Tutorial 5: Regression and Correlation

Goal: To provide a more in depth look at bivariate statistics and to explore quantitative and graphical methods necessary to perform these analyses.

Note: All text in the Arial font is instruction or explanation. All text in Courier font is input or output from R.

Step-1: Simple linear regression

Let’s consider an example where you wish to predict the relationship of one variable from the knowledge of another. Assuming the relationship is linear in nature; simple linear regression is usually the best place to start. Recall that the general linear model is given as:

$$y_i = \alpha + \beta x_i + \epsilon_i$$

In which the $\epsilon_i$ are assumed independent and $N(0,\sigma^2)$. The slope of the line (the regression coefficient) is $\beta$, the increase per unit change in $x$. The line intercepts the $y$-axis at $\alpha$. Recall that the parameters $\alpha$, $\beta$, and $\sigma^2$ can all be estimated using the approach of least squares trend line fitting (finding the solution that minimizes the sum of squared residuals).

Suppose that you were a wildlife manager responsible for maintaining a herd of trophy African lions. In order to manage the population, it is important to know the age structure of the population. Behavioral studies suggest that removal of lions in excess of six years has virtually no impact on the social structure of the group, whereas taking younger males is more disruptive. It has been suggested that the amount of black pigmentation on the nose of male lions increases with age. Whitman et al. (2004) measured the proportion of black on the noses of lions of known ages in Tanzania, East Africa. These proportions, obtained from photos and digitally analyzed are available for download (lions.txt). What we wish to know is how well does the proportion of black in the nose predict the animal’s age?

The very first place to start of course is with a simple scatter plot. We need to confirm that the data bear some resemblance to a linear cloud of points so that we may proceed. Download the data set (lion.txt), store in a temporary directory on your c: drive, then read in the data:

```r
> lion.data<-read.table("c:\temp\lion.txt",header=T)
> attach(lion.data)
> names(lion.data)
[1] "Age" "Nose"
```

Now, to make the scatterplot:

```r
> plot(Nose,Age)
```

Which yields:
Obviously, there appears to be a linear relationship between Percent Nose Blackness and Age, so proceeding via an ordinary least squares regression approach seems warranted. What other questions might you ask yourself at this stage of the process, especially concerning the nature of the variables (think about issues of natural variability and measurement error)?

To proceed, we call the linear model (lm) procedure from R. The tilda (~) in the model statement should be read as “described by”. Thus, the response variable goes to the left of the tilda and the explanatory variable goes to the right. In our case:

```r
> lm(Age~Nose)
```

Call:
`lm(formula = Age ~ Nose)`

Coefficients:
```
(Intercept)         Nose
    0.879       10.647
```

In its raw form, the output for lm is very brief. All we get is the two parameter estimates—enough to generate the linear model equation. We know that $\alpha = 0.879$ and $\beta = 10.647$ so the equation of the line must be:

$$y_i = 0.879 + 10.647 x_i + \varepsilon_i$$
We usually want tests of significance, etc. to gain more insight into the relationship. The result of \texttt{lm} is actually a "model object" in R. Where other statistical systems focus on generating printed output that can be controlled by settings options, you get instead the result of a model fit encapsulated in an object from which the desired quantities can be obtained using "extractor functions". Indeed, an \texttt{lm} object contains much more information than we just saw. An example of a basic extractor function is \texttt{summary}:

\begin{verbatim}
> summary(lm(Age~Nose))
Call:
\texttt{lm(formula = Age ~ Nose)}
Residuals:
  Min     1Q Median     3Q    Max
-2.5449 -1.1117 -0.5285  0.9635  4.3421
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.8790     0.5688   1.545    0.133
Nose           10.6471     1.5095   7.053 7.68e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.669 on 30 degrees of freedom
Multiple R-Squared: 0.6238,    Adjusted R-squared: 0.6113
F-statistic: 49.75 on 1 and 30 DF,  p-value: 7.677e-08
\end{verbatim}

