Data and Modeling

What makes something a statistical model?

What is the difference between prediction and inference?

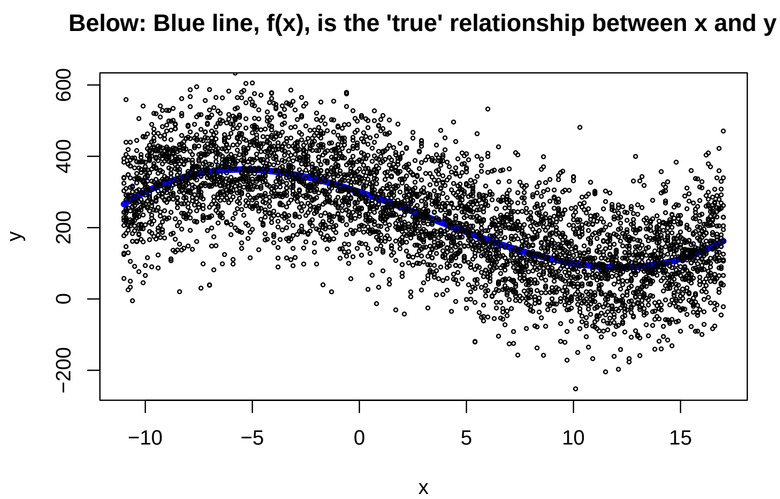
Data

- When modeling, what should our data look like?

Relating Explanatory Variables to a Response Variable

Consider the response Y as a random variable. We'll consider the x values fixed (for any explanatory variable). Our interest is in learning about the relationship between Y and x.

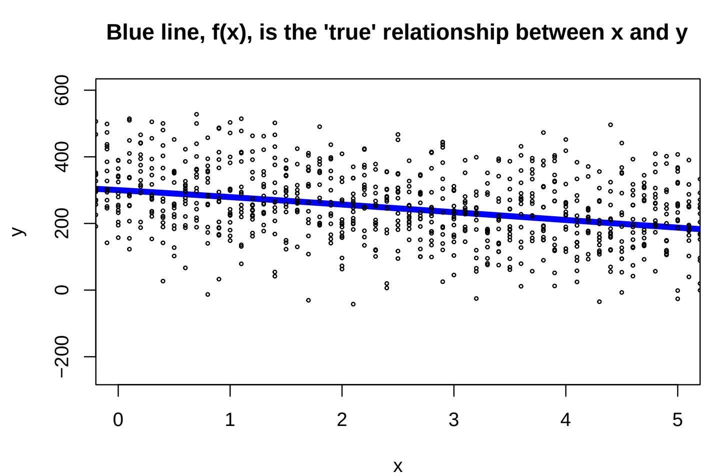
Y is random, so we don't have a **deterministic** relationship...



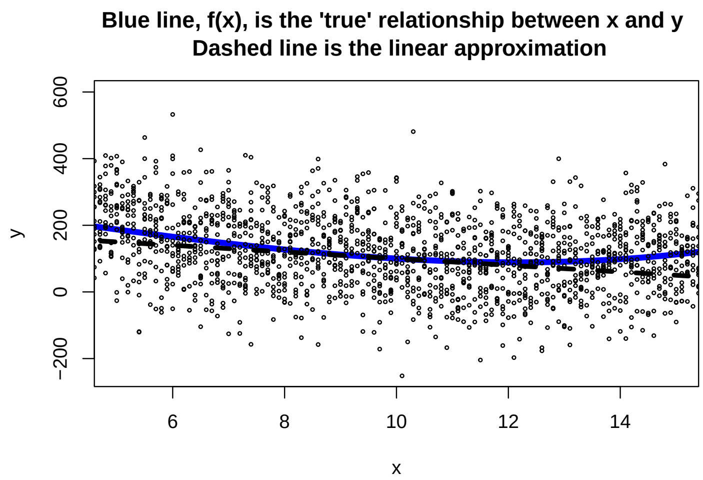
What should we try to relate/model?

Approximating $f(x)$

Although the true relationship is most certainly nonlinear, we may be ok approximating the relationship linearly. For example, consider the same plot as above but between 0 and 5 only:



That's pretty linear. Consider plot between 5 and 15:



Line still does a reasonable job and is often used as a basic approximation.

Exploratory Data Analysis (EDA)

What are our first steps with data?

Common steps to EDA

1.
2.
3.
4.
5.
6.

Data Intro

This dataset contains information about used motorcycles and their cost.

From the information page: This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning. The columns in the given dataset are as follows:

- name
- selling price
- year
- seller type
- owner
- km driven
- ex showroom price

The data are available to download from this URL:
https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv

Read in Data and Explore!

```
library(tidyverse)
bikeData <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
select(bikeData, selling_price, year, km_driven, ex_showroom_price, name, everything())
```

```
## # A tibble: 1,061 x 7
##    selling_price  year km_driven ex_showroom_price name        seller_type owner
##            <dbl> <dbl>     <dbl>             <dbl> <chr>       <chr>       <chr>
##  1        175000  2019       350                NA Royal Enfi~ Individual  1st ~
##  2         45000  2017      5650                NA Honda Dio   Individual  1st ~
##  3        150000  2018     12000            148114 Royal Enfi~ Individual  1st ~
##  4         65000  2015     23000             89643 Yamaha Faz~ Individual  1st ~
##  5         20000  2011     21000                NA Yamaha SZ ~ Individual  2nd ~
##  6         18000  2010     60000             53857 Honda CB T~ Individual  1st ~
##  7         78500  2018     17000             87719 Honda CB H~ Individual  1st ~
##  8        180000  2008     39000                NA Royal Enfi~ Individual  2nd ~
##  9         30000  2010     32000                NA Hero Honda~ Individual  1st ~
## 10         50000  2016     42000             60122 Bajaj Disc~ Individual  1st ~
## # ... with 1,051 more rows
```

Our 'response' variable here is the `selling_price` and we could use the variable `year`, `km_driven`, or `ex_showroom_price` as the explanatory variable. Let's make some plots and summaries to explore.

Linear Regression

Recap: Our goal is to predict a value of Y while including an explanatory variable x. We are assuming we have a sample of (x_i, y_i) pairs, $i = 1, ..., n$.

The Simple Linear Regression (SLR) model can be used:

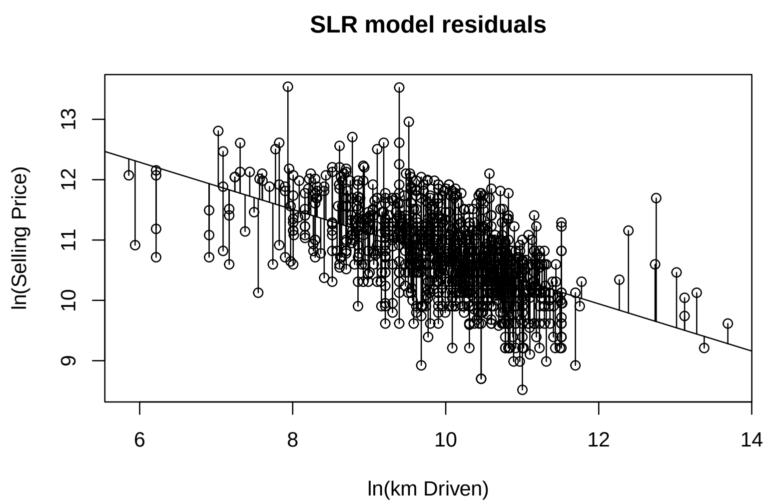
$$Y_i = \beta_0 + \beta_1 x_i + E_i$$

where

- y_i is our response for the i^{th} observation
- x_i is the value of our explanatory variable for the i^{th} observation
- β_0 is the y intercept
- β_1 is the slope
- $E_i \overset{iid}{\sim} N(0, \sigma^2)$

What is important to know from all that??

We **fit** this model to data. That is, find the **best** estimators of β_0 and β_1 (and σ^2) given the data. How to fit the line?



Fitting the line

The (fitted) linear regression model uses $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$. Calculus allows us to find the 'least squares' estimators, $\hat{\beta}_0$ and $\hat{\beta}_1$ in a nice closed-form!

Making Inference

What hypothesis are we interested in and why?

How can we form a confidence interval for the quantity of interest?

Checking assumptions

How can we check our assumptions on the errors?

Fitting a Linear Regression Model in R

We can fit the model with the `lm()` function. Provide a `formula`

$$response \sim explanatory_variable_equation (\text{intercept fit by default})$$

```
library(tidyverse)
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))

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

Determine the fitted model by looking at the `coefficients` element.

```
fit$coefficients
```

```
##   (Intercept) log_km_driven
##    14.6355683    -0.3910865
```

Look at the hypothesis test of interest with `summary()`

```
summary(fit)
```

```
##
## 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 here is important and why?

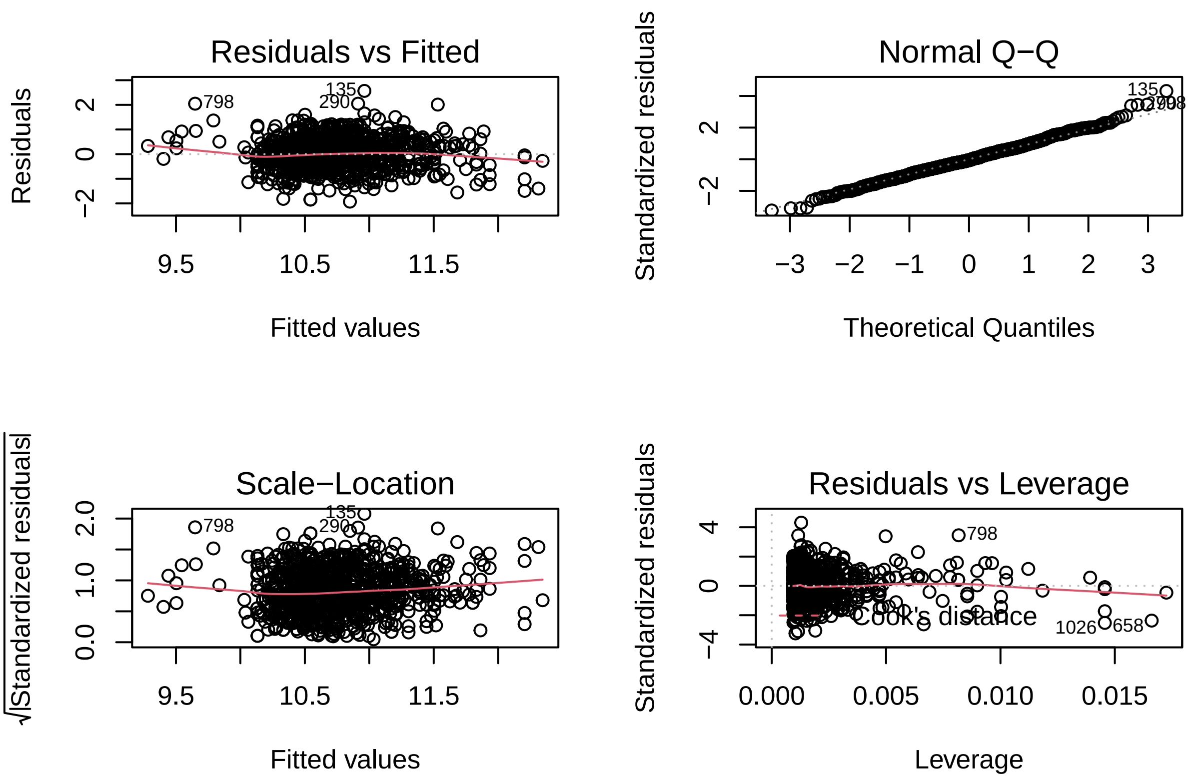
Find a confidence interval with `confint()`

```
confint(fit)
```

```
##                    2.5 %     97.5 %
## (Intercept)   14.2734501 14.9976864
## log_km_driven -0.4271342 -0.3550389
```

Check conditions! `plot()` on the model fit will work.

```
par(mfrow = c(2,2))
plot(fit)
```



Logistic Regression Model

Used when you have a **binary** response variable

- Using SLR is not appropriate!

