Homework Assignment 1: What’s That Got to Do with the Price of Condos in California?

36-402, Advanced Data Analysis, Spring 2011

SOLUTIONS

The easiest way to load the data is with \texttt{read.table}, but you have to tell R that the first line names the variables:

\begin{verbatim}
> calif = read.table("~/teaching/350/hw/06/cadata.dat",header=TRUE)
> dim(calif)
[1] 20640  9
> is.data.frame(calif)
[1] TRUE
> summary(calif)

MedianHouseValue  MedianIncome  MedianHouseAge  TotalRooms
Min. : 14999      Min. : 0.4999      Min. : 1.00      Min. : 2
1st Qu.:119600    1st Qu.: 2.5634     1st Qu.:18.00     1st Qu.: 1448
Median :179700     Median : 3.5348     Median :29.00      Median : 2127
Mean :206856       Mean : 3.8707      Mean :28.64       Mean : 2636
3rd Qu.:264725    3rd Qu.: 4.7432     3rd Qu.:37.00      3rd Qu.: 3148
Max. :500001      Max. :15.0001      Max. :52.00       Max. :39320

TotalBedrooms  Population  Households  Latitude
Min. : 1.0        Min. : 3        Min. : 1.0        Min. :32.54
1st Qu.:295.0    1st Qu.: 787      1st Qu.:280.0      1st Qu.:33.93
Median :435.0    Median :1166     Median :409.0       Median :34.26
Mean :537.9      Mean :1425      Mean :499.5        Mean :35.63
3rd Qu.:647.0    3rd Qu.:1725    3rd Qu.:605.0       3rd Qu.:37.71
Max. :6445.0     Max. :35682     Max. :6082.0       Max. :41.95

Longitude
Min. :-124.3
1st Qu.:-121.8
Median :-118.5
Mean :-119.6
3rd Qu.:-118.0
Max. :-114.3

So far nothing looks crazy\footnote{But, what’s up with the smallest census tracts being so small? We’d better hope that the}
\end{verbatim}
1. **Answer:** See figures. I am not very satisfied with either of the 3D plots — it’s too hard to see what’s going on, except for the fact that the coastal areas are expensive, especially around San Francisco and LA — but maybe you can do better.

2. **Answer:**

```r
fit = lm(MedianHouseValue ~ ., data=calif)
```

Handily, there is a function to extract the coefficients:

```r
> signif(coefficients(fit),3)
  (Intercept) MedianIncome MedianHouseAge TotalRooms
  -3.59e+06   4.02e+04     1.16e+03   -8.18e+00
TotalBedrooms Population Households Latitude
  1.13e+02   -3.85e+01     4.83e+01   -4.26e+04
Longitude
  -4.28e+04
```

where I’ve rounded to three significant digits[^1]. The coefficients mean that house prices increase with income; that older housing is more expensive; that tracts with more rooms have lower prices, but ones with more bedrooms have higher prices; that tracts with more people have lower prices, but ones with more families have higher prices. Finally, prices increase from north to south (towards the big cities), and from east to west (towards the coast).

To get $R^2$ I use the `summary` function:

```r
> signif(summary(fit)$r.squared,digits=3)
[1] 0.637
```

Finally, the mean squared error:

```r
> signif(mean((residuals(fit))^2),digits=3)
[1] 4.83e+09
```

This is somewhat more comprehensible when we take the square root, so it’s on the same scale as the response variable (dollars, rather than dollars squared):

```r
> signif(sqrt(mean((residuals(fit))^2)),digits=3)
[1] 69500
```

one household, the population 3, the 1 bedroom and 2 rooms all come from the same record. But in fact they don’t, and are almost certainly data-entry errors. For example, the row with `TotalBedrooms==1` is 16172, which records a population of 13 in one household — with a house value of $500,001. This isn’t mathematically impossible, but it’s remarkably implausible.

[^1]: Why three? Because one expects the relative error in parametric estimates to be $\propto \frac{1}{\sqrt{n}}$, and with $n = 2 \times 10^4$, this gives a relative error of about 1 in 100.
Figure 1: Color indicates price, in $50,000 increments. (This was convenient because the maximum price recorded was $500,001.) See `help(terrain.colors)` for the behavior of this color-making command. Note that colors here are indexed from 1, like other R vectors.
plot(calif$Longitude, calif$Latitude, pch=21,  
col=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],  
bg=terrain.colors(11)[1+floor(calif$MedianHouseValue/50e3)],  
cex=sqrt(calif$Population/median(calif$Population)),  
xlab="Longitude", ylab="Latitude", main="Median House Prices", sub="Circle area proportional to population")  
legend(x="topright", legend=(50*(1:11)), fill=terrain.colors(11))

Figure 2: Map with price indicated by color and population by the area of the circle. (To make area proportional to population, radius must be proportional to $\sqrt{\text{population}}$. Making radius proportional to population drastically exaggerates the differences; see Huff’s *How to Lie with Statistics*.) This shows that the census tracts are not precisely equal in population, though the range doesn’t seem unreasonable.
plot(calif$Longitude, calif$Latitude, pch=21,
      col=grey(0:10/11)[1+floor(calif$MedianHouseValue/50e3)],
      bg=grey(0:10/11)[1+floor(calif$MedianHouseValue/50e3)],
      xlab="Longitude", ylab="Latitude", main="Median House Prices")
legend(x="topright", legend=(50*(1:11)), fill=grey(0:10/11))

Figure 3: Greyscale. See help(grey) for the behavior of this color-making command, which differs from that of the terrain.colors family.
scatterplot3d(calif$Longitude, calif$Latitude, calif$MedianHouseValue, pch=20, scale.y=0.4, type="p", cex.symbol=0.1, xlab="Longitude", ylab="Latitude", zlab="Median House Value")

Figure 4: 3D scatterplot map of California housing prices.
scatterplot3d(calif$Longitude, calif$Latitude, calif$MedianHouseValue, pch=20, scale.y=0.4, type="h", cex.symbol=0.01, xlab="Longitude", ylab="Latitude", zlab="Median House Value")

Figure 5: As in the previous plot, but with vertical lines drawn from the $x - y$ plane to the appropriate $z$ value.
Many people used instead the estimate of the random variance provided by the linear regression:

\[
\text{signif((summary(fit)$\sigma)^2,digits=3)}
\]
\[
\text{signif(summary(fit)$\sigma,digits=3)}
\]

Strictly speaking, what \texttt{summary(fit)$\sigma$} gives is not the in-sample mean-squared error, but an estimate of what the generalization MSE should be,

\[
\hat{MSE}_{\text{generalization}} = \frac{n}{n - (p + 1)}MSE_{\text{in-sample}}
\]

where \( p \) is the number of input variables to the regression. The logic here is the same as that for scaling up the sample variance by \( n/(n - 1) \) when estimating the population variance: by estimating \( p + 1 \) parameters, we’ve used up \( p + 1 \) degrees of freedom, and really have only \( n - (p + 1) \) independent measurements of the residuals. With \( p = 8 \) and \( n = 20640 \), the difference is a factor of 1.000436, i.e., much too small to bother with.

3. **Answer:** To answer this, it’s not sufficient to just look at the size of the coefficients, because they’re being multiplied by features which are on very different scales! A reasonable thing to do to compare variables’ importance in a linear model is to multiply the coefficients by the standard deviations — the standard deviation is in some sense a typical or expected-sized change in the variable, so this compares the change in the prediction for typical changes in the inputs. (Of course there’s nothing magic about the standard deviation here, as opposed to say the inter-quartile range.) Let’s do that here:

\[
> \text{prediction.changes = input.sds} \ast \text{abs(coef(fit)[-1])}
\]

\[
> \text{signif(prediction.changes,3)}
\]

\[
\begin{array}{cccc}
\text{MedianIncome} & \text{MedianHouseAge} & \text{TotalRooms} & \text{TotalBedrooms} \\
1.90 & 12.60 & 2180.00 & 421.00 \\
\text{Population} & \text{Households} & \text{Latitude} & \text{Longitude} \\
1130.00 & 382.00 & 2.14 & 2.00 \\
\end{array}
\]

(I exclude the first column of \texttt{calif} from calculating the standard deviations because that’s the response variable, and exclude the first coefficient of \texttt{fit} because that’s the intercept.) By this standard, the two most important variables are latitude and longitude, followed by median income.
One-standard-deviation changes in the other variables make noticeably smaller changes in the linear prediction, though not by orders of magnitude or anything.

On balance, this seems like a sensible ordering. The old joke is that the three most important things in real estate are location, location and location, and the three biggest ones here are latitude (location), longitude (location), and income (proximity to rich people, i.e., location).

4. **Answer:** See Figure 6. Colloquially, there is no way these are Gaussian. Another way to see this is to compare the actual distribution of residuals to that of a Gaussian with the same mean and standard deviation (Figure 7). Compared to a Gaussian, the true residuals are asymmetric, more sharply peaked, and skewed to the right.

As for a formal test, there are a lot of tests for normality or Gaussianity. The one which is most convenient to use is the Shapiro-Wilk test, which is built into R in the form of the function `shapiro.test`. While the exact form of the Shapiro-Wilk test statistic is complicated, it basically measures the curvature of the Gaussian Q-Q plot. The one limitation is that the R implementation re-uses very old code which has an arbitrary limit of 5000 data points. (There is no such limit in the actual test procedure.) Since we have more than 5000 data points, we can just take a random sample of our residuals.

We need to be a little careful here: if the test rejects normality on the sample, that’s strong evidence that the residuals aren’t normal, but if it accepts normality, it might just be that the departure from a Gaussian distribution is too subtle to show up in only 5000 data points. (Sampling reduces power.)

```r
> shapiro.test(residuals(fit)[sample(nrow(calif),size=5000)])

Shapiro-Wilk normality test

data:  residuals(fit)[sample(nrow(calif), size = 5000)]
W = 0.9338, p-value < 2.2e-16

Let’s make sure the sample wasn’t just weird:

```r
> s = replicate(100,shapiro.test(residuals(fit)[sample(20640,5000)])$p.value)
> summary(s)

         Min.   1st Qu.    Median      Mean   3rd Qu.      Max.  
```

3 This uses the `density` command, which uses kernels to give a non-parametric estimate of the probability density function, rather than the regression function. We will look at these later in the class; for now you can treat it as a black box.

4 A C translation of Fortran code from the early 1980s.

5 If \( X_1, X_2, \ldots, X_n \) are all IID with common CDF \( F \), and we randomly select \( m < n \) of them, the selected random variables are still IID with common CDF \( F \).
Figure 6: Q – Q plot for the residuals against a Gaussian distribution. If the residuals were Gaussian, they should fall approximately on the straight line.

```r
qqnorm(residuals(fit))
qqline(residuals(fit))
```
plot(density(residuals(fit)))
curve(dnorm(x, mean(residuals(fit)), sd(residuals(fit))), add=TRUE, lty=2)

Figure 7: Estimated probability density function of the residuals (solid line), overlaid with the pdf of a Gaussian with the same mean and variance (dashed line).
So it’s incredibly unlikely the residuals came from a Gaussian distribution.

5. **Answer:** I found it clearer to make several maps: one of all the residuals, one of just the positive residuals, one of the negatives, and one of the magnitude of the residuals.

```r
resid = residuals(fit)
plot.calif <- function(z,subset=NULL,n.colors=11,x=calif$Longitude,
y=calif$Latitude,digits=3,pch=21,
xlab="Longitude",ylab="Latitude",...) {
  if (!is.null(subset)) {
    z=z[subset]
    x=x[subset]
    y=y[subset]
  }
  span = max(z) - min(z)
  stepsize = span/n.colors
  breaks = signif(min(z) + (1:n.colors)*stepsize,digits)
  z = 1+floor((z-min(z))/stepsize)
  my.greys = grey(((n.colors-1):0)/n.colors)
  colors = my.greys[z]
  plot(x,y,pch=pch,col=colors,bg=colors,
xlab=xlab,ylab=ylab,...)
  legend("topright",legend=breaks,fill=my.greys)
}
```

This plotting function is somewhat crude; a better one would let the user control the breaks, or perhaps set them in terms of quantiles rather than fractions of the range.

The size of the residuals is certainly not uniform over the state — positive residuals tend to disproportionately occur near the coasts, especially near the great conurbations (San Francisco, Los Angeles, San Diego). There are lots of negative residuals there too, though arguably central locations are a bit more negative than they ought to be. Looking at the absolute value of the residuals, however, shows that the biggest errors tend to be along the coasts and cities.

As for whether the residuals “should” be uniform, that depends. The standard calculations of significance levels and so forth assume homoskedastic (and Gaussian) errors, so the fact that residuals are not of uniform size over the state means you can’t trust any of those calculations. It also suggests that there is something systematically wrong with the model, since it’s doing much worse in some geographically-identifiable areas than others.

6. **Answer:** Rather than writing out commands over and over, I’ll write a function.
Figure 8: Map of the residuals. Reducing the point size keeps them from overlapping and clarifies the patterns. Notice that the darker colors (highest residuals) cluster around the coast, especially around the big cities.
plot.calif(resid, subset=(resid>0), cex=0.4, main="Positive-only residuals")

Figure 9: Positive residuals only. Fewer points so I can make each one larger.
plot.calif(-resid, subset=(resid<0), cex=0.4, main="Negative-only residuals")

Figure 10: Negative only residuals. (There should be a minus sign in front of the numbers in the legend, but this way bigger negative residuals show up as darker points.)
plot.calif(abs(resid), cex=0.2, main="Residual magnitude")

Figure 11: Absolute value of residuals.
calif.plot.residuals <- function(i,f=1.06*(nrow(calif)^(-0.2))) {
  x <- calif[,i+1]
  plot(x,resid,xlab=colnames(calif)[i+1],ylab="Residuals")
  lines(ksmooth(x,resid,bandwidth=sd(x)*f),col="red",lty=2)
}

This is less than ideal programming practice, because it presumes the objects calif and resid exist and have the right properties, and will fail obscurely if they don’t, but it’s better than typing everything out longhand. The f argument gives the bandwidth in terms of the standard deviation. The default is a “reference rule” which works well when everything is Gaussian, but playing around with it shows that it doesn’t matter too much.

There is a clear downward trend for the residuals against income; also a narrowing of the range of residuals. In fact, for all of the variables except house age, latitude and longitude, the range and variance of the residuals shrinks as we go to larger values of the feature. This means that the residuals are not independent of the features, as they should be if the linear model were correct.

7. The standard errors are

```r
> cbind(signif((summary(fit)$coef)[-1,2],3))
     [,1]     [,2]
MedianIncome  335.000  40900.00
MedianHouseAge 43.200   1240.00
TotalRooms      0.788    -6.64
TotalBedrooms   6.900    99.90
Population      1.080  1070.00
Households      7.510    127.00
Latitude        673.000  713.000
Longitude      673.000  713.000
```

The number of degrees of freedom is $20631 = 20640 - (8 + 1) = n - (p + 1)$.

So, if the standard formulas hold, we can get confidence intervals from the $t$ distribution. These are

```r
> signif( cbind((summary(fit)$coef)[-1,1] - qt(.975,20631) * (summary(fit)$coef)[-1,2],
               (summary(fit)$coef)[-1,1] + qt(.975,20631) * (summary(fit)$coef)[-1,2]) ,3)
     [,1]     [,2]
MedianIncome  39600.00  40900.00
MedianHouseAge 1070.00  1240.00
TotalRooms   -9.73     -6.64
TotalBedrooms 99.90   127.00
```
par(mfrow=c(4,2))
sapply(1:8,calif.plot.residuals)

Figure 12: Residuals plotted against the input features, decorated with red, dashed lowess smoothing lines.
Population -40.60 -36.40
Households 33.60 63.00
Latitude -43900.00 -41300.00
Longitude -44200.00 -41400.00

But the residuals are not Gaussian, and they are not independent of the input variables, so the distributional assumptions for the confidence interval formulas aren’t met. We can’t say whether this makes them too big, too small, skewed to one side, etc.

8. **Answer:** The coefficients:

```r
> signif(coefficients(logfit),3)
(Intercept) MedianIncome MedianHouseAge TotalRooms
  -1.18e+01  1.78e-01  3.26e-03  -3.19e-05
TotalBedrooms Population Households Latitude
  4.80e-04  -1.72e-04  2.49e-04  -2.80e-01
Longitude
  -2.76e-01
```

The MSE (and RMS error as well):

```r
> signif(summary(logfit)$sigma^2,3)
[1] 0.116
> signif(summary(logfit)$sigma,3)
[1] 0.34
```

Notice that this is on a log scale, so the claimed accuracy is to within a factor of \( \exp(0.34) = 1.4 \), i.e., \( \pm 40\% \). Finally, \( R^2 \):

```r
> signif(summary(logfit)$r.square,3)
[1] 0.643
```

One good reason to prefer the log-scale regression is that it always predicts a positive price, while the untransformed regression predicts negative median prices for 99 census tracts. Even in the least appealing parts of California, even today, people will not actually pay you to take houses off their hands, so that suggests something is seriously wrong with the untransformed model. (See Figure 13.)

A bad reason to prefer one regression over the other is to compare their \( R^2 \) or their MSEs; these are not comparable, because the response variables are different.

As you may have noticed, there are some weird features to the data. Median house prices never exceed $500,001, and there are 965 tracts with that house price. This is not because of an astonishing numerical coincidence, but because
Figure 13: Upper panel: actual median prices (horizontal axis) vs. those predicted by the untransformed regression (vertical axis). The dashed horizontal line is a zero price, the dashed diagonal the $x = y$ line. Lower panel: the same for the transformed regression; note that all predicted prices are positive.
of top-coding — prices higher than that were simply recorded as that max-
imum value. Similarly for the 49 tracts with the maximum income value and
the 1273 tracts with the maximum median house age. This sort of thing — to
say nothing of the sheer errors mentioned in footnote 1 — is entirely typical of
large real-world data sets.