# Lecture 1- A Bracing Tour of Simple Linear Regression with R

Peter Shaffery

1/19/2020

## Example: Penguins! (Con't)

```
library(palmerpenguins)
```

Say that we are researchers studying penguins in the Palmer Archipelago of Antarctica. We have collected data on three different penguin species (Adelie, Chinstrap, and Gentoo) on three different islands (Torgersen, Biscoe, and Dream). For each subject in our dataset, we have recorded the following:

- length and depth of the bill (mm)
- length of the flippers (mm)
- body mass (g)
- sex
- year of the measurement

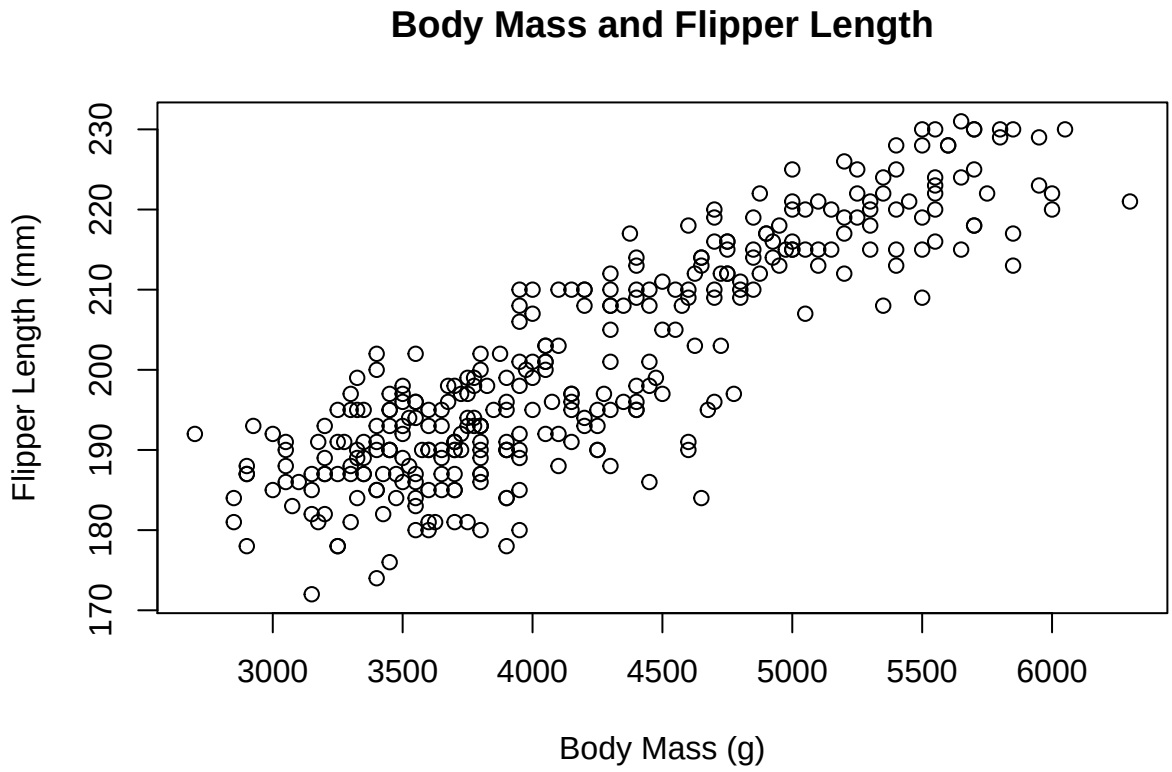
The data looks like this:

```
penguins
```

```
## # A tibble: 344 x 8
##    species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g
##    <fct>   <fct>           <dbl>         <dbl>            <int>       <int>
##  1 Adelie  Torge~           39.1          18.7              181        3750
##  2 Adelie  Torge~           39.5          17.4              186        3800
##  3 Adelie  Torge~           40.3          18                195        3250
##  4 Adelie  Torge~           NA            NA                 NA          NA
##  5 Adelie  Torge~           36.7          19.3              193        3450
##  6 Adelie  Torge~           39.3          20.6              190        3650
##  7 Adelie  Torge~           38.9          17.8              181        3625
##  8 Adelie  Torge~           39.2          19.6              195        4675
##  9 Adelie  Torge~           34.1          18.1              193        3475
## 10 Adelie  Torge~           42            20.2              190        4250
## # ... with 334 more rows, and 2 more variables: sex <fct>, year <int>
```

We are interested in the relationship between body mass ($x$) and flipper length ($y$)

```
dat = tidyr::drop_na(penguins)
x = dat$body_mass_g
y = dat$flipper_length_mm
plot(x,y,
     main='Body Mass and Flipper Length',
     xlab='Body Mass (g)',
     ylab='Flipper Length (mm)')
```




**Regression**

Linear regression comes in two flavors: simple (or 1-dimensional), and multivariate (sometimes shortened to "multiple") regression. For most of the course we will be dealing with multiple regression and its variants. However, in order to review some core statistical concepts we will start with an outline of simple regression first.

