Exploratory Data Analysis involves looking at data numerically and graphically in order to understand the data, oftern this is done prior to any attempts at *inference*. Exploratory analysis helps us understand if:

- Our ideas for center and spread are reasonable for the data.
- Our data is normally distributed – a typical assumption for standard statistical tests
- Our data is long-tailed or skewed – issues that can cause problems for standard statistical tests
- Variables within our data are highly correlated – an issue when doing multivariate analysis

In R there are numerous plots and numeric summaries that can be made and many different ways to make them. As to plots, there are 3 main plotting interfaces:

1. The basic ones – easier to use, not as pretty
2. lattice graphics – harder to use, but great for multivariate data sets, as you can create different panels for different groups
3. ggplot2 – a different approach to graphics that can produce "prettier" looking graphs and if you adjust to the approach, is not so difficult. We will skip this though.

Load the data

We will work with the two data sets from last time. Here are commands to load them from a web site:

```
> f <- "http://www.math.csi.cuny.edu/verzani/classes/MTH703/Data/bird-richness.csv"
> g <- "http://www.math.csi.cuny.edu/verzani/classes/MTH703/Data/alcids55.csv"
> birds <- read.csv(f)
> alcids <- read.csv(g, header = FALSE)
> names(alcids) <- c("year", "NAO", "climate", "ab1", "ab2", "ab3",
+     "ab4", "ab5")
```

The birds data set has a grouping variable `habitat` to keep track of which value something belongs to, the alcids uses 5 different variables. Both styles have their advantages, but we should be able to switch back and forth. Here we show how to go from the alcids style to the birds style using `stack`. (There are more flexible means, but harder to understand.) This function will create stack the values in a vector and create a corresponding variable to track where they came from:

```
> abunds <- stack(alcids[, -(1:3)])
> names(abunds) <- c("abundance","species")
> head(abunds)                              # first few
```

```
  abundance species
1       100     ab1
2       100     ab1
3       100     ab1
4     18000     ab1
5      1000     ab1
6      1500     ab1
```

Numeric summaries

Graphical exploration can be for univariate summaries – one variable at a time, or multivariate. We begin with some standard univariate summaries and why we use them.

Shape of a distribution: histogram and densityplots

The shape of a distribution is shown by a histogram or density plot. The reason we look at the shape is to investigate symmetry or skew, uni- or multi-modalness.

The histogram is one of the main graphics to do so. It is used for numeric data – categorical data uses a bar plot. For base graphics, the `hist` command will produce a histogram:

```
> hist(birds$log_total_en)
```

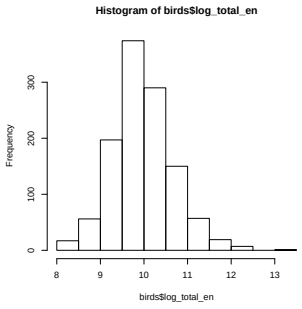

Figure 1: Histogram of log total enrichness

Question 0.1. By default the histogram is done on a count scale – the bars have height equal to the count of data that falls within its bin range. Another common style is to scale so the total area under the histogram is equal to 1. This is done by adding the argument `probability=TRUE`, as in

```
> hist(birds$log_total_en, probability = TRUE)
```

Make that graphic and see the difference

Question 0.2. If you are creating graphics for a paper, you might want to adjust the properties of the graphic: colors, shading, labels, axes, title, ... All these can be done if desired. To change the title, you specify the `main` argument, as in `main="my histogram"`.

Make a histogram of the `Richness` variable with a different title.

The lattice graphics version of the graphic is produced with the `histogram` function. To use lattice graphics, we must first load the package:

```
> library(lattice)
```

This is done only once per session, but must be done. For lattice graphics, we specify the data using a model formula. A model formula for this purpose has the following form:

```
response ~ predictor | group, data=dataset_name
```

Some of these are optional, but the funny tilde `~` is not, as that tells R you are specifying a model formula.

For a histogram, we have a univariate graphic, so we only need one side of the response/predictor pair. In the following command, note that we use the `data` argument to specify the data frame. In the previous example, we use the "dollar sign" notation to find a variable in a data frame, here the variable will be looked up first in the data set specified to `data`.

```
> histogram(~log_total_en, data = birds)
```

The lattice graphics helps us make graphics with groups, to see how easily this is done, we add in the information for the habitat:

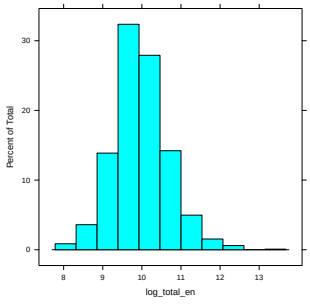

Figure 2: Histogram of log total enrichness. Note scale is on percent scale.

```
> histogram(~log_total_en | habitat, data = birds)
```

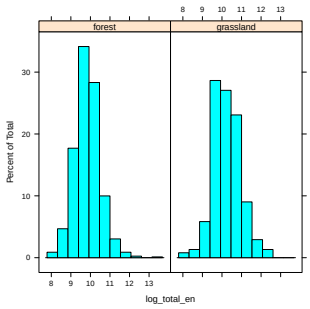

Figure 3: Histogram of log total enrichness for each of the two habitats.

