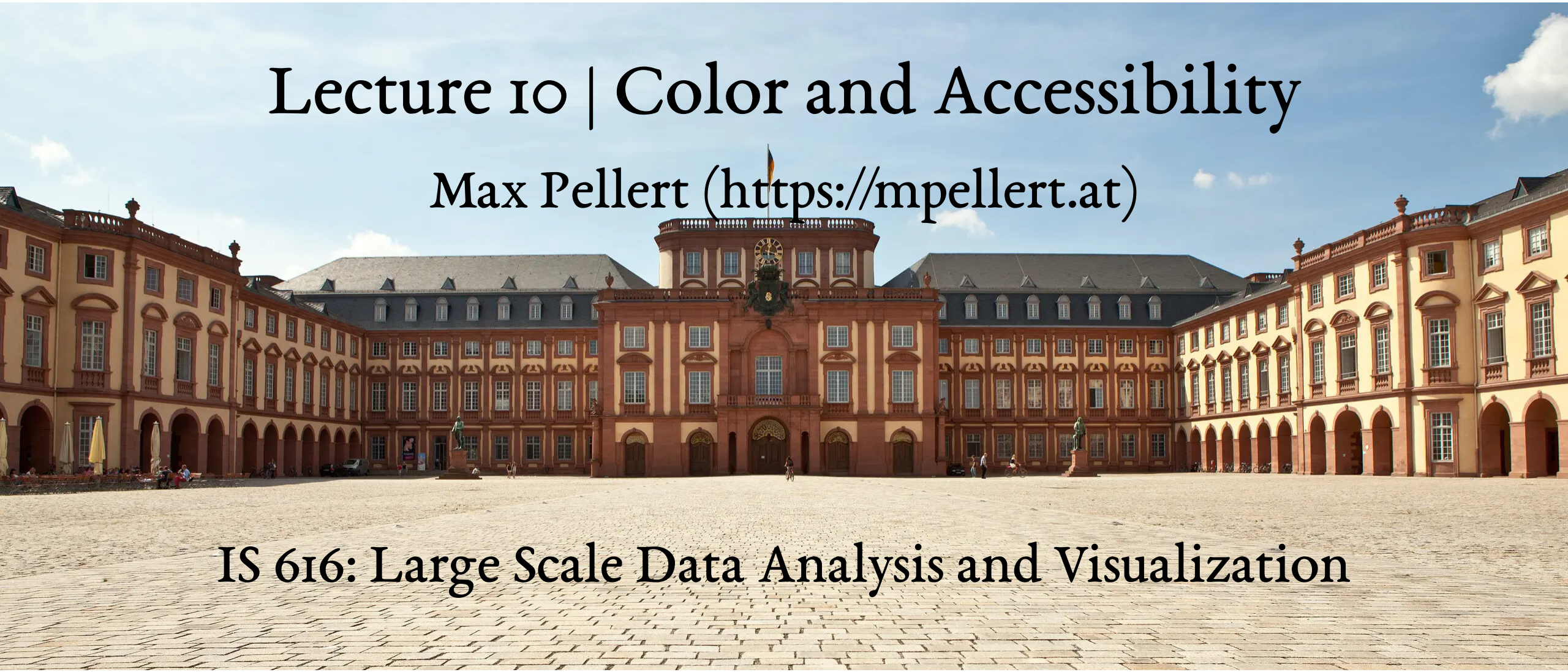
Lecture 10 | Color and Accessibility

Max Pellert (https://mpellert.at)



IS 616: Large Scale Data Analysis and Visualization


0:00 / 6:32

How does a computer represent color?

The RGB color space

- red R (0-255): amount of red light
- green G (0-255): amount of green light
- blue B (0-255): amount of blue light

R	G	B	hex code	color
0	0	0	#000000	
255	0	0	#FF0000	
0	255	255	#00FFFF	
128	128	128	#808080	
0	158	115	#009E73	
255	255	255	#FFFFFF	

Converting RGB to hexadecimal

"RGB values are usually given in the 0–255 range; if they are in the 0–1 range, the values are multiplied by 255 before conversion.

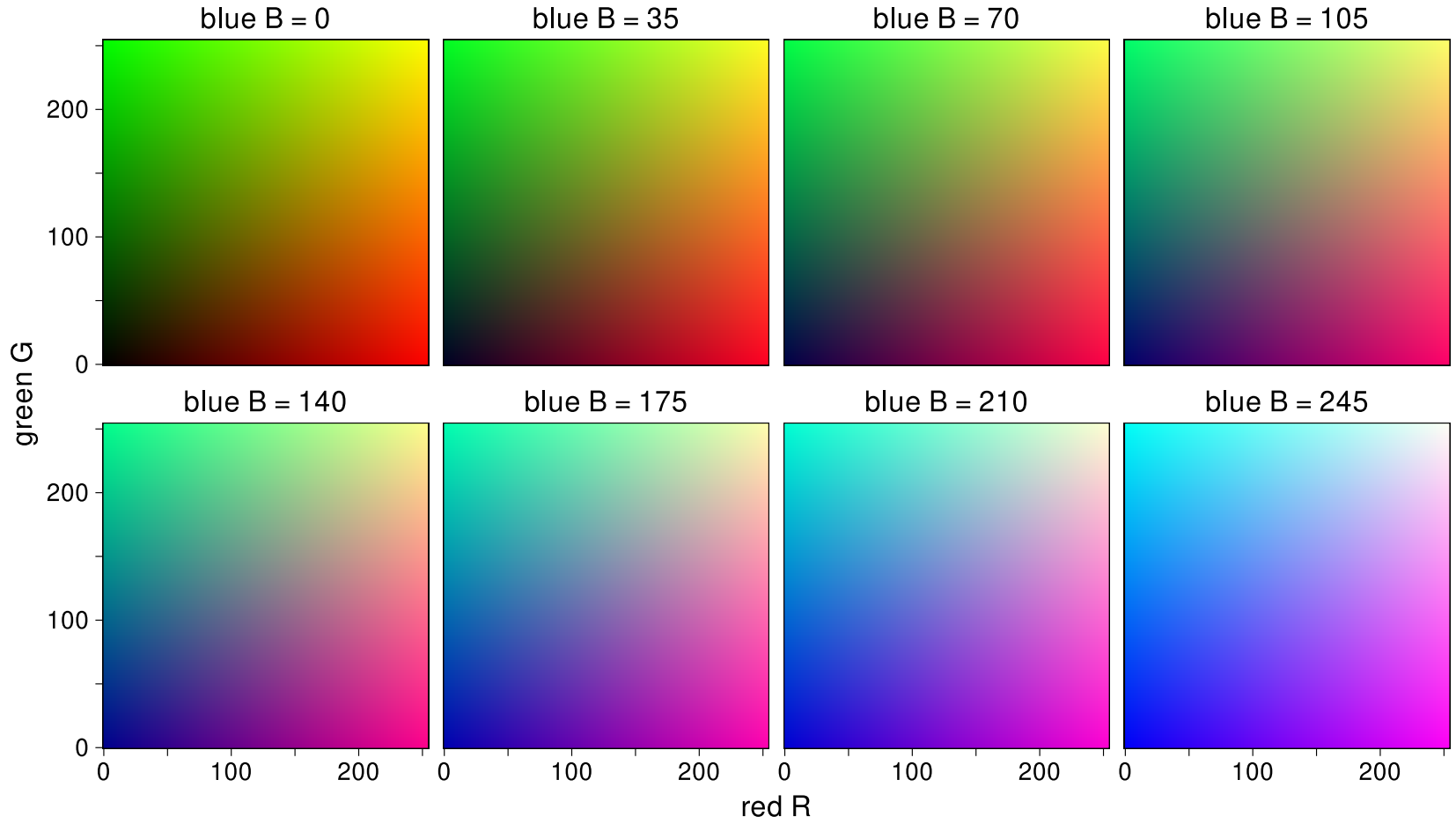
This number divided by sixteen (integer division; ignoring any remainder) gives the first hexadecimal digit (between 0 and F, where the letters A to F represent the numbers 10 to 15.

The remainder gives the second hexadecimal digit.

For instance, the RGB value 201 divides into 12 groups of 16, thus the first digit is C. A remainder of nine gives the hexadecimal number C9. This process is repeated for each of the three color values."

https://en.wikipedia.org/wiki/Web_colors

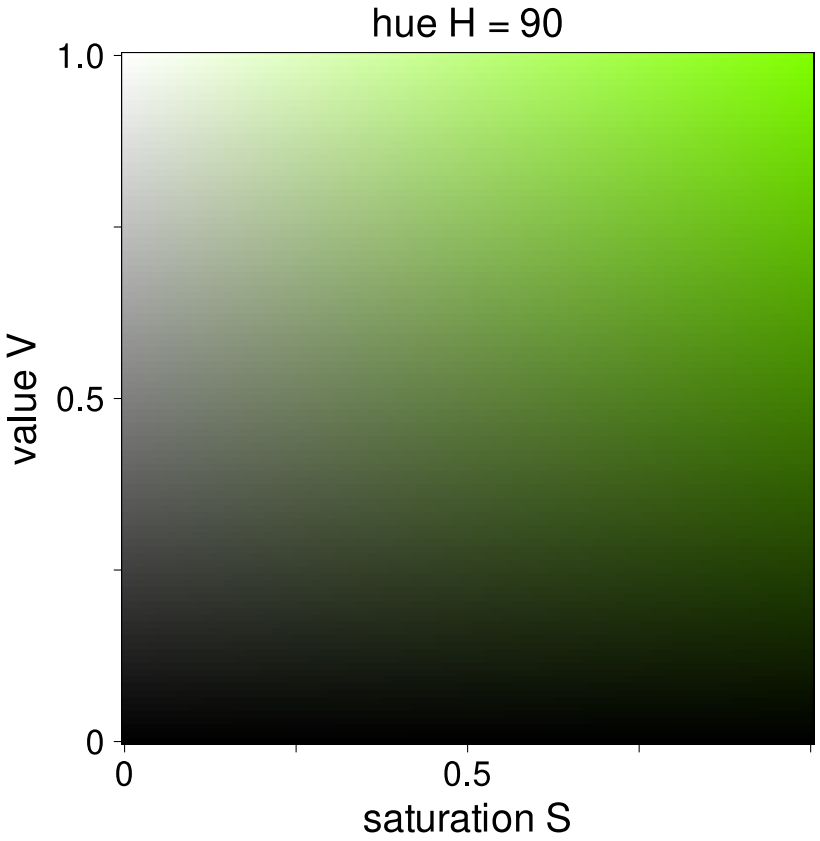
The RGB color space



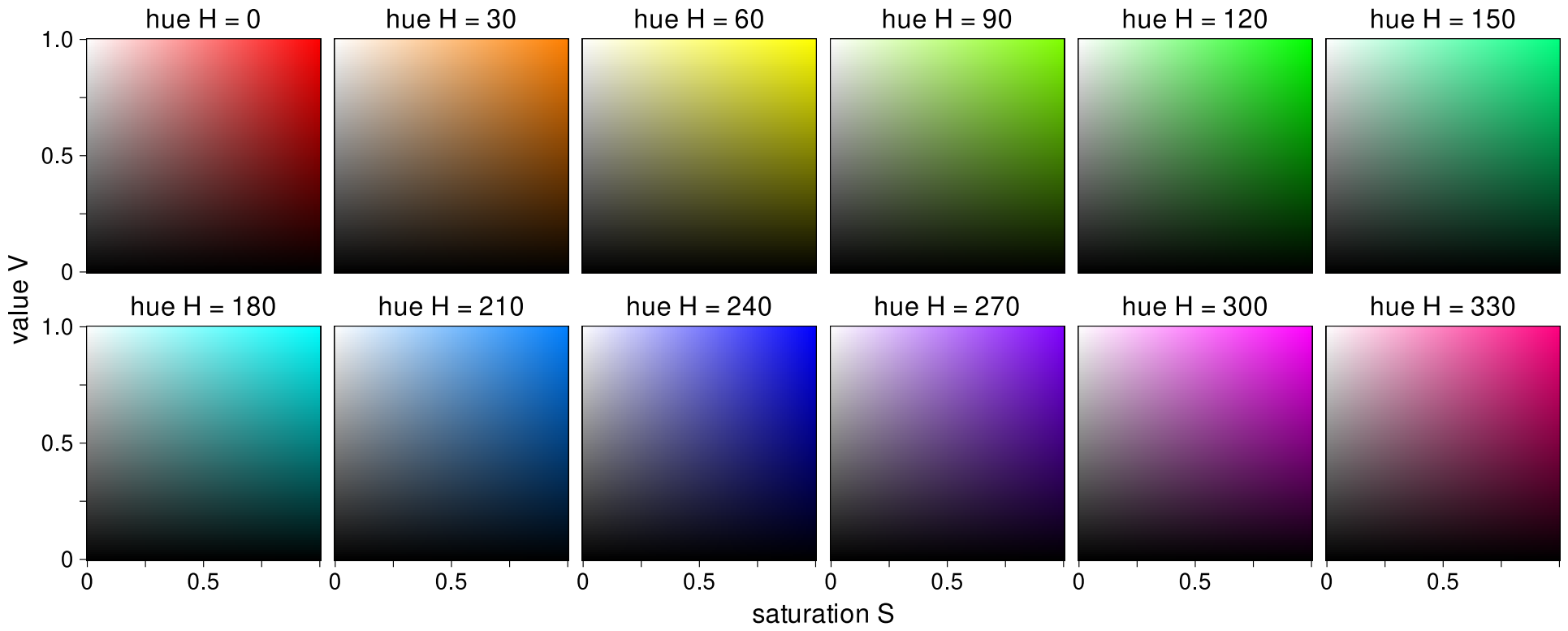
Humans cannot reason well about the RGB color space

The HSV color space

- hue H (0-360):
  hue of the color
- saturation S (0-1):
  colorfulness relative to the brightness of the color
- value V (0-1):
  subjective perception of amount of light emitted

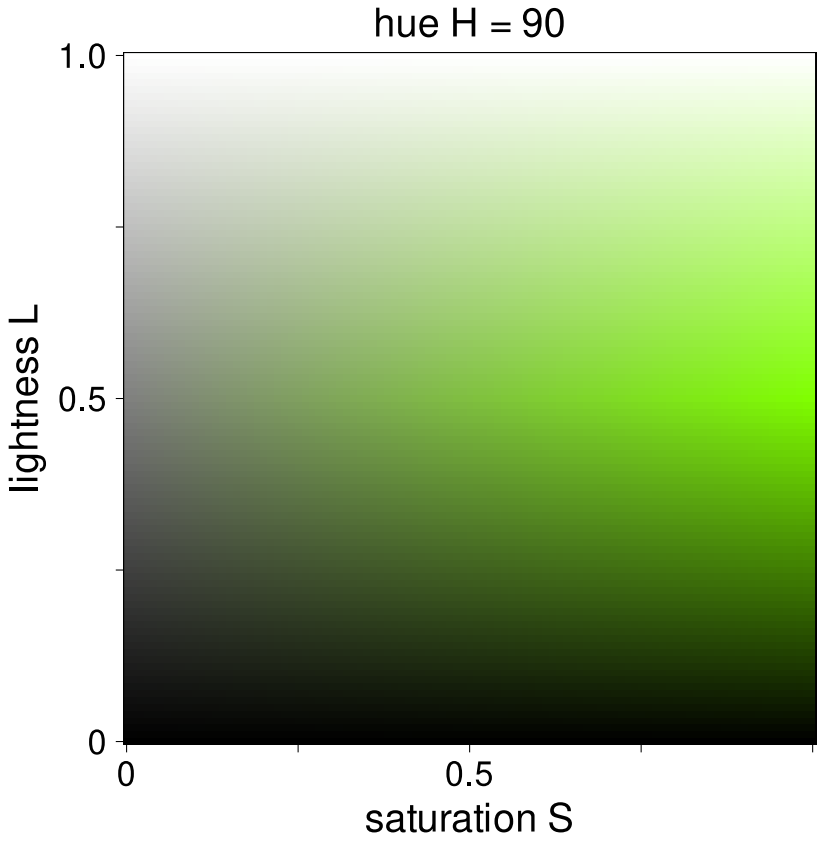


The HSV color space

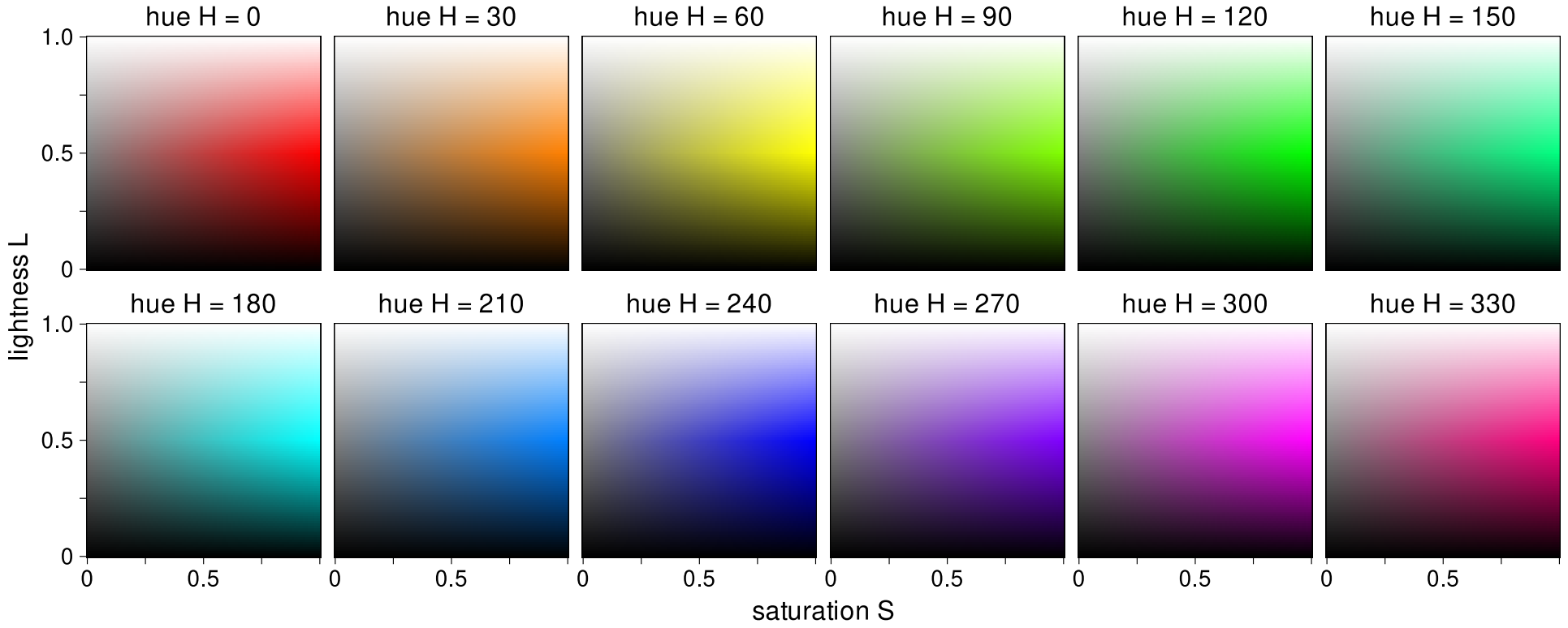


The HLS color space

- hue H (0-360):
  hue of the color
- lightness L (0-1):
  brightness relative to the brightness of a similarly illuminated white
- saturation S (0-1):
  colorfulness relative to the brightness of the color



