Linear models

Modeling. . . linearly!

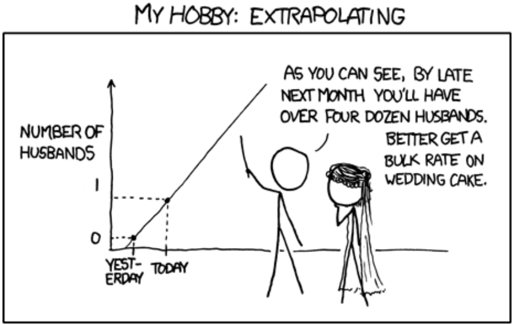
Samuel Robinson, Ph.D.

Sep. 22, 2023

Part 1: How do they work?

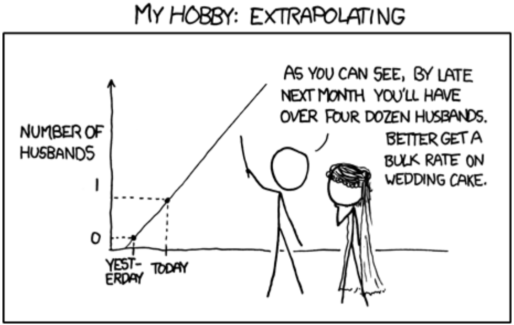
Outline

- What are linear models? How do I fit them?



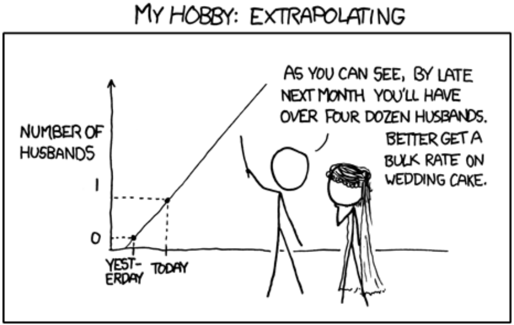
Outline

- What are linear models? How do I fit them?
- Making sure the model is working properly



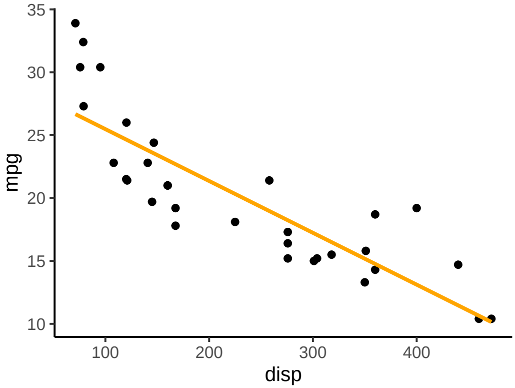
Outline

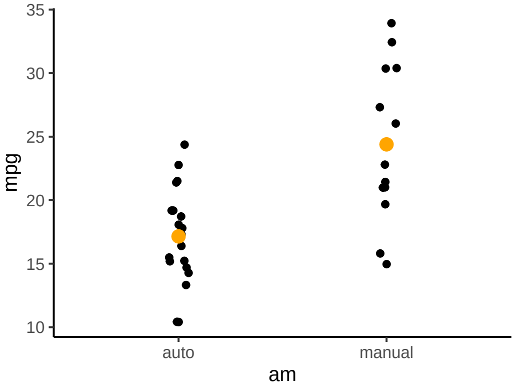
- What are linear models? How do I fit them?
- Making sure the model is working properly
- Plotting and interpreting model results



Motivation

- *I measured 2 things and I want to know if they're related to each other*
- *I have groups of data, and I want to know whether the means are different*





Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)
- Multiple linear model/regression

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)
- Multiple linear model/regression
- Analysis of Variance (ANOVA)

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)
- Multiple linear model/regression
- Analysis of Variance (ANOVA)
- Analysis of Covariance (ANCOVA)

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)
- Multiple linear model/regression
- Analysis of Variance (ANOVA)
- Analysis of Covariance (ANCOVA)

Terminology

Linear models go by many different names. All these models are all doing *exactly the same thing*:

- Linear regression
- Least-squares regression
- Simple linear model (SLM)
- Multiple linear model/regression
- Analysis of Variance (ANOVA)
- Analysis of Covariance (ANCOVA)

I use a set of terminology that I find very helpful, from Berliner (1996). I'll be using it here, as well as for describing more complex models.

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i
- b_0 is the *intercept*, a coefficient that doesn't depend on predictors

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i
- b_0 is the *intercept*, a coefficient that doesn't depend on predictors
- $y \sim Normal(\hat{y}, \sigma)$ means:

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i
- b_0 is the *intercept*, a coefficient that doesn't depend on predictors
- $y \sim Normal(\hat{y}, \sigma)$ means:
  - "y follows a Normal distribution with mean $\hat{y}$ and SD σ"

Model terminology

All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i
- b_0 is the *intercept*, a coefficient that doesn't depend on predictors
- $y \sim Normal(\hat{y}, \sigma)$ means:
    - "y follows a Normal distribution with mean $\hat{y}$ and SD σ"

Model terminology

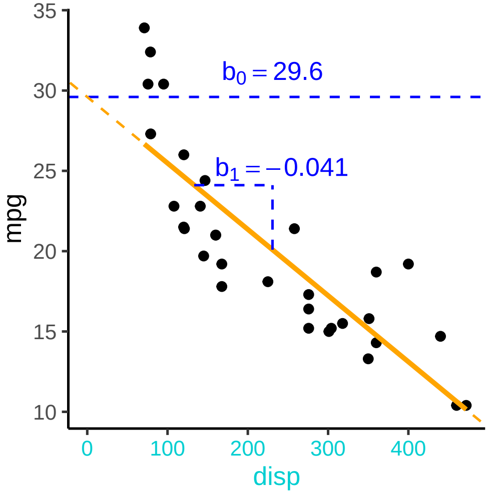
All linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- y is the thing you're interested in predicting
- $\hat{y}$ is the *predicted value* of y
- $x_1 ... x_i$ are *predictors* of y
- $b_1 ... b_i$ are *coefficients* for each predictor x_i
- b_0 is the *intercept*, a coefficient that doesn't depend on predictors
- $y \sim Normal(\hat{y}, \sigma)$ means:
    - "y follows a Normal distribution with mean $\hat{y}$ and SD σ"

This may look terrifying, but let's use a simple example:

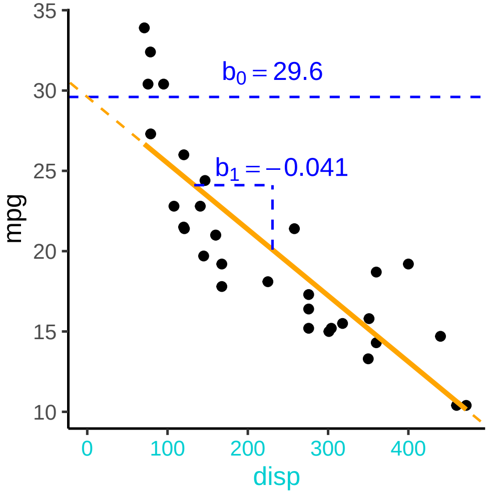
Example



- *mpg* is the thing you're interested in predicting

$$\hat{mpg} = b_0 + b_1 \, disp$$

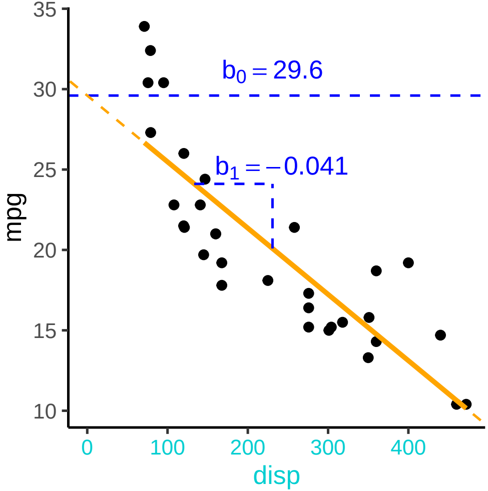
Example



$$\hat{mpg} = b_0 + b_1 disp$$

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*

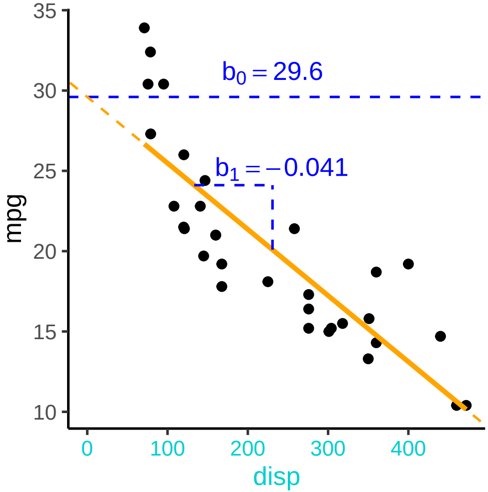
Example



- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *disp* is the *predictor* of *mpg*

$$\hat{mpg} = b_0 + b_1 disp$$

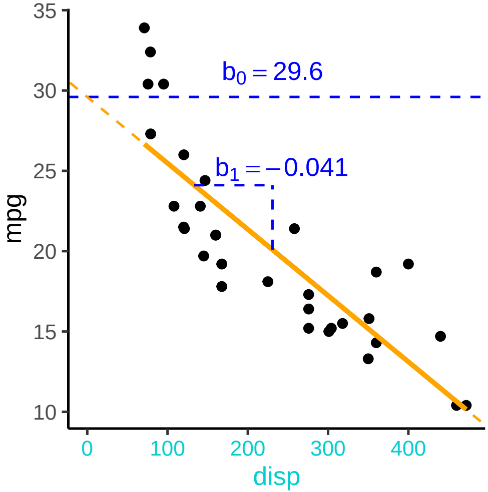
Example



- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *disp* is the *predictor* of *mpg*
- b_0 is the *intercept*, b_1 is the *coefficient* (slope) for *disp*

$$\hat{mpg} = b_0 + b_1 disp$$

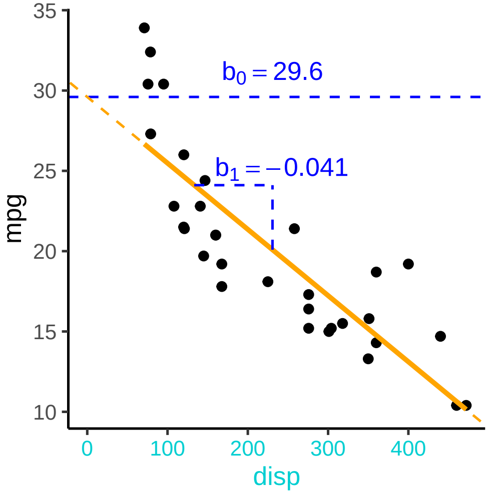
Example



$$\hat{mpg} = b_0 + b_1 disp$$

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *disp* is the *predictor* of *mpg*
- b_0 is the *intercept*, b_1 is the *coefficient* (slope) for *disp*
- $mpg \sim Normal(\hat{mpg}, \sigma)$ means:

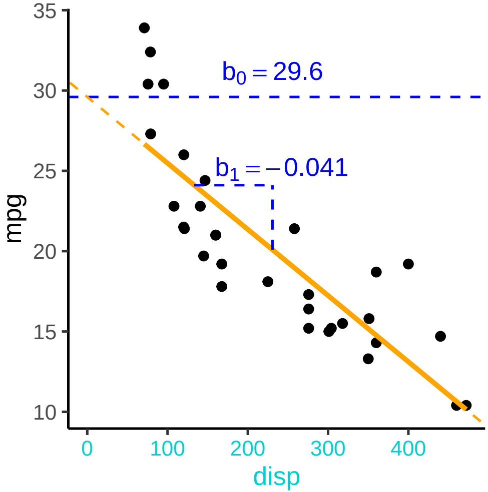
Example



- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *disp* is the *predictor* of *mpg*
- b_0 is the *intercept*, b_1 is the *coefficient* (slope) for *disp*
- $mpg \sim Normal(\hat{mpg}, \sigma)$ means:
  - "*mpg* follows a Normal distribution with mean $\hat{mpg}$ and SD σ"

$$\hat{mpg} = b_0 + b_1 disp$$

Example

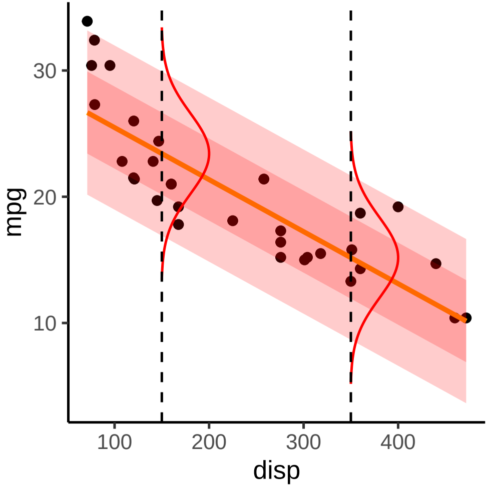


$$\hat{mpg} = b_0 + b_1 disp$$

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *disp* is the *predictor* of *mpg*
- b_0 is the *intercept*, b_1 is the *coefficient* (slope) for *disp*
- $mpg \sim Normal(\hat{mpg}, \sigma)$ means:
  - "*mpg* follows a Normal distribution with mean $\hat{mpg}$ and SD σ"
- σ isn't displayed on the figure. Where is it?

Example (cont.)

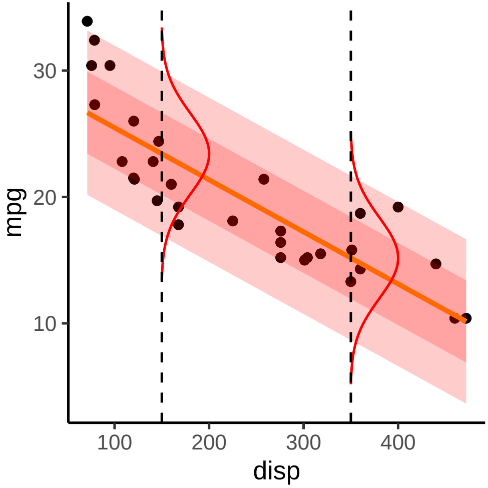
σ isn't displayed on the figure. Where is it?



- σ is the "leftover" or "residual" variance

Example (cont.)

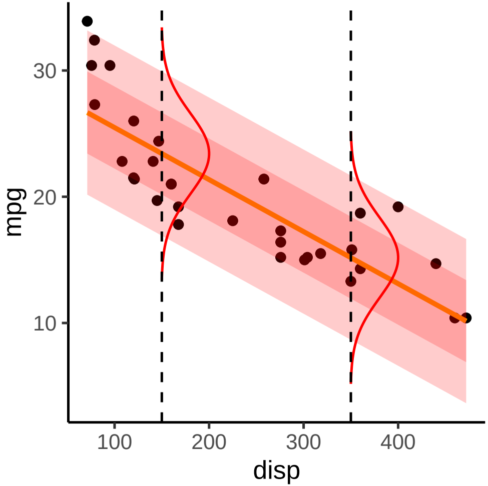
σ isn't displayed on the figure. Where is it?



- σ is the "leftover" or "residual" variance
- i.e. variation between samples that the model couldn't explain

Example (cont.)

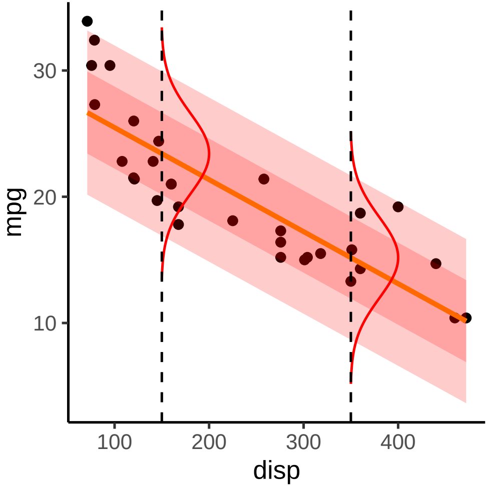
σ isn't displayed on the figure. Where is it?



- σ is the "leftover" or "residual" variance
- i.e. variation between samples that the model couldn't explain
- Since $y \sim \mathit{Normal}(\hat{y}, \sigma)$, this means that points are normally distributed around the *entire line* of $\hat{y}$

Example (cont.)

σ isn't displayed on the figure. Where is it?



- σ is the "leftover" or "residual" variance
- i.e. variation between samples that the model couldn't explain
- Since $y \sim \mathit{Normal}(\hat{y}, \sigma)$, this means that points are normally distributed around the *entire line* of $\hat{y}$
- If you took a vertical slice at each part of the x-axis, the distribution would be *Normal*

How do I get R to fit this model?

`lm` is one of the main functions used for linear modeling:

```r
#Formula= y ~ x, data = Name of the dataframe containing mpg & disp
mod1 <- lm(mpg ~ disp, data = mtcars); summary(mod1)
```

```
##
## Call:
## lm(formula = mpg ~ disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8922 -2.2022 -0.9631  1.6272  7.2305
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.599855   1.229720  24.070  < 2e-16 ***
## disp        -0.041215   0.004712  -8.747 9.38e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.251 on 30 degrees of freedom
## Multiple R-squared:  0.7183, Adjusted R-squared:  0.709
## F-statistic: 76.51 on 1 and 30 DF,  p-value: 9.38e-10
```

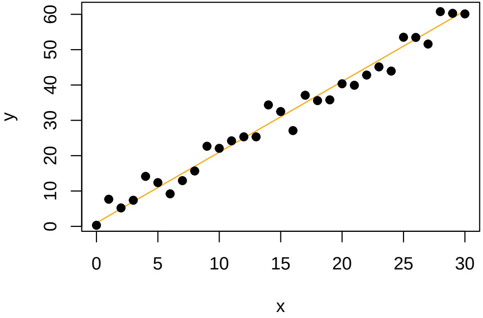
For a detailed breakdown of `lm`'s output, click here

Simulate data

Now that we know how linear models work, we can simulate our own data:

```r
#Parameters:

b0 <- 1 #Intercept
b1 <- 2 #Slope
sigma <- 3 #SD

#Make up some data:

x <- 0:30 #Predictor values

#Predicted y values
pred_y <- b0 + b1*x

#Add "noise" around pred_y
actual_y <- rnorm(n = length(pred_y),
                  mean = pred_y,
                  sd= sigma)
```

```r
#Plot the data we just made
plot(x,pred_y,col='orange',pch=19,type='l',
     ylab='y')
points(x,actual_y,col='black',pch=19)
```



Fit a model from simulated data

How does R do at finding the coefficients? Remember: $b_0 = 1, b_1 = 2, \sigma = 3$

```r
fakeDat <- data.frame(x = x, y = actual_y, pred = pred_y) #Simulated data in a dataframe
mod1sim <- lm(y ~ x, data = fakeDat); summary(mod1sim) #Fit model
```

```
##
## Call:
## lm(formula = y ~ x, data = fakeDat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7568 -1.7623 -0.2176  1.9419  5.3572
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.02974    1.00445   2.021   0.0526 .
## x            1.92670    0.05751  33.499   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.864 on 29 degrees of freedom
## Multiple R-squared:  0.9748, Adjusted R-squared:  0.9739
## F-statistic:  1122 on 1 and 29 DF,  p-value: < 2.2e-16
```

Fit a model from simulated data

How does R do at finding the coefficients? Remember: $b_0 = 1, b_1 = 2, \sigma = 3$

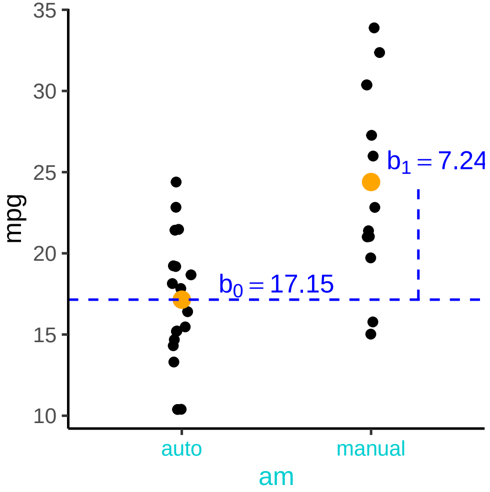
```r
fakeDat <- data.frame(x = x, y = actual_y, pred = pred_y) #Simulated data in a dataframe
mod1sim <- lm(y ~ x, data = fakeDat); summary(mod1sim) #Fit model
```

```
##
## Call:
## lm(formula = y ~ x, data = fakeDat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7568 -1.7623 -0.2176  1.9419  5.3572
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.02974    1.00445   2.021   0.0526 .
## x            1.92670    0.05751  33.499   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.864 on 29 degrees of freedom
## Multiple R-squared:  0.9748, Adjusted R-squared:  0.9739
## F-statistic:  1122 on 1 and 29 DF,  p-value: < 2.2e-16
```

Modeling philosophy: all models are approximating a **generative process**.

It is up to us to think about what this process might be like.

What about categorical data?

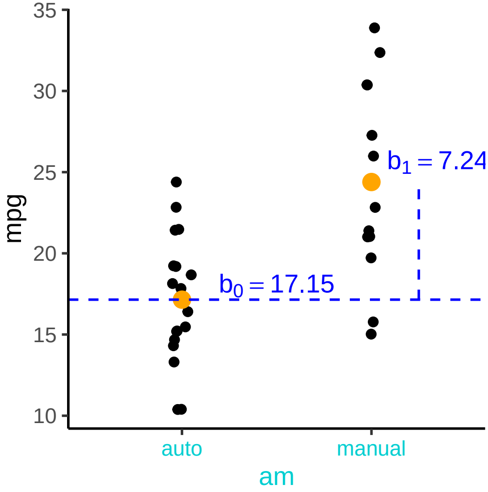


$$\hat{mpg} = b_0 + b_1 am$$

This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting

What about categorical data?

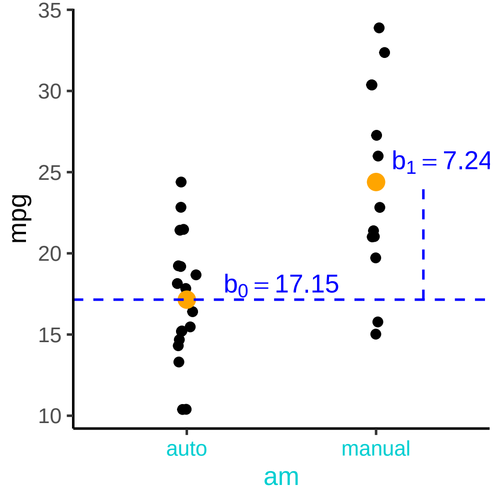


$$\hat{mpg} = b_0 + b_1 am$$

This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*

What about categorical data?

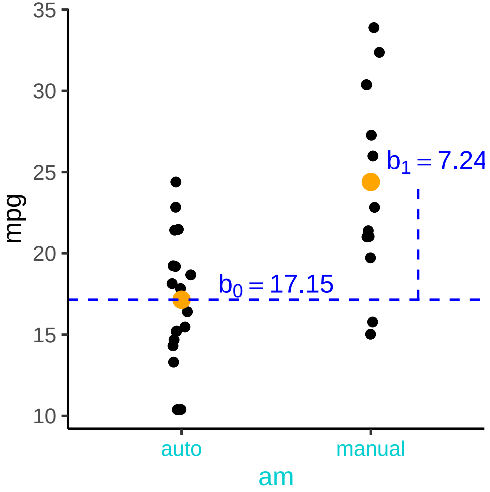


$$\hat{mpg} = b_0 + b_1 am$$

This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *am* is the *predictor* of *mpg*

What about categorical data?

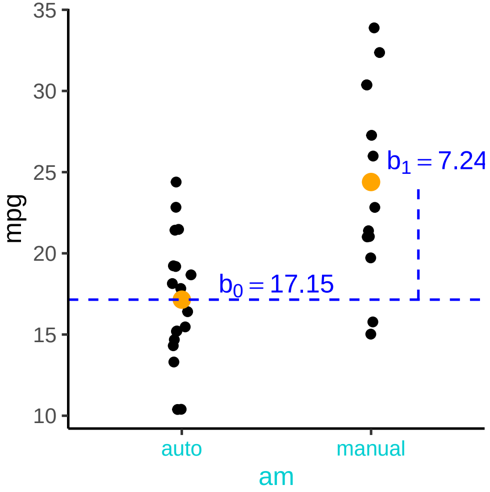


This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *am* is the *predictor* of *mpg*
  - set of 0s and 1s, not continuous

$$\hat{mpg} = b_0 + b_1 am$$

What about categorical data?

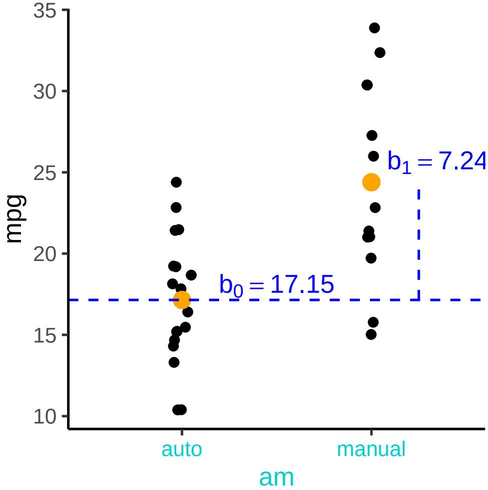


$$\hat{mpg} = b_0 + b_1 am$$

This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *am* is the *predictor* of *mpg*
  - set of 0s and 1s, not continuous
- b_0 is the *intercept*, b_1 is the *coefficient* for *am*

What about categorical data?



$$\hat{mpg} = b_0 + b_1 am$$

This uses *exactly the same* math!

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *am* is the *predictor* of *mpg*
  - set of 0s and 1s, not continuous
- b_0 is the *intercept*, b_1 is the *coefficient* for *am*
- Where is σ?

How do I get R to fit this model?

Syntax is exactly the same for this model

```r
#Formula structure: y ~ x
mod2 <- lm(mpg ~ am, #mpg depends on am
           data = mtcars) #Name of the dataframe containing mpg & am
summary(mod2)
```

```
##
## Call:
## lm(formula = mpg ~ am, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3923 -3.0923 -0.2974  3.2439  9.5077
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## am             7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)
    - e.g. `rep(x=c(0,1),each=10)`

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)
    - e.g. `rep(x=c(0,1),each=10)`
  - Useful command: `rnorm` (generate normally-distributed data)

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)
    - e.g. `rep(x=c(0,1),each=10)`
  - Useful command: `rnorm` (generate normally-distributed data)
    - e.g. `rnorm(n=100,mean=0,sd=1)`

A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)
    - e.g. `rep(x=c(0,1),each=10)`
  - Useful command: `rnorm` (generate normally-distributed data)
    - e.g. `rnorm(n=100,mean=0,sd=1)`
- Use `lm` to fit a model to the data you just simulated

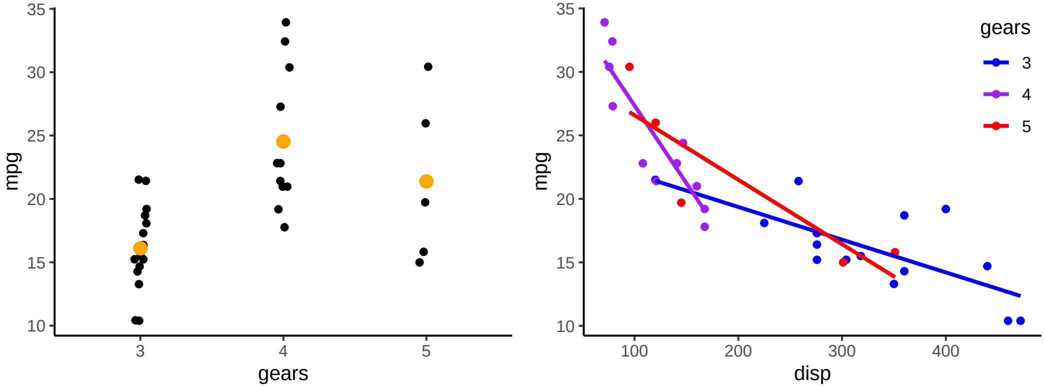
A challenger approaches!

- Simulate your own data with 2 discrete levels. My suggestion:
  - ~~Steal~~Borrow my code, and change the predictor from continuous to discrete
  - Useful command: `rep` (replicate)
    - e.g. `rep(x=c(0,1),each=10)`
  - Useful command: `rnorm` (generate normally-distributed data)
    - e.g. `rnorm(n=100,mean=0,sd=1)`
- Use `lm` to fit a model to the data you just simulated
  - How does R do at guessing your coefficients?

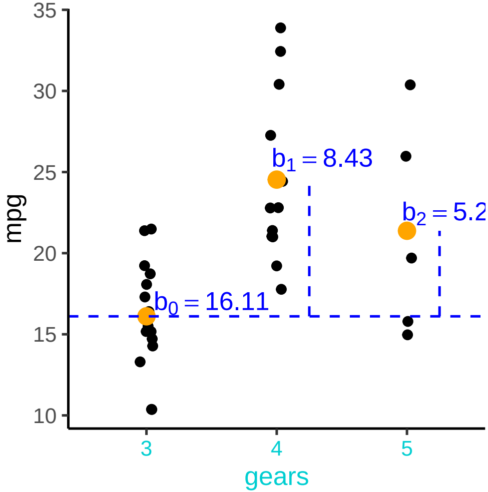
Part 2: More bells and whistles

Motivation

- *I have 2+ groups of data, and I want to know whether the means are different*
- *I have 2+ groups of bivariate data, and I want to know whether the relationships differ between groups*



Categorial data, 3 categories



The more factor levels, the more coefficients:

- *mpg* is the thing you're interested in predicting
- $\hat{mpg}$ is the *predicted value* of *mpg*
- *gear* is the *predictor* of *mpg*
- set of 0s and 1s
- $gears_4$ = "is this data point from a 4-gear car?"
- b_0 = *intercept*
- $[b_1, b_2]$ = are *coefficients* for *gears*

$$\hat{mpg} = b_0 + b_1 gears_4 + b_2 gears_5$$

How do I get R to fit this model?

```r
#Formula structure: y ~ x
mod1 <- lm(mpg ~ factor(gear), #mpg depends on gears
           data = mtcars) #Name of the dataframe containing mpg & gears
summary(mod1)
```

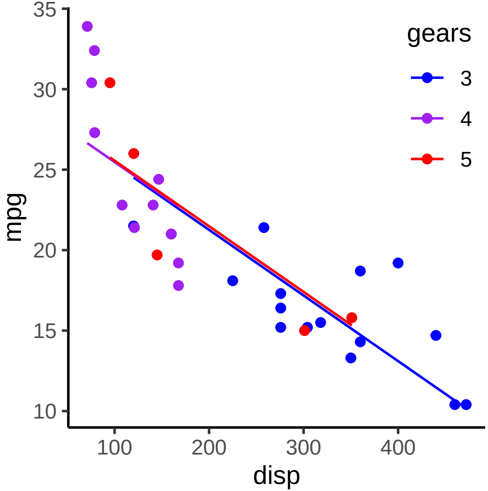
```
##
## Call:
## lm(formula = mpg ~ factor(gear), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7333 -3.2333 -0.9067  2.8483  9.3667
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     16.107      1.216  13.250 7.87e-14 ***
## factor(gear)4    8.427      1.823   4.621 7.26e-05 ***
## factor(gear)5    5.273      2.431   2.169   0.0384 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.708 on 29 degrees of freedom
## Multiple R-squared:  0.4292, Adjusted R-squared:  0.3898
## F-statistic:  10.9 on 2 and 29 DF,  p-value: 0.0002948
```

Dummy variables

```r
mod1Matrix <- model.matrix(mod1) #Get model matrix (columns used to predict mpg)
head(mod1Matrix,28) #Show first 28 rows of model matrix
```

```
##                     (Intercept) factor(gear)4 factor(gear)5
## Mazda RX4                     1             1             0
## Mazda RX4 Wag                 1             1             0
## Datsun 710                    1             1             0
## Hornet 4 Drive                1             0             0
## Hornet Sportabout             1             0             0
## Valiant                       1             0             0
## Duster 360                    1             0             0
## Merc 240D                     1             1             0
## Merc 230                      1             1             0
## Merc 280                      1             1             0
## Merc 280C                     1             1             0
## Merc 450SE                    1             0             0
## Merc 450SL                    1             0             0
## Merc 450SLC                   1             0             0
## Cadillac Fleetwood            1             0             0
## Lincoln Continental           1             0             0
## Chrysler Imperial             1             0             0
## Fiat 128                      1             1             0
## Honda Civic                   1             1             0
## Toyota Corolla                1             1             0
## Toyota Corona                 1             0             0
## Dodge Challenger              1             0             0
## AMC Javelin                   1             0             0
## Camaro Z28                    1             0             0
## Pontiac Firebird              1             0             0
## Fiat X1-9                     1             1             0
## Porsche 914-2                 1             0             1
## Lotus Europa                  1             0             1
```

What about if 2 things are both important?



- Suppose that both *disp* and *gears* are important for predicting *mpg*?
- This is very similar to the last example, except that now we've added *disp*
- *gears* now changes the intercept, while *disp* changes the slope of all the lines
- Does it look like *gear* is very important?

$$\hat{mpg} = b_0 + b_1 disp$$

How do I get R to fit this model?

```r
#mpg depends on disp and gears
mod2 <- lm(mpg ~ disp+factor(gear), data = mtcars)
summary(mod2)
```

```
##
## Call:
## lm(formula = mpg ~ disp + factor(gear), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9155 -2.1892 -0.9054  1.5790  7.2498
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   29.411183   2.627966  11.192 7.58e-12 ***
## disp          -0.040774   0.007601  -5.364 1.03e-05 ***
## factor(gear)4  0.138017   2.021332   0.068    0.946
## factor(gear)5  0.224712   1.976090   0.114    0.910
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.365 on 28 degrees of freedom
## Multiple R-squared:  0.7185, Adjusted R-squared:  0.6883
## F-statistic: 23.82 on 3 and 28 DF,  p-value: 7.31e-08
```

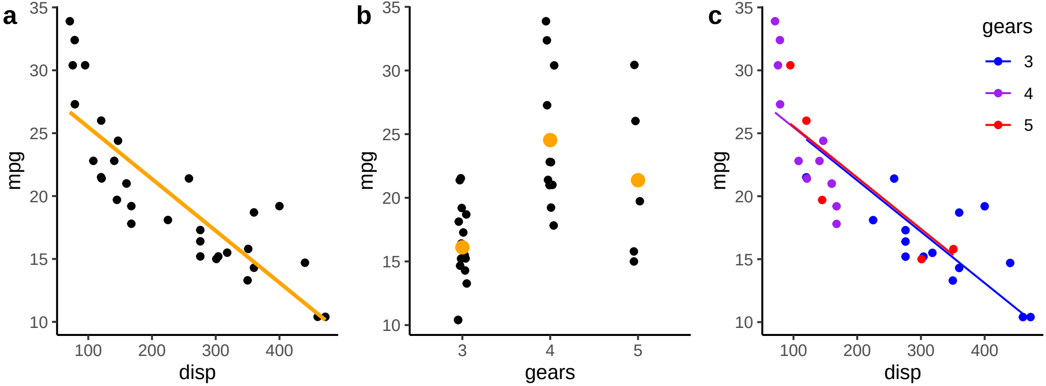
Dummy variables

```
mod2Matrix <- model.matrix(mod2) #Get model matrix (columns used to predict mpg)
colnames(mod2Matrix) <- gsub('factor\\(gear\\)','gear',colnames(mod2Matrix)) #Shorten colnames
head(mod2Matrix,28) #Show first 28 rows of model matrix
```

```
##                     (Intercept)  disp gear4 gear5
## Mazda RX4                     1 160.0     1     0
## Mazda RX4 Wag                 1 160.0     1     0
## Datsun 710                    1 108.0     1     0
## Hornet 4 Drive                1 258.0     0     0
## Hornet Sportabout             1 360.0     0     0
## Valiant                       1 225.0     0     0
## Duster 360                    1 360.0     0     0
## Merc 240D                     1 146.7     1     0
## Merc 230                      1 140.8     1     0
## Merc 280                      1 167.6     1     0
## Merc 280C                     1 167.6     1     0
## Merc 450SE                    1 275.8     0     0
## Merc 450SL                    1 275.8     0     0
## Merc 450SLC                   1 275.8     0     0
## Cadillac Fleetwood            1 472.0     0     0
## Lincoln Continental           1 460.0     0     0
## Chrysler Imperial             1 440.0     0     0
## Fiat 128                      1  78.7     1     0
## Honda Civic                   1  75.7     1     0
## Toyota Corolla                1  71.1     1     0
## Toyota Corona                 1 120.1     0     0
## Dodge Challenger              1 318.0     0     0
## AMC Javelin                   1 304.0     0     0
## Camaro Z28                    1 350.0     0     0
## Pontiac Firebird              1 400.0     0     0
## Fiat X1-9                     1  79.0     1     0
## Porsche 914-2                 1 120.3     0     1
```

Interlude: problems with plotting raw data

- Say that I've fit the following model:
  `mpg ~ disp + factor(gear)`
- All of the plots below are using raw data, but which one is "telling the truth"?
- Answer: **c**. *a* and *b* are hiding the effect of the other variable



How do I plot these model results?

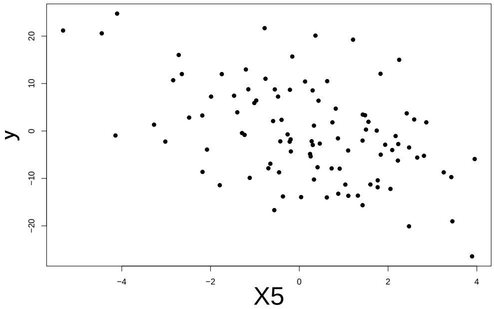
Rule for plotting model results:

1. If the model uses N variables, you should show all N effects *simultaneously*

Other names for partial effects:

Incorrect example, using raw data:

```
#Fit model with 5 variables (all important)
simMod <- lm(y~X1+X2+X3+X4+X5,data=pred)
#Incorrect way, using raw data
plot(y~X5,data=pred,pch=19,cex.lab=3)
```



The effect of *X5* is actually **very** strong ($p < 0.0001$), but it doesn't look like it from this plot!

How do I plot these model results?

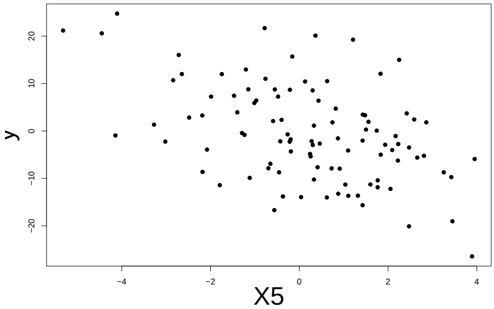
Rule for plotting model results:

1. If the model uses `N` variables, you should show all `N` effects *simultaneously*
2. If this is impractical, you should use a **partial effects plot**

Other names for partial effects:

Incorrect example, using raw data:

```
#Fit model with 5 variables (all important)
simMod <- lm(y~X1+X2+X3+X4+X5,data=pred)
#Incorrect way, using raw data
plot(y~X5,data=pred,pch=19,cex.lab=3)
```



The effect of *X5* is actually **very** strong ($p < 0.0001$), but it doesn't look like it from this plot!

How do I plot these model results?

Rule for plotting model results:

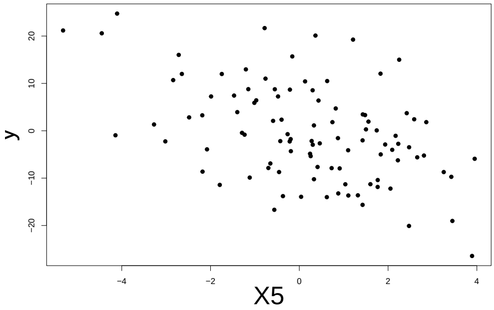
1. If the model uses `N` variables, you should show all `N` effects *simultaneously*
2. If this is impractical, you should use a **partial effects plot**

Other names for partial effects:

- *counterfactual* plot, *predictor effect* plot, *leverage* plot

Incorrect example, using raw data:

```
#Fit model with 5 variables (all important)
simMod <- lm(y~X1+X2+X3+X4+X5,data=pred)
#Incorrect way, using raw data
plot(y~X5,data=pred,pch=19,cex.lab=3)
```



The effect of $X5$ is actually **very** strong ($p < 0.0001$), but it doesn't look like it from this plot!

How do I plot these model results?

Rule for plotting model results:

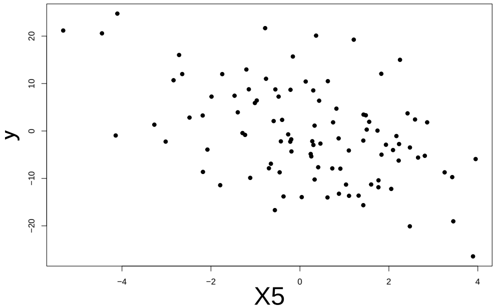
1. If the model uses `N` variables, you should show all `N` effects *simultaneously*
2. If this is impractical, you should use a **partial effects plot**

Other names for partial effects:

- *counterfactual* plot, *predictor effect* plot, *leverage* plot
- Try using `effects` or `ggeffects`. Requires the `effects` and `ggeffect` packages

Incorrect example, using raw data:

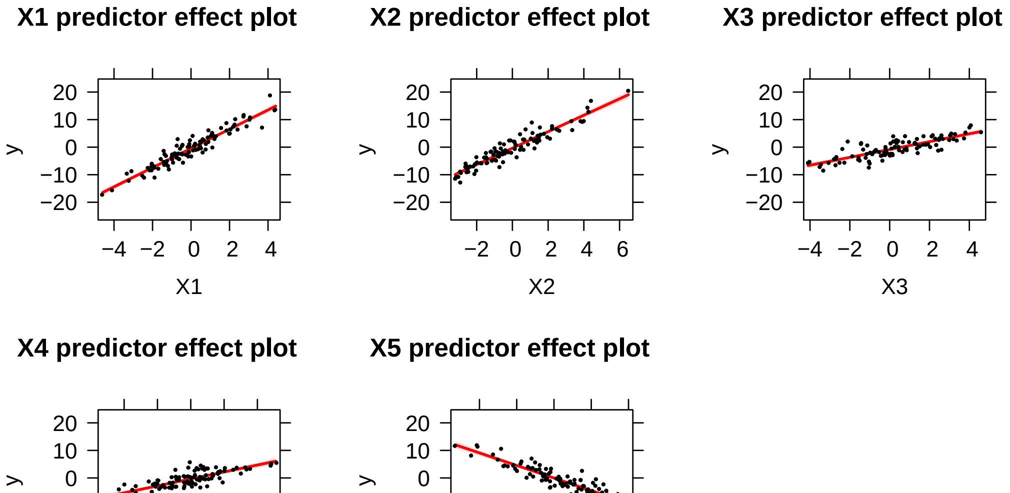
```
#Fit model with 5 variables (all important)
simMod <- lm(y~X1+X2+X3+X4+X5,data=pred)
#Incorrect way, using raw data
plot(y~X5,data=pred,pch=19,cex.lab=3)
```



The effect of *X5* is actually **very** strong ($p < 0.0001$), but it doesn't look like it from this plot!

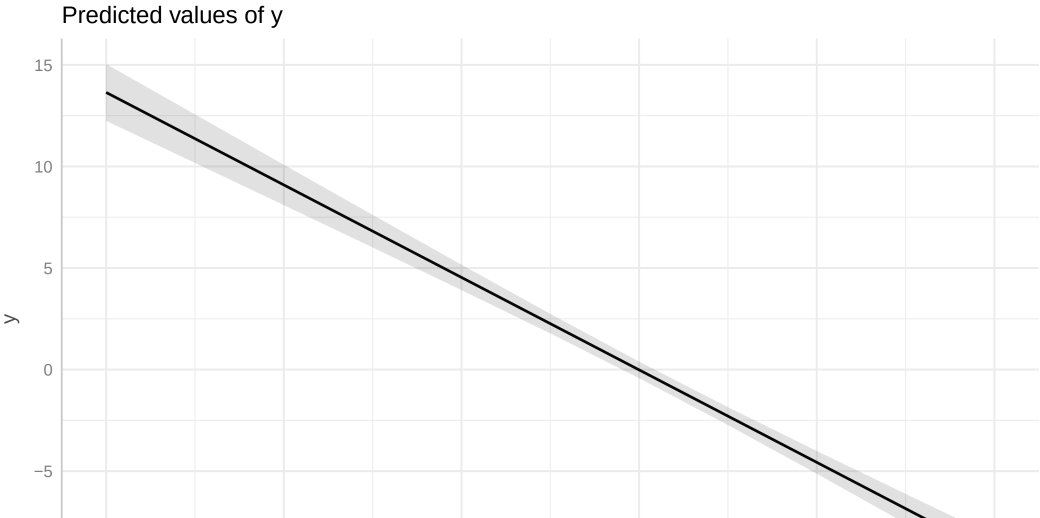
Partial effects plots - using *effects*

```
library(effects) #Load effects package
simModEff <- predictorEffects(simMod,partial.residuals=TRUE) #Calculate partial effects
#Plot partial effects
plot(simModEff,lines=list(col='red'), partial.residuals=list(pch=19,col='black',cex=0.25))
```



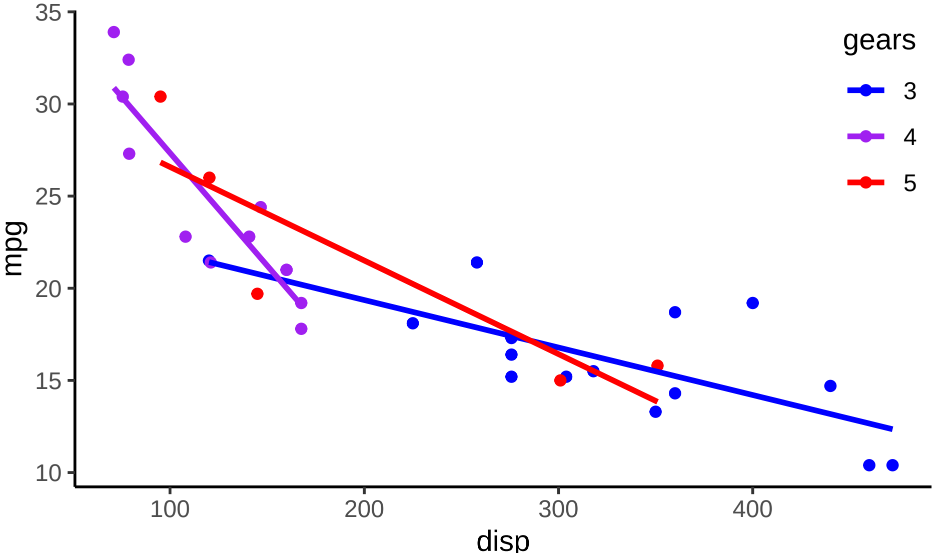
Partial effects plots - using *ggpredict*

```
library(ggeffects) #Load ggeffects package
simModEff2 <- ggeffect(simMod,terms=c('X5')) #Calculate partial effects for X5
plot(simModEff2) #Plot effect of X5
```

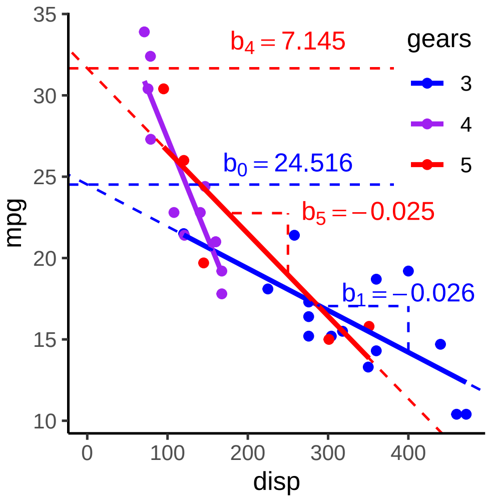


Interactions

What if the slopes *and* intercepts differ between groups?



Interactions



$$
\begin{aligned}
\hat{mpg} &= b_0 + b_1 disp \\
&\quad + b_2 gears_4 + b_3 gears_5 \\
&\quad + b_4 (disp \times gears_4) \\
&\quad + b_5 (disp \times gears_5) \\
mpg &\sim Normal(\hat{mpg}, \sigma)
\end{aligned}
$$

- Interactions occur when predictors are *multiplied*
- In this case, *disp* is multiplied by $gears_4$ and $gears_5$
- *gears* now changes the intercept and the slope of the relationship between *mpg* and *disp*

How do I get R to fit this model?

```r
#mpg depends on disp interacted (*) with gears
mod2 <- lm(mpg ~ disp*factor(gear), data = mtcars)
summary(mod2)
```

```
##
## Call:
## lm(formula = mpg ~ disp * factor(gear), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5986 -1.5990 -0.0143  1.6329  4.9926
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        24.515566   2.462431   9.956 2.32e-10 ***
## disp               -0.025770   0.007265  -3.547 0.001505 **
## factor(gear)4      15.051963   3.558043   4.230 0.000256 ***
## factor(gear)5       7.145380   3.535913   2.021 0.053711 .
## disp:factor(gear)4 -0.096442   0.021261  -4.536 0.000114 ***
## disp:factor(gear)5 -0.025005   0.013320  -1.877 0.071742 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.579 on 26 degrees of freedom
## Multiple R-squared:  0.8465, Adjusted R-squared:  0.817
## F-statistic: 28.67 on 5 and 26 DF,  p-value: 8.452e-10
```

Beware of fitting too many interactions, or else the *Bilbo effect* occurs!

Dummy variables

```r
mod2Matrix <- model.matrix(mod2) #Get model matrix (columns used to predict mpg)
colnames(mod2Matrix) <- gsub('factor\\(gear\\)','gear',colnames(mod2Matrix)) #Shorten colnames
head(mod2Matrix,28) #Show first 28 rows of model matrix
```

```
##                     (Intercept)  disp gear4 gear5 disp:gear4 disp:gear5
## Mazda RX4                     1 160.0     1     0      160.0        0.0
## Mazda RX4 Wag                 1 160.0     1     0      160.0        0.0
## Datsun 710                    1 108.0     1     0      108.0        0.0
## Hornet 4 Drive                1 258.0     0     0        0.0        0.0
## Hornet Sportabout             1 360.0     0     0        0.0        0.0
## Valiant                       1 225.0     0     0        0.0        0.0
## Duster 360                    1 360.0     0     0        0.0        0.0
## Merc 240D                     1 146.7     1     0      146.7        0.0
## Merc 230                      1 140.8     1     0      140.8        0.0
## Merc 280                      1 167.6     1     0      167.6        0.0
## Merc 280C                     1 167.6     1     0      167.6        0.0
## Merc 450SE                    1 275.8     0     0        0.0        0.0
## Merc 450SL                    1 275.8     0     0        0.0        0.0
## Merc 450SLC                   1 275.8     0     0        0.0        0.0
## Cadillac Fleetwood            1 472.0     0     0        0.0        0.0
## Lincoln Continental           1 460.0     0     0        0.0        0.0
## Chrysler Imperial             1 440.0     0     0        0.0        0.0
## Fiat 128                      1  78.7     1     0       78.7        0.0
## Honda Civic                   1  75.7     1     0       75.7        0.0
## Toyota Corolla                1  71.1     1     0       71.1        0.0
## Toyota Corona                 1 120.1     0     0        0.0        0.0
## Dodge Challenger              1 318.0     0     0        0.0        0.0
## AMC Javelin                   1 304.0     0     0        0.0        0.0
## Camaro Z28                    1 350.0     0     0        0.0        0.0
## Pontiac Firebird              1 400.0     0     0        0.0        0.0
## Fiat X1-9                     1  79.0     1     0       79.0        0.0
## Porsche 914-2                 1 120.3     0     1        0.0      120.3
```

A challenger approaches!

- Since you're all bat folks, here's some bat data!

A challenger approaches!

- Since you're all bat folks, here's some bat data!
    - `batDat.csv`

A challenger approaches!

- Since you're all bat folks, here's some bat data!
    - `batDat.csv`
- Data: 100 bat weights from 2 cities, recorded along with sex and age

A challenger approaches!

- Since you're all bat folks, here's some bat data!
  - `batDat.csv`
- Data: 100 bat weights from 2 cities, recorded along with sex and age
- How do these variables affect bat weight?

A challenger approaches!

- Since you're all bat folks, here's some bat data!
  - `batDat.csv`
- Data: 100 bat weights from 2 cities, recorded along with sex and age
- How do these variables affect bat weight?
  - Think about how these variables might be related to weight using your `brain`

A challenger approaches!

- Since you're all bat folks, here's some bat data!
  - `batDat.csv`
- Data: 100 bat weights from 2 cities, recorded along with sex and age
- How do these variables affect bat weight?
  - Think about how these variables might be related to weight using your `brain`
  - Fit a model using `lm`

A challenger approaches!

- Since you're all bat folks, here's some bat data!
  - `batDat.csv`
- Data: 100 bat weights from 2 cities, recorded along with sex and age
- How do these variables affect bat weight?
  - Think about how these variables might be related to weight using your `brain`
  - Fit a model using `lm`
  - Make some plots, using `effects` or `ggeffects`

Part 3: Models behaving badly

Motivation

Are my model results reliable?

- Residual checks

Motivation

Are my model results reliable?

- Residual checks
- Transformations

Motivation

Are my model results reliable?

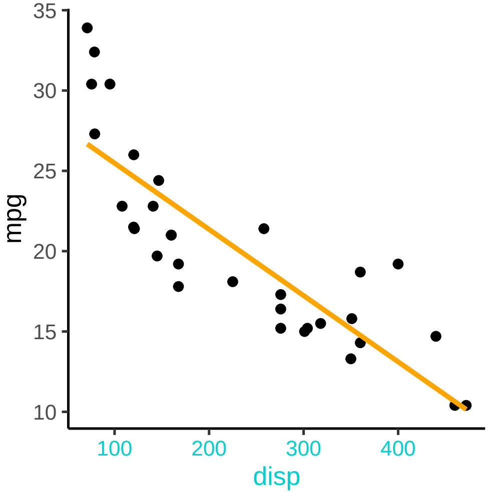
- Residual checks
- Transformations
- Collinearity

Motivation

Are my model results reliable?

- Residual checks
- Transformations
- Collinearity
- How much stuff should I put into my model?

Assumptions of linear regression



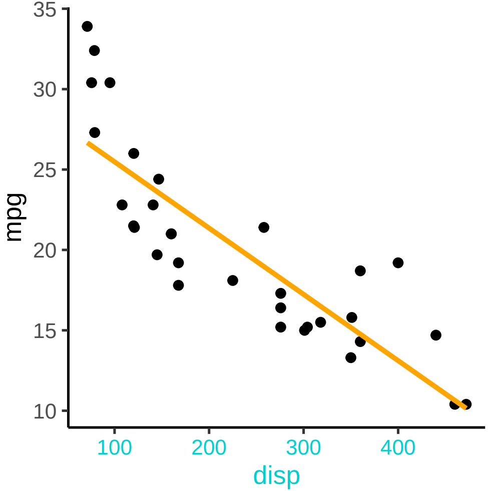
$$\hat{mpg} = b_0 + b_1 disp$$

There are 3 main assumptions to this model:

1. The relationship between *disp* and *mpg* is linear

This is pretty easy to see if you only have 1 variable, but...

Assumptions of linear regression



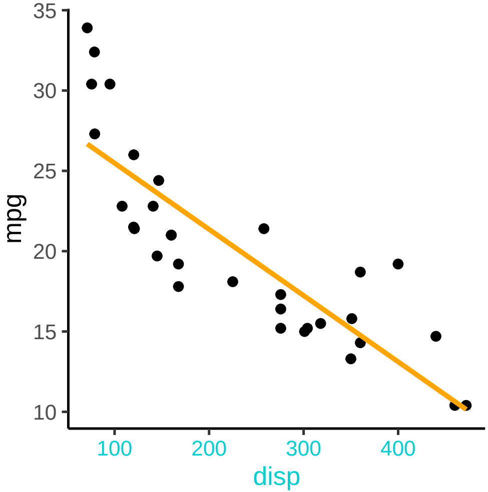
$\hat{mpg} = b_0 + b_1 disp$

There are 3 main assumptions to this model:

1. The relationship between *disp* and *mpg* is linear
2. *mpg* (the data) is Normally distributed around $\hat{mpg}$ (the line)

This is pretty easy to see if you only have 1 variable, but. . .

Assumptions of linear regression



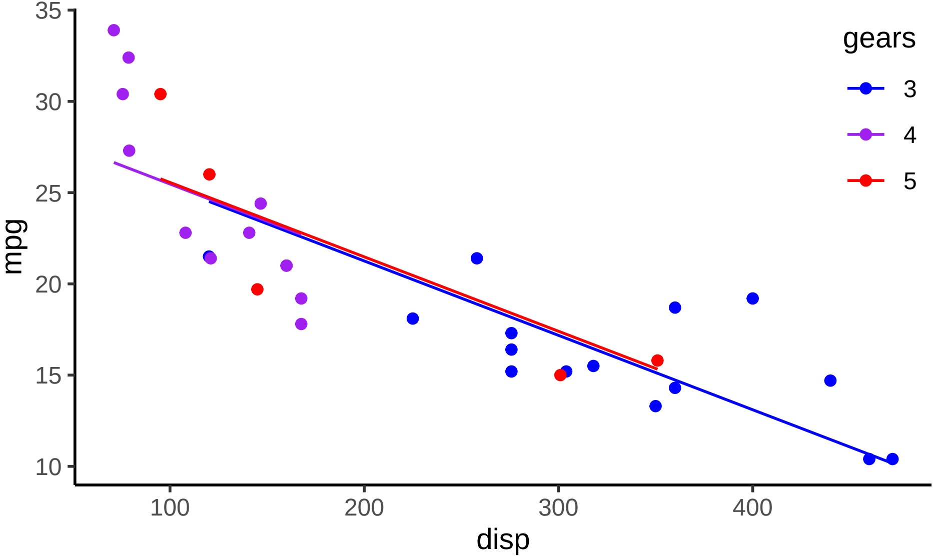
$$\hat{mpg} = b_0 + b_1 disp$$

There are 3 main assumptions to this model:

1. The relationship between *disp* and *mpg* is linear
2. *mpg* (the data) is Normally distributed around $\hat{mpg}$ (the line)
3. σ is the same everywhere

This is pretty easy to see if you only have 1 variable, but...

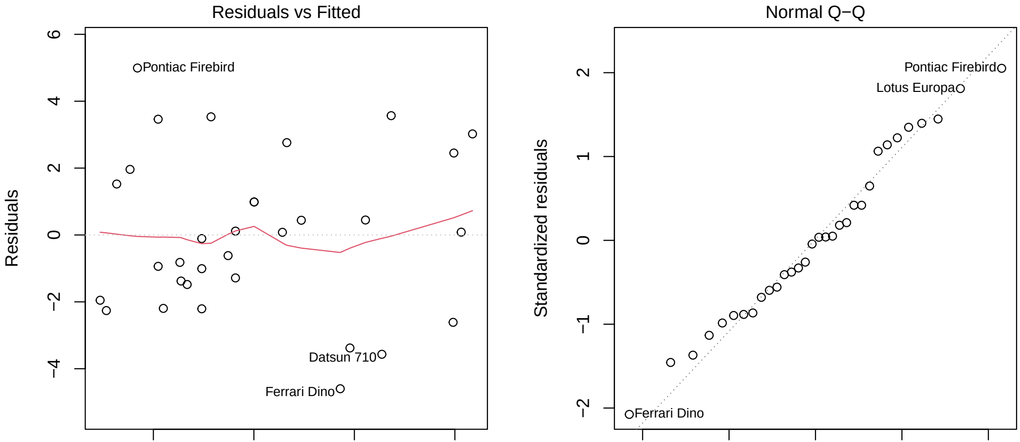
What if I have many variables?



Solution: residual checks

Some common ways of checking the assumptions: **residual plots**

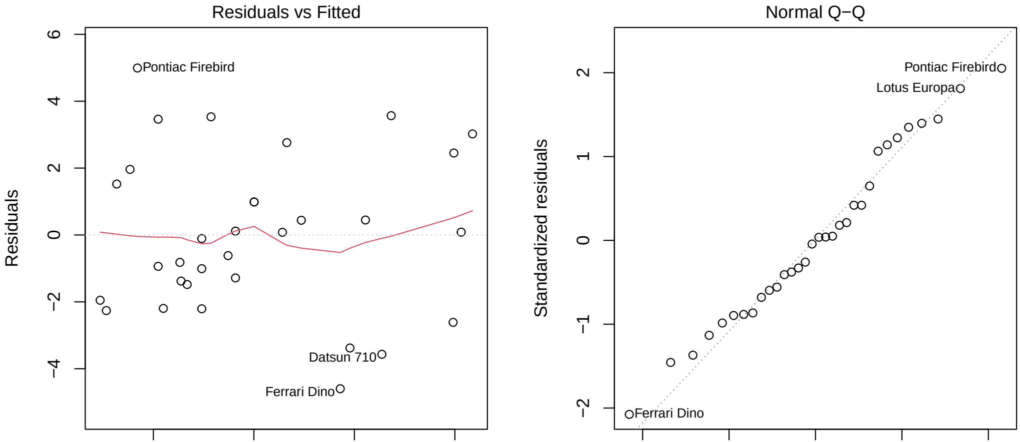
```
mod1 <- lm(mpg~disp*factor(gear),data=mtcars) #Fits model
par(mfrow=c(1,2),mar=c(3,3,1,1)+1) #Splits plot into 2
plot(mod1, which=c(1,2)) #1st and 2nd residual plots
```



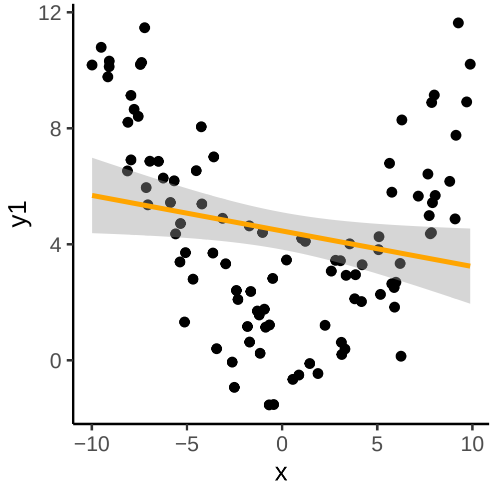
Solution: residual checks

Some common ways of checking the assumptions: **residual plots**

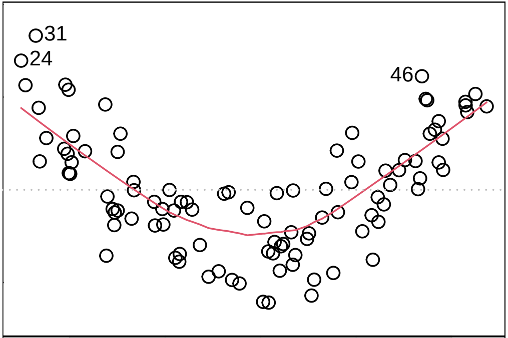
```
mod1 <- lm(mpg~disp*factor(gear),data=mtcars) #Fits model
par(mfrow=c(1,2),mar=c(3,3,1,1)+1) #Splits plot into 2
plot(mod1, which=c(1,2)) #1st and 2nd residual plots
```

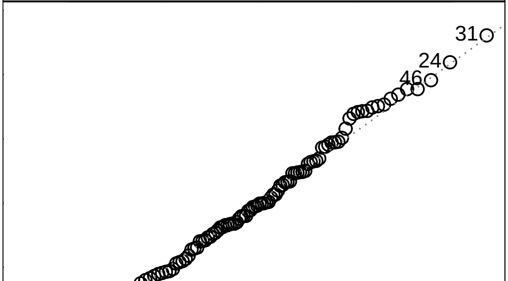


Problem 1: Non-linear relationship



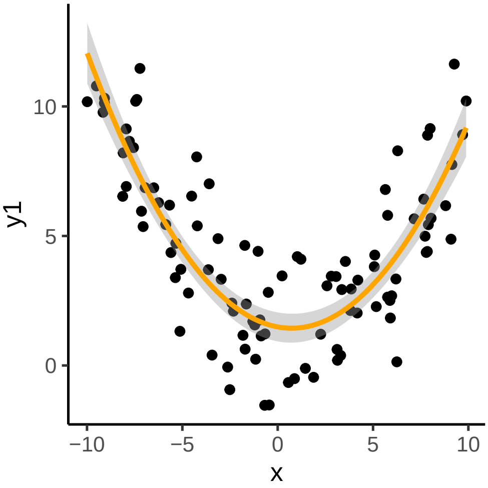

```r
lm(y1~x,data=d1)
```





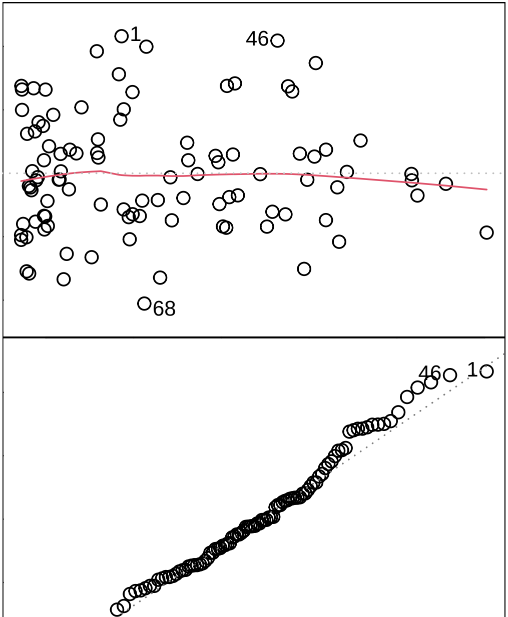
$y1$ clearly follows a hump-shaped relationship, not a linear one

Solution: transform predictors

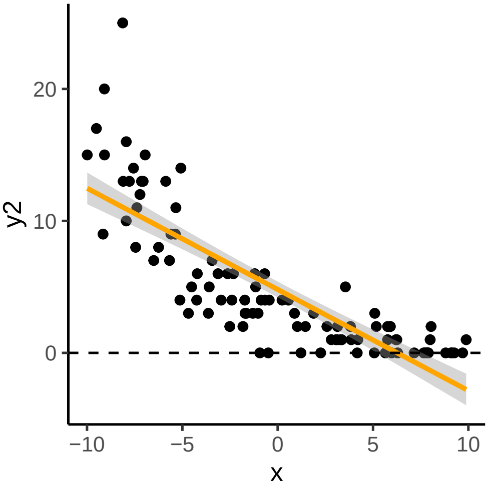



```
lm(y1~poly(x,2),data=d1)
```

log and *square-root* transformations are common

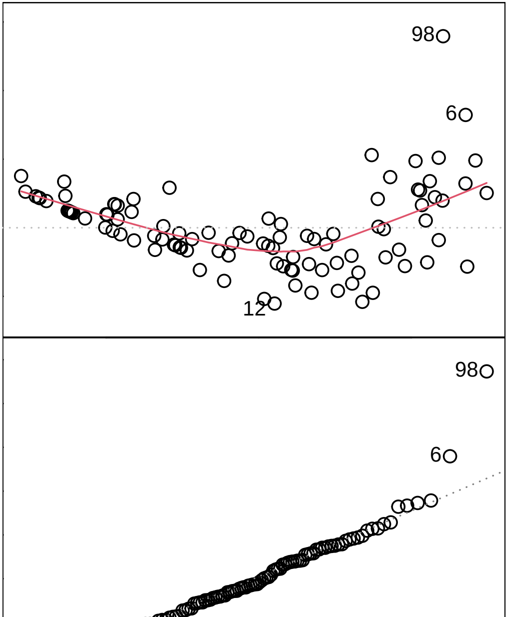


Problem 2a: Non-normal response

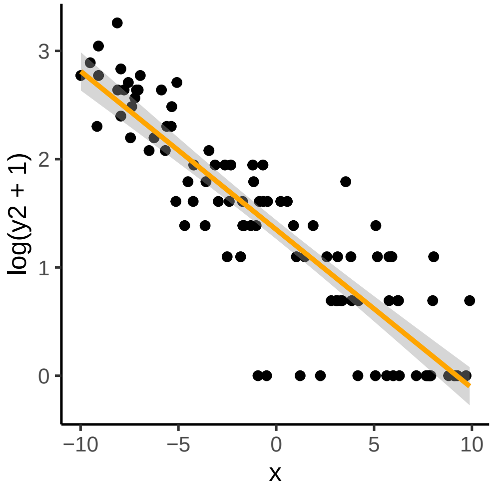


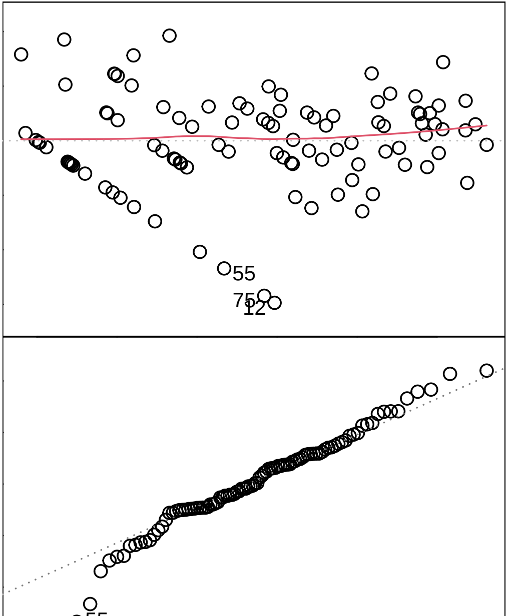

```r
lm(y2~x,data=d1)
```

$y2$ is count data (integers ≥ 0). *Very* common in ecological data



Solution: transform data to meet assumptions

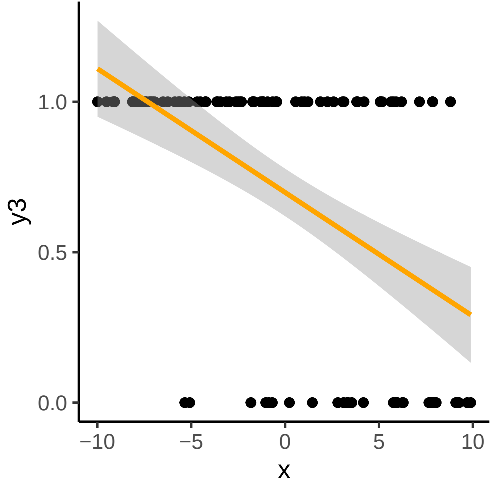





```
lm(log(y2+1)~x,data=d1)
```

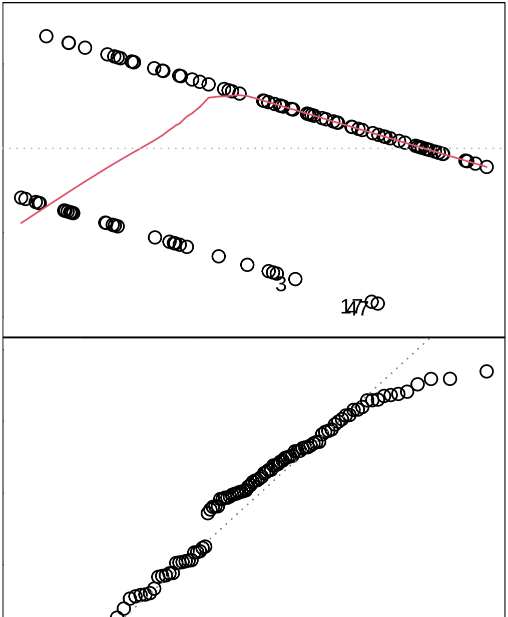
Square-root transformations are also common

Problem 2b: Non-normal response

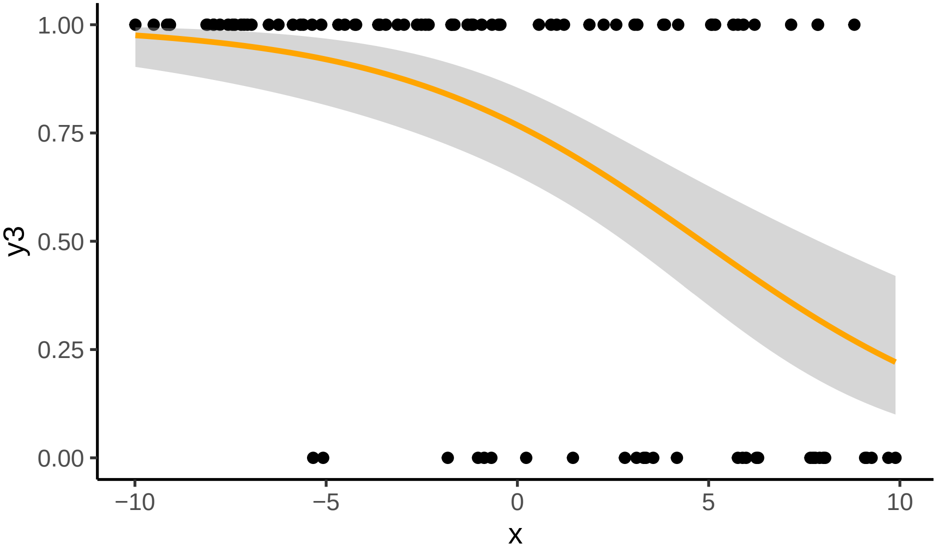



```r
lm(y3~x,data=d1)
```

y3 is binomial data (success/failure, 0 or 1). *Very common in ecological data*

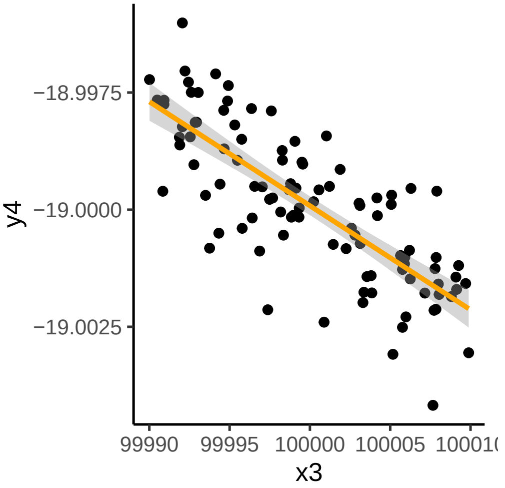


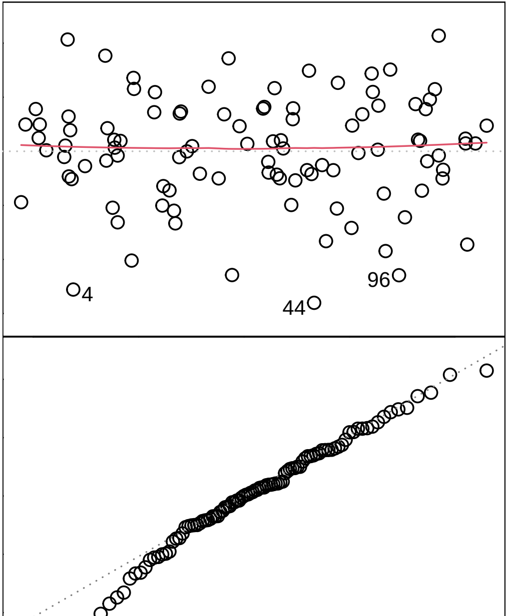
Solution: use a Generalized Linear Model (GLM)



- This is a topic for another lecture. Hold tight!

Problem: variables are on different scales

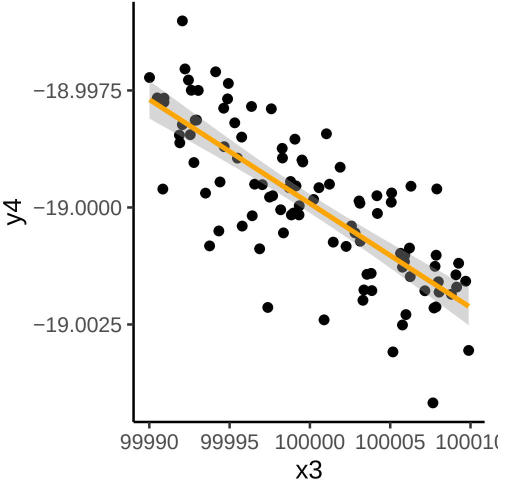


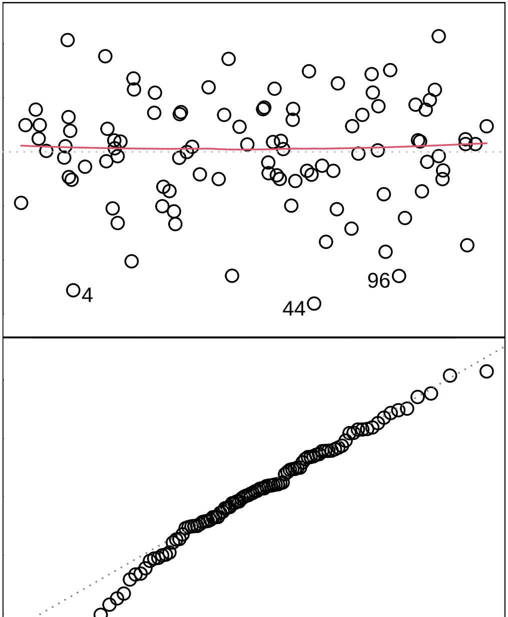



```r
lm(y4~x3,data=d1)
```

- $y4$ is tiny, while $x3$ is huge

Problem: variables are on different scales

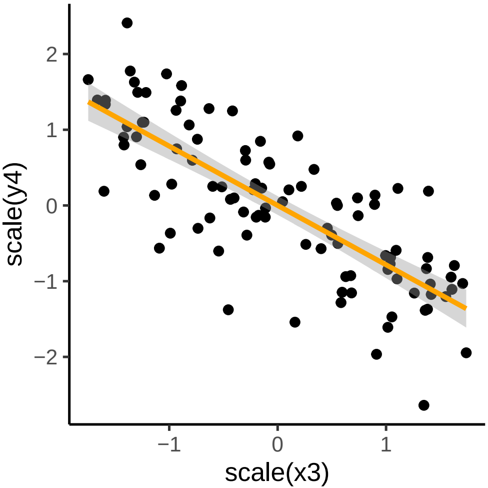


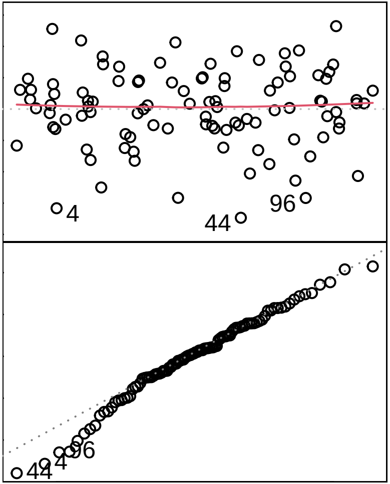



```r
lm(y4~x3,data=d1)
```

- $y4$ is tiny, while $x3$ is huge

Solution: scale data/predictors before fitting

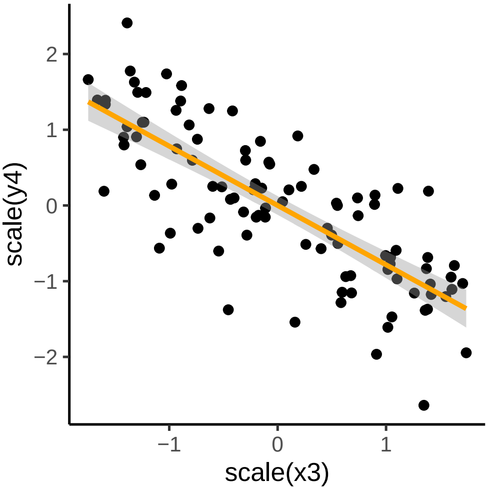


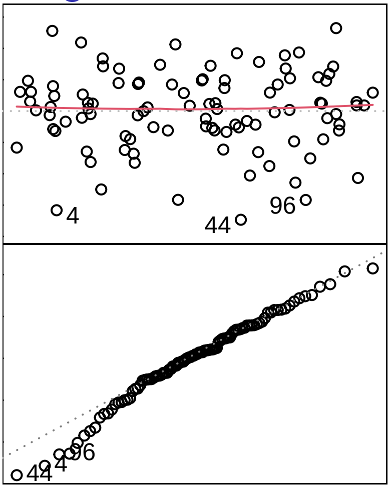



```
#Subtracts mean, divides by SD
d1$s.y4 <- scale(y4)
d1$s.x3 <- scale(x3)
lm(s.y4~s.x3,data=d1) #Refit
```

- Residuals are the same as before

Solution: scale data/predictors before fitting






```r
#Subtracts mean, divides by SD
d1$s.y4 <- scale(y4)
d1$s.x3 <- scale(x3)
lm(s.y4~s.x3,data=d1) #Refit
```

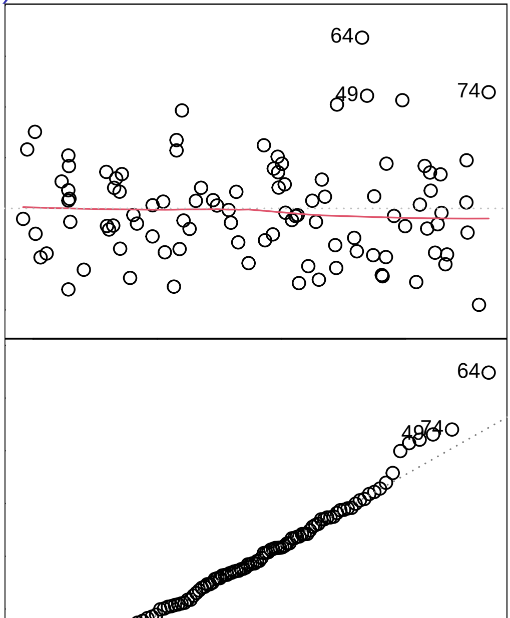
- Residuals are the same as before
- Coefficients are now related to *scaled* data and predictor

But wait... there's more (assumptions)!

One more assumption:

4. If you have 2+ predictors in your model, the predictors are not related to each other

```
lm(y0~x+x2,data=d1)
```



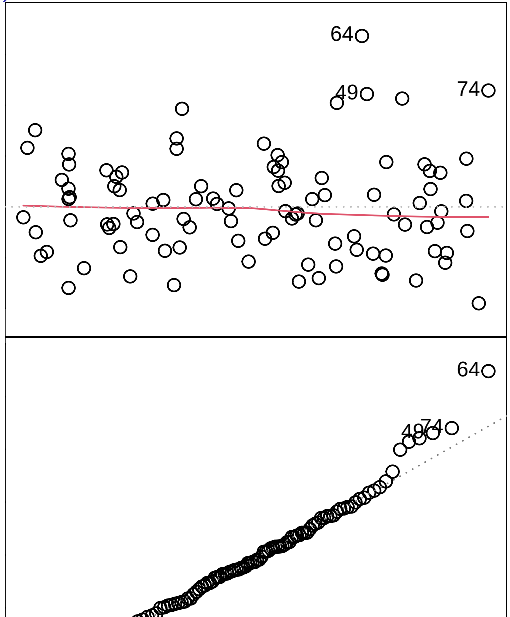
But wait... there's more (assumptions)!

One more assumption:

4. If you have 2+ predictors in your model, the predictors are not related to each other

- Say we have 2 predictors, x and $x2$:

```
lm(y0~x+x2,data=d1)
```



But wait… there's more (assumptions)!

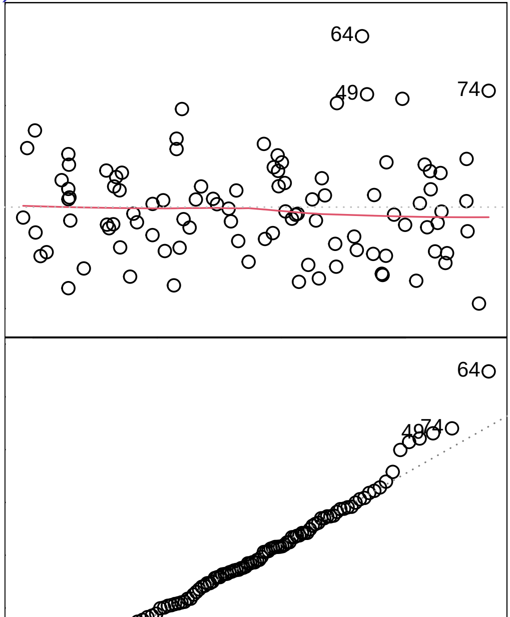
One more assumption:

4. If you have 2+ predictors in your model, the predictors are not related to each other

- Say we have 2 predictors, x and $x2$:

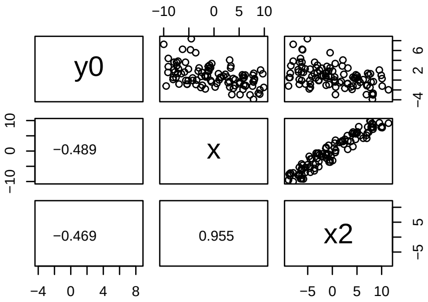
```
lm(y0~x+x2,data=d1)
```

- Model fits, and residuals look OK, but there's trouble ahead!



Uh oh! Collinearity!

```r
#Function to print correlation (r) value
corText <- function(x,y){
  text(0.5,0.5,round(cor(x,y),3))
}

#Pairplot of y0, x, and x2
pairs(d1[,c('y0','x','x2')],lower.panel=corText)
```



pairs() is useful for looking at relations

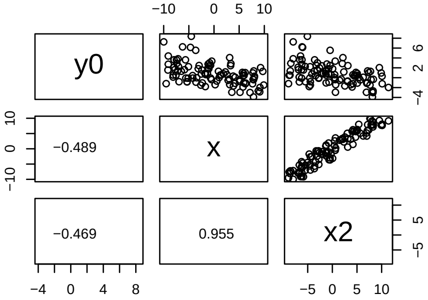
- x and $x2$ mean basically the same thing!

```r
library(car)
#VIF scores:
# 1 = no problem
# 1-5 = some problems
# 5+ = big problems!
vif(m2)
```

```
##        x       x2
## 11.31812 11.31812
```

Uh oh! Collinearity!

```r
#Function to print correlation (r) value
corText <- function(x,y){
  text(0.5,0.5,round(cor(x,y),3))
}

#Pairplot of y0, x, and x2
pairs(d1[,c('y0','x','x2')],lower.panel=corText)
```



- x and $x2$ mean basically the same thing!
- Also revealed using variance-inflation factors (VIFs):

```r
library(car)
#VIF scores:
# 1 = no problem
# 1-5 = some problems
# 5+ = big problems!
vif(m2)
```

```
##        x       x2
## 11.31812 11.31812
```

`pairs()` is useful for looking at relations

Is collinearity really that bad?

```r
#Correct model
m1 <- lm(y0~x,data=d1)
```

	Estimate	Std. Error	Pr(>|t|)
(Intercept)	0.7851936	0.1943002	0.0001059
x	-0.1900346	0.0342596	0.0000002

```r
#Incorrect model
m2 <- lm(y0~x+x2,data=d1)
```

	Estimate	Std. Error	Pr(>|t|)
(Intercept)	0.7860300	0.1955770	0.0001155
x	-0.1812556	0.1158464	0.1209288
x2	-0.0094931	0.1196074	0.9369028

- Increases SE of each term, so model may “miss” important terms

Is collinearity really that bad?

```r
#Correct model
m1 <- lm(y0~x,data=d1)
```

	Estimate	Std. Error	Pr(>|t|)
(Intercept)	0.7851936	0.1943002	0.0001059
x	-0.1900346	0.0342596	0.0000002

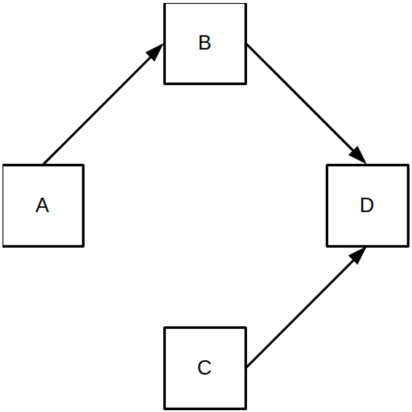
```r
#Incorrect model
m2 <- lm(y0~x+x2,data=d1)
```

	Estimate	Std. Error	Pr(>|t|)
(Intercept)	0.7860300	0.1955770	0.0001155
x	-0.1812556	0.1158464	0.1209288
x2	-0.0094931	0.1196074	0.9369028

- Increases SE of each term, so model may “miss” important terms
- Gets worse with increasing correlation, or if many terms are correlated!

How do we fix this? Depends on your goals:

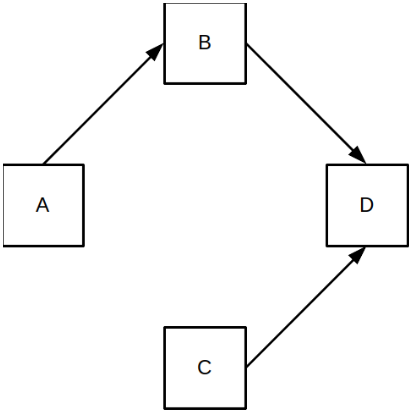
1. I care about predicting things




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

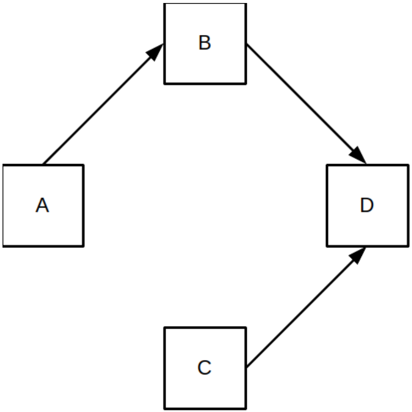
1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

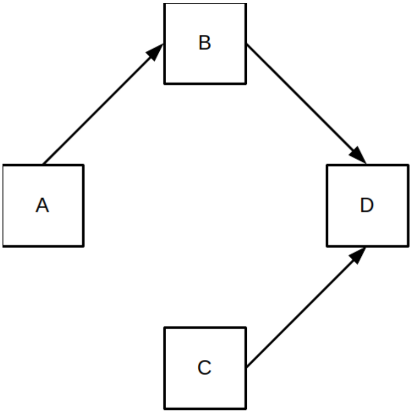
1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

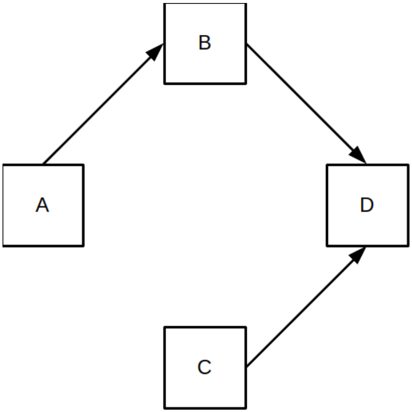
1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things
   - Design experiment to separate cause and effect




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

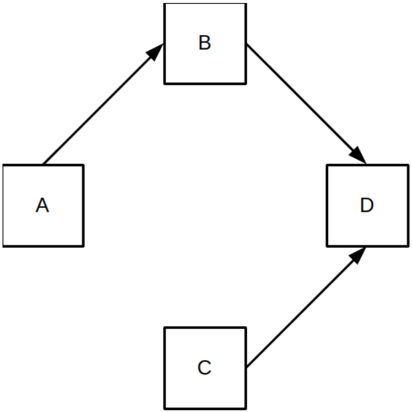
1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things
   - Design experiment to separate cause and effect
   - Think about what is causing what. *Graphical models* are helpful for this




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

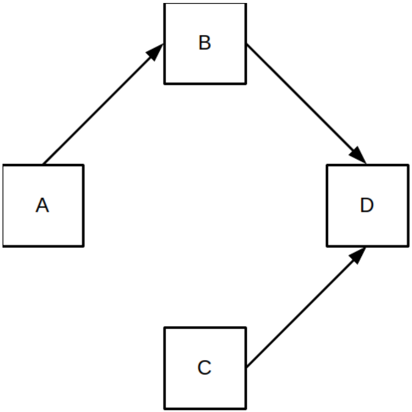
1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things
   - Design experiment to separate cause and effect
   - Think about what is causing what. *Graphical models* are helpful for this
     - Not all variables have to be included!




```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things
   - Design experiment to separate cause and effect
   - Think about what is causing what. *Graphical models* are helpful for this
     - Not all variables have to be included!

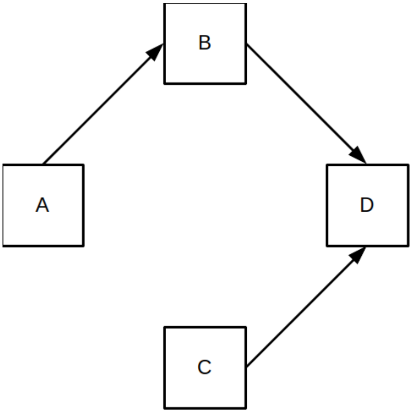


- Simple graphical model, where the effect of A on D is *mediated* by B.

```
lm(D ~ B + C)
```

How do we fix this? Depends on your goals:

1. I care about predicting things
   - Use dimensional reduction (e.g. PCA) and re-run model
2. I care about what's causing things
   - Design experiment to separate cause and effect
   - Think about what is causing what. *Graphical models* are helpful for this
     - Not all variables have to be included!



- Simple graphical model, where the effect of A on D is *mediated* by B.
- "Correct" `lm` model of D:

```
lm(D ~ B + C)
```

A challenger approaches!

- Guess what... more bat data! This time there are 6 variables that were measured. We're interested in predicting `bats` (counts of bats per night).

A challenger approaches!

- Guess what... more bat data! This time there are 6 variables that were measured. We're interested in predicting `bats` (counts of bats per night).
- Formulate a causal model that seems reasonable

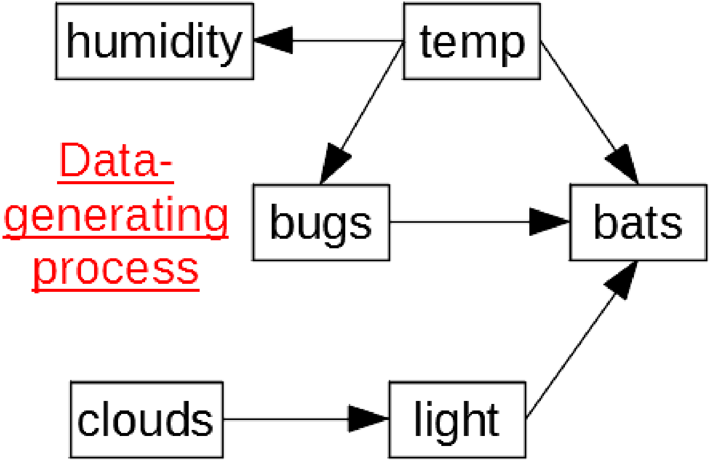
A challenger approaches!

- Guess what... more bat data! This time there are 6 variables that were measured. We're interested in predicting `bats` (counts of bats per night).
- Formulate a causal model that seems reasonable
  - Draw it out on paper/in PowerPoint using flow diagrams

A challenger approaches!

- Guess what... more bat data! This time there are 6 variables that were measured. We're interested in predicting `bats` (counts of bats per night).
- Formulate a causal model that seems reasonable
  - Draw it out on paper/in PowerPoint using flow diagrams
- Fit an `lm` model of `bats` from your causal model, check the assumptions, and update as necessary

Here's the answer



This is the **true** process that generated the data. Model for `bats` should look like:

```
lm(log(bats+0.1)~poly(temp,2)+light+bugs,data=dat)
```