Lab 1

Ethan Ashby

1/30/2019

##Lab 1: Simple Linear Regression

Everything in the shaded regions above (called a 'chunk') is interpreted as R code. You can run separate chunks by pushing the 'play' button. We will use an R package called 'mosaic', which primarily makes functions doing standard descriptive statistics behave like statistical models in terms of syntax. It is trying to get away from having to use the '$' very often.

Let's go ahead and generate some regression data that comes from exactly the model we have in mind, that is $Y = \beta_0 + \beta_1 x + \epsilon$ In R, a function comprised of the letter 'r' and a distribution name (or part of one) generates a random sample from that distribution. Let's let x be random numbers between 0 and 10, with a sample size of $n = 47$.

```
n <- 47
x <- runif(n, 0, 10)
head(x)
```

```
## [1] 4.1091172 1.1909061 0.4530101 6.0886685 5.5362644 4.6194872
```

Now let's create a response variable. First, I'll choose the parameters of the model:

```
beta0 <- 2
beta1 <- 3
sigma <- 4
```

Now we can generate the data

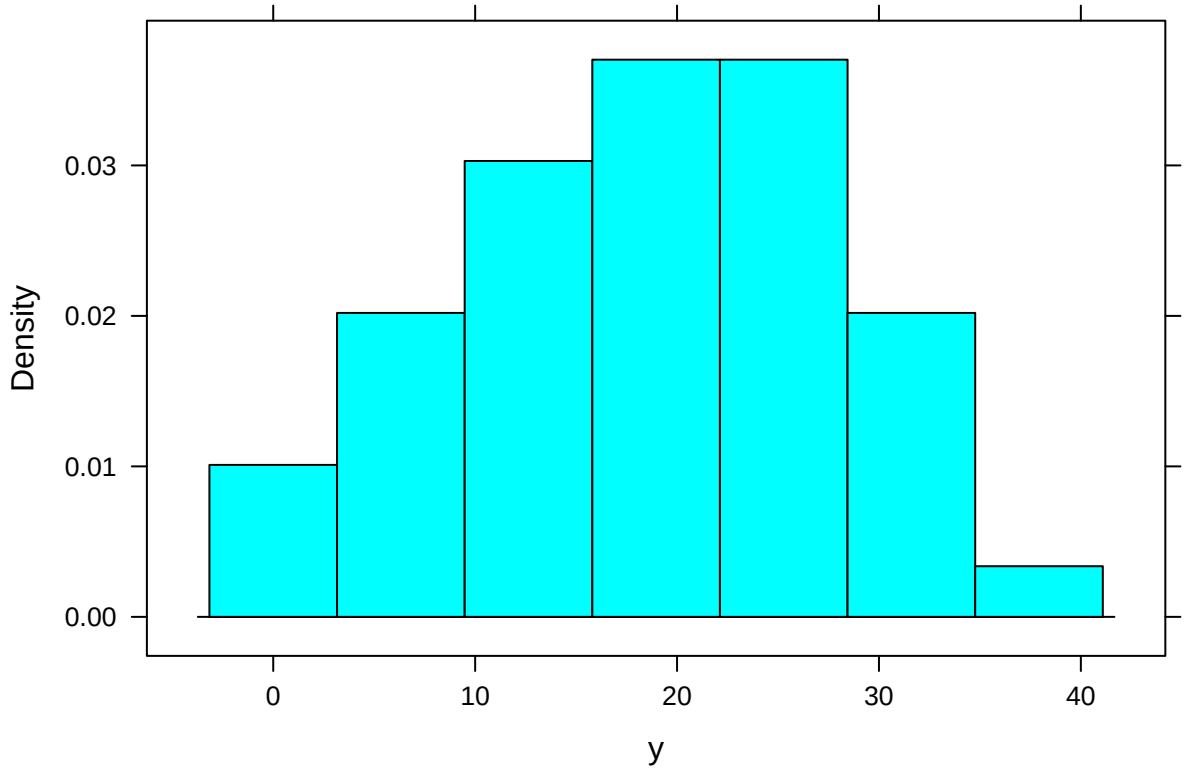
```
#generate normal error
epsilon <- rnorm(n,0,4)

#apply the model
y <- beta0 + beta1 * x + epsilon
data1 <- data.frame(x,y, epsilon)
head(data1)
```

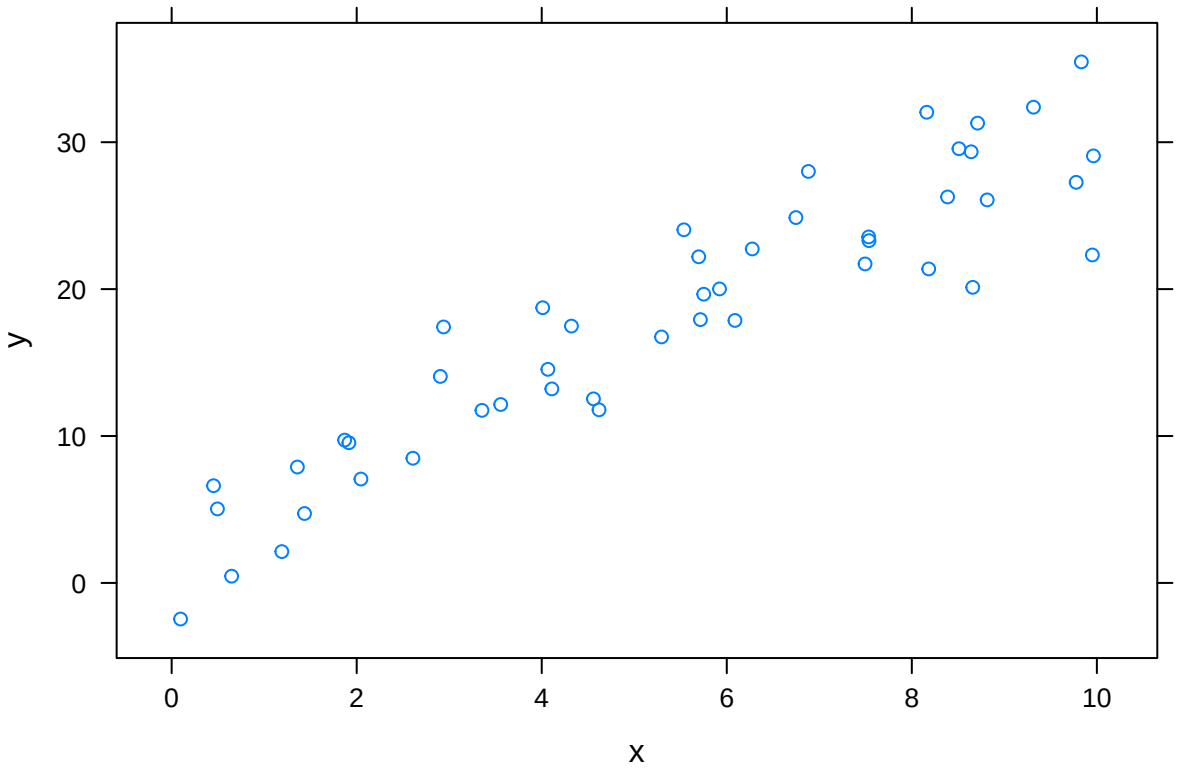
```
##           x         y   epsilon
## 1 4.1091172 13.207171 -1.120180
## 2 1.1909061  2.132656 -3.440062
## 3 0.4530101  6.618887  3.259857
## 4 6.0886685 17.866967 -2.399038
## 5 5.5362644 24.036730  5.427937
## 6 4.6194872 11.784882 -4.073580
```

Of course, in real life, we would never see the third column. Let's check some pictures.

```
histogram(~y, data=data1)
```

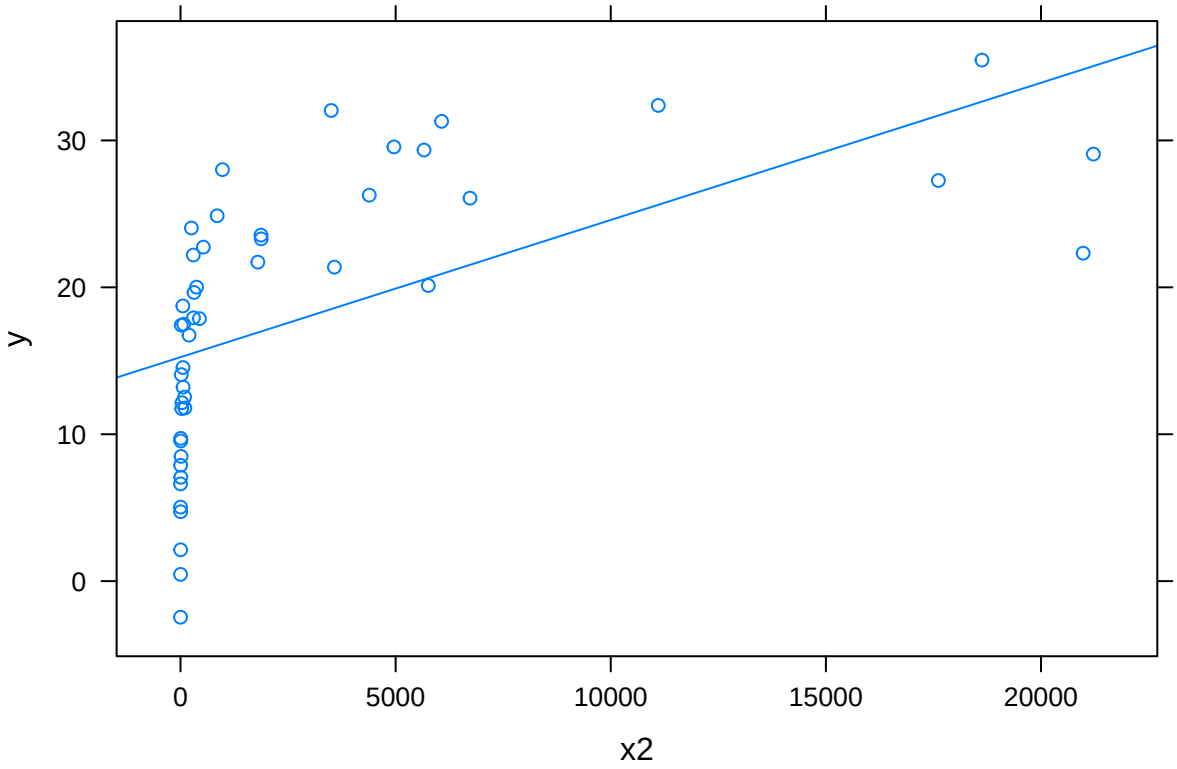



```
xyplot(y~x, data=data1, type='p')
```



We can also make some non-linear data by transforming one or both of the variables. We can add new variables to the data frame as such:

```
data1$x2 <- with(data1, exp(x))
xyplot(y~x2, data=data1, type=c('r', 'p'))
```



We can still fit a line to it! Heck, the correlation isn't even that bad.

```
cor(data1)
```

```
##                  x         y    epsilon         x2
## x        1.0000000 0.9265760 -0.1222311  0.6964295
## y        0.9265760 1.0000000  0.2600312  0.5728985
## epsilon -0.1222311 0.2600312  1.0000000 -0.2761703
## x2       0.6964295 0.5728985 -0.2761703  1.0000000
```

We know what transformation would fix this. log(x2)! In general, this is a much harder question.

The above plotted the regression line, but let's fit the model and see what the equation actually is.

```
fit <- lm(y~x, data=data1)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8386 -2.1164 -0.4178  2.5554  6.2987
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.7295     1.0590   2.577   0.0133 *
## x             2.8572     0.1729  16.526   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.514 on 45 degrees of freedom
## Multiple R-squared:  0.8585, Adjusted R-squared:  0.8554
## F-statistic: 273.1 on 1 and 45 DF,  p-value: < 2.2e-16
```

There's a bunch of stuff here, and we've only talked about a couple of them. r^2, and the $\hat{\beta}_i$ values (under Estimate). Still work to do!

The assumption for our model can be regarded as assumptions about the ϵ. The linearity assumption can be stated as: $\mu_{\epsilon|x} = 0$. The constant variance assumption is obviously: $\sigma^2_\epsilon = 0$. And the normality is: ϵ is distributed normally.

Let's add the residuals to our data frame (our best guess as the errors), as well as the fitted values.

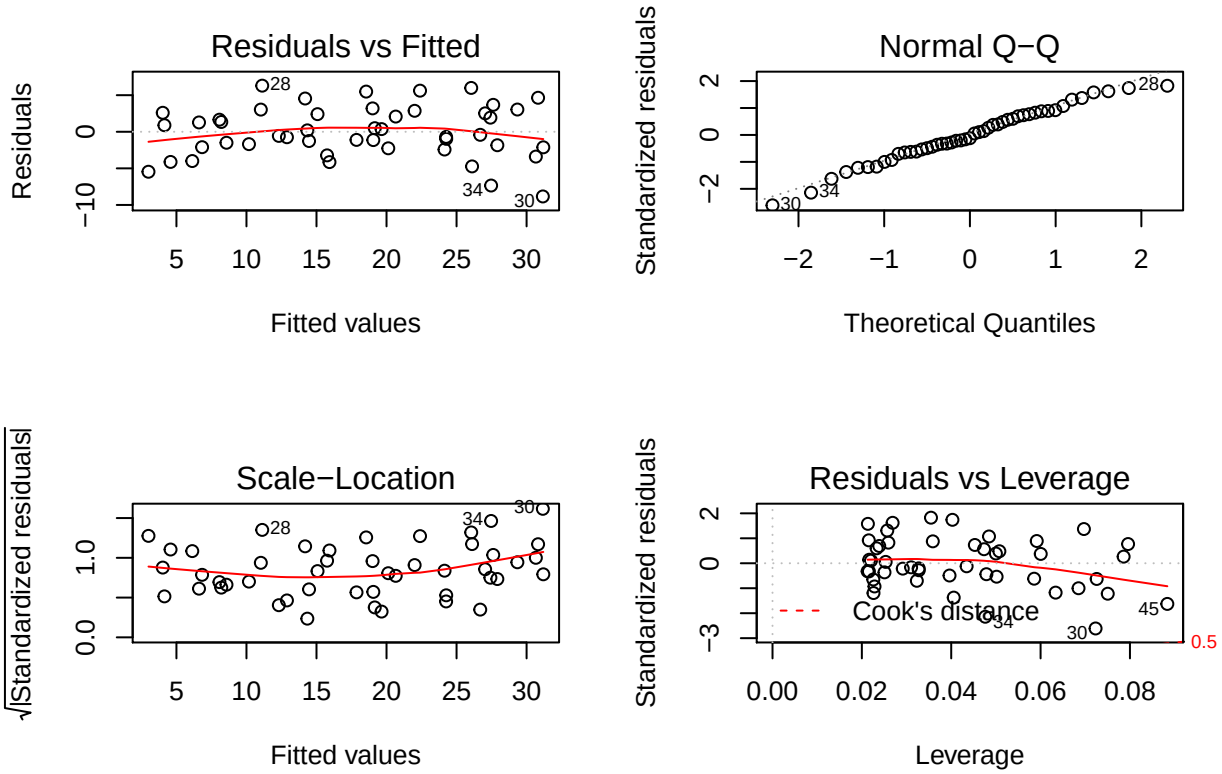
```
data1$resid <- residuals(fit)
data1$fitted <- fitted(fit)
head(data1)
```

```
##           x         y   epsilon         x2     resid    fitted
## 1 4.1091172 13.207171 -1.120180  60.892937 -1.262826 14.469997
## 2 1.1909061  2.132656 -3.440062   3.290061 -3.999515  6.132170
## 3 0.4530101  6.618887  3.259857   1.573040  2.595012  4.023876
## 4 6.0886685 17.866967 -2.399038 440.834052 -2.258946 20.125913
## 5 5.5362644 24.036730  5.427937 253.728403  5.489130 18.547600
## 6 4.6194872 11.784882 -4.073580 101.442003 -4.143330 15.928211
```

Think about how you might go about assessing these assumptions given these new variables.

```
#Checking linear model assumptions
#http://www.sthda.com/english/articles/39-regression-model-diagnostics/161-linear-regression-assumption
#THIS IS A GREAT RESOURCE

par(mfrow = c(2, 2))
plot(fit)
```



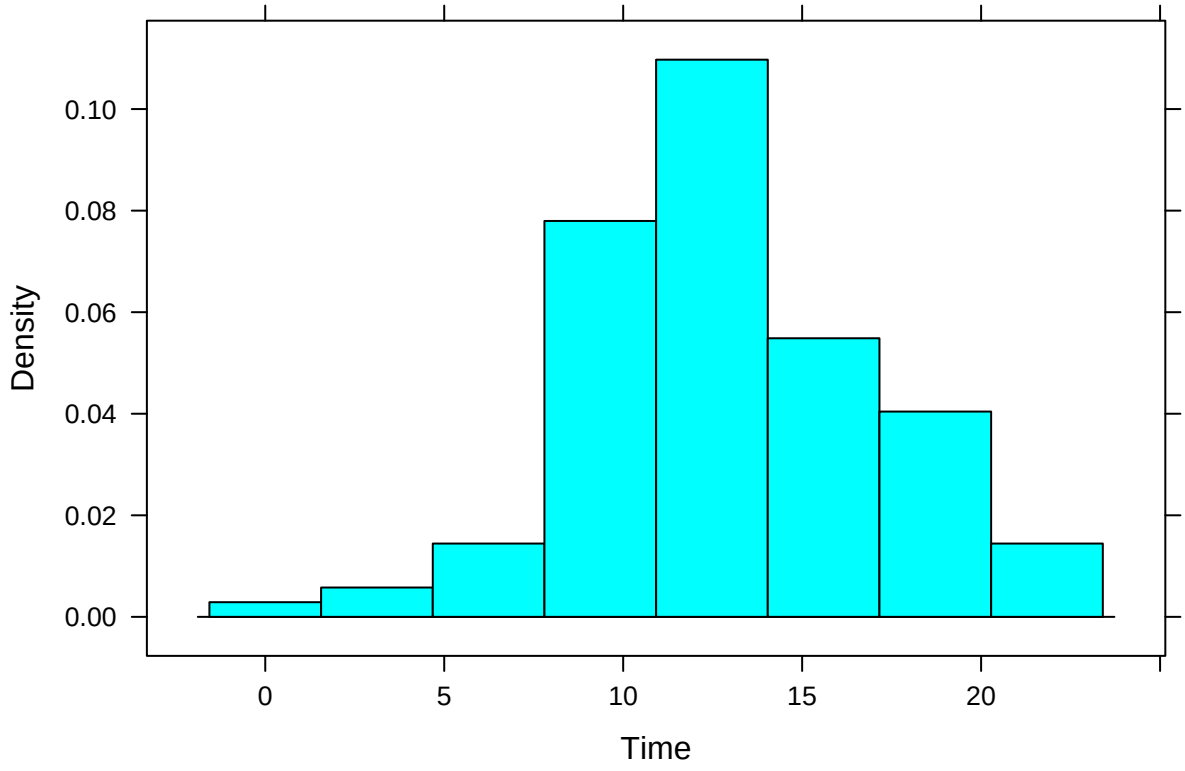
A horizontal residuals vs fitted plot demonstrates a good linear relationship. The Normal Q-Q plot that follows a straight line shows that the residuals are normally distributed. Scale-Location plot with horizontal line indicates homoskedasticity.

Now try to fit and interpret a model for the data on sakai. What would it mean for a linear model to be appropriate? What about a curve with decreasing slope? Increasing slope?

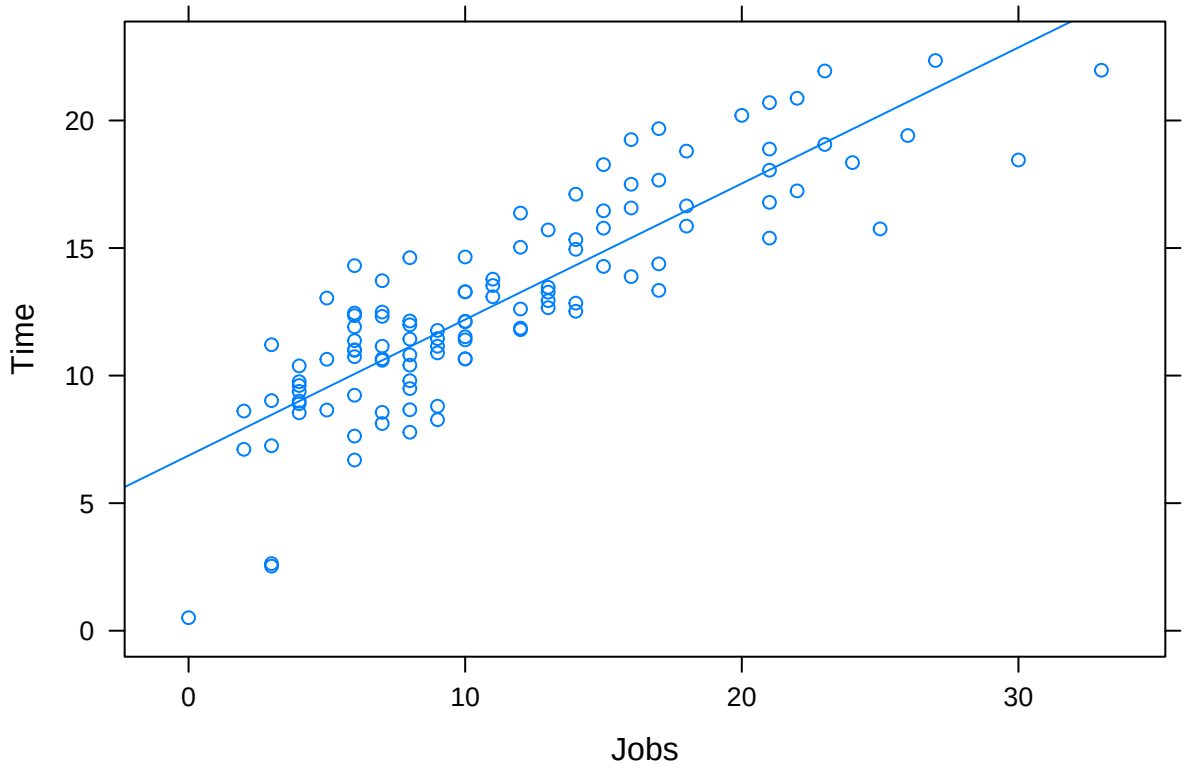
```
labdata<-read.csv("Lab_1_Data.txt")
summary(labdata)
```

```
##       Time            Jobs      
##  Min.   : 0.51   Min.   : 0.00  
##  1st Qu.:10.62   1st Qu.: 6.00  
##  Median :12.45   Median :10.00  
##  Mean   :12.93   Mean   :11.37  
##  3rd Qu.:15.73   3rd Qu.:15.00  
##  Max.   :22.35   Max.   :33.00
```

```
#some diagnostic plots
histogram(~Time, data=labdata)
```



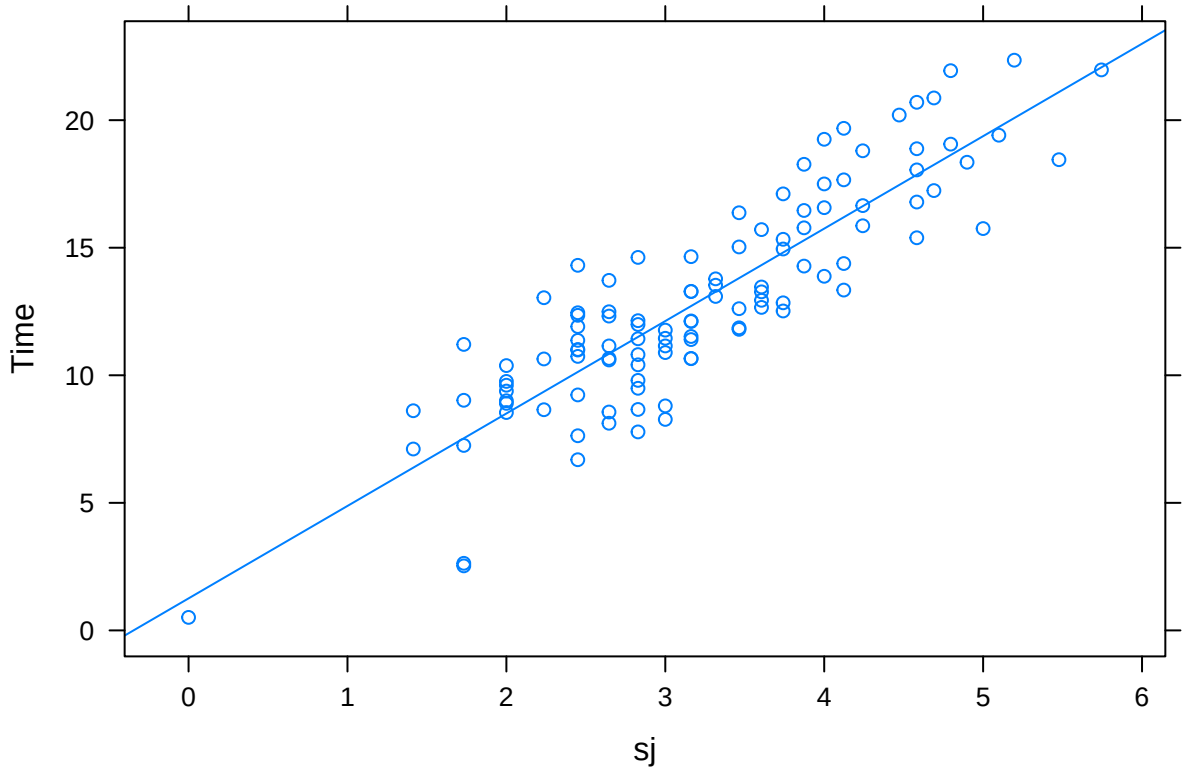

```
xyplot(Time~Jobs, data=labdata, type=c('r', 'p'))
```




```
#given some context, it shouldn't take us 7 minutes to do 0 jobs... can we do better?

labdata$sj <- with(labdata, sqrt(Jobs))
```

```
xyplot(Time~sj, data=labdata, type=c('r', 'p'))
```




```
#the sqrt transformation on 'Jobs' improves the linear fit a lot better! Yay!
```