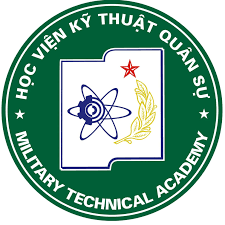
# khung doi

**Hà Nội – 05/2024**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO MÔN HỌC TRỰC QUAN HÓA DỮ LIỆU**

**Đề tài:** **Tìm hiểu về công cụ Bokeh**

|  |  |
| --- | --- |
| **Học viên thực hiện :** | **Lê Mạnh Hà – Lớp PTDL** |
| **Giáo viên hướng dẫn:** | **2// Vi Bảo Ngọc** |

# 

# MỤC LỤC

MỤC LỤC ............................................................................................................. 2

[CHƯƠNG 1:TỔNG QUAN VỀ CÔNG CỤ BOKEH 5](#_Toc166195274)

[1.1 Giới thiệu 5](#_Toc166195275)

[1.2 Ưu điểm 5](#_Toc166195276)

[1.3 Cài đặt 6](#_Toc166195277)

[CHƯƠNG 2: TRỰC QUAN HÓA SỐ LƯỢNG 7](#_Toc166195278)

[2.1 Bar plot 7](#_Toc166195279)

[2.2 Grouped and stacked bar 8](#_Toc166195280)

[A. Grouped Bars 8](#_Toc166195281)

[B. Grid of bar plots 9](#_Toc166195282)

[C. Stacked bars 11](#_Toc166195283)

[2.3 Dot plots and heatmaps 13](#_Toc166195284)

[A. Dot plots 13](#_Toc166195285)

[B. Heatmap 15](#_Toc166195286)

[CHƯƠNG 3: Trực quan hóa sự phân bố: Biểu đồ và biểu đồ mật độ 19](#_Toc166195287)

[3.1 Trực quan hóa một phân phối duy nhất 19](#_Toc166195288)

[A. Histograms 19](#_Toc166195289)

[B. Density plots 21](#_Toc166195290)

[3.2 Trực quan hóa nhiều bản phân phối cùng một lúc 23](#_Toc166195291)

[A. Multiple distribution density plots 23](#_Toc166195292)

[B. Multiple distribution density plots lồng nhau 26](#_Toc166195293)

[C. Multiple distribution histogram 28](#_Toc166195294)

[CHƯƠNG 4:  TRỰC QUAN HÓA PHÂN PHỐI: HÀM PHÂN PHỐI TÍCH LŨY VÀ BIỂU ĐỒ QQ 31](#_Toc166195295)

[4.1 Hàm phân phối tích lũy 31](#_Toc166195296)

[4.2 Phân phối Highly skewed 32](#_Toc166195297)

[A. Densify và Cumulative frequency 32](#_Toc166195298)

[B. Relative frequency 35](#_Toc166195299)

[4.3 Biểu đồ quantile–quantile 37](#_Toc166195300)

[A. Phân bố chuẩn 37](#_Toc166195301)

[B. Phân bố log-normally 39](#_Toc166195302)

[CHƯƠNG 5: Trực quan hóa nhiều bản phân phối cùng một lúc 41](#_Toc166195303)

[5.1 Trực quan hóa sự phân bố dọc theo trục tung 41](#_Toc166195304)

[A. Boxplot 41](#_Toc166195305)

[B. Sina plot 43](#_Toc166195306)

[5.2 Trực quan hóa phân bố dọc theo trục ngang 44](#_Toc166195307)

[A. Ridgeline plot 44](#_Toc166195308)

[CHƯƠNG 6: TRỰC QUAN HÓA TỈ LỆ 48](#_Toc166195309)

[6.1. Trường hợp biểu đồ tròn 48](#_Toc166195310)

[A. Pie chart 48](#_Toc166195311)

[6.2. Trường hợp các thanh cạnh nhau. 49](#_Toc166195312)

[A. side-by-side bars 49](#_Toc166195313)

[10.3 Trường hợp thanh xếp chồng và khối lượng riêng xếp chồng 50](#_Toc166195316)

[A. Stacked bars 50](#_Toc166195317)

[B. Stacked densities 52](#_Toc166195318)

[6.4 Trực quan hóa các tỷ lệ riêng biệt như một phần của tổng thể 52](#_Toc166195320)

[A. Multiple distribution density plots 52](#_Toc166195321)

[CHƯƠNG 7: TRỰC QUAN HÓA CÁC TỶ LỆ LỒNG NHAU 53](#_Toc166195322)

[7.2 Các ô khảm và sơ đồ cây 53](#_Toc166195323)

[A. Mosaic plots 53](#_Toc166195324)

[B. Treemap 54](#_Toc166195325)

[7.3 Nested pies 56](#_Toc166195326)

[7.4 Parallel sets 57](#_Toc166195327)

[CHƯƠNG 8:  TRỰC QUAN HÓA MỐI LIÊN HỆ GIỮA HAI HOẶC NHIỀU BIẾN ĐỊNH LƯỢNG 59](#_Toc166195328)

[8.1 Scatter plots 59](#_Toc166195329)

[A. Bubble chart 59](#_Toc166195330)

[B. All-against-all scatter plot matrix 60](#_Toc166195331)

[8.2 Correlograms 63](#_Toc166195332)

[8.3 Dimension reduction 65](#_Toc166195333)

[8.4 Paired data 66](#_Toc166195334)

[CHƯƠNG 9: TRỰC QUAN HÓA CHUỔI THỜI GIAN VÀ CÁC HÀM KHÁC CỦA MỘT BIẾN 69](#_Toc166195335)

[9.1 Chuỗi thời gian riêng lẻ 69](#_Toc166195336)

[A. Dot graph 69](#_Toc166195337)

[B. Line graph 70](#_Toc166195338)

[9.2 Multiple time series and dose–response curves 71](#_Toc166195341)

[9.3 Time series of two or more response variables 72](#_Toc166195342)

[CHƯƠNG 10: HÌNH DUNG TREND 74](#_Toc166195343)

[10.1 Smoothing 74](#_Toc166195344)

[10.2 Showing trends with a defined functional form 75](#_Toc166195345)

[10.3 Detrending and time-series decomposition 77](#_Toc166195346)

[A. Detrending 77](#_Toc166195347)

[B. Decompose the Keeling curve 78](#_Toc166195348)

[CHƯƠNG 11: TRỰC QUAN HÓA DỮ LIỆU KHÔNG GIAN ĐỊA LÝ 80](#_Toc166195349)

[11.1 World Map 80](#_Toc166195350)

[11.2 Layers 81](#_Toc166195351)

[11.3 Choropleth mapping 82](#_Toc166195352)

[CHƯƠNG 12: HÌNH DUNG SỰ KHÔNG CHẮC CHẮN 83](#_Toc166195353)

## 12.2 Visualizing the uncertainty of point estimates........................................ 84

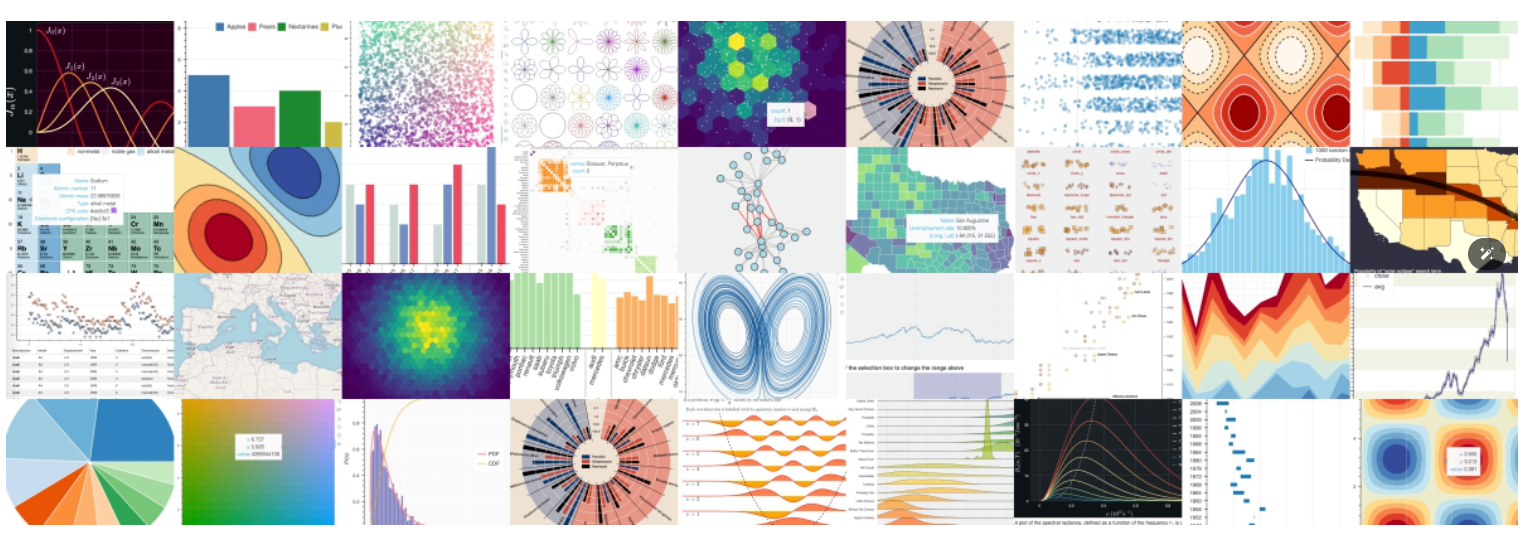
KẾT LUẬN ......................................................................................................... 86

# CHƯƠNG 1:TỔNG QUAN VỀ CÔNG CỤ BOKEH

## Giới thiệu

Bokeh là một thư viện trực quan hóa mạnh mẽ tương thích với các trình duyệt web hiện đại, mang lại một kết quả tinh tế. Nó rất hữu ích cho các bộ dữ liệu lớn hoặc dữ liệu đang truyền thời gian thực bằng cách cung cấp hiệu suất cao và tận dụng tài nguyên một cách tối ưu để trực quan hóa các đồ thị tương tác và đa dạng.

Với Bokeh, bạn có thể sử dụng Python để xây dựng các hình ảnh trực quan hóa dữ liệu đẹp mắt, từ các biểu đồ đơn giản đến các trang tổng quan phức tạp với các tập dữ liệu phát trực tuyến.



## Ưu điểm

Dưới đây là những điểm mạnh chính của thư viện Bokeh giúp phân biệt nó với các thư viện trực quan hóa dữ liệu khác:

* **Trực quan hóa tương tác**

Bokeh nổi bật với các tính năng trực quan tương tác phong phú. Người dùng có thể thực hiện các tương tác như thu phóng, xoay, chọn và tìm kiếm trên đồ họa. Điều này làm cho việc phân tích dữ liệu khám phá có tính tương tác cao hơn.

* **Hỗ trợ web**

Một ưu điểm thiết kế khác của Bokeh là nó hoạt động hài hòa với các ứng dụng và hình ảnh trực quan trên web. Bo mạch tương thích với HTML, CSS và JavaScript. Điều này mang lại cho người dùng lợi thế đáng kể trong việc tích hợp trực quan hóa dữ liệu vào các dự án dựa trên web.

* **Hỗ trợ đa nền tảng**

Bokeh có thể được sử dụng trên các nền tảng như: Jupyter Notebook, JupyterLab, Flask và Django cũng như các ứng dụng được phát triển bằng Python. Tính linh hoạt này cho phép sử dụng Bo mạch trong các môi trường và dự án khác nhau, biến nó thành một thư viện thân thiện với người dùng.

* **Vẽ linh hoạt**

Bokeh cung cấp các tùy chọn để sử dụng nhiều tính năng và bản vẽ tùy chọn. Nó là một thư viện được thiết kế để tạo ra những hình ảnh mang tính cá nhân hóa hơn. Kiểu hiển thị, bảng màu, nhãn và trục của đồ họa có thể được tùy chỉnh và có thể sử dụng các tính năng vẽ nâng cao như đồ họa lớp, bóng và hình nền.

* **Thích hợp cho các tập dữ liệu lớn**

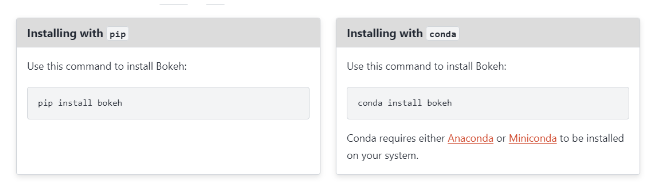
Một trong những tính năng nổi bật nhất của Hiệu ứng Bokeh là nó có thể hoạt động rất nhanh với các tập dữ liệu lớn. Hiệu ứng Bokeh có thể xử lý và trực quan hóa các tập dữ liệu mở rộng hiệu quả hơn. Hiệu suất của Hiệu ứng Bokeh không bị suy giảm khi kích thước tập dữ liệu tăng lên và tăng theo dấu gạch ngang. Đây là một lợi thế lớn trong các dự án phân tích dữ liệu có thể mở rộng.

Những ưu điểm này là một trong những lý lẽ quan trọng nhất giúp phân biệt Bokeh với các thư viện trực quan hóa dữ liệu khác. Chính vì vậy Bokeh cho phép người dùng trực quan hóa một cách mạnh mẽ và mang tính tương tác.

## Cài đặt

Bokeh được hỗ trợ và thử nghiệm chính thức trên Python 3.9 trở lên (CPython).

Bạn có thể cài đặt Bokeh bằng conda hoặc pip:



# CHƯƠNG 2: TRỰC QUAN HÓA SỐ LƯỢNG

## 2.1 Bar plot

* Những biểu đồ này thể hiện cho những bộ phim có doanh thu cao nhất vào cuối tuần từ 22 đến 24 tháng 12 năm 2017.
* Phương thức vbar()và hbar() được sử dụng để tạo các ô thanh dọc và ngang của bar plots.

**Cách vẽ:**

*Chuẩn bị dữ liệu*

file **=** "../data/csv\_files/movies.csv"

df **=** pd**.**read\_csv(file)

*# get only first part of title*

df["Title"] **=** df["Title"]**.**apply(**lambda** x: x**.**split(":")[0])

*# format the "weekend\_gross" values as integers*

*# divide values by 1,000,000 to save space on y-axis*

df["Weekend gross"] **=** df["Weekend gross"]**.**apply(

**lambda** x: (int(x**.**split("$")[1])) **/** 1\_000\_000

)

df

*Plotting*

*# plot a vertical bar*

*# create figure object*

p **=** figure(

x\_range**=**df**.**Title, *# range of categorical values for the x-axis*

height**=**300, *# plot height*

width**=**600, *# plot width*

title**=**"Figure 6.1 movie gross", *# title of plot*

y\_axis\_label**=**"weekend gross (million USD)", *# y-axis label*

)

p**.**vbar(

x**=**"Title", *# column name for x-axis values*

top**=**"Weekend gross", *# column name for the height of bars*

width**=**0.7, *# bar width*

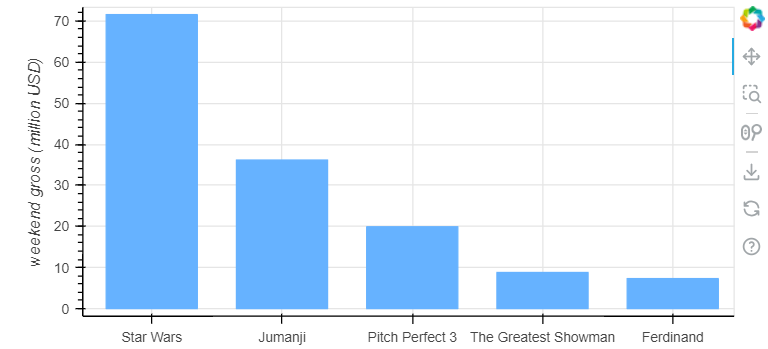
color**=**"#66B2FF", *# hex code for bar color*

source**=**df, *# data source for the column names*

)

p**.**y\_range**.**start **=** 0 *# start y-axis range from 0*

show(p) *# display plot*



*# plot a horizontal bar*

df **=** df**.**sort\_values("Weekend gross") *# sort the named column in ascending order*

p **=** figure(

y\_range**=**df**.**Title, *# range of categorical values for y-axis*

height**=**300,

title**=**"Figure 6.3 weekend gross",

x\_axis\_label**=**"weekend gross (million USD)",

sizing\_mode**=**"stretch\_width", *# make plot width responsive to size of screen*

)

p**.**hbar(

y**=**"Title", *# column name for y-axis values*

right**=**"Weekend gross", *# column name for right endpoints of bars*

height**=**0.9, *# bar height*

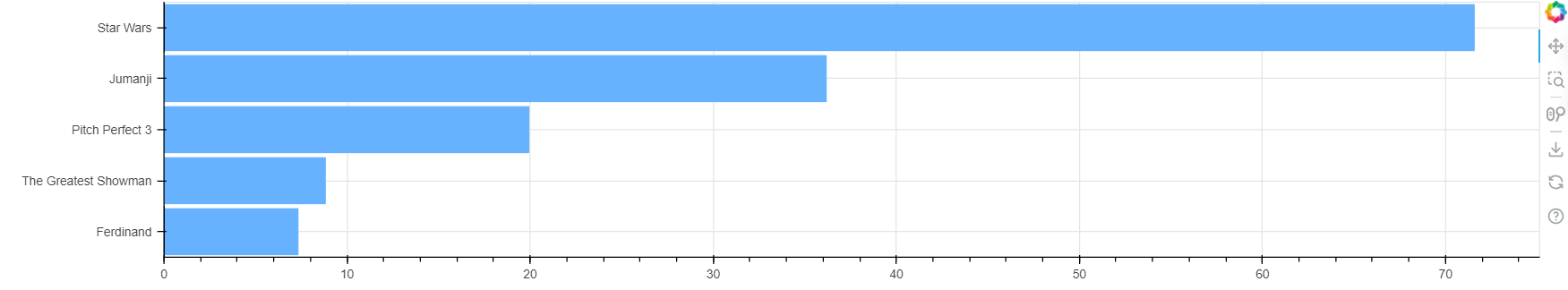
color**=**"#66B2FF",

source**=**df,

)

p**.**x\_range**.**start **=** 0

show(p)



**Lưu ý:**

- Dù đặt các thanh theo chiều dọc hay chiều ngang thì chúng ta cũng cần chú ý đến thứ tự sắp xếp các thanh khi không có thứ tự tự nhiên.

## 2.2 Grouped and stacked bar

### A. Grouped Bars

* Các ô này thể hiện thu nhập trung bình hàng năm của hộ gia đình Hoa Kỳ năm 2016 so với nhóm tuổi.
* Phương thức vbar() cũng được sử dụng để tạo các biểu đồ thanh được nhóm lại. Nó được gọi nhiều lần bằng số lượng nhóm bạn muốn vẽ.
* Phương thức [Dodge()](https://docs.bokeh.org/en/latest/docs/reference/transform.html#bokeh.transform.dodge) được sử dụng để bù đắp hoặc dịch chuyển các điểm dữ liệu phân loại một cách trực quan dọc theo một chiều nhất định (ở đây là trục x) để tránh chồng chéo trong biểu đồ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/income\_by\_age.csv"

df **=** pd**.**read\_csv(file)

*# select rows containing the different races and sort by "race" and "age"*

age\_group **=** (

df**.**sort\_values(["race", "age"])

**.**reset\_index(drop**=True**)

**.**iloc[7:35, :]

**.**reset\_index(drop**=True**)

)

*# group by "age" and "race" with the median income as values*

age\_group **=** age\_group**.**groupby(["age", "race"])[["median\_income"]]**.**sum()

age\_group **=** age\_group**.**unstack()**.**reset\_index()

age\_group**.**columns **=** age\_group**.**columns**.**droplevel(level**=**0)

age\_group **=** age\_group**.**rename(columns**=**{"": "age"})

age\_group

*Plotting*

**from** bokeh.models **import** NumeralTickFormatter **as** NTF

**from** bokeh.palettes **import** Blues5

**from** bokeh.transform **import** dodge

*# plot grouped bars*

p **=** figure(

title**=**"Figure 6.7 median income by age group",

height**=**350,

sizing\_mode**=**"stretch\_width",

x\_range**=**age\_group**.**age,

)

bar\_width **=** 0.2

p**.**vbar(

x**=**dodge("age", **-**0.3, range**=**p**.**x\_range), *# use dodge to create bar offsets on x-axis*

top**=**"asian",

source**=**age\_group,

width**=**bar\_width,

color**=**Blues5[0], *# add plot color using color palette*

legend\_label**=**"Asian", *# add legend label to plot*

)

p**.**vbar(

x**=**dodge("age", **-**0.1, range**=**p**.**x\_range),

top**=**"white",

source**=**age\_group,

width**=**bar\_width,

color**=**Blues5[1],

legend\_label**=**"White",

)

p**.**vbar(

x**=**dodge("age", 0.1, range**=**p**.**x\_range),

top**=**"hispanic",

source**=**age\_group,

width**=**bar\_width,

color**=**Blues5[2],

legend\_label**=**"Hispanic",

)

p**.**vbar(

x**=**dodge("age", 0.3, range**=**p**.**x\_range),

top**=**"black",

source**=**age\_group,

width**=**bar\_width,

color**=**Blues5[3],

legend\_label**=**"Black",

)

*# plot customization*

*# Remove x-axis lines, grid, and tick marks*

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_out **=** 0

p**.**xgrid**.**grid\_line\_color **=** **None**

*# remove y-axis lines and ticks*

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_out **=** 0

*# start and endpoints of y-axis*

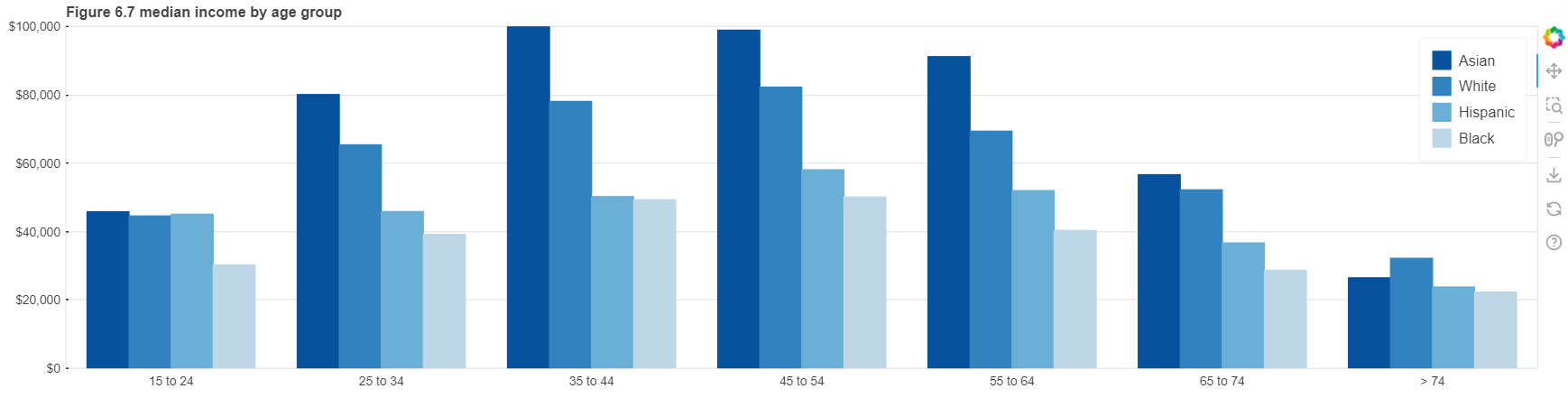
p**.**y\_range**.**start **=** 0

p**.**y\_range**.**end **=** 100\_000

*# format y-axis labels to $ sign with thousand delimiters*

p**.**yaxis**.**formatter **=** NTF(format**=**"$0,0")

show(p)



**Lưu ý:**

* Xác định đúng mục đích để lựa chọn các thuộc tính cho việc so sánh.
* Mã hóa theo màu thanh đòi hỏi nhiều sự nổ lực hơn, vì chúng ta phải so sánh màu sắc của các thanh với màu sắc trong chú giải trong đầu.

