Инструменты обработки и визуализации данных

Тема №6 – Визуализация в matplotlib II

Настройки

Основные используемые библиотеки следующие:

In [1]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import warnings; warnings.filterwarnings(action='ignore')
```

Здесь `seaborn` – это библиотека для создания статистических графиков, которая построена на основе `matplotlib` и тесно интегрирована со структурами данных `pandas` .

Также будут использованы некоторые другие графические библиотеки.

Гистограммы

Гистограмма — это способ представления распределения частот в наборе числовых данных. Принцип работы заключается в том, что ось X разбивается на интервалы, каждая точка данных в наборе данных присваивается интервалу, а затем подсчитывается количество точек данных, отнесённых к каждому интервалу. Таким образом, ось Y — это частота, или количество точек данных в каждом интервале. Обратите внимание, что можно изменить размер интервала, и часто для корректного отображения распределения требуется его корректировка.

In [2]:
```
# !pip3 install openpyxl
df_can = pd.read_excel(
    'Canada.xlsx',
    sheet_name='Canada by Citizenship',
    skiprows=range(20),
    skipfooter=2,
)
df_can.head()
```

Out[2]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions

5 rows × 43 columns

In [3]:
```
df_can.shape
```

Out[3]:
```
(195, 43)
```

1. Очистим набор данных, удалив столбцы, которые не представляют для нас информации для визуализации (например, Type, AREA, REG).

In [4]:
```
df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inp

df_can.head()
```

Out[4]:

	OdName	AreaName	RegName	DevName	1980	1981	1982	1983	1984	1985	...
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...

5 rows × 38 columns

2. Переименуем некоторые столбцы так, чтобы они имели смысл.

In [5]:
```
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent','

df_can.head()
```

Out[5]:

	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	

5 rows × 38 columns

3. Для обеспечения согласованности убедимся, что все заголовки столбцов имеют строковый тип.

In [6]:
```
all(isinstance(column, str) for column in df_can.columns)
```

Out[6]:
```
False
```

Обратите внимание, что строка кода выше вернула значение False при проверке, все ли заголовки столбцов имеют строковый тип, поэтому изменим их тип на строковый.

In [7]:
```
df_can.columns = list(map(str, df_can.columns))

all(isinstance(column, str) for column in df_can.columns)
```

Out[7]:
```
True
```

4. Установим название страны в качестве индекса — полезно для быстрого поиска стран с помощью метода .loc.

In [8]:
```
df_can.set_index('Country', inplace=True)

df_can.head()
```

Out[8]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	..
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	..
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	..
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	..
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	..
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	..

5 rows × 37 columns

5. Добавим столбец `Total` и переменную `years`.

In [9]:
```
df_can['Total'] = df_can.select_dtypes(include=np.number).sum(axis=

df_can.head()
```

Out[9]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	..
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	..
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	..
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	..
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	..
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	..

5 rows × 38 columns

In [10]:
```
years = list(map(str, range(1980, 2014)))

years
```

Out[10]:
```
['1980',
 '1981',
 '1982',
 '1983',
 '1984',
 '1985',
 '1986',
 '1987',
 '1988',
 '1989',
 '1990',
 '1991',
 '1992',
 '1993',
 '1994',
 '1995',
 '1996',
 '1997',
 '1998',
 '1999',
 '2000',
 '2001',
 '2002',
 '2003',
 '2004',
 '2005',
 '2006',
 '2007',
 '2008',
 '2009',
 '2010',
 '2011',
 '2012',
 '2013']
```

Построим частотное распределение числа (численности) новых иммигрантов из разных стран, прибывших в Канаду в 2013 году.

Прежде чем приступить к построению гистограммы, рассмотрим данные, разбитые на интервалы. Для этого воспользуемся функцией `histogram` из NumPy, чтобы получить диапазоны интервалов и частотные значения следующим образом:

In [11]:
```
df_can.sort_values(['Total'], ascending=False, axis=0, inplace=True

df_can['2013'].head()
```

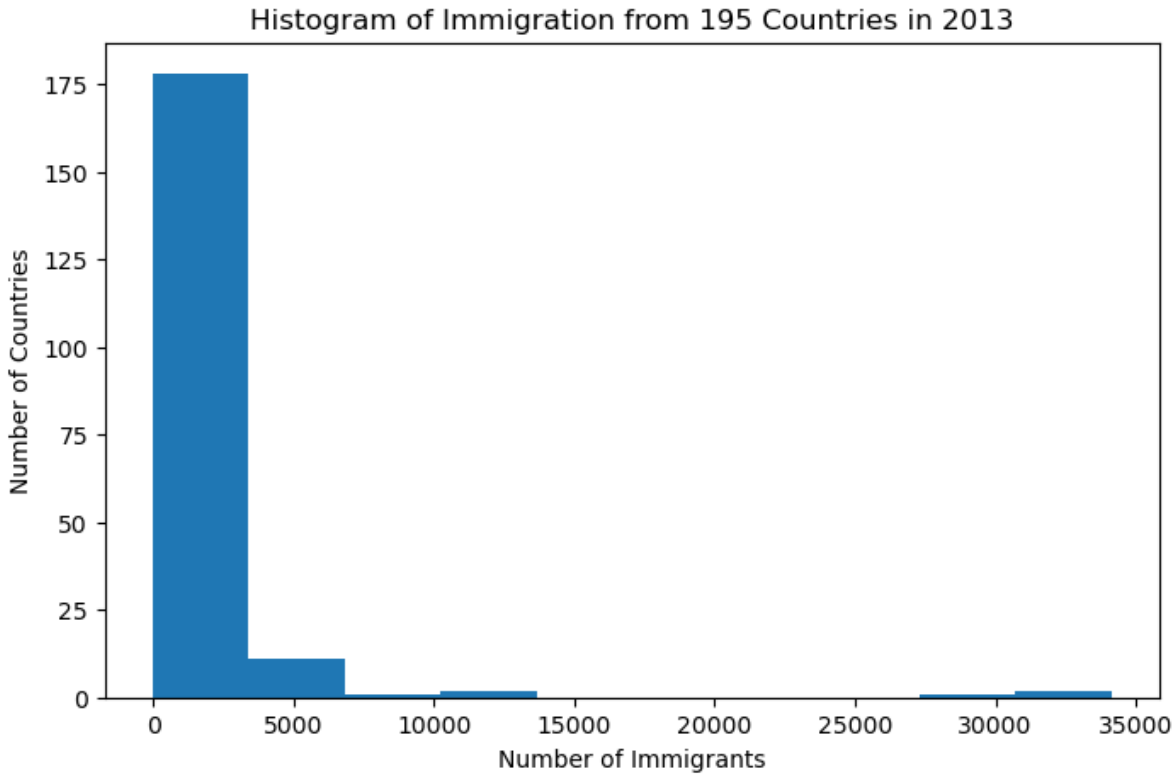
Out[11]:
```
Country
India                                                33087
China                                                34129
United Kingdom of Great Britain and Northern Ireland     5827
Philippines                                          29544
Pakistan                                             12603
Name: 2013, dtype: int64
```

In [12]:
```
# np.histogram returns 2 values
count, bin_edges = np.histogram(df_can['2013'])

print(count) # frequency count
print(bin_edges) # bin ranges, default = 10 bins
```

```
[178  11   1   2   0   0   0   0   1   2]
[    0.   3412.9  6825.8 10238.7 13651.6 17064.5 20477.4 23890.3 2
7303.2
 30716.1 34129. ]
```

По умолчанию функция `histogram` разбивает набор данных на 10 бинов.

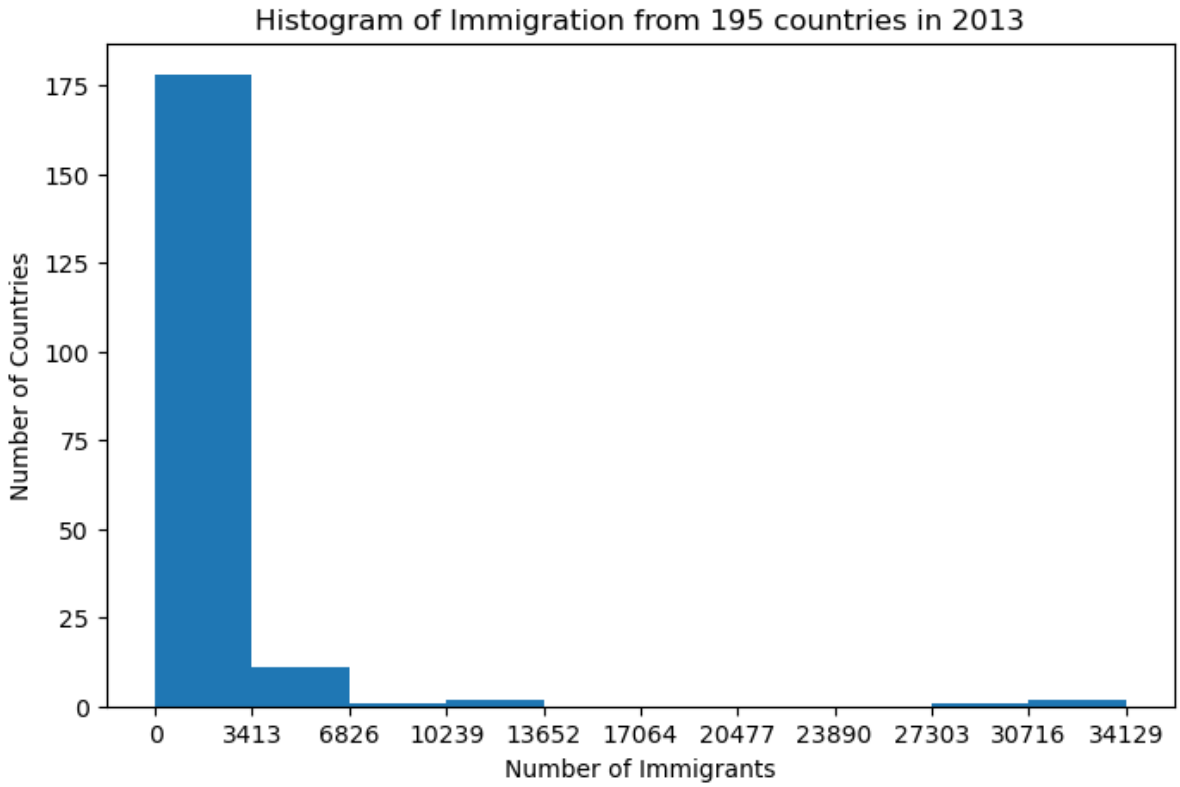
In [13]:
```
df_can['2013'].plot(kind='hist', figsize=(8, 5))

plt.title('Histogram of Immigration from 195 Countries in 2013') #
plt.ylabel('Number of Countries') # add y-label
plt.xlabel('Number of Immigrants') # add x-label

plt.show()
```



На представленном выше графике ось X представляет собой диапазон численности иммигрантов в интервалах от 0 до 34129. Ось Y представляет собой количество стран, внесших вклад в указанную численность.

Обратите внимание, что обозначения на оси X не соответствуют размеру интервала. Это можно исправить, передав ключевое слово `xticks` , содержащее список размеров интервалов, следующим образом:

In [14]:
```
# 'bin_edges' is a list of bin intervals
count, bin_edges = np.histogram(df_can['2013'])

df_can['2013'].plot(kind='hist', figsize=(8, 5), xticks=bin_edges)

plt.title('Histogram of Immigration from 195 countries in 2013') #
plt.ylabel('Number of Countries') # add y-label
plt.xlabel('Number of Immigrants') # add x-label

plt.show()
```



Визуализируем распределение иммиграции из Дании, Норвегии и Швеции в период с 1980 по 2013 год.

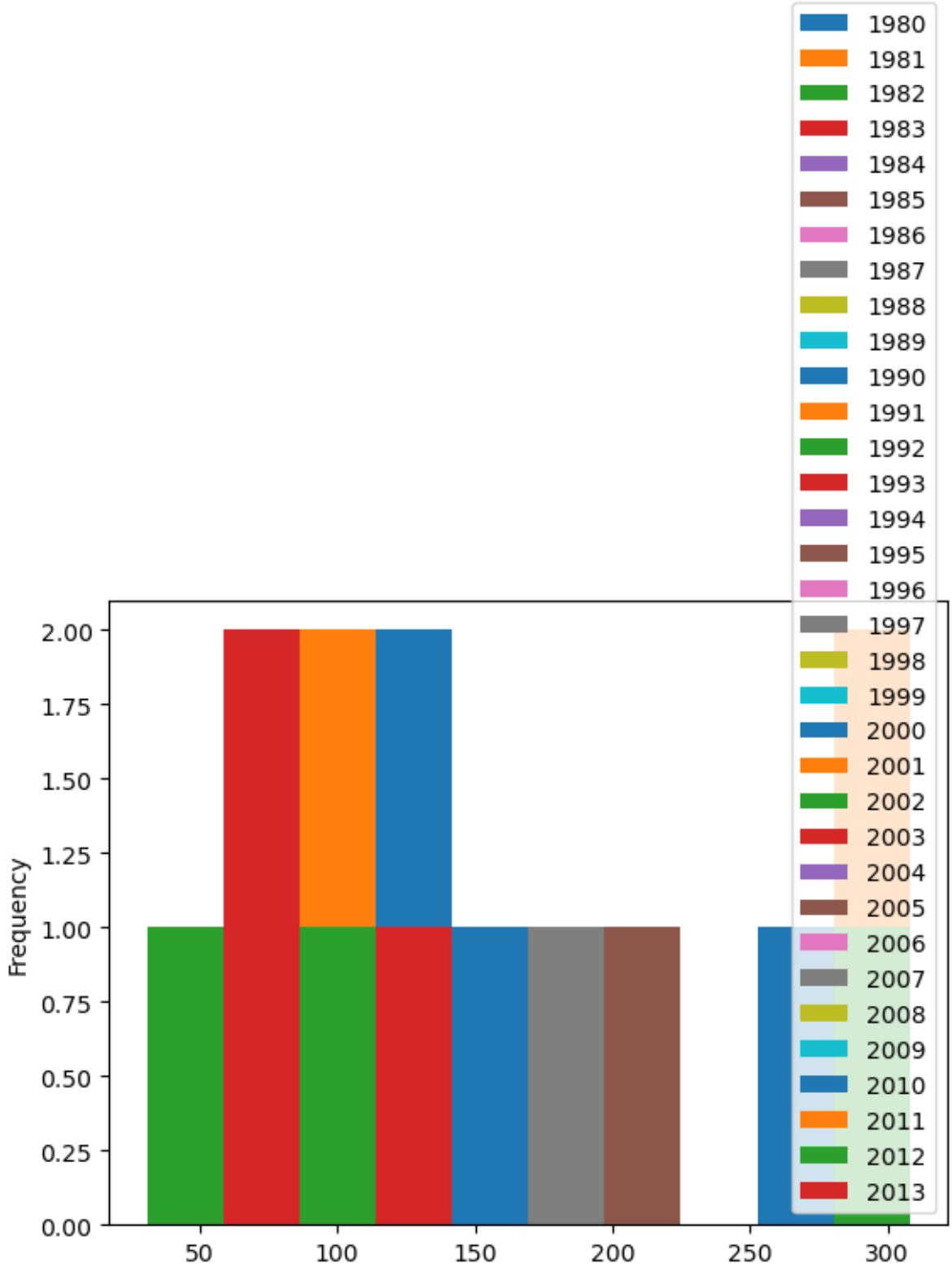
In [15]:
```
# let's quickly view the dataset
df_can.loc[['Denmark', 'Norway', 'Sweden'], years]
```

Out[15]:

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	20
Denmark	272	293	299	106	93	73	93	109	129	129	...	89	62	
Norway	116	77	106	51	31	54	56	80	73	76	...	73	57	
Sweden	281	308	222	176	128	158	187	198	171	182	...	129	205	

3 rows × 34 columns

In [18]:
```
# generate histogram
df_can.loc[['Denmark', 'Norway', 'Sweden'], years].plot.hist()
plt.show()
```



Вместо построения графика распределения численности населения по трём странам, построен график распределения численности населения по годам.

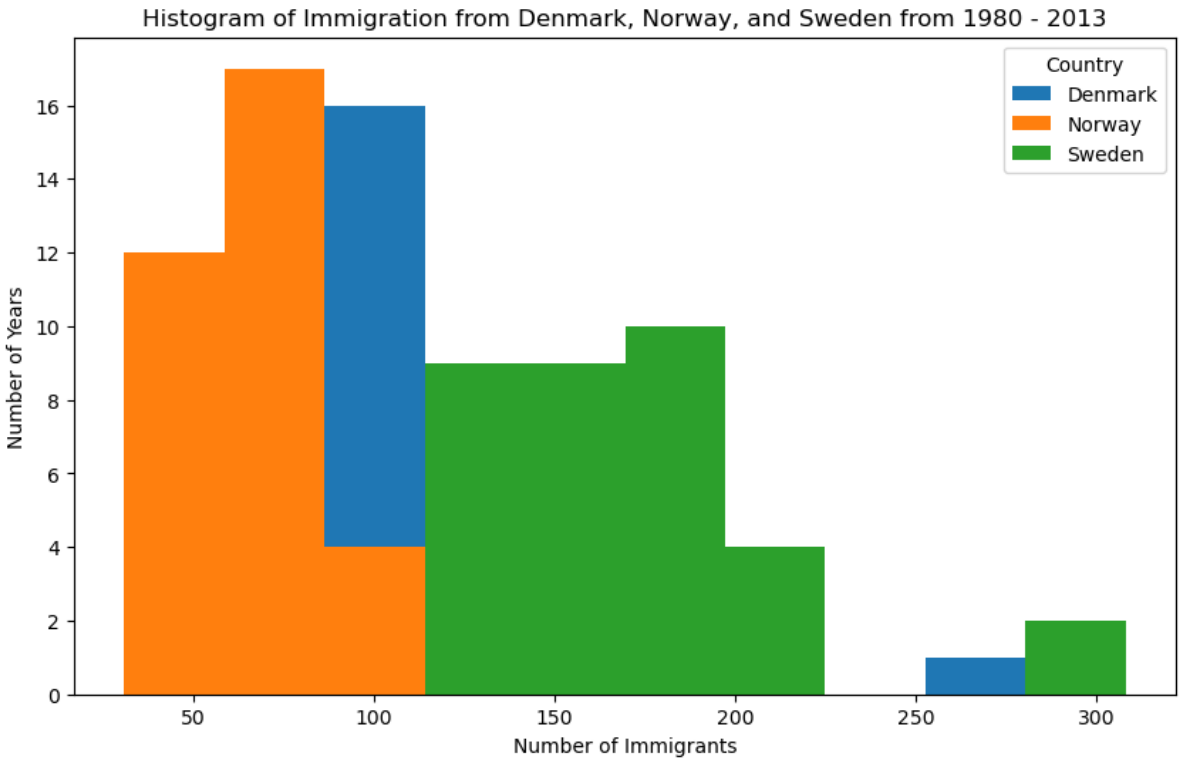
Это можно исправить, сначала транспонировав набор данных, а затем построив график, как показано ниже.

In [19]:
```
# transpose dataframe
df_t = df_can.loc[['Denmark', 'Norway', 'Sweden'], years].transpose
df_t.head()
```

Out[19]:

Country	Denmark	Norway	Sweden
1980	272	116	281
1981	293	77	308
1982	299	106	222
1983	106	51	176
1984	93	31	128

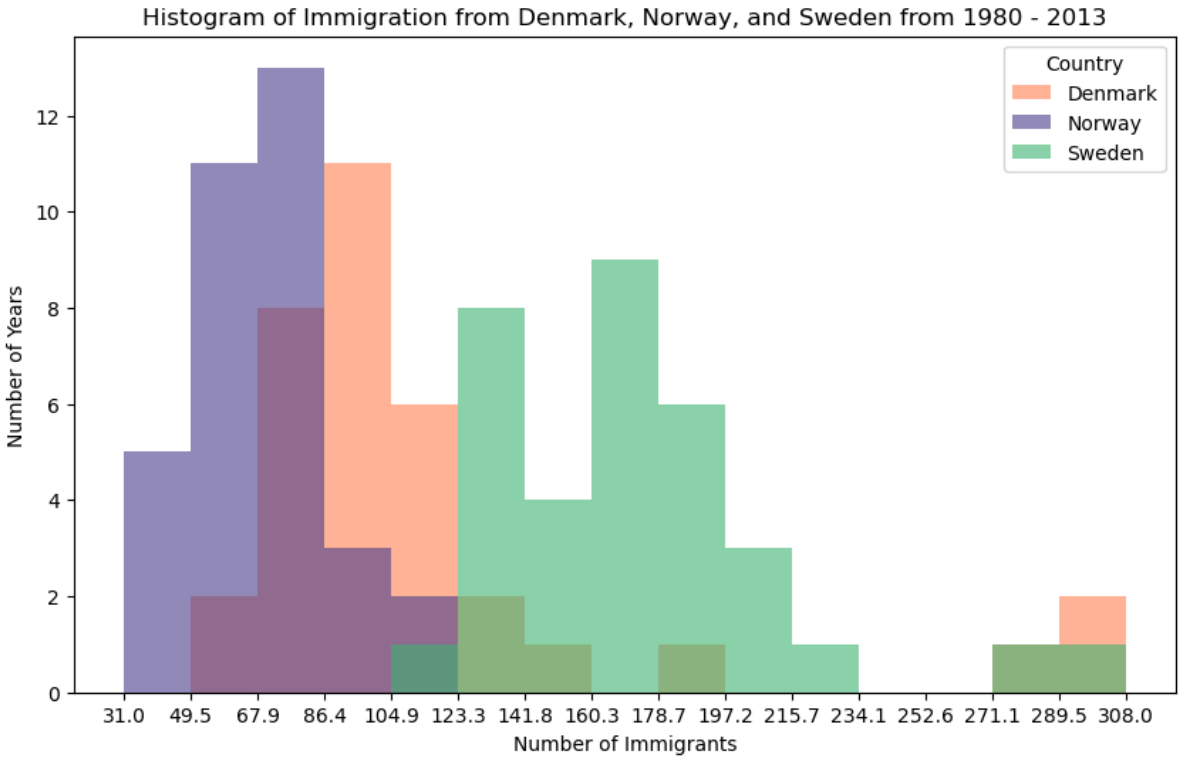
In [20]:
```
# generate histogram
df_t.plot(kind='hist', figsize=(10, 6))

plt.title('Histogram of Immigration from Denmark, Norway, and Swede
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



Внесем несколько изменений, чтобы улучшить эффект и эстетику предыдущего графика:

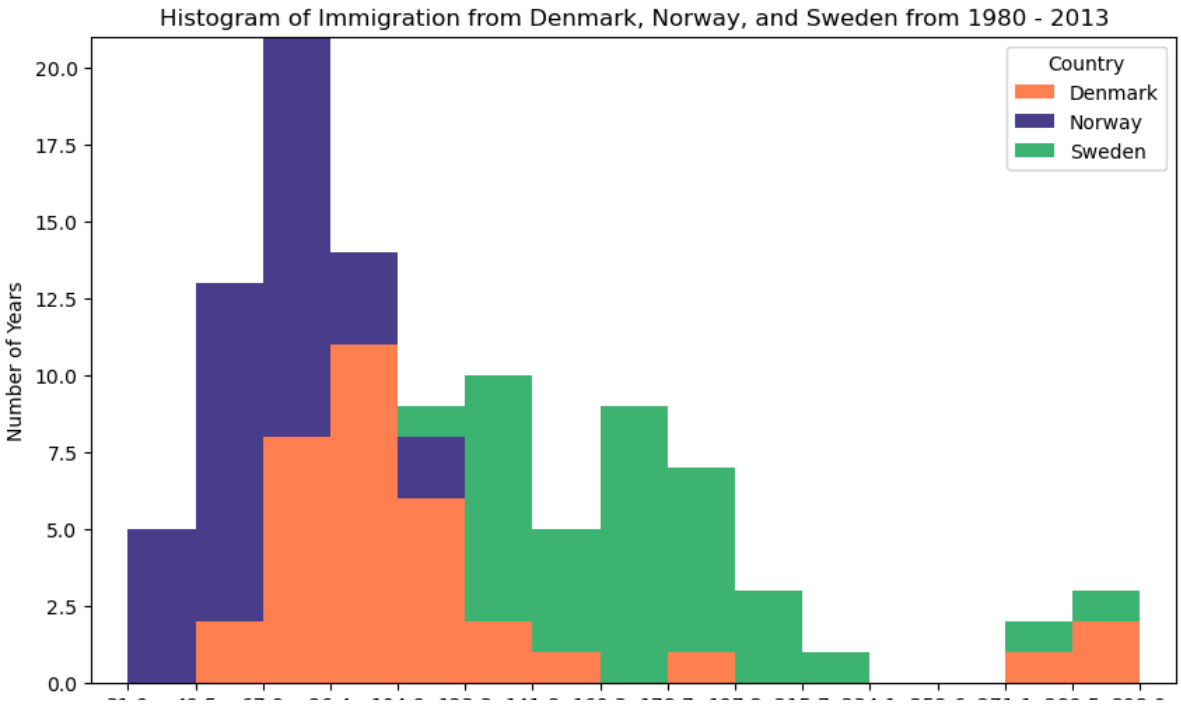
- увеличим кол-во бинов до 15, передав параметр `bins` ;
- установим прозрачность до 60%, передав параметр `alpha` ;
- подпишем ось X, передав параметр `x-label` ;
- изменим цвета графиков, передав параметр `color` .

In [21]:
```
# let's get the x-tick values
count, bin_edges = np.histogram(df_t, 15)

# un-stacked histogram
df_t.plot(kind ='hist',
          figsize=(10, 6),
          bins=15,
          alpha=0.6,
          xticks=bin_edges,
          color=['coral', 'darkslateblue', 'mediumseagreen']
         )

plt.title('Histogram of Immigration from Denmark, Norway, and Swede
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



Если мы не хотим, чтобы диаграммы перекрывали друг друга, мы можем сложить их в стопку, используя параметр `stacked`. Также скорректируем минимальные и максимальные значения по оси X, чтобы убрать лишние зазоры по краям графика. Мы можем передать кортеж `(min, max)` с помощью параметра `xlim`, как показано ниже.

In [22]:
```
count, bin_edges = np.histogram(df_t, 15)
xmin = bin_edges[0] - 10   #  first bin value is 31.0, adding buffe
xmax = bin_edges[-1] + 10  #  last bin value is 308.0, adding buffe

# stacked Histogram
df_t.plot(kind='hist',
          figsize=(10, 6),
          bins=15,
          xticks=bin_edges,
          color=['coral', 'darkslateblue', 'mediumseagreen'],
          stacked=True,
          xlim=(xmin, xmax)
         )

plt.title('Histogram of Immigration from Denmark, Norway, and Swede
plt.ylabel('Number of Years')
plt.xlabel('Number of Immigrants')

plt.show()
```



Для визуализаций далее будут использоваться следующие параметры:

In [23]:
```
large = 22; med = 16; small = 12
params = {'axes.titlesize': large,
          'legend.fontsize': med,
          'figure.figsize': (16, 10),
          'axes.labelsize': med,
          'axes.titlesize': med,
          'xtick.labelsize': med,
          'ytick.labelsize': med,
          'figure.titlesize': large}
plt.rcParams.update(params)
#plt.style.use('seaborn-whitegrid')
sns.set_style('whitegrid')
```

1. Гистограмма для непрерывной переменной

Гистограмма показывает распределение частот непрерывной переменной.

In [24]:
```
# Import Data
df = pd.read_csv("mpg_ggplot2.csv")
df.head()
```

Out[24]:

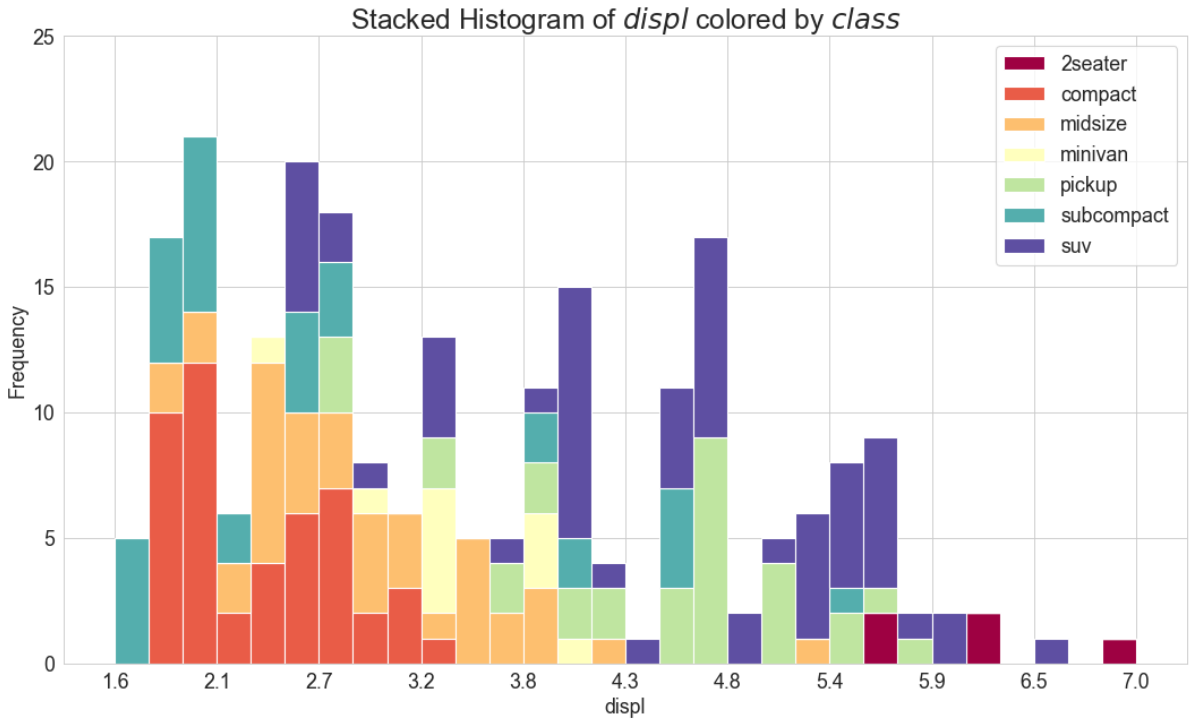
	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact

In [25]:
```
# Prepare data
x_var = 'displ'
groupby_var = 'class'
df_agg = df.loc[:, [x_var, groupby_var]].groupby(groupby_var)
vals = [df[x_var].values.tolist() for i, df in df_agg]
df_agg.head(3)
```

Out[25]:

	displ	class
0	1.8	compact
1	1.8	compact
2	2.0	compact
15	2.8	midsize
16	3.1	midsize
17	4.2	midsize
18	5.3	suv
19	5.3	suv
20	5.3	suv
23	5.7	2seater
24	5.7	2seater
25	6.2	2seater
37	2.4	minivan
38	3.0	minivan
39	3.3	minivan
48	3.7	pickup
49	3.7	pickup
50	3.9	pickup
90	3.8	subcompact
91	3.8	subcompact
92	4.0	subcompact

In [26]:
```
# Draw
plt.figure(figsize=(16,9), dpi= 80)
colors = [
    plt.cm.Spectral(i/float(len(vals)-1)) for i in range(len(vals))
]
n, bins, patches = plt.hist(
    vals, 30, stacked=True, density=False, color=colors[:len(vals)]
)

