Integrating multiple omics from GBM data

Bernal Rubio et al., (2018)

08/28/2018

The following scripts illustrate how to implement the models presented in *Bernal Rubio et al., G3, 2018*, the scripts are also provided at: https://github.com/anainesvs/BERNAL_RUBIO_etal_G3_2018, please refer to that webpage for updates.

Contact: avazquez@msu.edu **Contact**: jlbernalr@gmail.com

(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

- `XF`: a matrix containing age at the moment of diagnosis (numeric, years), gender (0/1 is female, or not), race (0/1 is white, or not).
- `X.ge`: a matrix for gene expression.
- `X.meth`: a matrix for methylation values at various sites.
- `X.cnv`: a matrix of mean CNV intensity per gen.
- `y`: a matrix with 4 columns. The first column 'time' is the time to last follow up or event, columns 2 and 3 are variables `a` and `b` required by BGLR to analyze censored data, see description on this analysis on BGLR can be found on https://cran.r-project.org/web/packages/BGLR/vignettes/BGLR-extdoc.pdf. Finally, the column 4 ('status 4 months') is an indicator variable (0/1) with 0 indicating whether the patient is alive at the 5th month after diagnosis.
- `batch`: A vector containing gene expression batch by subject.
- `folds`: A vector containing fold numbers (1 and 2) for cross validations. The two folds are balanced so they have the same proportions of dead/alive individuals.

After downloading, files can be loaded into R with:

```
load('GBM_DATA.rda')
#Checking files present and dimensions
ls()
#[1] "X.ge"   "X.meth" "X.cnv"  "XF"     "y"
```

Clinical covariates, as well as omic datasets should be edited by removing outliers (for instance, observations outside of mean +- 3SD) and ordering samples id equally across datasets. Also, incidence matrix should be conformable in dimensions for subsequent analysis.

(3) Pre-adjust gene expression matrix by batch

For demostration, in this section we show how to pre-correct the incidence matrix corresponding to gene expression by batch effects using mixed models. The input used is the matrix "X.ge" centered, and the batch code for each sample. Using a linear mixed model, the pre-correction regresses each column of the Xge matrix into the corresponding batch effect, assuming them as random effects.

```r
p <- ncol(X.ge)
#Load library lme4 to fit a linear mixed model
library(lme4)
for(j in 1:p) {
  tmp <- !is.na(X.ge[, j])
  cat("Column =",j,"of",p,".\n")
  X.ge[tmp, j] <- residuals(lmer(X.ge[tmp, j] ~ 1 | batch[tmp]))
}
#Center and scaling X.ge columns.
X.ge <- scale(X.ge, scale=TRUE, center=FALSE)
```

(4) 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 SNP genotypes. A similar code could be use to compute a G-matrix for methylation or other omics.

```r
#Check:
#https://stackoverflow.com/questions/32675820/how-to-handle-missing-values-in-crossprod-in-r
tcrossprod.na <- function(x, val=0) tcrossprod(replace(x, is.na(x), val))
#Computing a similarity matrix for SNP genotypes
X.meth_sc<- scale(X.meth, scale=TRUE, center=TRUE) #centering and scaling
G.meth<-tcrossprod.na(X.meth_sc)  #computing crossproductcts
G.meth<-G.meth/mean(diag(G.meth))  #scales to an average diagonal value of 1.

#The same procedure was followed for other omics.
```

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

(5) Exploring demographic effects in the principal components (e.g., Methylation-derived PC vs age)

The following code shows how to search for effects of demographics such as age on the omic derived PCs, for instance from Methylation.

```r
#Gmeth has been obtained as described in previous section
EVD<-eigen(G.meth,symmetric=TRUE)
tmp <- EVD$values >  1e-04
pc.meth <- EVD$vectors[,tmp] %*% diag(sqrt(EVD$values[tmp]))
rownames(pc.meth) <- rownames(G.meth)
colnames(pc.meth) <- paste0("PC",1:ncol(pc.meth))
```

A similar approach can be implemented for additional omics (gene expression and methylation).

(6) Variance explained by omics

The following code illustrates how to use BGLR to fit a fixed effects model. The incidence matrix “XF” as well as the object “y” described in section (3) are used. There is no column for intercept in XF because BGLR adds the intercept automatically. 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 Perez-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 and therefore, the user can increase the number of iterations or the burn-in using the arguments `nIter` and `burnIn` of `BGLR`.

```
### Inputs: XF, pc.meth and y
ETA.cov_meth<-list(cov = list(X = scale(XF), model = 'FIXED'),
                   meth = list(X = pc.meth, model = "BRR"))
# Fitting the model
fm.meth<-BGLR(y=y[,"time"],a=y[,"a"],b=y[,"b"],ETA=ETA.cov_meth,saveAt="Example_")
# Retrieving posterior variances
var.meth <- fm.meth$ETA$meth$varB
var.error <- fm.meth$varE
```

```
### Inputs: XF, G.meth, G.ge and y
ETA.cov_meth_ge <- ETA.cov_meth
ETA.cov_meth_ge$ge <- list(X = pc.ge, model = "BRR")
# Fitting the model
fm.meth_ge<-BGLR(y=y[,"time"],a=y[,"a"],b=y[,"b"],ETA=ETA.cov_meth_ge,saveAt="Example2_")
# Retrieving posterior variances
var.meth <- fm.meth_ge$ETA$meth$varB
var.ge <- fm.meth_ge$ETA$ge$varB
var.error <- fm.meth_ge$varE
```

(7) Evaluating prediction accuracy.

The following code illustrates how to use BGLR to calculate prediction accuracy of omic models. Note that in Bernal-Rubio et al (2018), AUC was calculated by estimating the proportion of deaths at different time points, equivalent to every month after diagnosis (e.g. month 1 after diagnosis, month 2 after diagnosis, …). Below is an example for month 5 after the diagnosis (vital status available in y[, 4]).

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

```
#Installing and loading library pROC to compute Area Under the ROC Curve.
install.packages(pkg='pROC')    # install pROC
library(pROC);
n <- nrow(y)
status.5months <- y[, "status 5 months"]
# Setting training and testing sets
trn <- (1:n)[folds == 1]
tst <- -trn
yNA <- y[, "time"]
yNA[tst] <- NA
# Fit the model only in the training set
fm.meth <- BGLR(y=yNA,a=y[,"a"],b=y[,"b"],ETA=ETA.cov_meth,saveAt="Example3_")
# Find probability of survival for the testing set
pred <-fm.meth$probs[tst,2]
# Estimate AUC
AUC <- auc(status.5months[tst], pred)
#For the first individual, area under the standard normal curve (CDF)
#of estimated y from full model:
pnorm(fm.meth$yHat[1])
```