The `lua-regression` package

George Allison*

v1.0.1
April 11, 2025

Abstract

The `lua-regression` package is a LuaLaTeX package that provides a simple interface for performing polynomial regression on data sets. It allows users to specify the order of the polynomial regression, the columns of the data set to use, and whether to plot the results. The package also includes options for calculating and plotting the confidence intervals of the data.

Keywords: LuaLaTeX, regression, plotting, data analysis

Contents

*mailto:GHAllison1@sheffield.ac.uk

1 What is `lua-regression`?

The `lua-regression` package is a LuaLaTeX package that provides a simple interface for performing polynomial regression on data sets within LaTeX. For example:

```
\luaregression[plot=true, order=2, xcol=1, ycol=2]{data.csv}
```

The above code will perform a polynomial regression of order 2 on the data in the file `data.csv`, using the first column as the x-values and the second column as the y-values. The plot result will only work in a tikzpicture environment.

1.1 About

The main functions of the `lua-regression` package are written purely in Lua and integrated into LaTeX via LuaLaTeX. This code was written to provide a LaTeX consistent interface for performing polynomial regression on data sets, without the need for external software or libraries. Additionally, keeping styling consistent too.
The package uses the Lua programming language to perform the regression calculations, and it can be easily integrated into existing LaTeX documents using the LuaLaTeX engine. Currently, if you wish to perform a more than a basic linear regression on a data set, you must use an external program to perform the regression and then import the results or pgf file into LaTeX or run a converter script like `tikzplotlib` or `matlab2tikz`. This requires extra steps and can be unnecessarily complicated to maintain styling. Even the native linear regression available in pgfplots can feel to verbose.
The `lua-regression` package aims to simplify this process by providing a simple interface, or single command, for performing polynomial regression directly within LaTeX. The target audience for this package is primarily, students, researchers, and academics who are already working in LaTeX and need to perform polynomial regression on data sets as part of their work. The package is designed to be easy to use and flexible, allowing users to specify the order of the polynomial regression, the columns of the data set to use, and whether to plot the results. The package also includes options for confidence intervals, making it a powerful tool for data analysis and visualization that creates plots similar to those produced by the Python library `Seaborn`.
Using Lua allows for a clearer and more efficient implementation of the regression calculations, as well as better integration with LaTeX thanks to LuaLaTeX. It further benefits from not requiring any external dependencies outside of LaTeX, or the need to use `--shell-escape` to run.

1.2 Features

Currently, the `lua-regression` package supports the following features:

- Polynomial regression of any order.
- Plotting of the regression results using PGFplots.
- Confidence intervals and error bands using the bootstrap method.
- Simple interface for specifying data sets and options.
- No external dependencies or `shell-escape` required.
- Support for CSV format data files.

- Perform R^2 tests on the data.
- Support for significant figures.
- Add and remove equation and R^2 from the legend.
- Outputs equations and R^2 values to LaTeX commands, so they can be called in the document.

1.3 Acknowledgements

Steve Smith., for introducing me to LaTeX.
Rob S., for constant encouragement and moral support.
Max K., for providing feedback on the package and its features.

2 Installation

2.1 Requirements

The `lua-regression` package requires compilation with LuaLaTeX. It has been tested on Lua 5.2 and higher. Further some additional packages are required:

- `ifthen`
- `luacode`
- `tikz`
- `pfgkeys`
- `pgfplots`

The packages `pgfplots` and `tikz` are not strictly required for running the package. However, they are needed for drawing the generated equations or confidence intervals on the plot.

2.2 Install `lua-regression`

The package manager for your local TeX distribution should install the package fine. However, the package can also be downloaded independently, from the central or main repository, and placed in your local texmf directory.
Once you have a copy of `lua-regression` installed, include the following in your preamble:

```
\usepackage{lua-regression}
```

3 Todo

There are probably bugs and use cases that I have not thought of. This code was originally written for my own use, and I have not tested it on all possible data sets. Thus, it only includes the features I needed at the time of writing. Future enhancements to `lua-regression` may include:

- Support for other regression types (e.g., exponential, etc.).
- Improved error handling and debugging options.
- More advanced plotting options and customization.
- Support for other data formats (e.g., JSON, XML, etc.).

- Robust regression methods.
- Support for plotting multiple regression lines with one command.
- Restructuring the code to be more modular and easier to maintain.
- Improved performance.
- Expand statistical visualisations if not already available in pgfplots, with Seaborn as a baseline.
- Improve bootstrap method/offer alternative bootstrap methods, like Bias-Corrected and Accelerated (BCa) or Studentised Bootstrap-t.

4 Usage

4.1 Calling the Package

INFO It should be noted that Lua indexes at 1 and therefore when specifying x and y columns you should treat the first column as index 1.

INFO Current `lua-regression` only takes data in long format.

The `lua-regression` package is called using the following command:

```
\luaregression[options]{data.csv}
```

The options for `lua-regression` are seen in table 1.
Additionally, specific values from the package can be called in the document using the commands found in table 2.
These can be called in the document at any point after the `lua-regression` command.

5 Package Performance

The `lua-regression` package has not been extensively tested on large data sets and no promises of performance can be made. However, Lua is a fast and generally efficient language that should perform well on most data sets with minimal impact to compilation speed. There are probably improvements that can be made.
I have run some tests below on standard data to get a gauge of general performance. Tests were run on the following hardware:

- R9 5950X, 32GB 3200 MHz
- TeXLive 2025
- Python 3.11

Option	Description	Type	Default
xcol	The column index for the x-values	integer	1
ycol	The column index for the y-values	integer	2
ci	Whether to include confidence intervals	boolean	false
z-threshold	The Z-score threshold for confidence intervals	number	null
sig-figures	The number of significant figures to display	integer	4
order	The order of the polynomial regression	integer	1
plot	Whether to plot the results	boolean	false
pgf-options	Additional PGF options for plotting	string	mark=none,smooth
eq	Whether to show the equation in the plot legend	boolean	false
r2	Whether to show the R^2 value in the plot legend	boolean	false
debug	Whether to enable debug mode	boolean	false
bootstrap	The number of bootstrap samples for confidence intervals	integer	1000
cicolor	The color for the confidence interval fill	string	blue
cifillopacity	The opacity for the confidence interval fill	number	0.2

Table 1: Options for the `lua-regression` package.

\polyR	The R^2 value of the regression.
\polyeq	The polynomial equation of the regression in a format pgfplots can interpret.
\printeq	The polynomial equation of the regression in a visually nice format.
\qlwr	The points for the lower confidence interval.
\qupr	The points for the upper confidence interval.

Table 2: Available commands from `lua-regression` in LaTeX document

5.1 Calculation Performance

For the following test, no plotting was performed to measure just the speed of calculation independently of PGFplots plotting speed. As can be seen `lua-regression` is highly performant. Tests below were performed with plotting on using the following code:

```
\luaregression[order=2, xcol=1, ycol=2]{data.csv}
```

5.1.1 Performance on 80 row data set

A test of the package took approximately **0.0025** seconds to run.

5.1.2 Performance on 400 row data set

A test of the package took approximately **0.00551** seconds to run.

5.2 Plotting Performance

Plotting performance is noticeably slow as data set size increases however this is more related to PGFplots than anything in `lua-regression` as this behaviour is present without the package. Tests below were performed with plotting on using the following code:

```
\luaregression[plot=true, order=2, xcol=1, ycol=2]{data.csv}
```

5.2.1 Performance on 80 row data set

For example a test took approximately **0.786** seconds to plot.
The comparative Python code took approximately **0.533** seconds to plot.

5.2.2 Performance on 400 row data set

A test took approximately **1.99** seconds to plot.
The comparative Python code took approximately **0.265** seconds to plot.

6 How `lua-regression` works

As previously mentioned `lua-regression` attempts not to have external dependencies for main functionality (aside from reasonable standard LaTeX packages see section 2.1). Therefore, the main brains for the code is Lua based.

6.1 Polyfitting

The `lua-regression` package uses the least squares method to perform polynomial regression on the data set. The least squares method is a mathematical optimization technique that minimizes the sum of the squares of the differences between the observed values and the values predicted by the model. The polynomial regression model can be represented as:

$$y = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \tag{1}$$

Where:

- y is the dependent variable.
- x is the independent variable.
- a_0 is the intercept.
- a_1, a_2, ..., a_n are the coefficients of the polynomial.
- n is the order of the polynomial.

As mentioned coefficients are calculated using the least squares method. The coefficients are calculated using the following formula:

$$a = (X^T X)^{-1} X^T y \tag{2}$$

Where:

- a is the vector of coefficients.
- X is the matrix of independent variables.
- y is the vector of dependent variables.
- T is the transpose operator.

The matrix X is constructed by adding a column of ones to the data set, which represents the intercept term. The matrix X is then raised to the power of the order of the polynomial, and the coefficients are calculated using the above formula. The coefficients are then used to calculate the predicted values of the dependent variable, which can be plotted against the observed values to visualize the fit of the model.

6.2 Confidence Intervals

The `lua-regression` package uses the normal approximation bootstrap method to calculate confidence intervals for the polynomial regression. The bootstrap method is a resampling technique that involves repeatedly sampling from the data set with replacement to create multiple bootstrap samples. The polynomial regression is then performed on each bootstrap sample, and the coefficients are calculated for each sample. The confidence intervals are then calculated by taking the percentiles of the coefficients from the bootstrap samples. The confidence intervals can be represented as:

$$CI = \left[\hat{a}_i - z_{\alpha/2} \cdot SE(\hat{a}_i), \hat{a}_i + z_{\alpha/2} \cdot SE(\hat{a}_i)\right] \tag{3}$$

Where:

- CI is the confidence interval.
- $\hat{a}_i$ is the estimated coefficient for the i-th term of the polynomial.
- $z_{\alpha/2}$ is the critical value from the standard normal distribution for a given significance level α.
- $SE(\hat{a}_i)$ is the standard error of the estimated coefficient.

The standard error of the estimated coefficient is calculated using the following formula:

$$SE(\hat{a}_i) = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}\left(y_j - \hat{y}_j\right)^2} \tag{4}$$

Where:

- n is the number of observations in the data set.
- y_j is the observed value for the j-th observation.
- $\hat{y}_j$ is the predicted value for the j-th observation.

The confidence intervals are then plotted on the graph to show the range of values within which the true coefficients are likely to fall. The confidence intervals can be used to assess the uncertainty of the estimated coefficients and to determine whether the coefficients are statistically significant. The confidence intervals are plotted as shaded areas around the fitted polynomial regression line, and they provide a visual representation of the uncertainty in the model.

6.3 R^2 Calculation

The `lua-regression` package calculates the R^2 value for the polynomial regression using the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} \tag{5}$$

Where:

- R^2 is the coefficient of determination.
- y_i is the observed value for the i-th observation.
- $\hat{y}_i$ is the predicted value for the i-th observation.
- $\bar{y}$ is the mean of the observed values.
- n is the number of observations in the data set.

7 Example

The following example demonstrates how to use the `lua-regression` package to perform polynomial regressions on a data set and plot the results. The data set used in this example is a CSV file from the Seaborn-data Github repository, which contains information about the miles per gallon (MPG) of various cars.

```
\luaregression[xcol = 4, ycol = 5,
 ↪  order = 1]{example/mpg.csv}

The equation for the linear
 ↪  regression for the MPG data set
 ↪  is $\printeq$ and the $R^2$
 ↪  value is $\polyR$.
```

The equation for the linear regression for the MPG data set is $19.0782x + 984.5003$ and the R^2 value is 0.7474.

7.1 A linear regression of order 1

The following code performs a polynomial regression of order 1 on the MPG data set, using the first column as the x-values and the second column as the y-values. Seen in figure 1.

```
\begin{tikzpicture}
    \begin{axis}[
        height=6.45cm,
        width=\textwidth,
        domain=0:300,
        samples=1000,
        xmin=25,
        xmax=240,
        xlabel=horsepower,
        ytick={},
        xtick={},
        ymax=6000,
        ymin=1250,
        ylabel=weight,
        grid=both,
        legend columns = 2,
        legend style={cells={align=left},at={(0.45,-0.22)},anchor=north},
        legend cell align=left,
        major grid style={line width=.2pt,draw=gray!20},
        every axis/.append style={axis line style={gray!80, line
         ↪  width=0.75pt}, tick style={gray!95}}
    ]

    \addlegendimage{p4, mark=*, thick}
    \addlegendimage{p8, thick}

    \pgfplotstableread[col sep=comma]{example/mpg.csv}\datatable

    \addplot [p4,mark=*,fill opacity=0.75, draw opacity=0] table [only
     ↪  marks,col sep=comma,x=horsepower,y=weight]{\datatable};

    \luaregression[xcol = 4, ycol = 5, plot = true, eq = true, r2 = true,
     ↪  order = 1, ci = true]{example/mpg.csv}

    \end{axis}
\end{tikzpicture}
```

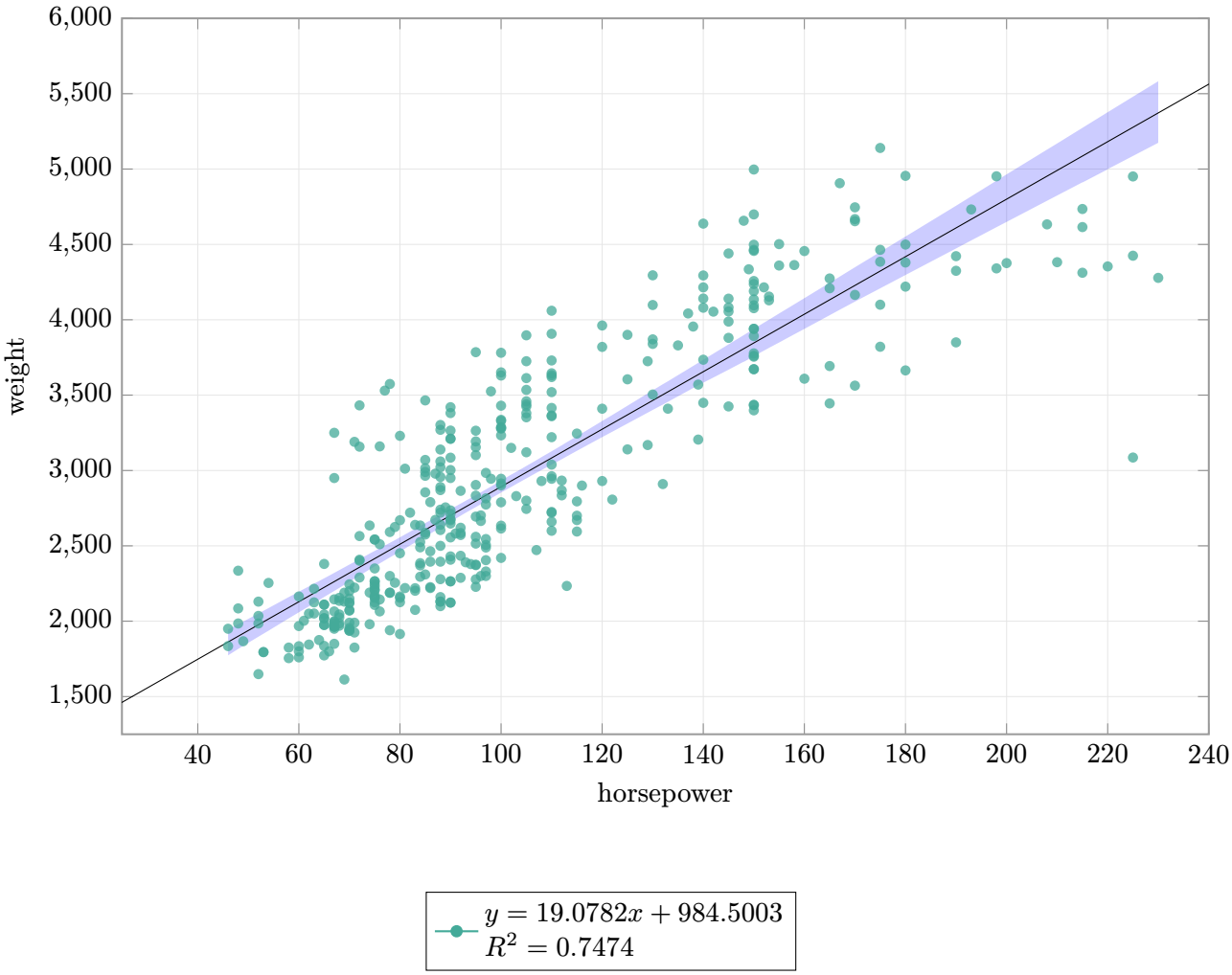

Figure 1: Polynomial regression of order 1 on the MPG data set. The plot shows the data points, the fitted polynomial regression line, and the confidence intervals.

7.2 A polynomial regression of order 2

The following example demonstrates how to use the `lua-regression` package to perform polynomial regression of order 2 on the same data set. Seen in figure 2.

```
\begin{tikzpicture}
    \begin{axis}[
        height=6.45cm,
        width=\textwidth,
        domain=0:300,
        samples=1000,
        xmin=25,
        xmax=240,
        xlabel=horsepower,
        ytick={},
        xtick={},
        ymax=6000,
        ymin=1250,
        ylabel=weight,
        grid=both,
        legend columns = 2,
        legend style={cells={align=left},at={(0.45,-0.22)},anchor=north},
        legend cell align=left,
        major grid style={line width=.2pt,draw=gray!20},
        every axis/.append style={axis line style={gray!80, line
        ↪  width=0.75pt}, tick style={gray!95}}
    ]

    \addlegendimage{p4, mark=*, thick}
    \addlegendimage{p8, thick}

    \pgfplotstableread[col sep=comma]{example/mpg.csv}\datatable

    \addplot [p4,mark=*,fill opacity=0.75, draw opacity=0] table [only
    ↪  marks,col sep=comma,x=horsepower,y=weight]{\datatable};

    \luaregression[xcol = 4, ycol = 5, plot = true, eq = true, r2 = true,
    ↪  order = 2, ci = true]{example/mpg.csv}

    \end{axis}
\end{tikzpicture}
```

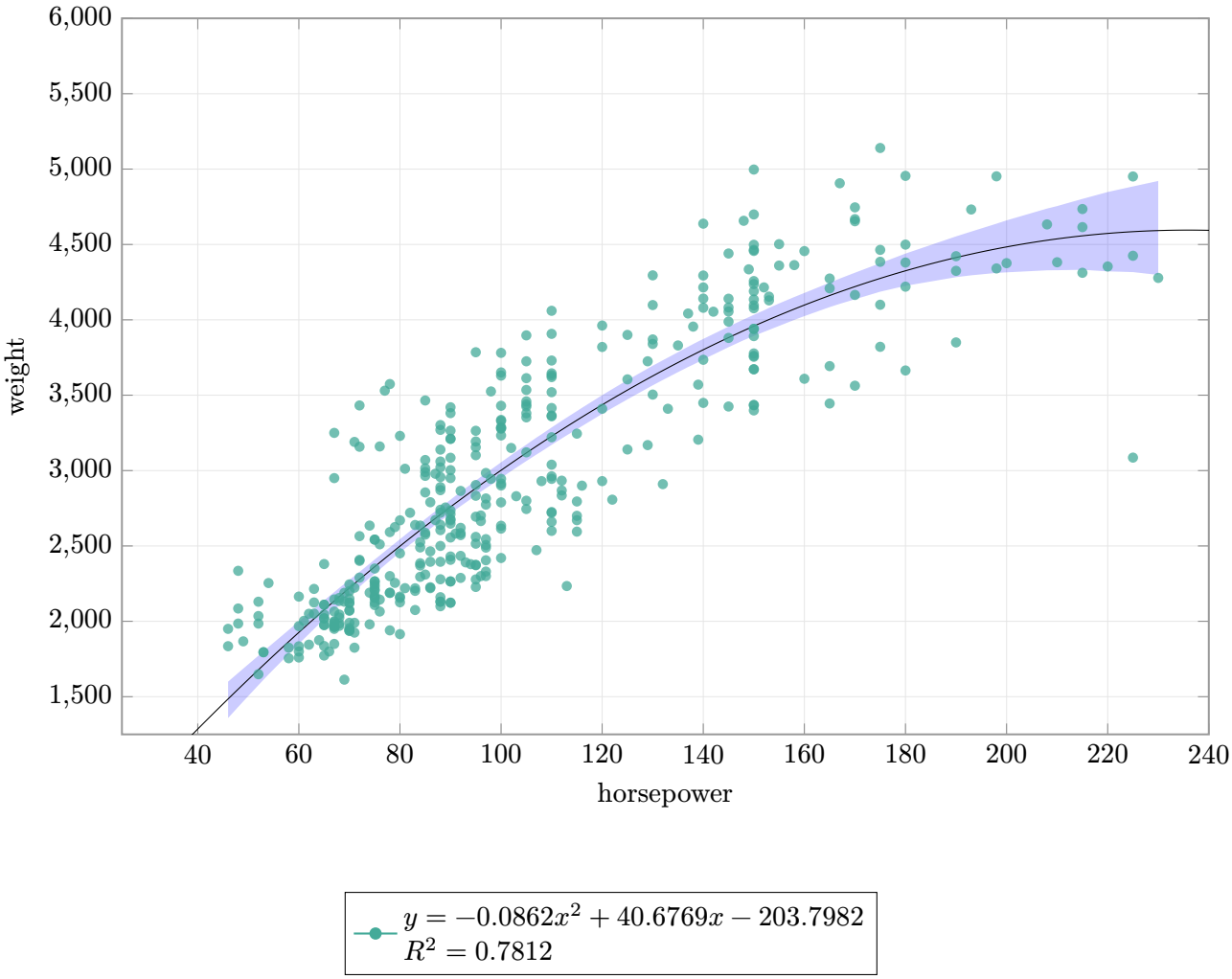

Figure 2: Polynomial regression of order 2 on the MPG data set. The plot shows the data points, the fitted polynomial regression line, and the confidence intervals.

8 Changelog

v1.0.1

- Added required package `ifthen`.

v1.0.0

- Initial release of the `lua-regression` package.
- Basic polynomial regression functionality.
- Plotting support using PGFPlots.
- Confidence intervals and error bands using the bootstrap method.
- Simple interface for specifying data sets and options.
- No external dependencies or `shell-escape` required.
- Support for CSV format data files.
- Perform R^2 tests on the data.
- Support for significant figures.
- Add and remove equation and R^2 from the legend.
- Outputs equations and R^2 values to LaTeX commands so they can be called in the document.

9 Code

```
\ProvidesPackage{lua-regression}[2025/04/06 1.0.1  Lua Regression Plotting project]

\RequirePackage{ifthen}
\ifluatex
  \RequirePackage{luacode}
\else
  {\PackageError{lua-regression}
  {Not running under LuaLaTeX}
  {This package requires LuaLaTeX. Try compiling this document with\MessageBreak 'lualatex' instead of 'latex'. This is a fatal error; I'm aborting now.}%
  }\stop
\fi

% Required packages
\RequirePackage{pgfkeys}
\RequirePackage{pgfplots}
\usepgfplotslibrary{fillbetween}

% Define the key-value options
% Define the key-value options
\pgfkeys{
    /luaregression/.is family, /luaregression,
    default/.style = {
        xcol=1,            % Default x-column index
        ycol=2,            % Default y-column index
        ci=false,  % Default: no error band
        z-threshold=null,  % Default Z-score threshold
        sig-figures=4,     % Default significant figures
        order=1,          % Default polynomial order
        plot=false,        % Default plotting behavior
        pgf-options={mark=none,smooth}, % Default PGF options
        eq=false, % Toggle for showing the equation
        r2=false,       % Toggle for showing R²
        debug=false,    % Debug toggle for csv
        bootstrap=1000, % Number of bootstrap samples for confidence intervals
        cicolor=blue,        % CI fill color
        cifillopacity=0.2,  % CI fill opacity
    },
    xcol/.estore in = \luaregressionxcol,
    ycol/.estore in = \luaregressionycol,
    ci/.estore in = \luaregressionci,
    z-threshold/.estore in = \luaregressionzthreshold,
    sig-figures/.estore in = \luaregressionsigfigures,
    order/.estore in = \luaregressionorder,
    plot/.estore in = \luaregressionplot,
    pgf-options/.estore in = \luaregressionpgfoptions,
```

```
    eq/.estore in = \luaregressionshowequation,
    r2/.estore in = \luaregressionshowrsquare,
    debug/.estore in = \luaregressiondebug,
    bootstrap/.estore in = \luaregressionbootstrapsamples,
    cicolor/.estore in = \luaregressioncicolor,
    cifillopacity/.estore in = \luaregressioncifillopacity,
}

% Define the macro
\newcommand{\luaregression}[2][]{%
    \pgfkeys{/luaregression, default, #1}% Parse the options
    \directlua{
        require("lua-regression")
        process_data_with_options(
            "#2",
            {
                ["xcol"] = tonumber("\luaregressionxcol"),
                ["ycol"] = tonumber("\luaregressionycol"),
                ["z_threshold"] = tonumber("\luaregressionzthreshold"),
                ["sig_figures"] = tonumber("\luaregressionsigfigures"),
                ["ci"] = ("\luaregressionci" == "true"),
                ["order"] = tonumber("\luaregressionorder"),
                ["debug"] = ("\luaregressiondebug" == "true"),
                ["bootstrap_samples"] =
                 ↪  tonumber("\luaregressionbootstrapsamples"),
            }
        )
    }%
    \ifthenelse{\equal{\luaregressionplot}{true}}{%
        \ifx\addplot\undefined
            \PackageError{lua-regression}{'plot=true' requires a tikzpicture
             ↪  environment and pgfplots}%
            {Use '\\begin{tikzpicture} ... \\end{tikzpicture}' with
             ↪  '\\usepackage{pgfplots}'.}%
        \fi
        \expandafter\addplot\expandafter[\luaregressionpgfoptions] {\polyeq};%
        % Construct the legend entry dynamically
        \begingroup
        \def\legendentry{}%
        \ifthenelse{\equal{\luaregressionshowequation}{true}}{%
            \edef\legendentry{$y = \printeq$}%
        }{}%
        \ifthenelse{\equal{\luaregressionshowrsquare}{true}}{%
            \ifx\legendentry\empty
                \edef\legendentry{$R^2 = \polyR$}%
            \else
                \edef\legendentry{\legendentry\\$R^2 = \polyR$}%
            \fi
        }{}%
```

```
        \ifx\legendentry\empty
        \else
            \expandafter\addlegendentry\expandafter{\legendentry}%
        \fi
        \endgroup
        % Plot confidence band if ci=true
        \ifthenelse{\equal{\luaregressionci}{true}}{%
        \addplot[name path=qlwrpath,draw=none] coordinates {\qlwr};
        \addplot[name path=quprpath,draw=none] coordinates {\qupr};
        \addplot[
            fill=\luaregressioncicolor,
            fill opacity=\luaregressioncifillopacity
        ] fill between[of=quprpath and qlwrpath];
        }{}%
    }{}%
}
```