Exploring Loop and Parameters Influence using Sensitivity Analysis with Explanatory Model Analysis

Jim Duggan[1], Enda Howley[1], Martina Curran[2], and Hamda Ajmal[3]

Abstract

In our paper, we show how sensitivity analysis can be used to identify influential model parameters using statistical learning methods. Our approach generates a sensitivity dataset (N=50,000) from a system dynamics model, and explores two aspects in order to provide model insights: (1) the influence of continuous valued parameters and (2) the influence of loop activation/deactivation flags. The analysis methods used are part of the overall method known as explanatory model analysis. To demonstrate this approach, we use a system dynamics model to support pandemic planning, and this includes feedback loops for public health interventions (vaccination, behaviour changes, government restrictions). The sensitivity data is used to train a supervision learning regression model (random forest), and the initial results from analysing the model show that the insights generated can provide useful information for the overall model building process.

[1] School of Computer Science, University of Galway, Ireland.
[2] Technological University of the Shannon, Athlone, Ireland.
[3] Central Statistics Office, Dublin, Ireland

1. Introduction

Sensitivity analysis (SA) characterises the response of model outputs to parameter variation, and can be used to elucidate the plausible range of system outcomes (Wu, Dhingra, Gambhir, & Remais, 2013). SA allows you to test the robustness of your interpretations to uncertainty in your assumptions (Sterman, 2000), and can be viewed as an integral part of model testing in system dynamics, for example, testing the structure validity of a model through behaviour sensitivity tests (Barlas, 1996).

SA has been used in system dynamics to search for the most important of uncertain parameters, for example, Ford and Flynn (2005) introduce the statistical screening process, which provides a method to identity key inputs through the use of the correlation coefficient (based on data from a set of sensitivity results), and this is extended into a repeatable six-step process by Taylor, Ford, and Ford (2010). A further application of SA to system dynamics is the work of Hekimoğlu and Barlas (2016), who develop a regression-based procedure for pattern sensitivity analysis, which estimates the changes in pattern measures such as equilibrium level and inflection point resulting from changes in model parameters.

While these methods are grounded in statistical approaches, there have also been studies in system dynamics that utilise machine learning methods. For example, Pruyt and Islam (2015) combine machine learning techniques to characterise the dynamic complexity of simulation runs and provide policy insight. Edali (2022) combines metamodeling and rule extraction to analyse SD models, based on the generation of input-parameter combinations (using methods such as Latin Hypercube sampling and Sobol sequences). These combinations and corresponding outputs (numerical or categorical) then form the metamodel training set (which is a similar process to what we propose).

Edali and Yücel (2019), in an earlier paper that utilises an agent-based stochastic model, also demonstrate the utility of using the metamodel approach to capture the boundaries of tipping point behaviours, and thereby supporting verification, validation and understanding of agent-based models.

Our approach contributes to this overall body of work by proposing a novel method to analyse system dynamics SA outputs based on:

(1) building a machine learning model[4] trained on the ouputs from a sensitivity analysis set of simulations, and

(2) applying explanatory model analysis methods to the machine learning model in order to generate policy insights into the original system dynamics model.

[4] The statistical/machine learning model we use is **random forest regression,** which is a form of supervised learning (it has the same aim as linear regression) to predict a continuous valued output (e.g. a system dynamics payoff variable) based on sets of inputs (e.g. different sets of parameter values generated from a sensitivity analysis run).

2. Explanatory Model Analysis

Statistical learning refers to a set of tools for making sense of complex datasets, and for supervised learning, this involves building a statistical model for predicting, or estimating, an output based on one or more inputs (James, Witten, Hastie, Tibshirani, & Taylor, 2023). Supervised learning models can be viewed as a trade-off between flexibility and interpretability, where methods range from lower flexibility and higher explainability (e.g. linear regression) to high flexibility with lower explainability (e.g. deep learning methods). In recent times the trend has shifted towards models with higher flexibility, due to the availability of large datasets, and higher computational resources. However, this trend towards the use of more flexible learning methods (for example, large deep-neural models may contain overall a billion coefficients/parameters), has not been matched with enhancement in methods to (1) explain models and generate insights and (2) evaluate model performance and weaknesses (Biecek & Burzykowski, 2021).

Biecek and Burzykowski (2021) provide a valuable structure for explanatory model analysis, and this is nicely summarise by Kuhn and Silge (2022) into two categories of explanation.

- The first set of methods are termed *local explanations,* and provide information about a prediction for a single observation in the dataset. In our usage of the techniques, this would focus on a single sensitivity run (for example, the best or worst outcome), and show the influence of different features on this outcome. A method known as Shapley additive explanations (SHAP) can be used to support local explanations (Lundberg & Lee, 2017), and we will show this in action in our results section.

- The second set of methods are referred to as *global explanations*, and these include an approach known as variable importance, which address the most important variables for a model in aggregated form.

In this paper our approach is to demonstrate how these *local* and *global explanation methods* can be used to provide insights into the range of outcomes from a system dynamics model. The central idea is that a sensitivity data contains information that can be detected by these approaches, and in doing so, confirm our understanding of the model dynamics and leverage points, or provide new insights that may not be evident from the traditional sensitivity analysis process. The methods used are now summarised.

Shapley additive explanations (SHAP)

Local-level exploration methods help us to understand how a model yields a prediction for a particular single observation, and "variable attributions" can be used to break down the model into contributions that can be attributed to different features (Biecek & Burzykowski, 2021). Shapley additive explanations (SHAP) for average attributions are a useful method, which averages the value of a variable's attribution over all (or a large number of) possible orderings. The reason this is required is that the orderings of features influence the final calculation of attribution, therefore we randomly sample from a range of different orderings. The value of this approach is that we can observe, for any individual simulation run, the average attribution (contribution) of each variable to the run. The SHAP analysis will include,

for each variable, the mean attribution and a boxplot summarising the distribution of contributions for each explanatory variable (i.e. for each sensitivity variable in the dataset). A practical example of this process is shown in the result section. A drawback of this approach is that they provide additive contributions of explanatory variables, and if the model is not additive, they can be misleading (Biecek & Burzykowski, 2021). Also, for larger models, calculation of the SHAP values can be time-consuming.

Variable Importance

The variable importance method (Kuhn & Silge, 2022) is useful for the evaluation of the importance of an explanatory variable, and can also be used to help assess the validity of the model based on domain knowledge (Biecek & Burzykowski, 2021). In general there are two methods: model-specific that have knowledge of the model structure, and model-agnostic that do not assume anything about the model structure. Our approach is to use model-agnostic methods. The main aim is to measure how much does a model's performance change if the effect of a selected explanatory variable, or a group of variables, is removed (Biecek & Burzykowski, 2021).

If a variable is important, we would expect that after permuting the values of the variable, the model performance will deteriorate. A fuller description of the algorithm is provide in Biecek and Burzykowski (2021), page 194, and we use this method in our results section in order to identify the most important variables for the experiment.

Partial dependence profiles

The main idea underlying partial dependence profiles (PDP) is to show how does the expected value of the model prediction behave as a function of a selected explanatory variable (Biecek & Burzykowski, 2021). For example, when predicting the price of a house, a PDP can show how the expected value of the model prediction changes as a function of a feature (e.g. the age of the building or number of rooms) (Kuhn & Silge, 2022). This can be used in a number of ways. First, when different models are being used (e.g. linear regression and random forest), it can be used to confirm overall alignment between the models. They can also be used for comparison purposes within sub-groups of the data.

A PDP is estimated from the mean of the ceteris-paribus profiles (not discussed here) for all instances of a dataset (Biecek & Burzykowski, 2021). In the results section, we will explore how to use this method to show how the model payoff changes with respect to (1) policy flags and (2) continuous sensitivity parameters.

Tools used for analysis – tidymodels and DALEX packages in R

In order to generate model insights, we rely on the tidy modelling approach (Kuhn & Silge, 2022) which is a metapackage in R that organises statistical learning as a workflow, and provides functions to build, test, and evaluate statistical models. For providing explanations for the models, we use the `DALEX` package in R (Biecek & Burzykowski, 2021), which is also

integrated into the `tidymodels` R package. The R code used for the analysis is contained in Appendix 2 – Explanatory Model Analysis.

We now present the overall proposed method, which can be applied to any system dynamics model.

3. Proposed Explanatory Model Analysis Method for System Dynamics

The starting point for explanatory model analysis is the completion of an initial working system dynamics model, along with feedback loops. The following method is proposed.

1. Augment the system dynamics model

    a. Identify a model **payoff variable**, which captures a value that indicates the overall result of the simulation output – a form of *key performance indicator* for the model. For example, in disease models for public health, this value could be the overall attack rate (#recovered/total population).

    b. Add flags (variables that can be 0 or 1) to activate/deactivate prioritised feedback loops, a process that is similar in intent to loop knockout analysis (Sterman, 2000).

    c. Identify sensitivity parameters, their ranges, and the statistical distributions to sample from (e.g. uniform, normal, etc).

2. Generate a single sensitivity dataset that samples from the flags (0 or 1) and the range of sensitivity parameters, and output the final result based on the payoff variable. Perform initial exploratory model analysis on this dataset to confirm overall expected range of outcomes for key variables (e.g. the use of box plots and/or pairwise plots).

