User's guide to meteoland

Miquel De Cáceres[1,2]

[1]Centre Tecnològic Forestal de Catalunya. Ctra. St. Llorenç de Morunys km 2, 25280, Solsona, Catalonia, Spain
[2]CREAF, Cerdanyola del Vallès, 08193, Spain

September 28, 2016

Contents

1 Short introduction to the package

1.1 Purpose

Reliable meteorological data are a basic requirement for hydrological and ecological studies at the landscape scale. Given the large spatial variation of meteorology over complex terrains, meteorological records from a single weather station are often not representative of entire landscapes. Studies made on multiple sites over a landscape require different meteorological series for each site; and other studies may require meteorological data series for all grid cells of a landscape, in a continuous way. In these cases, spatial correlation between the meteorology series of different sites or cells must be taken into account. For example, the sequence of days with rain of contiguous cells will normally be the same or very similar, even if precipitation amounts differ. Finally, studies addressing the impacts of climate change on forests and landscapes require downscaling coarse-scale predictions of global or regional climate models to the landscape scale. When downscaling predictions for several locations in a landscape, spatial correlation of downscaled predictions is also important.

With the aim to assist research of climatic impacts on forests, the R package `meteoland` provides utilities to estimate daily weather variables at any position in complex terrains:

1. Spatial interpolation of daily weather records from meteorological stations.
2. Statistical downscaling of coarse-scale meteorological data (coming from regional climate models or re-analyses) to the landscape scale.

1.2 Meteorological variables

Package `meteoland` assists in the estimation of the following meteorological variables over lanscapes (units in parentheses):

- `DOY`: Day of the year (Julian day).
- `MeanTemperature`: Mean daily temperature (in degrees Celsius).
- `MinTemperature`: Minimum daily temperature (in degrees Celsius).
- `MaxTemperature`: Maximum daily temperature (in degrees Celsius).
- `Precipitation`: Daily precipitation (in mm of water).
- `MeanRelativeHumidity`: Mean daily relative humidity (in percent).
- `MinRelativeHumidity`: Minimum daily relative humidity (in percent).

- `MaxRelativeHumidity`: Maximum daily relative humidity (in percent).
- `Radiation`: Incoming radiation (in MJ/m2).
- `WindSpeed`: Wind speed (in m/s).
- `WindDirection`: Wind direction (in degrees from North).
- `PET`: Potential evapo-transpiration (in mm of water).

Although internally it uses specific units, `meteoland` allows reading and writing meteology data in different units and formats.

1.3 Spatial variation of meteorology and topography

Package `meteoland` deals with two kinds of spatial structures: individual points and grids. The package includes four S4 spatial classes, which are defined as children of classes in package `sp`. Two classes are defined to represent the variation of topographic features (i.e., elevation, slope and aspect) over space:

- `SpatialPointsTopography` extends `SpatialPointsDataFrame` and represents the topographic features of a set of points in a landscape.

```
Class "SpatialPointsTopography" [package "meteoland"]

Slots:

Name:         data  coords.nrs      coords        bbox proj4string
Class:  data.frame     numeric      matrix      matrix         CRS

Extends:
Class "SpatialPointsDataFrame", directly
Class "SpatialPoints", by class "SpatialPointsDataFrame", distance 2
Class "Spatial", by class "SpatialPointsDataFrame", distance 3
```

- `SpatialGridTopography` extends `SpatialGridDataFrame` and represents the continuous variation of topography over a grid.

```
Class "SpatialGridTopography" [package "meteoland"]

Slots:

Name:           data         grid         bbox  proj4string
Class:    data.frame GridTopology       matrix          CRS
```

```
Extends:
Class "SpatialGridDataFrame", directly
Class "SpatialGrid", by class "SpatialGridDataFrame", distance 2
Class "Spatial", by class "SpatialGridDataFrame", distance 3
```

Although they have the same slots as their parent S4 classes, the data frames in `SpatialPointsTopography` and `SpatialGridTopography` have only three variables: '**elevation**' (in meters), '**slope**' (in degrees) and '**aspect**' (in degrees from North).

Two analogous spatial classes are used to represent the variation of daily meteorology over space:

- `SpatialPointsMeteorology` extends `SpatialPoints` and represents daily meteorology series for a set of points in a landscape.

  ```
  Class "SpatialPointsMeteorology" [package "meteoland"]

  Slots:

  Name:        dates        data      coords        bbox proj4string
  Class:        Date      vector      matrix      matrix         CRS

  Extends:
  Class "SpatialPoints", directly
  Class "Spatial", by class "SpatialPoints", distance 2
  ```

- `SpatialGridMeteorology` extends `SpatialGrid` and represents the continuous variation of daily meteorology over a grid of cells.

  ```
  Class "SpatialGridMeteorology" [package "meteoland"]

  Slots:

  Name:          dates          data          grid          bbox
  Class:          Date        vector GridTopology        matrix

  Name:    proj4string
  Class:           CRS

  Extends:
  Class "SpatialGrid", directly
  Class "Spatial", by class "SpatialGrid", distance 2
  ```

In addition to their corresponding inherited slots, `SpatialPointsMeteorology` and `SpatialGridMeteorology` have two additional slots: '`dates`' (a vector of days specifying a time period), and '`data`' (a vector of data frames with the meteorological data). Although both have a '`data`' with data frames, meteorological data is in different form in each class. In objects of `SpatialPointsMeteorology`, there is one data frame for each point with variables in columns and dates in rows. In objects of `SpatialGrid-Meteorology`, each data frames describes the meteorology over the grid for one day, with grid cells in rows and variables in columns.