### B. Grid of bar plots

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# add a name for each race dataframe*

asian **=** df**.**iloc[21:28, :]**.**drop(["year", "race"], axis**=**1)

asian**.**name **=** "Asian"

black **=** df**.**iloc[28:35, :]**.**drop(["year", "race"], axis**=**1)

black**.**name **=** "Black"

hispanic **=** df**.**iloc[35:, :]**.**drop(["year", "race"], axis**=**1)

hispanic**.**name **=** "Hispanic"

white **=** df**.**iloc[7:14, :]**.**drop(["year", "race"], axis**=**1)

white**.**name **=** "White"

races **=** (asian, white, hispanic, black)

*Plotting:*

**import** pandas **as** pd

**from** bokeh.plotting **import** figure

**from** bokeh.models **import** FactorRange

**from** bokeh.models.formatters **import** NumeralTickFormatter **as** NTF

*# create a function to plot individual bar plots*

**def** plot\_bars(df: pd**.**DataFrame) **->** figure:

"""

Creates a bar chart using Bokeh to visualize median income by age.

Parameters:

df (pd.DataFrame): The pandas DataFrame containing the data.

It should have the following columns:

- age: String or numeric values representing the age groups.

- median\_income: Numeric values representing the median income for each age group.

Returns:

figure: A Bokeh figure object representing the bar chart.

Raises:

ValueError: If the required columns (age and median\_income) are not present in the DataFrame.

TypeError: If the data types of the columns are not compatible with the plot.

Example:

df = pd.DataFrame({'age': ['18-24', '25-34', '35-44'], 'median\_income': [50000, 60000, 70000]})

plot = plot\_bars(df)

show(plot)

"""

*# Data validation*

**if** "age" **not** **in** df**.**columns **or** "median\_income" **not** **in** df**.**columns:

**raise** ValueError("The DataFrame must have 'age' and 'median\_income' columns.")

**if** **not** pd**.**api**.**types**.**is\_numeric\_dtype(df["median\_income"]):

**raise** TypeError("The 'median\_income' column must contain numeric values.")

**if** **not** pd**.**api**.**types**.**is\_numeric\_dtype(df["age"]):

factors **=** df["age"]**.**unique()**.**tolist()

df["age"] **=** pd**.**Categorical(df["age"], categories**=**factors, ordered**=True**)

*# Function implementation*

p **=** figure(

title**=**f"Figure 6.9: {df**.**name}",

height**=**300,

width**=**400,

x\_range**=**FactorRange(factors**=**df**.**age),

toolbar\_location**=None**,

)

p**.**vbar(x**=**"age", top**=**"median\_income", color**=**"#99CCFF", source**=**df, width**=**0.9)

p**.**xgrid**.**grid\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_out **=** 0

p**.**yaxis**.**formatter **=** NTF(format**=**"$0,0")

p**.**xaxis**.**axis\_label **=** "age (years)"

p**.**yaxis**.**axis\_label **=** "median income (USD)"

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_out **=** 0

p**.**y\_range**.**start **=** 0

p**.**y\_range**.**end **=** 110\_000

**return** p

**from** bokeh.layouts **import** gridplot

*# plot the individual bars and append plots in a list*

plots **=** []

**for** race **in** races:

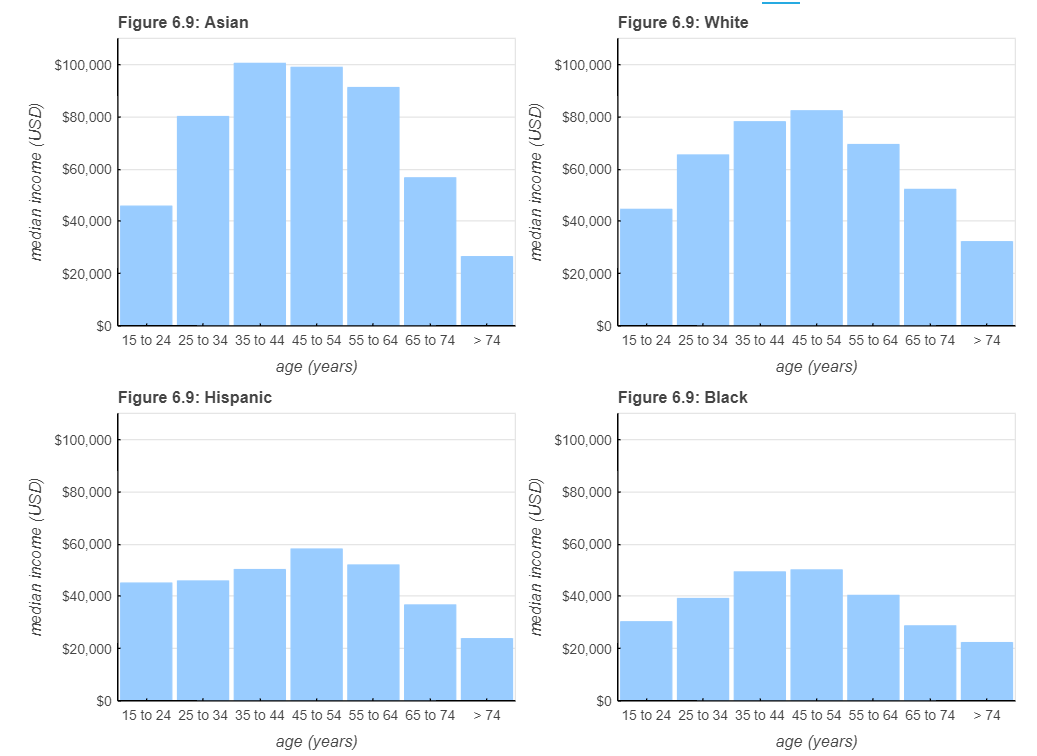
plot **=** plot\_bars(race)

plots**.**append(plot)

*# use gridplot to arrange the plots in rows of two*

layout **=** gridplot([plots[:2], plots[**-**2:]])

show(layout)



**Lưu ý**

* Lựa chọn này có ưu điểm là chúng ta không cần mã hóa biến phân loại theo màu thanh.
* Số lượng biểu đồ phụ thuộc vào số thuộc tính.

### C. Stacked bars

* Biểu đồ này thể hiện số lượng hành khách nam và nữ trên tàu Titanic ở khoang 1, 2 và 3.
* Phương thức vbar\_stack() được sử dụng để tạo biểu đồ thanh xếp chồng lên nhau.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/titanic\_all.csv"

df **=** pd**.**read\_csv(file)

*# group by class and count the passenger number by sex*

by\_class **=** (

df**.**groupby("class")**.**sex**.**value\_counts()**.**unstack()**.**drop("\*", axis**=**0)**.**reset\_index()

)

*# rename values in "class" column to match book labels*

by\_class["class"] **=** ["1st class", "2nd class", "3rd class"]

by\_class

*Plotting:*

*# plot stacked bars*

p **=** figure(

title**=**"Figure 6.10 titanic passengers",

height**=**300,

width**=**500,

x\_range**=**by\_class["class"],

toolbar\_location**=None**,

)

*# create stacked bars*

p**.**vbar\_stack(

["male", "female"], *# column names of bars to stack*

x**=**"class", *# column name for categorical x-axis*

source**=**by\_class, *# data source for column names*

width**=**0.9, *# bar width*

line\_width**=**2.5, *# increased line width to separate the bars*

line\_color**=**"white", *# line color separating the bars*

color**=**["#0000CC", "#CC6600"], *# colors for each bar stack*

legend\_label**=**["male passengers", "female passengers"],

)

*# add text to bars*

p**.**text(

x**=**"class", *# column name for horizontal position of text*

y**=**"male", *# column name for vertical position of text*

text**=**"male", *# text value*

source**=**by\_class, *# data source for column names*

y\_offset**=**40, *# vertical offset of text*

text\_color**=**"#FFFFFF",

text\_align**=**"center", *# align text to center of bars*

)

p**.**text(

x**=**"class",

y**=**"male",

text**=**"female",

source**=**by\_class,

y\_offset**=-**10,

text\_color**=**"#FFFFFF",

text\_align**=**"center",

)

*# plot customization*

*# remove x-axis line and ticks*

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**axis\_line\_width **=** 0

p**.**xaxis**.**major\_tick\_out **=** 0

*# remove y-axis lines, ticks, and labels. start y-axis at 0*

p**.**y\_range**.**start **=** 0

p**.**yaxis**.**visible **=** **False**

*# remove plot grid lines and outline*

p**.**grid**.**grid\_line\_color **=** **None**

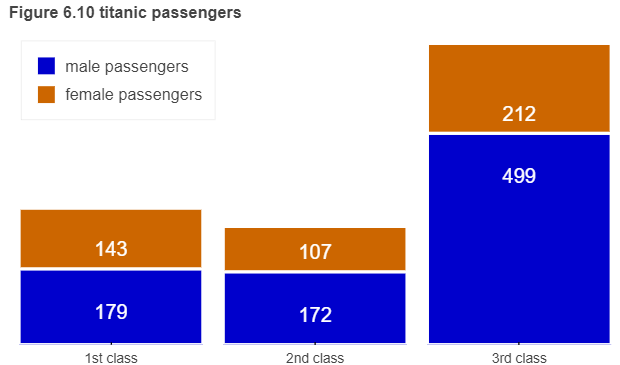
p**.**outline\_line\_color **=** **None**

*# position legend location and orientation*

p**.**legend**.**location **=** "top\_left"

p**.**legend**.**orientation **=** "vertical"

show(p)



**Lưu ý:**

- Phù hợp khi so sánh 2 thuộc tính, khi tăng số lượng thuộc tính gây khó khăn trong việc so sánh.

## 2.3 Dot plots and heatmaps

### A. Dot plots

* Đồ thị trong tiểu mục này thể hiện tuổi thọ trung bình của các quốc gia ở Châu Mỹ trong năm 2007.
* Phương thức scatter() được sử dụng để tạo các ô chấm.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/life\_expectancy.csv"

df **=** pd**.**read\_csv(file)

*# select only the relevant columns*

df **=** df**.**loc[:, ["country", "2007"]]

americas **=** (

"Argentina",

"Bolivia",

"Brazil",

"Canada",

"Chile",

"Colombia",

"Costa Rica",

"Cuba",

"Dominican Republic",

"Ecuador",

"El\_Salvador",

"Guatemala",

"Haiti",

"Honduras",

"Jamaica",

"Mexico",

"Nicaragua",

"Panama",

"Paraguay",

"Peru",

"Puerto Rico",

"Trinidad and Tobago",

"United States",

"Uruguay",

"Venezuela",

)

*# create a new dataframe consisting of only american countries*

df **=** df[df["country"]**.**isin(americas)]**.**reset\_index(drop**=True**)

df **=** df**.**rename(columns**=**{"2007": "years"})

df["years"] **=** df["years"]**.**round()

df

*Plotting:*

*# plot by country in alphabetical order*

*# sort dataframe by "country" column in descending order*

df **=** df**.**sort\_values("country", ascending**=False**)

*# create figure object*

p **=** figure(

title**=**"Figure 6.13 Life expectancy", *# plot title*

height**=**400, *# plot height*

y\_range**=**df**.**country, *# categorical range on the y-axis*

x\_axis\_label**=**"life expectancy (years)",

sizing\_mode**=**"stretch\_width", *# make plot width responsive to screen size*

)

*# create dot plot*

p**.**scatter(

x**=**"years", *# x-axis column name*

y**=**"country", *# y-axis column name*

source**=**df, *# data source for x and y axis*

size**=**8, *# circle size*

)

*# plot customization*

*# remove line color and minor ticks in x-axis*

p**.**xaxis**.**minor\_tick\_out **=** 0

p**.**xaxis**.**axis\_line\_color **=** **None**

*# remove line color in y-axis*

p**.**yaxis**.**axis\_line\_color **=** **None**

show(p) *# display plot*



**Lưu ý:**

* Cần chú ý đến thứ tự của các giá trị dữ liệu. Thay vào đó, nếu chúng ta sắp xếp chúng theo thứ tự bảng chữ cái, chúng ta sẽ có một đám mây điểm lộn xộn, gây nhầm lẫn và không truyền tải được thông điệp rõ ràng.
* Đối với các tập dữ liệu rất lớn, tùy chọn này đều không phù hợp vì con số thu được sẽ trở nên quá bận rộn khó trực quan.

### B. Heatmap

* Đồ thị trong tiểu mục này thể hiện việc áp dụng Internet theo thời gian ở một số quốc gia được chọn.
* Phương thức rect() được sử dụng để tạo bản đồ nhiệt.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/Internet\_user.csv"

df **=** pd**.**read\_csv(file, encoding**=**"ISO-8859-1")

countries **=** (

"Iceland",

"Norway",

"United Kingdom",

"Japan",

"Canada",

"Germany",

"New Zealand",

"France",

"Israel",

"United States",

"Argentina",

"Chile",

"Italy",

"Brazil",

"Mexico",

"South Africa",

"China",

"Algeria",

"India",

"Kenya",

)

*# create new dataframe with only the selected countries and columns*

df **=** df[df["country"]**.**isin(countries)]**.**reset\_index(drop**=True**)**.**fillna(0)

df **=** df**.**drop(["country\_code", "indicator", "indicator\_code"], axis**=**1)

*# stack dataframe columns*

df **=** pd**.**DataFrame(df**.**set\_index("country")**.**stack(), columns**=**["percentage"])

df **=** df**.**reset\_index()**.**rename(columns**=**{"level\_1": "year"})**.**fillna(0)

*# convert "year" column to integer type*

df["year"] **=** df**.**year**.**astype(int)

df

*Plotting:*

*# import relevenat libraries*

**from** bokeh.transform **import** transform

**from** bokeh.models **import** ColorBar, LinearColorMapper, FixedTicker

*# plot heatmap*

*# create figure object*

p **=** figure(

title**=**"Figure 6.15 Internet adoption over time", *# plot title*

height**=**400, *# plot height*

toolbar\_location**=None**, *# remove toolbars*

y\_axis\_location**=**"right", *# display y axis on the right of plot*

y\_range**=**countries[::**-**1], *# categorical range of y-axis in reverse order*

)

*# create color mapper object*

mapper **=** LinearColorMapper(

palette**=**"Magma256", low**=**min(df["percentage"]), high**=**max(df["percentage"])

)

*# create rectangle glyph*

p**.**rect(

x**=**"year", *# x-axis column name*

y**=**"country", *# y-axis column name*

width**=**2, *# rectangle width*

height**=**1, *# rectangle height*

source**=**df, *# data source for x and y axis columns*

*# map percentage values to color mapper object using transform*

fill\_color**=**transform("percentage", mapper),

line\_color**=**"white", *# rectangle line color*

)

*# plot customization*

*# configure x-axis ticks to show only specified tick labels*

p**.**xaxis**.**ticker **=** [1995, 2000, 2005, 2010, 2015]

*# start and end x-axis at the specified years*

p**.**x\_range**.**start **=** 1993

p**.**x\_range**.**end **=** 2016

*# remove x-axis major ticks*

p**.**xaxis**.**major\_tick\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_out **=** 0

*# remove y-axis lines and ticks*

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_line\_color **=** **None**

p**.**yaxis**.**axis\_line\_color **=** **None**

*# create color bar object*

color\_bar **=** ColorBar(

color\_mapper**=**mapper,

location**=**(0, 0),

ticker**=**FixedTicker(ticks**=**[0, 25, 50, 75, 100]),

title**=**"internet users / 100 people",

title\_text\_font\_style**=**"normal",

major\_tick\_line\_color**=None**,

width**=**300,

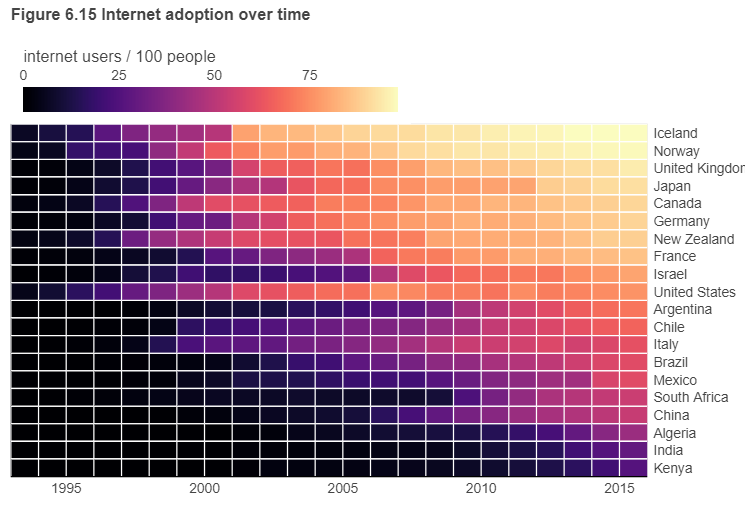
height**=**20,

)

*# add color bar above the plot*

p**.**add\_layout(color\_bar, "above")

show(p)



**Lưu ý:**

* Ánh xạ các giá trị dữ liệu vào màu sắc.
* Mặc dù cách trực quan hóa này khiến việc xác định các giá trị dữ liệu chính xác được hiển thị trở nên khó khăn hơn nhưng nó thực hiện rất tốt việc làm nổi bật các xu hướng rộng hơn.

# CHƯƠNG 3: Trực quan hóa sự phân bố: Biểu đồ và biểu đồ mật độ

## 3.1 Trực quan hóa một phân phối duy nhất

### A. Histograms

* Các đồ thị trong tiểu mục này thể hiện sự phân bổ độ tuổi của hành khách trên tàu Titanic.
* Phương thức quad() được sử dụng để tạo biểu đồ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/titanic\_all.csv"

df **=** pd**.**read\_csv(file)

ages **=** df**.**age**.**dropna()

ages

*Plotting:*

*# create a function to plot a histogram*

**def** plot\_hist(df, step, title):

"""

Create a histogram plot using Bokeh.

Args:

df (array-like): The data to be plotted as a histogram.

step (float): The width of each bin in the histogram.

title (str): The title of the plot.

Returns:

bokeh.plotting.figure.Figure: The Bokeh figure object representing the histogram plot.

Raises:

ValueError: If the input data is empty or contains invalid values.

"""

*# data validation*

**if** len(df) **==** 0:

**raise** ValueError("Input data is empty.")

**if** **not** all(isinstance(val, (int, float)) **for** val **in** df):

**raise** ValueError(

"Input data contains invalid values. Expected numerical values."

)

*# function implementation*

bins **=** np**.**arange(0, 80, step) *# create bins*

hist, edges **=** np**.**histogram(df, bins**=**bins) *# create histogram and bin edges*

*# create figure object*

p **=** figure(

title**=**title, *# plot title*

height**=**300, *# plot height*

width**=**400, *# plot width*

toolbar\_location**=None**, *# remove toolbars*

x\_axis\_label**=**"age (years)",

y\_axis\_label**=**"count",

)

*# create histogram plot*

p**.**quad(

top**=**hist, *# frequency value of each bin*

bottom**=**0, *# start of rectangle on y-axis*

left**=**edges[:**-**1], *# left edges of bin*

right**=**edges[1:], *# right edges of bin*

fill\_color**=**"#52A9EB", *# rectangle color*

line\_color**=**"white", *# line color of rectangle*

)

*# plot customization*

*# customise x-axis*

p**.**x\_range**.**start **=** 0

p**.**xaxis**.**ticker **=** [0, 20, 40, 60]

p**.**xgrid**.**grid\_line\_color **=** **None**

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_line\_color **=** "gray"

p**.**xaxis**.**major\_tick\_out **=** 2

*# customise y-axis*

p**.**y\_range**.**start **=** 0

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_line\_color **=** "gray"

p**.**yaxis**.**major\_tick\_out **=** 0

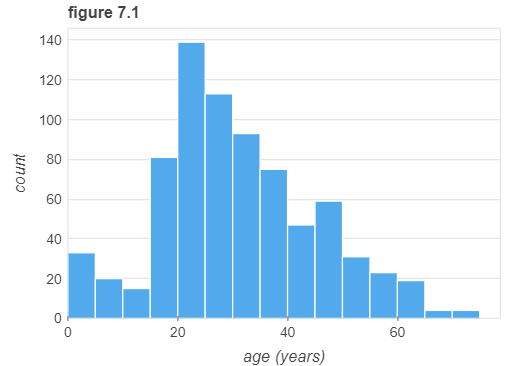
p**.**yaxis**.**major\_tick\_in **=** 0

**return** p

*# plot a single histogram*

single **=** plot\_hist(ages, 5, "figure 7.1")

show(single)

**

**Lưu ý:**

* Phụ thuộc vào chiều rộng của bin.
* Nếu chiều rộng của ngăn quá nhỏ thì biểu đồ sẽ trở nên quá đỉnh và rối mắt về mặt thị giác, đồng thời các xu hướng chính trong dữ liệu có thể bị che khuất.
* Mặt khác, nếu chiều rộng thùng quá lớn thì các đặc điểm nhỏ hơn trong việc phân bổ dữ liệu

### B. Density plots

* Các đồ thị trong tiểu mục này thể hiện ước tính mật độ hạt nhân về sự phân bổ độ tuổi của hành khách trên tàu Titanic.
* Phương thức patch() được sử dụng để tạo ra các ô mật độ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

values **=** ages**.**values

positions **=** np**.**linspace(**-**10, 80, 1000)

*Plotting:*

**from** sklearn.neighbors **import** KernelDensity

*# create a function to plot a Kernel density estimate*

**def** plot\_kde(kernel, bandwidth, title):

"""

Create a kernel density estimation (KDE) plot using Bokeh.

Args:

kernel (str): The type of kernel to be used in the KDE estimation.

bandwidth (float): The bandwidth parameter for the KDE.

title (str): The title of the plot.

Returns:

bokeh.plotting.figure.Figure: The Bokeh figure object representing the KDE plot.

Raises:

ValueError: If the input values for `kernel` or `bandwidth` are invalid.

"""

*# data validation*

supported\_kernels **=** [

"gaussian",

"tophat",

"epanechnikov",

"exponential",

"linear",

"cosine",

]

**if** kernel **not** **in** supported\_kernels:

**raise** ValueError(

f"Invalid kernel: '{kernel}'. Supported kernels are: {supported\_kernels}"

)

**if** bandwidth **<=** 0:

**raise** ValueError("Bandwidth must be a positive value.")

*# function implementation*

*# create kde object and fit object into 'values' parameter*

kde **=** KernelDensity(kernel**=**kernel, bandwidth**=**bandwidth)**.**fit(values[:, np**.**newaxis])

*# calculate log-density estimation (log\_dens) at each position using the 'score\_samples' method*

log\_dens **=** kde**.**score\_samples(positions[:, np**.**newaxis])

*# create figure object*

p **=** figure(

title**=**title, *# plot title*

height**=**300, *# plot height*

width**=**600, *# plot width*

x\_axis\_label**=**"age (years)",

y\_axis\_label**=**"density",

)

*# draw filled area on the figure using `patch` glyph*

p**.**patch(

positions, *# x-axis coordinates*

np**.**exp(log\_dens), *# y-axis coordinates*

fill\_alpha**=**0.9, *# fill transparency*

fill\_color**=**"#52A9EB",

line\_color**=**"black",

)

*# customize x-axis*

p**.**x\_range**.**start **=** 0

p**.**xaxis**.**ticker **=** [0, 20, 40, 60]

p**.**xgrid**.**grid\_line\_color **=** **None**

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_line\_color **=** "gray"

p**.**xaxis**.**major\_tick\_out **=** 2

*# customize y-axis*

p**.**y\_range**.**start **=** 0

p**.**yaxis**.**ticker **=** [0, 0.01, 0.02, 0.03, 0.04]

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_line\_color **=** "gray"

p**.**yaxis**.**major\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_in **=** 0

**return** p

*# plot four density plots and arrange them in a 2x2 grid*

half **=** plot\_kde("gaussian", 0.5, "a")

two **=** plot\_kde("gaussian", 2, "b")

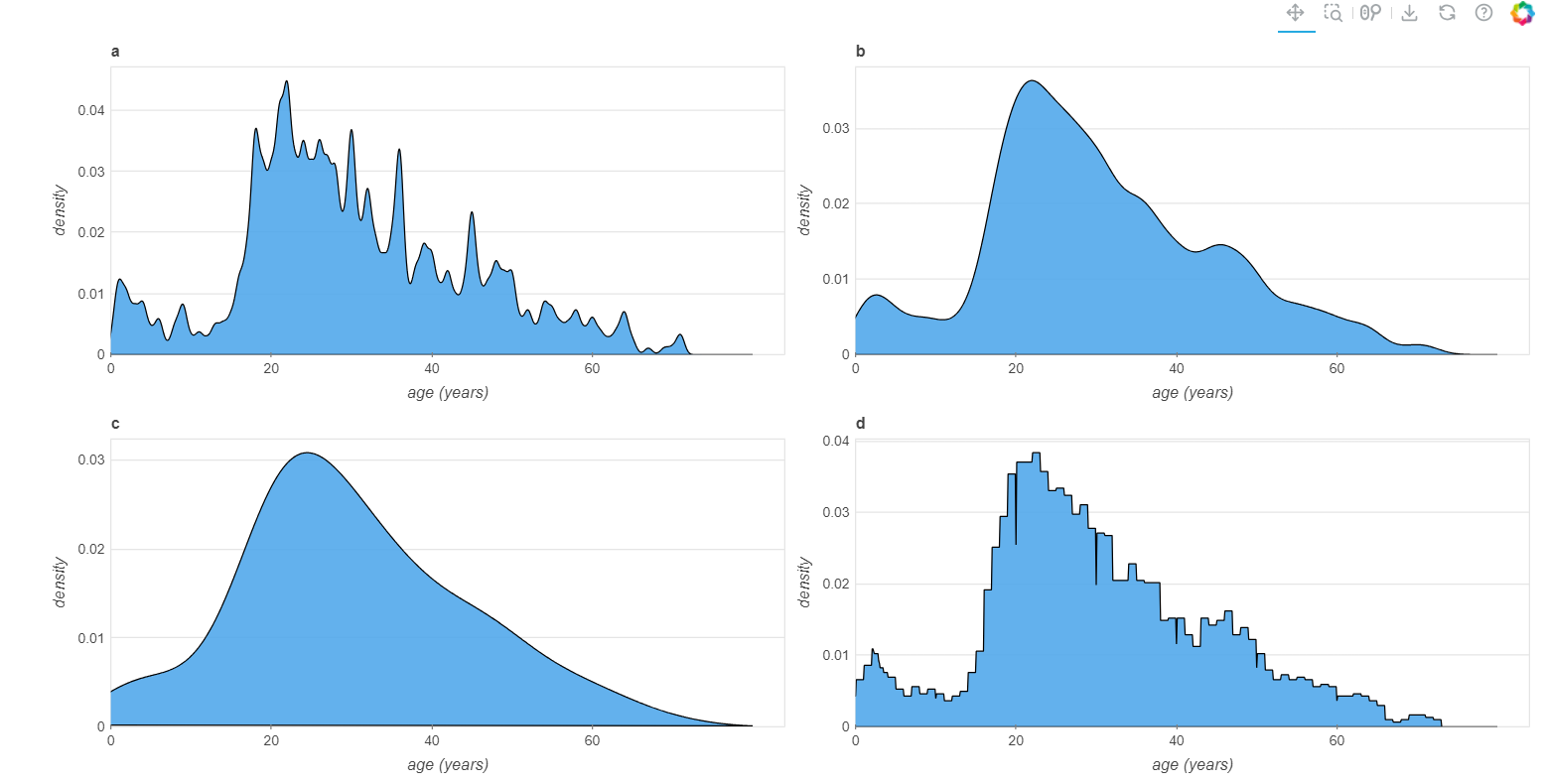
five **=** plot\_kde("gaussian", 5, "c")

rect **=** plot\_kde("tophat", 2, "d")

layout **=** gridplot([[half, two], [five, rect]])

show(layout)

export\_png(layout, filename**=**"../images/grid\_density.png");

**

**Lưu ý:**

* Tương tự Histogram, tính trực quan phụ thuộc vào việc lựa chọn băng thông

## 3.2 Trực quan hóa nhiều bản phân phối cùng một lúc

### A. Multiple distribution density plots

* Các đồ thị trong tiểu mục này thể hiện sự phân bổ độ tuổi của hành khách nam và nữ trên tàu Titanic.
* Phương thức varea() được sử dụng để tạo ra các ô mật độ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/titanic\_all.csv"

*# create new dataframe with only the relevant columns*

titanic **=** pd**.**read\_csv(file)

titanic **=** titanic**.**drop(["name", "class", "survived"], axis**=**1)

*# create new dataframe for male and female passengers respectively*

female **=** titanic[titanic["sex"] **==** "female"]

male **=** titanic[titanic["sex"] **==** "male"]

*# get age data for the density plots.*

f\_values **=** female**.**age**.**dropna()**.**values

m\_values **=** male**.**age**.**dropna()**.**values

t\_values **=** titanic**.**age**.**dropna()**.**values

*Plotting:*

**from** bokeh.plotting **import** figure, show

**from** sklearn.neighbors **import** KernelDensity

**import** numpy **as** np

*# create function to plot the multiple density estimates*

**def** plot\_kde(data\_dict, title, kernel**=**"gaussian", bandwidth**=**2, line\_color**=None**):

"""

Create a density plot using Kernel Density Estimation (KDE) for multiple datasets.

Parameters:

data\_dict (list of dicts): A list of dictionaries, where each dictionary contains the following keys:

- 'data': The data to be plotted.

- 'color': The color of the filled area.

- 'legend\_label': The legend label for the dataset.

title (str): The title of the plot.

kernel (str, optional): The type of kernel to use in creating the plot. Default is gaussian.

bandwidth (float, optional): The bandwidth of the KDE. Higher values result in smoother

but less accurate density plots. Default is 2.

line\_color (str, optional): The color of the lines around the filled areas.

Default is None, which means no lines will be drawn.

Returns:

bokeh.plotting.figure.Figure: The Bokeh figure containing the density plot.

"""

positions **=** np**.**linspace(**-**10, 80, 1000)

*# create figure object*

p **=** figure(

title**=**title, *# plot title*

height**=**300, *# plot height*

width**=**500, *# plot width*

toolbar\_location**=None**, *# remove toolbars*

x\_axis\_label**=**"age (years)",

y\_axis\_label**=**"scaled density",

)

*# loop through each data\_dict and plot a density plot for each*

**for** info **in** data\_dict:

data **=** info["data"]

color **=** info["color"]

legend\_label **=** info["legend\_label"]

*# create kde object and fit object into 'data' parameter*

kde **=** KernelDensity(kernel**=**kernel, bandwidth**=**bandwidth)**.**fit(data[:, np**.**newaxis])

*# calculate log-density estimation (log\_dens) at each position using the 'score\_samples' method*

density **=** np**.**exp(kde**.**score\_samples(positions[:, np**.**newaxis]))

*# scale the density estimation to correspond to the number of data values*

scaled\_density **=** density **\*** len(data)

p**.**varea(

x**=**positions, *# x-axis coordinates*

y1**=**scaled\_density, *# y-axis coordinates of one side of the area*

y2**=**0, *# y-axis coordinates of the other side of the area*

fill\_alpha**=**0.9, *# transparency of the filled area*

fill\_color**=**color, *# color of the filled area*

legend\_label**=**legend\_label,

)

*# customize the x-axis*

p**.**x\_range**.**start **=** 0

p**.**xaxis**.**ticker **=** [0, 20, 40, 60]

p**.**xgrid**.**grid\_line\_color **=** **None**

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_line\_color **=** "gray"

p**.**xaxis**.**major\_tick\_out **=** 2

*# customize the y-axis*

p**.**y\_range**.**start **=** 0

p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_line\_color **=** "gray"

p**.**yaxis**.**major\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_in **=** 0

p**.**legend**.**location **=** "top\_right"

**return** p

*# generate a single multiple density plot*

data\_dict **=** [

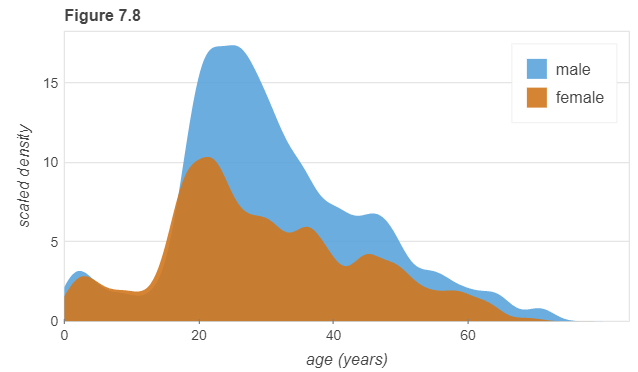
{"data": m\_values, "color": "#5BA4DB", "legend\_label": "male"},

{"data": f\_values, "color": "#D0771E", "legend\_label": "female"},

]

single **=** plot\_kde(data\_dict, "Figure 7.8", line\_color**=**"black")

show(single)

**

*# generate two multiple density plots and display them in a grid*

**from** bokeh.layouts **import** gridplot

male\_data **=** [

{"data": t\_values, "color": "#D5D4D3", "legend\_label": "all passengers"},

{"data": m\_values, "color": "#055BB2", "legend\_label": "male"},

]

female\_data **=** [

{"data": t\_values, "color": "#D5D4D3", "legend\_label": "all passengers"},

{"data": f\_values, "color": "#CB6805", "legend\_label": "female"},

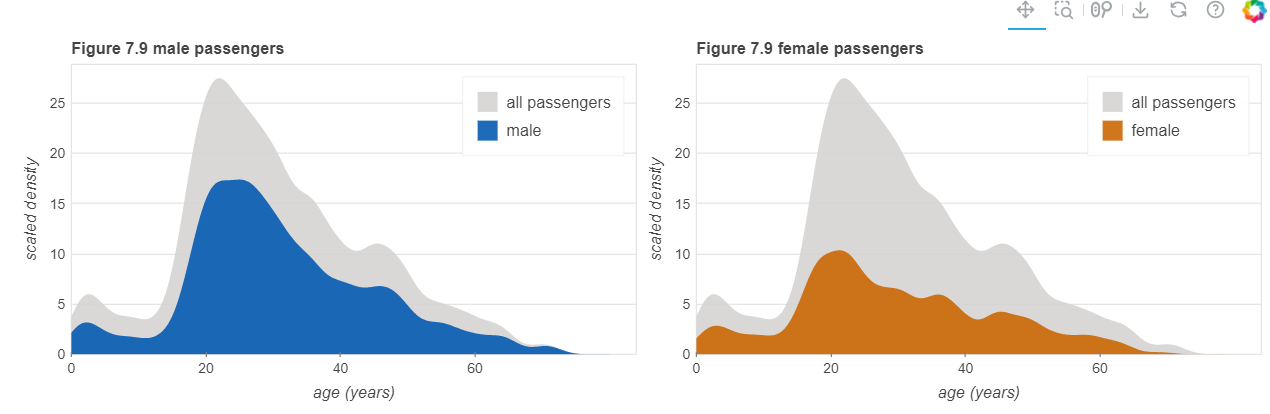
]

male **=** plot\_kde(male\_data, "Figure 7.9 male passengers")

female **=** plot\_kde(female\_data, "Figure 7.9 female passengers")

layout **=** gridplot([male, female], ncols**=**2)

show(layout)

**

**Lưu ý:**

* Đối với tập dữ liệu cụ thể này, sự phân bổ độ tuổi của hành khách nam và nữ gần như giống hệt nhau cho đến khoảng 17 tuổi và sau đó phân kỳ, do đó việc hiển thị kết quả vẫn chưa lý tưởng.

### B. Multiple distribution density plots lồng nhau

* Đồ thị trong tiểu mục này thể hiện tỷ lệ phần trăm chất béo trong sữa của bốn giống gia súc.
* Phương thức varea() cũng được sử dụng để tạo ra các ô mật độ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/cows.csv"

df **=** pd**.**read\_csv(file)

*# create dataframes for the four different cattle breeds*

jersey **=** df[df["breed"] **==** "Jersey"]

holstein **=** df[df["breed"] **==** "Holstein-Friesian"]

guernsey **=** df[df["breed"] **==** "Guernsey"]

ayrshire **=** df[df["breed"] **==** "Ayrshire"]

*# get butterfat data for the cattle breeds*

j\_values **=** jersey**.**butterfat**.**values

h\_values **=** holstein**.**butterfat**.**values

g\_values **=** guernsey**.**butterfat**.**values

a\_values **=** ayrshire**.**butterfat**.**values

positions **=** np**.**linspace(2, 8, 1000)

*Plotting:*

**from** bokeh.models **import** Label, CustomJSTickFormatter

*# arrange plotting data as pandas DataFrame*

data\_dict **=** {

"values": [a\_values, g\_values, h\_values, j\_values],

"bandwidths": [0.125, 0.25, 0.1, 0.3],

"colors": ["#409DFA", "#AC5703", "#9E5205", "green"],

"labels": ["Ayrshire", "Guernsey", "Holstein-Friesian", "Jersey"],

}

df **=** pd**.**DataFrame(data\_dict)

*# Create figure object*

p **=** figure(

title**=**"figure 7.11", *# plot title*

height**=**300, *# plot height*

width**=**600, *# plot width*

x\_axis\_label**=**"butterfat contents",

y\_axis\_label**=**"density",

)

*# Loop to calculate KDE and plot vareaes*

**for** \_, row **in** df**.**iterrows():

data, bandwidth, color, label **=** (

row["values"],

row["bandwidths"],

row["colors"],

row["labels"],

)

kde **=** KernelDensity(kernel**=**"gaussian", bandwidth**=**bandwidth)**.**fit(data[:, np**.**newaxis])

log\_dens **=** kde**.**score\_samples(positions[:, np**.**newaxis])

p**.**varea(

x**=**positions,

y1**=**np**.**exp(log\_dens),

y2**=**0,

fill\_alpha**=**0.3,

fill\_color**=**color,

)

*# Find the highest point and annotate with the label*

max\_idx **=** np**.**argmax(np**.**exp(log\_dens))

highest\_point\_label **=** Label(

x**=**positions[max\_idx],

y**=**np**.**exp(log\_dens[max\_idx]),

text**=**label,

text\_font\_size**=**"10pt",

x\_offset**=**20,

y\_offset**=-**5,

text\_color**=**color,

)

p**.**add\_layout(highest\_point\_label)

*# Convert x-axis labels to percentages*

x\_axis\_labels **=** {3: "3%", 4: "4%", 5: "5%", 6: "6%", 7: "7%"}

p**.**xaxis**.**formatter **=** CustomJSTickFormatter(

code**=**"""return tick in %s ? %s[tick] : '';""" **%** (x\_axis\_labels, x\_axis\_labels)

)

*# customize x-axis*

p**.**x\_range**.**start **=** 3

p**.**xgrid**.**grid\_line\_color **=** **None**

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_line\_color **=** "gray"

p**.**xaxis**.**major\_tick\_out **=** 2

p**.**xaxis**.**minor\_tick\_out **=** 0

*# customize y-axis*

p**.**yaxis**.**ticker **=** [0, 0.5, 1, 1.5]

p**.**y\_range**.**start **=** 0

p**.**yaxis**.**minor\_tick\_out **=** 0

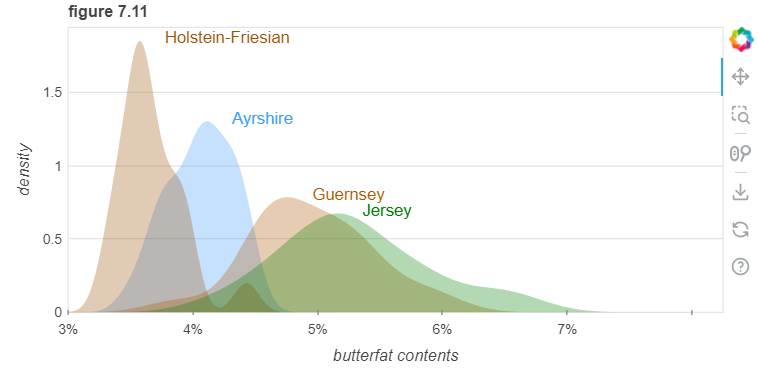
p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_line\_color **=** "gray"

p**.**yaxis**.**major\_tick\_out **=** 0

p**.**yaxis**.**major\_tick\_in **=** 0

show(p)

**

**Lưu ý:**

* Sử dụng trong trường hợp có nhiều phân phối.

### C. Multiple distribution histogram

* Cốt truyện trong tiểu mục này thể hiện sự phân bổ độ tuổi của hành khách nam và nữ trên tàu Titanic.
* Phương thức hbar() được sử dụng để tạo biểu đồ biểu đồ.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# create dataframe for only the males and females*

female **=** titanic[titanic["sex"] **==** "female"]

male **=** titanic[titanic["sex"] **==** "male"]

*# extract age data from the dataframes*

m\_age **=** male**.**age**.**dropna()

f\_age **=** female**.**age**.**dropna()

*# compute histograms for both datasets*

bins **=** np**.**arange(0, 80, 3)

m\_hist, edges **=** np**.**histogram(m\_age, bins**=**bins)

f\_hist, edges **=** np**.**histogram(f\_age, bins**=**bins)

*Plotting:*

**from** bokeh.models **import** Label, CustomJSTickFormatter

*# create figure object*

p **=** figure(

title**=**"Figure 7.10", *# plot title*

height**=**400, *# plot height*

width**=**600, *# plot width*

x\_range**=**(**-**60, 40), *# range of x-axis values to display*

toolbar\_location**=None**, *# remove toolbars*

x\_axis\_label**=**"count",

y\_axis\_label**=**"age (years)",

)

*# plot male histogram*

p**.**hbar(

right**=**m\_hist **\*** **-**1, *# right endpoints of bars*

y**=**edges[1:], *# y-axis values*

height**=**2, *# bar height*

color**=**"#055BB2",

)

*# plot female histogram*

p**.**hbar(

right**=**f\_hist,

y**=**edges[1:],

height**=**2,

color**=**"#CB6805",

)

*# customise x-axis and y-axis*

p**.**xaxis**.**ticker **=** [**-**40, **-**20, 0, 20, 40]

p**.**yaxis**.**ticker **=** [0, 20, 40, 60]

p**.**y\_range**.**start **=** 1.5

*# create custom formatter function to make all tick labels positive*

**def** positive\_labels():

**return** "return Math.abs(tick);"

*# apply the custom formatter to the x-axis using CustomJSTickFormatter*

p**.**xaxis**.**formatter **=** CustomJSTickFormatter(args**=**dict(), code**=**positive\_labels())

*# add labels*

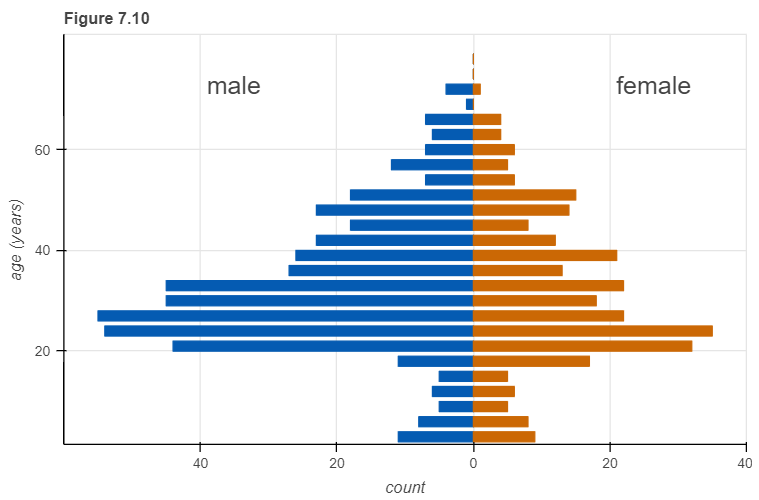
m\_label **=** Label(x**=-**40, y**=**70, text**=**"male", text\_font\_size**=**"15pt", x\_offset**=**5)

f\_label **=** Label(x**=**20, y**=**70, text**=**"female", text\_font\_size**=**"15pt", x\_offset**=**5)

p**.**add\_layout(m\_label)

p**.**add\_layout(f\_label)

show(p)

**

**Lưu ý:**

* Thủ thuật này không hoạt động khi có nhiều hơn hai phân phối mà chúng ta muốn hình dung cùng một lúc

# CHƯƠNG 4:  TRỰC QUAN HÓA PHÂN PHỐI: HÀM PHÂN PHỐI TÍCH LŨY VÀ BIỂU ĐỒ QQ

## 4.1 Hàm phân phối tích lũy

* Ecdf điểm học sinh. Thứ hạng học sinh đã được chuẩn hóa theo tổng số học sinh, sao cho các giá trị y được vẽ tương ứng với tỷ lệ học sinh trong lớp có tối đa số điểm đó.
* Phương thức p.step() được sử dụng để thể hiện ECD.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# Khởi tạo dữ liệu giống như ví dụ matplotlib*

*points = np.random.randint(40, 101, 50) # Giả định dữ liệu*

*sorted\_points = np.sort(points) # Sắp xếp điểm*

*Plotting:*

*# Tính ECDF*

*ecdf = np.arange(1, len(sorted\_points) + 1) / len(sorted\_points)*

*# Tạo ColumnDataSource từ dữ liệu*

*source = ColumnDataSource(data=dict(sorted\_points=sorted\_points, ecdf=ecdf))*

*# Khởi tạo đối tượng biểu đồ*

*p = figure(title='Empirical Cumulative Distribution Function (ECDF) of Student Grades', tools='', background\_fill\_color="#fafafa")*

*p.xgrid.grid\_line\_color = 'white'*

*# Vẽ đường ECDF*

*p.step('sorted\_points', 'ecdf', source=source, line\_width=2, mode="after", legend\_label='ECDF')*

*# Cài đặt nhãn cho trục*

*p.xaxis.axis\_label = 'Points'*

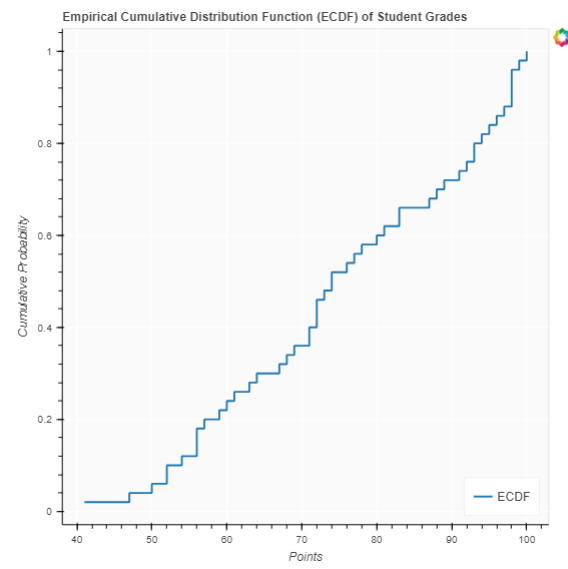
*p.yaxis.axis\_label = 'Cumulative Probability'*

*# Hiển thị lưới và cụm chú thích*

*p.grid.grid\_line\_color="white"*

*p.legend.location = "bottom\_right"*

*show(p)*

**

**Lưu ý:**

* Trong các ứng dụng thực tế, khá phổ biến là vẽ ECDF mà không làm nổi bật các điểm riêng lẻ và chuẩn hóa các thứ hạng theo thứ hạng tối đa.
* ECDF rất hữu ích cho việc ấn định ranh giới điểm vì chúng giúp xác định các giới hạn chính xác nhằm giảm thiểu sự không hài lòng của học sinh.

