Multiple Linear Regression

Justin Post

Recap

Given a model, we **fit** the model using data

- Must determine how well the model predicts on **new** data
- Create a test set
- Judge effectiveness using a **metric** on predictions made from the model

Regression Modeling Ideas

For a set of observations $y_1, \ldots, y_n$, we may want to predict a future value

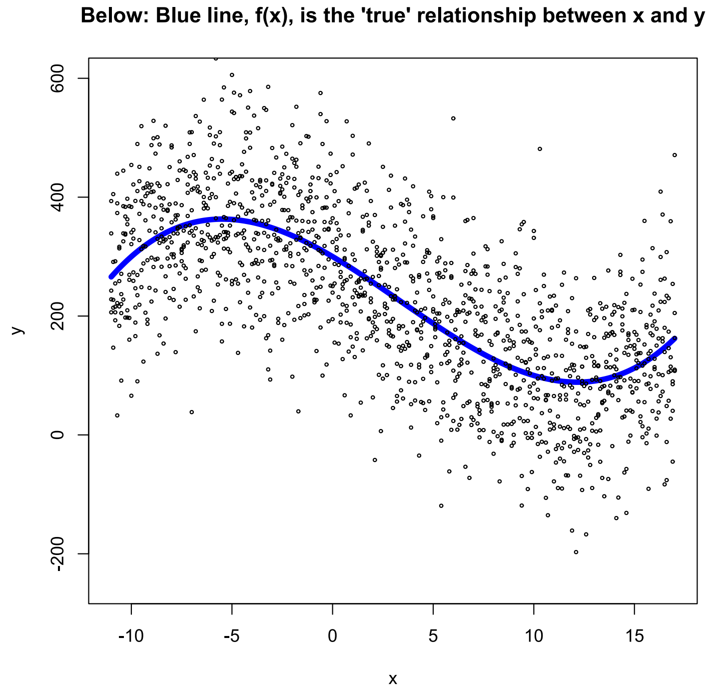
- Often use the sample mean to do so, $\bar{y}$ (an estimate of $E(Y)$)

Regression Modeling Ideas

For a set of observations $y_1, \ldots, y_n$, we may want to predict a future value

- Often use the sample mean to do so, $\bar{y}$ (an estimate of $E(Y)$)

Now consider having pairs $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$

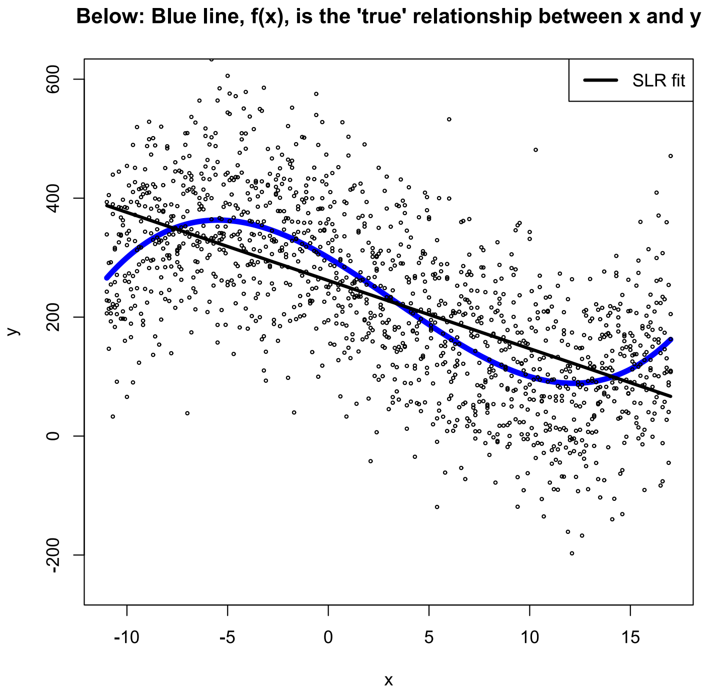


NC STATE UNIVERSITY

Regression Modeling Ideas

Often use a linear (in the parameters) model for prediction

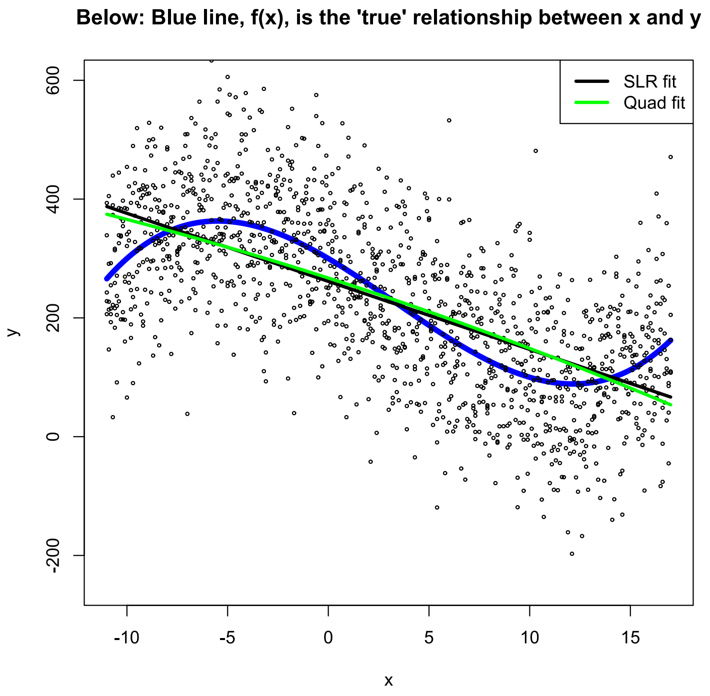
$$\text{SLR model: } E(Y|x) = \beta_0 + \beta_1 x$$



