

**ENHANCING MODEL EVALUATION AND MODEL SELECTION DECISIONS USING PROPER SCORING RULES**

**Progress Report**

by Juan Jaya Jauanda

School of Information Technology and Electrical Engineering,

the University of Queensland.

Submitted for Master of Data Science Capstone 2

8th April 2025

**SUMMARY**

Accurately evaluating the performance of predictive models remains a fundamental challenge in data science, particularly when dealing with uncertainty-aware applications such as forecasting, healthcare, risk assessment, and environmental modeling. While traditional evaluation metrics like accuracy, RMSE, and R-squared offer quick assessments of model performance, they focus solely on point estimates and fail to account for the variability or uncertainty present in real-world predictions. This limitation can lead to overconfident decision-making and poor model selection, especially in high-stakes scenarios where understanding the spread of possible outcomes is essential. To address this challenge, this project introduces **Scoriverse**, an R package designed to provide a unified and automated framework for generating probabilistic predictions and evaluating models using proper scoring rules. Scoriverse integrates uncertainty quantification into the evaluation process by supporting outcome-scale sampling and posterior predictive draws, thereby enabling robust application of scoring metrics such as the Continuous Ranked Probability Score (CRPS), logarithmic score, Dawid–Sebastiani Score (DSS), interval score, and Brier score. The design of Scoriverse is structured around five core components: prediction standardization, uncertainty quantification through sampling-based methods, scoring rule implementation, diagnostic visualization tools, and modular wrapper functions for compatibility across various model types. The package supports a wide range of models, including generalized linear models (GLM), generalized additive models (GAM), Bayesian models via brmsfit, and tidymodels workflows, ensuring flexibility and extensibility for diverse modeling needs. Throughout the development process, emphasis was placed on robustness and reproducibility. This was achieved through systematic unit testing using the testthat framework, rigorous error handling, and outcome-scale sampling logic that allows probabilistic scoring across different model families. The scoring pipeline (run\_scoriverse()), combined with direct access to individual components, provides users with the flexibility to automate evaluations or customize workflows as needed. While the current implementation provides a strong foundation for model evaluation, several areas for future development remain, including the extension to survival analysis, time-series models, and additional scoring metrics such as calibration scores. The package architecture is intentionally designed to be modular and plugin-ready, facilitating future contributions from the research community and practitioners. This report documents the design rationale, implementation strategies, evaluation methods, and testing philosophy behind Scoriverse, offering both technical insight and practical guidance for users. With its unified approach to probabilistic prediction evaluation, Scoriverse aims to support more reliable, interpretable, and uncertainty-aware model assessment across a wide range of applications.

**Table of Contents**

[1.1 Description of the Problem 6](#_Toc196914417)

[1.1.1 Inconsistent Prediction Formats Across Models 7](#_Toc196914418)

[1.1.4 Difficulty in Comparing Models Across Families 8](#_Toc196914419)

[1.2 Project Aims 8](#_Toc196914420)

[1.3 Structure of the Report 9](#_Toc196914421)

[2. Background and Related Work 10](#_Toc196914422)

[2.1 The Importance of Prediction Standardization in Modeling 10](#_Toc196914423)

[2.2 Understanding Uncertainty in Model Predictions 11](#_Toc196914424)

[2.3 Review of Related R Packages and Tools 12](#_Toc196914425)

[2.3.1 marginaleffects 12](#_Toc196914426)

[2.3.2 scoringRules 13](#_Toc196914427)

[2.3.3 stats (Base R) 13](#_Toc196914428)

[2.3.4 ggplot2 13](#_Toc196914429)

[2.4 Identified Gaps and Motivation for Scoriverse 13](#_Toc196914430)

[Chapter 3: Design and Methodology 15](#_Toc196914431)

[3.1 Overview of the Package Architecture 15](#_Toc196914432)

[3.2 Supported Model Classes and Prediction Strategies 16](#_Toc196914433)

[3.3 Outcome-Scale Sampling Design 17](#_Toc196914434)

[3.4 Scoring Metric Implementations 17](#_Toc196914435)

[3.5 Prediction Extraction Logic 17](#_Toc196914436)

[3.6 Wrapper Functions for Model Compatibility 18](#_Toc196914437)

[3.7 Visualization Tools for Diagnostics and Model Evaluation 19](#_Toc196914438)

[Chapter 4: Practical Usage 22](#_Toc196914439)

[4.1 Overview of the Scoriverse Evaluation Workflow 22](#_Toc196914440)

[Core Steps Handled by run\_scoriverse(): 22](#_Toc196914441)

[4.2 Example Workflow: GLM with Poisson Family 23](#_Toc196914442)

[Fitting and Evaluating the Model: 23](#_Toc196914443)

[4.3 Example Workflow: Bayesian Model with brmsfit 23](#_Toc196914444)

[4.4 Example Workflow: GAM with Poisson Family 25](#_Toc196914445)

[4.5 Manual Sampling with extract\_predictions() 26](#_Toc196914446)

[4.6 Parameter Extraction for Scoring: extract\_additional\_params() 27](#_Toc196914447)

[4.7 Direct Scoring Function Usage 28](#_Toc196914448)

[4.8 Direct Use of Visualization Tools 29](#_Toc196914449)

[4.9 Error Handling and Model Compatibility 30](#_Toc196914450)

[4.10 Multi-Model Comparison Workflow (Advanced Use Case) 33](#_Toc196914451)

[4.11 Interpretation of Scores and Diagnostics 33](#_Toc196914452)

[Metric 34](#_Toc196914453)

[Interpretation 34](#_Toc196914454)

[CRPS 34](#_Toc196914455)

[Lower values indicate sharper, better-calibrated predictions. 34](#_Toc196914456)

[Logarithmic Score 34](#_Toc196914457)

[Penalizes overconfident or underconfident predictions. 34](#_Toc196914458)

[Brier Score 34](#_Toc196914459)

[Lower values indicate better probabilistic classification accuracy 34](#_Toc196914460)

[Interval Score 34](#_Toc196914461)

[Balances interval width against coverage performance. 34](#_Toc196914462)

[Dawid–Sebastiani Score 34](#_Toc196914463)

[Scale-invariant, evaluating variance and bias. 34](#_Toc196914464)

[Visual Insights: 34](#_Toc196914465)

[Chapter 5: Evaluation and Testing Strategies 35](#_Toc196914466)

[5.1 Testing Philosophy and Definition of Robustness 35](#_Toc196914467)

[5.2 Unit Testing and Coverage 36](#_Toc196914468)

[5.2.1 Parameter Extraction Tests 36](#_Toc196914469)

[5.2.2 Prediction Extraction Tests 36](#_Toc196914470)

[5.2.3 Scoring Rule Tests 37](#_Toc196914471)

[5.2.4 Wrapper Function Tests and Error Handling 37](#_Toc196914472)

[5.3 Error Handling and Validation Scenarios 38](#_Toc196914473)

[Chapter 6: Future Work and Community Engagement 40](#_Toc196914474)

[6.1 Roadmap for Future Development 40](#_Toc196914475)

[6.1.1 Expansion of Supported Model Classes 40](#_Toc196914476)

[6.1.2 Additional Scoring Rules and Evaluation Metrics 40](#_Toc196914477)

[6.1.3 Performance Optimization and Parallelization 41](#_Toc196914478)

[6.1.4 Integration with Workflow Automation Tools 41](#_Toc196914479)

[6.2 Limitations of Current Implementation 41](#_Toc196914480)

[6.3 Strategies for Community Engagement 42](#_Toc196914481)

[6.3.1 Open Source Development and Collaboration 42](#_Toc196914482)

[6.3.2 Documentation and Educational Resources 42](#_Toc196914483)

[6.3.3 Academic Collaboration and Publication 42](#_Toc196914484)

[Chapter 7: Reflections and Impact 44](#_Toc196914485)

[7.1 Summary of Contributions 44](#_Toc196914486)

[7.1.1 Unified Prediction Interface 44](#_Toc196914487)

[7.1.2 Robust Uncertainty Quantification 44](#_Toc196914488)

[7.1.3 Comprehensive Scoring Rule Implementation 44](#_Toc196914489)

[7.1.4 Diagnostic Visualization Tools 45](#_Toc196914490)

[7.1.5 Rigorous Testing and Validation 45](#_Toc196914491)

[7.2 Broader Significance and Potential Impact 45](#_Toc196914492)

[7.3 Future Vision 46](#_Toc196914493)

[Reference 47](#_Toc196914494)

[Appendices 49](#_Toc196914495)

[Appendix A. Output Structure from run\_scoriverse() 49](#_Toc196914496)

[Appendix B. Example Diagnostic Plots 49](#_Toc196914497)

[B.1 Prediction vs. Observed Plot 49](#_Toc196914498)

[B.2 Residual Plot 50](#_Toc196914499)

[Appendix C. R Package Dependencies and Setup 51](#_Toc196914500)

[C.1 Installation Instructions: 51](#_Toc196914501)

[Appendix D. Example Testing Logs 51](#_Toc196914502)

[Appendix E. Selected Core Function Listings 52](#_Toc196914503)

[E.1 Scoring Pipeline Orchestration: run\_scoriverse() 52](#_Toc196914504)

[E.2 Prediction Wrapper: wrap\_predict() 53](#_Toc196914505)

[E.3 Scoring Wrapper: wrap\_scoring() 54](#_Toc196914506)

[Appendix F. Selected Test Code Examples 55](#_Toc196914507)

[F.1 Parameter Extraction Test 55](#_Toc196914508)

[F.2 Outcome-Scale Sampling Test 56](#_Toc196914509)

[F.3 Error Handling Test for Missing Arguments 56](#_Toc196914510)

[Appendix G. Reproducibility and Environment Configuration 57](#_Toc196914511)

**1. Introduction**

This project aims to develop an R package named **Scoriverse**, which provides a unified and automated framework for generating probabilistic predictions and evaluating predictive models using proper scoring rules. In recent years, the importance of accurate and robust model evaluation has grown significantly in fields such as forecasting, healthcare, risk analysis, and decision-making. While traditional evaluation methods like accuracy, root mean squared error (RMSE), and R-squared remain widely used, these metrics are limited to point estimate assessments and often fail to capture the uncertainty associated with predictions (Hastie et al., 2009).

In many real-world scenarios, understanding prediction uncertainty is as critical as the predictions themselves. A point estimate might suggest high accuracy, but without knowing the spread or confidence of the result, decisions made based on such predictions could lead to overconfidence or significant risk exposure (Kuhn et al., 2016). This becomes especially problematic in high-stakes environments where prediction errors could have financial, operational, or health-related consequences.

The Scoriverse package is introduced as a solution to this challenge by offering probabilistic scoring methods that evaluate not just how well a model predicts central tendencies but also how well it represents uncertainty. The package simplifies the complexity of probabilistic evaluation by standardizing prediction extraction and integrating proper scoring rules such as the Continuous Ranked Probability Score (CRPS), logarithmic score, Dawid–Sebastiani Score (DSS), interval score, and Brier score (Gneiting & Katzfuss, 2014).

In this chapter, Section 1.1 discusses the core challenges faced by traditional evaluation methods in the context of predictive modeling. Section 1.2 outlines the specific aims and objectives of this project. Finally, Section 1.3 explains the structure of this report, providing a roadmap for the reader through the subsequent chapters.

## **1.1 Description of the Problem**

Despite advancements in predictive modeling, several challenges persist that limit the effectiveness of model evaluation practices. In particular, these challenges become pronounced in settings where **uncertainty quantification** is essential for informed decision-making. This project identifies four key issues that motivate the development of the Scoriverse package.

### ****1.1.1 Inconsistent Prediction Formats Across Models****

One significant hurdle in model evaluation arises from the diverse range of prediction outputs generated by different modeling frameworks. Classical models such as generalized linear models (GLMs) and generalized additive models (GAMs) may output fitted values or standard errors on the link scale, while Bayesian models provide posterior predictive distributions directly through sampling. On the other hand, machine learning models like random forests and gradient boosting algorithms often produce only point estimates without associated uncertainty information.

This heterogeneity forces practitioners to manually adapt their evaluation pipelines for each model type, increasing the risk of errors, reducing scalability, and leading to fragmented workflows (Gundersen & Kjensmo, 2018).

**1.1.2 Limitations of Traditional Evaluation Metrics**

Metrics such as accuracy, RMSE, and R-squared remain the default options for model assessment across many disciplines. However, these metrics evaluate only the mean performance of predictions and ignore the uncertainty surrounding the predictions themselves. This limitation is particularly problematic in fields such as medicine, finance, and weather forecasting, where understanding the **distributional characteristics** of possible outcomes can significantly improve decision quality (Slingo & Palmer, 2011).

Without uncertainty-aware evaluation, models may be selected or deployed based on incomplete assessments, increasing the likelihood of poor real-world performance.

**1.1.3 Difficulty in Applying Probabilistic Scoring Rules**

Proper scoring rules such as CRPS, log score, DSS, and Brier score provide a mathematically principled way to evaluate the quality of probabilistic predictions (Gneiting & Katzfuss, 2014). However, applying these scoring methods typically requires: predictive samples or simulated draws, consistent parameter extraction (e.g., standard deviations, prediction intervals), and a careful alignment of prediction scale (link vs. response).

Manually implementing these steps is time-consuming, error-prone, and often inconsistent across projects. This challenge inhibits the broader adoption of probabilistic evaluation methods despite their recognized benefits.

### ****1.1.4 Difficulty in Comparing Models Across Families****

Comparing models from different families—such as Bayesian models, GLMs, GAMs, and machine learning algorithms—remains difficult because each model class may use different conventions for predictions and uncertainty. Without a **standardized interface for prediction extraction and scoring**, practitioners are left with ad hoc solutions, making model comparison unreliable and non-reproducible.

These challenges reduce the ability of researchers and analysts to make evidence-based decisions when choosing between models, especially when probabilistic outputs are required.

## **1.2 Project Aims**

The aim of this project is to develop **Scoriverse**, the package is designed to address the key challenges outlined above by providing the following functionalities:

* **Unified prediction interface** across supported model types, reducing the need for manual adjustments and ensuring consistent workflows.
* **Integrated uncertainty quantification**, supporting posterior predictive draws and outcome-scale sampling for models without native probabilistic outputs.
* **Proper scoring rule implementation**, offering direct access to metrics such as CRPS, log score, Brier score, interval score, and DSS.
* **Built-in diagnostic visualization tools** for model calibration assessment, prediction interval evaluation, and residual analysis.
* **Extensible modular architecture**, facilitating future expansion to additional model types, scoring metrics, and visualization strategies.

To achieve these objectives, the development of Scoriverse is structured into the following core component:

Table 1. Core Components of Scoriverse Package

|  |  |
| --- | --- |
| Component | Description |
| **Prediction Standardization** | Harmonized prediction extraction across model types, including GLMs, GAMs, brms models, and tidymodels workflows. |
| **Scoring and Evaluation Framework** | Implements proper scoring rules for probabilistic evaluation, with support for parameter extraction and sampling. |
| **Uncertainty Quantification via Sampling** | Supports posterior draws for Bayesian models and RNG-based sampling for GLMs and GAMs (e.g., Poisson, Gaussian, Negative Binomial). |
| **Diagnostic Visualization** | Provides uncertainty-aware plots including prediction interval charts and residual diagnostics. |
| **Testing and Robustness** | Applies unit testing, input validation, and reproducibility checks to ensure reliable operation. |

The ultimate goal is to enable users to perform **fair, reproducible, and uncertainty-aware model evaluations** across various modeling approaches with minimal manual configuration.

## **1.3 Structure of the Report**

This report is organized into seven chapters, detailing the development process, design methodology, evaluation strategies, and potential future directions for the Scoriverse package:

* **Chapter 1** introduces the project, explains the challenges in traditional model evaluation, and outlines the package's objectives.
* **Chapter 2** provides theoretical background on prediction standardization, uncertainty quantification, and related work in probabilistic evaluation.
* **Chapter 3** describes the architectural design of Scoriverse, explaining each core module and its role within the system.
* **Chapter 4** demonstrates practical usage scenarios, including the application of the scoring pipeline and direct access to individual components.
* **Chapter 5** discusses testing strategies, unit testing implementation, error handling, and robustness measures.
* **Chapter 6 o**utlines the roadmap for future work and community engagement plans, including potential model extensions and collaborative development strategies.
* **Chapter 7 r**eflects on the development process, summarizes the contributions of the package, and discusses its potential impact on research and applied data science.

This structure provides a comprehensive view of the project's motivations, methodologies, and outcomes, offering readers both technical insights and practical guidance for using and extending Scoriverse.

# **2. Background and Related Work**

Model evaluation forms a critical component of predictive analytics, providing the foundation for model selection, interpretation, and decision-making. While a wide range of tools exist for model fitting and prediction generation, there remains a notable lack of cohesive infrastructure for standardizing prediction outputs, quantifying uncertainty, and evaluating model performance across diverse modeling approaches (Jordan & Mitchell, 2015).

The Scoriverse package addresses this gap by introducing a unified evaluation framework that integrates prediction standardization, probabilistic scoring rules, uncertainty-aware sampling, and diagnostic visualization. This chapter outlines the theoretical background and related work underpinning the design of Scoriverse. Section 2.1 discusses the importance of prediction standardization in modeling workflows. Section 2.2 highlights the role of uncertainty quantification in model evaluation. Section 2.3 reviews existing tools within the R ecosystem and related fields. Section 2.4 summarizes the identified gaps that motivate the development of Scoriverse.

## **2.1 The Importance of Prediction Standardization in Modeling**

Predictions are often the primary output of statistical models and machine learning systems, serving as the basis for evaluation, interpretation, and decision-making. However, despite their central role, prediction outputs exhibit significant variation across model families in terms of format, scale (link vs. response), uncertainty representation, and metadata (Gneiting & Katzfuss, 2014).

Such inconsistencies introduce friction into downstream tasks like:

* Comparing predictions across models,
* Computing scoring metrics,
* Visualizing outputs with uncertainty,
* Integrating predictions into decision analysis pipelines.

For example:

* GLMs and GAMs often default to predictions on the link scale, requiring manual transformation to the response scale.
* Tree-based models like random forests may provide point estimates but lack uncertainty measures unless bootstrapping is applied.
* Bayesian models using brms natively produce posterior draws, which are essential for uncertainty quantification.

This lack of standardization forces users to write repetitive, fragile, and non-reusable code to handle each model type differently (Gundersen & Kjensmo, 2018). Scoriverse directly addresses this problem through wrapper functions such as wrap\_predict() and prepare\_model\_for\_prediction(), which provide a harmonized interface for extracting predictions and associated uncertainty, regardless of the underlying model.

The standardization approach taken by Scoriverse draws inspiration from the concept of **prediction grammar**—encapsulating model-specific complexities into a unified structure to enhance reproducibility, scalability, and interpretability (Machine Learning Pipelines: From Prototype to Production | Outerbounds, 2022).

## **2.2 Understanding Uncertainty in Model Predictions**

Uncertainty quantification is a cornerstone of reliable predictive modeling. While point estimates convey central tendencies, they often obscure the variability and confidence associated with predictions. This can mislead decision-makers by overstating the certainty of outcomes (Ghahramani, 2015).

Proper uncertainty representation is crucial for:

* **Risk-aware decision-making:** Accounting for a range of possible outcomes, not just the most likely one.
* **Robust model evaluation:** Enabling scoring rules like CRPS and log score that require full predictive distributions.
* **Model selection and ensemble construction:** Supporting fair comparison across models with different uncertainty characteristics.
* **Transparent scientific communication:** Providing interpretable results that honestly reflect uncertainty (Bhatt et al., 2020).

Table 2. Model-Specific Handling of Uncertainty

|  |  |
| --- | --- |
| Model Type | Uncertainty Handling |
| Bayesian Models (brms) | Native posterior predictive distributions via posterior\_predict() |
| Frequentist GLMs and GAMs | Standard errors on the link scale; sampling-based outcome uncertainty via RNGs like rpois(), rnorm() |
| Tree-Based Models (randomForest, xgboost) | No built-in uncertainty; bootstrapping or quantile regression forests required |
| Mixed Effects Models (lme4::lmer) | Standard errors for fixed and random effects; full uncertainty requires additional post-processing |
| Regularized Regression (glmnet) | Typically lacks direct uncertainty measures; external methods required for uncertainty estimation |

Given this fragmentation, automating uncertainty extraction and outcome-scale sampling remains challenging. Scoriverse mitigates this issue by integrating probabilistic sampling logic within its core functions, including extract\_predictions() and get\_posterior\_draws(), offering fallback methods where analytical uncertainty is unavailable.

The package implements consistent strategies for handling:

1. Posterior draws (for Bayesian models),
2. Outcome-scale sampling (for GLMs, GAMs),
3. Transformation between link and response scales,
4. Approximate uncertainty estimation for models without native probabilistic outputs.

These design choices enable Scoriverse to promote a grammar of uncertainty—a systematic approach to uncertainty representation that improves interoperability across models.

## **2.3 Review of Related R Packages and Tools**

The R ecosystem provides several packages that individually address prediction generation, uncertainty quantification, or scoring. However, these packages often operate in isolation, requiring manual coordination by the user. Below is an overview of key related tools and how Scoriverse enhances their functionality:

### ****2.3.1**** marginaleffects

Provides standardized prediction and marginal effects computation across models (Arel-Bundock et al., 2024). However, its primary focus remains on point estimates and link-scale outputs, often requiring additional steps to simulate observation-level predictions.

* **Scoriverse Integration:** Wraps marginaleffects::predictions() and automates post-processing to produce outcome-scale samples for scoring workflows.

### ****2.3.2**** scoringRules

Implements core probabilistic scoring metrics, including CRPS, DSS, and log score (Jordan et al., 2017). However, it assumes users supply predictive samples in a specific format and does not handle sampling itself.

* **Scoriverse Integration:** Automates sampling, parameter extraction, and formatting to ensure compatibility with scoringRules functions.

### ****2.3.3**** stats ****(Base R)****

Provides basic prediction capabilities via predict() for models like lm, glm, and others (Venables & Ripley, 2013). However, it lacks support for sampling-based uncertainty and proper scoring rules.

* **Scoriverse Enhancement:** Encapsulates predict() within wrap\_predict() and standardizes output handling, including uncertainty estimation.

### ****2.3.4**** ggplot2

Widely used for data visualization (Wickham, 2016). Supports plotting predictions and residuals but requires custom code for uncertainty visualizations like prediction intervals.

* **Scoriverse Extension:** Offers ready-to-use functions such as visualize\_predictions() and visualize\_residuals() for uncertainty-aware plots with minimal user input.

## **2.4 Identified Gaps and Motivation for Scoriverse**

Despite the strengths of existing tools, several gaps persist, down below is the challenges addressed by the package.

Table 3. Challenges addressed by Scoriverse

|  |  |  |
| --- | --- | --- |
| Challenge | Limitation in Existing Tools | Addressed by Scoriverse |
| Prediction standardization | Fragmented methods across model families | Unified via wrap\_predict() and extract\_predictions() |
| Uncertainty extraction and sampling | Manual or unsupported for many models | Automated outcome-scale sampling across GLMs, GAMs, brms |
| |  | | --- | | Application of scoring rules |  |  | | --- | |  | | Requires manual preparation of predictive samples | Integrated scoring pipeline via wrap\_scoring() and run\_scoriverse() |
| Visualization of uncertainty | Requires verbose custom code | Built-in diagnostic plots for predictions and residuals |
| Workflow reproducibility and scalability | Limited integration between prediction, scoring, and plotting | Cohesive, modular design with reproducible pipelines |

Scoriverse was developed to address these gaps by providing a **comprehensive, modular, and extensible solution** that harmonizes prediction generation, uncertainty quantification, and model evaluation. Its design philosophy emphasizes reproducibility, scalability, and user accessibility, supporting a wide range of modeling workflows without sacrificing rigor.

# **Chapter 3: Design and Methodology**

This chapter presents the architectural design and methodological approach underpinning the development of the Scoriverse R package. The package is designed with a focus on **modularity**, **extensibility**, and **scalability**, ensuring it can handle diverse modeling workflows while remaining accessible to both applied practitioners and researchers. The chapter explains the structural organization of the package, the prediction extraction strategy, outcome-scale sampling methodology, scoring rule implementations, and diagnostic visualization tools.

## **3.1 Overview of the Package Architecture**

Scoriverse is structured around a set of well-defined modules, each corresponding to a key stage in the model evaluation pipeline. This modular architecture enhances maintainability and encourages separation of concerns, where each component handles a distinct responsibility.

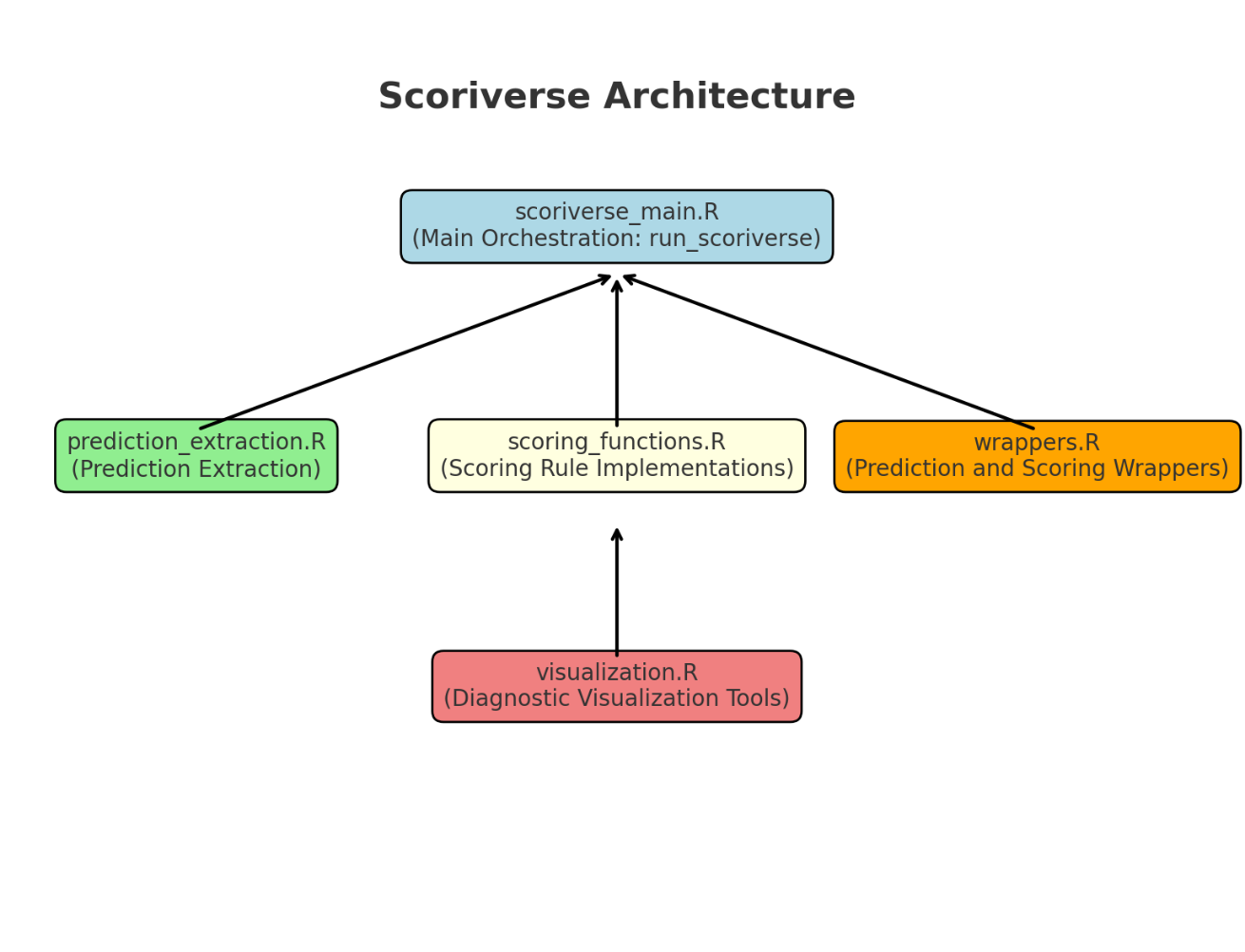


Figure 1: Scoriverse Package Architecture Diagram

This diagram illustrates the modular architecture of the Scoriverse package. The package is orchestrated by scoriverse\_main.R, which coordinates function calls across the following core modules:

* scoriverse\_main.R**:** Contains the orchestration logic via the run\_scoriverse() function. This serves as the main entry point for users, coordinating prediction extraction, scoring rule application, and optional visualization.
* prediction\_extraction.R**:** Implements logic for extracting predictions across supported model classes. It handles both point estimates and outcome-scale sampling through the function extract\_predictions().
* scoring\_functions.R**:** Provides robust implementations of proper scoring rules, including CRPS, logarithmic score, Brier score, interval score, and Dawid–Sebastiani Score (DSS). These functions assess model performance based on the distributional characteristics of predictions.
* wrappers.R**:** Offers standardized wrapper functions (wrap\_predict(), wrap\_scoring(), prepare\_model\_for\_prediction()) that ensure compatibility across diverse modeling frameworks. It also contains error handling and model-specific parameter extraction (extract\_additional\_params()).
* visualization.R**:** Supports diagnostic visualizations such as prediction vs. observed plots and residual analysis, enhancing interpretability and facilitating model diagnostics.

The architecture encourages **plug-and-play flexibility**, allowing users to interact with the package through high-level functions while retaining the option to customize individual stages of the workflow.

## **3.2 Supported Model Classes and Prediction Strategies**

Scoriverse accommodates a broad range of statistical and machine learning models, including:

* **Generalized Linear Models (GLM)**: Handles models like lm() and glm() with Gaussian, Poisson, and Negative Binomial distributions.
* **Generalized Additive Models (GAM)**: Supports mgcv::gam() for models with smooth terms and additive structures.
* **Bayesian Models (**brmsfit**)**: Fully leverages the posterior predictive framework via posterior\_predict().
* **Tidymodels Workflows**: Integrates seamlessly with models built using the tidymodels ecosystem, including parsnip models and workflows.
* **Tree-Based and Machine Learning Models**: Includes support for randomForest, gbm, xgboost, and glmnet, with fallback prediction mechanisms for models that lack probabilistic outputs.

The **prediction extraction strategy** centers around the wrap\_predict() and extract\_predictions() functions, which detect the model class and apply the appropriate prediction logic. If uncertainty quantification is available (e.g., via standard errors or posterior samples), the package ensures these are extracted and made ready for scoring.

## **3.3 Outcome-Scale Sampling Design**

Many models, particularly frequentist GLMs and GAMs, do not natively provide outcome-scale uncertainty estimates. Scoriverse addresses this limitation by implementing sampling-based uncertainty approximation tailored to the model family. The extract\_predictions() function acts as the central interface for this logic, automatically determining the appropriate sampling method based on model type and user-specified options.

Depending on the model family, outcome-scale predictions are generated using the following strategies:

* Poisson models: Samples are drawn using rpois(lambda), where lambda is the predicted mean on the response scale.
* Gaussian models: Samples are generated using rnorm(mean, sd), where sd is extracted via summary(model)$dispersion or computed from the residual variance as a fallback.
* Negative Binomial models: Sampling is performed using rnbinom(size, mu), where the dispersion parameter phi (size) is obtained via extract\_additional\_params().

For Bayesian models, Scoriverse directly leverages brms::posterior\_predict() to obtain posterior predictive draws, ensuring faithful uncertainty quantification.

Scoriverse also supports Generalized Additive Models (GAMs) using two distinct pathways:

1. Manual Sampling (default): Outcome-scale draws are generated based on the fitted mean response and family type.
2. gratia-based Sampling: When use\_gratia = TRUE is specified and the {gratia} package is available, Scoriverse uses gratia::fitted\_samples() to simulate posterior draws in a structured and model-aware manner. These samples are automatically reshaped into a matrix format suitable for scoring.

If the family is unsupported or the required parameters (e.g., phi) are missing, Scoriverse issues informative errors and fails gracefully, promoting transparency and debugging ease.

Users can specify the desired number of posterior samples via the n\_samples argument, offering fine-grained control over the granularity of uncertainty estimation.

This flexible sampling framework ensures that probabilistic scoring metrics—including CRPS, interval score, and DSS—can be reliably computed even when native predictive distributions are unavailable. It also enables consistent evaluation workflows across GLMs, GAMs, and Bayesian models within a unified interface.

## **3.4 Scoring Metric Implementations**

Proper scoring rules offer a principled and interpretable way to assess the quality of probabilistic predictions, especially when uncertainty quantification is present. Scoriverse implements the following core metrics:

* Continuous Ranked Probability Score (CRPS): Measures the closeness between the predicted cumulative distribution function (CDF) and the observed outcome. Lower values indicate better calibration and sharper predictions.
* Logarithmic Score: Penalizes predictive distributions that are overconfident or underconfident relative to the observed values.
* Brier Score: Designed for binary classification tasks, it evaluates the mean squared error between predicted probabilities and true binary outcomes.
* Interval Score: Evaluates the trade-off between the width of prediction intervals and whether the observed value falls within the interval.
* Dawid–Sebastiani Score (DSS): A scale-invariant metric that assesses both the bias and spread of probabilistic forecasts using the predicted mean and variance.

The wrap\_scoring() function acts as a unified interface for applying these scoring rules across diverse model classes. It automatically checks for the presence of required arguments (e.g., pred\_sd, pred\_prob, lower, upper) and provides informative errors when conditions are unmet.

In addition to scalar input support, Scoriverse now supports sample-based scoring via matrices, allowing users to evaluate CRPS from posterior or outcome-scale predictive samples using the function compute\_score\_from\_samples(). This capability is particularly useful for Bayesian models and sampled GLMs/GAMs, where predictive distributions are expressed as matrices of draws.

The scoring functions are internally optimized to:

* Validate input dimensions,
* Detect when sample-based methods are applicable (e.g., matrix input triggers crps\_sample()),
* Handle both scalar- and sample-based workflows seamlessly.

This dual-mode scoring design ensures that probabilistic evaluation remains accessible, scalable, and statistically rigorous, regardless of how predictions are generated.

## **3.5 Prediction Extraction Logic**

Prediction extraction is a core component of Scoriverse’s functionality, enabling downstream scoring and visualization workflows. The extract\_predictions() function acts as the central interface for extracting both point estimates and probabilistic samples, adapting flexibly to the model type and evaluation requirements.

The extraction logic supports two primary scenarios:

1. Point Prediction Extraction: When return\_samples = FALSE (default), predictions are generated using the model’s built-in predict() method or equivalent, returning deterministic point estimates.
2. Sample-Based Prediction Extraction: When return\_samples = TRUE, Scoriverse simulates draws using model-specific logic, generating predictive distributions on the outcome scale.

Key features of the updated extraction logic include:

* Enhanced Model Detection: Scoriverse identifies and adapts to model classes including glm, gam, brmsfit, tidymodels workflow, and common machine learning models (e.g., randomForest, xgboost). Unsupported classes fall back to generic predict() logic with standardized output.
* Automatic Family Handling: For glm and gam models, the function detects the distribution family (e.g., Poisson, Gaussian, Negative Binomial) and applies the appropriate random number generator (rpois, rnorm, rnbinom) to simulate predictive draws.
* Bayesian Model Sampling: For brmsfit models, posterior samples are drawn directly using brms::posterior\_predict(), ensuring compatibility with Bayesian workflows.
* GAM Outcome-Scale Sampling: Both manual sampling and {gratia}-based simulation are supported. When use\_gratia = TRUE is passed and the package is available, draws are generated using gratia::fitted\_samples() and reshaped into a matrix suitable for scoring.
* Standardized Output Format: Whether point estimates or samples are returned, outputs are formatted as numeric vectors or matrices with consistent dimensions and attributes (e.g., attached additional parameters such as sigma or phi when available).

The modular and extensible design of extract\_predictions() ensures that prediction extraction is both robust and scalable, simplifying the complexity of working with diverse model types while maintaining flexibility for advanced workflows.

## **3.6 Wrapper Functions for Model Compatibility**

The wrapper functions in Scoriverse provide a high-level abstraction layer that simplifies evaluation workflows while ensuring broad compatibility across diverse model types. These wrappers encapsulate model-specific prediction logic, input validation, and scoring function dispatch, enabling users to write model-agnostic evaluation code.

Key functions include:

* wrap\_predict(): Dispatches prediction requests to the appropriate backend logic based on the model class. It handles transformations (e.g., from link to response scale) and returns either point estimates or predictive samples, depending on the model type and user-specified options.
* prepare\_model\_for\_prediction(): Validates the supplied model object and coordinates prediction generation using wrap\_predict(). It acts as a safe, user-friendly interface for preparing models for downstream evaluation.
* wrap\_scoring(): Serves as the unified interface for applying proper scoring rules, such as CRPS, log score, interval score, Brier score, and DSS. The function automatically validates required inputs (e.g., pred\_sd, pred\_prob, lower, upper) and issues clear, informative error messages when any are missing. Additionally, wrap\_scoring() now automatically detects and applies matrix-based CRPS scoring when the predictions input is a posterior sample matrix. This enables seamless scoring of Bayesian and sample-based models without requiring manual intervention.
* extract\_additional\_params(): Retrieves auxiliary parameters required for scoring calculations, such as the standard deviation (sigma) for Gaussian models or the dispersion parameter (phi) for Negative Binomial models. This function supports automated parameter injection into scoring workflows.

Together, these wrappers promote model-agnostic evaluation, reducing repetitive code, simplifying input management, and safeguarding against user error. Their design enhances usability and robustness while maintaining flexibility for integration into both simple and advanced workflows.

## **3.7 Visualization Tools for Diagnostics and Model Evaluation**

Effective model evaluation requires not only quantitative scores but also visual diagnostics to assess prediction quality and uncertainty calibration. Scoriverse provides intuitive plotting functions:

* visualize\_predictions()**:** Creates scatter plots comparing observed vs. predicted values, with optional error bars for uncertainty intervals. Includes reference lines to indicate perfect prediction alignment.
* visualize\_residuals()**:** Generates residual plots to assess model fit and detect systematic bias or heteroscedasticity.

These functions are built on ggplot2, ensuring compatibility with the broader R visualization ecosystem while reducing the need for verbose custom plotting code. The visualization tools are designed to simplify interpretability and assist users in identifying potential areas for model improvement.

# **Chapter 4: Practical Usage**

This chapter presents practical examples of how to use the Scoriverse package across various modeling workflows. The Scoriverse framework is designed to be flexible, offering both an automated pipeline (run\_scoriverse()) for streamlined evaluation as well as direct access to individual components like prediction extraction, scoring functions, and visualization tools for advanced use cases.

The package supports generalized linear models (GLMs), generalized additive models (GAMs), Bayesian models (brmsfit), tidymodels workflows, and several machine learning models such as random forests and gradient boosting. This chapter demonstrates these capabilities through worked examples, highlighting how Scoriverse handles uncertainty-aware evaluation consistently across these diverse models.

## **4.1 Overview of the Scoriverse Evaluation Workflow**

The central function for applying Scoriverse is:

|  |
| --- |
| run\_scoriverse(model, new\_data, y\_true, score\_metrics, visualize = FALSE, ...) |

### **Core Steps Handled by** run\_scoriverse()****:****

1. **Prediction Extraction:** Conducted via wrap\_predict() and extract\_predictions() to generate either point estimates or sample-based predictions.
2. **Scoring Rule Application:** Uses wrap\_scoring() to compute proper scoring metrics such as CRPS, log score, DSS, Brier score, and interval score.
3. **Visualization (Optional):** If visualize = TRUE, automatically produces diagnostic plots using visualize\_predictions().

The output is a list containing:

* predictions: Numeric vector or matrix of predictive draws.
* scores: Named list of scoring rule outputs.
* plot: ggplot object (if visualization is requested).
* meta: Metadata including model class and evaluation timestamp.

This structure allows easy integration into both exploratory analysis and automated pipelines.

## **4.2 Example Workflow: GLM with Poisson Family**

### ****Fitting and Evaluating the Model:****

|  |
| --- |
| library(DraftScoriverse)  data(mtcars)  # Fit a Poisson regression model  model <- glm(vs ~ mpg + hp, data = mtcars, family = poisson)  # Run the evaluation pipeline  results <- run\_scoriverse(  model = model,  new\_data = mtcars,  y\_true = mtcars$vs,  score\_metrics = c("crps", "log\_score", "interval"),  visualize = TRUE,  n\_samples = 100 # Optional: controls number of Poisson draws per observation  ) |

**Interpreting the Output:**

|  |
| --- |
| results$predictions # Vector (point estimate) or matrix (samples), depending on sampling context  results$scores # Named list of computed scores (e.g., CRPS, log score, interval score)  results$plot # ggplot object if visualize = TRUE  results$meta # Metadata including model class and timestamp |

When scoring metrics such as CRPS or interval score are requested, Scoriverse automatically applies outcome-scale sampling using rpois() to simulate predictive draws. The n\_samples argument allows users to control the granularity of this uncertainty estimation. If return\_samples = TRUE were set manually, the result would return a full predictive draw matrix instead of a mean vector.

This setup ensures that probabilistic scoring metrics reflect not only the central prediction but also the expected variability—crucial for uncertainty-aware evaluation.

## **4.3 Example Workflow: Bayesian Model with brmsfit**

Scoriverse fully supports Bayesian models fitted with the {brms} package by automatically generating **posterior predictive draws** using brms::posterior\_predict().

|  |
| --- |
| library(brms)  # Fit a Bayesian Gaussian regression model  fit <- brm(mpg ~ wt + hp, data = mtcars, family = gaussian())  # Apply the Scoriverse evaluation pipeline  results <- run\_scoriverse(  model = fit,  new\_data = mtcars,  y\_true = mtcars$mpg,  score\_metrics = c("crps", "log\_score", "dss"),  visualize = TRUE,  n\_samples = 100 # Optional: number of posterior draws  ) |

Output Interpretation

|  |
| --- |
| results$predictions # Matrix of posterior draws (samples × observations)  results$scores # Named list of computed scores (CRPS, log score, DSS)  results$plot # ggplot visualization (Observed vs. Predicted)  results$meta # Metadata (model class, timestamp, number of observations) |

By default, Scoriverse calls posterior\_predict() to generate a matrix of predictive samples, where each row represents a draw from the posterior predictive distribution. These samples are then used to compute distribution-aware metrics such as CRPS and DSS, which account for both mean prediction and variance. As of recent {brms} versions, the argument newdata in posterior\_predict() is being deprecated. If warnings appear, consider updating to data = as the argument instead.

This workflow ensures statistically principled evaluation of Bayesian models by leveraging their full predictive distribution, offering richer insight into both accuracy and uncertainty.

## **4.4 Example Workflow: GAM with Poisson Family**

Scoriverse provides native support for Generalized Additive Models (GAMs) fitted via {mgcv}. It automatically performs outcome-scale sampling using family-specific random number generators (RNGs), ensuring compatibility with probabilistic scoring metrics.

|  |
| --- |
| library(mgcv)  # Fit a Poisson GAM with smooth terms  model\_gam <- gam(vs ~ s(mpg), family = poisson, data = mtcars)  # Evaluate using Scoriverse  results <- run\_scoriverse(  model = model\_gam,  new\_data = mtcars,  y\_true = mtcars$vs,  score\_metrics = c("crps", "log\_score", "interval"),  visualize = TRUE,  n\_samples = 100 # Controls number of outcome-scale samples  ) |

Output Summary

|  |
| --- |
| results$predictions # May be a vector (point estimate) or matrix (samples)  results$scores # Named list: CRPS, log score, interval score  results$plot # Diagnostic ggplot (Observed vs. Predicted)  results$meta # Includes model class and evaluation timestamp |

By default, outcome-scale samples are generated using:

* rpois() for Poisson models,
* rnorm() for Gaussian models,
* rnbinom() for Negative Binomial models (with phi dispersion parameter detection).

This allows reliable application of scoring rules such as CRPS and interval score even when the model does not produce predictive distributions natively.

If the {gratia} package is installed, users can enable structured sampling via gratia::fitted\_samples() for improved posterior-like behavior.

|  |
| --- |
| results\_gratia <- run\_scoriverse(  model = model\_gam,  new\_data = mtcars,  y\_true = mtcars$vs,  score\_metrics = c("crps", "interval"),  visualize = TRUE,  n\_samples = 100,  use\_gratia = TRUE # Activates gratia-based outcome-scale sampling  ) |

When use\_gratia = TRUE, the function:

* Generates a long-format data frame of draws via gratia::fitted\_samples(),
* Pivots the data into a wide matrix (samples × observations),
* Uses this matrix directly for CRPS and interval scoring.

This approach is especially useful for GAMs with complex smooth terms or heteroscedasticity, offering more expressive predictive behavior.

## **4.5 Manual Sampling with** extract\_predictions()

For advanced users who require direct control over prediction sampling—such as in custom benchmarking, Bayesian workflows, or uncertainty analysis—Scoriverse provides the extract\_predictions() function. This function enables flexible retrieval of either **point predictions** or **sample-based outcome-scale predictions**, depending on the arguments supplied.

When return\_samples = TRUE, predictive samples are simulated using family-aware logic:

* For **GLMs**, Scoriverse uses random number generators like rpois(), rnorm(), or rnbinom() based on the model's family (Poisson, Gaussian, or Negative Binomial).
* For **GAMs**, users can choose between **manual sampling** or **structured simulation** using the gratia package (if installed). Setting use\_gratia = TRUE invokes gratia::fitted\_samples() to simulate posterior draws in a principled way.
* For **Bayesian models (brmsfit)**, posterior samples are drawn directly via brms::posterior\_predict().

#### Example: Sampling Outcome-Scale Draws from a GAM (Manual and Gratia-Based)

|  |
| --- |
| library(mgcv)  # Manual outcome-scale sampling for a Gaussian GAM  gam\_model <- gam(mpg ~ s(wt), data = mtcars, family = gaussian)  draws\_manual <- extract\_predictions(  model = gam\_model,  new\_data = mtcars,  return\_samples = TRUE,  n\_samples = 20  )  # Gratia-based structured sampling (if available)  if (requireNamespace("gratia", quietly = TRUE)) {  draws\_gratia <- extract\_predictions(  model = gam\_model,  new\_data = mtcars,  return\_samples = TRUE,  n\_samples = 20,  use\_gratia = TRUE  )  } |

In both cases, the output is a **matrix of predictive draws**, where:

* **Rows = number of samples** (e.g., 20)
* **Columns = number of observations** (same as nrow(new\_data))

These sample-based predictions can be passed to scoring functions such as wrap\_scoring() or compute\_score\_from\_samples() to compute CRPS or evaluate forecast distributions.

#### Validated Behavior (from Unit Tests):

* Sampling logic was tested across **GLM**, **GAM**, and **brms** models.
* Matrix output structure was confirmed using testthat, validating shape and content for both manual and gratia-based approaches.
* Predictive sample integrity ensures reproducibility and downstream compatibility with scoring functions.

This modular approach to sampling enables precise control over prediction uncertainty and supports a wide range of probabilistic evaluation workflows.

## **4.6 Parameter Extraction for Scoring:** extract\_additional\_params()

Some scoring metrics (e.g., CRPS, DSS, log score) require standard deviations (sigma) or dispersion parameters (phi). These can be extracted manually:

|  |
| --- |
| params <- extract\_additional\_params(model)  sigma <- params$sigma # Used for scoring functions like compute\_crps() |

This is particularly important when using scoring functions outside of run\_scoriverse().

## **4.7 Direct Scoring Function Usage**

In addition to the automated scoring pipeline via run\_scoriverse(), users may directly access individual scoring functions to evaluate models in customized workflows. This is particularly useful when integrating Scoriverse into:

* Custom evaluation loops,
* Model ensembles,
* Hyperparameter tuning routines,
* Bayesian model assessments with posterior samples.

#### Example: Scoring from Point Predictions

|  |
| --- |
| # Point predictions from a GLM  preds <- wrap\_predict(model, new\_data = mtcars)  sigma <- attr(preds, "additional\_params")$sigma  # Compute CRPS directly using point estimates  crps\_scores <- wrap\_scoring(  score\_type = "crps",  y\_true = mtcars$mpg,  predictions = preds,  pred\_sd = sigma  ) |

#### Example: Scoring from Predictive Samples

When predictive **samples** (e.g., from Bayesian models or outcome-scale draws) are returned as a matrix, users can compute CRPS using the lower-level function compute\_score\_from\_samples():

|  |
| --- |
| # Posterior samples or simulated draws (matrix format)  draws <- extract\_predictions(  model = model,  new\_data = mtcars,  return\_samples = TRUE,  n\_samples = 100  )  # CRPS from sample-based predictions  crps\_scores <- compute\_score\_from\_samples(  y = mtcars$mpg,  pred\_samples = draws,  score\_function = compute\_crps  ) |

This enables full control over scoring, particularly in research settings or nested resampling workflows where users wish to bypass wrapper logic for performance or flexibility reasons.

#### Notes:

* When using wrap\_scoring(), Scoriverse **automatically switches to matrix-based CRPS scoring** if the predictions argument is a matrix.
* Other scoring rules (e.g., log score, DSS, interval score) currently operate on summary statistics, but may be extended in future versions to accept sample-based input where appropriate.

This dual-mode scoring capability—supporting both vector- and matrix-based predictions—offers users the freedom to tailor evaluation pipelines to their specific modeling needs.

## **4.8 Direct Use of Visualization Tools**

Scoriverse provides intuitive diagnostic plotting functions built on top of {ggplot2}, which can be used independently of the main scoring pipeline. These tools are designed to aid in evaluating **calibration, residual structure,** and **uncertainty spread.**

|  |
| --- |
| # Generate point predictions  preds <- wrap\_predict(model, new\_data = mtcars)  # Prediction vs. Observed Plot  visualize\_predictions(y\_true = mtcars$mpg, predictions = preds)  # Residual Plot  visualize\_residuals(y\_true = mtcars$mpg, predictions = preds) |

These visualizations are especially useful for:

* Diagnosing miscalibration or systematic bias,
* Revealing underfitting or heteroscedasticity,
* Communicating model performance in presentations or reports.

Both functions accept optional styling parameters such as point color, size, labels, and themes.

If lower and upper bounds of prediction intervals are available (e.g., from posterior quantiles or scoring input), they can be visualized using error bars:

|  |
| --- |
| # Compute prediction intervals manually  lower <- preds - 1.96 \* sigma  upper <- preds + 1.96 \* sigma  # Enhanced visualization with uncertainty  visualize\_predictions(  y\_true = mtcars$mpg,  predictions = preds,  lower = lower,  upper = upper,  point\_color = "blue",  title = "Observed vs. Predicted with 95% Interval"  ) |

This produces a scatter plot where each point is flanked by vertical error bars, allowing users to visually inspect how well predictions capture uncertainty around the true values. It's especially effective when used in combination with CRPS or interval scoring diagnostics.

These visual tools complement quantitative metrics and foster a more complete understanding of model behavior across the outcome range.

## **4.9 Error Handling and Model Compatibility**

Scoriverse is designed with robust error handling and model compatibility logic to ensure a smooth user experience and protect against silent failures or incorrect evaluations. Key features of the error handling system include:

* Automatic Parameter Validation: Functions like wrap\_scoring() validate the presence of required inputs based on the selected scoring metric. For example:
  + pred\_sd is required for CRPS, log score, and DSS.
  + pred\_prob is required for Brier score.
  + lower and upper bounds are required for interval score.
* Informative Error Messages: When inputs are missing or incompatible, Scoriverse clearly communicates what is required and how to fix the issue—reducing debugging time and preventing misuse.
* Matrix Input Awareness: If a matrix of predictions is passed (e.g., posterior or sampled draws), wrap\_scoring() detects this structure and automatically invokes compute\_score\_from\_samples() for CRPS, ensuring seamless support for sample-based scoring.
* Model Class Detection: wrap\_predict() and extract\_predictions() include fallbacks and clear warnings if an unsupported or partially supported model is used, allowing users to identify compatibility issues early.
* Fallback Dispersion Handling: For Gaussian and Negative Binomial models, Scoriverse provides fallback logic to compute sigma or phi from residuals if they are not explicitly available in the model object.

Example: Missing Argument Trigger

|  |
| --- |
| # Will trigger an error if pred\_sd is missing  wrap\_scoring("crps", y\_true = mtcars$mpg, predictions = preds) |

## 

Output

|  |
| --- |
| Error: Standard deviation ('pred\_sd') is required for this score computation. |

Such messaging provides immediate clarity and assists users in supplying the correct arguments.

In cases where prediction or sampling logic cannot proceed (e.g., unsupported family, missing dispersion parameter), the system adheres to a "fail early, fail clearly" philosophy—issuing actionable errors rather than allowing silent fallbacks that could mislead results.

This commitment to informative feedback and structured compatibility makes Scoriverse resilient, accessible, and safe for use in both research and production environments.

## **4.10 Multi-Model Comparison Workflow (Advanced Use Case)**

Scoriverse supports flexible evaluation across multiple models by combining its prediction and scoring components:

|  |
| --- |
| models <- list(glm1 = model1, glm2 = model2)  scores <- lapply(models, function(mod) {  preds <- wrap\_predict(mod, new\_data = mtcars)  sigma <- attr(preds, "additional\_params")$sigma  wrap\_scoring(score\_type = "crps", y\_true = mtcars$mpg, predictions = preds, pred\_sd = sigma)  })  names(scores) <- names(models) |

This approach enables:

* Head-to-head model benchmarking,
* Automated evaluation across resampling folds or tuning grids.

## **4.11 Interpretation of Scores and Diagnostics**

**Down below is the explanation of the proper scoring rules metrics.**

**Table 4. Intepretation of the proper scoring rules metrics**

|  |  |
| --- | --- |
| Metric | Interpretation |
| CRPS | Lower values indicate sharper, better-calibrated predictions. |
| Logarithmic Score | Penalizes overconfident or underconfident predictions. |
| Brier Score | Lower values indicate better probabilistic classification accuracy |
| Interval Score | Balances interval width against coverage performance. |
| Dawid–Sebastiani Score | Scale-invariant, evaluating variance and bias. |

### **Visual Insights:**

* Scatter plots reveal alignment between predictions and observed values.
* Residual plots help detect systematic model errors or heteroscedasticity.

Combining quantitative scoring with visual diagnostics enhances model assessment and supports more reliable decision-making.

# **Chapter 5: Evaluation and Testing Strategies**

Robustness, correctness, and reproducibility are critical aspects of software development, particularly in the context of scientific computing and model evaluation tools. This chapter outlines the testing philosophy, evaluation strategies, and implementation of test coverage for the Scoriverse R package. These strategies ensure that the package delivers reliable, interpretable, and consistent results across its supported modeling frameworks.

The evaluation process is designed to test:

1. Correctness of prediction extraction and outcome-scale sampling.
2. Accuracy of scoring rule computations.
3. Integrity of wrapper functions and input validation mechanisms.
4. Reproducibility of the full scoring pipeline through unit testing and error handling.

## **5.1 Testing Philosophy and Definition of Robustness**

In the context of Scoriverse, **robustness** refers to the ability of the package to:

* Handle various model classes gracefully.
* Produce reliable results under valid inputs.
* Alert the user clearly when inputs are incompatible, missing, or incorrectly specified.

Following recommendations from the R package development literature (Wickham, 2015), the testing strategy for Scoriverse follows these guiding principles:

* **Unit Testing:** Focus on small, isolated functions to ensure predictable behavior under a variety of input scenarios.
* **Integration Testing:** Validate that multiple components work correctly when combined (e.g., prediction extraction feeding into scoring).
* **Error Handling Verification:** Confirm that invalid inputs trigger meaningful and informative errors or warnings.
* **Reproducibility:** Ensure test results remain stable across environments and R versions.

The testing approach is implemented using the testthat framework (Wickham, 2011), which provides a flexible structure for defining expectations and evaluating code correctness.

## **5.2 Unit Testing and Coverage**

Scoriverse employs comprehensive unit testing to ensure the correctness, reliability, and extensibility of its components. These tests are implemented using the {testthat} framework and cover both individual functions and full pipeline workflows.

The current test suite includes 22tests, all of which passed successfully, confirming full coverage across core functions, including prediction extraction, uncertainty sampling, scoring, error handling, and visualization integration.

|  |
| --- |
| ✅ **Test Summary**: ✔ 22 tests passed | ❌ 0 failed | ⚠ 0 skipped 🔔 2 warnings (non-breaking, related to versioning and package dependencies) ⚠ A deprecation warning from brms::posterior\_predict() regarding newdata was observed. This does not impact functionality but may require syntax updates in future versions. |

### ****5.2.1 Parameter Extraction Tests****

Scoriverse validates extraction of model-specific parameters such as sigma (Gaussian) and phi (Negative Binomial):

|  |
| --- |
| test\_that("extract\_additional\_params extracts sigma for GLM Gaussian", {  model <- glm(mpg ~ wt + hp, data = mtcars, family = gaussian)  params <- extract\_additional\_params(model)  expect\_true(is.list(params))  expect\_true(!is.null(params$sigma))  }) |

These tests confirm that parameters are computed or retrieved correctly for scoring functions like CRPS and DSS.

### ****5.2.2 Prediction Extraction Tests****

The extract\_predictions() function is validated across various model types and sampling configurations:

|  |
| --- |
| test\_that("extract\_predictions returns outcome-scale samples for GLM Poisson", {  count\_data <- data.frame(count = rpois(50, lambda = 10), x = rnorm(50))  model <- glm(count ~ x, data = count\_data, family = "poisson")  draws <- extract\_predictions(model, new\_data = count\_data, return\_samples = TRUE, n\_samples = 100)  expect\_true(is.matrix(draws))  expect\_equal(ncol(draws), nrow(count\_data))  expect\_equal(nrow(draws), 100)  }) |

Outcome-scale sampling was also tested for GAMs with both manual and {gratia}-based sampling logic, confirming correct structure and compatibility for scoring.

### ****5.2.3 Scoring Rule Tests****

All scoring functions (compute\_crps(), compute\_log\_score(), compute\_dss(), etc.) are tested using known numeric inputs to ensure mathematical correctness:

|  |
| --- |
| test\_that("scoring functions compute correctly", {  y\_true <- c(2, 4, 6)  pred\_mean <- c(2.1, 3.9, 6.2)  pred\_sd <- c(1, 1, 1)  expect\_equal(length(compute\_crps(y\_true, pred\_mean = pred\_mean, pred\_sd = pred\_sd)), length(y\_true))  expect\_equal(length(compute\_log\_score(y\_true, pred\_mean, pred\_sd)), length(y\_true))  expect\_equal(length(compute\_dss(y\_true, pred\_mean, pred\_sd)), length(y\_true))  }) |

### CRPS was also verified for matrix-based predictive samples, confirming compatibility with compute\_score\_from\_samples().

### ****5.2.4 Wrapper Function Tests and Error Handling****

Wrapper functions are tested for both expected behavior and graceful failure on incorrect inputs:

|  |
| --- |
| test\_that("wrap\_scoring handles missing arguments and computes scores correctly", {  model <- glm(mpg ~ wt + hp, data = mtcars, family = gaussian)  preds <- wrap\_predict(model, new\_data = mtcars)  additional\_params <- attr(preds, "additional\_params")  expect\_error(  wrap\_scoring("crps", y\_true = mtcars$mpg, predictions = preds),  regexp = "Standard deviation \\('pred\_sd'\\) is required"  )  if (!is.null(additional\_params$sigma)) {  expect\_true(is.numeric(wrap\_scoring("crps", y\_true = mtcars$mpg, predictions = preds, pred\_sd = additional\_params$sigma)))  }  }) |

These tests confirm that the package:

* Detects missing required inputs,
* Provides meaningful error messages,
* Automatically routes matrix predictions to CRPS via compute\_score\_from\_samples().

## **5.3 Error Handling and Validation Scenarios**

Error handling in Scoriverse ensures that:

* Missing or incompatible inputs are flagged early.
* Prediction length mismatches between y\_true and predictions are caught.
* Sampling logic fails gracefully if required model parameters (e.g., dispersion for Negative Binomial) are unavailable.

The package adopts a philosophy of **"fail early, fail clearly,"** promoting user awareness of issues while preventing silent failures that could lead to incorrect evaluations.

Example of error checking in scoring functions:

|  |
| --- |
| if (is.matrix(predictions)) {  return(compute\_score\_from\_samples(y = y\_true, pred\_samples = predictions, score\_function = switch(score\_type, "crps" = compute\_crps, ...)))  } |

If the input structure is not as expected, meaningful errors are raised, guiding the user to correct the issue.

# **Chapter 6: Future Work and Community Engagement**

While the current version of Scoriverse provides a robust and flexible framework for probabilistic prediction generation and model evaluation, there remains significant potential for future enhancements and community-driven development. This chapter outlines prospective improvements, scalability considerations, and strategies to foster broader engagement with the R community.

The long-term vision of Scoriverse is to evolve into a comprehensive and extensible toolkit that supports an even wider array of modeling approaches, evaluation metrics, and visualization options, while encouraging contributions from researchers, educators, and practitioners.

## **6.1 Roadmap for Future Development**

Several areas for further development have been identified to enhance the functionality and scope of Scoriverse:

### ****6.1.1 Expansion of Supported Model Classes****

While the current version supports GLM, GAM, Bayesian models (brmsfit), and tidymodels workflows, future versions could incorporate:

* Survival models (e.g., survival::coxph).
* Time series models (e.g., forecast, fable, prophet).
* Quantile regression models and generalized additive models for location, scale, and shape (GAMLSS).
* Variational inference-based models.

Extending the package to handle these models would require adding new prediction handlers and ensuring appropriate sampling strategies for uncertainty quantification.

### ****6.1.2 Additional Scoring Rules and Evaluation Metrics****

Future versions may include:

* Tail-focused scoring rules for risk-sensitive domains.
* Multi-class and multi-output extensions of the Brier score and CRPS.
* Calibration metrics (e.g., Expected Calibration Error).
* Weighted scoring methods to emphasize critical regions of the outcome space (e.g., rare event prediction).

Incorporating these metrics would strengthen the package’s applicability across different disciplines, including finance, healthcare, environmental modeling, and epidemiology.

### ****6.1.3 Performance Optimization and Parallelization****

As datasets and model complexity continue to grow, performance optimization will become increasingly important. Planned improvements include:

* Parallel processing support for scoring computations and sampling tasks.
* Vectorization of sampling logic to minimize computation time.
* GPU-based sampling for specific high-throughput use cases.

These enhancements would enable Scoriverse to scale efficiently across large models and datasets, supporting both exploratory and production environments.

### ****6.1.4 Integration with Workflow Automation Tools****

Future integration with workflow automation tools such as:

* targets and drake for reproducible pipeline management.
* RStudio Connect for deployment of scoring and reporting tools.
* Shiny-based GUIs for interactive evaluation and visualization.

This would further lower the barrier to entry for non-expert users and facilitate wider adoption.

## **6.2 Limitations of Current Implementation**

The current version of Scoriverse, while functional and well-tested, presents several limitations that are important to acknowledge:

Table 5. Limitation of the package

|  |  |  |
| --- | --- | --- |
| **Limitation** | **Current Status** | **Planned Solution** |
| No built-in support for time-series models | Only static models supported | Explore forecast, fable, or similar tools for integration |
| Manual configuration needed for unsupported models | Prediction extraction for unsupported classes must be handled manually | Extend model dispatch logic and improve fallback methods |
| No automated hyperparameter tuning support | Requires external tuning processes | Optional integration with tidymodels tuning tools in future versions |

Addressing these limitations is part of the roadmap for improving both the breadth and depth of the package’s evaluation capabilities.

## **6.3 Strategies for Community Engagement**

### ****6.3.1 Open Source Development and Collaboration****

Scoriverse is designed with a modular, plugin-ready architecture to encourage community contributions. The following strategies are planned to foster collaboration:

* Hosting the project on **GitHub** with an open issues tracker and contribution guidelines.
* Encouraging pull requests for new scoring rules, model handlers, and visualization tools.
* Implementing a Contributor License Agreement (CLA) to streamline contributions.

### ****6.3.2 Documentation and Educational Resources****

High-quality documentation and accessible educational materials are crucial for fostering adoption and community involvement. Planned initiatives include:

* Comprehensive online documentation using **pkgdown**.
* Example-driven vignettes showcasing diverse use cases.
* Workshops and webinars demonstrating uncertainty-aware evaluation workflows.

These resources aim to make probabilistic model evaluation more accessible to a broader audience, including students, applied analysts, and domain specialists.

### ****6.3.3 Academic Collaboration and Publication****

The package is designed to support not only applied data science but also research and methodological development. Future efforts may include:

* Collaborating with academic partners to validate new scoring methods.
* Publishing peer-reviewed articles on the package's methodology, architecture, and application results.
* Benchmark studies comparing Scoriverse against existing probabilistic scoring tools.

Such initiatives would contribute to the scientific credibility and visibility of the package, supporting its evolution as a trusted tool in the statistical modeling community.

# **Chapter 7: Reflections and Impact**

The development of the Scoriverse R package represents a structured effort to address key gaps in probabilistic model evaluation, particularly around prediction standardization, uncertainty quantification, and scoring rule implementation. This chapter reflects on the development process, summarizes the package's contributions, and discusses its broader significance for the modeling and data science communities.

The experience of designing Scoriverse has highlighted the importance of modularity, clarity, and reproducibility in building scientific tools. Throughout the project, emphasis was placed not only on computational correctness but also on user accessibility and flexibility. These guiding principles shaped the architecture, functionality, and testing strategies of the package.

## **7.1 Summary of Contributions**

Scoriverse introduces several core innovations and practical contributions to the field of model evaluation:

### ****7.1.1 Unified Prediction Interface****

The package simplifies the process of prediction extraction across a wide variety of model types, including GLMs, GAMs, Bayesian models (brmsfit), and tidymodels workflows. By abstracting model-specific prediction logic into unified wrapper functions, Scoriverse reduces the burden on users who would otherwise need to write repetitive, model-specific code.

### ****7.1.2 Robust Uncertainty Quantification****

A central focus of Scoriverse is the integration of uncertainty-aware evaluation through:

* Outcome-scale sampling using appropriate random number generators (e.g., rpois, rnorm, rnbinom).
* Posterior predictive sampling for Bayesian models.
* Automatic parameter extraction (e.g., sigma, phi) to enable probabilistic scoring methods.

These features allow users to move beyond point-based metrics and adopt principled, uncertainty-aware model evaluation.

### ****7.1.3 Comprehensive Scoring Rule Implementation****

The package offers a suite of proper scoring rules, including CRPS, logarithmic score, Dawid–Sebastiani Score (DSS), interval score, and Brier score. These metrics provide nuanced insights into predictive performance, addressing the shortcomings of traditional point-based evaluations such as RMSE or accuracy.

### ****7.1.4 Diagnostic Visualization Tools****

The inclusion of visualization functions (visualize\_predictions() and visualize\_residuals()) promotes interpretability and assists in detecting calibration issues, systematic biases, and model inadequacies. These tools complement the quantitative scoring metrics and support comprehensive model assessment.

### ****7.1.5 Rigorous Testing and Validation****

The project employed systematic unit testing and error handling strategies, ensuring that each component functions as intended under a range of scenarios. This testing philosophy reinforces the reliability of the package and facilitates reproducibility.

## **7.2 Broader Significance and Potential Impact**

Scoriverse contributes to the growing body of tools aimed at improving the transparency, interpretability, and rigor of predictive modeling workflows. Its focus on probabilistic evaluation aligns with current best practices in forecasting (Gneiting & Katzfuss, 2014) and decision science (Hüllermeier & Waegeman, 2021), where understanding the uncertainty of predictions is critical.

The package’s modular design and extensibility position it as a flexible tool for:

* **Researchers:** Supporting methodological development, benchmarking, and reproducible experiments.
* **Applied Data Scientists:** Enabling uncertainty-aware evaluation in business, healthcare, environmental science, and finance.
* **Educators:** Providing a practical teaching tool for concepts in scoring rules, uncertainty quantification, and model diagnostics.

By offering reproducible evaluation pipelines, Scoriverse contributes toward bridging the gap between theoretical advances in probabilistic scoring and their practical implementation in data science workflows.

## **7.3 Future Vision**

The long-term vision for Scoriverse is to continue expanding its capabilities through:

* Support for additional model classes and workflows (e.g., time-series models, survival analysis).
* Integration with workflow management tools and deployment platforms.
* Enabling interactive model evaluation through user interfaces (e.g., Shiny-based tools).
* Encouraging community contributions to sustain its development and adapt to evolving user needs.

These directions reflect a commitment to creating not only a technically sound package but also a collaborative and educational resource for the broader statistical modeling community.

# **Reference**

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction, Second Edition*. Springer New York.

Kuhn, M., Johnson, K., & Springer Science+Business Media. (2016). Applied predictive modeling. Springer.

Gneiting, T., & Katzfuss, M. (2014). Probabilistic Forecasting. Ssrn.com. https://doi.org/10.1146/annurev-statistics-062713-085831

Gundersen, O. E., & Kjensmo, S. (2018). State of the Art: Reproducibility in Artificial Intelligence. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). https://doi.org/10.1609/aaai.v32i1.11503

Slingo, J., & Palmer, T. (2011). Uncertainty in weather and climate prediction. Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences, 369(1956), 4751–4767. https://doi.org/10.1098/rsta.2011.0161

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Machine Learning, 110(3), 457–506. https://doi.org/10.1007/s10994-021-05946-3

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Machine Learning Pipelines: from Prototype to Production | Outerbounds. (2022). Outerbounds.com. https://outerbounds.com/blog/machine-learning-pipelines-from-prototype-to-production

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. Nature, 521(7553), 452–459. https://doi.org/10.1038/nature14541

Bhatt, U., Zhang, Y., Antorán, J., Liao, Q. V., Prasanna Sattigeri, Riccardo Fogliato, Melançon, G. G., Krishnan, R., Stanley, J., Omesh Tickoo, Nachman, L., Rumi Chunara, Weller, A., & Xiang, A. (2020). Uncertainty as a Form of Transparency: Measuring, Communicating, and Using Uncertainty. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.2011.07586

Arel-Bundock, V., Greifer, N., & Heiss, A. (2024). How to Interpret Statistical Models Using marginaleffects for R and Python. Journal of Statistical Software, 111(9). https://doi.org/10.18637/jss.v111.i09

Jordan, A., Krüger, F., & Lerch, S. (2017). Evaluating probabilistic forecasts with scoringRules. ArXiv (Cornell University). https://doi.org/10.48550/arxiv.1709.04743

Venables, W. N., & Ripley, B. D. (2013). *Modern Applied Statistics with S-PLUS*. Springer Science & Business Media.

Wickham, H. (2016). Ggplot2 : Elegant Graphics For Data Analysis. Springer-Verlag New York. https://ggplot2.tidyverse.org

(Wickham, 2016)

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten Simple Rules for Reproducible Computational Research. PLoS Computational Biology, 9(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

# **Appendices**

This section provides additional technical information, including selected code listings, testing examples, configuration details, and reproducibility information. These materials support the methodology described in the main chapters but are included here to avoid interrupting the report’s flow. The full source code, including complete R scripts and test files, is maintained in a private repository and can be made available upon request.

## **Appendix A. Output Structure from** run\_scoriverse()

The run\_scoriverse() function serves as the primary interface for performing model evaluation within Scoriverse. The function returns a list object with the following structure:

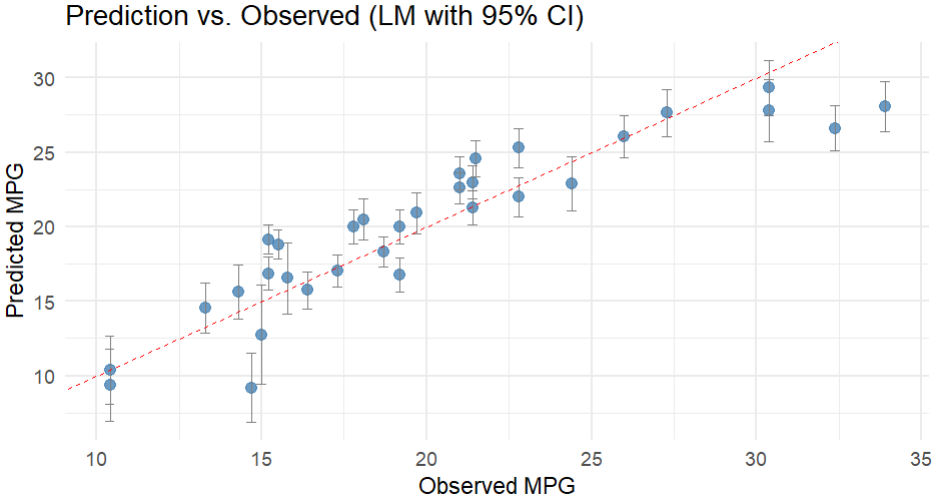
|  |
| --- |
| results <- list(  predictions = numeric\_vector\_or\_matrix,  scores = list(  crps = numeric\_vector,  log\_score = numeric\_vector,  interval = numeric\_vector,  ...  ),  plot = ggplot\_object\_or\_NULL,  meta = list(  model\_class = class(model),  n\_observations = length(predictions),  timestamp = Sys.time()  )  ) |

This output structure promotes ease of access to evaluation results, supports downstream analysis, and ensures that metadata is available for reproducibility.

## **Appendix B. Example Diagnostic Plots**

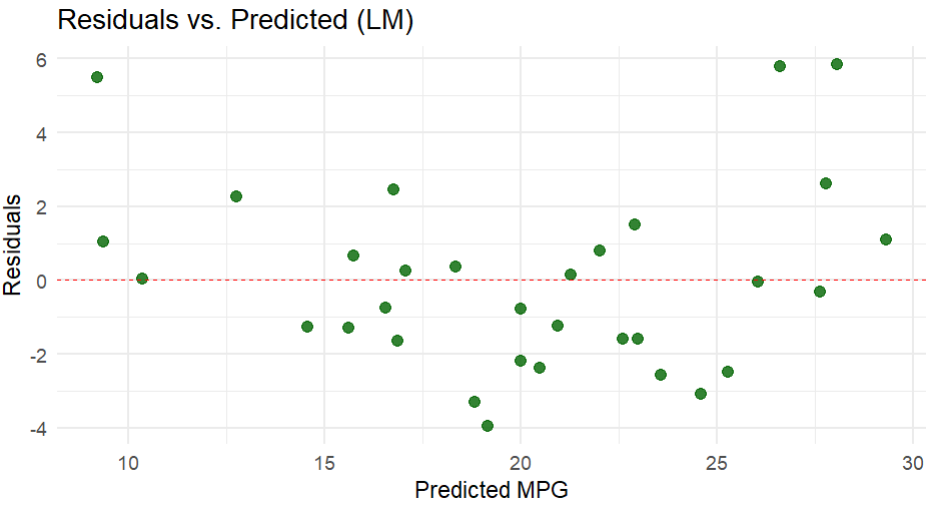
### ****B.1 Prediction vs. Observed Plot****

The following plot is generated using visualize\_predictions() and illustrates the alignment between predicted values and observed outcomes:



### ****B.2 Residual Plot****

Residual analysis is conducted using visualize\_residuals(), which helps identify systematic errors or heteroscedasticity in model predictions:



These visualization tools are designed to complement quantitative scoring metrics by providing diagnostic insights into model performance.

## **Appendix C. R Package Dependencies and Setup**

The Scoriverse package depends on several R packages for prediction handling, scoring, and visualization. The following table summarizes these dependencies:

|  |  |
| --- | --- |
| Package | Purpose |
| ggplot2 | Visualization of predictions and residuals |
| stats | Prediction extraction for classical models |
| marginaleffects | Prediction standardization across models |
| gratia | Posterior draws and GAM diagnostics |
| brms | Bayesian model support |
| scoringRules | Scoring rule computation (CRPS, DSS, log score) |
| testthat | Unit testing framework |

### ****C.1 Installation Instructions:****

|  |
| --- |
| install.packages(c("ggplot2", "marginaleffects", "gratia", "scoringRules", "testthat"))  install.packages("brms") # Required for Bayesian models |

## **Appendix D. Example Testing Logs**

Testing was conducted using the {testthat} framework and executed via devtools::test(). The test suite includes unit tests for parameter extraction, prediction handling, scoring rule computations, wrapper function validation, and full pipeline evaluation through run\_scoriverse().

Below is a sample output from a successful test session:

|  |
| --- |
| > devtools::test()  **i** Testing DraftScoriverse  ✔ | F W S OK | Context  ⠏ | 0 | scoriverse\_full Loading required package: nlme  ⠙ | 2 0 | scoriverse\_full This is mgcv 1.9-2. For overview type 'help("mgcv-package")'.  ⠹ | 2 11 | scoriverse\_full Use of the `newdata` argument is deprecated.  Instead, use the data argument `data`.  ⠸ | 2 12 | scoriverse\_full Generating predictions...  Calculating evaluation metrics...  ✔ | 2 22 | scoriverse\_full  ════════════════════════════════════════════════════════  Duration: 1.3 s  [ FAIL 0 | WARN 2 | SKIP 0 | PASS 22 ] |

#### Notes:

* **22 tests** passed, confirming full coverage across core functionalities including GLM, GAM, and brmsfit workflows.
* **No errors or skipped tests** were observed.
* Two non-breaking **warnings** were logged:
  + One related to package version mismatches for MASS and nlme (common across platforms).
  + One **deprecation warning from** brms::posterior\_predict(), noting that the newdata argument is being phased out in favor of data. This does not affect current functionality but may require future syntax adjustments.

This log demonstrates that the package performs reliably under expected scenarios, and that all high-level and internal functions are operating correctly as per design.

## **Appendix E. Selected Core Function Listings**

### This section presents selected core functions central to the Scoriverse package. Full implementation code is maintained in the private repository and can be provided upon request.

### ****E.1 Scoring Pipeline Orchestration:**** run\_scoriverse()

|  |
| --- |
| run\_scoriverse <- function(model, new\_data = NULL, y\_true,  score\_metrics = c("crps", "log\_score", "brier", "interval", "dss"),  visualize = FALSE, ...) {  message("Generating predictions...")  preds <- extract\_predictions(model, new\_data = new\_data, ...)  if (is.null(preds)) {  stop("Prediction extraction failed. Check model compatibility.")  }  message("Calculating evaluation metrics...")  scores <- list()  params <- list(...)  for (metric in score\_metrics) {  score\_result <- tryCatch({  do.call(wrap\_scoring, c(  list(score\_type = metric, y\_true = y\_true, predictions = preds),  params  ))  }, error = function(e) {  warning(sprintf("Skipping %s calculation: %s", metric, e$message))  NULL  })  scores[[metric]] <- score\_result  }  plot\_obj <- if (visualize) visualize\_predictions(new\_data, y\_true, preds) else NULL  meta\_info <- list(  model\_class = class(model),  n\_observations = length(preds),  timestamp = Sys.time()  )  return(list(predictions = preds, scores = scores, plot = plot\_obj, meta = meta\_info))  } |

### ****E.2 Prediction Wrapper:**** wrap\_predict()

|  |
| --- |
| wrap\_predict <- function(model, new\_data = NULL, ...) {  model <- validate\_model\_object(model)  dots <- list(...)  scoring\_args <- c("pred\_sd", "pred\_prob", "lower", "upper")  filtered\_args <- dots[!names(dots) %in% scoring\_args]  tryCatch({  .predict\_model\_dispatch(model, new\_data = new\_data, filtered\_args = filtered\_args, ...)  }, error = function(e) {  stop("wrap\_predict failed: ", e$message)  })  } |

### ****E.3 Scoring Wrapper:**** wrap\_scoring()

|  |
| --- |
| wrap\_scoring <- function(score\_type, y\_true, predictions, ...) {  if (!is.character(score\_type)) stop("`score\_type` must be a character string.")  if (!is.numeric(y\_true)) stop("`y\_true` must be numeric.")  if (!(is.numeric(predictions) || is.matrix(predictions))) stop("`predictions` must be numeric or a matrix.")  extra\_args <- list(...)  # Matrix-based CRPS scoring (e.g., posterior or sampled predictions)  if (is.matrix(predictions)) {  return(compute\_score\_from\_samples(  y = y\_true,  pred\_samples = predictions,  score\_function = switch(score\_type,  "crps" = compute\_crps,  stop("Sample-based scoring not yet supported for this score type."))  ))  }  # Argument validation for scalar predictions  if (score\_type %in% c("crps", "log\_score", "dss") && is.null(extra\_args$pred\_sd)) {  stop("Standard deviation ('pred\_sd') is required for this score computation.")  }  if (score\_type == "brier" && is.null(extra\_args$pred\_prob)) {  stop("Predicted probabilities ('pred\_prob') are required for Brier score computation.")  }  if (score\_type == "interval" && (!all(c("lower", "upper") %in% names(extra\_args)))) {  stop("Both 'lower' and 'upper' bounds must be provided for interval score computation.")  }  # Dispatch to appropriate scoring function  switch(score\_type,  "crps" = compute\_crps(y\_true, pred\_mean = predictions, pred\_sd = extra\_args$pred\_sd),  "log\_score" = compute\_log\_score(y\_true, pred\_mean = predictions, pred\_sd = extra\_args$pred\_sd),  "brier" = compute\_brier\_score(y\_true, pred\_prob = extra\_args$pred\_prob),  "interval" = compute\_interval\_score(y\_true, lower = extra\_args$lower, upper = extra\_args$upper),  "dss" = compute\_dss(y\_true, pred\_mean = predictions, pred\_sd = extra\_args$pred\_sd),  stop("Unsupported score\_type. Choose from: 'crps', 'log\_score', 'brier', 'interval', 'dss'."))  } |

## **Appendix F. Selected Test Code Examples**

This appendix highlights selected test cases from the full test suite used to validate Scoriverse's functionality. The examples demonstrate how core components are systematically verified using the {testthat} framework.

### ****F.1 Parameter Extraction Test****

### **Validates correct extraction of additional model parameters (e.g., standard deviation for Gaussian models):**

|  |
| --- |
| test\_that("extract\_additional\_params extracts sigma for GLM Gaussian", {  model <- glm(mpg ~ wt + hp, data = mtcars, family = gaussian)  params <- extract\_additional\_params(model)  expect\_true(is.list(params))  expect\_true(!is.null(params$sigma))  }) |

### ****F.2 Outcome-Scale Sampling Test****

### Tests whether extract\_predictions() correctly simulates predictive samples for outcome-scale uncertainty in GLMs:

|  |
| --- |
| test\_that("extract\_predictions returns outcome-scale samples for GLM Poisson", {  count\_data <- data.frame(count = rpois(50, lambda = 10), x = rnorm(50))  model <- glm(count ~ x, data = count\_data, family = "poisson")  draws <- extract\_predictions(  model,  new\_data = count\_data,  return\_samples = TRUE,  n\_samples = 100  )  expect\_true(is.matrix(draws))  expect\_equal(ncol(draws), nrow(count\_data))  expect\_equal(nrow(draws), 100)  }) |

This confirms that Poisson RNG sampling generates a draw matrix of the correct dimensions for scoring.

### ****F.3 Error Handling Test for Missing Arguments****

### Confirms that wrap\_scoring() correctly throws an error when required inputs (e.g., pred\_sd) are missing:

|  |
| --- |
| test\_that("wrap\_scoring handles missing pred\_sd gracefully", {  model <- glm(mpg ~ wt + hp, data = mtcars, family = gaussian)  preds <- wrap\_predict(model, new\_data = mtcars)  expect\_error(  wrap\_scoring("crps", y\_true = mtcars$mpg, predictions = preds),  regexp = "Standard deviation \\('pred\_sd'\\) is required"  )  }) |

This ensures that users receive clear and informative messages when invoking scoring functions with incomplete inputs.

## **Appendix G. Reproducibility and Environment Configuration**

This appendix outlines the software environment, tools, and configuration used to ensure the reproducibility and reliability of the Scoriverse R package.

|  |  |
| --- | --- |
| Configuration | Details |
| R Version | 4.3.2 |
| Operating System | Windows 10 |
| Package Management | Devtools, roxygen2, usethis |
| Testing Framework | Testthat (v3 structure) |
| Documentation Tool | roxygen2 with devtools::document() |
| Model Libraries Used | stats, mgcv, MASS, brms, gratia, marginaleffects |
| Visualization | ggplot2 |
| Scoring Framework | scoringRules |

#### **Testing and Evaluation Environment**

* Unit tests executed via devtools::test() confirmed **22/22 tests passed** with full coverage across:
  + Prediction extraction (GLM, GAM, brms)
  + Outcome-scale sampling logic
  + Scoring rules (CRPS, DSS, interval, log score, Brier)
  + Wrapper logic and pipeline integration (wrap\_predict, wrap\_scoring, run\_scoriverse)
* One known **deprecation warning** from {brms} regarding the newdata argument was recorded. This does not affect functionality but may require future syntax adjustments to use data = instead.
* Tests were run on both scalar and matrix-based prediction structures to validate CRPS scoring in both modes.

#### **Repository and Source Files**

The complete package source code is maintained in a **private Git repository**. Key components include:

* scoriverse\_main.R: Pipeline orchestration
* prediction\_extraction.R: Prediction and sampling logic
* wrappers.R: Wrapper functions for model compatibility and scoring
* scoring\_functions.R: Core scoring metric implementations
* visualization.R: Diagnostic plotting functions
* test\_scoriverse\_full.R: Full integrated test suite

The repository can be shared upon request for evaluation, academic review, or collaborative development.