Notes 7: Polynomials and Interactions

Nathaniel E. Helwig

Department of Statistics
University of Illinois at Urbana-Champaign

Stat 420: Methods of Applied Statistics
Section N1U/N1G – Spring 2014

Outline of Notes

1) Polynomial Regression:

- Polynomials review
- Model form
- Model assumptions
- OLS estimation
- Orthogonal polynomials
- Example: MPG vs HP

2) Interactions in Regression:

- Overview
- Nominal*Continuous
- Example #1: Real Estate
- Example #2: Depression
- Continuous*Continuous
- Example #3: Oceanography

Polynomial Function: Definition

Reminder: a polynomial function has the form

$$\begin{aligned} f(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n \\ &= \sum_{j=0}^{n} a_j x^j \end{aligned}$$

where $a_j \in \mathbb{R}$ are the coefficients and x is the indeterminate (variable).

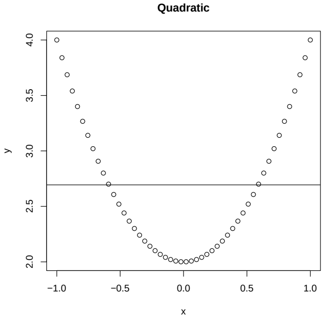
Note: x^j is the j-th order polynomial term

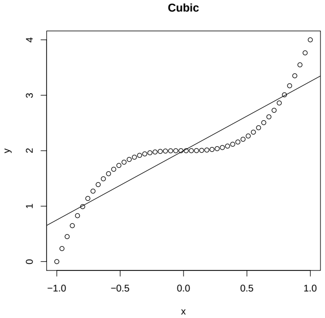
- x is first order term, x^2 is second order term, etc.
- The *degree* of a polynomial is the highest order term

Polynomial Function: Simple Regression

```
> x=seq(-1,1,l=50)
> y=2+2*(x^2)
> plot(x,y,main="Quadratic")
> qmod=lm(y~x)
> abline(qmod)
```

```
> x=seq(-1,1,l=50)
> y=2+2*(x^3)
> plot(x,y,main="Cubic")
> cmod=lm(y~x)
> abline(cmod)
```





Model Form (scalar)

The polynomial regression model has the form

$$y_i = b_0 + \sum_{j=1}^{p} b_j x_i^j + e_i$$

for $i \in \{1, \ldots, n\}$ where

- $y_i \in \mathbb{R}$ is the real-valued response for the i-th observation
- $b_0 \in \mathbb{R}$ is the regression intercept
- $b_j \in \mathbb{R}$ is the regression slope for the j-th degree polynomial
- $x_i \in \mathbb{R}$ is the predictor for the i-th observation
- $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is Gaussian measurement error

Model Form (matrix)

The polynomial regression model has the form

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e}$$

or

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^p \\ 1 & x_2 & x_2^2 & \cdots & x_2^p \\ 1 & x_3 & x_3^2 & \cdots & x_3^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^p \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{pmatrix}$$

Note that this is still a linear model, even though we have polynomial terms in the design matrix.

PR Model Assumptions (scalar)

The fundamental assumptions of the PR model are:

1. Relationship between x and y is polynomial
2. x_i and y_i are observed random variables (constants)
3. $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is an unobserved random variable
4. $b_0, b_1, \ldots, b_p$ are unknown constants
5. $(y_i | x_i) \overset{\text{ind}}{\sim} \mathrm{N}(b_0 + \sum_{j=1}^{p} b_j x_i^j, \sigma^2)$
   note: homogeneity of variance

Note: focus is estimation of the polynomial curve.

PR Model: Assumptions (matrix)

In matrix terms, the error vector is multivariate normal:

$$\mathbf{e} \sim \mathrm{N}(\mathbf{0}_n, \sigma^2 \mathbf{I}_n)$$

In matrix terms, the response vector is multivariate normal given $\mathbf{X}$:

$$(\mathbf{y}|\mathbf{X}) \sim \mathrm{N}(\mathbf{X}\mathbf{b}, \sigma^2 \mathbf{I}_n)$$

Polynomial Regression: Properties

Some important properties of the PR model include:

1. Need $n > p$ to fit the polynomial regression model
2. Setting $p = 1$ produces simple linear regression
3. Setting $p = 2$ is quadratic polynomial regression
4. Setting $p = 3$ is cubic polynomial regression
5. Rarely set $p > 3$; use cubic spline instead

Polynomial Regression: OLS Estimation

The ordinary least squares (OLS) problem is

$$\min_{\mathbf{b} \in \mathbb{R}^{p+1}} \|\mathbf{y} - \mathbf{X}\mathbf{b}\|^2$$

where $\|\cdot\|$ denotes the Frobenius norm.

The OLS solution has the form

$$\hat{\mathbf{b}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

which is the same formula from SLR and MLR!

Fitted Values and Residuals

SCALAR FORM:

Fitted values are given by

$$\hat{y}_i = \hat{b}_0 + \sum_{j=1}^{p} \hat{b}_j x_i^j$$

and *residuals* are given by

$$\hat{e}_i = y_i - \hat{y}_i$$

MATRIX FORM:

Fitted values are given by

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{b}} = \mathbf{H}\mathbf{y}$$

and *residuals* are given by

$$\hat{\mathbf{e}} = \mathbf{y} - \hat{\mathbf{y}} = (\mathbf{I}_n - \mathbf{H})\mathbf{y}$$

Estimated Error Variance (Mean Squared Error)

The estimated error variance is

$$\begin{aligned}\hat{\sigma}^2 &= SSE/(n-p-1) \\ &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2/(n-p-1) \\ &= \|(\mathbf{I}_n - \mathbf{H})\mathbf{y}\|^2/(n-p-1)\end{aligned}$$

which is an unbiased estimate of error variance σ^2.

The estimate $\hat{\sigma}^2$ is the *mean squared error (MSE)* of the model.

Distribution of Estimator, Fitted Values, and Residuals

Just like in SLR and MLR, the PR assumptions imply that

$$\hat{\mathbf{b}} \sim \mathrm{N}(\mathbf{b}, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$$

$$\hat{\mathbf{y}} \sim \mathrm{N}(\mathbf{X}\mathbf{b}, \sigma^2\mathbf{H})$$

$$\hat{\mathbf{e}} \sim \mathrm{N}(\mathbf{0}, \sigma^2(\mathbf{I}_n - \mathbf{H}))$$

Typically σ^2 is unknown, so we use the MSE $\hat{\sigma}^2$ in practice.

Multicollinearity: Problem

Note that x_i, x_i^2, x_i^3, etc. can be highly correlated with one another, which introduces multicollinearity problem.

```
> set.seed(123)
> x=runif(100)*2
> X=cbind(x,xsq=x^2,xcu=x^3)
> cor(X)
            x       xsq       xcu
x   1.0000000 0.9703084 0.9210726
xsq 0.9703084 1.0000000 0.9866033
xcu 0.9210726 0.9866033 1.0000000
```

Multicollinearity: Partial Solution

You could mean-center the x_i terms to reduce multicollinearity.

```
> set.seed(123)
> x=runif(100)*2
> x=x-mean(x)
> X=cbind(x,xsq=x^2,xcu=x^3)
> cor(X)
             x        xsq        xcu
x   1.00000000 0.03854803 0.91479660
xsq 0.03854803 1.00000000 0.04400704
xcu 0.91479660 0.04400704 1.00000000
```

But this doesn't fully solve our problem…

Orthogonal Polynomials: Definition

To deal with multicollinearity, define the set of variables

$$
\begin{aligned}
z_0 &= a_0 \\
z_1 &= a_1 + b_1 x \\
z_2 &= a_2 + b_2 x + c_2 x^2 \\
z_3 &= a_3 + b_3 x + c_3 x^2 + d_3 x^3
\end{aligned}
$$

where the coefficients are chosen so that $z_j' z_k = 0$ for all $j \neq k$.

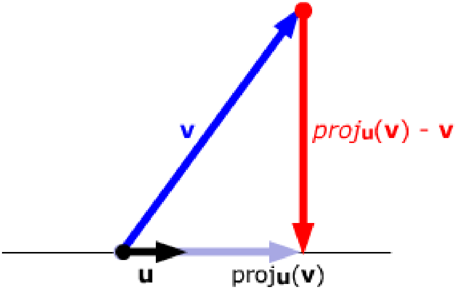
The transformed z_j variables are called *orthogonal polynomials*.

Orthogonal Polynomials: Orthogonal Projection

The orthogonal projection of a vector $\mathbf{v} = \{v_i\}_{n \times 1}$ on to the line spanned by the vector $\mathbf{u} = \{u_i\}_{n \times 1}$ is

$$\text{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$$

where $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}'\mathbf{y}$ and $\langle \mathbf{u}, \mathbf{u} \rangle = \mathbf{u}'\mathbf{u}$ denote the inner products.



http://thejuniverse.org/PUBLIC/LinearAlgebra/LOLA/dotProd/proj.html

Orthogonal Polynomials: Gram-Schmidt

Can use the Gram-Schmidt process to form orthogonal polynomials.

Start with a linearly independent design matrix $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4]$ where $\mathbf{x}_j = (x_1^j, \ldots, x_n^j)'$ is the j-th order polynomial vector.

Gram-Schmidt algorithm to form columnwise orthogonal matrix $\mathbf{Z}$ that spans the same column space as $\mathbf{X}$:

$$\begin{aligned}
\mathbf{z}_0 &= \mathbf{x}_0 \\
\mathbf{z}_1 &= \mathbf{x}_1 - \mathrm{proj}_{\mathbf{z}_0}(\mathbf{x}_1) \\
\mathbf{z}_2 &= \mathbf{x}_2 - \mathrm{proj}_{\mathbf{z}_0}(\mathbf{x}_2) - \mathrm{proj}_{\mathbf{z}_1}(\mathbf{x}_2) \\
\mathbf{z}_3 &= \mathbf{x}_3 - \mathrm{proj}_{\mathbf{z}_0}(\mathbf{x}_3) - \mathrm{proj}_{\mathbf{z}_1}(\mathbf{x}_3) - \mathrm{proj}_{\mathbf{z}_2}(\mathbf{x}_3)
\end{aligned}$$

Orthogonal Polynomials: R Functions

Simple R function to orthogonalize an input matrix:

```
orthog<-function(X,normalize=FALSE){
  np=dim(X)
  Z=matrix(0,np[1],np[2])
  Z[,1]=X[,1]
  for(k in 2:np[2]){
    Z[,k]=X[,k]
    for(j in 1:(k-1)){
      Z[,k]=Z[,k]-Z[,j]*sum(Z[,k]*Z[,j])/sum(Z[,j]^2)
    }
  }
  if(normalize){Z=Z%*%diag(colSums(Z^2)^-0.5)}
  Z
}
```

Orthogonal Polynomials: R Functions (continued)

```
> set.seed(123)
> X=cbind(1,runif(10),runif(10))
> crossprod(X)
          [,1]     [,2]     [,3]
[1,] 10.000000 5.782475 5.233693
[2,]  5.782475 4.125547 2.337238
[3,]  5.233693 2.337238 3.809269
> Z=orthog(X)
> crossprod(Z)
              [,1]          [,2]          [,3]
[1,]  1.000000e+01 -4.440892e-16 -4.440892e-16
[2,] -4.440892e-16  7.818448e-01 -1.387779e-17
[3,] -4.440892e-16 -1.387779e-17  4.627017e-01
> Z=orthog(X,norm=TRUE)
> crossprod(Z)
              [,1]          [,2]          [,3]
[1,]  1.000000e+00 -1.942890e-16 -2.220446e-16
[2,] -1.942890e-16  1.000000e+00  1.387779e-17
[3,] -2.220446e-16  1.387779e-17  1.000000e+00
```

Orthogonal Polynomials: R Functions (continued)

Can also use the default `poly` function in R.

```
> set.seed(123)
> x=runif(10)
> X=cbind(1,x,xsq=x^2,xcu=x^3)
> Z=orthog(X,norm=TRUE)
> z=poly(x,degree=3)
> Z[,2:4]=Z[,2:4]%*%diag(colSums(z^2)^0.5)
> Z[1:3,]
          [,1]       [,2]        [,3]       [,4]
[1,] 0.3162278 -0.3287304 -0.07537277  0.5363745
[2,] 0.3162278  0.2375627 -0.06651752 -0.5097714
[3,] 0.3162278 -0.1914349 -0.26206273  0.2473705
> cbind(Z[1:3,1],z[1:3,])
                         1           2          3
[1,] 0.3162278 -0.3287304 -0.07537277  0.5363745
[2,] 0.3162278  0.2375627 -0.06651752 -0.5097714
[3,] 0.3162278 -0.1914349 -0.26206273  0.2473705
```

Real Polynomial Data

Auto-MPG data from the UCI Machine Learning repository:
http://archive.ics.uci.edu/ml/datasets/Auto+MPG

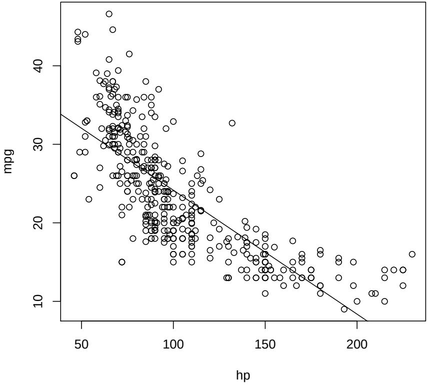
Have variables collected from $n = 398$ cars from years 1970–1982.

`mpg`	miles per gallon
`cylinder`	numer of cylinders
`disp`	displacement
`hp`	horsepower
`weight`	weight
`accel`	acceleration
`year`	model year
`origin`	origin
`name`	make and model

Best Linear Relationship

Best linear relationship predicting `mpg` from `hp`.

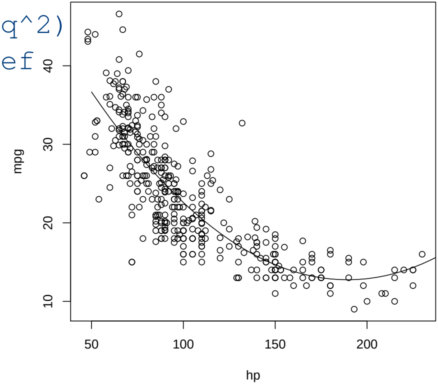
```
> plot(hp,mpg)
> linmod=lm(mpg~hp)
> abline(linmod)
```



Best Quadratic Relationship

Best quadratic relationship predicting `mpg` from `hp`.

```
> quadmod=lm(mpg~hp+I(hp^2))
> hpseq=seq(50,250,by=5)
> Xmat=cbind(1,hpseq,hpseq^2)
> hphat=Xmat%*%quadmod$coef
> plot(hp,mpg)
> lines(hpseq,hphat)
```



Best Cubic Relationship

Check for possible cubic relationship:

```
> cubmod=lm(mpg~hp+I(hp^2)+I(hp^3))
> summary(cubmod)

Call:
lm(formula = mpg ~ hp + I(hp^2) + I(hp^3))

Residuals:
     Min       1Q   Median       3Q      Max
-14.7039  -2.4491  -0.1519   2.2035  15.8159

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.068e+01  4.563e+00  13.298  < 2e-16 ***
hp          -5.689e-01  1.179e-01  -4.824 2.03e-06 ***
I(hp^2)      2.079e-03  9.479e-04   2.193   0.0289 *
I(hp^3)     -2.147e-06  2.378e-06  -0.903   0.3673
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.375 on 388 degrees of freedom
  (6 observations deleted due to missingness)
Multiple R-squared:  0.6882, Adjusted R-squared:  0.6858
F-statistic: 285.5 on 3 and 388 DF,  p-value: < 2.2e-16
```

Orthogonal versus Raw Polynomials

Compare orthogonal and raw polynomials:

```
> quadomod=lm(mpg~poly(hp,degree=2))
> summary(quadomod)$coef
                       Estimate Std. Error   t value      Pr(>|t|)
(Intercept)            23.44592  0.2209163 106.13030 2.752212e-289
poly(hp, degree = 2)1 -120.13774  4.3739206 -27.46683  4.169400e-93
poly(hp, degree = 2)2   44.08953  4.3739206  10.08009  2.196340e-21
> summary(quadmod)$coef
               Estimate   Std. Error   t value      Pr(>|t|)
(Intercept) 56.900099702 1.8004268063  31.60367 1.740911e-109
hp          -0.466189630 0.0311246171 -14.97816  2.289429e-40
I(hp^2)      0.001230536 0.0001220759  10.08009  2.196340e-21
```

Orthogonal Polynomials from Scratch

We can reproduce the same significance test results using `orthog`:

```
> widx=which(is.na(hp)==FALSE)
> hp=hp[widx]
> mpg=mpg[widx]
> X=orthog(cbind(1,hp,hp^2))
> quadomod=lm(mpg~X[,2]+X[,3])
> summary(quadomod)$coef
               Estimate   Std. Error   t value      Pr(>|t|)
(Intercept) 23.445918367 0.2209163488 106.13030 2.752212e-289
X[, 2]      -0.157844733 0.0057467395 -27.46683  4.169400e-93
X[, 3]       0.001230536 0.0001220759  10.08009  2.196340e-21
```

Interaction Term: Definition

MLR model with two predictors and an interaction

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} x_{i2} + e_i$$

where

- $y_i \in \mathbb{R}$ is the real-valued response for the i-th observation
- $b_0 \in \mathbb{R}$ is the regression intercept
- $b_1 \in \mathbb{R}$ is the first predictor's main effect slope
- $b_2 \in \mathbb{R}$ is the second predictor's main effect slope
- $b_3 \in \mathbb{R}$ is the interaction effect slope
- $e_i \stackrel{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is Gaussian measurement error

Interaction Term: Interpretation

An interaction between X_1 and X_2 means that the relationship between X_1 and Y differs depending on the value of X_2 (and vice versa).

Pro: model is more flexible (i.e., we've added a parameter)

Con: model is (sometimes) more difficult to interpret.

Nominal Variables

Suppose that $X \in \{x_1, \ldots, x_g\}$ is a nominal variable with g levels.

- Nominal variables are also called categorical variables
- Example: sex $\in \{\text{female}, \text{male}\}$ has two levels
- Example: drug $\in \{\text{A}, \text{B}, \text{C}\}$ has three levels

To code a nominal variable (with g levels) in a regression model, we need to include $g-1$ different variables in the model.

- Use "dummy" coding to absorb g-th level into intercept
- $x_{ij} = \begin{cases} 1 & \text{if } i\text{-th observation is in } j\text{-th level} \\ 0 & \text{otherwise} \end{cases}$
  for $j \in \{1, \ldots, g-1\}$

Nominal Interaction with Two Levels

Revisit the MLR model with two predictors and an interaction

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} x_{i2} + e_i$$

and suppose that $x_{i2} \in \{0, 1\}$ is a nominal predictor.

In this case, the model terms can be interpreted as:

- b_0 is expected value of Y when $x_{i1} = x_{i2} = 0$
- b_1 is expected change in Y corresponding to 1-unit change in x_{i1} for observations with $x_{i2} = 0$
- b_2 is expected change in Y corresponding to $x_{i2} = 1$ level
- $(b_1 + b_3)$ is expected change in Y corresponding to 1-unit change in x_{i1} for observations with $x_{i2} = 1$

Real Estate Data Description

Using house price data from Kutner, Nachtsheim, Neter, and Li (2005).

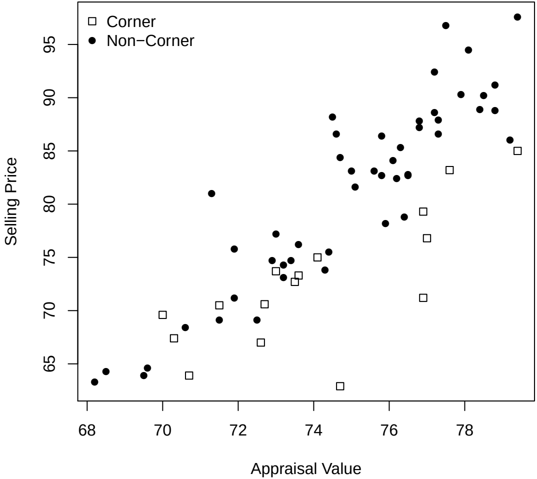
Have three variables in the data set:

- `price`: price house sells for (thousands of dollars)
- `value`: house value before selling (thousands of dollars)
- `corner`: indicator variable (=1 if house is on corner of block)

Total of $n = 64$ independent observations (16 corners).

Want to predict selling `price` from appraisal `value`, and determine if the relationship depends on the `corner` status of the house.

Real Estate Data Visualization



Real Estate Data Visualization (R Code)

```
> house=read.table("/Users/Nate/Desktop/houseprice.txt",header=TRUE)
> house[1:3,]
  price value corner
1  78.8  76.4      0
2  73.8  74.3      0
3  64.6  69.6      0
4  76.2  73.6      0
5  87.2  76.8      0
6  70.6  72.7      1
7  86.0  79.2      0
8  83.1  75.6      0
> plot(house$value,house$price,pch=ifelse(house$corner==1,0,16),
+      xlab="Appraisal Value",ylab="Selling Price")
> legend("topleft",c("Corner","Non-Corner"),pch=c(0,16),bty="n")
```

Real Estate Regression: Fit Model

Fit model with interaction between `value` and `corner`

```
> hmod=lm(price~value*corner,data=house)
> hmod

Call:
lm(formula = price ~ value * corner, data = house)

Coefficients:
 (Intercept)         value        corner  value:corner  
    -126.905         2.776        76.022        -1.107  
```

Real Estate Regression: Significance of Terms

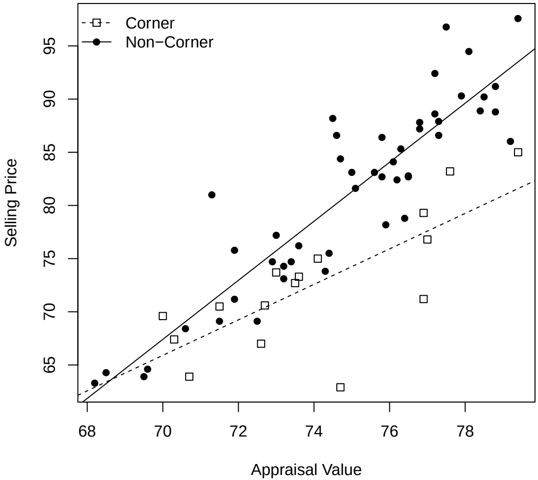
```
> summary(hmod)

Call:
lm(formula = price ~ value * corner, data = house)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.8470  -2.1639   0.0913   1.9348   9.9836 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -126.9052    14.7225  -8.620 4.33e-12 ***
value          2.7759     0.1963  14.142  < 2e-16 ***
corner        76.0215    30.1314   2.523  0.01430 *  
value:corner  -1.1075     0.4055  -2.731  0.00828 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.893 on 60 degrees of freedom
Multiple R-squared:  0.8233, Adjusted R-squared:  0.8145 
F-statistic: 93.21 on 3 and 60 DF,  p-value: < 2.2e-16
```

Real Estate Regression: Interpreting Results

```
> hmod$coef
 (Intercept)        value       corner value:corner
 -126.905171     2.775898    76.021532    -1.107482
```

- $\hat{b}_0 = -126.90$ is expected selling price (in thousands of dollars) for non-corner houses that were valued at \$0.
- $\hat{b}_0 + \hat{b}_2 = -126.90 + 76.022 = -50.878$ is expected selling price (in thousands of dollars) for corner houses that were valued at \$0.
- $\hat{b}_1 = 2.776$ is the expected increase in selling price (in thousands of dollars) corresponding to a 1-unit (\$1,000) increase in appraisal value for non-corner houses
- $\hat{b}_1 + \hat{b}_3 = 2.775 - 1.107 = 1.668$ is the expected increase in selling price (in thousands of dollars) corresponding to a 1-unit (\$1,000) increase in appraisal value for corner houses

Real Estate Regression: Plotting Results



Real Estate Regression: Plotting Results (R Code)

```
> plot(house$value,house$price,pch=ifelse(house$corner==1,0,16),
+      xlab="Appraisal Value",ylab="Selling Price")
> abline(hmod$coef[1],hmod$coef[2])
> abline(hmod$coef[1]+hmod$coef[3],hmod$coef[2]+hmod$coef[4],lty=2)
> legend("topleft",c("Corner","Non-Corner"),lty=2:1,pch=c(0,16),bty="n")
```

Note that if you input the (appropriate) coefficients, you can still use the `abline` function to draw the regression lines.

Depression Data Description

Using depression data from Daniel (1999) *Biostatistics: A Foundation for Analysis in the Health Sciences.*

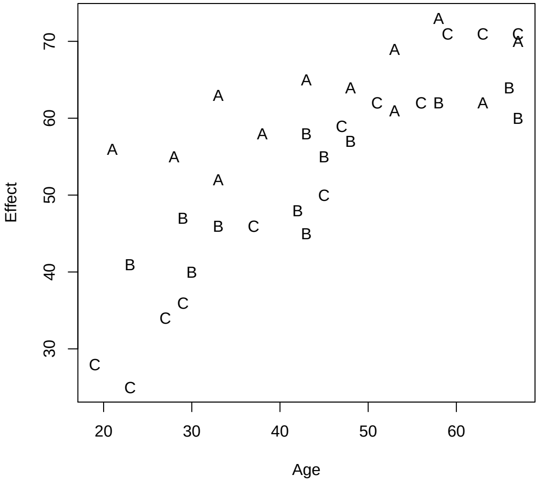
Total of $n = 36$ subjects subjects participated in a depression study.

Have three variables in the data set:

- `effect`: effectiveness of depression treatment (high=effective)
- `age`: age of the participant (in years)
- `method`: method of treatment (3 levels: A, B, C)

Predict effectiveness from participant's age and treatment method.

Depression Data Visualization



Depression Data Visualization (R Code)

```
> depression=read.table("/Users/Nate/Desktop/depression.txt",header=TRUE)
> depression[1:8,]
  effect age method
1     56  21      A
2     41  23      B
3     40  30      B
4     28  19      C
5     55  28      A
6     25  23      C
7     46  33      B
8     71  67      C
> plot(depression$age,depression$effect,xlab="Age",ylab="Effect",type="n")
> text(depression$age,depression$effect,depression$method)
```

Depression Regression: Fit Model

Fit model with interaction between `age` and `method`

```
> dmod=lm(effect~age*method,data=depression)
> dmod$coef
(Intercept)         age     methodB     methodC age:methodB age:methodC
 47.5155913   0.3305073 -18.5973852 -41.3042101   0.1931769   0.7028836
```

Note that R creates two indicator variables for `method`:

- $x_{iB} = \begin{cases} 1 & \text{if } i\text{-th observation is in treatment method B} \\ 0 & \text{otherwise} \end{cases}$
- $x_{iC} = \begin{cases} 1 & \text{if } i\text{-th observation is in treatment method C} \\ 0 & \text{otherwise} \end{cases}$

Depression Regression: Significance of Terms

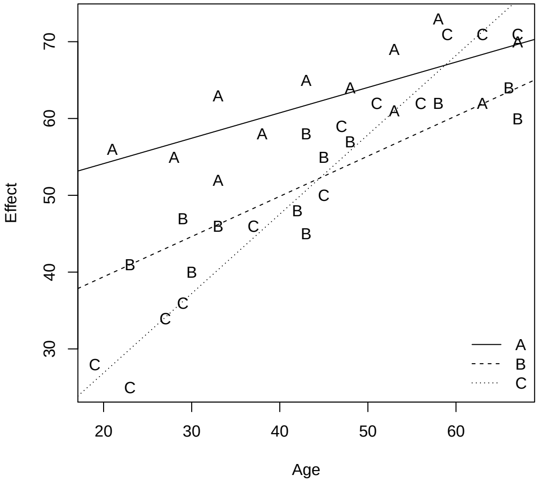
```
> summary(dmod)

Call:
lm(formula = effect ~ age * method, data = depression)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.4366 -2.7637  0.1887  2.9075  6.5634 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  47.51559    3.82523  12.422 2.34e-13 ***
age           0.33051    0.08149   4.056 0.000328 ***
methodB     -18.59739    5.41573  -3.434 0.001759 ** 
methodC     -41.30421    5.08453  -8.124 4.56e-09 ***
age:methodB   0.19318    0.11660   1.657 0.108001    
age:methodC   0.70288    0.10896   6.451 3.98e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.925 on 30 degrees of freedom
Multiple R-squared:  0.9143, Adjusted R-squared:  0.9001 
F-statistic: 64.04 on 5 and 30 DF,  p-value: 4.264e-15
```

Depression Regression: Interpreting Results

```
> dmod$coef
(Intercept)         age     methodB     methodC age:methodB age:methodC
 47.5155913   0.3305073 -18.5973852 -41.3042101   0.1931769   0.7028836
```

- $\hat{b}_0 = 47.516$ is expected treatment effectiveness for subjects in method A who are 0 y/o.
- $\hat{b}_0 + \hat{b}_2 = 47.516 - 18.598 = 28.918$ is expected treatment effectiveness for subjects in method B who are 0 years old.
- $\hat{b}_0 + \hat{b}_3 = 47.516 - 41.304 = 6.212$ is expected treatment effectiveness for subjects in method C who are 0 years old.
- $\hat{b}_1 = 0.331$ is the expected increase in treatment effectiveness corresponding to a 1-unit (1 year) increase in age for treatment method A
- $\hat{b}_1 + \hat{b}_4 = 0.331 + 0.193 = 0.524$ is the expected increase in treatment effectiveness corresponding to a 1-unit (1 year) increase in age for treatment method B
- $\hat{b}_1 + \hat{b}_5 = 0.331 + 0.703 = 1.033$ is the expected increase in treatment effectiveness corresponding to a 1-unit (1 year) increase in age for treatment method C

Depression Regression: Plotting Results



Depression Regression: Plotting Results

```
> plot(depression$age,depression$effect,xlab="Age",ylab="Effect",type="n")
> text(depression$age,depression$effect,depression$method)
> abline(dmod$coef[1],dmod$coef[2])
> abline(dmod$coef[1]+dmod$coef[3],dmod$coef[2]+dmod$coef[5],lty=2)
> abline(dmod$coef[1]+dmod$coef[4],dmod$coef[2]+dmod$coef[6],lty=3)
> legend("bottomright",c("A","B","C"),lty=1:3,bty="n")
```

Interactions between Continuous Variables

Revisit the MLR model with two predictors and an interaction

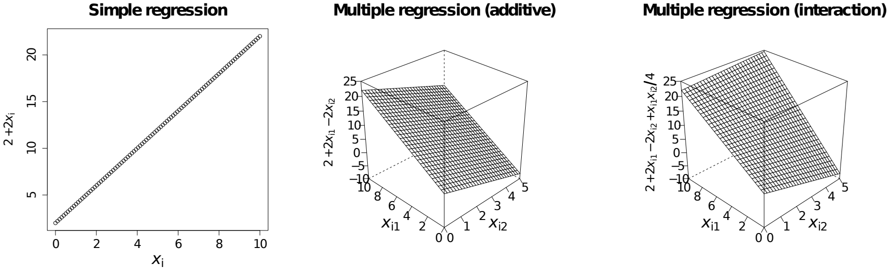
$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + b_3 x_{i1} x_{i2} + e_i$$

and suppose that $x_{i1}, x_{i2} \in \mathbb{R}$ are both continuous predictors.

In this case, we model terms can be interpreted as:

- b_0 is expected value of Y when $x_{i1} = x_{i2} = 0$
- $b_1 + b_3 x_{i2}$ is expected change in Y corresponding to 1-unit change in x_{i1} holding x_{i2} fixed (i.e., conditioning on x_{i2})
- $b_2 + b_3 x_{i1}$ is expected change in Y corresponding to 1-unit change in x_{i2} holding x_{i1} fixed (i.e., conditioning on x_{i1})

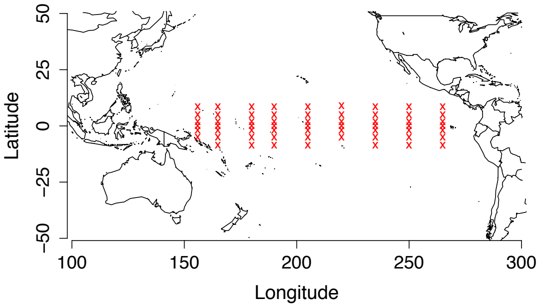
Visualizing Continuous*Continuous Interactions



Oceanography Data Description

Data from UCI Machine Learning: http://archive.ics.uci.edu/ml/

- Data originally from TAO project: http://www.pmel.noaa.gov/tao/
- Note that I have preprocessed the data a bit before analysis.



Oceanography Data Description (continued)

Buoys collect lots of different data:

```
> elnino[1:4,]
      obs year month day   date latitude longitude zon.winds mer.winds humidity air.temp ss.temp
4297 4297   94     1   1 940101    -0.01    250.01      -4.3       2.6     89.9    23.21   23.37
4298 4298   94     1   2 940102    -0.01    250.01      -4.1         1       90    23.16   23.45
4299 4299   94     1   3 940103    -0.01    250.01        -3       1.6     87.7    23.14   23.71
4300 4300   94     1   4 940104     0.00    250.00        -3       2.9     85.8    23.39   24.29
```

We will focus on predicting the sea surface temperatures (`ss.temp`) from the `latitude` and `longitude` locations of the buoys.

Oceanography Regression: Fit Additive Model

Fit additive model of `latitude` and `longitude`

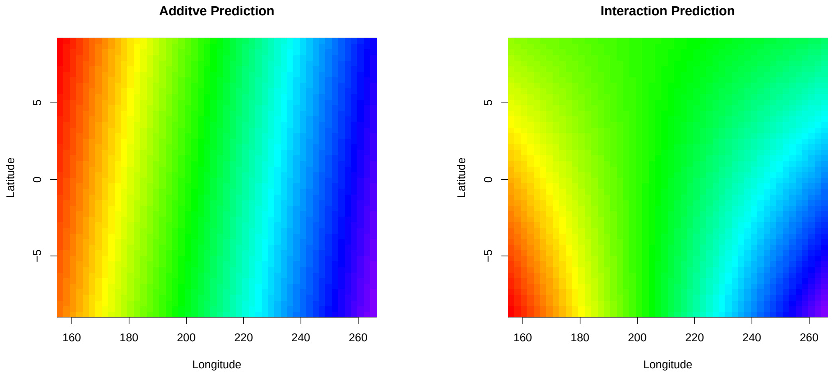
```
> eladd=lm(ss.temp~latitude+longitude,data=elnino)
> summary(eladd)

Call:
lm(formula = ss.temp ~ latitude + longitude, data = elnino)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.6055 -0.7229  0.1261  0.9039  5.0987 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 35.2636388  0.0305722 1153.45   <2e-16 ***
latitude     0.0257867  0.0010006   25.77   <2e-16 ***
longitude   -0.0357496  0.0001445 -247.33   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.48 on 86498 degrees of freedom
Multiple R-squared:  0.4184, Adjusted R-squared:  0.4184 
F-statistic: 3.112e+04 on 2 and 86498 DF,  p-value: < 2.2e-16
```

Oceanography Regression: Fit Interaction Model

Fit model with interaction between `latitude` and `longitude`

```
> elint=lm(ss.temp~latitude*longitude,data=elnino)
> summary(elint)

Call:
lm(formula = ss.temp ~ latitude * longitude, data = elnino)

Residuals:
    Min      1Q  Median      3Q     Max 
-7.5867 -0.6496  0.1000  0.8127  5.0922 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)         3.541e+01  2.913e-02 1215.61   <2e-16 ***
latitude           -5.245e-01  5.862e-03  -89.47   <2e-16 ***
longitude          -3.638e-02  1.377e-04 -264.22   <2e-16 ***
latitude:longitude  2.618e-03  2.752e-05   95.13   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.408 on 86497 degrees of freedom
Multiple R-squared:  0.4735, Adjusted R-squared:  0.4735 
F-statistic: 2.593e+04 on 3 and 86497 DF,  p-value: < 2.2e-16
```

Oceanography Regression: Visualize Results



Oceanography Regression: Visualize (R Code)

```
> newdata=expand.grid(longitude=seq(min(elnino$longitude),max(elnino$longitude),l=50),
+                     latitude=seq(min(elnino$latitude),max(elnino$latitude),l=50))
> yadd=predict(eladd,newdata)
> image(seq(min(elnino$longitude),max(elnino$longitude),l=50),
+       seq(min(elnino$latitude),max(elnino$latitude),l=50),
+       matrix(yadd,50,50),col=rev(rainbow(100,end=3/4)),
+       xlab="Longitude",ylab="Latitude",main="Additve Prediction")
> yint=predict(elint,newdata)
> image(seq(min(elnino$longitude),max(elnino$longitude),l=50),
+       seq(min(elnino$latitude),max(elnino$latitude),l=50),
+       matrix(yint,50,50),col=rev(rainbow(100,end=3/4)),
+       xlab="Longitude",ylab="Latitude",main="Interaction Prediction")
```

Oceanography Regression: Smoothing Spline Solution