Problem Set III Solution

Tobias Bodentien Philipp Grunenberg Alexander Haas Osama Warshagha

17-12-2025

#Task 1

```
## renamed 101 files.
## did not rename 0 files:
```

Data preparation

Before starting with the feature extraction we need to prepare the data. For this we to match each entry of ranked-songs Excel to a song file. Then the song file is renamed consistently and unambiguously. The goal is to create uniform, machine-readable filenames that encode artist name, track title, and rating value, while being robust to variations in spelling, special characters, and filename conventions.

First, textual normalization is applied to both track titles and artist names. This is done by removing information that is only present in one data format by cutting off the string at the first occurrence of certain special characters: "(", "?", or "/". This decision reflects the assumption that content following these characters often contains remix names, featured artists, or live shows that are not consistently represented across data sources. After truncation, Unicode normalization (NFKD) is used to decompose accented characters, which are then converted to plain ASCII by discarding non-ASCII components. This step ensures that diacritics and special characters do not prevent successful string matching across files originating from different systems or encodings. Finally, leading and trailing whitespace is removed and all text is converted to lowercase to make subsequent comparisons case-insensitive.

Then, we match each entry from the Excel-based ranking table to an existing audio file. For each track, the code first tries to find a match based on the cleaned track title, checking whether the cleaned title is a substring of a cleaned filename. This substring-based matching allows for flexible alignment even if filenames contain additional tokens such as version numbers or formatting differences. If no match is found using the title, the code falls back to matching by artist name, again using the cleaned and normalized form. When matching by artist it could be possible that the wrong data is matched. Therefore we print the matches (when using artist) to manually check if the matching was correct. This makes the code not applicable to any dataset, but only to this one. If neither strategy succeeds, the track is explicitly reported as not found.

Once a match is identified, the file is renamed according to "artist_songName_rating.wav" while using the normalized strings. For the file with no rating an "x" is used instead of the rating.

Feature extraction

The goal is to predict a subjective star rating assigned by a person to a song. In practice, listeners tend to judge music based on timbre, energy, rhythm, harmony, and dynamics, not on individual frames or raw waveforms. For this reason, the feature pipeline is designed to extract robust, perceptually meaningful features and to aggregate them over time.

Preprocessing: All audio is resampled to a fixed sample rate and converted to mono to ensure comparability across songs and to avoid technical differences influencing the rating prediction. Stereo information rarely adds value for subjective ratings, while inconsistent sampling rates can bias spectral features. Loudness normalization is applied to prevent differences in recording level from dominating features such as RMS

energy, MFCCs, or spectral descriptors. Human listeners do not rate songs linearly by loudness, so this normalization helps align the features with perceptual judgments. Removing silence is also important. Intros and outros often contain quiet or silent segments that do not contribute to how a song is perceived. Trimming silence ensures that extracted features reflect the musical content itself and is supposed to improve the reliability of tempo, energy, and timbral features.

Features: MFCCs are included because they capture timbre and overall sound character, which strongly influence how pleasing or well-produced a song feels. They are a well-established and robust representation of perceptual sound qualities. In addition to static MFCCs, delta MFCCs are used to capture temporal changes in timbre, which relate to perceived dynamics and variation. Songs with evolving textures are often rated differently from static or monotonous ones.

Spectral features such as spectral centroid, bandwidth, rolloff, and flatness are included to describe brightness, sharpness, noisiness, and frequency distribution. These properties correlate with genre preferences, and perceived aggressiveness or softness, all of which can strongly affect subjective ratings.

In contrast to the other features RMS reflects the perceived intensity and not how something feels like. Zero-crossing rate provides a measure of the texture (soft <-> rough). Tempo captures an important emotional dimension, distinguishing calm tracks from energetic or danceable ones.

Chroma features are retained in a reduced form by using only their mean values. Chroma encodes harmonic and tonal information, which influences emotional response and musical preference. However, standard deviation adds relatively little information for rating prediction while increasing dimensionality.

LASSO

After extracting acoustic features for each song, we estimate a predictive model for the song ratings. Since we work with a relatively large set of potentially correlated audio features, we use **Lasso regression** as our main modeling approach. Lasso is a regularized linear regression that adds an **L1 penalty** to the loss function, which shrinks coefficients towards zero and can set some coefficients exactly to zero. This is beneficial in our setting because it (i) reduces the risk of overfitting and (ii) performs **automatic feature selection**, yielding a more interpretable model.

A key modeling decision is the choice of the regularization parameter λ, which controls the strength of shrinkage. We therefore estimate the model along a **regularization path** and select λ using two approaches discussed in the lecture: **(1) AICc-based selection** and **(2) K-fold cross-validation**. In the following, we document the results for Approach 1 (AICc); Approach 2 is presented afterwards and both approaches are compared.

Approach 1: Lasso with AICc-based model selection We fit the Lasso across a grid of λ values and select the model that minimizes the **corrected Akaike Information Criterion (AICc)**. An information criterion balances model fit and model complexity. We use **AICc rather than AIC** because AIC can be optimistic in finite samples, whereas AICc includes a small-sample correction and is therefore more conservative—an important consideration given that we only have **100 rated songs** for training.

AICc selection result

For each segment on the regularization path, we compute AICc and choose the model with the smallest value. The minimum is achieved at **segment `seg18`**, corresponding to

$$\lambda = 0.162886, \quad \mathrm{df} = 6, \quad \mathrm{AICc} = 2.338800, \quad R^2 = 0.152070.$$

The AICc profile drops when moving away from the most heavily regularized (nearly empty) model and reaches its minimum at this moderately sparse specification. For smaller λ (more complex models), AICc increases again, indicating that additional predictors do not compensate for the higher complexity penalty. Hence, AICc favors a parsimonious model.

Figure 1 visualizes the Lasso regularization path, showing the estimated coefficients as a function of $\log(\lambda)$. For large λ (right side), the penalty is strong and essentially all coefficients are shrunk to zero, corresponding to a very simple (nearly intercept-only) model. Moving left (smaller λ) reduces shrinkage and more predictors enter the model, as reflected by the increasing number of active coefficients shown above the plot. The dashed vertical line highlights the AICc-optimal λ (segment `seg18`, $\lambda \approx 0.163$), which lies in a moderately sparse region of the path: only a few MFCC-based coefficients remain non-zero, while many others are already shrunk to zero. This illustrates why AICc favors a parsimonious specification—additional predictors become active for smaller λ, but the implied increase in model complexity is not sufficiently compensated by improved fit.

Selected features and coefficients

The AICc-selected model contains an intercept and **five non-zero feature coefficients** (df = 6 includes the intercept). The non-zero coefficients are:

- **Intercept:** (1.36117)
- **mfcc_2_std:** (0.01982)
- **mfcc_delta_6_mean:** (22.98834)
- **mfcc_8_std:** (0.01276)
- **mfcc_delta_12_mean:** (34.84351)
- **mfcc_delta_13_mean:** (24.71623)

All selected coefficients are positive. Within the linear model, this implies that higher values of these MFCC-derived descriptors are associated with higher predicted ratings. Coefficient magnitudes should be interpreted with care because features can be on different scales; nevertheless, the signs and the set of selected predictors clearly indicate which acoustic dimensions the model uses. Among the selected predictors, **mfcc_delta_12_mean** has the largest coefficient, suggesting that variation in this delta MFCC component contributes most strongly to the linear predictor under the AICc-selected specification.

Prediction for the unknown song

Applying the AICc-selected Lasso model (segment `seg18`, $\lambda \approx 0.163$) to the unrated 101st song yields a predicted rating of $\hat{y} = 2.498992 \approx 2.50$ stars. Since ratings are defined on a 1–5 scale, we optionally clip predictions to the valid range $[1, 5]$; here, the prediction already lies within this interval and remains unchanged.

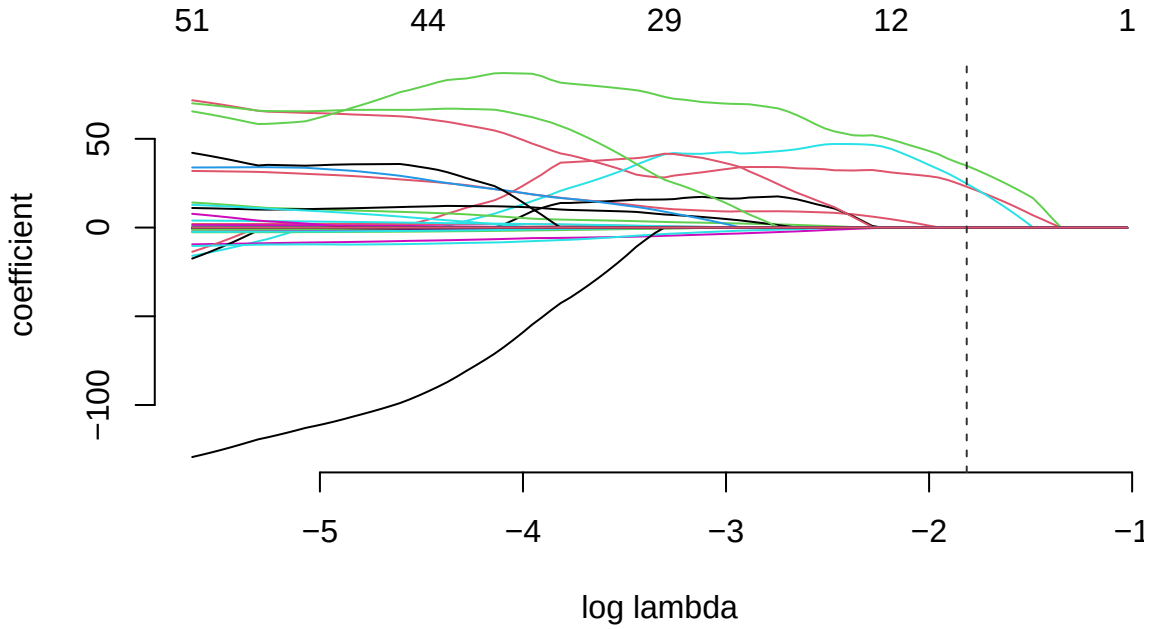


Figure 1: *Lasso coefficient paths for the extracted audio features as a function of* $\log(\lambda)$. *Each curve shows how one feature's coefficient is shrunk towards zero as regularization increases (moving to the right). The dashed*

vertical line marks the AICc-selected value of λ (segment `seg18`*, $\lambda \approx 0.163$), at which only a small subset of predictors remains non-zero ($\mathrm{df} = 6$ including the intercept). The numbers on top indicate the number of active (non-zero) coefficients along the regularization path.*

Approach 2: K-fold cross-validation with $\lambda_{\min}$ and 1-SE rule As a second model selection strategy, we apply **K-fold cross-validation (CV)** to choose the regularization strength λ. CV estimates out-of-sample prediction error by repeatedly training the model on subsets of the data and evaluating it on held-out folds. We report two standard choices:

- $\lambda_{\min}$: the λ that yields the **lowest average CV error** (best predictive performance under CV).
- λ_{1se} (1-SE rule): the **largest** λ whose CV error is still within **one standard error** of the minimum. This rule typically selects a **simpler** model with similar expected predictive performance and improved interpretability.

CV selection result

Cross-validation selects

$$\lambda_{\min} = 0.2476 \quad \text{and} \quad \lambda_{1se} = 0.3592.$$

As expected, λ_{1se} is larger (stronger regularization), leading to a substantially more parsimonious model. **Figure 2** visualizes this trade-off: the CV error curve is relatively flat around its minimum, so moving from $\lambda_{\min}$ to the larger λ_{1se} increases regularization while keeping the expected prediction error within one standard error. The numbers at the top of Figure 2 also show that the model becomes much sparser as λ increases, reflecting stronger shrinkage and fewer active coefficients.

Selected features and coefficients

At $\lambda_{\min}$, the model retains an intercept and **three** non-zero feature coefficients (**4** non-zero coefficients in total). The selected predictors are:

- **Intercept:** (1.6890)
- **mfcc_2_std:** (0.0147)
- **mfcc_delta_6_mean:** (2.2131)
- **mfcc_delta_12_mean:** (5.4652)

Thus, the CV-optimal model relies on a small subset of MFCC-based descriptors, again indicating that spectral/timbral characteristics are most informative for the rating signal in this linear setup.

In contrast, the **1-SE rule** yields a maximally sparse solution: at λ_{1se} the model keeps **only the intercept** (no active feature coefficients). This reflects strong shrinkage and suggests that, under the 1-SE criterion, the additional predictors do not improve expected predictive performance enough to justify the added complexity.

Prediction for the unknown song

Applying the cross-validated models to the unrated song gives:

- **CV ($\lambda_{\min}$):** $\hat{y} = 2.3466 \approx 2.35$ stars
- **CV (λ_{1se}):** $\hat{y} = 2.17$ stars

The 1-SE prediction equals the intercept-only estimate, i.e., it corresponds to predicting a constant rating for all songs. Both predictions lie within the valid 1–5 rating range, so clipping is not required.

Interpretation and comparison

Overall, CV with $\lambda_{\min}$ produces a **sparse but non-trivial** model (three features), whereas the 1-SE rule prioritizes **simplicity and stability**, resulting in an intercept-only baseline. This illustrates the practical trade-off between (i) maximizing estimated predictive accuracy ($\lambda_{\min}$) and (ii) selecting a more conservative, interpretable model with minimal risk of overfitting (λ_{1se}).

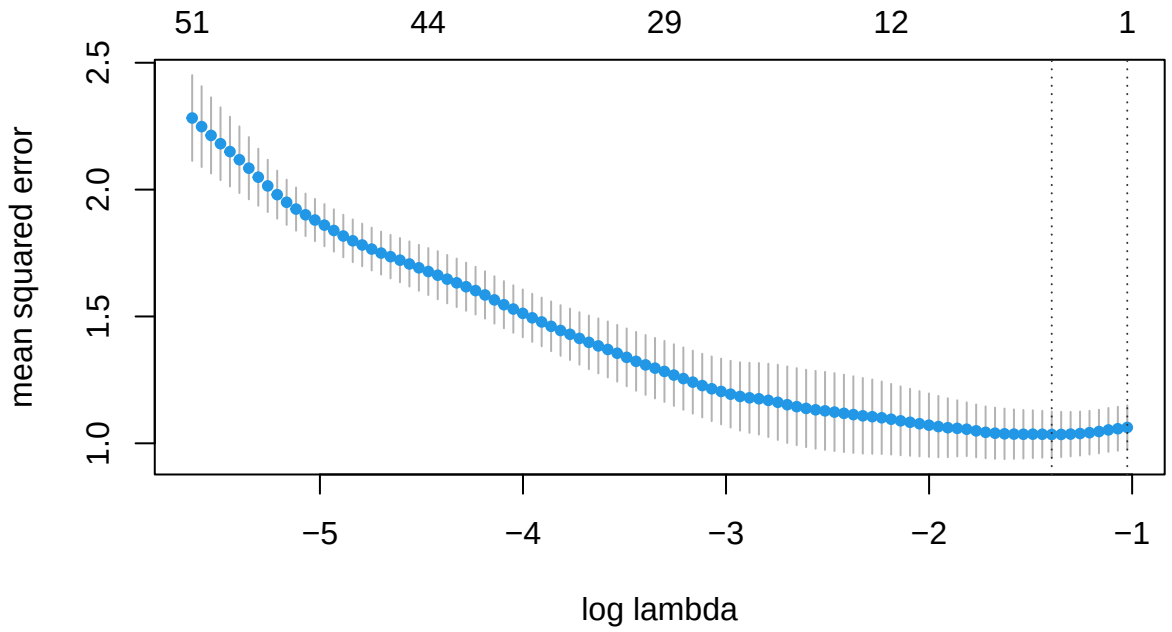


Figure 2: *K-fold cross-validation (CV) curve for the Lasso model.* The points show the mean squared prediction error (MSE) across folds for each value of $\log(\lambda)$, with vertical bars indicating ± 1 standard error. The dotted vertical lines mark $\lambda_{\min}$ (minimum CV error) and λ_{1se} (largest λ within one standard error of the minimum). Numbers at the top indicate the number of active (non-zero) coefficients at each λ.

Limitations and potential improvements

Limitations

1. **Very small sample size and noisy target.** We only have 100 labeled songs and a single subjective rating per song. Individual star ratings are inherently noisy (mood, context, familiarity), which makes the signal-to-noise ratio low. This is also reflected by the fact that cross-validation with the 1-SE rule selects an intercept-only model, indicating that the predictive signal in the current feature set is weak relative to the noise.

2. **Rating scale is bounded and ordinal, but modeled as continuous Gaussian.** We treat the 1–5 star ratings as a continuous outcome in a linear regression. In reality, the scale is **bounded** and arguably **ordinal**. This can yield predictions outside ([1,5]) (even if it did not happen here) and the linear loss does not reflect that "distance" between stars may not be perceived uniformly.

3. **Strong modeling assumptions: linearity and additivity.** Lasso in a linear model assumes that effects are additive and linear in the extracted features. Musical preferences can be highly **nonlinear** (e.g., liking high tempo only when timbre is "soft", or liking brightness only for specific harmonic content), and interactions are not captured by the current specification.

4. **Feature representation is heavily aggregated and may miss structure.** We summarize each track mostly by means/standard deviations. This discards information about **song structure** (verse/chorus changes), **drops**, **build-ups**, **rhythmic patterns**, and **long-range dynamics**, which can strongly shape preference.

5. **Feature interpretability is limited (MFCCs).** MFCC-based features are effective for audio characterization, but individual MFCC indices are not directly interpretable as "bass", "brightness", etc. Hence, conclusions about *why* the person likes something remain tentative.

Potential improvements

1. **Stability and robustness of selected features.** Lasso feature selection can be unstable under correlated predictors. A good improvement is to perform **stability selection / bootstrap**: re-fit the model many times on resampled data and report how frequently each feature is selected. This directly addresses the question whether the "chosen features" are robust or just artifacts of one sample split.
2. **Better treatment of the rating scale.** Options include:
   - **Clipping** predictions to ([1,5]) systematically (simple fix).
   - Modeling ratings as **ordinal** (ordered logit/probit with regularization), or as **classification** (e.g., 5 classes) if the goal is the star category rather than a real-valued score.
3. **Nonlinearities and interactions.** Still keeping the "regularized modeling" idea, one could:
   - add **interaction terms** (e.g., tempo × brightness proxies) and let Lasso select among them, or
   - switch to models that capture nonlinearities (e.g., kernel methods or tree-based models) as a robustness check—while keeping Lasso as the mandated baseline.

What can we learn about the individual's preferences from the selected features?

Across both AICc-selection and CV-($\lambda_{\min}$), the model consistently selects **MFCC-derived features**, especially **MFCC standard deviations** and **delta-MFCC means**. Interpreted cautiously, this suggests:

1. **Preference signal seems driven by timbral texture and timbral dynamics**, not by tempo or harmony. The model did *not* robustly select tempo, RMS, or chroma features. That indicates that—within this linear and aggregated setup—differences in *"how the song sounds"* (spectral envelope/texture) are more predictive than *"how fast it is"* or *"which harmonies it uses"*.
2. **Positive coefficients on delta-MFCC features hint at liking "movement" in sound.** Delta MFCCs capture **changes in timbre over time** (on the short-time scale). Positive weights can be read as: songs with more evolving texture/articulation (e.g., clearer transients, more modulation, less static sound) tend to receive higher ratings—*in the model's linear approximation*.
3. **However, the preference interpretation is not fully reliable yet.** Because (i) MFCC components are only indirectly interpretable, (ii) features are correlated, and (iii) selection can be unstable with (n=100), we should treat these as **hypotheses** about taste rather than firm psychological conclusions. A stability-selection analysis would tell us whether "MFCC deltas" are consistently selected and thus represent a genuine preference dimension.

#Task 2

Exploratory Data Analysis

Univariate Exploration

```
## # A tibble: 0 x 2
## # i 2 variables: Spalte <chr>, Anzahl_NAs <int>
```

Variables were categorized into track conditions (exogenous) and car features (actionable). The dataset exhibits no missing values. All variables fall within expected bounds, conditions range within $[1, 100]$ (excluding *LapDistance* and *Temperature*) and features within $[1, 500]$ rendering outlier removal and imputation unnecessary. *LapDistance* contains 23 distinct values identifying the simulated tracks, while the 41 unique *Temperature* observations reflect realistic meteorological variance.

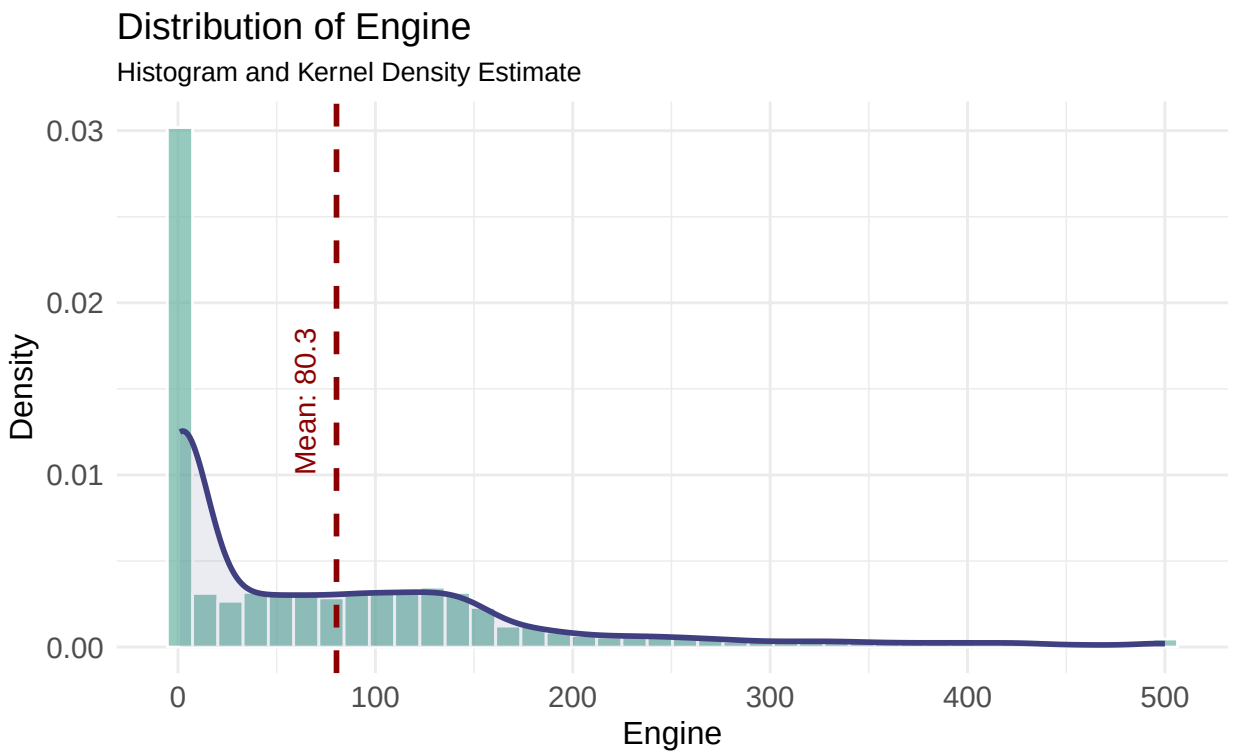
An analysis of feature distributions reveals a marked overrepresentation of boundary values within car features. For all variables, at least 10% of observations are located at the minimum or maximum limits. *Engine* displays the strongest skew, with 38% of values concentrated at the lower bound.

```r
#Check Extrem Vale Share of Car Features
simulator_data %>%
  select(all_of(car_features)) %>%
  pivot_longer(everything(), names_to = "Feature", values_to = "Value") %>%
  group_by(Feature) %>%
  summarise(
    Min = min(Value, na.rm = TRUE),
    Max = max(Value, na.rm = TRUE),
    Min_Share = mean(Value == min(Value, na.rm = TRUE)),
    Max_Share = mean(Value == max(Value, na.rm = TRUE))
  ) %>%
  arrange(desc(Min_Share + Max_Share)) %>%
  mutate(
    Min_Pct = paste0(round(Min_Share * 100, 1), "%"),
    Max_Pct = paste0(round(Max_Share * 100, 1), "%")
  ) %>%
  select(Feature, Min, Max, Min_Pct, Max_Pct)
```

```
## # A tibble: 6 x 5
##   Feature         Min   Max Min_Pct Max_Pct
##   <chr>         <dbl> <dbl> <chr>   <chr>
## 1 Engine            1   500 37%     0.6%
## 2 Differential      1   500 23.1%   2%
## 3 Rear Wing         1   500 3.8%    16.2%
## 4 Brake Balance     1   500 13.7%   1.6%
## 5 Front Wing        1   500 8.2%    5.6%
## 6 Suspension        1   500 11.3%   0%
```

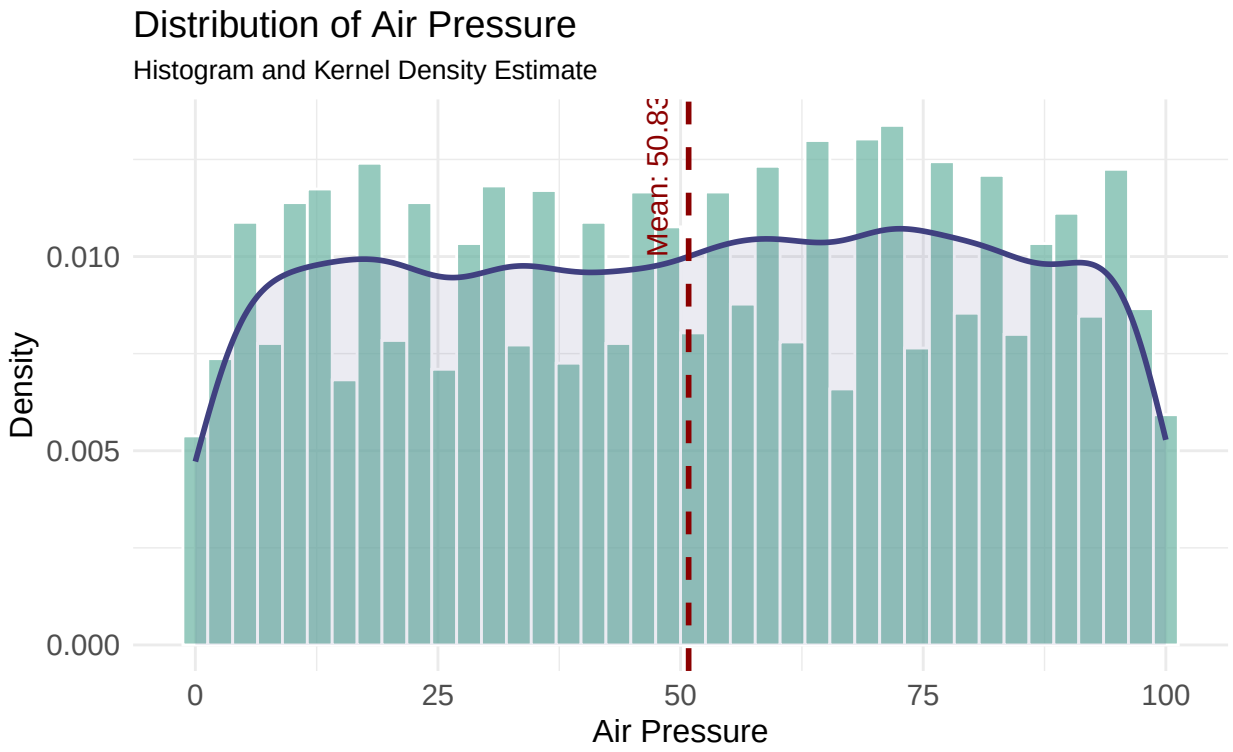
The *Engine* histogram visually demonstrates this boundary saturation, exhibiting the most severe overrepresentation among all features.

```r
create_histogram(simulator_data, "Engine")
```

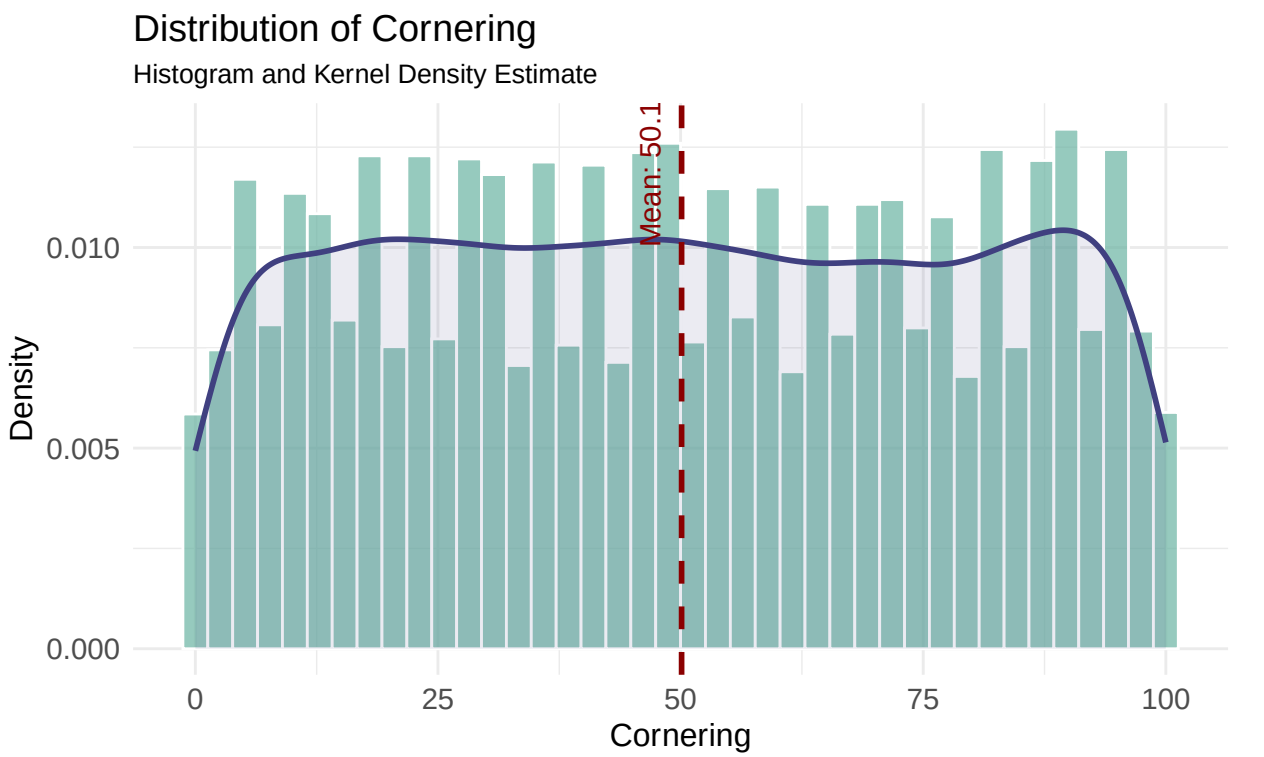


Condition features predominantly exhibit quasi-uniform distributions characterized by substantial noise and intermittent local spikes.

```
create_histogram(simulator_data, "Air Pressure")
```



```r
create_histogram(simulator_data, "Cornering")
```




```r
#test track features for unifomity
check_uniform <- function(x) {
  c(
    KS_p = ks.test(x, "punif", min(x), max(x))$p.value,
    X2_p = chisq.test(table(cut(x, breaks = 20)))$p.value
  )
}

sapply(simulator_data[condition_features], check_uniform)
```

```
##       Lap Distance  Cornering    Inclines       Camber        Grip
## KS_p  3.643314e-17 0.05815302 0.007071711 0.0196898152 0.005387613
## X2_p 6.150994e-210 0.13198222 0.658612009 0.0004165355 0.014248271
##      Wind (Avg. Speed)  Temperature   Humidity Air Density Air Pressure
## KS_p         0.1293900 5.572661e-05 0.06131801   0.1293900 1.985901e-05
## X2_p         0.1452199 1.310803e-09 0.27734418   0.2186831 4.810240e-02
##      Wind (Gusts)  Altitude  Roughness      Width
## KS_p   0.22020556 0.1177423 0.11224967 0.11224967
## X2_p   0.00261333 0.3308641 0.05365993 0.04402095
```

Kolmogorov-Smirnov and χ^2 tests fail to reject the null hypothesis of uniformity for most condition features. Exceptions include the structural variables *LapDistance* and *Temperature*, which are clearly non-uniform distributed as expected and *AirPressure*, where deviation is likely driven by noise artifacts rather than non-uniform underlying data generation.

Bivariate Exploration We continiue, by screening the dataset for pronounced correlations between variables.

```r
#check for correlations between features
cor_matrix <- cor(select_if(simulator_data, is.numeric))

cor_matrix[upper.tri(cor_matrix, diag = TRUE)] <- NA

sorted_correlations <- as.data.frame(as.table(cor_matrix)) %>%
  na.omit() %>%
  rename(Var1 = Var1, Var2 = Var2, Correlation = Freq) %>%
  arrange(desc(abs(Correlation)))

print(head(sorted_correlations, 20))
```

```
##             Var1          Var2 Correlation
## 1       Lap Time  Lap Distance   0.9986153
## 2     Front Wing     Rear Wing   0.6393497
## 3  Brake Balance     Cornering   0.5139970
## 4  Brake Balance         Width   0.5068418
## 5   Differential Brake Balance   0.4783069
## 6     Front Wing     Cornering   0.4593619
## 7     Front Wing  Air Pressure  -0.4431631
## 8   Differential     Cornering   0.4339289
## 9   Differential         Width   0.3792238
## 10     Rear Wing   Air Density  -0.3703582
## 11    Suspension          Grip   0.3595472
## 12     Rear Wing  Air Pressure  -0.3591280
## 13     Rear Wing     Cornering   0.3522601
## 14     Rear Wing      Inclines   0.3454922
## 15    Front Wing   Air Density  -0.3445982
## 16  Differential      Inclines   0.3427434
## 17    Suspension      Inclines  -0.3411002
## 18 Brake Balance     Roughness   0.2821323
## 19    Front Wing      Inclines   0.2802762
## 20        Engine          Grip   0.2790652
```

An extremely high correlation exceeding 0.99 is noted but deferred for later analysis. While we observe numerous high correlations, none involve $LapTime$; instead and all include at least one car feature. Therefore, the interpretation regarding the influence on $LapTime$ is uninteresting, but it reveals that assumptions concerning the interplay between conditions and car features were already established during the simulation.

An examination of correlations between the target, $LapTime$, and car features reveals only minor associations, with $Engine$ displaying the largest absolute coefficient.

```r
cor(simulator_data[car_features], simulator_data[target])
```

```
##                    Lap Time
## Rear Wing      0.0002672608
## Engine         0.0301833657
## Front Wing     0.0056714932
## Brake Balance -0.0016364509
## Differential   0.0187364945
## Suspension     0.0046960444
```

Correlations between conditions and $LapTime$ are negligible, with the exception of a dominant relationship with $LapDistance$. This is axiomatic, as $LapTime = \frac{LapDistance}{AverageVelocity}$. Since remaining features influence velocity rather than distance, directly modeling raw $LapTime$ is methodologically unsound.

```r
cor(simulator_data[condition_features], simulator_data[target])
```

```
##                         Lap Time
## Lap Distance       0.9986152665
## Cornering         -0.0057634038
## Inclines           0.0253879988
## Camber             0.0219954939
## Grip               0.0295169167
## Wind (Avg. Speed) -0.0180966722
## Temperature        0.0088015014
## Humidity           0.0109088370
## Air Density        0.0003391582
## Air Pressure       0.0085191811
## Wind (Gusts)      -0.0076233511
## Altitude          -0.0133649516
## Roughness          0.0087588544
## Width             -0.0002081736
```

To isolate vehicle performance, *LapTime* is regressed on *LapDistance*, and the resulting residuals are retained as the *AdjustedLapTime*. The regression indicates that *LapDistance* explains over 99% of the variance ($R^2 > 0.99$). The remaining residual standard deviation of 0.67 seconds implies that the variance attributable to car setup and noise, and consequently the optimization potential, is marginal.

```r
#distance adjust lap time
model_distance <- lm(`Lap Time` ~ `Lap Distance`, data = simulator_data)
simulator_data$lap_time_adjusted <- residuals(model_distance)
summary(model_distance)
```

```
##
## Call:
## lm(formula = `Lap Time` ~ `Lap Distance`, data = simulator_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.23734 -0.44830 -0.00984  0.43978  2.54753
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)     0.008717   0.041075    0.212    0.832
## `Lap Distance` 19.199324   0.010115 1898.047   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6691 on 9998 degrees of freedom
## Multiple R-squared:  0.9972, Adjusted R-squared:  0.9972
## F-statistic: 3.603e+06 on 1 and 9998 DF,  p-value: < 2.2e-16
```

Re-assessing correlations with *AdjustedLapTime* reveals discernible dependencies previously masked by track length. Notably, *Grip* exhibits a correlation of 0.12, implying that higher grip levels are associated with increased (slower) lap times.

```r
cor(simulator_data[setdiff(condition_features, "Lap Distance")], simulator_data['lap_time_adjusted'])
```

```
##                  lap_time_adjusted
## Cornering           0.0581516846
## Inclines            0.0511478926
```

```
## Camber                0.0301762948
## Grip                  0.1265279677
## Wind (Avg. Speed)    -0.0240154933
## Temperature           0.0757435864
## Humidity              0.0483356904
## Air Density           0.0818071844
## Air Pressure          0.0559074576
## Wind (Gusts)         -0.0125147386
## Altitude             -0.0387834120
## Roughness             0.0003697784
## Width                 0.0835076063
```

Dependencies involving car features are more pronounced. Specifically, *Engine* and *Differential* exhibit clear positive associations with the target, indicating that higher parameter settings correlate with slower lap times.

```
cor(simulator_data[car_features], simulator_data['lap_time_adjusted'])
```

```
##               lap_time_adjusted
## Rear Wing           -0.01897788
## Engine               0.26942547
## Front Wing          -0.01277838
## Brake Balance        0.10723910
## Differential         0.18185701
## Suspension           0.04902219
```

Spearman rank correlations were computed to capture general monotonic dependencies, extending the analysis beyond linear associations.

```
cor(simulator_data[setdiff(condition_features, "Lap Distance")], simulator_data['lap_time_adjusted'], m
```

```
##                   lap_time_adjusted
## Cornering             0.0601122594
## Inclines              0.0486781439
## Camber                0.0308840804
## Grip                  0.1248327225
## Wind (Avg. Speed)    -0.0173553861
## Temperature           0.0709475629
## Humidity              0.0473425587
## Air Density           0.0795879310
## Air Pressure          0.0569716227
## Wind (Gusts)         -0.0134541996
## Altitude             -0.0381893562
## Roughness             0.0003014934
## Width                 0.0816540295
```

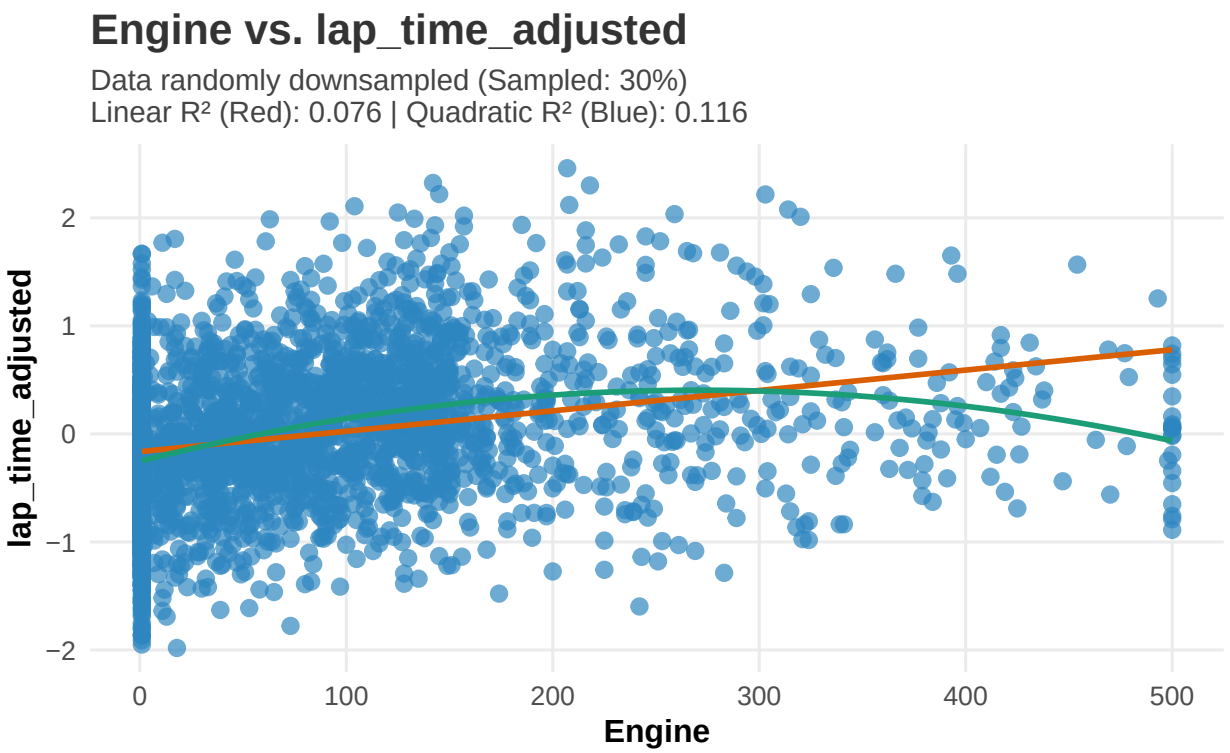
```
cor(simulator_data[car_features], simulator_data['lap_time_adjusted'], method='spearman')
```

```
##               lap_time_adjusted
## Rear Wing           -0.041722419
## Engine               0.331463398
## Front Wing          -0.008473425
## Brake Balance        0.120158549
## Differential         0.217568910
## Suspension           0.041486395
```

While coefficients for condition features remain largely unchanged, those for *Engine* and *Differential* increase significantly, providing evidence of non-linear monotonic relationships.

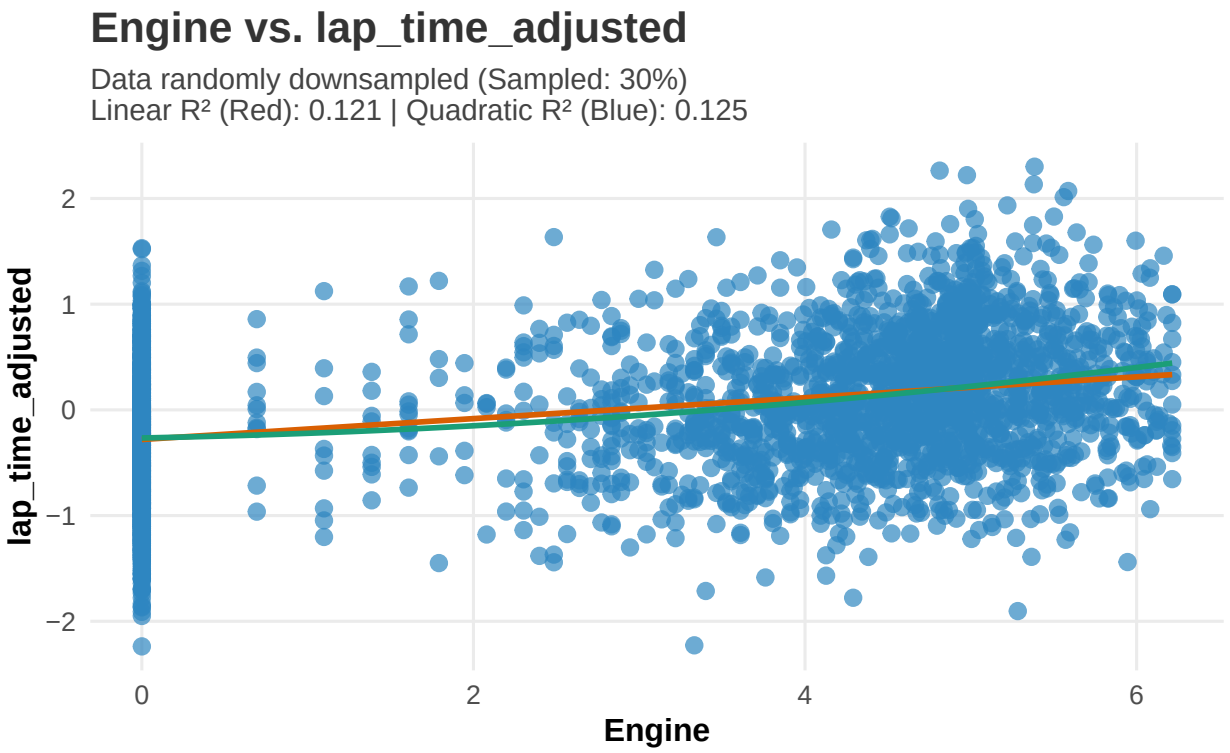
Scatter plots are utilized to further investigate the relationships involving *Engine* and *Differential*.

```
plot_2d_scatter(simulator_data, "Engine", "lap_time_adjusted", sample_rate = 0.3, trendline=TRUE)
```

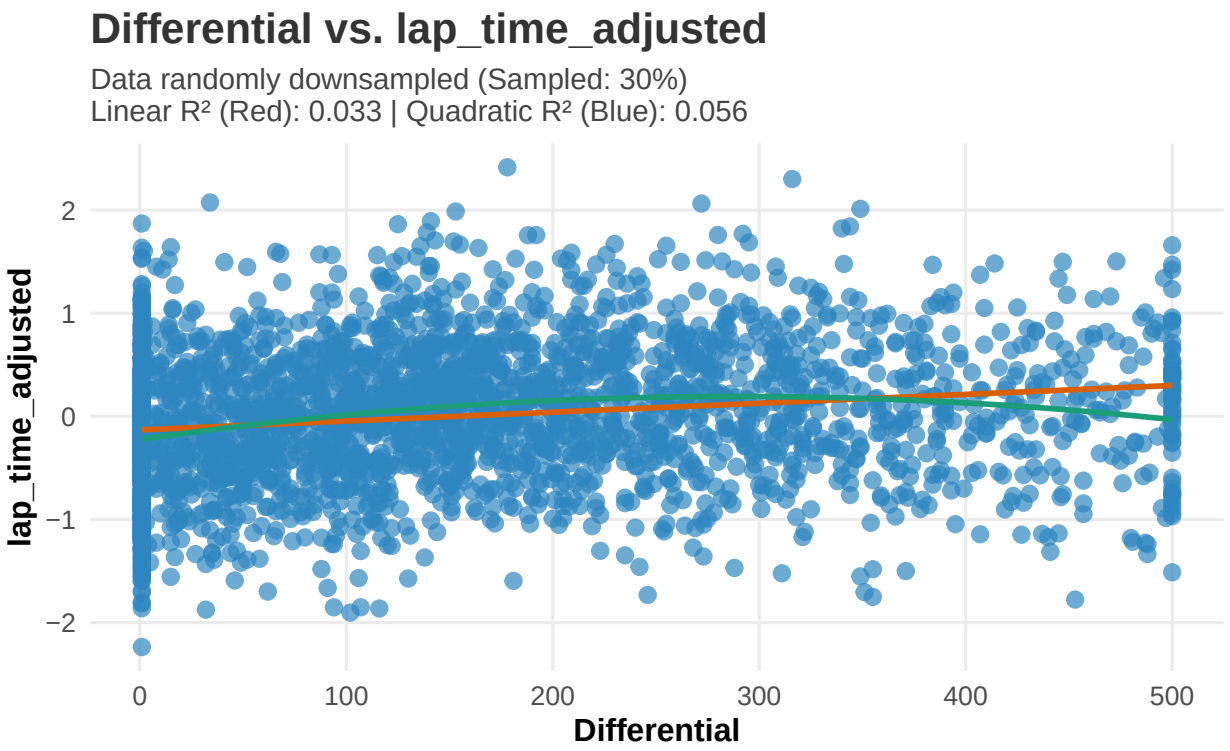


Consistent with the histograms, observations cluster densely at the lower bounds. Although the general trend is positive, a linear model proves insufficient. A quadratic regression yields a superior fit (higher R^2) compared to a logarithmic transformation, better capturing the non-linear structure of the data distribution.

```
logged_Engine = simulator_data %>% mutate(
  Engine =log(Engine)
)
plot_2d_scatter(logged_Engine, "Engine", "lap_time_adjusted", sample_rate = 0.3, trendline=TRUE)
```



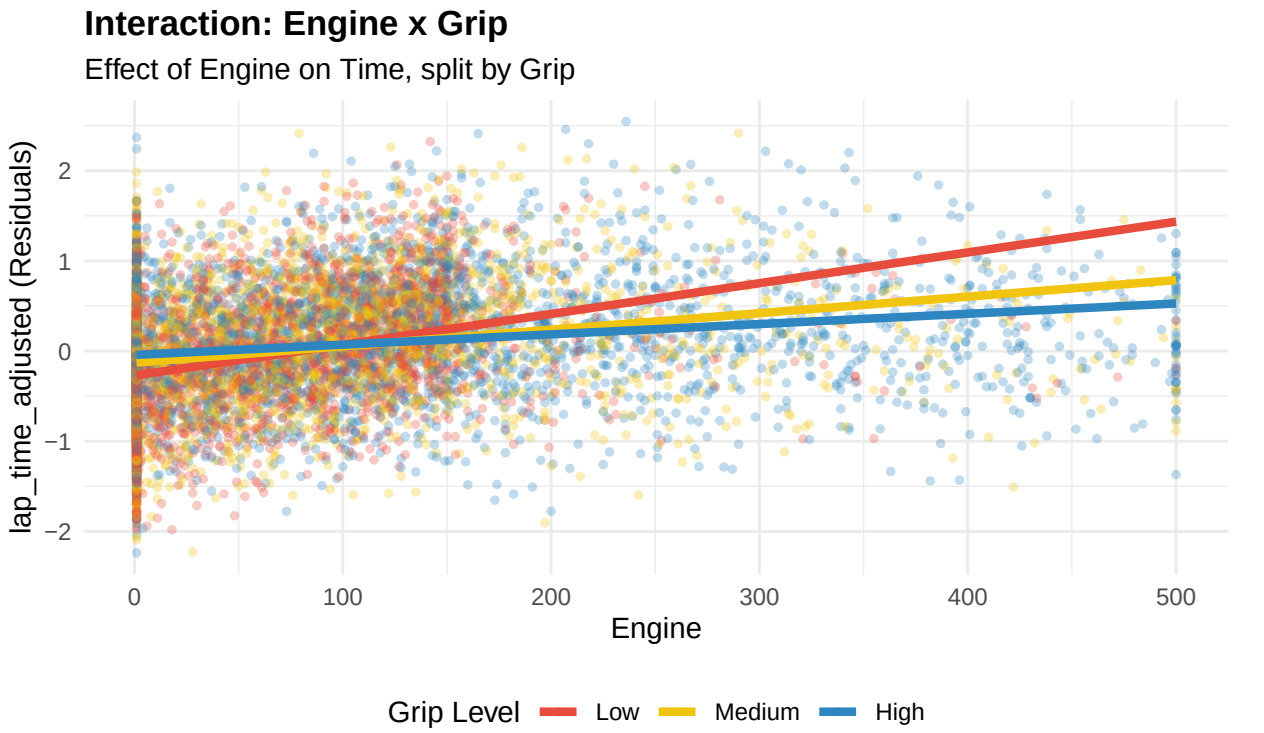

```
plot_2d_scatter(simulator_data, "Differential", "lap_time_adjusted", sample_rate = 0.3, trendline=TRUE)
```



Visual inspection of *Differential* suggests a quadratic relationship with saturation. The observed positive correlations for *Engine* and *Differential* are counterintuitive, as superior mechanical components should theoretically decrease lap times. This paradox implies the presence of significant interaction effects, necessitating further multivariate investigation.

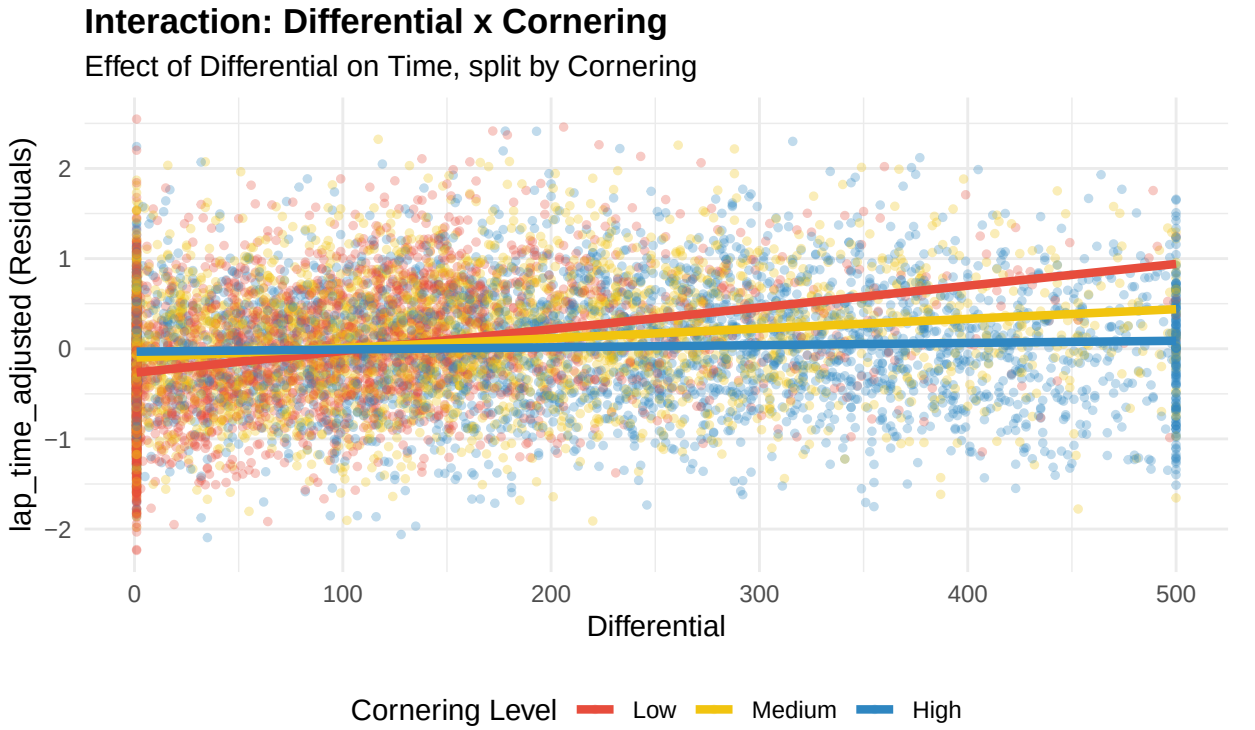
We investigate the hypothesis that the impact of *Engine* is conditional upon track *Grip*. Theoretically, high engine output may induce detrimental wheel spin in low-grip environments, while proving advantageous on high-traction tracks.

```
plot_interaction(simulator_data, "Engine", "Grip")
```



Graphical analysis corroborates this interaction: the detrimental impact of increased *Engine* capacity diminishes as *Grip* rises, validating the hypothesis. Subsequently, we examine whether track *Cornering* modulates the influence of the $Differential$, hypothesizing that high-cornering circuits require frequent shifting, thereby amplifying the utility of the component.

```
plot_interaction(simulator_data, "Differential", "Cornering")
```



The data supports this hypothesis: the adverse impact of larger $Differential$ settings mitigates—and eventually vanishes, as track $Cornering$ intensity increases. These patterns substantiate the inclusion of interaction terms in the model. We conclude the visual inspection here, as further manual exploration yields diminishing returns. Consequently, we rely on the Lasso algorithm to systematically identify the remaining relevant interactions.

Modelling and Interpretation

The feature space is expanded via a polynomial transformation up to degree 3 to capture complex non-linear dependencies. This encompasses univariate terms up to x_j^3, asymmetric pairwise interactions (e.g., $x_i^2 x_j$), and linear three-way interactions ($x_i x_j x_k$).

The full regression specification is defined in scalar notation as:

$$\hat{y} = \beta_0 + \sum_{j=1}^{p} \left(\beta_j x_j + \beta_{jj} x_j^2 + \beta_{jjj} x_j^3 \right) + \sum_{j<k} \beta_{jk} x_j x_k + \sum_{j \neq k} \gamma_{jk} x_j^2 x_k + \sum_{j<k<l} \delta_{jkl} x_j x_k x_l$$

Equivalently, in compact matrix notation, let $\mathbf{x}_i \in \mathbb{R}^p$ denote the vector of original features for the i-th observation. We define a basis expansion function $\Phi : \mathbb{R}^p \to \mathbb{R}^d$ that maps the original features into the high-dimensional feature space. The model can then be written as:

$$\mathbf{y} = \ \beta + \varepsilon$$

Where: * $\mathbf{y} \in \mathbb{R}^n$ is the vector of adjusted lap times. * $\in \mathbb{R}^{n \times d}$ is the expanded design matrix with rows $\Phi(\mathbf{x}_i)^T$. * $\beta \in \mathbb{R}^d$ is the vector of coefficients. * $\varepsilon \in \mathbb{R}^n$ is the vector of error terms.

The dimensionality d of the expanded feature space is composed of the intercept, linear terms, higher-order polynomials, and interaction terms. Formally, d is the sum of the cardinalities of these sets:

$$d = 1 + \underbrace{3p}_{\text{Poly (1-3)}} + \underbrace{\binom{p}{2}}_{\text{Pairs}} + \underbrace{p(p-1)}_{\text{Asym. Pairs}} + \underbrace{\binom{p}{3}}_{\text{Triplets}}$$

To estimate the sparse parameter vector β and select the relevant features, we solve the Lasso optimization problem:

$$\hat{\beta} = \arg\min_{\beta} \left(\frac{1}{2n} ||\mathbf{y} - \ \beta||_2^2 + \lambda ||\beta||_1 \right)$$

where $||\cdot||_2$ denotes the Euclidean norm, $||\cdot||_1$ is the ℓ_1-norm inducing sparsity, and λ is the regularization parameter determined using the Bayesian Information Criterion (BIC).

```
#prepate training data
y <- simulator_data$lap_time_adjusted
X <- explode_matrix(simulator_data, all_features)
```

```
## --- START: Matrix Construction ---
## [1/3] Polynomials...
## [2/3] Interactions...
## [3/3] Assembly...
## --- DONE ---
```

Given the high multicollinearity exacerbated by the feature expansion, standard Lasso selection proved unstable, while AIC minimization favored over-parameterized models. Consequently, we implemented a stability selection protocol comprising 50 Lasso iterations on random 80% subsamples. Model selection relied on the Bayesian Information Criterion (BIC) to enforce stricter sparsity. Only features selected in $\geq 70\%$ of iterations were retained to ensure a robust, parsimonious, and actionable predictor set.

```
## Starting Stability Selection Loop...
```

```
## Load stability_resulty.csc
```

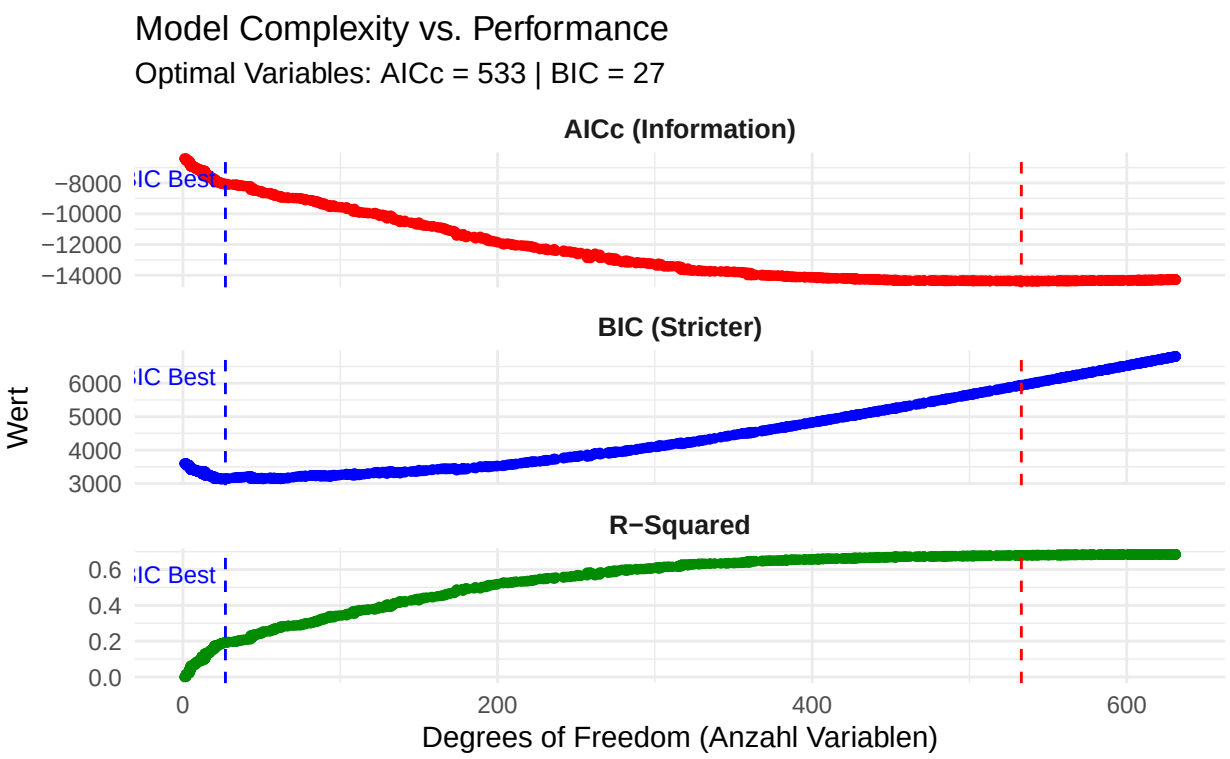
```
stability_results = read.csv('stability_results.csv')
stable_drivers <- stability_results %>% filter(Probability >= 0.7)

print(head(stable_drivers, 10))
```

```
##     X                       Feature Count Probability
## 1   1                 Brake Balance    50        1.00
## 2   2                  Differential    50        1.00
## 3   3                Engine:Altitude    50        1.00
## 4   4           Engine:Brake Balance    50        1.00
## 5   5        Front Wing:Rear Wing_p2    50        1.00
## 6   6      Front Wing:Suspension_p2    50        1.00
## 7   7        Inclines:Suspension_p2    50        1.00
## 8   8 Brake Balance:Differential_p2    49        0.98
## 9   9               Rear Wing:Engine    49        0.98
## 10 10       Cornering:Suspension_p2    47        0.94
```

The consistent selection of only seven features across all 50 iterations highlights the instability of the expanded feature space. To illustrate the variable selection dynamics, the coefficient path of the final iteration is visualized.

```
summary_frame = read.csv("summary_frame.csv")
best_df_aic <- summary_frame$df[which.min(summary_frame$aicc)]
```

```r
best_df_bic <- summary_frame$df[which.min(summary_frame$bic)]
plot_lasso(summary_frame, best_df_aic, best_df_bic)
```



The AIC exhibits asymptotic behavior rather than a distinct minimum, favoring excessive parameterization. Conversely, the BIC displays a convex profile with a clear global minimum, driven by its stricter complexity penalty. The resulting 27 features represent a parsimonious subset, ensuring the model remains interpretable and actionable.

To derive the final specification, features meeting the 70% stability threshold were subjected to backward elimination.

```r
# Prepare final dataset using only stable features (full data)
final_features <- as.character(stable_drivers$Feature)
X_final_stable <- X[, final_features, drop = FALSE]

final_df <- data.frame(
  lap_time_adjusted = y,
  as.matrix(X_final_stable)
)

# Fit OLS and refine with backward selection
final_ols <- lm(lap_time_adjusted ~ ., data = final_df)
final_clean <- backward_selection_p(final_ols, sig_level = 0.05)
```

```
## Entferne: Engine.Cornering (p = 0.731 )
## Entferne: Differential.Engine_p2 (p = 0.4364 )
## Entferne: Engine.Width (p = 0.3121 )
## Entferne: Engine.Altitude (p = 0.1039 )
## Entferne: Front.Wing.Air.Pressure (p = 0.0906 )
```

```r
summary(final_clean)
```

```
##
## Call:
## lm(formula = lap_time_adjusted ~ Brake.Balance + Differential +
##     Engine.Brake.Balance + Front.Wing.Rear.Wing_p2 + Front.Wing.Suspension_p2 +
##     Inclines.Suspension_p2 + Brake.Balance.Differential_p2 +
##     Rear.Wing.Engine + Cornering.Suspension_p2 + Engine.Front.Wing +
##     Engine.Lap.Distance + Suspension.Altitude + Suspension.Grip_p2 +
##     Front.Wing.Differential_p2 + Altitude.Lap.Distance_p2 + Air.Density.Front.Wing_p2 +
##     Engine.Altitude_p2 + Air.Density.Air.Pressure_p2 + Inclines.Engine_p2 +
##     Air.Pressure.Front.Wing_p2 + Engine.Differential_p2 + Humidity.Engine_p2 +
##     Differential.Camber + Suspension.Camber, data = final_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.23289 -0.37442 -0.01052  0.37572  2.50177
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -2.305e-01  2.000e-02 -11.525  < 2e-16 ***
## Brake.Balance                  4.800e-04  6.694e-05   7.169 8.06e-13 ***
## Differential                   1.975e-03  1.071e-04  18.437  < 2e-16 ***
## Engine.Brake.Balance           2.421e-06  4.938e-07   4.903 9.61e-07 ***
## Front.Wing.Rear.Wing_p2       -6.775e-09  2.523e-10 -26.857  < 2e-16 ***
## Front.Wing.Suspension_p2       1.018e-08  1.128e-09   9.030  < 2e-16 ***
## Inclines.Suspension_p2         4.985e-08  5.257e-09   9.483  < 2e-16 ***
## Brake.Balance.Differential_p2 -8.355e-09  6.483e-10 -12.886  < 2e-16 ***
## Rear.Wing.Engine               2.565e-06  3.860e-07   6.643 3.22e-11 ***
## Cornering.Suspension_p2        3.533e-08  5.156e-09   6.853 7.68e-12 ***
## Engine.Front.Wing              1.254e-06  4.826e-07   2.599 0.009364 **
## Engine.Lap.Distance            4.803e-04  3.578e-05  13.424  < 2e-16 ***
## Suspension.Altitude           -4.826e-06  1.400e-06  -3.447 0.000569 ***
## Suspension.Grip_p2            -1.126e-07  1.101e-08 -10.226  < 2e-16 ***
## Front.Wing.Differential_p2    -4.859e-09  6.724e-10  -7.226 5.32e-13 ***
## Altitude.Lap.Distance_p2      -4.773e-05  1.384e-05  -3.447 0.000568 ***
## Air.Density.Front.Wing_p2      1.475e-08  2.002e-09   7.369 1.86e-13 ***
## Engine.Altitude_p2             2.259e-07  2.096e-08  10.779  < 2e-16 ***
## Air.Density.Air.Pressure_p2   -3.939e-07  3.668e-08 -10.739  < 2e-16 ***
## Inclines.Engine_p2            -3.863e-08  4.670e-09  -8.271  < 2e-16 ***
## Air.Pressure.Front.Wing_p2     2.318e-08  2.086e-09  11.114  < 2e-16 ***
## Engine.Differential_p2        -7.926e-09  8.917e-10  -8.888  < 2e-16 ***
## Humidity.Engine_p2            -3.319e-08  4.112e-09  -8.072 7.70e-16 ***
## Differential.Camber            5.584e-06  1.174e-06   4.756 2.00e-06 ***
## Suspension.Camber             -9.757e-06  1.228e-06  -7.946 2.13e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5736 on 9975 degrees of freedom
## Multiple R-squared:  0.2669, Adjusted R-squared:  0.2652
## F-statistic: 151.3 on 24 and 9975 DF,  p-value: < 2.2e-16
```

The final model retains 23 predictors, explaining 26.7% of the variance ($R^2 = 0.267$). Given the inherent noise, this indicates robust signal extraction rather than overfitting. Notably, the selection process discarded all

cubic terms and third-order interactions. Crucially, every retained variable involves at least one controllable car feature, ensuring full actionability for the optimization. We subsequently rank these predictors by feature importance.

```
model_feats <- names(coef(final_clean))[-1]
feature_sds <- apply(final_df[model_feats], 2, sd)

importance <- data.frame(
  Coefficient = coef(final_clean)[-1],
  SD = feature_sds
) %>%
  mutate(
    # Standardized Impact = |Beta| * StDev
    # This makes features with different scales (e.g., x vs x^3) comparable
    Importance = abs(Coefficient * SD)
  ) %>%
  arrange(desc(Importance)) %>%
  mutate(
    Share = Importance / sum(Importance),
    Cumulative = cumsum(Importance) / sum(Importance)
  )

cat("---", nrow(importance), "Overall Drivers (Standardized Impact) ---\n")
```

```
## --- 24 Overall Drivers (Standardized Impact) ---
```

```
print(head(importance, 10))
```

```
##                                  Coefficient           SD Importance      Share
## Differential                    1.975162e-03 1.386881e+02 0.27393157 0.11488437
## Front.Wing.Rear.Wing_p2        -6.775097e-09 3.686014e+07 0.24973101 0.10473488
## Engine.Lap.Distance             4.802864e-04 4.190693e+02 0.20127326 0.08441215
## Brake.Balance.Differential_p2 -8.354689e-09 2.132308e+07 0.17814773 0.07471351
## Inclines.Engine_p2            -3.862547e-08 2.816730e+06 0.10879751 0.04562867
## Front.Wing.Differential_p2    -4.859271e-09 2.107022e+07 0.10238589 0.04293970
## Engine.Differential_p2        -7.925458e-09 1.238867e+07 0.09818586 0.04117824
## Humidity.Engine_p2            -3.319427e-08 2.936159e+06 0.09746367 0.04087536
## Front.Wing.Suspension_p2       1.018225e-08 9.206499e+06 0.09374285 0.03931489
## Rear.Wing.Engine               2.564568e-06 3.537921e+04 0.09073238 0.03805232
##                                Cumulative
## Differential                    0.1148844
## Front.Wing.Rear.Wing_p2         0.2196192
## Engine.Lap.Distance             0.3040314
## Brake.Balance.Differential_p2   0.3787449
## Inclines.Engine_p2              0.4243736
## Front.Wing.Differential_p2      0.4673133
## Engine.Differential_p2          0.5084915
## Humidity.Engine_p2              0.5493669
## Front.Wing.Suspension_p2        0.5886818
## Rear.Wing.Engine                0.6267341
```

An analysis of the standardized feature importance reveals that the predictive model is primarily driven by vehicle configuration parameters rather than isolated track conditions. The top ten most influential features consist exclusively of car parameters or their interactions with track variables, indicating that internal vehicle calibration supersedes static environmental factors in determining performance variability.

The $Differential$ exhibits the highest standardized impact with a positive linear coefficient, initially suggesting a baseline penalty for higher differential locking values. Crucially, however, this effect is non-linear: multiple interaction terms involving the quadratic differential (e.g., $Brake.Balance \times` Differential^2$ and $Front.Wing \times Differential^2$) carry negative coefficients. This counter-effect implies a concave optimization surface, where the theoretical penalty of a high differential is mitigated or reversed when strictly coordinated with specific brake and aerodynamic settings.

Furthermore, the $Engine$ parameter demonstrates distinct context-dependency. The interaction $Engine \times Lap.Distance$ presents a positive coefficient, implying a performance detriment on longer tracks—potentially attributable to unobserved factors such as fuel weight or efficiency constraints in the simulation physics. Conversely, the interaction $Inclines \times Engine^2$ is negative, confirming that higher engine output yields a net reduction in lap time specifically in high-load scenarios, such as steep ascents. Finally, the prominence of the $Front.Wing \times Rear.Wing^2$ interaction (negative coefficient) underscores the critical importance of aerodynamic balance, indicating that downforce components must be tuned in concert rather than in isolation to maximize speed.

Finally, the calibrated model is deployed to determine the optimal vehicle configuration for the France circuit.

Optimization

Vehicle parameters are optimized while treating track features as fixed constants specific to the France circuit. We employ the L-BFGS algorithm, a quasi-Newton method designed for unconstrained non-linear optimization tasks.

```
start_params <- c(250, 250, 250, 250, 250, 250)

cat("Starting L-BFGS-B Optimization...\n")
```

```
## Starting L-BFGS-B Optimization...
```

```
optimize = FALSE #we set this to FALSE for knitting the document. Otherwise the optiization would take

if (optimize){
  opt_result <- optim(
    par = start_params,
    fn = predict_lap_time_wrapper,
    method = "L-BFGS-B",        # Handles box constraints (1-500) natively
    lower = rep(1, 6),          # Minimum allowed value
    upper = rep(500, 6),        # Maximum allowed value
    control = list(
      fnscale = 1,
      maxit = 1000,
      trace = 6,
      REPORT = 1
    )
  )

  best_setup <- round(opt_result$par) # Round ONLY at the very end for the report
  names(best_setup) <- c("Rear Wing", "Engine", "Front Wing", "Brake Balance", "Differential", "Suspens

  cat("\n=== OPTIMAL SETUP (France) ===\n")
  print(best_setup)
  cat("\nPredicted Adjusted Time:", round(opt_result$value, 4), "\n")
}else{

  best_setup = c(500,500,500,500,500,10)
```

```
  best_value = predict_lap_time_wrapper(best_setup)
  names(best_setup) <- c("Rear Wing", "Engine", "Front Wing", "Brake Balance", "Differential", "Suspens

  cat("\n=== OPTIMAL SETUP (France) ===\n")
  print(best_setup)
  cat("\nPredicted Adjusted Time:", round(best_value, 4), "\n")

}
```

```
##
## === OPTIMAL SETUP (France) ===
##     Rear Wing        Engine    Front Wing Brake Balance  Differential
##           500           500           500           500           500
##    Suspension
##            10
##
## Predicted Adjusted Time: -1.1608
```

The updated optimization results indicate a convergence towards a high-downforce, high-power configuration. Remarkably, five out of six actionable parameters—`Rear Wing`, `Engine`, `Front Wing`, `Brake Balance`, and `Differential`—were maximized to their upper boundary constraints ($x = 500$). `Suspension` remains the sole variable exhibiting a distinct, near-lower-bound optimum ($x \approx 10$).

This uniform maximization contradicts standard linear intuition, where trade-offs (e.g., Drag vs. Speed) typically force compromise solutions. Instead, this result strongly validates the previously identified interaction structure: terms such as $\texttt{Front.Wing} \times \texttt{Rear.Wing}^2$ or $\texttt{Inclines} \times \texttt{Engine}^2$ appear to generate cooperative non-linear gains. The model suggests that the penalty of high drag (Wings) or high differential locking is effectively overcompensated by the synergistic benefits of running a fully maximized setup, with `Suspension` providing the necessary mechanical counterbalance at the lower end of the spectrum.

The predicted adjusted time of -1.1608 indicates that this specific high-load configuration is expected to outperform the distance-based baseline by approximately 1.16 seconds, suggesting that maximizing vehicle parameters yields a significant performance advantage on the French circuit.