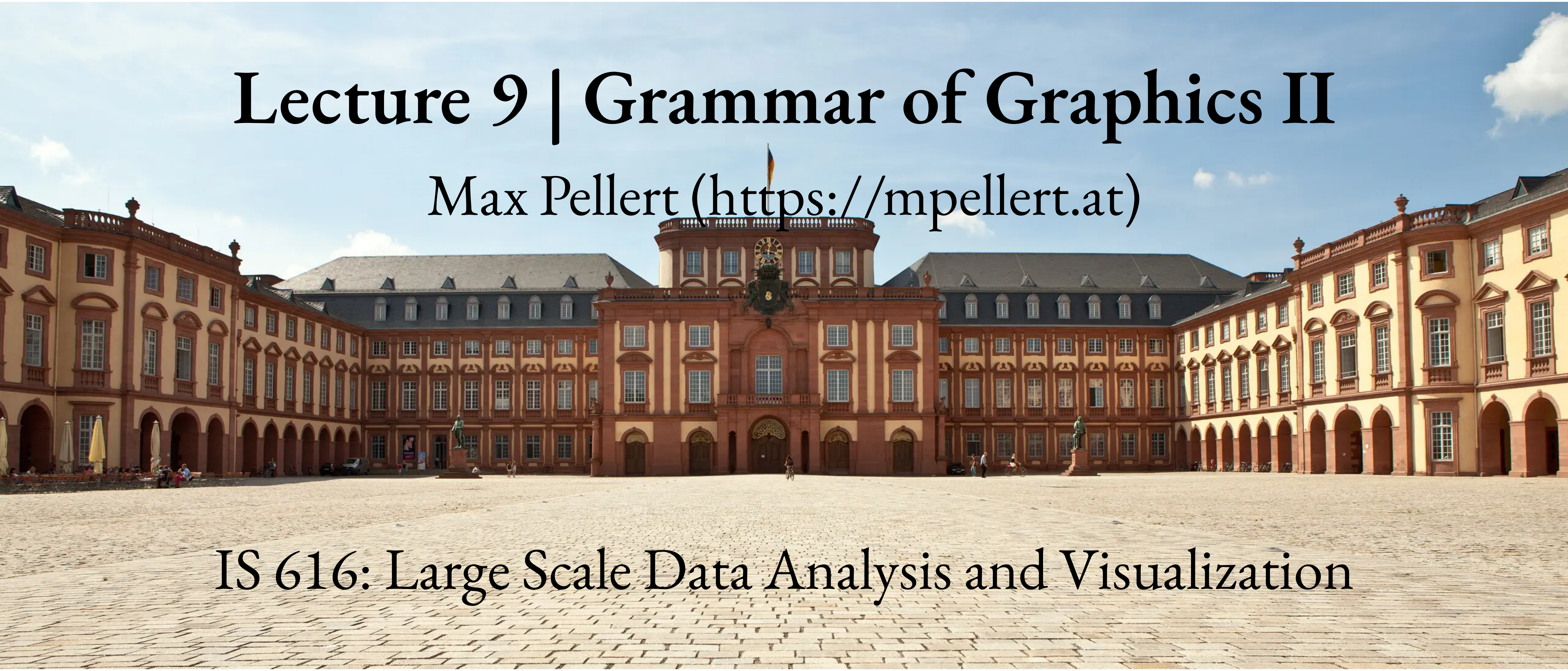


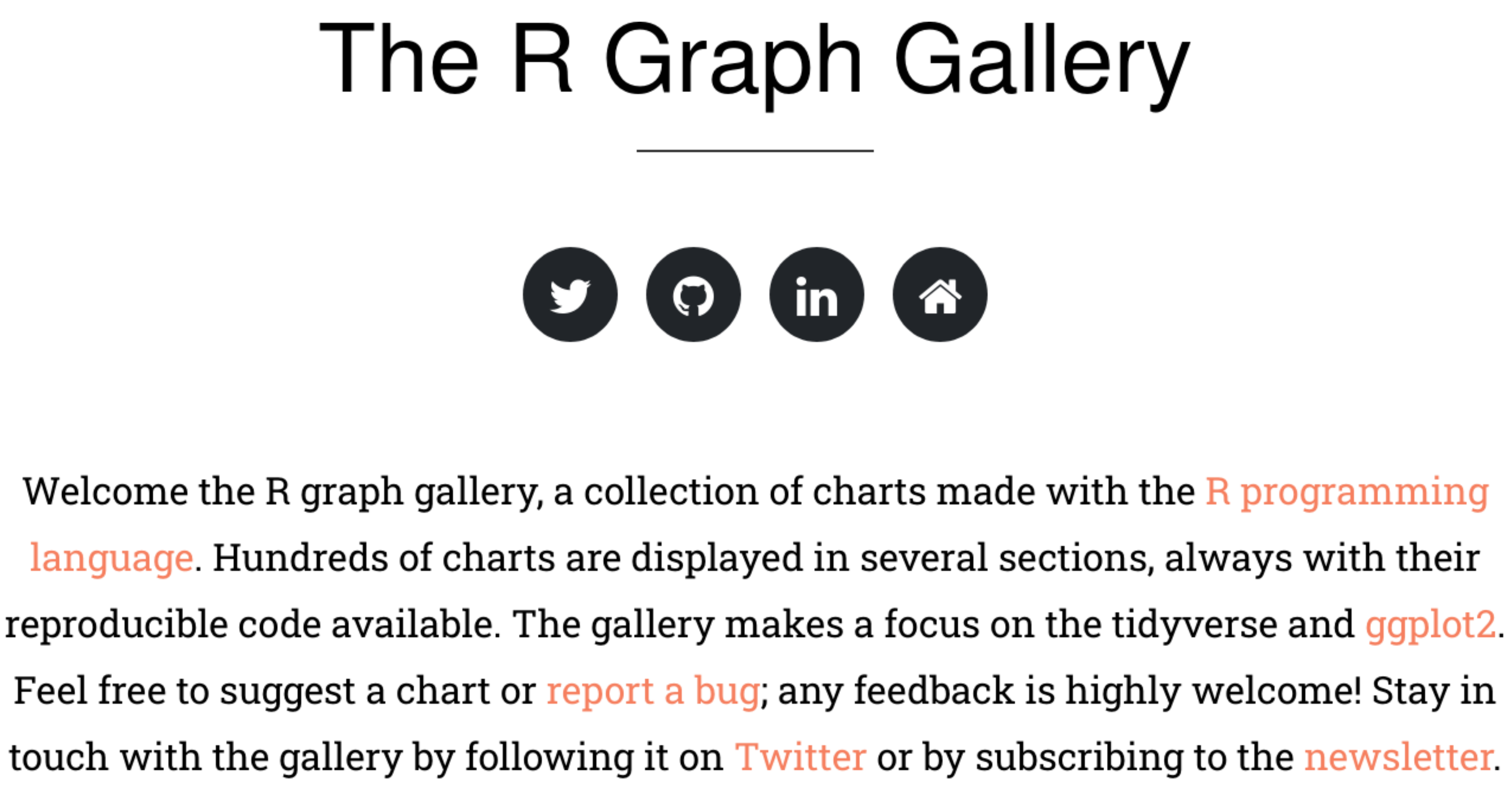
Lecture 9 | Grammar of Graphics II

Max Pellert (https://mpellert.at)

IS 616: Large Scale Data Analysis and Visualization

UNIVERSITY
OF MANNHEIM

HOVER: X
ORMATION MAP
LOT
CHAR
DESIGN
PYTHON
SCIENCE
GRAPH
GALLERY
CO
WORD
DAT
WEB
CLOUD

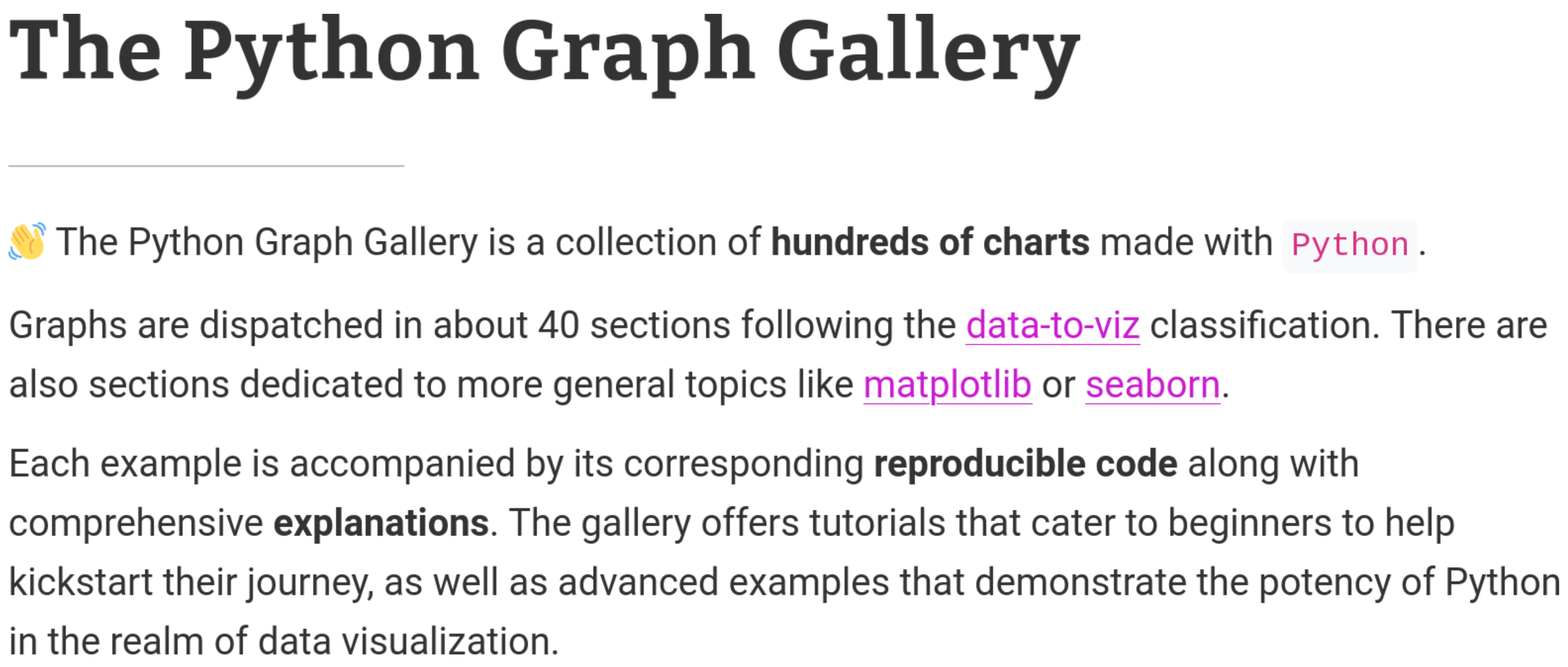


The R Graph Gallery

Welcome the R graph gallery, a collection of charts made with the R programming language. Hundreds of charts are displayed in several sections, always with their reproducible code available. The gallery makes a focus on the tidyverse and ggplot2. Feel free to suggest a chart or report a bug; any feedback is highly welcome! Stay in touch with the gallery by following it on Twitter or by subscribing to the newsletter.

https://r-graph-gallery.com/

https://github.com/holtzy/R-graph-gallery



The Python Graph Gallery

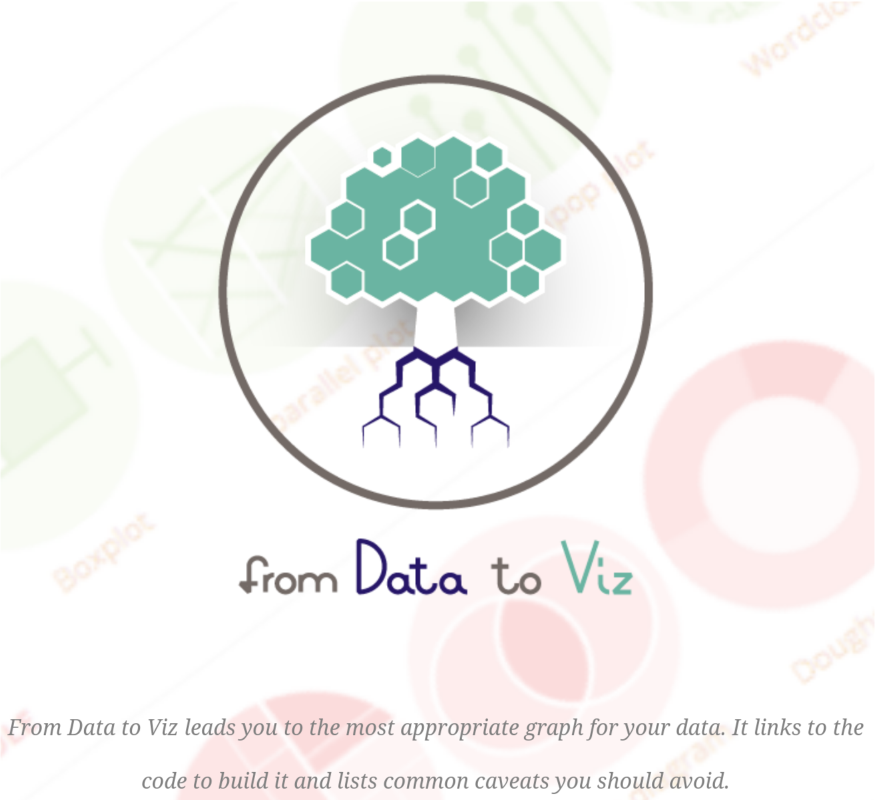
👋 The Python Graph Gallery is a collection of **hundreds of charts** made with `Python`.

Graphs are dispatched in about 40 sections following the data-to-viz classification. There are also sections dedicated to more general topics like matplotlib or seaborn.

