Chapter 11

Assumptions, diagnostics, and model evaluation

Note:
The code examples in this chapter are in R, but you've seen how to create similar figures in using seaborn in Python

In this chapter we turn to the assumptions of the regression model, along with diagnostics that can be used to assess whether some of these assumptions are reasonable. Some of the most important assumptions rely on the researcher's knowledge of the subject area and may not be directly testable from the available data alone. Hence it is good to understand the ideas underlying the model, while recognizing that there is no substitute for engagement with data and the purposes for which they are being used. We show different sorts of plots of data, fitted models, and residuals, developing these methods in the context of real and simulated-data examples. We consider diagnostics based on predictive simulation from the fitted model, along with numerical summaries of fit, including residual error, explained variance, external validation, and cross validation. The goal is to develop a set of tools that you can use in constructing, interpreting, and evaluating regression models with multiple predictors.

11.1 Assumptions of regression analysis

We list the assumptions of the regression model in decreasing order of importance.

1. *Validity.* Most important is that the data you are analyzing should map to the research question you are trying to answer. This sounds obvious but is often overlooked or ignored because it can be inconvenient. Optimally, this means that the outcome measure should accurately reflect the phenomenon of interest, the model should include all relevant predictors, and the model should generalize to the cases to which it will be applied.
   For example, with regard to the outcome variable, a model of incomes will not necessarily tell you about patterns of total assets. A model of test scores will not necessarily tell you about child intelligence or cognitive development.
   Choosing inputs to a regression is often the most challenging step in the analysis. We are generally encouraged to include all relevant predictors, but in practice it can be difficult to determine which are necessary and how to interpret coefficients with large standard errors. Chapter 19 discusses the choice of inputs for regressions used in causal inference.
   A sample that is representative of all mothers and children may not be the most appropriate for making inferences about mothers and children who participate in the Temporary Assistance for Needy Families program. However, a carefully selected subsample may reflect the distribution of this population well. Similarly, results regarding diet and exercise obtained from a study performed on patients at risk for heart disease may not be generally applicable to generally healthy individuals. In this case assumptions would have to be made about how results for the at-risk population might relate to those for the healthy population.
   Data used in empirical research rarely meet all (if any) of these criteria precisely. However, keeping these goals in mind can help you be precise about the types of questions you can and cannot answer reliably.

2. *Representativeness.* A regression model is fit to data and is used to make inferences about a larger population, hence the implicit assumption in interpreting regression coefficients is that the sample is representative of the population.
   To be more precise, the key assumption is that the data are representative of the distribution of the outcome y given the predictors $x_1, x_2, \ldots$, that are included in the model. For example, in a regression of earnings on height and sex, it would be acceptable for women and tall people to be overrepresented in the sample, compared to the general population, but problems would arise if the sample includes too many rich people. Selection on x does not interfere with inferences from the regression model, but selection on y does. This is one motivation to include more predictors in our regressions, to allow the assumption of representativeness, conditional on X, to be more reasonable.
   Representativeness is a concern even with data that would not conventionally be considered as a sample. For example, the forecasting model in Section 7.1 contains data from 16 consecutive elections, and these are not a sample from anything, but one purpose of the model is to predict future elections. Using the regression fit to past data to predict the next election is mathematically equivalent to considering the observed data and the new outcome as a random sample from a hypothetical superpopulation. Or, to be more precise, it is like treating the errors as a random sample from the normal error distribution. For another example, if a regression model is fit to data from the 50 states, we are not interested in making predictions for a hypothetical 51st state, but you may well be interested in the hypothetical outcome in the 50 states in a future year. As long as some generalization is involved, ideas of statistical sampling arise, and we need to think about representativeness of the sample to the implicit or explicit population about which inferences will be drawn. This is related to the idea of generative modeling, as discussed in Section 4.1.
3. *Additivity and linearity.* The most important mathematical assumption of the linear regression model is that its deterministic component is a linear function of the separate predictors: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots$.
   If additivity is violated, it might make sense to transform the data (for example, if $y = abc$, then $\log y = \log a + \log b + \log c$) or to add interactions. If linearity is violated, perhaps a predictor should be put in as $1/x$ or $\log(x)$ instead of simply linearly. Or a more complicated relationship could be expressed using a nonlinear function such as a spline or Gaussian process, which we discuss briefly in Chapter 22.
   For example, in medical and public health examples, a nonlinear function can allow a health measure to decline with higher ages, with the rate of decline becoming steeper as age increases. In political examples, including non-linear function allows the possibility of increasing slopes with age and also U-shaped patterns if, for example, the young and old favor taxes more than the middle-aged.
   Just about every problem we ever study will be nonlinear, but linear regression can still be useful in estimating an average relationship, for example as discussed in Section 8.3. As we get more data, and for problems where nonlinearity is of interest, it will make more sense to put in the effort to model nonlinearity. And for some problems a linear model would not make sense at all; for example, see Exercise 1.9.
4. *Independence of errors.* The simple regression model assumes that the errors from the prediction line are independent, an assumption that is violated in time series, spatial, and multilevel settings.
5. *Equal variance of errors.* Unequal error variance (also called heteroscedasticity, in contrast to equal variances, or homoscedasticity) can be an issue when a regression is used for probabilistic prediction, but it does not affect what is typically the most important aspect of a regression model, which is the information that goes into the predictors and how they are combined. If the variance of the regression errors are unequal, estimation is more efficiently performed by accounting for this in the model, as with weighted least squares discussed in Section 10.8. In most cases, however, this issue is minor.
6. *Normality of errors.* The distribution of the error term is relevant when predicting individual data

points. For the purpose of estimating the regression line (as compared to predicting individual data points), the assumption of normality is typically barely important at all. Thus we do *not* recommend diagnostics of the normality of regression residuals. For example, many textbooks recommend quantile-quantile (Q-Q) plots, in which the ordered residuals are plotted vs. the corresponding expected values of ordered draws from a normal distribution, with departures of this plot from linearity indicating nonnormality of the error term. There is nothing wrong with making such a plot, and it can be relevant when evaluating the use of the model for predicting individual data points, but we are typically more concerned with the assumptions of validity, representativeness, additivity, linearity, and so on, listed above.

If the distribution of errors is of interest, perhaps because of predictive goals, this should be distinguished from the distribution of the data, y. For example, consider a regression on a single discrete predictor, x, which takes on the values 0, 1, and 2, with one-third of the population in each category. Suppose the true regression line is $y = 0.2 + 0.5x$ with normally-distributed errors with standard deviation 0.1. Then a graph of the data y will show three fairly sharp modes centered at 0.2, 0.7, and 1.2. Other examples of such mixture distributions arise in economics, when including both employed and unemployed people, or the study of elections, when comparing districts with incumbent legislators of different parties.

The regression model does *not* assume or require that predictors be normally distributed. In addition, the normal distribution on the outcome refers to the regression errors, not to the raw data. Depending on the structure of the predictors, it is possible for data y to be far from normally distributed even when coming from a linear regression model.

Failures of the assumptions

What do we do when the assumptions break down?

Most directly, one can extend the model, for example, with measurement error models to address problems with validity, selection models to handle nonrepresentative data, nonadditive and nonlinear models, correlated errors or latent variables to capture violations of the independence assumption, and models for varying variances and nonnormal errors.

Other times, it is simpler to change the data or model so the assumptions are more reasonable: these steps could include obtaining cleaner data, adding predictors to make representativeness assumptions more reasonable, adding interactions to capture nonlinearity, and transforming predictors and outcomes so that an additive model can make more sense.

Alternatively, one can change or restrict the questions to align them closer to the data, making conclusions that are more descriptive and less causal or extrapolative, defining the population to match the sample, and predicting averages rather than individual cases. In practice, we typically meet in the middle, applying some mix of model expansion, data processing, and care in extrapolation beyond the data.

Causal inference

Further assumptions are necessary if a regression coefficient is to be given a causal interpretation, as we discuss in Part 4 of this book. From a regression context, causal inference can be considered as a form of prediction in which we are interested in what would happen if various predictors were set to particular values. This relates to the common, but often mistaken, interpretation of a regression coefficient as "the effect of a variable with all else held constant." To see the fundamental error in automatically giving a causal interpretation to a regression coefficient, consider the model predicting earnings from height, and imagine interpreting the slope as the effect of an additional inch of height. The problem here is the regression is fit to data from different people, but the causal question addresses what would happen to a single person. Strictly speaking, the slope of this regression represents the average difference in earnings, comparing two people who differ by an inch in height. (This

interpretation is not quite correct either, as it relies on the assumption of linearity, but set that aside for now.) Tall people make more money on average than short people, but that does not necessarily imply that making individual people taller would increase their earnings, even on average. Indeed, even to post this question makes it clear that it is not well defined, as we would need to consider how this (hypothetical) increase in individual heights was to be done, before considering its average effect on earnings. Similar issues arise in other fields: for example, if blood pressure correlates with some negative health outcome, this does not necessarily imply that a decrease in blood pressure will have beneficial effects. It can all depend on *how* blood pressure would be reduced—in causal terms, what is the treatment?—and, even when that is clear, assumptions need to be made for a model fit to existing data to directly apply to predictions of what would happen under an intervention.

11.2 Plotting the data and fitted model

Graphics are helpful for visualizing data, understanding models, and revealing patterns in the data not explained by fitted models. We first discuss general principles of data display in the context of linear regression and then in Section 11.3 consider plots specifically designed to reveal model misfit.

Displaying a regression line as a function of one input variable

Example: Children's IQ tests

We return to the children's test score example to demonstrate some of the details of displaying fitted regressions in R.[1] We displayed some aspects of the data in Figures 10.1–10.3. We can make a plot such as Figure 10.2 as follows:

```
fit_2 <- stan_glm(kid_score ~ mom_iq, data=kidiq)
plot(kidiq$mom_iq, kidiq$kid_score, xlab="Mother IQ score", ylab="Child test score")
abline(coef(fit_2)[1], coef(fit_2)[2])
```

The function `plot` creates the scatterplot of observations, and `abline` superimposes the line $y = \hat{a}+\hat{b}x$ using the estimated coefficients from the fitted regression.

Displaying two fitted regression lines

Model with no interaction. For the model with two predictors, we can create a graph with two sets of points and two regression lines, as in Figure 10.3:

```
fit_3 <- stan_glm(kid_score ~ mom_hs + mom_iq, data=kidiq)
colors <- ifelse(kidiq$mom_hs==1, "black", "gray")
plot(kidiq$mom_iq, kidiq$kid_score,
  xlab="Mother IQ score", ylab="Child test score", col=colors, pch=20)
b_hat <- coef(fit_3)
abline(b_hat[1] + b_hat[2], b_hat[3], col="black")
abline(b_hat[1], b_hat[3], col="gray")
```

Setting `pch=20` tells the `plot` function to display the data using small dots, and the `col` option sets the colors of the points, which we have assigned to black or gray according to the value of `mom_hs`. Finally, the calls to `abline` superimpose the fitted regression lines for the two groups defined by maternal high school completion.

If `mom_hs` were a continuous predictor here, we could display the fitted model by plotting `kid_score` vs. `mom_iq` for two different values of `mom_hs` that are in the range of the data.

Model with interaction. We can set up the same sort of plot for the model with interactions, with the only difference being that the two lines have different slopes:

[1] Data and code for this example are in the folder `KidIQ`.

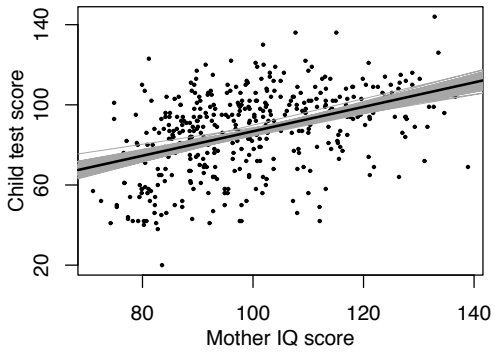

Figure 11.1 *Data and regression of child's test score on maternal IQ, with the solid line showing the fitted regression model and light lines indicating uncertainty in the fitted regression.*

```
fit_4 <- stan_glm(kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq, data=kidiq)
colors <- ifelse(kidiq$mom_hs==1, "black", "gray")
plot(kidiq$mom_iq, kidiq$kid_score,
  xlab="Mother IQ score", ylab="Child test score", col=colors, pch=20)
b_hat <- coef(fit_4)
abline(b_hat[1] + b_hat[2], b_hat[3] + b_hat[4], col="black")
abline(b_hat[1], b_hat[3], col="gray")
```

The result is shown in Figure 10.4.

Displaying uncertainty in the fitted regression

In Section 9.1 we discussed how posterior simulations represent our uncertainty in the estimated regression coefficients. Here we briefly describe how to use these simulations to display this inferential uncertainty graphically. Consider this simple model:

```
fit_2 <- stan_glm(kid_score ~ mom_iq, data=kidiq)
```

The following code creates Figure 11.1, which shows the fitted regression line along with 10 simulations representing uncertainty about the line:

```
sims_2 <- as.matrix(fit_2)
n_sims_2 <- nrow(sims_2)
beta_hat_2 <- apply(sims_2, 2, median)
plot(kidiq$mom_iq, kidiq$kid_score, xlab="Mother IQ score", ylab="Child test score")
sims_display <- sample(n_sims_2, 10)
for (i in sims_display){
  abline(sims_2[i,1], sims_2[i,2], col="gray")
}
abline(coef(fit_2)[1], coef(fit_2)[2], col="black")
```

We use the loop to display 10 different simulation draws.

Displaying using one plot for each input variable

Now consider the regression including the indicator for maternal high school completion:

```
fit_3 <- stan_glm(kid_score ~ mom_hs + mom_iq, data=kidiq)
sims_3 <- as.matrix(fit_3)
n_sims_3 <- nrow(sims_3)
```

We display this model in Figure 11.2 as two plots, one for each of the two input variables with the other held at its average value:

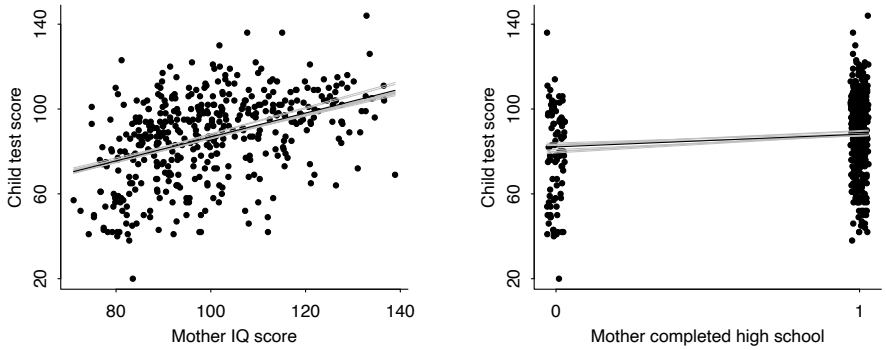

Figure 11.2 *Data and regression of child's test score on maternal IQ and high school completion, shown as a function of each of the two input variables with the other held at its average value. Light lines indicate uncertainty in the regressions. Values for mother's high school completion have been jittered to make the points more distinct.*

```
par(mfrow=c(1,2))
plot(kidiq$mom_iq, kidiq$kid_score, xlab="Mother IQ score", ylab="Child test score")
mom_hs_bar <- mean(kidiq$mom_hs)
sims_display <- sample(n_sims_3, 10)
for (i in sims_display){
  curve(cbind(1, mom_hs_bar, x) %*% sims_3[i,1:3], lwd=0.5, col="gray", add=TRUE)
}
curve(cbind(1, mom_hs_bar, x) %*% coef(fit_3), col="black", add=TRUE)

plot(kidiq$mom_hs, kidiq$kid_score, xlab="Mother completed high school",
  ylab="Child test score")
mom_iq_bar <- mean(kidiq$mom_iq)
for (i in sims_display){
  curve(cbind(1, x, mom_iq_bar) %*% sims_3[i,1:3], lwd=0.5, col="gray", add=TRUE)
}
curve(cbind(1, x, mom_iq_bar) %*% coef(fit_3), col="black", add=TRUE)
```

This example demonstrates how we can display a fitted model using multiple plots, one for each predictor.

Plotting the outcome vs. a continuous predictor

Consider the following common situation: a continuous outcome y is modeled given a treatment indicator z and a continuous pre-treatment predictor x:

$$y = a + bx + \theta z + \text{error}.$$

Example: Simulated regression plots

We simulate some fake data:[2]

```
N <- 100
x <- runif(N, 0, 1)
z <- sample(c(0, 1), N, replace=TRUE)
a <- 1
b <- 2
theta <- 5
sigma <- 2
y <- a + b*x + theta*z +  rnorm(N, 0, sigma)
fake <- data.frame(x=x, y=y, z=z)
```

[2]Code for this example is in the folder `Residuals`.

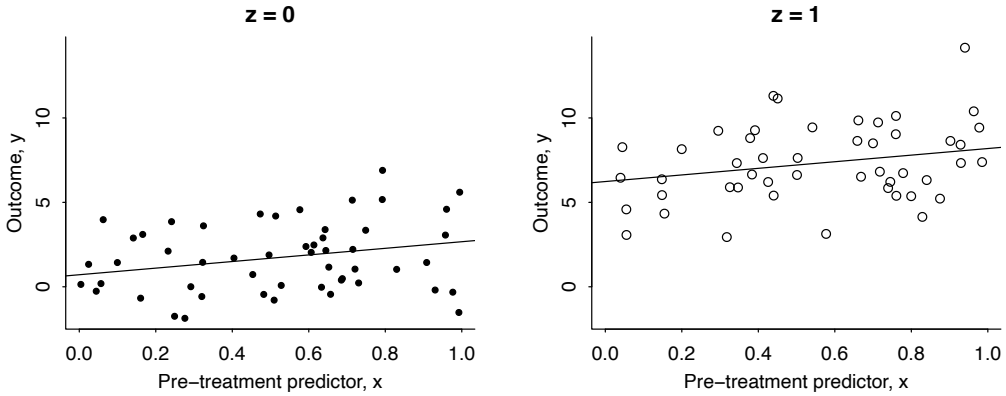

Figure 11.3 *From a hypothetical study of a pre-treatment variable x, treatment indicator z, and outcome y: data and fitted regression line plotted separately for control and treatment groups.*

We then fit a regression and plot the data and fitted model:

```
fit <- stan_glm(y ~ x + z, data=fake)
par(mfrow=c(1,2))
for (i in 0:1){
  plot(range(x), range(y), type="n", main=paste("z =", i))
  points(x[z==i], y[z==i], pch=20+i)
  abline(coef(fit)["(Intercept)"] + coef(fit)["z"]*i, coef(fit)["x"])
}
```

Figure 11.3 shows the result. The key here is to recognize that x is continuous and z is discrete, hence it makes sense to make graphs showing y vs. x for different values of z.

Forming a linear predictor from a multiple regression

Now extend the previous example so that the treatment indicator z is accompanied by K pre-treatment predictors x_k, $k = 1, \ldots, K$:

$$y = b_0 + b_1 x_1 + \cdots + b_K x_K + \theta z + \text{error}.$$

How should we graph these data?

One option is to graph y vs. x_k, for each k, holding each of the other pre-treatment predictors at its average, again making two graphs for $z = 0$ and $z = 1$ as in Figure 11.3. For each k, then, the data (x_{ki}, y_i) would be plotted along with the line, $y = \hat{\mu} + \hat{b}_k x_k$ (corresponding to $z = 0$) or $y = (\hat{\mu} + \hat{\theta}) + \hat{b}_k x_k$ (for $z = 1$), where $\hat{\mu} = (\hat{b}_0 + \hat{b}_1 \bar{x}_1 + \ldots \hat{b}_K \bar{x}_K) - \hat{b}_k \bar{x}_k$, and in total this would yield a $K \times 2$ grid of plots, with each row corresponding to a different predictor x_k.

Another approach is to plot the outcome, y, against the *linear predictor*, $\hat{y} = \hat{b}_0 + \hat{b}_1 x_1 + \cdots + \hat{b}_K x_K + \hat{\theta} z$, separately for the control ($z = 0$) and treatment ($z = 1$) groups. In either case, the linear predictor can be viewed as a unidimensional summary of the pre-treatment information.

Here is an example. First we simulate some fake data from a regression with $K = 10$ predictors and a constant term:

```
N <- 100
K <- 10
X <- array(runif(N*K, 0, 1), c(N, K))
z <- sample(c(0, 1), N, replace=TRUE)
a <- 1
b <- 1:K
theta <- 5
```

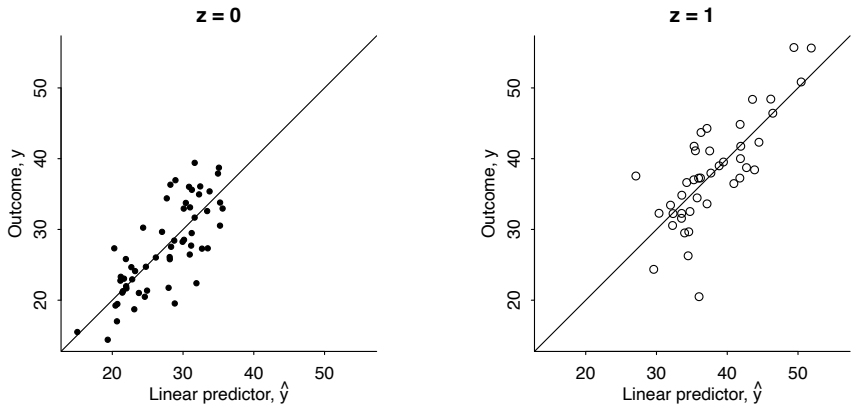

Figure 11.4 *From a hypothetical study of 10 pre-treatment variables X, treatment indicator z, and outcome y: Outcome plotted vs. fitted linear predictor $\hat{y}$ (so that the fitted regression line is by definition $y = \hat{y}$), plotted separately for control and treatment groups.*

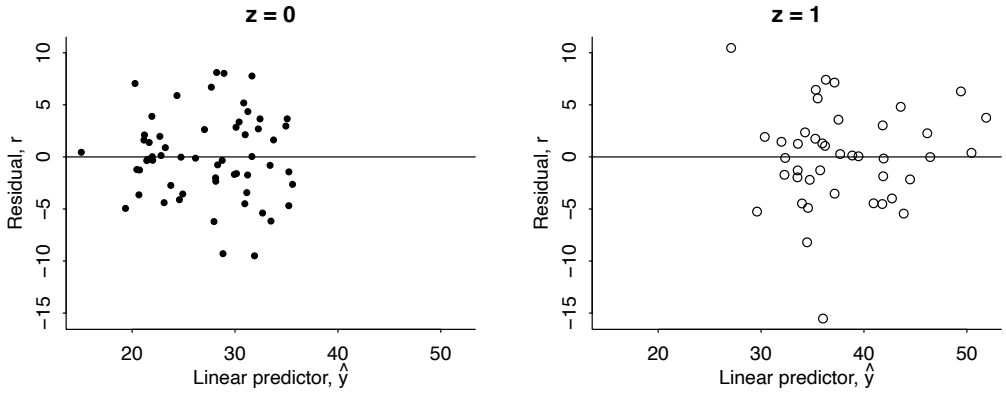

Figure 11.5 *Continuing Figure 11.4, the residual, $r = y - \hat{y}$, vs. the fitted linear predictor $\hat{y}$, plotted separately for control and treatment groups.*

```
sigma <- 2
y <- a + X %*% b + theta*z +  rnorm(N, 0, sigma)
fake <- data.frame(X=X, y=y, z=z)
fit <- stan_glm(y ~ X + z, data=fake)
```

Now we compute the linear predictor based on the point estimate of the model:

```
y_hat <- predict(fit)
```

And now we are ready to plot the data vs. the linear predictor for each value of z:

```
par(mfrow=c(1,2))
for (i in 0:1){
  plot(range(y_hat,y), range(y_hat,y), type="n", main=paste("z =", i))
  points(y_hat[z==i], y[z==i], pch=20+i)
  abline(0, 1)
}
```

Figure 11.4 shows the results. Because u is defined as the linear predictor, E(y) (the expected or predicted average value of the data given the estimated coefficients and all the predictors in the model), by construction the line tracing the expectation of y given u is the 45° line on each graph.

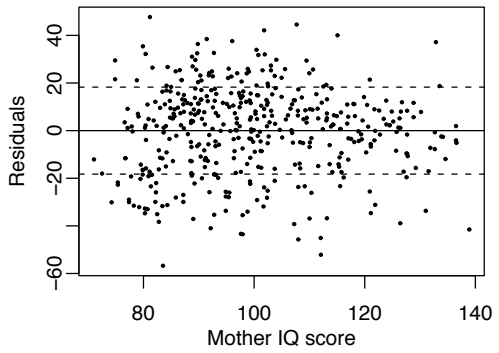

Figure 11.6 *Residual plot for child test score data when regressed on maternal IQ, with dotted lines showing* ±1 *standard-deviation bounds. The residuals show no striking patterns.*

11.3 Residual plots

Once we have plotted the data and fitted regression lines, we can evaluate this fit by looking at the difference between data and their expectations: the *residuals*,

$$r_i = y_i - X_i\hat{\beta}.$$

In the example concluding the previous section, this is $r_i = y_i - (\hat{b}_0 + \hat{b}_1 x_{i1} + \cdots + \hat{b}_K x_{iK} + \hat{\theta} z_i)$.

One advantage of plotting residuals is that, if the model is correct, they should look roughly randomly scattered in comparison to a horizontal line, which can be visually more clear than comparing to a fitted line. Figure 11.5 illustrates.

Figure 11.6 shows a residual plot for the test scores example where child's test score is regressed simply on mother's IQ.[3] The plot looks fine in that there do not appear to be any strong patterns. In other settings, residual plots can reveal systematic problems with model fit, as is illustrated, for example, in Chapter 15. Plots of residuals and binned residuals can be seen as visual comparisons to the hypothesis that the errors from a model are independent with zero mean.

Using fake-data simulation to understand residual plots

Simulation of fake data can be used to validate statistical algorithms and to check the properties of estimation procedures. We illustrated in Section 7.2 with a simple regression, where we simulated fake data from the model, $y = X\beta + \epsilon$, re-fit the model to the simulated data, and checked the coverage of the 68% and 95% intervals for the coefficients β.

For another illustration of the power of fake data, we simulate from a regression model to get insight into residual plots, in particular, to understand why we plot residuals versus fitted values rather than versus observed values.

A confusing choice: plot residuals vs. predicted values, or residuals vs. observed values?

Example: Midterm and final exams

We illustrate with a simple model fit to real data, predicting final exam scores from midterms in an introductory statistics class:[4]

```
fit_1 <- stan_glm(final ~ midterm, data=introclass)
print(fit_1)
```

yielding,

[3] Data and code for this example are in the folder `KidIQ`.

[4] Data and code for this example are in the folder `Introclass`.

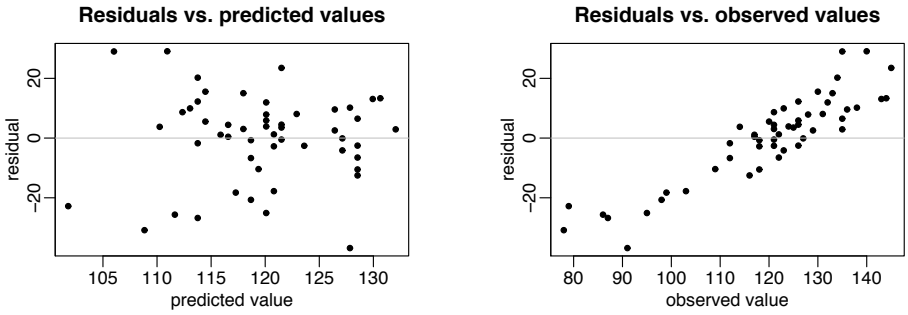

Figure 11.7 *From a model predicting final exam grades from midterms: plots of regression residuals versus predicted and versus observed values. The left plot looks reasonable but the right plot shows strong patterns. How to understand these? An exploration using fake data (see Figure 11.8) shows that, even if the model were correct, we would expect the right plot to show strong patterns. The plot of residuals versus observed thus does not indicate a problem with the model.*

```
            Median MAD_SD
(Intercept) 64.4   17.2
midterm      0.7    0.2

Auxiliary parameter(s):
      Median MAD_SD
sigma 14.9   1.5
```

We extract simulated coefficients and compute mean predictions y_i^{pred} and residuals $y_i - y_i^{\text{pred}}$:

```
sims <- as.matrix(fit_1)
predicted <- predict(fit_1)
resid <- introclass$final - predicted
```

Figure 11.7 shows the residuals from this model, plotted in two different ways: (a) residuals versus fitted values, and (b) residuals versus observed values. Figure 11.7a looks reasonable: the residuals are centered around zero for all fitted values. But Figure 11.7b looks troubling.

It turns out that the first plot is what we should be looking at, and the second plot is misleading. This can be understood using probability theory (from the regression model, the errors ϵ_i should be independent of the predictors x_i, not the data y_i) but a perhaps more convincing demonstration uses fake data, as we now illustrate.

Understanding the choice using fake-data simulation

For this example, we set the coefficients and residual standard deviation to reasonable values given the model estimates, and then we simulate fake final exam data using the real midterm as a predictor:

```
a <- 64.5
b <- 0.7
sigma <- 14.8
n <- nrow(introclass)
introclass$final_fake <- a + b*introclass$midterm + rnorm(n, 0, sigma)
```

Next we fit the regression model to the fake data and compute fitted values and residuals:

```
fit_fake <- stan_glm(final_fake ~ midterm, data=introclass)
sims <- as.matrix(fit_fake)
predicted_fake <- colMeans(sims[,1] + sims[,2] %*% t(introclass$midterm))
```

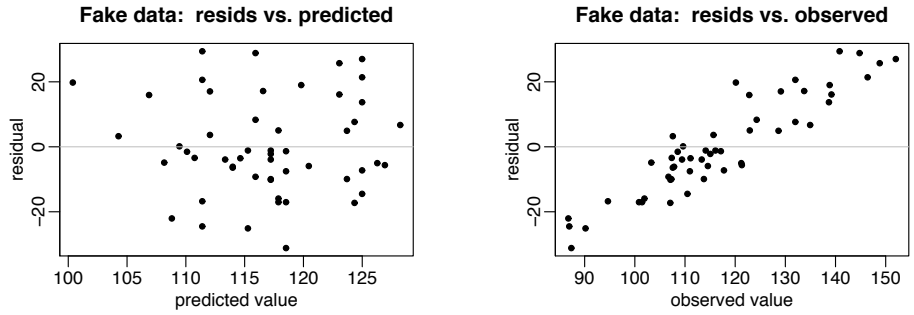

Figure 11.8 *From fake data: plots of regression residuals versus (a) predicted or (b) observed values. Data were simulated from the fitted family of regression models, and so we know that the pattern on the right does not represent any sort of model failure. This is an illustration of the use of fake data to evaluate diagnostic plots. Compare to the corresponding plots of real data in Figure 11.7.*

Figure 11.8 shows the plots of `resid_fake` versus `predicted_fake` and `final_fake`. These are the sorts of residual plots we would see *if the model were correct*. This simulation shows why we prefer to plot residuals versus predicted rather than observed values.

11.4 Comparing data to replications from a fitted model

So far we have considered several uses of simulation: exploring the implications of hypothesized probability models (Section 5.1); exploring the implications of statistical models that were fit to data (Section 10.6); and studying the properties of statistical procedures by comparing estimates to known true values of parameters (Sections 7.2 and 11.3). Here we introduce *posterior predictive checking*: simulating replicated datasets under the fitted model and then comparing these to the observed data.

Example: simulation-based checking of a fitted normal distribution

The most fundamental way to check model fit is to display replicated datasets and compare them to the actual data. Here we illustrate with a simple case from a famous historical dataset that did not fit the normal distribution. The goal of this example is to demonstrate how the lack of fit can be seen using predictive replications.

Example: Speed of light

Figure 11.9 shows the data, a set of measurements taken by Simon Newcomb in 1882 as part of an experiment to estimate the speed of light.[5] We (inappropriately) fit a normal distribution to these data, which in the regression context can be done by fitting a linear regression with no predictors. Our implicit model for linear regression is normally-distributed errors, and, as discussed in Section 7.3, estimating the mean is equivalent to regressing on a constant term:

```
fit <- stan_glm(y ~ 1, data=newcomb)
```

The next step is to simulate replications from the parameters in the fitted model (in this case, simply the constant term β_0 and the residual standard deviation σ):

```
sims <- as.matrix(fit)
n_sims <- nrow(sims)
```

We can then use these simulations to create `n_sims` fake datasets of 66 observations each:

```
n <- length(newcomb$y)
y_rep <- array(NA, c(n_sims, n))
for (s in 1:n_sims) {
```

[5] Data and code for this example are in the folder `Newcomb`.

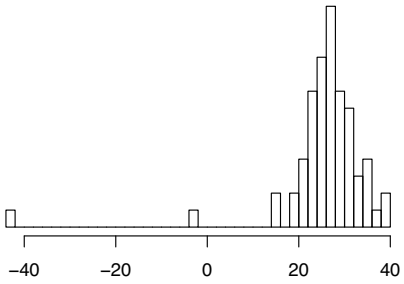

Figure 11.9 *Histogram of Simon Newcomb's measurements for estimating the speed of light, from Stigler (1977). The data represent the amount of time required for light to travel a distance of 7442 meters and are recorded as deviations from* 24 800 *nanoseconds.*

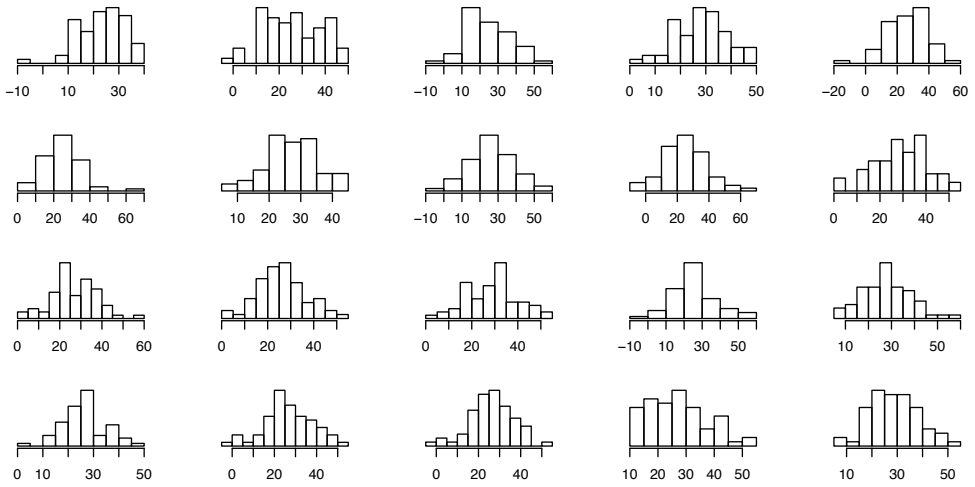

Figure 11.10 *Twenty replications, y^{rep}, of the speed-of-light data from the predictive distribution under the normal model; compare to observed data, y, in Figure 11.9. Each histogram displays the result of drawing 66 independent values y_i^{rep} from a common normal distribution with mean and standard deviation (μ, σ) simulated from the fitted model.*

```
  y_rep[s,] <- rnorm(n, sims[s,1], sims[s,2])
}
```

More simply we can just use the built-in function for posterior predictive replications:

```
y_rep <- posterior_predict(fit)
```

Visual comparison of actual and replicated datasets. Figure 11.10 shows a plot of 20 randomly sampled datasets, produced as follows:

```
par(mfrow=c(5,4))
for (s in sample(n_sims, 20)) {
  hist(y_rep[s,])
}
```

The systematic differences between data and replications are clear. Figure 11.11 shows an alternative visualization using overlaid density estimates of the data and the replicates. In more complicated problems, more effort may be needed to effectively display the data and replications for useful comparisons, but the same general idea holds.

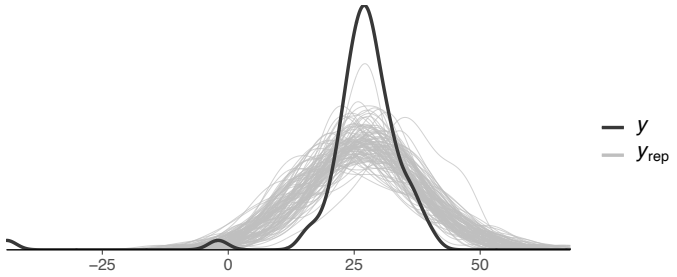

Figure 11.11 *Density estimates of the speed-of-light data y and 100 replications, y^{rep}, from the predictive distribution under the normal model. Each density estimate displays the result of original data or drawing 66 independent values y_i^{rep} from a common normal distribution with mean and standard deviation (μ, σ) simulated from the fitted model.*

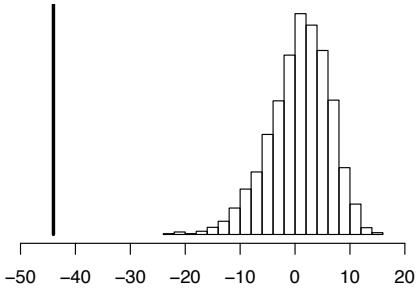

Figure 11.12 *Smallest observation of Newcomb's speed-of-light data (the vertical line at the left of the graph), compared to the smallest observations from each of 20 posterior predictive simulated datasets displayed in Figure 11.10.*

Checking model fit using a numerical data summary. Data displays can suggest more focused test statistics with which to check model fit. Here we demonstrate a simple example with the speed-of-light measurements. The graphical check in Figures 11.9 and 11.10 shows that the data have some extremely low values that do not appear in the replications. We can formalize this check by defining a *test statistic*, $T(y)$, equal to the minimum value of the data, and then calculating $T(y^{\text{rep}})$ for each of the replicated datasets:

```
test <- function(y) {
  min(y)
}
test_rep <- apply(y_rep, 1, test)
```

We then plot a histogram of the minima of the replicated datasets, with a vertical line indicating the minimum of the observed data:

```
hist(test_rep, xlim=range(test(y), test_rep))
lines(rep(test(y),2), c(0,n))
```

Figure 11.12 shows the result: the smallest observations in each of the hypothetical replications are all much larger than Newcomb's smallest observation, which is indicated by a vertical line on the graph. The normal model clearly does not capture the variation that Newcomb observed. A revised model might use an asymmetric contaminated normal distribution or a symmetric long-tailed distribution in place of the normal measurement model.

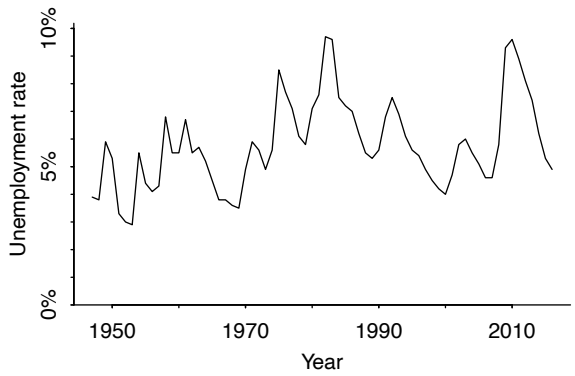

Figure 11.13 *Time series of U.S. annual unemployment rates from 1947 to 2016. We fit a first-order autoregression to these data and then simulate several datasets, shown in Figure 11.14, from the fitted model.*

11.5 Example: predictive simulation to check the fit of a time-series model

Predictive simulation is more complicated in time-series models, which are typically set up so that the distribution for each point depends on the earlier data. We illustrate with a simple autoregressive model.

Fitting a first-order autoregression to the unemployment series

Example: Unemployment time series

Figure 11.13 shows the time series of annual unemployment rates in the United States from 1947 to 2004.[6] We would like to see how well these data are fit by a first-order autoregression, that is, a regression on last year's unemployment rate. Such a model is easy to set up and fit:

```
n <- nrow(unemp)
unemp$y_lag <- c(NA, unemp$y[1:(n-1)])
fit_lag <- stan_glm(y ~ y_lag, data=unemp)
```

yielding the following fit:

```
            Median MAD_SD
(Intercept) 1.35   0.44
y_lag       0.77   0.07

Auxiliary parameter(s):
      Median MAD_SD
sigma 1.03   0.09
```

This is potentially informative but does not give a sense of the fit of model to data. To examine the fit, we will simulate replicated data from the fitted model.

Simulating replicated datasets

We can extract the simulations from the fitted model:

```
sims <- as.matrix(fit_lag)
n_sims <- nrow(sims)
```

Then we create a container for replicated datasets and fill it in with simulated time series:

[6] Data and code for this example are in the folder `Unemployment`.

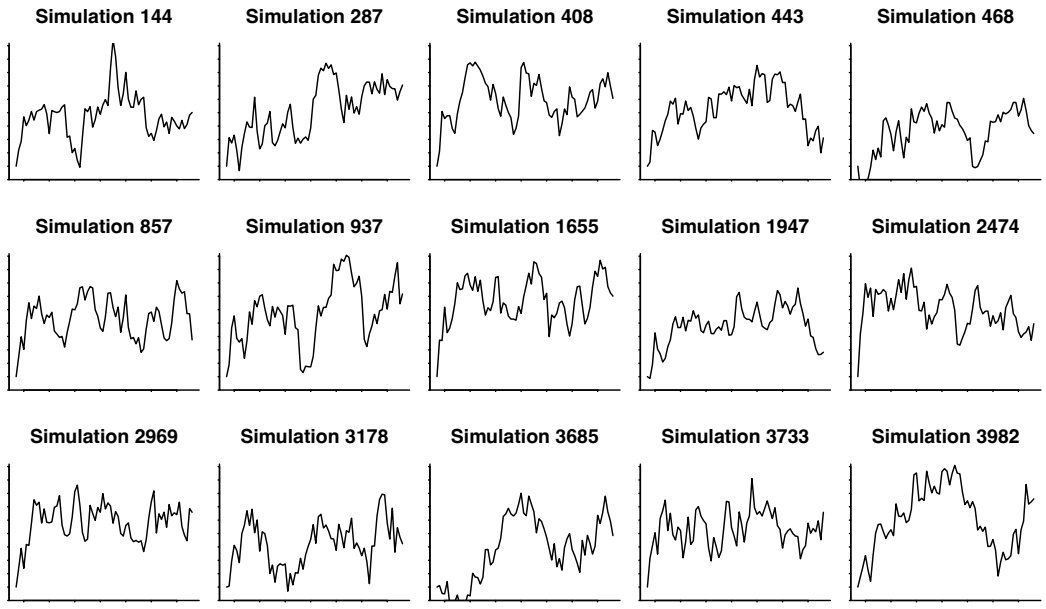

Figure 11.14 *Graphing a random sample of the simulated replications of the unemployment series from the fitted autoregressive model. The replications capture many of the features of the actual data in Figure 11.13 but show slightly more short-term variation.*

```
y_rep <- array(NA, c(n_sims, n))
for (s in 1:n_sims){
  y_rep[s,1] <- unemp$y[1]
  for (t in 2:n){
    y_rep[s,t] <- sims[s,"(Intercept)"] + sims[s,"y_lag"] * y_rep[s,t-1] +
                  rnorm(1, 0, sims[s,"sigma"])
  }
}
```

We could not simply create these simulations using `posterior_predict` because with this time-series model we need to simulate each year conditional on the last.

Visual and numerical comparisons of replicated to actual data

Our first step in model checking is to plot some simulated datasets, which we do in Figure 11.14, and compare them visually to the actual data in Figure 11.13. The 15 simulations show different patterns, with many of them capturing the broad features of the data—its range, lack of overall trend, and irregular rises and falls. This autoregressive model clearly can represent many different sorts of time-series patterns.

Looking carefully at Figure 11.14, we see one pattern in all these replicated data that was not in the original data in Figure 11.13, and that is a jaggedness, a level of short-term ups and downs that contrasts to the smoother appearance of the actual time series.

To quantify this discrepancy, we define a test statistic that is the frequency of "switches"—the number of years in which an increase in unemployment is immediately followed by a decrease, or vice versa:

```
test <- function(y){
  n <- length(y)
  y_lag <- c(NA, y[1:(n-1)])
```

```
  y_lag_2 <- c(NA, NA, y[1:(n-2)])
  sum(sign(y-y_lag) != sign(y_lag-y_lag_2), na.rm=TRUE)
}
```

As with the examples in the previous section, we compute this test for the data and for the replicated datasets:

```
test_y <- test(unemp$y)
test_rep <- apply(y_rep, 1, test)
```

The actual unemployment series featured 26 switches. In this case, 99% of the `n_sims` = 4000 replications had more than 26 switches, with 80% in the range [31, 41], implying that this aspect of the data was not captured well by the model. The point of this test is not to "reject" the autoregression—no model is perfect, after all—but rather to see that this particular aspect of the data, its smoothness, is not well captured by the fitted model. The real time series switched direction much less frequently than would be expected from the model. The test is a confirmation of our visual impression of a systematic difference between the observed and replicated datasets.

11.6 Residual standard deviation σ and explained variance R^2

The residual standard deviation, σ, summarizes the scale of the residuals $r_i = y_i - X_i\hat{\beta}$. For example, in the children's test scores example, $\hat{\sigma} = 18$, which tells us that the linear model can predict scores to about an accuracy of 18 points. Said another way, we can think of this standard deviation as a measure of the average distance each observation falls from its prediction from the model.

The magnitude of this standard deviation is more salient when compared to the total variation in the outcome variable. The fit of the model can be summarized by σ (the smaller the residual standard deviation, the better the fit) and by R^2 (also sometimes called the coefficient of determination), the fraction of variance "explained" by the model. The "unexplained" variance is σ^2, and if we label s_y as the standard deviation of the data, then, using a classic definition,

$$R^2 = 1 - \left(\hat{\sigma}^2/s_y^2\right). \tag{11.1}$$

When the model is fit using least squares, the expression above is equivalent to an alternative formula that uses explained variance directly:

$$R^2 = V_{i=1}^n \hat{y}_i / s_y^2, \tag{11.2}$$

where $\hat{y}_i = X_i\hat{\beta}$ and we are using the notation V for the sample variance:

$$V_{i=1}^n z_i = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - \bar{z})^2, \text{ for any vector } z \text{ of length } n. \tag{11.3}$$

In the model fit to test scores in Section 10.1, R^2 is a perhaps disappointing 22%. However, in a deeper sense, it is presumably a good thing that this regression has a low R^2—that is, that a child's achievement cannot be accurately predicted given only these maternal characteristics.

To understand R^2, consider two special cases in the one-predictor scenario, so that $\hat{y} = \hat{a} + \hat{b}x$.

1. First, suppose that the fitted line is nearly the same as a horizontal line at $\bar{y}$, so that $\hat{a} + \hat{b}x_i \approx \bar{y}$ for all i. In this case, each residual $y_i - (\hat{a} + \hat{b}x_i)$ is nearly identical to $y_i - \bar{y}$, so that $\sigma \approx s_y$. Hence, $R^2 \approx 0$. Thus, $R^2 \approx 0$ indicates that the regression line explains practically none of the variation in y. In this case, the estimated intercept $\hat{a}$ is approximately $\bar{y}$ and the estimated slope $\hat{b}$ is approximately 0.
2. At the other extreme, suppose that the fitted line passes through all the points nearly perfectly, so that each residual is almost 0. In this case, σ will be close to 0 and much smaller than s_y, so that $R^2 \approx 1$. Thus, $R^2 \approx 1$ indicates that the regression line explains practically all of the variation in y. In general, the closer R^2 is to 1, the happier we are with the model fit.

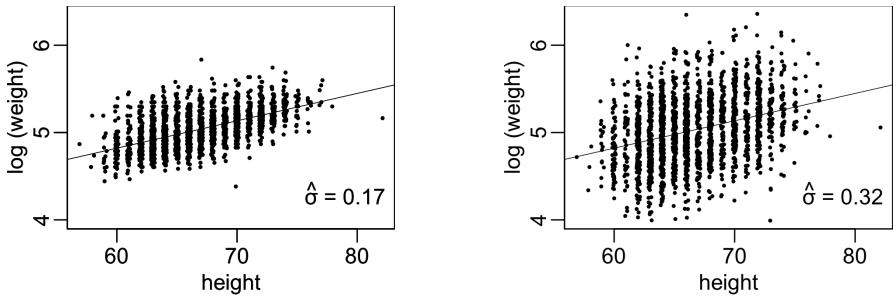

Figure 11.15 *Two hypothetical datasets with the same fitted regression line, $\hat{a} + \hat{b}x$, but different values of the residual standard deviation, σ. (a) Actual data from a survey of adults; (b) data with noise added to y.*

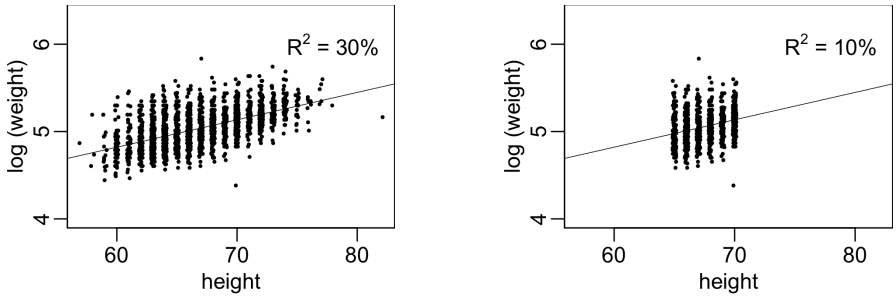

Figure 11.16 *Two hypothetical datasets with the same regression line, $\hat{a} + \hat{b}x$, and residual standard deviation, σ, but different values of the explained variance, R^2. (a) Actual data from a survey of adults; (b) data restricted to heights between 65 and 70 inches.*

There are a few things about R^2 that you should know. First, it does not change if you multiply x or y in the regression by a constant. So, if you want to change the units of the predictors or response to aid interpretation, you won't change the summary of the fit of the model to the data. Second, in a least squares regression with one predictor, one can show that R^2 equals the square of the correlation between x and y; see Exercise 11.7. There is no such interpretation for regressions with more than one predictor.

The quantity $n - k$, the number of data points minus the number of estimated coefficients, is called the *degrees of freedom* for estimating the residual errors. In classical regression, k must be less than n—otherwise, the data could be fit perfectly, and it would not be possible to estimate the regression errors at all.

Difficulties in interpreting residual standard deviation and explained variance

As we make clear throughout the book, we are generally more interested in the deterministic part of the model, $X\beta$, than in the variation, ϵ. However, when we do look at the residual standard deviation, σ, we are typically interested in it for its own sake—as a measure of the unexplained variation in the data—or because of its relevance to the precision of inferences about the regression coefficients β. Figure 11.15 illustrates two regressions with the same deterministic model, $\hat{a} + \hat{b}x$, but different values of σ.

Interpreting the proportion of explained variance, R^2, can be tricky because its numerator and denominator can be changed in different ways. Figure 11.16 illustrates with an example where the regression model is identical, but R^2 decreases because the model is estimated on a subset of the data. Going from the left to right plots in the figure, the residual standard deviation σ is unchanged but the standard deviation of the raw data, s_y, decreases when we restrict to this subset; thus, $R^2 = 1 - \hat{\sigma}^2/s_y^2$

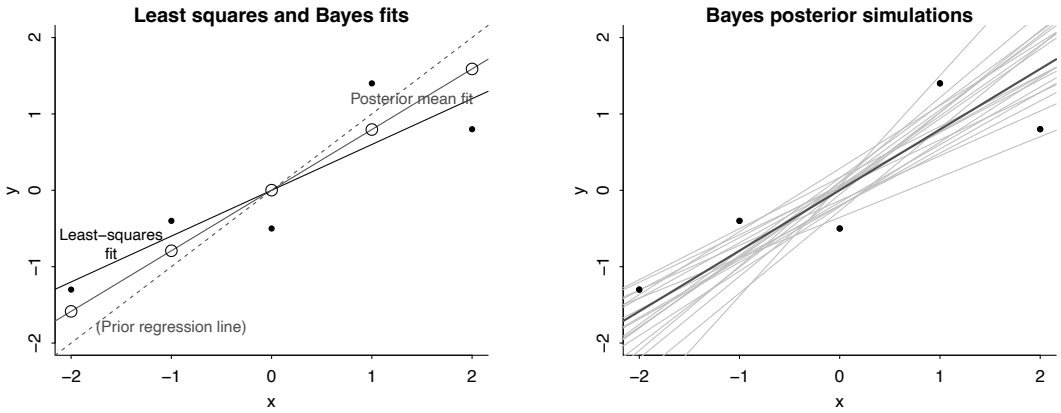

Figure 11.17 *Simple example showing the challenge of defining R^2 for a fitted Bayesian model. (a) Data, least squares regression line, and fitted Bayes line, which is a compromise between the prior and the least squares fit. The standard deviation of the fitted values from the Bayes model (the open circles on the line) is greater than the standard deviation of the data, so the usual definition of R^2 will not work. (b) Posterior mean fitted regression line along with 20 draws of the line from the posterior distribution. We compute Bayesian R^2 by evaluating formula (11.5) for each posterior simulation draw and then taking the median.*

declines. Even though R^2 is much lower in Figure 11.16b, the model fits the data just as well as in Figure 11.16a.

11.7 External validation: checking fitted model on new data

The most fundamental way to test a model is to use it to make predictions and then compare to actual data. Figure 11.19 illustrates with the children's test score model, which was fit to data collected from children who were born before 1987. Having fit the model using `stan_glm`, we then use `posterior_predict` to obtain simulations representing the predictive distribution for new cases.

We apply the model to predict the outcomes for children born in 1987 or later. This is not an ideal example for prediction, because we would not necessarily expect the model for the older children to be appropriate for the younger children, even though tests for all children were taken at age 3 or 4. However, we can use it to demonstrate the methods for computing and evaluating predictions.

The new data, y^{new}, are the outcomes for the 336 new children predicted from `mom_iq` and `mom_hs`, using the model fit using the data from the older children. Figure 11.19a plots actual values y_i^{new} versus predicted values $X_i^{\text{new}}\hat{\beta}$, and Figure 11.19b plots residuals versus predicted values with dotted lines at $\pm\,\hat{\sigma}$ (approximate 68% error bounds; see Section 3.5). The error plot shows no obvious

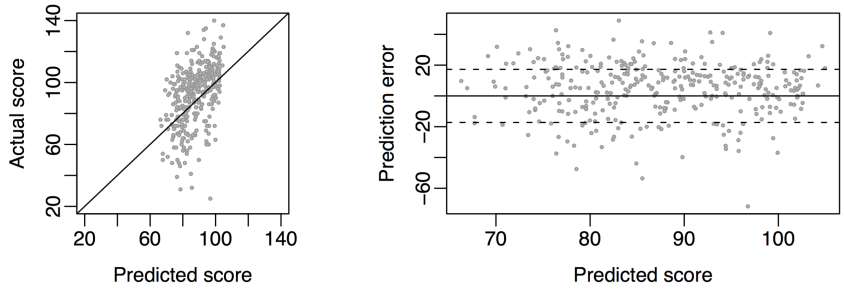

Figure 11.19 *Plots assessing how well the model fit to older children works in making predictions for younger children: (a) comparing predictions for younger children from a model against their actual values, (b) comparing residuals from these predictions against the predicted values.*

problems with applying the older-child model to the younger children, though from the scale we detect that the predictions have wide variability. In this example, what we are plotting as the "predicted score" for each child is simply the mean of the `n_sims` predictive simulation draws. We could also check coverage of the predictions by looking, for example, at how many of the actual data points fell within the 50% predictive intervals computed from the posterior predictive simulations.

Even if we had detected clear problems with these predictions, this would not mean necessarily that there is anything wrong with the model as fit to the original dataset. However, we would need to understand it further before generalizing to other children.

11.8 Cross validation

Often we would like to evaluate and compare models without waiting for new data. One can simply evaluate predictions on the observed data. But since these data have already been used to fit the model parameters, these predictions are optimistic for assessing generalization.

In cross validation, part of the data is used to fit the model and the rest of the data—the *hold-out set*—is used as a proxy for future data. When there is no natural prediction task for future data, we can think of cross validation as a way to assess generalization from one part of the data to another part.

Different data partitions can be used, depending on the modeling task. We can hold out individual observations (*leave-one-out (LOO)* cross validation) or groups of observations (*leave-one-group-out*), or use past data to predict future observations (*leave-future-out*). Leave-one-group-out and leave-future-out cross validation can be useful in multilevel and time series settings, which are beyond the scope of this book. In any form of cross validation, the model is re-fit leaving out one part of the data and then the prediction for the held-out part is evaluated. Cross validation removes the overfitting problem arising from using the same data for estimation and evaluation, but at the cost of requiring the model to be fit as many times as the number of partitions.

Leave-one-out cross validation

The naive implementation of LOO would require the model to be fit n times, once for each held-out data point. The good news is that we have built an R package, `loo`, containing a function, `loo`, with a computational shortcut that uses probability calculations to approximate the leave-one-out evaluation, while only fitting the model once, and also accounting for posterior uncertainty in the fit.

We first illustrate LOO cross validation with a small simulated dataset:[8]

```
n <- 20
x <- 1:n
a <- 0.2
```

[8]Code for this example is in the folder `CrossValidation`.

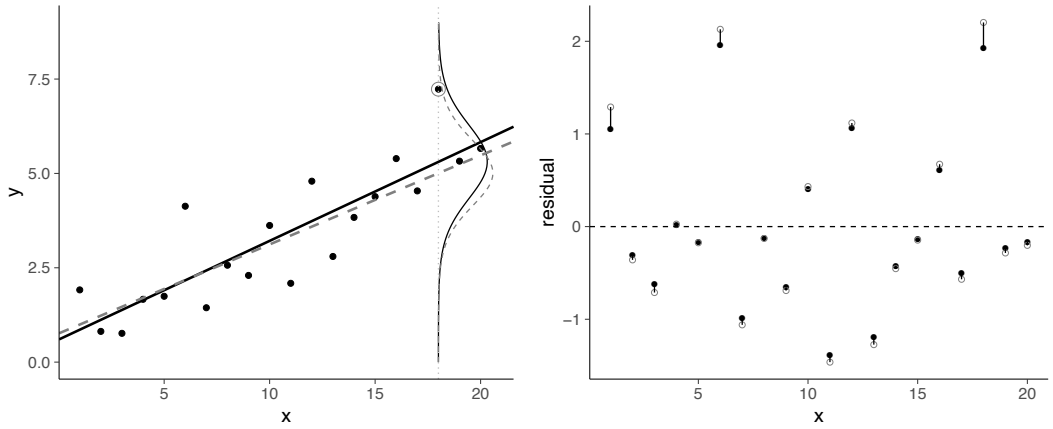

Figure 11.20 *(a) Simulated data, along with the posterior mean and predictive distribution for the* 18^{th} *data point from fitted linear regressions. The solid curve shows the posterior predictive distribution for* y_{18} *given the line fit to all the data, and the dashed line shows the posterior predictive distribution given the line fit to all the data except the* 18^{th} *observation. (b) Residuals from the fitted models. Solid circles show residuals from the model fit to all the data; open circles show leave-one-out residuals, where each observation is compared to its expected value from the model holding that data point out of the fit.*

```
b <- 0.3
sigma <- 1
set.seed(2141)
y <- a + b*x + sigma*rnorm(n)
fake <- data.frame(x, y)
```

We use the function `set.seed` to start the random number generator at a fixed value. The particular value chosen is arbitrary, but now we can inspect the data and choose a point with a large residual to see where cross validation can make a difference.

For this particular simulated dataset we notice that the 18^{th} data point is far from the line, and we demonstrate the principle of cross validation by fitting two regression models, one fit to all the data and one fit to the data excluding the 18^{th} observation:

```
fit_all <- stan_glm(y ~ x, data = fake)
fit_minus_18 <- stan_glm(y ~ x, data = fake[-18,])
```

We next evaluate the prediction for the 18^{th} data point. The posterior predictive distribution is obtained by averaging over all possible parameter values. Using posterior simulation draws, the posterior predictive distribution for a new data point y^{new} with predictors x^{new}, conditional on the observed data y and X, can be approximated in probability notation as,

$$p(y^{\text{new}}|x^{\text{new}}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y^{\text{new}}|x^{\text{new}}, \beta^s, \sigma^s). \tag{11.6}$$

Figure 11.20a shows our simulated data, along with the posterior mean and predictive distribution for y_{18} from two linear regressions, one fit to all the data and the other fit to the data excluding the held-out data point. Figure 11.20a also shows the fitted regression lines, revealing that the fit when excluding the data point is different from the fit from all the data. When a data point is held out, the posterior mean and predictive distribution move away from that observation.

In LOO cross validation, the process of leaving out one observation is repeated for all the data points. Figure 11.20b shows posterior and LOO residuals computed as the observation minus either the posterior mean predicted value (that is, the result from `predict`) applied to the fitted model, or the predicted mean from the fitted LOO model excluding the held-out data point (that is, the result

from `loo_predict`). In this example, the LOO residuals all have larger magnitudes, reflecting that it is more difficult to predict something that was not used in model fitting.

To further compare posterior residuals to LOO residuals, we can examine the standard deviation of the residuals. In our simulation, the residual standard deviation was set to 1, the LOO residual standard deviation is 1.01, and the posterior residual standard deviation is 0.92; this is a typical example in which the direct residuals show some overfitting.

The connection of residual standard deviation to explained variance R^2 was discussed in Section 11.6. Analogously to the explained proportion of variance, R^2, we can calculate LOO R^2, where the residual standard deviation is calculated from the LOO residuals. For the above simulation example, R^2 is 0.74 and LOO R^2 is 0.68.

Fast leave-one-out cross validation

LOO cross validation could be computed by fitting the model n times, once with each data point excluded. But this can be time consuming for large n. The `loo` function uses a shortcut that makes use of the mathematics of Bayesian inference, where the posterior distribution can be written as the prior distribution multiplied by the likelihood. If the observations are conditionally independent (as is typically the case in regression models), the likelihood is the product of n factors, one for each data point: in Bayesian notation, the posterior distribution is $p(\theta|y) \propto p(\theta) \prod_{i=1}^{n} p(y_i|\theta)$, where θ represents all the parameters in the model and conditioning on the predictors X is implicit.

In leave-one-out cross validation, we want to perform inferences excluding each data point i, one at a time. In the above expression, excluding that one data point is equivalent to multiplying the posterior distribution by the factor $1/p(y_i|\theta)$. The LOO posterior excluding point i is written as $p(\theta|y_{-i}) = p(\theta|y)/p(y_i|\theta)$, and the LOO distribution is computed by taking the posterior simulations for θ obtained from `stan_glm` and giving each simulation a weight of $1/p(y_i|\theta)$. This set of weighted simulations is used to approximate the predictive distribution of y_i, the held-out data point. Using the raw weights can be noisy and so the `loo` function smooths them before doing the computation.

Summarizing prediction error using the log score and deviance

There are various ways to assess the accuracy of a model's predictions. Residual standard deviation or R^2 are interpretable summaries for linear regression but do not always make sense for logistic and other discrete-data models. In addition, residual standard deviation and R^2 measure only the prediction error relative to the mean of the predictive distribution, ignoring the uncertainty represented by the predictive distribution. We should be able to do better using the entire predictive distribution.

A summary that applies to probability models more generally is the *log score*, which is defined as the logarithm of the probability or density of outcome y given predictors x. For linear regression with normally-distributed errors, the density function, $p(y_i|\beta, \sigma)$, is $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(y_i - X_i\beta)^2\right)$ (see equation (8.7)), and so the log score is $-\log\sigma - \frac{1}{2\sigma^2}(y_i - X_i\beta)^2$. Strictly speaking, this last expression should also include a term of $-\frac{1}{2}\log(2\pi)$, but typically we only look at differences in the log score, comparing models, so any constant shift can be ignored. For the purpose of this book, you will not need this formula, as the log score is computed automatically by `stan_glm`, but we include it here to connect practice to theory.

Figure 11.21 shows log posterior predictive densities and log LOO predictive densities for our simulated linear regression example. To summarize the predictive performance the usual practice is to add these log densities; the result is called the *expected log predictive density* (elpd).

The within-sample log score multiplied by -2 is called the *deviance*—for historical reasons, this has been a commonly used scale when using maximum likelihood inference—and it is reported when displaying a `glm` fit in R. Better fits to the data corresponds to higher values of the log score and lower values of the deviance for the fitted model.

More generally, the log score is also useful for different non-normal error distributions and can be

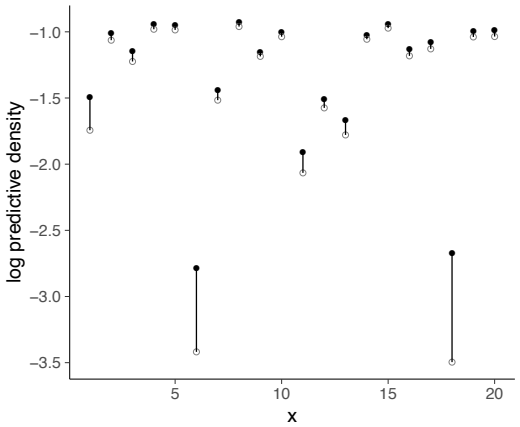

Figure 11.21 *Log posterior predictive densities and log LOO predictive densities from a regression model fitted to the data shown in Figure 11.20a. The solid circles represent log predictive densities relative to the model fit to all the data, and the open circles correspond to leave-one-out fits. The open circles are always lower than the solid circles, because removing an observation from the likelihood causes the model to fit less well to that data point.*

justified when there is interest in the accuracy of the whole predictive distribution instead of just some point estimates. We discuss the log score further in Section 13.5 in the context of logistic regression.

Overfitting and AIC

As discussed above, cross validation avoids the overfitting problem that arises from simple calculations of within-sample predictive error that use the same data to fit and evaluate the model. A simple alternative way to correct for overfitting is to adjust the log score to account for the number of parameters fitted in the model. For maximum likelihood models, this corresponds to the *Akaike information criterion* (AIC) and is defined as AIC = deviance + $2k$, where k is the number of coefficients fit in the model. AIC is an estimate of the deviance that would be expected if the fitted model is used to predict new data. Deviance is -2 times the log score, so AIC is equivalent to subtracting k from the log score to account for overfitting. The correction in AIC for number of parameters has a mathematical connection to the degrees-of-freedom adjustment in which $n - k$ rather than n is used in the denominator of formula (10.5) for $\hat{\sigma}$. We prefer to use cross validation as it is more general and directly interpretable as an estimate of prediction error, but for simple models the two methods should give similar answers.

Interpreting differences in log scores

The log score is rarely given a direct interpretation; almost always we look at differences in the log score, comparing two or more models. The reference point is that adding a linear predictor that is pure noise and with a weak prior on the coefficient should increase the log score of the fitted by 0.5, in expectation, but should result in an expected *decrease* of 0.5 in the LOO log score. This difference is additive if more predictors are included in the model. For example, adding 10 noise predictors should result in an increase of approximately 5 in the raw log score and a decrease of approximately 5 in the LOO log score. The log score relative to the full model increases, but this is an illusion due to overfitting; the LOO log score decreases because the inclusion of noise parameters really does, on average, degrade the fit.

Changes in the LOO log score can thus be calibrated in terms of the decrease in predictive accuracy arising from adding noise predictors to the model. That said, these log scores are themselves

directly computed from data and can be noisy, hence we should also be aware of our uncertainty in these calculations, as we illustrate next with an example.

Demonstration of adding pure noise predictors to a model

Example: Regression with noise predictors

When predictors are added to a model—even if the predictors are pure noise—the fit to data and within-sample predictive measures will generally improve. The ability of model to fit to the data is desired in modeling, but when the model is fit using finite data the model will fit partly also to random noise. We want to be able to separate which parts of the data allow us to generalize to new data or from one part of the data to the rest of the data and which part of the data is random noise which we should not be able to predict.

We demonstrate with an example. In Chapter 10 we introduced a model predicting children's test scores given maternal high school indicator and maternal IQ[9]. Here is the fitted model:

```
            Median MAD_SD
(Intercept) 25.8    5.8
mom_hs       5.9    2.2
mom_iq       0.6    0.1

Auxiliary parameter(s):
      Median MAD_SD
sigma 18.1   0.6
```

We then add five pure noise predictors:

```
n <- nrow(kidiq)
kidiqr <- kidiq
kidiqr$noise <- array(rnorm(5*n), c(n,5))
fit_3n <- stan_glm(kid_score ~ mom_hs + mom_iq + noise, data=kidiqr)
```

And here is the new result:

```
            Median MAD_SD
(Intercept) 25.4    5.8
mom_hs       5.9    2.3
mom_iq       0.6    0.1
noise1      -0.1    0.9
noise2      -1.2    0.9
noise3       0.1    1.0
noise4       0.2    0.9
noise5      -0.5    0.9

Auxiliary parameter(s):
      Median MAD_SD
sigma 18.1   0.6
```

Unsurprisingly, the estimated coefficients for these new predictors are statistically indistinguishable from zero. But the fit to data has improved: R^2 has gone from 0.21 to 0.22, an increase of 0.01, and the *posterior predictive log score* has gone from -1872.0 to -1871.1, an increase of 0.9. This pattern, that fit to data improves even with noise predictors, is an example of overfitting, and it is one reason we have to be careful when throwing predictors into a regression. Using LOO cross validation we see that the predictive performance for new data or generalization capability has not improved: LOO R^2 has gone from 0.2 to 0.19, a decrease of 0.01, and the LOO predictive log score has gone from -1876.1 to -1879.9, a decrease of 3.7.

In this simple example, the estimate of `elpd_loo`, -1876.1, is nearly the same as what is obtained by subtracting the number of additional parameters from the log score: $-1872 - 4 = -1876$. For

[9]Data and code for this example are in the folder `KidIQ`.

the model with five additional pure noise predictors, the estimate of elpd_loo is −1879.9, which is close to this: log score − k = −1871.1 − 9 = −1880. For more complicated problems, though, the two estimates can differ, and when that occurs, we prefer the LOO estimate, as in general it better accounts for uncertainty.

In this example, a default weak prior was used which makes it more likely that overfitting can happen when pure noise predictors are added. It is possible to reduce the overfitting by using more informative priors, but this is beyond the scope of this book.

Having fit the model (see p. 156), we use the loo to compute the leave-one-out cross validation log score:

```
loo_3 <- loo(fit_3)
print(loo_3)
```

which yields,

```
Computed from 4000 by 434 log-likelihood matrix
         Estimate   SE
elpd_loo  -1876.1 14.2
p_loo         4.1  0.4
looic      3752.3 28.4
------
Monte Carlo SE of elpd_loo is 0.0.
All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

Here is how we interpret all this:

- The "log-likelihood matrix" is log $p(y_i|x_i, \beta, \sigma)$ at n = 434 data points (x_i, y_i) using 4000 draws of β and σ from the posterior distribution.
- elpd_loo is the estimated log score along with a standard error representing uncertainty due to using only 434 data points.
- p_loo is the estimated "effective number of parameters" in the model. The above model has 4 parameters, so it makes sense that p_loo is close to 4 here, but in the case of models with stronger prior information, weaker data, or model misspecification, this direct identification will not always work.
- looic is the LOO information criterion, −2 elpd_loo, which we compute for comparability to deviance.
- The Monte Carlo SE of elpd_loo gives the uncertainty of the estimate based on the posterior simulations. If the number of simulations is large enough, this Monte Carlo SE will approach zero, in contrast to the standard errors of epld_loo, p_loo, and looic, whose uncertainties drive from the finite number of data points in the regression.
- The two lines at the end of the display give a warning if the Pareto k estimates are unstable, in which case an alternative computation is recommended, as we discuss on page 178.

We continue the example by performing a model comparison. Let's take the model above (without pure noise predictors) and compare it to the model using only the maternal high school indicator:

```
fit_1 <- stan_glm(kid_score ~ mom_hs, data=kidiq)
loo_1 <- loo(fit_1)
print(loo_1)
```

Here is what we get from the LOO computation:

```
Computed from 4000 by 434 log-likelihood matrix
         Estimate   SE
elpd_loo  -1914.9 13.8
p_loo         3.2  0.3
looic      3829.8 27.6
```

```
------
Monte Carlo SE of elpd_loo is 0.0.
All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

The simpler model performs worse in cross validation: elpd_{loo} has decreased from -1876.1 to -1914.9, a decline of 38.8.

To get a sense of the uncertainty in this improvement for future data, we need to compare the two models directly for each data point, which can be done using the `loo_compare` function:

```
loo_compare(loo_3, loo_1)
```

which yields,

```
Model comparison:
(negative 'elpd_diff' favors 1st model, positive favors 2nd)
elpd_diff        se
    -38.8       8.3
```

The difference of 38.8 has a standard error of only 8.3. When the difference (`elpd_diff`) is larger than 4, the number of observations is larger than 100, and the model is not badly misspecified then the standard error (`se_diff`) is a reliable measure of the uncertainty in the difference. Differences smaller than 4 are hard to distinguish from noise.

We can continue with other models. For example:

```
fit_4 <- stan_glm(kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq, data=kidiq)
loo_4 <- loo(fit_4)
loo_compare(loo_3, loo_4)
```

which yields,

```
Model comparison:
(negative 'elpd_diff' favors 1st model, positive favors 2nd)
elpd_diff        se
      3.6       2.6
```

There is no clear predictive advantage in adding an interaction between maternal high school indicator and maternal IQ.

To take into account the accuracy of the whole predictive distribution, we recommend the log score for model comparison in general. With care, log densities and log probabilities can be interpreted when transformed to densities and probabilities, but it can also be useful to compute more familiar measures of predictive performance such as R^2 to get an understanding of the fit of a model. For example, we can compute LOO R^2 for the above models. For model `fit_1`, the LOO R^2 is 0.05, a decrease of 0.16 with standard error 0.04 when compared to model `fit_3`. For model `fit_4`, the LOO R^2 is 0.22, an increase of 0.01 with standard error 0.05 when compared to model `fit_3`.

K-fold cross validation

As discussed earlier, leave-one-out cross validation in theory requires re-fitting the model n times, once with each data point left out, which can take too long for a dataset of even moderate size. The `loo` function has a computationally-efficient approximation that requires the model to be fit only once, but in some cases it can be unstable. Difficulties arise when the probability densities $p(y_i|\theta)$ take on very low values—if this density is near zero, then the resulting weight, $1/p(y_i|\theta)$, becomes very high. A data point for which $p(y_i|\theta)$ near zero can be thought of as an outlier in probabilistic sense, in that it is a data point from an area that has low probability under the fitted model.

When leave-one-out cross validation is unstable—which would be indicated by a warning message after running `loo`—you can switch to *K-fold cross validation* in which the data are randomly partitioned into K subsets, and the fit of each subset is evaluated based on a model fit to the rest of the data. This requires only K new model fits. It is conventional to use $K = 10$, which represents a

tradeoff between K being too low (in which case the estimate has larger bias) or too high (and then the procedure becomes too computationally expensive).

Demonstration of *K*-fold cross validation using simulated data

Example: Simulated data for cross validation

We illustrate 10-fold cross validation with another fake-data example. We start by simulating a 60×30 matrix representing 30 predictors that are random but not independent; rather, we draw them from a multivariate normal distribution with correlations 0.8, using the `mvrnorm` function, which is in the `MASS` package in R:[10]

```
library("MASS")
k <- 30
rho <- 0.8
Sigma <- rho*array(1, c(k,k)) + (1-rho)*diag(k)
X <- mvrnorm(n, rep(0,k), Sigma)
```

We simulate y from the model $X\beta$ + error with the three first coefficients $\beta_1, \beta_2, \beta_3$ set somewhat arbitrarily to $(-1, 1, 2)$ and the rest set to zero, and with independent errors with mean 0 and standard deviation 2:

```
b <- c(c(-1,1,2), rep(0, k-3))
y <- X %*% b + 2*rnorm(n)
fake <- data.frame(X, y)
```

We then fit linear regression using a weak prior distribution:

```
fit_1 <- stan_glm(y ~ ., prior=normal(0, 10), data=fake)
loo_1 <- loo(fit_1)
```

Now `loo(fit_1)` returns the following warning:

```
1: Found 14 observations with a pareto_k > 0.7.
With this many problematic observations we recommend calling 'kfold' with argument
'K=10' to perform 10-fold cross validation rather than LOO.
```

So we follow that advice:

```
kfold_1 <- kfold(fit_1, K=10)
print(kfold_1)
```

This randomly partitions the data into 10 roughly equal-sized pieces, then fits the model 10 times, each time to 9/10 of the data, and returns:

```
           Estimate  SE
elpd_kfold   -155.4 4.9
```

This is the estimated log score of the data under cross validation, and it is on the same scale as LOO (although we cannot use the result of `loo` for this example given the instability in the approximation).

We next compare the fit to that obtained from an alternative model using a more informative prior, the "regularized horseshoe," which is weakly informative, stating that it is likely that only a small number of predictors are relevant, but we don't know which ones. This prior is often useful when the number of predictors in the model is relatively large compared to the number of observations. Here is the computation:

```
fit_2 <- update(fit_1, prior=hs())
kfold_2 <- kfold(fit_2, K=10)
loo_compare(kfold_1, kfold_2)
```

Here is the result of comparing the two cross validation estimates of the log score:

[10] Code for this example is in the folder `FakeKCV`.

```
Model comparison:
(negative 'elpd_diff' favors 1st model, positive favors 2nd)
elpd_diff        se
     13.3       5.5
```

Model 2 outperforms model 1 on this measure—an increase of 13 in the estimated log score—so for predictive purposes it seems better here to use the regularized horseshoe prior.

Concerns about model selection

Given several candidate models, we can use cross validation to compare their predictive performance. But it does not always make sense to choose the model that optimizes estimated predicted performance, for several reasons. First, if the only goal is prediction on data similar to what was observed, it can be better to average across models rather than to choose just one. Second, if the goal is prediction on a new set of data, we should reweight the predictive errors accordingly to account for differences between sample and population, as with the poststratification discussed in Section 17.1. Third, rather than choosing among or averaging over an existing set of models, a better choice can be to perform continuous model expansion, constructing a larger model that encompasses the earlier models as special cases. For example, instead of selecting a subset of variables in a regression, or averaging over subsets, one can include all the candidate predictors, regularizing the inference by partially pooling the coefficient estimates toward zero. Finally, in causal inference it can become difficult to interpret regression coefficients when intermediate outcomes are included in the model (see Section 19.6); hence typically we recommend not including post-treatment variables in such models even when they increase predictive power. If post-treatment predictors are included in a causal regression, then additional effort is required to extract estimated treatment effects; it is not enough to simply look at the coefficient on the treatment variable.

11.9 Bibliographic note

Berk (2004) considers some of the assumptions implicit in regression analysis and Harrell (2001) discusses regression diagnostics.

Simulation-based posterior predictive checking was introduced by Rubin (1984) and developed further by Gelman, Meng, and Stern (1996) and Gelman et al. (2013). Gabry et al. (2019) discuss visualizations and prior, posterior, and cross validation predictive checks as part of Bayesian workflow.

Influential ideas in predictive model comparison related to the Akaike (1973) information criterion (AIC) include C_p (Mallows, 1973), cross validation (Stone, 1974, 1977, Geisser and Eddy, 1979), and the deviance information criterion (DIC; Spiegelhalter et al., 2002). See also Fox (2002) for an applied overview, and Vehtari and Ojanen (2012), and Gelman, Hwang, and Vehtari (2014) for a Bayesian perspective. Vehtari, Gelman, and Gabry (2017) summarize some of our thinking on cross validation, information criteria, and predictive model evaluation and discuss the calculation of leave-one-out cross validation in R and Stan. Yao et al. (2018) discuss how to use LOO for averaging over a set of predictive models. Buerkner and Vehtari (2019, 2020) discuss how to use cross validation for time series and certain non-factorized models. Magnusson, Andersen, Jonasson and Vehtari (2020) discuss how leave-one-out cross validation can be further sped up using subsampling.

Piironen and Vehtari (2017a) discuss and demonstrate the overfitting if cross validation or information criteria are used to select among large number of models. Bayesian R^2 is discussed by Gelman, Goodrich, et al. (2019). The regularized horseshoe prior was introduced by Piironen and Vehtari (2017b).

(b) Put the above step in a loop and repeat 1000 times. Calculate the confidence coverage for the 68% intervals for each of the three coefficients in the model.

11.7 *Correlation and explained variance*: In a least squares regression with one predictor, show that R^2 equals the square of the correlation between x and y.

11.8 *Using simulation to check the fit of a time-series model*: Find time-series data and fit a first-order autoregression model to it. Then use predictive simulation to check the fit of this model as in Section 11.5.

11.9 *Leave-one-out cross validation*: Use LOO to compare different models fit to the beauty and teaching evaluations example from Exercise 10.6:

(a) Discuss the LOO results for the different models and what this implies, or should imply, for model choice in this example.

(b) Compare predictive errors pointwise. Are there some data points that have high predictive errors for all the fitted models?

11.10 *K-fold cross validation*: Repeat part (a) of the previous example, but using 5-fold cross validation:

(a) Randomly partition the data into five parts using the `sample` function in R.

(b) For each part, re-fitting the model excluding that part, then use each fitted model to predict the outcomes for the left-out part, and compute the sum of squared errors for the prediction.

(c) For each model, add up the sum of squared errors for the five steps in (b). Compare the different models based on this fit.

11.11 *Working through your own example*: Continuing the example from the final exercises of the earlier chapters, graph and check the fit of the model you fit in Exercise 10.11 and discuss the assumptions needed to use it to make real-world inferences.