This gives us much more information. Let's look more closely at some of the output. The "Call:" section is simply a repeat of the model. Not particularly interesting, but useful if you are saving the data to a stored output file. The "Residuals:" section provides a coarse look at the response of your residuals from which to check certain distributional assumptions. For example, the assumption is that the mean of the residuals should be zero, so your median should not be too far from that. You can also examine the relative proportion allocated to the adjacent quartiles to see if the data look balanced. The "Coefficients:" section again provides our estimates of alpha and beta, but they are now accompanied by standard errors, \(t\)-tests, and \(p\)-values. The asterisks to the right indicate "significance level". Most of our interest is in beta. Whether or not alpha is significant or not is usually immaterial. It is useful in some cases to test whether or not the regression lines runs through the origin (or is statistically indistinguishable from zero). The "Residual standard error:" is the residual variation (an expression of variation of the observations around the regression line) and estimates the parameter \(\sigma\). The "R-squared" sections provide two \(r\)-squared estimates. The first \(R^2\) is simply the square of the Pearson correlation coefficient (\(R^2 = r^2\)). The adjusted \(R^2\), if multiplied by 100\%, can be interpreted as the "\% variance reduction". Lastly, an \(F\)-test is provided for the hypothesis that the regression coefficient is zero. When there is just one explanatory variable, as is the case with OLS, it is duplicative of the \(t\)-test already provided.
Finding a reasonably good linear model, the next step is to fit a line through the data and construct a plot.

> abline(lm(Age~Nose))

Which yields:

![Graph showing linear model](image)

**Step 2: The Residuals**

Remember that the difference between each data point and the value predicted by the model is referred to as the residual. Suppose we wished to find out what the residuals are in this model? Fortunately, there is a built-in function to do so. Two further extraction functions include `fitted` and `resid`.

> lm.lion<-(lm(Age~Nose))
> fitted(lm.lion)
> resid(lm.lion)

> lm.lion<-lm(Age~Nose)
> #lm.lion is just short for “linear model of lion”, we could have used any other name assignment#
> fitted(lm.lion)
> resid(lm.lion)
You now have both the observed and expected value for each of the 32 regression points. To create a plot where the residuals are displayed by connecting the fitted values to the observed values, we can do the following:

> segments(Nose, fitted(lm.lion), Nose, Age)

Which yields:

We can then start to examine the residuals in more detail. A simple plot of residuals vs. fitted values is obtained via:

> plot(fitted(lm.lion), resid(lm.lion))
And to examine the normality assumption:

```r
> qqnorm(resid(lm.lion))
```

Perhaps followed by a more explicit test:
> shapiro.test(resid(lm.lion))

    Shapiro-Wilk normality test

data:  resid(lm.lion)
W = 0.9388, p-value = 0.0692

Which in this case suggests that the data are normally distributed ($P > 0.05$, but not by much).

**Step 3: Prediction and confidence intervals**

Fitted lines are often presented with bands of uncertainty surrounding them. There are "narrow" bands and "wide" bands. The narrow bands, or confidence bands, reflect the degree of uncertainty of the fitted regression line itself. If there are a large number of points (i.e., $N$ is large) these bands may be quite narrow, reflecting a well determined line. The bands usually display a narrow or waist near the mean of $X$ and $Y$, since the mean is usually better determined here. This was explained at length in lecture.

The wide bands (to the outside of the confidence bands) usually refer to as the prediction bands or prediction interval. These reflect the uncertainty about future observations. The limits of these bands approach ± standard deviations around the line and in larger samples display little if any curvature.

Predicted values, with or without prediction and confidence bands can be determined using the function `predict`:

> predict(lm.lion)

```
1        2        3        4        5        6        7        8        9
3.114901 2.369603 2.050189 2.263132 2.156661 2.263132 2.156661 2.795488 3.327844
10       11       12       13       14       15       16       17       18
3.221372 3.008430 2.689017 2.476074 3.753728 3.647257 3.114901 4.073142 5.350796
19       20       21       22       23       24       25       26       27
5.457268 7.160807 7.267278 8.544932 3.966671 1.943718 5.989624 5.563739 4.499027
28       29       30       31       32
4.818440 4.499027 8.757875 9.290231 6.309037
```

If you add `interval="confidence"` or `interval="prediction"` you can generate a fuller picture. For example:

> predict(lm.lion, int="c")

```
   fit      lwr       upr
1  3.114901 2.420221  3.809582
2  2.369603 1.545942  3.193264
3  2.050189 1.160887  2.939492
   .       .         .
32 6.309037 5.473455  7.144619
```
\texttt{lwr} and \texttt{upr} indicate the upper and lower confidence limits for the expected values.