1.4 Reading and writing meteorological data

1.4.1 Point meteorology

Objects of class `SpatialPointsMeteorology` are stored using one **ascii** data file for each spatial point. Package `meteoland` provides four input/output functions for point meteorology:

- Function `readmeteorologypoint()` reads the meteorological data stored in one **ascii** data file and returns a data frame.
- Function `writemeteorologypoint()` writes the meteorological data of a single point as an **ascii** file in the file system.
- Function `readmeteorologypointfiles()` reads several **ascii** files and returns an object of class `SpatialPointsMeteorology`.
- Functions `writemeteorologypointfiles()` writes several **ascii** files in the disk, one per spatial point. Metadata (i.e. the spatial coordinates of each point and the corresponding file path) is stored in an additional file.

1.4.2 Grid meteorology

Objects of class `SpatialGridMeteorology` are stored using one **netCDF** file per day, which also contains the date and spatial projection. The following functions are available for input/output of grid meteorology:

- Function `readmeteorologygrid()` reads the meteorological data stored in one **netCDF** file and returns a `SpatialGridDataframe`.
- Function `writemeteorologygrid()` writes the meteorological data of a single date as a **netCDF** file.
- Function `readmeteorologygridfiles()` reads several **netCDF** files and returns an object of class `SpatialGridMeteorology`.
- Function `readmeteorologygridcells()` reads several **netCDF** files and returns an object of class `SpatialPointMeteorology` with the meteorological data of a set of specified grid cells.

- Functions `writemeteorologygridfiles()` writes several **netCDF** files in the disk, one per date. Metadata (i.e. the dates and their corresponding file path) is stored in an additional file.

1.5 Meteorology estimation functions

1.5.1 Spatial interpolation

Package `meteoland` provides two functions for interpolating meteorological data (i.e., one for each data structure):

- Function `interpolationpoints()` interpolates weather for a set of locations given in `SpatialPointsTopography` and returns an object of class `SpatialPointsMeteorology`.
- Function `interpolationgrid()` interpolates weather for a whole grid specified in `SpatialGridTopography` and returns an object of class `SpatialGridMeteorology`.

Both functions require an object of class `'MeteorologyInterpolationData'`, which contains the X-Y coordinates, the meteorological data and topography of a set of weather stations as well as weather interpolation parameters.

```
Class "MeteorologyInterpolationData" [package "meteoland"]

Slots:

Name:                   coords                  elevation
Class:                  matrix                    numeric

Name:                    slope                     aspect
Class:                 numeric                    numeric

Name:           MinTemperature             MaxTemperature
Class:                  matrix                     matrix

Name:     SmoothedPrecipitation             Precipitation
Class:                  matrix                     matrix

Name:  SmoothedTemperatureRange          RelativeHumidity
Class:                  matrix                     matrix

Name:                Radiation                  WindSpeed
Class:                     ANY                        ANY
```

```
Name:            WindDirection          WindFields
Class:                     ANY                 ANY

Name:                  WFIndex            WFFactor
Class:                     ANY                 ANY

Name:                   params               dates
Class:                    list                Date

Name:                     bbox         proj4string
Class:                  matrix                 CRS

Extends:
Class "MeteorologyProcedureData", directly
Class "Spatial", by class "MeteorologyProcedureData", distance 2
```

When calling functions `interpolationpoints()` or `interpolationgrid()`, the user may require interpolation outputs to be written into the file system, instead of being returned in memory. If `interpolationpoints()` is called with `export = TRUE`, the function will write the data frame produced for each point into an **ascii** text file. If `interpolationgrid()` is called with `export = TRUE`, the function will write an **netCDF** file for each day. Metadata files will also be written, so that results can later be loaded in memory.

1.5.2 Statistical downscaling

Analogously to interpolation, two functions are available for statistical downscaling of coarse-scale meteorological data (i.e., one function for each data structure):

- Function `downscalingpoints()` performs statistical downscaling of coarse-scale weather data on a set of locations and it returns an object of class `SpatialPointsMeteorology` containing corrected weather predictions.
- Function `downscalinggrid()` performs statistical downscaling of coarse-scale weather data over a grid and it returns an object of class `SpatialPointsMeteorology` containing corrected weather predictions.

