Agenda

1. Motivation
2. Randomization Tests
3. Bootstrap for Regression

Refresher on distributions There are many theoretical distributions we rely on in statistics, including

- Normal distribution
- Student's t-distribution
- F-distribution

```
require(mosaic)
xpnorm(2)
xpf(6, df1=4, df2=10)
xpt(2, df=4)
```

All of these distributions approach one another as $df \to \text{inf}$. How do the t- and F-distributions change as the degrees of freedom increase?

Motivation of randomization and permutation Draw some pictures of what the two methods look like in general.

Randomization (Permutation) Tests Recall that the validity of t-tests, etc. in MLR models rely on assumptions that may not be met. If these assumptions are not met, we still want to test for significance! A *randomization* (or permutation) test is a non-parametric way to approximate the null distribution of a test statistic. The key idea is to use randomization to scramble any existing relationships between variables.

For example, we denote the observed correlation coefficient between X and Y as r_{XY}. With real data r_{XY} will never be exactly 0, but how do we know that the true (unknown) correlation between X and Y, denoted ρ_{XY} is not equal to 0? The classical method assumes normality and uses the t-statistic

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

In a randomization test, our null hypthesis is that $\rho_{XY} = 0$, and in this case X and Y are not correlated. So then it wouldn't matter if we bothered to keep the X column lined up with the corresponding entries for the Y column. In fact, we might just as well shuffle the X's. The distribution of observed correlation coefficients when we do this provides us with an understanding of the null distribution. If r_{XY} is unlikely in that distribution, then we might doubt our hypothesis.

```
require(Stat2Data)

cor(Price ~ Age, data=ThreeCars)
cor(Price ~ shuffle(Age), data=ThreeCars)
rand.test <- do(5000) * cor(Price ~ shuffle(Age), data=ThreeCars)
densityplot(~cor, data=rand.test)
```

A similar procedure can be used to test the significance of nearly any sample statistic!

Bootstrap The bootstrap is a versatile computational technique that was discovered in just 1979. If the assumption of normality of residuals is not met, we can generate a *non-parametric* sampling distribution for virtually any statistic by sampling *with replacement* from the original data. The distribution of these statistics is called a *bootstrap* distribution, and is an approximation of the sampling distribution!

General procedure:

- Take bootstrap sample
- Compute statistic
- Repeat 5000 or more times
- Put original statistic in context of bootstrap distribution

Bootstrapped Confidence Intervals

1. CI based on standard deviation of bootstrapped statistics
   - point estimate $\pm z^*_{\alpha/2} \cdot sd_{bootstrap}$
   - Only makes sense when bootstrap distribution is approximately normal
2. CI based on quantiles of bootstrap distribution
   - $[q_L, q_U]$
   - Works when bootstrap distribution is approximately symmetric
3. CI based on reversing quantiles of bootstrap distribution
   - $[x - (q_U - x), x + (x - q_L)]$
   - Works when bootstrap distribution is skewed

Lab code Reproduced here for notetaking

```
require(mosaic)
require(Stat2Data)
data(FirstYearGPA)

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

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

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

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

# Do this 1000 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)

pdata(~cor, q=cor.actual, data = rtest)
qdata(~cor, p=c(0.025, 0.975), data = rtest)

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

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)), cex=0.75)
           panel.text(75, 75,
                      paste("sd intercept\n",
                            round(sd(~Intercept, data=bootstrap), 6)), cex=0.75)
           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)

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

fm <- lm(Price ~ Mileage, data=PorschePrice)
confint(fm)

# 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)

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

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

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