Density estimation (smoothing) and local regression

Eric Feigelson (Penn State) edf@astro.psu.edu

2nd East Asian Workshops in Astrostatistics Summer 2018

Adapted from R scripts in Appendix B, *Modern Statistical Methods for Astronomy With R Applications*, Eric D. Feigelson & G. Jogesh Babu 2012 http://astrostatistics.psu.edu/MSMA (http://astrostatistics.psu.edu/MSMA)

Nonparametric density estimation seeks an optimal continuous curve derived from low-dimensional point processes. Astronomers call this 'smoothing the data'. As with regression, a response variable ('y' axis) must be chosen so that the dispersion in that variable is minimized in some fashion. Typically, a smoothing 'bandwidth' must be chosen or calculated. Too large a bandwidth will miss structural details (increase bias) while too small a bandwidth will add noise (increase variance).

```
In [ ]: setwd('/Users/ericfeigelson/Desktop/Rdir')

        # Regenerate bivariate dataset from first tutorial

        set.seed(1)
        x <- sample(seq(0.01, 3, length.out=500))
        y <- 0.5*x + 0.3^(x^2) + rnorm(500, mean=0, sd=(0.05*(1+x^2)))
        xy <- cbind(x, y)
        plot(xy, pch=20)
```

We start with a kernel density estimation with a Gaussian kernel chosen with the direct plug-in algorithm. **Exercise:** Try different kernel widths: different 'scalest' parameter values within dpik. Try 'locpoly', another smoothing functions in the KernSmooth package.

In [ ]:
```
# I. Kernel density estimator with two visualizations

par(mfrow=c(1,2))
library(KernSmooth)
dpik(x) ; dpik(y)
smxy <- bkde2D(xy, bandwidth=c(dpik(x),dpik(y)))
image(smxy$x1, smxy$x2, smxy$fhat, col=topo.colors(30), xlab='Xvar',
ylab='Yvar', cex.lab=1.2)
contour(smxy$x1, smxy$x2, smxy$fhat, add=T)
persp(smxy$x1,smxy$x2, smxy$fhat, theta=100, phi=40, shade=0.1, col='g
reen1', xlab='Xvar', ylab='Yvar', zlab='Density')
```

II. Two spline fits

We start with a standard cubic smoothing spline fit. This function is based on code in the GAMFIT Fortran program by T. Hastie and R. Tibshirani (http://lib.stat.cmu.edu/general/ (http://lib.stat.cmu.edu/general/)), which makes use of spline code by Finbarr O'Sullivan. [Chair of Statistics, Univ College Cork IE]. We then proceed with a more modern spline regression that prunes knots based on the Bayesian Information Criterion likelihood measure, and computes spline function for any quantile of dispersion in the response variable.
Exercise: Try different parameter options within COBS such as nknots, degree, lambda, and ic. See help(cobs).

In [ ]:
```
cubsplxy <- smooth.spline(log10(xy))
plot(log10(xy), pch=20, cex=0.5, ylim=c(-0.7, 0.4), xlab='log(Xvar)',
     ylab='log(Yvar)', main='Cubic spline fit')  # Plot points
lines(cubsplxy, lwd=2, col='darkgreen')  # Plot the spline fit
knotx <- cubsplxy$fit$knot*cubsplxy$fit$range + cubsplxy$fit$min   # F
ind and plot the spline knots
rug(knotx,col='darkgreen')
```

In [ ]:

```
# COnstrained B-Splines Nonparametric Regression Quantiles
# Bartels, R. and Conn A. (1980) Linearly Constrained Discrete L_1 Pro
blems, ACM Transaction on Mathematical Software 6, 594Ã¢â,¬â€œ608.
# Ng, P. (1996) An Algorithm for Quantile Smoothing Splines, Computati
onal Statistics & Data Analysis 22, 99Ã¢â,¬â€œ118.
# He, X. and Ng, P. (1999) COBS: Qualitatively Constrained Smoothing v
ia Linear Programming; Computational Statistics 14, 315Ã¢â,¬â€œ337.
# Ng, P. and Maechler, M. (2007) A Fast and Efficient Implementation o
f Qualitatively Constrained Quantile Smoothing Splines, Statistical Mo
delling 7(4), 315-328.

install.packages('cobs') ; library(cobs)
plot(log10(xy), pch=20, cex=0.5, ylim=c(-0.7, 0.4), xlab='log(Xvar)',
     ylab='log(Yvar)', main='Spline quartile fit')  # Plot points
cobsxy50 <- cobs(log10(x), log10(y), ic='BIC', tau=0.5)  #  Median reg
ression fit
lines(sort(cobsxy50$x),cobsxy50$fitted[order(cobsxy50$x)], lwd=2, col=
2)
cobsxy25 <- cobs(log10(x), log10(y), ic='BIC', tau=0.25)
lines(sort(cobsxy25$x),cobsxy25$fitted[order(cobsxy25$x)], lwd=1, col=
2)
cobsxy75 <- cobs(log10(x), log10(y), ic='BIC', tau=0.75)
lines(sort(cobsxy75$x),cobsxy75$fitted[order(cobsxy75$x)], lwd=1, col=
2)
rug(cobsxy50$knots, lwd=2, col=2)
```

III. Five well-established bivariate semi-parametric local regression estimators

1. LOESS, 'LOcal regrESSion' in base-R, widely used (0.3M Google hits). Local polynomial regression with robust treatment of outliers. Description in Wikipedia (https://en.wikipedia.org/wiki/Local_regression (https://en.wikipedia.org/wiki/Local_regression)). Presented in the book W. S. Cleveland, `Visualizing Data', Hobart Press 1993
2. Nonlinear quantile regression in CRAN package 'quantreg', widely used (0.3M Google hits). This is mathematically quite advanced; see description in Wikipedia (https://en.wikipedia.org/wiki/Quantile_regression (https://en.wikipedia.org/wiki/Quantile_regression)). Presented in the book R. Koenker`Quantile Regression', Cambridge Univ Press 2005
3. Nonparametric regression with bootstrap errors in CRAN package 'np'. See Hayfield, T. & Racine, J. S. Nonparametric Econometrics: The np package, J. Statist. Software, 27(5), 2008 http://www.jstatsoft.org/v27/i05/ (http://www.jstatsoft.org/v27/i05/)
4. Locfit in CRAN package 'locfit', widely used (>200 downloads/day). Local kernel regression methods including heteroscedastic weighting (unequal error bars), censoring (upper limits), and outliers. Presented in the book Loader, C. (1999) Local Regression and Likelihood Springer, New York.
5. Gaussian Process regression, commonly known as `kriging`. Response variable errors and independent variable covariance are assumed to be normal. Maximum likelihood & Bayesian estimation. Description in book Rasmussen & Williams, Gaussian Processes for Machine Learning, 2006. Other Gaussian processes regression codes are given in CRAN packages 'mlegp' (Maximum Likelihood Estimates of Gaussian Processes) and 'gptk' (Gaussian Processes tool-kit). See also the tutorial at http://www.r-bloggers.com/gaussian-process-regression-with-r/ (http://www.r-bloggers.com/gaussian-process-regression-with-r/).

Finally, we plot all of these nonparametric regressions on the same plot. We find that, in this case, they are quite compatible with each other.

In [ ]:
```
par(mfrow=c(1,2))
sortx <- x[order(x)] ; sorty <- y[order(x)]
local_fit <- loess(sorty ~ sortx, span=0.5, data.frame(x=x,y=y))
summary(local_fit)
plot(x,y,pch=20, cex=0.5, main='LOESS')
lines(sortx, predict(local_fit), lwd=2, col=2)
# Save evenly-spaced LOESS fit to a file
x_seq <- seq(0.0, 3.0, by=0.03)
loc_dat <- predict(local_fit, newdata=x_seq)
write(rbind(x_seq,loc_dat), sep=' ', ncol=2, file='localfit.txt')

install.packages('quantreg') ; library(quantreg)
install.packages('MatrixModels') ; library(MatrixModels)
plot(sortx, sorty, pch=20, cex=0.5, xlab='Xvar', ylab='Yvar', main='Qu
antile')
fit_rqss.25 <- rqss(sorty ~ qss(sortx), tau=0.25)
lines(sortx[-1],fit_rqss.25$coef[1]+fit_rqss.25$coef[-1], col='red')
fit_rqss.50 <- rqss(sorty ~ qss(sortx), tau=0.50)
lines(sortx[-1],fit_rqss.50$coef[1]+fit_rqss.50$coef[-1], lwd=2)
fit_rqss.75 <- rqss(sorty ~ qss(sortx), tau=0.75)
lines(sortx[-1],fit_rqss.75$coef[1]+fit_rqss.75$coef[-1], col='red')
```

In [ ]:
```
install.packages("np") ; library(np)
bw.NW <- npregbw(x, y, regtype='lc', bwtype='fixed')
npplot(bws=bw.NW, ylim=c(0.0,2.5), plot.errors.method="bootstrap",
    plot.errors.bar='I', plot.errors.type='quantiles', main='NP with b
ootstrap CI')
points(x, y, pch=20, cex=0.5)
```

In [ ]:
```
install.packages('locfit')  ;  library(locfit)
locfit_model <- locfit(y~lp(x, nn=0.7))
plot(locfit_model, band='local', ylim=c(0,2.5), col=2, main='locfit ba
ndwidth=0.7')  ;  points(xy, pch=20, cex=0.5)
locfit_model <- locfit(y~lp(x, nn=0.3))
plot(locfit_model, band='local', ylim=c(0,2.5), col=3, main='locfit ba
ndwidth=0.3')  ;  points(xy, pch=20, cex=0.5)
```

In [ ]:
```
install.packages('kernlab') ; library(kernlab)
gpreg <- gausspr(x, y, variance.model=T, cross=10, kerne='polydot',
kpar=list(5))
gpreg
xtest <- seq(from=min(x), to=max(x), length.out=200)
plot(x, y, pch=20, cex=0.5, main='GP regression')
lines(xtest, predict(gpreg, xtest), col='red3', lwd=3)
```

In [ ]:
```
npplot(bws=bw.NW, ylim=c(0.5,1.5), plot.errors.method="bootstrap",
plot.errors.bar='I', plot.errors.type='quantiles')      # Nonparametri
c regression w/ bootstrap errors
points(x, y, pch=20, cex=0.5)
lines(xtest, predict(gpreg, xtest), col='red3', lwd=3)  #  Gaussian Pr
ocesses regression
lines(sortx[-1],fit_rqss.25$coef[1]+fit_rqss.25$coef[-1], col='blue3')
#  Linear quantile regression
lines(sortx[-1],fit_rqss.50$coef[1]+fit_rqss.50$coef[-1], lwd=2, col='
blue3')
lines(sortx[-1],fit_rqss.75$coef[1]+fit_rqss.75$coef[-1], col='blue3')
lines(sortx, predict(local_fit), lwd=2, col='darkgreen') # LOESS
locfit_values <- predict(locfit_model, seq(0,3,length.out=100))
lines(seq(0,3,length.out=100), locfit_values, lwd=2, col="chocolate")
# locfit
legend('topleft', lty=1, lwd=2, c("NP reg w/ bootstrap",'Gauss Proc re
g', 'Quantile reg', 'LOESS', 'locfit'), col=c('black', 'red3', 'blue3'
, 'darkgreen', 'chocolate'))
```