Polishing your plots

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

Department of Statistics / Rice University

February 2010

Communication graphics

When you need to **communicate** your findings, you need to spend a lot of time polishing your graphics to eliminate distractions and focus on the story.

Now it’s time to pay attention to the small stuff: labels, colour choices, tick marks...

1. **Scales**: used to override default perceptual mappings, and tune parameters of axes and legends.

2. **Themes**: control presentation of non-data elements.

3. **Saving your work**: to include in reports, presentations, etc.

Scales

Scales

Control how data is mapped to perceptual properties, and produce **guides** (axes and legends) which allow us to read the plot.

Important parameters: **name**, **breaks** & **labels**, **limits**.

Naming scheme: scale_*aesthetic_name*.
All default scales have name continuous or discrete.

```
# Default scales
scale_x_continuous()
scale_y_discrete()
scale_colour_discrete()

# Custom scales
scale_colour_hue()
scale_x_log10()
scale_fill_brewer()

# Scales with parameters
scale_x_continuous("X Label", limits = c(1, 10))
scale_colour_gradient(low = "blue", high = "red")
```

```
p <- qplot(cyl, displ, data = mpg)

# First argument (name) controls axis label
p + scale_y_continuous("Displacement (l)")
p + scale_x_continuous("Cylinders")

# Breaks and labels control tick marks
p + scale_x_continuous(breaks = c(4, 6, 8))
p + scale_x_continuous(breaks = c(4, 6, 8),
  labels = c("small", "medium", "big"))

# Limits control range of data
p + scale_y_continuous(limits = c(1, 8))
# same as:
p + ylim(1, 8)
```

Your turn

```
qplot(carat, price, data = diamonds, geom = "hex")
```

Manipulate the fill colour legend to:

- Change the title to “Count"
- Display breaks at 1000, 3500 & 7000
- Add commas to the keys (e.g. 1,000)
- Set the limit for the scale from 0 to 8000.

```
p <- qplot(carat, price, data = diamonds, geom = "hex")

# First argument (name) controls legend title
p + scale_fill_continuous("Count")

# Breaks and labels control legend keys
p + scale_fill_continuous(breaks = c(1000, 3500, 7000))
p + scale_fill_continuous(breaks = c(0, 4000, 8000))

# Why don't 0 and 8000 have colours?
p + scale_fill_continuous(breaks = c(0, 4000, 8000),
  limits = c(0, 8000))

# Can use labels to make more human readable
breaks <- c(0, 2000, 4000, 6000, 8000)
labels <- format(breaks, big.mark = ",")
p + scale_fill_continuous(breaks = breaks, labels = labels,
   limits = c(0, 8000))
```

```
p <- qplot(color, carat, data = diamonds)

# Basically the same for discrete variables
p + scale_x_discrete("Color")

# Except limits is now a character vector
p + scale_x_discrete(limits = c("D", "E", "F"))

# Should work for boxplots too
qplot(color, carat, data = diamonds,
  geom = "boxplot") +
  scale_x_discrete(limits = c("D", "E", "F"))
```

Alternate scales

Can also override the default choice of scales. You are most likely to want to do this with **colour**, as it is the most important aesthetic after position.

Need a little background to be able to use colour effectively: colour **spaces** & colour **blindness**.

Colour spaces

Most familiar is **rgb**: defines colour as mixture of **red**, **green** and **blue**. Matches the physics of eye, but the brain does a lot of post-processing, so it's hard to directly perceive these components.

A more useful colour space is hcl: **hue**, **chroma** and **luminance**

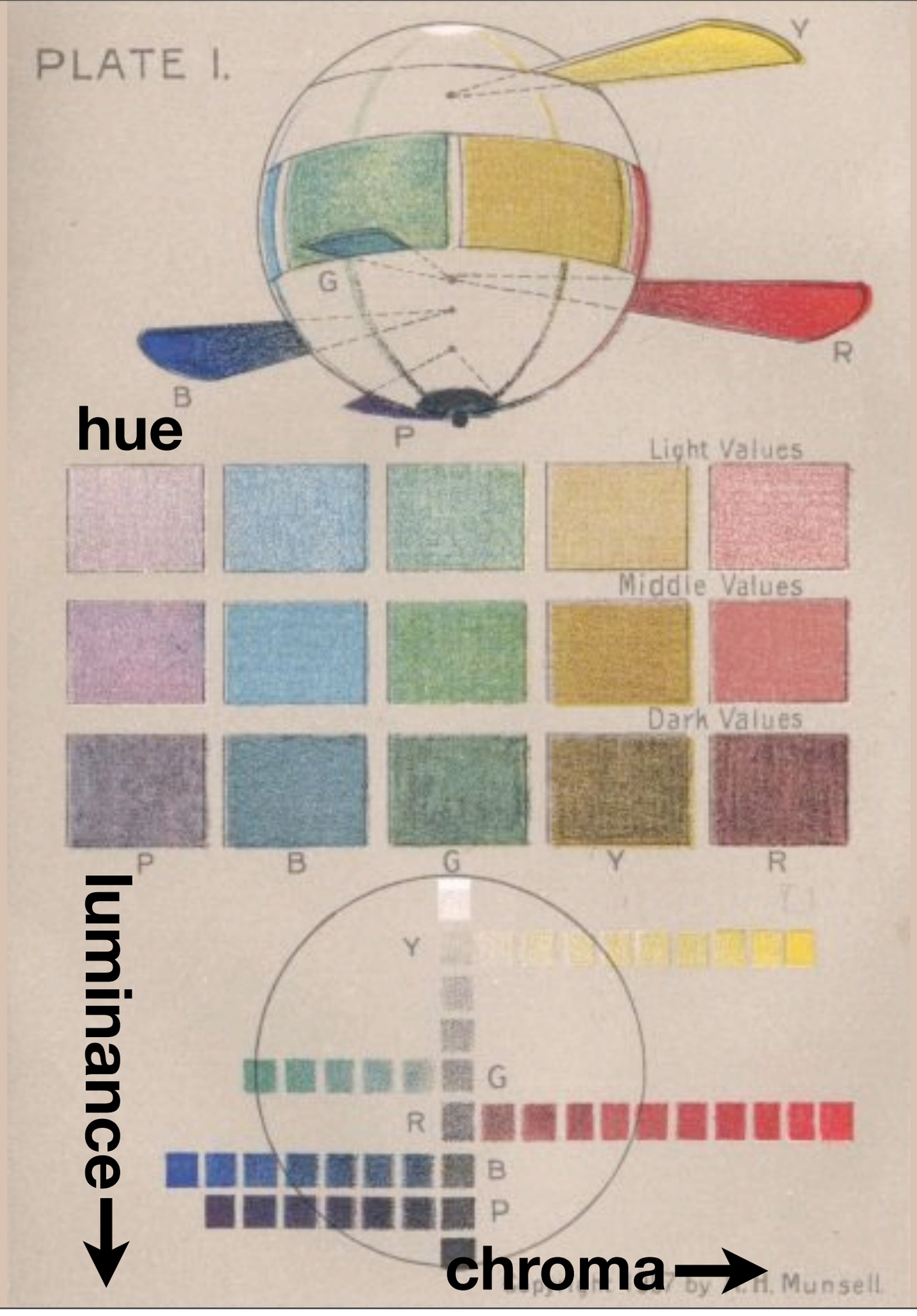
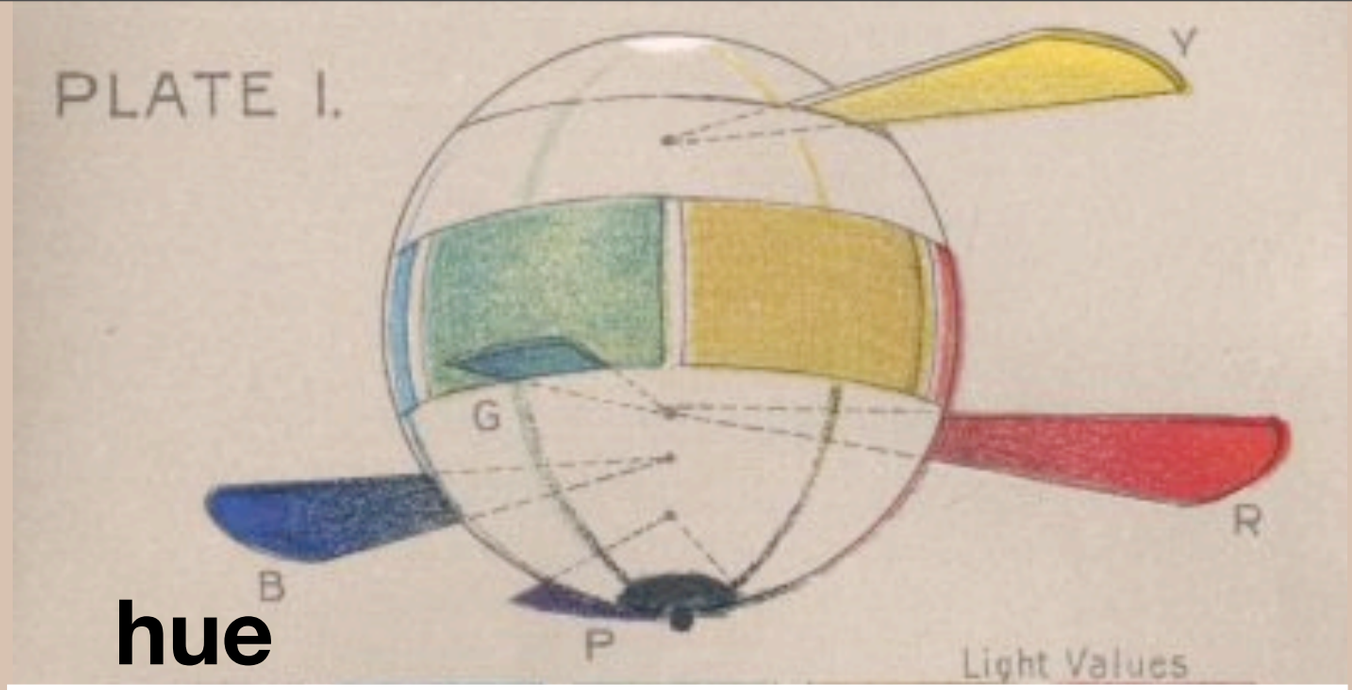
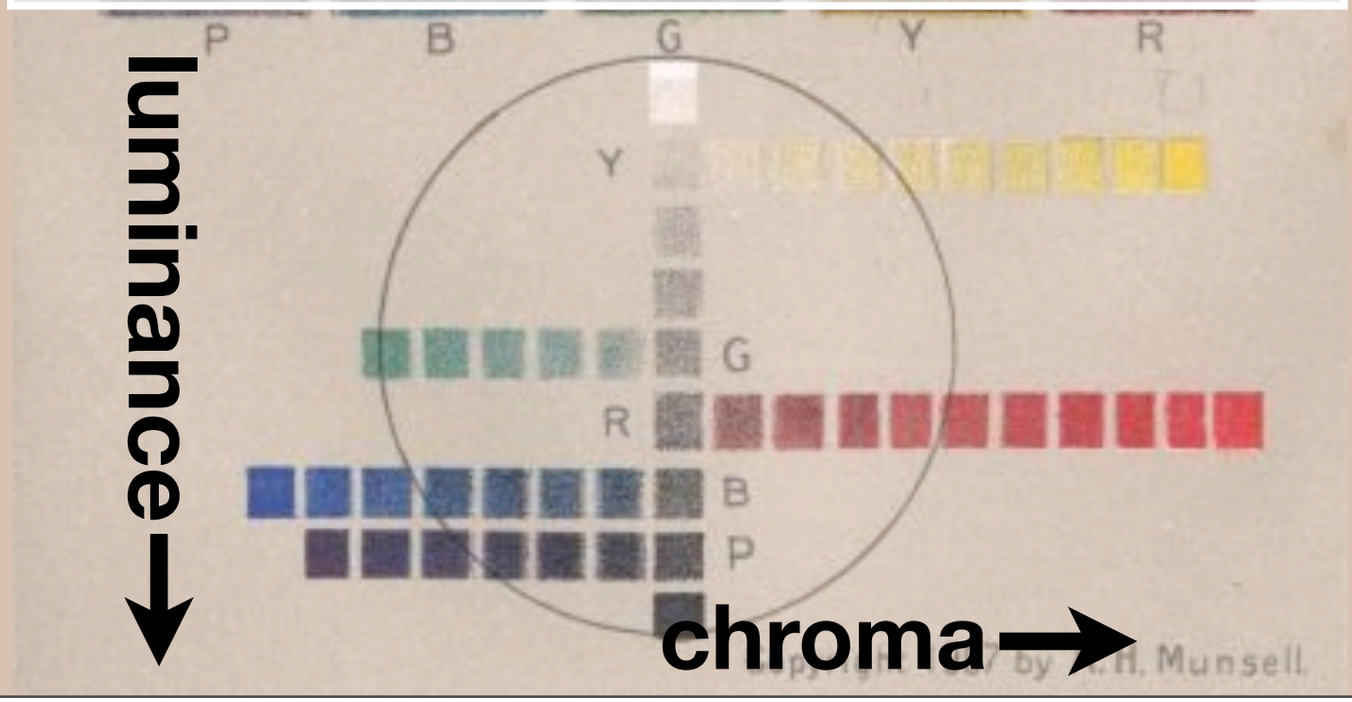


PLATE I.
Y
G
B
R
P
hue
Light Values
Middle Values
Dark Values
P
B
G
Y
R
luminance
Y
G
R
B
P
chroma
by A. H. Munsell



hue

Demo of real shape in 3d



Default 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.

Alternatives

Discrete: brewer, grey

Continuous: gradient2, gradientn

Color brewer

Cynthia Brewer applied basics principles and then rigorously tested to produce selection of good palettes, particularly tailored for maps: http://colorbrewer2.org/

Can use `cut_interval()` or `cut_number()` to convert continuous to discrete.

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

Read through the examples for `scale_colour_brewer`, `scale_colour_gradient2` and `scale_colour_gradientn`.

Experiment!

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())

# This is the best way of seeing all the default
# options
theme_bw()
theme_grey()
```

Elements

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

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")
```

Saving your work

```
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
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.

Friday, 12 February 2010