## 4.2 Phân phối Highly skewed

### A. Densify và Cumulative frequency

* Phân bố số lượng cư dân tại các quận của Hoa Kỳ, theo Điều tra dân số Hoa Kỳ năm 2010.
* Phương thức p.quad() và p.line() được sử dụng để vẽ

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*df = pd.read\_csv("data/csv\_files/US\_census.csv")*

*# Assuming 'population\_counts' stores the 'pop2010' column data as before*

*population\_counts = df['pop2010'].dropna() # Assuming 'df' is your DataFrame*

*# Prepare data for density plot*

*hist, edges = np.histogram(population\_counts, density=True, bins=50)*

*hist\_df = pd.DataFrame({'hist': hist, 'left': edges[:-1], 'right': edges[1:]})*

*Plotting:*

*# Create a Bokeh ColumnDataSource*

*source\_hist = ColumnDataSource(hist\_df)*

*# Density Plot*

*p\_density = figure(title='Density Plot of Population Counts', tools='', background\_fill\_color="#fafafa")*

*p\_density.quad(top='hist', bottom=0, left='left', right='right', source=source\_hist,*

*fill\_color='navy', line\_color='white', alpha=0.5)*

*# ECDF*

*sorted\_counts = np.sort(population\_counts)*

*ecdf = np.arange(1, len(sorted\_counts)+1) / len(sorted\_counts)*

*p\_ecdf = figure(title='Empirical Cumulative Distribution Function (ECDF)', tools='', background\_fill\_color="#fafafa")*

*p\_ecdf.line(sorted\_counts, ecdf, legend\_label='ECDF', line\_width=2)*

*# Layout*

*p\_density.xaxis.axis\_label = 'Population Counts'*

*p\_density.yaxis.axis\_label = 'Density'*

*p\_ecdf.xaxis.axis\_label = 'Population Counts'*

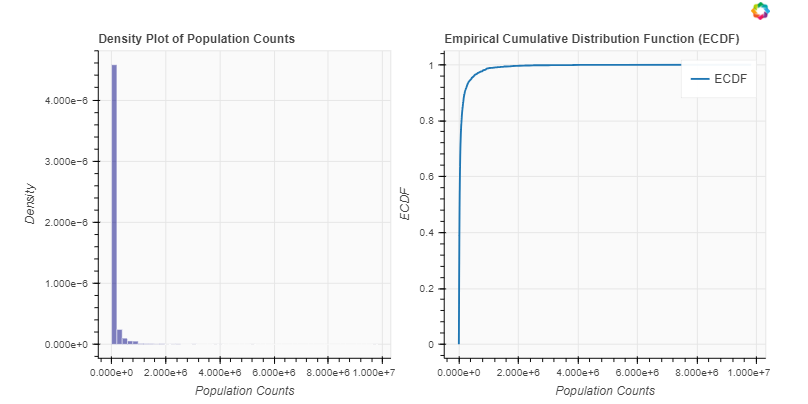
*p\_ecdf.yaxis.axis\_label = 'ECDF'*

*grid = gridplot([[p\_density, p\_ecdf]], width=400, height=400)*

*# Show the plot*

*output\_notebook()*

*show(grid)*

**

**Lưu ý:**

* Hiển thị các phân phối có độ lệch cao, đặc biệt là với các phần đuôi nặng ở bên phải và các phân phối này có thể khó hình dung.
* Do số lượng dân số ở các quận thực tế không phải là luật lũy thừa mà thay vào đó tuân theo phân bố log-chuẩn sẽ gần như hoàn hảo.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# First, log-transform the 'pop2010' column, similar to your Plotly example.*

*df['log\_pop2010'] = np.log10(df['pop2010'])*

*# Prepare the data for the density plot*

*data = df['log\_pop2010'].dropna()*

*hist, edges = np.histogram(data, bins=30, density=True)*

*hist\_df = pd.DataFrame({'top': hist, 'left': edges[:-1], 'right': edges[1:]})*

*# Prepare the data for the ECDF plot*

*df\_sorted = df.sort\_values(by='log\_pop2010')*

*df\_sorted['ecdf'] = np.arange(1, len(df\_sorted) + 1) / len(df\_sorted)*

*Plotting:*

*# Creating the density plot*

*density\_plot = figure(title='Density Plot - Log-transformed Distribution of Population Counts in US Counties (2010 Census)', tools='', background\_fill\_color="#fafafa")*

*density\_plot.quad(top='top', bottom=0, left='left', right='right', source=hist\_df, fill\_color="navy", line\_color="white", alpha=0.5)*

*density\_plot.add\_tools(HoverTool(tooltips=[("Population (log10)", "$index"), ("Density", "@top")]))*

*density\_plot.xaxis.axis\_label = 'log10(Number of Inhabitants)'*

*density\_plot.yaxis.axis\_label = 'Density'*

*# Creating the ECDF plot*

*ecdf\_plot = figure(title='ECDF - Log-transformed Distribution of Population Counts in US Counties (2010 Census)', tools='', background\_fill\_color="#fafafa")*

*ecdf\_plot.circle(df\_sorted['log\_pop2010'], df\_sorted['ecdf'], size=3, color="navy", alpha=0.5)*

*ecdf\_plot.add\_tools(HoverTool(tooltips=[("log10(Population)", "@x"), ("ECDF", "@y")]))*

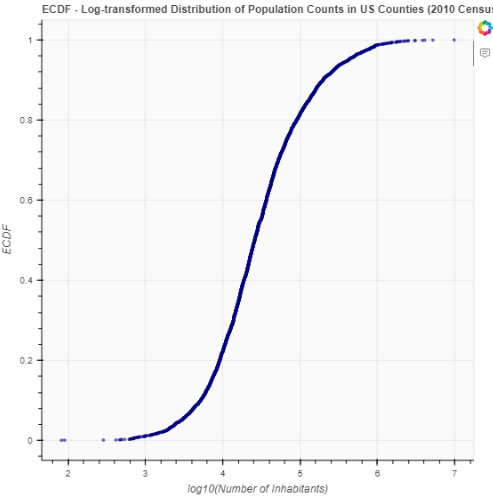
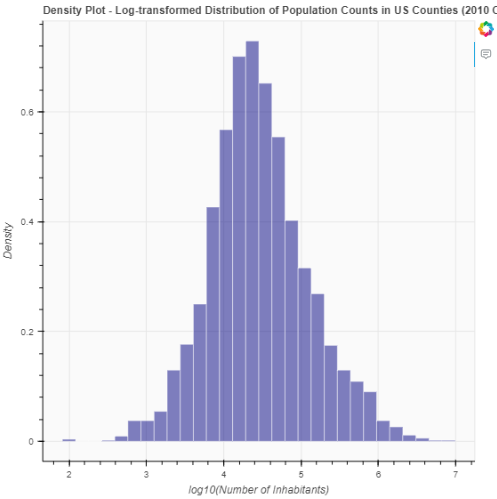
*ecdf\_plot.xaxis.axis\_label = 'log10(Number of Inhabitants)'*

*ecdf\_plot.yaxis.axis\_label = 'ECDF'*

*# Layout and show*

*layout = row(density\_plot, ecdf\_plot)*

*show(layout)*

**

### B. Relative frequency

- Tần suất tương đối của các quận có ít nhất số dân đó so với số dân của quận.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# Sort the DataFrame by 'pop2010' in descending order*

*df\_sorted = df.sort\_values(by='pop2010', ascending=False)*

*# Calculate the cumulative distribution function values as a fraction*

*cdf\_values = np.linspace(0., 1., len(df\_sorted))*

*Plotting:*

*# Creating a ColumnDataSource from the sorted dataframe and CDF values*

*source = ColumnDataSource(data=dict(pop2010=df\_sorted['pop2010'], cdf\_values=cdf\_values))*

*TOOLS = "pan,wheel\_zoom,box\_zoom,reset"*

*# Line plot for Relative Frequency of Counties versus Number of Inhabitants*

*p1 = figure(tools=TOOLS, width=600, height=600, min\_border=10, min\_border\_left=50,*

*toolbar\_location="above", x\_axis\_type="log", y\_axis\_type="log",*

*title="Relative Frequency of Counties versus Number of Inhabitants")*

*p1.background\_fill\_color = "#fafafa"*

*# Add a line glyph*

*p1.line(x='pop2010', y='cdf\_values', source=source, color="#3B8686", alpha=0.8, line\_width=2, legend\_label="pop2010")*

*p1.xgrid.grid\_line\_color = None*

*p1.ygrid.grid\_line\_color = None*

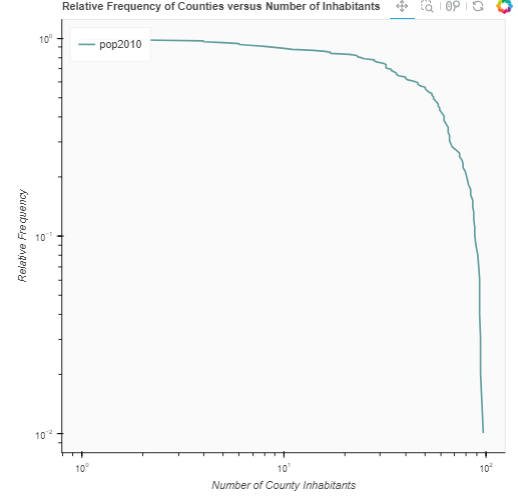
*p1.xaxis.axis\_label = "Number of County Inhabitants"*

*p1.yaxis.axis\_label = "Relative Frequency"*

*p1.legend.location = "top\_left"*

*# Displaying the plot*

*show(p1)*

**

**Lưu ý:**

* Để thấy rằng phân phối này không phải là một định luật lũy thừa, chúng ta vẽ nó dưới dạng ecdf giảm dần với trục x và y logarit.
* Trong hình dung này, định luật lũy thừa xuất hiện dưới dạng một đường thẳng hoàn hảo.

Phân bổ số từ trong tiểu thuyết Moby Dick

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*file = "data/csv\_files/Moby\_Dick.csv"*

*moby\_dick\_df = pd.read\_csv(file)*

*#sorting and cumulative fraction calculation as before*

*moby\_dick\_df = moby\_dick\_df.sort\_values(by='count', ascending=False)*

*cumulative\_fraction = np.cumsum(moby\_dick\_df['count']) / np.sum(moby\_dick\_df['count'])*

*Plotting:*

*hover\_tool = HoverTool(*

*tooltips=[('Number of times word is used', '@x'),*

*('Fraction of words', '@y')]*

*)*

*p = figure(title='Distribution of Word Counts in Moby Dick',*

*x\_axis\_label='Number of times word is used',*

*y\_axis\_label='Fraction of words',*

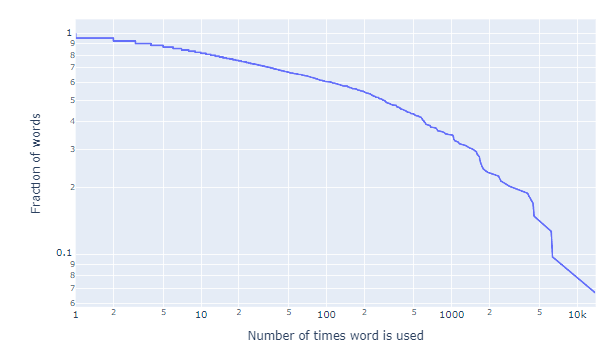
*x\_axis\_type='log',*

*y\_axis\_type='log',*

*tools=[hover\_tool])*

*p.line(x=moby\_dick\_df['count'], y=cumulative\_fraction, color='navy', alpha=0.5)*

*show(p)*

**

**Lưu ý:**

* Sự phân bố này tuân theo quy luật lũy thừa hoàn hảo

## 4.3 Biểu đồ quantile–quantile

### A. Phân bố chuẩn

* Đồ thị qq điểm của học sinh

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# Assume data with mean 10 and standard deviation 3*

*mean = 10*

*std\_dev = 3*

*# Generate random data for demonstration*

*np.random.seed(42)*

*data = np.random.normal(mean, std\_dev, 100)*

*# Calculate theoretical quantiles for a normal distribution*

*theoretical\_quantiles = stats.norm.ppf(np.linspace(0.01, 0.99, 100), loc=mean, scale=std\_dev)*

*# Sort observed data*

*sorted\_data = np.sort(data)*

*Plotting:*

*# Create q-q plot using Bokeh*

*p = figure(title='Quantile-Quantile (q-q) Plot',*

*x\_axis\_label='Observed Values',*

*y\_axis\_label='Theoretical Values',*

*width=500, height=400)*

*# Scatter plot for observed vs theoretical quantiles*

*p.circle(sorted\_data, theoretical\_quantiles, size=5, color="navy", alpha=0.5, legend\_label='q-q plot')*

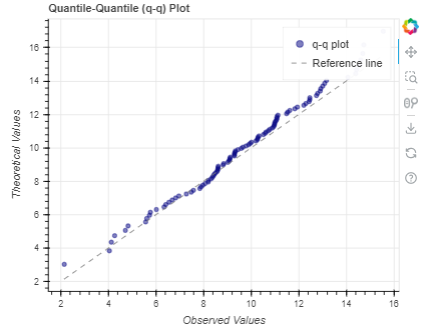
*# Diagonal line for reference*

*p.line([np.min(sorted\_data), np.max(sorted\_data)], [np.min(sorted\_data), np.max(sorted\_data)],*

*line\_dash='dashed', color="gray", legend\_label='Reference line')*

*# Show plot*

*show(p)*

**

**Lưu ý:**

* Trong biểu đồ qq không vẽ đồ thị thứ hạng một cách trực tiếp, chúng tôi sử dụng chúng để dự đoán vị trí của một điểm dữ liệu nhất định nếu dữ liệu được phân phối theo một phân phối tham chiếu cụ thể.
* Thông thường nhất, các ô qq được xây dựng bằng cách sử dụng phân phối chuẩn làm tham chiếu.

### B. Phân bố log-normally

* Biểu đồ qq logarit của số lượng cư dân tại các quận của Hoa Kỳ

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*# Đọc dữ liệu từ file CSV*

*file = "data/csv\_files/US\_census.csv"*

*df = pd.read\_csv(file)*

*# Chọn cột chứa số dân (ví dụ: pop2010)*

*population\_column = 'pop2010'*

*# Lọc các dòng có giá trị không rỗng trong cột số dân*

*filtered\_df = df.dropna(subset=[population\_column])*

*# Tính giá trị logarit của số dân*

*log\_population\_counts = np.log(filtered\_df[population\_column])*

*# Tính giá trị quantiles theo phân phối chuẩn*

*theoretical\_quantiles = stats.norm.ppf(np.linspace(0.01, 0.99, len(log\_population\_counts)),*

*loc=np.mean(log\_population\_counts),*

*scale=np.std(log\_population\_counts))*

*# Sắp xếp giá trị logarit*

*sorted\_data = np.sort(log\_population\_counts)*

*Plotting:*

*# Initiate Bokeh plot with similar features*

*p = figure(title=f'Quantile-Quantile (Q-Q) Plot of Log-transformed {population\_column} in US Counties',*

*x\_axis\_label='Theoretical Values',*

*y\_axis\_label='Observed Values')*

*# Adding scatter plot for observed vs theoretical quantiles*

*p.scatter(theoretical\_quantiles, sorted\_data, legend\_label='Q-Q plot')*

*# Adding reference line*

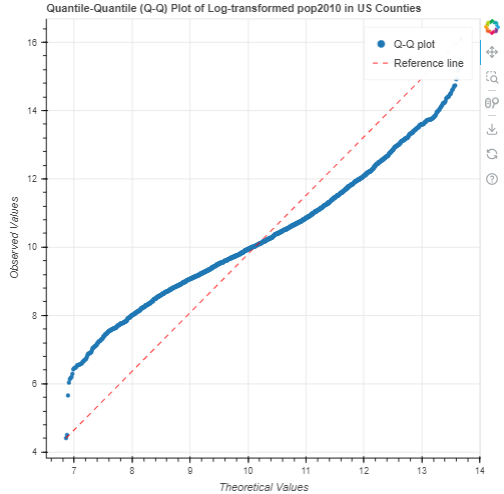
*p.line([np.min(theoretical\_quantiles), np.max(theoretical\_quantiles)],*

*[np.min(sorted\_data), np.max(sorted\_data)],*

*legend\_label='Reference line', line\_dash='dashed', color="red")*

*# Show plot*

*show(p)*

**

# CHƯƠNG 5: Trực quan hóa nhiều bản phân phối cùng một lúc

## 5.1 Trực quan hóa sự phân bố dọc theo trục tung

### A. Boxplot

* Biểu đồ trong tiểu mục này biểu thị nhiệt độ trung bình hàng ngày ở Lincoln, Nebraska vào năm 2016. Đường ở giữa biểu đồ hình hộp biểu thị giá trị trung bình và hộp bao quanh 50% dữ liệu ở giữa.
* Các râu trên và dưới kéo dài đến mức tối đa và tối thiểu nằm trong khoảng 1,5 lần chiều cao của hộp.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/lincoln.csv"

df **=** pd**.**read\_csv(file)

df["DATE"] **=** pd**.**to\_datetime(df["DATE"])

df["TAVG"] **=** (df["TMAX"] **+** df["TMIN"]) **/** 2

df["MONTH"] **=** df**.**DATE**.**dt**.**strftime("%b")

df **=** df[

[

"MONTH",

"TMIN",

"TMAX",

"TAVG",

]

]

qs **=** df**.**groupby("MONTH")**.**TAVG**.**quantile([0.25, 0.5, 0.75])**.**unstack()**.**reset\_index()

qs**.**columns **=** ["MONTH", "Q1", "Q2", "Q3"]

iqr **=** qs**.**Q3 **-** qs**.**Q1

qs["upper"] **=** qs**.**Q3 **+** 1.5 **\*** iqr

qs["lower"] **=** qs**.**Q1 **-** 1.5 **\*** iqr

df **=** pd**.**merge(df, qs, on**=**"MONTH", how**=**"left")

df**.**head()

*Plotting:*

**from** bokeh.models **import** ColumnDataSource, Whisker

**from** bokeh.plotting **import** figure, show

*# create figure object*

p **=** figure(

title**=**"Figure 9.3",

x\_range**=**df**.**MONTH**.**unique(),

toolbar\_location**=None**,

height**=**400,

width**=**600,

x\_axis\_label**=**"month",

y\_axis\_label**=**"mean temperature (F)",

)

*# create column data source object from the dataframe*

source **=** ColumnDataSource(df)

*# create whisker object and add it to figure*

whisker **=** Whisker(base**=**"MONTH", upper**=**"upper", lower**=**"lower", source**=**source)

whisker**.**upper\_head**.**size **=** whisker**.**lower\_head**.**size **=** 20

p**.**add\_layout(whisker)

*# create boxplot using two vbar() glyphs*

p**.**vbar(

x**=**"MONTH",

top**=**"Q2",

bottom**=**"Q1",

width**=**0.8,

source**=**source,

color**=**"#E0E0E0",

line\_color**=**"black",

)

p**.**vbar(

x**=**"MONTH",

top**=**"Q3",

bottom**=**"Q2",

width**=**0.8,

source**=**source,

color**=**"#E0E0E0",

line\_color**=**"black",

)

*# plot outliers using scatter() glyph*

outliers **=** df[**~**df**.**TAVG**.**between(df**.**lower, df**.**upper)]

p**.**scatter("MONTH", "TAVG", source**=**outliers, size**=**5, color**=**"black")

*# customize plot*

p**.**y\_range**.**start **=** **-**10

p**.**yaxis**.**ticker **=** [0, 25, 50, 75]

p**.**grid**.**grid\_line\_color **=** **None**

show(p)

**

**Lưu ý:**

* Biểu đồ hộp đơn giản nhưng chứa nhiều thông tin và chúng hoạt động tốt khi được vẽ cạnh nhau để trực quan hóa nhiều phân bố cùng một lúc.

### B. Sina plot

* Đồ thị trong tiểu mục này thể hiện nhiệt độ trung bình hàng ngày ở Lincoln, Nebraska vào năm 2016. Sử dụng cùng một tệp với boxplot.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

**import** numpy **as** np

months **=** list(df**.**MONTH**.**unique())

*Plotting:*

**from** sklearn.neighbors **import** KernelDensity

In [7]:

*# create figure object*

p **=** figure(

title**=**"Figure 9.8",

x\_range**=**months,

toolbar\_location**=None**,

height**=**400,

width**=**500,

x\_axis\_label**=**"month",

y\_axis\_label**=**"mean temperature (F)",

)

*# create an offset for each category in the data*

**def** offset(category, data, scale**=**7):

**return** list(zip([category] **\*** len(data), scale **\*** data))

**for** month **in** months:

month\_df **=** df[df**.**MONTH **==** month]**.**dropna()

tavg **=** month\_df**.**TAVG**.**values

temps **=** np**.**linspace(tavg**.**min(), tavg**.**max(), 50)

*# calculate the KDE for each month and plot the data*

kde **=** KernelDensity(kernel**=**"gaussian", bandwidth**=**3)**.**fit(tavg[:, np**.**newaxis])

density **=** np**.**exp(kde**.**score\_samples(temps[:, np**.**newaxis]))

x1, x2 **=** offset(month, density), offset(month, **-**density)

p**.**harea(x1**=**x1, x2**=**x2, y**=**temps, alpha**=**0.8, color**=**"#E0E0E0")

*# create jitter to offset temperature data points*

tavg\_density **=** np**.**exp(kde**.**score\_samples(tavg[:, np**.**newaxis]))

jitter **=** (np**.**random**.**random(len(tavg)) **\*** 2 **-** 1) **\*** tavg\_density

*# create a scatter plot for the average temp in each month*

p**.**scatter(

x**=**offset(month, jitter),

y**=**tavg,

color**=**"black",

)

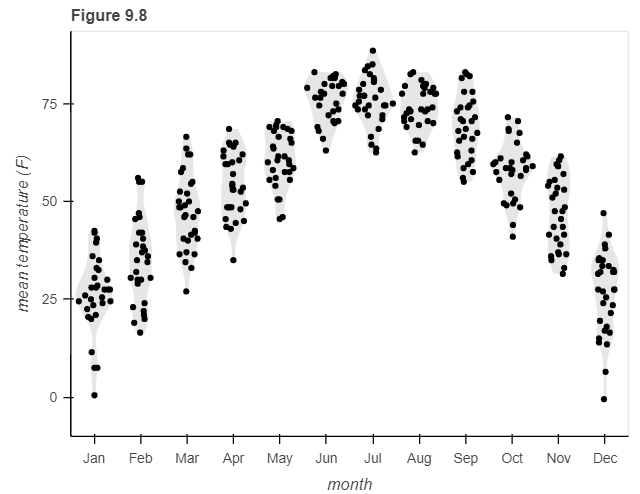
*# customize plot*

p**.**y\_range**.**start **=** **-**10

p**.**yaxis**.**ticker **=** [0, 25, 50, 75]

p**.**grid**.**grid\_line\_color **=** **None**

show(p)

**

**Lưu ý:**

* Sự kết hợp Violin + Điểm riêng lẻ, mang ưu điểm của cả 2.

