In [9]:
```
import pandas as pd
import numpy as np
```

In [10]:
```
bike = pd.read_csv("D:/dataset/Bike/biketrain.csv", parse_dates=["datetime"])
titanic = pd.read_csv("D:/dataset/titanic_data/train_modified.csv")
print(bike.shape, titanic.shape)
```

```
(10886, 12) (891, 12)
```

In [11]:
```
# !pip install brewer2mpl
```

```
Collecting brewer2mpl
  Downloading https://files.pythonhosted.org/packages/84/57/00c45a199719e617db0875181134fcb3aeef701deae346547ac722eaaf5e/brewer2mpl-1.4.1-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/84/57/00c45a199719e617db0875181134fcb3aeef701deae346547ac722eaaf5e/brewer2mpl-1.4.1-py2.py3-none-any.whl)
Installing collected packages: brewer2mpl
Successfully installed brewer2mpl-1.4.1
```

In [12]:
```
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='once')

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("white")
%matplotlib inline

# Version
print(mpl.__version__)  #> 3.0.0
print(sns.__version__)  #> 0.9.0
```

```
2.1.2
0.8.1
```

Correlation

In [4]:
```
bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.6 KB
```

In [5]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       891 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

01. Scatter Plot

- 2개 또는 그 이상의 변수의 관계를 확인

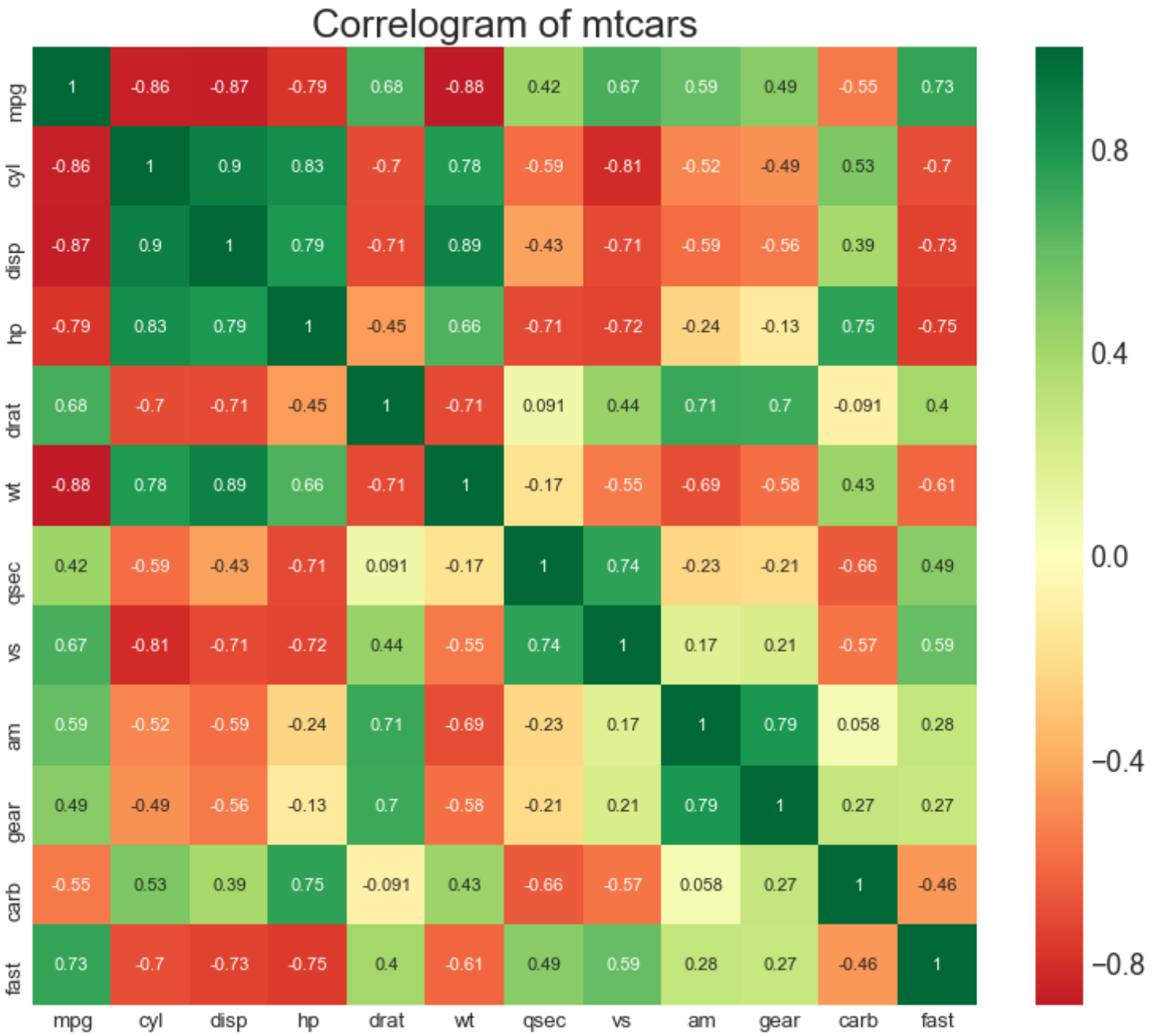
In [26]:
```
# Import Dataset
df = pd.read_csv("https://github.com/selva86/datasets/raw/master/mtcars.csv")
print(df.shape)
print(df.head())
print(df.columns)
print(df.info())
```

```
(32, 14)
        mpg  cyl   disp   hp  drat     wt   qsec  vs  am  gear  carb  fast  W
0  4.582576    6  160.0  110  3.90  2.620  16.46   0   1     4     4     1
1  4.582576    6  160.0  110  3.90  2.875  17.02   0   1     4     4     1
2  4.774935    4  108.0   93  3.85  2.320  18.61   1   1     4     1     1
3  4.626013    6  258.0  110  3.08  3.215  19.44   1   0     3     1     1
4  4.324350    8  360.0  175  3.15  3.440  17.02   0   0     3     2     1

                cars           carname
0          Mazda RX4          Mazda RX4
1      Mazda RX4 Wag      Mazda RX4 Wag
2         Datsun 710         Datsun 710
3     Hornet 4 Drive     Hornet 4 Drive
4  Hornet Sportabout  Hornet Sportabout
Index(['mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear',
       'carb', 'fast', 'cars', 'carname'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 14 columns):
mpg        32 non-null float64
cyl        32 non-null int64
disp       32 non-null float64
hp         32 non-null int64
drat       32 non-null float64
wt         32 non-null float64
qsec       32 non-null float64
vs         32 non-null int64
am         32 non-null int64
gear       32 non-null int64
carb       32 non-null int64
fast       32 non-null int64
cars       32 non-null object
carname    32 non-null object
```

```
dtypes: float64(5), int64(7), object(2)
memory usage: 3.6+ KB
None
```

In [ ]:

In [24]:
```
# Plot
plt.figure(figsize=(12,10), dpi= 80)
sns.heatmap(df.corr(),
            xticklabels=df.corr().columns,
            yticklabels=df.corr().columns, cmap='RdYlGn', center=0, annot=True)

# Decorations
plt.title('Correlogram of mtcars', fontsize=22)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

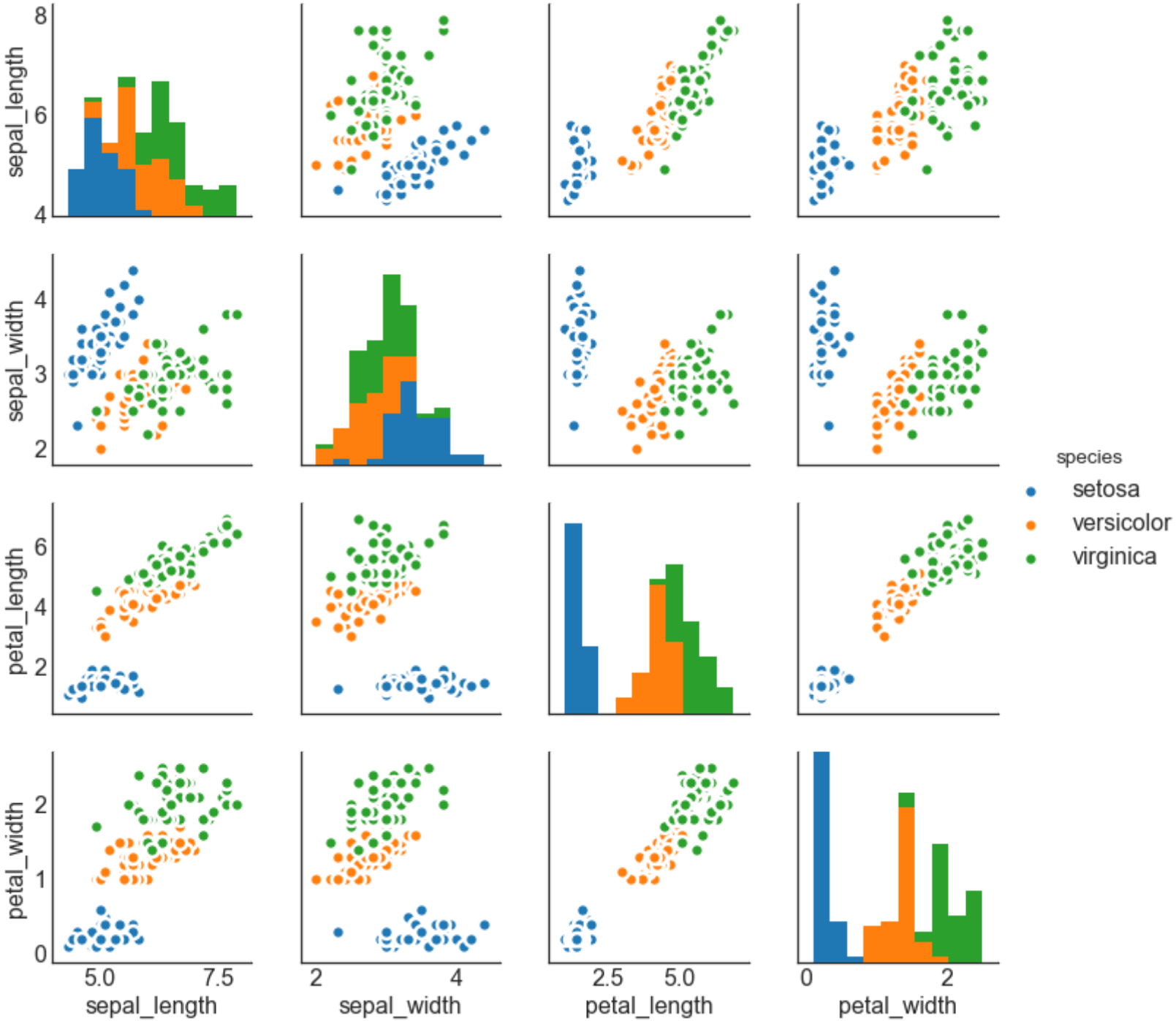

Correlogram of mtcars
mpg 1 -0.86 -0.87 -0.79 0.68 -0.88 0.42 0.67 0.59 0.49 -0.55 0.73
cyl -0.86 1 0.9 0.83 -0.7 0.78 -0.59 -0.81 -0.52 -0.49 0.53 -0.7
disp -0.87 0.9 1 0.79 -0.71 0.89 -0.43 -0.71 -0.59 -0.56 0.39 -0.73
hp -0.79 0.83 0.79 1 -0.45 0.66 -0.71 -0.72 -0.24 -0.13 0.75 -0.75
drat 0.68 -0.7 -0.71 -0.45 1 -0.71 0.091 0.44 0.71 0.7 -0.091 0.4
wt -0.88 0.78 0.89 0.66 -0.71 1 -0.17 -0.55 -0.69 -0.58 0.43 -0.61
qsec 0.42 -0.59 -0.43 -0.71 0.091 -0.17 1 0.74 -0.23 -0.21 -0.66 0.49
vs 0.67 -0.81 -0.71 -0.72 0.44 -0.55 0.74 1 0.17 0.21 -0.57 0.59
am 0.59 -0.52 -0.59 -0.24 0.71 -0.69 -0.23 0.17 1 0.79 0.058 0.28
gear 0.49 -0.49 -0.56 -0.13 0.7 -0.58 -0.21 0.21 0.79 1 0.27 0.27
carb -0.55 0.53 0.39 0.75 -0.091 0.43 -0.66 -0.57 0.058 0.27 1 -0.46
fast 0.73 -0.7 -0.73 -0.75 0.4 -0.61 0.49 0.59 0.28 0.27 -0.46 1
mpg cyl disp hp drat wt qsec vs am gear carb fast
0.8
0.4
0.0
−0.4
−0.8

Pairwise Plot

- Pairwise Plot is a favorite in exploratory analysis to understand the relationship between all possible pairs of numeric variables. It is a must have tool for bivariate analysis
- 수치형 변수의 가능한 모든 쌍 사이의 관계를 이해하기 위해 좋다.

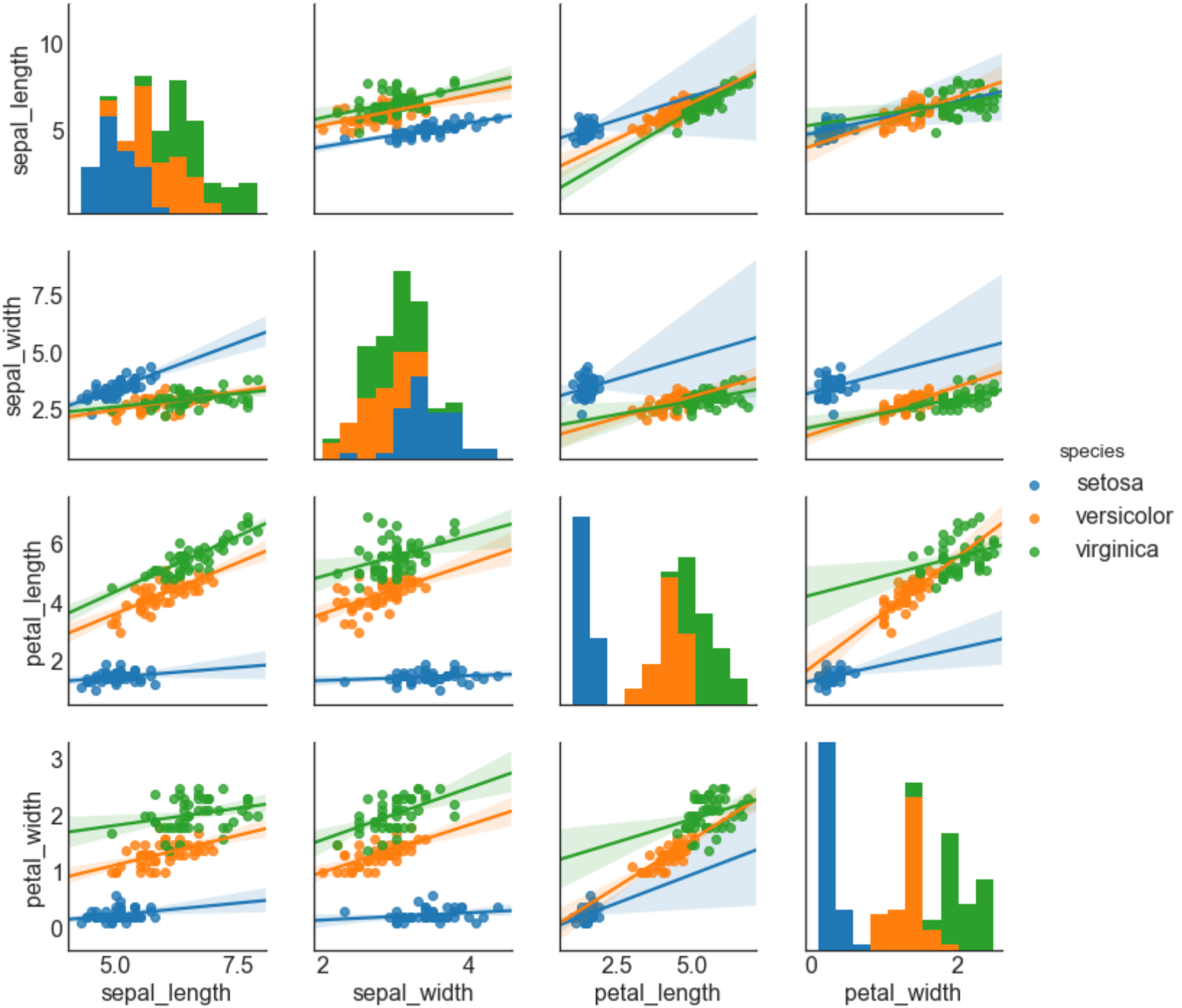
In [27]:
```
# Load Dataset
df = sns.load_dataset('iris')