The regression model is (mathematically) very simple:

$$y = \beta_0 + \beta_1 x + \epsilon$$

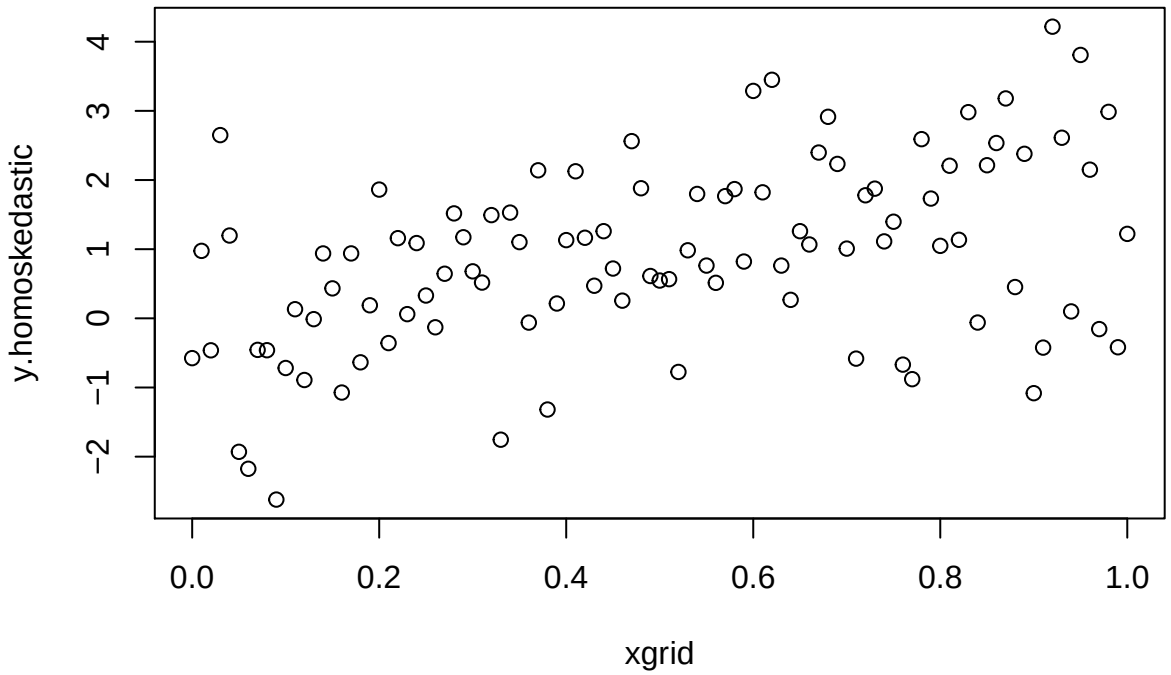
Here $\beta_0$ and $\beta_1$ are called *model coefficients*. $\beta_0$ by itself is the **intercept** and $\beta_1$ is the **slope**. We say that $\beta_1$ represents the **average increase in $y$ for a unit increase in $x$**.

We also introduce $\epsilon$, which is the **error term**. In this course we will always assume that $\epsilon$ is a *random variable*. For any linear regression model we further assume the following facts:

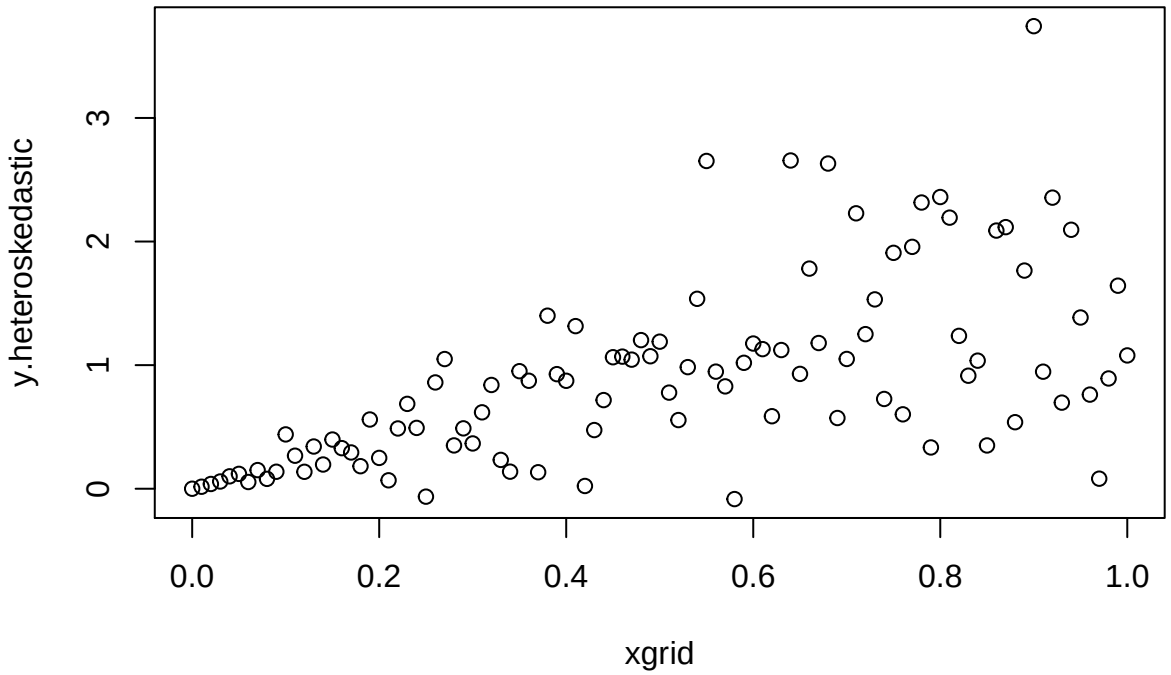
1. **Normality** $\epsilon_i$ is normally distributed with mean 0 and variance $\sigma^2$ (which we will assume is known, for now):
$$\epsilon \sim N(0, \sigma^2)$$
2. **Independence** The error terms $\epsilon_1, ..., \epsilon_n$ are *independent*
3. **Homoskedasticity** The value of $\sigma^2$ is constant for all values of $x$

```
xgrid = seq(0,1,.01)
n = length(xgrid)
y.homoskedastic = rnorm(n, 2*xgrid, 1)
y.heteroskedastic = rnorm(n, 2*xgrid, 1*xgrid)

plot(xgrid,y.homoskedastic)
```




```
plot(xgrid,y.heteroskedastic)
```




Assumption 1 and 2 are so fundamental to regression (and statistics) that we give them a special name: **independent and identically distributed** or **iid** (as in "we assume the errors $\epsilon_i$ are **iid** normal with mean 0 and variance $\sigma^2$"). Note that iid random variables can have any distribution, not just normal.

The aim of regression is to determine the *best guess* values of $\beta_0$ and $\beta_1$, denoted $\hat{\beta}_0$ and $\hat{\beta}_1$, given a sequence of observations $(x_1, y_1), ..., (x_n, y_n)$. To do so, we will choose values for $\hat{\beta}_0$ and $\hat{\beta}_1$ which minimize the *error* in our model.

For each observations $(x_i, y_i)$ and for a pair of guess $\hat{\beta}_0, \hat{\beta}_1$, note that we can re-arrange the regression model:

$$\epsilon_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i$$

Now recall that (by Assumption 1) $\epsilon_i$ is normally distributed with mean 0, thus:

$$P[\epsilon_i|\sigma^2] = N(\epsilon_i;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-\epsilon_i^2}{2\sigma^2}\right]$$

By substituting $y_i - \beta_0 - \beta_1 x_i$ for $\epsilon_i$ in the above equation, we can get the probability of observing $y_i$, given $\beta_0, \beta_1$ and $x_i$:

$$P[y_i|\hat{\beta_0},\hat{\beta_1},x_i,\sigma^2] = N(\hat{\beta_0}+\hat{\beta_1}x_i;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-(y_i-\hat{\beta_0}-\hat{\beta_1}x_i)^2}{2\sigma^2}\right]$$

Since $\beta_0$ and $\beta_1$ are *free variables* (we have no observed them, only $x_i$ and $y_i$), we could rewrite the above probability as a function of the $\beta$s. This function is known as the **likelihood** function:

$$L(\hat{\beta_0},\hat{\beta_1};x_i,y_i) = P[y_i|\hat{\beta_0},\hat{\beta_1},x_i,\sigma^2]$$

The likelihood function is a central concept in statistical modeling. Since the $\epsilon_i$ are independent (by Assumption 2), we can write the overall likelihood function:

$$\begin{aligned} L(\hat{\beta_0},\hat{\beta_1};x_1,...,x_n,y_1,...,y_n) &= \prod_{i=1}^{n} P[y_i|\hat{\beta_0},\hat{\beta_1},x_i,\sigma^2] \\ &= \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\hat{\beta_0}-\hat{\beta_1}x_i)^2\right] \end{aligned}$$

Recall that the likelihood represents the *probability of observing the data you got, given that the coefficient values are $\beta_0$ and $\beta_1$*. It follows that the "best guess" values of $\beta_0$ and $\beta_1$ are the ones which maximize the probability of the observed data, ie. the **maximum likelihood estimator**.

Because the function $L$ is the product of a bunch of probabilities, which can get very small, often the numerical value of $L$ is computationally difficult to work with (particularly for large $N$). It is far more convenient to maximize the *log-likelihood*:

$$\begin{aligned} l(\hat{\beta_0},\hat{\beta_1};x_1,...,x_n,y_1,...,y_n) &= \ln\left[L(\hat{\beta_0},\hat{\beta_1};x_1,...,x_n,y_1,...,y_n)\right] \\ &= -\ln(\sigma\sqrt{2\pi}) + \frac{-1}{2\sigma^2} * \sum_{i=1}^{n}(y_i-\hat{\beta_0}-\hat{\beta_1}x_i)^2 \\ &\propto -\sum_{i=1}^{n}(y_i-\hat{\beta_0}-\hat{\beta_1}x_i)^2 \end{aligned}$$

We finally observe that the log-likelihood is the same as the negative square error, and that choosing $\hat{\beta_0}$ and $\hat{\beta_1}$ to maximimize $l$ is equivalent to:

$$\hat{\beta_0},\hat{\beta_1} = \text{argmin}_{\beta_0,\beta_1}\sum_{i=1}^{n}(y_i-\beta_0-\beta_1x_i)^2$$

Coefficients estimated in this way are thus called the **least squares estimates**:

It can be shown (eg. by setting $\frac{\partial \text{Square Error}}{\partial\beta_1} = 0$ and solving for $\beta_1$) that the values which minimize the sum of squares are:

$$\begin{aligned} \hat{\beta_1} &= \frac{\sum_{i=1}^{n}(y_i-\bar{y})(x_i-\bar{x})}{\sum_{i=1}^{n}(x_i-\bar{x})^2} \\ \hat{\beta_0} &= \bar{y} - \hat{\beta_1}\bar{x} \end{aligned}$$

Note that we could also write $\hat{\beta}_1 = \frac{\text{Cov}(x,y)}{\text{Var}[x]} = \text{Cor}(x,y)\frac{s_y}{s_x}$. Simple linear regression is therefore fundamentally the same kind of procedure as calculating covariance (or correlation) between $x$ and $y$.

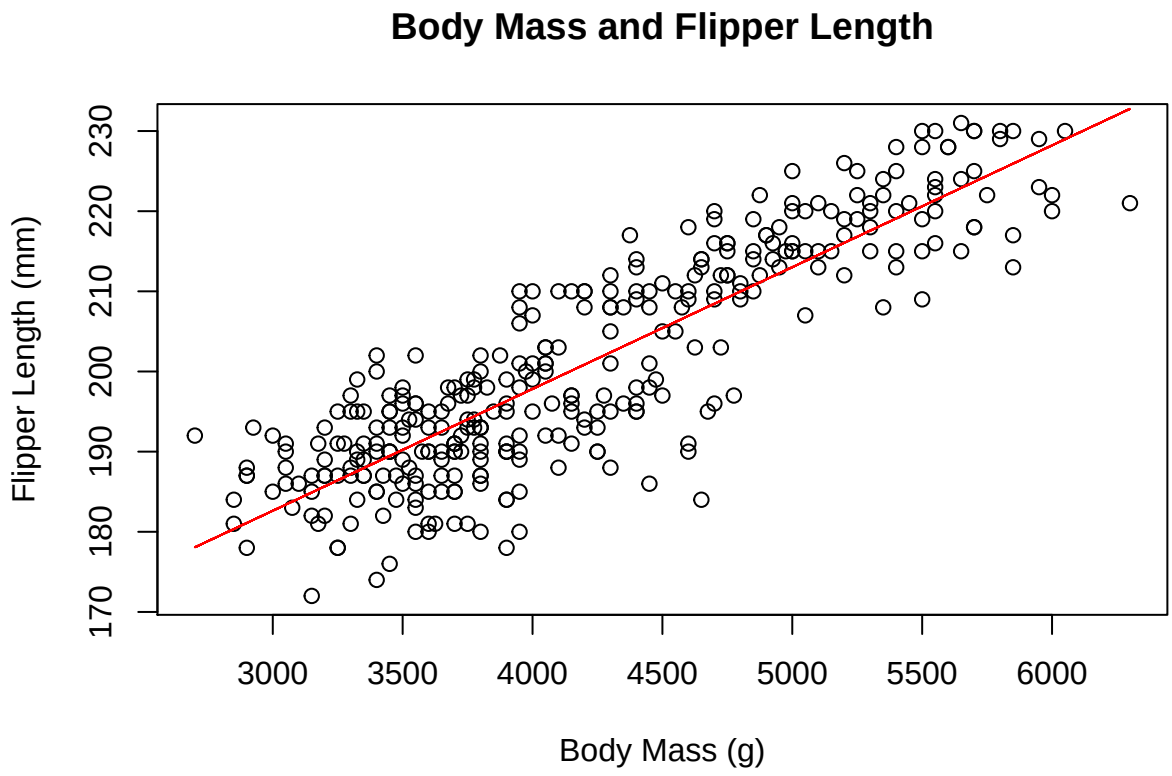
Let's use the covariance formula now to calculate $\hat{\beta}_0$ and $\hat{\beta}_1$ for our example data:

```
beta.1.hat = cov(x,y)/var(x)
beta.0.hat = mean(y) - beta.1.hat*mean(x)
c(beta.0.hat,beta.1.hat)
```

```
## [1] 137.03962089   0.01519526
```

**Prediction** Having estimated $\hat{\beta}_0 = 137$ and $\hat{\beta}_1 = .015$ we can make *predictions* about unobserved datapoints. Say that we measure a new penguin with body mass $x_{\text{new}} = 5000g$, our "best guess" for its flipper length is $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x = 137 + .015 * 5000 = 212mm$.

```
xgrid = seq(min(x),max(x),.01)
fit = beta.0.hat + beta.1.hat*xgrid
plot(x,y,main='Body Mass and Flipper Length',xlab='Body Mass (g)',ylab='Flipper Length (mm)')
lines(xgrid,fit,col='red')
```




### Hypothesis Testing

So far we've been focusing primarily on the *modeling* and not much on the *statistics*. Say that we now want to make a decisions: whether body mass and flipper length are actually related or not. One way to do so would be to decide whether $\beta_1$ is "big enough". Unfortunately, like covariance, the regression model coefficients are also not independent of scale, so it's not clear how "big" is "big enough"

```
x.lbs = x/454 # 454g per lb
beta.1.hat.lbs = cov(x.lbs,y)/var(x.lbs)
beta.0.hat.lbs = mean(y) - beta.1.hat.lbs*mean(x.lbs)
c(beta.0.hat.lbs,beta.1.hat.lbs)
```

```
## [1] 137.03962   6.89865
```

We will therefore use a procedure called **hypothesis testing**.

Hypothesis testing compares two "models":

1. The **null hypothesis** ($H_0$), $\beta_1 = 0$
2. The **alternate hypothesis** ($H_A$) $\beta_1 \neq 0$

$H_0$ represents the case of "no effect", and in this example it would be the model with $\beta_1 = 0$ (ie. no relationship between body mass and flipper length). The alternate hypothesis, $H_A$, would be that there *is* an effect, ie.that $\beta_1 \neq 0$.

Recall the formula for $\hat{\beta}_1$:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

It can be shown (through properties listed in the Appendix), that if the $x_i$ are constants and the $\epsilon_i$ are iid normal, then the $y_i$ are iid normal. Since $\hat{\beta}_1$ is a function of these $y_i$, then $\hat{\beta}_1$ is *also* a *random variable*. In fact, $\hat{\beta}_1$ is normally distributed:

$$\hat{\beta}_1 \sim N(\beta_1^*, \tau^2)$$

Where $\beta_1^*$ is the *true* value of $\beta_1$, and the variance $\tau^2$ is:

$$\tau^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Since the null hypothesis $H_0$ is equivalent $\beta_1^* = 0$, and it's equivalent to saying that:

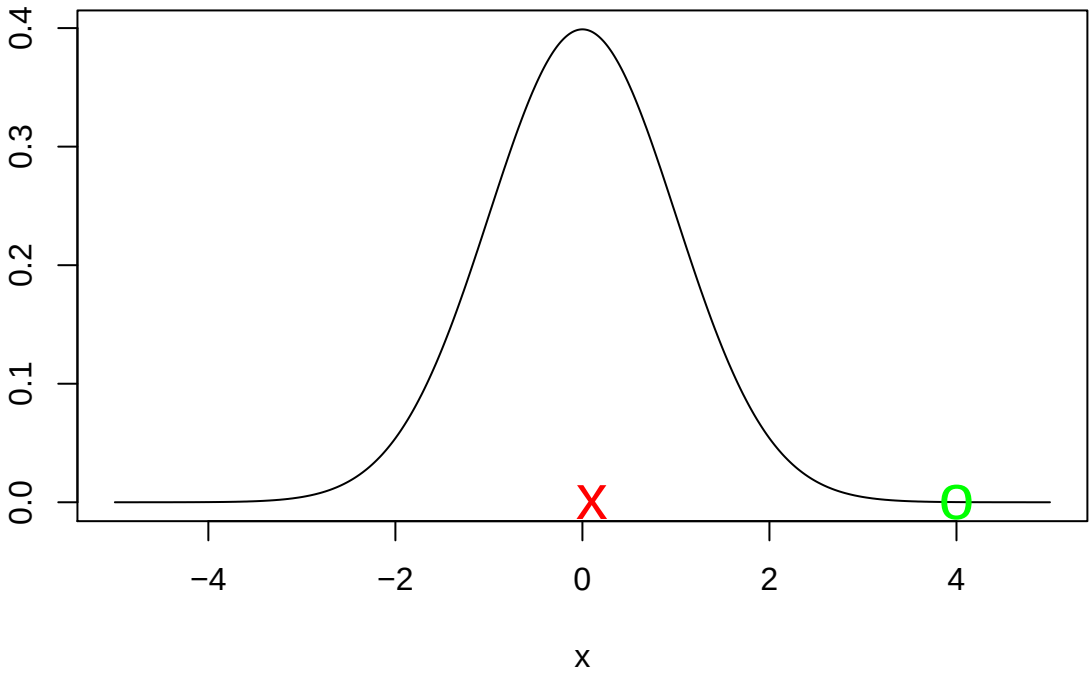
$$\hat{\beta}_1 \sim N(0, \tau^2)$$

We now have a way to determine whether $\hat{\beta}_1$ is "big enough"! If we scale $\hat{\beta}_1$ by $\tau$, then:

$$\frac{\hat{\beta}_1}{\tau} \sim N(0, 1)$$

Thus if $\frac{\hat{\beta}_1}{\tau}$ falls sufficiently far out on the standard normal curve ($N(0,1)$) then we'll say that it's "big" and reject $H_0$. Otherwise we'll "fail to reject $H_0$".

```
xgrid = seq(-5,5,.01)
std.norm.curve = dnorm(xgrid,0,1)
plot(xgrid,std.norm.curve, type='l',xlab='x',ylab='') #plot curve as a line, rather than series of poin
points(.1,0,col='red',pch='x',cex=2)
points(4,0,col='green',pch='o',cex=2)
```




There's only one snag: $\tau$ depends on the value of $\sigma$. Up until now we've assumed that we know this value, but in reality that never happens, so we'll have to estimate it:

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n-2}\sum_{i=1}^{n}\hat{\epsilon_i}^2 \\ &= \frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \\ &= \frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1 x_i)^2\end{aligned}$$

The sum of the squared error terms, $\sum_{i=1}^{n}\hat{\epsilon}_i^2$ is an important quantity, as it measures the overall "fit" of our model (on average, how close to $y_i$ was the estimate $\hat{y}_i$). It therefore gets a special name: Sum of Square Error (SSE). Alternatively, since the $\hat{\epsilon}_i$ are typically called "residuals", you will sometimes see the SSE called "residual error".

We can now estimate $\tau$:

$$\hat{\tau} = \sqrt{\frac{\hat{\sigma}^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$$

And calculate:

$$\hat{t} = \frac{\hat{\beta}_1}{\hat{\tau}}$$

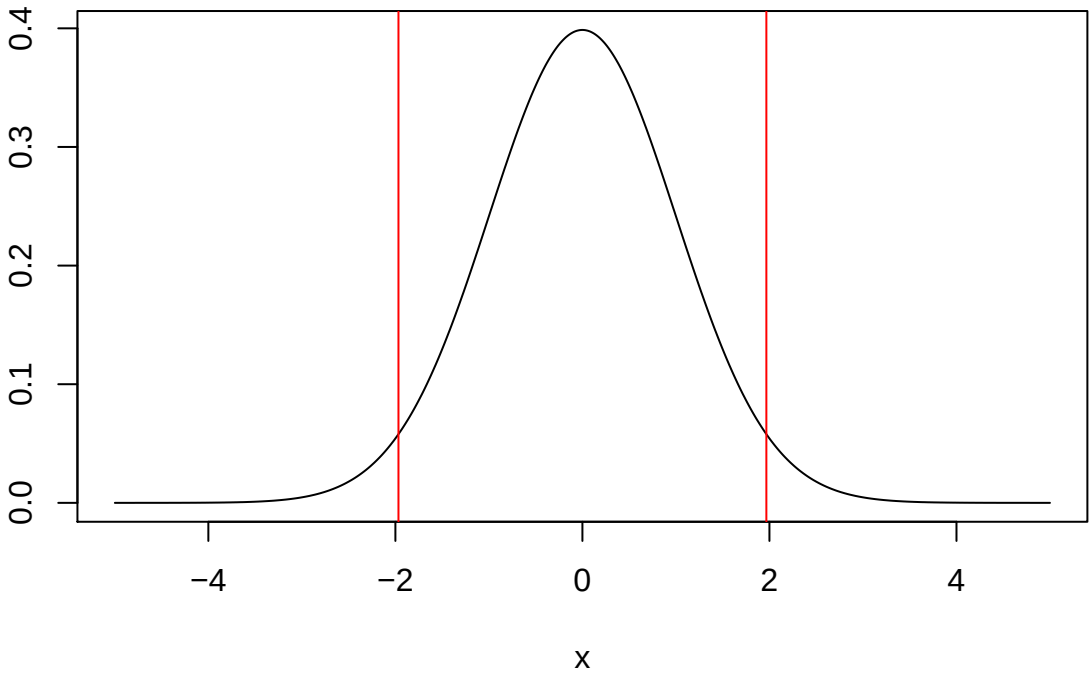
Unfortunately, because the estimate $\hat{\tau}$ is *also* a random variable, the random variable $t$ is no long normally distributed. Instead it has a *Student-T Distribution with n-2 degress of freedom.*

The distribution of $t$ has no dependence on the scale of $x$ and $y$, so we're free to choose a cutoff threshold $t^*$, such that $P[|t| \geq t^*] = \alpha$.

```
n = length(x)
xgrid = seq(-5,5,.01)
```

```
std.t = dt(xgrid,n-2)
alpha = .05
t.star = qt(1-alpha/2,df=n-2)

plot(xgrid,std.t, type='l',xlab='x',ylab='')
abline(v=t.star,col='red')
abline(v=-t.star,col='red')
```




If $|\hat{t}| \geq t^*$ then we'll reject $H_0$, otherwise we'll fail to reject.

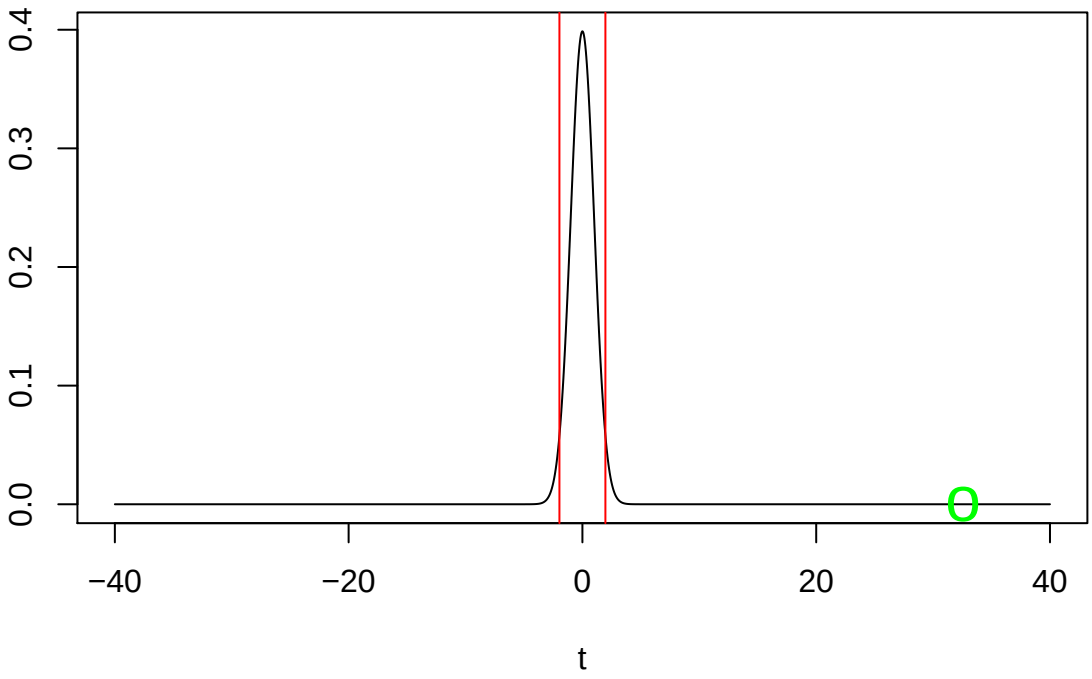
Let's test the hypothesis that penguin body mass is related to flipper length with a false positive rate of $\alpha = .05$:

```
y.hat = beta.0.hat + beta.1.hat * x
eps.hat = y - y.hat
SSE = sum( eps.hat**2 )
sigma2.hat = SSE/(n-2)
tau.hat = sqrt( sigma2.hat/sum( (x - mean(x))**2 ) )
t.hat = beta.1.hat/tau.hat
n=length(x)

t.grid = seq(-40,40,.01)
t.dist = dt(t.grid, n-2)

alpha = .05
t.star.high = qt(1-alpha/2,df=n-2)
t.star.low = -t.star.high

plot(t.grid,t.dist, type='l',xlab='t',ylab='')
abline(v=t.star.low,col='red')
abline(v=t.star.high,col='red')
points(t.hat,0,col='green',pch='o',cex=2)
```




```
print(t.hat)
```

```
## [1] 32.56217
```

Wow that's really far out there! We therefore confidently reject $H_0$.

### Alternative Formulations

**P-Values** A common alternative to selecting $t^*$ explicitly is to consider the **p-value** of $t$, that is the probability $P[|t| \geq |\hat{t}||\beta_1 = 0]$. This gives us a sense of how "extreme" the observed value of $t$ was, relative to what it would be under $H_0$. Let's calculate that now:

```
p.val = pt(t.hat,n-2,lower.tail=FALSE) + pt(-t.hat,n-2)
p.val
```

```
## [1] 3.132836e-105
```

Since this p-value is lower than $\alpha = .05$, we could come to the same conclusion and reject $H_0$.

**Confidence Intervals** Alternately we could assess the **uncertainty** in $\hat{\beta}_1$. Since $\hat{\beta}_1$ is normal, "most of the time" $\beta_1^*$ is going to fall inside the interval:

$$\text{CI} = [\hat{\beta}_1 - t_{\{n-2,\alpha/2\}}\hat{\tau}, \hat{\beta}_1 + t_{\{n-2,\alpha/2\}}\hat{\tau}]$$

Where $t_{\{n-2,\alpha/2\}}$ is chosen such that $P[t > t_{\{n-2,\alpha/2\}}] = \alpha/2$

If $0 \notin$ CI then we reject $H_0$

```
t.quant = qt(1-alpha/2,df=n-2)
CI = c( beta.1.hat-t.quant*tau.hat, beta.1.hat+t.quant*tau.hat )
print(CI)
```

```
## [1] 0.01427728 0.01611325
```

### Choosing $\alpha$

Notice that our decision process has four outcomes:

| | **Fail to Reject $H_0$** | **Reject $H_0$** |
|---|---|---|
| $H_0$ **True** | Good | False Positive (Type 1 Error) |
| $H_0$ **False** | False Negative (Type 2 Error) | Good |

The value of $\alpha$ is the probability of a False Positive. When $\alpha$ is small False Positives are rare (since our threshold of evidence is high). However, if $\alpha$ is too small then the probability of a False Negative is high. Choosing $\alpha$ is a compromise between these tradeoffs.

## Automating

Doing all this calculation by hand is a huge pain, it's way easier to use R's built in functionality:

```
linear.model = lm(y~x)
summary(linear.model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.698  -4.983   1.056   5.101  13.933
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.370e+02  1.999e+00   68.56   <2e-16 ***
## x           1.520e-02  4.667e-04   32.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.847 on 331 degrees of freedom
## Multiple R-squared:  0.7621, Adjusted R-squared:  0.7614
## F-statistic:  1060 on 1 and 331 DF,  p-value: < 2.2e-16
```

R has some smart logic around column names also:

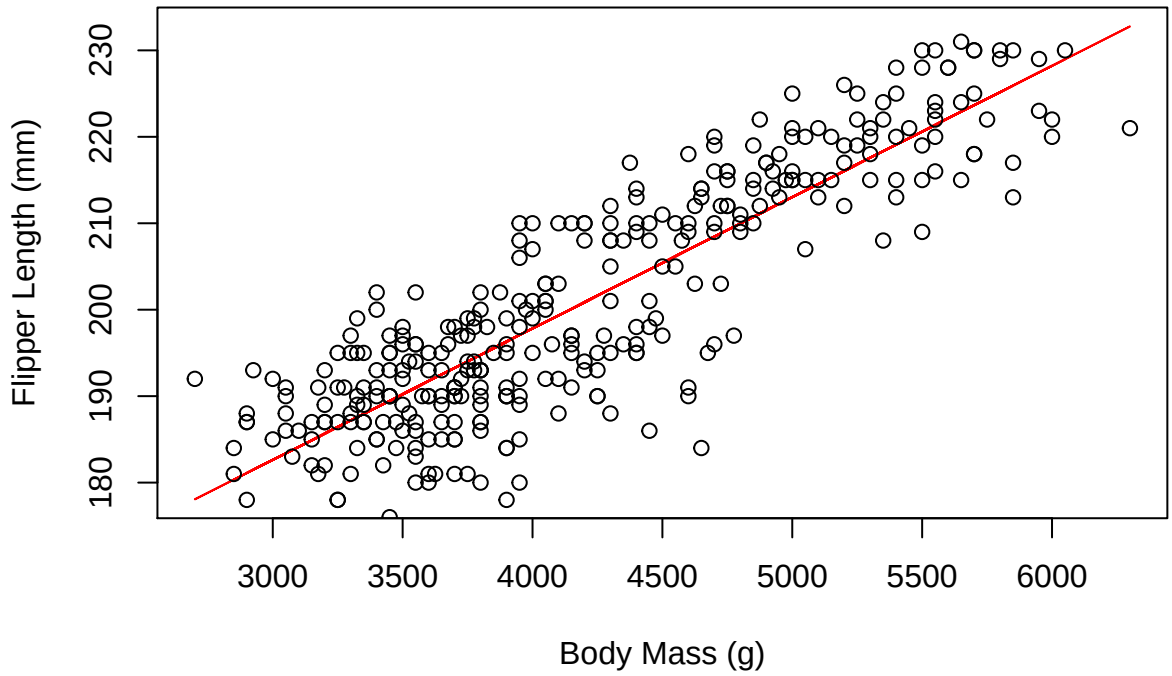
```
linear.model2 = lm(flipper_length_mm ~ body_mass_g, data= dat)
summary(linear.model2)
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ body_mass_g, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.698  -4.983   1.056   5.101  13.933
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.370e+02  1.999e+00   68.56   <2e-16 ***
## body_mass_g 1.520e-02  4.667e-04   32.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 6.847 on 331 degrees of freedom
## Multiple R-squared:  0.7621, Adjusted R-squared:  0.7614
## F-statistic:  1060 on 1 and 331 DF,  p-value: < 2.2e-16
```

We can plot the best-fit line use `predict`:

```
y.hat = predict(linear.model2)
plot(x,y.hat,type='l',col='red',xlab='Body Mass (g)',ylab='Flipper Length (mm)')
points(x,y)
```




Alternately we can predict new datapoints:

```
new.dat = data.frame('body_mass_g'=c(3500,4500))
predict(linear.model2, newdata=new.dat)
```

```
##        1        2
## 190.2230 205.4183
```

## Assessing Model Quality

### R-squared

P-values and t-statistics ($\hat{t}$) tell use how plausible it is that $\beta_1 = 0$, but they don't tell us how good our model fit is. For that, we need to turn back to the $\epsilon_i$.

Recall that (by definition):

$$\hat{\epsilon_i} = y_i - \hat{y_i}$$

So, rearranging:

$$y_i = \hat{y_i} + \hat{\epsilon_i}$$

Now let's subtract $\bar{y}$ from both sides:

$$y_i - \bar{y} = (\hat{y_i} - \bar{y}) + \hat{\epsilon_i}$$

$$\text{Deviation from mean} = \text{Deviation due to fit} + \text{Deviation due to error}$$

Some fancy algebra (which we will come back to), can be used to show:

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}\hat{\epsilon_i}^2$$
$$\text{SST} = \text{SSR} + \text{SSE}$$

We can rewrite this one more:

$$\text{SSR} = 1 - \frac{\text{SSE}}{\text{SST}}$$

We denote the quantity on the left $R^2$, and it can be shown that $R^2 = \text{Cor}(x, y)^2$

## Appendix: Properties of Normal Random Variables

In this course we'll make a lot of use of normally distributed random variables:

$$x \sim N(\mu, \sigma^2)$$

It's there useful to bear in mind some of the more convenient properties of a normal random variable.

1. If $x \sim N(\mu, \sigma^2)$ and $a$ is a fixed constant, then the sum $a + x \sim N(a + \mu, \sigma^2)$
2. If $x \sim N(\mu, \sigma^2)$ and $b$ is a fixed constant, then the product $\beta x \sim N(b\mu, (b\sigma)^2)$
3. If $x \sim N(\mu_x, \sigma_x^2)$ and $y$ $N(\mu_y, \sigma_y^2)$ are independent, then the sum $x + y = N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$