Correlation and regression

Patrick Breheny

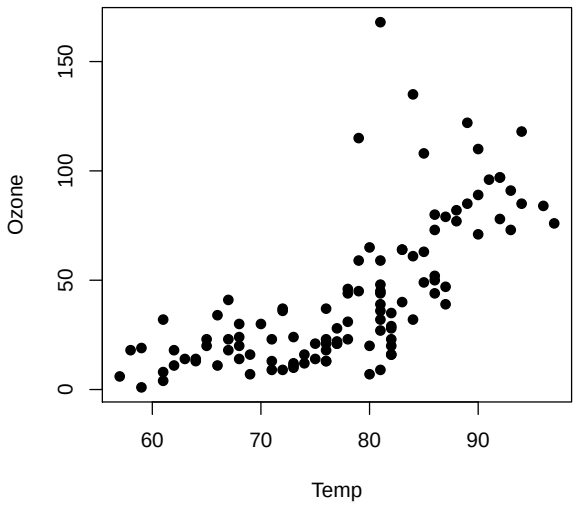
December 1, 2016

Today's lab is about correlation and regression. It will be somewhat shorter than some of our other labs, as I would also like to spend some time discussing another useful statistical software program called SAS. Our goals for this section of the lab are to cover R's built-in functions for calculating and carrying out inference with respect to the correlation coefficient and regression coefficient, then do a quick simulation comparing various methods for constructing confidence intervals for ρ.

1 Correlation

For our example today, we'll use a standard R data set (see `?airquality` for more details) on the relationship between ozone concentrations and various atmospheric factors; we'll focus on temperature. Ozone, while beneficial in the stratosphere in terms of providing protection from ultraviolet radiation, is a respiratory hazard at low altitudes, predominantly emitted by the burning of fossil fuels. Let's start by looking at the data (always a wise first step); note the use of `complete.cases` to filter out missing data.

```
> Ozone <- airquality[complete.cases(airquality), "Ozone"]
> Temp <- airquality[complete.cases(airquality), "Temp"]
> plot(Temp, Ozone, pch=19)
```



There appears to be a pretty clear positive relationship between the two quantities: Ozone concentratons are higher on hotter days. We can quantify this relationship using the correlation coefficient:

```
> cor(Ozone, Temp)

[1] 0.6985414
```

So, a pretty strong correlation of ≈ 0.7. We can carry out inference – testing, confidence intervals – with the `cor.test` function, which uses the Fisher Z transformation we discussed in class:

```
> cor.test(Ozone, Temp)


Pearson's product-moment correlation

data:  Ozone and Temp
t = 10.192, df = 109, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5888139 0.7829869
sample estimates:
      cor
0.6985414

> ## By hand
> r <- cor(Ozone, Temp)
> n <- length(Ozone)
> ci.trans <- atanh(r) + qnorm(c(.025, .975))*sqrt(1/(n-3))
> tanh(ci.trans)

[1] 0.5888139 0.7829869
```

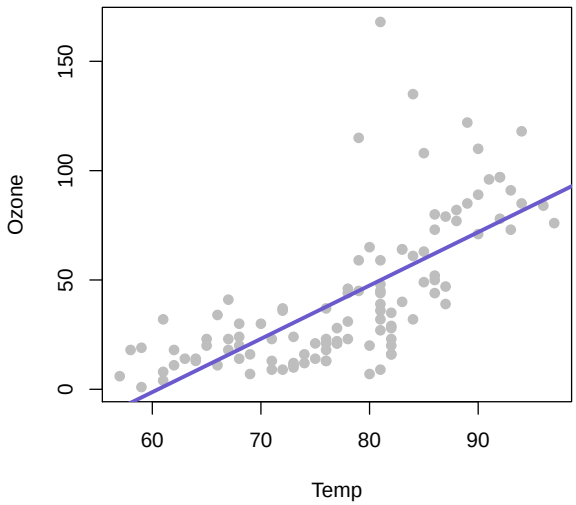
The interval tells us that we can be reasonably confident that the true correlation between ozone and temperature is between 0.6 and 0.8.

2 Regression

Now, let's approach the problem from a regression standpoint, and consider predicting ozone concentrations based on temperature. This is not just a theoretical concern – ozone is hazardous to health, especially for asthmatics, so predicting ozone concentrations is a meaningful public health objective. The function for carrying out linear regression in `R` is called `lm`, which stands for *linear model*. `lm` works slightly differently from the functions we have seen so far in that it separates the estimation (model fitting) from all the things (testing, confidence intervals, plotting, making predictions) you might want to do with the model once you've fit it. This might seem strange at first, but it's actually quite convenient. To illustrate:

```
> fit <- lm(Ozone~Temp)
> coef(fit)

(Intercept)        Temp
 -147.64607     2.43911

> confint(fit)

                 2.5 %      97.5 %
(Intercept) -184.818372 -110.473773
Temp           1.964787    2.913433
```

```
> summary(fit)


Call:
lm(formula = Ozone ~ Temp)

Residuals:
    Min      1Q  Median      3Q     Max
-40.922 -17.459  -0.874  10.444 118.078

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -147.6461    18.7553  -7.872 2.76e-12 ***
Temp           2.4391     0.2393  10.192  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.92 on 109 degrees of freedom
Multiple R-squared:  0.488,Adjusted R-squared:  0.4833
F-statistic: 103.9 on 1 and 109 DF,  p-value: < 2.2e-16

> predict(fit, newdata=data.frame(Temp=c(60, 70, 80, 90)))

        1         2         3         4
-1.299478 23.091621 47.482720 71.873819

> ## Plot the regression line
> plot(Temp, Ozone, pch=19, col="gray")
> abline(fit, col="slateblue", lwd=3)
```



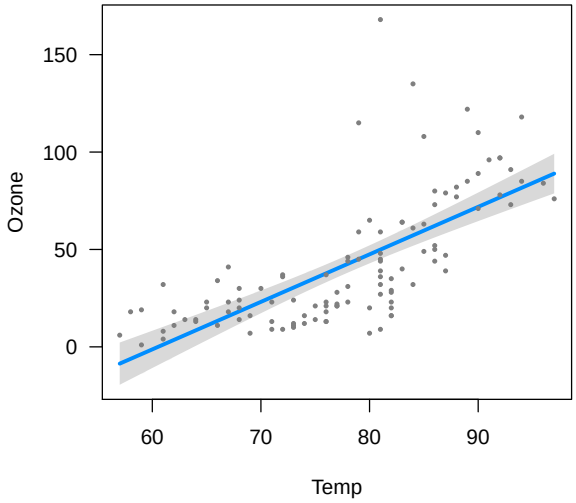
In particular, note that the regression coefficient is 2.4; for every additional degree of temperature, the expected ozone levels rises by 2.4 units (parts per billion). As we discussed in class, this is the same basic

statement we obtain from the correlation coefficient – which says that for every 1 SD change in temperature, the expected ozone level changes by 0.7 SDs – just in different units. We can easily convert between the two:

```
> coef(fit)[2]*sd(Temp)/sd(Ozone) ## Same as correlation coefficient

     Temp
0.6985414
```

Also note that `lm` works with `abline`, which makes plotting convenient. Another way to visualize regression models is with the `visreg` package:

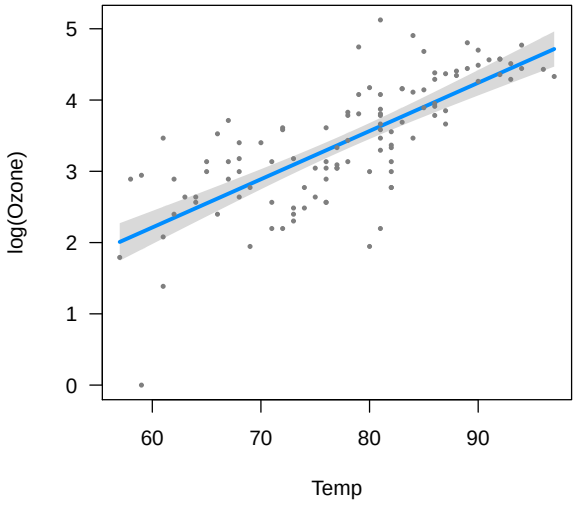
```
> require(visreg)
> visreg(fit)
```



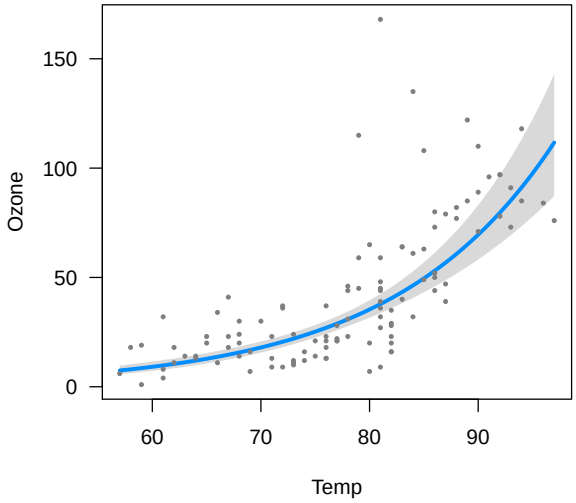
`visreg` plots the points and the regression line, but also adds 95% confidence intervals for the regression line in the form of a shaded band that lets you see the uncertainty of the fit. An intuitive, yet interesting, aspect of regression is that uncertainty about the mean is greater at the extremes (very high and low temperatures) than it is in the middle (average temperatures).

Looking carefully at these various plots, we can see a potential problem with our linear model – namely, at low temperatures, our model predicts negative ozone concentrations, which is nonsensical. A potential improvement to our model would be to transform ozone concentrations and model the log of the ozone concentration rather than the ozone itself:

```
> fit <- lm(log(Ozone)~Temp)
> visreg(fit)
```




```
> visreg(fit, trans=exp, partial=TRUE, ylab="Ozone") ## Transforming back to original scale
```



This appears to perhaps fit the data a little better, especially at low temperatures. There are many other ways one could address this issue besides the log transformation we considered here; the fascinating world of regression modeling will be the subject of the next course in this sequence, Biostatistical Methods II (BIOS 5720).

3 Simulation: Confidence intervals for ρ

Lastly, I'd like to carry out a simulation of various possible ways of calculating a confidence interval for the correlation coefficient, ρ. In class, I mentioned that one could construct Wald or Score intervals based on

the relationship

$$\frac{\hat{\rho} - \rho}{(1-\rho^2)/\sqrt{n}} \sim \mathrm{N}(0,1),$$

but that the Fisher Z-transformation approach was better. But is it really, or was I lying? Let's carry out a simulation study!

First, we'll have to write functions that calculate Wald and score intervals for the correlation coefficient:

```
> cor.wald <- function(x, y, alpha=.05) {
+   r <- cor(x,y)
+   n <- length(x)
+   SE <- (1-r^2)/sqrt(n)
+   z <- qnorm(c(alpha/2, 1-alpha/2))
+   r + z*SE
+ }
> cor.score <- function(x, y, alpha=.05) {
+   r <- cor(x,y)
+   n <- length(x)
+   f.u <- function(x) (r-x)/((1-x^2)/sqrt(n)) - qnorm(alpha/2)
+   f.l <- function(x) (r-x)/((1-x^2)/sqrt(n)) - qnorm(1-alpha/2)
+   c(uniroot(f.l, c(-1,1))$root, uniroot(f.u, c(-1,1))$root)
+ }
```

Note that we could analytically solve for the score interval, but we'd have to do some inspection of discriminants to make sure we're choosing the correct root of the quadratic equation, so I'm just using `uniroot` for simplicity. The Fisher Z approach is coded for us already (`cor.test`). I'll add one more method that could conceivably be used for constructing confidence intervals for ρ: constructing confidence intervals for the regression coefficient, β, and then transforming that confidence interval to a confidence interval for ρ:

```
> cor.lm <- function(x, y, alpha=0.05) {
+   fit <- lm(y~x)
+   ci <- confint(fit, 2, level=1-alpha)
+   ci*sd(x)/sd(y)
+ }
```

Note that all of these intervals seem to be in reasonable agreement for the air quality data:

```
> cor.test(Ozone, Temp)$conf.int

[1] 0.5888139 0.7829869
attr(,"conf.level")
[1] 0.95

> cor.wald(Ozone, Temp)

[1] 0.6032858 0.7937970

> cor.score(Ozone, Temp)

[1] 0.5737665 0.7733352

> cor.lm(Ozone, Temp)

      2.5 %    97.5 %
x 0.5626991 0.8343838
```

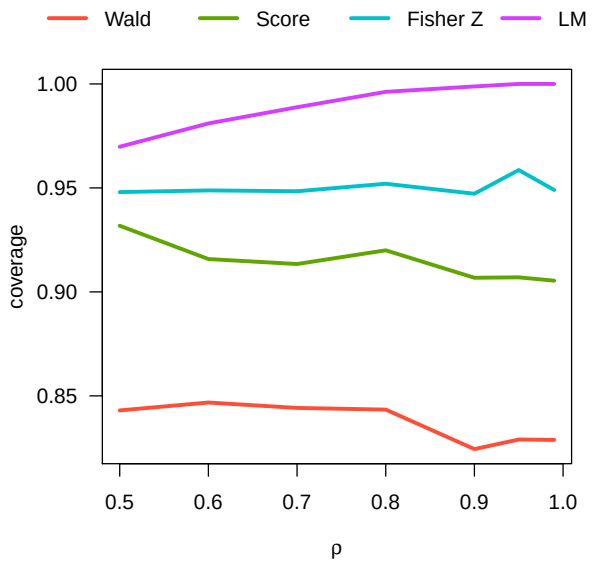
We'll also need a function to generate correlated data. There are packages (e.g., `mvtnorm`) for this, but I think it's instructive to just construct our own:

```
> genData <- function(n, rho) {
+   a <- sqrt(rho/(1-rho))
+   U <- rnorm(n)
+   V <- matrix(rnorm(n*2), n, 2)
+   (V + a*U)/sqrt(1+a^2)
+ }
```

OK, now we're ready to carry out the simulation:

```
> N <- 5000
> n <- 10
> Names <- c("Wald", "Score", "Fisher Z", "LM")
> covered <- matrix(NA, N, 4)
> rho <- c(0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99)
> coverage <- matrix(NA, length(rho), 4)
> colnames(covered) <- colnames(coverage) <- Names
> pb <- txtProgressBar(1, length(rho), style=3)
> for (j in 1:length(rho)) {
+   for (i in 1:N) {
+     X <- genData(n, rho=rho[j])
+     x <- X[,1]
+     y <- X[,2]
+     ci <- cor.wald(x,y)
+     covered[i,1] <- (ci[1] < rho[j]) & (ci[2] > rho[j])
+     ci <- cor.score(x,y)
+     covered[i,2] <- (ci[1] < rho[j]) & (ci[2] > rho[j])
+     ci <- cor.test(x,y)$conf.int
+     covered[i,3] <- (ci[1] < rho[j]) & (ci[2] > rho[j])
+     ci <- cor.lm(x,y)
+     covered[i,4] <- (ci[1] < rho[j]) & (ci[2] > rho[j])
+   }
+   coverage[j,] <- apply(covered, 2, mean)
+   setTxtProgressBar(pb, j)
+ }
```

```
> source("http://myweb.uiowa.edu/pbreheny/5710/f16/labs/toplegend.R")
> col <- hcl(seq(15, 375, len=5), l=60, c=150)[1:4]
> matplot(rho, coverage, type="l", lty=1, lwd=3, col=col, xlab=expression(rho), las=1)
> toplegend(legend=Names, lwd=3, col=col)
```



The Wald interval is far too liberal (low coverage) and the linear model interval is far too conservative. The score interval isn't too bad, but the Fisher Z interval is close to perfect, demonstrating the impressive accuracy of variance-stabilizing transformations.