

Installation

```
install.packages("ggvis")
library(ggvis)
```

-install.packages("ggvis") will install all the required packages you need for visualization through ggvis
-library(ggvis) will call the ggvis package to be used in your visualization

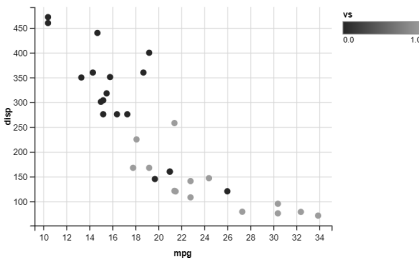
Layers

Simple Layer

```
mtcars %>% ggvis(~mpg, ~disp,
fill = ~vs) %>% layer_points()
```

Here I am using the dataset mtcars and visualising it through layer points.

Output

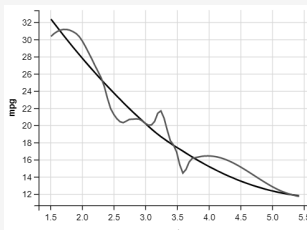


Multiple Layer

```
mtcars %>% ggvis(~wt,
~mpg) %>%
layer_smooths(span
= 1) %>%
layer_smooths(span
= 0.3, stroke := "-
red")
```

I have taken the mtcars dataset and visualized the multiple layers using different strokes

Output

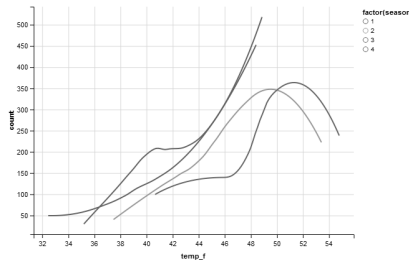


Group_by

```
train_tbl %>%
group_by(season) %>%
ggvis(~temp_f, ~count,
stroke = ~factor(se-
ason)) %>%
layer_smooths()
```

I have taken season dataset here, and season is a categorical variable. And we have grouped it and then used stroke to highlight the different seasons.

Output



Popular In-Built plot types

1. layer_points()
2. layer_lines()
3. layer_bars()
4. layer_smooths()
5. layer_histograms()

Global Vs Local properties

A property that is set inside ggvis() is applied globally. While a property set inside layer_<marks>() is applied locally. Local properties can override global properties when applicable.

Graphics

The graphics produced by ggvis are fundamentally web graphics and work very differently from traditional R graphics. This allows us to implement exciting new features like interactivity
The goal of ggvis is to make it easy to build interactive graphics for exploratory data analysis. ggvis has a similar underlying theory to ggplot2 (the grammar of graphics).

Scale Types

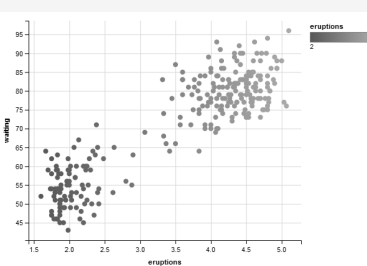
Any visual property in the visualization can be adjusted with scale(). ggvis provides several different functions for creating scales:



Scale Types (cont)

```
scale_datetime(), scale_logical(), scale_nominal(), scale_numeric(), scale_singular()
Code
faithful %>%
ggvis(~eruptions,~waiting,
fill = ~eruptions) %>%
layer_points() %>%
scale_numeric("fill", range
= c("red","orange"))
```

Output



ggvis & interaction ()

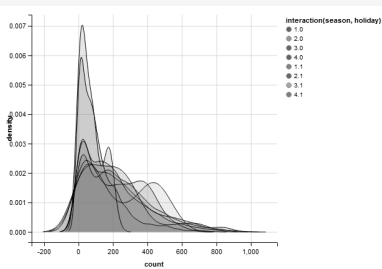
```
train_tbl %>%
group_by(season,holiday) %>%
ggvis(~count, fill = ~inter-
action(season,holiday)) %>%
layer_densities()
```

We can also group data based on interaction of two or more variables. `group_by()` creates unique groups for each distinct combination of values within the grouping variables.

`ungroup()` can remove the grouping information.

`interaction()` can map the properties to unique combinations of the variables

Output



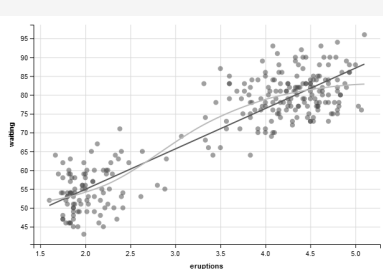
Model Prediction

```
faithful %>%
ggvis(~eruptions,~waiting)
%>%
layer_points(fill := "green", fillOpacity := 0.5) %>%
layer_model_predictions(-
model = "lm", stroke := "red") %>%
layer_smooths(stroke := "skyblue")
```

`layer_model_predictions()` plots the prediction line of a model fitted to the data.

`layer_model_predictions(model = "lm")`

Output



Interactive Plots

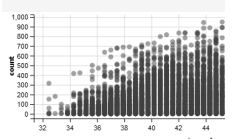
ggvis comes several

`input_checkbox()`,
`input_checkboxgroup`
`input_numeric()`,
`input_radiobuttons()`,
`input_select()`,
`input_slider()`, and `inp`

`label = "ABCD " , cho`
`black")` -
`value = "black"` - Use `text()`
`map = as.name` used
to return variable nam

Are the common argu
these functions.

Output



Choose colour
 red
 blue
 green
 orange



By **Anuj Shukla 18BCE0163**
(anujshukla)
cheatography.com/anujshukla/

Published 27th September, 2020.
Last updated 27th September, 2020.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>