Get Data For Regression

```
## If you have the ortann.rds file already, use it.
## Otherwise download
if (file.exists("ortann.rds")){
    print("Using saved file ortann.rds")
    dat <- readRDS("ortann.rds")
 } else {
    print("Wait. Downloading ortann.csv file")
 dat <- read.table(url("http://pj.freefaculty.org/guides/stat/
     DataSets/OregonTemps/ortann.csv"), header=T, sep=",")
    saveRDS(dat, file = "ortann.rds")
 }
```

```
[1] "Using saved file ortann.rds"
```

```
str(dat)
```

```
'data.frame':  92 obs. of  5 variables:
 $ station  : Factor w/ 92 levels "ANT","ARL","ASH",..: 1 2 3 4 7 8
     5 9 6 10 ...
 $ latitude : num  44.9 45.7 42.2 46.1 44.8 ...
 $ longitude: num  -121 -120 -123 -124 -118 ...
 $ elevation: int  846 96 543 2 1027 1050 24 1097 999 18 ...
 $ tann     : num  9.6 12.5 11.1 10.3 7.6 8.4 10.8 7.7 9.5 11.2 ...
```

summarize

```
library(rockchalk)
summarize(dat)
```

```
$numerics
      elevation  latitude  longitude    tann
0%         2.00    42.050   -124.600   3.200
25%       94.25    43.600   -123.200   8.700
50%      462.00    44.460   -121.700  10.400
75%      863.80    45.280   -119.600  11.200
100%    1974.00    46.150   -117.000  12.600
mean     557.00    44.320   -121.300   9.885
sd       498.30     1.116      2.278   1.874
var   248300.00     1.245      5.191   3.511
NA's       0.00     0.000      0.000   0.000
N         92.00    92.000     92.000  92.000

$factors
           station
 ANT            :  1.000
 ARL            :  1.000
 ASH            :  1.000
 AST            :  1.000
 (All Others)   :88.000
 NA's           :  0.000
 entropy        :  6.524
 normedEntropy  :  1.000
 N              :92.000
```

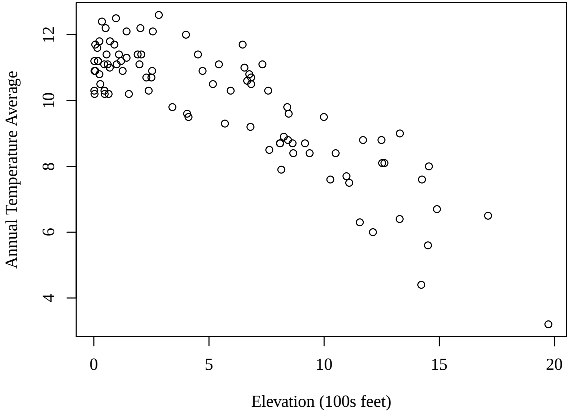
One recode before going too much further

- I know (from reading ahead in lecture ☺) that the calculations based on elevation will lead to coefficients that are too small, as in 0.000043.
- I create a new variable elevationP100 to fix that

```
dat$elevationP100 <- dat$elevation / 100
```

- Note my style
  1. DO NOT recode "over" old variable, create new one for bug-checking
  2. Create new variable with same name at front, suffix represents change

Plot




```
plot(tann ~ elevationP100, data = dat, xlab = "Elevation (100s feet
    )", ylab="Annual Temperature Average")
```

Regression Analysis

```
mod1 <- lm (tann ~ elevationP100 , data=dat)
summary(mod1)
```

```
Call:
lm(formula = tann ~ elevationP100 , data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6841 -0.6026 -0.1081  0.7613  2.1034

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   11.68814    0.15030   77.77   <2e-16 ***
elevationP100 -0.32377    0.02016  -16.06   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9582 on 90 degrees of freedom
Multiple R^2:  0.7413, Adjusted R^2:  0.7385
F-statistic: 257.9 on 1 and 90 DF,  p-value: < 2.2e-16
```

Regression Table

	One Predictor Model Estimate (S.E.)
(Intercept)	11.688*** (0.150)
elevation/100	-0.324*** (0.020)
N	92
RMSE	0.958
R^2	0.741

$*p \leq 0.05 ** \ p \leq 0.01 *** p \leq 0.001$

- Estimated Intercept
- Estimated Slope
- Standard Error of Intercept Estimate (estimated standard deviation of intercept estimator)
- Standard Error of Slope Estimate (estimated standard deviation of slope estimator)

Hypothesis Test for Slope

- Theory: $tann_i = \beta_0 + \beta_1 elevation_i + e_i$
  3 parameters: β_0 ,β_1 and σ_e^2 (recall $E[e_i] = 0$, $Var[e_i] = E[e_i^2] = \sigma_e^2$).
- The Null Hypothesis: $H_0 : \beta_1 = 0$
- $\hat{\beta}_1$ is approximately Normal, but its standard deviation, $\sqrt{Var[\hat{\beta}_1]}$ is unknown. However, using $std.err.(\hat{\beta}_1)$, we form the test statistic that has a T distribution:

$$\hat{t} = \frac{\hat{\beta}_1 - 0}{std.err.(\hat{\beta}_1)} = \frac{-0.3237}{0.02016} = -16.06$$

- The critical value of t is:

```
> qt( c(0.025, 0.975), df=90)
[1] -1.986675  1.986675
```

- Conclusion: "The estimate $\hat{\beta}_1$ is statistically significantly different from 0."

Confidence Intervals for Intercept and Slope

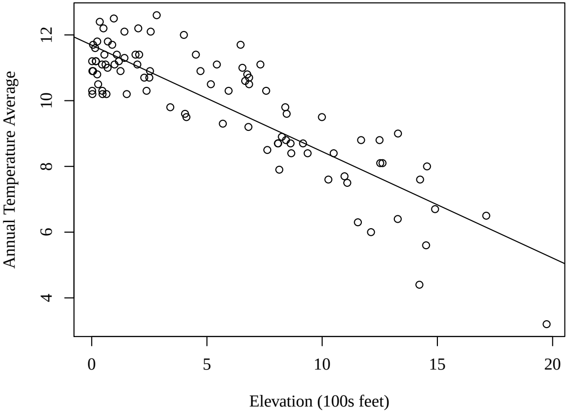
```
confint(mod1)
```

```
                 2.5 %      97.5 %
(Intercept)    11.3895457 11.9867314
elevationP100  -0.3638196 -0.2837176
```

Supposing the model's theory is correct, we believe

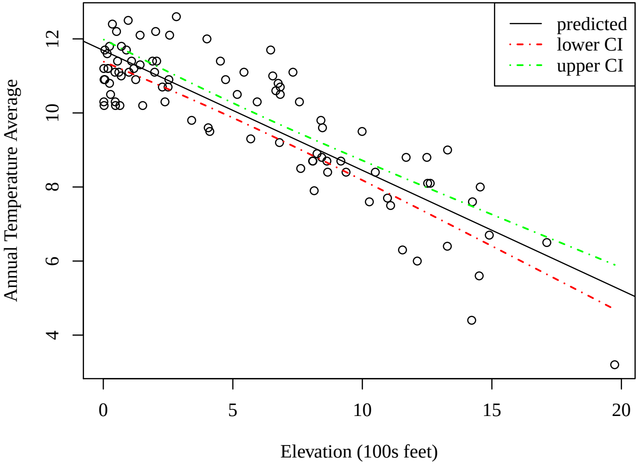
- the probability is 0.95 that the true slope β_1 is between -0.363 and -0.284.

Draw Predicted Values




```
plot(tann ~ elevationP100, data = dat, xlab = "Elevation (100s feet
    )", ylab = "Annual Temperature Average")
abline(mod1)
```

Superimpose Confidence Interval For Predicted Values



Code For Previous

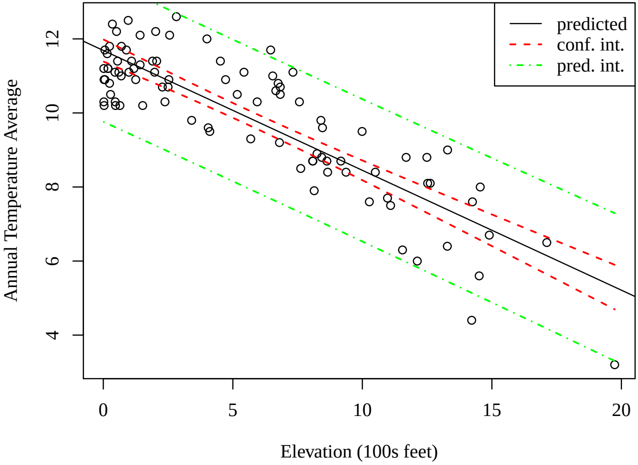
```
plot(tann ~ elevationP100, data = dat, xlab="Elevation (100s feet)"
     , ylab="Annual Temperature Average")
abline(mod1)
newdf <- data.frame(elevationP100 = plotSeq(dat$elevationP100, 20))
pconf <- predict(mod1, interval = "confidence", newdata = newdf)
lines(newdf$elevation, pconf[, "lwr"], lwd = 1.5, lty=4, col="red")
lines(newdf$elevation, pconf[, "upr"], lwd = 1.5, lty=4, col="green
     ")
legend("topright", legend=c("predicted","lower CI","upper CI"), lty
     =c(1,4,4), col=c("black", "red","green"), lwd = c(1, 1.5, 1.5))
```

VERY important

1. R idiom: Create newdata object, give that to predict.
2. Use lines or other R plotting functions to decorate the existing plot
3. All well-defined regression routines in R will include predict function

- rockchalk::plotSeq simply gives a selection of evenly spaced values. I should have named that something else. I was not aware of built-in R function pretty when I created that function.

Superimpose Confidence and Prediction Intervals



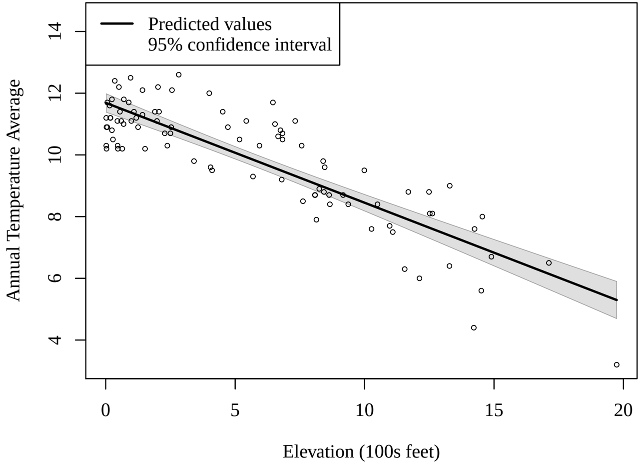
Code For Previous

```
plot(tann ~ elevationP100, data = dat, xlab = "Elevation (100s feet
    )", ylab="Annual Temperature Average")
abline(mod1)
newdf <- data.frame(elevationP100 = plotSeq(dat$elevationP100, 20))
pconf <- predict(mod1, interval="confidence", newdata=newdf)
lines(newdf$elevationP100, pconf[ ,"lwr"], lwd = 1.5, lty=2, col="
    red")
lines(newdf$elevationP100, pconf[ ,"upr"], lwd = 1.5, lty=2, col="
    red")
ppred <- predict(mod1, interval="prediction", newdata=newdf)
lines(newdf$elevationP100, ppred[ ,"lwr"], lwd = 1.5, lty=4, col="
    green")
lines(newdf$elevationP100, ppred[ ,"upr"], lwd = 1.5, lty=4, col="
    green")
legend("topright", legend=c("predicted","conf. int.","pred. int."),
     lty=c(1,2,4), col=c("black", "red","green"), lwd = c(1, 1.5, 1
    .5))
```

If you step through this line-by-line, you see pconf is a matrix, not a data.frame.

I was lazy here, did not try to bind together newdf and ppred. But in many cases, I would do that (as in lab notes)

Same plot from rockchalk plotSlopes



plotSlopes usage is a little simpler

```
plotSlopes(mod1, plotx = "elevationP100", xlab="Elevation (100s
    feet)", ylab="Annual Temperature Average", interval = "
    confidence")
```

- I didn't write this to "overlay" both confidence and prediction intervals. May think of way to do that, someday.
- plotSlopes was originally intended to help with purpose of plotting several lines on one plot. See argument modx.