Lecture 20: Interactive plots with ggvis

STAT598z: Intro. to computing for statistics

Vinayak Rao

Department of Statistics, Purdue University

`ggvis` is a simple way to get interactive plots

- provides a simpler interfact to shiny
- is still experimental

Some aesthetics also have different names:

- `color` becomes `stroke`

Like `ggplot` this expects a dataframe/tibble as an input

Some differences:

- add layes using `%>%` instead of `+`
- instead of `aes(color=group)`, write `color = ~group`
- we still write `color=clr_val`
- aesthetics have different names, e.g. `color` becomes `stroke`

In [59]:
```
library('tidyverse')
library('ggvis')
load('~/RSRCH/DATA/EconData/data/HomeValues.RData')
```

In [62]:
```
plt <-ggvis(HomeValues,x=~qtr,y=~Home.Value,stroke=~State) %>%
        layer_lines()
```

In [63]:
```
plt <- plt %>% hide_legend('stroke')
```

In [34]:
```
plt %>%  layer_points()
```

In [43]:
```
plt %>%  layer_points(size=1, fillOpacity=.1) # Bug!
```

`ggvis` uses both = and `:=` for assignments

Use `:=` when we expect a variable rather than a constant

- Use ':=' for quantities like size, color etc
- Use '=' for for quantities like `binwidth`, `opacity` etc

In [44]:
```
plt %>%  layer_points(size:=1, fillOpacity=.1)
```

So why use `ggvis` instead of `ggplot`?

- Interactive plots!

In [64]:
```
plt  %>%  add_tooltip(function(x) {paste(x$State,":",x$Home.Value)},'
hover')
```

Note: an *anonymous function* to print State and Value

In [65]:
```
plt <-ggvis(HomeValues,x=~qtr,y=~Home.Value,stroke=~State) %>%
     layer_lines() %>% layer_points(size:=input_slider(0,5))%>%
        hide_legend('stroke')
```

In [ ]:
```
plt  %>%  add_tooltip(function(x)
                {paste(x$State,":",x$Home.Value)},'hover')
```

In [66]:
```
plt <-ggvis(HomeValues x=~qtr,y=~Home.Value,stroke=~State) %>%
        layer_smooths(span:=input_slider(0,5)) %>%
        hide_legend('stroke')
```

```
Error in eval(expr, envir, enclos): object 'State' not found
Traceback:

1. ggvis(HomeValues, x = ~qtr, y = ~Home.Value, stroke = ~State) %
>%
 .     layer_smooths(`:=`(span, input_slider(0, 5))) %>% hide_lege
nd("stroke")
2. withVisible(eval(quote(`_fseq`(`_lhs`)), env, env))
3. eval(quote(`_fseq`(`_lhs`)), env, env)
4. eval(expr, envir, enclos)
5. `_fseq`(`_lhs`)
6. freduce(value, `_function_list`)
7. function_list[[i]](value)
8. layer_smooths(., `:=`(span, input_slider(0, 5)))
9. layer_model_predictions(vis, ..., model = "loess", formula = fo
rmula,
 .     model_args = list(span = span), se = se)
10. layer_f(vis, pipeline)
11. fun(vis)
12. emit_paths(x, props$stroke)
13. add_mark(vis, "line", props)
14. register_scales_from_props(vis, cur_props(vis))
15. add_scale_from_prop(vis, props[[i]])
16. vector_type(shiny::isolate(prop_value(prop, data())))
17. shiny::isolate(prop_value(prop, data()))
18. ..stacktraceoff..(ctx$run(function() {
  .     ..stacktraceon..(expr)
  . }))
19. ctx$run(function() {
  .     ..stacktraceon..(expr)
  . })
20. withReactiveDomain(.domain, {
  .     env <- .getReactiveEnvironment()
  .     .graphEnterContext(id)
  .     on.exit(.graphExitContext(id), add = TRUE)
  .     env$runWith(self, func)
  . })
21. env$runWith(self, func)
22. contextFunc()
23. ..stacktraceon..(expr)
24. prop_value(prop, data())
25. prop_value.prop_variable(prop, data())
26. eval(value(x$value), envir = data, enclos = x$env)
27. eval(expr, envir, enclos)
```

Error because `ggvis` doesn't do grouping for you (unlike `ggplot`)

In [69]:

```
plt <- HomeValues %>% group_by(State) %>% ggvis(x=~qtr,y=~Home.Value,
stroke=~State) %>%
        layer_smooths(span=input_slider(0,5)) %>%
        hide_legend('stroke')
```

In [70]:
```
ggvis(HomeValues) %>% layer_histograms(x=~Home.Value)
```

```
Guessing width = 50000 # range / 18
```

In [5]:
```
ggvis(HomeValues) %>%
  layer_histograms(x=~Home.Value,
                   width=input_slider(min=1000,max=100000))
```

```
1 2 3 4 5
```

In [1]:
```
plt <- HomeValues %>% group_by(State) %>%
        ggvis(x=~qtr,y=~Home.Value, stroke=~State) %>%
       filter(State %in% eval(input_select(choices =
                         unique(as.character(HomeValues$State)),
                  multiple=TRUE, label='States list'))) %>%
                    layer_lines(strokeWidth:=2)
```

```
Error in eval(expr, envir, enclos): could not find function "%>%"
Traceback:
```

Note the `eval`, this is because of we are calling `input_select` inside `filter`

In [58]:

```
state_map <- map_data('state');
my_state_map <- state_map
my_state_map$region <- tolower(state_map$region)
get_ab <- function(x)  state.abb[x ==tolower(state.name)]

state.name[51] <- "district of columbia"
state.abb[51] <- "DC"

 my_state_map %>%
  ggvis(~long, ~lat,fill=~input_slider(1,100,map= function(x) {x/1000
}))  %>%
  group_by(region) %>%
   layer_paths(strokeOpacity := 0.5,
                                       strokeWidth := 0.5) %>%
  hide_axis("x") %>% hide_axis("y") %>%
  set_options(width=960, height=600, keep_aspect=TRUE)
```

```
Error in vector_type.default(shiny::isolate(prop_value(prop, data(
))))): Unknown variable type: yearqtr
Traceback:

1. ggvis(HomeValues %>% group_by(State), x = ~qtr, y = ~Home.Value
) %>%
 .     filter(State %in% eval(input_select(choices = unique(as.cha
racter(HomeValues$State)),
 .         multiple = TRUE, label = "States list"))) %>% layer_lin
es(stroke = ~State,
 .     `:=`(strokeWidth, 2))
2. withVisible(eval(quote(`_fseq`(`_lhs`)), env, env))
3. eval(quote(`_fseq`(`_lhs`)), env, env)
4. eval(expr, envir, enclos)
5. `_fseq`(`_lhs`)
6. freduce(value, `_function_list`)
7. withVisible(function_list[[k]](value))
8. function_list[[k]](value)
9. layer_lines(., stroke = ~State, `:=`(strokeWidth, 2))
10. layer_f(vis, function(x) {
  .     x <- auto_group(x, exclude = c("x", "y"))
  .     x <- dplyr::arrange_(x, x_var)
  .     emit_paths(x, props(...))
  . })
11. fun(vis)
12. emit_paths(x, props(...))
13. add_mark(vis, "line", props)
14. register_scales_from_props(vis, cur_props(vis))
15. add_scale_from_prop(vis, props[[i]])
16. vector_type(shiny::isolate(prop_value(prop, data())))
17. vector_type.default(shiny::isolate(prop_value(prop, data())))
18. stop("Unknown variable type: ", paste0(class(x), collapse = "/
"))
```