microclima: micro- and meso-scale climate modelling with R

Ilya M. D. Maclean[aut, cre], Jonathan R. Mosedale[aut], Jonathan J. Bennie [aut], James P. Duffy [aut]

2019-10-14

Introduction

This vignette describes the R package 'microclima'. The package contains a series of functions for downscaling climate to micro- and meso-scales. The core assumption is that local anomolies from standard reference temperatures, for example weather station data or coarse-resolution gridded data, can be modelled using the mechanistic processes that govern variation in fine-scale climate. Functions associated with two types of model are presented: a mesoclimate model for estimating local variation in ambient air temperatures, and a microclimate model for estimating finer-scale variation in near-ground temperatures. The models are designed to be coupled together: microclimate temperatures can be derived from the outputs of the mesoclimate model. The microclimate accounts for the effects of vegetation and topography on solar radiation flux and wind speed. The mesoclimate model, as well as accounting for temperature variation driven by topographic effects, models the influences of elevation, coastal processes and cold-air drainage. Typically, the models are run in hourly time-steps. Throughout, we illustrate use of the package by applying the model to the Lizard Peninsula in Cornwall, UK.

Microclimate temperatures

The difference between near-surface temperature and reference air temperature is modelled as a linear function of net radiation. The gradient of this linear relationship is a measure of the thermal coupling of the surface to the atmosphere. If this relationship is applied to vegetation, assuming the canopy to act like a surface, the gradient varies as a function of both the structure of the vegetation and wind speed, and the relationship can be readily parametrised from field temperature data, using the function `fitmicro`, if local net radiation and wind speed are known, as shown below.

```
# = = = = = = = = = Example = = = = = = = = = = = = = =
head(microfitdata) # example field data
fitmicro(microfitdata) # derive model coefficients
```

Mesoclimate temperatures

The mesoclimate functions ignore the effects of radiation transmissions though canopies and variation in ground surface albedo, as these are accounted for in the microclimate model. Differences between local temperatures and reference air temperature are derived as a function of coastal, cold-air drainage and elevation effects and the effects of meso-scale topography on radiation, as in the topographic examples shown below. Coastal, elevation and cold air drainage effects are calculated prior to fitting the model. In consequence, the mesoclimate model can also be fitted using the `fitmicro` function, if local net radiation and wind speed are known.

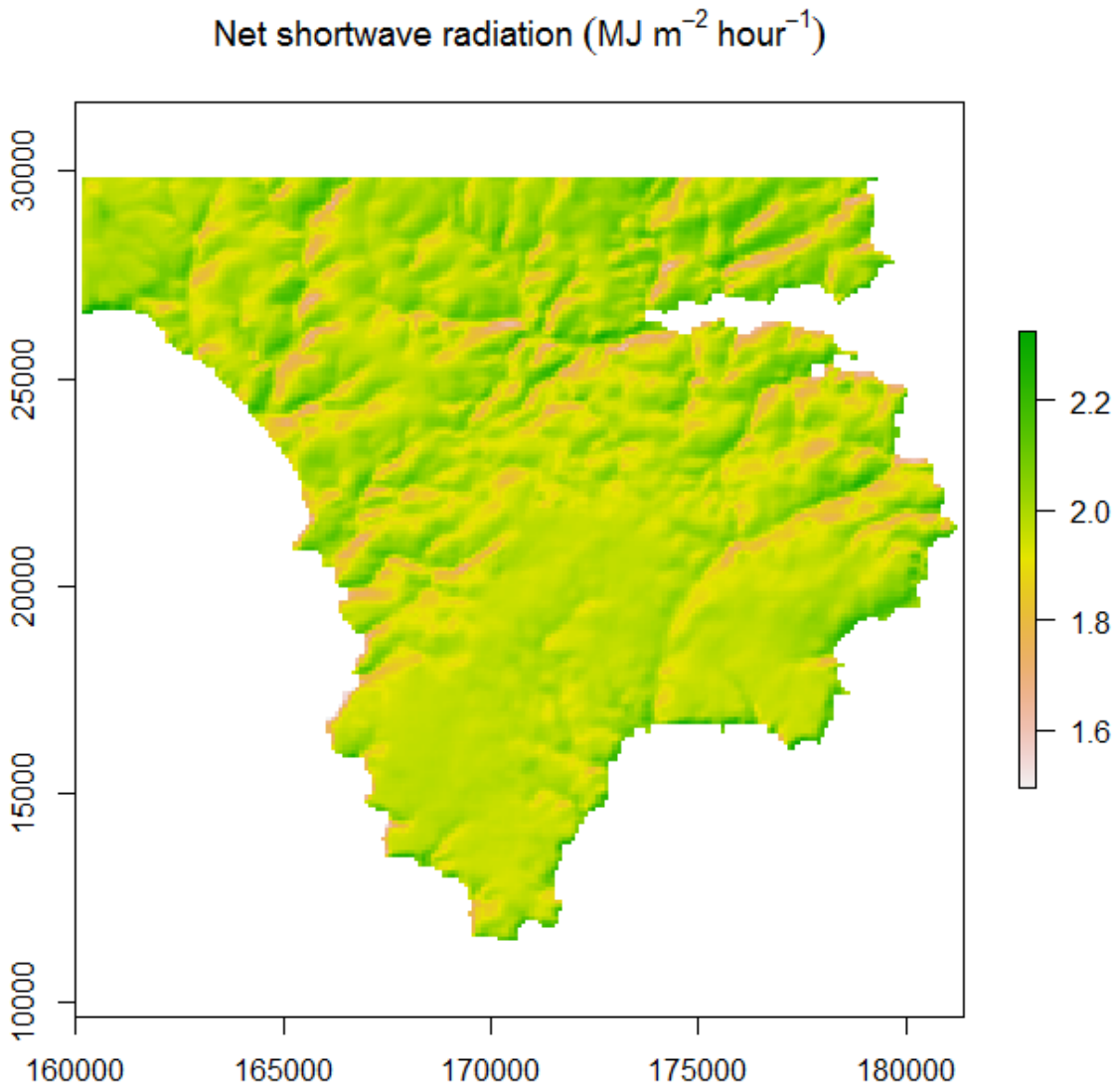
Examples of how the various factors influencing micro- and mesoclimate are downscaled are shown below. At the end of this document, fully executable code for running both models is provided.

Downscaling radiation

The net radiation flux is determined by the balance of incoming shortwave radiation and emitted longwave radiation, with the former partitioned between direct and diffuse components. Our model assumes that coarse-scale or point location measurements of these have been obtained, but provides methods for accounting for the effects of topography and vegetation on radiation. Topography determines whether a given location is shaded and also the angle at which the sunlight strikes the surface. Vegetation attenuates radiation as it passes though the canopy. Both also influence longwave radiation. If the direct and diffuse components of shortwave radiation are not known individually, diffuse radiation can be estimated from total incoming shortwave radiation using the function `difprop`.

```
# = = = = Example: topographic effects on shortwave radiation = = = = = =
# ================================
# Extract data for 2010-05-24 11:00
# ================================
dni <-dnirad[,,3444]
dif <-difrad[,,3444]
# ==========================
# Resample to 100m resolution
# ==========================
dnir <- raster(dni, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
difr <- raster(dif, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
crs(dnir) <- '+init=epsg:4326'
crs(difr) <- '+init=epsg:4326'
dnir <- projectRaster(dnir, crs = "+init=epsg:27700")
difr <- projectRaster(difr, crs = "+init=epsg:27700")
dni <- resample(dnir, dtm100m)
dif <- resample(difr, dtm100m)
sv <- skyviewtopo(dtm100m) # calculates skyview correction factor
jd <- julday(2010, 5, 24) # calculates astronomical julian day
ha <- mean_slope(dtm100m) # calculates mean slope to horizon
# ====================================================================
# Calculate and plot net shortwave radiation for 2010-05-24 11:00
# ====================================================================
netshort100m <- shortwavetopo(dni, dif, jd, 11, dtm = dtm100m,
                              svf = sv, ha = ha)
plot(mask(netshort100m, dtm100m),
     main = expression("Net shortwave radiation" ~ (MJ ~ m^{-2} ~   hour^{-1})))
```

