Agenda for Week 3, Hr 2 (Tuesday, Jan 21)

Hour 2: lm (regression),

plot (scatterplots),

cooks.distance and resid (diagnostics)

The basic form of the regression formula is lm(y ~ x)

I'm referring to it as the basic form because we will be expanding upon it throughout the semester.

Let's call weight the response variable and length the explanatory variable.

```
lm(weight ~ size, data=Beard)
```

We could use Beard$weight ~ Beard$size , but for the sake of later uses, it's better to specify the dataset with data=

Unlike t.test() and cor.test(), making a linear model with lm() doesn't seem to reveal a lot of information.

```
> lm(weight ~ size, data=Beard)

Call:
lm(formula = weight ~ size, data = Beard)

Coefficients:
(Intercept)         size
     108.77        15.24
```

It reveals that in the regression equation,

$$y = \alpha + \beta x + \epsilon$$

that our estimate for α (β_0 in the textbook) is 108.77, and that our estimate for β, the slope, is 15.24

lm() can give us a LOT more information than this.

Unlike t.test() and cor.test(), which simply return some text output, lm() makes an object that we can use and reuse.

We can save the linear model object that lm() creates, and examine it with the summary() function.

First, give the linear model a name, such as 'mod'

```
mod = lm(weight ~ size, data=Beard)
```

Next, examine it with summary()

```
summary(mod)
```

Linear Model output part 1/6

```
> mod = lm(weight ~ size, data=Beard)
> summary(mod)

Call:
lm(formula = weight ~ size, data = Beard)

Residuals:
     Min       1Q   Median       3Q      Max
-136.874  -32.926    4.267   30.633  114.875
```

Call:

[what the model is.]

Residuals:

[summary statistics of the residuals], mean is always zero.

Linear Model output part 2/6

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   108.77      38.16   2.850  0.00622 **
size           15.24       1.64   9.293 1.02e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coefficients:

[The name of the explanatory variable, if any]

Estimate: **[Value of slope or intercept]**

Std. Error: **[Standard error of slope or intercept]**

Linear Model output part 3/6

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   108.77      38.16   2.850  0.00622 **
size           15.24       1.64   9.293 1.02e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

t value: Obtained t-score. Calculated from Estimate / Std. Error

Pr(>|t|) is the **[*p-value against α = 0 or β = 0*]** with a two-sided alternative.

"Pr" stands for "probability", and this is the probability of getting a t-score of the obtained value or greater, given the null

The ** and *** are a quick way to determine significance.

Linear Model output part 4/6

```
Residual standard error: 59.35 on 53 degrees of freedom
Multiple R-squared:  0.6197,    Adjusted R-squared:  0.6125
F-statistic: 86.36 on 1 and 53 DF,  p-value: 1.025e-12
```

Residual standard error is a measure of variance unexplained. We won't be using this.

In this case, n=55 because we have 55 bearded dragons. There are two parameters, so df = 55 – 2 = 53

Multiple R-squared is the R^2 measure discussed in Week 2, Hr 3. This means 61.97% of the variation in Y (weight) is explained by all the X variables (only size in this case).

Linear Model output part 5/6

```
Residual standard error: 59.35 on 53 degrees of freedom
Multiple R-squared:  0.6197,    Adjusted R-squared:  0.6125
F-statistic: 86.36 on 1 and 53 DF,  p-value: 1.025e-12
```

Adjusted R-squared is similar to Multiple R-squared, but it reflects "true" variance explained.

Any randomly generated X variable will explain a tiny part of the variance by dumb luck. The adjusted r-squared removes that 'dumb luck' variance, and so is always a bit less than multiple r-squared.

The general term for this kind of adjustment is a ***penalty.***

Linear Model output part 6/6

```
Residual standard error: 59.35 on 53 degrees of freedom
Multiple R-squared:  0.6197,    Adjusted R-squared:  0.6125
F-statistic: 86.36 on 1 and 53 DF,  p-value: 1.025e-12
```

The F-statistic is used to test is any of the X variables explain any variance as a whole. This is used for AnOVa, but it isn't of much use yet.

