STScI Astrostatistics
R tutorials
Eric Feigelson (Penn State)
November 2011

SESSION 1
Here we have a variety of R/CRAN methods with some advanced analysis.

****
****

I Statistical distributions and random numbers

R provides ~20 statistical distributions. Dozens more available through CRAN's VGAM which provides maximum likelihood estimation for distribution parameters based on the data.

R's standard procedure is to give four functions for each distribution: d, (density or differential), p (distribution or integral), q (quantile), and r (random generation).

```
# Plot illustrative exponential p.d.f. and c.d.f.

help(seq) ; help(plot); help(par) ; help(lines); help(legend)

xdens <- seq(0,5,0.02) # create a vector of evenly-spaced values

par(mfrow=c(1,2))          # set up two-panel figure

plot(xdens,dexp(xdens,rate=0.5), type='l', ylim=c(0,1.5), xlab='', ylab='Exponential p.d.f.',
lty=1)
lines(xdens,dexp(xdens,rate=1), type='l', lty=2)
lines(xdens,dexp(xdens,rate=1.5), type='l', lty=3)

legend(2, 1.45, lty=1, substitute(lambda==0.5), box.lty=0)
legend(2, 1.30, lty=2, substitute(lambda==1.0), box.lty=0)
legend(2, 1.15, lty=3, substitute(lambda==1.5), box.lty=0)

#  Greek letters, super/subscripts, mathematical formulae, etc. can be written with the
`expression' function;  see `help(polymath)'

plot(xdens, pexp(xdens,rate=0.5), type='l', ylim=c(0,1.0), xlab='', ylab='Exponential c.d.f.',
lty=1)
lines(xdens, pexp(xdens,rate=1), type='l', lty=2)
lines(xdens, pexp(xdens,rate=1.5), type='l',lty=3)
par(mfrow=c(1,2))

# Random number generation
# Mersenne-Twister algorithm; 6 others available or user supplied

z <- runif(500)      # uniform random deviates
z                    # print values on the console
bigz <- rnorm(1000000)  # a million normal random numbers in 0.1 sec CPU time


par(mfrow=c(1,2))    # set up 3-panel figure

dotchart(z, main='')
```

```
x <- seq(0, 20, len=1000)  #  seq = vector of sequential values
plot(x, dnorm(x), type='l', xlim=c(0,4), ylim=c(0,1))
lines(x, pnorm(x), lty=2)
legend(1.5, 0.6, legend=c('log(p.d.f)', 'log(c.d.f)'), lty=c(1,2), cex=0.7)
par(mfrow=c(1,1))



# Is the distribution of Andromeda globular cluster K magnitudes consistent with a normal
distribution?

GC_M31 <- read.table('http://astrostatistics.psu.edu/MSMA/datasets/ GlobClus_M31.dat',header=T)
GC_M31
K <- GC_M31[,2]   # extract second column of a data.frame
hist(K, breaks=40)

# Fit normal distribution

library(MASS)
fitdistr(K,'normal')
fitdistr(K,'normal')$loglik   # likelihood useful for model selection (AIC, BIC, etc)

# A best-fit regression is not necessarily a satisfactory model.  Goodness-of-fit test is also
needed.

# Hypothesis tests for normality

install.packages('nortest') ; library(nortest)
install.packages('moments') ; library(moments)
lillie.test(K)       # Kolmogorov-Smirnov test
cvm.test(K)          # Cramer-vom Mises test
ad.test(K)           # Anderson-Darling test
shapiro.test(K)      # Shapiro-Wilks test
pearson.test(K)      # chi-square test
sf.test (K) ; skewness(K) ; agostino.test(K) ; jarque.test(K); kurtosis(K) ; anscombe.test(K) ;
bonett.test(K)


# Some other interesting hypothesis tests in R/CRAN:

#    ks.test, wilcox.test, mood.test (in R) for univariate 2-sample test
#    chisq.test, fisher.test (R) for contingency tables (categorical data)
#    cor (R) Pearson r, Kendall tau, Spearman rho tests for correlation
#    ad.test (ADGofTest) for univariate Anderson-Darling test (more sensitive than KS)
#    surv2.ks (surv2sample) for univariate 2-sample test with censoring (upper limits)
#    cenken (NADA) for bivariate correlation test with censoring
#    dip (diptest) for Hartigan's test for univariate multimodality
#    grubbs.test (outliers) test for outliers
#    durbin.watson (car) test for serial autocorrelation
#    cramer.test (Cramer) for multivariate 2-sample test with bootstrap resample
#    mshapiro.test (mvnormtest) for multivariate normality test
#    moran.test (sped) test for randomness vs. autocorrelation in 2 or more dimensions
#    kuiper.test, r.test, rao.test, watson.test (CircStats) tests for uniformity of circular data


# Parameter estimation using resampling data with replacement (= bootstrap)

mean(K) ; sd(K)
mean(sample(K,1000,replace=T))
sd(sample(K,1000,replace=T))
```

```
median(K) ; mad(K)
median(sample(K,1000,replace=T))
mad(sample(K,1000,replace=T))


*************************
*************************


# Estimating and characterizing point processes
# Density estimation = recovering the smooth population density from discrete data


# Construct large and small samples of SDSS quasar redshifts and r-i colors

qso <- read.table('http://astrostatistics.psu.edu/MSMA/datasets/SDSS_QSO.dat',head=T)
dim(qso) ; names(qso) ;
summary(qso) ; attach(qso)
z.20 <- z[1:20000]
r.i.all <- r_mag - i_mag ; r.i.10 <- r.i.all[1:20000]

# Plot histogram and quantile function

par(mfrow=c(1,2))
hist(z.20, breaks='scott', main='', xlab='Redshift', col='black')
plot(quantile(z.20, seq(1,100,1)/100, na.rm=T), pch=20, cex=0.5, xlab='Percentile',
ylab='Redshift')


# Constant kernel density estimator

plot(density(z.20), bw=bw.nrd(z.20), main='', xlab='Redshift', lwd=2)

# Adaptive kernel smoother

install.packages('quantreg') ; library(quantreg)
akern.zqso <- akj(z.20, z=seq(0,5,0.01), alpha=0.5)
str(akern.zqso)
plot.window(xlim=c(0,5), ylim=c(0,0.6))
plot(seq(0,5, 0.01), akern.zqso$dens, pch=20, cex=0.5, xlab='Redshift',
ylab='Density')
rug(sample(z.20, 500))

# Note that the constant kernel estimator missed some detailed structure around redshift z=1 and
created some likely-unreliable structure around z=4.  There is a vast literature on the selection
of kernel bandwidths:  too wide misses detail (bias), too narrow increases noise (variance).  A
standard procedures is to minimize the `mean integrated square error', MISE = Bias^2 + Variance.
Another common choice (for large samples) is to retain a portion of the sample for `cross-
validation' of kernel bandwidths; this is the default bandwidth in R's density estimator.
Despite research since the 1960s, there is no consensus on a best adaptive kernel algorithm.

# An important suite of methods in density estimation is known as `local regression' or
`nonparametric regression'. Here a simple parametric function (e.g. parabola) is fit to the local
data around each point in a chosen bandwidth (or `window').  This can produce reasonable smooth
curves for data with unusual behaviors (e.g. change points, heteroscedasticity), and is now
preferred over the older spline methods.  The widely-used method from the 1990s is called LOESS
by William Cleveland.   An important recent innovation is the calculation (using bootstrap
resampling) of confidence bands around the local regression curve.  This procedure is rather
complex.  See an example in astronomy (Wang, X., Woodroofe, M., Walker, M. G., Mateo, M. &
Olszewski, E. 2005, Estimating Dark Matter distributions, Astrophys. J., 626, 145-158) and a full
discussion with R code in a text (Takezawa, K. 2005 Introduction to Nonparametric Regression,
```

Wiley-Interscience).

```
# Distribution with and without measurement errors

# Few statistical methods treat heteroscedastic measurement errors.  See advanced texts on
measurement error regression models (Carroll, Ruppert & Stepanski 2006; Buonaccorsi 2010).  For
density estimation, methods are still in the research domain.  The method of Delaigle & Meister
(2008) has been coded in CRAN package `decon'.


aster <- read.table('http://astrostatistics.psu.edu/MSMA/datasets/asteroid_dens.dat', head=T)
summary(aster) ; dim(aster) ; attach(aster)

install.packages('sfsmisc') ; library(sfsmisc)
errbar(seq(1,26), Dens, Dens+Err, Dens-Err)  # plot data with error bars

install.packages('decon') ; library(decon)
par(mfrow=c(2,1))
plot(ecdf(Dens), main='', xlab=expression(Asteroid~density~g/cm^3), ylab='c.d.f.', verticals=T,
cex=0, lwd=1.5)
x <- seq(0, 6, 0.02)
cdf_sm <- DeconCdf(Dens,Err,x,bw=0.1)
lines(cdf_sm, lwd=1.5)
plot(DeconPdf(Dens,Err,x,bw=0.1), main='', xlab=expression(Asteroid~density~~ g/cm^3),
ylab='p.d.f.', lwd=1.5, add=T)
lines(density(Dens, adjust=1/2), lwd=1.5, lty=2)
par(mfrow=c(1,1))

# Note the result that the accentuates the evidence that there are two species of asteroids, one
icy or porous (density ~ 1 g/cm^3) and another rocky (~3 g/cm^3).  However, there is no test for
the significance of this estimator.


# Density estimation in three-dimensions: galaxy redshift survey
# CRAN package `spatstat' applied methods from geostatistics and spatial point processes.

shap <- read.table('http://astrostatistics.psu.edu/MSMA/datasets/Shapley_galaxy.dat', header=T,
fill=T)
attach(shap) ; dim(shap) ; summary(shap)
shap.lo <- shap[(R.A. < 214) & (R.A. > 209) & (Dec. > -34) & (Dec. < -27) ,]

********************
********************

# Some 3-D visualization capabilities

install.packages('rgl') ; library(rgl)
rgl.open()
rgl.points(scale(shap[,1]), scale(shap[,2]), scale(shap[,4]))
rgl.bbox()
rgl.close()

plot(shap.lo[,1], shap.lo[,2], cex=(scale(shap.lo[,4])+1.5)/2, xlab='Right Ascension (degrees)',
ylab='Declination (degrees)')

install.packages('scatterplot3d') ; library(scatterplot3d)
scatterplot3d(shap.hi[,c(1,2,4)], pch=20, cex.symbols=0.7, type='p', angl=40, zlim=c(0,50000))

# Preparation for spatstat analysis
```

```
install.packages('spatstat') ; library(spatstat)
shap.lo.win <- owin(range(shap.lo[,1]), range(shap.lo[,2]))
shap.lo.ppp <- as.ppp(shap.lo[,c(1,2,4)], shap.lo.win)
plot(density(shap.lo.ppp,0.3), col=topo.colors(20), main='', xlab='R.A.', ylab='Dec.')
plot(shap.lo.ppp, lwd=2, add=T)

# Ripley's K function for the Shapley low density region
# This is the cumulative function of the 2-point correlation function

shap.lo.K <- Kest(shap.lo.ppp, correction='isotropic')
shap.lo.K.bias <- Kest(shap.lo.ppp, correction='none')
plot.fv(shap.lo.K, lwd=2, col='black', main='', xlab='r (degrees)', legend=F)
plot.fv(shap.lo.K.bias, add=T, lty=3, lwd=2, col='black', legend=F)

# Draw envelope of 100 simulations of random spatial distribution

shap.lo.K.env <- envelope(shap.lo.ppp, fun=Kest, nsim=100, global=T)
xx <- c(0, shap.lo.K.env$r, rev(shap.lo.K.env$r), 0)
yy <- c(c(0, shap.lo.K.env$lo), rev(c(0,shap.lo.K.env$hi)))
polygon(xx, yy, col='gray')
plot.fv(shap.lo.K, lwd=2, col='black', main='', add=T, legend=F)
plot.fv(shap.lo.K.bias, add=T, lty=3, lwd=2, col='black', legend=F)

# Baddeley J function for the Shapley low density region

plot(Jest(shap.lo.ppp), lwd=2, col='black', cex.lab=1.3, cex.axis=1.3, main='', xlab='r
(degrees)', legend=F)

# Two-point correlation function

shap.lo.pcf <- pcf(shap.lo.ppp)
plot(log10(shap.lo.pcf$r[2:512]), log10(shap.lo.pcf$trans[2:512]), type='l', lwd=2, xlab='log r
(degrees)', ylab='log pair correlation fn')

# ~50 new R/CRAN functions are used in this tutorial:
# rnorm, par(mfrow), lillie.test, cvm.test, ad.test, shapiro.test,
# pearson.test, sf.test, skewness, agostino.test, jarque.test, kurtosis,
# anscombe.test, bonett.test, sample, hist, quantile, density, bw.nrd,
# akj, plot.window, rug, errbar, seq, expression, &, <, >, DeconCdf,
# DeconPdf, rgl.open, rgl.points, rgl.bbox, rgl.close, scale, scatterplot3d
# owin, range, as.ppp, topo.colors, Kest, plot.fv, envelope, rev, polygon,
# Jest, pcf, log10
```