# Plot
plt.figure(figsize=(10,8), dpi= 80)
sns.pairplot(df, kind="scatter", hue="species", plot_kws=dict(s=80, edgecolor="white", linewidth=2.5))
plt.show()
```

```
<matplotlib.figure.Figure at 0x1e919f8ccf8>
```


sepal_length
sepal_width
petal_length
petal_width
5.0
7.5
2
4
2.5
5.0
0
2
species
setosa
versicolor
virginica

In [28]:
```
# Load Dataset
df = sns.load_dataset('iris')

# Plot
plt.figure(figsize=(10,8), dpi= 80)
sns.pairplot(df, kind="reg", hue="species")
plt.show()
```

```
<matplotlib.figure.Figure at 0x1e919f41a58>
```


sepal_length
sepal_width
petal_length
petal_width
10
5
7.5
5.0
2.5
6
4
2
3
2
1
0
5.0
7.5
2
4
2.5
5.0
0
2
species
setosa
versicolor
virginica

15. Ordered Bar Chart

Fuel economy data from 1999 and 2008 for 38 popular models of car
A data frame with 234 rows and 11 variables

This dataset contains a subset of the fuel economy data that the EPA makes available on http://fueleconomy.gov. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

이 데이터 세트에는 EPA가 http://fueleconomy.gov에서 제공하는 연비 데이터의
일부를 포함하고 있습니다. 1999 년에서 2008 년 사이에 매년 새 버전이
출시 된 모델 만 포함되어 있으며 이는 자동차의 인기도를 위한 프록시로 사용되었습니다.

manufacturer model : model name
displ : engine displacement, in litres
year : year of manufacture
cyl   : number of cylinders
trans : type of transmission
drv   : f = front-wheel drive, r = rear wheel drive, 4 = 4wd
cty   : city miles per gallon (갤런당 도시 마일)
hwy   : highway miles per gallon
fl   : fuel type
class : "type" of car

In [29]:
```
# Import Dataset
df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")
print(df_raw.shape)
print(df_raw.head())
print(df_raw.columns)
print(df_raw.info())
```

```
(234, 11)
  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
Index(['manufacturer', 'model', 'displ', 'year', 'cyl', 'trans', 'drv', 'cty',
       'hwy', 'fl', 'class'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 11 columns):
manufacturer    234 non-null object
model           234 non-null object
displ           234 non-null float64
year            234 non-null int64
cyl             234 non-null int64
trans           234 non-null object
drv             234 non-null object
cty             234 non-null int64
hwy             234 non-null int64
fl              234 non-null object
class           234 non-null object
dtypes: float64(1), int64(4), object(6)
memory usage: 20.2+ KB
None
```

In [38]:
```
df = df_raw[['cty', 'manufacturer']].groupby('manufacturer').apply(lambda x: x.mean())
df
```

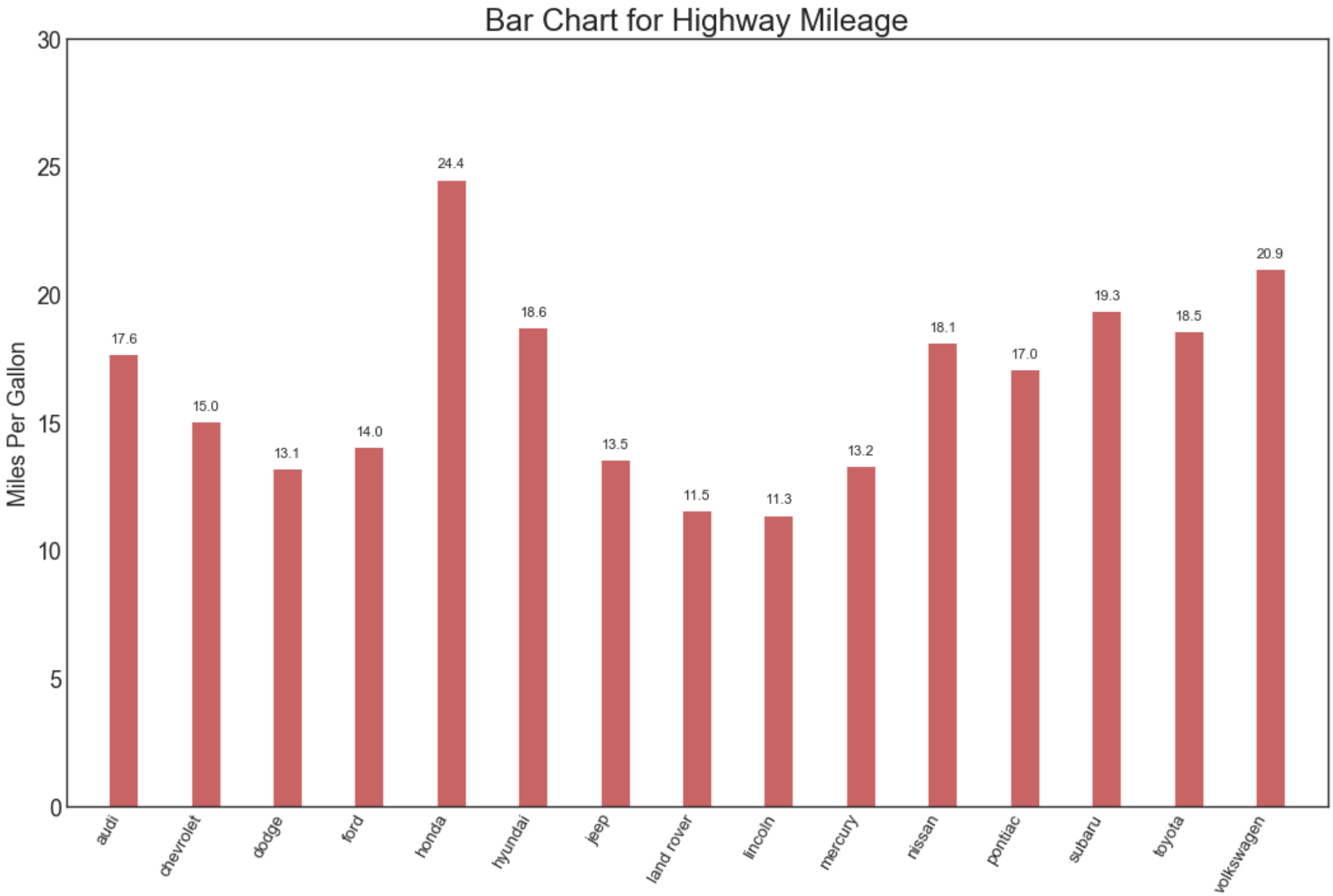
Out[38]:

manufacturer	cty
audi	17.611111
chevrolet	15.000000
dodge	13.135135
ford	14.000000
honda	24.444444
hyundai	18.642857
jeep	13.500000
land rover	11.500000
lincoln	11.333333
mercury	13.250000
nissan	18.076923
pontiac	17.000000
subaru	19.285714
toyota	18.529412
volkswagen	20.925926

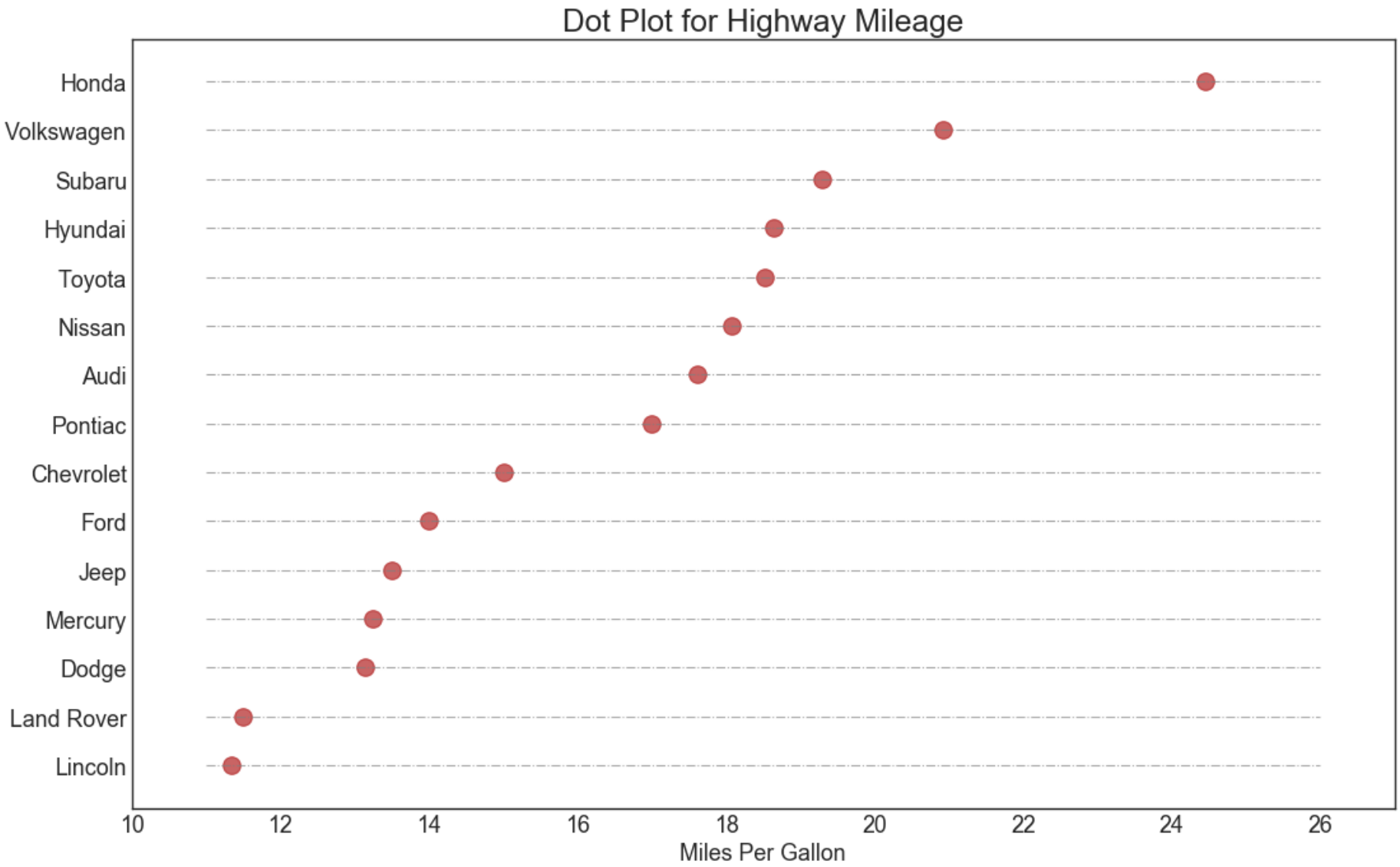
In [49]:
```
df.index
```

Out[49]:
```
Index(['audi', 'chevrolet', 'dodge', 'ford', 'honda', 'hyundai', 'jeep',
       'land rover', 'lincoln', 'mercury', 'nissan', 'pontiac', 'subaru',
       'toyota', 'volkswagen'],
      dtype='object', name='manufacturer')
```

In [52]:
```
# Draw plot
import matplotlib.patches as patches

fig, ax = plt.subplots(figsize=(16,10), facecolor='white', dpi= 80)
ax.vlines(x=df.index, ymin=0, ymax=df.cty, color='firebrick', alpha=0.7, linewidth=20)

# Annotate Text
for i, cty in enumerate(df.cty):
    ax.text(i, cty+0.5, round(cty, 1), horizontalalignment='center')


# Title, Label, Ticks and Ylim
ax.set_title('Bar Chart for Highway Mileage', fontdict={'size':22})
ax.set(ylabel='Miles Per Gallon', ylim=(0, 30))
plt.xticks(df.index, rotation=60, horizontalalignment='right', fontsize=12)

plt.show()
```


Bar Chart for Highway Mileage
Miles Per Gallon
0
5
10
15
20
25
30
17.6
15.0
13.1
14.0
24.4
18.6
13.5
11.5
11.3
13.2
18.1
17.0
19.3
18.5
20.9
audi
chevrolet
dodge
ford
honda
hyundai
jeep
land rover
lincoln
mercury
nissan
pontiac
subaru
toyota
volkswagen

In [59]:
```
df = df_raw[['cty', 'manufacturer']].groupby('manufacturer').apply(lambda x: x.mean())
df.sort_values('cty', inplace=True)
df.reset_index(inplace=True)

# Draw plot
fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
ax.hlines(y=df.index, xmin=11, xmax=26, color='gray', alpha=0.7, linewidth=1, linestyles='dashdot')
ax.scatter(y=df.index, x=df.cty, s=150, color='firebrick', alpha=0.7)

# Title, Label, Ticks and Ylim
ax.set_title('Dot Plot for Highway Mileage', fontdict={'size':22})
ax.set_xlabel('Miles Per Gallon')
ax.set_yticks(df.index)
ax.set_yticklabels(df.manufacturer.str.title(),
                   fontdict={'horizontalalignment': 'right'})
ax.set_xlim(10, 27)
plt.show()
```

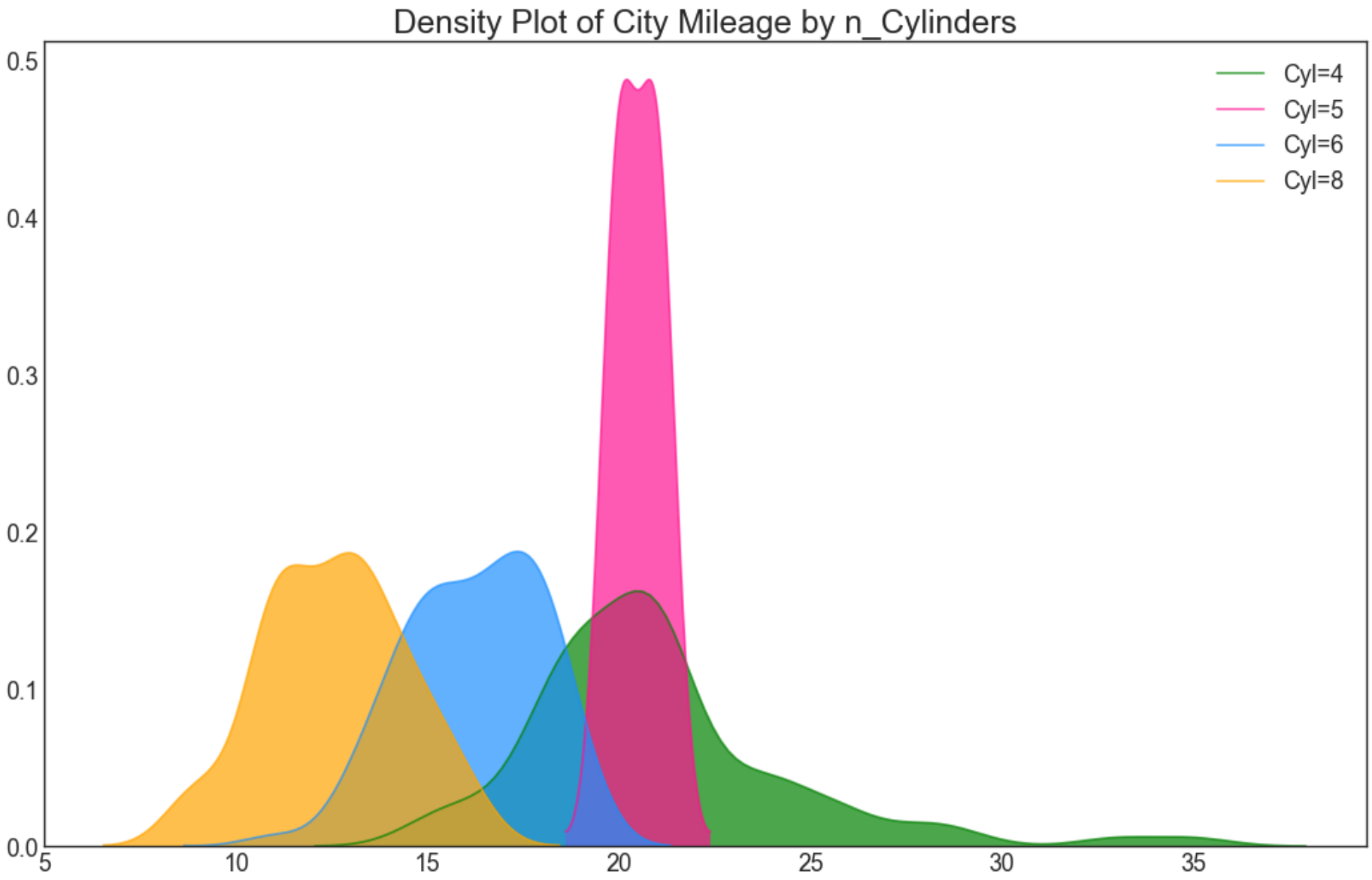


Density Plot

- 밀도 플롯은 일반적으로 사용되는 도구로 연속 변수의 분포를 시각화합니다. '응답'변수로 그룹화하면 X와 Y의 관계를 검사 할 수 있습니다. 도시 마일리지 분포가 실린더 수에 따라 어떻게 달라지는지를 표현하기위한 목적이 아래의 경우입니다.

In [60]:
```
# Import Data
df = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")

# Draw Plot
plt.figure(figsize=(16,10), dpi= 80)
sns.kdeplot(df.loc[df['cyl'] == 4, "cty"], shade=True, color="g", label="Cyl=4", alpha=.7)
sns.kdeplot(df.loc[df['cyl'] == 5, "cty"], shade=True, color="deeppink", label="Cyl=5", alpha=.7)
sns.kdeplot(df.loc[df['cyl'] == 6, "cty"], shade=True, color="dodgerblue", label="Cyl=6", alpha=.7)
sns.kdeplot(df.loc[df['cyl'] == 8, "cty"], shade=True, color="orange", label="Cyl=8", alpha=.7)

# Decoration
plt.title('Density Plot of City Mileage by n_Cylinders', fontsize=22)
plt.legend()
plt.show()
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩statsmodels₩nonparametric₩kde.py:475: DeprecationWarning: object of type <class 'nump
y.float64'> cannot be safely interpreted as an integer.
  grid,delta = np.linspace(a,b,gridsize,retstep=True)
```


Density Plot of City Mileage by n_Cylinders
Cyl=4
Cyl=5
Cyl=6
Cyl=8
0.5
0.4
0.3
0.2
0.1
0.0
5
10
15
20
25
30
35

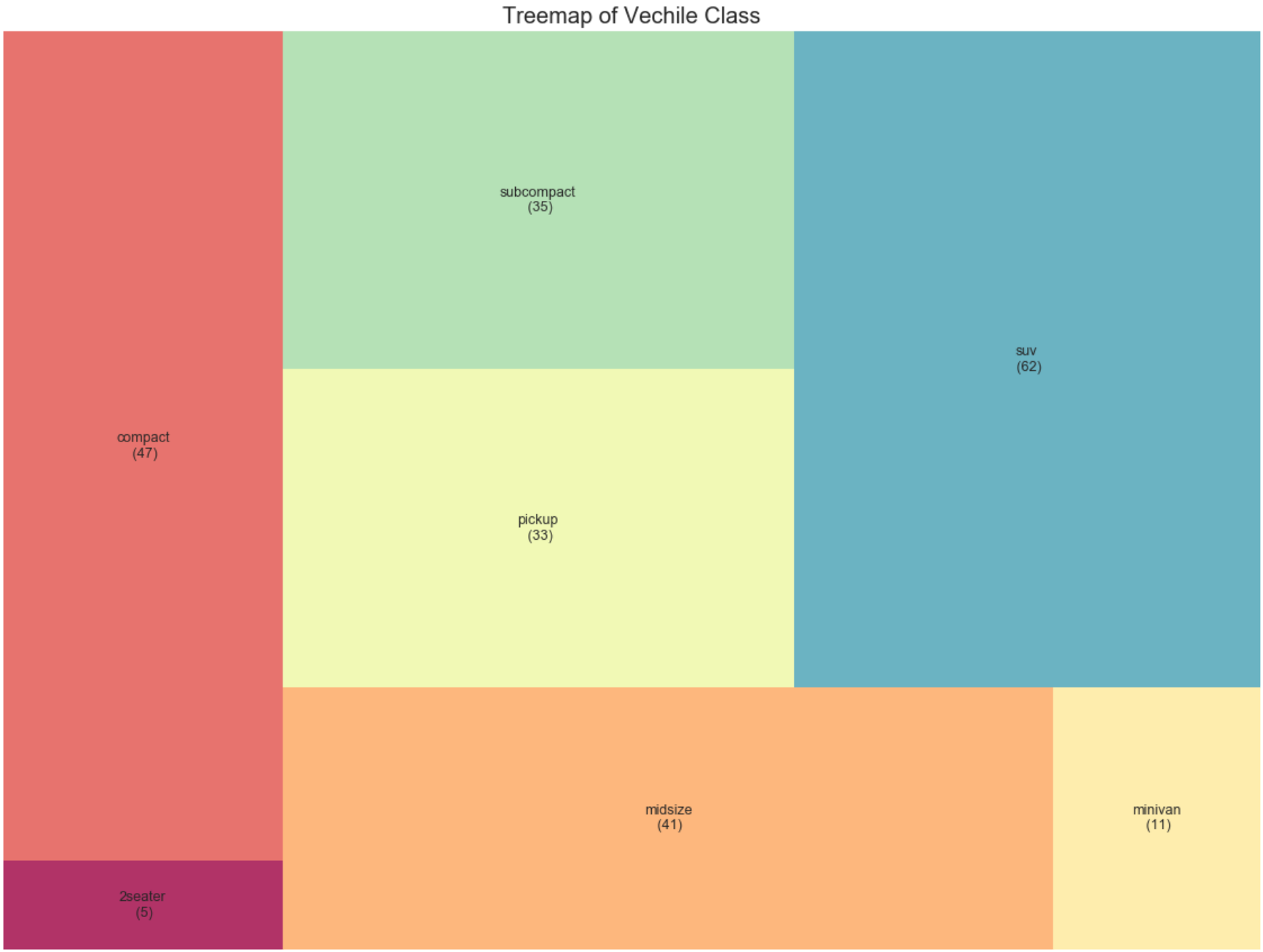
In [65]:
```
!pip install squarify
```

```
Requirement already satisfied: squarify in c:₩users₩withjs₩anaconda3₩lib₩site-packages (0.3.0)

C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩IPython₩utils₩_process_win32.py:131: ResourceWarning: unclosed file <_io.BufferedWrite
r name=6>
  return process_handler(cmd, _system_body)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩IPython₩utils₩_process_win32.py:131: ResourceWarning: unclosed file <_io.BufferedReade
r name=7>
  return process_handler(cmd, _system_body)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩IPython₩utils₩_process_win32.py:131: ResourceWarning: unclosed file <_io.BufferedReade
r name=8>
  return process_handler(cmd, _system_body)
```

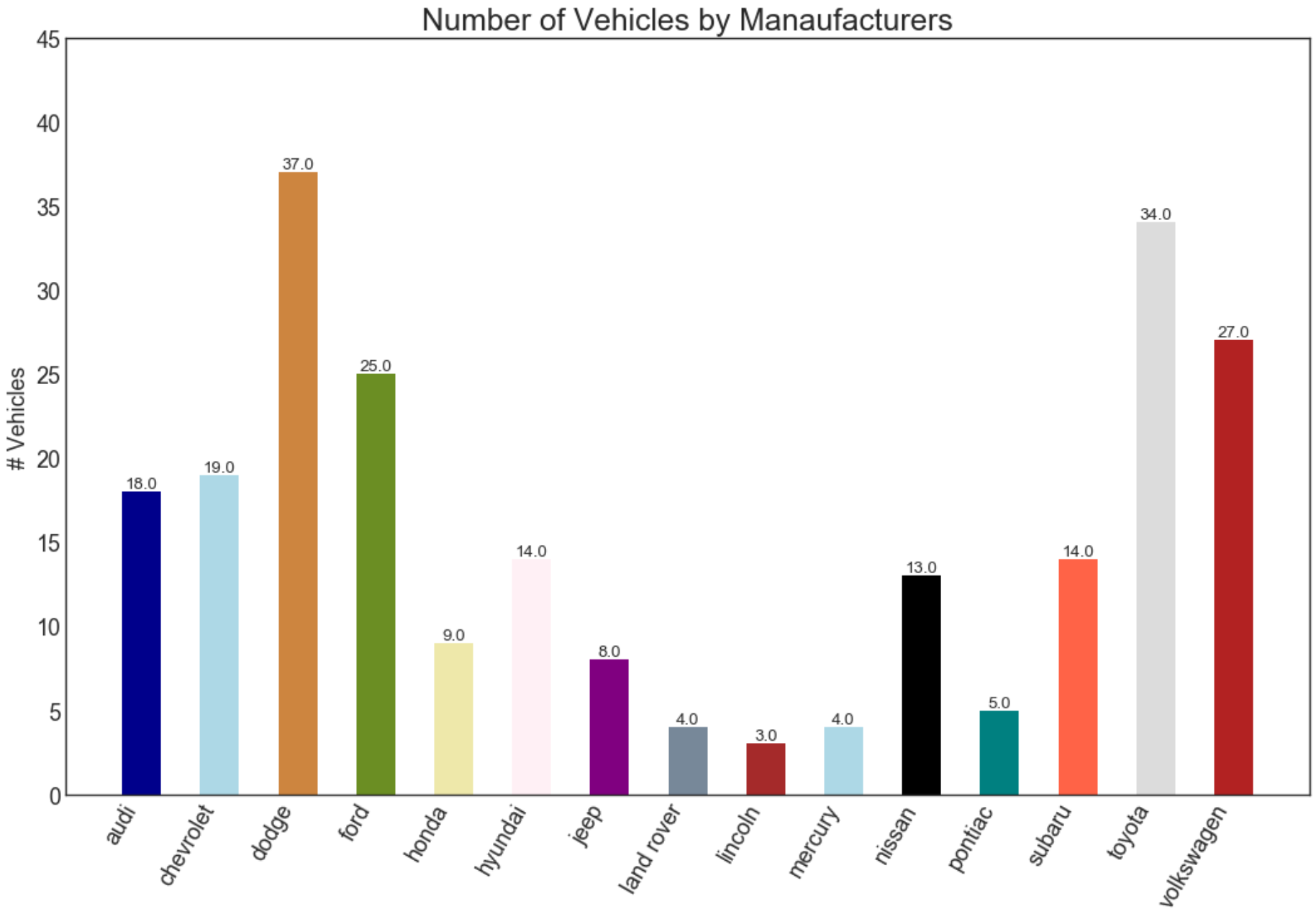
33. Treemap

- 트리 맵은 파이차트와 유사하다. 각 그룹의 기여도를 오도하지 않고 잘 보여준다.

In [70]:
```
import squarify
# Import Data
df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")

# Prepare Data
df = df_raw.groupby('class').size().reset_index(name='counts')
labels = df.apply(lambda x: str(x[0]) + "Wn (" + 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=(16,12), dpi= 90)
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=.8)

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


Treemap of Vechile Class
subcompact
(35)
suv
(62)
compact
(47)
pickup
(33)
midsize
(41)
minivan
(11)
2seater
(5)

In [71]:
```
import random

# Import Data
df_raw = pd.read_csv("https://github.com/selva86/datasets/raw/master/mpg_ggplot2.csv")

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

# 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()
```


Number of Vehicles by Manaufacturers
Vehicles
0
5
10
15
20
25
30
35
40
45
18.0
19.0
37.0
25.0
9.0
14.0
8.0
4.0
3.0
4.0
13.0
5.0
14.0
34.0
27.0
audi
chevrolet
dodge
ford
honda
hyundai
jeep
land rover
lincoln
mercury
nissan
pontiac
subaru
toyota
volkswagen

REF

https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/
(https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

In [ ]: