Thoughts About New lmGC()

Homer White

10/03/2014

New Version of GC Linear Model?

I'm working on new versions of `lmGC()`. First I'll source them into this document, along with an example data frame that is not yet in `tigerstats`:

```
require(tigerstats)
require(ggplot2)
source("better_lmGC.R")
source("better_predict_lmGC.R")
source("print_better_lmGC.R")
source("diag_lmGC.R")
load("henderson.rda")
```

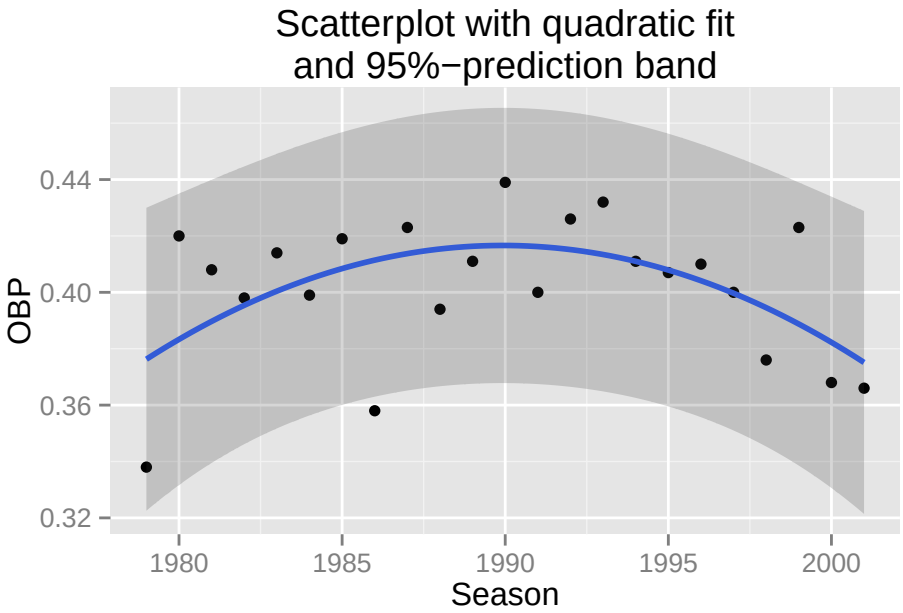
Basic Usage

The new `lmGC()` family of functions would behave a bit differently. There would be a new argument `degree` that would allow a polynomial fit. Also, when you graph the model, we'll use `ggplot2` graphics and we'll include a 95%-prediction ribbon.

The following is a model for Ricky Henderson's season-by-season OBP (on-base percentage):

```
mod <- lmGC2(OBP~Season,data=henderson,degree=2,graph=TRUE)
mod
```

```
##
##  Quadratic Regression
##
## Equation of Fitted Parabola:
##
##  OBP=-1339 + 1.346*Season + -0.0003382*Season^2
##
## Residual Standard Error: s   = 0.0223
## R^2 (unadjusted):        R^2 = 0.289
```



The idea is to tell students about the residual standard error s, and point out that when you make a predictions of y values based on a particular known x then the predictions are liable to be off by s or so, and that a rough 68-95 Rule applies:

- about 68% of the time, the real Y will be within an s of your prediction, and
- about 95% of the time it will be within $2s$ of your prediction

But this is only rough. The graph is trying to show you the more carefully-computed 95% -prediction intervals, which together form the ribbon you see around the fitted curve.

Numerical Prediction

To get predictions numerically, you still use the generic `predict()` function. Here's what we predict Ricky Henderson's OBP would have been if he had stayed on for the 2002 season:

```
predict(mod,x=2002)
```

