Using the `robHD` Package

Hao Chai; Shuangge Ma; Qingzhao Zhang

July 12, 2016

1 Introduction

We provide a brief introduction to the function `robHD`, which computes the solution to high-dimensional accelerated failure time models using robust loss functions. The robustness is achieved by using a exponential squared function, which is defined as follows.

$$\sum_{i=1}^{n} \omega_i \exp(-(T_i - X_i'\beta)^2), \qquad i = 1, \cdots, n, \tag{1}$$

Here $\beta \in \mathcal{R}^d$ represent the regression coefficients. ω_i, T_i, and X_i are Kaplan-Meier weight, logarithm of the event time, and covariates for subject i, respectively.

The overall objective function is defined as the exponential squared function minus the penalty term:

$$\sum_{i=1}^{n} \omega_i \exp(-(T_i - X_i'\beta)^2) - \sum_{j=1}^{d} r_{nj}\rho(|\beta_j|; \lambda; \kappa), \qquad i = 1, \cdots, n. \tag{2}$$

Here $\rho(|\beta_j|; \lambda; \kappa)$ is defined as $\lambda|\beta_j|$ for Lasso penalty and $\lambda \int_0^{|t|}(1 - \frac{\kappa x}{\lambda})^+ dx$ for MCP penalty. We would like to minimize the overall loss function.

The `robHD` package consists of one function.

`robHD` A function to compute the regression coefficients for robust penalized regression. The exponential squared loss function is used to provide the robustness, and penalization is employed to enforce the sparsity of estimators. It searches the solution over the $\lambda - \kappa$ grid. It uses Lasso solution ($\kappa = 0$) as initial values for MM-coordinate descent algorithm and computes the MCP solutions if MCP penalty is specified.

2 Examples

2.1 A basic example

A simple example illustrates how to use `robHD` to fit the accelerated failure time models. x is a 100×200 covariate matrix. b is a vector of length 200. Only

the first five elements of b are non-zero. y is the logarithm of the event time, and *delta* is the event indicator. *res* is an object of class "rob", which consists of seven components. See function `robHD` for details of these components.

```
> library(robHD)
> x = matrix(rnorm(20000), nrow = 100)
> b = c(0.5, 1, -1, 2, -0.5, rep(0, 195))
> y = crossprod(t(x), b) + rnorm(100, 0, 1)
> delta = rbinom(100, 1, .7)
> res = robHD(x, y, delta, theta = c(1, 10, 100))
> str(res)

List of 7
 $ betalasso: num [1:200, 1:60] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:200] "las b1" "las b2" "las b3" "las b4" ...
  .. ..$ : NULL
 $ betamcp  : num [1:200, 1:60] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:200] "mcp b1" "mcp b2" "mcp b3" "mcp b4" ...
  .. ..$ : NULL
 $ iter     : int [1:60] 1 1 1 1 1 1 1 1 1 ...
 $ lambda   : num [1:20] 1.78 1.61 1.44 1.28 1.13 ...
 $ kappa    : num 0.333
 $ theta    : num [1:3] 1 10 100
 $ omega    : num [1:100] 0 0 0.0102 0.0102 0.0102 ...
 - attr(*, "class")= chr "rob"
```

2.2 Marginal analysis of gene-environment interactions

In this example, we illustrate how to use `robHD` to analyze the gene-environment interactions using marginal models. We consider a data set with sample size 200. Environmental variables are represented as x and have $n.beta0 = 5$ component. The genetic variables are represented as z and have $P = 200$ components. xz represents the interaction terms. $beta0$, $theta0$, and $tau0$ are the true coefficients associated with x, z, and xz, respectively. t is the failure time, and c is the censoring time. y and *delta* are defined in the previous example. In each marginal model, there are $2 * n.beta0 + 1$ variables. Results are kept in *res* for each marginal model.

```
> n.beta0 = 5
> P = 200
> N = 200
> z = matrix(rnorm(N * (P + n.beta0)), nrow = N)
> x = z[, 1:n.beta0]
> z = z[, (n.beta0 + 1):(n.beta0 + P)]
> xz = matrix(apply(z, 2, "*", x), nrow = N)
```

```
> beta0 = rep(1, n.beta0)
> theta0 = c(rep(1, 5), rep(0, P - 5))
> tau0 = rep(0, n.beta0 * P)
> tau0[c(sort(sample(1:(n.beta0 * 5))[1:20]))] =
+   round(runif(20, 0.9, 1.1), 1)
> mu = (as.vector(crossprod(t(x), beta0) + crossprod(t(z), theta0) +
+                   crossprod(t(xz), tau0)))
> e1 = rnorm(N, 0, 1)
> e2 = rcauchy(N)
> e2 = ifelse(as.logical(rbinom(N, 1, 0.2)), e2, 0)
> t = e1 + mu
> c = log(rweibull(N, shape = 0.9, scale = exp(mu) * 6))
> delta = (t < c)
> y = (ifelse(t < c, t, c)) + e2
> res = robHD(cbind(x, z, xz), y, delta, theta = c(1, 10, 50))
> for (j in 1:P)
+ {
+   res = robHD(cbind(x, z[, j], xz[, (n.beta0 * (j - 1) + 1):(n.beta0 * j)]),
+               y, delta, theta = c(1, 10, 50))
+ }
> str(res)

List of 7
 $ betalasso: num [1:11, 1:60] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:11] "las b1" "las b2" "las b3" "las b4" ...
  .. ..$ : NULL
 $ betamcp  : num [1:11, 1:60] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:11] "mcp b1" "mcp b2" "mcp b3" "mcp b4" ...
  .. ..$ : NULL
 $ iter     : int [1:60] 1 1 1 1 1 1 1 1 1 ...
 $ lambda   : num [1:20] 2.04 1.88 1.72 1.57 1.43 ...
 $ kappa    : num 0.333
 $ theta    : num [1:3] 1 10 50
 $ omega    : num [1:200] 0.005 0.005 0 0.00503 0.00503 ...
 - attr(*, "class")= chr "rob"
```

2.3 Joint analysis of gene-environment interactions

In this example, we illustrate how to use `robHD` to analyze the gene-environment interactions using joint models. The setting is similar to the ones in the previous example, except that in joint model, there are $n.beta0+P+n.beta0*P$ variables. It is considerably larger comparing to the marginal models. Results are kept in *res* for the joint model analysis.

```
> n.beta0 = 5
> P = 200
> N = 200
> Z = matrix(rnorm(N * (P + n.beta0)), nrow = N)
> X = Z[, 1:n.beta0]
> Z = Z[, (n.beta0 + 1):(n.beta0 + P)]
> XZ = matrix(apply(Z, 2, "*", X), nrow = N)
> beta0 = rep(1, n.beta0)
> theta0 = c(rep(1, 5), rep(0, P - 5))
> tau0 = rep(0, n.beta0 * P)
> tau0[c(sort(sample(1:(n.beta0 * 5))[1:20]))] =
+   round(runif(20, 0.9, 1.1), 1)
> mu = (as.vector(crossprod(t(X), beta0) + crossprod(t(Z), theta0) +
+                   crossprod(t(XZ), tau0)))
> e1 = rnorm(N, 0, 1)
> e2 = rcauchy(N)
> e2 = ifelse(as.logical(rbinom(N, 1, 0.2)), e2, 0)
> t = e1 + mu
> c = log(rweibull(N, shape = 0.9, scale = exp(mu) * 6))
> delta = (t < c)
> y = (ifelse(t < c, t, c)) + e2
> res = robHD(cbind(X, Z, XZ), y, delta, theta = c(1, 10, 50))
```