Unfortunately, constructing the final scatterplot is a bit complicated. The best function to use is \texttt{matlines}, which plots the columns of a matrix as a vector. However, there are a number of limitations to using it directly here. The solution requires that you predict into a new data frame, containing suitable x values. Try the following:

\begin{verbatim}
> pred.frame<-data.frame(Nose=0:1)
> pp<-predict(lm.lion, int="p", newdata=pred.frame)
> pc<-predict(lm.lion, int="c", newdata=pred.frame)
> plot(Nose, Age, ylim=range(Age, pp, na.rm=T))
> pred.lion<-pred.frame$Nose
> matlines(pred.lion, pc, lty=c(1,2,2), col="black")
> matlines(pred.lion, pp, lty=c(1,3,3), col="black")
\end{verbatim}

What this is doing is creating a new data frame in which the Nose variable contains the values at which we want to predictions to be made (0 to 1). \texttt{pp} and \texttt{pc} are then made to contain the result of the \texttt{predict} for the new data in \texttt{pred.frame} with prediction and confidence intervals, respectively. Then the plot is called for Nose vs. Age with the \texttt{ylim} specification to provide sufficient room to overlay the plots. Finally, the CI and PI curves are added in the last two lines using the \texttt{matlines} function, with line types and colors specified.

You should now have a plot that looks like this:
Notice that the narrow waist on the CI plot is not in the “middle”. Why? Note that you can continue to dress this plot up a bit more for publication by adding units to the axes, the equation of the line into the box, and perhaps $R^2$ and $P$.

**Step 4: Regression diagnostics**

We have already looked at some basic plots of residual response above. A more direct route to get the basic residual plots of interest would be:

```r
> lm.lion<-lm(Age~Nose)
> par(mfrow=c(2,2), mex=0.6)
> plot(lm.lion)
> par(mfrow=c(1,1), mex=1)
```

Which yields:

![Residuals vs Fitted](image)

![Normal Q-Q](image)

![Scale-Location](image)

![Residuals vs Leverage](image)
The `par` command sets up a 2 x 2 layout of the graphs. The first panel provides a comparison of the fitted values vs. the residuals. We wish to see a band around zero for this plot. Note the dashed line at zero to draw your attention to this. The red line is a smoothed fitted line to the data to help your eye evaluate the data. **NOTE:** a nice feature is that all points that represent potential problems in that plot are numbered for your convenience so you can examine the original data. The three remaining panels all utilize standardized residuals; these have been corrected for differences in SD of residuals depending upon their position in the design. Residuals corresponding to extreme x-values generally have lower SD due to overfitting. The lower left panel is similar to the first, but just employs $\sqrt{\text{standardized residuals}}$ which helps reduce skewness and permits a better view of dispersion trends. The upper right panel is the Q-Q plot which you are already well familiar with. The lower right panel is one of the more instructive for regression diagnostics; it evaluates the leverage and Cook’s Distance of individual points. All of the plots identify points 22, 27, and 30 as potential problems. Based on Cook’s Distance, I would take the closest look at point 30, try deleting it, re-running the linear model, and watch your parameter estimates and fit ($R^2$) to see how it changes.

A suite of other diagnostics is available for interpreting regressions It is also possible to obtain individual diagnostics:

```r
> par(mfrow=c(2,2), mex=0.6)
> plot(rstandard(lm.lion))
> plot(rstudent(lm.lion))
> plot(dffits(lm.lion),type="l")
> matplot(dfbetas(lm.lion),type="l", col="black")
> lines(sqrt(cooks.distance(lm.lion)),lwd=2)
> par(mfrow=c(1,1), mex=1)
```

Which yields again a 2 x 2 array of plots. The function `rstandard` gives the standardized residuals discussed above. The function `rstudent` is similar but produces a "leaves-one-out-residual". In other words, the fitted value is calculated using all the points except the current point as it does the calculations. There is not a large difference between the two plots ordinarily, except in the case where there are extreme values, which tend to be further out on the latter plot. The function `dffits` expresses how much an observation affects the associated fitted value. As with previous plots, points 22 and 30 seem to stand out as potentially problematic. The final function, `dfbetas` shows the change in estimated parameters when that value is excluded.

The above statements yield the following plots for interpretation:
By virtually all measures, point 30 (and perhaps 22 & 27) are problematic. Let’s re-run the analysis after first deleting the most problematic point and see what happens. This is simple to do in R:

```r
> summary(lm(Age~Nose, subset=-30))
```

Which yields:
Call:
  lm(formula = Age ~ Nose, subset = -30)

Residuals:
  Min      1Q  Median      3Q     Max
-2.0522 -0.9810 -0.4072  0.6353  3.4973

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.2938     0.5089  2.542   0.0166 *
Nose          8.8498     1.4175  6.243 8.19e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.447 on 29 degrees of freedom
Multiple R-Squared: 0.5734,    Adjusted R-squared: 0.5587
F-statistic: 38.98 on 1 and 29 DF,  p-value: 8.191e-07

Note the changes from the earlier summary we did. The estimate for beta changed a fair bit (from 10.6471 to 8.8498), but is still highly significant. Likewise, the model also weakened with $R^2$ going from 0.6238 to 0.5734.

Taking a look at the standard diagnostic plots,

```r
> par(mfrow=c(1,2), mex=0.6)
> par(mfrow=c(2,2), mex=0.6)
> plot(lm.lion)
```

we now see that points 27, 28, 29 become potentially problematic and the Q-Q plot suggest the data are now non-normal (subject to testing) where they passed before. In this specific case, deleting a potentially problematic point did nothing to increase the clarity of our model one way or the other, if anything decreased it predictive power somewhat, and failed to influence the major parameter estimates. As a result, I would go with the original analysis.

Notice we indeed now have non-normal residuals:

```r
> shapiro.test(resid(lm.lion))

    Shapiro-Wilk normality test

data:  resid(lm.lion)
W = 0.9163, p-value = 0.01881
```

Diagnostic plots from revised data set (less point 30):
Step 5: Correlation:

Correlation analysis, while related to regression analysis, has an entirely different purpose. Here, you are not attempting to describe a linear model of how one variable explains another. You are simply trying to assess the level of association between two variables. The result is a correlation coefficient which ranges from -1 to 0 to +1; where the extremes indicate strong correlation (negative or positive, respectively) and 0 represents no correlation.
Assuming the data are normally distributed, the standard approach is the use of the Pearson product Moment Correlation Coefficient ($r$), or simply Pearson correlation, and usually inferred when you just see “correlation”. The calculation in R is a simple function:

```
> cor(Age, Nose)
[1] 0.7898272
```

You can extend the information to the entire correlation matrix by:

```
> cor(lion.data)
    Age      Nose
Age 1.0000000 0.7898272
Nose 0.7898272 1.0000000
```

Which is not very instructive in the case of only two variables; but, you can check to see that the main diagonal is reading 1.00 as it should and the off-diagonal values are the correlation coefficient $r$.

The problem with what we have done so far, while instructive, does not provide us with the usual piece of information that we are looking for: is the value of $r$ significant? To this end we need to run the `cor.test` procedure:

```
> cor.test(Age, Nose)

Pearson's product-moment correlation

data:  Age and Nose
    t = 7.0534, df = 30, p-value = 7.677e-08
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:  
  0.6088022 0.8926721
sample estimates:
   cor
0.7898272
```

As expected from our previous work above, we see that the correlation coefficient is highly significant.

As in many of the parametric procedures we have run to date, there may be times due to data structure, assumption violations, etc. that you will be in need of a nonparametric alternative. The Spearman Rank Correlation Coefficient ($\rho$) is usually the best alternative. Rho is derived by simply re-expressing the observations from their original scale onto a rank scale. There is no special spearman function; it is just a variant of the `cor` procedure:

```
> cor.test(Age, Nose, method="spearman")

Spearman's rank correlation rho
```

```
data: Age and Nose
S = 1392.065, p-value = 1.013e-06
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.7448561

Warning message:
Cannot compute exact p-values with ties in: cor.test.default(Age, Nose, method = "spearman")

A final correlation method that is sometimes useful is Kendall's \( \tau \) (tau). This is based on counting the number of concordant and discordant pairs. A pair of points is concordant if the difference in the x-coordinate is of the same sign as the difference in the y-coordinate. For a perfect monotone relationship, either all pairs will be concordant or all will be discordant. Given all the possible pairwise comparisons, this can be quite computationally intensive. This test usually provides similar results to Spearman's.

> cor.test(Age, Nose, method="kendall")

    Kendall's rank correlation tau

data: Age and Nose
z = 4.295, p-value = 1.747e-05
alternative hypothesis: true tau is not equal to 0
sample estimates:
  tau
0.542123

Warning message:
Cannot compute exact p-value with ties in: cor.test.default(Age, Nose, method = "kendall")

**Problem: Exercise 17.1 (p.358-9).**
Solve for each of the sections of this problem and produce a final plot showing the best fit regression line, CI, and PI.

**Problem: Example 19.1a, 19.1b, 19.3, 19.4, 19.13**
Solve for each of these boxed examples and produce a scatterplot of the data.