🎶 Don’t cha wish your ggplot had colours like me? 🎶

ISSC Mini-Tutorial

Liza Bolton

The University of Toronto

2020-05-12

🎶 The Pussycat Dolls - Don't Cha ft. Busta Rhymes

Image credit: Giphy

Colour matters!

Why?

Colours can:

- influence emotion, which can influence how people understand your data.
- connect to meaning. (Hot and cold? Red and blue.)
- show **more** or less with intensity/brightness.

Good discussion in this interview with Maureen Stone, a colour expert with Tableau.

What's this talk for?

- tips about what to think about when *picking* palettes
- showing some cool/useful palettes

Assumptions I'm making

- you are somewhat familiar with `ggplot` and R in general
- you don't need heaps of technical info on hex codes and colour theory for this to be fun and useful

How do you tell computers about colours?

You'll usually see a colour written in one of 4 ways:

- Just as a **colour name**, like "red". Has to be one R recognises - list here
- As a mix of **hue, saturation and lightness** (HSL)
- As a mix of **redness, greenness and blueness** (RGB)
- As 6 numbers/letters with a # at the front eg. #ff0000 (**Hex code**)

Useful colour tool

Chrome extension for picking colours from webpages

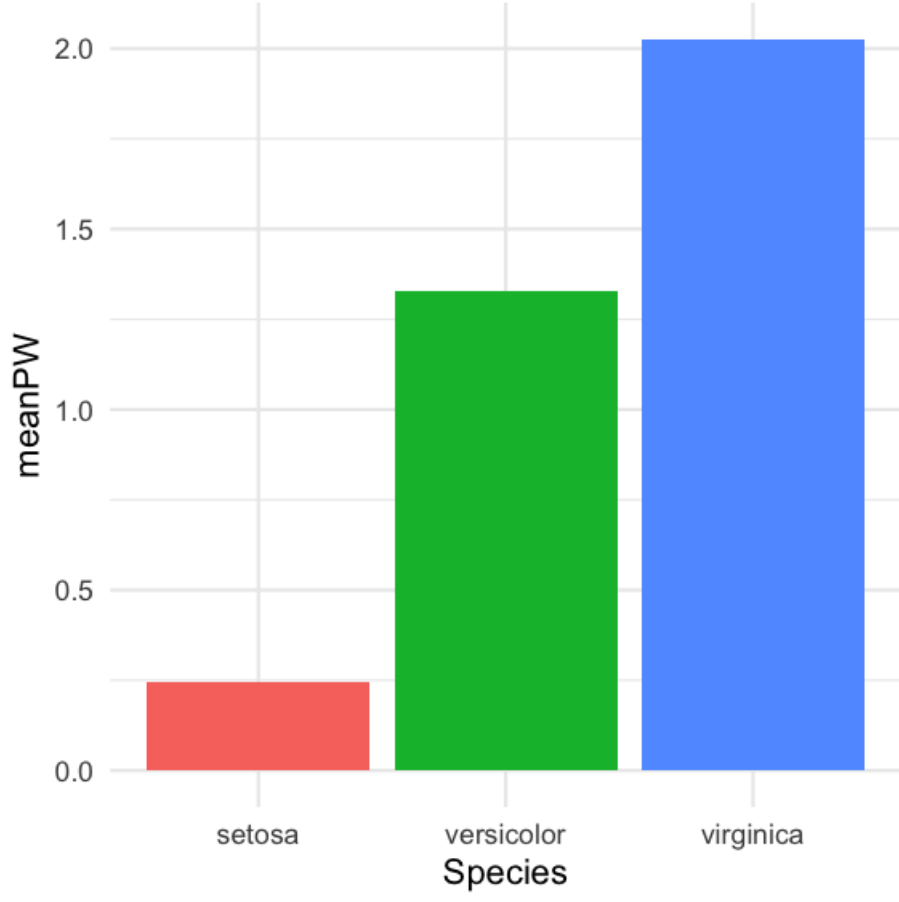
What do you need your colours to do?

Just look different from each other?

You want a **qualitative palette**.

Qualitative palette use case

```
# Use the iris dataset
data("iris")

# Create a simple barchart
# of mean petal width
bars <- iris %>%
  group_by(Species) %>%
  summarise(meanPW = mean(Petal.Width)) %>%
  ggplot(aes(x = Species,
             y = meanPW,
             fill = Species)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  guides(fill = FALSE)

bars
```