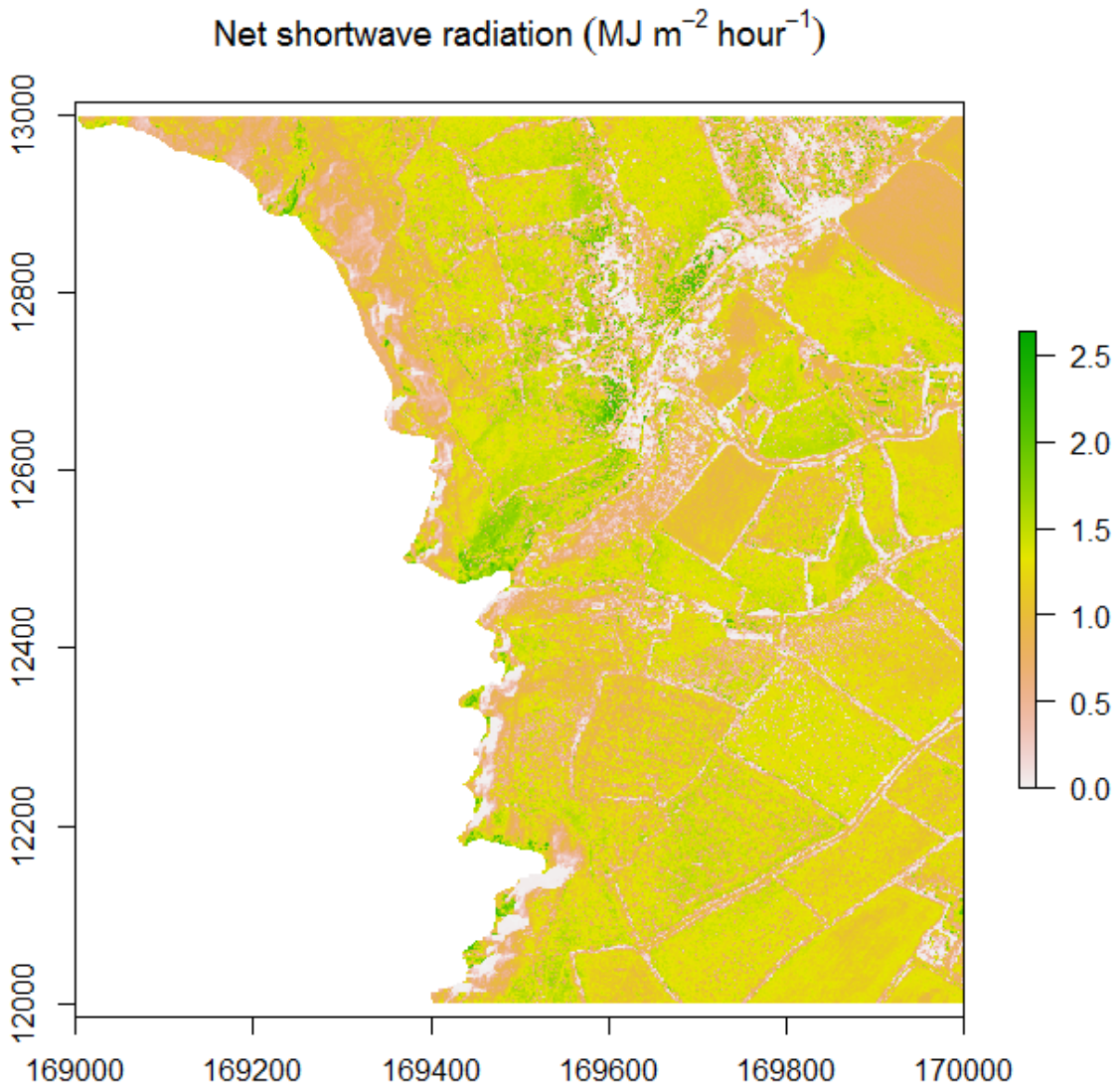
In the example above, 0.05° resolution direct and diffuse radiation are first extracted from the radiation datasets included with the package (2010-05-24 11:00). This is converted to a raster, reprojected to the British National Grid and resampled to 100 m resolution. Next, as diffuse radiation is downscaled by accounting for the proportion of the sky hemisphere in view, this is calculated using a 100 m resolution dtm using function `skyviewtopo`. Radiation reflected back from adjacent surfaces is dependent on the mean horizon angle, which is calculated using function `mean_slope`. The direct radiation flux reaching inclined surfaces depends on the solar altitude and azimuth. For this reason, the astronomical Julian day is calculated using function `julday`. Finally, the total shortwave radiation flux reaching inclined surfaces is calculated using function `shortwavetopo`. This function includes options to output different components of the shortwave radiation budget, and allows the user to specify surface albedo if net radiation is required. The code above produces the plot below.




```
# = Example: vegetation and topographic = = = = = =
# = = = effects on shortwave radiation  = = = = = =
#
#================================
# Extract data for 2010-05-24 11:00
# ================================
dni <- microvars$dni[564]
dif <- microvars$dif[564]
# ========================
# Calculate input paramaters
# ========================
x <- leaf_geometry(veg_hgt)
l <- lai(aerial_image[,,3], aerial_image[,,4])
```

```
l <- lai_adjust(l, veg_hgt)
fr <- canopy(l, x)
alb <- albedo(aerial_image[,,1], aerial_image[,,2], aerial_image[,,3],
              aerial_image[,,4])
albg <- albedo2(alb, fr)
sv <- skyviewveg(dtm1m, l, x)
ha <- mean_slope(dtm1m)
jd <- julday(2010, 5, 24)
# ================================================================
# Calculate and plot net shortwave radiation for 2010-05-24 11:00
# ================================================================
netshort1m <- shortwaveveg(dni, dif, jd, 11, dtm = dtm1m, svv = sv,
                           albg = albg, fr = fr, ha = ha, x = x, l = l)
plot(mask(netshort1m, dtm1m),
     main = expression("Net shortwave radiation" ~ (MJ ~ m^{-2} ~     hour^{-1})))
```

In the example above, point direct and diffuse radiation are first extracted from the radiation datasets included with the package for 2010-05-24 11:00. Next, as vertically oriented leaves shade out more radiation at low solar angles, the ratio of vertical to horizontal projections of leaf foliage are calculated from a one m resolution vegetation height dataset using function `leaf_geometry`, by assuming an allometric relationship between the two. The vegetation height dataset was originally derived as the difference between a dsm and dtm. Radiation transmission is also affected by leaf area, and this is calculated from a multispectral aerial image using function `lai`, by assuming a relationship between area and NDVI. Next, as diffuse radiation is downscaled by accounting for the proportion of the sky hemisphere in view, this is calculated using function `skyviewveg`. As previously, the mean horizon angle and astronomical Julian day are also calculated. To determine the net radiation absorbed at the ground below vegetation canopies, it is necessary to know both ground and canopy albedos. These are derived from aerial imagery using the functions `alb`, which calculates the albedo of the image, and `albg`, which partitions this value between ground and canopy albedo based on knowledge of fractional canopy cover, itself determined using function `canopy`. Finally, the total shortwave radiation flux reaching inclined surfaces underneath vegetation is calculated using function `shortwaveveg`. The code above produces the plot below.

