

Data Visualisation with R

Workshop Part 1



3

Scales



Diamonds data

The diamonds data is part of ggplot2 📦

```
glimpse(diamonds)
```

```
## Rows: 53,940
## Columns: 10
## $ carat    <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0...
## $ cut       <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver...
## $ color     <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I, ...
## $ clarity   <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ...
## $ depth     <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64...
## $ table     <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58...
## $ price     <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34...
## $ x         <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4...
## $ y         <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4...
## $ z         <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2...
```

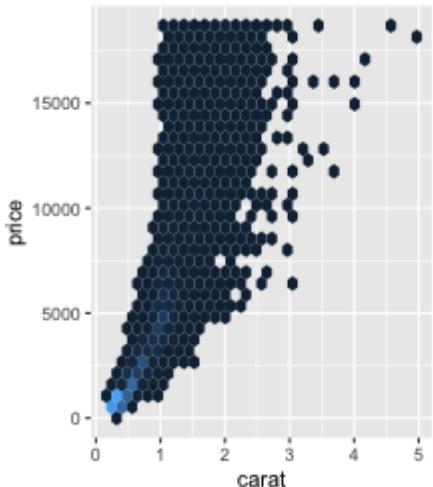
Scales

- Scales control the mapping from **data** to **aesthetics**.

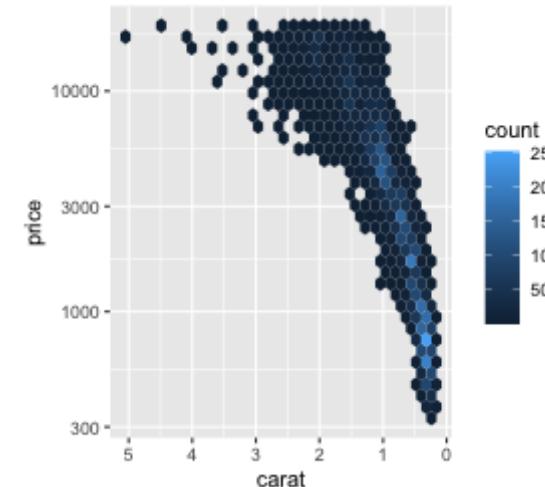
`scale_<aesthetic>_<type>`

```
g <- ggplot(diamonds, aes(carat, price) ) + geom_hex()
```

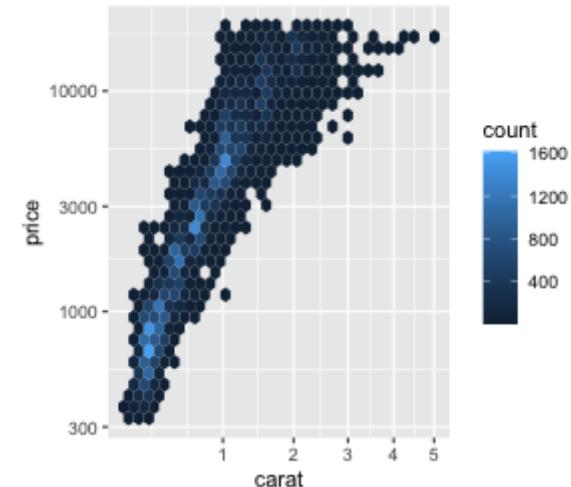
```
g + scale_y_continuous() +  
  scale_x_continuous()
```



```
g + scale_x_reverse() +  
  scale_y_continuous(trans="log10")
```



```
g + scale_y_log10() +  
  scale_x_sqrt()
```



scale

scales	Description
scale_alpha, scale_alpha_continuous, scale_alpha_binned, scale_alpha_discrete, scale_alpha_ordinal, scale_alpha_datetime, scale_alpha_date	Alpha transparency scales
scale_x_binned, scale_y_binned	Positional scales for binning continuous data (x & y)
scale_colour_brewer, scale_fill_brewer, scale_colour_distiller, scale_fill_distiller, scale_colour_fermenter, scale_fill_fermenter, scale_color_brewer, scale_color_distiller, scale_color_fermenter	Sequential, diverging and qualitative colour scales from ColorBrewer
scale_colour_continuous, scale_fill_continuous, scale_colour_binned, scale_fill_binned, scale_color_continuous, scale_color_binned	Continuous and binned colour scales

Previous

1

2

3

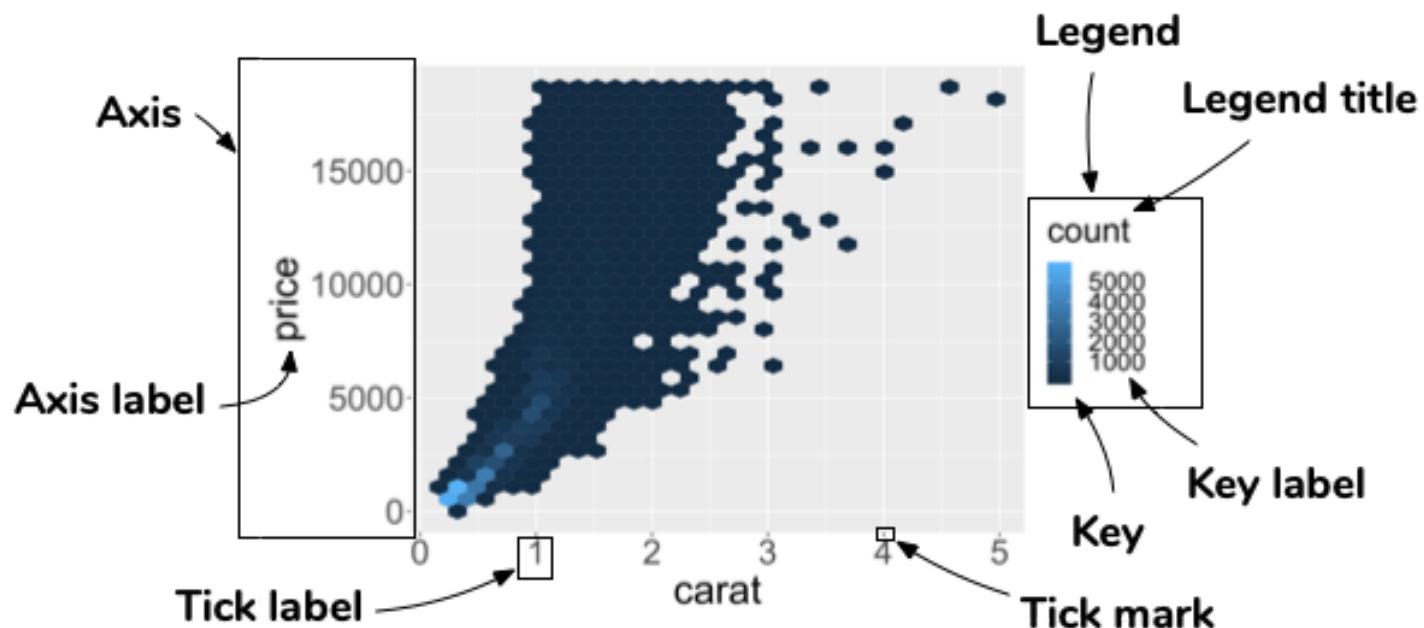
4

5

Next

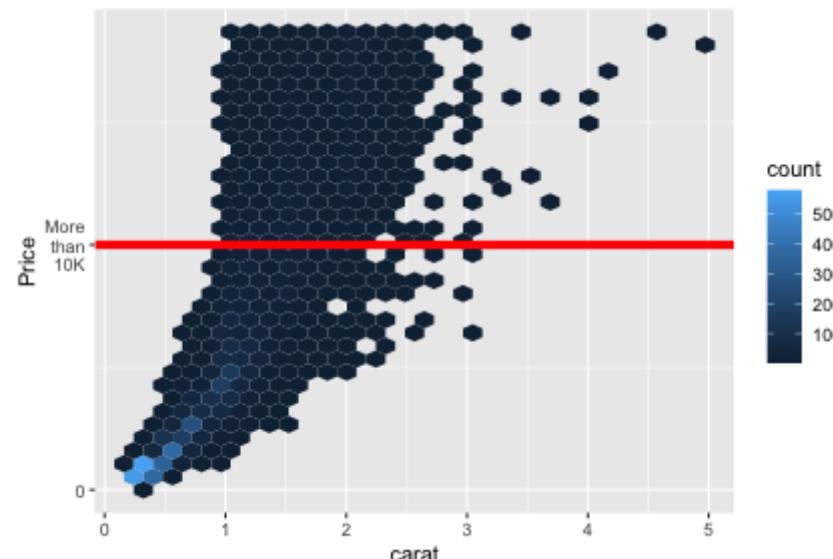
Guide: an axis or a legend

- The scale creates a **guide**: an **axis** or **legend**.
- So to modify these you generally use **scale_*** or other handy functions (guides, labs, xlab, ylab and so on).



Modify axis

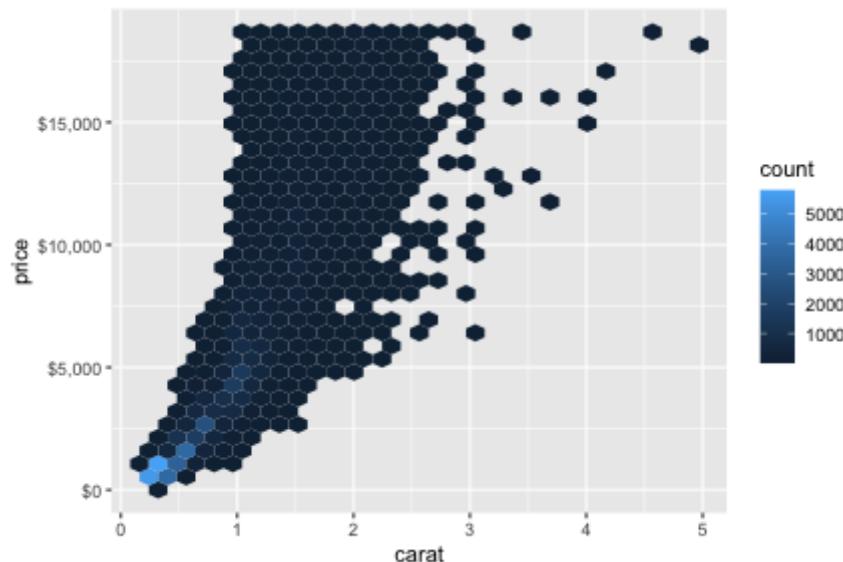
```
g +  
  scale_y_continuous(name = "Price",  
                     breaks = c(0, 10000),  
                     labels = c("0", "More\nthan\n10K")) +  
  geom_hline(yintercept = 10000, color = "red", size = 2)
```



Nicer formatting functions in scales

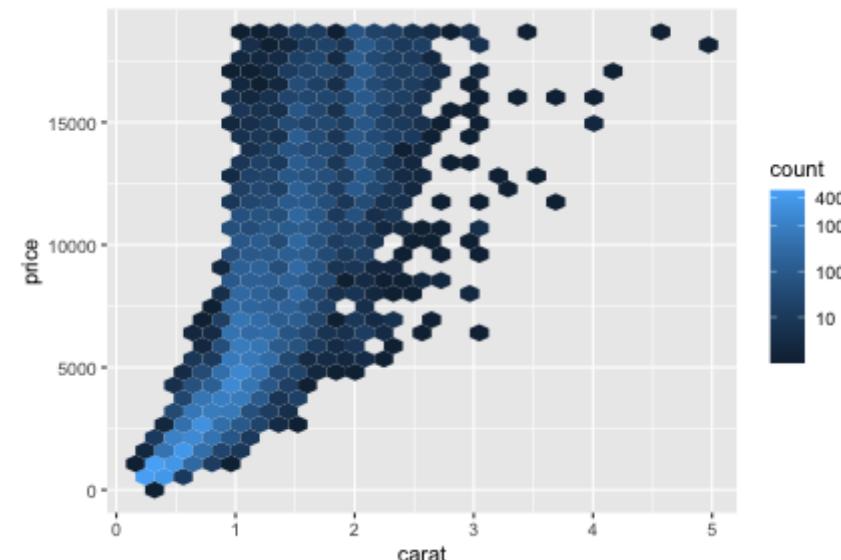


```
g +  
  scale_y_continuous(  
    label = scales::dollar_format()  
)
```



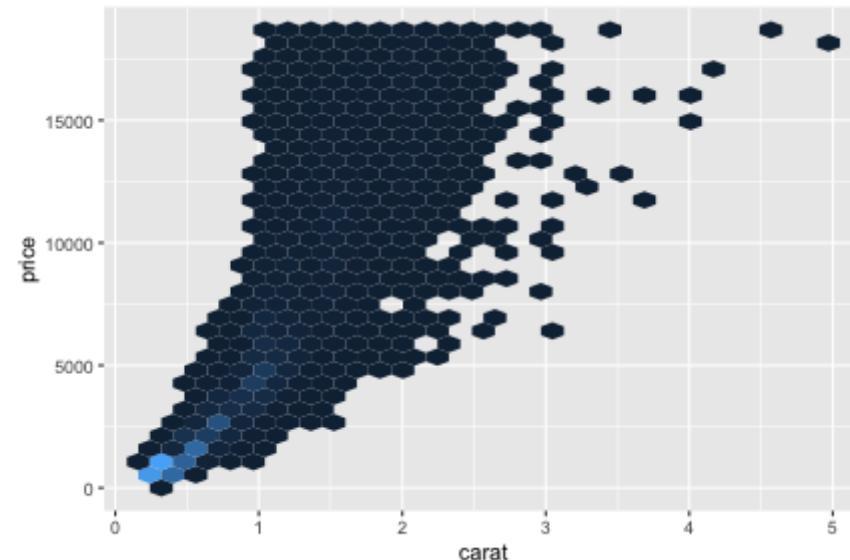
Modifying legend

```
g +  
  scale_fill_continuous(  
    breaks = c(0, 10, 100, 1000, 4000),  
    trans = "log10"  
)
```



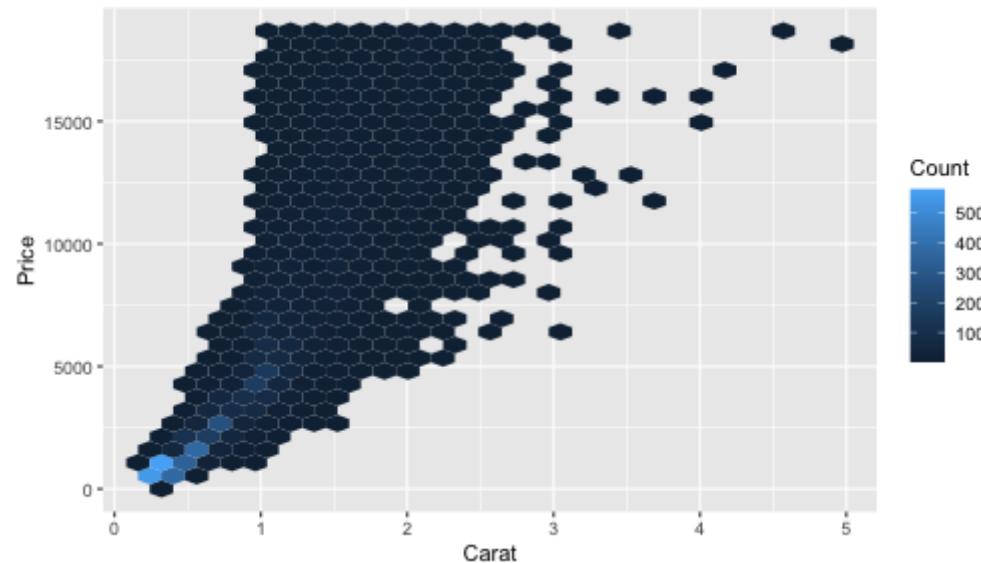
Removing legend

```
g +  
  scale_fill_continuous(  
    guide = "none"  
)
```



Alternative control of guides

```
g +  
  ylab("Price") + # Changes the y axis label  
  labs(x = "Carat", # Changes the x axis label  
        fill = "Count") # Changes the legend name
```



```
g + guides(fill = "none") # remove the legend
```

4

Color space

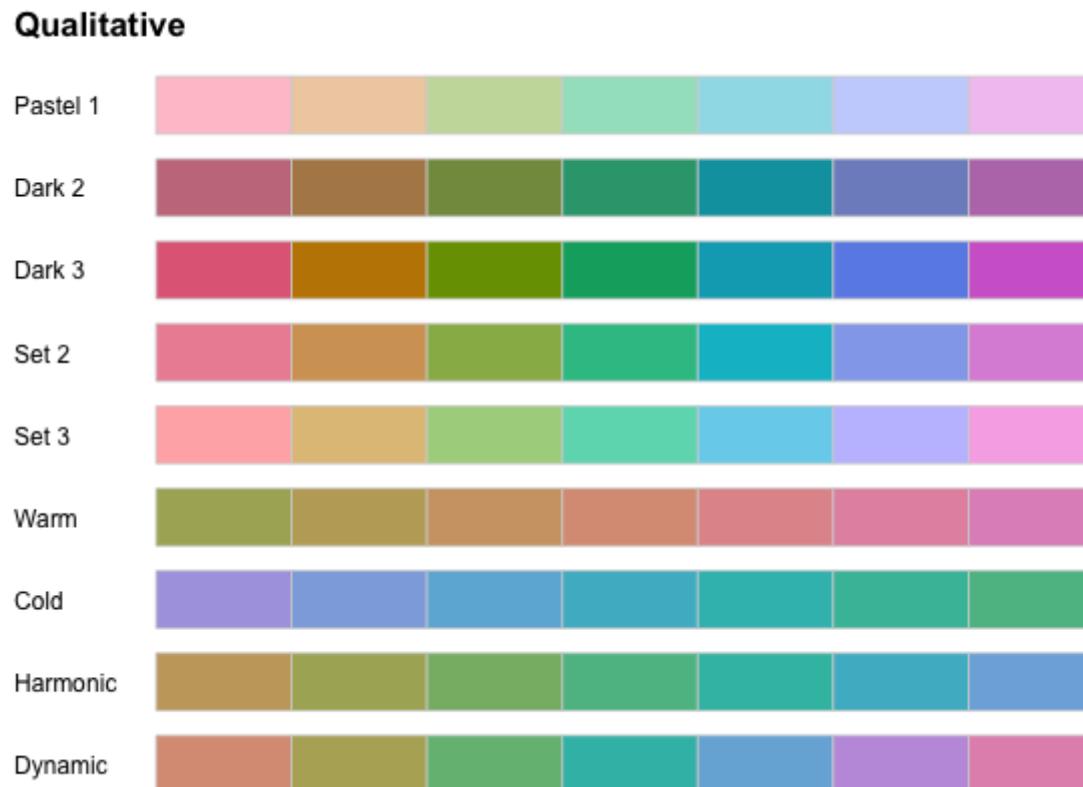
Zeileis, Fisher, Hornik, Ihaka, McWhite, Murrell, Stauffer, Wilke (2019). *colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes*. *arXiv 1903.06490*

Zeileis, Hornik, Murrell (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis* 53(9) 3259-3270

Qualitative palettes

designed for categorical variable with no particular ordering

```
colorspace::hcl_palettes("Qualitative", plot = "TRUE", n = 7)
```



Sequential palettes

designed for ordered categorical variable or number going from low to high (or vice-versa)

```
colorspace::hcl_palettes("Sequential", plot = "TRUE", n = 7)
```



Diverging palettes

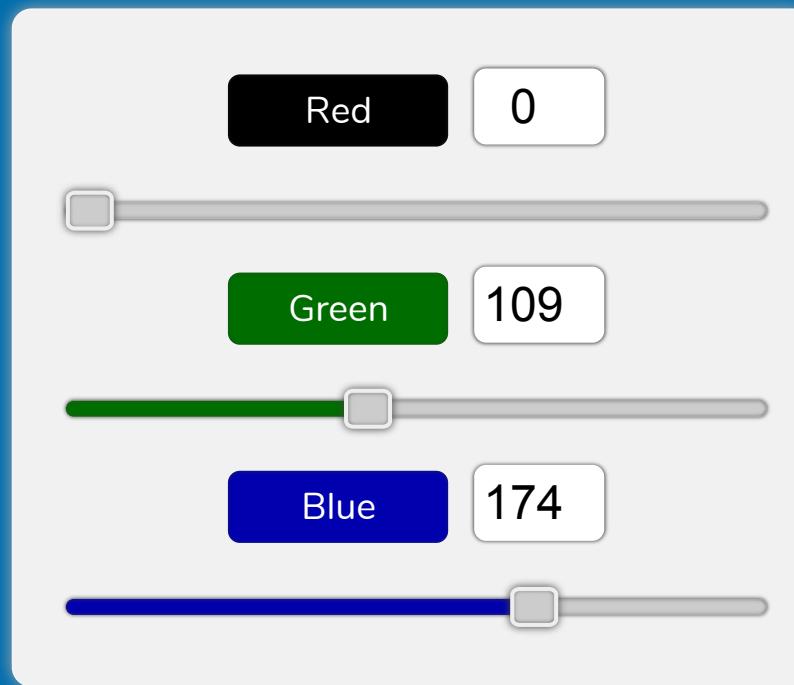
designed for ordered categorical variable or number going from low to high (or vice-versa) with a neutral value in between

```
colorspace::hcl_palettes("Diverging", plot = "TRUE", n = 7)
```



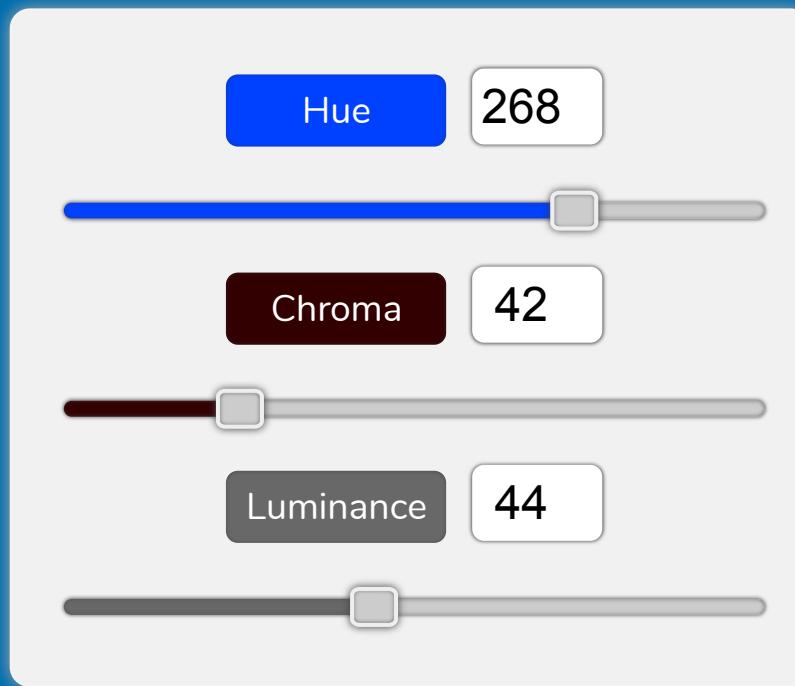
RGB color space

made for screen projection



HCL color space

made for human visual system



colorspace



Interactively choose/create a palette using the HCL color space.

```
library(colorspace)  
hcl_wizard() # OR choose_palette()
```

LIVE DEMO

hcl_wizard

Base Options

Type of palette
Basic: Sequential (multi-hue)

Base color scheme
Purple-Blue

Example
Map

Control Options

Reverse
 Correct colors
 Dark mode
 Desaturated

Vision
 Normal
 Deutan
 Protan
 Tritan

Color Settings

HUE 1: 300
HUE 2: 200
CHRON: 60
CHRON: 0
LUMIN.: 25
LUMIN.: 95
POWER: 0.7
POWER: 1.3
NUMBER: 7

Return to R

Example Plot Spectrum Color Plane Export Info

RAW GrADS Python matlab R Register

If you use *R* the preferred way to handle color palettes is to use [choose_palette\(\)](#) or [hclwizard\(\)](#) on your local machine. Both graphical user interfaces return an *R* color palette function. However, you can also use the function call below to use the current palette in your *R* scripts.

```
## Custom color palette
sequential_hcl(n = 7, h = c(300, 200), c = c(60, NA, 0), l = c(25, 95), power = c(0.7, 1.3), register = )
```

Custom color palettes can also be *registered* to be able to call custom palettes by name. If the optional argument *register* = "Custom-Palette" is set (where "Custom-Palette" is the name of your new palette) the palette will be added to the list of available color palettes. Existing palettes can also be overruled. Note that the graphical interfaces ([choose_palette\(\)](#)) will not use custom palettes. These register-calls can also be added to your local *~/.Rprofile*.

```
## Register custom color palette
colorspace::sequential_hcl(n = 7, h = c(300, 200), c = c(60, NA, 0), l = c(25, 95), power = c(0.7, 1.3), register = "Custom-Palette")
```

R colorspace 1.4.

Choose your palette > Export > R > Copy the command

Registering your own palette

```
library(colorspace)
# register your palette
sequential_hcl(n = 7,
               h = c(300, 200),
               c = c(60, 0),
               l = c(25, 95),
               power = c(2.1, 0.8),
               register = "my-set")

# now generate from your palette
sequential_hcl(n = 3,
               palette = "my-set")

## [1] "#6B0077" "#7C8393" "#F1F1F1"
```

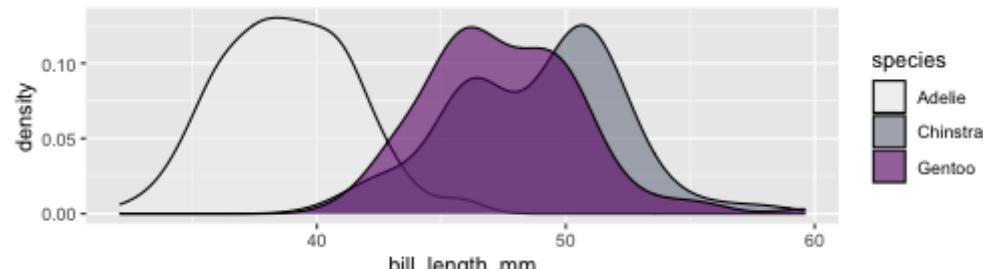
```
hcl_palettes(n = 5, palette = "my-set", plot = T)
```

Sequential (multi-hue)

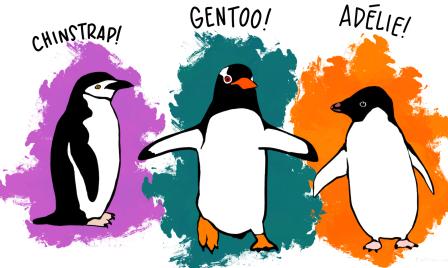


Combining with ggplot:

```
ggplot(penguins,
       aes(bill_length_mm, fill = species)) +
  geom_density(alpha = 0.6) +
  # notice here you don't need to specify the n!
  scale_fill_discrete_sequential(palette = "my-set")
```



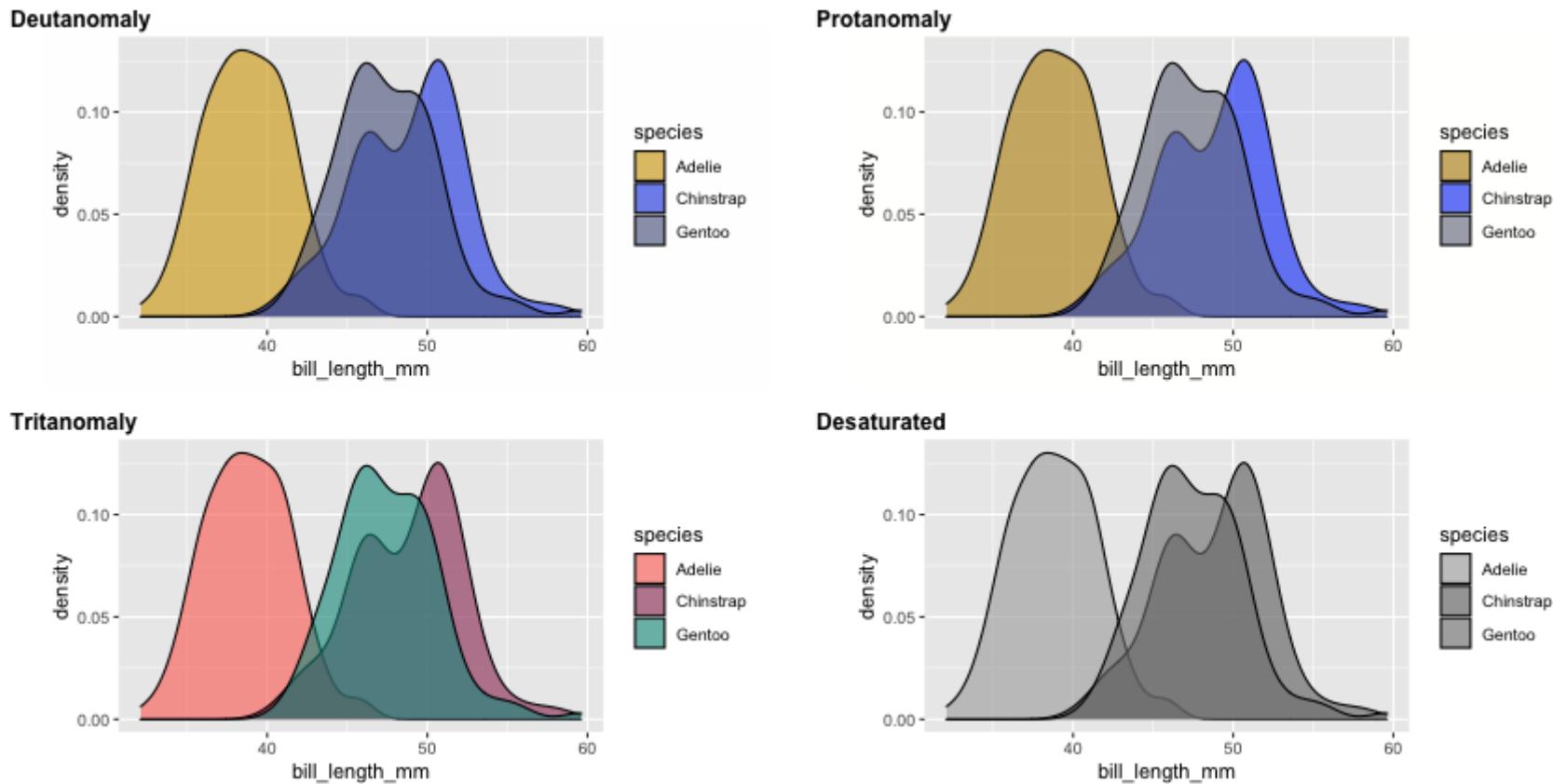
Manually selecting colors



```
g <- ggplot(penguins,  
            aes(bill_length_mm, fill = species)) +  
  geom_density(alpha = 0.6) +  
  scale_fill_manual(  
    breaks = c("Adelie", "Chinstrap", "Gentoo"), # optional but makes it more robust  
    values = c("darkorange", "purple", "cyan4"))  
g
```

Check that it's colour blind friendly!

```
colorblindr::cvd_grid(g)
```



See [here](#) for more about colorblindr.



`</> Open part1-exercise-03.Rmd`

15 : 00

Session Information

```
devtools::session_info()
```

```
## - Session info 🤴🏻 🧑🏿 😱 ━━━━  
## hash: baby: medium-dark skin tone, woman scientist: dark skin tone, face with open mouth  
##  
## setting value  
## version R version 4.1.2 (2021-11-01)  
## os      macOS Big Sur 10.16  
## system x86_64, darwin17.0  
## ui      X11  
## language (EN)  
## collate en_AU.UTF-8  
## ctype   en_AU.UTF-8  
## tz      Australia/Melbourne  
## date   2022-02-20
```

These slides are licensed under

