

# autoplotly - Automatic Generation of Interactive Visualizations for Popular Statistical Results

by Yuan Tang

**Abstract** The **autoplotly** package provides functionalities to automatically generate interactive visualizations for many popular statistical results supported by **ggfortify** package with **plotly** and **ggplot2** style. The generated visualizations can also be easily extended using **ggplot2** and **plotly** syntax while staying interactive.

## Introduction

With the help of base graphics, grid graphics, and **lattice** graphics (Sarkar, 2008), R users already have many plotting options to choose from. Each has their own unique customization and extensibility options. Nowadays, **ggplot2** has emerged as a popular choice for creating visualizations (Wickham, 2009) and provides a strong programming model based on a “grammar of graphics” which enables methodical production of virtually any kind of statistical chart. The **ggplot2** package provides a suit of succinct syntax and independent components and makes it possible to describe a wide range of graphics. It’s based on an object-oriented model that is modular and extensible, which becomes a widely used framework for producing statistical graphics in R.

The distinct syntax of **ggplot2** makes it a definite paradigm shift from base and **lattice** graphics and presents a somewhat steep learning curve for those used to existing R charting idioms. Many industry R users, especially the users that build web applications in R by leveraging **shiny** (Chang et al., 2017) package, may not be satisfied with static plots. Those web applications often involve user interactions so that users can dive into the plots, explore areas of interest, and select relevant data points for more details. **ggiraph** (Gohel, 2017) is an extension of **ggplot2** that provides building blocks for users to build interactive plots and when used within a shiny application, elements associated with an id can be selected and manipulated on client and server sides. There are also other packages such as **d3r** Bostock et al. (2017) and **plotly** Sievert et al. built on top of Javascript visualization frameworks that are totally isolated from **ggplot2** but become popular building blocks for creating interactive visualizations in R.

Often times users only want to quickly iterate the process of exploring data, building statistical models, and visualizing the model results, especially the models that focus on common tasks such as clustering and time series analysis. Some of these packages provide default base plot visualizations for the data and models they generate. However, they look out-of-fashion and these components require additional transformation and clean-up before using them in **ggplot2** and each of those transformation steps must be replicated by others when they wish to produce similar charts in their analyses. Creating a central repository for common/popular transformations and default plotting idioms would reduce the amount of effort needed by all to create compelling, consistent and informative charts. The **ggfortify** (Tang et al., 2016) package provides a unified **ggplot2** plotting interface to many statistics and machine-learning packages and functions in order to help these users achieve reproducibility goals with minimal effort. **ggfortify** package has a very easy-to-use and uniform programming interface that enables users to use one line of code to visualize statistical results of many popular R packages using **ggplot2** as building blocks. This helps statisticians, data scientists, and researchers avoid both repetitive work and the need to identify the correct **ggplot2** syntax to achieve what they need. Users are able to generate beautiful visualizations of their statistical results produced by popular packages with minimal effort.

The **autoplotly** (Tang) is an extension that builds on top of **ggplot2** and **ggfortify** to provide functionalities to automatically generate interactive visualizations for many popular statistical results supported by **ggfortify** package with **plotly** and **ggplot2** style. The generated visualizations can also be easily extended using **ggplot2** and **plotly** syntax while staying interactive.

## Background

### Software Architecture

This section may contain a figure such as Figure 1.

TODO: Hoverover metadata TODO: Zooming in details TODO: Extensibility with ggplot2 TODO:



Figure 1: The logo of R.