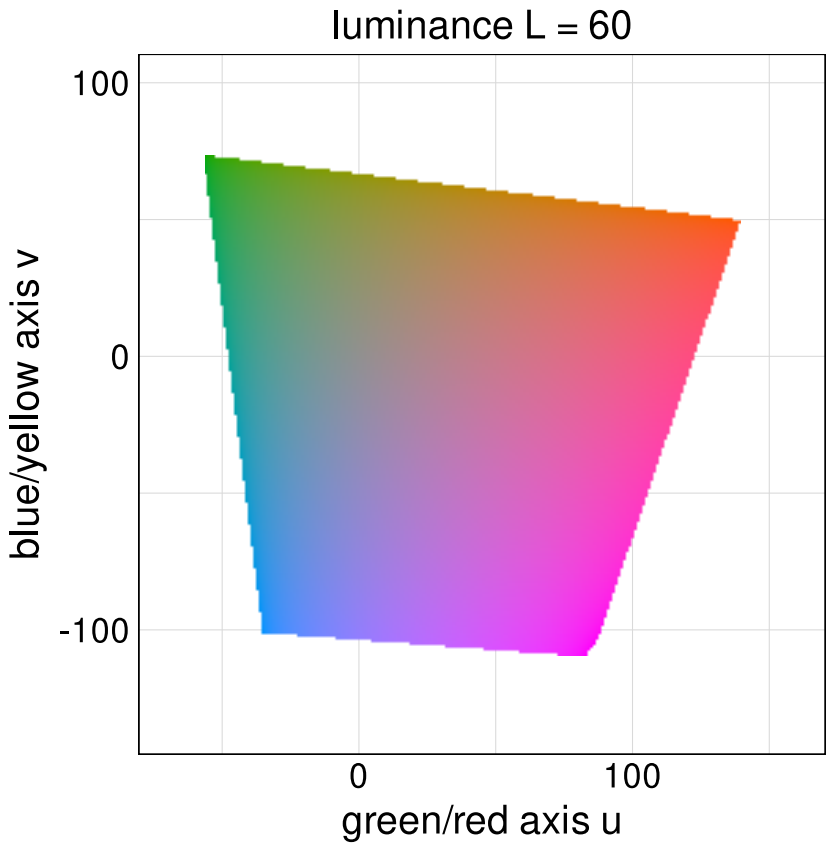
The HLS color space



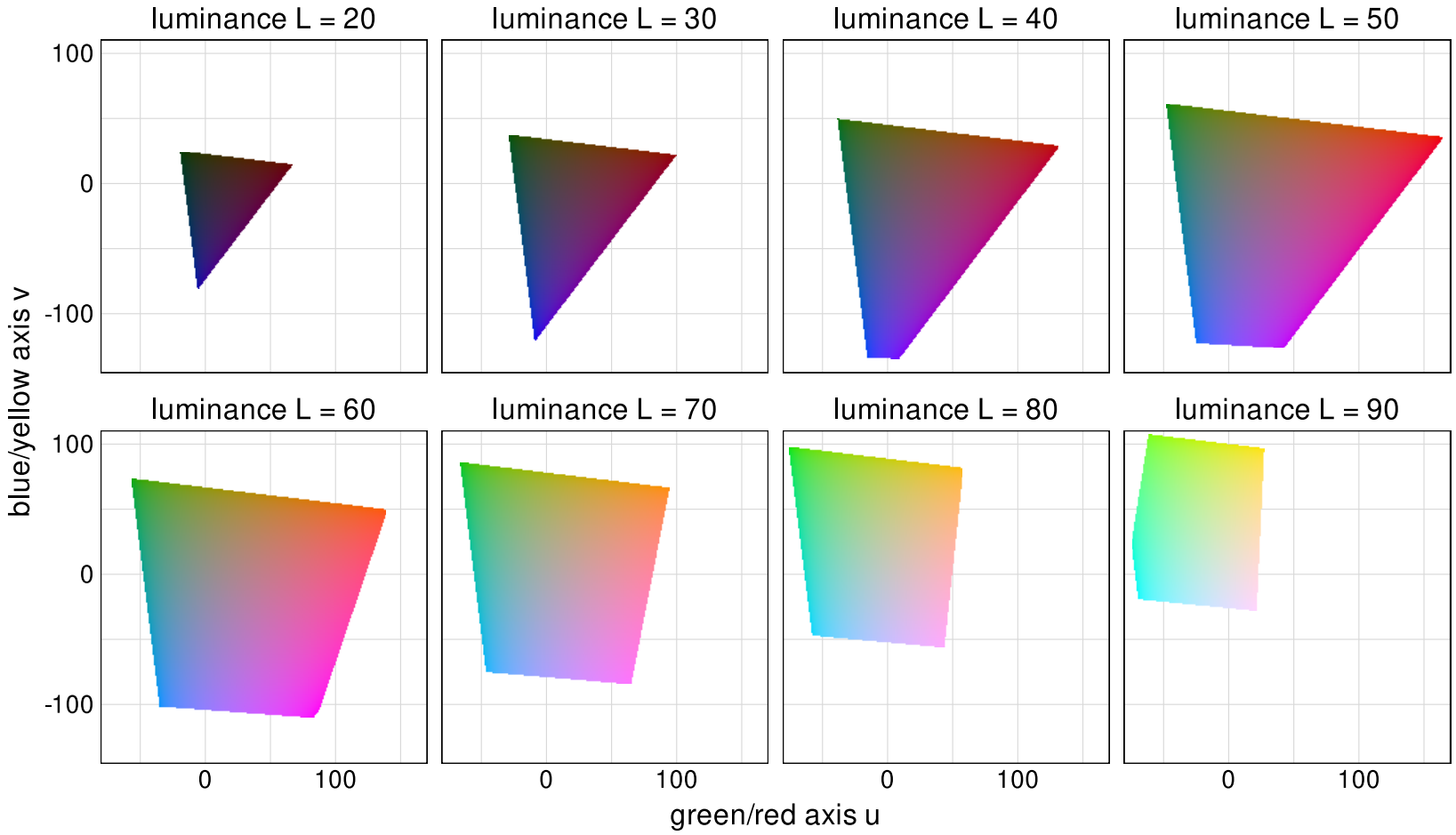
Humans cannot reason well about HSV or HLS either

The Luv color space

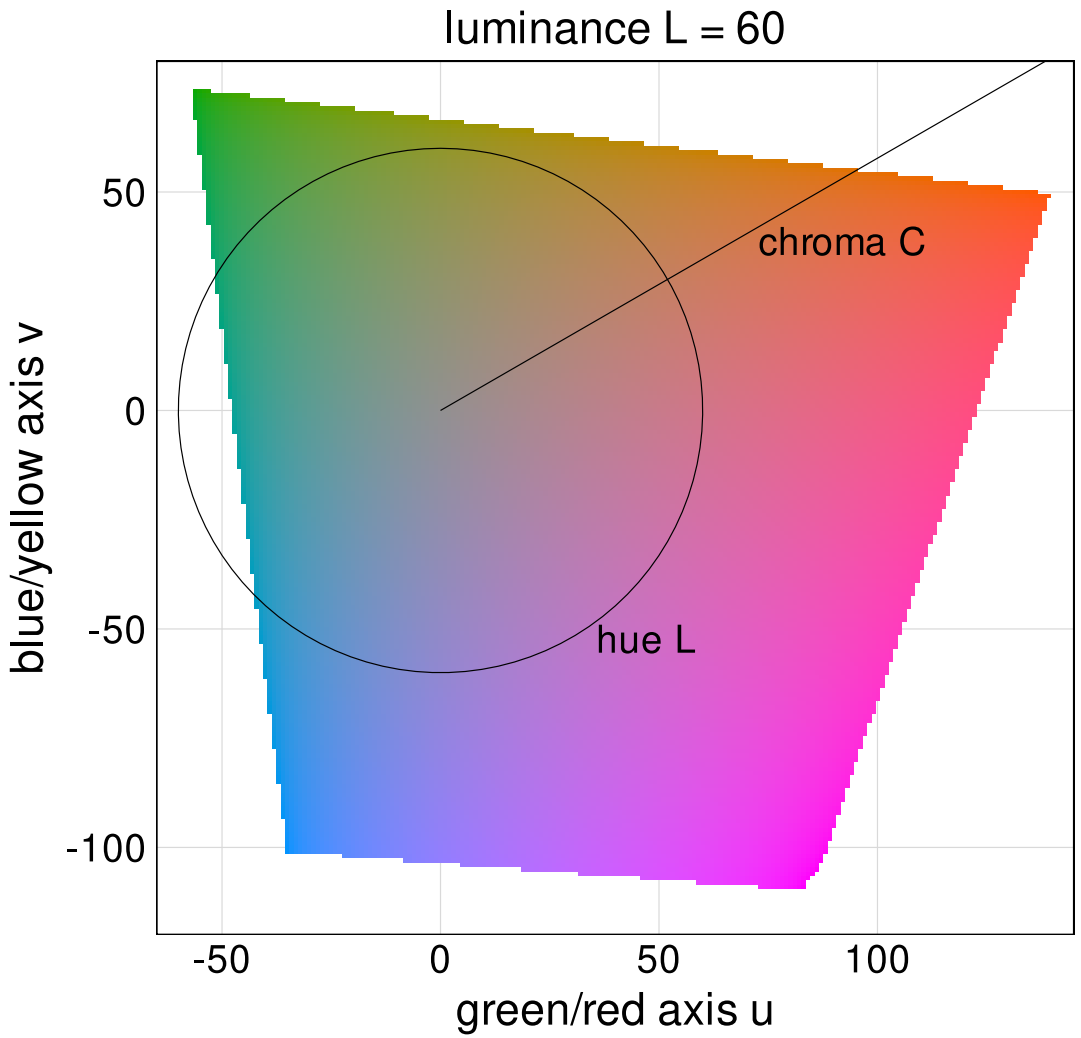
- luminance L (0-100):
  amount of light emitted
- green/red axis u (approx. -100 to 100):
  amount of green vs. red
- blue/yellow axis v (approx. -100 to 100):
  amount of blue vs. yellow



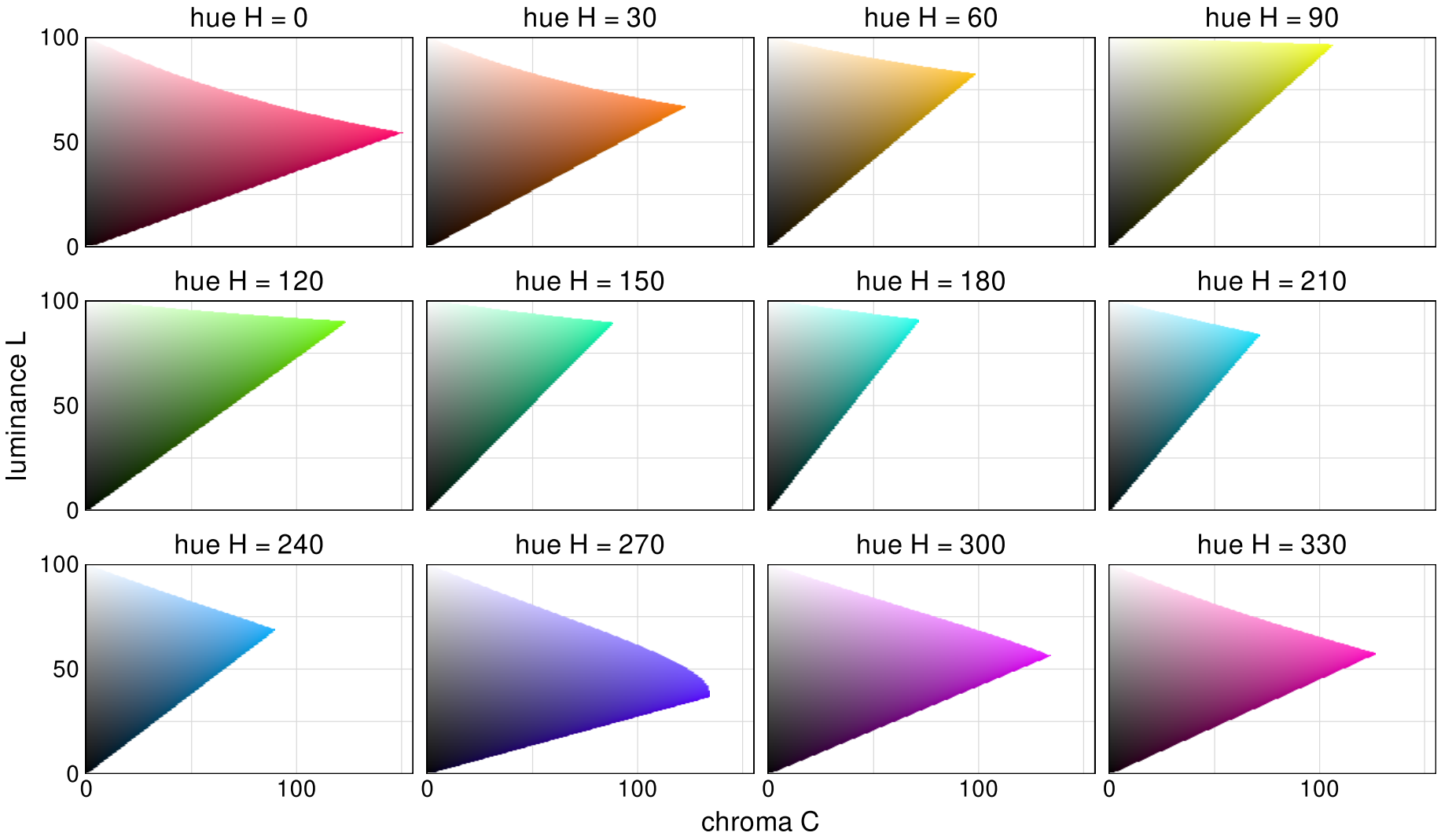
The Luv color space



The HCL color space: polar Luv



The HCL color space: polar Luv



https://colorspace.r-forge.r-project.org/

https://github.com/retostauffer/python-colorspace

Explore HCL colors interactively

In R:

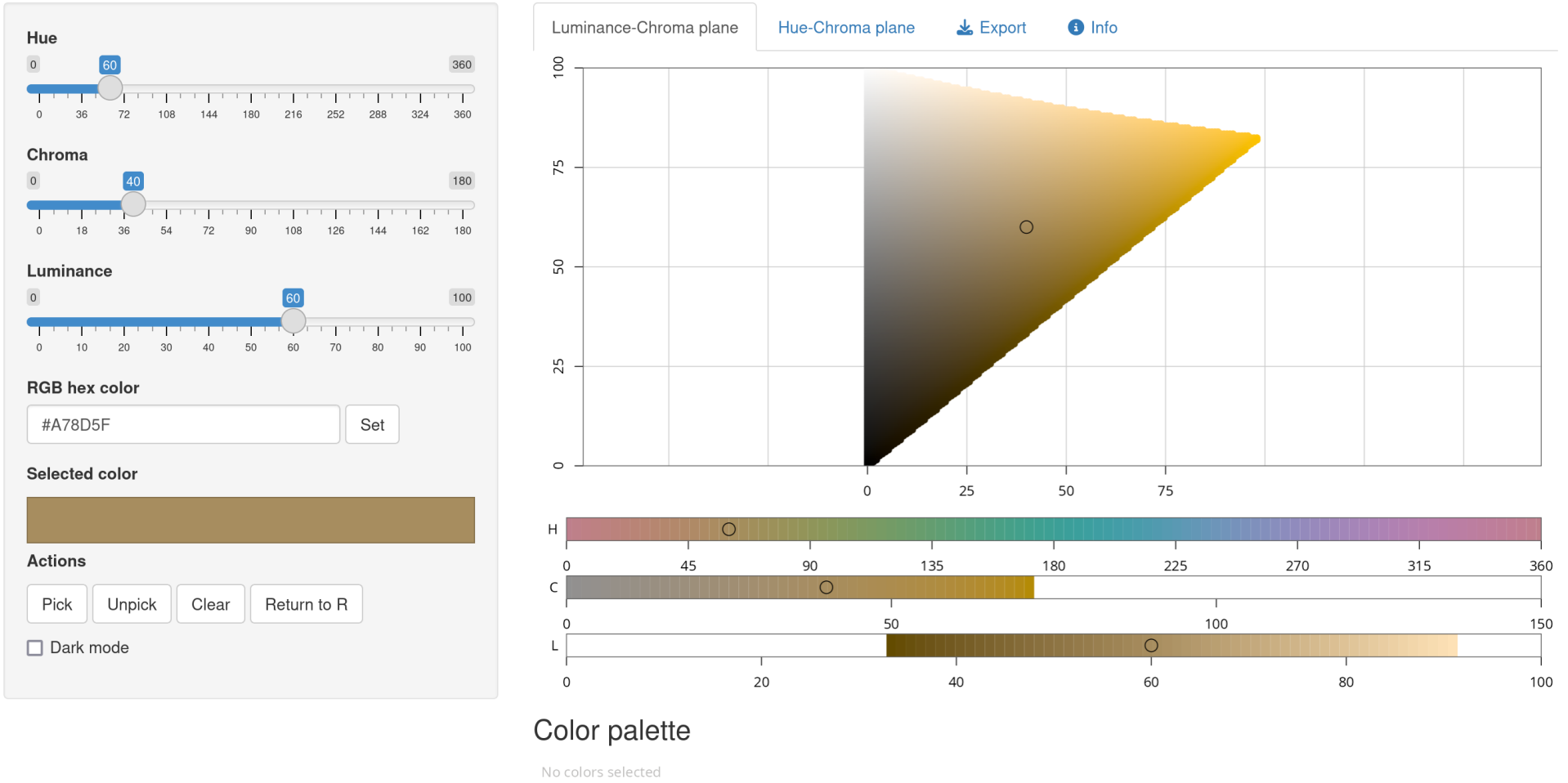
```r
colorspace::choose_color()
```

In Python:

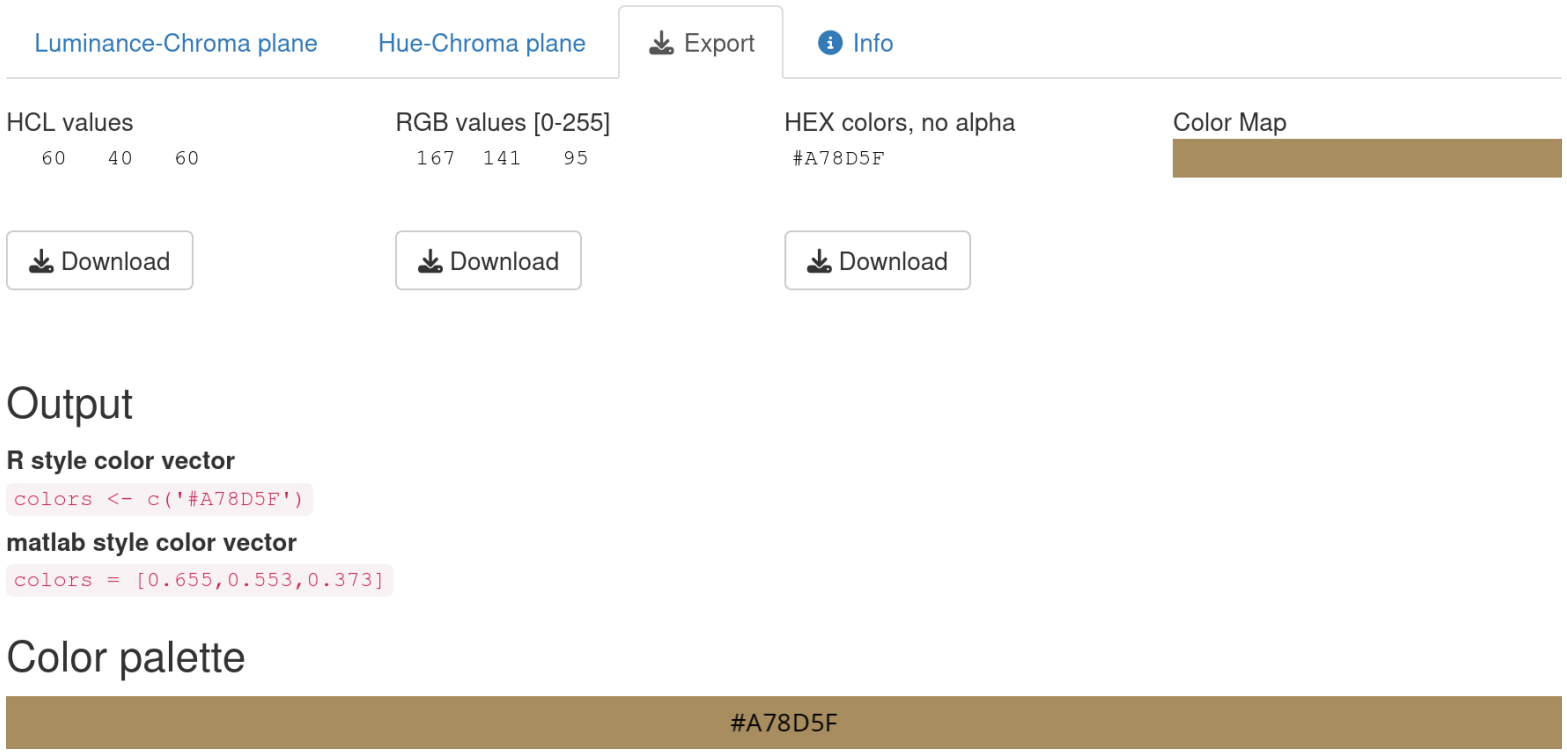
```python
from colorspace import choose_palette

pal=choose_palette()
```

```r
colorspace::choose_color()
```



Allows for easy exporting of palettes



A few considerations when choosing colors

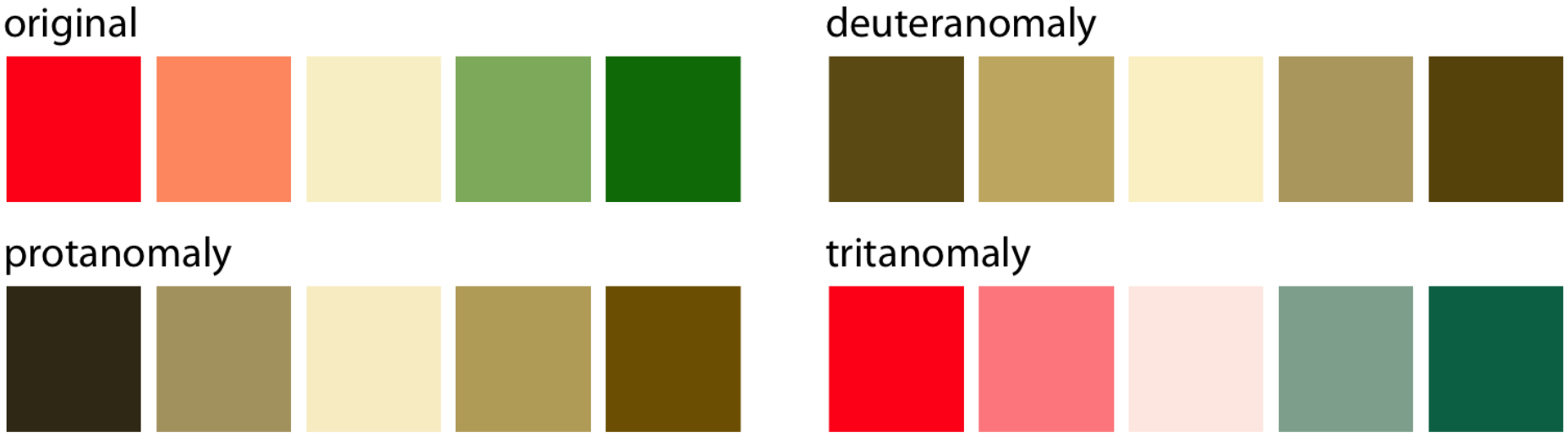
1. Avoid high chroma

High chroma: Toys

Low chroma: Elegance

2. Be aware of color-vision deficiency (to be continued...)

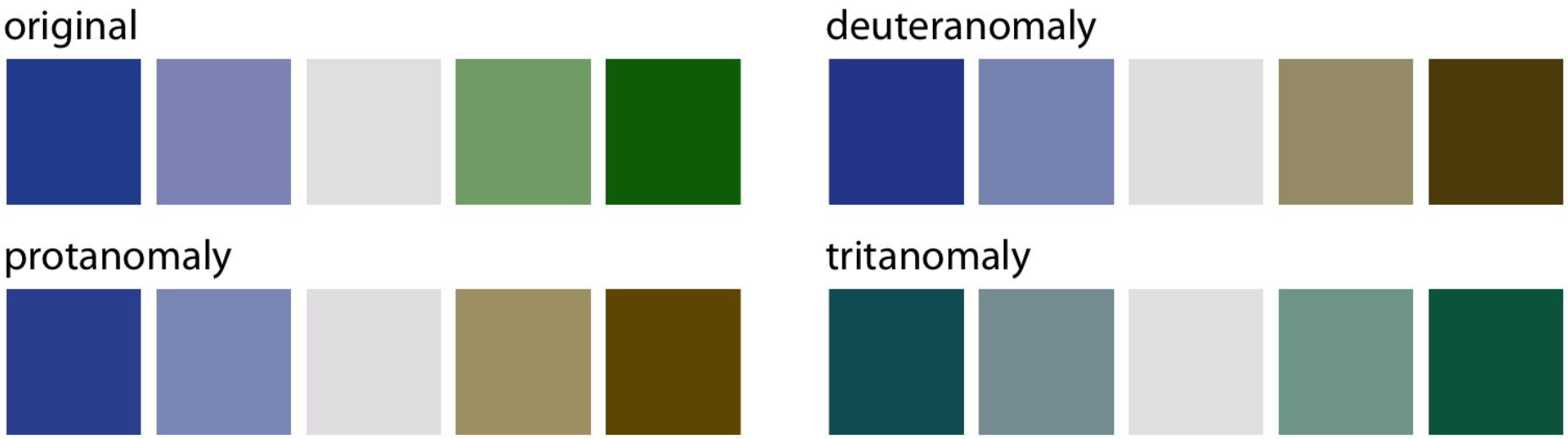
5%–8% of men are color blind!



Red-green color-vision deficiency is the most common

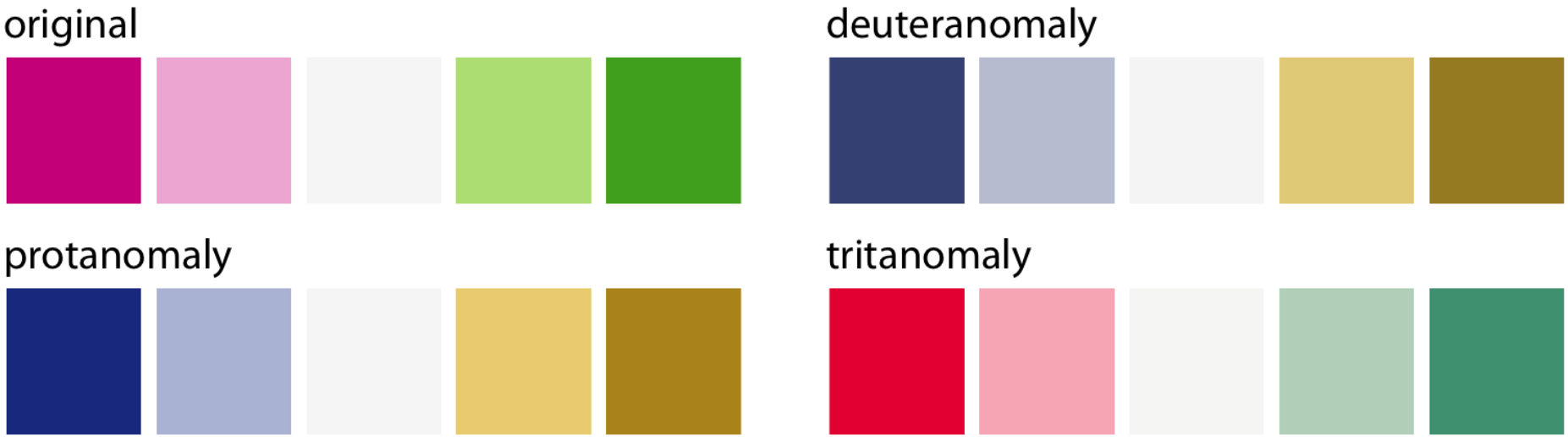
2. Be aware of color-vision deficiency (to be continued...)

5%–8% of men are color blind!



Blue-green color-vision deficiency is rare but does occur

Choose colors that can be distinguished with CVD

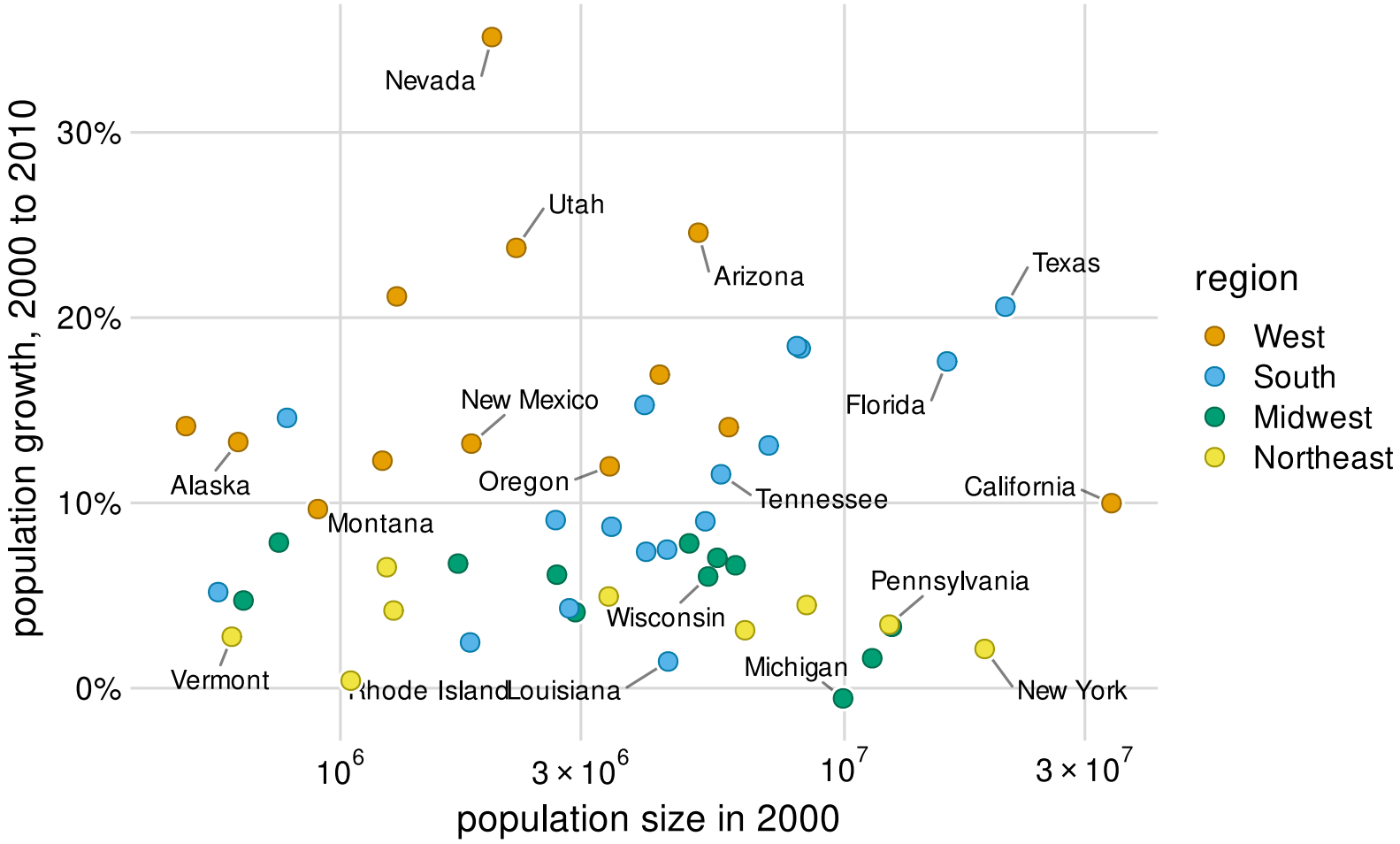


A good default for some use cases is the Okabe-Ito scale

Uses of color in data visualization

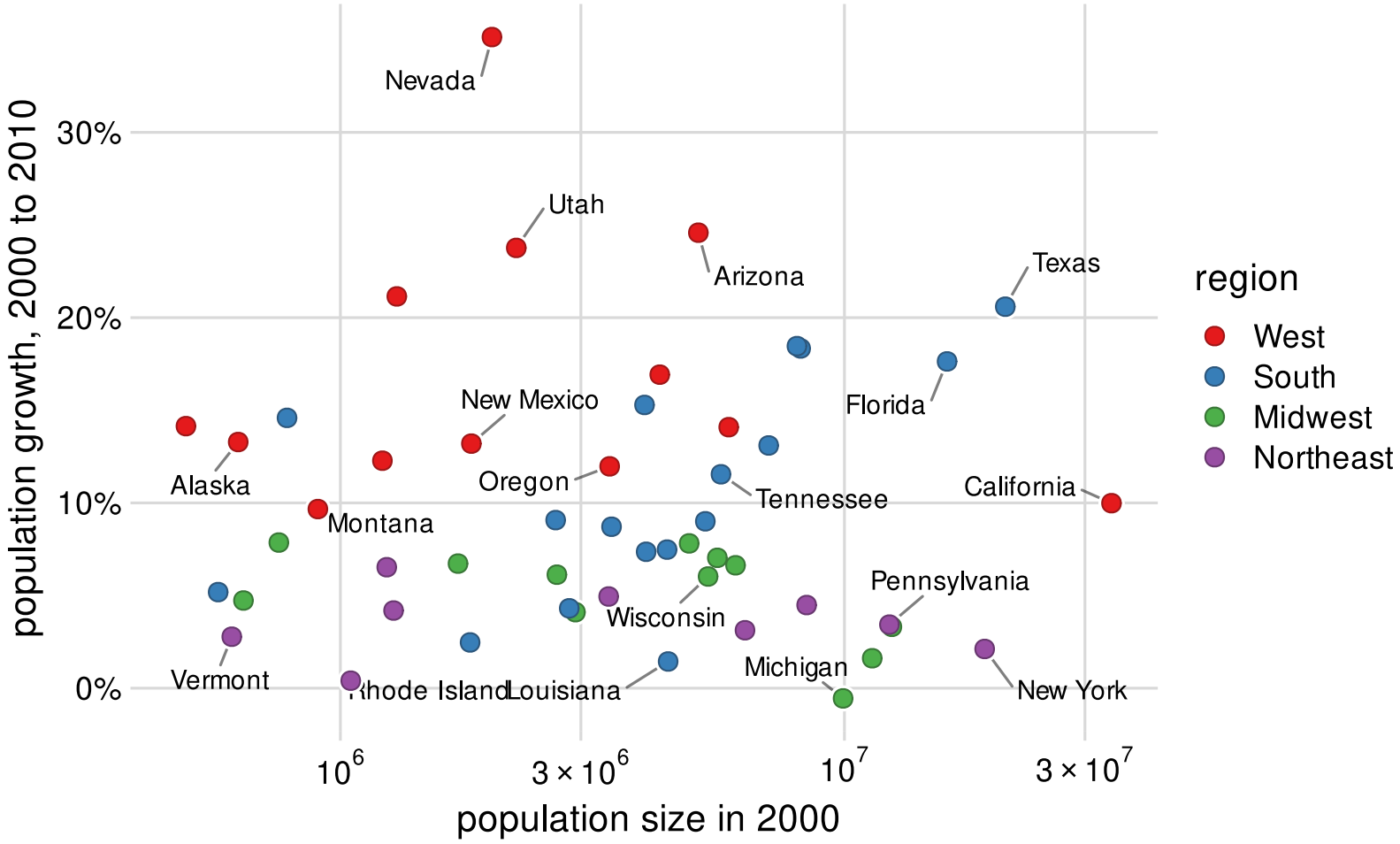
1. Distinguish categories (qualitative)