Question 0.3. Make two histograms of the `abundances` in the `abund` data set. The first in general, the second by species. Comment as to the shape of the data.

Question 0.4. The abunds data is very skewed. A logarithmic transform might help. To do so, you can specify the model as

```
~ log(abundance), data=abunds
```

Do so, and the comment on the shapes. Are they now symmetric?

Although we can use the `log` function here as we might expect, simple things like addition and subtraction and multiplication have a totally different interpretation in a model formula.

Density plots:

The density plot shows about the same information as a histogram, only is more closely related to the underlying continuous distribution for the variable. To make a density plot with lattice graphics is the same as for a histogram, only use `densityplot`:

```
> densityplot(~log_total_en, data = birds)
```

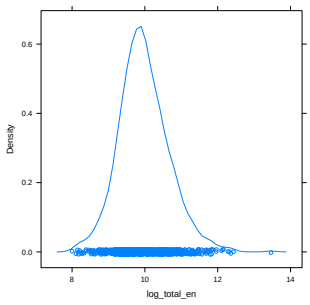

Figure 4: Density plot is like the histogram, but uses just one line and has total area under curve adding to 1.

Question 0.5. Make a similar plot, only break up by the `habitat` variable.

Question 0.6. Make a density plot for the `Richness` variable. Comment on the shape.

Boxplots

Boxplots are a great graphical device to quickly compare centers, spreads, symmetry and skew for many different variables. Boxplots can be univariate or multivariate.

The `boxplot` command will produce a basic boxplot. For example:

```
> boxplot(birds$Richness)
```

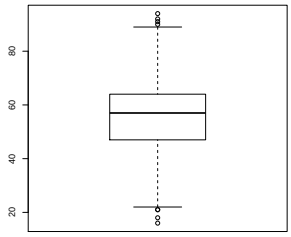

Figure 5: Basic boxplot shows center (middle line), spread (the box represents the middle 50% of the data – the IQR), skew (are the two halves similar) and marks suspected outliers (by dots)

In this case the boxplot shows a symmetric data set with outliers that are consistent with those of a normal distribution for 1100 observations. The center is a bit under 60 and the spread about 15. The IQR for normal data is larger than the standard deviation. For normal data –other data may be different – the relationship is around IQR=1.35σ.

Question 0.7. Make a boxplot of the `abundance` data. What do the outliers look like? What about the box? Try a log transform to see if you can make this nicer. (What is the error about?)

Question 0.8. The `boxplot` also allows for a model formula. For example, try this and see what comes of it:

```
> boxplot(Richness ~ habitat, data = birds)
```

The `lattice` package has the function `bwplot` to make a boxplot. There are a few different ways to call this function. For example

```
> bwplot(~ Richness, data=birds)          ## basic univariate display
> bwplot(~ Richness |habitat, data=birds) ## a panel for each level
> bwplot(Richness ~ habitat, data=birds)  ## basic multivariate display.
```

The second one has a different panel for each habitat, the last one has the plots side-by-side.

Question 0.9. Make a boxplot of `abundance` for the different `species`. Then redo after a logarithmic transform. Which species has the smalles median? Which has the smallest largest value? Why do somehave no lower whisker?

Quantile plots

The quantile-quantile plot is used to plot the quantiles of one distribution against that of another. If the plot shows a "more-or-less" straight line, then the two distributions are similarly shaped (they may have different means and or spreads). If the line has a pronounced curve it can indicate which distribution has longer tails.

The quantile-normal plot is the typical one, where one distribution is the standard normal distribution. This allows one to visually check for assumptions of normality.

The basic plot is generated by `qqnorm`, as with

```
> qqnorm(birds$Richness)
```

Question 0.10. We know the species abundance data is non-normal. Investigate how the graphic shows the following

```
> qqnorm(alcids$ab4)
```

Compare to

```
> qqnorm(log(alcids$ab4))
```

The lattice graphics function is `qqmath` . This function has an argument `distribution` to specify a distribution, but we use the default normal. The basic call is similar to that of histogram, as the graphic is univariate.

```
> qqmath(~Richness | habitat, data = birds)
```

Question 0.11. Using the conditioning form, one can get different panels for different levels of the conditioning variable. Try this for the `Richness` where you use `habitat` for conditioning.

