myPackage Weed Example

Liang

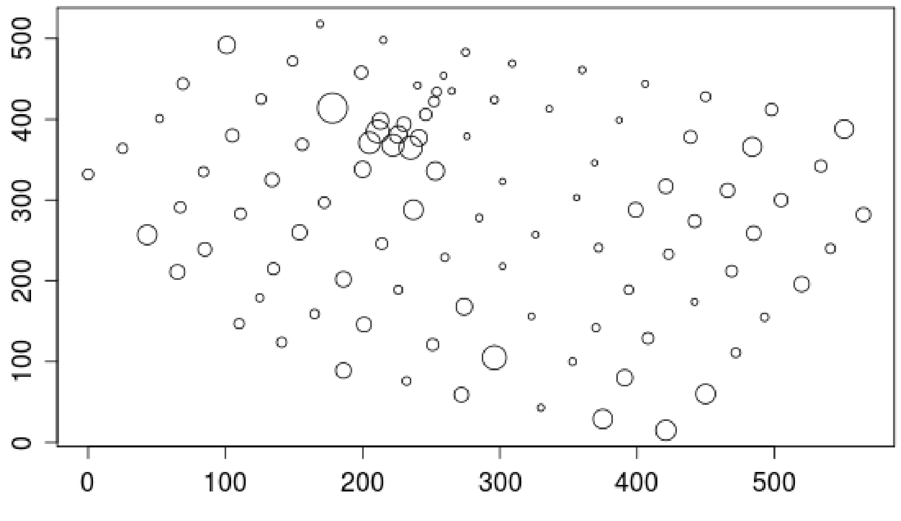
07-30-2011

1 Example for Weed Data

```
# Load myPackage
require(myPackage)
myPackage_hello_world()
```

1. Load and plot Weed data.

   ```
   data(datWeed)
   n <- nrow(dat)
   YRaw <- dat[,3]
   locRaw <- dat[,1:2]
   locRaw.u <- unifLoc(locRaw)
   plotData(YRaw, locRaw)
   ```



2. Randomly choose 60 observations as training data

   ```
   set.seed(111)
   ind <- sample(1:n, 60, replace=FALSE)
   L <- rep(1, length(ind))
   Y <- YRaw[ind]
   loc <- locRaw.u[ind,]
   plotData(Y/L, loc, size=c(0.3, 3.7))
   ```

3. Tuning and run MCMC algorithm: optimal acceptance rates for $S1$ and m are around 0.55 and 0.40 for s and a.

   - Single chain generation

     ```
     input0 <- MCMCinput( run=2000, run.S=1, rho.family="rhoPowerExp",
               Y.family = "Poisson", ifkappa=0,
               scales=c(0.6, 3.5, 0.9, 0.6, 0.5),
               phi.bound=c(0.005, 1),
               initials=list(c(-1), 1, 0.1, 1) )
     res <- runMCMC(Y, L = L, loc=loc, X=NULL, MCMCinput = input0 )
     # Cut chains
     res.m <- cutChain(res, chain.ind=1:4, burnin=500, thining=10)
     ```

   - Multiple chain generation with parallel computing

     ```
     input0 <- MCMCinput( run=2000, run.S=1, rho.family="rhoPowerExp",
               Y.family = "Poisson", ifkappa=0,
               scales=c(0.6, 3.5, 0.9, 0.6, 0.5),
               phi.bound=c(0.005, 1),
               initials=list(c(-1), 1, 0.1, 1) )
     require(multicore)
     options(cores=5)
     res.prl <- runMCMC.multiChain(Y, L = L, loc=loc, X=NULL,
                  MCMCinput = input0, n.chn = 5)
     res.m.prl <- lapply(res.prl, cutChain, chain.ind=1:4, burnin=200, thining=10)
     res.mix <- mixChain(res.m.prl)
     res.m <- res.mix
     ```

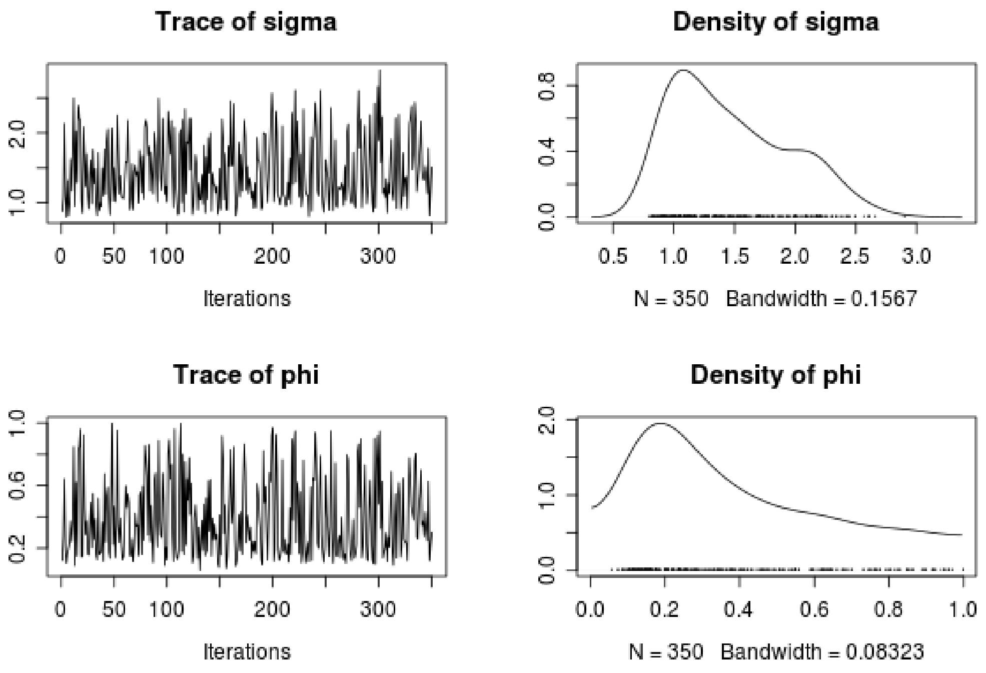
4. Examine chains and diagnostics

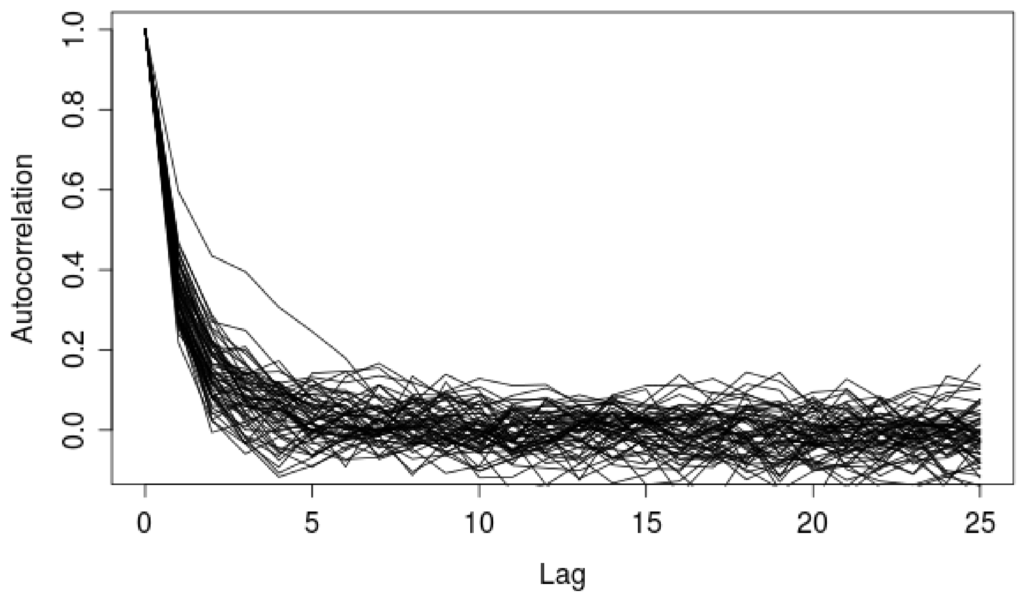
   ```
   require(coda)
   ## for covariance matrix parameters
   chn1 <- cbind(sigma=res.m$s, phi=res.m$a)
   chn1.mcmc <- mcmc(chn1); dim(chn1.mcmc)
   summary(chn1.mcmc)
   ```

```
plot(chn1.mcmc, auto.layout = TRUE)
crosscorr.plot(chn1.mcmc)
autocorr.plot(chn1.mcmc)
effectiveSize(chn1.mcmc)
geweke.diag(chn1.mcmc, frac1=0.1, frac2=0.5)
heidel.diag(chn1.mcmc, eps=0.1, pvalue=0.05)

## for coefficients
if(is.matrix(res.m$m)){
  beta.mcmc <- mcmc(t(res.m$m))
  } else beta.mcmc <- mcmc(res.m$m)
plot(beta.mcmc, auto.layout = TRUE)
crosscorr.plot(beta.mcmc)
autocorr.plot(beta.mcmc)

## for latent variables
S.mcmc <- mcmc(t(res.m$S))
dim(S.mcmc)
plotACF(S.mcmc)
crosscorr.plot(S.mcmc)
```



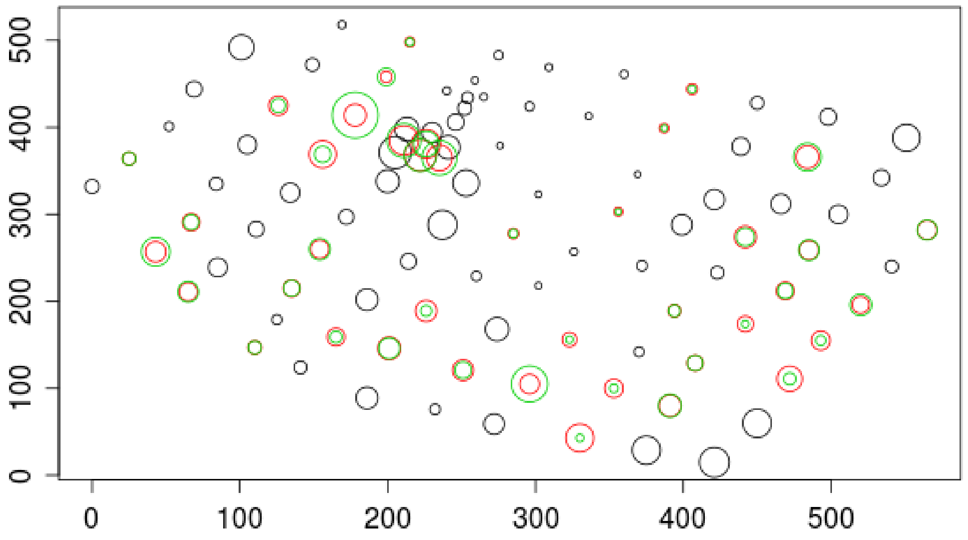


5. Select the remaining locations as predicting locations

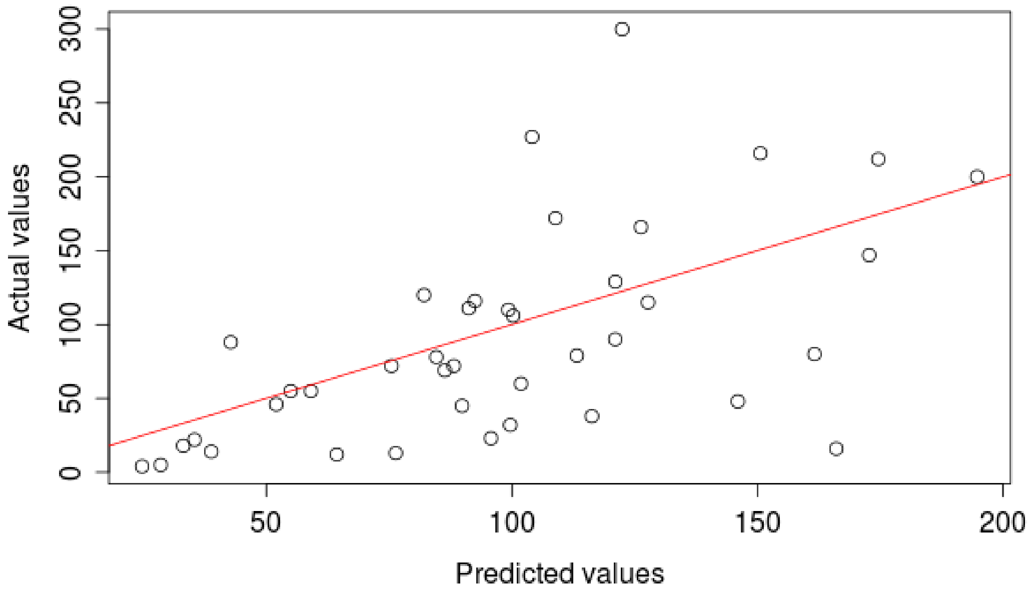
```
ind.left <- setdiff(1:n, ind)
ind.p <- ind.left
locp <- locRaw.u[ind.p,]
Lp <- rep(1, nrow(locp))
```

6. Predict and plot data

```
Ypred <- predY(res.m, loc, locp, X=NULL, Xp=NULL, Lp=Lp, k=1,
          rho.family="rhoPowerExp", Y.family="Poisson")
Ypred.avg <- rowMeans(Ypred$Y);
# plot observed, predicted and actual data
#op <- par(mfrow=c(2,1))
plotData(Y/L, locRaw[ind,], Ypred.avg/Lp, locRaw[ind.p,],
          YRaw[ind.p]/Lp, locRaw[ind.p,])
plot(Ypred.avg/Lp, YRaw[ind.p]/Lp)
abline(c(0,1), col=2)
#par(op)
```



(black circle = training, red circle = predicted, green circle = actual; the size of circles indicates the value of response)



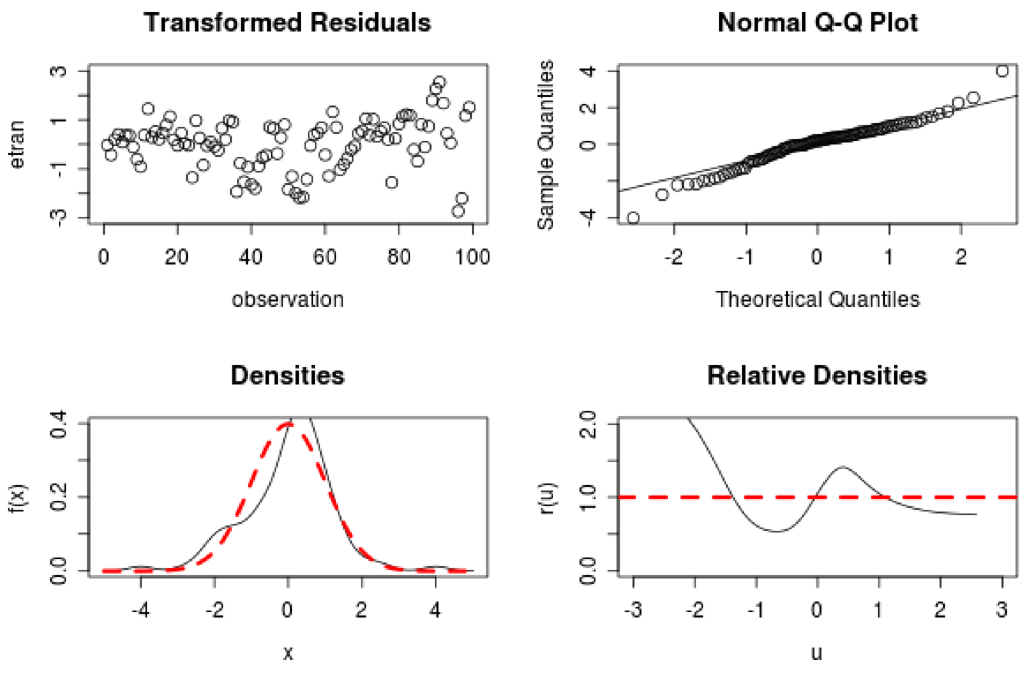
7. Model checking

```
# model checking procedure was ran on high performance computing cluster;
# and the code is too long to be included here
# ......
```

Weed data

	"expnt"	"explt"
$\hat{\boldsymbol{\beta}}$	4.07	(4.35, 0.10, -1.51)
$\hat{\sigma}$	1.09	1.22
$\hat{\phi}$	0.17	0.19
p-value	0.315	0.004
rej. rate	0.005	1.00

Weed data vs. "expnt"


Transformed Residuals
etran
observation
Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
Densities
f(x)
x
Relative Densities
r(u)
u