Homework 4 Solutions

Solution

23 March 2017

Source paper: Dani Rodrik, "The Real Exchange Rate and Economic Growth", *Brookings Papers on Economics Activity* (Fall 2008, pp. 365–412) https://www.brookings.edu/wp-content/uploads/2008/09/2008b_bpea_rodrik.pdf.

1 Linear models

This should, by now, be easy. We'll want to re-use the formula, so we keep it separate. There are no missing values in the dataset, so they don't need to be dealt with.

Table 1: Coefficients and standard errors from the regression with under-valuation index and log(GDP).

	lm1.bhat	2.5 %	97.5 %
underval	0.005	0.000	0.009
log(gdp)	0.006	0.005	0.008

Note: we don't need to add a new column to `uv` which contains log GDP, or store that as a separate vector, and in fact it makes things easier later if we do not. `lm` will handle the transformation for us, internally.

The coefficients and their confidence intervals are in Table 1. The coefficient of log(GDP) is positive, indicating that higher initial GDP predicts faster growth. This is the opposite of "catching up"; rather, it's "the rich get richer". On the other hand, the positive coefficient of under valuation index suggests that when under-valuation index is positive (i.e., the currency is under-valued), growth is faster on average.

To add the year and the country to the regression, we change the formula:

Table 2: Coefficients and standard errors for under-valuation index and log(GDP) from the regression with those variables, and fixed effects for year and country.

	lm2.bhat	2.5 %	97.5 %
underval	0.014	0.008	0.019
log(gdp)	0.029	0.023	0.035

The coefficients and their standard errors are in Table 2. The coefficients have changed, which must be due to correlations between these variables and the indicator variables for the years and countries. From the coefficients reported in Table 2, both coefficients are again positive and quite a bit larger. Therefore, similar arguments can be made as in for the regression without the year and country effects.

Treating `year` as a quantitative variable in a linear model would mean that every five-year increment of time would add or subtract the *same* amount to expected growth rates — growth would be either getting steadily faster or slower over time. Since this obviously isn't what happens, a hack is to treat `year` as categorical, estimating a separate additive "fixed effect" for year. (When we come to additive models, we'll see something less hackish.)

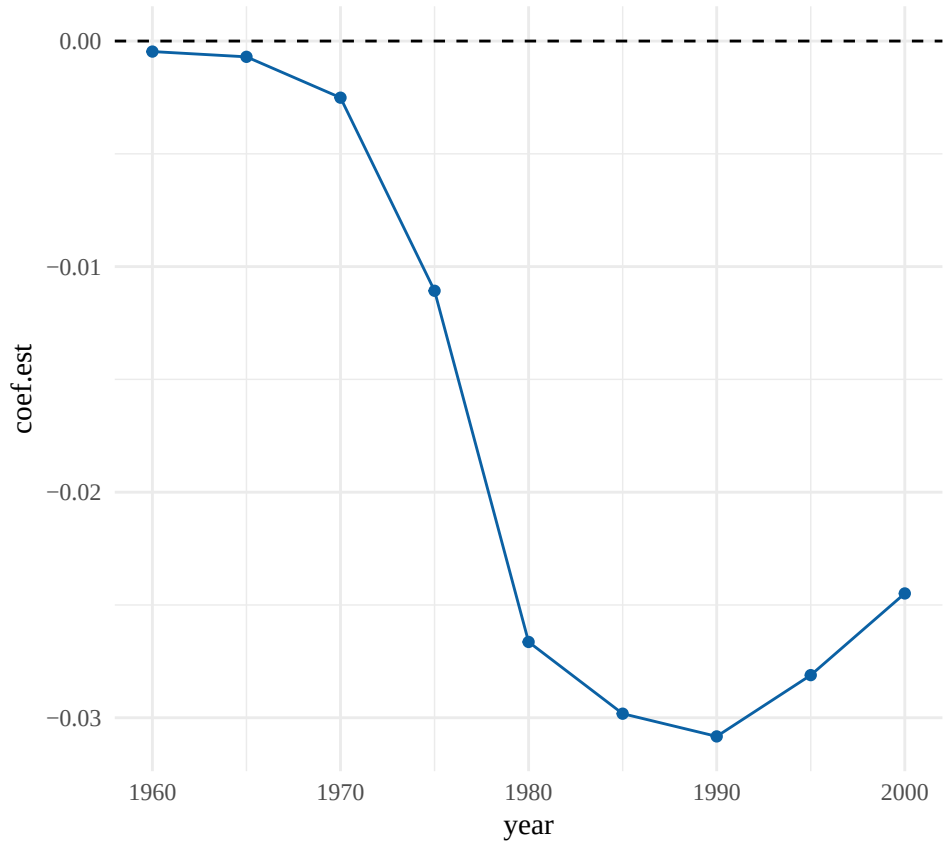

Figure 1: Plot of year coefficients from the regression model versus time

Figure 1 plots the coefficients of year versus time. The vertical distance from each point to the horizontal line at 0 is the difference in growth rate of a certain year compared to 1955, the first category, holding all other variables fixed. And the vertical distance between each pair of points is the differences in growth rate between those years. The graph shows that the jump between consecutive points is not constant, confirming again that modeling effect of each year separately is appropriate. It also shows that, at a global scale, growth slowed down drastically in the 1970s and never really recovered.

The R^2 and adjusted R^2s for two models are in Table 3. Model 2 has dramatically higher R^2, even with the conventional adjustment for extra parameters.

Table 3: R^2 and adjusted R^2 for model in problem 1 (with only undervalue index and log(GDP)) and for model in problem 2 (adding country and year fixed effects).

	R sq.	adj. R sq.
Model 1	0.049	0.047
Model 2	0.429	0.332

To use cross validation, I loaded the function `looCV.forNiceModels` from Chapter 4 lecture. You may then run it as you wish, without its their definition in any way. You could also use the `cv.lm` function from Chapter 3. To do leave-one-out cross-validation in that case, we set the number of folds to be the number of rows of the data frame (as explained in the notes). This does leave-one-out cross-validation for the two models specified by the two formulas, and returns the vector of the cross-validated MSEs. While the second model,

with fixed effects for country and year, seemed an order of magnitude better by R^2, it actually improves by only about 7.53% — better than nothing, but not dramatic.

As for why 5-fold is hard, we need at least one observation of each level of a qualitative variable to estimate its coefficient, and some countries have fewer than five observations. (Some have just two.) If we use five folds, some countries just won't appear in some training sets, and the model won't know what to do with them on the testing sets. With at least two observations on each country and each year, however, leave-one-out CV will be able to work. (Note: Prof. Rodrik's original data set contained a few countries with just one observation; these were removed for this assignment, with only minute changes to the estimates.)

Finally, for the bootstrap, it is *possible* that resampling points will not work for the same reasons that 5-fold CV doesn't work. For me, it worked every time I tried. The table below shows both of these versions. Resampling residuals doesn't do much to effect the CI. But resampling points results in wider CIs. The coefficient estimates are still significant, however (none of these CIs overlap with 0). The table below has the estimates along with the original CIs and both bootstrapped versions

		2.5 %	97.5 %	lower	upper	lower	upper
underval	0.014	0.008	0.019	0.008	0.018	0.007	0.021
log(gdp)	0.029	0.023	0.035	0.022	0.034	0.019	0.040

2 Kernel Smoothing

If necessary, install the `np` package first:

```
install.packages("np",dependencies=TRUE)
```

Then run `npreg` as instructed. Note that we don't have to tell `npreg` to treat `country` as a categorical variable (it does that automatically, because that column contains factors), and we don't even really need to treat `year` as qualitative, either, because the kernel regression won't impose an artificial linear trend over time. However, here it does little harm.

There are no coefficients for the kernel regression. We have coefficients in linear models, because they work by multiplying the covariates by constants, the coefficients, and adding those terms up. Kernel regressions just take weighted averages of the actual data points, with weights which shift according to where we're making predictions. The whole notion of "the coefficients" makes no sense for kernel regressions (or most other kinds of models).

Figure 2 plots the predicted values of the kernel regression against the predicted values of the linear model.

Figure 3 plots the residuals against predicted values for our kernel regression. Ideally, this should show a scatter around 0. The ideal regression function $\mu(x)$ is the conditional mean, $\mu(x) = \mathbb{E}[Y|X = x]$. The residuals of this ideal regression would be $Y - \mu(x)$, and their average value would be $\mathbb{E}[Y - \mu(X)|X = x] = \mathbb{E}[Y|X = x] - \mu(x) = 0$ everywhere. Instead, we see that the residuals tend to be positive when the predicted values $\hat{\mu}(x)$ are large, and negative when $\hat{\mu}(x)$ is small. This means that the model is systematically under-predicting at the high end and over-predicting at the low end. (This may be due to "boundary bias", whose nature and treatment will be covered in chapter 7.)

As described in the notes, `npreg` uses cross-validation to pick bandwidths, and stores the cross-validated MSE along with those bandwidths: 0.001.

This is about 5.887% better than the best of the two linear models — again, not a huge change, but real.

As also described in the notes, the (so to speak) "headline" MSE reported by `summary(kernel.model)` is merely the in-sample MSE. Likewise, the R^2, here 0.99963 (!), is in-sample value.

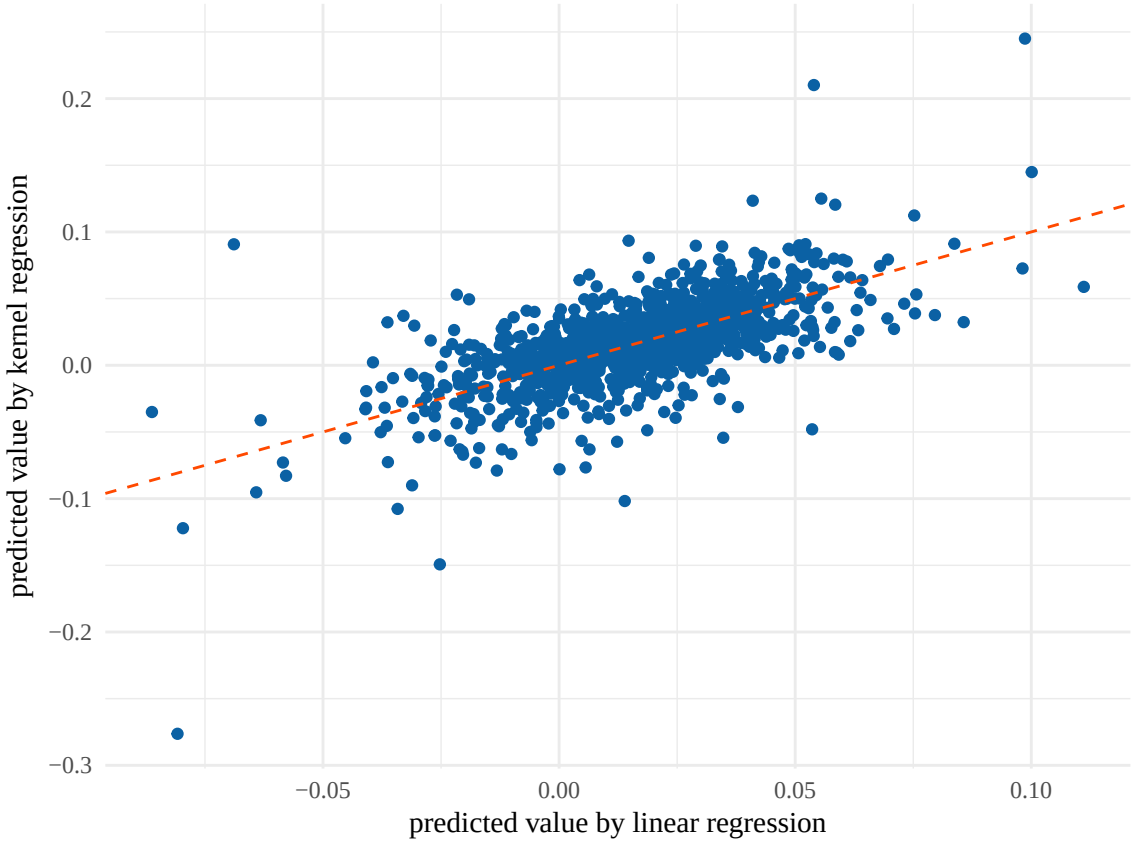

Figure 2: Predicted values of kernel regression against predicted values of the linear model; the dashed red line shows equality.

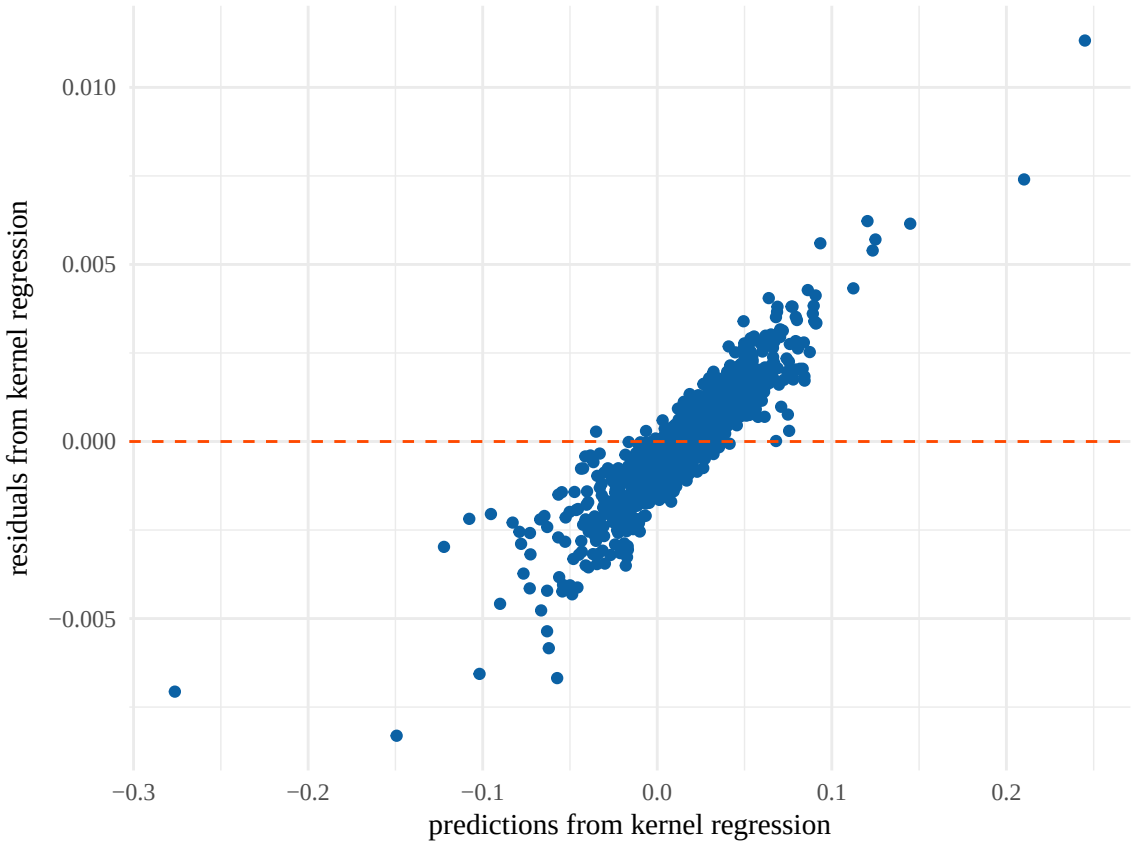

Figure 3: Residuals of the kernel regression versus predicted values.