GGFormula addons

A number of methods from the ggformula package can be used to provide additions to a graph. We'll look at some of these here.

gf_labs

gf_labs can be used for setting various labels on a graph.

```
driving %>% filter(miles <= 48) %>%
    gf_point(time~miles, color=~direction) %>%
    gf_labs(
        title = "The title",
        subtitle = "The subtitle",
        caption = "A caption",
        x = "The x axis label",
        y = "The y axis label",
        color = "The label for the color control")
```

gf_lims

gf_lims can be used to control the limits on the x and/or y axis, by specifying a start and/or an end for each direction.

```
driving %>% filter(miles <= 48) %>%
    gf_point(time~miles, color=~direction) %>%
    gf_lims(x = c(46, 48), y=c(NA, 70))
```

gf_abline, gf_hline, gf_vline

gf_abline, gf_hline and gf_vline can be used to add specific lines to a graph (horizontal, vertical or with arbitrary slope and intercept).

```
driving %>% filter(miles <= 48) %>%
    gf_point(time~miles) %>%
    gf_hline(yintercept=60, color="red")  %>%
    gf_abline(intercept=0, slope=60/55)    ## The y=x line, slope 1, 55 miles/hour
```

gf_facet_grid and gf_facet_wrap

gf_facet_grid and gf_facet_wrap allow us to create multiple panels depending on the option provided by a categorical variable. The _grid version allows one variable on each direction, while the _wrap variant allows a single variable, and it will wrap on as many lines as needed.

```
data(counties)
counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty) %>%
    gf_facet_wrap(~state)

counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty) %>%
    gf_facet_grid(~state)

counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty) %>%
    gf_lm() %>%
    gf_facet_grid(state~.)
```

gf_lm and gf_smooth

gf_lm and gf_smooth can be used for adding linear regression lines and smoothers to scatterplots.

```
data(counties)
counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty, color = ~state) %>%
    gf_lm() %>%
    gf_lm(hs_grad~poverty, color="black")

counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty, color = ~state) %>%
    gf_smooth() %>%
    gf_smooth(hs_grad~poverty, color="black")
```

gf_text

gf_text can be used to add text to specific locations. It can be a stand-alone graph or just an addon to a graph.

```
guns %>% gf_point(mort_rate~own_rate, size=2, color=~hdi, alpha=0.7) %>%
    gf_text(mort_rate~own_rate, label=~country, size=4, color="black", alpha=0.5, nudge_y=0.3
```

gf_theme

gf_theme can be used to set the overall theme for the graph, especially the background and axis colors etc. This can be one of the standard themes, and/or specific changes.

```
counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty, color = ~state) %>%
    gf_theme(theme_light())      ## also try theme_dark(), theme_linedraw(), theme_classic()
                                 ## theme_minimal(), theme_void()
```

gf_refine and scales

gf_refine can be used, in combination with a number of "scale" functions, to determine what colors and other attributes will be used for mapped parameters. These functions take the form: scale_xxx_yyy where the xxx part is the kind of attribute that we want to map (color, fill, shape, alpha etc) and the yyy part is the kind of functions we want to use for it.

For colors, a common solution is to use the ColorBrewer package[1] and choose one of the provided palettes.

For the axes, the scale_x_ ..... and scale_y_ .... functions offer some standard transformation (e.g. logarithmic).

```
counties %>% filter(state %in% c("Indiana", "Kentucky", "Ohio", "Illinois")) %>%
    gf_point(hs_grad~poverty, color = ~state) %>%
    gf_refine(scale_color_brewer(palette = "RdYlBu"),
              scale_y_continuous(labels = scales::percent_format(scale=1)),
              scale_x_log10())
```

[1] http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3