
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# ΟΠΤΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ PYTHON

Διπλωματική Εργασία

**Αλεξούλης Ευγένιος**

**Επιβλέπων:** Σταμούλης Γεώργιος


ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# ΟΠΤΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ PYTHON

Διπλωματική Εργασία

**Αλεξούλης Ευγένιος**

**Επιβλέπων:** Σταμούλης Γεώργιος


UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# DATA VISUALIZATION IN PYTHON

Diploma Thesis

**Alexoulis Evgenios**

**Supervisor:** Stamoulis Georgios

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Σταμούλης Γεώργιος**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών/Πανεπιστήμιο Θεσσαλίας

Μέλος **Κοζύρη Μαρία**
Επίκουρη Καθηγήτρια, Τμήμα Πληροφορικής και Τηλεπικοινωνιών/Πανεπιστήμιο Θεσσαλίας

Μέλος **Δημητρίου Γεώργιος**
Επίκουρος Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών/Πανεπιστήμιο Θεσσαλίας

# Ευχαριστίες

Θα ήθελα να εκφράσω την εκτίμηση μου για τον επιβλέπωντα καθηγητή μου κ. Γεώργιο Σταμούλη και τον κ. Δαδαλιάρη Αντώνιο για την άμεση ανταπόκριση καθ' ολη τη διάρκεια της συνεργασιάς μας. θα πρέπει να ευχαριστήσω ακόμα την οικογένειά μου για την απεριόριστη αγάπη τους. Τέλος είμαι ευγνώμων σε όλους τους φίλους μου που με βοήθησαν και στήριξαν όλα αυτά τα χρόνια.

Διπλωματική Εργασία

**ΟΠΤΙΚΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗ PYTHON**

**Αλεξούλης Ευγένιος**

# Περίληψη

Τα τελευταία χρόνια λόγω του τεράστιου όγκου πληροφορίας, έχει δημιουργηθεί η ανάγκη για την αποθήκευση και την επεξεργασία πληροφοριών με σκοπό την εξαγωγή σημαντικών συμπερασμάτων που έχουν αντίκτυπο σε ποικίλους τομείς όπως των Data Science και Business Analytics. Σε τομείς όπως στους προαναφερθέντες, η οπτικοποίηση δεδομένων εξυπηρετεί στην αναγνώριση μοτίβων καθώς και στην ανίχνευση περιοχών σε αυτά που θα οδηγήσουν είτε σε αδιέξοδο είτε σε ουσιαστική πληροφορία. Για να γίνει αυτό υπάρχουν πολλοί μέθοδοι δημιουργίας γραφημάτων με διαφορετικές δυνατότητες ανάλογα με το προγραμματιστικό περιβάλλον που χρησιμοποίειται. Μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού για τη διαδικασία οπτικοποίησης είναι η Python με την οποία και θα ασχοληθούμε. Βασιζόμενοι σε κάποιες από τις βιβλιοθήκες της θα μελετήσουμε την απόδοση, την ευχρηστία και τις δυνατότητες σχεδίασης που μας παρέχουν.

Diploma Thesis

**DATA VISUALIZATION IN PYTHON**

**Alexoulis Evgenios**

# Abstract

In recent years, due to the huge amount of information, there has been a need to store and process information in order to draw important conclusions that have an impact in various fields such as Data Science and Business Analytics. In areas such as those mentioned above, data visualization serves to identify patterns as well as detect areas in them that will lead to either dead ends or meaningful information. To do this there are many methods of creating graphs with different capabilities depending on the programming environment used. One of the most popular programming languages for the visualization process is Python, which we will deal with. Based on some of its libraries we will study the performance, usability and design possibilities they provide us.

# Πίνακας περιεχομένων

# Κατάλογος σχημάτων

# Κεφάλαιο 1

# Εισαγωγή

Η οπτικοποίηση δεδομένων είναι το τελευταίο βήμα που πραγματοποίειτε στην επιστήμη δεδομένων. Αυτό σημαίνει οτι αφού τα δεδομένα συλλεχθούν, επεξεργαστούν και μοντελοποίηθούν κατάλληλα το μόνο που απομένει είναι η οπτικοποίηση τους για την εξαγωγή του αποτελέσματος.

## 1.1 Η ανάγκη για οπτικοποίηση

Η οπτικοποίηση δεδομένων είναι ένα ισχυρό εργαλείο για την παρουσίαση πολύπλοκων πληροφοριών με τρόπο που είναι εύκολο να κατανοηθεί και να ερμηνευτεί. Με τον αυξανόμενο όγκο δεδομένων που διατίθενται στον σημερινό κόσμο, η ικανότητα μετατροπής των ακατέργαστων αριθμών σε ουσιαστικές γνώσεις έχει γίνει όλο και πιο σημαντική για τις επιχειρήσεις, τους ερευνητές και τους υπεύθυνους λήψης σημαντικών αποφάσεων γενικότερα. Η αποτελεσματική οπτικοποίηση δεδομένων μπορεί να αποκαλύψει μοτίβα και σχέσεις που μπορεί να είναι δύσκολο να δει κανείς στα ανεπεξέργαστα δεδομένα, συμβάλλοντας στη λήψη καλύτερων αποφάσεων και στη σαφέστερη επικοινωνία των ευρημάτων σε ένα ευρύτερο κοινό. Υπό αυτή την έννοια, η οπτικοποίηση δεδομένων δεν είναι απλώς μια ευχάριστη, αλλά απαραίτητη δεξιότητα για όποιον εργάζεται με δεδομένα.

## 1.2 Python

Η Python είναι ένα ισχυρό εργαλείο για την οπτικοποίηση δεδομένων, χάρη στην πλούσια συλλογή βιβλιοθηκών και πακέτων. Με την Python[1], είναι εύκολο να επεξεργαστείτε και να χειριστείτε δεδομένα και να δημιουργήσετε όμορφες, διαδραστικές απεικονίσεις που μπορούν να αποκαλύψουν μοτίβα και σχέσεις σε μεγάλα σύνολα δεδομένων. Η ευκολία χρήσης και η ευελιξία της Python την καθιστούν ιδανική γλώσσα για οπτικοποίηση δεδομένων, ιδιαίτερα για όσους εργάζονται στην υπολογιστική επιστήμη, τη μηχανική μάθηση και την ανάλυση δεδομένων. Συνδυάζοντας τις δυνατότητες χειρισμού δεδομένων της Python με τις πανίσχυρες βιβλιοθήκες οπτικοποίησης της, οι χρήστες μπορούν να δημιουργήσουν αποτελεσματικές και εντυπωσιακές απεικονίσεις δεδομένων που μπορούν να βοηθήσουν στο να ληφθούν καλύτερες αποφάσεις και να μεταδωθουν ευκολότερα πληροφορίες σε ένα ευρύτερο κοινό.

## 1.3 Βιβλιοθήκες που μελετήθηκαν

Συμμεριζόμενοι τα προαναφερθέντα, μελετήθηκαν οι παρακάτω βιβλιοθήκες της Python

1. Matplotlib
2. Bokeh
3. Bqplot
4. Seaborn

δημιουργώντας Histogram, bar plot, box plot, pie chart, line graph και scatter plot για τη κάθε μία ξεχωριστά και καταγράφηκαν οι χρόνοι δημιουργίας των αποικονίσεων.

## 1.4 Οργάνωση του τόμου

Στα κεφάλαια που ακολουθούν θα αναλυθούν μία προς μία οι βιβλιοθήκες, και οι δυνατότητες που αυτές μας παρέχουν μέσα από τον σχεδιασμό γραφημάτων. Το κάθε κεφάλαιο αποτελείται από μια βιβλιοθήκη με τα αποτελέσματα της. Θα παρουσιαστεί μέρος του κώδικα σε συνδυασμό με τα αποτελέσματα τρεξίματος τους μέσα από εικόνες. Μετά την ολοκλήρωση αυτών των κεφαλαίων παρουσιάζονται τα συμπεράσματα σε ένα ξεχωριστό κεφάλαιο(6), ως προς την συνολική ποιότητα τους και τη προοπτική για πιθανή βελτίωση τους.

# Κεφάλαιο 2

# Matplotlib

Η Matplotlib[2] είναι μια βιβλιοθήκη σχεδιασμού για τη γλώσσα προγραματισμού Python και βασίζεται πάνω στη βιβλιοθήκη της numpy. Είναι η πιο διαδεδομένη βιβλιοθήκη που χρησιμοποιείται σήμερα στο κομμάτι της γραφικής αναπαράστασης. Παρέχει ένα αντικειμενοστραφή API για να εισάγει σχεδιαγράμματα σε εφαρμογές κάνοντας χρήση ενός γενικού σκοπού GUI εργαλειοθηκών όπως το Tkinter, το wxPython, το QT και το GTK.

Η διεπαφή pylab κάνει τη matplotlib εύκολη στην εκμάθηση για έμπειρους χρήστες του MATLAB, καθιστώντας τη διαθέσιμο εναλλακτικό εργαλείο για ενασχόληση με την αριθμητική ανάλυση[3] και την επεξεργασία σήματος.

Πιο συγκεκριμένα με τη χρήση της βιβλιοθήκης είναι δυνατόν να:

1. Δημιουργηθούν αξιοδημοσίευτα γραφήματα.
2. Δημιουργήθούν διαδραστικές εικόνες που μεγενθύνονται, μετακινούνται και ενημερώνονται.
3. Προσαρμόστεί η μορφή και η διάταξη.
4. Εξαχθούν ποικίλες μορφές αρχείου.
5. Γραφεί κώδικας στο JupyterLab και σε άλλες πλατφόρμες σχεδιασμού.
6. Χρησιμοποιηθεί μια μεγάλη γκάμα από πακέτα ”τρίτων”.

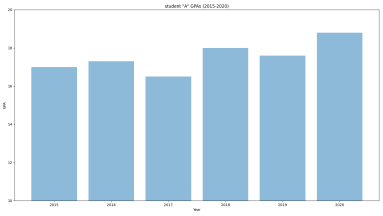
**ΘΕΤΙΚΑ**

- Υποστήριξη 3D σχεδίων.
- Μεγάλη γκάμα τύπων γραφικής παράστασης.
- Ενσωμάτωση με άλλες βιβλιοθήκες.
- Πλούσια εγχειρίδια και υποστήριξη κοινότητας.
- Διαπλατφορμική.

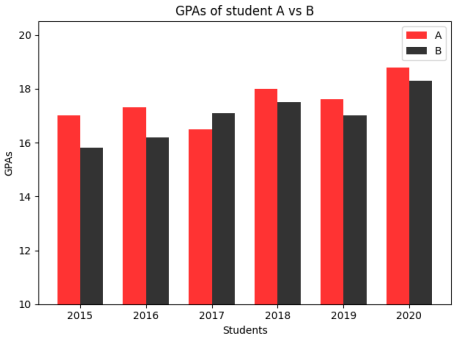
**ΑΡΝΗΤΙΚΑ**

- Απλά προεπιλεγμένα σχέδια.
- Αργή απόδοση γραφημάτων σε μεγάλο όγκο δεδομένων.
- Περιορισμένη διαδραστικότητα.
- Αδυναμία σε δεδομένα χρονοσειρών.

- **Barchart**



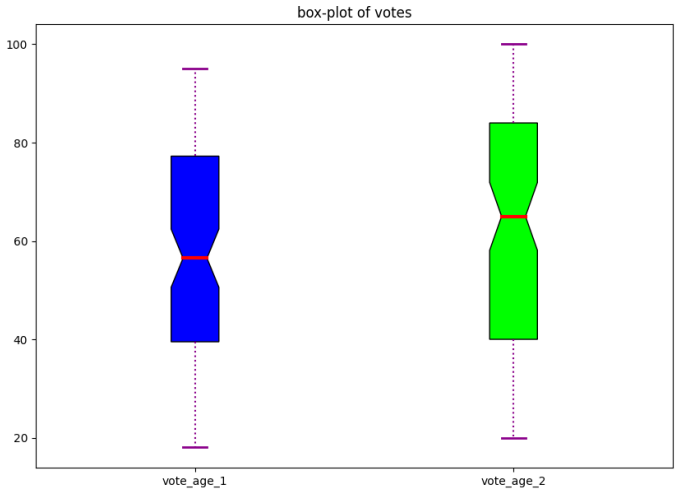
Σχήμα 2.1: Βαθμοί A 2015-20




Σχήμα 2.2: Σε σύγκριση με B

```
start_time = time()
year_arranged = np.arange(len(year))

plt.bar(year_arranged, MO_A, align='center', alpha=0.5)
plt.xticks(year_arranged, year)
plt.ylim(10,20)
plt.xlabel("Year")
plt.ylabel('GPA')
plt.title('student "A" GPAs (2015-2020)')

# plt.show()
plt.savefig("barchart_matplotlib")
print(f"Barchart Matplotlib time: {time()-start_time}")
plt.close()
```

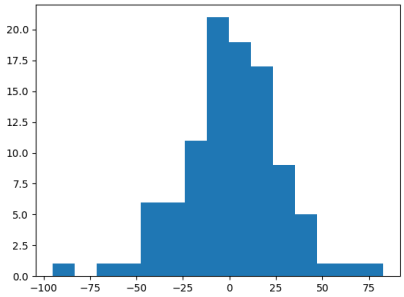
- **Boxplot**



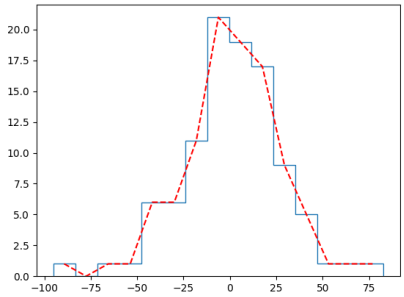

Σχήμα 2.3: Ηλικίες ψηφοφόρων

```
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(vote_age, patch_artist = True,notch ='True',)
colors = ['#0000FF', '#00FF00']
for patch, color in zip(bp['boxes'], colors):
patch.set_facecolor(color)
for whisker in bp['whiskers']:
whisker.set(color ='#8B008B',linewidth = 1.5,linestyle =":")
for cap in bp['caps']:
cap.set(color ='#8B008B',linewidth = 2)
for median in bp['medians']:
median.set(color ='red',linewidth = 3)
ax.set_xticklabels(['vote_age_1', 'vote_age_2',])
plt.title("box-plot of votes")
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
plt.show()
```

- **Histogram**



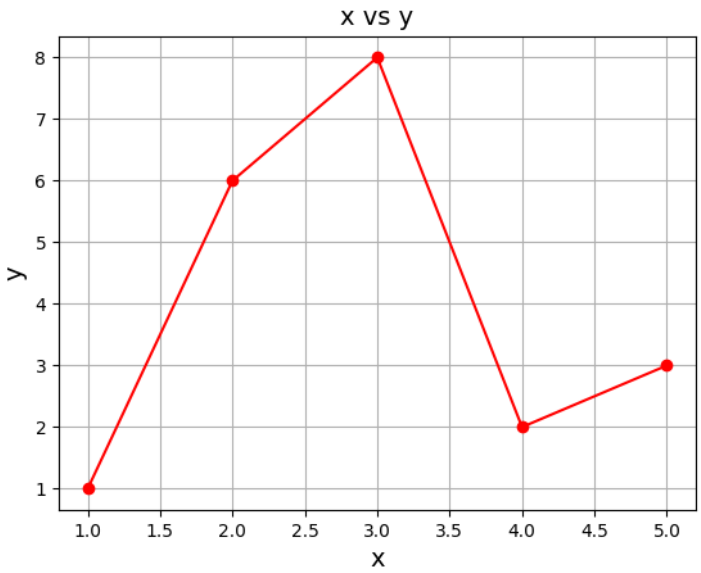

Σχήμα 2.4: Με γέμισμα




Σχήμα 2.5: Με γραμμές μέσων

```
start_time = time()
data = {'vote_age_1': vote_age1}
df = pd.DataFrame(data)
sns.boxplot(data=df, x="vote_age_1", notch=True,
    medianprops={"color":"coral"}
    )
plt.show()
# plt.savefig("boxplot_seaborn")
# print(f"Boxplot Seaborn time: {time()-start_time}")
# plt.close()
```

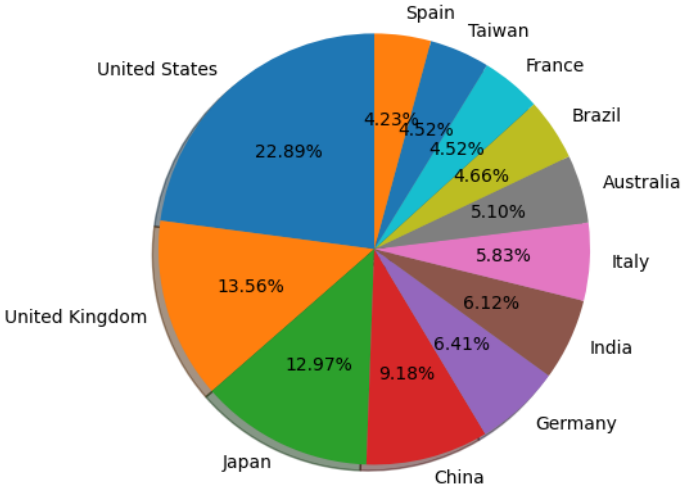
- **Linegraph**




Σχήμα 2.6: X vs Y

```
start_time = time()
data = {'x': [1, 2, 3, 4, 5],
        'y': [1, 6, 8, 2, 3]
        }
df = pd.DataFrame(data)
plt.plot(df['x'], df['y'], color='red', marker='o')
plt.title('x vs y', fontsize=14)
plt.xlabel('x', fontsize=14)
plt.ylabel('y', fontsize=14)
plt.grid(True)
# plt.show()
plt.savefig("line_matplotlib")
print(f"Line Matplotlib time: {time()-start_time}")
plt.close()
```

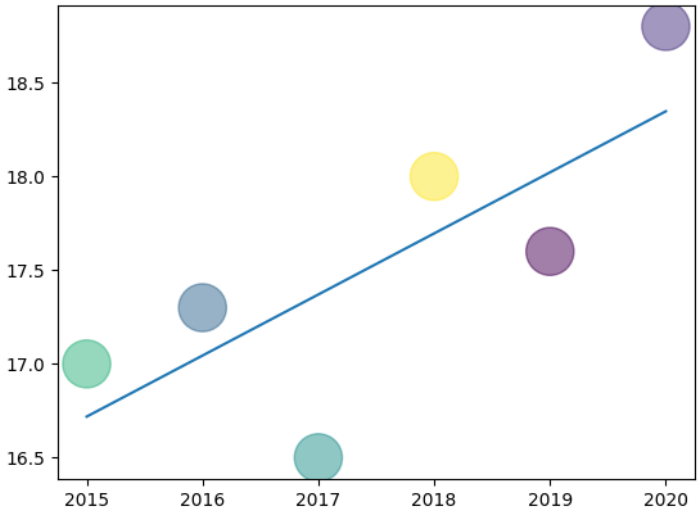
- **Piechart**




Σχήμα 2.7: Τυχαία ποσοστά χωρών σε αύξουσα σειρά

```
start_time = time()
sum_val = sum(x.values())
c_values = list(x.values())
res = [val/sum_val for val in c_values]

countries = list(x.keys())
t_countries = tuple(countries)

fig1, ax1 = plt.subplots()
ax1.pie(res, labels=t_countries, autopct='%1.2f%%',
        shadow=True, startangle=90)
ax1.axis('equal')

plt.savefig("pie_chart_matplotlib")
print(f"Pie Chart Matplotlib time: {time() - start_time}")
plt.close()
```

- **Scatter plot**




Σχήμα 2.8: Με γραμμή καλύτερου ταιριάσματος

```
start_time = time()
colors = np.random.rand(len(year))
area = (20 * (len(year)) **2)

plt.scatter(year, MO, s=area, c=colors, alpha=0.5)
y = [a*x+b for x in year1]
plt.plot(year, y)
# plt.show()
plt.savefig("scatter_matplotlib")
print(f"Scatter Plot Matplotlib time: {time() - start_time}")
plt.close()
```

# Κεφάλαιο 3

# Bokeh

Η Bokeh[4] είναι μια βιβλιοθήκη της Python για να δημιουργεί διαδραστικές οπτικοποιήσεις για σύγχρονους περιηγητές διαδικτύου. Βοηθάει στη δημιουργία αισθητικά ωραίων γραφημάτων, έχοντας δυνατότητες για σχεδιασμό απλών γραφημάτων έως πιο πολύπλοκων με συνεχή ροή από σύνολα δεδομένων. Ακόμα είναι εφικτή η δημιουργία JavaScript-τροφοδοτούμενων οπτικοποιήσεων χωρίς να χρειάζεται να γραφεί κάποιος επιπλέον κώδικας σε JavaScript από το χρήστη.

Είναι απλή στη χρήση, καθώς παρέχει μια απλή διεπαφή στους επιστήμονες δεδομένων που δεν θέλουν να αποσπώνται από την υλοποίησή της και παρέχει επίσης μια λεπτομερή διεπαφή σε προγραμματιστές και μηχανικούς λογισμικού που μπορεί να θέλουν περισσότερο έλεγχο στο Bokeh για να δημιουργήσουν πιο εξελιγμένα χαρακτηριστικά. Για να γίνει αυτό, η Bokeh[5] ακολουθεί μία πολυεπίπεδη προσέγγιση.

**ΘΕΤΙΚΑ**

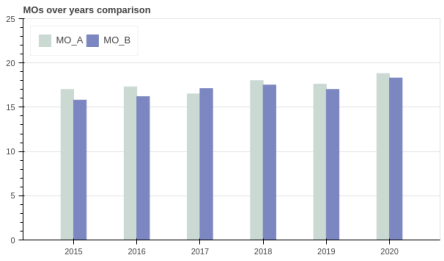
- Υψηλό επίπεδο ελέγχου για τη γρήγορη δημιουργία γραφημάτων.
- Υψηλή ανάλυση. Υψηλού επιπέδου API.
- Διαδραστικές απεικονίσεις.
- Πολλές δυνατότητες προσαρμογής.

**ΑΡΝΗΤΙΚΑ**

- Αργή σε πριπτώσεις που έχουμε μεγάλα σύνολα δεδομένων.
- Μη ενσωμετομένος 3D σχεδιασμός.
- Απαιτεί χρόνο στην εκμάθηση της.
- Περιορισμένοι τύποι γραφημάτων.

- **Bar chart**

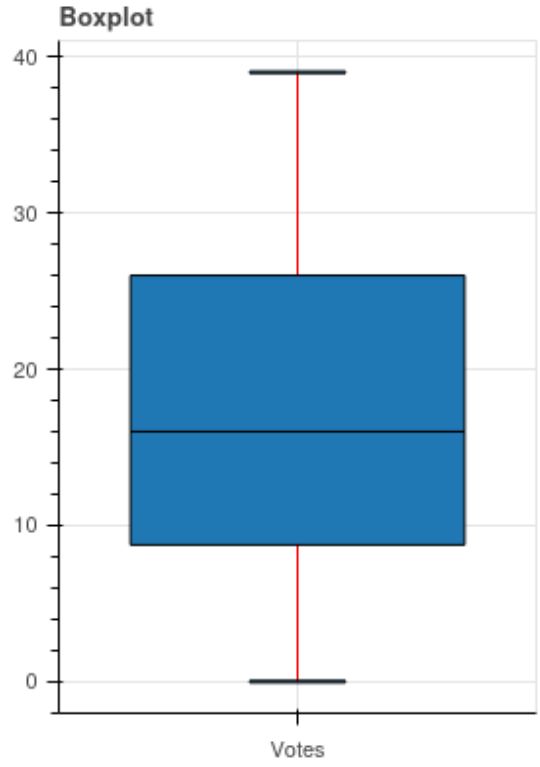

Σχήμα 3.1: MO A




Σχήμα 3.2: MO A vs B

```
start_time = time()
output_file("bar_bokeh.html")
year = ['2015','2016','2017','2018','2019','2020']
source = ColumnDataSource(data=dict(year=year,
                          MO_A=MO_A, color=Spectral6))
p = figure(x_range=year,y_range=(0,20),
           height=350, title="GPA of student",
           toolbar_location= None, tools="")
p.vbar(x='year', top='MO_A', width=0.9,
       color = 'color', source = source)

p.xgrid.grid_line_color = None
show(p)
print(f"Barchart Bokeh time: {time()-start_time}")
```

- **Box plot**



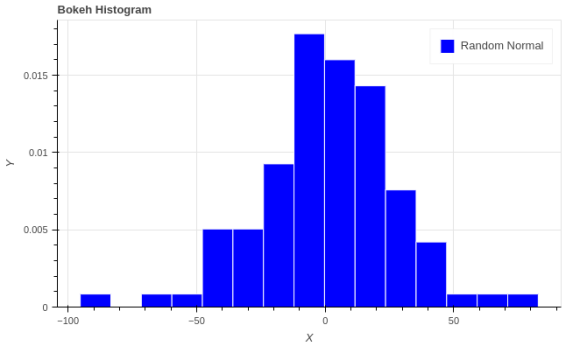

Σχήμα 3.3: Ηλικίες ψηφοφόρων

```
start_time = time()
series = pd.Series(list(np.random.randint(0,40,100)))
qmin, q1, q2, q3, qmax = (
            series.quantile([0, 0.25, 0.5, 0.75, 1]))
iqr = q3 - q1
upper = q3 + 1.5 * iqr
lower = q1 - 1.5 * iqr
mean = series.mean()
out = series[(series > upper) | (series < lower)]

if not out.empty:
    outlier = list(out.values)
v = 'Votes'
p = figure(
    tools="save",
```

```
    x_range= [v], # enable categorical axes
    title="Boxplot",
    plot_width=300,
    plot_height=400,
)
upper = min(qmax, upper)
lower = max(qmin, lower)
hbar_height = (qmax - qmin) / 300

# stems
p.segment([v], upper, [v], q3, line_color="red")
p.segment([v], lower, [v], q1, line_color="red")
# boxes
p.vbar([v], 0.7, q2, q3, line_color="black")
p.vbar([v], 0.7, q1, q2, line_color="black")
# whiskers
p.rect([v], lower, 0.2, hbar_height,
        line_color="black")
p.rect([v], upper, 0.2, hbar_height,
        line_color="black")

if not out.empty:
    p.circle([v] * len(outlier),
             outlier, size=6, fill_alpha=0.6)
plt.savefig("bokeh Barchart")
#show(p)
print(f"Boxplot Bokeh time: {time()-start_time}")
```

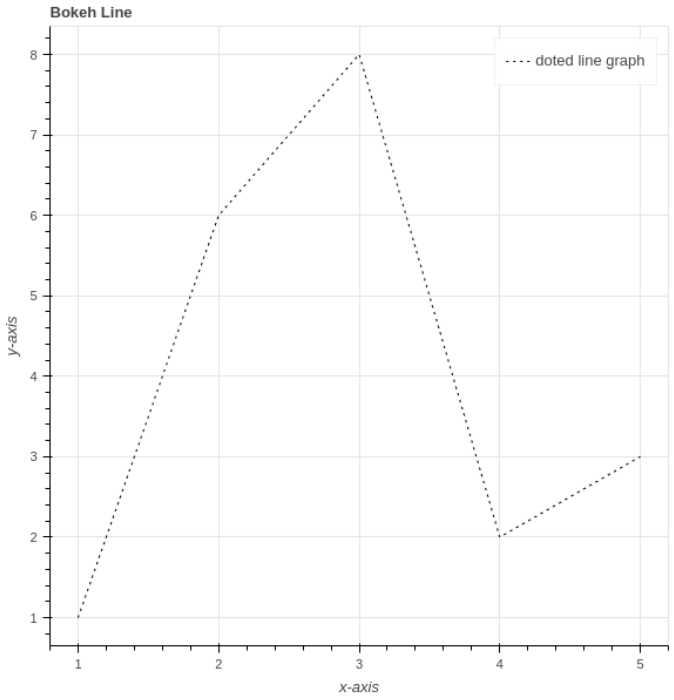
Όπως φαίνεται από τον παραπάνω κώδικα , η διαδικασία υλοποίησης αυτού του γραφήματος θεωρείται εξαιρετικά χρονοβόρα καθώς είναι απαραίτητος ο υπολογισμός όλων των σταδίων από τον χρήστη.

- **Histogram**



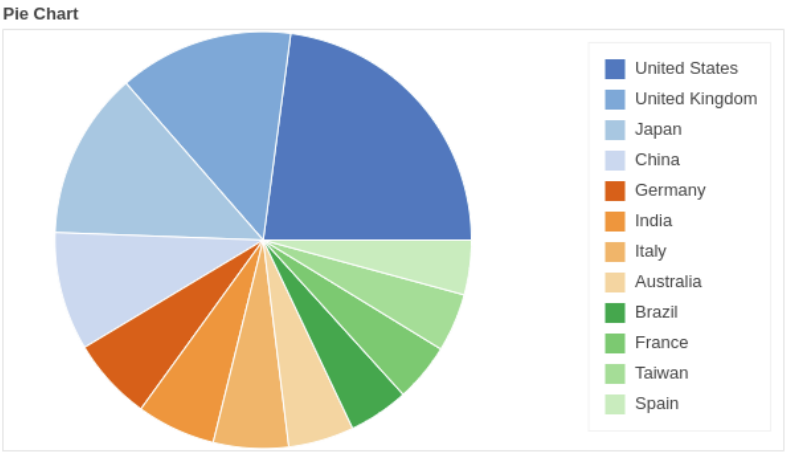

```
start_time = time()

p = figure(width=670, height=400, toolbar_location='right',
           title="Bokeh Histogram")
hist, edges = np.histogram(x, density=True, bins=15)

p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
         fill_color="blue", line_color="white",
         legend_label="Random Normal")

p.y_range.start = 0
p.xaxis.axis_label = "X"
p.yaxis.axis_label = "Y"

show(p)
print(f"Histogram Bokeh time: {time()-start_time}")
plt.close()
```

- **Line graph**




```
start_time = time()
output_file(”boke_line.html”)
graph = figure(title = ”Bokeh Line”)
graph.xaxis.axis_label = ”x-axis”
graph.yaxis.axis_label = ”y-axis”
x = [1, 2, 3, 4, 5]
y = [1, 6, 8, 2, 3]
graph.line(x, y,line_color=”black”,line_dash=”dotted”,
           legend_label=”doted line graph”,)
show(graph)
print(f”Line Bokeh time: {time()-start_time}”)
plt.close()
```

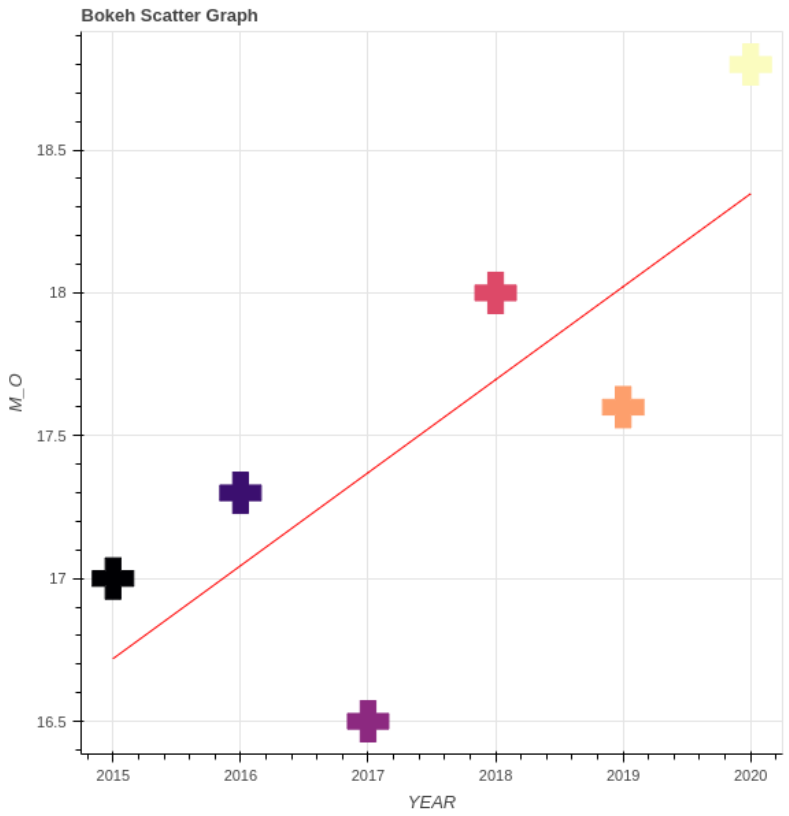
- **Pie chart**




Σχήμα 3.4: Τυχαία ποσοστά χωρών σε αύξουσα σειρά

```
p = figure(height=350, title="Pie Chart",
           toolbar_location=None,
           tools="hover", tooltips="@country: @value",
           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='country',
        source=data)

p.axis.axis_label = None
p.axis.visible = False
p.grid.grid_line_color = None
show(p)
print(f"Bokeh time to plot: {time() - start_time}")
```

- **Scatter plot**




Σχήμα 3.5: Με γραμμή καλύτερου ταιριάσματος

```
start_time = time()

graph = figure(title = ”Bokeh Scatter Graph”)
graph.xaxis.axis_label = ”YEAR”
graph.yaxis.axis_label = ”M_O”
color = magma(len(year1))
graph.scatter(year1, MO, size=30, color=color,marker=”plus”)
graph.line(year1, y, color=’red’)
show(graph)
print(f”Scatter Plot Bokeh time: {time() - start_time}”)
```

# Κεφάλαιο 4

# Seaborn

Η βιβλιοθήκη Seaborn[6] επικεντρώνεται κυρίως στη καλύτερη αναζήτηση και κατανόηση των δεδομένων από το χρήστη. Οι σχεδιαστικές της συναρτήσεις λειτουργούν πάνω σε πλαίσια δεδομένων και πίνακες που περιέχουν ολόκληρα σετ δεδομένων και εκτελούν σημασιολογική χαρτογράφηση και στατιστική συγκέντρωση για την παραγωγή πληροφοριακών γραφημάτων. Το προσανατολισμένο σε δεδομένα δηλωτικό API επιτρέπει στον χρήστη τον διαχωρισμό και τη κατηγοριοποιηση των διαφορετικών στοιχείων των γραφικών παραστάσεων, και δέν επικεντρώνεται στους τρόπους σχεδίασής τους.

Στόχος της, είναι να είναι ευέλικτη όσον αφορά τη μορφή των δεδομένων εισόδου της.Προσφέρει πολλά ενσωματωμένα θέματα που μπορούν να επιλέξουν οι χρήστες για να τροποποιήσουν την οπτική εμφάνιση των γραφημάτων τους. Τα θέματα κάνουν χρήση του συστήματος matplotlib rcParams, που σημαίνει ότι θα τεθούν σε ισχύ για κάθε φιγούρα/γραφική παράσταση που δημιουργείται χρησιμοποιώντας matplotlib, όχι μόνο για εκείνες που κατασκευάζονται από τη Seaborn.

Καθώς τα χρώματα είναι ιδιαίτερα σημαντικά στην οπτικοποίηση δεδομένων και κανένα μεμονωμένο σύνολο προεπιλογών δεν είναι καθολικά κατάλληλο, κάθε λειτουργία σχεδίασης διευκολύνει στην επιλογή μιάς εναλλακτικής παλέτας με συνεχή αντιστοίχιση κλίσης που είναι κατάλληλη για το συγκεκριμένο σύνολο δεδομένων και τον συγκεκριμένο τύπο σχεδίασης. Η ιστοσελλιδα της seaborn[7] περιέχει ένα σεμηνάριο σχετικά με τη χρήση χρώματος στην οπτικοποίηση δεδομένων για να βοηθήσει τους χρήστες να λάβουν αυτήν τη σημαντική απόφαση.

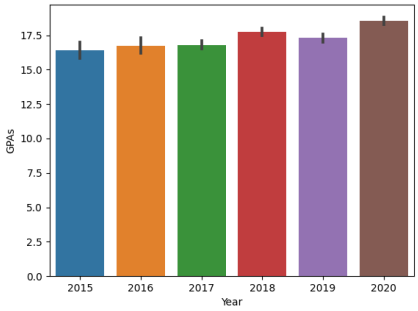
**ΘΕΤΙΚΑ**

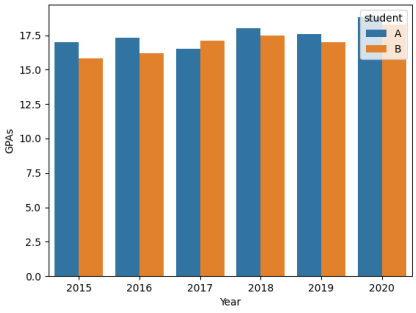
- ελκυστική οπτική αναπαράσταση.
- Εναλλαγή σε οποιαδήποτε άλλη μορφή δεδομένων.
- Εύκολη προσαρμογή των πτυχών του γραφήματος.
- Πλούσια εγχειρίδια και υποστήριξη κοινότητας.
- Υψηλού επιπέδου API.

**ΑΡΝΗΤΙΚΑ**

- Στηρίζεται στή Matplotlib λαμβάνοντας έτσι όλους τους περιορισμούς αυτής.
- Περιορισμένες προσαρμόσιμες επιλογές.
- Αργή απόδοση γραφημάτων σε μεγάλα σύνολα δεδομένων.
- Περιορισμένη προσαρμογή.

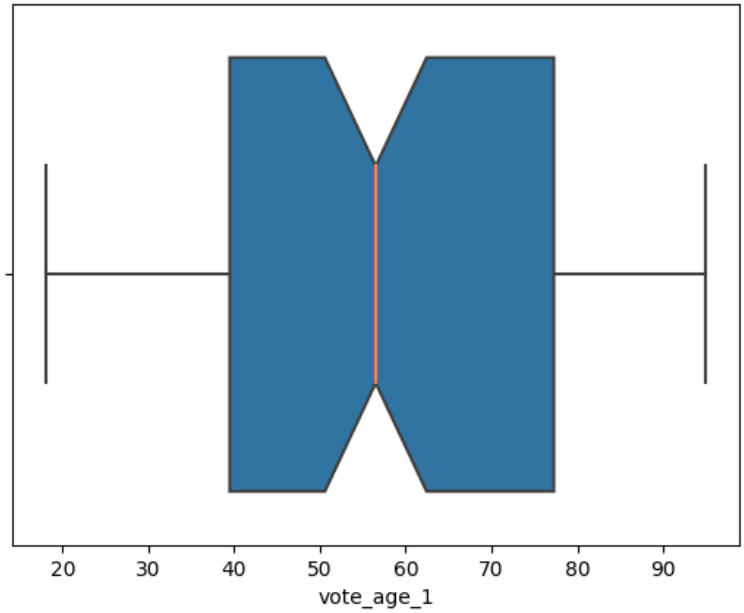
- **Bar chart**



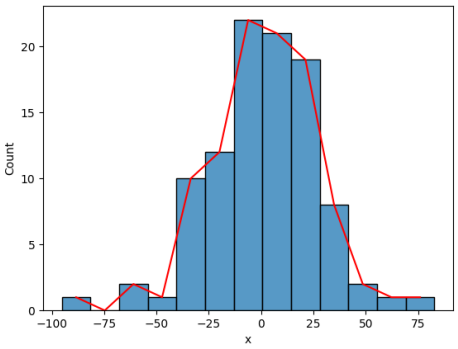

```
# plt.show()
plt.savefig(”barchart_seaborn”)
print(f”Barchart Seaborn time: {time()-start_time}”)
plt.close()

sns.barplot(data=df, x=”Year”, y=”GPAs”,hue=”student”)
# plt.show()
plt.savefig(”barchart_seaborn_grouped”)
print(f”Barchart Seaborn Grouped time:
      {time()-start_time}”)

plt.close()
```

- **Box plot**

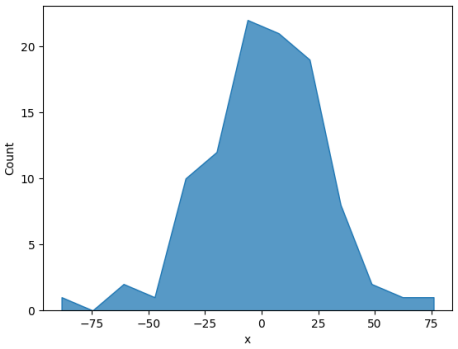



```
start_time = time()
data = {'vote_age_1': vote_age1}
df = pd.DataFrame(data)
sns.boxplot(data=df, x="vote_age_1", notch=True,
    medianprops={"color":"coral"}
    )
plt.show()
# plt.savefig("boxplot_seaborn")
# print(f"Boxplot Seaborn time: {time()-start_time}")
# plt.close()
```

- **Histogram**



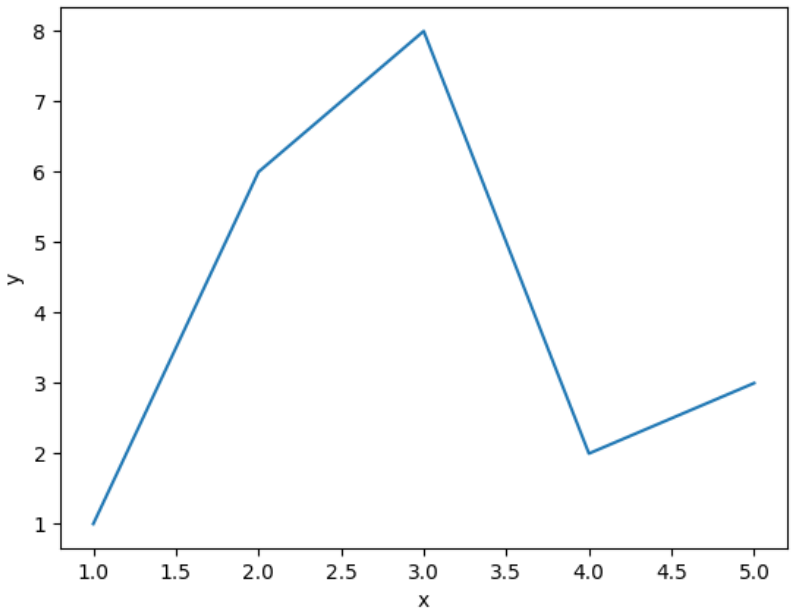

Σχήμα 4.1: Με γραμμές μέσων




Σχήμα 4.2: Σε μορφή πολύγωνου

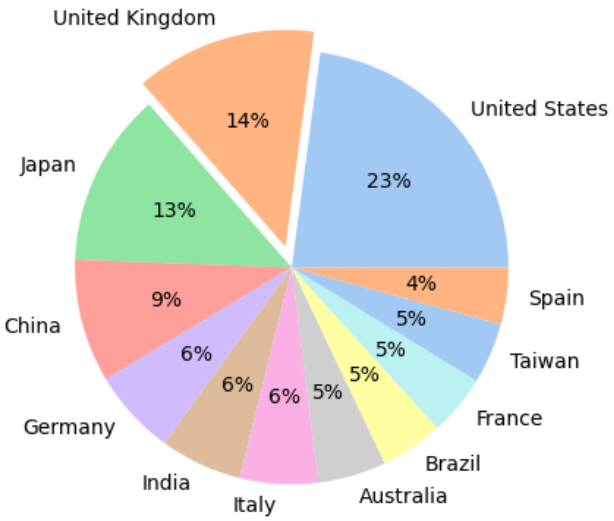
```
start_time = time()
data = {'x': x}

df = pd.DataFrame(data)
sns.histplot(data=data, x="x")
plt.show()
# plt.savefig("histogram_seaborn")
# print(f"Histogram Seaborn time: {time()-start_time}")
# plt.close()

sns.histplot(data=data, x="x", element='poly')
plt.show()
sns.histplot(data=data, x="x")
sns.histplot(data=data, x="x", element='poly', fill=False, color='red')
plt.show()
```

- **Line graph**




```
start_time = time()
data_plot = pd.DataFrame({”x”:x, ”y”:y})

sns.lineplot(x = ”x”, y = ”y”, data=data_plot)
# plt.show()
plt.savefig(”line_seaborn”)
print(f”Line Seaborn time: {time()-start_time}”)
plt.close()
```

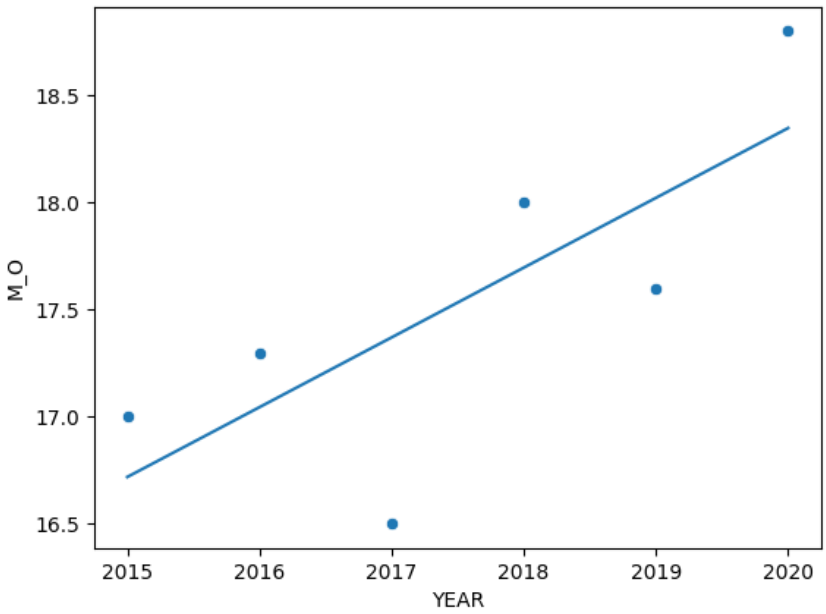
- **Pie chart**




Σχήμα 4.3: Τυχαία ποσοστά χωρών σε αύξουσα σειρά

```
start_time = time()
palette = sns.color_palette('pastel')
explode = [0,0.1,0,0,0,0,0,0,0,0,0,0]
plt.pie(res, labels = t_countries, colors = palette,
        autopct='%.0f%%',explode =explode)
# plt.show()
plt.savefig("pie_chart_seaborn")
print(f"Pie Chart Seaborn time: {time() - start_time}")
plt.close()
```

- **Scatter plot**




Σχήμα 4.4: Με γραμμή καλύτερου ταιριάσματος

```
start_time = time()
data = {'YEAR': year, 'poly': y, 'M_O': MO}
df = pd.DataFrame(data)

sns.scatterplot(data=data, x="YEAR", y="M_O")
sns.lineplot(x = "YEAR", y = "poly", data=data)
# plt.show()
plt.savefig("scatter_seaborn")
print(f"Scatter Plot Seaborn time: {time() - start_time}")
plt.close()
```

# Κεφάλαιο 5

# BQplot

Η bqplot[8] είναι ένα διαδραστικό πλαίσιο σχεδίασης βασισμένο σε Grammar of Graphics. Αναπτύχθηκε από την ομάδα ανάπτυξης του Bloomberg για το σημειωματάριο Jupyter. Κάθε χαρακτηριστικό της πλοκής είναι ένα διαδραστικό widget. Αναπτύσσεται πάνω από τη βιβλιοθήκη γραφικών στοιχείων ipywidgets, επομένως όλο το στοιχείο του γραφήματος bqplot είναι ένα γραφικό στοιχείο. Αυτό επιτρέπει στον χρήστη να ενσωματώσει οποιαδήποτε γραφική παράσταση με γραφικά στοιχεία IPython για να δημιουργήσει ένα σύνθετο και πλούσιο GUI από μερικές απλές γραμμές κώδικα Python. Επιπλέον, η bqplot παρέχει μια σειρά από διακριτικά εργαλεία, όπως το pan και zoom, τα hover tooltips και τα selection tools, που επιτρέπουν στους χρήστες να αλληλεπιδρούν με τα δεδομένα με διάφορους τρόπους[9].

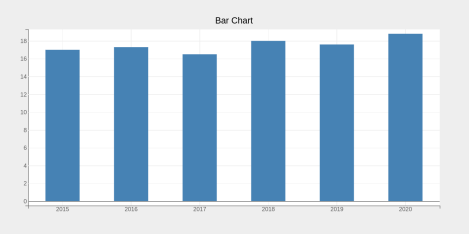
**ΘΕΤΙΚΑ**

- Άψογα αποτελέσματα με λίγες γραμμές κώδικα.
- Πλήρως διαδραστική.
- Εύκολη στη χρήση.
- Ποικιλία τύπων γραφημάτων.

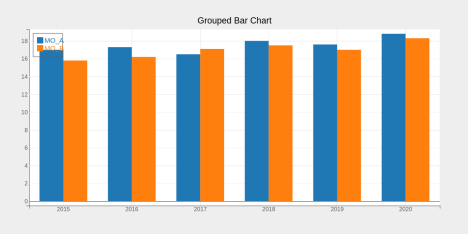
**ΑΡΝΗΤΙΚΑ**

- Δεν υποστηρίζει 3D σχεδίαση.
- Χρησιμοποιείται αποκλειστικά μέσω ”Jupyter Notebook”.
- Χρειάζεται χρόνο για εξοικείωση.
- Περιορισμένες πληροφορίες και υποστήριξη κοινότητας.
- Αργή απόδοση γραφημάτων σε μικρά σύνολα δεδομένων.

- **Bar chart**



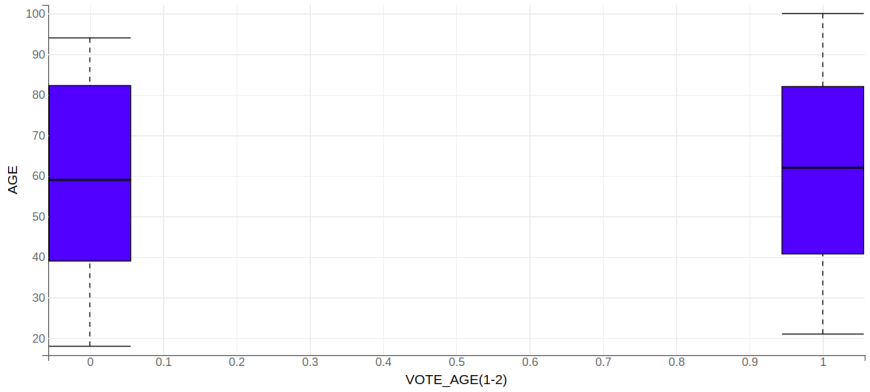

Σχήμα 5.1: Βαθμοί A 2015-20




Σχήμα 5.2: Σε σύγκριση με B

```
plt.figure(title=”Bar Chart”)
plt.bar(year, MO_A,padding=0.5)
plt.show()

plt.figure(title=”Grouped Bar Chart”,
           legend_location=”top-left”)
plt.bar(year, [MO_A,MO_B], padding=0.3,
        colors=CATEGORY10,labels=[”MO_A”,”MO_B”],
        display_legend=True,
        type=’grouped’)
plt.show()
```

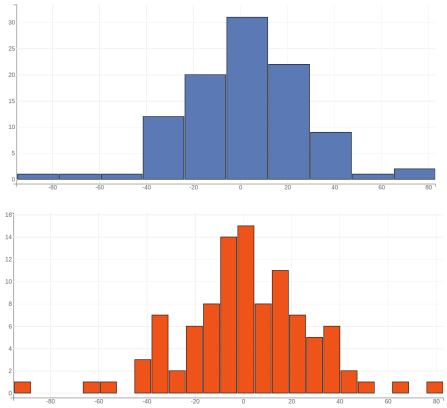
- **Box plot**

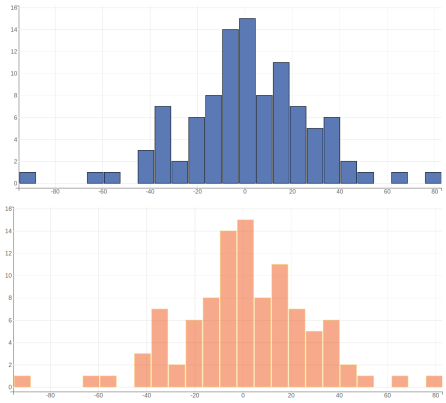



Όπως φαινεται και από το σχήμα οι θέσεις των κουτιών σε αυτή τη περίπτωση δεν μπορούν παρά να ειναι μόνο στα άκρα του γραφήματος.

```
fig = plt.figure()
axes_options = {"x": {"label": "VOTE_AGE(1-2)"},
                "y": {"label": "AGE"}}
boxes = plt.boxplot(
    x_data,
    vote_age,
    box_fill_color="blue",
    outlier_fill_color="black",
    axes_options=axes_options,
)
fig
```

- **Histogram**



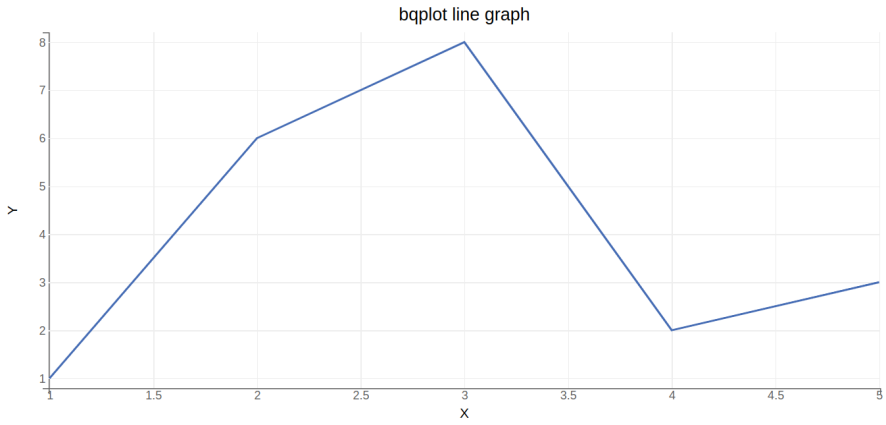






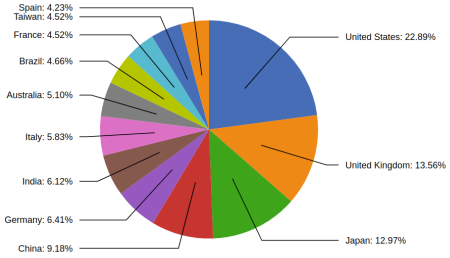
```
np.random.seed(50)
fig = plt.figure()
x= np.random.normal(0,25,100)
hist = plt.hist(x)
fig

hist.bins = 25
hist.colors = [”orangered”]
hist.stroke = ”orange”
hist.opacities = [0.5] * hist.bins
```

- **Line graph**

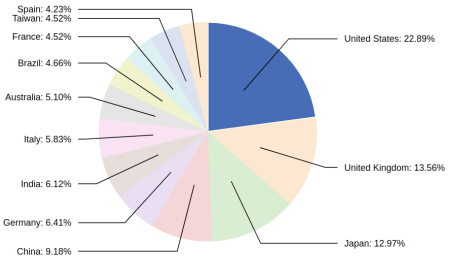



```
start_time = time()
fig = plt.figure(title="bqplot line graph")
axes_options = {"x": {"label": "X"}, "y": {"label": "Y"}}
# x values default to range of values when not specified
line = plt.plot(x,y, axes_options=axes_options)
fig
print(f"line bq time: {time()-start_time}")
```

- **Pie chart**

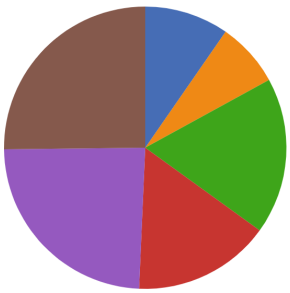



Σχήμα 5.3: Αναλυτικά ποσοστά




Σχήμα 5.4: Επιλογή κομματιού

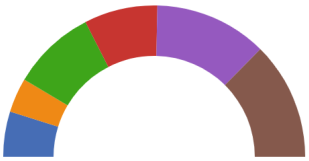
```
data = list(x.values())
pie = plt.pie(data, display_labels="outside",
              labels=list(x.keys()))
fig
pie.selected_style = {"opacity": 1, "stroke": "white",
                      "stroke-width": 2}
pie.unselected_style = {"opacity": 0.2}
pie.selected = [0]
```



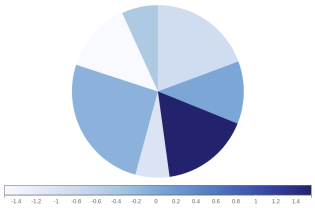
Σχήμα 5.5: Ολόκληρο Pie

Σχήμα 5.6: Donut

```
pie1 = plt.pie(np.random.rand(6), inner_radius=0.05)
fig1
with pie1.hold_sync():
    pie1.radius = 150
    pie1.inner_radius = 100
```



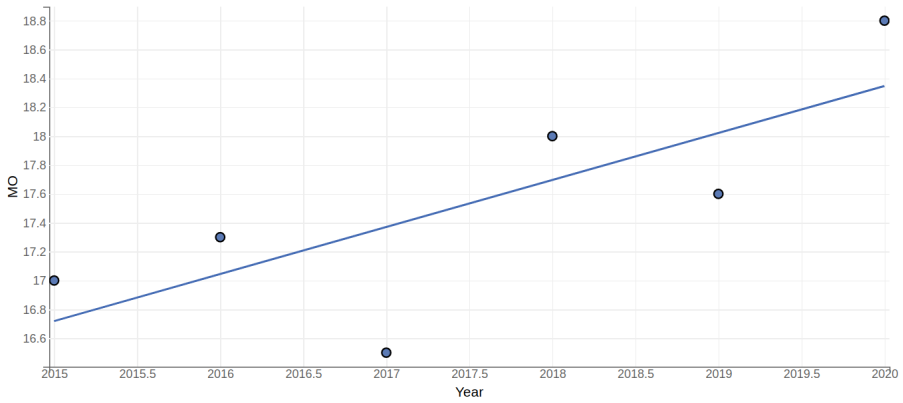
Σχήμα 5.7: Σε ημικύκλιο



Σχήμα 5.8: Με δείκτη χρωμάτων

```
with pie1.hold_sync():
    pie1.start_angle = -90
    pie1.end_angle = 90

n = 7
size_data = np.random.rand(n)
color_data = np.random.randn(n)

fig2 = plt.figure()
plt.scales(scales={”color”: ColorScale(scheme=”Blues”)})
pie2 = plt.pie(size_data, color=color_data)
fig2
```

- **Scatter plot**




Σχήμα 5.9: Με γραμμή καλύτερου ταιριάσματος

```
np.random.seed(100)
year = ['2015','2016','2017','2018','2019','2020']
year1 = [2015,2016,2017,2018,2019,2020]
MO = [17,17.3,16.5,18,17.6,18.8]
a, b = np.polyfit(year1, MO, 1)

fig = plt.figure()
axes_options = {"x": {"label": "Year"}, "y": {"label": "MO"}}
scatt = plt.scatter(year1, MO, axes_options=axes_options,
                    stroke="black")
y = [a*x+b for x in year1]
plt.plot(year, y)
fig
```

# Κεφάλαιο 6

# Συμπεράσματα

Ολοκληρώνοντας λοιπόν, έχει παρουσιαστεί ένα μεγάλο κομμάτι των δυνατοτήτων από τις υπο ανάλυση βιβλιοθήκες της Python. Κρίνοντας από τα όσα παρουσιάστηκαν και αναλύθηκαν παραπάνω όλες οι βιβλιοθήκες θεωρούνται αποτελεσματικές ανάλογα με το σκοπό και τη χρήση τους παρουσιάζοντας πλεονεκτήματα και μειονεκτήματα. Μπορούν να κατηγοριοποίηθουν και να αξιοποιήθουν κατάλληλα βάση των αναγκών του χρήστη[10].

Λαμβάνοντας υπόψη το βοηθητικό υλικό, τη συνεχή εξέλιξη, τους ενεργούς χρήστες και συνυπολογίζοντας τους χρόνους απόκρισης οπως παρουσιάζονται και στον παρακάτω πίνακα (6.1), η Matplotlib θεωρείται η πιο πλήρης βιβλιοθήκη. Είναι μια ισχυρή και ευέλικτη βιβλιοθήκη που παρέχει μια πληθόρα επιλογών για την σχεδίαση τόσο 2D αλλά και 3D μοντέλων, αλλά έχει περιορισμένες δυνατότητες στο κομμάτι του χειρισμού και της αλληλεπίδρασης μεταξύ χρήστη και γραφημάτων.

Κατατάσοντας τες λοιπόν, ακολουθεί η Seaborn η οποία δέν υστερεί στο σύνολο της σε τίποτα, αντιθέτως είναι η πιο δυνατή βιβλιοθήκη στο κομμάτι της καλαίσθητης παρουσίασης γραφημάτων, με καλές αποδόσεις σε χρόνους απεικόνισης. Είναι μια βιβλιοθήκη υψηλότερου επιπέδου που παρέχει επιστημονικά σύγχρονες οπτικοποιήσεις με ενσωματωμένες δυνατότητες στατιστικής ανάλυσης.

Η Bokeh είναι ισχυρή στη δημιουργία διαδραστικών και διαδικτυακών απεικονίσεων παρέχοντας υψηλού επιπέδου διεπαφή. Προσφέρει διαδραστικές και διαδικτυακές απεικονίσεις που είναι ιδανικές για μεγάλα και πολύπλοκα σύνολα δεδομένων αλλά είναι πιο αργή από τις άλλες βιβλιοθήκες οπτικοποίησης στα σύνολα αυτά.

Τέλος, η bqplot είναι μια βιβλιοθήκη που είναι στενά συνδεδεμένη με το Jupyter. Εχει μια καλά σχεδιασμένη και εύχρηστη διεπαφή που καθιστά σχετικά εύκολη τη δημιουργία σύνθετων απεικονίσεων με λίγες μόνο γραμμές κώδικα. Αν και μπορεί να μην είναι η καλύτερη επιλογή για χρήστες που πρέπει να δημιουργήσουν απλά στατικά σχέδια, είναι μια εξαιρετική επιλογή για όσους χρειάζονται να δημιουργήσουν δυναμικές και σύνθετες απεικονίσεις που μπορούν εύκολα να μοιραστούν και να έχουν πρόσβαση σε διαφορετικές πλατφόρμες.

Στο παρακάτω πίνακα (6.1) παρουσιάζονται οι χρόνοι που χρειάστηκαν για την υλοποίηση 6 γραφημάτων από τις 4 υπό εξέταση βιβλιοθήκες.

| Βιβλιοθήκες | Bar plot | Box plot | Histogram | Line graph | Pie chart | Scatter plot |
|---|---|---|---|---|---|---|
| Matplotlib | 0.1134 | 0.1105 | 0.0948 | 0.3396 | 0.0641 | 0.0794 |
| Bokeh | 0.3482 | 0.2545 | 0.3677 | 0.3427 | 0.3464 | 0.3864 |
| Seaborn | 0.3866 | 0.0642 | 0.1937 | 0.0837 | 0.1856 | 0.1685 |
| Bqplot | 0.0133 | 0.0137 | 0.0164 | 0.0169 | 0.0102 | 0.0122 |

Πίνακας 6.1: Χρόνοι απόκρισης σχεδιασμού για κάθε βιβλιοθήκη

Ανεξάρτητα από το ποια βιβλιοθήκη επιλέγεται, η ικανότητα οπτικοποίησης των δεδομένων με σαφή και ουσιαστικό τρόπο είναι ζωτικής σημασίας για την αποτελεσματική κατανόηση και τη λήψη αποφάσεων που βασίζονται σε δεδομένα. Με τη συνεχή ανάπτυξη και πρόοδο αυτών των εργαλείων, αναμένονται ακόμη πιο συναρπαστικές δυνατότητες οπτικοποίησης δεδομένων στο μέλλον.

# Βιβλιογραφία


[1] Python. https://www.python.org/. Ημερομηνία πρόσβασης: 12-02-2023.

[2] Matplotlib. https://matplotlib.org/. Ημερομηνία πρόσβασης: 12-02-2023.

[3] John D Hunter. Matplotlib: A 2d graphics environment. *IEEE Annals of the History of Computing*, 32(3):90–95, 2010.

[4] Bokeh. https://bokeh.org/. Ημερομηνία πρόσβασης: 12-02-2023.

[5] Bokeh Development Team. Bokeh: An interactive web plotting library. *Journal of Open Source Software*, 2(18), 2017.

[6] Michael Waskom. Seaborn: A library for statistical visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[7] Seaborn. https://seaborn.pydata.org/. Ημερομηνία πρόσβασης: 12-02-2023.

[8] Bqplot. https://bqplot.readthedocs.io/en/latest/. Ημερομηνία πρόσβασης: 12-02-2023.

[9] K. Daniel Wilkinson, Aurélien Boucaud, and Thomas Goodale. Bqplot: A 2d plotting library for the jupyter notebook. *Journal of Open Source Software*, 3(27), 2018.

[10] Mahesh Ganesh and V. Jagan Reddy. A comparative study of python visualization libraries for graphical representation of data. *International Journal of Engineering and Technology*, 10(1):167–174, 2018.