

VMC: A Grammar for Visualizing Statistical Model Checks

Ziyang Guo, Alex Kale, Matthew Kay, Jessica Hullman

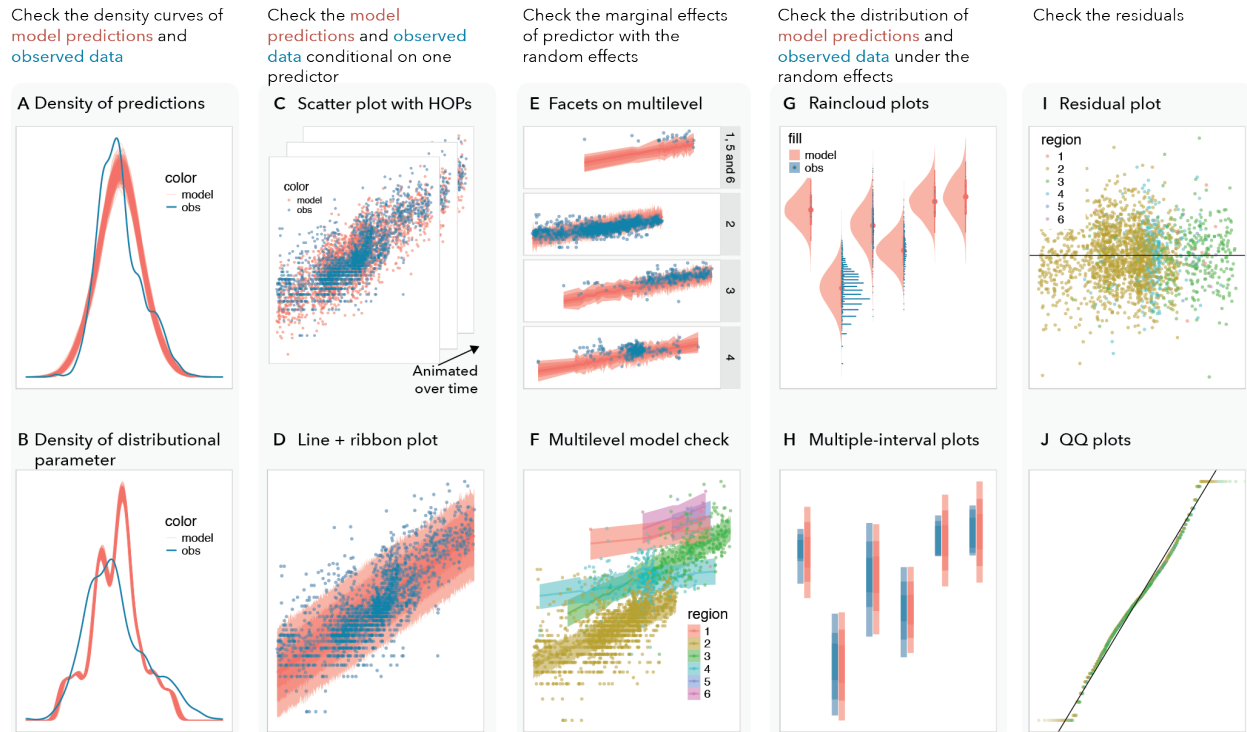


Fig. 1: Example model check visualizations authored with VMC, using data from [46]. From left to right: checks on the density curves of the distributions of model predictions and observed data from (A) response variable to (B) distributional parameter; follow-up checks conditional on the quantitative predictor, where VMC is used to specify (C) Hypothetical Outcome Plots and (D) a line + ribbon plot; (E) a facet check stratifying the random effects and (F) a multilevel check; more checks for the random effects specified by VMC, including (G) raincloud plots and (H) multiple-interval plots; and residual checks specified by VMC, including (I) residual plots revealing the heteroskedasticity of the model and (J) Q-Q plots, validating the normality of residuals.

Abstract—Visualizations play a critical role in validating and improving statistical models. However, the design space of model check visualizations is not well understood, making it difficult for authors to explore and specify effective graphical model checks. VMC defines a model check visualization using four components: (1) samples of distributions of checkable quantities generated from the model, including predictive distributions for new data and distributions of model parameters; (2) transformations on observed data to facilitate comparison; (3) visual representations of distributions; and (4) layouts to facilitate comparing model samples and observed data. We contribute an implementation of VMC as an R package. We validate VMC by reproducing a set of canonical model check examples, and show how using VMC to generate model checks reduces the edit distance between visualizations relative to existing visualization toolkits. The findings of an interview study with three expert modelers who used VMC highlight challenges and opportunities for encouraging exploration of correct, effective model check visualizations.

Index Terms—Model checking and evaluation; Uncertainty visualization; Grammar of Graphics

1 INTRODUCTION

- Ziyang Guo is with Northwestern University. E-mail: ziyang.guo@northwestern.edu.
- Alex Kale is with University of Chicago. E-mail: kalea@uchicago.edu.
- Matthew Kay is with Northwestern University. E-mail: mjskay@northwestern.edu.
- Jessica Hullman is with Northwestern University. E-mail: jhullman@northwestern.edu.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxx

Visualizations play a critical yet undervalued role for checking expectations in statistical modeling workflow. A rigorous statistical analysis often includes numerical or graphical checks of how well the model fits the data and performs for the purpose for which it is developed [13, 15, 16]. Experienced statisticians view numerical and graphical checks as serving complementary roles, and often use them in tandem. For example, it is well-known that while a high R-squared value in a regression model can suggest a good fit, a residual plot might reveal heteroskedasticity or other issues not evident from the R-squared value alone [34]. Model check visualizations also play a powerful role in communicating and helping validate visually-based inferences [15, 22], which can be strengthened by integrating functionality for generating model check visualizations in exploratory visual

data analysis tools [24, 28].

Graphical model checks (*model checks* hereafter) extend far beyond well-known examples like residual plots, and can take many different forms. For example, they can juxtapose the model predictions and observed data side by side. They can directly encode the difference between the two such as in residuals (Figure 1I, J). The visual representations used in model check visualizations also vary widely. They can use different mark types to represent the model and observed data, such as raincloud plot in Figure 1G. They can be animated to reveal the uncertainty in model distributions, such as hypothetical outcome plots in Figure 1C.

A lack of graphical tools for composing or exploring visualizations for model checking restricts the potential for these visualizations to improve statistical practices. Expert modelers tend to rely on their previous experience to guide their design choices. For example, some analysts may only think of residual plots to check assumptions of heteroskedasticity or density estimates to check for systematic discrepancies between model predictions and observed data [13, 16]. Others may gravitate toward other views based on their training (e.g., Bayesian workflow [18]). Reliance on prior experience alone can restrict exploration of model check visualization designs and employment of new techniques for displaying uncertainty [30] or comparative layouts [19].

One way to scale effective visual model checking to more scenarios is to make it easier to generate a space of possible model check visualizations in statistical workflow. Most current methods for specifying model checks either are too low-level to effectively support high-level tasks in model checks or have limited flexibility to explore diverse design choices. For example, the Grammar of Graphics (e.g., `ggplot2` [50] and `Vega-Lite` [45]) specify a visualization as a set of layers comprised of data, geometries, aesthetics, statistical transformations, scales, and guides. However, low-level representations can lead to problems like representational *viscosity*, *hidden dependencies*, and *error-proneness* [6] when applied to designing model check visualizations. For example, switching from a posterior predictive check on a density estimate (Fig. 1A) to a residual plot (Fig. 1I) using `ggplot2` requires changes to data, aesthetics, scales and geometry. On the other hand, task-oriented approaches like `bayesplot` [12, 13] and `performance` [34] only support fixed visualizations for specific tasks, discouraging further exploration. This trade-off between simplicity and expressiveness motivates careful development of appropriate abstractions for model check visualization.

We contribute `VMC`, a high-level declarative grammar for generating model check visualizations. `VMC` categorizes design choices in model check visualizations via four components: *sampling specification*, *data transformation*, *visual representation(s)*, and *comparative layout*. `VMC` improves the state-of-the-art in graphical model check specification in two ways: (1) it allows users to explore a wide range of model checks through relatively small changes to a specification as opposed to more substantial code restructuring, and (2) it simplifies the specification of model checks by defining a small number of semantically-meaningful design components tailored to model checking.

We implement `VMC` as a package in the popular statistical programming language R. By reproducing a range of model check techniques that occur in the statistical literature [16], we demonstrate how (1) the design space specified by the components of `VMC` covers canonical examples of model checks used in statistics and (2) the abstractions employed by `VMC` can reduce edit distance between model check visualization specifications compared to existing general purpose and specialized graphics libraries, e.g., `ggplot2` and `bayesplot`. We further contribute the results of a semi-structured interview study in which three expert modelers applied `VMC` to a model of their choice. We find that `VMC` (1) aligns with participants' understanding and practice of model checks and (2) encourages exploration of new model checks.

2 RELATED WORK

2.1 Visualizations as Model Checks

Model checking is a fundamental part of statistical workflow [7, 8, 16]. Among forms of checks, visualizations that enable graphical checking has long been viewed as an important complement to quantitative

statistical checks [9, 13, 48]. For example, modelers use residual plots to check model assumptions like heteroskedasticity [8] and scatter plots to check for multimodality [2]. Gabry et al. [13] summarizes the usage of graphical checks in each step of the Bayesian statistical workflow and implements these canonical checking examples in an R graphical tool, `bayesplot`.

Model checks are designed to facilitate the comparison between observed data and predictions under the model. The best model check visualizations are said to reduce this task to detecting deviation from symmetry, for which the human visual system is well-optimized [15]. Because model checks are inherently visual, they are hard to theorize, though statisticians have identified properties expected of a good model check. For example, Li et al. [33] suggest a model check should be well-calibrated, computationally efficient, and easy-to-use. We develop `VMC` to address obstructions to the last property when it comes to existing libraries for constructing model checks.

More broadly, some prior work has explored the idea of strengthening the connection between visualization and statistical model checking. Early work in statistical graphics emphasized the value of trellis plots for checking for main effects and interactions between variables in plots of observed data [5]. Gelman proposed a theoretical formulation of visualizations as model checks to help bridge seemingly opposed traditions of exploratory and confirmatory data analysis [14, 15]. Other authors have similarly proposed an analogy of a visualization as a statistical test, and contributed graphical tools like the line-up to facilitate graphical statistical inference [21, 37, 51]. Hullman and Gelman compare statistical perspectives on the analogy between visualizations and statistical modeling, likening the examination of visualization to posterior predictive checking in a Bayesian workflow and deriving design implications of this view for interactive visual analysis software [23, 24]. Kale et al. [28] realize a version of this vision in `EVM`, a Tableau-style visualization tool that incorporates lightweight specification and graphical checking of models to help users express and check their provisional interpretations of data in exploratory visual analysis. While generally associated with traditional statistical modeling, visual model checking also plays a role in machine learning pipelines, where the goals of a `VIS4ML` system often hinge on the ability of the system user to identify forms of model misfit [47]. The diversity of applications for model checking emphasize the need for tools and theory to be agnostic to the specific goals and form of the model. Despite its importance as a statistical tool, there is little practical guidance on how to evaluate or construct model check visualizations [24, 39].

2.1.1 Visualizations for comparison

Model checks are a particular type of visualization-aided comparison. Gleicher et al. [19] discuss the challenges in designing visualizations for aiding comparison and propose a taxonomy for comparison techniques in visualizations, which informs the comparative layout component of our grammar. Elsewhere, canonical work in visualization highlights the value of faceting data for checking implicit models [5]. Our work extends understanding of visual comparison by focusing specifically on the design space for facilitating comparison between model predictions and observed data.

2.2 Visualization Grammars

Visualization grammars can provide a more rigorous basis for generating visualizations over chart-type typologies, as well as a principled way of generating a search space of visualizations for visualization recommenders to act on. For example, the *Grammar of Graphics* (implemented by `ggplot2`) [52] specification uses layered components to encode visual representation layers, scales for aesthetic attributes, a coordinate system to map the position on the plot, and facets to partition data into subsets. Some visualization grammars target more specific domains. For example, `ggdist` [30] and `PGoG` [42] provide a particular formalism for specifying uncertainty visualizations as an extension to `ggplot2`, and *IEMA* [4] provides a formalism to specifying explanatory model analysis.

However, existing visualization grammars are not optimized for specifying model checks. Constructing model checks usually involves

techniques like extracting samples from the model, visualizing the samples in visual representation, and comparing model predictions and observed data in layouts. However, as discussed by literature [43], existing visualization grammars such as `ggplot2`, can lead to specific errors such as mismatched data and visualization due to the lack of tight couplings between data transformation and visualization code and the difficulty of keeping these in sync. Model check libraries like `bayesplot` [13] provide users a quick API to define a model check visualization by generating fixed plots for certain model check tasks, e.g., a density plot for posterior predictive checks and residual plots for checking heteroskedasticity. However, they are not meant to provide a systematic understanding of what constitutes a model check that users can rely on to explore or invent designs.

3 DESIGN OBJECTIVES

Drawing on the aforementioned prior work on model checking workflow, we derived four design objectives for a model check visualization grammar.

O1 Expressiveness. To scale the benefits of model checking to many scenarios, a model check visualization grammar should be able to express as many model check design strategies as possible, incorporating different visualization techniques. One way to enable visualization flexibility is to componentize data preparation and visual specification separately. Data preparation components should support varying methods to access the model and observed data. The visual aspects of a grammar should control how to render the model and observed data (e.g., represent the model by lines and observed data by points) and how to lay them out to facilitate comparison (e.g., aligning the plot for the model and the plot for the observed data spatially or overlaying them in one same coordinate system).

O2 Correctness Following other task-oriented visualization grammars [42], a model check visualization grammar should support generation of “correct” examples only. In the context of model check visualization, correctness corresponds to specifications of visualization that are a function of both model predictions and observed data. Prior work on visualization grammars suggest that one way to avoid generation of incorrect examples is for the visualization grammar to tightly couple the data and visualization specifications, as mismatch between them can lead to errors [43]. Maintaining this synchronization can be particularly complex in the case of model check visualizations, which requires keeping two different data pipelines (for observed data and model predictions) and their corresponding representations in sync.

O3 Closeness to the experts’ language A model check visualization grammar should use abstractions that resemble language experts use in specifying model checks, including both to describe the relevant data processes and to specify the visualization. Ideally, a grammar will enable specifying the data corresponding to a desired model check visualization at a high level corresponding to major distinctions between methods experts care about but with sufficient control over crucial details like sampling methods that experts may care about. For visual choices, possible specifications should correspond to the space of combinations of meaningful variations in design choices observed in expert model checking workflow.

O4 Exploration. Following other grammars that facilitate exploration, a grammar should be component reusable [20], coherent and systematic [10]. For component reusability, the components of the specification of a model check visualization grammar should be independent of one another such that changing one component does not necessarily require changing others. To be coherent and systematic, the specification of a model check visualization grammar should be hierarchical and the components should control different parts of model check visualization in a systematic way.

4 VMC: GRAMMAR FOR MODEL CHECKS

VMC is a declarative grammar designed to support concisely and flexibly specifying model check visualizations. VMC allows users to specify the data preparation and visual design of model check visualizations independently (**O1** and **O4**) with tailored components for common model checking tasks (**O3**). A single VMC specification defines how to generate

samples from the model (`Sampling_spec`), transform the observed data (`Data_transformation`), visualize the model samples and the observed data (`Visual_representation`) and set up comparison between the model and the observed data (`Comparative_layout`). VMC takes an input `model` and a set of `observed data` to be compared. The formal specification of VMC is shown in Figure 2.

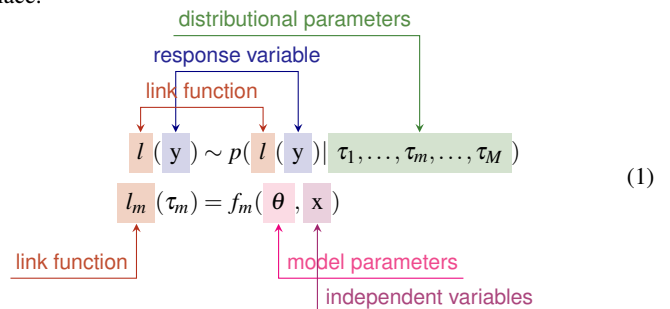
4.1 Sampling Specification

The first step to specifying a model check visualization is to define a way to access the fitted model. We build the `Sampling_spec` component of VMC to control how to sample from the model. We designed VMC’s `Sampling_spec` to be robust to model types and various sampling methods in model checks to fulfill the objective of expressiveness (**O1**). At the meantime, it should also have arguments to succinctly express conceptually distinct sample generation options to maintain closeness to experts’ language (**O3**) and facilitate exploration (**O4**). The `Sampling_spec` component of VMC covers varieties of sampling methods with three arguments: (1) the quantity to be generated from the model, (2) the data set of predictor values, and (3) the number of draws.

There are many different model-defined quantities that can be useful to check in model check visualizations [14, 44, 49]. For example, in Bayesian data analysis [14], modelers frequently sample from the posterior distribution of the model parameters in addition to drawing from the posterior predictive distribution. The quantities that are being checked also vary with the choice of link functions. For example, modelers often check the link-scale predictions of the logarithm of the odds ratio in logistic regression models [36].

`Sampling_spec` formalizes the quantities within a statistical model by combining link functions with the response variable and distributional parameters, if applicable. For example, in the model shown in Equation 1, the quantities included by `Sampling_spec` are y , $l(y)$, τ_1, \dots, τ_M and $l_1(\tau_1), \dots, l_M(\tau_M)$. The definition of quantities within `Sampling_spec` is designed to balance expressiveness and redundancy; we aimed for `Sampling_spec` to express as many model quantities as possible while ensuring that it only includes quantities that are *checkable*.

A *checkable* quantity is one for which there exists a method for comparing it with observed data. Distributional parameters often correspond to statistics computed on the observed data, such as μ in a Gaussian distribution corresponding to the mean of the observed data and σ corresponding to the standard deviation of observed data. There will however be cases where we cannot check arbitrary quantities generated under the model against observed data. For example, consider a scenario where an author wants to check α in the model: $y \sim Normal(\mu, \phi), \mu = \alpha + \beta x$. To compare with α in the model requires prior information on the true data generating process of the response variable y . However, uncertainty about the true data generating process is typically why explanatory models are used in the first place.



4.2 Data Transformation

Given the varieties of data transformations that can be applied in model check visualizations [13, 14, 17], we define `Data_transformation` as a function that takes in a data set of observed data and outputs a data set with the same format as the input data. For example, instead of the original empirical distribution of the observed data, modelers may want to compare the model to certain statistics computed on the


```

VMC_Spec := Model, Data, Sampling_spec, Data_transformation,
          Visual_representation, Comparative_layout

Model := function(<Predictors>) → <Response_var>
Data := <Predictors, Response_var>[]
Predictors := <Any>[]
Response_var := <Any>

Sampling_spec := Quantity, Predictor_values, Ndraws
Quantity := y | mu | sigma | phi | ...
Predictor_values := <Predictors>[]
Ndraws := <Number>

Data_transformation := function(<Data>) → <Data>

Visual_representation := <Target, Mark, Group_samples?>[]

Target := model | data
Mark := densityline | slab | violin | histogram | interval |
       gradient | heatmap | quantiledots | line | point | ...
Group_samples := collapsing | individualizing | animating |
               Aggregation
Aggregation := function(<<Response_var>[]>) →
              <Response_var>

Comparative_layout := superposition | juxtaposition |
                     nest_juxtaposition | Explicit_encoding
Explicit_encoding := residual | Q-Q | detrended_Q-Q |
                  Customized_encoding
Customized_encoding := function(<Samples, Data>) →
                       <Samples>
Samples := <Predictors, Response_var, Draw_id>[]
Draw_id := <Number>

```

Notation
 "a := b, c": a is defined as a tuple of b and c, "a?": a is an optional argument, "...": extensible argument,
 "<Abc>": data type, "a|b|c": either one of a, b, or c, "<A, B>[]": a list of a tuple of A and B

Fig. 2: The formal specification of VMC including the four main components: Sampling_spec, Data_transformation, Visual_representation and Comparative_layout.

observed data, e.g., mean or median [14]. Some transformations on the observed data are meant to help test a hypothesis, e.g., whether the model’s predictive distribution captures the mean or median of the observed data. Others are performed to ensure comparability of the model quantity and observed data. For example, when model samples are generated from the distribution of the σ parameter in a Gaussian regression model, the model quantity is on a variance scale. In this case, the observed data should first be transformed to its standard deviation.

4.3 Visual Representation

Effective visual representations—those that can enable the analyst to identify discrepancies between model predictions and observed data—are critical to the utility of model checking in a statistical workflow. The third component of VMC, `Visual_representation`, is used to specify the visual representations of the model and observed data separately. It enables specifying a list of visual representations to compose a model check visualization. Authors can specify more than one visual representation for model predictions and observed data. For each visual representation for model predictions, in addition to the mark type, they can specify the method to use to group the samples in that representation.

4.3.1 Choice of mark type

Drawing on research on visualizing distributional information [30, 42], we observe that visualizations of uncertainty can be distinguished based on whether they use the area or extent of a mark to encode distributional information (e.g., a confidence interval or range), a visual variable such as color to encode probability, or a discrete (countable) representation in which distinct marks are used to visually compose a distribution. VMC defines three categories for marks:

EXTENT : *densityline, slab, violin, histogram, interval, ...*

VISUAL VARIABLE : *gradient, heatmap, ...*

COUNTABLE : *quantiledots, line, point, ...*

In practice, the choice of mark type for visually representing distribution in model checks may be driven by the audience of the model checks and specific model checking tasks. For example, if the audience is broader than just experts, the authors may opt for a mark type from the countable category, e.g., quantile dotplots [31], since countable visual representations appear to outperform continuous equivalents like error bars or densities for some tasks for lay audiences [11, 25, 26, 41]. (The meaning of *collapsing* and *individualizing* in these specifications is described in section 4.3.2.)

```

Visual_representation ←
  [(model, quantiledots,
    collapsing),
   (data, quantiledots)]

```




The choice of mark type can also depend on the specific task the model check is designed for. For example, the author may prefer a mark type based on intervals if they want to check calibration (i.e., whether the model predictions capture the expectation of observed data) or heteroskedasticity (i.e., whether the variance of residuals is constant).

```

Visual_representation ←
  [(model, interval,
    collapsing),
   (data, interval)]

```

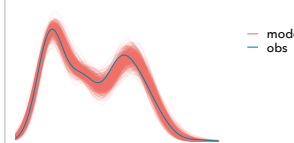


If they want to check the shape of the distribution, they may prefer densities.

```

Visual_representation ←
  [(model, densityline,
    individualizing),
   (data, densityline)]

```

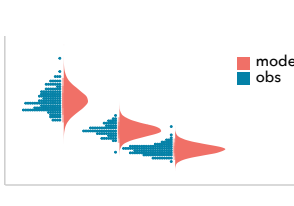


They may also want to show more complex information about the model predictions and observed data, leading them to adopt hybrid mark types. For example, statisticians have used raincloud plots [1], a combination of several chart types, to visualize the raw observed data, the distribution of the model predictions as density, and key summary statistics at the same time.

```

Visual_representation ←
  [(model, slab,
    collapsing),
   (model, interval,
    collapsing),
   (data, quantiledots)]

```



4.3.2 Grouping samples

By sampling from the model, we yield multiple sets of model samples for visualizing. Different approaches to grouping these samples prior to applying visual marks produce different visual formats that may differ in how well they support various tasks. We summarise four ways of grouping samples into visual marks observed in model checking practice.

First, one common approach is to *collapse* all samples into one visual mark. Consider the example of a line ribbon plot in Figure 1D: there is only one mark (line + ribbons for multiple confidence intervals), which is calculated from the data of all the samples. Another way is to use *individual* marks for each of the samples. For example, in the posterior predictive check in Figure 1A, each of the light red line represents the density estimate of one sample. Hypothetical outcome

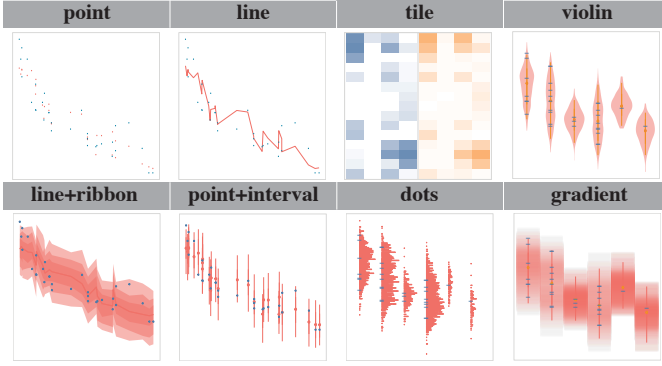


Fig. 3: A subset of visual representations that VMC can specify. Not shown: Animation (HOPs) can be applied upon any

plots (HOPs) [26] animate one mark of one sample at each time frame, e.g., Figure 1C. Finally, sometimes modelers may want to plot the distribution of aggregate statistics of each sample (e.g., a histogram of sample means [14]).

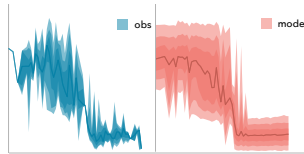
Visual_representation uses Grouping_samples to control how to group the model samples into the visual marks. VMC supports grouping samples using four options: *collapsing*, *individualizing*, *animating*, and *aggregating*. Let $\mathbf{q}^{(i)} = [q_1^{(i)}, \dots, q_n^{(i)}]$ be a sample from the model on a set of observed values of predictors, $\mathbf{x} = [x_1, \dots, x_n]$, where n is the number of instances. There are r samples total: $\mathbf{q}^{(1)}, \dots, \mathbf{q}^{(r)}$. Using *collapsing* will visualize all samples in one visual mark, i.e., the visual mark takes as input $\mathbf{q}^{(1)}, \dots, \mathbf{q}^{(r)}$. Using *individualizing* and *animating* are both specified to visualize samples in separate visual marks: *individualizing* overlays all marks in one coordinate system while *animating* shows each visual mark in one time frame. Thus, when using *individualizing* or *animating*, there are r visual marks with each of them takes input as $\mathbf{q}^{(i)}$. Using *aggregating* will aggregate statistics of each sample and then use one mark for the aggregated statistics. Denoting the aggregating function as $\pi: \mathbb{R}^n \rightarrow \mathbb{R}$, when using *aggregating*, the visual mark takes as input $\pi(\mathbf{q}^{(1)}), \dots, \pi(\mathbf{q}^{(r)})$.

4.4 Comparative layout

The final necessary decision in visualizing a model check is to specify a comparative layout to facilitate checking the alignment of data observations and model predictions. We summarise four classes of comparative layouts from the visual comparison literature [5, 19, 27, 40]: *superposition*, *juxtaposition*, *nested juxtaposition*, and *explicit encoding*. Our grammar makes these options entities of the same type (comparative layout), which is notably different from the grammar of graphics, where researchers have noted a mismatch between author’s comparison tasks and Grammar of Graphics components [43]. We represent the choices of comparative layouts directly in Comparative_layout, to better match model check visualization authors’ tasks. *Juxtaposition* and *superposition* do not require changes to the visual representations of model predictions and observed data, but organize the layout to make them spatially aligned [38].

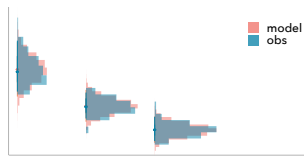
```
Visual_representation ←
  [(model, lineribbon,
    collapsing),
  (data, lineribbon)]
```

```
Comparative_layout ←
  juxtaposition
```



```
Visual_representation ←
  [(model, histogram,
    collapsing),
  (data, histogram)]
```

```
Comparative_layout ←
  superposition
```



Nested juxtaposition is a variant of juxtaposition, where the distributions of model predictions and observed data are placed side by side but within the coordinate system of the plot.

```
Visual_representation ←
```

```
[(model, pointinterval,
  collapsing),
  (data, pointinterval)]
```

```
Comparative_layout ←
```

```
nested_juxtaposition
```



To use *explicit encoding* requires calculating a transformation on the model predictions and observed data [19]. Explicit encoding is often used to facilitate checking specific assumptions of a model, such as heteroskedasticity, linearity, and normality of errors (Fig. 1I and J).

```
Visual_representation ←
```

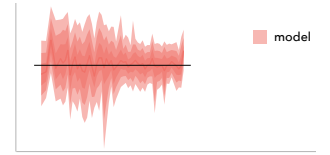
```
[(model, lineribbon,
  collapsing)]
```

```
Comparative_layout ←
```

```
Explicit_encoding
```

```
Explicit_encoding ←
```

```
residual
```



An effective comparative layout renders differences between model predictions and observed data visually salient. Which layout works best will depend on the specific circumstances. Superposition facilitates comparisons since the visual representations of model predictions and observed data are plotted along the same axes, but also introduces overlap that may result in loss of information on the part of the viewer. For example, when using a visual representation that encodes uncertainty information with opacity, a superposition layout will blend the opacity for model predictions and observed data. Juxtaposition avoids overlap but requires visuo-spatial working memory and eye movements to enable comparison. Nested juxtaposition is helpful for detecting local differences because it places the information being compared close in space on a common, aligned scale. It can avoid the shortcomings of superposition and juxtaposition, but sometimes requires more work to create. For example, creating a nested juxtaposition for a model check that is conditional on a continuous variable can be complicated; the author might need to cut the continuous axis into several segments and intermittently place the segments of model predictions and observed data. Explicit encoding presents the comparison directly, avoiding information loss, but also introduces abstraction by changing the unit. For example, Q-Q plots show the relationship between model predictions and observed data by plotting the quantiles of model predictions against the quantiles of observed data.

5 A WALK-THROUGH EXAMPLE: AIR QUALITY

We use the data from Shaddick et al. [46] and models from Gabry et al. [13] to demonstrate VMC’s usage in a statistical workflow. The focus is $PM_{2.5}$, representing exposure to air pollution from particulate matter measuring less than $2.5 \mu m$ in diameter, that is linked to a number of poor health outcomes. Direct measurements of $PM_{2.5}$ rely on a sparse network of ground monitors with heterogeneous spatial coverage. To obtain estimates of $PM_{2.5}$ concentration with high spatial resolution, direct measurements are supplemented with high resolution estimates from satellite data. By calibrating the satellite estimations using the ground monitor data, the goal is to obtain high-resolution, high-precision estimates of $PM_{2.5}$. We start with a simple linear regression:

$$\begin{aligned} \log(PM_{2.5}) &\sim Normal(\mu, \sigma) \\ \mu &= \alpha + \beta \log(\text{satellite}) \\ \log(\sigma) &\sim Normal(0, 1) \end{aligned} \quad (2)$$

After fitting the model, we use VMC to check the model’s predictions against the correlation between the observed $PM_{2.5}$ and satellite estimates. We start by checking the overall distribution of the model

predictions of $PM_{2.5}$ compared with the observed $PM_{2.5}$. We set the Quantity in `Sampling_spec` as `y`, which generates Figure 1A. With separate `Sampling_spec` and `Visual_representation` components in VMC, we can also flexibly check on different posterior distributions (e.g., Figure 1B) and use different visual representations of model distribution and observed distribution. This initial model check visualization provides a good starting point by revealing the misalignment between the shape of the distribution of predicted $PM_{2.5}$ and observed $PM_{2.5}$, where skewness in the observed data is not captured by the model.

Next we explore causes of this skewness to improve the initial model. We look at the marginal effects of the predictor to $PM_{2.5}$. VMC supports this step with various visual formats. First, we set `Mark` in `Visual_representation` as `point` for both the model distribution and observed distribution. We use `Group_samples` to animate the samples (Hypothetical Outcome plots [26]), which gives a view of what individual samples from the model distribution look like (Figure 1C). Using points reveals the original model predictions clearly but makes it hard to perceive the uncertainty directly, so we use a different `Mark`, `line + ribbons` (Figure 1D) to display the uncertainty intervals.

We continue to explore other visualizations by cycling through components of VMC, including trying different `Comparative_layouts` to compare the model predictions and observed data such as juxtaposition, and trying different faceting options to explicitly show how observed data is clustered (Figure 1E). When faceting by the regions, the marginal effects of $PM_{2.5}$ to satellite estimates have noticeable clusters in different regions (Figure 1E). In response, we update the model by adding submodels of regions.

$$\begin{aligned} \log(PM_{2.5})_r &\sim Normal(\mu_r, \sigma) \\ \mu_r &= \alpha_r + \beta_r \log(\text{satellite}) \\ \sigma &\sim Normal(0, 1) \end{aligned} \quad (3)$$

After fitting the new model, we check it, first by generating a canonical check used for multilevel models (Figure 1F). We also check how the submodels perform by specifying model checks that are conditional on the regions (Figure 1G and H). To do this, first, we set the `Mark` to encode uncertainty, e.g., `halfeye` for model distribution and dots for observed distribution in a raincloud plot (Figure 1G). Then, we adjust `Comparative_layout` to `nest_juxtaposition` to reduce visual overlap. Finally, to verify that this change improved the model, we return to our initial model check where we identified skewness (Figure 4A) and facet by regions this time. We validate that the density curves of the model predictions converge to the curve of the observed data well, but wonder how the overall density of all samples compared with the observed data. VMC’s design of `Group_sample` in `Visual_representation` supports flexible operations on model samples such as collapsing all the samples from the model prediction into one density curve (Figure 4C).

Next, we look more deeply into the assumptions of the model. We first check the residuals. By using VMC, we can do this quickly with `Explicit_encoding` in `Comparative_layout` component, where we specify a function encoding the comparison as the difference between model predictions and observed data (Figure 1I). Then we check the normality of the residuals by specifying a Q-Q plot (Figure 1J). Through VMC, we only need to change the encoding function set in `Explicit_encoding`. Looking at the residual plot, we validate that the variance of residuals is constant along the conditional variable. In the Q-Q plot, however, we observe further misalignment, where the residuals do not follow the standard normal distribution, motivating further changes to the model spec. The goal behind VMC is to make it easier for the analyst to focus on the modeling task at hand in such scenarios, with the grammar facilitating less tedious construction of checks.

6 IMPLEMENTATION IN R

We implement VMC in the R language as a package, provided in Supplemental Materials. The specification of VMC follows conventions popularized by the widely-used `ggplot2` package

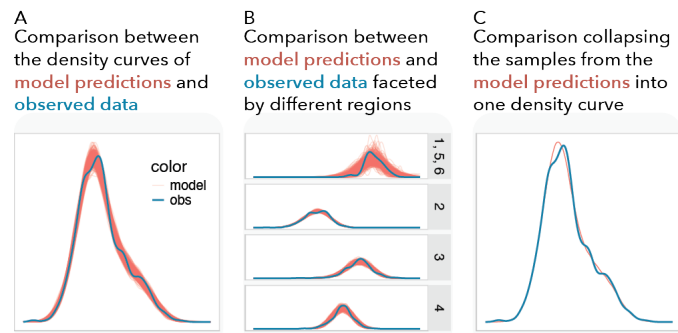


Fig. 4: The model check examples for updated model.

and outputs a `ggplot` specification to generate the model check plot. To start specifying a model check plot, the user issues a statement called `mcplot()` and then follows it with statements specifying the four requisite components of VMC: `mc_draw()` for `Sampling_spec`; `mc_observation_transformation()` for `Data_transformation`; `mc_model_*()` and `mc_obs_*()` for `Visual_representation` (where `*` stands for different mark types, e.g., `mc_model_slab()` for using `slab` for model predictions); and `mc_layout_*()` for `Comparative_layout` (where `*` stands for different layers, e.g., `mc_model_superposition()` for using `superposition`). The implementation of VMC also supports other supplementary components, e.g., `mc_condition_on()` to define the conditional variable used in the check and `mc_gglayer()` to append a `ggplot2` layer to the output specification.

7 REPRODUCING REAL-WORLD EXAMPLES

To validate the expressiveness of VMC as a visualization grammar, we collected and reproduced 27 model check visualization examples from statistics and visualization literature [11, 14, 15, 16, 24, 28, 53, 54] that cover a description of model check techniques from a popular Bayesian statistics textbook [16]¹. They highlight four model check techniques: *replicated or external checks*, to make predictions under the fitted data or future data, *choosing model quantities*, to check the aspect of the model we want, *marginal predictive checks*, to check the joint predictive distribution, and *residual checks*, to check the errors the model makes. We evaluate the syntax design of VMC by comparing it with `ggplot2` [50] and `bayesplot` [13] (a task-oriented R package for model check of Bayesian models) while reproducing these examples. We used an example Beta regression model in reproducing these model check visualizations². Across all the examples we implemented to compare APIs, there were an average of 30% fewer lines of code with VMC than with `ggplot2`.

Replicated or external checks. VMC supports replicated or external checks using the `Predictor_values` argument in the `Sampling_spec` component. Figure 1A shows a replicated check generated by VMC, which presents the kernel density estimate of the observed data (blue curve), with density estimates for samples from the posterior predictive distribution (orange, lighter lines). Figure 5A provides the full VMC specification for an external check where the user is creating a model check with a “raincloud” plot. The specifications of VMC are more concise than their `ggplot2` and `bayesplot` counterparts. With VMC, users need only specify one argument to input a new data set to predict on. In contrast, with `ggplot2` or `bayesplot`, since there is no grammar specification on how the data is generated from the model, users need to manually define the extract function using fitted or new data set and tidy the prediction’s data format.

Choosing model quantities. VMC’s `Sampling_spec` component enables various model quantities to be presented in model checks. Consider the previous example of the kernel density estimates check.

¹We reproduce these examples using the implementation of VMC in R.

²See the full specification of the example Beta regression model and the reproduced examples with their corresponding R code (using VMC, `ggplot2`, and `bayesplot` respectively) in Supplemental Materials

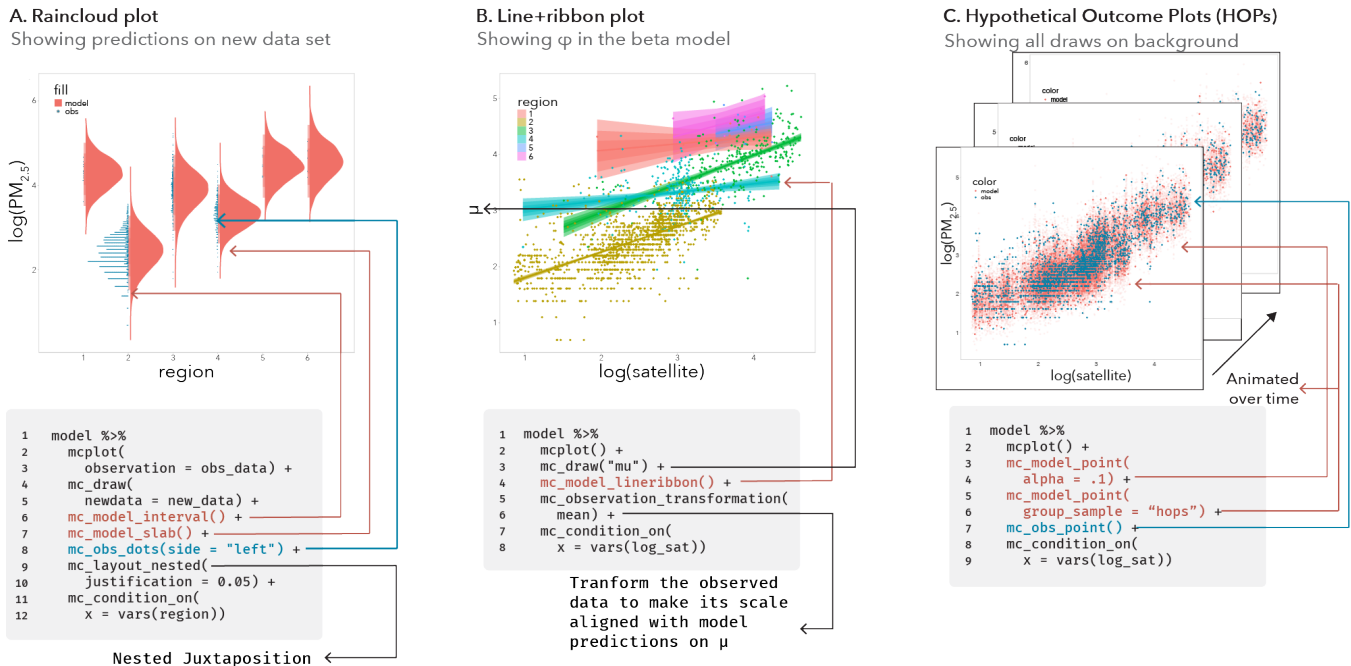


Fig. 5: (A) Raincloud plots are constructed by two visual representation layers for model predictions and one for the observed data. (B) A line + ribbon plot is a combination of a line showing the mean of predictions and ribbons showing the prediction intervals. (C) HOPs uses animation to combine the samples from the model.

We can specify the `Quantity` argument in `Sampling_spec` as μ (i.e., parameter μ in the beta regression model) (Figure 1B). Figure 5B shows a more concrete example with code in R. To align the scale of observed data with that of the model quantity μ , we apply a transformation on the observed data. The VMC specification for choosing the model quantity is, once again, more concise than the corresponding `ggplot2` and `bayesplot` example. When exploring the possible choices of model quantity, VMC allows users to consider the quantities as one entity, rather than a data processing step prior to plotting as when using `ggplot2` and `bayesplot`. This helps keep data processing and visualization consistent, potentially avoiding errors [43].

Marginal predictive checks VMC supports marginal predictive checks through flexible specification of visual representations and comparative layouts. For example, Figure 5C shows how VMC reproduces a canonical example of marginal checks introduced by Gelman [15] and combined with HOPs by Hullman and Gelman [24]. Besides conditioning on the independent variables on the x-axis, VMC also enables grouping marginal checks by color, rows and columns (i.e., faceting) Figure 1E and F. To check the marginal effects in the model, VMC requires a shorter *edit distance* to switch between conditional variables compared with `ggplot2`. In VMC specification, the conditional variables are specified by a component separated from the visual representation components, so to change the conditional variable only requires changing one component. This abstraction makes VMC closer to experts' language (O3), reducing the complexity compared to `ggplot2`.

Residual checks VMC can specify various residual checks with various tasks. For example, Figure 1I checks the heteroskedasticity by showing the residual values with a scatter plot and Figure 1J checks the normality of residuals with a Q-Q plot. Residual checks can be used to check other assumptions of the model, e.g., for a linear relationship between residuals and values of the response variable (see the example from Gelman [15] in Supplementary materials). VMC specification separates the `Comparative_layout` from the other components, enabling users to focus on the data transformations required for the residual checks. In `ggplot2`, however, the users need to consider the changes in geometries and aesthetics to incorporate with the changes in the data transformations. While `bayesplot` supports residual checks directly, there is some overhead when switching between different model checks, as code cannot easily be reused across different model checks.

8 OBSERVATIONAL INTERVIEW STUDY OF VMC

To understand how data analysts specify visualizations and interact with VMC, we conducted an interview-based study of its use by three experienced modelers. We aimed to answer the following research questions:

RQ1: Does the grammar capture core aspects of experienced analysts' understanding and practices of model check visualization?

RQ2: What aspects of the grammar's construction do participants find most useful, and what challenges or trade-offs do they face in using it?

RQ3: Can the grammar help experienced modelers explore novel (to them) model check visualizations, and what features encourage this?

8.1 Participants

We recruited three experienced statistical modelers who were comfortable working with generative models via social media posts. All three participants had obtained Ph.D. degrees in quantitative disciplines, and regularly conducted statistical analysis as part of their research. Two of three participants (P1 and P2) had more than 4 years of professional experience with statistical modeling, while the other one (P3) had over 2 years. Two of three participants (P2 and P3) teach bachelor- and graduate-level statistics, including the practice of model checking. All three participants reported having used `ggplot2` and `bayesplot` to generate the model check visualizations. We compensated the participants with a \$60 Visa Gift card for a one hour session.

8.2 Study Design

Before the session, each participant filled out a background questionnaire. We asked them to bring a model from their current work to apply VMC to. The session started with the participant describing their typical graphical model checking practices, followed by describing the model they brought to the session, including data, modeling goals, and model fitting and selection process. Participants then completed a brief VMC tutorial where they were asked to examine and run example code in an R notebook. These examples included 19 model check visualizations specified using VMC on an example multilevel Gaussian model.

In the first part of the tutorial, the examples demonstrated variations supported by the components of VMC by reproducing canonical model check visualizations in literature, including the examples from the statistical textbook [14]. The second part demonstrated the expressiveness of

VMC by incorporating visualization techniques that have not been widely used in current model check practices. For example, we included examples of using VMC to display uncertainty information by animation (i.e., HOPs) and the nested juxtaposition comparative layout. Participants were asked to experiment with new combinations of these components as they went through the tutorial, and to think aloud while navigating these tutorial examples. This portion of the study lasted approximately 20 minutes. After the tutorial, participants were asked to create model check visualizations for the model they brought, for approximately 20 more minutes. As they used the grammar, we occasionally prompted them with questions about their perceptions of the grammar, including about whether the visualizations were useful to them and how they conceived of aspects of the grammar specification. Each session ended with an exit interview (20 minutes) that asked participants to reflect on their process of using VMC and how they see VMC interfacing with their typical workflow.³

8.3 Results

We summarize observations from participants' usage of VMC and the exit interviews.

8.3.1 RQ1: Alignment with Participants' Model Check Understanding and Practice

In stepping through examples we provided in the tutorial, participants commented on how they recognized visualizations they commonly used. For example, P1 stated that *"this [posterior predictive check] is usually the first thing I do..."* and *"this [Q-Q] plot is exact what I would do on my model"*, and P2 stated that *"this [line+ribbon plot] is very nice. I like this. This is something that I usually kind of do on my own."* All the participants noted that the options included in VMC largely capture visualization variations that would interest them in practice. For example, when going through the examples that checked distributional parameters in the model, P1 stated that `mc_draw` and `mc_observation_transformation` are both interfaces that capture his practices of checking on different aspects of the model – *"I like this. So I can change between all the parameters that the model estimates"*.

Moreover, two of the three participants (P1 and P2) found that VMC also included some model check visualizations that they had never thought about before. P1 remarked that the raincloud plot (Figure 5A) displaying the uncertainty information in model predictions and observed data was new to him, saying *"I like this [violin plot]. I've never done anything like this before."* The participants were also impressed by the animation over samples supported by the grouping sampling argument. P1 said *"this [animation] is really cool. Initially I can't see the [model] draws within the Q-Q plot but now it's over the animation"*.

The `Comparative_layout` component in particular drew interest, leading to comments about how combining different layouts and visual representation could produce additional effective examples. For example, P1 found that juxtaposition can avoid visual clutter in some cases. He said *"I think I like this one [juxtaposition] better [than superposition], because it was pretty hard to see when the observed data was right on top of the the model [predictions]"*. When P2 saw our examples using nested juxtaposition, he was pleasantly surprised by its alternative to superposition—*"this [nested juxtaposition] is great. It separates those bars [confidence intervals] from one another, but doesn't overlay them together."*

Participants' perspectives on the direct comparison VMC encouraged between the model and the observed data in visualizations at times varied from that assumed in developing the grammar. For example, when adopting VMC to their own model, P2 created a model check visualization where the model quantity and the quantity in the observed data were not perfectly aligned. The model samples were generated from the posterior predictive distribution and the observed data were transformed using the mean function. Because we defined VMC such that quantities in model samples and observed data should be on the same scale, this use falls outside the scope of our correctness objective (O2). These observations were useful for identifying how such usage

can potentially be useful in scenarios like hypothesis testing on certain statistics. On the other hand, if comparability is not strictly ensured, additional actions on the visual front are necessary. For instance, when transforming the observed data by standard deviation function, specific visual techniques should be used to clarify that the observed data is now represented on the variance scale.

8.3.2 RQ2: Usability of the Grammar

While going through the tutorial, participants commented on how they reasoned about VMC specifications. Two of the three participants (P1 and P3) commented on how VMC's specification of the data preparation step was a good abstraction of how they think and express data targets in model check visualizations. P3 compared his experience with `bayesplot` to VMC, saying *"[In VMC,] I only need to specify mu [for the model quantity] and mean [for the data transformation]. It is really good compared to that [bayesplot]"*. P1 also found the naming convention and modular structure resembling that of `ggplot2` of VMC familiar to him, which made it intuitive to him – *"I had a pretty good idea of what it was gonna do... [because] it's modular, and the names make sense. So it's pretty intuitive."*

All the participants appreciated that VMC allows them to specify model check visualizations in a high-level and systematic way. P3 mentioned that high-level interfaces for the observed data and the model make it easy to think of model check visualizations. Referring to the specification, he said *"There is the data versus model. I do like the contrasting. I teach a Bayesian course in university and I can totally see myself like, teaching students how to use this. I like that. It's relatively easy to transform things or extract things."* That VMC is built on a succinct specification with a few conceptual components was deemed effective by multiple participants: *"[The part I like about the grammar is that] it is built on these four or five foundational functions that built these plots"* (P2).

However, participants also noted some negatives. One of the biggest concerns was that despite how VMC's design separates components, they often remain entangled in some ways. For example, when specifying the mark type as `line+ribbon` in `Visual_representation`, there must be a conditional variable on the x-axis. In trying to run a VMC specification component by component, P3 commented on this dependency between visual representation and conditional variable: *"The grammar is like [independent] layers added up. But it's not really [independent] layers. The visual representation layer can not work independently without the conditional variable layer."*

8.3.3 RQ3: Exploration

Participants used different approaches in generating the new visualizations by applying VMC to the models they brought. Each participant adapted five specifications on average in the time they had. P1 started by copying all the examples in the tutorial and changing the model input to his model. After he had reviewed all examples, he chose to explore the argument that he had not yet encountered: `grouping_sample` in the `Visual_representation` component. He tried combining different mark types with HOPs. P2 sought better alternatives starting from specific examples from the tutorial, trying to reason about what should be changed in applying them to his model. For example, when looking at the raincloud example Figure 5A, he said *"I guess one thing that...It would be easier to encourage people to think about data if these were histograms instead of dots"*. He changed the `mc_obs_dots` to `mc_obs_histogram` to check that it was straightforward to change the mark. P3 followed a more top-down approach. He started by thinking of the general goals he had for the plots then tried to translate those concepts into the components of our grammar. He first thought that he wanted to check the σ parameter in the Gaussian distribution, so he specified `mc_draw("sigma")` and `mc_observation_transformation(sd)`. Then he tried to find a suitable visual representation for the model samples and the observed data. He went through points, lines and `line+ribbon` mark types, choosing the `line+ribbon` as the final representation.

Although all the participants succeeded to explore new visualizations using VMC, they did not always find the newly generated visualizations

³See Supplementary Materials for interview questions and the R document.

effective. For example, P2 experimented with different mark types on the observed data. He discovered that marks utilizing smoothing methods failed to accurately represent the original data instances, potentially leading to misinterpretations. The flexibility of VMC also exposed some ineffective edge cases such as mismatched data and visual representations. For example, during P3's exploration, he found that mark types for density estimates should not be applied to the observed data when it is aggregated into a single value such as the mean, although this case is not precluded by VMC. Nonetheless, that participants could quickly identify the effectiveness of the newly generated model check visualizations is a promising sign that VMC could lead modelers to identify new, useful designs.

9 DISCUSSION

We developed VMC to explore the potential of using a small set of sensible abstractions to make model check visualizations easier to reason about and generate. In its current iteration, VMC can be of immediate use in statistical workflows. In exploratory data analysis, VMC can be used to explore provisional models used to drive understanding of features and heterogeneity in the data. In a Bayesian workflow, VMC could be used to check the prior distribution before model fitting, and support comparisons of alternatives during fake data simulations [13]. In any statistical workflow, after the model is fitted, VMC can be used to check predictive distributions, apply test quantities, and check model assumptions like outliers, distributional assumptions and heteroskedasticity.

While the development of VMC was inspired by Bayesian workflow where graphics have been more widely accepted as a crucial part of modeling workflow, one of our aspirations in developing VMC is that such tools may be helpful in popularizing graphical model checks in a wider range of statistical workflows as well as statistics education. For example, standard materials for learning regression often focus on only a handful of canonical visualizations (residual plots, QQ plots). By formulating visual model checks as a more extensible design space, tools like VMC may contribute to more effective modeling in practice by calling greater attention to the many subtle ways in which model predictions may align with or deviate from observed data.

Beyond its immediate practical value, developing and evaluating VMC led to several observations about graphical model checking as a tool in statistical workflow.

First, our work highlights the **need to account for both the visual judgment mechanisms and data alignment required to facilitate comparisons of relevant quantities**. Visualizations intended for comparison should not only facilitate visual judgments through effective comparative layouts, they should also consider how to ensure comparability of quantities through data alignment. For instance, drawing meaningful conclusions by comparing the intercept parameter from a linear regression model with the observed outcome values is unlikely, regardless of whether they are superposed or juxtaposed. While our constraints in developing VMC precluded some useful examples (e.g., when the observed data is simulated and the intercept used for simulation known), encouraging authors to think carefully about alignment is likely useful. There may be some value in type systems for data scales, similar to formal guarantees in typed programming languages, that can help people align the quantities effectively and avoid comparing them on scales that do not match.

Second, a tension we encountered in developing VMC points to **the value of extending API support to non-checkable model quantities to support important steps in a statistical workflow**, such as setting and interpreting priors or displaying fitted parameter values. According to our definition, a *checkable* model quantity must have a transformation function that can align the observed data with it on a common scale. However, for *non-checkable model quantities*, no such transformation function exists, and we must turn to other design strategies. When the goal is to make model checking accessible to a broader set of modelers beyond experienced statisticians, then it would be useful for future work to identify systematic ways of contextualizing abstract scales like parameter spaces. Doing so could make visualizations of non-checkable quantities more concrete, similar to how a model check that requires checking against observed data makes model quantities easier

to understand. This could be achieved, e.g., through analogies to known phenomena [32] or reference markings illustrating the relationship between the abstract scales, like the explode-y graphics in [55].

Third, widely applicable model checking visualization tools call for **appropriate software abstractions to overcome heterogeneity in model outputs, syntax, and checkable quantities**, so that model check visualization tools like VMC can be widely applied. In the implementation of VMC, we aimed to support different model objects and checkable quantities by combining different model extracting libraries, e.g., *tidybayes* [29] and *insight* [36]. Other model inspection R packages like *marginalEffects* [3] and *easystats* [35] take steps to write a layer of abstraction that accommodates all of these variations without requiring intervention by the user, increasing the interoperability of tools for modelers within the R ecosystem. Such efforts are a natural next step for extending VMC, though complete interoperability may not be possible as a result of the continually changing landscape of modeling libraries.

Our approach in VMC is also compatible with model comparison tools that statisticians often use, such as leave-one-out (LOO) cross-validation and multiverse analysis, but these applications require extensions that accommodate multiple model objects. For example, users can extend *mcpPlot* to take a list of models as input. As long as each model in this list has the same model quantities, it is straightforward to apply the visual representations and comparative layout in VMC to them. It is also worth extending operations to the predictions of these models after the *sample* stage in the compiler to support more complex abstractions in the statistical tools, e.g., model comparison in LOO cross-validation.

Along the lines of a proposal for more explicit support for model checking in visual analysis tools [24], VMC can be incorporated in interactive analysis tools to support the generation of model checks in exploratory data analysis, similar to the design approach by Kale et al. in EVM [28]. Adapting VMC for this purpose would allow the developer to avoid the diffuseness and viscosity of the notation in lower-level visualization grammars and focus squarely on the main substantive components of a model check.

Better tools for graphical model checking can also improve the development of predictive models through better designed VIS4ML systems, where principled processes are still in development. Components of VMC can be taken as a foundation for the development of such tools, and extended to support high dimensional datapoints like images. Understanding what makes a good model check with (parametric) statistical models may not be equivalent to what make a good model check with machine learning models, motivating empirical work.

9.1 Limitations

While we propose four necessary design objectives of VMC based on the practice and theory of model checking in the statistics and visualization literature, we cannot guarantee that these are the sufficient for a good model check visualization grammar. For example, a model check grammar could benefit by facilitating the specification of common interaction techniques in visualizations (e.g., brushing and linking) such that users can quickly find patterns that reflect important types of misfit [28]. Other missing design objectives may include expressing sampling primitives and facilitating model comparison workflows. Future work is needed to adjust the design objectives based on the target scenario.

VMC specifications are also limited to position encodings (i.e., x-axis, y-axis, rows, and columns), which rules out visualization techniques that display data without axes, e.g., maps or some space-filling techniques. Allowing display without standard axis encodings may require another abstract layer in VMC to map the data onto the display space, e.g., mapping a country variable to a spatial region on the map.

ACKNOWLEDGMENTS

We thank our anonymous reviewers for their helpful suggestions. Jessica Hullman thanks NSF #2211939 for supporting this work.

REFERENCES

- [1] M. Allen, D. Poggiali, K. Whitaker, T. R. Marshall, and R. A. Kievit. Raincloud plots: a multi-platform tool for robust data visualization. *Wellcome open research*, 4, 2019. 4
- [2] J. Ameijeiras-Alonso, R. M. Crujeiras, and A. Rodríguez-Casal. Mode testing, critical bandwidth and excess mass. *Test*, 28:900–919, 2019. 2
- [3] V. Arel-Bundock. *marginaleffects: Predictions, Comparisons, Slopes, Marginal Means, and Hypothesis Tests*, 2023. R package version 0.15.0.9000. 9
- [4] H. Baniecki, D. Parzych, and P. Biecek. The grammar of interactive explanatory model analysis. *Data Mining and Knowledge Discovery*, pp. 1–37, 2023. 2
- [5] R. A. Becker, W. S. Cleveland, and M.-J. Shyu. The visual design and control of trellis display. *Journal of computational and Graphical Statistics*, 5(2):123–155, 1996. 2, 5
- [6] A. F. Blackwell, C. Britton, A. Cox, T. R. Green, C. Gurr, G. Kadoda, M. S. Kutar, M. Loomes, C. L. Nehaniv, M. Petre, et al. Cognitive dimensions of notations: Design tools for cognitive technology. In *Cognitive Technology: Instruments of Mind: 4th International Conference, CT 2001 Coventry, UK, August 6–9, 2001 Proceedings*, pp. 325–341. Springer, 2001. 2
- [7] D. M. Blei. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1:203–232, 2014. 2
- [8] G. E. Box. Sampling and bayes’ inference in scientific modelling and robustness. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 143(4):383–404, 1980. 2
- [9] A. Buja, D. Cook, H. Hofmann, M. Lawrence, E.-K. Lee, D. F. Swayne, and H. Wickham. Statistical inference for exploratory data analysis and model diagnostics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4361–4383, 2009. 2
- [10] D. Cox. Some remarks on the role in statistics of graphical methods. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 27(1):4–9, 1978. 3
- [11] M. Fernandes, L. Walls, S. Munson, J. Hullman, and M. Kay. Uncertainty displays using quantile dotplots or cdfs improve transit decision-making. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pp. 1–12, 2018. 4, 6
- [12] J. Gabry and T. Mahr. bayesplot: Plotting for bayesian models, 2024. R package version 1.11.0. 2
- [13] J. Gabry, D. Simpson, A. Vehtari, M. Betancourt, and A. Gelman. Visualization in bayesian workflow. *arXiv preprint arXiv:1709.01449*, 2017. 1, 2, 3, 5, 6, 9
- [14] A. Gelman. A bayesian formulation of exploratory data analysis and goodness-of-fit testing. *International Statistical Review*, 71(2):369–382, 2003. 2, 3, 4, 5, 6, 7
- [15] A. Gelman. Exploratory data analysis for complex models. *Journal of Computational and Graphical Statistics*, 13(4):755–779, 2004. 1, 2, 6, 7
- [16] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995. 1, 2, 6
- [17] A. Gelman, X.-L. Meng, and H. Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pp. 733–760, 1996. 3
- [18] A. Gelman, A. Vehtari, D. Simpson, C. C. Margossian, B. Carpenter, Y. Yao, L. Kennedy, J. Gabry, P.-C. Bürkner, and M. Modrák. Bayesian workflow. *arXiv preprint arXiv:2011.01808*, 2020. 2
- [19] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011. 2, 5
- [20] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430, 2005. 3
- [21] H. Hofmann, L. Follett, M. Majumder, and D. Cook. Graphical tests for power comparison of competing designs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2441–2448, 2012. 2
- [22] J. Hullman. Why authors don’t visualize uncertainty. *IEEE transactions on visualization and computer graphics*, 26(1):130–139, 2019. 1
- [23] J. Hullman and A. Gelman. Challenges in Incorporating Exploratory Data Analysis Into Statistical Workflow. *Harvard Data Science Review*, 3(3), jul 30 2021. <https://hdsr.mitpress.mit.edu/pub/2ym7zm34>. 2
- [24] J. Hullman and A. Gelman. Designing for Interactive Exploratory Data Analysis Requires Theories of Graphical Inference. *Harvard Data Science Review*, 3(3), jul 30 2021. <https://hdsr.mitpress.mit.edu/pub/w075glo6>. 2, 6, 7, 9
- [25] J. Hullman, M. Kay, Y.-S. Kim, and S. Shrestha. Imagining replications: Graphical prediction & discrete visualizations improve recall & estimation of effect uncertainty. *IEEE transactions on visualization and computer graphics*, 24(1):446–456, 2017. 4
- [26] J. Hullman, P. Resnick, and E. Adar. Hypothetical outcome plots outperform error bars and violin plots for inferences about reliability of variable ordering. *PLoS one*, 10(11):e0142444, 2015. 4, 5, 6
- [27] N. Jardine, B. D. Ondov, N. Elmqvist, and S. Franconeri. The perceptual proxies of visual comparison. *IEEE transactions on visualization and computer graphics*, 26(1):1012–1021, 2019. 5
- [28] A. Kale, Z. Guo, X. L. Qiao, J. Heer, and J. Hullman. Evm: Incorporating model checking into exploratory visual analysis. *arXiv preprint arXiv:2308.13024*, 2023. 2, 6, 9
- [29] M. Kay. *tidybayes: Tidy Data and Geoms for Bayesian Models*. R package version 3.0.5. doi: 10.5281/zenodo.1308151 9
- [30] M. Kay. ggdist: Visualizations of distributions and uncertainty in the grammar of graphics. 2023. 2, 4
- [31] M. Kay, T. Kola, J. R. Hullman, and S. A. Munson. When (ish) is my bus? user-centered visualizations of uncertainty in everyday, mobile predictive systems. In *Proceedings of the 2016 chi conference on human factors in computing systems*, pp. 5092–5103, 2016. 4
- [32] Y.-S. Kim, K. Reinecke, and J. Hullman. Explaining the gap: Visualizing one’s predictions improves recall and comprehension of data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1375–1386, 2017. 9
- [33] J. Li and J. H. Huggins. Calibrated model criticism using split predictive checks. *arXiv preprint arXiv:2203.15897*, 2022. 2
- [34] D. Lüdecke, M. S. Ben-Shachar, I. Patil, P. Waggoner, and D. Makowski. performance: An R package for assessment, comparison and testing of statistical models. *Journal of Open Source Software*, 6(60):3139, 2021. doi: 10.21105/joss.03139 1, 2

- [35] D. Lüdecke, M. S. Ben-Shachar, I. Patil, B. M. Wiernik, E. Bacher, R. Thériault, and D. Makowski. easystats: Framework for easy statistical modeling, visualization, and reporting. *CRAN*, 2022. R package. 9
- [36] D. Lüdecke, P. Waggoner, and D. Makowski. insight: A unified interface to access information from model objects in R. *Journal of Open Source Software*, 4(38):1412, 2019. doi: 10.21105/joss.01412 3, 9
- [37] M. Majumder, H. Hofmann, and D. Cook. Validation of visual statistical inference, applied to linear models. *Journal of the American Statistical Association*, 108(503):942–956, 2013. 2
- [38] B. J. Matlen, D. Gentner, and S. L. Franconeri. Spatial alignment facilitates visual comparison. *Journal of Experimental Psychology: Human Perception and Performance*, 46(5):443, 2020. 5
- [39] M. Muller and A. Strohmayer. Forgetting practices in the data sciences. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2022. 2
- [40] B. Ondov, N. Jardine, N. Elmqvist, and S. Franconeri. Face to face: Evaluating visual comparison. *IEEE transactions on visualization and computer graphics*, 25(1):861–871, 2018. 5
- [41] L. Padilla, M. Kay, and J. Hullman. Uncertainty visualization. 2020. 4
- [42] X. Pu and M. Kay. A probabilistic grammar of graphics. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2020. 2, 3, 4
- [43] X. Pu and M. Kay. How data analysts use a visualization grammar in practice. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–22, 2023. 3, 5, 7
- [44] T. P. Ryan. *Modern regression methods*, vol. 655. John Wiley & Sons, 2008. 3
- [45] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics*, 23(1):341–350, 2016. 2
- [46] G. Shaddick, M. L. Thomas, A. Green, M. Brauer, A. Donkelaar, R. Burnett, H. H. Chang, A. Cohen, R. V. Dingenen, C. Dora, S. Gumy, Y. Liu, R. Martin, L. A. Waller, J. West, J. V. Zidek, and A. Prüss-Ustün. Data Integration Model for Air Quality: A Hierarchical Approach to the Global Estimation of Exposures to Ambient Air Pollution. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 67(1):231–253, 06 2017. doi: 10.1111/rssc.12227 1, 5
- [47] H. Subramonyam and J. Hullman. Are we closing the loop yet? gaps in the generalizability of vis4ml research. *arXiv preprint arXiv:2308.06290*, 2023. 2
- [48] J. W. Tukey et al. *Exploratory data analysis*, vol. 2. Reading, MA, 1977. 2
- [49] P. F. Velleman and R. E. Welsch. Efficient computing of regression diagnostics. *The American Statistician*, 35(4):234–242, 1981. 3
- [50] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. 2, 6
- [51] H. Wickham, D. Cook, H. Hofmann, and A. Buja. Graphical inference for infovis. *IEEE transactions on visualization and computer graphics*, 16(6):973–979, 2010. 2
- [52] L. Wilkinson. *The grammar of graphics*. Springer, 2012. 2
- [53] F. Yang, M. Cai, C. Mortenson, H. Fakhari, A. D. Lokmanoglu, J. Hullman, S. Franconeri, N. Diakopoulos, E. Nisbet, and M. Kay. Swaying the public? impacts of election forecast visualizations on emotion, trust, and intention in the 2022 us midterms. 2023. 6
- [54] F. Yang, L. T. Harrison, R. A. Rensink, S. L. Franconeri, and R. Chang. Correlation judgment and visualization features: A comparative study. *IEEE transactions on visualization and computer graphics*, 25(3):1474–1488, 2018. 6
- [55] F. Yang, M. Hedayati, and M. Kay. Subjective probability correction for uncertainty representations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2023. 9