Spatial Smoothing

Su Chen

November 8, 2016

```
## Warning: package 'microbenchmark' was built under R version 3.3.2
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

direct solver

```
direct_sparse_solver = function(lambda, D, y)
{
  require(Matrix)
  n = length(y)
  I = .sparseDiagonal(n)
  C = lambda* crossprod(D) + I
  x = Matrix::solve(C, y)
  return (x)
}
```

conjugate gradient method

```
conjugate_gradient = function(lambda, D, y, x0, conv=1e-10)
{
  require(Matrix)
  n = length(y)
  I = .sparseDiagonal(n)
  A = lambda* crossprod(D) + I

  r = A %*% x0 - y
  x = x0
  p = -r
  k = 0

  while( sum(abs(r)) >= conv )
  {
    r_square = crossprod(r)
    Ap = A %*% p
    alpha = as.numeric(r_square / crossprod(p, Ap))
    x_update = x + alpha * p
    r_update = r + alpha * Ap
    beta = as.numeric(crossprod(r_update) / r_square)
```

```
    p_update = -r_update + beta * p
    x = x_update
    r = r_update
    p = p_update
    k = k+1
  }

  return (x)
}
```

function to calculate soft threshold and L2_norm

graph fused lasso

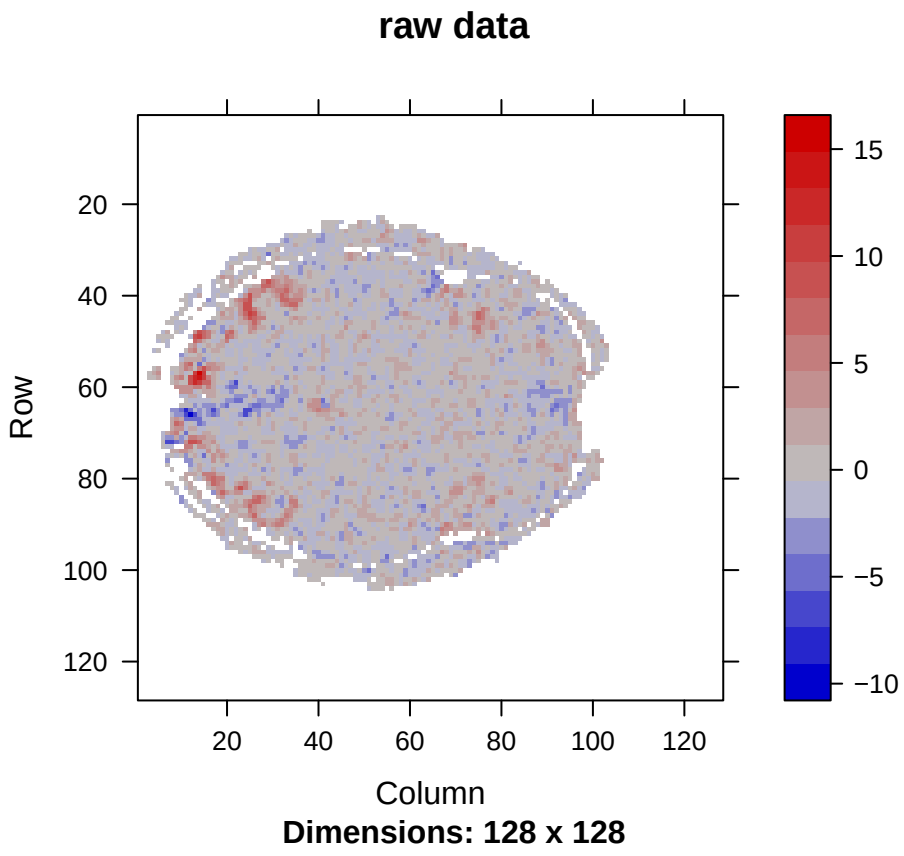
```
graph_fused_lasso = function(lambda, step, D, y, x0, iter_max=10000,
                             tol_abs=1e-10, tol_rel=1e-10)
{
  require(Matrix)
  n = length(y)
  m = nrow(D)
  I = .sparseDiagonal(n)
  A = lambda* crossprod(D) + I
  #U = Cholesky(A)

  x = x0
  r = rep(0,m)
  z = rep(0,n)
  s = rep(0,m)
  u = rep(0,n)
  t = rep(0,m)

  iter = 1
  err_pr = 1
  err_du = 1
  conv_pr = 0
  conv_du = 0

  while( ((err_pr >= conv_pr) || (err_du >= conv_du)) && (iter <= iter_max))
  {
    x_update = (y + step*(z - u)) / (1 + step)
    r_update = soft_threshold(y=s-t, l=lambda/step)
    w = x_update + u
    v = r_update + t
    b = w + crossprod(D, v)
    z_update = Matrix::solve(A, b)
    s_update = D %*% z_update
    u_update = u + x_update - z_update
    t_update = t + r_update - s_update

    delta_pr = c(as.vector(x_update-z_update), as.vector(r_update-s_update))
    delta_du = c(as.vector(step*(z_update-z)), as.vector(step*(s_update-s)))
    err_pr = L2_norm(delta_pr)
```

```
    err_du = L2_norm(delta_du)
    conv_pr = sqrt(n)*tol_abs + tol_rel* max(L2_norm(x_update),
                                             L2_norm(z_update))
    conv_du = sqrt(n)*tol_abs + tol_rel*step*L2_norm(u_update)

    if(err_pr >= 5*err_du) {
      step = step*2
    }
    if(err_du >= 5*err_pr) {
      step = step*0.5
    }

    x = x_update
    r = r_update
    z = z_update
    s = s_update
    u = u_update
    t = t_update

    iter = iter + 1
  }

  return(x)
}
```

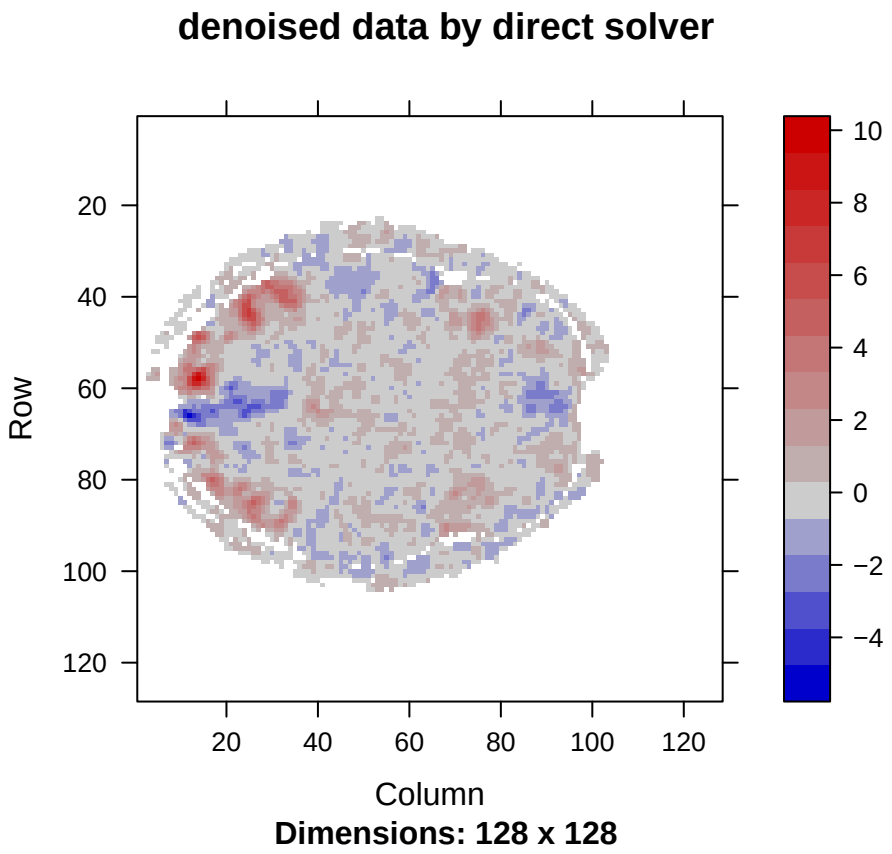
data

```
fmidata = read.csv("C:/Users/schen/Dropbox/toChensu/Stats/2016Fall/Big Data/Assignment8/fmri_z.csv", hea
fmidata = as.matrix(fmidata)
numofrow = nrow(fmidata)
numofcol = ncol(fmidata)
y_vector = as.vector(fmidata)

### generate the oriented edge matrix D ###
D = makeD2_sparse(dim1=numofrow, dim2=numofcol)

### run and compare results ###
result1 = direct_sparse_solver(lambda=1, D=D, y=y_vector)
denoised1 = matrix(result1, nrow=numofrow, ncol=numofcol)
denoised1[fmidata==0] = 0


result2 = conjugate_gradient(lambda=1, D=D, y=y_vector,
                             x0=y_vector, conv=1e-5)
denoised2 = matrix(result2, nrow=numofrow, ncol=numofcol)
denoised2[fmidata==0] = 0


result_gfl = graph_fused_lasso(lambda=1, step=2, D=D, y=y_vector,
                               x0=y_vector, iter_max=10000,
                             tol_abs=1e-5, tol_rel=1e-5)
denoised_gfl = matrix(result_gfl, nrow=numofrow, ncol=numofcol)
```

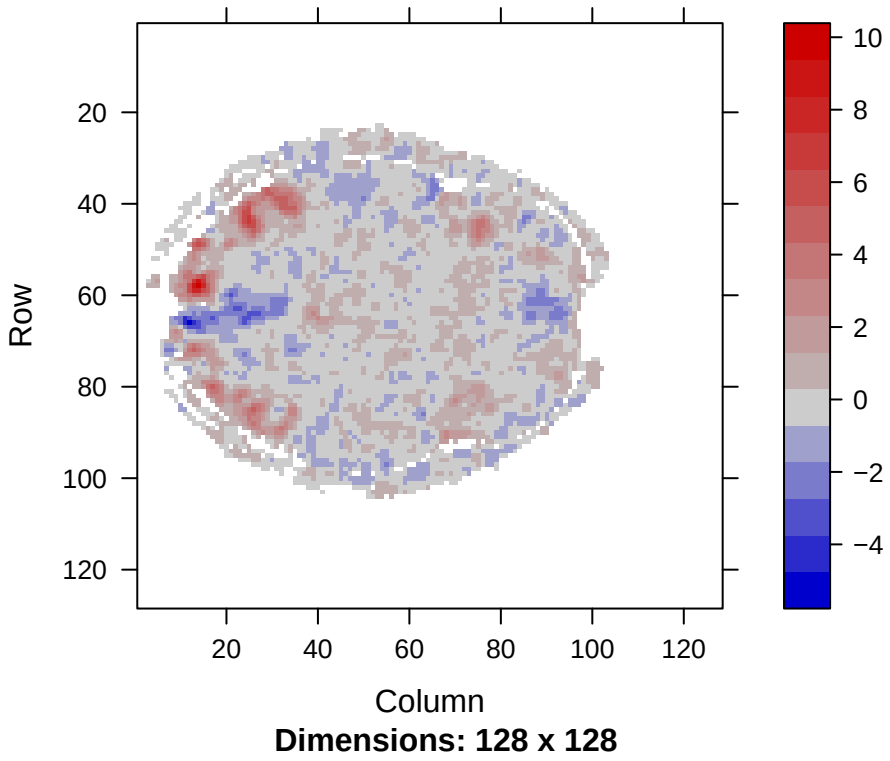
```
denoised_gfl[fmidata==0] = 0


### plot raw and denoised data side by side ###

par(mfrow=c(2,2))
image(Matrix(fmidata), useRaster=TRUE, main = "raw data")
```



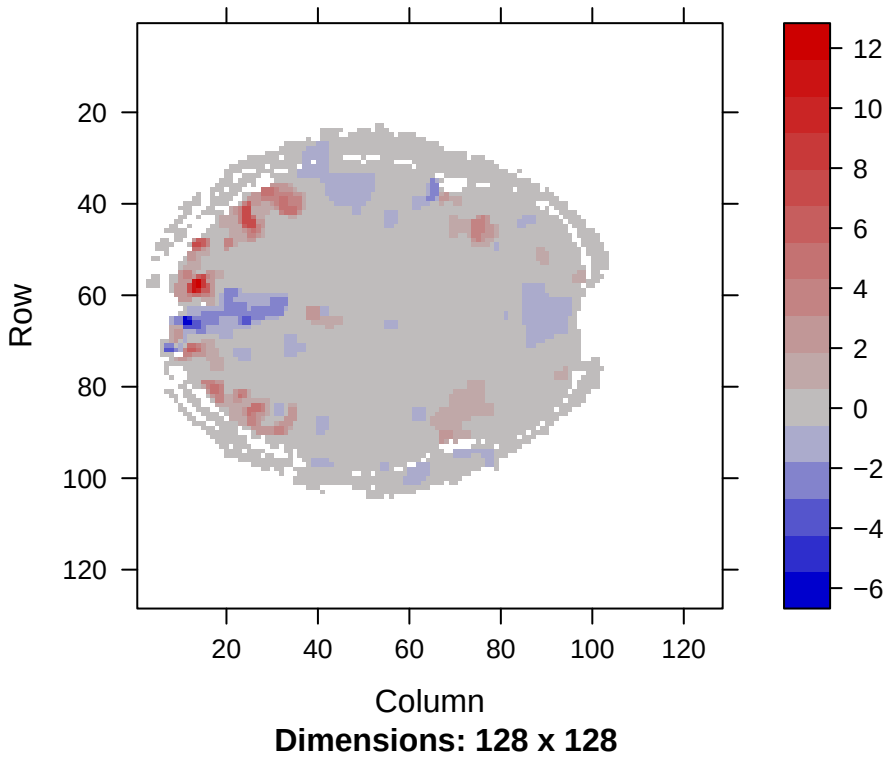

```
image(Matrix(denoised1), useRaster=TRUE, main = "denoised data by direct solver")
```




```
image(Matrix(denoised2), useRaster=TRUE, main = "denoised data by conjugate gradient")
```




```
image(Matrix(denoised_gfl), useRaster=TRUE, main = "denoised data by graph fused lasso")
```


denoised data by graph fused lasso
Row
20
40
60
80
100
120
Column
20
40
60
80
100
120
12
10
8
6
4
2
0
−2
−4
−6
Dimensions: 128 x 128