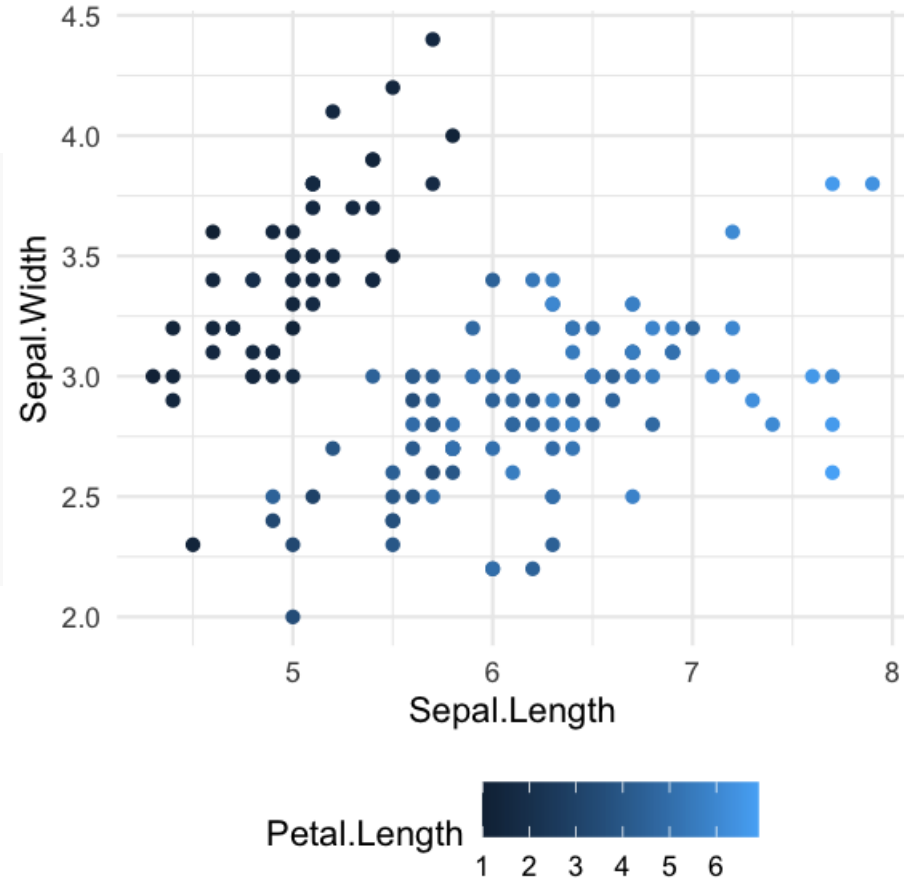


Show a range from more to less?

You want a **sequential palette**.

Sequential palette use case

```
# Use the iris dataset
data("iris")

# Make a scatterplot
# points coloured by petal length
petals <- iris %>%
  ggplot(aes(x = Sepal.Length,
             y = Sepal.Width,
             color = Petal.Length)) +
  geom_point() +
  theme_minimal() +
  theme(legend.position="bottom")

petals
```

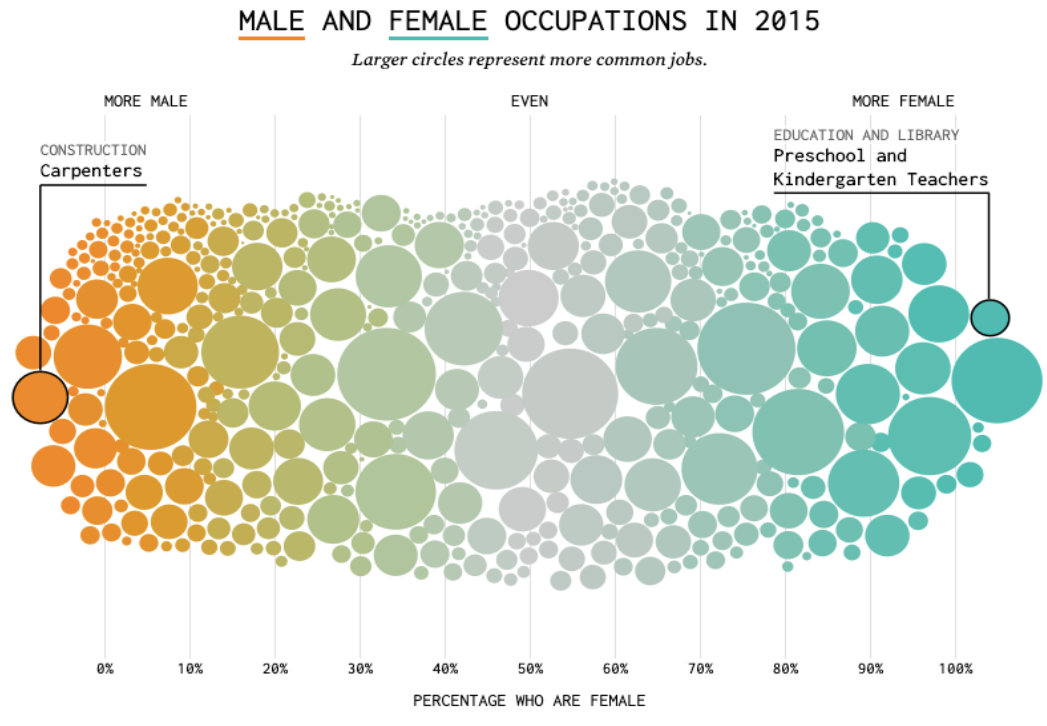


Contrast around a specific point?

You want a **diverging palette**.

Thanks to this Everyday analytics article for helping my clarify my thinking on this one.

Diverging palette use case



Source: Nathan Yau, Flowing Data

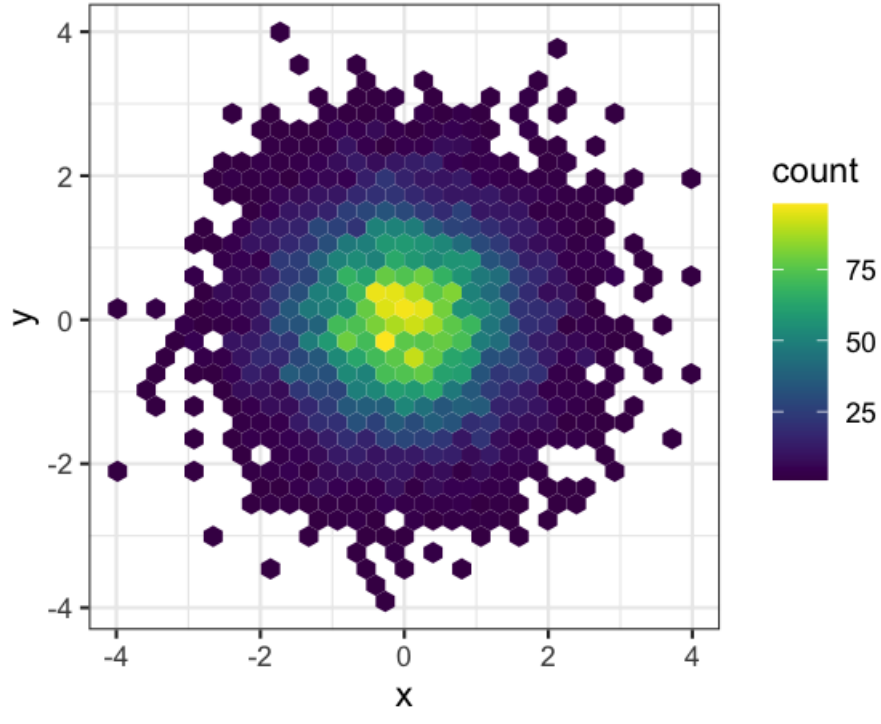
Palette time!

Viridis

Viridis

Perceptually uniform! Gorgeous for continuous data.

```
library(viridis)
library(hexbin)

ggplot(data.frame(x = rnorm(10000),
                  y = rnorm(10000)),
       aes(x = x, y = y)) +
  geom_hex() +
  coord_fixed() +
  scale_fill_viridis() +
  theme_bw()
```



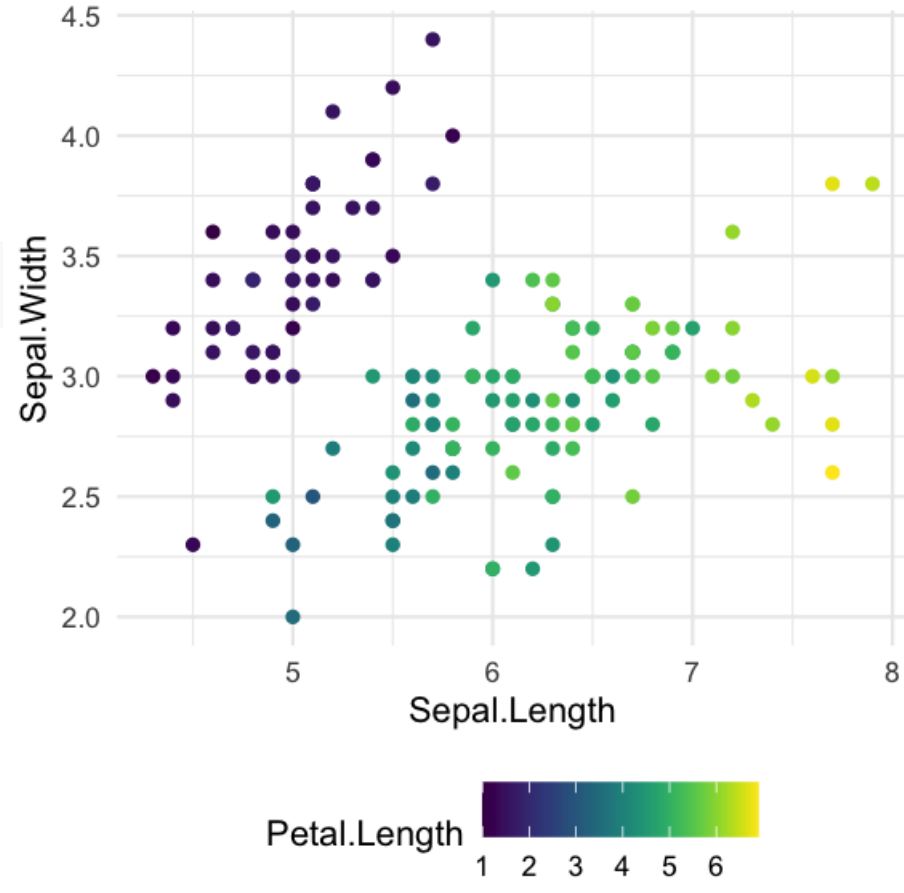
Viridis

Viridis

Viridis

Viridis

```
petals +
  scale_color_viridis()
```

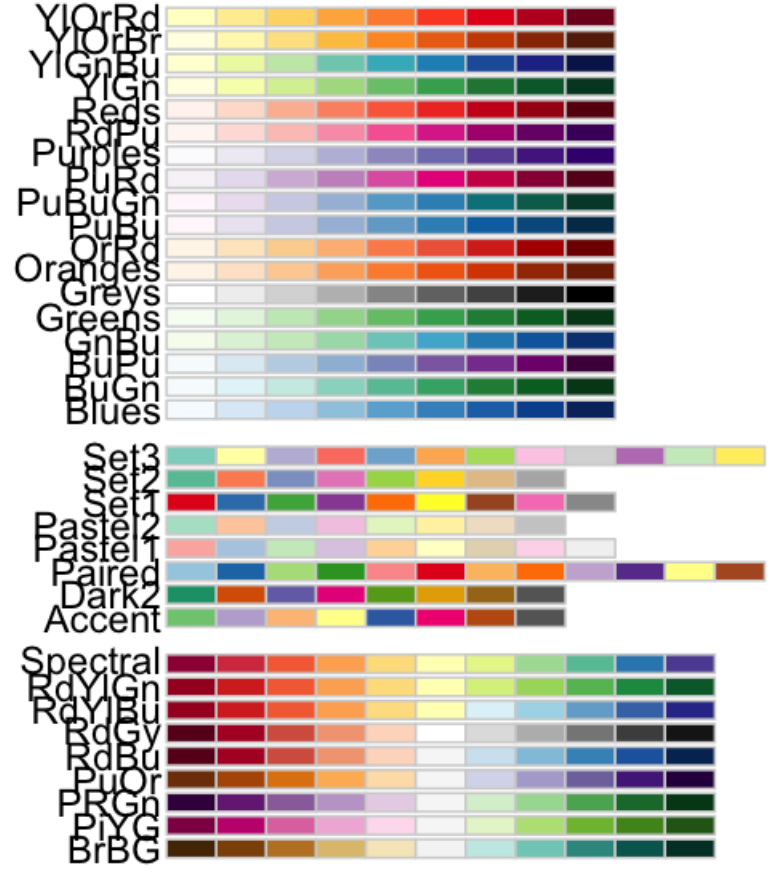


Color Brewer

RColorBrewer

```
# Display all the palettes
library(RColorBrewer)
par(mar=c(0,4,0,2))
display.brewer.all()
```

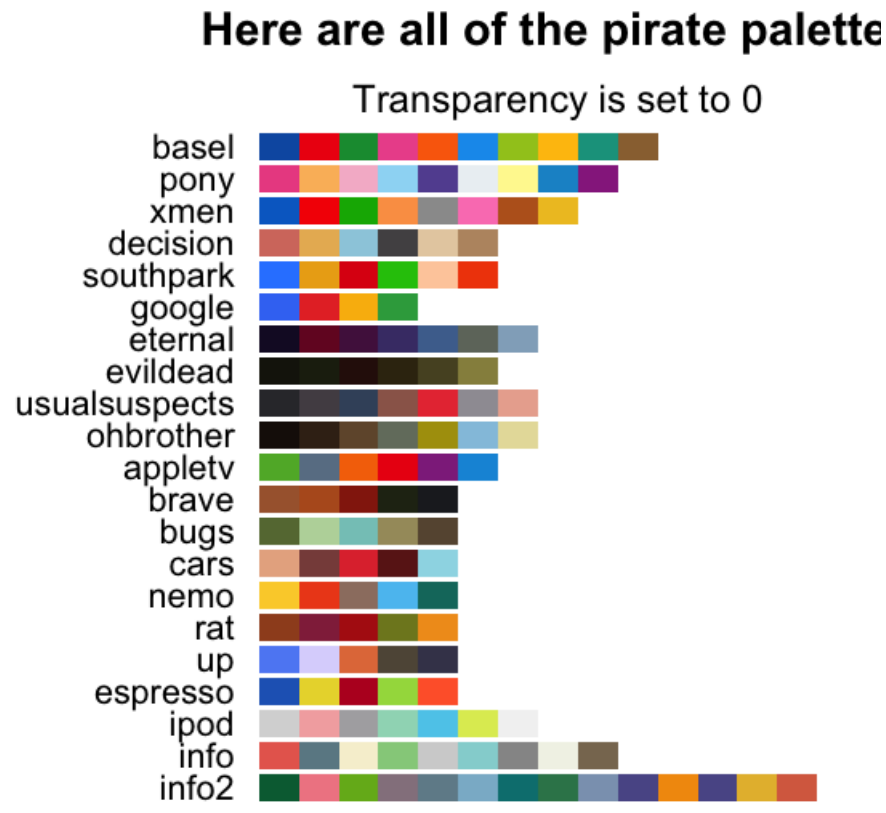
```
# Notice
# - sequential
# - qualitative
# - diverging
```



Pirate palette

yarrr

```
yarrr::piratepal(palette = "all")
```

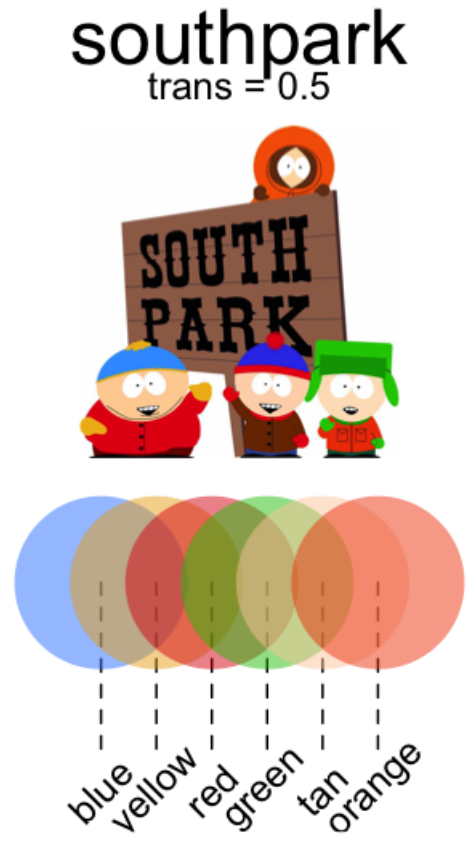


Pirate palette

yarrr

```
yarrr::piratepal(
          palette = "southpark",
          trans = .5,
          plot.result = T)
```

Pirate pal vignette



ochRe

ochRe 🎩 Di Cook @visnut

ochRe

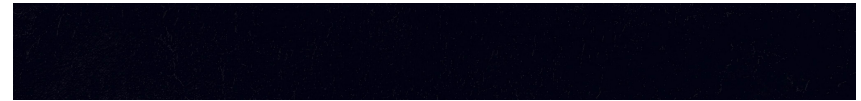
ochRe 🎩 Di Cook @visnut

```
# devtools::install_github("ropenscilabs/ochRe")
library(ochRe)

pal_names <- names(ochre_palettes)

par(mfrow=c(length(ochre_palettes)/2, 2),
    lheight = 2, mar=rep(1, 4), adj = 0)
for (i in 1:length(ochre_palettes)){
    viz_palette(ochre_palettes[[i]],
                pal_names[i])
}
```

Dutchmasters

dutchmasters

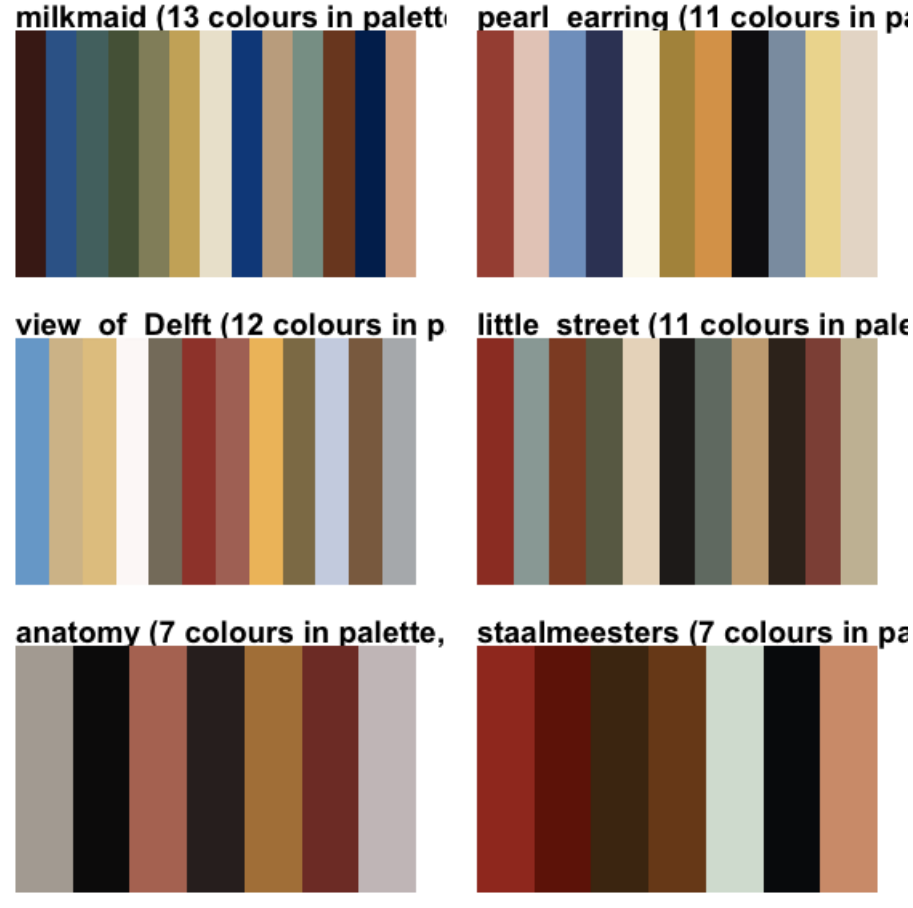
By Johannes Vermeer - Copied from Mauritshuis website, resampled and uploaded by Crisco 1492 (talk · contribs), October 2014, Public Domain, Link



Dutchmasters

dutchmasters

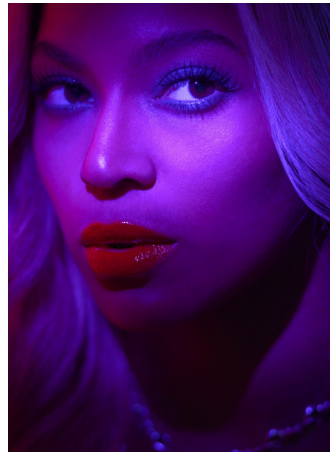
```
# devtools::install_github("EdwinTh/dutchmasters
library(dutchmasters)

pal_names <- names(dutchmasters)

par(mfrow=c(length(dutchmasters)/2, 2),
    lheight = 2, mar=rep(1, 4), adj = 0)
for (i in 1:length(dutchmasters)){
    viz_palette(dutchmasters[[i]], pal_names[i])
}
```

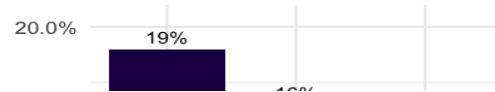


Beyonce

📦 beyonce 👑🐝

```
#install.packages("devtools")
#devtools::install_github("dill/beyonce")
#library(tidyverse)
library(beyonce)
```

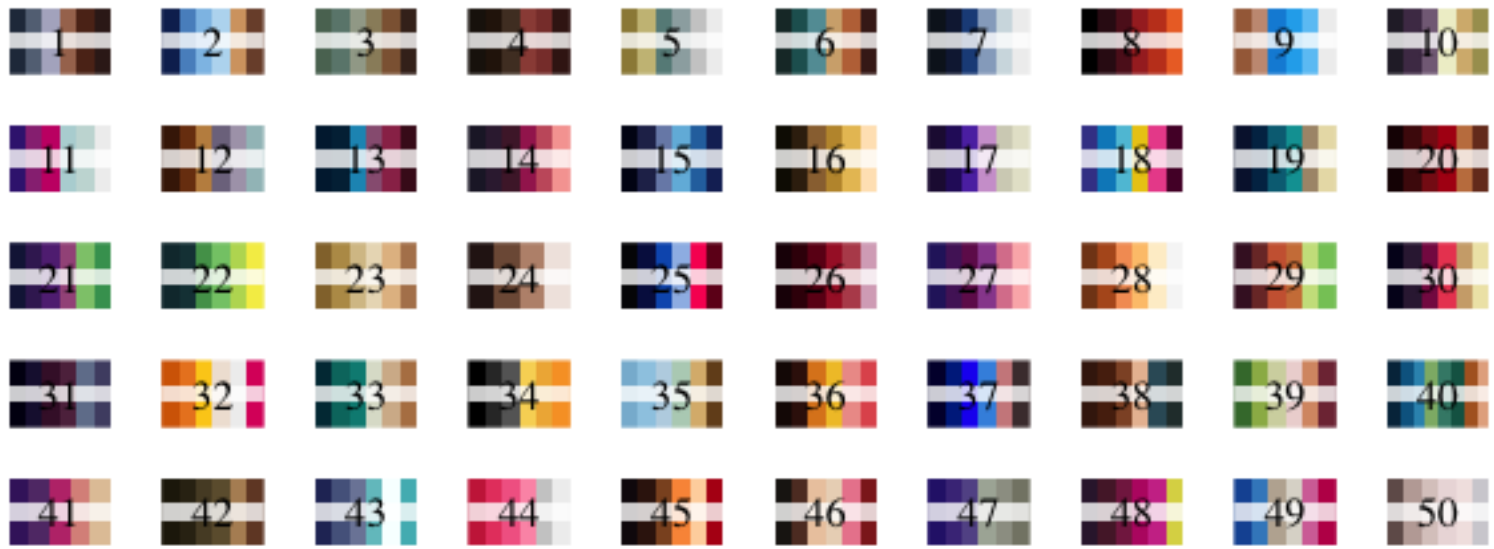




Beyonce cont.

There are lots of other colour palettes, all adapted from https://beyoncepalettes.tumblr.com/.

```
par(mfrow=c(13,10))
for(i in 1:50) print(beyonce_palette(i)) # just the first 50
```



Ghibli

ghibli 🎩 Jacquie Tran @jacquietran

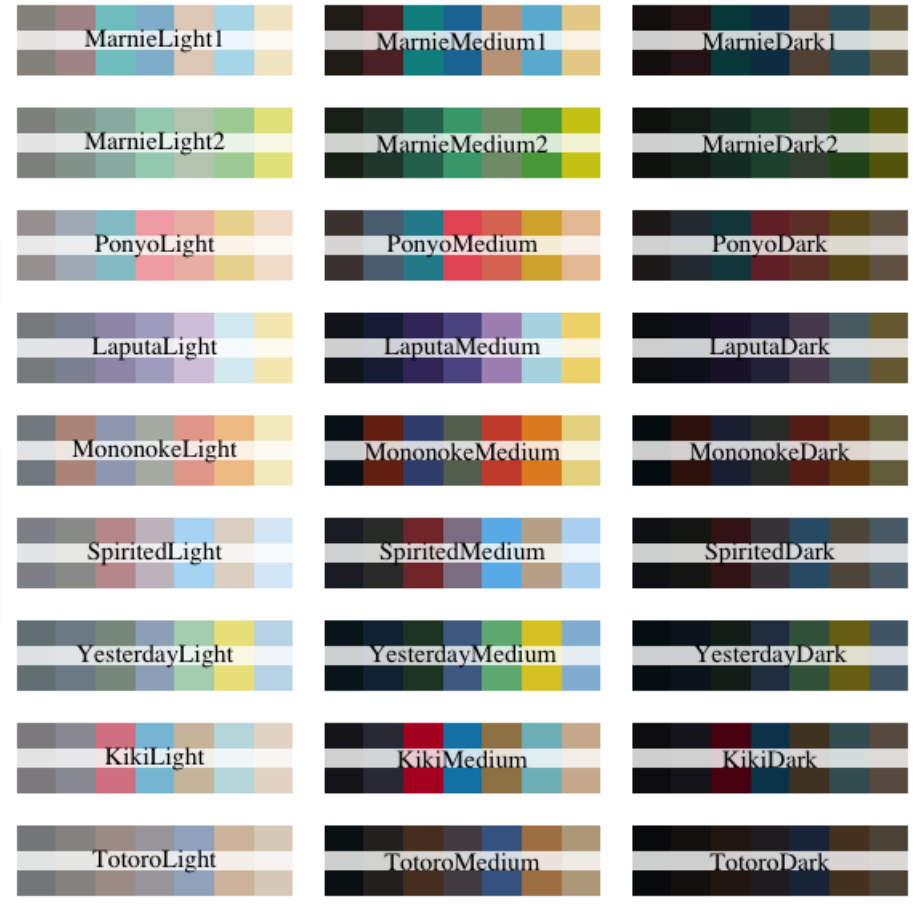
Ghibli

ghibli 🎩 Jacquie Tran @jacquietran

```
library(ghibli)
```

```
## Registered S3 method overwritten by 'ghibli':
##   method        from
##   print.palette beyonce
```

```
# Display palettes with names
par(mfrow=c(9,3))
for(i in names(ghibli_palettes)) {
  print(ghibli_palette(i))
}
```



Wes Anderson

wesanderson

```
library(wesanderson)
```

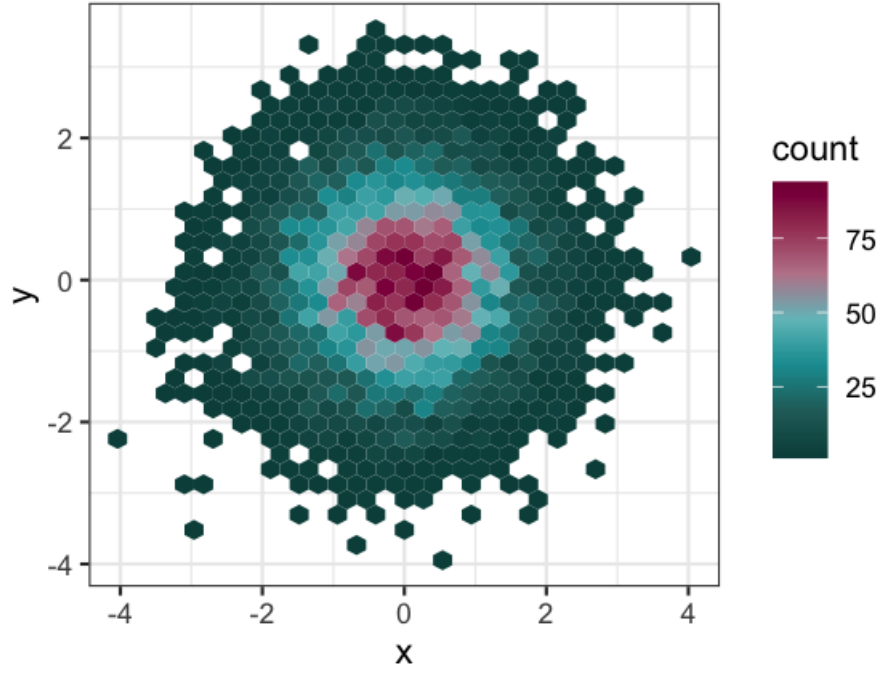
```
## Registered S3 method overwritten by 'wesanders
##   method        from
##   print.palette ghibli
```

```
# Display palettes with names
par(mfrow=c(10,2))
for(i in names(wes_palettes)) {
  print(wes_palette(i))
}
```

Harry Potter

harrypotter 🎩 Anna Fergusson @annafergusson

```
library(harrypotter)

ggplot(data.frame(x = rnorm(1e4),
                  y = rnorm(1e4)),
       aes(x = x, y = y)) +
  geom_hex() +
    coord_fixed() +
  scale_fill_hp(house = "LunaLovegood") +
    theme_bw()
```



Harry Potter

harrypotter 🎩 Anna Fergusson @annafergusson

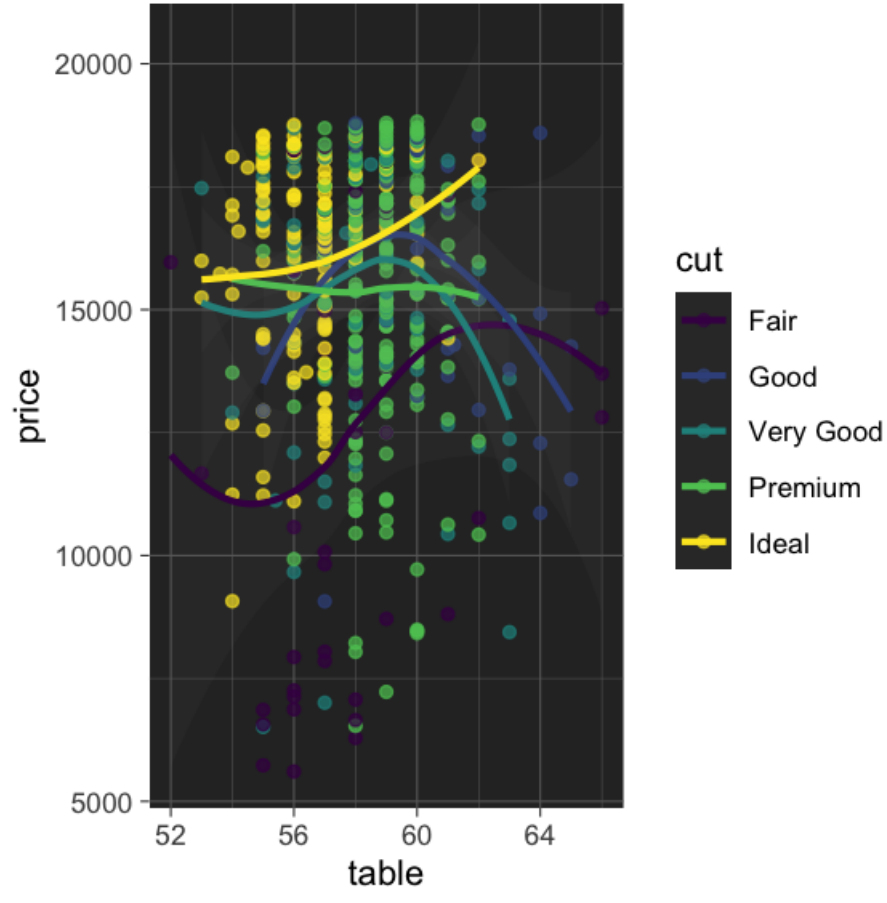
ggsci

ggsci

A mixture of academic journal palettes and pop culture references.

```
library(ggsci)

ggplot(subset(diamonds, carat >= 2.2),
       aes(x = table, y = price, colour = cut)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "loess",
              alpha = 0.05, size = 1, span = 1) +
  theme_dark() + theme(
    panel.background =
      element_rect(fill = "#2D2D2D"),
    legend.key =
      element_rect(fill = "#2D2D2D")) +
  scale_fill_tron()
```

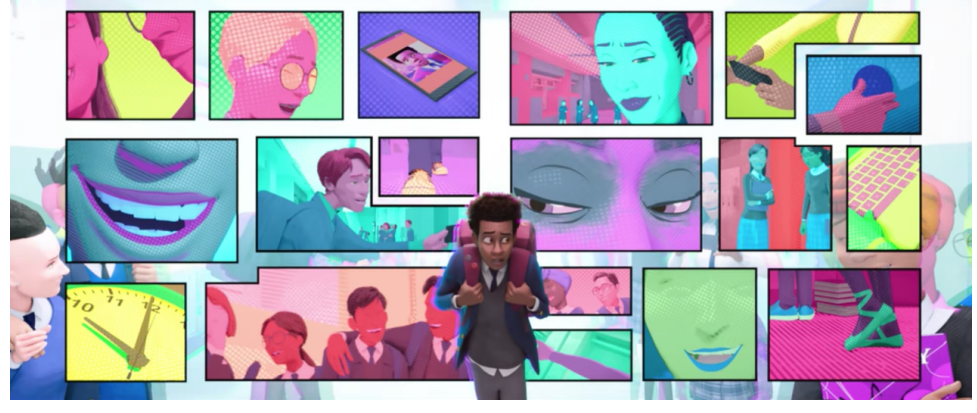
```
## `geom_smooth()` using formula 'y ~ x'
```



What about making your own palette?

Setting up your own palette: Pick your colours

I love this screenshot from *Into the Spiderverse* and used [this free palette generator] to lazily pull colours from it.



Turquoise	Khaki	Dark Slate Blue	Old Rose	Cadet Blue
#30E6C9	#D7EA85	#355E78	#A83B80	#77A5BF

Setting up your own palette: Load your colours

For a more complete discussion see this great article by Dr Simon Jackson @drsimonj.

```
# Set up the colours you want
my_colours <- c(
  `turquoise`  = "#30E6C9",
  `khaki`      = "#D7EA85",
  `slate`      = "#355E78",
  `old-rose`   = "#A83B80",
  `cadet-blue` = "#77A5BF",
  `light grey` = "#cccccc", # added extra neutrals
  `dark grey`  = "#8c8c8c")

#Function that converts names to hexcodes
get_spiderverse_col <- function(...) {
  cols <- c(...)

  if (is.null(cols)) return (my_colours)

  my_colours[cols]
}

# You could just use this function:
ggplot(mtcars, aes(hp, mpg)) +
    geom_point(color = get_spiderverse_col("old-ro
               size = 4, alpha = .8) +
  theme_minimal()
```

Setting up your own palette: Group colours into palettes

```
spiderverse_palettes <- list(
  `main`  = get_spiderverse_col("turquoise", "khaki", "old-rose"),

  `blues`  = get_spiderverse_col("cadet-blue", "slate"),

  `full` = get_spiderverse_col("turquoise", "khaki", "slate", "old-rose", "cadet-blue"),

  `neutral`  = get_spiderverse_col("slate", "light grey", "cadet-blue", "dark grey")
)

# Helpful function for retrieving the palettes you just names
spiderverse_pal <- function(palette = "main", reverse = FALSE, ...) {
  pal <- spiderverse_palettes[[palette]]
  if (reverse) pal <- rev(pal)
  colorRampPalette(pal, ...)
}
```

Setting up your own palette: Set up scales for ggplot

```
# Colour scale - good for scatter plots
scale_color_spiderverse <- function(palette = "main", discrete = TRUE, reverse = FALSE, ...) {
  pal <- spiderverse_pal(palette = palette, reverse = reverse)

  if (discrete) {
    discrete_scale("colour", paste0("spiderverse_", palette), palette = pal, ...)
  } else {
    scale_color_gradientn(colours = pal(256), ...)
  }
}

# Fill scale - good for barplots
scale_fill_spiderverse <- function(palette = "main", discrete = TRUE, reverse = FALSE, ...) {
  pal <- spiderverse_pal(palette = palette, reverse = reverse)

  if (discrete) {
    discrete_scale("fill", paste0("spiderverse_", palette), palette = pal, ...)
  } else {
    scale_fill_gradientn(colours = pal(256), ...)
  }
}
```

How did we do?

```
ggplot(iris, aes(Sepal.Width, Sepal.Length, color
    geom_point(size = 4) +
    scale_color_spiderverse() +
    theme_minimal()
```

```
bey +
    scale_fill_spiderverse("neutral") +
    theme_minimal()
```

Accessibility resources

- How to design for colour blindness with a great link to a colour blindness simulator
- `dichromat` - Collapse red-green or green-blue distinctions to simulate the effects of different types of color-blindness.

Other resources and lists of resources

- Paletti, make your own palettes - from the guy behind the Dutchmasters palette
- Good compilation of resources here
- Datawrapper blog with resources too, including Art Palette Experiment by Google, Movies
- `colorspace` - The Colour Space package has lots of great information and palette options.
- Greyscale functionality in ggplot is pretty good
- Datanovia list of R palettes to know
- This Color Palette Generator by Steven DeGraeve can help make colour palette from an image
- Things to think about - which colours do you put next to each other? Visual blurring is not good

Thanks!

Slides created via the R package **xaringan**.

The chakra comes from remark.js, **knitr**, and R Markdown.