mean() median() and sd() functions calculate the mean, median, and standard deviation. For data that is in a special kind of object, called a data frame, we can access columns of data by using "$" to select the column. Luckily, the output of read.csv() is a data frame.

```
> data$hyp
```

EXERCISE 6: What are the mean, median, and standard deviation of the second internode? hint1: you can get a list of column names by using the names() command. hint2: by default R will return "NA" if any of the data is missing. Get help on the mean function by typing

```
> ?mean
```

at the command line and try to figure out how to force the calculations to be done.

As you know, this experiment involved two different light treatments and several different species. So you might want to know the mean of each species separately. You can use either the by() or tapply() functions. The form for the functions is by(ARG1, ARG2, ARG3, ARG4) where ARG1 is the data that you want to calculate means for; ARG2 contains information on how you want to split the

data up, ARG3 gives the function that you want to apply and ARG4 is additional arguments to be passed to ARG3.

For example: this example is to help you with both ex. 6 and 8

```
> fruit.diameter <- c(4,3,6,4,5,4,5,5,NA,6)
> fruit.type <- rep(c("apple","orange"),5)
#look at what we have created
> fruit.diameter
> fruit.type
#if we want to calculate the standard deviation for each fruit type:
> tapply(fruit.diameter,fruit.type,sd,na.rm=T)
#or
> by(fruit.diameter,fruit.type,sd,na.rm=T)
```

EXERCISE 7: What does the rep() function do?

EXERCISE 8: Which species has the longest hypocotyls? Which light treatment causes longer hypocotyls? Create a new object to hold means calculated using the tapply function. Plot the means using barplot(HYP.MEAN) where you replace HYP.MEAN with the name of the object holding your means.

You can actually have tapply() or by() split your data by more than one variable if ARG2 takes the form of list(SPLIT1,SPLIT2) where SPLIT1 and SPLIT2 each contain grouping information.

EXERCISE 9: Calculate mean total internode length for each species/light combination. Do you prefer the output from tapply() or by() better?

It can also be helpful to graphically examine your data. One common display type is the histogram. In R this is provided with the hist() function.

```
> hist(data$hyp)
> hist(data$hyp,main="tomato hypocotyls")
```

EXERCISE 10: What is the difference in the two plots produced above? Examine the help page for the hist function. Can you figure out how to color your bars blue and to add better x-axis labels to the plot?

But what if you want a histogram for each species or light treatment? Additional R functions can be loaded from add-on packages. The lattice library contains functions for making multiple plots from grouped data. The histogram() function in this library allows us to plot multiple histograms.

```
> library(lattice)
> histogram(~data$hyp | data$trt * data$species)
```

EXERCISE 11: Plot histograms of the petiole data split by the researcher and light treatment.

R has many, many other plotting capabilities. I will just introduce one more, the xy plot. Perhaps you are curious about the relationship between leaf length and width. You might want to examine a scatter plot of these two variables. This can be done as:

```
> plot(data$leafleng,data$leafwid)
```

pairs() will produce scatter plots for each column in a data frame. For example

```
> pairs(data)
```

plots all columns of data against one another. This is not very sensical in this case.

EXERCISE 12: Use pairs to plot the hypocotyl and individual internode data against one another. (hint: think back to the matrix extraction functions that we used earlier). Which appear most closely correlated?

IV. Differences between groups

The two most common tests for determining if there is a significant difference between groups is the Student's t-test for comparison between two groups and the analysis of variance (ANOVA) for comparison among multiple groups.

The t-test function in R is t.test(). You can either use two separate arguments such as:

```
> tomato.height <- c(12,15,17,15,16,14,23,12)
> cucumber.height <- c(10,15,9,12,13,8,12,10,10,11)
> t.test(tomato.height,cucumber.height)
```

OR you can use a "formula" that describes how the data is grouped. Formulas use the ~ operator. You can think of ~ as "depends on" or "modeled as a function of". So in the example below we are testing whether fruit.diameter depends on (or can be modeled as a function of) fruit.type.

```
> #use previous fruit weight example
> t.test(fruit.diameter~fruit.type)
```

EXERCISE 13: Does total internode length depend on the light treatment?

An ANOVA asks whether there is more variance between group means than expected by random chance, given the amount of variance within groups. In R we use the aov() function. Below we place the results of an ANOVA in a new object and then use summary() to get a nice display of the results.

```
> aov1 <- aov(hyp~species,data=data)
> summary(aov1)
              Df Sum Sq Mean Sq F value    Pr(>F)
species        4   9632 2408.06  18.565 9.373e-15 ***
Residuals   1050 136196  129.71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We learn that hypocotyl length varies significantly by species.

The above example is a one-way ANOVA (one explanatory variable). You can also examine two (or more) grouping variable at once. This is specified as aov(RESPONSE ~ GRP1 + GRP 2). or aov(RESPONSE ~ GRP1 * GRP2). The “+” fits an additive model and the “*” asks if there are interactions among groups. That is whether the effect of GRP1 alters the effect of GRP2 (or vice versa).

EXERCISE 14: Fit a two-way ANOVA model to determine if internode length varies by species, if internode length varies by light treatment, and if the effect of light treatment varies by species.

V. Linear regression

Linear regression models determine the best-fit line described by y = m*x + b to a set of data where m describes the slope of the line and b is the intercept. One is asking whether or not the value of y is depends on the value of x. If the slope of the line is significantly different from 0, this indicates that there is a relationship between x and y.

For example you might want to know if hypocotyl length varied according to a plant’s original latitude of origin. In this case, hypocotyl length is “y” and latitude is x. You would fit a linear regression model testing this relationship as follows:

```
> lm1 <- lm(hyp~lat,data=data)
> summary(lm1)
```

Examine the output from the above commands. In this case the intercept is 26.2188 and the slope is -0.4926 and is significant. We can examine this relationship and plot the regression line:

```
> plot(hyp~lat,data=data) #xy plot
> abline(lm1) #add the best fit line from our linear model
```

The calculations underlying linear regression are similar to those used for ANOVA, and in R, anytime that you can fit an ANOVA model you can also fit a linear regression model. The output from the linear regression models often contain more information of interest.

```
> lm2 <- lm(hyp~species,data=data)
> summary(lm2)
```

```
Call:
lm(formula = hyp ~ species, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-32.340  -5.973  -1.092   5.269  39.843

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)             34.7571     0.6551  53.058  < 2e-16 ***
speciesS. chmielewskii  -2.6777     0.9067  -2.953  0.00322 **
speciesS. habrochaites  -3.6810     0.9067  -4.060 5.30e-05 ***
speciesS. pennellii     -5.9817     1.0498  -5.698 1.59e-08 ***
speciesS. peruvianum     3.7528     0.9157   4.098 4.50e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.425 on 1003 degrees of freedom
Multiple R-squared: 0.1053,  Adjusted R-squared: 0.1017
F-statistic:  29.5 on 4 and 1003 DF,  p-value: < 2.2e-16
```

In this case, the intercept is the mean of the first species (not named in the output) and the subsequent lines show the difference between each remaining species and the reference species.

EXERCISE 15: Repeat exercise 14, but using a linear model. Which species is being used as the reference? Which treatment is being used as reference? How much does the “L” treatment change internode elongation for the reference species? Is the effect significant? Which species has the smallest response to the “L” treatment?