Regression

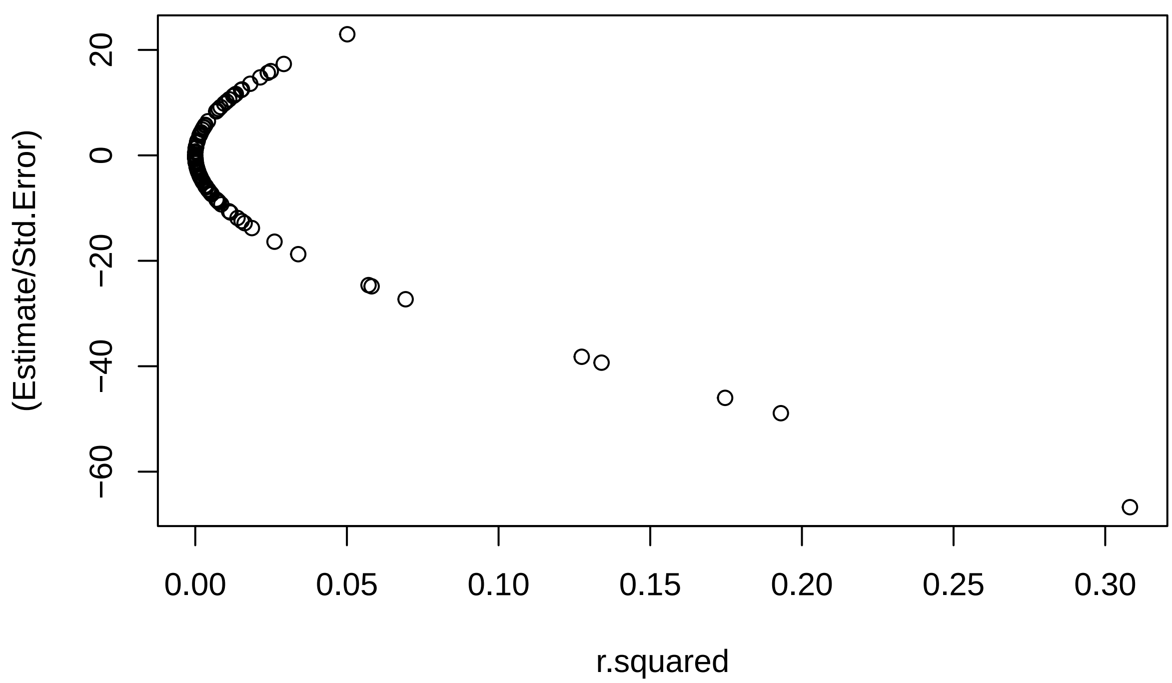
Phil Shea

2024-09-10

Relationship between the Estimate, Standard Error, and Regression Coeffficient

We saw a surprising relationship between the ratio of the estimate to the standard error of the estimate $(\widehat{\beta}/S_{\widehat{\beta}})$. Here was the surprise:

```
plot( (Estimate/Std.Error) ~ r.squared, data = MD2DF)
```




```
#abline( a = 0, b = 1/2)
```

The graph above was generated from data which had a different data, although all the data had the same number of samples. One other characteristic: the data was an over-sampled time series of 10,000 samples each.

Random Relationships

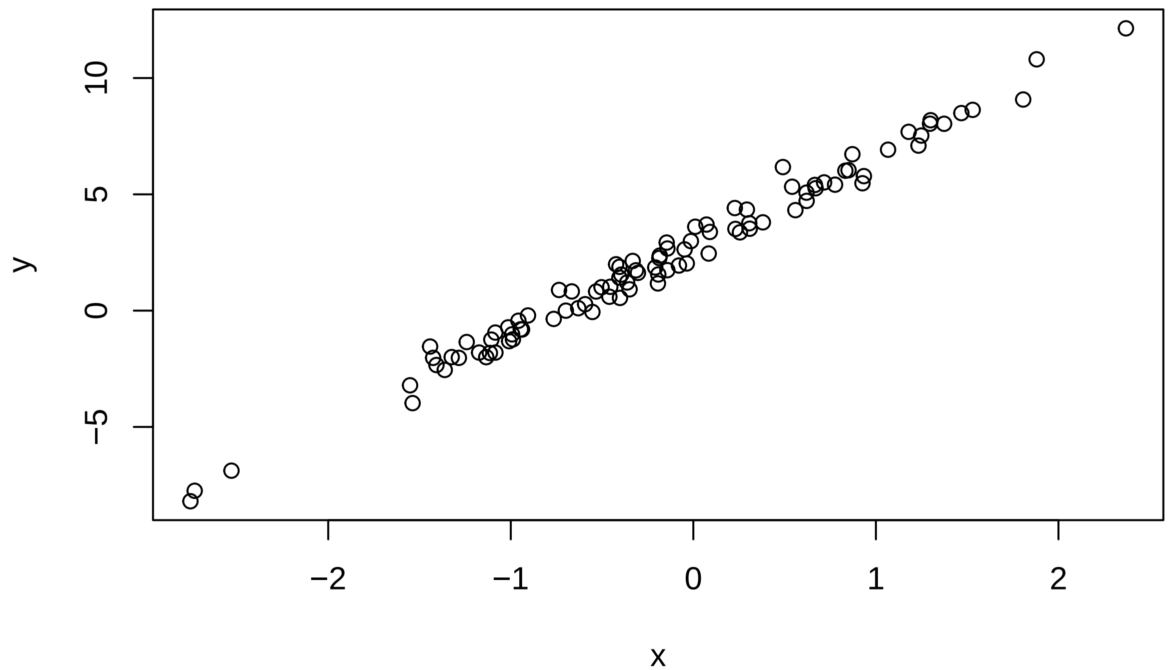
To examine the effect observed above, we generated a *lot* of random data. The function `sbor` below will generate a random number of samples (drawn from a Poisson distribution), and will generate random data from an equation $y_i = a + bx_i + \varepsilon_i$. The function's parameters set the means of the distribution of the equation parameters. The equation parameters are drawn from normal distributions with standard deviation of one, and x_i is drawn from a standard normal distribution $N(0, 1)$. The error terms are drawn from a normal distribution with mean zero and standard deviation with a default of 1/2.

```
sbor <- function(n, a, b, sde=0.5, plt=FALSE) {
   #  a & b are the means of normal distributions of sd one,
   # n is the mean of a Poisson distribution.
   nn <- 0
   while (nn == 0) nn <- rpois( 1, lambda = n)
   aa <- rnorm( 1, mean = a)
   bb <- rnorm( 1, mean = b)
   x  <- rnorm( nn) # mean zero
   y  <- aa + bb * x + rnorm( nn, mean = 0, sd = sde)
   if (plt) {
      cat("n: ", nn, " a: ", aa, " b: ", bb, "\n")
      plot( x, y)
   }
   fit  <- lm( y ~ x)
   sfit <- summary(fit)
   c( n=nn, a=aa, b=bb, coef(fit)['(Intercept)'], coef(fit)['x'],
      Sb=sfit$coefficients['x', 'Std. Error'], r=cov(x, y))
}
```

Here is an example of one such call. The seed is set so that we can discuss the results.

```
set.seed(31394)
sbor( 100, 2, 3, plt=TRUE)
```

```
## n:  96  a:  2.836889  b:  3.863852
```




```
##           n           a           b (Intercept)           x          Sb
## 96.00000000  2.83688880  3.86385241  2.81628284  3.82773393  0.04940028
##           r
##  3.73908279
```

The above call generated 96 samples of data of an equation $y_i = 2.84 + 3.86x_i + \varepsilon_i$. A second call would generate a different n, a, and b, as well as different ε_i. The other thing that `sbor` does is return the coefficients estimated, the standard error of the slope, and the regression coefficient.

Below we can call `sbor` 10,000 times, each call generating a random number of points with a mean of 1,000, $\bar{a} = 2$ and $\bar{b} = 3$ and generate a data frame of the results.

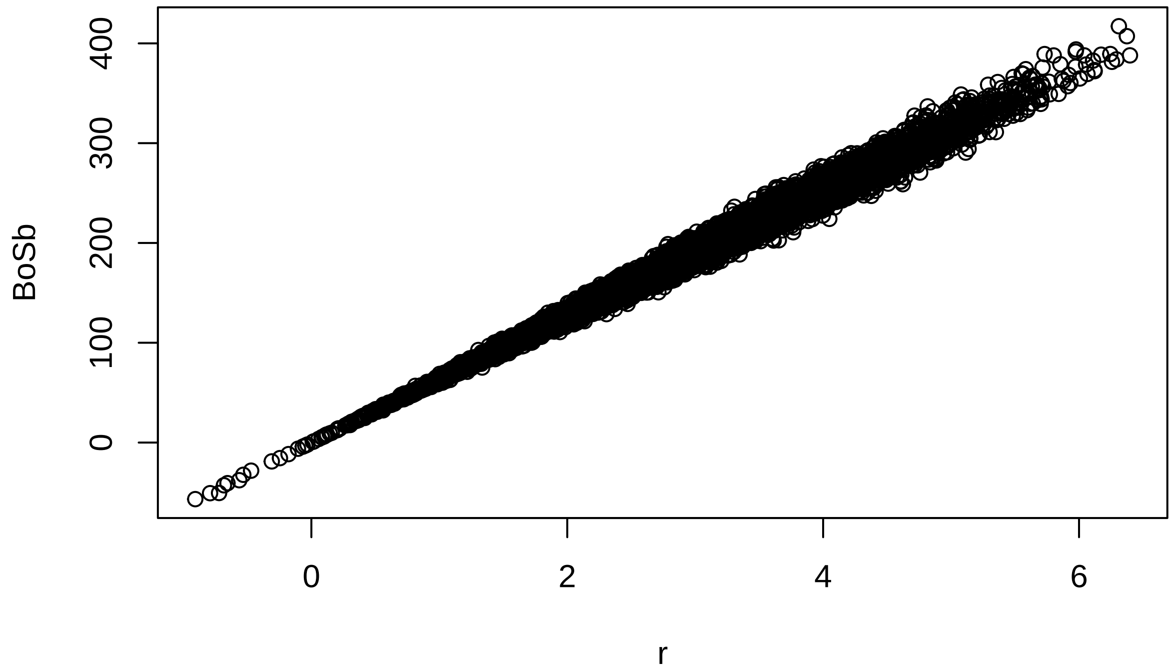
```
func <- function(i) c(trial=i, sbor( 1000, 2, 3))
# create a function of i for parLapply to call
require( parallel) # only load if needed
```

```
## Loading required package: parallel
```

```
cl <- parallel::makeCluster( 7)
parallel::clusterExport( cl, "sbor") # export the function.
parlist = parallel::parLapply( cl=cl, 1:10000, func)
parallel::stopCluster( cl)
newdf <- as.data.frame( t( simplify2array(( parlist))))
rm(parlist) #release this memory
```

The data shows a remarkable, although not one-to-one relationship between the ratio $\widehat{\beta}/S_{\widehat{\beta}}$ and r:

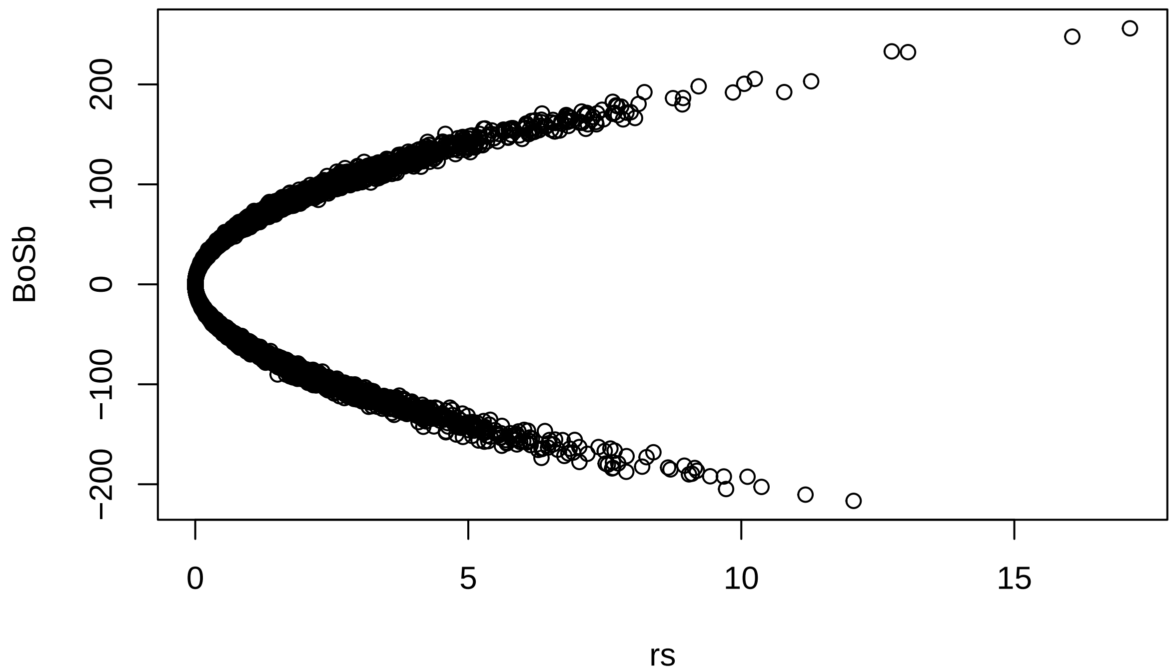
```
newdf$BoSb <- newdf$x / newdf$Sb
plot( BoSb ~ r, data=newdf)
```



Next we look at a situation closer to the first one, where the mean slope is zero.

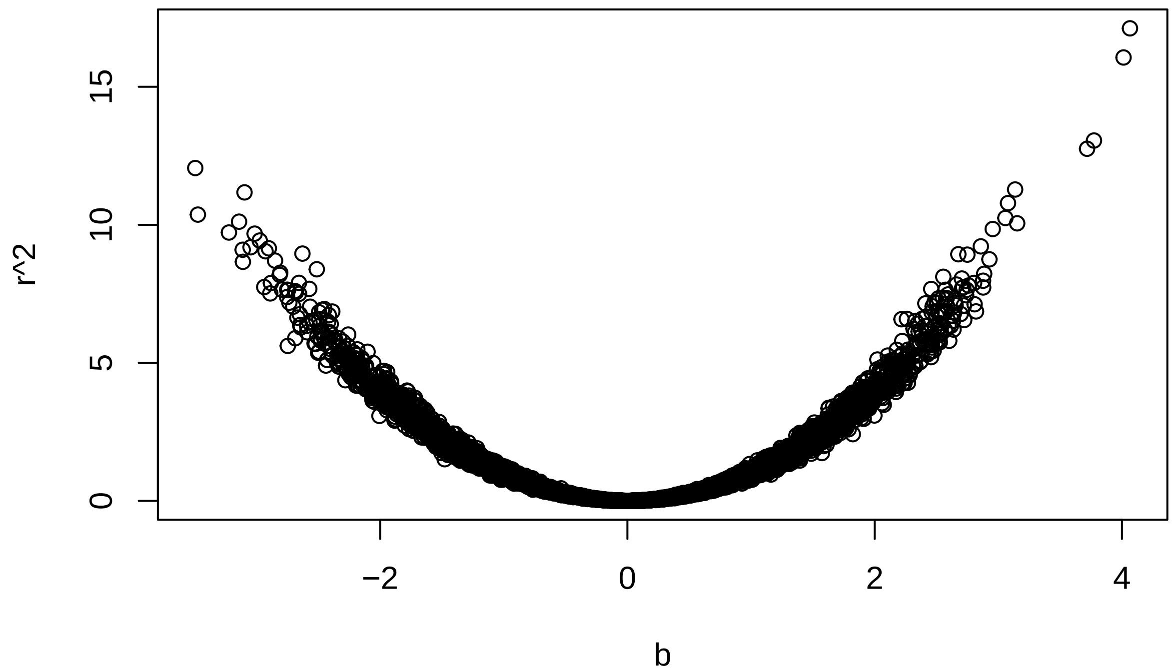
```
rm(newdf)
func <- function(i) c(trial=i, sbor( 1000, 2, 0))
require( parallel) # only load if needed
      cl <- parallel::makeCluster( 7)
      # registerDoParallel( cl)
      parallel::clusterExport( cl, "sbor")
      parlist = parallel::parLapply( cl=cl, 1:10000, func)
      parallel::stopCluster( cl)

   newdf <- as.data.frame( t( simplify2array(( parlist))))
   rm(parlist)
newdf$rs <- newdf$r^2
newdf$BoSb <- newdf$x / newdf$Sb
plot( BoSb ~ rs, data=newdf)
```

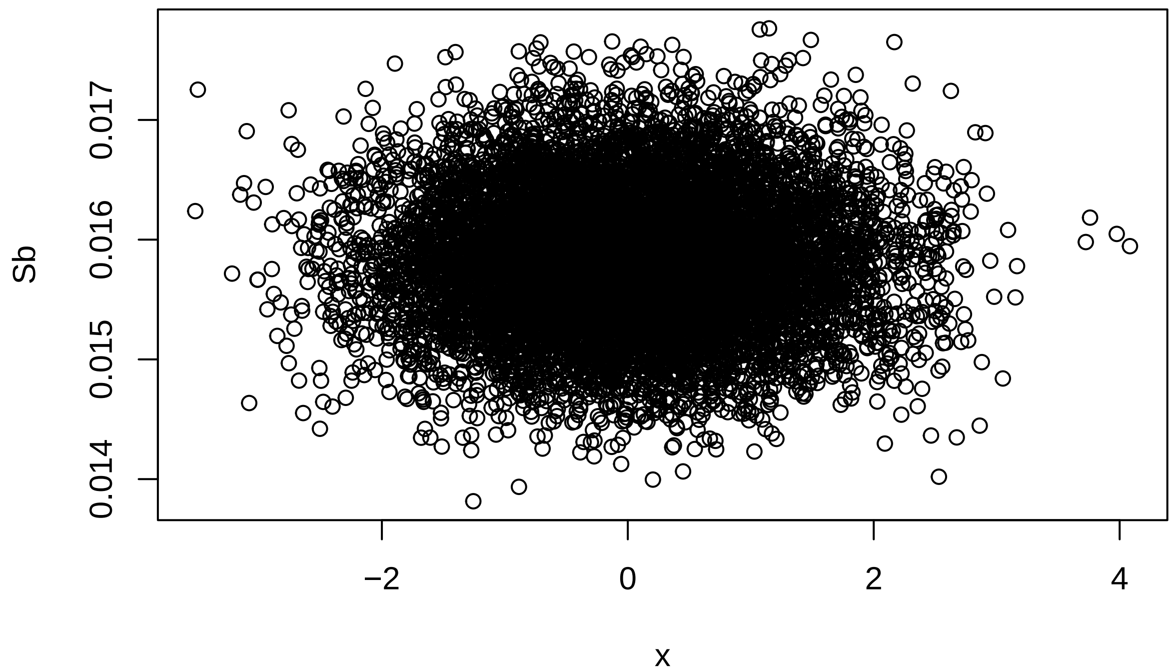


The graph below shows the slopes that were generated, and the resulting regression coefficients.

```
plot( r^2 ~ b, data=newdf)
```



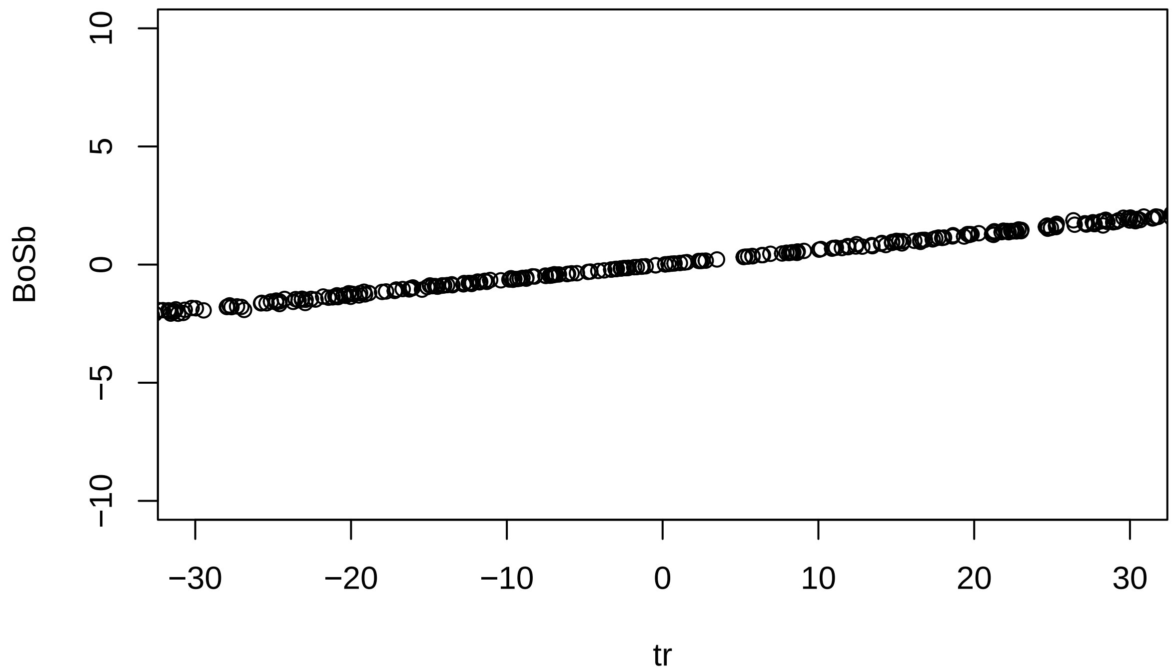

```
plot( Sb ~ x, data=newdf)
```




```
newdf$tr <- newdf$n * newdf$r / sqrt(1 - newdf$rs)
```

```
## Warning in sqrt(1 - newdf$rs): NaNs produced
```

```
plot( BoSb ~ tr, data=newdf, xlim=c(-30,30),ylim=c(-10,10))
```



The source of the parabola is the

Definitions

The equation for the estimate is (stolen from Wikipedia Simple Linear Regression):

$$\widehat{\beta} = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

The standard error of the estimate is:

$$s_{\widehat{\beta}} = \sqrt{\frac{\sum_{i=1}^{n} \widehat{\varepsilon}_i^2}{(n-1) \sum_{i=1}^{n} x_i^2}}$$

with $\widehat{\varepsilon}_i = y_i - \hat{y}_i$, So the numerator can be expanded as:

$$\sum_{i=1}^{n} \widehat{\varepsilon}_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i^2 - 2 y_i \hat{y}_i + \hat{y}_i^2)$$

Since $\hat{y}_i = \widehat{\alpha} + \widehat{\beta} x_i$ the above becomes

$$
\begin{aligned}
\sum_{i=1}^{n} \widehat{\varepsilon}_i^2 &= \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n} y_i(\widehat{\alpha} + \widehat{\beta}x_i) + \sum_{i=1}^{n} (\widehat{\alpha} + \widehat{\beta}x_i)^2 \\
&= \sum_{i=1}^{n} y_i^2 - 2\widehat{\alpha}\sum_{i=1}^{n} y_i + \widehat{\beta}\sum_{i=1}^{n} y_i x_i + n\widehat{\alpha}^2 + 2\widehat{\alpha}\widehat{\beta}\sum_{i=1}^{n} x_i + \widehat{\beta}^2 \sum_{i=1}^{n} x_i^2
\end{aligned}
$$

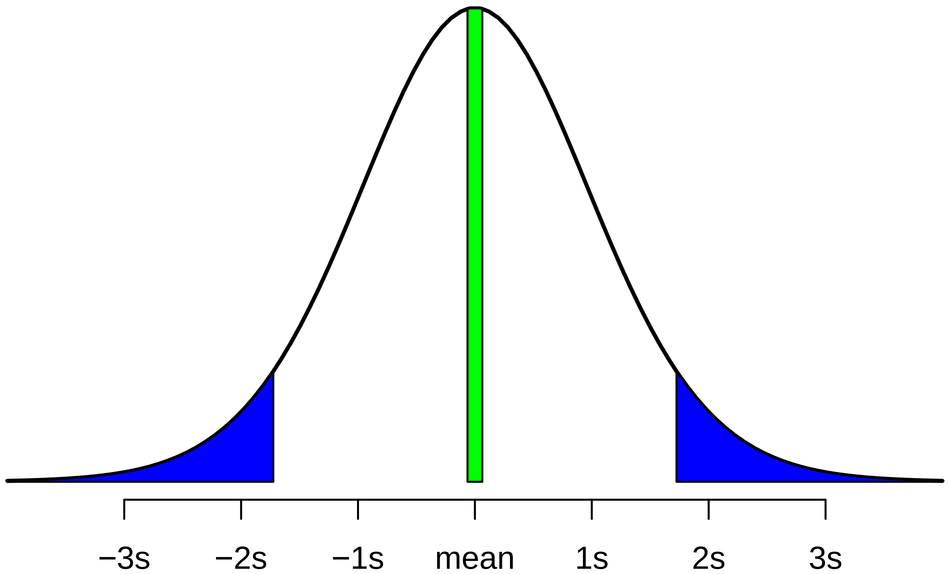
So.

The below from https://www.statology.org/working-with-the-student-t-distribution-in-r-dt-qt-pt-rt/

```r
#Create a sequence of 100 equally spaced numbers between -4 and 4
x <- seq(-4, 4, length=100)
p <- qt(0.05, 20)
p2 <- qt( .475, 20)
x <- sort( c( x, p, -p, p2, -p2))

#create a vector of values that shows the height of the probability distribution
#for each value in x, using 20 degrees of freedom
y <- dt(x = x, df = 20)

#plot x and y as a scatterplot with connected lines (type = "l") and add
#an x-axis with custom labels
plot(x, y, type = "l", lwd = 2, axes = FALSE, xlab = "", ylab = "")
axis(1, at = -3:3, labels = c("-3s", "-2s", "-1s", "mean", "1s", "2s", "3s"))
polygon( c(-4, x[x <= p], p), c( 0, y[ x <= p], 0), col='blue')
polygon( c(-p, x[x >= -p], 4), c( 0, y[ x >= -p], 0), col='blue')
l <- min( which( x >= p2))
h <- max( which( x <= -p2))
polygon( x[c( l, l:h, h)], c( 0, y[ l:h], 0), col='green')
```




```
pl <- recordPlot()
png("t20.png", width=480, height=240)
replayPlot(pl)
dev.off()
```

```
## pdf
##   2
```

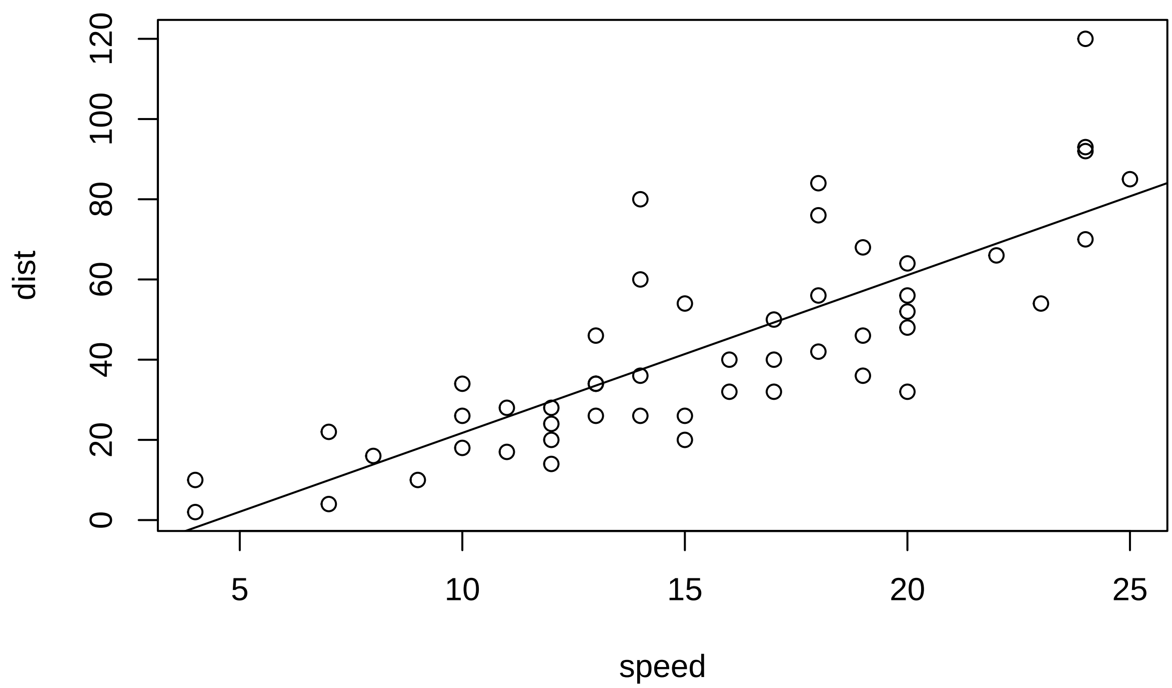
Replicating lm

```
carslm <- lm(dist ~ speed, data=cars )
(scarslm <- summary(carslm))
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

```r
plot(dist ~ speed, data=cars)
abline(coef( carslm))
```




```r
x <- cars$speed
y <- cars$dist
nr <- nrow( cars)
sumx  <- sum( x)
sumx2 <- sum( x^2)
sumy  <- sum (y)
sumy2 <- sum( y^2)
sumxy <- sum( x * y)
all.equal( mean( x), sumx / nr) # sumx == nr * mean(x)
```

```
## [1] TRUE
```

```
meanx <- mean( x)
all.equal( sumx, nr * meanx) # see?
```

```
## [1] TRUE
```

```
meany <- mean( y)
all.equal( var( x), sum( (x - meanx)^2) / (nr - 1))
```

```
## [1] TRUE
```

```
all.equal( var( x), (sumx2 - nr* meanx^2)  / (nr - 1)) # either way
```

```
## [1] TRUE
```

```
varx <- var(x)
vary <- var( y)
all.equal( sd( x), sqrt( varx))
```

```
## [1] TRUE
```

```
sdx <- sd( x)
sdy <- sd( y)
all.equal( cov( x, y), sum( (x - meanx)*(y - meany)) / (nr - 1))
```

```
## [1] TRUE
```

```
all.equal( cov( x, y), (sumxy - nr * meanx * meany) / (nr - 1))
```

```
## [1] TRUE
```

```
Sxy <- cov( x, y)
all.equal( cor(x, y), Sxy / (sdx * sdy))
```

```
## [1] TRUE
```

```
rxy <- cor(x, y)
b <- Sxy / sdx^2
all.equal( coef( carslm)['speed'], b,  check.attributes = FALSE)
```

```
## [1] TRUE
```

```
# Names from lm will cause the above to fail.
c( nr=nr, sumx=sumx, sumx2=sumx2, sumy=sumy, sumy2=sumy2, sumxy=sumxy,
   meanx=meanx, meany=meany, varx=varx, vary=vary, sdx=sdx, sdy=sdy,
   Sxy=Sxy, rxy=rxy, rxy2=rxy^2, b=b)
```

```
##            nr          sumx         sumx2          sumy         sumy2         sumxy
## 5.000000e+01 7.700000e+02 1.322800e+04 2.149000e+03 1.249030e+05 3.848200e+04
##        meanx         meany          varx          vary           sdx           sdy
## 1.540000e+01 4.298000e+01 2.795918e+01 6.640608e+02 5.287644e+00 2.576938e+01
##          Sxy           rxy          rxy2             b
## 1.099469e+02 8.068949e-01 6.510794e-01 3.932409e+00
```

The constant $\widehat{\alpha}$ is given simply by: $\hat{\bar{x}}$

$$\widehat{\alpha} = \bar{y} - \widehat{\beta}\bar{x}$$

```
a <- meany - b * meanx
c( a=a)
```

```
##         a
## -17.57909
```

```
ei <- y - a - b * x # this is a vector
sqrt( sum( ei^2) / ((nr - 2) * sum((x - meanx)^2) ))
```

```
## [1] 0.4155128
```

```
scarslm$coefficients['speed','Std. Error']
```

```
## [1] 0.4155128
```

```
all.equal( scarslm$coefficients['speed','Std. Error'],
           sqrt( sum( ei^2) / ((nr - 2) * sum((x - meanx)^2) ) ))
```

```
## [1] TRUE
```

```
Sb <- sqrt( sum( ei^2) / ((nr - 2) * (nr - 1) * varx))
all.equal( scarslm$coefficients['speed','Std. Error'], Sb)
```

```
## [1] TRUE
```