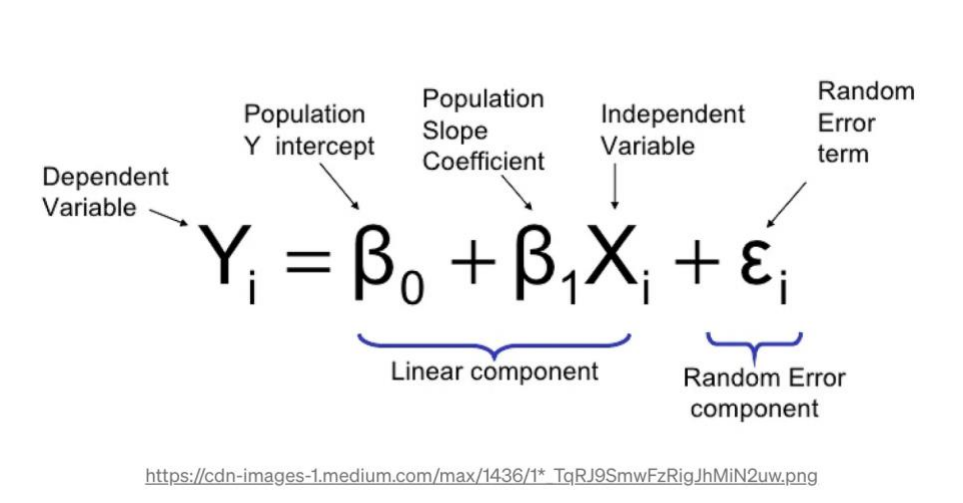
BIOSTATS 640 – Introduction to R
Fall 2023
https://people.umass.edu/biep640w/webpages/demonstrations.html



06
Introduction to Linear Regression in R
October 13, 2023

Dataset used
ers.Rdata

Packages Used:

```
tidyverse, summarytools, ggplot2, gridExtra, stargazer, jtools, broom
```

1. Introduction to the New York Auto Club Data

ers.Rdata

https://www.accidenttampa.com/driving-in-bad-weather-8-safety-tips-from-an-accident-lawyer/

Source:
Chatterjee, S; Handcock MS and Simonoff JS *A Casebook for a First Course in Statistics and Data Analysis.* New York, John Wiley, 1995, pp 145-152.

Setting:
This lesson (R06) uses the R dataset `ers.Rdata` to illustrate simple linear regression in R. There are n=28 observations of p=12 variables. The data are from the New York Auto Club. Of interest is the relationship between the number of calls to the autoclub in relationship to the weather. This illustration considers just two variables: Y=calls and X=low. R is used to produce numerical and graphical descriptions of the data and perform a simple linear regression.

In future R lessons, you will will learn how to fit multiple predictors and perform regression diagnosticss.

Simple Linear Regression Variables:
Outcome Y = calls
Predictor X = low.

2. Highlights of Lesson 05
Epidemiological Tables

2x2 Table

```
Enter 2x2 table counts row by row
table1 <- as.table(rbind(c(84,43), c(10,92)))

Create dataframe of individual observations
library(DescTools)
df.table1 <- Untable(table1)

Cross-tabs with row percents
library(summarytools)
ctable(df.table1$Culture,df.table1$Dipstick,
       prop='r',
       totals=FALSE)

Get Diagnostic Testing Statistics (e.g.,
sensitivity, specificity, etc.)
library(epiR)
epi.tests(table1, conf.level=0.90)

Get Relative Risk (RR) and Odds Ratio (OR)
library(DescTools)
RelRisk(table1, conf.level=0.90)
OddsRatio(table1, conf.level=0.90)

Hypothesis Tests of Null:  No Association
fisher.test(table1)
chisq.test(table1)

library(epiR)
epi.2by2(table1, method="case.control")
epi.2by2(table1, method="cohort.count")
```

Stratified Analysis of K 2x2 Tables

```
Enter K 2x2 tables, each column by column
tablek2x2 <- array(c( 1011, 81, 390, 77,
                      383, 66, 365, 123),
                       dim = c(2, 2, 2))

Test of Null:  Homogeneity of Association
("No effect modification")
library(epiDisplay)
mhor(mhtable=tablek2x2,decimal=2,
        graph=FALSE,
        design="case control")


Test of Null:  Common OR = 1
library(epiDisplay)
mhor(mhtable=tablek2x2,decimal=2,
        graph=FALSE,
        design="case control")
```

3. Explore Your Data

Initialize session

```
setwd("/cloud/project")                                  # Set working directory
options(scipen=999)                                      # Turn off scientific notation
rm(list = ls())                                          # Clear the Decks
```

Input Rdata. Inspect

```
load(file="ers.Rdata")                                   # load(file="FULL NAME IN QUOTES")
str(ersdata)                                             # str() to inspect

## 'data.frame':    28 obs. of  12 variables:
##  $ day    : int  12069 12070 12071 12072 12073 12074 12075 12076 12077 12078 ...
##  $ calls  : int  2298 1709 2395 2486 1849 1842 2100 1752 1776 1812 ...
##  $ fhigh  : int  38 41 33 29 40 44 46 47 53 38 ...
##  $ flow   : int  31 27 26 19 19 30 40 35 34 32 ...
##  $ high   : int  39 41 38 36 43 43 53 46 55 43 ...
##  $ low    : int  31 30 24 21 27 29 41 40 38 31 ...
##  $ rain   : int  0 0 0 0 0 0 1 0 1 0 ...
##  $ snow   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday: int  0 0 0 1 1 1 1 0 0 1 ...
##  $ year   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sunday : int  0 1 0 0 0 0 0 0 1 0 ...
##  $ subzero: int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "datalabel")= chr ""
##  - attr(*, "time.stamp")= chr ""
##  - attr(*, "formats")= chr [1:12] "%8.0g" "%8.0g" "%8.0g" "%8.0g" ...
##  - attr(*, "types")= int [1:12] 252 252 251 251 251 251 251 251 251 251 ...
##  - attr(*, "val.labels")= chr [1:12] "" "" "" "" ...
##  - attr(*, "var.labels")= chr [1:12] "" "" "" "" ...
##  - attr(*, "version")= int 8
```

Clean data.

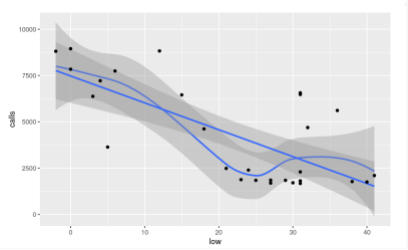
```
library(tidyverse)                                       # select() in {dplyr} included in {tidyverse}

mydata <- ersdata %>%
          dplyr::select(low,calls)                       # select() to choose variables X=low and Y=calls

mydata$low <- as.numeric(mydata$low)                     # as.numeric( ) to convert to numeric
mydata$calls <- as.numeric(mydata$calls)

mydata                                                   # show (okay since n=28 is modest)

##    low calls
## 1   31  2298
## 2   30  1709
## 3   24  2395
## 4   21  2486
## 5   27  1849
## 6   29  1842

---  several rows omitted ---

## 22  18  4619
## 23  31  6476
## 24  32  4692
## 25   5  3638
## 26   0  8947
## 27  31  6564
## 28  36  5613
```

Explore Data: Assess missing values, range, outliers

```
library(summarytools)

myvars <- c("low", "calls")                          # GOOD TO KNOW!  Handy for choosing variables to examine

descr(mydata[myvars],                                # Square brackets to identify columns of dataframe
      stats=c("n.valid", "min","max"),
      transpose=TRUE)
## Descriptive Statistics
## mydata
## N: 28
##
##               N.Valid       Min       Max
## ----------- --------- --------- ---------
##       calls     28.00   1674.00   8947.00
##         low     28.00     -2.00     41.00
```

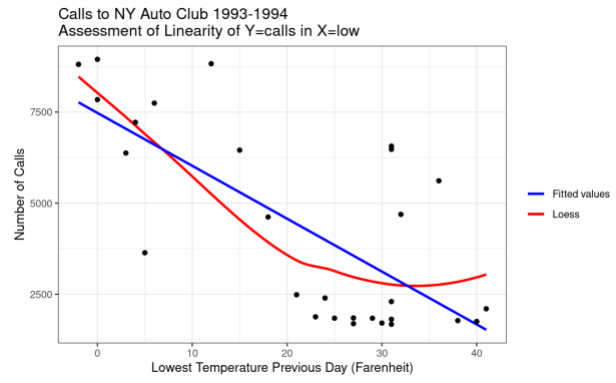
Explore Data: ggplot( ) to assess linearity - BASIC

```
library(ggplot2)

ggplot(data=mydata) +                                # required:  data =
  aes(x=low, y=calls) +                              # required:  aes(x=, y=)
  geom_smooth(method = "loess") +                    # loess fit
  geom_smooth(method = "lm") +                       # linear fit
  geom_point()                                       # TIP - plot points on top of lines
```



Explore Data: ggplot() to assess linearity - WITH AESTHETICS

```
library(ggplot2)

ggplot(data=mydata) +                                # required:  data =
  aes(x=low, y=calls) +                              # required:  aes(x=, y=)
  geom_smooth(method = "loess",span=1, aes(color="Loess"), se=FALSE) +    # loess fit, se=FALSE to remove CI
  geom_smooth(method = "lm", aes(color="Fitted values"), se=FALSE) +      # linear fit, se=FALSE to remove CI
  geom_point() +
  scale_colour_manual(name="",values=c("blue", "red")) +                  # blue for fit, red for loess
  xlab("Lowest Temperature Previous Day (Farenheit)") +
  ylab("Number of Calls") +
  ggtitle("Calls to NY Auto Club 1993-1994\nAssessment of Linearity of Y=calls in X=low") +
  theme_bw()
```



The scatterplot on the previous page suggests, as we might expect, that lower temperatures are associated with more calls to the NY Auto Club. We also see that the data are a bit messy.

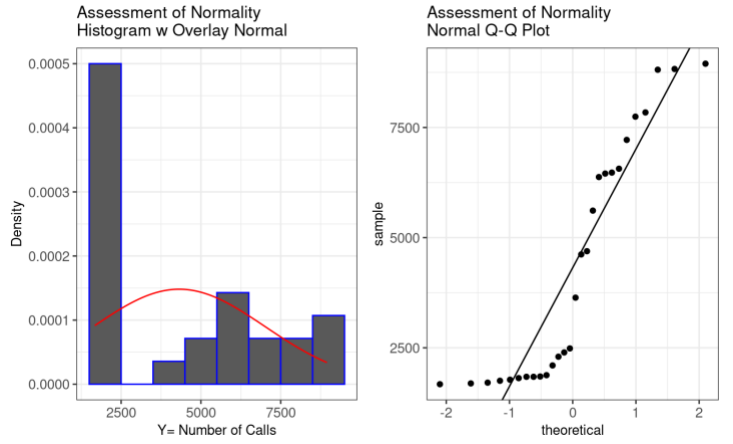
Unfamiliar with LOESS smoothing? LOESS regression stands for "locally weighted scatterplot smoother". It is a technique for drawing a smooth line through a scatter plot to obtain a sense for the nature of the functional form that relates X to Y, not necessarily linear. The method involves the following: At each observation (x,y), the observed data point is fit to a line using some "adjacent" points. It's handy for seeing where in the data linearity holds and where it no longer holds. Nifty!

Assess Normality of Y

Recall. In normal theory regression, we assume that the outcome variable (in this case, **Y=calls**) can reasonably be assumed to be distributed normal (more on violations of this later…) So a preliminary is often to check this assumption before doing any model fits. If gross violations are apparent then, possibly, Y will be replaced by some transformation of Y that is better behaved. Recall. It's okay for the predictor X (in this case X=low) to be NOT distributed normal. In fact, it is regarded as fixed (not random at all!) Here is a lengthy bit of R code for you, so that you can pick and choose between the basic and the more fancy!

```
Assess Assumption of Normality of Distribution of Y - Hypothesis Test
shapiro.test(mydata$calls)                                          # NULL:  Normality is reasonable

##
##  Shapiro-Wilk normality test
##
## data:  mydata$calls
## W = 0.82902, p-value = 0.0003628

The null hypothesis of normality of Y=calls is rejected (p-value = .00036).   But is it really a problem? Stay tuned..
```

```
Assess Assumption of Normality of Distribution of Y - Histogram and QQ Plot
library(ggplot2)
library(gridExtra)                                          # grid.arrange() in {gridExtra} to arrange panels

# panel 1 = histogram w overlay normal
p1 <-    ggplot(data=mydata) +
         aes(x=calls) +
         geom_histogram(binwidth=1000,
                        color="blue",
                        aes(y=..density..)) +
         stat_function(fun=dnorm,
                       color="red",
                       args=list(mean=mean(mydata$calls),
                                 sd=sd(mydata$calls))) +
         ggtitle("Assessment of Normality\nHistogram w Overlay Normal") +
         xlab("Y= Number of Calls") +
         ylab("Density") +
         theme_bw() +
         theme(axis.text = element_text(size = 10),
              axis.title = element_text(size = 10),
              plot.title = element_text(size = 12))


# panel 2 = quantile-quantile plot
p2 <- ggplot(data=mydata) +
          aes(sample=calls) +
          stat_qq() +
          geom_abline(intercept=mean(mydata$calls),
                      slope = sd(mydata$calls)) +
          ggtitle("Assessment of Normality\nNormal Q-Q Plot") +
          theme_bw() +
          theme(axis.text = element_text(size = 10),
               axis.title = element_text(size = 10),
               plot.title = element_text(size = 12))

gridExtra::grid.arrange(p1, p2, ncol=2)
```



We can see now why the null hypothesis of normality of Y=calls is rejected (p-value = .00036). Tip- sometimes the cure is worse than the original violation. For now, and because the focus here is elsewhere, we'll charge on.

4. Fit Your Model

lm() to fit linear regression model

```
lm_fit <- lm(calls ~ low, data=mydata)                                  # lm(y ~ x, data=)
```

Show Fit - Basic

```
lm_fit                                                                  # show saved model object

##
## Call:
## lm(formula = calls ~ low, data = mydata)
##
## Coefficients:
## (Intercept)          low
##      7475.8       -145.2

#(lm(calls ~ low, data=mydata))                                         # Also works. Remove hashtag to execute
```

The fitted model is thus

$$\widehat{calls} = 7475.8 - 145.2 \cdot low$$

Show Fit - summary( )

```
summary(lm_fit)

##
## Call:
## lm(formula = calls ~ low, data = mydata)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3112  -1468   -214   1144   3588
##
## Coefficients:
##             Estimate Std. Error t value       Pr(>|t|)
## (Intercept)  7475.85     704.63  10.610 0.000000000061 ***
## low          -145.15      27.79  -5.223 0.000018649091 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1917 on 26 degrees of freedom
## Multiple R-squared:  0.5121, Adjusted R-squared:  0.4933
## F-statistic: 27.28 on 1 and 26 DF,  p-value: 0.00001865

#summary(lm(calls ~ low, data=mydata))                                  # Also works. Remove hashtag to execute
```

Show Fit - stargazer() in package {stargazer}

```
library(stargazer)

stargazer(lm_fit,type="text",title="Calls to NY AutoClub")
```

```
##
## Calls to NY AutoClub
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                                calls
## ------------------------------------------------
## low                          -145.154***
##                               (27.789)
##
## Constant                    7,475.849***
##                               (704.630)
##
## ------------------------------------------------
## Observations                    28
## R2                             0.512
## Adjusted R2                    0.493
## Residual Std. Error     1,916.664 (df = 26)
## F Statistic           27.285*** (df = 1; 26)
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Show Fit - summ() in package {jtools}

```
library(jtools)

summ(lm_fit)
```

```
MODEL INFO:
Observations: 28
Dependent Variable: calls
Type: OLS linear regression

MODEL FIT:
F(1,26) = 27.28, p = 0.00
R² = 0.51
Adj. R² = 0.49

Standard errors: OLS
-------------------------------------------------
                      Est.     S.E.   t val.      p
----------------- --------- -------- -------- ------
(Intercept)         7475.85   704.63    10.61   0.00
low                 -145.15    27.79    -5.22   0.00
-------------------------------------------------
```

Show Fit (as you like) - betas and associated confidence interval limits

```
cbind(coef(lm_fit),confint(lm_fit))                                    # show betas and 95% CI - default
```

```
##                            2.5 %     97.5 %
## (Intercept) 7475.849 6027.4605 8924.23745
## low         -145.154 -202.2744  -88.03352
```

```
cbind(coef(lm_fit),confint(lm_fit,level=.90))                           # show betas and 90% CI
```

```
##                             5 %      95 %
## (Intercept) 7475.849 6274.0188 8677.6792
## low         -145.154 -192.5508  -97.7571
```

TIP! names() to show names of internal objects created by R

```
# model
cat("\nObjects in model\n")
names(lm_fit)                                                    # names(SAVED MODEL)

##
## Objects in model
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"

#names(lm(calls ~ low, data=mydata))                             # Also works.  Remove hashtag to use.

# summary of model
cat("\nObjects in summary()\n")
names(summary(lm_fit))

##
## Objects in summary()
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

TIP - How to display internal objects created by R

```
cat("\nBetas\n")
lm_fit$coefficients                                             # show betas

##
## Betas
## (Intercept)         low
##    7475.849    -145.154

cat("\nR-squared\n")
summary(lm_fit)$r.squared                                        # show R Squared

##
## R-squared
## [1] 0.5120567
```

5. Analysis of Variance

Analysis of Variance Table and Overall F Test - Basic

```
anova(lm_fit)

## Analysis of Variance Table
##
## Response: calls
##           Df    Sum Sq   Mean Sq F value     Pr(>F)
## low        1 100233719 100233719  27.285 0.00001865 ***
## Residuals 26  95513596   3673600
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Putting this all together (note- some of the notes below draw from earlier output)
Remarks

- *The fitted line is* $\widehat{\text{calls}} = 7{,}475.85 - 145.15*[\text{low}]$
- *R^2 = .51 indicates that 51% of the variability in calls is explained.*
- *The overall F test significance level "PROB > F" < .0001 suggests that the straight line fit performs better in explaining variability in calls than does* $\bar{Y}$ *= average # calls*
- *From this output, the analysis of variance is the following:*

Source	Df	Sum of Squares	Mean Square
Model "Regression"	1	MSS = $\sum_{i=1}^{n}\left(\hat{Y}_i - \bar{Y}\right)^2$ = 100,233,719	MSS/1 = 100,233,719
Residual "Error"	(n-2) = 26	RSS = $\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$ = 95,513,596	RSS/(n-2) = 3,673,600
Total, corrected	(n-1) = 27	TSS = $\sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2$ = 195,747,315	

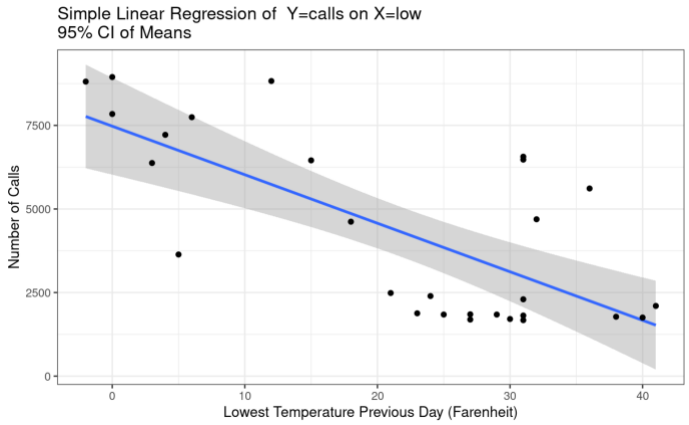
Overall F Test - Illustration of extracting internal R objects and use of {broom}

```
library(broom)                                          # glance() in {broom}

#names(glance(lm_fit))                                  # Remove hashtag to execute

overall_F <- glance(lm_fit)$statistic
numerator_df <- glance(lm_fit)$df
denominator_df <- glance(lm_fit)$df.residual
p_value <- glance(lm_fit)$p.value


cat("\nSimple Linear Regression of Y=calls on X=low")
cat("\nOverall F-Test (null: zero slope) =  ",  overall_F)
cat("\ndf = ",numerator_df,",",denominator_df)
cat("\np-value = ",p_value)

##
## Simple Linear Regression of Y=calls on X=low
## Overall F-Test (null: zero slope) =   27.28488                  [ 100,233,719 ] / [ 3,673,600 ] = 27.28
## df =  1 , 26
## p-value =  0.00001864909
```

6. Report Your Model

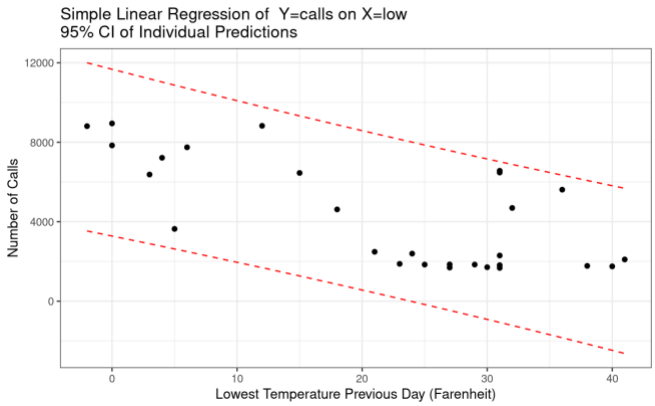
Report Your Model - XY Scatter with overlay fit and 95% CI of estimated MEANS

```
library(ggplot2)
ggplot(data=mydata) +                                                    # Required layer: data=
 aes(x=low, y=calls) +                                                   # Required layer: aes=
 geom_smooth(method=lm, level=.95, se=TRUE) +                            # Required layer: geom_smooth()
 geom_point() +                                                          # Required layer: geom_point()
 xlab("Lowest Temperature Previous Day (Farenheit)") +                   # Optional
 ylab("Number of Calls") +                                               # Optional
 ggtitle("Simple Linear Regression of  Y=calls on X=low\n95% CI of Means") +   # Optional
 theme_bw()                                                              # Optional
```



Report Your Model - XY Scatter with overlay fit and 95% CI of INDIVIDUAL PREDICTIONS

```
library(ggplot2)
yhat <- predict(lm_fit, interval="prediction")                           # get individual predictions as yhat
temp_df <- cbind(mydata, yhat)                                           # add to dataset
ggplot(temp_df, aes(x=low, y=calls)) +                                   # Required layers
  geom_line(aes(y=lwr), color = "red", linetype = "dashed") +
  geom_line(aes(y=upr), color = "red", linetype = "dashed") +
  geom_point() +
  xlab("Lowest Temperature Previous Day (Farenheit)") +                  # Optional
  ylab("Number of Calls") +
  ggtitle("Simple Linear Regression of  Y=calls on X=low\n95% CI of Individual Predictions") +
  theme_bw()
```



Report Your Model - 95% Confidence Interval Estimates of MEANS

```
#-- Predictions for new values
newvalues <- data.frame(low=c(0,10,20,30))

# Confidence Interval Estimates of Means
estimated_mean <- predict(lm_fit,newdata=newvalues, interval="confidence")
out_mean <- cbind(newvalues,estimated_mean)
cat("\n95% Confidence Interval Estimates of MEANS")
out_mean
```

```
##
## 95% Confidence Interval Estimates of MEANS
##   low      fit      lwr      upr
## 1   0 7475.849 6027.460 8924.237
## 2  10 6024.309 5021.907 7026.711
## 3  20 4572.769 3821.545 5323.994
## 4  30 3121.230 2240.084 4002.375
```

Report Your Model - 90% Confidence Interval Estimates of INDIVIDUAL OBSERVATIONS

```
#-- Predictions for new values
newvalues <- data.frame(low=c(0,10,20,30))

# Confidence Interval Estimates of Individual Predictions
estimated_individual <- predict(lm_fit,newdata=newvalues, interval="prediction", level=.90)
out_individual <- cbind(newvalues,estimated_individual)
cat("\n90% Confidence Interval Estimates of INDIVIDUAL OBSERVATIONS")
out_individual
```

```
##
## 90% Confidence Interval Estimates of INDIVIDUAL OBSERVATIONS
##   low      fit      lwr       upr
## 1   0 7475.849 3992.835 10958.863
## 2  10 6024.309 2651.059  9397.560
## 3  20 4572.769 1244.775  7900.764
## 4  30 3121.230 -228.631  6471.090
```

7. Commands Used in This Illustration

Fit model. Save as object.

```
fitobject <- lm(yvar ~ xvar, data=dataname)
model_simple <- lm(draft_number ~ day_birth, data=draftlottery1970)
```

Return names of model fit object.

```
names(fitobject)
names(model_simple)
```

Show model output.

```
summary(fitobject)
summary(model_simple)
```

Show show regression estimates and confidence intervals.

```
cbind(coef(fitobject),confint(fitobject))
cbind(coeff(model_simple), confint(model_simple))
```

Show analysis of variance table.

```
anova(fitobject)
anova(model_simple)
```

Nice tabular report using package {stargazer}

```
library(stargazer)
stargazer(fitobject, type="text")
stargazer(model_simple, type="text")
```