

Midterm Exam 1: Urban Scaling, Continued

36-402, Advanced Data Analysis, Spring 2011

SOLUTIONS

General set-up:

```
gmp = read.csv(file = "gmp-2006.csv")
```

Your data file was derived from this data file, plus or minus 4% noise for each observation.

1. ANSWER: The basic scatterplot function will drop points where either the horizontal or vertical coordinate is NA. Some but not all smoothers will also do so. However, `smooth.spline` will not. Because extracting the points where neither variable is a repetitive operation, it's better to turn it into a function. Fortunately, R has already done this for us, in the `na.omit` function. Thus `na.omit(gmp)` would be a version of the data set with only rows with complete data. This is however too aggressive for this problem and the next — if we're interested in finance, there's no sense dropping a city because we don't have information about professional services.

```
## Here is a function for adding spline fits to the desired plots
add.spline.fit = function(x.name = "pop", y.name = "finance"){
  # Remove rows of data with NA y-values
  relevant.data = na.omit(gmp[,c(x.name,y.name)])
  # note selecting just the columns needed for the plot:
  # relevant.data will have two columns, and x will come first
  # Fit the spline (using generalized cross validation)
  my.spline = smooth.spline(x=relevant.data[,1],y=relevant.data[,2])
  # Add to the current plot
  lines(my.spline, lwd = 2, col = 2)
  # The "$x" attribute of my.spline has the unique values of x IN ORDER
  # while the "$y" has the corresponding fitted values; since the lines()
  # function looks for either separate x and y arguments, or one argument with
  # components named x and y, this works.
}

# Open up a 4-pane plotting frame
par(mfrow = c(2,2))
```

```

plot(gmp$pop, gmp$finance,
     cex=0.4, lwd=1.4, pch=16,
     main = "Finance",
     xlab = "Population", ylab = "Share",log="x")
add.spline.fit(y.name = "finance")

plot(gmp$pop, gmp$professional.and.technical,
     cex=0.4, lwd=1.4, pch=16,
     main = "Professional and technical services",
     xlab = "Population", ylab = "Share",log="x")
add.spline.fit(y.name = "professional.and.technical")

plot(gmp$pop, gmp$ict,
     cex=0.4, lwd=1.4, pch=16,
     main = "Information technology",
     xlab = "Population", ylab = "Share",log="x")
add.spline.fit(y.name = "ict")

plot(gmp$pop, gmp$management,
     cex=0.4, lwd=1.4, pch=16,
     main = "Management",
     xlab = "Population", ylab = "Share",log="x")
add.spline.fit(y.name = "management")

```

See Figure 1. The relationship between `finance` and `pop` is nearly monotonic; increasing `pop` generally corresponds to increasing `finance`. For professional services and management, there is a leveling off on the right half of each plot, but no decline. Only information technology lacks a clear pattern.

Notice that all the plots show population on a logarithmic scale — this is just for visual clarity and is not required. Since there are not that many large cities, using an ordinary linear scale for population would bunch most of the data up on the left-hand side of each plot, making it harder to see what's happening.

If we used a different smoother, and still wanted to use `lines` to add the smoothed values to the plot, we would have to make sure they came in some sort of sensible order — otherwise we'd get line segments connecting points for cities in *alphabetical* order, which is a mess. Acceptable alternatives include: using `points` on the fitted values; re-arranging the fitted values in order of increasing (or decreasing) population; using `predict` to get values at regularly spaced and increasing grid points, and then `lines` to plot them (examples in the notes); or even just observing that there was a problem with the plot, which you did not know how to solve.

