piecewise-regression (aka segmented regression) in Python

Charlie Pilgrim

1 Centre for Doctoral Training in Mathematics for Real-World Systems, University of Warwick, Coventry, UK 2 The Alan Turing Institute, London, UK

Summary

Piecewise regression (also known as segmented regression, broken-line regression, or breakpoint analysis) fits a linear regression model to data that includes one or more breakpoints where the gradient changes. The piecewise-regression Python package uses the approach described by Muggeo (Muggeo, 2003), where the breakpoint positions and the straight line models are simultaneously fit using an iterative method. This easy-to-use package includes an automatic comprehensive statistical analysis that gives confidence intervals for all model variables and hypothesis testing for the existence of breakpoints.

Statement of Need

A common problem in many fields is to fit a continuous straight line model to data that includes some change(s) in gradient known as breakpoint(s). Examples include investigating medical interventions (Wagner et al., 2002), ecological thresholds (Toms & Lesperance, 2003), and geological phase transitions (Ryan et al., 2002). Fitting such models involves the global problem of finding estimates for the breakpoint positions and the local problem of fitting line segments given breakpoints. Possible approaches involve using linear regression to fit line segments together with a global optimisation algorithm to find breakpoints—for example, an evolutionary algorithm as in the pwlf python package (Jekel & Venter, 2019). Or one could take a non-linear least-squares approach using scipy (Virtanen et al., 2020) or the lmfit python package (Newville et al., 2016). Muggeo (Muggeo, 2003) derived an alternative method whereby the breakpoint positions and the line segment models are fitted simultaneously using an iterative method, which is computationally efficient and allows for robust statistical analysis. Many R packages implement this method, including the segmented R package written by Muggeo himself (Muggeo & others, 2008). However, before the piecewise-regression package, there were not comparable resources in Python.

Example

An example plot is shown in Figure 1. Data was generated with 3 breakpoints and some noise, and a model was then fit to that data. The plot shows the maximum likelihood estimators for the straight line segments and breakpoint positions. The package automatically carries out a Davies hypothesis test (Davies, 1987) for the existence of at least 1 breakpoint, in this example finding strong evidence for breakpoints with \( p < 0.001 \).
How It Works

We follow here the derivation by Muggeo (Muggeo, 2003). The general form of the model with one breakpoint is

\[ y = \alpha x + c + \beta (x - \psi)H(x - \psi) + \zeta, \]  

where given some data, \( x, y \), we are trying to estimate the gradient of the first segment, \( \alpha \), the intercept of the first segment, \( c \), the change in gradient from the first to second segments, \( \beta \), and the breakpoint position, \( \psi \). \( H \) is the Heaviside step function and \( \zeta \) is a noise term. This cannot be solved directly through linear regression as the relationship is non-linear. We can take a linear approximation by a Taylor expansion around some initial guess for the breakpoint, \( \psi^{(0)} \),

\[ y \approx \alpha x + c + \beta (x - \psi^{(0)})H(x - \psi^{(0)}) - \beta (\psi - \psi^{(0)})H(x - \psi^{(0)}) + \zeta. \]  

This is now a linear relationship and we can find a new breakpoint estimate, \( \psi^{(1)} \), through ordinary linear regression using the statsmodels python package (Seabold & Perktold, 2010). We iterate in this way until the breakpoint estimate converges, at which point we stop the algorithm. If considering multiple breakpoints, the same approach is followed using a multivariate Taylor expansion around an initial guess for each of the breakpoints.
Muggeo’s iterative algorithm is not guaranteed to converge on a globally optimal solution. Instead, it can converge to a local optimum or diverge. To address this limitation, we also implement bootstrap restarting (Wood, 2001), again following Muggeo’s approach (Muggeo & others, 2008). The bootstrap restarting algorithm generates a non-parametric bootstrap of the data through resampling, which is then used to find new breakpoint values that may find a better global solution. This is repeated several times to escape local optima.

Model Selection

The standard algorithm finds a good fit with a given number of breakpoints. In some instances we might not know how many breakpoints to expect in the data. We provide a tool to compare models with different numbers of breakpoints based on minimising the Bayesian Information Criterion (Wit et al., 2012), which takes into account the value of the likelihood function while including a penalty for the number of model parameters, to avoid overfitting. When applied to the example in Figure 1, a model with 3 breakpoints is the preferred choice.

Features

The package includes the following features:

- Standard fit using the iterative method described by Muggeo.
- Bootstrap restarting to escape local optima.
- Bootstrap restarting with randomised initial breakpoint guesses.
- Calculation of standard errors and confidence intervals.
- Davies hypothesis test for the existence of a breakpoint.
- Customisable plots of fits.
- Customisable plots of algorithm iterations.
- Printable summary.
- Summary data output.
- Comprehensive tests.
- Model comparison with an unknown number of breakpoints, with the best fit based on the Bayesian information criterion.

The package can be downloaded through the Python Package Index. The full code is publicly available on github. Documentation, including an API reference, can be found at Read The Docs.

Acknowledgements

I acknowledge support from Thomas Hills. The work was funded by the EPSRC grant for the Mathematics for Real-World Systems CDT at Warwick (grant number EP/L015374/1).

References

Davies, R. B. (1987). Hypothesis testing when a nuisance parameter is present only under the alternative. Biometrika, 74(1), 33–43.