Qualitative scale example



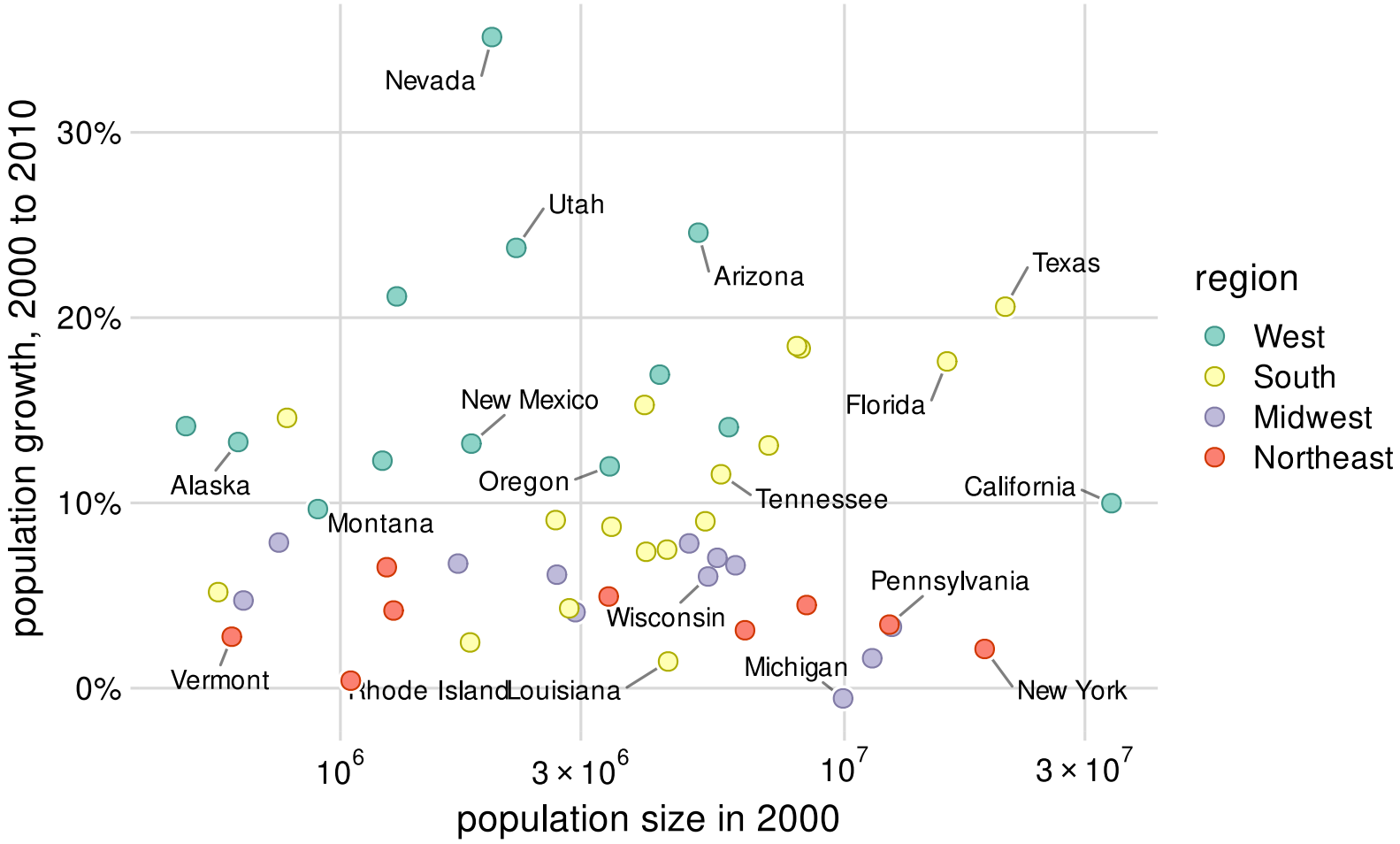
Palette name: **Okabe-Ito**

Qualitative scale example



Palette name: **ColorBrewer Set1**

Qualitative scale example



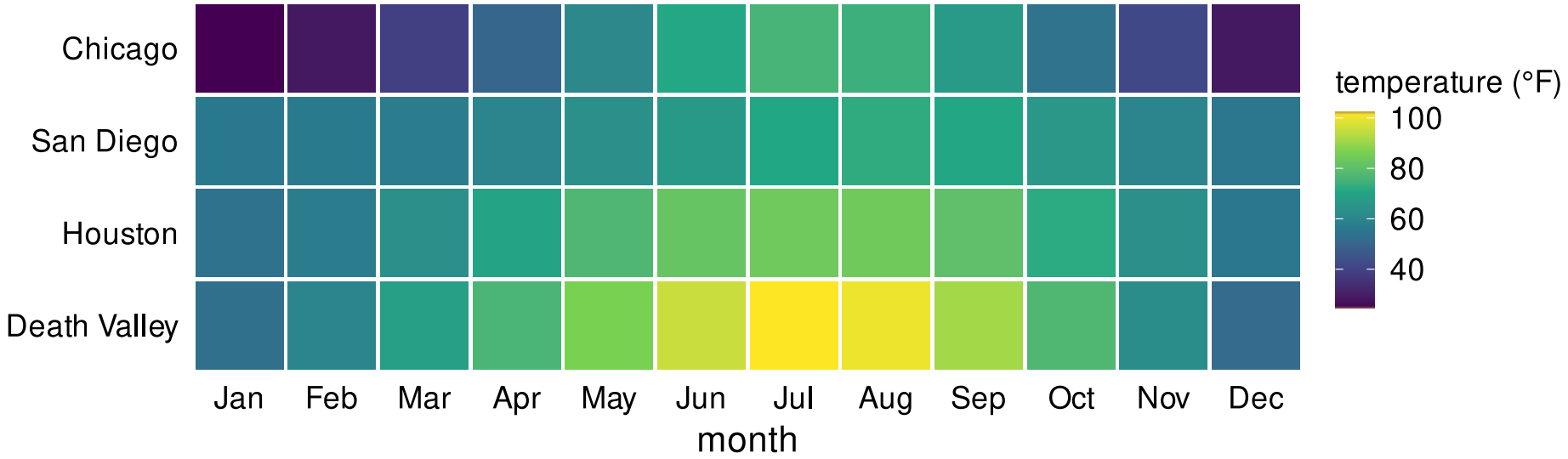
Palette name: **ColorBrewer Set3**

Uses of color in data visualization

1. Distinguish categories (qualitative)

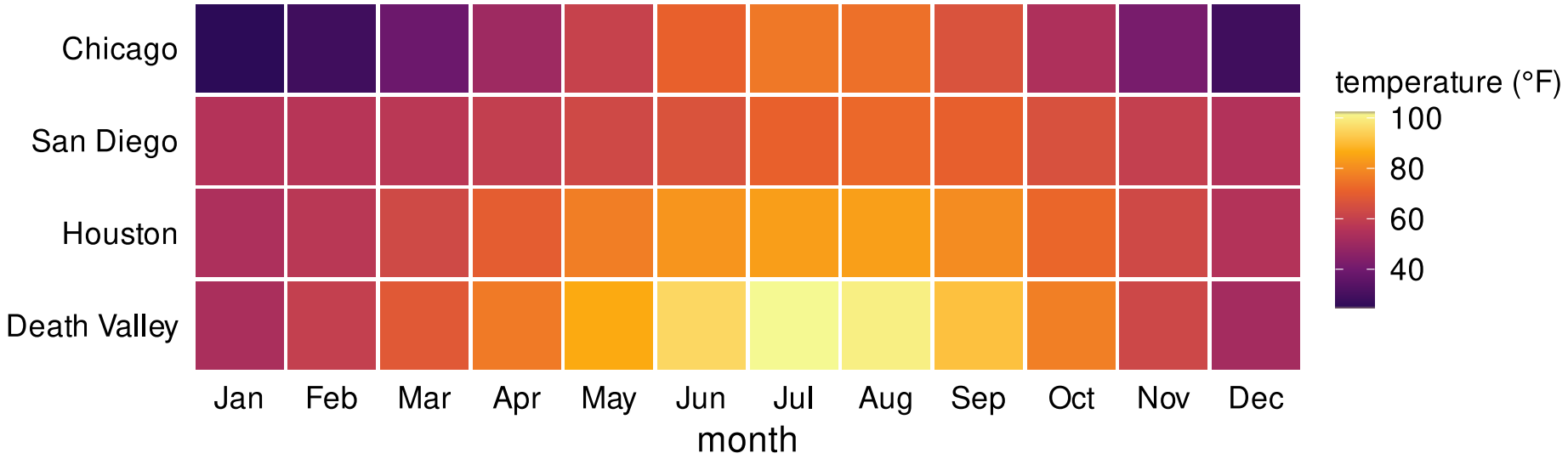
2. Represent numeric values (sequential)

Sequential scale example



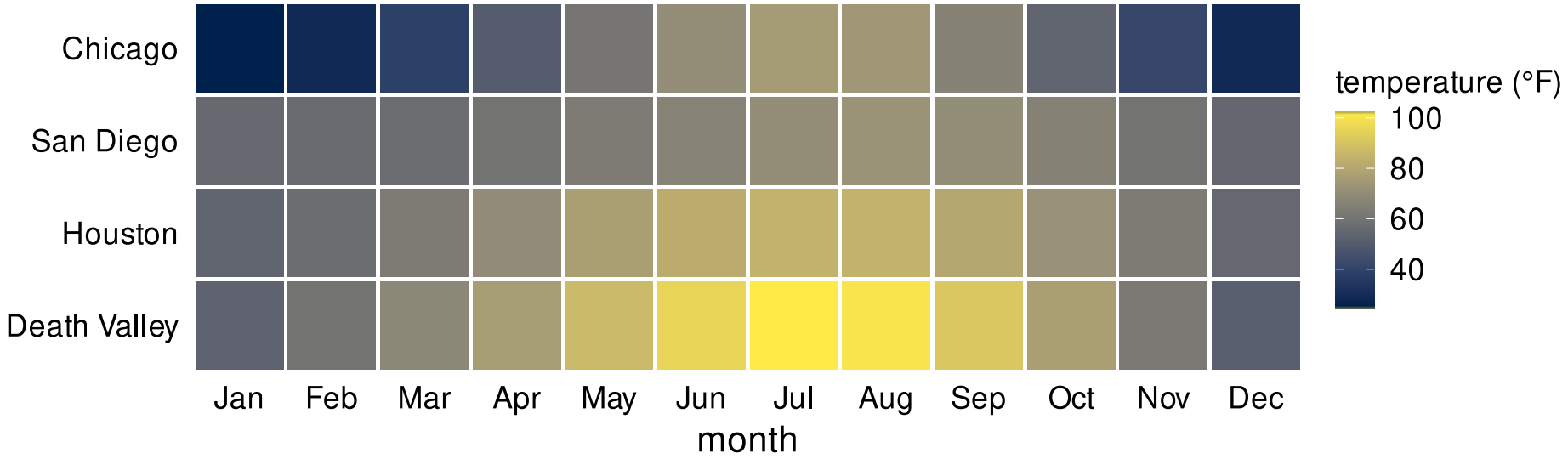
Palette name: **Viridis**

Sequential scale example



Palette name: **Inferno**

Sequential scale example



Palette name: **Cividis**

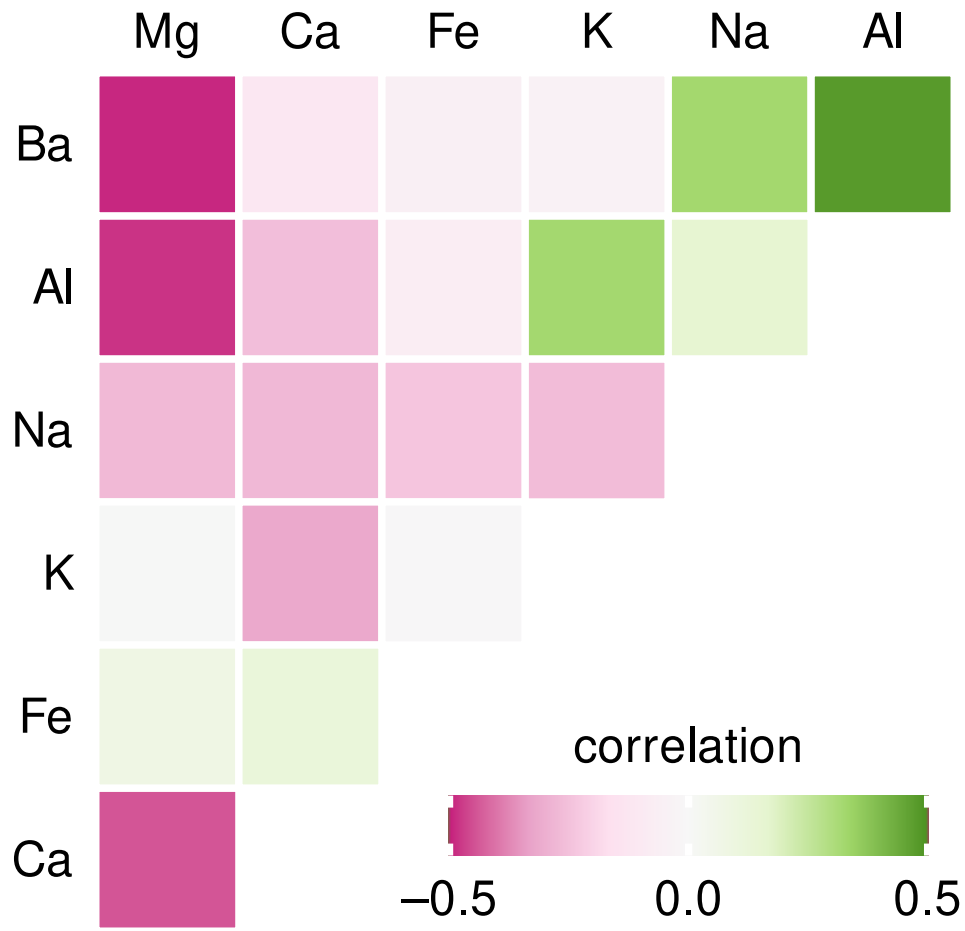
Uses of color in data visualization

1. Distinguish categories (qualitative)

2. Represent numeric values (sequential)

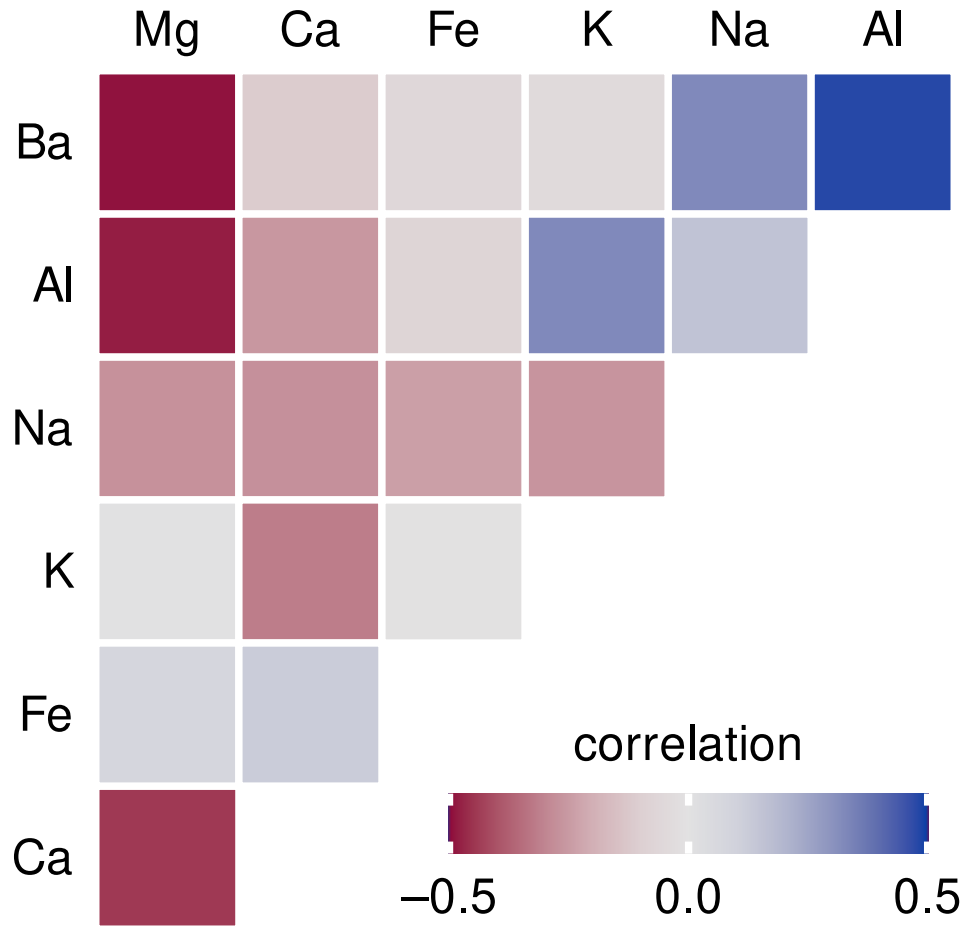
2. Represent numeric values (diverging)

Diverging scale example



Palette name: **Carto Earth**

Diverging scale example



Palette name: **Blue-Red**

Uses of color in data visualization

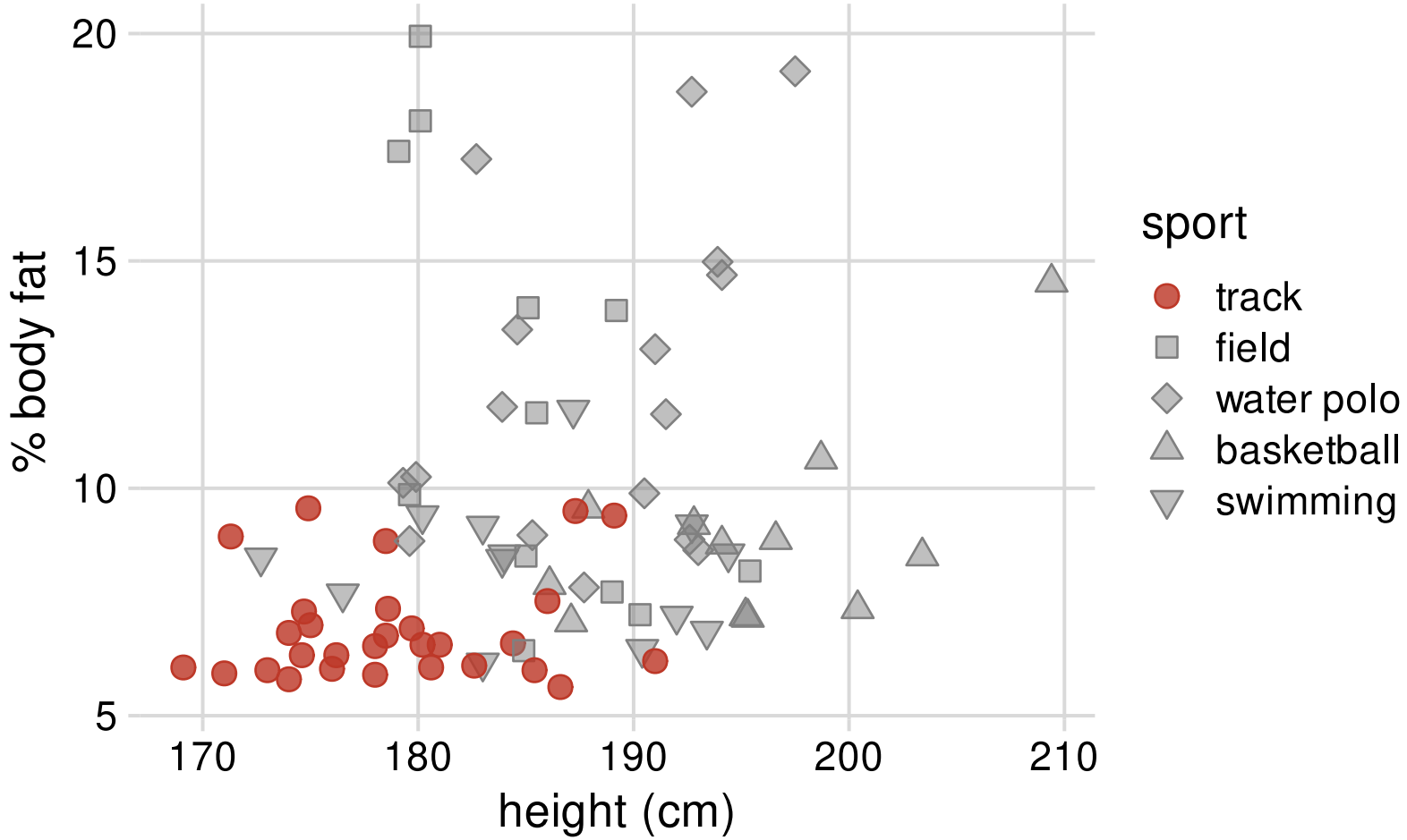
1. Distinguish categories (qualitative)

2. Represent numeric values (sequential)

3. Represent numeric values (diverging)

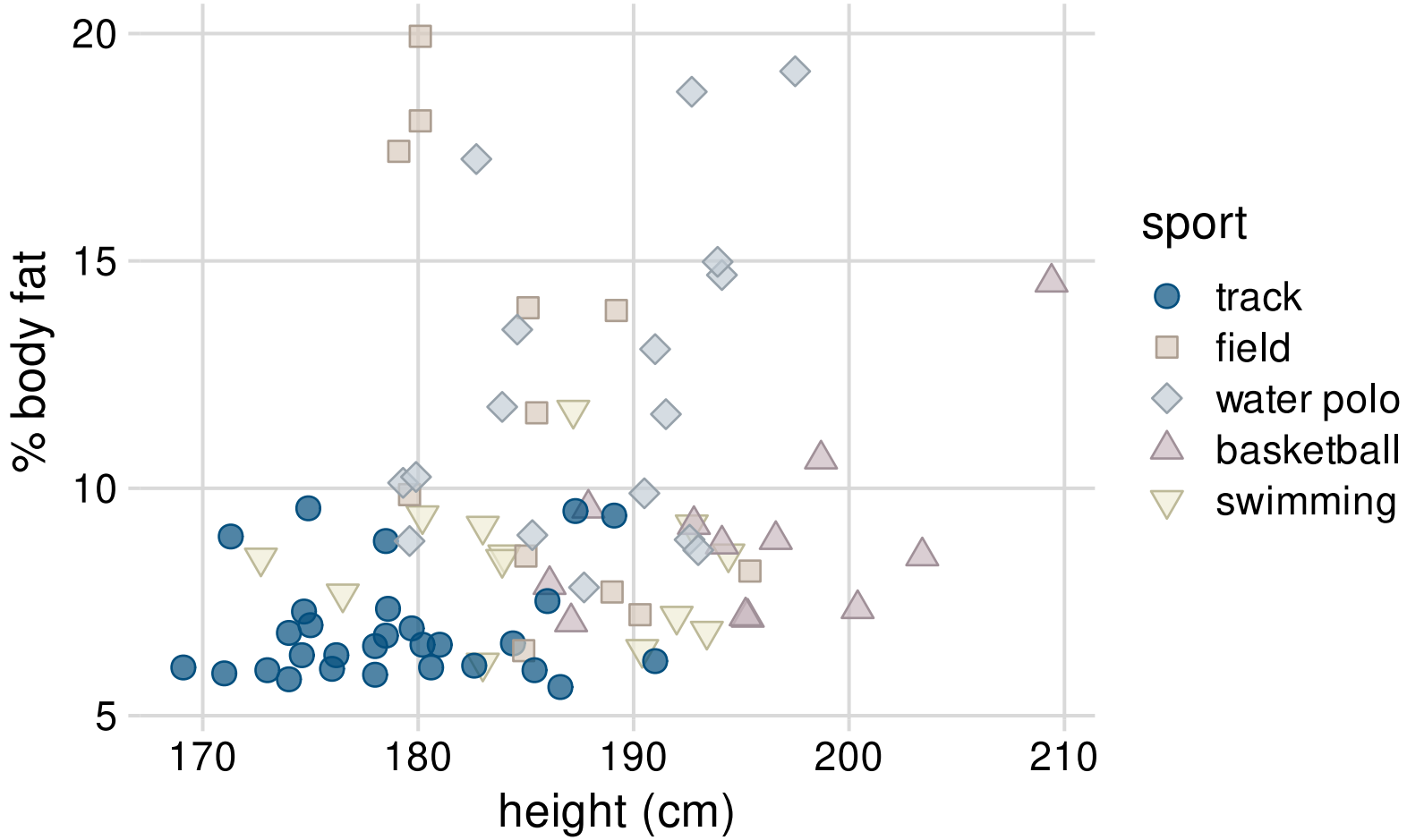
4. Highlight

Highlight example



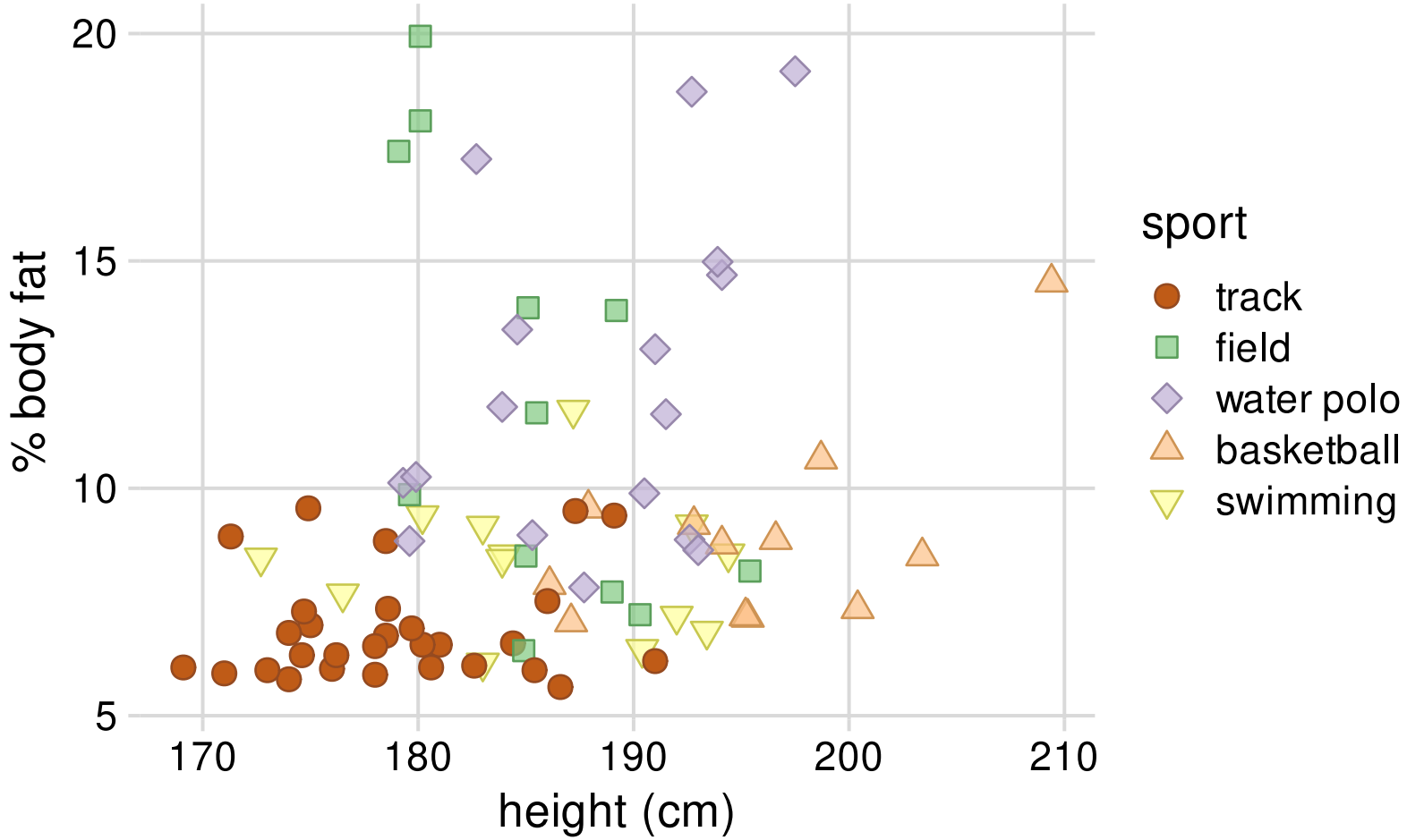
Palette: **Grays with accents**

Highlight example



Palette name: **Okabe-Ito accent**

Highlight example



Palette name: **ColorBrewer accent**

Color scales in **ggplot2**

Getting the data

The `temps_months` dataset:

```r
temps_months <-
  read_csv("https://wilkelab.org/SDS375/datasets/tempnormals.csv") %>%
  group_by(location, month_name) %>%
  summarize(mean = mean(temperature)) %>%
  mutate(
    month = factor(
      month_name,
      levels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                 "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
    ),
    location = factor(
      location, levels = c("Death Valley", "Houston",
                           "San Diego", "Chicago")
    )
  ) %>%
  select(-month_name)
```

location	mean	month
Chicago	50.45000	Apr
Chicago	74.14839	Aug
Chicago	29.00000	Dec
Chicago	28.90690	Feb
Chicago	24.84839	Jan
Chicago	75.84516	Jul
Chicago	71.00667	Jun
Chicago	38.84839	Mar
Chicago	60.90000	May
Chicago	41.55000	Nov
Chicago	54.25161	Oct
Chicago	66.40333	Sep

Getting the data

The `popgrowth` dataset:

```
US_census <-
  read_csv("https://wilkelab.org/SDS375/datasets/US_census.csv")
US_regions <-
  read_csv("https://wilkelab.org/SDS375/datasets/US_regions.csv")
popgrowth <- left_join(US_census, US_regions) %>%
    group_by(region, division, state) %>%
    summarize(
      pop2000 = sum(pop2000, na.rm = TRUE),
      pop2010 = sum(pop2010, na.rm = TRUE),
      popgrowth = (pop2010-pop2000)/pop2000,
      .groups = "drop"
    ) %>%
    mutate(
      region = factor(
        region,
        levels = c("West", "South",
                   "Midwest", "Northeast")))
```

region	division	state	pop2000	pop2010	popgrowth
Midwest	East North Central	Illinois	12419293	12830632	0.0331210
Midwest	East North Central	Indiana	6080485	6483802	0.0663297
Midwest	East North Central	Michigan	9938444	9883640	-0.0055143
Midwest	East North Central	Ohio	11353140	11536504	0.0161510
Midwest	East North Central	Wisconsin	5363675	5686986	0.0602779
Midwest	West North Central	Iowa	2926324	3046355	0.0410177
Midwest	West North Central	Kansas	2688418	2853118	0.0612628
Midwest	West North Central	Minnesota	4919479	5303925	0.0781477
Midwest	West North Central	Missouri	5595211	5988927	0.0703666
Midwest	West North Central	Nebraska	1711263	1826341	0.0672474
Midwest	West North Central	North Dakota	642200	672591	0.0473233
Midwest	West North Central	South Dakota	754844	814180	0.0786070

ggplot2 color scale functions are a bit of a mess

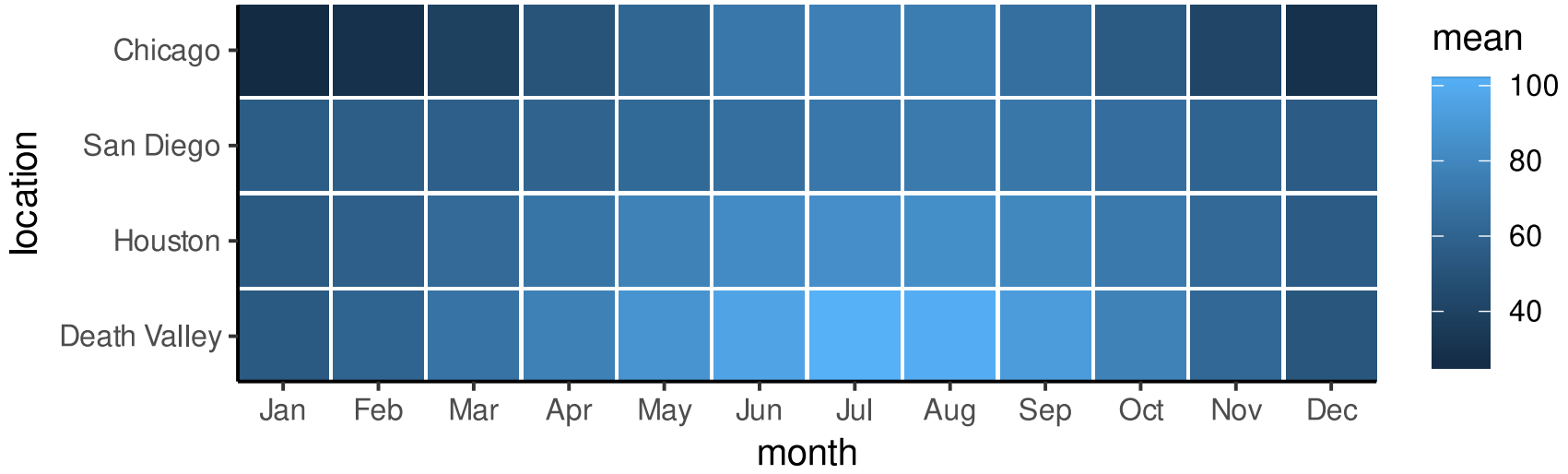
Scale function	Aesthetic	Data type	Palette type
`scale_color_hue()`	`color`	discrete	qualitative
`scale_fill_hue()`	`fill`	discrete	qualitative
`scale_color_gradient()`	`color`	continuous	sequential
`scale_color_gradient2()`	`color`	continuous	diverging

Scale function	Aesthetic	Data type	Palette type
`scale_color_hue()`	`color`	discrete	qualitative
`scale_fill_hue()`	`fill`	discrete	qualitative
`scale_color_gradient()`	`color`	continuous	sequential
`scale_color_gradient2()`	`color`	continuous	diverging
`scale_fill_viridis_c()`	`color`	continuous	sequential
`scale_fill_viridis_d()`	`fill`	discrete	sequential
`scale_color_brewer()`	`color`	discrete	qualitative, diverging, sequential
`scale_fill_brewer()`	`fill`	discrete	qualitative, diverging, sequential
`scale_color_distiller()`	`color`	continuous	qualitative, diverging, sequential

... and there are many many more

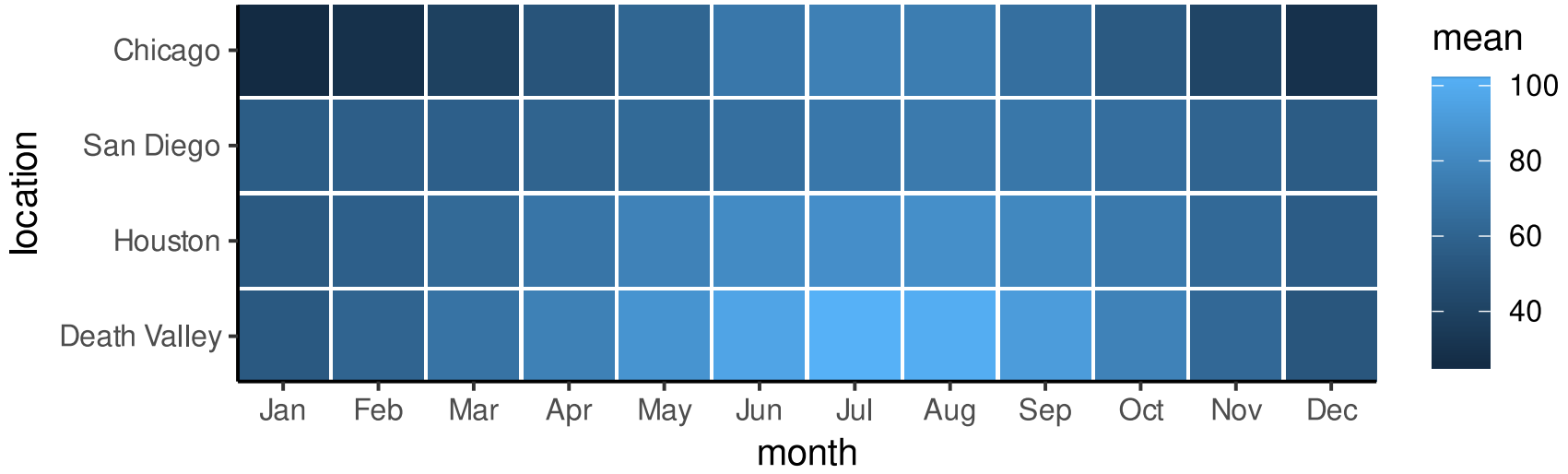
Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic()
  # no fill scale defined, default is scale_fill_gradient()
```



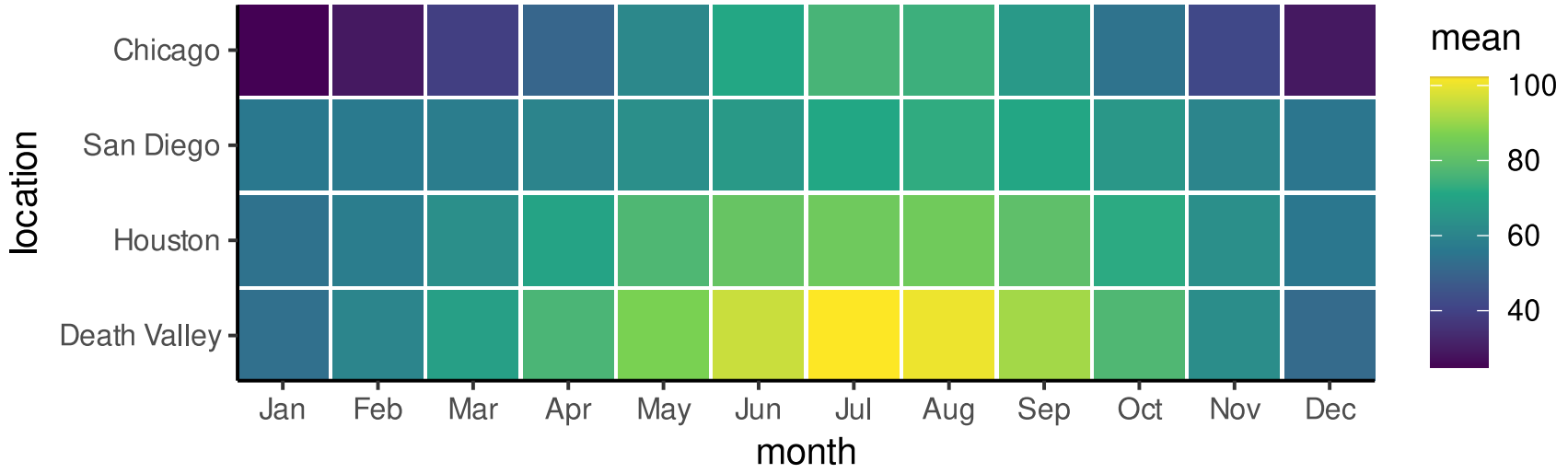
Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_gradient()
```



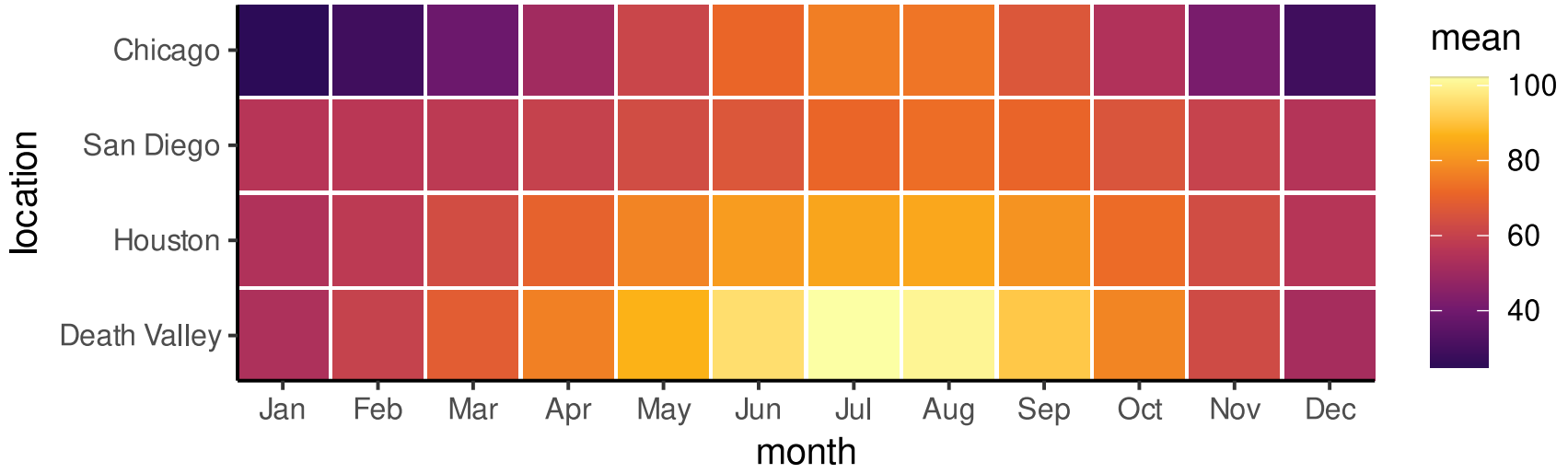
Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_viridis_c()
```



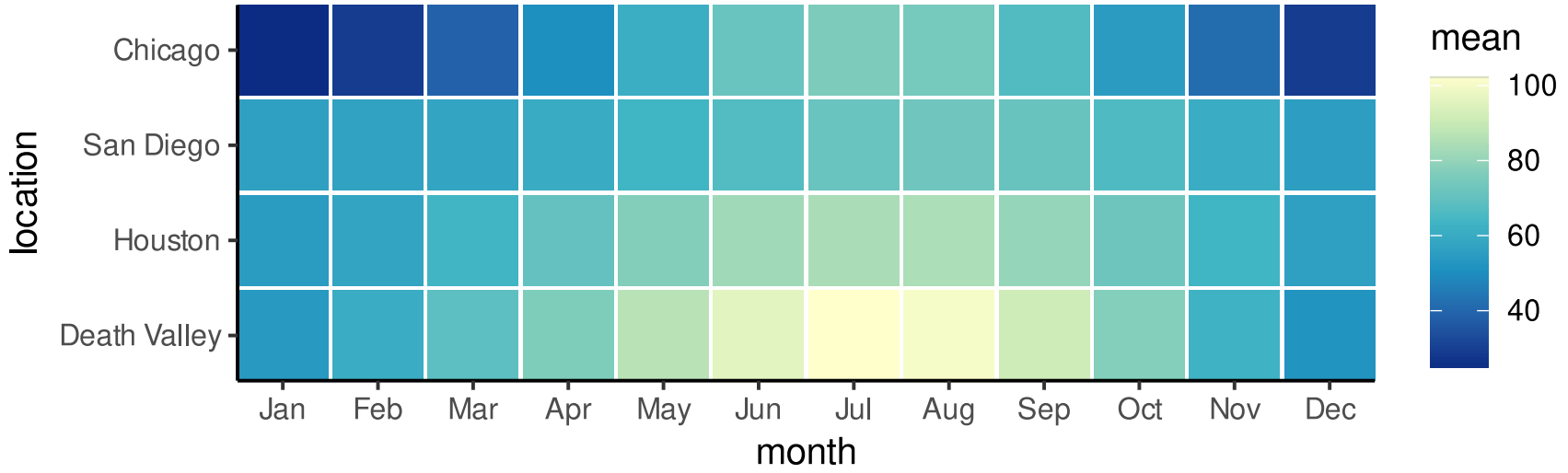
Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_viridis_c(option = "B", begin = 0.15)
```



Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_distiller(palette = "YlGnBu")
```



The **colorspace** package creates some order

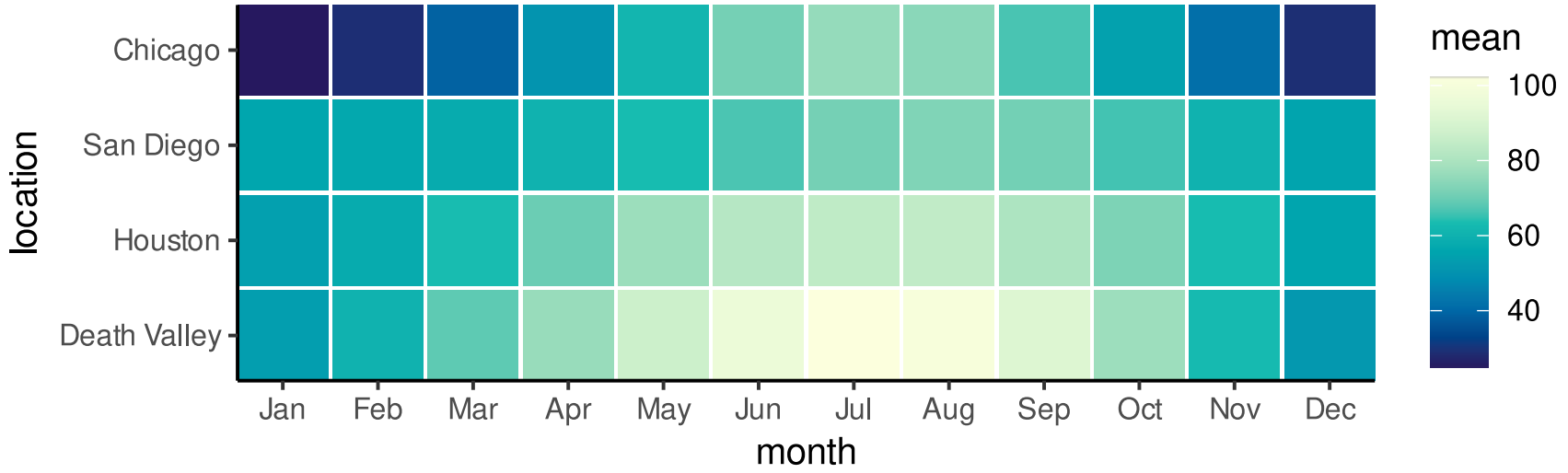
Scale name: `scale_<aesthetic>_<datatype>_<colorscale>()`

- `<aesthetic>`: name of the aesthetic (`fill`, `color`, `colour`)
- `<datatype>`: type of variable plotted (`discrete`, `continuous`, `binned`)
- `<colorscale>`: type of the color scale (`qualitative`, `sequential`, `diverging`, `divergingx`)

Scale function	Aesthetic	Data type	Palette type
`scale_color_discrete_qualitative()`	`color`	discrete	qualitative
`scale_fill_continuous_sequential()`	`fill`	continuous	sequential
`scale_colour_continous_divergingx()`	`colour`	continuous	diverging

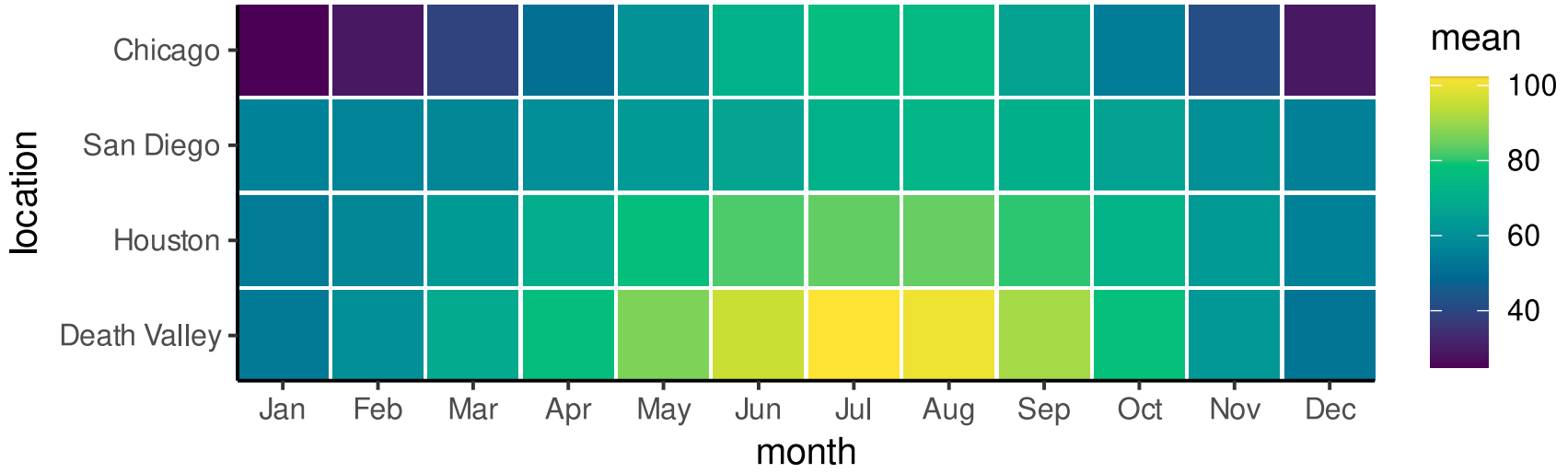
Examples

```
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_continuous_sequential(palette = "YlGnBu", rev = FALSE)
```



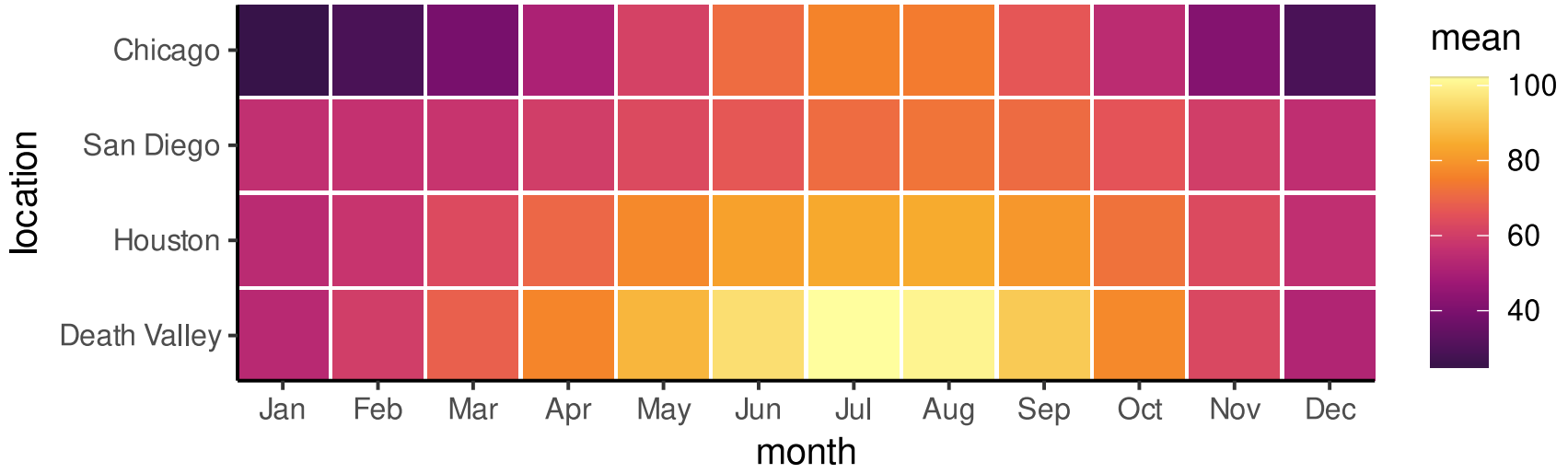
Examples

```r
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_continuous_sequential(palette = "Viridis", rev = FALSE)
```



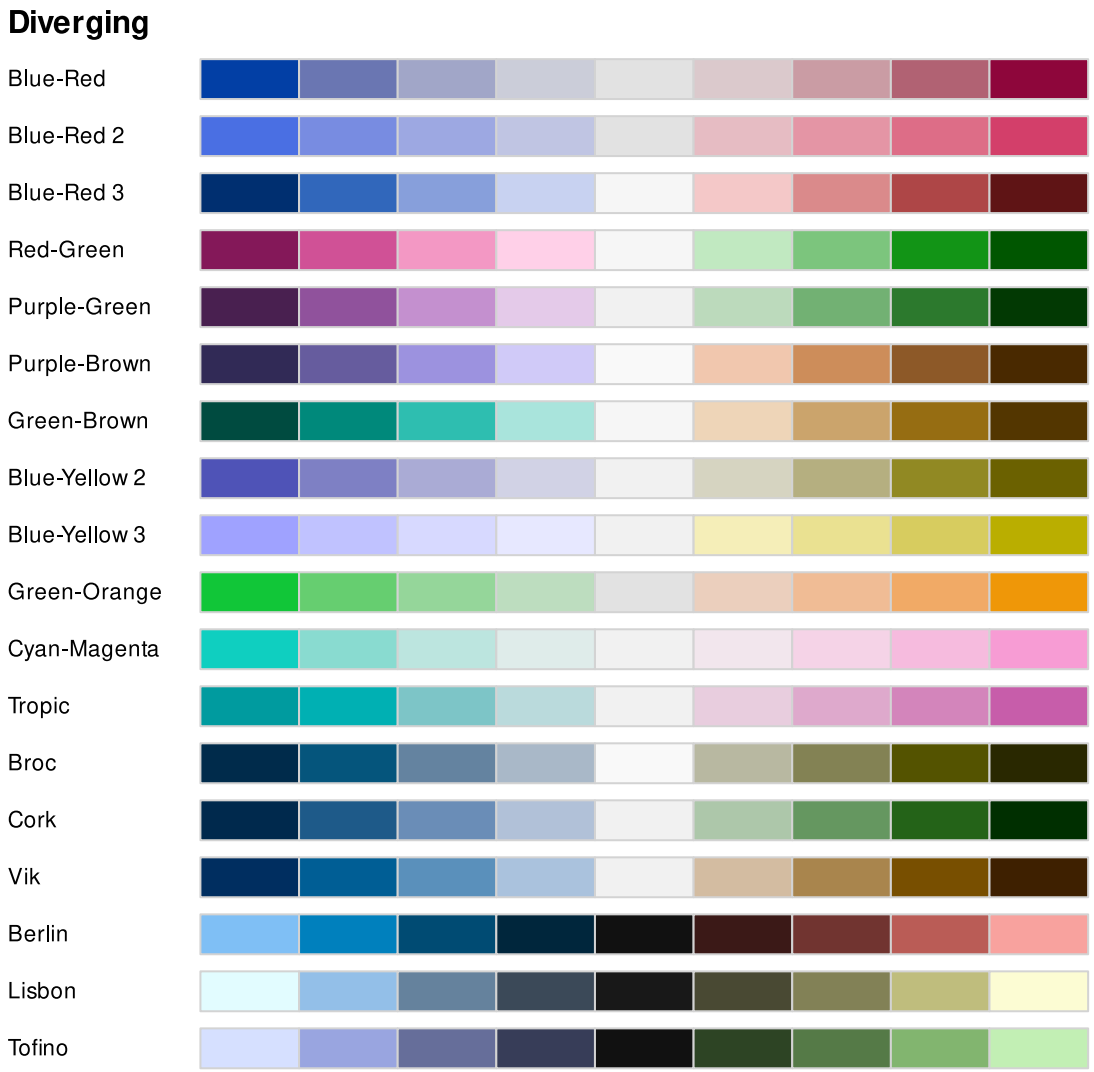
Examples

```r
ggplot(temps_months, aes(x = month, y = location, fill = mean)) +
  geom_tile(width = 0.95, height = 0.95) +
  coord_fixed(expand = FALSE) +
  theme_classic() +
  scale_fill_continuous_sequential(palette = "Inferno", begin = 0.15, rev = FALSE)
```

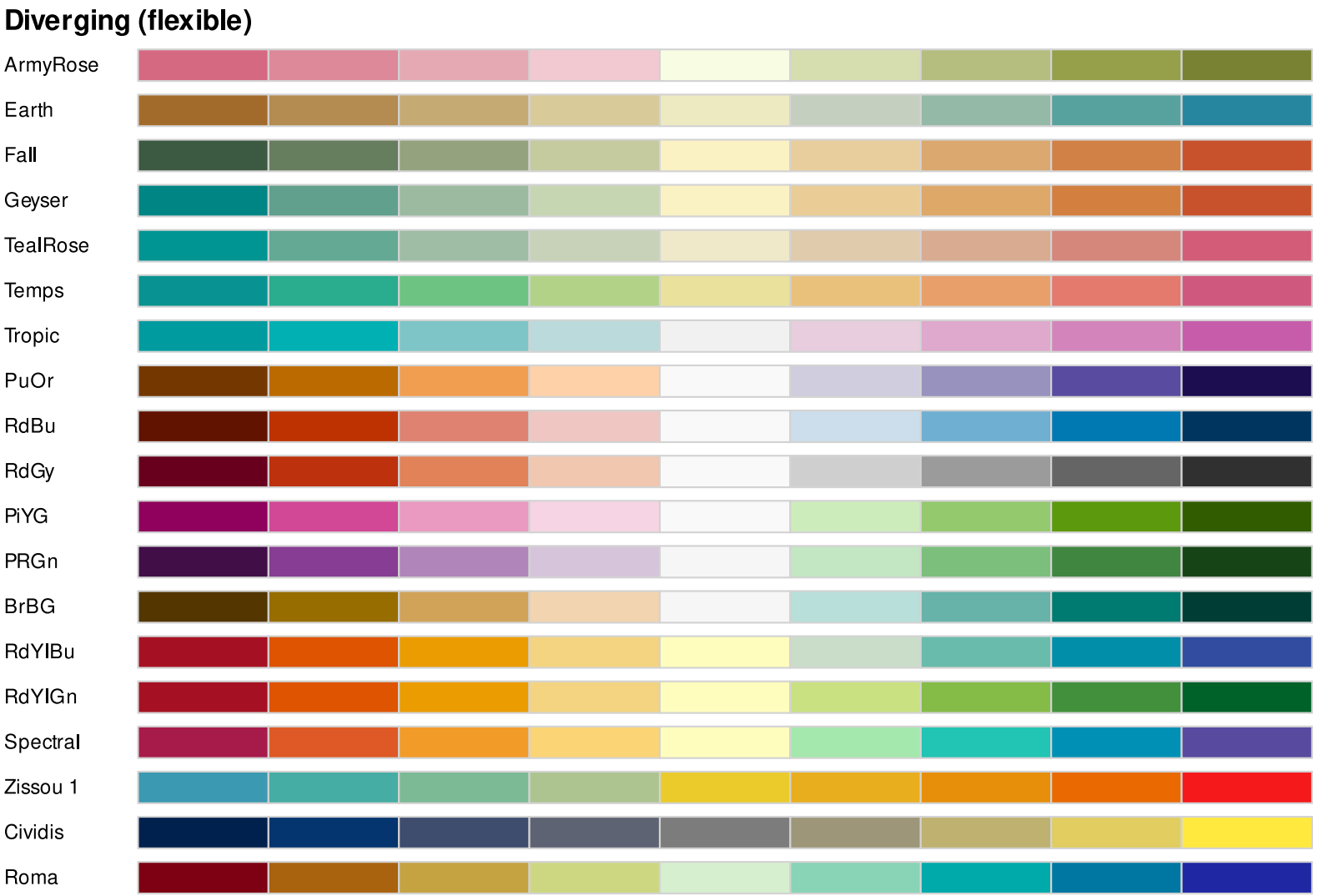


```r
colorspace::hcl_palettes(type = "sequential", plot = TRUE) # all sequential palettes
```

```r
colorspace::hcl_palettes(type = "diverging", plot = TRUE, n = 9) # all diverging palettes
```



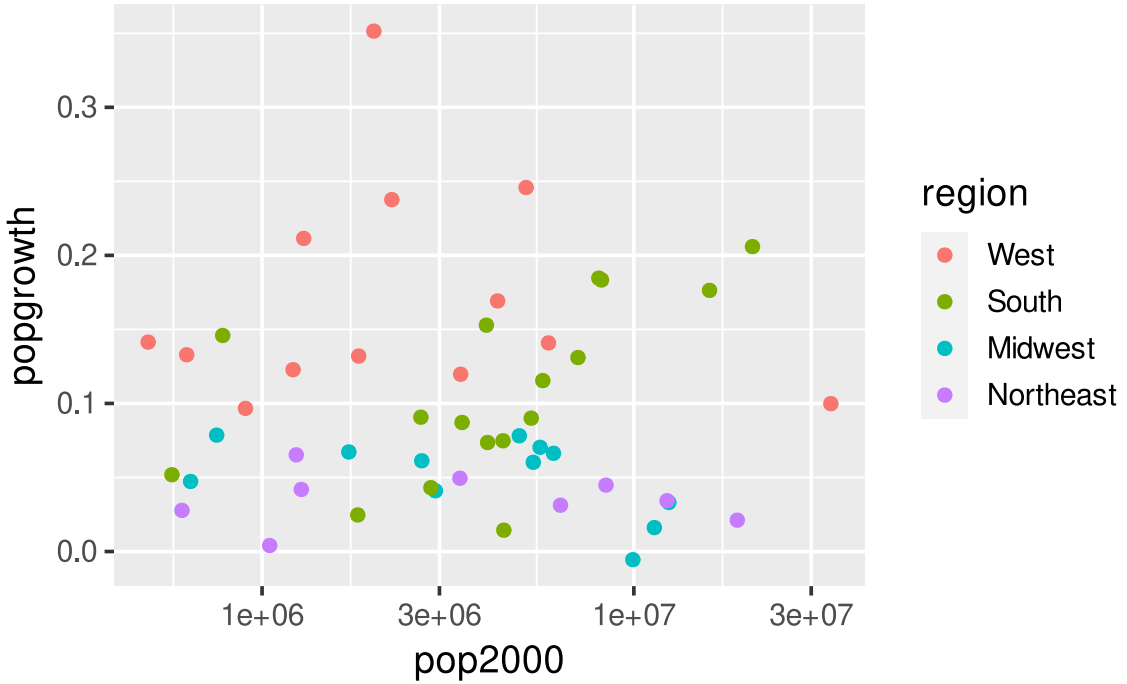
```r
colorspace::divergingx_palettes(plot = TRUE, n = 9) # all divergingx palettes
```



Setting colors manually for discrete, qualitative scales

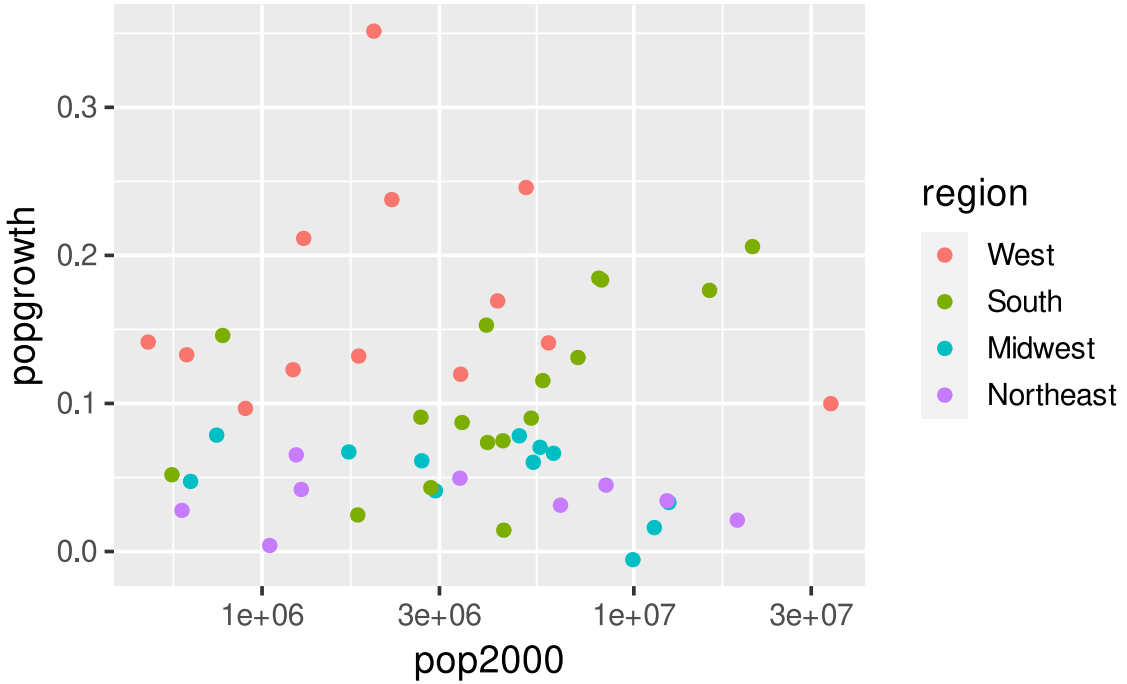
Discrete, qualitative scales are best set manually

```
1 ggplot(popgrowth, aes(x = pop2000, y = popgrowth, color = region)) +
2   geom_point() +
3   scale_x_log10()
4   # no color scale defined, default is scale_color_hue()
```



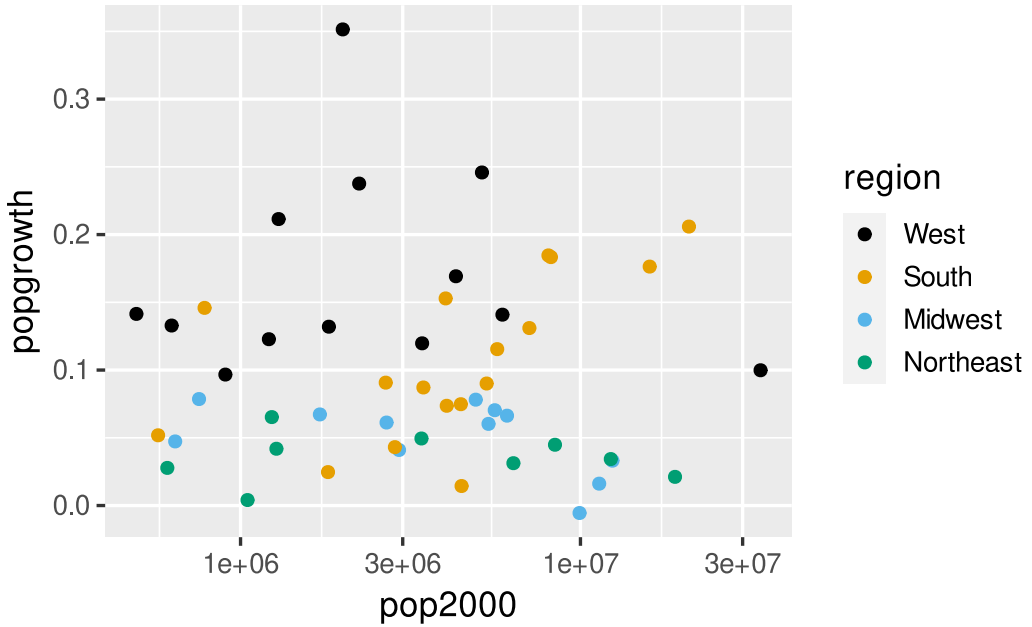
Discrete, qualitative scales are best set manually

```
1 ggplot(popgrowth, aes(x = pop2000, y = popgrowth, color = region)) +
2   geom_point() +
3   scale_x_log10() +
4   scale_color_hue()
```



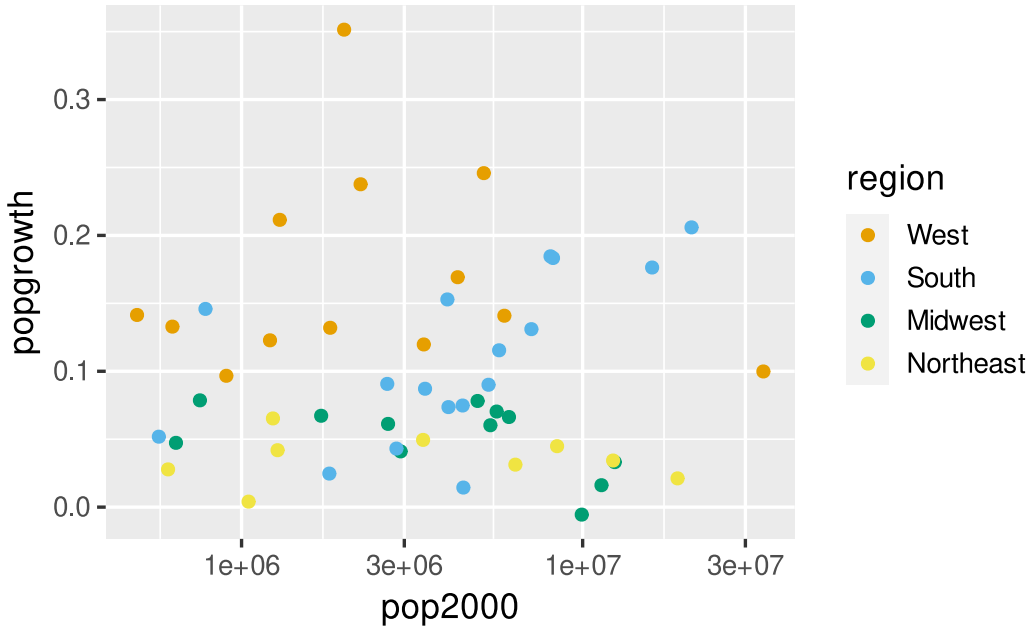
Discrete, qualitative scales are best set manually

```
1 library(ggthemes)  # for scale_color_colorblind()
2
3 ggplot(popgrowth, aes(x = pop2000, y = popgrowth, color = region)) +
4   geom_point() +
5   scale_x_log10() +
6   scale_color_colorblind()  # uses Okabe-Ito colors
```



Discrete, qualitative scales are best set manually

```
ggplot(popgrowth, aes(x = pop2000, y = popgrowth, color = region)) +
  geom_point() +
  scale_x_log10() +
  scale_color_manual(
    values = c(West = "#E69F00", South = "#56B4E9",
               Midwest = "#009E73", Northeast = "#F0E442")
  )
```



Okabe-Ito RGB codes

Name	Hex code	R, G, B (0-255)
orange	#E69F00	230, 159, 0
sky blue	#56B4E9	86, 180, 233
bluish green	#009E73	0, 158, 115
yellow	#F0E442	240, 228, 66
blue	#0072B2	0, 114, 178
vermilion	#D55E00	213, 94, 0
reddish purple	#CC79A7	204, 121, 167
black	#000000	0, 0, 0

Designing for color-vision deficiency

People with impaired color vision are not literally unable to see any colors

Instead, they will typically have difficulty to distinguish certain types of colors, for example red and green (red–green CVD) or blue and green (blue–yellow CVD)

The technical terms for these deficiencies are **deuteranomaly/deuteranopia** and **protanomaly/protanopia** for the red–green variant (where people have difficulty perceiving either green or red, respectively) and **tritanomaly/tritanopia** for the blue–yellow variant (where people have difficulty perceiving blue)

Designing for color-vision deficiency

The terms ending in “anomaly” refer to some impairment in the perception of the respective color, and the terms ending in “anopia” refer to complete absence of perception of that color

Approximately 8% of males and 0.5% of females suffer from some sort of color-vision deficiency, and deuteranomaly is the most common form whereas tritanomaly is relatively rare

CVD and type of scale

Let's look at sequential scales, diverging scales and qualitative scales

Of these three, sequential scales will generally not cause any problems for people with CVD, since a properly designed sequential scale should present a continuous gradient from dark to light colors

original

deuteranomaly

protanomaly

tritanomaly

CVD and type of scale

Things become more complicated for diverging scales, because popular color contrasts can become indistinguishable under CVD

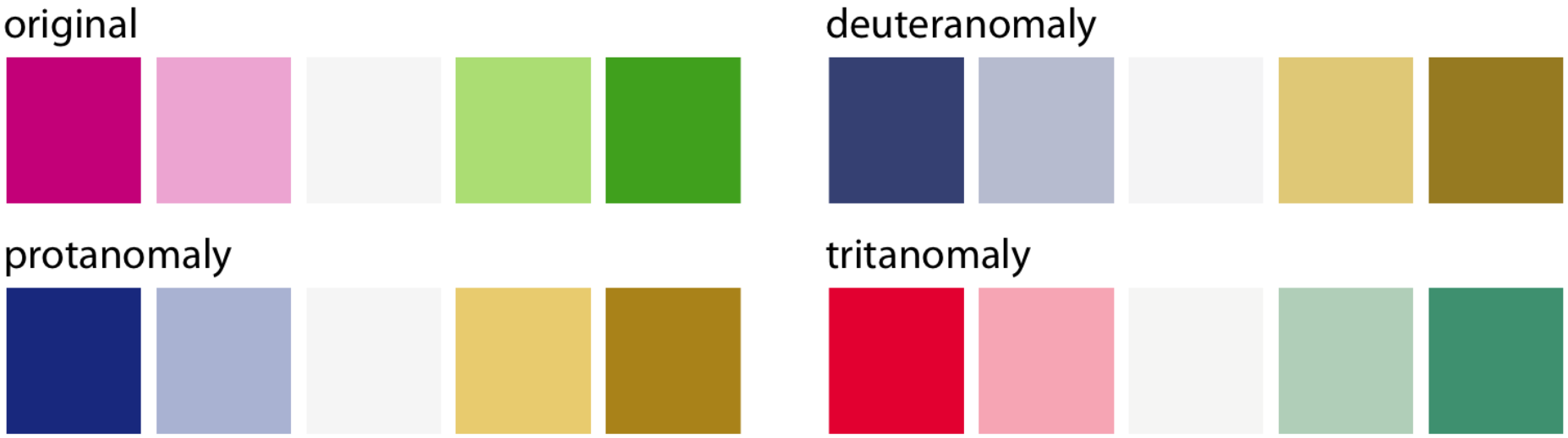
In particular, the colors red and green provide about the strongest contrast for people with normal color vision but become nearly indistinguishable for deutans or protans

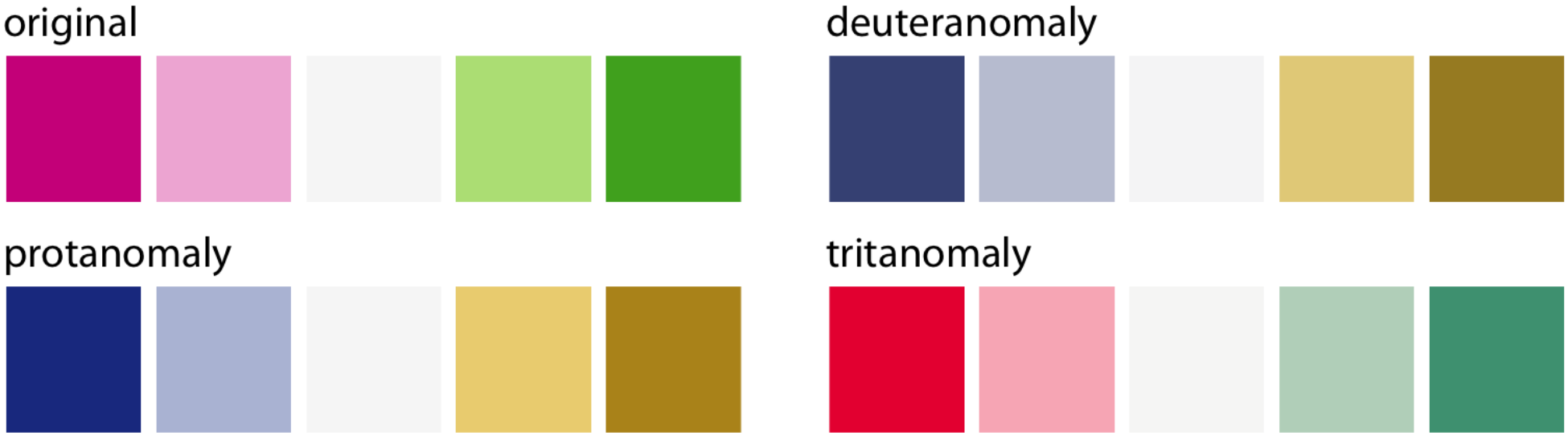
Similarly, blue-green contrasts are visible for deutans and protans but become indistinguishable for tritans

It might seem that it is nearly impossible to find two contrasting colors that are safe under all forms of CVD

However, the situation is not that dire. It is often possible to make slight modifications to the colors such that they have the desired character while also being safe for CVD

For example, the ColorBrewer PiYG (pink to yellow-green) scale looks red–green to people with normal color vision yet remains distinguishable for people with CVD





It works because the reddish color is actually pink (a mix of red and blue) while the greenish color also contains yellow

The difference in the blue component between the two colors can be picked up even by deutans or protans, and the difference in the red component can be picked up by tritans.

CVD and type of scale

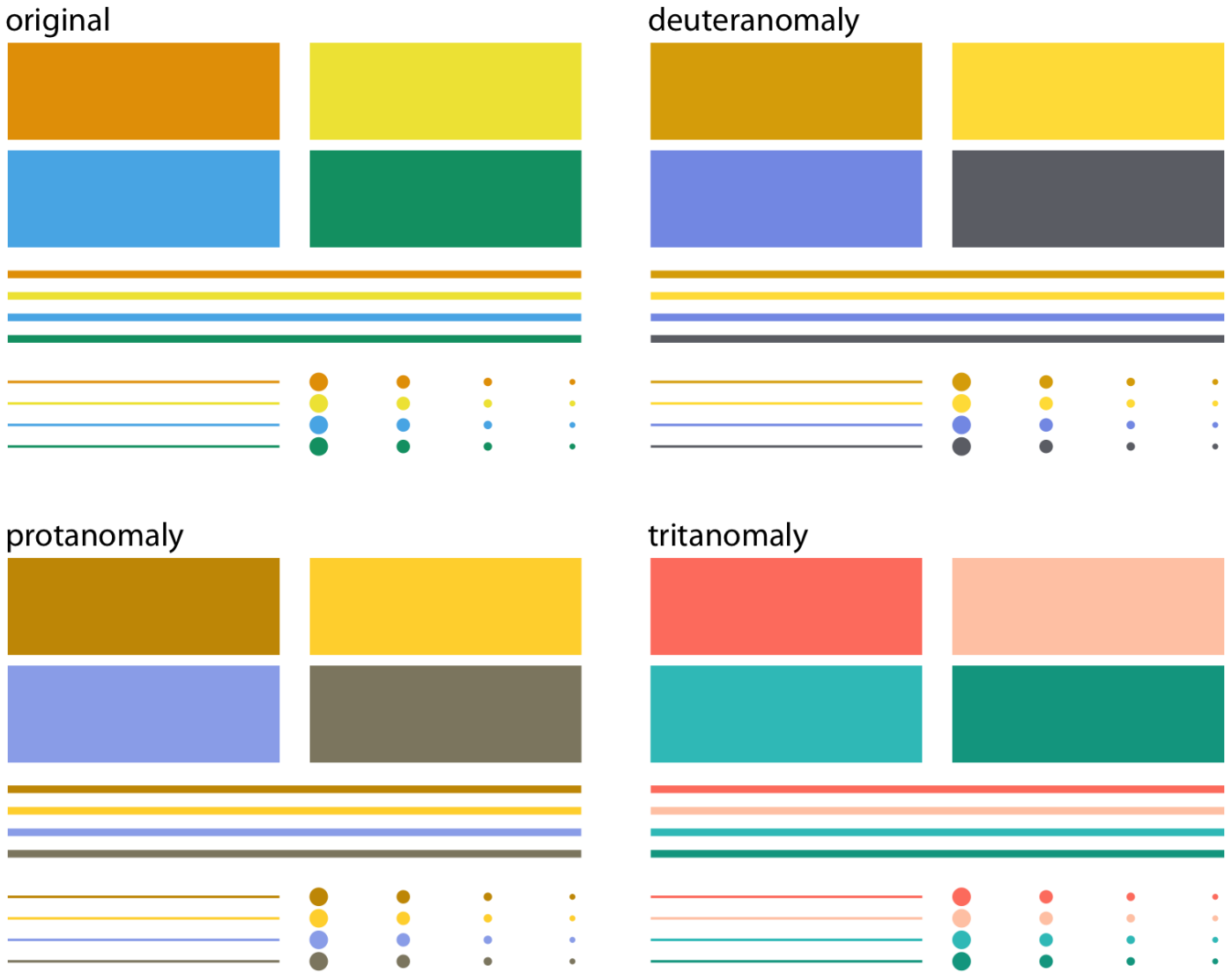
Things are most complicated for qualitative scales, because there we need many different colors and they all need to be distinguishable from each other under all forms of CVD

Just resort to using a qualitative color scale which was developed specifically to address this challenge, preferably Okabe-Ito

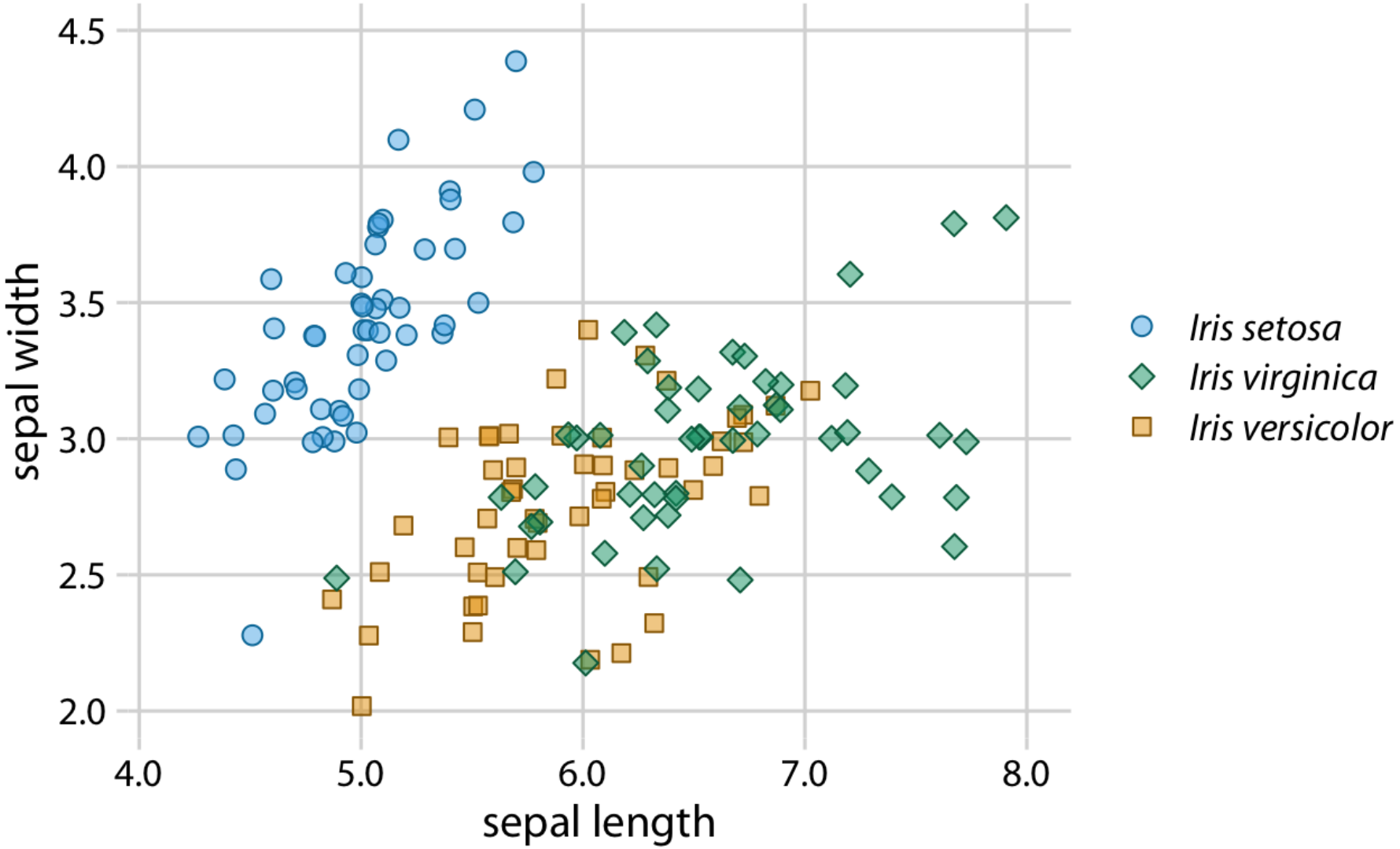
By providing eight different colors, the palette works for nearly any scenario with discrete colors

(You should probably not color-code more than eight different items in a plot anyways...)

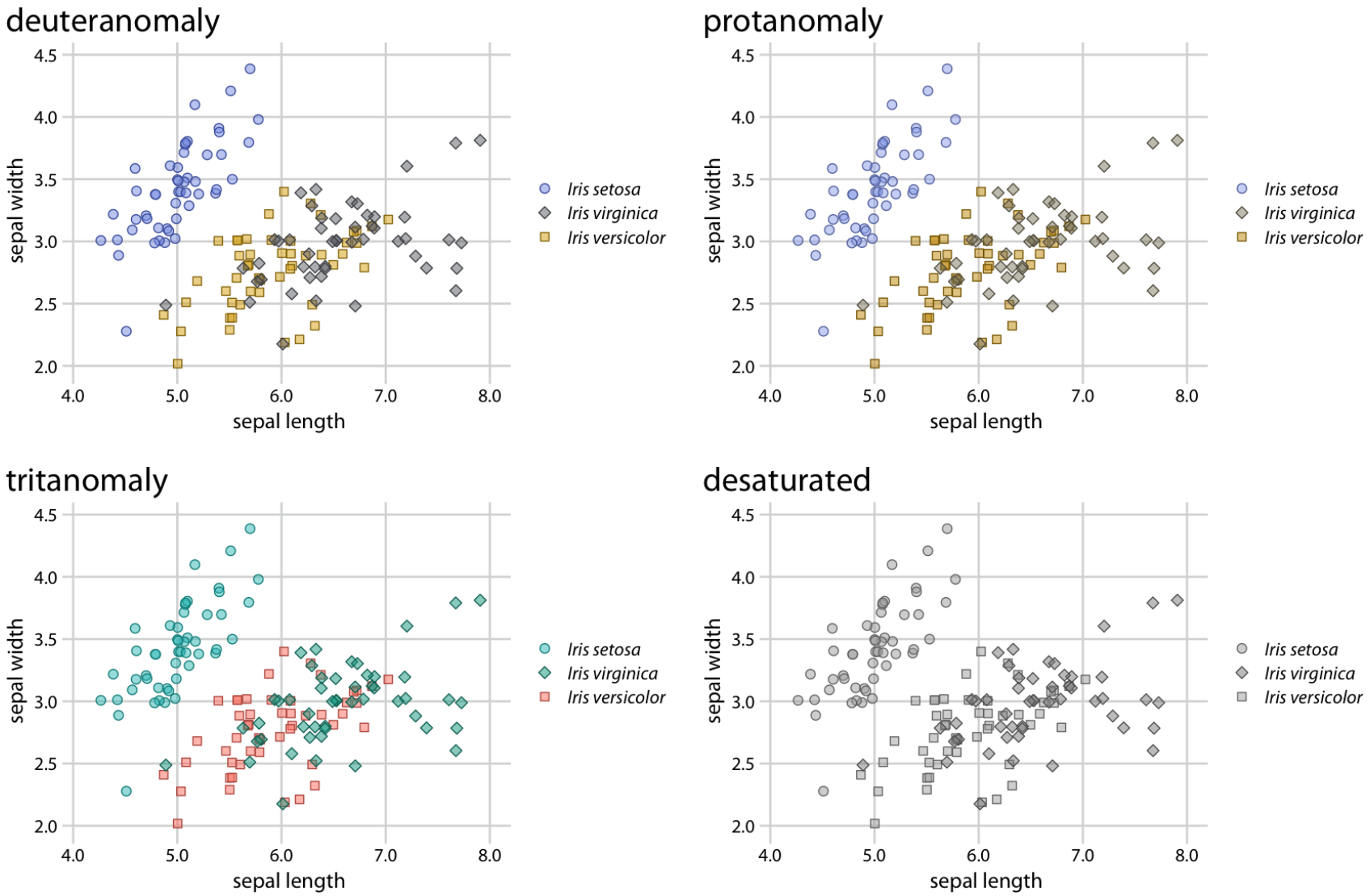
CVD is worse for thin lines and tiny dots



When in doubt, run CVD simulations

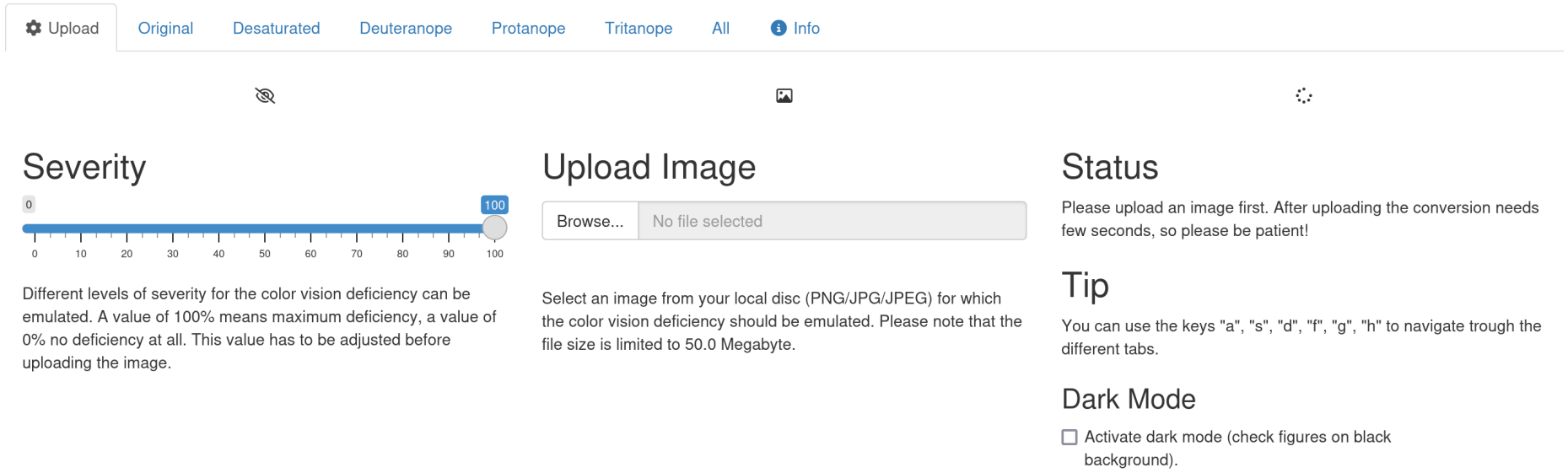


When in doubt, run CVD simulations



Emulator

```
1 colorspace::cvd_emulator()
```



And other online tools, such as:

http://hclwizard.org:3000/cvdemulator/

Notes on ggplot2 in Python

The code and the output is mostly the same with `plotnine` (some parts can be still missing unfortunately)

But in any case, there are small things you have to watch out for:

Multi-line code should be wrapped into parentheses

Some peculiarities, e.g. `shape = 21` needs to be removed in `geom_point()` because the "marks" in `plotnine` are encoded a little bit differently

Notes on ggplot2 in Python

All aesthetics need to be in quotation marks

In `theme()`, all "." need to be replaced by "_"

The Python style guide usually dictates that operators like + go to the front of the line and not the end (but that works either way since everything is enclosed in parentheses)

By default the background of the plot is transparent in plotnine and you will have to set the background to white manually should you need that

Further Ressources on colors

About defaults for visualization libraries: “A Better Default Colormap for Matplotlib | SciPy 2015 | Nathaniel Smith and Stéfan van der Walt”

https://www.youtube.com/watch?v=xAoljeRJ3lU

Your Friendly Guide to Colors in Data Visualisation:
https://blog.datawrapper.de/colorguide/

Acknowledgements

https://www.youtube.com/watch?v=_2LLXnUdUIc

https://wilkelab.org/SDS375/

https://wilkelab.org/SDS375/slides/color-spaces.html

https://wilkelab.org/DSC385/

https://wilkelab.org/dataviz_shortcourse/

https://alberts-newsletter.beehiiv.com/subscribe

https://www.pexels.com/photo/super-mario-and-yoshi-plastic-figure-163077/

https://www.pexels.com/photo/kitchen-island-2089698/

Acknowledgements

Fundamentals of Data Visualization: Chapter 4: Color scales

Fundamentals of Data Visualization: Figure 19.10: Okabe-Ito color palette

ggplot2 book: Colour scales and legends

ggplot2 reference documentation: Scales

colorspace package: HCL-Based Color Scales for ggplot2