

Homework Assignment 6: Nice Demo City, But Will It Scale?

36-402, Advanced Data Analysis, Spring 2011

SOLUTIONS

1. ANSWER: Taking the log of both sides gives

$$\begin{aligned}\log y = \log \frac{Y}{N} &= \log Y - \log N \\ &\approx \log(cN^b) - \log N \\ &= \log c + \log N^b - \log N \\ &= \log c + b \log N - \log N \\ &= \beta_0 + \beta_1 \log N\end{aligned}$$

where we have taken $\beta_0 = \log c$ and $\beta_1 = b - 1$.

2. ANSWER:

```
#### Problem 2
# Get the data:
gmp.data = read.csv(file =
  "http://www.stat.cmu.edu/~cshalizi/402/hw/06/gmp_2006.csv")
pcgmp.data = read.csv(file =
  "http://www.stat.cmu.edu/~cshalizi/402/hw/06/pcgmp_2006.csv")

# Using summary(gmp.data) helps to show that X2006 is the column of interest
# Check that both data sets are ordered the same way:
sum(gmp.data$Metropolitan != pcgmp.data$Metropolitan)

# The N, in millions: Divide gdp by gdp-per-capita
N = gmp.data$X2006/pcgmp.data$X2006
# The N, corrected (since gdp is in millions)
N = N*1000000
# Observe that the result matches what Cosma says it should:
summary(N)

# Compute variance of log y
y = pcgmp.data$X2006 # same as Y/N
```

```
log.y = log(y)
> var(log.y)
[1] 0.0714536
```

3. ANSWER:

```
# Compute log population:
log.N = log(N)
# Fit a linear model
lm.3 = lm(log.y ~ log.N)
# The output of summary(lm.3) includes the following
Coefficients:
      Estimate
(Intercept)  8.79641
log.N        0.12418
```

The first thing to notice is that the adjusted R-squared value is only about 0.24 (from the model summary). This means that the model does not explain a very large portion of the variability in the response. But it is still possible that the model is close to optimal, unless we have reason to think that the single predictor variable contains enough information to explain much more than 24% of the variability.

The model summary gives an estimate of the intercept $\hat{\beta}_0 = 8.80$ and the $\log.N$ coefficient $\hat{\beta}_1 = 0.12$. Using the expressions at the end of part (1) we can generate the corresponding estimates of c and b for the equation $Y \approx cN^b$ which motivates the log-linear model. We get $\hat{c} = \exp(\hat{\beta}_0) = 6610$ and $\hat{b} = \hat{\beta}_1 + 1 = 1.12$. There is nothing obvious here to suggest that the model $Y \approx 6610N^{1.12}$ is not a reasonable approximation.

Finally, we compute the MSE for the model in the log scale:

```
> mean(lm.3$residuals^2)
[1] 0.05392461
```

4. ANSWER:

```
plot(N, y, main = "The fitted power-law curve",
     cex = 0.4, lwd = 2, pch = 16)
# The fitted values are the exponentials of the log fit:
y.fit = exp(fitted(lm.3))
fitted.points = cbind(N,y.fit)
fitted.points = fitted.points[order(N),]
lines(fitted.points, lwd = 2, col = 2)
```

See Figure 1.

