./visuals/fig_1_47

February 6, 2018

1 Description

```
# Cars scatter plot

"Interactive Data Visualization: Foundations, Techniques, and Applications, Second Edition"
by M. Ward, G. Grinstein, and D. Kim.
Figure 1.47 in on page 45

Original data retrieved here:
[http://www.idvbook.com/teaching-aid/data-sets/2004-cars-and-trucks-data/]

Data converted to CSV with column headers available in this repository:
[../../data/cars04.csv]
```

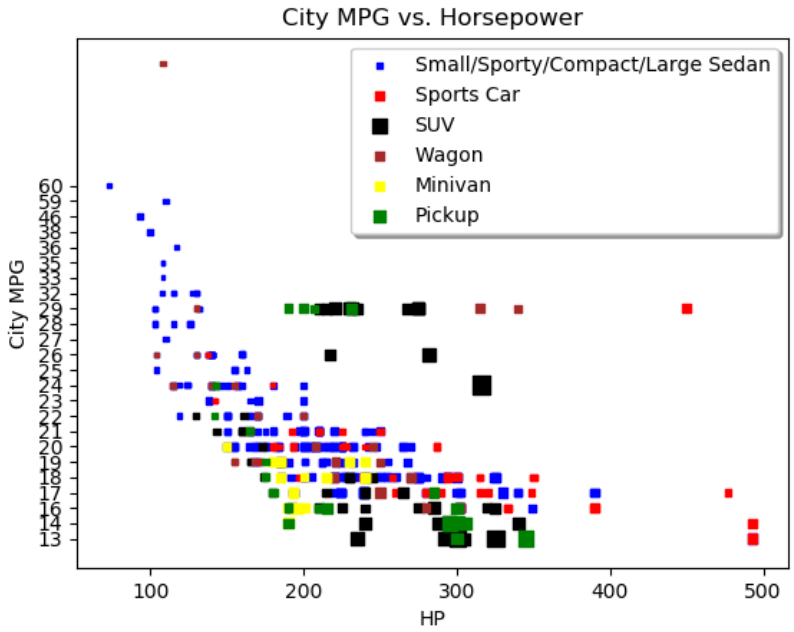
2 ./visuals/fig_1_47/caldwellc1

2.1 ./visuals/fig_1_47/caldwellc1/fig_1_47.py

```
import matplotlib.pyplot as plt
import sys
import pandas as pd
import numpy as np

# city MPG vs Horsepower
# color for vehicle type
# size for weight (area proportional to weight


def main():
    #car_data = pd.read_csv('cars04.csv')
    car_data = pd.read_csv(sys.argv[1])
    car_data = car_data.reset_index()
    car_data = car_data[['Small/Sporty/ Compact/Large Sedan','Sports Car','SUV','Wagon', 'Minivan','Pi
    car_data = car_data.rename(columns=lambda x:x.strip().replace(' ','_'))
    car_data = car_data.rename(columns=lambda x:x.strip().replace('/','_'))
    car_data = car_data.replace('*', np.nan)
    car_data = car_data.dropna(subset=['HP'])
    car_data = car_data.dropna(subset=['City_MPG'])
    car_data = car_data.dropna(subset=['Weight'])
    car_data = car_data.reset_index()

    small = car_data.drop(car_data[car_data.Small_Sporty__Compact_Large_Sedan < 1].index)
    small = small.reset_index()
    sport = car_data.drop(car_data[car_data.Sports_Car < 1].index)
    sport = sport.reset_index()
    suv = car_data.drop(car_data[car_data.SUV < 1].index)
    suv = suv.reset_index()
    wagon = car_data.drop(car_data[car_data.Wagon < 1].index)
    wagon = wagon.reset_index()
    minivan = car_data.drop(car_data[car_data.Minivan < 1].index)
    minivan = minivan.reset_index()
    pick = car_data.drop(car_data[car_data.Pickup < 1].index)
    pick = pick.reset_index()
    fig, ax = plt.subplots()
    ax.scatter(small['HP'], small['City_MPG'], s=[2**(float(n)/1000) for n in small['Weight']], c='blu
    ax.scatter(sport['HP'], sport['City_MPG'], s=[2**(float(n)/1000) for n in sport['Weight']], c='red
    ax.scatter(suv['HP'], suv['City_MPG'], s=[2**(float(n)/1000) for n in suv['Weight']], c='black', m
    ax.scatter(wagon['HP'], wagon['City_MPG'], s=[2**(float(n)/1000) for n in wagon['Weight']], c='bro
    ax.scatter(minivan['HP'], minivan['City_MPG'], s=[2**(float(n)/1000) for n in minivan['Weight']],
    ax.scatter(pick['HP'], pick['City_MPG'], s=[2**(float(n)/1000) for n in pick['Weight']], c='green'

    ax.set_xlabel('HP')
    ax.set_ylabel('City MPG')
    ax.set_title('City MPG vs. Horsepower')
```

```
    ax.legend(loc='upper right', shadow=True, markerscale=1)

    #plt.show()
    print(sys.argv[2])
    plt.savefig(sys.argv[2])


if __name__ == '__main__':
    main()
```

2.2 ./visuals/fig_1_47/caldwellc1/fig_1_47.png



3 ./visuals/fig_1_47/campellcl

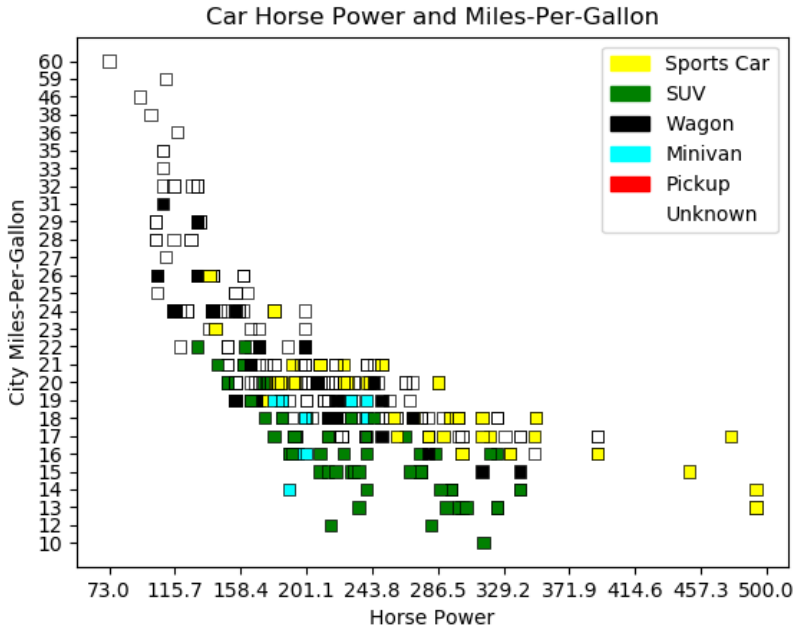
3.1 ./visuals/fig_1_47/campellcl/VisualizationOne.py

```
"""
VisualizationOne.py
Implementation of Programming Assignment One for CS5720.
"""

__author__ = "Chris Campell"
__version__ = "1/25/2018"

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib.patches as mpatches
from sklearn.preprocessing import normalize
from matplotlib import cm
import sys

# Load file:
with open(sys.argv[1], 'r') as fp:
    data = pd.read_csv(fp, header=0)

# Convert to dataframe:
df_cars = pd.DataFrame(data=data)

# Is there missing data?
df_cars.__str__().__contains__('*')

# Remove extraneous columns:
# Notice that 'Vehicle Name' is included because Figure 1.47 is only Toyotas
df_cars = df_cars[['Vehicle Name', 'HP', 'City MPG', 'Len', 'Width', 'Weight', 'Sports Car', 'SUV', 'W

# Remove records with an unknown HP, City MPG, Len, or Width:
df_cars = df_cars.replace(r'[*]', np.nan, regex=True)
df_cars = df_cars.dropna(axis=0, how='any')

# Add in column with vehicle area:
df_cars['Area'] = [int(l)*int(w) for l,w in zip(df_cars['Len'], df_cars['Width'])]

# Ensure all nan's have been dropped from 'HP':
# df = df[np.isfinite(df['HP'])]

# Filter by Toyota vehicles:
toyota_only = df_cars[df_cars['Vehicle Name'].str.contains('Toyota')]
toyota_hp_vs_mpg = toyota_only[['Vehicle Name', 'HP', 'City MPG', 'Area', 'Weight']]

# Create the scatter plot:
# Reference URL: https://stackoverflow.com/questions/17682216/scatter-plot-and-color-mapping-in-pythor
```

```
# https://stackoverflow.com/questions/4143502/how-to-do-a-scatter-plot-with-empty-circles-in-python
# http://nbviewer.jupyter.org/github/jvns/pandas-cookbook/blob/v0.1/cookbook/Chapter%207%20-%20Cleani
x = df_cars['HP']
y = df_cars['City MPG']
fig, ax = plt.subplots()
# Color based on vehicle type:
# https://stackoverflow.com/questions/26139423/plot-different-color-for-different-categorical-levels-u

def map_color_to_vehicle_type(df_row):
    if int(df_row['Sports Car']) == 1:
        color = 'Yellow'
    elif int(df_row['SUV']) == 1:
        color = 'Green'
    elif int(df_row['Wagon']) == 1:
        color = 'Black'
    elif int(df_row['Minivan']) == 1:
        color = 'Cyan'
    elif int(df_row['Pickup']) == 1:
        color = 'Red'
    else:
        # print("Vehicle type not identified")
        color = 'None'
    return color

def map_vehicle_type_to_string(df_row):
    if int(df_row['Sports Car']) == 1:
        vehicle_type = 'Sports'
    elif int(df_row['SUV']) == 1:
        vehicle_type = 'Sports'
    elif int(df_row['Wagon']) == 1:
        vehicle_type = 'Wagon'
    elif int(df_row['Minivan']) == 1:
        vehicle_type = 'Minivan'
    elif int(df_row['Pickup']) == 1:
        vehicle_type = 'Pickup'
    else:
        # print("Vehicle type not identified")
        vehicle_type = 'Unknown'
    return vehicle_type

df_cars['Color'] = df_cars.apply(map_color_to_vehicle_type, axis=1)
df_cars['Vehicle Type'] = df_cars.apply(map_vehicle_type_to_string, axis=1)

# y_min = int(toyota_hp_vs_mpg['City MPG'].min(0))
# y_max = int(toyota_hp_vs_mpg['City MPG'].max(0))
# x_min = int(toyota_hp_vs_mpg['HP'].min(0))
# x_max = int(toyota_hp_vs_mpg['HP'].max(0))

# Let the size of the marker represent the weight of the vehicle:
# https://stackoverflow.com/questions/14827650/pyplot-scatter-plot-marker-size
```

```
# size = [int(w) for w in df_cars['Area'].values]
# normalize:
# size = size / np.linalg.norm(size)
vehicle_scatter = plt.scatter(x, y, marker='s', facecolors=df_cars['Color'], edgecolor='black', linewi

# ax.scatter(x, y, marker='s', c='bue', facecolors='None')
# ax.scatter(toyota_only['HP'], toyota_only['City MPG'], marker='s', c='green')
# plt.axis(y=np.arange(10, 60, 5), x=np.arange(73, 500, 42.7))
plt.xticks(np.arange(73, 542.7, 42.7))
# ax.legend((df_cars['Sports Car'], df_cars['SUV'], df_cars['Wagon'], df_cars['Minivan'], df_cars['Pic
yellow_patch = mpatches.Patch(color='yellow', label='Sports Car')
green_patch = mpatches.Patch(color='green', label='SUV')
black_patch = mpatches.Patch(color='black', label='Wagon')
cyan_patch = mpatches.Patch(color='cyan', label='Minivan')
purple_patch = mpatches.Patch(color='red', label='Pickup')
none_patch = mpatches.Patch(color='none', label='Unknown')
plt.legend(handles=[yellow_patch, green_patch, black_patch, cyan_patch, purple_patch, none_patch])
plt.xlabel('Horse Power')
plt.ylabel('City Miles-Per-Gallon')
plt.title('Car Horse Power and Miles-Per-Gallon')
plt.savefig(fname=sys.argv[2])
# plt.show()
```

3.2 ./visuals/fig_1_47/campellcl/fig_1_47.png



4 ./visuals/fig_1_47/wascherb

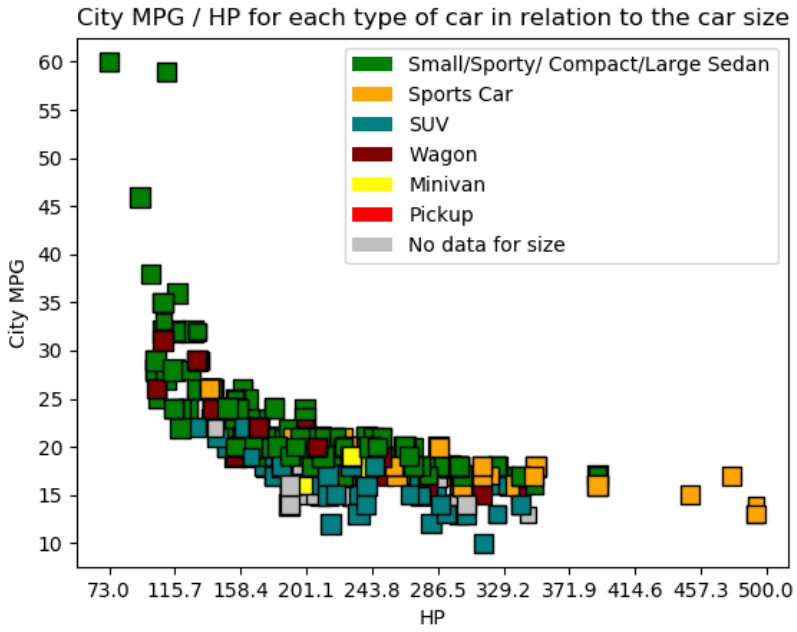
4.1 ./visuals/fig_1_47/wascherb/fig_1_41.py

```
import matplotlib.pylab as plt
import matplotlib.patches as mpatches
import numpy as np
import sys


def results(file_name):
    """
    clean data for representation
    :param file_name: input filename for data
    :return: None
    """

    x = []
    y = []
    car_type = []
    size = []
    leg_names = []
    cnt = 0
    index_x, index_y = 0, 0
    with open(file_name) as f:
        for line in f:
            content = str(line).strip()
            elements = content.split(',')

            if cnt != 0:
                last_element = len(elements) - 1
                try:
                    if elements[index_x].isdigit() and elements[index_y].isdigit():
                        x.append(int(elements[index_x]))
                        y.append(int(elements[index_y]))
                        if str(elements[last_element])[:-1].isdigit() and elements[last_element - 1].i
                                elements[last_element - 3].isdigit():
                            car_type.append(elements.index('1'))
                            area = int(str(elements[last_element])[:-1]) * int(elements[last_element -
                            weight = int(elements[last_element - 3])
                            size.append(area / weight * 250)
                        else:
                            car_type.append(7)
                            # print('%-30s: %s' % (elements[0], elements.index('1')))
                            # size.append(0.3567027132923991 * 250)
                            # print('FAULT')
                except:
                    print(elements[index_x])
                    print(elements[index_y])
            else:
```

```
                index_x = elements.index('HP')
                index_y = elements.index('City MPG')
                for i in range(1, 7):
                    leg_names.append(elements[i])
                leg_names.append('No data for size')
            cnt += 1

    plot(x, y, car_type, size, leg_names)


def plot(x, y, car_type, size, leg_names):
    """
    creating the scatter plot of the data
    :param x: HP data
    :param y: MPG data
    :param car_type: type classes
    :param size: sizes of rectangles
    :param leg_names: legend names extracted from the data sorces
    :return: None
    """

    plt.title('City MPG / HP for each type of car in relation to the car size')

    color_map = {1: 'green', 2: 'orange', 3: 'teal', 4: 'maroon', 5: 'yellow', 6: 'red', 7: 'silver'}
    colors = []
    for index, type in enumerate(car_type):
        colors.append(color_map[type])
        car_type[index] = color_map[type]

    plt.scatter(x, y, color=colors, s=size, marker='s', edgecolors='black')
    plt.xlabel('HP')
    plt.ylabel('City MPG')

    # scale steps
    plt.yticks(np.arange(10, 65, 5))
    plt.xticks(np.arange(min(x), max(x) + 42.7, 42.7))

    # Add legend
    recs = []
    for i in color_map.values():
        recs.append(mpatches.Rectangle((0, 0), 1, 1, fc=i))
    plt.legend(recs, leg_names, loc=1)

    # plt.show()
    plt.savefig(sys.argv[2])


if __name__ == '__main__':
    if sys.argv == 1:
        print('Accept one argument: No input file for data')
```

```
        exit(0)

results(sys.argv[1])
```

4.2 ./visuals/fig_1_47/wascherb/fig_1_47.png



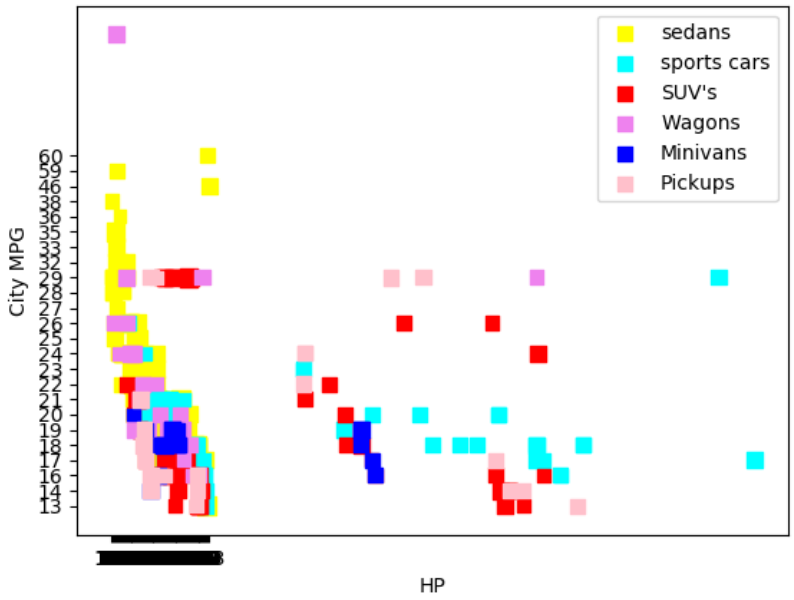
5 ./visuals/fig_1_47/stokesnl

5.1 ./visuals/fig_1_47/stokesnl/fig_1_47.py

```
#!/usr/bin/env python
import csv
import matplotlib.pyplot as plt
from collections import defaultdict
import sys

columns = defaultdict(list) # each value in each column is appended to a list

with open(sys.argv[1]) as f:
    reader = csv.DictReader(f) # read rows into a dictionary format
    for row in reader: # read a row as {column1: value1, column2: value2,...}
        for (k,v) in row.items(): # go over each column name and value
            columns[k].append(v) # append the value into the appropriate list
                                 # based on column name k
#test = ['225', '125', '231']
toremove = []
for i in range(1,len(columns['HP'])):
    if columns['City MPG'][i] == '*':
        toremove.append(i)
for i in reversed(toremove):
    del columns['HP'][i]
    del columns['City MPG'][i]
    del columns['Weight'][i]
    del columns['Small/Sporty/ Compact/Large Sedan'][i]
    del columns['Sports Car'][i]
    del columns['SUV'][i]
    del columns['Wagon'][i]
    del columns['Minivan'][i]
    del columns['Pickup'][i]
toremove = []
for i in range(1,len(columns['Weight'])):
    if columns['Weight'][i] == '*':
        toremove.append(i)
for i in reversed(toremove):
    del columns['HP'][i]
    del columns['City MPG'][i]
    del columns['Weight'][i]
    del columns['Small/Sporty/ Compact/Large Sedan'][i]
    del columns['Sports Car'][i]
    del columns['SUV'][i]
    del columns['Wagon'][i]
    del columns['Minivan'][i]
    del columns['Pickup'][i]
sedan = []
sports =[]
suv = []
```

```
wagon = []
minivan = []
pickup = []
for i in range(1,len(columns['Weight'])):
    if columns['Small/Sporty/ Compact/Large Sedan'][i] == '1':
        sedan.append(i)
    elif columns['Sports Car'][i] == '1':
        sports.append(i)
    elif columns['SUV'][i] == '1':
        suv.append(i)
    elif columns['Wagon'][i] == '1':
        wagon.append(i)
    elif columns['Minivan'][i] == '1':
        minivan.append(i)
    elif columns['Pickup'][i] == '1':
        pickup.append(i)
x = [float(i) for i in columns['HP']]
y = [float(i) for i in columns['City MPG']]
weight = [float(i) for i in columns['Weight']]
weight = [x / 70 for x in weight]
#[float(i) for i in columns['City MPG']]
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.scatter([columns['HP'][i] for i in sedan], [columns['City MPG'][i] for i in sedan],
s = weight, marker="s", c="yellow", edgecolors="face", label="sedans")

ax1.scatter([columns['HP'][i] for i in sports], [columns['City MPG'][i] for i in sports],
s = weight, marker="s", c="cyan",edgecolors="face", label="sports cars")

ax1.scatter([columns['HP'][i] for i in suv], [columns['City MPG'][i] for i in suv],
s = weight, marker="s", c="r",edgecolors="face", label="SUV's")

ax1.scatter([columns['HP'][i] for i in wagon], [columns['City MPG'][i] for i in wagon],
s = weight, marker="s", c="violet",edgecolors="face", label="Wagons")

ax1.scatter([columns['HP'][i] for i in minivan], [columns['City MPG'][i] for i in minivan],
s = weight, marker="s", c="b",edgecolors="face", label="Minivans")

ax1.scatter([columns['HP'][i] for i in pickup], [columns['City MPG'][i] for i in pickup],
s = weight, marker="s", c="pink",edgecolors="face", label="Pickups")
ax1.legend()
#plt.scatter(x, y, s=weight, marker="s");
plt.ylabel("City MPG")
plt.xlabel("HP")
plt.savefig(sys.argv[2]);
#print(columns['Vehicle Name'])
```

5.2 ./visuals/fig_1_47/stokesnl/fig_1_47.png

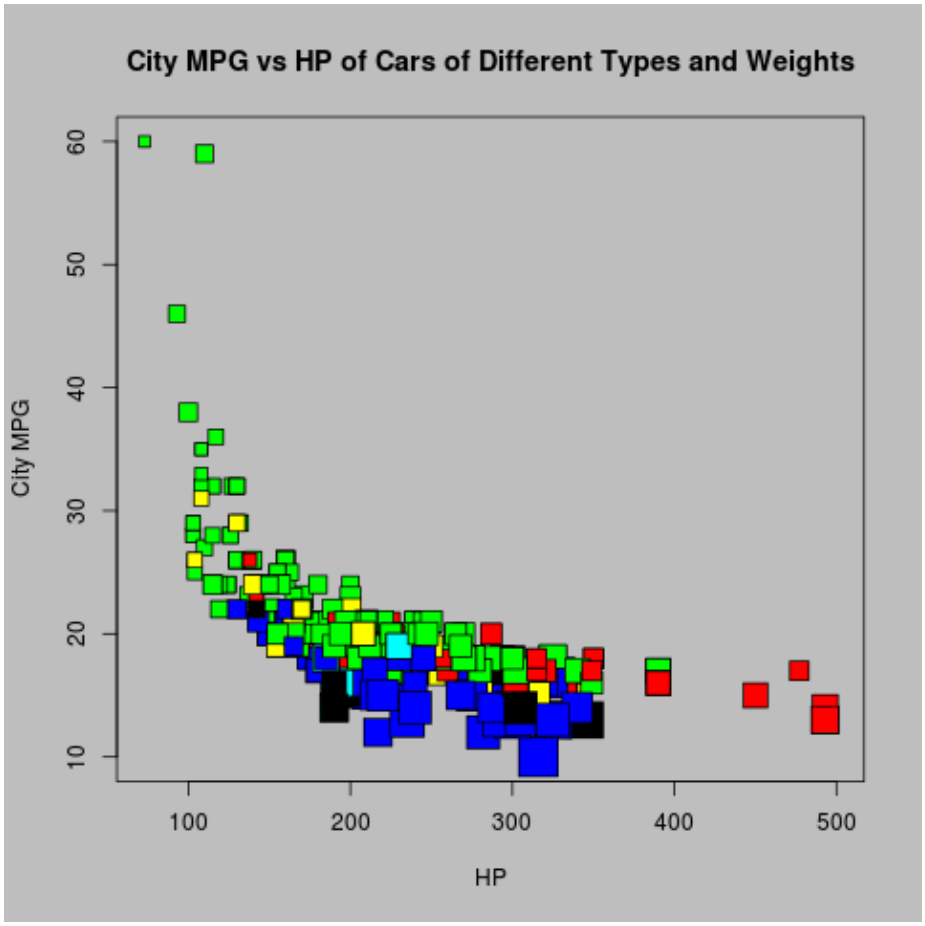


6 ./visuals/fig_1_47/davisjb2

6.1 ./visuals/fig_1_47/davisjb2/fig_1_41.R

```
args <- commandArgs()
x <- read.table(args[6],header=T,sep=",")
str(x)
ex <- as.numeric(as.character(x$City.MPG))
qu <- as.numeric(as.character(x$HP))
w <- as.numeric(as.character(x$Weight))
x$Color = "black"
x$Color[x$Small.Sporty..Compact.Large.Sedan == 1] = "green"
x$Color[x$Sports.Car == 1] = "red"
x$Color[x$SUV == 1] = "blue"
x$Color[x$Wagon == 1] = "yellow"
x$Color[x$Minivan == 1] = "cyan"
x$Color[x$Pickup == 1] = "black"

png(filename=args[7])
par(bg = "grey")
plot(qu,ex,pch=22,xlab = "HP",ylab = "City MPG", main = 'City MPG vs HP of Cars of Different Types and
dev.off()
```

6.2 ./visuals/fig_1_47/davisjb2/fig_1_47.png



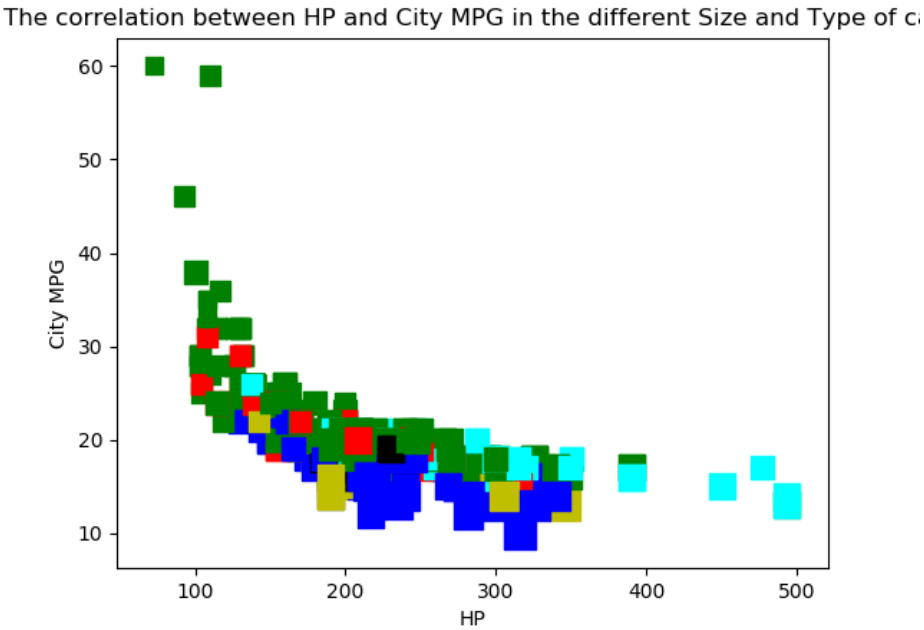
7 ./visuals/fig_1_47/zhengn

7.1 ./visuals/fig_1_47/zhengn/fig_1_41.py

```
__author__ = 'Naibin Zheng'
import numpy as np
import matplotlib.pyplot as plt
import csv
import pandas as pd
from matplotlib.patches import Rectangle
import sys


def main():
    data = pd.read_csv(sys.argv[1])
    hp = data['HP']
    mpg = data['City MPG']
    weight = data['Weight']
    sedan = data['Small/Sporty/ Compact/Large Sedan']
    sc = data['Sports Car']
    suv = data['SUV']
    wagon = data['Wagon']
    minivan = data['Minivan']
    pickup=data['Pickup']


    #print(type)

    plt.xlabel('HP')
    plt.ylabel('City MPG')
    plt.title('The correlation between HP and City MPG in the different Size and Type of car')

    #for h, m, w, s, su, w, m, p in zip(hp, mpg, weight, sc, suv, wagon, minivan, pickup):
    for h, m, w, sd, s, su, wg, mv, p in zip(hp, mpg, weight, sedan, sc, suv, wagon, minivan, pickup):
        if h != '*' and m != '*' and w != '*' and sd == 1:
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='g')
        elif h != '*' and m != '*' and w != '*' and s == 1:
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='cyan')
        elif h != '*' and m != '*' and w != '*' and su == 1:
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='blue')
        elif h != '*' and m != '*' and w != '*' and p == 1:
```

```
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='y')
        elif h != '*' and m != '*' and w != '*' and wg == 1:
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='r')
        elif h != '*' and m != '*' and w != '*' and mv == 1:
            area = float(w)/25
            h = float(h)
            m = float(m)
            plt.scatter(h, m, marker='s', s=area, c='black')

    #plt.show()
    plt.savefig(sys.argv[2])


if __name__ == '__main__':
    main()
```

7.2 ./visuals/fig_1_47/zhengn/fig_1_47.png

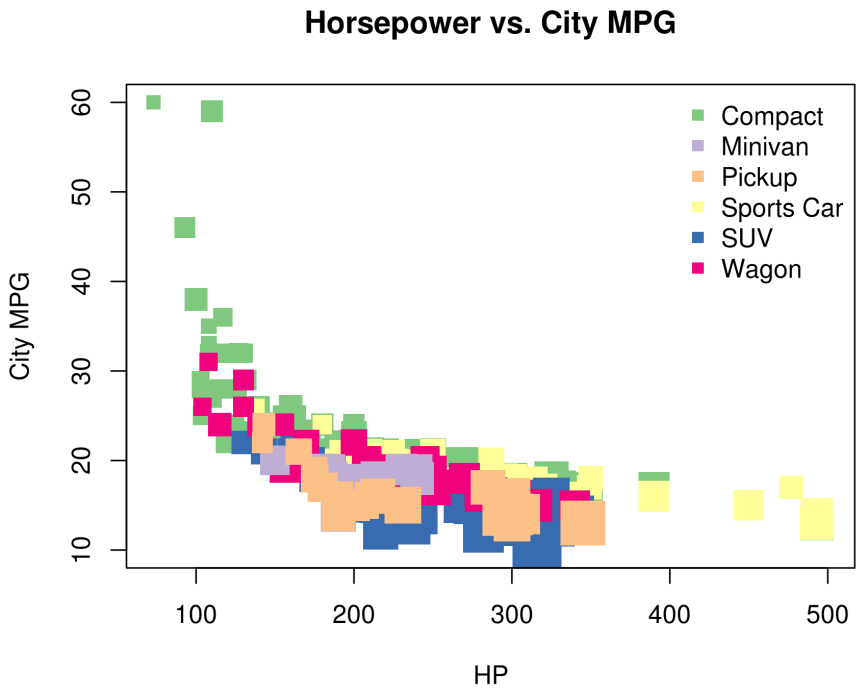


8 ./visuals/fig_1_47/smithkj2

8.1 ./visuals/fig_1_47/smithkj2/Program1.R

```
library(readr)
library(dplyr)
library(tidyr)
library(RColorBrewer)
t <- proc.time()

args <- commandArgs(T)
print(args)
name <- args[1]
cars <- read_csv(name)

cars <- cars %>%
  gather(class, value, c(`Small/Sporty/ Compact/Large Sedan`, `Sports Car`, SUV, Wagon, Minivan, Picku
  filter(value == 1)
cars$class <- ifelse(cars$class == 'Small/Sporty/ Compact/Large Sedan', 'Compact', cars$class)
cars$Weight[is.na(cars$Weight)] <- median(cars$Weight)
cars$Weight <- as.numeric(cars$Weight)

colors <- with(cars,
               data.frame(class = levels(factor(class)),
                          color = I(brewer.pal(nlevels(factor(cars$class)),
                                               name = 'Accent'))))
png(filename=args[2],
    width = 6,
    height = 5,
    units = 'in',
    res = 300)
plot(cars$HP,
     cars$`City MPG`,
     xlab = 'HP',
     ylab = 'City MPG',
     main = 'Horsepower vs. City MPG',
     col = colors$color[match(cars$class, colors$class)],
     pch = 15,
     cex = cars$Weight/1500)
legend(x = 'topright',
       legend = as.character(colors$class),
       col = colors$color,
       pch = 15,
       bty = 'n')

#dev.off()
t-proc.time()
```

8.2 ./visuals/fig_1_47/smithkj2/fig_1_47.png

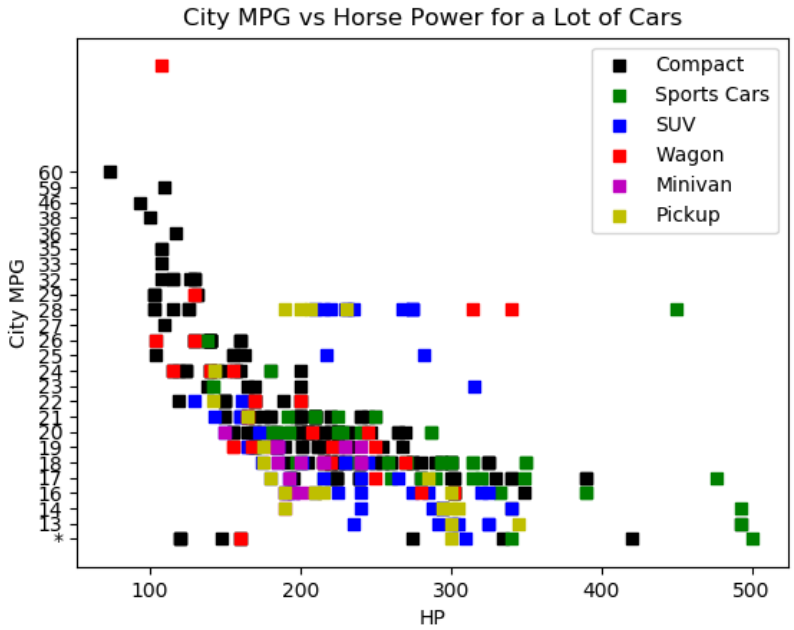


9 ./visuals/fig_1_47/carnsds

9.1 ./visuals/fig_1_47/carnsds/fig_1_47.py

```
#Dillon Carns
#1/28/2018
import matplotlib.pyplot as plt
import pandas as pand

import sys

fields = ['Small/Sporty/ Compact/Large Sedan','Sports Car','SUV','Wagon', 'Minivan',
'Pickup','HP', 'City MPG', 'Weight']
df = pand.read_csv(sys.argv[1], skipinitialspace=True, usecols=fields)


plt.title("City MPG vs Horse Power for a Lot of Cars")
plt.xlabel("HP")
plt.ylabel("City MPG")
plt.scatter(df.loc[df['Small/Sporty/ Compact/Large Sedan'] == 1, 'HP'],
 df.loc[df['Small/Sporty/ Compact/Large Sedan'] == 1, 'City MPG'], c='k', marker='s')
plt.scatter(df.loc[df['Sports Car'] == 1, 'HP'],
 df.loc[df['Sports Car'] == 1, 'City MPG'], c='g', marker='s')
plt.scatter(df.loc[df['SUV'] == 1, 'HP'],
 df.loc[df['SUV'] == 1, 'City MPG'], c='b', marker = 's')
plt.scatter(df.loc[df['Wagon'] == 1, 'HP'],
 df.loc[df['Wagon'] == 1, 'City MPG'], c='r', marker = 's')
plt.scatter(df.loc[df['Minivan'] == 1, 'HP'],
 df.loc[df['Minivan'] == 1, 'City MPG'], c='m', marker = 's')
plt.scatter(df.loc[df['Pickup'] == 1, 'HP'],
 df.loc[df['Pickup'] == 1, 'City MPG'], c='y', marker = 's')
plt.legend(['Compact', 'Sports Cars', 'SUV', 'Wagon', 'Minivan', 'Pickup'])
plt.savefig(sys.argv[2], Transparent=True)
```

9.2 ./visuals/fig_1_47/carnsds/fig_1_47.png

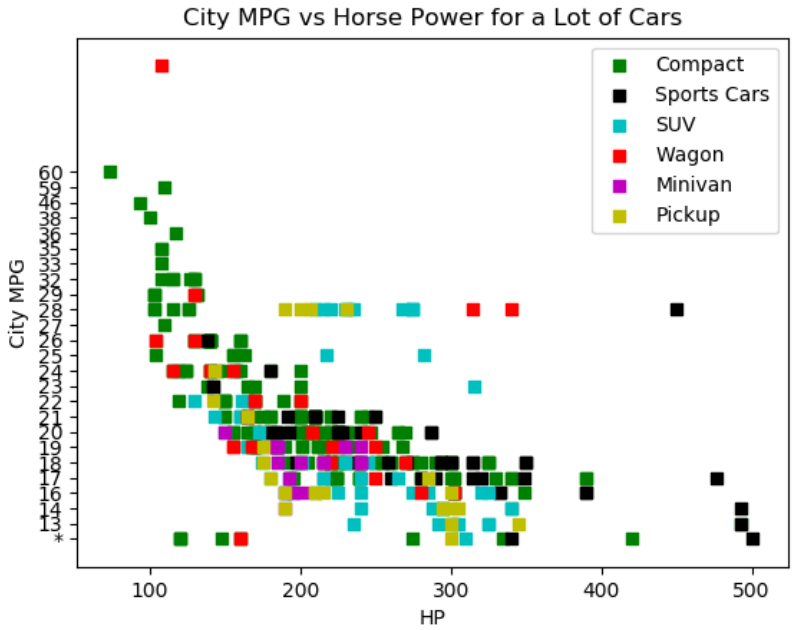


10 ./visuals/fig_1_47/oliverj

10.1 ./visuals/fig_1_47/oliverj/fig_1_47.py

```
#Hunter Oliver
#1/28/2018
import matplotlib.pyplot as plt
import pandas as pand
import sys

vehicles = ['Small/Sporty/ Compact/Large Sedan','Sports Car','SUV','Wagon', 'Minivan',
'Pickup','HP', 'City MPG', 'Weight']
df = pand.read_csv(sys.argv[1], skipinitialspace=True, usecols=vehicles)


plt.title("City MPG vs Horse Power for a Lot of Cars")
plt.xlabel("HP")
plt.ylabel("City MPG")
plt.scatter(df.loc[df['Small/Sporty/ Compact/Large Sedan'] == 1, 'HP'],
 df.loc[df['Small/Sporty/ Compact/Large Sedan'] == 1, 'City MPG'], c='g', marker='s')
plt.scatter(df.loc[df['Sports Car'] == 1, 'HP'],
 df.loc[df['Sports Car'] == 1, 'City MPG'], c='k', marker='s')
plt.scatter(df.loc[df['SUV'] == 1, 'HP'],
 df.loc[df['SUV'] == 1, 'City MPG'], c='c', marker = 's')
plt.scatter(df.loc[df['Wagon'] == 1, 'HP'],
 df.loc[df['Wagon'] == 1, 'City MPG'], c='r', marker = 's')
plt.scatter(df.loc[df['Minivan'] == 1, 'HP'],
 df.loc[df['Minivan'] == 1, 'City MPG'], c='m', marker = 's')
plt.scatter(df.loc[df['Pickup'] == 1, 'HP'],
 df.loc[df['Pickup'] == 1, 'City MPG'], c='y', marker = 's')
plt.legend(['Compact', 'Sports Cars', 'SUV', 'Wagon', 'Minivan', 'Pickup'])
plt.savefig(sys.argv[2], Transparent=True)
```

10.2 ./visuals/fig_1_47/oliverj/fig_1_47.png



11 ./visuals/fig_1_47/halvorsenca

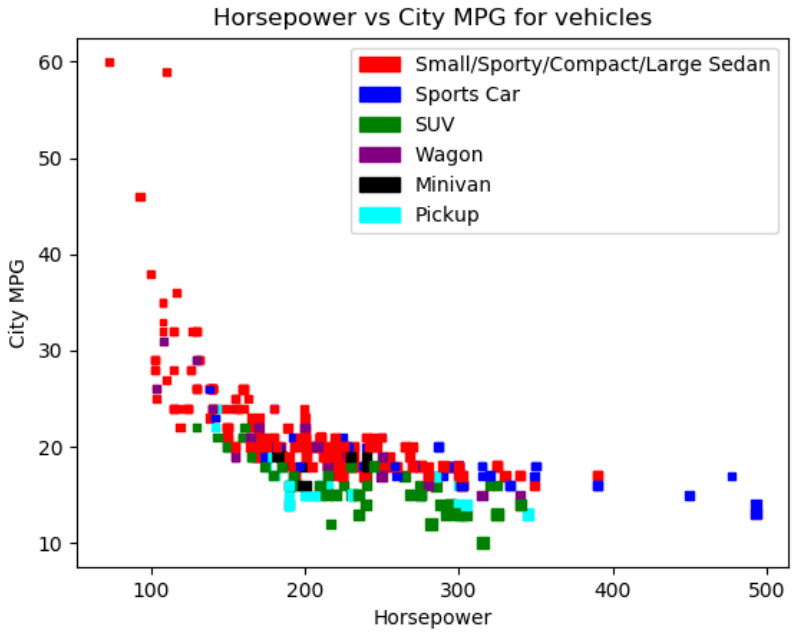
11.1 ./visuals/fig_1_47/halvorsenca/scatterCar.py

```
import csv
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
import sys

file = sys.argv[1]
data = pd.read_csv(file)

details= pd.DataFrame(data, columns=['Vehicle Name', 'Small/Sporty/ Compact/Large Sedan','Sports Car',

HP = list(details['HP'])
MPG = list(details['City MPG'])
WEIGHT = details['Weight']


temp = []
for m in MPG:
    if m == '*':
        temp.append(0)
    else:
        temp.append(float(m))

MPG = temp

temp2 = []
for w in WEIGHT:
    if w == '*':
        temp2.append(0)
    else:
        temp2.append(float(w))

WEIGHT=temp2

type={}

types = pd.DataFrame(data, columns=['Small/Sporty/ Compact/Large Sedan', 'Sports Car', 'SUV', 'Wagon',
cars = pd.DataFrame(types).to_dict()

for c in cars:
    for i in cars[c]:
        if cars[c][i] == 1:
            type[i] = c

color_dict = {'Small/Sporty/ Compact/Large Sedan': 'red', 'Sports Car': 'blue', 'SUV': 'green', 'Wagon
```

```
colors = {}
for every in type:
    colors[every] = color_dict[type[every]]

color_list = []
for i in range(len(colors)):
    color_list.append(colors[i])

x = zip(HP, MPG, color_list, WEIGHT)
x = filter(lambda item: item[0] != 0, x)
x = filter(lambda item: item[1] != 0, x)
x = filter(lambda item: item[3] != 0, x)

"""
for i in x:
    if 0 in i:
        x.remove(i)
"""
Hp, Mpg, colorr, weight = map(list, zip(*x))

"""
Mpg = np.array(MPG)
weight = np.array(WEIGHT)
Hp = np.array(HP)[Mpg != 0].tolist()
colorr = np.array(color_list)[Mpg != 0].tolist()
weight= np.array(WEIGHT)[Mpg != 0].tolist()
Mpg = Mpg[Mpg != 0].tolist()

"""


r_patch = mpatches.Patch(color='red', label= 'Small/Sporty/Compact/Large Sedan')
b_patch = mpatches.Patch(color='blue', label='Sports Car')
g_patch = mpatches.Patch(color='green', label='SUV')
p_patch = mpatches.Patch(color='purple', label='Wagon')
bl_patch = mpatches.Patch(color='black', label='Minivan')
c_patch = mpatches.Patch(color='cyan', label='Pickup')

tempw = []
for w in weight:
    tempw.append(w * .005)

weight = tempw


plt.scatter(Hp, Mpg, s=weight, marker="s", color=colorr)
plt.legend(handles=[r_patch,b_patch,g_patch,p_patch,bl_patch,c_patch])
plt.xlabel('Horsepower')
plt.ylabel('City MPG')
```

```
plt.title('Horsepower vs City MPG for vehicles')
plt.savefig(sys.argv[2])
```

11.2 ./visuals/fig_1_47/halvorsenca/fig_1_47.png



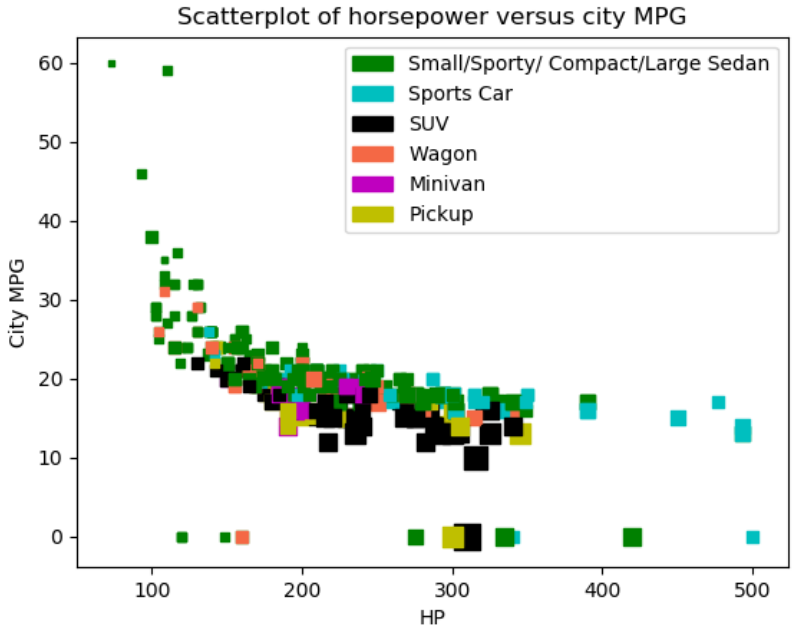
12 ./visuals/fig_1_47/laresaguilared

12.1 ./visuals/fig_1_47/laresaguilared/ScatterPlot.py

```
import csv
import matplotlib.patches as mpatches
from matplotlib import pyplot as plt
import sys

def read_file():
    table_list = []
    with open(sys.argv[1]) as csvfile:
            readCSV = csv.reader(csvfile, delimiter=',')
            for row in readCSV:
                if len(row) > 0:
                    table_list.append(row)
    return table_list



def main():
    table = read_file()
    # print(table)
    counter = 0
    for cars in table:
        if counter > 0:

            try:
                x = float(cars[13])
            except:
                x = 0
            try:
                y = float(cars[14])
            except:
                y = 0

            try:
                size = (float(cars[16])*float(cars[16])) * 0.000003
            except:
                size = 0

            if cars[1] == "1":
                plt.scatter(x, y, c="g", marker='s', s=size)
            elif cars[2] == "1":
                plt.scatter(x, y, c="c", marker='s', s=size)
            elif cars[3] == "1":
                plt.scatter(x, y, c="k", marker='s', s=size)
            elif cars[4] == "1":
                plt.scatter(x, y, c="#f46845", marker='s', s=size)
            elif cars[5] == "1":
```

```
                plt.scatter(x, y, c="m", marker='s', s=size)
            elif cars[6] == "1":
                plt.scatter(x, y, c="y", marker='s', s=size)
        else:
            counter += 1

    plt.xlabel("HP")
    plt.ylabel("City MPG")
    plt.title("Scatterplot of horsepower versus city MPG")
    small = mpatches.Patch(color='g', label='Small/Sporty/ Compact/Large Sedan')
    Sports = mpatches.Patch(color='c', label='Sports Car')
    SUV = mpatches.Patch(color='k', label='SUV')
    Wagon = mpatches.Patch(color='#f46845', label='Wagon')
    Minivan = mpatches.Patch(color='m', label='Minivan')
    Pickup = mpatches.Patch(color='y', label='Pickup')
    plt.legend(handles=[small, Sports, SUV, Wagon, Minivan, Pickup])
    plt.savefig(sys.argv[2])


if __name__ == '__main__':
    main()
```

12.2 ./visuals/fig_1_47/laresaguilared/fig_1_47.png



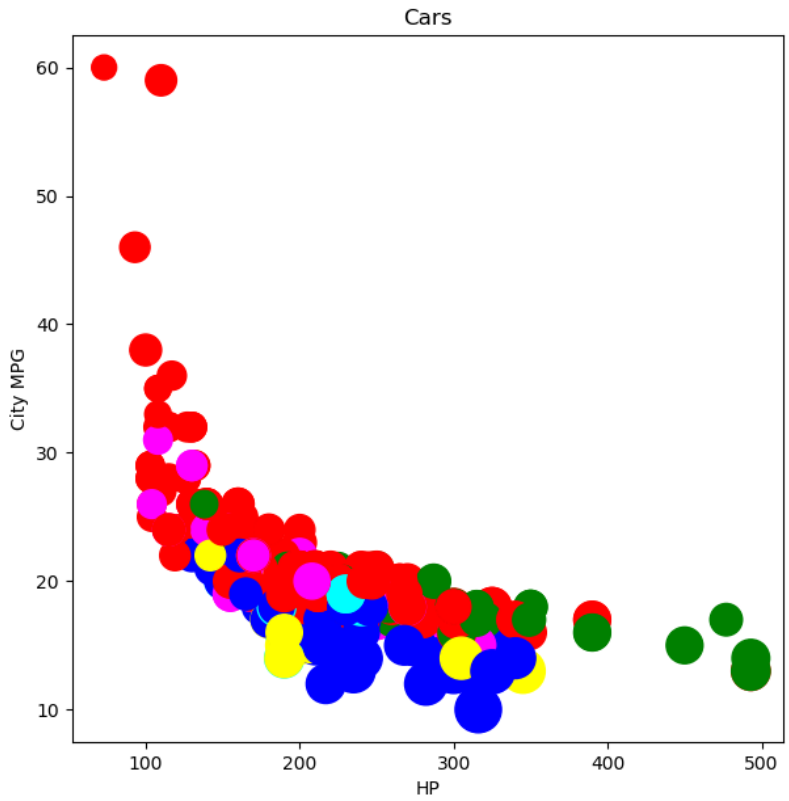
13 ./visuals/fig_1_47/beasonke

13.1 ./visuals/fig_1_47/beasonke/fig_1_47.py

```
import matplotlib.pyplot as plt
import pip
pip.main(['install','pandas'])
import pandas as pd
import seaborn
import sys

df = pd.read_csv(sys.argv[1])
df = df[["HP", "City MPG", "Weight", "Small/Sporty/ Compact/Large Sedan", "Sports Car", "SUV", "Wagon"
df = df[df["City MPG"] != '*']
df = df[df["Weight"] != '*']


def cat(row):
    if row['Small/Sporty/ Compact/Large Sedan'] == 1:
        return 'Small/Sporty/ Compact/Large Sedan'
    if row['Sports Car'] == 1:
        return 'Sports Car'
    if row['SUV'] == 1:
        return 'SUV'
    if row['Wagon'] == 1:
        return 'Wagon'
    if row['Minivan'] == 1:
        return 'Minivan'
    if row['Pickup'] == 1:
        return 'Pickup'
    else:
        return 'no cat'


categories = []
for index, row in df.iterrows():
    categories.append(cat(row))

df['category'] = categories
columns = ["Small/Sporty/ Compact/Large Sedan", "Sports Car", "SUV", "Wagon", "Minivan", "Pickup"]
df.drop(columns, inplace=True, axis=1)
# print(df)
# pd.set_option('display.max_rows', 500)
df["City MPG"] = pd.to_numeric(df["City MPG"])
df["Weight"] = pd.to_numeric(df["Weight"])

colors = {'Small/Sporty/ Compact/Large Sedan': 'red', 'Sports Car': 'green', 'SUV': 'blue',
          'Wagon': 'magenta', 'Minivan': 'cyan', 'Pickup': 'yellow'}
plot = df.plot(x="HP", y="City MPG", s=df['Weight'] / 10, c=df['category'].apply(lambda x: colors[x]),
               kind='scatter', figsize = (7,7), title='Cars')
```

```
plot.set_xlabel("HP")
plot.set_ylabel("City MPG")

plt.savefig(sys.argv[2])
```

13.2 ./visuals/fig_1_47/beasonke/fig_1_47.png



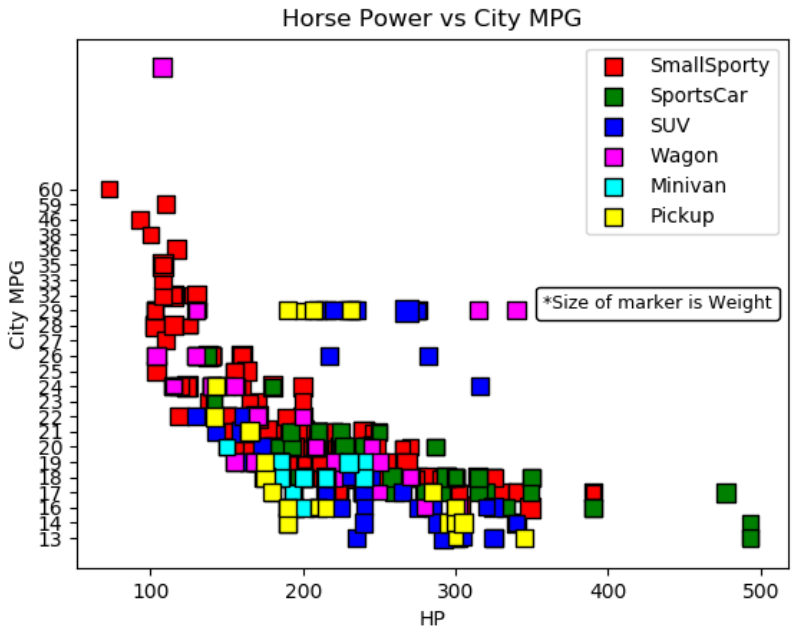
14 ./visuals/fig_1_47/beekmanpc

14.1 ./visuals/fig_1_47/beekmanpc/hw1.py

```
import sys
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid.anchored_artists import AnchoredText
import pandas as pd


def main():
    cars = pd.read_csv(sys.argv[1])
    cars.columns = ['VehicleName','SmallSporty','SportsCar','SUV','Wagon','Minivan','Pickup', 'AWD',
                    'RWD','RetailPrice','DealerCost','EngineSize(l)','Cyl','HP','CityMPG','HwyMPG','We

    cars['Type'] = 0

    # set the Type value based on the car
    cars.loc[cars.SmallSporty == 1, 'Type'] = 1
    cars.loc[cars.SportsCar == 1, 'Type'] = 2
    cars.loc[cars.SUV == 1, 'Type'] = 3
    cars.loc[cars.Wagon == 1, 'Type'] = 4
    cars.loc[cars.Minivan == 1, 'Type'] = 5
    cars.loc[cars.Pickup == 1, 'Type'] = 6

    # clean the data removing any '*' and converting str to ints
    cars = cars[cars.CityMPG != '*']
    cars = cars[cars.Weight != '*']
    cars.Weight = pd.to_numeric(cars.Weight, errors='coerce')

    # create and display the scatterplot
    num = 1
    fig, ax = plt.subplots()
    for color in ['red','green','blue','magenta','cyan','yellow']:
        X = cars['HP'].where(cars['Type'] == num).dropna()
        Y = cars['CityMPG'].where(cars['Type'] == num).dropna()
        size = cars['Weight'] / 50
        ax.scatter(X, Y, c=color, s=size, marker='s', edgecolors=(0,0,0), label=cars.columns[num])
        num = num + 1

    at = AnchoredText("*Size of marker is Weight",
                      prop=dict(size=9), frameon=True,
                      loc=7,
                      )
    at.patch.set_boxstyle("round,pad=0.,rounding_size=0.2")
    ax.add_artist(at)
    ax.legend()
    plt.xlabel("HP")
    plt.ylabel("City MPG")
    plt.title("Horse Power vs City MPG")
```

```
    plt.savefig(sys.argv[2])
    #plt.show()


if __name__ == "__main__":
    main()
```

14.2 ./visuals/fig_1_47/beekmanpc/fig_1_47.png

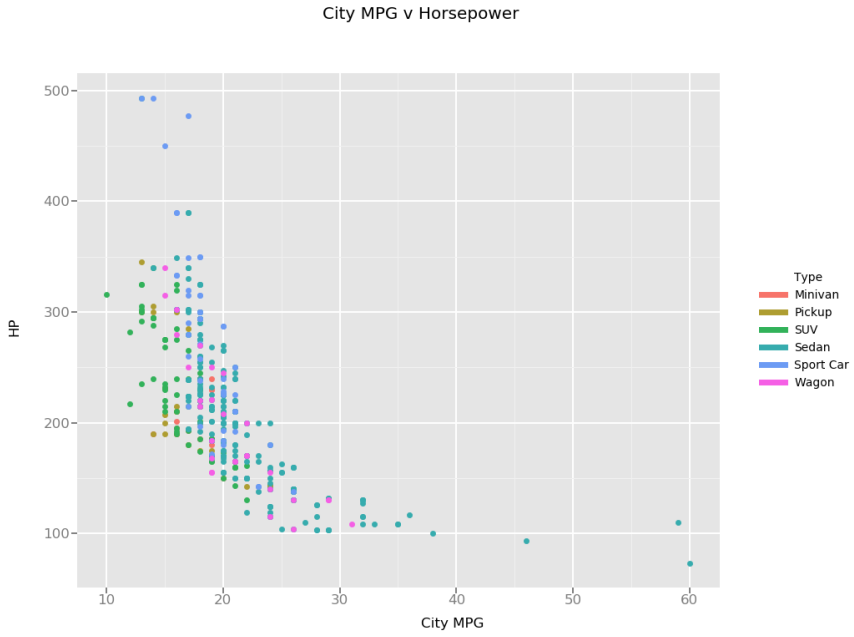


15 ./visuals/fig_1_47/emeryde

15.1 ./visuals/fig_1_47/emeryde/emery_fig_1_47.py

```
import sys

from pandas import read_csv
from ggplot import *

#df = read_csv('http://cs.appstate.edu/~rmp/cs5720/cars04.csv')
df = read_csv(sys.argv[1])

df = df.drop(df[(df['City MPG'] == '*')].index)
df = df.drop(df[(df['Weight'] == '*')].index)

def get_type():
    tmp = df['Vehicle Name'].values
    for i in range(len(tmp)):
        if(df['Small/Sporty/ Compact/Large Sedan'].values[i] == 1):
            tmp[i] = 'Sedan'
        elif df['SUV'].values[i] == 1:
            tmp[i] = 'SUV'
        elif df['Sports Car'].values[i] == 1:
            tmp[i] = 'Sport Car'
        elif df['Wagon'].values[i] == 1:
            tmp[i] = 'Wagon'
        elif df['Pickup'].values[i] == 1:
            tmp[i] = 'Pickup'
        else:
            tmp[i] = 'Minivan'
    return tmp

df['Type'] = get_type()
df['City MPG'] = df['City MPG'].astype(int)
df['Weight'] = df['Weight'].astype(int)/100

    #geom_point(aes(size = 'Weight')) +\
p=ggplot(df, aes(x='City MPG', y='HP', color = 'Type')) +\
    geom_point() +\
    xlab("City MPG") + ylab("HP") + ggtitle("City MPG v Horsepower")

p.save(sys.argv[2])
```

15.2 ./visuals/fig_1_47/emeryde/fig_1_47.png



16 ./visuals/fig_1_47/parkerat2

16.1 ./visuals/fig_1_47/parkerat2/fig_1_47.py

```
import pandas as pd
import sys
import numpy as np
import matplotlib.pyplot as plt
from collections import OrderedDict

def main():
    args = sys.argv
    carsdf = pd.read_csv(args[1])
    # carsdf = pd.read_csv("cars04.csv")

    h = []
    c = []
    cy = set()
    for cmpg, hp, cyl, in zip(carsdf['City MPG'],
                carsdf['HP'], carsdf['Cyl']):
        if hp is not '*' and cmpg is not '*' and cyl is not -1:
            h.append(int(hp))
            c.append(int(cmpg))
            cy.add(cyl)
            hp=float(hp)
            cmpg=float(cmpg)
            cyl=float(cyl)
            if cyl == 3:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='y', edgecolors='gray', labe
            if cyl == 4:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='g', edgecolors='gray', labe
            if cyl == 5:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='m', edgecolors='gray', labe
            if cyl == 6:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='k', edgecolors='gray', labe
            if cyl == 8:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='c', edgecolors='gray', labe
            if cyl == 10:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='gray', edgecolors='gray', l
            if cyl == 12:
                plt.scatter(x=hp, y=cmpg, marker='s', s=(cyl * 15), color='r', edgecolors='gray', labe

    plt.xticks(np.arange(min(h), max(h) + 10, 42.7))
    plt.yticks(np.arange(min(c), max(c)+1, 5))
    plt.xlim(xmin=min(h)-5, xmax=max(h)+13)
    plt.ylim(ymin=min(c)-2, ymax=max(c)+2)
    plt.xlabel("HP")
    plt.ylabel("City MPG")
    plt.title("2004 Vehicle Comparison of Horsepower, City MPG, and # of Cylinders")
    handles, labels = plt.gca().get_legend_handles_labels()
```

```
    by_label = OrderedDict(zip(labels, handles))
    plt.legend(by_label.values(), by_label.keys(), title="# Cylinders")
    # plt.show()
    plt.savefig(sys.argv[2])
    plt.clf()


if __name__ == '__main__':
    main()
```

16.2 ./visuals/fig_1_47/parkerat2/fig_1_47.png