CH 11: Assumptions - Part II

Posterior Predictive Checks

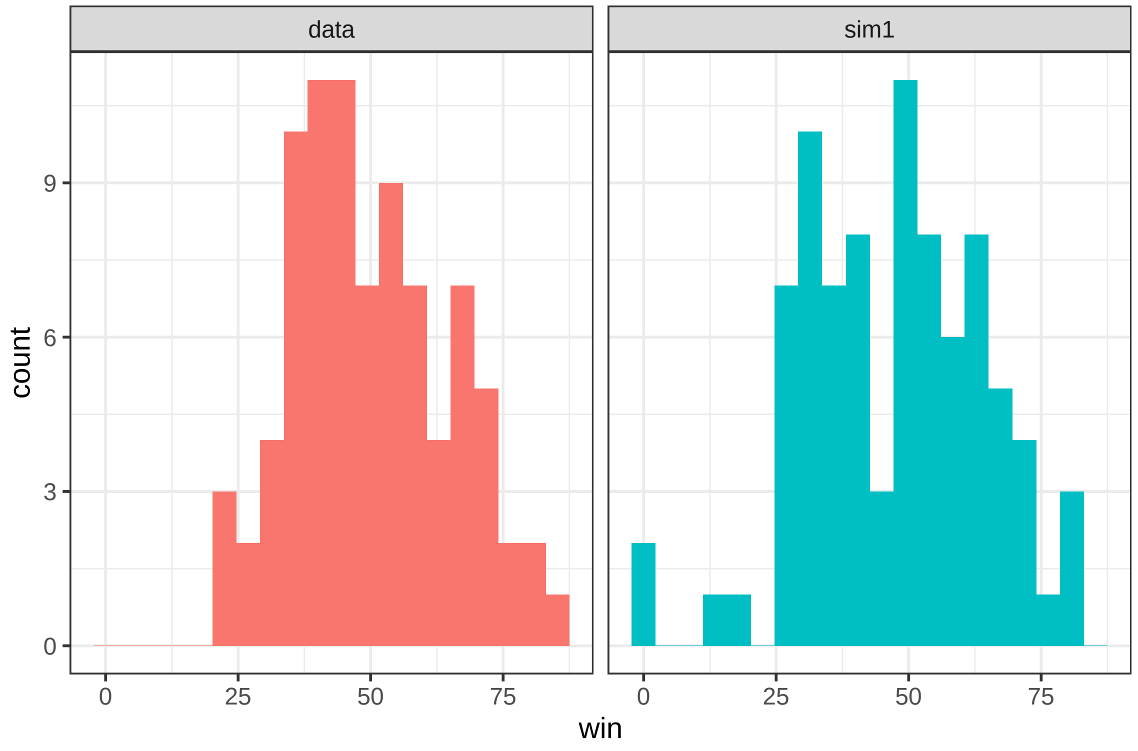
A another way of understanding the model fit is to use posterior predictive checks. *Posterior predictive checks create a predicted observation for each set of predictors, based on the assumed statistical model and prior distribution. Then the set of predictions can be combined and compared to the dataset. Generally a large set of predicted datasets are created.*

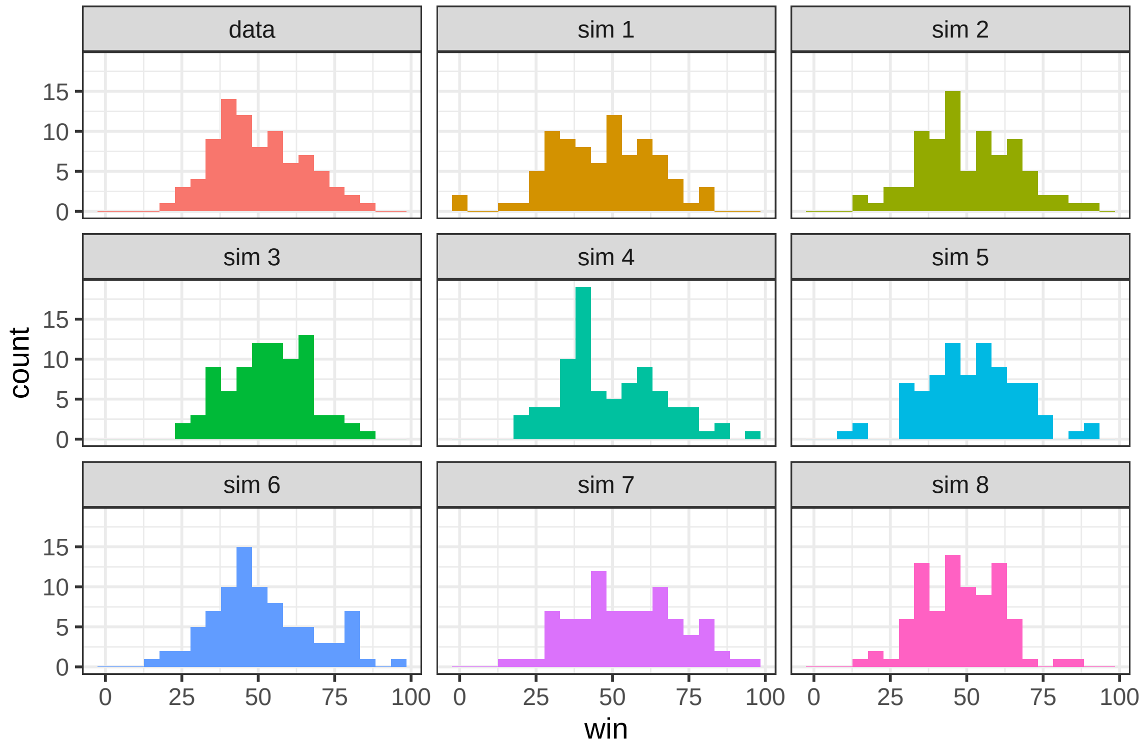
Consider the candy dataset.

```
candy <- read_csv("https://math.montana.edu/ahoegh/teaching/stat446/candy-data.csv") %>%
  mutate(pricepercent = pricepercent - mean(pricepercent),
         sugarpercent = sugarpercent - mean(sugarpercent))

candy_model <- stan_glm(winpercent ~ chocolate * caramel +
                        peanutyalmondy+ sugarpercent, data = candy, refresh = 0)

prediction_wins <- posterior_predict(candy_model, data = candy)
```

We can visually compare the simulated datasets with the true dataset.

```
tibble(win = c(candy$winpercent,prediction_wins[1,]), type = rep(c('data','sim1'), each = nrow(candy)))
  ggplot(aes(x =win, fill = type)) + geom_histogram(bins=20) +
  facet_wrap(.~type)  + theme_bw() +
  theme(legend.position = 'none')
```





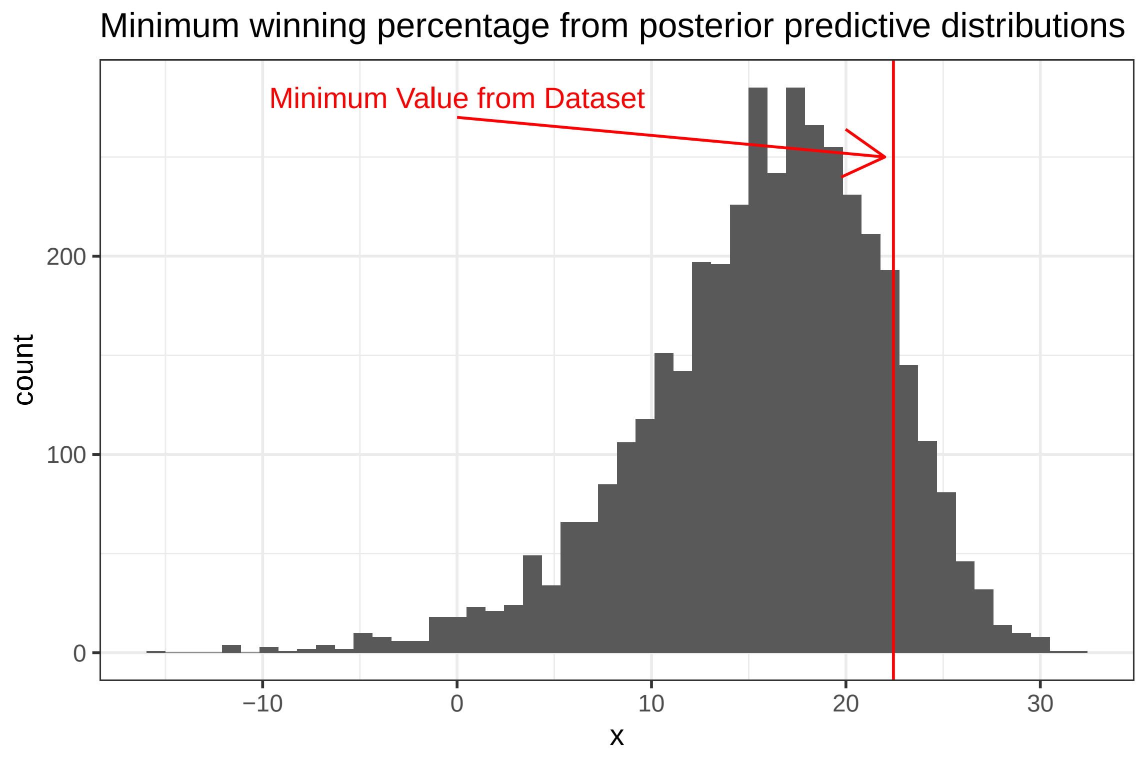
In addition to the visual inspection of the distributions of the data, we can also look at summary statistics from the simulations vs the observed data.

From the simulations, the minimum value of the simulation is less than the observed minimum value *87 percent of the time.*

Similarly, the maximum value of the simulation is greater than the observed maximum value *68 percent of the time.*

The goal here isn't necessary to accept or reject the model, but it can by useful in lacking for model defiencies.

```r
tibble(x = apply(prediction_wins,1, min)) %>%
  ggplot(aes(x = x)) + geom_histogram(bins = 50) +
  theme_bw() + ggtitle('Minimum winning percentage from posterior predictive distributions') +
  geom_vline(xintercept = min(candy$winpercent), col = 'red') +
  annotate('text', x = 0, y = 280, label ='Minimum Value from Dataset', color = 'red') +
  annotate('segment', x = 0, xend = 22, y =270, yend = 250, arrow = arrow(), color = 'red')
```



Residual Standard Deviation and explained variance (R^2)

The residual standard deviation in the model summarizes how accurately the model can predict the outcome. WHen comparing different models, smaller standard deviation would be better. While the magnitude of the standard deviation may be useful in some settings, it is more common to compare this to the total variation in the response.

The coefficient of determination,

$$R^2 = 1 - \frac{\hat{\sigma}^2}{s_y^2}$$

summarizes what proportion of the variation of the data is explained by the model, where $\hat{\sigma}^2$ is the estimated variance of the model and s_y^2 is the standard variance of the observations.

At the extreme values, $\hat{y} = X\hat{\beta} \approx \bar{y}$, *thus* $\hat{\sigma}^2 \approx s_y^2$ *and* $R^2 \approx 0$.

At the other extreme, $\hat{\sigma} \approx 0$ *and* $R^2 \approx 1$.

Note that the R^2 value does not account for the number of predictors in the model

Bayesian R^2

Conceptually, R^2 can be constructed as $\left(\frac{\text{Explained Variance}}{\text{Explained Variance + Residual Variance}}\right)$.

Using this framework, a Bayesian analog can be defined as

$$\text{Bayesian } R^2_s = \frac{V(\hat{y}_i^s)}{V(\hat{y}_i^s) + \sigma^2_s},$$

where $V(\hat{y}_i^s)$ is the variance of the predicted values for simulation s.

```
bayes_R2(candy_model) %>% head()
```

```
## [1] 0.4983184 0.3887992 0.3638838 0.6633818 0.4034933 0.5176570
```

```
bayes_R2(candy_model) %>% mean() %>% round(3)
```

```
## [1] 0.476
```

Cross-Validation

Another way to compare models is based on the predictive ability of the model. *It is important that the same observations are not used to fit the model and then be predicted.*

One way to do this uses cross-validation, where a chunk of the data is removed from the data for prediction.

Leave one out (LOO) cross-validation fits the model n different times, removing a single data point each time. In certain scenarios, it may be desirable to not run the model n times. Rather, the data can be broked into k folds. Then one fold is removed from the dataset at a time.

From a classical perspective, or in many machine learning scenarios, it may make sense to compare point predictions with something like *mean-squared error:* $\frac{1}{n}\sum(y_i - \hat{y})^2$ *or mean absolute deviation* $\frac{1}{n}\sum|y_i - \hat{y}|$.

AIC (Akaike information criteria) is a common method to compare models. This uses the likelihood function of the model fit but includes a penalty for additional parameters in the model.

Using a Bayesian framework, it can be useful to incorporate the uncertainty in the predictions. Formally this is done using the predicted distribution $p(y_i|\beta,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}\left(y_i - X_i\beta\right)^2\right)$. The results can be summarized with expected log predictive density (elpd) or simply the log score, which is the sum for all of the data points.

```r
sugar <- stan_glm(winpercent ~ sugarpercent, data = candy, refresh = 0)
loo1 <- loo(sugar)

chocolate <- stan_glm(winpercent ~ chocolate, data = candy, refresh = 0)
loo2 <- loo(chocolate)
print(loo_compare(loo2, loo1))
```

```
##           elpd_diff se_diff
## chocolate   0.0       0.0  
## sugar     -19.6       5.9  
```

```r
k1 <- kfold(sugar, K = 5)
```

```
## Fitting model 1 out of 5

## Fitting model 2 out of 5

## Fitting model 3 out of 5

## Fitting model 4 out of 5

## Fitting model 5 out of 5
```

```r
k2 <- kfold(chocolate, K = 5)
```

```
## Fitting model 1 out of 5

## Fitting model 2 out of 5

## Fitting model 3 out of 5

## Fitting model 4 out of 5

## Fitting model 5 out of 5
```

```r
loo_compare(k1, k2)
```

```
##           elpd_diff se_diff
## chocolate   0.0       0.0  
## sugar     -18.6       5.9  
```