Each example is accompanied by its corresponding **reproducible code** along with comprehensive **explanations**. The gallery offers tutorials that cater to beginners to help kickstart their journey, as well as advanced examples that demonstrate the potency of Python in the realm of data visualization.

https://python-graph-gallery.com/

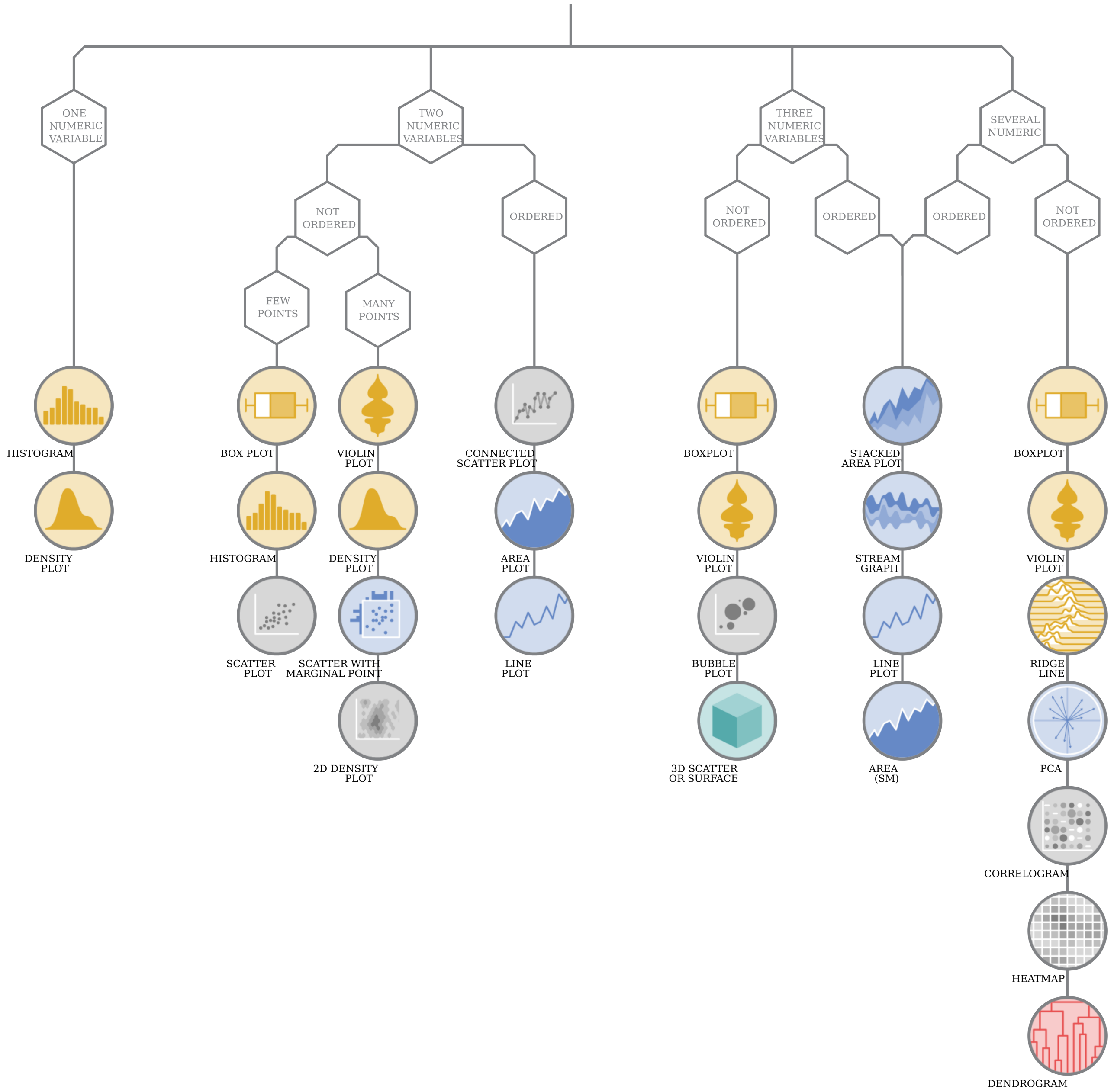
https://github.com/holtzy/The-Python-Graph-Gallery

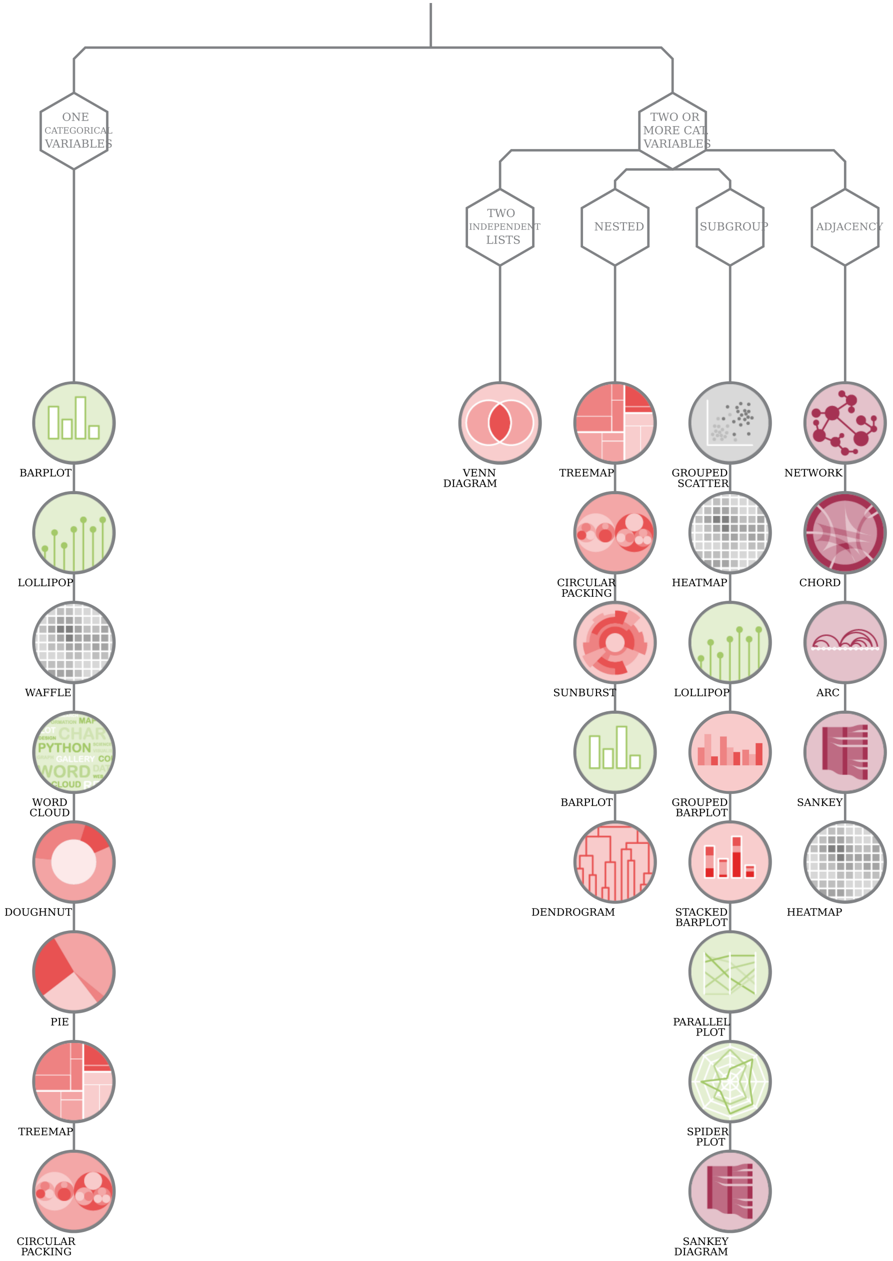


From Data to Viz leads you to the most appropriate graph for your data. It links to the code to build it and lists common caveats you should avoid.

Using the “data-to-viz classification”

https://www.data-to-viz.com/


ONE NUMERIC VARIABLE
TWO NUMERIC VARIABLES
THREE NUMERIC VARIABLES
SEVERAL NUMERIC
NOT ORDERED
ORDERED
NOT ORDERED
ORDERED
ORDERED
NOT ORDERED
FEW POINTS
MANY POINTS
HISTOGRAM
DENSITY PLOT
BOX PLOT
HISTOGRAM
SCATTER PLOT
VIOLIN PLOT
DENSITY PLOT
SCATTER WITH MARGINAL POINT
2D DENSITY PLOT
CONNECTED SCATTER PLOT
AREA PLOT
LINE PLOT
BOXPLOT
VIOLIN PLOT
BUBBLE PLOT
3D SCATTER OR SURFACE
STACKED AREA PLOT
STREAM GRAPH
LINE PLOT
AREA (SM)
BOXPLOT
VIOLIN PLOT
RIDGE LINE
PCA
CORRELOGRAM
HEATMAP
DENDROGRAM



ONE CATEGORICAL VARIABLES

BARPLOT
LOLLIPOP
WAFFLE
WORD CLOUD
DOUGHNUT
PIE
TREEMAP
CIRCULAR PACKING

TWO OR MORE CAT VARIABLES

TWO INDEPENDENT LISTS

VENN DIAGRAM

NESTED

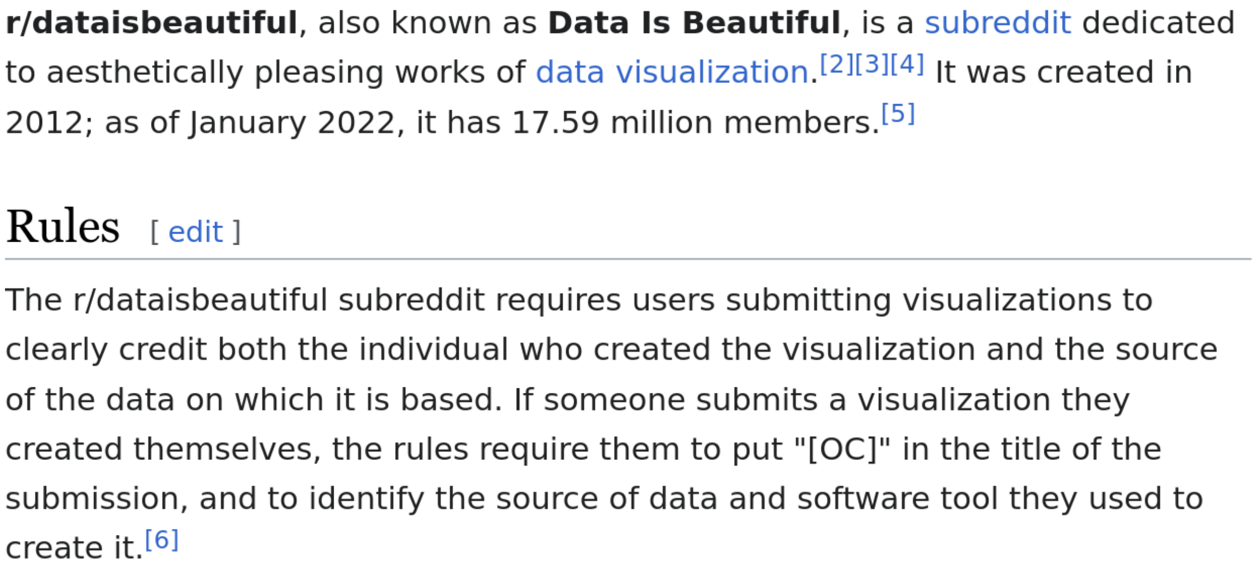
TREEMAP
CIRCULAR PACKING
SUNBURST
BARPLOT
DENDROGRAM

SUBGROUP

GROUPED SCATTER
HEATMAP
LOLLIPOP
GROUPED BARPLOT
STACKED BARPLOT
PARALLEL PLOT
SPIDER PLOT
SANKEY DIAGRAM

ADJACENCY

NETWORK
CHORD
ARC
SANKEY
HEATMAP



r/dataisbeautiful, also known as **Data Is Beautiful**, is a subreddit dedicated to aesthetically pleasing works of data visualization.[2][3][4] It was created in 2012; as of January 2022, it has 17.59 million members.[5]

Rules [edit]

The r/dataisbeautiful subreddit requires users submitting visualizations to clearly credit both the individual who created the visualization and the source of the data on which it is based. If someone submits a visualization they created themselves, the rules require them to put "[OC]" in the title of the submission, and to identify the source of data and software tool they used to create it.[6]

https://en.wikipedia.org/wiki/R/dataisbeautiful

https://www.reddit.com/r/dataisbeautiful/

```python
import datetime

import pandas as pd
from plotnine import *
```

“Visualizing Baby Sleep Times in Python”

All code from: https://github.com/dodger487/snoo_plots

For more analyses of the same data that we won’t cover: https://www.relevantmisc.com/python/r/data/2020/06/10/baby-sleep-night-day/

```python
df = pd.read_csv(
  'https://github.com/\
dodger487/snoo_plots/\
raw/master/sleep_data.csv'
  )

df.head()
```

```
                  start_time
end_time   duration  asleep  soothing
0  2019-11-21 18:30:32  2019-11-21
18:31:52        80       80         0
1  2019-11-22 04:03:22  2019-11-22
05:07:17      3835     2422      1413
2  2019-11-22 05:53:24  2019-11-22
05:54:27        63       36        27
3  2019-11-22 05:52:51  2019-11-22
07:01:11      4100     3140       960
4  2019-11-22 22:51:19  2019-11-22
23:20:48      1769     1289       480
```

We want to plot date on the x-axis and the time of day on the y-axis. We'll have a vertical line from the start to the end of a sleep session.

The datetimes include both dates and times so we'll have to break them apart. Pandas `.dt` is great here, it allows you to use the standard library datetime methods on a Pandas series in parallel.

```python
# Break out dates and times.
df["start_datetime"]\
= pd.to_datetime(\
df["start_time"])

df["end_datetime"]\
= pd.to_datetime(\
df["end_time"])

df["start_time"]\
= df["start_datetime"].dt.time

df["end_time"]\
= df["end_datetime"].dt.time

df["start_date"]\
= df["start_datetime"].dt.date
```

What about sleep sessions that span days?

```python
# Deal with sessions that
# cross day boundaries.

df_no_cross =\
df[df["start_datetime"].dt.day\
== df["end_datetime"].dt.day].copy()

df_cross =\
df[df["start_datetime"].dt.day\
!= df["end_datetime"].dt.day]

df_cross_1 = df_cross.copy()

df_cross_2 = df_cross.copy()
```

First, we separate out sessions into those that cross midnight and those that don't. We don't need to do anything about the former and can set that data aside.

For sessions that do cross midnight, we make two copies.

```
df_cross_1["end_time"]\
= datetime.time(\
hour=23, minute=59, second=59)

df_cross_2["start_date"]\
= df_cross_2["start_date"]\
+ datetime.timedelta(days=1)

df_cross_2["start_time"]\
= datetime.time(\
hour=0, minute=0, second=0)
```

For the first copy, we set the end date to just before midnight.

For the second, we set the start time to midnight and increment the date to be the next day.

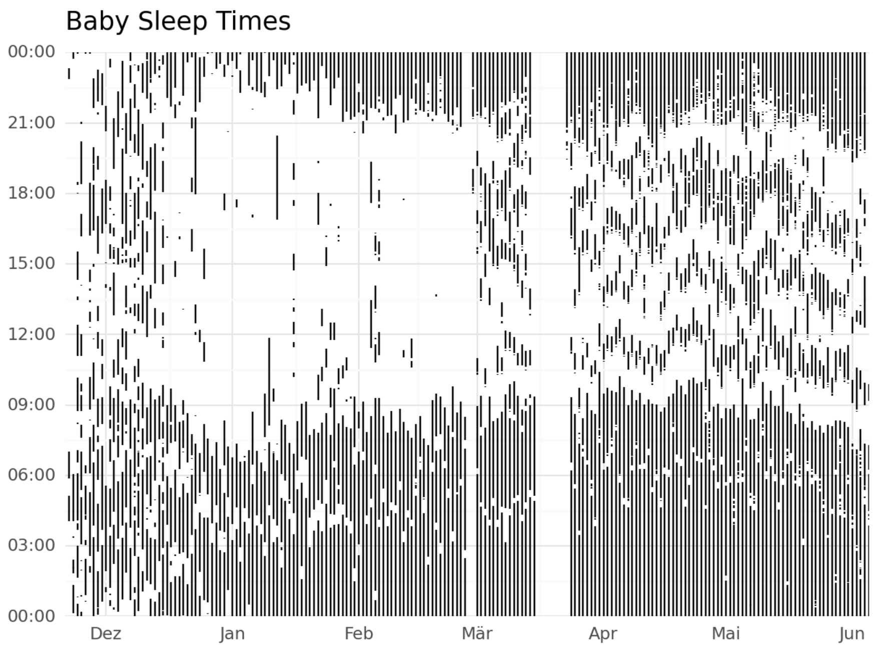
```python
# Combine dataframes

rows_no_cross =\
df_no_cross[["start_date",
"start_time", "end_time"]]

rows_cross_1 =\
df_cross_1[["start_date",
"start_time", "end_time"]]

rows_cross_2 =\
df_cross_2[["start_date",
"start_time", "end_time"]]

rows =\
pd.concat([rows_no_cross,
rows_cross_1,
rows_cross_2])
```

Finally, we combine these dataframes into one new dataframe, which we'll use for plotting.

```python
# Convert back to datetime
# so plotnine can understand it

rows["start_time"] =\
pd.to_datetime(rows["start_time"],
format='%H:%M:%S')

rows["end_time"] =\
pd.to_datetime(rows["end_time"],
format='%H:%M:%S')
```

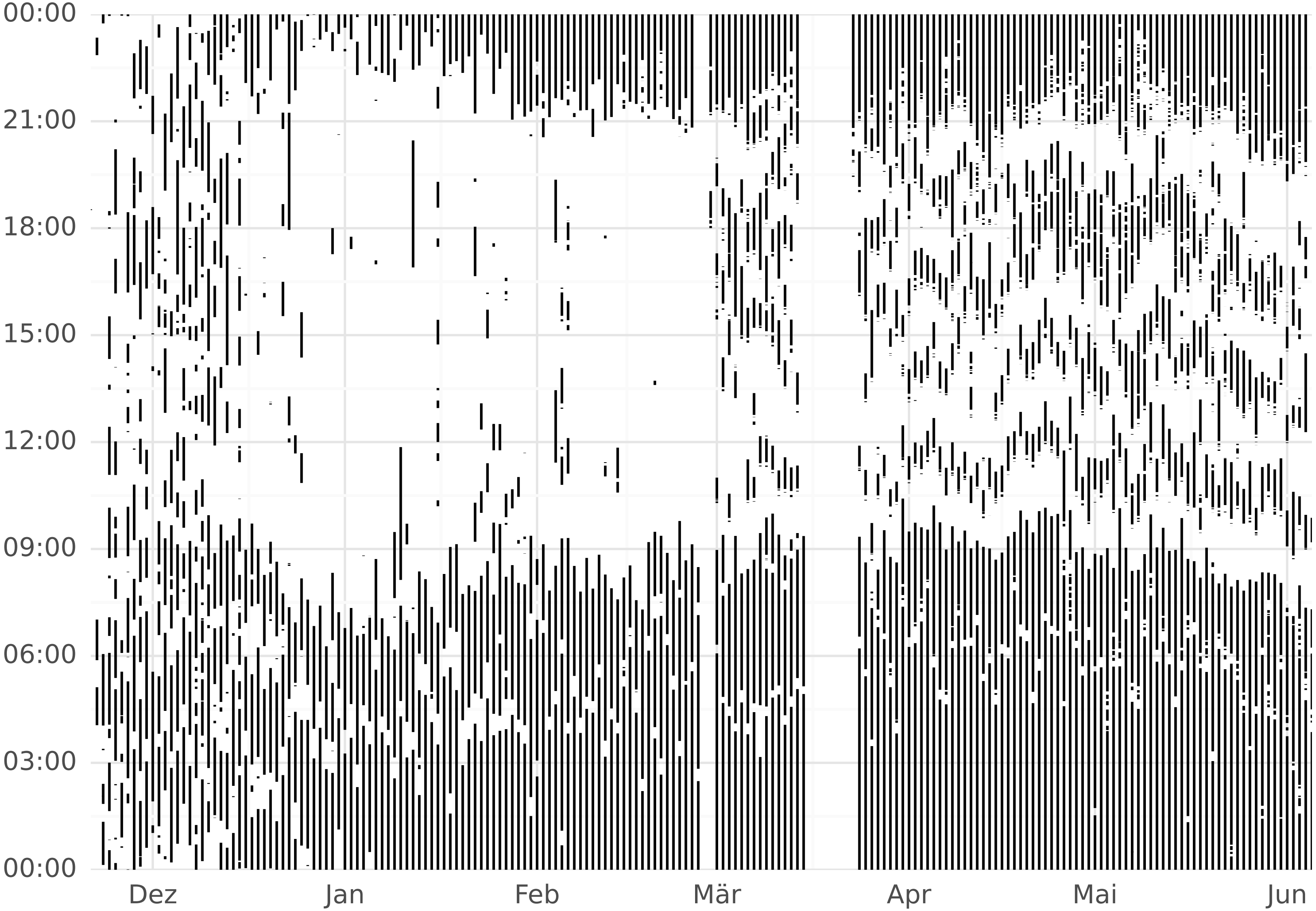
We will use `plotnine` in Python to make the visualization using the `ggplot2` syntax.

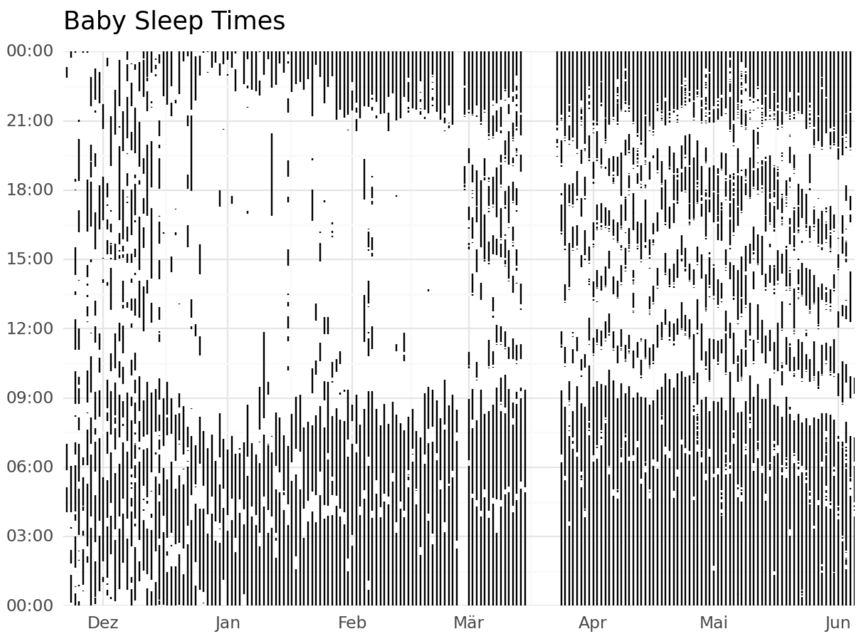
```
plot = (ggplot(data=rows)
  + aes(x="start_date")
  + geom_linerange(aes(
    ymin = "start_time",
    ymax = "end_time"))
  + scale_x_date(name="",
    date_labels="%b",
    expand=(0, 0))
  + scale_y_datetime(
    date_labels="%H:%M",
    expand=(0, 0, 0, 0.0001))
  + ggtitle(
    "Baby Sleep Times")
  + theme_minimal()
  + theme(
    plot_background=\
    element_rect(
      color="white")))
```



There’s clearly some missing data: “We didn’t track lots of naps, and a few days have no data at all.”

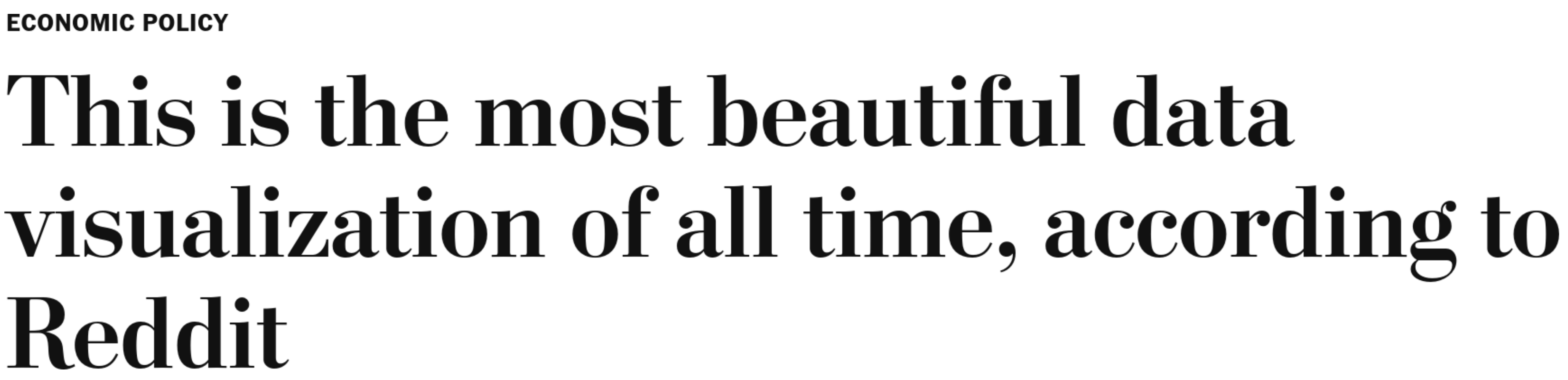
Baby Sleep Times





"That said, some clear patterns emerge. During the first few days, sleep is all over the place, but gradually settles into a routine. Later, nighttime wake-ups become fewer and shorter." (https://www.relevantmisc.com/r/python/2020/05/26/visualizing-baby-sleep/)

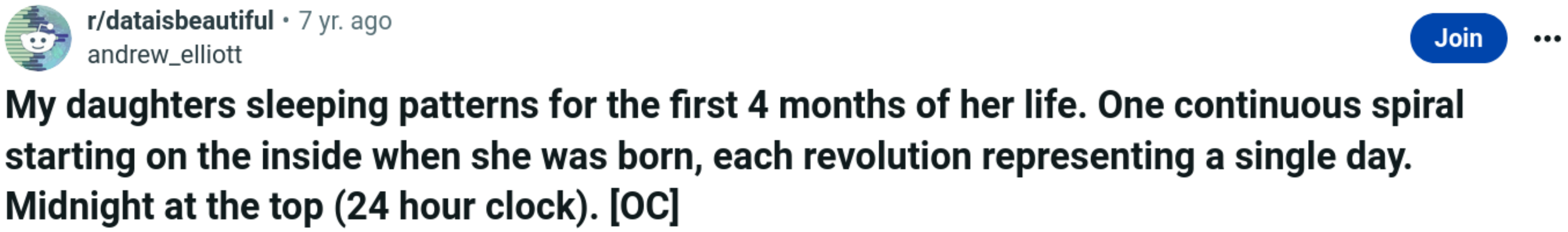
What makes the baby sleep data interesting?



ECONOMIC POLICY

This is the most beautiful data visualization of all time, according to Reddit

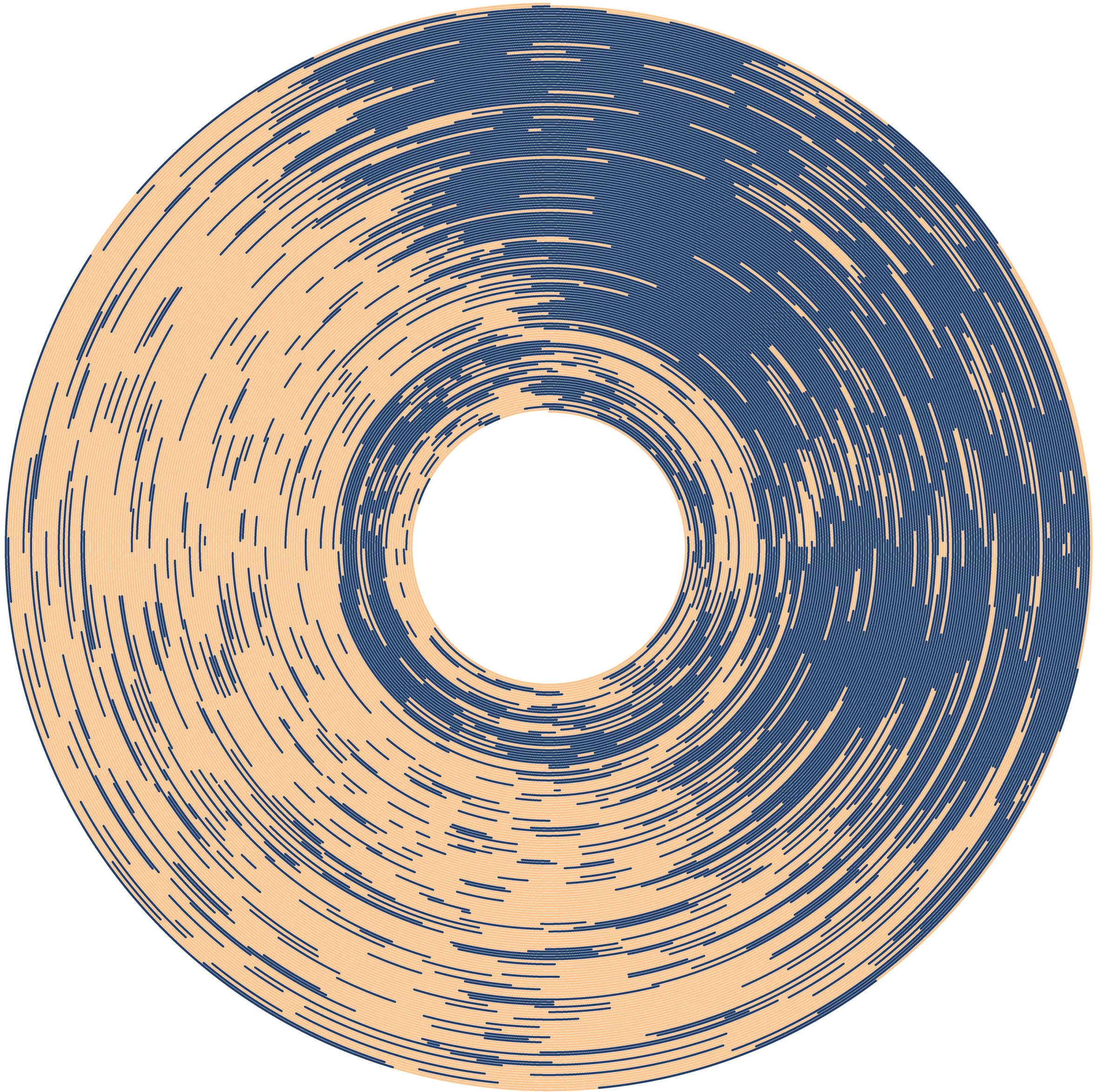
https://www.washingtonpost.com/news/wonk/wp/2017/01/05/what-its-like-to-sleep-like-a-baby-visualized-by-a-dad/



r/dataisbeautiful • 7 yr. ago
andrew_elliott

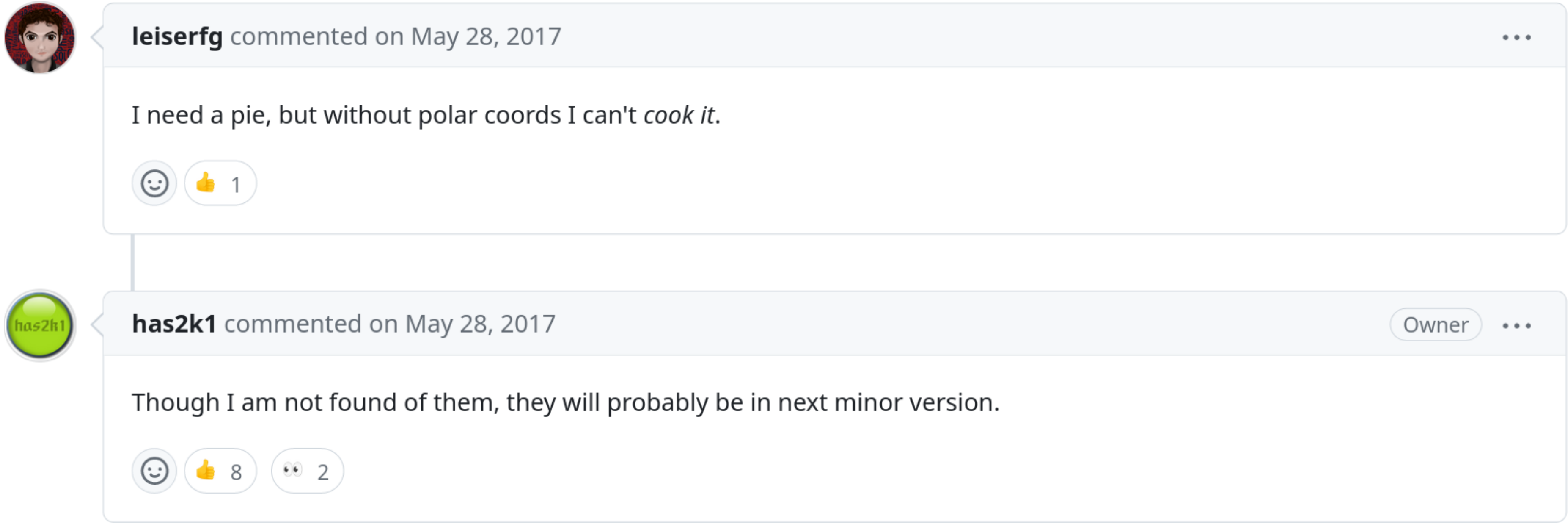
My daughters sleeping patterns for the first 4 months of her life. One continuous spiral starting on the inside when she was born, each revolution representing a single day. Midnight at the top (24 hour clock). [OC]

https://www.reddit.com/r/dataisbeautiful/comments/5l39mu/my_daughters_sleeping_patterns_for_the_first_4/



Short switch to R…

No `coord_polar` in plotnine



leiserfg commented on May 28, 2017

I need a pie, but without polar coords I can't *cook it*.

👍 1

has2k1 commented on May 28, 2017 Owner

Though I am not found of them, they will probably be in next minor version.

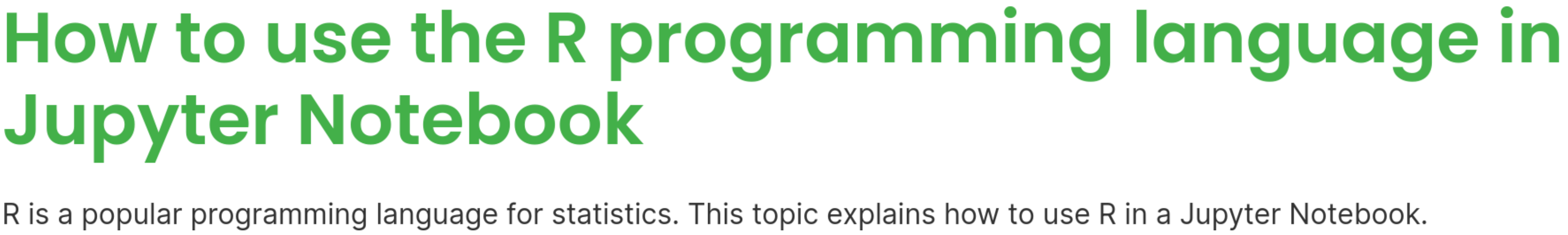
👍 8 👀 2

https://github.com/has2k1/plotnine/issues/10

Maybe at some point?

Make yourself flexible

Using R in Jupyter Notebooks



How to use the R programming language in Jupyter Notebook

R is a popular programming language for statistics. This topic explains how to use R in a Jupyter Notebook.

https://docs.anaconda.com/free/navigator/tutorials/r-lang/

Using Python in Rmarkdown code chunks

~~~
```{r}
library(data.table)
```
~~~

~~~
```{python}
import pandas as pd
```
~~~

```r
library(ggplot2)
library(dplyr)
library(readr)

l1 <- 'https://github.com/dodger487/'
l2 <- 'snoo_plots/raw/master/sleep_data.csv'

df <- read_csv(paste0(l1,l2))

# We need to add rows for when baby is awake
# and do the inverse when the baby is asleep

df <- df %>%
  select(-duration, -asleep, -soothing) %>%
  mutate(session_type = "asleep")
```

Credits: https://www.relevantmisc.com/r/2020/06/01/baby-sleep-radial/

```r
inverse_df <- df %>%
  arrange(start_time) %>%
  mutate(
    start_time_new = end_time,
    end_time_new = lead(start_time),
    session_type = "awake",
    start_time = start_time_new,
    end_time = end_time_new
  ) %>%
  select(-start_time_new, -end_time_new) %>%
  filter(!is.na(start_time) & !is.na(end_time))

# Combine the "awake" and "asleep" rows

df <- rbind(df, inverse_df) %>%
arrange(start_time)
```

Again, we need to break up sessions that cross the midnight boundary into two sessions, one pre-midnight and one-after midnight, so that all sessions only take place in one day.

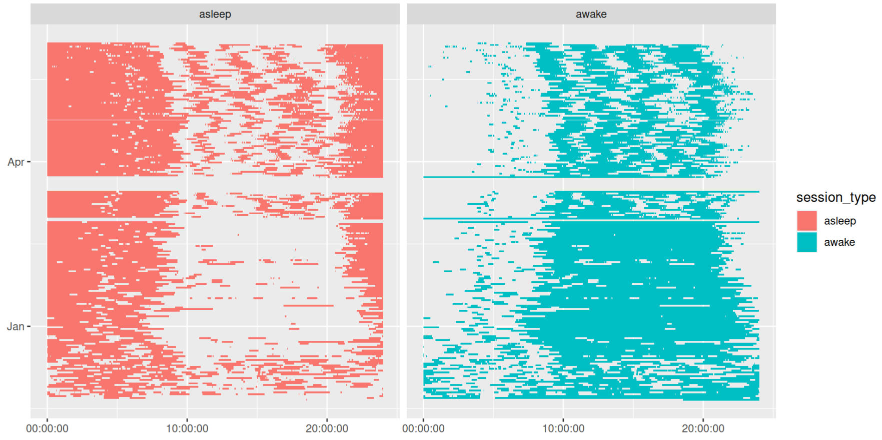
```r
df_no_cross <- df %>%
  filter(date(start_time) == date(end_time)) %>%
  mutate(
    start_date = date(start_time),
    next_date = start_date + days(1),
    start_time = hms::as_hms(start_time),
    end_time = hms::as_hms(end_time))

df_cross <- df %>% filter(date(start_time) != date(end_time))

df_cross_1 <- df_cross %>%
  mutate(
    start_date = date(start_time),
    next_date = start_date + days(1),
    start_time = hms::as_hms(start_time),
    end_time = hms::as_hms("23:59:59")
  )
```

We'll simply break any row that crosses midnight into 2 sessions, one that ends at midnight and one that starts at midnight (as previously in Python, now in R).

```r
df_cross_2 <- df_cross %>%
  mutate(
    start_date = date(end_time),
    next_date = start_date + days(1),
    start_time = hms::as_hms("00:00:00"),
    end_time = hms::as_hms(end_time)
  )

# Combine dataframes

rows <- rbind(
  df_no_cross,
  df_cross_1,
  df_cross_2
)
```

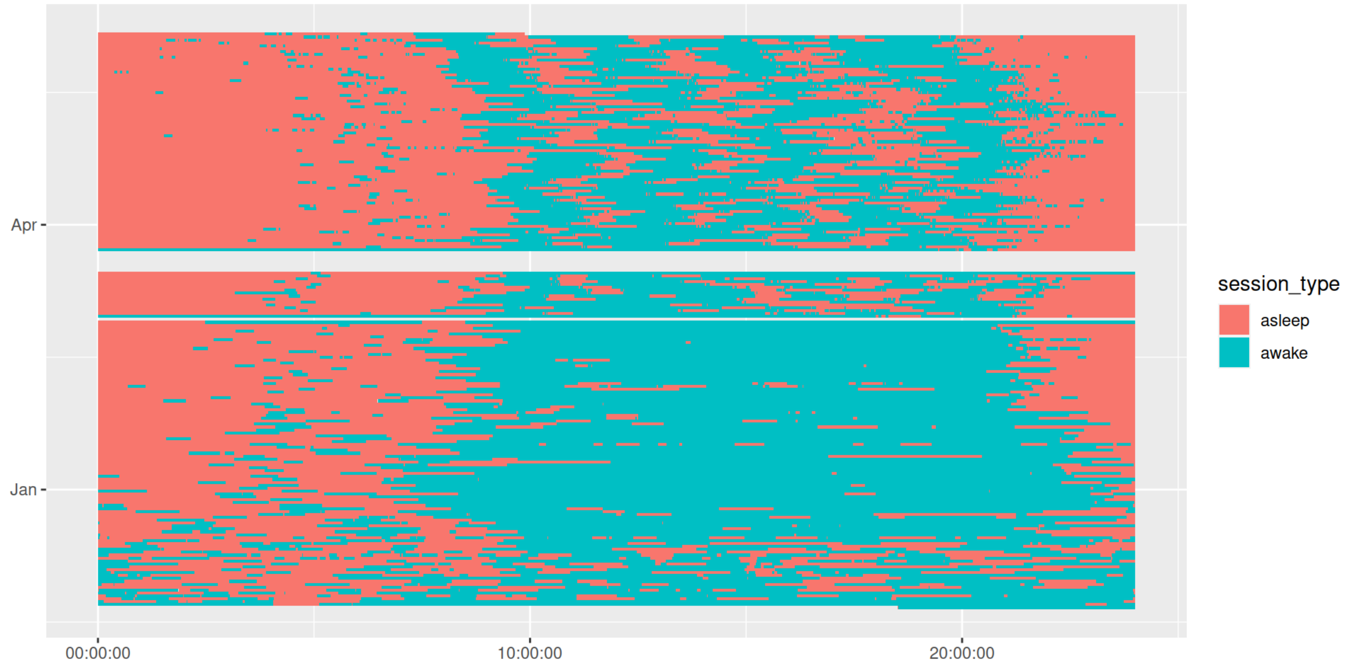
Now on to the visualization! We can use much of the code from `plotnine` before right away.

```r
rows %>%
  ggplot(.) +
  aes(xmin=start_time,
  xmax=end_time,
  ymin=start_date,
  ymax=next_date,
  fill=session_type) +
  geom_rect() +
  facet_wrap(~session_type)
```

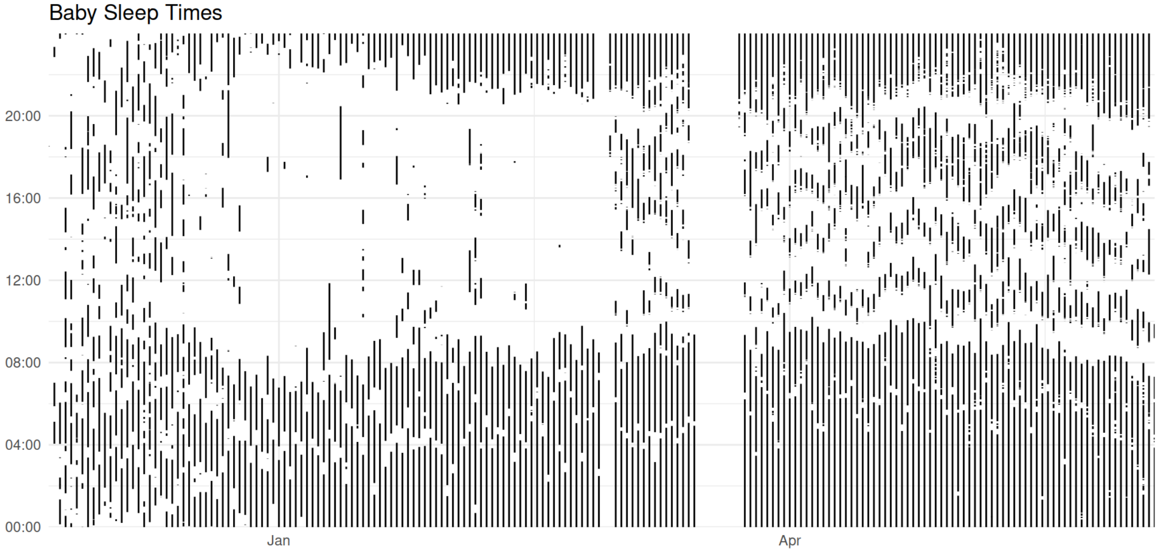


First, let's look at this Cartesian axis plot that faceted by `awake` and `asleep` to check if everything looks OK.

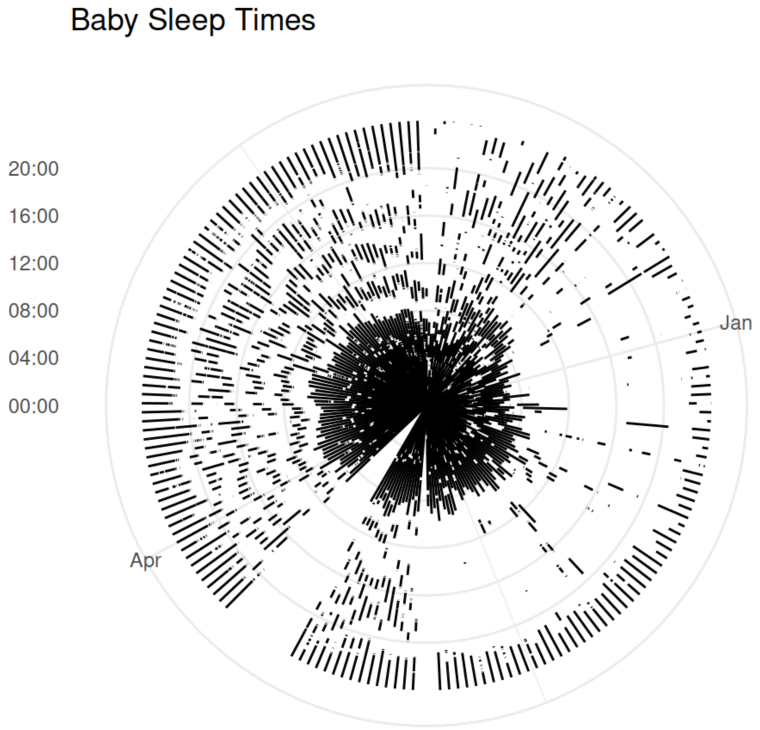
```
rows %>%
  ggplot(.) +
  aes(xmin=start_time,
  xmax=end_time,
  ymin=start_date,
  ymax=next_date,
  fill=session_type) +
  geom_rect() #+
  # facet_wrap(~session_type)
```



```r
p <- (rows %>%
    filter(session_type == "asleep") %>%
    ggplot(aes(x=start_date), data=.)
  + geom_linerange(aes(ymin = start_time, ymax = end_time))
  + scale_x_date(name="", date_labels="%b", expand=c(0, 0))
  + scale_y_time(labels = function(x)
    format(as.POSIXct(x),format = '%H:%M'),
                 expand=c(0, 0, 0, 0.0001))
  + ggtitle("Baby Sleep Times")
  + theme_minimal())
```



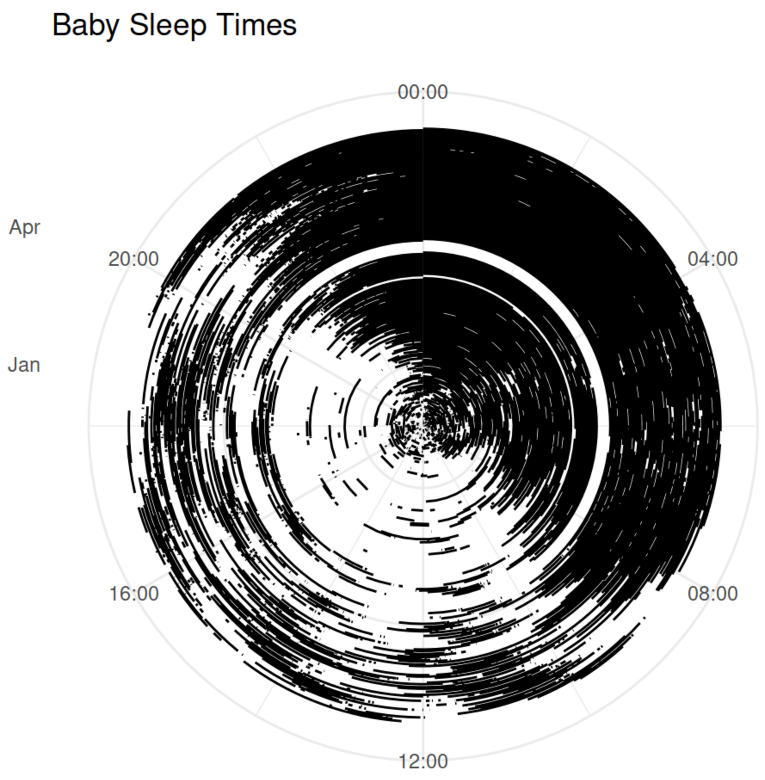
```r
p + coord_polar(start=0)
```



Baby Sleep Times

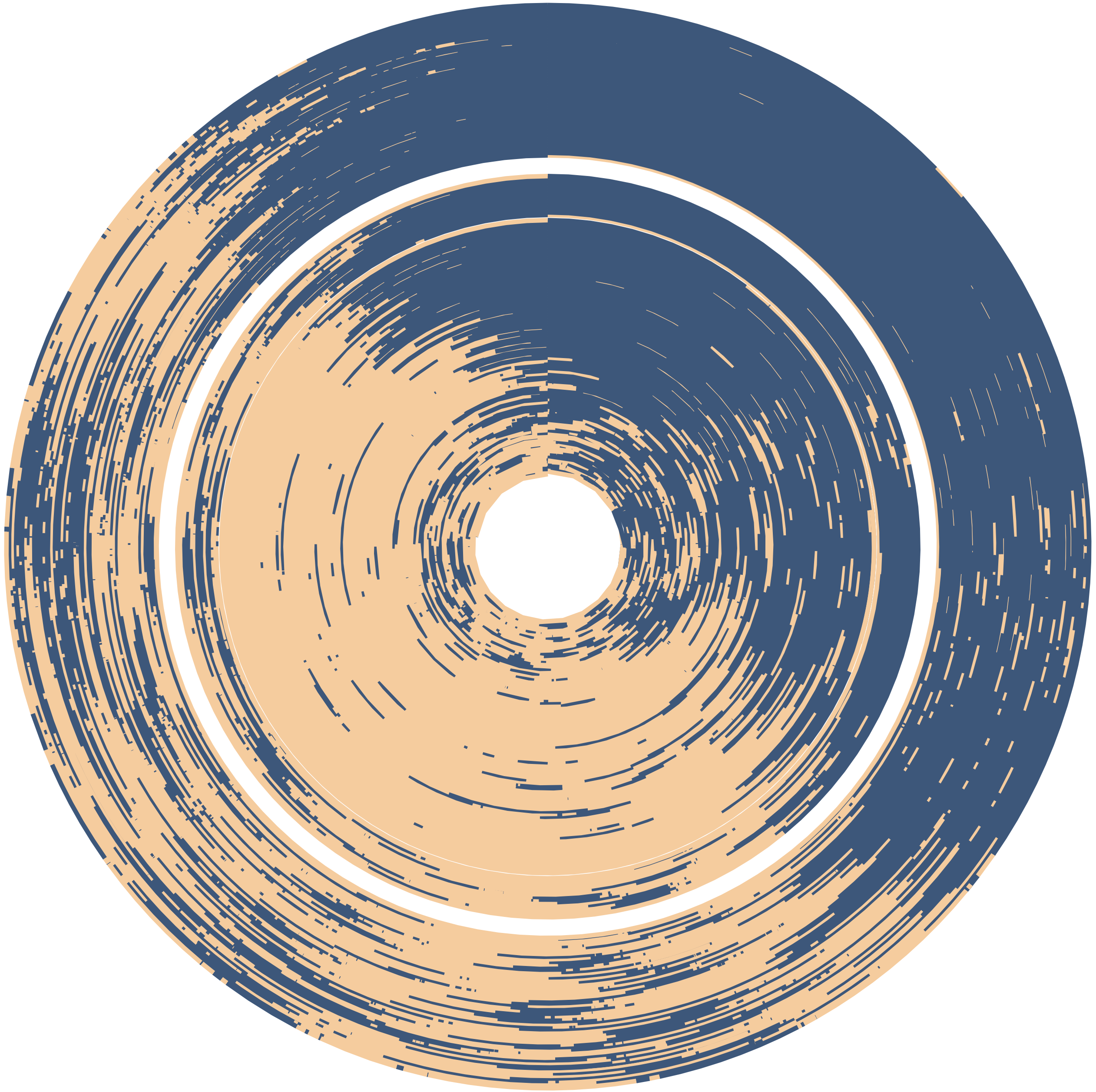
It appears that our axes are flipped: We want the time of day to be the angle, and the radius to be the day.

```r
p + coord_polar(start = 0, theta = "y")
```

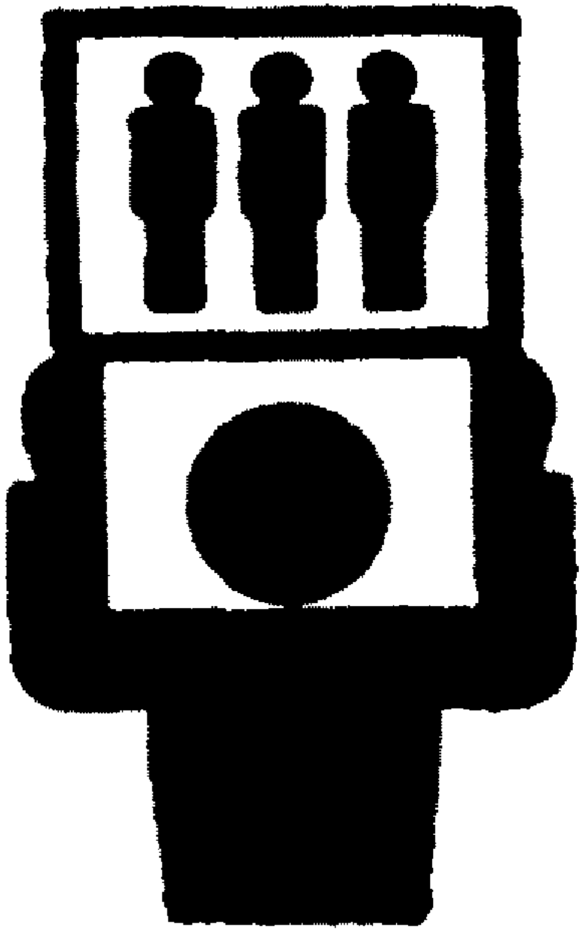


Not bad! Let’s add colors and create the final plot!

```r
# Create custom colors, pulled from original plot
color_awake <- rgb(248/256, 205/256, 160/256)
color_sleep <- rgb(63/256, 89/256, 123/256)

# Create radial plot
rows %>%
  filter(start_date <= "2020-05-20") %>%
  ggplot(aes(x=start_date), data=.) +
    geom_linerange(aes(ymin = start_time,
                       ymax = end_time,
                       color = session_type)) +
  scale_x_date(name="", date_labels="%b", expand=c(0, 28)) +
  scale_y_time(expand=c(0, 0, 0, 0.0001)) +
  scale_color_manual(values = c(color_sleep, color_awake)) +
  theme_void() +
  coord_polar(theta = "y") +
  theme(legend.position = "none")
```



UNIVERSITY
OF MANNHEIM

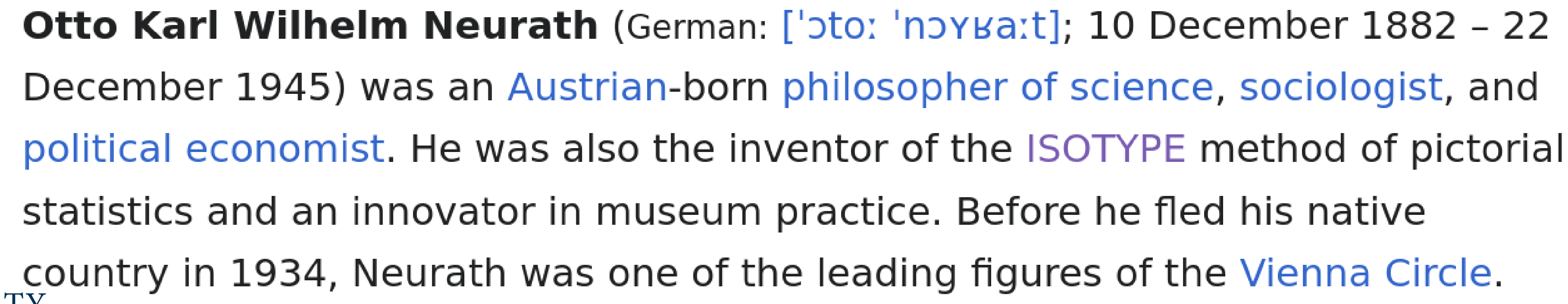


ISOTYPE

International System Of TYpographic Picture Education

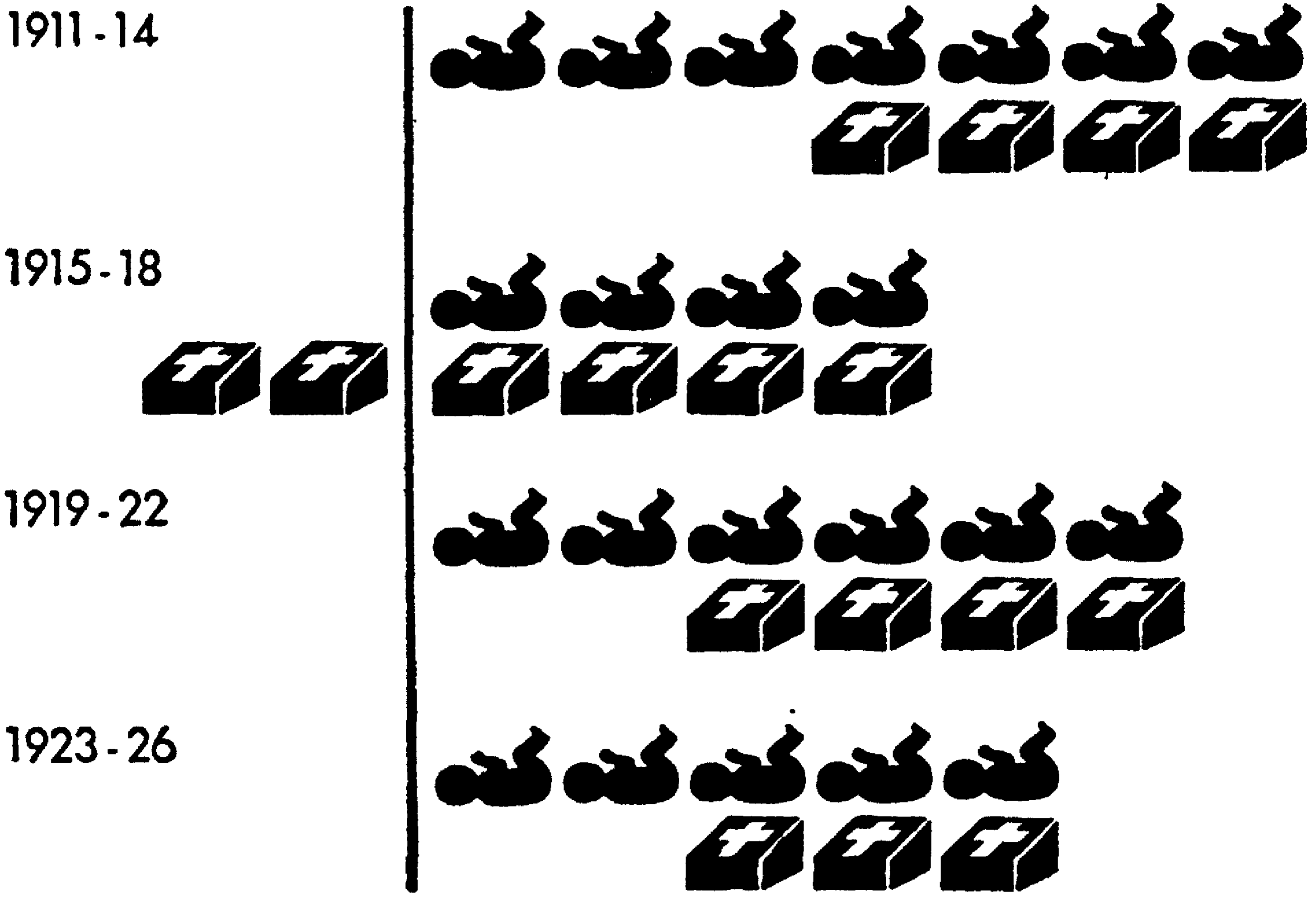
“The Vienna school, on the other hand, postulates: *to remember simplified pictures is better than to forget accurate figures*.” (Neurath, 1973; p. 220)

Neurath, O. (1973). Empiricism and Sociology (M. Neurath & R. S. Cohen, Eds.). Springer Netherlands. https://doi.org/10.1007/978-94-010-2525-6

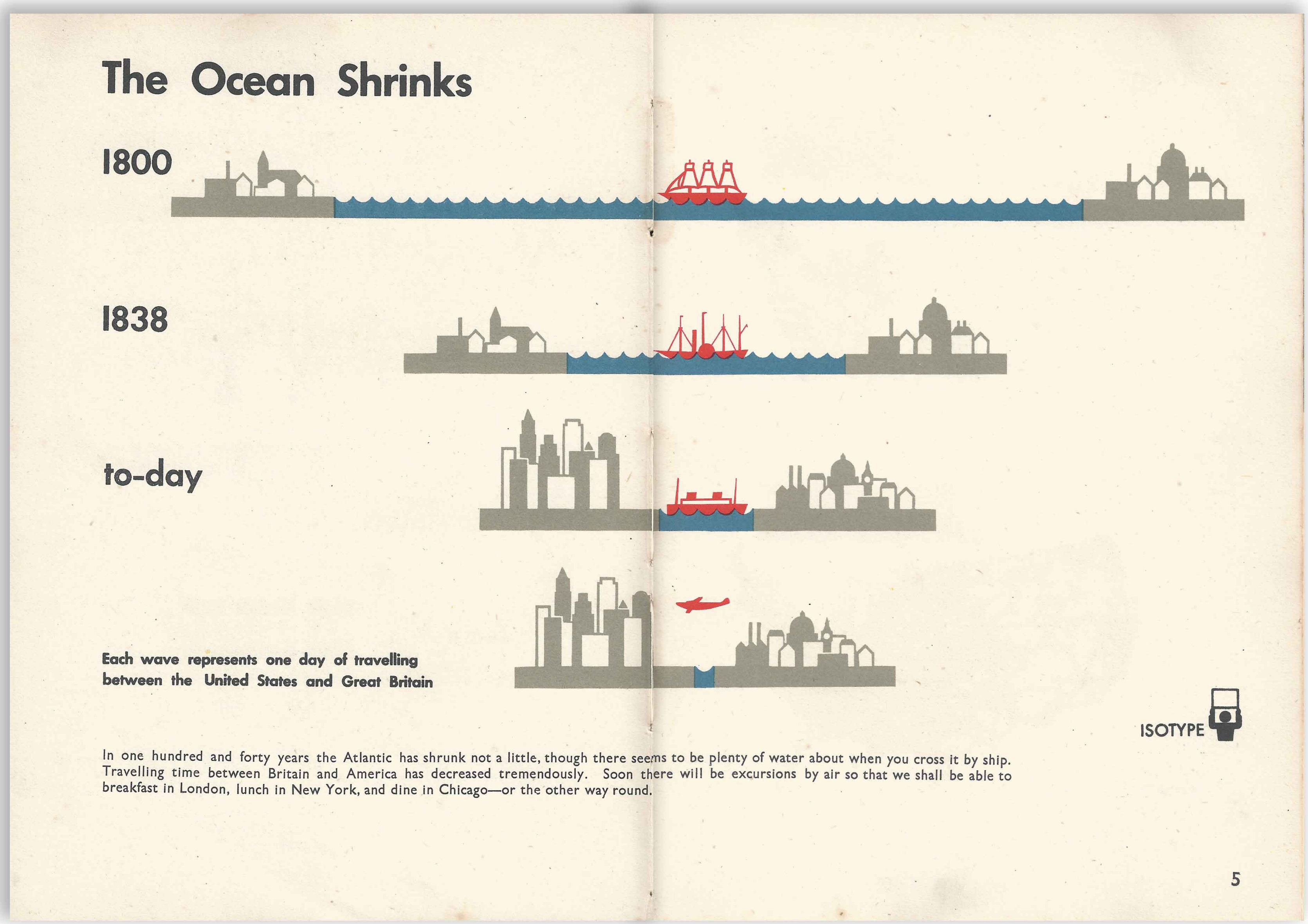


Otto Karl Wilhelm Neurath (German: [ˈɔtoː ˈnɔʏʁaːt]; 10 December 1882 – 22 December 1945) was an Austrian-born philosopher of science, sociologist, and political economist. He was also the inventor of the ISOTYPE method of pictorial statistics and an innovator in museum practice. Before he fled his native country in 1934, Neurath was one of the leading figures of the Vienna Circle.

Births and Deaths in Germany in a Year



1 child for 250,000 births a year
1 coffin for 250,000 deaths a year

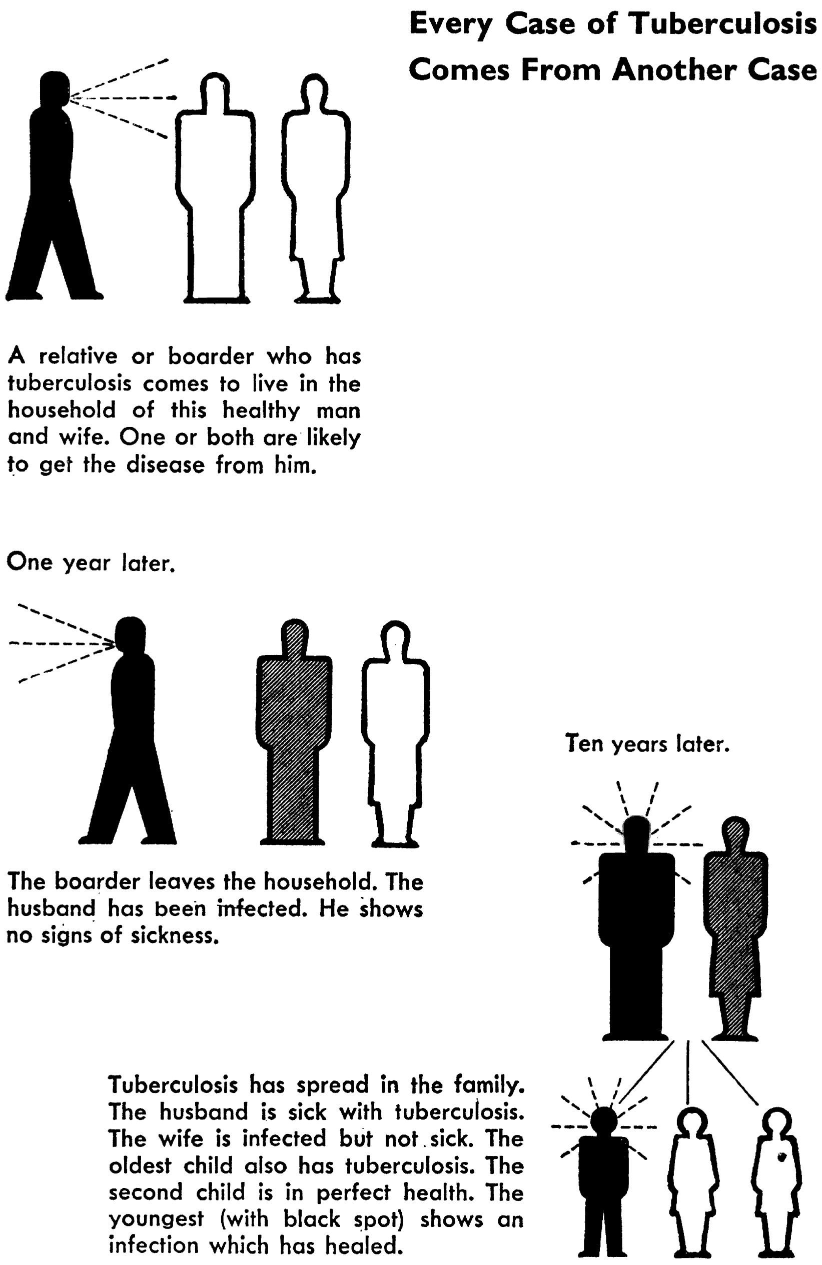


The Ocean Shrinks

Each wave represents one day of travelling between the United States and Great Britain

In one hundred and forty years the Atlantic has shrunk not a little, though there seems to be plenty of water about when you cross it by ship. Travelling time between Britain and America has decreased tremendously. Soon there will be excursions by air so that we shall be able to breakfast in London, lunch in New York, and dine in Chicago—or the other way round.

5



Every Case of Tuberculosis Comes From Another Case

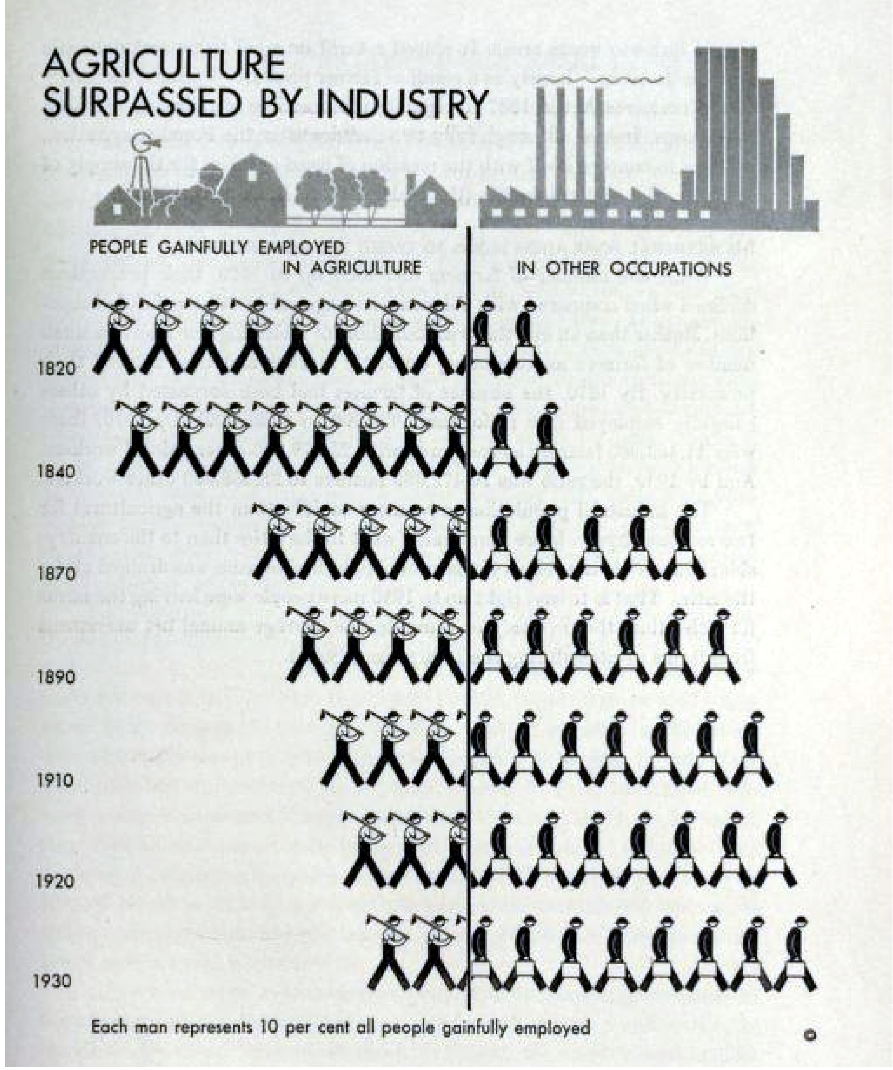
A relative or boarder who has tuberculosis comes to live in the household of this healthy man and wife. One or both are likely to get the disease from him.

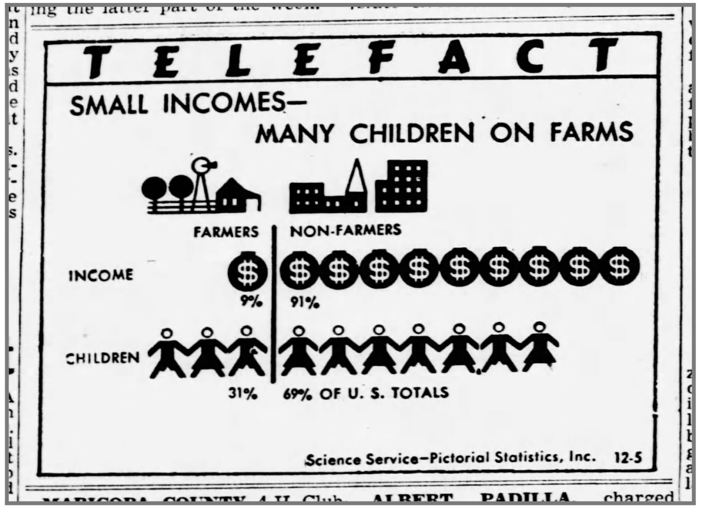
One year later.

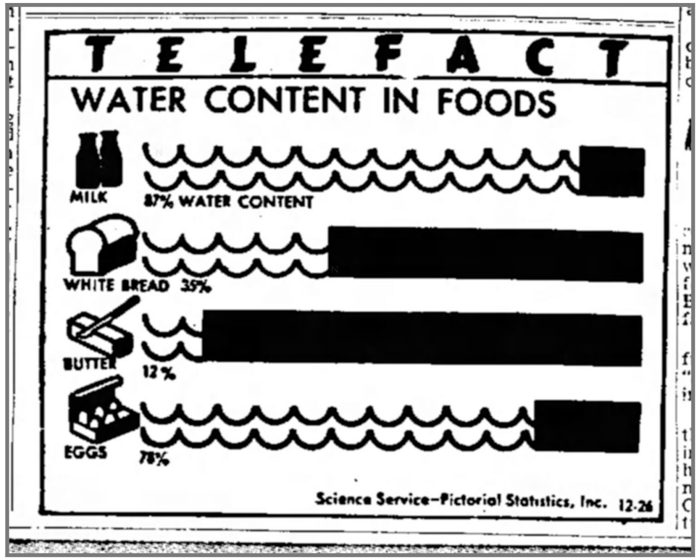
The boarder leaves the household. The husband has been infected. He shows no signs of sickness.

Ten years later.

Tuberculosis has spread in the family. The husband is sick with tuberculosis. The wife is infected but not sick. The oldest child also has tuberculosis. The second child is in perfect health. The youngest (with black spot) shows an infection which has healed.


AGRICULTURE SURPASSED BY INDUSTRY
PEOPLE GAINFULLY EMPLOYED
IN AGRICULTURE
IN OTHER OCCUPATIONS
1820
1840
1870
1890
1910
1920
1930
Each man represents 10 per cent all people gainfully employed


TELEFACT
SMALL INCOMES—
MANY CHILDREN ON FARMS
FARMERS
NON-FARMERS
INCOME
$
9%
$$$$$$$$$
91%
CHILDREN
31%
69% OF U. S. TOTALS
Science Service—Pictorial Statistics, Inc. 12-5


TELEFACT
WATER CONTENT IN FOODS
MILK
87% WATER CONTENT
WHITE BREAD 35%
BUTTER
12%
EGGS
78%
Science Service—Pictorial Statistics, Inc. 12-26

A symbol should

- “Follow principles of good design.
- Be usable in either large or small sizes.
- Represent a general concept, not an individual one.
- Be clearly distinguishable from other symbols.
- Be interesting.
- Be capable of being used as a counting unit.
- Be usable in outline or in silhouette.”

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

Isotype as chartjunk?

Isotype is **not** prone to the distortions measured with the Lie Factor, when certain principles are taken into account:

“The first rule of Isotype is that greater quantities are not represented by an enlarged pictogram but by a greater number of the same-sized pictogram.

In Neurath’s view, variation in size does not allow accurate comparison (what is to be compared – height/length or area?) whereas repeated pictograms, which always represent a fixed value within a certain chart, can be counted if necessary.

Isotype pictograms almost never depicted things in perspective in order to preserve this clarity, and there were other guidelines for graphic configuration and use of colour.”

https://en.wikipedia.org/wiki/Isotype_(picture_language)

Isotype in the context of this course

So far, we modified:

Scales, for example with the logarithm for `gdppercapita`

The coordinate system with `coord_polar` for example

And now we will take a look at geoms and try to emulate ISOTYPE style; as it will turn out, we can pretty easily choose arbitrary symbols to be mapped to our data (that's also often the starting point of how new `ggplot2` extensions get developed)

We heard already earlier about evidence that even "unnecessary" purely *ornamental* chartjunk helps to remember

If we add meaningful symbols directly to graphs, we make visualizations more immediate for the viewer and more self explanatory, because the geometric objects themselves serve as a legend

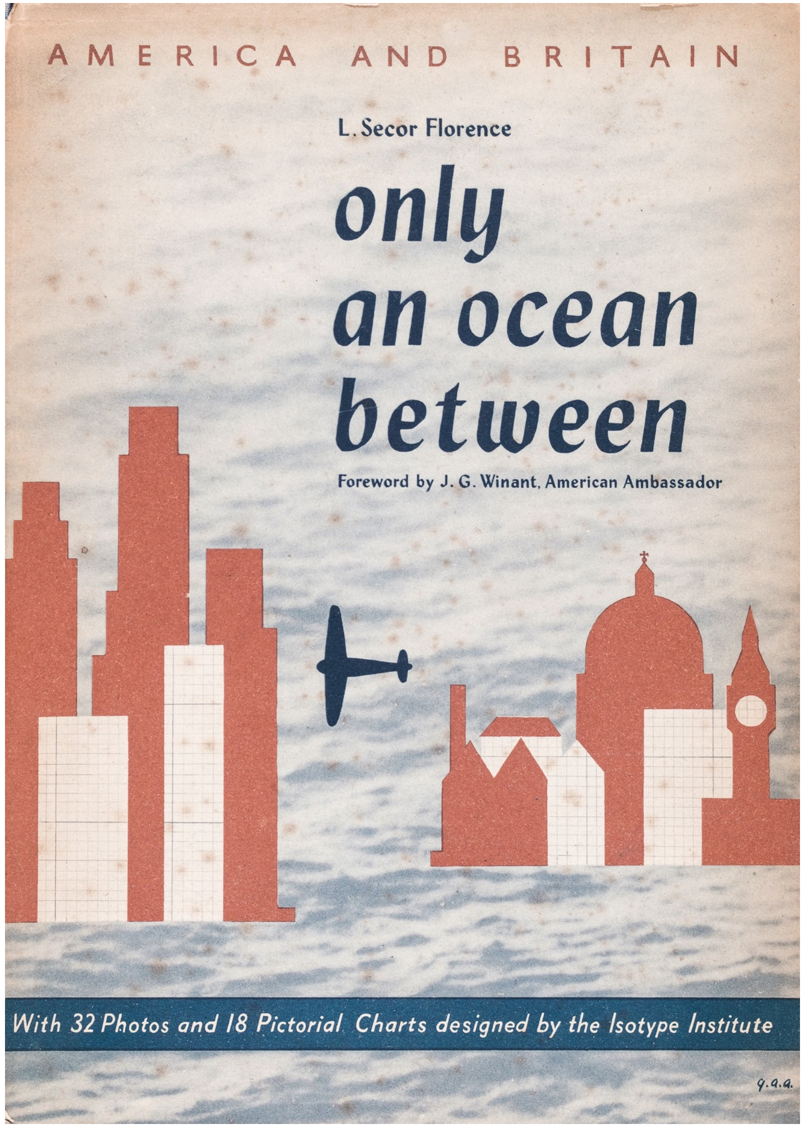
For more information and experiments on the perception of Isotype graphics, check out http://steveharoz.com/research/isotype/

“The first concept to understand is that Isotypes often mix qualitative and quantitative data.

By simplifying the concepts trying to be communicated (often qualitative) and then elaborating with pictograms (quantitative), Isotypes aggregate both types of information into an easy-to-understand message.”

https://nightingaledvs.com/lessons-of-isotype-part-1-only-an-ocean-between/

Our example comes from a series of books that promoted cultural understanding between Britain and its allies during World War II



AMERICA AND BRITAIN

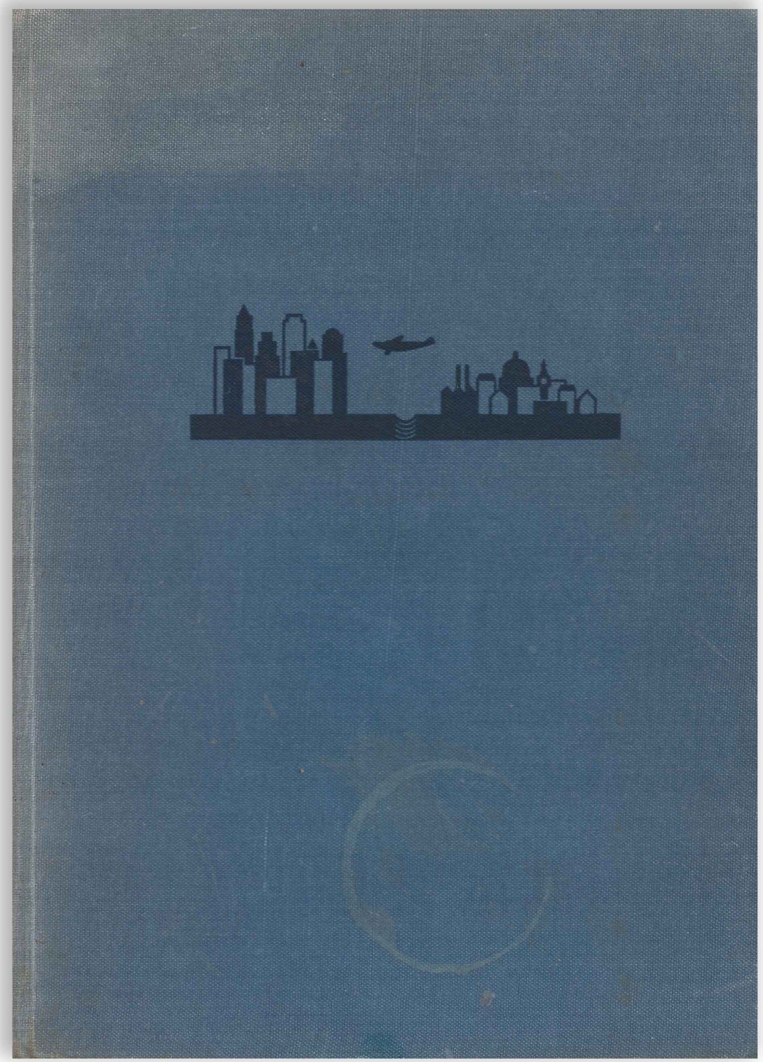
L. Secor Florence

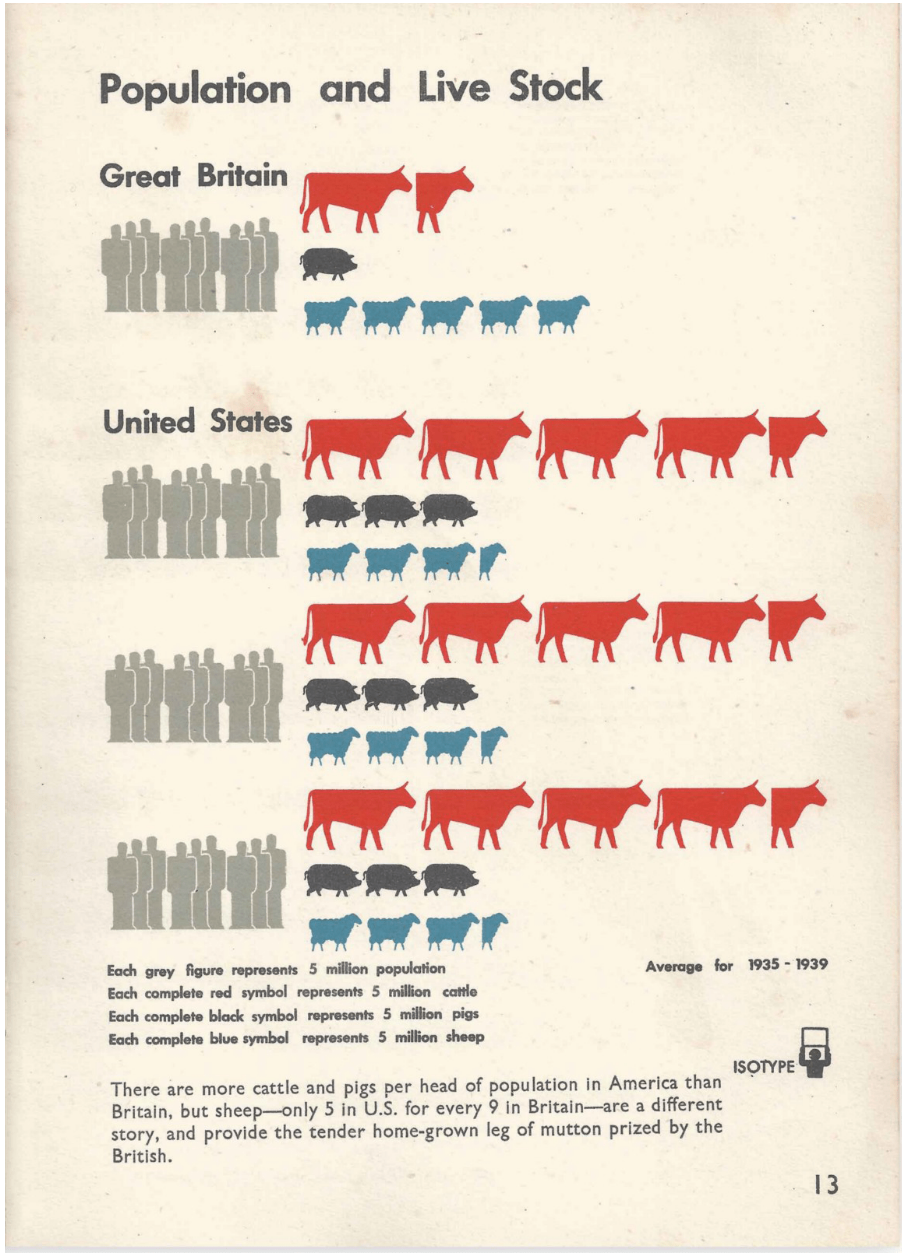
only an ocean between

Foreword by J. G. Winant, American Ambassador

With 32 Photos and 18 Pictorial Charts designed by the Isotype Institute

g.a.a.





Population and Live Stock

Great Britain

United States

Each grey figure represents 5 million population
Each complete red symbol represents 5 million cattle
Each complete black symbol represents 5 million pigs
Each complete blue symbol represents 5 million sheep

Average for 1935 - 1939

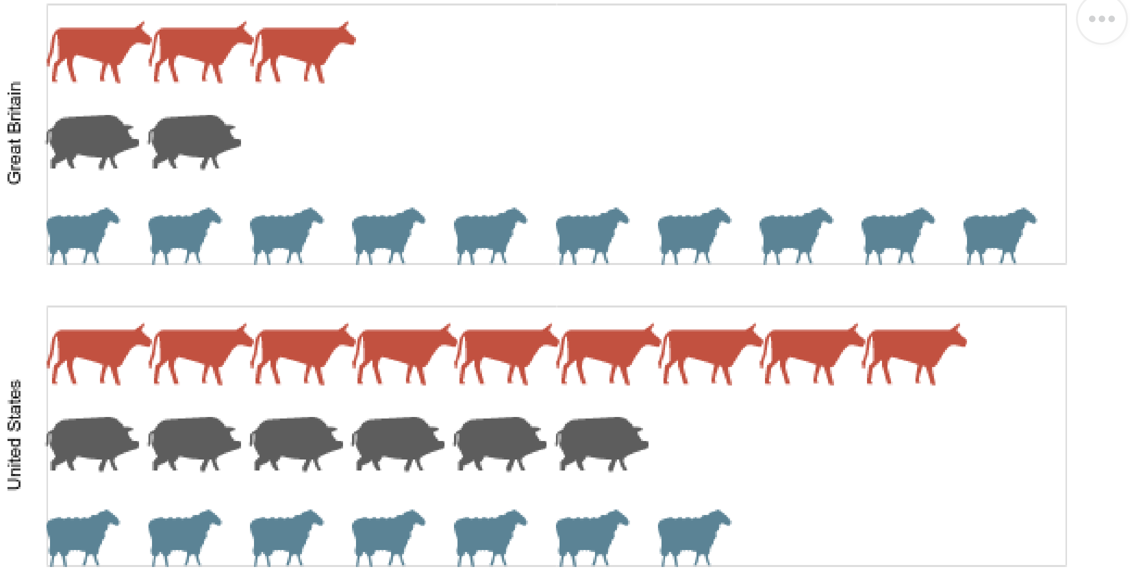
ISOTYPE

There are more cattle and pigs per head of population in America than Britain, but sheep—only 5 in U.S. for every 9 in Britain—are a different story, and provide the tender home-grown leg of mutton prized by the British.

13

```python
import altair as alt
import pandas as pd

source = pd.DataFrame([
      {'country': 'Great Britain', 'animal': 'cattle'},
      {'country': 'Great Britain', 'animal': 'cattle'},
      {'country': 'Great Britain', 'animal': 'cattle'},
      {'country': 'Great Britain', 'animal': 'pigs'},
      {'country': 'Great Britain', 'animal': 'pigs'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'Great Britain', 'animal': 'sheep'},
      {'country': 'United States', 'animal': 'cattle'},
      {'country': 'United States', 'animal': 'cattle'},
      {'country': 'United States', 'animal': 'cattle'},
domains = ['person', 'cattle', 'pigs', 'sheep']

shape_scale = alt.Scale(
    domain=domains,
    range=[
        'M1.7 -1.7h-0.8c0.3 -0.2 0.6 -0.5 0.6 -0.9c0 -0.6 -0.4 -1 -1 -1c-0.
        'M4 -2c0 0.9 -0.7 1.1 -0.8c0.1 -0.1 -0.1 0.5 -0.3 0.7c-0.2 0.2 1.
        'M1.2 -2c0 0 0.7 0 1.2 0.5c0.5 0.5 0.4 0.6 0.5 0.6c0.1 0 0.7 0 0.8
        'M-4.1 -0.5c0.2 0 0.2 0.2 0.5 0.2c0.3 0 0.3 -0.2 0.5 -0.2c0.2 0 0.2
    ]
)

color_scale = alt.Scale(
    domain=domains,
    range=['rgb(162,160,152)',
    'rgb(194,81,64)',
    'rgb(93,93,93)',
    'rgb(91,131,149)']
)
```

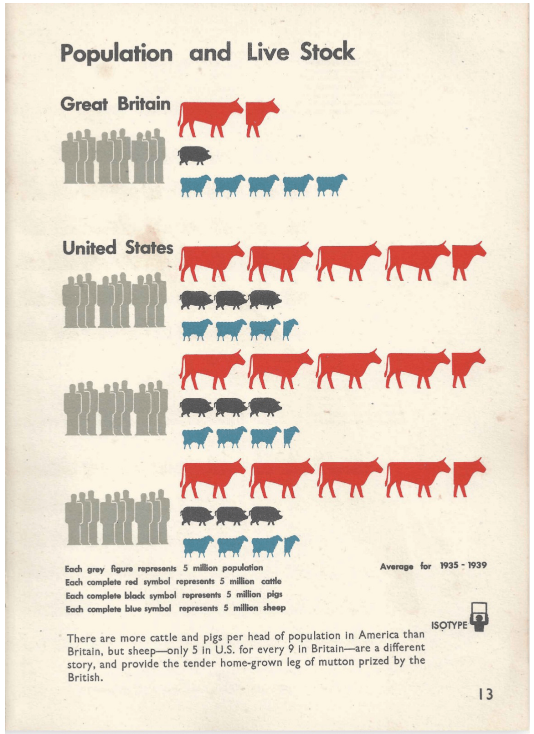
```python
alt.Chart(source).mark_point(filled=True, opacity=1, size=100).encode(
    alt.X('x:O').axis(None),
    alt.Y('animal:O').axis(None),
    alt.Row('country:N').header(title=''),
    alt.Shape('animal:N').legend(None).scale(shape_scale),
    alt.Color('animal:N').legend(None).scale(color_scale),
).transform_window(
    x='rank()',
    groupby=['country', 'animal']
).properties(
    width=550,
    height=140
)
```



“We’ve seen how Isotypes group and organize both qualitative and quantitative data [..] “Population and Live Stock” is a great example of how the design of the grouped icons helps to reveal the story despite groups that are very different sizes.

The focus of the chart is not the overall size of the populations, but a subtle insight to tell a more interesting story.

By breaking the larger US population into three equal rows, it helps to make a more natural comparison between all four rows.”

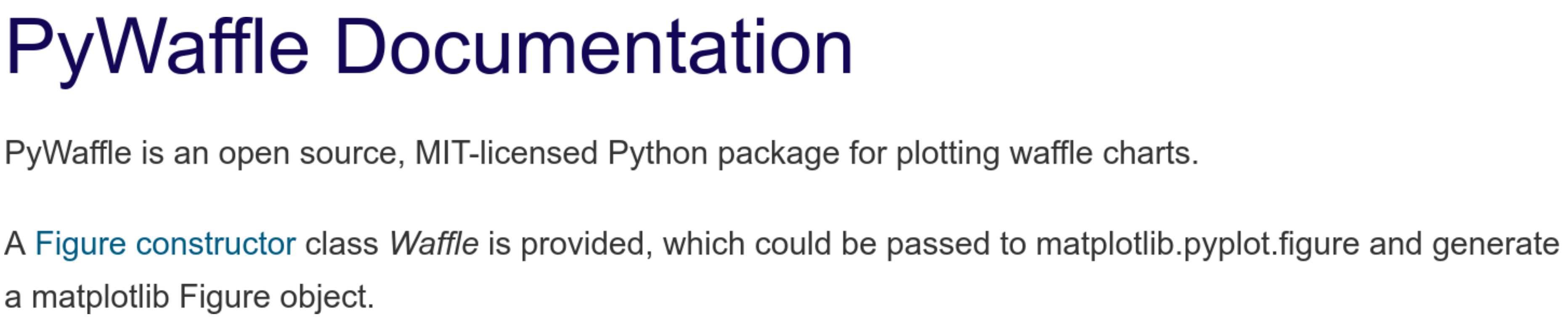


There are more cattle and pigs per head of population in America than Britain, but sheep—only 5 in U.S. for every 9 in Britain—are a different story, and provide the tender home-grown leg of mutton prized by the British.

“UK population in 1939 was 47.5M. US population in 1939 was 130.9M. This chart simplifies these numbers to emphasize the 1:3 ratio. So you can see there are three times as many Americans who eat less sheep and more pigs and cows.”

https://nightingaledvs.com/lessons-of-isotype-part-1-only-an-ocean-between/

Let’s make our lives a little bit easier…



PyWaffle Documentation

PyWaffle is an open source, MIT-licensed Python package for plotting waffle charts.

A Figure constructor class *Waffle* is provided, which could be passed to matplotlib.pyplot.figure and generate a matplotlib Figure object.

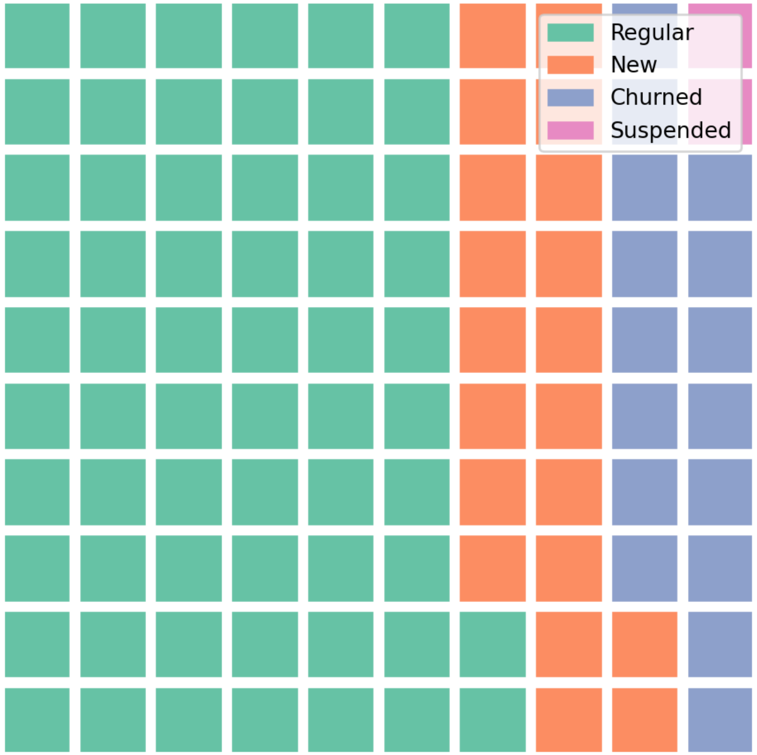
https://github.com/gyli/PyWaffle

https://pywaffle.readthedocs.io/en/latest/index.html

```
pip install pywaffle
```

Often, this is already all we need as in the case of our next example

```python
import pandas as pd

dict_users = {
  'Regular': 62,
  'New': 20,
  'Churned': 16,
  'Suspended': 2
  }
df = pd.Series(dict_users)

from pywaffle import Waffle
import matplotlib.pyplot as plt

fig = plt.figure(
  FigureClass=Waffle,
  figsize=(5,5),
  values=dict_users,
  rows=10
  )

plt.show()
```



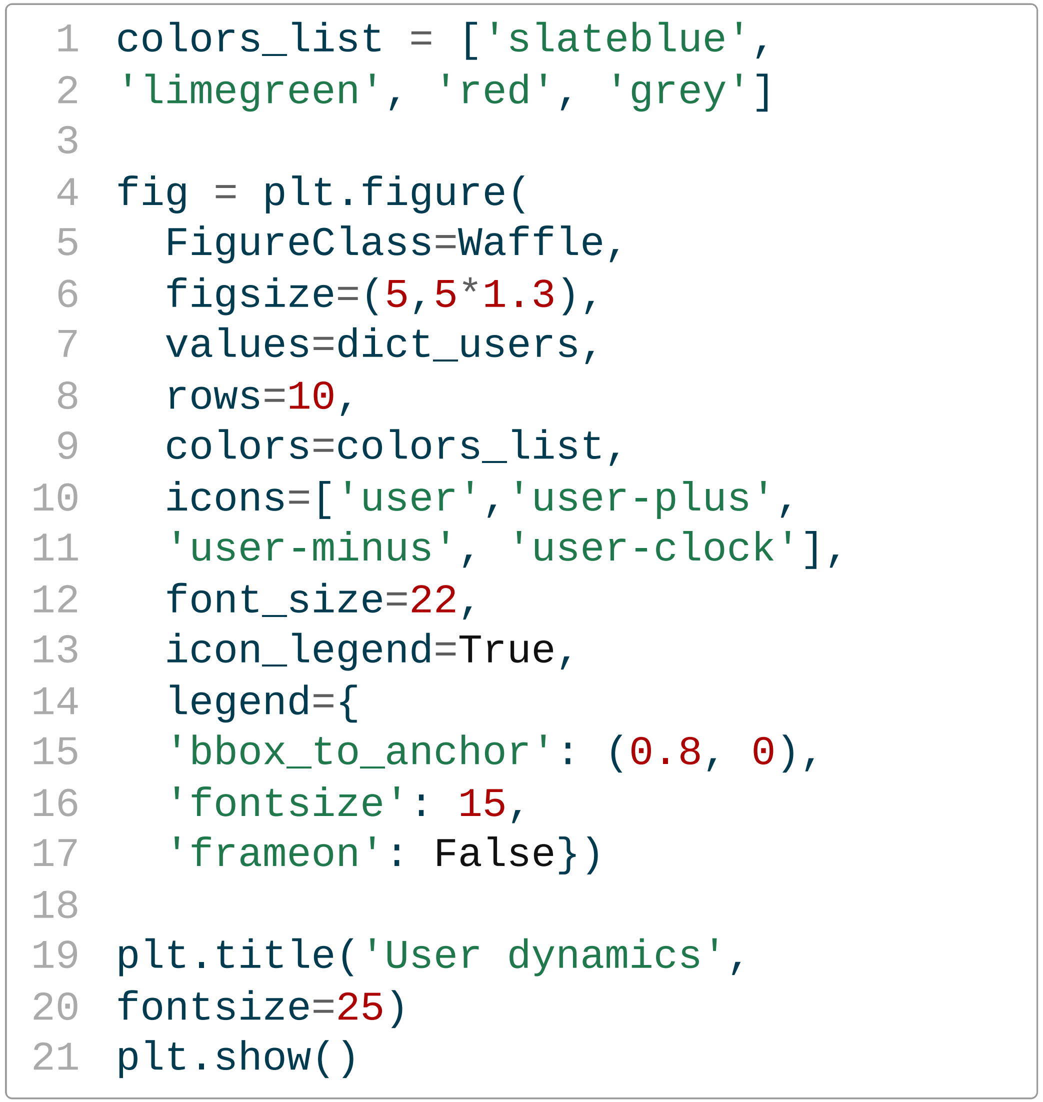
In `PyWaffle`, we can use the `characters` parameter to provide a list of Unicode characters of the same length as the number of categories

Instead, if we want to use the same symbol for all the categories making them differ only by color, we can pass in a string with that character, for example, `characters = '❤️'`

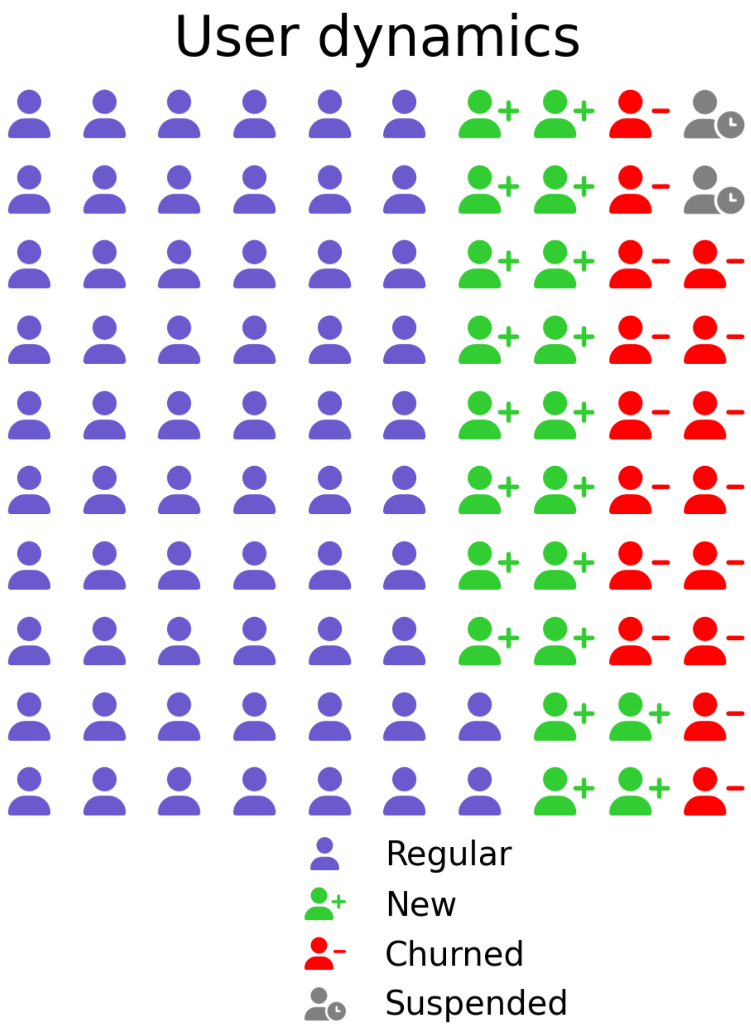
We can also use the `icons` parameter in the same way that accepts a list of strings representing Font Awesome icons:

Font Awesome is the Internet's icon library and toolkit, used by millions of designers, developers, and content creators.

https://fontawesome.com/

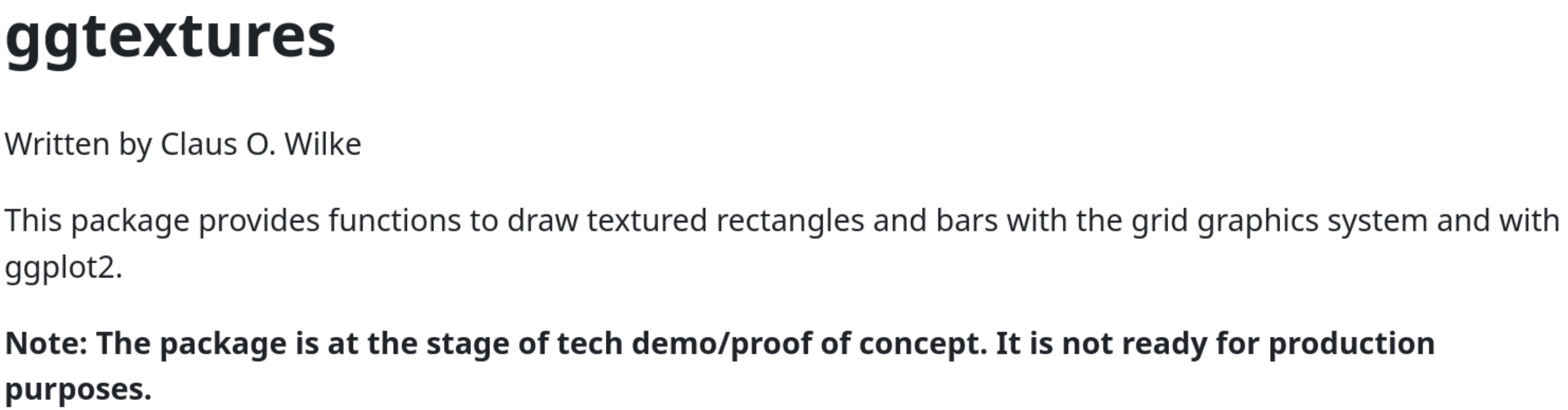


```
colors_list = ['slateblue',
'limegreen', 'red', 'grey']

fig = plt.figure(
  FigureClass=Waffle,
  figsize=(5,5*1.3),
  values=dict_users,
  rows=10,
  colors=colors_list,
  icons=['user','user-plus',
  'user-minus', 'user-clock'],
  font_size=22,
  icon_legend=True,
  legend={
  'bbox_to_anchor': (0.8, 0),
  'fontsize': 15,
  'frameon': False})

plt.title('User dynamics',
fontsize=25)
plt.show()
```



To do similiar things in R

https://github.com/clauswilke/ggtextures



ggtextures

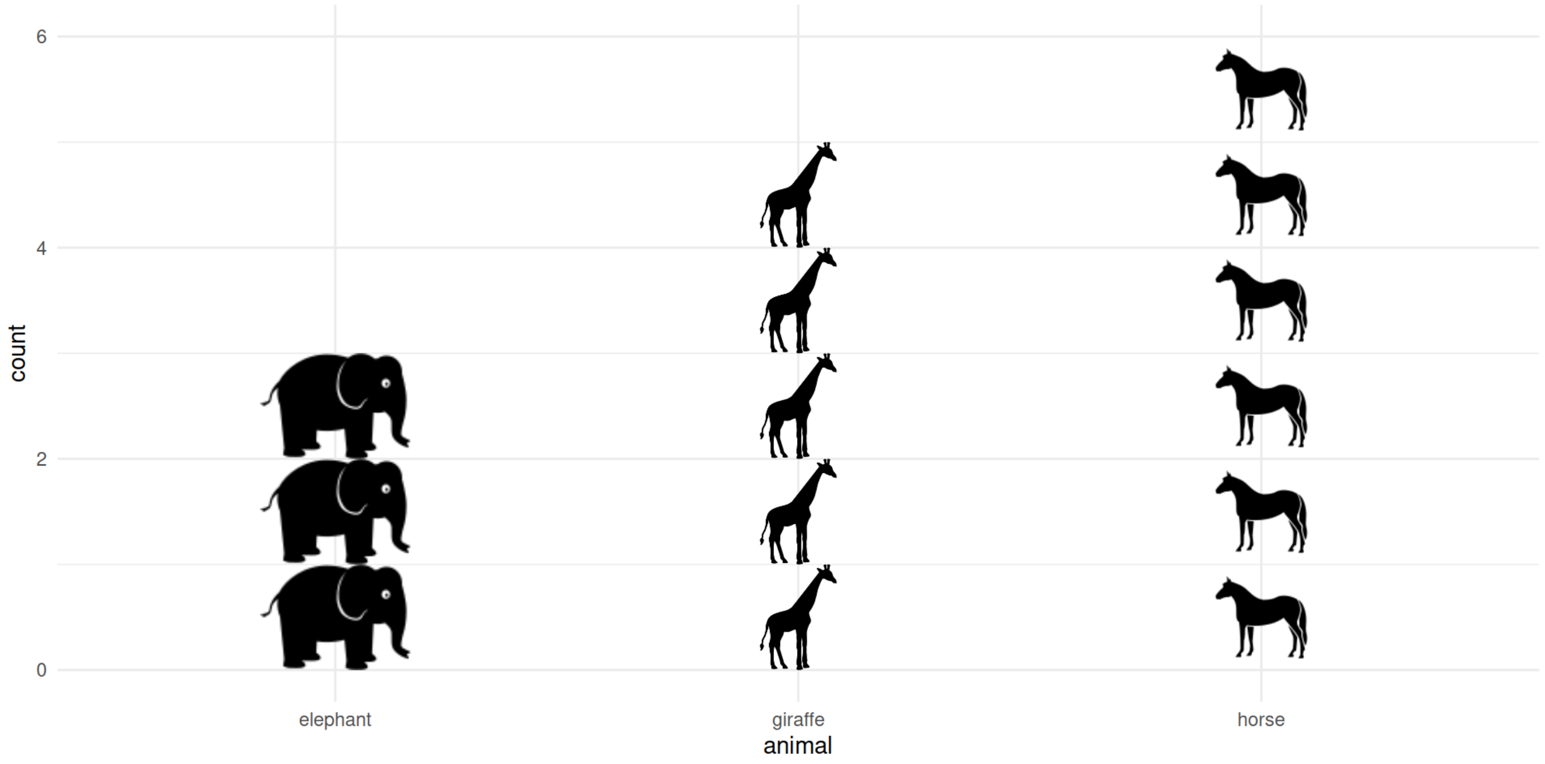
Written by Claus O. Wilke

This package provides functions to draw textured rectangles and bars with the grid graphics system and with ggplot2.

Note: The package is at the stage of tech demo/proof of concept. It is not ready for production purposes.

```r
library(ggtextures)
library(grid)
library(magick)
```

```r
data <- tibble(count = c(5, 3, 6), animal = c("giraffe", "elephant", "horse
  image = list(
    image_read_svg("http://steveharoz.com/research/isotype/icons/giraffe.sv
    image_read_svg("http://steveharoz.com/research/isotype/icons/elephant.s
    image_read_svg("http://steveharoz.com/research/isotype/icons/horse.svg"

ggplot(data, aes(animal, count, image = image)) +
  geom_isotype_col() + theme_minimal()
```



Acknowledgements

https://modley-telefact-1939-1945.tumblr.com/

https://medium.com/nightingale/the-telefacts-of-life-rudolf-modleys-isotypes-in-american-newspapers-1938-1945-d5478faa5647

http://www.thomwhite.co.uk/?p=1303

https://github.com/clauswilke/ggtextures

https://archive.ph/2023.02.07-211651/https://towardsdatascience.com/2-efficient-ways-of-creating-fancy-pictogram-charts-in-python-8b77d361d500