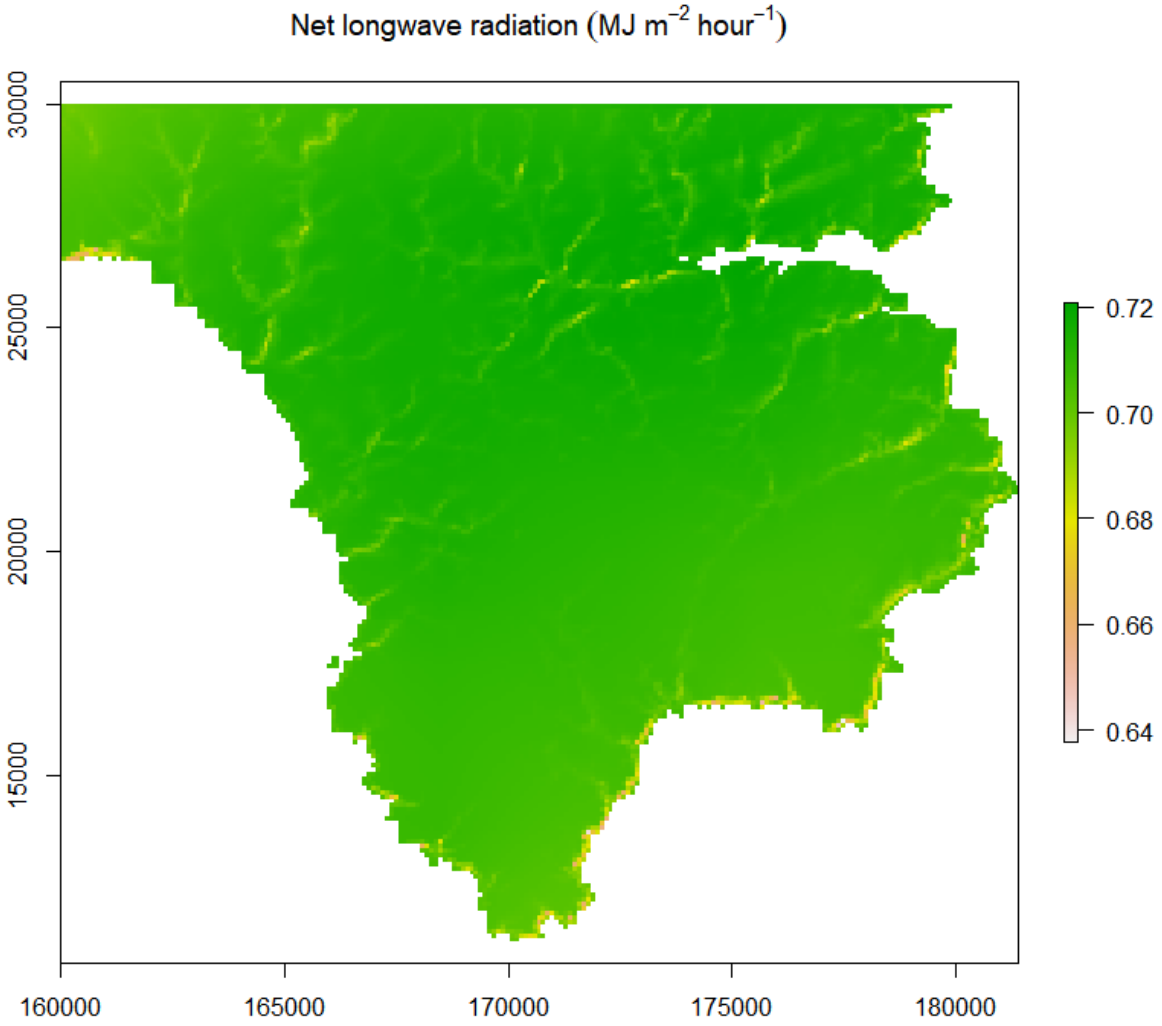


Net longwave radiation is assumed to be a function of reference temperature and the emissivity of the atmosphere, with emissivity dependent on fractional cloud cover and humidity. In areas where part of the sky view is obscured by topography, a sky view correction factor is applied. Underneath vegetation, a significant proportion of the downward longwave radiation from the atmosphere is reflected, absorbed and re-emitted or scattered by the canopy.

```
# = Example: topographic effects on longwave radiation = = = =
# ================================
# Extract data for 2010-05-24 11:00
# ================================
h <- huss[,,144] # specific humidity
p <- pres[,,144] # pressure
n <-cfc[,,3444]  # fractional cloud cover
tc <- tas[,,144] + 0.5*dtr[,,144] # temperature (deg C)
```

```
sv <- skyviewtopo(dtm100m)
# ============================
# Resample to 100m resolution
# ============================
hr <- if_raster(h, dtm1km)
tr <- if_raster(tc, dtm1km)
pr <- if_raster(p, dtm1km)
nr <- raster(n, xmn = -5.40, xmx = -5.00, ymn = 49.90, ymx = 50.15)
crs(nr) <- '+init=epsg:4326'
nr <- projectRaster(nr, crs = '+init=epsg:27700')
hr <- resample(hr, dtm100m)
tr <- resample(tr, dtm100m)
pr <- resample(pr, dtm100m)
nr <- resample(nr, dtm100m)
# ========================================
# Calculate and plot net longwave radiation
# ========================================
netlong100m <- longwavetopo(hr, tr, pr, nr, sv)
netlong100m <- mask(netlong100m, dtm100m)
plot(netlong100m,
     main = expression("Net longwave radiation" ~ (MJ ~ m^{-2} ~ hour^{-1})))
```

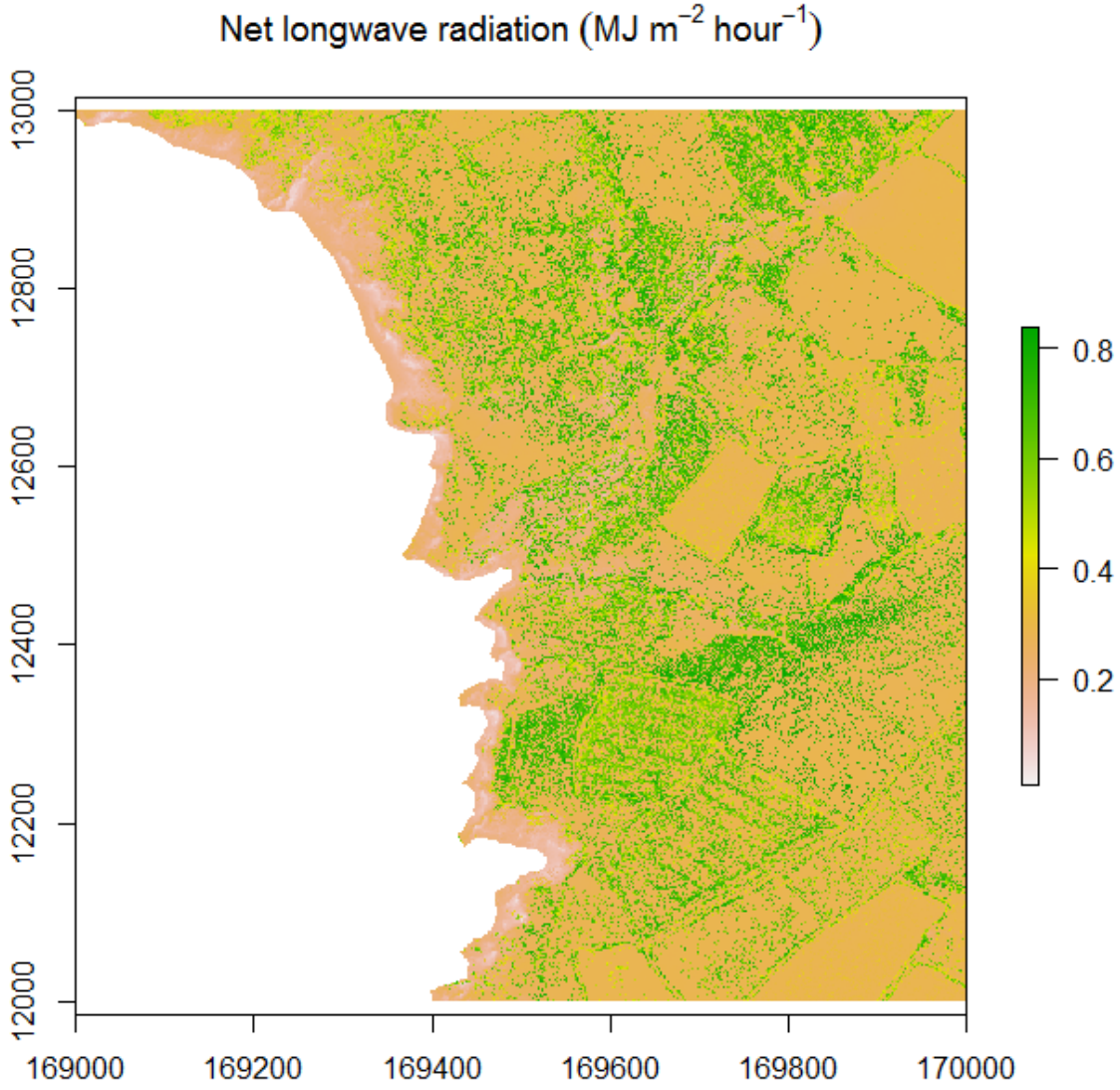
In the example above, the meteorological variables needed to calculate atmospheric emissivity for 2010-05-24 11:00 are first extracted from the one km resolution dataset include with the package. The package also includes datasets of daily mean temperature and the diurnal temperature range. For the sake of simplicity and for illustration only, we assume the temperature at 11am is the mean daily temperature + half the diurnal temperature range. As spatial variation in longwave radiation is affected by sky view, this is calculated using function `skyviewtopo`. The meteorological variables are then converted to a raster and resampled to 100 m resolution. Finally, longwave radiation is downscaled to 100m resolution using function `longwavetopo`. This function calculates atmospheric emissivity and applies the sky view correction factor. The functions `if_raster` and `is_raster` are included with the package to permit flexibility in using either raster datasets or matrices to run components of the model. The code above produces the plot below.




```
# = Example: vegetation and topographic = = = = = =
# = = = effects on longwave radiation = = = = = = =
# ================================
# Extract data for 2010-05-24 11:00
# ================================
h <- microvars$humidity[564]
p <- microvars$pressure[564]
n <- microvars$cloudcover[264]
tcr <- raster(temp100[,,564], xmn = 169000, xmx = 170000, ymn = 12000, ymx = 13000)
tc <- resample(tcr, dtm1m) # Resample temperature raster to 1m
# ========================
# calculate input variables
# ========================
x <- leaf_geometry(veg_hgt)
l <- lai(aerial_image[,,3], aerial_image[,,4])
l <- lai_adjust(l, veg_hgt)
```

```
svv <- skyviewveg(dtm1m, l, x)
fr <- canopy(l, x)
alb <- albedo(aerial_image[,,1], aerial_image[,,2], aerial_image[,,3],
             aerial_image[,,4])
albc <- albedo2(alb, fr, ground = FALSE)
# ====================================
# calculate and plot longwave radiation
# ====================================
netlong1m <-longwaveveg(h, tc, p, n, x, fr, svv, albc)
nlr <- mask(netlong1m, dtm1m)
plot(nlr, main = expression("Net longwave radiation" ~ (MJ ~ m^{-2} ~   hour^{-1})))
```

