Introduction to Linear Regression – Part 2

James H. Steiger

Department of Psychology and Human Development
Vanderbilt University

Introduction to Linear Regression – Part 2

James H. Steiger

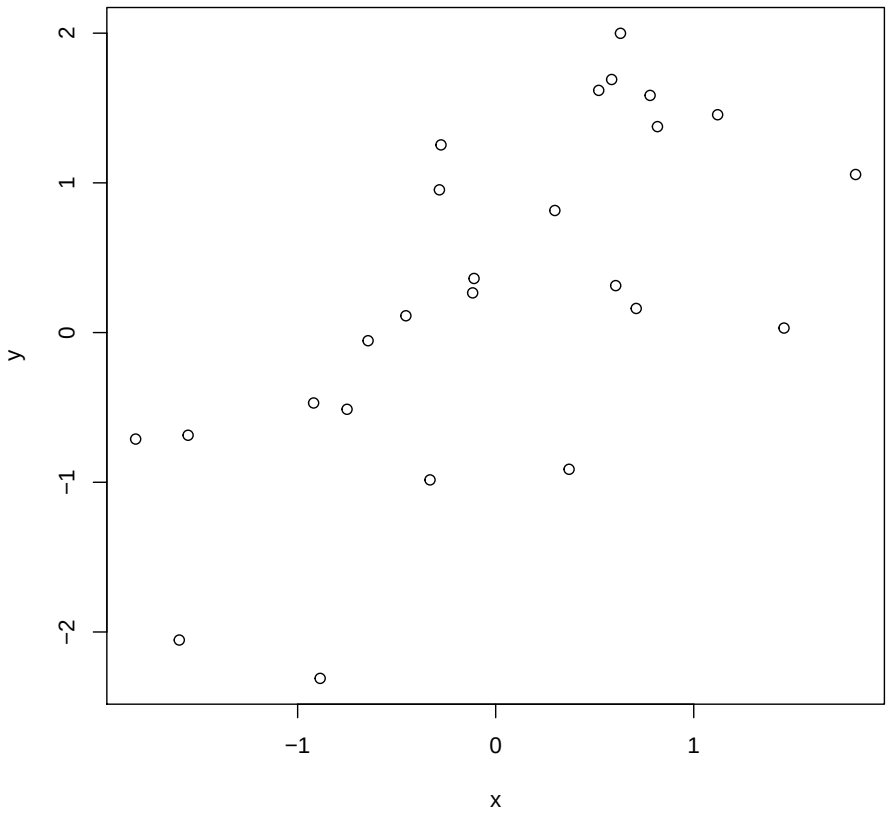
Department of Psychology and Human Development
Vanderbilt University

Introduction to Multiple Regression – Part 2

Basic Linear Regression in R

Let's define and plot some artificial data on two variables.

```
> set.seed(12345)
> x <- rnorm(25)
> y <- sqrt(1/2) * x + sqrt(1/2) * rnorm(25)
> plot(x, y)
```



Basic Linear Regression in R

We want to predict y from x using least squares linear regression.

We seek to fit a model of the form

$$y_i = \beta_0 + \beta_1 x_i + e_i = \hat{y}_i + e_i$$

while minimizing the sum of squared errors in the "up-down" plot direction.

We fit such a model in R by creating a "fit object" and examining its contents. We see that the formula for $\hat{y}_i$ is a straight line with slope β_1 and intercept β_0.

Basic Linear Regression in R

We start by creating the model with a model specification formula. This formula corresponds to the model stated on the previous slide in a specific way:

1. Instead of an equal sign, a "$\sim$" is used.
2. The coefficients themselves are not listed, only the predictor variables.
3. The error term is not listed
4. The intercept term generally does not need to be listed, but can be listed with a "1".

So the model on the previous page is translated as `y ~ x`.

Basic Linear Regression in R

We create the fit object as follows.

```
> fit.1 <- lm(y ~ x)
```

Once we have created the fit object, we can examine its contents.

```
> summary(fit.1)


Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max
-1.846 -0.669  0.213  0.508  1.233

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.255      0.175    1.45  0.15971
x              0.811      0.189    4.28  0.00028 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.877 on 23 degrees of freedom
Multiple R-squared:  0.444, Adjusted R-squared:  0.419
F-statistic: 18.3 on 1 and 23 DF,  p-value: 0.000279
```

Basic Linear Regression in R

We see the printed coefficients for the intercept and for x.

There are statistical t tests for each coefficient. These are tests of the null hypothesis that the coefficient is zero.

There is also a test of the hypothesis that the squared multiple correlation (the square of the correlation between $\hat{y}$ and y) is zero.

Standard errors are also printed, so you can compute confidence intervals. (How would you do that quickly "in your head?" (C.P.)

