



• Data Visualisation with R & Python

with Gihanna Galindez & Julia Philipp



Sign up to our newsletter for
news about future events!

 www.ghga.de

In cooperation with



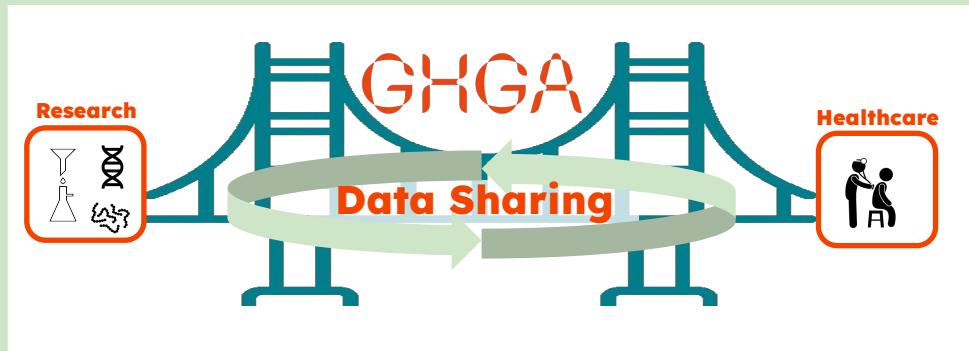
The German Human Genome-Phenome Archive

Mission - GDPR-compliant Human Omics Data Sharing

- Secondary use of clinical omics data in research: biological discovery & replication of findings to show validity
- Translation of research insights: delivering value in genomic medicine
- Involves interactions between diverse communities: computational biologists, molecular geneticists, clinician scientists, physicians, and patients

nfdi
Nationale
Forschungsdaten
Infrastruktur

GHGA



Our next events

A circular graphic with a light blue gradient background. Inside, the letters "GHGA" are written vertically in red. The text "Lecture Series" is at the top in dark teal, followed by "Advances in Data-Driven Biomedicine" in orange. Below that is a photo of a man with glasses and a suit, identified as "Alexander Sczyrba" from "University Bielefeld". The title of his talk, "The de.NBI Cloud - Computing for Life Sciences", is in dark teal. At the bottom, the date and time are given as "17.04.2024 16:00 (CET), virtually".

Lecture Series
Advances in Data-Driven Biomedicine

Alexander Sczyrba
University Bielefeld

"The de.NBI Cloud - Computing
for Life Sciences"

17.04.2024 16:00 (CET), virtually

ghga.de/events



Further Resources

ghga.de/resources/training

The screenshot shows the GHGA Training page with a navigation bar at the top. Below the navigation, there are three main content blocks:

- Sequencing Techniques and Bioinformatic Analyses**: Features a DNA helix icon. Description: Material covering information from DNA and RNA sequencing methods to bioinformatic analyses and analysis workflows. A "Learn more" button is present.
- Ethical and Legal Implications**: Features a document icon. Description: Learn more about ELSI topics, e.g. consent in general, but also GHGA tools that can help you assess existing consent forms. A "Learn more" button is present.
- FAIR and Metadata**: Features a computer monitor icon. Description: Interested in Research Data Management? Want to learn more about research metadata? Curious about what the FAIR principles are and how to apply them in the context of biomedical research? A "Learn more" button is present.



Sign up to our newsletter for news about future events!

<https://t1p.de/rgsse>

The screenshot shows a YouTube channel page for "GHGA Webinar". The channel has 1 video, 1,946 subscribers, and 1,946 views. The video list includes:

- 1 GHGA Webinar DNA & Sequencing with Nick Smith ... (37:04)
- 2 GHGA Webinar DNA & Sequencing & FAIR workflows... (19:46)
- 3 GHGA Webinar - Introduction to Research Consent w/... (27:48)
- 4 GHGA Webinar - GHGA consent tools w/ Simon Parker (21:45)
- 5 GHGA Webinar - Introduction to Research Metadata w/... (23:03)
- 6 GHGA Webinar - Introduction to the GHGA Metadata Model... (23:27)
- 7 GHGA Webinar: Good Statistical Practice for the... (48:43)
- 8 GHGA Webinar - A Beginner's Guide to scRNAseq Analysis... (28:55)
- 9 GHGA Webinar - A Beginner's...

Our Speakers

Gihanna Galindez & Julia Philipp

PhD student in
Bioinformatics - Technische
Universität Braunschweig

Bioinformatician, Chair of
Computational Molecular
Medicine, Technical
University of Munich

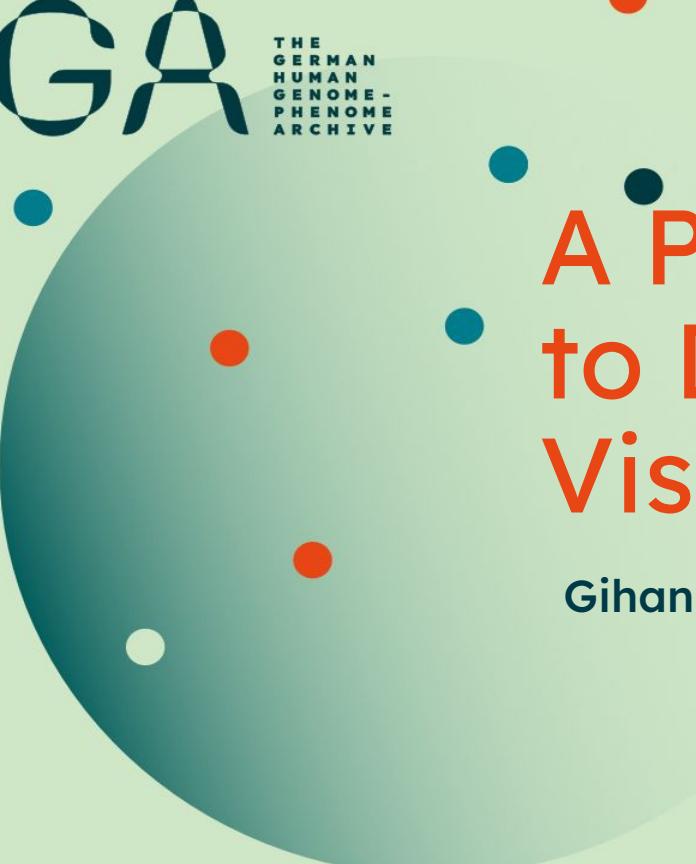
Data steward at the Munich
Data Science Institute



Ph.D. Molecular, Cell, and
Developmental Biology -
University of California,
Santa Cruz

Data Scientist & Data
Manager at EMBL

Training Coordinator for
GHGA since July 2021



A Practical Guide to Data Visualisation

Gihanna Galindez & Julia Philipp

In cooperation with



A quick recap

Data visualization is a form of communication

- Data
- Form of visualization
- Audience
- Location / context



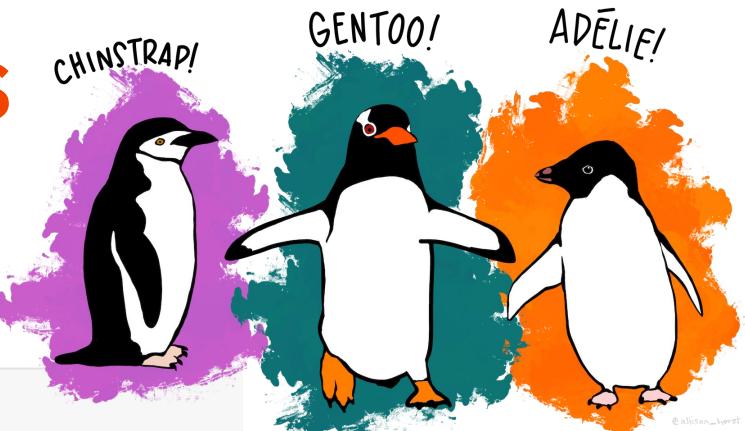
Today

1. Base R plotting
2. The **grammar of graphics** concept
 - a. Introduce, explain and use the ggplot() function
 - b. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use facetting
 - c. Themes & colors
3. Briefly discuss interactive visualization

Q&A

4. Basic plotting in python with matplotlib
5. Seaborn as industry standard & plot customization
6. The grammar of graphics concept for python
7. Interactive visualization

Meet the penguins



```
penguins.head()
```

✓ 0.0s

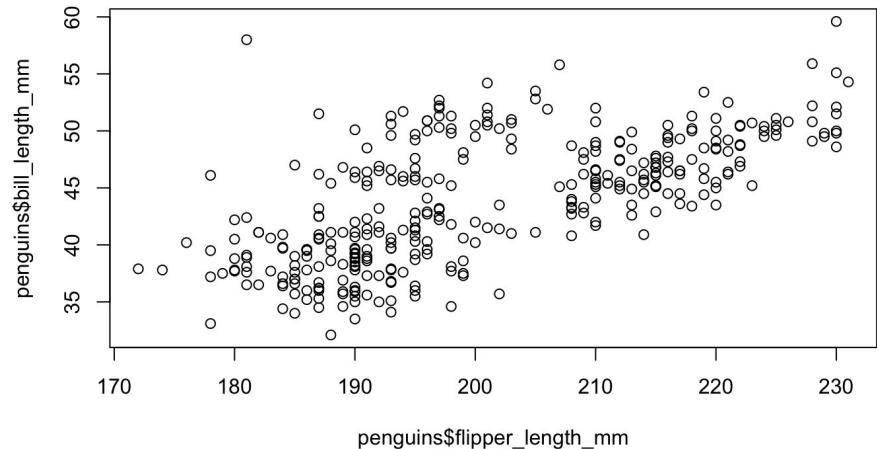
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

base R plotting

canvas model:

a series of instructions that
sequentially fill the plotting canvas

Great for quick data exploration!



```
plot(penguins$flipper_length_mm, penguins$bill_length_mm)
```

