R-squared for Bayesian regression models*

Andrew Gelman† Ben Goodrich‡ Jonah Gabry‡ Imad Ali§

8 August 2017

Abstract

The usual definition of R^2 (variance of the predicted values divided by the variance of the data) has a problem for Bayesian fits, as the numerator can be larger than the denominator. We propose an alternative definition similar to one that has appeared in the survival analysis literature: the variance of the predicted values divided by the variance of predicted values plus the variance of the errors. Our plan is for this to be printed automatically for linear and generalized linear regression models fit using `rstanarm`, our R package for fitting Bayesian applied regression models with Stan.

1. The problem

Consider a regression model of outcomes y and predictors X with predicted values $\mathrm{E}(y|X,\theta)$, fit to data $(X,y)_n$, $n = 1,\ldots,N$. Ordinary least squares regression yields an estimated parameter vector $\hat{\theta}$ with predicted values $\hat{y}_n = \mathrm{E}(y|X_n,\hat{\theta})$ and residual variance $V_{n=1}^N \hat{y}_n$, where we are using the notation,

$$V_{n=1}^N z_n = \frac{1}{n-1}\sum_{n=1}^N (z_n - \bar{z})^2, \text{ for any vector } z.$$

The proportion of variance explained,

$$\text{classical } R^2 = \frac{V_{n=1}^N \hat{y}_n}{V_{n=1}^N y_n}, \tag{1}$$

is a commonly used measure of model fit, and there is a long literature on interpreting it, adjusting it for degrees of freedom used in fitting the model, and generalizing it to other settings such as hierarchical models; see Xu (2003) and Gelman and Pardoe (2006).

Here we consider how to extend the concept of R^2 to apply to Bayesian model fitting. Our motivation is the `rstanarm` R package (Gabry and Goodrich, 2017) for fitting applied regression models using Stan (Stan Development Team, 2017). `rstanarm` contains a set of wrapper functions that enable the user to express regression models with traditional R syntax (R Core Team, 2017), for example, $\texttt{y} \sim \texttt{x1} + \texttt{x2} + \texttt{x3}$, and then fit these models using Bayesian inference, allowing the incorporation of prior information in the input stage and yielding posterior simulations and predictions in the output. After fitting a model using the `stan_glm` or `stan_glmer` function, we would like to display an estimate of R^2 along with the estimated coefficients and standard errors.

2. Defining R^2 based on the variance of estimated prediction errors

Our first thought for Bayesian R^2 is to simply use the posterior mean estimate of θ to create Bayesian predictions $\hat{y}_n$ and then plug these into the classical formula (1). This has two problems:

*We thank Frank Harrell and Aki Vehtari for helpful comments and the National Science Foundation, Office of Naval Research, Institute for Education Sciences, and Sloan Foundation for partial support of this work.

†Department of Statistics and Department of Political Science, Columbia University.

‡Institute for Social and Economic Research and Policy, Columbia University.

§Department of Statistics, Columbia University.

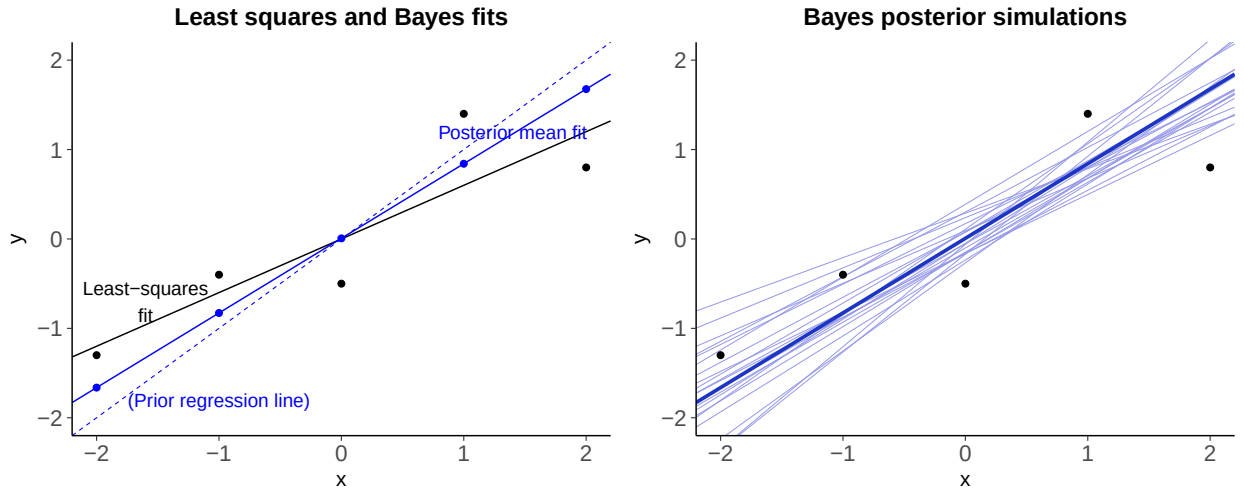

Figure 1: *Simple example showing the challenge of defining R^2 for a fitted Bayesian model.* Left plot: *data, least-squares regression line, and fitted Bayes line, which is a compromise between the prior and the least-squares fit. The standard deviation of the fitted values from the Bayes model (the blue dots on the line) is greater than the standard deviation of the data, so the usual definition of R^2 will not work.* Right plot: *posterior mean fitted regression line along with 20 draws of the line from the posterior distribution. To define a Bayesian R^2 we compute equation* (3) *for each posterior simulation draw and then take the median.*

first, it is dismisses uncertainty to use a point estimate in Bayesian computation; and, second, the ratio as thus defined can be greater than 1. When $\hat{\theta}$ is estimated using ordinary least squares, and assuming the regression model includes a constant term, the numerator of (1) is less than or equal to the denominator by definition; more generally, though, there is no requirement that this be the case, and it would be awkward to say that a fitted model explains more than 100% of the variance.

To see an example where the simple R^2 would be inappropriate, consider a model $y = \alpha + \beta x +$ error with a strong prior on (α, β) and only a few data points. Figure 1a shows data and the least-squares regression line (with R^2 of 0.77). We then do a Bayes fit with informative priors $\alpha \sim \text{N}(0, 0.2^2)$ and $\beta \sim \text{N}(1, 0.2^2)$, and a flat prior on the error standard deviation σ. The standard deviation of the fitted values from the Bayes model is 1.3, while the standard deviation of the data is only 1.08, so the square of this ratio—R^2 as defined in (1)—is greater than 1. Figure 1b shows the posterior mean fitted regression line along with 20 draws of the line $y = \alpha + \beta x$ from the fitted posterior distribution of (α, β).[1]

Instead of working with (1) directly, we generalize a different formula for explained variance:

$$\text{alternative } R^2 = \frac{V_{n=1}^N \hat{y}_n}{V_{n=1}^N \hat{y}_n + V_{n=1}^N r_n}, \tag{2}$$

where $r_n = y_n - \hat{y}_n$ are the residuals of the fitted model. The advantage of (2) is that it is always between 0 and 1 by construction, no matter what procedure is used to construct the estimate $\hat{y}$.

Versions of expression (2) have appeared in the survival analysis literature (Kent and O'Quigley, 1988; Choodari-Oskoo et al., 2010), where it makes sense to use expected rather than observed data variance in the denominator. Our motivation is slightly different but the same mathematical principles apply.

[1]Code for this example is available at `https://github.com/jgabry/bayes_R2`.

In Bayesian inference we do not have a point estimate $\hat{\theta}$ but rather a set of posterior simulation draws, θ^s, $s = 1, \ldots, S$. For each θ^s, we can compute the vector of predicted values $\hat{y}_n^s = \mathrm{E}(y|X_n, \theta^s)$, the vector of errors $e_n^s = y_n - \hat{y}_n^s$, and thus the proportion of variance explained,

$$\text{Bayesian } R_s^2 = \frac{V_{n=1}^N \hat{y}_n^s}{V_{n=1}^N \hat{y}_n^s + V_{n=1}^N e_n^s}. \tag{3}$$

We can then summarize this by its posterior median, for example. It would also be possible to obtain an estimate of the posterior uncertainty in R^2, although it is not clear that this would be of much interest.

In expectation (3) is equivalent to the ratio of sums of squares

$$\text{alternative Bayesian } R_s^2 = \frac{\sum_{n=1}^N (\hat{y}_n^s)^2}{\sum_{n=1}^N (\tilde{y}_n^s)^2}, \tag{4}$$

where $\tilde{y}$ is distributed according to the posterior predictive distribution. When computed in this way, each Bayesian R_s^2 is the ratio of the sum of squares of a posterior draw of the conditional mean to the sum of squares of a draw from the posterior predictive distribution. Using `rstanarm` naming conventions this corresponds to

$$\text{alternative Bayesian } R_s^2 = \frac{\sum_{n=1}^N \texttt{posterior\_linpred(fit)[s,n]}^2}{\sum_{n=1}^N \texttt{posterior\_predict(fit)[s,n]}^2}.$$

In fact, to calculate the entire S-vector of R^2 values for a linear regression would only require a single line of R code:

```
rsq <- rowSums(posterior_linpred(fit)^2) / rowSums(posterior_predict(fit)^2)
```

However, although equal in expectation to (3), in practice the denominator can be smaller than the numerator due to Monte Carlo variation and we can obtain R^2 values greater than 1. Rarely would R^2 be large enough in practical applications for Monte Carlo variation to produce a ratio greater than 1, but we prefer to carry out the computations using the (3) to avoid the possibility entirely.

For a linear regression model we can compute the Bayesian R^2 defined in (3) using the `posterior_linpred` function in the `rstanarm` package and a few additional lines of code:

```
bayes_R2 <- function(fit) {
  y <- get_y(fit)
  yhat <- posterior_linpred(fit)
  e <- -1 * sweep(yhat, 2, y)
  var_yhat <- apply(yhat, 1, var)
  var_e <- apply(e, 1, var)
  var_yhat / (var_yhat + var_e)
}

## Example
M1 <- stan_glm(y ~ x)
print(median(bayes_R2(M1)))
```

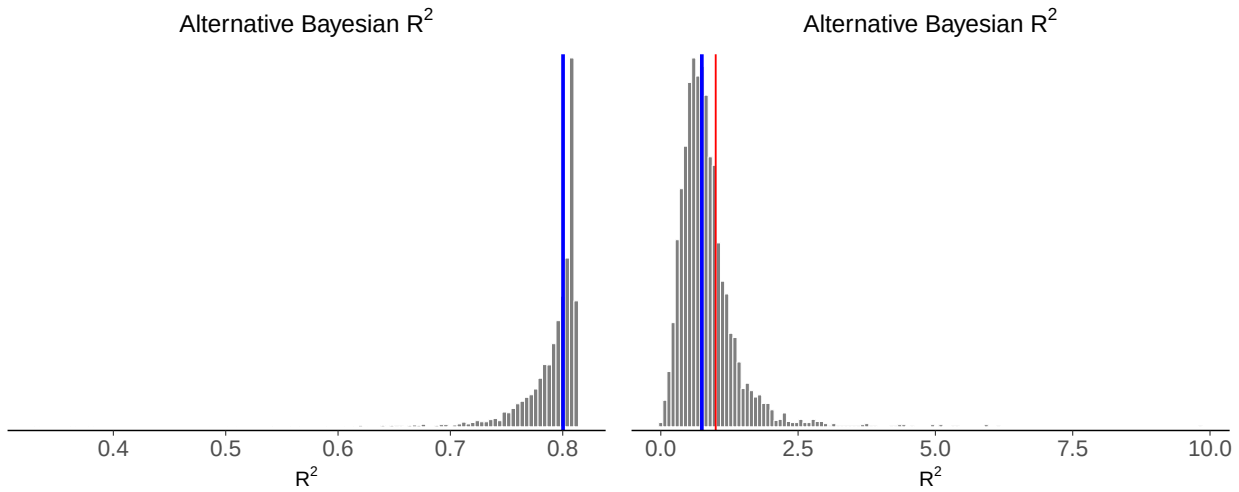

Figure 2: *Comparison of Bayesian R^2 approaches. The vertical blue lines are drawn at median values.* Left plot: *Bayesian R^2 computed using* (3) . Right plot: *Alternative Bayesian R^2 computed using equation* (4)*. The red line is drawn at 1.*

For the example in Figure 1, the median R^2 from equation (3), as calculated by the above code, is 0.80. Also, in case it might be of interest, we report the posterior mean and standard deviation of R^2 in this example: they are 0.79 and 0.03. In comparison, if we were to replace the denominator of (3) by $V_{n=1}^{N} y_n$, we would get a median R^2 of 1.44. If we were to use the alternative Bayesian R^2 as defined in (4) we would get a lower median (0.75) but, as shown in Figure 2, we also get many individual draws much greater than 1.

3. Models with group-specific terms

The `rstanarm` package also provides the `stan_glmer` function for fitting models with group-specific coefficients that have unknown covariance matrices. The linear predictor for these models is often written as $X\beta + Zb$, following the parameterization used by the `glmer` function in the `lme4` package (Bates et al., 2015). In the classical formulation Zb can be considered as part of the model's error term rather than the conditional mean of the outcome, but from a Bayesian perspective we condition on Z when fitting the model and it makes sense by default to include the Zb term when computing $\hat{y}$. However, the `bayes_R2` function in `rstanarm` provides a `re.form` argument that allows the user to override this default and specify which, if any, of the group-level terms should factor into the computation of $\hat{y}$ and thus R^2.

4. Discussion

R^2 has well-known problems as a measure of model fit, but it can be a handy quick summary for linear regressions and generalized linear models (see, for example, Hu et al. (2006)) and we would like to produce it by default when fitting Bayesian regressions. Our preferred solution is to use the posterior median of the ratio (3): predicted variance divided by predicted variance plus error variance.

A new issue then arises, though, when fitting a set of a models to a single dataset. Now that the denominator of R^2 is no longer fixed, we can no longer interpret an increase in R^2 as a improved fit to a fixed target. We think this particular loss of interpretation is necessary: from a

Bayesian perspective, a concept such as "explained variance" can ultimately only be interpreted in the context of a model. The denominator of (3) can be interpreted as an estimate of the expected variance of predicted future data from the model under the assumption that the predictors X are held fixed; alternatively the predictors can be taken as random, as suggested by Helland (1987) and Tjur (2009). In either case, we can consider our Bayesian R^2 as a data-based estimate of the proportion of variance explained for new data. If the goal is to see continual progress of the fit to existing data, one can simply track the decline in the estimated error variance, $V_{n=1}^N e_n^s$.

Moving forward, we plan to implement our Bayesian R^2 by default in `rstanarm`, not just for linear regression but also for generalized linear models; see the appendix for a generalized version of the `bayes_R2` function. The concept of "explained variance" makes most sense for linear models with equal variance, but given that we can essentially compute R^2 for free, we might as well do so. An alternative is to summarize residual error on the log-probability scale, in which case we recommend using approximate leave-one-out cross-validation (Vehtari et al., 2017), which can be done using the `loo` function within `rstanarm`. If desired, changes in expected log-probability scores can be converted back to an R^2 scale as discussed by Nagelkerke (1991).

One issue that arises when using R^2 to evaluate and compare models is overfitting. As with other measures of predictive model fit, overfitting should be less of an issue with Bayesian inference because averaging over the posterior distribution is more conservative than taking a least-squares or maximum likelihood fit, but predictive accuracy for new data will still on average be lower, in expectation, than for the data used to fit the model (Gelman et al., 2014). We will include a `newdata` argument in the `bayes_R2` function that can be used in the case that new or held-out data are actually available, but otherwise to correct for that bias one might want to replace $V_{n=1}^N e_n^s$ in the denominator of (3) by its expectation for new data, or more generally one could construct an overfitting-corrected R^2 in the same way that is done for log-score measures via cross-validation. In the present paper we are trying to stay close to the sprit of the original R^2 in quantifying the model's fit to the data at hand.

References

Bates, D., M. Mächler, B. Bolker, and S. Walker (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software 67*, 1–48.

Choodari-Oskoo, B., P. Royston, and M. K. B. Parmar (2010). A simulation study of predictive ability measures in a survival model I: Explained variation measures. *Statistics in Medicine 31*, 2627–2643.

Gabry, J. and B. Goodrich (2017). rstanarm: Bayesian applied regression modeling via Stan. R package version 2.16.1. `http://mc-stan.org`.

Gelman, A., J. Hwang, and A. Vehtari (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing 24*, 997–1016.

Gelman, A. and I. Pardoe (2006). Bayesian measures of explained variance and pooling in multilevel (hierarchical) models. *Technometrics 48*, 241–251.

Helland, I. S. (1987). On the interpretation and use of R^2 in regression analysis. *Biometrics 43*, 61–69.

Hu, B., M. Palta, and J. Shao (2006). Properties of R^2 statistics for logistic regression. *Statistics in Medicine 25*, 1383–1395.

Kent, J. T. and J. O'Quigley (1988). Measures of dependence for censored survival data. *Biometrika 75*, 525–534.

Nagelkerke, N. J. D. (1991). A note on a general definition of the coefficient of determination. *Biometrika 78*, 691–692.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Stan Development Team (2017). The Stan C++ library, version 2.16.0. `http://mc-stan.org`.

Tjur, T. (2009). Coefficient of determination in logistic regression models—A new proposal: The coefficient of discrimination. *American Statistician 63*, 366–372.

Vehtari, A., A. Gelman, and J. Gabry (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing 27*, 1413–1432.

Xu, R. (2003). Measuring explained variation in linear mixed-effects models. *Statistics in Medicine 22*, 3527–3541.

Appendix

This modified version of the `bayes_R2` function works with Bayesian linear and generalized linear models fit using the `stan_glm` function in the `rstanarm` package. The only differences compared to the function presented in the body of the paper are the requirement of specifying `transform=TRUE` in the call to `posterior_linpred` (to apply the inverse-link function to the linear predictor) and a few lines to account for the binomial models that have a number of trials greater than 1.

```
# Compute Bayesian R-squared for linear and
# generalized linear models.
#
# @param fit A fitted model object returned by stan_glm.
# @return A vector of R-squared values with length equal to
#       the number of posterior draws.
#
bayes_R2 <- function(fit) {
  y <- get_y(fit)
  yhat <- posterior_linpred(fit, transform = TRUE)
  if (family(fit)$family == "binomial" && NCOL(y) == 2) {
    trials <- rowSums(y)
    y <- y[, 1]
    yhat <- yhat %*% diag(trials)
  }
  e <- -1 * sweep(yhat, 2, y)
  var_yhat <- apply(yhat, 1, var)
  var_e <- apply(e, 1, var)
  var_yhat / (var_yhat + var_e)
}
```

The code for the `bayes_R2` function included in `rstanarm` is more complicated than the function presented here in order to accommodate models fit using `stan_glmer` and various special cases. The full source code is available in the `rstanarm` GitHub repository (`http://github.com/stan-dev/rstanarm`).