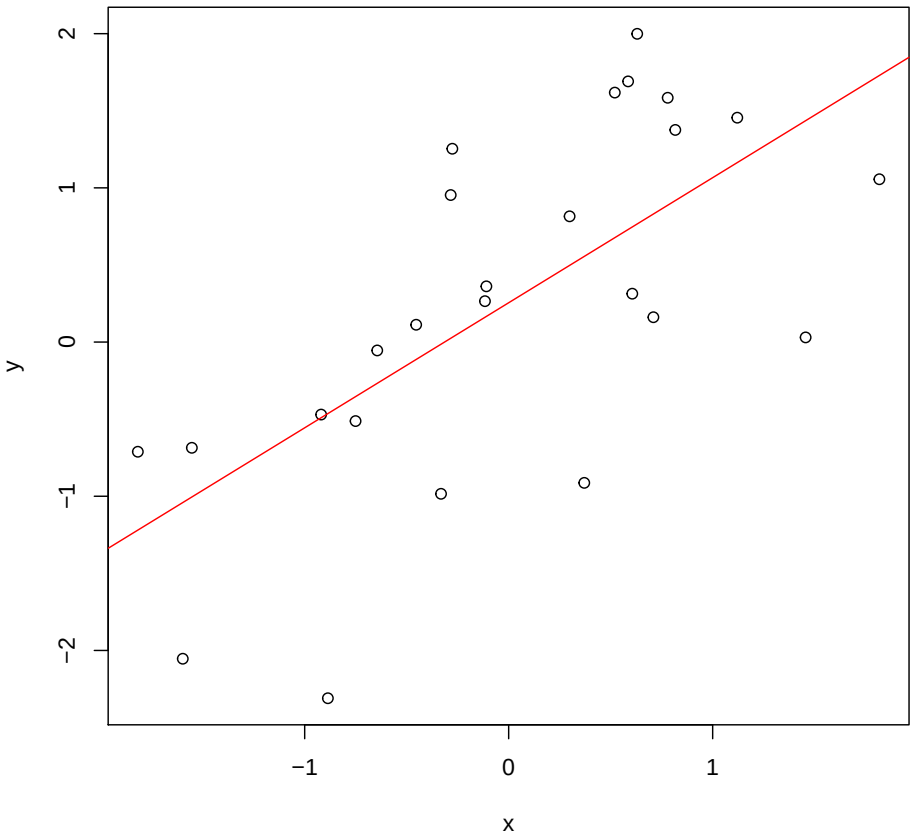
The intercept is not significantly different from zero. Does that surprise you? (C.P.)

The squared correlation is .4435. What is the correlation in the population? (C.P.)

Basic Linear Regression in R

If we want, we can, in the case of simple bivariate regression, add a regression line to the plot automatically using the `abline` function.

```
> plot(x, y)
> abline(fit.1, col = "red")
```



Assumptions of the Simple Linear Regression Model

The bivariate normal distribution is a bivariate continuous distribution characterized by the fact that any linear combination of the two variables is normal, and all conditional distributions of one variable for a given value of the other are normal, with constant variance. (See lecture notes on *Conditional Distributions and the Bivariate Normal Distribution*).

Consequences of a bivariate normal model for two variables Y and X include:

1. The conditional distribution of Y given X is normal
2. The conditional distribution of X given Y is normal
3. The conditional means for Y given X follow the linear regression line for predicting Y from X.
4. The conditional means for X given Y follow the linear regression line for predicting X from Y.
5. The conditional variance for Y given X is constant and is given by $(1 - \rho^2_{Y,X})\sigma^2_Y$, and the conditional variance for X given Y is constant and is given by $(1 - \rho^2_{Y,X})\sigma^2_X$.

The general linear regression model in actual use is *not* a bivariate normal model.

It does not assume that the independent variable X is a random variable at all.

Assumptions of the Simple Linear Regression Model

The standard simple linear regression model assumes only that the dependent (criterion) variable is a random variable.

The predictor variable (or variables) is not assumed to be a random variable.

Rather the X values are treated as observed constants.

However, the conditional mean of Y given X still follows a linear regression rule, the conditional variance of Y given X is still assumed to be constant, and, in the classic cases, the conditional distribution of Y given X is still assumed to be normal.

Assumptions of the Simple Linear Regression Model

Examining Residuals

Examination of residuals is a key technique for checking model assumptions in linear regression.

A correct model should show a "null plot" (essentially random) of residuals versus predicted scores.

But many other patterns can occur, and are symptomatic of model misfit or violation of model assumptions.

The next slide (Weisberg Figure 8.2) shows some typical patterns.

Assumptions of the Simple Linear Regression Model

Examining Residuals

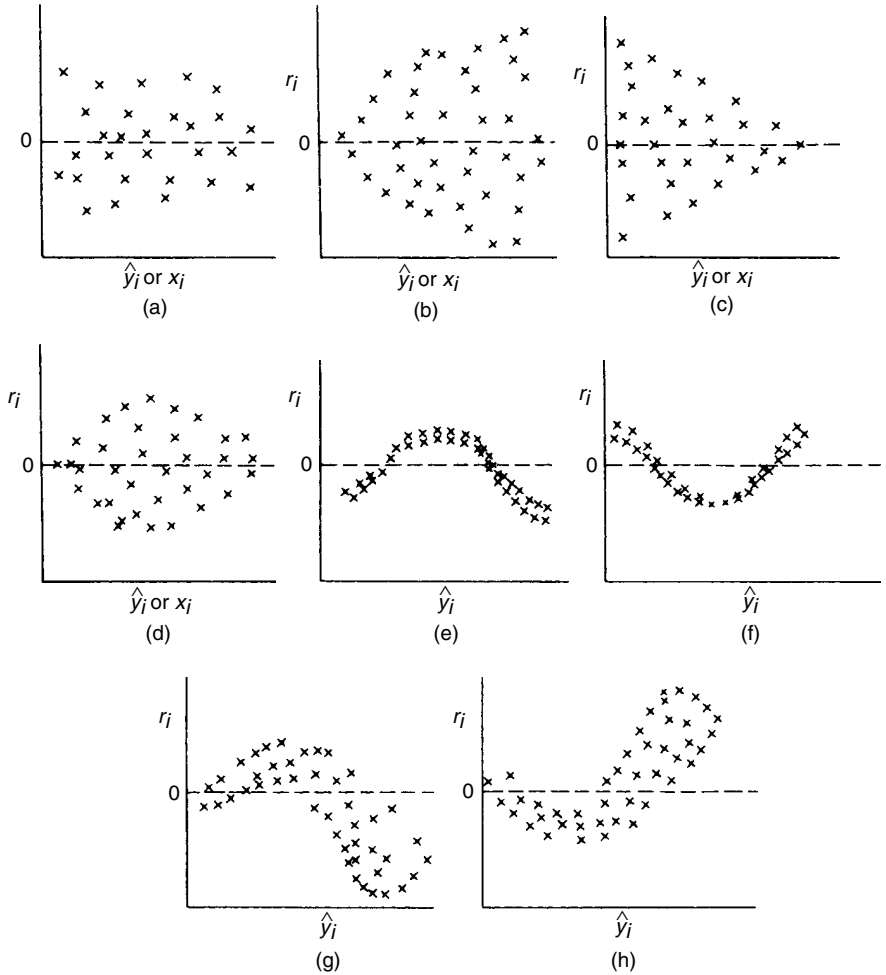


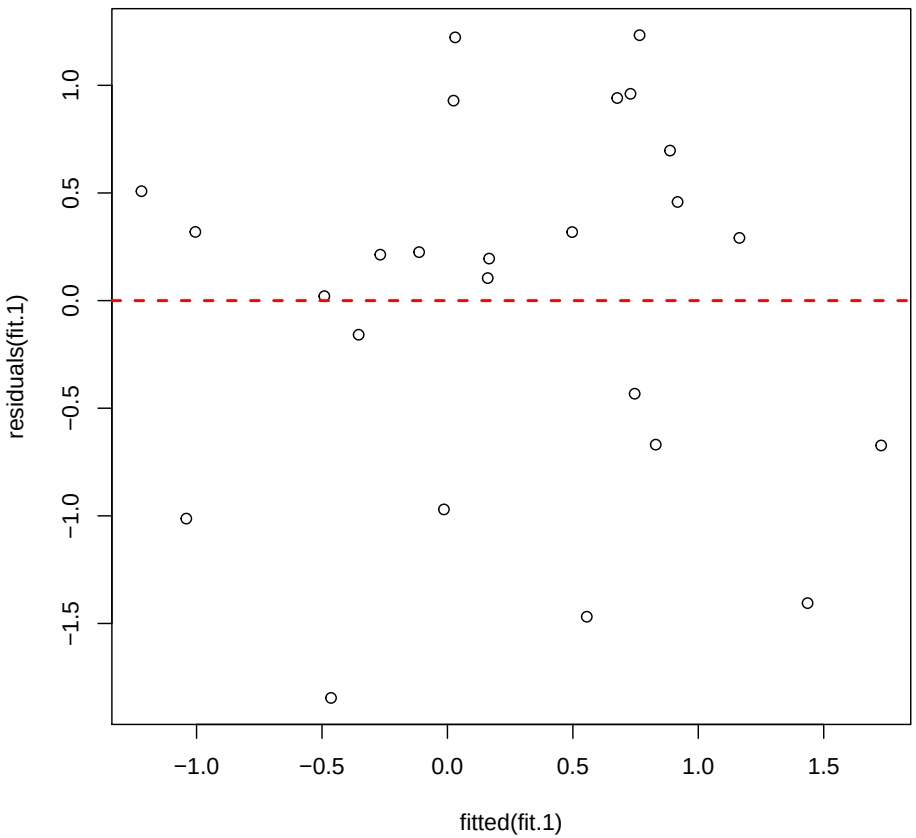
FIG. 8.2 Residual plots: (a) null plot; (b) right-opening megaphone; (c) left-opening megaphone; (d) double outward box; (e)–(f) nonlinearity; (g)–(h) combinations of nonlinearity and nonconstant variance function.

Assumptions of the Simple Linear Regression Model

Examining Residuals

Plotting the residuals is straightforward. We can see here that there is no noticeable departure from a null plot.

```
> plot(fitted(fit.1), residuals(fit.1))
> abline(h = 0, col = "red", lty = 2, lwd = 2)
```



Multiple Regression in R

If we have more than one predictor, we have a multiple regression model. Suppose, for example, we add another predictor w to our artificial data set. We design this predictor to be completely uncorrelated with the other predictor and the criterion, so this predictor is, in the population, of no value. Now our model becomes

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 w_i + e_i$$

```
> w <- rnorm(25)
```

Multiple Regression in R

How would we set up and fit the model

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 w_i + e_i$$

in R?

Multiple Regression in R

How would we set up and fit the model

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 w_i + e_i$$

in R? That's right,

```
> fit.2 <- lm(y ~ x + w)
```

Multiple Regression in R

```
> summary(fit.2)


Call:
lm(formula = y ~ x + w)

Residuals:
   Min     1Q Median     3Q    Max
-1.847 -0.669  0.220  0.511  1.230

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.25404    0.18183    1.40  0.17631
x            0.81273    0.20213    4.02  0.00057 ***
w            0.00437    0.15224    0.03  0.97738
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.897 on 22 degrees of freedom
Multiple R-squared:  0.444,Adjusted R-squared:  0.393
F-statistic: 8.77 on 2 and 22 DF,  p-value: 0.00158
```

Nested Models

Introduction

The situation we examined in the previous sections is a simple example of a *sequence of nested models*. One model is *nested within* another if it is a special case of the other in which some model coefficients are constrained to be zero. The model with only x as a predictor is a special case of the model with x and w as predictors, with the coefficient β_2 constrained to be zero.

Nested Models

Model Comparison

When two models are nested multiple regression models, there is a simple procedure for comparing them. This procedure tests whether the more complex model is significantly better than the simpler model. In the sample, of course, the more complex of two nested models will always fit at least as well as the less complex model.

Nested Models

Partial F-Tests: A General Approach

Suppose Model A includes Model B as a special case. That is, Model B is a special case of Model A where some terms have coefficients of zero. Then Model B is nested within Model A. If we define SS_a to be the sum of squared residuals for Model A, SS_b the sum of squared residuals for Model B. Since Model B is a special case of Model A, model A is more complex so SS_b will always be as least as large as SS_a. We define df_a to be $n - p_a$, where p_a is the number of terms in Model A including the intercept, and correspondingly $df_b = n - p_b$. Then, to compare Model B against Model A, we compute the partial $F-$statistic as follows.

$$F_{df_a - df_b, df_a} = \frac{MS_{comparison}}{MS_{res}} = \frac{(SS_b - SS_a)/(p_a - p_b)}{SS_a/df_a} \tag{1}$$

Nested Models

Partial *F*-Tests: A General Approach

R will perform the partial *F*-test automatically, using the `anova` command.

```
> anova(fit.1, fit.2)

Analysis of Variance Table

Model 1: y ~ x
Model 2: y ~ x + w
  Res.Df  RSS Df Sum of Sq  F Pr(>F)
1     23 17.7
2     22 17.7  1  0.000661  0   0.98
```

Note that the *p* value for the model difference test is the same as the *p* value for the *t*-test of the significance of the coefficient for *w* shown previously.

Nested Models

Partial *F*-Tests: A General Approach

What happens if we call the `anova` command with just a single model?

```
> anova(fit.1)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x          1   14.1   14.10    18.3 0.00028 ***
Residuals 23   17.7    0.77
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the *p*-value for this test is the same as the *p*-value for the overall test of zero squared multiple correlation shown in the output summary for `fit.1`. What is going on?

Nested Models

Partial *F*-Tests: A General Approach

It turns out, if you call the `anova` command with a single fit object, it startes by comparing the first non-intercept term in the model against a baseline model with no predictors (i.e., just an intercept). If there is a second predictor, it compares the model with both predictors against the model with just one predictor. It produces this sequence of comparisons automatically. To demonstrate, let's fit a model with just an intercept.

```
> fit.0 <- lm(y ~ 1)
```

Recall that the 1 in the model formula stands for the intercept. No let's perform a partial *F*-test comparing `fit.0` with `fit.1`.

Nested Models

Partial *F*-Tests: A General Approach

Here we go.

```
> anova(fit.0, fit.1)

Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ x
  Res.Df  RSS Df Sum of Sq    F  Pr(>F)
1     24 31.8
2     23 17.7  1      14.1 18.3 0.00028 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that we get exactly the same result for the model comparison as we got when we ran `anova` on just the `fit.1` object.

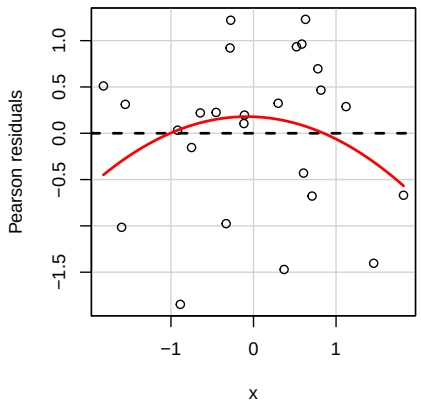
Residual Plots for Multiple Regression

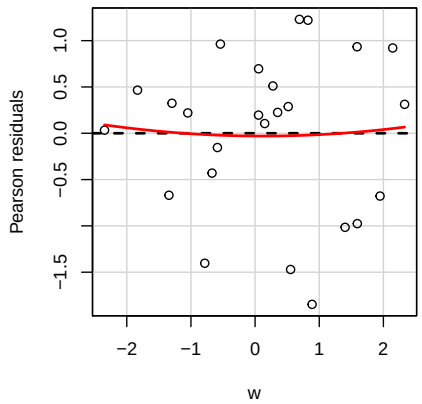
If you have the ALR4 library loaded, you can construct several residual plots at once with a single command. The function also prints results from Tukey's test for nonadditivity. A significant result indicates a departure from a null residual plot. In this case, none of the tests is significant.

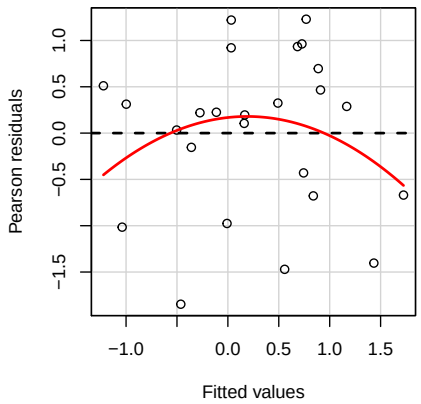
Nested Models

Partial *F*-Tests: A General Approach

```
> library(alr4)
> residualPlots(fit.2)
```





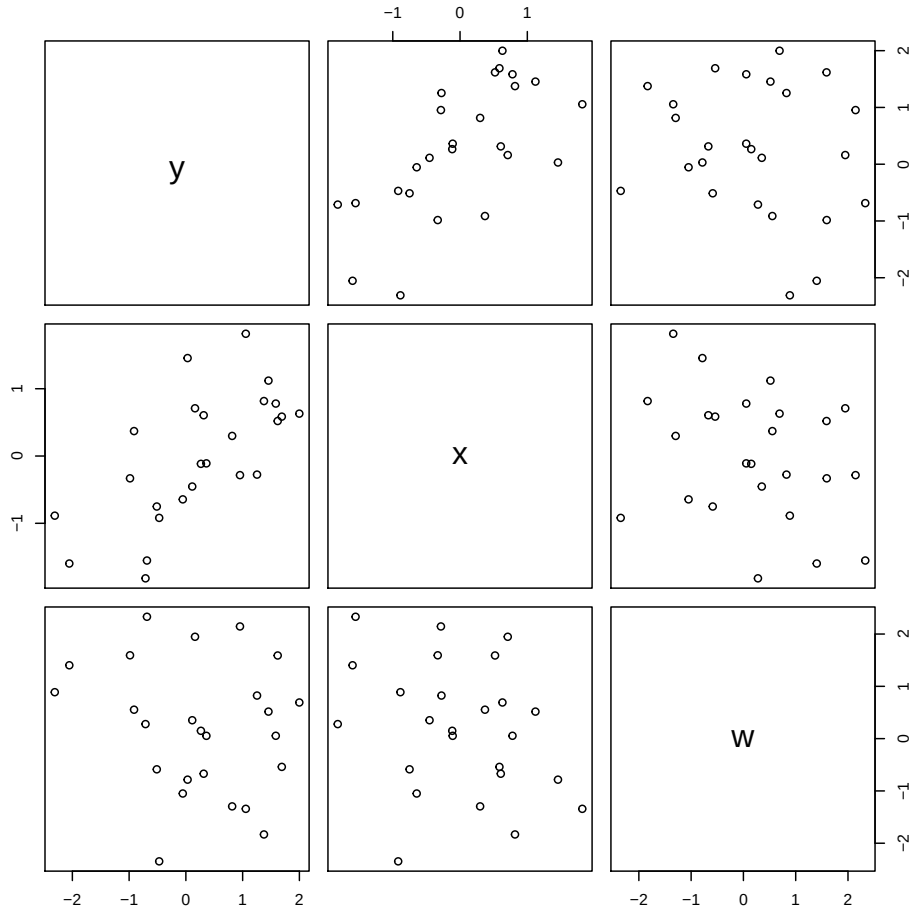



```
           Test stat Pr(>|t|)
x             -1.167    0.256
w              0.184    0.856
Tukey test    -1.164    0.244
```

The Scatterplot Matrix

The *scatterplot matrix* presents several scatterplots in an array. It uses the `pairs` command in its rudimentary form, as shown below. If you load the `car` library and use the `scatterplotMatrix` command, you can get much more detailed information, as shown on the next slide.

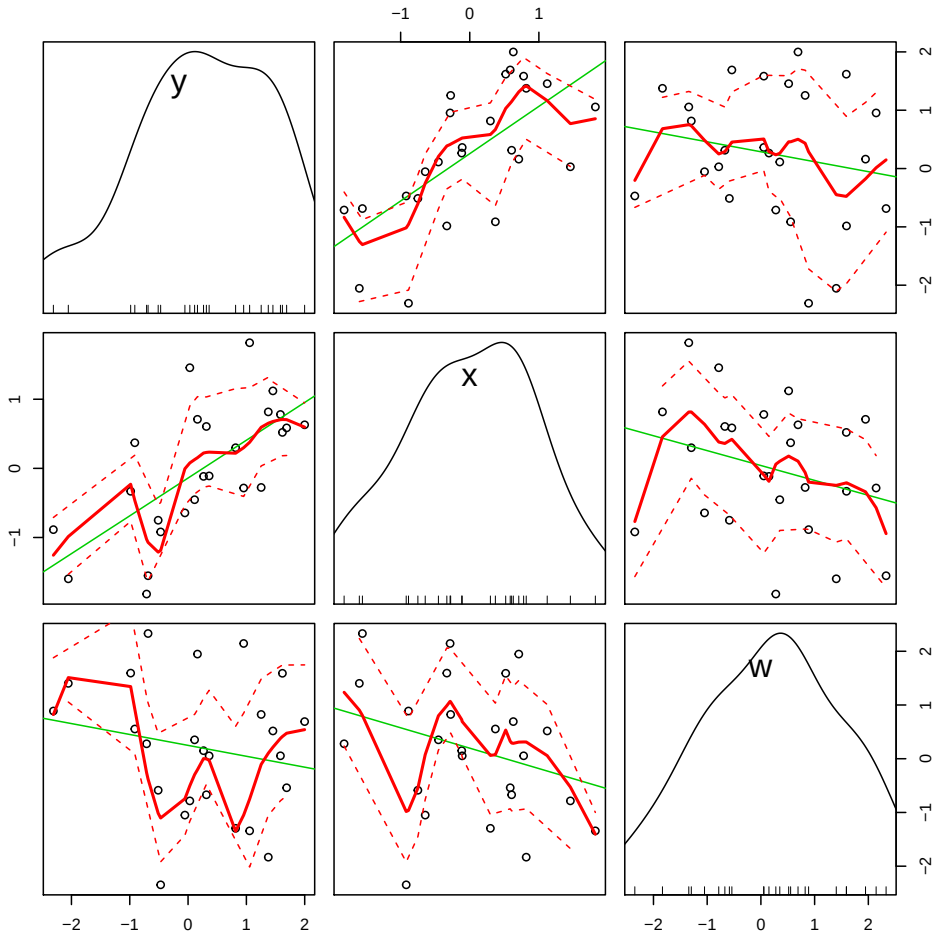
```
> pairs(cbind(y, x, w))
```



The Scatterplot Matrix

This plot includes linear and non-parametric fits to the data, as well as giving density plots on the diagonals. These plots are rough, because of the small sample size.

```
> scatterplotMatrix(cbind(y, x, w))
```



Fitting Regression Models

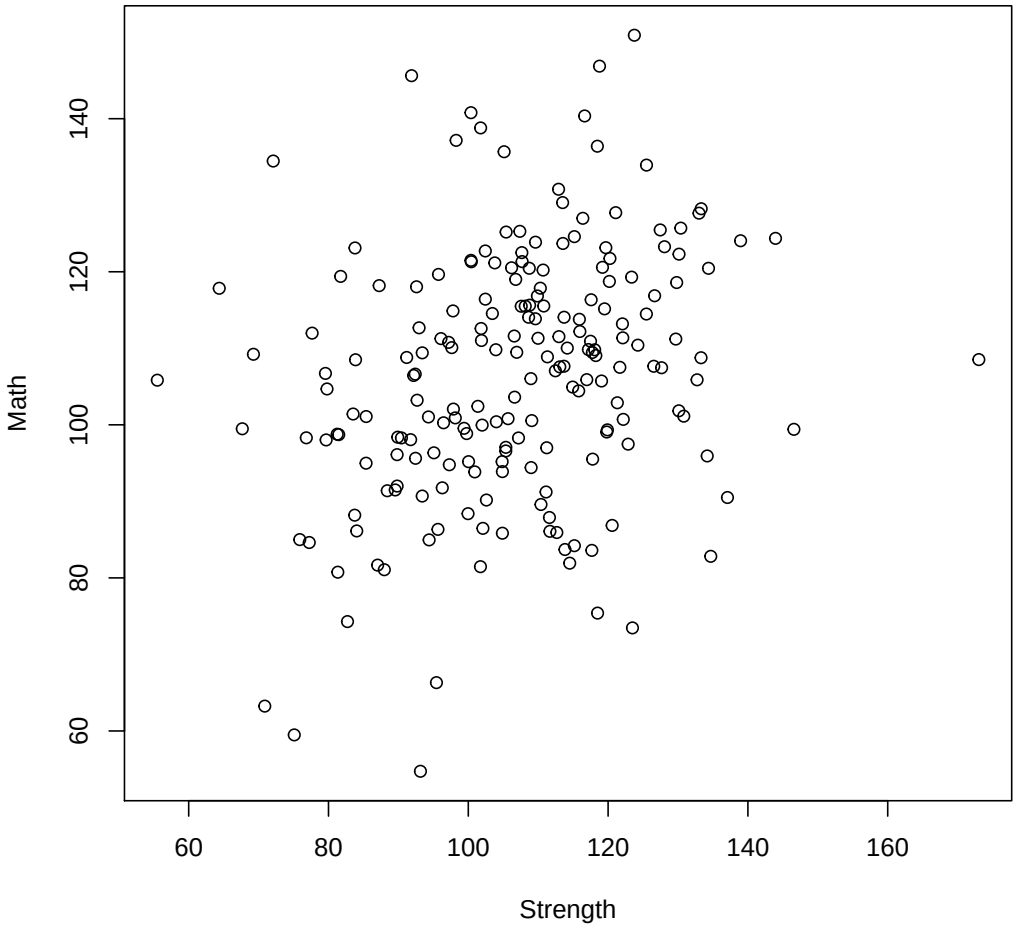
In an earlier lecture, we saw formulas for the slope and intercept of the best-fitting linear regression line relating two variables Y and X. Let's load in some artificial data and review those formulas. The data can be downloaded from a file called *regression.data.txt*. There are 3 variables in the file, which represents scores on *Math* and *Strength* for 100 sixth grade boys and 100 eighth grade boys. The *Grade* variable is coded 0 for sixth graders and 1 for eighth graders.

```
> data <- read.csv("http://www.statpower.net/data/regression.data.csv")
> attach(data)
```

Fitting Regression Models

If we plot all the data together, we get a scatterplot like this:

```
> plot(Strength, Math)
```



Fitting Regression Models

In the earlier lecture, we saw that the regression slope and intercept for the best fitting straight line $\hat{Y} = \beta_1 X + \beta_0$ can be estimated as

$$\begin{aligned} \hat{\beta}_1 &= r_{yx} s_y / s_x \\ \hat{\beta}_0 &= \overline{Y}_\bullet - \beta_1 \overline{x}_\bullet \end{aligned}$$

We can compute the values easily for predicting *Math* from *Strength* as

```
> beta.hat.1 <- cor(Math, Strength) * sd(Math)/sd(Strength)
> beta.hat.0 <- mean(Math) - beta.hat.1 * mean(Strength)
> beta.hat.1

[1] 0.2568

> beta.hat.0

[1] 79.26
```

Fitting Regression Models

We can fit the model with R, of course.

```
> model.1 <- lm(Math ~ 1 + Strength)
```

The call to `lm` included a model specification. The "1" stands for the intercept term. All variable names are included as predictors. So, the above function call fits the linear model

$$Math = \beta_0 + \beta_1 Strength + e.$$

Fitting Regression Models

To see the numerical results of the model fit, you can use the function `summary` on the model fit object.

```
> summary(model.1)


Call:
lm(formula = Math ~ 1 + Strength)

Residuals:
   Min     1Q Median     3Q    Max
-48.44 -10.60   0.07   9.79  42.77

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   79.255      7.010   11.31  < 2e-16 ***
Strength       0.257      0.065    3.95  0.00011 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.7 on 198 degrees of freedom
Multiple R-squared:  0.073,Adjusted R-squared:  0.0684
F-statistic: 15.6 on 1 and 198 DF,  p-value: 0.000109
```

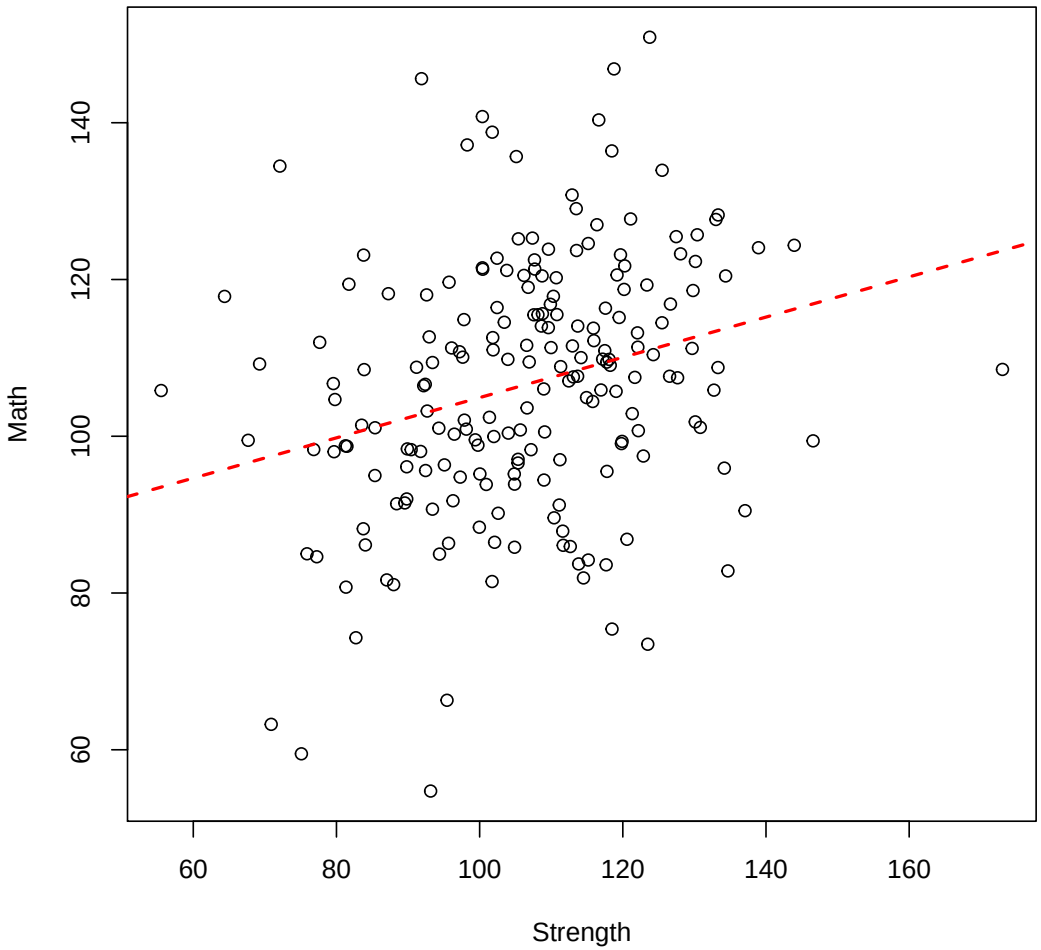
Notice that, in this case, both β_0 (the intercept) and β_1 (the coefficient of *Strength*) are statistically significant, having p-values less than .001.

Fitting Regression Models

To plot the regression line, you can use the `abline` function on the linear model object. I chose to plot a dotted red line.

Fitting Regression Models

```
> plot(Strength, Math)
> abline(model.1, col = "red", lty = 2, lwd = 2)
```



The Multiple Regression Model

The multiple regression model includes additional terms besides the single predictor in the linear regression model. As a simple example, consider the model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e \tag{2}$$

As we saw before, it is easy to fit this model using the `lm` function. Below, we fit the model predicting *Math* from *Strength* and *Grade*. Multiple R^2 is the square of the correlation between the predicted scores and the criterion. With only one predictor, it is equal to the squared correlation between the predictor and the criterion. Note that, with *Grade* in the equation, the R^2 value increased to .20, while coefficient for *Strength* is no longer significant. On the other hand, the coefficient for *Grade* is highly significant.

The Multiple Regression Model

How should we interpret these results?

```
> model.2 <- lm(Math ~ 1 + Strength + Grade)
> summary(model.2)


Call:
lm(formula = Math ~ 1 + Strength + Grade)

Residuals:
   Min     1Q Median     3Q    Max
-44.23 -10.16   0.26  10.00  37.25

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  91.2182     6.8697   13.28  < 2e-16 ***
Strength      0.0832     0.0680    1.22     0.22
Grade        13.0328     2.3296    5.59  7.3e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.7 on 197 degrees of freedom
Multiple R-squared:   0.2,Adjusted R-squared:  0.192
F-statistic: 24.6 on 2 and 197 DF,  p-value: 2.81e-10
```

The Multiple Regression Model

In this case, one of our predictors, *Strength*, is continuous, while the other, *Grade*, is categorical (binary) and is scored 0-1. This has an important implication. Because *Grade* is categorical 0-1, for 6th graders, the model becomes

$$Y = \beta_0 + \beta_1 Strength + e \tag{3}$$

For 8th graders, the equation becomes

$$Y = (\beta_0 + \beta_2) + \beta_1 Strength + e \tag{4}$$

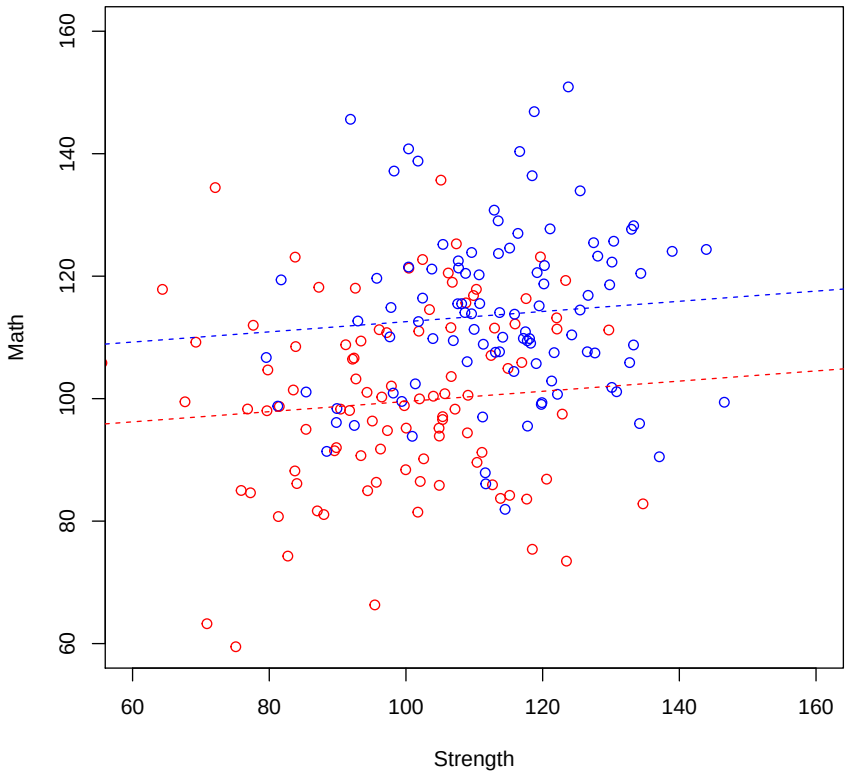
In other words, this model, in effect, simultaneously fits two regression models, with different intercepts but the same slope, to the *Strength-Math* data. The 6th graders have an intercept of β_0 and a slope of β_1, while the 8th graders have an intercept of $\beta_0 + \beta_2$, and a slope of β_1. So, a test that $\beta_2 = 0$ is also a test of equal intercepts (given equal slopes). Most textbooks begin the discussion of multiple regression with two continuous predictors. This example helps emphasize that multiple linear regression modeling offers "more than meets the eye" in analyzing data.

The Multiple Regression Model

Separate Intercepts, Same Slopes

Here is a picture of the data with the two separate regression lines

```
> plot(Strength[1:100], Math[1:100], col = "red", xlim = c(60, 160), ylim = c(60,
+     160), xlab = "Strength", ylab = "Math")
> points(Strength[101:200], Math[101:200], col = "blue")
> beta <- coef(model.2)
> abline(beta[1], beta[2], col = "red", lty = 2)
> abline(beta[1] + beta[3], beta[2], col = "blue", lty = 2)
```



The Multiple Regression Model

Notice that the preceding model assumed, implicitly, that there is no difference in the slopes of the regression lines for 6th and 8th graders. Can we fit a model that allows different slopes *and* different intercepts for the two grades?

The Multiple Regression Model with Interactions

Suppose we fit the following model to our *Strength-Math* data:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + e \tag{5}$$

For the special case where X_2 is a binary variable coded 0-1, 6th graders have $X_2 = 0$, and so the model becomes

$$Y = \beta_0 + \beta_1 X_1 + e \tag{6}$$

For 8th graders with $X_2 = 1$, we get

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 + \beta_3 X_1 + e \\ &= (\beta_0 + \beta_2) + (\beta_1 + \beta_3) X_1 + e \end{aligned}$$

Note that this is a model that specifies different slopes and intercepts for 6th and 8th graders. The 6th graders have a slope of β_1 and and intercept of β_0, while the 8th graders have a slope of $\beta_1 + \beta_3$ and an intercept of $\beta_0 + \beta_2$. A test that $\beta_2 = 0$ corresponds to a test of equal intercepts, while a test that $\beta_3 = 0$ corresponds to a test of equal slopes.

The Multiple Regression Model with Interactions

Here is how you specify this model in R. Scanning the results, notice now that the only significant term is the intercept. It now appears that the slope is not significantly different from zero for 6th graders, nor is there a significant difference in the slope between 6th and 8th graders. Note that the sizeable difference in intercepts between the grades is no longer statistically significant.

```
> model.3 <- lm(Math ~ Strength + Grade + Strength:Grade)
> summary(model.3)


Call:
lm(formula = Math ~ Strength + Grade + Strength:Grade)

Residuals:
   Min     1Q Median     3Q    Max
-44.14  -9.93   0.26  10.20  37.70

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     89.5608     9.5556    9.37   <2e-16 ***
Strength         0.1000     0.0957    1.04     0.30
Grade           16.6696    14.7243    1.13     0.26
Strength:Grade  -0.0341     0.1364   -0.25     0.80
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.7 on 196 degrees of freedom
Multiple R-squared:   0.2,Adjusted R-squared:  0.188
F-statistic: 16.4 on 3 and 196 DF,  p-value: 1.56e-09
```

The Multiple Regression Model

Separate Slopes and Intercepts

Here is a picture of the data with the two separate regression lines

```
> plot(Strength[1:100], Math[1:100], col = "red", xlim = c(60, 160), ylim = c(60,
+     160), xlab = "Strength", ylab = "Math")
> points(Strength[101:200], Math[101:200], col = "blue")
> beta <- coef(model.3)
> abline(beta[1], beta[2], col = "red", lty = 2)
> abline(beta[1] + beta[3], beta[2] + beta[4], col = "blue", lty = 2)
```