anainesvs Update README.md 4b6dbab Apr 22, 2016

2 contributors

124 lines (111 sloc) | 7.2 KB Raw Blame History

Integrating multiple Omics for Prediction of BC Survival

The following scripts illustrate how to fit some of the models presented in *Vazquez et al., Genetics, 2016*, the scripts are also provided at: https://github.com/anainesvs/VAZQUEZ_etal_GENETICS_2016, please refer to that webpage for updates.

Contact: avazquez@msu.edu

(1) Installing BGLR

The code below illustrates how to install and load the necessary package from CRAN using `install.packages()`.

```
install.packages(pkg='BGLR')     # install BGLR
```

```
library(BGLR);
```

(2) Loading data

Data: The code assumes that the user has saved in the file `OMIC_DATA.rda` the objects that contain the phenotypic information, clinical covariates, and omic data. The code assumes that the file `OMIC_DATA.rda` contain the following objects:

- `XF` : an incidence matrix for clinical covariates.
- `Xge` : an incidence matrix for gene expression.
- `Xmt` : an incidence matrix for methylation values at various sites.
- `y` : a vector with the response, in this case a 0/1 where 0 denotes alive. The code below assumes that all the predictors were edited by removing outliers and predictors that did not vary in the sample, transformed if needed, and missing values were imputed.

(3) Computing similarity matrices

Some of the models fitted in the study use similarity matrices of the form G=XX' computed from omics. The following code illustrates how to compute this matrix for gene expression. A similar code could be use to compute a G-matrix for methylation or other omics (see (6)).

```
load('OMIC_DATA.rda')
#Computing a similarity matrix for gene-expression data
 Xge<- scale(Xge, scale=true, center=TRUE) #centering and scaling
 Gge<-tcrossprod(Xge)                      #computing crossproductcts
 Gge<-Gge/mean(diag(Gge))                  #scales to an average diagonal value of 1.
```

NOTE: for larger data sets it may be more convinient to use the `geG()` function of the BGData R-package. This function allows computing G without loading all the data in RAM and offers methods for multi-core computing.

(4) Fitting a binary regression for (the "fixed effects" of) Clinical Coavariates using BGLR (COV)

The following code illustrates how to use BGLR to fit a fixed effects model. The matrix XF is an incidence matrix for clinical covariates. There is no column for intercept in XF because BGLR adds the intercept automatically. The response variable `y` is assumed to be coded with two lables (e.g., 0/1), the argument `response_type` is used to indicate to BGLR that the response is ordinal (the binary case is a special case with only two levels). Predictors are given to BGLR in the form a two-level list. The argument `save_at` can be used to provide a path and a pre-fix to be added to the files saved by BGLR. For further details see Pérez-Rodriguez and de los Campos, Genetics, 2014. The code also shows how to retrieve estimates of effects and of success probabilities. In the examples below we fit the model using the default number of iterations (1,500) and burn-in (500). In practice longer chains are needed, the user can increase the numbrer of iterations or the burn-in using the arguments `nIter` and `burnIn` of `BGLR`.

```
### Inputs
# centering and scaling the incidence matrix for fixed effects.
 XF<- scale(XF, scale=FALSE, center=TRUE)
 ETA.COV<-list( COV=list(X=XF, model='FIXED') )
# Fitting the model
 fm=BGLR(y=y, ETA=ETA.COV, saveAt='cov_', response_type='ordinal')
# Retrieving estimates
 fm$ETA$COV$b      # posterior means of fixed effects
 fm$ETA$COV$SD.b   # posteriro SD of fixed effects
 head(fm$probs)    # estimated probabilities for the 0/1 outcomes.
```

(5) Fitting a binary model for fixed effects and whole genome gene expression (GE) using BGLR (COV+GE)

The following code illustrates how to use BGLR to fit a mixed effects model that accomodates both clinical covariates and whole-genome-gene expression.

```
# Setting the linear predictor
```

```
  ETA.COV.GE<-list( COV=list(X=XF, model='FIXED'), GE=list(K=Gge, model='RKHS'))
# Fitting the model
  fm.COV.GE<- BGLR(y=y, ETA=ETA.COV.GE, response_type='ordinal',saveAt='cov_ge_')
#  Retrieving predictors
  fm.COV.GE$mu             # intercept
  fm.COV.GE$ETA$COV$b      # effects of covariates
  fm$COV.GE$ETA$GE$varU    # variance associated to GE SD.varU gives posterior SD
  fm.COV.GE$ETA$GE$u       # random effects associated to gene expression
  plot(scan('cov_ge_ETA_GE_varU.dat'),type='o',col=4) # trace plot of variance of GE.
```

NOTE: to fit a similar model for COV+METH one just needs to change the inputs in the defintiion of the linear predictor by providing Gmt instead of Gge.

(6) Fitting a binary model for fixed effects covariates and 2 omics (COV+GE+METH)

The following code shows how to extend the the model `COV+GE` with addition of methylation data.

```
#Computing a similarity matrix for methylation data
Xmt<- scale(Xmt, scale=TRUE, center=TRUE)  #centering and scaling
Gmt<-tcrossprod(Xmt)                       #computing crossproductcts
Gmt<-Gmt/mean(diag(Gmt))                   #scales to an average diagonal value of 1.
ETA.COV.GE.MT<-list( COV=list(X=XF, model='FIXED'),
                     GE=list(K=Gge, model='RKHS'),
                     METH=list(K=Gmt, model='RKHS'))
# Fitting models
fm.COV.GE.MT<- BGLR(y=y, ETA=ETA.COV.GE.MT,
                 response_type='ordinal',saveAt='cov_ge_mt_')
```

(7) Fitting a binary model for fixed effects covariates and 2 omics and their interactions (COV+GE+METH+GExMETH)

The following code shows how to extend the the model `COV+GE+METH` with addition of interactions between gene expression and methylation profiles.

```
 G.mg=Gmt*Gge
 G.mg=G.mg/mean(diag(G.mg))
 ETA.COV.GE.MT.GExMT<-list( COV=list(X=XF, model='FIXED'),
                      GE=list(K=Gge, model='RKHS'),
                      METH=list(K=Gmt, model='RKHS'),
                      GExMETH=list(K=G.mg, model='RKHS'))
# Fitting models
fm.COV.GE.MT.GExMT<- BGLR(y=y, ETA=ETA.COV.GE.MT.GExMT,
                  response_type='ordinal',saveAt='cov_ge_mt_gexmt')
```

(8) Validation

The following illustrates how to select a validation set using the model `COV` as example.

```
#Installing and loading library pROC to compute Area Under the ROC Curve.
install.packages(pkg='pROC')    # install pROC
library(pROC);
n <- length(y)
  # Randomly select a 20% of the data to be the testing set
tst<- runif(n) <0.2
yNA = y; yNA[tst] <-NA
  # Fit the model only in the training set
fm.COVtr<- BGLR(y=yNA, ETA=ETA.COV, response_type='ordinal')
  # Find probability of survival for the testing set
pred <-fm.COVtr$probs[tst,2]
  # Estimate AUC
AUC_train<-auc(y[!tst],fm.COVtr$yHat[!tst])
AUC_test<-auc(y[tst], pred)
#For the first individual, area under the standard normal curve (CDF)
```

```
#of estimated y from full model:
pnorm(fm.COVtr$yHat[1])
```

NOTE: if sample size is small (like TCGA data) and uneven in the number of 1s and 0s it will be wise to randomize 1s and 0s to be part of the testing sets, and repeate the validation multiple times. In Vazquez et al., 2016 (Genetics) we implement 200 cross-validations.