File\_5\_Extending\_ggplotly

# 1.2 Extending ggplotly()

## 1.2.1 Customizing the layout

### dragmode

p <- ggplot(fortify(gold), aes(x, y)) + geom\_line()  
gg <- ggplotly(p)

## We recommend that you use the dev version of ggplot2 with `ggplotly()`  
## Install it with: `devtools::install\_github('hadley/ggplot2')`

layout(gg, dragmode = "pan")

### range sliders

p <- ggplot(fortify(gold), aes(x, y)) + geom\_line()  
gg <- ggplotly(p)

## We recommend that you use the dev version of ggplot2 with `ggplotly()`  
## Install it with: `devtools::install\_github('hadley/ggplot2')`

rangeslider(gg)

## 1.2.2 Modifying layers

As mentioned previously, ggplotly() translates each ggplot2 layer into one or more plotly.js traces. In this translation, it is forced to make a number of assumptions about trace attribute values that may or may not be appropriate for the use case. The style() function is useful in this scenario, as it provides a way to modify trace attribute values in a plotly object. Before using it, you may want to inspect the actual traces in a given plotly object using the plotly\_json() function. This function uses the listviewer package to display a convenient interactive view of the JSON object sent to plotly.js (de Jong and Russell 2016). By clicking on the arrow next to the data element, you can see the traces (data) behind the plot. In this case, we have three traces: one for the geom\_point() layer and two for the geom\_smooth() layer.

p <- ggplot(mtcars, aes(x = wt, y = mpg)) +  
 geom\_point() + geom\_smooth()  
  
plotly\_json(p)

## We recommend that you use the dev version of ggplot2 with `ggplotly()`  
## Install it with: `devtools::install\_github('hadley/ggplot2')`

## `geom\_smooth()` using method = 'loess'

Say, for example, we’d like to display information when hovering over points, but not when hovering over the fitted values or error bounds. The ggplot2 API has no semantics for making this distinction, but this is easily done in plotly.js by setting the hoverinfo attribute to “none”. Since the fitted values or error bounds are contained in the second and third traces, we can hide the information on just these traces using the traces attribute in the style() function. Generally speaking, the style() function is designed modify attribute values of trace(s) within a plotly object, which is primarily useful for customizing defaults produced via ggplotly().

style(p, hoverinfo = "none", traces = 2:3)

## We recommend that you use the dev version of ggplot2 with `ggplotly()`  
## Install it with: `devtools::install\_github('hadley/ggplot2')`

## `geom\_smooth()` using method = 'loess'