

ggvis & Group_by

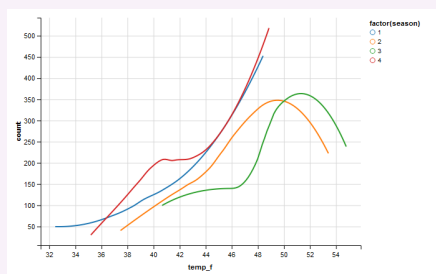
When these 2 are used in conjunction, we can create a visual property in the visualization can be adjusted.

Code:

```
train_tbl %>%
  group_by( season ) %>%
  ggvis( temp_f, count, stroke = ~factor( season ) ) %>%
  >%
  layer_ smoo ths()
```

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

Output



In-Built plot types

1. layer_points()
2. layer_lines()
3. layer_bars()
4. layer_smoo ths()
5. layer_ his tog rams()

Most popular ones cited

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.

Scale Types

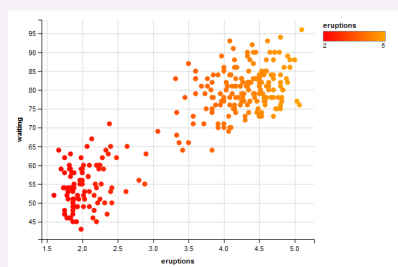
Any visual property in the visualization can be adjusted. ggvis provides several different functions for creating scales.

```
scale_ dat eti me(), scale_ log ical(),
um eric(), scale_ sin gular()
```

Code

```
ggvis( eruptions, waiting, fill = ~eruptions ) %>%
  layer_ poi nts() %>%
  scale_ num eri c("fill", range = c("red", "orange"))
```

Output



ggvis & interaction ()

We can also group data based on interaction of two or more variables.

group_by() creates unique groups for each distinct values within the grouping variables.

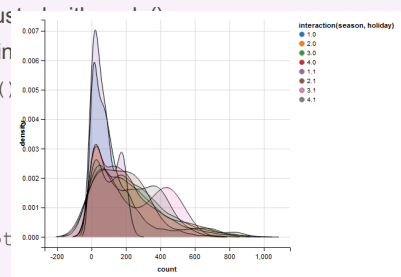
ungroup() can remove the grouping information.

interaction() can map the properties to unique values

Code:

```
train_tbl %>%
  group_by( sea son , ho liday ) %>%
  ggvis( count, fill = intera cti on( sea son, ho liday ) ) %>%
  layer_ den sit ies()
```

Output



Model Prediction

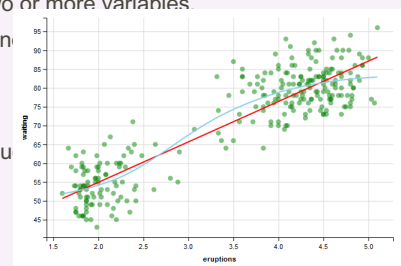
layer_model_predictions() plots the prediction

layer_ mod el_ pre dic tio ns(mod

Code:

```
faithful %>%
  ggvis( eruptions, waiting ) %>%
  layer_ poi nts ( fill := "gre en" )
  layer_ mod el_ pre dic tio ns( m
ed ) %>%
  layer_ smoo th s( stroke := "sk
```

Output



Interactive Plots

ggvis comes several widgets such as

```
input_ che ckb ox(),
input_ che ckb oxg roup(),
input_ num eric(),
input_ rad iob utt ons(),
input_ sel ect(),
input_ sli der(), and input_ text
().
```

label = "ABCD", choices = c("red", "black") - value = "black" - Used with input_text()



By shanly3011

Published 10th April, 2015.

Last updated 12th May, 2016.

Page 1 of 2.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

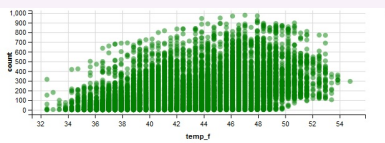
<https://readable.com>

Interactive Plots (cont)

map = as.name used when we want to return variable names

Are the common arguments inside these functions.

Output



Choose colour

- red
- blue
- green
- orange

Legends & Axis

Axis

You can add axes with `add_axis()`

Syntax:

```
faithful %>%
```

```
ggvis(eruptions, waiting) %>%
```

```
add_axis("x", label = "Eruptions", values = c(1, 2, 3, 4), subdivide = 9, orient = "p") %>%
```

```
layer_points()
```

Legends

ggvis adds a legend for each property that is specified. To combine multiple legends into a single legend with common values, use a vector of property names.

```
add_legend()
```

```
hide_legend()
```

Syntax

```
faithful %>%
```

```
ggvis(waiting, eruptions, opacity := 0.6,
```

```
  fill = factor(round(eruptions)), shape = factor(round(eruptions)),
```

```
  size = ~round(eruptions)) %>%
```

```
layer_points() %>%
```

```
add_legend(c("fill", "shape", "size"),
```

```
  title = "~duration (m)", values = c(2, 3, 4, 5))
```



By shanly3011

Published 10th April, 2015.

Last updated 12th May, 2016.

Page 2 of 2.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>