```
#-- Simplex-based simulation of spheres in 3-space
#
# X is 3xN where each column is (x,y,z) of a single ball
# r is a vector containing the radii of the balls above
# returns - a list of positions after compaction
compactSpheres = function(X,r,Xmax=1,Ymax=1,Zmax=1) {
# Constrains
  require(glpk)
  Xmin=Ymin=Zmin=0
  N = length(r)
  Nc2 = choose(N,2)

  prob = lpx_create_prob()
  lpx_add_cols(prob,3*N)
  for (i in 0:(N-1)) {
    lpx_set_col_name(prob,i*3 + 1, paste("x",i+1,sep=""))
    lpx_set_col_name(prob,i*3 + 2, paste("y",i+1,sep=""))
    lpx_set_col_name(prob,i*3 + 3, paste("z",i+1,sep=""))
  }

# (i) box constraints x+r <= Xmax (x - r) >= Xmin
  for (i in 1:N) {
    crow = lpx_get_num_rows(prob)
    lpx_add_rows(prob,6)

    lpx_set_mat_row(prob,crow + 1,1,3*(i-1) + 1,1)
    lpx_set_mat_row(prob,crow + 2,1,3*(i-1) + 2,1)
    lpx_set_mat_row(prob,crow + 3,1,3*(i-1) + 3,1)
    lpx_set_row_bnds(prob,crow+1,LPX_DB,Xmin+r[i],Xmax-r[i])
    lpx_set_row_bnds(prob,crow+2,LPX_DB,Xmin+r[i],Xmax-r[i])
    lpx_set_row_bnds(prob,crow+3,LPX_DB,Xmin+r[i],Xmax-r[i])
  }
# (ii) collision constraints
  for (i in 1:N) {
    if (i < N)
      for (j in (i+1):N) {
        pi = X[,i]
        pj = X[,j]
        u = (pj - pi) /  sqrt( (pj - pi) %*% (pj - pi) )
        a = matrix(0,nrow(X),ncol(X))
        a[,j] = u
        a[,i] = -u
        a = as.vector(a)
        crow = lpx_get_num_rows(prob)
        lpx_add_rows(prob,1)
        nz = which(a!=0)
        lpx_set_mat_row(prob,crow + 1, length(nz),nz, -a[nz])
        lpx_set_row_bnds(prob,crow+1, LPX_UP, 0,-(r[i] + r[j]))
      }
  }

# (iii) distance constraints
#

  lpx_add_cols(prob,Nc2)
  ij = 1
  for (i in 1:N)
    if (i < N)
      for (j in (i+1):N) {
        pi = X[,i]
        pj = X[,j]
        u = (pj - pi) /  sqrt( (pj - pi) %*% (pj - pi) )
        U = cbind(u, c(0,0,1),c(0,0,-1),c(0,1,0),c(0,-1,0),c(1,0,0),c(-
1,0,0))
        for (k in 1:ncol(U)) {
          u = U[,k]
          a = matrix(0,nrow(X),ncol(X))
          a[,j] = u
          a[,i] = -u
          D = rep(0,Nc2)
          D[ij] = -1
          a = c(as.vector(a),D)
          crow = lpx_get_num_rows(prob)
          lpx_add_rows(prob,1)
          nz = which(a!=0)
          lpx_set_mat_row(prob,crow + 1, length(nz),nz, a[nz])
          lpx_set_row_bnds(prob,crow+1, LPX_UP, 0,0)
        }

        ij = ij + 1
      }

  for (i in 1:Nc2)
    lpx_set_obj_coef(prob,3*N+i,1)

  for (i in 1: lpx_get_num_cols(prob))
    lpx_set_col_bnds(prob,i,LPX_FR,0,0)

  lpx_simplex(prob)

  X = matrix(0,ncol=N, nrow=3)
  for (i in 1:(3*N))
    X[i]=lpx_get_col_prim(prob,i)
  list(positions=X,status=lpx_get_status(prob))
}



# Generate balls for a given vector of radii,
# randomly position the balls in space, and then compact
# them using compactSphers function
#
# r - vector of radii
genBalls = function(r, scale=10, iterations=4) {
  N = length(r)
  V = sum(4/3* pi * r ^ 3)
  cellSize= (V * scale)^(1/3)
  X = rbind(runif(N), runif(N),runif(N)) * cellSize

  Y = X
  for ( i in 1:iterations) {
    sol = compactSpheres(Y,r,Xmax=cellSize,Ymax=cellSize,Zmax=cellSize)
    Y = sol$pos
  }
  list(orig=X,compact=Y)
}


# Take a slice of a virtual cell
#
```

```
# balls - ball locations
# r - ball radii
# z - location (in z dimension) of slice
# n * dt - slice thikness (we take n infinitesmal slices at distance dt
apart)
# RES - resolution of the screen
slice2 = function(balls,r, z, n=7, dt=10, RES=200) {
  screen = ns(RES)
  xmin=min(balls[1,]-r)
  ymin=min(balls[2,]-r)
  xmax=max(balls[1,]+r)
  ymax=max(balls[2,]+r)
  unit = max(xmax-xmin,ymax-ymin)

  Y =  matrix(rep(1:RES,RES),RES)
  X =  matrix(rep(1:RES,each=RES),RES)
  for (i in 1:n) {
    for (j in 1:length(r)) {
      zoffset = abs(balls[3,j] - z)
      if (zoffset < r[j]) {
                                         # in this case, the ball is in
the slice
        rr = sqrt(r[j]^2-zoffset^2) / unit * nrow(screen)
        x= (balls[1,j] - xmin) / unit * nrow(screen)
        y= (balls[2,j] - ymin) / unit * nrow(screen)

        screen[(X-x)^2 + (Y-y)^2 < rr^2] = j

      }
    }
    z = z + dt
  }
  list(unit=unit,screen=screen)
}


ns = function(r) {
  x = matrix(0,r*r)
  dim(x) = c(r,r)
  rownames(x) = 1:r
  colnames(x) = 1:r
  x
}

# Caclulated areas from the output of the slice2 function
# screen - screen calculated by slice2
areas = function (screen) {
  s = screen$screen
  s = factor(s)
  x = (xtabs(~ s, s)[-1])
  x * (screen$unit/nrow(screen$screen))^2
}

# 20 cells, 40 vesicles each, log-normal distribution with inputed mu &
sigma
# Use function genBalls, compactSpheres, slice2, areas to generate
section areas
resim = function (mu,sigma){
  A = NULL
  for (cell in  1:20) {
    r=rlnorm(40,mean=mu,sd=sigma);
    ret=genBalls(r,scale=20,iterations=3)
    s = slice2(ret$compact, r, z = mean(ret$compact[3,])+20, RES=300, n=
7,dt=10)
    a = areas(s)
    A = c(A,a)
  }
  A
}
```