An illustration

Required packages

The package “gurobi” requires installing Gurobi in local computer, and obtain a license from its website.

```
require(Matrix)
```

```
## Loading required package: Matrix
```

```
require(gurobi)
```

```
## Loading required package: gurobi
## Loading required package: slam
```

A function to generate optimal designs

This function generates optimal design with and without network connections. If no network adjacency matrix W is loaded in the function, the function will give optimal design with respect to covariates balance only. The default settings are $\alpha = 0.001$, $\rho = 0.5$, and the timelimit to solve an optimization problem in Gurobi is 500 secs.

```
opt.design <- function(Z, W=NA, alpha=0.001, rho=0.5, timelimit=500) {
  #################################################################
  # Function for optimal design with and without network
  # Inputs:
  # Z: covariate matrix, intercept not included
  # W: network adjacency matrix (NOT USED FOR network==F)
  # alpha: quantile parameter (NOT USED FOR network==F)
  # rho: network correlation parameter  (NOT USED FOR network==F)
  # timelimit (in sec): timelime in solving the optimization problem
  #
  # Outputs:
  # optimal allocation (-1, 1)
  ################################################################
  require(gurobi)
  require(Matrix)

  # common statistics
  Fmat <- cbind(1, Z)


  # setup global parameters
  params <- list(MIPGap=0.01, TimeLimit=timelimit)


  # setup optimization model
  model <- list()
```

```r
  # setup linear constraint
  model$A     <- matrix(1, 1, n)
  model$rhs   <- round(n/2) # to compile with the case that n is odd
  model$sense <- c("=")

  # set up decision variable type
  model$vtype <- rep('B', n)

  # with or without network

  network <- !is.na(sum(W))

  if(network == TRUE) {
    # network statistics
    n <- nrow(W)
    nedge <- rowSums(W)

    # set up quadratic objective
    Sigma.net <- Diagonal(n, x = nedge)-rho*W
    Fsig <- t(Fmat) %*% Sigma.net
    Sigma <- Fsig %*% Fmat
    Q <- as.matrix(t(Fsig) %*% solve(Sigma, Fsig))
    model$Q     <- Q
    model$obj   <- -rowSums(Q)

    # set up quadratic constraint
    model$quadcon <- list()
    qc <- list()
    qc$Qc <- 4*W
    qc$q <- -4*rowSums(W)
    qc$rhs <- sqrt(sum(nedge))*qnorm(alpha)-sum(W)
    model$quadcon[[1]] <- qc

  }

  if(network == FALSE) {
    # set up quadratic objective
    Finv <- solve(t(Fmat) %*% Fmat, t(Fmat))
    Q <- as.matrix(Fmat %*% Finv)
    model$Q     <- Q
    model$obj   <- -rowSums(Q)
  }

  # solve the problem with gurobi
  result <-gurobi(model, params)
  return(2*result$x-1)
}
```

Generate Data

```r
n <- 100
p <- 10
```

```r
net.p <- 0.08 # network density

# generate the covariates
Z <- matrix(sample(c(-1, 1), n*p, replace=TRUE), n, p)

# generate the network
W <- sample(c(0, 1), n*n, replace=TRUE, prob=c(1-net.p/2,net.p/2))
W <- matrix(W, n, n)
W <- 1*((W+t(W))>0)
diag(W) <- 0
```

Generate Optimal Designs

```r
# optimal design with network
x.opt <- opt.design(Z, W)
```

```
## Gurobi Optimizer version 9.0.0 build v9.0.0rc2 (mac64)
## Optimize a model with 1 rows, 100 columns and 100 nonzeros
## Model fingerprint: 0x5629f885
## Model has 5050 quadratic objective terms
## Model has 1 quadratic constraint
## Variable types: 0 continuous, 100 integer (100 binary)
## Coefficient statistics:
##   Matrix range     [1e+00, 1e+00]
##   QMatrix range    [8e+00, 8e+00]
##   QLMatrix range   [1e+01, 6e+01]
##   Objective range  [1e+00, 8e+00]
##   QObjective range [2e-04, 6e+00]
##   Bounds range     [0e+00, 0e+00]
##   RHS range        [5e+01, 5e+01]
##   QRHS range       [9e+02, 9e+02]
## Presolve time: 0.00s
## Presolved: 1 rows, 100 columns, 100 nonzeros
## Presolved model has 5050 quadratic objective terms
## Presolved model has 1 quadratic constraint(s)
## Variable types: 0 continuous, 100 integer (100 binary)
##
## Root relaxation: objective -9.775000e+01, 74 iterations, 0.00 seconds
##
##     Nodes    |    Current Node    |     Objective Bounds      |     Work
##  Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
##
##      0     0  -97.75000    0   12          -  -97.75000      -     -    0s
##      0     0  -97.75000    0   13          -  -97.75000      -     -    0s
##      0     0  -97.75000    0   14          -  -97.75000      -     -    0s
##      0     0  -97.75000    0   14          -  -97.75000      -     -    0s
##      0     2  -97.75000    0   14          -  -97.75000      -     -    0s
## H    9    12                     -97.1879080  -97.75000  0.58%   7.6    0s
##
## Explored 11 nodes (172 simplex iterations) in 0.03 seconds
## Thread count was 4 (of 4 available processors)
##
```

```
## Solution count 1: -97.1879
##
## Optimal solution found (tolerance 1.00e-02)
## Best objective -9.718790795899e+01, best bound -9.775000000000e+01, gap 0.5784%
```

```
# optimal design without network
x.opt0 <- opt.design(Z)
```

```
## Gurobi Optimizer version 9.0.0 build v9.0.0rc2 (mac64)
## Optimize a model with 1 rows, 100 columns and 100 nonzeros
## Model fingerprint: 0x8c843a36
## Model has 5050 quadratic objective terms
## Variable types: 0 continuous, 100 integer (100 binary)
## Coefficient statistics:
##   Matrix range     [1e+00, 1e+00]
##   Objective range  [1e+00, 1e+00]
##   QObjective range [5e-05, 5e-01]
##   Bounds range     [0e+00, 0e+00]
##   RHS range        [5e+01, 5e+01]
## Found heuristic solution: objective -21.2337611
## Presolve time: 0.00s
## Presolved: 1 rows, 100 columns, 100 nonzeros
## Presolved model has 5050 quadratic objective terms
## Variable types: 0 continuous, 100 integer (100 binary)
##
## Root relaxation: objective -2.500000e+01, 58 iterations, 0.00 seconds
##
##     Nodes    |    Current Node    |     Objective Bounds      |     Work
##  Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
##
##      0     0  -25.00000    0    8  -21.23376  -25.00000  17.7%     -    0s
## H    0     0                      -24.9291484  -25.00000  0.28%     -    0s
##
## Explored 1 nodes (58 simplex iterations) in 0.00 seconds
## Thread count was 4 (of 4 available processors)
##
## Solution count 2: -24.9291 -21.2338
##
## Optimal solution found (tolerance 1.00e-02)
## Best objective -2.492914840421e+01, best bound -2.500000000000e+01, gap 0.2842%
```

Evaluation Function

A function to compute the key statistics given a design. The value ρ is the true correlation coefficient. The outputs are explained as follows:

- Impre: the improvement to precision in equation 17
- obj: the objective value in equation 9 of the given design
- Erand: expected objective value of the random design
- T1: T1 of the given design
- T2: T2 of the given design
- ET1: expected T1 value of the random design
- ET2: expected T2 value of the random design
- 1*(flag==0): a flag demonstrate an appropriate design is loaded.

```r
evaluate <- function(x, W, Z, rho) {
  ###################################################################
  # Function to compute evaluation criterion for a design under car model assumption
  # Inputs:
  # x: design vector allocation (-1, 1)
  # W: network adjacency matrix
  # Z: covariates
  # rho: network correlation parameter
  #
  # Outputs:
  # Precentage of Improvement to Precision and others
  ##################################################################

  # compute statistics
  n <- nrow(W)
    nedge <- rowSums(W)
  nedge.total <- sum(W)
  p <- ncol(Z)
  Erand <- -999
  T1 <- -999
  T2 <- -999
  ImPre <- -999
  obj <- -999
  flag <- length(x)

  if(flag!=0) {
    Rmat <- Diagonal(n, x =nedge)-rho*W
    Fmat <- cbind(1, Z)
      Fsig <- t(Fmat) %*% Rmat
      xFsig <- Fsig %*% x
    Sigma <- Fsig %*% Fmat

    # compute the objective
    T1 <- rho * as.numeric(t(x)%*% W %*% x)
    T2 <- as.numeric(t(xFsig) %*% solve(Sigma, xFsig))
    obj <- nedge.total-T1-T2

    # compute the expected objective of random designs
    KF <- t(Fsig) %*% solve(Sigma, Fsig)
    K <- Rmat - KF
    Erand <- sum(diag(K))
    ET1 <- sum(diag(W))
    ET2 <- sum(diag(KF))

    # compute improvementment of precision
    ImPre <- 1-Erand/obj
  }
  return(c(rho, ImPre, obj, Erand, T1, T2, ET1, ET2, 1*(flag==0)))
}
```

Evaluation of the optimal designs

```
rho_true <- 0.7
results <- evaluate(x.opt, W, Z, rho=rho_true)
results <- rbind(results, evaluate(x.opt0, W, Z, rho=rho_true))
results <- data.frame(results)
names(results) <- c("rho_true", "ImprovePrecision", "Obj", "E(obj)", "T1", "T2", "ET1", "ET2", "error")
results$network <- c(1, 0)
results
```

```
##         rho_true ImprovePrecision      Obj   E(obj)    T1        T2 ET1
## results      0.7       0.18424391 845.5905 689.7956 -65.8  2.209496   0
##              0.7       0.08907473 757.2472 689.7956  12.6 12.152809   0
##              ET2 error network
## results 92.20439     0       1
##         92.20439     0       0
```

Connection of submitted files to numerical sections

- Section 6: "synthetic.R" is the main file to generate all the results and figures.
- Section 7: "real.R" is the main file to generate the results in Figure 7, and draw the figure; "robust_re_real.R" is the main file to generates the results in Figure 8, and "report_results_robust.R" generates Figure 8.
- Supplement: "gap.R" is the main file to generate figure S1.