Multiple Linear Regression

We saw that we could fit a simple linear regression model when we have a numeric response and numeric explanatory variable. For instance,

```
bikeData <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
bikeData <- bikeData %>% mutate(log_selling_price = log(selling_price),
                                log_km_driven = log(km_driven))

slr_fit1 <- lm(log_selling_price ~ log_km_driven, data = bikeData)
summary(slr_fit1)
```

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven, data = bikeData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9271 -0.3822 -0.0337  0.3794  2.5656
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.63557    0.18455   79.31   <2e-16 ***
## log_km_driven -0.39109    0.01837  -21.29   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5953 on 1059 degrees of freedom
## Multiple R-squared:  0.2997, Adjusted R-squared:  0.299
## F-statistic: 453.2 on 1 and 1059 DF,  p-value: < 2.2e-16
```

What if we had another explanatory variable of interest (say `year`). We could fit another SLR model.

```
slr_fit2 <- lm(log_selling_price ~ year, data = bikeData)
summary(slr_fit2)
```

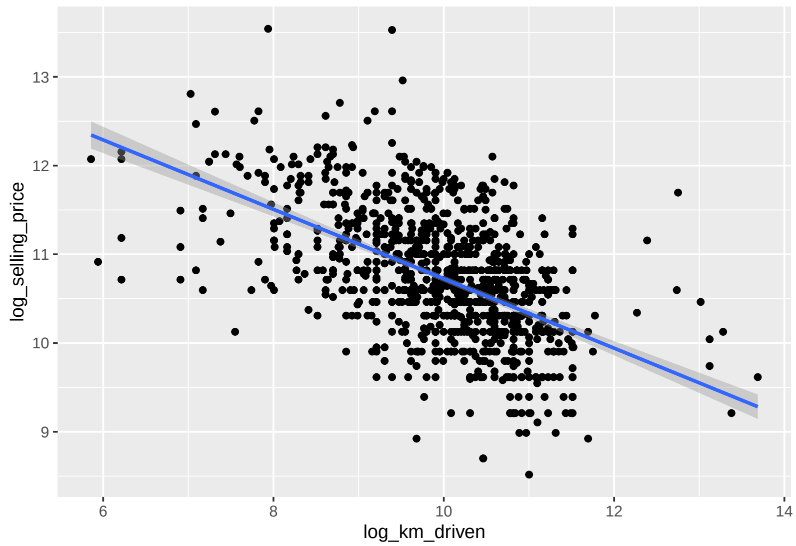
```
##
## Call:
## lm(formula = log_selling_price ~ year, data = bikeData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2917 -0.3814 -0.0948  0.2368  3.2436
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.011e+02  7.892e+00  -25.48   <2e-16 ***
## year         1.052e-01  3.919e-03   26.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5488 on 1059 degrees of freedom
```

```
## Multiple R-squared:  0.4048, Adjusted R-squared:  0.4042
## F-statistic: 720.1 on 1 and 1059 DF,  p-value: < 2.2e-16
```

Two x variables each used to predict our response y:

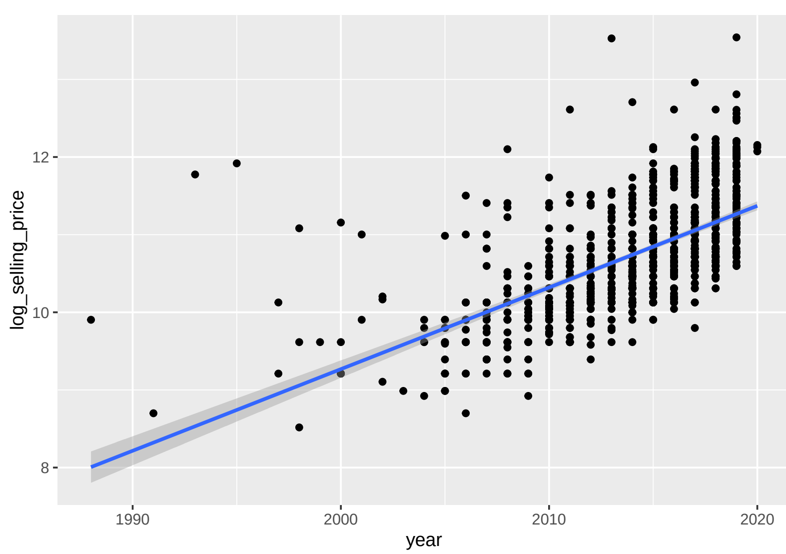
```r
ggplot(bikeData, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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




```r
ggplot(bikeData, aes(x = year, y = log_selling_price)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



How to include both in our model? Use a multiple linear regression model (MLR)!

$$\underbrace{Y_i}_{\text{response}} = \beta_0 + \beta_1 \underbrace{x_{1i}}_{\text{log km driven}} + \beta_2 \underbrace{x_{2i}}_{\text{year}} + \epsilon_i$$

Fitting the model in R

Just add to the right-hand side of our equation!

