The `lua-regression` package

George Allison[*]

v1.0.0
April 8, 2025

Abstract

The `lua-regression` package is a LuaLaTeXpackage 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 confidence intervals and error bands.

Keywords: LuaLaTeX, regression, plotting, data analysis

Contents

[*]`mailto:GHAllison1@sheffield.ac.uk`

1 What is `lua-regression`?

The `lua-regression` package is a LuaLaTeXpackage 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 LaTeXvia LuaLaTeX. This code was written to provide a LaTeXconsistent interface for performing polynomial regression on data sets, without the need for external software or libraries. The package uses the Lua programming language to perform the regression calculations, and it can be easily integrated into existing LaTeXdocuments using the LuaLaTeXengine. Currently, if you wish to perform a regression on a data set, you must use an external program to preform the regression and then import the results or pgf file into LaTeX. This requires extra steps and can be unnecessarily complicated to maintain styling.

The `lua-regression` package aims to simplify this process by providing a simple interface for performing polynomial regression directly within LaTeX. The target audience for this package is primarily, students, researchers, and academics who are already working in LaTeXand 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 and error bands, 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 LaTeXthanks to LuaLaTeX. It further benefits from not requiring any external dependencies, 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

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 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}
```

2.3 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.

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.

3 Usage

3.1 Calling the Package

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

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

The options for `lua-regression` are seen in table 1.

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.

Additionally, specific values from the package can be called in the document using the following commands:

```
\polyR - The R squared 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.
```

These can be called in the document at any point after the `lua-regression` command.

4 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.

4.1 Performance on 80 row data set

For example a test of the package on a data set of 80 rows and took approximately **0.786** seconds to run on a R9 5950X, using TexLive 2025.

The comparative Python code took approximately **0.533** seconds to run on the same data set.

4.2 Performance on 400 row data set

A test on data of 400 rows took approximately **1.99** seconds to run on a R9 5950X, using TexLive 2025.

The comparative Python code took approximately **0.265** seconds to run on the same data set.

5 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 for the seaborn-data GitHub repository, which contains information about the miles per gallon (MPG) of various cars.
Seaborn-data Github repository

```
\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.

5.1 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}
```

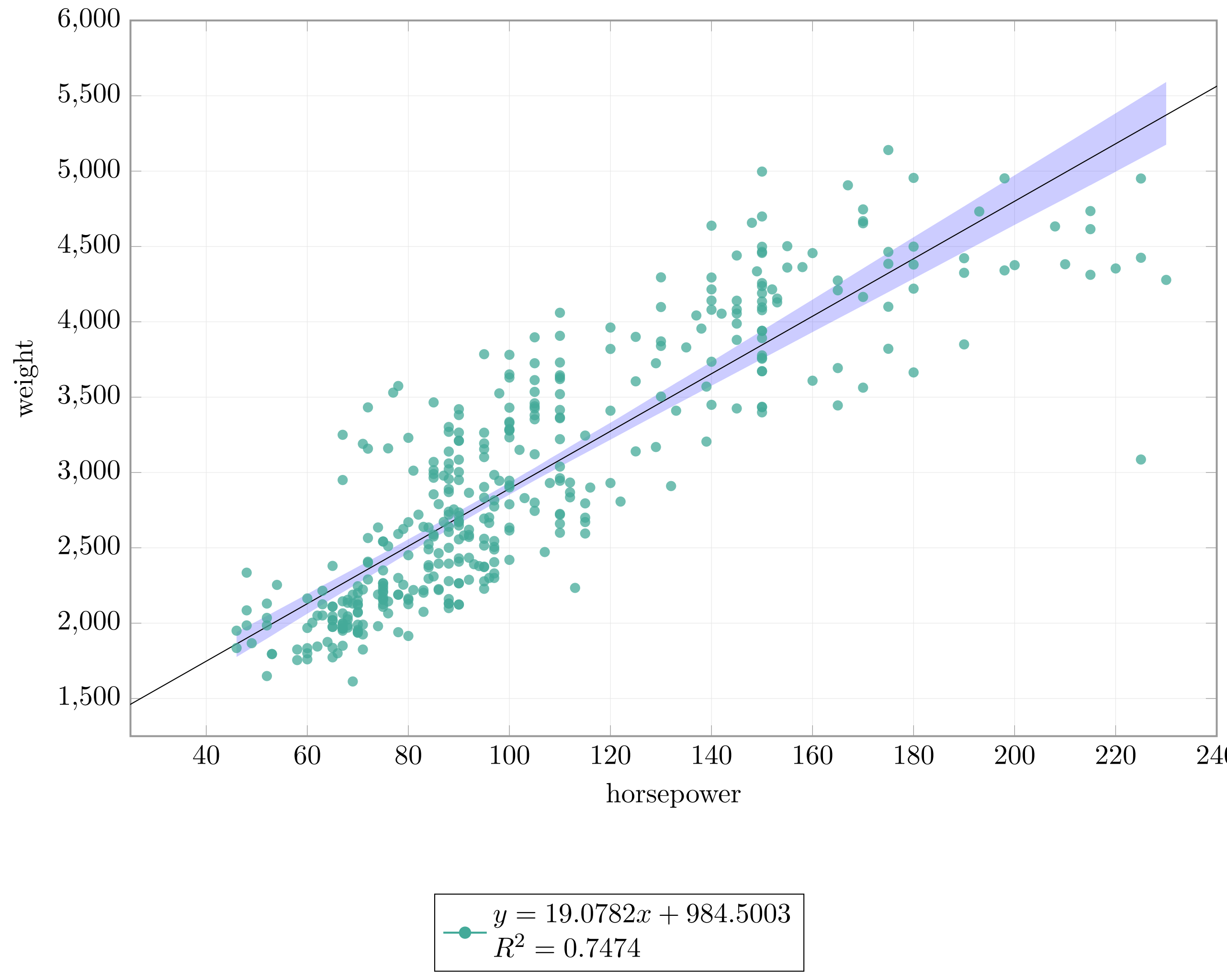
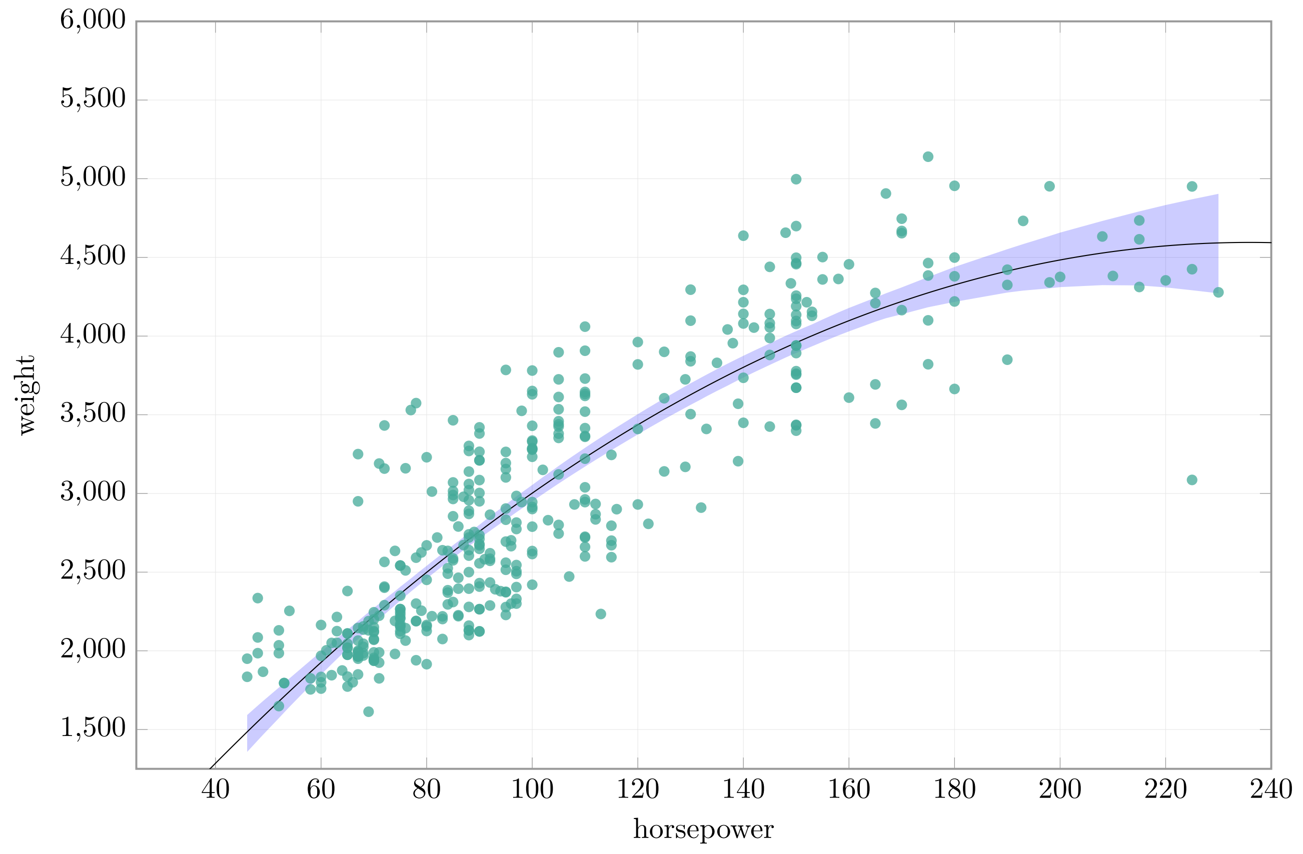


Figure 1: Polynomial regression of order 1 on the MPG data set. The plot shows the data points, the fitted polynomial regression line, and the confidence intervals.

5.2 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}
```



● $y = -0.0862x^2 + 40.6769x - 203.7982$
$R^2 = 0.7812$

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.

6 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.

7 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];
        }{}%
    }{}%
}
```