Addressing Uncertainty in MultiSector Dynamics Research

Patrick M. Reed, Antonia Hadjimichael, Keyvan Malek
Tina Karimi, Chris R. Vernon, Vivek Srikrishnan
Rohini Gupta, David Gold, B. Lee, Klaus Keller, Jennie S. Rice

Sep 07, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

This guidance text has been developed in support of the Integrated Multisector Multiscale Modeling (IM3) Science Focus Area's objective to formally integrate uncertainty into its research tasks. IM3 is focused on innovative modeling to explore how human and natural system landscapes in the United States co-evolve in response to short-term shocks and long-term influences. The project's challenging scope is to advance our ability to study the interactions between energy, water, land, and urban systems, at scales ranging from local (~1km) to the contiguous United States, while consistently addressing influences such as population change, technology change, heat waves, and drought. Uncertainty and careful model-driven scientific insights are central to IM3's key MultiSector Dynamics (MSD) science objectives shown below.

IM3 key MSD science objectives include:

Develop flexible, open-source, and integrated modeling capabilities that capture the structure, dynamic behavior, and emergent properties of the multiscale interactions within and between human and natural systems.

Use these capabilities to study the evolution, vulnerability, and resilience of interacting human and natural systems and landscapes from local to continental scales, including their responses to the compounding effects of long-term influences and short-term shocks.

Understand the implications of uncertainty in data, observations, models, and model coupling approaches for projections of human-natural system dynamics.

Addressing the objectives above poses a strong transdisciplinary challenge that heavily depends on a diversity of models and, more specifically, a consistent framing for making model-based science inferences. The term transdisciplinary science as used here formally implies a deep integration of disciplines to aid our hypothesis-driven understanding of coupled human-natural systems–bridging differences in theory, hypothesis generation, modeling, and modes of inference [1]. The IM3 MSD research focus and questions require a deep integration across disciplines, where new modes of analysis can emerge that rapidly synthesize and exploit advances for making decision-relevant insights that at minimum acknowledge uncertainty and more ideally promote a rigorous quantitative mapping of its effects on the generality of claimed scientific insights. More broadly, diverse scientific disciplines engaged in the science of coupled human-natural systems, ranging from natural sciences to engineering and economics, employ a diversity of numerical computer models to study and understand their underlying systems of focus. The utility of these computer models hinges on their ability to represent the underlying real systems with sufficient fidelity and enable the inference of novel insights. This is particularly challenging in the case of coupled human-natural systems where there exists a multitude of interdependent human and natural processes taking place that could potentially be represented. These processes usually translate into modeled representations that are highly complex, non-linear, and exhibit strong interactions and threshold behaviors [2, 3, 4]. Model complexity and detail have also been increasing as a result of our improving understanding of these processes, the availability of data, and the rapid growth in computing power [5]. As model complexity grows, modelers need to specify a lot more information than before: additional model inputs and relationships as more processes are represented, higher resolution data as more observations are collected, new coupling relationships and interactions as models are put together to answer multisector questions (e.g., the land-water-energy nexus). Typically, not all of this information is well known, nor is the impact of these many uncertainties on model outputs well understood. It is further especially difficult to distinguish the effects of individual as well as interacting sources of uncertainty when modeling coupled systems with multisector and multiscale dynamics [6].

Given the challenge and opportunity posed by the disciplinary diversity of IM3, we utilized a team-wide survey to

allow the project's membership to provide their views on how their areas typically address uncertainty, emphasizing key literature examples and domain-specific reviews. Our synthesis of this survey information in Fig. 1.1 summaries the team's perspectives, enabling a summary of the commonalities and differences for how different disciplinary areas are typically addressing uncertainty. Fig. 1.1 highlights the non-trivial challenge posed by seeking to carefully consider uncertainty across an MSD focused transdisciplinary team. There are significant differences across the team's contributing disciplines in terms of the methodological approaches and tools used in the treatment of uncertainty. The horizontal axis of the figure represents a conceptual continuum of methodological approaches, ranging from deterministic (no uncertainty) modeling to the theoretical case of fully engaging in modeling all sources of uncertainty. The vertical axis of plot maps the analysis tools that are used in the disciplines' literature, spanning error-driven historical analyses to full uncertainty quantification. Given that Fig. 1.1 is a conceptual illustration, the mapping of each discipline's boundaries is not meant to imply exactness. They encompass the scope of feedback attained in the team-wide survey responses. The color circles designate specific sources of uncertainty that could be considered. Within the mapped disciplinary approaches, the color circles distinguish those sources of uncertainty that are addressed in the bodies of literature reported by respondents. Note the complete absence of grey circles designating that, at present, few if any studies report results for understanding how model coupling relationships shape uncertainty. We can briefly distinguish the key terms of uncertainty quantification (UQ) and uncertainty characterization (UC). UQ refers to the formal focus on the full specification of likelihoods as well as distributional forms necessary to infer the joint probabilistic response across all modeled factors of interest [7]. Alternatively, uncertainty characterization as defined here, refers to exploratory modeling of alternative hypotheses for the co-evolutionary dynamics of influences, stressors, as well as path dependent changes in the form and function of modelled systems [8, 9]. Uncertain factors are any model component which is affected by uncertainty: inputs, resolution levels, coupling relationships, model relationships and parameters. When a model has been established as a sufficiently accurate representation of the system some of these factors may reflect elements of the real-world system that the model represents (for example, a population level parameter would reflect a sufficiently accurate representation of the population level in the system under study). As discussed in later sections, the choice of UQ or UC depends on the specific goals of studies, the availability of data, the types of uncertainties (e.g., well-characterized or deep), the complexity of underlying models as well as the computational limits. Deep uncertainty (as opposed to well-characterized) refers to situations where expert opinions consulted on a decision do not know or cannot agree on system boundaries, or the outcomes of interest and their relative importance, or the prior probability distribution for the various uncertain factors present [10, 11].

At present, there is no singular guide for confronting the computational and conceptual challenges of the multi-model, transdisciplinary workflows that characterize ambitious projects such as IM3 [12]. The primary aim of this text is to begin to address this gap and provide guidance for facing these challenges. Section 2 provides an overview of diagnostic modeling and the different perspectives for how we should evaluate our models, Section 3 summarizes basic methods and concepts for sensitivity analysis, and Section 4 delves into more technical applications of sensitivity analysis to support diagnostic model evaluation and exploratory modeling. Finally, Section 5 provides some concluding remarks across the UC and UQ topics covered in this text. The appendices of this text include a glossary of the key concepts as well as an overview of UQ methods.

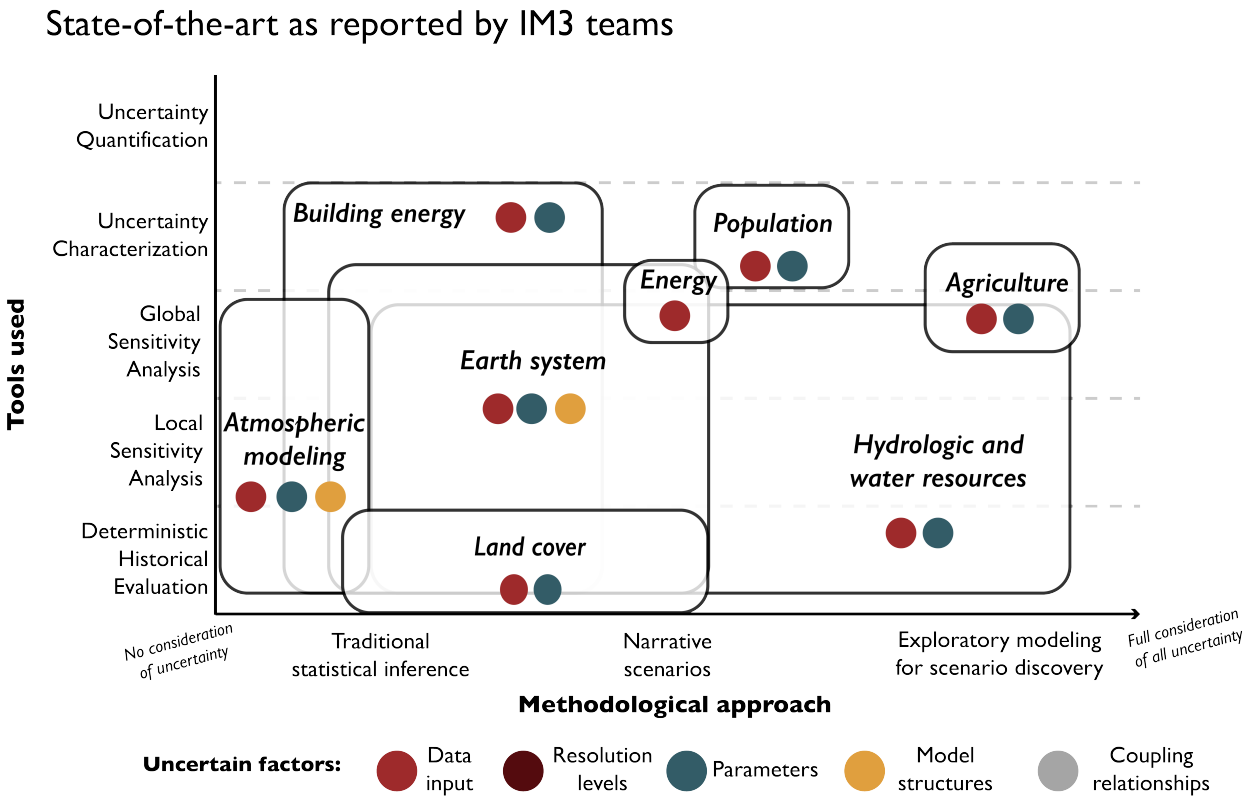


Fig. 1.1: State-of-the-art in different modeling communities, as reported in the survey distributed to IM3 teams. *Deterministic Historical Evaluation*: model evaluation under fully determined conditions defined using historical observations; *Local Sensitivity Analysis*: model evaluation performed by varying uncertain factors around specific reference values; *Global Sensitivity Analysis*: model evaluation performed by varying uncertain factors throughout their entire feasible value space; *Uncertainty Characterization*: model evaluation under alternative factor hypotheses to explore their implications for model output uncertainty; *Uncertainty Quantification*: representation of model output uncertainty using probability distributions; *Traditional statistical inference*: use of analysis results to describe deterministic or probabilistic outcomes resulting from the presence of uncertainty; *Narrative scenarios*: use of a limited decision-relevant number of scenarios to describe (sets of) changing system outcomes; *Exploratory modeling for scenario discovery*: use of large ensembles of uncertain conditions to discover decision-relevant combinations of uncertain factors

CHAPTER

TWO

DIAGNOSTIC MODELING OVERVIEW AND PERSPECTIVES

This text prescribes a formal model diagnostic approach to IM3 computational experimentation that is a deliberative and iterative combination of state-of-the-art UC and global sensitivity analysis techniques that progresses from observed history-based fidelity evaluations to forward looking resilience and vulnerability inferences [13, 14].

2.1 Overview of model diagnostics

Model diagnostics provide a rich basis for hypothesis testing, model innovation, and improved inferences when classifying what is controlling highly consequential results (e.g., vulnerability or resilience in coupled human-natural systems). Fig. 2.1, adapted from [5], presents idealized illustrations of the relationship between UC and global sensitivity analysis (GSA) for two coupled simulation models. The figure illustrates how UC can be used to address how uncertainties in various modeling decisions (data inputs, parameters, model structures, coupling relationships, and elsewhere) can be sampled and simulated to yield the empirical model output distribution(s) of interest. Monte Carlo frameworks allow us to sample and propagate (or integrate) the ensemble response of the model(s) of focus. The first step of any UC analysis is the specification of the initial input distributions as illustrated in Fig. 2.1. The second step is to perform the Monte Carlo simulations. The question can then be raised, which of the modeling assumptions in our Monte Carlo experiment are the most responsible for the resulting output uncertainty. We can answer this question using "global sensitivity analysis" (GSA) as illustrated in Fig. 2.1. GSA can be defined as a formal Monte Carlo sampling and analysis of modeling choices (structures, parameters, inputs) to quantify their influence on direct model outputs (or output-informed metrics). UC experiments by themselves do not explain why a particular uncertain outcome is produced, but produce distributions of model outcomes, as portrayed by the yellow curve. The pie chart shown in Fig. 2.1 is a conceptual representation of the results of GSA to identify those factors that are most dominantly influencing results, either individually or interactively [15].

UC and GSA are not independent modeling analyses. As illustrated here, any GSA requires an initial UC hypothesis in the form of statistical assumptions and representations for the modeling choices of focus (structural, parametric, and data inputs). Information from these two model diagnostic tools can then be used to inform data needs for future model runs, experiments to reduce the uncertainty present, or the simplification or enhancement of the model where necessary. Together UC and GSA provide a foundation for diagnostic exploratory modeling that has a consistent focus on the assumptions, structural model forms, alternative parameterizations, and input data sets that are used to characterize the behavioral space of one or more models.

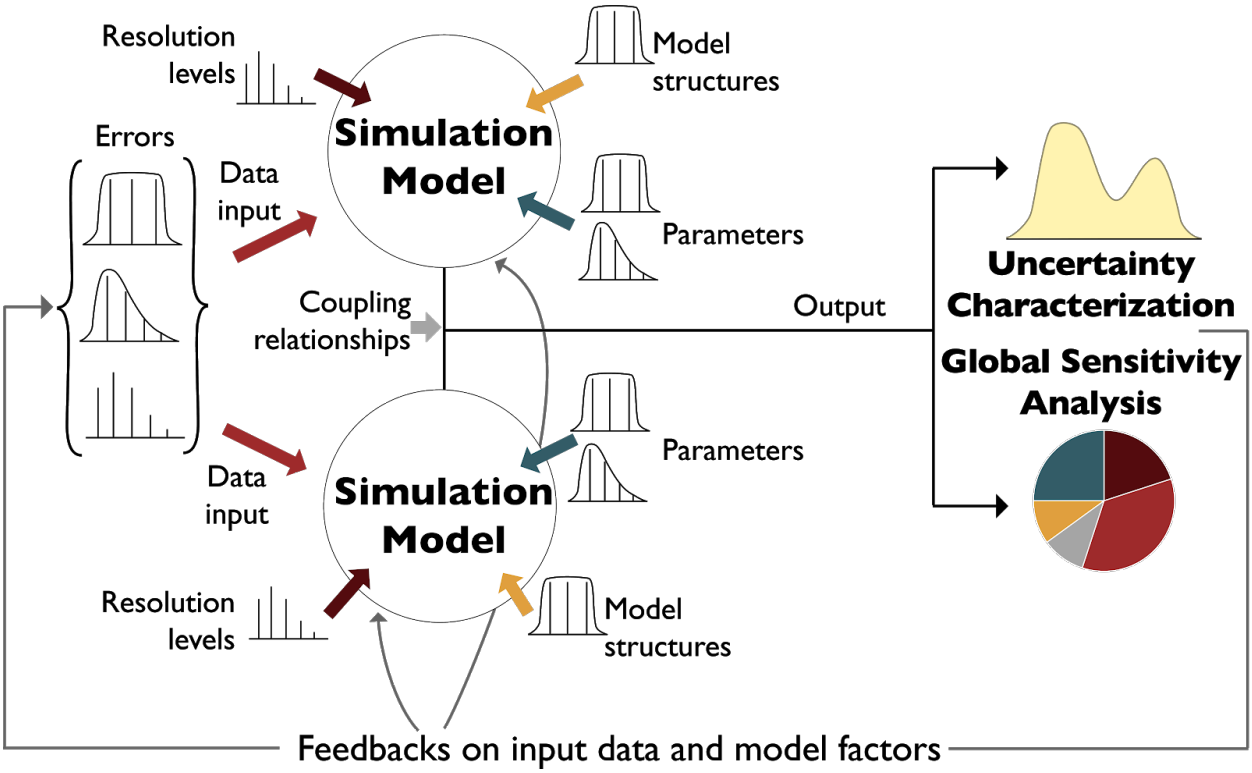


Fig. 2.1: Idealized uncertainty characterization and global sensitivity analysis for two coupled simulation models. Uncertainty coming from various sources (inputs, model structures, coupling relationships, and elsewhere) is propagated through the coupled model(s) to generate empirical distributions of outputs of interest (uncertainty characterization). This model output uncertainty can be decomposed to its origins, by means of sensitivity analysis. Figure adapted from Saltelli *et al.* [5].

2.2 Perspectives on diagnostic model evaluation

When we judge or diagnose models the terms "verification" and "validation" are commonly used. However, their appropriateness in the context of numerical models representing complex coupled human-natural systems is questionable [16, 17]. The core issue relates to the fact that these systems are often not fully known or perfectly implemented when modeled. Rather, they are defined within specific system framings and boundary conditions in an evolving learning process with the goal of making continual progress towards attaining higher levels of fidelity. For example, observations used to evaluate the fidelity of parameterized processes are often measured at a finer resolution than is represented in the model and then must be scaled up for the evaluation. In other cases, numerical models may neglect or simplify system processes because the data is not available or the physical mechanisms are not fully known. If sufficient agreement between prediction and observation is not achieved, it is challenging to know whether these types of modeling choices are the cause, or if other issues, such as deficiencies in the input parameters and/or other modeling assumptions are the true cause of errors. Even if there is high agreement between prediction and observation, the model cannot necessarily be considered validated, as it is always possible that the right values were produced for the wrong reasons. For example, low error can stem from a situation where different errors in underlying assumptions or parameters cancel each other out ("compensatory errors"). Furthermore, coupled human-natural system models are often subject to "equifinality", a situation where multiple parameterized formulations can produce similar outputs or equally acceptable representations of the observed data. There is therefore no uniquely "true" or validated model, and the common practice of selecting "the best" deterministic calibration set is more of an assumption than a finding [18, 19]. The situation becomes even more tenuous when observational data is limited in its scope and/or quality to be insufficient to distinguish model representations or their performance differences.

These limitations on model verification undermine any purely positivist treatment of model validity: that a model should correctly and precisely represent reality to be valid. Under this perspective, closely related to empiricism, statistical tests should be used to compare the model's output with observations and only through empirical verification can a model or theory be deemed credible. A criticism to this viewpoint (besides the aforementioned challenges for model verification) is that it reduces the justification of a model to the single criterion of predictive ability and accuracy [20]. Authors have argued that this ignores the explanatory power held in models and other procedures, which can also advance scientific knowledge [21]. These views gave rise to relativist perspectives of science, which instead place more value on model utility in terms of fitness for a specific purpose or inquiry, rather than representational accuracy and predictive ability [22]. This viewpoint appears to be most prevalent among practitioners seeking decision-relevant insights (i.e., inspire new views vs. predict future conditions). The relativist perspective argues for the use of models as heuristics that can enhance our understanding and conceptions of system behaviors or possibilities [23]. In contrast, natural sciences favor a positivist perspective, emphasizing similarity between simulation and observation even in application contexts where it is clear that projections are being made for conditions that have never been observed and the system of focus will have evolved structurally beyond the model representation being employed (e.g., decadal to centennial evolution of human-natural systems).

These differences in prevalent perspectives are mirrored in how model validation is defined by the two camps: From the relativist perspective, validation is seen as a process of incremental "confidence building" in a model as a mechanism for insight [24], whereas in natural sciences validation is framed as a way to classify a model as having an acceptable representation of physical reality [17]. Even though the relativist viewpoint does not dismiss the importance of representational accuracy, it does place it within a larger process of establishing confidence through a variety of tools. These tools, not necessarily quantitative, include communicating information between practitioners and modelers, interpreting a multitude of model outputs, and contrasting preferences and viewpoints.

On the technical side of the argument, differing views on the methodology of model validation appear as early as in the 1960's. Naylor and Finger [25] argue that model validation should not be limited to a single metric or test of performance (e.g., a single error metric), but should rather be extended to multiple tests that reflect different aspects of a model's structure and behavior. This and similar arguments are made in literature to this day [13, 26, 27, 28, 29] and are primarily founded on two premises. First, that even though modelers widely recognize that their models are abstractions of the truth, they still make truth claims based on traditional performance metrics that measure the divergence of their model from observation [29]. Second, that the natural systems mimicked by the models contain many processes that exhibit significant heterogeneity at various temporal and spatial scales. This heterogeneity is lost when a single performance measure is used, as a result of the inherent loss of process information occurring

when transitioning from a highly dimensional and interactive system to the dimension of a single metric [16]. These arguments are further elaborated in Section 4.

Multiple authors have proposed that the traditional reliance on single measures of model performance should be replaced by the evaluation of several model signatures (characteristics) to identify model structural errors and achieve a sufficient assessment of model performance [13, 30, 31, 32]. There is however a point of departure here, especially when models are used to produce inferences that can inform decisions. When agencies and practitioners use models of their systems for public decisions, those models have already met sufficient conditions for credibility (e.g., acceptable representational fidelity), but may face broader tests on their salience and legitimacy in informing negotiated decisions [23, 33, 34]. This presents a new challenge to model validation, that of selecting decision-relevant performance metrics, reflective of the system's stakeholders' viewpoints, so that the most consequential uncertainties are identified and addressed [35]. For complex multisector models at the intersection of climatic, hydrologic, agricultural, energy, or other processes, the output space is made up of a multitude of states and variables, with very different levels of salience to the system's stakeholders and to their goals being achieved [36]. This is further complicated when such systems are also institutionally and dynamically complex. As a result, a broader set of qualitative and quantitative performance metrics is necessary to evaluate models of such complex systems, one that embraces the plurality of value systems, agencies and perspectives present. For IM3, even though the goal is to develop better projections of future vulnerability and resilience in co-evolving human-natural systems and not to provide decision support per se, it is critical for our multisector, multiscale model evaluation processes to represent stakeholders' adaptive decision processes credibly.

As a final point, when a model is used in a projection mode, its results are also subject to additional uncertainty, as there is no guarantee that the model's functionality and predictive ability will stay the same as the baseline, where the verification and validation tests were conducted. This challenge requires an additional expansion of the scope of model evaluation: a broader set of uncertain conditions needs to be explored, spanning beyond historical observation and exploring a wide range of unprecedented conditions. This perspective on modeling, termed exploratory [37], views models as computational experiments that can be used to explore vast ensembles of potential scenarios to identify those with consequential effects. Exploratory modeling literature explicitly orients experiments toward stakeholder consequences and decision-relevant inferences and shifts the focus from predicting future conditions to *discovering* which conditions lead to undesirable or desirable consequences.

This evolution in modeling perspectives can be mirrored by the IM3 family of models in a progression from evaluating models relative to observed history to advanced formalized analyses to make inferences on multisector, multiscale vulnerabilities and resilience. Exploratory modeling approaches can help fashion experiments with large numbers of alternative hypotheses on the co-evolutionary dynamics of influences, stressors, as well as path-dependent changes in the form and function of human-natural systems [38]. The aim of this text is to therefore guide the reader through the use of sensitivity analysis (SA) methods across these perspectives on diagnostic and exploratory modeling.

Note: The following articles are suggested as fundamental reading for the information presented in this section:

- Naomi Oreskes, Kristin Shrader–Frechette, and Kenneth Belitz. Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences. *Science*, 263 (5147): 641-646, February 1994. URL: https://science.sciencemag.org/content/263/5147/641. DOI: https://doi.org/10.1126/science.263.5147.641.
- Keith Beven. Towards a coherent philosophy for modelling the environment. *Proceedings of the royal society of London*. Series A: mathematical, physical and engineering sciences, 458 (2026): 2465-2484, 2002.
- Eker, S., Rovenskaya, E., Obersteiner, M., Langan, S., 2018. Practice and perspectives in the validation of resource management models. *Nature Communications* 9, 1–10. https://doi.org/10.1038/s41467-018-07811-9

The following articles can be used as supplemental reading: * Canham, C.D., Cole, J.J., Lauenroth, W.K. (Eds.), 2003. Models in Ecosystem Science. Princeton University Press. https://doi.org/10.2307/j.ctv1dwq0tq

---

CHAPTER

THREE

SENSITIVITY ANALYSIS: THE BASICS

3.1 Global Versus Local Sensitivity

Out of the several definitions for sensitivity analysis presented in literature, the most widely used has been proposed by Saltelli *et al.* [39] as "the study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input". In other words, sensitivity analysis explores the relationship between the model's N input variables, $x = [x_1, x_2, ..., x_N]$, and M output variables, $y = [y_1, y_2, ..., y_M]$ with $y = g(x)$, where g is the model that maps the model inputs to the outputs [40]. Therefore, sensitivity analysis provides us with a set of alternatives to conducting large empirical experiments which are costly and often, in practice, next to impossible.

Historically, there have been two broad categories of sensitivity analysis techniques: local and global. Local sensitivity analysis is performed by varying model parameters around specific reference values, with the goal of exploring how small input perturbations influence model performance. Due to its convenience, this approach has been widely used in literature, but has important limitations [41, 42]. If the model is not linear, the results of local sensitivity analysis can be heavily biased, as they will vary depending on the range of the chosen input (e.g., Tang *et al.* [43]). If the model's factors interact, local sensitivity analysis will underestimate their importance, as it does not account for those effects (e.g., [44]). In general, as local sensitivity analysis only partially and locally explores the parametric space, it is not considered a valid approach for nonlinear models [45]. This is illustrated in Fig. 3.1 (a-b), presenting contour plots of a model response (y) with an additive linear model (a) and with a nonlinear model (b). In a linear model without interactions between the terms x_1 and x_2, local sensitivity analysis can produce appropriate sensitivity indices (Fig. 3.1 (a)). If however, factors x_1 and x_1 interact, the local and partial consideration of the space can not properly account for each factor's effects on the model response (Fig. 3.1 (b)), as it is only informative at the base point where it is applied. In contrast, a global sensitivity analysis varies uncertain factors within the entire feasible space of variability (Fig. 3.1 (c)). This approach reveals the global effects of each parameter on the model output, including any interactive effects. For models that cannot be proven linear, global sensitivity analysis is preferred and this text is primarily discussing global sensitivity analysis methods. In general, whenever we use the term sensitivity analysis we are referring to its global application.

3.2 Why Perform Sensitivity Analysis

It is important to understand the many ways in which a SA might be of use to your modeling effort which can help shape the framing of model-informed study. Most commonly, one might be motivated to perform sensitivity analysis for the following reasons:

Model evaluation: Sensitivity analysis can be used to gauge model inferences when assumptions about the structure of the model or its parameterization are dubious or have changed. For instance, consider a numerical model that uses a set of calibrated parameter values to produce outputs, which we then use to inform decisions about the real-world system represented. One might like to know if small changes in these parameter values significantly change this model's output and the decisions it informs or if, instead, our parameter inferences yield stable model behavior regardless of

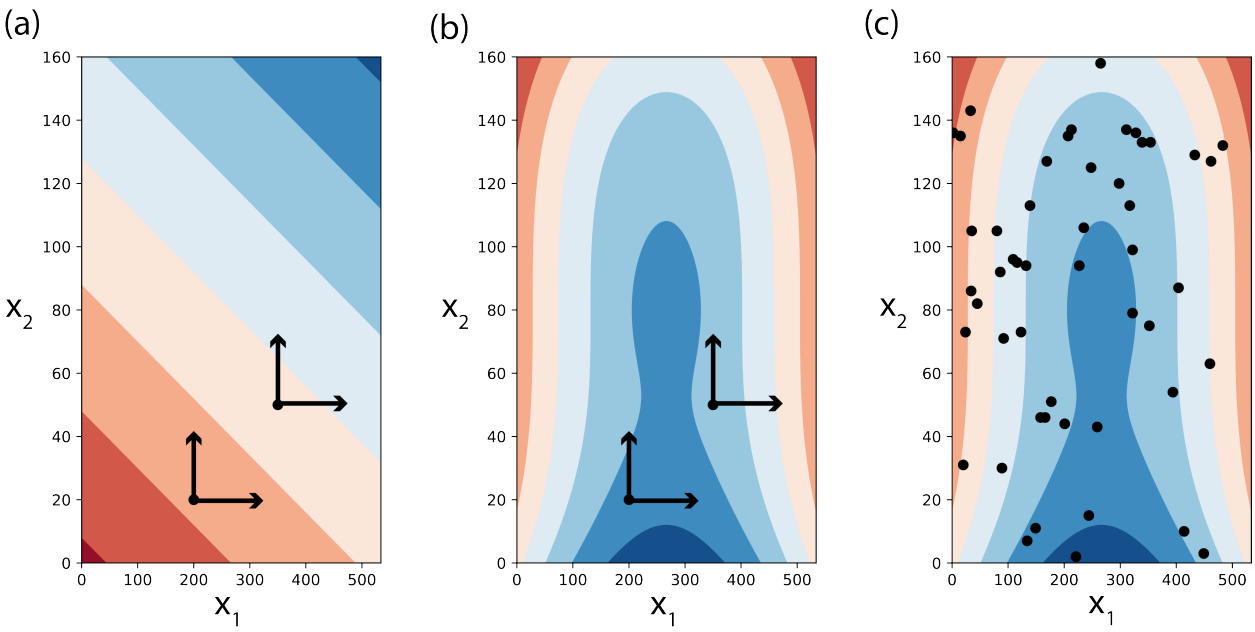


Fig. 3.1: Treatment of a two-dimensional space of variability by local (panels a-b) and global (panel c) sensitivity analyses. Local sensitivity analysis is only an appropriate approach to sensitivity in the case of linear models without interactions between terms, for example in panel (a), where $y = 3x_1 + 5x_2$. In the case of more complex models, for example in panels (b-c), where $y = \frac{1}{e^{x_1^2+x_2^2}} + \frac{50}{e^{(0.1x_1)^2+(0.1x_2)^3}}$ local sensitivity will miscalculate sensitivity indices (b), and global sensitivity methods should be used instead (c). The points in panel (c) are generated using a uniform random sample of $n = 50$, but many other methods are available.

the uncertainty present in the specific parameterized processes or properties. This can either discredit or lend credence to the model at hand, as well as any inferences drawn that are founded on its accurate representation of the system. Sensitivity analysis can identify which uncertain model factors cause this undesirable model behavior.

Model simplification: Sensitivity analysis can also be used to identify factors or components of the model that appear to have limited effects on direct outputs or metrics of interest. Consider a model that has been developed in an organization for the purposes of a specific research question and is later used in the context of a different application. Some processes represented in significant detail might no longer be of the same importance while consuming significant data or computational resources, as different outputs might be pertinent to the new application. Sensitivity analysis can be used to identify unimportant model components and simplify them to nominal values and reduced model forms. Model complexity and computational costs can therefore be reduced, and, by extension, monetary investments.

Model refinement: Alternatively, sensitivity analysis can reveal the factors or processes that are highly influential to the outputs or metrics of interest, by assessing their relative importance. In the context of model evaluation, this can inform which model components warrant additional investigation or measurement so the uncertainty surrounding them and the resulting model outputs or metrics of interest can be reduced.

Exploratory modeling: When sufficient credence has been established in the model, sensitivity analysis can be applied to a host of other inquiries. Inferences about the factors and processes that most (or least) control a model's outputs of interest can be extrapolated to the real system they represent and be used in a heuristic manner to inform model-based inferences. On this foundation, a model paired with the advanced techniques presented in this text can be used to "discover" decision relevant and highly consequential outcomes (i.e., scenario discovery, [37, 46]).

The nature and context of the model shapes the specific objectives of applying a sensitivity analysis, as well as methods and tools most appropriate and defensible for each application setting [36, 39, 47]. The three most common sensitivity analysis modes (*Factor Prioritization*, *Factor Fixing*, and *Factor Mapping*) are presented below, but the reader should be aware that other uses have been proposed in the literature (e.g., [48, 49]).

Factor prioritization: This sensitivity analysis application mode (also referred to as *factor ranking*) refers to when one would like to identify the uncertain factors which, when fixed to their true value (i.e., if there were no uncertainty regarding their value), would lead to the greatest reduction in output variability [50]. Information from this type of analysis can be crucial to model improvement as these factors can become the focus of future measurement campaigns or numerical experiments so that uncertainty in the model output can be reduced. The impact of each uncertain input on the variance of the model output is often used as the criterion for factor prioritization. Fig. 3.2 (a) shows the effects of three uncertain variables (X_1, X_2, and X_3) on the variance of output Y. $V(E(Y|X_i))$ indicates the variance in Y if factor X_i is left to vary freely while all other factors remain fixed to nominal values. In this case, factor X_2 makes the largest contribution to the variability of output Y and it should therefore be prioritized. In the context of risk analysis, factor prioritization can be used to reduce output variance to below a given tolerable threshold (also known as variance cutting).

Factor fixing: This mode of sensitivity analysis (also referred to as *factor screening*) aims to identify the model components that have a negligible effect or make no significant contributions to the variability of the outputs or metrics of interest (usually referred to as non-influential [50]). In the stylized example of Fig. 3.2 (a), X_1 makes the smallest contribution to the variability of output Y suggesting that the uncertainty in its value could be negligible and the factor itself fixed in subsequent model executions. Eliminating these factors or processes in the model or fixing them to a nominal value can help reduce model complexity as well as the unnecessary computational burden of subsequent model runs, results processing, or other sensitivity analyses (the fewer uncertain factors considered, the fewer runs are necessary to illuminate their effects on the output). Significance of the outcome can be gauged in a variety of manners, depending on the application. For instance, if applying a variance-based method, a minimum threshold value of contribution to the variance could be considered as a significance 'cutoff', and factors with indices below that value can be considered non-influential. Nb: conclusions about factor fixing should be made based on total-order effects, i.e., considering all the effects a factor has, individually and in interaction with other factors (explained in more detail in the Section 3.4.5).

Factor mapping: Finally, factor mapping can be used to pinpoint which values of uncertain factors lead to model outputs within a given range of the output space [50]. In the context of model diagnostics, it is possible that the model's output changes in ways considered impossible based on the represented processes, or other observed evidence. In this situation, factor mapping can be used to identify which uncertain model factors cause this undesirable model behavior by 'filtering' model runs that are considered 'non-behavioral' [51, 52, 53]. In Fig. 3.2 (b), region B of the output space denotes the set of behavioral model outcomes, which can be traced back to input space X (e.g., using Monte Carlo Filtering or pre-calibration, see Section 7).

The language used above reflects a use of sensitivity analysis for model fidelity evaluation and refinement. However, as previously mentioned, when a model has been established as a sufficiently accurate representation of the system, sensitivity analysis can produce additional inferences. For instance, under the factor mapping use, the analyst can now focus on undesirable system states and discover which factors are most responsible for them: for instance, "population growth of above 25% would be responsible for unacceptably high energy demands". Factor prioritization and factor fixing can be used to make equivalent inferences, such as "growing populations and increasing temperatures are the leading factors for changing energy demands" (prioritizing of factors) or "changing dietary needs are inconsequential to increasing energy demands for this region" (a factor that can be fixed in subsequent model runs). All these inferences hinge on the assumption that the real system's stakeholders consider the model states faithful enough representations of system states. As elaborated in Section 2.2, this view on sensitivity analysis is founded on a relativist perspective on modeling, which tends to place more value on model usefulness rather than strict accuracy of representation in terms of error. As such, sensitivity analysis performed with decision-making relevance in mind will focus on model outputs or metrics that are consequential and decision relevant (e.g., energy demand in the examples above).

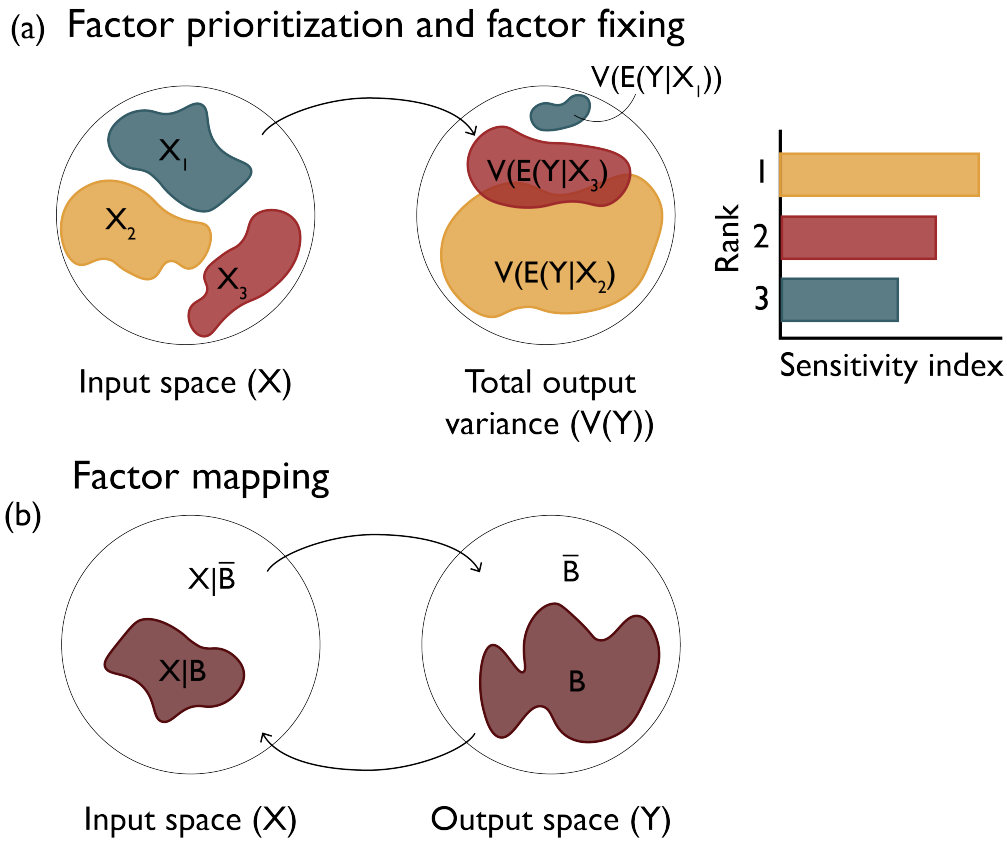


Fig. 3.2: Factor prioritization, factor fixing and factor mapping settings of sensitivity analysis.

3.3 Design of Experiments

Before conducting a sensitivity analysis, the first element that needs to be clarified is the uncertainty space of the model [52, 54]. In other words, how many and which factors making up the mathematical model are considered uncertain and can potentially affect the model output and the inferences drawn from it. Uncertain factors can be model parameters, model structures, inputs, or alternative model resolution levels (scales), all of which can be assessed through the tools presented in this text. Depending on the kind of factor, its variability can be elicited through various means: expert opinion, values reported in the literature, historical observations, its physical meaning (e.g., population values in a city can never be negative), or through the use of more formal UQ methods (Section 7). The model uncertainty space represents the entire space of variability present in each of the uncertain factors of a model. The complexity of most real-world models means that the response function, $y = g(x)$, mapping inputs to outputs, is hardly ever available in an analytical form and therefore analytically computing the sensitivity of the output to each uncertain factor becomes impossible. In these cases, sensitivity analysis is only feasible through numerical procedures that employ different strategies to sample the uncertainty space and calculate sensitivity indices.

A sampling strategy is often referred to as a *design of experiments* and represents a methodological choice made before conducting any sensitivity analysis. Experimental design was first introduced by Fisher [55] in the context of laboratory or field-based experiments. Its application in sensitivity analysis is similar to setting up a physical experiment in that it is used to discover the behavior of a system under specific conditions. An ideal design of experiments should provide a framework for the extraction of all plausible information about the impact of each factor on the output of the numerical model. The design of experiments is used to set up a simulation platform with the minimum computational cost to answer specific questions that cannot be readily drawn from the data through analytical or common data mining techniques. Models representing coupled human-natural systems usually have a large number of inputs, state variables and parameters, but not all of them exert fundamental control over the numerical process, despite their uncertainty, nor have substantial impacts on the model output, either independently or through their interactions. Each factor influences the model output in different ways that need to be discovered. For example, the influence of a parameter on model output can be linear or non-linear and can be continuous or only be active during specific times or at particular states of the system [56, 57]. An effective and efficient design of experiments allows the analyst to explore these complex relationships and evaluate different behaviors of the model for various scientific questions [58].

There are a few different approaches to the design of experiments, closely related to the chosen sensitivity analysis approach, which is in turn shaped by the research motivations, scientific questions, and computational constraints at hand (additional discussion of this can be found at the end of Section 3). For example, in a sensitivity analysis using perturbation and derivatives methods, the model input parameters vary from their nominal values one at a time, something that the design of experiments needs to reflect. If, instead, one were to perform sensitivity analysis using a multiple-starts perturbation method, the design of experiments needs to consider that multiple points across the factor space are used. The design of experiments specifically defines two key characteristics of samples that are fed to the numerical model: the number of samples and the range of each factor.

Generally, sampling can be performed randomly or by applying a stratifying approach. In random sampling, such as Monte Carlo [59], samples are randomly generated by a pseudo-random number generator with an a-priori assumption about the distribution of parameters and their possible ranges. Random seeds can also be used to ensure consistency and higher control over the random process. However, this method could leave some holes in the parameter space and cause clustering in some spaces, especially for a large number of parameters [60]. Most sampling strategies use stratified sampling to mitigate these disadvantages. Stratified sampling techniques divide the domain of each factor into subintervals, often of equal lengths. From each subinterval, an equal number of samples is drawn randomly, or based on the specific locations within the subintervals [50]. The rest of this section overviews some of the most commonly used designs of experiments. Table 1 summarizes the designs discussed.

Table 3.1: Summary of designs of experiments overviewed in this section.
* Depends on the sample size.

Design of experiments	*Factor interactions considered*	*Treatment of factor domains*
One-At-a-Time (OAT)	No - main effects only	Continuous (distributions)
Full Factorial Sampling	Yes - including total effects	Discrete (levels)
Fractional Factorial Sampling	Yes - only lower-order effects*	Discrete (levels)
Latin Hypercube (LH) Sampling	Yes - including total effects*	Continuous (distributions)
Quasi-Random Sampling with Low-Discrepancy Sequences	Yes - including total effects*	Continuous (distributions)

3.3.1 One-At-a-Time (OAT)

In this approach, only one model factor is changed at a time while all others are kept fixed across each iteration in a sampling sequence. The OAT method assumes that model factors of focus are linearly independent (i.e., there are no interactions) and can analyze how factors individually influence model outputs or metrics of interest. While highly popular given its ease of implementation, OAT is ultimately highly limited in its exploration of a model's sensitivities [50]. It is primarily used with local sensitivity techniques with similar criticisms: applying this sampling scheme on a system with nonlinear and interactive processes will miss important information on the effect uncertain factors have on the model. OAT samplings can be repeated multiple times in a more sophisticated manner and across different locations of the parameter space to overcome some of these challenges, which would increase computational costs and negate the main reasons for its selection.

3.3.2 Full and Fractional Factorial Sampling

In full factorial sampling, each factor is treated as being discrete, by considering two or more levels (or intervals). The sampling process then generates samples within each possible combination of levels, corresponding to each parameter. This scheme produces a more comprehensive sampling of the factors' variability space, as it accounts for all candidate combinations of factor levels (Fig. 3.3 (a)). If the number of levels is the same across all factors, the number of generated samples is estimated using nk, where n is the number of levels and k is the number of factors. For example, Fig. 3.3 (a) presents a full factorial sampling of three uncertain factors (x_1, x_2, and x_3), each considered as having four discrete levels. The total number of samples necessary for such an experiment is $4^3 = 64$. As the number of factors increases, the number of simulations necessary can also grow exponentially, making full factorial sampling computationally burdensome (Fig. 3.3 (b)). As a result, literature has commonly applied full factorial sampling at only two levels per factor, typically the two extremes [61]. This significantly reduces computational burden but is only considered appropriate in cases where factors can indeed only assume two discrete values (e.g., when testing the effects of epistemic uncertainty and comparing between model structure A and model structure B). In the case of physical parameters on continuous distributions (e.g., when considering the effects of measurement uncertainty in a temperature sensor), discretizing the range of a factor to only extreme levels can bias its estimated importance.

Fractional factorial design is a widely used alternative to full factorial that allows the analyst to significantly reduce the number of simulations by confounding the main effects of a factor with its interactive effects [50]. In other words, if one can reasonably assume that higher-order interactions are negligible, information about the main effects and lower-order interactions can be obtained using a fraction of the full factorial design. Traditionally, fractional factorial design has also been limited to two levels [61], referred to as Fractional Factorial designs 2k-p [62]. Recently, Generalized Fractional Factorial designs have also been proposed that allow for the structured generation of samples at more than two levels per factor [63]. Consider a case where the modeling team dealing with the problem in Fig. 3.3 (a) cannot afford to perform 64 simulations of their model. They can afford 32 runs for their experiment and instead decide to fractionally sample the variability space of their factors. A potential design of such a sampling strategy is presented in Fig. 3.3 (c).

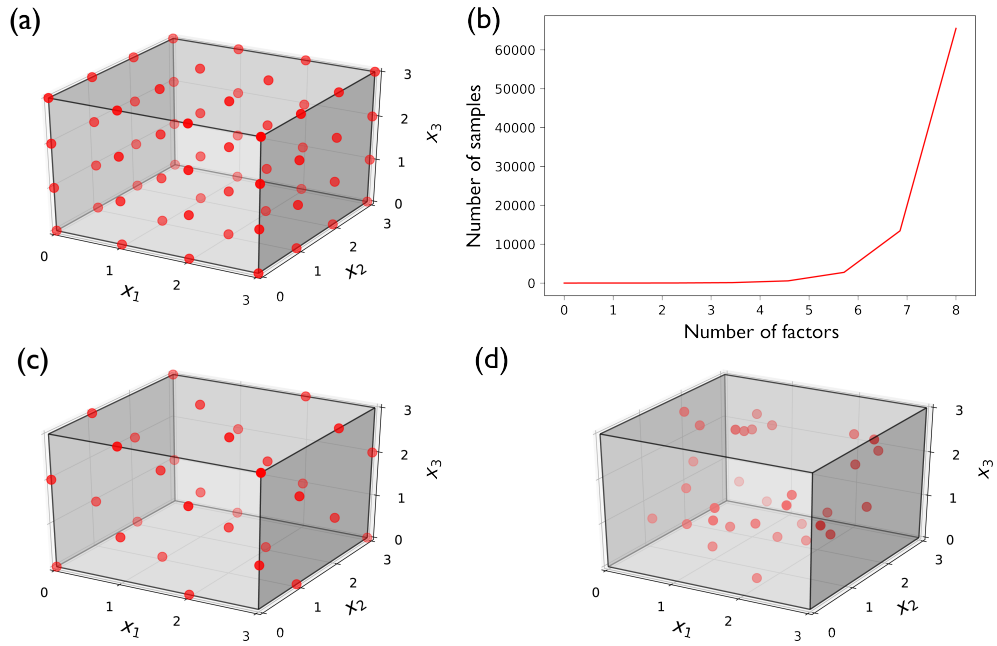


Fig. 3.3: Alternative designs of experiments and their computational costs for three uncertain factors (x_1, x_2, and x_3). (a) Full factorial design sampling of three factors at four levels, at a total of 64 samples; (b) exponential growth of necessary number of samples when applying full factorial design at four levels; (c) fractional factorial design of three factors at four levels, at a total of 32 samples; and (d) Latin Hypercube sample of three factors with uniform distributions, at a total of 32 samples.

3.3.3 Latin Hypercube (LH) Sampling

Latin hypercube sampling [64] is one of the most common methods in space-filling experimental designs. With this sampling technique, for N uncertain factors, an N-dimensional hypercube is generated, with each factor divided into an equal number of levels depending on the sample to be generated. Equal numbers of samples are then randomly generated at each level, across all factors. In this manner, LH design guarantees sampling from every level of the variability space and without any overlaps. When the number of samples generated is much larger than the number of uncertain factors, LH sampling can be very effective in examining the effects of each factor [50]. LH sampling is an attractive technique, because it guarantees a diverse coverage of the space, through the use of subintervals, without being constrained to discrete levels for each factor - compare Fig. 3.3 (c) with Fig. 3.3 (d) for the same number of samples.

LH sampling is less effective when the number of samples is not much larger than the number of uncertain factors, and the effects of each factor cannot be appropriately distinguished. The samples between factors can also be highly correlated, biasing any subsequent sensitivity analysis results. To address this, the sampling scheme can be modified to control for the correlation in parameters while maximizing the information derived. An example of such modification is through the use of orthogonal arrays [65].

3.3.4 Low-Discrepancy Sequences

Low-discrepancy sequences is another sampling technique that employs a pseudo-random generator for Monte Carlo sampling [66, 67]. These quasi-Monte Carlo methods eliminate potential gaps and clusters between samples by minimizing discrepancy when generating uniformly distributed random samples within the hypercube. The discrepancy property is mathematically measured by characterizing the lumpiness of a sequence of samples in a multidimensional space, which results in evenly distributed samples [66]. Discrepancy can be quantitatively measured using the deviations of sampled points from the uniform distribution [68]. Low-discrepancy sequences ensure that the number of samples in any subspace of the variability hypercube is approximately the same. This is not something guaranteed by LH sampling, and even though the design can be improved through optimization with various criteria, such adjustments are limited to small sample sizes and low dimensions [68, 69, 70, 71, 72]. The Sobol sequence [73, 74], one of the most widely used sampling techniques, utilizes the low-discrepancy approach to uniformly fill the sampled factor space. A core advantage of this style of sampling is that it takes far fewer samples (i.e., simulations) to attain a much lower level of error in estimating model output statistics (e.g., the mean and variance of outputs).

Note: Put this into practice! Click the following link to try out an interactive tutorial on implementing a Sobol SA using SALib: Sobol SA using SALib Jupyter Notebook

---

3.3.5 Other types of sampling

The sampling techniques mentioned so far are general sampling methods useful for a variety of applications beyond sensitivity analysis. There are however techniques that have been developed for specific sensitivity analysis methods. Examples of these methods include the Morris One-At-a-Time [75], Fourier Amplitude Sensitivity Test (FAST; [76]), Extended FAST [77], and Extended Sobol methods [78]. For example, the Morris sampling strategy builds a number of trajectories (usually referred to as repetitions and denoted by r) in the input space each composed of $N+1$ factor points, where N is the number of uncertain factors. The first point of the trajectory is selected randomly and the subsequent N points are generated by moving one factor at a time by a fixed amount. Each factor is perturbed once along the trajectory, while the starting points of all of the trajectories are randomly and uniformly distributed. Several variations of this strategy also exist in the literature; for more details on each approach and their differences the reader is directed to Pianosi *et al.* [52].

3.3.6 Synthetic generation of input time series

Models often have input time series or processes with strong temporal and/or spatial correlations (e.g., streamflow, energy demand, price of commodities, etc.) that, while they might not immediately come to mind as factors to be examined in sensitivity analysis, can be treated as such. Synthetic input time series are used for a variety of reasons, for example, when observations are not available or are limited, or when past observations are not considered sufficiently representative to capture rare or extreme events of interest [79, 80]. Synthetic generation of input time series provides a valuable tool to consider non-stationarity and incorporate potential stressors, such as climate change impacts into input time series [81]. For example, a century of record will be insufficient to capture very high impact rare extreme events (e.g., persistent multi-year droughts). A large body of statistical literature exists focusing on the topics of synthetic weather [82, 83] and streamflow [84, 85] generation that provides a rich suite of approaches for developing history-informed, well-characterized stochastic process models to better estimate rare individual or compound (hot, severe drought) extremes. It is beyond the scope of this text to review these methods, but readers are encouraged to explore the studies cited above as well as the following publications for discussions and comparisons of these methods: [79, 81, 86, 87, 88, 89, 90]. The use of these methods for the purposes of exploratory modeling, especially in the context of well-characterized versus deep uncertainty, is further discussed in Section 4.3.

3.4 Sensitivity Analysis Methods

In this section, we describe some of the most widely applied sensitivity analysis methods along with their mathematical definitions. We also provide a detailed discussion on applying each method, as well as a comparison of and their features and limitations.

3.4.1 Derivative-based Methods

Derivative-based methods explore how model outputs are affected by perturbations in a single model input around a particular input value. These methods are local and are performed using OAT sampling. For simplicity of mathematical notations, let us assume that the model $g(X)$ only returns one output. Following [91] and [52], the sensitivity index, S_i , of the model's *i*-th input factor, x_i , can be measured using the partial derivative evaluated at a nominal value, $\bar{x}$, of the vector of inputs:

$$S_i(\bar{x}) = \frac{\partial g}{\partial x}|_{\bar{x}^{c}{}_i}$$

where c_i is the scaling factor. In most applications however, the relationship $g(X)$ is not fully known in its analytical form, and therefore the above partial derivative is usually approximated:

$$S_i(\bar{x}) = \frac{g(\bar{x}_1, ...\bar{x}_i + \Delta_i, ...\bar{x}_N) - g(\bar{x}_1, ...\bar{x}_i, ...\bar{x}_N)}{\Delta_i} c_i$$

Using this approximation, the *i*-th input factor is perturbed by a magnitude of Δ_i, and its relative importance is calculated. Derivative-based methods are some of the oldest sensitivity analysis methods as they only require $N + 1$ model evaluations to estimate indices for N uncertain factors. As described above, being computationally very cheap comes at the cost of not being able to explore the entire input space, but only (local) perturbations to the nominal value. Additionally, as these methods examine the effects of each input factor one at a time, they cannot assess parametric interactions or capture the interacting nature of many real systems and the models that abstract them.

3.4.2 Elementary Effect Methods

Elementary effect (EE) SA methods provide a solution to the local nature of the derivative-based methods by exploring the entire parametric range of each input parameter [92]. However, EE methods still use OAT sampling and do not vary all input parameters simultaneously while exploring the parametric space. The OAT nature of EEs methods therefore prevents them from properly capturing the interactions between uncertain factors. EEs methods are computationally efficient compared to their All-At-a-Time (AAT) counterparts, making them more suitable when computational capacity is a limiting factor, while still allowing for some inferences regarding factor interactions. The most popular EE method is the Method of Morris [75]. Following the notation by [52], this method calculates global sensitivity using the mean of the EEs (finite differences) of each parameter at different locations:

$$S_i = \mu_i^* = \frac{1}{r}\sum_{j=1}^{r} EE_i^j = \frac{1}{r}\sum_{j=1}^{r} \frac{g(\bar{x}_1, ...\bar{x}_i + \Delta_i, ...\bar{x}_N) - g(\bar{x}_1, ...\bar{x}_i, ...\bar{x}_N)}{\Delta_i} c_i$$

with r representing the number of repetitions (or trajectories) in the input space, usually set between 4 and 10 [39]. Each xj represents the points of each trajectory, with $j = 1, \ldots, r$, selected as described in the sampling strategy for this method, found above. This method also produces the standard deviation of the EEs:

$$\sigma_i = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(EE_i^j - \frac{1}{r}\sum_{j=1}^{r} EE_i^j)^2}$$

which is a measure of parametric interactions. Higher values of σ_i suggest model responses at different levels of factor x_i are significantly different, which indicates considerable interactions between that and other uncertain factors. The

values of μ_i^* and σ_i for each factor allow us to draw several different conclusions, illustrated in Fig. 3.4, following the example by [92]. In this example, factors x_1, x_2, x_4, and x_5 can be said to have an influence on the model outputs, with x_1, x_4, and x_5 having some interactive or non-linear effects. Depending on the orders of magnitude of μ_i^* and σ_i one can indirectly deduce whether the factors have strong interactive effects, for example if a factor $\sigma_i << \mu_i^*$ then the relationship between that factor and the output can be assumed to be largely linear (nb: this is still an OAT method and assumptions on factor interactions should be strongly caveated). Extensions of the Method of Morris have also been developed specifically for the purposes of factor fixing and explorations of parametric interactions (e.g., [49, 93, 94]).

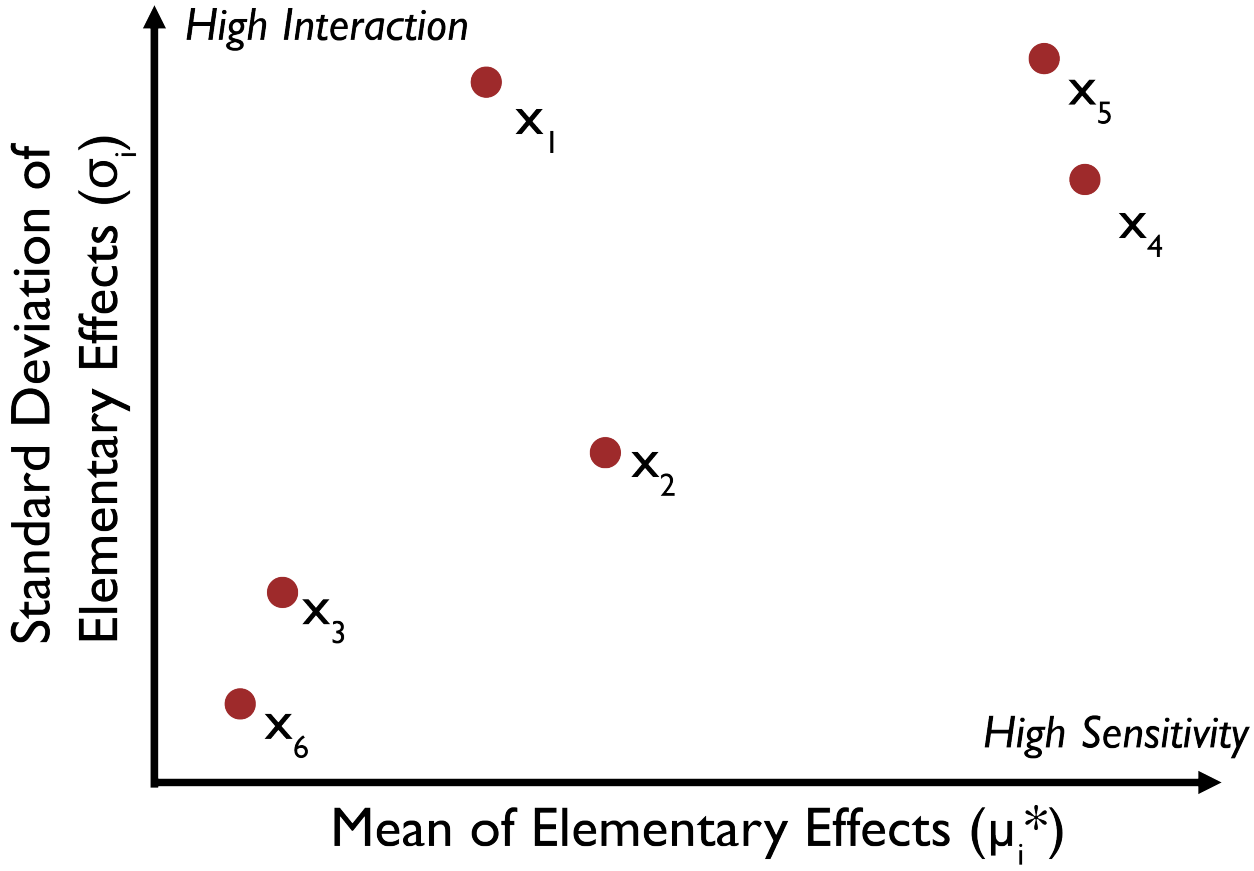


Fig. 3.4: Illustrative results of the Morris Method. Factors x_1, x_2, x_4, and x_5 have an influence on the model outputs, with x_1, x_4, and x_5 having interactive or non-linear effects.

3.4.3 Regression-based Methods

Regression analysis is another one of the oldest ways of investigating parametric importance and sensitivity [39]. Here, we describe some of the most popular regression-based sensitivity indices. One of the main sensitivity indices of this category is the standardized regression coefficient (SRC). To calculate SRC, a linear regression relationship needs to be fitted between the input vector, x, and the model output of interest by using a least-square minimizing method:

$$y = b_0 + \sum_{i=1}^{N} b_i x_i$$

where b_0 and b_i (corresponding to the i-th model input) are regression coefficients. The following relationship can then be used to calculate the SRCs for different input values:

$$S_i = SRC_i = b_i \frac{\sigma_i}{\sigma_y}$$

where σ_i and σ_y are standard deviations of i-th model input and output, respectively.

Several other regression-based indices explore the correlation between input and output parameters as a proxy to model parametric sensitivity [92, 95, 96]. The Pearson correlation coefficient (PCC) can be used when a linear relationship exists between an uncertain factor, x_i, and the output y:

$$S_i = PCC = \frac{cov(x_i, y)}{\sigma_i \sigma_y}$$

In cases when there are outliers in the data or the relationship between the uncertain factors and the output is not linear, rank-based correlation coefficients are preferred, for example, Spearman's rank correlation coefficient (SRCC):

$$S_i = SRCC = \frac{cov(rx_i, ryi)}{\sigma_{ri} \sigma_{ry}}$$

where the raw values of x_i and y and converted to ranks rx_i and ry respectively, which instead represents a measurement of the strength of the monotonic relationship, rather than linear relationship, between the input and output. Other regression-based metrics include the partial correlations coefficient, the partial rank correlations coefficient, and the Nash-Sutcliffe coefficient, more discussion on which can be found in [40, 92].

Tree-based regression techniques have also been used for sensitivity analysis in an effort to address the challenges faced with nonlinear models [97]. Examples of these methods include the Patient Rule Induction Method (PRIMl; [98]) and Classification And Regression Trees (CART; [99]). CART-based approaches also include their boosting and bagging extensions [100, 101]. These methods are particularly useful when sensitivity analysis is used for factor mapping (i.e., when trying to identify which uncertain model factors produce a certain model behavior). Section 4.3 elaborates on the use of these methods. Regression-based sensitivity analysis methods are global by nature and can explore the entire space of variables. However, the true level of comprehensiveness depends on the design of experiments and the number of simulations providing data to establish the regression relationships. Although they are usually computationally efficient, they do not produce significant information about parametric interactions [39, 40].

3.4.4 Regional Sensitivity Analysis

Another method primarily applied for basic factor mapping applications is Regional Sensitivity Analysis (RSA; [102]). RSA is a global sensitivity analysis method that is typically implemented using standard sampling methods such as LH sampling. It is performed by specifying a condition on the output space (e.g., an upper threshold) and classifying outputs that meet the condition as behavioral and the ones that fail it as non-behavioral (illustrated in Fig. 3.2 (b)). Note that the specified threshold depends on the nature of the problem, model, and the research question. It can reflect model-performance metrics (such as errors) or consequential decision-relevant metrics (such as unacceptable system outcomes). The behavioral and non-behavioral outputs are then traced back to their originating sampled factor sets, where differences between the distributions of samples can be used to determine their significance in producing each part of the output. The Kolmogorov-Smirnov divergence is commonly used to quantify the difference between the distribution of behavioral and non-behavioral parameters [52].

$$S_i = |F_{x_i|y_b}(y \in Y_b) - F_{x_i|y_{nb}}(y \in Y_{nb})|$$

where Y_b represents the set of behavioral outputs, and $F_{x_i|y_b}$ is the empirical cumulative distribution function of the values of x_i associated with values of y that belong in the behavioral set. The nb notation indicates the equivalent elements related to the non-behavioral set. Large differences between the two distributions indicate stronger effects by the parameters on the respective part of the output space.

Used in a factor mapping setting, RSA can be applied for scenario discovery [103, 104], the Generalized Likelihood Uncertainty Estimation method (GLUE; [19, 105, 106]) and other hybrid sensitivity analysis methods (e.g., [107, 108]). The fundamental shortcomings of RSA are that, in some cases, it could be hard to interpret the difference between behavioral and non-behavioral sample sets, and that insights about parametric correlations and interactions cannot always be uncovered [39]. For more elaborate discussions and illustrations of the RSA method, readers are directed to Tang *et al.* [43], Saltelli *et al.* [50], Young [109] and references therein.

3.4.5 Variance-based Methods

Variance-based sensitivity analysis methods hypothesize that various specified model factors contribute differently to the variation of model outputs; therefore, decomposition and analysis of output variance can determine a model's sensitivity to input parameters [39, 78]. The most popular variance-based method is the Sobol method, which is a global sensitivity analysis method that takes into account complex and nonlinear factor interaction when calculating sensitivity indices, and employs more sophisticated sampling methods (e.g., the Sobol sampling method). The Sobol method is able to calculate three types of sensitivity indices that provide different types of information about model sensitivities. These indices include first-order, higher-order (e.g., second-, third-, etc. orders), and total-order sensitivities.

The first-order sensitivity index indicates the percent of model output variance contributed by a factor individually (i.e., the effect of varying x_i alone) and is obtained using the following [78, 110].

$$S_i^1 = \frac{V_{x_i}[E_{x_{\sim i}}(x_i)]}{V(y)}$$

with E and V denoting the expected value and the variance, respectively. $x_{\sim i}$ denotes all factors expect from x_i. The first-order sensitivity index (S_i^1) can therefore also be thought of as the portion of total output variance (V_y) that can be reduced if the uncertainty in factor x_i is eliminated [111]. First-order sensitivity indices are usually used to understand the independent effect of a factor and to distinguish its individual versus interactive influence. It would be expected for linearly independent factors that they would only have first order indices (no interactions) that should correspond well with sensitivities obtained from simpler methods using OAT sampling.

Higher-order sensitivity indices explore the interaction between two or more parameters that contribute to model output variations. For example, a second-order index indicates how interactions between a pair of parameter input variables can lead to change in model output variations and is calculated using the following relationship:

$$S_i^2 = \frac{V_{x_{i,j}}[E_{x_{\sim i,j}}(x_i, x_j)]}{V(y)}$$

with $i \neq j$. Higher order indices can be calculated by similar extensions (i.e., fixing additional operators together), but it is usually computationally expensive in practice. There are some software packages that calculate indices for orders higher than second; for example, the "senssobol" R package calculates third-order indices (more information can be found in Section 3.6).

The total sensitivity analysis index represents the entire influence of an input factor on model outputs including all of its interactions with other factors [112]. In other words, total SA indices include first-order and all higher-order interactions associated with each factor and can be estimated calculated using the following:

$$S_i^T = \frac{E_{x_{\sim i}}[V_{x_i}(x_{\sim i})]}{V(y)} = 1 - \frac{V_{x_{\sim i}}[E_{x_i}(x_{\sim i})]}{V(y)}$$

This index reveals the expected portion of variance that remains if uncertainty is eliminated in all factors but x_i [111]. The total sensitivity index is the overall best measure of sensitivity as it captures the full individual and interactive effects of model factors.

Besides the Sobol method, there are some other variance-based sensitivity analysis methods, such as the Fourier amplitude sensitivity test (FAST; [76, 113]) and extended-FAST [114, 115], that have been used by the scientific community. However, Sobol remains by far the most common method of this class. Variance-based techniques have been widely used and have proved to be powerful in a variety of applications. Despite their popularity, some authors have expressed concerns about the methods' appropriateness in some settings. Specifically, the presence of heavy-tailed distributions or outliers, or when model outputs are multimodal can bias the sensitivity indices produced by these methods [116, 117, 118]. Moment-independent measures, discussed below, attempt to overcome these challenges.

Note: Put this into practice! Click the following badge to try out an interactive tutorial on performing a sensitivity analysis to discover important factors: Factor Discovery Jupyter Notebook

3.4.6 Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) was first introduced by Fisher and others [119] and has since become a popular factor analysis method in physical experiments. ANOVA can be used as a sensitivity analysis method in computational experiments (referred to as factorial ANOVA) with a factorial design of experiment. Note that Sobol can also be categorized as an ANOVA sensitivity analysis method, and that is why Sobol is sometimes referred to as a functional ANOVA [120]. Factorial ANOVA methods are particularly suited for models and problems that have discrete input spaces, significantly reducing the computational time. More information about these methods can be found in [120, 121, 122].

3.4.7 Moment-Independent (Density-Based) Methods

These methods typically compare the entire distribution (i.e., not just the variance) of input and output parameters in order to determine the sensitivity of the output to a particular input variable. Several moment-independent sensitivity analysis methods have been proposed in recent years. The delta (δ) moment-independent method calculates the difference between unconditional and conditional cumulative distribution functions of the output. The method was first introduced by [123, 124] and has become widely used in various disciplines. The δ sensitivity index is defined as follows:

$$S_i = \delta_i = \frac{1}{2} E_{x_i} |f_y(y) - f_{y|x_i}(y)| dy$$

where $f_y(y)$ is the probability density function of the entire model output y, and $f_{y|x_i}(y)$ is the conditional density of y, given that factor x_i assumes a fixed value. The δ_i sensitivity indicator therefore represents the normalized expected shift in the distribution of y provoked by x_i. Moment-independent methods are advantageous in cases where we are concerned about the entire distribution of events, such as when uncertain factors lead to more extreme events in a system [14]. Further, they can be used with a pre-existing sample of data, without requiring a specific sampling scheme, unlike the previously reviewed methods [125]. The δ sensitivity index does not include interactions between factors and it is therefore akin to the first order index produced by the Sobol method. Interactions between factors can still be estimated using this method, by conditioning the calculation on more than one uncertain factor being fixed [124].

3.5 How To Choose A Sensitivity Analysis Method: Model Traits And Dimensionality

Fig. 3.5 presents a graphical synthesis of the methods overviewed in this section, with regards to their appropriateness of application based on the complexity of the model at hand and the computational limits on the number of model evaluations afforded. The bars below each method also indicate the sensitivity analysis purposes they are most appropriate to address, which are in turn a reflection of the motivations and research questions the sensitivity analysis is called to address. Computational intensity is measured as a multiple of the number of model factors that are considered uncertain (d). Increasing model complexity mandates that more advanced sensitivity analysis methods are applied to address potential nonlinearities, factor interactions and discontinuities. Such methods can only be performed at increasing computational expense. For example, computationally cheap linear regression should not be used to assess factors' importance if the model cannot be proven linear and the factors independent, because important relationships will invariably be missed (recall the example in Fig. 3.5). When computational limits do constrain applications to make simplified assumptions and sensitivity techniques, any conclusions in such cases should be delivered with clear statements of the appropriate caveats.

The reader should also be aware that the estimates of computational intensity that are given here are indicative of magnitude and would vary depending on the sampling technique, model complexity and the level of information being asked. For example, a Sobol sensitivity analysis typically requires a sample of size $n * d + 2$ to produce first- and total-order indices, where d is the number of uncertain factors and n is a scaling factor, selected ad hoc, depending on model complexity [47]. The scaling factor n is typically set to at least 1000, but it should most appropriately be set on the basis of index convergence. In other words, a prudent analyst would perform the analysis several times with

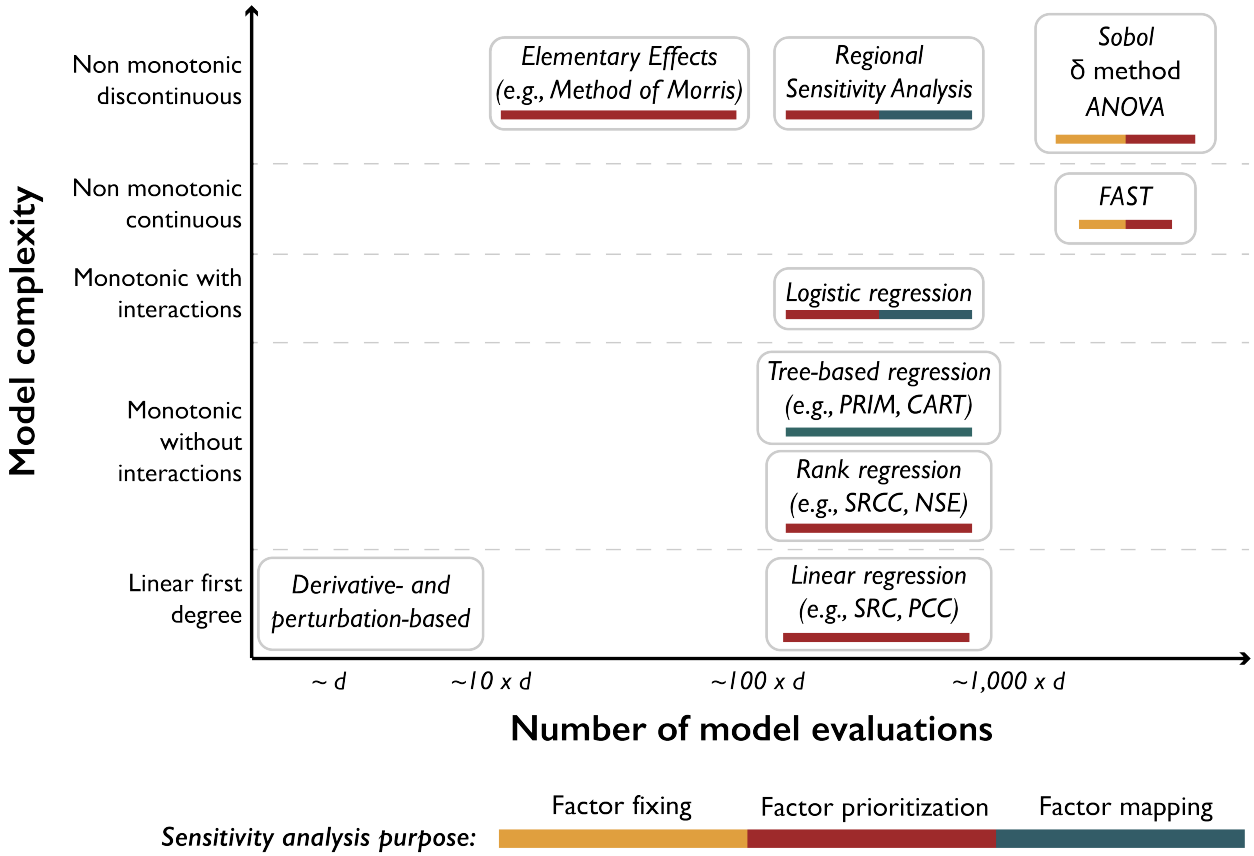


Fig. 3.5: Classification of the sensitivity analysis methods overviewed in this section, with regards to their computational cost (horizontal axis), their appropriateness to model complexity (vertical axis), and the purpose they can be used for (colored bars). d: number of uncertain factors considered; ANOVA: Analysis of Variance; FAST: Fourier Amplitude Sensitivity Test; PRIM: Patient Rule Induction Method; CART: Classification and Regression Trees; SRCC: Spearman's rank correlation coefficient: NSE: Nash–Sutcliffe efficiency; SRC: standardized regression coefficient; PCC: Pearson correlation coefficient

increasing n and observe at what level the indices converge to stable values [126]. The level should be the minimum sample size used in subsequent sensitivity analyses of the same system. Furthermore, if the analyst would like to better understand the degrees of interaction between factors, requiring second-order indices, the sample size would have to increase to $n * 2d + 2$ [47].

Another important consideration is that methods that do not require specific sampling schemes, can be performed in conjunction with others without requiring additional model evaluations. None of the regression-based methods, for example, require samples of specific structures or sizes, and can be combined with other methods for complementary purposes. For instance, one could complement a Sobol analysis with an application of CART, using the same data, but to address questions relating to factor mapping (e.g., we know factor x_i is important for a model output, but we would like to also know which of its values specifically push the output to undesirable states). Lastly, comparing results from different methods performed together can be especially useful in model diagnostic settings. For example, [14] used δ indices, first-order Sobol indices, and R^2 values from linear regression, all performed on the same factors, to derive insights about the effects on factors on different moments of the output distribution and about the linearity of their relationship.

3.6 Software Toolkits

This section presents available open source sensitivity analysis software tools, based on the programming language they use and the methods they support Fig. 3.6. Our review covers five widely used programming languages: R, MATLAB, Julia, Python, and C++, as well as one tool that provides a graphical user interface (GUI). Even though there are other commercial GUI-based SA tools available, only open source tools were considered within the scope of this overview. Each available SA tool was assessed on the number of SA methods and design of experiments methods it supports. For example, the *sensobol* package in R only supports the variance-based Sobol method. However, it is the only package we came across that calculates third-order interactions among parameters. On the other side of the spectrum, there are SA software packages that contain several popular SA methods. For example, *SALib* in Python [127] supports seven different SA methods. The *DifferentialEquations* package is a comprehensive package developed for Julia, and *GlobalSensitivityAnalysis* is another Julia package that has mostly adapted SALib methods. Fig. 3.6 also identifies the SA packages that have been updated since 2018, indicating active support and development.

Note: The following articles are suggested as fundamental reading for the information presented in this section:

- Wagener, T., Pianosi, F., 2019. What has Global Sensitivity Analysis ever done for us? A systematic review to support scientific advancement and to inform policy-making in earth system modelling. *Earth-Science Reviews* 194, 1–18. https://doi.org/10.1016/j.earscirev.2019.04.006
- Pianosi, F., Beven, K., Freer, J., Hall, J.W., Rougier, J., Stephenson, D.B., Wagener, T., 2016. Sensitivity analysis of environmental models: A systematic review with practical workflow. Environmental Modelling & Software 79, 214–232. https://doi.org/10.1016/j.envsoft.2016.02.008

The following articles can be used as supplemental reading: * Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S., 2008. Global Sensitivity Analysis: The Primer, 1st edition. ed. Wiley-Interscience, Chichester, England; Hoboken, NJ. * Montgomery, D.C., 2017. Design and analysis of experiments. John Wiley & Sons. * Iooss, B., Lemaître, P., 2015. A Review on Global Sensitivity Analysis Methods, in: Dellino, G., Meloni, C. (Eds.), Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications, Operations Research/Computer Science Interfaces Series. Springer US, Boston, MA, pp. 101–122. https://doi.org/10.1007/978-1-4899-7547-8_5

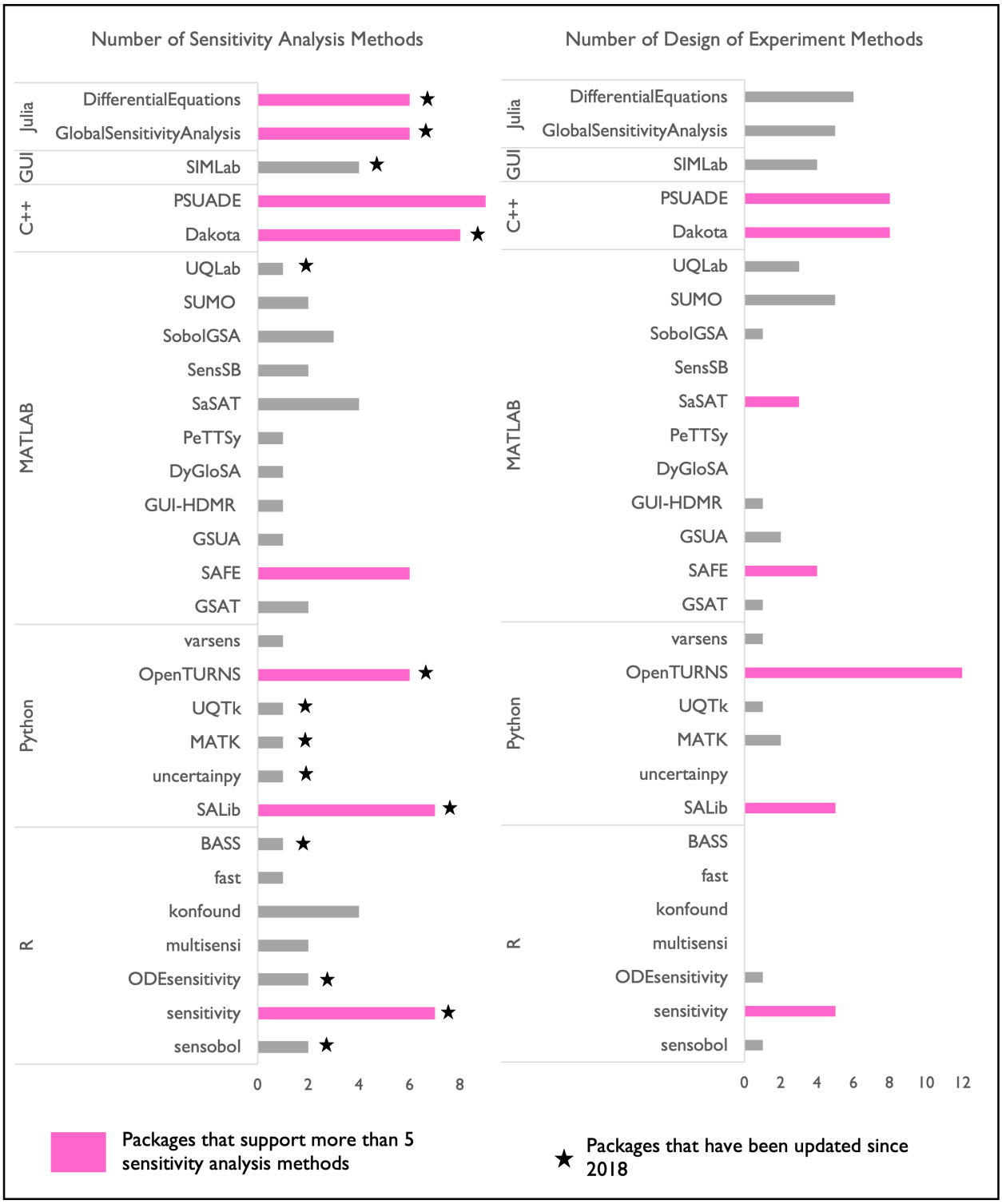


Fig. 3.6: Sensitivity analysis packages available in different programming language platforms (R, Python, Julia, MATLAB, and C++), with the number of methods they support. Packages supporting more than five methods are indicated in pink. Packages updated since 2018 are indicated with asterisks.

