

# GENERALIZED ADDITIVE MODELS

---

Dr. Aric LaBarr

Institute for Advanced Analytics

# GENERAL STRUCTURE

---

# Generalized Additive Models (GAMs)

- Provides **general** framework for **adding** non-linear functions to standard **linear model**.

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \varepsilon$$

- Can be used for regression or classification problems.

# Generalized Additive Models (GAMs)

- Provides **general** framework for **adding** non-linear functions to standard **linear model**.

$$y = \beta_0 + \boxed{f_1(x_1)} + \boxed{f_2(x_2)} + \cdots + \boxed{f_p(x_p)} + \varepsilon$$

- Can be used for regression or classification problems.



Adding **potentially** complex, individual relationships together.

# Generalized Additive Models (GAMs)

- Provides **general** framework for **adding** non-linear functions to standard **linear model**.

$$y = \beta_0 + \boxed{f_1(x_1)} + \boxed{f_2(x_2)} + \cdots + \boxed{f_p(x_p)} + \varepsilon$$

- Can be used for regression or classification problems.



Many potential complex relationships  
to try and model with.



# PIECEWISE LINEAR REGRESSION

---

# Changing Slopes

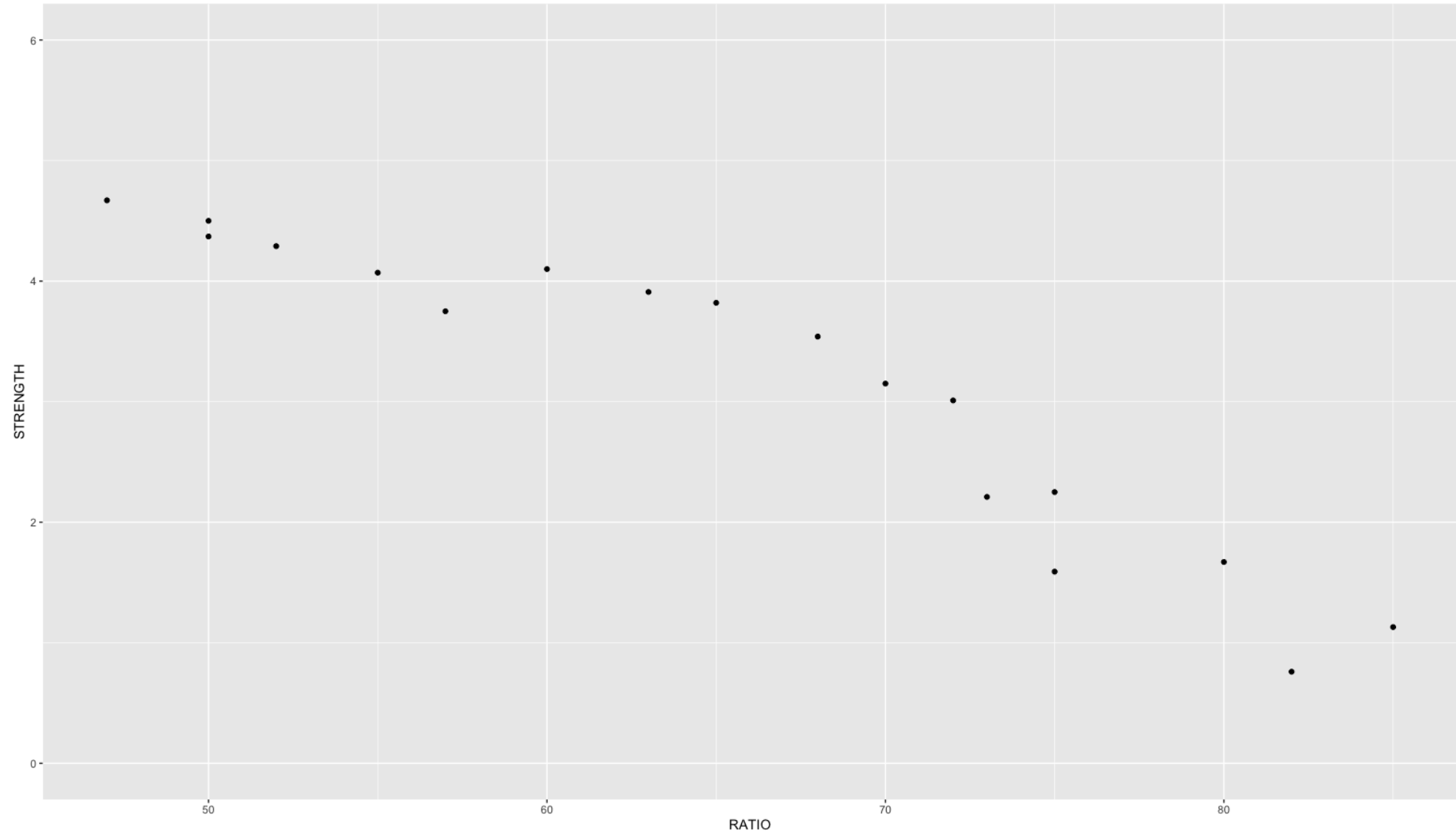
- The slope of the linear relationship between a predictor variable and a response variable can change over different values of the predictor variable.
- Typical straight-line model  $\hat{y} = \beta_0 + \beta_1 x_1$  will not be a good fit for this type of data.



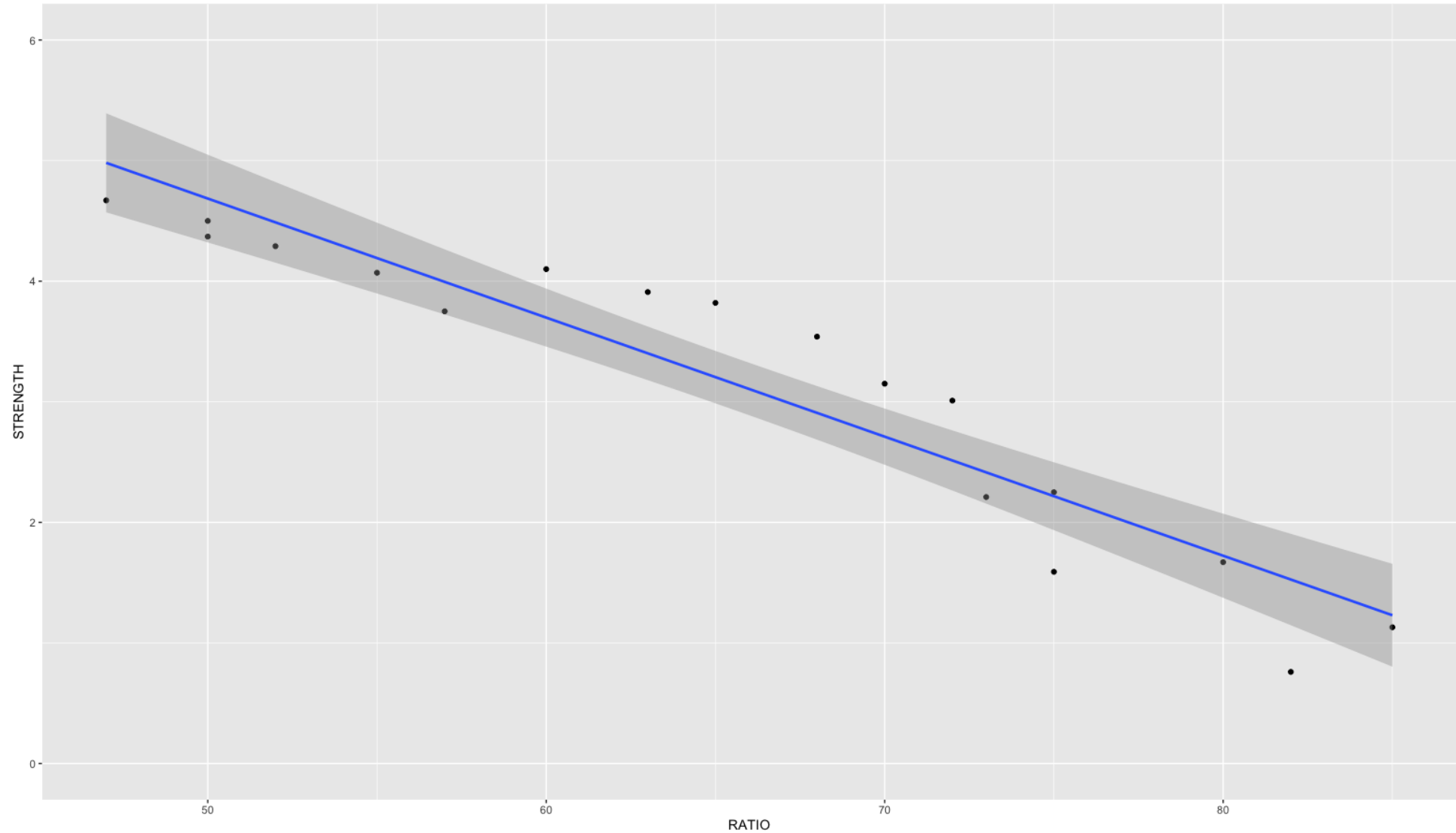
# Cement Data

- The comprehensive strength of concrete depends on the proportion of water mixed with cement.
- The comprehensive strength decreases at a much faster rate for batches with a greater than 70% water/cement ratio.

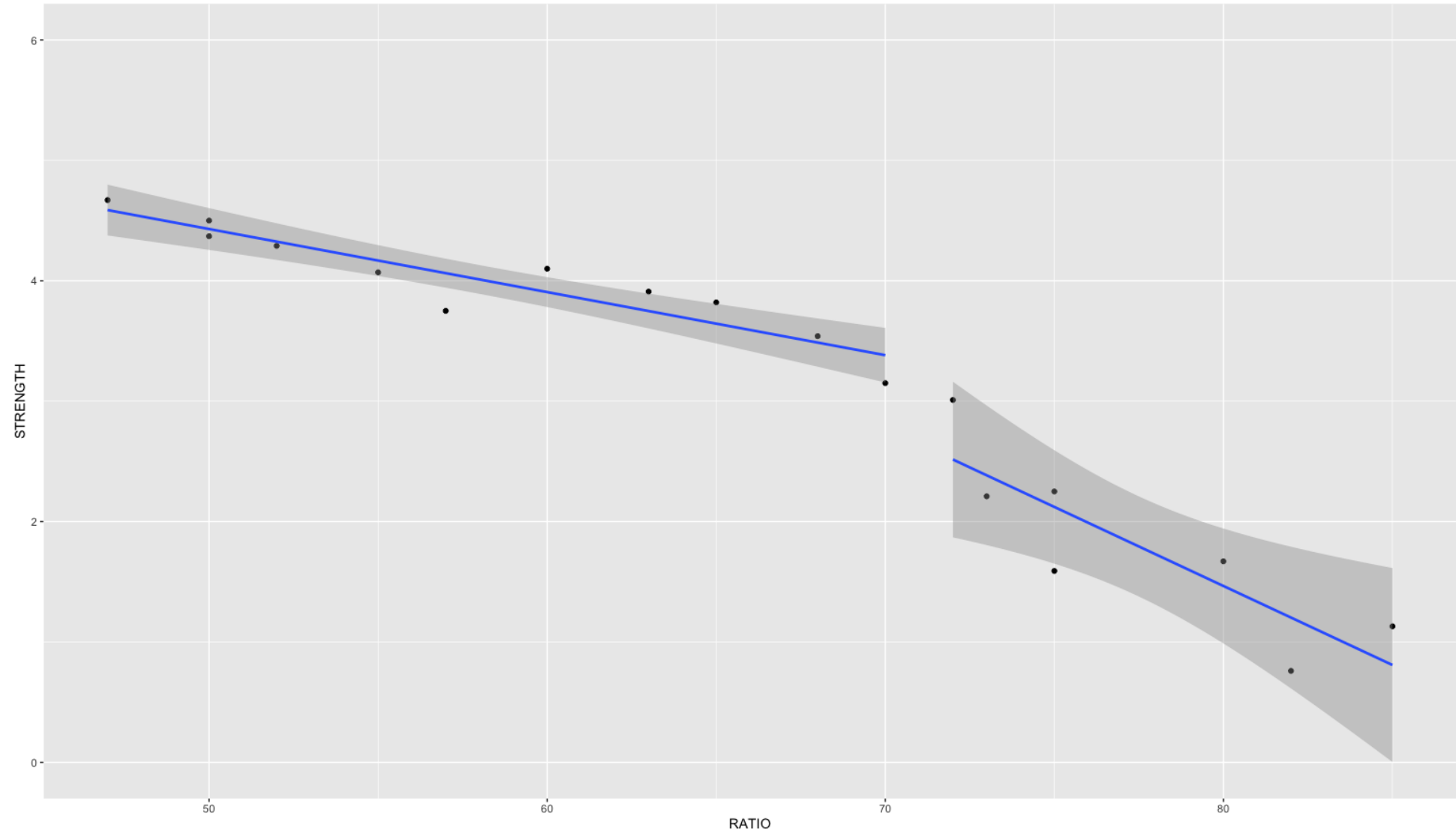
# Changing Slopes



# Changing Slopes



# Changing Slopes



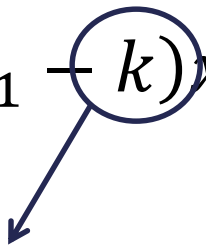
# Piecewise Linear Regression

- A model where different straight-line relationships for different intervals in the predictor variable is called the **piecewise linear regression model**.
- The model is the following for two slopes:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \varepsilon$$

