The meteoland reference book

true

2022-01-05

Contents

Preface

This is a reference book for the data structures and functions implemented in **meteoland**, an R package that provides functions to estimate daily weather at any position over given terrains.

How to use this book

This reference book is meant to help you understand the data structures and functions included in package **meteoland** (ver. **1.0.2**). Hands-on user guides can be found as package vignettes within the package. As any reference book, you are not expected to read the book linearly, but to jump to sections whenever you have doubts about the design of the package or implementation of certain calculations.

The first two chapters of the book present the package, its data structures and main function. Chapter 3 focuses on the functions supplied for interpolating weather records and chapter 4 explains how solar radiation is estimated. Chapter 5 is devoted to statistical correction of weather series (i.e. bias correction). The remaining chapters detail other functions meant to complete the package.

In this book we use `objectname` or `variablename` to indicate R code or to refer to an R objects or a variable within data frames, and `functionname()` to refer to a package function. Whenever relevant, we indicate the correspondence between mathematical symbols, their units and the names used within the R package.

The reference book describes in detail the design and functioning of the package. An introduction to the package was provided by De Cáceres et al. (2018) at the time of its presentation, but this reference book should be preferred for an up-to-date description of the package. Our aim is to update this reference book along with package developments, so that users have detailed and up-to-date information about the models at the time functions are run. As the book will not be static, after a given application we recommend users to store a **PDF version** of the reference book to be sure it matches the version the package reported in their application report or article.

Chapter 1

Introduction

1.1 Purpose

Reliable meteorological data are a basic requirement for hydrological and ecological studies at the landscape scale. Given the large variation of weather 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 may differ. Finally, studies addressing the impacts of climate change on forested 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 predictions is also important.

With the aim to assist research of climatic impacts, the R package **meteoland** (De Cáceres et al. 2018) provides utilities to estimate daily weather variables at any position over complex terrains. The package provide functions to assist the following tasks:

- Spatial interpolation of daily weather records from meteorological stations.
- Statistical correction of meteorological data series (e.g. from climate models).
- Multisite and multivariate stochastic weather generation.

Spatial interpolation is required when meteorology for the area and period of interest cannot be obtained from local sensors. The nearest weather station may not have data for the period of interest or it may be located too far away to be representative of the target area. Correcting the biases of a meteorological data series containing biases using a more accurate meteorological series is necessary when the more accurate series does not cover the period of interest and the less accurate series does. The less accurate series may be at coarser scale, as with climate model predictions or climate reanalysis data. In this case one can speak of statistical correction and downscaling. However, one may also correct the predictions of climate models using reanalysis data estimated at the same spatial resolution. Finally, stochastic weather generators are algorithms that produce series of synthetic daily weather data. The parameters of the model are conditioned on existing meteorological records to ensure the characteristics of input weather series emerge in the daily stochastic process.

1.2 Package installation

Package **meteoland** is officially distributed via CRAN. Hence, it can be installed using:

```r
install.packages("meteoland")
```

Users can also download and install the latest stable versions GitHub as follows (required package `devtools` should be installed/updated first):

```r
devtools::install_github("emf-creaf/meteoland")
```

Chapter 2

Data structures and main functions

2.1 Meteorological variables

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

- `DOY`: Day of the year ([1-366]).
- `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).

Since specific humidity is often specified instead of relative humidity, the package also allows reading and using this variable, although it will not be written in files:

- `SpecificHumidity`: Specific humidity (in kg/kg).

2.2 Spatial classes

The package deals with three kinds of spatial structures: individual **points**, a set of **pixels** from a spatial grid and full (i.e. complete) **grids**. The package includes six S4 spatial classes, which are defined as children of classes in package **sp**. Spatial structures are used to represent both topographical and meteorological data. These are described in the following subsections.

2.2.1 Topography

Three classes are defined to represent the variation of topographic features (i.e., elevation, slope and aspect) over space:

- Class `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
## Class "SpatialVector", by class "SpatialPointsDataFrame", distance 3
```

- Class `SpatialGridTopography` extends `SpatialGridDataFrame` and represents the continuous variation of topographic features over a full spatial 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
```

- Class `SpatialPixelsTopography` extends `SpatialPixelsDataFrame` and represents the continuous variation of topographic features over a set if cells in a grid.

```
## Class "SpatialPixelsTopography" [package "meteoland"]
##
## Slots:
##
## Name:          data   coords.nrs         grid   grid.index       coords         bbox  proj4str
## Class:   data.frame      numeric GridTopology      integer       matrix       matrix
##
## Extends:
## Class "SpatialPixelsDataFrame", directly
## Class "SpatialPixels", by class "SpatialPixelsDataFrame", distance 2
## Class "SpatialPointsDataFrame", by class "SpatialPixelsDataFrame", distance 2
## Class "SpatialPoints", by class "SpatialPixelsDataFrame", distance 3
## Class "Spatial", by class "SpatialPixelsDataFrame", distance 4
## Class "SpatialVector", by class "SpatialPixelsDataFrame", distance 4
```

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

2.2.2 Meteorology

Analogously to topography classes, three spatial classes are used to represent the variation of daily meteorology over space:

- Class `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
## Class "SpatialVector", by class "SpatialPoints", distance 2
```

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

```
## Class "SpatialGridMeteorology" [package "meteoland"]
##
## Slots:
##
## Name:         dates         data         grid         bbox  proj4string
## Class:         Date       vector GridTopology       matrix          CRS
##
## Extends:
## Class "SpatialGrid", directly
## Class "Spatial", by class "SpatialGrid", distance 2
```

- Class `SpatialPixelsMeteorology` extends `SpatialPixels` and represents the variation of daily meteorology for a set of pixels (cells) of a spatial grid.

```
## Class "SpatialPixelsMeteorology" [package "meteoland"]
##
## Slots:
##
## Name:         dates         data         grid   grid.index       coords         bbo
## Class:         Date       vector GridTopology      integer       matrix       matri
##
## Extends:
## Class "SpatialPixels", directly
## Class "SpatialPoints", by class "SpatialPixels", distance 2
## Class "Spatial", by class "SpatialPixels", distance 3
## Class "SpatialVector", by class "SpatialPixels", distance 3
```

In addition to their corresponding inherited slots, classes `SpatialPointsMeteorology`, `SpatialGridMeteorology` and `SpatialPixelsMeteorology` 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 the three classes have a `data` slot containing data frames, meteorological data is in different form in each class. In objects of `SpatialPointsMeteorology`, there is one data frame for each point where variables are in columns and dates are in rows. In objects of `SpatialGridMeteorology` and `SpatialPixelsMeteorology`, each data frame describes the meteorology over a complete grid, or a subset of cells, for a single day. In these cases, the data frame has grid cells in rows and variables in columns.

2.3 Reading and writing meteorological data

2.3.1 Point meteorology (ascii/rds files)

Objects of class `SpatialPointsMeteorology` are stored in the disk using one data file for each of their spatial points. Files can be stored in **ascii** (i.e. text) format or **rds** (i.e. R data of a single object compressed) format. Package **meteoland** provides four input/output functions for point meteorology:

- Function `readmeteorologypoint()` reads the meteorological data stored in one **ascii**/**rds** data file and returns a data frame.
- Function `writemeteorologypoint()` writes the meteorological data of a single point as an **ascii**/**rds** file in the file system.
- Function `readmeteorologypointfiles()` reads several **ascii**/**rds** files and returns an object of class `SpatialPointsMeteorology`.
- Functions `writemeteorologypointfiles()` writes several **ascii**/**rds** 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.

2.3.2 Point/gridded meteorology (netCDF)

NetCDF is a set of libraries to write machine-independent data formats that support the creation and sharing of array-oriented scientific data. **NetCDF** is very useful useful to store spatio-temporal datasets, as it allows storing the grid topology, the time period being described and the spatial projection as well as the actual data all in the same file. Moreover, **NetCDF** can be read and written lazily, without having to load all the file in memory.

Package **meteoland** allows objects of classes `SpatialPointsMeteorology`, `SpatialPixelsMeteorology` and `SpatialGridMeteorology` to be stored in the disk as **NetCDF** thanks to the functions provided by **ncdf4** R package, which provides an interface to Unidata's netCDF library (version 4 or earlier). We try to conform to CF specifications in files written by **meteoland**, but improvements can always be made.

The following functions are available for input/output of point meteorology:

- Functions `writemeteorologypoints()` writes meteorological point data on a **NetCDF**.
- Function `readmeteorologypoints()` reads point meteorological data stored in one or several **NetCDFs** and returns an object of class `SpatialPointsMeteorology`. If several files are read, the function tries to merge the result.

And the following functions are available for input/output of meteorology on pixels or full grids:

- Functions `readmeteorologygrid()` and `readmeteorologypixels()` read the meteorological data stored in one or several **NetCDF** file and return an object of class `SpatialGridMeteorology` or `SpatialPixelsMeteorology`, respectively. If several files are read, the function tries to merge the result. Rotated grids should not be read using this functions.
- Function `readmeteorologygridpoints()` allows reading grid pixels as points. It returns an object of class `SpatialPointsMeteorology`. This feature becomes useful to read data from rotated grids.
- Functions `writemeteorologygrid()` and `writemeteorologypixels()` write the meteorological data of the full grid or the subset of grid cells, for a **NetCDF**. The same functions allow adding and replacing data content in a previously existing **NetCDF**.
- Function `writemeteorologygridpixel()` allows writing/replacing data for specific pixels in an existing **NetCDF**.

2.4 Visualizing input topography and meteorological data

Although very simple, the package provides two kinds of functions to visualize the temporal and spatial variation of meteorology:

- Function `spplot()` has been redefined from package **sp** to draw maps of specific weather variables corresponding to specific dates. The function can be used on objects of class `SpatialGridMeteorology` and `SpatialPixelsMeteorology`.
- Function `meteoplot()` allows the temporal series of specific variables on specific spatial points to be plotted. The function can read the data from the disk or from objects of class `SpatialPointsMeteorology`.

Similarly, function `spplot()` also accepts objects of classes `SpatialGridTopography` and `SpatialPixelsTopography`, so that topography can also be easily displayed.

2.5 Subsetting, merging and reshaping data

A number of functions are available to manipulate data structures and to make them available for other R packages.

2.5.1 Subsetting and reshaping topography data

Following the design of package **sp**, package meteoland provides several functions to manipulate topographic data structures (see fig. 2.1).

1. It is possible to coerce objects between from topography classes to the corresponding class in package **sp** using functions `as()`. This includes coercing to spatial objects with or without data columns. Additionally, an object of `SpatialPointsTopography` can be coerced into a `SpatialPixelsTopography` using a call like `as(x, "SpatialPixelsTopography")`.

2. It is possible to coerce objects to the general-purpose class `sf` of package **sf** using `as(..., "sf")`.
3. Subsetting (i.e. `[` operator) is allowed on all three spatial topographic classes. The valid arguments of this function call depend on the kind of structure.

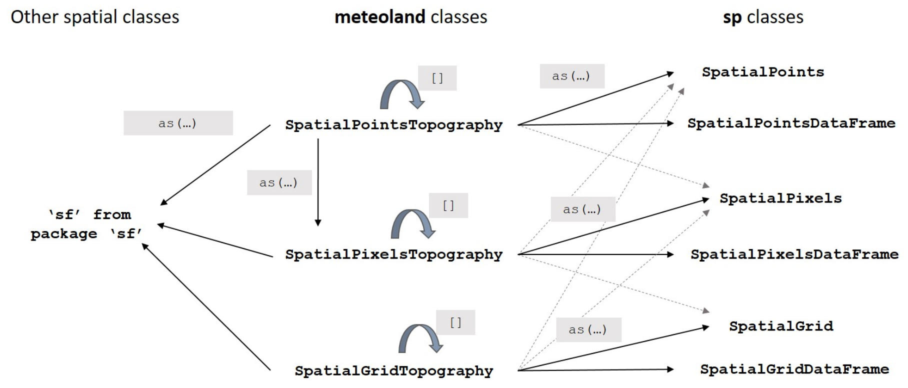


Figure 2.1: Conversion between meteoland classes for topography and classes from other packages

2.5.2 Subsetting and reshaping meteorology data

Analogously, package meteoland provides several functions to manipulate spatio-temporal data structures (see fig. 2.2).

1. It is possible to coerce objects between spatial classes using functions `as()`.
2. Subsetting (i.e. `[` operator) is allowed on all six spatial classes (topographic or meteorological). The valid arguments of this function call depend on the kind of structure.

3. The package provides functions to extract meteorological data (these allow reading data in memory or from the disk):

- Functions `extractgridindex()` extracts the meteorology of a particular pixel index from a grid and returns a data frame with dates in rows and variables in columns.
- Function `extractgridpoints()` extract the meteorology of point locations from a grid and returns an object of `SpatialPointsMeteorology`, similarly to the `[` operator.
- Functions `extractdates()` and `extractvars()` extract the meteorology of a set of dates (or variables, respectively) from a `SpatialPointsMeteorology` object, (or `SpatialGridMeteorology` or `SpatialPixelsMeteorology`) and returns a `SpatialPointDataFrame` (or `SpatialGridDataFrame` or `SpatialPixelsDataFrame`) for each date (resp. variable) in a named list. This kind of conversion is also useful to export meteoland outputs to package **raster**, which provides wrappers for the abovementioned **sp** classes.

4. Finally, the package provides some functions to reshape meteorological data structures into more general structures of spatio-temporal data, so that they can be more easily processed, combined and stored using other R packages. In particular, objects of classes `SpatialPointsMeteoroloy`, `SpatialGridMeteorology` and `SpatialPixelsMeteorology` can be reshaped into objects of packages **stars** and **spacetime** by using calls like `as(x, "stars")` or `as(x, "STFDF")`.

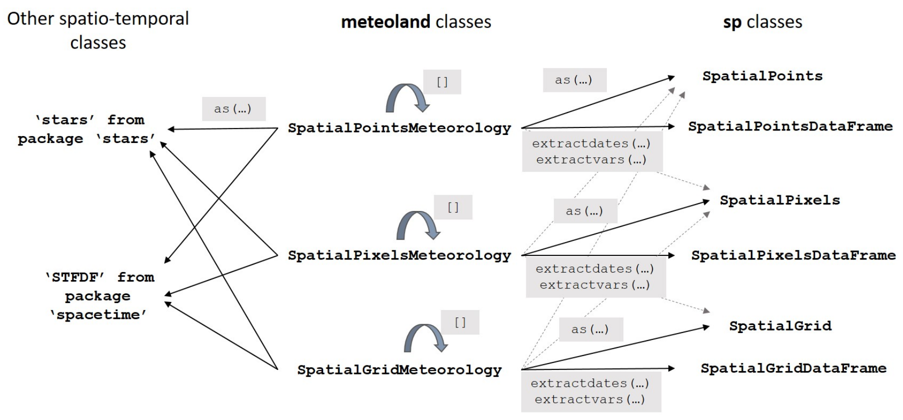


Figure 2.2: Conversion between meteoland classes for meteorology and classes from other packages

2.5.3 Merging meteorology data

Functions `mergegrid()` and `mergepoints()` take any number of objects of the same spatial class and return a merged object. The functions pool over dates and variables, but the objects to merge should have the same spatial structures (points, grids, reference systems, …). This is specially useful to merge data corresponding to different periods.

2.6 Summarizing meteorological data

2.6.1 Temporal summaries

The package provides functions to generate *temporal* summaries of meteorological data. These accept meteorology objects as input and return their corresponding spatial dataframe structure with the summary statistics in columns:

- Function `summarypoints()` summarizes the meteorology of spatial points. It accepts objects of `SpatialPointsMeteorology` as input and returns an object of `SpatialPointsDataFrame` with point summaries for the requested variable. Temporal summaries can be calculated for different periods and using different summarizing functions (e.g. mean, sum, minimum, maximum, etc.).
- Functions `summarygrid()` and `summarypixels()` summarize the meteorology of full grids or of subset of grid cells, respectively. They accept objects of `SpatialGridMeteorology` and `SpatialPixelsMeteorology`, respectively, as input and return an object of `SpatialGridDataFrame` and `SpatialPixelsDataFrame`, respectively, with temporal summaries for the requested variable over the range of dates indicated.
- Function `summaryinterpolationdata()` works similarly to `summarypoints()`, but takes an object of class `MeteorologyInterpolationData` as input.

All temporal summary functions except `summaryinterpolationdata()` accept a data file, or a metadata file as input. This allows, producing summaries lazily, without loading complete data structures in memory. Of course, this mode of producing summaries may be slower than operating on objects already loaded in memory.

2.6.2 Spatial summaries

The package provides one function to produce *spatial* summaries. Function `averagearea()` averages the coordinates and meteorological values of any spatial meteorology object, returning an object of class `SpatialPointMeteorology` with a single point representing the average.

2.7 Meteorology estimation functions

2.7.1 Spatial interpolation

Package **meteoland** provides three 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 `interpolationpixels()` interpolates weather for pixels in a grid specified in `SpatialPixelsTopography` and returns an object of class `SpatialPixelsMeteorology`.
- 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                    slope
## Class:                   matrix                  numeric                  numeric
##
## Name:            MinTemperature           MaxTemperature   SmoothedPrecipitation
## Class:                   matrix                   matrix                   matrix
##
## Name:  SmoothedTemperatureRange         RelativeHumidity                Radiation
## Class:                   matrix                   matrix                      ANY
##
## Name:             WindDirection               WindFields                  WFIndex
## Class:                      ANY                      ANY                      ANY
##
## Name:                    params                    dates                     bbox
## Class:                     list                     Date                   matrix
##
## Extends:
## Class "MeteorologyProcedureData", directly
## Class "Spatial", by class "MeteorologyProcedureData", distance 2
```

When calling functions `interpolationpoints()`, `interpolationpixels()`, or `interpolationgrid()`, the user may require interpolation outputs to be written into the file system, instead of being stored in memory. For example, if `interpolationpoints()` is called with `export = TRUE` and a location directory, the function will write the data frame produced for each point into an **ascii** text file or a **rds** file. Metadata files will also be written, so that results can later be loaded in memory. Alternatively, `interpolationpoints()` can be forced to write results in a **netCDF** by specifying `export = TRUE` and `exportFile = <filename>`. Similarly, if `interpolationpixels()` or `interpolationgrid()` are called specifying `exportFile = <filename>`, the functions will create and add data to a **netCDF**. These options becomes important when dealing with very large spatial structures.

Functions `interpolation.calibration()` and `interpolation.cv()` are included in **meteoland** to calibrate interpolation parameters and evaluate predictive performance of interpolation routines before using them. Details of interpolation routines are described in chapter 3.

2.7.2 Statistical correction

One function is available for statistical correction of meteorological data series (i.e., one function for each data structure). Function `correctionpoints()` performs statistical correction of weather data series on a set of locations and it returns an object of class `SpatialPointsMeteorology` containing corrected weather predictions. Statistical correction requires an object of class `MeteorologyUncorrectedData`, which contains the X-Y coordinates and the coarse-scale meteorological data to be corrected, which includes a reference (historic) period and projected (e.g. future) period:

```
## Class "MeteorologyUncorrectedData" [package "meteoland"]
##
## Slots:
##
## Name:           coords  reference_data projection_data          params           dates
## Class:          matrix             ANY             ANY            list            Date
##
## Name:      proj4string
## Class:             CRS
##
## Extends:
## Class "MeteorologyProcedureData", directly
## Class "Spatial", by class "MeteorologyProcedureData", distance 2
```

The reference (historical) period is compared with observed meteorological data of the same period, and the routine uses this information to correct the projected (e.g. future) period. Therefore, apart from the `MeteorologyUncorrectedData` object, the correction function requires accurate meteorological 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 `correctionpoints()`, the user may require the outputs to be written into the file system, instead of being returned in memory. The options are the same as described for interpolation.

Function `correctionpoints.errors()` was included in the package to evaluate the errors of the less accurate and more accurate series. Comparisons can be made before and after applying statistical corrections. In the latter case, cross-validation is also available. Details of correction routines are described in chapter 5.

2.7.3 Weather generation

Function `weathergeneration()` can be used to generate synthetic weather series for a range of inputs, including a (non-spatial) data frame (corresponding to weather series of a single location) as well as objects of classes `SpatialPointsDataFrame`, `SpatialPixelsDataFrame` or `SpatialGridDataFrame`. The output of the function is of the same class as the input object. Weather generation algorithms are described in chapter 7.

Chapter 3

Spatial interpolation of weather records

3.1 Overview

Ecological research studies conducted for historical periods can be perfomed using meteorological records obtained from surface weather stations of the area under study. The general procedure for interpolation is very similar to the one that underpins the U.S. DAYMET dataset (https://daymet.ornl.gov/). 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 (Peter E. Thornton, Running, and White 1997). Relative humidity can be either interpolated (in fact, dew-point temperature 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 Garnier and Ohmura (1968). Potential solar radiation is then corrected to account for atmosphere transmittance using the predictions of temperature range, relative humidity and precipitation (P. E. Thornton and Running 1999). Finally, the wind vector (wind direction and wind speed) is interpolated by using weather station records and static wind fields.

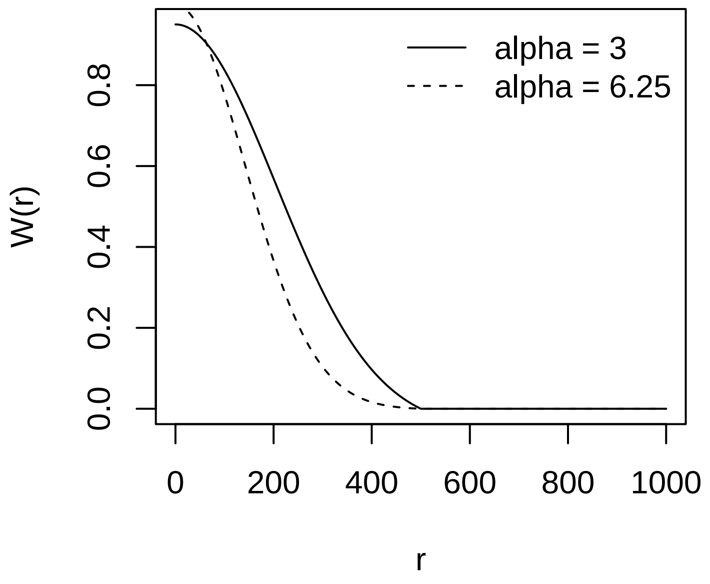
In the following sections we detail the general algorithm used to obtain interpolation weights and the interpolation procedure for temperature, precipitation, relative humidity and wind. The estimation of potential and actual solar radiation is explained in the next chapter.

3.2 Interpolation weights

Peter E. Thornton, Running, and White (1997) suggested interpolating meteorological data using a truncated Gaussian filter. Its form with respect to a central point p is:

$$W(r) = e^{-\alpha \cdot (r/R_p)^2} - e^{-\alpha} \tag{3.1}$$

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 automatically adjusted so that it has lower values in data-rich regions and is increased in data-poor regions. The method, however, requires 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:

$$D_p = \frac{\sum_{i=1}^{n} (W_i/\hat{W})}{\pi \cdot R_p^2} \tag{3.2}$$

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

$$\hat{W} = \left(\frac{1 - e^{-\alpha}}{\alpha}\right) - e^{-\alpha} \tag{3.3}$$

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

$$R_p = \sqrt{\frac{N^*}{D_p \cdot \pi}} \tag{3.4}$$

   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.

Peter E. Thornton, Running, and White (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 Peter E. Thornton, Running, and White (1997), we set $R = 140000$ meters and $I = 3$ by default (see parameters `initial_Rp` and `iterations` given in function `defaultInterpolationParams()`). The other parameters (N and α) depend on the variable to be interpolated.

3.3 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. Peter E. Thornton, Running, and White (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:

$$(T_1 - T_2) = \beta_0 + \beta_1 \cdot (z_1 - z_2) \tag{3.5}$$

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:

$$T_p = \frac{\sum_{i=1}^{n} W_i \cdot (T_i + \beta_0 + \beta_1 \cdot (z_p - z_i))}{\sum_{i=1}^{n} W_i} \tag{3.6}$$

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

3.4 Relative humidity

Relative humidity is a parameter not always recorded in weather stations. When input station weather data does not include relative humidity, **meteoland** estimates it directly from minimum and maximum temperature (Peter E. Thornton, Running, and White 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 kPa) as:

$$e_p = 0.61078 \cdot e^{\left(\frac{17.269 \cdot T_{d,p}}{237.3 + T_{d,p}}\right)} \tag{3.7}$$

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

$$e_{s,p} = 0.61078 \cdot e^{\left(\frac{17.269 \cdot T_{a,p}}{237.3 + T_{a,p}}\right)} \tag{3.8}$$

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:

$$RH_p = 100 \cdot \frac{e_p}{e_{s,p}} \tag{3.9}$$

When relative humidity has been measured at weather stations, interpolation should be preferred to estimation from minimum and maximum temperature. However, because relative humidity depends on temperature, 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:

$$e_i = e_{s,i} \cdot (RH_i/100) \tag{3.10}$$

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:

$$T_{d,i} = \frac{237.3 \cdot \ln(e_i/0.61078)}{17.269 - \ln(e_i/0.61078)} \tag{3.11}$$

Unlike temperature, interpolation of dew temperature is not corrected for elevation differences. The dew-point temperature for the target point, $T_{d,p}$ is predicted as:

$$T_{d,p} = \frac{\sum_{i=1}^{n} W_i \cdot T_{d,i}}{\sum_{i=1}^{n} W_i} \tag{3.12}$$

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 ($RH_{max,p}$, $RH_{min,p}$). After their estimation, the routine checks that the predicted maximum and minimum relative humidity values stay within the physical limits 0% and 100%. Although interpolation of dew-point temperature does not account for elevation differences, interpolated values of relative humidity will normally exhibit a pattern following elevation differences because temperature is involved in the calculation of relative humidity.

3.5 Precipitation

Predictions of precipitation are complicated by the need to predict both daily occurrence and, conditioned on this, daily precipitation amount. Peter E. Thornton, Running, and White (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:

$$POP_p = \frac{\sum_{i=1}^{n} W_{o,i} \cdot PO_i}{\sum_{i=1}^{n} W_{o,i}} \tag{3.13}$$

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, Peter E. Thornton, Running, and White (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 Peter E. Thornton, Running, and White (1997), who use 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:

$$\left(\frac{P_1 - P_2}{P_1 + P_2}\right) = \beta_0 + \beta_1 \cdot (z_1 - z_2) \tag{3.14}$$

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

$$P_p = \frac{\sum_{i=1}^{n} W_{o,i} \cdot P_i \cdot PO_i \cdot \left(\frac{1+f}{1-f}\right)}{\sum_{i=1}^{n} W_{o,i} \cdot PO_i} \tag{3.15}$$

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 $|f| > f_{max}$.

3.6 Wind

Interpolation of wind characteristics depends on the amount of information available:

- Interpolation of wind speed only
- Interpolation of wind vectors (speed and direction)
- Interpolation of wind vectors using wind fields

The following subsections detail the calculations in each case.

3.6.1 Interpolation of wind speed

The predicted wind speed u_p for a target point p is the weighted average of station wind speed values $\{u_i\}$ $i = (1, ..., n)$ using the interpolation weights W_i determined from the truncated Gaussian filter:

$$u_p = \frac{\sum_{i=1}^{n} W_i \cdot u_i}{\sum_{i=1}^{n} W_i} \tag{3.16}$$

3.6.2 Interpolation of wind vectors

Interpolation of wind vectors 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:

$$x_i = u_i \cdot \sin(\theta_i) \quad y_i = u_i \cdot \cos(\theta_i) \tag{3.17}$$

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:

$$x_p = \frac{\sum_{i=1}^{n} W_i \cdot x_i}{\sum_{i=1}^{n} W_i} \quad y_p = \frac{\sum_{i=1}^{n} W_i \cdot y_i}{\sum_{i=1}^{n} W_i} \tag{3.18}$$

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

$$u_p = \sqrt{x_p^2 + y_p^2} \quad \theta_p = \tan^{-1}(x_p/y_p) \tag{3.19}$$

3.6.3 Interpolation of wind vectors using wind fields

More precise wind interpolation of wind vectors 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:

$$x_{m_i,p} = u_{m_i,p} \cdot \sin(\theta_{m_i,p}) \quad y_{m_i,p} = u_{m_i,p} \cdot \cos(\theta_{m_i,p}) \tag{3.20}$$

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:

$$x_p = \frac{\sum_{i=1}^{n} W_i \cdot x_{m_i,p}}{\sum_{i=1}^{n} W_i} \quad y_p = \frac{\sum_{i=1}^{n} W_i \cdot y_{m_i,p}}{\sum_{i=1}^{n} W_i} \tag{3.21}$$

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

Chapter 4

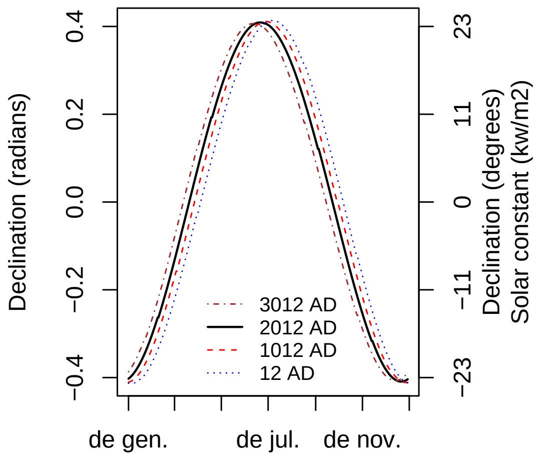
Estimation of solar radiation

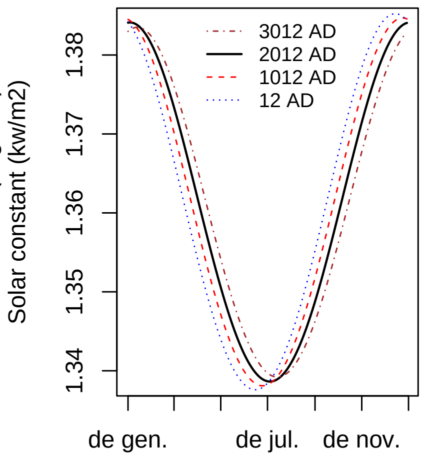
Incident daily solar radiation is not interpolated, but estimated from topography and measurements of temperature, humidity and precipitation.

4.1 Solar declination and solar constant

The declination of the sun δ is the angle between the rays of the sun and the plan of the Earth's equator. Solar declination varies with years and seasons. However, the Earth's axial tilt changes slowly over thousands of years but it is nearly constant for shorter periods, so the change in solar declination during one year is nearly the same as during the next year. Solar constant (I_0) is normally given a nominal value of 1.361 $kW \cdot m^{-2}$ but in fact it also varies through the year and over years. Both can be calculated from Julian day (J), the number of days number of days since January 1, 4713 BCE at noon UTC. from Julian day. In **meteoland**, julian days, solar declination and solar constant are calculated using an adaptation of the code as in package **insol** by J.G. Corripio, which is based on Danby (1988) and Reda and Nrel (2008).

The following figures show the variation of solar declination and the value of solar constant over a year (see functions `radiation_solarDeclination()` and `radiation_solarConstant()`):





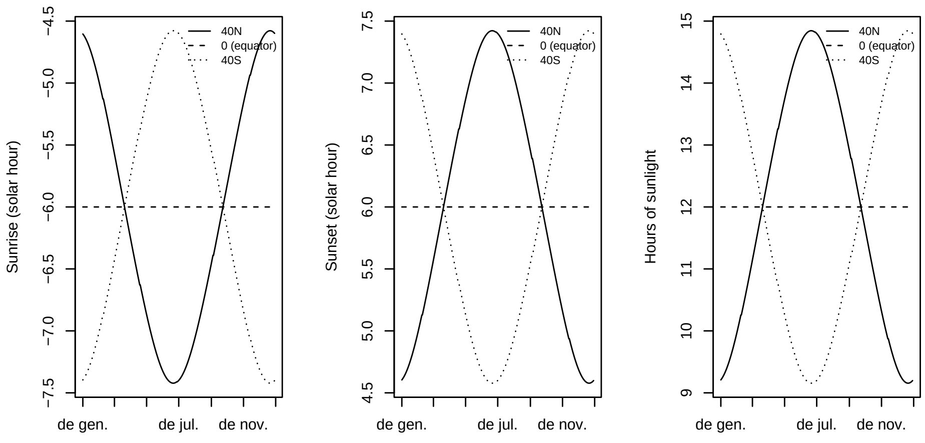
4.2 Day length

Calculation of sunrise and sunset on a horizontal surface is rather straightforward. The hour angles of sunrise and sunset (sr and ss, both in radians) for a horizontal surface of latitude ϕ on a day with declination δ (both expressed in radians) are:

$$sr = T_1 = \cos^{-1}(\max(\min(-\tan(\phi)\cdot\tan(\delta),1),-1)) \tag{4.1}$$

$$ss = T_0 = -T_1 \tag{4.2}$$

Knowing that each hour corresponds to 15 degrees of rotation, hour angles can be transformed to solar hours. The following figures show the seasonal variation of sunrise and sunset hours, as well as day length, for horizontal surfaces in three latitudes (40North, equator and 40South) (see functions `radiation_sunRiseSet()` and ‘radiation_daylength()}):



For inclinated slopes, the calculation of day length is based on the concept of equivalent slopes, which are places on earth where the slope of earth's surface is equal to the slope of interest. The calculations start with the determination of the latitude L_1 of the equivalent slope:

$$L_1 = \sin^{-1}\left(\cos(Z_x)\cdot\sin(\phi) + \sin(Z_x)\cdot\cos(\phi)\cdot\cos(A)\right) \tag{4.3}$$
$$D = \cos(Z_x)\cdot\cos(\phi) - \sin(Z_x)\cdot\sin(\phi)\cdot\cos(A) \tag{4.4}$$

where ϕ is the latitude, 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). Then L_2 is defined depending on the value of D. If $D < 0$ then:

$$L_2 = \tan^{-1}\left(\frac{\sin(Z_x)\cdot\sin(A)}{D}\right) + \pi \tag{4.5}$$

Otherwise, L_2 is calculated as:

$$L_2 = \tan^{-1}\left(\frac{\sin(Z_x)\cdot\sin(A)}{D}\right) \tag{4.6}$$

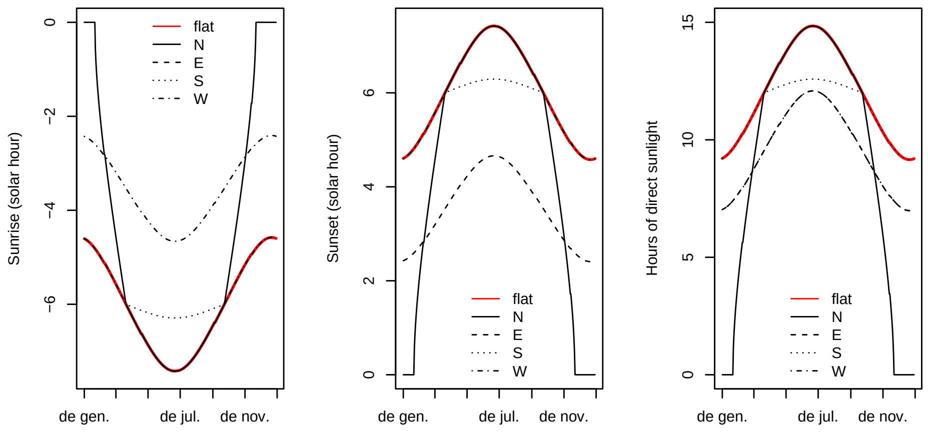
Once L_1 and L_2 are available, we can calculate solar hours on equivalent slopes:

$$T_7 = \cos^{-1}\left(\max(\min(-\tan(L_1)\cdot\tan(\delta),1),-1)\right) - L2 \tag{4.7}$$
$$T_6 = -\cos^{-1}\left(\max(\min(-\tan(L_1)\cdot\tan(\delta),1),-1)\right) - L2 \tag{4.8}$$

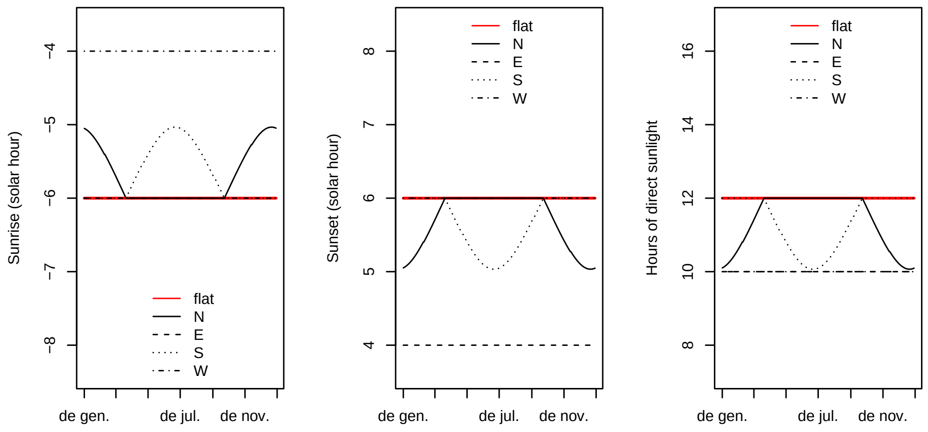
Being T_6 and T_7 the hour angle of sunrise and sunset on equivalent slopes, respectively. and the hour angles of sunrise (sr) and sunset (ss) on the slope (both in radians) are found comparing the hour angles on equivalent surfaces with the hour angles on the horizontal surface:

$$sr = \max(T_0, T_6) \tag{4.9}$$
$$ss = \min(T_1, T_7) \tag{4.10}$$

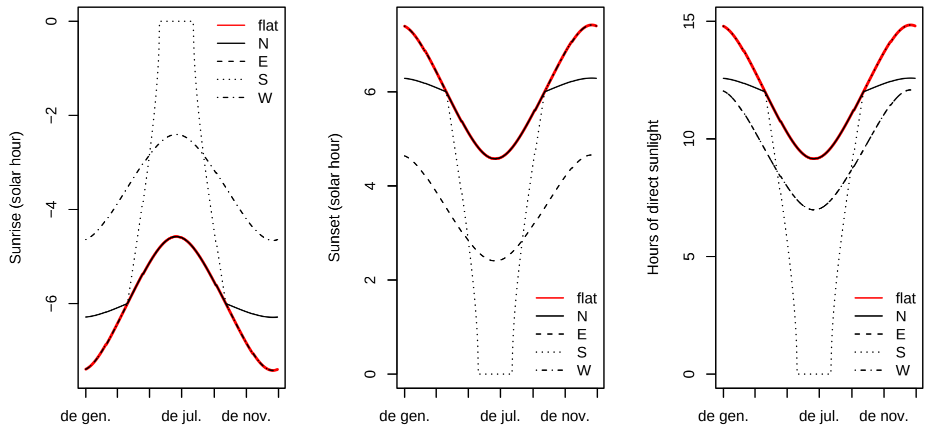
The following three figures show the seasonal variation of sunrise and sunset hours, as well as day length, for slopes of 30 inclination, facing to the four cardinal points. Curves for flat surfaces are shown for comparison. If the slopes are at latitude 40North:



whereas if they are at Equator (i.e. $\phi = 0$):



and if they are at latitude 40South:



4.3 Potential radiation

Potential solar radiation is the radiation that a surface on earth would receive if atmosphere was not present (i.e. without the effects of cloud reflection, scattering, ...). In **meteoland**, potential solar radiation is estimated from solar declination, latitude, aspect and slope according to Garnier and Ohmura (1968). Daily potential radiation (R_{pot}, in $MJ \cdot m^{-2}$) is calculated by integrating instantaneous potential radiation $R_{pot,s}$ (in $kW \cdot m^{-2}$) over the day between sunrise (sr) and sunset (ss), using 10 min (i.e. 600 sec) intervals:

$$R_{pot} = \frac{1}{1000} \cdot \sum_{s=sr}^{ss} 600 \cdot R_{pot,s} \tag{4.11}$$

In turn, instantaneous potential solar radiation $R_{pot,s}$ is calculated using:

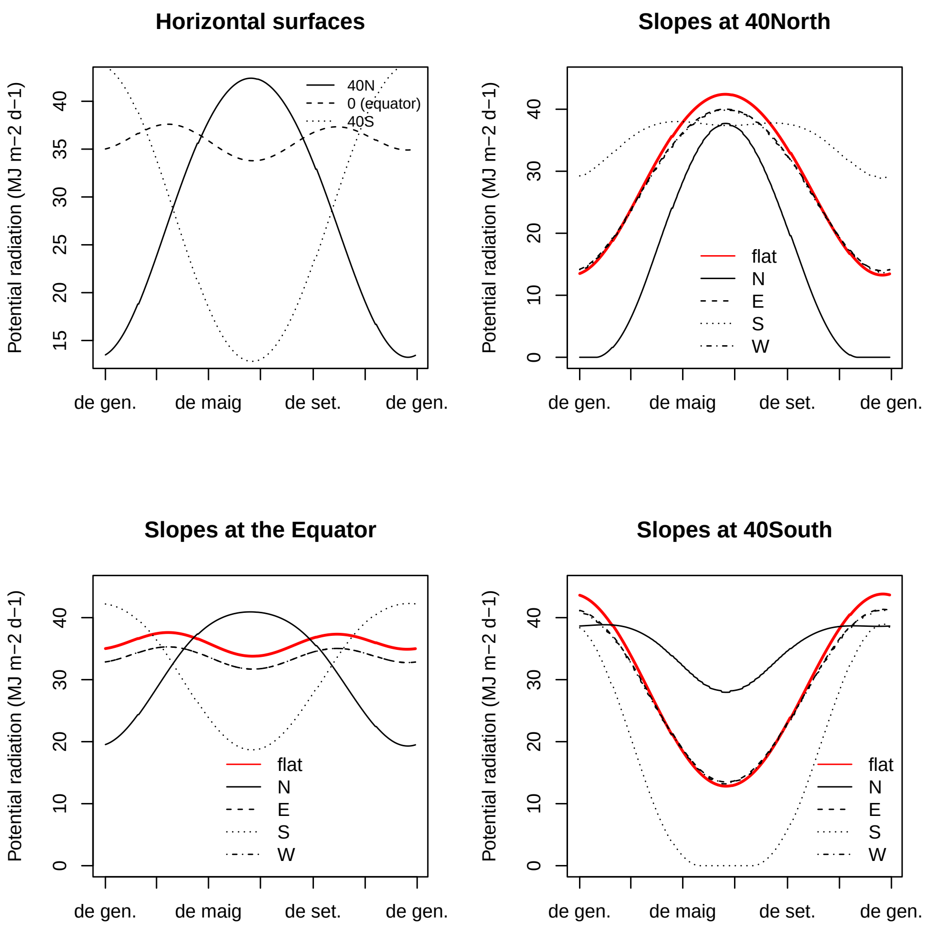
$$\begin{aligned} R_{pot,s} = \; & I_0 \cdot [(\sin\phi \cdot \cos H)(-\cos A \cdot \sin Z_x) - \sin H \cdot (\sin A \cdot \sin Z_x) \\ & + [(\cos\phi \cdot \cos H) \cdot \cos Z_x] \cdot \cos\delta \\ & + [\cos\phi \cdot (\cos A \cdot \sin Z_x) + \sin\phi \cdot \cos Z_x] \cdot \sin\delta] \end{aligned} \tag{4.12}$$

where I_0 is the solar constant, 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), 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. Note that in the case of a flat surface the previous equation reduces to:

$$R_{pot,s} = I_0 \cdot [\cos\phi \cdot \cos H \cdot \cos\delta + \sin\phi \cdot \sin\delta] = I_0 \cdot \sin\beta \tag{4.13}$$

where β is called the solar elevation angle.

The following figures illustrate seasonal variation of potential solar radiation for the horizontal inclinated surfaces presented above (see function `radiation_potentialRadiation()`):



4.4 Incident solar radiation

Incident solar radiation is the amount of (direct) solar radiation reaching the surface after accounting for the atmosphere. Improving the method proposed in Peter E. Thornton, Running, and White (1997), P. E. Thornton and Running (1999) calculate incident daily total solar radiation R_g as:

$$R_g = R_{pot} \cdot T_{t,max} \cdot T_{f,max} \tag{4.14}$$

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:

$$T_{t,max} = \left[\frac{\sum_{s=sr}^{ss} R_{pot,s} \cdot \tau^{(P_z/P_0) \cdot m_\theta}}{\sum_{s=sr}^{ss} R_{pot,s}} \right] + (\alpha_{e_p} \cdot e_p) \tag{4.15}$$

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 kPa), estimated as explained before; $\alpha_{e_p} = -6.1 \cdot 10^{-2}\text{kPa}^{-1}$ 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 $\cos(\theta) = \sin\phi \cdot \sin\delta + \cos\phi \cdot \cos\delta \cdot \cos H$; and P_z/P_0 is the ratio between air pressure at elevation z_p and air pressure at the sea level, calculated as:

$$(P_z/P_0) = (1.0 - 2.2569 \cdot 10^{-5} \cdot z_p)^{5.2553} \tag{4.16}$$

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:

$$T_{f,max} = 1.0 - 0.9 \cdot e^{-B \cdot \Delta T^C} \tag{4.17}$$

being $C = 1.5$ and B calculated from:

$$B = b_0 + b_1 \cdot e^{-b_2 \cdot \hat{\Delta T}} \tag{4.18}$$

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.

Although the calculation of incident solar radiation can be done independently of interpolation (see function `radiation_solarRadiation()`), it is automatically done in functions `interpolationpoints()` and `interpolationgrid()`.

4.5 Outgoing longwave radiation and net radiation

Potential and actual evapotranspiration calculations require estimating the energy actually absorbed by evaporating surfaces. Daily net radiation R_n (in $MJ \cdot m^{-2} \cdot day^{-1}$) is calculated using:

$$R_n = R_s \cdot (1 - \alpha) - R_{nl} \tag{4.19}$$

where R_s is the input solar radiation (in $MJ \cdot m^{-2} \cdot day^{-1}$), $\alpha = 0.08$ accounts for surface albedo, and R_{nl} is the net longwave radiation. Outgoing longwave radiation is the radiation emitted by earth. Following McMahon et al. (2013) to obtain R_{nl} one first calculates clear sky radiation R_{so} using:

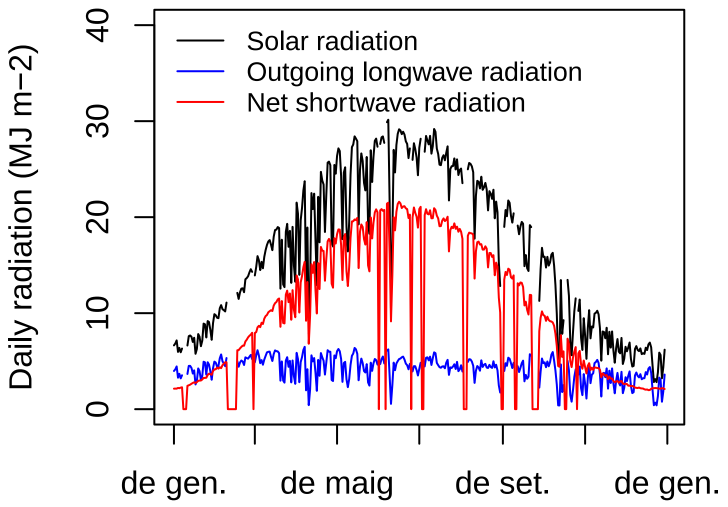
$$R_{so} = (0.75 + \cdot 0.00002 \cdot z) \cdot R_{pot} \tag{4.20}$$

where z is elevation and R_{pot} is potential radiation. R_{nl} is then calculated using:

$$R_{nl} = \sigma \cdot (0.34 - 0.14 \cdot \sqrt{e}) \cdot \frac{T_{\max}^4 + T_{\min}^4}{2} \cdot (1.35 \cdot \min(\frac{R_s}{R_{so}}, 1.0) - 0.35) \tag{4.21}$$

where e is the actual vapor pressure (kPa), $T_{\max}$ and $T_{\min}$ are the maximum and minimum temperatures (in Kelvin) and $\sigma = 4.903 \cdot 10^{-9} MJ \cdot K^{-4} \cdot m^{-2}$ is the Stephan-Boltzmann constant.

The following figure shows an example of radiation balance for a whole year for a single site (see functions `radiation_outgoingLongwaveRadiation()` and `radiation_netRadiation()`):



4.6 Diurnal trends in diffuse and direct radiation

Ecological studies sometimes require radiation information at a subdaily scale. This is particularly true for modeling studies that need to calculate canopy photosynthesis. Although meteoland has been designed to assist studies requiring meteorological data at daily scale, a function called `radiation_directDiffuseDay()` is provided to divide daily radiation into instantaneous direct and diffuse radiation. Values of instantaneous direct and diffuse radiation (shortwave and photosynthetic active radiation) are calculated following Spitters, Toussaint, and Goudriaan (1986). First, the ratio between daily diffuse and global radiation (R_d/R_g) is inferred from the ratio between daily potential and global radiation (R_g/R_{pot}):

$$R_d/R_g = 1 \qquad R_g/R_{pot} < 0.07 \qquad (4.22)$$

$$R_d/R_g = 1 - 2.3 \cdot (R_g/R_{pot} - 0.7)^2 \qquad 0.07 \leq R_g/R_{pot} < 0.35 \qquad (4.23)$$

$$R_d/R_g = 1.33 - 1.46 \cdot R_g/R_{pot} \qquad 0.35 \leq R_g/R_{pot} < 0.75 \qquad (4.24)$$

$$R_d/R_g = 0.23 \qquad 0.75 \leq R_g/R_{pot} \qquad (4.25)$$

In a clear day (e.g. not rainy) the ratio is modified to account for the circumsolar part of diffuse radiation:

$$R'_d/R_g = \frac{R_d/R_g}{1 + (1 - (R_d/R_g)^2) \cdot \cos^2(\pi/4 - \beta) \cdot \cos^3 \beta} \tag{4.26}$$

where β is the solar elevation angle. Otherwise $R'_d/R_g = R_d/R_g$. The daily diffuse shortwave radiation (R_d) is found by multiplying global radiation by the (modified) ratio:

$$R_d = R_g \cdot (R'_d/R_g) \tag{4.27}$$

The diurnal trend of the irradiance is derived from the daily global radiation and the daily course of potential (i.e. extra-terrestrial) radiation. If we assume that the atmospheric transmission is constant during the daylight period:

$$R_{g,s}/R_{pot,s} = R_g/R_{pot} \tag{4.28}$$

this leads to an estimation of the instantaneous global radiation (assuming compatible units):

$$R_{g,s} = R_g \cdot (R_{pot,s}/R_{pot}) \tag{4.29}$$

and the instantaneous diffuse and direct beam fluxes are estimated using:

$$R_{d,s} = R_d \cdot (R_{pot,s}/R_{pot}) \tag{4.30}$$

$$R_{b,s} = R_{g,s} - R_{d,s} \tag{4.31}$$

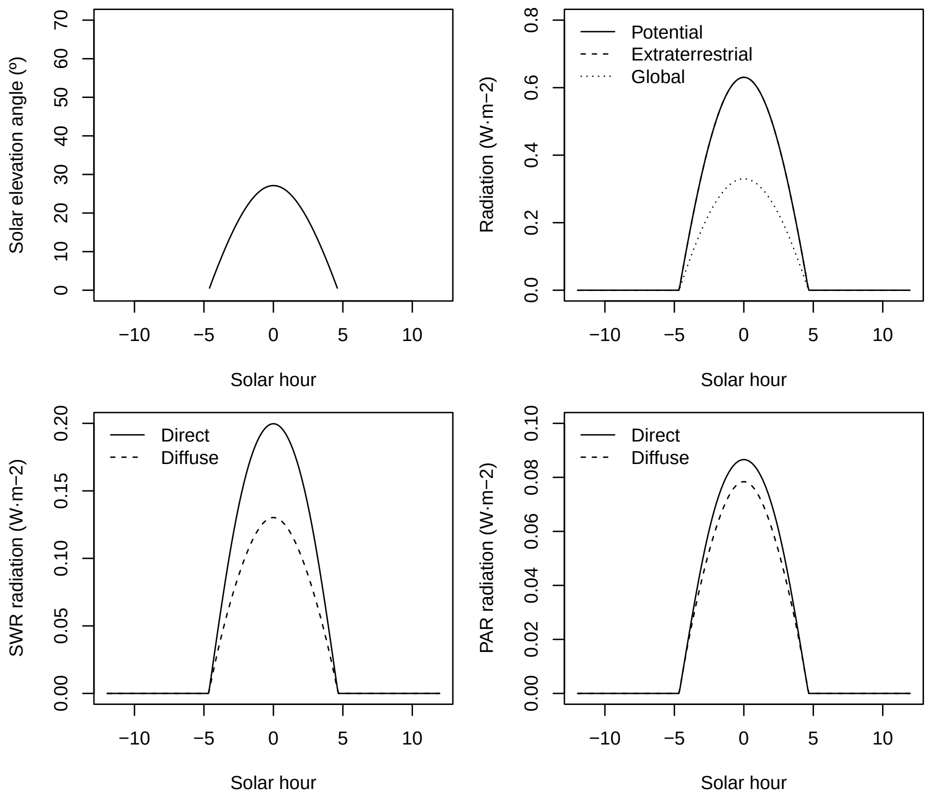
The whole procedure to calculate direct and diffuse radiation depends on the solar elevation angle, which changes through the day. Although R'_d/R_g is formulated as a ratio of daily values, the ratio needs to be calculated for every instant, as $R_{pot,s}$.

The procedure for photosynthetic active radiation (PAR) is similar. Daily PAR is assumed to be half of daily global radiation (i.e. $R_{PAR} = 0.5 \cdot R_g$. The scattered diffuse component of PAR is bigger than that of global radiation, and the ratio of diffuse over total PAR radiation is:

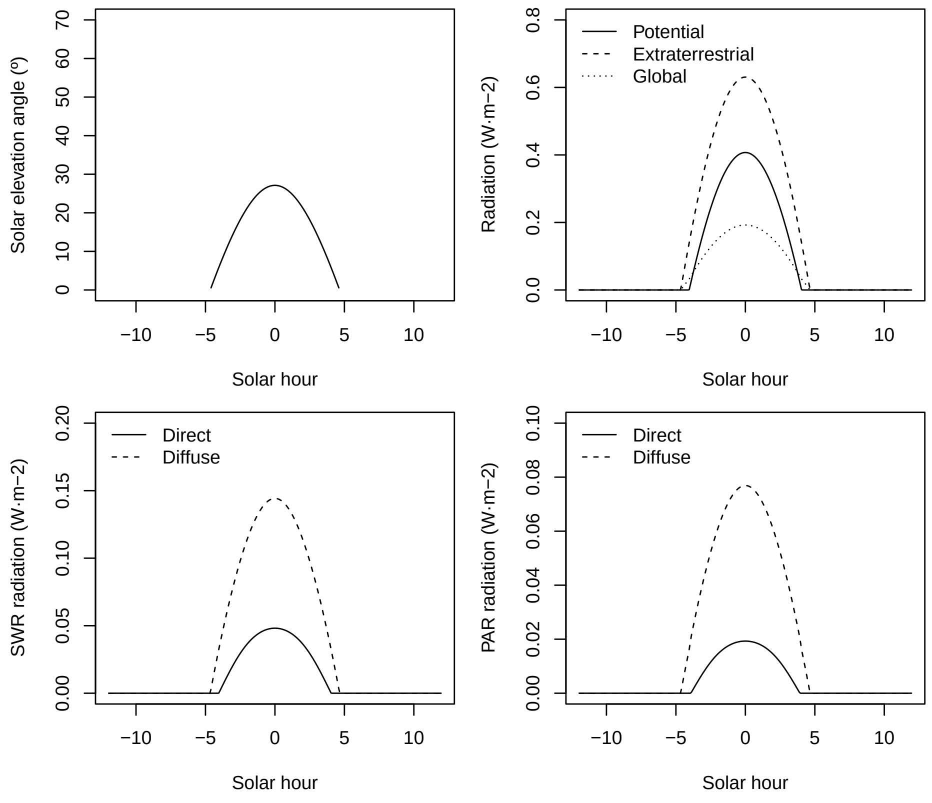
$$R_{PAR,d}/R_{PAR} = \left[1 + 0.3 \cdot (1 - (R_d/R_g)^2)\right] \cdot (R'_d/R_g) \tag{4.32}$$

The ratio $R_{PAR,d}/R_{PAR}$ is used to determine daily diffuse PAR and the calculation of instant rates are the same as for global radiation.

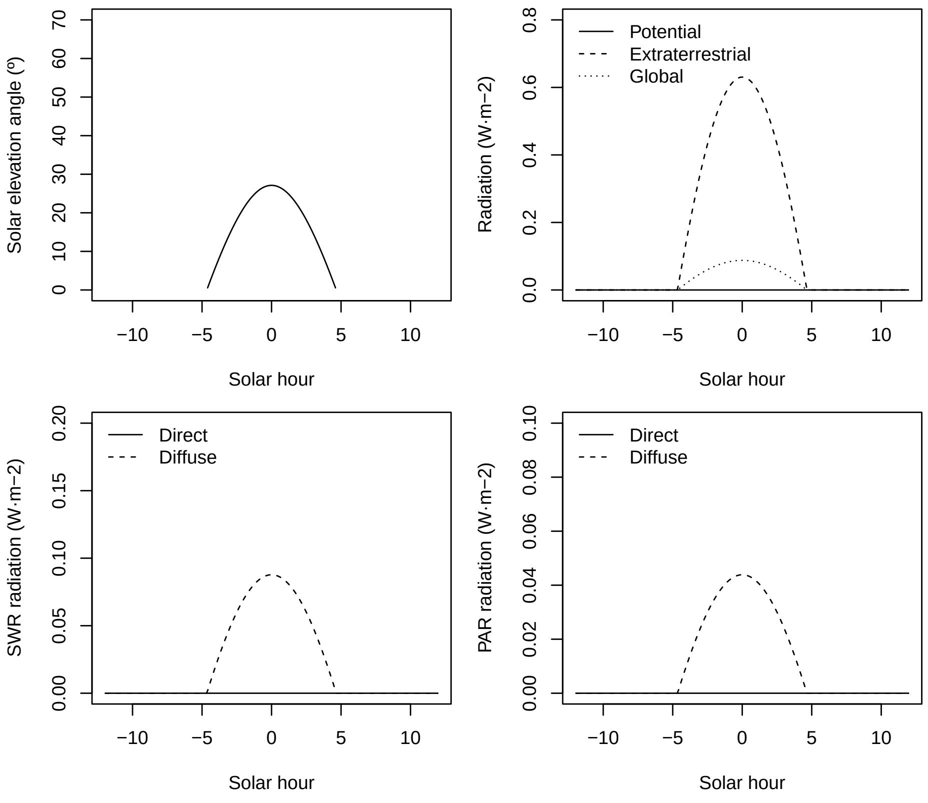
To illustrate the above calculations, we assume a target location in a flat terrain located at 42ºN latitude and 100 m.a.s.l, having 7.2 MJ $\cdot$ m^{-2} of daily global radiation on the 2001/January/15 in a clear day, the hourly variation in solar elevation, potential/global radiation and diffuse/direct light for PAR and SWR would be:



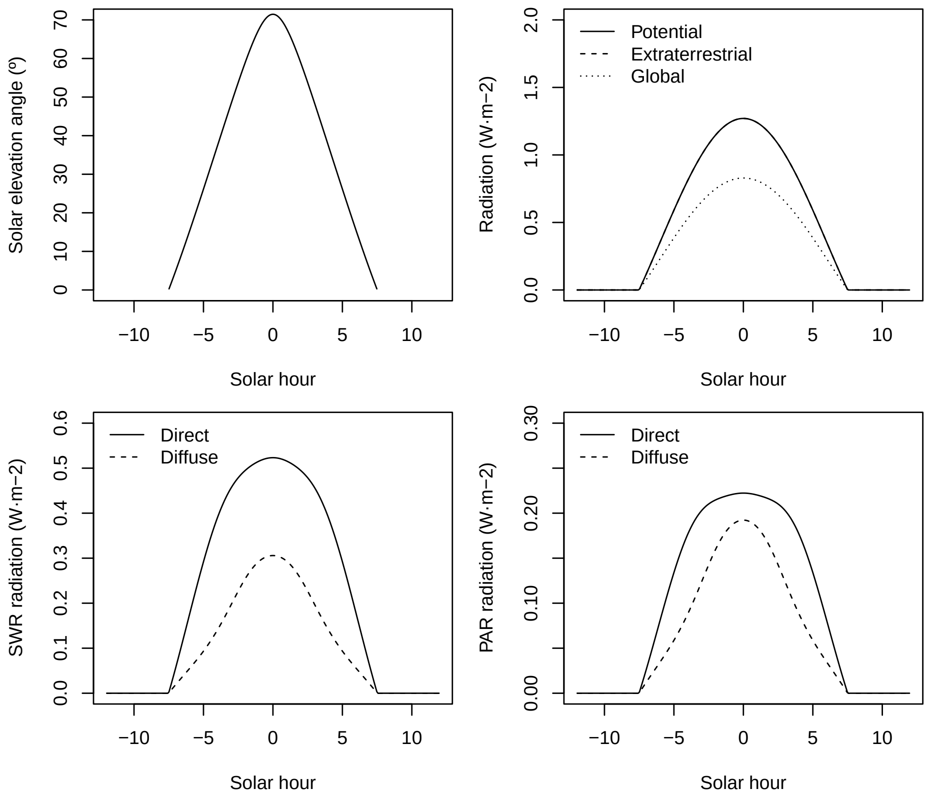
For a flat surface, the extraterrestrial radiation is the same that potential radiation, so that direct and diffuse light start and end at the same hours. If the same plot was on a north-facing slope of 10 degrees the daily global radiation would be 4.06 $MJ \cdot m^{-2}$ and the extraterrestrial radiation is larger than potential (i.e., accounting for topography) radiation, which has been reduced because of the orientation of the slope:



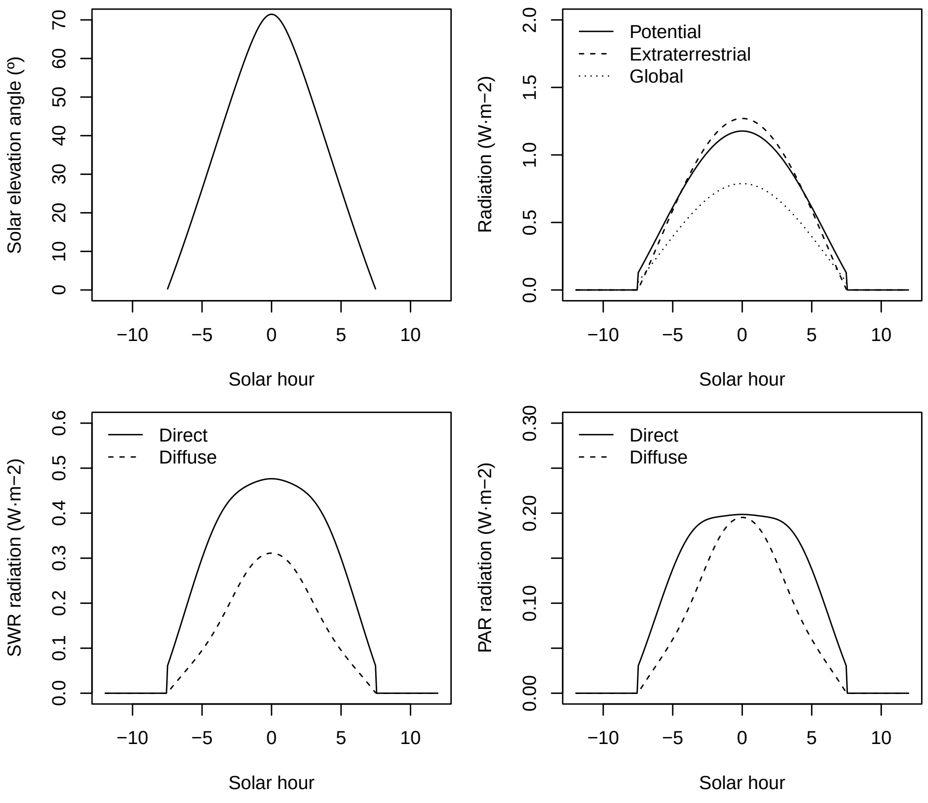
In this situation, diffuse light should follow extraterrestrial radiation hours whereas direct light should follow potential radiation hours. Finally, if the same plot was on a north-facing slope of 30 degrees the daily global radiation would be 1.96 $MJ\mathring{u}m^{-2}$ and potential radiation would be 0. In this situation all radiation should correspond to diffuse radiation:



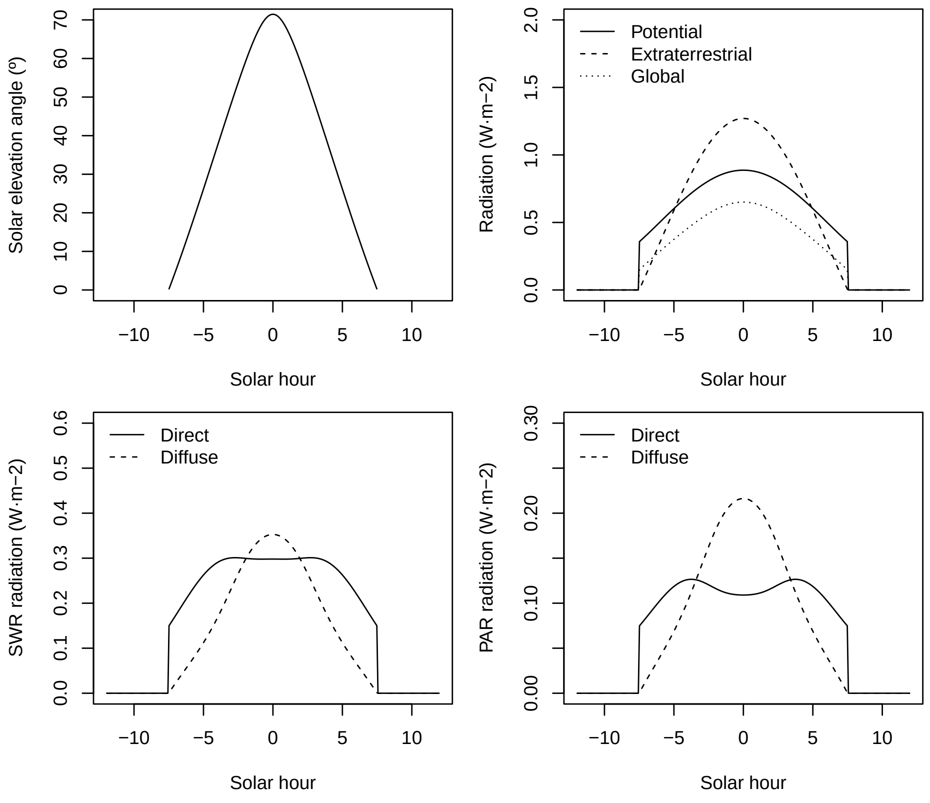
Let us now set the date to 15th of june, the flat surface would receive 27.7 $MJ\mathring{u}m^{-2}$ divided into:



whereas the 10 degree north-facing slope would receive 27.3 $MJ \mathring{u} m^{-2}$ divided into:



and the 30 degree slope would have 24.2 $MJ\mathring{u}m^{-2}$ and a repartition:


Solar elevation angle (º)
Solar hour
Radiation (W·m−2)
Potential
Extraterrestrial
Global
Solar hour
SWR radiation (W·m−2)
Direct
Diffuse
Solar hour
PAR radiation (W·m−2)
Direct
Diffuse
Solar hour

Chapter 5

Statistical correction of weather data

Statistical correction 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 correction 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).

5.1 Correction methods

Users of **meteoland** can choose between three different types of corrections:

- *Unbiasing*: consists in subtracting, from the series to be corrected, the average difference between the two series for the reference period (Déqué 2007). Let x_i be the value of the variable of the more accurate (e.g. local) series for a given day i and u_i the corresponding value for the less accurate series (e.g., climate model output). The bias, θ, is the average difference over all n days of the reference period:

$$\theta = \sum_{i}^{n} (u_i - x_i)/n \qquad (5.1)$$

  The bias calculated in the reference period is then subtracted from the value of u for any day of the period of interest.

- *Scaling*: A slope is calculated by regressing u on x through the origin (i.e. zero intercept) using data of the reference period. The slope can then be used as scaling factor to multiply the values of u for any day of the period of interest.
- *Empirical quantile mapping*: Due to its distributional properties, neither multiplicative or additive factors are appropriate for daily precipitation (Gudmundsson et al. 2012; Ruffault et al. 2014). In this case, it has been recommended to compare the empirical cumulative distribution function (CDF) of the two series for the reference period (Déqué 2007). The empirical CDFs of x and u for the reference period are approximated using tables of empirical percentiles, and this mapping is used to correct values of u for the period of interest:

  $$c_d = ecdf_x^{-1}(ecdf_u(u_d)) \tag{5.2}$$

  where $ecdf_x$ and $ecdf_u$ are the empirical CDFs of x and u respectively. Values between percentiles are approximated using linear interpolation. A difficulty arises for quantile mapping when the variables bounded by zero, such as precipitation. As the models tend to drizzle (or may have lower frequency of precipitation events), the probability of precipitation in the model may be greater or lower than that observed. To correct this, when model precipitation is zero an observed value is randomly chosen in the interval where the observed cumulative frequency is less than or equal to the probability of no precipitation in the model. This procedure ensures that the probability of precipitation after correction is equal to that observed (Boé et al. 2007).

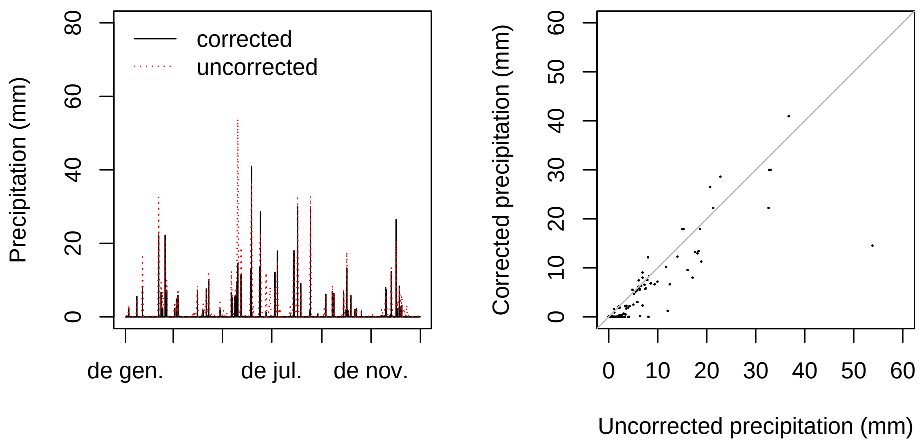
For each target location to be processed, the correction routine first determines which is the nearest climate model cell and extracts its weather data series for the reference period and the period of interest. Then, the correction method chosen by the user for each variable is applied. Statistical corrections are done for each of the twelve months separately to account for seasonal variation of distributional differences (Ruffault et al. 2014).

5.2 Default approaches by variable

Although users can choose their preferred correction method for each variable, meteoland has default approaches.

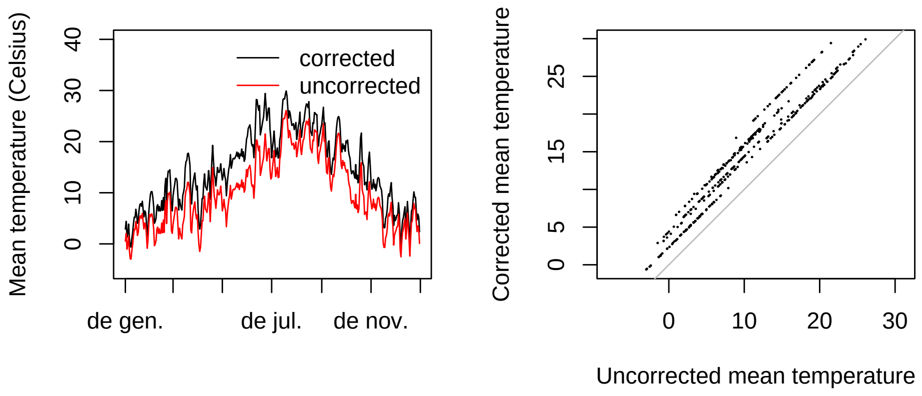
Precipitation

By default, correction of precipitation is done using empirical quantile mapping. The following figures show the correction of RCM precipitation predictions for 2023 using interpolated data from 2002-2003 as observations for the reference period. Note that at least 15 years of observations (and not two!) would be needed for a correct estimation of monthly CDFs.



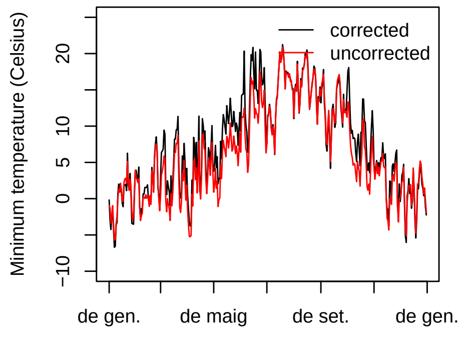
Mean temperature

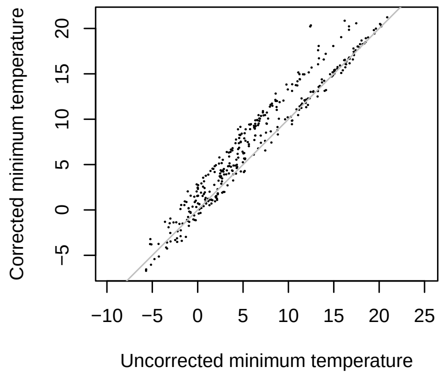
Unbiasing method is used by default to correct mean temperature. The following figures show the correction of RCM mean temperature predictions for 2023 using interpolated data from 2002-2003 as observations for the reference period:

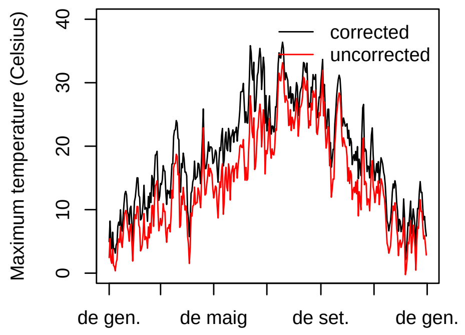


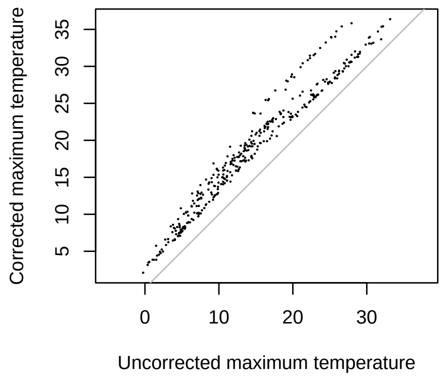
Minimum and maximum temperatures

To correct minimum (respectively maximum) temperature values, by default scaling is applied to the difference between minimum (resp. maximum) temperature and mean temperature.



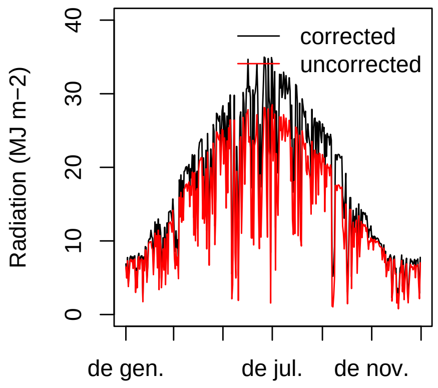


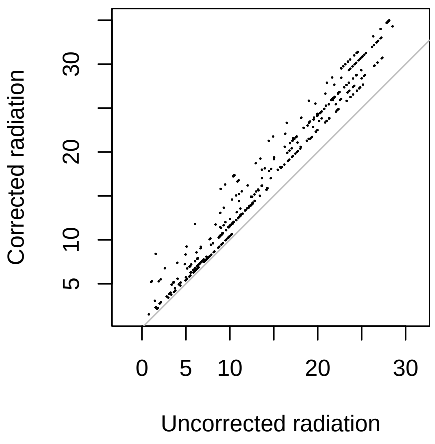




Radiation

Radiation is by default corrected using the unbiasing procedure. The following figures show the correction of RCM radiation predictions for 2023 using interpolated data from 2002-2003 as observations for the reference period:





Relative humidity

Mean relative humidity is first transformed to specific humidity, the unbiasing method is applied by default to this variable and the result is back transformed to mean, minimum and maximum relative humidity using the previously corrected series of mean, maximum and minimum temperature, respectively.

Wind speed

By default, wind speed is corrected using the scaling method. Since historic wind data is often not available, however, if wind speed data is missing the coarse-scale wind estimate is taken directly without correction.

Chapter 6

Estimation of potential evapo-transpiration

Package **meteoland** allows calculating daily potential evapo-transpiration (PET) using Penman's formulation (Penman 1948, 1956) or Penman-Monteith formulation. PET is automatically calculated after meteorological data have been interpolated (i.e. within functions `interpolationpoints()`, `interpolationpixels()` and `interpolationgrid()`) or bias-corrected (i.e. within function `correctionpoint()` or `correctionpoints()`), but PET values can also be calculated for a single point using functions `penman()` or `penmanmonteith()`. For other formulations of PET, the reader is referred to the R package **Evapotranspiration**.

6.1 Penman formulation

Penman (1948) proposed an equation to calculate daily potential evaporation that combined an energy equation based on net incoming radiation with an aerodynamic approach. The Penman or Penman combination equation is:

$$E_{pot} = \frac{\Delta}{\Delta + \gamma} \cdot \frac{R_n}{\lambda} + \frac{\lambda}{\Delta + \lambda} \cdot E_a \tag{6.1}$$

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 ($kPa{\cdot}^{\circ}C^{-1}$) at air temperature, γ is the psychrometric constant ($kPa{\cdot}^{\circ}C^{-1}$), 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):

$$E_a = f(u) \cdot D = f(u) \cdot (v_a^* - v_a) \tag{6.2}$$

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):

$$f(u) = 1.313 + 1.381 \cdot u \tag{6.3}$$
$$f(u) = 2.626 + 1.381 \cdot u \tag{6.4}$$

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

$$PET \simeq 0.047 \cdot R_s \cdot (T_a + 9.5)^{0.5} - 2.4 \cdot (\frac{R_s}{R_{pot}})^2 + 0.09 \cdot (T_a - 20) \cdot (1 - \frac{RH_{mean}}{100}) \tag{6.5}$$

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_{pot} is the potential (i.e. extraterrestrial) solar radiation (in $MJ \cdot m^{-2} \cdot day^{-1}$) and RH_{mean} is the mean relative humidity (in percent).

6.2 Penman-Monteith formulation

The Penman-Monteith combination equation:

$$E_{pot} = \frac{1}{\lambda} \cdot \frac{\Delta \cdot R_n + D \cdot (\rho \cdot C_p / r_a)}{\Delta + \gamma \cdot (1 + r_c / r_a)} \tag{6.6}$$

where D is the vapour pressure deficit (in kPa), Δ is the slope of the saturated vapor pressure (in $Pa \cdot K^{-1}$), γ is the psychrometer constant (in $kPa \cdot K^{-1}$), λ is the latent heat vaporization of water (in $MJ \cdot kg^{-1}$) and C_p is the specific heat of air (in $MJ \cdot kg^{-1} \cdot K^{-1}$). r_c is the canopy resistance (in $s \cdot m^{-1}$). For simplicity, aerodynamic resistance (r_a) is currently set to $r_a = 208.0/u$ where u is the input wind speed.

Chapter 7

Weather generation

Since version 0.8.6, **meteoland** incorporates the possibility of generating stochastic weather series. Stochastic weather generators are algorithms that produce series of synthetic daily weather data. The parameters of the model are conditioned on existing meteorological records to ensure the characteristics of input weather series emerge in the daily stochastic process. The weather generation approaches available in **meteoland** are intended to be used to generate daily series of the same length as the input. It can be understood as a bootstrap resampling algorithm that tries to preserve some properties of the original weather series. The approach implemented in **meteoland** can be applied to any spatial structure (points/pixels/grid) and it preserves the spatial correlation and multivariate covariance structure of weather series (because it works on area-averaged weather and the chosen resampled days are applied to all points/pixels).

Two modes of weather generation are offered:

1. Unconditional weather generation is based on a first order Markov chain (MC) to simulate a series of precipitation states (dry/wet/extreme wet) and a K-nearest neighbor (k-NN) algorithm to select a pair of consecutive days with the same transition and similar weather for the first day.
2. Conditional weather generation couples the former generation algorithm with a second algorithm to generate multiyear variation.

7.1 Unconditional weather generation algorithm

The algorithm is based on Apipattanavis et al. (2007) and combines a Markov Chain (MC) to generate the sequence of precipitation states with a k-nearest neighbor (k-NN) bootstrap resampler to generate multivariate and multisite weather variables. The MC is used to better represent wet and dry spell statistics while the k-NN bootstrap resampler preserves the covariance structure between weather variables and across space.

The weather generation approach is based on the common practice of first simulating precipitation occurence as a chain-dependent process. A three-state (dry / wet / extremely wet) Markov chain (MC) of order 1 is used to simulate an (area-averaged) precipitation state series. Nine transition probabilities are fit to the (area-averaged) input precipitation state series by month using maximum likelihood. A threshold of 0.3 mm (by default) is chosen to distinguish between dry and wet days, while the 80th percentile of precipitation (on wet days) is used as a threshold for extremely wet conditions. The fitted MC can be used to simulate (area-averaged) precipitation state series. A k-NN resampling algorithm of lag-1 is used to generate the values for all the weather variables, including disaggregation to L locations. The k-NN bootstrap resampler algorithm follows six steps (description adapted from Steinschneider et al. 2013):

1. Let $\bar{\mathbf{x}}_{t-1}$ be a vector of (area-averaged) weather variables already simulated for day $t-1$. Also assume that the MC has simulated precipitation state series for days $t-1$ and t.
2. Partition the historic record to find all pairs of consecutive days in a 7-day window (by default) centered on the day of the year of day t (i.e. if t is 15 January, then the window includes days from 12 to 18 January) that has the same sequence of (area-averaged) precipitation states simulated by the MC for days $t-1$ and t. Assume there are Q such pairs.
3. Compute the distance between vector $\bar{\mathbf{x}}_{t-1}$ and the vectors $\bar{\mathbf{x}}_q^1$ corresponding to the first day of each q pair of consecutive days. The weighted Euclidean distance d_q between vector $\bar{\mathbf{x}}_{t-1}$ and $\bar{\mathbf{x}}_q^1$ is calculated as:

$$d_q = \sqrt{\sum_{i=1}^{r} w_i \cdot (\bar{x}_{i,t-1} - \bar{x}_{i,q}^1)^2} \tag{7.1}$$

   where w_i is set to the inverse of the standard deviation of the ith weather variable. Here mean temperature and precipitation are used and we force a ten-fold weight for precipitation compared to temperature.
4. Order the distances d_q from smallest to largests and select the k smallest distances, where $k = \sqrt{Q}$. These corresponding k neighbors are assigned resapling weights K using a discrete kernel function:

$$K[j] = \frac{1/j}{\sum_{j=1}^{k} 1/j} \tag{7.2}$$

   where j indexes the k neighbors.
5. Sample one of the k neighbors and record the historic date associated with that selected neigbor. Then, use vectors of weather variables on the

successive day to the recorded date for each of the L locations to simulate, multisite, weather for day t.

6. Repeat steps 1-5 for all days of the simulation.

To begin the resampling algorithm and generate initial (area-averaged) $\bar{\mathbf{x}}_1$ values for all weather variables, data from a random day from the simulation starting month is selected from the historic record that is consistent with the first precipitation state simulated by the MC.

7.2 Conditional weather generation

The algorithm described in the previous section allows generating intra-annual (i.e. seasonal) weather variation because of monthly calibration of the MC and k-NN selection of pairs of days on the basis of their day of the year. However, year-to-year variation in annual precipitation or mean temperature will arise due to stochasticity only. Multi-year variation (i.e. low-frequency periodic variations or long-term trends) cannot be simulated. Two alternative approaches are offered in **meteoland** to condition stochastic weather generation on multi-year variation.

1. Annual precipitation can be conditioned by using a stationary auto-regressive (ARIMA) model annual precipitation and then using a k-nearest neighbor algorithm (similar to that of the previous section) to select years of the original weather series with annual precipitation similar to the simulated one and produce a bootstrap resample of weather data to train the MC-KNN algorithm (Steinschneider and Brown 2013). This option is recommended if low-frequency variation of annual precipitation is to be acounted for in long series.

2. Multi-year temperature or precipitation trends of the original series can be preserved by using a moving window. Each target year, a window around the target year is used to subset the original series. Annual precipitation to be simulated is then conditioned using a lognormal random trial of the precipitation corresponding to the selected years. Then, the same k-NN algorithm is used to produce a bootstrap resample of weather data from the years included in the moving window. This strategy is recommended to generate multiple stochastic series from (already corrected) climate change projections, as it preserves long-term trends.

Chapter 8

Miscellaneous functions

8.1 Downloading data from weather station networks

National meteorological agencies are increasingly adopting an open data philosophy. Package **meteoland** is able to retrieve data from three agencies:

1. The Spanish meteorological bureau (Agencia Española de Meteorología, AEMET)
2. The Catalan meteorology service (Servei Català de Meteorologia, SMC).
3. The Galician meteorology agency (MeteoGalicia, MG)

Functions that retrieve AEMET, SMC data use their OpenData application programming interface (API) and API keys need to be obtained from AEMET or SMC. These are:

- `downloadAEMEThistoricalstationlist()` : Gets the list of AEMET stations from which historical daily meteorological data is available.
- `downloadAEMEThistorical()` : Downloads historical daily meteorological data corresponding to a input set of AEMET stations and a given period.
- `downloadAEMETcurrentday()` : Downloads the last 24h of meteorological data corresponding to a input set of AEMET stations.
- `downloadSMCvarmetadata()`: Downloads the definition of meteorological variables, their units and codes for SMC.
- `downloadSMCcurrentday()` : Downloads the last 24h of meteorological data corresponding to a input set of SMC stations.
- `downloadSMChistorical()` : Downloads historical daily meteorological data corresponding to a input set of SMC stations and a given period.

Similar functions are provided to access data from MG without needing any key:

- `downloadMGstationlist()` : Gets information of the list of MG stations that are presently in function.
- `downloadMGcurrentday()` : Downloads the last 24h of meteorological data corresponding to a input set of MG stations.
- `downloadMGhistorical()` : Downloads historical daily meteorological data corresponding to a input set of MG stations and a given period.

Please, acknowledge the corresponding administration as source of information when using data downloaded by these functions.

IMPORTANT NOTE: The package functions listed above will not be maintained in releases of **meteoland** after version 1.0.1. Instead, the user is strongly recommended to use package **meteospain**.

8.2 Reshaping data obtained from other packages

meteoland provides functions to facilitate reshaping weather data acquired using other R packages into the meteoland format. So far, three packages are supported, **meteospain**, **worldmet** and **weathercan**. The corresponding data reshape functions are called `reshapemeteospain()`, `reshapeworldmet()` and `reshapeweathercan()`, respectively.

8.3 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 (http://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 (Sanjuan, Brun, and Cort 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.

8.4 Physical utility functions

Several utility functions are included in the package corresponding to physical calculations:

- `utils_atmosphericPressure()`: Atmospheric pressure P_{atm} in kPa from elevation z in m.

$$P_{atm}(z) = 101.32500 \cdot \left[1.0 - 2.2569 \cdot 10^{-5} \cdot z\right]^{5.2353} \tag{8.1}$$

- `utils_airDensity()`: Air density in $kg \cdot m^{-3}$ from temperature in Celsius and atmospheric pressure:

$$\rho_{air} = \frac{P_{atm}}{1.01 \cdot (T + 273.16) \cdot 0.287} \tag{8.2}$$

- `utils_saturationVP()`: Saturation water vapour pressure VP in kPa from temperature T in degrees Celsius:

$$VP(T) = 0.61078 \cdot e^{\left(\frac{17.269 \cdot T}{237.3 + T}\right)} \tag{8.3}$$

- `saturationVaporPressureCurveSlope()`: Saturation water vapour pressure curve slope s_{vp} in $kPa \cdot ^\circ C^{-1}$ from temperature T in degrees Celsius:

$$s_{vp}(T) = 4098.0 \cdot \frac{0.6108 \cdot e^{(17.27 \cdot T)/(T+237.3)}}{(T+237.3)^2} \tag{8.4}$$

- `utils_averageDailyVP()`: Average daily water vapour pressure vp_{atm} in kPa calculated from minimum and maximum temperatures and relative humidities:

$$vp_{atm} = \frac{VP(T_{min}) \cdot (RH_{max}/100) + VP(T_{max}) \cdot (RH_{min}/100)}{2} \tag{8.5}$$

- `utils_latentHeatVaporisation()`: Latent heat of vaporisation λ_v in $MJ \mathring{u} kg^{-1}$ from temperature in degrees Celsius:

$$\lambda_v(T) = (2.5023 - (0.00243054 \cdot T)) \tag{8.6}$$

- `utils_latentHeatVaporisationMol()`: Latent heat of vaporisation λ_v in $J \mathring{u} mol^{-1}$ from temperature in degrees Celsius:

$$\lambda_v(T) = (2.5023 \cdot 10^6 - (2430.54 \cdot T)) \cdot 0.018 \tag{8.7}$$

- `utils_psychrometricConstant()`: Psychrometric constant in $kPa \mathring{u} ^\circ C^{-1}$ from temperature in degrees Celsius and atmospheric pressure in kPa:

$$\gamma_v = \frac{0.00163 \cdot P_{atm}}{\lambda_v(T)} \tag{8.8}$$

Apipattanavis, Somkiat, Guillermo Podestá, Balaji Rajagopalan, and Richard W. Katz. 2007. "A semiparametric multivariate and multisite weather generator." *Water Resources Research* 43 (11). https://doi.org/10.1029/2006WR005714.

Boé, J, L Terray, F Habets, and E Martin. 2007. "Statistical and dynamical downscaling of the Seine basin climate for hydro-meteorological studies." *International Journal of Climatology* 27: 1643–55. https://doi.org/10.1002/joc.1602.

Danby, J. M. 1988. *Fundamentals of Celestial Mechanics.* 2nd ed. Richmond, VA: Willmann-Bell.

De Cáceres, Miquel, Nicolas Martin-StPaul, Marco Turco, Antoine Cabon, and Victor Granda. 2018. "Estimating daily meteorological data and downscaling climate models over landscapes." *Environmental Modelling & Software* 108 (August): 186–96. https://doi.org/10.1016/j.envsoft.2018.08.003.

Déqué, Michel. 2007. "Frequency of precipitation and temperature extremes over France in an anthropogenic scenario: Model results and statistical correction according to observed values." *Global and Planetary Change* 57 (1-2): 16–26. https://doi.org/10.1016/j.gloplacha.2006.11.030.

Garnier, B. J., and Atsumu Ohmura. 1968. "A method of calculating the direct shortwave radiation income of slopes." *Journal of Applied Meteorology* 7 (5): 796–800. https://doi.org/10.1175/1520-0450(1968)007%3C0796:AMOCTD%3E2.0.CO;2.

Gudmundsson, L., J. B. Bremnes, J. E. Haugen, and T. Engen-Skaugen. 2012. "Technical Note: Downscaling RCM precipitation to the station scale using statistical transformations - A comparison of methods." *Hydrology and Earth System Sciences* 16 (9): 3383–90. https://doi.org/10.5194/hess-16-3383-2012.

McMahon, T. A., M. C. Peel, L. Lowe, R. Srikanthan, and T. R. McVicar. 2013. "Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis." *Hydrology & Earth System Sciences* 17: 1331–63. https://doi.org/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: 129–45.

———. 1956. "Evaporation: An introductory survey." *Netherlands Journal of Agricultural Science* 4: 9–29.

Reda, Ibrahim, and Afshin Andreas Nrel. 2008. "Solar Position Algorithm for Solar Radiation Applications (Revised)." *Nrel/Tp-560-34302*, no. January: 1–56. https://doi.org/10.1016/j.solener.2003.12.003.

Ruffault, Julien, Nicolas K Martin-StPaul, Carole Duffet, Fabien Goge, and Florent Mouillot. 2014. "Projecting future drought in Mediterranean forests: bias correction of climate models matters!" *Theoretical and Applied Climatology* 117 (1-2): 113–22. https://doi.org/10.1007/s00704-013-0992-z.

Sanjuan, Gemma, Carlos Brun, and Ana Cort. 2014. "Wind field uncertainty in forest fire propagation prediction." *Procedia Computer Science* 29: 1535–45. https://doi.org/10.1016/j.procs.2014.05.139.

Spitters, C. J. T., H. A. J. M. Toussaint, and J. Goudriaan. 1986. "Separating the diffuse and direct components of global radiation and its implications for modeling canopy photosynthesis. I. Components of incoming radiation." *Agricultural and Forest Meteorology* 38: 231–42. https://doi.org/10.1016/0168-1923(86)90060-2.

Steinschneider, Scott, and Casey Brown. 2013. "A semiparametric multivariate, multisite weather generator with low-frequency variability for use in climate risk assessments." *Water Resources Research* 49 (11): 7205–20. https://doi.org/10.1002/wrcr.20528.

Thornton, P. E., and S. W. Running. 1999. "An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation." *Agricultural and Forest Meteorology* 93: 211–28. https://doi.org/10.1016/S0168-1923(98)00126-9.

Thornton, Peter E., Steven W. Running, and Michael a. White. 1997. "Generating surfaces of daily meteorological variables over large regions of complex terrain." *Journal of Hydrology* 190 (3-4): 214–51. https://doi.org/10.1016/S0022-1694(96)03128-9.

Tymstra, C, R W Bryce, B M Wotton, S W Taylor, and O B Armitage. 2010. *Development and Structure of Prometheus : the Canadian Wildland Fire Growth Simulation Model*. https://doi.org/ISBN%20978-1-100-14674-4.

Valiantzas, John D. 2006. "Simplified versions for the Penman evaporation equation using routine weather data." *Journal of Hydrology* 331 (3-4): 690–702. https://doi.org/10.1016/j.jhydrol.2006.06.012.