CHAPTER

FOUR

SENSITIVITY ANALYSIS: DIAGNOSTIC & EXPLORATORY MODELING

4.1 Understanding Errors: What Is Controlling Model Performance?

Sensitivity analysis is a diagnostic tool when reconciling model outputs with observed data. It is helpful for clarifying how and under what conditions modeling choices (structure, parameterization, data inputs, etc.) propagate through model components and manifest in their effects on model outputs. This exploration is performed through carefully designed perturbation of multiple combinations of input parameters and subsequent evaluation of the model structures that are emerging as controlling factors. Model structure and parameterization are two of the most commonly explored aspects of models that have been a strong focus when evaluating their performance relative to available observations [18]. Addressing these issues plays an important role in establishing credibility in model predictions, particularly in the positivist natural sciences literature. Traditional model evaluations compare the model with observed data, and then rely on expert judgements of its acceptability based on the closeness between simulation and observation with a single metric. However, this approach can be strongly myopic and misleading [128], as it is often impossible to use one metric to attribute a certain error and to link that with different parts of the model and its parameters. This means that, even when the error or fitting measure between the model estimates and observations is very small, it is not guaranteed that all the components in the model perfectly represent the conceptual reality that the model is abstracting: propagated errors in different parts of the model might cancel out each other, or there might be equifinality [18]. This also implies that the opposite situation does not prove that the model is completely wrong either.

The inherent complexity of a system hinders accepting or rejecting a model based on one performance measure and different types of measures can potentially evaluate components of the model [26, 27] as essentially a multiobjective problem. In addition, natural systems mimicked by the models contain various interacting components that might act differently across the spatial domain and time periods [52, 56]. This heterogeneity is lost when a single performance measure is used, as a result of the inherent loss of process information occurring when transitioning from a highly dimensional and interactive system to a highly aggregated averaging of spatial or temporal output errors [16]. One specific measure might be a good representative of a phenomena at a particular condition (e.g., time of year or location), but not at a different state of the system.

Therefore, diagnostic error analyses should consider multiple error signatures across different scales and states of concern when seeking to understand how our theoretical conceptions relate to observed data (Fig. 4.1). Diverse error signatures can be used to measure the consistency of underlying processes and behaviors of the model and to evaluate the dynamics of model controls under changing temporal and spatial conditions [129]. Within this framework, even minimal information extracted from the data is beneficial as it helps us to understand the complexity of the model which potentially unearths structural inadequacies in the model, and even the underlying data itself. In this context, proper selection of measures of model performance and the number of measures could play consequential roles in our understanding of the model and its predictions [130].

As discussed earlier, instead of the traditional focus on using deterministic prediction that results in a single error measure, many plausible states and spaces could be searched for making different inferences and quantifying uncertainties. This process also requires estimates of prior probability distributions of all the important parameters and quantification of model behavior across input space. One strategy to reduce the search space is filtering of some model alternatives that are not consistent with observations and known system behaviors. Those implausible parts of the search space

can be referred to as non-physical or non-behavioral alternatives [51, 105]. This step is conducted before the Bayesian calibration exercise (see Section 7).

A comprehensive model diagnostic workflow typically entails the components demonstrated in Fig. 4.1. The workflow begins with the selection of model input parameters and their plausible ranges. After the parameter selection, we need to specify the design of experiment (Section 3.3) and the sensitivity analysis method (Section 3.4) to be used. As previously discussed, these methods require different numbers of model simulations, and each method provides a different insights into the direct effects and interactions of the parameters. In addition, the simulation time of the model and the available computational resources are two of the primary considerations that influence these decisions. After identifying the appropriate methods, we generate a matrix of input parameters, where each set of input parameters will be used to conduct a model simulation. The model can include one or more output variables that fluctuate in time and space. The next step is to analyze model performance by comparing model outputs with observations. As discussed earlier, the positivist model evaluation paradigm focuses on a single model performance metric (error), leading to a loss of information about model parameters and the suitability of the model's structure. However, a thorough investigation of the temporal and spatial signatures of model outputs using various performance metrics or time- and space-varying sensitivity analyses can shed more light on the fitness of each parameter set and the model's internal structure. This analysis provides diagnostic feedback on the importance and range of model parameters and can guide further improvement of the model algorithm.

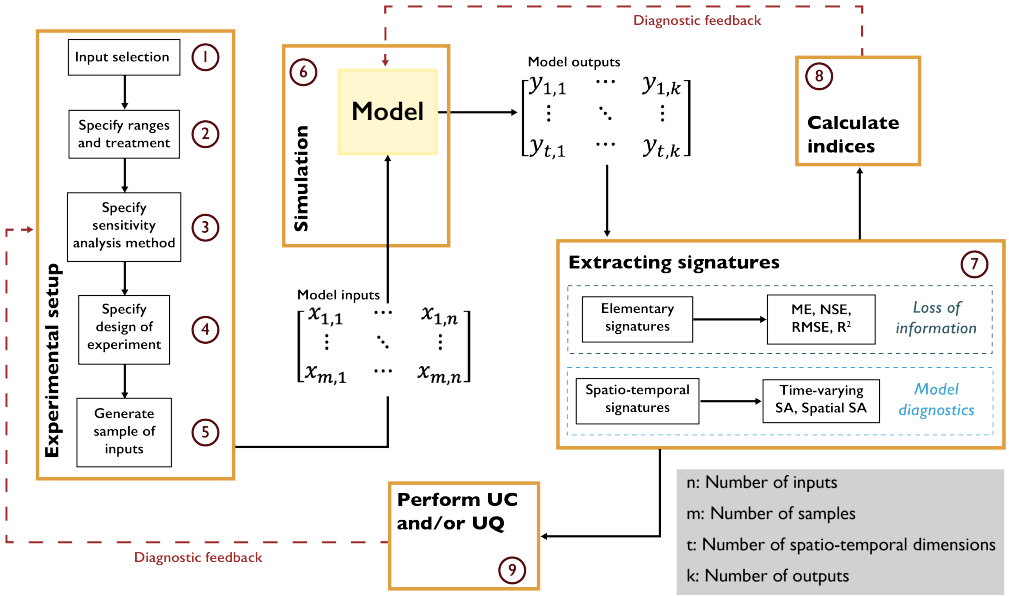


Fig. 4.1: Diagnostic evaluation of model fidelity using sensitivity analysis methods.

Note: Put this into practice! Click the following badge to try out an interactive tutorial on implementing a time-varying sensitivity analysis of HYMOD model parameters: HYMOD Jupyter Notebook

4.2 Consequential Dynamics: What is Controlling Model Behaviors of Interest?

Consequential changes in dynamic systems can take many forms, but most dynamic behavior can be categorized in a few basic patterns. Feedback structures inherent in a system, be they positive or negative, generate these patterns which can be grouped into three groups for the simplest of systems: exponential growth, goal seeking, and oscillation (Fig. 4.2). A positive (or self-reinforcing) feedback gives rise to exponential growth, a negative (or self-correcting) feedback gives rise to a goal seeking mode, and negative feedbacks with time delays give rise to oscillatory behavior. Nonlinear interactions between the system's feedback structures can give rise to more complex dynamic behavior modes, examples of which are also shown in Fig. 4.2, adapted from Sterman [131].

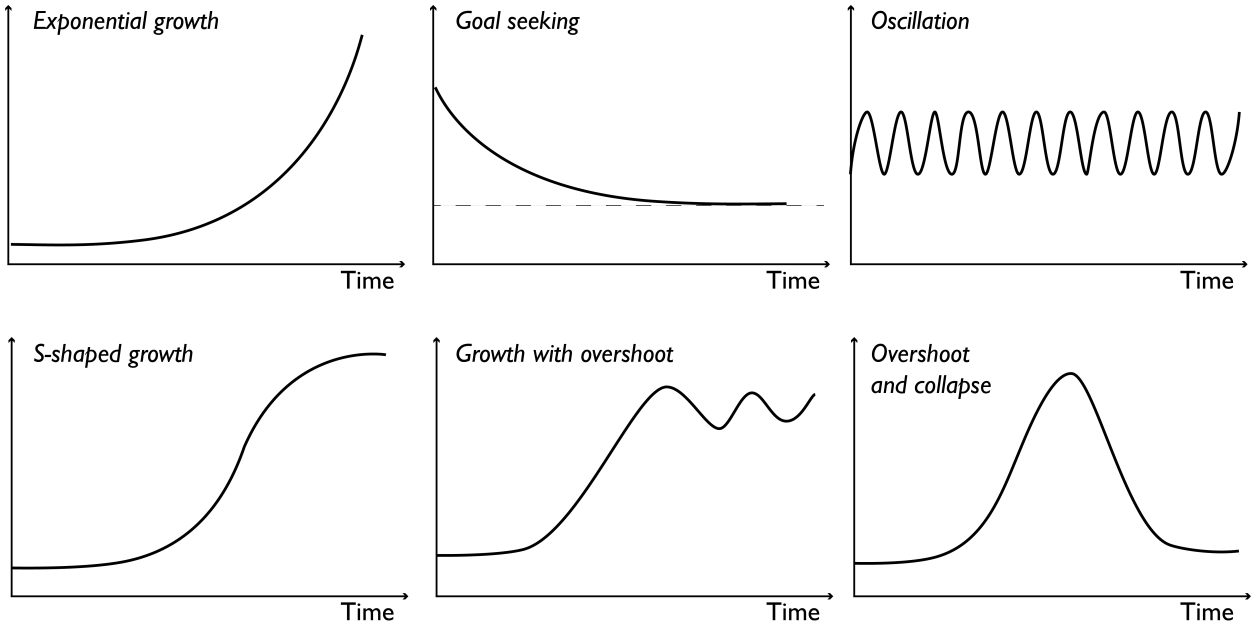


Fig. 4.2: Common modes of behavior in dynamic systems, occurring based on the presence of positive and negative feedback relationships, and linear and non-linear interactions. Adapted from Sterman [131].

The nature of feedback processes in a dynamic system shapes its fundamental behavior: positive feedbacks generate their own growth, negative feedbacks self-limit, seeking balance and equilibrium. In this manner, feedback processes give rise to different regimes, multiples of which could be present in each mode of behavior. Consider a population of mammals growing exponentially until it reaches the carrying capacity of its environment (referred to as S-shaped growth). When the population is exponentially growing, the regime is dominated by positive feedback relationships that reinforce its growth. As the population approaches its carrying capacity limit, negative feedback structures begin to dominate, counteracting the growth and establishing a stable equilibrium. Shifts between regimes can be thought of as tipping points, mathematically defined as unstable equilibria, where the presence of positive feedbacks amplifies disturbances and moves the system to a new equilibrium point. In the case of stable equilibria, the presence of negative feedbacks dampens any small disturbance and maintains the system at a stable state. As different feedback relationships govern each regime, different factors (those making up each feedback mechanism) are activated and shape the states the system is found in, as well as define the points of equilibria.

For simple stylized models with a small number of states, system dynamics analysis can analytically derive these equilibria, the conditions for their stability and the factors determining them. The ability for this to be performed is, however, significantly challenged when it comes to systems that attempt to more closely resemble real complex systems. We argue this is the case for several reasons. First, besides generally exhibiting complex nonlinear dynamics, real world systems are also made up from larger numbers of interacting elements, which often makes the analytic derivation of system characteristics intractable [132, 133]. Second, human-natural systems temporally evolve and transform when

human state-aware action is present. Consider, for instance, humans recreationally hunting the aforementioned population of mammals. Humans act based on the mammal population levels by enforcing hunting quotas or establishing protected territories or eliminating other predators. The mammal population reacts in a response, giving birth to ever changing state-action-consequence feedbacks, the path dependencies of which become difficult to diagnose and understand (e.g., [134]). Trying to simulate the combination of these two challenges (large numbers of state-aware agents interacting with a natural resource and with each other) produces intractable models that require advanced heuristics to analyze their properties and establish useful inferences.

Sensitivity analysis paired with exploratory modeling methods offers a promising set of tools to address these challenges. We present a simple demonstrative application based on Quinn *et al.* [135]. This stylistic example was first developed by Carpenter *et al.* [136] and represents a town that must balance its agricultural and industrial productivity with the pollution it creates in a downstream lake. Increased productivity allows for increased profits, which the town aims to maximize, but it also produces more pollution for the lake. Too much phosphorus pollution can cause irreversible eutrophication, a process known as "tipping" the lake. The model of phosphorus in the lake X_t at time t is governed by:

$$X_{t+1} = X_t + a_t + \frac{X_t^q}{1 + X_t^q} - bX_t + \varepsilon$$

where $a_t \in [0, 0.1]$ is the town's pollution release at each timestep, b is the natural decay rate of phosphorus in the lake, q defines the lake's recycling rate (primarily through sediments), and ε represents uncontrollable natural inflows of pollution modeled as a log-normal distribution with a given mean, μ, and standard deviation σ.

Panels (a-c) in Fig. 4.3 plot the fluxes of phosphorus into the lake versus the mass accumulation of phosphorus in the lake. The red line corresponds to the phosphorus sinks in the lake (natural decay), given by bX_t. The grey shaded area represents the lake's phosphorus recycling flux, given by $\frac{X_t^q}{1+X_t^q}$. The points of intersection indicate the system's equilibria, two of which are stable, and one is unstable (also known as the tipping point). The stable equilibrium in the bottom left of the figure reflects an oligotrophic lake, whereas the stable equilibrium in the top right represents a eutrophic lake. With increasing phosphorus values, the tipping point can be crossed, and the lake will experience irreversible eutrophication, as the recycling rate would exceed the removal rate even if the town's pollution became zero. In the absence of anthropogenic and natural inflows of pollution in the lake (a_t and ε respectively), the area between the bottom-left black point and the white point in the middle can be considered as the safe operating space, before emission levels cross the tipping point.

The town has identified two potential policies that can be used to manage this lake, one that maximizes its economic profits ("best economic policy") and one that maximizes the time below the tipping point ("most reliable policy"). Panels (b-c) in Fig. 4.3 add the emissions from these policies to the recycling flux and show how the equilibria points shift as a result. In both cases the stable oligotrophic equilibrium increases and the tipping point decreases, narrowing the safe operating space [132, 139]. The best economic policy results in a much narrower space of action, with the tipping point very close to the oligotrophic equilibrium. The performance of both policies depends significantly on the system parameters. For example, a higher value of b, the natural decay rate, would shift the red line upward, moving the equilibria points and widening the safe operating space. Inversely, a higher value of q, the lake's recycling rate, would shift the recycling line upward, moving the tipping point lower and decreasing the safe operating space. The assumptions under which these policies were identified are therefore critical to their performance and any potential uncertainty in the parameter values could be detrimental to the system's objectives being met.

Sensitivity analysis can be used to clarify the role these parameters play on policy performance. Fig. 4.3 (d) shows the results of a Sobol sensitivity analysis on the reliability of the "most reliable" policy in a radial convergence diagram. The significance of each parameter is indicated by the size of circles corresponding to it. The size of the interior dark circle indicates the parameter's first-order effects and the size of the exterior circle indicates the parameter's total-order effects. The thickness of the lines between two parameters indicated the extent of their interaction (second-order effects). In this case, parameters b and q appear to have the most significant importance on the system, followed by the mean, μ, of the natural inflows. All these parameters function in a manner that shifts the location of the three equilibria and therefore policies that are identified ignoring this parametric uncertainty might fail to meet their intended goals.

It is worth mentioning that current sensitivity analysis methods are somewhat challenged in addressing several system dynamics analysis questions. The fundamental reason is that sensitivity analysis methods and tools have been developed

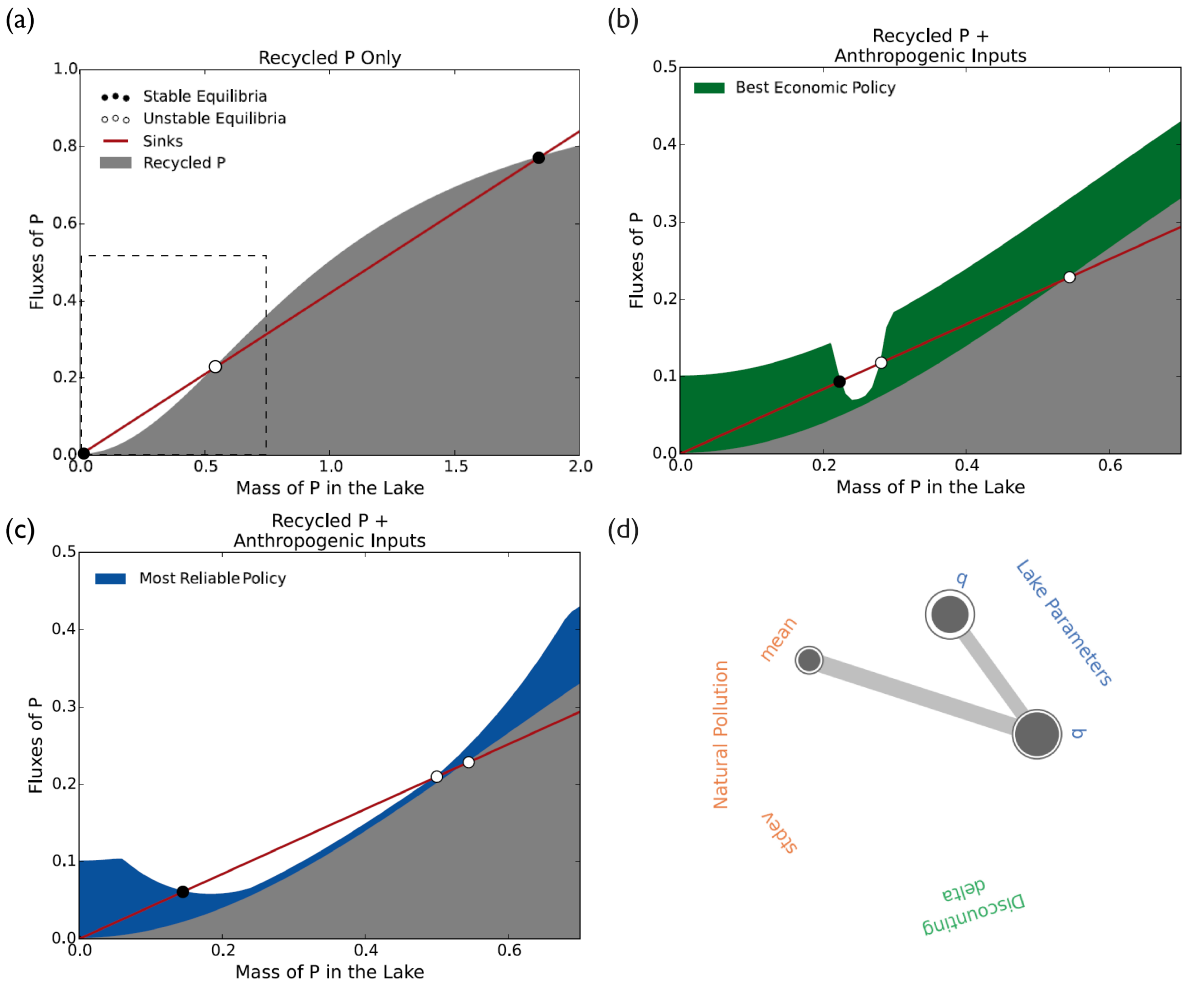


Fig. 4.3: Fluxes of phosphorus with regards to mass of phosphorus in the lake and sensitivity analysis results, assuming $b = 0.42$ and $q = 2$. (a) Fluxes of phosphorus assuming no emmisions policy and no natural inflows. (b-c) Fluxes phosphorus when applying two different emissions policies. The "Best economic policy" and the "Most realiable policy" have been identified by Quinn *et al.* [135] and can be found at Quinn [137]. (d) Results of a sensitivity analysis on the parameters of the model most consequential to the reliability of the "Most reliable policy". The code to replicate the sensitivity analysis can be found at Hadka [138].

to gauge numerical sensitivity of model output to changes in factor values. This is natural, as most simulation studies (e.g., all aforementioned examples) have been traditionally concerned with this type of sensitivity. In system dynamics modeling, however, a more important and pertinent concern is changes between regimes or between behavior modes (also known as bifurcations) as a result of changes in model factors [131, 140]. This poses two new challenges. First, identifying a change in regime depends on several characteristics besides a change in output value, like the rate and direction of change. Second, behavior mode changes are qualitative and discontinuous, as equilibria change in stability but also move in and out of existence.

Despite these challenges, recent advanced sensitivity analysis methods can help illuminate which factors in a system are most important in shaping boundary conditions (tipping points) between different regimes and determining changes in behavior modes. Reviewing such methods is outside the scope of this text, but the reader is directed to the examples of Eker *et al.* [23] and Hadjimichael *et al.* [134], who apply parameterised perturbation on the functional relationships of a system to study the effects of model structural uncertainty on model outputs and bifurcations, and Hekimoğlu and Barlas [140] and Steinmann *et al.* [141] who, following wide sampling of uncertain inputs, cluster the resulting time series in modes of behavior and identify most important factors for each.

4.3 Consequential Scenarios: What is Controlling Consequential Outcomes?

As overviewed in Section 2.2, most models are abstractions of systems in the real world. When sufficient confidence has been established in a model, it can then act as a surrogate for the actual system, in that the consequences of potential stressors, proposed actions or other changes can be evaluated by computer model simulations [37]. A model simulation then represents a computational experiment, which can be used to assess how the modeled system would behave should the various changes come to be. Steven Bankes coined the term exploratory modeling to describe the use of large sets of such computational experiments to investigate their implications on the system. Fig. 4.4 presents a typical workflow of an exploratory modeling application. Exploratory modeling approaches typically use sampling designs to generate large ensembles of states that represent combinations of changes happening together, spanning the entire range of potential values a factor might take (indicated in Fig. 4.4 by numbers 2-5). This perspective on modeling is particularly relevant to studies making long term projections into the future.

In the long-term policy analysis literature, exploratory modeling has prominently placed itself as an alternative to traditional narrative scenario or assumptions-based planning approaches, in what can be summarized in the following two-pronged critique [37, 142, 143]. The most prevalent criticism sees that the future and how it might evolve is both highly complex and deeply uncertain. Despite its benefits for interpretation and intuitive appeal, a small number of scenarios invariably misses many other potential futures that did not get selected as sufficiently representative of the future. This is especially the case for aggregate, narrative scenarios that describe simultaneous changes in multiple sectors together (e.g., "increased energy demand, combined with high agricultural land use and large economic growth"), such as the emission scenarios produced by the Intergovernmental Panel on Climate Change [144]. The bias introduced by this reduced set of potential changes can skew inferences drawn from the model, particularly when the original narrative scenarios are focused on a single or narrow set of measures of system behavior.

The second main criticism of traditional narrative scenario-based planning methods is that they provide no systematic way to distinguish which of the constituent factors lead to the undesirable consequences produced by a scenario. Narrative scenarios (e.g., the scenario matrix framework of RCPs-SSPs-SPAs; [145]) encompass multiple changes happening together selected to span the range of potential changes but are not typically generated in a systematic factorial manner that considers the multiple ways the factors can be combined. This has two critical limitations. It obfuscates the role each component factor plays in the system, both in isolation and in combination with others (e.g., "is it the increased energy demand or the high agricultural land use that cause unbearable water stress?"). It also renders the delineation of how much change in a factor is critical near impossible. Consider, for example, narrative scenario A with a 5% increase in energy demand, and scenario B with a 30% increase in energy demand, which would have dire consequences. At which point between 5% and 30% do the dire consequences actually begin to occur? Such questions cannot be answered without a wide exploration of the space of potential changes. It should be noted that for some levels of model complexity and computational demands (e.g., global-scale models) there is little feasible recourse beyond the

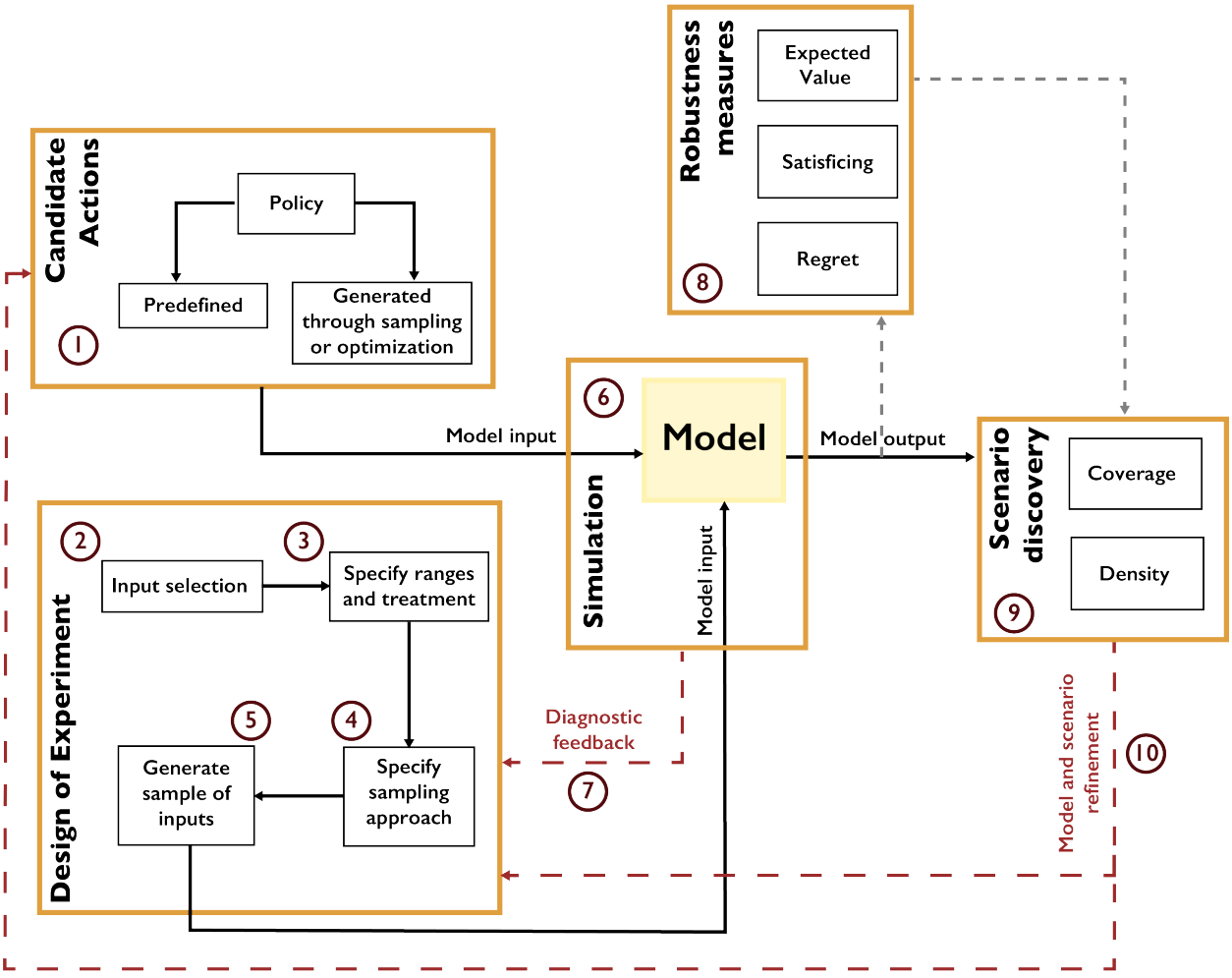


Fig. 4.4: A typical exploratory modeling workflow

use of narrative scenarios.

Exploratory modeling is typically paired with scenario discovery methods (indicated by number 9 in Fig. 4.4) that identify which of the scenarios (also known as states of the world) generated indeed have consequences of interest for stakeholders and policy makers, in an approach referred to as ensemble-based scenario-discovery [46, 103, 104]. This approach therefore flips the planning analysis from one that attempts to predict future system conditions to one that attempts to discover the (un)desirable future conditions. Ensemble-based scenario discovery can thus inform what modeling choices yield the most consequential behavioral changes or outcomes, especially when considering deeply uncertain, scenario-informed projections [8, 146]. The relative likelihoods and relevance of the discovered scenarios can be subsequently evaluated by the practitioners a posteriori, within a richer context of knowing the wider set of potential consequences [147]. This can include changing how an analysis is framed (number 10 in Fig. 4.4). For instance, one could initially focus on ensemble modeling of vulnerability using a single uncertain factor that is assumed to be well characterized by historical observations (e.g., streamflow; this step is represented by numbers 2-3 in Fig. 4.4). The analysis can then shift to include projections of more factors treated as deeply uncertain (e.g., urbanization, population demands, temperature, and snow-melt) to yield a far wider space of challenging projected futures. UC experiments contrasting these two framings can be highly valuable for tracing how vulnerability inferences change as the modeled space of futures expands from the historical baseline [135].

An important nuance to be clarified here is that the focus or purpose of a modeling exercise plays a major role in whether a given factor of interest is considered well-characterized or deeply uncertain. Take the example context of characterizing temperature or streamflow extremes, where for each state variable of interest for a given location of focus there is a century of historical observations. Clearly, the observation technologies will have evolved over time uniquely for temperature and streamflow measurements and they likely lack replicate experiments (data uncertainty). A century of record will be insufficient to capture very high impact and rare extreme events (i.e., increasingly poor structural/parametric inference for the distributions of specific extreme single or compound events). The mechanistic processes as well as their evolving variability will be interdependent but uniquely different for each of these state variables. A large body of statistical literature exists focusing on the topics of synthetic weather [82, 83] or streamflow [84, 85] generation that provides a rich suite of approaches for developing history-informed, well-characterized stochastic process models to better estimate rare individual or compound extremes. These history-focused approaches can be viewed as providing well-characterized quantifications of streamflow or temperature distributions; however, they do not capture how coupled natural-human processes can fundamentally change their dynamics when transitioning to projections of longer-term futures (e.g., streamflow and temperature in 2055). Consequently, changing the focus of the modeling to making long term projections of future streamflow or temperature now makes these processes deeply uncertain.

Scenario discovery methods (number 9 in Fig. 4.4) can be qualitative or quantitative and they generally attempt to distinguish futures in which a system or proposed policies to manage the system meet or miss their goals [104]. The emphasis placed by exploratory modeling on model outputs that have decision relevant consequences represents a shift toward a broader class of metrics that are reflective of the stakeholders' concerns, agency and preferences (also discussed in Section 2.2). As a result, sensitivity analysis and scenario discovery methods in this context are therefore applied to performance metrics that go beyond model error but are rather focused on broader metrics such as the resilience of a sector, the reliability of a process, or the vulnerability of a population in the face of uncertainty. In exploratory modeling literature, this metric is most typically—but not always—a measure of robustness (number 8 in Fig. 13). Robustness is a property of a system or a design choice capturing its insensitivity to uncertainty and can be measured via a variety of means, most recently reviewed by Herman *et al.* [148] and McPhail *et al.* [130].

Scenario discovery is typically performed through the use of algorithms applied on large databases of model runs, generated through exploratory modeling, with each model run representing the performance of the system in one potential state of the world. The algorithms seek to identify the combinations of factor values (e.g., future conditions) that best distinguish the cases in which the system does or does not meet its objectives. The most widely known classification algorithms are the Patient Rule Induction Method (PRIM; [98]) and Classification and Regression Trees (CART; [99]). These factor mapping algorithms create orthogonal boundaries (multi-dimensional hypercubes) between states of the world that are successful or unsuccessful in meeting the system's goals [66]. The algorithms attempt to strike a balance between simplicity of classification (and as a result, interpretability) and accuracy [46, 104, 149].

Even though these approaches have been shown to yield interpretable and relevant scenarios [150], several authors have

pointed out the limitations of these methods with regards to their division of space in orthogonal behavioral and non-behavioral regions [151]. Due to their reliance on boundaries orthogonal to the uncertainty axes, PRIM and CART cannot capture interactions between the various uncertain factors considered, which can often be significant [152]. More advanced methods have been proposed to address this drawback, with logistic regression being perhaps the most prominent [152, 153, 154]. Logistic regression can produce boundaries that are not necessarily orthogonal to each uncertainty axis, nor necessarily linear, if interactive terms between two parameters are used to build the regression model. It also describes the probability that a state of the world belongs to the scenarios that lead to failure. This feature allows users to define regions of success based on a gradient of estimated probability of success in those worlds, unlike PRIM which only classifies states of the world in two regions [152, 155].

Another more advanced factor mapping method is boosted trees [100, 156]. Boosted trees can avoid two limitations inherent to the application of logistic regression: i) to build a nonlinear classification model the interactive term between two uncertainties needs to be pre-specified and cannot be discovered (e.g., we need to know a-priori whether factor x_1 interacts with x_2 in a relationship that looks like $x_1 \cdot x_2$ or $x_1^{x_2}$); and ii) the subspaces defined are always convex. The application of such a factor mapping algorithm is limited in the presence of threshold-based rules with discrete actions in a modeled system (e.g., "if network capacity is low, build new infrastructure"), which results in failure regions that are nonlinear and non-convex [151]. Boosting works by creating an ensemble of classifiers and forcing some of them to focus on the hard-to-learn parts of the problem, and others to focus on the easy-to-learn parts. Boosting applied to CART trees can avoid the aforementioned challenges faced by other scenario discovery methods, while resisting overfitting [157], assuring the identified success and failure regions are still easy to interpret.

Below we provide an example application of two scenario discovery methods, PRIM and logistic regression, using the lake problem introduced in the previous section. From the sensitivity analysis results presented in Fig. 4.3 (d), we can already infer that parameters b and q have important effects on model outputs (i.e., we have performed factor prioritization). Scenario discovery (i.e., factor mapping) complements this analysis by further identifying the specific values of b and q that can lead to consequential and undesirable outcomes. For the purposes of demonstration, we can assume the undesirable outcome in this case is defined as the management policy failing to achieve 90% reliability in a state of the world.

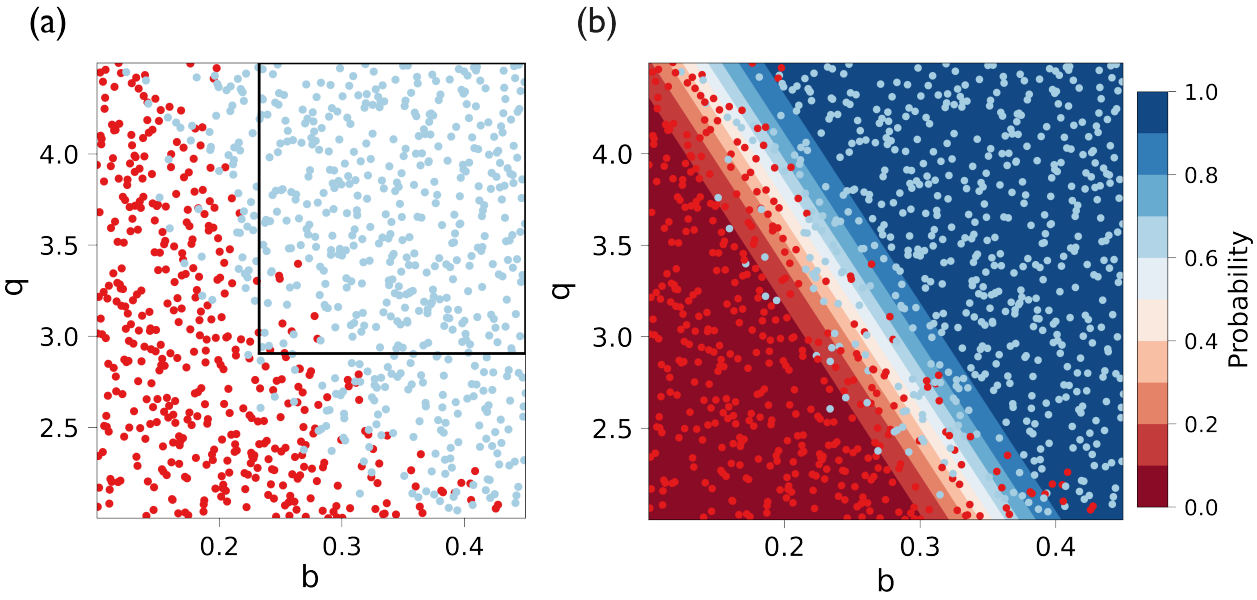


Fig. 4.5: Scenario discovery for the lake problem, using (a) PRIM and (b) logistic regression.

Fig. 4.5 shows the results of scenario discovery, performed through (a) PRIM and (b) logistic regression. Each point in the two panels indicates a potential state of the world, generated through Latin Hypercube Sampling. Each point is colored by whether the policy meets the above performance criterion, with blue indicating success and red indicating failure. PRIM identifies several orthogonal areas of interest, one of which is shown in panel (a). As discussed above, this necessary orthogonality limits how PRIM identifies areas of success (the area within the box). As factors b and

q interact in this system, the transition boundary between the regions of success and failure is not orthogonal to any of the axes. As a result, a large number of points in the bottom right and the top left of the figure are left outside of the identified region. Logistic regression can overcome this limitation by identifying a diagonal boundary between the two regions, seen in panel (b). This method also produces a gradient of estimated probability of success across these regions.

Note: Put this into practice! Click the following link to try out an interactive tutorial on performing factor mapping using logistic regression: Logistic Regression Jupyter Notebook

---

Note: The following articles are suggested as fundamental reading for the information presented in this section:

- Gupta, H.V., Wagener, T., Liu, Y., 2008. Reconciling theory with observations: elements of a diagnostic approach to model evaluation. *Hydrological Processes: An International Journal* 22, 3802–3813.
- Bankes, S., 1993. Exploratory Modeling for Policy Analysis. *Operations Research* 41, 435–449. https://doi.org/10.1287/opre.41.3.435
- Groves, D.G., Lempert, R.J., 2007. A new analytic method for finding policy-relevant scenarios. *Global Environmental Change* 17, 73–85. https://doi.org/10.1016/j.gloenvcha.2006.11.006

The following articles can be used as supplemental reading: * Marchau, V.A.W.J., Walker, W.E., Bloemen, P.J.T.M., Popper, S.W. (Eds.), 2019. Decision Making under Deep Uncertainty: From Theory to Practice. Springer International Publishing. https://doi.org/10.1007/978-3-030-05252-2 * Sterman, J.D., 1994. Learning in and about complex systems. System Dynamics Review 10, 291–330. https://doi.org/10.1002/sdr.4260100214

CHAPTER

FIVE

CONCLUSION

As noted in the Introduction (Section 1), the computational and conceptual challenges of the multi-model, transdisciplinary workflows that characterize ambitious projects such as IM3 have limited UC and UQ analyses. Moreover, the very nature and purpose of modeling and diagnostic model evaluation can have very diverse philosophical framings depending on the disciplines involved (see Fig. 1.1 and Section 2.2). The guidance provided in this text can be used to frame consistent and rigorous experimental designs for better understanding the consequences and insights from our modeling choices when seeking to capture complex human-natural systems. The progression of sections of this text provide a thorough introduction of the concepts and definitions of diagnostic model evaluation, sensitivity analysis and UC. In addition, we comprehensively discuss how specific modeling objectives and applications should guide the selection of appropriate techniques; broadly, these can include model diagnostics, in-depth analysis of the behavior of the abstracted system, and projections under conditions of deep uncertainty. This text also contains a detailed presentation of the main sensitivity analysis methods and a discussion of their features and main limitations. Readers are also provided with an overview of computer tools and platforms that have been developed and could be considered in addressing IM3 scientific questions. The appendices of this text include an overview of UQ methods, a terminology glossary of the key concepts as well as example test cases and scripts to showcase various UC related capabilities.

Although we distinguish the UC and UQ model diagnostics, the reader should note that we suggest an overall consistent approach to both in this text by emphasizing "exploratory modeling" (see review by Moallemi *et al.* [8]). Although data support, model complexity, and computational limits strongly distinguish the feasibility and appropriateness of various UC diagnostic tools (e.g., see Fig. 3.5), we overall recommend that modelers view their work through the lens of cycles of learning. Iterative and deliberative exploration of model-based hypotheses and inferences for transdisciplinary teams is non-trivial and ultimately critical for mapping where innovations or insights are most consequential. Overall, we recommend approaching modeling with an openness to the diverse disciplinary perspectives such as those mirrored by the IM3 family of models in a progression from evaluating models relative to observed history to advanced formalized analyses to make inferences on multi-sector, multi-scale vulnerabilities and resilience. Exploratory modeling approaches can help fashion experiments with large numbers of alternative hypotheses on the co-evolutionary dynamics of influences, stressors, as well as path-dependent changes in the form and function of coupled human-natural systems [38]. This text guides the reader through the use of sensitivity analysis and uncertainty methods across the diverse perspectives that have shaped modern diagnostic and exploratory modeling.

CHAPTER

SIX

GLOSSARY

CHAPTER

SEVEN

UNCERTAINTY QUANTIFICATION

7.1 Introduction

As defined in Section 1, uncertainty quantification (UQ) refers to the formal focus on the full specification of likelihoods as well as distributional forms necessary to infer the joint probabilistic response across all modeled factors of interest [7]. This is in contrast to UC (the primary focus of the main document of this book), which is instead aimed at identifying which modeling choices yield the most consequential changes or outcomes and exploring alternative hypotheses related to the form and function of modeled systems [8, 9].

UQ is important for quantifying the relative merits of hypotheses for at least three main reasons. First, identifying model parameters that are consistent with observations is an important part of model development. Due to several effects, including correlations between parameters, simplified or incomplete model structures (relative to the full real-world dynamics), and uncertainty in the observations, many different combinations of parameter values can be consistent with the model structure and the observations to varying extents. Accounting for this uncertainty is conceptually preferable to selecting a single "best fit" parameter vector, particularly as consistency with historical or present observations does not necessarily guarantee skillful future projections.

The act of quantification requires specific assumptions about distributional forms and likelihoods, which may be more or less justified depending on prior information about the system or model behavior. As a result, UQ is well-suited for studies accounting for or addressing hypotheses related to systems with a relatively large amount of available data and models which are computationally inexpensive, particularly when the emphasis is on prediction. As shown in Fig. 7.1, there is a fundamental tradeoff between the available number of model evaluations (for a fixed computational budget) and the number of parameters treated as uncertain. Sensitivity analyses are therefore part of a typical UQ workflow to identify which factors can be fixed and which ought to be prioritized in the UQ.

The choice of a particular UQ method depends on both the desired level of quantification and the ability to navigate the tradeoff between computational expense and the number of uncertain parameters (Fig. 7.1). For example, Markov chain Monte Carlo with a full system model can provide an improved representation of uncertainty compared to the coarser pre-calibration approach [158], but requires many more model evaluations. The use of a surrogate model to approximate the full system model can reduce the number of needed model evaluations by several orders of magnitude, but the uncertainty quantification can only accommodate a limited number of parameters.

The remainder of this appendix will focus on introducing workflows for particular UQ methods, including a brief discussion of advantages and limitations.

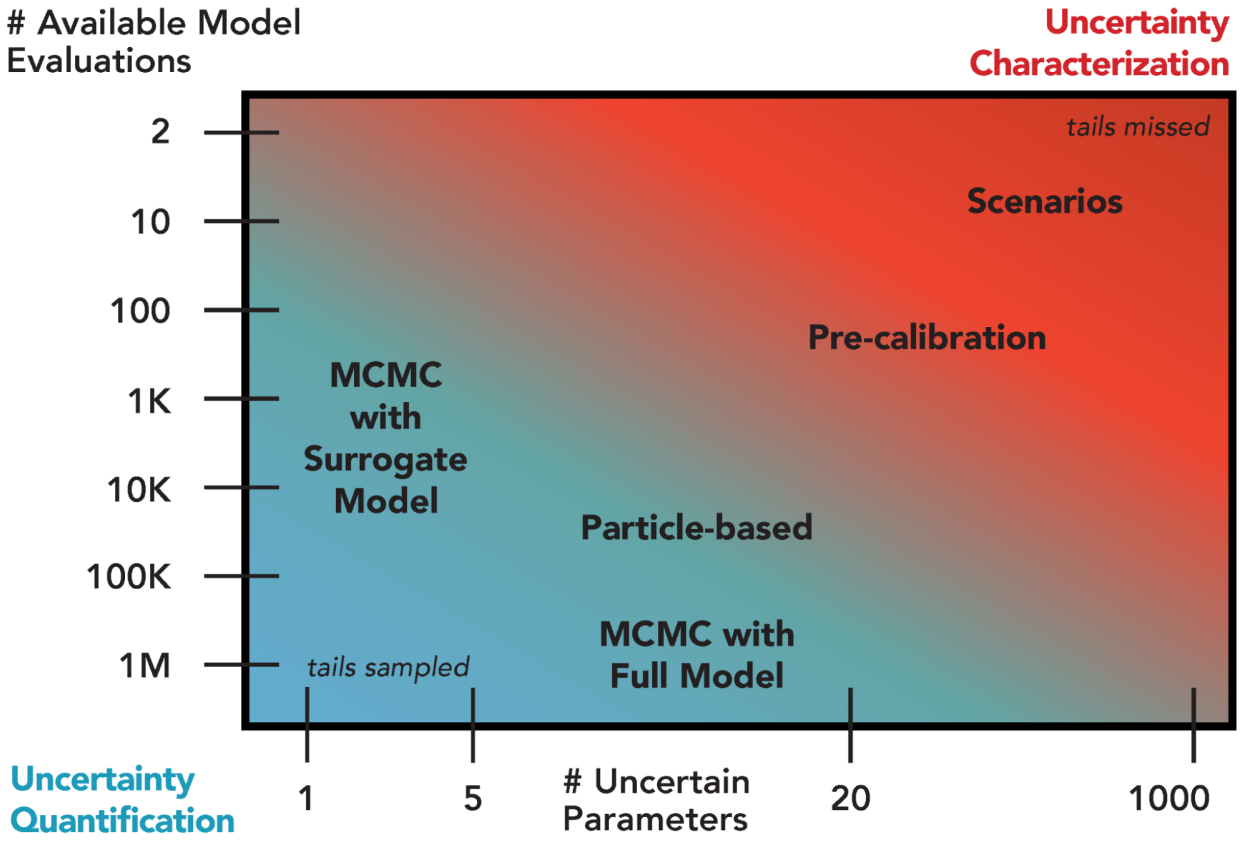


Fig. 7.1: Overview of selected existing approaches for uncertainty quantification and their appropriateness given the number of uncertain model parameters and the number of available model simulations. Green shading denotes regions suitable for uncertainty quantification and red shading indicates regions more appropriate for uncertainty characterization.

7.2 Parametric Bootstrap

The parametric bootstrap [159] refers to a process of model recalibration to alternate realizations of the data. The bootstrap was originally developed to estimate standard errors and confidence intervals without ascertaining key assumptions that might not hold given the available data. In a setting where observations can be viewed as independent realizations of an underlying stochastic process, a sufficiently rich dataset can be treated as a population representing the data distribution. New datasets are then generated by resampling from the data with replacement, and the model can be refit to each new dataset using maximum-likelihood estimation. The resulting distribution of estimates can then be viewed as a representation of parametric uncertainty.

A typical workflow for the parametric bootstrap is shown in Fig. 7.2. After identifying outputs of interest and preparing the data, the parametric model is fit by some procedure such as minimizing root-mean-square-error or maximizing the likelihood. Alternate datasets are constructed by resampling from the population or by generating new samples from the fitted data-generating process. It is important at this step that the resampled quantities are independent of one another. For example, in the context of temporally- or spatially-correlated data, such as time series, the raw observations cannot be treated as independent realizations. However, the residuals resulting from fitting the model to the data could be (depending on their structure). For example, if the residuals are treated as independent, they can then be resampled with replacement, and these residuals added to the original model fit to create new realizations. If the residuals are assumed to be the result of an autoregressive process, this process could be fit to the original residual series and new residuals be created using this model [160]. The model is then refit to each new realization.

The bootstrap is computationally convenient, particularly as the process of fitting the model to each realization can be easily parallelized. This approach also requires minimal prior assumptions. However, due to the assumption that the available data are representative of the underlying data distribution, the bootstrap can neglect key uncertainties which might influence the results. For example, when using an autoregressive process to generate new residuals, uncertainty in the autocorrelation parameter and innovation variance is neglected, which may bias estimates of, for example, low-probability but high-impact events [161].

7.3 Pre-Calibration

Pre-calibration [19, 162, 163] involves the identification of a plausible set of parameters using some prespecified screening criterion, such as the distance from the model results to the observations (based on an appropriate metric for the desired matching features, such as root-mean-squared error). A typical workflow is shown in Fig. 7.3. Parameter values are obtained by systematically sampling the input space (see Section 3.3). After the model is evaluated at the samples, only those passing the distance criterion are retained. This selects a subset of the parameter space as "plausible" based on the screening criterion, though there is no assignment of probabilities within this plausible region.

Pre-calibration can be useful for models which are inexpensive enough that a reasonable number of samples can be used to represent the parameter space, but which are too expensive to facilitate full uncertainty quantification. High-dimensional parameter spaces, which can be problematic for the uncertainty quantification methods below, may also be explored using pre-calibration. One key prerequisite to using this method is the ability to place a meaningful distance metric on the output space.

However, pre-calibration results in a very coarse characterization of uncertainty, especially when considering a large number of parameters, as more samples are needed to fully characterize the parameter space. Due to the inability to evaluate the relative probability of regions of the parameter space beyond the binary plausible-and-implausible characterization, pre-calibration can also result in degraded hindcast and projection skills and parameter estimates [158, 164, 165].

A related method, widely used in hydrological studies, is generalized likelihood uncertainty estimation, or GLUE [19]. Unlike pre-calibration, the underlying argument for GLUE relies on the concept of equifinality [166], which posits that it is impossible to find a uniquely well-performing parameter vector for models of abstract environmental systems [166, 167]. In other words, there exist multiple parameter vectors which perform equally or similarly well. As

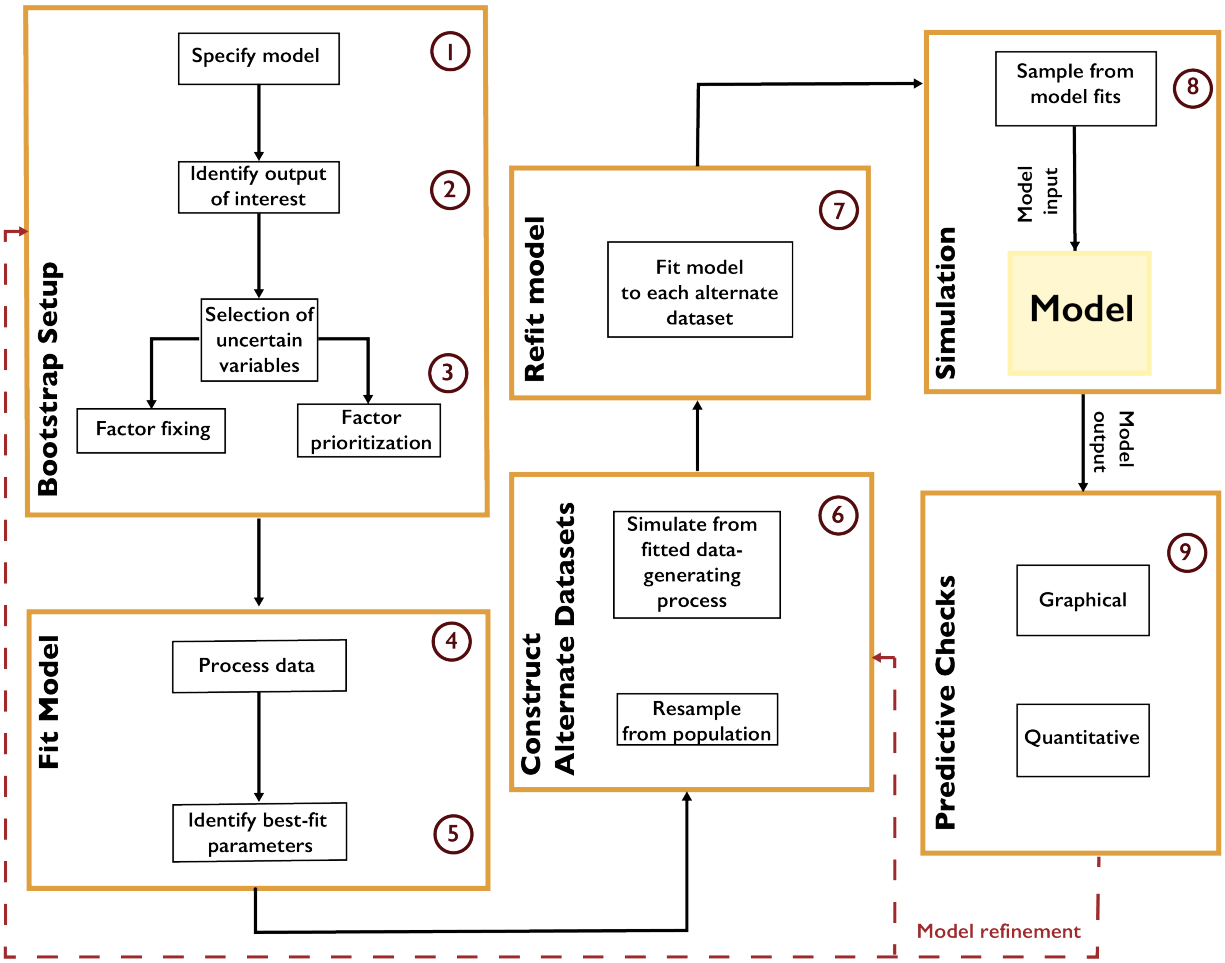


Fig. 7.2: Workflow for the parametric bootstrap.

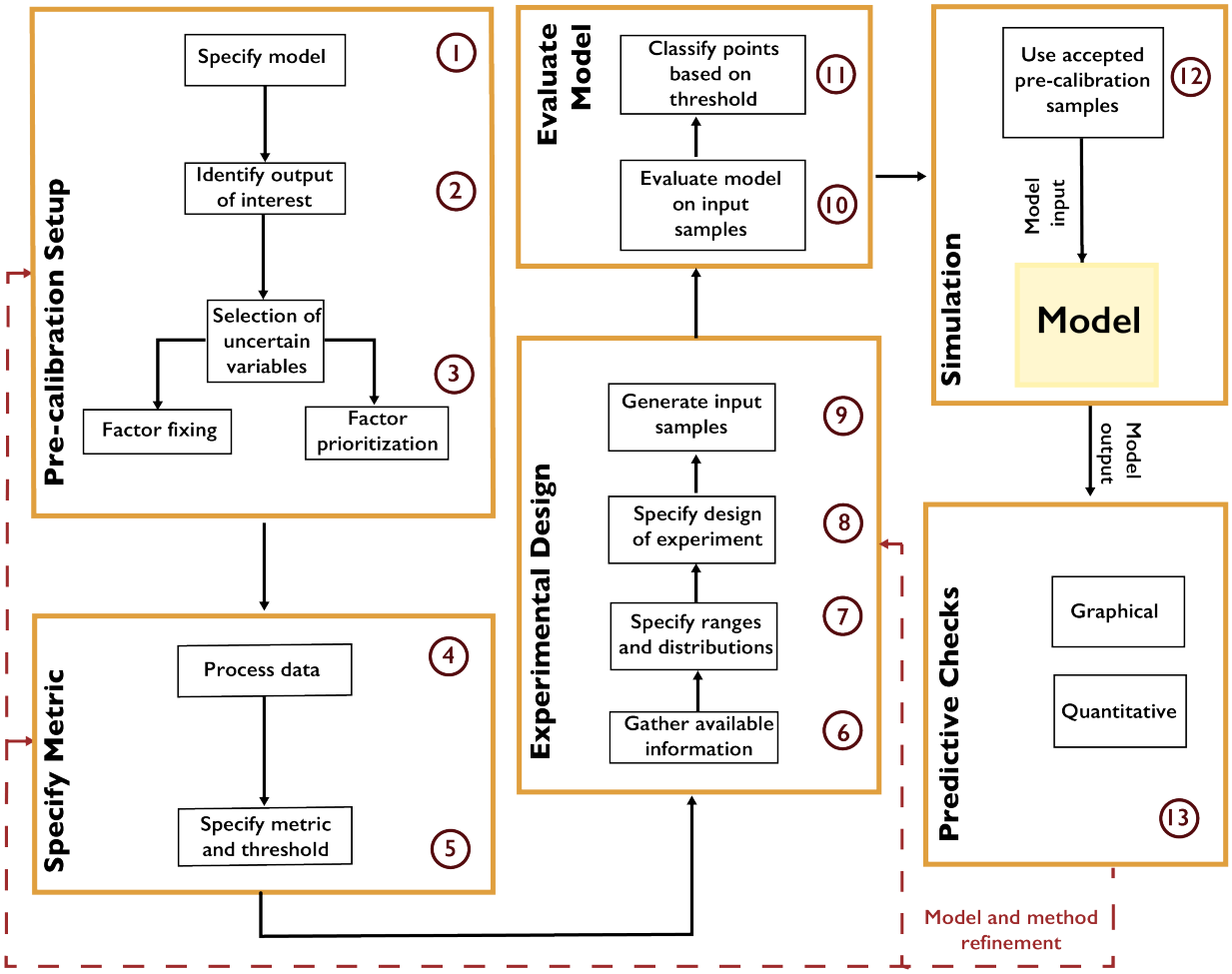


Fig. 7.3: Workflow for pre-calibration.

with pre-calibration, GLUE uses a goodness-of-fit measure (though this is called a "likelihood" in the GLUE literature, as opposed to a statistical likelihood function [168]) to evaluate samples. After setting a threshold of acceptable performance with respect to that measure, samples are evaluated and classified into "behavioral" or "non-behavioral" according to the threshold.

Note: Put this into practice! Click the following badge to try out an interactive tutorial on utilizing Pre-Calibration and GLUE for HYMOD model calibration: Pre-Calibration Jupyter Notebook

7.4 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is a "gold standard" approach to full uncertainty quantification. MCMC refers to a category of algorithms which systematically sample from a target distribution (in this case, the posterior distribution) by constructing a Markov chain. A Markov chain is a probabilistic structure consisting of a state space, an initial probability distribution over the states, and a transition distribution between states. If a Markov chain satisfies certain properties [169, 170], the probability of being in each state will eventually converge to a stable, or stationary, distribution, regardless of the initial probabilities.

MCMC algorithms construct a Markov chain of samples from a parameter space (the combination of model and statistical parameters). This Markov chain is constructed so that the stationary distribution is a target distribution, in this case the (Bayesian) posterior distribution. As a result, after the transient period, the resulting samples can be viewed as a set of dependent samples from the posterior (the dependence is due to the autocorrelation between samples resulting from the Markov chain transitions). Expected values can be computed from these samples (for example, using batch-means estimators [171]), or the chain can be sub-sampled or thinned and the resulting samples used as independent Monte Carlo samples due to the reduced or eliminated autocorrelation.

A general workflow for MCMC is shown in Fig. 7.4. The first decision is whether to use the full model or a surrogate model (or emulator). Typical surrogates include Gaussian process emulation [172, 173], polynomial chaos expansions [174, 175], support vector machines :cite:p`ciccazzo_svm_2016, pruett_creation_2016`, and neural networks [176, 177]. Surrogate modeling can be faster, but requires a sufficient number of model evaluations for the surrogate to accurately represent the model's response surface, and this typically limits the number of parameters which can be included in the analysis.

After selecting the variables which will be treated as uncertain, the next step is to specify the likelihood based on the selected surrogate model or the structure of the data-model residuals. For example, it may not always be appropriate to treat the residuals as independent and identically distributed (as is commonly done in linear regression). A misspecification of the residual structure can result in biases and over- or under-confident inferences and projections [178].

After specifying the prior distributions (see Section 7.6), the selected MCMC algorithm should be used to draw samples from the posterior distribution. There are many MCMC algorithms, all of which have advantages and disadvantages for a particular problem. These include the Metropolis-Hastings algorithm [170] and Hamiltonian Monte Carlo [179, 180]. Software packages typically implement one MCMC method, sometimes designed for a particular problem setting or likelihood specification. For example, R's *adaptMCMC* implements an adaptive Metropolis-Hastings algorithm [181], while *NIMBLE* [182, 183] uses a user-customizable Metropolis-Hastings implementation, as well as functionality for Gibbs sampling (which is a special case of Metropolis-Hastings where the prior distribution has a convenient mathematical form). Some recent implementations, such as *Stan* [184], *pyMC3* [185], and *Turing* [186] allow different algorithms to be used.

A main consideration when using MCMC algorithms is testing for convergence to the target distribution. As convergence is guaranteed only for a sufficiently large number of transitions, it is impossible to conclude for certain that a chain has converged for a fixed number of iterations. However, several heuristics have been developed [171, 187] to increase evidence that convergence has occurred.

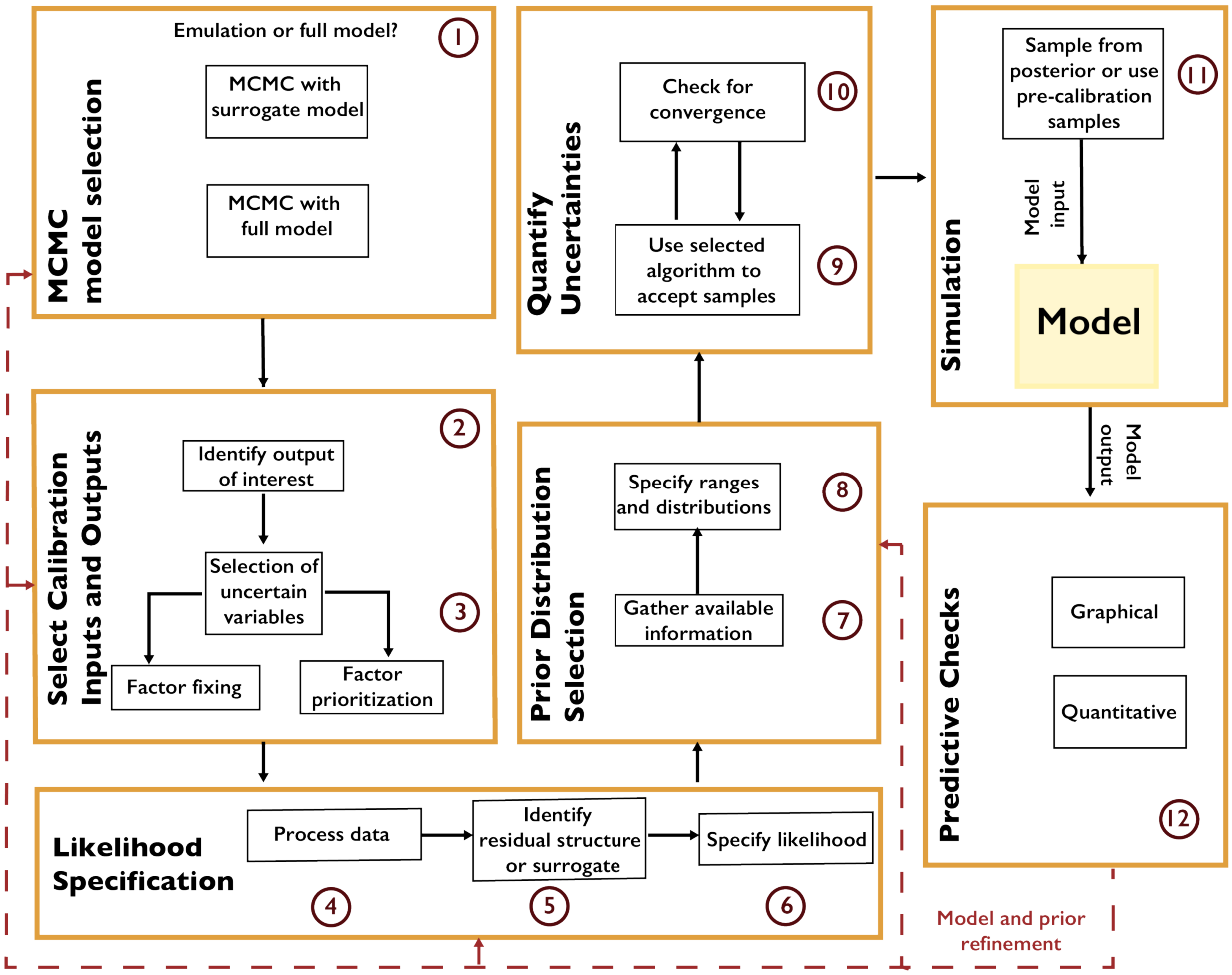


Fig. 7.4: Workflow for Markov chain Monte Carlo.

7.5 Other Methods

Other common methods for UQ exist. These include sequential Monte Carlo, otherwise known as particle filtering [188, 189, 190], where a number of particles are used to evaluate samples. An advantage of sequential Monte Carlo is that the vast majority of the computation can be parallelized, unlike with standard MCMC. A major weakness is the potential for degeneracy [189], where many particles have extremely small weights, resulting in the effective use of only a few samples.

Another method is approximate Bayesian computation (ABC) [191, 192, 193]. ABC is a likelihood-free approach that compares model output to a set of summary statistics. ABC is therefore well-suited for models and residual structures which do not lend themselves to a computationally-tractable likelihood, but the resulting inferences are known to be biased if the set of summary statistics is not sufficient, which can be difficult to know a-priori.

7.6 The Critical First Step: How to Choose a Prior Distribution

Prior distributions play an important role in Bayesian uncertainty quantification, particularly when data is limited relative to the dimension of the model. Bayesian updating can be thought of as an information filter, where each additional datum is added to the information contained in the prior; eventually, the prior makes relatively little impact. In real world problems, it can be extremely difficult to assess how much data is required for the choice of prior to become less relevant. The choice of prior can also be influential when conducting SA prior to or without UQ. This is because a prior distribution for a sensitivity analysis for a given parameter which is much wider than the region where the model response surface is sensitive to the parameter value might cause the sensitivity calculation to underestimate the response in that potentially critical region. Similarly, a prior which is too narrow may miss regions where the model responds to the parameter altogether.

Ideally, prior distributions are constructed independently of any analysis of the new data considered. This is because using data to inform the prior as well as to compute the likelihood reuses information in a potentially inappropriate way, which can lead to overconfident inferences. Following Jaynes [194], Gelman *et al.* [195] refers to the ideal prior as one which encodes all available information about the model. For practical reasons (difficulty of construction or computational inconvenience), most priors fail to achieve this ideal. These compromises mean that priors should be transparently articulated and justified, so that the impact of the choice of prior can be fully understood. When there is ambiguity about an appropriate prior, such as how fat the tails should be, an analyst should examine how sensitive the UQ results are to the choice of prior.

Priors can also be classified in terms of the information encoded by them, demonstrated in Fig. 7.5. Non-informative priors (illustrated in Fig. 7.5 (a)) allegedly correspond to (and are frequently justified by) a position of ignorance. A classic example is the use of a uniform distribution. A uniform prior can, however, be problematic, as it can lead to improper inferences by giving extremely large values the same prior probability as values which may seem more likely [195, 196], and therefore does not really reflect a state of complete ignorance. In the extreme case of a uniform prior over the entire real line, every particular region has effectively a prior weight of zero, even though not all regions are a priori unlikely [196]. Moreover, a uniform prior which excludes possible parameter values is not actually noninformative, as it assigns zero probability to those values while jumping to a nonzero probability as soon as the boundary is crossed. While a uniform prior can be problematic for the task of uncertainty quantification, it may be useful for an initial sensitivity analysis to identify the boundary of any regions where the model is sensitive to the parameter.

Informative priors strongly bound the range of probable values (illustrated in Fig. 7.5 (c)). One example is a Gaussian distribution with a relatively small standard deviation, so that large values are assigned a close to null prior probability. Another example is the jump from zero to non-zero probability occurring at the truncation point of a truncated Gaussian, which could be justified based on information that the parameter cannot take on values beyond this point. Without this type of justification, however, priors may be too informative, failing to allow the information contained in the available data to update them.

Finally, weakly informative priors (illustrated in Fig. 7.5 (b)) fall in between [195]. They regularize better than non-informative priors, but allow for more inference flexibility than fully informative priors. An example might be a Gaus-

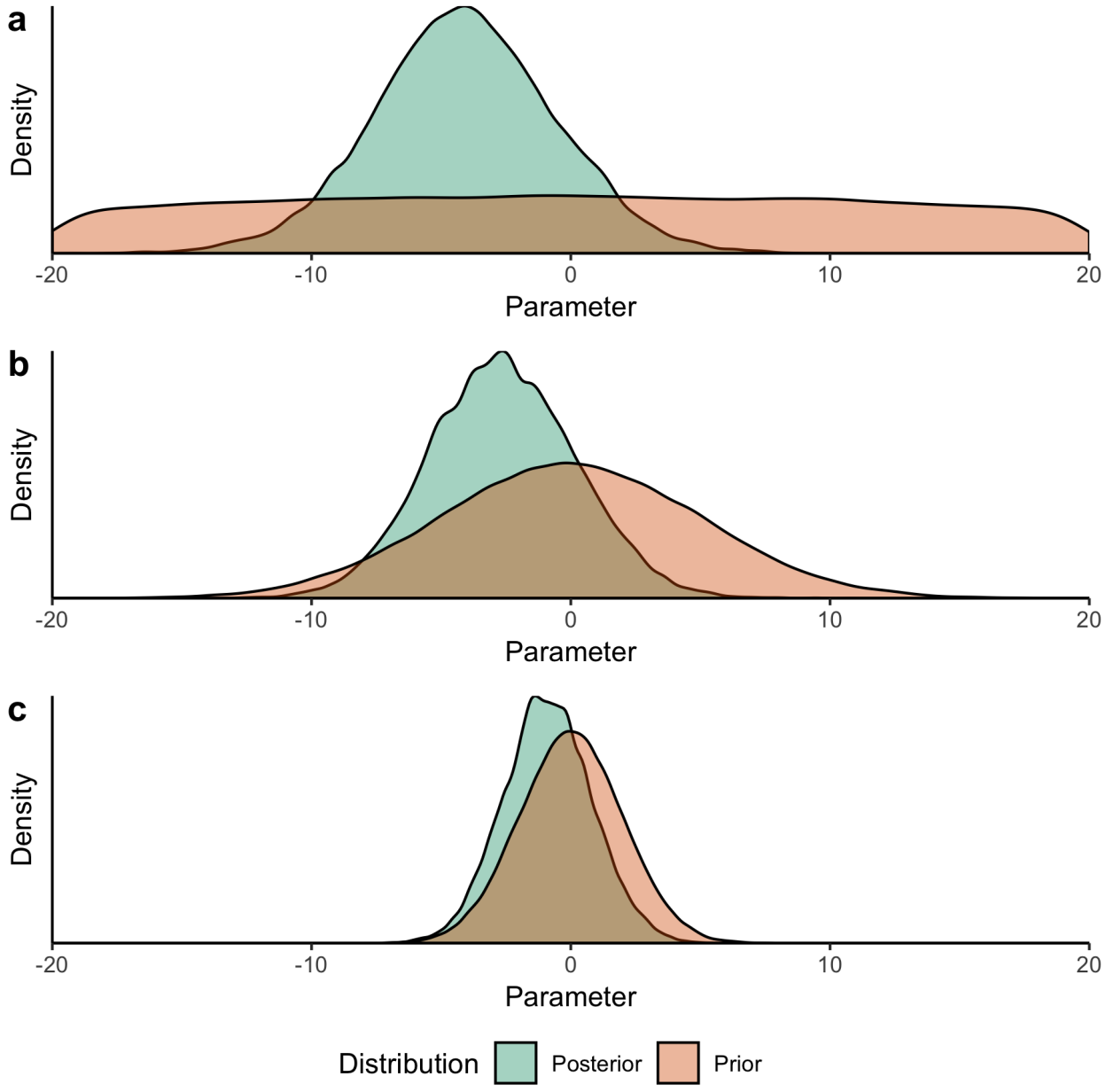


Fig. 7.5: Impact of priors on posterior inferences. These plots show the results of inference for a linear regression model with 15 data points. The true value of the parameter is equal to -3. All priors have mean 0. In panel (a), a non-informative prior allows the tails of the posterior to extend freely, which may result in unreasonably large parameter values. In panel (b), a weakly informative prior constrains the tails more, but allows them to extend without too much restriction. In panel (c), an informative prior strongly constrains the tails of the posterior and biases the inference closer towards the prior mean (the posterior mean is -0.89 in this case, and closer to -3 in the other two cases).

sian distribution with a moderate standard deviation, which still assigns negligible probability for values far away from the mean, but is less constrained than a narrow Gaussian for a reasonably large area. A key note is that it is not necessarily better to be more informative if this cannot be justified by the available information.

7.7 The Critical Final Step: Predictive Checks

Every UQ workflow requires a number of choices, potentially including selecting prior distributions, the likelihood specification, and any used numerical models. Checking the appropriateness of these choices is an essential step for sound inferences, as misspecification can produce biased results [178]. Model checking in this fashion is part of an iterative UQ process, as the results can reveal adjustments to the statistical model or the need to select a different numerical model [197, 198, 199].

A classic example is the need to check the structure of residuals for correlations. Many standard statistical models, such as linear regression, assume that the residuals are independent and identically distributed from the error distribution. The presence of correlations, including temporal autocorrelations and spatial correlations, indicates a structural mismatch between the likelihood and the data. In these cases, the likelihood adjusted to account for these correlations.

Checking residuals in this fashion is one example of a predictive check (or a posterior predictive check in the Bayesian setting). One way to view UQ is as a means to recover data-generating processes (associated with each parameter vector) consistent with the observations. Predictive checks compare the inferred data-generating process to the observations to determine whether the model is capable of appropriately capturing uncertainty. After conducting the UQ analysis, alternatively realized datasets are simulated from sampled parameters. These alternative datasets, or their summary statistics, can be tested against the observations to determine adequacy of the fit. Predictive checks are therefore a way of probing various model components to identify shortcomings that might result in biased inferences or poor projections, depending on the goal of the analysis.

One example of a graphical predictive check for time series models is hindcasting, where predictive intervals are constructed from the alternative datasets and plotted along with the data. Hindcasts demonstrate how well the model is capable of capturing the broader dynamics of the data, as well as whether the parameter distributions produce appropriate levels of output uncertainty. A related quantitative check is the surprise index, which calculates the percentage of data points located within a fixed predictive interval. For example, the 90% predictive interval should contain approximately 90% of the data. More uncertainty than this reflects underconfidence, while less uncertainty reflects overconfidence. This could be the result of priors that are not appropriately informative, or a likelihood that does not account for correlations between data points appropriately. It could also be the result of a numerical model that isn't sufficiently sensitive to the parameters that are treated as uncertain.

7.8 Key Take-Home Points

When appropriate, UQ is an important component of the exploratory modeling workflow. While a number of parameter sets could be consistent with observations, they may result in divergent model outputs when exposed to different future conditions. This can result in identifying risks which are not visible when selecting a "best fit" parameterization. Quantifying uncertainties also allows us to quantify the support for hypotheses, which is an essential part of the scientific process.

Due to the scale and complexity of the experiments taking place in IM3, UQ has not been extensively used. The tradeoff between the available number of function evaluations and the number of uncertain parameters illustrated in Fig. 7.1 is particularly challenging due to the increasing complexity of state-of-the-art models and the movement towards coupled, multisector models. This tradeoff can be addressed somewhat through the use of emulators and parallelizable methods. In particular, when attempting to navigate this tradeoff by limiting the number of uncertain parameters, it is important to carefully iterate with sensitivity analyses to ensure that critical parameters are identified.

Specifying prior distributions and likelihoods is an ongoing challenge. Prior distributions, in particular, should be treated as deeply uncertain when appropriate. One key advantage of the methods described in this chapter is that they

have the potential for increased transparency. When it is not possible to conduct a sensitivity analysis on a number of critical priors due to limited computational budgets, fully specifying and providing a justification for the utilized distributions allows other researchers to identify key assumptions and build on existing work. The same is true for the specification of likelihoods—while likelihood-free methods avoid the need to specify a likelihood function, they require other assumptions or choices, which should be described and justified as transparently as possible.

We conclude this appendix with some key recommendations: 1. UQ analysis does not require full confidence in priors and likelihoods. Rather, UQ should be treated as part of an exploratory modeling workflow, where hypotheses related to model structures, prior distributions, and likelihoods can be tested. 2. For complex multisectoral models, UQ will typically require the use of a reduced set of parameters, either through emulation or by fixing the others to their best-fit values. These parameters should be selected through a careful sensitivity analysis. 3. Avoid the use of supposedly "non-informative" priors, such as uniform priors, whenever possible. In the absence of strong information about parameter values, the use of weakly informative priors, such as diffuse normals, is preferable. 4. Be cognizant of the limitations of conclusions that can be drawn by using each method. The bootstrap, for example, may result in overconfidence if the dataset is limited and is not truly representative of the underlying stochastic process. 5. When using MCMC, Markov chains can not be shown to have converged to the target distribution, but rather evidence can be collected to demonstrate that it is likely that they have. 6. Conduct predictive checks based on the assumptions underlying the choices made in the analysis, and iteratively update those choices if the assumptions prove to be ill-suited for the problem at hand.

CHAPTER

EIGHT

JUPYTER NOTEBOOK TUTORIALS

8.1 Fishery Dynamics Tutorial

Note: Run the tutorial interactively: Fishery Dynamics Notebook

8.1.1 Tutorial: Sensitivity Analysis (SA) to discover factors shaping consequential dynamics

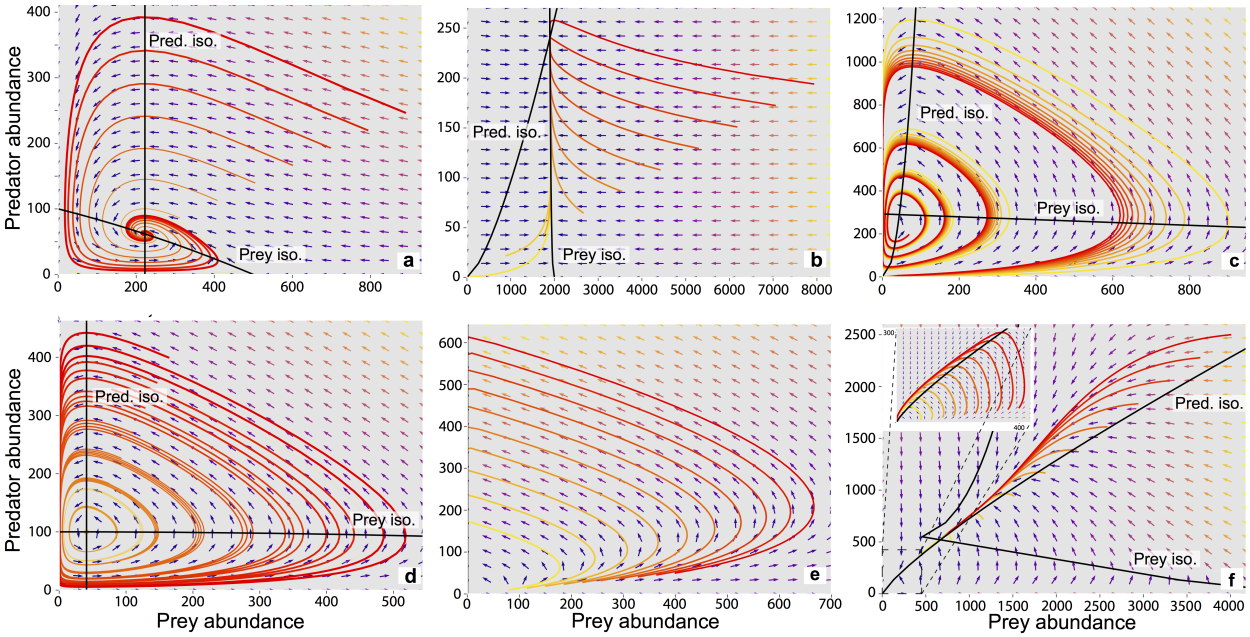
This notebook demonstrates the application of sensitivity analysis to discover factors that shape the behavior modes of a socio-ecological system with dynamic human action.

The model represents a system of prey and predator fish, with a human actor harvesting the prey fish. The system is simple but very rich in the dynamic behaviors it exhibits. You can read more about this system at Hadjimichael et al. (2020).

$$\frac{\mathrm{d}x}{\mathrm{d}t} = bx\left(1 - \frac{x}{K}\right) - \frac{\alpha x y}{y^m + \alpha h x} - zx$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{c\alpha x y}{y^m + \alpha h x} - dy$$

This complexity is accompanied by the presence of several equilibria that come in and out of existence with different parameter values. The equilibria also change in their stability according to different parameter values, giving rise to different behavior modes.



In the unharvested system (without the human actor) the stability of several of these equilibria can be derived analytically. The task becomes significantly more difficult when the adaptive human actor is introduced, deciding to harvest the system at different rates according to their objectives and preferences.

Sensitivity analysis methods can help us identify the factors that most control these dynamics by exploring the space of parameter values and seeing how system outputs change as a result.

Through previously conducted optimization, there already exists a set of potential harvesting strategies that were identified in pursuit of five objectives:

- Maximize Harvesting Discounted Profits (Net Present Value)
- Minimize Prey Population Deficit
- Minimize Longest Duration of Consecutive Low Harvest
- Maximize Worst Harvest Instance
- Minimize Harvest Variance

The identified harvesting strategies also meet the necessary constraint of not causing inadvertent predator collapse.

We will be examining the effects of parametric uncertainty on these identified strategies, particularly focusing on two strategies: one selected to maximize harvesting profits and one identified through previous analysis to perform 'well enough' for all objectives across a wide range of states of the world (referred to as the 'robust' harvesting policy).

```
import msdbook

import numpy as np
import matplotlib.pyplot as plt

from SALib.sample import saltelli
from SALib.analyze import sobol
from matplotlib import patheffects as pe

# load example data
msdbook.install_package_data()
```

(continues on next page)

(continued from previous page)

```
%matplotlib inline
%config InlineBackend.print_figure_kwargs = {'bbox_inches':None}
```

```
Downloading example data for msdbook version 0.1.5...
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/uncertain_
↪params_bounds.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_metric_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/solutions.
↪resultfile
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LHsamples_
↪original_1000.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/param_values.
↪csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/collapse_
↪days.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_params_
↪256samples.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LeafCatch.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_
↪simulations_256samples.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/Robustness.
↪txt
```

8.1.1.1 Step 1: Load identified solutions and explore performance

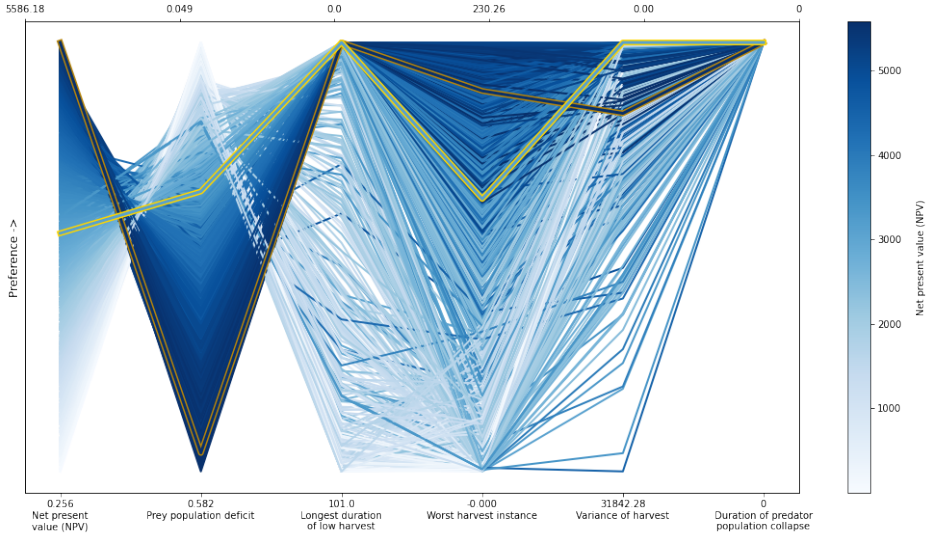
Identify and load the most robust and profit-maximizing solutions

```
robustness = msdbook.load_robustness_data()
results = msdbook.load_profit_maximization_data()

robust_solution = np.argmax(robustness[:,-1]) #pick robust solution
profit_solution = np.argmin(results[:,6]) #pick profitable solution
objective_performance = -results[:,6:]

# Get decision variables for each of the policies
highprofitpolicy = results[profit_solution,0:6]
mostrobustpolicy = results[robust_solution,0:6]
```

Plot the identified solutions with regards to their objective performance in a parallel axis plot

Note: **Tip:** View the source code used to create this plot here: plot_objective_performance

```
ax, ax1 = msdbook.plot_objective_performance(objective_performance, profit_solution,␣
↪robust_solution)
```



The results of the optimization are presented in a parallel axis plot where each of the five objectives (and one constraint) are represented as an axis. Each solution on the Pareto front is represented as a line where the color of the line indicates the value of the NPV objective. The preference for objective values is in the upward direction. Therefore, the ideal solution would be a line straight across the top of the plot that satisfies every objective. However, no such line exists because there are tradeoffs when sets of objectives are prioritized over the others. When lines cross in between axes, this indicates a tradeoff between objectives (as seen in the first two axes).The solution that is most robust in the NPV objective has the highest value on the first axis and is outlined in dark gold. The solution that is most robust across all objectives is outlined in a brighter yellow.

8.1.1.2 Step 2: Use SALib to generate a sample for a Sobol sensitivity analysis

To do so, we first need to define the problem dictionary that allows us to generate alternative states of the world.

```
# Set up SALib problem
problem = {
  'num_vars': 9,
  'names': ['a', 'b', 'c', 'd', 'h', 'K', 'm', 'sigmaX', 'sigmaY'],
  'bounds': [[0.002, 2], [0.005, 1], [0.2, 1], [0.05, 0.2], [0.001, 1],
             [100, 5000], [0.1, 1.5], [0.001, 0.01], [0.001, 0.01]]
}
```

You can use the following to generate a Saltelli sample using the following:

```
param_values = saltelli.sample(problem, 1024, calc_second_order=False)
```

Generally, it is a good idea to save the result of the sample since it is often reused and regenerating it produces a different sample set. For this reason, we will load one from file that was previously generated.

```
# load previously generated Saltelli sample from our msdbook package data
param_values = msdbook.load_saltelli_param_values()
```

8.1.1.3 Step 3: Evaluate the system over all generated states of the world

We need to identify the states where the predator population collapses, as an inadvertent consequence of applying the harvesting strategy under a state of the world different from the one originally assumed. Due to how long this step takes to execute within the tutorial, we will read in the solutions from an external file. However, the block of code below shows how evaluation can be implemented.

```
# create array to store collapse values under both policies
collapse_days = np.zeros([len(param_values), 2])

# evaluate performance under every state
for i in range(len(param_values)):

    additional_inputs = np.append(['Previous_Prey'],
                                  [param_values[i,0],
                                   param_values[i,1],
                                   param_values[i,2],
                                   param_values[i,3],
                                   param_values[i,4],
                                   param_values[i,5],
                                   param_values[i,6],
                                   param_values[i,7],
                                   param_values[i,8]])

    collapse_days[i,0]=fish_game(highprofitpolicy, additional_inputs)[1][0]
    collapse_days[i,1]=fish_game(mostrobustpolicy, additional_inputs)[1][0]
```

```
# load the simulation data from our msdbook package data
collapse_days = msdbook.load_collapse_data()
```

8.1.1.4 Step 4: Calculate sensitivity indices

```
Si_profit = sobol.analyze(problem, collapse_days[:, 0],
                          calc_second_order=False,
                          conf_level=0.95,
                          print_to_console=True)
```

```
              ST   ST_conf
a       0.278724  0.051918
b       0.188124  0.027986
c       0.015588  0.012159
d       0.077655  0.016051
h       0.025096  0.014796
K       0.033239  0.014006
m       0.845465  0.071372
sigmaX  0.000708  0.000851
sigmaY  0.000849  0.000470
              S1   S1_conf
a       0.126405  0.042938
b       0.060739  0.034380
c       0.003333  0.008758
d       0.011388  0.025792
h       0.010233  0.013034
K       0.016699  0.015731
m       0.609991  0.072196
sigmaX  0.000531  0.001607
sigmaY  0.000337  0.002014
```

```
Si_robustness = sobol.analyze(problem,
                              collapse_days[:, 1],
                              calc_second_order=False,
                              conf_level=0.95,
                              print_to_console=True)
```

```
              ST   ST_conf
a       0.226402  0.038177
b       0.066819  0.017905
c       0.004395  0.004478
d       0.024509  0.006695
h       0.009765  0.006605
K       0.020625  0.010860
m       0.897971  0.070086
sigmaX  0.000136  0.000152
sigmaY  0.000739  0.001088
              S1   S1_conf
a       0.087936  0.045617
b       0.000554  0.019070
c      -0.002970  0.004227
d       0.001206  0.015897
h       0.004554  0.008202
K       0.003843  0.012294
m       0.751301  0.063511
```

(continues on next page)

(continued from previous page)

```
sigmaX -0.000325  0.001155
sigmaY -0.001887  0.003287
```

Looking at the total-order indices, (ST) factors m, a, b, d and K appear to affect the stability of this system. Looking at the first-order indices (S1), we also see that besides factors m and a, all other factors are important in this system through their interactions, which make up the difference between their S1 and ST indices. This shows the danger of limiting sensitivity analyses to first order effects, as factor importance might be significantly misjudged.

These findings are supported by the analytical condition of equilibrium stability in this system:

$$\alpha(hK)^{1-m} < (b)^m$$

In an unharvested system, this condition is both necessary and sufficient for the equilibrium of the two species coexisting to be stable.

When adaptive human action is introduced however, this condition is still necessary, but no longer sufficient, as harvesting reduces the numbers of prey fish and as a result reduces the resources for the predator fish. Since this harvesting value is not constant, but can dynamically adapt according to the harvester's objectives, it cannot be introduced into this simple equation.

8.1.1.5 Step 5: Explore relationship between uncertain factors and performance

In the following steps, we will use the results of our sensitivity analysis to investigate the relationships between parametric uncertainty, equilibrium stability and the performance of the two policies.

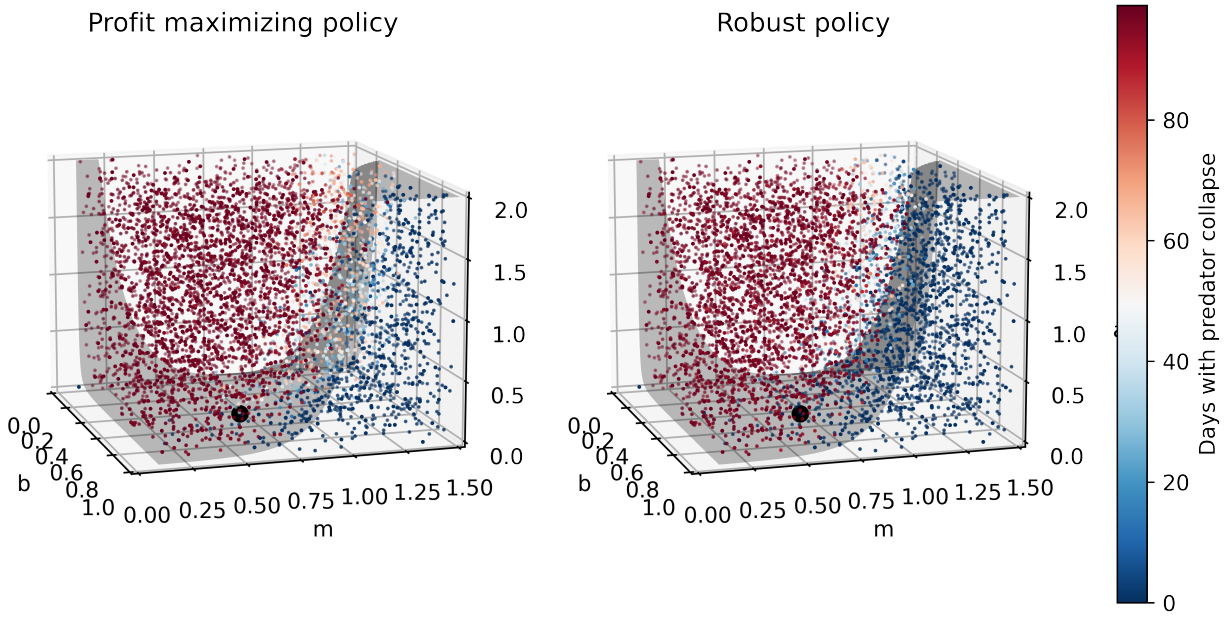
We can use the top three factors identified (m, a, and b) to visualize the performance of our policies in this three-dimensional parametric space.

We first define the stability condition, as a function of b and m, and calculate the corresponding values of a.

```
def inequality(b, m, h, K):
    return ((b**m)/(h*K)**(1-m))

# boundary interval that separates successful and failed states of the world
b = np.linspace(start=0.005, stop=1, num=1000)
m = np.linspace(start=0.1, stop=1.5, num=1000)
h = np.linspace(start=0.001, stop=1, num=1000)
K = np.linspace(start=100, stop=2000, num=1000)
b, m = np.meshgrid(b, m)
a = inequality(b, m, h, K)
a = a.clip(0,2)
```

Note: **Tip:** View the source code used to create this plot here: plot_factor_performance

```
# generate plot
ax1, ax2 = msdbook.plot_factor_performance(param_values, collapse_days, b, m, a)
```



These figures show the combinations of factors that lead to success or failure in different states of the world when the NPV-maximizing and Robust policies are utilized. Each point is a state of the world, characterized by specific values of the parameters, and ideally, we would like the color of the point to be blue, to represent that there are a low number of days with a predator collapse in that world. The gray curve denotes the highly non-linear nature of the boundary that separates successful and failed states of the world. The figures demonstrate the following key points:

First, as asserted above, the policies interact with the system in different and complex ways. In the presence of human action, the stability condition is not sufficient in determining whether the policy will succeed, even though it clearly shapes the system in a fundamental manner.

Secondly, the robust policy manages to avoid collapse in many more of the sampled states of the world, indicated by the number of blue points. This presents a clear tradeoff between profit-maximizing performance and robustness against uncertainty.

8.2 Sobol SA Tutorial

Note: Run the tutorial interactively: Sobol SA Tutorial

8.2.1 Tutorial: Sensitivity Analysis (SA) using the Saltelli sampling scheme with Sobol SA

In this tutorial, we will use the popular Python Sensitivity Analysis Library (SALib) to: 1. Generate a problem set as a dictionary for our Ishigami function that has three inputs 2. Generate 8000 samples for our problem set using the Saltelli1,2 sampling scheme 3. Execute the Ishigami function for each of our samples and gather the outputs 4. Compute the sensitivity analysis to generate first-order and total-order sensitivity indices using the Sobol3 method 5. Interpret the meaning of our results

8.2.1.1 Let's get started!

NOTE: Content from this tutorial is taken directly from the SALib "Basics" walkthrough.

```
import numpy as np

from SALib.sample import saltelli
from SALib.analyze import sobol
from SALib.test_functions import Ishigami
```

8.2.1.2 Step 1: Generate the problem dictionary

The Ishigami function is of the form:

$$f(x) = sin(x_1) + asin^2(x_2) + bx_3^4 sin(x_1)$$

and has three inputs, 1, 2, 3 where [,].

```
problem = {
    'num_vars': 3,
    'names': ['x1', 'x2', 'x3'],
    'bounds': [[-3.14159265359, 3.14159265359],
               [-3.14159265359, 3.14159265359],
               [-3.14159265359, 3.14159265359]]
}
```

8.2.1.3 Step 2: Generate samples using the Saltelli sampling scheme

Sobol SA requires the use of the Saltelli sampling scheme. The output of the `saltelli.sample` function is a NumPy array that is of shape 2048 by 3. The sampler generates (2+2) samples, where in this example N is 256 (the argument we supplied) and D is 3 (the number of model inputs), yielding 2048 samples. The keyword argument `calc_second_order=False` will exclude second-order indices, resulting in a smaller sample matrix with (+2) rows instead.

```
param_values = saltelli.sample(problem, 256)

print(f"`param_values` shape:  {param_values.shape}")
```

param_values shape: (2048, 3)

8.2.1.4 Step 3: Execute the Ishigami function over our sample set

SALib provides a nice wrapper to the Ishigami function that allows the user to directly pass the `param_values` array we just generated into the function directly.

```
Y = Ishigami.evaluate(param_values)
```

8.2.1.5 Step 4: Compute first-, second-, and total-order sensitivity indices using the Sobol method

The `sobol.analyze` function will use our problem dictionary and the result of the Ishigami runs (`Y`) to compute first-, second-, and total-order indicies.

```
Si = sobol.analyze(problem, Y)
```

`Si` is a Python dict with the keys "S1", "S2", "ST", "S1_conf", "S2_conf", and "ST_conf". The `_conf` keys store the corresponding confidence intervals, typically with a confidence level of 95%. Use the keyword argument `print_to_console=True` to print all indices. Or, we can print the individual values from `Si` as shown in the next step.

8.2.1.6 Step 5: Interpret our results

When we execute the following code to take a look at our first-order indices (`S1`) for each of our three parameters, we see that 1 and 2 exibit first-order sensitivities. This means that there is contribution to the output variance by those parameters independently, whereas 3 does not contribute to the output variance.

```
first_order = Si['S1']

print('First-order:')
print(f"x1: {first_order[0]}, x2: {first_order[1]}, x3: {first_order[2]}")
```

```
First-order:
x1: 0.3260389719592443, x2: 0.4820072841939227, x3: 0.011125510338583004
```

Next, we evaluate the total-order indices and find that they are substantially larger than the first-order indices, which reveals that higher-order interactions are occurring. Our total-order indices measure the contribution to the output variance caused by a model input, including both its first-order effects (the input varying alone) and all higher-order interactions. Now we see that 3 has non-negligible total order indices.

```
total_order = Si['ST']

print('Total-order:')
print(f"x1: {total_order[0]}, x2: {total_order[1]}, x3: {total_order[2]}")
```

```
Total-order:
x1: 0.5646024820275896, x2: 0.4570071429804512, x3: 0.2506488435438359
```

Finally, we can investigate these higher order interactions by viewing the second-order outputs. The second-order indicies measure the contribution to the output variance caused by the interaction between any two model inputs.

```
second_order = Si['S2']

print("Second-order:")
print(f"x1-x2:  {second_order[0,1]}")
print(f"x1-x3:  {second_order[0,2]}")
print(f"x2-x3:  {second_order[1,2]}")
```

```
Second-order:
x1-x2:  -0.018110907981879032
x1-x3:  0.2648898732606599
x2-x3:  -0.005645845624612848
```

We can see that there are strong interactions between 1 and 3. Note in the Ishigami function, these two variables are multiplied in the last term, which creates these interactive effects. If we were considering first order indices alone, we would erroneously assume that 3 has no effect on our output, but the second-order and total order indices reveal that this is not the case. It's easy to understand where we might see iteractive effects in the case of the simple Ishigami function. However, it's important to remember that in more complex systems, there may be many higher-order interactions that are not apparent, but could be extremely consequential in contributing to the variance of the output. Additionally, some computing error will appear in the sensitivity indices. For example, we observe a negative value for the 2-3 index. Typically, these computing errors shrink as the number of samples increases.

8.2.1.7 References

[1] Saltelli, A. (2002). "Making best use of model evaluations to compute sensitivity indices." Computer Physics Communications, 145(2):280-297, doi:10.1016/S0010-4655(02)00280-1.

[2] Saltelli, A., P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola (2010). "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index." Computer Physics Communications, 181(2):259-270, doi:10.1016/j.cpc.2009.09.018.

[3] Sobol, I. M. (2001). "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates." Mathematics and Computers in Simulation, 55(1-3):271-280, doi:10.1016/S0378-4754(00)00270-6.

8.3 Logistic Regression Tutorial

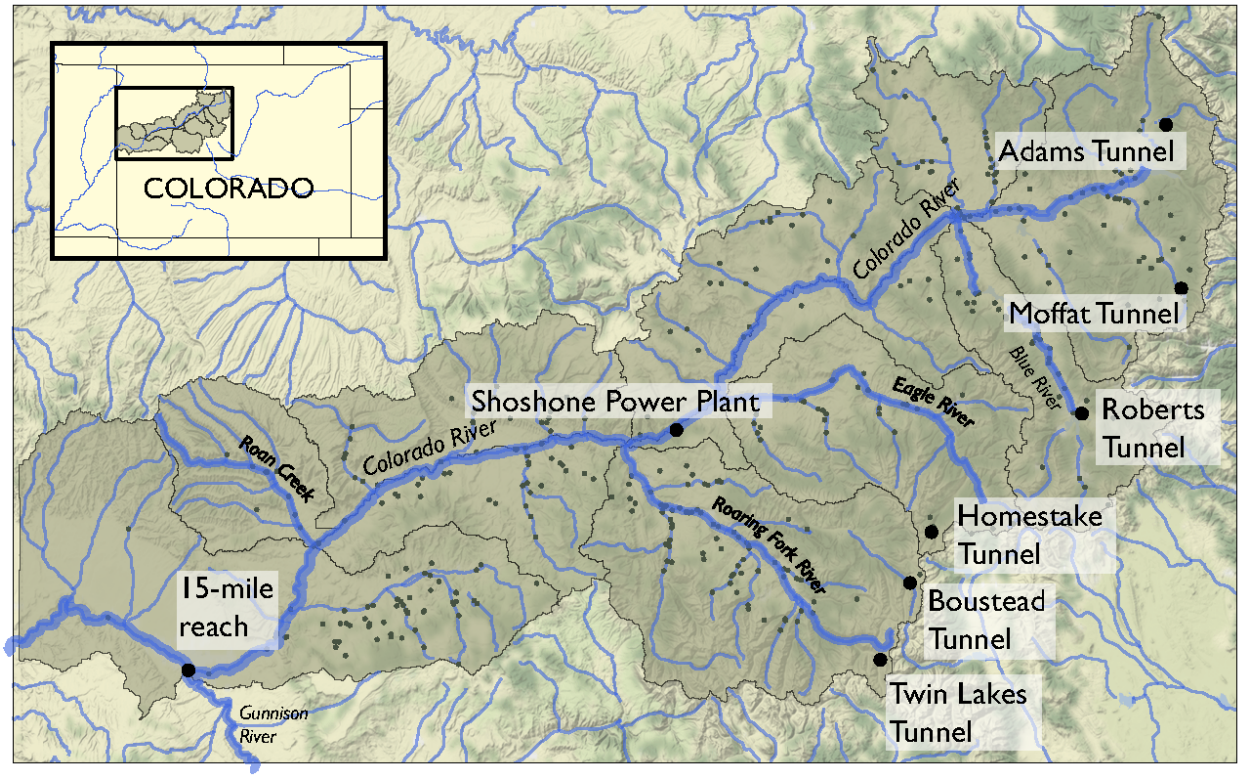
Note: Run the tutorial interactively: Logistic Regression Tutorial

8.3.1 Tutorial: Logistic Regression for Factor Mapping

This tutorial replicates a scenario discovery analysis performed in Hadjimichael et al. (2020).

8.3.1.1 Background

Planners in the the Upper Colorado River Basin (UCRB, shown in the figure below) are seeking to understand the vulnerability of water users to uncertainties stemming from climate change, population growth and water policy changes. The UCRB spans 25,682 km2 in western Colorado and is home to approximately 300,000 residents and 1,012 km2 of irrigated land. Several thousand irrigation ditches divert water from the main river and its tributaties for irrigation (shown as small black dots in the figure). Transmountain diversions of approximately 567,4000,000 m3 per year are exported for irrigation, industrial and municipal uses in northern and eastern Colorado, serving the major population centers of Denver and Colorado Springs. These diversions are carried through tunnels, shown as large black dots in the figure.



An important planning consideration is the water rights of each user, defined by seniority across all water uses (irrigation diversions, transboundary diversions, power plants etc.) in the basin. To assess the vulnerability of users with varying degrees of water rights seniority, planners simulate the system across an ensemble of scenarios using the state of Colorado's StateMod platform. The model simulates streamflow, diversions, instream demands, and reservoir operations.

Focusing on decision-relevant metrics, the scenario discovery is applied to the water shortages experienced by each individual user (i.e., not on a single basin-wide or sector-wide metric). For this training example, we'll be performing scenario discovery for three different water users, two irrigation users and one municipal user.

Before we start our analysis, we'll load the relevant Python libraries and create lists storing the names of the users of interest.

```
import msdbook

import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

# load example data
msdbook.install_package_data()

all_IDs = ['7000550','7200799','3704614'] # IDs for three that we will perform the␣
↪analysis for
usernames = ['Medium seniority irrigation',
             'Low seniority irrigation',
             'Transbasin municipal diversion']
nStructures = len(all_IDs)
```

```
Downloading example data for msdbook version 0.1.5...
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/uncertain_
↪params_bounds.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_metric_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/solutions.
↪resultfile
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LHsamples_
↪original_1000.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/param_values.
↪csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/collapse_
↪days.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_params_
↪256samples.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LeafCatch.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_
↪simulations_256samples.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/Robustness.
↪txt
```

8.3.1.2 Step 1: Load Latin Hypercube Sample and set up problem

To examine regional vulnerability, we generate an ensemble of plausible future states of the worlds (SOWs) using Latin Hypercube Sampling. For this tutorial, we'll load a file containing 1,000 parameter samples.

```
LHsamples = msdbook.load_lhs_basin_sample()
param_bounds = msdbook.load_basin_param_bounds()

param_names=['Irrigation demand multiplier','Reservoir loss','Transbasin demand␣
↪multiplier',
             'Municipal & industrial multiplier', 'Shoshone','Environmental flows',
             'Evaporation change','Mean dry flow','Dry flow variance',
             'Mean wet flow','Wet flow variance','Dry-dry probability',
             'Wet-wet probability', 'Snowmelt shift']
```

8.3.1.3 Step 2: Define decision-relevant metrics for illustration

Scenario discovery attempts to identify parametric regions that lead to 'success' and 'failure'. For this demonstration we'll be defining 'success' as states of the world where a shortage level doesn't exceed its historical frequency.

8.3.1.4 Step 3: Run the logistic regression

Logistic regression estimates the probability that a future SOW will be classified as a success or failure given a set of performance criteria. A logistic regression model is defined by:

$$ln\left(\frac{p_i}{1-p_i}\right) = X_i^T\beta$$

where p_i is the probability the performance in the i^{th} SOW will be classified as a success, X_i is the vector of covariates describing the i^{th} SOW, and β is the vector of coefficients describing the relationship between the covariates and the response, which here will be estimated using maximum likelihood estimation.

A logistic regression model was fit to the ensemble of SOWs using the performance criteria defined in step 2. Logistic regression modeling was conducted using the Statsmodel Python package. The data required for the full analysis is too large to include in this tutorial, but results can be found in the data file loaded below.

The results files contain the occurence of different frequency and magnitude combinations under the experiment, in increments of 10, between 0 and 100. These combinations (100 for each user) are alternative decision-relevant metrics that can be used for scenario discovery.

```
# Set arrays for shortage frequencies and magnitudes
frequencies = np.arange(10, 110, 10)
magnitudes = np.arange(10, 110, 10)
realizations = 10


# Load performance and pseudo r scores for each of the users
results = [msdbook.load_user_heatmap_array(all_IDs[i]) / 100 for i in range(len(all_
↪IDs))]
```

8.3.1.5 Step 4: Factor ranking

To rank the importance of each uncertain factor, we utilize McFadden's psuedo-R2, a measure that quantifies the improvement of the model when utilizing each given predictor as compared to prediction using the mean of the data set:

$$R^2_{McFadden} = 1 - \frac{ln\hat{L}(M_{full})}{ln\hat{L}(M_{intercept})}$$

Where $ln\hat{L}(M_{full})$ is the log likelihood of the full model (including the predictor) and $ln\hat{L}(M_{intercept})$ is the log likelihood of the intercept model (which predicts the mean probability of success across all SOWs).

Higher values of McFadden's psuedo-R2 indicate higher factor importance (when the likelihood of the full model approaches one, the ratio of the likelihood of the full model compared to the intercept model will get very small).

```
scores = [msdbook.load_user_pseudo_scores(all_IDs[i]) for i in range(len(all_IDs))]

freq = [1,0,0]
mag = [7,3,7]
```

8.3.1.6 Step 5: Draw factor maps

The McFadden's psuedo-R2 scores files contain preliminary logistic regression results on parameter importance for each of these combinations. Using these psuedo-R2 scores we will identify the two most important factors for each metric which we'll use to generate the final scenario discovery maps (note: there may be more than two important metrics for each user, but here we will demonstrate by mapping two).

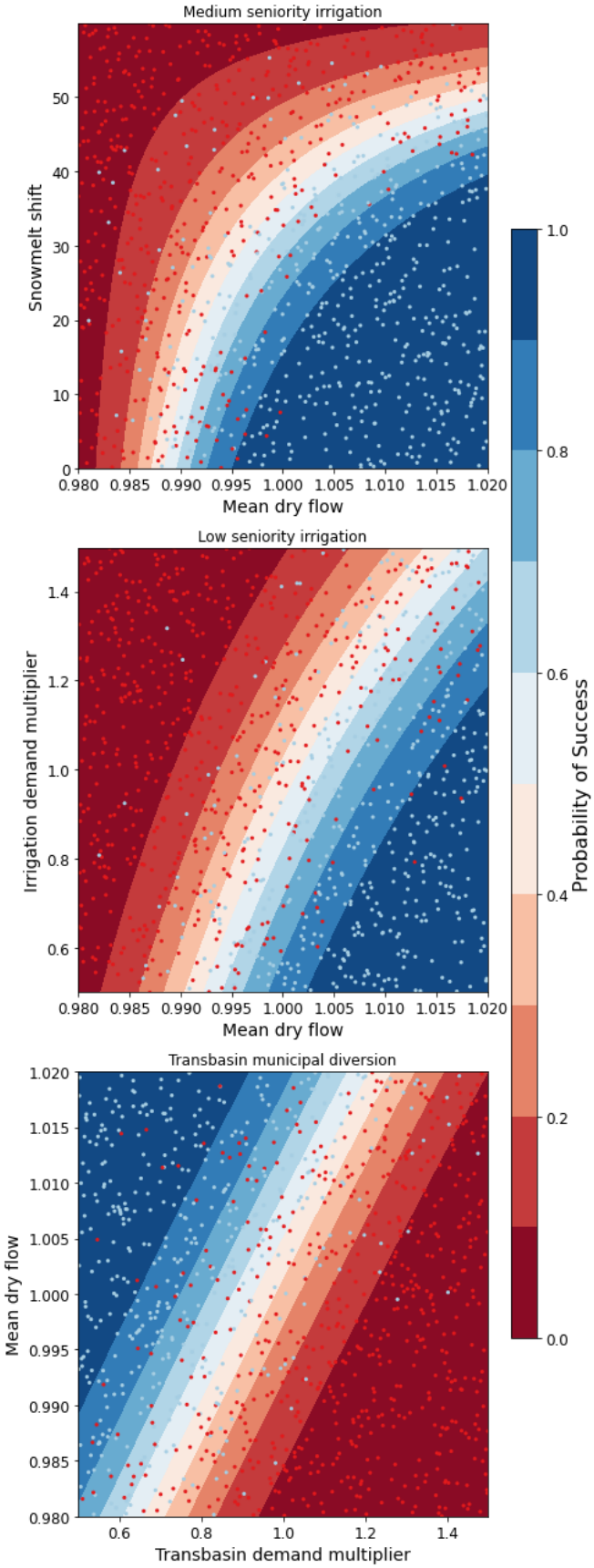
```
# setup figure
fig, axes = plt.subplots(3,1, figsize=(6,18), tight_layout=True)
fig.patch.set_facecolor('white')

for i in range(len(axes.flat)):

    ax = axes.flat[i]

    allSOWsperformance = results[i]
    all_pseudo_r_scores = scores[i]

    # construct dataframe
    dta = pd.DataFrame(data=np.repeat(LHsamples, realizations, axis = 0), columns=param_
↪names)
    dta['Success'] = allSOWsperformance[freq[i],mag[i],:]

    pseudo_r_scores = all_pseudo_r_scores[str(frequencies[freq[i]])+'yrs_
↪'+str(magnitudes[mag[i]])+'prc'].values
    top_predictors = np.argsort(pseudo_r_scores)[::-1][:2] #Sort scores and pick top 2␣
↪predictors

    # define color map for dots representing SOWs in which the policy
    # succeeds (light blue) and fails (dark red)
    dot_cmap = mpl.colors.ListedColormap(np.array([[227,26,28],[166,206,227]])/255.0)

    # define color map for probability contours
```

(continues on next page)

(continued from previous page)

```
    contour_cmap = mpl.cm.get_cmap('RdBu')

    # define probability contours
    contour_levels = np.arange(0.0, 1.05,0.1)

    # define base values of the predictors
    SOW_values = np.array([1,1,1,1,0,0,1,1,1,1,1,0,0,0]) # default parameter values for␣
↪base SOW
    base = SOW_values[top_predictors]
    ranges = param_bounds[top_predictors]

    # define grid of x (1st predictor), and y (2nd predictor) dimensions
    # to plot contour map over
    xgrid = np.arange(param_bounds[top_predictors[0]][0],
                      param_bounds[top_predictors[0]][1], np.around((ranges[0][1]-
↪ranges[0][0])/500,decimals=4))
    ygrid = np.arange(param_bounds[top_predictors[1]][0],
                      param_bounds[top_predictors[1]][1], np.around((ranges[1][1]-
↪ranges[1][0])/500,decimals=4))
    all_predictors = [ dta.columns.tolist()[i] for i in top_predictors]
    dta['Interaction'] = dta[all_predictors[0]]*dta[all_predictors[1]]

    # logistic regression here
    predictor_list = [all_predictors[i] for i in [0,1]]
    result = msdbook.fit_logit(dta, predictor_list)

    # plot contour map
    contourset = msdbook.plot_contour_map(ax, result, dta, contour_cmap,
                                          dot_cmap, contour_levels, xgrid,
                                          ygrid, all_predictors[0], all_predictors[1],␣
↪base)

    ax.set_title(usernames[i])

# set up colorbar
cbar_ax = fig.add_axes([0.98, 0.15, 0.05, 0.7])
cbar = fig.colorbar(contourset, cax=cbar_ax)
cbar_ax.set_ylabel('Probability of Success', fontsize=16)
cbar_ax.tick_params(axis='y', which='major', labelsize=12)
```

```
Optimization terminated successfully.
         Current function value: 0.378619
         Iterations 8
Optimization terminated successfully.
         Current function value: 0.397285
         Iterations 8
Optimization terminated successfully.
         Current function value: 0.377323
         Iterations 8
```


Medium seniority irrigation
Snowmelt shift
Mean dry flow
Low seniority irrigation
Irrigation demand multiplier
Mean dry flow
Transbasin municipal diversion
Mean dry flow
Transbasin demand multiplier
Probability of Success

The figure above demonstrates how different combinations of the uncertain factors lead to success or failure in different states of the world, which are denoted by the blue and red dots. The probability of success and failure are further denoted by the contours in the figure. Several insights can be drawn from this figure.

First, using metrics chosen to be decision-relevant (specific to each user) causes different factors to be identified as most important by this scenario-discovery exercise (the x- and y-axes for each of the subplots). In other words, depending on what the decision makers of this system want to prioritize they might choose to monitor different uncertain factors to track performance.

Second, in the top panel, the two identified factors appear to also have an interactive effect on the metric used (shortages of a certain level and frequency in this example). In terms of scenario discovery, the Patient Rule Induction Method (PRIM) or Classification And Regression Trees (CART) would not be able to delineate this non-linear space and would therefore misclassify parameter combinations as 'desirable' when they were in fact undesirable, and vice versa.

Lastly, logistic regression also produces contours of probability of success, i.e. different factor-value combinations are assigned different probabilities that a shortage level will be exceeded. This allows the decision makers to evaluate these insights while considering their risk aversion.

8.4 HYMOD Dynamics Tutorial

Note: Run the tutorial interactively: HYMOD Notebook

8.4.1 Tutorial: Sensitivity Analysis of the HYMOD Model

This tutorial has been developed to showcase some of the sensitivity analysis and uncertainty quantification concepts and tools established in the main text in the context of HYMOD, a rainfall-runoff model. The tutorial includes the following sections:

8.4.1.1 Introduction to HYMOD and Sensitivity Analysis

1- Introduction to a simple hydrologic model (HYMOD) *2- An overview of sensitivity analysis using SALib* 3- Calculation of sensitivity analysis metrics

8.4.1.2 Time-Varying Sensitivity Analysis

4- Time-varying sensitivity analysis

8.4.1.3 Ensemble-based Parametric Uncertainty

5- Generalized Likelihood Uncertainty Estimation (GLUE) *6- Pre-Calibration*

It is important to note that, although in this tutorial we focus on HYMOD, which is a hydrologic model, it can also be thought of as an example of a model that abstracts complex non-linear systems. Therefore, many of the methods that we use in the tutorial can be applied to numerical models that simulate other complex non-linear systems.

8.4.2 Introduction to HYMOD and Sensitivity Analysis

8.4.3 1- HYMOD

HYMOD is a simple hydrologic model (rainfall-runoff model) that simulates key hydrologic fluxes such as infiltration, streamflow and evapotranspiration. The model has been originally developed and used for river flow forecasting. However, in the last two decades the model has been widely used to explore different sensitivity analysis (e.g., Herman et al., 2013), uncertainty quantification (e.g., Smith et al., 2008), and optimization (e.g., Ye et al., 2014) concepts.

There are two main assumptions in the model: 1) Rainfall is generated through infiltration excess overland flow. 2) Runoff generation can be formulated by the probability-distributed principle (Moore, 1985). Therefore the cumulative rate storage capacity $(F(C))$ can be calculated using the following relationship:

$$F(C) = 1 - (1 - \frac{C}{C_{MAX}}) - B_{exp}$$

where *:math:`C`* is the water storage capacity; **:math:`C_{MAX}`** is the parameter describing basin maximum water storage capacity (mm); and **:math:`B_{exp}`** is the parameter describing the degree of spatial variability within the basin.

The portion of precipitation that exceeds the water storage capacity is treated as runoff. The evapotranspiration is equal to potential evapotranspiration (PET) if enough water is available. Otherwise, it equals the available water storage.

Then, based on a parameter $Alpha$, the runoff is divided into quick flow and slow flow, which are routed through three identical quick flow tanks $Q1, Q2, Q3$ and a parallel slow flow tank, respectively.

The flow rates in the routing system are described by the resident time in the quick tanks K_q (day) and the slow tank K_s (day), respectively.

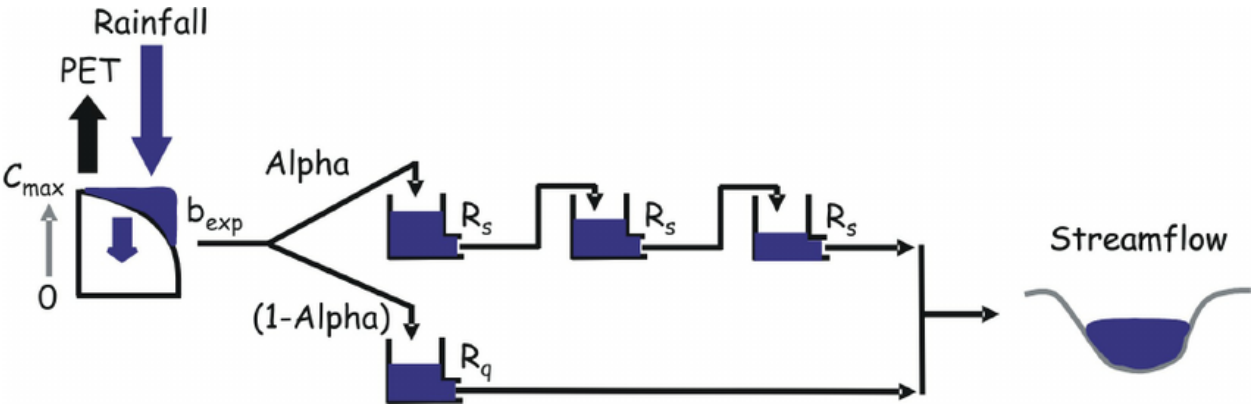


Fig. 8.1: alt text

Vrugt et al., (2008)

8.4.3.1 1-1 Model Parameters

C_{MAX}: parameters describing basin maximum water storage capacity (mm)

B_{exp}: parameters describing the degree of spatial variability within the basin between 0 and Huz

Alp: Fraction of runoff contributing to quick flow

K_q: Quick flow residence time of linear infinite reservoir (the Kq values of all three linear reservoirs are the same)

K_s: Slow flow residence time of linear infinite reservoir

8.4.3.2 1-2 Input data

The HYMOD model only requires precipitation and potential evapotranspiration as inputs. The Leaf River example that we use here is also a widely used test case of HYMOD. The dataset also includes observed runoff that we later use to evaluate the performace of each sensitvity analysis sample set.

We can use the following to read the input file:

```
import msdbook

import numpy as np
import pandas as pd
import seaborn as sns

from sklearn import metrics
from matplotlib import pyplot as plt

# load example data
msdbook.install_package_data()
```

```
Downloading example data for msdbook version 0.1.5...
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/uncertain_
↪params_bounds.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_metric_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/solutions.
↪resultfile
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LHsamples_
↪original_1000.txt
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/3704614_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/param_values.
↪csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_mth_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/collapse_
↪days.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_params_
↪256samples.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_vary_s1.
↪npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7000550_
↪heatmap.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪heatmap.npy
```

(continues on next page)

(continued from previous page)

```
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/sa_by_yr_
↪delta.npy
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/7200799_
↪pseudo_r_scores.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/LeafCatch.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/hymod_
↪simulations_256samples.csv
Unzipped: /srv/conda/envs/notebook/lib/python3.7/site-packages/msdbook/data/Robustness.
↪txt
```

```
# load the HYMOD input file
leaf_data = msdbook.load_hymod_input_file()

# extract the first eleven years of data
leaf_data = leaf_data.iloc[0:4015].copy()
```

```
# There are only three columns in the file including precipitation, potential␣
↪evapotranspiration and  streamflow
leaf_data.head()
```

8.4.3.3 1-3 Baseline Model Simulation

We can start our sensitivity analysis experiment with running HYMOD using its default parameters.

```
# assign input parameters to generate a baseline simulated streamflow
Nq = 3  # Number of quickflow routing tanks
Kq = 0.5 # Quickflow routing tanks' rate parameter
Ks =  0.001 # Slowflow routing tank's rate parameter
Alp = 0.5 # Quick/slow split parameter
Huz = 100 # Maximum height of soil moisture accounting tank
B = 1.0 # Scaled distribution function shape parameter

# Note that the number of years is 11 years. One year of model warm-up and ten years are␣
↪used for actual simulation
model = msdbook.hymod(Nq, Kq, Ks, Alp, Huz, B, leaf_data, ndays=4015)
```

8.4.3.4 1-4 Model Outputs

Model outputs include actual evapotranspiration, quick and fast streamflow, and combined runoff. In this tutorial we focus on the total daily runoff ($m-3/s$). We can use the following script to plot simulated streamflow against observed streamflow.

8.4.3.5 Variables

PP: Precipitation

ET: Evapotranspiration

OV: Runoff

Qq: Quick Flow

Qs: Slow Flow

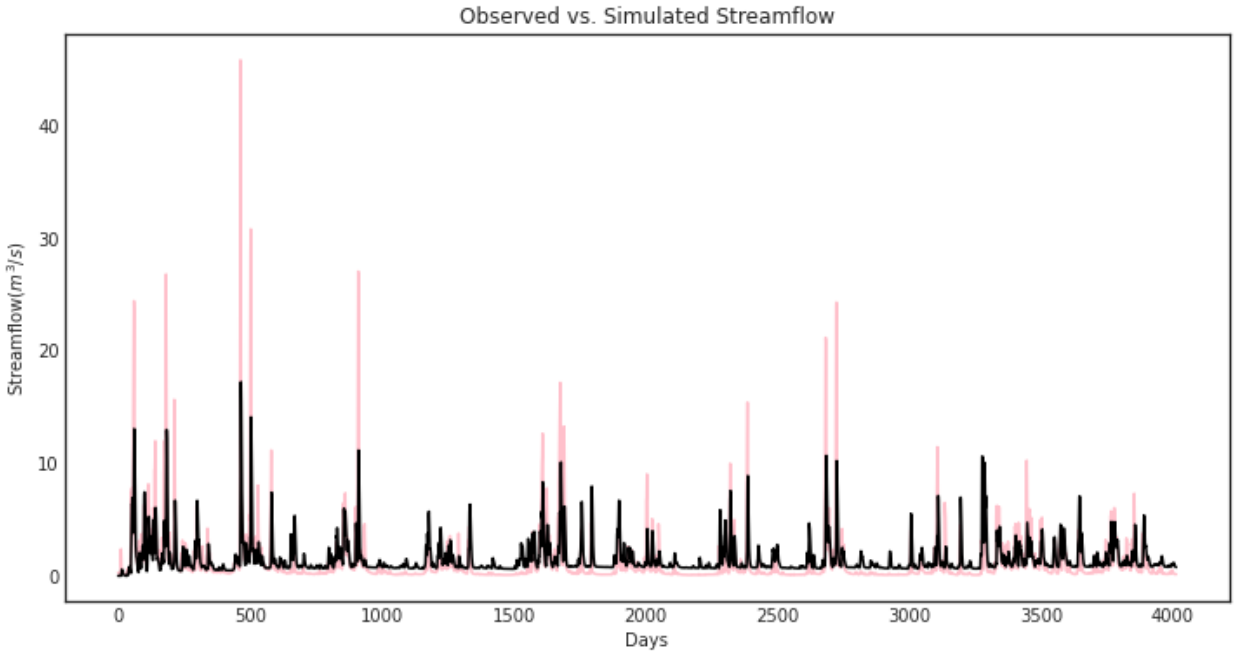
QQ: Streamflow (Quick Flow + Slow Flow)

XHuz and **XCuz**: Current moisture state of soil moisture accounting component (as depth XH or volume XC)

8.4.3.6 Plot the observed versus simulated streamflow.

Note: **Tip:** View the source code used to create this plot here: plot_observed_vs_simulated_streamflow

```
ax = msdbook.plot_observed_vs_simulated_streamflow(df=leaf_data, hymod_dict=model)
```



8.4.4 2- Sensitivity Analysis

Here we use the SALib Python library to explore how different HYMOD input parameters affect model streamflow simulations. For this exercise, we only use Sobol variance-based method. The following commands can be used to import SALib

```
from SALib.sample import saltelli
from SALib.analyze import sobol
from SALib.analyze import delta
```

8.4.4.1 2-2 Model simulations for sensitivity analysis

We first define the model input and their ranges.

```
problem_hymod = {
    'num_vars': 5,
    'names': ['Kq', 'Ks', 'Alp', 'Huz', 'B'],
    'bounds': [[0.1, 1],  # Kq
               [0, 0.1],  # Ks
               [0, 1],    # Alp
               [0.1, 500],  # Huz
               [0, 1.9]]  # B
}
```

Now we need to sample and then run the model for each of the sample sets. We will load a sample that has already been created `param_values_hymod` for demonstration purposes. The actual model simulation takes an extended period, so we also load the simulation data from a previous run. The following demonstrates how to conduct this analysis:

```
# generate 256 samples. This is an arbitrary number.
param_values_hymod = saltelli.sample(problem_hymod, 256)

# dictionary to store outputs in
d_outputs = {}

# run simulation for each parameter sample
for i in range(0, len(param_values_hymod)):

    # run model for each sensitivity analysis parameter sets
    hymod_output = msdbook.hymod(Nq,
                                 param_values_hymod[i, 0],
                                 param_values_hymod[i, 1],
                                 param_values_hymod[i, 2],
                                 param_values_hymod[i, 3],
                                 param_values_hymod[i, 4],
                                 leaf_data,
                                 ndays=4015)

    # store the simulated total flow discharge
    d_outputs[f"Q{i}"] = hymod_output["Q"]


Q_df_bw = pd.DataFrame(d_outputs)
```

```
# load previously generated parameter values
param_values_hymod = msdbook.load_hymod_params()

# number of samples
n_samples = len(param_values_hymod)

# load previously generated hymod simulated outputs
Q_df_bw = msdbook.load_hymod_simulation()

# column names of each sample simulation number
sample_column_names = [i for i in Q_df_bw.columns if i[0] == 'Q']
```

Model Warm-up

A hydrological model such as HYMOD usually includes ordinary differential equations that are sensitive to their initial condition. They also have components in their underlying formulation that have long memory such that prior time steps can affect their current simulations. For example, soil moisture or groundwater can hold water for a long time and therefore they are often considered to exhibit a long memory. This can affect the partitioning of water to runoff and infiltration, while also controlling the generation of base flow. Therefore, it is important to have a reasonable initial value for them. To achieve this, hydrologists usually extend their simulation period and after the simulations, they remove that extended time period that has unreasonable groundwater or surface water values. This time period is called the warm-up time period.

Here we extended our simulation for one year (from 10 years to 11 years) and we removed the first year of simulation, therefore our warm-up period is one year.

```
# exclude the first year of simulation from the simulations and reset the index
Q_df = Q_df_bw.iloc[365:4015].copy().reset_index(drop=True)

# exclude the first year of the input data and reset the index
leaf_data = leaf_data.iloc[365:4015].copy().reset_index(drop=True)
```

8.4.4.2 2-3 Visual inspection of the model outputs

Here we create a figure that shows HYMOD streamflow outputs under different sample sets, and compare them with the observed streamflow.

```
# add date columns to our simulation data frame; for this data our start date is 1/1/2000
date_ts = pd.date_range(start='1/1/2000', periods=3650, freq='D')
Q_df['date'] = date_ts
Q_df['year'] = date_ts.year
Q_df['month'] = date_ts.month
Q_df['day'] = date_ts.day

# aggregate the simulated observed streamflow to monthly mean
df_sim_mth_mean = Q_df.groupby(['year', 'month'])[sample_column_names].mean()

# do the same for the observed data
date_ts = pd.date_range(start='1/1/2000', periods=len(leaf_data), freq='D')
leaf_data['date'] = date_ts
leaf_data['year'] = date_ts.year
```

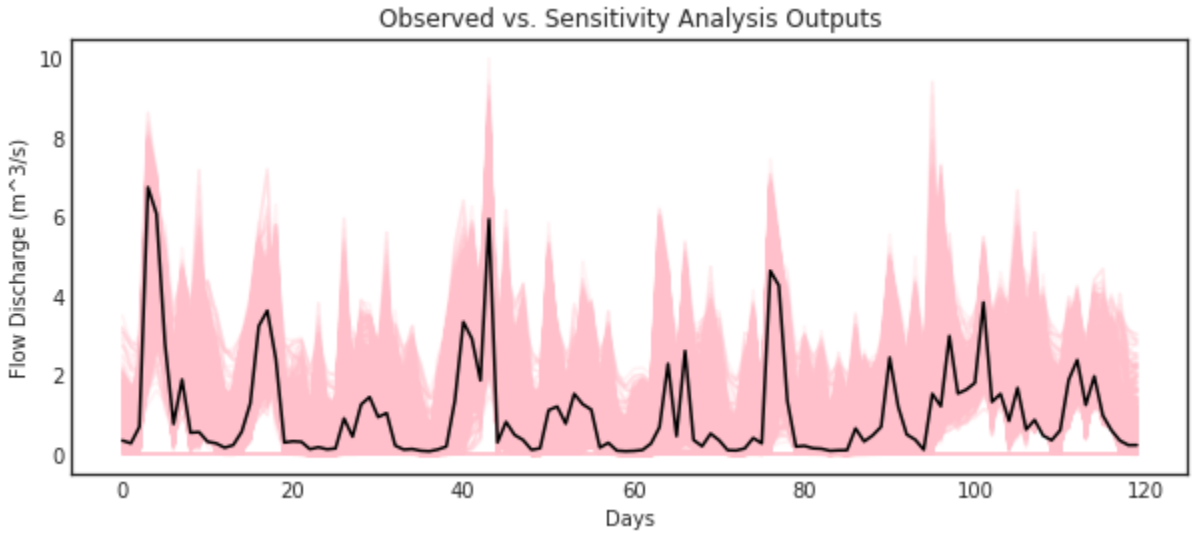
(continues on next page)

(continued from previous page)

```
leaf_data['month'] = date_ts.month
leaf_data['day'] = date_ts.day

# aggregate the daily observed streamflow to monthly mean
df_obs_mth_mean = leaf_data.groupby(['year', 'month']).mean()
```

Note: **Tip:** View the source code used to create this plot here: plot_observed_vs_sensitivity_streamflow

```
ax = msdbook.plot_observed_vs_sensitivity_streamflow(df_obs=df_obs_mth_mean,
                                                     df_sim=df_sim_mth_mean)
```



8.4.5 3- Calculation of Sensitivity Analysis Indices

There are different options to calculate sensitivity indices. The following section aggregates model streamflow outputs and calculates the sensitivity indices.

8.4.5.1 3-1 Aggregated sensitivity analysis indices

This is the simplest way of calculating sensitivity analysis metrics, however, averaging all model response can lead to loss of information that we further explore in the following sections.

```
# overall aggregated indices
Y = Q_df[sample_column_names].mean().to_numpy()

# Perform analysis
Si = delta.analyze(problem_hymod, param_values_hymod, Y, print_to_console=False)
```

```
print('First order indices = ', Si['S1'])
```

```
First order indices =  [0.00810372 0.0049972  0.00508833 0.60039872 0.28942293]
```

```
Si['S1'].sum()
```

```
0.9080109105125653
```

8.4.5.2 3-2 How do different performance metrics affect the results of our sensitivity analysis?

Streamflow has many different properties. In this section, we discuss how the selection of metrics can lead to fundamentally different sensitivity analysis results. For example, one can only focus on aggregated streamflow metrics such as mean (what has been presented so far), or only on extreme events such as drought or floods.

Here we compare three different metrics: 1- Mean error (ME) 2- Root Mean Square Error (RMSE) 3- Log-Root Mean Square Error (Log(RMSE))

Each of these metrics focuses on a specific attribute of streamflow. For example, RMSE highlights the impacts of extreme flood events, while LOG(RMSE) focuses on model performance during low-flow events.

```
# calculate error metrics
me = Q_df[sample_column_names].apply(lambda x: (x-leaf_data["Strmflw"]), axis=0)
mse = Q_df[sample_column_names].apply(lambda x: metrics.mean_squared_error(x, leaf_data[
↪"Strmflw"]), axis=0)
rmse = mse**(1/2)

# add error metrics to a dictionary
d_metrics = {'ME': me.mean().values,
             'RMSE': rmse.values,
             'LOG[RMSE]': np.log10(rmse.values)}

# convert to a dataframe
df_metrics_SA = pd.DataFrame(d_metrics)
```

We can use the following to calculate the SA indices for each metric and visualize it. Results are pre-loaded for efficiency.

```
# performance analysis
df_metric_sa_result = pd.DataFrame(np.zeros((3, 5)), columns=['Kq', 'Ks', 'Alp', 'Huz',
↪'B'])

# conduct sensitivity analysis for each metric
for index, i in enumerate(d_metrics.keys()):

    # get the data as a numpy array for the target metric
    Y = d_metrics[i]

    # use the metric to conduct SA
    Si = delta.analyze(problem_hymod, param_values_hymod, Y, print_to_console=False)

    # add the sensitivity indices to the output data frame
    df_metric_sa_result.iloc[index, :] = Si['S1']
```

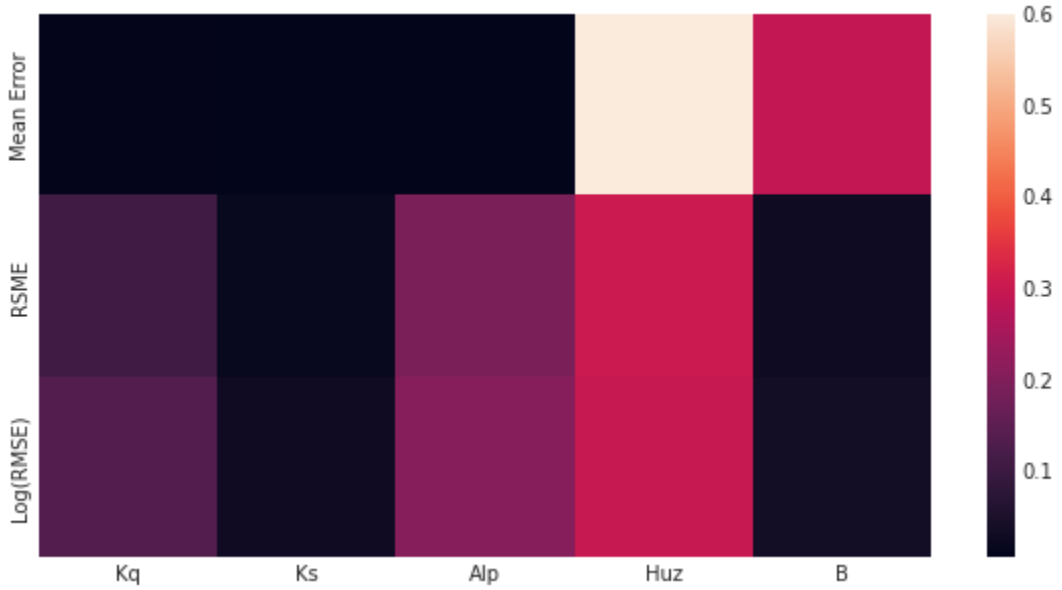
```
# load previously ran simulation
df_metric_sa_result = msdbook.load_hymod_metric_simulation()
```

(continues on next page)

(continued from previous page)

```
# view results
df_metric_sa_result
```

```
# create seaborn heatmap with required labels
plt.subplots(figsize=(10, 5))

# labels for y-axis
y_axis_labels = ['Mean Error', 'RSME', 'Log(RMSE)']

# plot heatmap
ax = sns.heatmap(df_metric_sa_result, yticklabels=y_axis_labels,  cmap='rocket')
```



The results indicate that different goodness-of-fit metrics can produce different sensitivity indices. This is because streamflow time series have several dimensions and regimes (e.g., extreme high flow and low flow) and focusing on only one metric will neglect the sensitivity of other dimensions.

Therefore, we can argue that a single goodness-of-fit measure will never be able to capture the entire response of model to different parametric combinations. For more discussion about this topic readers can refer to Liu and Sun (2010) and Foglia et al., (2009).

8.4.6 4- Time-Varying Sensitivity Analysis

Hydrological processes are often state-dependent, meaning that their responses are affected by the time-varying condition that they are in. For example, rainfall-runoff processes are different in winter and summer. These processes are also different during wet years and dry years.

Hydrological processes are also path-dependent, meaning that previous time-steps on the model affect the present and future simulation of different hydrologic components. To take these properties into account, we can zoom into different time periods to explore how the sensitivity of model parameters evolve in different time steps. This is referred to as time-varying sensitivity analysis.

For more information about time-varying sensitivity analysis, readers can refer to Herman et al. (2013) and Xu et al. (2018).

8.4.6.1 4-1 Sensitivity analysis indices for each month

```
# aggregate simulated streamflow data to monthly time series
df_sim_by_mth_mean = Q_df.groupby('month')[sample_column_names].mean()

# aggregate observed streamflow data to monthly time series
df_obs_by_mth_mean = leaf_data.groupby('month').mean()
```

We can use the following to calculate the SA indices for each month and visualize it. Results are pre-loaded for efficiency.

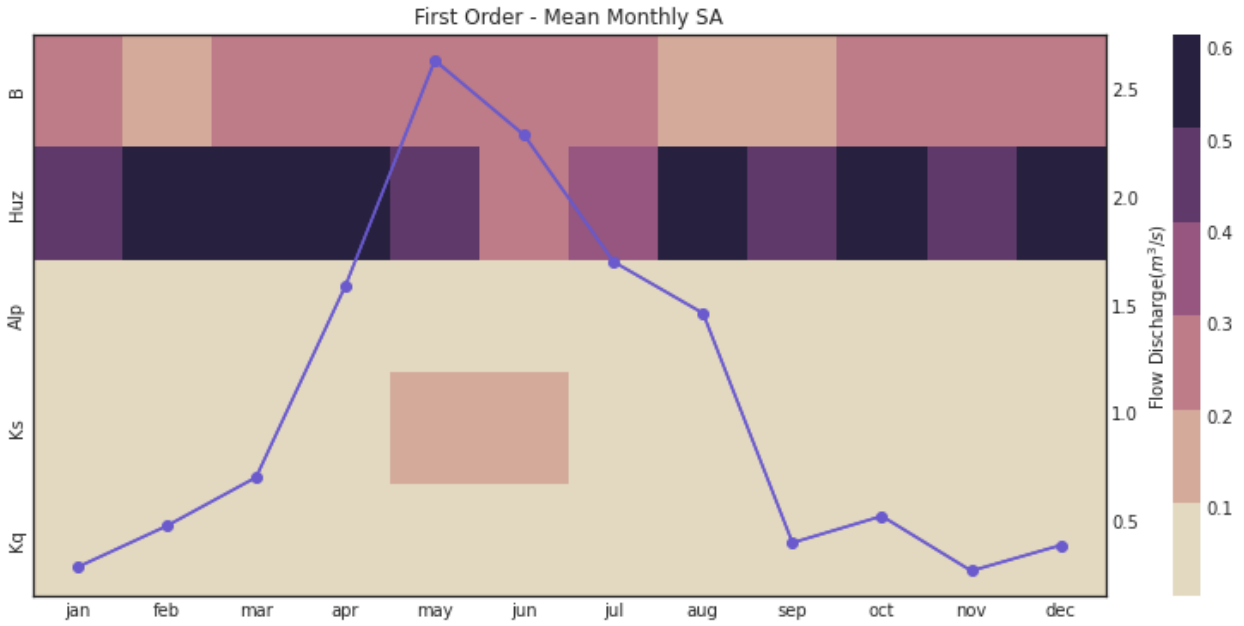
```
# set up dataframes to store outputs
df_mth_s1 = pd.DataFrame(np.zeros((12,5)), columns=['Kq', 'Ks', 'Alp', 'Huz', 'B'])
df_mth_delta = df_mth_s1.copy()

# iterate through each month
for i in range(0, 12):

    # generate the simulation data
    Y = df_sim_by_mth_mean.iloc[i, :].to_numpy()

    # run SA
    Si = delta.analyze(problem_hymod, param_values_hymod, Y, print_to_console=False)

    # add to output dataframes
    df_mth_s1.iloc[i, :] = np.maximum(Si['S1'], 0)
    df_mth_delta.iloc[i, :] = np.maximum(Si['delta'], 0)

# convert to arrays
arr_mth_s1 = df_mth_s1.values
arr_mth_delta = df_mth_delta.values
```

First-order Indices

The following can be used to visualize the time-varying first-order indices. The first order represents the direct impacts of a specific parameter on model outputs.

Note: Tip: View the source code used to create this plot here: plot_monthly_heatmap

```
# load previously ran data
arr_mth_delta, arr_mth_s1 = msdbook.load_hymod_monthly_simulations()

# plot figure
ax, ax2 = msdbook.plot_monthly_heatmap(arr_sim=arr_mth_s1.T,
                                       df_obs=df_obs_by_mth_mean,
                                       title='First Order - Mean Monthly SA')
```

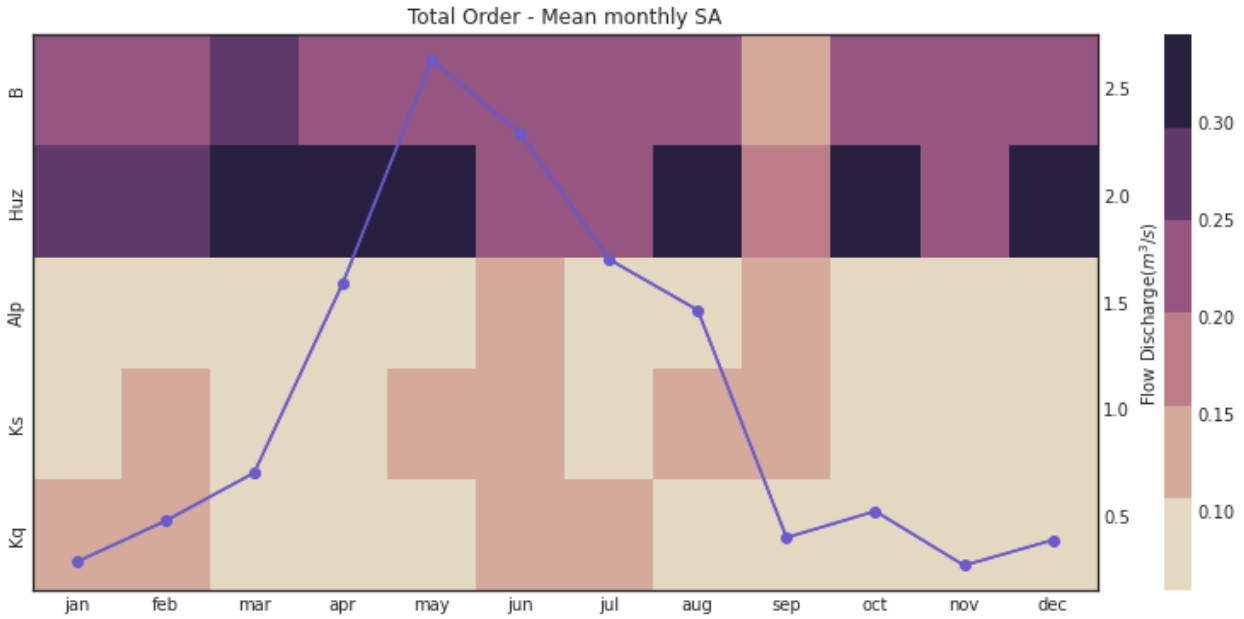


This figure demonstrates the first order sensitivity indices when the streamflow data are aggregated by month. The purple line represents the observed monthly discharge. The figure indicates that the first order indices are highest for B and Huz across all months and lowest for Alp, Ks, and Kq.

Total-order indices

We can also focus on the total order sensitivity index that includes first-order SA indices and interactions between parameters

```
# plot figure
ax, ax2 = msdbook.plot_monthly_heatmap(arr_sim=arr_mth_delta.T,
                                       df_obs=df_obs_by_mth_mean,
                                       title='Total Order - Mean monthly SA')
```



Notably, the total order sensitivity results are different than the first order sensitivity results, which indicates that interactions between the parameters (particularly in regards to Kq, Ks, and Alp) contribute to variance in the HYMOD output.

8.4.6.2 4-2 Annual sensitivity analysis indices

```
# group by year and get mean
df_sim_by_yr_mean = Q_df.groupby(['year'])[sample_column_names].mean()

# group input data and get mean
df_obs_by_yr_mean = leaf_data.groupby(['year']).mean()
```

We can also calculate the sensitivity analysis indices for each individual year. This will allow us to understand if model control changes during different years. The following code first aggregates the outputs to annual time steps, and then calculates the SA indices.

```
# set up dataframes to store outputs
df_yr_s1 = pd.DataFrame(np.zeros((10, 5)), columns=['Kq', 'Ks', 'Alp', 'Huz', 'B'])
df_yr_delta = df_yr_s1.copy()

# iterate through each year
for i in range(0, 10):

    # generate the simulation data
    Y = df_sim_by_yr_mean.iloc[i, :].to_numpy()

    # run SA
    Si = delta.analyze(problem_hymod, param_values_hymod, Y, print_to_console=False)

    # add to output dataframes
    df_yr_s1.iloc[i, :] = np.maximum(Si['S1'], 0)
    df_yr_delta.iloc[i, :] = np.maximum(Si['delta'], 0)
```

(continues on next page)

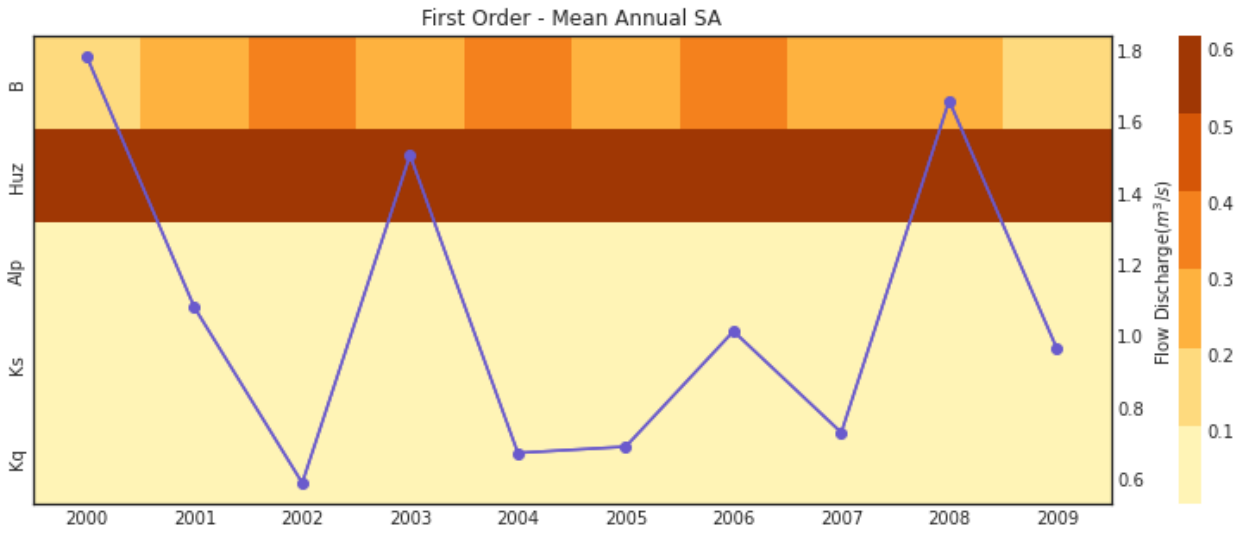
(continued from previous page)

```
# convert to arrays
arr_yr_s1 = df_mth_s1.values
arr_yr_delta = df_mth_delta.values
```

First-order indices

Note: **Tip:** View the source code used to create this plot here: plot_annual_heatmap

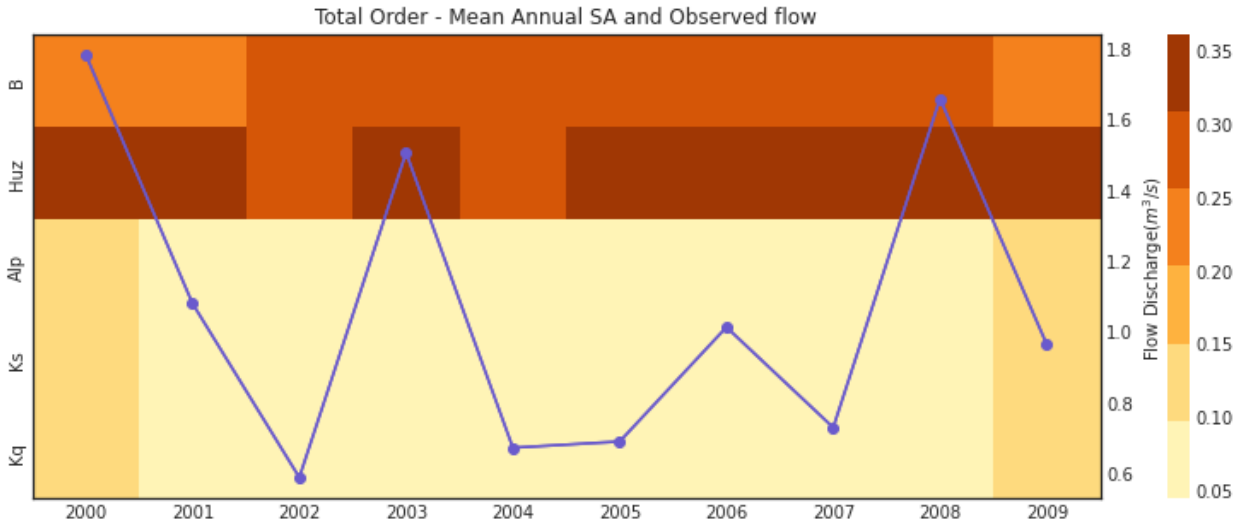
```
# load previously ran data
arr_yr_delta, arr_yr_s1 = msdbook.load_hymod_annual_simulations()

# plot figure
ax, ax2 = msdbook.plot_annual_heatmap(arr_sim=arr_yr_s1.T,
                                      df_obs=df_obs_by_yr_mean,
                                      title='First Order - Mean Annual SA')
```



The first order sensitivities at the annual scale are not unlike the first order monthly sensitivities. Once again, sensitivities vary across year and Huz and B are the most consequential parameters.

Total-order indices

```
# plot figure
ax, ax2 = msdbook.plot_annual_heatmap(arr_sim=arr_yr_delta.T,
                                      df_obs=df_obs_by_yr_mean,
                                      title='Total Order - Mean Annual SA and Observed␣
↪flow')
```



Our results indicate that sensitivity analysis indices vary in different years and now that interactions are included, the Kq, Ks, and Alp variables impact the sensitivity of the streamflow output.

8.4.6.3 4-3 Monthly time-varying sensitivity analysis

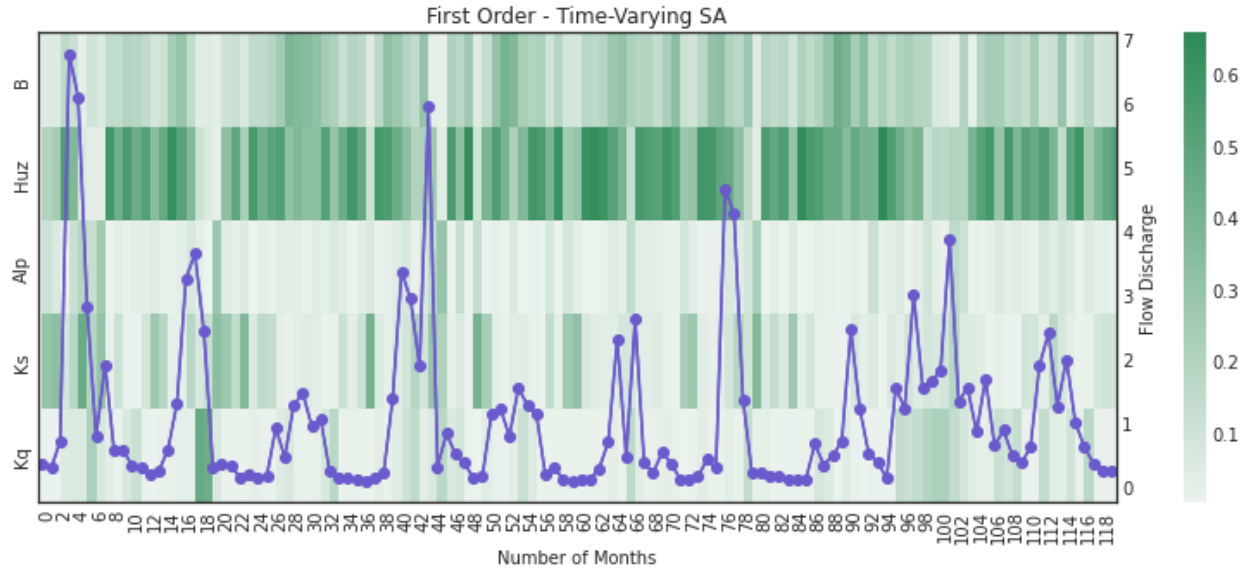
Although time-varying sensitivity analysis at average monthly and average annual temporal resolutions is informative, TVSA is susceptible to the aggregation issue that we discussed earlier in section 3-2. To avoid that we can further discretize our time domain to zoom into individual months. This will provide us with even more information about model behavior and the sensitivity of different parameters in different states of the system. The block of code demonstrates how to implement the monthly TVSA.

```
# set up dataframes to store outputs
df_vary_s1 = pd.DataFrame(np.zeros((df_obs_mth_mean.shape[0], 5)),
                          columns=['Kq', 'Ks', 'Alp', 'Huz', 'B'])

df_vary_delta = df_vary_s1.copy()

# iterate through each month
for i in range(0, df_obs_mth_mean.shape[0]):

    # generate the simulation data
    Y = df_sim_mth_mean.iloc[i, :].to_numpy()

    # run SA
    Si = delta.analyze(problem_hymod, param_values_hymod, Y, print_to_console=False)

    # add to output dataframes
    df_vary_s1.iloc[i, :] = np.maximum(Si['S1'], 0)
    df_vary_delta.iloc[i, :] = np.maximum(Si['delta'], 0)

# convert to arrays
arr_vary_s1 = df_vary_s1.values
arr_vary_delta = df_vary_delta.values
```

First-order indices

Note: **Tip:** View the source code used to create this plot here: plot_varying_heatmap

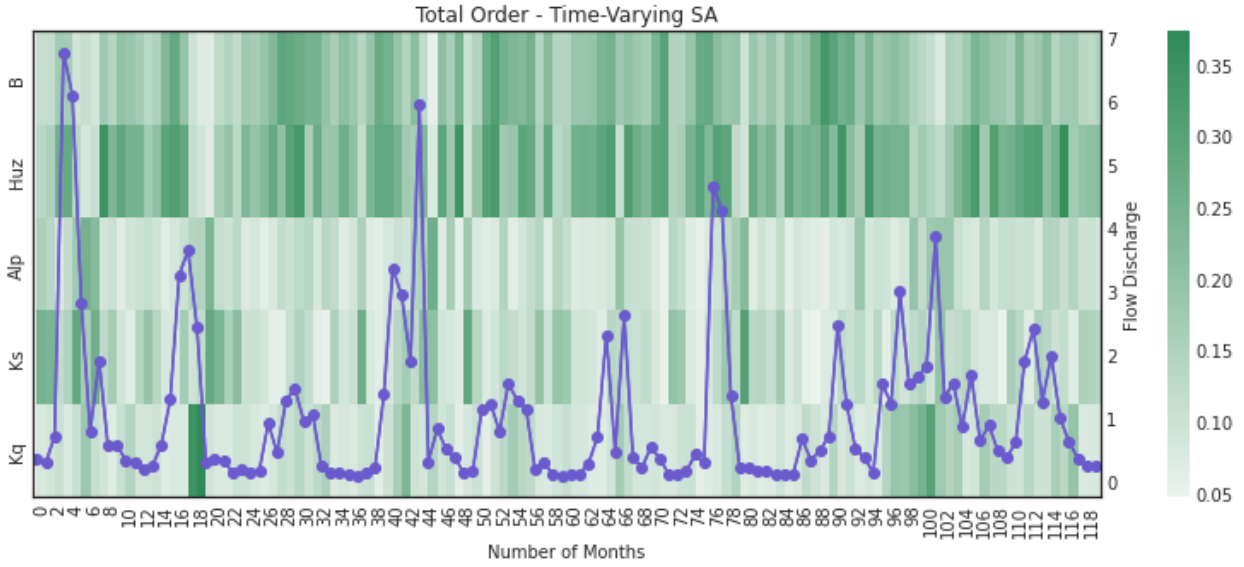
```
# load in previously ran data
arr_vary_delta, arr_vary_s1 = msdbook.load_hymod_varying_simulations()

# plot figure
ax, ax2 = msdbook.plot_varying_heatmap(arr_sim=arr_vary_s1.T,
                                       df_obs=df_obs_mth_mean,
                                       title='First Order - Time-Varying SA')
```



Compared to the TVSA when streamflow was aggregated, this figure suggests that Kq is indeed a relevant parameter for influencing streamflow output when individual months are considered.

Total order - time varying sensitivity analysis

```
# plot figure
ax, ax2 = msdbook.plot_varying_heatmap(arr_sim=arr_vary_delta.T,
                                       df_obs=df_obs_mth_mean,
                                       title='Total Order - Time-Varying SA')
```



As above, the total order sensitivities further indicate the importance of Kq that is not apparent if aggregation is utilized.

8.4.7 Ensemble-based Parametric Uncertainty

8.4.8 5- Generalized Likelihood Uncertainty Estimation (GLUE)

The Generalized Likelihood Uncertainty Estimation (GLUE) is an uncertainty analysis algorithm that has been widely used in hydrologic studies. The main argument behind GLUE is rooted in model calibration and the concept of equifinality. Calibration of complex simulation tools such as hydrological models often produces more than one optimal or near-optimal solutions and these solutions have equivalent chances to be chosen (Beven and Freer, 2001). This situation is called equifinality. GLUE provides a methodological framework to handle this problem and consider more than one optimal calibration set.

GLUE usually includes the following steps (Beven and Bineley, 1992):

1) Definition of a likelihood function
2) Definition of ranges of parameters
3) Sensitivity analysis
4) Calculating likelihood (goodness-of-fit) values for each model simulation
5) Define a threshold and find sample sets that have higher likelihoods than the threshold
6) Visualize the sample sets

8.4.8.1 5-1 Calculation of GLUE metrics

Likelihood calculation (inverse error variance)

There are various likelihood metrics that have been used in previous studies that use GLUE. A widely used example is inverse error variance (IEV; Vrugt et al. 2009 and Beven and Bineley, 1992):

$$IEV = (\sigma_e - 2) - -T = (\frac{SSR}{n-2}) - -T$$

The other metric that can be used as an estimation of likelihood is normalized inverse error variance:

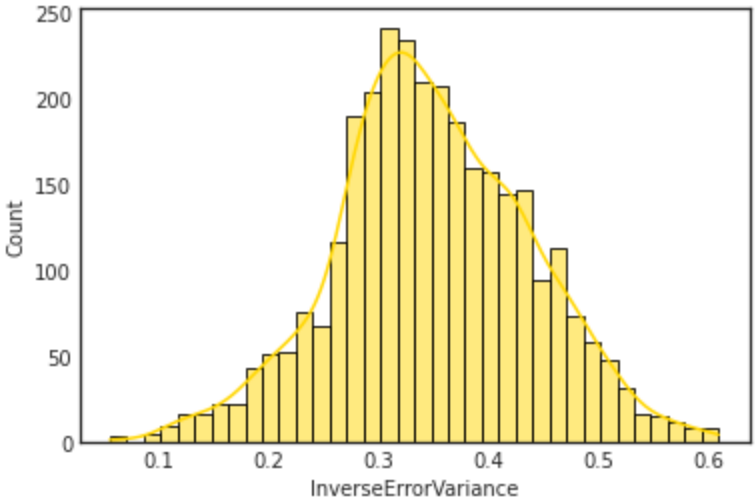
$$NIEV = \frac{IEV}{\sum_{i=1} -nIEV(i)}$$

where *SSR* is the sum of squared residuals; n is the number of samples; and T is an arbitrary coefficient. Low T values lead to equal weights placed on each sample set while higher T values concentrate on the best parameter sets.

```
# From Vrugt et al (2008) : inverse error variance

# T=0 means that we only select the best simulated values that are closer to the␣
↪observed values
#  T=infinity means that all sample sets have the same probability
T= 0.70

# calculate metrics from Beven and Binley, 1992
df_metrics_SA["SSR"] = df_metrics_SA["RMSE"]**2 * 3650
df_metrics_SA["InverseErrorVariance"] = (df_metrics_SA["SSR"] / df_metrics_SA.
↪shape[0])**(-T)
df_metrics_SA["Normalized_IEV"] = df_metrics_SA["InverseErrorVariance"] / df_metrics_SA[
↪"InverseErrorVariance"].sum()

# convert array to dataframe
param_values_hymod_df = pd.DataFrame(param_values_hymod, columns=['Kq', 'Ks', 'Alp', 'Huz
↪', 'B'])

# combine the metrics and param values dataframes and calculate combined metrics
concat_df = pd.concat([df_metrics_SA, param_values_hymod_df], axis=1)
concat_df["Ks_rescale"] = concat_df["Ks"] / 0.1
concat_df["Huz_rescale"] = concat_df["Huz"] / 500
concat_df["B_rescale"] = concat_df["B"] / 2

# display the first 5 rows of the dataframe
df_metrics_SA.head()
```

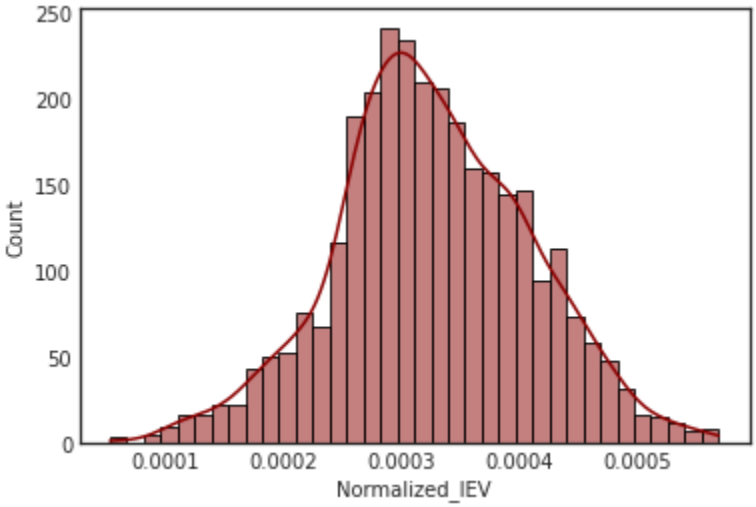
Distribution of likelihoods (inverse error variance) values

```
# density plot and histogram of the inverse error variance
h = sns.histplot(data=df_metrics_SA,
                 x="InverseErrorVariance",
                 kde=True,
                 bins=int(180/5),
                 color = 'gold')
```



Distribution of normalized inverse error variance values

```
# density plot and histogram of the normalized inverse error variance
h = sns.histplot(data=df_metrics_SA,
                 x="Normalized_IEV",
                 kde=True,
                 bins=int(180/5),
                 color='darkred')
```



Selection of important sample sets and setting a threshold for physical/non-physical sample sets

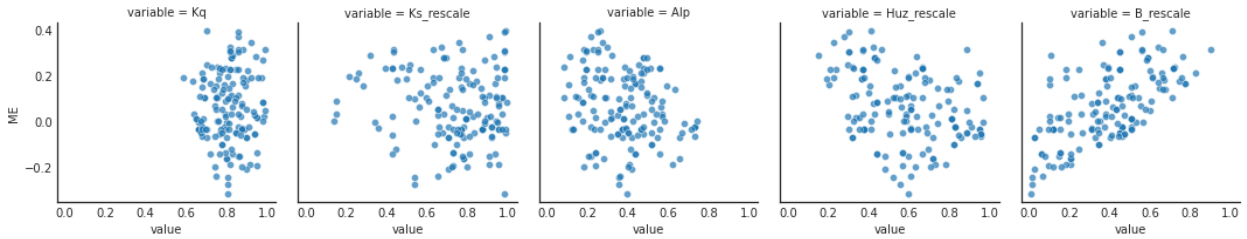
```
# selection of important sample sets
percentile = 95

threshold = np.percentile(concat_df["InverseErrorVariance"], percentile)
print(f"Threshold using the {percentile} percentile:  {threshold}")

# select values greater than the threshold
selected_values_glue = concat_df[concat_df["InverseErrorVariance"] > threshold]
```

```
Threshold using the 95 percentile:  0.4981408917485908
```

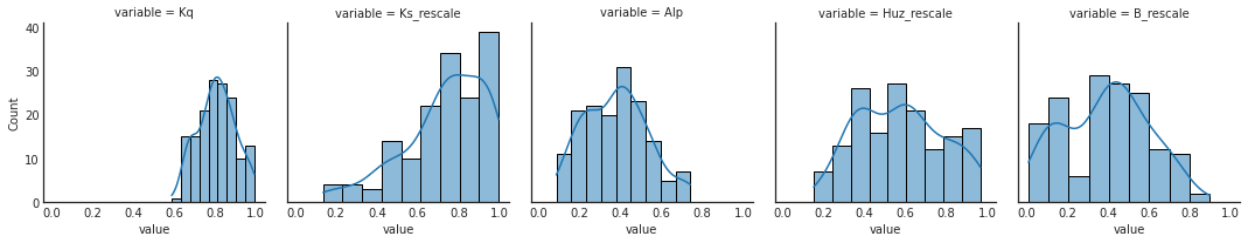
8.4.8.2 5-2 Visual inspection of GLUE results

```
# format the data frame so that it may be used for plotting
to_plot = pd.melt(selected_values_glue,
                  id_vars=['ME'],
                  value_vars=['Kq', 'Ks_rescale', 'Alp', 'Huz_rescale', 'B_rescale'])
```

```
# build a plot with multiple panels of scatter plots where ME is the target metric
g = sns.FacetGrid(to_plot, col="variable")

# map the scatter plots to the facet grid panels
gf = g.map(sns.scatterplot, "value", "ME",  alpha=0.7)
```




```
# build a plot with multiple panels of histogram plots where ME is the target metric
g = sns.FacetGrid(to_plot, col="variable")

# map the plots to the facet grid panels
gf = g.map(sns.histplot, "value", kde=True)
```



8.4.8.3 5-3 Comment on the GLUE results

Our results suggest that it is challenging to find an clear and interpretable relationship between different selected near-optimal sample sets at least by visual inspection. The main reason for this is that HYMOD includes a complex non-linear system of equations that is also affected by initial conditions and complexity of its input time series. Therefore, it does not have a clear control.

Glue has been widely used in hydrology, the original paper has more than 5000 citations. However, the likelihood measure that GLUE uses is not actually a statistically sound likelihood metric and is in fact a goodness-of-fit measure. Therefore, it might not produce valid insights when dealing with situations of non-normality, heteroscedasticity, and serial correlation. For more on these issues reader can refer to Stedinger et al., (2008), Mantovan and Todini, (2006), and Beven And Binley, (2014).

8.4.9 6- Pre-Calibration

Pre-calibration (Edwards et al, 2010) is a simplified method to deal with uncertainty in complex environmental models. Pre-calibration can also be thought of as another method that tackles the shortcomings and conceptual challenges involved in calibration of complex environmental models. In pre-calibration instead of finding the best solutions, we focus on finding the sample sets that create outputs that are against the common understanding of the system. These parameter sets are called non-physical parameter sets. In other words, the probability that these parameters are among the best sample sets is zero or near zero and can be neglected in practice.

Pre-calibration can include the following steps: 1) Sensitivity analysis 2) Definition of non-physical boundaries 3) Delineating regions in the output space which are non-physical (Implausible) 4) Map non-physical sets back to input space 5) Interpret the non-physical sample sets

8.4.9.1 6-1 Pre-calibration calculations

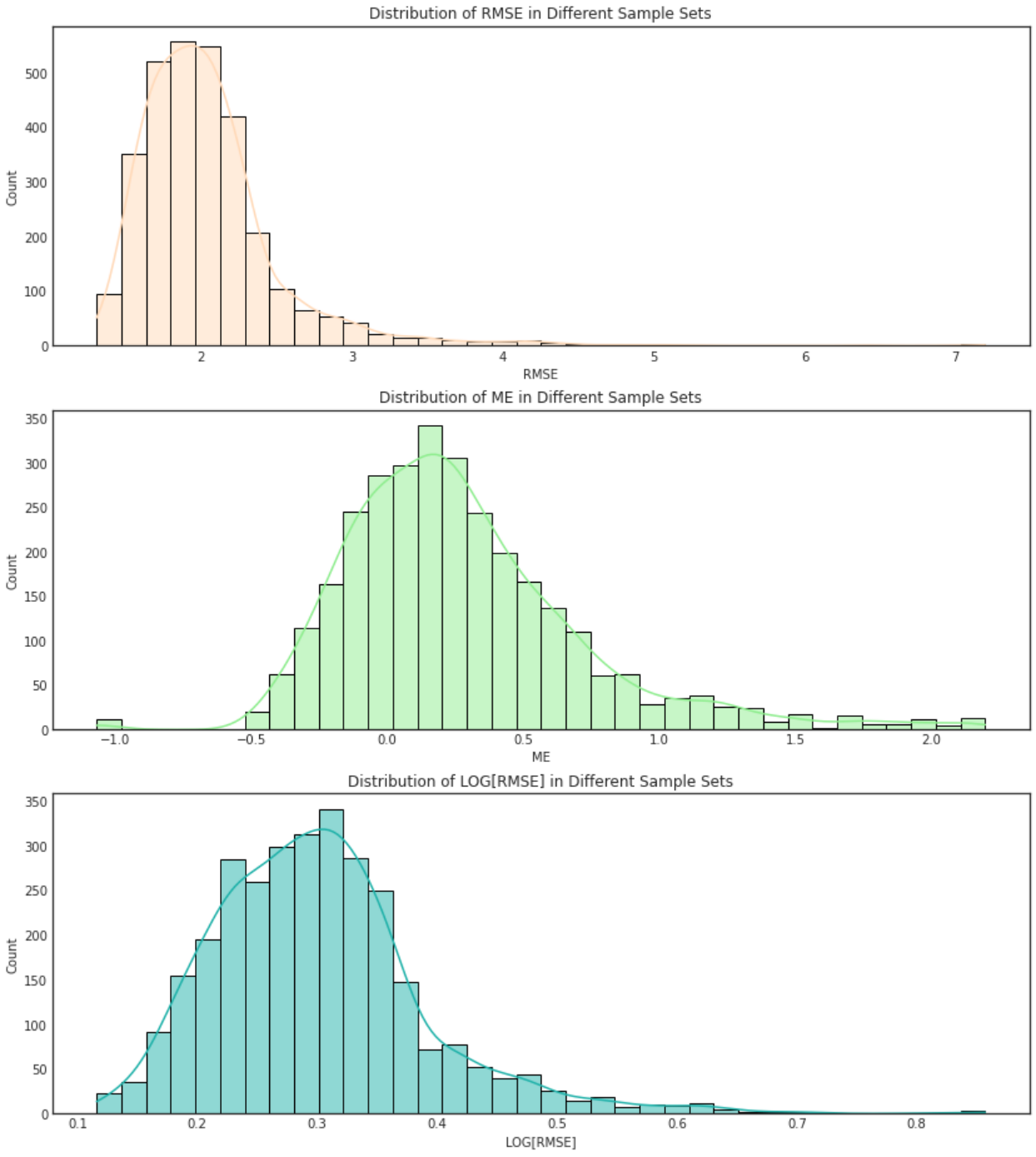
Distribution of mean error, RMSE, and Log[RMSE] under different sample sets

```
# set up figure and axis objects
fig, axs = plt.subplots(nrows=3, figsize=(14,16))

# axis 1 (first row in figure)
a = sns.histplot(data=concat_df,
                 x="RMSE",
                 ax=axs[0],
                 kde=True,
                 bins=int(180/5),
                 color='peachpuff').set_title('Distribution of RMSE in Different Sample␣
↪Sets')

# axis 2 (second row in figure)
b = sns.histplot(data=concat_df,
                 x="ME",
                 ax=axs[1],
                 kde=True,
                 bins=int(180/5),
                 color='lightgreen').set_title('Distribution of ME in Different Sample␣
↪Sets')

# axis 3 (third row in figure)
```

(continues on next page)

(continued from previous page)

```
c = sns.histplot(data=concat_df,
                 x="LOG[RMSE]",
                 ax=axs[2],
                 kde=True,
                 bins=int(180/5),
                 color = 'lightseagreen').set_title('Distribution of LOG[RMSE] in␣
↪Different Sample Sets')
```



Setting a threshold for physical/non-physical sample sets

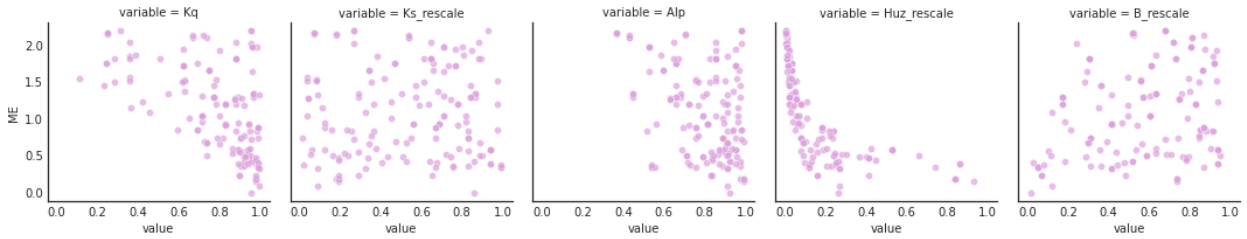
```
# selection of physical/non-physical sample sets
percentile = 95

threshold_precal = np.percentile(concat_df["RMSE"], percentile)
print(f"Threshold using the {percentile} percentile:  {threshold}")

# select values greater than the threshold
selected_values_precal = concat_df[concat_df["RMSE"] > threshold_precal]
```

```
Threshold using the 95 percentile:  0.4981408917485908
```

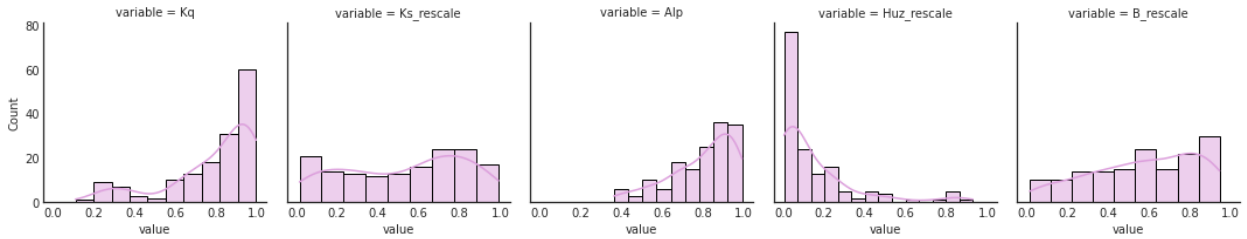
Visual inspection of non-physical sample sets

```
# format the data frame so that it may be used for plotting
to_plot_precal = pd.melt(selected_values_precal,
                         id_vars=['ME'],
                         value_vars=['Kq', 'Ks_rescale', 'Alp', 'Huz_rescale', 'B_rescale
↪'])
```

```
# build a plot with multiple panels of scatter plots where ME is the target metric
g = sns.FacetGrid(to_plot_precal, col="variable")

# map the scatter plots to the facet grid panels
gh = g.map(sns.scatterplot, "value", "ME",  alpha=0.7, color='plum')
```



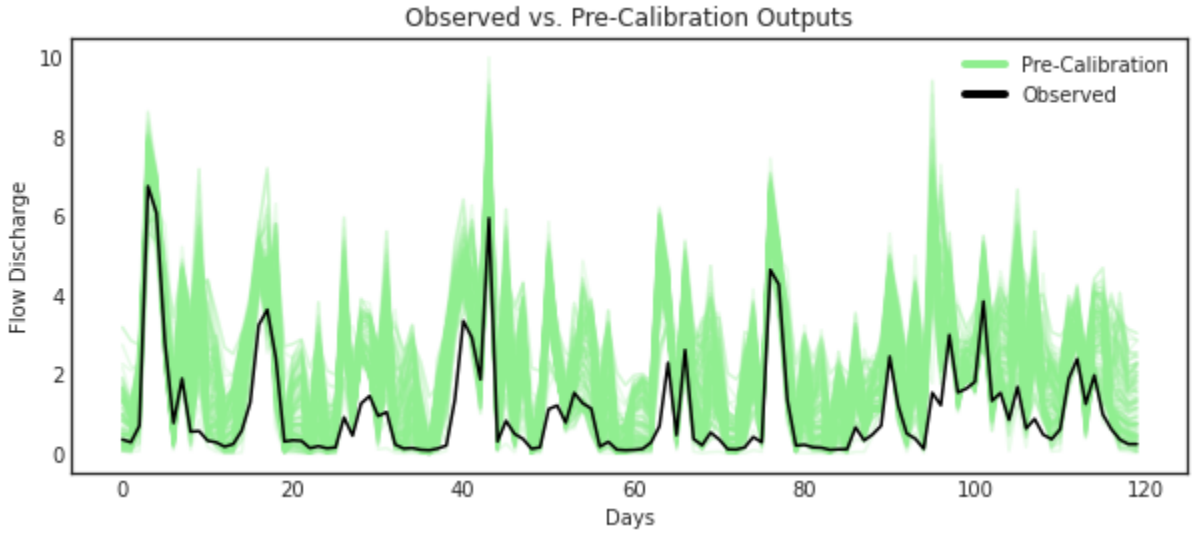

```
# build a plot with multiple panels of histogram plots where ME is the target metric
g = sns.FacetGrid(to_plot_precal, col="variable")

# map the plots to the facet grid panels
gh = g.map(sns.histplot, "value", kde=True, color='plum')
```



The following figure shows the flow discharge provided by the ensemble of parameters sets from Pre-Calibration versus the observed flow data.

Note: **Tip:** View the source code used to create this plot here: plot_precalibration_flow

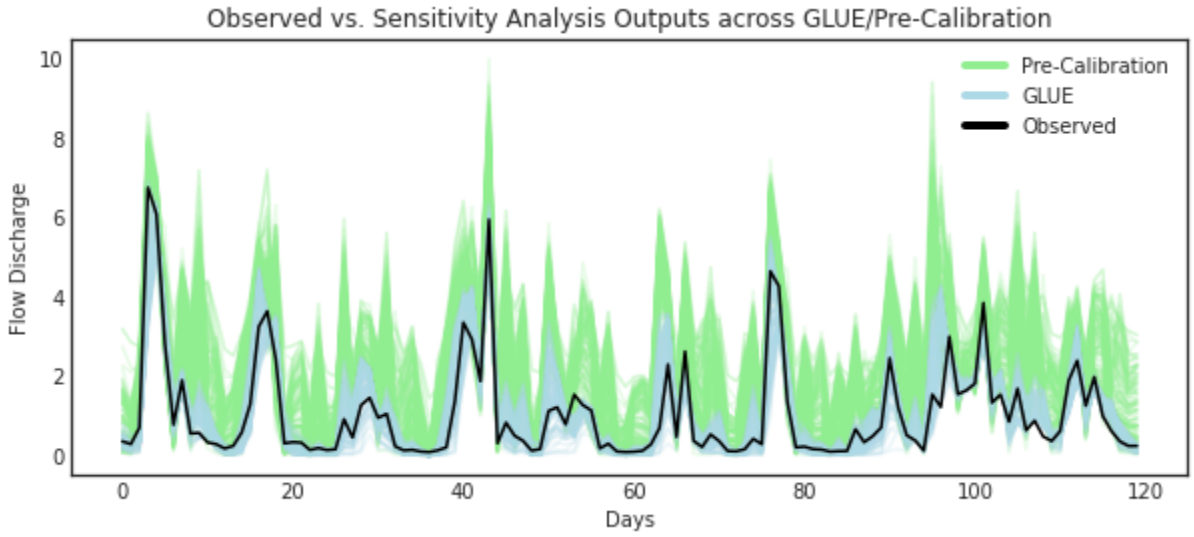
```
# mean monthly indices
Q_df_precal = pd.concat([Q_df.iloc[:, selected_values_precal.index],
                         Q_df[['month' , 'year']]],
                        axis=1)

# calculate year, month mean
df_precal_mth_mean = Q_df_precal.groupby(['year', 'month']).mean()

# plot observed versus pre-calibration outputs
ax = msdbook.plot_precalibration_flow(df_sim=df_precal_mth_mean,
                                      df_obs=df_obs_mth_mean)
```



The following figure shows the flow discharge provided by the ensemble of parameters sets from both pre-calibration and GLUE and how these flows compare to the observed flow data.

Note: **Tip:** View the source code used to create this plot here: plot_precalibration_glue

```
Q_df_glue = pd.concat([Q_df.iloc[:, selected_values_glue.index],
                       Q_df.loc[:,['month' , 'year']]],
                      axis=1)

# calculate year, month mean
df_glue_mth_mean = Q_df_glue.groupby(['year', 'month']).mean()


# plot observed versus pre-calibration and GLUE outputs
ax = msdbook.plot_precalibration_glue(df_precal=df_precal_mth_mean,
                                      df_glue=df_glue_mth_mean,
                                      df_obs=df_obs_mth_mean)
```



8.4.9.2 6-2 Comments on Pre-Calibration

Although Pre-calibrations provides a helpful and relatively simple alternative for statistical inferences, it has some disadvantages. For example, it is often subjective and challenging to provide a threshold for non-physical model outputs. Moreover, as discussed earlier different goodness-of-fit metrics can produce distinct physical and non-physical sample sets.

More information about pre-calibration can be found in (Edwards et al., 2010), and Ruckert et al., (2017).

CHAPTER

NINE

PLOTTING CODE SAMPLES

9.1 hymod.ipynb

The following are the plotting functions as described in the `hymod.ipynb` Jupyter notebook tutorial.

The following are the necessary package imports to run these functions:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from matplotlib.lines import Line2D
```

9.1.1 plot_observed_vs_simulated_streamflow()

```
def plot_observed_vs_simulated_streamflow(df, hymod_dict, figsize=[12, 6]):
    """Plot observed versus simulated streamflow.

    :param df:              Dataframe of hymod input data including columns for precip,␣
↪potential evapotranspiration,
                            and streamflow

    :param hymod_dict:      A dictionary of hymod outputs
    :type hymod_dict:       dict

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set plot style
    plt.style.use('seaborn-white')

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # plot observed streamflow
    ax.plot(range(0, len(df['Strmflw'])), df['Strmflw'], color='pink')
```

(continues on next page)

(continued from previous page)

```
    # plot simulated streamflow
    ax.plot(range(0, len(df['Strmflw'])), hymod_dict['Q'], color='black')

    # set axis labels
    ax.set_ylabel('Streamflow($m^3/s$)')
    ax.set_xlabel('Days')

    # set plot title
    plt.title('Observed vs. Simulated Streamflow')

    return ax
```

9.1.2 plot_observed_vs_sensitivity_streamflow()

```
def plot_observed_vs_sensitivity_streamflow(df_obs, df_sim, figsize=[10, 4]):
    """Plot observed streamflow versus simulations generated from sensitivity analysis.

    :param df_obs:          Dataframe of mean monthly hymod input data including columns␣
↪for precip,
                            potential evapotranspiration, and streamflow

    :param df_sim:          Dataframe of mean monthly simulation data from sensitivity␣
↪analysis

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    month_list = range(len(df_sim))

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # set labels
    ax.set_xlabel('Days')
    ax.set_ylabel('Flow Discharge (m^3/s)')

    # plots all simulated streamflow cases under different sample sets
    for i in df_sim.columns:
        plt.plot(month_list, df_sim[i], color="pink", alpha=0.2)

    # plot observed streamflow
    plt.plot(month_list, df_obs['Strmflw'], color="black")

    plt.title('Observed vs. Sensitivity Analysis Outputs')

    return ax
```

9.1.3 plot_monthly_heatmap()

```
def plot_monthly_heatmap(arr_sim, df_obs, title='', figsize=[14, 6]):
    """Plot a sensitivity metric overlain by observed flow.

    :param arr_sim:         Numpy array of simulated metrics

    :param df_obs:          Dataframe of mean monthly observed data from sensitivity␣
↪analysis

    :param title:           Title of plot
    :type title:            str

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # plot heatmap
    sns.heatmap(arr_sim,
                ax=ax,
                yticklabels=['Kq', 'Ks', 'Alp', 'Huz', 'B'],
                cmap=sns.color_palette("ch:s=-.2,r=.6"))

    # setup overlay axis
    ax2 = ax.twinx()

    # plot line
    ax2.plot(np.arange(0.5, 12.5), df_obs['Strmflw'], color='slateblue')

    # plot points on line
    ax2.plot(np.arange(0.5, 12.5), df_obs['Strmflw'], color='slateblue', marker='o')

    # set axis limits and labels
    ax.set_ylim(0, 5)
    ax.set_xlim(0, 12)
    ax.set_xticklabels(['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep',
↪'oct', 'nov', 'dec'])
    ax2.set_ylabel('Flow Discharge($m^3/s$)')

    plt.title(title)

    plt.show()

    return ax, ax2
```

9.1.4 plot_annual_heatmap()

```
def plot_annual_heatmap(arr_sim, df_obs, title='', figsize=[14,5]):
    """Plot a sensitivity metric overlain by observed flow..

    :param arr_sim:         Numpy array of simulated metrics

    :param df_obs:          Dataframe of mean monthly observed data from sensitivity␣
↪analysis

    :param title:           Title of plot
    :type title:            str

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # plot heatmap
    sns.heatmap(arr_sim, ax=ax, cmap=sns.color_palette("YlOrBr"))

    # setup overlay axis
    ax2 = ax.twinx()

    # plot line
    ax2.plot(np.arange(0.5, 10.5), df_obs['Strmflw'], color='slateblue')

    # plot points on line
    ax2.plot(np.arange(0.5, 10.5), df_obs['Strmflw'], color='slateblue', marker='o')

    # set up axis lables and limits
    ax.set_ylim(0, 5)
    ax.set_xlim(0, 10)
    ax.set_yticklabels(['Kq', 'Ks', 'Alp', 'Huz', 'B'])
    ax.set_xticklabels(range(2000, 2010))
    ax2.set_ylabel('Flow Discharge($m^3/s$)')

    plt.title(title)

    return ax, ax2
```

9.1.5 plot_varying_heatmap()

```
def plot_varying_heatmap(arr_sim, df_obs, title='', figsize=[14,5]):
    """Plot a sensitivity metric overlain by observed flow..

    :param arr_sim:         Numpy array of simulated metrics

    :param df_obs:          Dataframe of mean monthly observed data from sensitivity␣
↪analysis

    :param title:           Title of plot
    :type title:            str

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # plot heatmap
    sns.heatmap(arr_sim,
                ax=ax,
                yticklabels=['Kq', 'Ks', 'Alp', 'Huz', 'B'],
                cmap=sns.light_palette("seagreen", as_cmap=True))

    n_years = df_obs.shape[0]

    # setup overlay axis
    ax2 = ax.twinx()

    # plot line
    ax2.plot(range(0, n_years), df_obs['Strmflw'], color='slateblue')

    # plot points on line
    ax2.plot(range(0, n_years), df_obs['Strmflw'], color='slateblue', marker='o')

    # set up axis lables and limits
    ax.set_ylim(0, 5)
    ax.set_xlim(-0.5, 119.5)
    ax2.set_ylabel('Flow Discharge')
    ax.set_xlabel('Number of Months')

    plt.title(title)

    return ax, ax2
```

9.1.6 plot_precalibration_flow()

```
def plot_precalibration_flow(df_sim, df_obs, figsize=[10, 4]):
    """Plot flow discharge provided by the ensemble of parameters sets from Pre-
↪Calibration versus the observed
    flow data.

    :param df_sim:          Dataframe of simulated metrics

    :param df_obs:          Dataframe of mean monthly observed data from sensitivity␣
↪analysis

    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # set axis labels
    ax.set_xlabel('Days')
    ax.set_ylabel('Flow Discharge')

    # plot pre-calibration results
    for i in range(df_sim.shape[1]):
        plt.plot(range(len(df_sim)), df_sim.iloc[:, i],  color="lightgreen", alpha=0.2)

    # plot observed
    plt.plot(range(len(df_sim)), df_obs['Strmflw'],  color="black")

    plt.title('Observed vs. Pre-Calibration Outputs')

    # customize legend
    custom_lines = [Line2D([0], [0],  color="lightgreen", lw=4),
                    Line2D([0], [0], color="black", lw=4)]
    plt.legend(custom_lines, ['Pre-Calibration', 'Observed'])

    return ax
```

9.1.7 plot_precalibration_glue()

```
def plot_precalibration_glue(df_precal, df_glue, df_obs, figsize=[10, 4]):
    """Plot flow discharge provided by the ensemble of parameters sets from Pre-
↪Calibration versus the observed
    flow data.

    :param df_sim:          Dataframe of simulated metrics

    :param df_obs:          Dataframe of mean monthly observed data from sensitivity␣
↪analysis
```

(continues on next page)

(continued from previous page)

```
    :param figsize:         Matplotlib figure size
    :type figsize:          list

    """

    # set up figure
    fig, ax = plt.subplots(figsize=figsize)

    # set axis labels
    ax.set_xlabel('Days')
    ax.set_ylabel('Flow Discharge')

    # plot pre-calibration results
    for i in range(df_precal.shape[1]):
        plt.plot(range(len(df_precal)), df_precal.iloc[:, i],  color="lightgreen",␣
↪alpha=0.2)

    # plot glue
    for i in range(df_glue.shape[1]):
        plt.plot(range(len(df_glue)), df_glue.iloc[:, i], color="lightblue", alpha=0.2)

    # plot observed
    plt.plot(range(len(df_precal)), df_obs['Strmflw'],  color="black")

    plt.title('Observed vs. Sensitivity Analysis Outputs across GLUE/Pre-Calibration')

    # customize legend
    custom_lines = [Line2D([0], [0],  color="lightgreen", lw=4),
                    Line2D([0], [0], color="lightblue", lw=4),
                    Line2D([0], [0], color="black", lw=4)]
    plt.legend(custom_lines, ['Pre-Calibration', 'GLUE', 'Observed'])

    return ax
```

9.2 fishery_dynamics.ipynb

The following are the plotting functions as described in the `fishery_dynamics.ipynb` Jupyter notebook tutorial.

The following are the necessary package imports to run these functions:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import patheffects as pe
```

9.2.1 plot_objective_performance()

```
def plot_objective_performance(objective_performance, profit_solution, robust_solution,␣
↪figsize=(18, 9)):
    """Plot the identified solutions with regards to their objective performance
    in a parallel axis plot

    :param objective_performance:               Objective performance array
    :param profit_solution:                     Profitable solutions array
    :param robust_solution:                     Robust solutions array
    :param figsize:                             Figure size
    :type figsize:                              tuple

    """

    # create the figure object
    fig = plt.figure(figsize=figsize)

    # set up subplot axis object
    ax = fig.add_subplot(1, 1, 1)

    # labels where constraint is always 0
    objs_labels = ['Net present\nvalue (NPV)',
                   'Prey population deficit',
                   'Longest duration\nof low harvest',
                   'Worst harvest instance',
                   'Variance of harvest',
                   'Duration of predator\npopulation collapse']

    # normalization across objectives
    mins = objective_performance.min(axis=0)
    maxs = objective_performance.max(axis=0)
    norm_reference = objective_performance.copy()

    for i in range(5):
        mm = objective_performance[:, i].min()
        mx = objective_performance[:, i].max()
        if mm != mx:
            norm_reference[:, i] = (objective_performance[:, i] - mm) / (mx - mm)
        else:
            norm_reference[:, i] = 1

    # colormap from matplotlib
    cmap = plt.cm.get_cmap("Blues")

    # plot all solutions
    for i in range(len(norm_reference[:, 0])):
        ys = np.append(norm_reference[i, :], 1.0)
        xs = range(len(ys))
        ax.plot(xs, ys, c=cmap(ys[0]), linewidth=2)

    # to highlight robust solutions
    ys = np.append(norm_reference[profit_solution, :], 1.0)  # Most profitable
```

(continues on next page)

(continued from previous page)

```
    xs = range(len(ys))
    l1 = ax.plot(xs[0:6],
                 ys[0:6],
                 c=cmap(ys[0]),
                 linewidth=3,
                 label='Most robust in NPV',
                 path_effects=[pe.Stroke(linewidth=6, foreground='darkgoldenrod'), pe.
↪Normal()])

    ys = np.append(norm_reference[robust_solution, :], 1.0)  # Most robust in all␣
↪criteria
    xs = range(len(ys))
    l2 = ax.plot(xs[0:6],
                 ys[0:6],
                 c=cmap(ys[0]),
                 linewidth=3,
                 label='Most robust across criteria',
                 path_effects=[pe.Stroke(linewidth=6, foreground='gold'), pe.Normal()])

    # build colorbar
    sm = plt.cm.ScalarMappable(cmap=cmap)
    sm.set_array([objective_performance[:, 0].min(), objective_performance[:, 0].max()])
    cbar = fig.colorbar(sm)
    cbar.ax.set_ylabel("\nNet present value (NPV)")

    # tick values
    minvalues = ["{0:.3f}".format(mins[0]),
                 "{0:.3f}".format(-mins[1]),
                 str(-mins[2]),
                 "{0:.3f}".format(-mins[3]),
                 "{0:.2f}".format(-mins[4]),
                 str(0)]

    maxvalues = ["{0:.2f}".format(maxs[0]),
                 "{0:.3f}".format(-maxs[1]),
                 str(-maxs[2]),
                 "{0:.2f}".format(maxs[3]),
                 "{0:.2f}".format(-maxs[4]),
                 str(0)]

    ax.set_ylabel("Preference ->", size=12)
    ax.set_yticks([])
    ax.set_xticks([0, 1, 2, 3, 4, 5])
    ax.set_xticklabels([minvalues[i] + '\n' + objs_labels[i] for i in range(len(objs_
↪labels))])

    # make a twin axis for toplabels
    ax1 = ax.twiny()
    ax1.set_yticks([])
    ax1.set_xticks([0, 1, 2, 3, 4, 5])
    ax1.set_xticklabels([maxvalues[i] for i in range(len(maxs) + 1)])
```

(continues on next page)

(continued from previous page)

```
    return ax, ax1
```

9.2.2 plot_factor_performance()

```
def plot_factor_performance(param_values, collapse_days, b, m, a):
    """Visualize the performance of our policies in three-dimensional
    parametric space.

    :param param_values:                Saltelli sample array
    :param collapse_days:               Simulation array
    :param b:                           b parameter boundary interval
    :param m:                           m parameter boundary interval
    :param a:                           a parameter boundary interval

    """

    # set colormap
    cmap = plt.cm.get_cmap("RdBu_r")

    # build figure object
    fig = plt.figure(figsize=plt.figaspect(0.5), dpi=600, constrained_layout=True)

    # set up scalable colormap
    sm = plt.cm.ScalarMappable(cmap=cmap)

    # set up subplot for profit maximizing policy
    ax1 = fig.add_subplot(1, 2, 1, projection='3d')

    # add point data for profit plot
    sows = ax1.scatter(param_values[:,1],
                       param_values[:,6],
                       param_values[:,0],
                       c=collapse_days[:,0],
                       cmap=cmap,
                       s=0.5)

    # add surface data for boundary separating successful and failed states of the world
    pts_ineq = ax1.plot_surface(b, m, a, color='black', alpha=0.25, zorder=1)

    # add reference point to plot
    pt_ref = ax1.scatter(0.5, 0.7, 0.005, c='black', s=50, zorder=0)

    # set up plot aesthetics and labels
    ax1.set_xlabel("b")
    ax1.set_ylabel("m")
    ax1.set_zlabel("a")
    ax1.set_zlim([0.0, 2.0])
    ax1.set_xlim([0.0, 1.0])
    ax1.set_ylim([0.0, 1.5])
    ax1.xaxis.set_view_interval(0,  0.5)
```

(continues on next page)

(continued from previous page)

```
    ax1.set_facecolor('white')
    ax1.view_init(12, -17)
    ax1.set_title('Profit maximizing policy')

    # set up subplot for robust policy
    ax2 = fig.add_subplot(1, 2, 2, projection='3d')

    # add point data for robust plot
    sows = ax2.scatter(param_values[:,1],
                       param_values[:,6],
                       param_values[:,0],
                       c=collapse_days[:,1],
                       cmap=cmap,
                       s=0.5)

    # add surface data for boundary separating successful and failed states of the world
    pts_ineq = ax2.plot_surface(b, m, a, color='black', alpha=0.25, zorder=1)

    # add reference point to plot
    pt_ref = ax2.scatter(0.5, 0.7, 0.005, c='black', s=50, zorder=0)

    # set up plot aesthetics and labels
    ax2.set_xlabel("b")
    ax2.set_ylabel("m")
    ax2.set_zlabel("a")
    ax2.set_zlim([0.0, 2.0])
    ax2.set_xlim([0.0, 1.0])
    ax2.set_ylim([0.0, 1.5])
    ax2.xaxis.set_view_interval(0, 0.5)
    ax2.set_facecolor('white')
    ax2.view_init(12, -17)
    ax2.set_title('Robust policy')

    # set up colorbar
    sm.set_array([collapse_days.min(), collapse_days.max()])
    cbar = fig.colorbar(sm)
    cbar.set_label('Days with predator collapse')

    return ax1, ax2
```

BIBLIOGRAPHY

[1] National Research Council and others. *Convergence: Facilitating transdisciplinary integration of life sciences, physical sciences, engineering, and beyond*. National Academies Press, 2014.

[2] Sondoss Elsawah, Tatiana Filatova, Anthony J Jakeman, Albert J Kettner, Moira L Zellner, Ioannis N Athanasiadis, Serena H Hamilton, Robert L Axtell, Daniel G Brown, Jonathan M Gilligan, and others. Eight grand challenges in socio-environmental systems modeling. *Socio-Environmental Systems Modelling*, 2:16226–16226, 2020.

[3] Yacov Y Haimes. Risk modeling of interdependent complex systems of systems: theory and practice. *Risk analysis*, 38(1):84–98, 2018.

[4] Dirk Helbing. Globally networked risks and how to respond. *Nature*, 497(7447):51–59, 2013.

[5] Andrea Saltelli, Ksenia Aleksankina, William Becker, Pamela Fennell, Federico Ferretti, Niels Holst, Sushan Li, and Qiongli Wu. Why so many published sensitivity analyses are false: a systematic review of sensitivity analysis practices. *Environmental modelling & software*, 114:29–39, 2019.

[6] Daniel Wirtz and Wolfgang Nowak. The rocky road to extended simulation frameworks covering uncertainty, inversion, optimization and control. *Environmental Modelling & Software*, 93:180–192, 2017.

[7] Roger Cooke and others. *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press on Demand, 1991.

[8] Enayat A Moallemi, Jan Kwakkel, Fjalar J de Haan, and Brett A Bryan. Exploratory modeling for analyzing coupled human-natural systems under uncertainty. *Global Environmental Change*, 65:102186, 2020.

[9] Warren E Walker, Poul Harremoës, Jan Rotmans, Jeroen P Van Der Sluijs, Marjolein BA Van Asselt, Peter Janssen, and Martin P Krayer von Krauss. Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. *Integrated assessment*, 4(1):5–17, 2003.

[10] Jan H Kwakkel, Warren E Walker, and Marjolijn Haasnoot. Coping with the wickedness of public policy problems: approaches for decision making under deep uncertainty. 2016.

[11] Saul I Gass and Carl M Harris. Encyclopedia of operations research and management science. *Journal of the Operational Research Society*, 48(7):759–760, 1997.

[12] Andrea Saltelli, Philip B Stark, William Becker, and Pawel Stano. Climate models as economic guides scientific challenge or quixotic quest? *Issues in Science and Technology*, 31(3):79–84, 2015.

[13] Hoshin V. Gupta, Thorsten Wagener, and Yuqiong Liu. Reconciling theory with observations: elements of a diagnostic approach to model evaluation. *Hydrological Processes: An International Journal*, 22(18):3802–3813, 2008. Publisher: Wiley Online Library.

[14] Antonia Hadjimichael, Julianne Quinn, and Patrick Reed. Advancing Diagnostic Model Evaluation to Better Understand Water Shortage Mechanisms in Institutionally Complex River Basins. *Water Re-*

sources Research, 56(10):e2020WR028079, 2020. URL: http://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR028079 (visited on 2020-10-16), doi:10.1029/2020WR028079.

[15] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis: The Primer*. Wiley-Interscience, Chichester, England ; Hoboken, NJ, 1 edition edition, February 2008. ISBN 978-0-470-05997-5.

[16] Keith Beven. Towards a coherent philosophy for modelling the environment. *Proceedings of the royal society of London. Series A: mathematical, physical and engineering sciences*, 458(2026):2465–2484, 2002.

[17] Naomi Oreskes, Kristin Shrader-Frechette, and Kenneth Belitz. Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences. *Science*, 263(5147):641–646, February 1994. URL: https://science.sciencemag.org/content/263/5147/641 (visited on 2020-04-15), doi:10.1126/science.263.5147.641.

[18] Keith Beven. Prophecy, reality and uncertainty in distributed hydrological modelling. *Advances in water resources*, 16(1):41–51, 1993.

[19] Keith Beven and Andrew Binley. The future of distributed models: Model calibration and uncertainty prediction. *Hydrological Processes*, 6(3):279–298, 1992. doi:10.1002/hyp.3360060305.

[20] Yaman Barlas and Stanley Carpenter. Philosophical roots of model validation: Two paradigms. *System Dynamics Review*, 6(2):148–166, 1990. doi:10.1002/sdr.4260060203.

[21] Stephen Toulmin. From form to function: philosophy and history of science in the 1950s and now. *Daedalus*, pages 143–162, 1977. Publisher: JSTOR.

[22] George B. Kleindorfer, Liam O'Neill, and Ram Ganeshan. Validation in simulation: Various positions in the philosophy of science. *Management Science*, 44(8):1087–1099, 1998. Publisher: INFORMS.

[23] Sibel Eker, Elena Rovenskaya, Michael Obersteiner, and Simon Langan. Practice and perspectives in the validation of resource management models. *Nature communications*, 9(1):1–10, 2018.

[24] Yaman Barlas. Formal aspects of model validity and validation in system dynamics. *System Dynamics Review: The Journal of the System Dynamics Society*, 12(3):183–210, 1996. Publisher: Wiley Online Library.

[25] Thomas H. Naylor and Joseph Michael Finger. Verification of computer simulation models. *Management science*, 14(2):B–92, 1967. Publisher: INFORMS.

[26] Keith J Beven. On hypothesis testing in hydrology: why falsification of models is still a really good idea. *Wiley Interdisciplinary Reviews: Water*, 5(3):e1278, 2018.

[27] Hoshin V Gupta, Martyn P Clark, Jasper A Vrugt, Gab Abramowitz, and Ming Ye. Towards a comprehensive assessment of model structural adequacy. *Water Resources Research*, 2012.

[28] Praveen Kumar. Typology of hydrologic predictability. *Water Resources Research*, 2011. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2010WR009769 (visited on 2020-04-15), doi:10.1029/2010WR009769.

[29] Grey S. Nearing, Benjamin L. Ruddell, Andrew R. Bennett, Cristina Prieto, and Hoshin V. Gupta. Does Information Theory Provide a New Paradigm for Earth Science? Hypothesis Testing. *Water Resources Research*, 56(2):e2019WR024918, 2020. doi:10.1029/2019WR024918.

[30] Hoshin Vijai Gupta, Soroosh Sorooshian, and Patrice Ogou Yapo. Toward improved calibration of hydrologic models: Multiple and noncommensurable measures of information. *Water Resources Research*, 34(4):751–763, 1998. URL: http://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/97WR03495 (visited on 2020-04-07), doi:10.1029/97WR03495.

[31] Francesca Pianosi and Thorsten Wagener. Understanding the time-varying importance of different uncertainty sources in hydrological modelling using global sensitivity analysis. *Hydrological Processes*, pages 3991–4003, November 2017. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/hyp.10968%4010.1111/%28ISSN%291099-1085.Kieth-Beven, doi:10.1002/hyp.10968@10.1111/(ISSN)1099-1085.Kieth-Beven.

[32] Charles Rougé, Patrick M. Reed, Danielle S. Grogan, Shan Zuidema, Alexander Prusevich, Stanley Glidden, Jonathan R. Lamontagne, and Richard B. Lammers. Coordination and Control: Limits in Standard Representations of Multi-Reservoir Operations in Hydrological Modeling. *Hydrology and Earth System Sciences Discussions*, pages 1–37, November 2019. URL: https://www.hydrol-earth-syst-sci-discuss.net/hess-2019-589/, doi:https://doi.org/10.5194/hess-2019-589.

[33] David W. Cash, William C. Clark, Frank Alcock, Nancy M. Dickson, Noelle Eckley, David H. Guston, Jill Jäger, and Ronald B. Mitchell. Knowledge systems for sustainable development. *Proceedings of the national academy of sciences*, 100(14):8086–8091, 2003. Publisher: National Acad Sciences.

[34] Dave D. White, Amber Wutich, Kelli L. Larson, Patricia Gober, Timothy Lant, and Clea Senneville. Credibility, salience, and legitimacy of boundary objects: water managers' assessment of a simulation model in an immersive decision theater. *Science and Public Policy*, 37(3):219–232, April 2010. Publisher: Oxford Academic. URL: https://academic.oup.com/spp/article/37/3/219/1626552 (visited on 2020-05-12), doi:10.3152/030234210X497726.

[35] Andrea Saltelli and Silvio Funtowicz. When all models are wrong. *Issues in Science and Technology*, 30(2):79–85, 2014. Publisher: JSTOR.

[36] Thorsten Wagener and Francesca Pianosi. What has Global Sensitivity Analysis ever done for us? A systematic review to support scientific advancement and to inform policy-making in earth system modelling. *Earth-Science Reviews*, 194:1–18, July 2019. URL: https://www.sciencedirect.com/science/article/pii/S0012825218300990 (visited on 2021-08-30), doi:10.1016/j.earscirev.2019.04.006.

[37] Steve Bankes. Exploratory Modeling for Policy Analysis. *Operations Research*, 41(3):435–449, June 1993. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.41.3.435 (visited on 2018-09-11), doi:10.1287/opre.41.3.435.

[38] Christopher P. Weaver, Robert J. Lempert, Casey Brown, John A. Hall, David Revell, and Daniel Sarewitz. Improving the contribution of climate model information to decision making: the value and demands of robust decision frameworks. *Wiley Interdisciplinary Reviews: Climate Change*, 4(1):39–60, 2013. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wcc.202 (visited on 2019-10-01), doi:10.1002/wcc.202.

[39] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity analysis in practice: a guide to assessing scientific models*. Volume 1. Wiley Online Library, 2004.

[40] Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: a review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016.

[41] O Rakovec, Mary C Hill, MP Clark, AH Weerts, AJ Teuling, and R Uijlenhoet. Distributed evaluation of local sensitivity analysis (delsa), with application to hydrologic models. *Water Resources Research*, 50(1):409–426, 2014.

[42] Andrea Saltelli and Paola Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12):1508–1517, 2010.

[43] Yong Tang, Patrick Reed, Thibaut Wagener, and K van Werkhoven. Comparing sensitivity analysis methods to advance lumped watershed model identification and evaluation. *Hydrology and Earth System Sciences*, 11(2):793–817, 2007.

[44] Nicholas AS Hamm, Jim W Hall, and MG Anderson. Variance-based sensitivity analysis of the probability of hydrologically induced slope instability. *Computers & geosciences*, 32(6):803–817, 2006.

[45] Andrea Saltelli, Ksenia Aleksankina, William Becker, Pamela Fennell, Federico Ferretti, Niels Holst, Sushan Li, and Qiongli Wu. Why so many published sensitivity analyses are false: a systematic review of sensitivity analysis practices. *Environmental modelling & software*, 114:29–39, 2019.

[46] Benjamin P Bryant and Robert J Lempert. Thinking inside the box: a participatory, computer-assisted approach to scenario discovery. *Technological Forecasting and Social Change*, 77(1):34–49, 2010.

[47] Andrea Saltelli and Stefano Tarantola. On the relative importance of input factors in mathematical models: safety assessment for nuclear waste disposal. *Journal of the American Statistical Association*, 97(459):702–709, 2002.

[48] Barry Anderson, Emanuele Borgonovo, Marzio Galeotti, and Roberto Roson. Uncertainty in climate change modeling: can global sensitivity analysis be of help? *Risk analysis*, 34(2):271–293, 2014.

[49] Emanuele Borgonovo. Sensitivity analysis with finite changes: an application to modified eoq models. *European Journal of Operational Research*, 200(1):127–138, 2010.

[50] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[51] Neil R Edwards, David Cameron, and Jonathan Rougier. Precalibrating an intermediate complexity climate model. *Climate dynamics*, 37(7):1469–1482, 2011.

[52] Francesca Pianosi, Keith Beven, Jim Freer, Jim W Hall, Jonathan Rougier, David B Stephenson, and Thorsten Wagener. Sensitivity analysis of environmental models: a systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016.

[53] RC Spear and GM Hornberger. Eutrophication in peel inlet—ii. identification of critical uncertainties via generalized sensitivity analysis. *Water research*, 14(1):43–49, 1980.

[54] Jon C Helton, Jay Dean Johnson, Cedric J Sallaberry, and Curt B Storlie. Survey of sampling-based methods for uncertainty and sensitivity analysis. *Reliability Engineering & System Safety*, 91(10-11):1175–1209, 2006.

[55] Ronald Aylmer Fisher. Design of experiments. *Br Med J*, 1(3923):554–554, 1936.

[56] JD Herman, PM Reed, and T Wagener. Time-varying sensitivity analysis clarifies the effects of watershed model formulation on model behavior. *Water Resources Research*, 49(3):1400–1414, 2013.

[57] Carolina Massmann, Thorsten Wagener, and Hubert Holzmann. A new approach to visualizing time-varying sensitivity indices for environmental model diagnostics across evaluation time-scales. *Environmental modelling & software*, 51:190–194, 2014.

[58] An Van Schepdael, Aurélie Carlier, and Liesbet Geris. Sensitivity analysis by design of experiments. In *Uncertainty in Biology*, pages 327–366. Springer, 2016.

[59] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[60] John Norton. An introduction to sensitivity assessment of simulation models. *Environmental Modelling & Software*, 69:166–174, 2015.

[61] Douglas C Montgomery. *Design and analysis of experiments*. John wiley & sons, 2017.

[62] George EP Box and J Stuart Hunter. The 2 k—p fractional factorial designs. *Technometrics*, 3(3):311–351, 1961.

[63] Izabella Surowiec, Ludvig Vikstrom, Gustaf Hector, Erik Johansson, Conny Vikstrom, and Johan Trygg. Generalized subset designs in analytical chemistry. *Analytical chemistry*, 89(12):6491–6497, 2017.

[64] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(1):239–2451, 1979.

[65] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American statistical association*, 88(424):1392–1397, 1993.

[66] Ishaan L Dalal, Deian Stefan, and Jared Harwayne-Gidansky. Low discrepancy sequences for monte carlo simulations on reconfigurable platforms. In *2008 International Conference on Application-Specific Systems, Architectures and Processors*, 108–113. IEEE, 2008.

[67] SK Zaremba. The mathematical basis of monte carlo and quasi-monte carlo methods. *SIAM review*, 10(3):303–314, 1968.

[68] Sergei Kucherenko, Daniel Albrecht, and Andrea Saltelli. Exploring multi-dimensional spaces: a comparison of latin hypercube and quasi monte carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015.

[69] Bertrand Iooss, Loïc Boussouf, Vincent Feuillard, and Amandine Marrel. Numerical studies of the metamodel fitting and validation processes. *arXiv preprint arXiv:1001.1049*, 2010.

[70] Ruichen Jin, Wei Chen, and Agus Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 37009, 545–554. 2003.

[71] Max D Morris and Toby J Mitchell. Exploratory designs for computational experiments. *Journal of statistical planning and inference*, 43(3):381–402, 1995.

[72] Jeong-Soo Park. Optimal latin-hypercube designs for computer experiments. *Journal of statistical planning and inference*, 39(1):95–111, 1994.

[73] Ilya M Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[74] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[75] Max D Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[76] RI Cukier, CM Fortuin, Kurt E Shuler, AG Petschek, and JH Schaibly. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory. *The Journal of chemical physics*, 59(8):3873–3878, 1973.

[77] Andrea Saltelli, Stefano Tarantola, and KP-S Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.

[78] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[79] Jonathan D Herman, Harrison B Zeff, Jonathan R Lamontagne, Patrick M Reed, and Gregory W Characklis. Synthetic drought scenario generation to support bottom-up water supply vulnerability assessments. *Journal of Water Resources Planning and Management*, 142(11):04016050, 2016.

[80] PCD Milly, Julio Betancourt, Malin Falkenmark, Robert M Hirsch, Zbigniew W Kundzewicz, Dennis P Lettenmaier, and Ronald J Stouffer. Stationarity is dead: whither water management? *Earth*, 4:20, 2008.

[81] Edoardo Borgomeo, Christopher L Farmer, and Jim W Hall. Numerical rivers: a synthetic streamflow generator for water resources vulnerability assessments. *Water Resources Research*, 51(7):5382–5405, 2015.

[82] Manuel Herrera, Sukumar Natarajan, David A Coley, Tristan Kershaw, Alfonso P Ramallo-González, Matthew Eames, Daniel Fosas, and Michael Wood. A review of current and future weather data for building simulation. *Building Services Engineering Research and Technology*, 38(5):602–627, 2017.

[83] Daniel S Wilks and Robert L Wilby. The weather generation game: a review of stochastic weather models. *Progress in physical geography*, 23(3):329–357, 1999.

[84] JR Lamontagne and JR Stedinger. Generating synthetic streamflow forecasts with specified precision. *Journal of Water Resources Planning and Management*, 144(4):04018007, 2018.

[85] Sanghamitra Medda and Kalyan Kumar Bhar. Comparison of single-site and multi-site stochastic models for streamflow generation. *Applied Water Science*, 9(3):67, 2019.

[86] Brian R Kirsch, Gregory W Characklis, and Harrison B Zeff. Evaluating the impact of alternative hydro-climate scenarios on transfer agreements: practical improvement for generating synthetic streamflows. *Journal of Water Resources Planning and Management*, 139(4):396–406, 2013.

[87] Daniel P Loucks and Eelco Van Beek. *Water resource systems planning and management: An introduction to methods, models, and applications*. Springer, 2017.

[88] Scott Steinschneider, Sungwook Wi, and Casey Brown. The integrated effects of climate and hydrologic uncertainty on future flood risk assessments. *Hydrological Processes*, 29(12):2823–2839, 2015.

[89] Richard M Vogel. Stochastic watershed models for hydrologic risk management. *Water Security*, 1:28–35, 2017.

[90] Richard M Vogel and Jery R Stedinger. The value of stochastic streamflow models in overyear reservoir design applications. *Water Resources Research*, 24(9):1483–1490, 1988.

[91] Emanuele Borgonovo. Sensitivity analysis of model output with input constraints: a generalized rationale for local methods. *Risk Analysis: An International Journal*, 28(3):667–680, 2008.

[92] Bertrand Iooss and Paul Lemaître. A review on global sensitivity analysis methods. In *Uncertainty management in simulation-optimization of complex systems*, pages 101–122. Springer, 2015.

[93] Francesca Campolongo and Roger Braddock. The use of graph theory in the sensitivity analysis of the model output: a second order screening method. *Reliability Engineering & System Safety*, 64(1):1–12, 1999.

[94] Roger A Cropp and Roger D Braddock. The new morris method: an efficient second-order screening method. *Reliability Engineering & System Safety*, 78(1):77–83, 2002.

[95] Jon C Helton. Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal. *Reliability Engineering & System Safety*, 42(2-3):327–367, 1993.

[96] Gemma Manache and Charles S Melching. Identification of reliable regression-and correlation-based sensitivity measures for importance ranking of water-quality model parameters. *Environmental Modelling & Software*, 23(5):549–562, 2008.

[97] F Pappenberger and Keith J Beven. Ignorance is bliss: or seven reasons not to use uncertainty analysis. *Water resources research*, 2006.

[98] Jerome H Friedman and Nicholas I Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, 1999.

[99] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[100] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[101] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[102] George M Hornberger and Robert C Spear. Approach to the preliminary analysis of environmental systems. *J. Environ. Mgmt.*, 12(1):7–18, 1981.

[103] Robert J Lempert, David G Groves, Steven W Popper, and Steve C Bankes. A general, analytic method for generating robust strategies and narrative scenarios. *Management science*, 52(4):514–528, 2006.

[104] David G Groves and Robert J Lempert. A new analytic method for finding policy-relevant scenarios. *Global Environmental Change*, 17(1):73–85, 2007.

[105] Keith Beven and Andrew Binley. Glue: 20 years on. *Hydrological processes*, 28(24):5897–5918, 2014.

[106] Roberta-Serena Blasone, Jasper A Vrugt, Henrik Madsen, Dan Rosbjerg, Bruce A Robinson, and George A Zyvoloski. Generalized likelihood uncertainty estimation (glue) using adaptive markov chain monte carlo sampling. *Advances in Water Resources*, 31(4):630–648, 2008.

[107] SA Cryer and PL Havens. Regional sensitivity analysis using a fractional factorial method for the usda model gleams. *Environmental modelling & software*, 14(6):613–624, 1999.

[108] Pengfei Wei, Zhenzhou Lu, and Xiukai Yuan. Monte carlo simulation for moment-independent sensitivity analysis. *Reliability Engineering & System Safety*, 110:60–67, 2013.

[109] Peter Young. Data-based mechanistic modelling, generalised sensitivity and dominant mode analysis. *Computer Physics Communications*, 117(1-2):113–129, 1999.

[110] Andrea Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer physics communications*, 145(2):280–297, 2002.

[111] Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590, 2002.

[112] Toshimitsu Homma and Andrea Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

[113] Gregory J McRae, William R Goodin, and John H Seinfeld. Development of a second-generation mathematical model for urban air pollution—i. model formulation. *Atmospheric Environment (1967)*, 16(4):679–696, 1982.

[114] Andrea Saltelli and Ricardo Bolado. An alternative way to compute fourier amplitude sensitivity test (fast). *Computational Statistics & Data Analysis*, 26(4):445–460, 1998.

[115] MA Vazquez-Cruz, R Guzman-Cruz, IL Lopez-Cruz, O Cornejo-Perez, I Torres-Pacheco, and RG Guevara-Gonzalez. Global sensitivity analysis by means of efast and sobol'methods and calibration of reduced state-variable tomgro model using genetic algorithms. *Computers and Electronics in Agriculture*, 100:1–12, 2014.

[116] Benjamin Auder and Bertrand Iooss. Global sensitivity analysis based on entropy. In *Safety, reliability and risk analysis-Proceedings of the ESREL 2008 Conference*, 2107–2115. 2008.

[117] Farkhondeh Khorashadi Zadeh, Jiri Nossent, Fanny Sarrazin, Francesca Pianosi, Ann van Griensven, Thorsten Wagener, and Willy Bauwens. Comparison of variance-based and moment-independent global sensitivity analysis approaches by application to the swat model. *Environmental Modelling & Software*, 91:210–222, 2017.

[118] Francesca Pianosi and Thorsten Wagener. A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environmental Modelling & Software*, 67:1–11, 2015.

[119] Ronald Aylmer Fisher and others. Statistical methods for research workers. *Statistical methods for research workers.*, 1934.

[120] Loıc Brevault, Mathieu Balesdent, Nicolas Bérend, and Rodolphe Le Riche. Comparison of different global sensitivity analysis methods for aerospace vehicle optimal design. In *10th World Congress on Structural and Multidisciplinary Optimization, WCSMO-10*. 2013.

[121] GEB Archer, Andrea Saltelli, and IM Sobol. Sensitivity measures, anova-like techniques and the use of bootstrap. *Journal of Statistical Computation and Simulation*, 58(2):99–120, 1997.

[122] Art B Owen. Variance components and generalized sobol'indices. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):19–41, 2013.

[123] Emanuele Borgonovo. Measuring uncertainty importance: investigation and comparison of alternative approaches. *Risk analysis*, 26(5):1349–1361, 2006.

[124] Emanuele Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007.

[125] Elmar Plischke, Emanuele Borgonovo, and Curtis L Smith. Global sensitivity measures from given data. *European Journal of Operational Research*, 226(3):536–550, 2013.

[126] Jiri Nossent, Pieter Elsen, and Willy Bauwens. Sobol' sensitivity analysis of a complex environmental model. *Environmental Modelling & Software*, 26(12):1515–1525, 2011. URL: https://www.sciencedirect.com/science/article/pii/S1364815211001939, doi:https://doi.org/10.1016/j.envsoft.2011.08.010.

[127] Jon Herman and Will Usher. Salib: an open-source python library for sensitivity analysis. *Journal of Open Source Software*, 2(9):97, 2017.

[128] Hoshin V Gupta, Thorsten Wagener, and Yuqiong Liu. Reconciling theory with observations: elements of a diagnostic approach to model evaluation. *Hydrological Processes: An International Journal*, 22(18):3802–3813, 2008.

[129] Keith Beven and Jim Freer. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the glue methodology. *Journal of hydrology*, 249(1-4):11–29, 2001.

[130] Cameron McPhail, HR Maier, JH Kwakkel, M Giuliani, A Castelletti, and S Westra. Robustness metrics: how are they calculated, when should they be used and why do they give different results? *Earth's Future*, 6(2):169–191, 2018.

[131] John D Sterman. System dynamics modeling: tools for learning in a complex world. *California management review*, 43(4):8–25, 2001.

[132] John M Anderies, Jean-Denis Mathias, and Marco A Janssen. Knowledge infrastructure and safe operating spaces in social–ecological systems. *Proceedings of the National Academy of Sciences*, 116(12):5277–5284, 2019.

[133] Rachata Muneepeerakul and John M Anderies. The emergence and resilience of self-organized governance in coupled infrastructure systems. *Proceedings of the National Academy of Sciences*, 117(9):4617–4622, 2020.

[134] Antonia Hadjimichael, Patrick M Reed, and Julianne D Quinn. Navigating deeply uncertain tradeoffs in harvested predator-prey systems. *Complexity*, 2020.

[135] Julianne D Quinn, Patrick M Reed, and Klaus Keller. Direct policy search for robust multi-objective management of deeply uncertain socio-ecological tipping points. *Environmental modelling & software*, 92:125–141, 2017.

[136] S. R. Carpenter, D. Ludwig, and W. A. Brock. Management of Eutrophication for Lakes Subject to Potentially Irreversible Change. *Ecological Applications*, 9(3):751–771, August 1999. URL: http://onlinelibrary.wiley.com/doi/10.1890/1051-0761(1999)009{[}0751:MOEFLS{]}2.0.CO;2/abstract, doi:10.1890/1051-0761(1999)009[0751:MOEFLS]2.0.CO;2.

[137] Julianne Quinn. Julianneq/Lake_problem_dps. December 2017. original-date: 2017-02-06T18:33:54Z. URL: https://github.com/julianneq/Lake_Problem_DPS (visited on 2021-06-14).

[138] David Hadka. Project-Platypus/Rhodium. 2017. original-date: 2015-10-29T18:08:43Z. URL: https://github.com/Project-Platypus/Rhodium (visited on 2021-06-14).

[139] Stephen R. Carpenter, William A. Brock, Carl Folke, Egbert H. van Nes, and Marten Scheffer. Allowing variance may enlarge the safe operating space for exploited ecosystems. *Proceedings of the National Academy of Sciences*, 112(46):14384–14389, November 2015. URL: http://www.pnas.org/content/112/46/14384 (visited on 2017-08-18), doi:10.1073/pnas.1511804112.

[140] Mustafa Hekimoğlu and Yaman Barlas. Sensitivity analysis for models with multiple behavior modes: a method based on behavior pattern measures. *System Dynamics Review*, 32(3-4):332–362, 2016.

[141] Patrick Steinmann, Willem L Auping, and Jan H Kwakkel. Behavior-based scenario discovery using time series clustering. *Technological Forecasting and Social Change*, 156:120052, 2020.

[142] Steven C Bankes, Robert J Lempert, and Steven W Popper. Computer-assisted reasoning. *Computing in Science & Engineering*, 3(2):71–77, 2001.

[143] Robert J. Lempert, Steven W. Popper, and Steven C. Bankes. *Shaping the Next One Hundred Years*. RAND Corporation, 2003. URL: https://www.rand.org/pubs/monograph_reports/MR1626.html (visited on 2017-09-14).

[144] Jonathan R Lamontagne, Patrick M Reed, Robert Link, Katherine V Calvin, Leon E Clarke, and James A Edmonds. Large ensemble analytic framework for consequence-driven discovery of climate change scenarios. *Earth's Future*, 6(3):488–504, 2018.

[145] Brian C O'Neill, Elmar Kriegler, Keywan Riahi, Kristie L Ebi, Stephane Hallegatte, Timothy R Carter, Ritu Mathur, and Detlef P van Vuuren. A new scenario framework for climate change research: the concept of shared socioeconomic pathways. *Climatic change*, 122(3):387–400, 2014.

[146] Warren E Walker, Marjolijn Haasnoot, and Jan H Kwakkel. Adapt or perish: a review of planning approaches for adaptation under deep uncertainty. *Sustainability*, 5(3):955–979, 2013.

[147] Suraje Dessai, Mike Hulme, Robert Lempert, and Roger Pielke Jr. Climate prediction: a limit to adaptation. *Adapting to climate change: thresholds, values, governance*, 64:78, 2009.

[148] Jonathan D Herman, Patrick M Reed, Harrison B Zeff, and Gregory W Characklis. How should robustness be defined for water systems planning under change? *Journal of Water Resources Planning and Management*, 141(10):04015012, 2015.

[149] Robert J Lempert. Robust decision making (rdm). In *Decision making under deep uncertainty*, pages 23–51. Springer, Cham, 2019.

[150] Jan H Kwakkel and Marjolijn Haasnoot. Supporting dmdu: a taxonomy of approaches and tools. In *Decision Making under Deep Uncertainty*, pages 355–374. Springer, Cham, 2019.

[151] BC Trindade, PM Reed, and GW Characklis. Deeply uncertain pathways: integrated multi-city regional water supply infrastructure investment and portfolio management. *Advances in Water Resources*, 134:103442, 2019.

[152] Julianne D Quinn, Patrick M Reed, Matteo Giuliani, Andrea Castelletti, Jared W Oyler, and Robert E Nicholas. Exploring how changing monsoonal dynamics and human pressures challenge multireservoir management for flood protection, hydropower production, and agricultural water supply. *Water Resources Research*, 54(7):4638–4662, 2018.

[153] DF Gold, PM Reed, BC Trindade, and GW Characklis. Identifying actionable compromises: navigating multi-city robustness conflicts to discover cooperative safe operating spaces for regional water supply portfolios. *Water Resources Research*, 55(11):9024–9050, 2019.

[154] JR Lamontagne, PM Reed, G Marangoni, K Keller, and GG Garner. Robust abatement pathways to tolerable climate futures require immediate global action. *Nature Climate Change*, 9(4):290–294, 2019.

[155] Antonia Hadjimichael, Julianne Quinn, Erin Wilson, Patrick Reed, Leon Basdekas, David Yates, and Michelle Garrison. Defining Robustness, Vulnerabilities, and Consequential Scenarios for Diverse Stakeholder Interests in Institutionally Complex River Basins. *Earth's Future*, 8(7):e2020EF001503, 2020. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020EF001503 (visited on 2020-07-13), doi:10.1029/2020EF001503.

[156] Harris Drucker and Corinna Cortes. Boosting decision trees. *Advances in neural information processing systems*, pages 479–485, 1996.

[157] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[158] Kelsey L. Ruckert, Gary Shaffer, David Pollard, Yawen Guan, Tony E. Wong, Chris E. Forest, and Klaus Keller. Assessing the Impact of Retreat Mechanisms in a Simple Antarctic Ice Sheet Model Using Bayesian Calibration. *PLOS ONE*, 12(1):e0170052, January 2017. doi:10.1371/journal.pone.0170052.

[159] B. Efron and R. Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1):54–75, February 1986. doi:10.1214/ss/1177013815.

[160] Ryan L. Sriver, Robert J. Lempert, Per Wikman-Svahn, and Klaus Keller. Characterizing uncertain sea-level rise projections to support investment decisions. *PLOS ONE*, 13(2):e0190641, February 2018. URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0190641 (visited on 2021-06-09), doi:10.1371/journal.pone.0190641.

[161] Kelsey L. Ruckert, Yawen Guan, Alexander M. R. Bakker, Chris E. Forest, and Klaus Keller. The effects of time-varying observation errors on semi-empirical sea-level projections. *Climatic Change*, 140(3):349–360, February 2017. URL: https://doi.org/10.1007/s10584-016-1858-z (visited on 2021-06-09), doi:10.1007/s10584-016-1858-z.

[162] Neil R Edwards, David Cameron, and Jonathan Rougier. Precalibrating an intermediate complexity climate model. *Clim. Dyn.*, 37(7-8):1469–1482, 2011. URL: http://dx.doi.org/10.1007/s00382-010-0921-0, doi:10.1007/s00382-010-0921-0.

[163] Alexis Boukouvalas, Pete Sykes, Dan Cornford, and Hugo Maruri-Aguilar. Bayesian Precalibration of a Large Stochastic Microsimulation Model. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1337–1347, June 2014. doi:10.1109/TITS.2014.2304394.

[164] David Makowski, Daniel Wallach, and Marie Tremblay. Using a Bayesian approach to parameter estimation; comparison of the GLUE and MCMC methods. *Agronomie*, 22(2):191–203, 2002. Publisher: EDP Sciences.

[165] Mahyar Shafii, Bryan Tolson, and Loren Shawn Matott. Uncertainty-based multi-criteria calibration of rainfall-runoff models: a comparative study. *Stochastic Environmental Research and Risk Assessment*, 28(6):1493–1510, August 2014. doi:10.1007/s00477-014-0855-x.

[166] Keith Beven and Jim Freer. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. *Journal of hydrology*, 249(1-4):11–29, 2001. Publisher: Elsevier.

[167] Jasper A. Vrugt and Keith J. Beven. Embracing equifinality with efficiency: Limits of Acceptability sampling using the DREAM(LOA) algorithm. *Journal of Hydrology*, 559:954–971, April 2018. doi:10.1016/j.jhydrol.2018.02.026.

[168] Jery R. Stedinger, Richard M. Vogel, Seung Uk Lee, and Rebecca Batchelder. Appraisal of the generalized likelihood uncertainty estimation (GLUE) method. *Water Resources Research*, 2008. doi:10.1029/2008WR006822.

[169] Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, March 2013. ISBN 978-1-4757-3071-5.

[170] Christian P. Robert. The Metropolis–Hastings Algorithm. In *Wiley StatsRef: Statistics Reference Online*, pages 1–15. American Cancer Society, 2015. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat07834 (visited on 2021-06-14), doi:10.1002/9781118445112.stat07834.

[171] James M. Flegal, Murali Haran, and Galin L. Jones. Markov Chain Monte Carlo: Can We Trust the Third Significant Figure? *Statistical Science*, 23(2):250–260, May 2008. Publisher: Institute of Mathematical Statistics. URL: https://projecteuclid.org/journals/statistical-science/volume-23/issue-2/Markov-Chain-Monte-Carlo--Can-We-Trust-the-Third/10.1214/08-STS257.full (visited on 2021-06-14), doi:10.1214/08-STS257.

[172] Carla Currin, Toby Mitchell, Max Morris, and Don Ylvisaker. Bayesian Prediction of Deterministic Functions, with Applications to the Design and Analysis of Computer Experiments. *Journal of the American Statistical Association*, 86(416):953–963, December 1991. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1991.10475138. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1991.10475138 (visited on 2021-06-14), doi:10.1080/01621459.1991.10475138.

[173] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409–423, 1989. Publisher: Institute of Mathematical Statistics. URL: https://www.jstor.org/stable/2245858 (visited on 2021-06-14).

[174] Roger G. Ghanem and Pol D. Spanos. Spectral Stochastic Finite-Element Formulation for Reliability Analysis. *Journal of Engineering Mechanics*, 117(10):2351–2372, October 1991. Publisher: American Society of Civil Engineers. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9399%281991%29117%3A10%282351%29 (visited on 2021-06-14), doi:10.1061/(ASCE)0733-9399(1991)117:10(2351).

[175] Dongbin Xiu and George Em Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, January 2002. Publisher: Society for Industrial and Applied Mathematics. URL: https://epubs.siam.org/doi/abs/10.1137/S1064827501387826 (visited on 2021-06-14), doi:10.1137/S1064827501387826.

[176] John Eason and Selen Cremaschi. Adaptive sequential sampling for surrogate model generation with artificial neural networks. *Computers & Chemical Engineering*, 68:220–232, September 2014. URL: https://www.sciencedirect.com/science/article/pii/S0098135414001719 (visited on 2021-06-14), doi:10.1016/j.compchemeng.2014.05.021.

[177] Dirk Gorissen, Luciano De Tommasi, Karel Crombecq, and Tom Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5):485–494, June 2009. URL: https://doi.org/10.1007/s00521-008-0223-1 (visited on 2021-06-14), doi:10.1007/s00521-008-0223-1.

[178] Jenný Brynjarsdóttir and Anthony O'Hagan. Learning about physical parameters: the importance of model discrepancy. *Inverse Problems*, 30(11):114007, October 2014. Publisher: IOP Publishing. URL: https://doi.org/10.1088/0266-5611/30/11/114007 (visited on 2021-06-14), doi:10.1088/0266-5611/30/11/114007.

[179] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[180] Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[181] Matti Vihola. Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22(5):997–1008, September 2012. URL: https://doi.org/10.1007/s11222-011-9269-5 (visited on 2021-06-14), doi:10.1007/s11222-011-9269-5.

[182] Perry de Valpine, Daniel Turek, Christopher J. Paciorek, Clifford Anderson-Bergman, Duncan Temple Lang, and Rastislav Bodik. Programming With Models: Writing Statistical Algorithms for General Model Structures With NIMBLE. *Journal of Computational and Graphical Statistics*, 26(2):403–413, April 2017. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2016.1172487. URL: https://doi.org/10.1080/10618600.2016.1172487 (visited on 2021-06-14), doi:10.1080/10618600.2016.1172487.

[183] NIMBLE Development Team. NIMBLE: MCMC, Particle Filtering, and Programmable Hierarchical Modeling. May 2021. URL: https://zenodo.org/record/4829693 (visited on 2021-06-14), doi:10.5281/zenodo.4829693.

[184] Stan Development Team. Stan Modeling Language Users Guide and Reference Manual. 2021. URL: https://mc-stan.org/docs/2_27/stan-users-guide/index.html (visited on 2021-06-14).

[185] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, April 2016. Publisher: PeerJ Inc. URL: https://peerj.com/articles/cs-55 (visited on 2021-06-14), doi:10.7717/peerj-cs.55.

[186] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A Language for Flexible Probabilistic Inference. In *International Conference on Artificial Intelligence and Statistics*, 1682–1690. PMLR, March 2018. ISSN: 2640-3498. URL: http://proceedings.mlr.press/v84/ge18b.html (visited on 2021-06-14).

[187] Andrew Gelman and Donald B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, November 1992. Publisher: Institute of Mathematical Statistics. URL: https://projecteuclid.org/journals/statistical-science/volume-7/issue-4/Inference-from-Iterative-Simulation-Using-Multiple-Sequences/10.1214/ss/1177011136.full (visited on 2021-06-14), doi:10.1214/ss/1177011136.

[188] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2006.00553.x. URL: http://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2006.00553.x (visited on 2021-06-14), doi:10.1111/j.1467-9868.2006.00553.x.

[189] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, July 2000. URL: https://doi.org/10.1023/A:1008935410038 (visited on 2021-06-14), doi:10.1023/A:1008935410038.

[190] Jane Liu and Mike West. Combined Parameter and State Estimation in Simulation-Based Filtering. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 197–223. Springer, New York, NY, 2001. URL: https://doi.org/10.1007/978-1-4757-3437-9_10 (visited on 2021-06-14), doi:10.1007/978-1-4757-3437-9_10.

[191] Stefano Cabras, Maria Eugenia Castellanos Nueda, and Erlis Ruli. Approximate Bayesian Computation by Modelling Summary Statistics in a Quasi-likelihood Framework. *Bayesian Analysis*, 10(2):411–439, June 2015. Publisher: International Society for Bayesian Analysis. URL: https://projecteuclid.org/journals/bayesian-analysis/volume-10/issue-2/Approximate-Bayesian-Computation-by-Modelling-Summary-Statistics-in-a-Quasi/10.1214/14-BA921.full (visited on 2021-06-14), doi:10.1214/14-BA921.

[192] Jarno Lintusaari, Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Fundamentals and Recent Developments in Approximate Bayesian Computation. *Systematic Biology*, 66(1):e66–e82, January 2017. URL: https://doi.org/10.1093/sysbio/syw077 (visited on 2021-06-14), doi:10.1093/sysbio/syw077.

[193] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate Bayesian Computation. *PLOS Computational Biology*, 9(1):e1002803, January 2013. Publisher: Public Library of Science. URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1002803 (visited on 2021-06-14), doi:10.1371/journal.pcbi.1002803.

[194] Edwin T. Jaynes. *Probability theory: the logic of science*. Washington University St. Louis, MO, 1996.

[195] Andrew Gelman, Daniel Simpson, and Michael Betancourt. The Prior Can Often Only Be Understood in the Context of the Likelihood. *Entropy*, 19(10):555, October 2017. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute. URL: https://www.mdpi.com/1099-4300/19/10/555 (visited on 2021-06-14), doi:10.3390/e19100555.

[196] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[197] Andrew Gelman, Xiao-Li Meng, and Hal Stern. Posterior Predictive Assessment of Model Fitness via Realized Discrepancies. *Statistica Sinica*, 6(4):733–760, 1996. Publisher: Institute of Statistical Science, Academia Sinica. URL: https://www.jstor.org/stable/24306036 (visited on 2021-06-14).

[198] Andrew Gelman, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. Bayesian Workflow. *arXiv:2011.01808 [stat]*, November 2020. arXiv: 2011.01808. URL: http://arxiv.org/abs/2011.01808 (visited on 2021-06-14).

[199] Andrew Gelman and Cosma Rohilla Shalizi. Philosophy and the practice of Bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38, 2013. _eprint: https://bpspsychub.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2044-8317.2011.02037.x. URL: https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1111/j.2044-8317.2011.02037.x (visited on 2021-06-14), doi:10.1111/j.2044-8317.2011.02037.x.