**Early Analysis Report**

The first thing to note is the densities and descriptive statistics of the training R2 that the algorithms produce. The ‘proportion impossible values’ that are seen in the figure are when R2 < 0 or > 1.

*Figure 1.* Densities and Descriptive statistics for the algorithms’ training R2

A picture containing chart

Description automatically generated

Graphical user interface, chart

Description automatically generated

The densities and descriptive statistics for the testing R2 look almost identical to those of the training R2 and are therefore not reported. The densities already show some of the problems with the horseshoe and lasso algorithms, but let the following descriptive table make it more clear.

Table 1. *Descriptives of algorithms on train\_r2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Mean** | **Sd** | **min** | **max** | **Median** |
| **Horseshoe** | -1.88 | 5.54 | -177 | 0.98 | -0.04 |
| **Lasso** | -1.92 | 5.80 | -197 | 0.98 | -0.02 |
| **Metaforest** | 0.80 | 0.11 | -0.01 | 0.99 | 0.82 |
| **RMA** | 0.52 | 0.30 | -0.33 | 0.99 | 0.62 |

Interesting to note that none of the algorithms will exceed R2 = 1. Metaforest and RMA rarely produce impossible values , while horseshoe and lasso go well below 0 around 60% of the times each. It could be that there are a lot of values that are just barely below 0, that have a huge impact on the proportion of impossible values even though they are not so problematic. However, if we use a cut-off value of

-0.33, based on the lowest value of the RMA algorithm, so that R2 [-0.33 , 1] we still find ~33% of impossible values for both Pema algorithms. This number reduces down to ~0.24% for both algorithms when R [-1 , 1].

Due to some severe outliers, the median is a better estimate for an algorithms calculated R2 than the mean, but note the medians for both Pema algorithms are below 0, indicating that at least 50% of the values are smaller than 0.

I identified on which conditions/design factors R2 showed impossible values for different lower limit cut-off values. I used different lower limits to determine if there are patterns in the data. In other words, are the proportion of faulty R2 per condition the same if we use different lower limits? The lower bounds I ran were c(0, -0.33, -1, -10, -20) which are quite arbitrary, but do seem to provide useful insights to see what problems arise as we set stricter lower bounds. It is interesting to note that the cases for which R2 < lowerboundi are almost exactly the same for both Lasso and Horseshoe. Ranging from 95%-99.8% overlap, increasing with every lowerboundi.

To see what conditions contribute to faulty[[1]](#footnote-1) R2 values, I made a few tables showing the number of faulty iterations per conditions and for every lowerboundi. I do this only for 1 algorithm and only for the testing R2, considering the algorithms overlap at least 95% for every lowerboundi, and both the testing and training R2 are faulty during similar conditions. Still, I created two separate lists with all the values for both the training and testing R2 which can be loaded in using loadRDS() if desired. The tables do not consider interaction effects, but even without interactions, clear patterns can be observed. As said before, the tables contain estimates for the testing R2, but the same pattern is observed for the training R2. The tables of the conditions that seem to have an effect on faulty R2 values are shown in the appendix. In summary, a few things can be noted:

1.) **Effect size**. The table clearly shows that as the true effect size increases, the number of faulty R2 values increases. It thus seem like the Pema algorithms have difficulty estimating R2 when the true effect size increases.

2.) **Alpha mod.** The table also shows that as the skewness of the data increases, the number of faulty R2 increases. the Pema algorithms have difficulty estimating R2 when the skewness of the data increase.

3.) **Number of moderators**. The table shows something especially interesting. The Pema algorithms estimate R2 best when number of moderators is 4 or 7, while for the other levels (2,3,6) they have more difficulty.

4.) **Model type**. The table shows the PEMA models perform worse when cubic relationships have to be estimated, while it also has difficulty with the exponential relationships. It seems to perform best under simple main effect models and the models with two-way interactions.

5.) **Mean n**. I did not include a table for this condition, but the algorithms performed slightly better when mean n was smaller. This indicates the lower the mean sample size per study, the better the prediction.

In summary, Pema performs best when the true effect size is small, the data is close to normally distributed, the number of moderators is either 4 or 7, the model only uses main effects and interactions and when the mean n is small. For k\_train and tau2 there seems to be no considerable effect. I also performed a logistic regression where the outcome was being either a faulty or a non-faulty R2 value. I set the boundary to -0.33 and the results roughly supported the findings above. No output is provided because it provides little new information.

**Traceplots**

To see where in the dataset most problems occur, a traceplots for each algorithms’ test and training R2 were plotted. The images can be found under the tables in the appendix. What is interesting to note is that the Metaforest algorithm seems to perform equally well over all conditions and combinations of conditions. This is evident as the R2 values seem to not diverge too much over the iterations. The RMA algorithm seems to have the hardest time at the end of every 100.000 iterations. This can be derived from the traceplot as we see that the R2 values become more implausible every 100.000 iterations or so. Subsetting the data for cases 80.000-100.000, 180.000-200.000, 280.000-300.000 and 350.000-388.000 show during what conditions the algorithm has more difficulty calculating R2. It is obvious that the RMA algorithm has most problems with the exponential model as 55% of the subsetted cases contain the condition where the algorithm estimates the exponential model.

As for the Lasso and Horseshoe algorithms, we see something similar. We see that most problems arise between cases 200.000-300.000. Subsetting the data for this range reveals the problematic conditions. The exponential model as well as the cubic models are estimated in this range by the algorithms. However, the exponential model is included in only 8410 iterations, whereas the cubic model is included in 91591 iterations. Interestingly, this range only contains 2,3 or 6 moderators. No problems occur when 4 or 7 moderators are included in the model. It seems that the Pema algorithms have most difficulty estimating the cubic models. Likely due to very large estimates and thus very large variances. The reason why it only happens when 2, 3 or 6 moderators are included remains a mystery however.

**Appendix**

Tables. *Number of faulty iterations per condition for different lowerbounds (Testing R2)*

***For effect size***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lowerbound** | **Condition value** | **Numb. faulty R2** |
| **True effect size** | *0* | 0 | 53524 |
|  |  | 0.2 | 24717 |
|  |  | 0.5 | 56730 |
|  |  | 0.8 | 73411 |
|  | *-0.33* | 0 | 1192 |
|  |  | 0.2 | 10429 |
|  |  | 0.5 | 43477 |
|  |  | 0.8 | 64279 |
|  | *-1* | 0 | 54 |
|  |  | 0.2 | 3528 |
|  |  | 0.5 | 30772 |
|  |  | 0.8 | 54086 |
|  | *-10* | 0 | 0 |
|  |  | 0.2 | 4 |
|  |  | 0.5 | 3506 |
|  |  | 0.8 | 17259 |
|  | *-20* | 0 | 0 |
|  |  | 0.2 | 0 |
|  |  | 0.5 | 362 |
|  |  | 0.8 | 7820 |

***For Alpha\_mod***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lowerbound** | **Condition value** | **Numb. faulty R2** |
| **Alpha mod** | *0* | 0 | 48304 |
|  |  | 2 | 77664 |
|  |  | 10 | 82414 |
|  | *-0.33* | 0 | 16889 |
|  |  | 2 | 47746 |
|  |  | 10 | 54742 |
|  | *-1* | 0 | 9424 |
|  |  | 2 | 35893 |
|  |  | 10 | 43123 |
|  | *-10* | 0 | 221 |
|  |  | 2 | 8048 |
|  |  | 10 | 12500 |
|  | *-20* | 0 | 27 |
|  |  | 2 | 2730 |
|  |  | 10 | 5425 |

***For number of moderators***

|  |  |  |  |
| --- | --- | --- | --- |
| **Condition** | **Lowerbound** | **Condition value** | **Numb. faulty R2** |
| **Moderators** | *0* | 2 | 54440 |
|  |  | 3 | 69545 |
|  |  | 4 | **14932** |
|  |  | 6 | 54624 |
|  |  | 7 | **14841** |
|  | *-0.33* | 2 | 31964 |
|  |  | 3 | 40060 |
|  |  | 4 | **8422** |
|  |  | 6 | 30817 |
|  |  | 7 | **8114** |
|  |  | 2 | 23980 |
|  | *-1* | 3 | 29762 |
|  |  | 4 | **5735** |
|  |  | 6 | 23431 |
|  |  | 7 | **5532** |
|  |  | 2 | 6959 |
|  | *-10* | 3 | 6956 |
|  |  | 4 | **6** |
|  |  | 6 | 6845 |
|  |  | 7 | **3** |
|  |  | 2 | 2743 |
|  | *-20* | 3 | 2766 |
|  |  | 4 | **0** |
|  |  | 6 | 2673 |
|  |  | 7 | **0** |

***For Model type***

|  |  |  |  |
| --- | --- | --- | --- |
| **Condition** | **Lowerbound** | **Condition value** | **Numb. faulty R2** |
| **Model type** | *0* | 1\* | 58577 |
|  |  | 2\* | **39316** |
|  |  | 3\* | 65806 |
|  |  | 4\* | **44683** |
|  | *-0.33* | 1 | 37330 |
|  |  | 2 | **9178** |
|  |  | 3 | 47905 |
|  |  | 4 | **24964** |
|  | *-1* | 1 | 28913 |
|  |  | 2 | **1294** |
|  |  | 3 | 41121 |
|  |  | 4 | **17112** |
|  | *-10* | 1 | 3749 |
|  |  | 2 | **0** |
|  |  | 3 | 17004 |
|  |  | 4 | **16** |
|  | *-20* | 1 | 482 |
|  |  | 2 | **0** |
|  |  | 3 | 7700 |
|  |  | 4 | **0** |

*\*1 = es \* exp(x[, 1])*

*\*2 = es \* x[, 1]*

*\*3 = es \* x[, 1] + es \* (x[, 1] ^ 2) + es \* (x[, 1] ^ 3)*

*\*4 = es \* x[, 1] + es \* x[, 2] + es \* (x[, 1] \* x[, 2])*

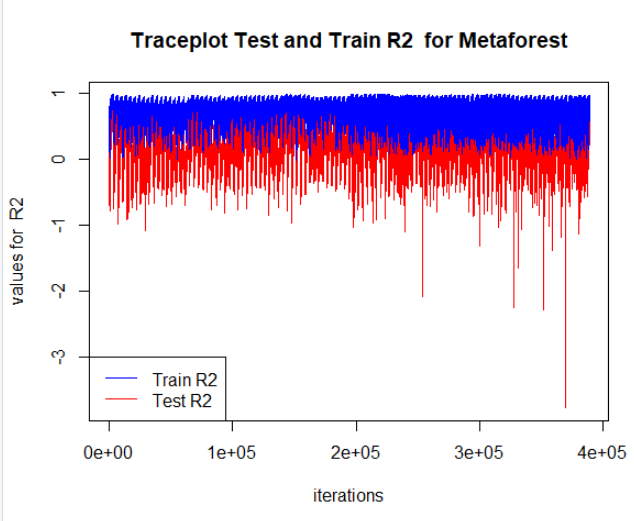
Images. *Training and testing R2 traceplots per algorithm*

***RMA***

***Chart

Description automatically generated***

***Metaforest***

******

***Lasso***

***Chart, bar chart

Description automatically generated***

***Horseshoe***

***Chart, bar chart

Description automatically generated***

1. Faulty = when R2 drops below a particular limit [↑](#footnote-ref-1)