```
function results_density, x, y, z, output, regions, tree, volume=volume, points=points

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Version History
;;   4.5: 4/21/2011
;;     * First version that works with parallelized kd_tree nearest neighbor
;;       search
;;   4.4: 4/20/2011
;;     * Same as 4.5 but still has the debug code in it -- use this when
;;       writing up the comparisons
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants
common results

if (n_elements(x) NE n_elements(y)) then stop
if (n_elements(x) NE n_elements(z)) then stop
npts = n_elements(x)

volume = dblarr(npts)
density = dblarr(npts)
points = replicate(-1L, npts)

;; Enforce density outside modeled region or inside planet = 0
r = sqrt(x^2 + y^2 + z^2)
if (input.options.fullsystem) $
  then nonzero = where(r GT 1, num, comp=zero) $
  else nonzero = where((r GT 1) and (r LT input.options.outeredge), num, comp=zero)

if (num GT 0) then begin
  x2 = float(x[nonzero])
  y2 = float(y[nonzero])
  z2 = float(z[nonzero])

  ;; Determine closest packet to each point
  outpts = ptr_new([[*output.x], [*output.y], [*output.z]])
  results_find_closest, outpts, tree, [[x2], [y2], [z2]], pmin=pt
  outpts = 0

  ;; Determine the volume for each of the needed regions
  results_voronoi_volume, regions, pt
  volume2 = (*regions.volume)[pt]
  q = where(volume2 EQ 0, nq) & if (nq NE 0) then stop

  density2 = (*output.frac)[pt]/volume2/(SystemConsts.rplan*1e5)^3 ;; volume = cm^3
  q = where(volume2 GT 1e10, nq)
  if (nq NE 0) then density2[q] = 0.

  volume[nonzero] = volume2
  density[nonzero] = density2
  points[nonzero] = pt
endif

q = where(density LT 0, nq) & if (nq NE 0) then stop
return, density

end
```