import Libraries

In [96]:
```
import numpy as np
import pandas as pd
```

In [97]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Loading The Dataset

In [98]:
```
df=sns.load_dataset('mpg')
```

In [99]:
```
df.head()
```

Out[99]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

In [100]:
```
df.shape
```

Out[100]:
```
(398, 9)
```

In [101]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64  
 7   origin        398 non-null    object 
 8   name          398 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

In [102]:
```
df.drop(['name'] , axis = 1, inplace = True)
```

In [103]:
```
df.head()
```

Out[103]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa

In [104]:
```
df.describe()
```

Out[104]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	398.000000	398.000000	398.000000	392.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	104.469388	2970.424623	15.568090	76.010050
std	7.815984	1.701004	104.269838	38.491160	846.841774	2.757689	3.697627
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.500000	4.000000	104.250000	75.000000	2223.750000	13.825000	73.000000
50%	23.000000	4.000000	148.500000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	262.000000	126.000000	3608.000000	17.175000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

Data Preprocessing

In [105]:
```
df.isnull().sum()
```

Out[105]:
```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
dtype: int64
```

In [106]:
```
df.dropna(inplace=True)
```

In [107]:
```
df.head()
```

Out[107]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	70	usa
1	15.0	8	350.0	165.0	3693	11.5	70	usa
2	18.0	8	318.0	150.0	3436	11.0	70	usa
3	16.0	8	304.0	150.0	3433	12.0	70	usa
4	17.0	8	302.0	140.0	3449	10.5	70	usa

In [108]:
```
df.isnull().sum().any()
```

Out[108]:
```
False
```

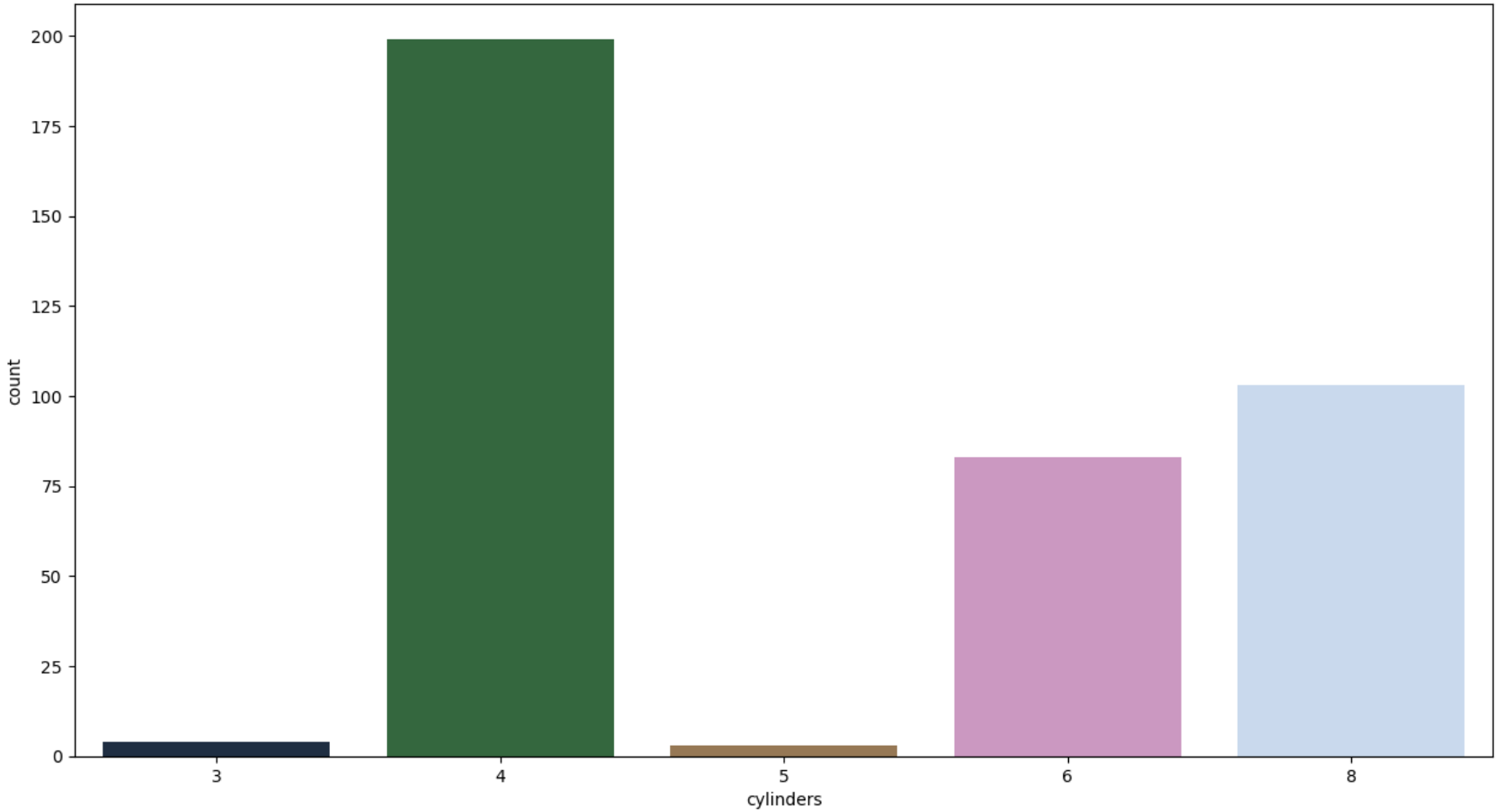
In [109]:
```
df.shape
```

Out[109]:
```
(392, 8)
```

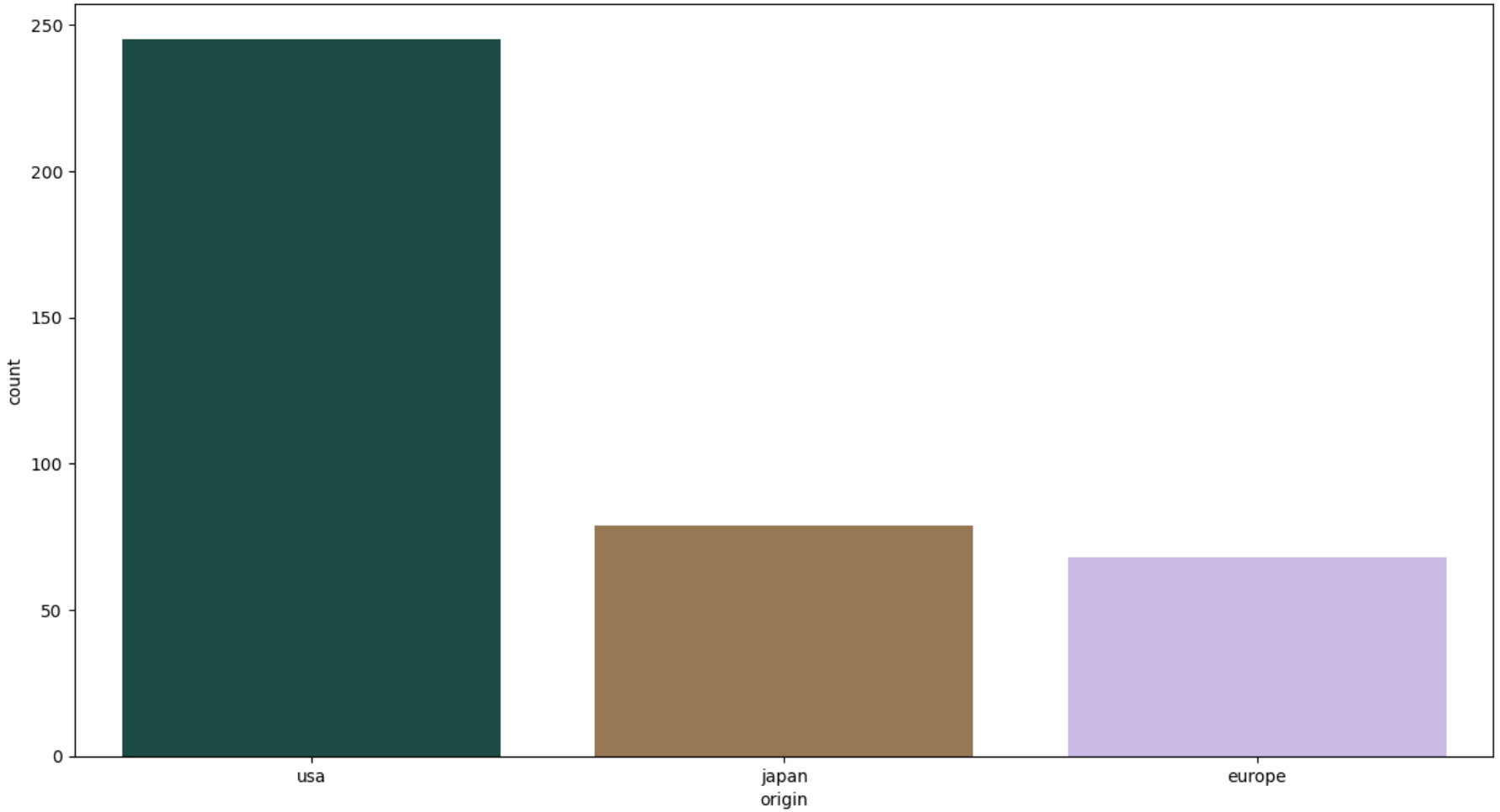
Exploratoory Data Analysis

Univariate Data Analysis

In [110]:
```
plt.figure(figsize = (15, 8))
sns.countplot(x=df["cylinders"], data = df, palette='cubehelix')
plt.show()
```



In [111]:
```
plt.figure(figsize = (15, 8))
sns.countplot(x=df["origin"], data = df, palette='cubehelix')
plt.show()
```



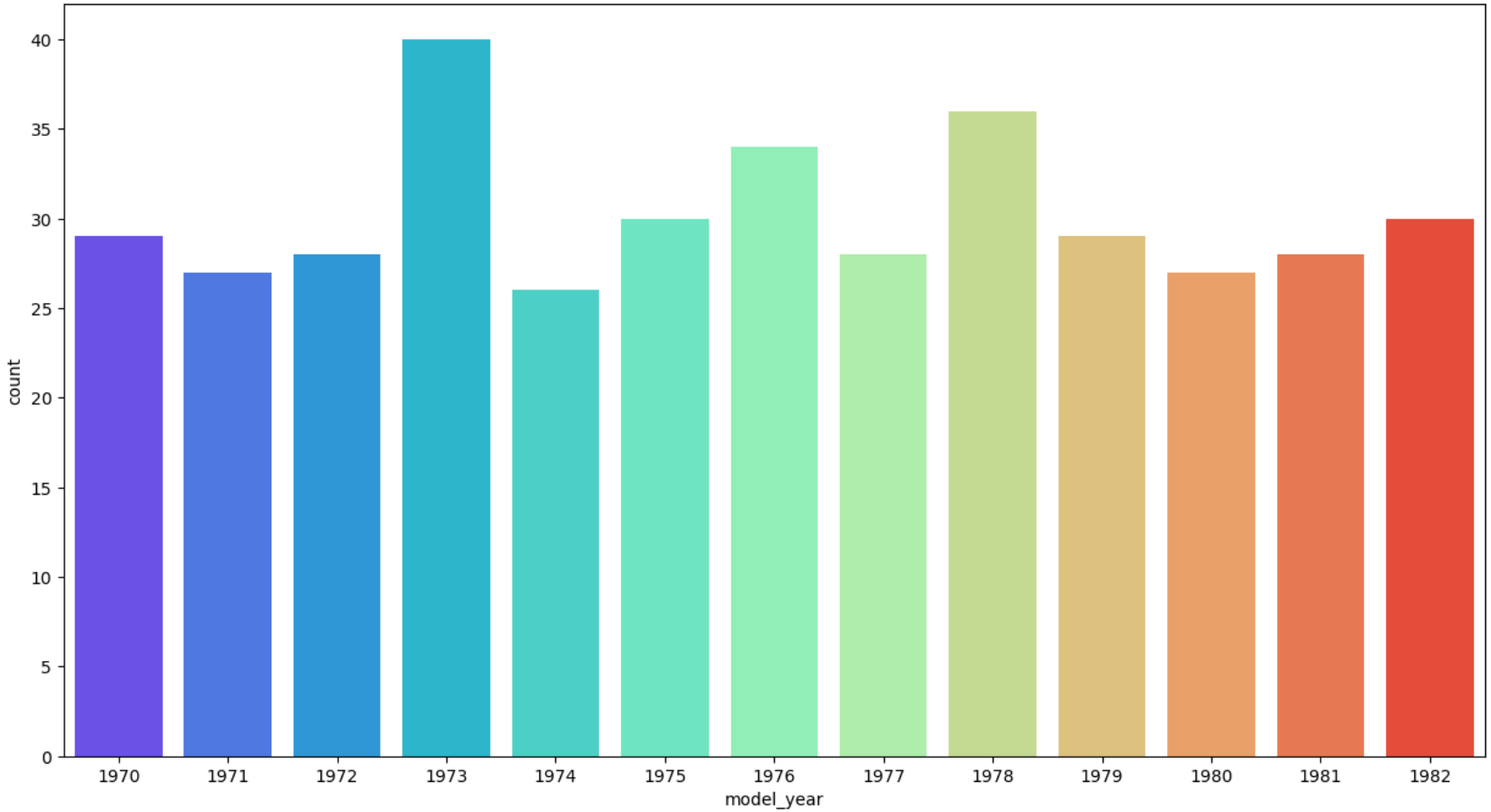
In [112]:
```
df['model_year'] = 1900 + df['model_year']
```

In [113]:
```
df.head()
```

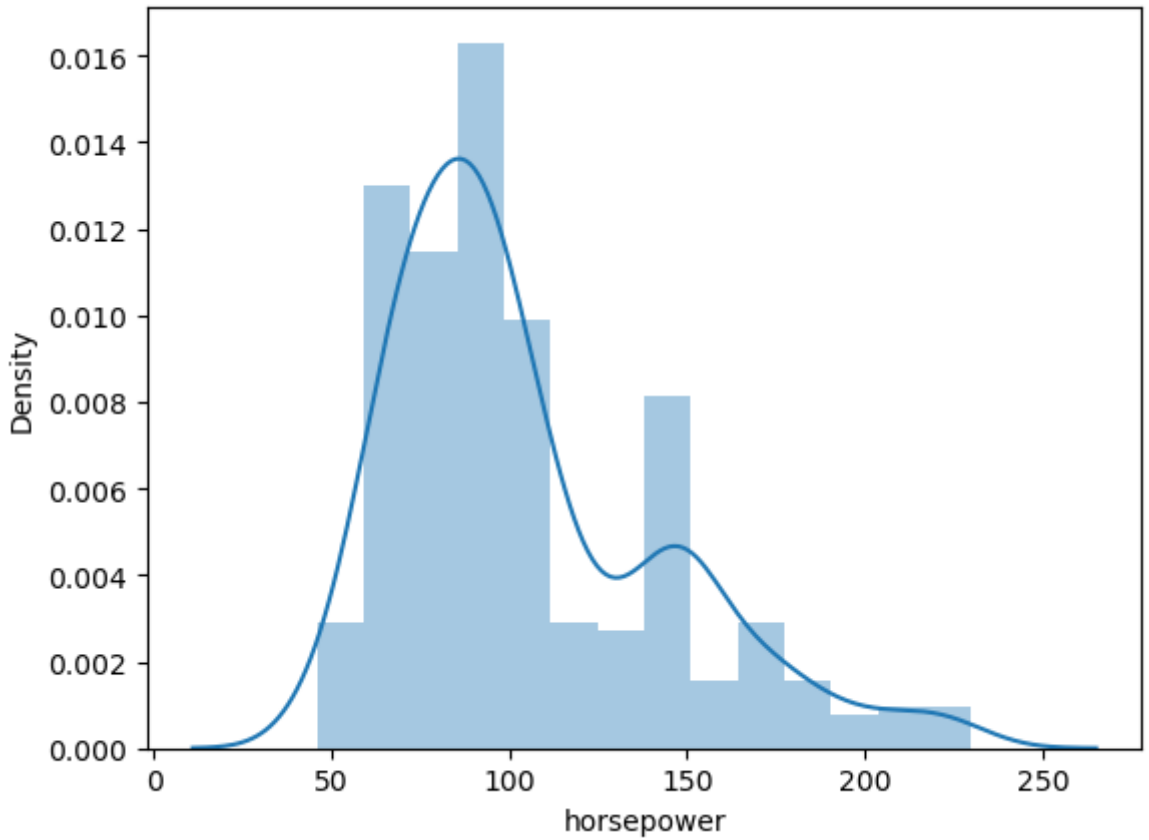
Out[113]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
0	18.0	8	307.0	130.0	3504	12.0	1970	usa
1	15.0	8	350.0	165.0	3693	11.5	1970	usa
2	18.0	8	318.0	150.0	3436	11.0	1970	usa
3	16.0	8	304.0	150.0	3433	12.0	1970	usa
4	17.0	8	302.0	140.0	3449	10.5	1970	usa

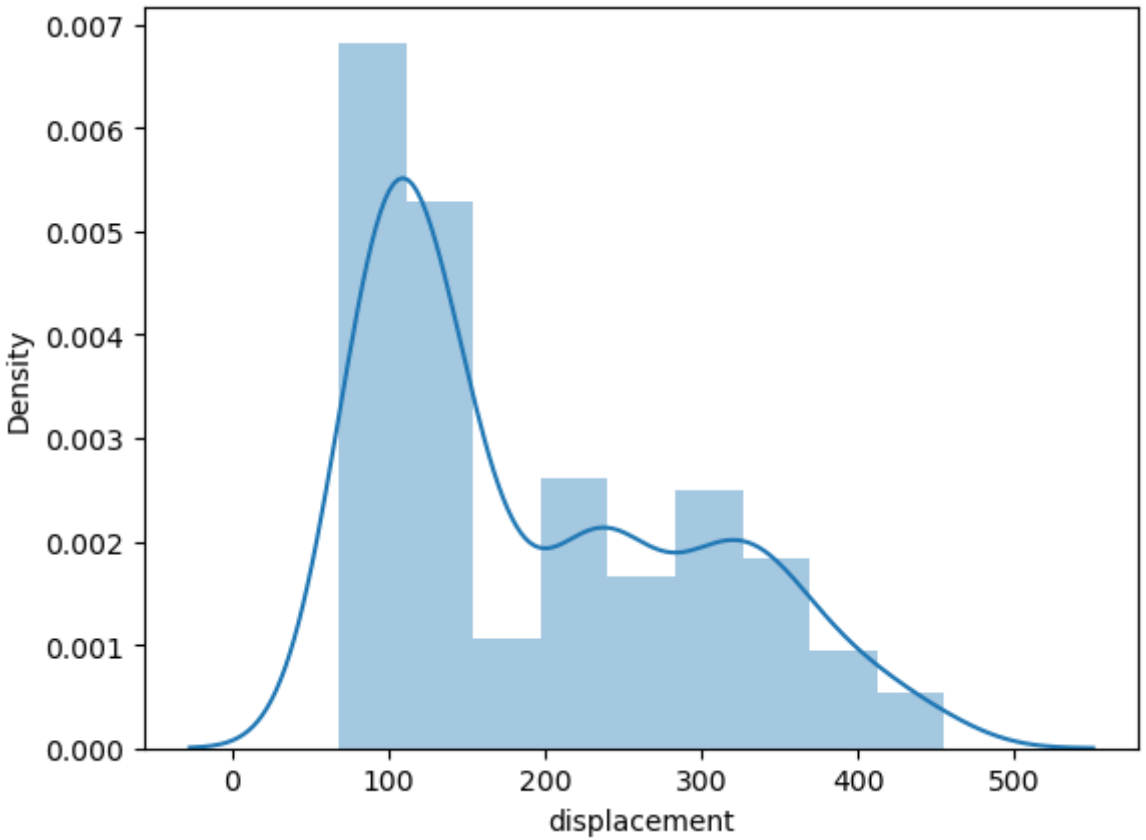
In [114]:
```
plt.figure(figsize = (15, 8))
sns.countplot(x=df["model_year"], data = df, palette='rainbow')
plt.show()
```



In [115]:
```
sns.distplot(df['horsepower'])
plt.show()
```

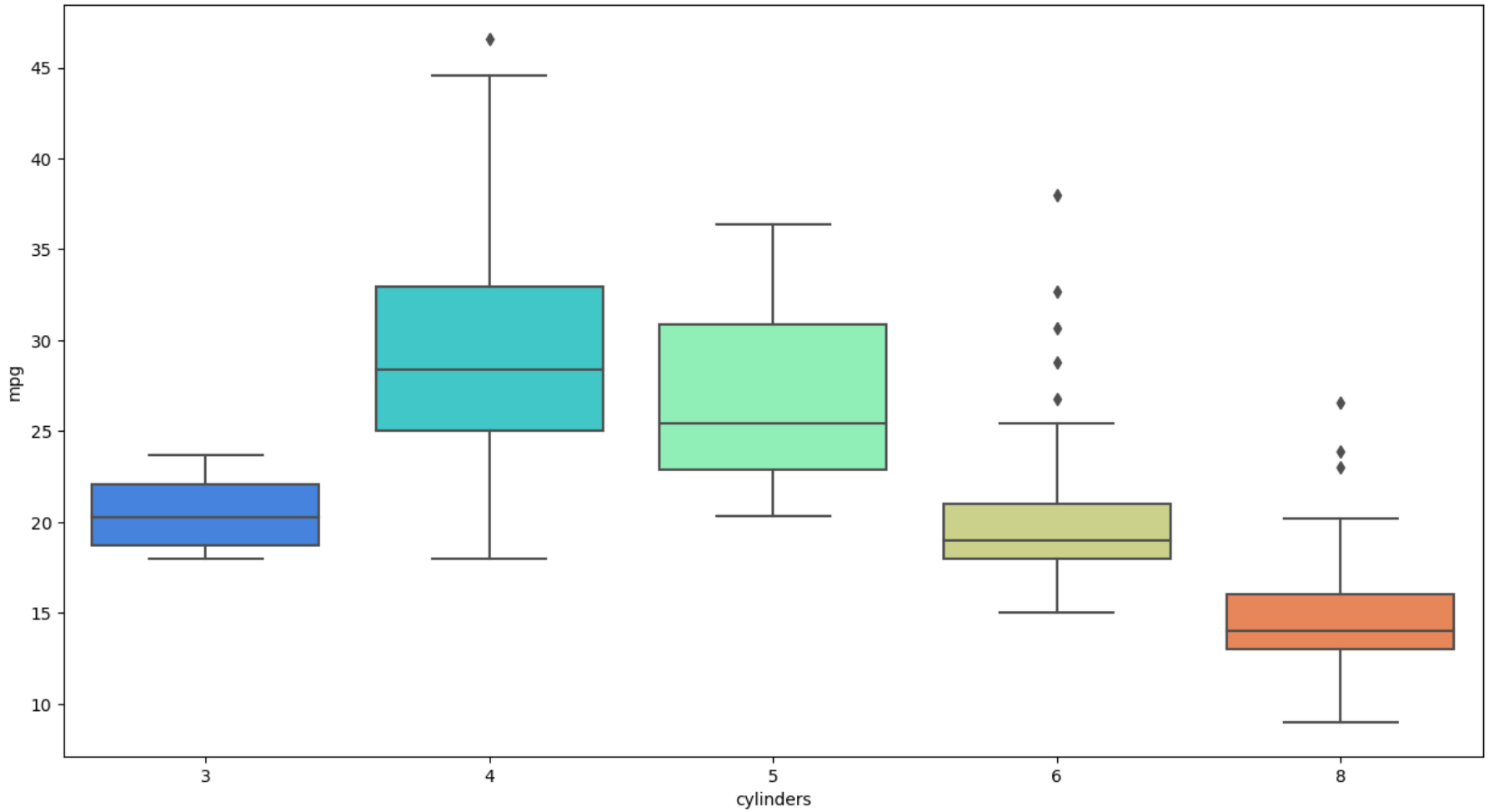


In [116]:
```
sns.distplot(df['displacement'])
plt.show()
```

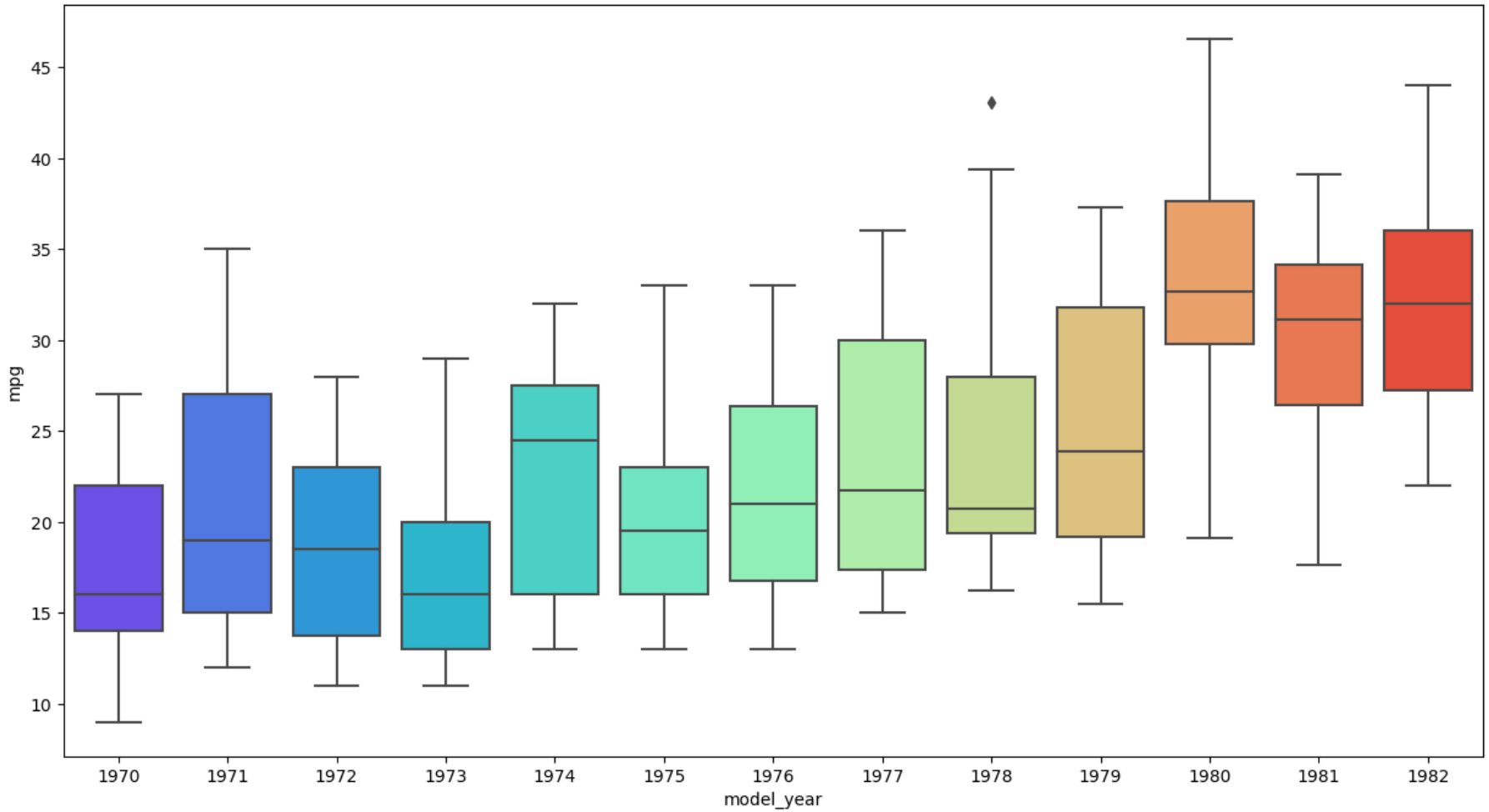


Bivariate Analysis

In [117]:
```
plt.figure(figsize=(15,8))
sns.boxplot(x='cylinders', y = 'mpg', data = df , palette='rainbow')
plt.show()
```

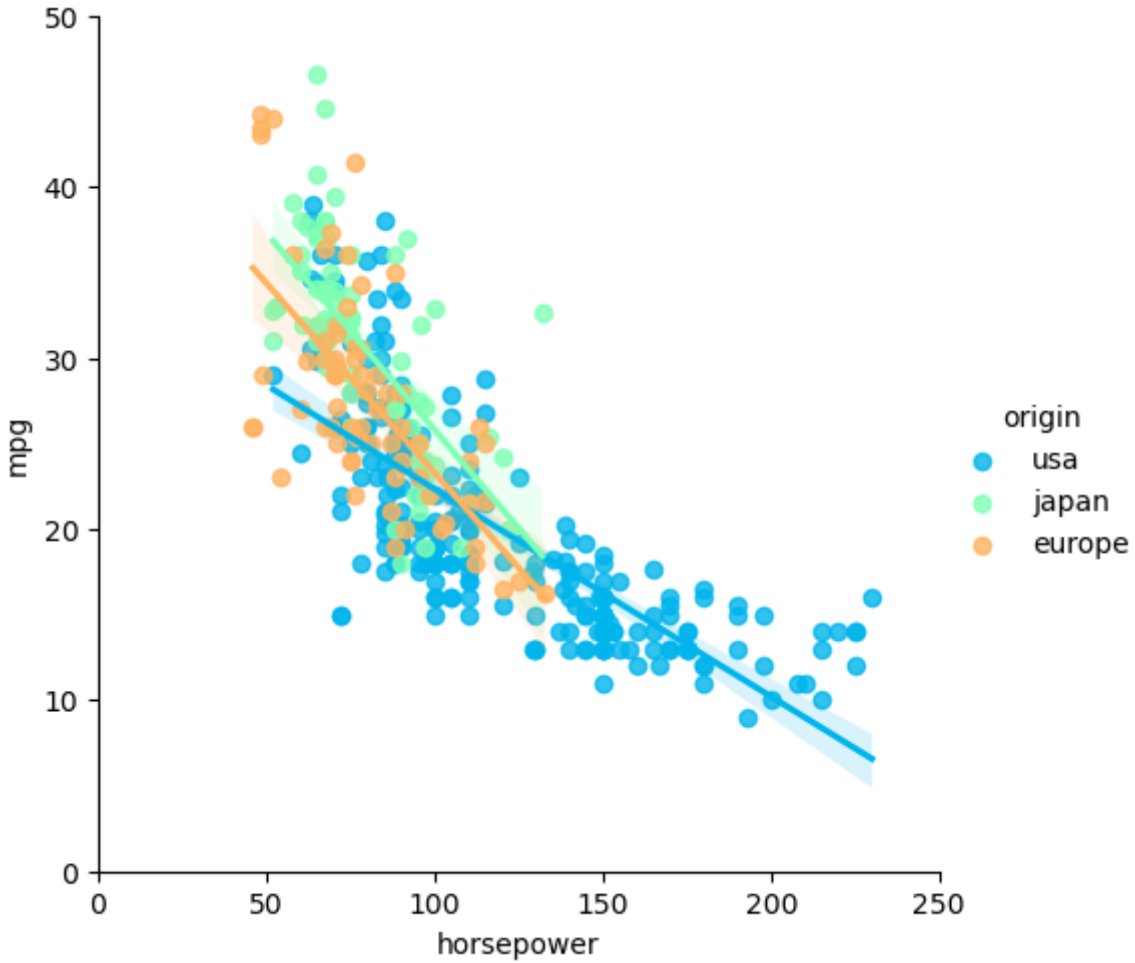


In [118]:
```
plt.figure(figsize=(15,8))
sns.boxplot(x='model_year', y = 'mpg', data = df , palette='rainbow')
plt.show()
```

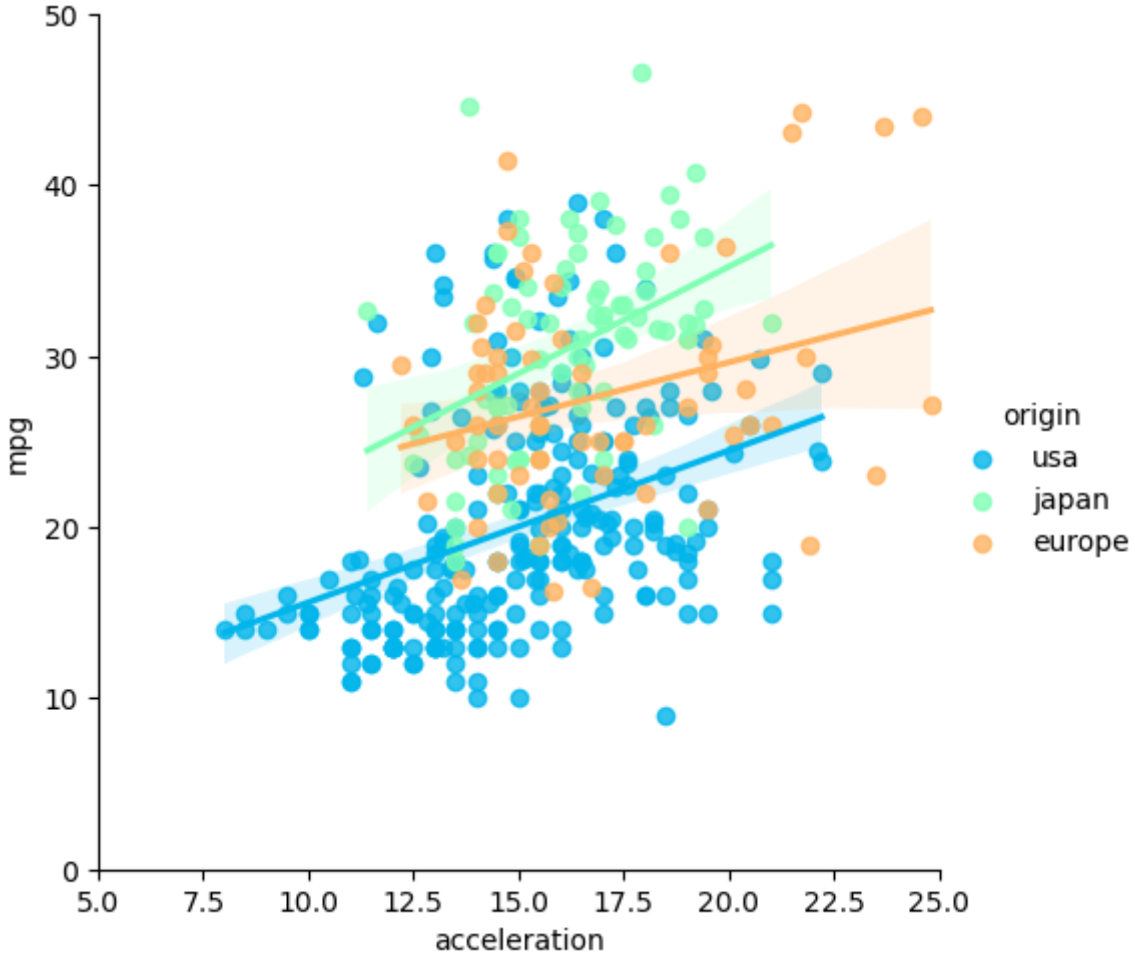


Multivariate Data Analysis

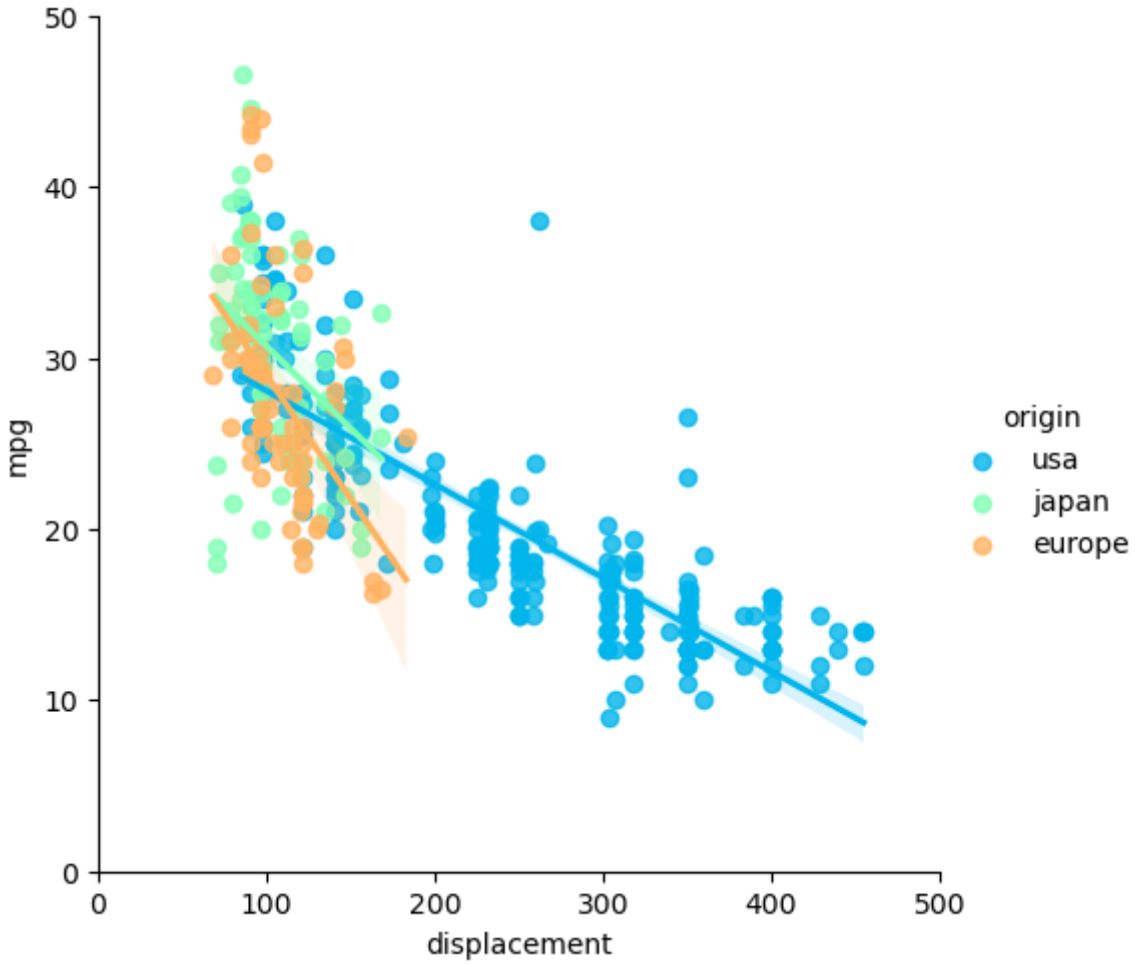
In [119]:
```
graph = sns.lmplot(x = "horsepower", y = "mpg", hue = "origin", data = df, palette = "rainbow")
graph.set(xlim = (0, 250))
graph.set(ylim = (0, 50))
plt.show()
```



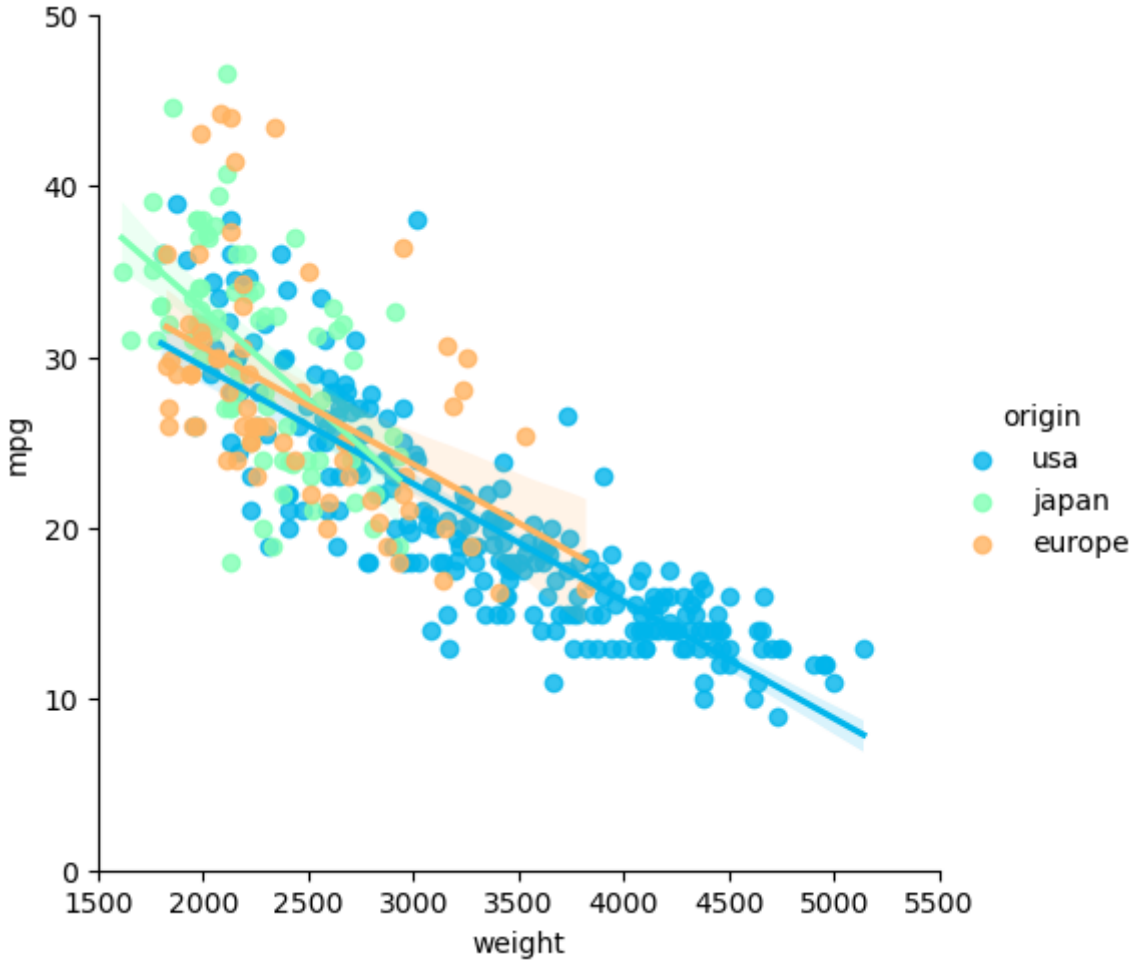
In [120]:
```
graph = sns.lmplot(x = "acceleration", y = "mpg", hue = "origin", data = df, palette = "rainbow")
graph.set(xlim = (5, 25))
graph.set(ylim = (0, 50))
plt.show()
```



In [121]:
```
graph = sns.lmplot(x = "displacement", y = "mpg", hue = "origin", data = df, palette = "rainbow")
graph.set(xlim = (0,500))
graph.set(ylim = (0, 50))
plt.show()
```

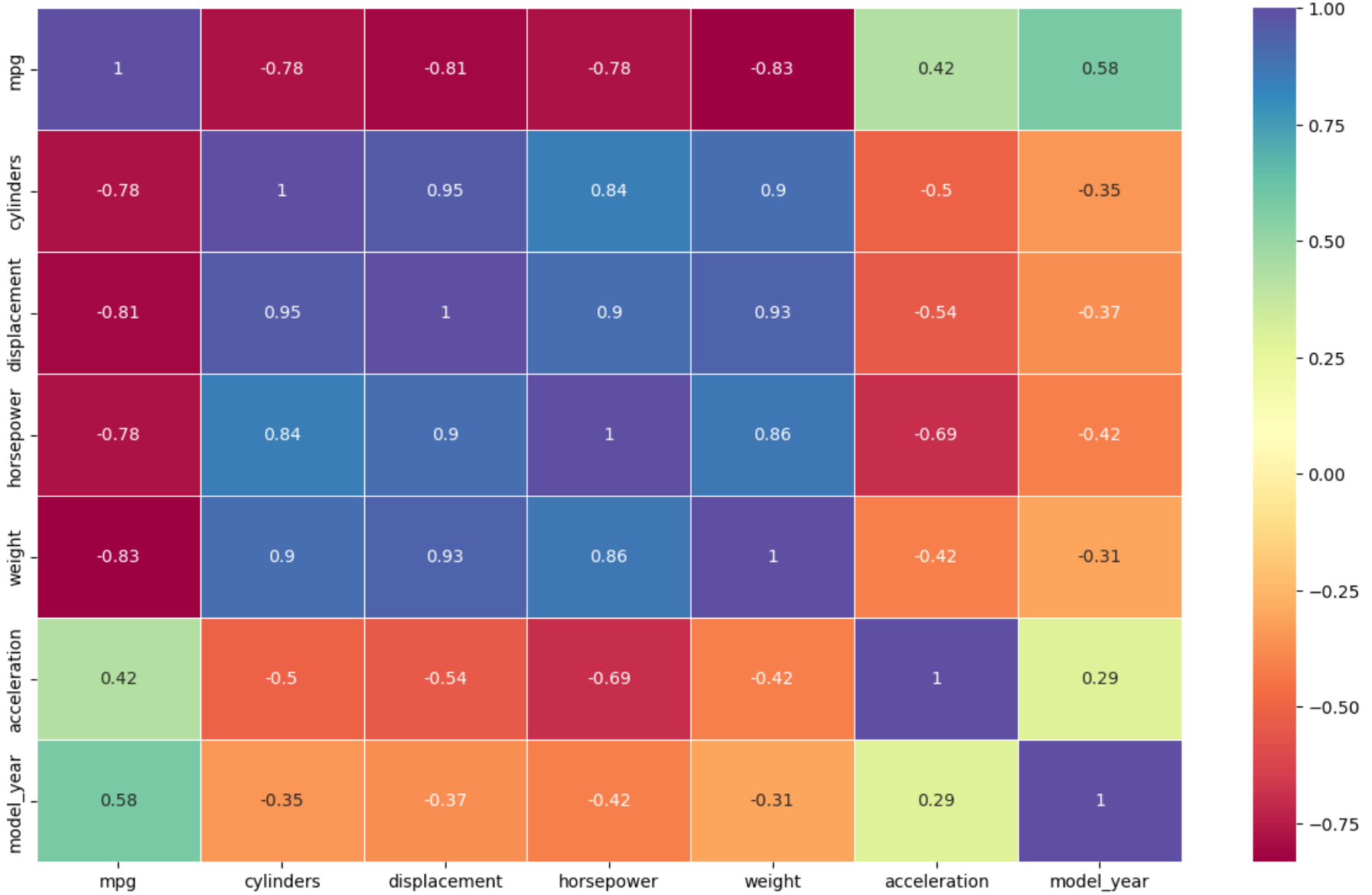


In [122]:
```
graph = sns.lmplot(x = "weight", y = "mpg", hue = "origin", data = df, palette = "rainbow")
graph.set(ylim = (0, 50))
graph.set(xlim = (1500, 5500))
plt.show()
```



Heat map of correlation matrix

In [123]:
```
plt.figure(figsize = (15, 9))
sns.heatmap(df.corr(), annot = True, linewidth = 0.5, cmap = "Spectral")
plt.show()
```



In [124]:
```
df.drop(['acceleration','displacement'],axis=1,inplace=True)
```

In [125]:
```
df.head()
```

Out[125]:

	mpg	cylinders	horsepower	weight	model_year	origin
0	18.0	8	130.0	3504	1970	usa
1	15.0	8	165.0	3693	1970	usa
2	18.0	8	150.0	3436	1970	usa
3	16.0	8	150.0	3433	1970	usa
4	17.0	8	140.0	3449	1970	usa

In [126]:
```
df = pd.get_dummies(df, drop_first = True)
```

In [127]:
```
df.head()
```

Out[127]:

	mpg	cylinders	horsepower	weight	model_year	origin_japan	origin_usa
0	18.0	8	130.0	3504	1970	0	1
1	15.0	8	165.0	3693	1970	0	1
2	18.0	8	150.0	3436	1970	0	1
3	16.0	8	150.0	3433	1970	0	1
4	17.0	8	140.0	3449	1970	0	1

Modeling

In [128]:
```
X = df.drop(["mpg"], axis = 1)
y = df["mpg"]
```

In [129]:
```
from sklearn.model_selection import train_test_split
```

In [130]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [131]:
```
X_train
```

Out[131]:

	cylinders	horsepower	weight	model_year	origin_japan	origin_usa
260	6	110.0	3620	1978	0	1
184	4	92.0	2572	1976	0	1
174	6	97.0	2984	1975	0	1
64	8	150.0	4135	1972	0	1
344	4	64.0	1875	1981	0	1
...	...	...	...	...	...	...
72	8	150.0	3892	1972	0	1
107	6	100.0	2789	1973	0	1
272	4	85.0	2855	1978	0	1
352	4	65.0	2380	1981	0	1
103	8	150.0	4997	1973	0	1

313 rows × 6 columns

In [132]:
```
y_train
```

Out[132]:
```
260    18.6
184    25.0
174    18.0
64     15.0
344    39.0
       ...
72     15.0
107    18.0
272    23.8
352    29.9
103    11.0
Name: mpg, Length: 313, dtype: float64
```

In [133]:
```
from sklearn.linear_model import LinearRegression
```

In [135]:
```
reg=LinearRegression()
reg.fit(X_train, y_train)
```

Out[135]:
```
▾ LinearRegression

LinearRegression()
```

In [136]:
```
reg.intercept_
```

Out[136]:
```
-1498.7218785122882
```

In [137]:
```
coef_param = pd.DataFrame(reg.coef_, index = X.columns, columns = ["Coefficient"])
coef_param
```

Out[137]:

	Coefficient
cylinders	0.203056
horsepower	-0.014143
weight	-0.005729
model_year	0.779904
origin_japan	0.401853
origin_usa	-2.385047

In [139]:
```
y_pred = reg.predict(X_test)
```

In [140]:
```
my_dict = {"Actual" : y_test, "Pred" : y_pred}
compare = pd.DataFrame(my_dict)
```

In [141]:
```
compare.sample(10)
```

Out[141]:

	Actual	Pred
205	28.0	30.176571
241	22.0	27.270363
79	26.0	26.545174
40	14.0	11.747138
43	13.0	8.115192
270	21.1	29.391116
115	15.0	13.832570
83	28.0	24.147781
47	19.0	17.086203
291	19.2	21.527543

In [142]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [143]:
```
def evaluation_metrics(actual, pred):
    MAE = mean_absolute_error(actual, pred)
    MSE = mean_squared_error(actual, pred)
    RMSE = np.sqrt(mean_squared_error(actual, pred))
    SCORE = r2_score(actual, pred)
    return print("r2_score:", SCORE, "\n","mae:", MAE, "\n","mse:",MSE, "\n","rmse:",RMSE)
```

In [144]:
```
evaluation_metrics(y_test, y_pred)
```

```
r2_score: 0.7798249880881908
 mae: 2.5188281576150886
 mse: 11.237861022823058
 rmse: 3.35229190596867
```

In [ ]: