MMM Budget - Probabilistic Frameworks

Giovanni P.

August 2024

1 Introduction

This document presents a set of different recipes that could be used to analyse the probability distribution of Predicted Optimal Revenues.

- Predicted Optimal Revenue: the predicted revenue after a budget optimization.
- Realized Optimal Revenue: the actual revenue obtained by the client using the optimized budget.

2 Statistical approach

A very basic, but nonetheless useful, approach would be to collect a dataset of Predicted Optimal Revenue alongside the Realized Optimal Revenue, for each week in which the optimization is performed. This would be different from the historical data of the non-optimized predicted revenues.

This dataset could merge data points coming from multiple client, and it would be the most direct measure of the distribution of Realized Optimal Revenue vs Predicted Optimal Revenue. This dataset could be enhanced with all other dataset variables to allow for segmentation of the analysis (i.e. analyse only predicted revenues where $GoogleCost > X$, or analyse only predicted revenues for $Client = Y$).

3 Optimal Parameter Sets and Bootsrapping

Another approach is analysing the distribution of the optimal parameters ($c$oeff, alpha, gamma, shape, scale), and how changes in the parameters affect the distribution of optimized revenue.
A set of optimal parameters in the marketing mix model can be defined as:

$$\Theta^* = \{Coeff, alpha, gamma, shape, scale\}$$

Creating a distribution of the optimal parameters can be achieved in different ways:

- **Distribution of Parameters from Models Near the Optimal Model**: Selecting models that perform close to the optimal model (picking from the entire set of Robyn's optimized model), it is possible to analyze the range of parameter values that still yield near-optimal results.
- **Distribution of Optimal Models Parameters via Bootstrapping**: Applying bootstrapping to the dataset generates multiple variations, each slightly different due to resampling (For example, select 80% of the datapoints starting by the first available data point, run optimization and select the optimal model, then move the 80% window forward by 1 week, and repeat the optimization process, continue repeating until when no samples are available). Running the model on these bootstrapped datasets produces a distribution of optimal models parameters.
- **Introducing Noise on Parameters Based on Assumed Variance**: Introducing noise to model parameters, guided by assumptions about their variance, simulates uncertainty. This approach evaluates the impact of small perturbations in the parameters on the overall optimized revenue, helping to identify which parameters are critical and which can vary without significantly affecting results.

All above methods lead to the generation of multiple optimal parameter sets:

$$\{\Theta_1^*, \Theta_2^*, \ldots, \Theta_n^*\}$$

These sets of parameters can then be used to generate a distribution of Optimal Budgets and Optimal Predicted Revenues.

4 Estimating Confidence of Revenues

After Adstock and Saturation are applied, the relationship between predicted revenue by channel and saturation is linear, with a coefficient selected by the model (for each channel), $P = C \cdot S$, where C is the coefficient from Robyn selected model.
We focus all subsequent analysis on the relationship between S and P.

To estimate the confidence of a certain revenue prediction, we opt for a density based method, specifically using a Gaussian Kernel Density Estimation. The idea is that saturation regions with higher amount of samples are gonna have more density (and confidence), while regions less populated are gonna have lower density (and lower confidence).

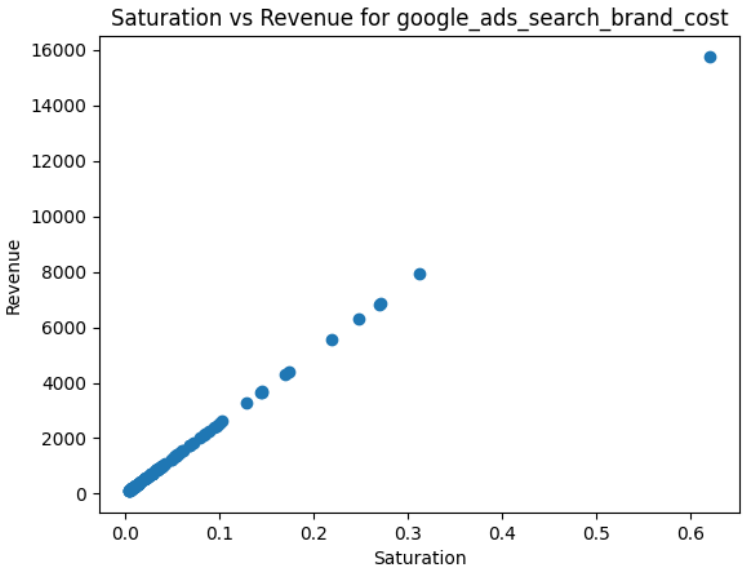

Figure 1: Saturation values vs Revenue values

For each channel, the steps are:

- Starting from Saturated Values S, calculate the local density $D(i)$ of each sample using Gaussian Kernel Density Estimation (KDE). *See scipy.stats gaussian_kde.*
- Calculate Confidence F by applying a Min Max normalization to the density. $F(i) = (D(i) - D_{min})/(D_{max} - D_{min})$
- Calculate upper bounds and lower bounds of the Revenue by applying a margin inversely proportional to the confidence.
  $R_{upper}(i) = R(i)(1 + M(1 - F(i)))$
  $R_{lower}(i) = R(i)(1 - M(1 - F(i)))$

Note that the margin factor M is the only parameter of this formulation (and it can adjusted to give more or less weight to regions of low confidence).

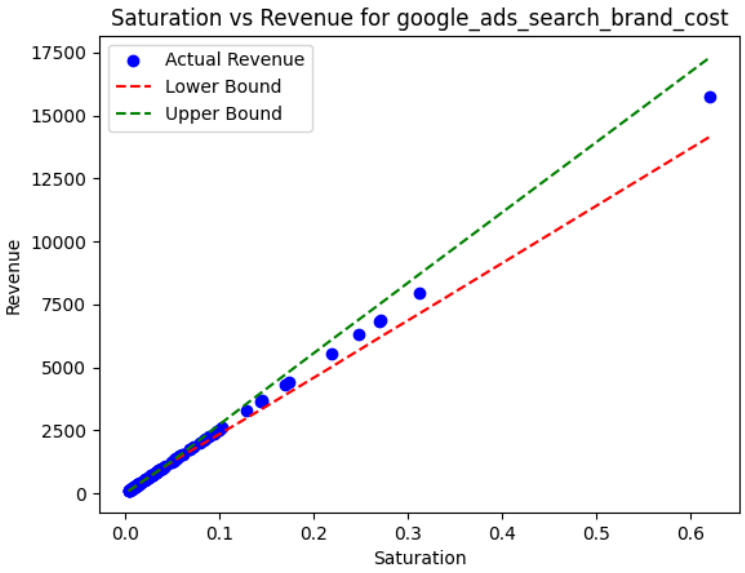

Figure 2: Upper and Lower confidence bounds on the saturation

This method doesn't make assumption on which regions have higher or lower confidence, if a middle-cost region has fewer samples, the density estimation will correctly detect it.

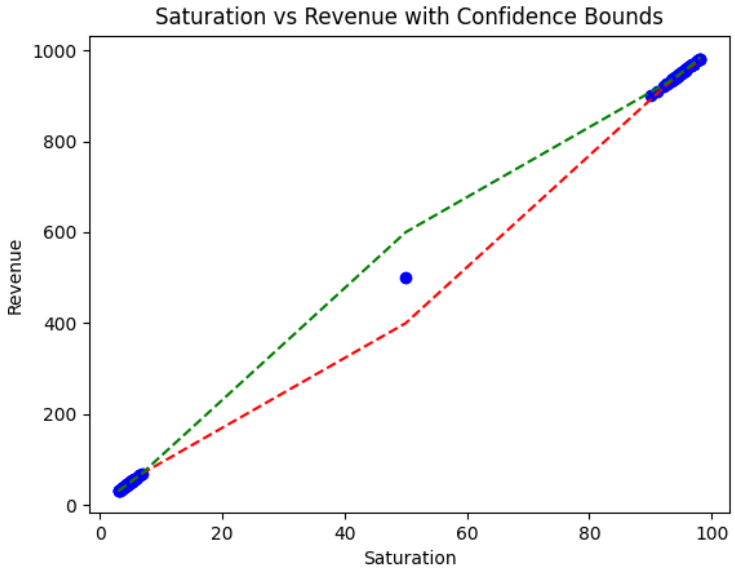

Figure 3: Example with an artificial dataset, low amount of samples and low confidence for middle cost region

5 Estimating Revenues Distribution

After computing the Confidence Intervals and the Revenue Upper and Lower Bound for each Channel, we can implement an algorithm to find the probability distribution of the predicted revenue.

The algorithm is:

- Run the budget optimizer and find the optimal budget. Recompute Adstock, Saturation and Decomp dataset with the new budget rows.
- Starting from the first newly added row, and for each media channel:
  - Given the saturation of new row j, $S(j)$, find the closest Saturation values in the historical dataset.
  - find the confidence of j, $F(j)$ by performing a linear interpolation of the closest historical saturation's confidences. If $S(j)$ is outside of the know regions, pick the closest historical sample's confidence.
  - Apply the confidence $F(j)$ to the predicted revenue for that channel, generating in this way two predicted revenues bounds, $P(j)_{upper}$ and $P(j)_{lower}$.

- By repeating this process for each channel we obtain an Upper Bound Decomp ($Decomp_{upper)}$), and a Lower Bound Decomp ($Decomp_{lower)}$).
- Sum the columns of these dataframes to generate Upper and Lower bounds on the paid media variables Total Revenue.

```
Confidence for criteo_cost: 0.084
Confidence for facebook_ads_brand_awareness_cost: 0.909
Confidence for facebook_ads_conversions_cost: 0.929
Confidence for facebook_ads_performance_engagement_cost: 0.997
Confidence for google_ads_display_only_cost: 0.995
Confidence for google_ads_search_brand_cost: 0.071
Confidence for google_ads_search_kids_cost: 0.754
Confidence for google_ads_search_men_cost: 0.582
Confidence for google_ads_search_others_cost: 0.654
Confidence for google_ads_search_women_cost: 0.311
Confidence for google_ads_shopping_cost: 0.199
Confidence for google_ads_video_cost: 0.966
```

Figure 4: Confidences of an optimized budget

Finally, by assuming that the bounds will be 95% accurate (two standard deviations), we can estimate the revenue probability by using a Gaussian distribution.

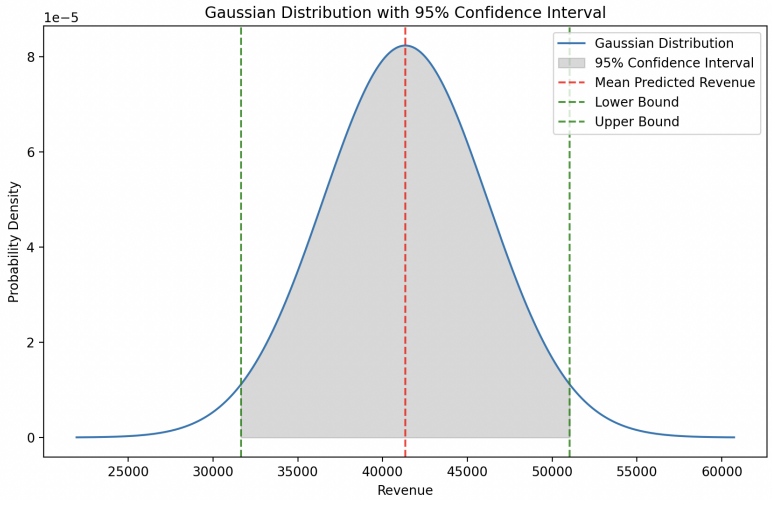

Figure 5: Probability distribution of the predicted revenue, M=0.5

5.1 Note on Margin factor M

The margin factor M allows to model elegantly the uncertainty regions.

If $M = 0$ the model collapse back to a deterministic one (confidence is always 100%). If M=0.5, when a new budget sample is in a low confidence region, there is a 95% chance that that for that channel the revenue is between 50% and 150% of the predicted revenue. If M=1, when a new budget sample is in a low confidence region, there is a 95% chance that that the revenue for that channel is between 0% and 200% of the predicted revenue. M can assume any value between 0 and 1, values higher than 1 would lead to negative contribution to the revenue.

Note that with a dataset of Predicted Optimal Revenue and Real Optimal Revenue it should be possible to calibrate M in an automated way.

For the sake of the exercise we set M=0.5 because in the historical dataset approximately 95% of the values of DepVarHat are between +/- 50% of DepVar (this is computed manually on the historical dataset).

6 Mentions to other approaches

Other approaches could be based on switching the ridge regression of Robyn to some other regression models that also produce confidence interval as an output or models that account for variance of the input dataset. However, none of these models would account for the density of the sample on their own, and they would all require some custom density/confidence calculation implemented on top of them. Just as a mention, these models could be:

- Weighted Least Squares (WLS): Adjusts the influence of each data point in the regression based on known or estimated variances. Higher confidence values receive more weight, making it ideal when you can quantify the reliability of each observation.
- Generalized Least Squares (GLS): Models the error variance as a function of the predictors. It directly accounts for situations where the variability of the response changes systematically with the level of the predictor variables.
- Heteroscedasticity-Robust Standard Errors: Provides valid standard errors and confidence intervals by adjusting for heteroscedasticity (non-constant variance) in the data. It doesn't model the variance explicitly but corrects for its presence.
- Bayesian Regression: Incorporates prior beliefs about the data, allowing you to model and account for varying levels of uncertainty across the

dataset. It's flexible and can adapt to different levels of confidence in the data.

(Summaries of these models generated with chatGPT)

7 Final Thoughts

- Implement the data collection described in "Statistical Approach" (if not already implemented.) Aggregating data from multiple clients could be cumbersome as they will have different independent variables, nonetheless this type of type of dataset would be extremely useful.
- Implementing Bootstrapping methods is an optional, it allows to estimate the probability of the prediction but this analysis is based only on the model parameters and on their robustness.
- The density based bounds allows to model the confidence of each individual channel, the confidence is based on real historical data, and the confidence measure itself can be used to generate alerts when samples of a new row are in a 'low confidence region'.

8 Code references

- An example of 'artificial' bootstrapping is implemented in "3_Bootstrapping_example.py", in this case by just adding noise to the parameters (a proper implementation would be based on multiple Robyn models, or on multiple Robyn optimizations)
- Code for confidence intervals by channel is in "4_finding_confidence_intervals.py".
- Code for the revenue probability distribution is in "5_revenue_probability_distribution.py".

Note that the revenue bounds are computed only on the paid media channels, and the code is focused only on the first optimal budget row (but it can be extended for all following rows in the same manner).