```r
mlr_fit <- lm(log_selling_price ~ log_km_driven + year, data = bikeData)
summary(mlr_fit)
```

RHS

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven + year, data = bikeData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48418 -0.34707 -0.06875  0.26960  2.73438
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.488e+02  8.438e+00  -17.63   <2e-16 ***
## log_km_driven -2.269e-01  1.792e-02  -12.66   <2e-16 ***
## year           8.034e-02  4.147e-03   19.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5117 on 1058 degrees of freedom
## Multiple R-squared:  0.483,  Adjusted R-squared:  0.4821
## F-statistic: 494.3 on 2 and 1058 DF,  p-value: < 2.2e-16
```

$\hat{\beta}_0$ $\hat{\beta}_1$ $\hat{\beta}_2$

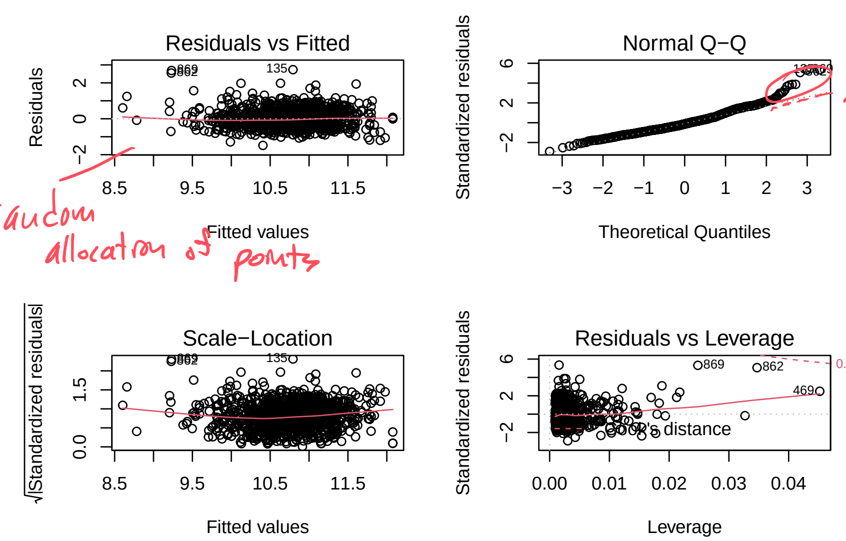
$H_0: \beta_1 = 0$ vs $H_A: \beta_1 \neq 0$

$H_0: \beta_2 = 0$ vs $H_A: \beta_2 \neq 0$

quality of model fit
close to 0, no predictive power, close to 1 lots of predictive power

Check assumptions as before:

```r
par(mfrow = c(2,2))
plot(mlr_fit)
```

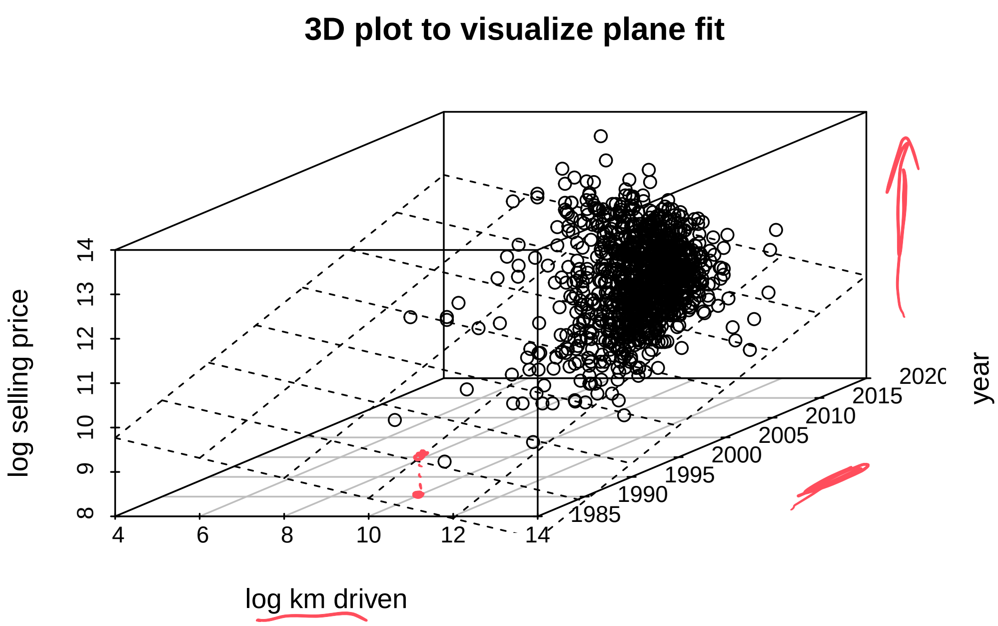


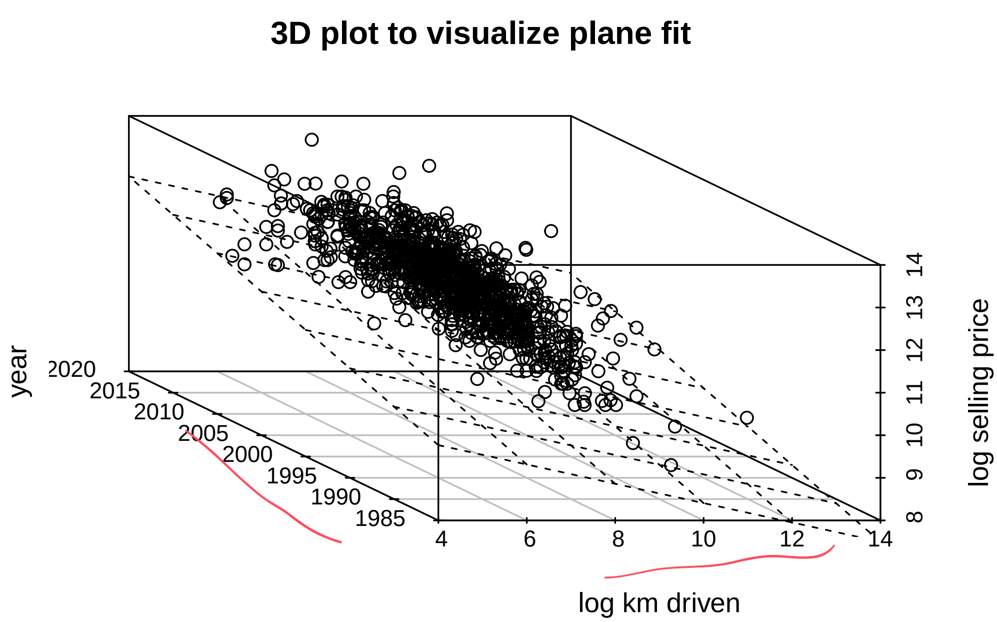
random allocation of points

Straight line

What is the model doing (visually)?

```
## Warning: package 'scatterplot3d' was built under R version 4.1.3
```





Including a Categorical Explanatory Variable

Consider adding a variable corresponding to 1st owner or multiple owners:

```
bikeData <- bikeData %>%
  mutate(owner_indicator = as.factor(ifelse(owner == "1st owner", "One", "Multiple")))
table(bikeData$owner_indicator)
```

```
##
## Multiple      One
##      137      924
```

Add this to one of the SLR models:

```
mlr_with_cat <- lm(log_selling_price ~ log_km_driven + owner_indicator, data = bikeData)
summary(mlr_with_cat)
```

change RHS

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven + owner_indicator,
##     data = bikeData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88281 -0.38518 -0.03601  0.37502  2.61047
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        14.57054    0.19790   73.63   <2e-16 ***
## log_km_driven      -0.38894    0.01852  -21.00   <2e-16 ***
## owner_indicatorOne  0.05003    0.05495    0.91    0.363
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5953 on 1058 degrees of freedom
## Multiple R-squared:  0.3002, Adjusted R-squared:  0.2989
## F-statistic:   227 on 2 and 1058 DF,  p-value: < 2.2e-16
```

test for whether or not different intercepts is important (i.e. owner-indicator is important)

What does `owner_indicatorOne` mean?

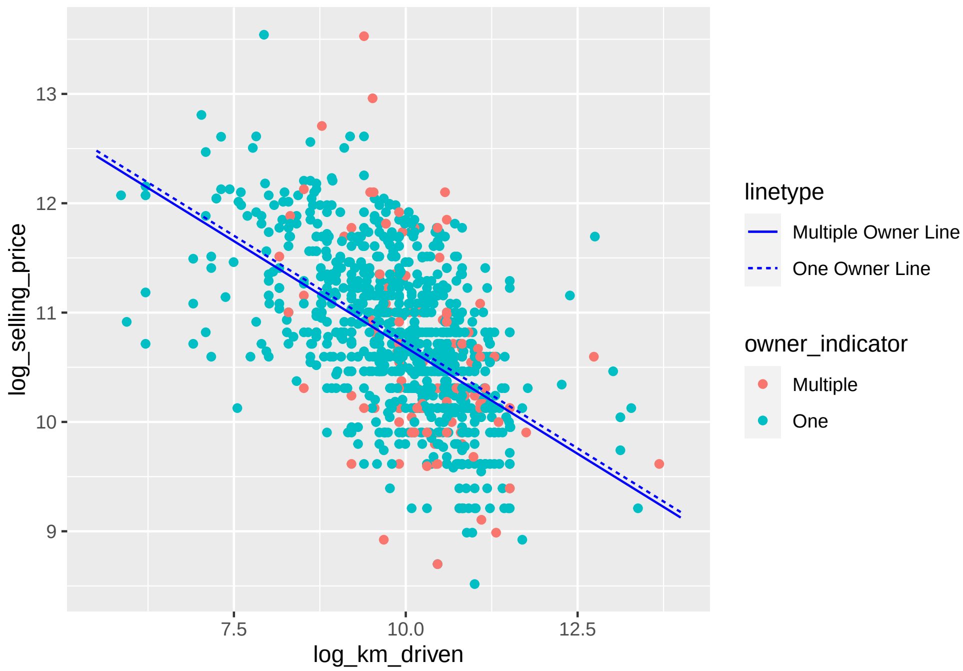
$$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + E_i$$

indicator variable for taking on 'one' for owner-indicator

$$\left(\begin{array}{ll} x_{2i} = 1 & \text{if one owner} \\ \phantom{x_{2i}} = 0 & \text{if multiple} \end{array} \right)$$

$x_{2i} = 1$ $\quad \beta_2 x_{2i} = \beta_2$

What does this do to our model?



If we add an **interaction term** we get completely different lines:

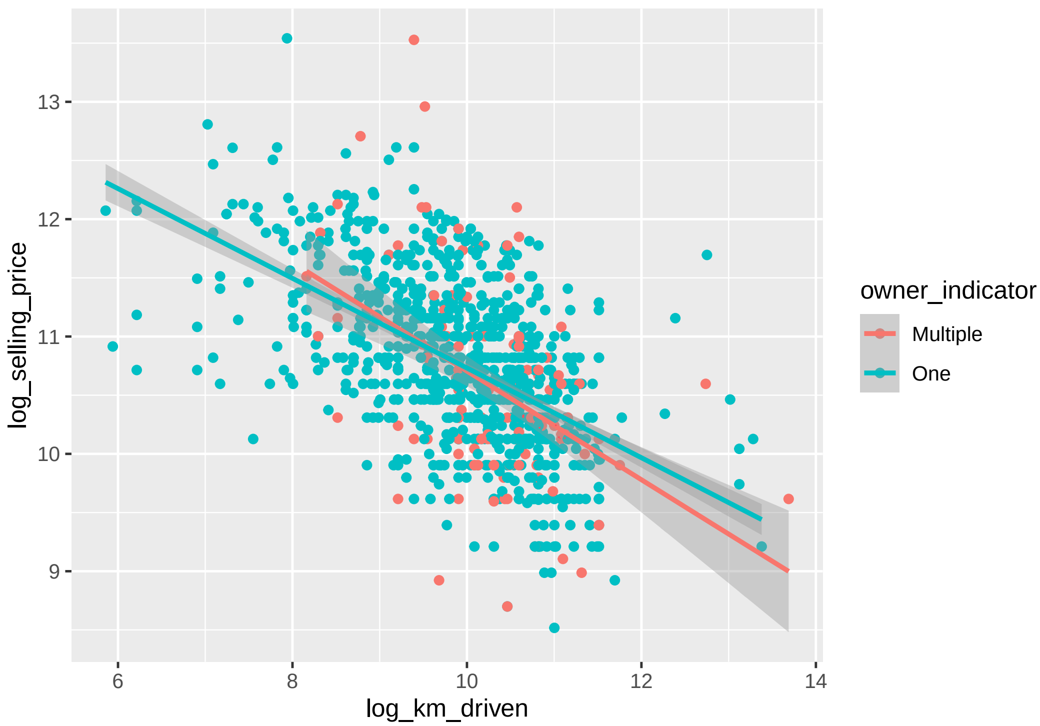
```
mlr_with_cat_interaction <- lm(log_selling_price ~ log_km_driven + owner_indicator +
                                log_km_driven:owner_indicator, data = bikeData)
summary(mlr_with_cat_interaction)
```

$Y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 (x_{1i} x_{2i}) + \epsilon_i$

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven + owner_indicator +
##     log_km_driven:owner_indicator, data = bikeData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.93041 -0.38473 -0.02977  0.37570  2.54163
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    15.33290    0.66286  23.131  < 2e-16 ***
## log_km_driven                  -0.46278    0.06401  -7.230 9.27e-13 ***
## owner_indicatorOne             -0.77943    0.69051  -1.129    0.259
## log_km_driven:owner_indicatorOne  0.08058    0.06687   1.205    0.228
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5952 on 1057 degrees of freedom
## Multiple R-squared:  0.3012, Adjusted R-squared:  0.2992
## F-statistic: 151.9 on 3 and 1057 DF,  p-value: < 2.2e-16
```

```
ggplot(bikeData, aes(x = log_km_driven, y = log_selling_price, color = owner_indicator)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



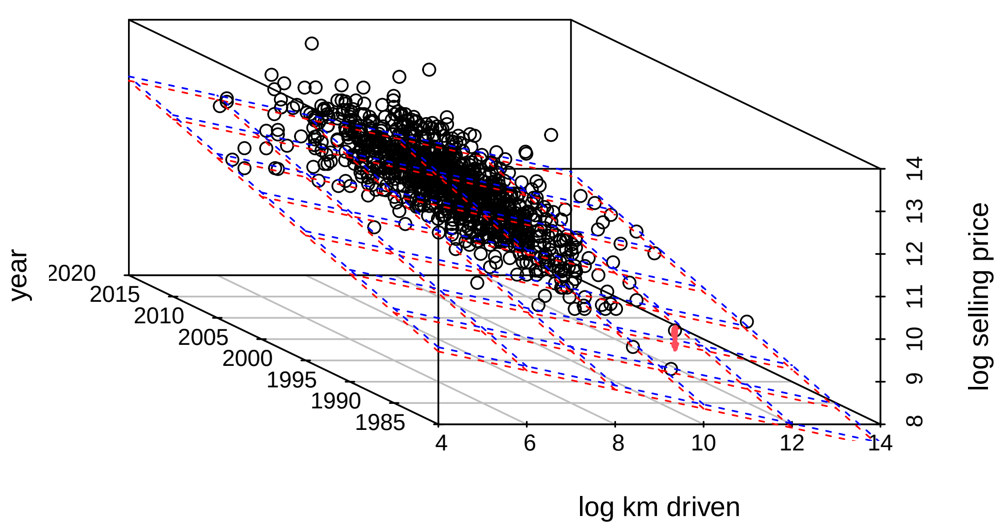
Note: The same idea works for the earlier MLR model! log-km-driven : year

```r
mlr_fit2 <- lm(log_selling_price ~ log_km_driven + year + owner_indicator, data = bikeData)
summary(mlr_fit2)
```

num num binary

```
##
## Call:
## lm(formula = log_selling_price ~ log_km_driven + year + owner_indicator,
##     data = bikeData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.56499 -0.35115 -0.06186  0.27405  2.64749
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.516e+02  8.526e+00 -17.774   <2e-16 ***
## log_km_driven       -2.283e-01  1.791e-02 -12.747   <2e-16 ***
## year                 8.176e-02  4.195e-03  19.487   <2e-16 ***
## owner_indicatorOne -1.002e-01  4.778e-02  -2.097   0.0362 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5109 on 1057 degrees of freedom
## Multiple R-squared:  0.4852, Adjusted R-squared:  0.4837
## F-statistic: 332.1 on 3 and 1057 DF,  p-value: < 2.2e-16
```

3D plot to visualize plane fit



Logistic Regression

Can include more explanatory variables in these models too. Same ideas apply (but the differences in fit are slightly more complicated).

```
water
```

```
## # A tibble: 3,276 x 10
##       ph Hardness Solids Chloramines Sulfate Conductivity Organic_carbon
##    <dbl>    <dbl>  <dbl>       <dbl>   <dbl>        <dbl>          <dbl>
##  1 NA        205. 20791.        7.30    369.         564.           10.4
##  2  3.72     129. 18630.        6.64     NA          593.           15.2
##  3  8.10     224. 19910.        9.28     NA          419.           16.9
##  4  8.32     214. 22018.        8.06    357.         363.           18.4
##  5  9.09     181. 17979.        6.55    310.         398.           11.6
##  6  5.58     188. 28749.        7.54    327.         280.            8.40
##  7 10.2      248. 28750.        7.51    394.         284.           13.8
##  8  8.64     203. 13672.        4.56    303.         475.           12.4
##  9 NA        119. 14286.        7.80    269.         389.           12.7
## 10 11.2      227. 25485.        9.08    404.         564.           17.9
## # ... with 3,266 more rows, and 3 more variables: Trihalomethanes <dbl>,
## #   Turbidity <dbl>, Potability <dbl>
```

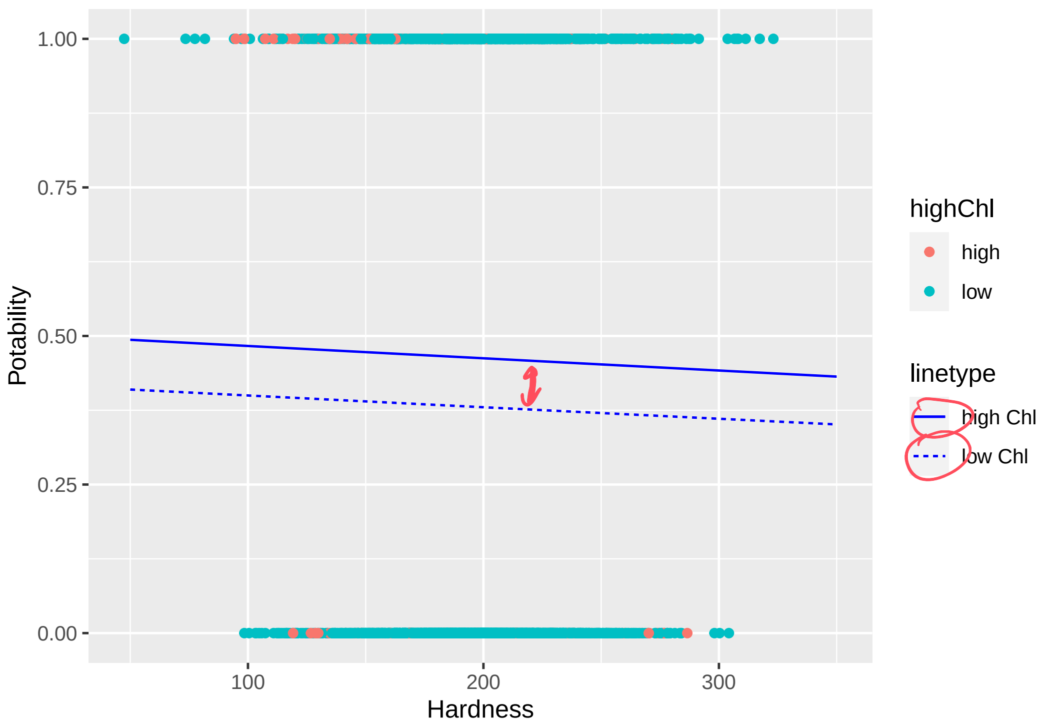
```
water <- water %>%
  mutate(highChl = ifelse(Chloramines > 9, "high", "low"))

log_reg_fit <- glm(Potability ~ Hardness + highChl, data = water, family = "binomial")
summary(log_reg_fit)
```

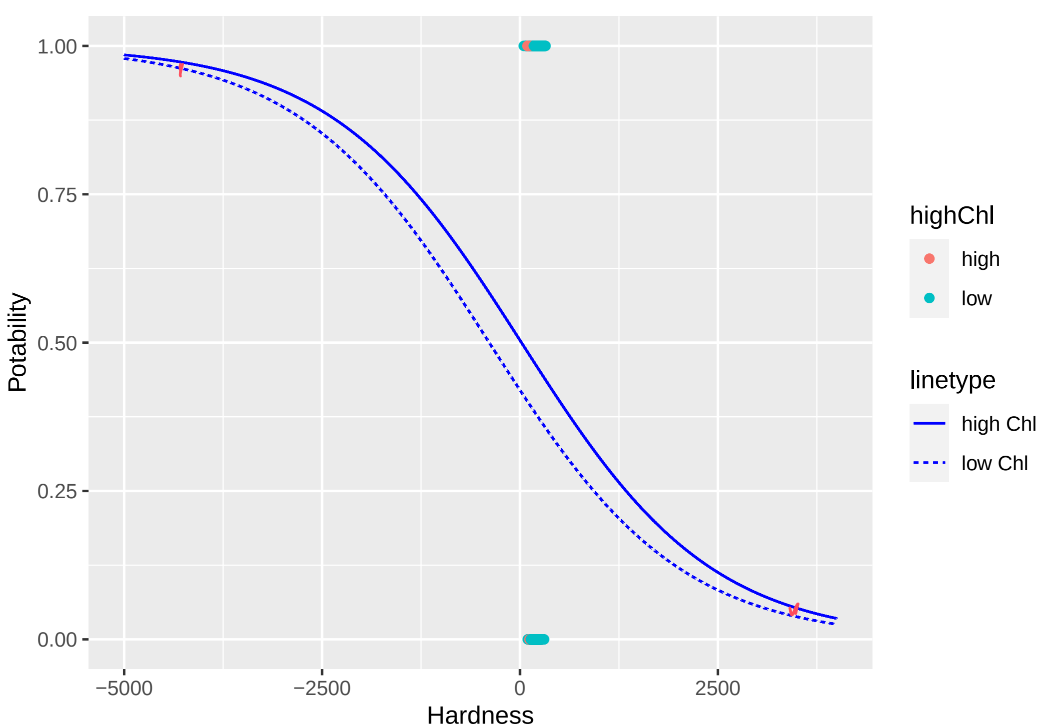
generalized

binary indicator variable

```
##
## Call:
## glm(formula = Potability ~ Hardness + highChl, family = "binomial",
##     data = water)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1423  -0.9825  -0.9713   1.3823   1.4367
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.0158503  0.2372393   0.067  0.94673
## Hardness    -0.0008313  0.0010903  -0.762  0.44581
## highChllow  -0.3387384  0.1111813  -3.047  0.00231 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4382.0  on 3275  degrees of freedom
## Residual deviance: 4372.1  on 3273  degrees of freedom
## AIC: 4378.1
##
## Number of Fisher Scoring iterations: 4
```

`highChl` just changes the 'intercept'. Mostly just shifts the logistic curve over in the part we care about...

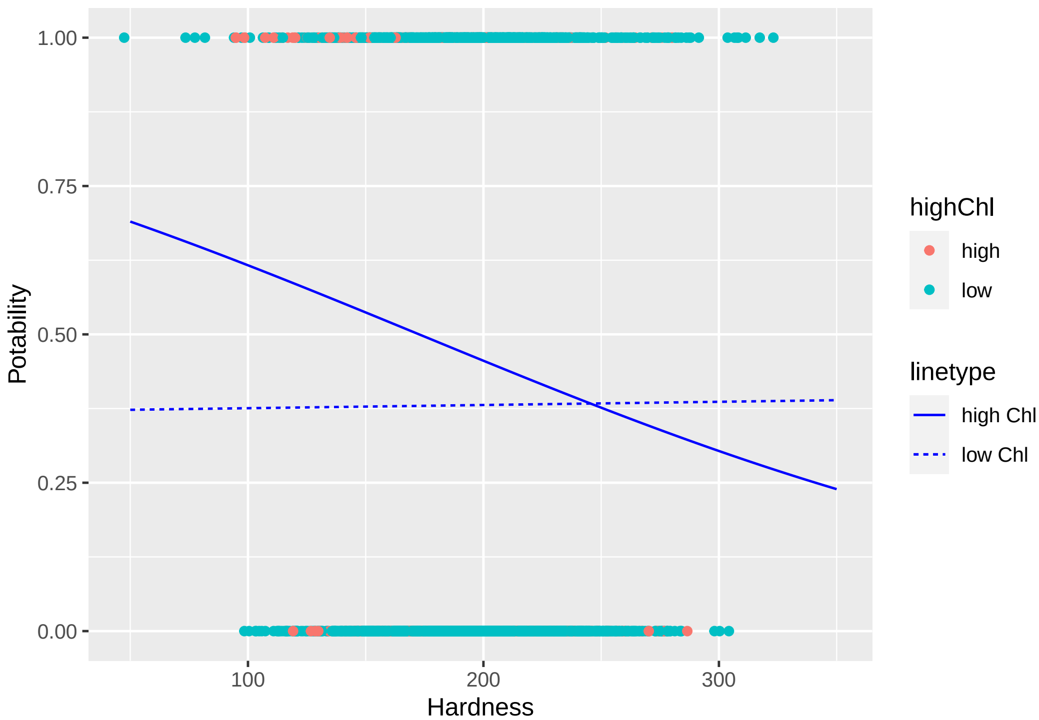




If we include an interaction between **Hardness** and **highChl** we get two separate logistic curves fit (one for the high group and one for the low group).

```
log_reg_fit2 <- glm(Potability ~ Hardness + highChl + Hardness:highChl,
                    data = water, family = "binomial")
summary(log_reg_fit2)
```

```
##
## Call:
## glm(formula = Potability ~ Hardness + highChl + Hardness:highChl,
##     family = "binomial", data = water)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3289  -0.9800  -0.9769   1.3876   1.4857
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)            1.126908   0.553358   2.036  0.04170 *
## Hardness              -0.006525   0.002788  -2.341  0.01924 *
## highChllow            -1.657998   0.601850  -2.755  0.00587 **
## Hardness:highChllow    0.006754   0.003030   2.229  0.02582 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4382.0  on 3275  degrees of freedom
## Residual deviance: 4367.1  on 3272  degrees of freedom
## AIC: 4375.1
##
## Number of Fisher Scoring iterations: 4
```



Just to see the curvature for the 'high Chl' group: