```
function produce_density, files, data, savefile=savefile

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; determine density at points data.x, data.y, data.z
;;
;; if format.dr = 0, then determines density from the voronoi region
;; if format.dr > 0, then determines density from packets within sphere
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants
common results

;; Determine how dr is set
formtags = strlowcase(tag_names(geometry))
q = fix(total(strmatch(tag_names(format), 'dr', /fold)))
if (q) $
  then dr = format.dr $
  else dr = geometry.dr

if (size(data, /type) NE 8) then stop ;; data must be given as a structure

nf = n_elements(files)
nspec = n_elements(*data.x)

loadem = (savefile EQ !null) ? 1 : ~file_test(savefile)

if (loadem) then begin
  xx = !null & yy = !null & zz = !null & frac = !null & radvel_sun = !null
  for ff=0,nf-1 do begin
    results_loadfile, files[ff], pts, vels_sun, frac2 ;; note - not keeing frac=0
    xx = [xx, pts[*,0]] & yy = [yy, pts[*,1]] & zz = [zz, pts[*,2]]
    frac = [frac, frac2]
    radvel_sun = [radvel_sun, vels_sun[*,1]+stuff.vrplanet]  ;; for g-value
    print, 'Loaded inputs ' + strint(ff+1) + ' of ' + strint(nf)
  endfor

  out = {x:ptr_new(temporary(xx)), y:ptr_new(temporary(yy)), $
    z:ptr_new(temporary(zz)), frac:ptr_new(temporary(frac)), $
    radvel_sun:ptr_new(temporary(radvel_sun))}
  rhosqr_sun = *out.x^2 + *out.z^2
  if (savefile NE !null) then save, out, rhosqr_sun, file=savefile
endif else restore, savefile

;; remove packets outside region of interest
q = where((*out.x GE min(*data.x)-.1) and (*out.x LE max(*data.x)+.1) and $
  (*out.y GE min(*data.y)-.1) and (*out.y LE max(*data.y)+.1) and $
  (*out.z GE min(*data.z)-.1) and (*out.z LE max(*data.z)+.1) and $
  (*out.frac GT 0), nq)
if (nq GT 0) then begin
```

```
    *out.x = (*out.x)[q]
    *out.y = (*out.y)[q]
    *out.z = (*out.z)[q]
    *out.frac = (*out.frac)[q]
    *out.radvel_sun = (*out.radvel_sun)[q]
  endif

  ;; determine packet weighting
  *out.frac = results_packet_weighting(out)

  if (format.quantity NE 'density') then stop ;; only can do points at the moment

  if (dr EQ 0) then begin
    print, 'Using voronoi regions to determine density'

    regions = results_voronoi(out)
    tree = results_kd_tree(out)

    density = results_density(*data.x, *data.y, *data.z, out, regions, tree)
  endif else begin
    vpix = 4./3.*!pi*(dr*SystemConsts.rplan*1e5)^3
    density = fltarr(n_elements(*data.x))
    for i=0,n_elements(*data.x)-1 do begin
      xpr = *out.x-(*data.x)[i]
      ypr = *out.y-(*data.y)[i]
      zpr = *out.z-(*data.z)[i]
      rpr = sqrt(xpr^2 + ypr^2 + zpr^2)
      q = where(rpr LT dr, nq)
      if (nq GT 0) then density[i] = total((*out.frac)[q])/vpix
    endfor
  endelse

  result = {density:ptr_new(density), format:format}
  return, result

  end
```