Example:

- Consider data about water potability

```
library(tidyverse)
water <- read_csv("water_potability.csv")
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>
```

- Summarize water potability

```
table(water$Potability)
```

```
##
##    0    1
## 1998 1278
```

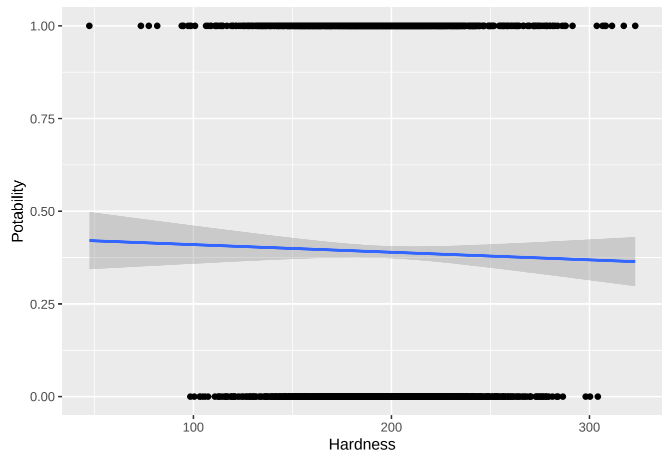
```
water %>%
  group_by(Potability) %>%
  select(Hardness, Chloramines, Potability) %>%
  summarize(meanH = mean(Hardness), meanC = mean(Chloramines))
```

```
## # A tibble: 2 x 3
##   Potability meanH meanC
##        <dbl> <dbl> <dbl>
## 1          0  197.  7.09
## 2          1  196.  7.17
```

Why is linear regression not appropriate?

```r
fit <- lm(Potability ~ Hardness, data = water)
ggplot(water, aes(x = Hardness, y = Potability)) +
  geom_point() +
  geom_smooth(method = "lm")
```

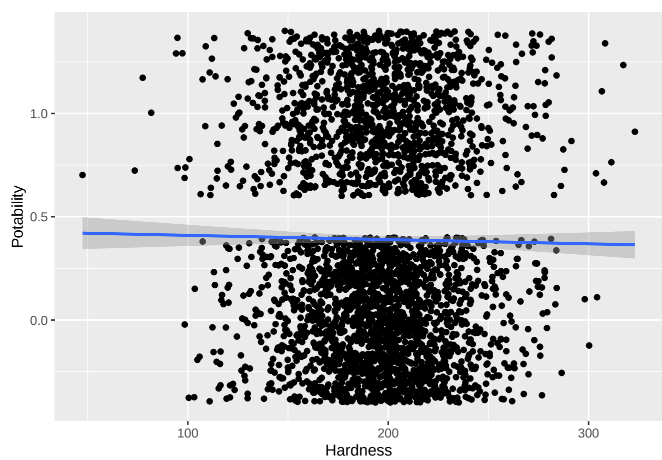
```
## `geom_smooth()` using formula 'y ~ x'
```



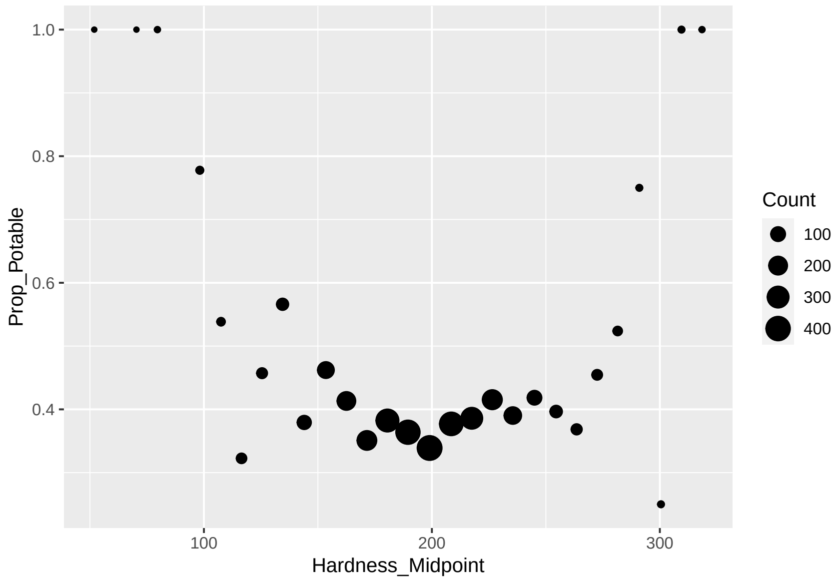
Better view...

```r
fit <- lm(Potability ~ Hardness, data = water)
ggplot(water, aes(x = Hardness, y = Potability)) +
  geom_jitter() +
  geom_smooth(method = "lm")
```

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



An even better view of the data is to visualize the proportions of successes as a function of hardness.

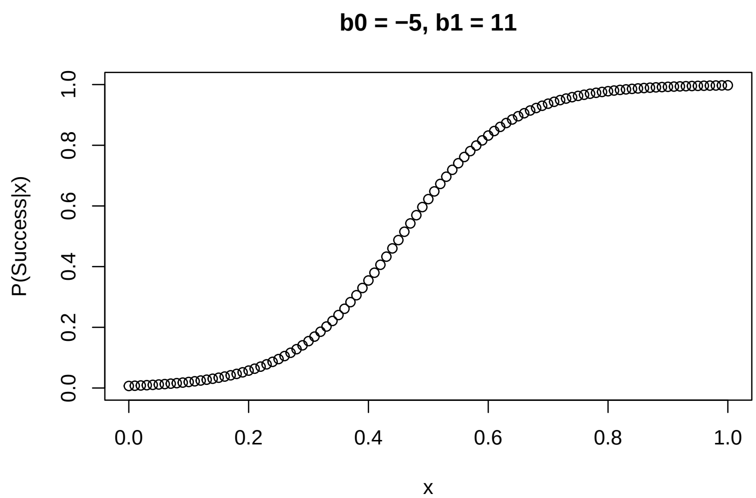


- In SLR, we modeled the average of the response as a linear function. What does the average of the responses represent here? Why does using a linear function not make sense?

- Basic Logistic Regression models success probability using the *logistic function*

$$P(success|x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$$

- This function never goes below 0 and never above 1 - works great for many applications!
- The logistic regression model doesn't have a closed form solution (maximum likelihood often used to fit parameters)



- Back-solving shows the *logit* or *log-odds* of success is linear in the parameters!

- Coefficient interpretation changes greatly from linear regression model!
- β_1 represents a change in the **log-odds of success**

Hypotheses of Interest

What do you think would indicate that x is related to the probability of success here?