5. ANSWER:

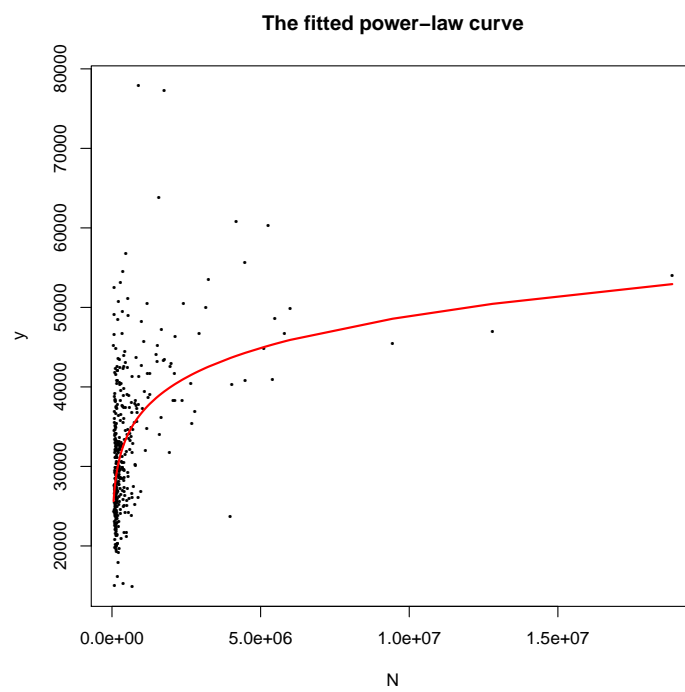


Figure 1:

```

# For the nonparametric fit we will use the function smooth.spline,
# letting R use a default process to choose the bandwidth (or
# number of knots).
spline.mod = smooth.spline(log.y~log.N)
spline.fit = predict(spline.mod, log.N)

# MSE:
> mean((spline.fit$y - log.y)^2)
[1] 0.05078971

# comparison: plotting may help
plot(N, y)
fitted.points = cbind(exp(spline.fit$x),exp(spline.fit$y))
fitted.points = fitted.points[order(fitted.points[,1]),]
lines(fitted.points, lwd = 2, col = 2)
# RESULTS NOT SHOWN HERE

```

In words, the spline curve is rather similar to the power-law curve. The most noticeable difference is that the spline curve is not monotonically increasing; near small populations the fit sharply decreases.

6. ANSWER:

```

# We have completed steps 1-3 in the method outlined
# in lecture 10 section 1
# Step 4: How much bigger is the parametric MSE from the nonparametric MSE?
t.hat = mean(lm.3$residuals^2) - mean((spline.fit$y - log.y)^2)

# For steps 5-9, we need some functions:

# make.fake (below) is a function that simulates
# faked data based on the parametric model lm.3
# The new x's are the same as the old x's; the new y's are the lm.3
# fitted values plus Gaussian noise with variance approximated by the
# in-sample MSE
make.fake = function(){
  fake.log.y = fitted(lm.3) +
    + rnorm(length(log.y), mean = 0, sd = sqrt(mean(lm.3$residuals^2)))
  return(fake.log.y)
}

# This function estimates the parametric model on the simulated data:
sim.parametric = function(fake.log.y){
  lm.on.fake = lm(fake.log.y ~ log.N)
  MSE = mean(lm.on.fake$residuals^2)
  return(MSE)
}

```

```

    }

# This function estimates the nonparametric model on the simulated data:
sim.non.parametric = function(fake.log.y){
  spline.mod = smooth.spline(fake.log.y~log.N)
  spline.fit = predict(spline.mod, log.N)
  MSE = mean((spline.fit$y - fake.log.y)^2)
  return(MSE)
}

# Use the functions above to simulate t.hat as T.hat
sim.T.hat = function(){
  fake.log.y      = make.fake()
  MSE.parametric = sim.parametric(fake.log.y)
  MSE.non.parametric = sim.non.parametric(fake.log.y)
  T.hat = MSE.parametric - MSE.non.parametric
  return(T.hat)
}

# Produce 1000 T.hat replicates
T.hat.replicates = replicate(1000, sim.T.hat())

# The p-value
> (1+sum(T.hat.replicates > t.hat))/(1+length(T.hat.replicates))
[1] 0.4145854

```

The p-value is 0.04. Note that the p-value is small whenever most of the simulated values \hat{T} are smaller than the observed \hat{t} , where \hat{t} is the amount by which the nonparametric model does better than the parametric model (as measured by MSE). If nearly all of the simulated values were less than the observed value, we might conclude that the observed value is large enough to be statistically significant, indicating that the nonparametric model is better. Since the p -value is small, we have reasonable evidence that the parametric model is worse than the non-parametric model. (A p -value of only 4% is not however especially overwhelming evidence.)

Further diagnosis of what is wrong with the parametric model (which is not required by the problem) could involve either looking at the linearity or the Gaussian distribution of noise around the regression. The shape of the spline strongly suggests nonlinearity (at least for small city sizes), but rather than leaping to that conclusion we could look at replacing the Gaussian noise in the simulation with resampling the residuals. (This would involve changing the `make.fake` function above.) Doing so in this case does not change the p -value very much. This still assumes the noise around the regression has the same distribution for all city sizes, so saving the log-linear, power-law model is only going to be possible if we allow a lot of heteroskedasticity (and perhaps not even then).

We emphasize that any model that relies only on population will explain only a small portion of the variability in income, because population does not appear to be especially informative.

A crucial part of the simulation process that is easy to forget since R does it for us here is that the nonparametric model must have its bandwidth chosen by cross-validation. If the nonparametric model is allowed to overfit, the nonparametric MSE will always be far less than the parametric MSE, leading us to reject the parametric model every time.