Regression Modeling Ideas

Can include more terms on the right hand side (RHS)

$$\text{Multiple Linear Regression Model: } E(Y|x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

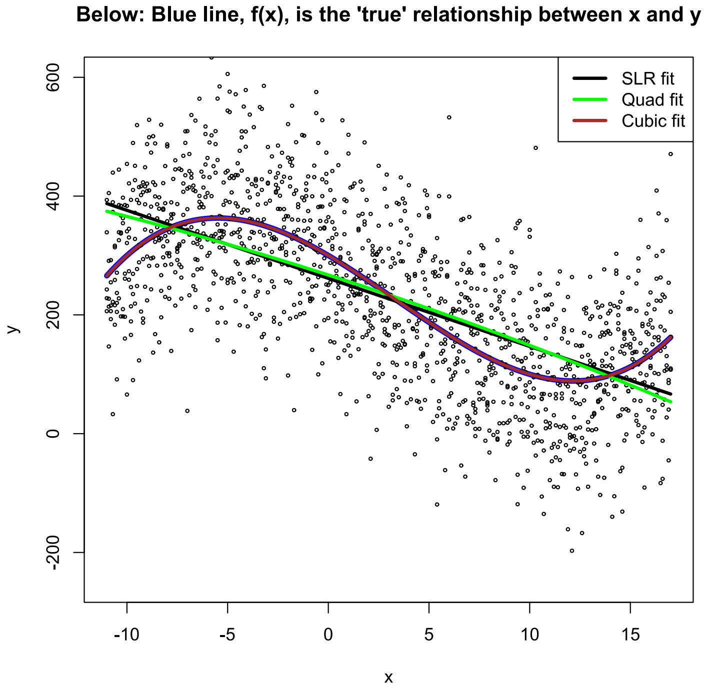


NC STATE UNIVERSITY

Regression Modeling Ideas

Can include more terms on the right hand side (RHS)

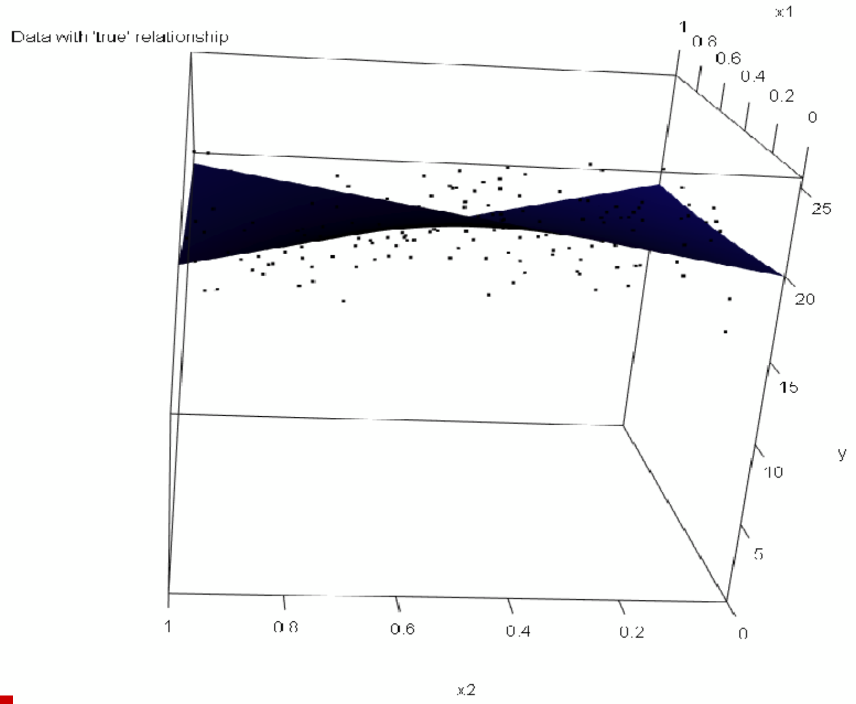
$$\text{Multiple Linear Regression Model: } E(Y|x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$