## 5.2 Trực quan hóa phân bố dọc theo trục ngang

### A. Ridgeline plot

* Đồ thị trong tiểu mục này thể hiện mô hình bỏ phiếu tại Hạ viện Hoa Kỳ trong những năm qua. Điểm DW-NOMINATE thường được sử dụng để so sánh mô hình bỏ phiếu giữa các bên và theo thời gian. Tại đây, sự phân bổ điểm số được hiển thị riêng cho từng Quốc hội từ năm 1963 đến năm 2013 cho Đảng Dân chủ và Đảng Cộng hòa. Mỗi Đại hội được đại diện bởi năm đầu tiên (cột dim\_1 trong khung dữ liệu).
* Phương thức patch() được sử dụng cho cốt truyện.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/dw\_nominate\_house.csv"

df **=** pd**.**read\_csv(file)

*# add year column by multiplying each congress by 2 from the year 1787*

df["year"] **=** (df**.**congress) **\*** 2 **+** 1787

*# select only the relevant columns from the year 1963 onwards*

year **=** df["year"] **>=** 1962

parties **=** (df["party\_code"] **==** 100) **|** (df["party\_code"] **==** 200)

dn **=** (df["cd"] **!=** 0) **&** (df["cd"] **!=** 98) **&** (df["cd"] **!=** 99)

df **=** df[year **&** parties **&** dn]**.**reset\_index(drop**=True**)

*# create two dataframes for both political parties*

dems **=** df[df["party\_code"] **==** 100]

repubs **=** df[df["party\_code"] **==** 200]

In [9]:

**from** bokeh.models **import** ColumnDataSource

**import** numpy **as** np

**from** scipy.stats **import** gaussian\_kde

**from** sklearn.neighbors **import** KernelDensity

*# create a column data source containing the KDE data for each*

**def** tweak\_df(df):

"""

Calculate Kernel Density Estimates (KDE) for each year in the input DataFrame.

Parameters:

df (pd.DataFrame): Input DataFrame with columns "year" and "dim\_1".

Returns:

bokeh.models.ColumnDataSource: A ColumnDataSource containing KDE data for each year.

Raises:

ValueError: If the input DataFrame does not have the required columns or if there's an issue with KDE calculation.

"""

*# Validate input DataFrame*

**if** **not** isinstance(df, pd**.**DataFrame):

**raise** ValueError("Input must be a pandas DataFrame.")

**if** "year" **not** **in** df**.**columns **or** "dim\_1" **not** **in** df**.**columns:

**raise** ValueError("Input DataFrame must have columns 'year' and 'dim\_1'.")

grouped **=** df**.**groupby("year")**.**dim\_1

*# create Column Data Source object to hold the KDE data*

scores **=** np**.**linspace(**-**1.5, 1.5, 500)

blank **=** np**.**linspace(100, 200, 500)

source **=** ColumnDataSource(data**=**dict(scores**=**scores))

*# add extra columns to create blank space in the KDE plot*

source**.**add(blank, "1957")

source**.**add(blank, "1959")

source**.**add(blank, "1961")

**def** ridge(category, data):

**return** list(zip([category] **\*** len(data), data))

*# Calculate KDE for each column and create plotting ridge*

**try**:

**for** year, vote **in** grouped:

year **=** str(year)

pdf **=** gaussian\_kde(vote)

y **=** ridge(year, pdf(scores))

source**.**add(y, year)

**except** Exception **as** e:

**raise** ValueError(f"Error occurred during KDE calculation: {str(e)}")

**return** source

*Plotting:*

**from** bokeh.models **import** ColumnDataSource

**from** bokeh.plotting **import** figure

**def** plot\_ridges(df1, df2**=None**):

"""

Plot ridges for two DataFrames.

Parameters:

df1: ColumnDataSource containing data for the first category.

df2 (optional): ColumnDataSource containing data for the second category.

Returns:

bokeh.plotting.figure: A Bokeh figure showing the ridges for both categories.

"""

*# Input validation*

**if** **not** isinstance(df1, ColumnDataSource):

**raise** ValueError(f"{df1} must be a valid ColumnDataSource")

**if** df2 **is** **not** **None** **and** **not** isinstance(df2, ColumnDataSource):

**raise** ValueError(f"{df2} must be a valid ColumnDataSource")

*# Get the list of categories from the data keys*

years **=** list(reversed(df1**.**data**.**keys()))[:**-**1]

*# Create a figure object*

p **=** figure(

title**=**"Figure 9.12",

x\_axis\_label**=**"DW-NOMINATE score",

y\_axis\_label**=**"year",

toolbar\_location**=None**,

x\_range**=**(**-**0.75, 1.5),

y\_range**=**years,

height**=**400,

width**=**600,

)

*# Plot ridges for each category*

**for** i, year **in** enumerate(years):

*# Plot ridges for the first category*

p**.**patch(

x**=**"scores",

y**=**year,

source**=**df1,

fill\_color**=**"blue",

line\_color**=**"white",

legend\_label**=**"Democrats",

fill\_alpha**=**0.5,

)

*# Plot ridges for the second category if df2 is provided*

**if** df2 **is** **not** **None**:

*# Input validation: Check if df2 has the same keys as df1*

**if** **not** set(df2**.**data**.**keys()) **==** set(df1**.**data**.**keys()):

**raise** ValueError(f"{df2} must have the same keys as {df1}")

p**.**patch(

x**=**"scores",

y**=**year,

source**=**df2,

fill\_color**=**"red",

line\_color**=**"white",

legend\_label**=**"Republicans",

alpha**=**0.4,

)

*# Customize plot*

p**.**xaxis**.**ticker **=** [**-**0.75, **-**0.5, **-**0.25, 0.00, 0.25, 0.5, 0.75, 1.00]

p**.**ygrid**.**grid\_line\_color **=** **None**

p**.**legend**.**orientation **=** "horizontal"

p**.**legend**.**border\_line\_color **=** **None**

**return** p

In [11]:

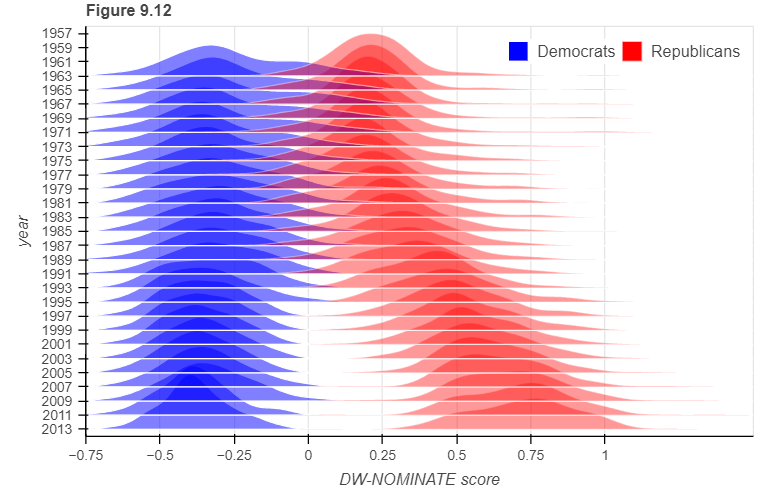
*# create ridgeline plot for both dataframes*

rep\_cds **=** tweak\_df(repubs)

dem\_cds **=** tweak\_df(dems)

ridgeplot **=** plot\_ridges(dem\_cds, rep\_cds)

show(ridgeplot)

**

**Lưu ý:**

* Biểu đồ Ridgeline cũng hoạt động tốt nếu chúng ta muốn so sánh hai xu hướng theo thời gian

# CHƯƠNG 6: TRỰC QUAN HÓA TỈ LỆ

## 6.1. Trường hợp biểu đồ tròn

### A. Pie chart

* Thành phần đảng của Hạ viện Đức lần thứ 8, 1976–1980, được minh họa dưới dạng biểu đồ hình tròn. Hình dung này cho thấy rõ ràng rằng liên minh cầm quyền của SPD và FDP chiếm đa số nhỏ so với CDU/CSU đối lập.
* Phương thức wedge() được sử dụng để tạo biểu đồ tròn.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*df = pd.read\_csv('data/csv\_files/bundestag.csv')*

*df['angle'] = df['seats']/df['seats'].sum() \* 2\*pi*

*Plotting:*

*p = figure(height=350, title="Figure 10.1", toolbar\_location=None,*

*tools="hover", tooltips="@party: @seats", x\_range=(-0.5, 1.0))*

*p.wedge(x=0, y=1, radius=0.4,*

*start\_angle=cumsum('angle', include\_zero=True), end\_angle=cumsum('angle'),*

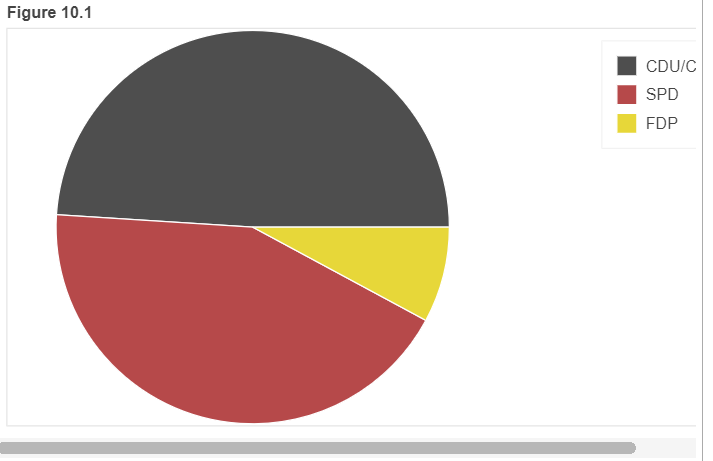
*line\_color="white", fill\_color='colors', legend\_field='party', source=df)*

*p.axis.axis\_label = None*

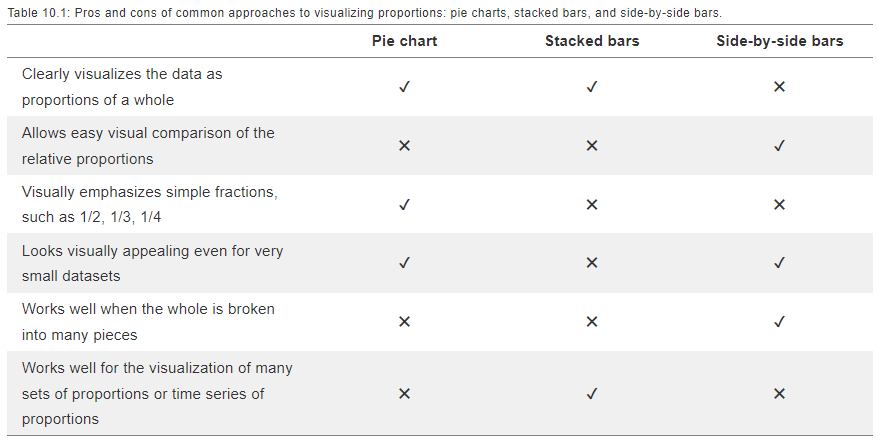
*p.axis.visible = False*

*p.grid.grid\_line\_color = None*

*show(p)*

**

**Lưu ý:**

****

## 6.2. Trường hợp các thanh cạnh nhau.

### A. side-by-side bars

### Biểu đồ thể hiện Thị phần của năm công ty giả định trong giai đoạn 2015–2017, được hiển thị dưới dạng các thanh cạnh nhau.

### Phương thức vbar\_stack() được sử dụng nhiều lần để tạo các nhóm cột.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*df = pd.read\_csv('marketshare.csv')*

*# Prepare the data*

*df['year'] = df['year'].astype(str) # Ensure the years are strings for the ColumnDataSource*

*factors = list(zip(df['year'], df['company']))*

*Plotting:*

*source = ColumnDataSource(data=dict(factors=factors, percent=df['percent'], company=df['year']))*

*p = figure(x\_range=FactorRange(\*factors), height=250, title="Market Share by Company",*

*toolbar\_location=None, tools="")*

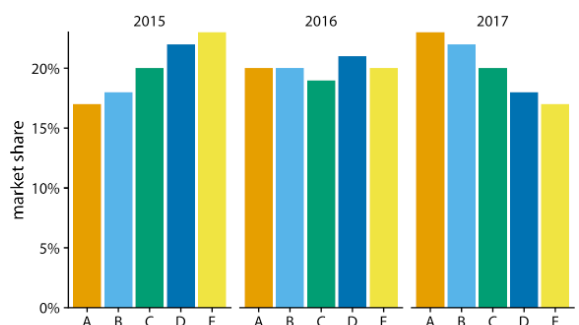
*p.vbar(x='factors', top='percent', width=0.9, source=source,*

*line\_color="white", fill\_color=factor\_cmap('factors', palette=Viridis5, factors=sorted(df['year'].unique()), start=1, end=2))*

*p.y\_range.start = 0*

*p.x\_range.range\_padding = 0.1*

*show(p)*

**

**Lưu ý:**

* Thể hiện xu hướng theo thang đo nhất định
* Nên sắp xếp theo thứ tự để tránh gây khó nhìn.

## 10.3 Trường hợp thanh xếp chồng và khối lượng riêng xếp chồng

### A. Stacked bars

* Biểu đồ thể hiện thay đổi sức khỏe theo độ tuổi.
* Sử dụng vbar\_stack() để vẽ

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*age\_groups = ["[10, 20)", "[20, 30)", "[30, 40)", "[40, 50)", "[50, 60)", "[60, 70)", "[70, 80)", "[80, 90)"]*

*healthdata = {'excellent': [0.395683, 0.393610, 0.369036, 0.325994, 0.276467, 0.211799, 0.167411, 0.154081],*

*'good': [0.446043, 0.467081, 0.482771, 0.462074, 0.428977, 0.406520, 0.391263, 0.385202],*

*'fair': [0.148201, 0.125598, 0.127229, 0.169886, 0.214222, 0.275893, 0.303253, 0.302822],*

*'poor': [0.010072, 0.013711, 0.020964, 0.042045, 0.080333, 0.105788, 0.138074, 0.157895]}*

*Plotting:*

*from bokeh.plotting import figure, show*

*from bokeh.models import ColumnDataSource, FactorRange*

*from bokeh.palettes import Spectral4*

*# Create a ColumnDataSource object*

*source = ColumnDataSource(data=dict(age\_groups=age\_groups, \*\*healthdata))*

*# Create the figure*

*p = figure(x\_range=FactorRange(\*age\_groups), height=350, title="Health Status by Age Group",*

*toolbar\_location=None, tools="")*

*# Add the vertical stacked bars*

*p.vbar\_stack(['excellent', 'good', 'fair', 'poor'], x='age\_groups', width=0.9, source=source,*

*color=Spectral4, legend\_label=['excellent', 'good', 'fair', 'poor'])*

*# Set some properties to make the plot look better*

*p.y\_range.start = 0*

*p.y\_range.end = 1*

*p.xaxis.axis\_label = "Age Group"*

*p.yaxis.axis\_label = "Proportion"*

*p.add\_layout(p.legend[0], 'right')*

*show(p)*

**

**Lưu ý:**

* Khó nhìn khi thanh xếp nhiều ngăn, vị trí các thanh bên trong thay đổi dọc theo chuổi.

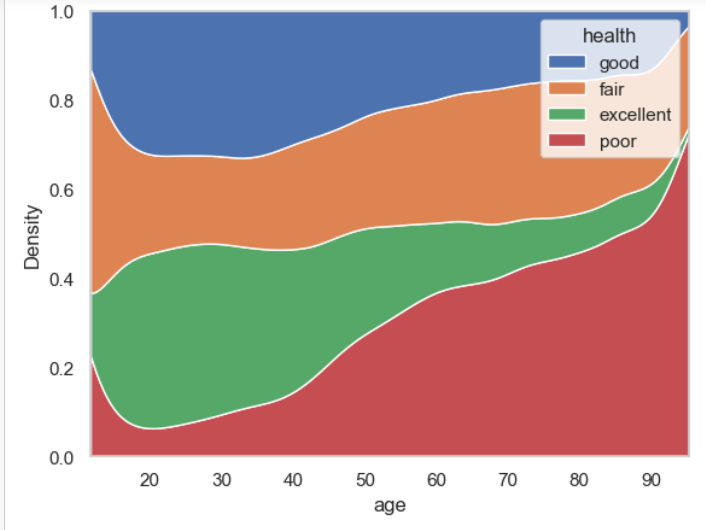
### B. Stacked densities

### Tình trạng sức khỏe theo độ tuổi theo báo cáo của Tổng điều tra xã hội (GSS).

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*Plotting:*

**

**Lưu ý:**

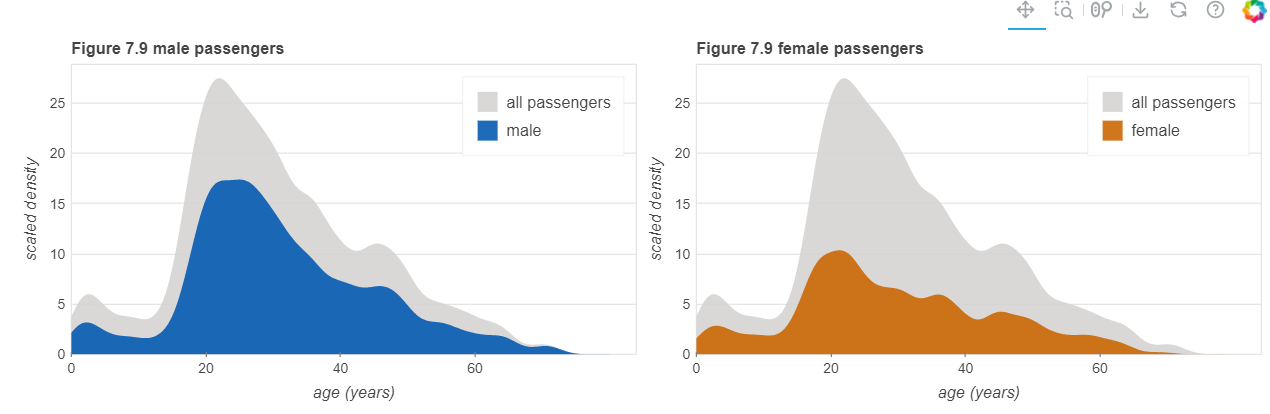
* Che khuất có nhiều người trẻ hơn người già trong tập dữ liệu.
  + *số lượng người có sức khỏe tốt tuyệt đối* vẫn giảm khi tổng số người ở một độ tuổi nhất định giảm mặc dù *tỷ lệ* người báo cáo có sức khỏe good là như nhau.

## 6.4 Trực quan hóa các tỷ lệ riêng biệt như một phần của tổng thể

### A. Multiple distribution density plots

**Cách vẽ: Tương tự chương 7**

*Plotting:*

**

# CHƯƠNG 7: TRỰC QUAN HÓA CÁC TỶ LỆ LỒNG NHAU

## 7.2 Các ô khảm và sơ đồ cây

## A. Mosaic plots

* Phân tích các cây cầu ở Pittsburgh theo vật liệu xây dựng (thép, gỗ, sắt) và theo thời kỳ xây dựng (thủ công, mới nổi, trưởng thành, hiện đại), được thể hiện dưới dạng sơ đồ cây. Diện tích của mỗi hình chữ nhật tỉ lệ thuận với số lượng cầu thuộc loại đó.
* Bokeh không hỗ trợ Mosasic plots

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*import pandas as pd*

*from statsmodels.graphics.mosaicplot import mosaic*

*from bokeh.plotting import figure, ColumnDataSource, show*

*from bokeh.transform import factor\_cmap*

*df = pd.DataFrame({*

*'size' : ['small', 'medium', 'medium', 'large', 'small', 'large', 'small', 'medium'],*

*'length' : ['long', 'short', 'long', 'short', 'long', 'long', 'short', 'short']*

*})*

*\_, rects\_dict = mosaic(df, ['size', 'length'], gap=0, ax=None)*

*rects = rects\_dict.values()*

*cats = rects\_dict.keys()*

*Plotting:*

*source = ColumnDataSource(data=dict(*

*x = [r[0]+r[2]/2 for r in rects], # bokeh wants x center, not corner*

*y = [r[1]+r[3]/2 for r in rects], # bokeh wants y center, not corner*

*w = [r[2] for r in rects],*

*h = [r[3] for r in rects],*

*size = [c[0] for c in cats ],*

*len = [c[1] for c in cats ],*

*))*

*fill\_cmap = factor\_cmap('size', palette="Pastel1\_3", factors=['small', 'medium', 'large'])*

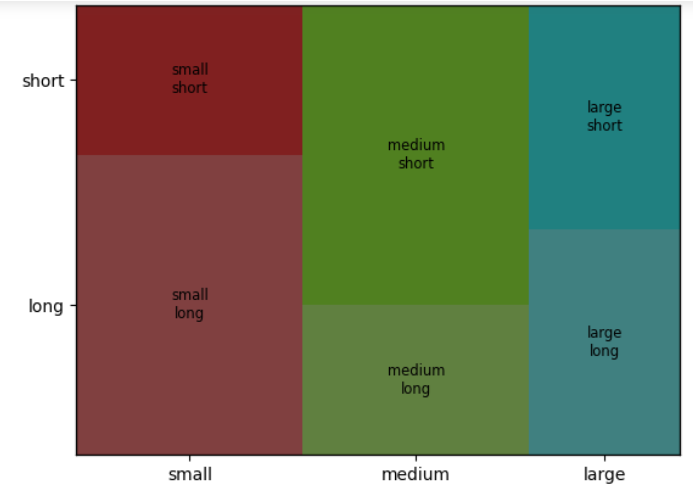
*p = figure(x\_range=(0,1), y\_range=(0,1), x\_axis\_location=None, y\_axis\_location=None,*

*tools="", toolbar\_location=None, tooltips="@size @len")*

*p.rect(x='x', y='y', width='w', height='h', line\_color="white", source=source,*

*fill\_color=fill\_cmap)*

*show(p)*

**

**Lưu ý:**

* Thể hiện mối quan hệ liên quan của các danh mục chồng chéo nhau.
* Giả định rằng tất cả các tỷ lệ được hiển thị có thể được xác định thông qua sự kết hợp của hai hoặc nhiều biến phân loại trực giao.

### B. Treemap

* Thể hiện sự giảm giá trong mỗi thành phố theo từng khu vực.
* Xây dựng hàm treemap()

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*import pandas as pd*

*from squarify import normalize\_sizes, squarify*

*from bokeh.plotting import figure, show*

*from bokeh.sampledata.sample\_superstore import data*

*from bokeh.transform import factor\_cmap*

*data = data[["City", "Region", "Sales"]]*

*regions = ("West", "Central", "South", "East")*

*sales\_by\_city = data.groupby(["Region", "City"]).sum("Sales")*

*sales\_by\_city = sales\_by\_city.sort\_values(by="Sales").reset\_index()*

*sales\_by\_region = sales\_by\_city.groupby("Region").sum("Sales").sort\_values(by="Sales")*

*Plotting:*

*def treemap(df, col, x, y, dx, dy, \*, N=100):*

*sub\_df = df.nlargest(N, col)*

*normed = normalize\_sizes(sub\_df[col], dx, dy)*

*blocks = squarify(normed, x, y, dx, dy)*

*blocks\_df = pd.DataFrame.from\_dict(blocks).set\_index(sub\_df.index)*

*return sub\_df.join(blocks\_df, how='left').reset\_index()*

*x, y, w, h = 0, 0, 800, 450*

*blocks\_by\_region = treemap(sales\_by\_region, "Sales", x, y, w, h)*

*dfs = []*

*for index, (Region, Sales, x, y, dx, dy) in blocks\_by\_region.iterrows():*

*df = sales\_by\_city[sales\_by\_city.Region==Region]*

*dfs.append(treemap(df, "Sales", x, y, dx, dy, N=10))*

*blocks = pd.concat(dfs)*

*p = figure(width=w, height=h, tooltips="@City", toolbar\_location=None,*

*x\_axis\_location=None, y\_axis\_location=None)*

*p.x\_range.range\_padding = p.y\_range.range\_padding = 0*

*p.grid.grid\_line\_color = None*

*p.block('x', 'y', 'dx', 'dy', source=blocks, line\_width=1, line\_color="white",*

*fill\_alpha=0.8, fill\_color=factor\_cmap("Region", "MediumContrast4", regions))*

*p.text('x', 'y', x\_offset=2, text="Region", source=blocks\_by\_region,*

*text\_font\_size="18pt", text\_color="white")*

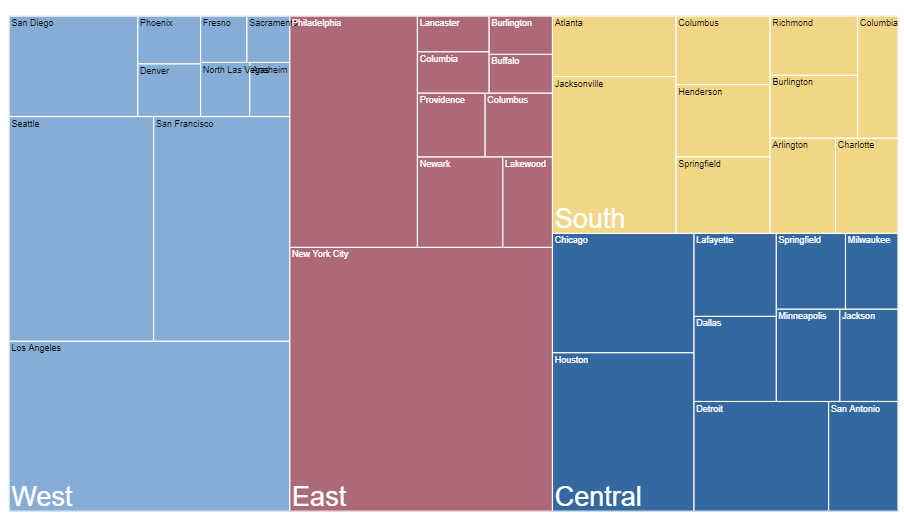
*blocks["ytop"] = blocks.y + blocks.dy*

*p.text('x', 'ytop', x\_offset=2, y\_offset=2, text="City", source=blocks,*

*text\_font\_size="6pt", text\_baseline="top",*

*text\_color=factor\_cmap("Region", ("black", "white", "black", "white"), regions))*

*show(p)*

**

**Lưu ý:**

* Không yêu cầu mối quan hệ của các biến phân loại
* Hạn chế tương tự như các thanh xếp chồng lên nhau.
* Việc so sánh trực tiếp giữa các điều kiện có thể khó khăn vì các hình chữ nhật khác nhau không nhất thiết phải chia sẻ đường cơ sở để có thể so sánh trực quan.

## 7.3 Nested pies

* Phân tích các cây cầu ở Pittsburgh theo vật liệu xây dựng (thép, gỗ, sắt) và theo thời kỳ xây dựng (thủ công, mới nổi, trưởng thành, hiện đại). Các con số biểu thị số lượng cầu trong mỗi loại.
* Phương thức wedge() được sử dụng

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*data = pd.read\_csv('bridges.csv')*

*data = data[['ERECTED', 'MATERIAL']]*

*df = data.dropna(subset=['MATERIAL'])*

*df['COMBINED'] = df['ERECTED'] + "-" + df['MATERIAL']*

*counts = df['COMBINED'].value\_counts().reset\_index(name='count')*

*counts['angle'] = counts['count']/counts['count'].sum() \* 2\*pi*

*# Phân loại màu sắc*

*counts['color'] = Category20c[len(counts)]*

*Plotting:*

*p = figure(height=350, title="Pie Chart of ERECTED-MATERIAL pairs", toolbar\_location=None,*

*tools="hover", tooltips="@index: @count", x\_range=(-0.5, 1.0))*

*p.wedge(x=0, y=1, radius=0.4,*

*start\_angle=cumsum('angle', include\_zero=True), end\_angle=cumsum('angle'),*

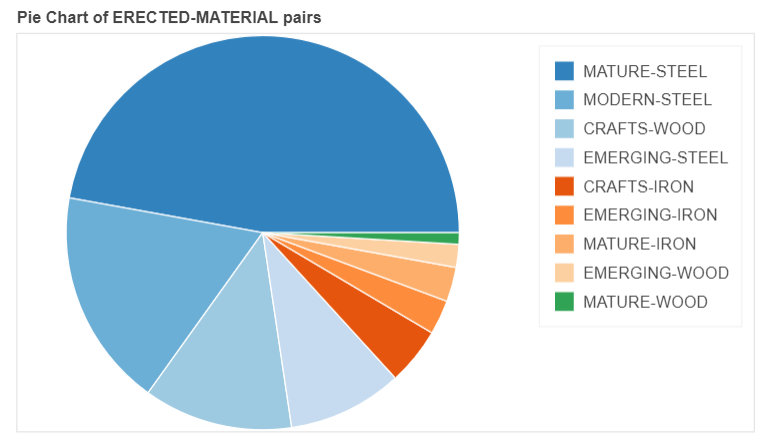
*line\_color='white', fill\_color='color', legend\_field='index', source=counts)*

*p.axis.axis\_label=None*

*p.axis.visible=False*

*p.grid.grid\_line\_color = None*

*show(p)*

**

**Lưu ý:**

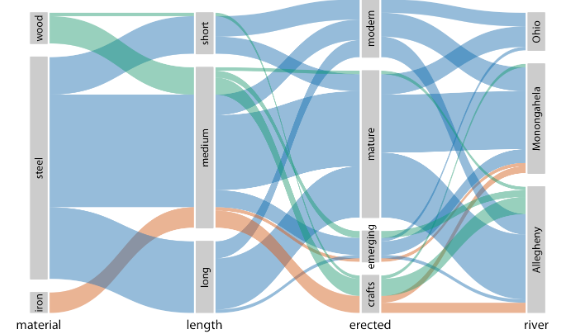
* Chia biểu đồ thành nhiều phần thể hiện tỷ lệ theo một biến

## 7.4 Parallel sets

* Phân tích các cây cầu ở Pittsburgh theo vật liệu xây dựng, chiều dài, thời kỳ xây dựng và con sông mà chúng bắc qua, được thể hiện dưới dạng biểu đồ song song. Màu sắc của các dải làm nổi bật vật liệu xây dựng của các cây cầu khác nhau
* Bokeh không hỗ trợ Parallel sets

**Cách vẽ:**

*Plotting:*

**

**Lưu ý:**

* Các tập hợp xác định màu sắc sẽ xuất hiện từ trái sang phải.
* Thay đổi thứ tự của các tập hợp sao cho số lượng dải đan chéo được giảm thiểu.

# CHƯƠNG 8:  TRỰC QUAN HÓA MỐI LIÊN HỆ GIỮA HAI HOẶC NHIỀU BIẾN ĐỊNH LƯỢNG

## 8.1 Scatter plots

### A. Bubble chart

* Các đồ thị trong phần này thể hiện chiều dài đầu so với khối lượng cơ thể của 123 con giẻ cùi xanh. Giới tính của loài chim được biểu thị bằng màu sắc và kích thước hộp sọ của loài chim được biểu thị bằng kích thước vòng tròn.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/blue\_jays.csv"

df **=** pd**.**read\_csv(file)

*# add a "skull size" column to shrink circle size in the plot.*

df["skull\_size"] **=** [1.5 **\*** (2 **\*\*** (size **/** 10)) **for** size **in** df["Skull"]]

df**.**head()

*Plotting:*

**from** bokeh.plotting **import** figure, show

**def** plot\_scatter(df, fill\_color**=None**, title**=None**, yaxis**=False**):

"""

Create a scatter plot using Bokeh.

Parameters:

df (pandas.DataFrame): DataFrame containing the data to be plotted.

fill\_color (str, optional): Color to fill the scatter points.

title (str, optional): Title text for the plot.

yaxis (bool, optional): Whether to display the y-axis or not.

Returns:

bokeh.plotting.figure.Figure: Scatter plot figure.

"""

p **=** figure(

title**=**title,

height**=**400,

width**=**400,

y\_range**=**(50, 62),

x\_axis\_label**=**"body mass (g)",

y\_axis\_label**=**"head length (mm)",

toolbar\_location**=None**,

)

p**.**scatter(

x**=**"Mass", *# column name for x-axis of plot*

y**=**"Head", *# column name for y-axis of plot*

size**=**"skull\_size", *# column name to use for circle size*

source**=**df, *# data source*

fill\_color**=**fill\_color,

line\_color**=**"white",

)

*# Customize plot axis and title*

p**.**yaxis**.**ticker **=** [52, 54, 56, 58, 60]

p**.**yaxis**.**visible **=** yaxis

p**.**title**.**text\_font\_size **=** "12px"

p**.**title**.**align **=** "center"

p**.**title**.**background\_fill\_color **=** "darkgrey"

p**.**title**.**background\_fill\_alpha **=** 0.4

**return** p

**from** bokeh.layouts **import** gridplot

*# create separate dataframes for the bird sexes*

male **=** df[df["Sex"] **==** 1]

female **=** df[df["Sex"] **==** 0]

*# plot each scatter plot using the appropriate dataframe*

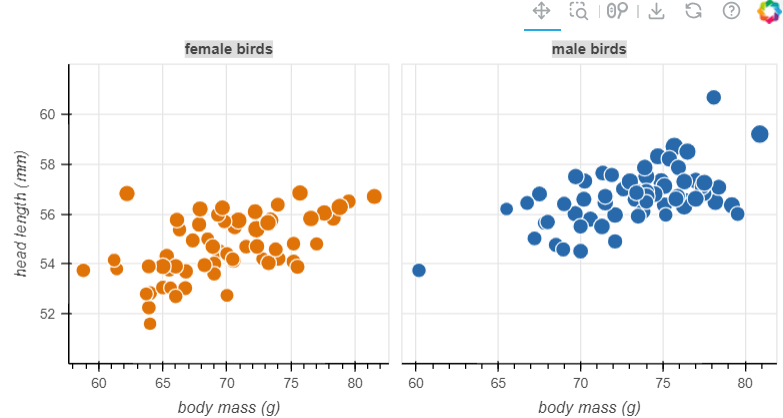
f **=** plot\_scatter(female, "#DF7307", "female birds", yaxis**=True**)

m **=** plot\_scatter(male, "#2769AB", "male birds")

*# display both plots in a grd layout*

layout **=** gridplot([[f, m]], sizing\_mode**=**"scale\_both")

show(layout)

**

**Lưu ý:**

* Nhược điểm là thể hiện cùng loại biến, biến định lượng, với hai loại tỷ lệ, vị trí và kích thước khác nhau. Điều này gây khó khăn cho việc xác định trực quan sức mạnh của mối liên hệ giữa các biến khác nhau.

### B. All-against-all scatter plot matrix

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

* Dữ liệu tương tự từ phần phụ trước đó được sử dụng ở đây. Trong tiểu mục này, mỗi biểu đồ riêng lẻ hiển thị hai chiều dữ liệu được vẽ đối lập với nhau.

*Plotting:*

**from** itertools **import** product

**from** bokeh.io **import** show

**from** bokeh.layouts **import** gridplot

**from** bokeh.models **import** (

BasicTicker,

Scatter,

ColumnDataSource,

DataRange1d,

Grid,

LinearAxis,

Plot,

SaveTool,

)

**from** bokeh.transform **import** factor\_cmap

sex **=** df**.**KnownSex**.**unique()

VARIABLES **=** ("Head", "Mass", "Skull")

LABELS **=** ("head length (mm)", "body mass (g)", "skull size (mm)")

N **=** len(VARIABLES)

source **=** ColumnDataSource(data**=**df)

xdrs **=** [DataRange1d(bounds**=None**) **for** \_ **in** range(N)]

ydrs **=** [DataRange1d(bounds**=None**) **for** \_ **in** range(N)]

plots **=** []

**for** i, (y, x) **in** enumerate(product(VARIABLES, VARIABLES)):

*# create Plot object*

p **=** Plot(

x\_range**=**xdrs[i **%** N],

y\_range**=**ydrs[i **//** N],

background\_fill\_color**=**"white",

border\_fill\_color**=**"white",

width**=**200,

height**=**200,

)

**if** i **%** N **==** 0: *# for first column:*

*# set up y-axis label, label orientation, and grid line ticker.*

p**.**min\_border\_left **=** p**.**min\_border **+** 4

p**.**width **+=** 40

yaxis **=** LinearAxis(axis\_label**=**LABELS[VARIABLES**.**index(y)])

yaxis**.**major\_label\_orientation **=** "vertical"

p**.**add\_layout(yaxis, "left")

yticker **=** yaxis**.**ticker

**else**:

yticker **=** BasicTicker()

p**.**add\_layout(Grid(dimension**=**1, ticker**=**yticker))

**if** i **>=** N **\*** (N **-** 1): *# for last row:*

*# set up x-axis label, label orientation, and grid line ticker.*

p**.**min\_border\_bottom **=** p**.**min\_border **+** 40

p**.**height **+=** 40

xaxis **=** LinearAxis(axis\_label**=**LABELS[VARIABLES**.**index(x)])

p**.**add\_layout(xaxis, "below")

xticker **=** xaxis**.**ticker

**else**:

xticker **=** BasicTicker()

p**.**add\_layout(Grid(dimension**=**0, ticker**=**xticker))

*# create scatter glyph*

scatter **=** Scatter(

x**=**x,

y**=**y,

size**=**6,

line\_color**=**"white",

fill\_color**=**factor\_cmap("KnownSex", ["#2769AB", "#DF7307"], sex),

)

*# add scatter glyph to plot*

r **=** p**.**add\_glyph(source, scatter)

p**.**x\_range**.**renderers**.**append(r)

p**.**y\_range**.**renderers**.**append(r)

*# remove minor ticks in x and y axis of each plot*

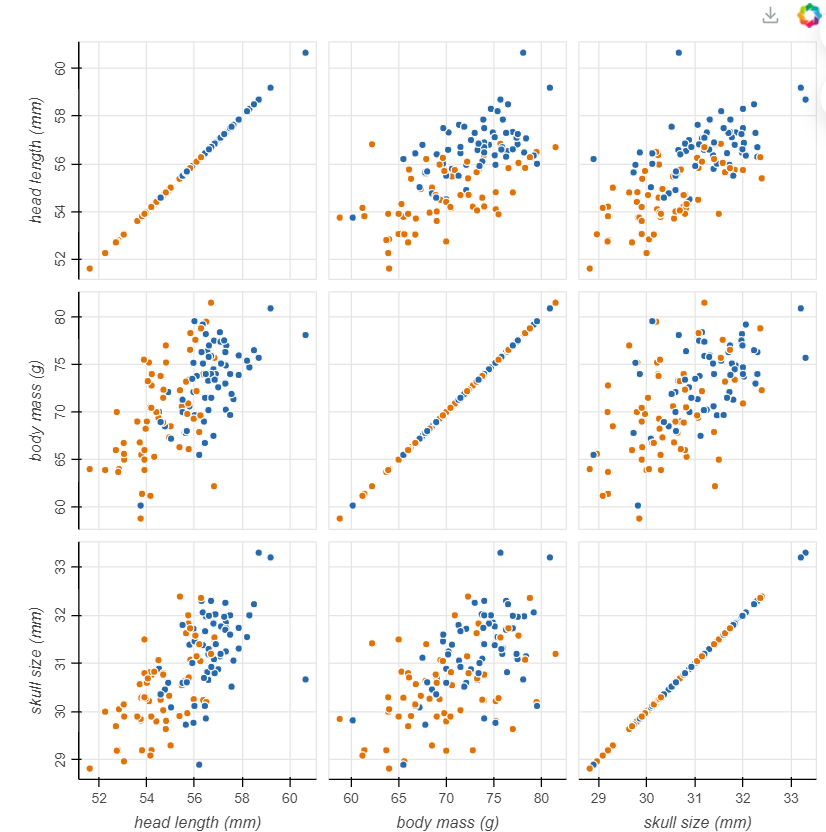
p**.**yaxis**.**minor\_tick\_out **=** 0

p**.**xaxis**.**minor\_tick\_out **=** 0

p**.**add\_tools(SaveTool())

plots**.**append(p)

show(gridplot(plots, ncols**=**N))

**

**Lưu ý:**

* Điều này không đúng với mối quan hệ giữa chiều dài đầu và khối lượng cơ thể.
* Có sự phân biệt rõ ràng về giới tính.
* Chim đực có xu hướng có mỏ dài hơn chim cái, các yếu tố khác đều bằng nhau.

## 8.2 Correlograms

* Các sơ đồ trong tiểu mục này thể hiện mối tương quan về hàm lượng khoáng chất của 214 mẫu mảnh thủy tinh thu được trong quá trình khám nghiệm pháp y.
* Bộ dữ liệu chứa bảy biến số đo lượng magie (Mg), canxi (Ca), sắt (Fe), kali (K), natri (Na), nhôm (Al) và bari (Ba) được tìm thấy trong mỗi mảnh thủy tinh.
* Độ lớn của mỗi mối tương quan cũng được mã hóa theo kích thước của các vòng tròn màu.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/forensic\_glass.csv"

df **=** pd**.**read\_csv(file)

pairs **=** [

"Mg-Ba",

"Mg-Al",

"Mg-Na",

"Mg-K",

"Mg-Fe",

"Mg-Ca",

"Ca-Ba",

"Ca-Al",

"Ca-Na",

"Ca-K",

"Ca-Fe",

"Fe-Ba",

"Fe-Al",

"Fe-Na",

"Fe-K",

"K-Ba",

"K-Al",

"K-Na",

"Na-Ba",

"Na-Al",

"Al-Ba",

]

pair\_split **=** [pair**.**split("-") **for** pair **in** pairs]

correlations **=** []

**for** pair **in** pair\_split:

matrix **=** np**.**corrcoef(df[f"{pair[0]}"], df[f"{pair[1]}"])

correlation **=** matrix[0, 1]

correlations**.**append(correlation)

new\_df **=** pd**.**DataFrame(

{

"oxide\_1": [x[0] **for** x **in** pair\_split],

"oxide\_2": [x[1] **for** x **in** pair\_split],

"correlation": correlations,

"dot\_size": [abs(corr) **\*** 120 **for** corr **in** correlations],

}

)

new\_df**.**head()

*Plotting:*

**from** bokeh.transform **import** transform

**from** bokeh.models **import** ColorBar, LinearColorMapper, FixedTicker

x\_range **=** new\_df["oxide\_1"]**.**unique()

y\_range **=** list(reversed(new\_df["oxide\_2"]**.**unique()))

p **=** figure(

title**=**"Figure 12.7",

sizing\_mode**=**"scale\_both",

x\_axis\_location**=**"above",

toolbar\_location**=None**,

x\_range**=**x\_range,

y\_range**=**y\_range,

)

*# add color mapper to plot*

colors **=** [

"#8B4513",

"#D8AF85",

"#CD853F",

"#E2CAB7",

"#DEB887",

"#A4D2D2",

"#8ABDBD",

"#ADD8E6",

"#009999",

"#188A8A",

]

mapper **=** LinearColorMapper(palette**=**colors, low**=-**0.5, high**=**0.5)

p**.**scatter(

x**=**"oxide\_1",

y**=**"oxide\_2",

size**=**"dot\_size",

source**=**new\_df,

fill\_color**=**transform("correlation", mapper),

line\_color**=None**,

)

*# create color bar object*

color\_bar **=** ColorBar(

color\_mapper**=**mapper,

location**=**(200, 0),

ticker**=**FixedTicker(ticks**=**[**-**0.5, 0.0, 0.5]),

title**=**"correlation",

title\_text\_align**=**"center",

title\_text\_font\_style**=**"normal",

major\_tick\_line\_color**=None**,

width**=**150,

height**=**20,

)

p**.**add\_layout(color\_bar, "below")

*# customize plot*

p**.**xaxis**.**major\_tick\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_out **=** 0

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**major\_tick\_out **=** 0

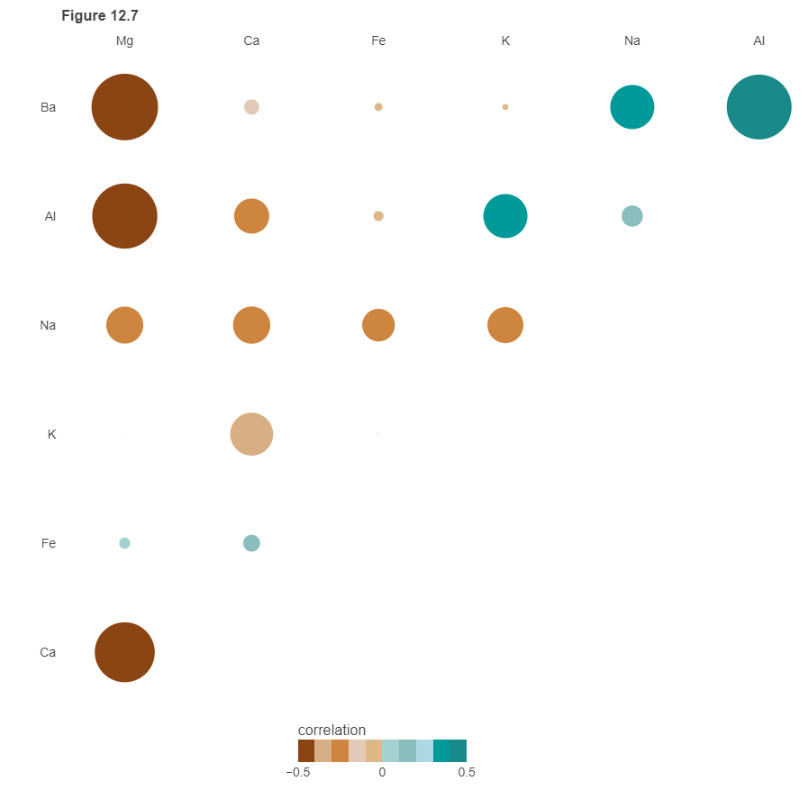
p**.**yaxis**.**major\_tick\_line\_color **=** **None**

p**.**yaxis**.**axis\_line\_color **=** **None**

p**.**grid**.**grid\_line\_color **=** **None**

p**.**outline\_line\_color **=** **None**

show(p)



**Lưu ý:**

* Nhược điểm: Chúng khá trừu tượng.
* Mặc dù chúng hiển thị cho chúng ta các mẫu quan trọng trong dữ liệu nhưng chúng cũng ẩn các điểm dữ liệu cơ bản và có thể khiến chúng ta đưa ra kết luận sai

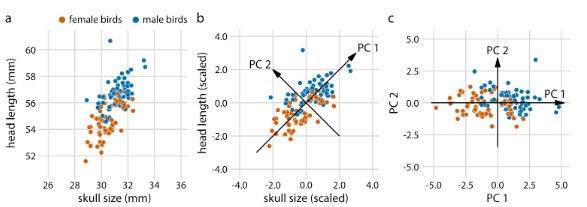
## 8.3 Dimension reduction

* Phân tích các thành phần chính (PC) theo hai chiều:
  + (a) dữ liệu gốc.
  + (b) bước đầu của PCA.
  + (c) chiếu dữ liệu vào tọa độ mới.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*Plotting:*

**

**Lưu ý:**

* Khi thực hiện PCA, chúng tôi thường quan tâm đến hai thông tin:
  + (i) thành phần của PC v
  + (ii) vị trí của các điểm dữ liệu riêng lẻ trong không gian thành phần chính.

### 8.4 Paired data

* Đồ thị trong tiểu mục này thể hiện lượng phát thải carbon dioxide trên mỗi người trong năm 2000 và 2010, của 10 quốc gia có sự khác biệt lớn nhất giữa hai năm này.
* Phương thức p.scatter() được sử dụng

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

file **=** "../data/csv\_files/CO2\_emissions.csv"

df **=** pd**.**read\_csv(file)

countries **=** [

"Trinidad and Tobago",

"Qatar",

"United Arab Emirates",

"Oman",

"Bahrain",

"Singapore",

"Netherlands Antilles",

"Kazakhstan",

"Equatorial Guinea",

"Kuwait",

]

*# create new dataframe for only the relevant countries*

new\_df **=** df[df["country"]**.**isin(countries)]**.**reset\_index(drop**=True**)

years **=** (new\_df["year"] **==** 2000.0) **|** (new\_df["year"] **==** 2010.0)

new\_df **=** new\_df[years]**.**reset\_index(drop**=True**)

new\_df["year"] **=** new\_df**.**year**.**astype(int)

new\_df["year"] **=** new\_df**.**year**.**astype(str)

*# create new columns for different years*

a **=** new\_df[new\_df["year"] **==** "2000"]

b **=** new\_df[new\_df["year"] **==** "2010"]

plot\_df **=** a**.**merge(b, on**=**"country")

plot\_df**.**head()

*Plotting:*

**from** bokeh.models **import** LabelSet, ColumnDataSource

source **=** ColumnDataSource(plot\_df)

p **=** figure(

y\_range**=**(0, 60),

x\_range**=**("2000", "2010"),

sizing\_mode**=**"scale\_both",

toolbar\_location**=None**,

x\_axis\_location**=**"above",

y\_axis\_label**=**"CO2 emissions (tons / person)",

)

*# add scatter plot to figure*

p**.**scatter(x**=**"year\_x", y**=**"emissions\_x", source**=**source, size**=**7)

p**.**scatter(

x**=**"year\_y",

y**=**"emissions\_y",

source**=**source,

size**=**7,

)

*# add individual line segments for each country*

p**.**segment(

x0**=**"year\_x",

y0**=**"emissions\_x",

x1**=**"year\_y",

y1**=**"emissions\_y",

source**=**source,

color**=**"black",

)

*# create country label*

label **=** LabelSet(

x**=**"year\_y",

y**=**"emissions\_y",

text**=**"country",

source**=**source,

text\_font\_size**=**"11px",

x\_offset**=**8,

y\_offset**=-**7,

)

p**.**add\_layout(label)

*# customize plot*

p**.**xaxis**.**major\_tick\_line\_color **=** **None**

p**.**xaxis**.**major\_tick\_out **=** 0

p**.**xaxis**.**axis\_line\_color **=** **None**

p**.**yaxis**.**minor\_tick\_out **=** 0

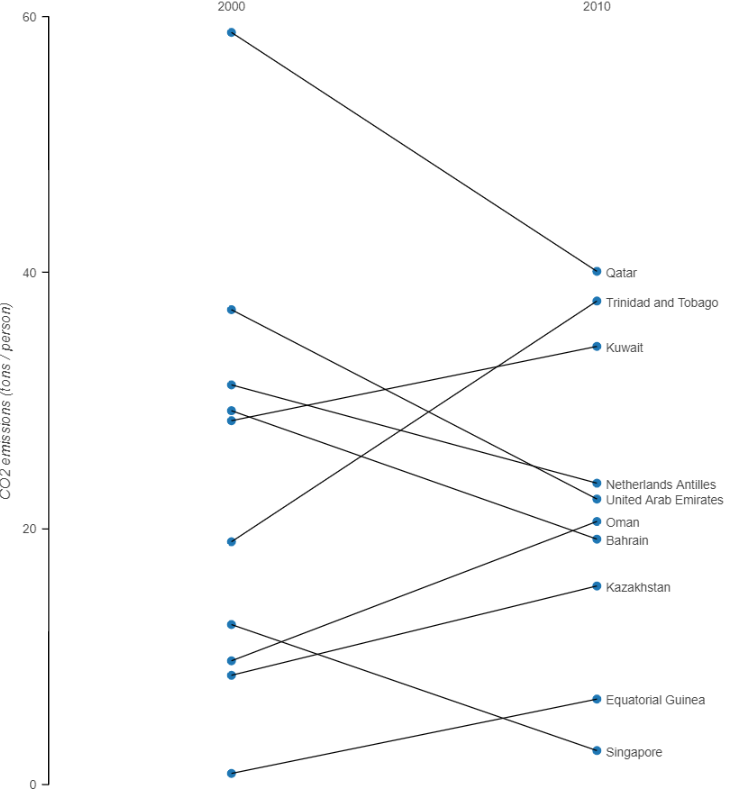
p**.**yaxis**.**major\_tick\_in **=** 0

p**.**yaxis**.**ticker **=** [0, 20, 40, 60]

p**.**grid**.**grid\_line\_color **=** **None**

p**.**outline\_line\_color **=** **None**

show(p)



**Lưu ý:**

* Đồ thị dốc có một lợi thế quan trọng so với biểu đồ phân tán: Chúng có thể được sử dụng để so sánh nhiều hơn hai phép đo cùng một lúc.

# CHƯƠNG 9: TRỰC QUAN HÓA CHUỔI THỜI GIAN VÀ CÁC HÀM KHÁC CỦA MỘT BIẾN

## 9.1 Chuỗi thời gian riêng lẻ

### A. Dot graph

* Số lần gửi hàng tháng tới máy chủ in sẵn bioRxiv, kể từ khi thành lập vào tháng 11 năm 2014 cho đến tháng 4 năm 2018. Mỗi dấu chấm biểu thị số lần gửi trong một tháng. Đã có sự gia tăng ổn định về số lượng đệ trình trong suốt thời gian 4,5 năm.
* Phương thức p.cricle() được sử dụng để thể hiện dấu chấm

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*data = pd.read\_csv('data/csv\_files/preprint\_growth.csv')*

*data['date'] = pd.to\_datetime(data['date'])*

*filtered\_data = data[data['date'] >= '2013-10-01']*

*monthly\_data = filtered\_data.groupby(filtered\_data['date'].dt.to\_period('M')).sum()*

*x = monthly\_data.index.to\_timestamp()*

*y = monthly\_data['count']*

*Plotting:*

*p = figure(height=400, width=600, x\_axis\_type='datetime',*

*title='Monthly submissions to the preprint server bioRxiv')*

*p.circle(x, y, size=10, alpha=0.5)*

*p.xaxis.axis\_label = 'Year'*

*p.yaxis.axis\_label = 'preprints / month'*

*p.add\_tools(HoverTool(*

*tooltips=[*

*('Year', '@x{%F}'),*

*('preprints / month', '@y'),*

*],*

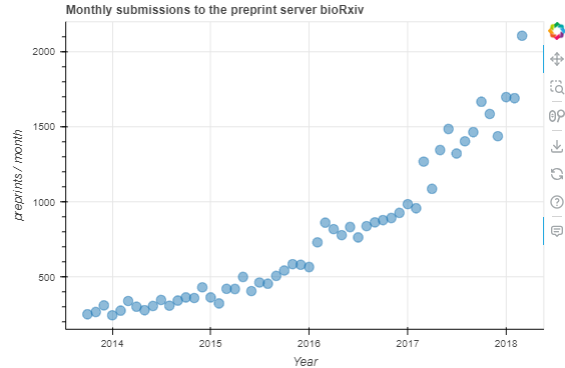
*formatters={*

*'@x': 'datetime',*

*},*

*))*

*show(p)*

**

**Lưu ý:**

* Tuy nhiên, có một sự khác biệt quan trọng giữa Hình [13.1](https://clauswilke.com/dataviz/time-series.html#fig:biorxiv-dots) và các biểu đồ phân tán được thảo luận trong Chương [12](https://clauswilke.com/dataviz/visualizing-associations.html#visualizing-associations) . Trong Hình [13.1](https://clauswilke.com/dataviz/time-series.html#fig:biorxiv-dots) , các chấm cách đều nhau dọc theo trục x và có một thứ tự xác định giữa chúng. Mỗi dấu chấm có chính xác một điểm lân cận bên trái và một điểm lân cận bên phải (ngoại trừ điểm ngoài cùng bên trái và ngoài cùng bên phải chỉ có một điểm lân cận

### B. Line graph

### Các bản gửi hàng tháng tới máy chủ in sẵn bioRxiv, được hiển thị dưới dạng biểu đồ đường không có dấu chấm.

### Phương thức p.line() được sử dụng để thể hiện các đường.

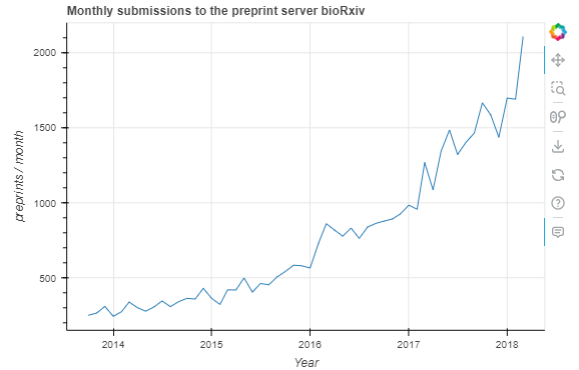
**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*Plotting:*

*#p.circle(x, y, size=10, alpha=0.5)*

*p.line(x, y)*

**

**Lưu ý:**

* Việc bỏ qua các dấu chấm nhấn mạnh xu hướng thời gian tổng thể trong khi không nhấn mạnh đến các quan sát riêng lẻ tại các thời điểm cụ thể.
* Nó đặc biệt hữu ích khi các điểm thời gian cách nhau rất dày đặc.

## 9.2 Multiple time series and dose–response curves

* Các bản gửi hàng tháng tới ba máy chủ in sẵn bao gồm nghiên cứu y sinh. Bằng cách gắn nhãn trực tiếp cho các dòng thay vì cung cấp chú giải, chúng tôi đã giảm tải nhận thức cần thiết để đọc hình.
* Phương thức p.line() được sử dụng.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*filtered\_data = data[(data['date'] >= '2013-10-01') & (data['date'] <= '2017-1-31')]*

*bioRxiv\_data = filtered\_data[filtered\_data['archive'] == 'bioRxiv']*

*arXiv\_qbio\_data = filtered\_data[filtered\_data['archive'] == 'arXiv q-bio']*

*PeerJ\_Preprints\_data = filtered\_data[filtered\_data['archive'] == 'PeerJ Preprints']*

*Plotting:*

*p = figure(x\_axis\_type='datetime', title='Monthly submissions to three preprint servers covering biomedical research', height=350, width=800)*

*p.xaxis.axis\_label = 'Year'*

*p.yaxis.axis\_label = 'preprints / month'*

*bioRxiv\_source = ColumnDataSource(bioRxiv\_data)*

*arXiv\_qbio\_source = ColumnDataSource(arXiv\_qbio\_data)*

*PeerJ\_Preprints\_source = ColumnDataSource(PeerJ\_Preprints\_data)*

*# Vẽ các dòng biểu đồ*

*line\_bioRxiv = p.line('date', 'count', source=bioRxiv\_source, color='green')*

*line\_arXiv\_qbio = p.line('date', 'count', source=arXiv\_qbio\_source, color='red')*

*line\_PeerJ\_Preprints = p.line('date', 'count', source=PeerJ\_Preprints\_source, color='blue')*

*# Thêm nhãn*

*bioRxiv\_label = Label(x=bioRxiv\_data['date'].iloc[-1], y=bioRxiv\_data['count'].iloc[-1], x\_offset=5, text='bioRxiv', text\_color='green')*

*arXiv\_qbio\_label = Label(x=arXiv\_qbio\_data['date'].iloc[-1], y=arXiv\_qbio\_data['count'].iloc[-1], x\_offset=5, text='arXiv q-bio', text\_color='red')*

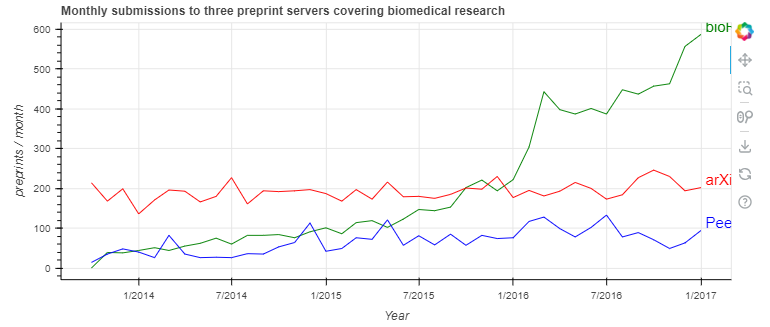
*PeerJ\_Preprints\_label = Label(x=PeerJ\_Preprints\_data['date'].iloc[-1], y=PeerJ\_Preprints\_data['count'].iloc[-1], x\_offset=5, text='PeerJ Preprints', text\_color='blue')*

*p.add\_layout(bioRxiv\_label)*

*p.add\_layout(arXiv\_qbio\_label)*

*p.add\_layout(PeerJ\_Preprints\_label)*

*show(p)*

**

**Lưu ý:**

* Việc loại bỏ chú giải sẽ loại bỏ nhu cầu về các điểm có hình dạng khác nhau.
* Vì vậy, chúng ta có thể sắp xếp hợp lý hình hơn nữa bằng cách loại bỏ các dấu chấm

## 9.3 Time series of two or more response variables

* Thay đổi giá nhà (a) và tỷ lệ thất nghiệp (b) trong 12 tháng theo thời gian, từ tháng 1 năm 2001 đến tháng 12 năm 2017

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*df = pd.read\_csv('data/csv\_files/house\_prices.csv')*

*df['date'] = pd.to\_datetime(df['date'])*

*df = df[(df['state'] == 'California') & (df['date'].dt.year > 2000)]*

*Plotting:*

*df\_source = ColumnDataSource(df)*

*p = figure(x\_axis\_type='datetime', title='12-month change in house prices', height=350, width=800)*

*p.xaxis.axis\_label = 'Year'*

*p.yaxis.axis\_label = '12-month change\\nin house prices'*

*p.line('date', 'house\_price\_perc', source=df\_source, color='#0072b2', line\_width=2)*

*p.y\_range.start = -0.3*

*p.y\_range.end = 0.32*

*p.yaxis.ticker = [-0.3, -0.15, 0, 0.15, 0.3]*

*# output\_file("line\_chart.html") # Uncomment để xuất ra tệp HTML nếu muốn*

*show(p)*

*# Tính toán 'unemploy\_perc' chia cho 100 ngay trong ColumnDataSource*

*df\_source.data['unemploy\_perc\_scaled'] = df['unemploy\_perc']/100*

*# Khởi tạo một đối tượng biểu đồ figure*

*p = figure(x\_axis\_type='datetime', title='Unemployment Rate', height=350, width=800)*

*p.xaxis.axis\_label = 'Year'*

*p.yaxis.axis\_label = 'Unemployment Rate'*

*# Vẽ đường biểu đồ*

*p.line(x='date', y='unemploy\_perc\_scaled', source=df\_source, color='#0072b2', line\_width=2)*

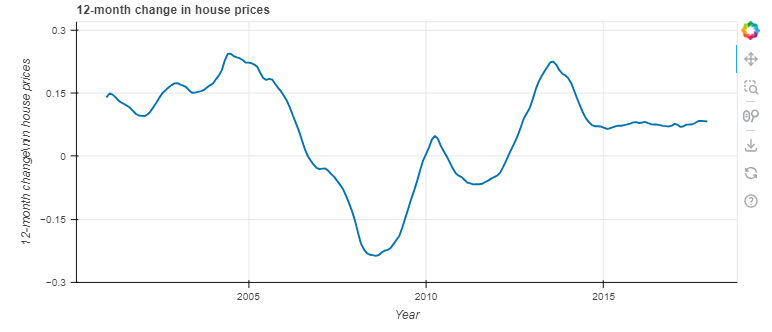
*# Cài đặt giới hạn và giá trị cố định cho trục y*

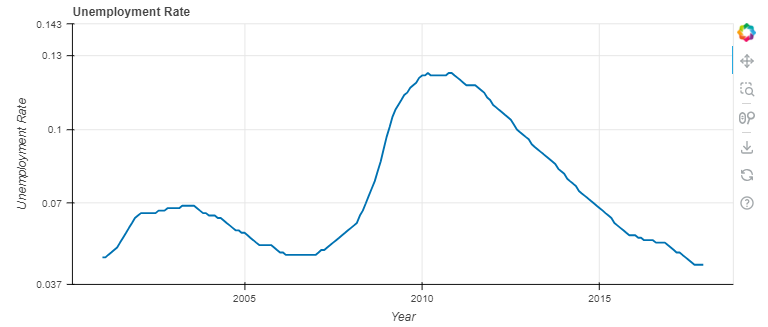
*p.y\_range.start = 0.037*

*p.y\_range.end = 0.143*

*p.yaxis.ticker = [0.037, 0.07, 0.1, 0.13, 0.143]*

*show(p)*

**



# CHƯƠNG 10: HÌNH DUNG TREND

## 10.1 Smoothing

* Giá trị đóng cửa hàng ngày của Chỉ số Trung bình Công nghiệp Dow Jones trong năm 2009, được hiển thị cùng với các đường trung bình động 20 ngày, 50 ngày và 100 ngày của chúng. Các đường trung bình động được vẽ ở giữa cửa sổ thời gian động.
* So sánh mức LOESS phù hợp với đường trung bình động 100 ngày của dữ liệu Dow Jones.
* Phương thức p.line() được sử dụng để thể hiện xu hướng

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*from statsmodels.nonparametric.smoothers\_lowess import lowess*

*loess\_smoothed = lowess(data\_2009['close'], range(len(data\_2009['close'])), frac=0.1)*

*data\_2009['loess\_smoothed'] = loess\_smoothed[:, 1]*

*data = pd.read\_csv('./data/csv\_files/dow\_jones\_industrial.csv')*

*# Chuyển cột 'date' thành kiểu datetime*

*data['date'] = pd.to\_datetime(data['date'])*

*# Chọn dữ liệu cho năm 2009*

*data\_2009 = data[(data['date'] >= '2009-01-01') & (data['date'] < '2010-01-01')]*

*# Tính trung bình trượt (moving average) cho cột 'close' trong cửa sổ 20 ngày*

*window\_size\_20 = 20*

*window\_size\_50 = 50*

*window\_size\_100 = 100*

*data\_2009['moving\_average\_20'] = data\_2009['close'].rolling(window=window\_size\_20, center=True).mean()*

*data\_2009['moving\_average\_50'] = data\_2009['close'].rolling(window=window\_size\_50, center=True).mean()*

*data\_2009['moving\_average\_100'] = data\_2009['close'].rolling(window=window\_size\_100, center=True).mean()*

*Plotting:*

*# Tạo figure*

*p = figure(title='Figure 14.3: Comparison of LOESS fit to 100-day Moving Average', x\_axis\_label='Date', y\_axis\_label='Closing Price', x\_axis\_type='datetime')*

*# Biểu diễn dữ liệu gốc*

*p.line(x='date', y='close', source=ColumnDataSource(data\_2009), line\_color=Category10[3][0], legend\_label='Original Data')*

*# Biểu diễn trung bình trượt 100 ngày*

*p.line(x='date', y='moving\_average\_100', source=ColumnDataSource(data\_2009), line\_color=Category10[3][1], legend\_label='100-day Moving Average')*

*# Biểu diễn đường cong LOESS*

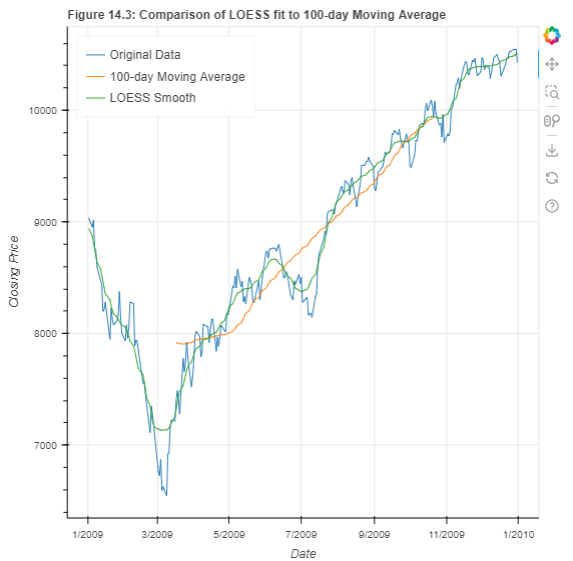
*p.line(x='date', y='loess\_smoothed', source=ColumnDataSource(data\_2009), line\_color=Category10[3][2], legend\_label='LOESS Smooth')*

*# Hiển thị legend ở vị trí top\_left*

*p.legend.location = 'top\_left'*

*# Hiển thị biểu đồ*

*show(p)*

**

**Lưu ý:**

* Thường làm mịn dữ liệu thị trường chứng khoán bằng cách tính toán đường trung bình động.
* Xu hướng tổng thể được thể hiện bằng đường LOESS trơn tru gần giống với đường trung bình động 100 ngày, nhưng đường cong LOESS mượt mà hơn nhiều và nó mở rộng ra toàn bộ phạm vi dữ liệu.

## 10.2 Showing trends with a defined functional form

* Các bản gửi hàng tháng tới máy chủ in sẵn bioRxiv. Đường liền nét màu xanh biểu thị số lượng bản in trước thực tế hàng tháng và đường màu đen nét đứt biểu thị mức độ phù hợp theo cấp số nhân của dữ liệu  y=60exp[0.77(x−2014)]
* Phương thức p.line() sử dụng để thể hiện dữ liệu theo hàm y=60exp[0.77(x−2014)]

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*data = pd.read\_csv('./data/csv\_files/preprint\_growth.csv')*

*# Extract date and count data for bioRxiv*

*date = pd.to\_datetime(data[data['archive'] == 'bioRxiv']['date'])*

*count = data[data['archive'] == 'bioRxiv']['count']*

*# Define the exponential function*

*def exponential\_func(x, A, m):*

*return A \* np.exp(m \* (x - date.iloc[0]).dt.days / 365)*

*# Fit exponential curve to the data*

*popt, pcov = curve\_fit(exponential\_func, date, count)*

*# Generate exponential fit values*

*exponential\_fit = exponential\_func(date, \*popt)*

*Plotting:*

*# Create a ColumnDataSource*

*source = ColumnDataSource(data=dict(date=date, count=count, exponential\_fit=exponential\_fit))*

*# Create the figure*

*p = figure(title='Monthly submissions to bioRxiv', x\_axis\_label='Year', y\_axis\_label='Preprints / month')*

*# Plot the actual counts*

*p.line(x='date', y='count', source=source, line\_width=2, legend\_label='Actual counts', color='blue')*

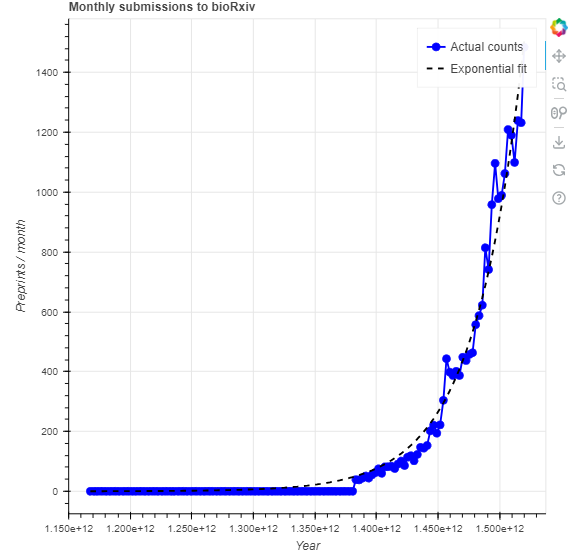
*p.circle(x='date', y='count', source=source, size=8, color='blue', legend\_label='Actual counts')*

*# Plot the exponential fit*

*p.line(x='date', y='exponential\_fit', source=source, line\_width=2, legend\_label='Exponential fit', color='black', line\_dash='dashed')*

*# Show the plot*

*show(p)*

**

**Lưu ý:**

* Để hiển thị mối quan hệ phi tuyến tính, chúng ta cần đoán xem dạng hàm thích hợp có thể là gì.

## 10.3 Detrending and time-series decomposition

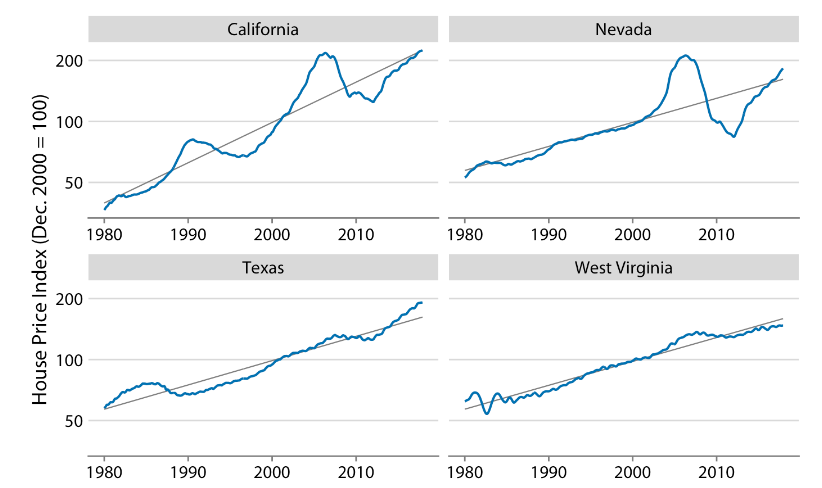
### A. Detrending

* Chỉ số giá nhà của Freddie Mac từ năm 1980 đến năm 2017, ở bốn bang được chọn (California, Nevada, Texas và West Virginia)

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*Plotting:*

**

### B. Decompose the Keeling curve

* Phân rã chuỗi thời gian của đường cong Keeling, hiển thị mức trung bình hàng tháng (như trong Hình [14.12](https://clauswilke.com/dataviz/visualizing-trends.html#fig:keeling-curve) ), xu hướng dài hạn, biến động theo mùa và phần còn lại. Phần còn lại là sự khác biệt giữa số đọc thực tế và tổng của xu hướng dài hạn cũng như biến động theo mùa và nó thể hiện tiếng ồn ngẫu nhiên.
* Phương thức p.line được sử dụng để thể hiện xu hướng.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*data = pd.read\_csv('./data/csv\_files/CO2.csv')*

*Plotting:*

*# Create figures*

*p\_ave = figure(title='Monthly Average CO2 Concentration (ppm)', x\_axis\_label='Year', y\_axis\_label='CO2 Concentration (ppm)')*

*p\_trend = figure(title='Long-term Trend', x\_axis\_label='Year', y\_axis\_label='CO2 Concentration (ppm)')*

*p\_seasonal = figure(title='Seasonal Fluctuations', x\_axis\_label='Year', y\_axis\_label='CO2 Concentration (ppm)')*

*p\_remainder = figure(title='Remainder', x\_axis\_label='Year', y\_axis\_label='CO2 Concentration (ppm)')*

*# Plot monthly average*

*p\_ave.line(x=data['date\_dec'], y=data['co2\_ave'], line\_width=2, legend\_label='Monthly Average CO2 Concentration (ppm)', color='blue')*

*# Plot long-term trend*

*p\_trend.line(x=data['date\_dec'], y=data['co2\_trend'], line\_width=2, legend\_label='Long-term Trend', color='green')*

*# Plot seasonal fluctuations*

*p\_seasonal.line(x=data['date\_dec'], y=data['co2\_ave'] - data['co2\_trend'], line\_width=2, legend\_label='Seasonal Fluctuations', color='orange')*

*# Plot remainder*

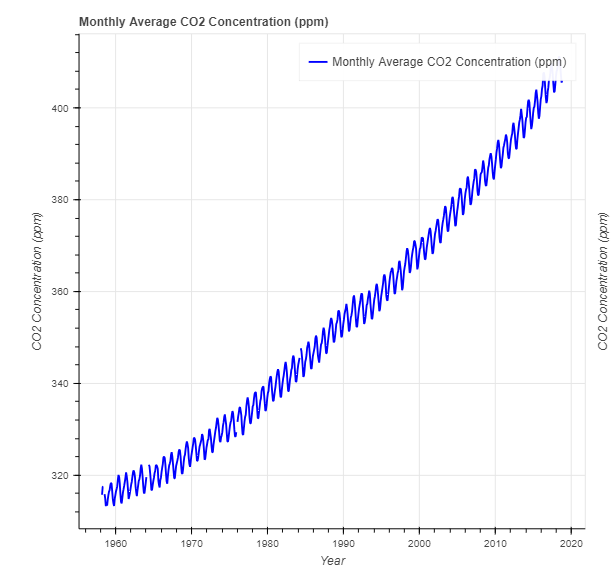
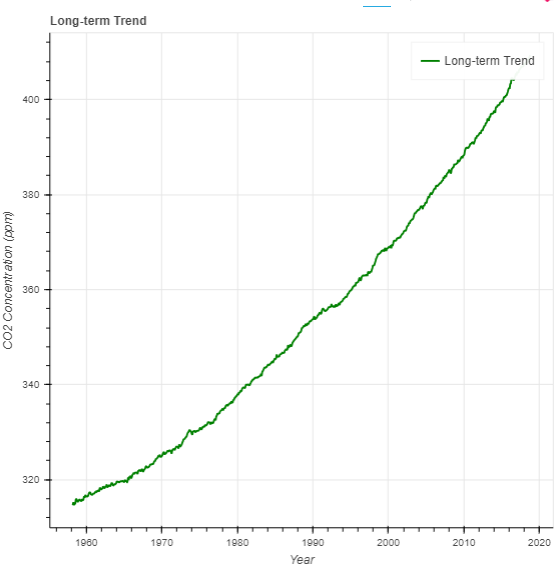
*p\_remainder.line(x=data['date\_dec'], y=data['co2\_interp'] - data['co2\_ave'], line\_width=2, legend\_label='Remainder', color='red')*

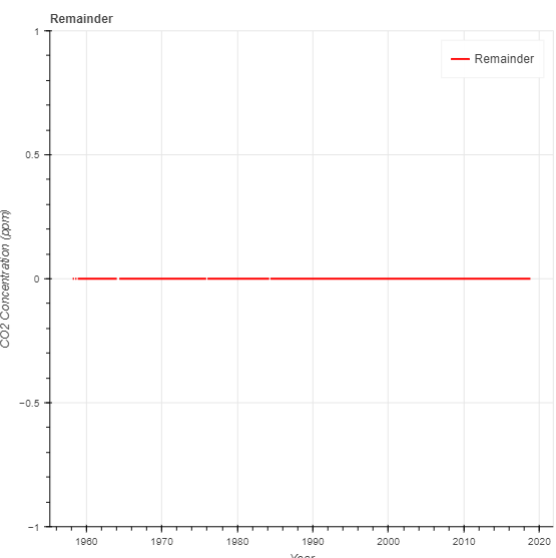
*# Create grid layout*

*grid = gridplot([[p\_ave, p\_trend], [p\_seasonal, p\_remainder]])*

*# Show the plot*

*show(grid)*

** **

** **

**Lưu ý:**

* Được sử dụng để so sánh những biến động theo các tiêu chí đặt ra

# CHƯƠNG 11: TRỰC QUAN HÓA DỮ LIỆU KHÔNG GIAN ĐỊA LÝ

## 11.1 World Map

* Dưới đây chúng tôi đã vẽ bản đồ thế giới bằng cách sử dụng ô CARTODBPOSITRON xếp. Lần đầu tiên chúng tôi tìm thấy phép chiếu Mercator trên web cho một thế giới dựa trên vĩ độ và kinh độ của thế giới. Sau đó, chúng tôi đang sử dụng dữ liệu chiếu Mercator trên web đó để đặt phạm vi trục x và y.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*from pyproj import Proj, transform*

*inProj = Proj(init='epsg:3857')*

*outProj = Proj(init='epsg:4326')*

*world\_lon1, world\_lat1 = transform(outProj,inProj,-180,-85)*

*world\_lon2, world\_lat2 = transform(outProj,inProj,180,85)*

*Plotting:*

*cartodb = get\_provider(CARTODBPOSITRON)*

*fig = figure(plot\_width=800, plot\_height=700,*

*x\_range=(world\_lon1, world\_lon2),*

*y\_range=(world\_lat1, world\_lat2),*

*x\_axis\_type="mercator", y\_axis\_type="mercator")*

*fig.add\_tile(cartodb)*

*show(fig)*

**

## 11.2 Layers

* Phương thức bokeh.models.GMapOptionslớp chuyên dụng để đặt các tùy chọn bản đồ.
* bokeh.plotting.gmapđể tạo hình.
* Sau khi tạo hình, việc thiết lập nguồn dữ liệu và thêm glyphs sẽ diễn ra như bình thường.

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*df = pd.read\_csv('datasets/pxd\_permian\_wells.csv')*

*Plotting:*

*map\_options = bokeh.models.GMapOptions(lat=np.mean(df['latitude\_surface\_hole'].values),*

*lng=np.mean(df['longitude\_surface\_hole'].values),*

*map\_type="terrain", zoom=5)*

*p = bokeh.plotting.gmap(os.environ['GOOGLE\_API\_KEY'], map\_options, title="Well*

*Locations", tools='box\_select,tap,pan,wheel\_zoom,reset',*

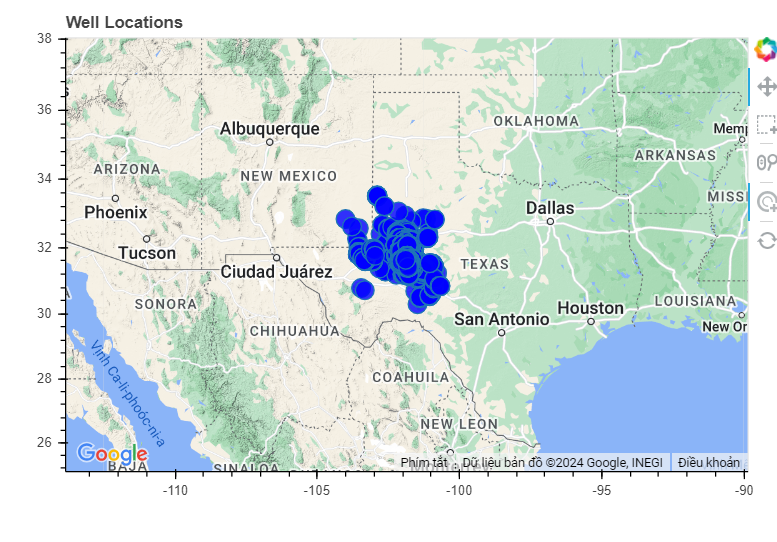
*width=600, height=400)*

*source = bokeh.models.ColumnDataSource(df)*

*p.circle(x='longitude\_surface\_hole', y='latitude\_surface\_hole', size=15, fill\_color="blue",*

*fill\_alpha=0.8, source=source)*

*bokeh.io.show(p)*

**

**Lưu ý:**

* Lớp này gmap yêu cầu khóa API của Google làm đối số đầu tiên

## 11.3 Choropleth mapping

* bản đồ dân số của tiểu bang Hoa Kỳ. Chúng tôi đã hợp nhất khung dữ liệu dân số của tiểu bang Hoa Kỳ được tải ở đầu với khung dữ liệu của bước trước và cũng đã thêm số lượng dân số vào từ điển nguồn dữ liệu được tạo ở bước trước

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*us\_states\_df = us\_states\_df.merge(us\_state\_population, how="left", left\_on="name", right\_on="State")*

*us\_states\_datasource["Population"] = us\_states\_df.Population.values.tolist()*

*Plotting:*

*from bokeh.palettes import YlGnBu9 as YlGnBu*

*fig = figure(plot\_width=900, plot\_height=600,*

*title="United States Population Per State [2018] Choropleth Map",*

*x\_axis\_location=None, y\_axis\_location=None,*

*tooltips=[*

*("Name", "@name"), ("Population", "@Population"), ("(Long, Lat)", "($x, $y)")*

*])*

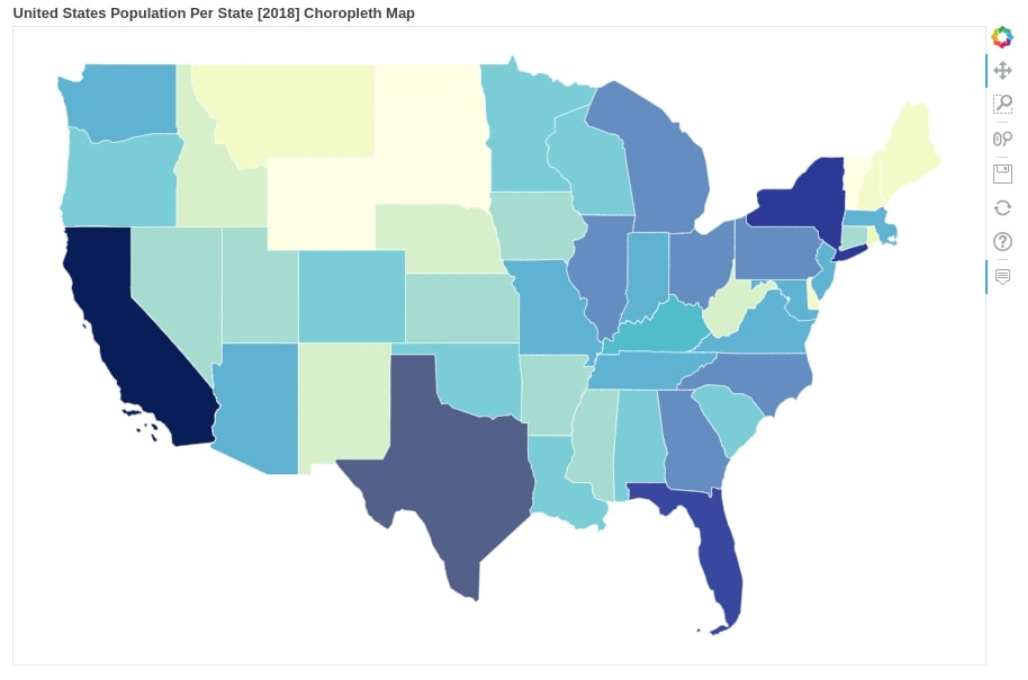
*fig.grid.grid\_line\_color = None*

*fig.patches("lons", "lats", source=us\_states\_datasource,*

*fill\_color={'field': 'Population', 'transform': LogColorMapper(palette=YlGnBu[::-1])},*

*fill\_alpha=0.7, line\_color="white", line\_width=0.5)*

*show(fig)*

**

**Lưu ý:**

* Choropleth hoạt động tốt nhất khi màu sắc thể hiện mật độ (tức là một số lượng chia cho diện tích bề mặt

# CHƯƠNG 12: HÌNH DUNG SỰ KHÔNG CHẮC CHẮN

## 12.2 Visualizing the uncertainty of point estimates

**Cách vẽ:**

*Chuẩn bị dữ liệu:*

*data = pd.read\_csv('data/csv\_files/cacao.csv')*

*# Data preparation*

*mean = 3.5*

*std\_dev = 0.5*

*std\_error = 0.2*

*confidence\_intervals = {*

*"80%": (3.2, 3.8),*

*"95%": (3.0, 4.0),*

*"99%": (2.8, 4.2)*

*}*

*# Construct the error bars data*

*error\_bars = {*

*"Standard Deviation": [mean - 2 \* std\_dev, mean + 2 \* std\_dev],*

*"Standard Error": [mean - 2 \* std\_error, mean + 2 \* std\_error],*

*"80% Confidence Interval": confidence\_intervals["80%"],*

*"95% Confidence Interval": confidence\_intervals["95%"],*

*"99% Confidence Interval": confidence\_intervals["99%"]*

*}*

*# Extracting the lower and upper limits for the error bars*

*lower\_bound = [mean - val[0] for val in error\_bars.values()]*

*upper\_bound = [val[1] - mean for val in error\_bars.values()]*

*x = list(error\_bars.keys())*

*y = [mean] \* len(x)*

*Plotting:*

*# Create a ColumnDataSource with the necessary data*

*source = ColumnDataSource(data=dict(x=x, y=y, lower=lower\_bound, upper=upper\_bound))*

*# Create a Bokeh figure*

*p = figure(x\_range=x, height=250, title="Relationship between Sample, Sample Mean, and Various Confidence Intervals",*

*toolbar\_location=None, tools="")*

*# Add bars to the figure*

*p.vbar(x='x', top='y', width=0.5, source=source, line\_color="white", fill\_color='rgba(50, 171, 96, 0.6)')*

*# Add Whiskers for error bars*

*p.add\_layout(Whisker(source=source, base="x", upper="upper", lower="lower", level="overlay"))*

*# Set the figure's axises and titles*

*p.xgrid.grid\_line\_color = None*

*p.y\_range.start = 0*

*p.y\_range.end = 5*

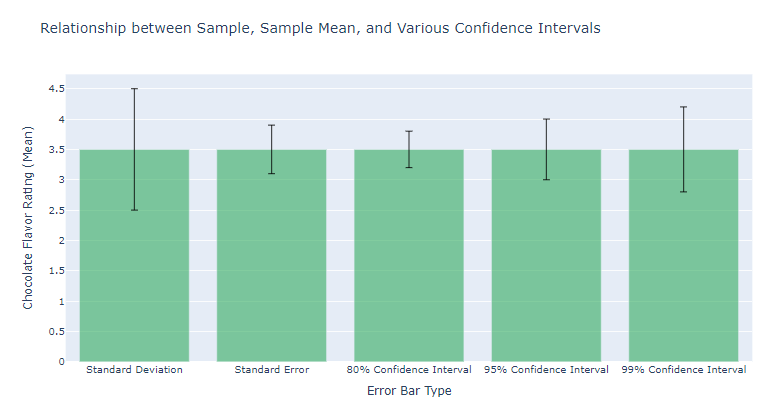
*p.xaxis.axis\_label = "Error Bar Type"*

*p.yaxis.axis\_label = "Chocolate Flavor Rating (Mean)"*

*p.xaxis.major\_label\_orientation = 1.2*

*# Show the plot*

*show(p)*

**

# KẾT LUẬN

Bokeh là một thư viện đồ họa tương tác và trực quan đặc biệt được thiết kế để tạo ra các biểu đồ và đồ họa trên web vô cùng dễ dàng và tiện lợi. Nhờ vào khả năng tương tác cao và việc hỗ trợ một loạt các loại biểu đồ phức tạp, Bokeh đã trở nên ngày càng phổ biến trong các cộng đồng khoa học dữ liệu, phân tích dữ liệu và ngành công nghiệp liên quan đến dữ liệu.

Chức năng chính của Bokeh là cung cấp một API đơn giản nhưng mạnh mẽ để tạo ra các đồ họa phong phú, cho phép người dùng tương tác trực tiếp với dữ liệu thông qua trình duyệt web mà không cần phải dựa vào các ứng dụng bên thứ ba. Điều này được thực hiện thông qua việc sử dụng các tính năng tương tác như zoom, pan, lựa chọn, và nhiều hơn nữa, giúp mang lại cái nhìn sâu sắc và hiểu biết hơn về dữ liệu.

Bokeh có thể dễ dàng tích hợp vào các dự án Python, hỗ trợ cả mã nguồn mở và thương mại. Điều này làm cho nó trở thành một giải pháp lý tưởng cho cả nghiên cứu và phát triển ứng dụng dữ liệu lớn. Ngoài ra, thư viện cũng hỗ trợ xuất dữ liệu ra nhiều định dạng khác nhau, giúp việc chia sẻ kết quả trở nên dễ dàng.

Một trong những điểm mạnh nhất của Bokeh là sự linh hoạt cao. Nó không chỉ cho phép người dùng tạo ra các biểu đồ cơ bản như đường, cột, và hình tròn, mà còn hỗ trợ các loại biểu đồ phức tạp hơn như biểu đồ tương quan và hồ sơ dữ liệu. Tính linh hoạt này mở rộng khả năng sử dụng của Bokeh, từ việc trực quan hóa dữ liệu đơn giản đến phân tích dữ liệu phức tạp.

Tuy nhiên, điều quan trọng cần lưu ý là dù Bokeh mang lại khả năng tương tác và trực quan hóa mạnh mẽ, việc sử dụng nó có thể yêu cầu một lượng kiến thức cơ bản về lập trình Python và hiểu biết sâu sắc về dữ liệu đang được làm việc. Điều này có nghĩa là, để tận dụng tối đa khả năng của Bokeh, người dùng cần phải có sự hiểu biết cơ bản về khoa học dữ liệu và kỹ năng phân tích dữ liệu.

Kết luận, Bokeh là một công cụ tuyệt vời và linh hoạt cho việc trực quan hóa dữ liệu động và tương tác trên web. Với khả năng tích hợp dễ dàng và đa dạng các loại biểu đồ, Bokeh phục vụ rộng rãi cho cả cộng đồng khoa học dữ liệu và các nhà phát triển ứng dụng web. Trên hết, sự tương tác và linh hoạt của nó giúp đẩy mạnh việc khám phá và hiểu biết dữ liệu trong nhiều bối cảnh khác nhau.