3. Using the sensitivity dataset, build a supervised statistical learning regression model (here we use random forest regression). Test this across a range of model hyperparameters, and, following cross validation, select a model with the lowest root mean squared error. This model then becomes the basis for the explanatory model analysis tasks.

4. Perform explanatory model analysis, which can include instance and dataset level computations.

5. Communicate insights as part of the model building process, and refine/update system dynamics model as appropriate.

4. A Model to Support Pandemic Preparedness

In order to demonstrate how this approach can be used, we focus on a public health model to support pandemic preparedness. System dynamics is a valuable method to address a range of healthcare problems, including the prevalence of disease (infectious and non-infectious), and healthcare delivery systems (Darabi & Hosseinichimeh, 2020). In our model, we combine both infectious disease transmission (e.g. the outbreak), and its impact on healthcare delivered (modelled as a simple hospital queue (Wolstenholme & McKelvie, 2019) that deals with both regular care (e.g. scheduled procedures), and pandemic surges (which are given priority). The advantage of the model is that it combines upstream effects for the hospital, for example, public health restrictions and population vaccination can impact the hospital performance (as measure by the average queuing delay). The model is shown in Figure 1 (the full set of equations[5] is listed in Appendix 1), and has the following features:

a. *A population-based SEIR model* (Brauer, Van den Driessche, Wu, & Allen, 2008), with an endogenous transmission coefficient.

b. *A vaccination structure*, governed by a time-varying (goal seeking) vaccination fraction (Andrade, Beishuizen, Stein, Connolly, & Duggan, 2024), which models the ramp-up of available vaccination capacity. The vaccination policy is captured via feedback loop B_2 (vaccinations).

c. *A capacity-constrained hospital system* that can accommodate *regular care* and *pandemic care*. A stock management structure (Sterman, 2000) is used to control the inflow of patients (there are two queues, *Pandemic Patients Waiting* and *Regular Patients Waiting*). Pandemic patients are prioritised (this models the cancellation of planned procedures), and nurses are a constraint on bed availability. Nurses also are vulnerable to infection, and therefore an increasing force of infection can lead to higher rates of absenteeism, as shown in loop B_4 (absenteeism). The average length of stay (ALOS) variable is used to model the disease severity, and there is a multiplier that is used for sensitivity runs.

d. *A payoff mechanism* which is based on the accumulation of the queuing time calculation in the model. The queuing time uses Little's Law to as a means to estimate the variation in delay times given the size of the queue, and the available capacity.

e. *Endogenous behaviours* captured via a dynamic transmission coefficient (BetaT in the model). This varies based on human behaviour reactions to reported cases (LeJeune, Ghaffarzadegan, Childs, & Saucedo, 2025) and also reactions to government restrictions. These two reactions are capture in the feedback loops B_1 (social distancing) and B_3 (government restrictions). The formulations from LeJeune et al. (2025) are used to specify these feedback effects.

[5] The Stella model is also included in the submission, along with the R files used for explanatory model analysis.

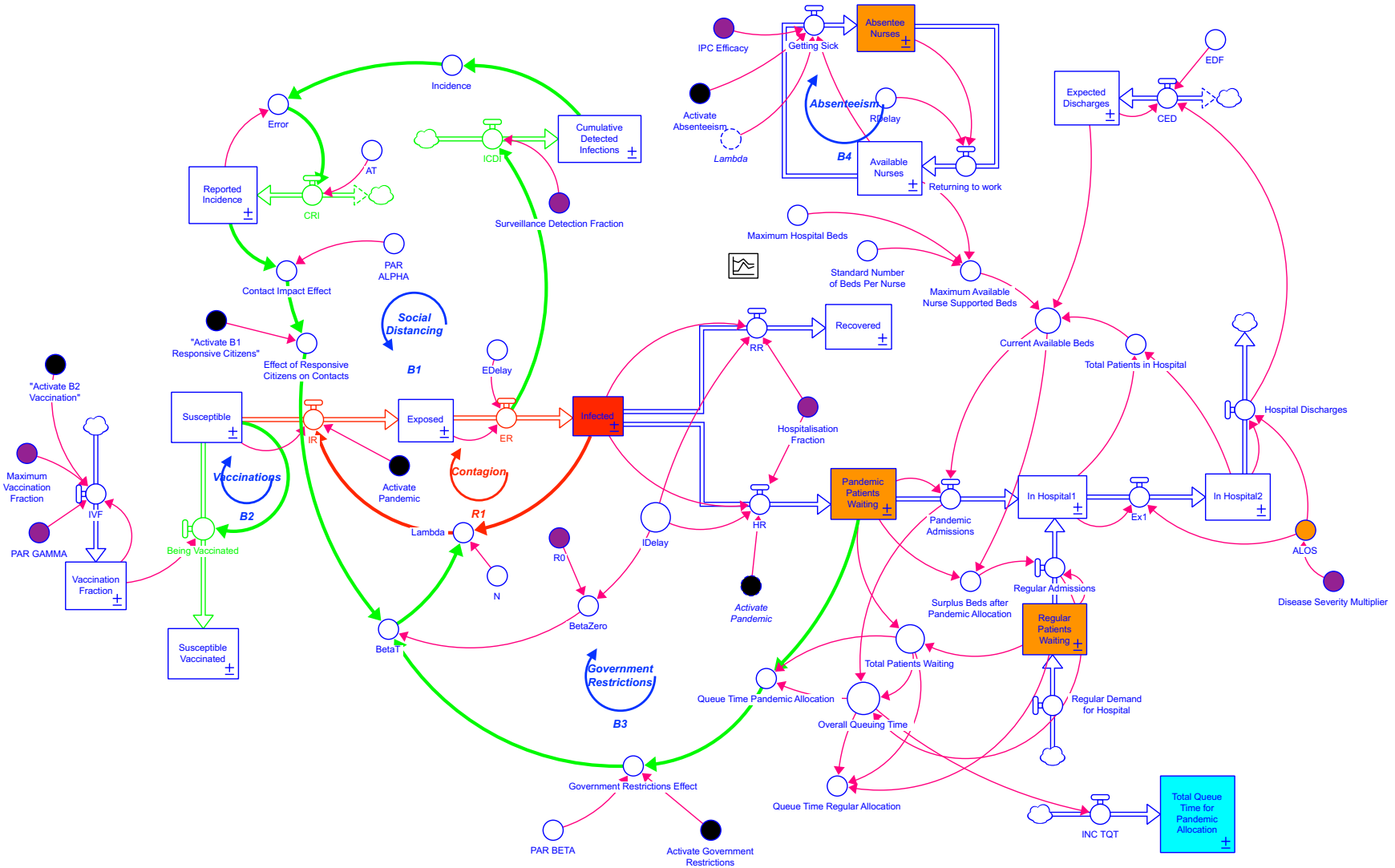

Figure 1: A system dynamics model for pandemic preparedness

For the sensitivity analysis process, there are two categories of parameter:

1. Those with a value of 0 or 1, which act as flags to either activate or deactivate a negative feedback loop, as shown in Table 1.

2. Continuous value parameters with ranges and distributions – these are summarised in Table 2.

Table 1. List of Boolean (flag) parameters used for the sensitivity analysis

Parameter	Description	Range
Activate B1 Responsive Citizens	Activate the feedback loop that models people reacting to higher incidence rates and adjusting their behaviour in light of this information.	0 or 1
Activate B2 Vaccination	Vaccinations active so that susceptible people remove their risk of infection	0 or 1
Activate Government Restrictions	Activate the feedback loop that takes models reduced contacts due to government restrictions. These restrictions are based on a signal from the total number of pandemic patients waiting for treatment.	0 or 1
Activate Absenteeism	Activates a negative feedback loop for modelling nursing staff becoming ill due to the overall force of infection.	0 or 1

Table 2 Continuous parameters used for the sensitivity analysis

Parameter	Description	Range
PAR ALPHA	Activate the feedback loop that takes models people reacting to higher incidence rates and adjusting their behaviour in light of this information.	U(0,0.3)
PAR BETA	Activate the feedback loop that takes models people reacting to government restrictions and adjusting their behaviour in light of this information.	U(0,2.0)
Maximum Vaccination Fraction	The maximum rate at which people are vaccinated. The maximum value corresponds to a time delay of 1/0.15, or ~ 7 days.	U(0,0.15)
PAR GAMMA	Controls the speed at which the maximum vaccination fraction is reached. A value of 0 implies that no vaccination will occur.	U(0,0.75)
R0	The reproduction number of the pathogen, the range is based on plausible values for season influenza.	U(1.5,2.5)
Hospitalisation Fraction	The fraction of infected people who end up being admitted to hospital.	U(0.01,0.05)
Surveillance Detection Fraction	The fraction of infections that get reported. Low detection fractions would impact the behavioural feedback loop.	U(0.2,0.8)
Disease Severity Multiplier	A multiplier that increase disease severity, which in turn impacts the average length of stay (ALOS)	U(0.5,2.0)
IPC Efficacy	The efficacy of infection control measures in the hospital setting.	U(0.5,0.95)

Using the sensitivity analysis tool in Stella Architect[6], and specifying Latin Hypercube sampling, 50,000 simulation runs are performed, and the results save to a csv file for further analysis. The results from this analysis is presented in the following section.

[6] https://www.iseesystems.com/store/products/stella-architect.aspx

5. Experimental Results

Our experiment results are grouped into a number of different analyses:

a. Exploratory data analysis, which summarise the sensitivity runs (quantile values) and also box plots of the payoff variable shown by loop activation status.

b. The random forest model, which is supervised learning model that is trained on the sensitivity data, and provides the basis for the explanatory model analysis techniques.

c. Local explanations, and how these can be used to explore the influences on a specific sensitivity run.

d. Global explanations, which uses variable of importance and partial dependence profiles to generate addition insights.

Exploratory data analysis

Our first set of results contain what could be termed the classic output of a sensitivity analysis run, which is a summary of the trajectories in terms of the confidence bounds for multivariate sensitivity analysis (Sterman, 2000). These results (time 0 to 100) are shown in Figure 2, and capture several variables of interest, including:

- The force of infection which depletes the susceptible stock.

- The model payoff, which is focused on the performance of the hospital system. This was a choice made for this paper, and there could of course be other candidate payoff variables, for example, the population attack rate.

- The contact impact effect, which is the endogenous behaviour multiplier that leads to reduced contacts (LeJeune et al., 2025).

- The total number of patients in the hospital. In steady state (with no pandemic and an ALOS of 10, this would be 400).

- The number of pandemic patients waiting for treatment.

- The regular patients queue, which would be steady-state if there were no pandemic.

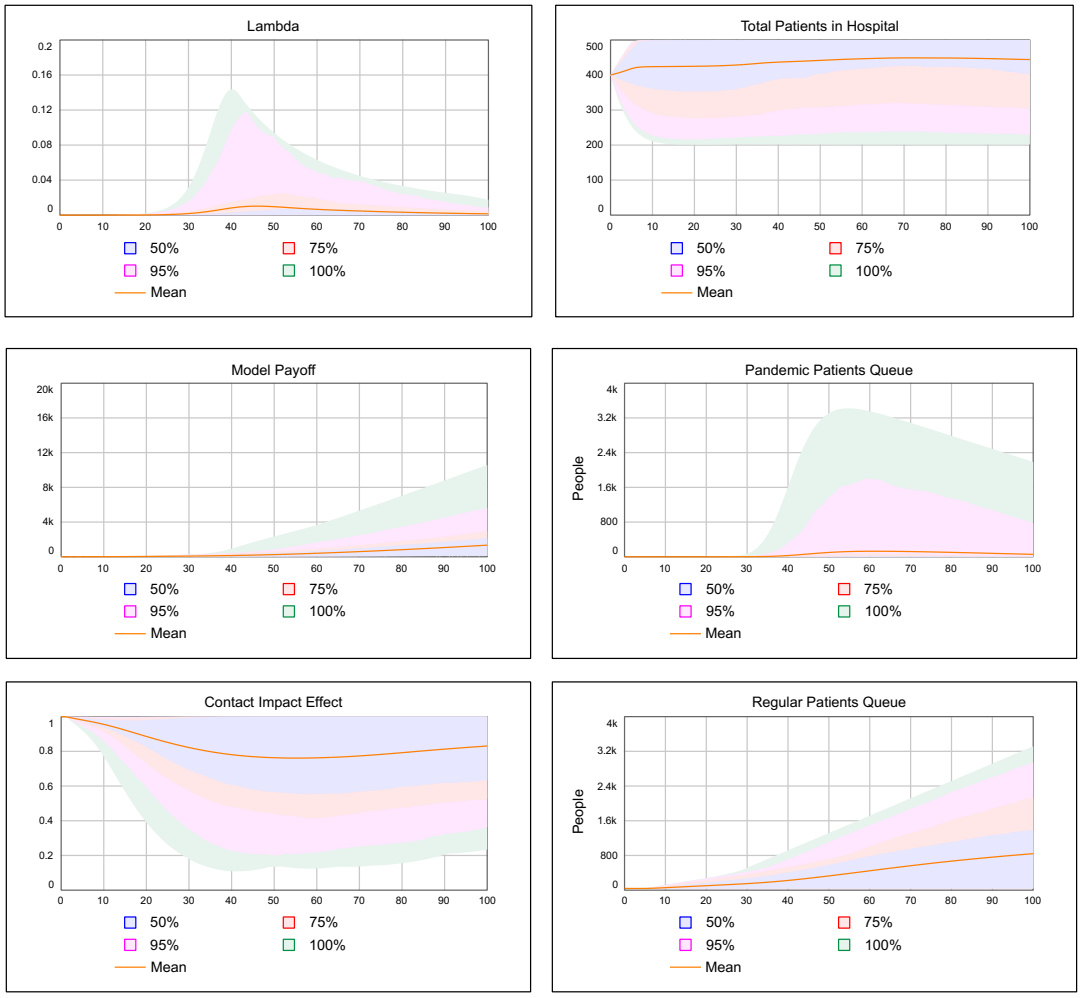

Figure 2 Selection of sensitivity results over the first 100 days.

We can now start to explore the sensitivity dataset that has one observation for every simulation, as we just record one final value of the payoff variable along with the parameter values that generated the result. The full list of variables in the dataset is shown in Table 3, which have already been described in Table 1 and Table 2. The majority are directly from the simulation, although some (loop combinations) were added.

Table 3 The full list of features in the sensitivity dataset (N=50,000)

Variable/Feature	**Variable Type**	**Variable/Feature**	**Variable Type**
Payoff	*Numeric*	P_R0	*Numeric*
L_B1_Citizens	*Boolean*	P_Fr_Hosp	*Numeric*
L_B2_Vaccinate	*Boolean*	P_Mult_Disease	*Numeric*
L_B3_Restrict	*Boolean*	P_Fr_Surv	*Numeric*
L_B4_Absentee	*Boolean*	P_Fr_IPC	*Numeric*
P_Alpha_Citizens	*Numeric*	L_B1_B2	*Boolean*
P_Beta_Govt	*Numeric*	L_B1_B3	*Boolean*
P_Fr_Max_Vacc	*Numeric*	L_B2_B3	*Boolean*
P_Fr_Vacc_Speed	*Numeric*	L_B1_B2_B3	*Boolean*

A useful exploration is to generate box-plot representations for the payoff variable (log due to wide payoff ranges), and group these by the different policy switches. This provides an overall insight into how a policy can impact the outcome, and shows the impact of how activating the balancing loops reduces the outcome. These results are shown in Figure 3.

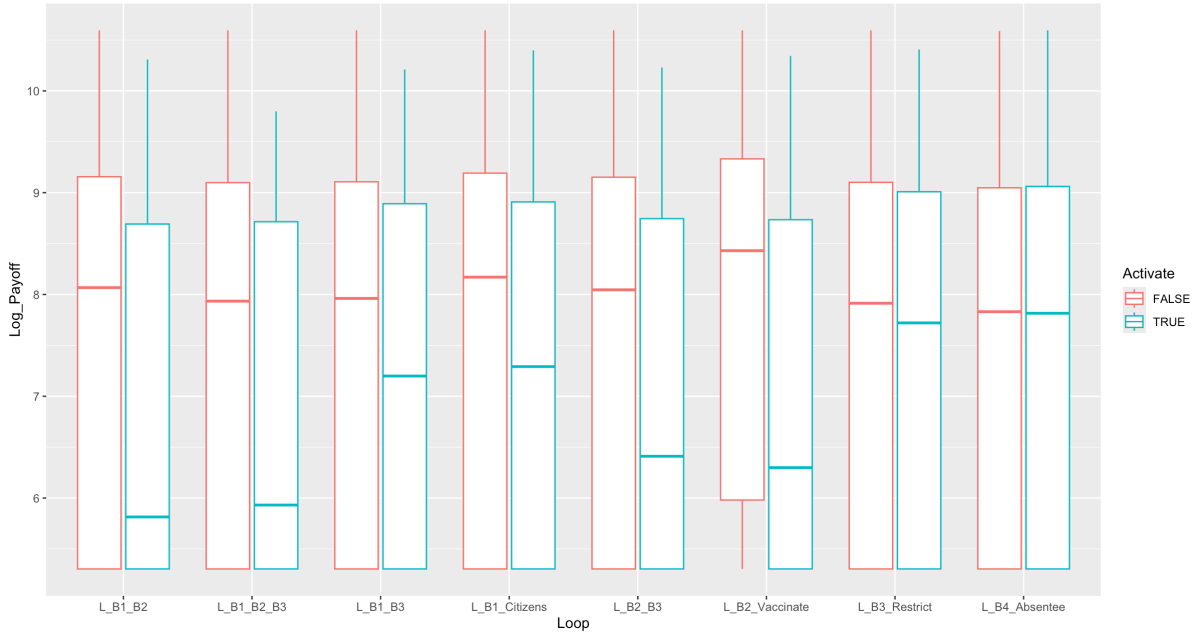

Figure 3 Exploring the effect of policy switches/flags on the model outcome (log of payoff used to generate more informative graphs)

However, while useful, here is a limitation to this type of analysis, because we have not used formal approaches to generate insights. This is where the role of explanatory model analysis can be deployed. The first step in this process is to build a random forest regression model to predict the output (e.g. the payoff variable) as a function of all of the inputs. We use the tidy modelling process to build the model.

The random forest regression model

Random forests are a *tree-based* method for regression and classification, where, in order to make a prediction, the mean or mode response values for training observations in the region to which it belongs are used (James et al., 2023). In contrast to linear regression, random forest regression does not make any assumptions about the distribution of the data (Smith, Ganesh, & Liu, 2013), and can be viewed as a more flexible approach to ordinary least squares regression (James et al., 2023). Our approach to model building is based on the standard machine learning pipeline of activities. Because a random forest can be parameterised (e.g. the number of trees to be used within an ensemble, which is often called a hyperparameter), we use a grid search and k-fold cross-validation approach to select the best performing random forest model. The result of this process is shown in Table 1.

Table 4 tidymodel results for cross-validation process yielding the best performing random forest model (1st row)

Random forest hyperparameters			*Other validation information*					
mtry	**trees**	**min_n**	**metric**	**estimator**	**mean**	**n**	**std_err**	**model#**
17	**722**	**2**	**rmse**	**standard**	**9.24**	**10**	**2.69**	**9**
20	833	7	rmse	standard	10.6	10	3.17	10
13	1000	4	rmse	standard	11.8	10	3.20	7
15	555	8	rmse	standard	12.0	10	3.62	8
11	888	10	rmse	standard	17.5	10	4.21	6

In order to appreciate the overall performance of the fitted regression tree, it is useful to plot the actual payoff values (from the sensitivity data set) against those predicted by the best performing random forest, and these are shown in Figure 4. The overall performance looks good, although it can be seen that for very high payoff values, the model exhibits some bias as fewer data points appear below the line of 45°.

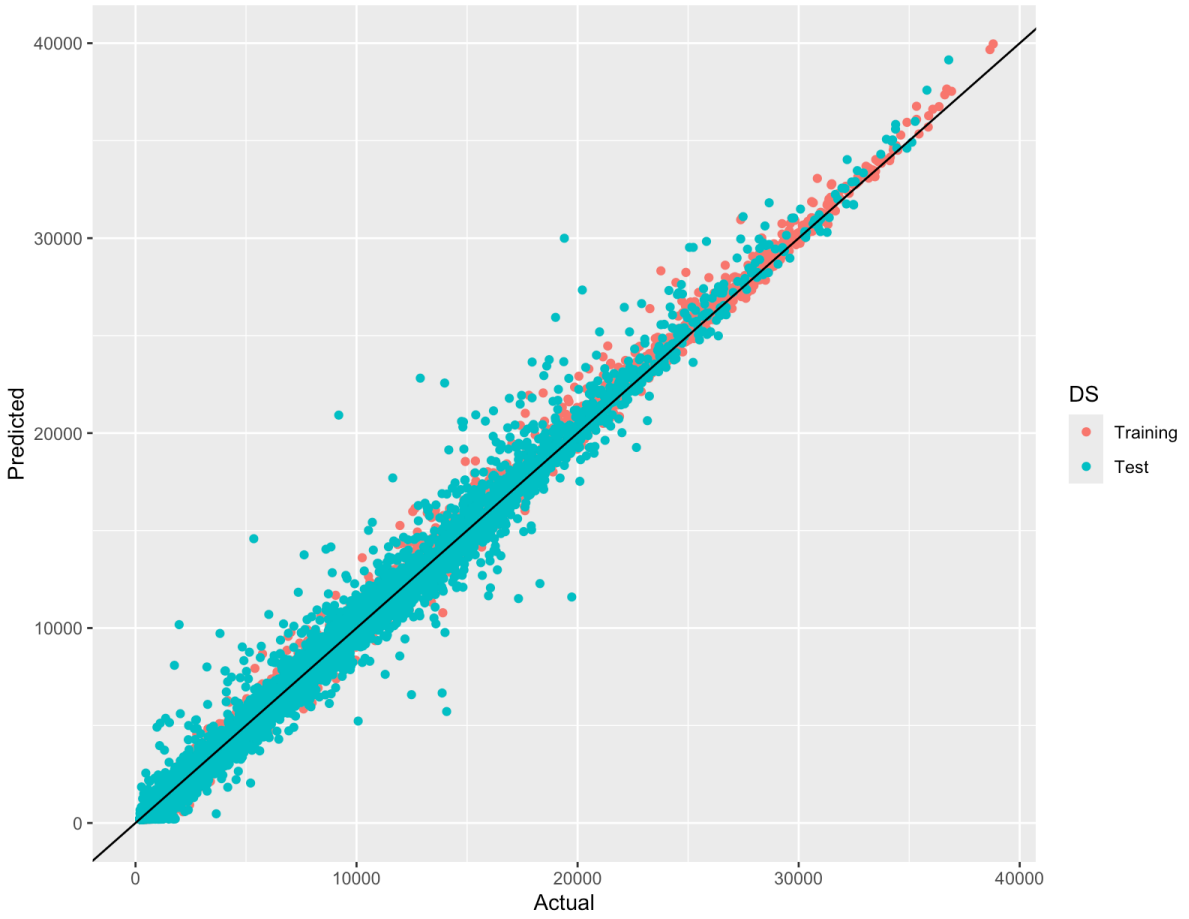

Figure 4 Showing the relationship between actual and predicted values generated by the best performing random forest model.

Our random forest model now becomes the input to the process of explanatory model analysis, which will generate insights on the overall system dynamics model. These insights will be at a local level (Shapley values), and at a global level (variable importance and partial dependence plots).

Shapley values

Based on the fitted model, we can calculate Shapley values for any of the 50,000 simulation runs. As mentioned in the earlier section, these values indicate a value that measures the attribution of an input parameter to the overall payoff value for an individual simulation run. Two scenarios are shown in Figure 5, and for example, the highest payoff value indicates that the most influential parameters included P_Mult_Disease (1.999), L_B1_Citizens (FALSE/deactivated), L_B2_Vaccinate (FALSE/deactivated), and P_Fr_Hosp (0.4969). A useful feature of this approach is that the actual parameter values that led to the outcome are displayed on the plot.

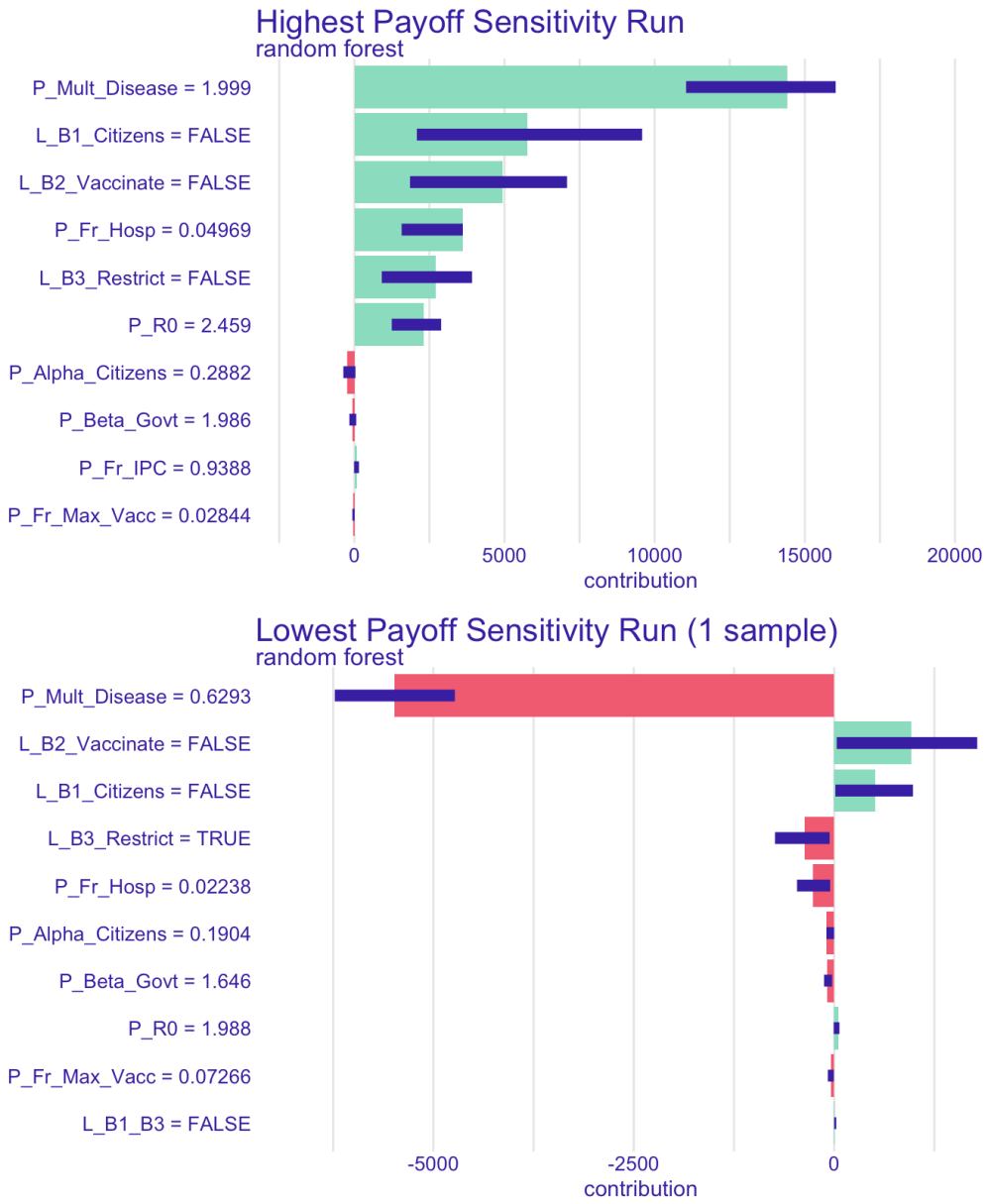

Figure 5 Average attributions for two observations (highest payoff and lowest payoff). 20 random orderings were used. The red and green bars denote the means, while the box plots summarise the distributions of contributions for each explanatory variable across the ordering

Variable importance and partial dependence profiles

At the global dataset level (as opposed to individual simulations), we can use the variable of importance measure to provide insights. This is shown in Figure 6, which indicates that the disease severity multiplier is the most influential parameter. When we reflect back on the model equations we can see that this makes sense, as this multiplier is used to change the ALOS, and the longer people spend in hospital, the more pressure is exerted on the queues. Interestingly two of the balancing loops are listed, confirming the potential of effective disease countermeasures on the over health service pressure.

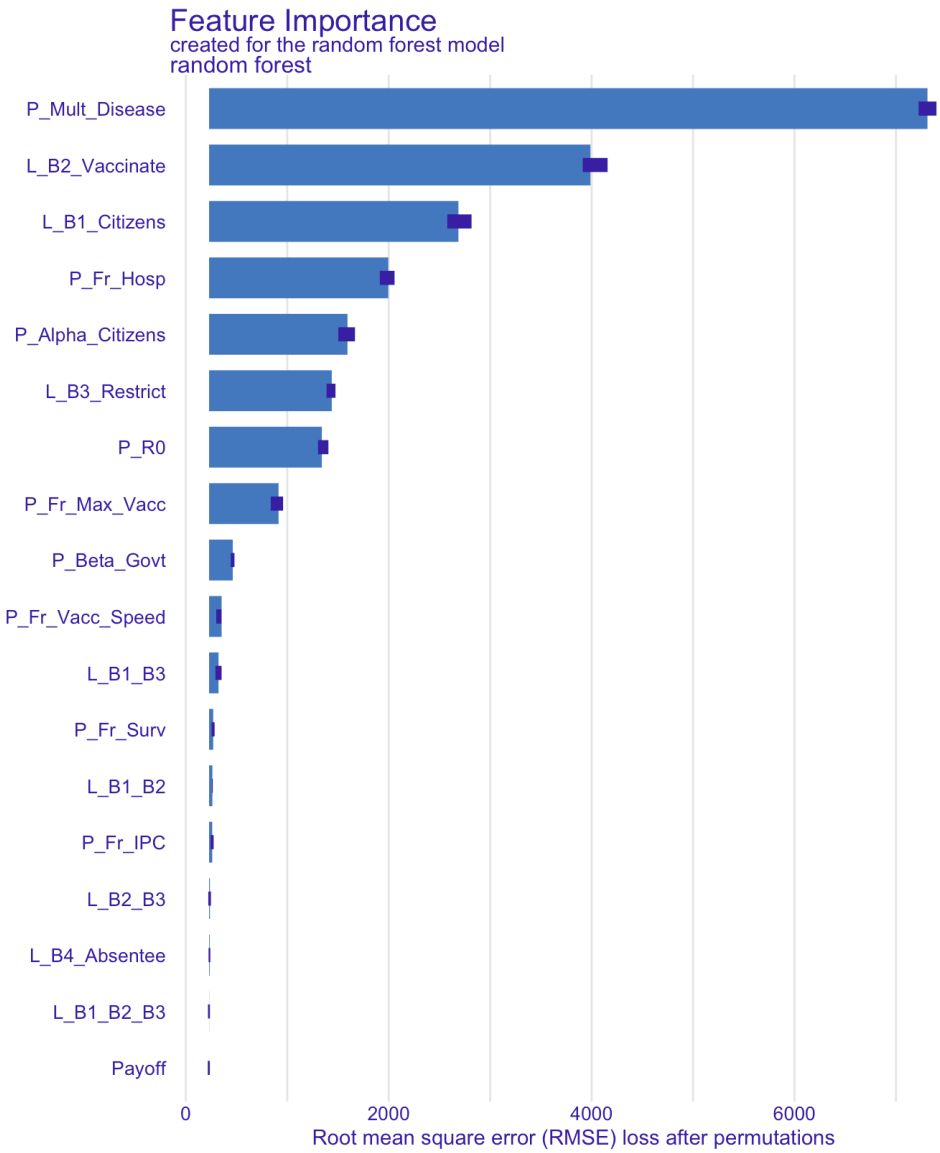

Figure 6 Variable of importance plot for the overall dataset (global explanation)

Partial dependence profiles

As already discussed, partial dependence profiles can show how the expected value of a model prediction can behave based on the input of an explanatory variable. For our analysis, we can categorise the analysis into two elements: (1) variables that are values either true (1) or false (0), for example, whether a loop is active or not, and (2) variables that vary across a range of values (e.g. ones that may be drawn from a uniform distribution).

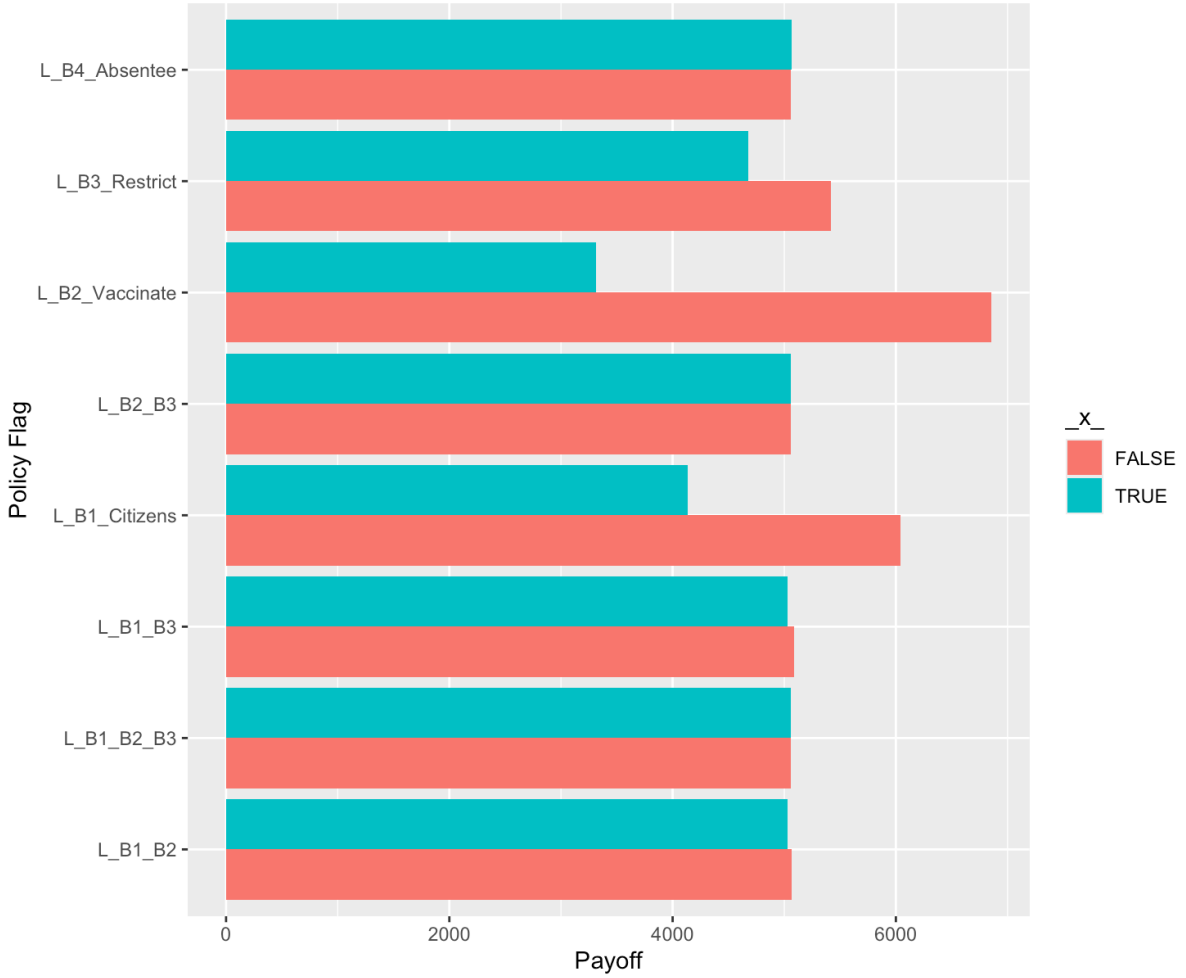

Figure 7 Partial dependence profiles for TRUE|FALSE variables - namely loop activation variables.

Figure 7 shows the results from the dataset, and the interesting results from the model are those variables that show a large difference between the two possible values, indicating that they have a measurable effect on the outcome. In our runs, this shows that *L_B2_Vaccinate* and *L_B1_Citizens* have the largest impact[7]. In Figure 8, the partial dependence profiles are shown for the continuous values parameters, and here the ones that show interesting relationships are *P_Mult_Disease* (influences ALOS), *P_R0* (influences transmissibility), *P_Fr_Hosp* (hospitalisation fraction), *P_Alpha_Citizens* (influences mobility), and *P_Fr_Max_Vacc* (influences speed of vaccination).

[7] It is important to note that this conclusion is valid only for the specifics of this particular model and the parameters that were selected.

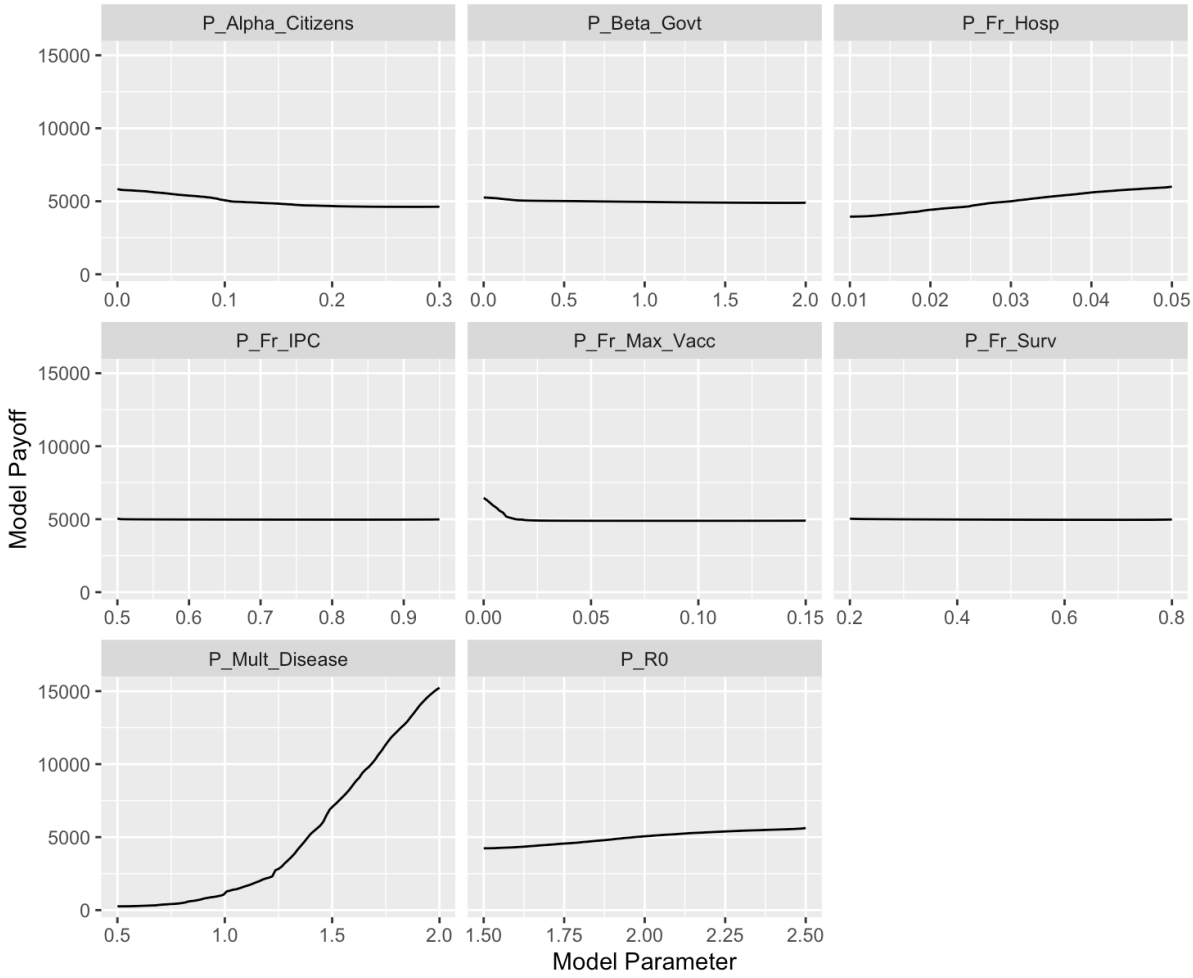

Figure 8 Partial dependence profiles for continuous values sensitivity analysis parameters

6. Conclusion

We believe this method can generate useful results to contribute to the system dynamics modelling process. In terms of alignment, it could benefit two steps of an iterative approach to the modelling process, namely: step (4) *testing* and step (5) *policy formulation and evaluation* (Sterman, 2000). Overall, we see the following advantages and disadvantages.

Advantages

a. The toolsets used (R tidymodels and DALEX) are open source and can be integrated with outputs from system dynamics modelling tools via comma separated value (csv) files.

b. The model building and explanation libraries provide additional insights that are not achievable using previous approaches to analysing sensitivity data generated by system dynamics models.

c. It could be integrated with the overall model building process, and dynamic document tools such as R markdown (Xie, Allaire, & Grolemund, 2018) could be used to support communication of results

Disadvantages

a. It would require SD modellers to have require skills in open source data science tools such as R/Python, and knowledge such as statistical learning, and the use of methods such as linear regression and random forests, and resampling approaches like crossfold-validation. These methods are covered comprehensively in the R and Python textbooks by (James et al., 2023).

b. The use of these methods has not been tested or evaluated in SD modelling practice, therefore there is no evidence as to how they could measurably improve the model building process.

c. Depending on the size of the dataset, calculations of the measures can be computationally intensive. Also, the analysis is sensitive to the choice or parameters, therefore a well-calibrated model with carefully chosen parameters should yield more insightful results.

d. Further modelling is needed to examining the stability of parameter importance rankings across different random seeds, subsamples, or alternative model hyperparameters.

7. Funding

This study was carried out as part of the EU RAPIDE project. RAPIDE is an interdisciplinary research project with thirteen partners including the University of Galway, and is coordinated by the Radboud University Medical Center in the Netherlands. The RAPIDE project is funded by the European Union's research and innovation programme under Grant Agreement No. 101136348.

8. References

Andrade, J., Beishuizen, B., Stein, M., Connolly, M., & Duggan, J. (2024). Preparing for pandemic response in the context of limited resources. *System Dynamics Review, 40*(3), e1775. doi:https://doi.org/10.1002/sdr.1775

Barlas, Y. (1996). Formal aspects of model validity and validation in system dynamics. *System Dynamics Review, 12*(3), 183-210. doi:https://doi.org/10.1002/(SICI)1099-1727(199623)12:3<183::AID-SDR103>3.0.CO;2-4

Biecek, P., & Burzykowski, T. (2021). *Explanatory model analysis: explore, explain, and examine predictive models*: Chapman and Hall/CRC.

Brauer, F., Van den Driessche, P., Wu, J., & Allen, L. J. (2008). *Mathematical epidemiology* (Vol. 1945): Springer.

Darabi, N., & Hosseinichimeh, N. (2020). System dynamics modeling in health and medicine: a systematic literature review. *System Dynamics Review, 36*(1), 29-73. doi:https://doi.org/10.1002/sdr.1646

Edali, M. (2022). Pattern-oriented analysis of system dynamics models via random forests. *System Dynamics Review, 38*(2), 135-166. doi:https://doi.org/10.1002/sdr.1706

Edali, M., & Yücel, G. (2019). Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling. *Simulation Modelling Practice and Theory, 92*, 62-81. doi:https://doi.org/10.1016/j.simpat.2018.12.006

Ford, A., & Flynn, H. (2005). Statistical screening of system dynamics models. *System Dynamics Review, 21*(4), 273-303. doi:https://doi.org/10.1002/sdr.322

Hekimoğlu, M., & Barlas, Y. (2016). Sensitivity analysis for models with multiple behavior modes: a method based on behavior pattern measures. *System Dynamics Review, 32*(3-4), 332-362. doi:https://doi.org/10.1002/sdr.1568

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). Statistical learning. In *An introduction to statistical learning: With applications in Python* (pp. 15-67): Springer.

Kuhn, M., & Silge, J. (2022). *Tidy modeling with R: A framework for modeling in the tidyverse*: " O'Reilly Media, Inc.".

LeJeune, L., Ghaffarzadegan, N., Childs, L. M., & Saucedo, O. (2025). Formulating human risk response in epidemic models: Exogenous vs endogenous approaches. *European Journal of Operational Research, 324*(1), 246-258. doi:https://doi.org/10.1016/j.ejor.2025.01.004

Lundberg, S. M., & Lee, S.-I. (2017). *A unified approach to interpreting model predictions*. Paper presented at the Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA.

Pruyt, E., & Islam, T. (2015). On generating and exploring the behavior space of complex models. *System Dynamics Review, 31*(4), 220-249. doi:https://doi.org/10.1002/sdr.1544

Smith, P. F., Ganesh, S., & Liu, P. (2013). A comparison of random forest regression and multiple linear regression for prediction in neuroscience. *Journal of Neuroscience Methods, 220*(1), 85-91. doi:https://doi.org/10.1016/j.jneumeth.2013.08.024

Sterman, J. (2000). *Business dynamics: Systems thinking and modeling for a complex world*: McGraw-Hill.

Taylor, T. R. B., Ford, D. N., & Ford, A. (2010). Improving model understanding using statistical screening. *System Dynamics Review, 26*(1), 73-87. doi:https://doi.org/10.1002/sdr.428

Wolstenholme, E. F., & McKelvie, D. (2019). *The Dynamics of Care: Understanding People Flows in Health and Social Care*: Springer.

Wu, J., Dhingra, R., Gambhir, M., & Remais, J. V. (2013). Sensitivity analysis of infectious disease models: methods, advances and their application. *J R Soc Interface, 10*(86), 20121018. doi:10.1098/rsif.2012.1018

Xie, Y., Allaire, J. J., & Grolemund, G. (2018). *R markdown: The definitive guide*: Chapman and Hall/CRC.

Appendix 1 – The Pandemic Preparedness Model

Pandemic Preparedness Model - Equations		
Absentee_Nurses(t)	Absentee_Nurses(t - dt) + (Getting_Sick - Returning_to_work) * dt	INIT Absentee_Nurses = 0
Available_Nurses(t)	Available_Nurses(t - dt) + (Returning_to_work - Getting_Sick) * dt	INIT Available_Nurses = 100
Cumulative_Detected_Infections(t)	Cumulative_Detected_Infections(t - dt) + (ICDI) * dt	INIT Cumulative_Detected_Infections = 0
Expected_Discharges(t)	Expected_Discharges(t - dt) + (CED) * dt	INIT Expected_Discharges = 40
Exposed(t)	Exposed(t - dt) + (IR - ER) * dt	INIT Exposed = 0
In_Hospital1(t)	In_Hospital1(t - dt) + (Pandemic_Admissions + Regular_Admissions - Ex1) * dt	INIT In_Hospital1 = 200
In_Hospital2(t)	In_Hospital2(t - dt) + (Ex1 - Hospital_Discharges) * dt	INIT In_Hospital2 = 200
Infected(t)	Infected(t - dt) + (ER - RR - HR) * dt	INIT Infected = 1
Pandemic_Patients_Waiting(t)	Pandemic_Patients_Waiting(t - dt) + (HR - Pandemic_Admissions) * dt	INIT Pandemic_Patients_Waiting = 0
Recovered(t)	Recovered(t - dt) + (RR) * dt	INIT Recovered = 0
Regular_Patients_Waiting(t)	Regular_Patients_Waiting(t - dt) + (Regular_Demand_for_Hospital - Regular_Admissions) * dt	INIT Regular_Patients_Waiting = 40
Reported_Incidence(t)	Reported_Incidence(t - dt) + (CRI) * dt	INIT Reported_Incidence = 0
Susceptible(t)	Susceptible(t - dt) + (- Being_Vaccinated - IR) * dt	INIT Susceptible = 99999
Susceptible_Vaccinated(t)	Susceptible_Vaccinated(t - dt) + (Being_Vaccinated) * dt	INIT Susceptible_Vaccinated = 0
Total_Queue_Time_for_Pandemic_Allocation(t)	Total_Queue_Time_for_Pandemic_Allocation(t - dt) + (INC_TQT) * dt	INIT Total_Queue_Time_for_Pandemic_Allocation = 0
Vaccination_Fraction(t)	Vaccination_Fraction(t - dt) + (IVF) * dt	INIT Vaccination_Fraction = 0
Being_Vaccinated	Susceptible*Vaccination_Fraction	
CED	(Hospital_Discharges-Expected_Discharges)*EDF	
CRI	Error/AT	
ER	Exposed/EDelay	

Ex1	In_Hospital1/(ALOS/2)	
Getting_Sick	Activate_Absenteeism*(1-IPC_Efficacy)*Lambda*Available_Nurses	
Hospital_Discharges	In_Hospital2/(ALOS/2)	
HR	Activate_Pandemic*Hospitalisation_Fraction*Infected/IDelay	
ICDI	ER * Surveillance_Detection_Fraction	
INC_TQT	Overall_Queuing_Time	
IR	Activate_Pandemic*Lambda*Susceptible	
IVF	"\"Activate_B2_Vaccination\""*(Maximum_Vaccination_Fraction-Vaccination_Fraction)*PAR_GAMMA	
Pandemic_Admissions	MIN(Pandemic_Patients_Waiting, Current_Available_Beds)	
Regular_Admissions	MIN(Surplus_Beds_after_Pandemic_Allocation, Regular_Patients_Waiting)	
Regular_Demand_for_Hospital	40	
Returning_to_work	Absentee_Nurses/RDelay	
RR	(1-Hospitalisation_Fraction)*Infected/IDelay	
"\"Activate_B1_Responsive_Citizens\""	0	
"\"Activate_B2_Vaccination\""	0	
Activate_Absenteeism	0	
Activate_Government_Restrictions	0	
Activate_Pandemic	1	
ALOS	10 *Disease_Severity_Multiplier	
AT	2	
BetaT	BetaZero*Effect_of_Responsive_Citizens_on_Contacts*Government_Restrictions_Effect	
BetaZero	R0/IDelay	
Contact_Impact_Effect	1/(1+0.5*Reported_Incidence)^PAR_ALPHA	
Current_Available_Beds	(Maximum_Available_Nurse_Supported_Beds-Total_Patients_in_Hospital)+Expected_Discharges	

Disease_Severity_Multiplier	1	
EDelay	2	
EDF	0.5	
Effect_of_Responsive_Citizens_on_Contacts	IF "\"Activate_B1_Responsive_Citizens\"" = 1 THEN Contact_Impact_Effect ELSE 1	
Error	Incidence-Reported_Incidence	
Government_Restrictions_Effect	IF Activate_Government_Restrictions = 1 THEN 1/(1+0.5*Queue_Time_Pandemic_Allocation)^PAR_BETA ELSE 1	
Hospitalisation_Fraction	0.025	
IDelay	2	
Incidence	Cumulative_Detected_Infections-HISTORY(Cumulative_Detected_Infections, TIME-1)	
IPC_Efficacy	0.80	
Lambda	BetaT/N*Infected	
Maximum_Available_Nurse_Supported_Beds	MIN(Available_Nurses*Standard_Number_of_Beds_Per_Nurse, Maximum_Hospital_Beds)	
Maximum_Hospital_Beds	500	
Maximum_Vaccination_Fraction	0.10	
N	100000	
Overall_Queuing_Time	Total_Patients_Waiting/(Pandemic_Admissions+Regular_Admissions)	
PAR_ALPHA	0.2	
PAR_BETA	2	
PAR_GAMMA	0.0125	
Queue_Time_Pandemic_Allocation	Pandemic_Patients_Waiting/Total_Patients_Waiting*Overall_Queuing_Time	
Queue_Time_Regular_Allocation	Overall_Queuing_Time*Regular_Patients_Waiting/Total_Patients_Waiting	
R0	2	
RDelay	7	
Standard_Number_of_Beds_Per_Nurse	5	

Surplus_Beds_after_Pandemic_Allocation	MAX(0, Current_Available_Beds-Pandemic_Patients_Waiting)	
Surveillance_Detection_Fraction	.5	
Total_Patients_in_Hospital	In_Hospital1+In_Hospital2	
Total_Patients_Waiting	Pandemic_Patients_Waiting+Regular_Patients_Waiting	

Appendix 2 – Explanatory Model Analysis

Step 1: Prepare the data

```
library(readr)
library(stringr)
library(dplyr)
library(tidyr)
library(purrr)

# Target stella output file
s_file     <- c("models/RAPIDE/SensitivityRuns.csv")



organise_as_wide <- function(f0){
  f1 <- read_csv(f0)
  st_1 <- separate_wider_delim(f1,
                               Days,delim = ":",
                               names = c("Run", "Variable")) %>%
    mutate(Run=str_replace_all(Run,'\"',"")) %>%
    mutate(Variable=trimws(str_replace_all(Variable,'\"',""))) %>%
    mutate(Variable=str_replace_all(Variable,"-","_")) %>%
    mutate(Variable=str_replace_all(Variable,"\\+","_Plus")) %>%
    mutate(Variable=str_remove_all(Variable," ")) %>%
    mutate(Run=str_replace_all(Run,"=","")) %>%
    mutate(Run=as.integer(str_extract(Run, "(\\d)+"))) %>%
    rename(Value=`200`) %>%
    arrange(Run)

  st_wide <- st_1 %>%
    pivot_wider(id_cols=Run,
                names_from = Variable,
                values_from = Value) %>%
    select(Run,everything())

  st_wide
}


sens_data_1 <- organise_as_wide(s_file)
# Organise the data format and tidy up column names
sens_data_2 <- sens_data_1 %>%
               rename(Payoff=TotalQueueTimeforPandemicAllocation,
                      L_B1_Citizens=ActivateB1ResponsiveCitizens,
                      L_B2_Vaccinate=ActivateB2Vaccination,
                      L_B3_Restrict=ActivateGovernmentRestrictions,
                      L_B4_Absentee=ActivateAbsenteeism,
                      P_R0=R0,
                      P_Alpha_Citizens=PARALPHA,
                      P_Beta_Govt=PARBETA,
```

```
                    P_Fr_Max_Vacc=MaximumVaccinationFraction,
                    P_Fr_Vacc_Speed=PARGAMMA,
                    P_Fr_Hosp=HospitalisationFraction,
                    P_Fr_Surv=SurveillanceDetectionFraction,
                    P_Fr_IPC=IPCEfficacy,
                    P_Mult_Disease=DiseaseSeverityMultiplier
                    ) %>%
             mutate(L_B1_Citizens=as.logical(L_B1_Citizens),
                    L_B2_Vaccinate=as.logical(L_B2_Vaccinate),
                    L_B3_Restrict=as.logical(L_B3_Restrict),
                    L_B4_Absentee=as.logical(L_B4_Absentee),
                    L_B1_B2=L_B1_Citizens & L_B2_Vaccinate,
                    L_B1_B3=L_B1_Citizens & L_B3_Restrict,
                    L_B2_B3=L_B2_Vaccinate & L_B3_Restrict,
                    L_B1_B2_B3=L_B1_Citizens & L_B2_Vaccinate & L_B3_Restrict,
                    Sqrt_Payoff=sqrt(Payoff),
                    Log_Payoff=log(Payoff+1))



write_csv(sens_data_2,"ML/datasets/ML_Sensitivity_Input.csv")
```

Step 2: Explore the data

```
library(readr)
library(dplyr)
library(ggplot2)
library(tidyr)


sens_data_ml <- read_csv("ML/datasets/ML_Sensitivity_Input.csv")


loops_long <- sens_data_ml %>%
                select(Run, Log_Payoff,starts_with("L_B")) %>%
                pivot_longer(-c(Run,Log_Payoff),
                             names_to = "Loop",
                             values_to = "Activate")

ggplot(loops_long,aes(y=Log_Payoff,x=Loop,colour=Activate))+geom_boxplot()

# Pairwise plots

pars <- sens_data_ml %>%
          select(Log_Payoff,starts_with("P_"))

p <- get_pair_wise(pars,"Plots")
```

Step 3: Build the random forest models

```
library(tidyverse)
library(tidymodels)
library(vip)
library(ggplot2)
library(glue)

# Overall process acknowledgment:
https://stackoverflow.com/questions/60368047/tidymodels-ranger-with-cross-
validation


# Read the simulation data and generate subset
sim_data <- read_csv("ML/datasets/ML_Sensitivity_Input.csv") %>%
             select(-Run)


# Create training and testing datasets
sim_split <- initial_split(sim_data)
sim_train <- training(sim_split)
sim_test  <- testing(sim_split)

# Create the model object
rf_model <- rand_forest(mtry=tune(),
                        trees=tune(),
                        min_n = tune()) %>%
              set_engine("ranger",importance="impurity") %>%
              set_mode("regression")

rf_model %>% translate()

# Set the workflow
rf_wkfl <- workflow() %>%
            add_formula(Payoff ~ .) %>%
            add_model(rf_model)

# Choose hyperparameter grid
rf_grid <- grid_space_filling(mtry(range=c(1,20)),
                              trees(range=c(500,1000)),
                              min_n(range=c(2,10)),
                              size=10)

# Setup crossfold validation method
cv_fold <- vfold_cv(sim_train,10)

# Choose metrics
rf_mets <- metric_set(rmse)

# Tune mode based on hyperparameter samples
rf_fit <- tune_grid(rf_wkfl,
```

```
                    resamples = cv_fold,
                    grid=rf_grid,
                    metrics=rf_mets,
                    control=control_grid(verbose = TRUE))

saveRDS(list(FIT=rf_fit,
             WORK=rf_wkfl),
        "ML/03 model/tuned_models.rds")
```

Step 4: Compute SHAP values

```
library(tidyverse)
library(tidymodels)
library(vip)
library(ggplot2)
library(ranger)
library(DALEXtra)
library(ggpubr)
# Read the simulation data and generate subset
sim_data <- read_csv("ML/datasets/ML_Sensitivity_Input.csv") %>%
  select(-c(Run,Sqrt_Payoff,Log_Payoff))

sim_split <- initial_split(sim_data)
sim_train <- training(sim_split)
sim_test  <- testing(sim_split)

tune_info <- readRDS("ML/03 model/tuned_models_HF0.rds")


rf_fit <- tune_info$FIT
rf_wkfl <- tune_info$WORK

collect_metrics(rf_fit)
show_best(rf_fit,metric="rmse")
select_best(rf_fit,metric="rmse")
autoplot(rf_fit,metric="rmse")

# Fit according the the best hyperparams
rf_tuned <- rf_wkfl %>%
  finalize_workflow(select_best(rf_fit,metric="rmse")) %>%
  fit(data=sim_train)


explainer_rf <- explain_tidymodels(rf_tuned,
                                   data=sim_train,
                                   y=sim_train$Payoff,
                                   label="random forest",
                                   verbose = FALSE)

set.seed(2026)
```

```r
sim_train %>% arrange(desc(Payoff)) -> sim_train_desc

shap_duplex <- predict_parts(explainer = explainer_rf,
                             new_observation = sim_train_desc[1,],
                             type = "shap",
                             B = 20)

shap_duplex_highest_payoff <- shap_duplex
p1 <- plot(shap_duplex_highest_payoff)+ggtitle("Highest Payoff Sensitivity Run")

shap_duplex_lowest_payoff <- predict_parts(explainer = explainer_rf,
                             new_observation =
sim_train_desc[nrow(sim_train_desc),],
                             type = "shap",
                             B = 20)

p2 <- plot(shap_duplex_lowest_payoff)+ggtitle("Lowest Payoff Sensitivity Run (1
sample)")

saveRDS(list(highest=shap_duplex_highest_payoff,
             lowest=shap_duplex_lowest_payoff),
        "ML/05 local/shap_results.rds")

p3 <- ggarrange(p1, p2, ncol = 1, nrow = 2)
```

Step 5: Compute variable of importance

```
library(tidyverse)
library(tidymodels)
library(vip)
library(ggplot2)
library(ranger)
library(DALEXtra)
library(ggpubr)



# Read the simulation data and generate subset
sim_data <- read_csv("ML/datasets/ML_Sensitivity_Input.csv") %>%
  select(-c(Run,Sqrt_Payoff,Log_Payoff))

sim_split <- initial_split(sim_data)
sim_train <- training(sim_split)
sim_test  <- testing(sim_split)

tune_info <- readRDS("ML/03 model/tuned_models_HF0.rds")


rf_fit <- tune_info$FIT
rf_wkfl <- tune_info$WORK

collect_metrics(rf_fit)
show_best(rf_fit,metric="rmse")
select_best(rf_fit,metric="rmse")
autoplot(rf_fit,metric="rmse")

# Fit according the the best hyperparams
rf_tuned <- rf_wkfl %>%
  finalize_workflow(select_best(rf_fit,metric="rmse")) %>%
  fit(data=sim_train)


explainer_rf <- explain_tidymodels(rf_tuned,
                                   data=sim_train,
                                   y=sim_train$Payoff,
                                   label="random forest",
                                   verbose = FALSE)

vip_rf <- model_parts(explainer_rf,loss_function = loss_root_mean_square)

plot(vip_rf)
```

Step 6: Compute partial dependence profiles

```
library(tidyverse)
library(tidymodels)
library(vip)
library(ggplot2)
library(ranger)
library(DALEXtra)
library(ggpubr)


# Read the simulation data and generate subset
sim_data <- read_csv("ML/datasets/ML_Sensitivity_Input.csv") %>%
  select(-c(Run,Sqrt_Payoff,Log_Payoff))

sim_split <- initial_split(sim_data)
sim_train <- training(sim_split)
sim_test  <- testing(sim_split)

tune_info <- readRDS("ML/03 model/tuned_models_HF0.rds")


rf_fit <- tune_info$FIT
rf_wkfl <- tune_info$WORK

collect_metrics(rf_fit)
show_best(rf_fit,metric="rmse")
select_best(rf_fit,metric="rmse")
autoplot(rf_fit,metric="rmse")

# Fit according the the best hyperparams
rf_tuned <- rf_wkfl %>%
  finalize_workflow(select_best(rf_fit,metric="rmse")) %>%
  fit(data=sim_train)


explainer_rf <- explain_tidymodels(rf_tuned,
                                   data=sim_train,
                                   y=sim_train$Payoff,
                                   label="random forest",
                                   verbose = FALSE)

set.seed(2000)


vars_pol <- c("L_B1_Citizens",
              "L_B2_Vaccinate",
              "L_B3_Restrict",
              "L_B4_Absentee",
              "L_B1_B2",
              "L_B1_B3",
```

```
              "L_B2_B3",
              "L_B1_B2_B3")

rf_pdp_pol <- model_profile(explainer_rf,
                            N=10000,
                            variables=vars_pol)

rf_pdp_pol_tbl <- as_tibble(rf_pdp_pol$agr_profiles)


ggplot(rf_pdp_pol_tbl, aes(x=`_yhat_`, y=`_vname_`,fill=`_x_`))+
  geom_col(position="dodge")+xlab("Payoff")+ylab("Policy Flag")


vars_contin <- c("P_Alpha_Citizens",
                 "P_Beta_Govt",
                 "P_Fr_Max_Vacc",
                 "P_R0",
                 "P_Fr_Hosp",
                 "P_Mult_Disease",
                 "P_Fr_Surv",
                 "P_Fr_IPC")

rf_pdp_pars <- model_profile(explainer_rf,
                            N=10000,
                            variables=vars_contin)



ggplot(rf_pdp_pars$agr_profiles,aes(x=`_x_`,y=`_yhat_`))+
  geom_line()+facet_wrap(~`_vname_`,scales="free_x")+
  ylab("Model Payoff")+xlab("Model Parameter")
```