Extensibility with plotly TODO: Exportability with export(p, "inst/images/iris\_pca\_full.png")

```
library(splines)
subplot(
  autoplotly(ns(diamonds$price, df = 6)),
  autoplotly(ns(diamonds$price, df = 3)), nrow = 2, margin = 0.01)
```

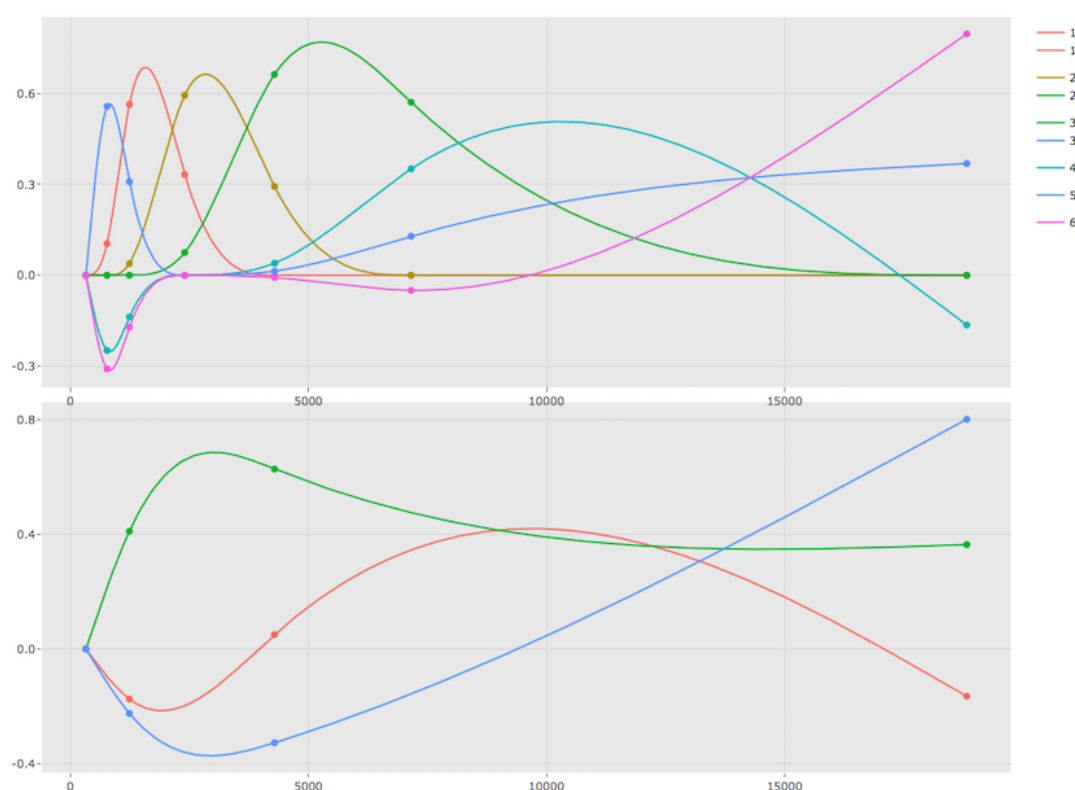


Figure 2: Multiple splines visualizations with different degree of freedom.

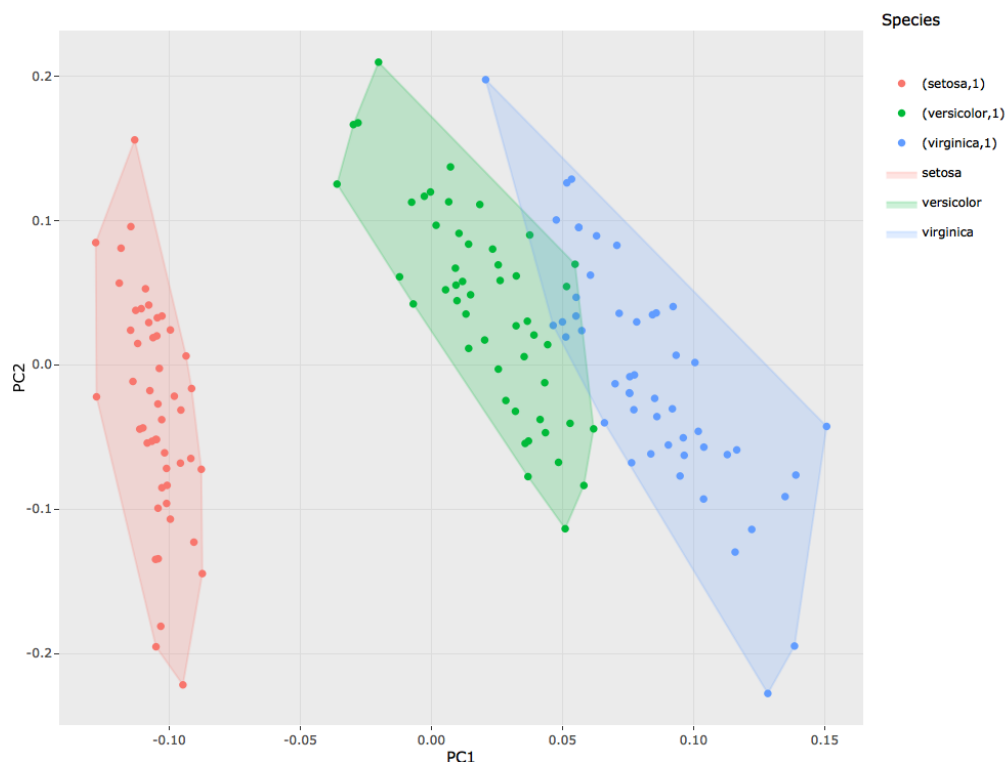
## Illustrations

There will likely be several sections, perhaps including code snippets, such as:

```
autoplotly(prcomp(iris[c(1, 2, 3, 4)]), data = iris, frame = TRUE, colour = 'Species')
```

Forecasting packages such as **forecast** (Hyndman, 2015), **change point** (Killick et al., 2016), **struc-change** (Zeileis et al., 2002), and **d1m** (Petrus, 2010), are popular choices for statisticians and researchers. Interactive visualizations of predictions and statistical results from those packages can be generated automatically using the functions provided by **autoplotly** with the help of **ggfortify**.

The **autoplotly** function automatically plots the change points with optimal positioning for the AirPassengers data set found in the **change point** package using the `cpt.meanvar` function, shown in Figure 4.



**Figure 3:** PCA with colors and boundary for each class.

```
library(changepoint)
autoplotly(cpt.meanvar(AirPassengers))
```

The **autoplotly** function automatically plots the original and smoothed line from Kalman filter function in **dlm** package as shown in Figure 5.

```
library(dlm)
form <- function(theta){
  dlmModPoly(order = 1, dV = exp(theta[1]), dW = exp(theta[2]))
}
model <- form(dlmMLE(Nile, parm = c(1, 1), form)$par)
filtered <- dlmFilter(Nile, model)
autoplotly(filtered)
```

Additionally, **autoplotly** plots the optimal break points where possible structural changes happen in the regression models built by the `strucchange::breakpoints`, shown in Figure 6.

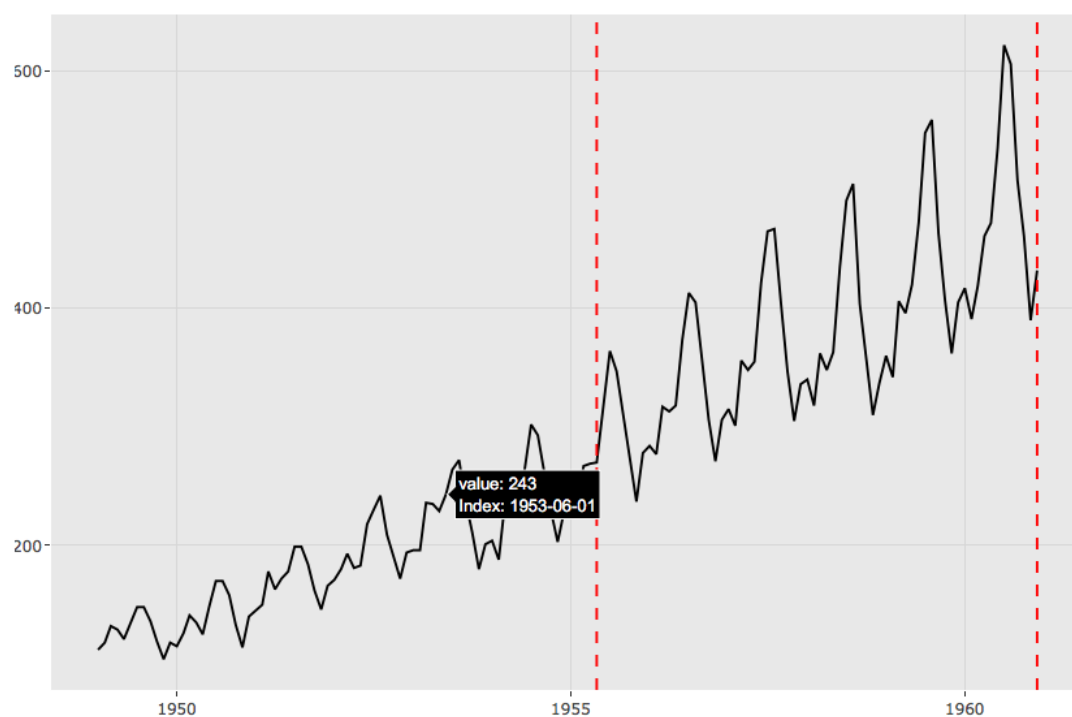
```
library(strucchange)
autoplotly(breakpoints(Nile ~ 1), ts.colour = "blue", ts.linetype = "dashed",
           cpt.colour = "dodgerblue3", cpt.linetype = "solid")
```

The **autoplotly** can also automatically generate interactive plots for results produced by **splines**, shown in Figure 7

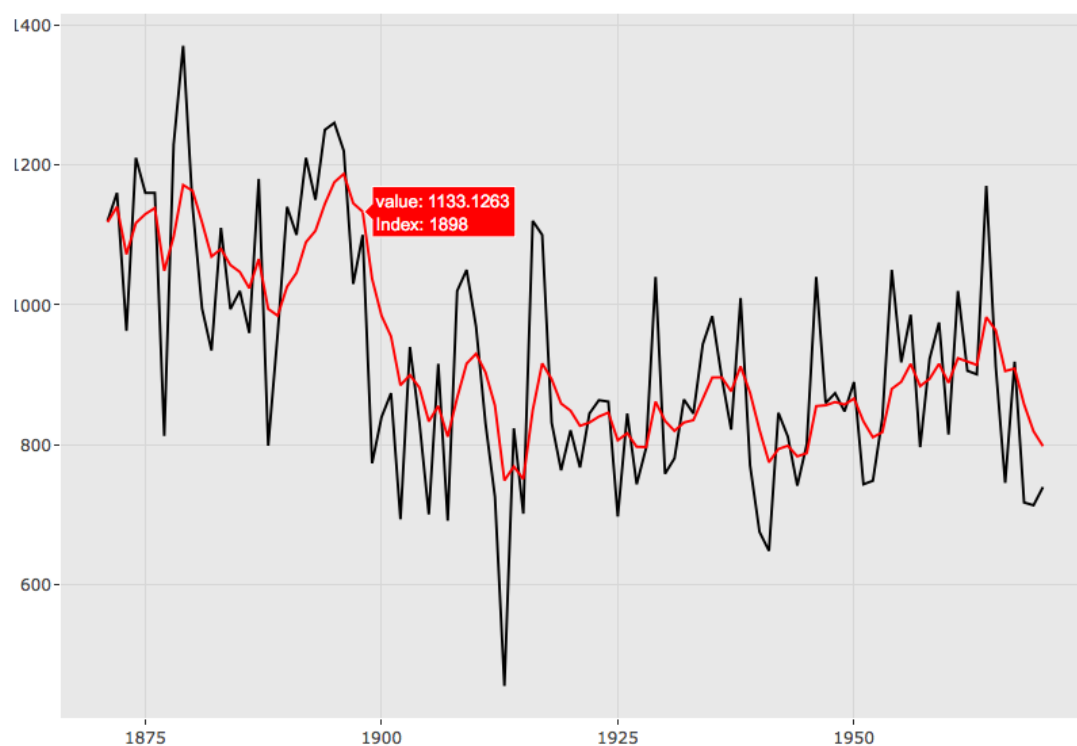
```
library(splines)
autoplotly(ns(diamonds$price, df = 6))
```

## Summary

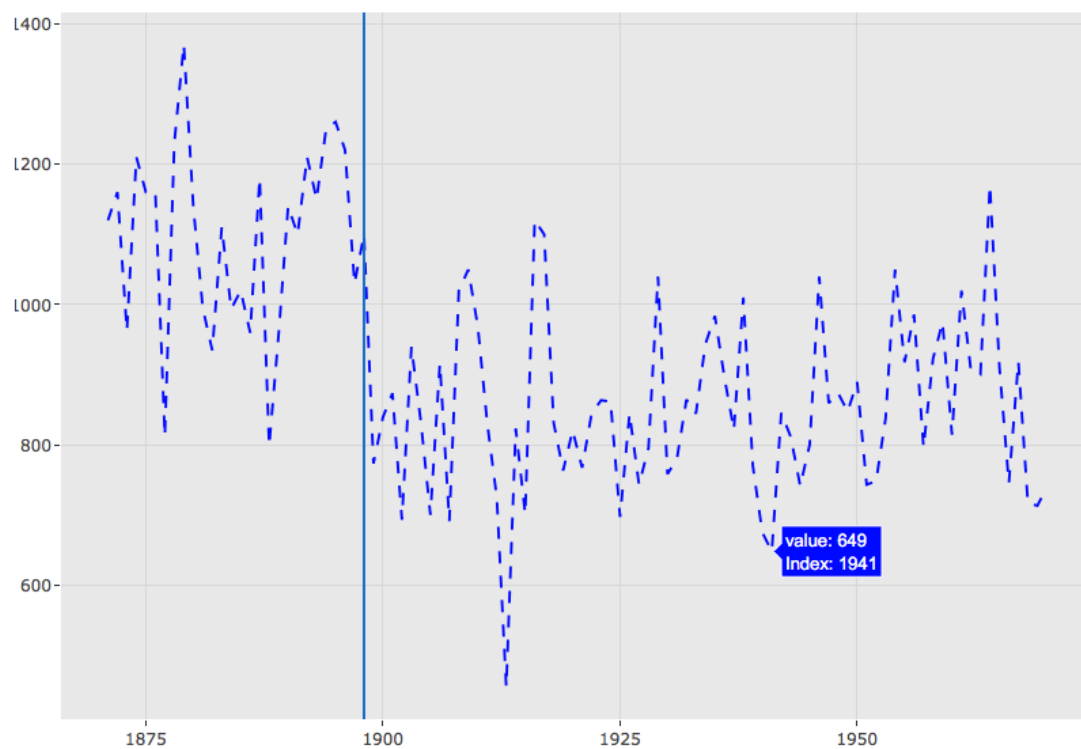
This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the [Instructions for Authors](#).



