Prediction intervals with R

```
> kars = read.table("http://www.utstat.utoronto.ca/~brunner/data/legal/mcars4.data")
> head(kars); attach(kars)
  Cntry lper100k weight length
1    US     19.8   2178   5.92
2 Japan      9.9   1026   4.32
3    US     10.8   1188   4.27
4    US     12.5   1444   5.11
5    US     12.5   1485   5.03
6    US     12.5   1485   5.03

> Cntry = factor(Cntry); contrasts(Cntry)
      Japan US
Europ     0  0
Japan     1  0
US        0  1
> fullmodel = lm(lper100k ~ weight + length + Cntry)
> summary(fullmodel)

Call:
lm(formula = lper100k ~ weight + length + Cntry)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5063 -0.8813  0.0147  1.3043  2.9432 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -5.789215   2.855736  -2.027 0.045441 *  
weight       0.005457   0.001472   3.707 0.000352 ***
length       2.345968   0.980329   2.393 0.018676 *  
CntryJapan   0.506517   0.660158   0.767 0.444826    
CntryUS     -1.487722   0.575633  -2.584 0.011274 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.703 on 95 degrees of freedom
Multiple R-squared:  0.7431,	Adjusted R-squared:  0.7323 
F-statistic: 68.71 on 4 and 95 DF,  p-value: < 2.2e-16

> fullmodel$df.residual # n-k-1
[1] 95
```

```
>
> ###### Predictions, confidence intervals and prediction intervals ######
>
> # Predict litres per 100 km for a Japanese car weighing
> # 1295kg, 4.52m long (1990 Toyota Camry)
>
> betahat = fullmodel$coefficients; betahat
 (Intercept)       weight       length   CntryJapan      CntryUS 
-5.789214693  0.005456609  2.345968436  0.506517030 -1.487721833 
> x0 = c(1,1295,4.52,1,0)
> yhat = sum(x0*betahat); # x0-prime betahat
> yhat
[1] 12.38739
>
> # Confidence interval for E(x0-prime beta)
> # First the hard way
```

$$\mathbf{x}_0'\widehat{\boldsymbol{\beta}} \pm t_{\alpha/2}\sqrt{MSE\,\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0}$$

```
>
> tcrit = qt(0.975,df=fullmodel$df.residual) # t_alpha/2
> MSE.XpXinv = vcov(fullmodel)
> x0 = as.matrix(x0) # Now it's a column vector
> me95 = tcrit * sqrt( as.numeric(t(x0) %*% MSE.XpXinv %*% x0) )
> lower95 = yhat - me95; upper95 = yhat + me95
> c(lower95, upper95) # 95% Confidence interval for x0-prime beta
[1] 11.37128 13.40349
>
> # Use the predict function
> # help(predict.lm)
>
> camry1990 = data.frame(weight=1295,length=4.52,Cntry='Japan')
> camry1990
  weight length Cntry
1   1295   4.52 Japan
> predict(fullmodel,newdata=camry1990) # Compare yhat = 12.38739
       1 
12.38739 
> predict(fullmodel,newdata=camry1990, interval='confidence')
       fit      lwr      upr
1 12.38739 11.37128 13.40349
>
```

```
> # With 95 percent prediction interval (95 is default)
```

$$\mathbf{x}_0'\widehat{\boldsymbol{\beta}} \pm t_{\alpha/2}\sqrt{MSE\,(1+\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0)}$$

```
> predict(fullmodel,newdata=camry1990, interval='prediction')
       fit      lwr      upr
1 12.38739 8.856608 15.91817
>

>  # Multiple predictions
> cadillac1990 = data.frame(weight=1800,length=5.22,Cntry='US')
> volvo1990 = data.frame(weight=1371,length=4.823,Cntry='Europ')
> newcars = rbind(camry1990,cadillac1990,volvo1990); newcars
  weight length Cntry
1   1295  4.520 Japan
2   1800  5.220    US
3   1371  4.823 Europ
>
> is.data.frame(newcars)
[1] TRUE
>
> predict(fullmodel,newdata=newcars, interval='prediction')
       fit       lwr      upr
1 12.38739  8.856608 15.91817
2 14.79091 11.354379 18.22745
3 13.00640  9.481598 16.53121
>
> # With cell means dummy variable coding
> cellmeans = lm(lper100k ~ 0+Cntry+weight+length)
> summary(cellmeans) # Beware! R-squared was 0.7431 for an equivalent model.

Call:
lm(formula = lper100k ~ 0 + Cntry + weight + length)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5063 -0.8813  0.0147  1.3043  2.9432 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
CntryEurop -5.789215   2.855736  -2.027 0.045441 *  
CntryJapan -5.282698   2.926052  -1.805 0.074179 .  
CntryUS    -7.276937   3.006354  -2.421 0.017399 *  
weight      0.005457   0.001472   3.707 0.000352 ***
length      2.345968   0.980329   2.393 0.018676 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

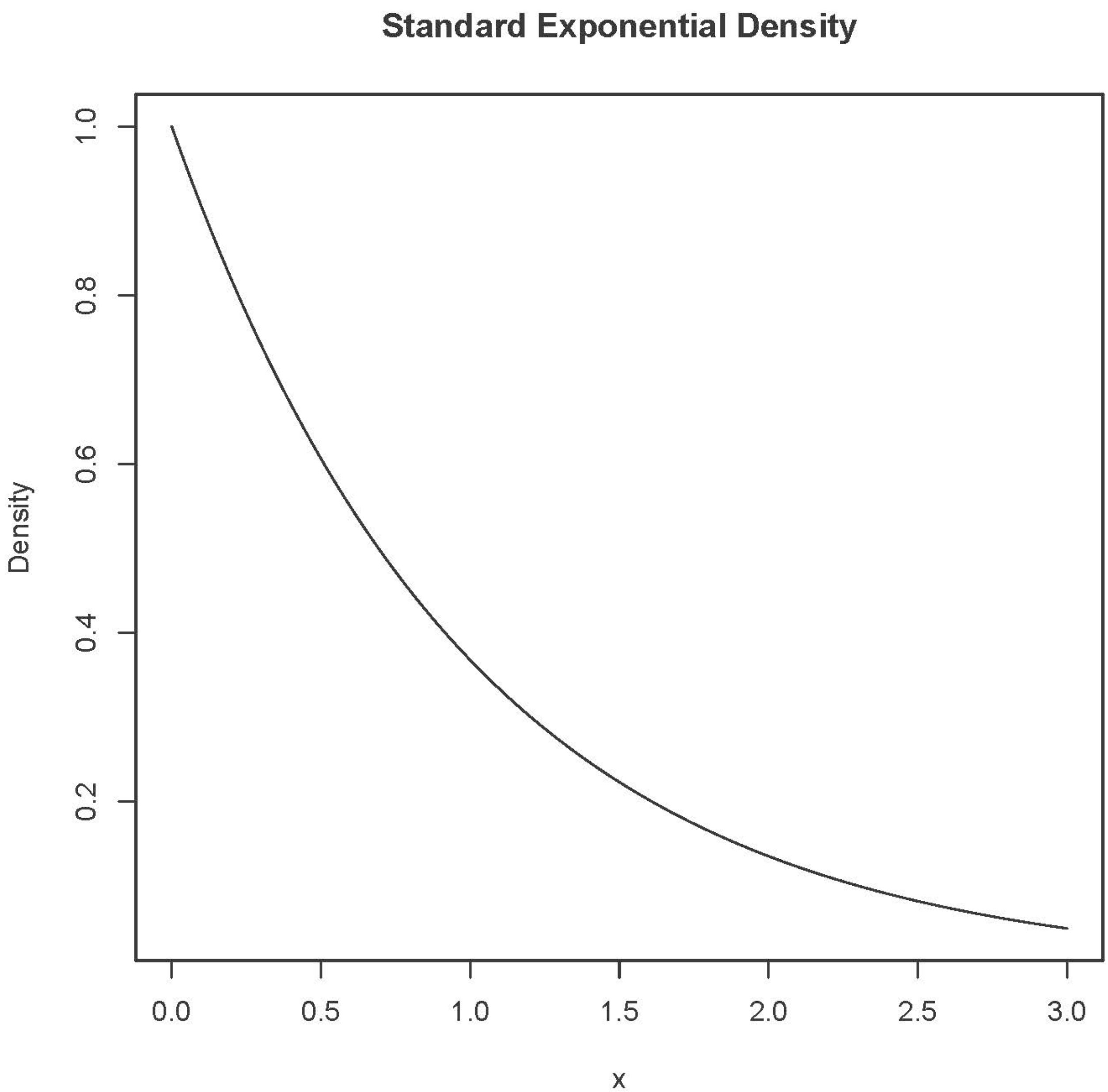
```
Residual standard error: 1.703 on 95 degrees of freedom
Multiple R-squared:  0.9829,  Adjusted R-squared:  0.982 
F-statistic:  1094 on 5 and 95 DF,  p-value: < 2.2e-16

> predict(cellmeans,newdata=newcars, interval='prediction')
       fit       lwr      upr
1 12.38739  8.856608 15.91817
2 14.79091 11.354379 18.22745
3 13.00640  9.481598 16.53121

> 
> # For comparison again, model with intercept
> predict(fullmodel,newdata=newcars, interval='prediction')
       fit       lwr      upr
1 12.38739  8.856608 15.91817
2 14.79091 11.354379 18.22745
3 13.00640  9.481598 16.53121
> 
> # "Predict" the whole data set
> # predict(fullmodel, interval='prediction')
> data.frame( kars ,predict(fullmodel, interval='prediction') )

    Cntry lper100k weight length       fit       lwr      upr
1      US     19.8   2178   5.92 18.495691 15.012790 21.97859
2   Japan      9.9   1026   4.32 10.450367  6.940724 13.96001
3      US     10.8   1188   4.27  9.222800  5.747978 12.69762
4      US     12.5   1444   5.11 12.590305  9.162660 16.01795
5      US     12.5   1485   5.03 12.626349  9.219284 16.03341
6      US     12.5   1485   5.03 12.626349  9.219284 16.03341
7   Europ     10.4    972   4.37  9.766491  6.236956 13.29603
8      US     13.2   1665   5.44 14.570386 11.135730 18.00504
9   Europ     17.0   1539   4.88 14.056832 10.515850 17.59782
10     US      9.2   1003   4.32  8.330626  4.870298 11.79095

                        Skipping . . .

97  Europ     14.0   1426   4.90 13.487155  9.955616 17.01869
98     US      9.5    990   4.19  7.954714  4.486636 11.42279
99     US     14.9   1660   5.38 14.402344 10.979352 17.82534
100    US     13.2   1498   5.11 12.884962  9.471886 16.29804
Warning message:
In predict.lm(fullmodel, interval = "prediction") :
  predictions on current data refer to _future_ responses

>
```

Some interesting things you will not be asked to do

```
> # Predict each observation based on the others. How many are in the
> # prediction interval?
>
> dim(kars)
[1] 100   4
> n = dim(kars)[1]
> in_interval = logical(n) # Logical, all FALSE to start
>
> for(j in 1:n)
+     {
+     kars99 = kars[-j,]
+     model99 = lm(lper100k ~ weight + length + Cntry, data=kars99)
+     xzero = kars[j,]
+     yzero = kars$lper100k[j]
+     pint = predict(model99,newdata=xzero, interval='prediction')
+          # (fit, lwr, upr)
+     lwr = pint[2]; upr = pint[3]
+     in_interval[j] = (yzero > lwr) && (yzero < upr)
+     } # Next j
>
> table(in_interval)
in_interval
FALSE  TRUE
    5    95
> mean(in_interval)
[1] 0.95
```

```
>
> ################################ SAT Data ################################
>
> rm(list=ls()) # Remove everything
> sat =
read.table("http://www.utstat.utoronto.ca/~brunner/data/legal/openSAT.data.txt")
> head(sat)
  VERBAL MATH  GPA
1    578  567 2.68
2    474  653 2.51
3    546  657 1.95
4    664  686 2.81
5    600  619 2.79
6    488  738 2.36
> n = dim(sat)[1]; n
[1] 200
> in_interval = logical(n) # Logical, all FALSE to start
>
> for(j in 1:n)
+     {
+     sat199 = sat[-j,]
+     model199 = lm(GPA ~ VERBAL+MATH, data=sat199)
+     xzero = sat[j,]
+     yzero = sat$GPA[j]
+     pint = predict(model199,newdata=xzero, interval='prediction')
+     lwr = pint[2]; upr = pint[3]
+     in_interval[j] = (yzero > lwr) && (yzero < upr)
+     } # Next j
>
> table(in_interval)
in_interval
FALSE  TRUE 
    8   192 
> mean(in_interval)
[1] 0.96
>
```

```
> # Try some simulated data
> # Remember the exponential distribution? Right skewed, E(X) = 1
> x = seq(from=0,to=3,by=0.01); Density = dexp(x); plot(x,Density,type='l')
> title('Standard Exponential Density')
```



```
>
> ############## Simulated data, exponential errors ########################
>
> rm(list=ls()) # Remove everything
> beta0 = 2; beta1=1; n=300
> nsim = 10000; in_interval = logical(nsim)
>
> set.seed(9999)
>
> for(j in 1:nsim)
+     {
+     x = runif(n,-3,3)
+     epsilon = rexp(n)-1 # Subtract one to make E(epsilon) = 0
+     y = beta0 + beta1*x + epsilon # These are all length n
+     regmod = lm(y~x)
+     # Simulate a new x0 and y0
+     x0 = runif(1,-3,3); y0 = beta0 + beta1*x0 + rexp(1)-1 # Length one
+     # Predict
+     newx = data.frame(x=x0)
+     pint = predict(regmod,newdata=newx, interval='prediction') # (fit, lwr, upr)
+     lwr = pint[2]; upr = pint[3]
+     in_interval[j] = (y0 > lwr) && (y0 < upr)
+     } # Next j

>
> table(in_interval)
in_interval
FALSE  TRUE
  566  9434
> mean(in_interval)
[1] 0.9434
```

```
> ############# Simulated data, Cauchy errors (extreme outliers) ##########
```

$$\int_{-\infty}^{\infty} f(x)\,dx = 1, \text{ but } \int_{0}^{\infty} x f(x)\,dx = \infty$$

```
> rm(list=ls()) # Remove everything
> beta0 = 2; beta1=1; n=300
> nsim = 10000; in_interval = logical(nsim)
>
> set.seed(9999)
>
> for(j in 1:nsim)
+     {
+     x = runif(n,-3,3)
+     epsilon = rcauchy(n)
+     y = beta0 + beta1*x + epsilon # These are all length n
+     regmod = lm(y~x)
+     # Simulate a new x0 and y0
+     x0 = runif(1,-3,3); y0 = beta0 + beta1*x0 + rcauchy(1) # Length one
+     # Predict
+     newx = data.frame(x=x0)
+     pint = predict(regmod,newdata=newx, interval='prediction') # (fit, lwr, upr)
+     lwr = pint[2]; upr = pint[3]
+     in_interval[j] = (y0 > lwr) && (y0 < upr)
+     } # Next j

>
> table(in_interval)
in_interval
FALSE  TRUE
  200  9800
> mean(in_interval)
[1] 0.98
```

The prediction intervals from mainstream regression are almost unbelievably good. Here is a challenge to machine learning and other modern data science methods. I know you can give me a prediction, but can you give me a prediction interval around it?

```
>
> ############ Back to the SAT data ##############
>
> rm(list=ls()) # Remove everything
> sat =
read.table("http://www.utstat.utoronto.ca/~brunner/data/legal/openSAT.data.txt")
> satmodel = lm(GPA ~ VERBAL+MATH, data=sat)
>
> cbind(sat$GPA, predict(satmodel, interval = 'prediction') )
              fit       lwr      upr
1   2.68 2.507025 1.4187097 3.595340
2   2.51 2.352870 1.2605612 3.445178
3   1.95 2.522963 1.4373577 3.608569
4   2.81 2.824113 1.7374065 3.910819
5   2.79 2.609641 1.5248684 3.694414
6   2.36 2.469943 1.3712290 3.568658
7   2.17 2.829482 1.7418822 3.917082
8   2.77 2.543535 1.4574587 3.629611
9   2.15 2.781911 1.6952499 3.868573
10  2.76 2.851223 1.7621232 3.940323
11  3.30 2.814391 1.7276517 3.901130
12  2.69 2.560095 1.4654324 3.654758
13  3.57 2.558887 1.4741772 3.643598
14  3.99 2.741705 1.6531451 3.830264
15  2.85 2.216594 1.1205590 3.312630

            Skipping . . .

193 3.16 2.637587 1.5479546 3.727219
194 2.02 2.656740 1.5609353 3.752545
195 3.23 2.573979 1.4889577 3.658999
196 1.96 2.238023 1.1400171 3.336029
197 2.32 2.437301 1.3505489 3.524054
198 2.04 2.502169 1.4102075 3.594130
199 2.40 2.458447 1.3716580 3.545236
200 2.38 2.317449 1.2203254 3.414572

Warning message:
In predict.lm(satmodel, interval = "prediction") :
  predictions on current data refer to _future_ responses
```