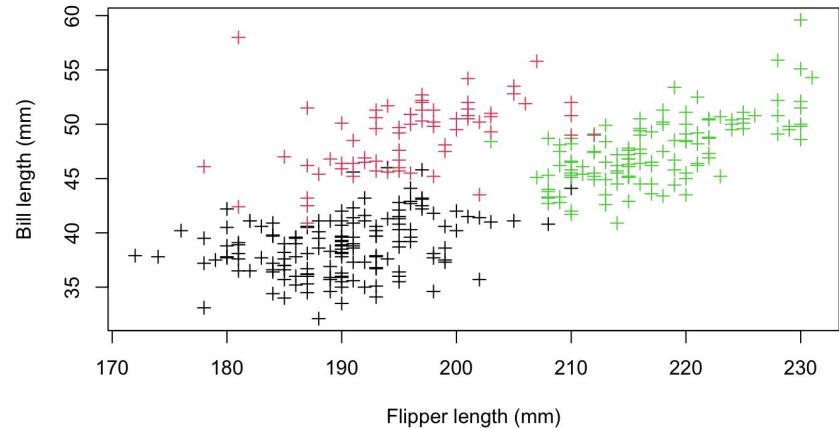
base R plotting

canvas model:

a series of instructions that
sequentially fill the plotting canvas

**Inefficient for customizing &
creating complex plots**

```
plot(penguins$flipper_length_mm, penguins$bill_length_mm, col=penguins$species,  
      ylab = "Bill length (mm)",  
      xlab = "Flipper length (mm)",  
      pch = 3)
```

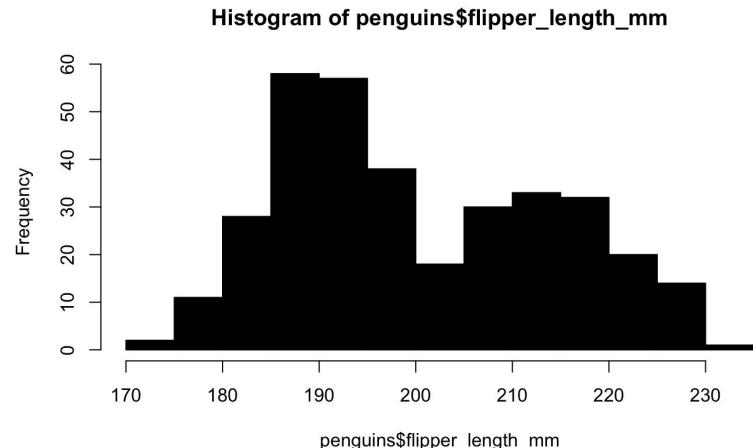


base R plotting

canvas model:

a series of instructions that
sequentially fill the plotting canvas

**Inefficient for customizing &
creating complex plots**

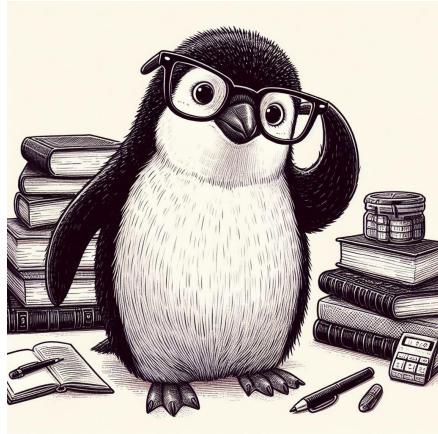


```
hist(penguins$flipper_length_mm, col=penguins$species)
```

base R plotting

Drawbacks:

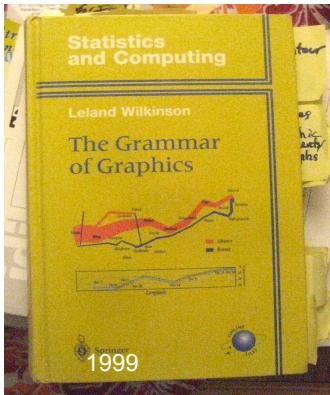
- **Layout choices have to be made at the beginning** with no overview over what may still be coming
- **Different functions for different plot types**, with different interfaces
- Routine tasks can require lots of **boilerplate code**
- **No concept of facets** - Only a **single global coordinate system** allowed per plot
- **Poor default colours**
- **Resizing** often leads to unsatisfactory results



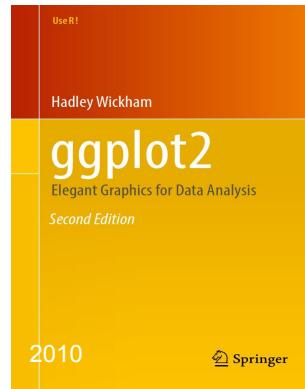
Grammar of Graphics

GHGA

The Grammar of Graphics



Concept **coined by Leland Wilkinson in 1999**.
An **abstraction** which facilitates reasoning and communicating graphics.



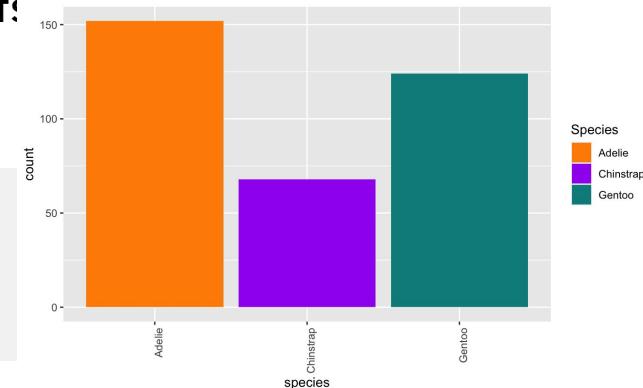
ggplot2 is an implementation of a **layered grammar of graphics** that enables users to independently specify the building blocks of a plot and combine them to create just about any kind of graphical display.

ggplot2 grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,
- **facets** separating subsets of data into multiple subplots
- optional parameter settings e.g. text size, font, alignment, legend positions

```
ggplot(penguins, aes(x=species, fill=species)) +  
  geom_bar(stat = "count") +  
  scale_fill_manual(values=penguin_colors, name="Species") +  
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



geometric objects



`geom_boxplot()` [stat_boxplot\(\)](#)

A box and whiskers plot (in the style of Tukey)



`geom_violin()` [stat_ydensity\(\)](#)

Violin plot



`geom_path()` `geom_line()` `geom_step()`

Connect observations



`geom_point()`

Points



`geom_smooth()` [stat_smooth\(\)](#)

Smoothed conditional means



`geom_raster()` `geom_rect()`

`geom_tile()`

Rectangles



`geom_density()` [stat_density\(\)](#)

Smoothed density estimates

`geom_density_2d()` [stat_density_2d\(\)](#)

Contours of a 2d density estimate

Cheat sheet:

<https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

ggplot () template

```
ggplot(data = <default data set>,
  aes(x = <default x axis variable>,
      y = <default y axis variable>,
      ... <other default aesthetic mappings>),
  ... <other plot defaults>) +
  geom_<geom type>(aes(size = <size variable for this geom>,
                        ... <other aesthetic mappings>),
                     data = <data for this point geom>,
                     stat = <statistic string or function>,
                     position = <position string or function>,
                     color = <"fixed color specification">,
                     ... <other arguments, possibly passed to the _stat_ function>) +
  scale_<aesthetic>_<type>(name = <"scale label">,
                            breaks = <where to put tick marks>,
                            labels = <labels for tick marks>,
                            ... <other options for the scale>) +
  theme(plot.background = element_rect(fill = "gray"),
        ... <other theme elements>)
```

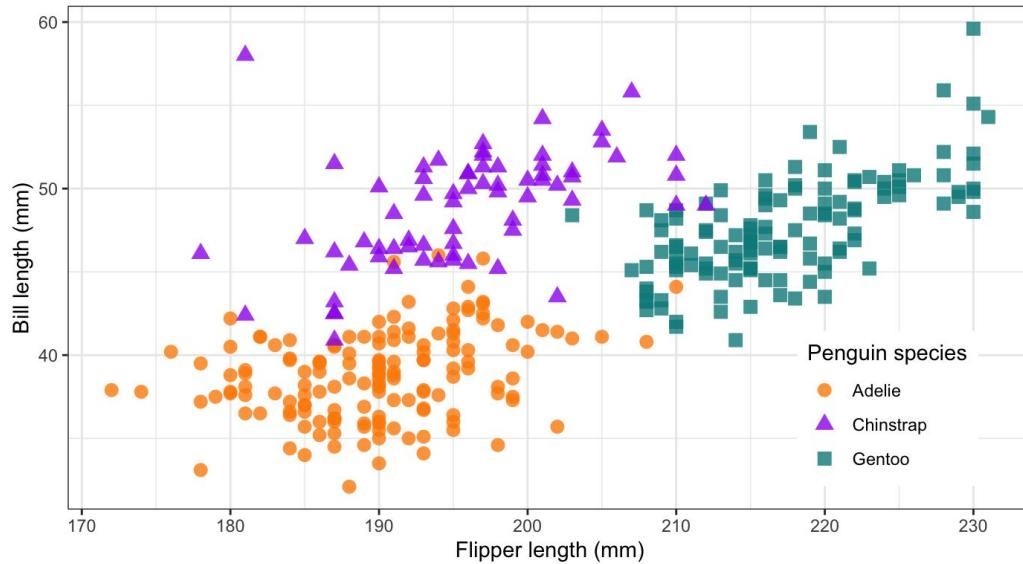
Cheat sheet:

<https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf>

ggplot vs base R

Flipper and bill length

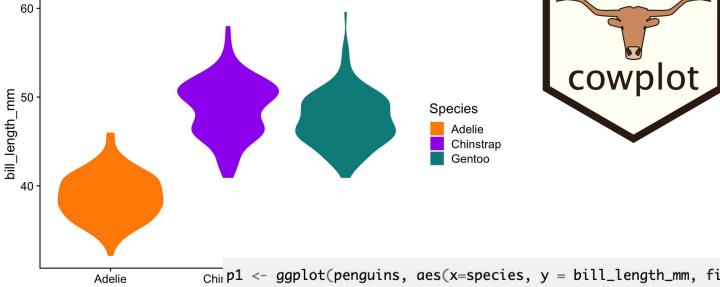
Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER



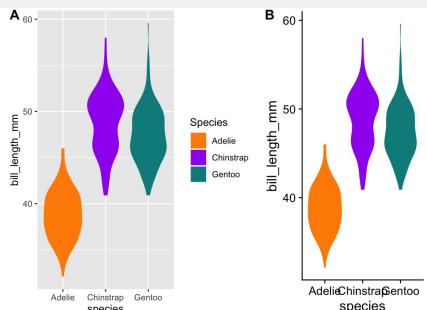
Themes can change the look

```
library(cowplot)
```

```
g + theme_cowplot()
```



```
Chi:p1 <- ggplot(penguins, aes(x=species, y = bill_length_mm, fill=species)) +  
  spec  
  geom_violin(col = NA) +  
  scale_fill_manual(values=penguin_colors, name="Species")  
  
p2 <- ggplot(penguins, aes(x=species, y = bill_length_mm, fill=species)) +  
  geom_violin(col = NA) +  
  scale_fill_manual(values=penguin_colors, name="Species") +  
  theme_cowplot()  
  
plot_grid(p1, p2, labels = c('A', 'B'), rel_widths = c(1, 1))
```



<https://cran.r-project.org/web/packages/cowplot/vignettes/introduction.html>

Create your own theme:

```
#define gpr_theme() function  
theme_gpr <- function(){  
  font <- "Georgia" #assign font family up front  
  
  theme_minimal() %>%replace% #replace elements we want to change  
  theme(  
  
    #grid elements  
    panel.grid.major = element_blank(), #strip major gridlines  
    panel.grid.minor = element_blank(), #strip minor gridlines  
    axis.ticks = element_blank(), #strip axis ticks  
  
    #since theme_minimal() already strips axis lines,  
    #we don't need to do that again  
  
    #text elements  
    plot.title = element_text( #title  
      family = font,  
      size = 20,  
      face = "bold",  
      hjust = 0,  
      vjust = 2), #raise slightly  
  
    plot.subtitle = element_text( #subtitle  
      family = font,  
      size = 14), #font family  
      #font size  
    plot.caption = element_text( #caption  
      family = font,  
      size = 9,  
      hjust = 1), #font family  
      #font size  
      #right align  
    axis.title = element_text( #axis titles  
      family = font,  
      size = 10), #font family  
      #font size  
    axis.text = element_text( #axis text  
      family = font,  
      size = 9), #font family  
      #font size  
    axis.text.x = element_text( #margin for axis text  
      margin = margin(5, b = 18))  
  
    #since the legend often requires manual tweaking  
    #based on plot content, don't define it here  
  )  
}
```

<https://rpubs.com/mclaire19/ggplot2-custom-themes>



GHGA

How to show more than 2D?

3-5D: aesthetics allow to show more than 2D geom-point's

aesthetics

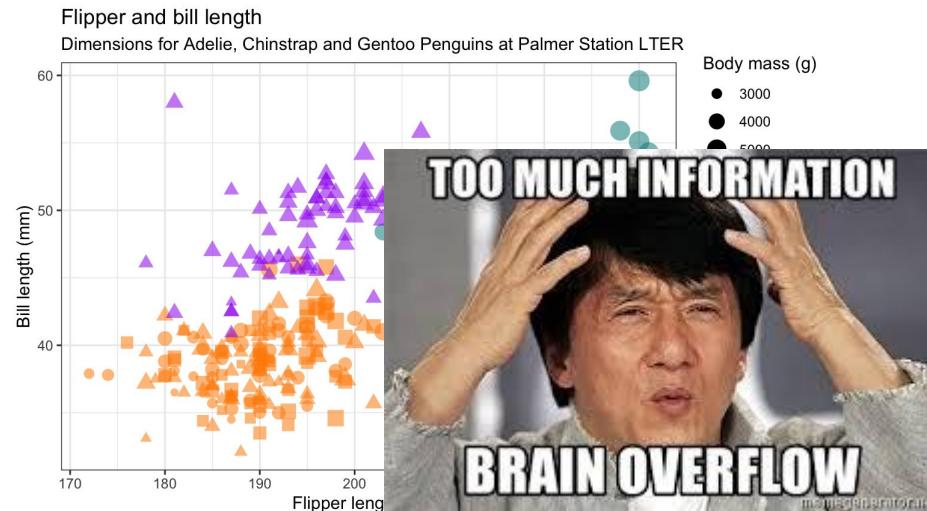
(apart from x and y):

- fill / color
- shape
- size

```
ggplot(data = penguins,aes(x = flipper_length_mm,  
                           y = bill_length_mm)) +  
  geom_point(aes(color = species,  
                 shape = island,  
                 size = body_mass_g,  
                 alpha = 0.8)) +
```

```
```{r}  
head(penguins)
```
```

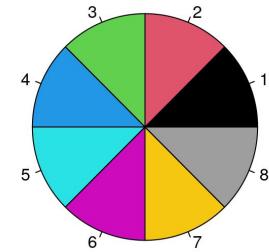
| species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | sex | year |
|---------|-----------|----------------|---------------|-------------------|-------------|--------|-------|
| <fctr> | <fctr> | <dbl> | <dbl> | <int> | <int> | <fctr> | <int> |
| Adelie | Torgersen | 39.1 | 18.7 | 181 | 3750 | male | 2007 |
| Adelie | Torgersen | 39.5 | 17.4 | 186 | 3800 | female | 2007 |
| Adelie | Torgersen | 40.3 | 18.0 | 195 | 3250 | female | 2007 |
| Adelie | Torgersen | NA | NA | NA | NA | NA | 2007 |
| Adelie | Torgersen | 36.7 | 19.3 | 193 | 3450 | female | 2007 |
| Adelie | Torgersen | 39.3 | 20.6 | 190 | 3650 | male | 2007 |



Color Usage

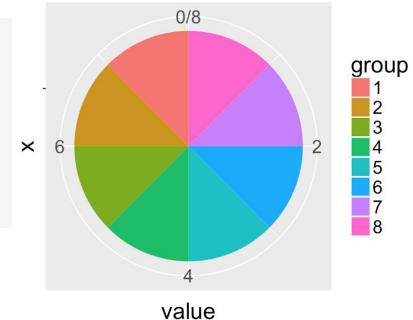
Default color scheme in base R:

```
pie(rep(1, 8), col=1:8)
```



Default color scheme in ggplot:

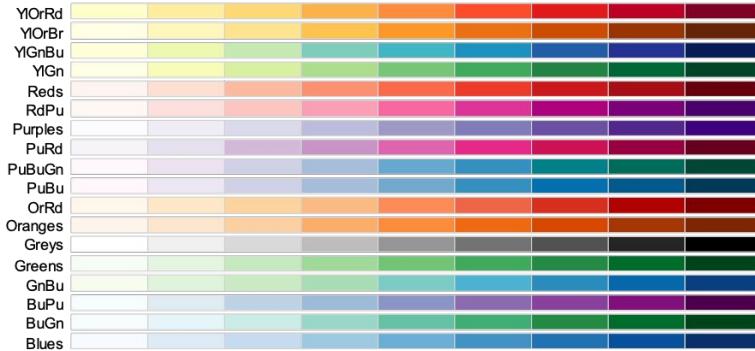
```
ggplot(data.frame(group = factor(seq_len(8)), value = rep(1, 8)),
       aes(x="", y=value, fill=group)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  theme(text = element_text(size = 20))
```



RColorBrewer

```
display.brewer.all()
```

sequential



qualitative



diverging



Viridis Palettes

```
install.packages("viridis")
library(viridis)
```

add

```
scale_color_viridis()
scale_fill_viridis()
```

to your plot



Viridis Palettes

Color scales are designed to be:

- **Colorful and Pretty**, spanning as wide a palette as possible so as to make differences easy to see,
- **Perceptually uniform**, the perceived difference between two colors is proportional to the Euclidian distance within the color space
- **Robust to colorblindness**, looks good in grey scale and to people with common forms of colorblindness

You can hear more about the science behind creating these color scales, on Walt and Smith's [talk at SciPy 2015](#).



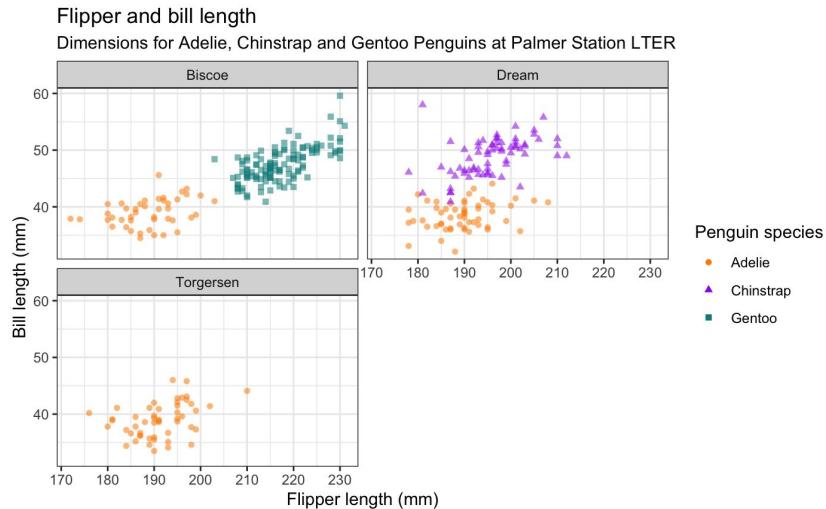
not perceptually uniform



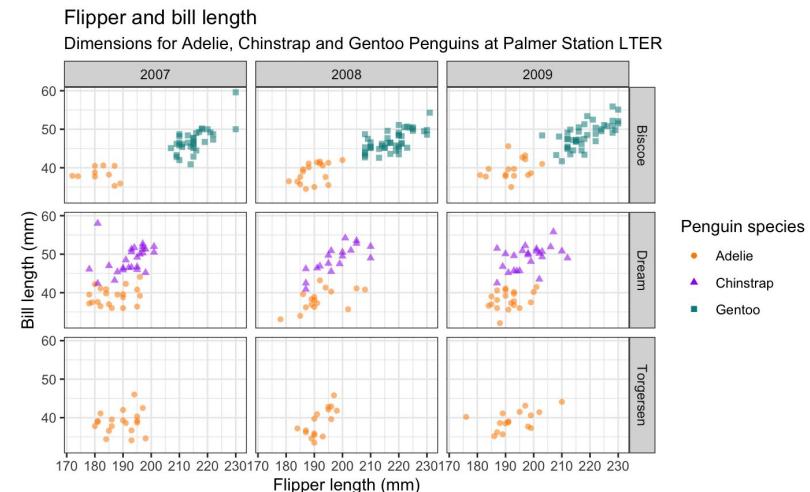
perceptually uniform

Faceting is useful to show more dimensions without overcrowding the graph