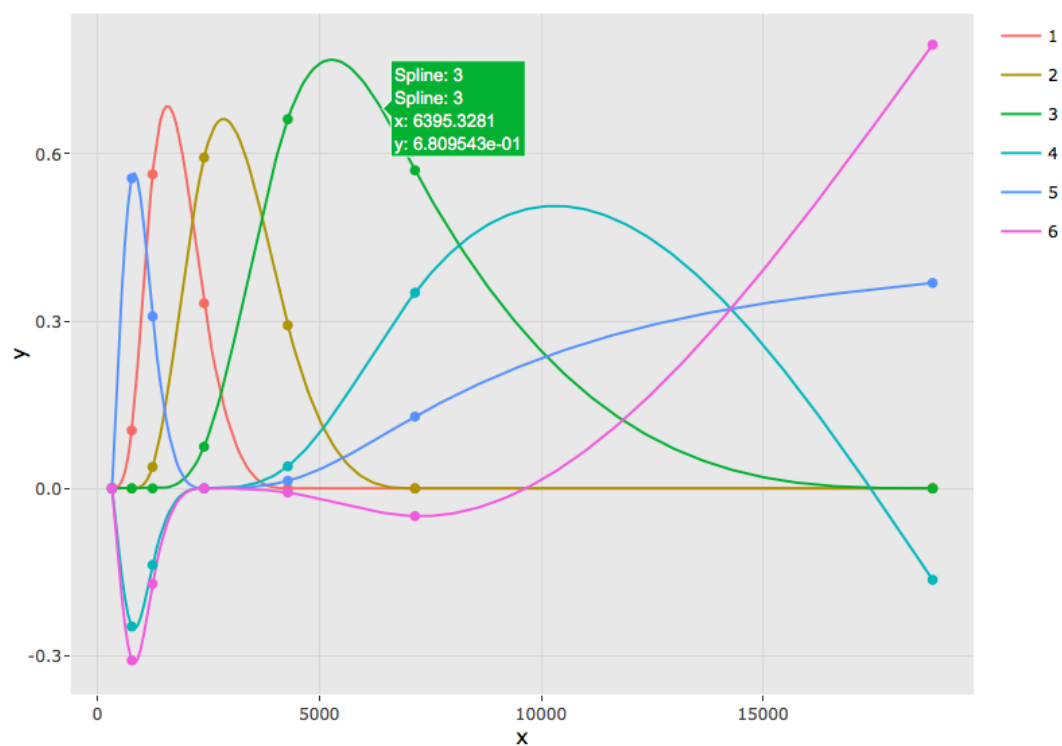
**Figure 4:** Change points with optimal positioning for AirPassengers.



**Figure 5:** Smoothed time series by Kalman filter.



**Figure 6:** Optimal break points with possible structural changes.



**Figure 7:** B-spline basis points for natural cubic spline with boundary knots.

## Bibliography

- M. Bostock, K. Russell, G. Aisch, and A. Pearce. *d3r: 'd3.js' Utilities for R*, 2017. URL <https://CRAN.R-project.org/package=d3r>. R package version 0.7.1. [p]
- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2017. URL <https://CRAN.R-project.org/package=shiny>. R package version 1.0.5. [p]
- D. Gohel. *ggiraph: Make 'ggplot2' Graphics Interactive*, 2017. URL <https://CRAN.R-project.org/package=ggiraph>. R package version 0.4.2. [p]
- R. J. Hyndman. *forecast: Forecasting functions for time series and linear models*, 2015. URL <http://github.com/robjhyndman/forecast>. R package version 6.2. [p]
- R. Killick, K. Haynes, and I. A. Eckley. *changepoint: An R package for changepoint analysis*, 2016. URL <http://CRAN.R-project.org/package=changepoint>. R package version 2.2.1. [p]
- G. Petris. An R package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, 2010. URL <http://www.jstatsoft.org/v36/i12/>. [p]
- D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. URL <http://lmdvr.r-forge.r-project.org>. ISBN 978-0-387-75968-5. [p]
- C. Sievert, C. Parmer, T. Hocking, S. Chamberlain, K. Ram, M. Corvellec, and P. Despouy. *plotly: Create Interactive Web Graphics via 'plotly.js'*. <https://plot.ly/r/>, [https://cpsievert.github.io/plotly\\_book/](https://cpsievert.github.io/plotly_book/), <https://github.com/ropensci/plotlyreadme>. [p]
- Y. Tang. *autoplotly: Automatic Generation of Interactive Visualizations for Popular Statistical Results*. URL <https://github.com/terrytangyuan/autoplotly>. R package version 0.1.0. [p]
- Y. Tang, M. Horikoshi, and W. Li. ggfortify: Unified interface to visualize statistical result of popular r packages. *The R Journal*, 8, 2016. URL <https://journal.r-project.org/>. [p]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer Science & Business Media, 2009. [p]
- A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. strucchange: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38, 2002. URL <http://www.jstatsoft.org/v07/i02/>. [p]

Yuan Tang  
H2O.ai  
2309 Wake Robin Drive  
West Lafayette, IN 47906  
[terrytangyuan@gmail.com](mailto:terrytangyuan@gmail.com)