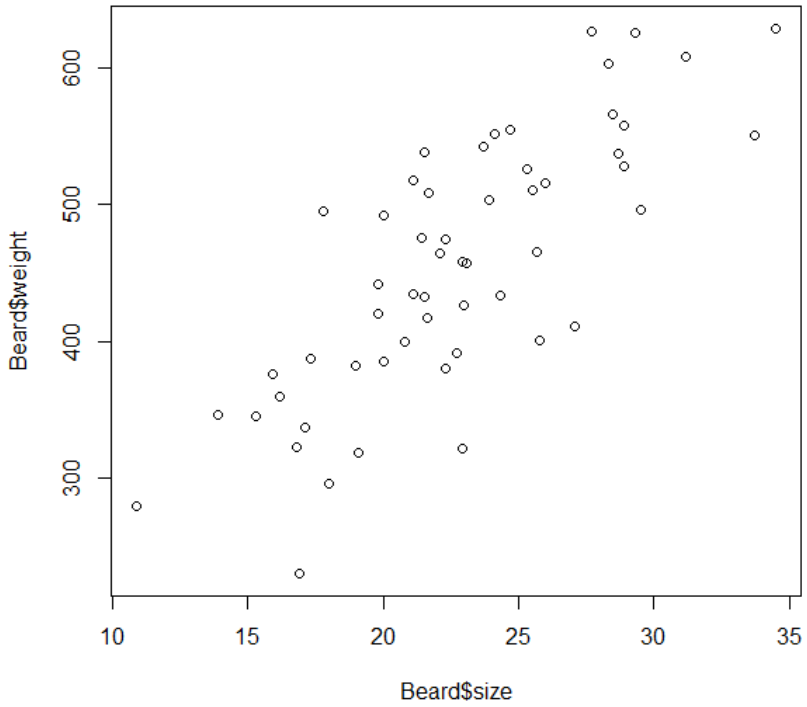
The p-value here is for this F-test. However, since there is only one variable being used a slope right now, the p-value here will be the same as the p-value for the slope coefficient.

Linear models: Bigger than you expected.

All about plot()

plot() is a generic function for graphs. When you give it sets of (x,y) coordinates, however, it draws a ***scatterplot*****.**

```
plot(Beard$size, Beard$weight)
```



plot() has many options for changing the look of graphs. There are also packages that do entirely different plotting systems. Here are some quick options:

Double the size of the points. ('cex' stands for **C**haracter **EX**pansion)

```
plot(Beard$size, Beard$weight, cex = 2)
```

Set the colour of the points to blue

```
plot(... col="Blue")
```

Make the points filled in circles

```
plot(... pch=16)
```

Give the graph a title

```
plot(... main="Bearded Dragons")
```

Give the graph axis labels

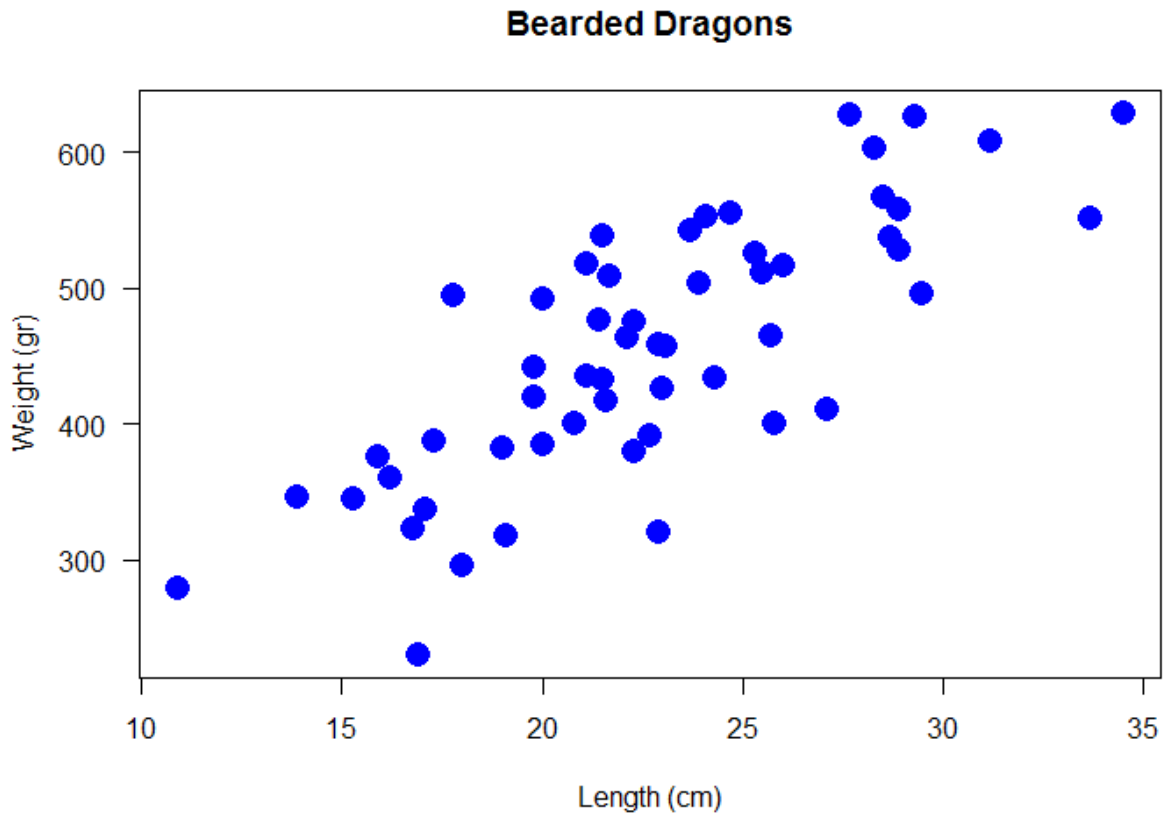
```
plot(... xlab="Length (cm)",
         ylab="Weight (grams)")
```

Make all the axis numbers horizontal

```
plot(... las=1)
```

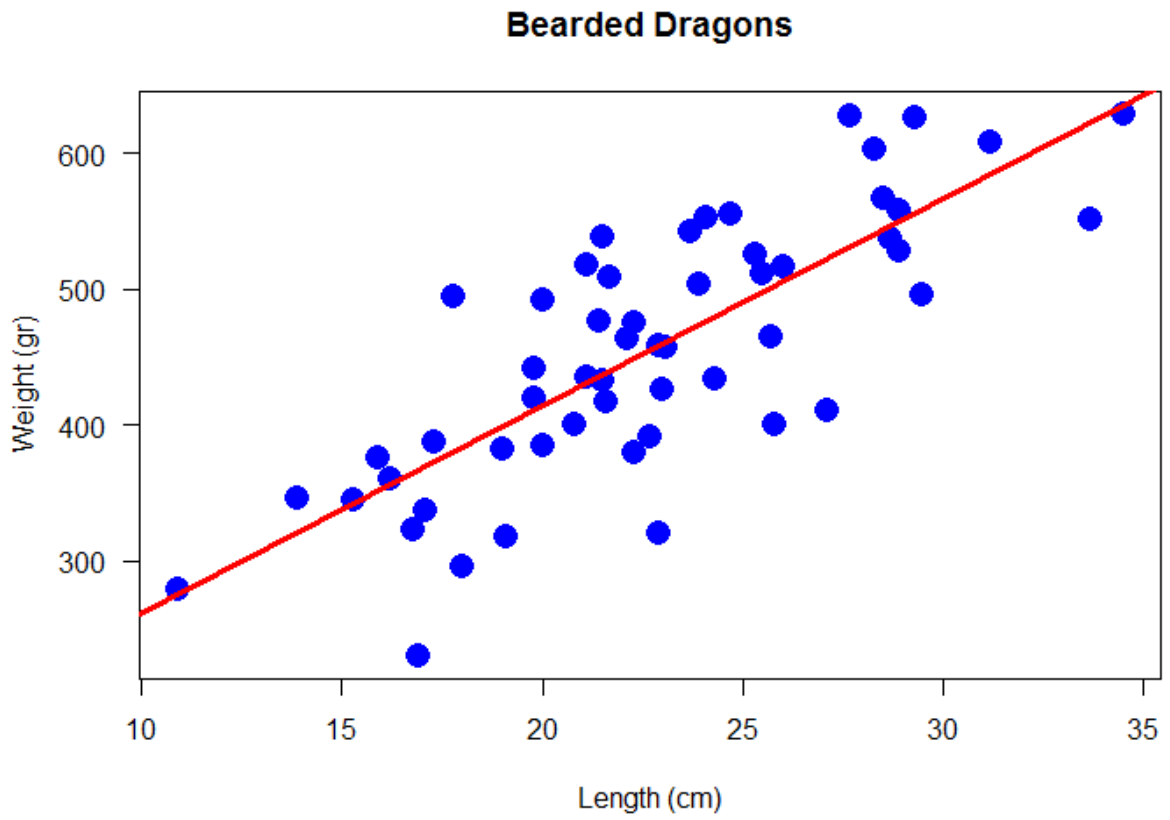
All of these can used together in any order.

```
plot(Beard$size, Beard$weight, cex = 2,
pch=16, col="Blue", main="Bearded Dragons",
xlab="Length (cm)", ylab="Weight (gr)", las=1)
```



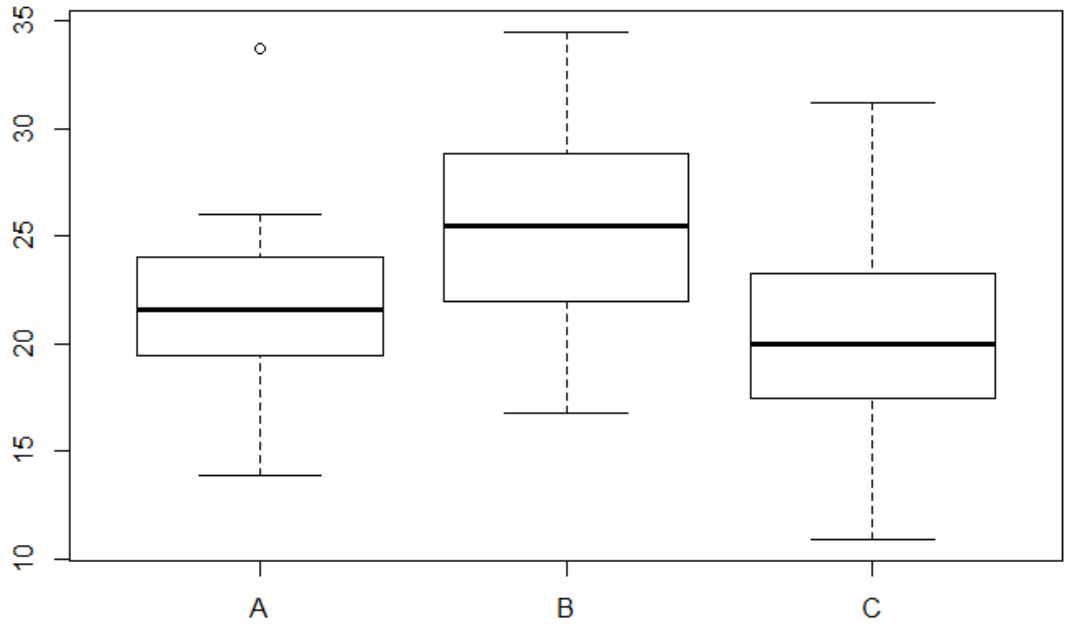
We can add lines with the abline() function. abline() can use the linear model object we already created with lm(). It also uses similar graphical options.

```
abline(mod, lwd=3, col="Red")
```



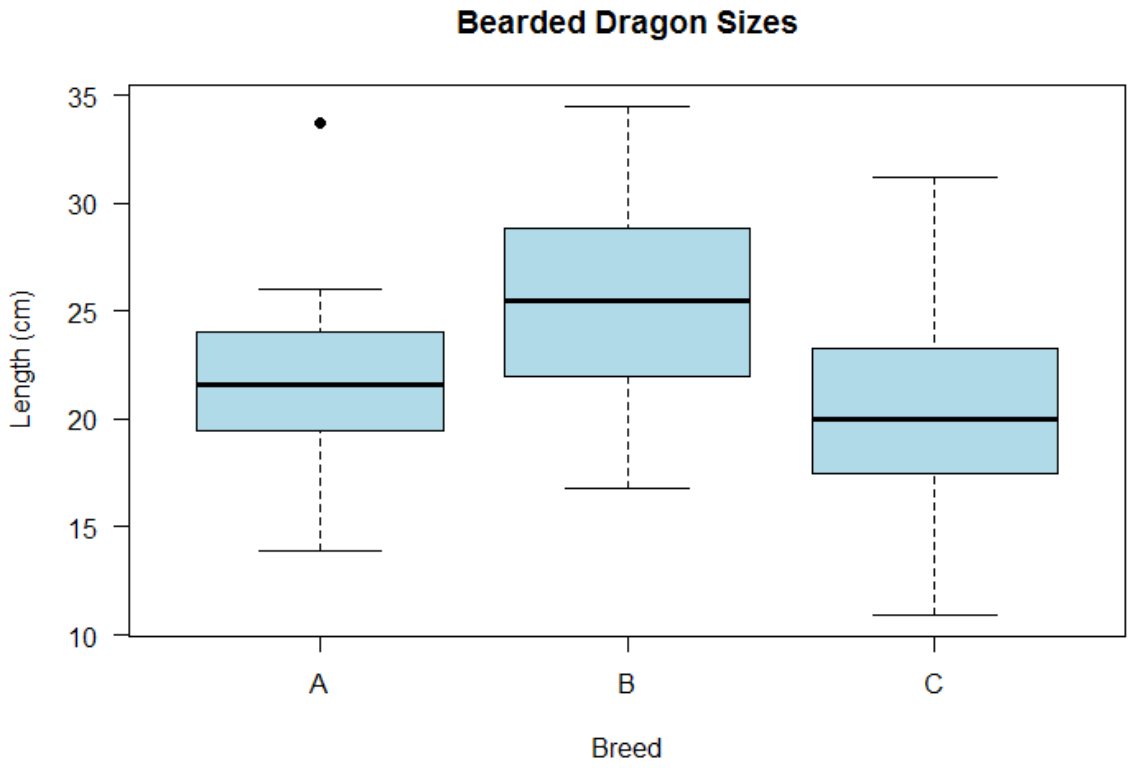
If you plot a categorical variable in the 'X' place of plot, instead of a scatterplot, you get a side-by-side boxplot.

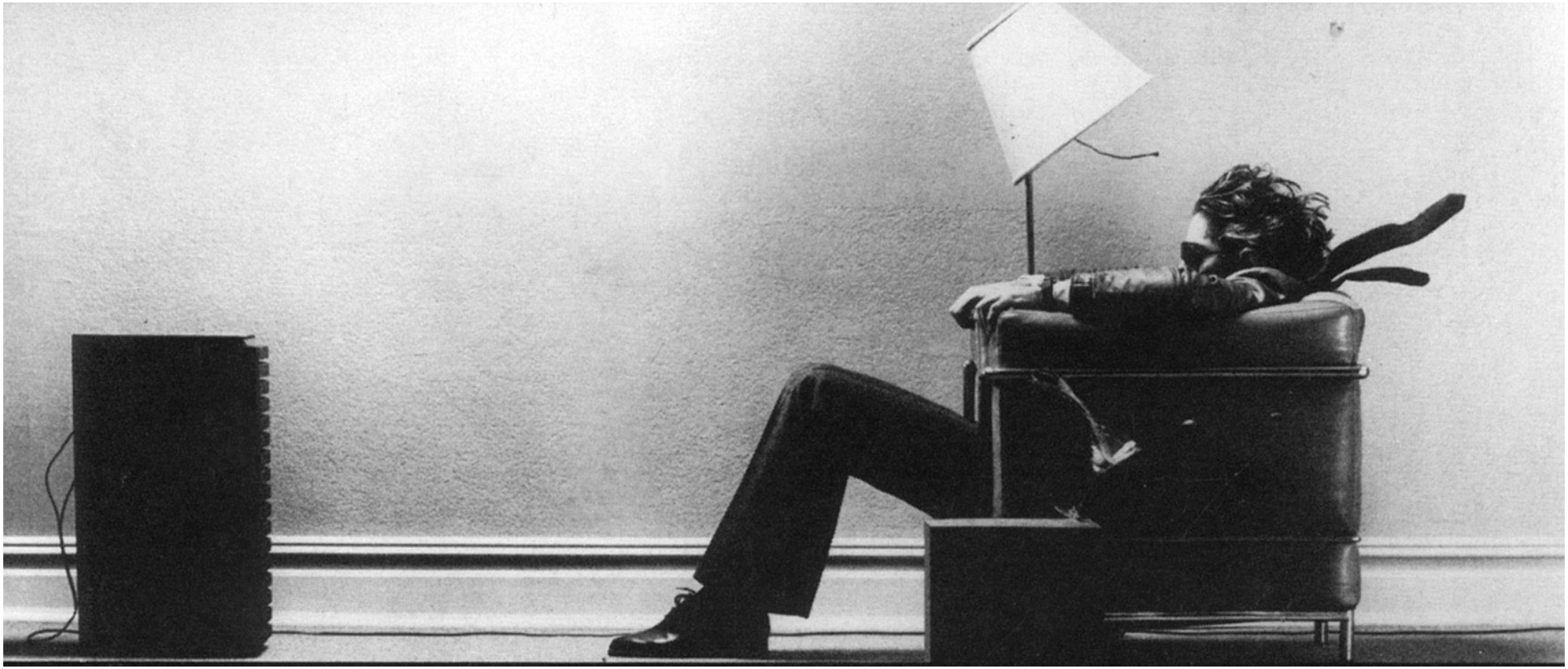
```
plot(Beard$breed, Beard$size)
```



Many of the same graphical settings are available.

```
plot(Beard$breed, Beard$size, pch=16,
col="Lightblue", main="Bearded Dragon Sizes",
xlab="Breed", ylab="Length (cm)", las=1)
```





It's a lot of output. Don't get b lo w n a w a y.

Residual plots deserve special attention. There are capable of revealing problems in a model that may not show up in the scatterplot.

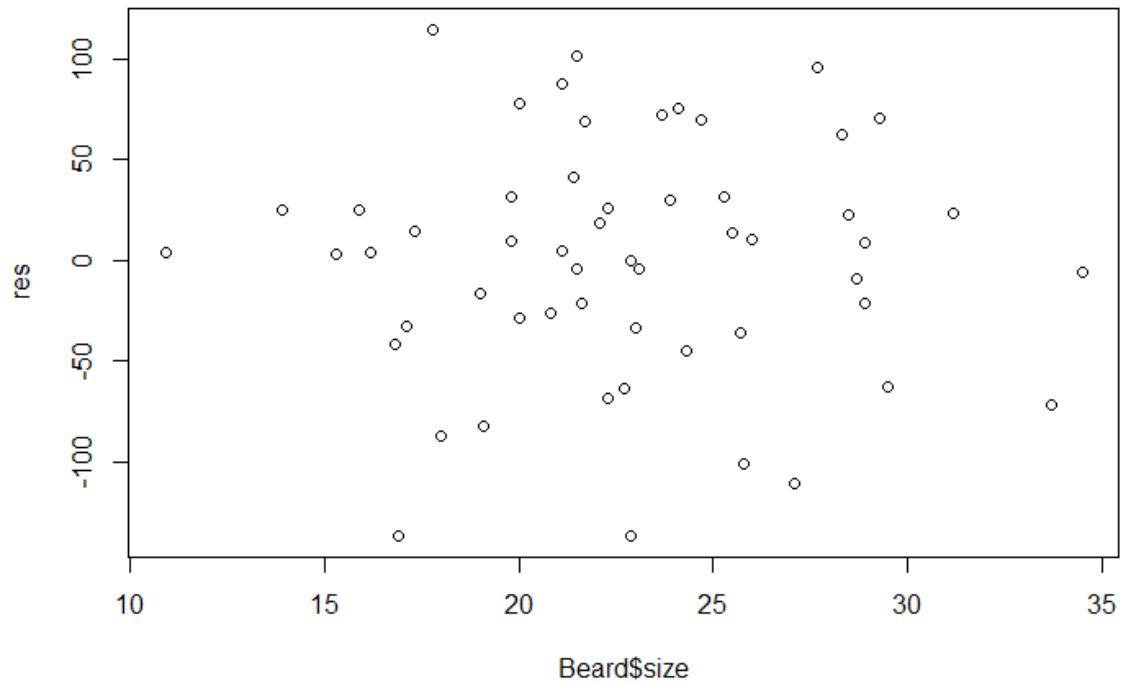
Residual comes from “residue”, meaning “that little bit left over”.

Since a good model should describe all the meaningful patterns there are in the relationship between X and Y, the only thing that should be left over is random chaos.

In other words, there should be no patterns apparent in the residuals. Just a random scatter.

To make a residual plot, take the residual values from the model with the resid() function. Plot those against X (or time, or Y).

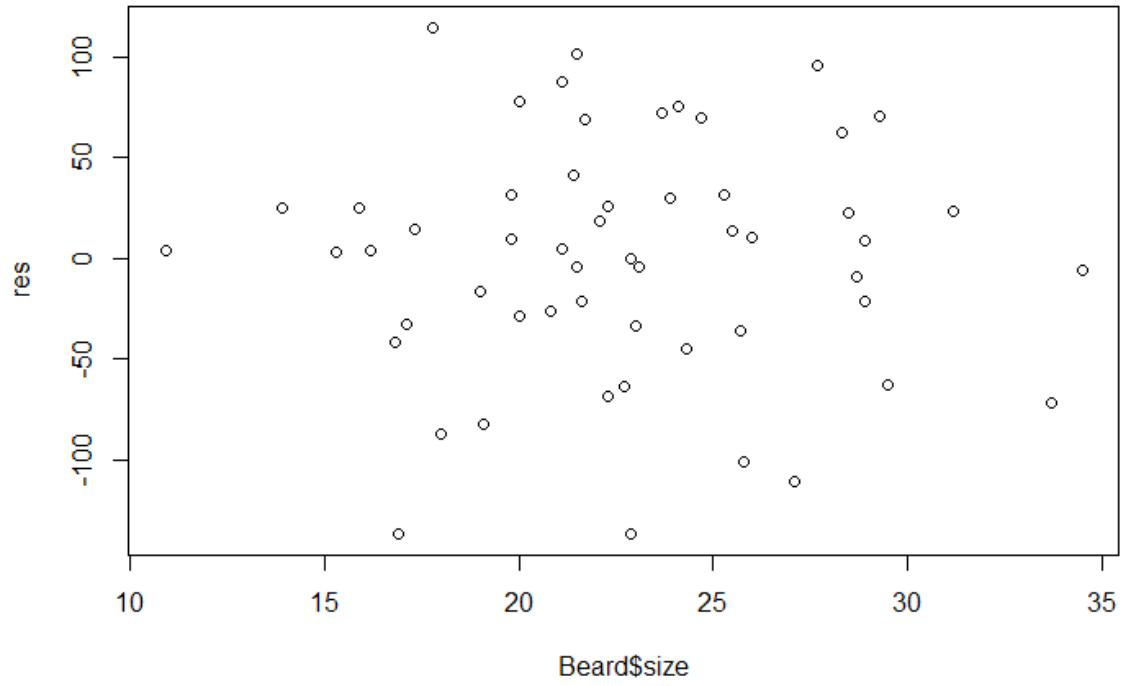
```
res = resid(mod)
plot(Beard$size, res)
```



Look for: Very high or low points, i.e. ***outliers*.**

Cone shapes, which would indicate ***unequal variance*.**

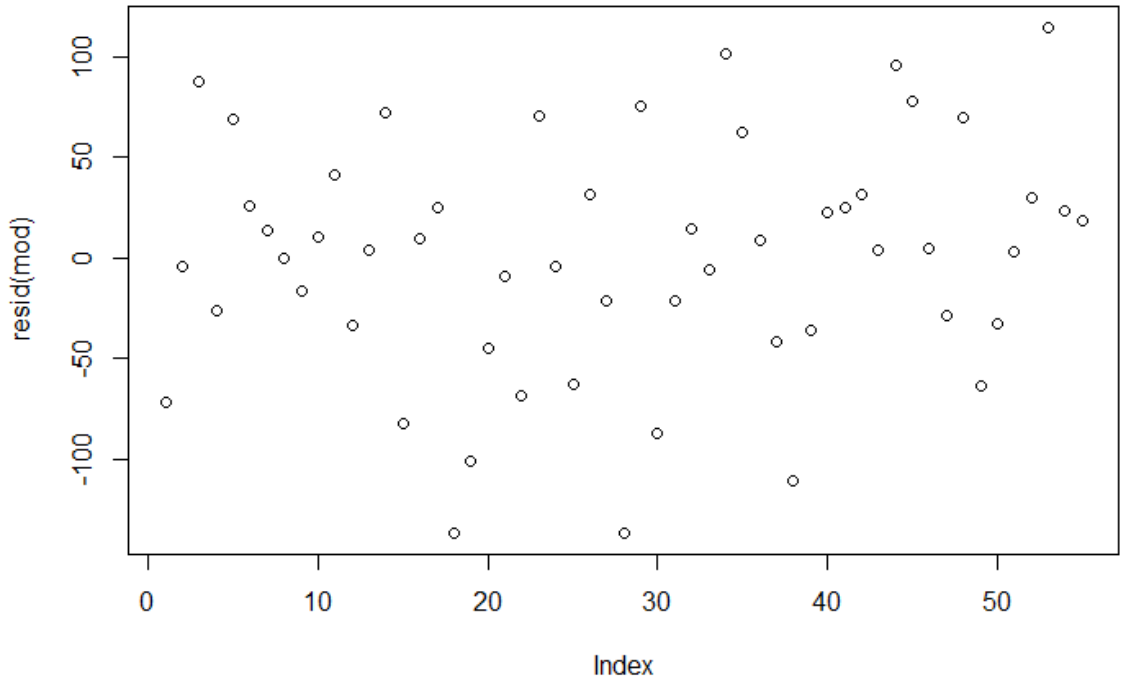
Curves, slopes, and other trends in the average, indicating non-linearity, or that additional variables are necessary.



If measurement time is a component, especially in environmental or biological field work, plot the residuals by observation order.

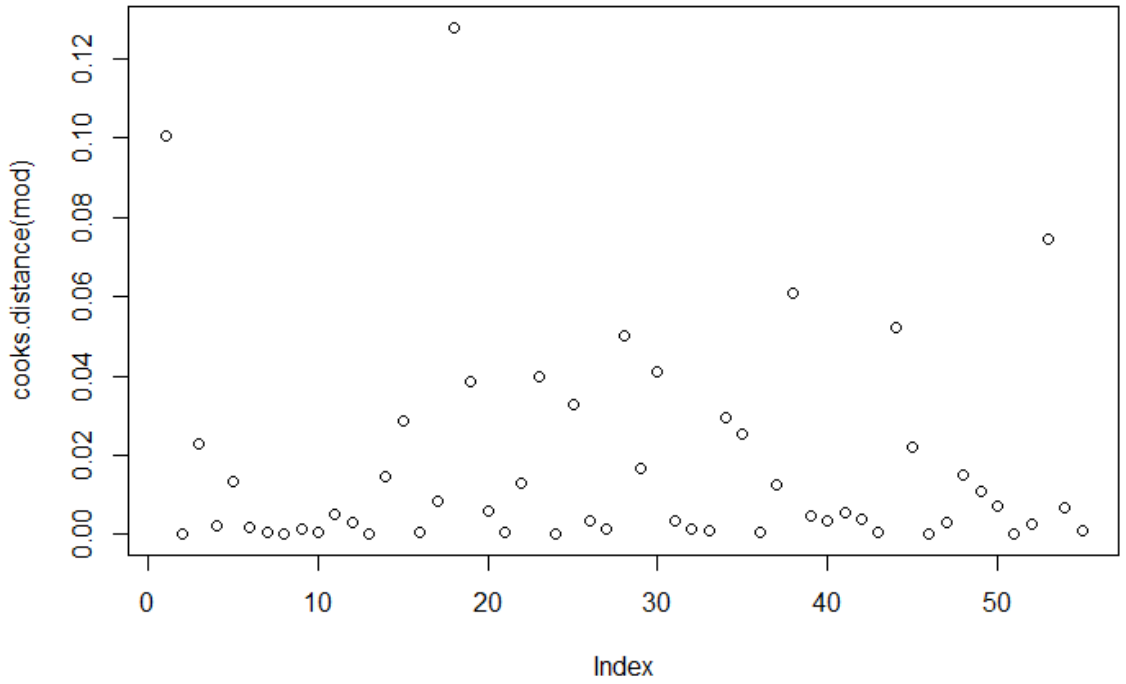
Sometimes fatigue, weather variance, or measurement drift happen

```
plot(resid(mod))
```



Finally, sometimes values can influence your model greatly, and still not appear as outliers. Cook's Distance can detect this.

```
plot(cooks.distance(mod))
```



Hopefully this isn't too fuzzy for you.

Optional Links

Details about other graphics options can be found at
http://www.statmethods.net/advgraphs/parameters.html

More details on regression diagnostics:

http://www.statmethods.net/stats/rdiagnostics.html

Thursday: Discussing causality (see Rubin readings)

That giant rodent is called a Capybara, and they are considered a nuisance in Brazil. They are like the geese of the Amazon.