Randomization and the Bootstrap

```
require(mosaic)
require(Stat2Data)
```

Randomization (Permutation) Tests

Recall that the inference for linear regression relies on the assumptions that the following three conditions are met:

1. Linearity
2. Constant Variance
3. Normality of Residuals

We know how to assess whether these conditions are met, and we have learned a few techniques for correcting them when they are not (e.g. transformations). Today, we will learn a few techinques based on randomization for making *non-parametric* inferences. Such inferences do not rely on stringent assumptions about the distribution on of the residuals.

In this example, we are trying once again to explain a college college student's `GPA` as a function of her score on the verbal section of the SAT.

```
data(FirstYearGPA)
```

First, let's examine the relationship between these two variables graphically.

```
xyplot(GPA ~ SATV, data=FirstYearGPA, pch=19, cex=2, alpha=0.5, type=c("p","r"), lwd=5)
```

There appears to be some linear assocation between these variables, but it is not particularly strong. We can quantify this relationship using the correlation coefficient.

```
cor.actual <- cor(GPA ~ SATV, data=FirstYearGPA)
cor.actual
```

In this case the value of the correlation coefficient is about 0.3, which is not large, but does appear to be significantly different from 0. Recall the t-test for correlation that we learned earlier:

$$t = r\sqrt{\frac{n-2}{1-r^2}}, \qquad n-2 \text{ d.f.}$$

```
cor.test(FirstYearGPA$GPA, FirstYearGPA$SATV)
```

This test confirms that the correlation between college `GPA` and `SATV` is most likely not zero, but the validity of this test requires the assumptions for simple linear regression to be met (specifically, normality). Let's assume that in this case the assumptions are **not** met. Can we still feel confident that the correlation is non-zero?

If `GPA` and `SATV` were really correlated, then there is a real relationship binding the i^{th} value of `GPA` to the i^{th} value of `SATV`. In this case it would not make sense link the i^{th} value of `GPA` to the some other value of `SATV`. But if the correlation between these two variables was in fact zero, then it wouldn't matter how we matched up the entries in the variables!

The basic idea of the permutation test is to shuffle the mapping between the two variables many times (i.e. sample *without replacement*), and examine the distribution of the resulting correlation coefficient. If the actual value of the correlation coefficient is a rare member of that distribution, then we have evidence that the null hypothesis of zero correlation might not be true.

- Execute the following code several times. Get a feel for how the regression line can change, depending on the permutation of the data.
- Do you still believe that the slope correlation is non-zero?

```
xyplot(GPA ~ shuffle(SATV), data=FirstYearGPA, pch=19, cex=2, alpha=0.5, type=c("p","r"), lwd=5)
```

The procedure for the randomization test is simple. We simply shuffle the response variable and compute the resulting correlation coefficient with the explanatory variable. But we do this many times, until we have a sense of the distribution of that correlation coefficient. We then examine where the observed correlation falls in that distribution.

```
# Do this 5000 times
rtest <- do(5000) * cor(GPA ~ shuffle(SATV), data=FirstYearGPA)
p1 <- densityplot(~cor, data=rtest, xlim=c(-0.4,0.4), xlab="Correlation Coefficient")
ladd(panel.abline(v=cor.actual, col="red", lwd=3), plot=p1)
```

Of course, we can also explicitly find where in the distribution the observed correlation lies.

```
pdata(~cor, q=cor.actual, data = rtest, lower.tail=FALSE)
```

`lower.tail` tells R whether to compute the amount of data to the right or the left of the test statistic. In this case, we want the amount of data to the right of the line, so we set `lower.tail=FALSE`.

Finally, we can find a non-parametric 95% confidence interval for the correlation coefficient. The interpretation here is if our actual correlation value fell in this interval, we would **not** consider it statistically significant.

```
qdata(~cor, p=c(0.025, 0.975), data = rtest)
```

- Compare this confidence interval to the one returned by the correlation test above. Why are they different?
- Perform the above procedure to test the correlation between `GPA` and `SATM`.
- Perform the above procedure to test the correlation between `GPA` and `HSGPA`.

The Bootstrap

A similar idea is the bootstrap. Here, we will repeatedly sample cases from our data set, *with replacement*, and approximate the sampling distribution of a statistic. The key idea here is that even though the solution to the regression model is deterministic, the data itself is assumed to be randomly sampled, and so all of the estimates that we make in a regression model are **random**. If the data changes, then the estimates change as well. The bootstrap gives us a non-parametric understanding of the distribution of those estimates. Once again, the advantage to this method is that we can construct meaningful confidence intervals for, say, the slope of the regression line, without having to assume that the residuals are normally distributed.

In this example we are examining the prices of Porsches. We want to estimate the `Price` as a function of the `Mileage`. Our sample of 30 cars looks like this:

```
data(PorschePrice)
xyplot(Price ~ Mileage, data=PorschePrice, pch=19, cex=2, alpha=0.5, type=c("p","r"), lwd=5)
```

To create a bootstrap sample, we select rows from the data frame uniformly at random, but with replacement.

```
xyplot(Price ~ Mileage, data=resample(PorschePrice), pch=19, cex=2, alpha=0.5, type=c("p","r"), lwd=5)
```

- Note the differences between this plot and the previous one. What do you notice?
- Run the preceeding code several times. What is changing? What is staying the same?

Interactive ways to think about the bootstrap

The **manipulate** package allows us to create bootstrap samples on the fly.

```
bslope <- function (formula, data, n) {
  # Original data
  # Now do the bootstrap
  bootstrap <- do(n) * coef(lm(formula, data=resample(data)))
  xyplot(formula, data=data,
         panel = function (x, y, ...) {
           panel.xyplot(x,y, pch=19, cex=2, alpha=0.5)
           fm <- lm(formula, data=data)
           panel.abline(fm, col="red", lwd=5)
           # Add the bootstrap slopes
           for (i in 1:n) {
             panel.abline(t(bootstrap[i,]), -0.5, col="lightgray", lwd=0.3)
             }
           panel.text(75, 80, paste("mean intercept\n", round(mean(~Intercept, data=bootstrap), 6)), ce
           panel.text(75, 75, paste("sd intercept\n", round(sd(~Intercept, data=bootstrap), 6)), cex=0.
           panel.text(75, 70, paste("mean slope\n", round(mean(~Mileage, data=bootstrap), 6)), cex=0.75
           panel.text(75, 65, paste("sd slope\n", round(sd(~Mileage, data=bootstrap), 6)), cex=0.75)
           }
  )
  }
# bslope(Price ~ Mileage, data=PorschePrice, 100)
```

- Play around with the following demonstration until you get a feel for what is happening.

```
# install.packages("manipulate")
require(manipulate)
manipulate(
  bslope(Price ~ Mileage, data=PorschePrice, n),
  n = slider(2, 500, initial=100)
)
```

Bootstrap distributions and confidence intervals

One advantage of the bootstrap is that it allows us to construct a sampling distribution for the slope coefficient that is not depedent upon the conditions for linear regression being met.

The original confidence intervals for our SLR model depend upon the conditions being true.

```
fm <- lm(Price ~ Mileage, data=PorschePrice)
confint(fm) # Hint, if you want to change the confidence level, read the documentation
```

Now let's create a bootstrap distribution for the regression coefficients.

```
# I'm only doing 100 samples, but you should do more!
bootstrap <- do(100) * coef(lm(Price ~ Mileage, data=resample(PorschePrice)))
p2 <- densityplot(~Mileage, data=bootstrap)
ladd(panel.abline(v=coef(fm)["Mileage"], col="red", lwd=3), plot=p2)
```

The bootstrap distribution will always be centered around the value from our real data, but shows us some other likely values for the coefficient (essentially, sampling error). One way to quantify this variability is to create a confidence interval.

In the book, there are three methods for constructing confidence intervals from the bootstrap. The most robust method is Method #2, but we'll talk about all three.

Method #1

The first method assumes that the bootstrap distribution is **normal**. (This is a weird assumption. If it were normal, we would just use a normal distribution to approximate the sampling distribution.) In this case we could use the standard deviation of the bootstrap distribution to construct a confidence interval for the slope coefficient.

```
zs <- qnorm(c(0.025, 0.975))
coef(fm)["Mileage"] + zs * sd(~Mileage, data=bootstrap)
```

Method #2

The second method does not require that the bootstrap distribution be normal, but works best when it is roughly symmetric. In this case we simply use the percentiles of the bootstrap distribution to build confidence intervals. This method makes the most sense in the most cases.

```
qdata(~Mileage, p=c(0.025, 0.975), data=bootstrap)
```

Method #3

If the bootstrap distribution is skewed, then we need to modify the second method to work in reverse. This is because our estimate may already differ from the true population parameter.

The bootstrap distribution for the slope coefficient in this case, however, is not skewed.

```
qs <- qdata(~Mileage, p = c(0.025, 0.975), data=bootstrap)$quantile
coef(fm)["Mileage"] - (qs - coef(fm)["Mileage"])
```

- Create a bootstrap sample of at least 2000 and construct the three confidence intervals discussed above. How do they differ from the typical confidence interval?
- Construct bootstrapped confidence intervals for the $GPA \sim SATV$ and $GPA \sim SATM$ models.

Further Reading

- Tim Hesterberg's paper on *What Teachers Should Know about the Bootstrap*
- Brad Efron's original paper