Data visualization

With ggvis

Cheat sheet

Basic

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), but it's expressed a little differently, and adds new features to make your plots interactive.

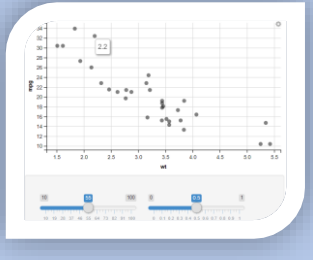
Build a graph with ggvis()

data | aesthetic mappings

```
ggvis(mtcars, x=~wt, y=~disp)
```

Template

```
mtcars %>%
ggvis(x = ~wt, y = ~disp) +
    layer_points(size := input_slider(10, 100),
                 opacity := input_slider(0, 1))%>%
    add_tooltip(function(df) df$wt)
```

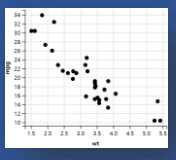


ggvis vs ggplot2

- Basic naming conversions from **ggplot2** to **ggvis**:
  - layer, geom -> layer function
  - stat -> compute function
  - aes() -> props()
  - ggplot() -> ggvis()
- ggvis has a function interface so you combine components using %>%, not + as in ggplot2.
- Using **ggvis**() without adding any layers is analogous to **qplot**()
- **ggvis** makes fewer assumptions about the type of data - data does not have to be a **data frame** until it has been processed by a transform.

ggvis()

How write basic ggvis?

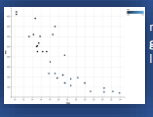


```
P <- ggvis(mtcars, x = ~wt, y = ~ mpg)
layer_points(p)
```

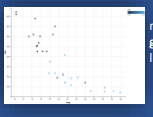
```
layer_points(ggvis(mtcars, x = ~wt, y = ~ mpg))
```

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points
```

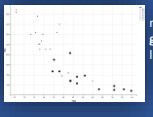
aesthetics



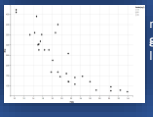
```
mtcars %>%
ggvis(~mpg, ~disp, stroke = ~vs) %>%
layer_points()
```



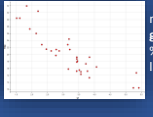
```
mtcars %>%
ggvis(~mpg, ~disp, fill = ~vs) %>%
layer_points()
```



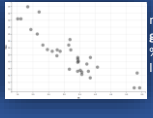
```
mtcars %>%
ggvis(~mpg, ~disp, size = ~vs) %>%
layer_points()
```



```
mtcars %>%
ggvis(~mpg, ~disp, shape = ~factor(cyl)) %>%
layer_points()
```

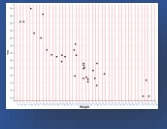


```
mtcars %>%
ggvis(~wt, ~mpg, fill := "red",  stroke := "black")
%>%
layer_points()
```



```
mtcars %>%
ggvis(~wt, ~mpg, size := 300, opacity := 0.4)
%>%
layer_points()
```

Add Theme

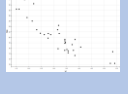


```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points() %>%
  add_axis("x", title = "Weight", ticks = 40,
    properties = axis_props(
      ticks = list(stroke = "red"),
      majorTicks = list(strokeWidth = 2),
      grid = list(stroke = "red"),
      labels = list(
        fill = "steelblue",
        angle = 50,
        fontSize = 14,
        align = "left",
        baseline = "middle",
        dx = 3
      ),
      title = list(fontSize = 16),
      axis = list(stroke = "#333", strokeWidth = 1.5)
    )
  )
```

Layers

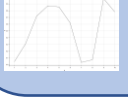
Simple layers

include primitives like points, lines and rectangles.



mtcars %>% **ggvis**(~wt, ~mpg) %>% **layer_points**()

with properties x, y, shape, stroke, fill, strokeOpacity, fillOpacity, and opacity



df %>% **ggvis**(~x, ~y) %>% **layer_paths**()

Compound layers

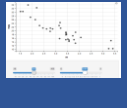
```
df <- data.frame(x = sin(t), y = cos(t))
```

df %>% ggvis(~x, ~y) %>% layer_paths()

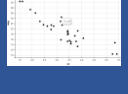
mtcars %>% ggvis(~mpg) %>% layer_histograms()

Interaction

Basic interaction



```
mtcars %>%
ggvis(~wt, ~mpg,
      size := input_slider(10, 100),
      opacity := input_slider(0, 1)
) %>%
layer_points()
```

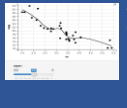


Tooltip

```
mtcars %>%
ggvis(~wt, ~mpg) %>%
layer_points()%>%
add_tooltip(function(df) df$wt)
```

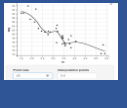
```
mtcars %>%
ggvis(~wt,
      width = input_slider(0, 2, step = 0.10, label = "width"),
      center = input_slider(0, 2, step = 0.05, label = "center")
) %>%
layer_histograms()
```

Input option



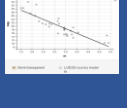
Slider input

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points() %>%
layer_smooths(
  span = input_slider(0.2, 1, value = 0.5, step = 0.05, label = "span")
)
```



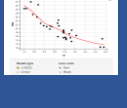
Numeric input

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points(size := input_numeric(value = 25, label = "Point size")) %>%
layer_smooths(span = input_numeric(value = 0.5, label = "Interpolation
points"))
```



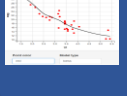
Checkbox input

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points(
  opacity := input_checkbox(label = "Semi-transparent",
                            map = function(val) ifelse(val, .3, 1)) %>%
layer_model_predictions(
  model = input_checkbox(label = "LOESS (curve) model fit",
                         map = function(val) ifelse(val, "loess", "lm")))
```



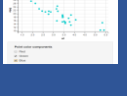
Radio buttons

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points() %>%
layer_model_predictions(
  model = input_radiobuttons(c("LOESS" = "loess", "Linear" = "lm"),
  label = "Model type"),
  stroke := input_radiobuttons(c("Red" = "red", "Black" = "black"),
  label = "Line color")
)
```



Text input

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points(fill := input_text(label = "Point color",
value = "red")) %>%
layer_model_predictions(model = input_text(label =
"Model type", value = "loess"))
```



Checkbox group

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg) %>%
layer_points(
  fill := input_checkboxgroup(
    choices = c("Red" = "r", "Green" = "g", "Blue" = "b"),
    label = "Point color components",
    map = function(val) {
      rgb(0.8 * "r" %in% val, 0.8 * "g" %in% val, 0.8 * "b" %in% val)
    }
  )
)
```

Map

A function with single argument x, the value of the control on the client. Returns a modified value.

```
new_vals <- input_select(c("Set A" = "A", "Set B" = "B"),
label = "Dynamically-generated column",
map = function(value) {
  vals <- switch(value,
    "A" = rep(c("One", "Two")),
    "B" = c("First", "Second", "Third", "Fourth"))
  rep(vals, length = nrow(mtcars))
})

mtcars %>% ggvis(x = ~wt, y = ~mpg, fill = new_vals) %>%
layer_points()
```

```
mtcars %>%
ggvis(x = ~wt, y = ~mpg,
  fill = input_select(c("mpg", "wt"),
  map = as.name)
) %>%
layer_points()
```