Changing the plot type

- In ggplot2 we use geom functions to determine the type of plot we create
- In ggvis we use layer functions
- Not all geoms are currently available as layers

Layers

Function Description

layer$_{p}oints Addsdataaspointslayer_{h}istograms Addsdataasahistograml$

```
> tubeData $ \%\>\% $
+ ggvis(x = ~Line, y = ~Excess) $ \%\>\% $
+ layer_boxplots()
```

Exercise

- Update the plot of mpg against wt to include a smooth line of the data
- Add a confidence interval to the smooth line and colour in red
- Add a regression line and colour it blue
- Create a boxplot of mpg split by cylinder (hint: the cylinder variable will need to be a factor)

Layers So far, you seen two layer functions:
layer$_p oints()and layer_h istograms().Therearemanyotherlayers, andthe$
Simple, which include primitives like points, lines and rectangles.
Compound, which combine data transformations with one or more simple layers.
All layer functions use the plural, not the singular. Think the verb, not the noun: Im going to layer some points onto my plot.
Simple layers There are five simple layers:
Points,
layer$_p oints(), withpropertiesx, y, shape, stroke, fill, strokeOpacity, fil$
mtcars 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 wt 10 12 14 16 18 20 22 24 26 28 30 32 34 mpg Paths and polygons, layer$_p aths()$.
df ¡- data.frame(x = 1:10, y = runif(10)) df 1 2 3 4 5 6 7 8 9 10 x 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 y If you supply a fill, youll get a polygon
t ¡- seq(0, 2 * pi, length = 100) df ¡- data.frame(x = sin(t), y = cos(t)) df -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 x -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 y Filled areas,

layer$_ribbons()$.$Usepropertiesyandy2tocontroltheextentofthearea$.
df ¡- data.frame(x = 1:10, y = runif(10)) df 1 2 3 4 5 6 7 8 9 10 x
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 y df 1 2 3 4 5 6 7 8
9 10 x -0.1 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 y + 0.1 Rectangles,
layer$_rects()$.$Thelocationandsizeoftherectangleiscontrolledbythex, x2,$
set.seed(1014) df ¡- data.frame(x1 = runif(5), x2 = runif(5), y1 =
runif(5), y2 = runif(5)) df 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 x1
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 y1 Text,
layer$_text()$.$Thetextlayerhasmanynewoptionstocontroltheapperanceof$
$text(thelabel), dxanddy(margininpixelsbetweentextandanchorpoint),$
df ¡- data.frame(x = 3:1, y = c(1, 3, 2), label = c("a", "b", "c"))
df 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 x 1.0 1.2 1.4 1.6 1.8
2.0 2.2 2.4 2.6 2.8 3.0 y a b c df 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6
2.8 3.0 x 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 y a b c df 1.0
1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 x 1.0 1.2 1.4 1.6 1.8 2.0 2.2
2.4 2.6 2.8 3.0 y a b c
Compound layers The four most common compound layers are:
layer$_lines()whichautomaticallyordersbythexvariable$:

t ¡- seq(0, 2 * pi, length = 20) df ¡- data.frame(x = sin(t), y = cos(t)) df -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 x -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 y df -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 x -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 y layer$_lines()isequivalenttoarrange() + layer_paths()$: df -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 x -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0 y layer$_histograms()andlayer_freqpolys()whichallowsyoutoexplorethedis$ mtcars ¿ Guessing width = 1 range / 24 10 12 14 16 18 20 22 24 26 28 30 32 34 mpg 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 count Or equivalently binned ¡- mtcars ¿ Guessing width = 1 range / 24 binned ggvis(x = xmin$_,x2 = xmax_,y2 = 0, y = count_,fill :=$ "$black$")$layer_rects()10121416182022242628303234 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0$ mtcars 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 wt 12 14 16 18 20 22 24 26 28 30 32 mpg Or equivalently smoothed ¡- mtcars smoothed 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 12 14 16 18 20 22 24 26 28 30 32 You can control the degree of wiggliness with the span

parameter:

span ¡- input$_{s}lider(0.2, 1, value = 0.75)mtcars > Warning : Can'toutputdynamic/interactiveggvisplotsinaknitrdocument. > Generatingastatic(non - dynamic, non - interactive)versionoftheplot.1.52.02.53.03.54.04.55.05.5wt1214161820

Interactivity

As well as mapping visual properties to variables or setting them to specific values, you can also connect them to interactive controls.

MAKING PLOTS INTERACTIVE Basic interactivity

The most basic interactivity we can add is "hover over" changes

We can change properties by using property.hover arguments
fill.hover := "red" ¿ tubeData %% + ggvis(Excess) %% + layer$_h istograms(fill.hover = "red")$

Interactive Input

We can also set properties to be the output of an interactive control opacity := input$_s lider(0, 1, label =$
$"Opacity")Weusethesetting" := "forthisinput$

We can optionally set labels next to the control

Interactive Input Functions Function Description
input$_{s}lider Slider to select values or ranges of values input_{c}heckbox A sing

¿ tubeData %% + ggvis(x = Month, y = Excess, + opacity :=
input$_slider(0, 1, +value = 0.7, label = "Opacity"), +size :=
$input_numeric(30, label = "Pointsize"), +fill :=$
$input_select(c("red","orange","blue"), +label =$
"Pointcolour")) %% +

Tooltips

add$_{t}ooltipallowsustoincludeotherbehaviourwhenwehover$

layer$_{p}oints()$

¿ tubeData %% + ggvis(x = Month, y = Excess) %% + layer$_{p}oints()$ %% + add$_{t}ooltip(function(data)data$Excess)

Exercise

- Update the previous plot of mpg against wt so points change colour when they hover over
- Add a tooltip that shows the value of mpg when the point is hovered over
- Add a slider for the span of the smooth line so that values can be set between 0 and 1

The following example allows you to control the size and opacity of points with two sliders:

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

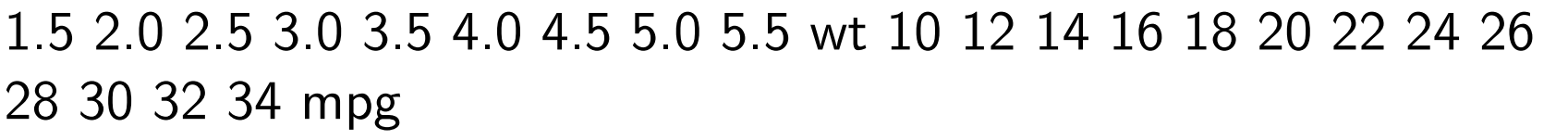
You can also connect interactive components to other plot parameters like the width and centers of histogram bins:

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

Behind the scenes, interactive plots are built with shiny, and you can currently only have one running at a time in a given R session. To finish with a plot, press the stop button in Rstudio, or close the browser window and then press Escape or Ctrl + C in R.

As well as
input$_slider(), ggvisprovidesinput_checkbox(), input_checkboxgroup(), inp$
You can also use keyboard controls with
left$_right()$and up$_down().Pressthelefta ndrightarrowstocontrolthesize$

```
keys_s <- left_right(10, 1000, step = 50)
mtcars %>% ggvis(~wt, ~mpg, size := keys_s, opacity := 0.
#> Warning: Can't output dynamic/interactive ggvis plots
#> Generating a static (non-dynamic, non-interactive) ver
```



Interactvity : Tooltips

You can also add on more complex types of interaction like tooltips:

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

Youll learn more about complex interaction in the interactivity vignette.

Finding out more

Rstudio are maintaining documentation on their webpage: http://ggvis.rstudio.com/

More details

There are other optional components that you can include:

- scales, to control the mapping between data and visual properties.
  These are described in the properties and scales vignette.
- legends and axes to control the appearance of the guides produced by the scales.
  See the axes and legends vignette for more details.

Viewing ggvis graphics

- ggvis uses **Vega** to render graphics in a web browser
- In RStudio the default it to use the "Viewer" pane
- From the web browser we can download SVG or png version of our graphics

ggvis and Vega/D3

- While ggvis is built on top of **vega**, which in turn borrows many ideas from **d3**, it is designed more for data exploration than data presentation.
- This means that ggvis makes many more assumptions about what youre trying to do: this allows it to be much more concise, at some cost of generality.

ggvis and Vega/D3

The main difference to vega is that ggvis provides a tree like structure allowing properties and data to be specified once and then inherited by children.

ggvis and Vega/D3

- Vega plays a similar role to ggvis that grid does to ggplot2. That means that you shouldnt have to know anything about vega to use ggvis effectively, and you shouldnt have to refer to the vega docs to solve common problems.
- However, some knowledge of how vega works is likely to be necessary when you start doing more complex layouts or when you start pushing the limits of the ggvis DSL.