Cheatography

Data Visualization in R through ggvis Cheat Sheet

by Anuj Shukla 18BCE0163 (anujshukla) via cheatography.com/126293/cs/24471/

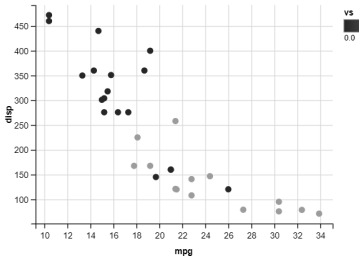
Layers

Simple Layer

```
mtcars %>% ggvis(mpg, disp, fill = ~vs) %>% layer_ poi nts()
```

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

Output

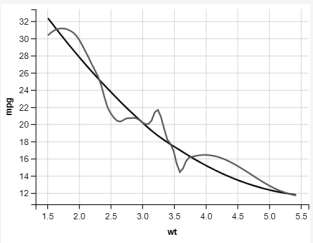


Multiple Layer

```
mtcars %>% ggvis(wt, mpg) %>%
    lay er_ smo oth - s(span = 1) %>%
    lay er_ smo oth - s(span = 0.3, stroke := " red ")
```

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

Output



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

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.

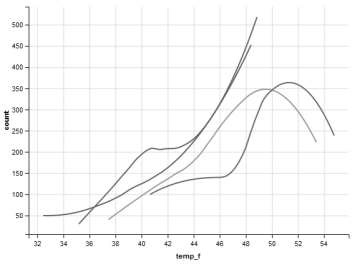
ics

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).

Popular In-Built plot types

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

Output



Group_by

```
train_tbl %>%
group  _by  (s e ason) %>%
ggvis ( t emp _f, count, stroke = ~facto  r( s - e ason)) %>%
layer  _sm  oo t hs()
```

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.

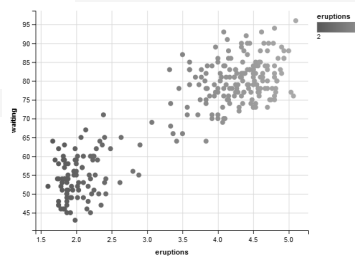
Scale Types

```
Any visual property in the visual ization 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 %>%
ggivs(eruptions, waiting, fill = ~eruptions) %>%
layer_points() %>%
scale_numeric("fill", range = c("red","orange"))

t



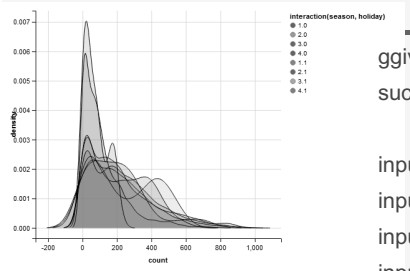
ggvis & interaction ()

```
train_tbl %>%
group  _by  (s e a so  - n,h  ol iday) %>%
ggvis ( c ount, fill = inter  act  io n ( se  - aso  n, h o li  day)) %>%
layer  _de  ns i t ies()
```

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

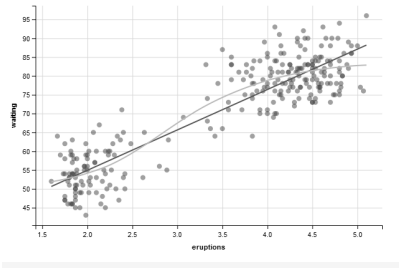


Model Prediction

```
faithful %>%
ggvis(er upt  ion-s,
wai ting) %>%
layer_  po i n ts  (fill
:= "  gr e e n",
fillOp  acity := 0.5)
%>%
layer_  mo d e l_  -
pre  di c t io  ns(  -
model = "  lm ", stroke
:= "  re d ") %>%
layer_  sm o o th  -
s(s  troke := "  sk y -
b lu  e")
```

layer_model_predictions() plots the prediction line of a model fitted to the data.
layer_model_predictions(model = "lm")

Output



Interactive Plots

ggivs comes several widgets such as

input_checkbox(),
input_checkboxgroup(),
input_numeric(),
input_radiobuttons(),
input_select(),
input_slider(), and input_text().

label = "ABCD " , choices = c("r-ed","black") -
value = "black" - Used with input_text()
map = as.name used when we want to return variable names

Are the common arguments inside these functions.

Output