Fitting a Logistic Regression Model in R

Fit in R using `glm()` with `family = binomial` and a formula just like `lm()`.

```
fit <- glm(Potability ~ Hardness, data = water, family = "binomial")
```

Get coefficients by looking at `coefficients` element:

```
fit$coefficients
```

```
##   (Intercept)      Hardness
## -0.2774792831 -0.0008629619
```

Get hypothesis test via `summary()`:

```
summary(fit)
```

```
##
## Call:
## glm(formula = Potability ~ Hardness, family = "binomial", data = water)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0279  -0.9963  -0.9853   1.3678   1.4209
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.277479   0.216758  -1.280    0.200
## Hardness    -0.000863   0.001090  -0.792    0.428
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4382.0  on 3275  degrees of freedom
## Residual deviance: 4381.3  on 3274  degrees of freedom
## AIC: 4385.3
##
## Number of Fisher Scoring iterations: 4
```

Get confidence interval for β_1 with:

```r
confint(fit)
```

```
## Waiting for profiling to be done...
```

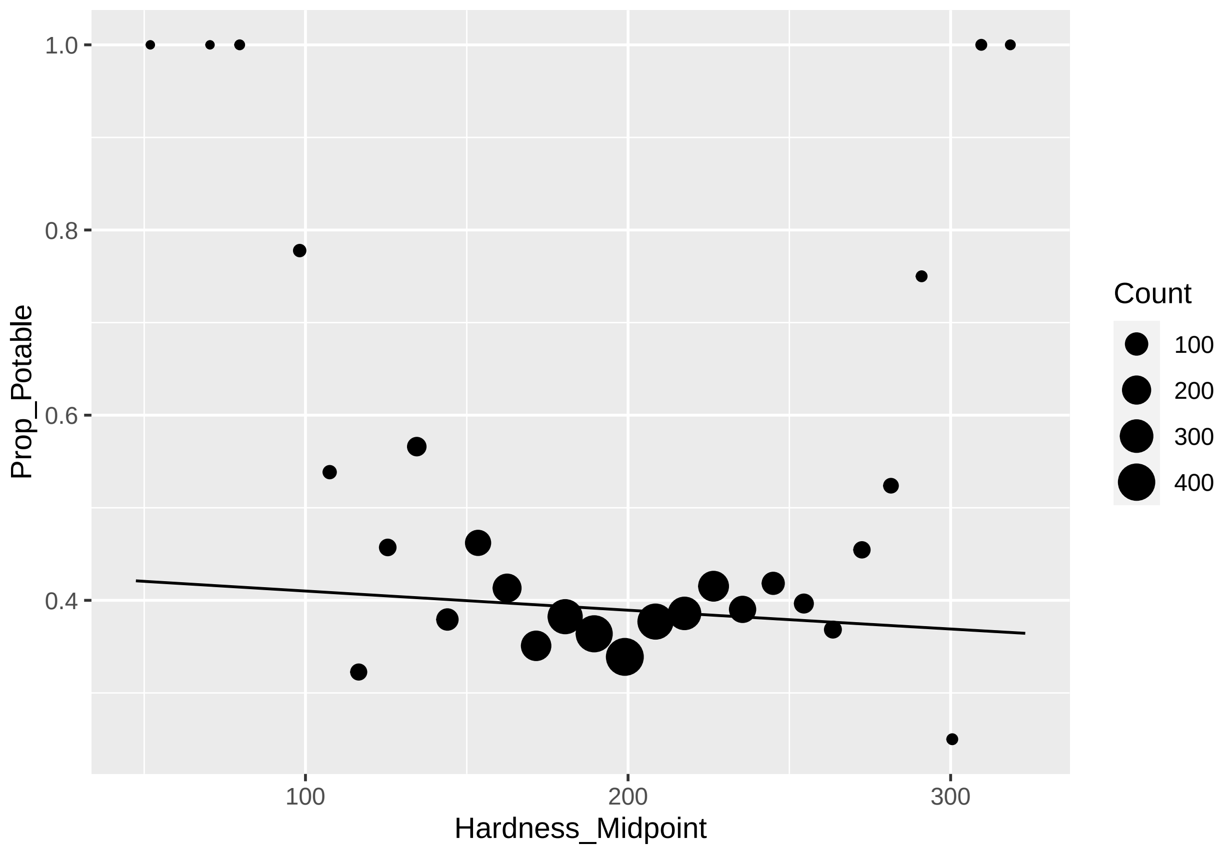
```
##                    2.5 %      97.5 %
## (Intercept) -0.702803063 0.147169863
## Hardness    -0.003000628 0.001272738
```

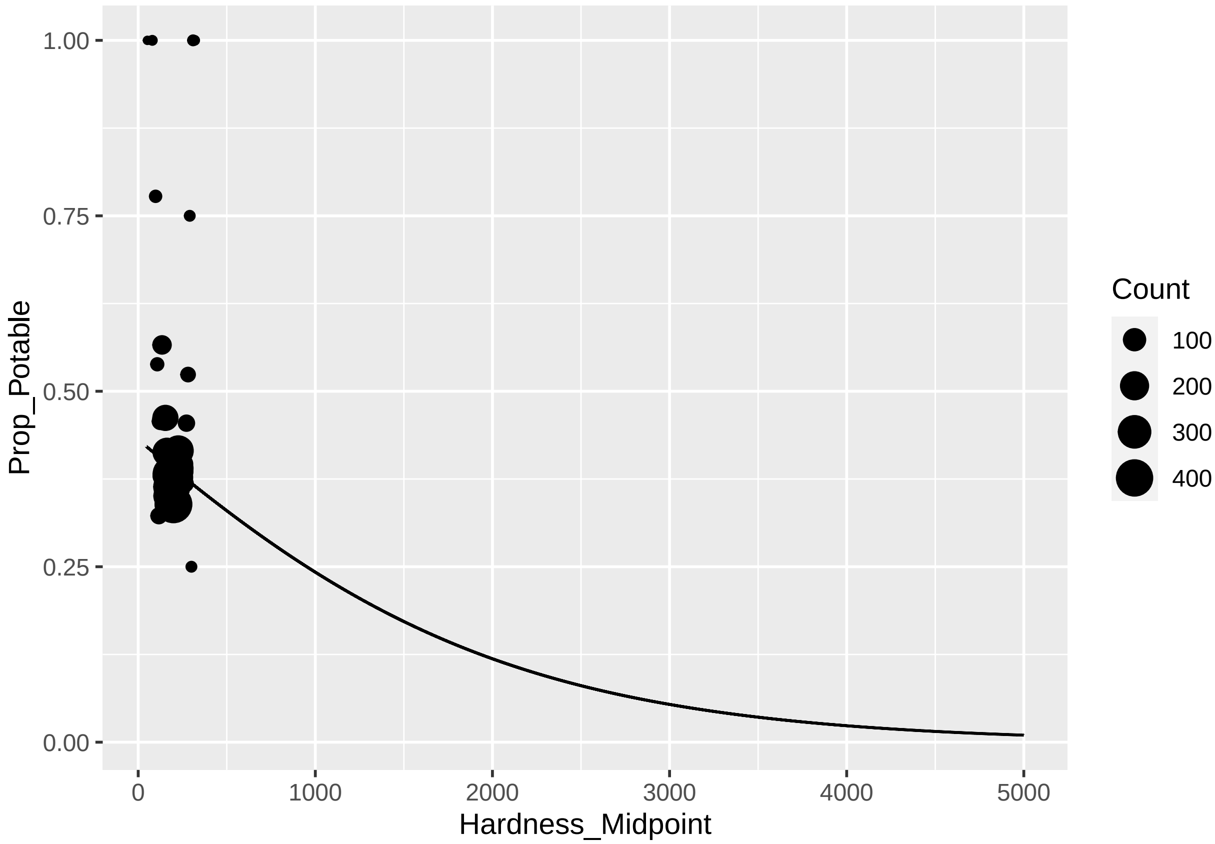
If we want a probability estimate back, use `predict()` with `type = 'response'`:

```r
predict(fit, newdata = data.frame(Hardness = c(200, 300)), type = "response", se.fit = TRUE)
```

```
## $fit
##         1         2
## 0.3893437 0.3690329
##
## $se.fit
##          1          2
## 0.00857465 0.02763731
##
## $residual.scale
## [1] 1
```