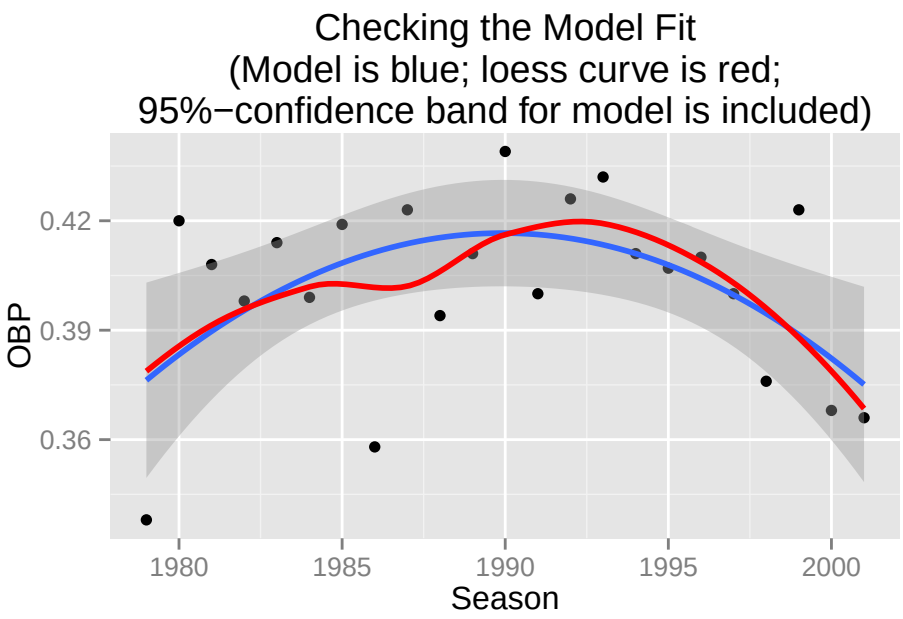
```
## Predict OBP is about 0.3673,
## give or take 0.02706 or so for chance variation.
##
## 95%-prediction interval:
##           lower.bound         upper.bound
##           0.310853            0.423728
```

Note that a prediction interval is now provided. The default level is 95%, but you can change that with the `level` argument. If you use this with students, point out that the prediction standard error that is provided as the "give-or-take" figure is almost but not quite that same as s. (It will vary a bit depending on the value of x.) Also, the prediction intervals are not quite $\hat{y} \pm 2s$, they are t-intervals instead. We would just say to the students that the routine routine is just trying to be a bit less "rough" that than the 68-95 rule was.

A Simple Model-Selection Criterion

You have an option to check to whether you have a "big-enough" degree fit.

```
lmGC2(OBP~Season,data=henderson,degree=2,check=TRUE)
```



I'm showing just the graph above. For students the console output would appear as well.

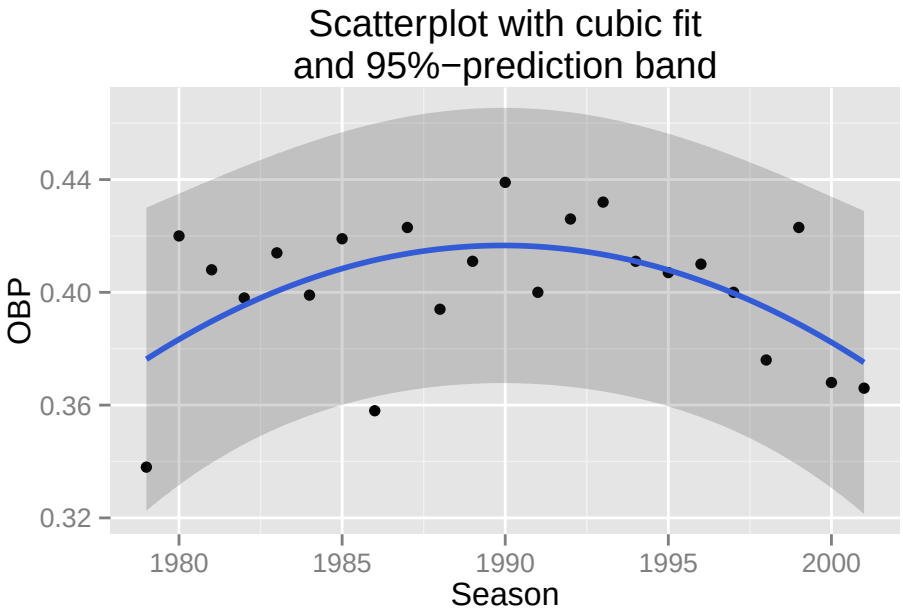
The ribbon switches: now it is a 95%-confidence band. For each x, the vertical segment of the ribbon over x is the 95%-confidence interval for the expected value of Y, given that $X = x$. If some degree-2 curve plus bell-shaped chance variation was the "real" way the data was produced, then you can be pretty confident that it would lie mostly. Now look at the loess curve (or gam curve if the scatter plot has more than 1000 points). This is your estimate of the curve that generates the data, based "only on the data" and not on the data plus some idea about what form the curve has (e.g., polynomial of a particular degree). If the loess curve does not stray too far outside the band, then there is probably not much benefit to using a higher-degree curve as a fit.

One way to show that to students is to try some higher-than-needed degree:

```
lmGC2(OBP~Season,data=henderson,degree=3,graph=TRUE)
```

```
##
##  Cubic Regression
##
## Equation of Fitted Cubic:
##
##  OBP = -1339 + 1.346*Season + -0.0003382*Season^2 + 469.9*Season^3
##
## Residual Standard Error: s   = 0.0223
## R^2 (unadjusted):        R^2 = 0.289

## Warning: prediction from a rank-deficient fit may be misleading
```



Note that the R^2 is not much higher (and the s is actually worse than at degree two).

But there is a complication, here. Note thw warning about rank-deficiency, the result of a a call of R's `predict.lm()` from the `ggplot()` function.

I've also set it up so that the same warning is issued when you use `predict()`:

```
mod2 <- lmGC2(OBP~Season,data=henderson,degree=3)
predict(mod2,x=2002)
```

```
## Predict OBP is about 0.3673,
## give or take 0.02706 or so for chance variation.

## Warning: prediction from a rank-deficient fit may be misleading

## 95%-prediction interval:
##           lower.bound         upper.bound
##           0.310853            0.423728
```

Now there are many reasons for rank-deficiency, but only two are likely, I think, to occur for students:

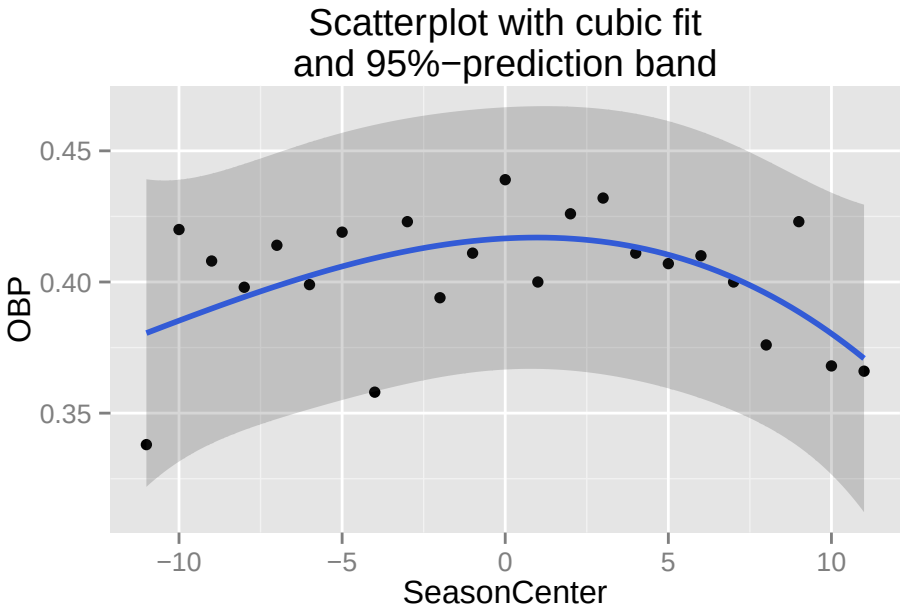
- the degree you want to fit is too high, given the number and/or placement of the points on the scatterplot, bascially it's a stupid or overly-complex model;
- the x-values are so large (or so small) that when they are raised to those big powers the numerical routines deliver matrices that don't have full rank.

There is no simple rule to give to student,s or to anyone, that lets you know which of the two is the culprit. I think that if students are exploring responsibly, then they are far more likely to run into the latter cause rather than the former. Hence we might have to teach them that when they get a warning they should "center" the data so as to avoid computations with very large numbers. (Very small numbers would require re-scaling the data, maybe by taking z-scores).

For example:

```
SeasonCenter <- with(henderson,Season-1990)
lmGC2(OBP~SeasonCenter,data=henderson,degree=3,graph=TRUE)
```

```
##
##  Cubic Regression
##
## Equation of Fitted Cubic:
##
##  OBP = 0.4166 + 0.0006762*SeasonCenter + -0.0003382*SeasonCenter^2 + -9.223e-06*SeasonCenter^3
##
## Residual Standard Error: s   = 0.0228
## R^2 (unadjusted):        R^2 = 0.2962
```

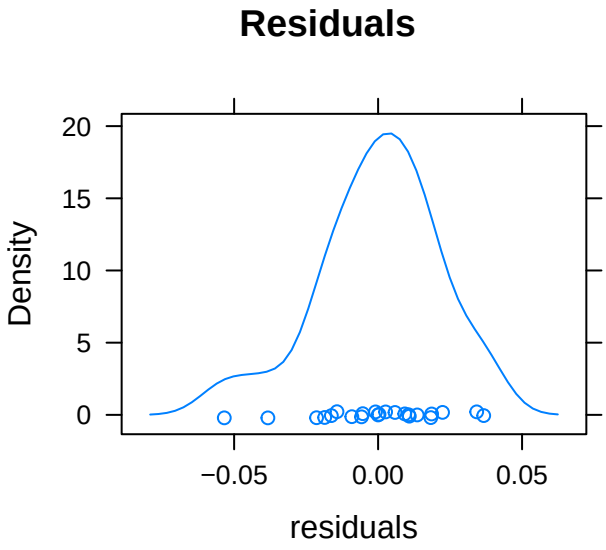


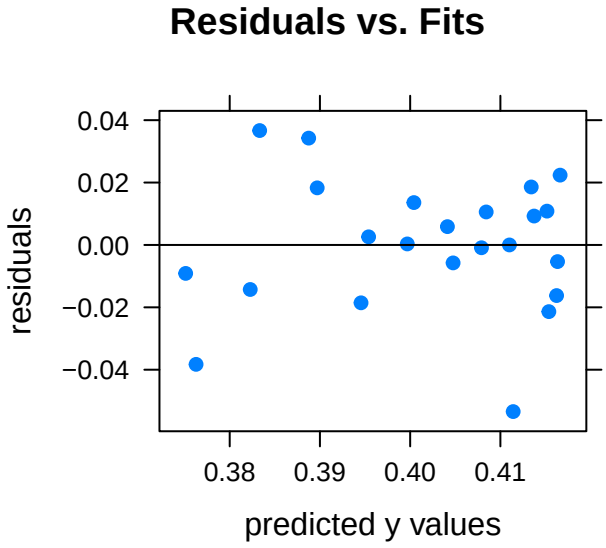
No more warning. Now students can tell, from the R^2 and s, that the data don't support a move from degree 2 to degree 3.

Diagnostics

Under the new set-up the argument list for `lmGC()` is getting pretty long. I'm thinking I would like to move the diagnostics out (they are the least likely topic to cover in MAT 111) and handle them as R does, through the generic `plot()` function:

```
plot(mod)
```





Feedback?

If you have ideas about how all of this should work, let me know. Clearly there is a benefit in being able to explore and to fit various models, but there is a big cost in that students will have to be taught to deal with numerical issues (e.g., centering and possibly even re-scaling) so as to separate warnings based on numerical issues from warnings based on stupidity of the chosen model.

Should there rather be a separate, optional function (maybe called `polyFit()`) to handle fits of degree greater than one?

Or should I build re-scaling into `lmGC()`, so that it occurs automatically when numerical issues are going to arise? I might be able to do so in a way that "hides" the re-scaling from the students.

Also, maybe advise me on timing issues. I would like to include the new functions in the next CRAN release, which should occur later this Fall, but of course we do not need to install it on the Server this semester if we think it would be confusing to students. On the other hand it might be helpful for them to take a look near the end of class, as a way of thinking about something like "confidence" intervals in a regression setting, and of course it is empowering to be able to fit a modest curve to data that does look, after all, curvilinear.