Model Evaluation, Selection & Visualization with easystats : : **CHEAT SHEET**

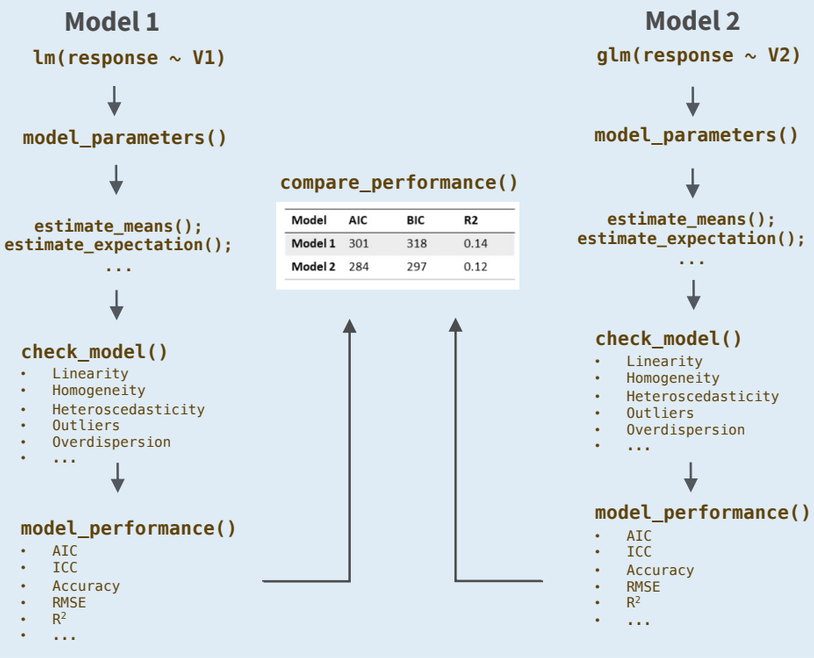
Kaiyuan Liu (kl3447) & Zhuolin Luo (zl2852)

Basics

easystats is a **collective** framework of packages, such as "performance", "parameters", etc., installed in R for statistical modeling, and statistics reporting and visualization.

This cheat sheet will focus on the functions, examples, and visualizations related to **model parameters, predictions, assumptions, performance, and selection** within **easystats**.

Workflow



Parameters Reports the parameter features of a model

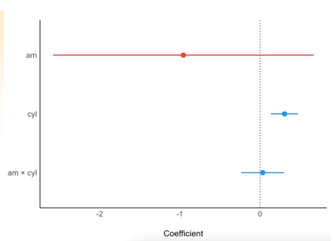
- **parameters(model) / model_parameters(model)**
  - This function computes and extracts model parameters.
  - Taking Classical Regression Model as an example → it returns Parameter | Coefficient | SE | 95% CI | t(143) | p

```
model_parameters(model)
#> Parameter                          | Coefficient |   SE |         95% CI | t(143) |      p
#> ---------------------------------------------------------------------------------------
#> (Intercept)                        |        2.89 | 0.36 | [ 2.18,  3.60] |   8.01 | < .001
#> Petal Length                       |        0.26 | 0.25 | [-0.22,  0.75] |   1.07 | 0.287
#> Species [versicolor]               |       -1.66 | 0.53 | [-2.71, -0.62] |  -3.14 | 0.002
#> Species [virginica]                |       -1.92 | 0.59 | [-3.08, -0.76] |  -3.28 | 0.001
#> Petal Width                        |        0.62 | 0.14 | [ 0.34,  0.89] |   4.41 | < .001
#> Petal Length × Species [versicolor] |       -0.09 | 0.26 | [-0.61,  0.42] |  -0.36 | 0.721
#> Petal Length × Species [virginica]  |       -0.13 | 0.26 | [-0.64,  0.38] |  -0.50 | 0.618
```

For Visualization

The result can be visualized by a dot-and-whisker plot (x-axis: coefficient, y-axis: coefficient name) by 'see' package.

- **plot(parameters(model))**



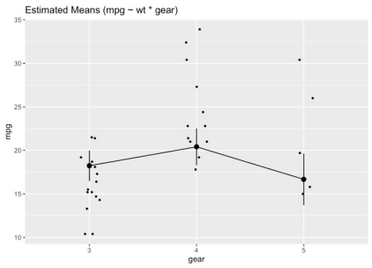
Predictions makes model-based estimations and predictions

- **estimate_means(model,** at, fixed, transform, ci, backend**)**
  - This function estimates the average values at each factor level.
- **estimate_contrasts(model,** contrast, at, fixed, transform, ci, p_adjust, method, adjust**)**
  - This function estimates and tests contrasts between different factor levels.
- **estimate_slopes(model,** trend, at, ci**)**
  - This function estimates the slopes of numeric predictors at different factor levels or alongside a numeric predictor.
- **estimate_expectation(model)**
  - This function predicts the response variable using the model.

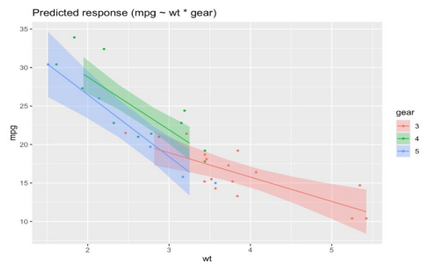
For Visualization

Notice: The output of above functions is in table form. The results can be visualized by 'see' package. Here are two Examples:

- **plot(estimate_means(model))**



- **plot(estimate_expectation(model))**



Assumption Diagnostics

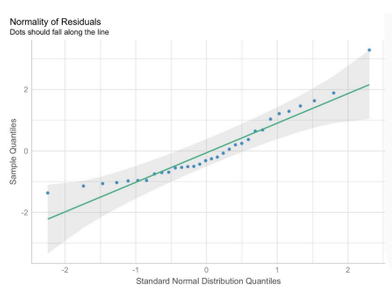
Tests whether the model satisfies a specific assumption and returns the corresponding value

- **check_autocorrelation(model,** nsim**)**
  - This function checks model for independence of residuals, i.e., for autocorrelation of error terms.
- **check_collinearity(model,** verbose, component**)**
  - This function checks regression models for multicollinearity by calculating the variance inflation factor (VIF).
- **check_convergence(model,** tolerance**)**
  - This function provides an alternative convergence test for merMod-objects.
- **check_distribution(model)**
  - This function may help to check a model's distributional family and see if the model-family should probably be reconsidered.
- **check_heterogeneity_bias(model,** select, group**)**
  - This function checks if model predictors or variables may cause a heterogeneity bias, i.e., if variables have a within- and/or between-effect.
- **check_multimodal(model)**
  - This function checks if a distribution is unimodal or multimodal.
- **check_heteroscedasticity(model)**
  - This function provides significance testing for linear regression models assumes that the model errors (or residuals) have constant variance. If this assumption is violated the p-values from the model are no longer reliable.
- **check_homogeneity(model,** method**)**
  - This function checks model for homogeneity of variances between groups described by independent variables in a model.
- **check_itemscale(model)**
  - This function computes various measures of internal consistencies applied to (sub)scales, which items were extracted using parameters::principal_components().
- **check_normality(model,** effects**)**
  - This function checks the model for (non-)normality of residuals.
- **check_outliers(model,** method, threshold**)**
  - This function checks for and locates influential observations (i.e., "outliers") via several distance and/or clustering methods.
- **check_overdispersion(model)**
  - This function checks generalized linear (mixed) models for overdispersion.
- **check_predictions(model,** iterations, check_range, re_formula**)**
  - This function conducts posterior predictive checks to figure out the difference between real data and simulated data produced by the fitted model.
- **check_singularity(model,** tolerance**)**
  - This function checks mixed models for boundary fits.
- **check_sphericity(model)**
  - This function checks the model for violation of sphericity.
- **check_zeroinflation(model,** tolerance**)**
  - This function checks whether count models are overfitting or underfitting zeros in the outcome.

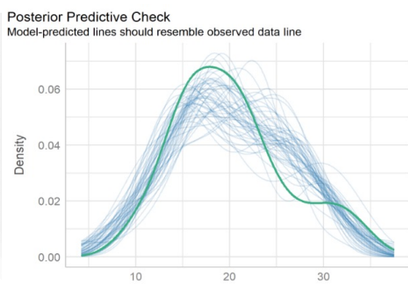
For Visualization

Notice that most results representing a single model assumption can be visualized by 'see' package: **plot(check_xxx(model))**. For example,

- **plot(check_normality(model))**



- **plot(check_predictions(model))**



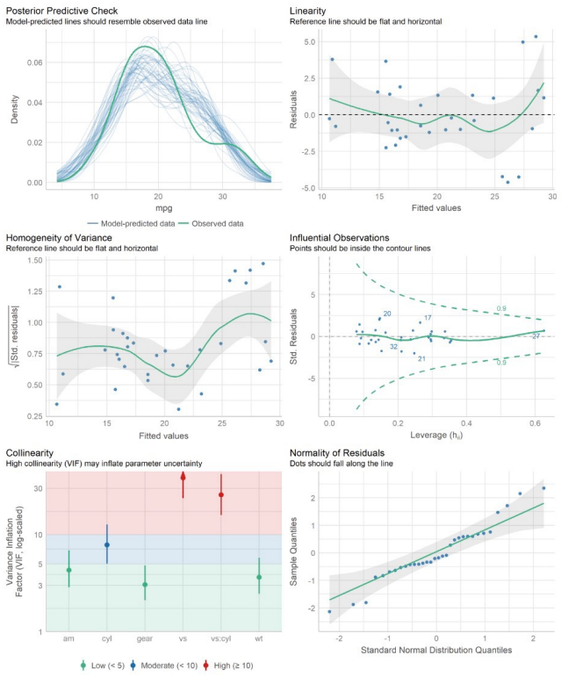
Kaiyuan Liu (kl3447) • kl3447@columbia.edu • Zhuolin Luo (zl2852) • zl2852@columbia.edu • package version 0.5.2 • Updated: 2022-11

Model Evaluation, Selection & Visualization with easystats : : **CHEAT SHEET**

Kaiyuan Liu (kl3447) & Zhuolin Luo (zl2852)

Complex Diagnosis

- **check_model(model,** dot_size, line_size, panel, check, alpha, dot_alpha, colors, theme, detrend, verbose**)**
  - This function provides a visual check of model various assumptions (normality of residuals, normality of random effects, linear relationship, homogeneity of variance, multicollinearity, etc.)



Performance

Reports the metrics evaluating a model's performance

- **performance_accuracy(model,** method, k, n, verbose**)**
  - This function calculates the predictive accuracy of linear or logistic regression models.
- **performance_aic(model,** estimator, verbose**)**
  - This function returns the AIC. For models with a transformed response variable, it returns the corrected AIC value.
- **performance_hosmer(model,** n_bins**)**
  - This function checks the model quality of logistic regression models.
- **performance_logloss(model,** verbose**)**
  - This function computes the log loss for models with binary outcomes.
- **performance_mae(model)**
  - This function computes the mean absolute error of models.
- **performance_mse(model)**
  - This function computes the mean square error of linear models.
- **performance_pcp(model,** ci, method, verbose**)**
  - This function computes the percentage of correct predictions (PCP) for models with binary outcomes.
- **performance_rmse(model,** normalized, verbose**)**
  - This function computes root mean squared error for (mixed-effects) models, including Bayesian regression models.
- **performance_roc(model,** predictions, new_data**)**
  - This function calculates simple ROC curves of x/y coordinates based on the response and predictions of a binomial model.
- **performance_rse(model)**
  - This function computes the residual standard error of linear models.
- **performance_score(model,** verbose**)**
  - This function calculates the logarithmic, quadratic/Brier, and spherical scores from a model with binary or count outcome.
- **r2(model)**
  - This function returns a list containing values related to the "most appropriate" r-squared for the given model.
- **icc(model)**
  - This function calculates the intraclass-correlation coefficient (ICC) for various mixed-model objects. Similar to R-squared, the ICC provides information on the explained variance and can be interpreted as "the proportion of the variance explained by the grouping structure in the population".

Model Performance Summary

- **model_performance(model) / performance(model)**
  - This function computes indices of model performance for regression models. Depending on the model object, typical indices might be r-squared, AIC, BIC, RMSE, ICC or LOOIC.
  - Taking linear model as an example, it returns AIC | BIC | R2 | R2 (adj.) | RMSE | Sigma

```
#> # Indices of model performance
#>
#> AIC     |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
#> ---------------------------------------------------------
#> 156.010 | 161.873 | 0.830 |     0.819 | 2.444 | 2.568
```

  - Taking logistic regression as an example, it returns AIC | BIC | Tjur's R2 | RMSE | Sigma | Log_loss | Score_log | Score_spherical | PCP

```
#> # Indices of model performance
#>
#> AIC    |    BIC | Tjur's R2 |  RMSE | Sigma | Log_loss | Score_log | Score_spherical |   PCP
#> ------------------------------------------------------------------------------------------
#> 31.298 | 35.695 |     0.478 | 0.359 | 0.934 |    0.395 |   -14.903 |           0.095 | 0.743
```

  - Taking linear mixed model as an example, it returns AIC | AICc | BIC | R2 (cond.) | R2 (marg.) | ICC | RMSE | Sigma

```
#> # Indices of model performance
#>
#> AIC      |     AICc |      BIC | R2 (cond.) | R2 (marg.) |   ICC |   RMSE |  Sigma
#> ---------------------------------------------------------------------------------
#> 1755.628 | 1756.114 | 1774.786 |      0.799 |      0.279 | 0.722 | 23.438 | 25.592
```

Model Selection

Compares the performance among different models to help select the model

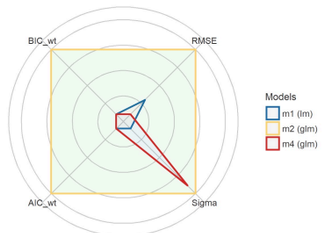
- **compare_performance(model1, model2, model3, ……,** metrics, rank, estimator, verbose**)**
  - This function can be used to compare the performance and quality of several models (including models of different types).
  - Notice that one can compute a composite index of model performance and sort the models from the best one to the worse by setting **rank=True**

```
#> # Comparison of Model Performance Indices
#>
#> Name |   Model |   RMSE |  Sigma | AIC weights | BIC weights | Performance-Score
#> ---------------------------------------------------------------------------------
#> m2   |     glm |  0.359 |  0.934 |       1.000 |       1.000 |           100.00%
#> m4   |     glm |  3.043 |  1.132 |    2.96e-06 |    1.63e-05 |            46.89%
#> m1   |      lm |  2.444 |  2.568 |    8.30e-28 |    3.99e-28 |            46.09%
#> m3   | lmerMod | 23.438 | 25.592 |    0.00e+00 |    0.00e+00 |             0.00%
```

For Visualization

Notice that visualizations for the metrics evaluating performance of different models are available for model comparison and selection. For example,

- **plot(compare_performance(m1, m2, m4, rank = True)**



References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Aki Vehtari, & Rubin, D. B. (2014). *Bayesian data analysis*. Chapman & Hall/Crc.

Gelman, A., & Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Hox, J. J. (2010). *Multilevel analysis : techniques and applications*. Lawrence Erlbaum Associates.

Lüdecke, D. (2022, October 14). *Assessment of Regression Models Performance*. R Functions and Packages for Political Science Analysis. https://cran.r-project.org/web/packages/performance/performance.pdf

Lüdecke, D., Makowski, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Wiernik, B. M. (n.d.-a). *Assessment of Regression Models Performance*. Easystats.github.io. https://easystats.github.io/performance/

Lüdecke, D., Makowski, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Wiernik, B. M. (n.d.-b). *Estimation of Model-Based Predictions, Contrasts and Means*. Easystats.github.io. Retrieved November 13, 2022, from https://easystats.github.io/modelbased/

Lüdecke, D., Makowski, D., Ben-Shachar, M. S., Wiernik, B. M., Højsgaard, S., & Patil, I. (n.d.). *Processing of Model Parameters*. Easystats.github.io. Retrieved November 13, 2022, from https://easystats.github.io/parameters/

Lüdecke, D., Makowski, D., Patil, I., Ben-Shachar, M. S., Wiernik, B. M., & Waggoner, P. (n.d.). *Model Visualisation Toolbox for easystats and ggplot2*. Easystats.github.io. Retrieved November 13, 2022, from https://easystats.github.io/see/

Explanation

Data scientists have lots of work highly relevant to statistical modeling, and model evaluation and selection are crucial steps for bringing an effective model to solve problems. For them, R is a commonly-used programming language that contains various useful packages for statistical modeling. For example, “easystats” is a powerful collective package in R for statistical modeling, statistics reporting, visualization, and so on. Currently, no excellent summary of the functions related to model evaluation and selection with “easystats” exists. Therefore, we intend to create a cheat sheet summarizing the functions and examples of model evaluation, model selection, and corresponding visualization in “easystats” to provide data scientists with more convenience. It covers multiple aspects of model evaluation and selection, such as parameter checking, model-based estimations and predictions, model assumption diagnostics, performance evaluation, and comparison among different models. In each selection, functions with concise descriptions and examples are listed. For more intuitive selection and evaluation, the usage for visualizing model parameters, assumptions, and comparisons among models is also provided. Although we have learned how to check model assumptions and plot using other R packages in class, “easystats” provides another way to do so, featured with much easier to use.

Through creating this cheat sheet, we got familiar with a new useful package in R, “easystats”, and we had a deeper understanding of the process and aspects of model evaluation and selection. Firstly, except for the common assumptions like collinearity we learned previously, we learned more model assumptions and characteristics, including singularity, sphericity, a model’s distributive family, and so on. We also learned how to check them using “easystats” functions. Secondly, we learned various ways to visualize model parameters, normality, collinearity, performance, etc. In this way, we can evaluate and select models more intuitively and straightforwardly instead of only observing the tables or values. Next time, we may attempt to create a cheat sheet for the whole statistical modeling process with “easystats”, from preprocessing, training, and evaluation. We may also try summarizing more relevant examples and visualizations in this cheat sheet.