NC STATE UNIVERSITY

Regression Modeling Ideas

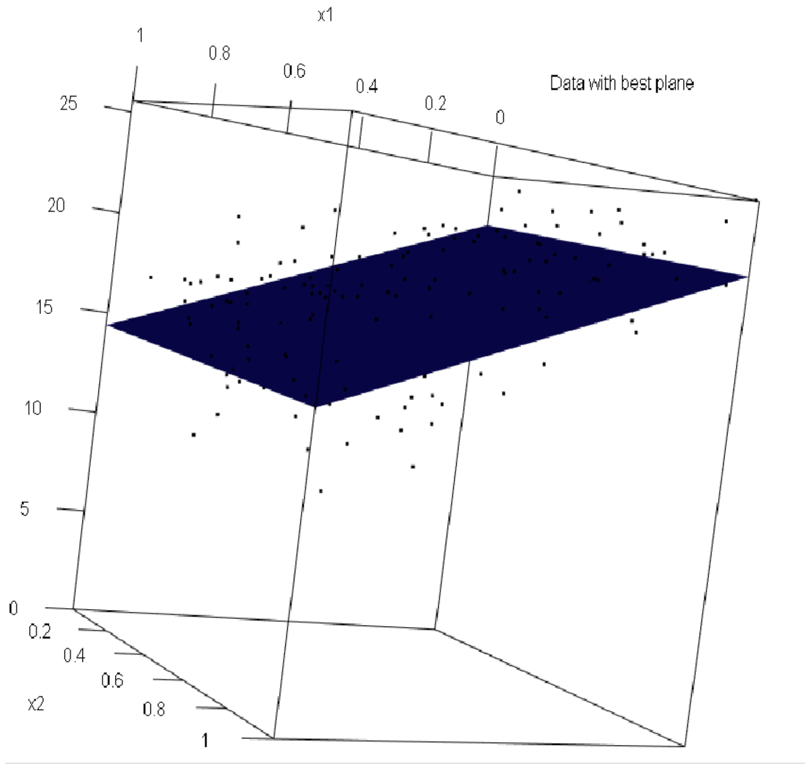
- We model the mean response for a given x value
- With multiple predictors or x's, we do the same idea!



Regression Modeling Ideas

- Including a **main effect** for two predictors fits the best plane through the data

$$\text{Multiple Linear Regression Model: } E(Y|x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

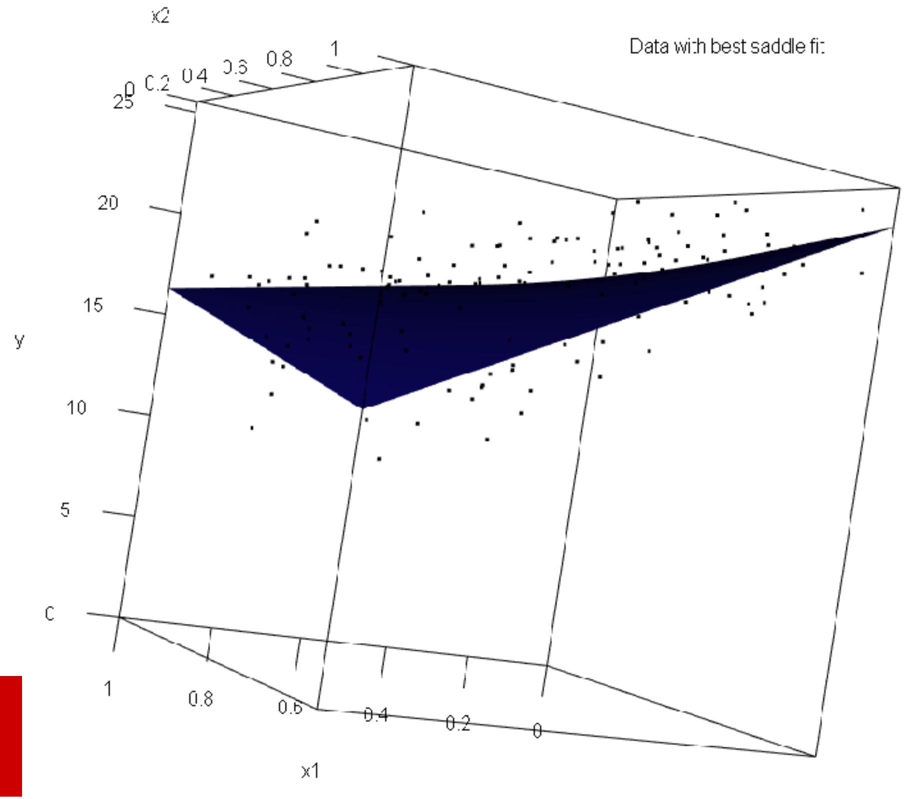


NC STATE UNIVERSITY

Regression Modeling Ideas

- Including **main effects** and an **interaction effect** allows for a more flexible surface

$$\text{Multiple Linear Regression Model: } E(Y|x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$



NC STATE UNIVERSITY

Regression Modeling Ideas

- Including **main effects** and an **interaction effect** allows for a more flexible surface
- Interaction effects allow for the **effect** of one variable to depend on the value of another
- Model fit previously gives
  - $\hat{y}$ = (19.005) + (-0.791)x1 + (5.631)x2 + (-12.918)x1x2

Regression Modeling Ideas

- Including **main effects** and an **interaction effect** allows for a more flexible surface
- Interaction effects allow for the **effect** of one variable to depend on the value of another
- Model fit previously gives
  - $\hat{y}$ = (19.005) + (-0.791)x1 + (5.631)x2 + (-12.918)x1x2
  - For x_1 = 0, the slope on x_2 is (5.631)`+0*` (-12.918) = 5.631

NC STATE UNIVERSITY

Regression Modeling Ideas

- Including **main effects** and an **interaction effect** allows for a more flexible surface
- Interaction effects allow for the **effect** of one variable to depend on the value of another
- Model fit previously gives
  - $\hat{y}$ = (19.005) + (-0.791)x1 + (5.631)x2 + (-12.918)x1x2
  - For x_1 = 0, the slope on x_2 is (5.631)`+0*` (-12.918) = 5.631
  - For x_1 = 0.5, the slope on x_2 is (5.631)`+0.5*`(-12.918) = -0.828

NC STATE UNIVERSITY

Regression Modeling Ideas

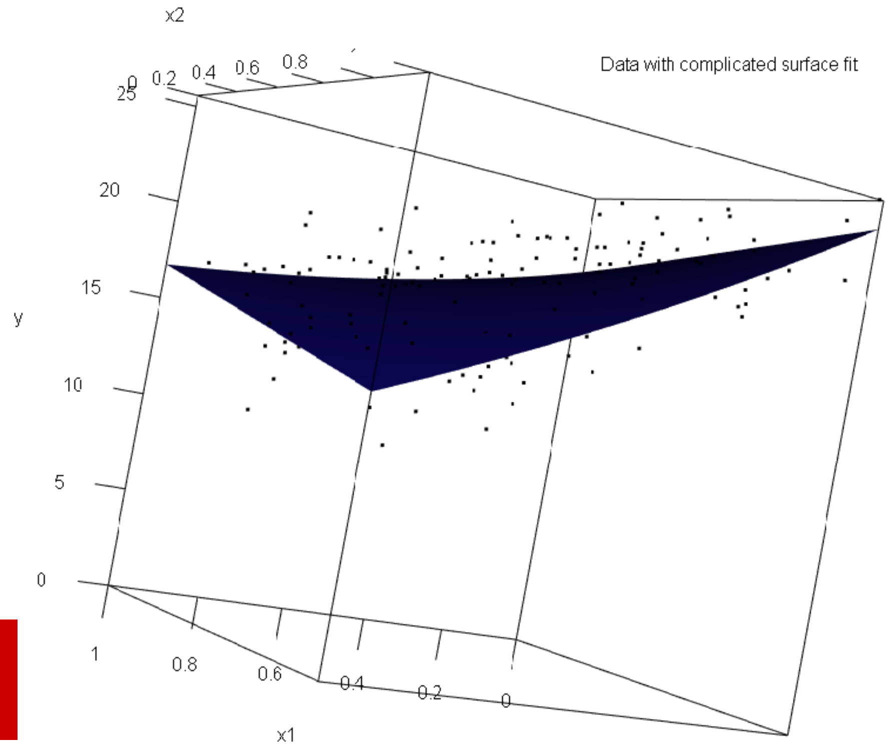
- Including **main effects** and an **interaction effect** allows for a more flexible surface
- Interaction effects allow for the **effect** of one variable to depend on the value of another
- Model fit previously gives
  - $\hat{y}$ = (19.005) + (-0.791)x1 + (5.631)x2 + (-12.918)x1x2
  - For x_1 = 0, the slope on x_2 is (5.631)`+0*` (-12.918) = 5.631
  - For x_1 = 0.5, the slope on x_2 is (5.631)`+0.5*`(-12.918) = -0.828
  - For x_1 = 1, the slope on x_2 is (5.631)`+1*`(-12.918) = -7.286
- Similarly, the slope on x_1 depends on x_2!

NC STATE UNIVERSITY

Regression Modeling Ideas

- Including **main effects** and an **interaction effect** allows for a more flexible surface
- Can also include higher order polynomial terms

Multiple Linear Regression Model: $E(Y|x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2$



NC STATE UNIVERSITY

Regression Modeling Ideas

Can also include categorical variables through **dummy** or **indicator** variables

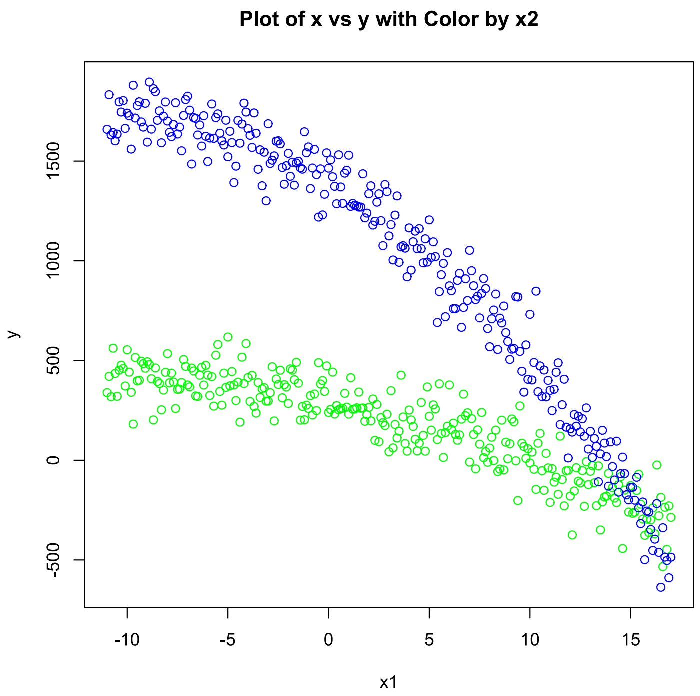
- Categorical variable with value of $Success$ and $Failure$
- Define $x_2 = 0$ if variable is $Failure$
- Define $x_2 = 1$ if variable is $Success$

NC STATE UNIVERSITY

Regression Modeling Ideas

Can also include categorical variables through **dummy** or **indicator** variables

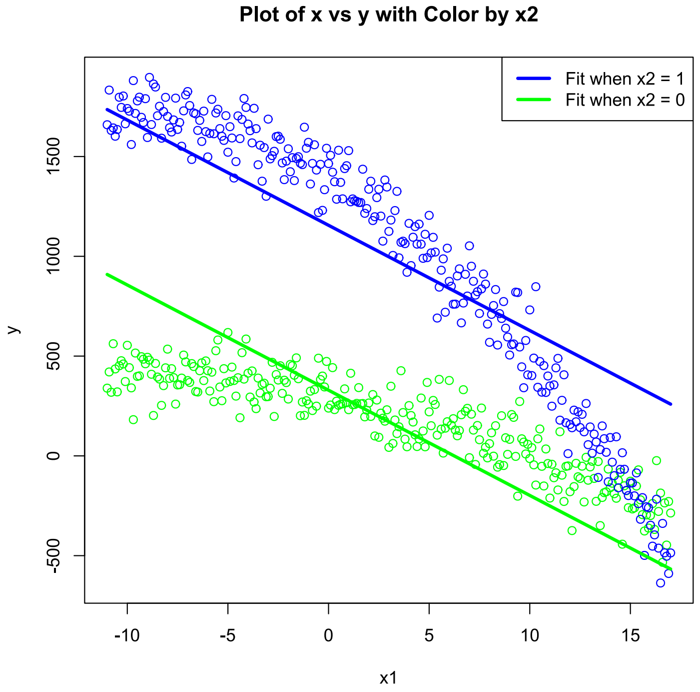
- Categorical variable with value of $Success$ and $Failure$
- Define $x_2 = 0$ if variable is $Failure$
- Define $x_2 = 1$ if variable is $Success$



Regression Modeling Ideas

- Define $x_2 = 0$ if variable is $Failure$
- Define $x_2 = 1$ if variable is $Success$

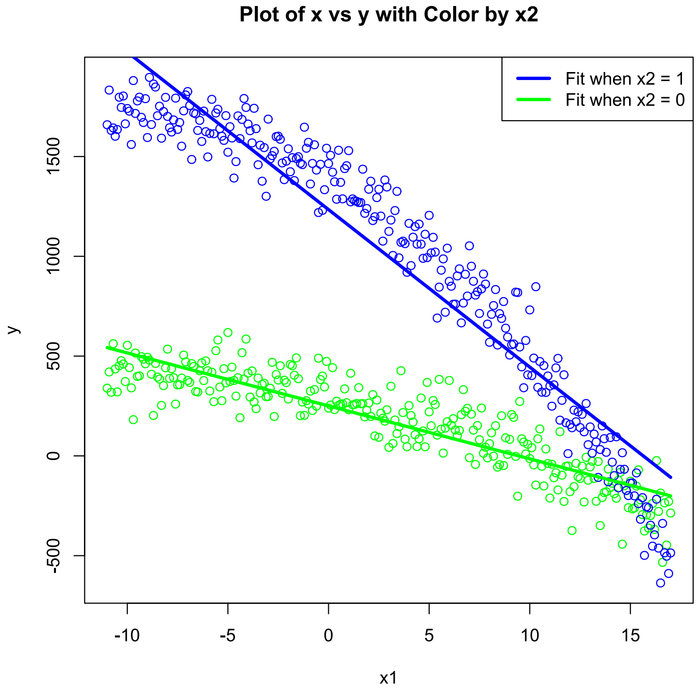
$$\text{Separate Intercept Model: } E(Y|x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$



Regression Modeling Ideas

- Define $x_2 = 0$ if variable is $Failure$
- Define $x_2 = 1$ if variable is $Success$

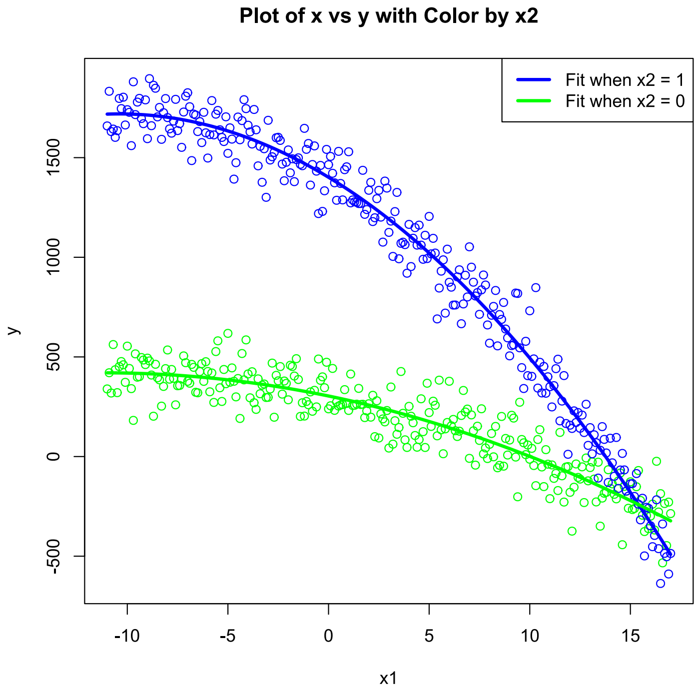
$$\text{Separate Intercept and Slopes Model: } E(Y|x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$



Regression Modeling Ideas

- Define $x_2 = 0$ if variable is $Failure$
- Define $x_2 = 1$ if variable is $Success$

Separate Quadratics Model: $E(Y|x) = \beta_0 + \beta_1 x_2 + \beta_2 x_1 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_1^2 x_2$



Regression Modeling Ideas

If your categorical variable has more than k>2 categories, define k-1 dummy variables

- Categorical variable with values of "Assistant", "Contractor", "Executive"
- Define $x_2 = 0$ if variable is $Executive$ or $Contractor$
- Define $x_2 = 1$ if variable is $Assistant$
- Define $x_3 = 0$ if variable is $Contractor$ or $Assistant$
- Define $x_3 = 1$ if variable is $Executive$

Regression Modeling Ideas

If your categorical variable has more than k>2 categories, define k-1 dummy variables

- Categorical variable with values of "Assistant", "Contractor", "Executive"
- Define $x_2 = 0$ if variable is $Executive$ or $Contractor$
- Define $x_2 = 1$ if variable is $Assistant$
- Define $x_3 = 0$ if variable is $Contractor$ or $Assistant$
- Define $x_3 = 1$ if variable is $Executive$

$$\text{Separate Intercepts Model: } E(Y|x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

What is implied if x_2 and x_3 are both zero?

Fitting an MLR Model

Big Idea: Trying to find the line, plane, saddle, etc. **of best fit** through points

- How do we do the fit??
  - Usually minimize the sum of squared residuals (errors)

Fitting an MLR Model

Big Idea: Trying to find the line, plane, saddle, etc. **of best fit** through points

- How do we do the fit??
  - Usually minimize the sum of squared residuals (errors)
- Residual = observed - predicted or $y_i - \hat{y}_i$

$$\min_{\hat{\beta}'s} \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_p x_{pi}))^2$$

Fitting an MLR Model

Big Idea: Trying to find the line, plane, saddle, etc. **of best fit** through points

- How do we do the fit??
  - Usually minimize the sum of squared residuals (errors)
- Residual = observed - predicted or $y_i - \hat{y}_i$

$$\min_{\hat{\beta}'s} \sum_{i=1}^{n} (y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \ldots + \hat{\beta}_p x_{pi}))^2$$

- Closed-form results exist for easy calculation via software!

Fitting a Linear Regression Model in Python

- Use `sklearn` package
- Create a `LinearRegression()` instance
- Use the `.fit()` method to fit the model

Fitting a Linear Regression Model in Python

- Use `sklearn` package
- Create a `LinearRegression()` instance
- Use the `.fit()` method to fit the model

```
import pandas as pd
import numpy as np
bike_data = pd.read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
#create response and new predictor
bike_data['log_selling_price'] = np.log(bike_data['selling_price'])
bike_data['log_km_driven'] = np.log(bike_data['km_driven'])
```

Fitting a Linear Regression Model in Python

- Use `sklearn` package
- Create a `LinearRegression()` instance
- Use the `.fit()` method to fit the model

```
import pandas as pd
import numpy as np
bike_data = pd.read_csv("https://www4.stat.ncsu.edu/~online/datasets/bikeDetails.csv")
#create response and new predictor
bike_data['log_selling_price'] = np.log(bike_data['selling_price'])
bike_data['log_km_driven'] = np.log(bike_data['km_driven'])
```

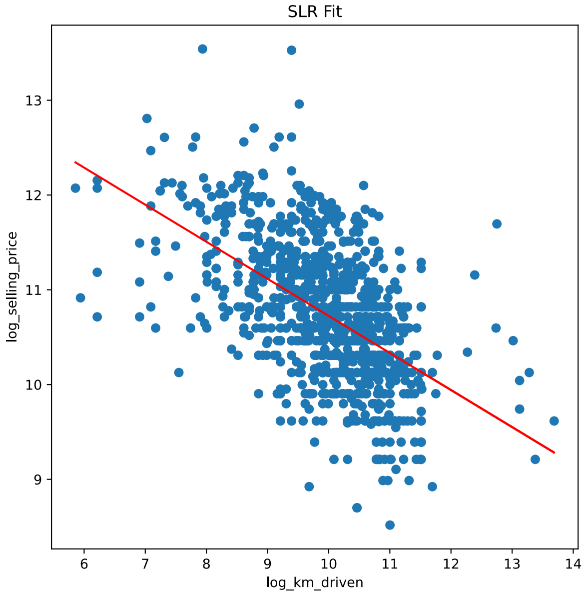
```
from sklearn import linear_model
slr_fit = linear_model.LinearRegression() #Create a reg object
slr_fit.fit(bike_data['log_km_driven'].values.reshape(-1,1), bike_data['log_selling_price'].values)
```

```
print(slr_fit.intercept_, slr_fit.coef_)
```

```
## 14.6355682846293 [-0.39108654]
```

Fitting a Linear Regression Model in Python

```
import matplotlib.pyplot as plt
preds = slr_fit.predict(bike_data['log_km_driven'].values.reshape(-1,1))
plt.scatter(bike_data['log_km_driven'].values.reshape(-1,1), bike_data['log_selling_price'].values)
plt.plot(bike_data['log_km_driven'].values.reshape(-1,1), preds, 'red')
plt.title("SLR Fit")
plt.xlabel('log_km_driven')
plt.ylabel('log_selling_price')
```



Add Dummy Variables for a Categorical Predictor

- `get_dummies()` function from `pandas` is useful

```
pd.get_dummies(data = bike_data['owner'])
```

```
##       1st owner  2nd owner  3rd owner  4th owner
## 0             1          0          0          0
## 1             1          0          0          0
## 2             1          0          0          0
## 3             1          0          0          0
## 4             0          1          0          0
## ...         ...        ...        ...        ...
## 1056          1          0          0          0
## 1057          1          0          0          0
## 1058          0          1          0          0
## 1059          1          0          0          0
## 1060          1          0          0          0
## 
## [1061 rows x 4 columns]
```

Add Dummy Variables for a Categorical Predictor

- `get_dummies()` function from `pandas` is useful

```
pd.get_dummies(data = bike_data['owner'])
```

```
##       1st owner  2nd owner  3rd owner  4th owner
## 0             1          0          0          0
## 1             1          0          0          0
## 2             1          0          0          0
## 3             1          0          0          0
## 4             0          1          0          0
## ...         ...        ...        ...        ...
## 1056          1          0          0          0
## 1057          1          0          0          0
## 1058          0          1          0          0
## 1059          1          0          0          0
## 1060          1          0          0          0
## 
## [1061 rows x 4 columns]
```

- If we use just the first variable created, we'll have a 1 owner vs more than 1 owner binary variable

Fit a Model with Dummy Variables

- Add the binary variable and fit the model

```
bike_data['one_owner'] = pd.get_dummies(data = bike_data['owner'])['1st owner']
mlr_fit = linear_model.LinearRegression() #Create a reg object
mlr_fit.fit(bike_data[['log_km_driven','one_owner']], bike_data['log_selling_price'].values)
```

```
print(mlr_fit.intercept_, mlr_fit.coef_)
```

```
## 14.570541645783868 [-0.38893985  0.05002779]
```

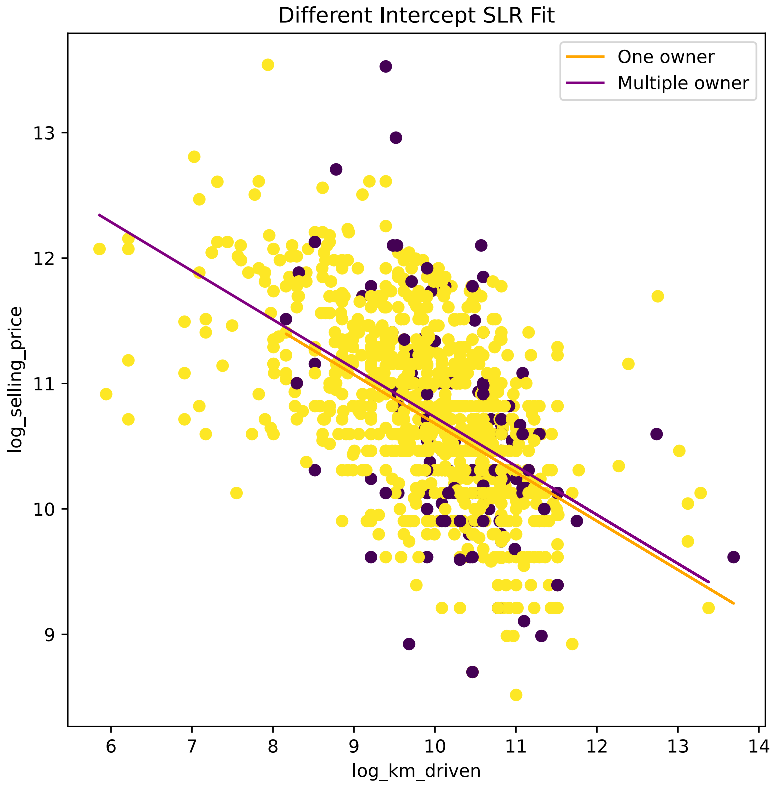
Fit a Model with Dummy Variables

Plot these fits on the same graph

```
preds1 = mlr_fit.predict(bike_data.loc[bike_data['one_owner'] == 0, ['log_km_driven', 'one_owner']])
preds2 = mlr_fit.predict(bike_data.loc[bike_data['one_owner'] == 1, ['log_km_driven', 'one_owner']])
plt.scatter(bike_data['log_km_driven'].values.reshape(-1,1), bike_data['log_selling_price'].values,
            c = bike_data['one_owner'].values)
plt.plot(bike_data.loc[bike_data['one_owner'] == 0, ['log_km_driven']], preds1,
         c = 'orange', label = "One owner")
plt.plot(bike_data.loc[bike_data['one_owner'] == 1, ['log_km_driven']], preds2,
         c = 'purple', label = "Multiple owner")
plt.title("Different Intercept SLR Fit")
plt.xlabel('log_km_driven')
plt.ylabel('log_selling_price')
plt.legend()
```

Fit a Model with Dummy Variables

Plot these fits on the same graph



Choosing an MLR Model

- Given a bunch of predictors, tons of models you could fit! How to choose?
- Many variable selection methods exist…
- If you care mainly about prediction, just use *cross-validation* or training/test split!

Compare Multiple Models

- We've seen how to split our data into a training and test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
  bike_data[["year", "log_km_driven", "one_owner"]],
  bike_data["log_selling_price"],
  test_size=0.20,
  random_state=42)
```

Compare Multiple Models

- We've seen how to split our data into a training and test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
  bike_data[["year", "log_km_driven", "one_owner"]],
  bike_data["log_selling_price"],
  test_size=0.20,
  random_state=42)
```

- Fit competing models on the training set

```
mlr_fit = linear_model.LinearRegression().fit(X_train[['log_km_driven','one_owner']], y_train)
slr_fit = linear_model.LinearRegression().fit(X_train['log_km_driven'].values.reshape(-1,1), y_train)
cat_fit = linear_model.LinearRegression().fit(X_train['one_owner'].values.reshape(-1,1), y_train)
```

Compare Multiple Models

- Look at training RMSE for comparison

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_train, mlr_fit.predict(X_train[["log_km_driven", "one_owner"]])))
```

```
## 0.5944388369922229
```

```
np.sqrt(mean_squared_error(y_train, slr_fit.predict(X_train["log_km_driven"].values.reshape(-1,1))))
```

```
## 0.594953681655801
```

```
np.sqrt(mean_squared_error(y_train, cat_fit.predict(X_train["one_owner"].values.reshape(-1,1))))
```

```
## 0.7014857593074377
```

Compare Multiple Models

- What we care about is test set RMSE though!

```
np.sqrt(mean_squared_error(y_test, mlr_fit.predict(X_test[["log_km_driven", "one_owner"]])))
```

```
## 0.5954962522276137
```

```
np.sqrt(mean_squared_error(y_test, slr_fit.predict(X_test["log_km_driven"].values.reshape(-1,1))))
```

```
## 0.594318016111905
```

```
np.sqrt(mean_squared_error(y_test, cat_fit.predict(X_test["one_owner"].values.reshape(-1,1))))
```

```
## 0.7319099074576736
```

Recap

- Multiple Linear Regression models are a common model used for a numeric response
- Generally fit via minimizing the sum of squared residuals or errors
  - Could fit using sum of absolute deviation, or other metric
- Can include polynomial terms, interaction terms, and categorical variables through dummy (indicator) variables
- Good metric to compare models on is the RMSE on a test set