```
ggplot(data = penguins,aes(x = flipper_length_mm,  
                           y = bill_length_mm)) +  
  geom_point(aes(color = species,  
                  shape = species,  
                  alpha = 0.8)) +  
  facet_wrap(~island, ncol = 2) +
```



```
ggplot(data = penguins,aes(x = flipper_length_mm,  
                           y = bill_length_mm)) +  
  geom_point(aes(color = species,  
                  shape = species,  
                  alpha = 0.8)) +  
  facet_grid(island ~ year) +
```



Interactivity

- More data / dimensions than you can comfortably visualise
- Need for exploratory data analysis

Use shiny or plotly

<https://shiny.rstudio.com/gallery/genome-browser.html>

<https://shiny.posit.co/>

Animations (time-dependent plots):

<https://gganimate.com>

Linked Charts

<https://anders-biostat.github.io/linked-charts/>

plotly interactive graphics

- plotly is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).
- can **translate ‘ggplot2’ graphs to an interactive version.**
- You can open a **‘plotly’ account** to upload ‘plotly’ graphs and view or modify them in a web browser.
- Use RStudio to generate interactive plots as you go
- Save & distribute them as html

More on plotly:
<https://plotly-book.cpsievert.me>

Quick plotly code & demo

```
```{r}
library(ggplot2)
library(palmerpenguins)
library(plotly)
```

```{r}
scatterplot with more customization
fig <- plot_ly(data = penguins, x = ~flipper_length_mm, y = ~bill_length_mm, color = ~species, colors = penguin_colors,
 mode = "markers", symbol = ~species, symbols = c("circle","triangle-up","square"), size = 10)

fig
```



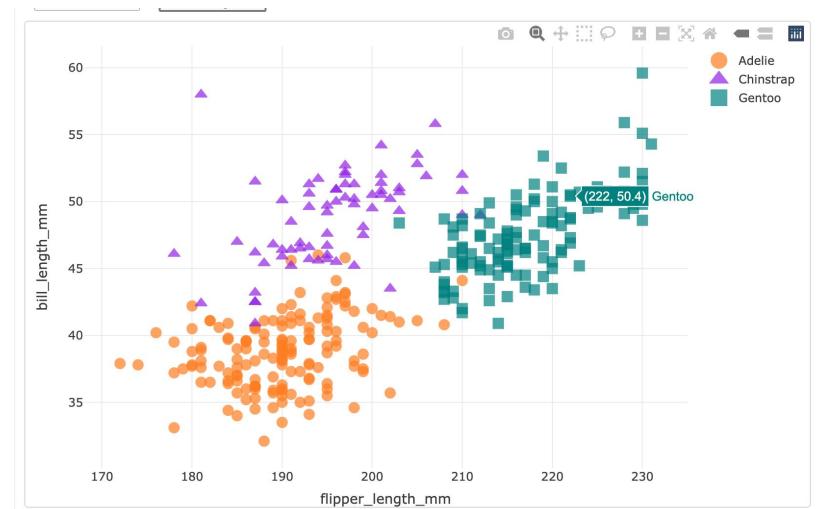
```

p <- ggplot(data = penguins,aes(x = flipper_length_mm,
 y = bill_length_mm)) +
 geom_point(aes(color = species,
 shape = species,
 alpha = 0.8)) +
 scale_color_manual(values = c("darkorange", "purple", "cyan4")) +
 scale_alpha(guide = 'none') +
 labs(title = "Flipper and bill length",
 subtitle = "Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER",
 x = "Flipper length (mm)",
 y = "Bill length (mm)",
 color = "Penguin species",
 shape = "Penguin species") +
 theme_bw()

ggplotly(p)

```


```



The ggplot universe

Pros & Cons:

- **High level of customization** with colors, layouts, scales and themes
- **Grammar of graphics** - similar code for all types of graphs
- **Learning curve**
- **Basis of interactive packages**

Using ChatGPT in data visualisation



You

can you take the R code below and adjust the colors to be from the viridis palette?

```
ggplot(data = penguins,aes(x = flipper_length_mm,  
                           y = bill_length_mm)) +  
  geom_point(aes(color = species,  
                 shape = species,  
                 alpha = 0.8)) +  
  scale_color_manual(values = c("darkorange","purple","cyan4")) +  
  scale_alpha(guide = 'none') +  
  labs(title = "Flipper and bill length",  
       subtitle = "Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER",  
       x = "Flipper length (mm)",  
       y = "Bill length (mm)",  
       color = "Penguin species",  
       shape = "Penguin species") +  
  theme_bw()
```



Using ChatGPT in data visualisation



ChatGPT

Certainly! Here's the adjusted code using the viridis color palette:

```
R
library(ggplot2)
library(palmerpenguins)
library(viridis)

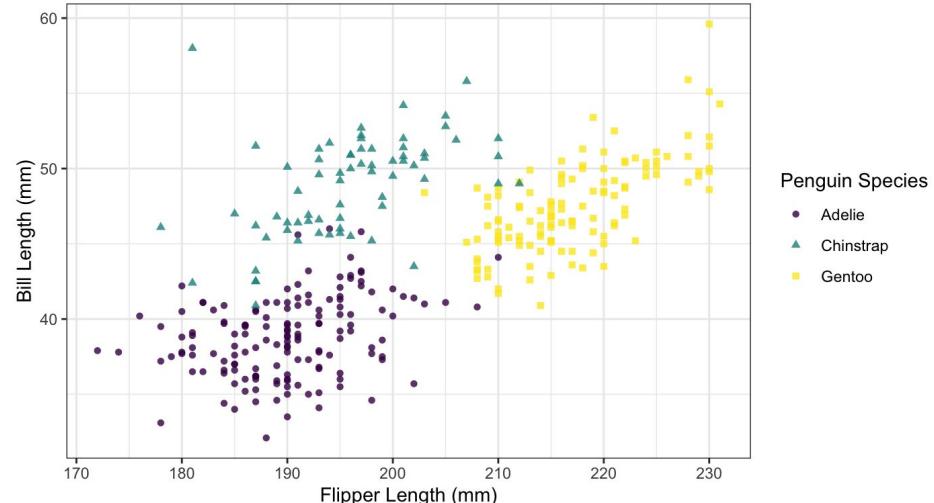
# Load Palmer Penguin dataset
data("penguins")

# Create plot with viridis color palette
ggplot(data = penguins, aes(x = flipper_length_mm, y = bill_length_mm)) +
  geom_point(aes(color = species, shape = species), alpha = 0.8) +
  scale_color_viridis(discrete = TRUE) + # Using viridis color palette
  scale_alpha(guide = 'none') +
  labs(title = "Flipper and Bill Length",
       subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins at Palmer Station LTER",
       x = "Flipper Length (mm)",
       y = "Bill Length (mm)",
       color = "Penguin Species",
       shape = "Penguin Species") +
  theme_bw()
```

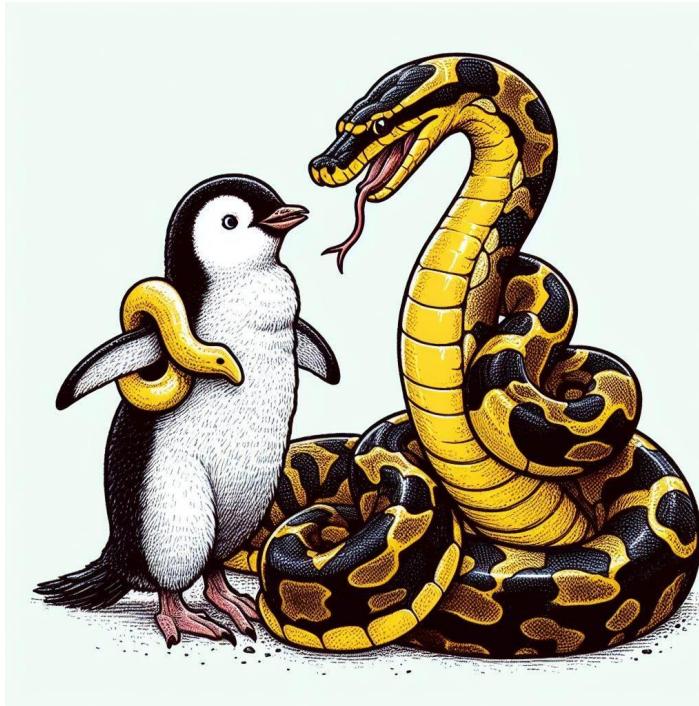
This code will create a scatter plot of flipper length vs. bill length for the penguins dataset, using the viridis color palette for the species categories.

Flipper and Bill Length

Dimensions for Adelie, Chinstrap, and Gentoo Penguins at Palmer Station LTER



Data visualisation in Python



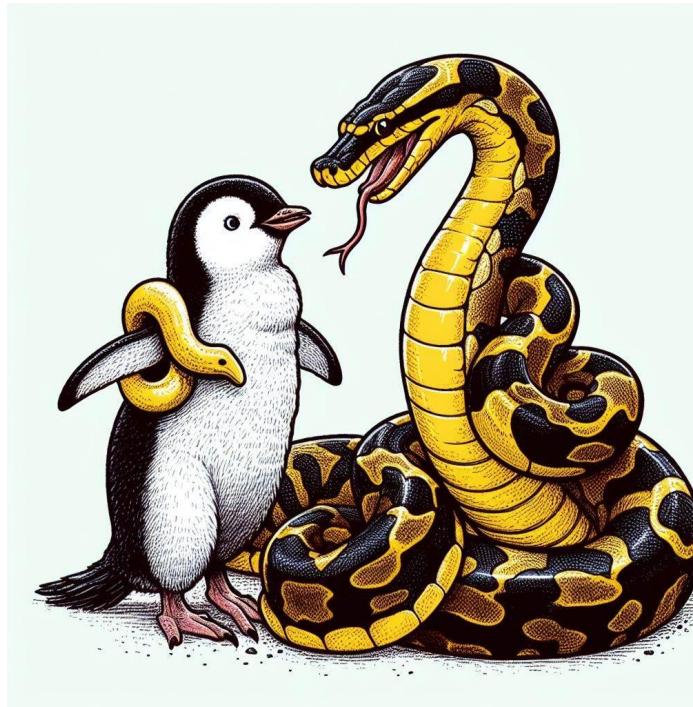
Data visualisation in Python

matplotlib

seaborn



plotly



Matplotlib - the basic one

```
fig, ax = plt.subplots()

colors = {'Adelie': 'darkorange', 'Gentoo': 'darkcyan', 'Chinstrap': 'purple'}

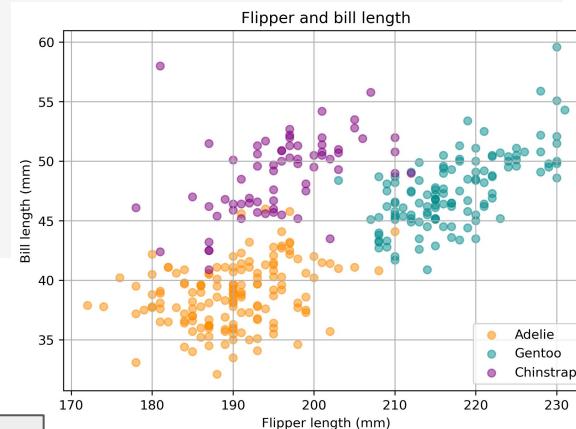
# Scatter plot for each species
for species, color in colors.items():
    species_data = penguins[penguins['species'] == species]
    ax.scatter(species_data["flipper_length_mm"], species_data["bill_length_mm"], c=color, alpha=0.5, label=species)

ax.set_xlabel('Flipper length (mm)', fontsize=10)
ax.set_ylabel('Bill length (mm)', fontsize=10)
ax.set_title('Flipper and bill length')

# Add Legend
ax.legend()

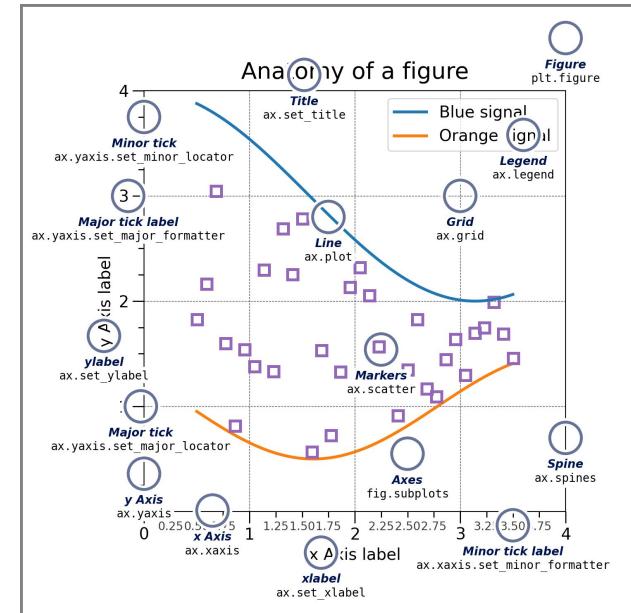
ax.grid(True)
fig.tight_layout()

plt.show()
```



Matplotlib cheatsheets:

<https://matplotlib.org/cheatsheets>



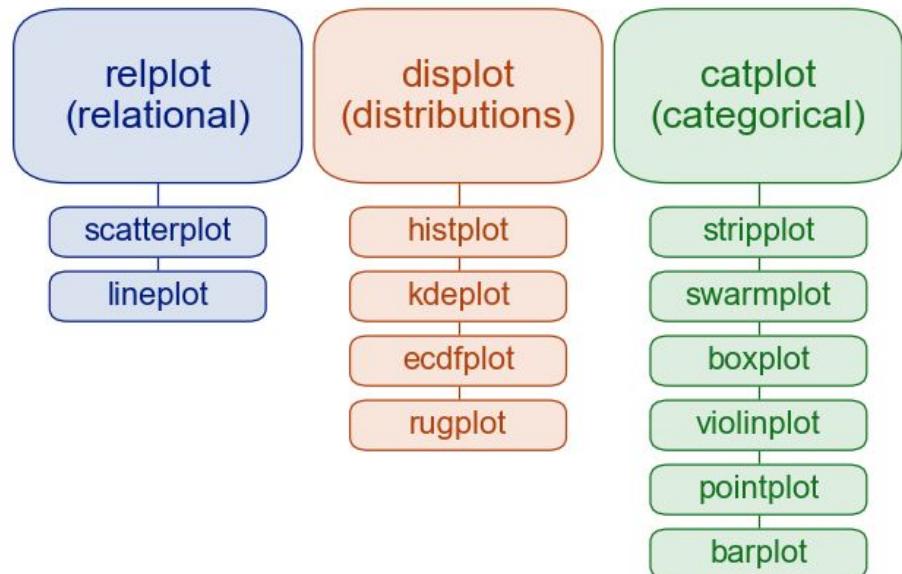
Matplotlib - the basic one

Pros & Cons:

- Can create very **detailed visualizations**
- Routine tasks can require lots of **boilerplate code**
- **No concept of facets / lattices**
- Only a **single global coordinate** system allowed per plot
- Other packages that build/extend Matplotlib:
<https://matplotlib.org/mpl-third-party/>

Seaborn - the industry standard

- Based on Matplotlib
- Figure-level functions
 - Relational
 - Distributional
 - Categorical
- Axes-level functions
 - Scatterplot
 - Boxplot
 - ...

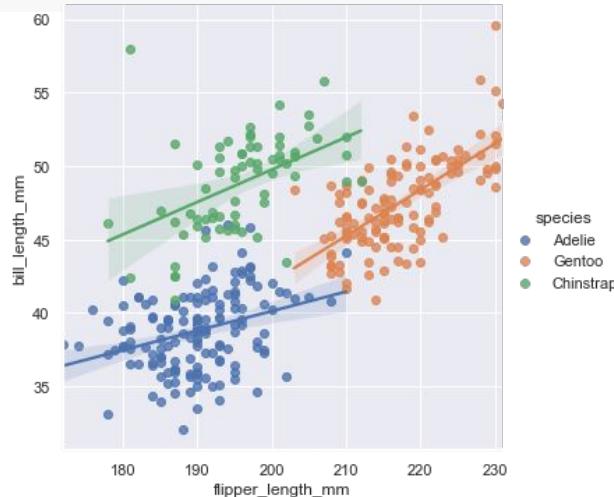
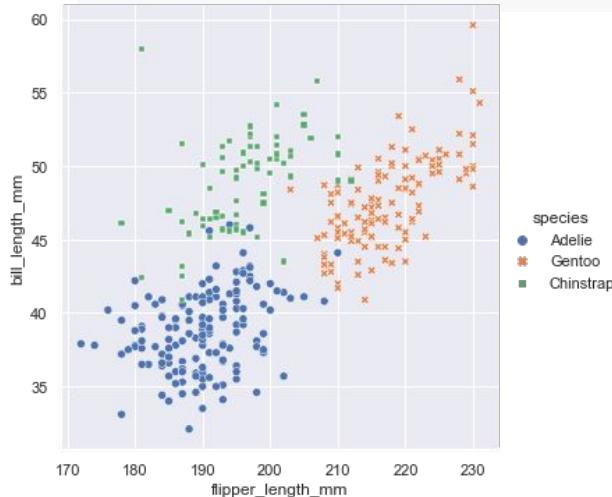


[https://seaborn.pydata.org/tutorial/
function_overview.html](https://seaborn.pydata.org/tutorial/function_overview.html)

Seaborn - figure-level functions

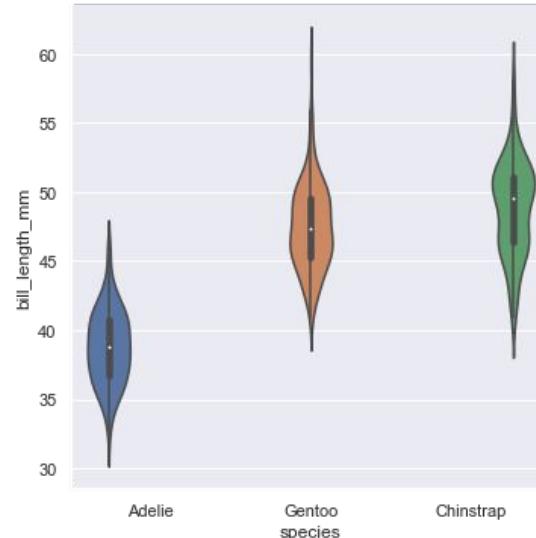
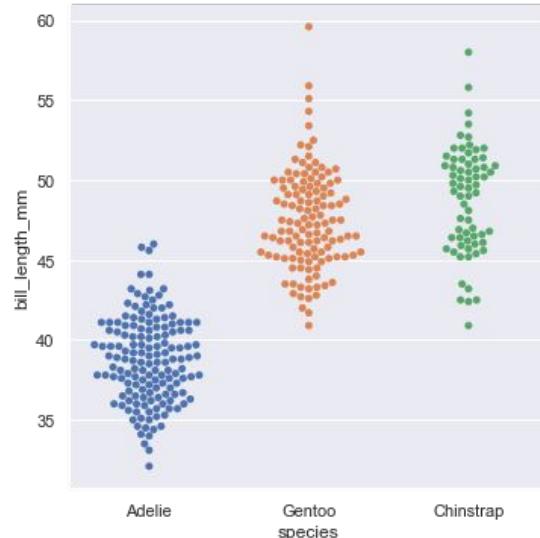
```
# scatterplot
sns.relplot(
    data=penguins,
    x="flipper_length_mm", y="bill_length_mm", hue="species",
    style="species",
)

# scatterplot w/ LM
sns.lmplot(
    data=penguins, x="flipper_length_mm", y="bill_length_mm", hue="species",
)
```



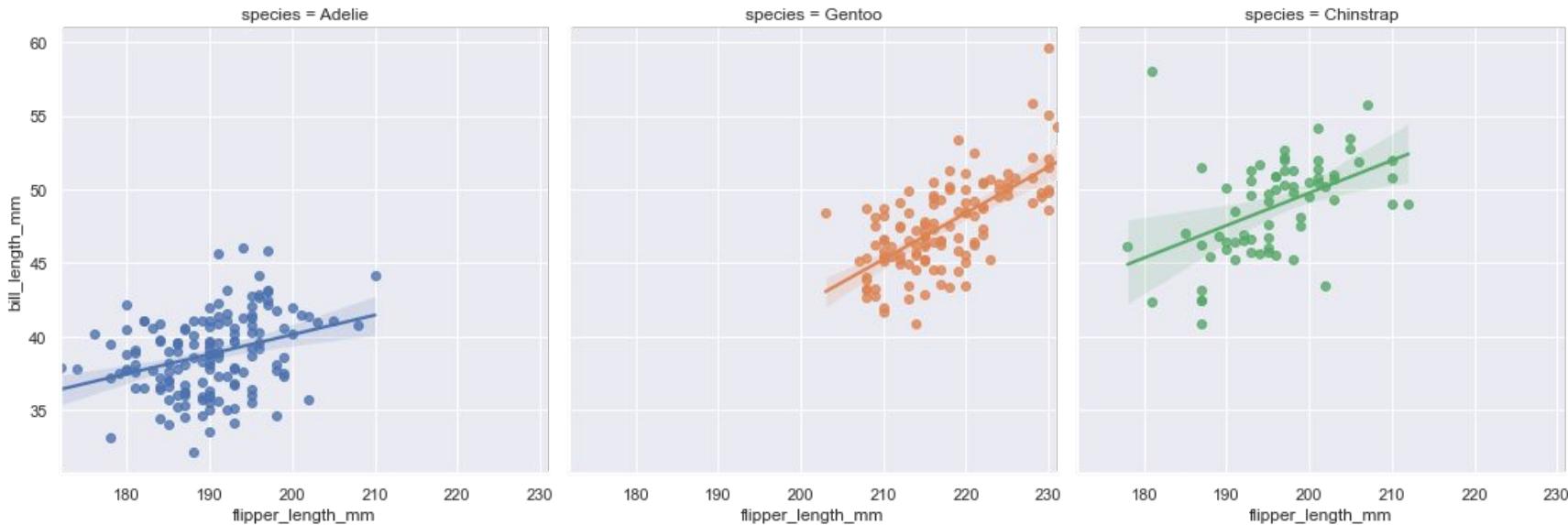
Seaborn - figure-level functions

```
sns.catplot(  
    data=penguins, kind="swarm", x="species", y="bill_length_mm", hue="species"  
)  
  
sns.catplot(  
    data=penguins, kind="violin", x="species", y="bill_length_mm", hue="species"  
)
```



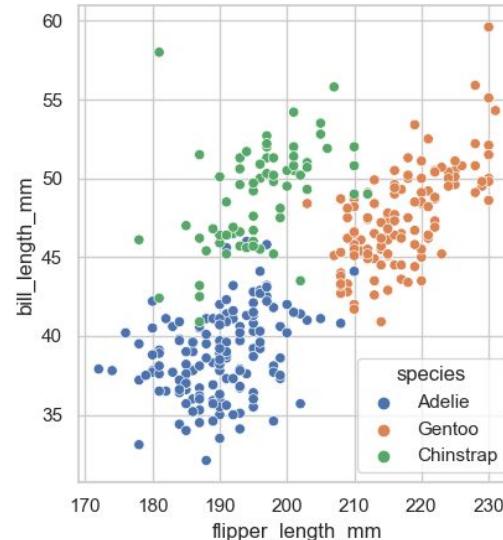
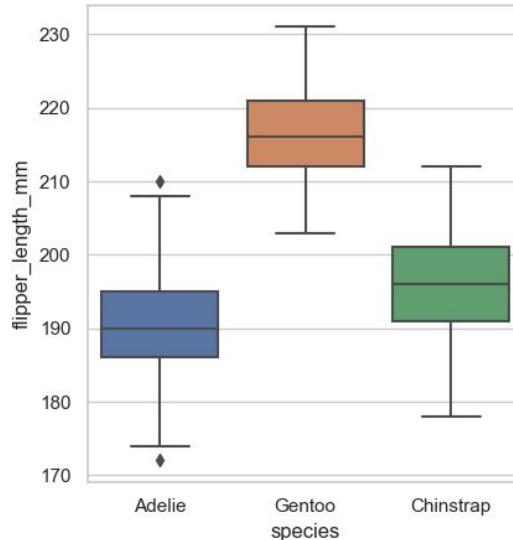
Seaborn - faceting

```
# facetting
sns.lmplot(
    data=penguins, x="flipper_length_mm", y="bill_length_mm", hue="species", col="species",
)
```



Seaborn - axes-level functions

```
fig, axes = plt.subplots(1, 2, sharey=False, figsize=(10, 5))
sns.boxplot(x="species", y="flipper_length_mm", data=penguins, ax=axes[0])
sns.scatterplot(x="flipper_length_mm", y="bill_length_mm", hue="species", data=penguins, ax=axes[1])
plt.show()
```



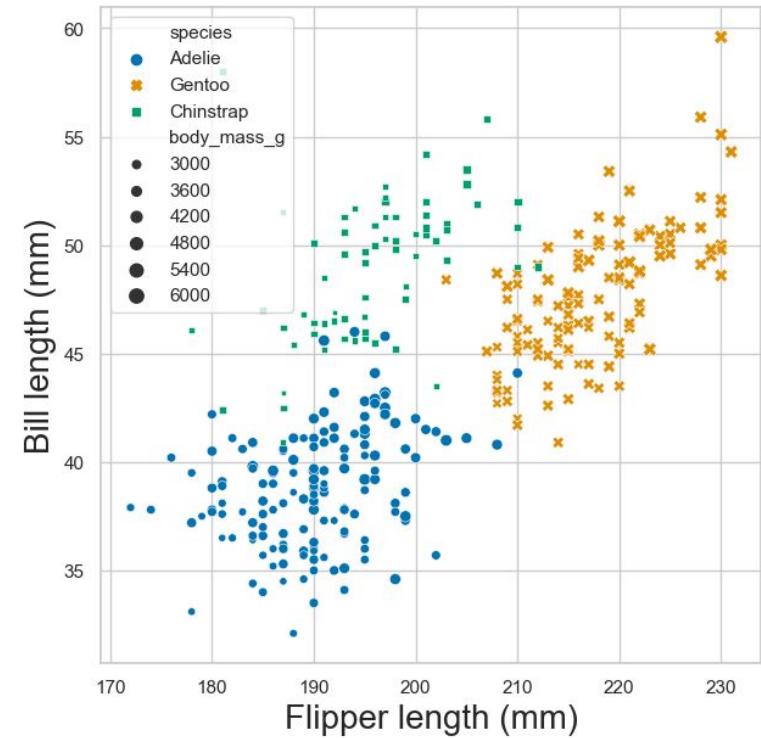
Seaborn - customization

```
# More customized scatterplot
import matplotlib.pyplot as plt

sns.set_style("whitegrid")

plt.figure(figsize=(7,7))

# Add body mass as 3rd dimension
sc = sns.scatterplot(
    data=penguins,
    x="flipper_length_mm", y="bill_length_mm", hue="species",
    style="species", palette="colorblind", size="body_mass_g"
)
sc.set_xlabel("Flipper length (mm)", fontsize=20)
sc.set_ylabel("Bill length (mm)", fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



Seaborn - the industry standard

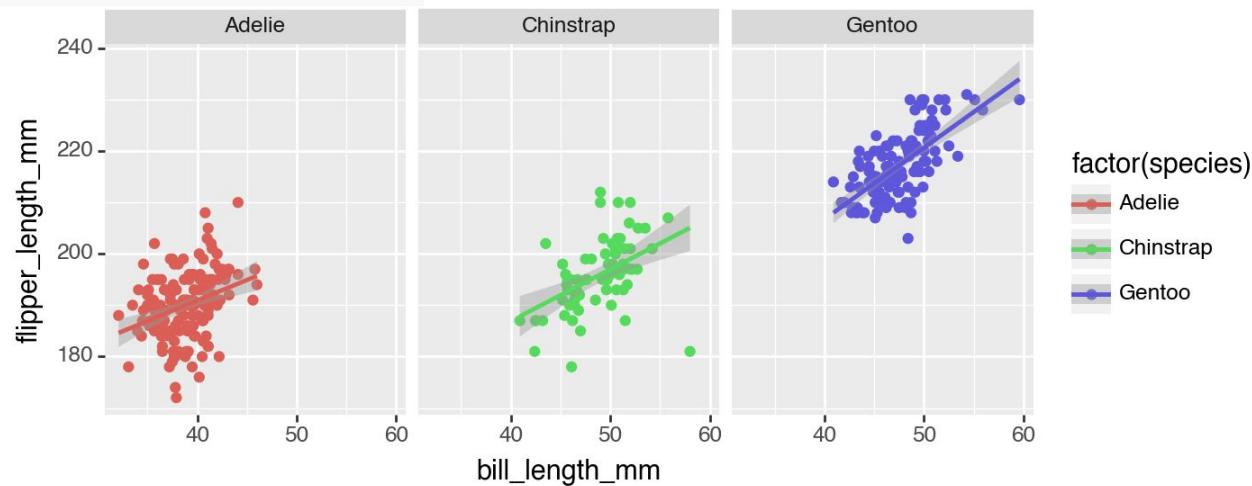
Pros & Cons:

- **Different functions for different plot types**, with different interfaces
- Quickly generate nice visualizations for simpler tasks
- Routine tasks can require lots of **boilerplate code**
- **Limited flexibility** and might need to be customized with Matplotlib
- Mostly for **two-dimensional** data analysis

Plotnine - Grammar of Graphics for Python

```
from plotnine import ggplot, geom_point, aes, stat_smooth, facet_wrap, scale_color_manual, labs, theme
from palmerpenguins import load_penguins
penguins = load_penguins()
```

```
( ggplot(penguins, aes("bill_length_mm", "flipper_length_mm", color="factor(species)"))
+ geom_point()
+ stat_smooth(method="lm")
+ facet_wrap(~species)
+ theme(figure_size=(7, 3))
)
```

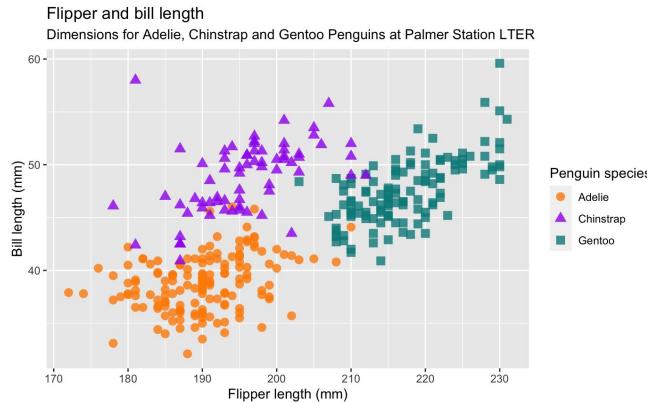


ggplot vs. plotnine code

```
```{r}
library(palmerpenguins)
library(ggplot2)
```

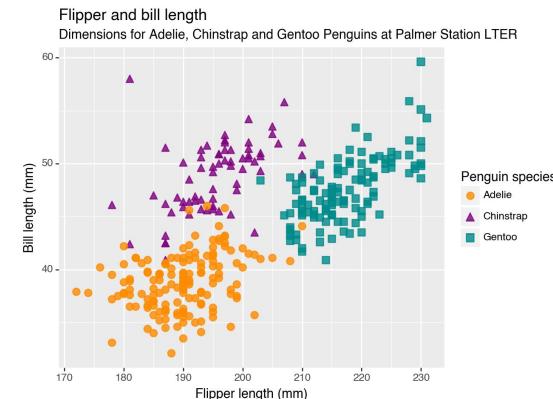
```{r}
ggplot(data = penguins, aes(x = flipper_length_mm, y = bill_length_mm)) +
 geom_point(aes(color = species, shape = species), size = 3, alpha = 0.8) +
 scale_color_manual(values = c("darkorange", "purple", "cyan4")) +
 labs(title = "Flipper and bill length",
 subtitle = "Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER",
 x = "Flipper length (mm)",
 y = "Bill length (mm)",
 color = "Penguin species",
 shape = "Penguin species")
```
```

```



```
from plotnine import ggplot, geom_point, aes, stat_smooth, facet_wrap, scale_color_manual, labs, theme
from palmerpenguins import load_penguins
penguins = load_penguins()

(
 ggplot(penguins, aes(x = "flipper_length_mm", y = "bill_length_mm")) +
 geom_point(aes(color = "species", shape = "species"), size = 3, alpha = 0.8) +
 scale_color_manual(values = ("darkorange", "purple", "darkcyan")) +
 labs(title = "Flipper and bill length",
 subtitle = "Dimensions for Adelie, Chinstrap and Gentoo Penguins at Palmer Station LTER",
 x = "Flipper length (mm)",
 y = "Bill length (mm)",
 color = "Penguin species",
 shape = "Penguin species")
)
```



# Colors in python

- Hex colors
- Named colors
- Colormaps
- Palettable
- ColorBrewer for Python

Matplotlib named colors:

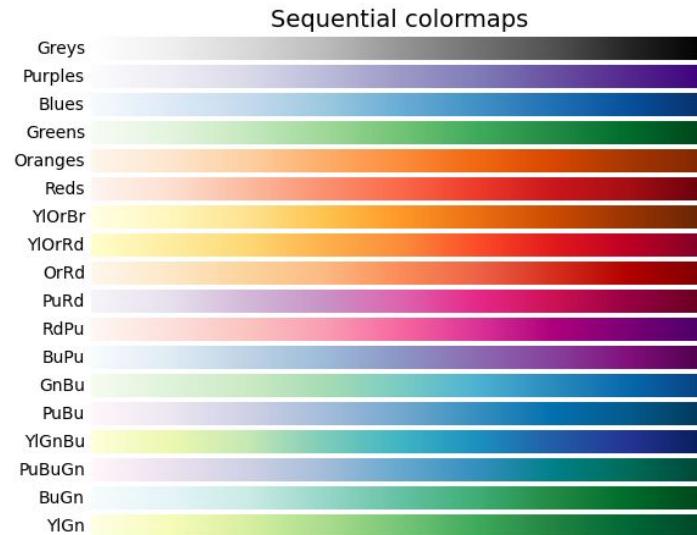
[https://matplotlib.org/stable/gallery/color/named\\_colors.html#sphx-glr-gallery-color-named-colors-py](https://matplotlib.org/stable/gallery/color/named_colors.html#sphx-glr-gallery-color-named-colors-py)

## CSS Colors

```
plot_colortable(mcolors.CSS4_COLORS)
plt.show()
```

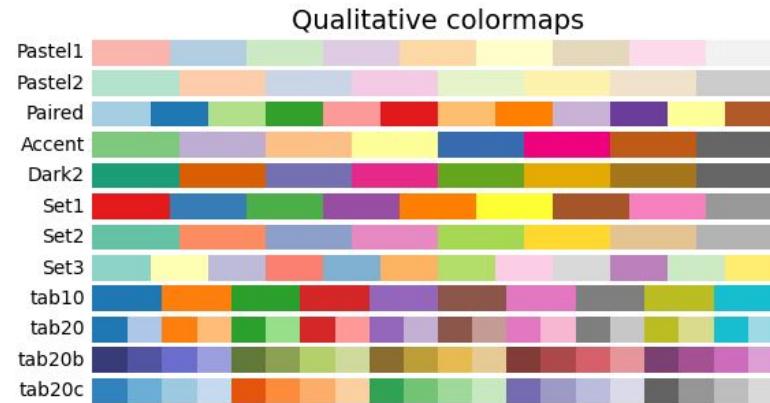


# Matplotlib - Color maps



Matplotlib color maps:

<https://matplotlib.org/stable/users/explain/colors/colormaps.html>



# Plotly for Python

plotly | Graphing Libraries

Search...

Quick Reference

- Getting Started
- Is Plotly Free?
- Figure Reference
- API Reference
- Dash
- GitHub
- community.plotly.com

Examples

- Fundamentals
- Basic Charts
- Scatter Plots
- Line Charts
- Bar Charts
- Heatmaps
- Box Plots
- Violin Plots
- Area Charts
- Waterfall Charts
- Choropleth Maps

<https://dash.plotly.com/tutorial>

Python (v5.18.0) ▾ Forum

## Plotly Open Source Graphing Library for Python

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

Plotly.py is [free and open source](#) and you can [view the source](#), [report issues](#) or [contribute on GitHub](#).

Deploy Python AI Dash apps on private Kubernetes clusters: [Pricing](#) | [Demo](#) | [Overview](#) | [AI App Services](#)

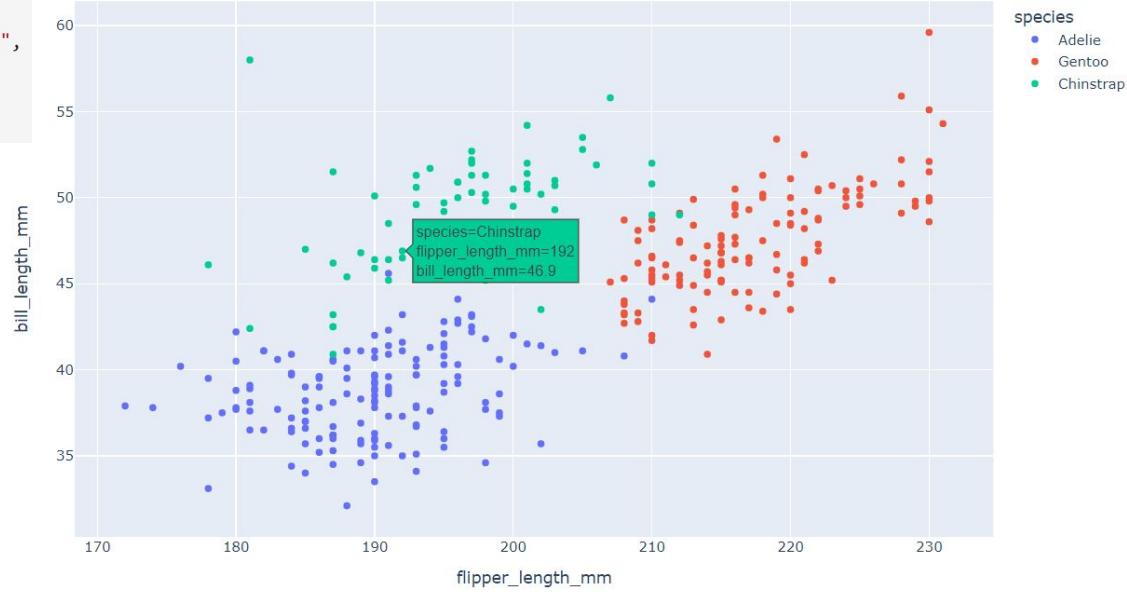
### Fundamentals

[More Fundamentals »](#)



# Plotly.express for quick plots

```
Using plotly.express
fig = px.scatter(penguins, x="flipper_length_mm", y="bill_length_mm",
 color="species")
fig.update_layout(width=1000, height=600)
fig.show()
```



Tutorial:

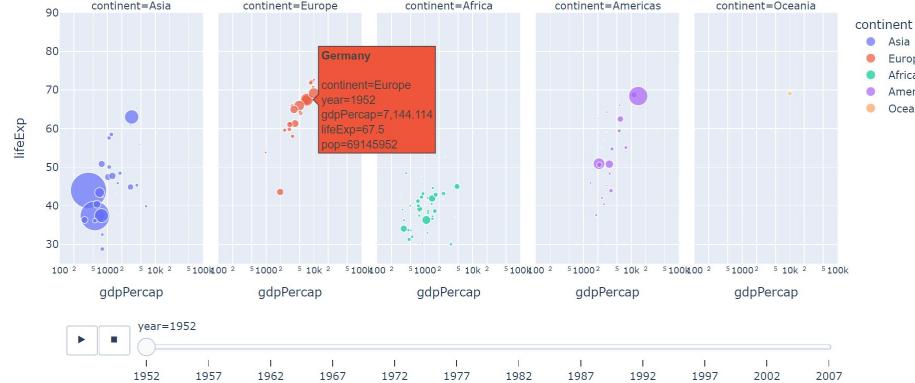
<https://towardsdatascience.com/visualization-with-plotly-express-comprehensive-guide-eb5ee4b50b57>

# Plotly.express for quick plots

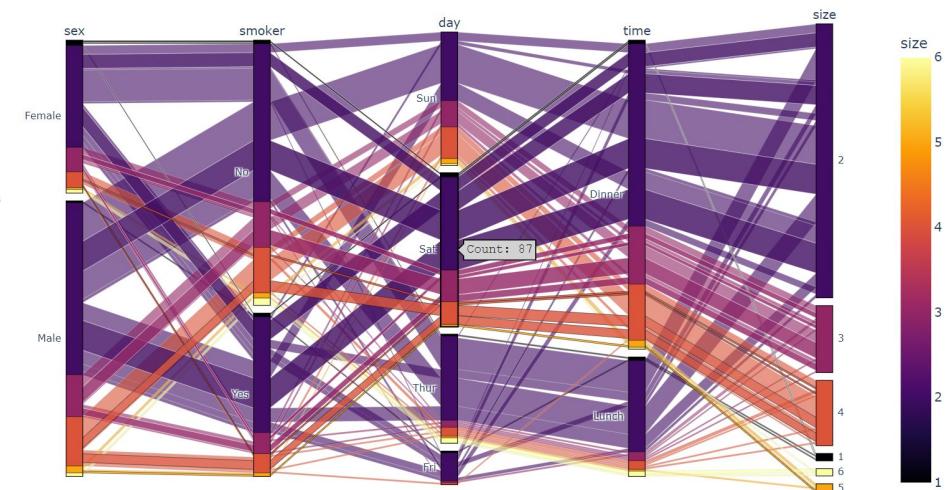
```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter(df, x="gdpPercap", y="lifeExp", animation_frame="year",
 animation_group="country", size="pop", color="continent",
 hover_name="country", facet_col="continent",
 log_x=True, size_max=45, range_x=[100,100000], range_y=[25,90])

fig.update_layout(width=1100, height=500, uniformtext_minsize=2, uniformtext_mode='hide')
fig.update_xaxes(tickfont=dict(size=10))

fig.show()
```



```
import plotly.express as px
df = px.data.tips()
fig = px.parallel_categories(df, color="size",
 color_continuous_scale=px.colors.sequential.Inferno)
fig.update_layout(width=1000, height=600)
fig.show()
```



# Plotly.graph\_objects

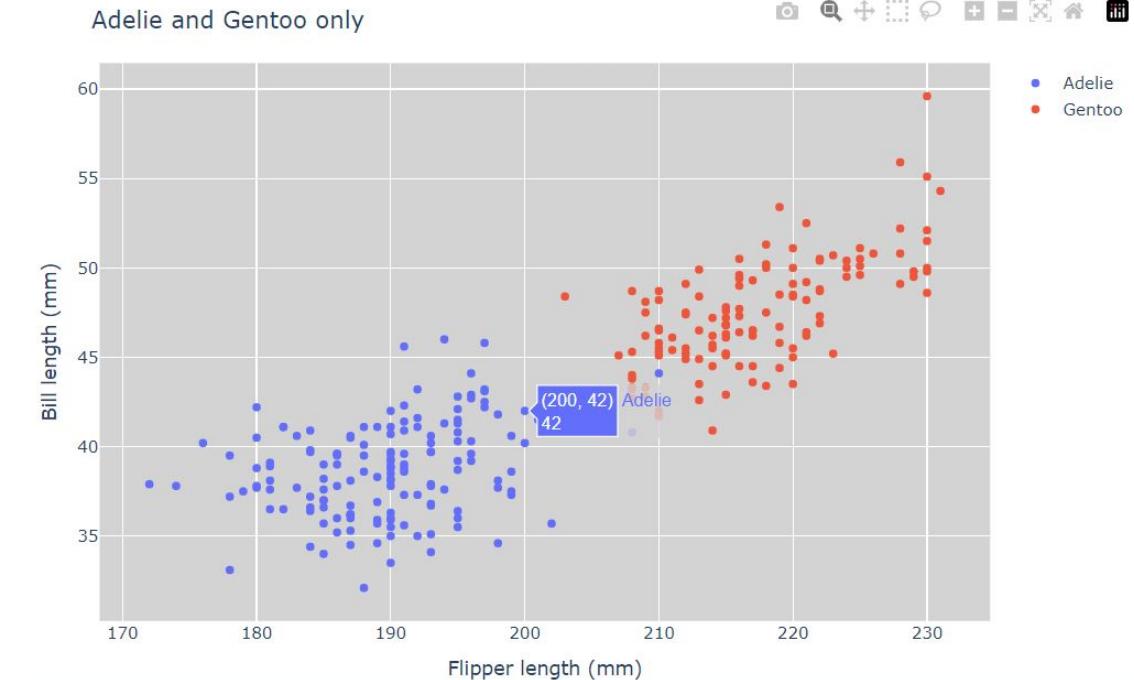
```
Using plotly.graph_objects

Create the figure
fig=go.Figure()

Add plot for Adelie species
fig.add_trace(go.Scatter(
 x=penguins[penguins.species == "Adelie"].flipper_length_mm,
 y=penguins[penguins.species == "Adelie"].bill_length_mm,
 text=penguins[penguins.species == "Adelie"].bill_length_mm,
 name = "Adelie",
 mode="markers",
 textposition='bottom right'
))

Add plot for Gentoo species
fig.add_trace(go.Scatter(
 x=penguins[penguins.species == "Gentoo"].flipper_length_mm,
 y=penguins[penguins.species == "Gentoo"].bill_length_mm,
 text=penguins[penguins.species == "Gentoo"].bill_length_mm,
 mode="markers",
 name = "Gentoo"
))

Customize
fig.update_layout(margin=dict(l=15, r=5, t=50, b=5),
 width=800, height=500, showlegend=True,
 plot_bgcolor='lightgray',
 xaxis_title = "Flipper length (mm)",
 yaxis_title = "Bill length (mm)",
 title = "Adelie and Gentoo only",
 xaxis=dict(showticklabels=True,showgrid=True),
 yaxis=dict(showticklabels=True,showgrid=True)
)
fig.show()
```



# Dash

## Equivalent to Shiny, to make apps/dashboards

plotly | Dash

### Clinical Analytics

#### Welcome to the Clinical Analytics Dashboard

Explore clinic patient volume by time of day, waiting time, and care score. Click on the heatmap to visualize patient experience at different time points.

##### Select Clinic

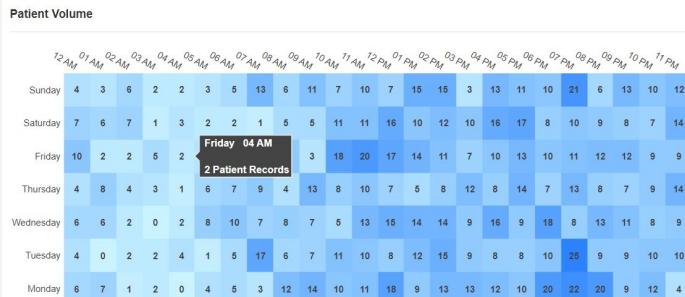
Madison Center

##### Select Check-In Time

01/01/2014 → 01/15/2014

##### Select Admit Source

- Emergency Room
- Physician Referral
- Not Identified
- Outside Health Care Facility
- Admitted as transfer from another unit
- Court/Law Enforcement
- Outside Hospital
- Transfer from Critical Access Hospital



#### Patient Wait Time and Satisfactory Scores

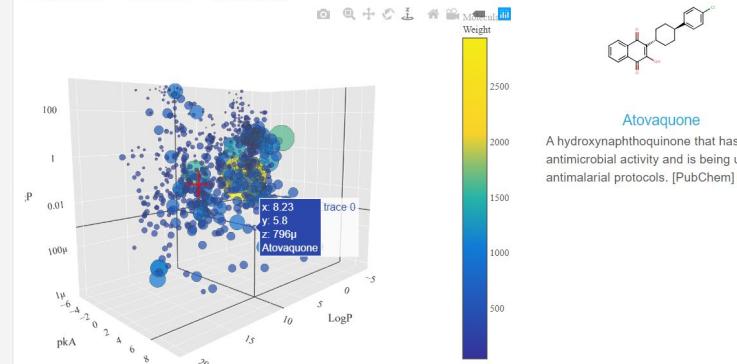
Department	Wait Time Minutes	Care Score
Emergency	100+ minutes	5+

#### DASH FOR DRUG DISCOVERY

HOVER over a drug in the graph to see its structure.  
SELECT a drug in the dropdown to add it to the drug candidates at the bottom.

Levobupivacaine

● 3D Scatter ○ 2D Scatter ○ 2D Histogram



Dash Bio: <https://dash.plotly.com/dash-bio>

# Summary

- **Visualization and customization of plots is an important aspect of scientific storytelling**
- R has ggplot2, which has extensive visualization capability and applies the layered grammar of graphics approach
- Python has tools like Plotly, Matplotlib, Seaborn, and Plotnine
- Shiny or Plotly for interactive visualizations
- There is a wide range of libraries that can be used...

# Resources

R BasicsBasics -

<https://rladiessydney.org/courses/ryouwithme/01-basicbasics-0/>

Getting started w/ Markdown -

<https://posit.co/resources/videos/getting-started-with-r-markdown/>

Modern Statistics for Modern Biology -

<https://www.huber.embl.de/msmb/> - Chapter 03

TidyTuesday -

<https://github.com/rfordatascience/tidytuesday>

# Acknowledgements

- Wolfgang Huber & Susan Holmes
- Florian Heyl
- Julien Gagneur

**Go forth & visualise!**