

# Polishing your plots

**Hadley Wickham**

**October 2009**



# Polishing

Most (99%?) of your plots don't need any polishing. They are **exploratory** plots, produced to help you understand the data.

When you need to **communicate** your findings, you need to spend time polishing your plots to eliminate distractions and focus on the story you want to tell.

1. Saving your work

2. Colour

3. Labels & ticks

4. Themes

**Saving your work**

```
# Classical approach
# If you're doing this inside a loop or function
# you'll need to explicitly print the plot
png("diamonds.png", width = 6, height = 6)
qplot(price, carat, data = diamonds)
dev.off()

# ggsave
qplot(price, carat, data = diamonds)
ggsave("diamonds.png")
```

```
# Selects graphics device based on extension
ggsave("diamonds.png")
ggsave("diamonds.pdf")

# Uses on-screen device size, or override with
# width & height (to be reproducible)
ggsave("diamonds.png", width = 6, height = 6)

# Outputs last plot by default, override
# with plot:
dplot <- qplot(carat, price, data = diamonds)
ggsave("diamonds.png", plot = dplot)

# Defaults to 300 dpi for png
ggsave("diamonds.png", dpi = 72)
```

Raster	Vector
pixel-based	instruction-based
png	pdf, wmf, eps
for plots with many points	for all other plots
ms office, web	latex

# Your turn

Save a pdf of a scatterplot of price vs carat. Open it up in adobe acrobat.

Save a png of the same scatterplot and embed it into a word or latex document.



Colour

# Colour theory

Colour most important aesthetic after position. Need to know a little theory to be able to use it effectively.

Colour spaces & colour blindness.

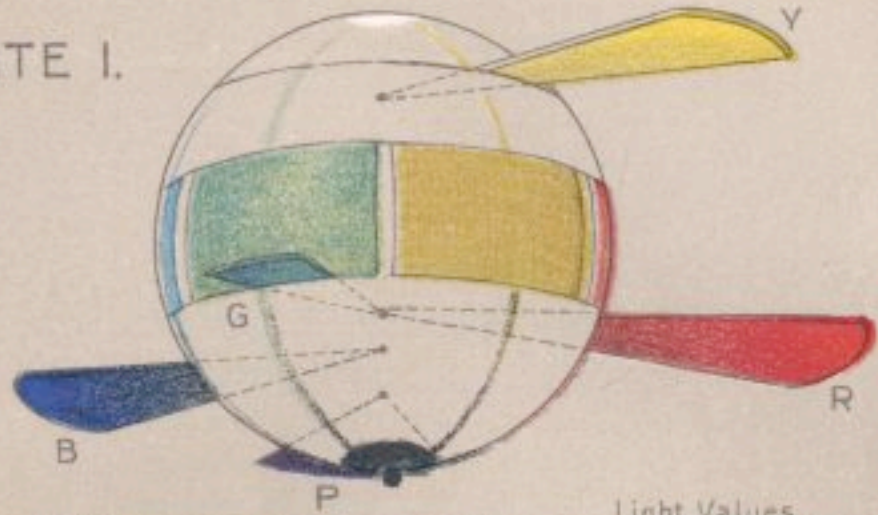
# Colour spaces

Probably most familiar with rgb: defines colour as mixture of red, green and blue. Matches physics of eye.

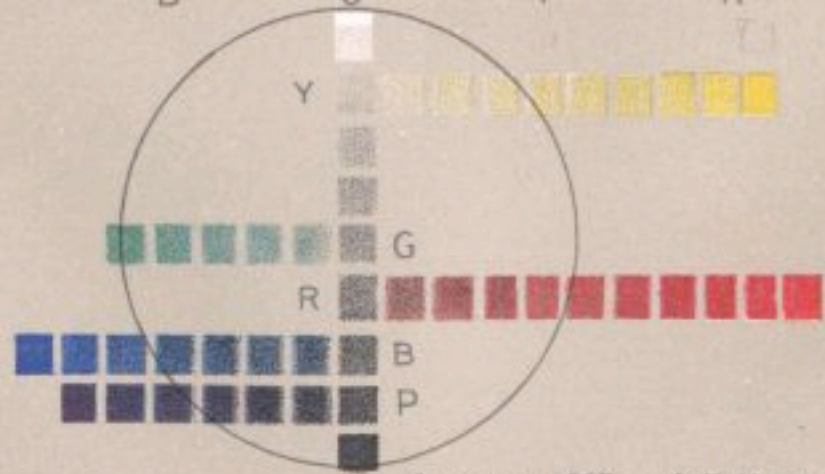
Brain does quite a lot of post-processing, so hard to directly perceive amount of red, green and blue.

A more useful colour space is hcl: hue, chroma (intensity) and luminance (lightness)

PLATE I.



Light Values				
Middle Values				
Dark Values				
P	B	G	Y	R



**HCL in 3d**

# Colour scales

**Discrete:** evenly spaced hues of equal chroma and luminance. No colour appears more important than any other. Does not imply order.

**Continuous:** evenly spaced hues between two colours.

Munsell package to makes it easy to select matched colours.

```
library(munsell)
```

```
hue_slice()
```

```
hue_slice("5R")
```

```
chroma_slice("6")
```

```
value_slice("7")
```

# Color brewer

<http://colorbrewer2.org/>

Cynthia Brewer applied many of these principles to come up with a selection of good palettes (particularly tailored for maps).

Use `cut_interval()` or `cut_number()` to convert continuous to categorical



```
vals <- seq(-4 * pi, 4 * pi, len = 50)
df <- expand.grid(x = vals, y = vals)
df$r <- with(df, sqrt(x ^ 2 + y ^ 2))
df$z <- with(df, cos(r ^ 2) * exp(- r / 6))
df$z_cut <- cut_interval(df$z, 9)

(p1 <- qplot(x, y, data = df, fill = z,
  geom = "tile"))
(p2 <- qplot(x, y, data = df, fill = z_cut,
  geom = "tile"))
```

```
p1 + scale_fill_gradient(low = "white",
  high = "black")

# Highlight deviations
p1 + scale_fill_gradient2()
p1 + scale_fill_gradient2(breaks = seq(-1, 1,
  by = 0.25), limits = c(-1, 1))
p1 + scale_fill_gradient2(mid = "white",
  low = "black", high = "black")

p2 + scale_fill_brewer(pal = "Blues")
```

# Colour blindness

7-10% of men are red-green colour “blind”. (Many other rarer types of colour blindness)

Solutions: avoid red-green contrasts; use redundant mappings; **test**. I like color oracle: <http://colororacle.cartography.ch>

# Your turn

Look up a diverging colorbrewer scale and use that instead.

Use <http://www.vischeck.com/vischeck/> to check the colour schemes we've been using.

# Other resources

A. Zeileis, K. Hornik, and P. Murrell.  
Escaping RGBland: Selecting colors for  
statistical graphics. Computational  
Statistics & Data Analysis, 2008.

[http://statmath.wu-wien.ac.at/~zeileis/papers/  
Zeileis+Hornik+Murrell-2008.pdf](http://statmath.wu-wien.ac.at/~zeileis/papers/Zeileis+Hornik+Murrell-2008.pdf).

# Labels & ticks

# Overview

All display aspect of legends and axes controlled by the scales.

To modify axes & legend titles, tick marks and legend keys, you need to modify parameters of the scales: **name**, **breaks**, **labels**.

```
# Labels
scale_x_continuous("My new x")
# All techniques in plotmath also work
scale_x_continuous(expression(x ^ alpha))
# Similarly for colour
scale_colour_discrete("Colour scale")

# Short cuts:
xlab("My new x")
xlab(expression(x ^ alpha))
labs(colour = "Colour scale")
```



```
qplot(carat, price, data = diamonds)
qplot(carat, price, data = diamonds) +
  scale_x_log10() +
  scale_y_log10()
```

```
prices <- c(100, 500, 1000, 5000, 10000)
last_plot() +
  scale_y_log10(breaks = prices)
```

```
last_plot() +
  scale_y_log10(breaks = prices, labels = prices)
```

# Your turn

Also improve the scale for the x axis.

Use `geom = "hexbin"` and adjust the keys and labels on the colour legend.

# Themes

# Visual appearance

So far have only discussed how to get the data displayed the way you want, focussing on the essence of the plot.

Themes give you a huge amount of control over the appearance of the plot, the choice of background colours, fonts and so on.

```
# Two built in themes. The default:  
qplot(carat, price, data = diamonds)  
  
# And a theme with a white background:  
qplot(carat, price, data = diamonds) + theme_bw()  
  
# Use theme_set if you want it to apply to every  
# future plot.  
theme_set(theme_bw())  
  
theme_bw()  
theme_grey()
```

# Elements

You can also make your own theme, or modify an existing one.

Themes are made up of elements which can be one of: `theme_line`, `theme_segment`, `theme_text`, `theme_rect`, `theme_blank`

Gives you a lot of control over plot appearance.

# Elements

**Axis:** axis.line, axis.text.x, axis.text.y,  
axis.ticks, axis.title.x, axis.title.y

**Legend:** legend.background, legend.key,  
legend.text, legend.title

**Panel:** panel.background, panel.border,  
panel.grid.major, panel.grid.minor

**Strip:** strip.background, strip.text.x,  
strip.text.y

```
p <- qplot(displ, hwy, data = mpg) +  
  opts(title = "Bigger engines are less efficient")
```

```
# To modify a plot
```

```
p
```

```
p + opts(plot.title =  
  theme_text(size = 12, face = "bold"))
```

```
p + opts(plot.title = theme_text(colour = "red"))
```

```
p + opts(plot.title = theme_text(angle = 45))
```

```
p + opts(plot.title = theme_text(hjust = 1))
```



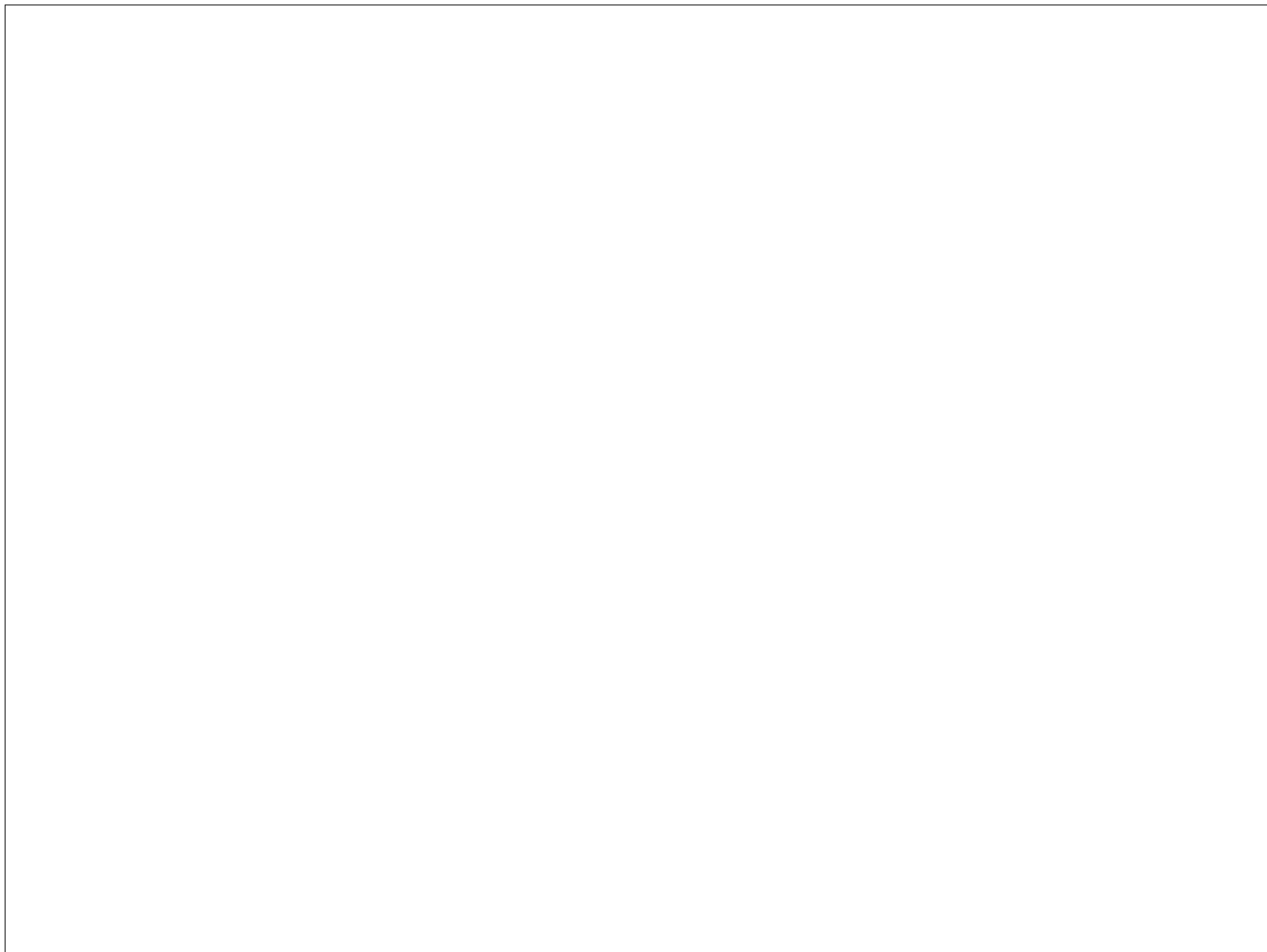
# Your turn

Fix the overlapping y labels on this plot:

```
qplot(reorder(model, hwy), hwy, data =  
mpg)
```

Rotate the labels on these strips so they are easier to read.

```
qplot(hwy, reorder(model, hwy), data =  
mpg) + facet_grid(manufacturer ~ .,  
scales = "free", space = "free")
```



This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.