Prediction and Bayesian Inference

Bayesian Inference

Bayesian inference has three steps that are fundamentally different than classical estimation.

1. Additional information can be included using a prior distribution (for parameters)

2. The data, sampling model, and prior are combined to form a posterior distribution for model parameters, which are generally summarized with simulation.

3. Uncertainty in the posterior distribution can be propogated to get simulation-based predictions for unobserved or future outcomes.

Propogating Uncertainty

Recall the linear model fit for the beer data.

```
beer <- read_csv('http://math.montana.edu/ahoegh/Data/Brazil_cerveja.csv')
stan_fit <- stan_glm(consumed ~ max_tmp, data = beer, refresh = 0)
print(stan_fit)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ max_tmp
##  observations: 365
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 8.0    1.1
## max_tmp     0.7    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 3.4    0.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

While the parameters are summarized with a single point estimator (and a standard error), the model actually contains a collection of posterior simulations that capture the uncertainty in the model.

```
post_sims <- as.matrix(stan_fit)
head(post_sims)
```

```
##           parameters
## iterations (Intercept)   max_tmp    sigma
##       [1,]    8.977380 0.6141112 3.441056
##       [2,]    9.027112 0.6034721 3.625470
##       [3,]    8.834960 0.6345936 3.200719
##       [4,]    6.436217 0.7050201 3.221590
##       [5,]    5.559764 0.7402291 3.268298
##       [6,]    5.743474 0.7355585 3.347907
```

By default, the `stan_glm()` output calculates the scaled median absolute deviation to summarize uncertainty.

Formally the scaled median absolute deviation is

$$median_{i=1}^{N}|z_i - M| \times 1.483$$

where M is the median and z_i are the simulation values. *The 1.483 recovers the standard error with the normal distribution (and the nice properties associated with that). The author recommend the median absolute deviation due to computational stability.*

The point estimate and `standard errors` can be calculated directly

```
apply(post_sims, 2, median)
```

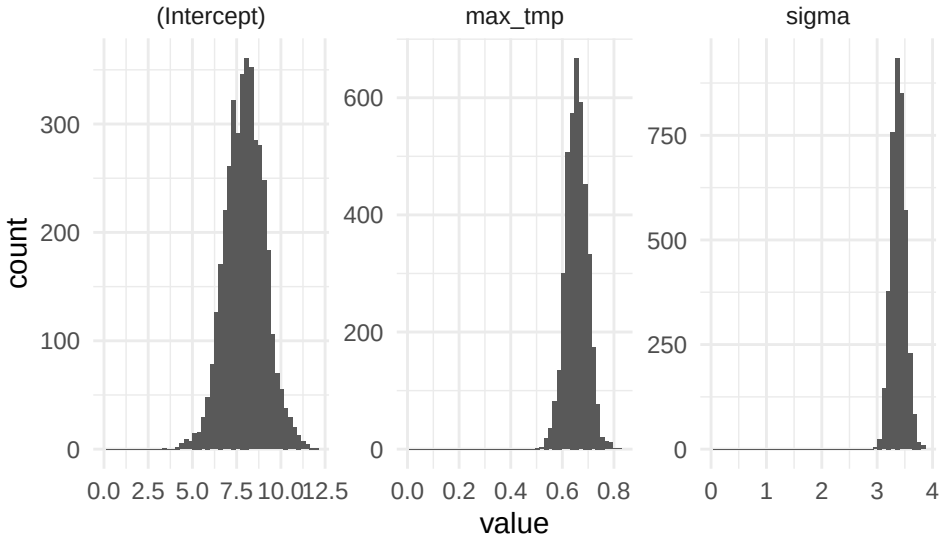
```
## (Intercept)     max_tmp       sigma
##   7.9913932   0.6541072   3.3816794
```

```
apply(post_sims, 2, mad)
```

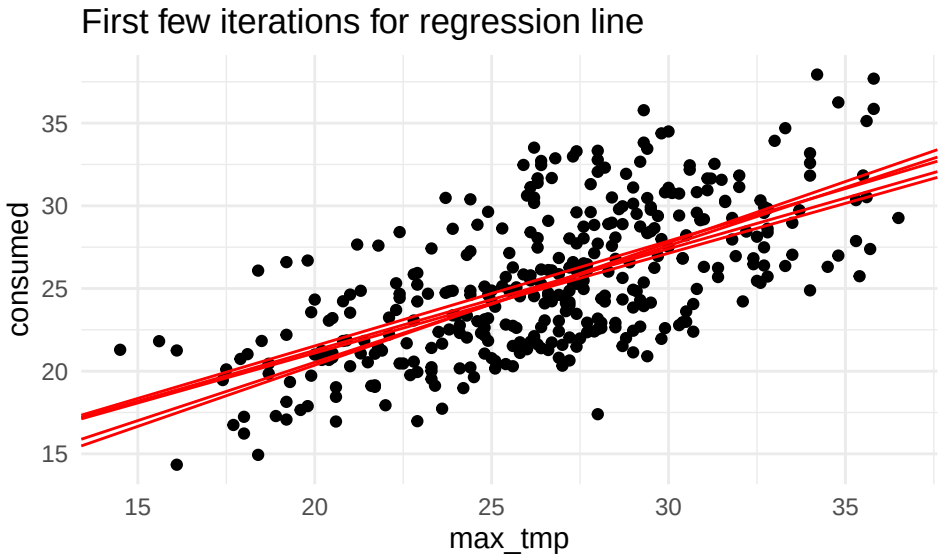
```
## (Intercept)     max_tmp       sigma
##  1.11615431  0.04105913  0.12786761
```

Visualizing Uncertainty We can visualize uncertainty in the parameter estimates using the simulation results directly.

```
post_sims %>% as.data.frame %>%
  pivot_longer(cols = c('(Intercept)', 'max_tmp', 'sigma')) %>%
  ggplot(aes( x= value)) + geom_histogram(bins = 50) +
  facet_wrap(.~ name, scales = 'free') + theme_minimal() + xlim(0, NA)
```



Furthermore, each iteration results in a joint set of parameters that could be used to create a regression line.



Contrasts and Functions of Parameters Bayesian inference also allows easy computation of contrasts and more generally functions of parameters.

Consider the contrast (predicted mean difference) between a *weekend day with* $max\_tmp = 20$ *and a weekday with* $max\_tmp = 30$*. This would be a difficult problem to solve analytically, but it is straightforward using simulation.*

Note that a contrast of this sort (or any time), could also be calculated from a classical perspective, but, strictly speaking, the simulation approach is not permitted. Rather an analytical calculation, likely with a normal approximation (delta method?) would be necessary.

1. Fit the model.

```
stan_fit2 <- stan_glm(consumed ~ weekend + max_tmp, data = beer, refresh = 0)
print(stan_fit2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ weekend + max_tmp
##  observations: 365
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) 5.9    0.8
## weekend     5.2    0.3
## max_tmp     0.7    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 2.4    0.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

2. Extract the simulations.

```
contrast_sims <- as.matrix(stan_fit2)
```

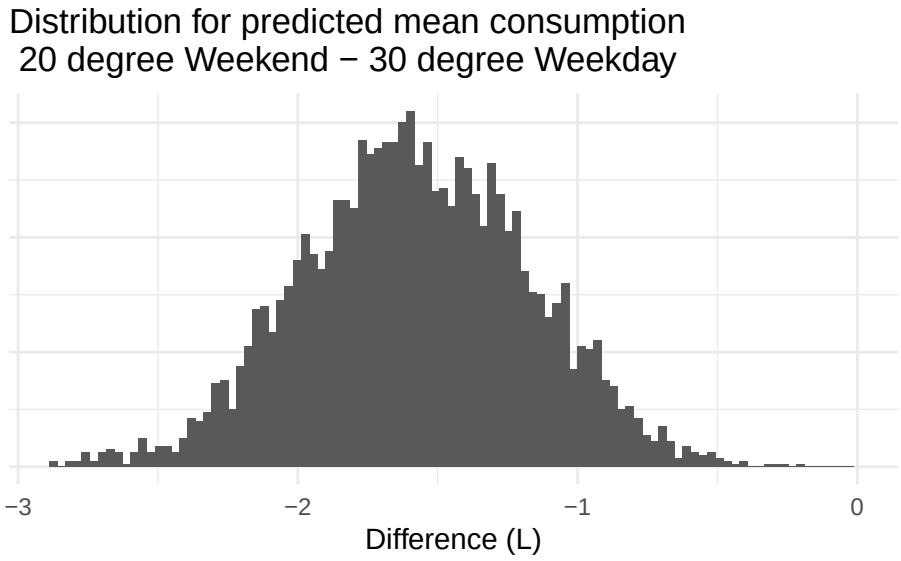
3. Compare differences

```
diff_consumed <- as.numeric(contrast_sims %*% matrix(c(1,1,20,0)) -
  contrast_sims %*% matrix(c(1,0,30,0)))
```

4. Calculate interval and plot difference

```
##       2.5%      97.5%
## -2.3537635 -0.8061307
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```



Prediction

When considering prediction, there are a few different values that can be predicted. For context, consider predicting consumption for max temperature of 25, from the first model that does not consider day of week.

1. point prediction: *at x' would be $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x'$. This is a point prediction, with no consideration of uncertainty - rarely used (imo).*

```
x_prime <- tibble(max_tmp = 25)
predict(stan_fit, newdata = x_prime)
```

```
##       1
## 24.3434
```

2. linear prediction with uncertainty: *at x' would be $\hat{y} = \beta_0 + \beta_1 x'$. This might be a bit of abuse of notation, but we can directly use simulated values of β_0 and β_1 and propogate this uncertainty. The interpretation of this is the distribution for the expected or average value of y at x', not an individual point.*

```
posterior_linpred(stan_fit, newdata = x_prime) %>% quantile( probs = c(.025, .975))
```

```
##     2.5%    97.5%
## 23.98307 24.71338
```

3. predictive distribution for a new observation: *at x' would be $\hat{y} = \beta_0 + \beta_1 x' + error$. This gives us the predictive distribution for a new observation, rather than the average response at a specific value.*

```
posterior_predict(stan_fit, newdata = x_prime) %>% quantile( probs = c(.025, .975))
```

```
##     2.5%    97.5%
## 17.76881 30.80955
```

Each of these values could easily be computed using the simulations with a little matrix algebra in R.

Priors

With Bayesian analysis, a major positive (and potentially a negative) is the ability to specify a prior distribution that prior belief about the parameters.

While this is not a formal Bayesian class, and we won't deviate much from the default priors, it is still important to understand how they impact our results.

With a normal prior distribution and a normal likelihood (sampling model), the Maximum A'Posteriori (MAP) estimator can be written as a weighted average of the prior mean and the sample mean. Formally, this is

$$\theta_{MAP} = \left(\frac{1}{\sigma^2_{prior}} \theta_{prior} + \frac{1}{\frac{\sigma^2_{data}}{n}} \bar{y} \right) / \left(\frac{1}{\sigma^2_{prior}} + \frac{1}{\frac{\sigma^2_{data}}{n}} \right)$$

Furthermore, the standard error of θ is

$$se_{bayes} = \frac{1}{\sqrt{\frac{1}{\sigma^2_{prior}} + \frac{1}{\frac{\sigma^2_{data}}{n}}}}$$

With a uninformative prior $\theta \sim N(\mu, \sigma^2 \to \infty)$, the point estimate and standard errors converge to those from the classical, least squares setting.

The `stan_glm()` function does not actually use uninformative priors, but rather uses **weakly informative** priors. This corresponds to a normal prior with scaled standard deviation of 2.5.

For more details on the `stan_glm()` prior structure, see http://mc-stan.org/rstanarm/articles/priors.html

In our work, if you do use `stan_glm()` make sure to report the your prior. (For project 2, this would technically be part of the model statement.)