Hmwk #5 - Regression Analysis

Problem #1 - Trees

Foresters want to measure the volume of a tree in order to estimate how much lumber they would get for that tree. Knowing the girth and height of the tree, one could apply a cylinder approximation, but the tree does not have a uniform girth along its entire length -- the tree gets much smaller at the top. The UsingR library has a data set "trees", that contains a set of measurements of the girth and height of a tree. Use the library to assess a linear model. For example, you might try:

In [76]:
```
library(UsingR)
summary(trees)
```

Out[76]:
```
     Girth           Height       Volume     
 Min.   : 8.30   Min.   :63   Min.   :10.20  
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40  
 Median :12.90   Median :76   Median :24.20  
 Mean   :13.25   Mean   :76   Mean   :30.17  
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30  
 Max.   :20.60   Max.   :87   Max.   :77.00  
```

If we look at the regression model

In [77]:
```
m = lm(Volume ~ Height + Girth, data=trees)
summary(m)
```

Out[77]:
```
Call:
lm(formula = Volume ~ Height + Girth, data = trees)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.4065 -2.6493 -0.2876  2.2003  8.4847 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
Height        0.3393     0.1302   2.607   0.0145 *  
Girth         4.7082     0.2643  17.816  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948,     Adjusted R-squared:  0.9442 
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

This provides a model, but doesn't say if it's any good. Do the following using R -- i.e. use the output of the 'lm' linear model where possible. Show your work.

What is the R^2 for this model

In [78]:
```
print("R2 value = ")
summary(m)$r.squared
```

```
[1] "R2 value = "
```

Out[78]:
0.947950037781675

Compute the 95% confidence interval for the Height parameter

(**nb:** you're being asked to calculate the CI of the parameter from the multi-linear model, not the CI of the Height data itself). To do this, you would use the standard error (shown in the summary of the linear model -- e.g. 0.1302 for parameter Height) for each parameter -- this is the standard error for that parameter derived from the MSE as described in the slides concerning multi-linear regression. You would then conduct the T-distribution using that standard error with n-k-1 degrees of freedom. See the slides on linear regression.

In [79]:
```
n <- length(trees$Height)
t_val <- qt(.975, n-1)
h_est <- 0.3393
h_std <- 0.1302
#estimate +- t*std
print("95 CI for height +- ")
(h_est + t_val*h_std)
(h_est - t_val*h_std)
```

```
[1] "95 CI for height +- "
```

Out[79]: 0.605203873810421

Out[79]: 0.0733961261895788

Compute the 95% confidence interval for the Girth parameter

(**nb:** as above). Verify your value using the **confint** function in R.

In [80]:
```
n <- length(trees$Height)
t_val <- qt(.975, n-1)
g_est <- 4.7082
g_std <- 0.2643
#estimate +- t*std
print("95 CI for girth +- ")
(g_est + t_val*g_std)
(g_est - t_val*g_std)

#Check work
confint(m, ,.95)
```

```
[1] "95 CI for girth +- "
```

Out[80]: 5.24797261020042

Out[80]: 4.16842738979958

Out[80]:

	2.5 %	**97.5 %**
(Intercept)	-75.68226	-40.29306
Height	0.07264863	0.60585384
Girth	4.166839	5.249482

Does the Height parameter statistically equal zero? Does the Girth?

In [81]:
```
print("The tvalues for height and girth are 2.6 and 17.8. t with 28dof is 2.05.")
print("Thus neither are statistically equal to zero.")
```

```
[1] "The tvalues for height and girth are 2.6 and 17.8. t with 28dof is 2.05."
[1] "Thus neither are statistically equal to zero."
```

Complete this sentence:

"For every unit increase in Girth, the Volume increases by __ and for every unit increase in Height, the Volume increases by __"

In [82]:
```
(m)
print("For every unit in Girth, Volume increases by :")
(4.7082)
print("for every unit in Height, the Volume increases by :")
(0.3393)
```

Out[82]:
```
Call:
lm(formula = Volume ~ Height + Girth, data = trees)

Coefficients:
(Intercept)       Height        Girth
   -57.9877       0.3393       4.7082

[1] "For every unit in Girth, Volume increases by :"
```

Out[82]: 4.7082

```
[1] "for every unit in Height, the Volume increases by :"
```

Out[82]: 0.3393

Problem #2 - Body Fat

In [83]:
```
summary(fat)
```

Out[83]:
```
      case           body.fat     body.fat.siri      density
 Min.   :  1.00   Min.   : 0.00   Min.   : 0.00   Min.   :0.995
 1st Qu.: 63.75   1st Qu.:12.80   1st Qu.:12.47   1st Qu.:1.041
 Median :126.50   Median :19.00   Median :19.20   Median :1.055
 Mean   :126.50   Mean   :18.94   Mean   :19.15   Mean   :1.056
 3rd Qu.:189.25   3rd Qu.:24.60   3rd Qu.:25.30   3rd Qu.:1.070
 Max.   :252.00   Max.   :45.10   Max.   :47.50   Max.   :1.109
      age            weight          height           BMI
 Min.   :22.00   Min.   :118.5   Min.   :29.50   Min.   :18.10
 1st Qu.:35.75   1st Qu.:159.0   1st Qu.:68.25   1st Qu.:23.10
 Median :43.00   Median :176.5   Median :70.00   Median :25.05
 Mean   :44.88   Mean   :178.9   Mean   :70.15   Mean   :25.44
 3rd Qu.:54.00   3rd Qu.:197.0   3rd Qu.:72.25   3rd Qu.:27.32
 Max.   :81.00   Max.   :363.1   Max.   :77.75   Max.   :48.90
    ffweight          neck           chest           abdomen
 Min.   :105.9   Min.   :31.10   Min.   : 79.30   Min.   : 69.40
 1st Qu.:131.3   1st Qu.:36.40   1st Qu.: 94.35   1st Qu.: 84.58
 Median :141.6   Median :38.00   Median : 99.65   Median : 90.95
 Mean   :143.7   Mean   :37.99   Mean   :100.82   Mean   : 92.56
 3rd Qu.:153.9   3rd Qu.:39.42   3rd Qu.:105.38   3rd Qu.: 99.33
 Max.   :240.5   Max.   :51.20   Max.   :136.20   Max.   :148.10
      hip            thigh            knee           ankle          bicep
 Min.   : 85.0   Min.   :47.20   Min.   :33.00   Min.   :19.1   Min.   :24.80
 1st Qu.: 95.5   1st Qu.:56.00   1st Qu.:36.98   1st Qu.:22.0   1st Qu.:30.20
 Median : 99.3   Median :59.00   Median :38.50   Median :22.8   Median :32.05
 Mean   : 99.9   Mean   :59.41   Mean   :38.59   Mean   :23.1   Mean   :32.27
 3rd Qu.:103.5   3rd Qu.:62.35   3rd Qu.:39.92   3rd Qu.:24.0   3rd Qu.:34.33
 Max.   :147.7   Max.   :87.30   Max.   :49.10   Max.   :33.9   Max.   :45.00
    forearm          wrist
 Min.   :21.00   Min.   :15.80
 1st Qu.:27.30   1st Qu.:17.60
 Median :28.70   Median :18.30
 Mean   :28.66   Mean   :18.23
 3rd Qu.:30.00   3rd Qu.:18.80
 Max.   :34.90   Max.   :21.40
```

The Body Mass Index (BMI) is a model to predict the percentage of body fat based on your weight and height. The VMI is defined as the ratio of weight (in kilograms) to the square of height (in metres). A BMI of 18.5 to 25 is considered "healthy", a BMI of 25 to 30 is "overweight" and a BMI over 30 is "obese". The dataset 'fat' from UsingR contains 19 factors. The true body fat is body.fat and BMI field is the BMI.

Using the 'fat' data set, build a linear model

of the body.fat predicted by BMI. Describe the linear model -- what are the intercept and slope. What is the r^2 of that model?

In [84]:
```
(fatty.lm=lm(body.fat~BMI,data=fat))
summary(fatty.lm)$r.squared
```

Out[84]:
```
Call:
lm(formula = body.fat ~ BMI, data = fat)

Coefficients:
(Intercept)          BMI
    -20.405        1.547
```

Out[84]:
```
0.529975533278187
```

Come up with a minimal model that predicts body.fat using the other factors with an R2 of at least 0.72

We will use these different factors to come up with a minimal model that predicts body.fat using the other factors with an R2 of at least 0.72. Each factor in the final model should be significant at the 95% level -- this means that the confidence interval of the parameter at the 95% confidence level should not include zero.

Include your description showing both your final result and the process by which you achieve that result. Your model shouldn't use the "density" or "body.fat.siri" factors, as those are the "gold standard" measurements used to calculate body fat (using a dunk tank). I'm not certain what "ffweight" is, but don't use that either.

You may want to understand the "p-value" for problem - basically, the p-value is the probability of observing a value at least as large as the "t-value" (which is the estimate / std. error). Effectively, this is providing a way to determine if that value is statistically equal to zero.

You should describe the process you use, not just the end result. You should also insure that the linear model is valid, meaning that the predictors are not correlated, etc. Wikipedia has a good article on such "stepwise refinement" mechanisms [http://en.wikipedia.org/wiki/Stepwise_regression (http://en.wikipedia.org/wiki/Stepwise_regression)]. See also the discussion on page 125 of the book by Faraway on Practical Regression Analysis using R (on the couse web page or http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf (http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf)). They describe a "backward" and "forward" mechanism where you either start with all terms & toss some away or start adding terms to a null model.

In [85]:
```
fatty.lm=lm(body.fat~.,data=fat)

#My process is to remove the data sources with a high p value (approx >5%),
#starting with the highest p-values first. I remove them 1 at a time to account for
#dependencies between parameters.

#Remove density, siri, and ffweight as per instructions
#Remove case as the case number shouldn't be important with a valid study
#summary(fatty.lm)$r.squared
fatty.lm=update(fatty.lm, . ~ . - density)
#summary(fatty.lm)$r.squared
fatty.lm=update(fatty.lm, . ~ . - body.fat.siri)
#summary(fatty.lm)$r.squared
fatty.lm=update(fatty.lm, . ~ . - ffweight)
#summary(fatty.lm)$r.squared
fatty.lm=update(fatty.lm, . ~ . - case)
#summary(fatty.lm)

#Remove knee as it has highest P-value
fatty.lm=update(fatty.lm, . ~ . - knee)
#summary(fatty.lm)

#Remove BMI as it has highest P-value
fatty.lm=update(fatty.lm, . ~ . - BMI)
#summary(fatty.lm)

#Remove Chest as it has highest P-value
fatty.lm=update(fatty.lm, . ~ . - chest)
#summary(fatty.lm)

#Remove height
fatty.lm=update(fatty.lm, . ~ . - height)
#summary(fatty.lm)

#Remove ankle
fatty.lm=update(fatty.lm, . ~ . - ankle)
#summary(fatty.lm)

#Remove bicep
fatty.lm=update(fatty.lm, . ~ . - bicep)
#summary(fatty.lm)

#Remove hip
fatty.lm=update(fatty.lm, . ~ . - hip)
#summary(fatty.lm)

#Remove neck
fatty.lm=update(fatty.lm, . ~ . - neck)
#summary(fatty.lm)

#Remove thigh
fatty.lm=update(fatty.lm, . ~ . - thigh)
#summary(fatty.lm)

#Remove age
fatty.lm=update(fatty.lm, . ~ . - age)
#summary(fatty.lm)

#Remove forearm
fatty.lm=update(fatty.lm, . ~ . - forearm)
summary(fatty.lm)

#find correlations between remaining factors and body fat
writeLines("\nCorrelations between fat-weight, fat-abdoment, fat-wrist:")
cor(fat$body.fat, fat$weight)
cor(fat$body.fat, fat$abdomen)
cor(fat$body.fat, fat$wrist)
```

Out[85]:
```
Call:
lm(formula = body.fat ~ weight + abdomen + wrist, data = fat)

Residuals:
     Min       1Q   Median       3Q      Max
-12.1509  -2.9833  -0.1719   2.9955   9.1195

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -24.76129    6.31599  -3.920 0.000114 ***
weight       -0.10558    0.02191  -4.820 2.51e-06 ***
abdomen       0.90191    0.05202  17.338  < 2e-16 ***
```

```
wrist        -1.14570    0.40412  -2.835 0.004959 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.07 on 248 degrees of freedom
Multiple R-squared:  0.7276,    Adjusted R-squared:  0.7243
F-statistic: 220.8 on 3 and 248 DF,  p-value: < 2.2e-16

Correlations between fat-weight, fat-abdoment, fat-wrist:
```

Out[85]: 0.613156110031314

Out[85]: 0.813706221642791

Out[85]: 0.347572756593503

Problem #3 - Wireless networks

This data contains measurements from 3 wireless network devices. One is an 802.11 "Wifi" interface running at 1Mbs/s. The second is that same interface running at 11Mb/s. Lastly, the same interface running at 54Mb/s. The data is stored as R vectors named x1, y1, x11, y11 and x54, y54. The X value is the packet size for the time measurement recorded at the corresponding Y value. The units are milliseconds. For example, x11[1] is 350 bytes and y11[1] is 1.436782 millseconds. It took 1.436782 milliseconds to transmit a 350 byte packet.

In [86]:
```
x11 = c(350, 350, 350, 350, 350, 450, 450, 450, 450, 450, 550, 550, 550,
550, 550, 650, 650, 650, 650, 650, 750, 750, 750, 750, 750, 850, 850,
850, 850, 850, 950, 950, 950, 950, 950, 1050, 1050, 1050, 1050, 1050,
1150, 1150, 1150, 1150, 1150, 1250, 1250, 1250, 1250, 1250, 1350,
1350, 1350, 1350, 1350, 1450, 1450, 1450, 1450, 1450 )

y11 = c(
1.436782, 1.407063, 1.436782, 1.426737, 1.416832, 1.50015, 1.533978,
1.50015, 1.522533, 1.511259, 1.619433, 1.619433, 1.576541, 1.576541,
1.587050, 1.682935, 1.662787, 1.693193, 1.682935, 1.693193, 1.745505,
1.755310, 1.765225, 1.755310, 1.755002, 1.833853, 1.805054, 1.824152,
1.853568, 1.853568, 1.915342, 1.915342, 1.915342, 1.904762, 1.915342,
1.992429, 1.982161, 1.992429, 1.961169, 1.971998, 2.072539, 2.052124,
2.062281, 2.062281, 2.082466, 2.15378, 2.143623, 2.133561, 2.113718,
2.103934, 2.182453, 2.182453, 2.182453, 2.154244, 2.182453, 2.241147,
2.26142, 2.241147, 2.301496, 2.282063)

x1 = c( 350, 350, 350, 350, 350, 450, 450, 450, 450, 450, 550, 550,
550, 550, 550, 650, 650, 650, 650, 650, 750, 750, 750, 750, 750, 850,
850, 850, 850, 850, 950, 950, 950, 950, 950, 1050, 1050, 1050, 1050,
1050, 1150, 1150, 1150, 1150, 1150, 1250, 1250, 1250, 1250, 1250,
1350, 1350, 1350, 1350, 1350, 1450, 1450, 1450, 1450, 1450)

y1 = c( 4.161465, 4.078303, 4.078303, 4.078303, 4.078303, 4.741584,
4.741584, 4.741584, 4.741584, 4.741584, 5.534034, 5.534034, 5.534034,
5.534034, 5.534034, 6.30517, 6.30517, 6.30517, 6.30517, 6.30517,
7.097232, 7.097232, 7.097232, 6.939625, 7.097232, 7.830854, 7.830854,
7.830854, 7.830854, 7.830854, 8.403361, 8.403361, 8.403361, 8.403361,
8.403361, 9.14913, 9.14913, 9.14913, 9.14913, 9.14913, 9.910803,
9.910803, 9.910803, 9.910803, 9.910803, 10.81081, 10.81081, 10.55966,
10.81081, 10.81081, 11.60093, 11.60093, 11.33787, 11.33787, 11.33787,
12.16545, 12.16545, 12.16545, 12.16545, 12.16545)

x54 = c( 350, 350, 350, 350, 450, 450, 450, 450, 550, 550, 550, 550,
650, 650, 650, 650, 750, 750, 750, 750, 850, 850, 850, 850, 950, 950,
950, 950, 1050, 1050, 1050, 1050, 1150, 1150, 1150, 1150, 1250, 1250,
1250, 1250, 1350, 1350, 1350, 1350, 1450, 1450, 1450, 1450)

y54 = c( 0.2812386, 0.2804341, 0.2798769, 0.2815553, 0.2995088,
0.2986679, 0.3006705, 0.298454, 0.311886, 0.3199386, 0.3163556,
0.3186439, 0.3333333, 0.3369953, 0.3340236, 0.3311258, 0.3412969,
0.3373933, 0.341006, 0.3411456, 0.3531198, 0.3563284, 0.3578714,
0.3521871, 0.3733154, 0.3752768, 0.3761803, 0.3780432, 0.3957888,
0.3914660, 0.3928656, 0.3961651, 0.4116921, 0.4088307, 0.4083966,
0.4038935, 0.4210704, 0.4251339, 0.4259488, 0.4252243, 0.4428698,
0.4405869, 0.4386157, 0.4402959, 0.4577287, 0.4561211, 0.4573938,
0.4607658)
```

Answer the following questions:

To answer these questions, you should use the lecture slides on linear regression and section 13 from the SimpleR.pdf book from the course website (page 77). You can also refer to the material by Boudec, but his write-up is, as per normal, complex.

- Using R as a calculator, but not using the built-in regression functions, calculate using the equations in the Linear Regression slides:
- The slope (b) and intercept (a) of the regression model for the 1Mb/s network (using x1, y1)
- The coefficient of determination for the model for the 1Mb/s network (using x1,y1)
- The standard deviation for slope & intercept and the 95% confidence interval for the 1Mb/s network (using x1,y1)
- The predicted time, standard deviation and 95% confidence interval for the predicted time to transmit a 40 byte Wifi packet for the 1Mb/s network (using x1,y1)
- The predicted time, standard deviation and 95% confidence interval to transmit a 750 byte packet for the 1Mb/s network (using x1,y1)

In [87]:

```
#Pre-Calculations
x1_sum <- sum(x1)
y1_sum <- sum(y1)
x1y1_sum <- sum(x1*y1)
x1_sq_sum <- sum(x1*x1)
y1_sq_sum <- sum(y1*y1)
N <- length(x1)

#Slope
print("Slope (b):")
(b <- ((N*x1y1_sum - x1_sum*y1_sum) /
         (N*x1_sq_sum - x1_sum*x1_sum)) )

#Intercept
print("Intercept (a):")
(a <- ((x1_sq_sum*y1_sum - x1_sum*x1y1_sum) /
         (N*x1_sq_sum - x1_sum*x1_sum)) )

#Coefficient of Determiniation
print("Coefficient of Determination (R2) :")
(R2 <- ((N*x1y1_sum-x1_sum*y1_sum)^2) /
((N*x1_sq_sum-x1_sum*x1_sum)*(N*y1_sq_sum-y1_sum*y1_sum)) )
writeLines("\n")

#Check work
#summary(lm(y1~x1))

# MSE=SSE/dof,  dof=n-2
# se=sqrt(MSE)
Sxx = sum( (x1-mean(x1))^2 )
Sxy = sum( (x1-mean(x1))*(y1-mean(y1)) )
Syy = sum( (y1-mean(y1))^2 )
b2 <- Sxy/Sxx
SSE <- Syy - b2*Sxy

MSE <- SSE/(length(x1)-2)
se <- (MSE)^(0.5)

writeLines("\n")
print("Std(b), Std(a):")
(std_b <- se/sqrt(Sxx))
(std_a <- se*sqrt(sum(x1^2))/(sqrt(N*Sxx)))

writeLines("\n")
print("95 Conf interval b +-")
(b+qt(.975, N-1)*std_b)
(b-qt(.975, N-1)*std_b)


print("95 Conf interval a +-")
(a+qt(.975, N-1)*std_a)
(a-qt(.975, N-1)*std_a)


writeLines("\n")
print("Predicted value y_p 40")
(y_p = b*40+a)

print("Confidence interval for y_p 40 +-")
(y_p + qt(.975, N-1)*se*sqrt(1+1/N+((40-mean(x1))^2)/Sxx))
(y_p - qt(.975, N-1)*se*sqrt(1+1/N+((40-mean(x1))^2)/Sxx))

print("Std for y_p 40")
se*sqrt(1+1/N+((40-mean(x1))^2)/Sxx)


writeLines("\n")
print("Predicted value y_p 750")
(y_p = b*750+a)

print("Confidence interval for y_p 750 +-")
(y_p + qt(.975, N-1)*se*sqrt(1+1/N+((750-mean(x1))^2)/Sxx))
(y_p - qt(.975, N-1)*se*sqrt(1+1/N+((750-mean(x1))^2)/Sxx))

print("Std for y_p 750")
se*sqrt(1+1/N+((750-mean(x1))^2)/Sxx)
```

```
[1] "Slope (b):"
```

Out[87]: 0.00736104904895105

```
[1] "Intercept (a):"
```

Out[87]: 1.49211468927739

```
[1] "Coefficient of Determination (R2) :"
```

Out[87]: 0.999155065938872

```
[1] "Std(b), Std(a):"
```

Out[87]: 2.81074001391503e-05

Out[87]: 0.0270936481790362

```
[1] "95 Conf interval b +-"
```

Out[87]: 0.00741729182671956

Out[87]: 0.00730480627118253

```
[1] "95 Conf interval a +-"
```

Out[87]: 1.54632895405919

Out[87]: 1.43790042449559

```
[1] "Predicted value y_p 40"
```

Out[87]: 1.78655665123543

```
[1] "Confidence interval for y_p 40 +-"
```

Out[87]: 1.94572238739662

Out[87]: 1.62739091507425

```
[1] "Std for y_p 40"
```

Out[87]: 0.0795432802614719

```
[1] "Predicted value y_p 750"
```

Out[87]: 7.01290147599068

```
[1] "Confidence interval for y_p 750 +-"
```

Out[87]: 7.16477433385565

Out[87]: 6.8610286181257

```
[1] "Std for y_p 750"
```

Out[87]: 0.0758986550034274

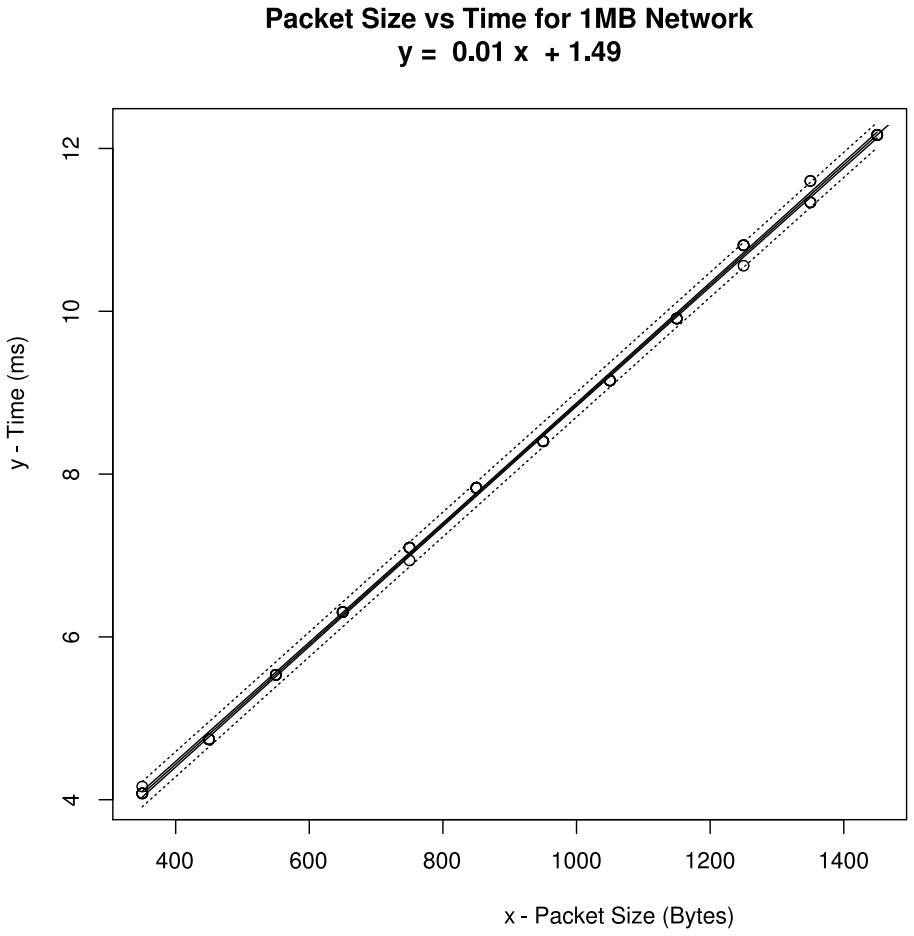
Settle a law suit

Using the 'lm' functions in R or the simple.lm functions from the UsingR package (documented in the Simple R guide, section 13 page 77), answer the following questions

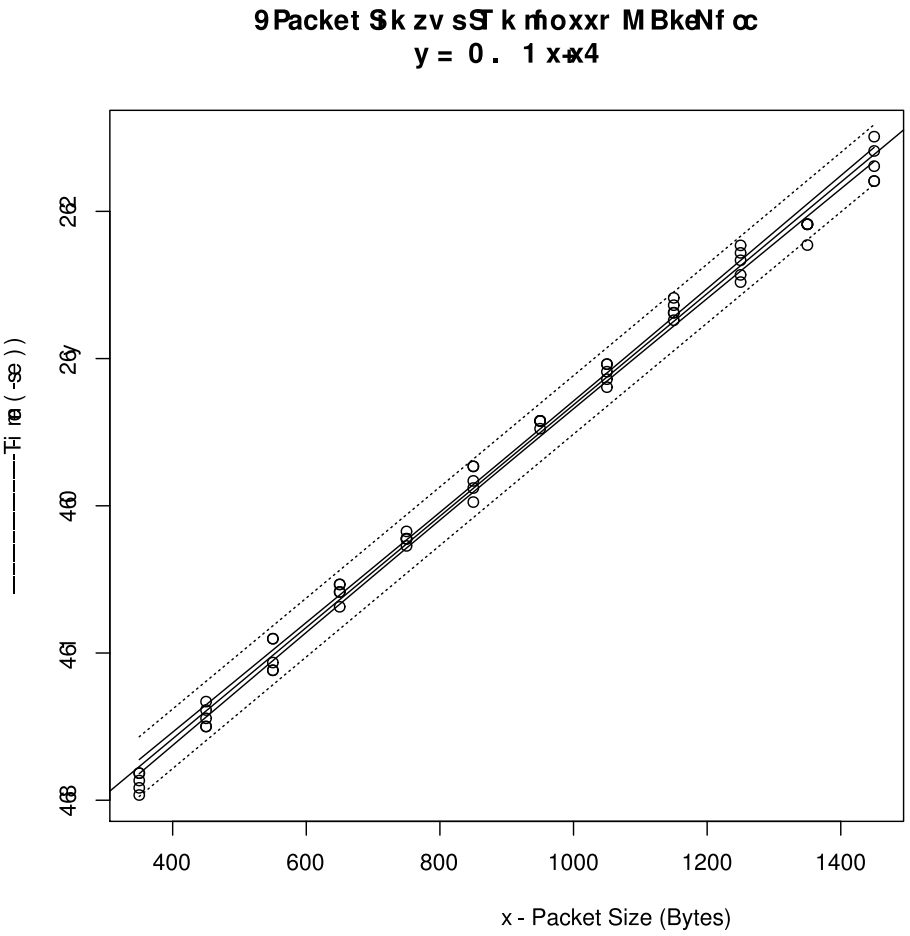
- Prepare a plot of the data, the regression model and the 95% confidence interval for each of the data sets. Label the plot with the parameters of the regression model. If you plot all the data on a single plot, you can put the parameters in the legend rather than the title.
- Larry the Lawyer wants to sue because the 11Mb/s network is not 11 times "faster" than the 1Mb/s network. Compare the slope (time per byte) and intercept (overhead per packet). Is the 11Mb/s network 11 times faster at a 95% confidence level? Is the overhead different?
- Using your models, predict the time to transmit a 40 byte packet using each network (using a 95% confidence interval). What's Larry doing now (i.e. crying or filiing a suit?)
- Repeat that for a 750 and 1500 byte packet. Does Larry still have a case? What if you compare the time to transmit a 40 byte packet to a 1500 byte packet, which is 37 times bigger?
- For the 54Mb/s data, argue that the regression model is or is not appropriate for the data. Use the full range of techniques described in Jain and in class. Are there specific measurement samples which seem to be more problematic than others? Which ones?

In [88]:
```
simple.lm(x1,y1,show.ci=TRUE, conf.level=0.95)
title(xlab="                                    - Packet Size (Bytes)")
title(ylab="                        - Time (ms)")
title(main="Packet Size vs Time for 1MB Network\n\n")
simple.lm(x11,y11,show.ci=TRUE, conf.level=0.95)
title(xlab="                                    - Packet Size (Bytes)")
title(ylab="                        - Time (ms)")
title(main="Packet Size vs Time for 11MB Network\n\n")
```

Out[88]:
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   1.492115     0.007361
```



Out[88]:
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  1.1810648    0.0007561
```


9Packet Sk zv sST k mhoxxr MBkeNf oc
y = 0 . 1 x+x4
——————Ti me (-se))
400
600
800
1000
1200
1400
x - Packet Size (Bytes)

In [89]:
```
#1MB (slope,int)
#1.492115     0.007361
#11MB
#1.1810648    0.0007561

# 1 Mb = 1048576 b



m1 = lm(y1~x1)
confint(m1, ,.95)

m11 = lm(y11~x11)
confint(m11, ,.95)


##BANDWIDTH
#1av/11av
(bw_m = (0.007361/0.0007561))

#1_low/11_high
(0.007304786/0.0007707959)
#1_high/11_low
(0.007417312/0.0007413121)

print("The overhead for the 11MB is about 0.3ms faster, and the bandwidth is about 10 times faster")
```

Out[89]:

	2.5 %	**97.5 %**
(Intercept)	1.437881	1.546349
x1	0.007304786	0.007417312

Out[89]:

	2.5 %	**97.5 %**
(Intercept)	1.166855	1.195275
x11	0.0007413121	0.0007707959

Out[89]: 9.73548472424283

Out[89]: 9.47693935580093

Out[89]: 10.0056534892659

```
[1] "The overhead for the 11MB is about 0.3ms faster, and the bandwidth is about 10 times faster"
```

In [90]:
```
#Predict 40:
print("1MB, 40B packet:")
#1MB 2.5%
(0.007304786*40+1.437881)
#bandwidth
(((0.007304786/1000)^(-1))*8/1048576)
#1MB 97.5%
(0.007417312*40+1.546349)
#bandwidth
(((0.007417312/1000)^(-1))*8/1048576)

print("11MB, 40B packet:")
#11MB 2.5%
(1.166855+0.0007413121*40)
#bandwidth
(((0.0007413121/1000)^(-1))*8/1048576)
#11MB 97.5%
(1.195275+0.0007707959*40)
#bandwidth
(((0.0007707959/1000)^(-1))*8/1048576)



#Time to rate:
print("1Mb bandwidth is 1.02Mb/s-1.04Mb/s>")
print("11Mb bandwidth is 9.90Mb/s-10.29Mb/s")
print("If larry doesn't believe latency is real, he is filing a suit.")
print("If larry thinks 10Mb/s is not 11Mb/s he is filing a suit.")
```

```
[1] "1MB, 40B packet:"
```

Out[90]: 1.73007244

Out[90]: 1.04443778794478

Out[90]: 1.84304148

Out[90]: 1.02859290956751

```
[1] "11MB, 40B packet:"
```

Out[90]: 1.196507484

Out[90]: 10.2917442346483

Out[90]: 1.226106836

Out[90]: 9.89807357725956

```
[1] "1Mb bandwidth is 1.02Mb/s-1.04Mb/s>"
[1] "11Mb bandwidth is 9.90Mb/s-10.29Mb/s"
[1] "If larry doesn't believe latency is real, he is filing a suit."
[1] "If larry thinks 10Mb/s is not 11Mb/s he is filing a suit."
```

In [91]:
```
#Predict 750:
print("1MB, 750B packet:")
#1MB 2.5%
(0.007304786*750+1.437881)
#1MB 97.5%
(0.007417312*750+1.546349)

print("11MB, 750B packet:")
#11MB 2.5%
(1.166855+0.0007413121*750)
#11MB 97.5%
(1.195275+0.0007707959*750)
```

```
[1] "1MB, 750B packet:"
```

Out[91]: 6.9164705

Out[91]: 7.109333

```
[1] "11MB, 750B packet:"
```

Out[91]: 1.722839075

Out[91]: 1.773371925

In [92]:
```
#Predict 1500:
print("1MB, 1500B packet:")
#1MB 2.5%
(0.007304786*1500+1.437881)
#1MB 97.5%
(0.007417312*1500+1.546349)

print("11MB, 1500B packet:")
#11MB 2.5%
(1.166855+0.0007413121*1500)
#11MB 97.5%
(1.195275+0.0007707959*1500)
```

```
[1] "1MB, 1500B packet:"
```

Out[92]: 12.39506

Out[92]: 12.672317

```
[1] "11MB, 1500B packet:"
```

Out[92]: 2.27882315

Out[92]: 2.35146885

In [93]:
```
#Slowest 1MB 1500/ Fastest 11MB 1500
(12.672317/2.27882315)

#Slowest 1MB 40/ Fastest 11MB 40
(1.84304148/1.196507484)

print("Larry is a lawyer and thus he can probably make this a case.")
print("Looking at %faster is slightly misleading as the 1Mb network performs better than 1Mb,")
print("and the 11Mb/s network slightly underperforms.")
print("That being said bringing up that the 11Mb network only performs 5x better could be a nice point.")
print("Looking at bandwidth, the 11Mb network slightly underperforms at around 10Mb/s")
print("Unless the ISP said up to 11Mb/s, Larry should sue.")
```

Out[93]: 5.56090410087329

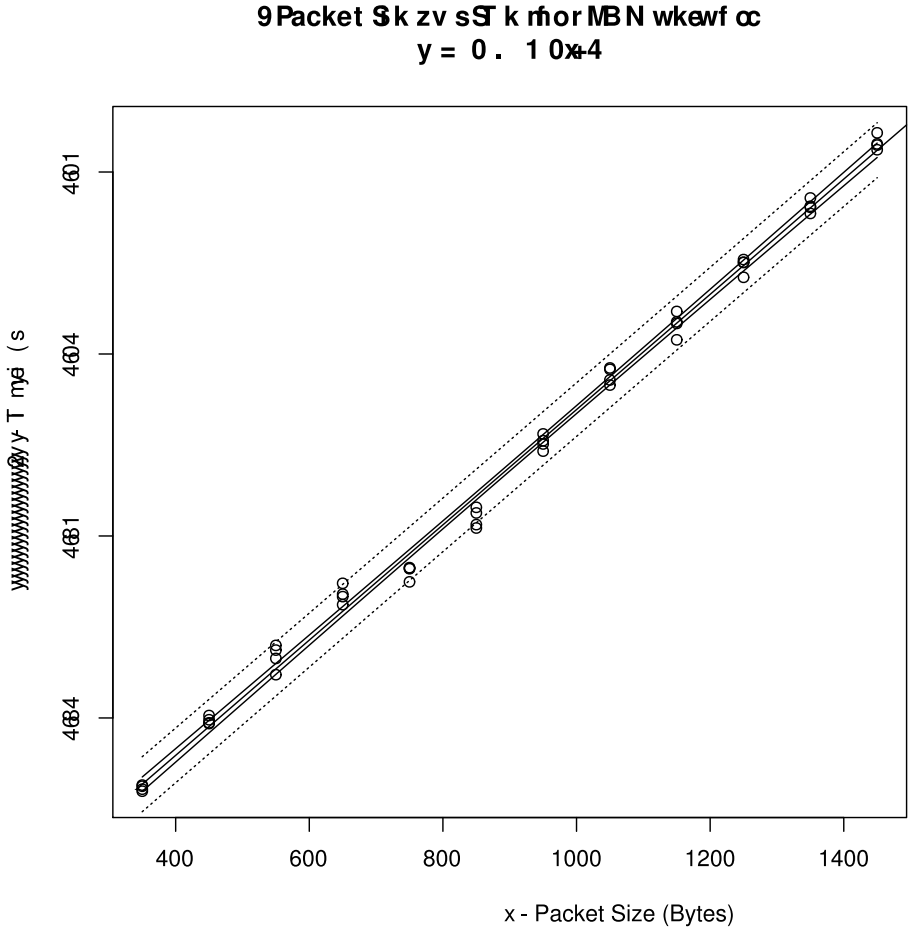
Out[93]: 1.54035098371353

```
[1] "Larry is a lawyer and thus he can probably make this a case."
[1] "Looking at %faster is slightly misleading as the 1Mb network performs better than 1Mb,"
[1] "and the 11Mb/s network slightly underperforms."
[1] "That being said bringing up that the 11Mb network only performs 5x better could be a nice point
[1] "Looking at bandwidth, the 11Mb network slightly underperforms at around 10Mb/s"
[1] "Unless the ISP said up to 11Mb/s, Larry should sue."
```

In [94]:
```
simple.lm(x54,y54,show.ci=TRUE, conf.level=0.95)
title(xlab="                                    - Packet Size (Bytes)")
title(ylab="                        - Time (ms)")
title(main="Packet Size vs Time for 54MB Network\n\n")
m54 <- lm(y54~x54)
summary(m54)
```

Out[94]:
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  0.2263869    0.0001583
```

Out[94]:
```
Call:
lm(formula = y54 ~ x54)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0087862 -0.0016716  0.0002906  0.0020847  0.0076894

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.264e-01  1.459e-03   155.1   <2e-16 ***
x54         1.583e-04  1.514e-06   104.6   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.003621 on 46 degrees of freedom
Multiple R-squared:  0.9958,    Adjusted R-squared:  0.9957
F-statistic: 1.094e+04 on 1 and 46 DF,  p-value: < 2.2e-16
```



In [95]:
```
print("This is a fairly good data set for linear regression.")
print("The R2 value is .995, meaning the data is fit very well.")
print("There are a few data points that are outside the prediction line (between x=600 adn 900),")
print("however they still fall within the 95% confidence interval.")
```

```
[1] "This is a fairly good data set for linear regression."
[1] "The R2 value is .995, meaning the data is fit very well."
[1] "There are a few data points that are outside the prediction line (between x=600 adn 900),"
[1] "however they still fall within the 95% confidence interval."
```

In [ ]: