SetUp2: For the Toluca Company ex, p.19

Yes

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Background of the example

The Toluca Company manufactures refrigeration equipment as well as many replacement parts. In the past, one of the replacement parts has been produced periodically in lots of varying sizes.

When a cost improvement program was undertaken, company officials wished to determine the optimum lot size for producing this part.

The production of this part involves setting up the production process (which must be done no matter what is the lot size) and machining and assembly operations.

One key input for the model to ascertain the optimum lot size was the relationship between lot size and the labor hours required to produce the lot.

To determine this relationship, data on lot size and work hours for 25 recent production runs were utilized.

The production conditions were stable during the six-month period in which the 25 runs were made and were expected to continue to be the same during the next three years, the planning period for which the cost inprovement program was being conducted.

```
# Scan a data set into R
# Change the working directory to the appropriate one
# Also before scan the data set into R, check the data set first to see
# the number of columns it has, you need it for assigning ncol below.

ch01ta01<-matrix(scan("CH01TA01.txt"),ncol=2, byrow=T);
```

```
ch01ta01
```

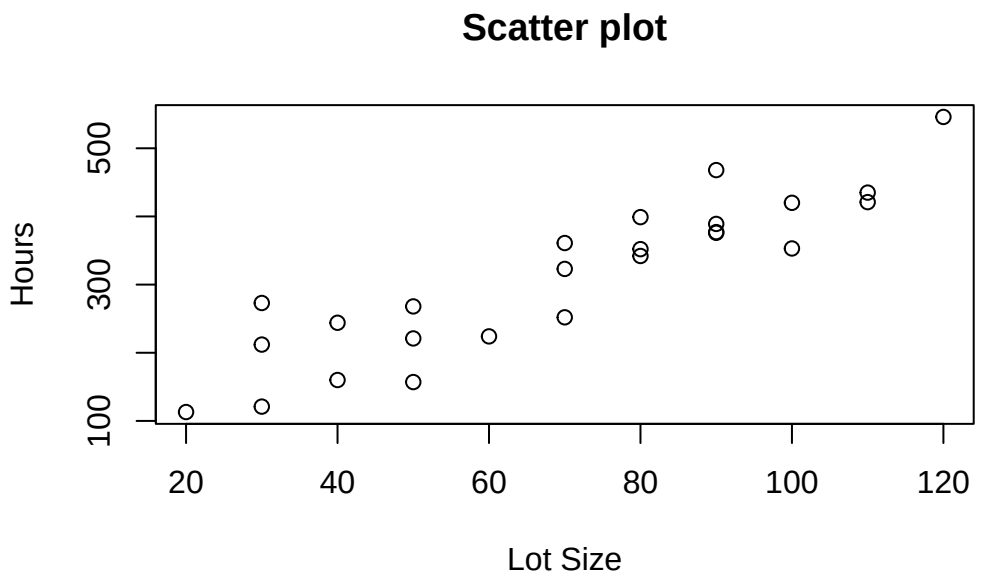
```
      [,1] [,2]
 [1,]   80  399
 [2,]   30  121
 [3,]   50  221
 [4,]   90  376
 [5,]   70  361
 [6,]   60  224
 [7,]  120  546
 [8,]   80  352
 [9,]  100  353
[10,]   50  157
[11,]   40  160
[12,]   70  252
[13,]   90  389
[14,]   20  113
[15,]  110  435
[16,]  100  420
[17,]   30  212
[18,]   50  268
[19,]   90  377
[20,]  110  421
[21,]   30  273
[22,]   90  468
[23,]   40  244
[24,]   80  342
[25,]   70  323
```

```
dum<-data.frame(x=ch01ta01[,1],y=ch01ta01[,2]) # to name the columns #
dum       # do you see any difference in dum and ch01ta01? #
```

```
    x   y
1  80 399
2  30 121
3  50 221
4  90 376
5  70 361
6  60 224
7 120 546
```

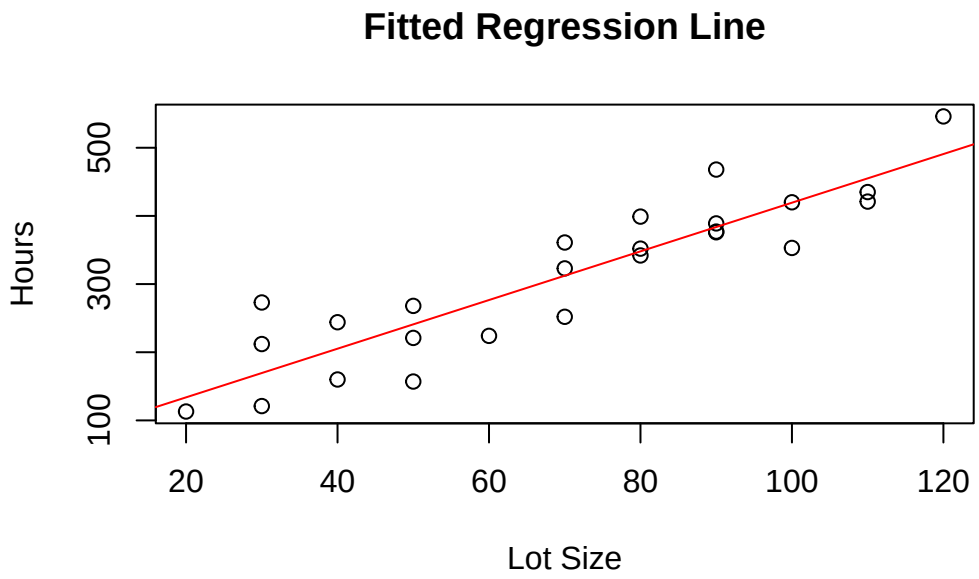
```
8   80 352
9  100 353
10  50 157
11  40 160
12  70 252
13  90 389
14  20 113
15 110 435
16 100 420
17  30 212
18  50 268
19  90 377
20 110 421
21  30 273
22  90 468
23  40 244
24  80 342
25  70 323
```

```
attach(dum)
# opar <- par(mfrow = c(1,2), oma = c(0, 0, 1.1, 0))
plot(x,y,xlab="Lot Size", ylab="Hours", main="Scatter plot");
```



```r
fm<-lm(y~x,dum)

plot(x,y,xlab="Lot Size", ylab="Hours", main="Fitted Regression Line");
abline(coef(fm), col="red");
```




```r
mean(x);          # sample mean
```

```
[1] 70
```

```r
var(x);           # sample variance
```

```
[1] 825
```

```r
sum((x-mean(x))^2)/(length(x)-1);
```

```
[1] 825
```

S-L-R model:

$$Y = \beta_0 + \beta_1 x + \epsilon$$

Fitted regression line:

$$\hat{Y} = b_0 + b_1 x$$

```r
# To get Table 1.2 on p.22 ....... #
ta12<-data.frame("x"=x,"y"=y,"y_hat"=fitted(fm),"residual"=resid(fm))
ta12
```

```
     x   y    y_hat    residual
1   80 399 347.9820  51.0179798
2   30 121 169.4719 -48.4719192
3   50 221 240.8760 -19.8759596
4   90 376 383.6840  -7.6840404
5   70 361 312.2800  48.7200000
6   60 224 276.5780 -52.5779798
7  120 546 490.7901  55.2098990
8   80 352 347.9820   4.0179798
9  100 353 419.3861 -66.3860606
10  50 157 240.8760 -83.8759596
11  40 160 205.1739 -45.1739394
12  70 252 312.2800 -60.2800000
13  90 389 383.6840   5.3159596
14  20 113 133.7699 -20.7698990
15 110 435 455.0881 -20.0880808
16 100 420 419.3861   0.6139394
17  30 212 169.4719  42.5280808
18  50 268 240.8760  27.1240404
19  90 377 383.6840  -6.6840404
20 110 421 455.0881 -34.0880808
21  30 273 169.4719 103.5280808
22  90 468 383.6840  84.3159596
23  40 244 205.1739  38.8260606
24  80 342 347.9820  -5.9820202
25  70 323 312.2800  10.7200000
```

```r
# For Fig. 1.11, p.20 and Fig. 2.2, p.46 ..... #
summary(fm)     # Please compare the outputs of R and those in text
```

```
Call:
lm(formula = y ~ x, data = dum)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-83.876 -34.088  -5.982  38.826 103.528

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   62.366     26.177   2.382   0.0259 *
x              3.570      0.347  10.290 4.45e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 48.82 on 23 degrees of freedom
Multiple R-squared:  0.8215,    Adjusted R-squared:  0.8138
F-statistic: 105.9 on 1 and 23 DF,  p-value: 4.449e-10
```

```
anova(fm)
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 252378  252378  105.88 4.449e-10 ***
Residuals 23  54825    2384
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
names(fm)  #See what you can get from fm
```

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

```
# Load Mass package to construct confidence intervals for
# beta's

confint(fm)    # conf. int.'s for reg. coeff.'s  default: 95% #
```

```
                2.5 %     97.5 %
(Intercept) 8.213711 116.518006
x           2.852435   4.287969
```

```r
confint(fm, level=0.9)
```

```
                 5 %       95 %
(Intercept) 17.501100 107.230617
x            2.975536   4.164868
```

```r
# Estimation of mean response, its confidence intervals  Ex. on p. 55, p. 59
# and prediction intervals

# This is how you
# construct confidence/prediction  intervals  for the mean response at given predictor x's

new<- data.frame(x=seq(10,210,10))
new
```

```
    x
1   10
2   20
3   30
4   40
5   50
6   60
7   70
8   80
9   90
10 100
11 110
12 120
13 130
14 140
15 150
16 160
17 170
18 180
19 190
20 200
21 210
```

```r
predict(lm(y~x), new, se.fit = TRUE)
```

```
$fit
       1         2         3         4         5         6         7         8
 98.06788 133.76990 169.47192 205.17394 240.87596 276.57798 312.28000 347.98202
       9        10        11        12        13        14        15        16
383.68404 419.38606 455.08808 490.79010 526.49212 562.19414 597.89616 633.59818
      17        18        19        20        21
669.30020 705.00222 740.70424 776.40626 812.10828

$se.fit
       1         2         3         4         5         6         7         8
22.994596 19.907858 16.969741 14.272328 11.979336 10.362798  9.764662 10.362798
       9        10        11        12        13        14        15        16
11.979336 14.272328 16.969741 19.907858 22.994596 26.177434 29.425203 32.718573
      17        18        19        20        21
36.045047 39.396240 42.766342 46.151210 49.547818

$df
[1] 23

$residual.scale
[1] 48.82331
```

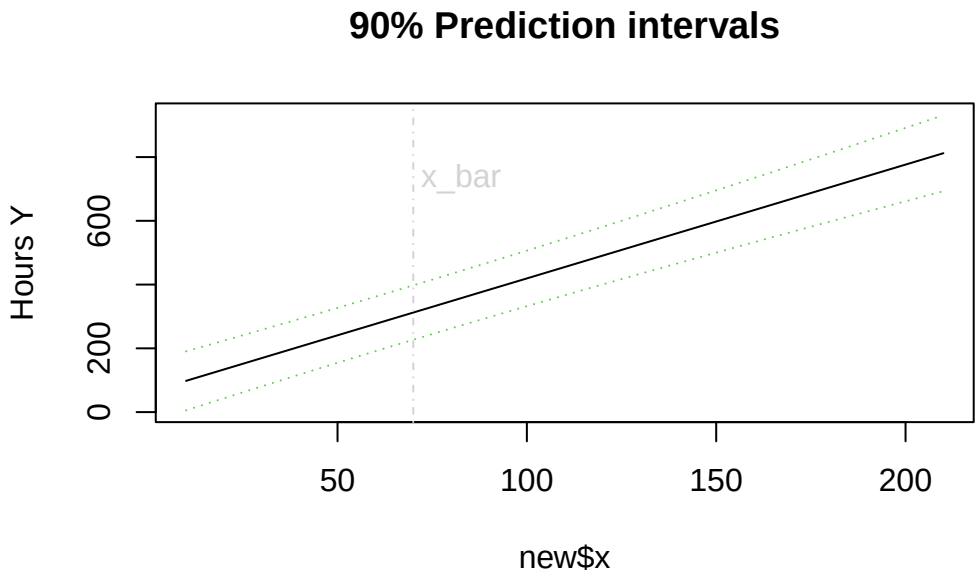
```r
pred.w.plim <- predict(lm(y~x), new, interval="prediction",level=0.9)
pred.w.clim <- predict(lm(y~x), new, interval="confidence",level=0.9)

cbind(new$x,pred.w.plim) # prediction intervals for x at seq(10,210,10)
```

```
             fit        lwr      upr
1   10  98.06788   5.574898 190.5609
2   20 133.76990  43.404189 224.1356
3   30 169.47192  80.884691 258.0591
4   40 205.17394 117.995055 292.3528
5   50 240.87596 154.717128 327.0348
6   60 276.57798 191.037019 362.1189
7   70 312.28000 226.945990 397.6140
8   80 347.98202 262.441059 433.5230
9   90 383.68404 297.525209 469.8429
10 100 419.38606 332.207177 506.5649
11 110 455.08808 366.500853 543.6753
12 120 490.79010 400.424391 581.1558
13 130 526.49212 433.999140 618.9851
```

```
14 140 562.19414 467.248541 657.1397
15 150 597.89616 500.197093 695.5952
16 160 633.59818 532.869465 734.3269
17 170 669.30020 565.289788 773.3106
18 180 705.00222 597.481138 812.5233
19 190 740.70424 629.465192 851.9433
20 200 776.40626 661.262029 891.5505
21 210 812.10828 692.890045 931.3265
```

```
# Graphical display
matplot(new$x,pred.w.plim,
        col=c(1,3,3), lty=c(1,3,3),type="l", #ylim=c(70,500),
main="90% Prediction intervals",ylab="Hours Y")
abline(v=mean(x), col="lightgray", lty=4);
text(70,700, "x_bar",col="lightgray",adj=c(-.1,-.1))
```

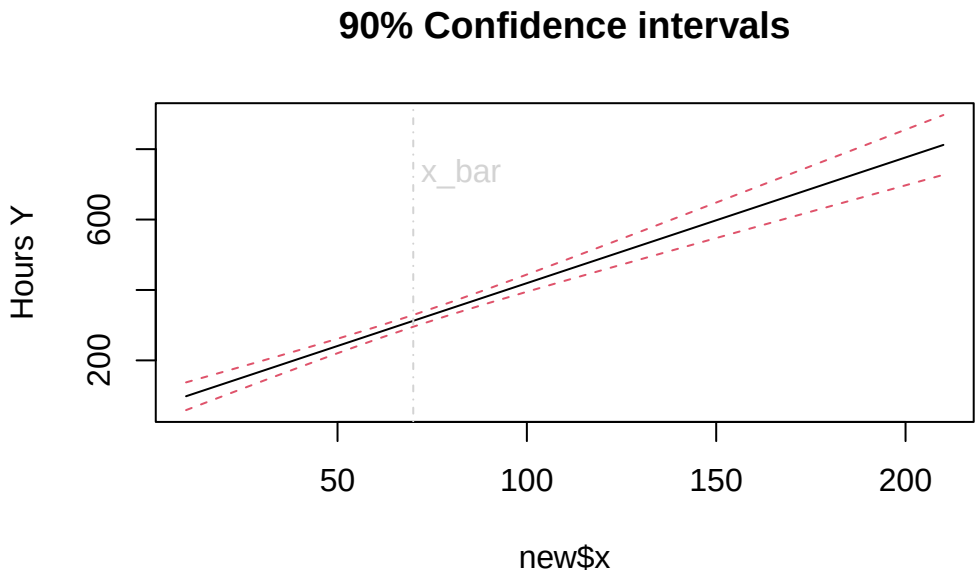



```
cbind(new$x,pred.w.clim) # confidence intervals for x at seq(10,210,10)
```

```
            fit       lwr      upr
1   10  98.06788  58.65809 137.4777
2   20 133.76990  99.65039 167.8894
3   30 169.47192 140.38796 198.5559
4   40 205.17394 180.71300 229.6349
5   50 240.87596 220.34492 261.4070
```

```
6   60 276.57798 258.81747 294.3385
7   70 312.28000 295.54462 329.0154
8   80 347.98202 330.22151 365.7425
9   90 383.68404 363.15300 404.2151
10 100 419.38606 394.92512 443.8470
11 110 455.08808 426.00412 484.1720
12 120 490.79010 456.67059 524.9096
13 130 526.49212 487.08234 565.9019
14 140 562.19414 517.32938 607.0589
15 150 597.89616 547.46514 648.3272
16 160 633.59818 577.52275 689.6736
17 170 669.30020 607.52362 731.0768
18 180 705.00222 637.48213 772.5223
19 190 740.70424 667.40823 814.0003
20 200 776.40626 697.30902 855.5035
21 210 812.10828 727.18969 897.0269
```

```r
# Graphical display
matplot(new$x,pred.w.clim,
        col=c(1,2,2), lty=c(1,2,2),type="l",# ylim=c(70,500),
main="90% Confidence intervals",ylab="Hours Y")
abline(v=mean(x), col="lightgray", lty=4);
text(70,700, "x_bar",col="lightgray",adj=c(-.1,-.1))
```



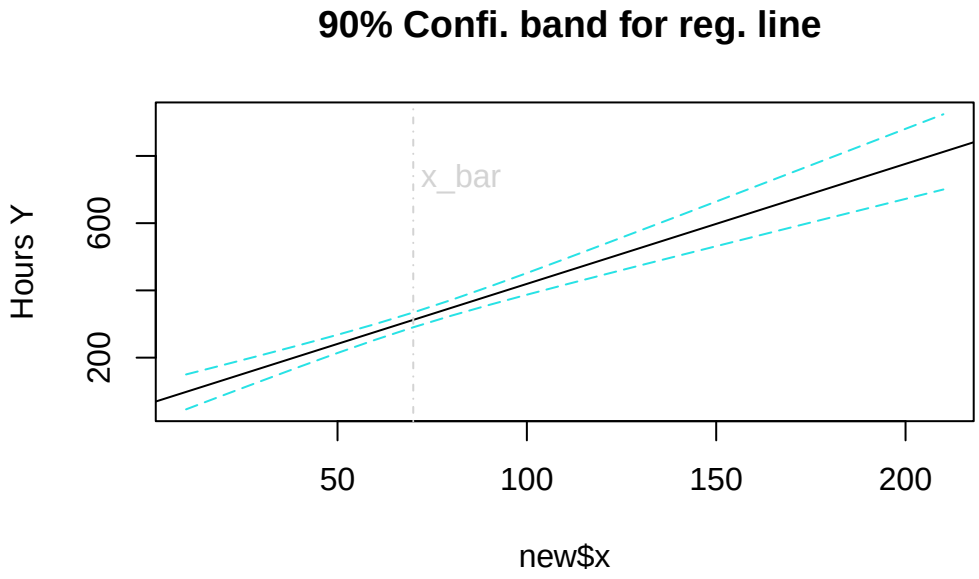
```
# Confidence band for the entire reg. line  Ex. on p. 62, Fig. 2.6

width<-function(n,xh,alpha){
sqrt(2*qf(1-alpha,2,n-2)*sum((y-fitted(fm))^2)/(n-2)*(1/n+(xh-mean(x))^2/((n-1)*var(x))))
}
n<-length(y)
u<-coef(fm)[1]+coef(fm)[2]*new$x+width(n,new$x,0.1)
l<-coef(fm)[1]+coef(fm)[2]*new$x-width(n,new$x,0.1)
band<-cbind(l,u)

cbind(new$x,band)
```

```
                l        u
 [1,]  10  46.14600 149.9898
 [2,]  20  88.81789 178.7219
 [3,]  30 131.15419 207.7897
 [4,]  40 172.94698 237.4009
 [5,]  50 213.82658 267.9253
 [6,]  60 253.17875 299.9772
 [7,]  70 290.23136 334.3286
 [8,]  80 324.58279 371.3813
 [9,]  90 356.63466 410.7334
[10,] 100 387.15910 451.6130
[11,] 110 416.77035 493.4058
[12,] 120 445.83809 535.7421
[13,] 130 474.57025 578.4140
[14,] 140 503.08541 621.3029
[15,] 150 531.45396 664.3384
[16,] 160 559.71954 707.4768
[17,] 170 587.91037 750.6900
[18,] 180 616.04538 793.9591
[19,] 190 644.13770 837.2708
[20,] 200 672.19668 880.6158
[21,] 210 700.22914 923.9874
```

```
# Graphical display
matplot(new$x,band,col=c(5,5),lty=c(5,5),type="l", #ylim=c(70,500),
main=" 90% Confi. band for reg. line",ylab="Hours Y")
abline(coef(fm),col=1)
abline(v=mean(x), col="lightgray", lty=4);
text(70,700, "x_bar",col="lightgray",adj=c(-.1,-.1))
```

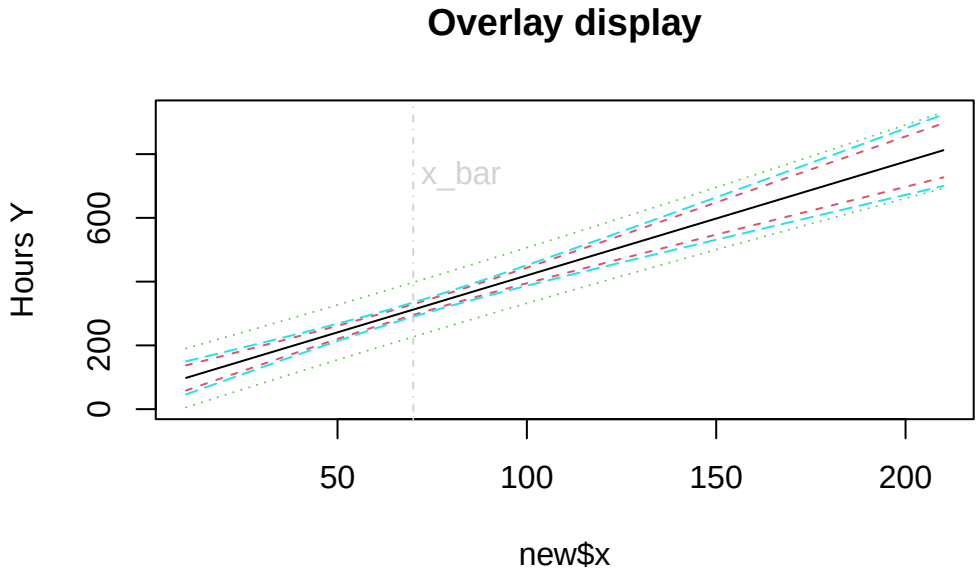



```
# It is hard to compare among plots......


# Graphical display these intervals for easy comparison

# Overlay CI, PI, Conf band at those x values

matplot(new$x,cbind(pred.w.clim, pred.w.plim[,-1],band),
        col=c(1,2,2,3,3,5,5), lty=c(1,2,2,3,3,5,5),type="l",
        main="Overlay display",ylab="Hours Y")
abline(v=mean(x), col="lightgray", lty=4);
text(70,700, "x_bar",col="lightgray",adj=c(-.1,-.1))
```




```
# Before you quit R, clean up the objects, please. #
detach()

# rm(list=ls())
# q()

# Also try the "Save History" under "File" of the R Console window,
# and you may find out how Yes prepare these handouts.....  :> #
```