CH 11: Assumptions - Part II

Posterior Predictive Checks

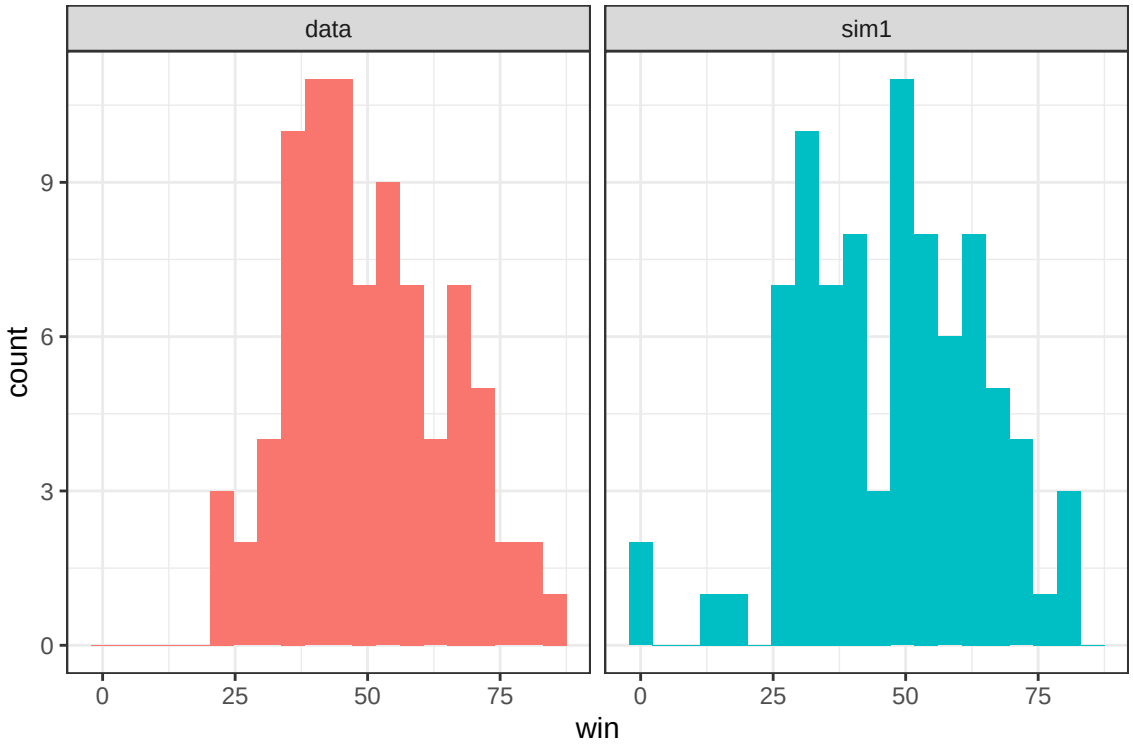
A another way of understanding the model fit is to use posterior predictive checks.

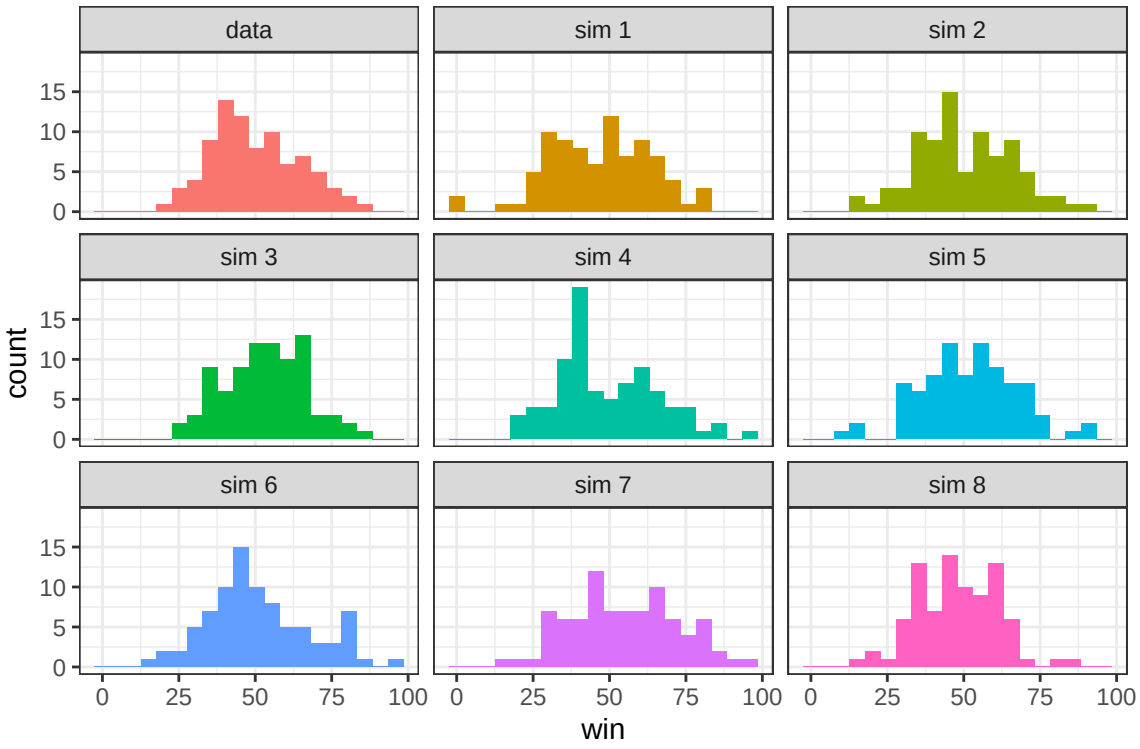
Consider the candy dataset.

```
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')
```



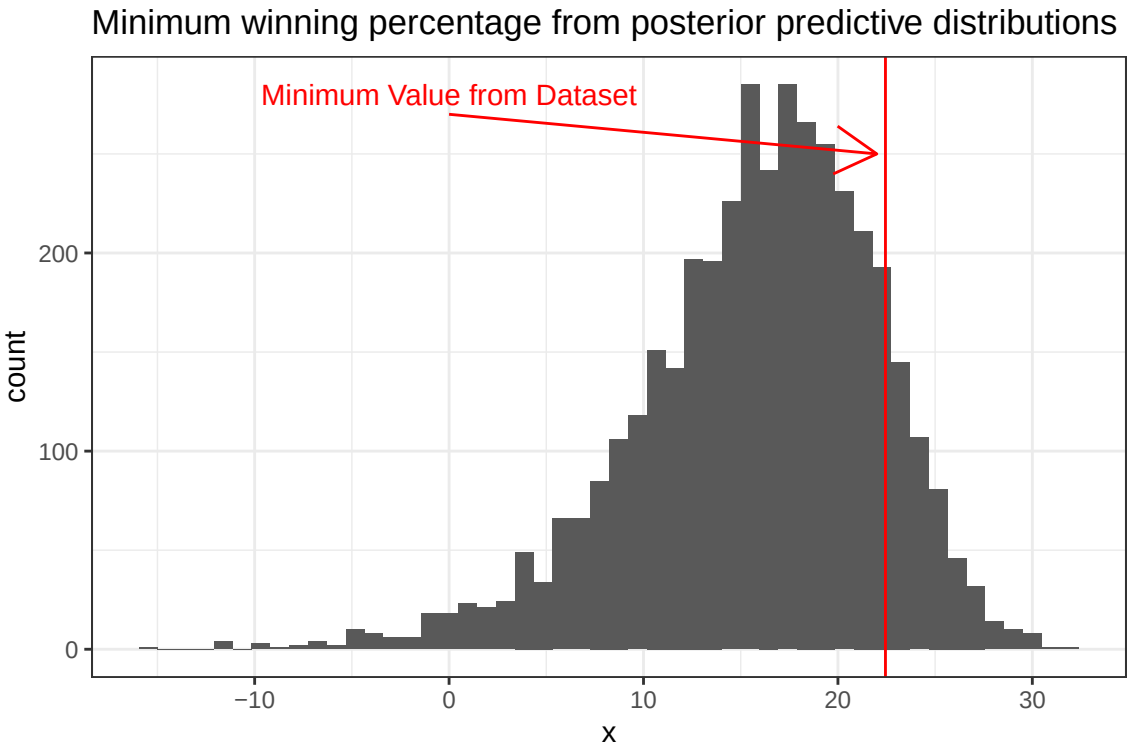


In addition to the visual inspection of the distributions of the data, we can also look at summary statistics from the simulations vs the observed data.

From the simulations, the minimum value of the simulation is less than the observed minimum value

Similarly, the maximum value of the simulation is greater than the observed maximum value

```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 coefficient of determination,

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

At the extreme values,

At the other extreme,

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

$$\text{Bayesian } R_s^2 = \frac{V(\hat{y}_i^s)}{V(\hat{y}_i^s) + \sigma_s^2},$$

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.

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

From a classical perspective, or in many machine learning scenarios, it may make sense to compare point predictions with something like

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 $p(y_i|\beta,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2\sigma^2}\left(y_i - X_i\beta\right)^2\right)$.

```
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
```