```
#program windmodel
#This is modified from the third version of
Holland2_batts_v2_for_Walter_v3.f90
#This program can produce U and V wind with direction
#lat in deg N, lon in deg W
#modified by SQ on Aug. 24 to calc wind for Hurricane Irene

#setwd('D:/work')

wind_model <- function(path, tracts){


# source('tdiff.R')
# source('calcangle.R')
# source('f.R')
# source('newton2.R')
#Reading input file
mlines = 1000
yr = vector( mode="integer", length=mlines)
mon = vector( mode="integer", length=mlines)
day = vector( mode="integer", length=mlines)
hr = vector( mode="integer", length=mlines)
latr = vector( mode="numeric", length=mlines)
lonr = vector( mode="numeric", length=mlines)
vmaxr = vector( mode="numeric", length=mlines)

#Interpolation to uniform time intervals
nlines = nrow(path)
ntime = integer()
interval = integer()
dhr = integer()
tint = 0.25 #time interval for interpolation, hrs
dellat = numeric()
dellon = numeric()
delvmx = numeric()
mcase = 10000
lat = vector( mode="numeric", length=mcase)
lon = vector( mode="numeric", length=mcase)
vmax = vector( mode="numeric", length=mcase)
cp = vector( mode="numeric", length=mcase)
r_vmax = vector( mode="numeric", length=mcase)

#Defining the lat/lon grid for calculating surface winds
lonmin = -107.0
latmin = 25.0
nlat = nrow(tracts)
dlon = 1
dlat = 1
glon = vector( mode="numeric", length=nlat)
maxwindspd = vector( mode="numeric", length=nlat)
```

```
duration = vector( mode="numeric", length=nlat)
glat = vector( mode="numeric", length=nlat)
gridid = vector( mode="integer", length=nlat)

#Land/sea arrays
xls = integer()
yls = integer()
ncls=4218
nrls=3643
landsea = matrix(data = -9999, nrow=nrls,ncol=ncls)
icoast = integer()

#Misc values and arrays used within routine
pc = numeric()
track = vector( mode="numeric", length=nlat)
rad = vector( mode="numeric", length=nlat)
uwind = matrix( data=NA, nrow=1000, ncol=nlat )
vwind = matrix( data=NA, nrow=1000, ncol=nlat )
windspd = matrix( data=NA, nrow=1000, ncol=nlat )
pinfinit=1013.25
ts_hr = numeric()
max = numeric()
timeid = integer()
time = integer()
I1 = integer()
nhurr = integer()
ux = integer()
uy = integer()
i = integer()
j = integer()
k = integer()
j1 = integer()
usign = integer()
vsign = integer()
Rmax = numeric()
B = numeric()
temp1 = numeric()
x = numeric()
y = numeric()
r = numeric()
xdiff = numeric()
ydiff = numeric()
xx = integer()
yy = integer()
xx2 = integer()
yy2 = integer()
X2 = integer()
Y2 = integer()
xx1 = integer()
yy1 = integer()
III = integer()
```

```
stormID = integer()
II = integer()
KK = integer()
A = numeric()
n = numeric()
X1 = numeric()
known = numeric()
sigma = numeric()
R1 = numeric()
R2 = numeric()
eps = numeric()
v_temp1 = numeric()
v_temp2 = numeric()
w = numeric()
known1 = numeric()
THETA = numeric()
MDA = numeric()
C = numeric()
THETA1 = numeric()
YY3 = numeric()
XX3 = numeric()
gwd = numeric()
swd = numeric()
dx = numeric()
dy = numeric()
c_x = numeric()
c_y = numeric()
cspeed = numeric()
mult = numeric()
beta = numeric()
ANGLE = numeric()
lat_temp = numeric()
lon_temp = numeric()
MDA1 = numeric()
rcv = numeric()
rcvi = numeric()
xll = numeric()
yll = numeric()
# luinp = 12
# ludat = 13
# lulnd = 14
# luout = 15
# lulog = 16
fninp = character()
fnout = character()
fnlog = character()

#
#Open log, input and output files
#
```

```
luinp <- path
fntemp <- tracts

gridid = fntemp$GEOID10
glat = fntemp$y
glon = fntemp$x
gpop = fntemp$pop


#Read input file, each line must of of form: YY MM DD HH Lat Lon Vmax

r = 0.0
for( i in 1:nlines ){
  yr[i]=luinp[i,1]
  mon[i]=luinp[i,2]
  day[i]=luinp[i,3]
  hr[i]=luinp[i,4]
  latr[i]=luinp[i,5]
  lonr[i]=luinp[i,6]
  vmaxr[i]=0.51444*luinp[i,7] #convert kts to m/s
}
nlines = nlines - 1
if(nlines<1){
  stop("ERROR: input file must have at least 2 data points")
}

#Linearly intepolate to predefined time intervals (based on parameter
"tint" in hours)
kk = 0
for(i in 1:nlines){
  dhr <-
tdiff(yr[i+1],mon[i+1],day[i+1],hr[i+1],yr[i+1],mon[i],day[i],hr[i])
  interval = floor(dhr/tint)
  dellat = (latr[i+1]-latr[i])/interval
  dellon = (lonr[i+1]-lonr[i])/interval
  delvmx = (vmaxr[i+1]-vmaxr[i])/interval
  for(k in 1:interval){
    kk = kk + 1
    lat[kk] = latr[i] + (k-1)*dellat
    lon[kk] = lonr[i] + (k-1)*dellon
    vmax[kk] = vmaxr[i] + (k-1)*delvmx
  }
}
lat[kk+1] = latr[nlines]
lon[kk+1] = lonr[nlines]
vmax[kk+1] = vmaxr[nlines]

ntime = kk+1 # why kk+1 and not kk
# convert 30 minute wind at 10 m to central pressure
for(i in 1:ntime){
  cp[i] = (262.8 - vmax[i]*0.8139)/0.23 # need to convert vmax(i) to 30
```

```
minute wind at 10 m ^CHECK THIS!!!!
}
# convert 1-min sustained wind at 10m to gradient level wind speed (for
use in Holland wind profile calculation)
for(i in 1:ntime){
  r_vmax[i] = vmax[i]/0.9
}
# conversion factor for degrees to radians
rcv = 3.14/180.0
rcvi = 180.0/3.14

# Start loop over time steps
for(i in 1:ntime){
  # calculate the x and y components of forward speed, in m/s
  lon2km = 111.32*cos(rcv*(lat[i]))
  lat2km = 110.54
  if(i==1){
  dx = lon2km*(lon[i+1]-lon[i])
  dy = lat2km*(lat[i+1]-lat[i])
  c_x = (1000.*dx)/(tint*3600)
  c_y = (1000.*dy)/(tint*3600)
  cspeed = sqrt(dx*dx+dy*dy)
  } else if (i == ntime){
  dx = lon2km*(lon[i]-lon[i-1])
  dy = lat2km*(lat[i]-lat[i-1])
  c_x = (1000.*dx)/(tint*3600.)
  c_y = (1000.*dy)/(tint*3600.)
  cspeed = sqrt(dx*dx+dy*dy)
  } else {
  dx = lon2km*(lon[i+1]-lon[i-1])
  dy = lat2km*(lat[i+1]-lat[i-1])
  c_x = (1000.*dx)/(2.0*tint*3600.)
  c_y = (1000.*dy)/(2.0*tint*3600.)
  cspeed = sqrt(dx*dx+dy*dy)
  }
  #Reduce VMAX by forward speed (will be added back in after calculating
wind profile)
  r_vmax[i] = r_vmax[i] - cspeed
  if (r_vmax[i] < 0.0){
    r_vmax[i] = 0.0
  }
  # Calculate Radii for wind model calculations
  Rmax = 46.4*exp(-0.0155*r_vmax[i]+0.0169*lat[i])   # Willoughby et al.
2006, Eqn 7a
  A = 0.0696 + 0.0049*r_vmax[i]-0.0064*lat[i]   # Willoughby et al.
2006, Eqn 10c
  if(A<0){
    A=0
  }
  n=0.4067 + 0.0144*r_vmax[i]-0.0038*lat[i]                      #
Willoughby et al. 2006, Eqn 10b
```

```
  X1=317.1-2.026*r_vmax[i]+1.915*lat[i]                         #
Willoughby et al. 2006, Eqn 10a
  X2=25.0
  known=n*((1-A)*X1+A*X2)/(n*((1-A)*X1+A*X2)+Rmax)      # Willoughby et
al. 2006, Eqn 3 (RHS)

  #sub function newton2( unfinished )
  sigma = newton2(known)
  if(Rmax>20){
    R1 = Rmax - 25*sigma
    R2 = R1 + 25.0
  } else {
    R1 = Rmax - 15*sigma
    R2 = R1 + 15.0
  }

   if (r == 0.0){
      pc = cp[i]
    } else {
      pc=cp[i]+(pinfinit-cp[i])*exp((-1)*Rmax/r)
   }

  # calculate motion direction angle
  # Use trig angles, so E=0, N=90, W=180, S=270)
  mda <- calcangle(dx,dy)
  # start loop over all lat/lon grid points
  for(j in 1:nlat){
    uwind[i,j] = 0.0
    vwind[i,j] = 0.0
    windspd[i,j] = 0.0
    track[j] = 0.0

    dx = lon2km*(lon[i]-glon[j])
    dy = lat2km*(glat[j]-lat[i]) #changed k to j
    r = sqrt(dx*dx+dy*dy)
    # calculate the gradient wind direction (gwd) at this grid point
    gwd <- calcangle(dx, dy)
    gwd = gwd - 90.0

    if (gwd < 0.0){
      gwd = 360.0 + gwd
    } else if (gwd > 360.0){
      gwd = gwd - 360.0
    }

    # Begin Holland2 model to calculate gradient windspeed distribution
    # *Note:  Wind speed scaled by 100 - must undo after calculation
    if(r < R1){
      track[j]=r_vmax[i]*(r/Rmax)^n*100
    } else if (r > R2){
      track[j]=r_vmax[i]*((1-A)*exp((Rmax-r)/X1)+A*exp((Rmax-r)/X2))*100
    } else if(r > R1 && r < R2){
      eps=(r-R1)/25
      w=126*eps^5-420*eps^6+540*eps^7-315*eps^8+70*eps^9
      v_temp1=r_vmax[i]*(r/Rmax)^n
      v_temp2=r_vmax[i]*((1-A)*exp((Rmax-r)/X1)+A*exp((Rmax-r)/X2))
      track[j]=(v_temp1*(1-w)+v_temp2*w)*100
    }
    track[j] = track[j]/100.0
    if (track[j] < 0.0){
      track[j] = 0.0
    }
    swd = gwd + 20
    mult = 0.9
    if (swd < 0.0){
      swd = 360.0 + swd
    } else if (swd > 360.0){
      swd = swd - 360.0
    }
    rad[j] = X2
    # Calculate the u and v components of surface wind
    uwind[i,j] = mult*cos(rcv*swd)*track[j]
    vwind[i,j] = mult*sin(rcv*swd)*track[j]
    # Calculate total wind speed
    windspd[i,j] = sqrt((uwind[i,j])^2+(vwind[i,j])^2)
    # Add back in wind component due to storm motion (only where windspd
> 0 m/s)
    # From NOAA Technical Report 23, Schwerdt et al., pg. 25
    if (windspd[i,j] > 0.0){
      beta = swd-mda
      windspd[i,j] = windspd[i,j]+1.5*
((cspeed)^0.63)^((0.514751)^0.37)*cos(rcv*beta)
      if (windspd[i,j] < 0.0){
        windspd[i,j] = 0.0
      }
    }
    # print*, beta,mda,swd

    # Convert 1-min winds at 10-m to 3-sec gust at surface
    windspd[i,j] = windspd[i,j]*(1.3)
  }
}

for(j in 1:nlat){
  maxwindspd[j] = max(windspd[1:ntime,j])
}

for( j in 1:nlat ){
  duration[j] = 0
}

for(i in 1:ntime){
  for(j in 1:nlat){
    if(windspd[i,j]>20){
      duration[j] = duration[j] + 15.0
    }
  }
}
#output

output <- cbind(gridid, glat, glon, maxwindspd, duration, gpop)
return(output)
}
```