In the example above, the meteorological variables needed to calculate atmospheric emissivity for 2010-05-24 11:00 are first extracted from the dataset include with the package. The package also includes datasets of temperature at 100 m, produced using the mesoclimate model. Temperatures are extracted from this dataset, converted to a raster and resampled to one m resolution. Below vegetation, a significant proportion of the downward longwave radiation from the atmosphere is reflected, absorbed and re-emitted or scattered by the canopy. For this reason, leaf metrics and albedos are calculated as in the example for shortwave radiation above. As spatial variation in longwave radiation is affected by sky view this is calculated using function `skyviewveg`. The meteorological variables are then converted to a raster and resampled to 100 m resolution. Finally, longwave radiation is downscaled to one m resolution using function `longwaveveg`. This function calculates atmospheric emissivity, applies the sky view correction factor and accounts for vegetation and albedo effects. The code above produces the plot below.



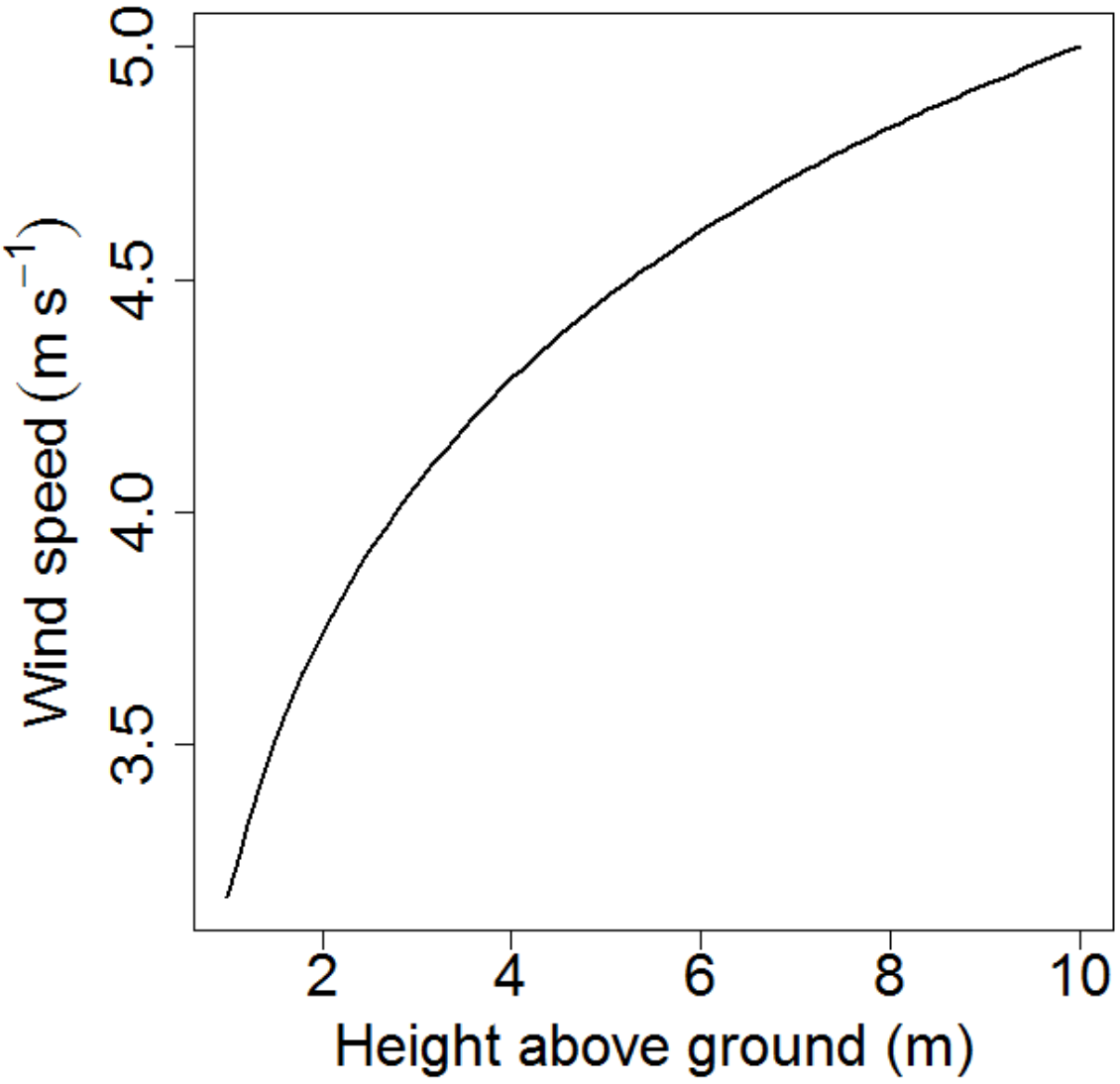
Downscaling wind speed

Wind speeds measured at different heights above the ground vary. Surface friction tends to slow down wind passing over it. Wind speed is slowest at the surface and increases with height. For this reason anemometers are placed at a chosen standard height, e.g., 10 m. Our models require wind speed estimates at one metre above the ground. The function `windheight` allows estimates of wind speed to be derived at different heights by assuming a logarithmic wind speed profile.

```
# = = = = = Example: wind speed and height = = = = =
z <- c(10:100) / 10 # heights for which wind speed is required
u <- 0 # wind speed
for (i in 1:length(z)) {
```

```
  # Derives wind speed at height z, assuming that wind speed at 10m is 5 m/s
  u[i] <- windheight(5, 10, z[i])
}
par(mar=c(7,7,2,2))
plot (u~z, type = "l", xlab ="Height above ground (m)",
      ylab = expression("Wind speed" ~ (m ~ s^{-1})),
      cex.axis = 2, cex.lab = 2, lwd = 2)
```

In the example above, wind speeds at varying heights above the ground are derived from a wind speed of 5 m/s at 10 m to produce the logarithmic height profile plot shown below.

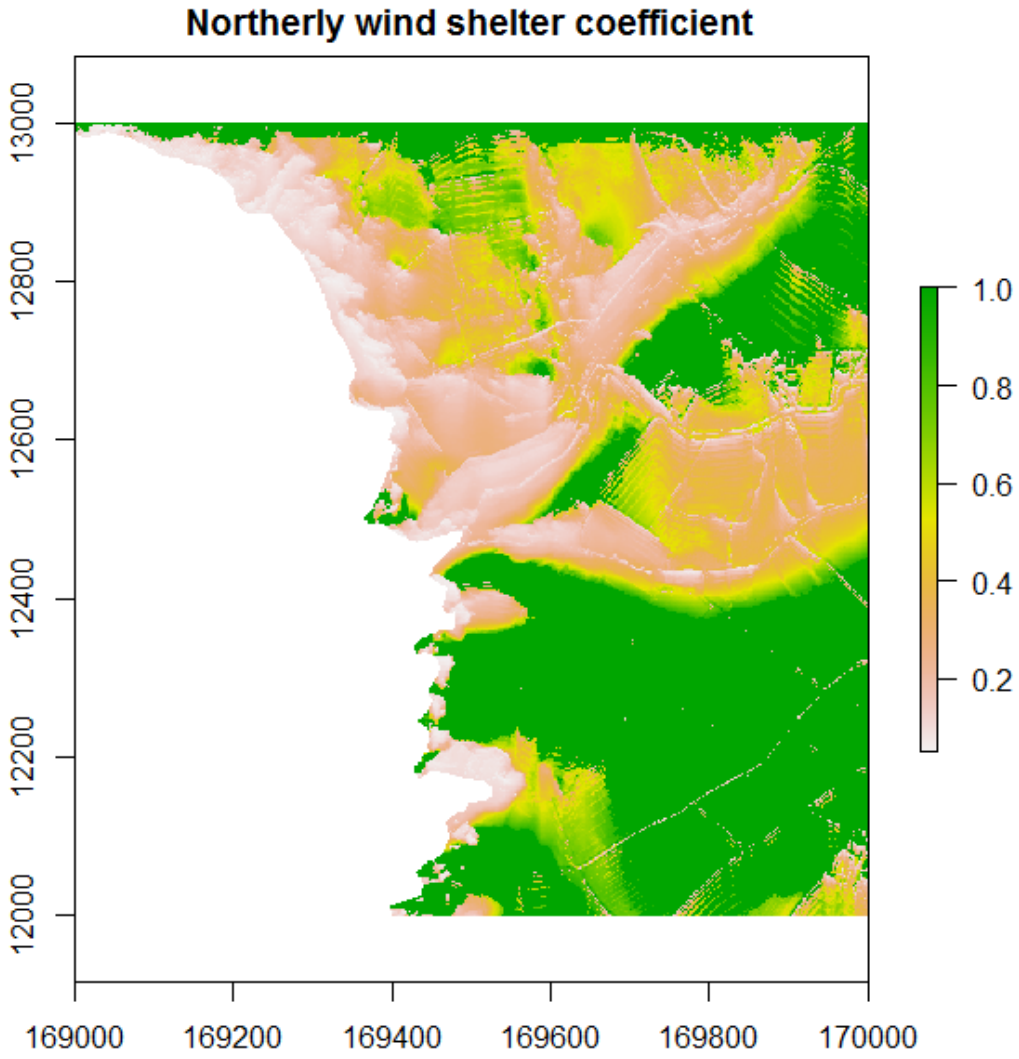


Topography also exerts a sheltering effect, with higher daytime temperatures experienced in sheltered valleys because the linear relationship between radiation and local temperature anomalies has steeper gradient when wind speeds are lower. To account for this, the `windcoef` function is used to calculate a topographic shelter

coefficient.

```
# = = = = = Example: wind shelter coefficient = = = = =
dsm <- dtm1m + veg_hgt
wc <- windcoef(dsm, 0)
plot(mask(wc, dtm1m), main ="Northerly wind shelter coefficient")
```

In the example above, the shelter coefficient is derived from a one metre resolution digital surface model to calculate the degree of sheltering from northerly winds. The digital surface model is producing by adding a vegetation height dataset to a digital terrain model. Both datasets are included with the package. The example above produces the plot below.

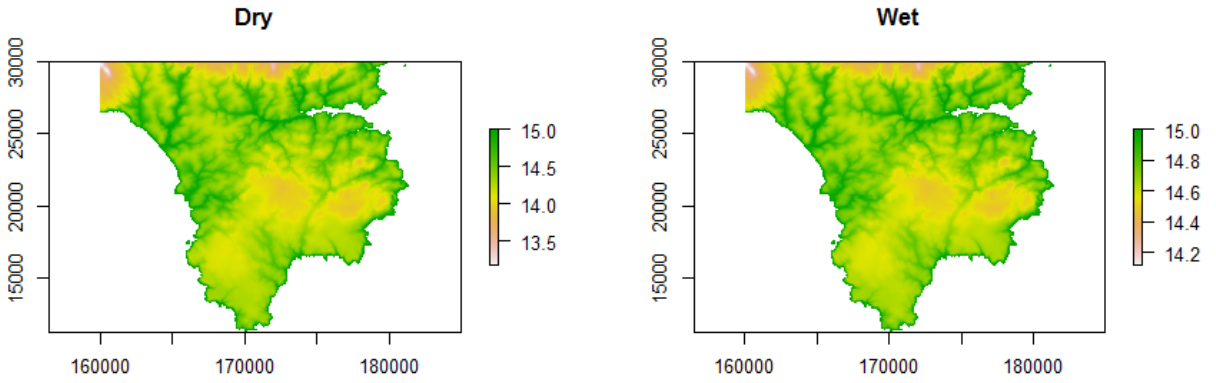


Elevation effects

Elevation effects are calculated using function `lapserate` and applying the resulting adjustment to all pixels of a digital terrain model. The lapse rate is allowed to vary as a function of humidity.

```
# = = = = = = Example of applying lapse rates = = = = = = = = =
# = = dry lapse rate per m = = #
lrd <- lapserate(15, 0)
# = = lapse rate when relative humidity is 100%
h <- humidityconvert(100, intype = "relative", 15)
lrw <- lapserate(15, h$specific) #
# lapse rate applied assuming 15 deg C sea-level temperature
tempdry <- 15 + dtm100m * lrd
tempwet <- 15 + dtm100m * lrw
par(mfrow=c(2,1))
plot(tempdry, main = "Dry")
plot(tempwet, main = "Wet")
```

In the example above, first the dry adiabatic lapse rate is calculated, using function `lapserate` and assuming a sea-level temperature of 15 degrees C. Next the lapse rate when relative humidity is 100% is calculated. The `lapserate` function requires that humidity data are expressed as specific humidity, so the humidityconvert function included with package is used to do this. The lapse rates are then applied to a 100m dtm included with the package to produce the plots shown below.

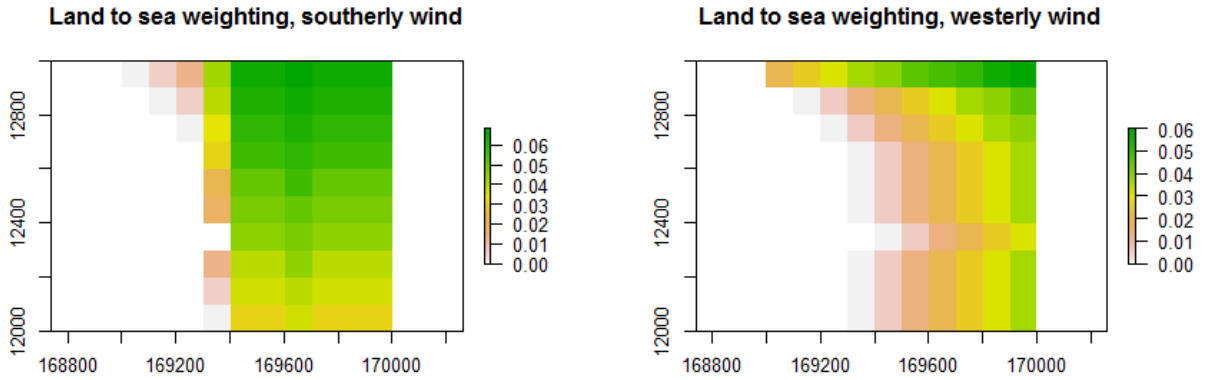


Coastal effects

The model assumes that coastal effects are a function of differences in sea temperature from land temperatures, coastal exposure in an upwind direction and coastal exposure irrespective of direction. The function `invls` permits coastal exposure in a specified direction to be calculated, based on the proportion of pixels upwind that are land or sea. The same function can be applied at fixed intervals over the full 360º to determine general exposure.

```
# = =  Example of calculating coastal exposure = = #
ls1 <- invls(dtm100m, extent(dtm1m), 180)
ls2 <- invls(dtm100m, extent(dtm1m), 270)
par(mfrow=c(2,1))
plot(ls1, main = "Coastal exposure, southerly wind")
plot(ls2, main = "Coastal exposure, westerly wind")
```

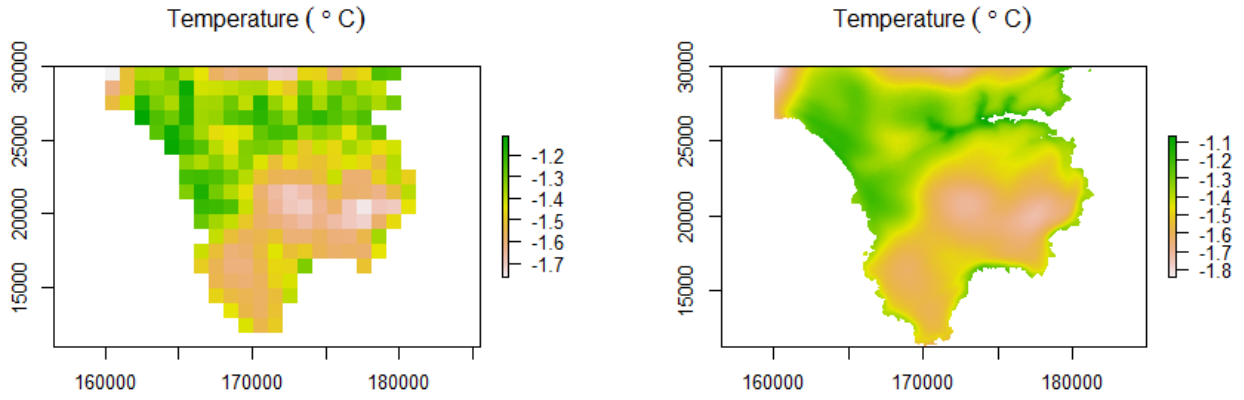
In the example above, the `invls` function is applied to a raster object in which NAs represent sea. For speed, the raster is cropped as the function is quite slow when applied to datasets. Exposure to both southerly and westerly winds is determined. The code above produces the plots below.



The overall coastal effects on temperature are derived using function `coastalTps`, which uses thin-plate spline interpolation with the covariates described above, to derive higher resolution temperature estimates for each time step from coarse-gridded reference temperature data.

```
# = =  Example of calculating coastal effects = = #
# ========================================
# Calculate land-sea temperature difference
# ========================================
temp <- tas[,,1] # land temperature
sst <- 10.665 # sea temperature
dT <- if_raster(sst - temp, dtm1km)
# ============================
# Obtain coastal exposure data
# ============================
lsw <- landsearatios[,,7] # upwind (NNE wind) (dataset derived using invls function)
lsa <- apply(landsearatios, c(1, 2), mean) # mean, all directions
lsw <- if_raster(lsw, dtm100m)
lsa <- if_raster(lsa, dtm100m)
# ============================================================
# Calculate coastal effects using thin-plate spline and plot
# ============================================================
dTf <- coastalTps(dT, lsw, lsa)
par(mfrow = c(2, 1))
plot(sst - dT, main = expression(paste("Temperature ",(~degree~C))))
plot(sst - dTf, main = expression(paste("Temperature ",(~degree~C))))
```

In the example above, land-sea temperature differences are calculated for 2010-01-01 from a one km gridded dataset of temperature data included with the package, and assigned a sea temperature for that day of 10.665 degrees C. The package includes data for coastal exposures, calculated using the `invls` function. Coastal effects are then derived using thin-late spline interpolation using the function `coastalTps` to produce the right-hand plot below. The originally one km gridded data are shown in the left-hand plot.



Cold air drainange effects

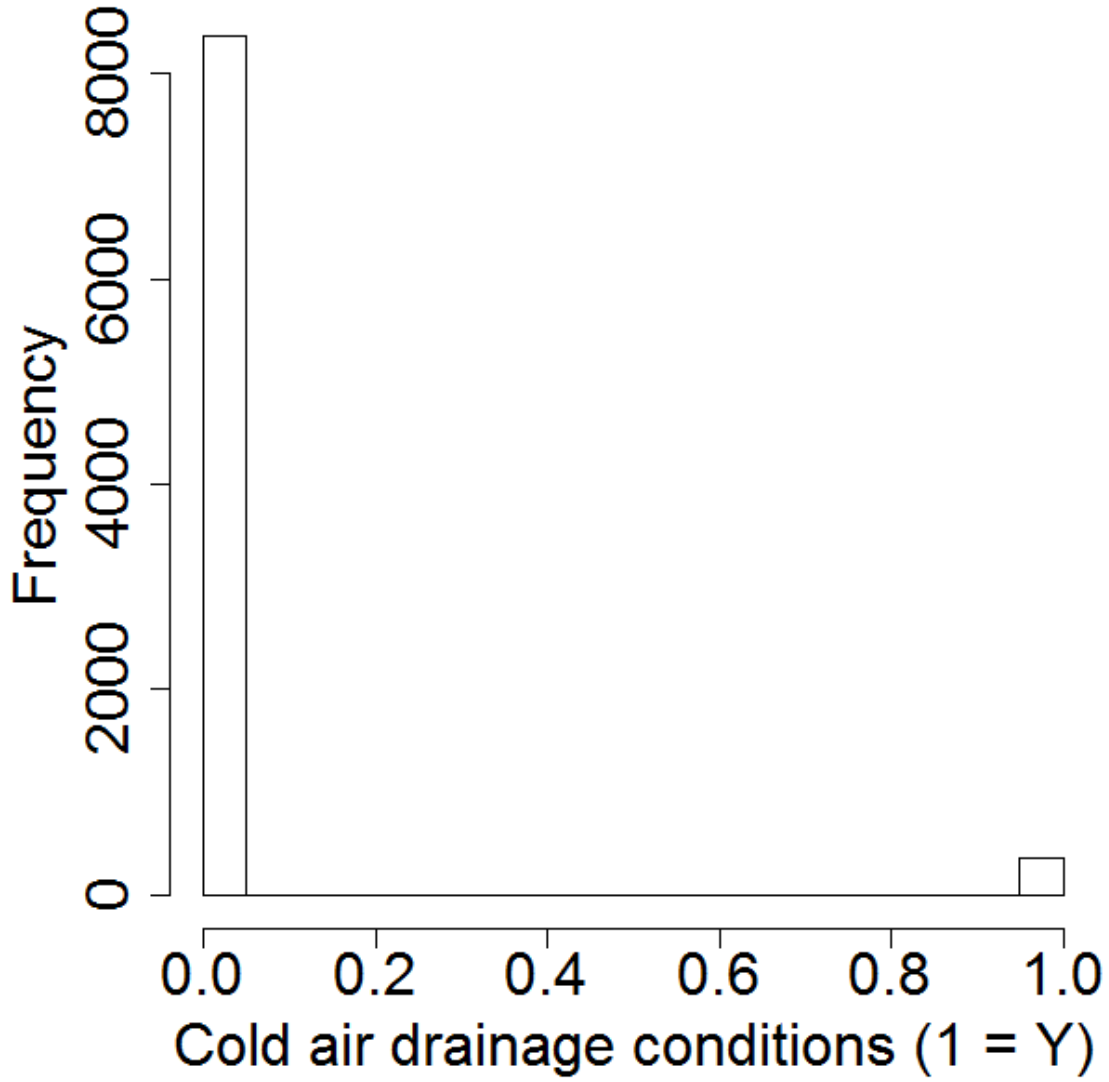
Cold air drainage is modelled as a function of a binary variable indicating whether meteorological conditions are such that cold air drainage occurs, the lapse rate, the elevation difference in metres of a given location and the highest point of a drainage basin, and the log accumulated flow expressed as a proportion of the maximum in each basin.

The function `pcad` calculates a spatial dataset of cold air drainage potential - i.e. the expected temperature differences resulting from cold air drainage should it occur. The function `cadconditions` is used to determine whether meteorological conditions are such that this dataset should be applied. Cold air drainage typically occurs at night or shortly after dawn, in clear sky and calm conditions.

```
# = =  Example of calculating cold air drainage conditions = = #
# ==============================================
# Mean daily climate for Lizard, Cornwall in 2010
# ==============================================
h <- apply(huss, 3, mean, na.rm = TRUE)
p <- apply(pres, 3, mean, na.rm = TRUE)
tmin <- apply((tas - 0.5 * dtr), 3, mean, na.rm = TRUE)[2:364]
tmax <- apply((tas + 0.5 * dtr), 3, mean, na.rm = TRUE)[2:364]
# ====================================
# hourly climate 2nd Jan to 30 Dec 2010
# ====================================
h <- spline(h, n = 8737)$y[13:8724]
p <- spline(p, n = 8737)$y[13:8724]
n <- apply(cfc[,,13:8724], 3, mean)
rdni <- apply(dnirad[,,13:8724], 3, mean)
rdif <- apply(difrad[,,13:8724], 3, mean)
jd <- julday(2010, 2, 1)
jd <- c(jd:(jd+362))
tc <- hourlytemp(jd, em = NA, h, n, p, rdni, rdif, tmin, tmax, 50.05, -5.19)
ws10m <- spline(wind2010$wind10m, n = 8755)$y[25:8736]
ws1m <- windheight(ws10m, 10, 1)
# ================================================================
# Calculate whether cold air drainage, persists and plot proportion
# ================================================================
startjul <- julday(2010,1,2)
par(mar=c(7,7,2,2))
```

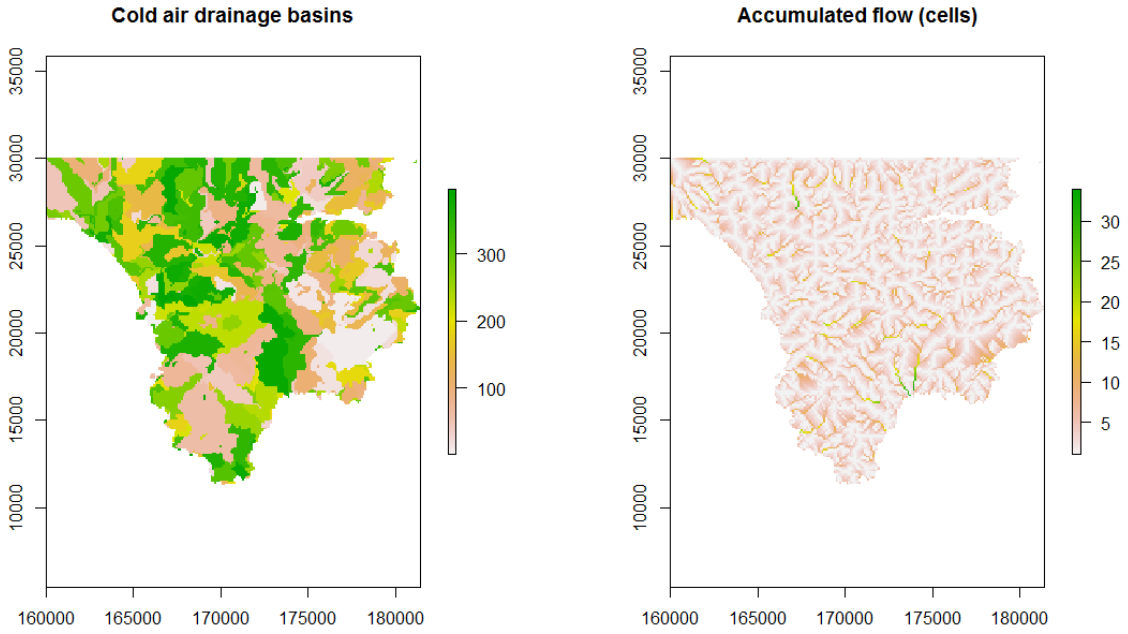
```
hist(cadconditions(h, tc, n, p, ws1m, startjul, 50.05, -5.19),
     main = "", xlab = "Cold air drainage conditions (1 = Y)",
     cex.axis = 2, cex.lab = 2)
```

In the example above, the proportion of the time in 2010 that cold air drainage conditions existed across the Lizard is calculated from meteorological data. First, mean daily data are derived from gridded datasets included with the package. Since hourly data are required, spline interpolation is applied to the humidity, cloud cover and pressure data. Temperatures exhibit more complex diurnal cycles and a function `hourlytemp` is included with package for deriving hourly data from daily maxima and minima. This function assumes that diurnal temperature patterns follow a predictable periodic day-night cycle, but deviate from this due to cloud conditions at night and sub-daily variation in the shortwave radiation flux during the day. The final part of the code calculates a binary variable indicating whether conditions are suitable for cold air drainage and plots this variable as a histogram as shown below.



Prior to applying function `pcad`, cold air drainage basins must be delineated and flow accumulation calculated. In the example below, drainage basins are first delineated from a one m resolution dtm coarsened to 20m using function `basindelin`. The dtm is coarsened as the function is quite slow on large datasets. When working with large datasets, the `basindelin_big` should be used instead. Flow accumulated is then calculated using function `flowacc` to produce the plots shown below the example code.

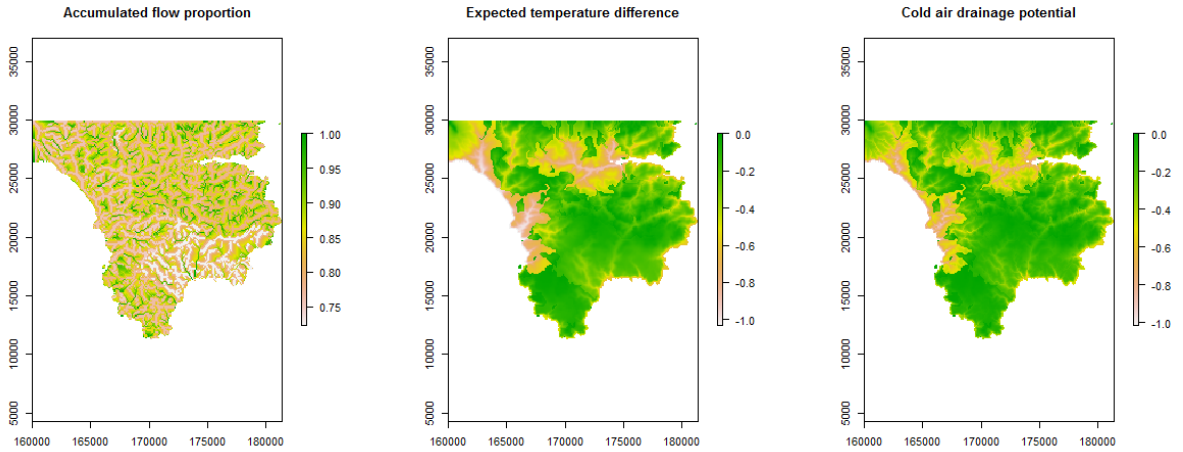
```
# = =   Example of calculating accumulated flow
# delineate basins
basins <- basindelin(dtm100m) # takes a few seconds
# Calculate flow accumulation
fa <- flowacc(dtm100m)
par(mfrow=c(1,2))
plot(basins, main = "Cold air drainage basins")
plot(fa, main = "Accumulated flow (cells)")
```



Once flow accumulation has been calculated, cold air drainage potential can be calculated using function `pcad` as in the example below. Here, because cold air can often flow between shallow basins, those basins separated by a boundary of less than 2 m are first merged using function `basinmerge`. The function `pcad` is then applied. The default is for this function to output the expected temperature difference in degrees C due to both the lapse rate and accumulated flow proportion, but the components parts can also be outputted individually as in the plots below.

```
# = = = Example of how cold air drainage is calculated = = =
# basins <- basindelin(dtm100m) # run if not already run
basins <- basinmerge(dtm100m, basins, 2)
h <- humidityconvert(50, intype = "relative", 20)$specific
fa <- flowacc(dtm100m)
cp1 <- pcad(dtm100m, basins, fa, 20, h)
cp2 <- pcad(dtm100m, basins, fa, 20, h, out = "tempdif")
cp3 <- pcad(dtm100m, basins, fa, 20, h, out = "pflow")
par(mfrow=c(1, 3))
```

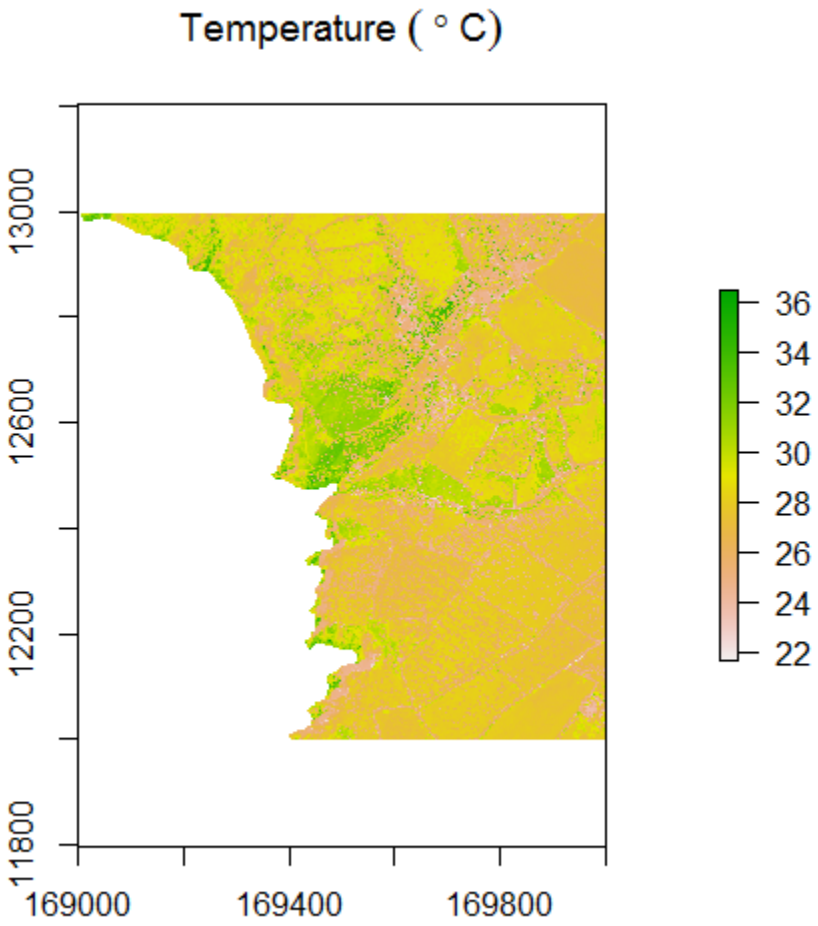
```
plot(cp3, main = "Accumulated flow proportion")
plot(cp2, main = "Expected temperature difference")
plot(cp1, main = "Cold air drainage potential")
```



Running the models

In the example below in which the microclimate model is run, the `fitmicro` function is applied to the data included with the package to derive model coefficients. Net radiation is then calculated from the net short and longwave radiation datasets included with the package, but derived as in the examples above. The `runmicro` function is then used to obtain the anomalies from reference temperatures, and actual temperature derived as reference temperature + the anomaly. The plot below the example is produced by the code.

```
# ============================================================================
# Run microclimate model for 2010-05-24 11:00 (one of the warmest hours)
# ============================================================================
params <- fitmicro(microfitdata)
netrad <- netshort1m - netlong1m
tempanom <- runmicro(params, netrad, wind1m)
tempanom <-if_raster(tempanom, dtm1m) # converts to raster
reftemp <- raster(temp100[,,564])
extent(reftemp) <- extent(dtm1m)
reftemp <- resample(reftemp, dtm1m)
temps <- tempanom + reftemp
plot(temps, main =
     expression(paste("Temperature ",(~degree~C))))
```

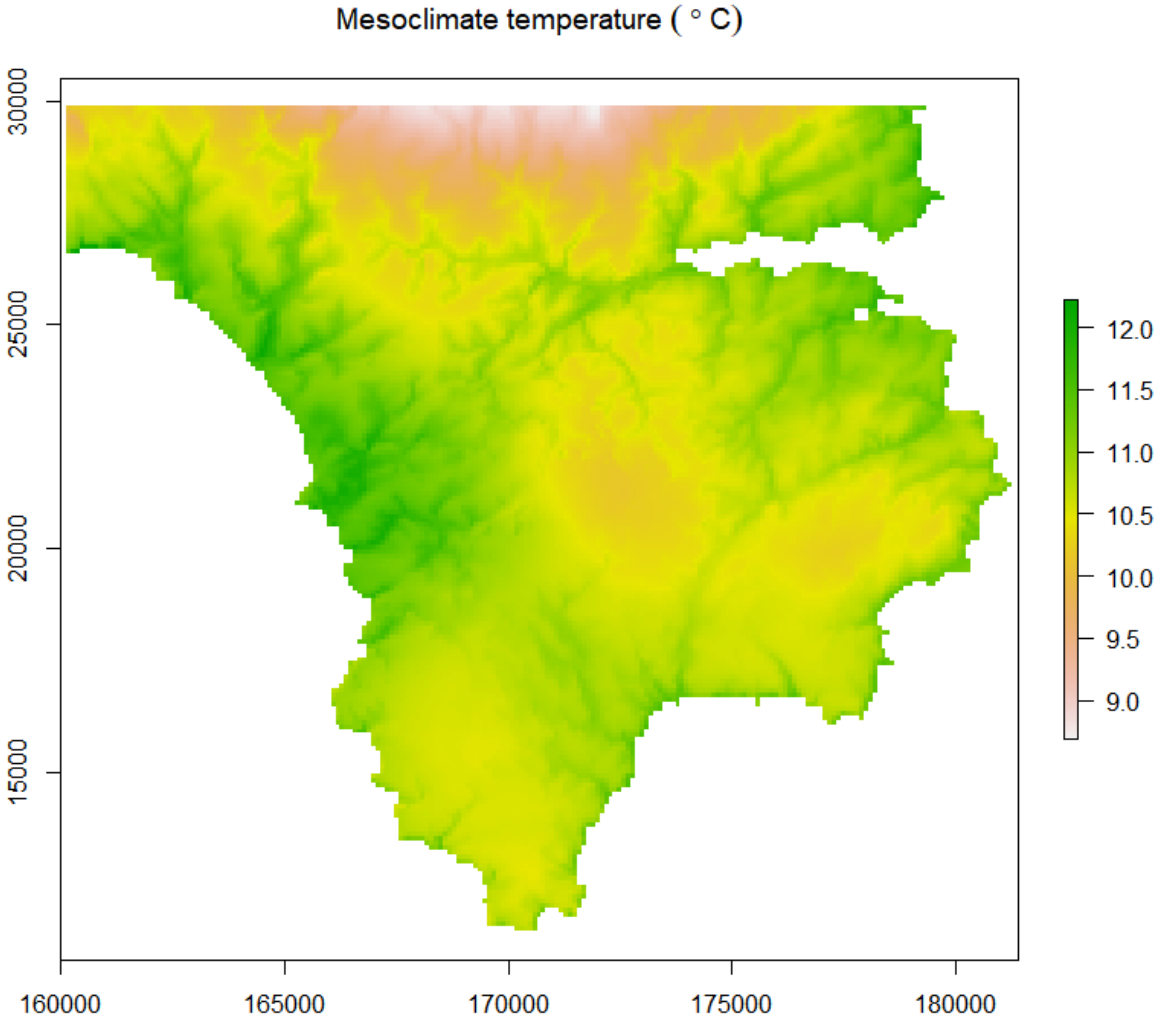


In the example below the mesoclimate model is run from first principles. First, hourly reference temperature data are calculating from daily data included with the package using the `hourlytemp` function. Coastal and elevation effects are then derived using the same approaches as in the examples presented previously. Cold air drainage effects are not calculated, as the model is applied in daylight hours in which cold air drainage would not occur. Radiation and wind are then downscaled as in the examples presented previously. The `fitmicro` function is then used to derive model coefficients, using a dataset of mesoclimate temperature measurements included with the package. The `runmicro` function is then used to calculate temperature anomalies due to radiation and wind. These anomalies are added to the reference temperatures, to which coastal and elevation effects have already been applied, to calculate a gridded dataset of mesoclimate temperatures as shown in the plot below the example.

```
#=====================================================================
# Run mesoclimate model for 2010-05-01 11:00 from first principles
#=====================================================================
# -------------------------
# Resample raster function
# -------------------------
resampleraster <- function(a, ro) {
  r <- raster(a)
  extent(r) <- c(-5.40, -5.00, 49.90, 50.15)
```

```
  crs(r) <- "+init=epsg:4326"
  r <- projectRaster(r, crs = "+init=epsg:27700")
  r <- resample(r, ro)
  as.matrix(r)
}
# --------------------------
# Resample raster: 24 hours
# --------------------------
get24 <- function(a) {
  ao <- array(NA, dim = c(dim(dtm1km)[1:2], 24))
  for (i in 1:24) {
    ai <- a[,,2880 + i]
    ao[,,i] <- resampleraster(ai, dtm1km)
  }
  ao
}
# ----------------------------
# Derive hourly temperatures
# ----------------------------
tmax <- tas[,,121] + dtr[,,121] / 2
tmin <- tas[,,121] - dtr[,,121] / 2
tme <- as.POSIXct(c(0:364) * 24 * 3600, origin="2010-01-01", tz = "GMT")
out <- as.POSIXct(c(0:23) * 3600, origin="2010-05-01", tz = "GMT")
h <- arrayspline(huss, tme, out = out)
p <- arrayspline(pres, tme, out = out)
n <- get24(cfc)
dni <- get24(dnirad)
dif <- get24(difrad)
jd <- julday(2010, 5 , 1)
tc <- h[,,1] * NA
lr <- h[,,1] * NA # Also calculates lapse rate
for (i in 1:19) {
  for (j in 1:22) {
    if (is.na(tmax[i,j]) == F){
      ht <- hourlytemp(jd, em = NA, h[i, j, ],
                       n[i, j, ], p[i, j, ], dni[i, j, ],
                       dif[i, j, ], tmin[i,j], tmax[i,j],
                       50.02, -5.20)
      tc[i, j]<- ht[12]
      lr[i, j] <- lapserate(tc[i, j], h[i, j, 12], p[i, j, 12])
    }
  }
}
# ----------------------------
# Calculate coastal effects
# ----------------------------
sst <- 10.771
dT <- if_raster(sst - tc, dtm1km)
lsw <- if_raster(landsearatios[,,28], dtm100m)  # upwind
# mean, all directions
lsa <- if_raster(apply(landsearatios, c(1, 2), mean), dtm100m)
dTf <- coastalTps(dT, lsw, lsa)
# -----------------------------
# Calculate altitudinal effects
```

```
# ------------------------------
lrr <- if_raster(lr, dtm1km)
lrr <- resample (lrr, dtm100m)
tc <- sst - dTf + lrr * dtm100m
# ------------------------------
# Downscale radiation
# ------------------------------
dni <- resampleraster(dnirad[,,2891], dtm100m)
dif <- resampleraster(difrad[,,2891], dtm100m)
n <- resampleraster(cfc[,,2891], dtm100m)
h <- resample(if_raster(h[,,12], dtm1km), dtm100m)
p <- resample(if_raster(p[,,12], dtm1km), dtm100m)
sv <- skyviewtopo(dtm100m)
netshort <- shortwavetopo(dni, dif, jd, 11, dtm = dtm100m, svf = sv)
netlong <- longwavetopo(h, tc, p, n, sv)
netrad <- netshort - netlong
# ------------------
# Downscale wind
# ------------------
ws <- array(windheight(wind2010$wind10m, 10, 1), dim = c(1, 1, 8760))
wh <- arrayspline(ws, as.POSIXct(wind2010$obs_time), 6,
                  "2010-05-01 11:00")
ws <- windcoef(dtm100m, 270, res = 100) * wh
# ------------------
# Fit and run model (continuous = TRUE)
# ------------------
params <- fitmicro(mesofitdata, continuous = TRUE)
anom <- runmicro(params, netrad, ws, continuous = TRUE)
tc <- tc + anom
plot(mask(tc, dtm100m), main =
     expression(paste("Mesoclimate temperature ",(~degree~C))))
```



Package dependences

- raster (>= 2.5.8)
- rgdal (>= 1.2)
- stringr (>=1.2)
- dplyr (>=0.7)
- sp (>=1.2)
- Rcpp (>=1.0.1)