



# Data Visualisation with R

## Workshop Part 1

### Scales and color

Presented by Emi Tanaka

Department of Econometrics and Business Statistics



MONASH University

6th Dec 2021 @ Statistical Society of Australia NSW Branch | Zoom

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

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

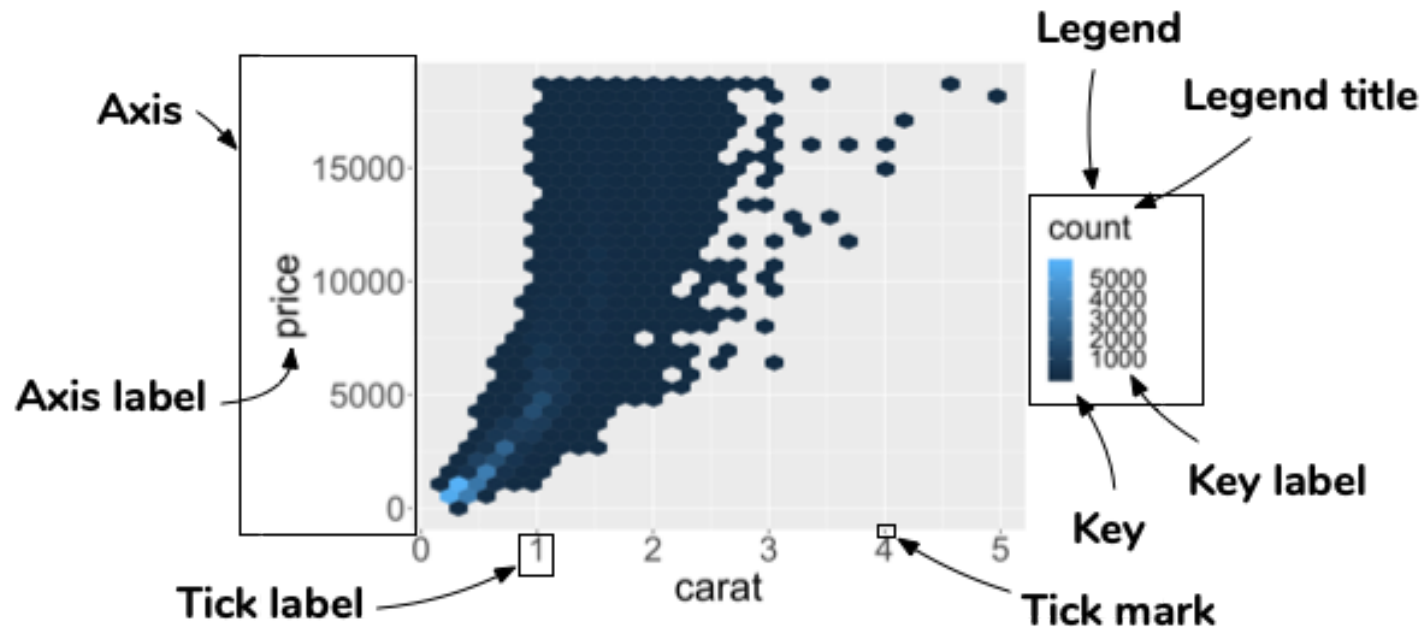
4

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\n than\n 10K")) +  
  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*

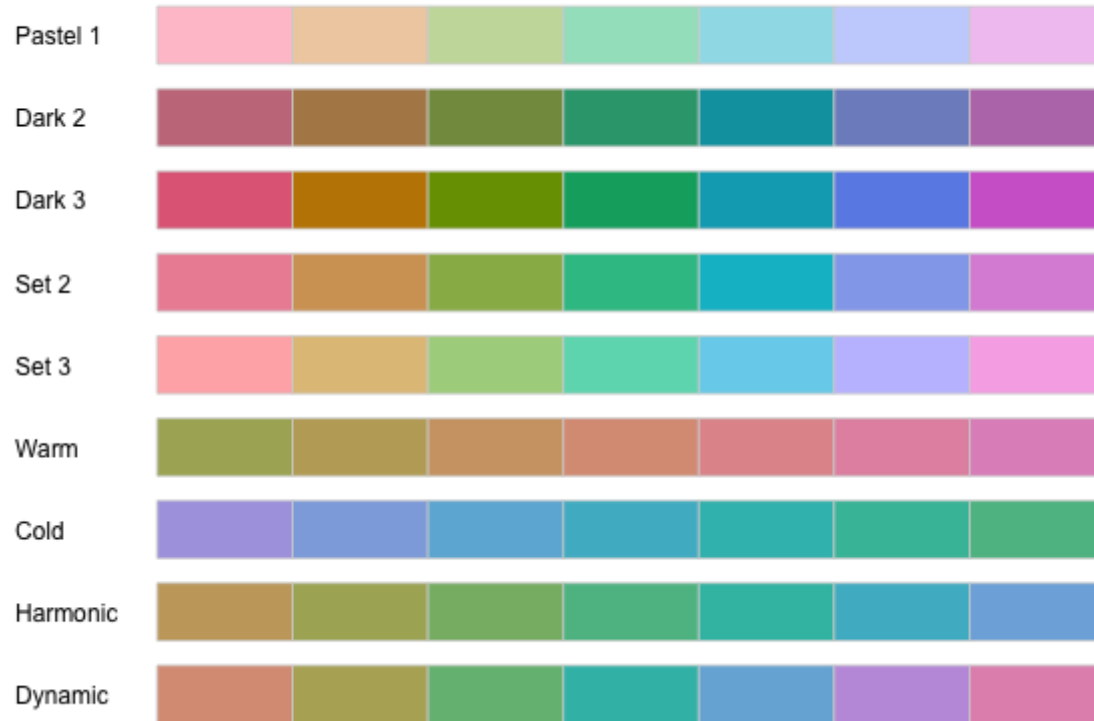
Zeileis, Hornik, Murrell (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis* 53(9) 3259-3270

# Qualitative palettes

designed for categorical variable with no particular ordering

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

## Qualitative



# 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

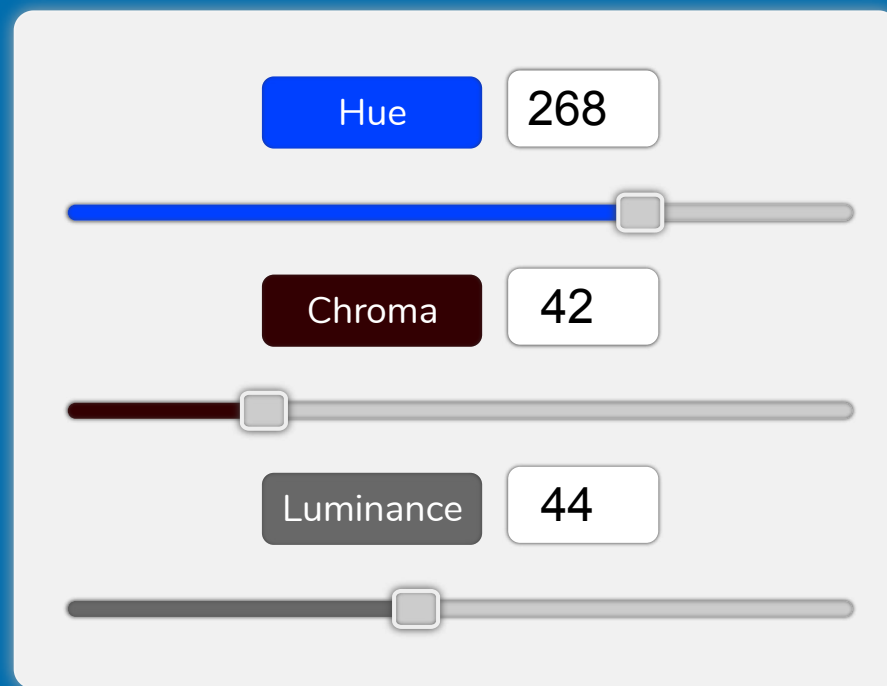
made for screen projection

The image shows a user interface for adjusting RGB color values. It consists of three vertical sliders. The top slider is for Red, with a value of 0. The middle slider is for Green, with a value of 109. The bottom slider is for Blue, with a value of 174. Each slider has a colored bar representing the current value and a small square handle to adjust it.

Color	Value
Red	0
Green	109
Blue	174

# HCL color space

made for human visual system



The image shows a control interface for the HCL color space. It consists of three horizontal sliders, each with a label, a value box, and a slider track. The top slider is labeled 'Hue' with a value of 268 and a blue track. The middle slider is labeled 'Chroma' with a value of 42 and a dark red track. The bottom slider is labeled 'Luminance' with a value of 44 and a grey track. Each slider has a small square knob indicating the current value.

Property	Value
Hue	268
Chroma	42
Luminance	44

# colorspace

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

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

LIVE DEMO

# hcl\_wizard

The screenshot displays the hcl\_wizard web interface. On the left, the 'Base Options' panel includes a 'Type of palette' dropdown set to 'Basic: Sequential (multi-hue)', a 'Base color scheme' dropdown set to 'Purple-Blue', and an 'Example' dropdown set to 'Map'. Below these are 'Control Options' with checkboxes for 'Reverse', 'Correct colors' (checked), 'Dark mode', and 'Desaturated'. The 'Vision' section has radio buttons for 'Normal' (selected), 'Deutan', 'Protan', and 'Tritan'. The 'Color Settings' panel features sliders for HUE 1 (300), HUE 2 (200), CHROM (60), LUMIN. (25), LUMIN. (95), POWER (0.7), POWER (1.3), and NUMBE (7). A 'Return to R' button is at the bottom of this panel. The top navigation bar includes 'Example Plot', 'Spectrum', 'Color Plane', 'Export', and 'Info'. Below this, a row of buttons shows 'RAW', 'GrADS', 'Python', 'matlab', 'R' (selected), and 'Register'. The main content area contains text explaining the use of the R function `choose_palette()` or `hclwizard()` and provides two code snippets for custom color palettes. The first snippet shows the registration of a custom palette, and the second shows its use in a `sequential_hcl` call. The bottom right corner of the interface indicates 'R colorspace 1.4.'.

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")
```

Return to R

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"
```

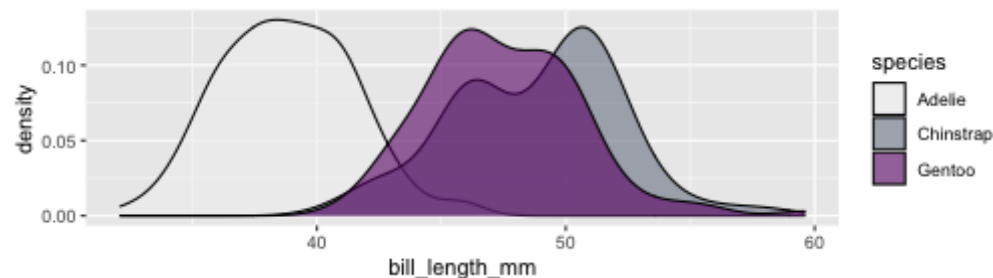
```
hcl_palettes(n = 5, palette = "my-set", plot = T)
```

Sequential (multi-hue)

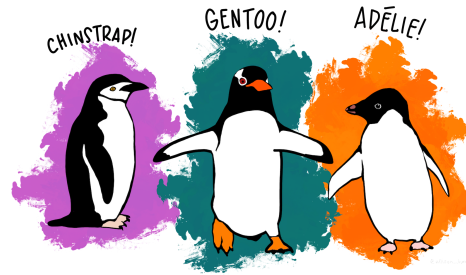


## Combining with ggplot:

```
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



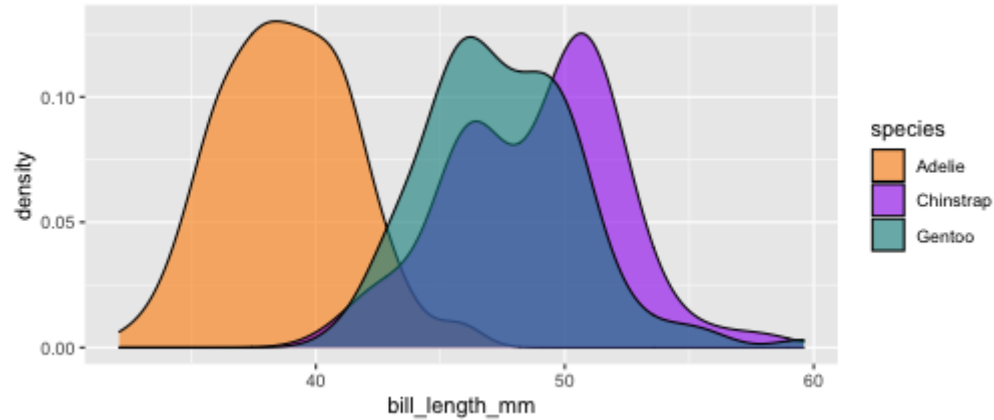
```
g <- ggplot(penguins,  
            aes(bill_length_mm, fill = species)) +  
  geom_density(alpha = 0.6) +  
  scale_fill_manual(  
    breaks = c("Adelie", "Chinstrap", "Gentoo"), # optional but makes it more robust  
    values = c("darkorange", "purple", "cyan4"))
```

g

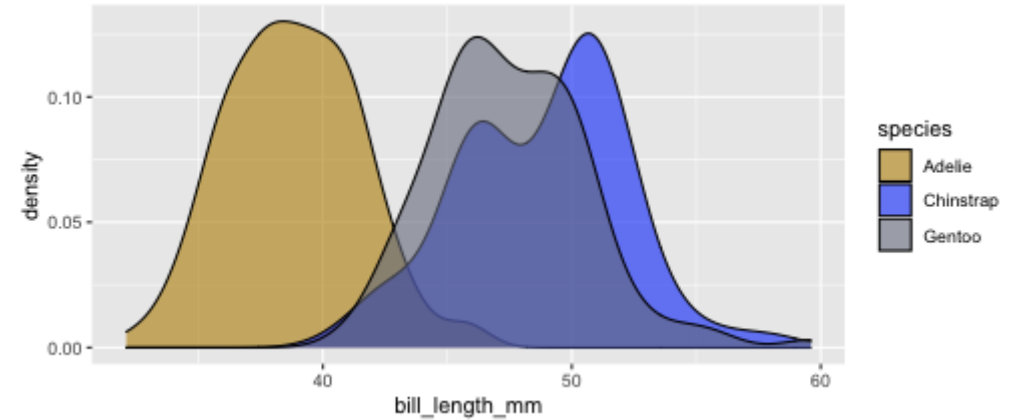
# Check that it's colour blind friendly!

```
cols <- c("darkorange", "purple", "cyan4")
```

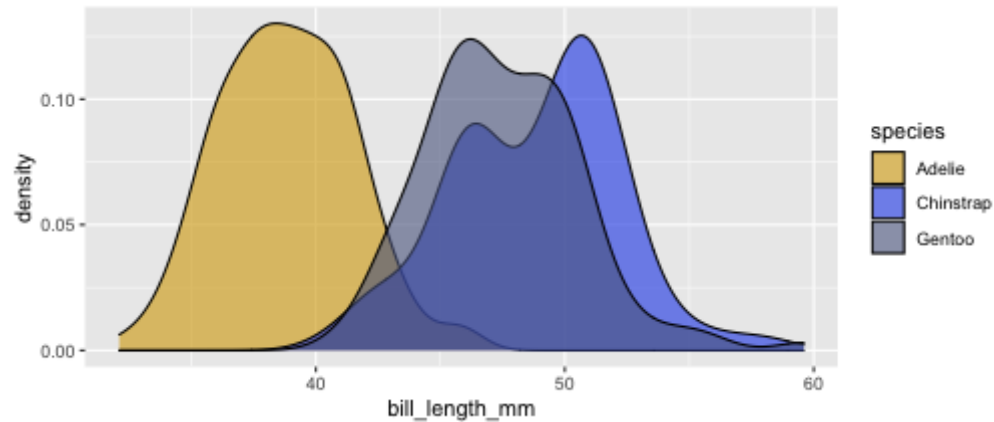
```
g # original
```



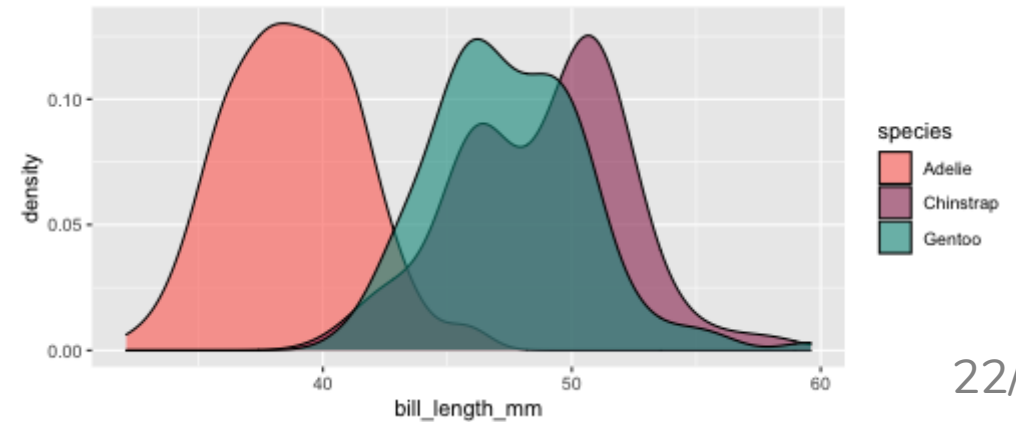
```
g + scale_fill_manual(values = protan(cols))
```



```
g + scale_fill_manual(values = deutan(cols))
```



```
g + scale_fill_manual(values = tritan(cols))
```








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

**15:00**

# Session Information

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

```
## - Session info    _____  
## hash: pencil, globe showing Americas, bowling  
##  
## 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 2021-11-30  
## pandoc 2.11.4.2 (Applications/PSStudio.app/Contents/MacOS/pandoc/ (via markdown))
```

These slides are licensed under