# Piecewise Linear Regression

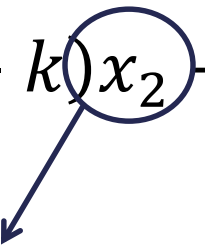
- A model where different straight-line relationships for different intervals in the predictor variable is called the **piecewise linear regression model**.
- The model is the following for two slopes:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \varepsilon$$


**Knot value** for  $x_1$ .

# Piecewise Linear Regression

- A model where different straight-line relationships for different intervals in the predictor variable is called the **piecewise linear regression model**.
- The model is the following for two slopes:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \varepsilon$$


$$x_2 = \begin{cases} 1, & x_1 > k \\ 0, & x_1 \leq k \end{cases}$$

# Piecewise Linear Regression

- A model where different straight-line relationships for different intervals in the predictor variable is called the **piecewise linear regression model**.
- The model is the following for two slopes:

$$x_2 = 0 \quad \begin{cases} y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \varepsilon \\ y = \beta_0 + \beta_1 x_1 + \varepsilon \end{cases}$$



# Piecewise Linear Regression

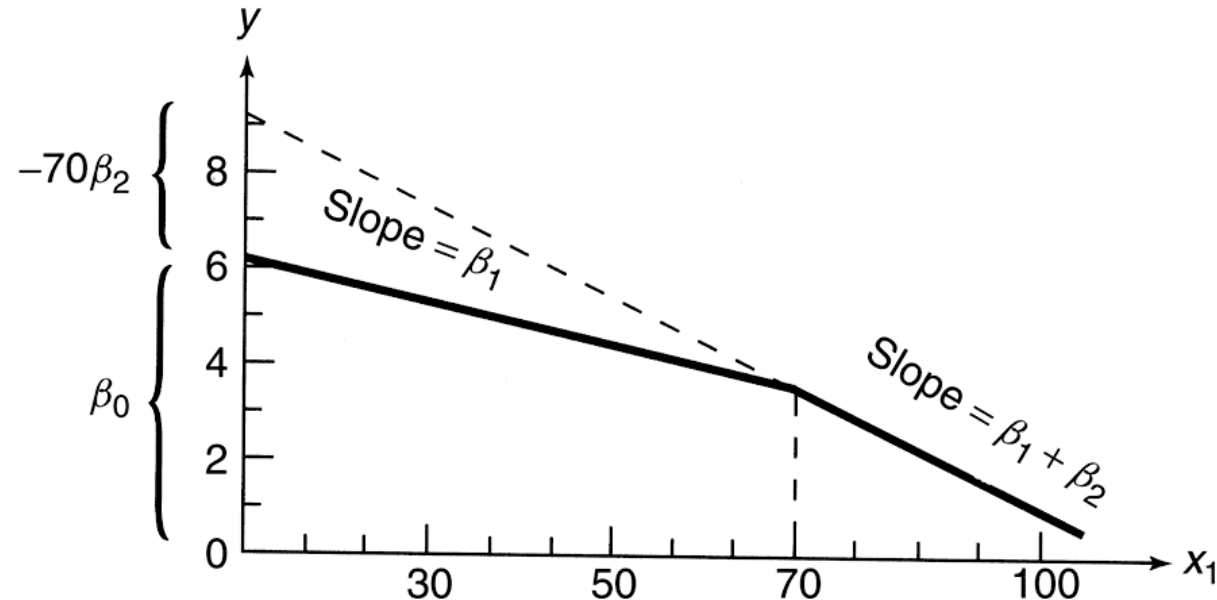
- A model where different straight-line relationships for different intervals in the predictor variable is called the **piecewise linear regression model**.
- The model is the following for two slopes:

The diagram illustrates the simplification of a piecewise linear regression equation for two values of the predictor variable  $x_2$ . A large curved arrow on the left points from the general equation at the top to the simplified equations at the bottom, with a smaller curved arrow pointing from the top equation to the middle equation.

$$\begin{array}{l} x_2 = 0 \\ x_2 = 1 \end{array} \quad \begin{array}{l} y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \varepsilon \\ y = \beta_0 + \beta_1 x_1 + \varepsilon \\ y = (\beta_0 - k\beta_2) + (\beta_1 + \beta_2) x_1 + \varepsilon \end{array}$$

# Cement Data

- The comprehensive strength of concrete depends on the proportion of water mixed with cement.
- The comprehensive strength decreases at a much faster rate for batches with a greater than 70% water/cement ratio.



# Piecewise Linear Regression

```
cement.lm <- lm(STRENGTH ~ RATIO + X2STAR, data = cement)
```

```
summary(cement.lm)
```

$$(x_1 - k)x_2$$


```
## Call:
## lm(formula = STRENGTH ~ RATIO + X2STAR, data = cement)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.72124	-0.09753	-0.00163	0.24297	0.49393

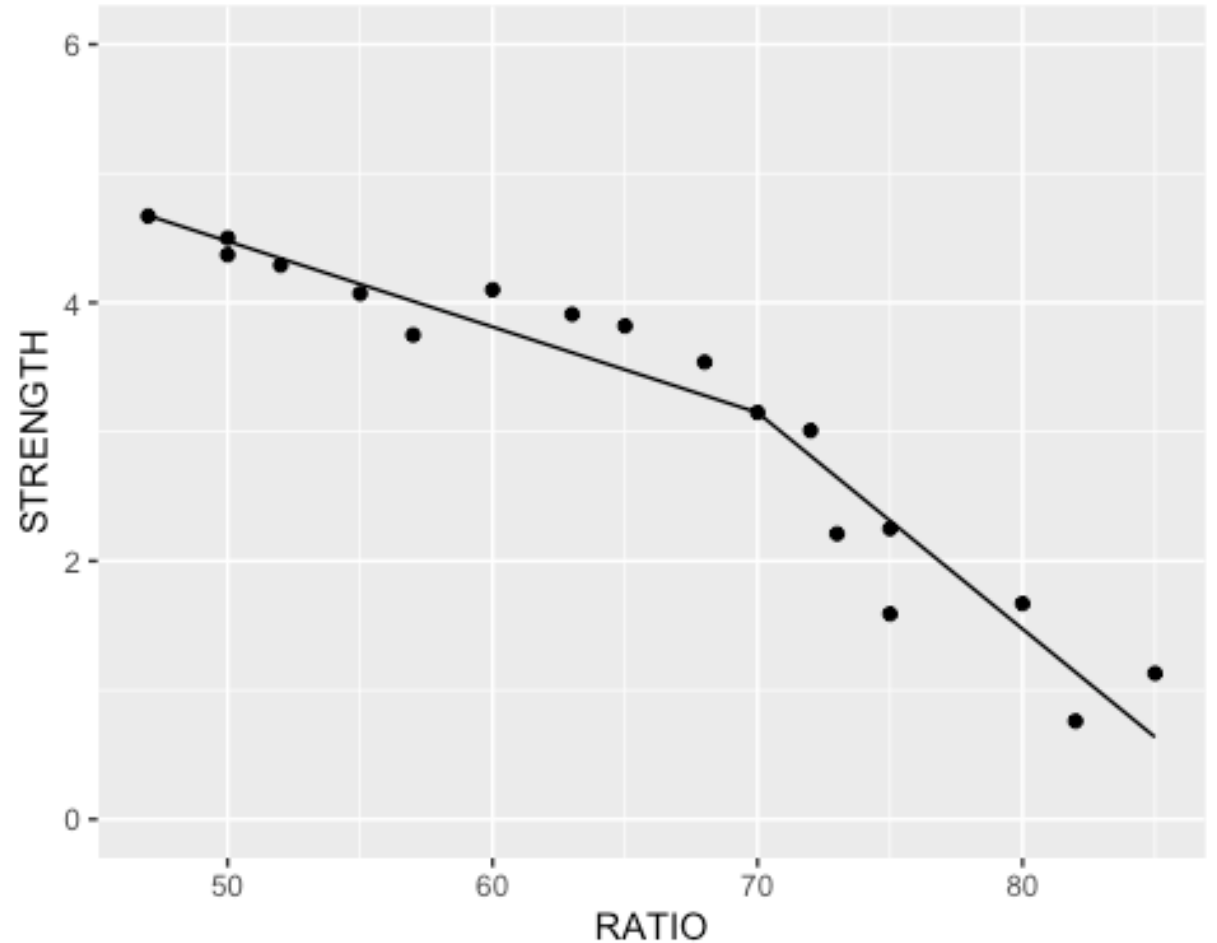
```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	7.79198	0.67696	11.510	7.62e-09	***
RATIO	-0.06633	0.01123	-5.904	2.89e-05	***
X2STAR	-0.10119	0.02812	-3.598	0.00264	**

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3286 on 15 degrees of freedom
## Multiple R-squared:  0.9385, Adjusted R-squared:  0.9303
## F-statistic: 114.4 on 2 and 15 DF,  p-value: 8.257e-10
```

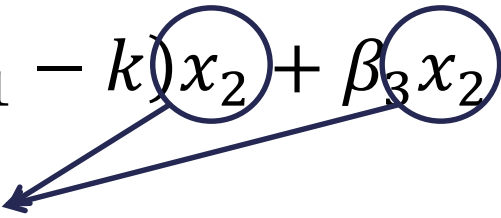
# Piecewise Linear Regression

```
ggplot(cement, aes(x = RATIO, y = STRENGTH)) +  
  geom_point() +  
  geom_line(data = cement, aes(x = RATIO,  
                                y = cement.lm$fitted.values)) +  
  ylim(0,6)
```



# Extensions – Discontinuous

- The previous approach dealt with piecewise functions that are continuous.
- The following is the discontinuous set-up for two straight lines:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k) x_2 + \beta_3 x_2 + \varepsilon$$


$$x_2 = \begin{cases} 1, & x_1 > k \\ 0, & x_1 \leq k \end{cases}$$

# Extensions – Discontinuous

```
cement.lm <- lm(STRENGTH ~ RATIO + X2STAR + X2, data = cement)
```

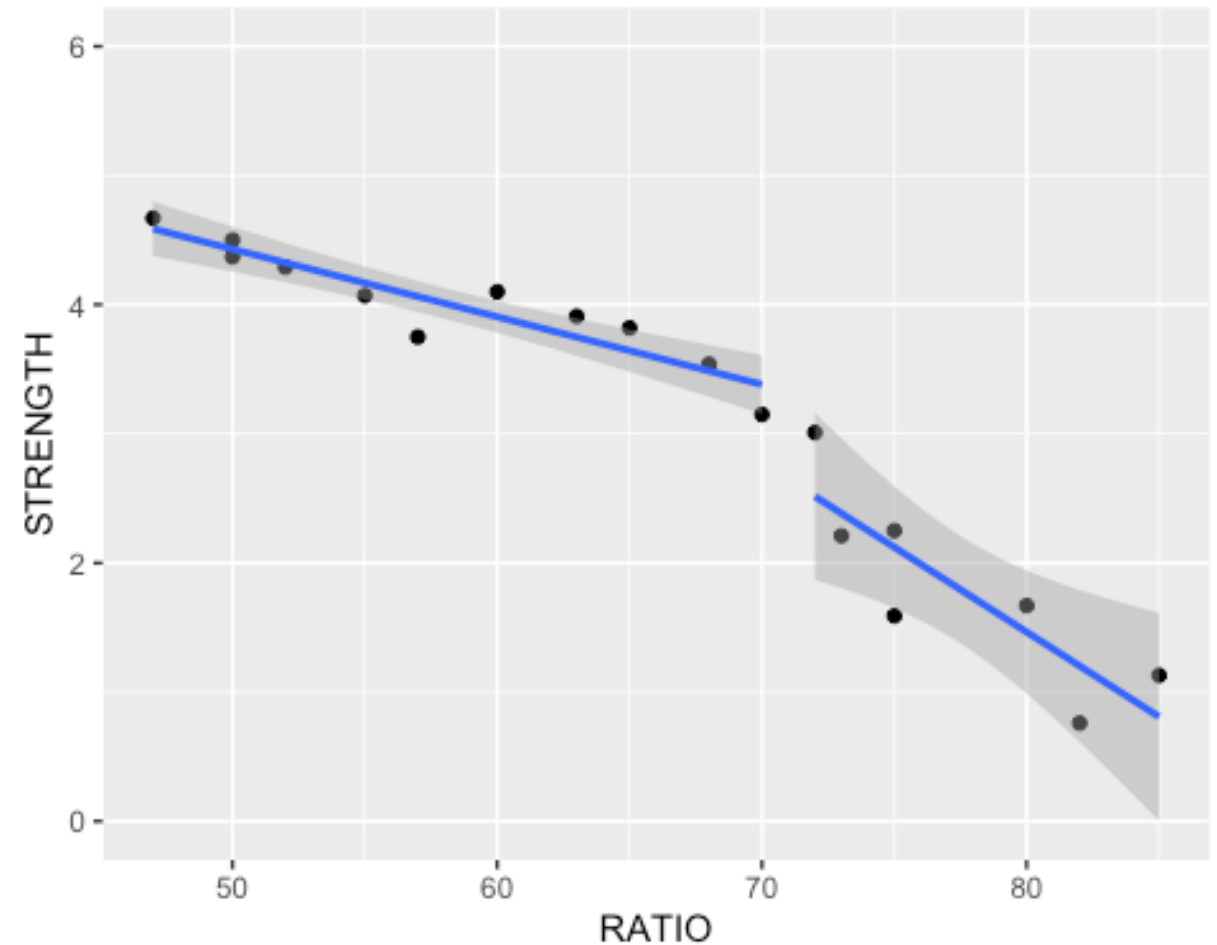
```
summary(cement.lm)
```

$$(x_1 - k)x_2$$


```
##
## Call:
## lm(formula = STRENGTH ~ RATIO + X2STAR + X2, data = cement)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53167 -0.15513  0.06171  0.17239  0.49451
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.04975     0.68558   10.283  6.6e-08 ***
## RATIO         -0.05240     0.01174   -4.463  0.000536 ***
## X2STAR        -0.07888     0.02686   -2.937  0.010830 *
## X2            -0.60388     0.26877   -2.247  0.041302 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2916 on 14 degrees of freedom
## Multiple R-squared:  0.9548, Adjusted R-squared:  0.9451
## F-statistic: 98.57 on 3 and 14 DF,  p-value: 1.188e-09
```

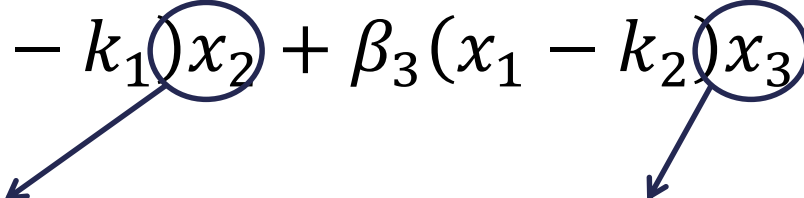
# Extensions – Discontinuous

```
qplot(RATIO, STRENGTH, group = X2,  
      geom = c('point', 'smooth'),  
      method = 'lm', data = cement,  
      ylim = c(0,6))
```



# Extensions

- The same modeling approach can be applied to any piecewise regression.
- The following is the set-up for three straight lines:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 (x_1 - k_1) x_2 + \beta_3 (x_1 - k_2) x_3 + \varepsilon$$
The diagram shows two arrows originating from the circled terms in the equation above. One arrow points from the circled  $x_2$  to the definition of  $x_2$  below. The other arrow points from the circled  $x_3$  to the definition of  $x_3$  below.

$$x_2 = \begin{cases} 1, & x_1 > k_1 \\ 0, & \text{if not} \end{cases}$$

$$x_3 = \begin{cases} 1, & x_1 > k_2 \\ 0, & \text{if not} \end{cases}$$





# MARS (AND EARTH)

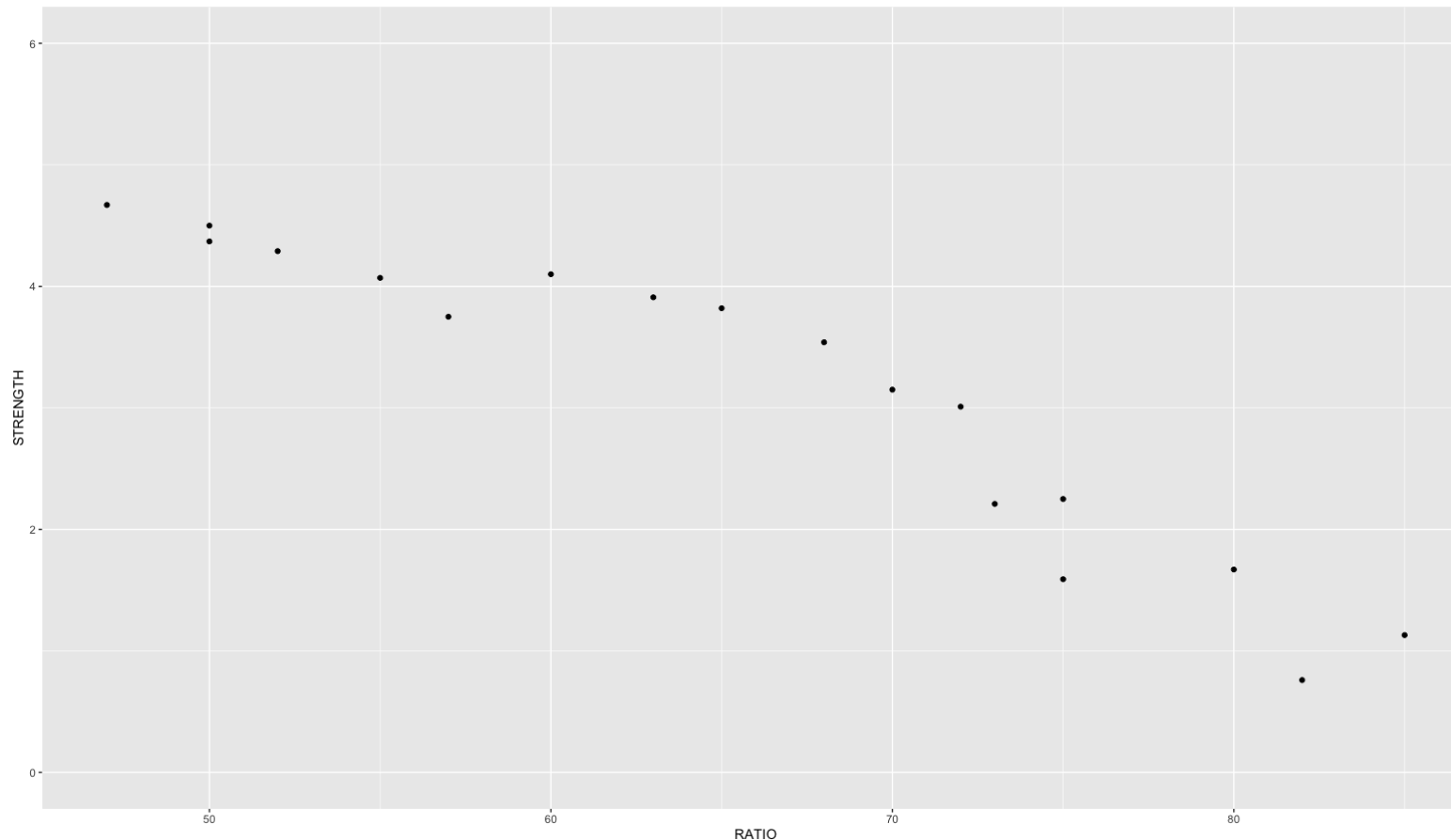
---

# Multivariate Adaptive Regression Splines (MARS)

- Multivariate adaptive regression splines (MARS) is a non-parametric technique that is still has a linear form to the model (additive) but has nonlinearities and interaction between variables.
- Essentially, uses **piecewise** regression approach to split into pieces then potentially uses either linear or nonlinear patterns for each piece.

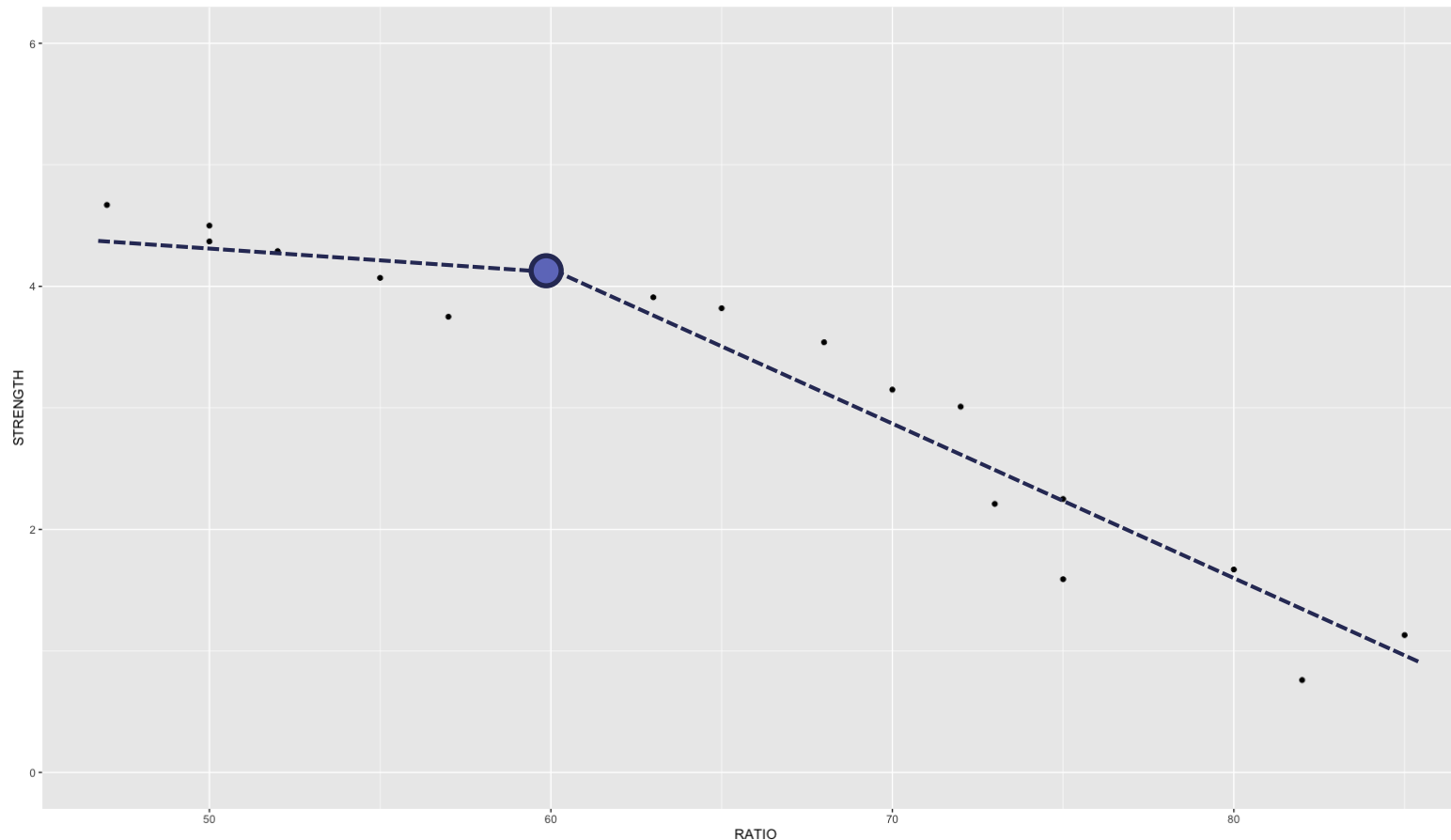
# Knots

- MARS first looks for the point in the range of  $x$  where two linear functions on either side of the point provides the least squared error (linear regression).



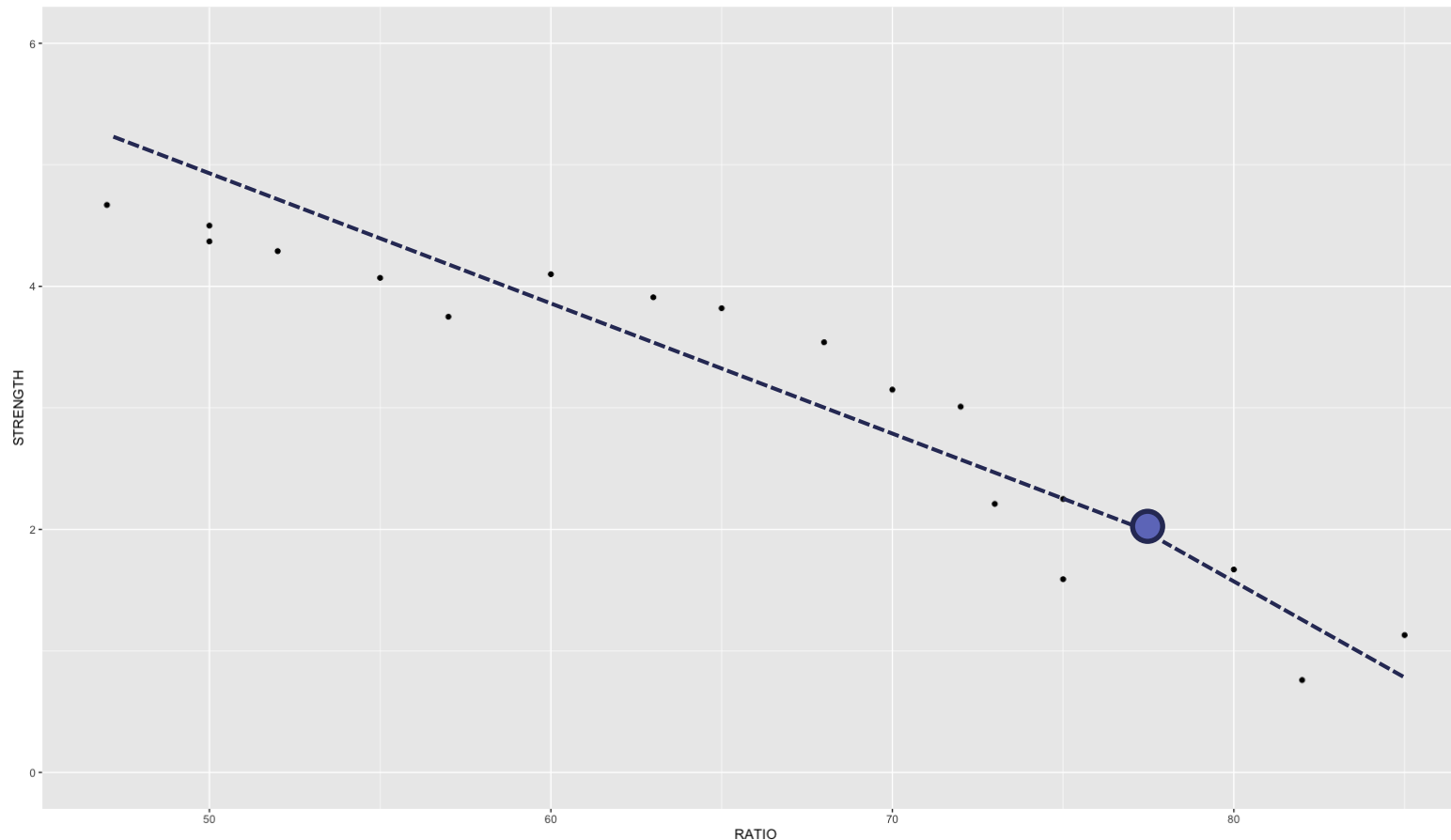
# Knots

- MARS first looks for the point in the range of  $x$  where two linear functions on either side of the point provides the least squared error (linear regression).



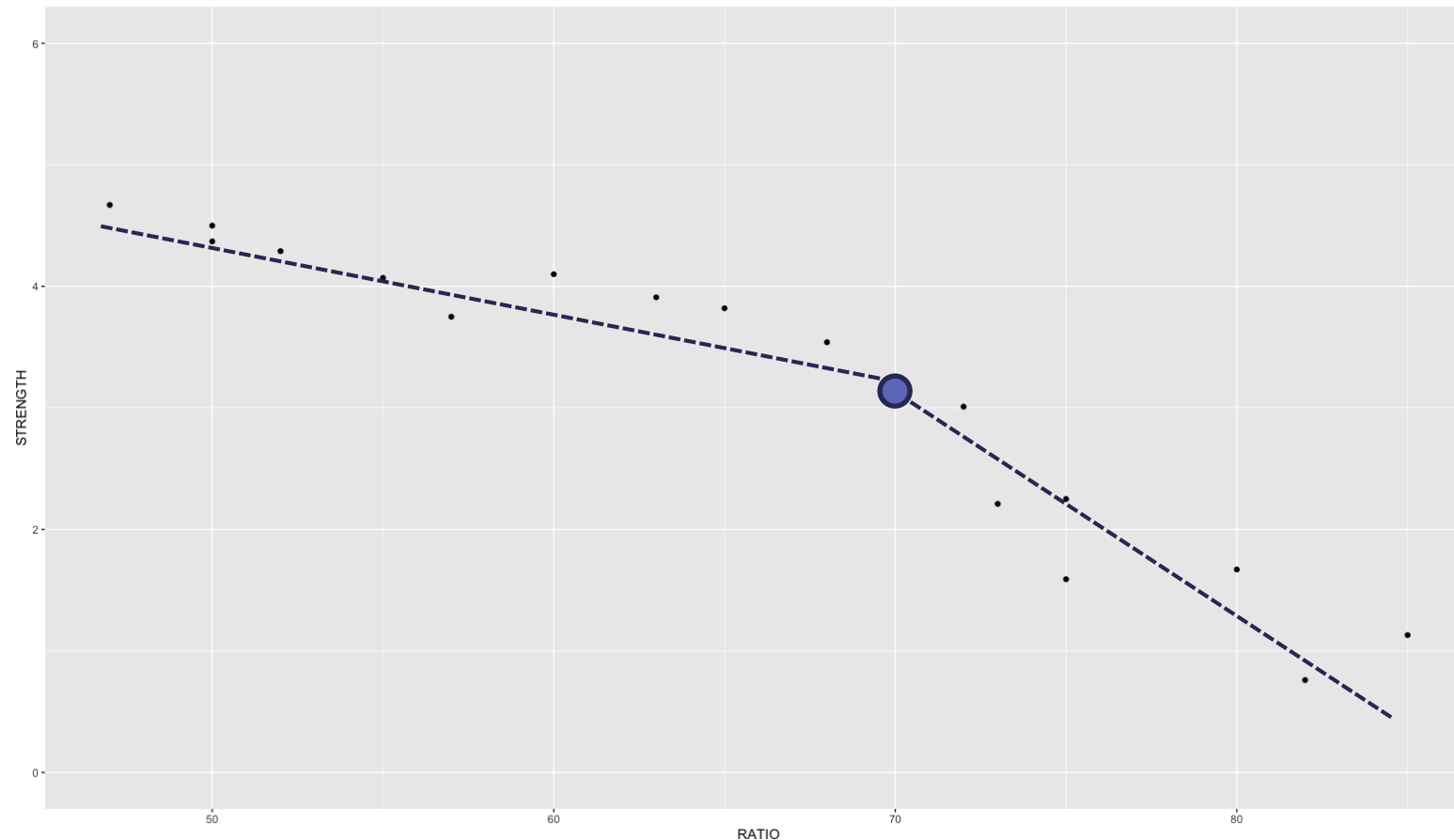
# Knots

- MARS first looks for the point in the range of  $x$  where two linear functions on either side of the point provides the least squared error (linear regression).

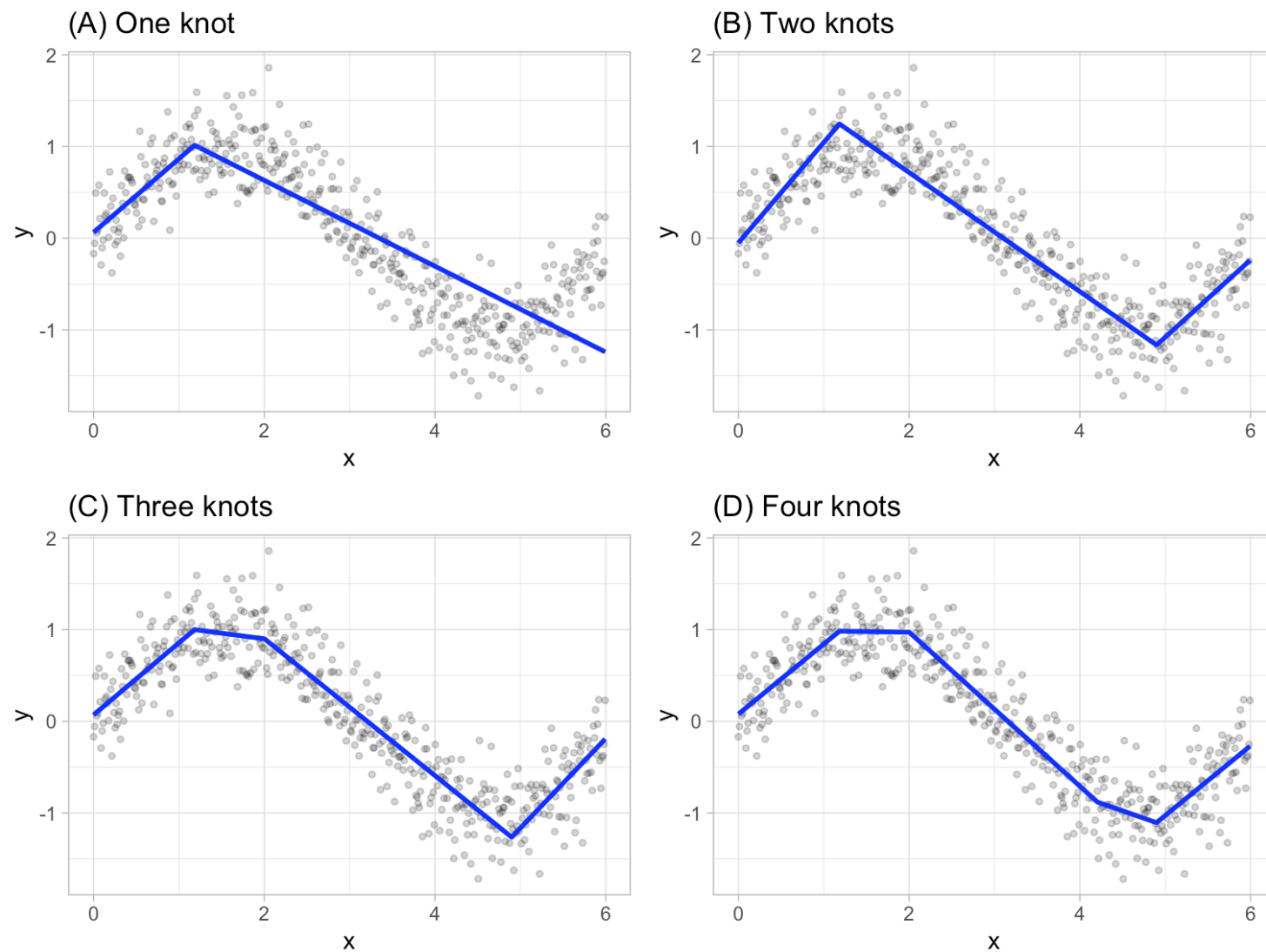


# Knots

- MARS first looks for the point in the range of  $x$  where two linear functions on either side of the point provides the least squared error (linear regression).



# Many Knots





# How Many Knots?

- Algorithm continues on each piece of the piecewise function until many knots are found (WILL OVERFIT YOUR DATA).
- Then works backwards (“prunes”) to remove knots that do not contribute significantly to out of sample accuracy.
  - Calculation is performed by the generalized cross-validation (GCV) procedure – computational shortcut for leave-one-out cross-validation.
- **Does this for all variables!**

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \varepsilon$$

# EARTH

- **E**nhanced **A**daptive **R**egression **T**hrough **H**inges (EARTH) is the implementation of the MARS algorithm in most software.
- MARS is trademarked by Salford Systems, so instead we use EARTH.

# EARTH (and MARS)

