STA 410/2102, Fall 2000 — Assignment #1

Due in class on October 19. Worth 16% of the final mark.

When the relationship of a response variable, y, to a predictor variable, x, appears to be non-linear, it is common to consider power transformations of x, in an attempt to find a model that fits better. One might, for instance, attempt to model y as $a + bx^2 + \epsilon$, for some values of the regression coefficients a and b, with ϵ being the random residual. Or if the data looks different, one might instead model y in terms of $x^{1/2}$. In this context, using the p-value for the regression that was done as an indication of the strength of evidence against the null hypothesis of no relationship would be naive, as it fails to account for the fact that the transformation of x was based on an examination of the data. However, one can obtain a valid p-value from a permutation test.

In this assignment, you will write four short functions in S or R, and use them to investigate statistical issues arising from this situation.

The functions to write are as follows:

1. `pow.reg` fits a regression model using the best power transformation. It takes as arguments the vector of predictor values, the vector of response values, and a vector of possible powers by which to transform the predictor. It return a list with elements `power`, set to the power that produces the best fit, and `model`, set to the result of modeling y in terms of x raised to that power, using the `lm` function.

2. `test.naive.pvalue` tests whether the naive p-values for a model fit using `pow.reg` are valid. It takes as arguments a vector of predictor values, x, a vector of possible powers for transforming x, and a sample size for the test, N. It should produce N simulated datasets in which x contains the specified values and y contains i.i.d. normal values having no relationship to x. It should apply `pow.reg` to each such dataset, and for each, find the p-value for the regression model that is selected as the best. It should return the vector of these p-values. Since the null hypothesis of no relationship is true for these datasets, these p-values should be uniformly distributed between 0 and 1, if they are valid.

3. `pow.perm.pvalue` finds a valid p-value with a permutation test. It takes the same arguments as `pow.reg`, plus an argument giving the number of random permutations to use in finding a permutation p-value. It should return a single number which is a p-value for the test of the null hypothesis of no relationship versus the alternative of a relationship using any of the power transformations specified.

4. `test.perm.power` determines the power of the permutation test in a given situation. It takes as arguments a vector of predictor values, x, the power, p, by which to transform these values, the standard deviation of the normally-distributed residuals, the number of permutations to use in the permutation test, and the sample size, N for the test of the power. N simulated datasets should be produced, in each of which $y = x^p + \epsilon$, with the datasets differing only in the random residuals ϵ. The value returned by `test.perm.power` should be the vector of p-values for these datasets, as found by `pow.perm.pvalue`, from which the power of the test at any level α can be found by simply seeing what fraction of these p-values are less than or equal to α.

You will need to use the `lm` function to fit linear models. Here is an example of its use:

```
> sqrtx <- x^0.5
> m <- lm(y~sqrtx)
> coef(summary(m))
              Estimate Std. Error   t value   Pr(>|t|)
(Intercept) -0.5677125  0.4132038 -1.373928 0.24142747
sqrtx        1.3579578  0.3513184  3.865319 0.01806527
```

The hypothesis of no relationship is the same as the hypothesis that the regression coefficient for `sqrtx` is zero. The p-value for the test of this hypothesis is in the lower right, and can be accessed as `coef(summary(m))[2,4]`.

The easiest way to see which power transformation, p, is best is to look at the absolute value of the correlation between y and x^p. The transformation that produces the highest correlation will also produce the smallest residual standard deviation when used in a regression model.

You will need to generate normally-distributed values. The function call `rnorm(10,0,2)` generates a vector of 10 independent normal values with mean 0 and standard deviation 2.

The function call `sample(z)` returns a random permutation of the vector z.

You should try out your functions on several datasets to make sure they work correctly, including the following one:

```
x:  0.00  0.10  0.20  1.00  2.00  5.00
y: -0.15  0.44 -0.91  0.73  1.20  2.72
```

Use possible powers of 0.1, 0.2, ..., 1.9, 2.0 when applying `pow.reg` to this dataset. When testing `pow.perm.pvalue` on this dataset, use the same possible powers, and set the number of permutations to 999. When testing whether the naive p-values are correct, use the values for x and the possible powers above, and use a sample size of at least $N = 100$, preferably larger.

When testing the power of the permutation test, use the situation where the x values are as above, these x values are transformed by raising to the power 0.6, and the residual standard deviation is 0.5. Use a sample size of at least $N = 100$. Compare the power of the permutation test at the level $\alpha = 0.1$ when 9 permutations are used and when 29 permutations are used. (Note: Consider carefully how you should do the comparison of p-values to this value for α.) These computations will take a fair amount of time.

You should hand in listings of your functions, which should be formatted for easy reading, with an appropriate amount of documentation, along with the output of your tests showing that the functions work correct, and of those tests you did to investigate the statistical issues. You should also hand in your discussion of the statistical issues, in particular of whether the naive p-values are correct, whether the naive and permutation p-values differed for the particular dataset above, and whether and if so how the power of the permutation test varies with the number of permutations.