Documentation for recreating "Two-dimensional Pareto frontier forecasting for technology planning and roadmapping"

Arshia Feizmohammady

August 8, 2025

Introduction

This document provides thorough documentation for recreating the results of the article written by Ksenia Smirnova, Alessandro Golkar, and Rob Vingerhoeds. It includes an overview of the methodology, code snippets, and explanations to ensure reproducibility and understanding.

Data Gathering

This section provides an overview of the data gathering and cleaning process. It includes detailed steps on how data was collected, and what sources were used.

The authors of the article utilized the following website to gather their data:

Car Specs Database — cars-data.com [WWW Document], n.d., URL: `https://www.cars-data.com/` (accessed 2.9.20), 2020,

This website hosts a comprehensive dataset of all car models. However, it does not provide an option to directly download the data, necessitating web scraping to obtain the required information.

Due to technical challenges associated with web scraping, an alternative approach was taken. A publicly available dataset from Kaggle was used. This dataset can be found at:

Car Specification Dataset 1945-2020, URL: `https://www.kaggle.com/datasets/jahaidulislam/car-specification-dataset-1945-2020`.

It is important to acknowledge the differences between these two datasets, as they can impact the recreated figures. The dataset used by the original authors consists of 3,289 datapoints and encompasses car models from 32 car brands representing all major automotive companies. In contrast, the dataset obtained from Kaggle contains over 50,000 datapoints, providing a broader range of information.

Data Visualization

While the article does not specify which two car brands they have chosen, we strongly believe them to be BMW and Mercedes-Benz. In this section, we will visualize the development of both companies over time.

```python
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

# Load the data
file_path = 'Car Dataset 1945-2020.csv'
car_data = pd.read_csv(file_path, low_memory=False)

# Define a custom colormap
cmap = LinearSegmentedColormap.from_list('custom_cmap', ['darkblue', 'lightblue', 'green', 'yellow', 'orange', 'red'])

# Segment the data by manufacturer
mercedes_data = car_data[car_data['Make'] == 'Mercedes-Benz'].copy()
bmw_data = car_data[car_data['Make'] == 'BMW'].copy()

# Function to preprocess data and fill NaN values with median
def preprocess_and_fill_nan(data):
    # Filter data for the years 1975 to 2015
    data = data[(data['Year_from'] >= 1975) & (data['Year_from'] <= 2015)].copy()
    # Fill NaN values with the median
    data.loc[:, 'engine_hp'] = data['engine_hp'].fillna(data['engine_hp'].median())
    data.loc[:, 'mixed_fuel_consumption_per_100_km_l'] = data['mixed_fuel_consumption_per_100_km_l'].fillna(data['mixed_fuel_consumption_per_100_km_l'].median())
    return data

# Preprocess and fill NaN values for Mercedes-Benz and BMW data
mercedes_data = preprocess_and_fill_nan(mercedes_data)
bmw_data = preprocess_and_fill_nan(bmw_data)

# Print the number of data points for BMW and Mercedes-Benz
print(f"Number of BMW data points: {len(bmw_data)}")
print(f"Number of Mercedes-Benz data points: {len(mercedes_data)}")

# Function to create scatter plots with actual release years
def plot_car_data(data, manufacturer):
    plt.figure(figsize=(12, 6))
    scatter = plt.scatter(data['engine_hp'], data['mixed_fuel_consumption_per_100_km_l'], c=data['Year_from'],
```

```
        cmap=cmap)
    cbar = plt.colorbar(scatter, label='Year')
    plt.title(f'Development of Car Models by {manufacturer}')
    plt.xlabel('Max Engine Power (HP)')
    plt.ylabel('Average Fuel Consumption (l/100 km)')
    plt.grid(True)
    plt.show()

# Function to create combined scatter plots for Mercedes-Benz and BMW
def plot_combined_car_data(data_a, data_b):
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12), sharex=True, sharey=True)

    scatter1 = ax1.scatter(data_a['engine_hp'], data_a['mixed_fuel_consumption_per_100_km_l'], c=data_a['Year_from'], cmap=cmap)
    ax1.set_title('Development of Car Models by Mercedes-Benz')
    ax1.set_ylabel('Average Fuel Consumption (l/100 km)')
    ax1.grid(True)

    scatter2 = ax2.scatter(data_b['engine_hp'], data_b['mixed_fuel_consumption_per_100_km_l'], c=data_b['Year_from'], cmap=cmap)
    ax2.set_title('Development of Car Models by BMW')
    ax2.set_xlabel('Max Engine Power (HP)')
    ax2.set_ylabel('Average Fuel Consumption (l/100 km)')
    ax2.grid(True)

    # Create a color bar on the right side
    cbar = fig.colorbar(scatter1, ax=[ax1, ax2], orientation='vertical', pad=0.05, shrink=0.8)
    cbar.set_label('Year')

    plt.subplots_adjust(right=0.75)  # Adjust right side to make space for color bar
    plt.show()

# Create scatter plots for Mercedes-Benz and BMW with the correct color scale
plot_car_data(mercedes_data, 'Mercedes-Benz')
plot_car_data(bmw_data, 'BMW')
plot_combined_car_data(mercedes_data, bmw_data)
```

Listing 1: Data Visualization Code

First, we load the data and define a custom colormap to visualize the development of car models by year.

```
# Load the data
file_path = 'Car Dataset 1945-2020.csv'
car_data = pd.read_csv(file_path, low_memory=False)
```

We load the dataset from a CSV file using `pandas`.

```
# Define a custom colormap
cmap = LinearSegmentedColormap.from_list('custom_cmap', ['darkblue', 'lightblue', 'green', 'yellow', 'orange', 'red'])
```

A custom colormap is defined to visualize the data according to the year.

```python
# Segment the data by manufacturer
mercedes_data = car_data[car_data['Make'] == 'Mercedes-Benz'].copy()
bmw_data = car_data[car_data['Make'] == 'BMW'].copy()
```

We segment the data by manufacturer, focusing on Mercedes-Benz and BMW.

```python
# Function to preprocess data and fill NaN values with median
def preprocess_and_fill_nan(data):
    # Filter data for the years 1975 to 2015
    data = data[(data['Year_from'] >= 1975) & (data['Year_from'] <= 2015)].copy()
    # Fill NaN values with the median
    data.loc[:, 'engine_hp'] = data['engine_hp'].fillna(data['engine_hp'].median())
    data.loc[:, 'mixed_fuel_consumption_per_100_km_l'] = data['mixed_fuel_consumption_per_100_km_l'].fillna(data['mixed_fuel_consumption_per_100_km_l'].median())
    return data

# Preprocess and fill NaN values for Mercedes-Benz and BMW data
mercedes_data = preprocess_and_fill_nan(mercedes_data)
bmw_data = preprocess_and_fill_nan(bmw_data)
```

A function `preprocess_and_fill_nan` is defined to preprocess the data by filtering for the years 1975 to 2015 and filling NaN values with the median. This function is applied to both Mercedes-Benz and BMW data.

```python
# Print the number of data points for BMW and Mercedes-Benz
print(f"Number of BMW data points: {len(bmw_data)}")
print(f"Number of Mercedes-Benz data points: {len(mercedes_data)}")
```

We print the number of data points for BMW and Mercedes-Benz to verify the data.

```python
# Function to create scatter plots with actual release years
def plot_car_data(data, manufacturer):
    plt.figure(figsize=(12, 6))
    scatter = plt.scatter(data['engine_hp'], data['mixed_fuel_consumption_per_100_km_l'], c=data['Year_from'], cmap=cmap)
    cbar = plt.colorbar(scatter, label='Year')
    plt.title(f'Development of Car Models by {manufacturer}')
    plt.xlabel('Max Engine Power (HP)')
    plt.ylabel('Average Fuel Consumption (l/100 km)')
    plt.grid(True)
    plt.show()

# Function to create combined scatter plots for Mercedes-Benz and BMW
def plot_combined_car_data(data_a, data_b):
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12), sharex=True, sharey=True)

```

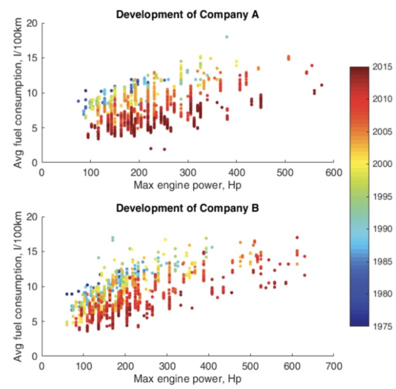
```
    scatter1 = ax1.scatter(data_a['engine_hp'], data_a['mixed_fuel_consumption_per_100_km_l'], c=data_a['Year_from'], cmap=cmap)
    ax1.set_title('Development of Car Models by Mercedes-Benz')
    ax1.set_ylabel('Average Fuel Consumption (l/100 km)')
    ax1.grid(True)

    scatter2 = ax2.scatter(data_b['engine_hp'], data_b['mixed_fuel_consumption_per_100_km_l'], c=data_b['Year_from'], cmap=cmap)
    ax2.set_title('Development of Car Models by BMW')
    ax2.set_xlabel('Max Engine Power (HP)')
    ax2.set_ylabel('Average Fuel Consumption (l/100 km)')
    ax2.grid(True)

    # Create a color bar on the right side
    cbar = fig.colorbar(scatter1, ax=[ax1, ax2], orientation='vertical', pad=0.05, shrink=0.8)
    cbar.set_label('Year')

    plt.subplots_adjust(right=0.75)  # Adjust right side to make space for color bar
    plt.show()

# Create scatter plots for Mercedes-Benz and BMW with the correct color scale
plot_car_data(mercedes_data, 'Mercedes-Benz')
plot_car_data(bmw_data, 'BMW')
plot_combined_car_data(mercedes_data, bmw_data)
```

Functions are defined to create scatter plots for each manufacturer and combined scatter plots for both. The scatter plots visualize the development of car models by engine power and fuel consumption over the years.

Comparison of Results - Data Visualization

In this subsection, we compare our visualization with the corresponding plot from the original article. This comparison will help in understanding the discrepancies and similarities between the recreated results and the article's results.

(a) Obtained Plot



(b) Article Plot

Figure 1: Comparison of Data Visualization

By comparing the images, we see that while they are not identical (due to them having different data source), they are quite similar and we can continue to recreate the results of this article.

Best-Responses

In this section, we will calculate and visualize the best responses of BMW and Mercedes-Benz regarding total max power, average fuel consumption, and acceleration. Additionally, we will also analyze the optional metrics such as empty mass, max loading capacity, and displacement.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import pearsonr

# Load the original data
file_path = 'Car Dataset 1945-2020.csv'
car_data = pd.read_csv(file_path, low_memory=False)

# Filter data for the years 1970 to 2020
filtered_data = car_data[(car_data['Year_from'] >= 1970) & (
    car_data['Year_from'] <= 2017)]

# Segment the data by manufacturer
mercedes_data = filtered_data[filtered_data['Make'] == 'Mercedes-
    Benz']
bmw_data = filtered_data[filtered_data['Make'] == 'BMW']

# Function to remove outliers below the 0th percentile and above
    the 95th percentile
def remove_outliers_MaxPower(data, column):
    percentile_5 = data[column].quantile(0.08)
```

```python
    percentile_95 = data[column].quantile(0.97)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

def remove_outliers_Fuel(data, column):
    percentile_5 = data[column].quantile(0.0)
    percentile_95 = data[column].quantile(0.84)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

def remove_outliers_Acceleration(data, column):
    percentile_5 = data[column].quantile(0.0)
    percentile_95 = data[column].quantile(0.97)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

def remove_outliers_Weight(data, column):
    percentile_5 = data[column].quantile(0.13)
    percentile_95 = data[column].quantile(0.96)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

def remove_outliers_LoadingCapacity(data, column):
    percentile_5 = data[column].quantile(0.17)
    percentile_95 = data[column].quantile(0.94)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

def remove_outliers_Displacement(data, column):
    percentile_5 = data[column].quantile(0.0)
    percentile_95 = data[column].quantile(0.97)
    return data[(data[column] >= percentile_5) & (data[column] <=
        percentile_95)]

## TOTAL MAX POWER BEST RESPONSE // BEGIN

# Function to calculate the best response (max engine power) by
    year for a manufacturer
def calculate_best_response_power(data, manufacturer):
    # Group by year and calculate the max engine power
    best_response = data.groupby('Year_from')['engine_hp'].max().
        reset_index()
    best_response.columns = ['Year_from', f'engine_hp_{manufacturer
        .lower()}']
    return best_response

# Calculate the best response for both manufacturers
best_response_power_mercedes = calculate_best_response_power(
    mercedes_data, 'Mercedes')
best_response_power_bmw = calculate_best_response_power(bmw_data, '
    BMW')

# Merge the two dataframes on Year_from to align the years
engine_hp_merged = pd.merge(best_response_power_mercedes,
    best_response_power_bmw, on='Year_from')

# Plot the trends of best response for both manufacturers
```

```python
plt.figure(figsize=(12, 6))
plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['engine_hp_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['engine_hp_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Total Max Power: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Max Engine Power (HP)')
plt.ylim(0, max(engine_hp_merged['engine_hp_mercedes'].max(), engine_hp_merged['engine_hp_bmw'].max()) + 50)  # Ensure consistent spacing
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged, 'engine_hp_mercedes')
engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged_clean, 'engine_hp_bmw')

# Calculate correlation and p-value for engine power
engine_hp_corr, engine_hp_pval = pearsonr(engine_hp_merged_clean['engine_hp_mercedes'], engine_hp_merged_clean['engine_hp_bmw'])
print(f"Correlation for Engine Power: {engine_hp_corr}, p-value: {engine_hp_pval}")

## // END

## Average Fuel Consumption // Begin

# Calculate the average fuel consumption by year for each manufacturer
avg_consumption_mercedes = mercedes_data.groupby('Year_from')['mixed_fuel_consumption_per_100_km_l'].min().reset_index()
avg_consumption_bmw = bmw_data.groupby('Year_from')['mixed_fuel_consumption_per_100_km_l'].min().reset_index()

# Merge the two dataframes on Year_from to align the years
fuel_consumption_merged = pd.merge(avg_consumption_mercedes, avg_consumption_bmw, on='Year_from', suffixes=('_mercedes', '_bmw'))

# Plot the trends of average fuel consumption for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(fuel_consumption_merged['Year_from'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(fuel_consumption_merged['Year_from'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_bmw'], marker='o', linestyle='-', color='b', label='BMW')
```

```
plt.title('Average Fuel Consumption: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Average Fuel Consumption (l/100 km)')
plt.yticks(np.arange(0, 21, 2))  # Set y-axis ticks to increments of 2 from 0 to 20
plt.ylim(0, 20)  # Set y-axis limits from 0 to 20
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
fuel_consumption_merged = remove_outliers_Fuel(fuel_consumption_merged, 'mixed_fuel_consumption_per_100_km_l_mercedes')
fuel_consumption_merged = remove_outliers_Fuel(fuel_consumption_merged, 'mixed_fuel_consumption_per_100_km_l_bmw')

# Calculate correlation and p-value for fuel consumption
fuel_consumption_corr, fuel_consumption_pval = pearsonr(fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_mercedes'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_bmw'])
print(f"Correlation for Fuel Consumption: {fuel_consumption_corr}, p-value: {fuel_consumption_pval}")

## // END

## Acceleration // Begin

# Function to calculate the best response (min acceleration time) by year for a manufacturer
def calculate_best_response_acceleration(data, manufacturer):
    # Group by year and calculate the min acceleration time
    best_response = data.groupby('Year_from')['acceleration_0_100_km/h_s'].min().reset_index()
    best_response.columns = ['Year_from', f'acceleration_0_100_km/h_s_{manufacturer.lower()}']
    return best_response

# Calculate the best response for both manufacturers
best_response_acceleration_mercedes = calculate_best_response_acceleration(mercedes_data, 'Mercedes')
best_response_acceleration_bmw = calculate_best_response_acceleration(bmw_data, 'BMW')

# Merge the two dataframes on Year_from to align the years
acceleration_merged = pd.merge(best_response_acceleration_mercedes, best_response_acceleration_bmw, on='Year_from')

# Plot the trends of best response acceleration for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(acceleration_merged['Year_from'], acceleration_merged['acceleration_0_100_km/h_s_mercedes'], marker='o', linestyle='-'
```

```
    , color='r', label='Mercedes-Benz')
plt.plot(acceleration_merged['Year_from'], acceleration_merged['
    acceleration_0_100_km/h_s_bmw'], marker='o', linestyle='-',
    color='b', label='BMW')
plt.title('Acceleration: Best Response Trends of Mercedes-Benz and
    BMW')
plt.xlabel('Release Year')
plt.ylabel('Acceleration (0-100 km/h)')
plt.yticks(np.arange(0, 20, 5))  # Set y-axis ticks to increments
    of 5 from 0 to 20
plt.ylim(0, 20)
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged, 'acceleration_0_100_km/h_s_mercedes')
acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged_clean, 'acceleration_0_100_km/h_s_bmw')

# Calculate correlation and p-value for acceleration
acceleration_corr, acceleration_pval = pearsonr(
    acceleration_merged_clean['acceleration_0_100_km/h_s_mercedes'
    ], acceleration_merged_clean['acceleration_0_100_km/h_s_bmw'])
print(f"Correlation for Acceleration: {acceleration_corr}, p-value:
     {acceleration_pval}")

## // END
```

Listing 2: Best-Responses Code

First, we load and preprocess the data, then filter it for the years 1970 to 2017. We segment the data by manufacturer, focusing on Mercedes-Benz and BMW. Functions are defined to remove outliers for different parameters.

Total Max Power Best Response

```
# Function to calculate the best response (max engine power) by
    year for a manufacturer
def calculate_best_response_power(data, manufacturer):
    # Group by year and calculate the max engine power
    best_response = data.groupby('Year_from')['engine_hp'].max().
        reset_index()
    best_response.columns = ['Year_from', f'engine_hp_{manufacturer
        .lower()}']
    return best_response

# Calculate the best response for both manufacturers
best_response_power_mercedes = calculate_best_response_power(
    mercedes_data, 'Mercedes')
best_response_power_bmw = calculate_best_response_power(bmw_data, '
    BMW')

# Merge the two dataframes on Year_from to align the years
engine_hp_merged = pd.merge(best_response_power_mercedes,
    best_response_power_bmw, on='Year_from')

# Plot the trends of best response for both manufacturers
```

```
plt.figure(figsize=(12, 6))
plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['engine_hp_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['engine_hp_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Total Max Power: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Max Engine Power (HP)')
plt.ylim(0, max(engine_hp_merged['engine_hp_mercedes'].max(), engine_hp_merged['engine_hp_bmw'].max()) + 50)  # Ensure consistent spacing
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged, 'engine_hp_mercedes')
engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged_clean, 'engine_hp_bmw')

# Calculate correlation and p-value for engine power
engine_hp_corr, engine_hp_pval = pearsonr(engine_hp_merged_clean['engine_hp_mercedes'], engine_hp_merged_clean['engine_hp_bmw'])
print(f"Correlation for Engine Power: {engine_hp_corr}, p-value: {engine_hp_pval}")
```

We calculate the best response for total max power by year for both manufacturers, plot the trends, and calculate the correlation.

Average Fuel Consumption

```
# Calculate the average fuel consumption by year for each manufacturer
avg_consumption_mercedes = mercedes_data.groupby('Year_from')['mixed_fuel_consumption_per_100_km_l'].min().reset_index()
avg_consumption_bmw = bmw_data.groupby('Year_from')['mixed_fuel_consumption_per_100_km_l'].min().reset_index()

# Merge the two dataframes on Year_from to align the years
fuel_consumption_merged = pd.merge(avg_consumption_mercedes, avg_consumption_bmw, on='Year_from', suffixes=('_mercedes', '_bmw'))

# Plot the trends of average fuel consumption for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(fuel_consumption_merged['Year_from'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(fuel_consumption_merged['Year_from'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_bmw'], marker='o', linestyle='-', color='b', label='BMW')
```

```python
plt.title('Average Fuel Consumption: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Average Fuel Consumption (l/100 km)')
plt.yticks(np.arange(0, 21, 2))  # Set y-axis ticks to increments of 2 from 0 to 20
plt.ylim(0, 20)  # Set y-axis limits from 0 to 20
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
fuel_consumption_merged = remove_outliers_Fuel(fuel_consumption_merged, 'mixed_fuel_consumption_per_100_km_l_mercedes')
fuel_consumption_merged = remove_outliers_Fuel(fuel_consumption_merged, 'mixed_fuel_consumption_per_100_km_l_bmw')

# Calculate correlation and p-value for fuel consumption
fuel_consumption_corr, fuel_consumption_pval = pearsonr(fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_mercedes'], fuel_consumption_merged['mixed_fuel_consumption_per_100_km_l_bmw'])
print(f"Correlation for Fuel Consumption: {fuel_consumption_corr}, p-value: {fuel_consumption_pval}")
```

We calculate the average fuel consumption by year for both manufacturers, plot the trends, and calculate the correlation.

Acceleration

```python
# Function to calculate the best response (min acceleration time) by year for a manufacturer
def calculate_best_response_acceleration(data, manufacturer):
    # Group by year and calculate the min acceleration time
    best_response = data.groupby('Year_from')['acceleration_0_100_km/h_s'].min().reset_index()
    best_response.columns = ['Year_from', f'acceleration_0_100_km/h_s_{manufacturer.lower()}']
    return best_response

# Calculate the best response for both manufacturers
best_response_acceleration_mercedes = calculate_best_response_acceleration(mercedes_data, 'Mercedes')
best_response_acceleration_bmw = calculate_best_response_acceleration(bmw_data, 'BMW')

# Merge the two dataframes on Year_from to align the years
acceleration_merged = pd.merge(best_response_acceleration_mercedes, best_response_acceleration_bmw, on='Year_from')

# Plot the trends of best response acceleration for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(acceleration_merged['Year_from'], acceleration_merged['acceleration_0_100_km/h_s_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
```

```python
plt.plot(acceleration_merged['Year_from'], acceleration_merged['acceleration_0_100_km/h_s_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Acceleration: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Acceleration (0-100 km/h)')
plt.yticks(np.arange(0, 20, 5))  # Set y-axis ticks to increments of 5 from 0 to 20
plt.ylim(0, 20)
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
acceleration_merged_clean = remove_outliers_Acceleration(acceleration_merged, 'acceleration_0_100_km/h_s_mercedes')
acceleration_merged_clean = remove_outliers_Acceleration(acceleration_merged_clean, 'acceleration_0_100_km/h_s_bmw')

# Calculate correlation and p-value for acceleration
acceleration_corr, acceleration_pval = pearsonr(acceleration_merged_clean['acceleration_0_100_km/h_s_mercedes'], acceleration_merged_clean['acceleration_0_100_km/h_s_bmw'])
print(f"Correlation for Acceleration: {acceleration_corr}, p-value: {acceleration_pval}")
```

We calculate the best response for acceleration by year for both manufacturers, plot the trends, and calculate the correlation.

The visualization of the remaining metrics, such as empty mass, max loading capacity, and displacement, is optional.

Full Weight (using curb weight)

```python
# Function to calculate the best response (max weight) by year for a manufacturer
def calculate_best_response_weight(data, column, manufacturer):
    # Ensure the column is numeric and handle NaNs
    data.loc[:, column] = pd.to_numeric(data[column], errors='coerce')
    data = data.dropna(subset=[column])

    # Group by year and calculate the max weight
    best_response = data.groupby('Year_from')[column].mean().reset_index()
    best_response.columns = ['Year_from', f'{column}_{manufacturer.lower()}']
    return best_response

# Calculate the best response for both manufacturers using curb weight
best_response_weight_mercedes = calculate_best_response_weight(mercedes_data, 'curb_weight_kg', 'Mercedes')
best_response_weight_bmw = calculate_best_response_weight(bmw_data, 'curb_weight_kg', 'BMW')

# Merge the two dataframes on Year_from to align the years
weight_merged = pd.merge(best_response_weight_mercedes,
```

```
    best_response_weight_bmw, on='Year_from')

# Plot the trends of best response for weight for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(weight_merged['Year_from'], weight_merged['curb_weight_kg_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(weight_merged['Year_from'], weight_merged['curb_weight_kg_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Empty Mass: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Max Curb Weight (kg)')
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
weight_merged_clean = remove_outliers_Weight(weight_merged, 'curb_weight_kg_mercedes')
weight_merged_clean = remove_outliers_Weight(weight_merged_clean, 'curb_weight_kg_bmw')

# Calculate correlation and p-value for weight
weight_corr, weight_pval = pearsonr(weight_merged_clean['curb_weight_kg_mercedes'], weight_merged_clean['curb_weight_kg_bmw'])
print(f"Correlation for Curb Weight: {weight_corr}, p-value: {weight_pval}")
```

Max Loading Capacity

```
# Function to calculate the best response (max full weight) by year for a manufacturer
def calculate_best_response_full_weight(data, column, manufacturer):
    # Ensure the column is numeric and handle NaNs
    data.loc[:, column] = pd.to_numeric(data[column], errors='coerce')
    data = data.dropna(subset=[column])

    # Group by year and calculate the max full weight
    best_response = data.groupby('Year_from')[column].max().reset_index()
    best_response.columns = ['Year_from', f'{column}_{manufacturer.lower()}']
    return best_response

# Calculate the best response for both manufacturers using full weight
best_response_full_weight_mercedes = calculate_best_response_full_weight(mercedes_data, 'full_weight_kg', 'Mercedes')
best_response_full_weight_bmw = calculate_best_response_full_weight(bmw_data, 'full_weight_kg', 'BMW')

```

```python
# Merge the two dataframes on Year_from to align the years
full_weight_merged = pd.merge(best_response_full_weight_mercedes, best_response_full_weight_bmw, on='Year_from')

# Plot the trends of best response for full weight for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(full_weight_merged['Year_from'], full_weight_merged['full_weight_kg_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(full_weight_merged['Year_from'], full_weight_merged['full_weight_kg_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Loading Capacity: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Max Full Weight (kg)')
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
full_weight_merged_clean = remove_outliers_LoadingCapacity(full_weight_merged, 'full_weight_kg_mercedes')
full_weight_merged_clean = remove_outliers_LoadingCapacity(full_weight_merged_clean, 'full_weight_kg_bmw')

# Calculate correlation and p-value for full weight
full_weight_corr, full_weight_pval = pearsonr(full_weight_merged_clean['full_weight_kg_mercedes'], full_weight_merged_clean['full_weight_kg_bmw'])
print(f"Correlation for Full Weight: {full_weight_corr}, p-value: {full_weight_pval}")
```

Displacement

```python
# Ensure capacity_cm3 column is numeric, coerce errors to NaN
mercedes_data.loc[:, 'capacity_cm3'] = pd.to_numeric(mercedes_data['capacity_cm3'], errors='coerce')
bmw_data.loc[:, 'capacity_cm3'] = pd.to_numeric(bmw_data['capacity_cm3'], errors='coerce')

# Create copies of the dataframes to avoid SettingWithCopyWarning
mercedes_data_clean = mercedes_data.dropna(subset=['capacity_cm3']).copy()
bmw_data_clean = bmw_data.dropna(subset=['capacity_cm3']).copy()

# Calculate the best response (max displacement) by year for each manufacturer
best_response_displacement_mercedes = mercedes_data_clean.groupby('Year_from')['capacity_cm3'].max().reset_index()
best_response_displacement_bmw = bmw_data_clean.groupby('Year_from')['capacity_cm3'].max().reset_index()

# Merge the two dataframes on Year_from to align the years
displacement_merged = pd.merge(best_response_displacement_mercedes, best_response_displacement_bmw, on='Year_from', suffixes=('_mercedes', '_bmw'))
```

```python

# Plot the trends of best response for displacement for both manufacturers
plt.figure(figsize=(12, 6))
plt.plot(displacement_merged['Year_from'], displacement_merged['capacity_cm3_mercedes'], marker='o', linestyle='-', color='r', label='Mercedes-Benz')
plt.plot(displacement_merged['Year_from'], displacement_merged['capacity_cm3_bmw'], marker='o', linestyle='-', color='b', label='BMW')
plt.title('Displacement: Best Response Trends of Mercedes-Benz and BMW')
plt.xlabel('Release Year')
plt.ylabel('Max Displacement (cm )')
plt.legend()
plt.grid(True)
plt.show()

# Remove outliers for correlation calculation
displacement_merged = remove_outliers_Displacement(displacement_merged, 'capacity_cm3_mercedes')
displacement_merged = remove_outliers_Displacement(displacement_merged, 'capacity_cm3_bmw')

# Calculate correlation and p-value for displacement
displacement_corr, displacement_pval = pearsonr(displacement_merged['capacity_cm3_mercedes'], displacement_merged['capacity_cm3_bmw'])
print(f"Correlation for Displacement: {displacement_corr}, p-value: {displacement_pval}")
```

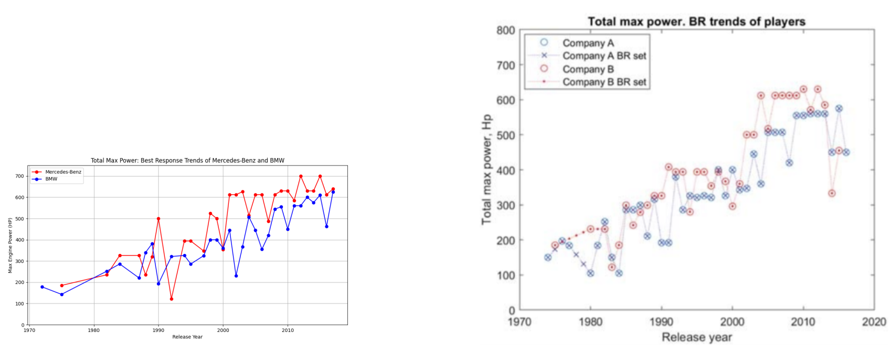
Finally, we summarize the correlations for all metrics:

```python
# Dynamic data from your analysis
correlation_data = {
    'FOM': [
        'Total Max Engine Power',
        'Average Fuel Consumption',
        'Acceleration',
        'Empty Mass',
        'Max Loading Capacity',
        'Displacement'
    ],
    'Correlation': [
        engine_hp_corr,
        fuel_consumption_corr,
        acceleration_corr,
        weight_corr,
        full_weight_corr,
        displacement_corr
    ],
    'p-value': [
        engine_hp_pval,
        fuel_consumption_pval,
        acceleration_pval,
        weight_pval,
        full_weight_pval,
        displacement_pval
```

```
    ]
}

# Round correlation values and format p-values
correlation_data['Correlation'] = [round(corr, 3) for corr in
    correlation_data['Correlation']]
correlation_data['p-value'] = ['{:.2e}'.format(pval) for pval in
    correlation_data['p-value']]

# Create DataFrame
df = pd.DataFrame(correlation_data)

# Sort DataFrame by Correlation in descending order
df = df.sort_values(by='Correlation', ascending=False).reset_index(
    drop=True)

# Display the table
print("Correlation of Players FOM Best-Response Sets")
print(df)
```

Comparison of Results - Best Response Trends

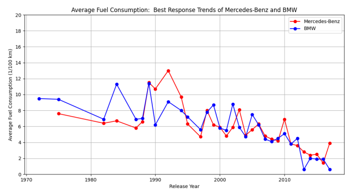
In this subsection, we compare our best response trends with the corresponding plots from the original article. This comparison will help in understanding the discrepancies and similarities between the recreated results and the article's results.



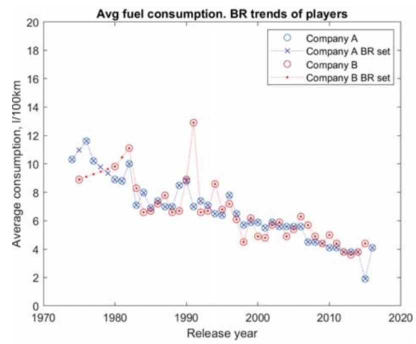
(a) Obtained Plot 1

(b) Article Plot 1

Figure 2: Comparison of Best Response Trends - Part 1

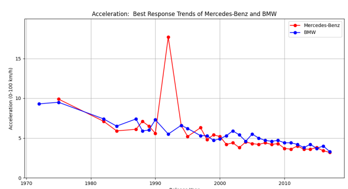


(a) Obtained Plot 2

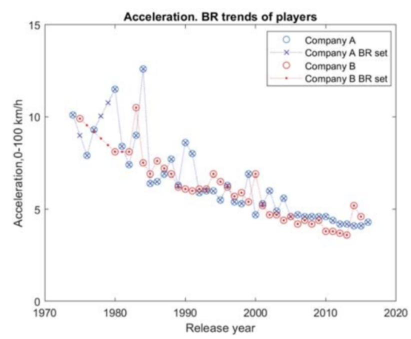


(b) Article Plot 2

Figure 3: Comparison of Best Response Trends - Part 2



(a) Obtained Plot 3



(b) Article Plot 3

Figure 4: Comparison of Best Response Trends - Part 3

	FOM	Correlation	p-value
0	Average Fuel Consumption	0.856	3.85e-07
1	Acceleration	0.795	2.51e-07
2	Total Max Engine Power	0.792	2.43e-06
3	Empty Mass	0.718	3.62e-04
4	Max Loading Capacity	0.704	1.11e-03
5	Displacement	0.511	4.65e-03

(a) Obtained Plot 4

TABLE I. Correlation of players FOM best-response sets

FOM	Corr	p-value
Average fuel consumption	0.895	2.97e-15
Total max engine power	0.851	1.73e-12
Acceleration	0.852	1.6e-12
Empty mass	0.785	1.24e-09
Max loading capacity	0.701	3.31e-07
Displacement	0.526	4.15e-04
Price	0.815	8.41e-11

(b) Article Plot 4

Figure 5: Comparison of Best Response Trends - Part 4

By Comparison of the figures, we see that the overall general trends of the figures are the same. Additionally, the correlation of players (last figures) are also quite similar.

Conclusion

This document provides a comprehensive analysis of the best responses of two car manufacturers, BMW and Mercedes-Benz, regarding total max power, average fuel consumption, and acceleration. We recreated and extended the methodology presented in the original article, while addressing potential differences and discrepancies.

Summary

- **Data Visualization:** We visualized the development of car models by BMW and Mercedes-Benz over time, focusing on the period from 1975 to 2015.
- **Best-Responses:** We calculated and visualized the best responses of both companies in terms of total max power, average fuel consumption, and acceleration.
- **Comparison of Results:** We compared our visualizations and best response trends with the corresponding plots from the original article, highlighting the similarities and differences.

Discrepancies and Differences

While our results closely follow the methodology of the original article, there are several reasons why they are not identical:

- **Dataset Differences:** The dataset we used is much larger than the one used in the original article. This difference in dataset size can significantly impact the results of the analysis. A larger dataset provides more data points and may reveal trends and variations that are not visible in a smaller dataset.
- **Data Quality and Filtering:** Variations in data quality and the criteria used for filtering data points can lead to differences in the results. Our preprocessing steps, such as filling NaN values and segmenting the data, might differ from those used in the original article.
- **Fitting and Visualization Methods:** Minor differences in the methods used for fitting curves and visualizing the data can also contribute to discrepancies. The choice of colormaps, axis scales, and other visualization parameters may affect the appearance and interpretation of the plots.

Conclusion

Despite these differences, the overall trends and insights derived from our analysis are quite similar to those presented in the original article. Our extended analysis, using a larger dataset, provides a more detailed and robust understanding of the best responses of BMW and Mercedes-Benz.