Topics in Data Visualization

Facetting And Themes

Jul 7 2014

Charlotte Wickham

vis.cwick.co.nz

Today

Quick recap of `group`

Facetting

Themes

Saving plots

Time for questions on Assn. #1?

Groups

For some geometric objects, you get one geometric object per group.

By default, groups are set to be the interaction of of all discrete variables in the plot.

Previous example:

- when Action is mapped to color, two groups, two lines
- when Action isn't in the plot, one group, one line

If the default isn't what you want you can map a variable to the group aesthetic.

```
ggplot(movies_lens, aes(x = year)) +
  geom_line(aes(y = n_movies, group = Action))
```

Switching discrete and continuous

Continuous -> Discrete

```
factor() # if the variable already only takes on
a fixed number of values

cut_interval() # to slice a continuous variable
in to bins of the same width

cut_number() # to slice a continuous variable in
to bins with the same number of observations
```

Discrete -> Continuous

```
as.numeric() # but be careful with factors!
```

```
ggplot(diamonds) +
  geom_point(aes(carat, price))

# hard to see anything going on
ggplot(diamonds) +
  geom_point(aes(carat, price, colour = depth))

# cut depth into five equisized groups, instead
ggplot(diamonds) +
  geom_point(aes(carat, price, colour = cut_number(depth, 5)))

# clarity is treated as a discrete variable becasue it's a factor
ggplot(diamonds) +
  geom_point(aes(carat, price, colour = clarity))

# this works, but in general I wouldn't reccomend turning categories
# into numbers.
ggplot(diamonds) +
  geom_point(aes(carat, price, colour = as.numeric(clarity)))

# You can also use these techniques to add summaries to a plot
ggplot(diamonds, aes(carat, price)) +
  geom_point(size = 0.5) +
  geom_boxplot(aes(group = cut_interval(carat, 20)))
```

More geoms

variability	**distribution**
errorbar	dotplot
linerange	boxplot
pointrange	violin
crossbar	
ribbon	histogram
	freqpoly
	density

Facets

Facets repeat a plot on splits of the data

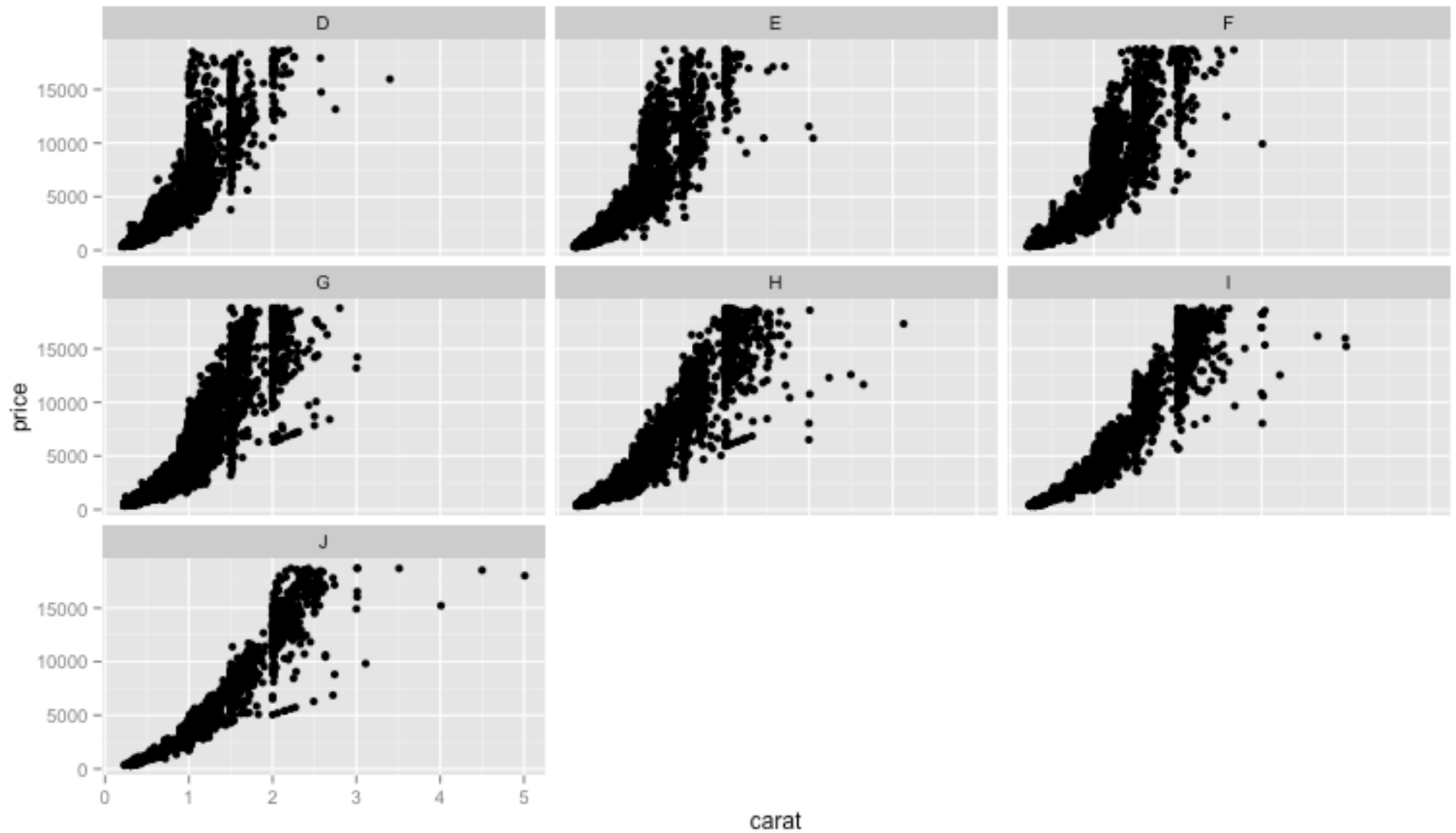
`+ facet_wrap(~ variable)` - one plot for each level of variable, arranged by wrapping the plots into a rectangular array.

`+ facet_grid(var_x ~ var_y)` - one plot for each combination of var_x and var_y levels, arranged in a grid, rows are constant var_y, columns constant var_x.

Allows the incorporation of additional variables

(rather than using aesthetics)

```
ggplot(diamonds) +
  geom_point(aes(carat, price)) +
  facet_wrap(~ color)
```

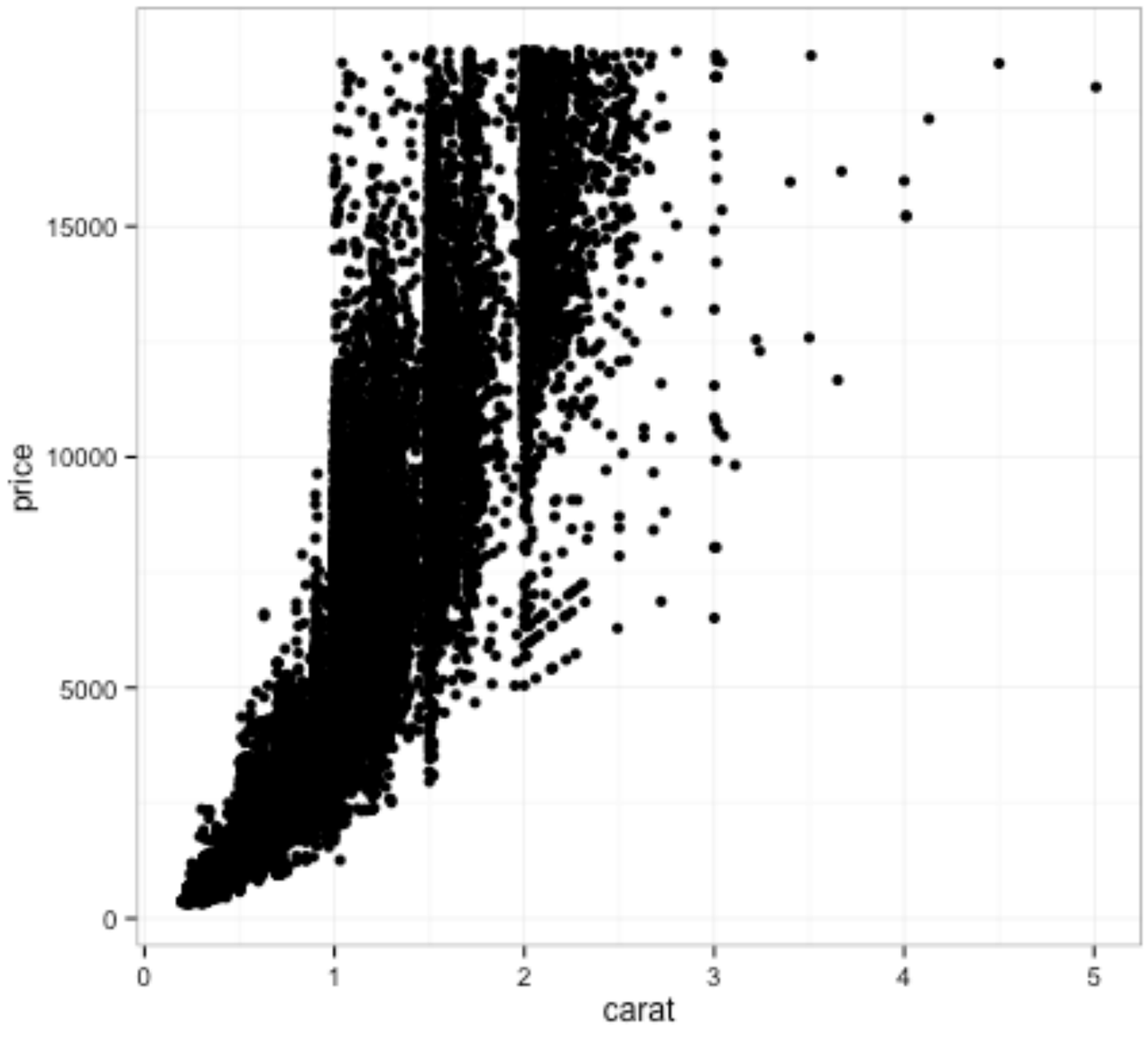


Themes

http://docs.ggplot2.org/dev/vignettes/themes.html

Themes control the appearance of all the non-data ink

```
+ theme_bw()
```



```
> theme_grey()
List of 40
 $ line                :List of 4
  ..$ colour  : chr "black"
  ..$ size    : num 0.5
  ..$ linetype: num 1
  ..$ lineend : chr "butt"
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ rect                :List of 4
  ..$ fill    : chr "white"
  ..$ colour  : chr "black"
  ..$ size    : num 0.5
  ..$ linetype: num 1
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ text                :List of 8
  ..$ family    : chr ""
  ..$ face      : chr "plain"
  ..$ colour    : chr "black"
  ..$ size      : num 12
  ..$ hjust     : num 0.5
  ..$ vjust     : num 0.5
  ..$ angle     : num 0
  ..$ lineheight: num 0.9
…
```


```
> theme_grey()
List of 40
 $ line                :List of 4
  ..$ colour  : chr "black"
  ..$ size    : num 0.5
  ..$ linetype: num 1
  ..$ lineend : chr "butt"
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ rect                :List of 4
  ..$ fill    : chr "white"
  ..$ colour  : chr "black"
  ..$ size    : num 0.5
  ..$ linetype: num 1
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ text                :List of 8
  ..$ family    : chr ""
  ..$ face      : chr "plain"
  ..$ colour    : chr "black"
  ..$ size      :Class 'rel'  num 0.8
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ strip.text          :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      :Class 'rel'  num 0.8
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ axis.line           : list()
  ..- attr(*, "class")= chr [1:2] "element_blank" "element"
 $ axis.text.x         :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : NULL
  ..$ vjust     : num 1
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ axis.text.y         :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : num 1
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ axis.ticks          :List of 4
  ..$ colour  : chr "grey50"
  ..$ size    : NULL
  ..$ linetype: NULL
  ..$ lineend : NULL
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ axis.title.x        :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ axis.title.y        :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : num 90
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ axis.ticks.length   :Class 'unit'  atomic [1:1] 0.15
  .. ..- attr(*, "unit")= chr "cm"
  .. ..- attr(*, "valid.unit")= int 1
 $ axis.ticks.margin   :Class 'unit'  atomic [1:1] 0.1
  .. ..- attr(*, "unit")= chr "cm"
  .. ..- attr(*, "valid.unit")= int 1
 $ legend.background   :List of 4
  ..$ fill    : NULL
  ..$ colour  : logi NA
  ..$ size    : NULL
  ..$ linetype: NULL
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ legend.margin       :Class 'unit'  atomic [1:1] 0.2
  .. ..- attr(*, "unit")= chr "cm"
  .. ..- attr(*, "valid.unit")= int 1
 $ legend.key          :List of 4
  ..$ fill    : chr "grey95"
  ..$ colour  : chr "white"
  ..$ size    : NULL
  ..$ linetype: NULL
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ legend.key.size     :Class 'unit'  atomic [1:1] 1.2
  .. ..- attr(*, "unit")= chr "lines"
  .. ..- attr(*, "valid.unit")= int 3
 $ legend.key.height   : NULL
 $ legend.key.width    : NULL
 $ legend.text         :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      :Class 'rel'  num 0.8
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ legend.text.align   : NULL
 $ legend.title        :List of 8
  ..$ family    : NULL
  ..$ face      : chr "bold"
  ..$ colour    : NULL
  ..$ size      :Class 'rel'  num 0.8
  ..$ hjust     : num 0
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ legend.title.align  : NULL
 $ legend.position     : chr "right"
 $ legend.direction    : NULL
 $ legend.justification: chr "center"
 $ legend.box          : NULL
 $ panel.background    :List of 4
  ..$ fill    : chr "grey90"
  ..$ colour  : logi NA
  ..$ size    : NULL
  ..$ linetype: NULL
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ panel.border        : list()
  ..- attr(*, "class")= chr [1:2] "element_blank" "element"
 $ panel.grid.major    :List of 4
  ..$ colour  : chr "white"
  ..$ size    : NULL
  ..$ linetype: NULL
  ..$ lineend : NULL
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ panel.grid.minor    :List of 4
  ..$ colour  : chr "grey95"
  ..$ size    : num 0.25
  ..$ linetype: NULL
  ..$ lineend : NULL
  ..- attr(*, "class")= chr [1:2] "element_line" "element"
 $ panel.margin        :Class 'unit'  atomic [1:1] 0.25
  .. ..- attr(*, "unit")= chr "lines"
  .. ..- attr(*, "valid.unit")= int 3
 $ panel.margin.x      : NULL
 $ panel.margin.y      : NULL
 $ strip.background    :List of 4
  ..$ fill    : chr "grey80"
  ..$ colour  : logi NA
  ..$ size    : NULL
  ..$ linetype: NULL
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ strip.text.x        :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ strip.text.y        :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      : NULL
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : num -90
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ plot.background     :List of 4
  ..$ fill    : NULL
  ..$ colour  : chr "white"
  ..$ size    : NULL
  ..$ linetype: NULL
  ..- attr(*, "class")= chr [1:2] "element_rect" "element"
 $ plot.title          :List of 8
  ..$ family    : NULL
  ..$ face      : NULL
  ..$ colour    : NULL
  ..$ size      :Class 'rel'  num 1.2
  ..$ hjust     : NULL
  ..$ vjust     : NULL
  ..$ angle     : NULL
  ..$ lineheight: NULL
  ..- attr(*, "class")= chr [1:2] "element_text" "element"
 $ plot.margin         :Class 'unit'  atomic [1:4] 1 1 0.5 0.5
  .. ..- attr(*, "unit")= chr "lines"
  .. ..- attr(*, "valid.unit")= int 3
 - attr(*, "class")= chr [1:2] "theme" "gg"
 - attr(*, "complete")= logi TRUE
```

Complete themes:

```
theme_grey()
theme_bw()
theme_minimal()
```

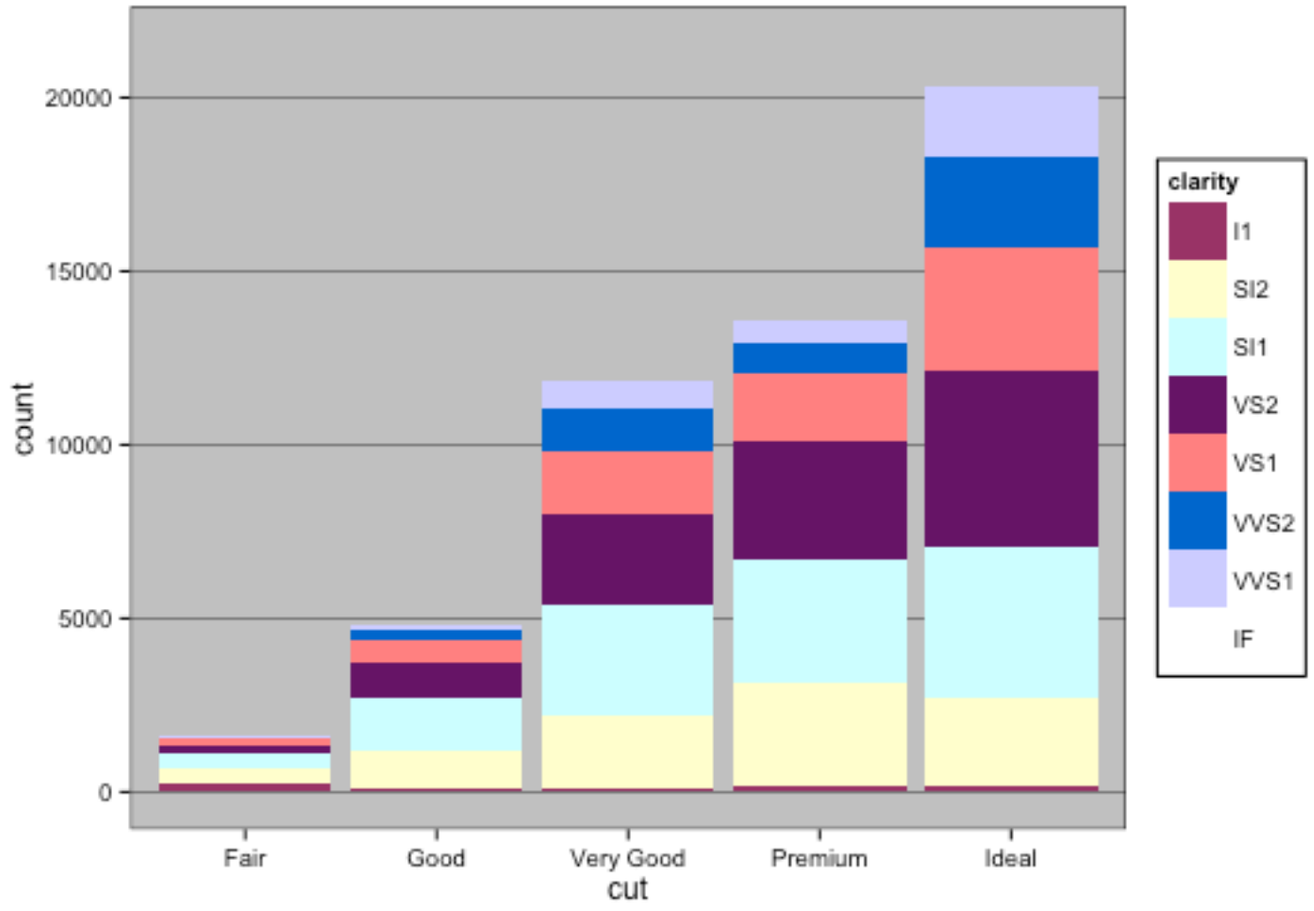
Change particular parts of a theme:

```
theme()
theme(axis.title.x =
element_text(family = "mono"))
?theme
```

More fonts

http://blog.revolutionanalytics.com/2012/09/how-to-use-your-favorite-fonts-in-r-charts.html

```
library(ggthemes)
ggplot(diamonds, aes(cut)) +
  geom_bar(aes(fill = clarity)) +
  scale_fill_excel() +
  theme_excel()
```



https://github.com/jrnold/ggthemes

Saving plots

To a variable:

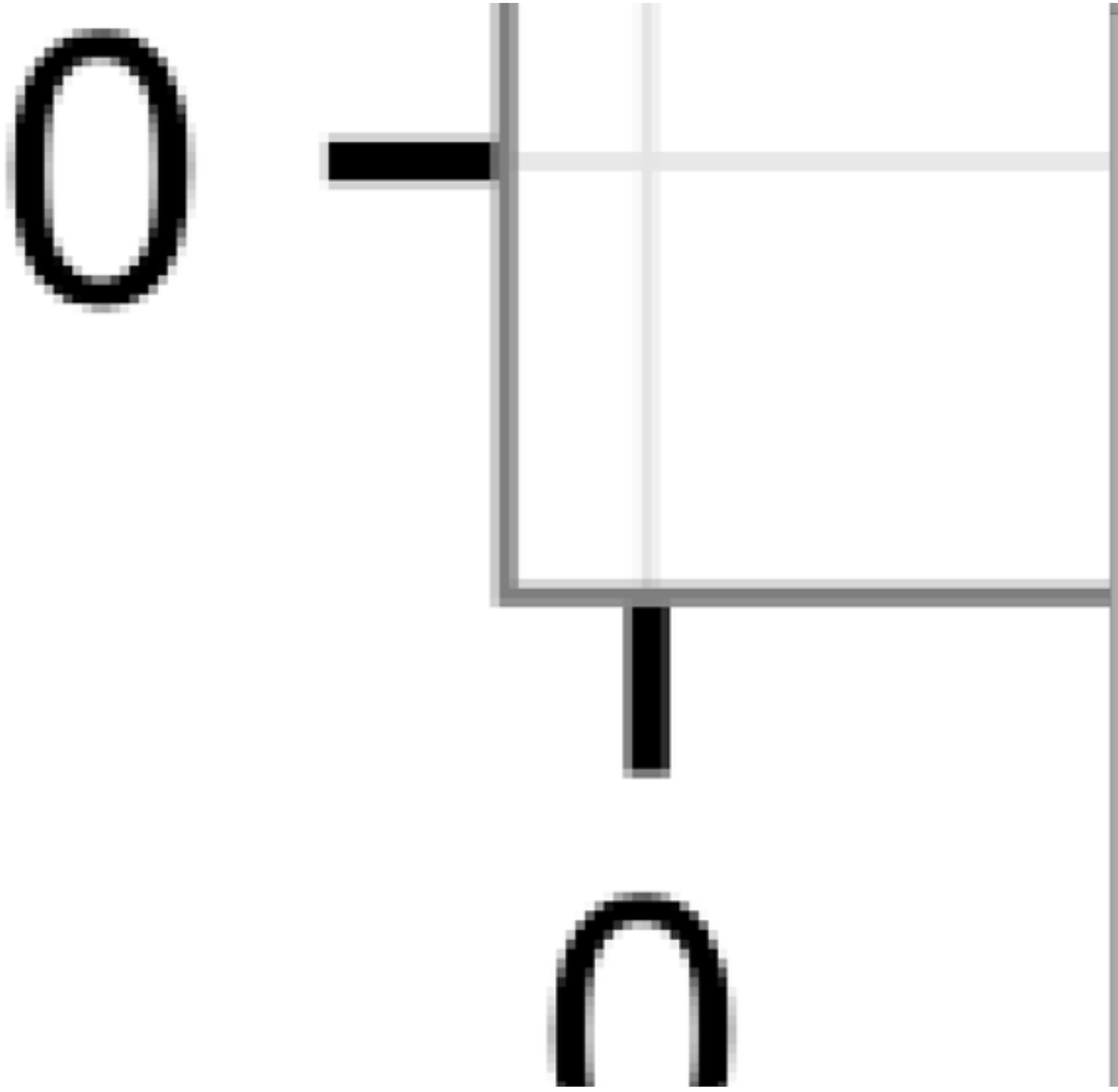
```
p <- ggplot(diamonds) +
  geom_point(aes(carat, price))
print(p)   # just p, if working interactively
p + theme_bw()
p + facet_wrap(~ color)
```

To a file:

```
ggsave("my_plot.pdf",height = 6, width = 8) # last plot
ggsave("my_plot.png",height = 6, width = 8)
ggsave("my_plot.pdf", p, height = 6, width = 8) # plot p
```

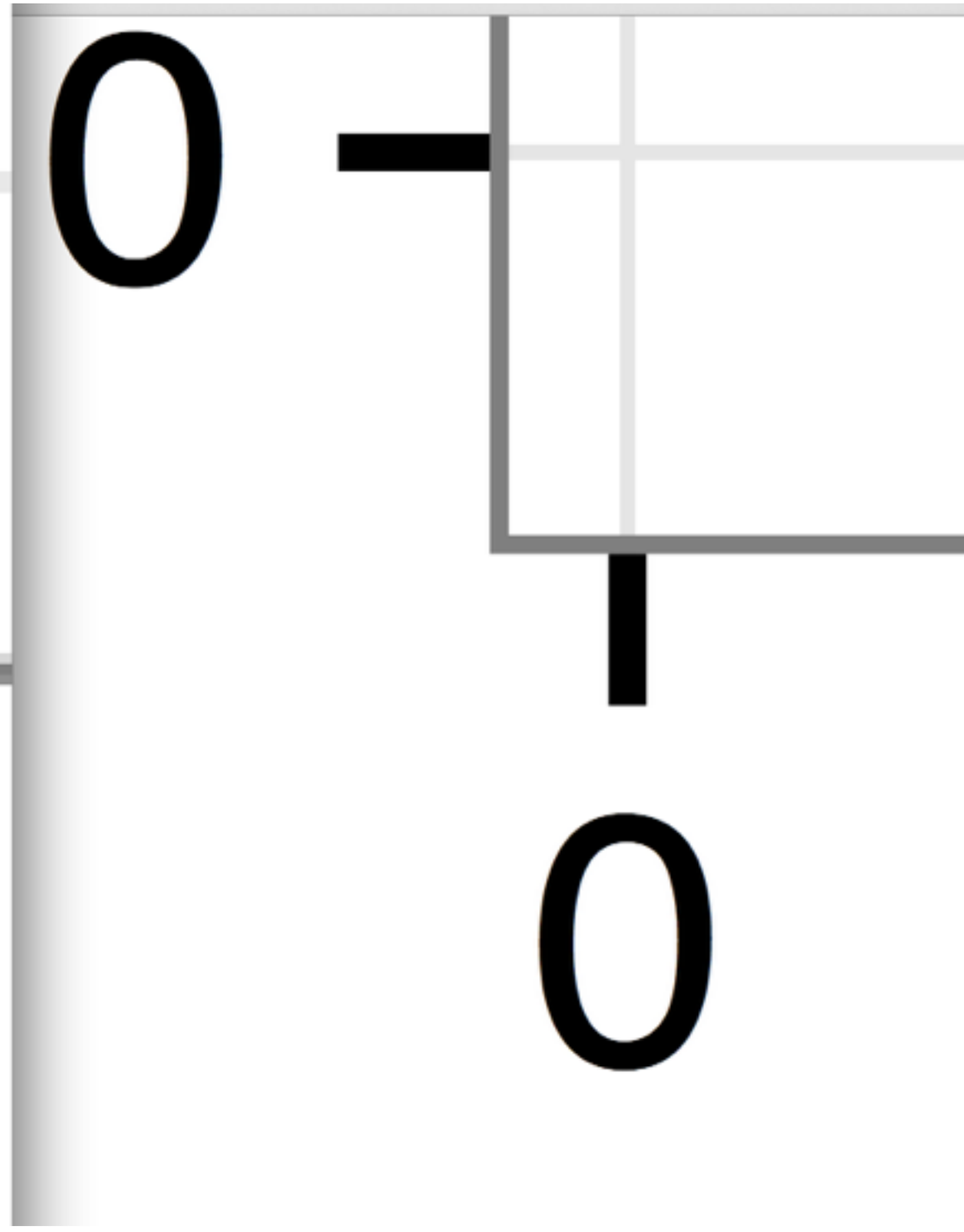
png

raster: coloring pixels



pdf

vector: instructions for drawing



Recommendation: use .pdf, unless there are 10,000s of points.