# Decoration
plt.legend(
    {group:col for group, col in \
     zip(np.unique(df[groupby_var]).tolist(), colors[:len(vals)])}
)
plt.title(
    f"Stacked Histogram of ${x_var}$ colored by ${groupby_var}$",
    fontsize=22
)
plt.xlabel(x_var)
plt.ylabel("Frequency")
plt.ylim(0, 25)
plt.xticks(ticks=bins[::3], labels=[round(b,1) for b in bins[::3]])
plt.show()
```



2. Гистограмма для категориальной переменной

Здесь гистограмма показывает распределение частот категориальной переменной.

In [27]:
```
# Import Data
df = pd.read_csv("mpg_ggplot2.csv")
df.head()
```

Out[27]:

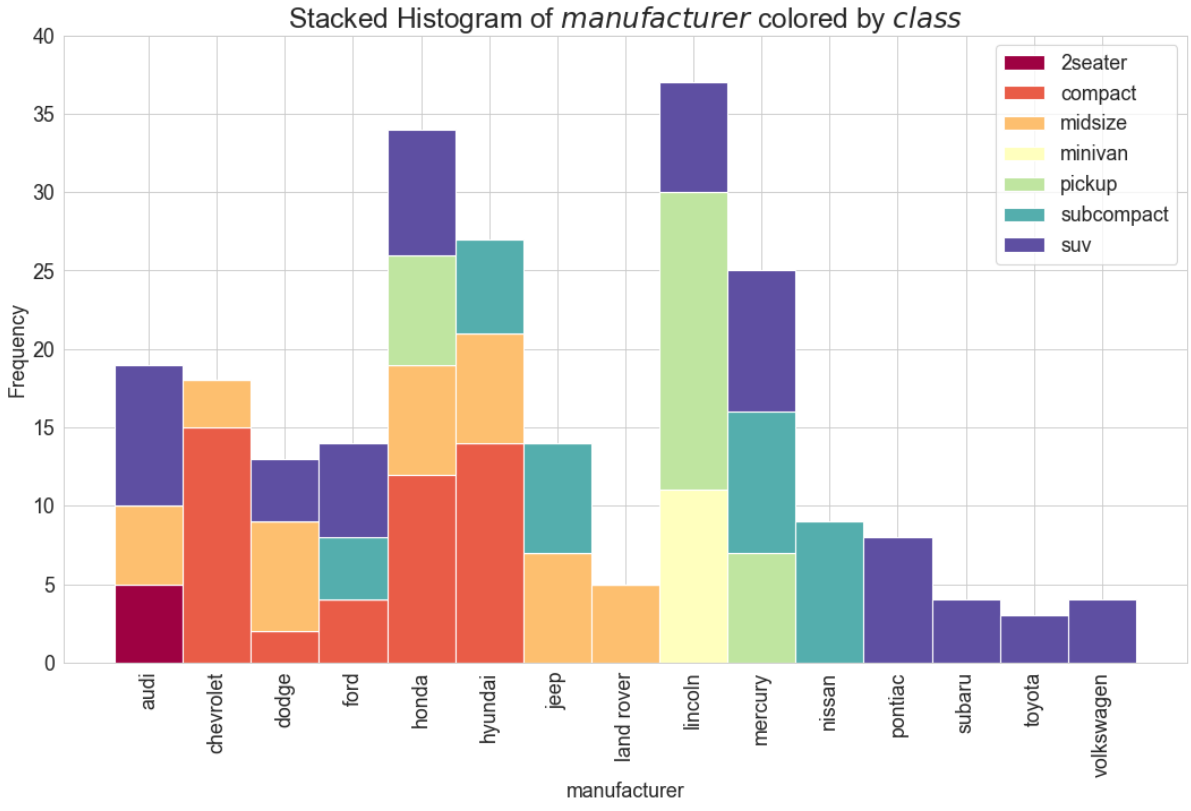
	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact

In [28]:
```
# Prepare data
x_var = 'manufacturer'
groupby_var = 'class'
df_agg = df.loc[:, [x_var, groupby_var]].groupby(groupby_var)
vals = [df[x_var].values.tolist() for i, df in df_agg]
df_agg.head(3)
```

Out[28]:

	manufacturer	class
0	audi	compact
1	audi	compact
2	audi	compact
15	audi	midsize
16	audi	midsize
17	audi	midsize
18	chevrolet	suv
19	chevrolet	suv
20	chevrolet	suv
23	chevrolet	2seater
24	chevrolet	2seater
25	chevrolet	2seater
37	dodge	minivan
38	dodge	minivan
39	dodge	minivan
48	dodge	pickup
49	dodge	pickup
50	dodge	pickup
90	ford	subcompact
91	ford	subcompact
92	ford	subcompact

In [29]:
```
# Draw
plt.figure(figsize=(16,9), dpi= 80)
colors = [
    plt.cm.Spectral(i/float(len(vals)-1)) for i in range(len(vals))
]
_, bins, _ = plt.hist(
    vals, len(df[x_var].unique()),
    stacked=True, density=False, color=colors[:len(vals)]
)
bins_ticks = bins[:-1] + (bins[1:] - bins[:-1])/2

# Decoration
plt.legend(
    {group:col for group, col in
    zip(np.unique(df[groupby_var]).tolist(), colors[:len(vals)])}
)
plt.title(
    f"Stacked Histogram of ${x_var}$ colored by ${groupby_var}$",
    fontsize=22
)
plt.xlabel(x_var)
plt.ylabel("Frequency")
plt.ylim(0, 40)
plt.xticks(
    ticks=bins_ticks, labels=np.unique(df[x_var]).tolist(),
    rotation=90, horizontalalignment='center'
)
plt.show()
```



Столбчатые диаграммы (Bar Charts)

Столбчатая диаграмма — это способ представления данных, в котором длина столбцов соответствует величине/размеру признака/переменной. Столбчатые диаграммы обычно представляют числовые и категориальные переменные, сгруппированные в интервалы.

Для создания столбчатой диаграммы можно передать один из двух аргументов через параметр `kind` в функцию `plot()`:

- `kind=bar` создаёт вертикальную столбчатую диаграмму
- `kind=barh` создаёт горизонтальную столбчатую диаграмму

Вертикальная столбчатая диаграмма

В вертикальных столбчатых диаграммах ось X используется для маркировки, а длина столбцов на оси Y соответствует величине измеряемой переменной. Вертикальные столбчатые диаграммы особенно полезны для анализа временных рядов. Одним из их недостатков является отсутствие места для текстовой маркировки внизу каждого столбца.

Проанализируем последствия финансового кризиса в Исландии:

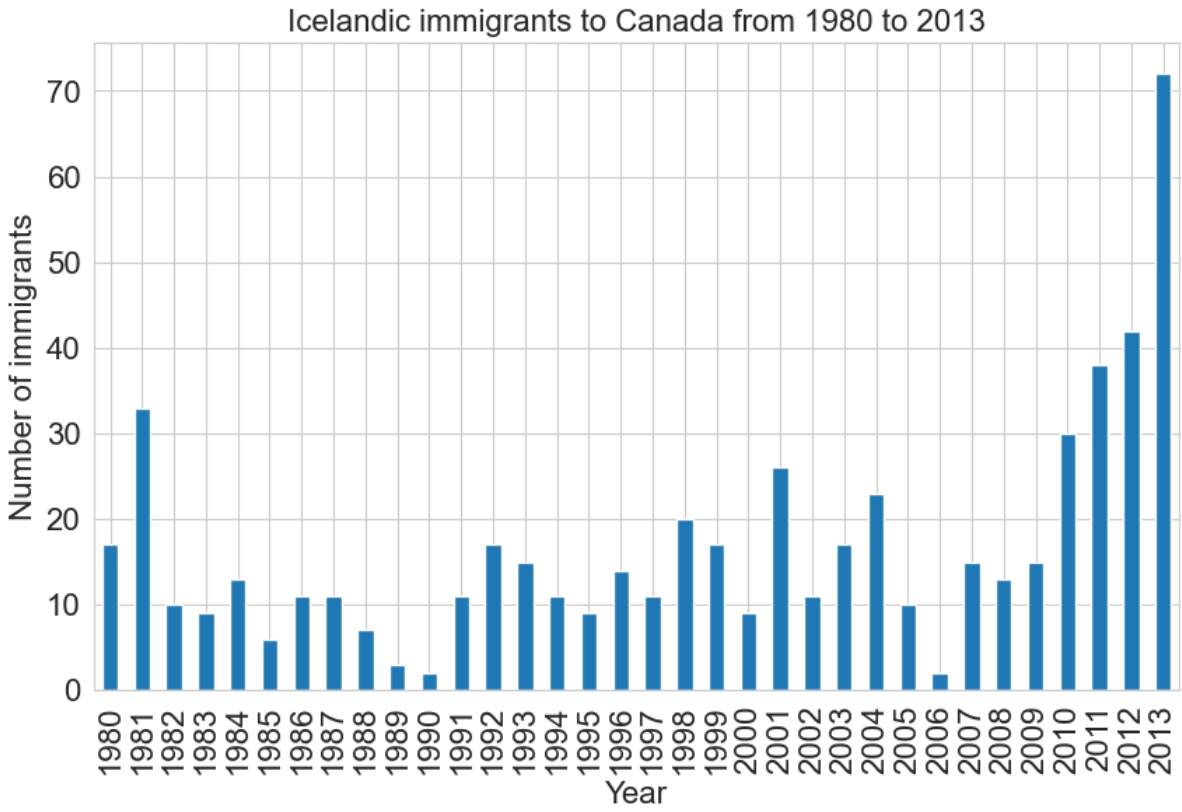
Исландский финансовый кризис 2008–2011 годов стал крупным экономическим и политическим событием в Исландии. По сравнению с размером экономики страны системный банковский крах Исландии стал крупнейшим в истории экономики страны. Кризис привёл к глубокой экономической депрессии 2008–2011 годов и серьёзным политическим волнениям.

Сравним количество исландских иммигрантов (страна — «Исландия») в Канаде с 1980 по 2013 год.

In [30]:
```
# step 1: get the data
df_iceland = df_can.loc['Iceland', years]
df_iceland.head()
```

Out[30]:
```
1980    17
1981    33
1982    10
1983     9
1984    13
Name: Iceland, dtype: object
```

In [31]:
```
# step 2: plot data
df_iceland.plot(kind='bar', figsize=(10, 6))

plt.xlabel('Year') # add to x-label to the plot
plt.ylabel('Number of immigrants') # add y-label to the plot
plt.title('Icelandic immigrants to Canada from 1980 to 2013') # add

plt.show()
```

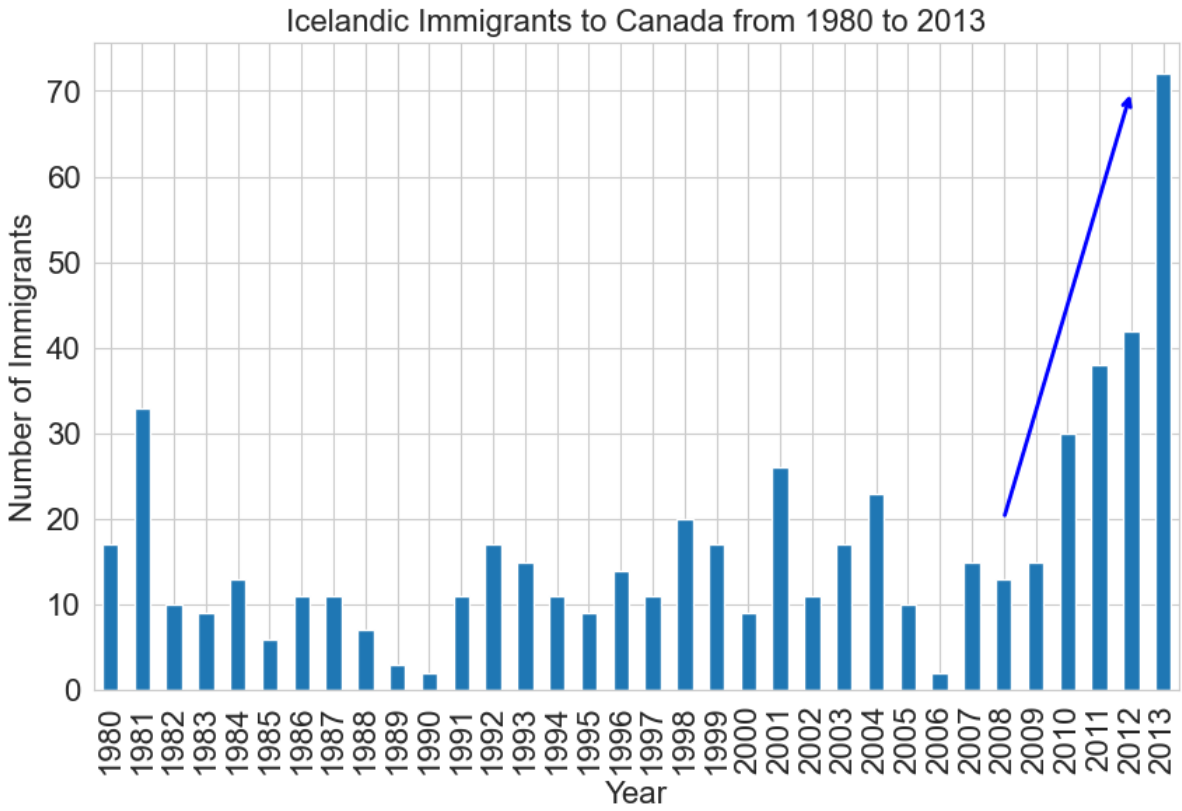


Столбчатая диаграмма выше показывает общее число иммигрантов по годам. Мы ясно видим влияние финансового кризиса: число иммигрантов в Канаде начало быстро расти после 2008 года.

Добавим аннотацию на график, используя функцию `annotate` интерфейса pyplot. Мы передадим следующие параметры:

- s: str, текст аннотации.
- xy: Кортеж, задающий точку (x,y) для аннотации (в данном случае — конечную точку стрелки).
- xytext: Кортеж, задающий точку (x,y) для размещения текста (в данном случае — начальную точку стрелки).
- xycoords: Система координат, в которой задан xy — 'data' использует систему координат аннотируемого объекта (по умолчанию).
- arrowprops: Принимает словарь свойств для отрисовки стрелки:
  - arrowstyle: Задает стиль стрелки, '->' — стандартная стрелка.
  - connectionstyle: Указывает тип соединения. arc3 — прямая линия.
  - color: Указывает цвет дуги.
  - lw: Указывает толщину линии.

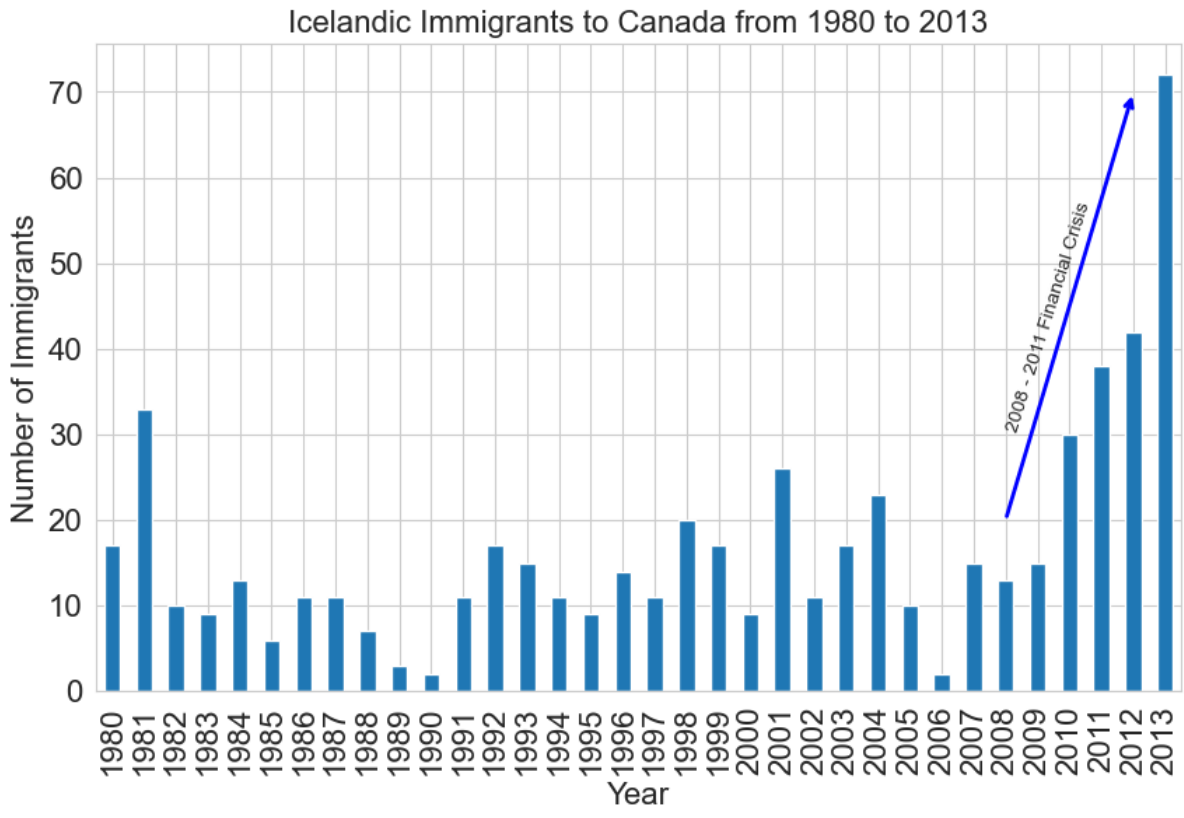
In [32]:
```
df_iceland.plot(kind='bar', figsize=(10, 6), rot=90) # rotate the b

plt.xlabel('Year')
plt.ylabel('Number of Immigrants')
plt.title('Icelandic Immigrants to Canada from 1980 to 2013')

# Annotate arrow
plt.annotate('',                      # s: str. Will leave it blank
             xy=(32, 70),             # place head of the arrow at
             xytext=(28, 20),         # place base of the arrow at
             xycoords='data',         # will use the coordinate sys
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3
            )

plt.show()
```



Также добавим текст, который будет располагаться над стрелкой. Передадим следующие дополнительные параметры:

- rotation: угол поворота текста в градусах (против часовой стрелки);
- va: вертикальное выравнивание текста `['center' | 'top' | 'bottom' | 'baseline']`;
- ha: горизонтальное выравнивание текста `['center' | 'right' | 'left']`;

In [33]:
```
df_iceland.plot(kind='bar', figsize=(10, 6), rot=90)

plt.xlabel('Year')
plt.ylabel('Number of Immigrants')
plt.title('Icelandic Immigrants to Canada from 1980 to 2013')

# Annotate arrow
plt.annotate('',                      # s: str. will leave it blank
             xy=(32, 70),             # place head of the arrow at
             xytext=(28, 20),         # place base of the arrow at
             xycoords='data',         # will use the coordinate sys
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3
            )

# Annotate Text
plt.annotate('2008 - 2011 Financial Crisis', # text to display
             xy=(28, 30),                    # start the text at at
             rotation=72.5,                  # based on trial and e
             va='bottom',                    # want the text to be
             ha='left',                      # want the text to be
            )

plt.show()
```



Горизонтальная столбчатая диаграмма

Иногда удобнее представлять данные горизонтально, особенно если нужно больше места для подписей столбцов. В горизонтальных столбчатых диаграммах ось Y используется для подписей, а длина столбцов по оси X соответствует величине измеряемой переменной. Как вы увидите, на оси Y больше места для подписей категориальных переменных.

Создадим горизонтальную столбчатую диаграмму, показывающую общее число иммигрантов в Канаде из 15 стран с наибольшим числом иммигрантов за период с 1980 по 2013 год. Укажем для каждой страны общее число иммигрантов.

Шаг 1: Получим данные по 15 странам с наибольшим числом иммигрантов.

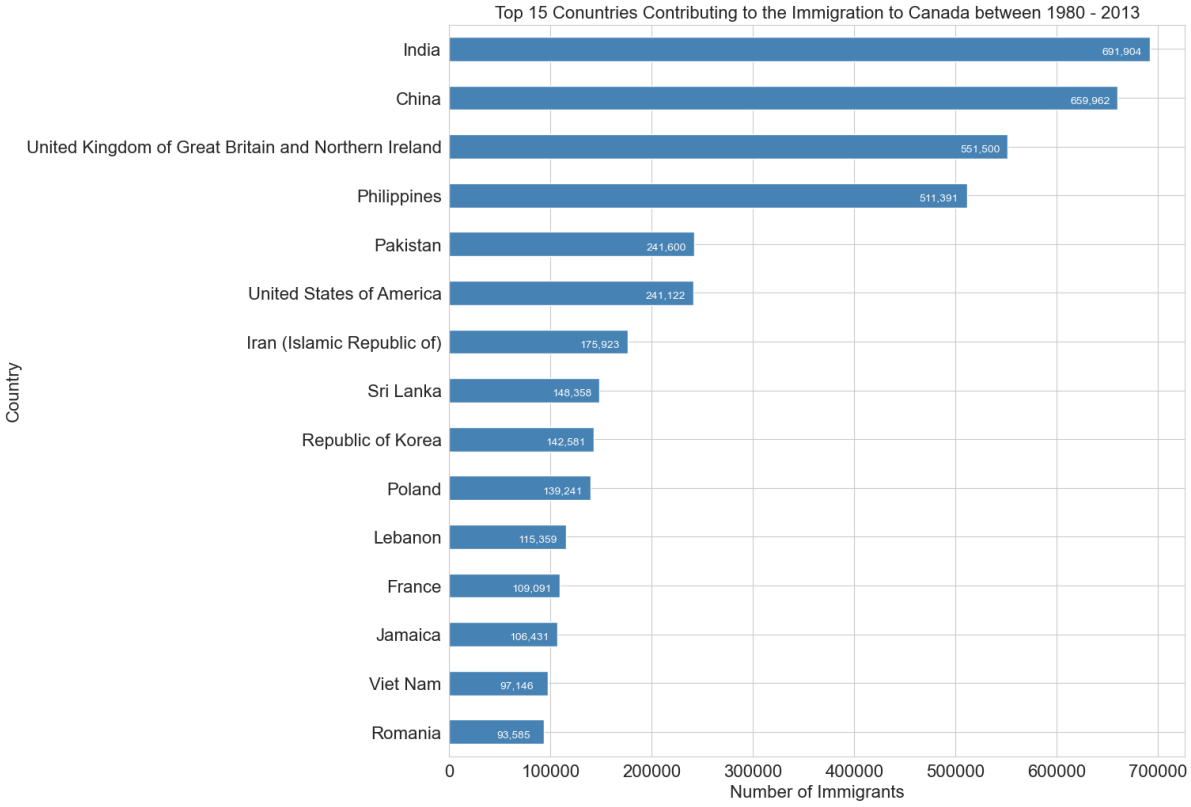
In [34]:
```
df_can.sort_values(by='Total', ascending=True, inplace=True)

df_top15 = df_can['Total'].tail(15)

df_top15
```

Out[34]:
```
Country
Romania                                                 93585
Viet Nam                                                97146
Jamaica                                                106431
France                                                 109091
Lebanon                                                115359
Poland                                                 139241
Republic of Korea                                      142581
Sri Lanka                                              148358
Iran (Islamic Republic of)                             175923
United States of America                               241122
Pakistan                                               241600
Philippines                                            511391
United Kingdom of Great Britain and Northern Ireland   551500
China                                                  659962
India                                                  691904
Name: Total, dtype: int64
```

Шаг 2: Отобразим данные на графике:

1. Используем `kind='barh'` для создания столбчатой диаграммы с горизонтальными полосами.
2. Убедимся, что выбрали подходящий размер графика, маркировали оси и дали диаграмме название.
3. Пройдемся по странам и добавим аннотации к данным о численности иммигрантов, используя функцию `anotate` .

In [35]:
```
df_top15.plot(kind='barh', figsize=(12, 12), color='steelblue')
plt.xlabel('Number of Immigrants')
plt.title('Top 15 Conuntries Contributing to the Immigration to Can

for index, value in enumerate(df_top15):
    label = format(int(value), ',') # format int with commas

    # place text at the end of bar (subtracting 47000 from x, and 0
    plt.annotate(label, xy=(value - 47000, index - 0.10), color='wh

plt.show()
```



1. Столбчатая диаграмма (Bar Chart)

Столбчатая диаграмма — это классический способ визуализации элементов на основе количества или любой заданной метрики.

In [36]:
```
import random

# Import Data
df_raw = pd.read_csv("mpg_ggplot2.csv")
df_raw.head()
```

Out[36]:

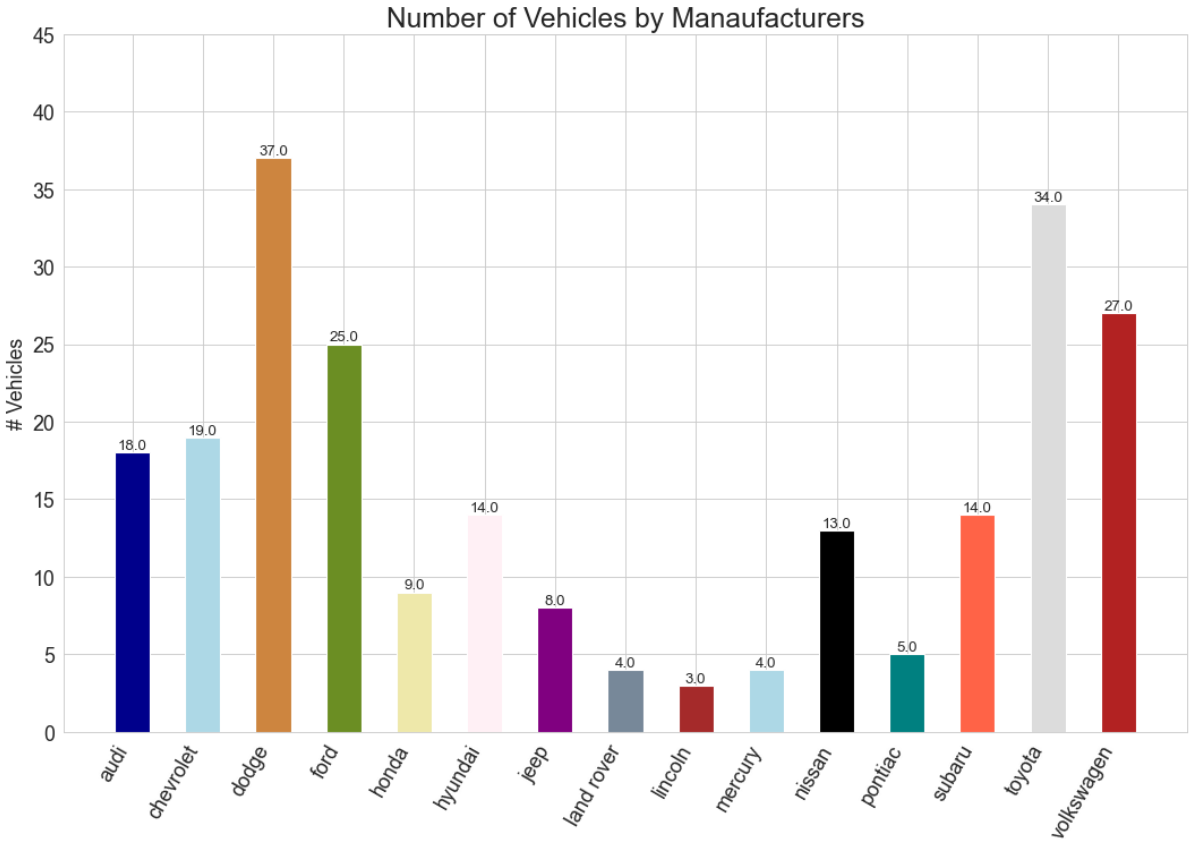
	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
0	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
1	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
2	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
3	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
4	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact

In [37]:
```
# Prepare Data
df = df_raw.groupby('manufacturer').size().reset_index(name='counts
n = len(df['manufacturer'].unique())+1
all_colors = list(plt.cm.colors.cnames.keys())
random.seed(100)
c = random.choices(all_colors, k=n)
df.head()
```

Out[37]:

	manufacturer	counts
0	audi	18
1	chevrolet	19
2	dodge	37
3	ford	25
4	honda	9

In [38]:
```
# Plot Bars
plt.figure(figsize=(16,10), dpi= 80)
plt.bar(df['manufacturer'], df['counts'], color=c, width=.5)
for i, val in enumerate(df['counts'].values):
    plt.text(
        i, val, float(val), horizontalalignment='center',
        verticalalignment='bottom',
        fontdict={'fontweight':500, 'size':12}
    )

# Decoration
plt.gca().set_xticklabels(
    df['manufacturer'], rotation=60, horizontalalignment= 'right'
)
plt.title("Number of Vehicles by Manaufacturers", fontsize=22)
plt.ylabel('# Vehicles')
plt.ylim(0, 45)
plt.show()
```



2. Расходящиеся полоски (diverging bars)

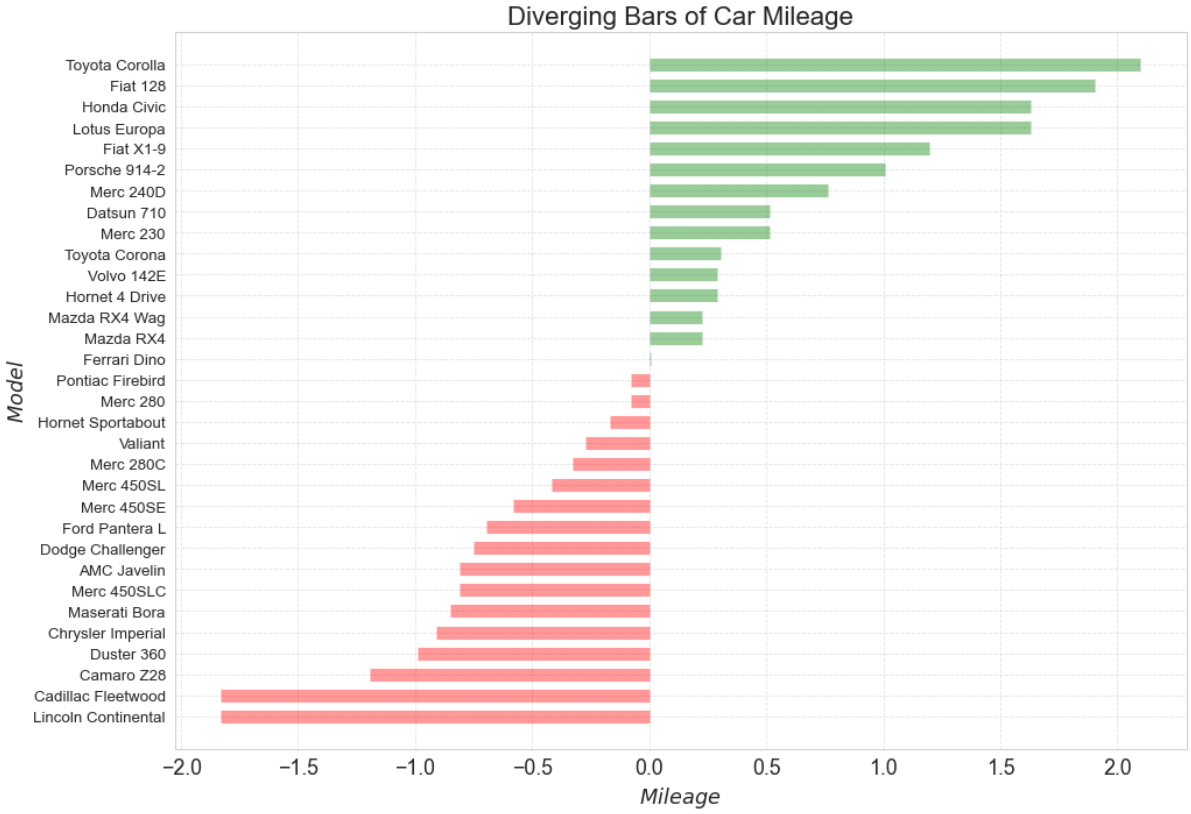
Диаграммы "расходящиеся полоски" позволяют визуализировать порядок и величину отклонения данных.

In [39]:
```
# Prepare Data
df = pd.read_csv("mtcars.csv")
x = df.loc[:, ['mpg']]
df['mpg_z'] = (x - x.mean())/x.std()
df['colors'] = ['red' if x < 0 else 'green' for x in df['mpg_z']]
df.sort_values('mpg_z', inplace=True)
df.reset_index(inplace=True)
df.head()
```

Out[39]:

	index	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	fast	ca
0	15	3.224903	8	460.0	215	3.00	5.424	17.82	0	0	3	4	0	Linco Continent
1	14	3.224903	8	472.0	205	2.93	5.250	17.98	0	0	3	4	0	Cadill Fleetwoo
2	23	3.646917	8	350.0	245	3.73	3.840	15.41	0	0	3	4	0	Cama Z2
3	6	3.781534	8	360.0	245	3.21	3.570	15.84	0	0	3	4	0	Duster 36
4	16	3.834058	8	440.0	230	3.23	5.345	17.42	0	0	3	4	0	Chrysl Imperi

In [40]:
```
# Draw plot
plt.figure(figsize=(14,10), dpi= 80)
plt.hlines(
    y=df.index, xmin=0, xmax=df.mpg_z,
    color=df.colors, alpha=0.4, linewidth=10
)

# Decorations
plt.gca().set(ylabel='$Model$', xlabel='$Mileage$')
plt.yticks(df.index, df.cars, fontsize=12)
plt.title('Diverging Bars of Car Mileage', fontdict={'size':20})
plt.grid(linestyle='--', alpha=0.5)
plt.show()
```



3. Расходящиеся полоски с текстом (diverging texts)

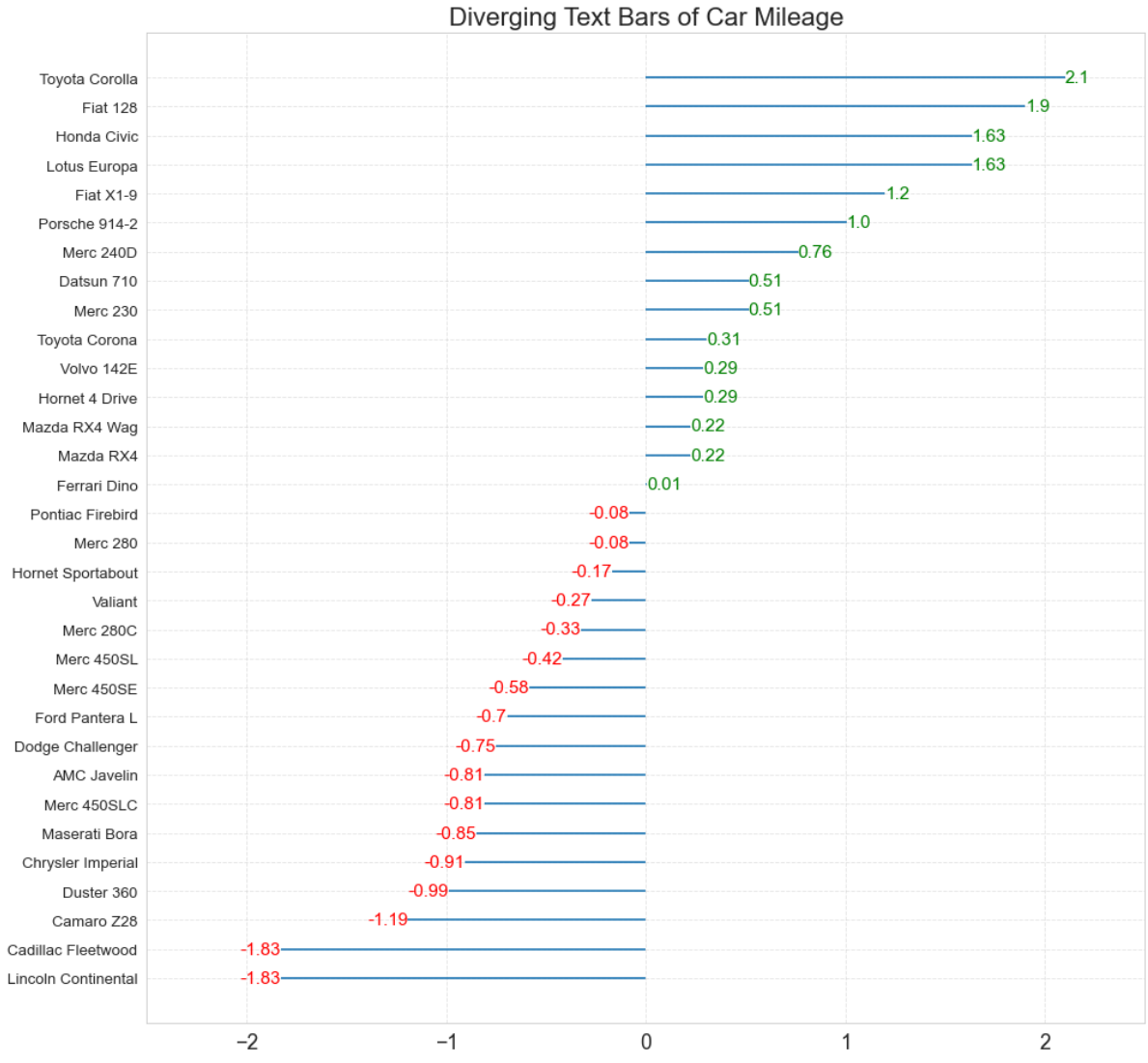
Диаграммы "расходящиеся полоски с текстом" показывают значимость каждого элемента в диаграмме в хорошем и презентабельном виде.

In [41]:
```
# Prepare Data
df = pd.read_csv("mtcars.csv")
x = df.loc[:, ['mpg']]
df['mpg_z'] = (x - x.mean())/x.std()
df['colors'] = ['red' if x < 0 else 'green' for x in df['mpg_z']]
df.sort_values('mpg_z', inplace=True)
df.reset_index(inplace=True)
df.head()
```

Out[41]:

	index	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	fast	ca
0	15	3.224903	8	460.0	215	3.00	5.424	17.82	0	0	3	4	0	Linco Continent
1	14	3.224903	8	472.0	205	2.93	5.250	17.98	0	0	3	4	0	Cadill Fleetwoo
2	23	3.646917	8	350.0	245	3.73	3.840	15.41	0	0	3	4	0	Cama Z2
3	6	3.781534	8	360.0	245	3.21	3.570	15.84	0	0	3	4	0	Duster 36
4	16	3.834058	8	440.0	230	3.23	5.345	17.42	0	0	3	4	0	Chrysl Imperi

In [42]:
```
# Draw plot
plt.figure(figsize=(14,14), dpi= 80)
plt.hlines(y=df.index, xmin=0, xmax=df.mpg_z)
for x, y, tex in zip(df.mpg_z, df.index, df.mpg_z):
    plt.text(
        x, y, round(tex, 2),
        horizontalalignment='right' if x < 0 else 'left',
        verticalalignment='center',
        fontdict={'color':'red' if x < 0 else 'green', 'size':14}
    )

# Decorations
plt.yticks(df.index, df.cars, fontsize=12)
plt.title('Diverging Text Bars of Car Mileage', fontdict={'size':20
plt.grid(linestyle='--', alpha=0.5)
plt.xlim(-2.5, 2.5)
plt.show()
```



4. Расходящийся точечный график (Diverging Dot Plot)

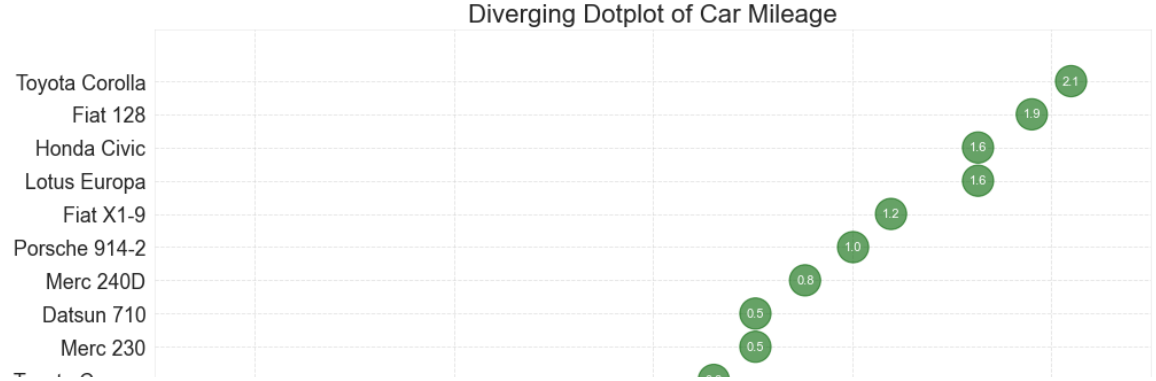
Если сравнивать с расходящимися полосками, то отсутствие столбцов уменьшает степень контрастности и несоответствия между группами.

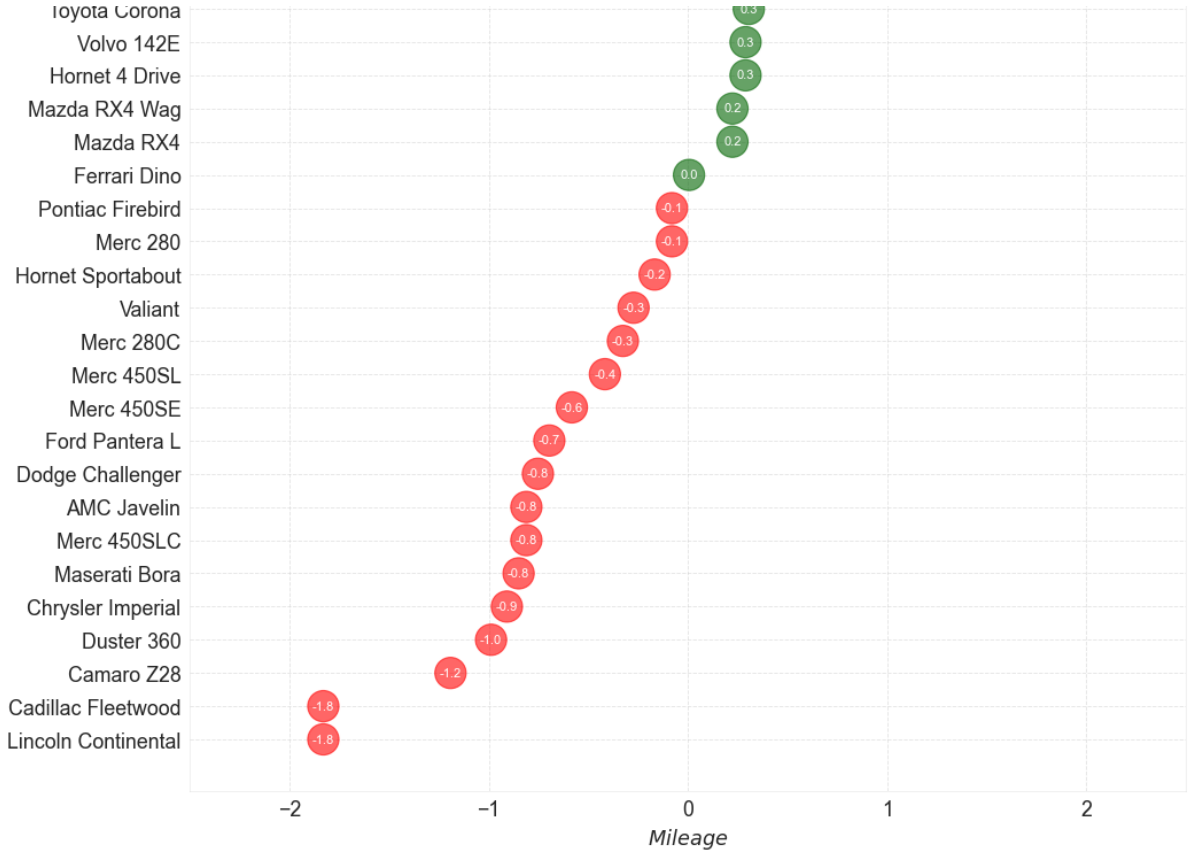
In [43]:
```
# Prepare Data
df = pd.read_csv("mtcars.csv")
x = df.loc[:, ['mpg']]
df['mpg_z'] = (x - x.mean())/x.std()
df['colors'] = ['red' if x < 0 else 'darkgreen' for x in df['mpg_z'
df.sort_values('mpg_z', inplace=True)
df.reset_index(inplace=True)
df.head()
```

Out[43]:

	index	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	fast	ca
0	15	3.224903	8	460.0	215	3.00	5.424	17.82	0	0	3	4	0	Linco Continent
1	14	3.224903	8	472.0	205	2.93	5.250	17.98	0	0	3	4	0	Cadill Fleetwoo
2	23	3.646917	8	350.0	245	3.73	3.840	15.41	0	0	3	4	0	Cama Z2
3	6	3.781534	8	360.0	245	3.21	3.570	15.84	0	0	3	4	0	Duster 36
4	16	3.834058	8	440.0	230	3.23	5.345	17.42	0	0	3	4	0	Chrysl Imperi

In [44]:
```
# Draw plot
plt.figure(figsize=(14,16), dpi= 80)
plt.scatter(df.mpg_z, df.index, s=600, alpha=.6, color=df.colors)
for x, y, tex in zip(df.mpg_z, df.index, df.mpg_z):
    plt.text(
        x, y, round(tex, 1), horizontalalignment='center',
        verticalalignment='center', fontdict={'color':'white'}
    )

# Decorations
# Lighten borders
plt.gca().spines["top"].set_alpha(.3)
plt.gca().spines["bottom"].set_alpha(.3)
plt.gca().spines["right"].set_alpha(.3)
plt.gca().spines["left"].set_alpha(.3)

plt.yticks(df.index, df.cars)
plt.title('Diverging Dotplot of Car Mileage', fontdict={'size':20})
plt.xlabel('$Mileage$')
plt.grid(linestyle='--', alpha=0.5)
plt.xlim(-2.5, 2.5)
plt.show()
```





5. Расходящаяся диаграмма леденцы с маркерами (Diverging Lollipop Chart with Markers)

Диаграмма "леденцы" (lollipop) обеспечивает гибкий способ визуализации отклонения, делая акцент на любых значимых точках данных, на которые вы хотите обратить внимание.

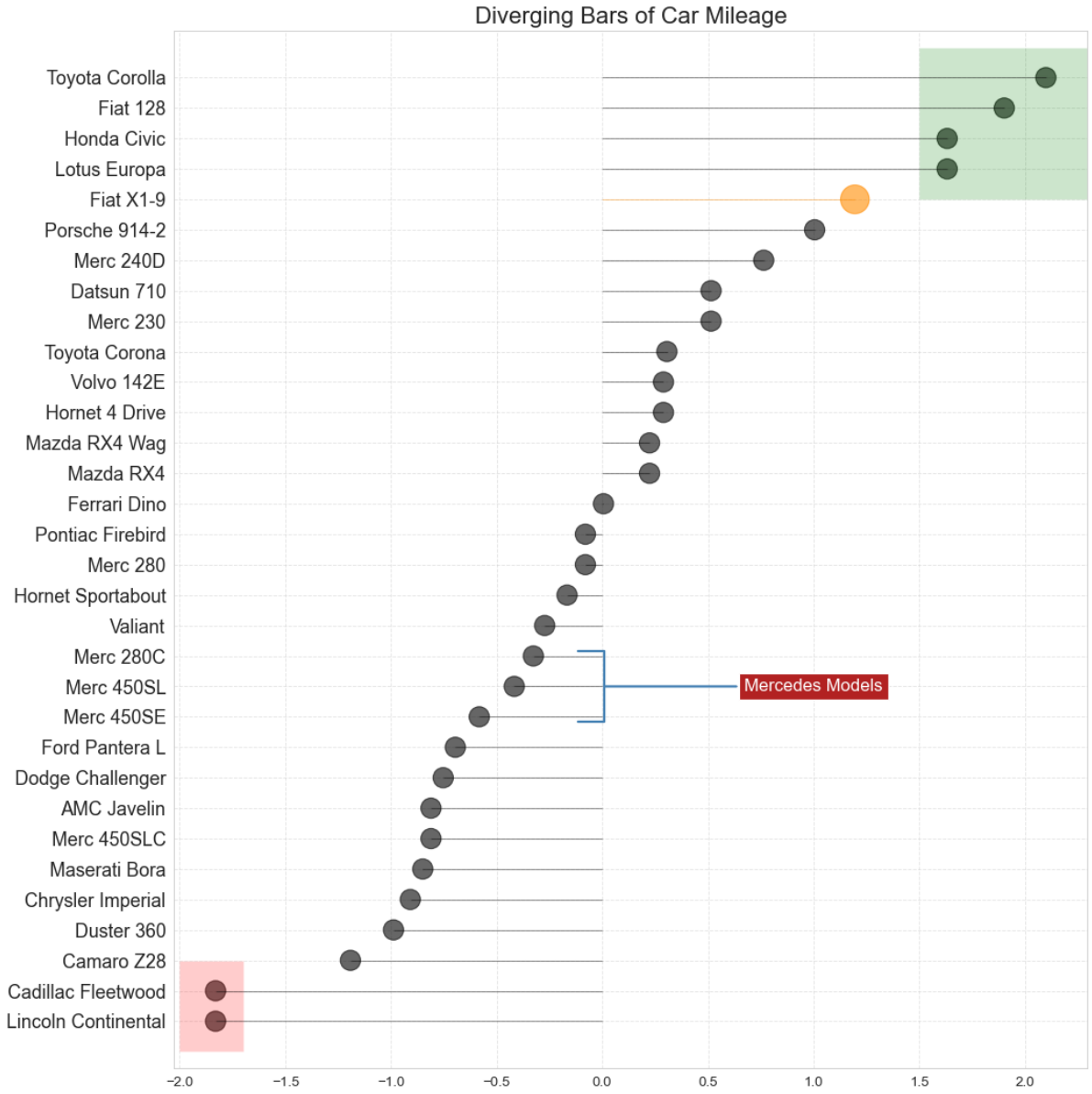
In [45]:
```
# Prepare Data
df = pd.read_csv("mtcars.csv")
x = df.loc[:, ['mpg']]
df['mpg_z'] = (x - x.mean())/x.std()
df['colors'] = 'black'

# color fiat differently
df.loc[df.cars == 'Fiat X1-9', 'colors'] = 'darkorange'

df.sort_values('mpg_z', inplace=True)
df.reset_index(inplace=True)
df.head()
```

Out[45]:

	index	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	fast	ca
0	15	3.224903	8	460.0	215	3.00	5.424	17.82	0	0	3	4	0	Linco Continent
1	14	3.224903	8	472.0	205	2.93	5.250	17.98	0	0	3	4	0	Cadill Fleetwoo
2	23	3.646917	8	350.0	245	3.73	3.840	15.41	0	0	3	4	0	Cama Z2
3	6	3.781534	8	360.0	245	3.21	3.570	15.84	0	0	3	4	0	Duster 36
4	16	3.834058	8	440.0	230	3.23	5.345	17.42	0	0	3	4	0	Chrysl Imperi

In [46]:
```
# Draw plot
import matplotlib.patches as patches

plt.figure(figsize=(14,16), dpi= 80)
plt.hlines(y=df.index, xmin=0, xmax=df.mpg_z,
           color=df.colors, alpha=0.4, linewidth=1)
plt.scatter(df.mpg_z, df.index, color=df.colors,
            s=[600 if x == 'Fiat X1-9' else 300 for x in df.cars],
            alpha=0.6)
plt.yticks(df.index, df.cars)
plt.xticks(fontsize=12)

# Annotate
plt.annotate(
    'Mercedes Models', xy=(0.0, 11.0),
    xytext=(1.0, 11), xycoords='data',
    fontsize=15, ha='center', va='center',
    bbox=dict(boxstyle='square', fc='firebrick'),
    arrowprops=dict(arrowstyle='-[, widthB=2.0, lengthB=1.5',
                    lw=2.0, color='steelblue'),
    color='white'
)

# Add Patches
p1 = patches.Rectangle(
    (-2.0, -1), width=.3, height=3, alpha=.2, facecolor='red'
)
p2 = patches.Rectangle(
    (1.5, 27), width=.8, height=5, alpha=.2, facecolor='green'
```

```
)
plt.gca().add_patch(p1)
plt.gca().add_patch(p2)

# Decorate
plt.title('Diverging Bars of Car Mileage', fontdict={'size':20})
plt.grid(linestyle='--', alpha=0.5)
plt.show()
```



6. Диаграмма с областями (Area Chart)

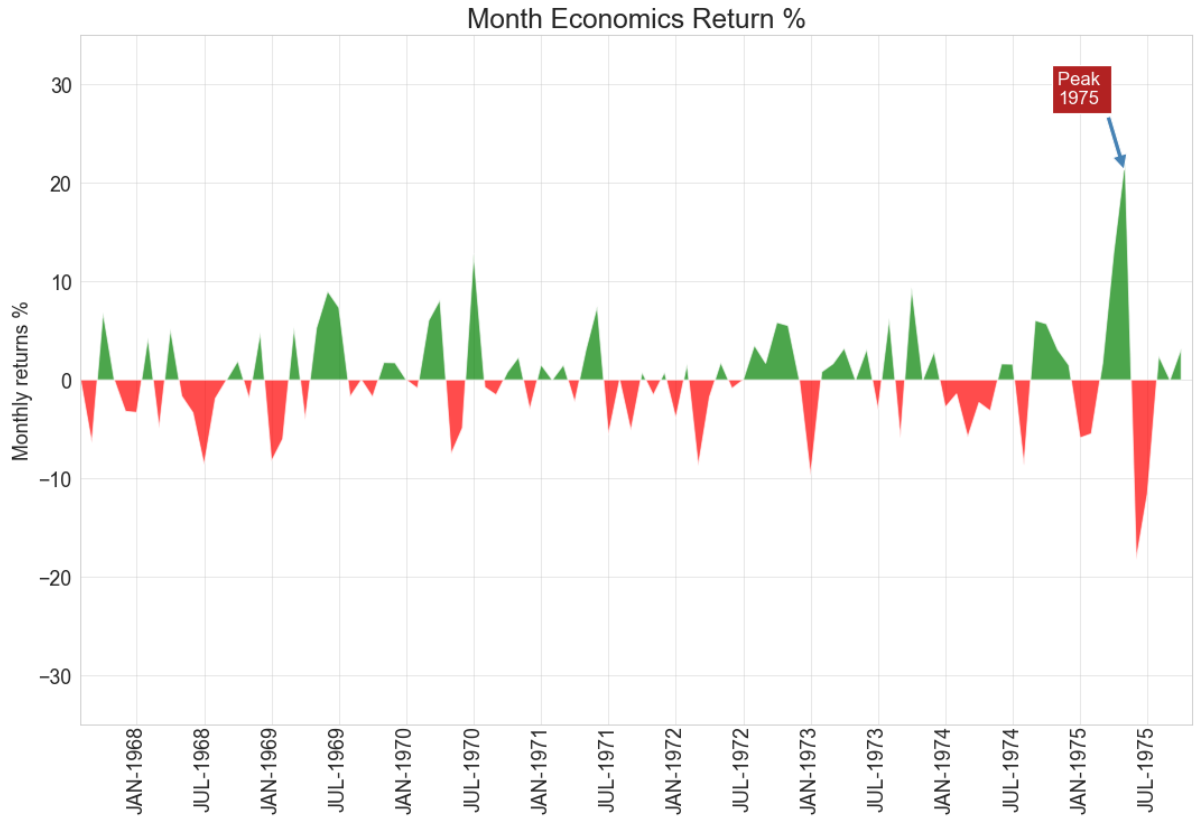
Раскрашивая область между осью и линиями, диаграмма с областями подчеркивает пики и впадины, а также продолжительность максимумов и минимумов. Чем больше продолжительность максимумов (минимумов), тем больше площадь над (под) линией.

In [47]:
```
# Prepare Data
df = pd.read_csv("economics.csv", parse_dates=['date']).head(100)
x = np.arange(df.shape[0])
y_returns = (df.psavert.diff().fillna(0)/df.psavert.shift(1)).filln
df.head()
```

Out[47]:

	date	pce	pop	psavert	uempmed	unemploy
0	1967-07-01	507.4	198712	12.5	4.5	2944
1	1967-08-01	510.5	198911	12.5	4.7	2945
2	1967-09-01	516.3	199113	11.7	4.6	2958
3	1967-10-01	512.9	199311	12.5	4.9	3143
4	1967-11-01	518.1	199498	12.5	4.7	3066

In [48]:
```
# Plot
plt.figure(figsize=(16,10), dpi= 80)
plt.fill_between(x[1:], y_returns[1:], 0, where=y_returns[1:] >= 0,
                 facecolor='green', interpolate=True, alpha=0.7)
plt.fill_between(x[1:], y_returns[1:], 0, where=y_returns[1:] <= 0,
                 facecolor='red', interpolate=True, alpha=0.7)

# Annotate
plt.annotate('Peak \n1975', xy=(94.0, 21.0), xytext=(88.0, 28),
             bbox=dict(boxstyle='square', fc='firebrick'),
             arrowprops=dict(facecolor='steelblue', shrink=0.05),
             fontsize=15, color='white')


# Decorations
xtickvals = [str(m)[:3].upper()+"-"+str(y) for y,m in
             zip(df.date.dt.year, df.date.dt.month_name())]
plt.gca().set_xticks(x[::6])
plt.gca().set_xticklabels(
    xtickvals[::6], rotation=90,
    fontdict={'horizontalalignment': 'center',
              'verticalalignment': 'center_baseline'}
)
plt.ylim(-35,35)
plt.xlim(1,100)
plt.title("Month Economics Return %", fontsize=22)
plt.ylabel('Monthly returns %')
plt.grid(alpha=0.5)
plt.show()
```



Круговые диаграммы

Круговая диаграмма — это круговой график, отображающий числовые пропорции путём деления круга (или пирога) на пропорциональные части. Круговые диаграммы широко используются в бизнесе. Круговые диаграммы создаются в Matplotlib при помощи ключевого слова `kind=pie` .

Используем круговую диаграмму для исследования доли (процента) новых иммигрантов, сгруппированных по континентам за весь период с 1980 по 2013 год.

Шаг 1: Сбор данных. Воспользуемся методом `groupby` библиотеки Pandas для суммирования данных об иммиграции по континентам. Общий процесс `groupby` включает следующие этапы:

1. Разделение (Split): разделение данных на группы на основе определённого критерия.
2. Применение (Apply): применение функции к каждой группе независимо: `.sum()` `.count()` `.mean()` `.std()` `.aggregate()` `.apply()` и т. д.
3. Объединение (Combine): объединение результатов в структуру данных.

In [49]:
```
# group countries by continents and apply sum() function
df_continents = df_can.groupby('Continent', axis=0).sum()

# note: the output of the groupby method is a `groupby' object.
# we can not use it further until we apply a function (eg .sum())
print(type(df_can.groupby('Continent', axis=0)))

df_continents.head()
```

```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

Out[49]:

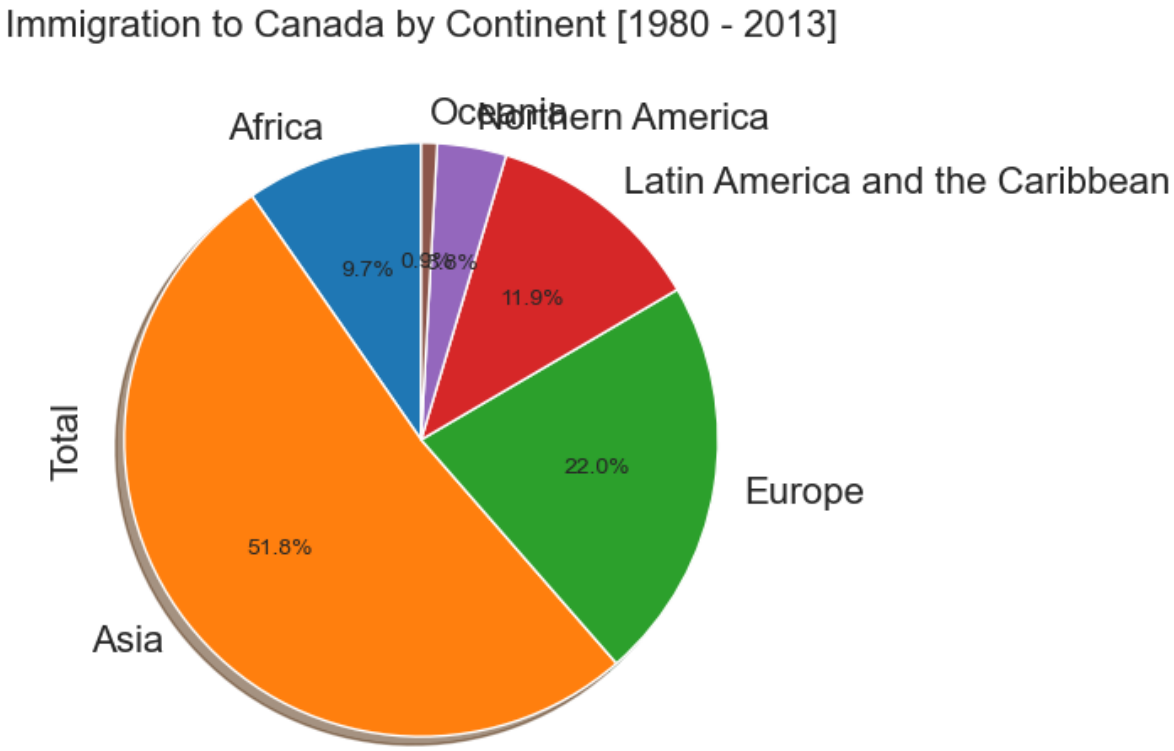
Continent	Region	DevName	1980	1981	1982	19
Africa	Northern AfricaMiddle AfricaMiddle AfricaWeste...	Developing regionsDeveloping regionsDeveloping...	3951	4363	3819	26
Asia	Southern AsiaWestern AsiaWestern AsiaEastern A...	Developing regionsDeveloping regionsDeveloping...	31025	34314	30214	246
Europe	Southern EuropeSouthern EuropeWestern EuropeWe...	Developed regionsDeveloped regionsDeveloped re...	39760	44802	42720	246
Latin America and the Caribbean	CaribbeanSouth AmericaCaribbeanCaribbeanCentra...	Developing regionsDeveloping regionsDeveloping...	13081	15215	16769	154
Northern America	Northern AmericaNorthern America	Developed regionsDeveloped regions	9378	10030	9074	71

5 rows × 37 columns

Шаг 2: Построим диаграмму. Передаем ключевое слово `kind = 'pie'` вместе со следующими дополнительными параметрами:

- `autopct` — строка или функция, используемая для маркировки секторов с их числовыми значениями. Метка будет размещена внутри сектора. Если это строка формата, метка будет иметь вид `fmt%pct` .
- `startangle` — поворачивает начало круговой диаграммы на угол в градусах против часовой стрелки относительно оси X.
- `shadow` — рисует тень под круговой диаграммой (для создания эффекта трёхмерности).

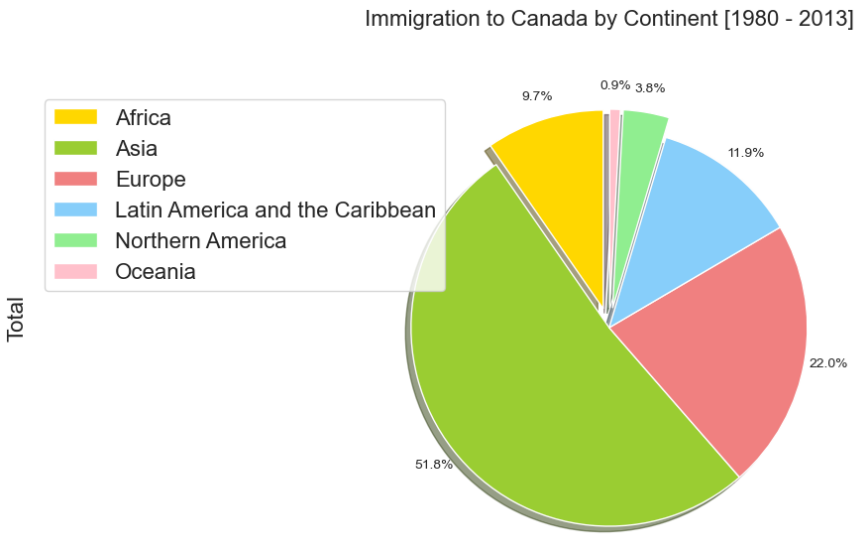
In [50]:
```
# autopct create %, start angle represent starting point
df_continents['Total'].plot(
    kind='pie',
    figsize=(5, 6),
    autopct='%1.1f%%', # add in percentages
    startangle=90,     # start angle 90° (Africa)
    shadow=True,       # add shadow
)

plt.title('Immigration to Canada by Continent [1980 - 2013]')
plt.axis('equal') # Sets the pie chart to look like a circle.

plt.show()
```



Изображение выше не очень понятное, числа и текст в некоторых местах перекрываются. Внесем несколько изменений для улучшения изображения:

- Удалим текстовые подписи с круговой диаграммы, передав параметр `legend`, и добавьте его как отдельную легенду с помощью `plt.legend()`.
- Вынесем процентные значения за пределы круговой диаграммы, передав параметр `pctdistance`.
- Передадим пользовательский набор цветов для континентов, передав параметр `colors`.
- Разорвем круговую диаграмму, чтобы выделить три нижних континента (Африку, Северную Америку, Латинскую Америку и Карибский бассейн), передав параметр `explode`.

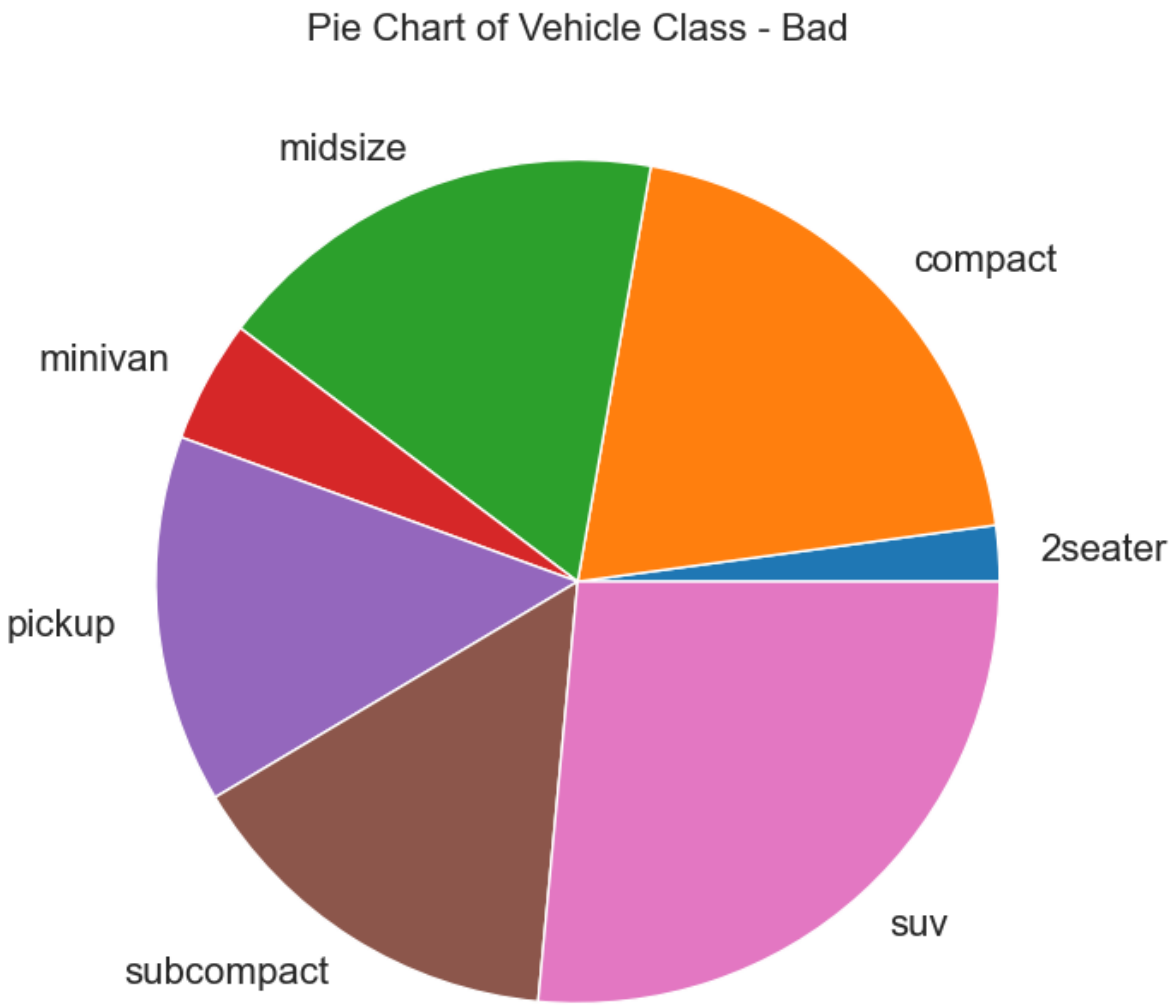
In [51]:
```
colors_list = ['gold', 'yellowgreen', 'lightcoral',
               'lightskyblue', 'lightgreen', 'pink']
# ratio for each continent with which to offset each wedge.
explode_list = [0.1, 0, 0, 0, 0.1, 0.1]

df_continents['Total'].plot(
    kind='pie',
    figsize=(15, 6),
    autopct='%1.1f%%',
    startangle=90,
    shadow=True,
    labels=None,         # turn off labels on pie chart
    pctdistance=1.12,    # the ratio between the center of each pie
                         # slice and the start of the text generate
                         # by autopct
    colors=colors_list,  # add custom colors
    explode=explode_list # 'explode' lowest 3 continents
)

# scale the title up by 12% to match pctdistance
plt.title('Immigration to Canada by Continent [1980 - 2013]', y=1.1

plt.axis('equal')

# add legend
plt.legend(labels=df_continents.index, loc='upper left')

plt.show()
```



1. Круговая диаграмма (Pie Chart)

Круговая диаграмма — это классический способ показать состав групп.

In [52]:
```
# Import
df_raw = pd.read_csv("mpg_ggplot2.csv")

# Prepare Data
df = df_raw.groupby('class').size()

# Make the plot with pandas
df.plot(kind='pie', subplots=True, figsize=(8, 8)) #, dpi= 80)
plt.title("Pie Chart of Vehicle Class - Bad")
plt.ylabel("")
plt.show()
```

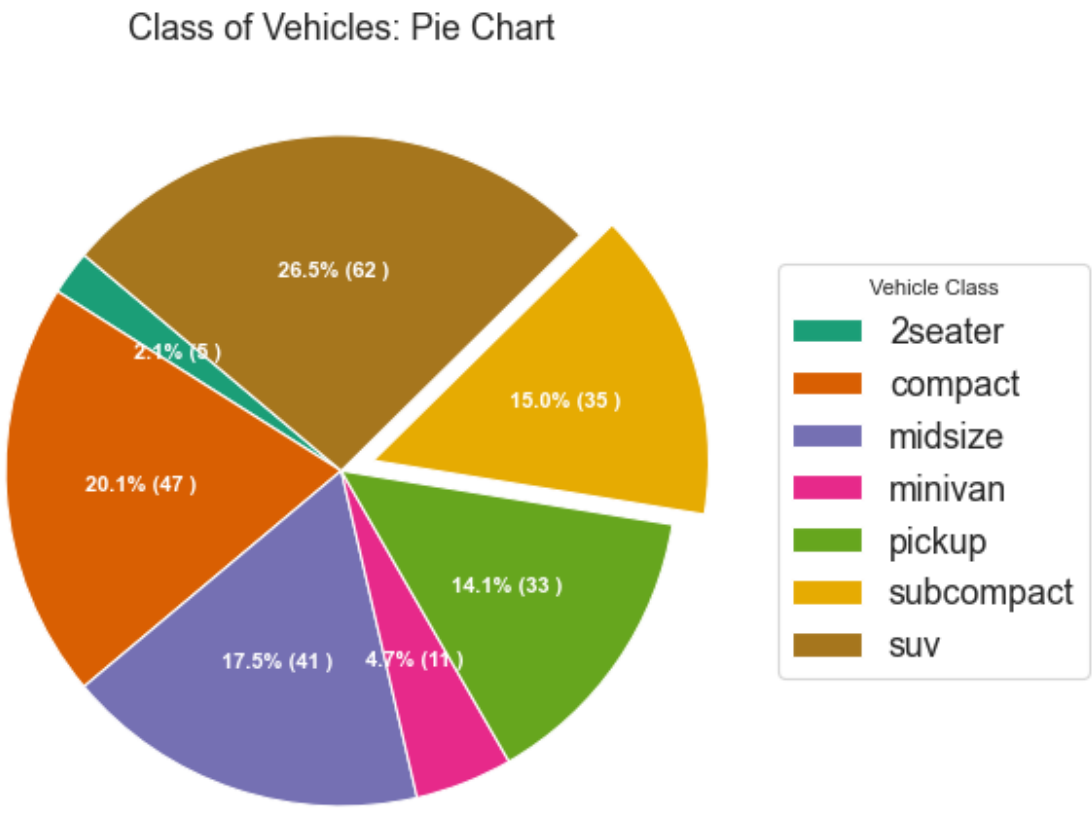


In [53]:
```
# Import
df_raw = pd.read_csv("mpg_ggplot2.csv")

# Prepare Data
df = df_raw.groupby('class').size().reset_index(name='counts')
df.head()
```

Out[53]:

	class	counts
0	2seater	5
1	compact	47
2	midsize	41
3	minivan	11
4	pickup	33

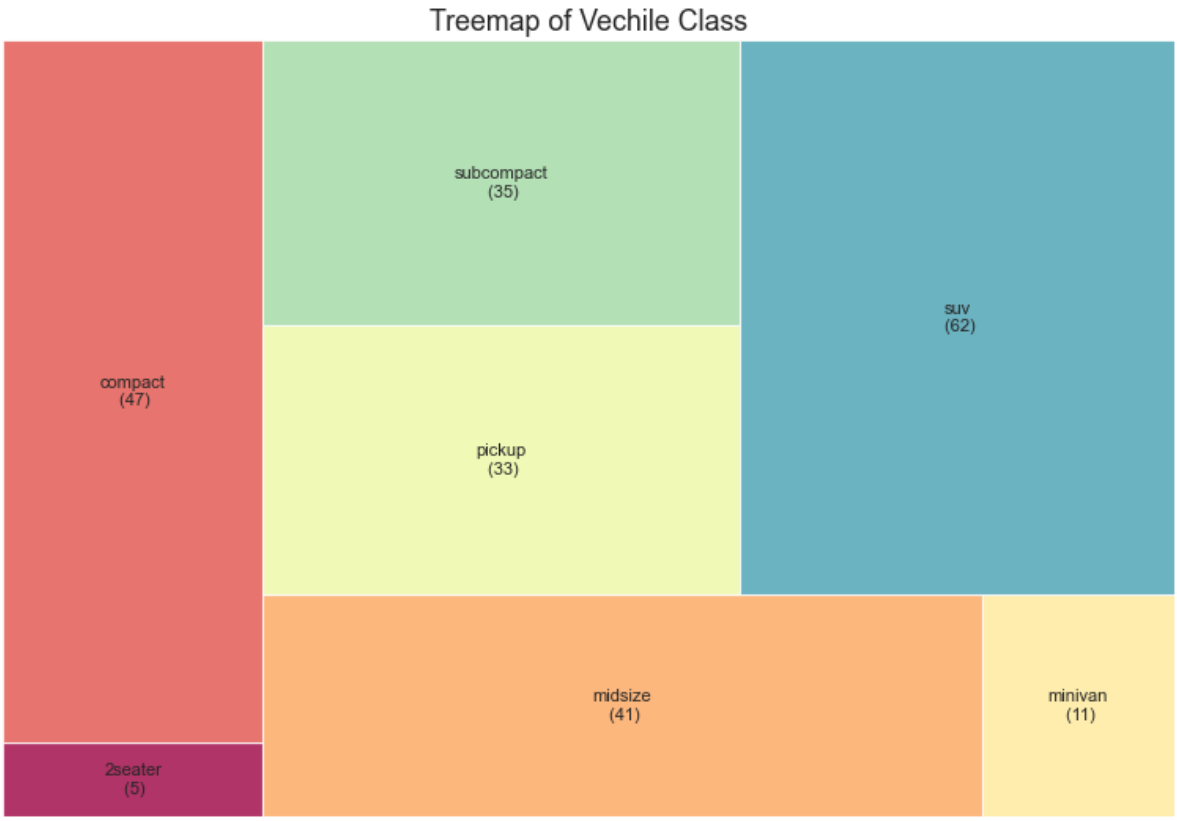
In [54]:
```
# Draw Plot
fig, ax = plt.subplots(
    figsize=(12, 7), subplot_kw=dict(aspect="equal"), dpi= 80
)

data = df['counts']
categories = df['class']
explode = [0,0,0,0,0,0.1,0]

def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}% ({:d} )".format(pct, absolute)

wedges, texts, autotexts = ax.pie(
    data,
    autopct=lambda pct: func(pct, data),
    textprops=dict(color="w"),
    colors=plt.cm.Dark2.colors,
    startangle=140,
    explode=explode
)

# Decoration
ax.legend(wedges, categories, title="Vehicle Class", loc="center le
          bbox_to_anchor=(1, 0, 0.5, 1))
plt.setp(autotexts, size=10, weight=700)
ax.set_title("Class of Vehicles: Pie Chart")
plt.show()
```



2. Древовидная карта (Treemap)

Древовидная карта похожа на круговую диаграмму и работает лучше, точно отображая долю каждой группы.

In [55]:
```
#!pip install squarify
import squarify

# Import Data
df_raw = pd.read_csv("mpg_ggplot2.csv")

# Prepare Data
df = df_raw.groupby('class').size().reset_index(name='counts')
labels = df.apply(lambda x: str(x[0])+"\n ("+str(x[1])+")", axis=1)
sizes = df['counts'].values.tolist()
colors = [
    plt.cm.Spectral(i/float(len(labels)))
    for i in range(len(labels))
]

# Draw Plot
plt.figure(figsize=(12,8), dpi= 80)
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=.8)

# Decorate
plt.title('Treemap of Vechile Class')
plt.axis('off')
plt.show()
```



Диаграммы рассеяния

Диаграммы рассеяния (2D) — это полезный метод сравнения переменных. Диаграммы рассеяния похожи на линейные графики тем, что отображают независимые и зависимые переменные на двумерном графике. В то время как на линейном графике точки данных соединены линией, на диаграмме рассеяния они не связаны. Данные на диаграмме рассеяния считаются отражающими тенденцию. Дальнейший анализ с использованием таких инструментов, как регрессия, позволяет математически рассчитать эту зависимость и использовать её для прогнозирования тенденций за пределами набора данных.

Пузырьковая диаграмма (bubble plot) с выделением группы

Иногда необходимо показать группу точек внутри области, чтобы подчеркнуть их важность.

In [56]:
```
from matplotlib import patches
from scipy.spatial import ConvexHull

# Step 1: Prepare Data
midwest = pd.read_csv("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))
]
midwest.head()
```

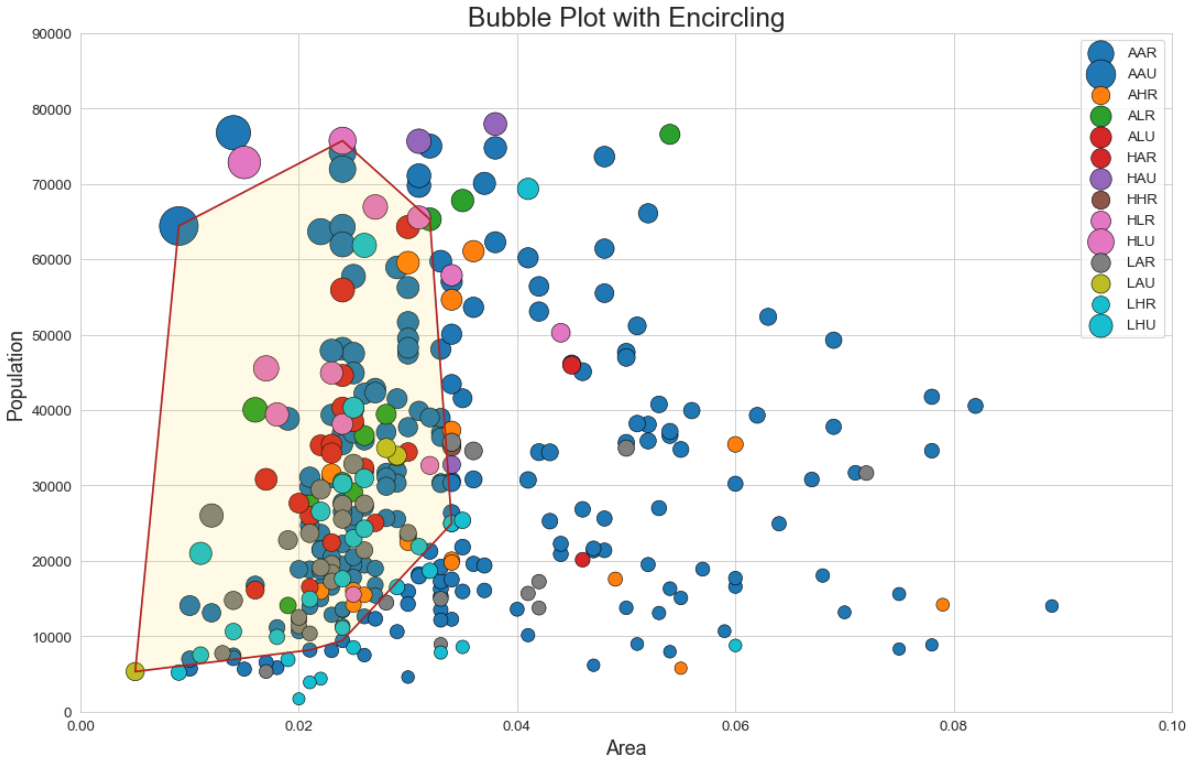
Out[56]:

	PID	county	state	area	poptotal	popdensity	popwhite	popblack	popamerindi
0	561	ADAMS	IL	0.052	66090	1270.961540	63917	1702	
1	562	ALEXANDER	IL	0.014	10626	759.000000	7054	3496	
2	563	BOND	IL	0.022	14991	681.409091	14477	429	
3	564	BOONE	IL	0.017	30806	1812.117650	29344	127	
4	565	BROWN	IL	0.018	5836	324.222222	5264	547	

5 rows × 29 columns

In [57]:
```
# 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', color=colors[i], label=str(category),
                edgecolors='black', linewidths=.5)

# Step 3: Encircling
def encircle(x,y, ax=None, **kw):
    if not ax: ax=plt.gca()
    # translates slice objects to concatenation along the second ax
    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()
```



Задание на ЛР №5

В соответствии с индивидуальным заданием (вариантом), переданным через программу «Мессенджер Яндекс», выполните следующие работы:

1. Считайте из заданного набора данных репозитария UCI значения трех признаков и метки класса.
2. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Если в признаках имеются пропущенные значения, то выведите процент записей набора данных с пропущенными значениями и замените пропущенные значения на значения, указанные в индивидуальном задании. При отсутствии пропущенных значений удалите не менее 5% точек набор данных как выбросов при помощи стандартизованной оценки и выведите процент удаленных точек.
3. Постройте **гистограмму** (Histogram) частот для дискретизированного признака с наибольшей дисперсией (количество бинов по Вашему выбору), визуализируя вклад каждого класса в виде цветного столбика. Подпишите оси и рисунок, создайте легенду для классов.
4. Постройте **столбчатую диаграмму** (Bar Chart) типа, указанного в индивидуальном задании, показывающую зависимость среднего значения признака с наибольшей дисперсией от класса. Сделайте аннотацию для класса с наибольшим средним значением этого признака.
5. Постройте **круговую диаграмму** (Pie Chart) или древовидную карту (Treemap) согласно индивидуальному заданию для количеств точек данных, принадлежащих разным классам. Укажите процент точек класса и создайте легенду для классов.
6. Постройте **пузырьковую диаграмму** (Bubble Plot) на плоскости с координатами, соответствующими двум признакам с большей дисперсией, отображая точки различных классов разными цветами и маркерами переменных размеров в зависимости от значений признака с меньшей дисперсией. Выделите область диаграммы согласно индивидуальному заданию. Подпишите оси и рисунок, создайте легенду для классов.

In [ ]: