Example using the generic post-processing functions

Heather Savoy

January 23, 2015

First we need to configure some project specifics:

```
xpath <- "C:/Users/Heather/Desktop/SimulationData"  #note the slash direction
projectName <- "transp_2d"
resultName <- "local_03"
aprioriFile <- "ext_01"
numTimeSteps <- 48  #can't use 49
numSamples <- 200
numRealizations <- 100
```

Load the necessary libraries and auxillary scripts

```
suppressMessages(library(np))
```

```
## Warning: package 'np' was built under R version 3.1.1
```

```
suppressMessages(library(RSQLite))
```

```
## Warning: package 'RSQLite' was built under R version 3.1.1
```

```
## Warning: package 'DBI' was built under R version 3.1.1
```

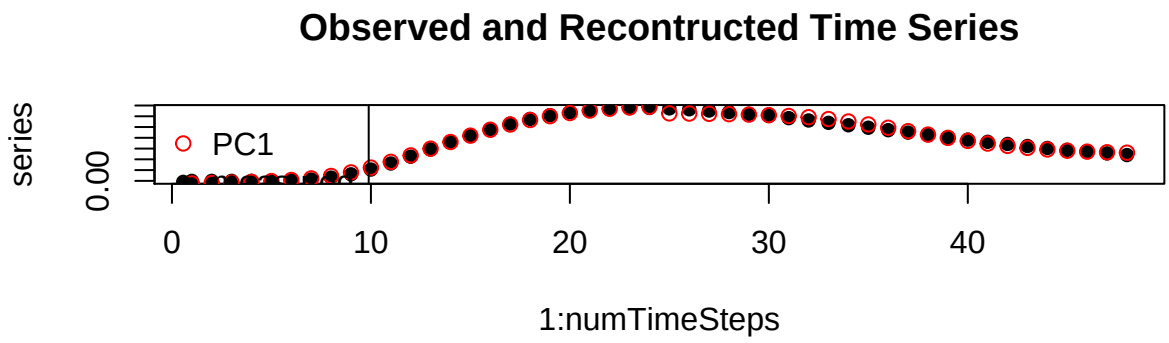
```
source("sql_func.R")
source("postproc.R")
```

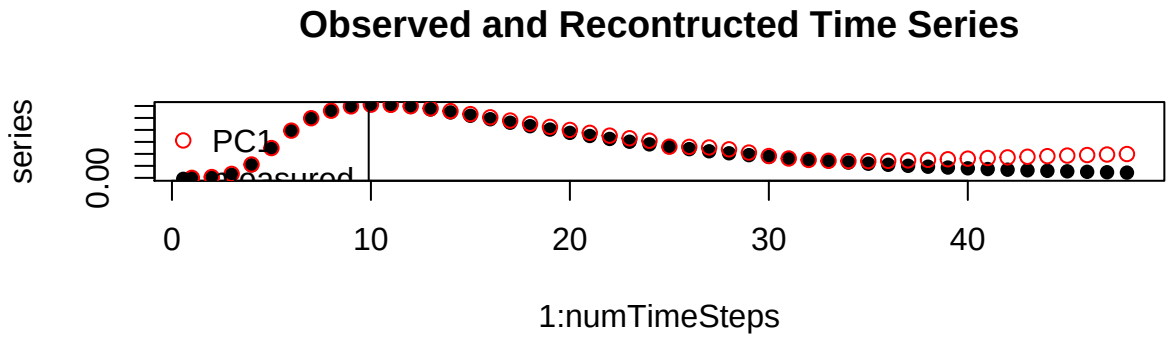
We need to read in the measurements and apply PCA, and we do this with the **grabObs** function (defined in *sql_func.R*) and the **timePCA** function (defined in *postproc.R*). We can also plot the reconstructed time series to get a visual idea of what information we are retaining by using the **plotReconstruct** function (defined in *postproc.R*). We can see that PCA is probably not the best fitting method for these breakthrough curves.

```
# read in the observed time series
obs1 <- grabObs(1:numTimeSteps)
obs2 <- grabObs((1+numTimeSteps)+1:numTimeSteps)
# save the first component's coefficients and the means
coeffs1 <- timePCA(obs1)
coeffs2 <- timePCA(obs2)
# reconstruct time series and plot
par(mfrow = c(2, 1))
plotReconstruct(obs1,coeffs1)
```

```
## NULL
```

```
plotReconstruct(obs2,coeffs2)
```






```
## NULL
```

Next we can apply PCA to every time series in every realization of every sample using the **grabSamp** function (defined in *sql_func.R*) to retrieve the realization time series and the **timePCA** function (defined in *postproc.R*) to apply PCA to each time series. Then, we calculate the likelihood value for each sample by comparing the individual observed time series to the respective realizations in each sample by using the function
npLike (defined in **postproc.R**).

```
likelihood <- c()
for(sample in 1:numSamples){
  # Read from database
  realz <- grabSamp(sample)
  realzCoeffs1 <- realzCoeffs2 <- matrix(NA,ncol=4,nrow=numRealizations)
  for(realization in 1:numRealizations){
    # Apply PCA to both time series
    realzCoeffs1[realization,] <- timePCA(realz[realization,1:numTimeSteps])
    realzCoeffs2[realization,] <- timePCA(realz[realization,
                                                (1+numTimeSteps)+ 1:numTimeSteps])
  }
  # Calculate likelihood
  lik1 <- npLike(realzCoeffs1, coeffs1)
```

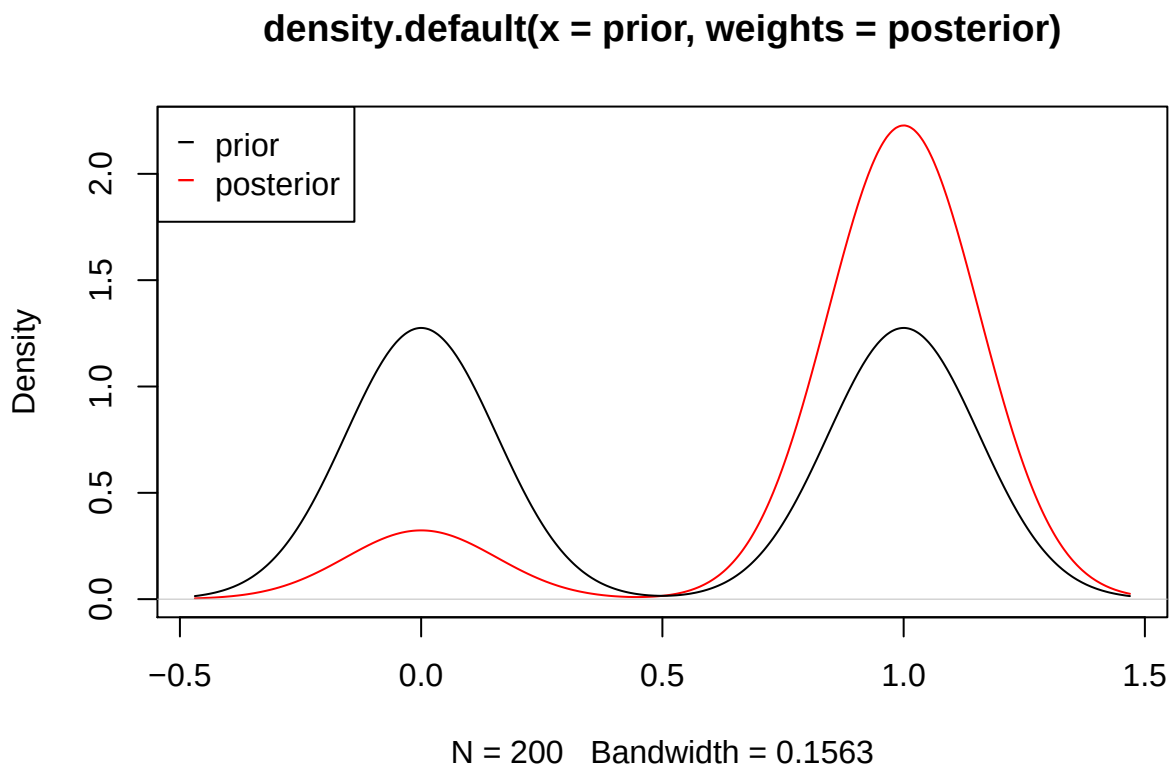
```
  lik2 <- npLike(realzCoeffs2, coeffs2)
  likelihood[sample] <- lik1 * lik2
}
```

Now we can calculate the posterior. First by calling the **grabPrior** function (defined in *sql_func.R*). I only request the samples for which I ran simulations and for only the omega parameter (number five in the list of parameters in the database). Then I call **posterior** (defined in *postproc.R*) to calculate the posterior densities.

```
prior <- grabPrior(1:numSamples, param=5)
post <- posterior(prior,likelihood)
```

And finally we can look at our posterior versus prior. I set the graphics window for only one plot and then call **plotPosterior** (defined in *postproc.R*) to plot the probability density functions of a prior and the respective (marginal) posterior.

```
par(mfrow = c(1, 1))
plotPosterior(prior,post)
```




```
## NULL
```