

ggvis & Group_by

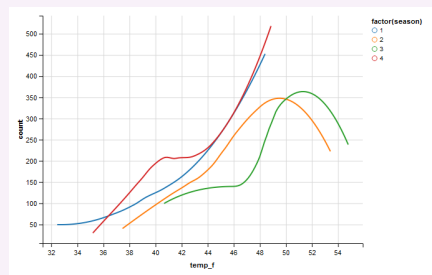
When these 2 are used in conjunction, we can create a useful property in the visualization can be adjusted. ggvis provides several different functions for creating visualizations.

Code:

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

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_smooths()
5. layer_histograms()

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

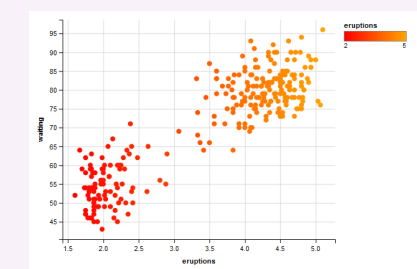
Any visual property in the visualization can be adjusted. ggvis provides several different functions for creating visualizations.

```
scale_datetime(), scale_logical(),
umeric(), scale_singular()
```

Code

```
ggvis(eruptions,waiting, fill = ~eruptions) %>%
layer_points() %>%
scale_numeric("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.

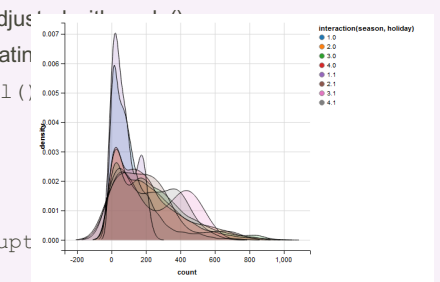
ungroup() can remove the grouping information.

interaction() can map the properties to unique variables

Code:

```
train_tbl %>%
group_by( season ,holiday) %>%
ggvis(count, fill = interaction( season ,holiday
)) %>%
layer_densities()
```

Output



Model Prediction

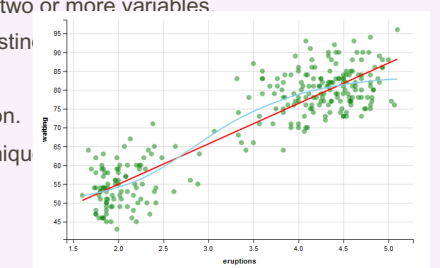
layer_model_predictions() plots the prediction

layer_model_predictions(model)

Code:

```
faithful %>%
ggvis(eruptions,waiting) %>%
layer_points (fill := "green")
layer_model_predictions( model ) %>%
layer_smooths(stroke := "sk")
```

Output



Interactive Plots

ggvis comes several widgets such as

```
input_checkbox(),
input_checkbox_group(),
input_umeric(),
input_rad iob utt ons(),
input_ select(),
input_ slider(), and input_ text
().
label = "ABCD ", choices = c("red","black") -
value = "black" - Used with input_text()
```



By shanly3011

cheatography.com/shanly3011/

Published 10th April, 2015.

Last updated 12th May, 2016.

Page 1 of 2.

Sponsored by Readable.com

Measure your website readability!

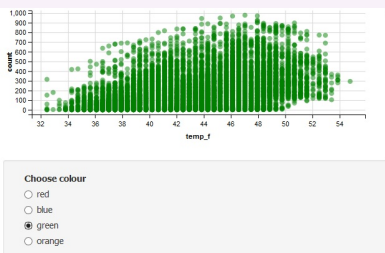
<https://readable.com>

Interactive Plots (cont)

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

Are the common arguments inside these functions.

Output



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 = "top") %>%
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**

Measure your website readability!

<https://readable.com>