Donwscaling functions require an object of class `'MeteorologyDownscalingData'`, which contains the X-Y coordinates and the coarse-scale meteorological data to be downscaled, which includes a historical (reference) period and future (projected) period:

```
Class "MeteorologyDownscalingData" [package "meteoland"]
```

```
Slots:

Name:         coords historicdata   futuredata       params
Class:        matrix          ANY          ANY         list

Name:          dates         bbox  proj4string
Class:          Date       matrix          CRS

Extends:
Class "MeteorologyProcedureData", directly
Class "Spatial", by class "MeteorologyProcedureData", distance 2
```

The historical (reference) period is compared with fine-scale meteorological data of the same period, and the routine uses this information to correct the future (projected) period. Therefore, apart from the 'MeteorologyDownscalingData' object, the dowscaling functions require fine-scale historical data (for a set of spatial points or a grid). Normally, these data will be the result of spatial interpolation.

As before, when calling functions `downscalingpoints()` or `downscalinggrid()`, the user may require the outputs to be written into the file system, instead of being returned in memory. If `downscalingpoints()` is called with `export = TRUE`, the function will write the data frame produced for each point into an **ascii** text file. If `downscalinggrid()` is called with `export = TRUE`, the function will write a **netCDF** file for each day. Metadata files will also be written, so that results can later be loaded in memory.

2 Spatial interpolation of weather records

2.1 Overview

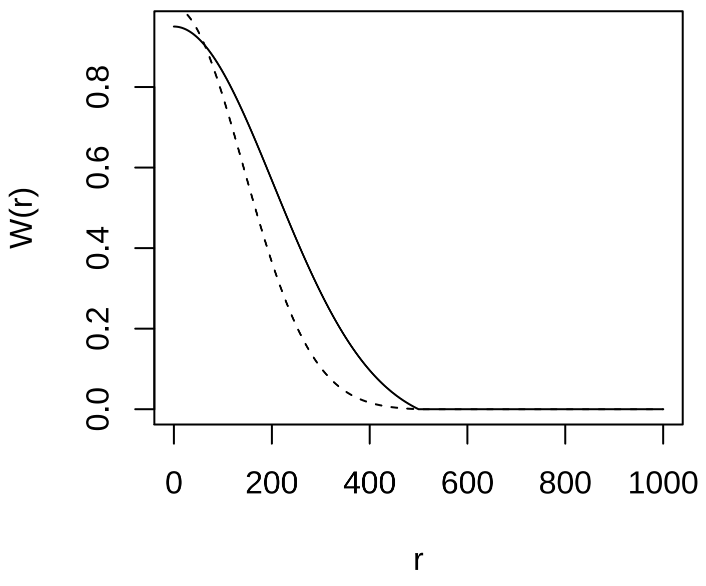
Landscape research studies conducted for historical periods can be perfomed using meteorological records obtained from surface weather stations of the area under study. For any target point, minimum temperature, maximum temperature and precipitation are interpolated from weather records using truncated Gaussian filters, while accounting for the relationship between these variables and elevation (Thornton et al. 1997). Relative humidity can be either interpolated (in fact, water vapour pressure is the variable being interpolated) or predicted from temperature estimates, depending on whether it has been measured in weather stations or not. Potential (i.e. top-of-atmosphere) solar radiation is estimated taking into account latitude, seasonality, aspect and slope, following Granier & Ohmura (1968). Potential solar radiation is then corrected to account for atmosphere transmittance using the predictions of temperature range, relative humidity and precipitation (Thornton & Running 1999). Finally, the wind vector (wind direction and wind speed) is interpolated by using weather station records and static wind fields. In the following subsections we detail the general algorithm used to obtain interpolation weights and the interpolation/estimation procedure for each variable.

2.2 Interpolation weights

Thornton et al. (1997) suggested interpolating meteorological data using a truncated Gaussian filter. Its form with respect to a central point p is:

$$W(r) = e^{-\alpha \cdot (r/R_p)^2} - e^{-\alpha} \tag{1}$$

if $r \leq R_p$ and $W(r) = 0$ otherwise. Here r is the radial distance from p, R_p is the truncation distance and α is the shape parameter. The spatial convolution of this filter with a set of weather station locations results, for each target point, in a vector of weights associated with observations. The following figure illustrates the Gaussian filter for $R_p = 500$ and either $\alpha = 3.0$ (continuous line) or $\alpha = 6.25$ (dashed line):



R_p is estimated so that it has lower values in data-rich regions and is increased in data-poor regions. We require the user to specify N, the average number of observations to be included for each target point. R_p is then varied as a smooth function of the local density in such a way that this average is achieved over the spatial domain. Estimation of R_p is as follows:

1. A user-specified value is used to initialize R_p.
2. Interpolation weights W_i are calculated for all $i = (1, ..., n)$ stations, and the local station density is calculated as:

$$D_p = \frac{\sum_{i=1}^{n} (W_i/\hat{W})}{\pi \cdot R_p^2} \tag{2}$$

   where $\hat{W}$ is the average weight over the untruncated region of the kernel, calculated as:

$$\hat{W} = \left(\frac{1 - e^{-\alpha}}{\alpha}\right) - e^{-\alpha} \tag{3}$$

3. A new R_p value is calculated as a function of N and D_p, as:

$$R_p = \sqrt{\frac{N^*}{D_p \cdot \pi}} \tag{4}$$

   where $N^* = 2N$ for the first $I - 1$ iterations, and $N^* = N$ for the final iteration.

4. The new R_p is substituted in step (2) and steps (2-4) are iterated a specified number of times I. The final R_p value is used to generate interpolation weights W_i.

Thornton et al. (1997) suggested to use this algorithm only once per point (and variable to be estimated), but since missing meteorological values can occur only in some days, we apply the algorithm for each target point and day. The interpolation method for a given set of observations is defined by four parameters R, I, N and α. Following Thornton et al. (1997), we set $R = 140000$ meters and $I = 3$ by default. The other parameters depend on the variable to be interpolated.

2.3 Temperature

Predictions for minimum temperature and maximum temperature are done in the same way, so we refer to a general variable T. We focus on the prediction of T_p, the temperature at a single target point p and for a single day, based on observations T_i and interpolation weights W_i for the $i = (1, ..., n)$ weather stations. Prediction of T_p requires a correction for the effects of elevation differences between observation points $z_1, ..., z_n$ and the prediction point z_p. Thornton et al. (1997) established the relationship between elevation and temperature using transformed variables (temporal or spatial moving window averages) for temperature and elevation, instead of the original variables, but we did not implement this feature here. A weighted least-squares regression is used to assess the relationship between temperature and elevation. Instead of regressing z_i on T_i, the independent variable is the difference in elevations associated with a pair of stations, and the dependent variable is the corresponding difference in temperatures. This gives a regression of the form:

$$(T_1 - T_2) = \beta_0 + \beta_1 \cdot (z_1 - z_2) \tag{5}$$

where subscripts 1 and 2 indicate the two stations of a pair and β_0 and β_1 are the regression coefficients. Regression is performed using all possible pairs of stations and the regression weight associated with each point is the product of the interpolation weights associated with the stations in a pair. The temperature for the target point, T_p is finally predicted as follows:

$$T_p = \frac{\sum_{i=1}^{n} W_i \cdot (T_i + \beta_0 + \beta_1 \cdot (z_p - z_i))}{\sum_{i=1}^{n} W_i} \tag{6}$$

where z_p is the elevation of the target point and z_i is the elevation of the weather station.

2.4 Relative humidity

Relative humidity is a parameter not always recorded in weather stations. When input station weather data does not include relative humidity, `medfate` allows estimating it directly from minimum and maximum temperature (Thornton et al. 1997). Assuming that minimum daily air temperature $T_{min,p}$ at the target point is a good surrogate of dew-point temperature $T_{d,p}$ (i.e. $T_{d,p} = T_{min,p}$; note that this assumption may not be valid in arid climates), one can estimate actual vapor pressure e_p (in Pa) as:

$$e_p = 610.78 \cdot e^{\left(\frac{17.269 \cdot T_{d,p}}{237.3 + T_{d,p}}\right)} \tag{7}$$

and saturated vapor pressure $e_{s,p}$ (in Pa) as:

$$e_{s,p} = 610.78 \cdot e^{\left(\frac{17.269 \cdot T_{a,p}}{237.3 + T_{a,p}}\right)} \tag{8}$$

where $T_{a,p} = 0.606 \cdot T_{max,p} + 0.394 \cdot T_{min,p}$ is the average daily temperature. Finally, relative humidity RH_p (in percentage) is calculated as:

$$RH_p = 100 \cdot \frac{e_p}{e_{s,p}} \tag{9}$$

When relative humidity has been measured at weather stations, interpolation should be preferred to estimation from minimum and maximum temperature. However, because relative humidity cannot be corrected for elevation differences between the station and the target climatic grid cell, relative humidity RH_i of each weather station i has to be converted to dew-point temperature $T_{d,i}$ before interpolation (Tymstra et al. 2010). To obtain the dew-point temperature one first needs to calculate vapor pressure:

$$e_i = e_{s,i} \cdot (RH_i/100) \tag{10}$$

where $e_{s,i}$ is the saturated water vapor pressure of station i, calculated as indicated above. Then, dew-point temperature of station i is obtained from:

$$T_{d,i} = \frac{237.3 \cdot \ln(e_i/610.78)}{17.269 - \ln(e_i/610.78)} \tag{11}$$

As for temperature, a weighted least-squares regression is used to assess the relationship between dew-point temperature and elevation. Again, the independent variable is the difference in elevations associated with a pair of stations, and the dependent variable is the corresponding difference in dew-point temperature. This gives a regression of the form:

$$(T_{d,1} - T_{d,2}) = \beta_0 + \beta_1 \cdot (z_1 - z_2) \tag{12}$$

where subscripts 1 and 2 indicate the two stations of a pair and β_0 and β_1 are the regression coefficients. The dew-point temperature for the target point, $T_{d,p}$ is predicted as:

$$T_{d,p} = \frac{\sum_{i=1}^{n} W_i \cdot (T_{d,i} + \beta_0 + \beta_1 \cdot (z_p - z_i))}{\sum_{i=1}^{n} W_i} \tag{13}$$

where z_p is the elevation of the target point and z_i is the elevation of the weather station. From the interpolated dew-point temperature one can obtain actual vapour pressure e_p and, together with saturated vapour pressure at point p, one calculates relative humidity as indicated above. If saturated vapour pressure is referred to average temperature $T_{a,p}$, then relative humidity is average relative humidity $RH_{a,p}$. If, instead, one refers saturated vapour pressure to minimum and maximum daily temperatures one obtains, respectively, maximum and minimum relative humidity values ($RH_{max,p}$, $RH_{min,p}$). Because of the regression step, one must check that the predicted relative humidity value stays within the physical limits 0% and 100%.

2.5 Precipitation

Predictions of precipitation are complicated by the need to predict both daily occurrence and, conditioned on this, daily total precipitation. Thornton et al. (1997) define a binomial predictor of spatial precipitation occurrence as a function of the weighted occurrence at surrounding stations. The precipitation occurrence probability POP_p is:

$$POP_p = \frac{\sum_{i=1}^{n} W_{o,i} \cdot PO_i}{\sum_{i=1}^{n} W_{o,i}} \tag{14}$$

where PO_i is the binomial precipitation occurrence in station i (i.e., $PO_i = 0$ if $P_i = 0$ and $PO_i = 1$ if $P_i > 0$) and $W_{o,i}$ is the interpolation weight for precipitation occurrence. Once POP_p is calculated, then precipitation occurs if POP_p is smaller than a critical value (i.e. $PO_p = 1$ if $POP_p < POP_{crit}$ and $PO_p = 0$ otherwise). Conditional on precipitation occurrence we calculate the prediction of daily total precipitation, P_p. Like with temperature, Thornton et al. (1997) established the relationship between elevation and precipitation using transformed variables (temporal or spatial moving window averages) for precipitation and elevation. Following their results, we transform precipitation values using a temporal window with side of 5 days. Weighted least-squares, where the weight associated with each point is the product of the interpolation weights associated with the stations in a pair, is used to account for elevation effects on precipitation. Unlike Thornton et al. (1997), who used the same set of interpolation weights (i.e. $W_{o,i}$) for precipitation occurrence and regression, we use a second set of interpolation weights $W_{r,i}$ for the calculation of regression weights. The dependent variable in the regression function is defined as the normalized difference of the precipitation observations P_i for any given pair of stations:

$$\left(\frac{P_1 - P_2}{P_1 + P_2}\right) = \beta_0 + \beta_1 \cdot (z_1 - z_2) \tag{15}$$

To obtain the predicted daily total P_p we use the following equation:

$$P_p = \frac{\sum_{i=1}^{n} W_{o,i} \cdot P_i \cdot PO_i \cdot \left(\frac{1+f}{1-f}\right)}{\sum_{i=1}^{n} W_{o,i} \cdot PO_i} \tag{16}$$

where $f = \beta_0 + \beta_1 \cdot (z_p - z_i)$. Note the usage of interpolation weight $W_{o,i}$ (and not $W_{r,i}$). The form of prediction requires that $|f| < 1$. A parameter f_{max} (with default $f_{max} = 0.95$) is introduced to force $|f| = f_{max}$ whenever $|f| > f_{max}$.

2.6 Radiation

Incident daily solar radiation is not interpolated, but estimated from topography and measurements of temperature, humidity and precipitation.

Potential solar radiation is estimated from latitude, aspect and slope according to Granier & Ohmura (1968). In particular, instant potential solar radiation $R_{pot,s}$ is calculated as:

$$\begin{aligned} R_{pot,s} &= I_0[(\sin\phi\cos H)(-\cos A\sin Z_x) - \sin H(\sin A\sin Z_x) \\ &\quad +(\cos\phi\cos H)\cos Z_x]\cos\delta \\ &\quad +[\cos\phi(\cos A\sin Z_x) + \sin\phi\cos Z_x]\sin\delta \end{aligned} \tag{17}$$

where ϕ is the latitude, H is the hour angle measured from solar noon, positively towards the west, A is the azimuth of the slope (aspect) and Z_x is the zenith angle of the vector normal to the slope (equal to the slope angle) and δ is the sun's declination, which can be derived from the day of the year J (Julian day):

$$\delta = 0.4093 \cdot \sin(2 \cdot \pi \cdot (J/365) - 1.405) \tag{18}$$

Daily potential radiation, R_{pot} is calculated by integrating instant potential radiation over the day between sunrise (sr) and sunset (ss), using 20 min (i.e. 1200 sec) intervals:

$$R_{pot} = \sum_{s=sr}^{ss} 1200 \cdot R_{pot,s} \tag{19}$$

Improving the method proposed in Thornton et al. (1997), Thornton & Running (1999) calculate incident daily total solar radiation R_g as:

$$R_g = R_{pot} \cdot T_{t,max} \cdot T_{f,max} \tag{20}$$

where $T_{t,max}$ is the maximum (cloud-free) daily total transmittance and $T_{f,max}$ is the proportion of $T_{t,max}$ realized on a given day (cloud correction). The maximum daily total transmittance $T_{t,max}$ is estimated as:

$$T_{t,max} = \left[\frac{\sum_{s=sr}^{ss} R_{pot,s} \cdot \tau^{(P_z/P_0) \cdot m_\theta}}{\sum_{s=sr}^{ss} R_{pot,s}} \right] + (\alpha_{e_p} \cdot e_p) \tag{21}$$

where $\tau = 0.87$ is the instantaneous transmittance at sea level, at nadir, for a dry atmosphere; e_p is the actual water vapor pressure (in Pa), estimated as explained before; $\alpha_{e_p} = -6.1 \cdot 10^{-5} Pa^{-1}$ is a parameter describing the effect of vapour pressure on $T_{t,max}$; $m_\theta = 1/\cos\theta$ is the optical air mass at solar zenith angle $\theta = \sin\phi \cdot \sin\delta + \cos\phi \cdot \cos\delta \cdot \cos H$; and P_z/P_0 is the ratio between air pressure at elevation z_p and air pressure at the sea level, calculated as:

$$(P_z/P_0) = (1.0 - 2.2569 \cdot 10^{-5} \cdot z_p)^{5.2553} \tag{22}$$

In turn, $T_{f,max}$ was empirically related to $\Delta T = T_{\max} - T_{\min}$, the difference between maximum and minimum temperatures for the target point:

$$T_{f,max} = 1.0 - 0.9 \cdot e^{-B \cdot \Delta T^C} \tag{23}$$

being $C = 1.5$ and B calculated from:

$$B = b_0 + b_1 \cdot e^{-b_2 \cdot \hat{\Delta T}} \tag{24}$$

with $b_0 = 0.031$, $b_1 = 0.201$ and $b_2 = 0.185$. In this last equation, $\hat{\Delta T}$ is a 30-day moving average for the temperature range ΔT. For computational reasons, we do not estimate $\hat{\Delta T}$ from the 30-day moving window average of predicted ΔT values, but from the interpolation of pre-calculated $\hat{\Delta T}$ values in weather stations. On wet days (i.e. if $P_p > 0$) the estimation of $T_{f,max}$ is multiplied by a factor of 0.75 to account for clouds.

2.7 Wind

2.7.1 Simple wind interpolation

Simple wind interpolation for a target point p is as follows. Let $\mathbf{v}_i$ be the wind vector in weather station i. $\mathbf{v}_i$ is initially expressed using polar coordinates. Indeed, we have u_i and θ_i, the wind speed and wind direction, respectively. If we express $\mathbf{v}_i$ in cartesian coordinates we have:

$$x_i = u_i \cdot \sin(\theta_i) \quad y_i = u_i \cdot \cos(\theta_i) \tag{25}$$

The predicted wind vector $\mathbf{v}_p$ is the weighted average of the wind vectors $\{\mathbf{v}_i\}$ $i = (1, ..., n)$ predicted for point p using the interpolation weights W_i determined from the truncated Gaussian filter:

$$x_p = \frac{\sum_{i=1}^{n} W_i \cdot x_i}{\sum_{i=1}^{n} W_i} \quad y_p = \frac{\sum_{i=1}^{n} W_i \cdot y_i}{\sum_{i=1}^{n} W_i} \tag{26}$$

The polar coordinates of the predicted wind vector $\mathbf{v}_p$ are:

$$u_p = \sqrt{x_p^2 + y_p^2} \quad \theta_p = \tan^{-1}(x_p/y_p) \tag{27}$$

2.7.2 Interpolation using wind fields

More precise wind interpolation requires a set of static wind fields covering the landscape of interest. Each of these wind fields has been calculated assuming a domain-level combination of wind speed and wind direction. The set of domain-level combinations should cover all possible winds in the landscape under study. For example, one could decide to include the combinations of eight different wind directions (i.e., N, NE, E, SE, ...) and three wind speed classes. The wind estimation of a given target point depends on both the wind observations at weather stations and these static wind fields.

In a given day (and before processing target points) we begin by identifying, for each weather station $i = (1, ..., n)$, the wind field m_i corresponding to a minimum difference between the observed wind vector $\mathbf{v}_i$ and the wind vector of the station in the wind field (i.e., minimum distance between the corresponding cartesian coordinates). The set of wind fields $\{m_i\}$ $i = (1, ..., n)$ chosen for each weather station conform the information for wind interpolation in a given day.

Actual wind interpolation details for a target point p are as follows. We first draw for each $i = (1, ..., n)$ the wind vector $\mathbf{v}_{m_i,p}$ corresponding to the location of the target point p in wind fields m_i. Let $u_{m_i,p}$ and $\theta_{m_i,p}$ be the wind speed and wind direction of $\mathbf{v}_{m_i,p}$, respectively. The cartesian coordinates of $\mathbf{v}_{m_i,p}$ are:

$$x_{m_i,p} = u_{m_i,p} \cdot \sin(\theta_{m_i,p}) \quad y_{m_i,p} = u_{m_i,p} \cdot \cos(\theta_{m_i,p}) \tag{28}$$

The predicted wind vector $\mathbf{v}_p$ is the weighted average of the wind vectors $\{\mathbf{v}_{m_i,p}\}$ $i = (1, ..., n)$ predicted for point p using the interpolation weights W_i determined from the truncated Gaussian filter:

$$x_p = \frac{\sum_{i=1}^{n} W_i \cdot x_{m_i,p}}{\sum_{i=1}^{n} W_i} \quad y_p = \frac{\sum_{i=1}^{n} W_i \cdot y_{m_i,p}}{\sum_{i=1}^{n} W_i} \tag{29}$$

The polar coordinates of the predicted wind vector $\mathbf{v}_p$ are found as before.

3 Statistical downscaling of coarse-scale weather data

Statistical downscaling is necessary when meteorological data is available at a spatial scale that is too coarse for landscape-level analysis. This is usually the case when taking predictions from global or regional climate models. The general idea of downscaling to the landscape level is that a fine-scale meteorological series is to be compared to coarse-scale series for the a historical (reference) period. The result of this comparison can be used to correct coarse-scale meteorological series for other periods (normally future projections). In the case of most meteorological variables, the comparison consists in calculating a bias using the reference period and applying this bias to the future period. However, in the case of precipitation a different procedure is needed.

3.1 Precipitation

TO DO: Describe precipitation quantile mapping (Gudmundsson et al. 2012).

3.2 Other variables

Other meteorological variables (temperature, relative humidity, radiation and wind) are downscaled by simple bias correction applied monthly. Let x be the meteorological variable observed for the target point in the reference period, and u the same variable, but estimated at the coarse-scale for the reference period. Month m bias of the coarse-scale data (θ_m) is the average of the difference between u and x over all n_m days of month m in the reference period:

$$\theta_m = \sum_{i=1}^{n_m} (u_i - x_i)/n_m \tag{30}$$

Month biases are used to correct the coarse-scale value of day i in the projected period using:

$$c_i = u_i - \theta_{m(i)} \tag{31}$$

where $m(i)$ is the month of day i.

Mean temperature, minimum temperature, maximum temperature, radiation and wind speed are downscaled using this general bias correction procedure. In the case of relative humidity, values are transformed to specific humidity, bias correction is applied to this variable and the result is back-transformed to relative humidity. Since historic wind data is often not available, if wind speed data is missing the coarse-scale wind estimate is taken directly without correction.

4 Miscellaneous functions

4.1 Extraction of climatic netCDF data

NetCDF is a standard data format for meteorological data. In particular, this format is used to store the predictions of global and regional climate models. Function `extractNetCDF()` parses a set of **NetCDFs** and extracts the daily meteorological data of a landscape boundary box for use within `meteoland`. The function first identifies which cells in NetCDF data should be extracted (according to the input boundary box), and the overall period. For each cell to be processed, the function loops over all files (which can describe different variables and time periods) and extracts the corresponding data. The function transforms units to the units used in meteoland. If specific humidity and mean temperature are available, the function also calculates mean relative humidity.

4.2 Potential evapo-transpiration

Package `meteoland` provides a supplementary function to calculate potential evapo-transpiration (PET) using Penman's formulation (Penman 1948; 1956). The code was taken from package `'Evapotranspiration'`, which follows McMahon et al. (2013). Penman (1948) proposed an equation to calculate potential evaporation that combined an energy equation based on net incoming radiation with an aerodynamic approach. The Penman or Penman combination equation is:

$$E_{pot} = \frac{\Delta}{\Delta + \gamma} \cdot \frac{R_n}{\lambda} + \frac{\lambda}{\Delta + \lambda} \cdot E_a \qquad (32)$$

where PET is the daily potential evaporation (in $mm \cdot day^{-1}$) from a saturated surface, R_n is the daily radiation to the evaporating surface (in $MJ \cdot m^{-2} \cdot day^{-1}$), Δ is the slope of the vapour pressure curve ($kPa \cdot °C^{-1}$) at air temperature, γ is the psychrometric constant ($kPa \cdot °C^{-1}$), and λ is the latent heat of vaporization (in $MJ \cdot kg^{-1}$). E_a (in $mm \cdot day^{-1}$) is a function of the average daily windspeed (u, in $m \cdot s^{-1}$), and vapour pressure deficit (D, in kPa):

$$E_a = f(u) \cdot D = f(u) \cdot (v_a^* - v_a) \qquad (33)$$

where v_a^* is the saturation vapour pressure (kPa) and v_a the actual vapour pressure (kPa) and $f(u)$ is a function of wind speed, for which there are two alternatives (Penman 1948; 1956):

$$f(u) = 1.313 + 1.381 \cdot u \quad (34)$$

$$f(u) = 2.626 + 1.381 \cdot u \quad (35)$$

If wind speed is not available, an alternative formulation for E_{pot} is used as an approximation (Valiantzas 2006):

$$PET \simeq 0.047 \cdot R_s \cdot (T_a + 9.5)^{0.5} - 2.4 \cdot (\frac{R_s}{R_a})^2 + 0.09 \cdot (T_a - 20) \cdot (1 - \frac{RH_{mean}}{100}) \quad (36)$$

where R_s is the incoming solar radiation (in $MJ \cdot m^{-2} \cdot day^{-1}$), T_a is the mean daily temperature (in °C), R_a is the extraterrestrial solar radiation (in $MJ \cdot m^{-2} \cdot day^{-1}$) and RH_{mean} is the mean relative humidity (in percent). Details about the calculation of all these quantities can be found in McMahon et al. (2013).

PET is normally calculated after meteorological data have been interpolated (i.e. within functions `interpolationpoints()` and `interpolationgrid()`) or downscaled (i.e. within functions `downscalingpoints()` and `downscalinggrid()`), but PET series can also be calculated for a single point using function `penmanpoint()`. For other formulations of PET, the reader is referred to package '`Evapotranspiration`'.

4.3 Obtaining static wind fields

External software is necessary to calculate the set of wind fields for the study area under different domain-level average situations. For this we recommend using **WindNinja**, a computer program that calculates spatially varying wind fields for wildland fire applications. WindNinja allows simulating the spatial variation of wind for one instant in time. It was developed to be used by emergency responders within their typical operational constraints of fast simulation times (seconds), low CPU requirements (single processor laptops), and low technical expertise. WindNinja is typically run on domain sizes up to 50 kilometers by 50 kilometers and at resolutions of around 100 meters. The program is free and can be downloaded from www.firemodels.org.

The inputs for a basic run of WindNinja are an elevation data file for the study area, a domain-averaged input wind speed and direction and a specification of the dominant vegetation in the area. In order to obtain a set of pre-computed rasters of wind direction and speed, we suggest the following procedure:

- Export the elevation raster of the study area in one of the file formats accepted by WindNinja ('.asc', '.tif' or '.img'). In the case of a large study area (e.g. $>$ 100 x 100 km) one should run WindNinja in subsets of the area and then integrate the results (e.g., Sanjuan et al. 2014).

- Run WindNinja, using the elevation of the study area, for all combinations of wind direction and wind speed class (for each wind speed class an mean class value has to be chosen). Several combinations of domain-level wind speed and wind direction can be specified for a single run, and the program can also be run in batch mode.
- Read raster files created by WindNinja (a wind speed file, a wind direction file) for each combination of domain-level wind speed and direction.

Function `readWindNinjaOutput()` can be used to conduct this last step. The function allows parsing all the ASCII raster files produced by WindNinja for combinations of wind direction (e.g., 0, 45, 90, 135, 180, 225, 270 and 315 degrees) and wind speed (e.g., 2, 7 and 10 m/s). The function returns a list with the following elements:

- The vector of domain-level wind directions corresponding to WindNinja Runs
- The vector of domain-level wind speed corresponding to WindNinja Runs
- An object `SpatialGridDataFrame` containing the wind directions (in degrees from North) for all WindNinja runs.
- An object `SpatialGridDataFrame` containing the wind speeds (in m/s) for all WindNinja runs.

5 References

- Garnier, B.J., Ohmura, A., 1968. A method of calculating the direct shortwave radiation income of slopes. J. Appl. Meteorol. 7, 796–800. doi:10.1175/1520-0450(1968)007<0796:AMOCTD>2.0.CO;2.
- Gudmundsson L, Bremnes JB, Haugen JE, Engen-Skaugen T (2012) Technical Note: Downscaling predicted climatic precipitation to the station scale using statistical transformations - A comparison of methods. Hydrology and Earth System Sciences 16, 3383–3390. doi:10.5194/hess-16-3383-2012.
- McMahon, T.A., Peel, M.C., Lowe, L., Srikanthan, R., McVicar, T.R. 2013. Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. Hydrology & Earth System Sciences 17, 1331–1363. doi:10.5194/hess-17-1331-2013.

- Penman, H. L. 1948. Natural evaporation from open water, bare soil and grass. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 193, 120-145.
- Penman, H. L. 1956. Evaporation: An introductory survey. Netherlands Journal of Agricultural Science, 4, 9-29.
- Sanjuan, G., Brun, C., Cort, A., 2014. Wind field uncertainty in forest fire propagation prediction. Procedia Comput. Sci. 29, 1535–1545. doi:10.1016/j.procs.2014.05.139.
- Thornton, P.E., Running, S.W., White, M. a., 1997. Generating surfaces of daily meteorological variables over large regions of complex terrain. J. Hydrol. 190, 214–251. doi:10.1016/S0022-1694(96)03128-9.
- Thornton, P.E., Running, S.W., 1999. An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. Agric. For. Meteorol. 93, 211–228. doi:10.1016/S0168-1923(98)00126-9.
- Tymstra, C., Bryce, R.W., Wotton, B.M., Taylor, S.W., Armitage, O.B., 2010. Development and Structure of Prometheus : the Canadian Wildland Fire Growth Simulation Model, Information report NOR-X-417. doi:ISBN 978-1-100-14674-4
- Valiantzas J.D. 2006. Simplified versions for the Penman evaporation equation using routine weather data. Journal of Hydrology 331, 690–702. doi:10.1016/j.jhydrol.2006.06.012.