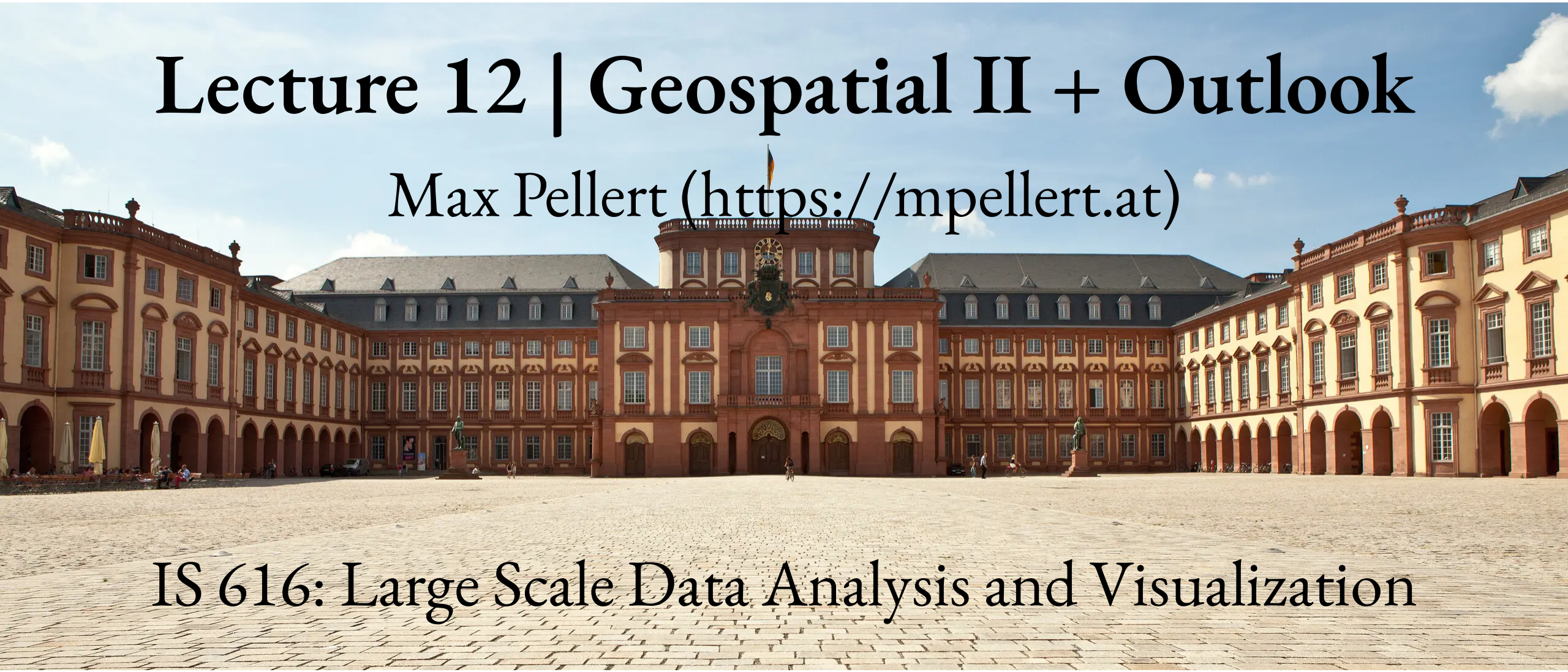
Lecture 12 | Geospatial II + Outlook

Max Pellert (https://mpellert.at)



IS 616: Large Scale Data Analysis and Visualization

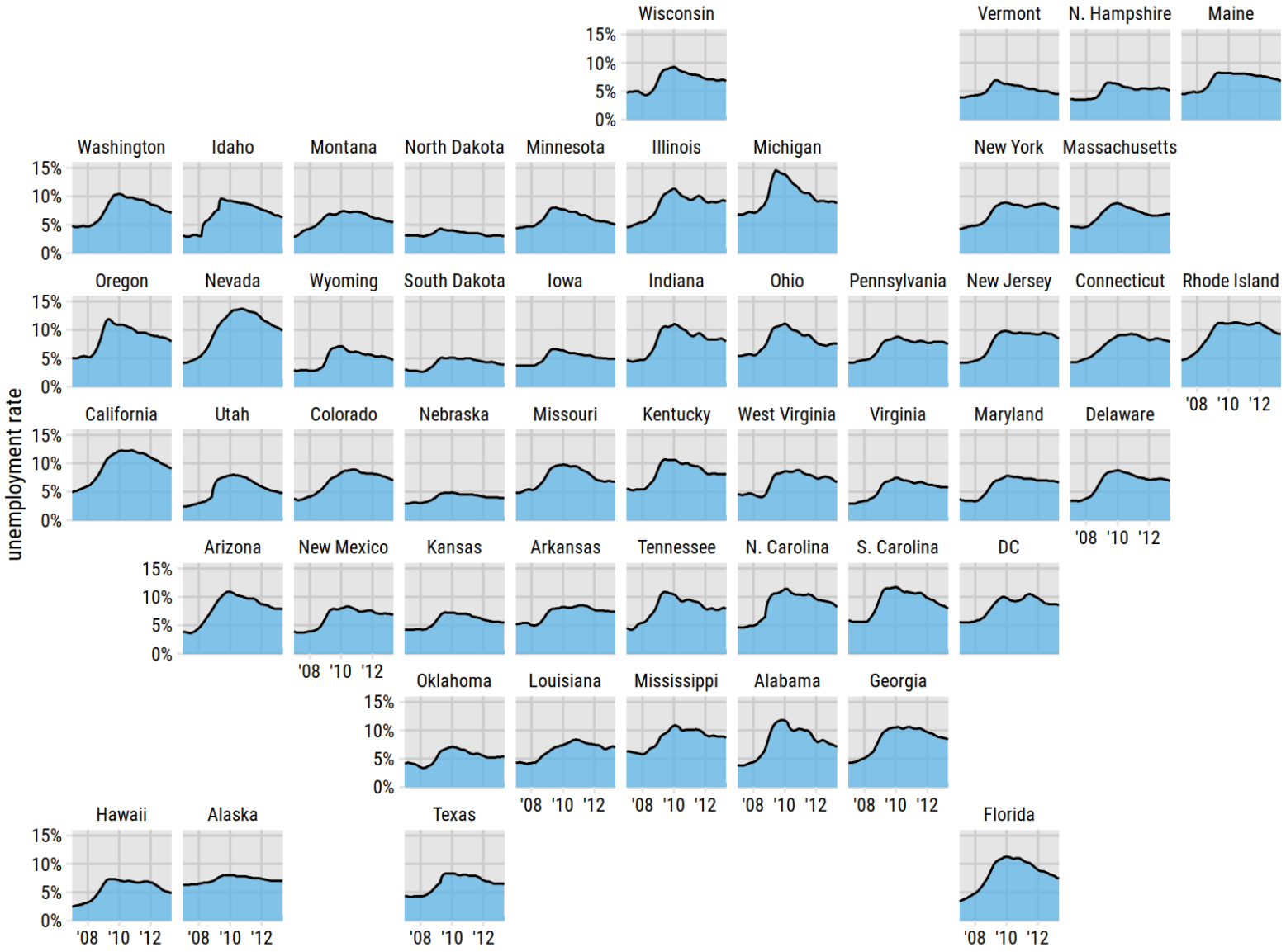
Let's complete the cartogram from last time

We want to draw more complex cartograms by placing individual plots at the location of each state of the US

As an example, if we want to visualize the evolution of the unemployment rate over time for each state, it can help to draw an individual graph for each state and then arrange the graphs based on the approximate relative position of the states to each other.

```r
library(ggplot2)
library(dplyr)
library(lubridate)
library(geofacet)

library(extrafont)

# download.file("https://github.com/clauswilke/dviz.supp/raw/1a510b08a3af75717ab5fc8121ad3fd2de

load("house_prices.rda")

dviz_font_family_condensed <- "Roboto Condensed"

theme_dviz_grid <- function(font_size = 14, font_family = dviz_font_family, line_size = .5,
                            rel_small = 12/14, rel_tiny = 11/14, rel_large = 16/14,
                            colour = "grey90") {
  half_line <- font_size / 2

  cowplot::theme_minimal_grid(font_size = font_size, font_family = font_family, line_size = lin
                              rel_small = rel_small, rel_tiny = rel_tiny, rel_large = rel_large
                              colour = colour)  %+replace%
    theme(
      plot.margin = margin(half_line/2, 1.5, half_line/2, 1.5),
```

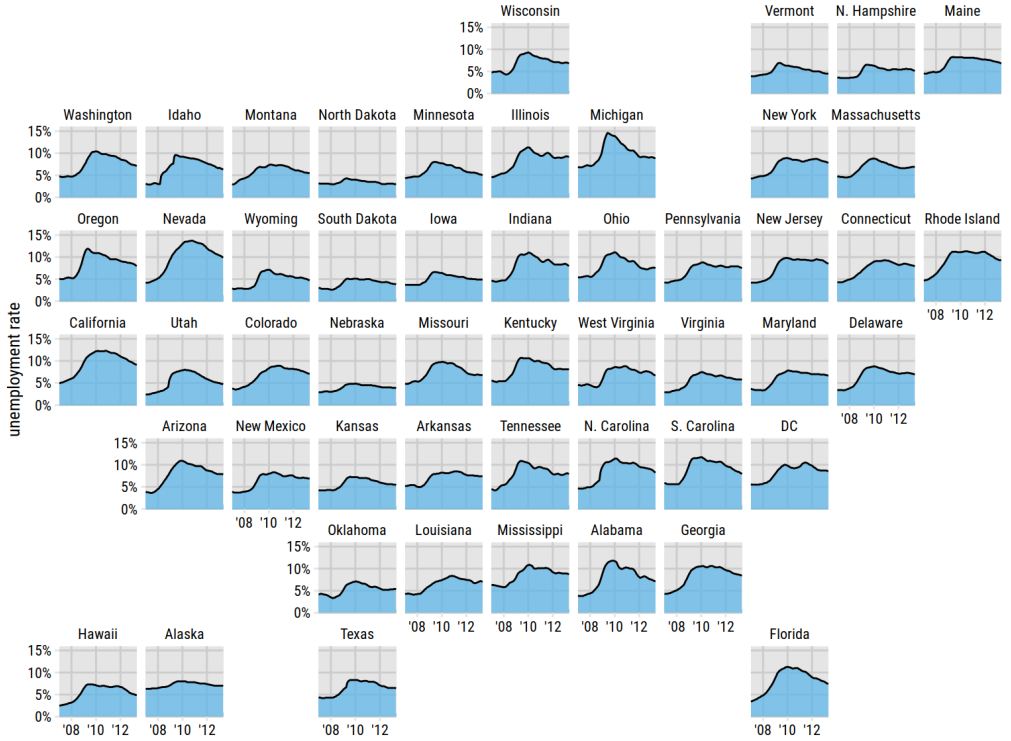
```r
house_prices %>%
  filter(
    date >= ymd("2007-01-01"),
    date <= ymd("2013-05-31")
  ) %>%
  ggplot(aes(date, unemploy_perc)) +
  geom_area(fill = "#56B4E9", alpha = 0.7) +
  geom_line() +
  scale_y_continuous(
    name = "unemployment rate",
    limits = c(0, 16), expand = c(0, 0),
    breaks = c(0, 5, 10, 15),
    labels = c("0%", "5%", "10%", "15%")
  ) +
  scale_x_date(
    name = NULL,
    breaks = ymd(c("2008-01-01", "2010-01-01", "2012-01-01")),
    labels = c("'08", "'10", "'12"),
    expand = c(0, 0)
  ) +
  coord_cartesian(clip = "off") +
  facet_geo(~state, grid = "us_state_grid1", labeller = adjust_labels) +
  theme_dviz_grid(10, dviz_font_family_condensed, rel_small = 10/12) +
```


unemployment rate
Wisconsin
Vermont
N. Hampshire
Maine
Washington
Idaho
Montana
North Dakota
Minnesota
Illinois
Michigan
New York
Massachusetts
Oregon
Nevada
Wyoming
South Dakota
Iowa
Indiana
Ohio
Pennsylvania
New Jersey
Connecticut
Rhode Island
California
Utah
Colorado
Nebraska
Missouri
Kentucky
West Virginia
Virginia
Maryland
Delaware
Arizona
New Mexico
Kansas
Arkansas
Tennessee
N. Carolina
S. Carolina
DC
Oklahoma
Louisiana
Mississippi
Alabama
Georgia
Hawaii
Alaska
Texas
Florida
15%
10%
5%
0%
'08
'10
'12

Unemployment rate leading up to and following the 2008 financial crisis, by state for the US.

Each panel shows the unemployment rate for one state, including the District of Columbia (DC), from January 2007 through May 2013.

Vertical grid lines mark January of 2008, 2010, and 2012. The data comes from the U.S. Bureau of Labor Statistics

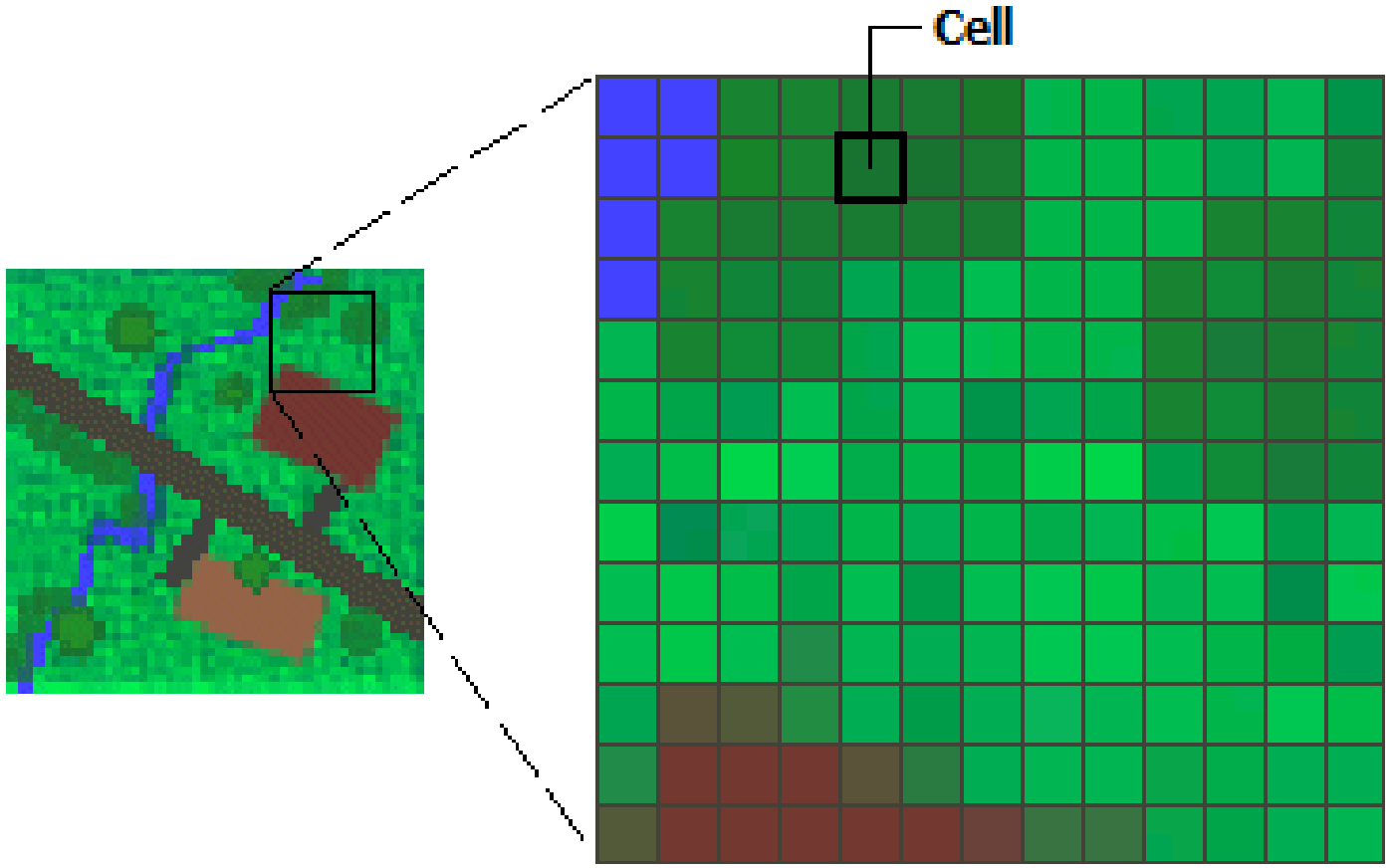


For somebody who is familiar with the geography of the United States, this arrangement may make it easier to find the graphs for specific states than arranging them, for example, in alphabetical order.

Furthermore, one would expect neighboring states to display similar patterns, and we see that this is indeed the case.

https://www.youtube.com/watch?v=8NV5MxcaWR4

Also taking altitude into account...



https://spencerschien.info/post/data_viz_how_to/high_quality_rayshader_visuals/

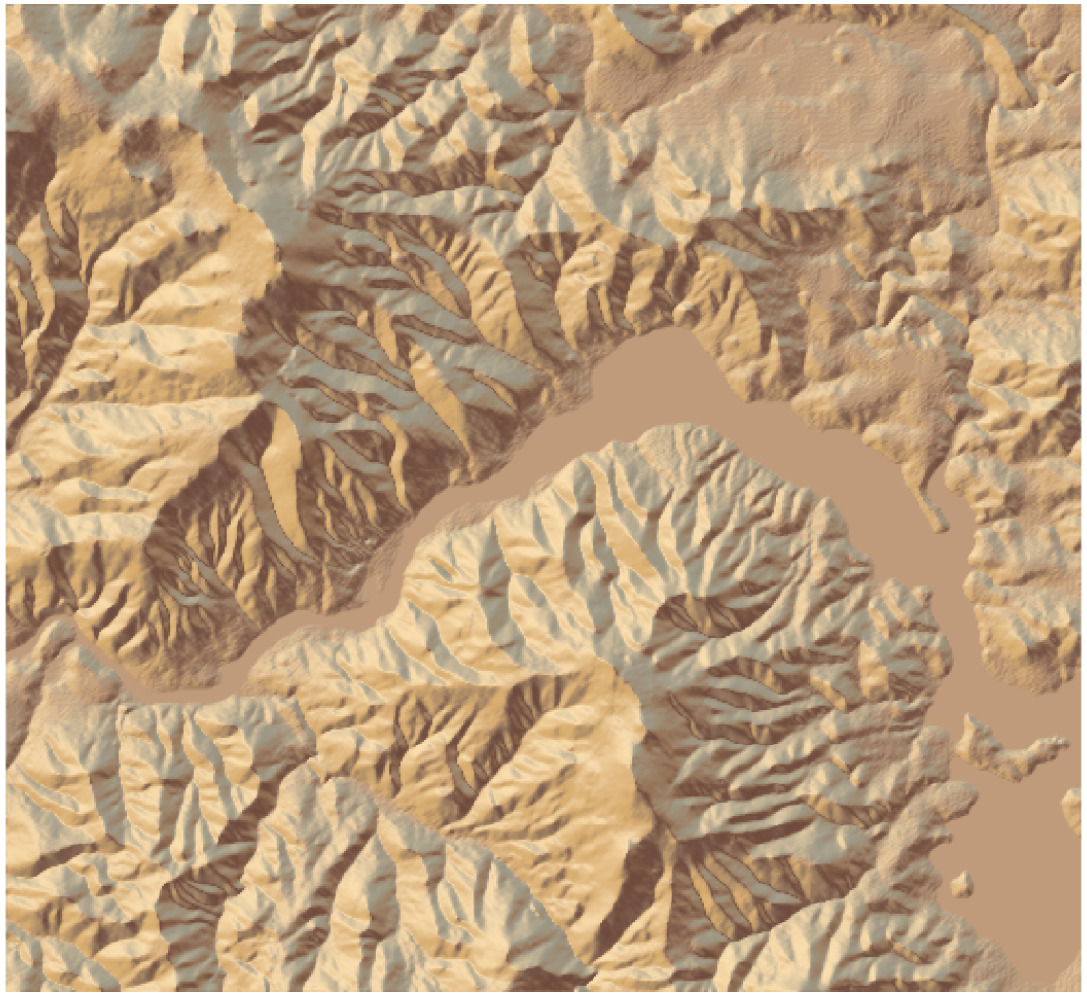
Rayshading

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

Rayshading

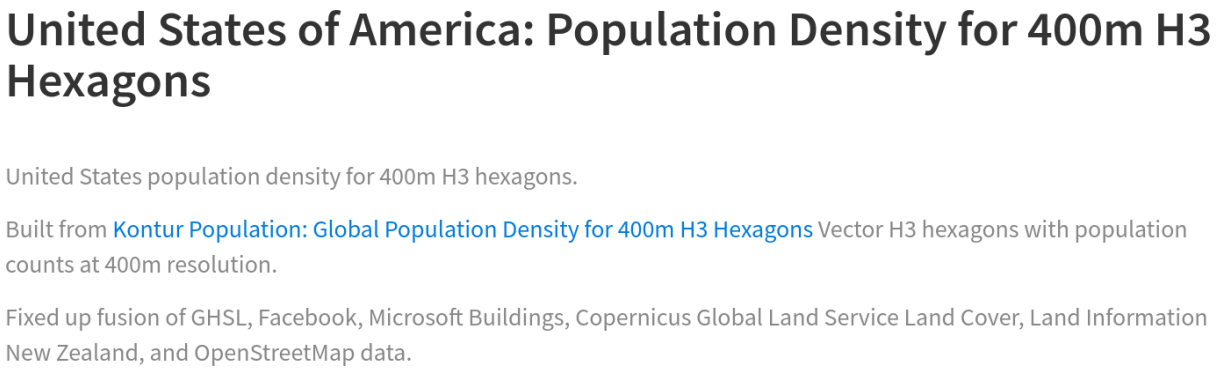
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

```r
library(rayshader)

#Here, I load a map with the raster package.
loadzip = tempfile()
download.file("https://tylermw.com/data/dem_01.tif.zip", loadzip)
localtif = raster::raster(unzip(loadzip, "dem_01.tif"))
unlink(loadzip)

#And convert it to a matrix:
elmat = raster_to_matrix(localtif)

#We use another one of rayshader's built-in textures:
elmat %>%
  sphere_shade(texture = "desert") %>%
  plot_map()
```



```
#detect_water and add_water adds a water layer to the map:
elmat %>%
  sphere_shade(texture = "desert") %>%
  add_water(detect_water(elmat), color = "desert") %>%
  plot_map()
```

Florida Population Density



https://data.humdata.org/dataset/kontur-population-united-states-of-america?

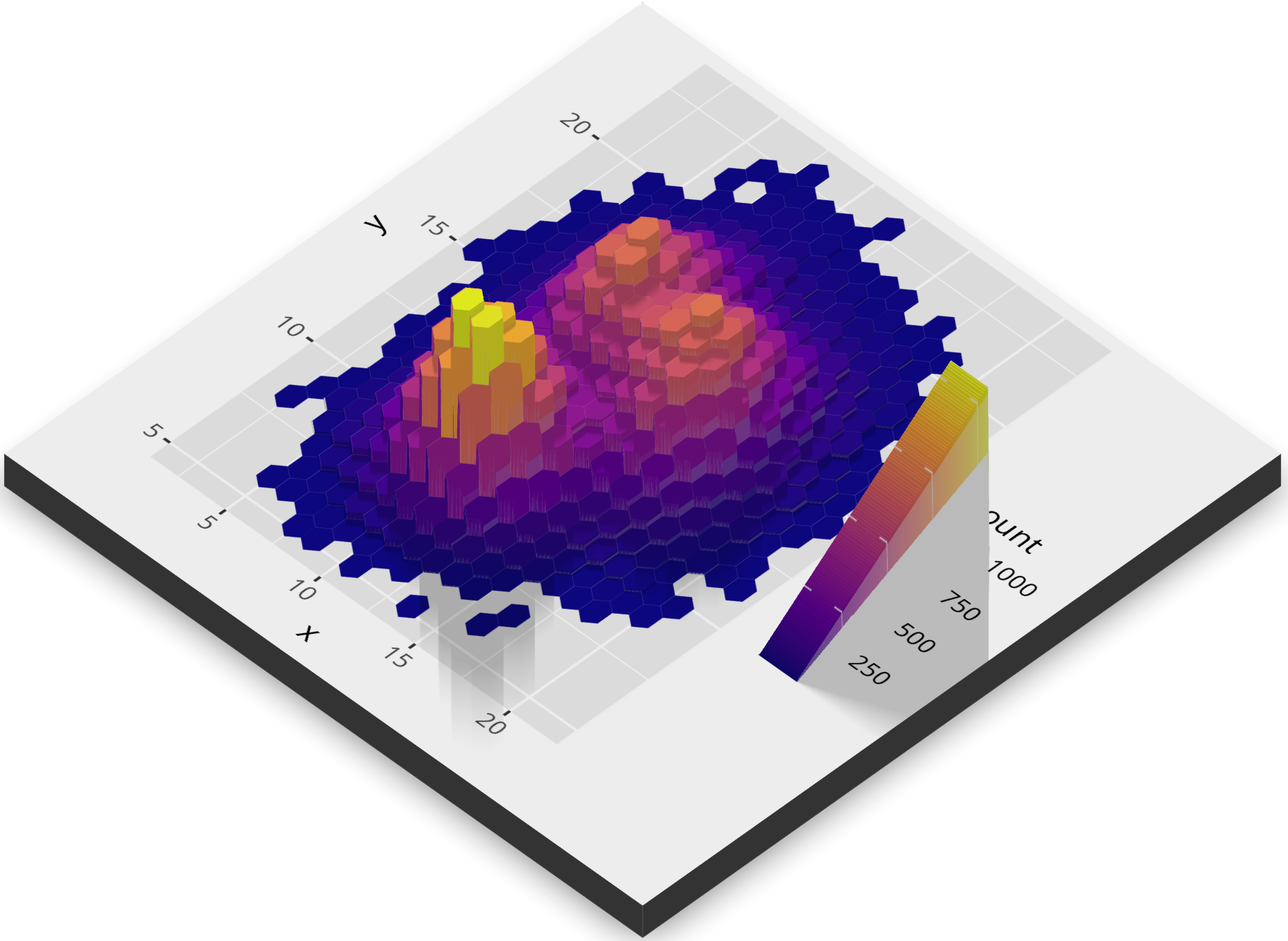
Credits to: https://github.com/Pecners/kontur_rayshader_tutorial/tree/main

Florida Population Density

This map shows population density of Florida. Population estimates are bucketed into 400 meter (about 1/4 mile) hexagons.

Graphic by Spencer Schien (@MrPecners) | Data: Kontur Population (Released 2022-06-30)

```r
a = data.frame(x=rnorm(20000, 10, 1.9), y=rnorm(20000, 10, 1.2) )
b = data.frame(x=rnorm(20000, 14.5, 1.9), y=rnorm(20000, 14.5, 1.9) )
c = data.frame(x=rnorm(20000, 9.5, 1.9), y=rnorm(20000, 15.5, 1.9) )
data = rbind(a,b,c)

pp_nolines = ggplot(data, aes(x=x, y=y)) +
  geom_hex(bins = 20, size = 0) +
  scale_fill_viridis_c(option = "C")

plot_gg(pp_nolines, width = 4, height = 4, scale = 300, multicore = TRUE)

render_snapshot()
```


y
20
15
10
5
x
5
10
15
20
ount
1000
750
500
250

Interactive visualizations...

plotly | Graphing Libraries

Interactive plots in Python

https://plotly.com/python/plotly-express/

https://plotly.com/python/

```
pip install plotly
```

Plotly Express

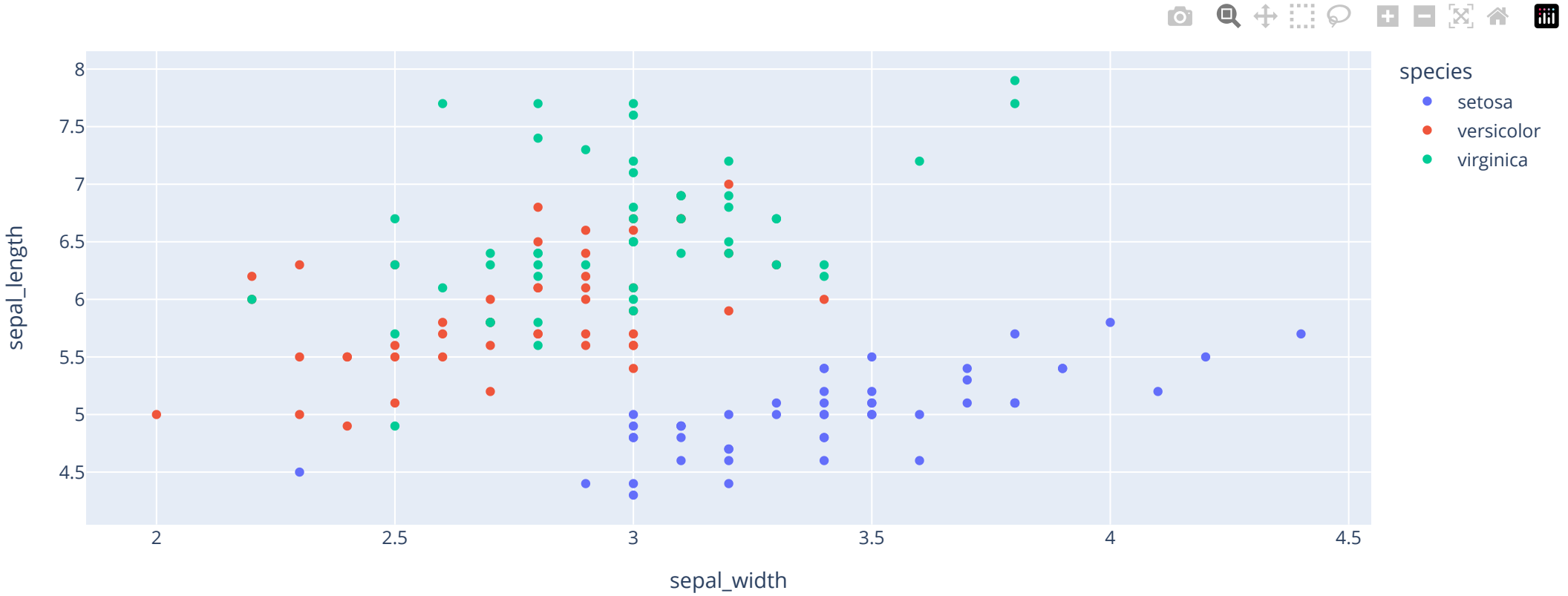
Currently includes the following functions:

- Basics: `scatter`, `line`, `area`, `bar`, `funnel`, `timeline`
- Part-of-Whole: `pie`, `sunburst`, `treemap`, `icicle`, `funnel_area`
- 1D Distributions: `histogram`, `box`, `violin`, `strip`, `ecdf`
- 2D Distributions: `density_heatmap`, `density_contour`
- Matrix or Image Input: `imshow`
- 3-Dimensional: `scatter_3d`, `line_3d`

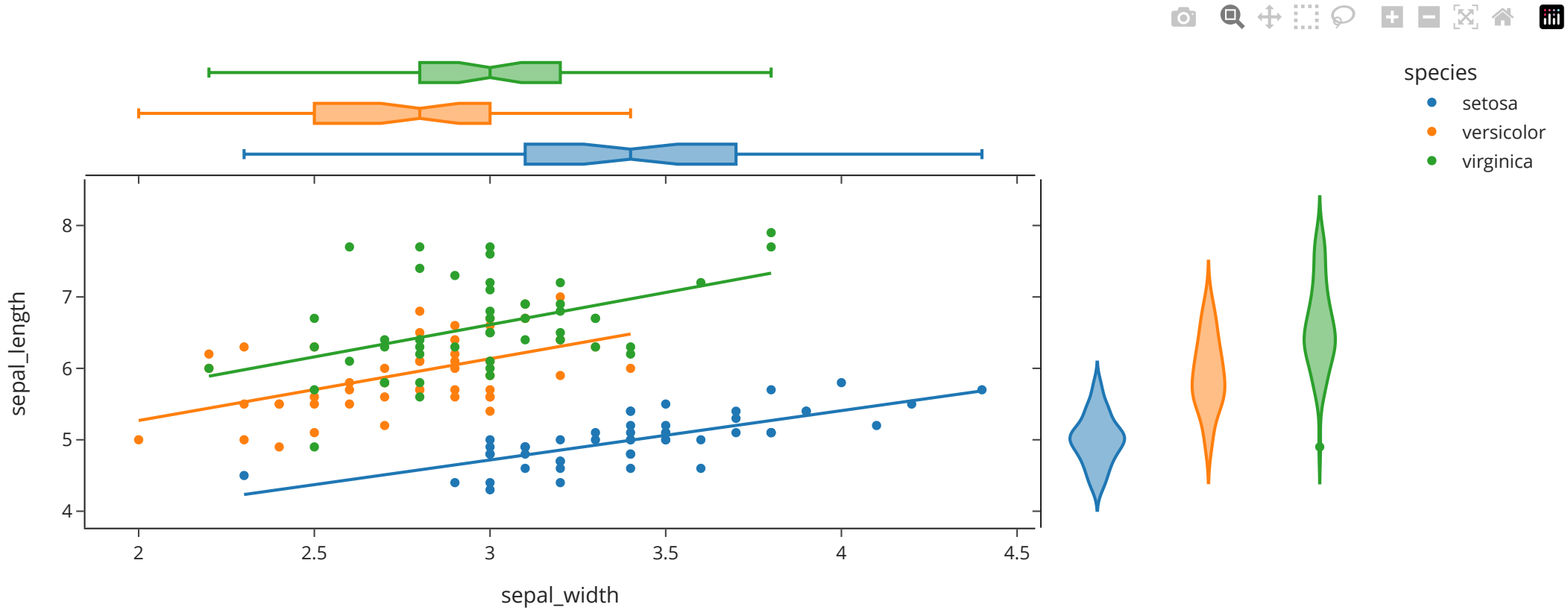
Plotly Express

- Multidimensional: `scatter_matrix`, `parallel_coordinates`, `parallel_categories`
- Tile Maps: `scatter_mapbox`, `line_mapbox`, `choropleth_mapbox`, `density_mapbox`
- Outline Maps: `scatter_geo`, `line_geo`, `choropleth`
- Polar Charts: `scatter_polar`, `line_polar`, `bar_polar`
- Ternary Charts: `scatter_ternary`, `line_ternary`

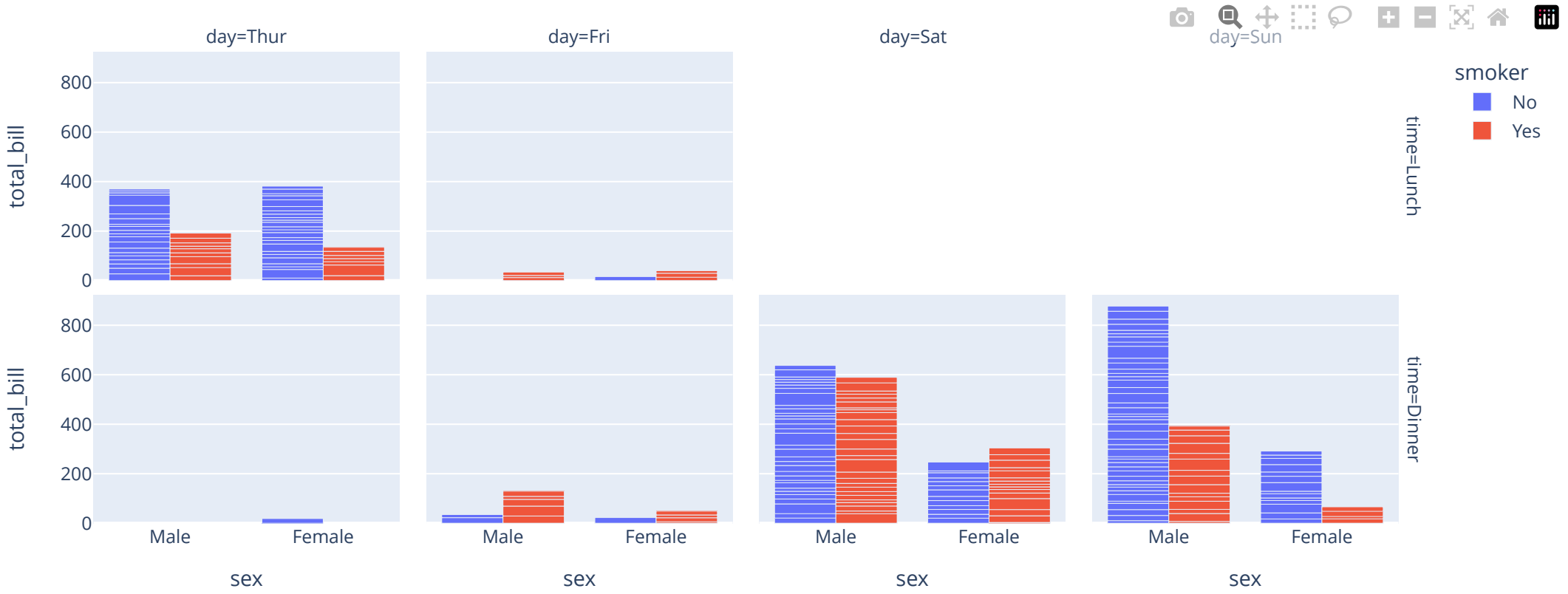
```python
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```



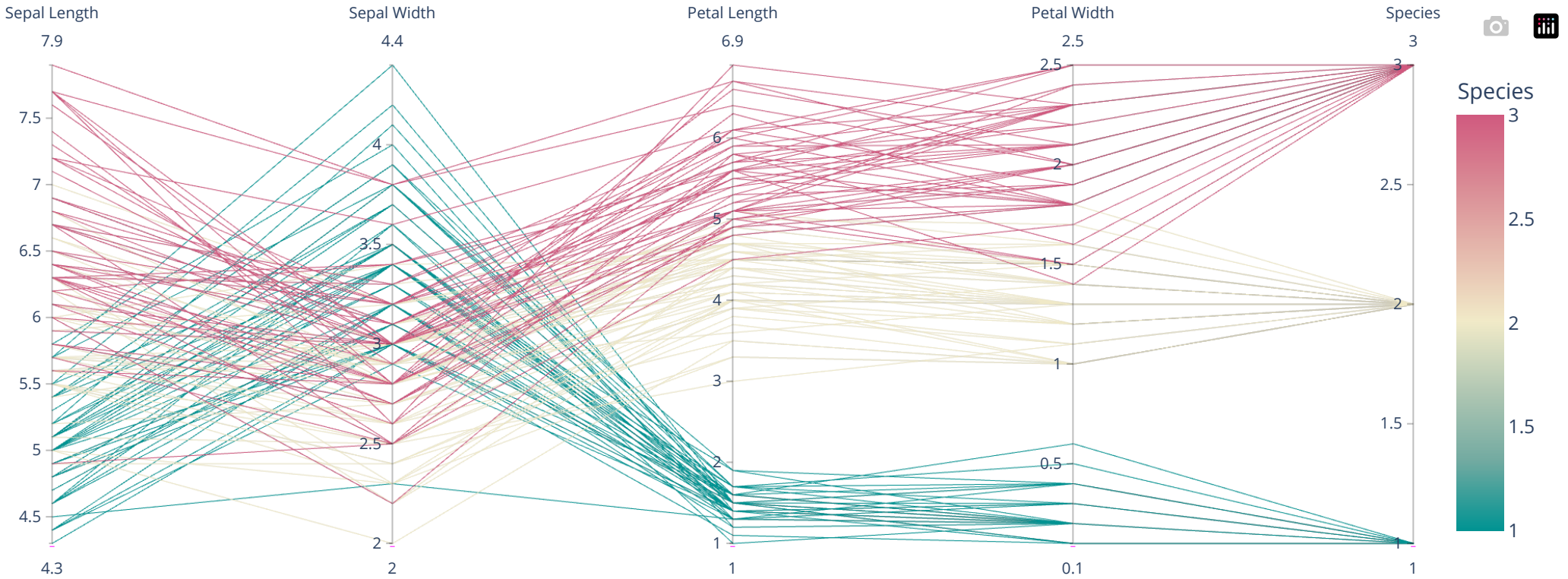
```python
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width",
y="sepal_length", color="species", marginal_y="violin",
marginal_x="box", trendline="ols", template="simple_white")
fig.show()
```



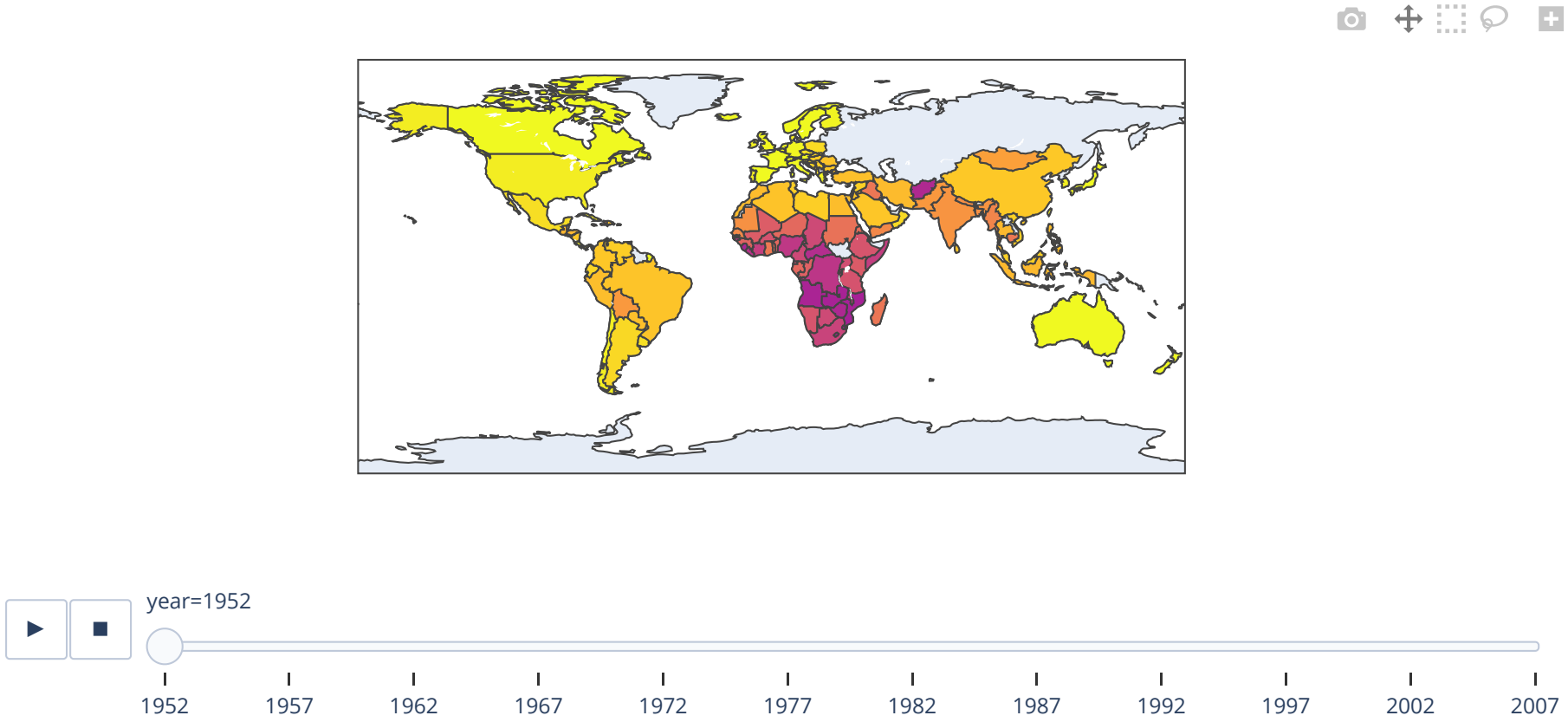
```python
import plotly.express as px
df = px.data.tips()
fig = px.bar(df, x="sex", y="total_bill", color="smoker", barmode="group", facet_row="time", fa
       category_orders={"day": ["Thur", "Fri", "Sat", "Sun"], "time": ["Lunch", "Dinner"]})
fig.show()
```



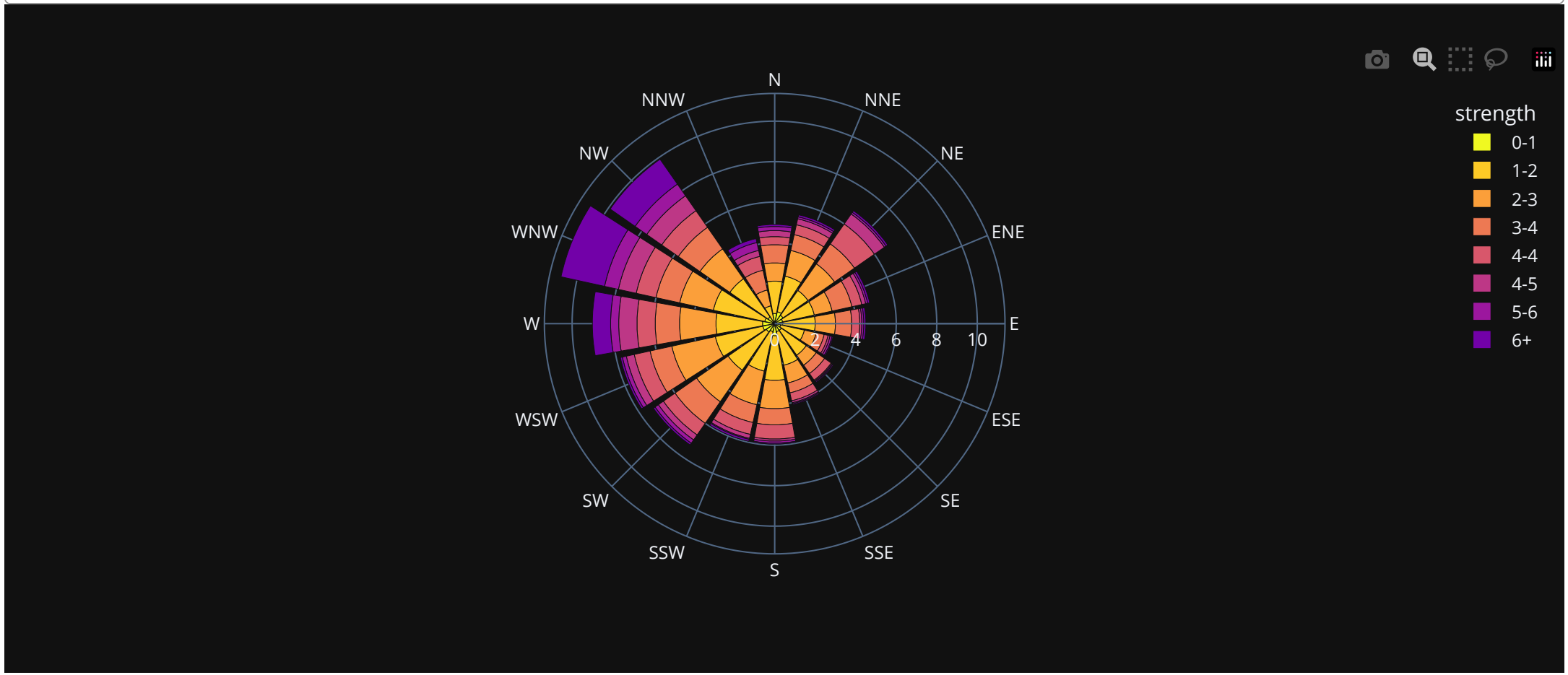
```python
import plotly.express as px
df = px.data.iris()
fig = px.parallel_coordinates(df, color="species_id", labels={"species_id": "Species",
                    "sepal_width": "Sepal Width", "sepal_length": "Sepal Length",
                    "petal_width": "Petal Width", "petal_length": "Petal Length", },
                      color_continuous_scale=px.colors.diverging.Tealrose, color_continuous_midpo
fig.show()
```



```python
import plotly.express as px
df = px.data.gapminder()
fig = px.choropleth(df, locations="iso_alpha", color="lifeExp", hover_name="country", animation
fig.show()
```



```python
import plotly.express as px
df = px.data.wind()
fig = px.bar_polar(df, r="frequency", theta="direction", color="strength", template="plotly_dar
            color_discrete_sequence= px.colors.sequential.Plasma_r)
fig.show()
```



Interactive plots in R

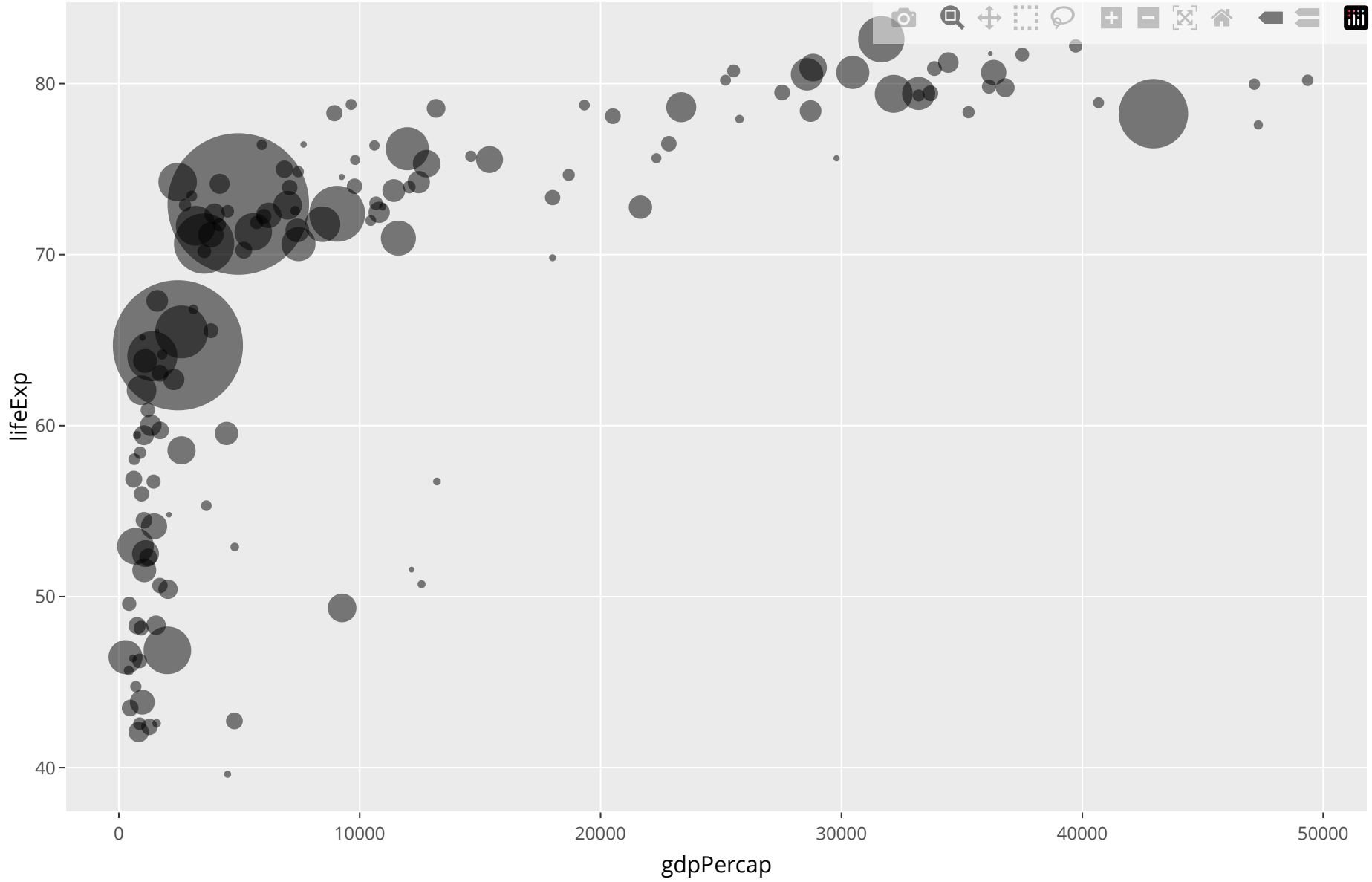
Plotly ggplot2 Open Source Graphing Library

With ggplotly() by Plotly, you can convert your ggplot2 figures into interactive ones powered by plotly.js, ready for embedding into Dash applications.

ggplotly is free and open source and you can view the source, report issues or contribute on GitHub.

Head over to the community forum to ask questions and get help.

```r
library(plotly)
library(dplyr)
library(gapminder)

data <- gapminder %>% filter(year=="2007") %>% dplyr::select(-year)

p <- data %>%
      arrange(desc(pop)) %>%
      mutate(country = factor(country, country)) %>%

ggplot(aes(x=gdpPercap, y=lifeExp, size = pop)) +
  geom_point(alpha=0.5) +
  scale_size(range = c(.1, 24), name="Population (M)")

ggplotly(p, height = 600, width=900)
```


lifeExp
80
70
60
50
40
0
10000
20000
30000
40000
50000
gdpPercap

Visualizing text...

Prosocial

Anxiety

Anger

Social

Sadness

Posemo

Word clouds in Python

https://amueller.github.io/word_cloud/auto_examples/index.html#example-gallery

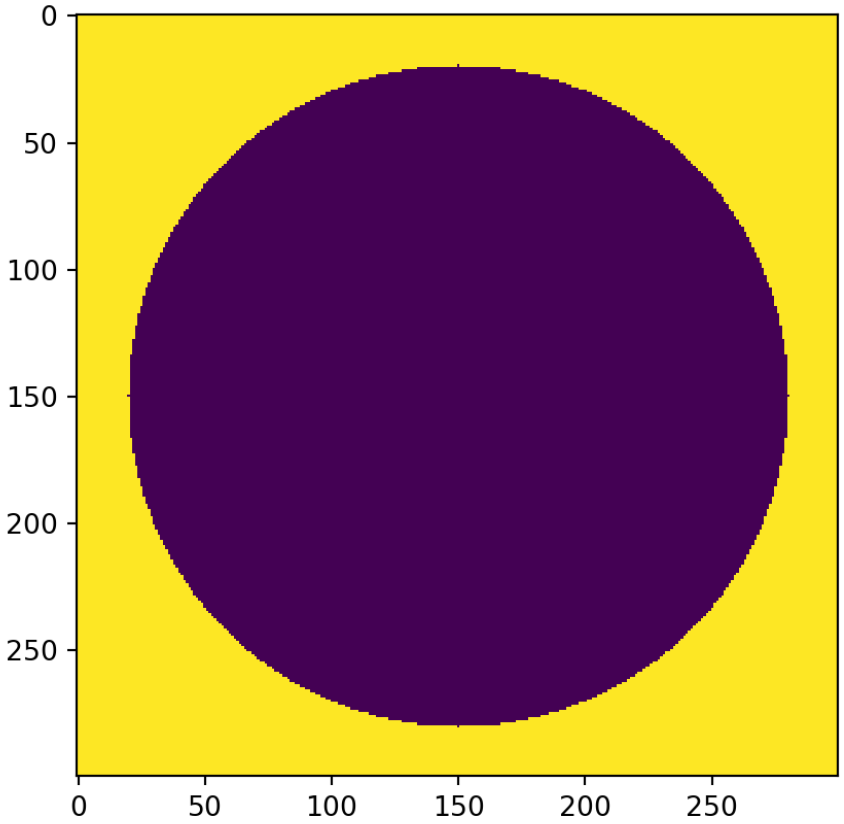
https://github.com/amueller/word_cloud

```
pip install wordcloud
```

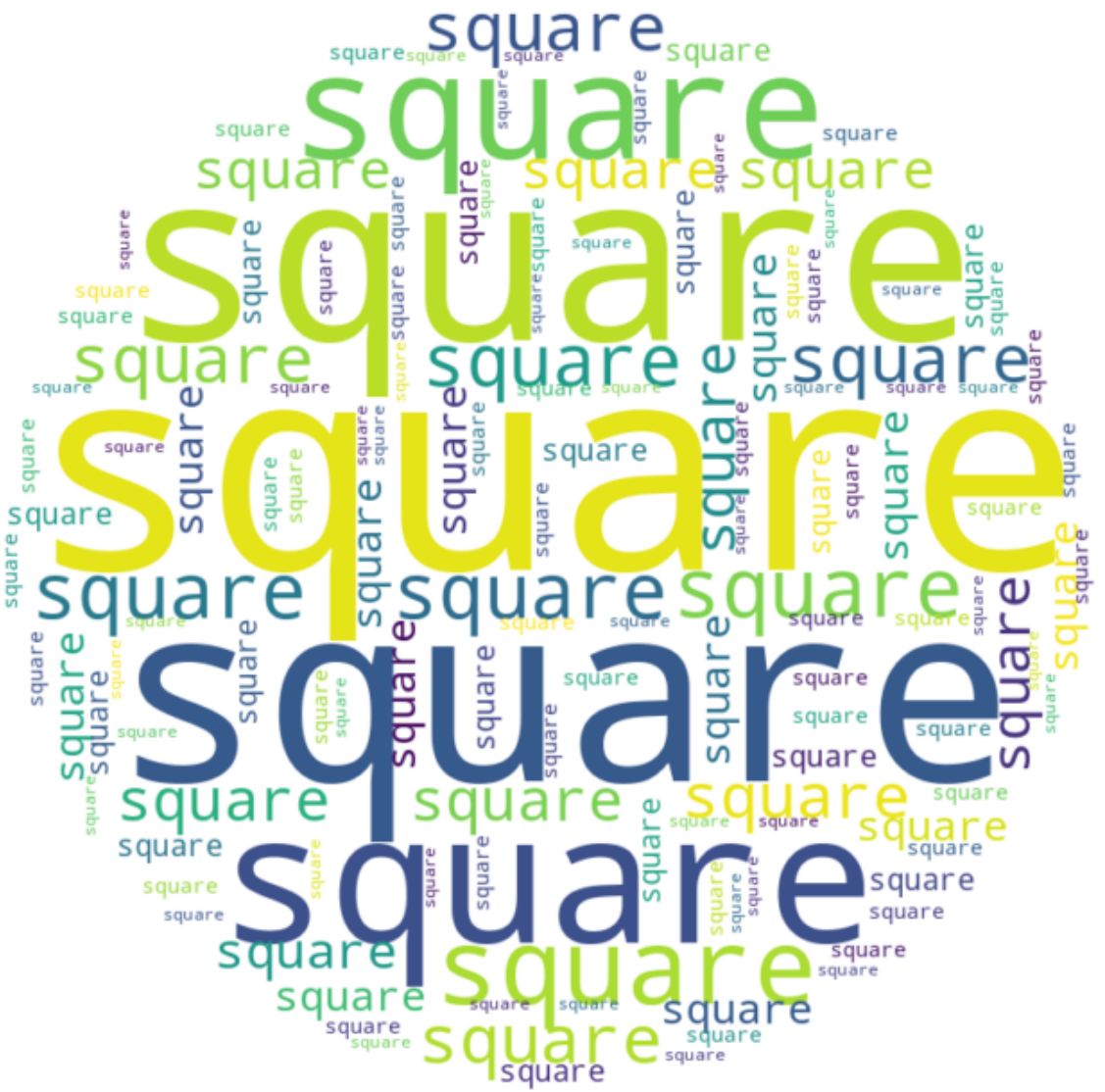
```python
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud

text = "square"

x, y = np.ogrid[:300, :300]

mask = (x - 150) ** 2 + (y - 150) ** 2 > 130 ** 2
mask = 255 * mask.astype(int)
```

```
array([[255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       ...,
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255]])
```

```
plt.imshow(mask, interpolation='nearest')
plt.show()
```

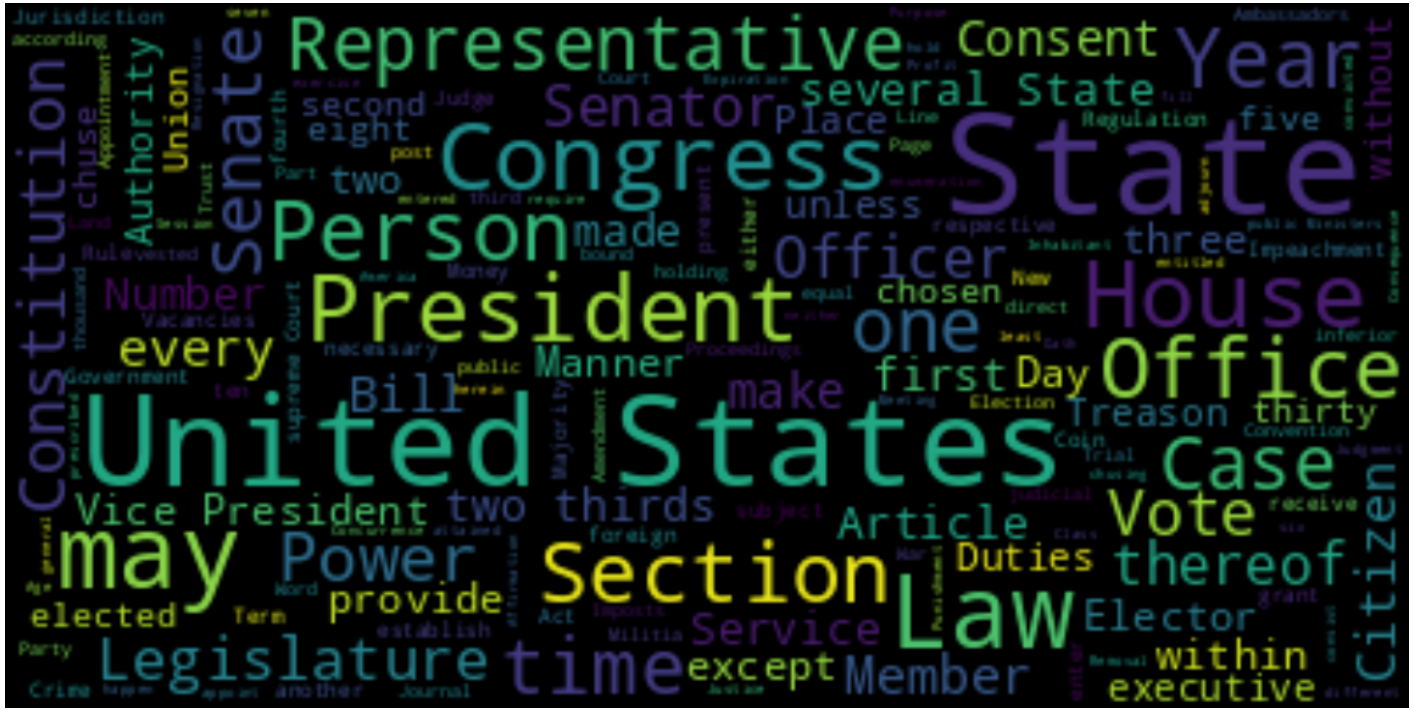


```
wc = WordCloud(width=1200,height=1200,
background_color="white", repeat=True, mask=mask, scale=3)
_ = wc.generate(text)

_ = plt.figure(figsize=(4,4),dpi=300)
_ = plt.axis("off")
_ = plt.imshow(wc, interpolation="bilinear")
plt.tight_layout()
plt.show()
```


square

```
from wordcloud import WordCloud

import urllib.request
# _ = urllib.request.urlretrieve("https://raw.githubusercontent.com/amueller/word_cloud/main/ex


# Read the whole text.
text = open('constitution.txt').read()

# Generate a word cloud image
wordcloud = WordCloud(width=1800, height=900).generate(text)

# Display the generated image:
# the matplotlib way:
import matplotlib.pyplot as plt
_ = plt.imshow(wordcloud, interpolation='bilinear')
_ = plt.axis("off")
```

```python
# lower max_font_size
wordcloud = WordCloud(max_font_size=40).generate(text)
_ = plt.figure(figsize=(6,3), dpi=300)
_ = plt.imshow(wordcloud, interpolation="bilinear")
_ = plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

# The pil way (if you don't have matplotlib)
# image = wordcloud.to_image()
# image.show()
```



Word clouds in R

https://cran.r-project.org/web/packages/wordcloud2/vignettes/wordcloud.html

```
1 library(wordcloud2)
2 wordcloud2(data = demoFreq)
```

```r
head(wordcloud2::demoFreq,15)
```

```
         word freq
oil       oil   85
said     said   73
prices prices   48
opec     opec   42
mln       mln   31
the       the   26
last     last   24
bpd       bpd   23
dlrs     dlrs   23
crude   crude   21
market market   20
reuter reuter   20
saudi   saudi   18
will     will   18
one       one   17
```

Wordshift Graphs

https://ryanjgallagher.github.io/code/word_shift/overview

https://shifterator.readthedocs.io/en/latest/cookbook/getting_started.html#case-study

Gallagher, R. J., Frank, M. R., Mitchell, L., Schwartz, A. J., Reagan, A. J., Danforth, C. M., & Dodds, P. S. (2021). Generalized word shift graphs: A method for visualizing and explaining pairwise comparisons between texts. EPJ Data Science, 10(1), 4. https://doi.org/10.1140/epjds/s13688-021-00260-3

Data: Reviews of “Animal Crossing - New Horizons”

https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-05-05/readme.md

```
1 import urllib.request
2 # _ = urllib.request.urlretrieve("https://raw.githubusercontent.com/rfordatascience/tidytuesday
3
4 import pandas as pd
5 pd.set_option('display.max_columns', 4)
6
7 reviews = pd.read_csv("user_reviews.tsv", sep="\t")
8 reviews.iloc[:5,:3]
```

	grade	user_name	text
0	4	mds27272	My gf started playing before me. No option to ...
1	5	lolo2178	While the game itself is great, really relaxin...
2	0	Roachant	My wife and I were looking forward to playing ...
3	0	Houndf	We need equal values and opportunities for all...
4	0	ProfessorFox	BEWARE! If you have multiple people in your h...

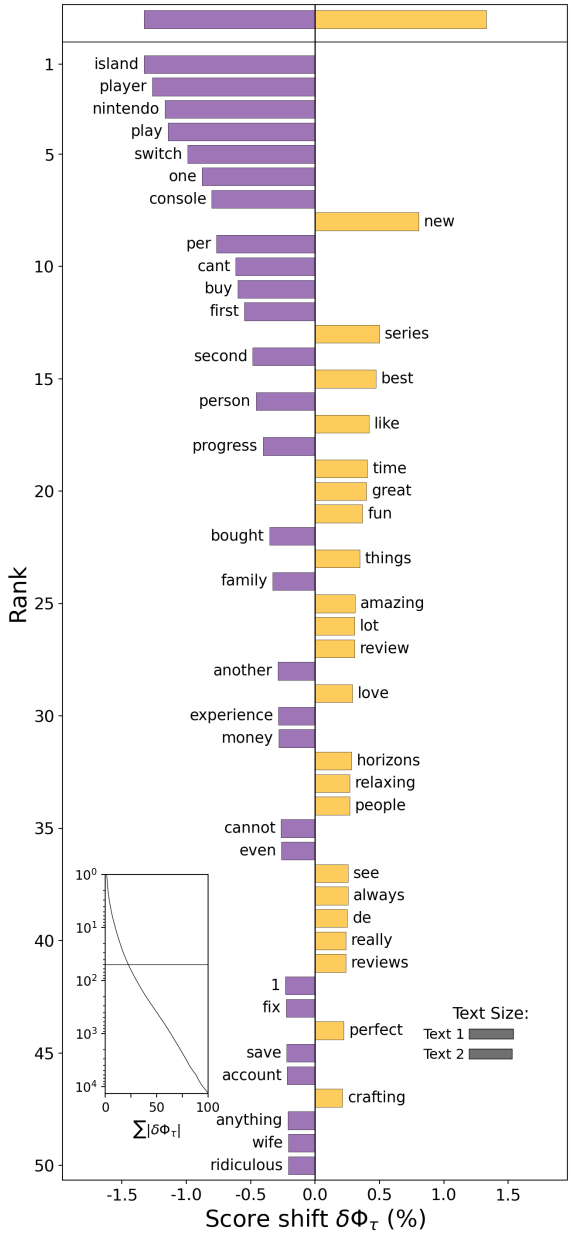
```
import numpy as np
import itertools
import collections
import nltk
from nltk.corpus import stopwords
import re

stop_words = set(stopwords.words('english'))

def remove_punctuation(txt):
    """Replace URLs and other punctuation found in a text string with nothing
    (i.e. it will remove the URL from the string).

    Parameters
    ----------
    txt : string
        A text string that you want to parse and remove urls.

    Returns
    -------
    The same txt string with URLs and punctuation removed.
    """

```

```
# Divide reviews into positive and negative based on the median grade for the dataset
median_grade = reviews.grade.median()

reviews.loc[reviews['grade'] <= median_grade, 'review_category'] = 'Negative'
reviews.loc[reviews['grade'] > median_grade, 'review_category'] = 'Positive'

reviews_neg = reviews[reviews['review_category'] == 'Negative']
reviews_pos = reviews[reviews['review_category'] == 'Positive']

texts = reviews['text'].tolist()
texts_neg = reviews_neg['text'].tolist()
texts_pos = reviews_pos['text'].tolist()
```

```
# Clean up the review texts and count frequencies
clean_texts_neg = clean_text(texts_neg)
clean_texts_pos = clean_text(texts_pos)

dict(list(clean_texts_pos.items())[:60])
```

```
{'gf': 12, 'started': 68, 'playing': 419, 'option': 57, 'create': 77, 'island': 1520, 'guys': 13,
'2nd': 19, 'player': 496, 'start': 135, 'console': 467, 'sucks': 54, 'miss': 25, 'much': 383,
'1st': 13, 'gets': 117, 'terms': 15, 'activities': 24, 'resources': 74, 'etc': 57, 'absolutely':
139, 'terrible': 51, 'buy': 242, 'one': 1299, 'person': 152, 'household': 54, 'wants': 37, 'get':
465, 'full': 116, 'game': 4276, 'experience': 335, 'thats': 163, 'unacceptable': 6, 'great': 445,
'really': 535, 'relaxing': 200, 'gorgeous': 46, 'cant': 381, 'ignore': 28, 'thing': 222, 'ruins':
14, 'whole': 99, 'lot': 283, 'people': 592, 'seen': 58, 'different': 102, 'user': 87,
'reviewsthat': 2, '1': 237, 'per': 558, 'decision': 60, 'limits': 10, 'able': 188, 'enjoy': 174,
'also': 267, 'nukes': 2, 'creative': 38, 'control': 46, 'since': 221, 'havewhile': 1}
```

```
import shifterator as sh

proportion_shift = sh.ProportionShift(type2freq_1=dict(clean_texts_neg),
                                      type2freq_2=dict(clean_texts_pos))

_ = proportion_shift.get_shift_graph()
```


Rank
Score shift $\delta\Phi_\tau$ (%)
island
player
nintendo
play
switch
one
console
per
cant
buy
first
second
person
progress
bought
family
another
experience
money
cannot
even
1
fix
save
account
anything
wife
ridiculous
new
series
best
like
time
great
fun
things
amazing
lot
review
love
horizons
relaxing
people
see
always
de
really
reviews
perfect
crafting
$\sum|\delta\Phi_\tau|$
Text Size:
Text 1
Text 2

Acknowledgments

https://www.youtube.com/watch?v=8NV5MxcaWR4

https://spencerschien.info/post/data_viz_how_to/high_quality_rayshader_visuals/

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)

https://www.rayshader.com/

https://www.tylermw.com/adding-open-street-map-data-to-rayshader-maps-in-r/

Acknowledgments

https://www.tylermw.com/3d-ggplots-with-rayshader/

https://www.youtube.com/watch?v=8NV5MxcaWR4

https://wilkelab.org/SDS375/slides/interactive-plots.html

https://www.kaggle.com/code/mrisdal/shifterator-analysis-on-animal-crossing-reviews