CH 10.1 - 10.6: Multiple Predictors

Regression with Multiple Predictors

Binary Predictor With a single binary predictor, using the beer dataset, the model can be written as

$$y = \beta_0 + \beta_1 x_{weekend=1} + \epsilon,$$

where:

- y is the beer consumption,
- β_0 is the consumption on a weekday,
- β_1 is the difference in consumption between a weekend day and a weekend,
- $x_{weekend=1}$ is an indicator function for whether the observation is a weekend.

This is known as the reference case parameterization.

```
stan_glm(consumed ~ weekend, data = beer, refresh = 0)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ weekend
##  observations: 365
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 24.0    0.2
## weekend      4.9    0.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 3.8    0.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Alternatively, the cell means model can be constructed, where

$$y = \beta_1 x_{weekend=0} + \beta_2 x_{weekend=1} + \epsilon,$$

in this case

- β_1 is the mean consumption on a weekday
- β_2 is the mean consumption on a weekend.

```r
beer <- beer %>% mutate(weekend = factor(weekend))
stan_cm <- stan_glm(consumed ~ weekend - 1, data = beer, refresh = 0)
stan_cm
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ weekend - 1
##  observations: 365
##  predictors:   2
## ------
##          Median MAD_SD
## weekend0 24.0    0.2
## weekend1 28.9    0.4
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 3.8    0.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

There is not a direct contrast in the model specification, but that can easily be computed (especially using simulation).

```r
as.data.frame(stan_cm) %>% mutate(contrast = weekend1 - weekend0) %>%
  summarise(median_diff = median(contrast), lower_interval = quantile(contrast, probs = .025),
            upper_interval = quantile(contrast, probs = .975))
```

```
##   median_diff lower_interval upper_interval
## 1    4.924486       4.062651        5.82757
```

Binary Predictor + Continuous Predictor Now consider jointly considering both weekend/weekday and maximum temperature. The model can now be written as

$$y = \beta_0 + \beta_1 x_{weekend=1} + \beta_2 x_{tmp} + \epsilon,$$

where:

- y is the beer consumption,

- *β_0 is the consumption on a weekday with maximum temperature of 0*

- *β_1 is the expected difference in consumption between a weekend day and a weekend, holding maximum temperature constant*

- *β_2 is the expected difference in consumption for a 1 degree change in maximum temperature, holding the day of week constant.*

```
ml_regression <- beer %>% stan_glm(consumed ~ weekend + max_tmp, data = ., refresh = 0)
ml_regression
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ weekend + max_tmp
##  observations: 365
##  predictors:   3
## ------
##             Median MAD_SD
## (Intercept) 5.9    0.8
## weekend1    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
```

Coefficient Interpretation

- When interpreting coefficients in a multiple regression model, it is important to understand that these values control for other predictors in the model! *The values will change with the inclusion/exclusion of other predictors.*

- Sometimes we cannot necessarily hold all other predictors constant in a model. *A simple example would be* $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$

The textbook puts an emphasis on differentiating predictive and counterfactual interpretations:

- The predictive interpretation focuses on how the outcome differs, on average, when *comparing* two groups of items that differ by 1 unit (and all other predictors are the same). *The coefficient is the expected difference in y between these two items.*

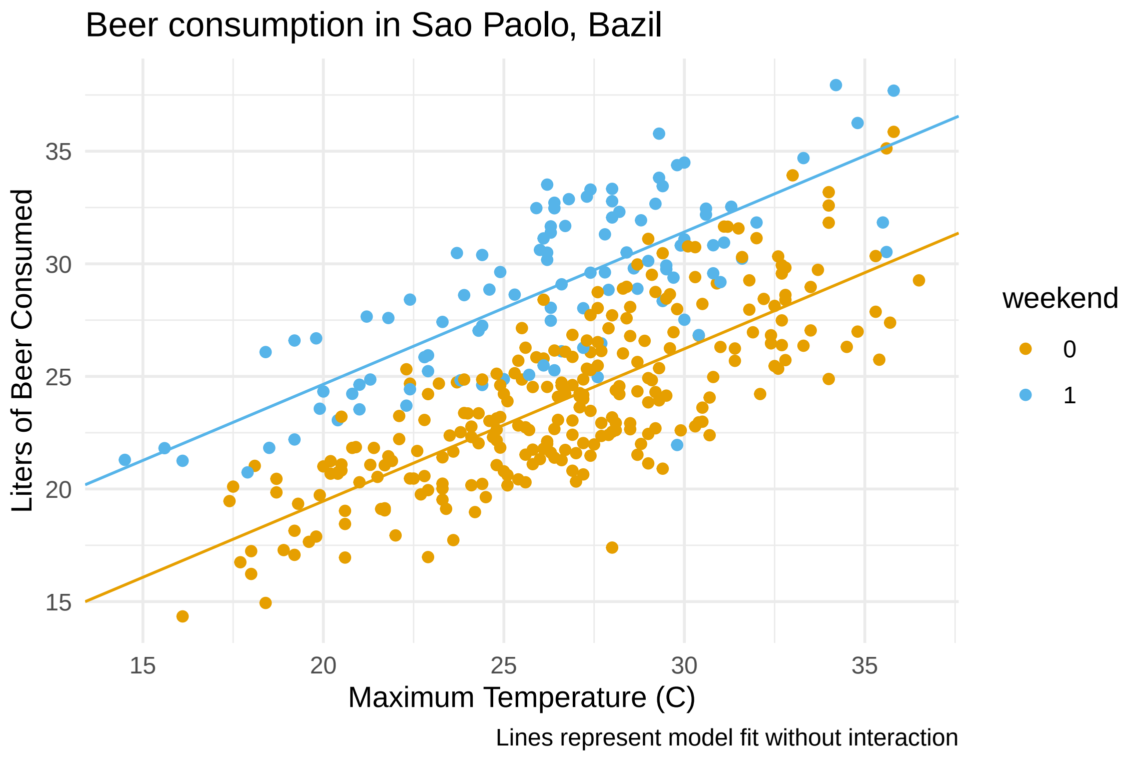
- The counterfactual interpretation focuses on how the outcome would differ with an individual, rather than between individuals. *The coefficient is the expected change in y caused by adding one to the predictor (holding all other predictors the same)*

The counterfactual interpretation should be reserved for a situation where causal inferences are reasonable, such as a completely randomized experimental design.

It is easy to get careless with wording and say things like "a change in temperature is associated with a change in consumption," but *the safest interpretation focuses on comparisons between units rather than changes within units.*

Interaction The model what we have fit, results in two parallel lines.

```r
beer %>% ggplot(aes(y = consumed, x = max_tmp, color = weekend)) +
  geom_point() +
  geom_abline(intercept = as.numeric(ml_regression$coefficients[1]),
              slope = as.numeric(ml_regression$coefficients[3]), color = "#E69F00") +
    geom_abline(intercept = as.numeric(ml_regression$coefficients[1] + ml_regression$coefficients[2]),
              slope = as.numeric(ml_regression$coefficients[3]), color = "#56B4E9") +
  scale_color_manual(values = c("#E69F00", "#56B4E9")) +
  theme_minimal() +
  xlab("Maximum Temperature (C)") +
  ylab("Liters of Beer Consumed") +
  ggtitle('Beer consumption in Sao Paolo, Bazil') +
  labs(caption = 'Lines represent model fit without interaction')
```

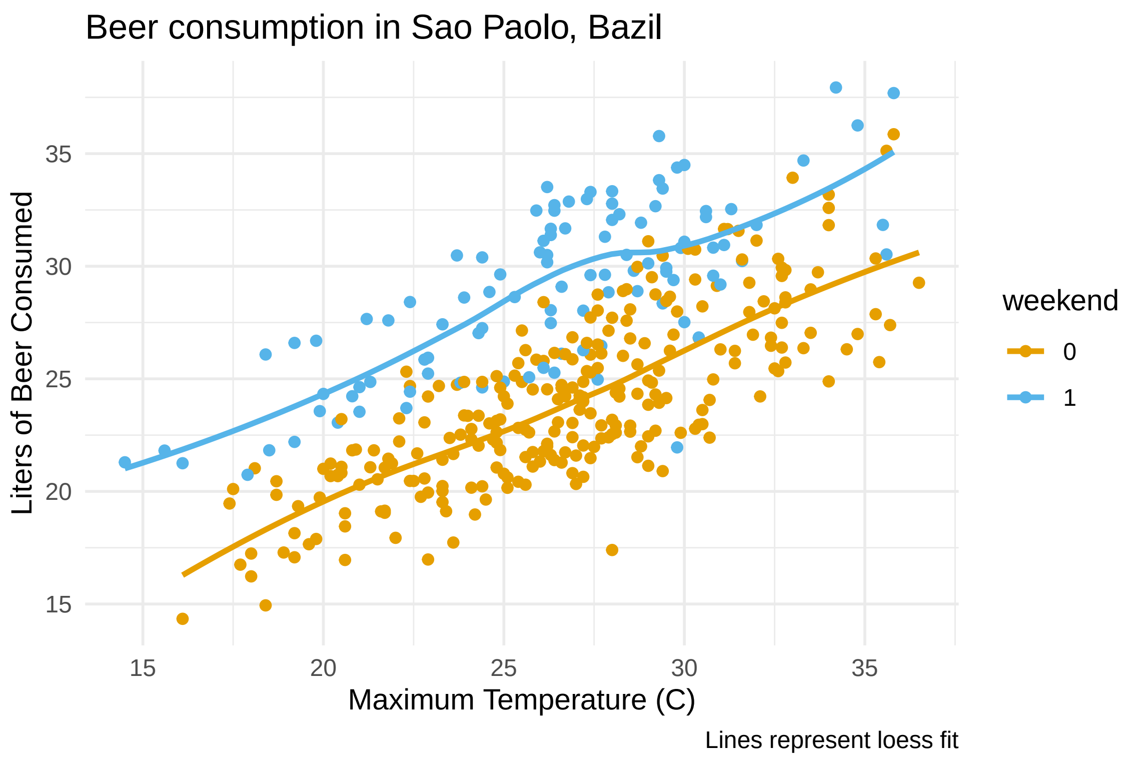


In this situation, the assumption of an additive model (parallel lines) seems reasonable. However, in many situations we'd expect that the relationship between a continuous covariate (e.g. an extra degree of maximum temperature) could depend upon another covariate (e.g. day of week).

With our data, an interaction would mean that the two lines *are not* parallel.

The next figure allows some flexibility to fit non-parallel (and non-linear) functional relationships.

```r
beer %>% ggplot(aes(y = consumed, x = max_tmp, color = weekend)) + geom_point() + geom_smooth(formula =
  scale_color_manual(values = c("#E69F00", "#56B4E9")) +
    theme_minimal() +
  xlab("Maximum Temperature (C)") +
  ylab("Liters of Beer Consumed") +
  ggtitle('Beer consumption in Sao Paolo, Bazil') +
  labs(caption = 'Lines represent loess fit')
```



This figure doesn't suggest a non-additive relationship, but nevertheless, let's explore the interaction model.

$$y = \beta_0 + \beta_1 x_{weekend=1} + \beta_2 x_{tmp} + \beta_3 x_{weekend=1} x_{tmp} + \epsilon,$$

- *β_0 is the consumption on a weekday with maximum temperature of 0*

- *β_1 is the expected difference in consumption between a weekend day and a weekend, with maximum temperature equal to zero. So intercept for weekend is $\beta_0 + \beta_1$*

- *β_2 is the expected difference in consumption for a 1 degree change in maximum temperature for weekdays*

- *β_3 is the difference between the slope of maximum temperature for weekdays and weekends. Thus the slope for weekends is $\beta_2 + \beta_3$*

```r
stan_glm(consumed ~ max_tmp * weekend, data = beer, refresh = 0)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      consumed ~ max_tmp * weekend
##  observations: 365
##  predictors:   4
## ------
##                  Median MAD_SD
## (Intercept)      5.7    0.9
## max_tmp          0.7    0.0
## weekend1         5.9    1.6
## max_tmp:weekend1 0.0    0.1
##
## 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
```

Interaction coefficients are more easily interpreted when the the continuous variables are centered or standardized.

R automatically creates indicator variables for categorical data (that are stored as factors).

```r
model.matrix(consumed ~ weekend, data = beer) %>% head(3)
```

```
##   (Intercept) weekend1
## 1           1        0
## 2           1        0
## 3           1        1
```

```r
model.matrix(consumed ~ weekend - 1, data = beer) %>% head(3)
```

```
##   weekend0 weekend1
## 1        1        0
## 2        1        0
## 3        0        1
```

To change the reference level, it is necessary to either directly specify the levels of the factor or reorder them (see `forcats`)

```r
beer %>% mutate(weekend_fact = factor(weekend, levels = c('1', '0'))) %>%
  lm(consumed ~ weekend_fact, data = .) %>% display()
```

```
## lm(formula = consumed ~ weekend_fact, data = .)
##               coef.est coef.se
## (Intercept)   28.92     0.37
## weekend_fact0 -4.92     0.44
## ---
## n = 365, k = 2
## residual sd = 3.80, R-Squared = 0.26
```

Computational Lab The purpose of this activity is to better understand an interaction model.

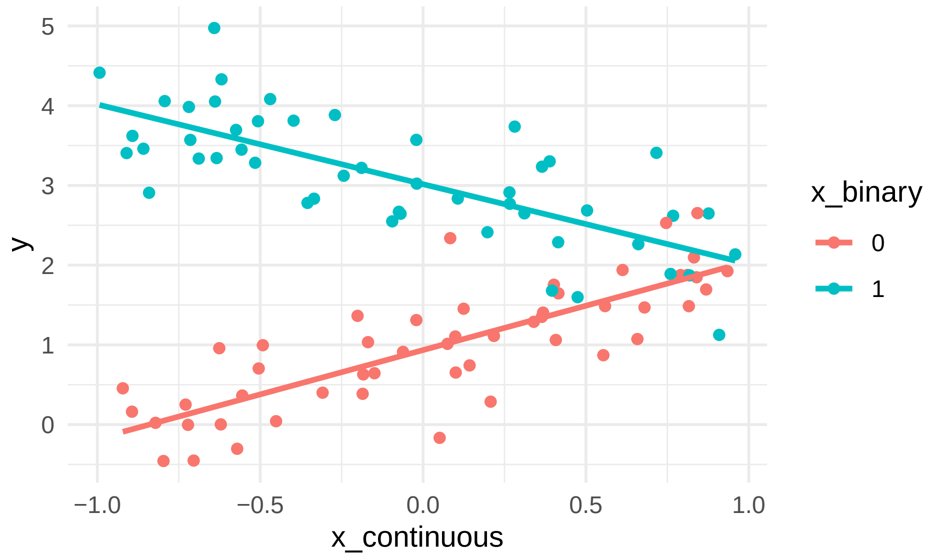
1. Simulate fake data that has an interaction.

```
n <- 100
fake_data <- tibble( x_binary = rbinom(n, 1, .5),
             x_continuous = runif(100, -1, 1))

beta <- c(1, 2, 1, -2)
sigma <- .5

fake_data <- fake_data %>%
  mutate(y =  rnorm(n, mean = beta[1] + x_binary * beta[2] + x_continuous * beta[3] + x_continuous * x_
        sd = sigma)) %>% mutate(x_binary =factor(x_binary))
```

2. Visualize the interaction.

```
fake_data %>% ggplot(aes(y=y, x=x_continuous, color = x_binary)) + geom_point() + geom_smooth(method =
  theme_minimal()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



3. Fit interaction model.

```
stan_glm(y ~ x_binary * x_continuous, data = fake_data, refresh = 0)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x_binary * x_continuous
##  observations: 100
##  predictors:   4
## ------
##                        Median MAD_SD
## (Intercept)             0.9    0.1
## x_binary1               2.1    0.1
## x_continuous            1.1    0.1
## x_binary1:x_continuous -2.1    0.2
##
## Auxiliary parameter(s):
```

```
##       Median MAD_SD
## sigma 0.5    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```