myPackage Introduction

Liang

07-30-2011

1 General Information

1.1 Package Dependency

- R (>= 2.12.0)
- GNU Compiler Collection (GCC): C++ compiler needed
- GNU Scientific Library (GSL): a numerical library for C and C++
- LAPACK and BLAS (or ATLAS): for numerical linear algebra operations
- R packages
  - Rcpp (>= 0.9.4): provides a C++ API as an extension to the R system
  - RcppArmadillo (>= 0.2.19): integration for "Armadillo" which is a templated C++ linear algebra library
  - RcppGSL (>= 0.1.1): Rcpp integration for GNU GSL
  - {coda}: for Markov chain diagnostics
  - {distrEx}: for calculating Hellinger and Kolmogorov distances between two distributions
  - {multicore, snow, Rmpi}: for parallel computing
- Standard development tools such as make etc.

1.2 Package Structure

0– myPackage
  1– data
    2– datalist
    2– datEarthquake.RData
    2– ...
  1– man
    2– locCircle.Rd

2– ...
1– R
2– checkingBMC.R
2– ...
1– src
2– Makevars
2– runMCMC.h
2– runMCMCBP.cpp
2– ...
1– configure
1– cinfigure.in
1– DESCRIPTION
1– NAMESPACE

- **data** directory: contains data sets; use "data(...)" to load data sets after the package is loaded in R.
- **man** directory: contains documentation files for the objects in the package in R documentation (Rd) format. (under development)
- **R** directory: contains R code files.
- **src** directory: contains C++ source and header files, plus Makevars.
- ...

*See `http://cran.r-project.org/doc/manuals/R-exts.html#Package-structure` for details.

1.3 Installation

(The package is only configured for linux/unix system for now.)

- In R: *install.packages("myPackage_1.0.tar.gz")*
- Outside R: unzip "*myPackage_1.0.tar.gz*"; in terminal use "*R CMD INSTALL myPackage*"

*See `http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages` for details.

2 Major Functions

(Note: functions are constantly added due to further development of methodologies and techniques and their generalization to more models.)

2.1 Simulating/Plotting Data

- simData(*loc, L=0, X=NULL, beta=0, cov.par, rho.family = "rhoPowerExp", Y.family="Poisson"*)
  - simulate response and latent Gaussian process for any given locations
- locCircle(*r, np*); locSquad(*r, np*)
  - produce circular or square locations
- unifLoc(*loc, scale=1*)
  - scales locations into unit grid
- plotData(*Y, loc, Yp=NULL, locp=NULL, Yt=NULL, loct=NULL, col=1:2, colt=3, pch=1, size=c(0.3, 2.7)*)
  - plots observed (predicted, actual) data
- plotDataBD(*bdry, Y=NULL, loc=NULL, Yp=NULL, locp=NULL, Yt=NULL, loct=NULL, col=1:2, colt=3, pch=1, size=c(0.3, 2.7)*)
  - plots observed (predicted, actual) data and their boundaries
- plotTexas(*bdry=TexasCounty.boundary, ind.col=NULL*)
  - plots Texas counties with boundaries

2.2 Posterior Sampling and Prediction

- runMCMC(*Y, L=0, loc, X=NULL, run = 200, run.S = 1, rho.family = "rhoPowerExp", Y.family = "Poisson", ifkappa = 0, scales = c(0.5, 3, 0.8, 0.7, 0.15), phi.bound = c(0.005, 1), initials = list(c(1), 1.5, 0.2, 1), MCMCinput=NULL, partial = FALSE, famT=1*)
  - applies robust MCMC algorithm on geostatistical data
  - computation is conducted in embedded C++ programs
  - assumed model: Poisson log-link Gaussian process model; Binomial logit-link Gaussian process model
  - correlation function: powered.exponential or matern family
  - support covariates
  - support partial posterior sample generation for several diagnostic statistics

  - output: a list containing posterior samples of $(S, \beta, \sigma, \phi, \kappa)$ and their acceptance rates

- runMCMC.multiChain(*Y, L=0, loc=loc, X=NULL, run = 200, run.S = 1, rho.family = "rhoPowerExp", Y.family = "Poisson", ifkappa = 0, scales = c(0.5, 3, 0.8, 0.7, 0.15), phi.bound = c(0.005, 1), initials = list(c(1), 1.5, 0.2, 1), MCMCinput=NULL, partial = FALSE, famT=1, n.chn = 2*)
  - parallel computing version of "runMCMC()"
  - chains with different starting values run simultaneously
- predY(*res.m, loc, locp, X=NULL, Xp=NULL, Lp=0, k=1, rho.family="rhoPowerExp", Y.family="Poisson"*)
  - predict for latent and response variables at given locations

2.3 Chain Handling and Diagnostics

- cutChain(*res, chain.ind, burnin, thining*)
- mixChain(*res.m.prl*)
  - mix parallel chains into one
- plot_acf(*S.mcmc*)
- findMode(*x*)
- functions from library {coda}

2.4 Generating Replicated Data

- repYeb(*N.sim, res.m, U, L, X=NULL, rho.family="rhoPowerExp", k=1*)
- repYpost(*res.m, L*)

2.5 Bayesian Model Checking

- BMCT(*Y.obs, Y.rep, funcT, ifplot=FALSE*)
- pRPS(*T.obs, T.rep*)
- plot_pRPS(*T.obs, T.rep, "t"*)

2.6 Transformed Residual Checking

- cdfU(*Y.obs,Y.rep, discrete=TRUE*)
- tranR(*Y.obs, Y.rep, discrete=TRUE*)
- plot_etran(*e.tran, fig=1:4*)
- e2dist(*e.tran*)
- baseline.dist(*n, iter*)
- plot_baseline(*res.in, dist.name*)
- pOne(*d.obs, d.base*)
- plot_pRPS(*d.obs, d.base, "d"*)

2.7 Other Functions

- loc2U(*loc*)
- locUloc(*loc, locp*)
- U2Z(*U, cov.par, rho.family = "rhoPowerExp"*)
- rhoPowerExp(*u, a, k*)
- rhoMatern(*u, a, k*)
- ...

3 To Do List

- develop more diagnostic tools, such as Rubin-Gelman diagnostics for parallel chains
- support for more link functions: probit, complementary log-log, reciprocal
- apply robust MCMC algorithm on other Hierarchical models
- support for non-constant variances
- employ other parallel computing techniques
- generalize to LM/GLM/GLMM
- ...