2. ANSWER: Re-use the machinery from the last problem.

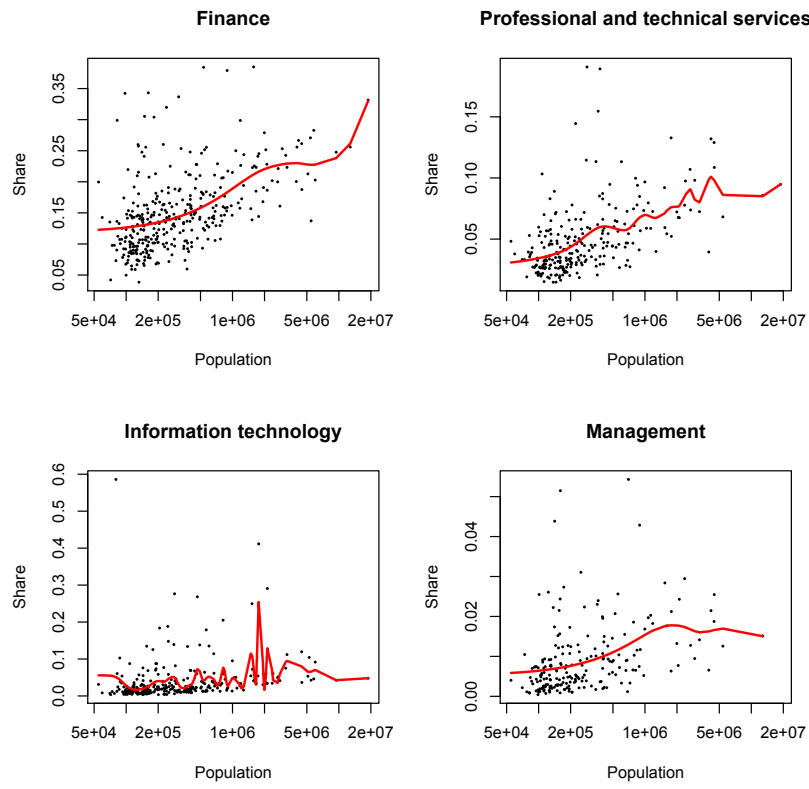


Figure 1:

```

par(mfrow = c(2,2))

plot(gmp$finance, gmp$pcgmp,
     cex=0.4, lwd=1.4, pch=16,
     main = "pcgmp vs finance share",
     xlab = "Share", ylab = "pcgmp")
add.spline.fit(x.name = "finance", y.name = "pcgmp")

plot(gmp$professional.and.technical, gmp$pcgmp,
     cex=0.4, lwd=1.4, pch=16,
     main = "pcgmp vs professional share",
     xlab = "Share", ylab = "pcgmp")
add.spline.fit(x.name = "professional.and.technical", y.name = "pcgmp")

plot(gmp$ict, gmp$pcgmp,
     cex=0.4, lwd=1.4, pch=16,
     main = "pcgmp vs ict share",
     xlab = "Share", ylab = "pcgmp")
add.spline.fit(x.name = "ict", y.name = "pcgmp")

plot(gmp$management, gmp$pcgmp,
     cex=0.4, lwd=1.4, pch=16,
     main = "pcgmp vs management share",
     xlab = "Share", ylab = "pcgmp")
add.spline.fit(x.name = "management", y.name = "pcgmp")

```

See Figure 2. The plots generally suggest that increasing the share for specialist service industries corresponds to increase productivity. The relationship is not especially clean for professional and technical services. For ICT, the only non-monotonicity is driven by the contrast between the city with the second-highest share of ICT (San Jose-Sunnyvale-Santa Clara, CA) and that with the highest share (Corvallis, OR) — which is the unsurprising fact that Silicon Valley is quite rich.

3. (a) ANSWER: The first two terms, α_0 and $b \ln N$, are the power-law scaling part of the model. These terms are parametric: they assume a specific functional form for the relationship between the response and $\ln N$, involving only a finite number (in this case, two) unknown constants. The f_j are the non-parametric terms — or alternatively, “infinitely parametric”, since in principle they need to be estimated at infinitely many points, and we (or the `gam` function) use smoothing to interpolate between the points we see. — It’s defensible to argue that α_0 is not really a parametric term, since it just does the work of centering the fitted values.
- (b) ANSWER: We used `gam` from the `mgcv` package:

```

library(mgcv)

```

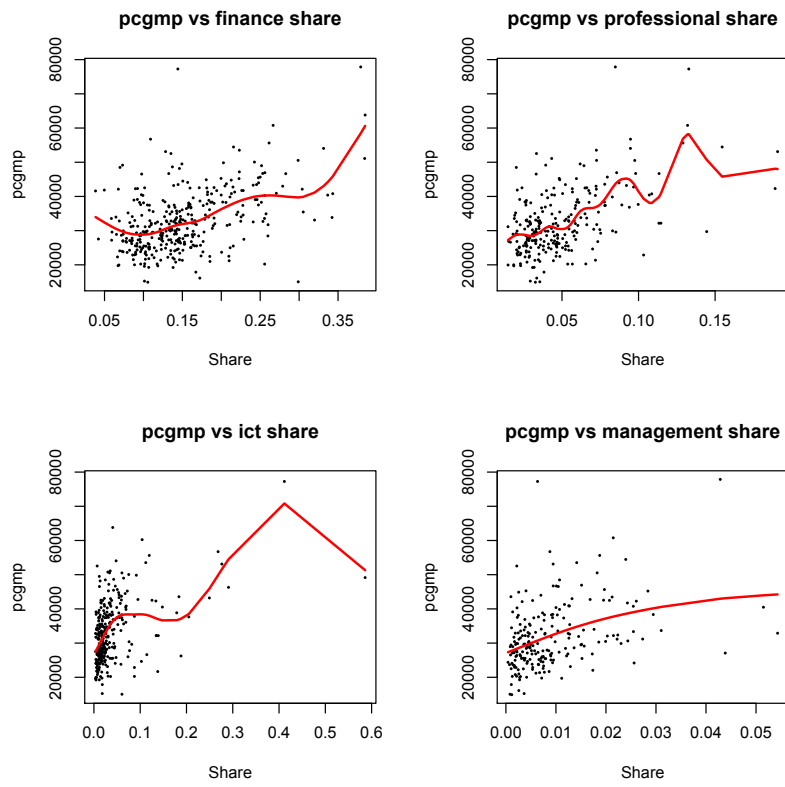


Figure 2:

```
gam.3 = gam(log(pcgmp) ~ log(pop) + s(finance) + s(professional.and.technical)
+ s(ict) + s(management), data = gmp)
```

- (c) ANSWER: Using `summary(gam.3)`, we find that the intercept α_0 is estimated to be 10.37, b is estimated as -0.0027 , and the residual standard error is given as the sample standard deviation of the residuals:

```
> sd(gam.3$residuals)
[1] 0.2183118
```

(Equivalently, `sqrt(mean(residuals(gam.3)2))`.)

- (d) ANSWER:

```
par(mfrow = c(2,2))
plot(gam.3,seWithMean=TRUE)
```

(Alternately, `plot(gam.3,pages=1,seWithMean=TRUE)`, without the `par`.) See Figure 3. Just as in Figure 2, the relationship between industry share and productivity seems generally positive. The relationship is nonlinear for information technology and for management. Notice however that the confidence bands tend to get quite wide at the right (high-share) ends, since there are not many observations there.

- (e) ANSWER:

```
par(mfrow = c(2,1))
qqnorm(residuals(gam.3))
qqline(residuals(gam.3))
plot(fitted(gam.3), residuals(gam.3),
     main = "Residuals vs. fitted values",
     xlab = "Fitted values", ylab = "Residuals")
rug(fitted(gam.3))
abline(h= 0)
```

The $Q-Q$ plot in Figure 4 shows that while the center of the distribution of residuals is not too far from Gaussian, the tails are somewhat heavier than one would expect from a Gaussian. A Shapiro-Wilk test is inconclusive ($p = 0.23$).¹ It would also be reasonable to use a histogram or a density estimate plot here, provided it was compared with a matched Gaussian.

The scatterplot looks reasonable — there may be a faint tendency for less variance at higher fitted values, hinting at heteroskedasticity, but since there are few observations with high fitted values it is hard to tell. Of course, it is not possible to say whether the residuals are Gaussian from the *scatterplot*.

¹A more powerful but more complicated procedure called a “data-driven smooth test” rejects the Gaussian, $p = 0.02$. Time permitting, we will cover this later in the course, but in the meanwhile see the package `ddst`.

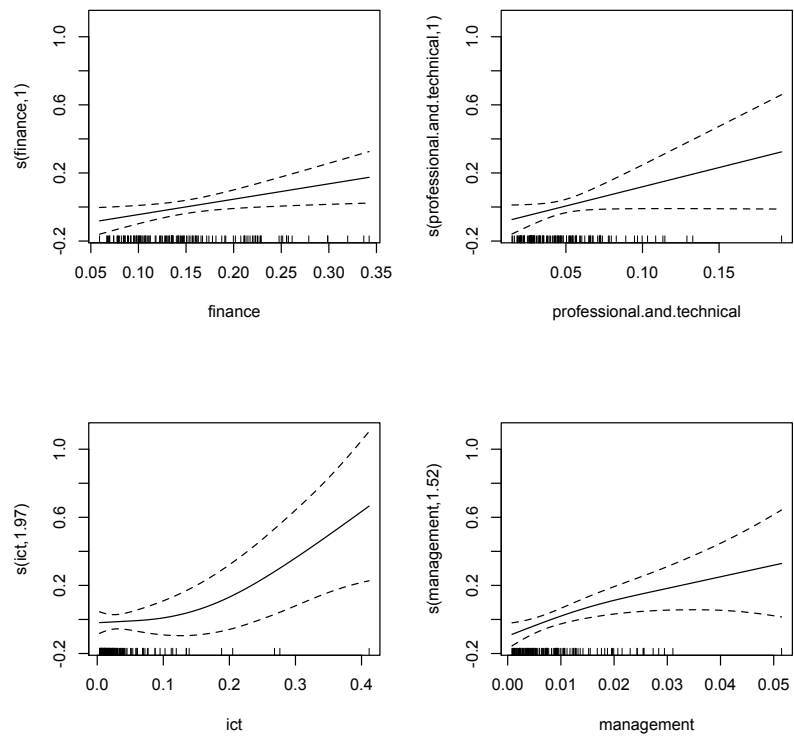


Figure 3:

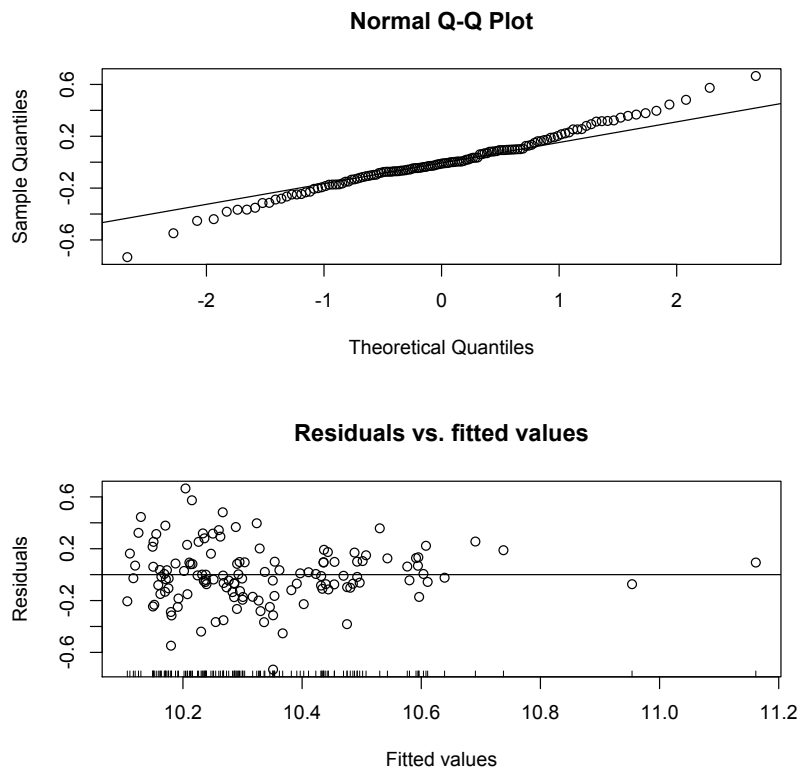


Figure 4:

- (f) ANSWER: The nominal standard error for b is 0.028 with p-value 0.93, suggesting that b is not a significant term, given the others. (Remember, this means that *if* b were truly zero, estimating the model with b and with the four smoothing terms would often give us a \hat{b} at least this big.) One way to interpret this is that, given the presence of the non-parametric terms in the model, the variable `pop` does not help predict the average productivity level.

The extent to which we can trust the calculation R is doing depends on whether the assumptions it makes about the data are plausible. These assumptions include (i) Gaussian noise, with (ii) the same variance for all values of the explanatory variables (homoskedasticity) and (iii) independent noise across cities. Items (i) and (ii) were addressed in part (e) and look reasonable (if not outstanding). A more tricky issue is the missing data for industries. If this is missing at random, then it essentially makes no difference. But if the values are *not* missing at random, it could introduce quite substantial errors into calculations for the p -values. At the least this would deserve investigation.

4. (a) ANSWER: First, let's reproduce the power-law scaling model with the current data

```
lm.p.law = lm(log(pcgmp) ~ log(pop), data = gmp)
# Get the coefficient b:
coefficients(lm.p.law)[2] # = 0.124
```

The problem asks us what the model predicts would happen if the population increased. This is *not* about re-estimating the model with larger population values.

The direct way of doing this is to use the `predict` function. This takes two arguments. The first is a fitted model (as from `lm`, `npreg`, `gam`, etc.). The second is a data frame with columns whose names match the terms appearing on the right-hand (explanatory) side of the formula for the model. (If this argument is omitted, `predict` just gives back the original fitted values.) These values are fed through the formula, including any transformations that specifies, and the output is a vector of predictions, one for each row of the new data.²

```
old.p.law.preds <- predict(lm.p.law, newdata=gmp)
```

To see what happens when population increases, we make a new data frame. One way to do this is:

```
new.p.law.preds <- predict(lm.p.law, newdata=data.frame(pop=1.1*gmp$pop))
```

Another would be to make the whole new data frame and keep it around:

²There have been examples of using `predict` in the notes, for the kernel regressions from `npreg`.

```

gmp.plus.10 <- gmp
gmp.plus.10$pop <- 1.1*gmp.plus.10$pop
new.p.law.preds <- predict(lm.p.law,newdata=gmp.plus.10)

```

Now we just look at the average change from old to new predictions:

```

> mean(new.p.law.preds - old.p.law.preds)
[1] 0.01183608

```

(You can check that the answer doesn't depend on how the new predictions are calculated.)

This approach will work for any kind of model which has a `predict` method. There is a short-cut, however, in this case, since we are looking at a linear model without interactions. Since, for any numbers M and N , $\log MN = \log M + \log N$, increasing population by a factor of 1.1 is the same as adding $\log 1.1$ to $\log N$. In the power law model, then, the change in the predicted value is

$$\begin{aligned}
\Delta \widehat{\log y} &= (b_0 + b_1 \log 1.1N) - (b_0 + b_1 \log N) \\
&= (b_0 + b_1 \log 1.1 + b_1 \log N) - (b_0 + b_1 \log N) \\
&= b_1 \log 1.1
\end{aligned}$$

no matter what N was to start with. This is

```

> log(1.1) * coefficients(lm.p.law)[2]
log(pop)
0.01183608

```

which agrees with what we got by direct calculation, as it should.

- (b) ANSWER: Again, we can use direct calculation. Because the log-additive model needs the industry shares, the new data frame we give `predict` must have those columns; so we'll use the `gmp.plus.10` data frame made in part (a); but we need to drop rows with NAs, since the model doesn't know what to do with them.

```

old.lam.preds <- predict(gam.3) # Gives fitted values without newdata
new.lam.preds <- predict(gam.3,newdata=na.omit(gmp.plus.10))

```

Taking the mean change:

```

> mean(new.lam.preds - old.lam.preds)
[1] -0.0002534905

```

Alternatively, the shortcut for part (a) still works here, because we (i) we are making an additive change to $\log N$ and to no other input variable, (ii) the response, in the model, depends linearly on $\log N$, and (iii) there is no interaction in the model between $\log N$ and any other variable. We do however need to extract the parametric coefficient b .

```
> log(1.1)*coefficients(gam.3) ["log(pop)"]
      log(pop)
-0.0002534905
```

Again, this agrees exactly with direct calculation.

- (c) ANSWER: The two models lead to different conclusions: The power-law model says that increases in population lead to (or at least predict) increases in productivity, whereas the additive model suggests that increases in population, all else being equal, lead to a slight decrease in productivity. (Notice that this matches the sign the two models give to the coefficient for $\log N$.)

To the extent that larger populations go along with higher productivity in the power-law model, the additive model suggests that this is because bigger cities have bigger shares of high-productivity industries. Since the data arise from observing how cities happen to be, rather than experimentally *making* cities be certain ways and seeing what happens, it is hazardous to draw causal conclusions from either model.

5. (a) ANSWER:

```
# in-sample MSE for log.y in problem 3
> mean(residuals(gam.3)^2)
[1] 0.047
# in-sample MSE squared error for log.y in power-law model
> mean(residuals(lm.p.law)^2)
[1] 0.054
```

We see that the in-sample MSE for the additive model is about 0.007 less than the in-sample MSE for the power-law model. This should be no surprise; the additive model includes all the terms of the power-law model and then some, meaning that additive model has the flexibility to fit the data more closely than the power-law model.

- (b) ANSWER: There are several ways to do this problem.

Since the log-additive model is more flexible than the power law model, and includes it as a special case, we cannot just compare in-sample MSEs. (See part (a) again.) We could however ask whether the *amount* by which the log-additive model does better in sample is compatible with the power law model actually being right. This is what the partial F -test from 401 checks for linear models (with Gaussian noise, etc.); similarly for the log-likelihood ratio test. The testing procedure of the notes for lecture 10 works along the same lines. Any of them could be used, but all of them would require simulating from the null model to get the right sampling distribution. (Just plugging in to a table for the F distribution would not work.) So this could be done along the lines of Lecture 10 and Homework 6, or of Homework 5.

Alternately, we could compare generalization ability more directly, by using cross-validation. The cross-validated MSEs for the two models would estimate how well each could generalize to new data, and, on average, the model with lower cross-validated error really does predict better. However, if we want to attach an actual confidence level to our choice, we cannot just do cross-validation once. We would need to know whether the difference in CV scores between the models is large or small compared to sampling noise. Since that difference won't follow any standard distribution, again we need to get it by simulating.

We will use the bootstrap to simulate the distribution for our estimate of the difference between MSEs. Suppose the difference is expressed as $\text{MSE}(\text{power law model}) - \text{MSE}(\text{additive model})$. Now, taking the power law model as the null model, we may conclude that the additive model is superior if 0 lies far (as measured in quantiles) out on the left tail of the distribution of bootstrapped values.

- (c) ANSWER: To fairly compare the two models, we will only use the cities for which complete data is available.

```
gmp.clean = na.omit(gmp)
```

We will also want a function that performs CV and returns an estimate of the difference in generalization errors.

```
do.nfold.cv = function(nfolds = 5, data=gmp.clean){
  # make test vs train labels
  case.folds = rep(1:nfolds,length.out=nrow(data))
  case.folds = sample(case.folds)
  fold.mses.gam = rep(0,nfolds)
  fold.mses.lm = fold.mses.gam
  for (fold in 1:nfolds) {
    # Split data into training and test data
    train = data[case.folds!=fold,]
    test = data[case.folds==fold,]
    # Fit the gam model:
    my.gam = gam(log(pcgmp) ~ log(pop) + s(finance)
      + s(professional.and.technical)
      + s(ict) + s(management), data = train)
    # Fit the old model:
    my.lm = lm(log(pcgmp) ~ log(pop), data = train)
    # Predict on the test set
    predictions.gam = predict(my.gam, newdata=test)
    predictions.lm = predict(my.lm, newdata=test)
    # We want to compare predictions from both models against the truth
    truth = log(test$pcgmp)
    # gam MSE:
```

```

    fold.mses.gam[fold] = mean((truth - predictions.gam)^2)
    # power-law MSE:
    fold.mses.lm[fold] = mean((truth - predictions.lm)^2)
  }
  # Report an estimate of how much the power-law model MSE exceeds
  # the power-law MSE:
  return(mean(fold.mses.gam) - mean(fold.mses.lm))
}

```

Running on the data,

```

> observed.cv <- do.nfold.cv(data=gmp.clean)
> observed.cv
[1] -0.01086810

```

So the additive model seems to generalize better than the power law. This does not tell us, however, whether it does so much better that we ought to give up on the power law.

Using `system.time`, the cross-validation takes about 0.5 seconds to run on my laptop; in fact just doing each fold takes about 0.1 second, mostly from fitting the additive model. Doing 1000 cross-validations should therefore take about $0.5 * 1000 = 500$ seconds, or a little over 8 minutes.

We also need to simulate from the null model, i.e., the power law. This model says nothing about how city size or any of the other explanatory variables are distributed, so we will follow the resampling-the-residuals idea from the lecture on bootstrapping. This takes no position on whether the noise is Gaussian, but it does assume that the noise is independent of the explanatory variables. One could do something more elaborate, e.g. a kernel density estimate of the distribution of residuals conditional on the input variables, but what we're doing here is sufficient.

```

simulate.power.law <- function(data=gmp.clean,model=lm.p.law) {
  n <- nrow(data)
  fitted.pcgmp <- predict(model,newdata=data)
  noise <- sample(residuals(model),size=n,replace=TRUE)
  # Or use the resample() command defined in the lecture notes
  data$pcgmp <- exp(fitted.pcgmp + noise)
  return(data)
}

```

Finally, we put this together:

```

> bootstrap.cv.mses <- replicate(1000,do.nfold.cv(data=simulate.power.law()))
> (1+sum(bootstrap.cv.mses < observed.cv))/(1+length(bootstrap.cv.mses))
[1] 0.000999001

```

This is 1/1001, which means that the amount by which the additive model beats the power law in real life is bigger than on *all* the 1000 simulations. In fact, the smallest value in `bootstrap.cv.mses` is -5.5×10^{-5} ; the observed value is almost 200 times as large. We can conclude with high confidence that the additive model does in fact generalize better than the power law.

(This indeed took eight and a half minutes to run, long enough to work on something small.)

6. (a) ANSWER: If we take the power law model at face value, then cities should become more productive simply by increasing their population; the economic changes would take care of themselves. Presumably there are limits to this (it is hard to see how it could work if neighboring cities simply dumped all their retirees, disabled people, homeless people, etc. into one city), but the model suggests that, within the ordinary demographic range for America, a city could indeed become more productive just by attracting more people.

On the other hand, the log-additive model implies that population as such does not matter, that what's important is the city's industrial mix, and that the correlation of city size with productivity is due to higher-productivity industries tending to be located in bigger cities. This suggests that cities should concentrate on attracting those industries (which might in fact lead to population growth). It does not say anything about how best to do that.³ Since the additive model seems to be superior to the power law model, I would *tentatively* say that changing the industrial mix is more important than changing population, which seems to have no effect for cities with the same industry shares.

(The question was limited to "American cities", because the data provides no information on cities elsewhere.)

- (b) ANSWER: The simplest way to improve the analysis would be to get the missing data values! We could also try incorporating interactions between variables (perhaps by a kernel regression model), or additional demographic variables (e.g., what fraction of the population is of working age?), or additional industrial variables (e.g., shares of other industries, more detailed industry categories).

More generally, it is hard to tell from static data like this whether increasing a city's population attracts more high-productivity industries, or population tends to accumulate around such industries. Carefully looking at changes over time could be helpful to figuring out which comes first. (Perhaps they feed back on each other in a virtuous cycle.) So would including additional variables. If we could incorporate other variables which might effect both population and

³It is also not clear whether *every* city could follow this strategy at once — it might be that there is only so much demand for financial or management services.

industry shares, we can try to control for them and so remove spurious associations.

A final modeling issue is that it seems very odd to treat cities as independent of each other, even conditional on the values of their explanatory variables. It would not be surprising if productivity levels in New York city were related to those in New Jersey, Connecticut, etc., but right now both our models ignore geography.

The ideal additional study would in fact be to do controlled experiments where we try out different interventions (increasing or decreasing population levels, varying the industrial mix) and see what happens. This is hard to arrange.

References:

- Luís M. A. Bettencourt, José Lobo, Dirk Helbing, Christian Kühnert and Geoffrey B. West, “Growth, Innovation, Scaling, and the Pace of Life in Cities”, *Proceedings of the National Academy of Sciences (USA)* **104** (2007): 7301–7306, doi:10.1073/pnas.0610172104.
- Cosma Rohilla Shalizi, “Scaling and Hierarchy in Urban Economies”, submitted (2011), <http://arxiv.org/abs/1102.4101>.