Time series

The abundance data is collected over time. A typical graph to display this is a time-series plot where time is on the x axis and the count on the y axis. R has facilities to plot time series that make life easy, but one must put tell R the data is a time series. For regularly spaced data, the `ts` function does this. The default size is steps of size 1 (years in our example), so to make a time series object we have

```
> ab1 <- ts(alcids$ab1, start = 1954)
```

Note although they have the same name, we have two variables `ab1` – the one we just defined, and the one inside the alcids data set.

A time series plot is then made by the `plot` command:

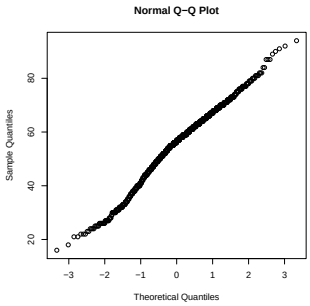

Figure 6: Basic quantile-normal plot for `Richness` data. The grahic is basically a straight line which indicates that two distributions have essentially the same shape. In this case, we conlclude the distribution for `Richness` is approximately normal. Or `qq` but it requires the data to be in a special format – one variable with the data and a factor with only 2 variables. As such, we don't describe it here.

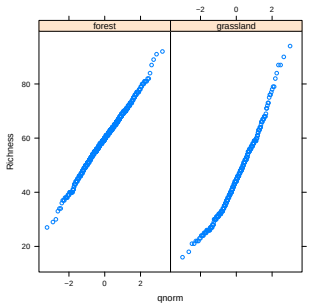

Figure 7: A qqplot for the bird `Richness` by `habitat` using the `qqmath` function.

```
> plot(ab1)
```

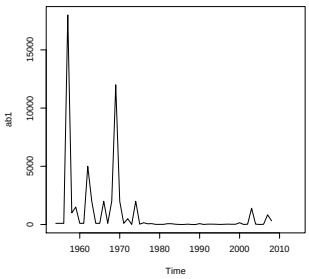

Figure 8: Time series plot of abundance for species 1. It appears to show a cylical behaviour. To investigate this, you can try `acf(ab1[!is.na(ab1)])`.

Question 0.12. Is there a similar pattern for `ab2`? `ab3`?

The time series plot is almost a scatterplot with points that are connected, but since the computer knows you are plotting times, it can guess what nice axes' labels would be.

Scatterplots

We saw how to make scatterplots (or *x*-*y* plots) previously. The base `plot` command will do so, and the relationship can be specified two different ways (as two variables, and as a formula). Which is appropriate depends on how you have your data. For example, to plot abundance 1 against abundance 2 we would likely use the following:

```
> plot(alcids$ab1, alcids$ab2)
```

If you make this plot you will see no apparent correlation. Even if you plot the logarithms of each.

Question 0.13. Okay, make both plots and see.

When we plot `Richness` against `log_total_en` we can use either style, but prefer the formula style:

```
> plot(log_total_en ~ Richness, data = birds)
```

This data set shows positive correlation.

We saw how to add to a base scatterplot. The regression line was added with the command:

```
> abline(lm(log_total_en ~ Richness, data = birds))
```

We can add points with different colors and shapes (`pch`), as follows:

```
> points(log_total_en ~ Richness, data=birds,
+        subset=habitat=="forest",
+        pch=15, col="blue")
```

In lattice graphics, the function `xyplot` generates these graphics. The basic call is similar:

```
> xyplot(log_total_en ~ Richness, data = birds)
```

We can break up by levels of `habitat` is desired:

```
> xyplot(log_total_en ~ Richness | habitat, data = birds)
```

However, adding to the graphic is more involved. We do so by making a panel function. These are the steps involved to add a regression to each panel above.

```
> xyplot(log_total_en ~ Richness | habitat, data=birds,
+        panel=function(x,y) {
+          panel.xyplot(x,y);              # add points
+          panel.loess(x,y, col="red")     # add line
+        })
```

The panel function draws the points, then adds a loess line (local regression). These functions direct what happens on each panel so there is great flexibility – but that makes it harder to learn.

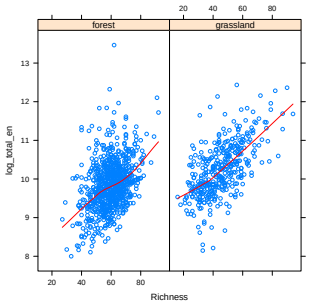

Figure 9: Illustration of panel function