```
ames <- make_ordinal_ames()
```

```
ames <- ames %>% mutate(id = row_number())
```

```
set.seed(4321)
```

```
training <- ames %>% sample_frac(0.7)
```

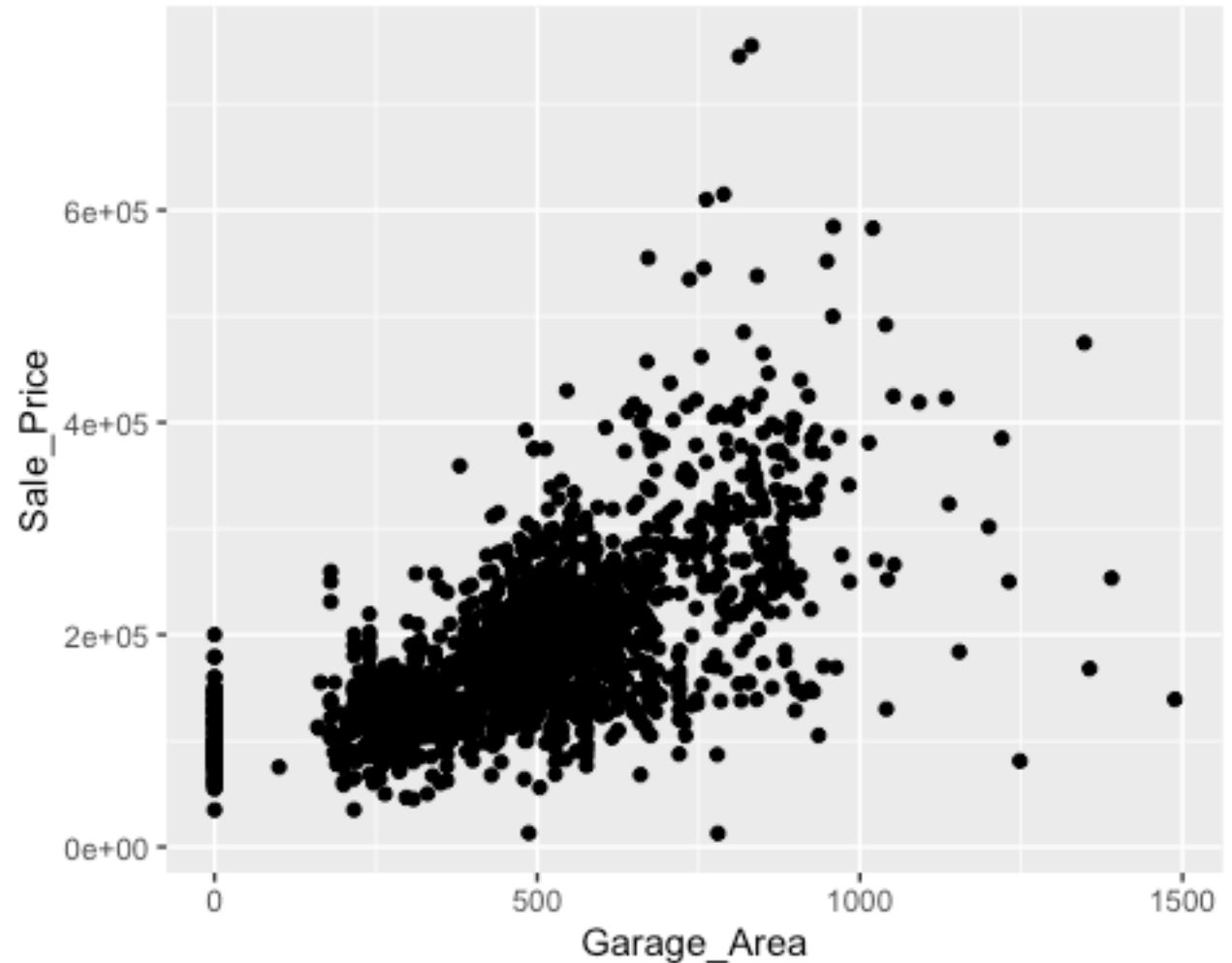
```
testing <- anti_join(ames, training, by = 'id')
```

```
training <- training %>%
```

```
  select(Sale_Price, Bedroom_AbvGr, Year_Built, Mo_Sold, Lot_Area, Street,  
         Central_Air, First_Flr_SF, Second_Flr_SF, Full_Bath, Half_Bath, Fireplaces,  
         Garage_Area, Gr_Liv_Area, TotRms_AbvGrd)
```

# EARTH (and MARS)

```
ggplot(training, aes(x = Garage_Area,  
                     y = Sale_Price)) +  
  geom_point()
```



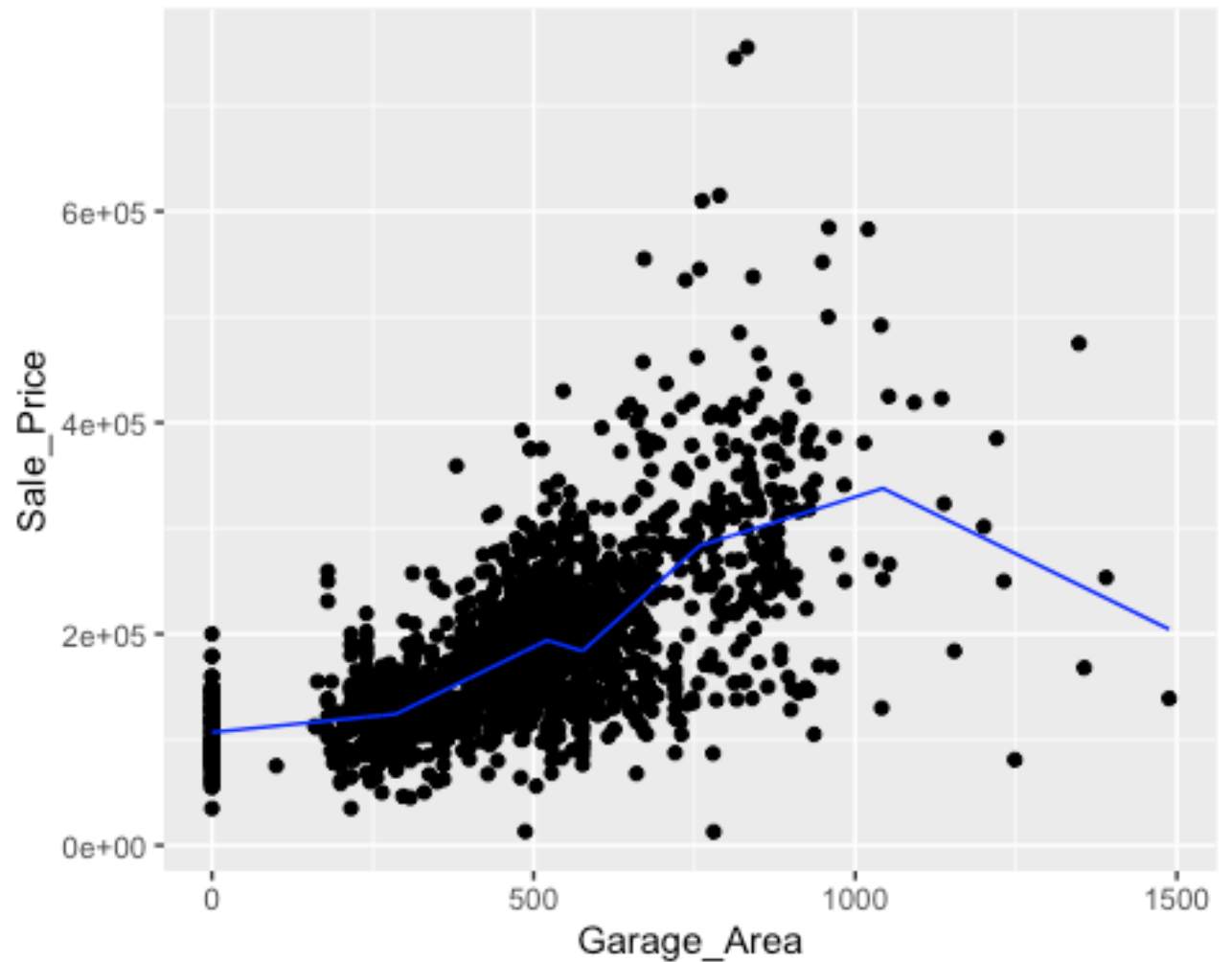
# EARTH (and MARS)

```
mars1 <- earth(Sale_Price ~ Garage_Area, data = training)
summary(mars1)
```

```
## Call: earth(formula=Sale_Price~Garage_Area, data=training)
##
##               coefficients
## (Intercept)      124159.039
## h(286-Garage_Area)    -60.257
## h(Garage_Area-286)    297.277
## h(Garage_Area-521)   -483.642
## h(Garage_Area-576)    733.859
## h(Garage_Area-758)   -356.460
## h(Garage_Area-1043)  -490.873
##
## Selected 7 of 7 terms, and 1 of 1 predictors
## Termination condition: RSq changed by less than 0.001 at 7 terms
## Importance: Garage_Area
## Number of terms at each degree of interaction: 1 6 (additive model)
## GCV 3427475346    RSS 6.94092e+12    GRSq 0.4492014    RSq 0.4556309
```

# EARTH (and MARS)

```
ggplot(training, aes(x = Garage_Area,  
                     y = Sale_Price)) +  
  geom_point() +  
  geom_line(data = training,  
            aes(x = Garage_Area,  
                y = mars1$fitted.values),  
            color = "blue")
```



# EARTH (and MARS)

```
mars1 <- earth(Sale_Price ~ ., data = training)
summary(mars1)

## Call: earth(formula=Sale_Price~., data=training)
##
##               coefficients
## (Intercept)      319493.46
## Central_AirY      20289.49
## h(4-Bedroom_AbvGr)  9214.66
## h(Bedroom_AbvGr-4) -23009.05
## ...
## h(Half_Bath-1)      -45378.31
## h(2-Fireplaces)     -14408.56
## h(Fireplaces-2)     -26072.58
## h(Garage_Area-539)   101.97
## h(Garage_Area-1043) -294.30
## h(Gr_Liv_Area-2049)  65.21
## h(Gr_Liv_Area-3194) -159.79
##
## Selected 21 of 24 terms, and 10 of 14 predictors
## Termination condition: Reached nk 29
## Importance: First_Flr_SF, Second_Flr_SF, Year_Built, Garage_Area, ...
## Number of terms at each degree of interaction: 1 20 (additive model)
## GCV 1033819964    RSS 2.036439e+12    GRSq 0.8338641    RSq 0.8402842
```

# Variable Importance

```

mars1 <- earth(Sale_Price ~ ., data = training)
summary(mars1)

## Call: earth(formula=Sale_Price~., data=training)
##
##
##              coefficients
## (Intercept)      319493.46
## Central_AirY      20289.49
## h(4-Bedroom_AbvGr)   9214.66
## h(Bedroom_AbvGr-4) -23009.05
## ...
## h(Half_Bath-1)      -45378.31
## h(2-Fireplaces)     -14408.56
## h(Fireplaces-2)     -26072.58
## h(Garage_Area-539)    101.97
## h(Garage_Area-1043)  -294.30
## h(Gr_Liv_Area-2049)   65.21
## h(Gr_Liv_Area-3194) -159.79
##
## Selected 21 of 24 terms, and 10 of 14 predictors
## Termination condition: Reached nk 29
## Importance: First Flr SF, Second Flr SF, Year Built, Garage Area, ...
## Number of terms at each degree of interaction: 1 20 (additive model)
## GCV 1033819964    RSS 2.036439e+12    GRSq 0.8338641    RSq 0.8402842

```



# Variable Importance

- There is one “subset” for each model size (1 term, 2 terms, etc.) – the best model of that size.
- Ranks variables by how many of the subsets that variable appears in.
  - More subsets of models it appears in (in the best 1 variable model, in the best 2 variable model, etc.), then the better the variable.
- RSS (residual sum of squares or sum of squares error) is scaled version of decrease in residual sum of squares relative to the previous subset.
- GCV is approximation of RSS on leave-one-out-cross validation.

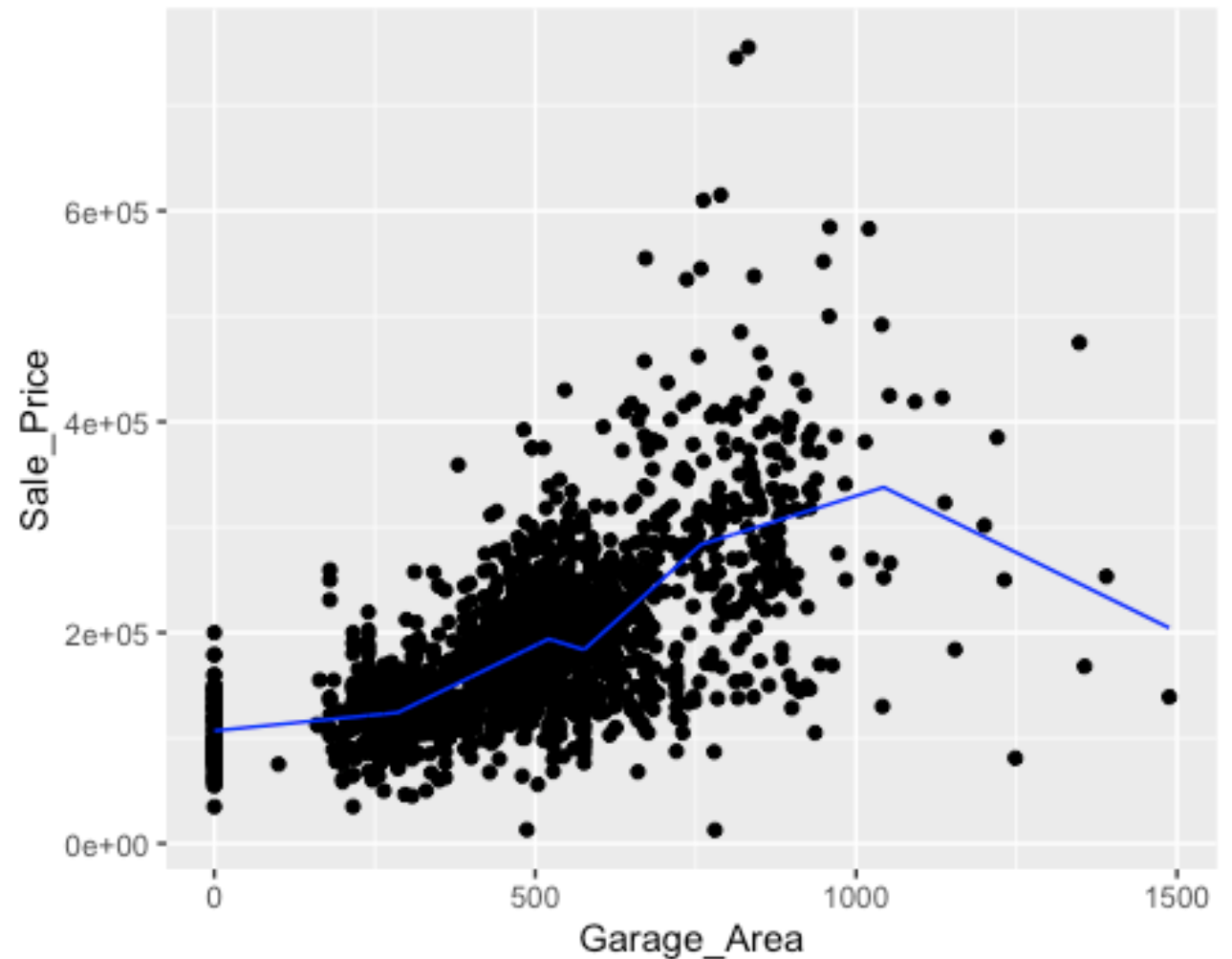
# Variable Importance

```
evimp(mars1)
```

##	nsubsets	gcv	rss
## First_Flr_SF	20	100.0	100.0
## Second_Flr_SF	19	71.7	71.9
## Year_Built	18	50.9	51.3
## Garage_Area	17	34.3	35.0
## Fireplaces	16	31.0	31.7
## Gr_Liv_Area	15	27.6	28.4
## Central_AirY	12	20.0	20.9
## Bedroom_AbvGr	11	18.1	19.0
## Lot_Area	10	16.2	17.2
## Half_Bath	4	7.4	8.2

# Interpretability

- Can view the “relationship” between predictors and target variable.





# SMOOTHING

---

# Smoothing

- GAMs can be made up of any non-parametric function of the predictor variables.
- Another popular technique is to use **smoothing functions** so the piecewise linear regressions are not so jagged.
- Many different types of smoothing functions:
  - LOESS (localized regression)
  - Smoothing splines
  - Regression splines

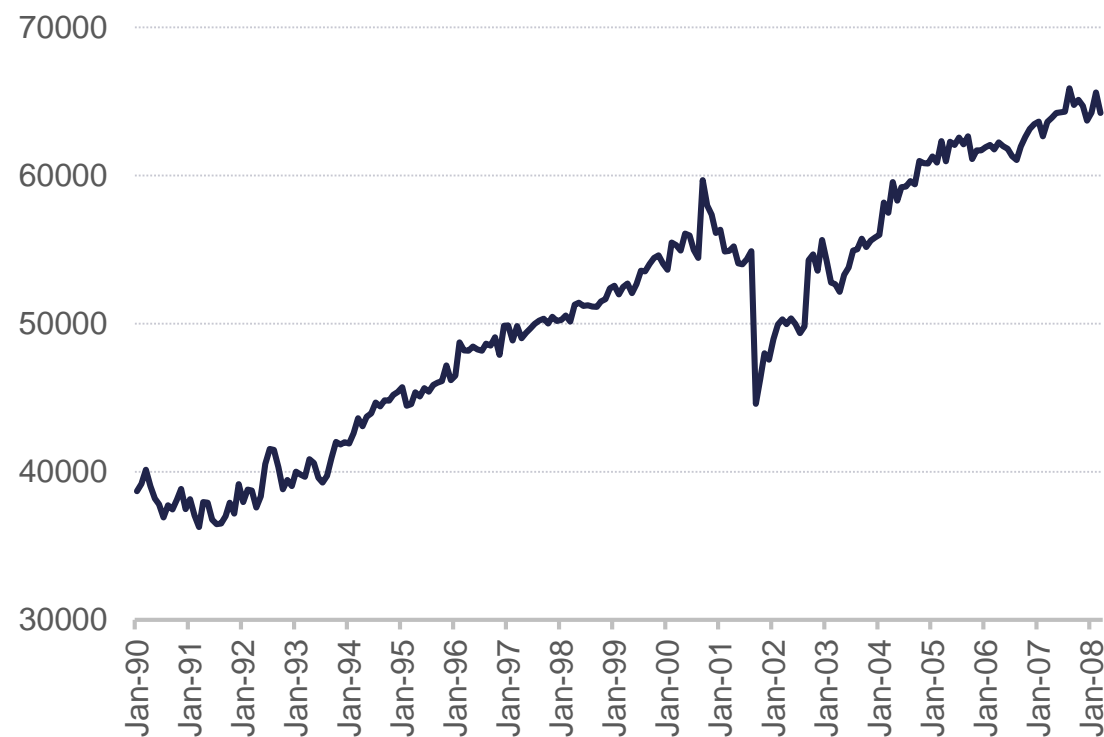
# Smoothing

- GAMs can be made up of any non-parametric function of the predictor variables.
- Another popular technique is to use **smoothing functions** so the piecewise linear regressions are not so jagged.
- Many different types of smoothing functions:
  - LOESS (localized regression)
  - Smoothing splines
  - Regression splines

# LOESS

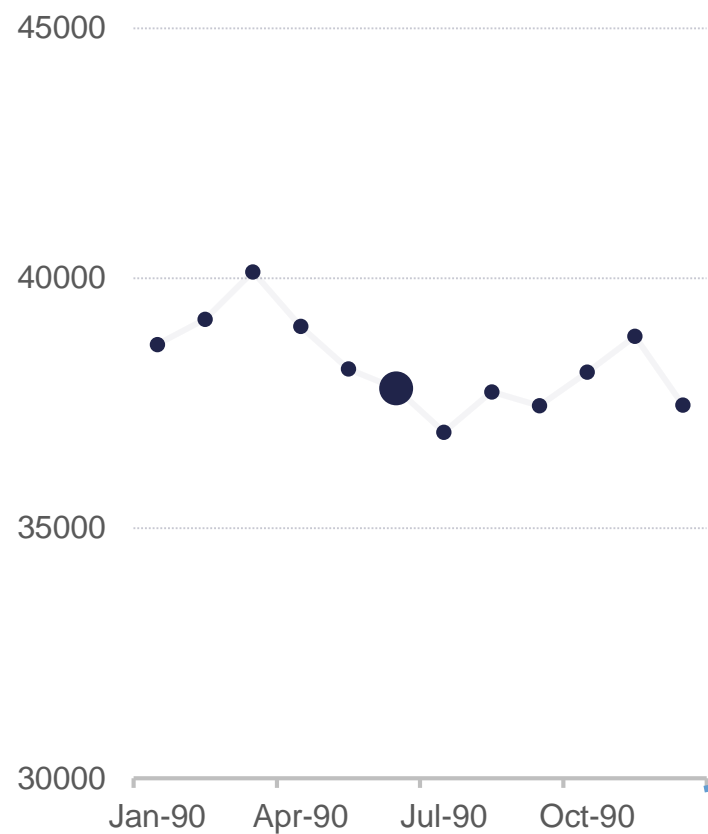
- Locally estimated scatterplot smoothing (LOESS) is a popular smoothing technique.
- Same technique used in STL decomposition in time series

U.S. AIRLINE PASSENGERS  
(SEASONALLY ADJUSTED)

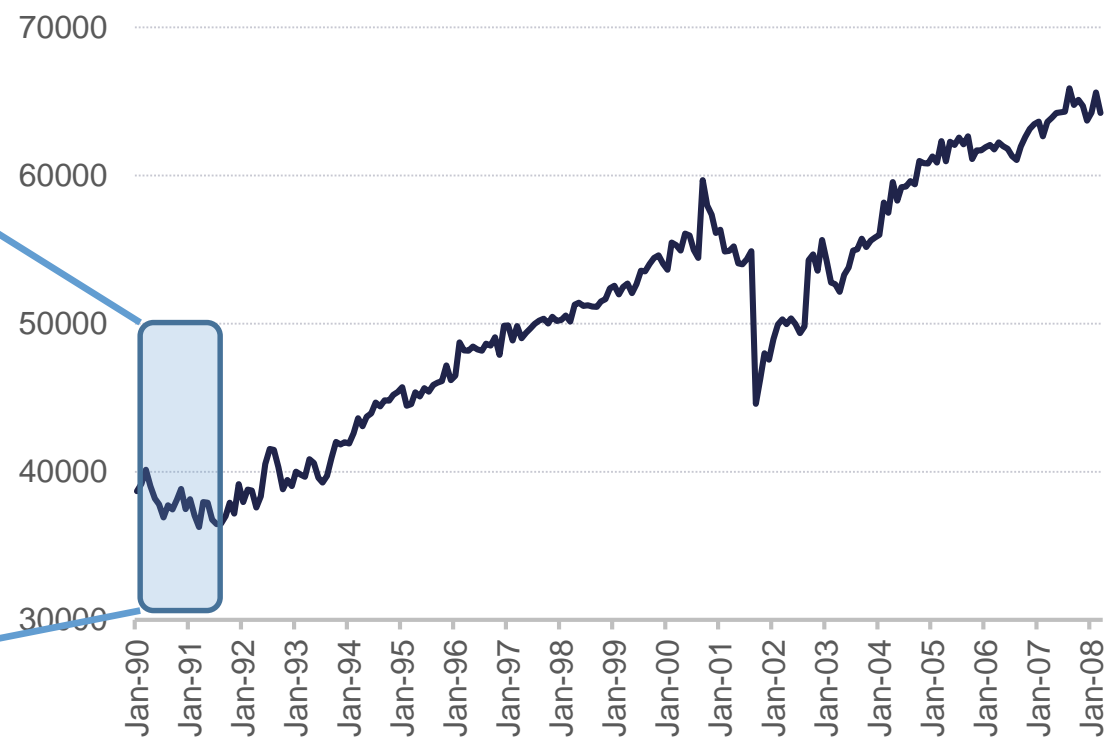




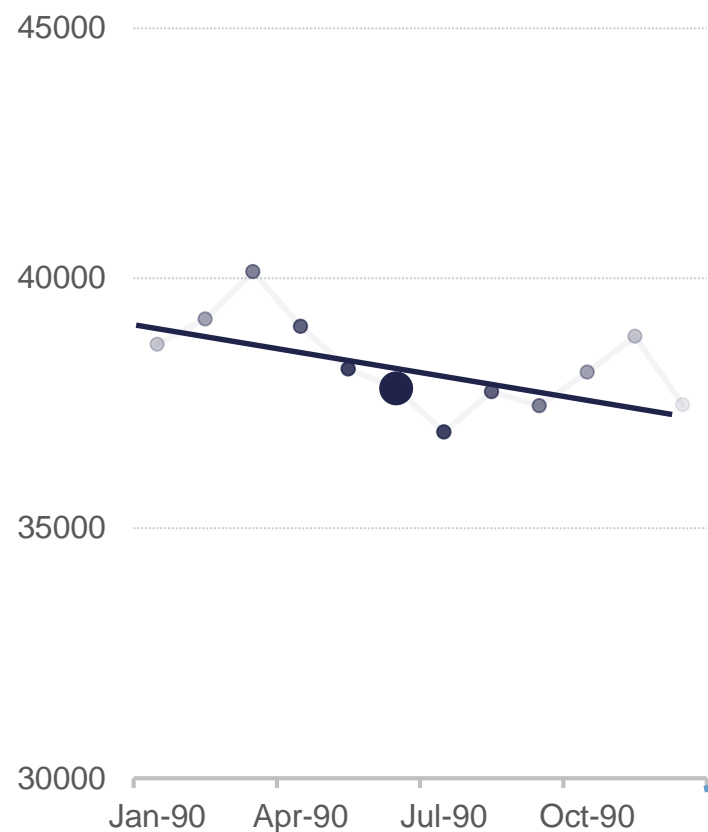
# LOESS



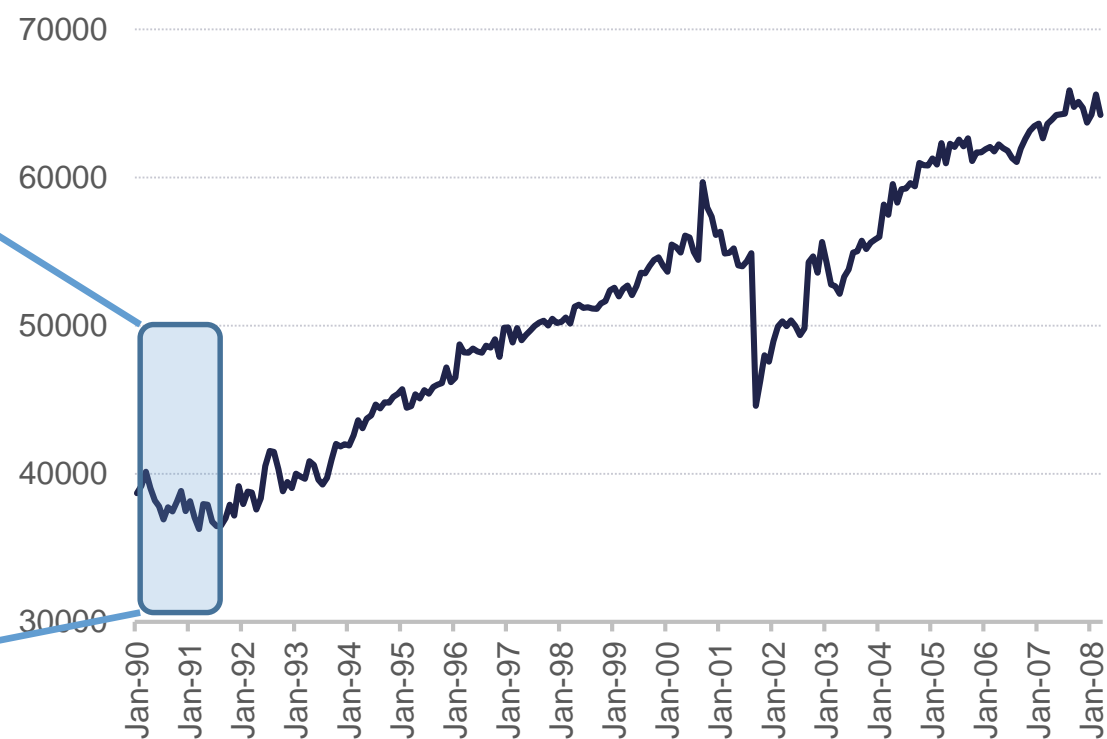
## U.S. AIRLINE PASSENGERS (SEASONALLY ADJUSTED)



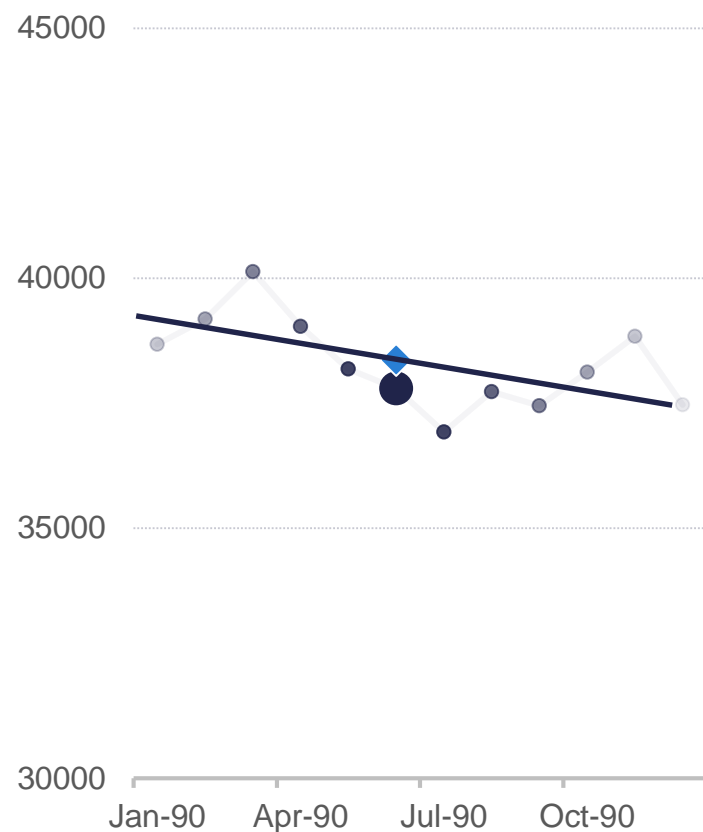
# LOESS



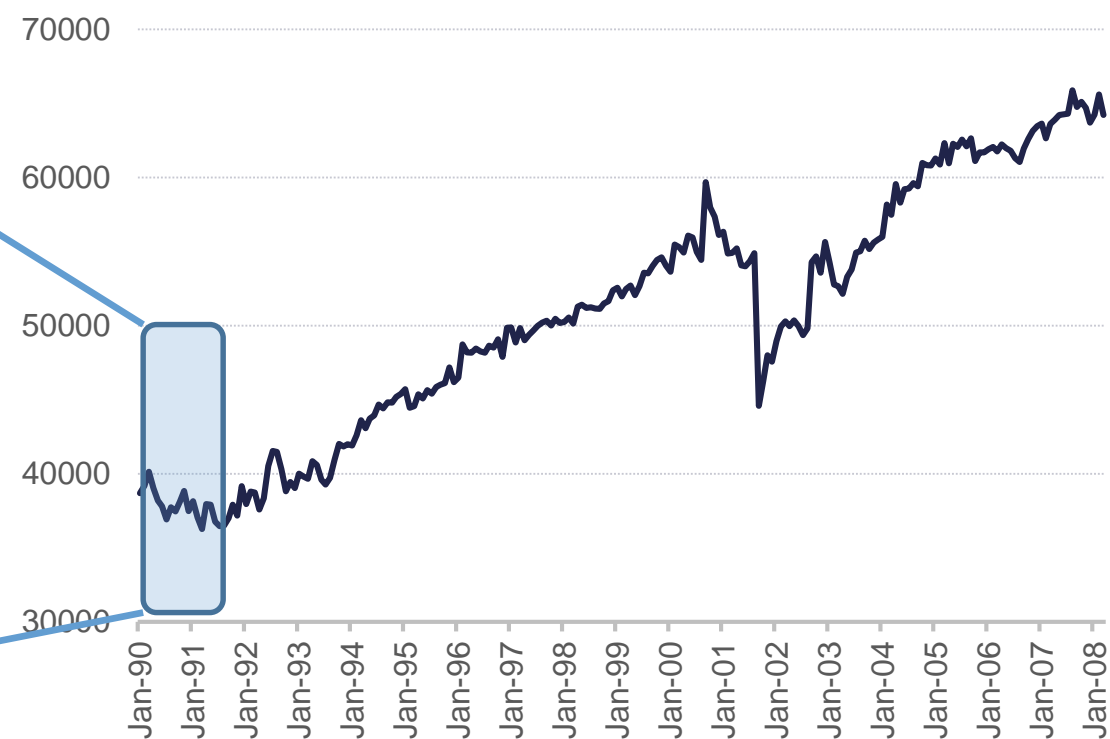
## U.S. AIRLINE PASSENGERS (SEASONALLY ADJUSTED)



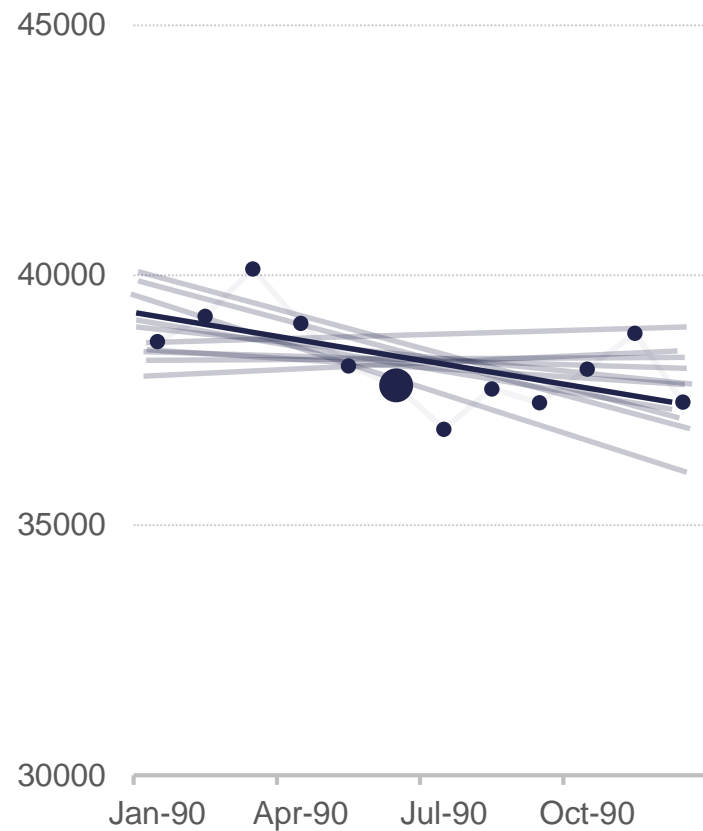
# LOESS



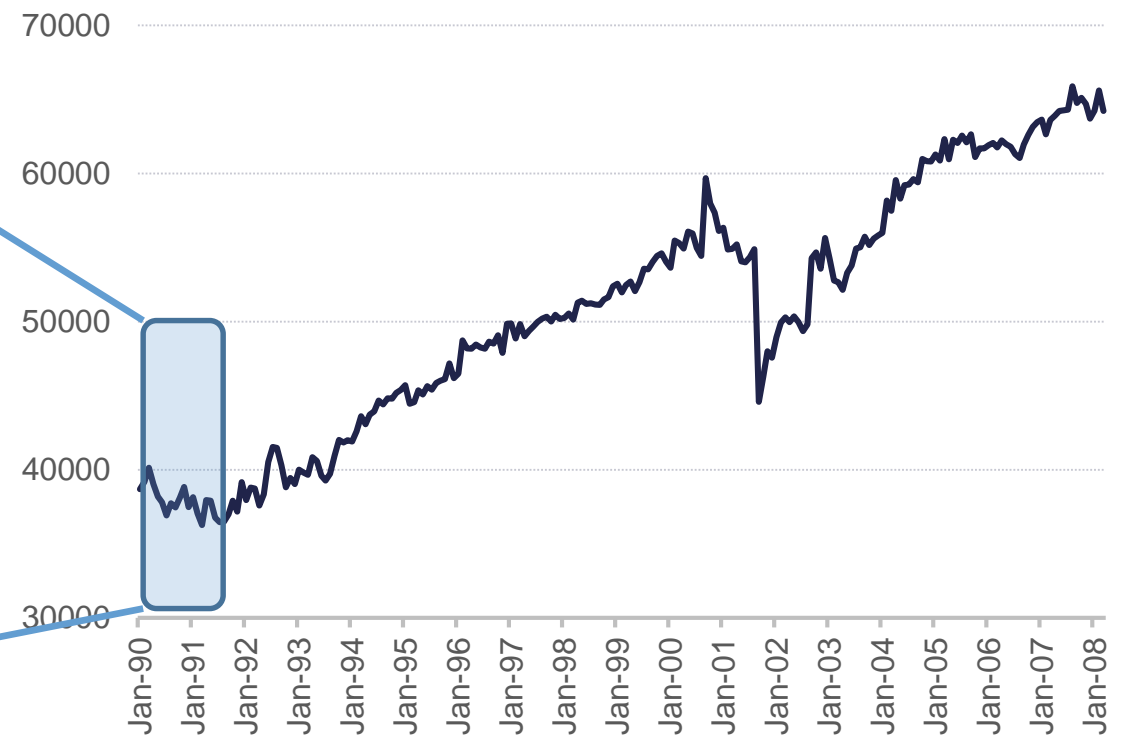
## U.S. AIRLINE PASSENGERS (SEASONALLY ADJUSTED)



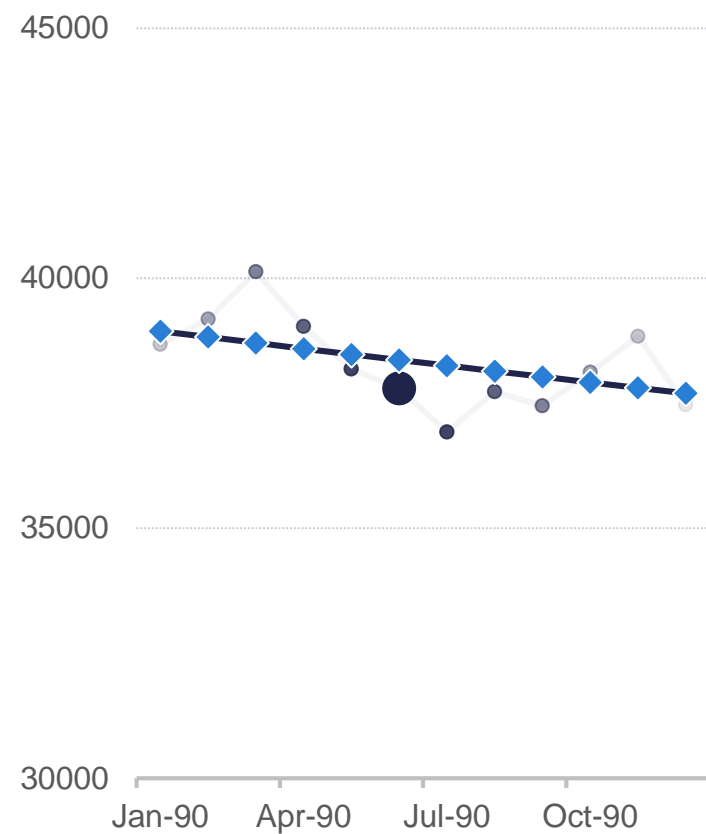
# LOESS



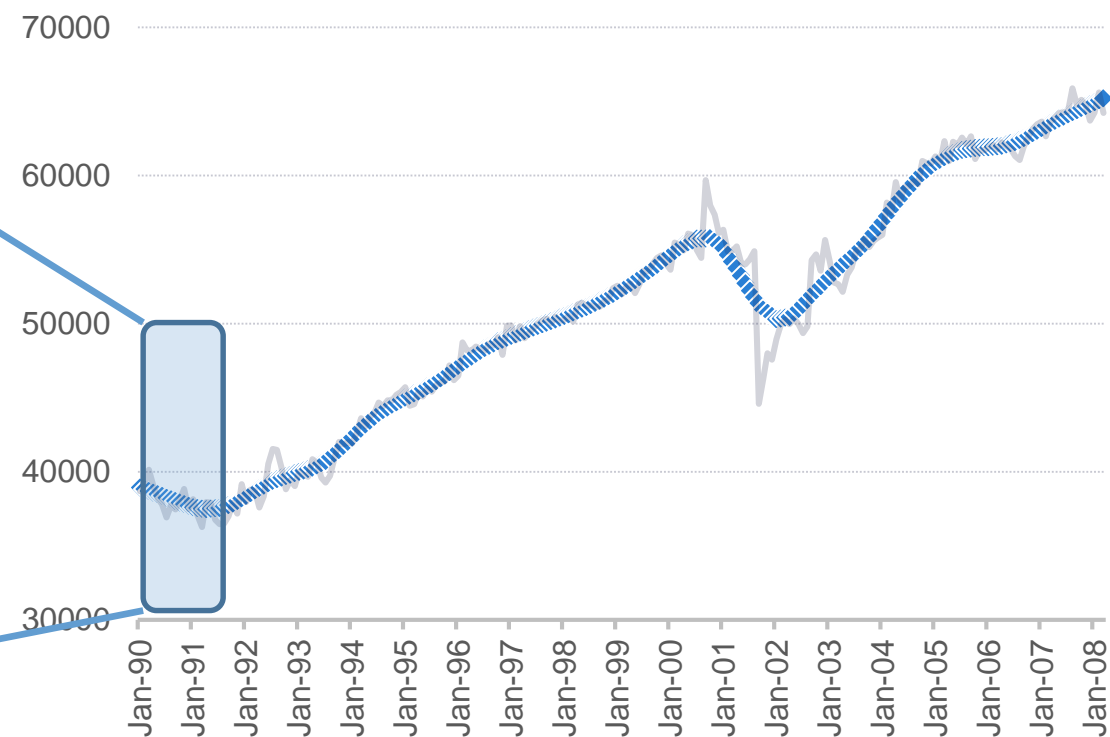
## U.S. AIRLINE PASSENGERS (SEASONALLY ADJUSTED)



# LOESS



## U.S. AIRLINE PASSENGERS (SEASONALLY ADJUSTED)

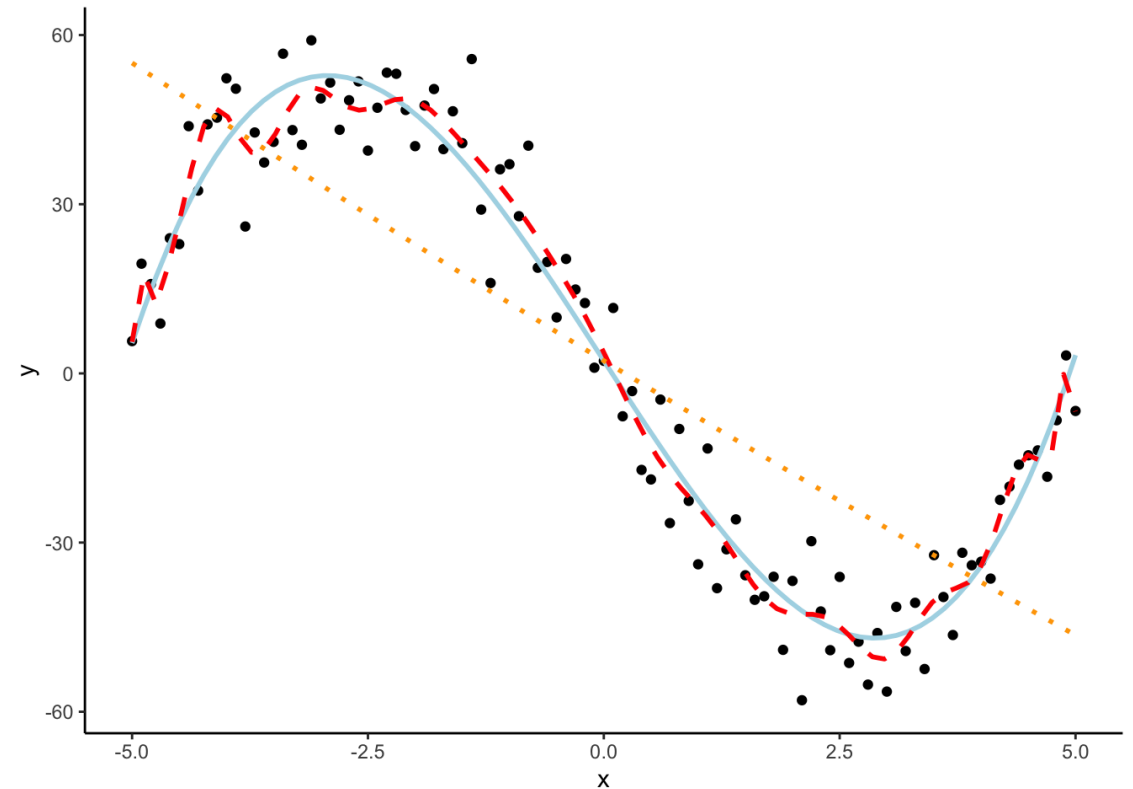


# Smoothing

- GAMs can be made up of any non-parametric function of the predictor variables.
- Another popular technique is to use **smoothing functions** so the piecewise linear regressions are not so jagged.
- Many different types of smoothing functions:
  - LOESS (localized regression)
  - Smoothing splines
  - Regression splines

# Smoothing Splines

- Smoothing splines take a different approach as compared to LOESS.
- Smoothing splines have a knot **at every single observation** for piecewise regression – OVERFITTING!
- Use penalty parameter to counterbalance the “wiggle” of the spline.



# Smoothing Splines

- Smoothing splines try to find the function  $s(x_i)$  that optimally fits  $x$  to the target variable  $y$  through this equation:

$$\min \sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(t)^2 dt$$



# Smoothing Splines

- Smoothing splines try to find the function  $s(x_i)$  that optimally fits  $x$  to the target variable  $y$  through this equation:

$$\min \sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(t_i)^2 dt$$

Sum of squared error!

# Smoothing Splines

- Smoothing splines try to find the function  $s(x_i)$  that optimally fits  $x$  to the target variable  $y$  through this equation:

$$\min \sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(t_i)^2 dt$$

Penalty ( $\lambda$ ) applied to  
integral of second  
derivative of smoothing  
function

# Smoothing Splines

- Smoothing splines try to find the function  $s(x_i)$  that optimally fits  $x$  to the target variable  $y$  through this equation:

$$\min \sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(t_i)^2 dt$$

Penalty ( $\lambda$ ) applied to integral of “slope of slopes” which is large when lots of “wiggle”

# Smoothing Splines

- Smoothing splines try to find the function  $s(x_i)$  that optimally fits  $x$  to the target variable  $y$  through this equation:

$$\min \sum_{i=1}^n (y_i - s(x_i))^2 + \lambda \int s''(t_i)^2 dt$$

Penalty ( $\lambda$ ) estimated with another approximation of leave one out cross validation

# Smoothing

- GAMs can be made up of any non-parametric function of the predictor variables.
- Another popular technique is to use **smoothing functions** so the piecewise linear regressions are not so jagged.
- Many different types of smoothing functions:
  - LOESS (localized regression)
  - Smoothing splines
  - Regression splines – computationally nicer version of smoothing splines

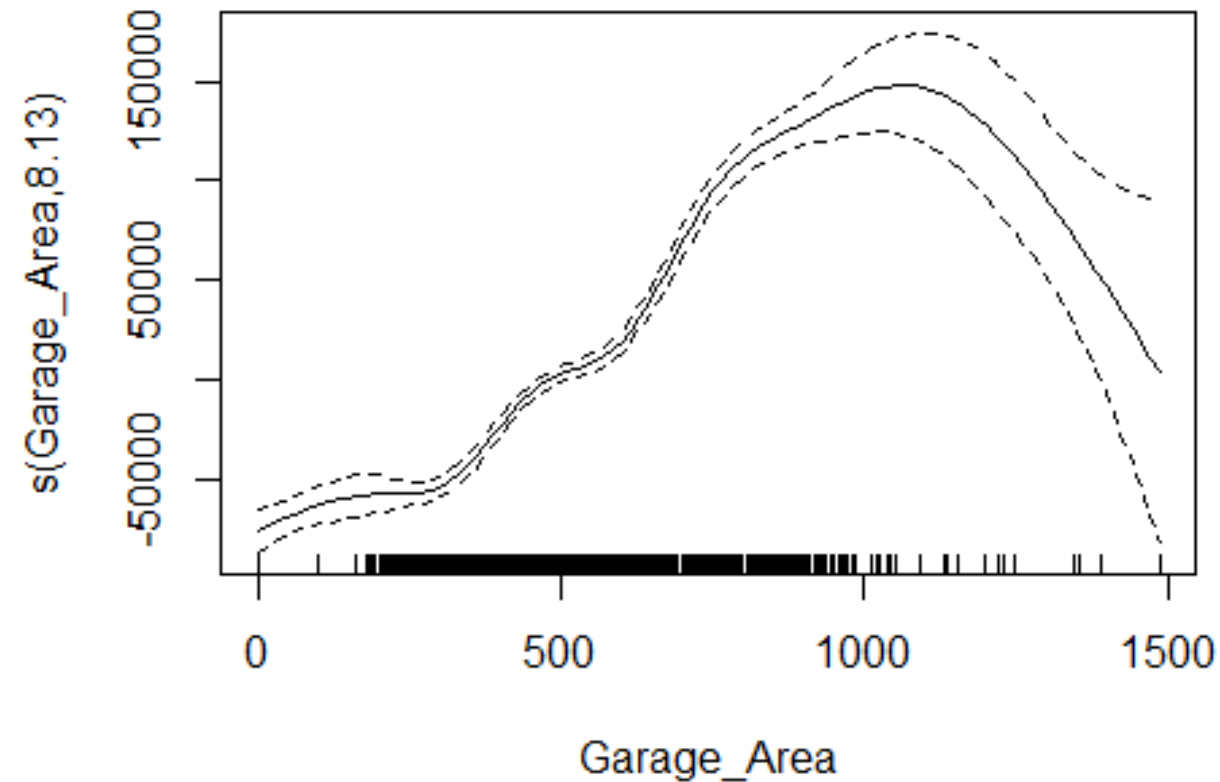
# GAM's with Splines

```
gam1 <- mgcv::gam(Sale_Price ~ s(Garage_Area), data = training)
summary(gam1)
```

```
## Family: gaussian
## Link function: identity
##
## Formula:
## Sale_Price ~ s(Garage_Area)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   180897      1290    140.2  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df      F p-value
## s(Garage_Area) 8.134  8.769 192.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.451   Deviance explained = 45.3%
## GCV = 3.4301e+09   Scale est. = 3.4148e+09   n = 2051
```

# GAM's with Splines

```
plot(gam1)
```



# GAM's with Splines

```
gam2 <- mgcv::gam(Sale_Price ~ s(Bedroom_AbvGr, k = 5) +  
                    s(Year_Built) +  
                    s(Mo_Sold) +  
                    s(Lot_Area) +  
                    s(First_Flr_SF) +  
                    s(Second_Flr_SF) +  
                    s(Garage_Area) +  
                    s(Gr_Liv_Area) +  
                    s(TotRms_AbvGrd) +  
                    Street +  
                    Central_Air +  
                    factor(Fireplaces) +  
                    factor(Full_Bath) +  
                    factor(Half_Bath)  
                    , method = 'REML', data = training)  
summary(gam2)
```



# GAM's with Splines

```
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    136139     19681   6.917 6.19e-12 ***
## StreetPave      27689     12710   2.178  0.0295  *
## Central_AirY    18012      3168   5.685 1.50e-08 ***
## factor(Fireplaces)1  14070      1666   8.443 < 2e-16 ***
## factor(Fireplaces)2  27137      3146   8.626 < 2e-16 ***
## factor(Fireplaces)3  15704     10552   1.488  0.1368
## factor(Fireplaces)4 -79595     31469  -2.529  0.0115  *
## factor(Full_Bath)1   -5341     14528  -0.368  0.7132
## factor(Full_Bath)2  -11074     14827  -0.747  0.4552
## factor(Full_Bath)3    1226     15787   0.078  0.9381
## factor(Full_Bath)4  -16271     24326  -0.669  0.5037
## factor(Half_Bath)1    2102      2206   0.953  0.3408
## factor(Half_Bath)2  -38507      9111  -4.226 2.48e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
...

```

# GAM's with Splines

```
...
## Approximate significance of smooth terms:
##              edf Ref.df      F p-value
## s(Bedroom_AbvGr) 2.653  3.165  18.789 <2e-16 ***
## s(Year_Built)    6.445  7.543 101.758 <2e-16 ***
## s(Mo_Sold)       1.516  1.868   0.993  0.4507
## s(Lot_Area)      7.186  8.193  11.726 <2e-16 ***
## s(First_Flr_SF)  8.063  8.765  15.548 <2e-16 ***
## s(Second_Flr_SF) 8.212  8.818   7.806 <2e-16 ***
## s(Garage_Area)   7.426  8.328  21.654 <2e-16 ***
## s(Gr_Liv_Area)   8.545  8.882  14.834 <2e-16 ***
## s(TotRms_AbvGrd) 3.805  4.738   1.921  0.0783 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) = 0.851   Deviance explained = 85.6%
## -REML = 23957   Scale est. = 9.2392e+08   n = 2051
```

# GAM's with Selected Splines

```
sel.gam2 <- mgcv::gam(Sale_Price ~ s(Bedroom_AbvGr, k = 5) +
  s(Year_Built) +
  s(Mo_Sold) +
  s(Lot_Area) +
  s(First_Flr_SF) +
  s(Second_Flr_SF) +
  s(Garage_Area) +
  s(Gr_Liv_Area) +
  s(TotRms_AbvGrd) +
  Street +
  Central_Air +
  factor(Fireplaces) +
  factor(Full_Bath) +
  factor(Half_Bath)
, method = 'REML',
  select = TRUE, data = training)

summary(sel.gam2)
```

```
## Approximate significance of smooth terms:
##               edf Ref.df      F  p-value
## s(Bedroom_AbvGr) 2.333381     4 14.833 < 2e-16 ***
## s(Year_Built)    6.945994     9 83.962 < 2e-16 ***
## s(Mo_Sold)       0.007522     9  0.001 0.33142
## s(Lot_Area)      7.561273     9 11.790 < 2e-16 ***
## s(First_Flr_SF)  8.602695     9 32.390 < 2e-16 ***
## s(Second_Flr_SF) 0.940878     9  1.716 3.89e-06 ***
## s(Garage_Area)   6.688676     9 19.626 < 2e-16 ***
## s(Gr_Liv_Area)   4.839941     9  7.634 < 2e-16 ***
## s(TotRms_AbvGrd) 3.740058     9  1.280 0.00845 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.845   Deviance explained = 84.9%
## -REML = 24081   Scale est. = 9.6657e+08   n = 2051
```



# SUMMARY

---

# Generalized Additive Models (GAMs)

- Provides **general** framework for **adding** non-linear functions to standard **linear model**.

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \varepsilon$$

- Can be used for regression or classification problems.

# Generalized Additive Models (GAMs)

## Advantages

- Allows a nonlinear relationship without trying out many transformations manually
- Improved predictions
- Still has some “interpretation”
- Computationally fast

## Disadvantages

- Can incorporate interactions but can take time
- Not good for large numbers of variables – **prescreening needed!**
- Multicollinearity still a problem