Visualize the fit:


1.0
0.8
0.6
0.4
Prop_Potable
100
200
300
Hardness_Midpoint
Count
100
200
300
400



Is a logistic curve!

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)
```

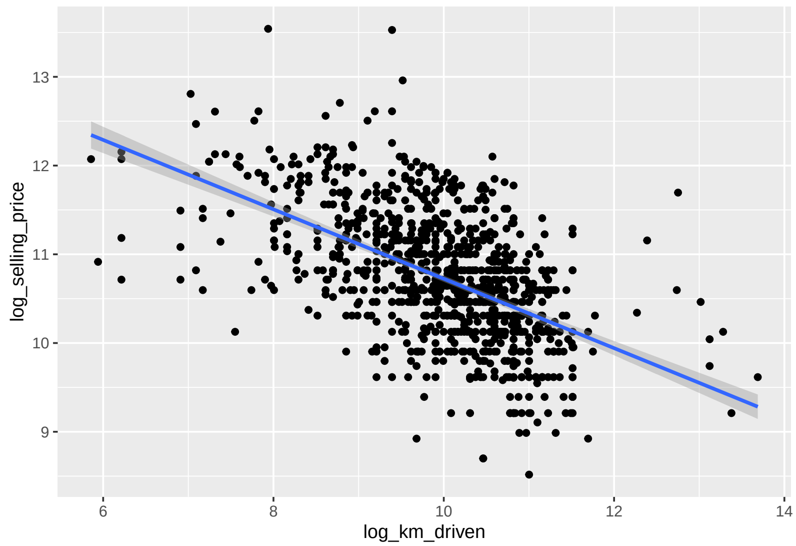
```
##
## Call:
## lm(formula = log_selling_price ~ year, data = bikeData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2917 -0.3814 -0.0948  0.2368  3.2436
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.011e+02  7.892e+00  -25.48   <2e-16 ***
## 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:

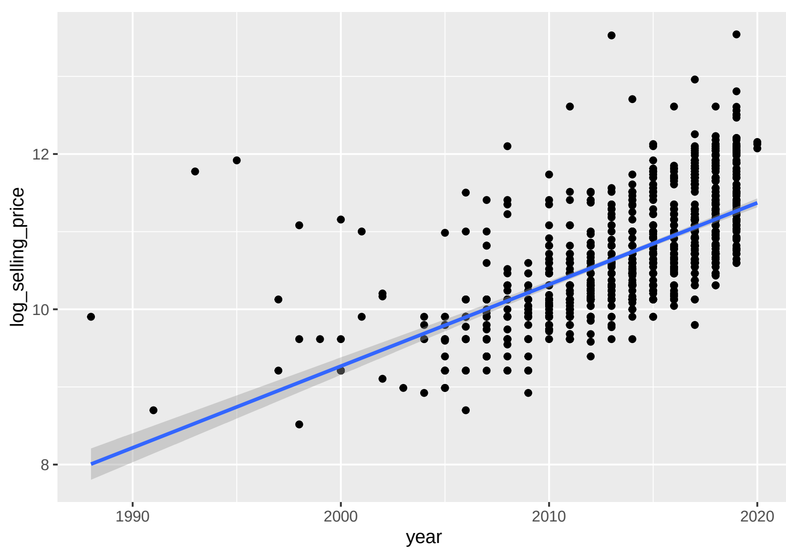
```
ggplot(bikeData, aes(x = log_km_driven, y = log_selling_price)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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




```
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)!

Fitting the model in R

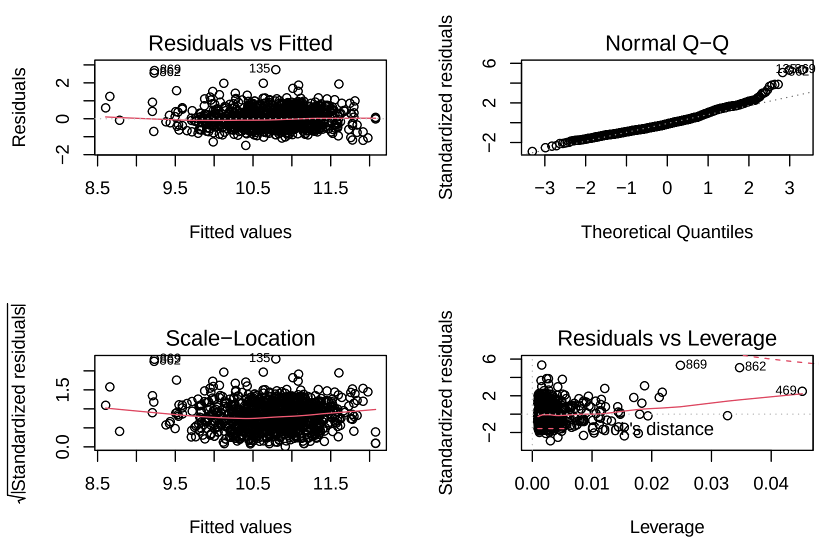
Just add to the right-hand side of our equation!

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

```
##
## 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
```

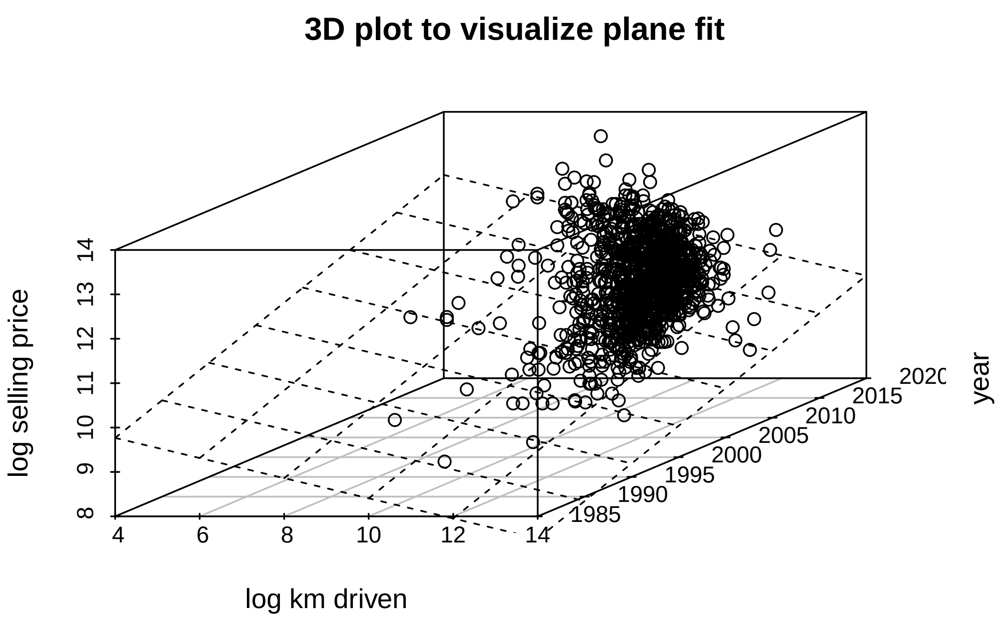
Check assumptions as before:

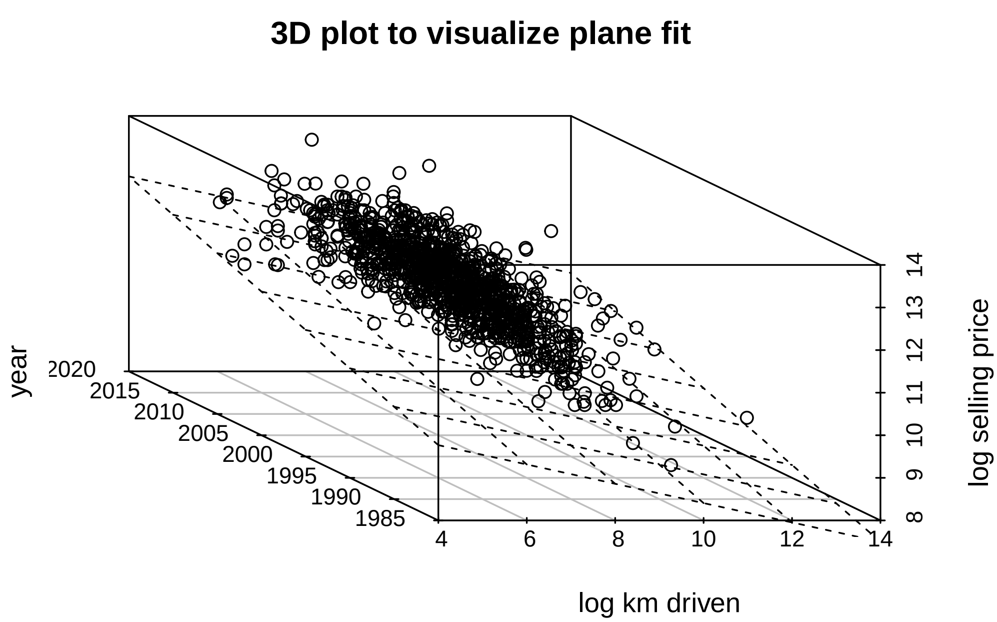
```
par(mfrow = c(2,2))
plot(mlr_fit)
```



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
```

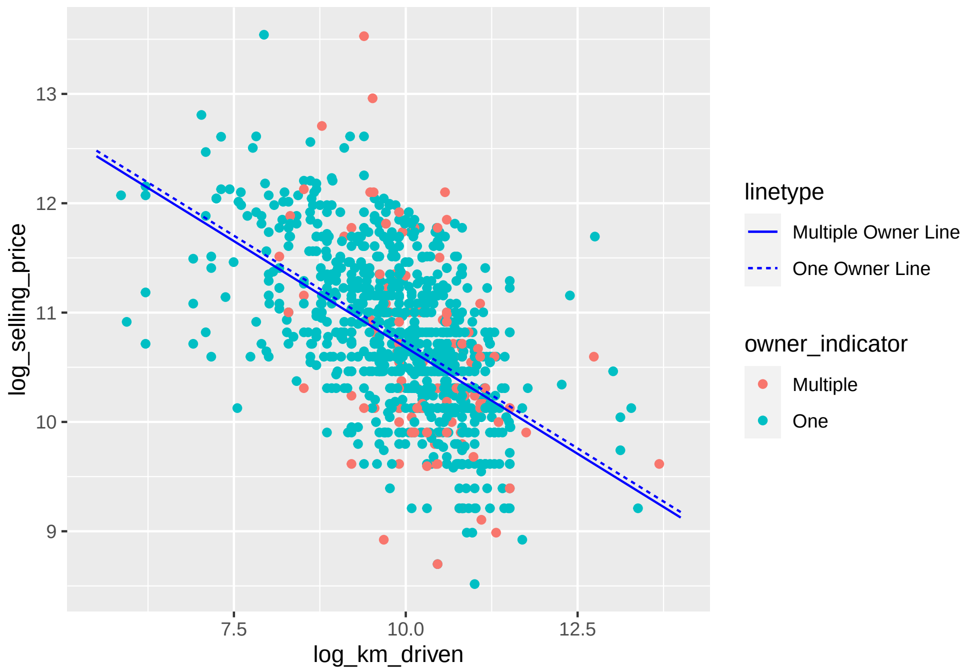
Add this to one of the SLR models:

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

```
##
## 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
```

What does `owner_indicatorOne` mean?

What does this do to our model?



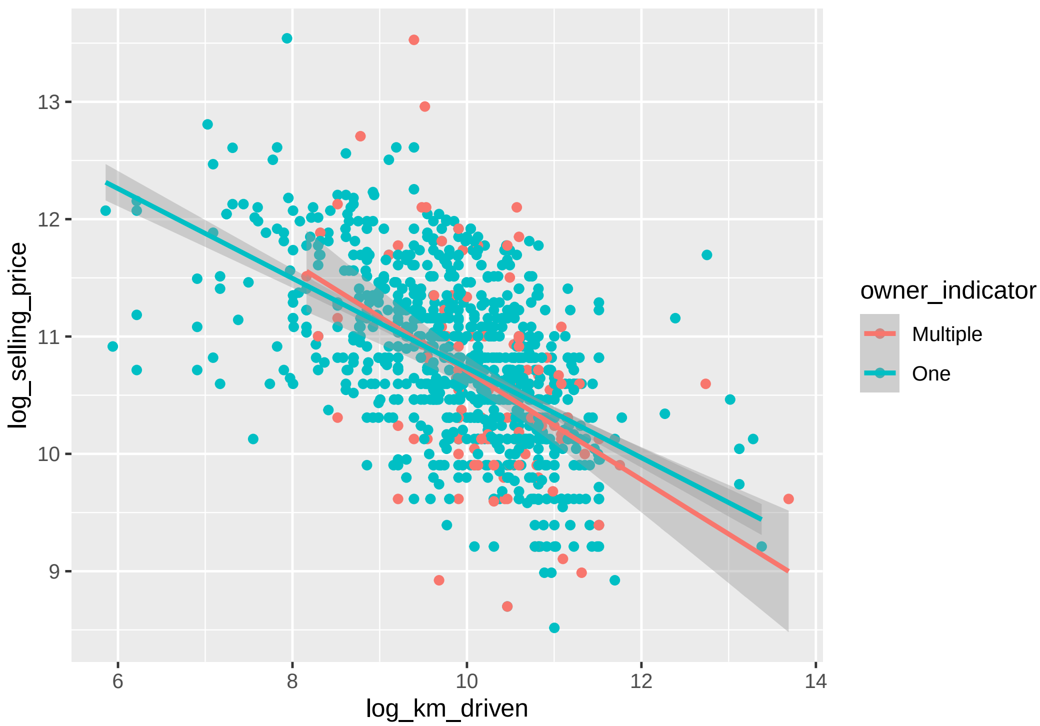
If we add an **interaction term** we get completely different lines:

```
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)
```

```
##
## 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
```

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

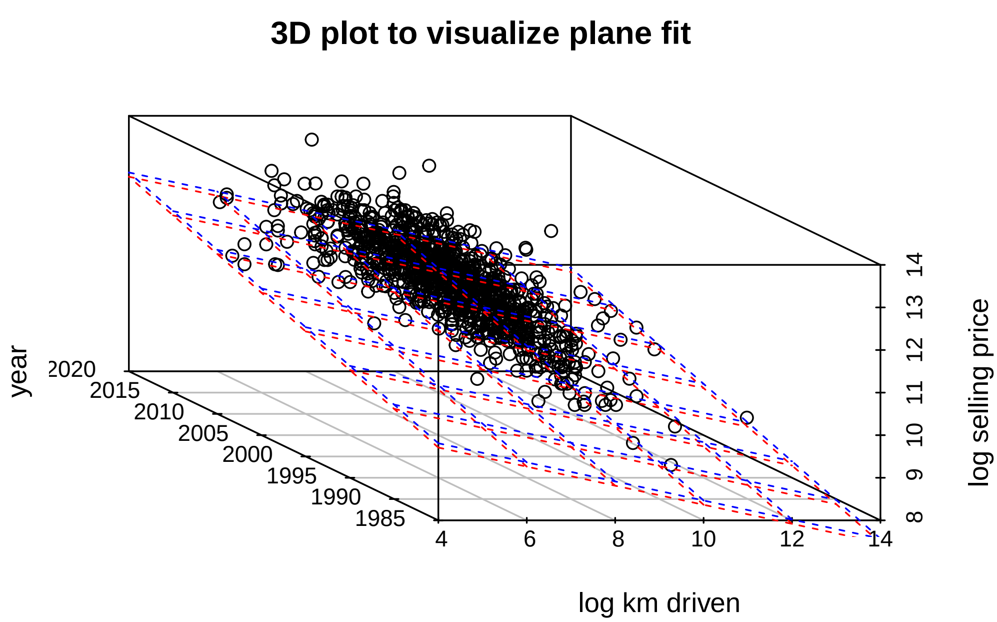
```
## `geom_smooth()` using formula 'y ~ x'
```



Note: The same idea works for the earlier MLR model!

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

```
##
## 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
```



Logistic Regression

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

```
water
```

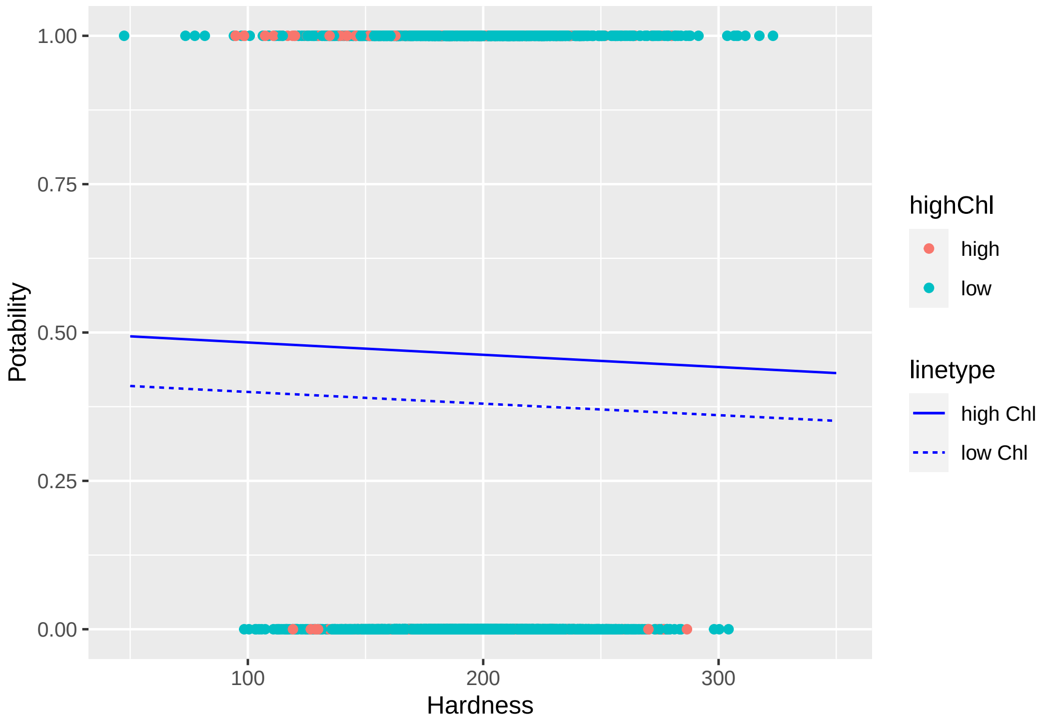
```
## # A tibble: 3,276 x 10
##       ph Hardness Solids Chloramines Sulfate Conductivity Organic_carbon
##    <dbl>    <dbl>  <dbl>       <dbl>   <dbl>        <dbl>          <dbl>
##  1 NA        205. 20791.        7.30    369.         564.          10.4
##  2  3.72     129. 18630.        6.64     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>
```

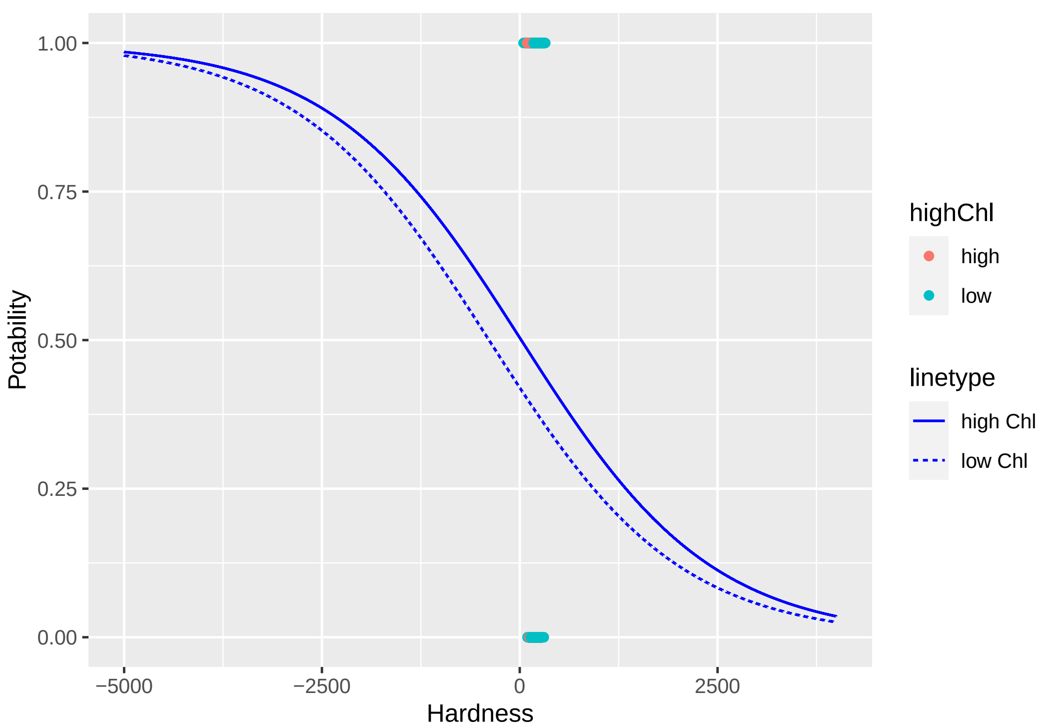
```
water <- water %>%
  mutate(highChl = ifelse(Chloramines > 9, "high", "low"))

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

```
##
## 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...

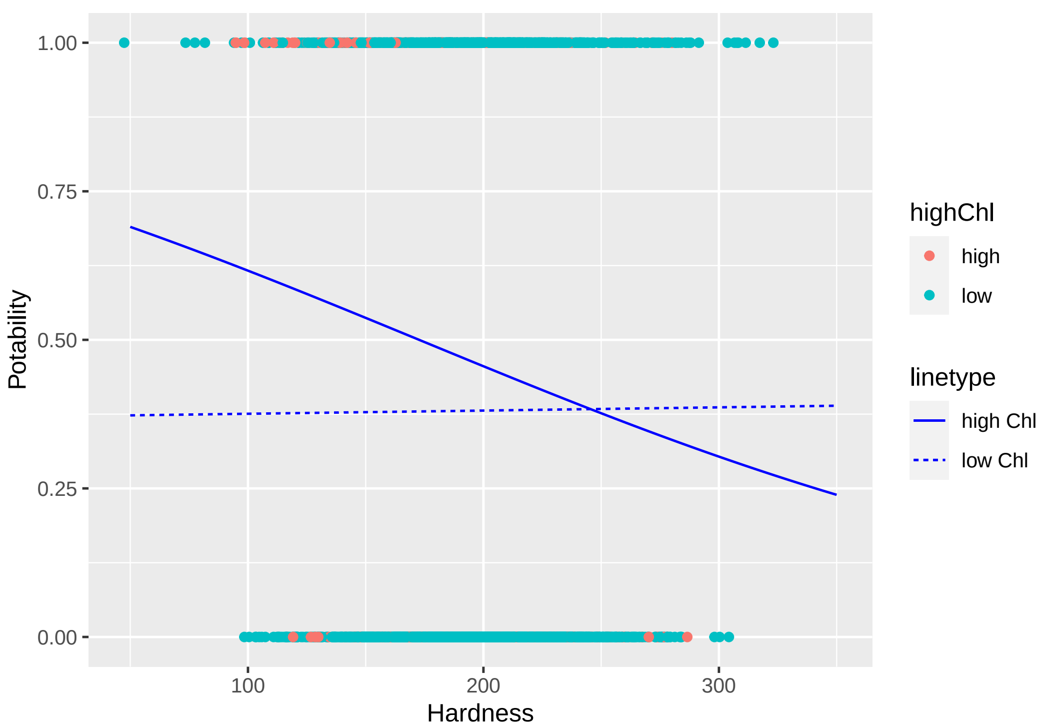




If we include an interaction between `Hardness` and `highChl` we get two separate logistic curves fit (one for the high group and one for the low group).

```
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: