Assessment Ratio Analysis using the "aratio" Package in R

Daniel P. McMillen

September 10, 2010

1 Introduction

The aratio package is designed to help perform analyses of the accuracy of property assessments. It includes commands that produce tables of the statistics commonly used in assessment ratio studies, including the mean, median, coefficient of dispersion, and price–related differential. It also includes more advanced commands that are designed to help with analyses of assessment regressivity. A nonparametric regression estimator, locally weighted regression, can be used to estimate nonlinear regressions of assessment ratios on sales prices. In addition, aratio includes two series–expansion approaches for estimating nonlinear regressions cubic splines functions and fourier expansions. The package also includes procedures designed to analyze the full distribution of assessment ratios and its relationship to sales price. These procedures include a nonparametric quantile estimator, implemented with the command qreglwr, and an estimator for conditional density estimation, condens. The commands are designed to produce a rich set of results with the default specifications.

This document illustrates the use of the aratio for a representative data set which is included in the package. The dupage99 data set has 2000 observations for two variables, the assessment value (av) and the sales price (price). The data set is a random draw of 2000 combinations of assessments and sales prices from DuPage County, IL in 1999. Sales took place in 1999; the assessments were in place in 1998. Statutory assessment rates in DuPage County were 0.33.

2 Data

The simplest way to read data into R is to work directly from data sets stored by standard statistical programs such as Stata, SPSS, and SAS. To do so, the package "foreign" must be installed from one of the cran mirrors (http://www.r-project.org/). For example, suppose the Stata data set mydata.dta is stored in a directly called c:\assessments. It can be read into R using library(foreign) combined with one of the specialized "read" commands:

```
> library(foreign)
> mydata  <- read.dta("/assessments/mydata.dta")   # Stata
```

The read command only has to be altered slightly to read data files from other common software packages:

```
> mydata <- read.dbf("/assessments/mydata.dbf") # dBASE
> mydata <- read.spss("/assessments/mydata.sav",to.data.frame=T) # SPSS
> mydata <- read.xport("/assessments/mydata.tpt") # SAS export file
> mydata <- read.systat("/assessments/mydata.syd") # SYSTAT
```

Note that R uses the assignment operator "<-" rather than an equal sign. The equal sign works for scalars but not for data frames, variables, or other multi-dimensional items. Also, note that R uses the forward slash "/" for file locations rather than the conventional backward slash "\".

The data set can also be read in to the workspace directly as an excel file that has been stored in a text format using either the "txt" or "csv" extensions:

```
> mydata <- read.table("/assessments/mydata.txt",header=T)
> mydata <- read.csv("/assessments/mydata.csv",header=T)
```

R is different from many programs in that the data are read into specific objects, in this case a data frame, rather than into the workspace. This means that commands like

```
> ratio <- av/price
> summary(ratio)
```

will not work. Instead, the commands must acknowledge that the variables are part of a particular data frame. The following command adds the variable "ratio" to the mydata data frame:

```
> mydata$ratio <- mydata$av/mydata $price
```

Alternatively, the data can be placed in the workspace using the "attach" command:

```
> attach(mydata)
> ratio <- av/price
```

It is important to note that the workspace and date frame versions of the data set are completely separate. A transformation such as price <- price/1000 will alter the version of price that is stored in the workspace without affecting the version stored in mydata. Similarly, mydata$price <- mydata$price/1000 alters the version stored in mydata without altering the version stored in the workspace.

It also is important to note that variable names are case sensitive. PRICE is not the same as price or Price.

3 Outlier Trimming

After reading the data into the program, the first step in any data analysis is to calculate some simple summary statistics. R's summary command produces basic summary statistics for the aratio data frame, dupage99:

```
> library(aratio)
```

```
locfit 1.5-6          2010-01-20
```

```
> data(dupage99)
> dupage99$ratio <- dupage99$av/dupage99$price
```

The "library" command loads the aratio data set, which first should be installed from one of the cran mirrors. The "data" command loads the dupage99 data frame that is included in the aratio package. The "summary" command then produces the following results:

```
> summary(dupage99)
```

```
       av              price              ratio       
 Min.   :  1380   Min.   :   4500   Min.   :0.1533  
 1st Qu.: 39075   1st Qu.: 129000   1st Qu.:0.2785  
 Median : 49905   Median : 166600   Median :0.2993  
 Mean   : 58099   Mean   : 198991   Mean   :0.2986  
 3rd Qu.: 67870   3rd Qu.: 234000   3rd Qu.:0.3188  
 Max.   :426760   Max.   :1900000   Max.   :0.6126  
```

Summary statistics can also be obtained for each variable individually as follows:

```
> summary(dupage99$av)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1380   39080   49900   58100   67870  426800 
```

```
> summary(dupage99$price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   4500  129000  166600  199000  234000 1900000 
```

```
> summary(dupage99$ratio)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.1533  0.2785  0.2993  0.2986  0.3188  0.6126 
```

The function nptrim_obs trims outliers from the data set using a nonparametric trimming procedure recommended by the IAAO. Letting q25 represent the 25th percentile assessment ratio and letting q75 represent the 75th percentile, the lower bound for trimming is lo = q25 k*(q75-q25) and the upper bound is hi = q75 + k*(q75-q25). The parameter k can be set by the user. Typical values for k are 1.5 and 3.0; the default is 3.0. The output of the function is a dummy variable whose values equal one if the observation is identified as an outlier. If print = T, the function also produces useful summary information:

```
> dropobs <- nptrim_obs(dupage99$ratio, k = 3)
```

```
Total number of observations                      2000.0000000
Number of missing observations                       0.0000000
Number of non-missing observations                2000.0000000
Number of non-missing observatons trimmed            8.0000000
Total number of observations dropped                 8.0000000
Number of non-missing observations after trimming 1992.0000000
minimum                                              0.1533382
25th percentile                                      0.2785128
75th percentile                                      0.3188254
maximum                                              0.6126000
lower bound for trimming                             0.1575750
upper bound for trimming                             0.4397633
```

The results show that 8 of the original observations were identified as outliers when k=3. The lower bound for trimming, 0.1576, is slightly higher than the minimum assessment ratio value that appears in this data set. The upper bound of 0.4397 is well below the maximum value of 0.6126.

The following command then produces a new version of dupage99 that is trimmed of the 8 outlier observations:

```
> dupage99 <- dupage99[dropobs == 0, ]
```

This command could also be written:

```
> dupage99  <- dupage99[dropobs==F,]
```

because R interprets 0 and F (or 1 and T) as equivalent in logical expressions.

Some explanation is needed for these expressions. R treats data frames as though they are matrices. Thus, dupage99 is actually a 2000x3 matrix once the ratio variable is added to it. Individual elements for the matrix can be identified by the row and column number, e.g., for the matrix amat, amat[1,3] is the entry in the first row and third column. Empty entries can be used to return the full set of rows and columns, e.g., amat[1,] and acol3[,3] return the first row and third column of amat, respective. Logical operators can be used to return observations that meet certain conditions. Thus, dupage99[dropobs==0,] returns all columns of dupage99 (i.e., all variables) and all rows (i.e., observations) for which dropobs==0.

Note that this command works even though dropobs is not one of the variables in dupage99. The variable dropobs is stored in the workspace rather than in dupage99. It could have been added to dupage99 by writing dupage99$dropobs <- dropobs. The data frame dupage99 then could be trimmed of outliers by issuing the command dupage99 <- dupage99[dupage99$dropbs==0,]. This version of dupage has 4 variables –av, price, ratio, and dropobs –rather than only three.

The document "An Introduction to R" provides much useful information on data handling and other topics. It is available in the "manuals" section of the help facility of the R program.

4 Traditional Assessment Ratio Analysis

Once the data set is trimmed of outliers it is ready for traditional assessment ratio analysis. The assessment ratio for observation i is defined as $ratio_i = \frac{A_i}{P_i}$, where A represents the assessed value and P represents price. The corresponding variables in the dupage99 data frame are av and price.

The main statistics used to evaluate assessment performance are the following:

1. Mean
2. Median
3. Value-weighted mean: $\sum_i \frac{(P_i * ratio_i)}{\sum_i (P_i)} = \frac{mean(A)}{mean(P)}$
4. Coefficient of dispersion: $100 * \sum_i \frac{|ratio_i - median(ratio)|}{n * median(ratio)}$
5. Value-weighed coefficient of dispersion: $100 * \sum_i \frac{P_i * |ratio_i - median(ratio)|}{median(ratio) * \sum_i P_i}$
6. Price-related differential: mean/(value weighted mean)

These statistics can be calculated in aratio using the ratio_stats command. The following commands produce a simple table of statistics:

```
> rvect <- ratio_stats(dupage99$av, dupage99$price)

     mean    median  wgt mean       cod   wgt cod       prd
0.2979827 0.2992920 0.2915292 8.8375620 9.7734204 1.0221365
```

The output for this application of the ratio_stats command is stored in the vector rvect. The full vector can be written by simply typing rvect; individual entries are rvect[1]=0.297983, rvect[2]=0.299292, etc.

Although it is easy to calculate standard errors and confidence intervals for the mean, simple analytical results are not available for more complicated statistics such as the cod or prd. A bootstrap procedure can be used to construct standard error estimates and confidence intervals in this situation. The ratio_stats command uses the package *boot* to calculate standard errors for all six statistics. The procedure involves drawing with replacement *nboot* new samples of size n from the original set of (av, price) pairs and re-calculating the statistics for each of the nboot samples. The standard error is then simply the standard deviation of the *nboot* values of the statistic in question, and the bootstrap 95% confidence interval is the original sample statistic plus/minus 1.96 times the standard error. These values appear under the headings "normal-lo" and "normal-hi" in the ratio_stats command when the option nboot is specified. This confidence interval is most appropriate if the statistic follows a normal distribution. If the statistic does not follow a symmetric distribution, the "percentile method" will produce a more accurate confidence interval. Let B denote the vector containing the nboot re-calculations of the test statistics; the percentile method 95% lower bound is simply quantile(b, .025) and the upper bound is quantile(b, .975), i.e., the 2.5 percentile and 97.5 percentile of the bootstrap values of the statistics. These values are labeled "percentile-lo" and "percentile-hi" by ratio_stats.

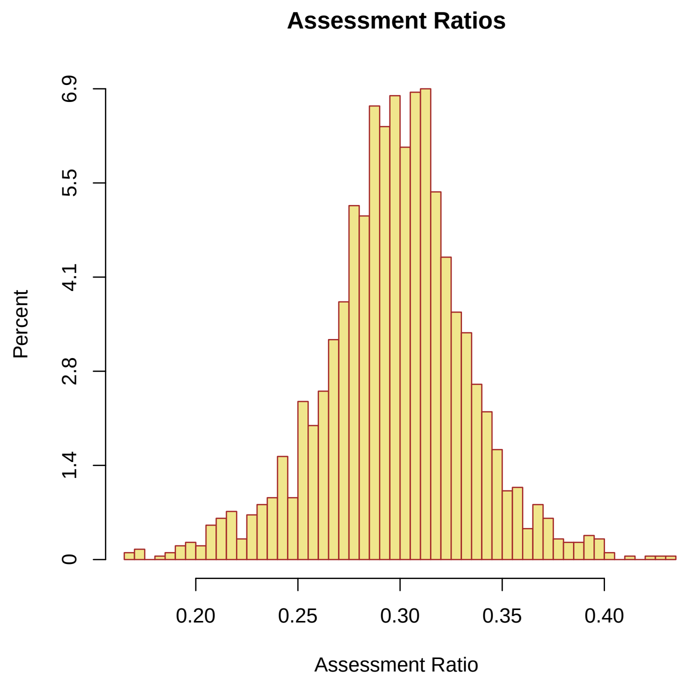
Since the boostrap procedure requires draws from random number generators, the results will vary slightly each time the program is run even though the data set does not change. The "seed" of the random number generator can be set with the "set.seed" command to ensure that the same set of random numbers is drawn each time the program is run. The argument in the set.seed command can be any odd-valued integer. The option nboot controls the number of bootstrap repetitions. For most applications, a value of 100 is probably sufficient. There will be less variation in the values for the standard errors and confidence intervals as nboot increases, although there is unlikely to be any reason to set nboot higher than 1000 or so. The following series of commands produces a full set of results for ratio_stats:

```
> set.seed(5849301)
> rvect <- ratio_stats(dupage99$av, dupage99$price, nboot = 1000)

          observed      Std Err Normal-lo  Normal-hi Percentile-lo
mean     0.2979827 0.0007735260 0.2964665  0.2994988     0.2965488
median   0.2992920 0.0006972902 0.2979254  0.3006587     0.2978864
wgt mean 0.2915292 0.0011657453 0.2892443  0.2938141     0.2892219
cod      8.8375620 0.1713530128 8.5017101  9.1734139     8.5187898
wgt cod  9.7734204 0.2821079250 9.2204889 10.3263520     9.2208660
prd      1.0221365 0.0027577930 1.0167313  1.0275418     1.0170238
         Percentile-hi
mean         0.2995592
median       0.3007101
wgt mean     0.2938701
cod          9.1820094
wgt cod     10.3469003
prd          1.0280090
```

The results indicate reasonably good assessment performance in DuPage County in 1999. Mean and median assessments are fairly close to the statutory rate of .33, although they both differ significant from .33 at a 95% confidence level. The COD of 8.84 is well below and statistically different from the IAAO upper limit guideline of 15, indicating that assessment ratios do not exhibit an excessive degree of variability. The weighted mean and weighted COD are

Figure 1: Demonstration of “agraph”



close to the unweighted values, suggesting that there is no systematic difference across sales prices in mean assessment ratios or their degree of variability. The PRD is close to 1 and well below the IAAO recommended upper limit of 1.06, indicating that there is only a small tendency toward lower assessments ratios for higher-priced properties.

For this example, the results of the ratio_stats command are stored in the 6x6 matrix rvect. Individual results are accessible by specifying the appropriate entry of rvect. For example, rvect[2,3]=0.297925 and rvect[6,4]=1.027542.

The command agraph provides a convenient interface to some relevant graphing facilities in R. Simply typing

```
> agraph(dupage99$av, dupage99$price)
```

produces the histogram in Figure 1.

The agraph command includes several options to control the appearance of the graph. For example, the title can be changed using the “title” option, the width of the bars (and therefore the number of bins) can be changed using the “width” option, and the y-axis can be expressed in frequencies rather than percentages by specifying freq=T. In addition, a normal distribution can be superimposed on the graph using the normdens=T option, and a nonparametric kernel density function is included by specifying kdens=T. One vertical line can be added to the graph at an arbitrary value using the "statute" option. The "statute" option is meant to show the statutory assessment rate; the statutory rate in DuPage County is .33 so it may be desirable to add the option statute=.33. Vertical bars can also be added for two other points using the "cfint" option. The "cfint" option is meant for showing confidence intervals on the graph. Two values are required for cfint, e.g, cfint=c(.25,.35) or some other combination of values. (The term c(.25, .35) is an array with two elements; see the "Introduction to R" documentation file for a discussion of how arrays are handled in R.) The option "legloc" adds a legend for these options. The location of the legend can be varied using standard R options, including "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", and "center."

The following example shows how agraph can be combined with the ratio_stats command to produce a histogram with normal density and kernel density functions, the statutory .33 rate, and the 95% confidence interval for the median all shown on the diagram in Figure 2.

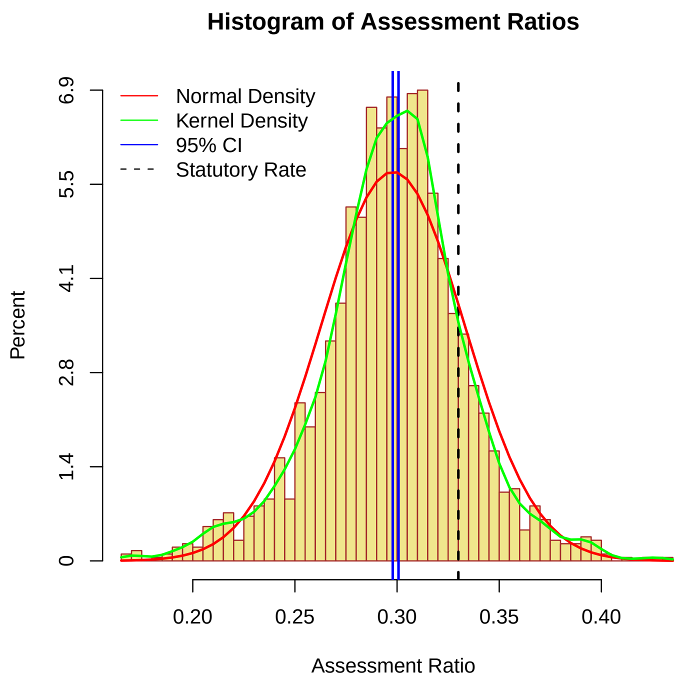
```
> set.seed(5849301)
> rstat <- ratio_stats(dupage99$av, dupage99$price, nboot = 1000)

          observed      Std Err Normal-lo  Normal-hi Percentile-lo
mean     0.2979827 0.0007735260 0.2964665  0.2994988     0.2965488
median   0.2992920 0.0006972902 0.2979254  0.3006587     0.2978864
wgt mean 0.2915292 0.0011657453 0.2892443  0.2938141     0.2892219
cod      8.8375620 0.1713530128 8.5017101  9.1734139     8.5187898
wgt cod  9.7734204 0.2821079250 9.2204889 10.3263520     9.2208660
prd      1.0221365 0.0027577930 1.0167313  1.0275418     1.0170238
         Percentile-hi
mean         0.2995592
median       0.3007101
wgt mean     0.2938701
cod          9.1820094
wgt cod     10.3469003
prd          1.0280090

> lo = rstat[2, 5]
> hi = rstat[2, 6]
> agraph(dupage99$av, dupage99$price, normdens = T, kdens = T,
+     legloc = "topleft", cfint = c(lo, hi), title = "Histogram of Assessment Ratios",
+     statute = 0.33)
```

Figure 2: “agraph” with confidence intervals, statutory rate, and median

```
          observed      Std Err Normal-lo  Normal-hi Percentile-lo
mean     0.2979827 0.0007735260 0.2964665  0.2994988     0.2965488
median   0.2992920 0.0006972902 0.2979254  0.3006587     0.2978864
wgt mean 0.2915292 0.0011657453 0.2892443  0.2938141     0.2892219
cod      8.8375620 0.1713530128 8.5017101  9.1734139     8.5187898
wgt cod  9.7734204 0.2821079250 9.2204889 10.3263520     9.2208660
prd      1.0221365 0.0027577930 1.0167313  1.0275418     1.0170238
         Percentile-hi
mean         0.2995592
median       0.3007101
wgt mean     0.2938701
cod          9.1820094
wgt cod     10.3469003
prd          1.0280090
```



5 Regression Analysis of Assessment Regressivity

The PRD of 1.022 suggests that assessments in DuPage County were somewhat regressive in 1999, meaning that higher priced properties were assessed at somewhat lower rates on average than lower priced properties. Regression procedures can also be used to analyze this relationship. The most direct way to determine whether assessment ratios decline with sales price is to estimate a regression of the form ; a negative value for ndicates that assessment ratios decline with sales price.

The following commands produce a scatterplot of the raw data for the DuPage County data set:

```
> dupage99$price <- dupage99$price/1000
> plot(dupage99$price, dupage99$ratio, xlab = "Sales Price (1000s)",
+     ylab = "Assessment Ratio")
```

The relatively small number of homes with very high sales prices suggests that it may be preferable to limit the sample to lower-priced homes for which a linear relationship between sales price and the assessment ratio may provide a good fit. Thus, we might limit the sample to observations with sales prices below the 99th percentile. In general, it is preferable to trim observations symmetrically at both the lower and upper ends of the distributions. The following commands limit the sample to sales prices between the 1st and 99th percentiles for price, plot the results, and add the fitted regression line:

```
> keepobs <- dupage99$price > quantile(dupage99$price, 0.01) &
+     dupage99$price < quantile(dupage99$price, 0.99)
> dupage99 <- dupage99[keepobs == T, ]
> o <- order(dupage99$price)
> dupage99 <- dupage99[o, ]
> fit <- lm(ratio ~ price, data = dupage99)
```

Together, the commands o <- order(dupage99$price) and dupage99 <- dupage99[o,] sort the data by sales price. Sorting makes it possible to add lines to the scatterplot; these two operations are done with the "plot" and "lines" commands. The command "lm" (which stands for "linear model") is the R command for a regression. The results are stored in the object "fit". The regression output is not automatically printed by R; the "summary" command prints the following regression results:

```
> summary(fit)

Call:
lm(formula = ratio ~ price, data = dupage99)

Residuals:
       Min         1Q     Median         3Q        Max
-0.1236397 -0.0181863 -0.0003016  0.0184428  0.1278409

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.184e-01  1.695e-03  187.81   <2e-16 ***
price       -1.042e-04  7.890e-06  -13.20   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0331 on 1950 degrees of freedom
Multiple R-squared: 0.08205,        Adjusted R-squared: 0.08158
F-statistic: 174.3 on 1 and 1950 DF,  p-value: < 2.2e-16
```

The scatterplot of the raw data and the estimated regression looks as follows:

```
> plot(dupage99$price, dupage99$ratio, xlab = "Sales Price (1000s)",
+     ylab = "Assessment Ratio")
> lines(dupage99$price, fitted(fit), col = "red")
```

Overall, the results indicate some regressivity: each additional $1,000 in sales price is associated with a decline of 0.0001042 in the assessment ratio, or a decline of 0.1042 percentage points for each $10,000 increase in sales price.

6 Non-linear Regression

There is no particular reason why we should expect the relationship between sales prices and assessment ratios to be linear. The aratio package includes three commands to simplify the estimation of nonlinear relationships "cubespline" for smooth cubic splines, "fourier" for fourier expansions, and "lwr1" for a locally weighted regression.

The cubic spline function is written as follows:

$$y = cons + \lambda_1(x - x_0) + \lambda_2(x - x_0)^2 + \lambda_3(x - x_0)^3 + \sum_{k=1}^{K} \gamma_k(x - x_k)^3 D_k + u$$

where y is the dependent variable, x is the single explanatory variable, x0 = min(x). The function divides the range of x into knots+1 equal intervals. The knots are x1xK and the Dk are dummy variables that equal one if . Estimation can be carried out for a fixed value of K or for a range of K. In the latter case, the function indicates the value of K that produces the lowest value of one of the following criteria: the AIC (or "Akaike Information Criterion"), the Schwarz information criterion, or the gcv (or "generalized cross-validation"), each of which is a commonly used information criterion that is used to choose the optimal number of explanatory variable in a regression.

Estimation can be carried out for a fixed number of knots by specifying the "knots" options. For example, a cubic spline with two knots is written:

```
> fit <- cubespline(ratio ~ price, knots = 2, data = dupage99)

Reminder:  first explanatory variable is used for spline

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.1195049 -0.0180819 -0.0002521  0.0186051  0.1274393 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
Intercept  3.071e-01  1.243e-02  24.714   <2e-16 ***
price      2.171e-04  2.152e-04   1.009   0.3130    
square    -2.251e-06  1.151e-06  -1.956   0.0506 .  
cube       4.291e-09  1.903e-09   2.255   0.0242 *  
kvar1     -8.097e-09  3.243e-09  -2.497   0.0126 *  
kvar2      1.114e-08  7.374e-09   1.511   0.1311    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.033 on 1946 degrees of freedom
Multiple R-squared: 0.08943,	Adjusted R-squared: 0.08709 
F-statistic: 38.22 on 5 and 1946 DF,  p-value: < 2.2e-16
```

If the “knots” option is not specified, cubespline fits a cubic spline with a single knot. It is also possible to use cubespline to search for the optimal number of knots, i.e., the number that minimizes the value of the criterion function. The search is conducted by specifying the mink and maxk options rather than the knots options, e.g.,

```
> fit <- cubespline(ratio ~ price, mink = 1, maxk = 5, data = dupage99)

Reminder:  first explanatory variable is used for spline
Information Criterion, Linear:     0.001093946 
Information Criterion, k = 1 : 0.001094147 
Information Criterion, k = 2 : 0.001092562 
Information Criterion, k = 3 : 0.001093360 
Information Criterion, k = 4 : 0.001094006 
Information Criterion, k = 5 : 0.001094454 
Information Criterion minimizing k =  2 

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.1195049 -0.0180819 -0.0002521  0.0186051  0.1274393 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
Intercept  3.071e-01  1.243e-02  24.714   <2e-16 ***
price      2.171e-04  2.152e-04   1.009   0.3130    
square    -2.251e-06  1.151e-06  -1.956   0.0506 .  
cube       4.291e-09  1.903e-09   2.255   0.0242 *  
kvar1     -8.097e-09  3.243e-09  -2.497   0.0126 *  
kvar2      1.114e-08  7.374e-09   1.511   0.1311    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.033 on 1946 degrees of freedom
Multiple R-squared: 0.08943,        Adjusted R-squared: 0.08709 
F-statistic: 38.22 on 5 and 1946 DF,  p-value: < 2.2e-16
```

As it turns out, the optimal number of knots is K = 2 according to the default criterion function, the gcv.

The use of the fourier command is similar. Again the general form of the model is y = f(x), using a fourier expansion for x, but the fourier model uses sine and cosine terms to expand the model. The variable x is first transformed to form z <- 2*pi*(x-min(x))/(max(x)-min(x)). The fourier model is $y = \alpha_1 z + \alpha_2 z^2 + \sum_{q=1}^{Q} (\lambda_q sin(qz) + \delta_q cos(qz))$. Estimation can be carried out for a fixed value of Q or for a range of Q. These options are controlled with the options “q” or “minq” combined with “maxq”. For example,

```
> fit <- fourier(dupage99$ratio ~ dupage99$price, minq = 1, maxq = 5)
```

```
Reminder:  first explanatory variable is used for fourier expansion 
Information Criterion, Linear:     0.001094011 
Information Criterion, q = 1 : 0.001092920 
Information Criterion, q = 2 : 0.00109387 
Information Criterion, q = 3 : 0.001095529 
Information Criterion, q = 4 : 0.001096658 
Information Criterion, q = 5 : 0.001095338 
Information Criterion minimizing q =  1 

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.1203103 -0.0181056 -0.0003924  0.0185667  0.1279510 

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
Intercept  0.3078984  0.0083341  36.944   <2e-16 ***
z          0.0002245  0.0094015   0.024   0.9809
square    -0.0018828  0.0016826  -1.119   0.2633
sin(1z)   -0.0092653  0.0038085  -2.433   0.0151 *
cos(1z)    0.0076087  0.0056814   1.339   0.1807
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03302 on 1947 degrees of freedom
Multiple R-squared: 0.08819,       Adjusted R-squared: 0.08632
F-statistic: 47.08 on 4 and 1947 DF,  p-value: < 2.2e-16
```

The initial series of values for "Information Criterion" indicates the optimal expansion length is Q = 1, which implies a model with z, z^2, $sin(z)$, and $cos(z)$ as the explanatory variables. The same model could have been obtained using the following command:

```
> fit <- fourier(ratio ~ price, q = 1, data = dupage99)

Reminder:  first explanatory variable is used for fourier expansion

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max
-0.1203103 -0.0181056 -0.0003924  0.0185667  0.1279510

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
Intercept  0.3078984  0.0083341  36.944   <2e-16 ***
z          0.0002245  0.0094015   0.024   0.9809
square    -0.0018828  0.0016826  -1.119   0.2633
sin(1z)   -0.0092653  0.0038085  -2.433   0.0151 *
cos(1z)    0.0076087  0.0056814   1.339   0.1807
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03302 on 1947 degrees of freedom
Multiple R-squared: 0.08819,       Adjusted R-squared: 0.08632
F-statistic: 47.08 on 4 and 1947 DF,  p-value: < 2.2e-16
```

or, since q=1 is the default value of q, by:

```
> fit <- fourier(ratio ~ price, data = dupage99)
```

```
Reminder:  first explanatory variable is used for fourier expansion

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max
-0.1203103 -0.0181056 -0.0003924  0.0185667  0.1279510

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
Intercept  0.3078984  0.0083341  36.944   <2e-16 ***
z          0.0002245  0.0094015   0.024   0.9809
square    -0.0018828  0.0016826  -1.119   0.2633
sin(1z)   -0.0092653  0.0038085  -2.433   0.0151 *
cos(1z)    0.0076087  0.0056814   1.339   0.1807
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03302 on 1947 degrees of freedom
Multiple R-squared: 0.08819,        Adjusted R-squared: 0.08632
F-statistic: 47.08 on 4 and 1947 DF,  p-value: < 2.2e-16
```

Additional explanatory variables can be added to the cubespline and fourier models by altering the initial form, e.g., fit <- cubespline(y x+z). The spline or fourier expansion is then applied to the first variable, x, and the remaining variable or variables in z are added as written.

The predicted values from the cubespline and fourier estimates are stored in a variable called "yhat". Since the models are simple linear regressions, the mean value of yhat will be equal to the mean of the dependent variable. In addition, the nonlinear part of the model is stored as "splinehat" for the cubic spline and "fourierhat" for the fourier expansion. If the model has only a single explanatory variable x so that y = f(x), there is no difference between the cubic spline "yhat" and "splinehat" or the fourier "yhat" and "fourierhat". However, the terms are different when additional explanatory variables, z, are added to the regressions. The "yhat" predictions include the effect of z; the "splinehat" and "fourierhat" values include the effects of x alone. Both "splinehat" and "fourierhat" are rescaled to have the same mean as the dependent variable. Thus, mean(y) = mean(yhat) = mean(splinehat) = mean(fourierhat).

The predictions are returned as output from the cubespline and fourier commands. For example, if the cubepline model is estimated using the command fit <- cubespline(yx); yhat and splinehat are accessible as fit$yhat and fit$splinehat.

The following series of commands compares the predictions for a standard linear regression model, the cubic spline with knots = 2, and the fourier model with q = 1, and produces a graph of the three:

```
> fit1 <- lm(ratio ~ price, data = dupage99)
> yhat1 <- fitted(fit1)
> fit2 <- cubespline(ratio ~ price, knots = 2, data = dupage99)

Reminder:  first explanatory variable is used for spline

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.1195049 -0.0180819 -0.0002521  0.0186051  0.1274393 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
Intercept  3.071e-01  1.243e-02  24.714   <2e-16 ***
price      2.171e-04  2.152e-04   1.009   0.3130    
square    -2.251e-06  1.151e-06  -1.956   0.0506 .  
cube       4.291e-09  1.903e-09   2.255   0.0242 *  
kvar1     -8.097e-09  3.243e-09  -2.497   0.0126 *  
kvar2      1.114e-08  7.374e-09   1.511   0.1311    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.033 on 1946 degrees of freedom
Multiple R-squared: 0.08943,        Adjusted R-squared: 0.08709 
F-statistic: 38.22 on 5 and 1946 DF,  p-value: < 2.2e-16 

> yhat2 <- fit2$splinehat
> fit3 <- fourier(ratio ~ price, q = 1, data = dupage99)

Reminder:  first explanatory variable is used for fourier expansion

Call:
lm(formula = newform)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.1203103 -0.0181056 -0.0003924  0.0185667  0.1279510 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
Intercept  0.3078984  0.0083341  36.944   <2e-16 ***
z          0.0002245  0.0094015   0.024   0.9809    
square    -0.0018828  0.0016826  -1.119   0.2633    
sin(1z)   -0.0092653  0.0038085  -2.433   0.0151 *  
cos(1z)    0.0076087  0.0056814   1.339   0.1807    
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.03302 on 1947 degrees of freedom
Multiple R-squared: 0.08819,       Adjusted R-squared: 0.08632
F-statistic: 47.08 on 4 and 1947 DF,  p-value: < 2.2e-16
```

```
> yhat3 <- fit3$fourierhat
> ymin = min(yhat1, yhat2, yhat3)
> ymax = max(yhat1, yhat2, yhat3)
> plot(dupage99$price, yhat1, type = "l", xlab = "Sales Price (1000s)",
+     ylab = "Assessment Ratio", ylim = c(ymin, ymax))
> lines(dupage99$price, yhat2, col = "blue")
> lines(dupage99$price, yhat3, col = "red")
> legend("topright", c("OLS", "Spline", "Fourier"), col = c("black",
+     "blue", "red"), lwd = 1)
```

The nonlinear functions suggest a flat region in the 250,000 400,000 range, followed by a sharp decline thereafter. The spline function appears to suggest an unrealistic upward slope at the very highest values; the endpoints of polynomial functions are often heavily influenced by a small number of data points.

Various nonparametric procedures might be considered as an alternative to the cubic spline for fourier models. The lwr1 command in the aratio package uses locally weighted regression (LWR) to approximate a function of the form y = f(x) + u. Details of the procedure can be found in the help facility for the lwr1 command. LWR is implemented by fitting a series of weighted least squares regressions to various target points for x, with more weight place on observations that are closer to the target point, x0. The relative weight given to close and distant observations is controlled by the "bandwidth" and "window" options. The window option is preferable for most applications. It specifies the proportion of the observations that are included in the regression for each target point. The default window size of .25 means that the n*.25 observations whose values of x are closest to x0 are given positive weight in the regression, and among this 25% of the observations more weight is placed on the observation whose values of x are closer to x0. The predictions from the LWR regressions are stored as "yhat".

Although the lwr1 command offers many options for windows sizes and weighting schemes, the default version of the command is very simple. The following commands add an LWR fit with a 25% window size to the previous diagram:

```
> fit4 <- lwr1(ratio ~ price, data = dupage99)
> yhat4 <- fit4$yhat
> ymin = min(yhat1, yhat2, yhat3, yhat4)
> ymax = max(yhat1, yhat2, yhat3, yhat4)
> plot(dupage99$price, yhat1, type = "l", xlab = "Sales Price (1000s)",
+     ylab = "Assessment Ratio", ylim = c(ymin, ymax))
```

```
> lines(dupage99$price, yhat2, col = "blue")
> lines(dupage99$price, yhat3, col = "red")
> lines(dupage99$price, yhat4, col = "green")
> legend("topright", c("OLS", "Spline", "Fourier", "LWR"), col = c("black",
+     "blue", "red", "green"), lwd = 1)
```

The lwr1 command can also be used to choose a value of the window size or bandwidth that minimizes an information criterion function. As with the cubespline or fourier function, the default is the generalized cross-validation criterion. The option is implemented by specifying an array of possible values for "window" (or "bandwidth"). Possibilities might be c(.10, .15, .20, .25, .30) or seq(from=.10, to=.90, by=.10). See the "Introduction to R" documentation file for a discussion of vectors in R. The following results suggest that the default window size of 25% is too small for the dupage99 data set:

```
> fit5 <- lwr1(ratio ~ price, data = dupage99, window = seq(0.1,
+     0.9, 0.1))
```

```
[1] 0.100000000 0.001099664
[1] 0.200000000 0.001094039
[1] 0.300000000 0.001092076
[1] 0.400000000 0.001091576
[1] 0.500000000 0.001092178
[1] 0.600000000 0.001092309
[1] 0.700000000 0.001092366
[1] 0.800000000 0.001092265
[1] 0.900000000 0.001091868

 h =  0.4 Function Value =  0.001091576
```

The following diagram shows the estimated LWR functions with windows sizes of 25% and 40%:

```
> yhat4 <- fit4$yhat
> yhat5 <- fit5$yhat
> plot(dupage99$price, yhat4, xlab = "Sales Price (1000s)", ylab = "Assessment Ratio",
+     type = "l")
> lines(dupage99$price, yhat5, col = "red")
> legend("topright", c("25%", "40%"), col = c("black", "red"),
+     lwd = 1)
```

Though the functions are quite similar, the smaller window size produces somewhat noisier estimates at low sales prices. All of the nonlinear estimates suggest that assessment ratios decline more quickly with sales price for prices less than about $200,00 after which the rate of decline moderates somewhat.

Regressions are not necessarily the best way to approach modeling the potential regressivity of assessments. As an example, consider the following set of simulated data:

```
> n = 1000
> set.seed(277)
> price <- 1 + round(9 * runif(n), digits = 0)
> o <- order(price)
> price <- price[o]
> a = 0.12
> b = 0.12
> ratiohi <- runif(n, 0.33, (0.33 + a) - (a - 0.02) * price/10)
> ratiolo <- runif(n, (0.33 - b) + (b - 0.02) * price/10, 0.33)
> p = b/(a + b)
> ratio <- ifelse(runif(n) < p, ratiohi, ratiolo)
> plot(price, ratio, xlab = "Sales Price ($100,000)", ylab = "Assessment Ratio")
> fit <- lm(ratio ~ price)
> lines(price, fitted(fit), col = "red")
```

To simplify the picture, sales prices are randomly drawn from the set of integers ranging from 1 to 10. For each sales price, the assessment ratios are randomly drawn from a uniform distribution centered on the statutory assessment ratio of 0.33. The variance is much higher at lower sales prices. This may be a reasonable representation of a case where assessors are trying to estimated values correctly while low-priced properties are simply more variable and hence harder to assess accurately.

The red line is the set of predictions from a regression of the assessment ratios on sales prices. The results are:

```
> summary(fit)

Call:
lm(formula = ratio ~ price)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.100501 -0.026345 -0.002235  0.025680  0.105986 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.3324290  0.0029168 113.972   <2e-16 ***
price       -0.0003869  0.0004860  -0.796    0.426    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.04041 on 998 degrees of freedom
Multiple R-squared: 0.0006346,	Adjusted R-squared: -0.0003668 
F-statistic: 0.6337 on 1 and 998 DF,  p-value: 0.4262
```

These results suggest that assessments are neither progressive nor regressive. Nonetheless, it is clear from the scatterplot that owners of low-priced homes are much more likely to be both the beneficiaries of quite low assessment rates and the victims of very high rates.

Quantile regression is a good alternative to the standard linear regression model in this situation. Roughly speaking, a standard regression fit is the line connecting the average values of the assessment ratio for each sales price the expected value of the assessment ratio when price =1, 2, 3, and so on. Since the mean price is .33 in each case, the regression line is horizontal.

Quantile regression generalizes this approach by analyzing other parts of the distribution. This type of regression computes conditional "quantiles," commonly referred to as "percentiles." For example, suppose we wanted to know how assessment ratios respond to price when the ratio is in the top part of the distribution for each price. We might focus on the 90th percentile of the assessment ratio distribution what is the 90th percentile ratio when price = 1, when price = 2, and so on? The following series of commands calculates the 10th and 90th percentiles at each price:

```
> p <- seq(1:10)
> q10vect <- p
> q90vect <- p
> q10 <- array(0, dim = n)
> q90 <- array(0, dim = n)
> for (i in seq(1:10)) {
+     samp <- price == p[i]
+     q10[samp] <- quantile(ratio[samp], 0.1)
+     q90[samp] <- quantile(ratio[samp], 0.9)
+ }
> plot(price, ratio, xlab = "Sales Price ($100,000)", ylab = "Assessment Ratio")
> lines(price, q10, col = "red")
> lines(price, q90, col = "red")
```

The red lines show that the 10th and 90th percentiles of the assessment ratio distributions are honing in on .33 as sales price increases. In other words, assessments are becoming less variable as sales price increases.

Note that there is nothing in the above procedure that guarantees that the red lines will be straight. The quantile lines could appear quite irregular, particularly if the data set included many different sales prices rather than only ten. Quantile regression does roughly the same thing as this procedure but it estimates a line for each target quantile across all actual values of the sales prices. Straight lines can be fit easily using the quantreg package. For example, the following commands show the .10 and .90 quantile lines (tau = .10 and tau = .90) for the simulated data set:

```
> library(quantreg)
> fit10 <- rq(ratio ~ price, tau = 0.1)
> yhat10 <- fitted(fit10)
> fit90 <- rq(ratio ~ price, tau = 0.9)
> yhat90 <- fitted(fit90)
> lines(price, yhat10, col = "blue")
> lines(price, yhat90, col = "blue")
```

The blue lines are the predictions from the quantile regressions. In the case of this simulated data set, the straight-line quantile fits are clearly very close to the values that would be found by calculating the 10th and 90th percentiles of the assessment ratios at each sales price. The results would clearly look much different if the sales could have many values rather than 10, e.g., 1.125, 1.487, 8.265, etc. The quantreg package's "rq" command will compute quantile lines for any target "tau", e.g., specifying tau=.25 will fit a line for the quantile .25 and tau=.50 will fit a line for quantile .50, the median.

It also should be borne in mind that there is no reason why the quantile lines must be mirror images above and below the mean or median. For example, the following simulated data set has much higher variability among high assessment ratios than for low ratios:

```
> set.seed(987)
> a = 0.12
> b = 0.03
> ratiohi <- runif(n, 0.33, (0.33 + a) - (a - 0.02) * price/10)
> ratiolo <- runif(n, (0.33 - b) + (b - 0.02) * price/10, 0.33)
> p = b/(a + b)
> ratio <- ifelse(runif(n) < p, ratiohi, ratiolo)
> plot(price, ratio, xlab = "Sales Price ($100,000)", ylab = "Assessment Ratio")
> fit10 <- rq(ratio ~ price, tau = 0.1)
> fit90 <- rq(ratio ~ price, tau = 0.9)
> lines(price, fitted(fit10), col = "red")
> lines(price, fitted(fit90), col = "red")
```

Finally, consider the case that matches a textbook regression model in which the errors are normally distributed with constant variance:

```
> set.seed(48597)
> n = 1000
> price <- 1 + round(9 * runif(n), digits = 0)
> ratio <- 0.33 + 0 * price + rnorm(n, 0, 0.05)
> plot(price, ratio, xlab = "Sales Price ($100,000)", ylab = "Assessment Ratio")
> fit10 <- rq(ratio ~ price, tau = 0.1)
> fit90 <- rq(ratio ~ price, tau = 0.9)
> lines(price, fitted(fit10), col = "red")
> lines(price, fitted(fit90), col = "red")
```

In this classic case, the quantile lines are close to parallel, i.e., there is not a clear tendency toward increasing or decreasing variability in assessment ratios as sales price increases.

The aratio package supplements the package quantreg by providing a convenient way to calculate nonlinear quantile lines. The qreglwr command uses methods analogous to those used in standard locally weighted regression to estimate nonlinear quantile models. As is the case with the lwr1 command, qreglwr fits quantile regressions at a set of target points, with more weight placed on observations with prices that are close to the target values. The following set of commands illustrates the simplest use of qreglwr:

```
> fit <- qreglwr(ratio ~ price, graph.yhat = T)
```

By default, nonlinear quantile lines are estimated for five target quantiles: .10, .25, .50, .75, and .90. The lines in the diagram correspond to these quantiles; the highest line is the highest quantile, the second highest line is the second highest quantile, etc.

The predictions are stored in the matrix yhat. The following commands would duplicate the previous diagram:

```
> plot(price, fit$yhat[, 1], xlab = "price", ylab = "ratio", ylim = c(min(fit$yhat),
+     max(fit$yhat)), type = "l")
> lines(price, fit$yhat[, 2])
> lines(price, fit$yhat[, 3])
> lines(price, fit$yhat[, 4])
> lines(price, fit$yhat[, 5])
```

The graph.yhat option controls whether the graph is printed when the qreglwr command is issued.

The qreglwr command does not have an explicit limit on the number of quantiles. Thus, qreglw(ratio price,graph.yhat=T, taumat=seq(.10,.90,.10)) will lead to a graph with 9 quantiles ranging from .10 to .90 in increments of .10. By default, qreglwr specifies a window of .50. Larger values will produce smoother quantile lines; smaller values will produce noisier lines.

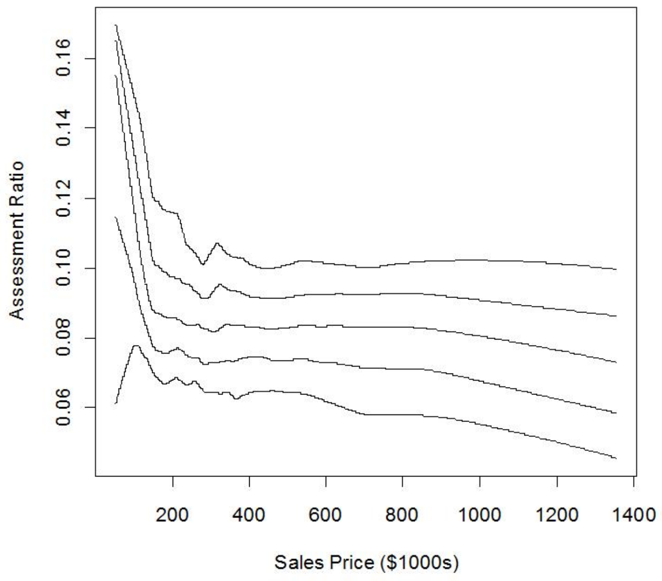
The following commands apply the qreglwr estimator to the dupage99 data set:

```
> fit <- qreglwr(dupage99$ratio ~ dupage99$price)
> plot(dupage99$price, fit$yhat[, 1], type = "l", ylim = c(min(fit$yhat),
+     max(fit$yhat)), xlab = "Sales Price (1000s)", ylab = "Assessment Ratio")
> lines(dupage99$price, fit$yhat[, 2])
> lines(dupage99$price, fit$yhat[, 3])
> lines(dupage99$price, fit$yhat[, 4])
> lines(dupage99$price, fit$yhat[, 5])
```

Like the nonlinear regression estimates, the quantile estimator suggests that assessment ratios decrease with sales price through most parts of the distribution. In this case, the quantile approach does not provide much additional information. Apart from a slight tendency for a greater variance as sales price increase, the lines are close to being parallel, which suggests that simple regression procedures provide an adequate portrayal of the assessment process in this case.

Perhaps a better example of what can be learned from the quantile approach comes from a sample of assessment ratios in Cook County IL. The data set (not included here) comprises 28,010 residential assessments from 2006. All of the homes sold in 2006. Unlike DuPage County, the statutory assessment rate in Cook County is 0.16, although systematic under-assessment produces an average assessment ratio for the sample of 0.087. The qreglwr command produces the results for this data set found in Figure 3.

Figure 3: Results from “qreglwr”



The outstanding feature of this graph is the extraordinarily high degree of variability in assessment ratios at low sales prices. From about $200,000 to $800,000, assessment ratios are roughly proportional to sales price. Approximately 10% of the assessment ratios are above the highest line (the 90% quantile line) and 10% are below the lowest line (the 10% quantile line). Thus, approximately 80% of the assessment ratios line in a range of about 6% - 10%, even though the statutory rate is 16%. A pattern of regressivity is primarily evident only at very low sales prices, but even this pattern is overwhelmed by the sheer variability of assessments for low prices.

7 Conditional Density Estimation

Quantile regressions are one way of analyzing conditional densities: given a sales price, what does the distribution of assessment ratios look like? A point on the 10% quantile line shows (approximately) the 10% percentile of this conditional distribution. Another way to approach this problem is to directly estimate the conditional density function. If x and y are two random variables, then f(x,y) is the joint density function and the conditional density for y given x is f(y|x) = f(x,y)/f(x), where f(x) is the unconditional density function for x. These functions are easy to estimate using nonparametric kernel density estimators.

The condens command calculates conditional density estimate for f(assessment ratio | sales price): conditional on the sales price, what does the distribution of assessment ratios look like? The lattice package is used to produce a rich set of graphs that provide alternative ways of viewing the conditional density estimates.

The following series of command illustrates the use of condens for the dupage99 data set.

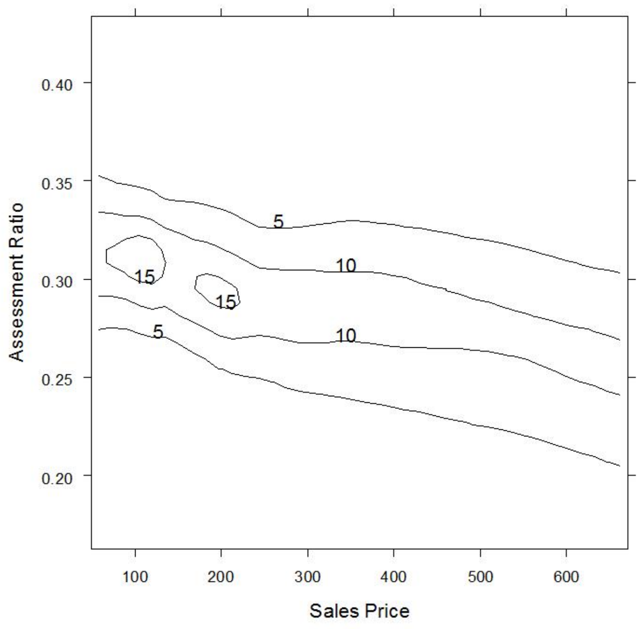
```
> library(aratio)
> par(ask = T)
> data(dupage99)
> dupage99$ratio <- dupage99$av/dupage99$price
> dropobs1 <- nptrim_obs(dupage99$ratio, k = 3)

Total number of observations                       2000.0000000
Number of missing observations                        0.0000000
Number of non-missing observations                 2000.0000000
Number of non-missing observatons trimmed             8.0000000
Total number of observations dropped                  8.0000000
Number of non-missing observations after trimming  1992.0000000
minimum                                               0.1533382
25th percentile                                       0.2785128
75th percentile                                       0.3188254
maximum                                               0.6126000
lower bound for trimming                              0.1575750
upper bound for trimming                              0.4397633

> dropobs2 <- (dupage99$price < quantile(dupage99$price, 0.01)) |
+     (dupage99$price > quantile(dupage99$price, 0.99))
> dupage99 <- dupage99[dropobs1 == FALSE & dropobs2 == FALSE, ]
> o <- order(dupage99$price)
> dupage99 <- dupage99[o, ]
> dupage99$price <- dupage99$price/1000
> fit <- condens(price, ratio, window = 0.5, contour = TRUE, level = TRUE,
+     wire = TRUE, dens = TRUE, targetx.dens = c(100, 200, 300,
+         400), ngrid = 40, data = dupage99)
```

By default, condens only produces a contour plot. In this case, all of the plot options have been specified, so contour, level, wire, and density graphs all are produced. Figure 4 contains the contour map. The level map and the wire map are contained if Figures 5 and 6 respectively. The density map is Figure 7.

Figure 4: The Contour Map

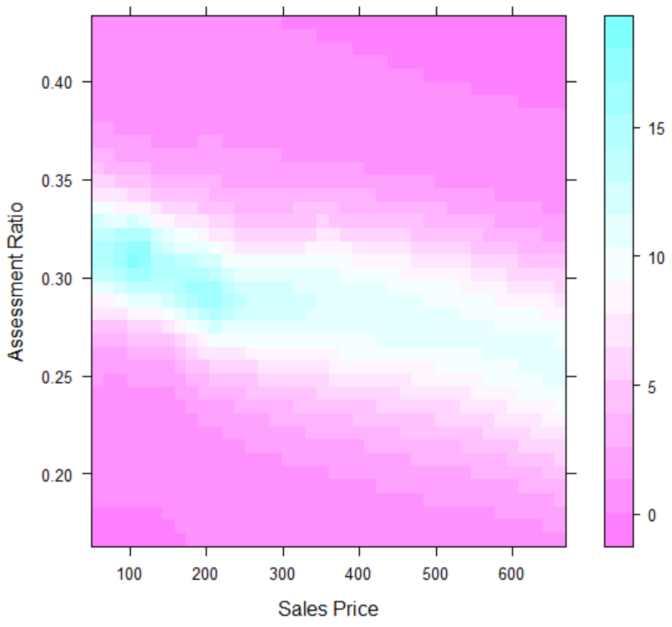


The contour and level plots show the same thing in different formats. The bands or colors show contours of equal densities. Much like the quantile lines, they suggest that assessment ratios tend to decline with sales price. The distribution of ratios widens somewhat as sales price increase. The heavy concentration of assessment ratios in the region with relatively low sales prices shows that there is relatively low variability in assessment rates in these regions.

The wire plot is not very informative in this case. The lattice package can be used to vary the point of view for wire plots. Rotating the plot can sometimes provide some useful perspectives. Overall, I have found the contour and level plots to be much more useful than the wire plot.

The density plots provide a much different perspective on the conditional density function. In this case, the density plot shows four conditional density functions for assessment ratios –the density function conditional on a $100,000 sales price, a $200,000 sale price, etc. As the sales price increases, the density functions shift to the left; this result again implies that higher sales prices are associated with lower assessment ratios. It also is clear from this density plot

Figure 5: The Level Map



that the conditional density functions spread out as sales price increases. The graph shows clearly that the variance of the conditional density functions increases as sales price increases. Together, the contour, level, and density plots provide useful visual summaries of the relationships between sales prices and assessment ratios across their full distribution.

By default, the density plot presents five conditional density estimates one each for the 10th, 25th, 50th, 75th, and 90th percentiles of the sales price distribution. These percentiles can be altered using the quantile.dens option. Alternatively, target sales prices can be entered directly using the targetx.dens option, as was done here. No more than 5 values should be listed for quantile.dens or targetx.dens, although fewer entries are acceptable. If quantile.dens or targetx.dens has more than 5 entries, only the first 5 are used.

Figure 6: The Wire Map

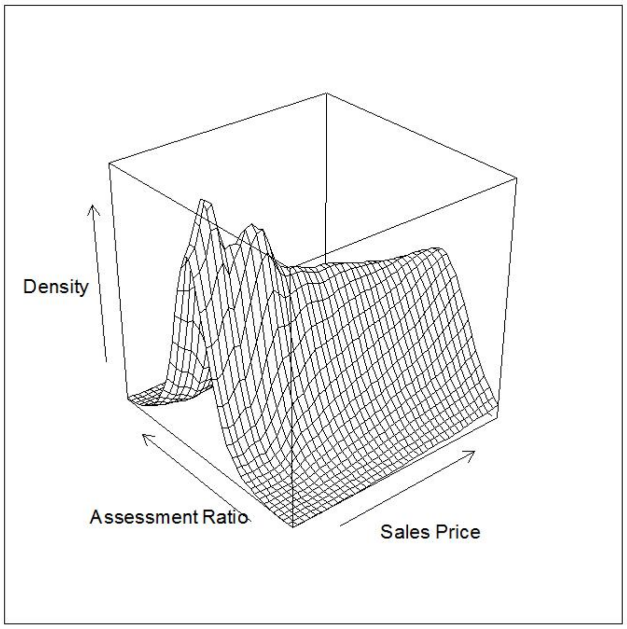
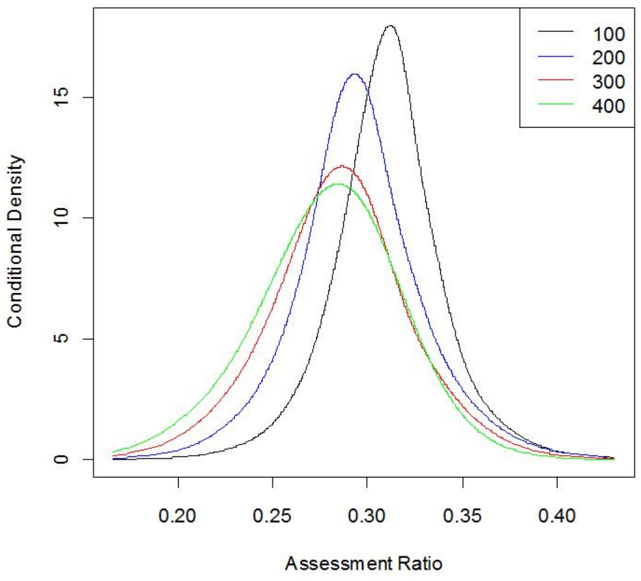


Figure 7: The Density Map


Conditional Density
Assessment Ratio