1- Bubble Plot- Dağılım Grafiği:

In [ ]:

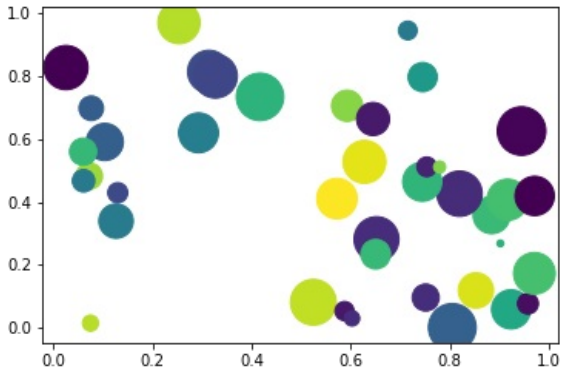
```
```

Tanım:

Balon grafiği dağılım grafiğinin (Sactter Plot) üçüncü boyutu eklenmiş halidir. Bu eklenmiş olan üçüncü boyut noktanın büyüklüğü veya balonun büyüklüğü ile ifade edlilir.

In [3]:

```
import matplotlib.pyplot as plt
import numpy as np

# create data
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
colors = np.random.rand(40)
# use the scatter function
plt.scatter(x, y, s=z*1000,c=colors)
plt.show()
```

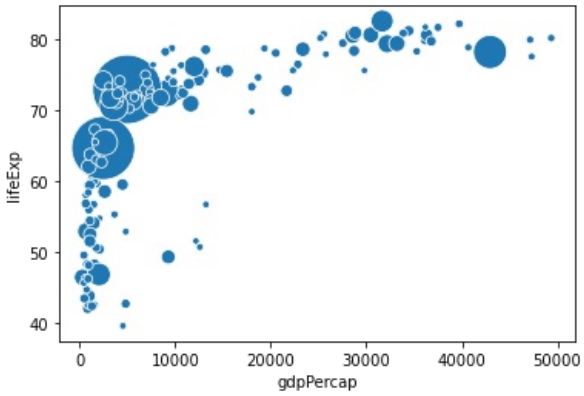


Bu grafikte her bir veri noktası için X ekseni değeri, Y ekseni değeri ve üçüncü eksen olarak da balounun büyüklüğü mevcuttur.

Bunun için Hans Rosling 'in ortalama yaşam, gdp, ve nufus miktarı grafiği en bariz örnektir.

In [4]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from gapminder import gapminder # data set

# data
data = gapminder.loc[gapminder.year == 2007]

# use the scatterplot function to build the bubble map
sns.scatterplot(data=data, x="gdpPercap", y="lifeExp", size="pop", legend=False, sizes=(20, 2000))

# show the graph
plt.show()
```



Ne için Kullanılır:

Bu grafik türü dağılım grafiğinde olduğu gibi 3 veri setinin bir birileri ile ilişkisinin incelenmesi amacıyla kullanılır.

In [ ]:

```
```

Çeşitleri:

Bu grafik türünde her bir balocuğun büyüklüğünün ne anlama geldiğinin veya etiketlenmesinin sorun olması nedeniyle bir sinir bozucu olma olasılığı mevcuttur ancak ekstrem örneklere açıklama (anotation) eklenerek bu sorun nispeten gderilebilir.

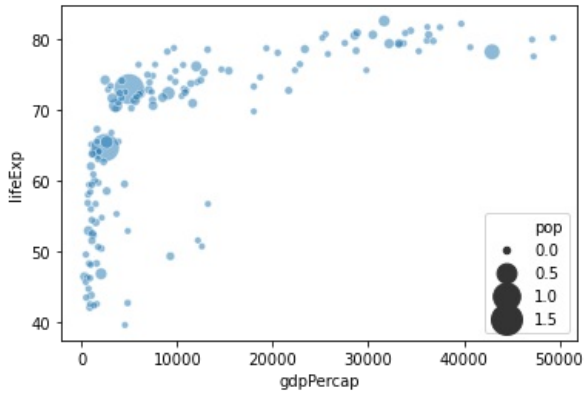
Eğer interactive bir istem kualanıyorsanız bu grafik türündeki bu etiket ve zoom sorunu da engellemiş olacaksınız.

Diğer bir tavsiye ise bu grafik türünün Hans Rosling'in animasyonları gibi anime edilmesi sunumunuza oldukça katkı sağlayacaktır. Tabiki sunucunun da bay Rosling gibi akıcı bir sunucu olması gereklidir.

Ayrıca her değişken ayrı renkte olursa daha kolay seçilir ve iniş trendi kırmızı yükseliş trendi yeşil yapılabilir yerine göre...

In [5]:

```
# libraries
import matplotlib.pyplot as plt
import seaborn as sns
from gapminder import gapminder # import data set

# data
data = gapminder.loc[gapminder.year == 2007]

# use the scatterplot function
sns.scatterplot(data=data, x="gdpPercap", y="lifeExp", size="pop", alpha=0.5, sizes=(20, 400))
plt.savefig("gdpandlifeexpendancy.png",dpi=200)
# show the graph
plt.show()
```



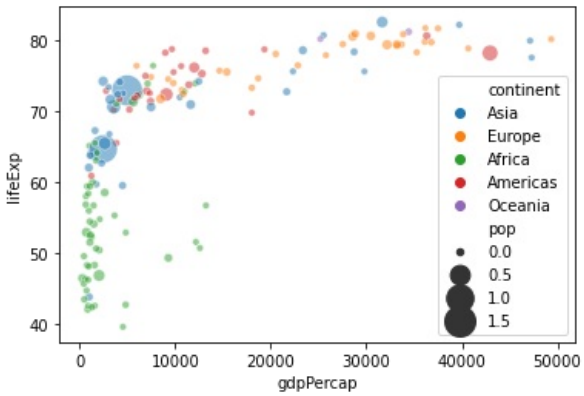
Yukarıdaki grafikte balon büyüklükleri aşağıda etiket olarak eklenmiştir.

In [ ]:

```
```

In [6]:

```
# use the scatterplot function
sns.scatterplot(data=data, x="gdpPercap", y="lifeExp", size="pop", hue="continent", alpha=0.5, sizes=(20, 400))


plt.savefig("gdpandlifeexpendancy2.png",dpi=400, figsize=400)
# show the graph
plt.show()
```



Yukaridaki grafikte hem büyüklükler belirtilmiş hem de kıta bilgisi farklı renk olarak eklenerek ilave bilgi eklenmiştir. Örneğin özellikle Afrika kıtasının hangi alanda olduğuna gözlemleyebilirsiniz.

In [ ]:

```

```

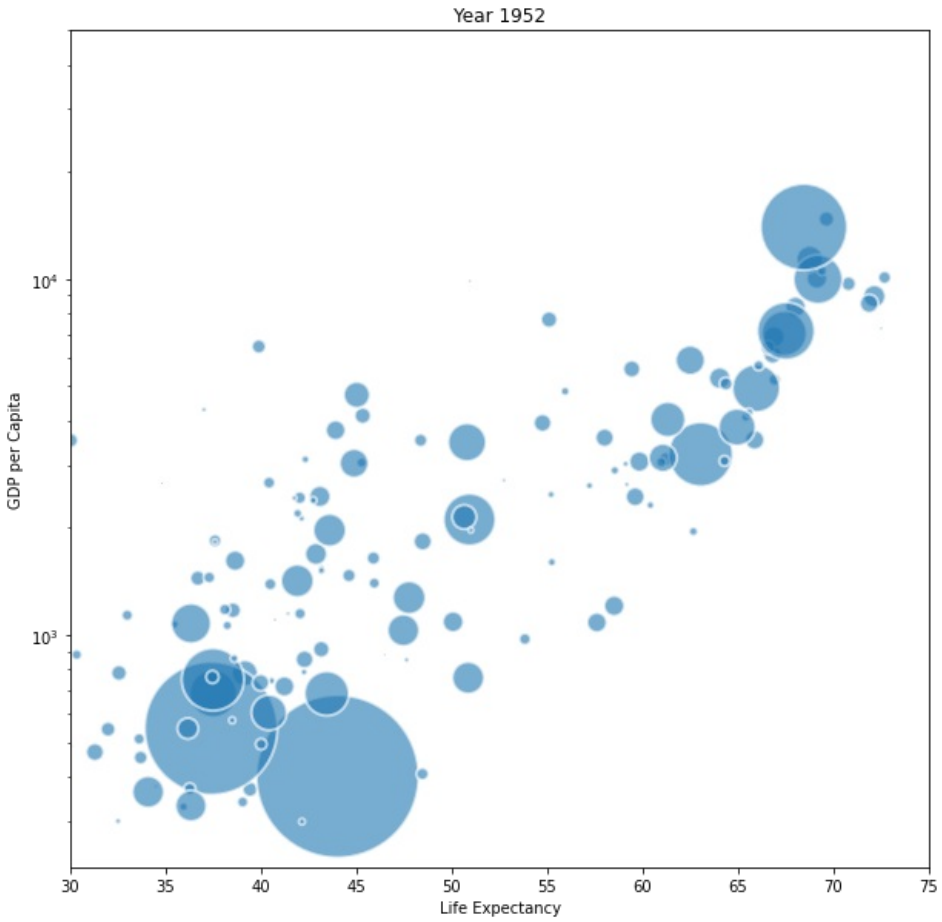
In [ ]:

```

```

In [7]:

```
import pandas as pd

# read the data (on the web)
data = pd.read_csv('gapminderData.csv')
#data = pd.read_csv('https://raw.githubusercontent.com/holtzy/The-Python-Graph-Gallery/master/static/data/gapmind
erData.csv')
# Check the first 2 rows
data.head(2)

# Set the figure size
plt.figure(figsize=(10, 10))

# Subset of the data for year 1952
data1952 = data[ data.year == 1952 ]

# Scatterplot
plt.scatter(
    x = data1952['lifeExp'],
    y = data1952['gdpPercap'],
    s=data1952['pop']/50000,
    #c=data1952['continent'].cat.codes,
    cmap="Accent",
    alpha=0.6,
    edgecolors="white",
    linewidth=2);

# Add titles (main and on axis)
plt.yscale('log')
plt.xlabel("Life Expectancy")
plt.ylabel("GDP per Capita")
plt.title("Year 1952")
plt.ylim(0,50000)
plt.xlim(30, 75);
```

```
<ipython-input-7-7f7a6e4f868e>:31: UserWarning: Attempted to set non-positive bottom ylim on a log-s
caled axis.
Invalid limit will be ignored.
  plt.ylim(0,50000)
```



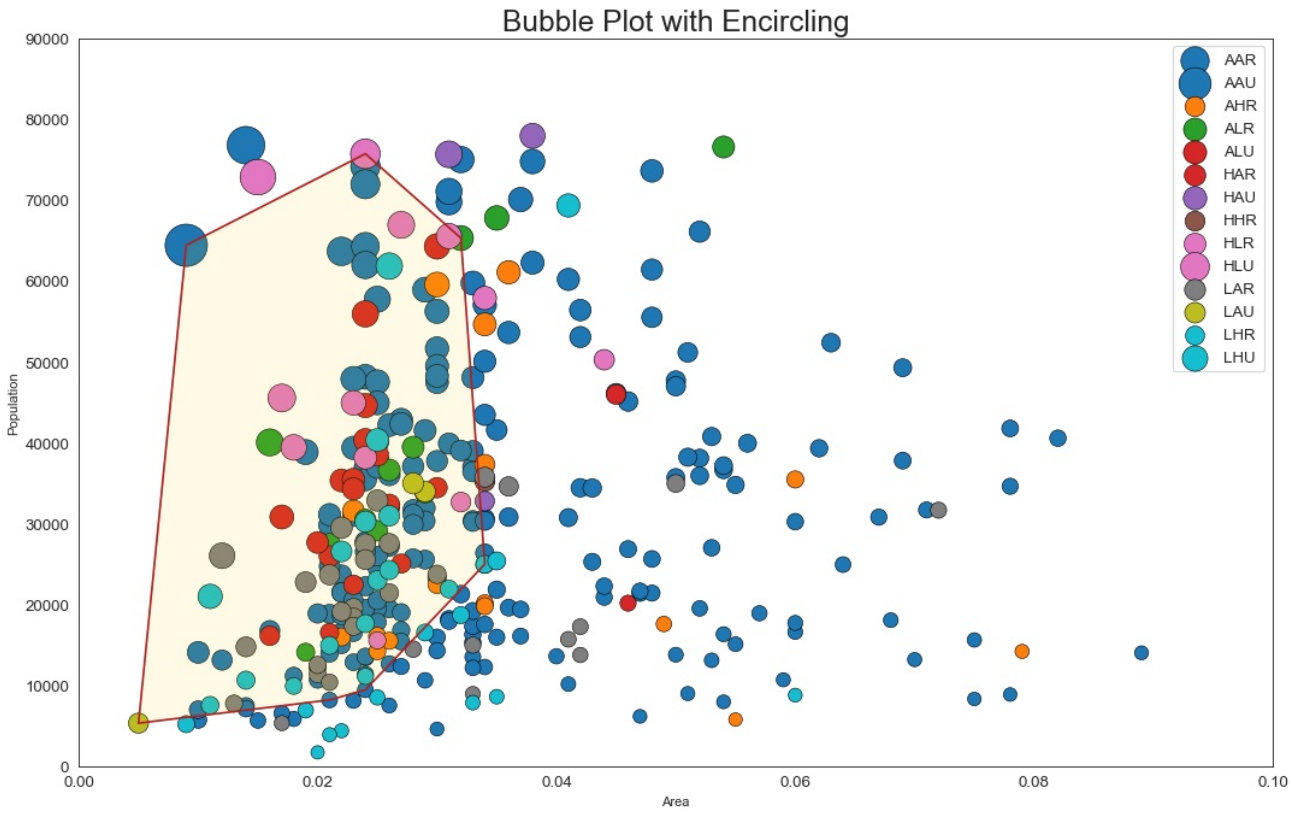
In [ ]:

Aşağıdaki örnekteki gibi belirli bir alanı bir daire veya şekil içerisine alıp bu alana vurgu yapabilirisiniz.

In [8]:

```
import numpy as np
from matplotlib import patches
from scipy.spatial import ConvexHull
import warnings; warnings.simplefilter('ignore')
sns.set_style("white")

# Step 1: Prepare Data
midwest = pd.read_csv("midwest_filter.csv")
#midwest = pd.read_csv("https://raw.githubusercontent.com/selva86/datasets/master/midwest_filter.csv")


# As many colors as there are unique midwest['category']
categories = np.unique(midwest['category'])
colors = [plt.cm.tab10(i/float(len(categories)-1)) for i in range(len(categories))]

# Step 2: Draw Scatterplot with unique color for each category
fig = plt.figure(figsize=(16, 10), dpi= 80, facecolor='w', edgecolor='k')

for i, category in enumerate(categories):
    plt.scatter('area', 'poptotal', data=midwest.loc[midwest.category==category, :], s='dot_size', c=colors[i], l
abel=str(category), edgecolors='black', linewidths=.5)

# Step 3: Encircling
# https://stackoverflow.com/questions/44575681/how-do-i-encircle-different-data-sets-in-scatter-plot
def encircle(x,y, ax=None, **kw):
    if not ax: ax=plt.gca()
    p = np.c_[x,y]
    hull = ConvexHull(p)
    poly = plt.Polygon(p[hull.vertices,:], **kw)
    ax.add_patch(poly)

# Select data to be encircled
midwest_encircle_data = midwest.loc[midwest.state=='IN', :]

# Draw polygon surrounding vertices
encircle(midwest_encircle_data.area, midwest_encircle_data.poptotal, ec="k", fc="gold", alpha=0.1)
encircle(midwest_encircle_data.area, midwest_encircle_data.poptotal, ec="firebrick", fc="none", linewidth=1.5)

# Step 4: Decorations
plt.gca().set(xlim=(0.0, 0.1), ylim=(0, 90000),
              xlabel='Area', ylabel='Population')

plt.xticks(fontsize=12); plt.yticks(fontsize=12)
plt.title("Bubble Plot with Encircling", fontsize=22)
plt.legend(fontsize=12)
plt.show()
```

c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.
c argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
ing will have precedence in case its length matches with *x* & *y*. Please use the *color* keyword-
argument or provide a 2-D array with a single row if you intend to specify the same RGB or RGBA valu
e for all points.



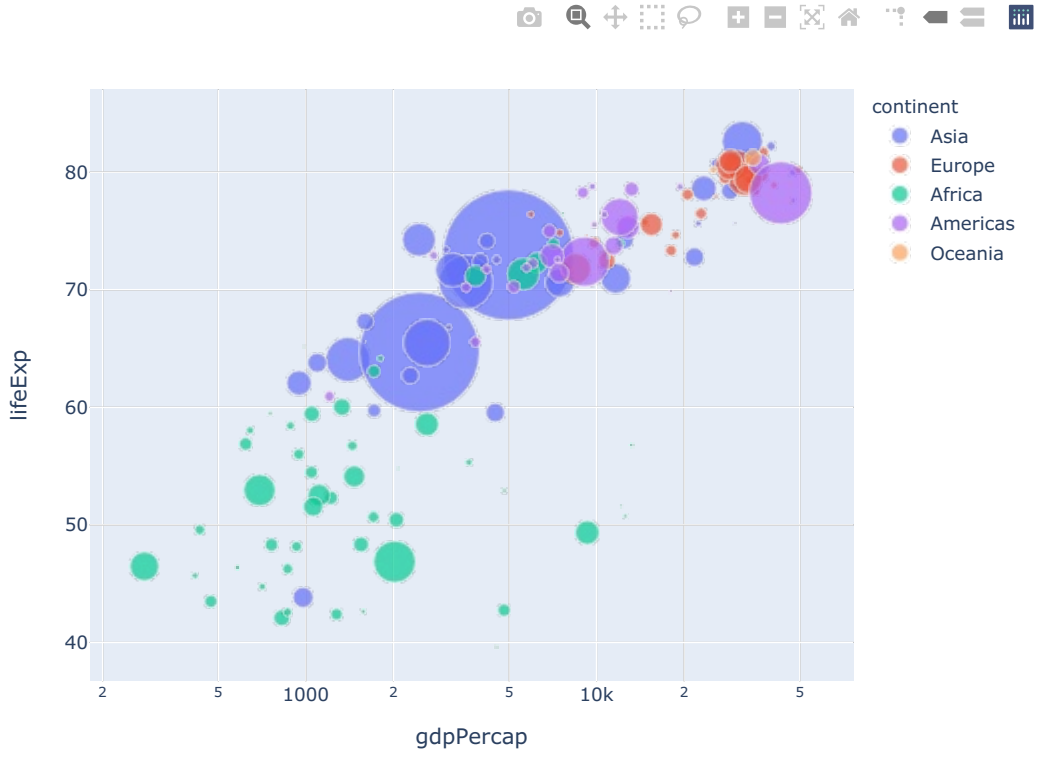
In [ ]:

```
```

Şimdi de Interactive bir Bubble Chart inceleyelim. Tabi siz bu grafiği interaktif olarak görüntülemek için kodları kopyalayıp çalıştırmalısınız

In [9]:

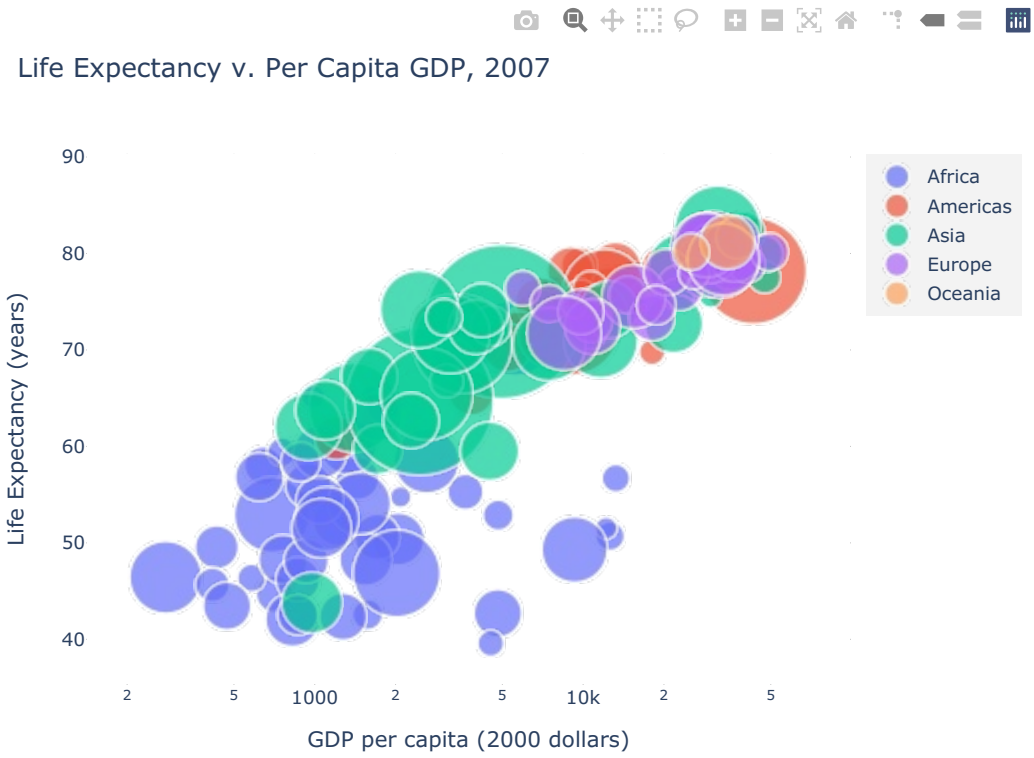
```
import plotly.express as px
df = px.data.gapminder()

fig = px.scatter(df.query("year==2007"), x="gdpPercap", y="lifeExp",
                 size="pop", color="continent",
                 hover_name="country", log_x=True, size_max=60)
fig.show()
```



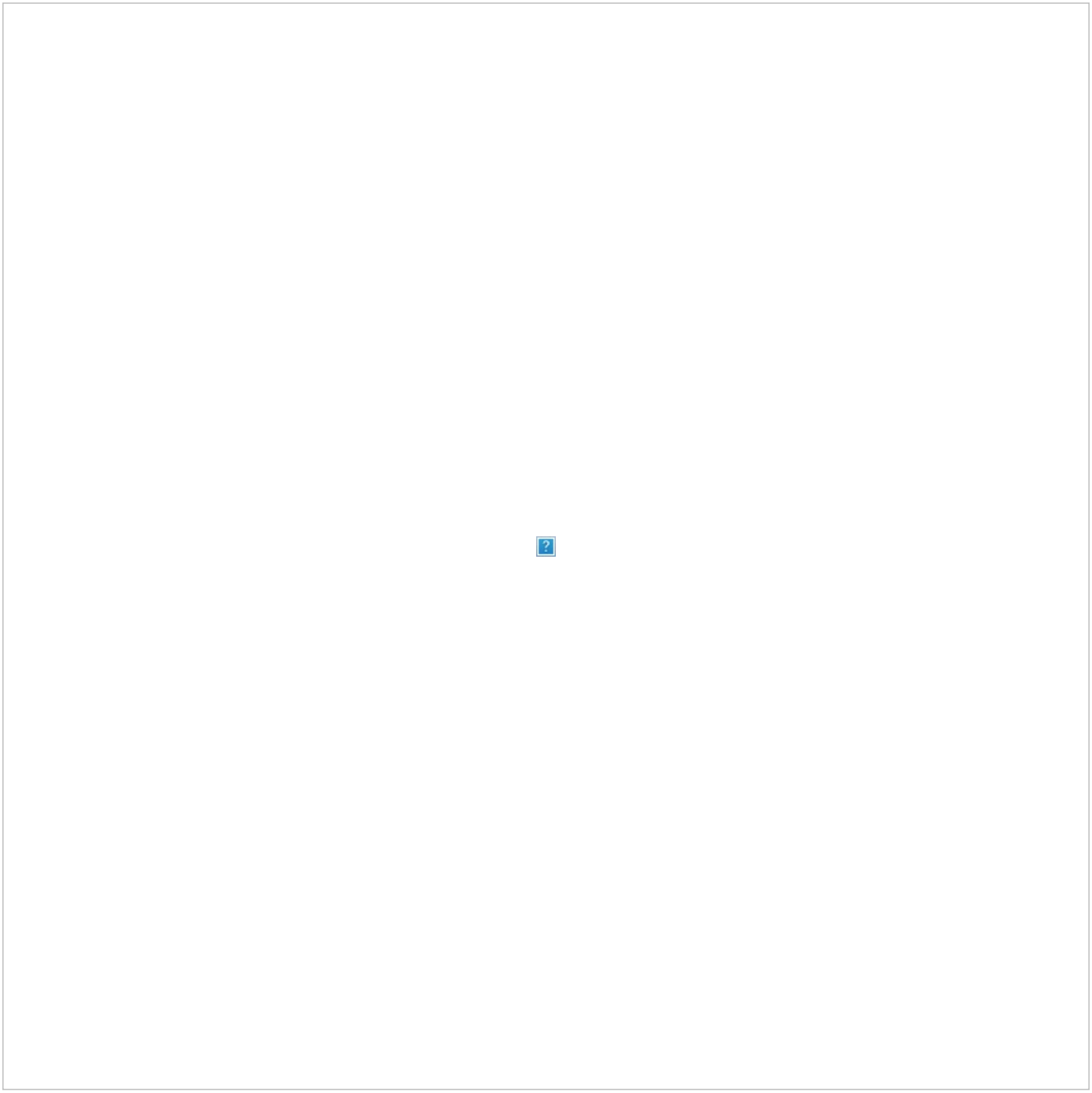
Yukarıdaki grafiği biraz düzenlersek daha güzel bir interaktif grafik elde ederiz. Bu grafiği siz Jupyter Notebook ile deneyerek interaktif olarak zoom yapabilir ve etiket ve diğer bilgileri fare ile üzerine gittiğiniz balonlarda okuyabilirsiniz.

In [12]:

```
import plotly.graph_objects as go
import plotly.express as px
import pandas as pd
import math

# Load data, define hover text and bubble size
data = px.data.gapminder()
df_2007 = data[data['year']==2007]
df_2007 = df_2007.sort_values(['continent', 'country'])

hover_text = []
bubble_size = []

for index, row in df_2007.iterrows():
    hover_text.append(('Country: {country}<br>'+
                       'Life Expectancy: {lifeExp}<br>'+
                       'GDP per capita: {gdp}<br>'+
                       'Population: {pop}<br>'+
                       'Year: {year}').format(country=row['country'],
                                              lifeExp=row['lifeExp'],
                                              gdp=row['gdpPercap'],
                                              pop=row['pop'],
                                              year=row['year']))
    bubble_size.append(math.sqrt(row['pop']))

df_2007['text'] = hover_text
df_2007['size'] = bubble_size
sizeref = 2.*max(df_2007['size'])/(100**2)

# Dictionary with dataframes for each continent
continent_names = ['Africa', 'Americas', 'Asia', 'Europe', 'Oceania']
continent_data = {continent:df_2007.query("continent == '%s'" %continent)
                            for continent in continent_names}

# Create figure
fig = go.Figure()

for continent_name, continent in continent_data.items():
    fig.add_trace(go.Scatter(
        x=continent['gdpPercap'], y=continent['lifeExp'],
        name=continent_name, text=continent['text'],
        marker_size=continent['size'],
        ))

# Tune marker appearance and layout
fig.update_traces(mode='markers', marker=dict(sizemode='area',
                                              sizeref=sizeref, line_width=2))

fig.update_layout(
    title='Life Expectancy v. Per Capita GDP, 2007',
    xaxis=dict(
        title='GDP per capita (2000 dollars)',
        gridcolor='white',
        type='log',
        gridwidth=2,
    ),
    yaxis=dict(
        title='Life Expectancy (years)',
        gridcolor='white',
        gridwidth=2,
    ),
    paper_bgcolor='rgb(243, 243, 243)',
    plot_bgcolor='rgb(243, 243, 243)',
)
fig.show()
```



Yukaridaki grafik bir interaktif grafiktir interaktivite için kodları çalıştırmalısınız.



In [ ]:

```
```

Kullanım Hataları:

Bu grafik türündeki başlıca problem X ve Y eksenindeki ilişkinin bolanun büyüklüğü ile kıyaslandığında daha göz önünde olmasıdır. Bu surumda hangi veri setini daha ön plana çıkarmak istediğinize göre birkaç deneme yapmanızı tavsiye ederim.

Ayrıca balon büyüklüğü ve renklerin ne anlama geldiğini gösteren etiket -Legend- eklemek oldukça faydalı olacaktır.

Büyük boyutlu balonların üste çizilmesi nedeniyle overplot hatası meydana gelebilmektedir, bu hatanın giderimi için ya büyük balonlar arka tarafa çizilecek ve/veya alpha şefaflık kullanılarak diğerlerinin de görünmesi sağlanacaktır. Yukarıdaki grafiklerde her ikisi içinde gerekli örnekler mevcuttur.

In [ ]:

```
```