

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.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from matplotlib.colors import LinearSegmentedColormap
4
5 # Load the data
6 file_path = 'Car Dataset 1945-2020.csv'
7 car_data = pd.read_csv(file_path, low_memory=False)
8
9 # Define a custom colormap
10 cmap = LinearSegmentedColormap.from_list('custom_cmap', ['darkblue',
11     'lightblue', 'green', 'yellow', 'orange', 'red'])
12
13 # Segment the data by manufacturer
14 mercedes_data = car_data[car_data['Make'] == 'Mercedes-Benz'].copy()
15
16 bmw_data = car_data[car_data['Make'] == 'BMW'].copy()
17
18 # Function to preprocess data and fill NaN values with median
19 def preprocess_and_fill_nan(data):
20     # Filter data for the years 1975 to 2015
21     data = data[(data['Year_from'] >= 1975) & (data['Year_from'] <=
22         2015)].copy()
23     # Fill NaN values with the median
24     data.loc[:, 'engine_hp'] = data['engine_hp'].fillna(data['
25         engine_hp'].median())
26     data.loc[:, 'mixed_fuel_consumption_per_100_km_l'] = data['
27         mixed_fuel_consumption_per_100_km_l'].fillna(data['
28         mixed_fuel_consumption_per_100_km_l'].median())
29     return data
30
31 # Preprocess and fill NaN values for Mercedes-Benz and BMW data
32 mercedes_data = preprocess_and_fill_nan(mercedes_data)
33 bmw_data = preprocess_and_fill_nan(bmw_data)
34
35 # Print the number of data points for BMW and Mercedes-Benz
36 print(f"Number of BMW data points: {len(bmw_data)}")
37 print(f"Number of Mercedes-Benz data points: {len(mercedes_data)}")
38
39 # Function to create scatter plots with actual release years
40 def plot_car_data(data, manufacturer):
41     plt.figure(figsize=(12, 6))
42     scatter = plt.scatter(data['engine_hp'], data['
43         mixed_fuel_consumption_per_100_km_l'], c=data['Year_from'],
```

```

        cmap=cmap)
37     cbar = plt.colorbar(scatter, label='Year')
38     plt.title(f'Development of Car Models by {manufacturer}')
39     plt.xlabel('Max Engine Power (HP)')
40     plt.ylabel('Average Fuel Consumption (l/100 km)')
41     plt.grid(True)
42     plt.show()
43
44 # Function to create combined scatter plots for Mercedes-Benz and
    BMW
45 def plot_combined_car_data(data_a, data_b):
46     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12), sharex=
        True, sharey=True)
47
48     scatter1 = ax1.scatter(data_a['engine_hp'], data_a['
        mixed_fuel_consumption_per_100_km_l'], c=data_a['Year_from'
        ], cmap=cmap)
49     ax1.set_title('Development of Car Models by Mercedes-Benz')
50     ax1.set_ylabel('Average Fuel Consumption (l/100 km)')
51     ax1.grid(True)
52
53     scatter2 = ax2.scatter(data_b['engine_hp'], data_b['
        mixed_fuel_consumption_per_100_km_l'], c=data_b['Year_from'
        ], cmap=cmap)
54     ax2.set_title('Development of Car Models by BMW')
55     ax2.set_xlabel('Max Engine Power (HP)')
56     ax2.set_ylabel('Average Fuel Consumption (l/100 km)')
57     ax2.grid(True)
58
59     # Create a color bar on the right side
60     cbar = fig.colorbar(scatter1, ax=[ax1, ax2], orientation='
        vertical', pad=0.05, shrink=0.8)
61     cbar.set_label('Year')
62
63     plt.subplots_adjust(right=0.75) # Adjust right side to make
        space for color bar
64     plt.show()
65
66 # Create scatter plots for Mercedes-Benz and BMW with the correct
    color scale
67 plot_car_data(mercedes_data, 'Mercedes-Benz')
68 plot_car_data(bmw_data, 'BMW')
69 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.

```

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

```

We load the dataset from a CSV file using pandas.

```

1 # Define a custom colormap
2 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.

```
1 # Segment the data by manufacturer
2 mercedes_data = car_data[car_data['Make'] == 'Mercedes-Benz'].copy()
3   ()
4   ()
5   ()
6   ()
7   ()
8   ()
9   ()
10  ()
11  ()
12  ()
13  ()
14  ()
15  ()
16  ()
17  ()
18  ()
19  ()
20  ()
21  ()
22  ()
23  ()
24  ()
25  ()
26  ()
27  ()
28  ()
29  ()
30  ()
31  ()
32  ()
33  ()
34  ()
35  ()
36  ()
37  ()
38  ()
39  ()
40  ()
41  ()
42  ()
43  ()
44  ()
45  ()
46  ()
47  ()
48  ()
49  ()
50  ()
51  ()
52  ()
53  ()
54  ()
55  ()
56  ()
57  ()
58  ()
59  ()
60  ()
61  ()
62  ()
63  ()
64  ()
65  ()
66  ()
67  ()
68  ()
69  ()
70  ()
71  ()
72  ()
73  ()
74  ()
75  ()
76  ()
77  ()
78  ()
79  ()
80  ()
81  ()
82  ()
83  ()
84  ()
85  ()
86  ()
87  ()
88  ()
89  ()
90  ()
91  ()
92  ()
93  ()
94  ()
95  ()
96  ()
97  ()
98  ()
99  ()
100  ()
101  ()
102  ()
103  ()
104  ()
105  ()
106  ()
107  ()
108  ()
109  ()
110  ()
111  ()
112  ()
113  ()
114  ()
115  ()
116  ()
117  ()
118  ()
119  ()
120  ()
121  ()
122  ()
123  ()
124  ()
125  ()
126  ()
127  ()
128  ()
129  ()
130  ()
131  ()
132  ()
133  ()
134  ()
135  ()
136  ()
137  ()
138  ()
139  ()
140  ()
141  ()
142  ()
143  ()
144  ()
145  ()
146  ()
147  ()
148  ()
149  ()
150  ()
151  ()
152  ()
153  ()
154  ()
155  ()
156  ()
157  ()
158  ()
159  ()
160  ()
161  ()
162  ()
163  ()
164  ()
165  ()
166  ()
167  ()
168  ()
169  ()
170  ()
171  ()
172  ()
173  ()
174  ()
175  ()
176  ()
177  ()
178  ()
179  ()
180  ()
181  ()
182  ()
183  ()
184  ()
185  ()
186  ()
187  ()
188  ()
189  ()
190  ()
191  ()
192  ()
193  ()
194  ()
195  ()
196  ()
197  ()
198  ()
199  ()
200  ()
201  ()
202  ()
203  ()
204  ()
205  ()
206  ()
207  ()
208  ()
209  ()
210  ()
211  ()
212  ()
213  ()
214  ()
215  ()
216  ()
217  ()
218  ()
219  ()
220  ()
221  ()
222  ()
223  ()
224  ()
225  ()
226  ()
227  ()
228  ()
229  ()
230  ()
231  ()
232  ()
233  ()
234  ()
235  ()
236  ()
237  ()
238  ()
239  ()
240  ()
241  ()
242  ()
243  ()
244  ()
245  ()
246  ()
247  ()
248  ()
249  ()
250  ()
251  ()
252  ()
253  ()
254  ()
255  ()
256  ()
257  ()
258  ()
259  ()
260  ()
261  ()
262  ()
263  ()
264  ()
265  ()
266  ()
267  ()
268  ()
269  ()
270  ()
271  ()
272  ()
273  ()
274  ()
275  ()
276  ()
277  ()
278  ()
279  ()
280  ()
281  ()
282  ()
283  ()
284  ()
285  ()
286  ()
287  ()
288  ()
289  ()
290  ()
291  ()
292  ()
293  ()
294  ()
295  ()
296  ()
297  ()
298  ()
299  ()
300  ()
301  ()
302  ()
303  ()
304  ()
305  ()
306  ()
307  ()
308  ()
309  ()
310  ()
311  ()
312  ()
313  ()
314  ()
315  ()
316  ()
317  ()
318  ()
319  ()
320  ()
321  ()
322  ()
323  ()
324  ()
325  ()
326  ()
327  ()
328  ()
329  ()
330  ()
331  ()
332  ()
333  ()
334  ()
335  ()
336  ()
337  ()
338  ()
339  ()
340  ()
341  ()
342  ()
343  ()
344  ()
345  ()
346  ()
347  ()
348  ()
349  ()
350  ()
351  ()
352  ()
353  ()
354  ()
355  ()
356  ()
357  ()
358  ()
359  ()
360  ()
361  ()
362  ()
363  ()
364  ()
365  ()
366  ()
367  ()
368  ()
369  ()
370  ()
371  ()
372  ()
373  ()
374  ()
375  ()
376  ()
377  ()
378  ()
379  ()
380  ()
381  ()
382  ()
383  ()
384  ()
385  ()
386  ()
387  ()
388  ()
389  ()
390  ()
391  ()
392  ()
393  ()
394  ()
395  ()
396  ()
397  ()
398  ()
399  ()
400  ()
401  ()
402  ()
403  ()
404  ()
405  ()
406  ()
407  ()
408  ()
409  ()
410  ()
411  ()
412  ()
413  ()
414  ()
415  ()
416  ()
417  ()
418  ()
419  ()
420  ()
421  ()
422  ()
423  ()
424  ()
425  ()
426  ()
427  ()
428  ()
429  ()
430  ()
431  ()
432  ()
433  ()
434  ()
435  ()
436  ()
437  ()
438  ()
439  ()
440  ()
441  ()
442  ()
443  ()
444  ()
445  ()
446  ()
447  ()
448  ()
449  ()
450  ()
451  ()
452  ()
453  ()
454  ()
455  ()
456  ()
457  ()
458  ()
459  ()
460  ()
461  ()
462  ()
463  ()
464  ()
465  ()
466  ()
467  ()
468  ()
469  ()
470  ()
471  ()
472  ()
473  ()
474  ()
475  ()
476  ()
477  ()
478  ()
479  ()
480  ()
481  ()
482  ()
483  ()
484  ()
485  ()
486  ()
487  ()
488  ()
489  ()
490  ()
491  ()
492  ()
493  ()
494  ()
495  ()
496  ()
497  ()
498  ()
499  ()
500  ()
501  ()
502  ()
503  ()
504  ()
505  ()
506  ()
507  ()
508  ()
509  ()
510  ()
511  ()
512  ()
513  ()
514  ()
515  ()
516  ()
517  ()
518  ()
519  ()
520  ()
521  ()
522  ()
523  ()
524  ()
525  ()
526  ()
527  ()
528  ()
529  ()
530  ()
531  ()
532  ()
533  ()
534  ()
535  ()
536  ()
537  ()
538  ()
539  ()
540  ()
541  ()
542  ()
543  ()
544  ()
545  ()
546  ()
547  ()
548  ()
549  ()
550  ()
551  ()
552  ()
553  ()
554  ()
555  ()
556  ()
557  ()
558  ()
559  ()
560  ()
561  ()
562  ()
563  ()
564  ()
565  ()
566  ()
567  ()
568  ()
569  ()
570  ()
571  ()
572  ()
573  ()
574  ()
575  ()
576  ()
577  ()
578  ()
579  ()
580  ()
581  ()
582  ()
583  ()
584  ()
585  ()
586  ()
587  ()
588  ()
589  ()
590  ()
591  ()
592  ()
593  ()
594  ()
595  ()
596  ()
597  ()
598  ()
599  ()
600  ()
601  ()
602  ()
603  ()
604  ()
605  ()
606  ()
607  ()
608  ()
609  ()
610  ()
611  ()
612  ()
613  ()
614  ()
615  ()
616  ()
617  ()
618  ()
619  ()
620  ()
621  ()
622  ()
623  ()
624  ()
625  ()
626  ()
627  ()
628  ()
629  ()
630  ()
631  ()
632  ()
633  ()
634  ()
635  ()
636  ()
637  ()
638  ()
639  ()
640  ()
641  ()
642  ()
643  ()
644  ()
645  ()
646  ()
647  ()
648  ()
649  ()
650  ()
651  ()
652  ()
653  ()
654  ()
655  ()
656  ()
657  ()
658  ()
659  ()
660  ()
661  ()
662  ()
663  ()
664  ()
665  ()
666  ()
667  ()
668  ()
669  ()
670  ()
671  ()
672  ()
673  ()
674  ()
675  ()
676  ()
677  ()
678  ()
679  ()
680  ()
681  ()
682  ()
683  ()
684  ()
685  ()
686  ()
687  ()
688  ()
689  ()
690  ()
691  ()
692  ()
693  ()
694  ()
695  ()
696  ()
697  ()
698  ()
699  ()
700  ()
701  ()
702  ()
703  ()
704  ()
705  ()
706  ()
707  ()
708  ()
709  ()
710  ()
711  ()
712  ()
713  ()
714  ()
715  ()
716  ()
717  ()
718  ()
719  ()
720  ()
721  ()
722  ()
723  ()
724  ()
725  ()
726  ()
727  ()
728  ()
729  ()
730  ()
731  ()
732  ()
733  ()
734  ()
735  ()
736  ()
737  ()
738  ()
739  ()
740  ()
741  ()
742  ()
743  ()
744  ()
745  ()
746  ()
747  ()
748  ()
749  ()
750  ()
751  ()
752  ()
753  ()
754  ()
755  ()
756  ()
757  ()
758  ()
759  ()
760  ()
761  ()
762  ()
763  ()
764  ()
765  ()
766  ()
767  ()
768  ()
769  ()
770  ()
771  ()
772  ()
773  ()
774  ()
775  ()
776  ()
777  ()
778  ()
779  ()
780  ()
781  ()
782  ()
783  ()
784  ()
785  ()
786  ()
787  ()
788  ()
789  ()
790  ()
791  ()
792  ()
793  ()
794  ()
795  ()
796  ()
797  ()
798  ()
799  ()
800  ()
801  ()
802  ()
803  ()
804  ()
805  ()
806  ()
807  ()
808  ()
809  ()
810  ()
811  ()
812  ()
813  ()
814  ()
815  ()
816  ()
817  ()
818  ()
819  ()
820  ()
821  ()
822  ()
823  ()
824  ()
825  ()
826  ()
827  ()
828  ()
829  ()
830  ()
831  ()
832  ()
833  ()
834  ()
835  ()
836  ()
837  ()
838  ()
839  ()
840  ()
841  ()
842  ()
843  ()
844  ()
845  ()
846  ()
847  ()
848  ()
849  ()
850  ()
851  ()
852  ()
853  ()
854  ()
855  ()
856  ()
857  ()
858  ()
859  ()
860  ()
861  ()
862  ()
863  ()
864  ()
865  ()
866  ()
867  ()
868  ()
869  ()
870  ()
871  ()
872  ()
873  ()
874  ()
875  ()
876  ()
877  ()
878  ()
879  ()
880  ()
881  ()
882  ()
883  ()
884  ()
885  ()
886  ()
887  ()
888  ()
889  ()
890  ()
891  ()
892  ()
893  ()
894  ()
895  ()
896  ()
897  ()
898  ()
899  ()
900  ()
901  ()
902  ()
903  ()
904  ()
905  ()
906  ()
907  ()
908  ()
909  ()
910  ()
911  ()
912  ()
913  ()
914  ()
915  ()
916  ()
917  ()
918  ()
919  ()
920  ()
921  ()
922  ()
923  ()
924  ()
925  ()
926  ()
927  ()
928  ()
929  ()
930  ()
931  ()
932  ()
933  ()
934  ()
935  ()
936  ()
937  ()
938  ()
939  ()
940  ()
941  ()
942  ()
943  ()
944  ()
945  ()
946  ()
947  ()
948  ()
949  ()
950  ()
951  ()
952  ()
953  ()
954  ()
955  ()
956  ()
957  ()
958  ()
959  ()
960  ()
961  ()
962  ()
963  ()
964  ()
965  ()
966  ()
967  ()
968  ()
969  ()
970  ()
971  ()
972  ()
973  ()
974  ()
975  ()
976  ()
977  ()
978  ()
979  ()
980  ()
981  ()
982  ()
983  ()
984  ()
985  ()
986  ()
987  ()
988  ()
989  ()
990  ()
991  ()
992  ()
993  ()
994  ()
995  ()
996  ()
997  ()
998  ()
999  ()
1000  ()
```

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

```
1 # Function to preprocess data and fill NaN values with median
2 def preprocess_and_fill_nan(data):
3     # Filter data for the years 1975 to 2015
4     data = data[(data['Year_from'] >= 1975) & (data['Year_from'] <=
5         2015)].copy()
6     # Fill NaN values with the median
7     data.loc[:, 'engine_hp'] = data['engine_hp'].fillna(data['
8         engine_hp'].median())
9     data.loc[:, 'mixed_fuel_consumption_per_100_km_l'] = data['
10         mixed_fuel_consumption_per_100_km_l'].fillna(data['
11         mixed_fuel_consumption_per_100_km_l'].median())
12     return data
13
14 # Preprocess and fill NaN values for Mercedes-Benz and BMW data
15 mercedes_data = preprocess_and_fill_nan(mercedes_data)
16 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.

```
1 # Print the number of data points for BMW and Mercedes-Benz
2 print(f"Number of BMW data points: {len(bmw_data)}")
3 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.

```
1 # Function to create scatter plots with actual release years
2 def plot_car_data(data, manufacturer):
3     plt.figure(figsize=(12, 6))
4     scatter = plt.scatter(data['engine_hp'], data['
5         mixed_fuel_consumption_per_100_km_l'], c=data['Year_from'],
6         cmap=cmap)
7     cbar = plt.colorbar(scatter, label='Year')
8     plt.title(f'Development of Car Models by {manufacturer}')
9     plt.xlabel('Max Engine Power (HP)')
10    plt.ylabel('Average Fuel Consumption (l/100 km)')
11    plt.grid(True)
12    plt.show()
13
14 # Function to create combined scatter plots for Mercedes-Benz and
15 BMW
16 def plot_combined_car_data(data_a, data_b):
17     fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12), sharex=
18         True, sharey=True)
```

```

16 scatter1 = ax1.scatter(data_a['engine_hp'], data_a['
    mixed_fuel_consumption_per_100_km_l'], c=data_a['Year_from',
    ], cmap=cmap)
17 ax1.set_title('Development of Car Models by Mercedes-Benz')
18 ax1.set_ylabel('Average Fuel Consumption (l/100 km)')
19 ax1.grid(True)
20
21 scatter2 = ax2.scatter(data_b['engine_hp'], data_b['
    mixed_fuel_consumption_per_100_km_l'], c=data_b['Year_from',
    ], cmap=cmap)
22 ax2.set_title('Development of Car Models by BMW')
23 ax2.set_xlabel('Max Engine Power (HP)')
24 ax2.set_ylabel('Average Fuel Consumption (l/100 km)')
25 ax2.grid(True)
26
27 # Create a color bar on the right side
28 cbar = fig.colorbar(scatter1, ax=[ax1, ax2], orientation='
    vertical', pad=0.05, shrink=0.8)
29 cbar.set_label('Year')
30
31 plt.subplots_adjust(right=0.75) # Adjust right side to make
    space for color bar
32 plt.show()
33
34 # Create scatter plots for Mercedes-Benz and BMW with the correct
    color scale
35 plot_car_data(mercedes_data, 'Mercedes-Benz')
36 plot_car_data(bmw_data, 'BMW')
37 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.

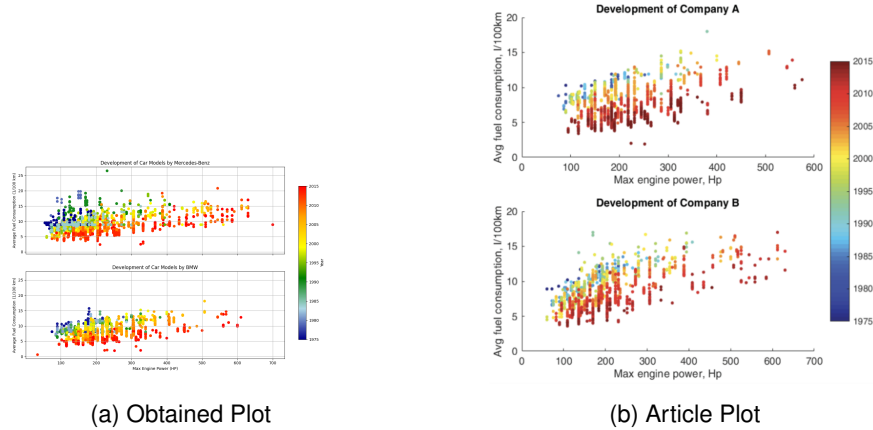


Figure 1: Comparison of Data Visualization

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.

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from scipy.stats import pearsonr
5
6 # Load the original data
7 file_path = 'Car Dataset 1945-2020.csv'
8 car_data = pd.read_csv(file_path, low_memory=False)
9
10 # Filter data for the years 1970 to 2020
11 filtered_data = car_data[(car_data['Year_from'] >= 1970) & (
12     car_data['Year_from'] <= 2017)]
13
14 # Segment the data by manufacturer
15 mercedes_data = filtered_data[filtered_data['Make'] == 'Mercedes-Benz']
16 bmw_data = filtered_data[filtered_data['Make'] == 'BMW']
17
18 # Function to remove outliers below the 0th percentile and above
19 # the 95th percentile
20 def remove_outliers_MaxPower(data, column):
21     percentile_5 = data[column].quantile(0.08)

```

```

20     percentile_95 = data[column].quantile(0.97)
21     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
22
23 def remove_outliers_Fuel(data, column):
24     percentile_5 = data[column].quantile(0.0)
25     percentile_95 = data[column].quantile(0.84)
26     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
27
28 def remove_outliers_Acceleration(data, column):
29     percentile_5 = data[column].quantile(0.0)
30     percentile_95 = data[column].quantile(0.97)
31     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
32
33 def remove_outliers_Weight(data, column):
34     percentile_5 = data[column].quantile(0.13)
35     percentile_95 = data[column].quantile(0.96)
36     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
37
38 def remove_outliers>LoadingCapacity(data, column):
39     percentile_5 = data[column].quantile(0.17)
40     percentile_95 = data[column].quantile(0.94)
41     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
42
43 def remove_outliers_Displacement(data, column):
44     percentile_5 = data[column].quantile(0.0)
45     percentile_95 = data[column].quantile(0.97)
46     return data[(data[column] >= percentile_5) & (data[column] <=
    percentile_95)]
47
48 ## TOTAL MAX POWER BEST RESPONSE // BEGIN
49
50 # Function to calculate the best response (max engine power) by
    year for a manufacturer
51 def calculate_best_response_power(data, manufacturer):
52     # Group by year and calculate the max engine power
53     best_response = data.groupby('Year_from')['engine_hp'].max().
        reset_index()
54     best_response.columns = ['Year_from', f'engine_hp_{manufacturer}
        .lower()}']
55     return best_response
56
57 # Calculate the best response for both manufacturers
58 best_response_power_mercedes = calculate_best_response_power(
    mercedes_data, 'Mercedes')
59 best_response_power_bmw = calculate_best_response_power(bmw_data, '
    BMW')
60
61 # Merge the two dataframes on Year_from to align the years
62 engine_hp_merged = pd.merge(best_response_power_mercedes,
    best_response_power_bmw, on='Year_from')
63
64 # Plot the trends of best response for both manufacturers

```

```

65 plt.figure(figsize=(12, 6))
66 plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['
    engine_hp_mercedes'], marker='o', linestyle='-', color='r',
    label='Mercedes-Benz')
67 plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['
    engine_hp_bmw'], marker='o', linestyle='-', color='b', label='
    BMW')
68 plt.title('Total Max Power: Best Response Trends of Mercedes-Benz
    and BMW')
69 plt.xlabel('Release Year')
70 plt.ylabel('Max Engine Power (HP)')
71 plt.ylim(0, max(engine_hp_merged['engine_hp_mercedes'].max(),
    engine_hp_merged['engine_hp_bmw'].max()) + 50) # Ensure
    consistent spacing
72 plt.legend()
73 plt.grid(True)
74 plt.show()
75
76 # Remove outliers for correlation calculation
77 engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged,
    'engine_hp_mercedes')
78 engine_hp_merged_clean = remove_outliers_MaxPower(
    engine_hp_merged_clean, 'engine_hp_bmw')
79
80 # Calculate correlation and p-value for engine power
81 engine_hp_corr, engine_hp_pval = pearsonr(engine_hp_merged_clean['
    engine_hp_mercedes'], engine_hp_merged_clean['engine_hp_bmw'])
82 print(f"Correlation for Engine Power: {engine_hp_corr}, p-value: {
    engine_hp_pval}")
83
84 ## // END
85
86 ## Average Fuel Consumption // Begin
87
88 # Calculate the average fuel consumption by year for each
    manufacturer
89 avg_consumption_mercedes = mercedes_data.groupby('Year_from')['
    mixed_fuel_consumption_per_100_km_l'].min().reset_index()
90 avg_consumption_bmw = bmw_data.groupby('Year_from')['
    mixed_fuel_consumption_per_100_km_l'].min().reset_index()
91
92 # Merge the two dataframes on Year_from to align the years
93 fuel_consumption_merged = pd.merge(avg_consumption_mercedes,
    avg_consumption_bmw, on='Year_from', suffixes=('_mercedes', '
    _bmw'))
94
95 # Plot the trends of average fuel consumption for both
    manufacturers
96 plt.figure(figsize=(12, 6))
97 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')
98 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')

```



```

99 plt.title('Average Fuel Consumption: Best Response Trends of
    Mercedes-Benz and BMW')
100 plt.xlabel('Release Year')
101 plt.ylabel('Average Fuel Consumption (1/100 km)')
102 plt.yticks(np.arange(0, 21, 2)) # Set y-axis ticks to increments
    of 2 from 0 to 20
103 plt.ylim(0, 20) # Set y-axis limits from 0 to 20
104 plt.legend()
105 plt.grid(True)
106 plt.show()
107
108 # Remove outliers for correlation calculation
109 fuel_consumption_merged = remove_outliers_Fuel(
    fuel_consumption_merged, '
    mixed_fuel_consumption_per_100_km_l_mercedes')
110 fuel_consumption_merged = remove_outliers_Fuel(
    fuel_consumption_merged, '
    mixed_fuel_consumption_per_100_km_l_bmw')
111
112 # Calculate correlation and p-value for fuel consumption
113 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'])
114 print(f"Correlation for Fuel Consumption: {fuel_consumption_corr},
    p-value: {fuel_consumption_pval}")
115
116 ## // END
117
118 ## Acceleration // Begin
119
120 # Function to calculate the best response (min acceleration time)
    by year for a manufacturer
121 def calculate_best_response_acceleration(data, manufacturer):
122     # Group by year and calculate the min acceleration time
123     best_response = data.groupby('Year_from')['
    acceleration_0_100_km/h_s'].min().reset_index()
124     best_response.columns = ['Year_from', f'acceleration_0_100_km/
    h_s_{manufacturer.lower()}']
125     return best_response
126
127 # Calculate the best response for both manufacturers
128 best_response_acceleration_mercedes =
    calculate_best_response_acceleration(mercedes_data, 'Mercedes')
129 best_response_acceleration_bmw =
    calculate_best_response_acceleration(bmw_data, 'BMW')
130
131 # Merge the two dataframes on Year_from to align the years
132 acceleration_merged = pd.merge(best_response_acceleration_mercedes,
    best_response_acceleration_bmw, on='Year_from')
133
134 # Plot the trends of best response acceleration for both
    manufacturers
135 plt.figure(figsize=(12, 6))
136 plt.plot(acceleration_merged['Year_from'], acceleration_merged['
    acceleration_0_100_km/h_s_mercedes'], marker='o', linestyle='-',

```

```

    , color='r', label='Mercedes-Benz')
137 plt.plot(acceleration_merged['Year_from'], acceleration_merged['
    acceleration_0_100_kmh_s_bmw'], marker='o', linestyle='-',
    color='b', label='BMW')
138 plt.title('Acceleration: Best Response Trends of Mercedes-Benz and
    BMW')
139 plt.xlabel('Release Year')
140 plt.ylabel('Acceleration (0-100 km/h)')
141 plt.yticks(np.arange(0, 20, 5)) # Set y-axis ticks to increments
    of 5 from 0 to 20
142 plt.ylim(0, 20)
143 plt.legend()
144 plt.grid(True)
145 plt.show()
146
147 # Remove outliers for correlation calculation
148 acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged, 'acceleration_0_100_kmh_s_mercedes')
149 acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged_clean, 'acceleration_0_100_kmh_s_bmw')
150
151 # Calculate correlation and p-value for acceleration
152 acceleration_corr, acceleration_pval = pearsonr(
    acceleration_merged_clean['acceleration_0_100_kmh_s_mercedes']
    , acceleration_merged_clean['acceleration_0_100_kmh_s_bmw'])
153 print(f"Correlation for Acceleration: {acceleration_corr}, p-value:
    {acceleration_pval}")
154
155 ## // 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

```

1 # Function to calculate the best response (max engine power) by
    year for a manufacturer
2 def calculate_best_response_power(data, manufacturer):
3     # Group by year and calculate the max engine power
4     best_response = data.groupby('Year_from')['engine_hp'].max().
        reset_index()
5     best_response.columns = ['Year_from', f'engine_hp_{manufacturer
        .lower()}']
6     return best_response
7
8 # Calculate the best response for both manufacturers
9 best_response_power_mercedes = calculate_best_response_power(
    mercedes_data, 'Mercedes')
10 best_response_power_bmw = calculate_best_response_power(bmw_data, '
    BMW')
11
12 # Merge the two dataframes on Year_from to align the years
13 engine_hp_merged = pd.merge(best_response_power_mercedes,
    best_response_power_bmw, on='Year_from')
14
15 # Plot the trends of best response for both manufacturers

```

```

16 plt.figure(figsize=(12, 6))
17 plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['
    engine_hp_mercedes'], marker='o', linestyle='-', color='r',
    label='Mercedes-Benz')
18 plt.plot(engine_hp_merged['Year_from'], engine_hp_merged['
    engine_hp_bmw'], marker='o', linestyle='-', color='b', label='
    BMW')
19 plt.title('Total Max Power: Best Response Trends of Mercedes-Benz
    and BMW')
20 plt.xlabel('Release Year')
21 plt.ylabel('Max Engine Power (HP)')
22 plt.ylim(0, max(engine_hp_merged['engine_hp_mercedes'].max(),
    engine_hp_merged['engine_hp_bmw'].max()) + 50) # Ensure
    consistent spacing
23 plt.legend()
24 plt.grid(True)
25 plt.show()
26
27 # Remove outliers for correlation calculation
28 engine_hp_merged_clean = remove_outliers_MaxPower(engine_hp_merged,
    'engine_hp_mercedes')
29 engine_hp_merged_clean = remove_outliers_MaxPower(
    engine_hp_merged_clean, 'engine_hp_bmw')
30
31 # Calculate correlation and p-value for engine power
32 engine_hp_corr, engine_hp_pval = pearsonr(engine_hp_merged_clean['
    engine_hp_mercedes'], engine_hp_merged_clean['engine_hp_bmw'])
33 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

```

1 # Calculate the average fuel consumption by year for each
    manufacturer
2 avg_consumption_mercedes = mercedes_data.groupby('Year_from')['
    mixed_fuel_consumption_per_100_km_l'].min().reset_index()
3 avg_consumption_bmw = bmw_data.groupby('Year_from')['
    mixed_fuel_consumption_per_100_km_l'].min().reset_index()
4
5 # Merge the two dataframes on Year_from to align the years
6 fuel_consumption_merged = pd.merge(avg_consumption_mercedes,
    avg_consumption_bmw, on='Year_from', suffixes=('_mercedes', '
    _bmw'))
7
8 # Plot the trends of average fuel consumption for both
    manufacturers
9 plt.figure(figsize=(12, 6))
10 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')
11 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')

```

```

12 plt.title('Average Fuel Consumption: Best Response Trends of
    Mercedes-Benz and BMW')
13 plt.xlabel('Release Year')
14 plt.ylabel('Average Fuel Consumption (l/100 km)')
15 plt.yticks(np.arange(0, 21, 2)) # Set y-axis ticks to increments
    of 2 from 0 to 20
16 plt.ylim(0, 20) # Set y-axis limits from 0 to 20
17 plt.legend()
18 plt.grid(True)
19 plt.show()
20
21 # Remove outliers for correlation calculation
22 fuel_consumption_merged = remove_outliers_Fuel(
    fuel_consumption_merged, '
    mixed_fuel_consumption_per_100_km_l_mercedes')
23 fuel_consumption_merged = remove_outliers_Fuel(
    fuel_consumption_merged, '
    mixed_fuel_consumption_per_100_km_l_bmw')
24
25 # Calculate correlation and p-value for fuel consumption
26 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'])
27 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

```

1 # Function to calculate the best response (min acceleration time)
    by year for a manufacturer
2 def calculate_best_response_acceleration(data, manufacturer):
3     # Group by year and calculate the min acceleration time
4     best_response = data.groupby('Year_from')['
        acceleration_0_100_km/h_s'].min().reset_index()
5     best_response.columns = ['Year_from', f'acceleration_0_100_km/
        h_s_{manufacturer.lower()}']
6     return best_response
7
8 # Calculate the best response for both manufacturers
9 best_response_acceleration_mercedes =
    calculate_best_response_acceleration(mercedes_data, 'Mercedes')
10 best_response_acceleration_bmw =
    calculate_best_response_acceleration(bmw_data, 'BMW')
11
12 # Merge the two dataframes on Year_from to align the years
13 acceleration_merged = pd.merge(best_response_acceleration_mercedes,
    best_response_acceleration_bmw, on='Year_from')
14
15 # Plot the trends of best response acceleration for both
    manufacturers
16 plt.figure(figsize=(12, 6))
17 plt.plot(acceleration_merged['Year_from'], acceleration_merged['
    acceleration_0_100_km/h_s_mercedes'], marker='o', linestyle='-',
    color='r', label='Mercedes-Benz')

```

```

18 plt.plot(acceleration_merged['Year_from'], acceleration_merged['
    acceleration_0_100_kmh_s_bmw'], marker='o', linestyle='--',
    color='b', label='BMW')
19 plt.title('Acceleration: Best Response Trends of Mercedes-Benz and
    BMW')
20 plt.xlabel('Release Year')
21 plt.ylabel('Acceleration (0-100 km/h)')
22 plt.yticks(np.arange(0, 20, 5)) # Set y-axis ticks to increments
    of 5 from 0 to 20
23 plt.ylim(0, 20)
24 plt.legend()
25 plt.grid(True)
26 plt.show()
27
28 # Remove outliers for correlation calculation
29 acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged, 'acceleration_0_100_kmh_s_mercedes')
30 acceleration_merged_clean = remove_outliers_Acceleration(
    acceleration_merged_clean, 'acceleration_0_100_kmh_s_bmw')
31
32 # Calculate correlation and p-value for acceleration
33 acceleration_corr, acceleration_pval = pearsonr(
    acceleration_merged_clean['acceleration_0_100_kmh_s_mercedes'],
    acceleration_merged_clean['acceleration_0_100_kmh_s_bmw'])
34 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 load capacity, and displacement, is optional.

Full Weight (using curb weight)

```

1 # Function to calculate the best response (max weight) by year for
    a manufacturer
2 def calculate_best_response_weight(data, column, manufacturer):
3     # Ensure the column is numeric and handle NaNs
4     data.loc[:, column] = pd.to_numeric(data[column], errors='
        coerce')
5     data = data.dropna(subset=[column])
6
7     # Group by year and calculate the max weight
8     best_response = data.groupby('Year_from')[column].mean().
        reset_index()
9     best_response.columns = ['Year_from', f'{column}_{manufacturer.
        lower()}']
10    return best_response
11
12 # Calculate the best response for both manufacturers using curb
    weight
13 best_response_weight_mercedes = calculate_best_response_weight(
    mercedes_data, 'curb_weight_kg', 'Mercedes')
14 best_response_weight_bmw = calculate_best_response_weight(bmw_data,
    'curb_weight_kg', 'BMW')
15
16 # Merge the two dataframes on Year_from to align the years
17 weight_merged = pd.merge(best_response_weight_mercedes,

```

```

    best_response_weight_bmw, on='Year_from')
18
19 # Plot the trends of best response for weight for both
    manufacturers
20 plt.figure(figsize=(12, 6))
21 plt.plot(weight_merged['Year_from'], weight_merged['
    curb_weight_kg_mercedes'], marker='o', linestyle='-', color='r',
    label='Mercedes-Benz')
22 plt.plot(weight_merged['Year_from'], weight_merged['
    curb_weight_kg_bmw'], marker='o', linestyle='-', color='b',
    label='BMW')
23 plt.title('Empty Mass: Best Response Trends of Mercedes-Benz and
    BMW')
24 plt.xlabel('Release Year')
25 plt.ylabel('Max Curb Weight (kg)')
26 plt.legend()
27 plt.grid(True)
28 plt.show()
29
30 # Remove outliers for correlation calculation
31 weight_merged_clean = remove_outliers_Weight(weight_merged, '
    curb_weight_kg_mercedes')
32 weight_merged_clean = remove_outliers_Weight(weight_merged_clean, '
    curb_weight_kg_bmw')
33
34 # Calculate correlation and p-value for weight
35 weight_corr, weight_pval = pearsonr(weight_merged_clean['
    curb_weight_kg_mercedes'], weight_merged_clean['
    curb_weight_kg_bmw'])
36 print(f"Correlation for Curb Weight: {weight_corr}, p-value: {
    weight_pval}")

```

Max Loading Capacity

```

1 # Function to calculate the best response (max full weight) by year
    for a manufacturer
2 def calculate_best_response_full_weight(data, column, manufacturer)
    :
3     # Ensure the column is numeric and handle NaNs
4     data.loc[:, column] = pd.to_numeric(data[column], errors='
        coerce')
5     data = data.dropna(subset=[column])
6
7     # Group by year and calculate the max full weight
8     best_response = data.groupby('Year_from')[column].max().
        reset_index()
9     best_response.columns = ['Year_from', f'{column}_{manufacturer.
        lower()}']
10    return best_response
11
12 # Calculate the best response for both manufacturers using full
    weight
13 best_response_full_weight_mercedes =
    calculate_best_response_full_weight(mercedes_data, '
    full_weight_kg', 'Mercedes')
14 best_response_full_weight_bmw = calculate_best_response_full_weight
    (bmw_data, 'full_weight_kg', 'BMW')
15

```

```

16 # Merge the two dataframes on Year_from to align the years
17 full_weight_merged = pd.merge(best_response_full_weight_mercedes,
    best_response_full_weight_bmw, on='Year_from')
18
19 # Plot the trends of best response for full weight for both
    manufacturers
20 plt.figure(figsize=(12, 6))
21 plt.plot(full_weight_merged['Year_from'], full_weight_merged['
    full_weight_kg_mercedes'], marker='o', linestyle='--', color='r',
    label='Mercedes-Benz')
22 plt.plot(full_weight_merged['Year_from'], full_weight_merged['
    full_weight_kg_bmw'], marker='o', linestyle='--', color='b',
    label='BMW')
23 plt.title('Loading Capacity: Best Response Trends of Mercedes-Benz
    and BMW')
24 plt.xlabel('Release Year')
25 plt.ylabel('Max Full Weight (kg)')
26 plt.legend()
27 plt.grid(True)
28 plt.show()
29
30 # Remove outliers for correlation calculation
31 full_weight_merged_clean = remove_outliers>LoadingCapacity(
    full_weight_merged, 'full_weight_kg_mercedes')
32 full_weight_merged_clean = remove_outliers>LoadingCapacity(
    full_weight_merged_clean, 'full_weight_kg_bmw')
33
34 # Calculate correlation and p-value for full weight
35 full_weight_corr, full_weight_pval = pearsonr(
    full_weight_merged_clean['full_weight_kg_mercedes'],
    full_weight_merged_clean['full_weight_kg_bmw'])
36 print(f"Correlation for Full Weight: {full_weight_corr}, p-value: {
    full_weight_pval}")

```

Displacement

```

1 # Ensure capacity_cm3 column is numeric, coerce errors to NaN
2 mercedes_data.loc[:, 'capacity_cm3'] = pd.to_numeric(mercedes_data[
    'capacity_cm3'], errors='coerce')
3 bmw_data.loc[:, 'capacity_cm3'] = pd.to_numeric(bmw_data['
    capacity_cm3'], errors='coerce')
4
5 # Create copies of the dataframes to avoid SettingWithCopyWarning
6 mercedes_data_clean = mercedes_data.dropna(subset=['capacity_cm3'])
    .copy()
7 bmw_data_clean = bmw_data.dropna(subset=['capacity_cm3']).copy()
8
9 # Calculate the best response (max displacement) by year for each
    manufacturer
10 best_response_displacement_mercedes = mercedes_data_clean.groupby('
    Year_from')['capacity_cm3'].max().reset_index()
11 best_response_displacement_bmw = bmw_data_clean.groupby('Year_from'
    )['capacity_cm3'].max().reset_index()
12
13 # Merge the two dataframes on Year_from to align the years
14 displacement_merged = pd.merge(best_response_displacement_mercedes,
    best_response_displacement_bmw, on='Year_from', suffixes=(
    '_mercedes', '_bmw'))

```

```

15
16 # Plot the trends of best response for displacement for both
    manufacturers
17 plt.figure(figsize=(12, 6))
18 plt.plot(displacement_merged['Year_from'], displacement_merged['
    capacity_cm3_mercedes'], marker='o', linestyle='--', color='r',
    label='Mercedes-Benz')
19 plt.plot(displacement_merged['Year_from'], displacement_merged['
    capacity_cm3_bmw'], marker='o', linestyle='--', color='b', label
    ='BMW')
20 plt.title('Displacement: Best Response Trends of Mercedes-Benz and
    BMW')
21 plt.xlabel('Release Year')
22 plt.ylabel('Max Displacement (cm )')
23 plt.legend()
24 plt.grid(True)
25 plt.show()
26
27 # Remove outliers for correlation calculation
28 displacement_merged = remove_outliers_Displacement(
    displacement_merged, 'capacity_cm3_mercedes')
29 displacement_merged = remove_outliers_Displacement(
    displacement_merged, 'capacity_cm3_bmw')
30
31 # Calculate correlation and p-value for displacement
32 displacement_corr, displacement_pval = pearsonr(displacement_merged
    ['capacity_cm3_mercedes'], displacement_merged['
    capacity_cm3_bmw'])
33 print(f"Correlation for Displacement: {displacement_corr}, p-value:
    {displacement_pval}")

```

Finally, we summarize the correlations for all metrics:

```

1 # Dynamic data from your analysis
2 correlation_data = {
3     'FOM': [
4         'Total Max Engine Power',
5         'Average Fuel Consumption',
6         'Acceleration',
7         'Empty Mass',
8         'Max Loading Capacity',
9         'Displacement'
10    ],
11    'Correlation': [
12        engine_hp_corr,
13        fuel_consumption_corr,
14        acceleration_corr,
15        weight_corr,
16        full_weight_corr,
17        displacement_corr
18    ],
19    'p-value': [
20        engine_hp_pval,
21        fuel_consumption_pval,
22        acceleration_pval,
23        weight_pval,
24        full_weight_pval,
25        displacement_pval

```



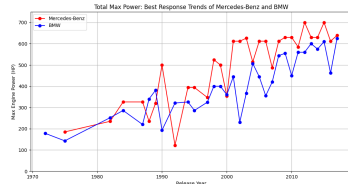
```

26     ]
27 }
28
29 # Round correlation values and format p-values
30 correlation_data['Correlation'] = [round(corr, 3) for corr in
    correlation_data['Correlation']]
31 correlation_data['p-value'] = ['{:0.2e}'.format(pval) for pval in
    correlation_data['p-value']]
32
33 # Create DataFrame
34 df = pd.DataFrame(correlation_data)
35
36 # Sort DataFrame by Correlation in descending order
37 df = df.sort_values(by='Correlation', ascending=False).reset_index(
    drop=True)
38
39 # Display the table
40 print("Correlation of Players FOM Best-Response Sets")
41 print(df)

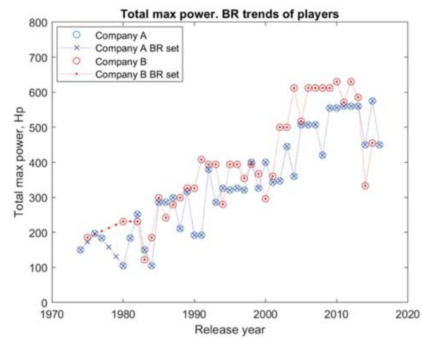
```

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.

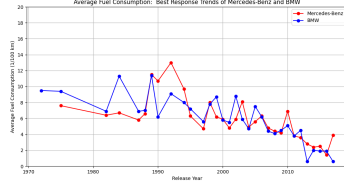


(a) Obtained Plot 1

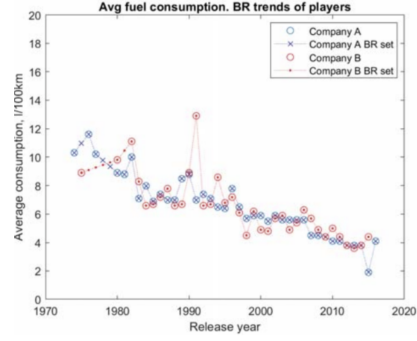


(b) Article Plot 1

Figure 2: Comparison of Best Response Trends - Part 1

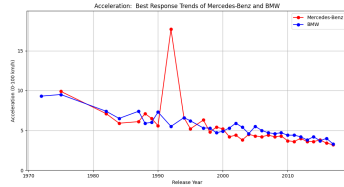


(a) Obtained Plot 2

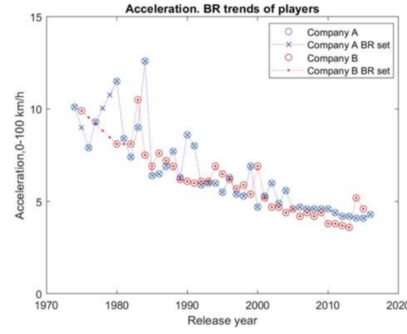


(b) Article Plot 2

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.