

Package: solaR2 (via r-universe)

February 19, 2025

Type Package

Title Radiation and Photovoltaic Systems

Version 0.11

Encoding UTF-8

Description Provides tools for calculating solar geometry, solar radiation on horizontal and inclined planes, and simulating the performance of various photovoltaic (PV) systems. Supports daily and intradaily irradiation data, enabling detailed analysis of grid-connected and water-pumping PV systems, including shading effects and solar angle calculations.

URL <https://solarization.github.io/solaR2/>

BugReports <https://github.com/solarization/solaR2/issues>

License GPL-3

LazyData yes

Depends R (>= 4.0.0), lattice

Imports data.table, latticeExtra, RColorBrewer, httr2, graphics, grDevices, stats, methods, utils

Suggests zoo, sp, raster, rasterVis, tdr, meteoForecast, jsonlite, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/pak/sysreqs libjpeg-dev libpng-dev libssl-dev

Repository <https://solarization.r-universe.dev>

RemoteUrl <https://github.com/solarization/solar2>

RemoteRef HEAD

RemoteSha e99562ec6ecb34546036f820cc210ada4caac3e6

Contents

solarR2-package	3
A1_calcSol	5
A2_calcG0	7
A3_calcGef	10
A4_prodGCPV	12
A5_prodPVPS	17
A6_calcShd	19
A7_optimShd	20
A8_Meteo2Meteo	24
A8_readBD	26
A8_readG0dm	28
A8_readSIAR	29
B1_Meteo-class	30
B2_Sol-class	31
B3_G0-class	32
B4_Gef-class	33
B5_ProdGCPV-class	35
B6_ProdPVPS-class	36
B7_Shade-class	38
C_corrFdKt	39
C_fBTd	41
C_fBTi	43
C_fCompD	44
C_fCompI	45
C_fInclin	47
C_fProd	49
C_fPump	52
C_fSolD	53
C_fSolI	55
C_fSombra	57
C_fTemp	60
C_fTheta	61
C_HQCurve	62
C_local2Solar	63
C_NmgPVPS	65
C_sample2Diff	66
C_solarAngles	68
C_utils-angle	70
C_utils-time	71
D_as.data.tableD-methods	72
D_as.data.tableI-methods	73
D_as.data.tableM-methods	74
D_as.data.tableY-methods	76
D_compare-methods	77
D_getData-methods	78
D_getG0-methods	78

D_getLat-methods	79
D_indexD-methods	79
D_indexI-methods	80
D_levelplot-methods	80
D_Losses-methods	81
D_mergesolaR-methods	82
D_shadeplot-methods	83
D_window-methods	83
D_writeSolar-methods	85
D_xyplot-methods	86
E_aguiar	87
E_helios	88
E_prodEx	88
E_pumpCoef	89
E_SIAR	90
E_solaR.theme	90

Index 91

solaR2-package	<i>Solar Radiation and Photovoltaic Systems with R version 2</i>
----------------	--

Description

The solaR2 package allows for reproducible research both for photovoltaics (PV) systems performance and solar radiation. It includes a set of classes, methods and functions to calculate the sun geometry and the solar radiation incident on a photovoltaic generator and to simulate the performance of several applications of the photovoltaic energy. This package performs the whole calculation procedure from both daily and intradaily global horizontal irradiation to the final productivity of grid-connected PV systems and water pumping PV systems.

Details

solaRd is designed using a set of S4 classes whose core is a group of slots with multivariate time series. The classes share a variety of methods to access the information and several visualization methods. In addition, the package provides a tool for the visual statistical analysis of the performance of a large PV plant composed of several systems.

Although solaRd is primarily designed for time series associated to a location defined by its latitude/longitude values and the temperature and irradiation conditions, it can be easily combined with spatial packages for space-time analysis.

Please note that this package needs to set the timezone to UTC. Every ‘data.table’ object created by the package will have an index with this time zone as a synonym of mean solar time..

You can check it after loading solaR2 with:

```
Sys.getenv('TZ')
```

If you need to change it, use:

```
Sys.setenv(TZ = 'YourTimeZone')
```

Index of functions and classes:

G0-class	Class "G0": irradiation and irradiance on the horizontal plane.
Gef-class	Class "Gef": irradiation and irradiance on the generator plane.
HQCurve	H-Q curves of a centrifugal pump
Meteo-class	Class "Meteo"
NmgPVPS	Nomogram of a photovoltaic pumping system
ProdGCPV-class	Class "ProdGCPV": performance of a grid connected PV system.
ProdPVPS-class	Class "ProdPVPS": performance of a PV pumping system.
Shade-class	Class "Shade": shadows in a PV system.
Sol-class	Class "Sol": Apparent movement of the Sun from the Earth
aguiar	Markov Transition Matrices for the Aguiar etal. procedure
as.data.tableD	Methods for Function as.data.frameD
as.data.tableI	Methods for Function as.data.frameI
as.data.tableM	Methods for Function as.data.frameM
as.data.tableY	Methods for Function as.data.frameY
calcG0	Irradiation and irradiance on the horizontal plane.
calcGef	Irradiation and irradiance on the generator plane.
calcShd	Shadows on PV systems.
calcSol	Apparent movement of the Sun from the Earth
compare	Compare G0, Gef and ProdGCPV objects
compareLosses	Losses of a GCPV system
corrFdKt	Correlations between the fraction of diffuse irradiation and the clearness index.
d2r	Conversion between angle units.
diff2Hours	Small utilities for difftime objects.
fBTd	Daily time base
fCompD	Components of daily global solar irradiation on a horizontal surface
fCompI	Calculation of solar irradiance on a horizontal surface
fInclin	Solar irradiance on an inclined surface
fProd	Performance of a PV system
fPump	Performance of a centrifugal pump
fSolD	Daily apparent movement of the Sun from the Earth
fSolI	Instantaneous apparent movement of the Sun from the Earth
fSombra	Shadows on PV systems
fTemp	Intradaily evolution of ambient temperature
fTheta	Angle of incidence of solar irradiation on a inclined surface

getData	Methods for function getData
getG0	Methods for function getG0
getLat	Methods for Function getLat
helios	Daily irradiation and ambient temperature from the Helios-IES database
hour	Utilities for time indexes.
indexD	Methods for Function indexD
indexI	Methods for Function indexI
levelplot-methods	Methods for function levelplot.
local2Solar	Local time, mean solar time and UTC time zone.
mergesolaR	Merge solaR objects
optimShd	Shadows calculation for a set of distances between elements of a PV grid connected plant.
prodEx	Productivity of a set of PV systems of a PV plant.
prodGCPV	Performance of a grid connected PV system.
prodPVPS	Performance of a PV pumping system
pumpCoef	Coefficients of centrifugal pumps.
readBD	Daily or intradaily values of global horizontal irradiation and ambient temperature from a local file or a data.frame.
readG0dm	Monthly mean values of global horizontal irradiation.
readSIAR	Meteorological data exported from the SIAR network
shadeplot	Methods for Function shadeplot
solaR.theme	solaR theme
window	Methods for extracting a time window
writeSolar	Exporter of solaR results
xyplot-methods	Methods for function xyplot in Package 'solaR'

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

A1_calcSol

Apparent movement of the Sun from the Earth

Description

Compute the apparent movement of the Sun from the Earth with the functions [fSolD](#) and [fSolI](#).

Usage

```
calcSol(lat, BTd, sample = 'hour', BTi,
        EoT = TRUE, keep.night = TRUE,
        method = 'michalsky')
```

Arguments

lat	Latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
BTd	Daily time base, a POSIXct object which may be the result of <code>fBTd</code> . It is not considered if BTi is provided.
sample	Increment of the intradaily sequence. It is a character string, containing one of "sec", "min", "hour". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s". It is used by <code>seq.POSIXt</code> . It is not considered if BTi is provided.
BTi	Intradaily time base, a POSIXct object to be used by <code>fSolI</code> . It may be the result of <code>fBTi</code> .
EoT	logical, if TRUE the Equation of Time is used. Default is TRUE.
keep.night	logical, if TRUE (default) the night is included in the time series.
method	character, method for the sun geometry calculations to be chosen from 'cooper', 'spencer', 'michalsky' and 'strous'. See references for details.

Value

A `Sol-class` object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Cooper, P.I., Solar Energy, 12, 3 (1969). "The Absorption of Solar Radiation in Solar Stills"
- Spencer, Search 2 (5), 172, <https://www.mail-archive.com/sundial@uni-koeln.de/msg01050.html>
- Strous: <https://www.aa.quae.nl/en/reken/zonpositie.html>
- Michalsky, J., 1988: The Astronomical Almanac's algorithm for approximate solar position (1950-2050), Solar Energy 40, 227-235
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

Examples

```
library("data.table")

BTd <- fBTd(mode <- 'serie')

lat <- 37.2
sol <- calcSol(lat, BTd[100])
print(as.data.tableD(sol))
```

```

library(lattice)
xyplot(as.data.tableI(sol))

solStrous <- calcSol(lat, BTd[100], method = 'strous')
print(as.data.tableD(solStrous))

solSpencer <- calcSol(lat, BTd[100], method = 'spencer')
print(as.data.tableD(solSpencer))

solCooper <- calcSol(lat, BTd[100], method = 'cooper')
print(as.data.tableD(solCooper))

```

A2_calcG0

*Irradiation and irradiance on the horizontal plane.***Description**

This function obtains the global, diffuse and direct irradiation and irradiance on the horizontal plane from the values of *daily* and *intradaily* global irradiation on the horizontal plane. It makes use of the functions [calcSol](#), [fCompD](#), [fCompI](#), [fBTd](#) and [readBDd](#) (or equivalent).

Besides, if information about maximum and minimum temperatures values are available it obtains a series of temperature values with [fTemp](#).

Usage

```

calcG0(lat, modeRad = 'prom', dataRad,
       sample = 'hour', keep.night = TRUE,
       sunGeometry = 'michalsky',
       corr, f, ...)

```

Arguments

lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
modeRad	<p>A character string, describes the kind of source data of the global irradiation and ambient temperature.</p> <p>It can be modeRad = 'prom' for monthly mean calculations. With this option, a set of 12 values inside dataRad must be provided, as defined in readG0dm.</p> <p>modeRad = 'aguiar' uses a set of 12 monthly average values (provided with dataRad) and produces a synthetic daily irradiation time series following the procedure by Aguiar et al. (see reference below).</p> <p>If modeRad = 'bd' the information of <i>daily</i> irradiation is read from a file, a data.table defined by dataRad, a zoo or a Meteo object. (See readBDd, dt2Meteo and zoo2Meteo for details).</p> <p>If modeRad = 'bdI' the information of <i>intradaily</i> irradiation is read from a file, a data.table defined by dataRad, a zoo or a Meteo object. (See readBDi, dt2Meteo and zoo2Meteo for details).</p>

dataRad	<ul style="list-style-type: none"> • If modeRad = 'prom' or modeRad = 'aguiar', a numeric with 12 values or a named list whose components will be processed with readG0dm. • If modeRad = 'bd' a character (name of the file to be read with readBDd), a <code>data.table</code> (to be processed with dt2Meteo), a zoo (to be processed with zoo2Meteo), a Meteo object, or a list as defined by readBDd, dt2Meteo or zoo2Meteo. The resulting object will include a column named Ta, with information about ambient temperature. • If modeRad = 'bdI' a character (name of the file to be read with readBDi), a <code>data.table</code> (to be processed with dt2Meteo), a zoo (to be processed with zoo2Meteo), a Meteo object, or a list as defined by readBDi, dt2Meteo or zoo2Meteo. The resulting object will include a column named Ta, with information about ambient temperature.
sample	character, containing one of "sec", "min", "hour". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s" (used by seq.POSIXt). It is not used when modeRad = "bdI".
keep.night	logical. When it is TRUE (default) the time series includes the night.
sunGeometry	character, method for the sun geometry calculations. See calcSol , fSolD and fSolI .
corr	<p>A character, the correlation between the fraction of diffuse irradiation and the clearness index to be used.</p> <p>With this version several options are available, as described in corrFdKt. For example, the FdKtPage is selected with corr = 'Page' while the FdKtCPR with corr = 'CPR'.</p> <p>If corr = 'user' the use of a correlation defined by a function f is possible.</p> <p>If corr = 'none' the object defined by dataRad should include information about global, diffuse and direct daily irradiation with columns named G0d, D0d and B0d, respectively (or G0, D0 and B0 if modeRad = 'bdI'). If corr is missing, then it is internally set to CPR when modeRad = 'bd', to Page when modeRad = 'prom' and to BRL when modeRad = 'bdI'.</p>
f	A function defining a correlation between the fraction of diffuse irradiation and the clearness index. It is only necessary when corr = 'user'
...	Additional arguments for fCompD or fCompI

Value

A G0 object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

- Aguiar, Collares-Pereira and Conde, "Simple procedure for generating sequences of daily radiation values using a library of Markov transition matrices", Solar Energy, Volume 40, Issue 3, 1988, Pages 269–279

See Also

[calcSol](#), [fCompD](#), [fCompI](#), [readG0dm](#), [readBDd](#), [readBDi](#), [dt2Meteo](#), [corrFdKt](#).

Examples

```
library("data.table")

G0dm <- c(2.766, 3.491, 4.494, 5.912, 6.989, 7.742, 7.919, 7.027, 5.369, 3.562, 2.814, 2.179)*1000;
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2,
       15.2)

g0 <- calcG0(lat = 37.2, modeRad = 'prom', dataRad = list(G0dm = G0dm, Ta = Ta))
print(g0)
xyplot(g0)

## Aguiar et al.

g0 <- calcG0(lat = 37.2, modeRad = 'aguiar', dataRad = G0dm)
print(g0)
xyplot(g0)

##Now the G0I component of g0 is used as
##the bdI argument to calcG0 in order to
##test the intradaily correlations of fd-kt

BDi = as.data.tableI(g0)
BDi$Ta = 25 ##Information about temperature must be contained in BDi

g02 <- calcG0(lat = 37.2,
              modeRad = 'bdI',
              dataRad = list(lat = 37.2, file = BDi),
              corr = 'none')

print(g02)

g03 <- calcG0(lat = 37.2,
              modeRad = 'bdI',
              dataRad = list(lat = 37.2, file = BDi),
              corr = 'BRL')

print(g03)

xyplot(Fd ~ Kt, data = g03, pch = 19, alpha = 0.3)
```

A3_calcGef

*Irradiation and irradiance on the generator plane.***Description**

This function obtains the global, diffuse and direct irradiation and irradiance on the generator plane from the values of *daily* or *intradaily* global irradiation on the horizontal plane. It makes use of the functions [calcG0](#), [fTheta](#), [fInclin](#). Besides, it can calculate the shadows effect with the [calcShd](#) function.

Usage

```
calcGef(lat,
        modeTrk = 'fixed',
        modeRad = 'prom',
        dataRad,
        sample = 'hour',
        keep.night = TRUE,
        sunGeometry = 'michalsky',
        corr, f,
        betaLim = 90, beta = abs(lat)-10, alpha = 0,
        iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE,
        modeShd = '',
        struct = list(),
        distances = data.table(),
        ...)
```

Arguments

lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
modeTrk	character, to be chosen from 'fixed', 'two' or 'horiz'. When modeTrk = 'fixed' the surface is fixed (inclination and azimuth angles are constant). The performance of a two-axis tracker is calculated with modeTrk = 'two', and modeTrk = 'horiz' is the option for an horizontal N-S tracker. Its default value is modeTrk = 'fixed'
modeRad, dataRad	Information about the source data of the global irradiation. See calcG0 for details.
sample, keep.night	See calcSol for details.
sunGeometry	character, method for the sun geometry calculations. See calcSol , fSold and fSolI .
corr, f	See calcG0 for details.
beta	numeric, inclination angle of the surface (degrees). It is only needed when modeTrk = 'fixed'.

betaLim	numeric, maximum value of the inclination angle for a tracking surface. Its default value is 90 (no limitation)
alpha	numeric, azimuth angle of the surface (degrees). It is measured from the south (alpha = 0), and it is negative to the east and positive to the west. It is only needed when modeTrk = 'fixed'. Its default value is alpha = 0
iS	integer, degree of dirtiness. Its value must be included in the set (1,2,3,4). iS = 1 corresponds to a clean surface while iS = 4 is the selection for a dirty surface. Its default value is 2.
alb	numeric, albedo reflection coefficient. Its default value is 0.2
modeShd, struct, distances	See calcShd for details.
horizBright	logical, if TRUE, the horizon brightness correction proposed by Reind et al. is used.
HCPV	logical, if TRUE the diffuse and albedo components of the <i>effective</i> irradiance are set to zero. HCPV is the acronym of High Concentration PV system.
...	Additional arguments for calcSol and calcG0

Value

A Gef object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Hay, J. E. and McKay, D. C.: Estimating Solar Irradiance on Inclined Surfaces: A Review and Assessment of Methodologies. *Int. J. Solar Energy*, (3):pp. 203, 1985.
- Martín, N. and Ruiz, J.M.: Calculation of the PV modules angular losses under field conditions by means of an analytical model. *Solar Energy Materials & Solar Cells*, 70:25–38, 2001.
- D. T. Reindl and W. A. Beckman and J. A. Duffie: Evaluation of hourly tilted surface radiation models, *Solar Energy*, 45:9-17, 1990.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[calcG0](#), [fTheta](#), [fInclin](#), [calcShd](#).

Examples

```
library("data.table")
```

```
lat <- 37.2
```

```

###12 Average days.

G0dm = c(2.766, 3.491, 4.494, 5.912, 6.989, 7.742, 7.919, 7.027, 5.369,
         3.562, 2.814, 2.179)*1000;
Ta = c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2,
       15.2)

##Fixed surface, default values of inclination and azimuth.

gef <- calcGef(lat = lat, modeRad = 'prom', dataRad = list(G0dm = G0dm, Ta = Ta))
print(gef)
xyplot(gef)

##Two-axis surface, no limitation angle.

gef2 <- calcGef(lat = lat, modeRad = 'prom',
               dataRad = list(G0dm = G0dm, Ta = Ta),
               modeTrk = 'two')
print(gef2)
xyplot(gef2)

struct = list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 8)
distances = data.table(Lew = 40, Lns = 30, H = 0)

gefShd <- calcGef(lat = lat, modeRad = 'prom',
                 dataRad = list(G0dm = G0dm, Ta = Ta),
                 modeTrk = 'two',
                 modeShd = c('area', 'prom'),
                 struct = struct, distances = distances)

print(gefShd)

```

A4_prodGCPV

Performance of a grid connected PV system.

Description

Compute every step from solar angles to effective irradiance to calculate the performance of a grid connected PV system.

Usage

```

prodGCPV(lat,
         modeTrk = 'fixed',
         modeRad = 'prom',
         dataRad,
         sample = 'hour',
         keep.night = TRUE,
         sunGeometry = 'michalsky',

```

```

corr, f,
betaLim = 90, beta = abs(lat)-10, alpha = 0,
iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE,
module = list(),
generator = list(),
inverter = list(),
effSys = list(),
modeShd = '',
struct = list(),
distances = data.table(),
...)

```

Arguments

lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
modeTrk	A character string, describing the tracking method of the generator. See calcGef for details.
modeRad, dataRad	Information about the source data of the global irradiation. See calcG0 for details.
sample, keep.night	See calcSol for details.
sunGeometry	character, method for the sun geometry calculations. See calcSol , fSolD and fSolI .
corr, f	See calcG0 for details.
betaLim, beta, alpha, iS, alb, horizBright, HCPV	See calcGef for details.
module	list of numeric values with information about the PV module, Vocn open-circuit voltage of the module at Standard Test Conditions (default value 57.6 volts.) Iscn short circuit current of the module at Standard Test Conditions (default value 4.7 amperes.) Vmn maximum power point voltage of the module at Standard Test Conditions (default value 46.08 amperes.) Imn Maximum power current of the module at Standard Test Conditions (default value 4.35 amperes.) Ncs number of cells in series inside the module (default value 96) Ncp number of cells in parallel inside the module (default value 1) CoefVT coefficient of decrement of voltage of each cell with the temperature (default value 0.0023 volts per celsius degree) TONC nominal operational cell temperature, celsius degree (default value 47).
generator	list of numeric values with information about the generator, Nms number of modules in series (default value 12) Nmp number of modules in parallel (default value 11)

inverter	list of numeric values with information about the DC/AC inverter, Ki vector of three values, coefficients of the efficiency curve of the inverter (default c(0.01, 0.025, 0.05)), or a matrix of nine values (3x3) if there is dependence with the voltage (see references). Pinv nominal inverter power (W) (default value 25000 watts.) Vmin, Vmax minimum and maximum voltages of the MPP range of the inverter (default values 420 and 750 volts) Gumb minimum irradiance for the inverter to start (W/m ²) (default value 20 W/m ²)
effSys	list of numeric values with information about the system losses, ModQual average tolerance of the set of modules (%), default value is 3 ModDisp module parameter dispersion losses (%), default value is 2 OhmDC Joule losses due to the DC wiring (%), default value is 1.5 OhmAC Joule losses due to the AC wiring (%), default value is 1.5 MPP average error of the MPP algorithm of the inverter (%), default value is 1 TrafoMT losses due to the MT transformer (%), default value is 1 Disp losses due to stops of the system (%), default value is 0.5
modeShd, struct, distances	See calcShd for details.
...	Additional arguments for calcG0 or calcGef

Details

The calculation of the irradiance on the horizontal plane is carried out with the function [calcG0](#). The transformation to the inclined surface makes use of the [fTheta](#) and [fInclin](#) functions inside the [calcGef](#) function. The shadows are computed with [calcShd](#) while the performance of the PV system is simulated with [fProd](#).

Value

A ProdGCPV object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fProd](#), [calcGef](#), [calcShd](#), [calcG0](#), [compare](#), [compareLosses](#), [mergesolaR](#)

Examples

```

library("data.table")

lat <- 37.2;

G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562,
         2814, 2179)

Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
       17.2, 15.2)

prom <- list(G0dm = G0dm, Ta = Ta)

###Comparison of different tracker methods
prodFixed <- prodGCPV(lat = lat, dataRad = prom,
                    keep.night = FALSE)

prod2x <- prodGCPV(lat = lat, dataRad = prom,
                  modeTrk = 'two',
                  keep.night = FALSE)

prodHoriz <- prodGCPV(lat = lat, dataRad = prom,
                    modeTrk = 'horiz',
                    keep.night = FALSE)

##Comparison of yearly productivities
compare(prodFixed, prod2x, prodHoriz)
compareLosses(prodFixed, prod2x, prodHoriz)

##Comparison of power time series
ComparePac <- data.table(Dates = indexI(prod2x),
                       two = as.data.tableI(prod2x)$Pac,
                       horiz = as.data.tableI(prodHoriz)$Pac,
                       fixed = as.data.tableI(prodFixed)$Pac)

AngSol <- as.data.tableI(as(prodFixed, 'Sol'))

ComparePac <- merge(AngSol, ComparePac, by = 'Dates')

ComparePac[, Month := as.factor(month(Dates))]

xyplot(two + horiz + fixed ~ AzS|Month, data = ComparePac,
       type = 'l',
       auto.key = list(space = 'right',
                      lines = TRUE,
                      points = FALSE),
       ylab = 'Pac')

###Shadows

```

```

#Two-axis trackers
struct2x <- list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 8)
dist2x <- data.table(Lew = 40, Lns = 30, H = 0)
prod2xShd <- prodGCPV(lat = lat, dataRad = prom,
                     modeTrk = 'two',
                     modeShd = 'area',
                     struct = struct2x,
                     distances = dist2x)

print(prod2xShd)

#Horizontal N-S tracker
structHoriz <- list(L = 4.83);
distHoriz <- data.table(Lew = structHoriz$L*4);

#Without Backtracking
prodHorizShd <- prodGCPV(lat = lat, dataRad = prom,
                        sample = '10 min',
                        modeTrk = 'horiz',
                        modeShd = 'area', betaLim = 60,
                        distances = distHoriz,
                        struct = structHoriz)

print(prodHorizShd)

xyplot(r2d(Beta)~r2d(w),
       data = prodHorizShd,
       type = 'l',
       main = 'Inclination angle of a horizontal axis tracker',
       xlab = expression(omega (degrees)),
       ylab = expression(beta (degrees)))

#With Backtracking
prodHorizBT <- prodGCPV(lat = lat, dataRad = prom,
                       sample = '10 min',
                       modeTrk = 'horiz',
                       modeShd = 'bt', betaLim = 60,
                       distances = distHoriz,
                       struct = structHoriz)

print(prodHorizBT)

xyplot(r2d(Beta)~r2d(w),
       data = prodHorizBT,
       type = 'l',
       main = 'Inclination angle of a horizontal axis tracker\n with backtracking',
       xlab = expression(omega (degrees)),
       ylab = expression(beta (degrees)))

compare(prodFixed, prod2x, prodHoriz, prod2xShd,
        prodHorizShd, prodHorizBT)

compareLosses(prodFixed, prod2x, prodHoriz, prod2xShd,
              prodHorizShd, prodHorizBT)

```



```

compareYf2 <- mergesolaR(prodFixed, prod2x, prodHoriz, prod2xShd,
                        prodHorizShd, prodHorizBT)

xyplot(prodFixed + prod2x +prodHoriz + prod2xShd + prodHorizShd + prodHorizBT ~ Dates,
       data = compareYf2, type = 'l', ylab = 'kWh/kWp',
       main = 'Daily productivity',
       auto.key = list(space = 'right'))

```

A5_prodPVPS

Performance of a PV pumping system

Description

Compute every step from solar angles to effective irradiance to calculate the performance of a PV pumping system.

Usage

```

prodPVPS(lat,
         modeTrk = 'fixed',
         modeRad = 'prom',
         dataRad,
         sample = 'hour',
         keep.night = TRUE,
         sunGeometry = 'michalsky',
         corr, f,
         betaLim = 90, beta = abs(lat)-10, alpha = 0,
         iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE,
         pump , H,
         Pg, converter= list(),
         effSys = list(),
         ...)

```

Arguments

lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
modeTrk	A character string, describing the tracking method of the generator. See calcGef for details.
modeRad, dataRad	Information about the source data of the global irradiation. See calcG0 for details.
sample, keep.night	See calcSol for details.
sunGeometry	character, method for the sun geometry calculations. See calcSol , fSold and fSolI .

corr, f	See calcG0 for details.
betaLim, beta, alpha, iS, alb, horizBright, HCPV	See calcGef for details.
pump	A list extracted from pumpCoef
H	Total manometric head (m)
Pg	Nominal power of the PV generator (Wp)
converter	list containing the nominal power of the frequency converter, Pnom, and Ki, vector of three values, coefficients of the efficiency curve.
effSys	list of numeric values with information about the system losses, ModQual average tolerance of the set of modules (%), default value is 3 ModDisp module parameter disperssion losses (%), default value is 2 OhmDC Joule losses due to the DC wiring (%), default value is 1.5 OhmAC Joule losses due to the AC wiring (%), default value is 1.5
...	Additional arguments for calcSol , calcG0 and calcGef .

Details

The calculation of the irradiance on the generator is carried out with the function [calcGef](#). The performance of the PV system is simulated with [fPump](#).

Value

A [ProdPVPS](#) object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Abella, M. A., Lorenzo, E. y Chenlo, F.: PV water pumping systems based on standard frequency converters. *Progress in Photovoltaics: Research and Applications*, 11(3):179–191, 2003, ISSN 1099-159X.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[NmgPVPS](#), [fPump](#), [pumpCoef](#)

Description

Compute the irradiance and irradiation including shadows for two-axis and horizontal N-S axis trackers and fixed surfaces. It makes use of the function [fSombra](#) for the shadows factor calculation. It is used by the function [calcGef](#).

Usage

```
calcShd(radEf,
        modeShd = '',
        struct = list(),
        distances = data.table())
```

Arguments

radEf	A Gef object. It may be the result of the calcGef function.
modeShd	character, defines the type of shadow calculation. In this version of the package the effect of the shadow is calculated as a proportional reduction of the circumsolar diffuse and direct irradiances. This type of approach is selected with modeShd = 'area'. In future versions other approaches which relate the geometric shadow and the electrical connections of the PV generator will be available. If radEf@modeTrk = 'horiz' it is possible to calculate the effect of backtracking with modeShd = 'bt'. If modeShd = c('area', 'bt') the backtracking method will be carried out and therefore no shadows will appear. Finally, for two-axis trackers it is possible to select modeShd = 'prom' in order to calculate the effect of shadows on an average tracker (see fSombra6). The result will include three variables (Gef0, Def0 and Bef0) with the irradiance/irradiation without shadows as a reference.
struct	list. When radEf@modeTrk = 'fixed' or modeTrk = 'horiz' only a component named L, which is the height (meters) of the tracker, is needed. For two-axis trackers (radEf@modeTrk = 'two'), an additional component named W, the width of the tracker, is required. Moreover, only when radEf@modeTrk = 'two' two components named Nrow and Ncol are included under this list. These components define, respectively, the number of rows and columns of the whole set of two-axis trackers in the PV plant.
distances	data.frame. When radEf@modeTrk = 'fixed' it includes a component named D for the distance between fixed surfaces. An additional component named H can be included with the relative height between surfaces. When radEf@modeTrk = 'horiz' it only includes a component named Lew, being the distance between horizontal NS trackers along the East-West direction.

When `radEf@modeTrk = 'two'` it includes a component named `Lns` being the distance between trackers along the North-South direction, a component named `Lew`, being the distance between trackers along the East-West direction and a (optional) component named `H` with the relative height between surfaces.

The distances, in meters, are defined between axis of the trackers.

Value

A `Gef` object including three additional variables (`Gef0`, `Def0` and `Bef0`) in the slots `GefI`, `GefD`, `Gefdm` and `Gefy` with the irradiance/irradiation without shadows as a reference.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[calcG0](#), [fTheta](#), [fInclin](#), [calcShd](#).

A7_optimShd

Shadows calculation for a set of distances between elements of a PV grid connected plant.

Description

The optimum distance between trackers or static structures of a PV grid connected plant depends on two main factors: the ground requirement ratio (defined as the ratio of the total ground area to the generator PV array area), and the productivity of the system including shadow losses. Therefore, the optimum separation may be the one which achieves the highest productivity with the lowest ground requirement ratio.

However, this definition is not complete since the terrain characteristics and the costs of wiring or civil works could alter the decision. This function is a help for choosing this distance: it computes the productivity for a set of combinations of distances between the elements of the plant.

Usage

```
optimShd(lat,
         modeTrk = 'fixed',
         modeRad = 'prom',
         dataRad,
         sample = 'hour',
```

```

keep.night = TRUE,
sunGeometry = 'michalsky',
betaLim = 90, beta = abs(lat)-10, alpha = 0,
iS = 2, alb = 0.2, HCPV = FALSE,
module = list(),
generator = list(),
inverter = list(),
effSys = list(),
modeShd = '',
struct = list(),
distances = data.table(),
res = 2,
prog = TRUE)

```

Arguments

lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
modeTrk	character, to be chosen from 'fixed', 'two' or 'horiz'. When modeTrk = 'fixed' the surface is fixed (inclination and azimuth angles are constant). The performance of a two-axis tracker is calculated with modeTrk = 'two', and modeTrk = 'horiz' is the option for an horizontal N-S tracker. Its default value is modeTrk = 'fixed'
modeRad, dataRad	Information about the source data of the global irradiation. See calcG0 for details. For this function the option modeRad = 'bdI' is not supported.
sample	character, containing one of "sec", "min", "hour". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s" (used by seq.POSIXt)
keep.night	logical When it is TRUE (default) the time series includes the night.
sunGeometry	character, method for the sun geometry calculations. See calcSol , fSold and fSolI .
betaLim, beta, alpha, iS, alb, HCPV	See calcGef for details.
module	list of numeric values with information about the PV module, Vocn open-circuit voltage of the module at Standard Test Conditions (default value 57.6 volts.) Iscn short circuit current of the module at Standard Test Conditions (default value 4.7 amperes.) Vmn maximum power point voltage of the module at Standard Test Conditions (default value 46.08 amperes.) Imn Maximum power current of the module at Standard Test Conditions (default value 4.35 amperes.) Ncs number of cells in series inside the module (default value 96) Ncp number of cells in parallel inside the module (default value 1)

	<p>CoefVT coefficient of decrement of voltage of each cell with the temperature (default value 0.0023 volts per celsius degree)</p> <p>TONC nominal operational cell temperature, celsius degree (default value 47).</p>
generator	<p>list of numeric values with information about the generator,</p> <p>Nms number of modules in series (default value 12)</p> <p>Nmp number of modules in parallel (default value 11)</p>
inverter	<p>list of numeric values with information about the DC/AC inverter,</p> <p>Ki vector of three values, coefficients of the efficiency curve of the inverter (default c(0.01, 0.025, 0.05)), or a matrix of nine values (3x3) if there is dependence with the voltage (see references).</p> <p>Pinv nominal inverter power (W) (default value 25000 watts.)</p> <p>Vmin, Vmax minimum and maximum voltages of the MPP range of the inverter (default values 420 and 750 volts)</p> <p>Gumb minimum irradiance for the inverter to start (W/m²) (default value 20 W/m²)</p>
effSys	<p>list of numeric values with information about the system losses,</p> <p>ModQual average tolerance of the set of modules (%), default value is 3</p> <p>ModDisp module parameter dispersion losses (%), default value is 2</p> <p>OhmDC Joule losses due to the DC wiring (%), default value is 1.5</p> <p>OhmAC Joule losses due to the AC wiring (%), default value is 1.5</p> <p>MPP average error of the MPP algorithm of the inverter (%), default value is 1</p> <p>TrafoMT losses due to the MT transformer (%), default value is 1</p> <p>Disp losses due to stops of the system (%), default value is 0.5</p>
modeShd	<p>character, defines the type of shadow calculation. In this version of the package the effect of the shadow is calculated as a proportional reduction of the circumsolar diffuse and direct irradiances. This type of approach is selected with modeShd = 'area'. In future versions other approaches which relate the geometric shadow and the electrical connections of the PV generator will be available. If modeTrk = 'horiz' it is possible to calculate the effect of backtracking with modeShd = 'bt'. If modeShd = c('area', 'bt') the backtracking method will be carried out and therefore no shadows will appear. Finally, for two-axis trackers it is possible to select modeShd = 'prom' in order to calculate the effect of shadows on an average tracker (see fSombra6). The result will include three variables (Gef0, Def0 and Bef0) with the irradiance/irradiation without shadows as a reference.</p>
struct	<p>list. When modeTrk = 'fixed' or modeTrk = 'horiz' only a component named L, which is the height (meters) of the tracker, is needed. For two-axis trackers (modeTrk = 'two'), an additional component named W, the width of the tracker, is required. Moreover, two components named Nrow and Ncol are included under this list. These components define, respectively, the number of rows and columns of the whole set of trackers in the PV plant.</p>
distances	<p>list, whose three components are vectors of length 2:</p> <p>Lew (only when modeTrk = 'horiz' or modeTrk = 'two'), minimum and maximum distance (meters) between horizontal NS and two-axis trackers along the East-West direction.</p>

	Lns (only when modeTrk = 'two'), minimum and maximum distance (meters) between two-axis trackers along the North-South direction.
	D (only when modeTrk = 'fixed'), minimum and maximum distance (meters) between fixed surfaces.
	These distances, in meters, are defined between the axis of the trackers.
res	numeric; optimShd constructs a sequence from the minimum to the maximum value of distances, with res as the increment, in meters, of the sequence.
prog	logical, show a progress bar; default value is TRUE

Details

optimShd calculates the energy produced for every combination of distances as defined by distances and res. The result of this function is a [Shade-class](#) object. A method of shadeplot for this class is defined ([shadeplot-methods](#)), and it shows the graphical relation between the productivity and the distance between trackers or fixed surfaces.

Value

A [Shade](#) object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O.: Grandes Centrales Fotovoltaicas: producción, seguimiento y ciclo de vida. PhD Thesis, UNED, 2008. https://www.researchgate.net/publication/39419806_Grandes_Centrales_Fotovoltaicas_Produccion_Seguimiento_y_Ciclo_de_Vida.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[prodGCPV](#), [calcShd](#)

Examples

```
library("data.table")

lat <- 37.2;
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562, 2814,
2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)

###Two-axis trackers
```

```

library("latticeExtra")

struct2x <- list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 3)
dist2x <- list(Lew = c(30, 45), Lns = c(20, 40))

ShdM2x <- optimShd(lat = lat, dataRad = prom, modeTrk = 'two',
                  modeShd = c('area', 'prom'),
                  distances = dist2x, struct = struct2x,
                  res = 5)

shadeplot(ShdM2x)

pLew <- xyplot(Yf ~ GRR, data = ShdM2x, groups = factor(Lew), type = c('l', 'g'),
              main = 'Productivity for each Lew value')
pLew + glayer(panel.text(x[1], y[1], group.value))

pLns <- xyplot(Yf ~ GRR, data = ShdM2x, groups = factor(Lns), type = c('l', 'g'),
              main = 'Productivity for each Lns value')
pLns + glayer(panel.text(x[1], y[1], group.value))

## 1-axis tracker with Backtracking
structHoriz <- list(L = 4.83);
distHoriz <- list(Lew = structHoriz$L * c(2,5));

Shd12HorizBT <- optimShd(lat = lat, dataRad = prom,
                       modeTrk = 'horiz',
                       betaLim = 60,
                       distances = distHoriz, res = 2,
                       struct = structHoriz,
                       modeShd = 'bt')

shadeplot(Shd12HorizBT)

xyplot(diff(Yf) ~ GRR[-1], data = Shd12HorizBT, type = c('l', 'g'))

### Fixed system
structFixed = list(L = 5);
distFixed <- list(D = structFixed$L * c(1,3));
Shd12Fixed <- optimShd(lat = lat, dataRad = prom,
                      modeTrk = 'fixed',
                      distances = distFixed, res = 2,
                      struct = structFixed,
                      modeShd = 'area')
shadeplot(Shd12Fixed)

```


Description

Functions for the class `Meteo` that transforms an intradaily `Meteo` object into a daily and a daily into a monthly.

Usage

```
Meteoi2Meteod(G0i)
```

```
Meteod2Meteom(G0d)
```

Arguments

`G0i` A `Meteo` object with intradaily data

`G0d` A `Meteo` object with daily data

Value

A `Meteo` object

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[readBDd](#), [readG0dm](#), [readSIAR](#)

Examples

```
library("data.table")

G0dm <- c(2.766,3.491,4.494,5.912,6.989,7.742,7.919,
         7.027,5.369,3.562,2.814,2.179) * 1000;
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2,
        28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)

g0 <- calcG0(lat = 37.2, dataRad = prom, modeRad = 'aguiar')
G0i <- as.data.tableI(g0)
G0i <- dt2Meteo(G0i, lat = 37.2)
G0i

G0d <- Meteoi2Meteod(G0i)
G0d

G0m <- Meteod2Meteom(G0d)
G0m
```

A8_readBD	<i>Daily or intradaily values of global horizontal irradiation and ambient temperature from a local file or a data.frame.</i>
-----------	---

Description

Constructor for the class `Meteo` with values of *daily* or *intradaily* values of global horizontal irradiation and ambient temperature from a local file or a `data.frame`.

Usage

```
readBDd(file, lat,
        format = '%d/%m/%Y',
        header = TRUE, fill = TRUE, dec = '.', sep = ';',
        dates.col = 'Dates', ta.col = 'Ta',
        g0.col = 'G0', keep.cols = FALSE, ...)

readBDi(file, lat,
        format = '%d/%m/%Y %H:%M:%S',
        header = TRUE, fill = TRUE, dec = '.',
        sep = ';', dates.col = 'Dates', times.col,
        ta.col = 'Ta', g0.col = 'G0', keep.cols = FALSE, ...)

dt2Meteo(file, lat, source = '', type)

zoo2Meteo(file, lat, source = '')
```

Arguments

file	<p>The name of the file (<code>readBDd</code> and <code>readBDi</code>), <code>data.frame</code> (or <code>data.table</code>) (<code>dt2Meteo</code>) or <code>zoo</code> (<code>zoo2Meteo</code>) which the data are to be read from. It should contain a column <code>G0d</code> with <i>daily</i> (<code>readBDd</code>) or <code>G0</code> with <i>intradaily</i> (<code>readBDi</code>) values of global horizontal irradiation (Wh/m^2). It should also include a column named <code>Ta</code> with values of ambient temperature. However, if the object is only a vector with irradiation values, it will be converted to a <code>data.table</code> with two columns named <code>G0</code> and <code>Ta</code> (filled with constant values)</p> <p>If the <code>Meteo</code> object is to be used with <code>calcG0</code> (or <code>fCompD</code>, <code>fCompI</code>) and the option <code>corr = 'none'</code>, the file/<code>data.frame</code> must include three columns named <code>G0</code>, <code>B0</code> and <code>D0</code> with values of global, direct and diffuse irradiation on the horizontal plane.</p> <p>Only for daily data: if the ambient temperature is not available, the file should include two columns named <code>TempMax</code> and <code>TempMin</code> with daily values of maximum and minimum ambient temperature, respectively (see fTemp for details).</p>
header, fill, dec, sep	See fread
format	character string with the format of the dates or time index. (Default for daily time bases: <code>%d/%m/%Y</code>). (Default for intradaily time bases: <code>%d/%m/%Y %H:%M:%S</code>)

lat	numeric, latitude (degrees) of the location.
dates.col	character string with the name of the column wich contains the dates of the time series.
times.col	character string with the name of the column wich contains the time index of the series in case is in a different column than the dates.
source	character string with information about the source of the values. (Default: the name of the file).
ta.col, g0.col	character, the name of the columns with the information of ambient temperature and radiation in the provided file
keep.cols	If keep.cols=FALSE(default value), the Meteo object does not include the columns that are not important for the rest of operations
...	Arguments for fread
type	character, type of the data in dt2Meteo. To choose between 'prom', 'bd' and 'bdI'. If it is not provided, the function dt2Meteo calculate the type.

Value

A Meteo object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[fread](#), [readG0dm](#).

Examples

```
library("data.table")

data(helios)
names(helios) = c('Dates', 'G0d', 'TempMax', 'TempMin')

bd = dt2Meteo(helios, lat = 41, source = 'helios-IES', type = 'bd')

getData(bd)

xyplot(bd)
```

A8_readG0dm

Monthly mean values of global horizontal irradiation.

Description

Constructor for the class Meteo with 12 values of monthly means of irradiation.

Usage

```
readG0dm(G0dm, Ta = 25, lat = 0,
  year= as.POSIXlt(Sys.Date())$year+1900,
  promDays = c(17,14,15,15,15,10,18,18,18,19,18,13),
  source = '')
```

Arguments

G0dm	numeric, 12 values of monthly means of daily global horizontal irradiation (Wh/m ²).
Ta	numeric, 12 values of monthly means of ambient temperature (degrees Celsius).
lat	numeric, latitude (degrees) of the location.
year	numeric (Default: current year).
promDays	numeric, set of the average days for each month.
source	character string with information about the source of the values.

Value

Meteo object

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[readBDd](#)

Examples

```
library("data.table")
```

```
G0dm <-
  c(2.766,3.491,4.494,5.912,6.989,7.742,7.919,7.027,5.369,3.562,2.814,2.179) * 1000;
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
BD <- readG0dm(G0dm = G0dm, Ta = Ta, lat = 37.2)
print(BD)
getData(BD)
xyplot(BD)
```

`A8_readSIAR`*Meteorological data from the SIAR network.*

Description

Download, interpolate and transform meteorological data from the SIAR network.

Usage

```
readSIAR(Lon = 0, Lat = 0,
         inicio = paste(year(Sys.Date())-1, '01-01', sep = '-'),
         final = paste(year(Sys.Date())-1, '12-31', sep = '-'),
         tipo = 'Mensuales', n_est = 3)
```

Arguments

Lon	numeric, longitude (degrees) of the location.
Lat	numeric, latitude (degrees) of the location.
inicio	character or Date, first day of the records.
final	character or Date, last day of the records.
tipo	character, type of the records. To choose between Mensuales, Semanales, Diarios, Horarios.
n_est	integer, select that number of stations closest to the given point and then perform an IDW (Inverse Distance Weighting) interpolation with these data.

Value

A Meteo object

Author(s)

Francisco Delgado López, Oscar Perpiñán Lamigueiro.

See Also

[readG0dm](#), [readBDd](#)

Examples

```
library("data.table")

library("httr2")
library("jsonlite")

SIAR = readSIAR(Lon = -3.603, Lat = 40.033,
               ## Aranjuez, Community of Madrid, Spain
               inicio = '2023-01-01',
```

```

        final = '2023-05-01',
        tipo = 'Mensuales', n_est = 3)
SIAR

```

B1_Meteo-class *Class "Meteo"*

Description

A class for meteorological data.

Objects from the Class

Objects can be created by the family of [readBDd](#) functions.

Slots

latm: Latitude (degrees) of the meteorological station or source of the data.

data: A `data.table` object with the time series of daily irradiation (G_0 , Wh/m²), the ambient temperature (T_a) or the maximum and minimum ambient temperature (`TempMax` and `TempMin`).

source: A character with a short description of the source of the data.

type: A character, `prom`, `bd` or `bdI` depending on the constructor.

Methods

getData signature(`object = "Meteo"`): extracts the data slot as a `data.table` object.

getG0 signature(`object = "Meteo"`): extracts the irradiation as vector.

getLat signature(`object = "Meteo"`): extracts the latitude value.

indexD signature(`object = "Meteo"`): extracts the index of the data slot.

xyplot signature(`x = "formula"`, `data = "Meteo"`): plot the content of the object according to the `formula` argument.

xyplot signature(`x = "Meteo"`, `data = "missing"`): plot the data slot using the `xyplot` method for zoo objects.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[readBDd](#), [readBDi](#), [zoo2Meteo](#), [dt2Meteo](#), [readG0dm](#),

B2_Sol-class

Class "Sol": Apparent movement of the Sun from the Earth

Description

A class which describe the apparent movement of the Sun from the Earth.

Objects from the Class

Objects can be created by `calcSol`.

Slots

`lat`: numeric, latitude (degrees) as defined in the call to `calcSol`.

`solD`: Object of class "data.table" created by `fSolD`.

`solI`: Object of class "data.table" created by `fSolI`.

`method`: character, method for the sun geometry calculations.

`sample`: difftime, increment of the intradaily sequence.

Methods

as.data.tableD signature(object = "Sol"): conversion to a data.table with daily values.

as.data.tableI signature(object = "Sol"): conversion to a data.table with intradaily values.

getLat signature(object = "Sol"): latitude (degrees) as defined in the call to `calcSol`.

indexD signature(object = "Sol"): index of the solD slot.

indexI signature(object = "Sol"): index of the solI object.

xyplot signature(x = "formula", data = "Sol"): displays the contents of a Sol object with the xyplot method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, doi:10.18637/jss.v050.i09

See Also

`G0`, `Gef`.

 B3_G0-class

 Class "G0": irradiation and irradiance on the horizontal plane.

Description

This class contains the global, diffuse and direct irradiation and irradiance on the horizontal plane, and ambient temperature.

Objects from the Class

Objects can be created by the function `calcG0`.

Slots

`G0D`: Object of class `data.table` created by `fCompD`. It includes daily values of:

Fd: numeric, the diffuse fraction

Ktd: numeric, the clearness index

G0d: numeric, the global irradiation on a horizontal surface (Wh/m²)

D0d: numeric, the diffuse irradiation on a horizontal surface (Wh/m²)

B0d: numeric, the direct irradiation on a horizontal surface (Wh/m²)

`G0I`: Object of class `data.table` created by `fCompI`. It includes values of:

kt: numeric, clearness index

G0: numeric, global irradiance on a horizontal surface, (W/m²)

D0: numeric, diffuse irradiance on a horizontal surface, (W/m²)

B0: numeric, direct irradiance on a horizontal surface, (W/m²)

`G0dm`: Object of class `data.table` with monthly mean values of daily irradiation.

`G0y`: Object of class `data.table` with yearly sums of irradiation.

`Ta`: Object of class `data.table` with intradaily ambient temperature values.

Besides, this class contains the slots from the `Sol` and `Meteo` classes.

Extends

Class "`Meteo`", directly. Class "`Sol`", directly.

Methods

as.data.tableD signature(object = "G0"): conversion to a `data.table` with daily values.

as.data.tableI signature(object = "G0"): conversion to a `data.table` with intradaily values.

as.data.tableM signature(object = "G0"): conversion to a `data.table` with monthly values.

as.data.tableY signature(object = "G0"): conversion to a `data.frame` with yearly values.

indexD signature(object = "G0"): index of the `solD` slot.

indexI signature(object = "G0"): index of the `solI` slot.

getLat signature(object = "G0"): latitude of the inherited [Sol](#) object.

xyplot signature(x = "G0", data = "missing"): display the time series of daily values of irradiation.

xyplot signature(x = "formula", data = "G0"): displays the contents of a G0 object with the xyplot method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[Sol](#), [Gef](#).

B4_Gef-class

Class "Gef": irradiation and irradiance on the generator plane.

Description

This class contains the global, diffuse and direct irradiation and irradiance on the horizontal plane, and ambient temperature.

Objects from the Class

Objects can be created by the function [calcGef](#).

Slots

GefI: Object of class `data.table` created by [fInclin](#). It contains these components:

Bo: Extra-atmospheric irradiance on the inclined surface (W/m²)

Bn: Direct normal irradiance (W/m²)

G, B, D, Di, Dc, R: Global, direct, diffuse (total, isotropic and anisotropic) and albedo irradiance incident on an inclined surface (W/m²)

Gef, Bef, Def, Dief, Dcef, Ref: Effective global, direct, diffuse (total, isotropic and anisotropic) and albedo irradiance incident on an inclined surface (W/m²)

FTb, FTd, FTTr: Factor of angular losses for the direct, diffuse and albedo components

GefD: Object of class `data.table` with daily values of global, diffuse and direct irradiation.

Gefdm: Object of class `data.table` with monthly means of daily global, diffuse and direct irradiation.

Gefy: Object of class `data.table` with yearly sums of global, diffuse and direct irradiation.

Theta: Object of class `data.table` created by `fTheta`. It contains these components:

Beta: numeric, inclination angle of the surface (radians). When `modeTrk='fixed'` it is the value of the argument `beta` converted from degrees to radians.

Alpha: numeric, azimuth angle of the surface (radians). When `modeTrk='fixed'` it is the value of the argument `alpha` converted from degrees to radians.

cosTheta: numeric, cosine of the incidence angle of the solar irradiance on the surface

iS: numeric, degree of dirtiness.

alb: numeric, albedo reflection coefficient.

modeTrk: character, mode of tracking.

modeShd: character, mode of shadows.

angGen: A list with the values of `alpha`, `beta` and `betaLim`.

struct: A list with the dimensions of the structure.

distances: A `data.frame` with the distances between structures.

Extends

Class "`G0`", directly. Class "`Meteo`", by class "`G0`", distance 2. Class "`Sol`", by class "`G0`", distance 2.

Methods

as.data.tableD signature(object = "Gef"): conversion to a `data.table` with daily values.

as.data.tableI signature(object = "Gef"): conversion to a `data.table` with intradaily values.

as.data.tableM signature(object = "Gef"): conversion to a `data.table` with monthly values.

as.data.tableY signature(object = "Gef"): conversion to a `data.table` with yearly values.

indexD signature(object = "Gef"): index of the `solD` slot.

indexI signature(object = "Gef"): index of the `solI` slot.

getLat signature(object = "Gef"): latitude of the inherited `Sol` object.

xyplot signature(x = "Gef", data = "missing"): display the time series of daily values of irradiation.

xyplot signature(x = "formula", data = "Gef"): displays the contents of a `Gef` object with the `xyplot` method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, doi:10.18637/jss.v050.i09

See Also

[Sol](#), [G0](#).

B5_ProdGCPV-class	<i>Class "ProdGCPV": performance of a grid connected PV system.</i>
-------------------	---

Description

A class containing values of the performance of a grid connected PV system.

Objects from the Class

Objects can be created by [prodGCPV](#).

Slots

`prodI`: Object of class `data.table` created by [fProd](#). It includes these components:

Tc: cell temperature, °C.

Voc, Isc, Vmpp, Impp: open circuit voltage, short circuit current, MPP voltage and current, respectively.

Vdc, Idc: voltage and current at the input of the inverter.

Pdc: power at the input of the inverter, W

Pac: power at the output of the inverter, W

EffI: efficiency of the inverter

`prodD`: A `data.table` object with daily values of AC (`Eac`) and DC (`Edc`) energy (Wh), and productivity (`Yf`, Wh/Wp) of the system.

`prodDm`: A `data.table` object with monthly means of daily values of AC and DC energy (kWh), and productivity of the system.

`prody`: A `data.table` object with yearly sums of AC and DC energy (kWh), and productivity of the system.

`module`: A list with the characteristics of the module.

`generator`: A list with the characteristics of the PV generator.

`inverter`: A list with the characteristics of the inverter.

`effSys`: A list with the efficiency values of the system.

Besides, this class contains the slots from the ["Meteo"](#), ["Sol"](#), ["G0"](#) and ["Gef"](#) classes.

Extends

Class ["Gef"](#), directly. Class ["G0"](#), by class ["Gef"](#), distance 2. Class ["Meteo"](#), by class ["Gef"](#), distance 3. Class ["Sol"](#), by class ["Gef"](#), distance 3.

Methods

as.data.tableD signature(object = "ProdGCPV"): conversion to a data.table with daily values.

as.data.tableI signature(object = "ProdGCPV"): conversion to a data.table with intradaily values.

as.data.tableM signature(object = "ProdGCPV"): conversion to a data.table with monthly values.

as.data.tableY signature(object = "ProdGCPV"): conversion to a data.table with yearly values.

indexD signature(object = "ProdGCPV"): index of the solD slot.

indexI signature(object = "ProdGCPV"): index of the solI object.

getLat signature(object = "ProdGCPV"): latitude of the inherited [Sol](#) object.

xyplot signature(x = "ProdGCPV", data = "missing"): display the time series of daily values.

xyplot signature(x = "formula", data = "ProdGCPV"): displays the contents of a ProdGCPV object with the xyplot method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[Sol](#), [G0](#), [Gef](#), [Shade](#).

B6_ProdPVPS-class

Class "ProdPVPS": performance of a PV pumping system.

Description

Performance of a PV pumping system with a centrifugal pump and a variable frequency converter.

Objects from the Class

Objects can be created by [prodPVPS](#).

Slots

prodI: Object of class `data.table` with these components:

Q: Flow rate, (m³/h)

Pb, Ph: Pump shaft power and hydraulic power (W), respectively.

etam, etab: Motor and pump efficiency, respectively.

f: Frequency (Hz)

prodD: A `data.table` object with daily values of AC energy (Wh), flow (m³) and productivity of the system.

prodDm: A `data.table` object with monthly means of daily values of AC energy (kWh), flow (m³) and productivity of the system.

prody: A `data.table` object with yearly sums of AC energy (kWh), flow (m³) and productivity of the system.

pump A list extracted from `pumpCoef`

H Total manometric head (m)

Pg Nominal power of the PV generator (Wp)

converter list containing the nominal power of the frequency converter, `Pnom`, and `Ki`, vector of three values, coefficients of the efficiency curve.

effSys list of numeric values with information about the system losses

Besides, this class contains the slots from the `Gef` class.

Extends

Class "`Gef`", directly. Class "`G0`", by class "`Gef`", distance 2. Class "`Meteo`", by class "`Gef`", distance 3. Class "`Sol`", by class "`Gef`", distance 3.

Methods

as.data.tableD signature(object = "ProdPVPS"): conversion to a `data.table` with daily values.

as.data.tableI signature(object = "ProdPVPS"): conversion to a `data.table` with intradaily values.

as.data.tableM signature(object = "ProdPVPS"): conversion to a `data.table` with monthly values.

as.data.tableY signature(object = "ProdPVPS"): conversion to a `data.table` with yearly values.

indexD signature(object = "ProdPVPS"): index of the `solD` slot.

indexI signature(object = "ProdPVPS"): index of the `solI` object.

getLat signature(object = "ProdPVPS"): latitude of the inherited `Sol` object.

xyplot signature(x = "ProdPVPS", data = "missing"): display the time series of daily values.

xyplot signature(x = "formula", data = "ProdPVPS"): displays the contents of a `ProdPVPS` object with the `xyplot` method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Abella, M. A., Lorenzo, E. y Chenlo, F.: PV water pumping systems based on standard frequency converters. *Progress in Photovoltaics: Research and Applications*, 11(3):179–191, 2003, ISSN 1099-159X.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[prodPVPS](#), [fPump](#).

B7_Shade-class

Class "Shade": shadows in a PV system.

Description

A class for the optimization of shadows in a PV system.

Objects from the Class

Objects can be created by [optimShd](#).

Slots

FS: numeric, shadows factor values for each combination of distances.

GRR: numeric, Ground Requirement Ratio for each combination.

Yf: numeric, final productivity for each combination.

FS.loess: A local fitting of FS with loess.

Yf.loess: A local fitting of Yf with loess.

modeShd: character, mode of shadows.

struct: A list with the dimensions of the structure.

distances: A data.frame with the distances between structures.

res numeric, difference (meters) between the different steps of the calculation.

Besides, as a reference, this class includes a [ProdGCPV](#) object with the performance of a PV systems without shadows.

Extends

Class "[ProdGCPV](#)", directly. Class "[Gef](#)", by class "[ProdGCPV](#)", distance 2. Class "[G0](#)", by class "[ProdGCPV](#)", distance 3. Class "[Meteo](#)", by class "[ProdGCPV](#)", distance 4. Class "[Sol](#)", by class "[ProdGCPV](#)", distance 4.

Methods

as.data.frame signature(x = "Shade"): conversion to a data.frame including columns for distances (Lew, Lns, and D) and results (FS, GRR and Yf).

shadeplot signature(x = "Shade"): display the results of the iteration with a level plot for the two-axis tracking, or with conventional plot for horizontal tracking and fixed systems.

xyplot signature(x = "formula", data = "Shade"): display the content of the Shade object with the xyplot method for formulas.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O.: Grandes Centrales Fotovoltaicas: producción, seguimiento y ciclo de vida. PhD Thesis, UNED, 2008. https://www.researchgate.net/publication/39419806_Grandes_Centrales_Fotovoltaicas_Produccion_Seguimiento_y_Ciclo_de_Vida.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[Gef](#), [ProdGCPV](#).

C_corrFdKt

Correlations between the fraction of diffuse irradiation and the clearness index.

Description

A set of correlations between the fraction of diffuse irradiation and the clearness index used by [fCompD](#) and [fCompI](#).

Usage

```
## Monthly means of daily values
Ktm(sol, G0dm)
FdKtPage(sol, G0dm)
FdKtLJ(sol, G0dm)

## Daily values
Ktd(sol, G0d)
FdKtCPR(sol, G0d)
FdKtEKDd(sol, G0d)
FdKtCLIMEDd(sol, G0d)
```

```
## Intradaily values
Kti(sol, G0i)
FdKtEKDh(sol, G0i)
FdKtCLIMEDh(sol, G0i)
FdKtBRL(sol, G0i)
```

Arguments

sol	A Sol object, it may be the result of the calcSol function.
G0dm	A Meteo object with monthly means of radiation. It may be the result of the readG0dm function.
G0d	A Meteo object with daily values of radiation. It may be the result of the readBDd (or equivalent) function.
G0i	A Meteo object with intraindaily values of radiation. It may be the result of the readBDi (or equivalent) function.

Value

A `data.table`, with two columns:

Fd	A numeric, the diffuse fraction.
Kt	A numeric, the clearness index(provided by the Kt functions).

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López; The BRL model was suggested by Kevin Ummel.

References

- Page, J. K., The calculation of monthly mean solar radiation for horizontal and inclined surfaces from sunshine records for latitudes 40N-40S. En U.N. Conference on New Sources of Energy, vol. 4, págs. 378–390, 1961.
- Collares-Pereira, M. y Rabl, A., The average distribution of solar radiation: correlations between diffuse and hemispherical and between daily and hourly insolation values. *Solar Energy*, 22:155–164, 1979.
- Erbs, D.G, Klein, S.A. and Duffie, J.A., Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation. *Solar Energy*, 28:293:302, 1982.
- De Miguel, A. et al., Diffuse solar irradiation model evaluation in the north mediterranean belt area, *Solar Energy*, 70:143-153, 2001.
- Ridley, B., Boland, J. and Lauret, P., Modelling of diffuse solar fraction with multiple predictors, *Renewable Energy*, 35:478-482, 2010.

See Also

[fCompD](#), [fCompI](#)

Examples

```

library("data.table")

lat <- 37.2
BTd <- fBTd(mode = 'prom')
G0dm <- c(2.766, 3.491, 4.494, 5.912, 6.989, 7.742, 7.919, 7.027, 5.369,
         3.562, 2.814, 2.179)*1000;
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2,
        15.2)

prom <- readG0dm(G0dm = G0dm, Ta = Ta, lat = lat)
sol <- calcSol(lat = lat, BTd = BTd)

Kt <- Ktm(sol = sol, G0dm = prom)
Kt

Page <- FdKtPage(sol = sol, G0dm = prom)
LJ <- FdKtLJ(sol = sol, G0dm = prom)
Monthly <- merge(Page, LJ, by = 'Kt',
                 suffixes = c('.Page', '.LJ'))
Monthly

xyplot(Fd.Page+Fd.LJ~Kt, data = Monthly,
       type = c('l', 'g'), auto.key = list(space = 'right'))

Kt = Ktd(sol = sol, G0d = prom)
Kt

CPR <- FdKtCPR(sol = sol, G0d = prom)
CLIMEDd <- FdKtCLIMEDd(sol = sol, G0d = prom)
Daily <- merge(CPR, CLIMEDd, by = 'Kt',
               suffixes = c('.CPR', '.CLIMEDd'))
Daily

xyplot(Fd.CPR + Fd.CLIMEDd ~ Kt, data = Daily,
       type = c('l', 'g'), auto.key = list(space = 'right'))

```

C_fBTd

Daily time base

Description

Construction of a daily time base for solar irradiation calculation

Usage

```
fBTd(mode = "prom",
```

```

year = as.POSIXlt(Sys.Date())$year+1900,
start = paste('01-01-',year,sep = ''),
end = paste('31-12-',year,sep = ''),
      format = '%d-%m-%Y')

```

Arguments

mode	character, controls the type of time base to be created. With mode = 'serie' the result is a daily time series from start to end. With mode = 'prom' only twelve days, one for each month, are included. During these 'average days' the declination angle is equal to the monthly mean of this angle.
year	which year is to be used for the time base when mode = 'prom'. Its default value is the current year.
start	first day of the time base for mode = 'serie'. Its default value is the first of January of the current year.
end	last day of the time base for mode = 'serie'. Its default value is the last day of December of the current year.
format	format of start and end.

Details

This function is commonly used inside fSoLD.

Value

This function returns a POSIXct object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fSoLD](#), [as.POSIXct](#), [seq.POSIXt](#).

Examples

```

library("data.table")

#Average days
fBTd(mode = 'prom')

```

```
#The day #100 of the year 2008
BTd = fBTd(mode = 'serie', year = 2008)
BTd[100]
```

C_fBTi	<i>Intra-daily time base</i>
--------	------------------------------

Description

Construction of an intra-daily time base for solar irradiation calculation

Usage

```
fBTi(BTd, sample = 'hour')
```

Arguments

BTd	vector, it may be a result for fBTd or indexD
sample	character, identify the sample of the time set. Its default value is 'hour'.

Details

This function is commonly used inside fSoII.

Value

This function returns a POSIXct object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```
library("data.table")

#Average days
BTd <- fBTd(mode = 'prom')

#Intradaily base time for the first day
BTi <- fBTi(BTd = BTd[1], sample = 'hour')
BTi
```

C_fCompD

*Components of daily global solar irradiation on a horizontal surface***Description**

Extract the diffuse and direct components from the daily global irradiation on a horizontal surface by means of regressions between the clearness index and the diffuse fraction parameters.

Usage

```
fCompD(sol, G0d, corr = "CPR", f)
```

Arguments

sol	A Sol object from calcSol or a data.table object from fSold . Both of them include a component named B0d, which stands for the extra-atmospheric daily irradiation incident on a horizontal surface
G0d	A Meteo object from readG0dm , readBDd , or a data.table object containing daily global irradiation (Wh/m ²) on a horizontal surface. See below for corr = 'none'.
corr	A character, the correlation between the fraction of diffuse irradiation and the clearness index to be used. With this version several options are available, as described in corrFdKt . For example, the FdKtPage is selected with corr = 'Page' and the FdKtCPR with corr = 'CPR'. If corr = 'user' the use of a correlation defined by a function f is possible. If corr = 'none' the G0d object should include information about global, diffuse and direct daily irradiation with columns named G0d, D0d and B0d, respectively.
f	A function defining a correlation between the fraction of diffuse irradiation and the clearness index. It is only necessary when corr = 'user'

Value

A data.table object which includes:

Fd	numeric, the diffuse fraction
Ktd	numeric, the clearness index
G0d	numeric, the global irradiation on a horizontal surface (Wh/m ²)
D0d	numeric, the diffuse irradiation on a horizontal surface (Wh/m ²)
B0d	numeric, the direct irradiation on a horizontal surface (Wh/m ²)

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, doi:10.18637/jss.v050.i09

See Also

[fCompI](#)

Examples

```
library("data.table")

lat <- 37.2;
BTd <- fBTd(mode = 'serie')

SolD <- fSolD(lat, BTd[100])

G0d <- 5000
fCompD(SolD, G0d, corr = "Page")
fCompD(SolD, G0d, corr = "CPR")

#define a function fKtd with the correlation of CPR
fKtd <- function(sol, G0d){
  Kt <- Ktm(sol, G0d)
  Fd <- (0.99*(Kt <= 0.17))+ (Kt>0.17)*(1.188 -2.272 * Kt + 9.473 * Kt^2 -
  21.856 * Kt^3 + 14.648 * Kt^4)
  return(data.table(Fd, Kt))}
#The same as with corr = "CPR"
fCompD(SolD, G0d, corr = "user", f = fKtd)

lat <- -37.2;
SolDs <- fSolD(lat, BTd[283])
G0d <- data.table(Dates = SolDs$Dates, G0d = 5000)
fCompD(SolDs, G0d, corr = "CPR")

lat <- 37.2;
G0dm <- c(2.766,3.491,4.494,5.912,6.989,7.742,7.919,7.027,5.369,3.562,2.814,2.179)*1000;
Rad <- readG0dm(G0dm, lat = lat)
solD <- fSolD(lat, fBTd(mode = 'prom'))
fCompD(solD, Rad, corr = 'Page')
```

C_fCompI

Calculation of solar irradiance on a horizontal surface

Description

From the daily global, diffuse and direct irradiation values supplied by fCompD, the profile of the global, diffuse and direct irradiance is calculated with the rd and rg components of fSolI.

Usage

```
fCompI(sol, compD, G0I, corr = 'none', f, filterG0 = TRUE)
```

Arguments

sol	A Sol object as provided by <code>calcSol</code> or a <code>data.table</code> object as provided by <code>fSolI</code> .
compD	A <code>data.table</code> object as provided by <code>fCompD</code> . It is not considered if <code>G0I</code> is provided.
G0I	A Meteo object from <code>readBDi</code> , <code>dt2Meteo</code> or <code>zoo2Meteo</code> , or a <code>data.table</code> object containing <i>intradaily</i> global irradiance (W/m ²) on a horizontal surface. See below for <code>corr = 'none'</code> .
corr	A character, the correlation between the the fraction of intradaily diffuse irradiation and the clearness index to be used. It is ignored if <code>G0I</code> is not provided. With this version several correlations are available, as described in <code>corrFdkT</code> . You should choose one of <i>intradaily</i> proposals. For example, the <code>FdkTCLIMEDh</code> is selected with <code>corr = 'CLIMEDh'</code> . If <code>corr = 'user'</code> the use of a correlation defined by a function <code>f</code> is possible. If <code>corr = 'none'</code> the <code>G0I</code> object must include information about global, diffuse and direct intradaily irradiation with columns named <code>G0</code> , <code>D0</code> and <code>B0</code> , respectively.
f	A function defining a correlation between the fraction of diffuse irradiation and the clearness index. It is only necessary when <code>corr = 'user'</code>
filterG0	A logical. If TRUE (default) this function sets the global irradiation values to NA when they are higher than the extra-atmospheric irradiation values.

Value

A `data.table` with these components:

kt	numeric, clearness index.
fd	numeric, diffuse fraction.
G0	numeric, global irradiance on a horizontal surface, (W/m ²)
D0	numeric, diffuse irradiance on a horizontal surface, (W/m ²)
B0	numeric, direct irradiance on a horizontal surface, (W/m ²)

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Collares-Pereira, M. y Rabl, A., The average distribution of solar radiation: correlations between diffuse and hemispherical and between daily and hourly insolation values. *Solar Energy*, 22:155–164, 1979.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, doi:10.18637/jss.v050.i09

See Also

[fCompD](#), [fSolI](#), [calcSol](#), [corrFdKt](#).

Examples

```
library("data.table")

lat <- 37.2

BTd <- fBTd(mode = 'serie')
solD <- fSolD(lat, BTd[100])
solI <- fSolI(solD, sample = 'hour')
G0d <- data.table(Dates = solD$Dates, G0d = 5000)
compD <- fCompD(solD, G0d, corr = "Page")
fCompI(solI, compD)

sol <- calcSol(lat, fBTd(mode = 'prom'), sample = 'hour', keep.night = FALSE)

G0dm <- c(2.766, 3.491, 4.494, 5.912, 6.989, 7.742,
          7.919, 7.027, 5.369, 3.562, 2.814, 2.179)*1000

Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9,
        24.3, 18.2, 17.2, 15.2)

BD <- readG0dm(G0dm = G0dm, Ta = Ta, lat = lat)
compD <- fCompD(sol, BD, corr = 'Page')
compI <- fCompI(sol, compD)
head(compI)

## Use of 'corr'. The help page of calcG0 includes additional examples
## with intradaily data xyplot(fd ~ kt, data = compI)

climed <- fCompI(sol, G0I = compI, corr = 'CLIMEDh')
xyplot(Fd ~ Kt, data = climed)

ekdh <- fCompI(sol, G0I = compI, corr = 'EKDh')
xyplot(Fd ~ Kt, data = ekdh)

brl <- fCompI(sol, G0I = compI, corr = 'BRL')
xyplot(Fd ~ Kt, data = brl)
```

Description

The solar irradiance incident on an inclined surface is calculated from the direct and diffuse irradiance on a horizontal surface, and from the evolution of the angles of the Sun and the surface.

Moreover, the effect of the angle of incidence and dust on the PV module is included to obtain the effective irradiance.

This function is used by the `calcGef` function.

Usage

```
fInclin(compI, angGen, iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE)
```

Arguments

<code>compI</code>	A G_0 object. It may be the result of <code>calcG0</code> .
<code>angGen</code>	A <code>data.table</code> object, including at least three variables named <code>Beta</code> , <code>Alpha</code> and <code>cosTheta</code> . It may be the result of <code>fTheta</code> .
<code>iS</code>	integer, degree of dirtiness. Its value must be included in the set (1,2,3,4). <code>iS = 1</code> corresponds to a clean surface while <code>iS = 4</code> is the choice for a dirty surface. Its default value is 2
<code>alb</code>	numeric, albedo reflection coefficient. Its default value is 0.2
<code>horizBright</code>	logical, if <code>TRUE</code> , the horizon brightness correction proposed by Reind et al. is used.
<code>HCPV</code>	logical, if <code>TRUE</code> the diffuse and albedo components of the <i>effective</i> irradiance are set to zero. HCPV is the acronym of High Concentration PV system.

Details

The solar irradiance incident on an inclined surface can be calculated from the direct and diffuse irradiance on a horizontal surface, and from the evolution of the angles of the Sun and the surface. The transformation of the direct radiation is straightforward since only geometric considerations are needed. However, the treatment of the diffuse irradiance is more complex since it involves the modelling of the atmosphere. There are several models for the estimation of diffuse irradiance on an inclined surface. The one which combines simplicity and acceptable results is the proposal of Hay and McKay. This model divides the diffuse component in isotropic and anisotropic whose values depends on a anisotropy index. On the other hand, the effective irradiance, the fraction of the incident irradiance that reaches the cells inside a PV module, is calculated with the losses due to the angle of incidence and dirtiness. This behaviour can be simulated with a model proposed by Martin and Ruiz requiring information about the angles of the surface and the level of dirtiness (`iS`)

Value

A `data.table` object with these components:

<code>Bo</code>	Extra-atmospheric irradiance on the inclined surface (W/m^2)
<code>Bn</code>	Direct normal irradiance (W/m^2)
<code>G, B, D, Di, Dc, R</code>	Global, direct, diffuse (total, isotropic and anisotropic) and albedo irradiance incident on an inclined surface (W/m^2)
<code>Gef, Bef, Def, Dief, Dcef, Ref</code>	Effective global, direct, diffuse (total, isotropic and anisotropic) and albedo irradiance incident on an inclined surface (W/m^2)

FTb, FTd, FTr Factor of angular losses for the direct, diffuse and albedo components

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Hay, J. E. and McKay, D. C.: Estimating Solar Irradiance on Inclined Surfaces: A Review and Assessment of Methodologies. *Int. J. Solar Energy*, (3):pp. 203, 1985.
- Martin, N. and Ruiz, J.M.: Calculation of the PV modules angular losses under field conditions by means of an analytical model. *Solar Energy Materials & Solar Cells*, 70:25–38, 2001.
- D. T. Reindl and W. A. Beckman and J. A. Duffie: Evaluation of hourly tilted surface radiation models, *Solar Energy*, 45:9-17, 1990.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fTheta](#), [fCompI](#), [calcGef](#).

C_fProd

Performance of a PV system

Description

Simulate the behaviour of a grid connected PV system under different conditions of irradiance and temperature. This function is used by the [prodGCPV](#) function.

Usage

```
fProd(inclin, module, generator, inverter, effSys)
```

Arguments

inclin	A Gef object, a <code>data.table</code> object. In case of being <code>data.table</code> it must include a component named <code>Gef</code> (effective irradiance, W/m^2) and another named <code>Ta</code> (ambient temperature, $^{\circ}C$).
module	list of numeric values with information about the PV module, <code>Vocn</code> open-circuit voltage of the module at Standard Test Conditions (default value 51.91 volts.) <code>Iscn</code> short circuit current of the module at Standard Test Conditions (default value 14.07 amperes.) <code>Vmn</code> maximum power point voltage of the module at Standard Test Conditions (default value 43.76 volts.)

	<p>Imn Maximum power current of the module at Standard Test Conditions (default value 13.03 amperes.)</p> <p>Ncs number of cells in series inside the module (default value 24)</p> <p>Ncp number of cells in parallel inside the module (default value 6)</p> <p>CoefVT coefficient of decrement of voltage of each cell with the temperature (default value 0.0049 volts per celsius degree)</p> <p>TONC nominal operational cell temperature, celsius degree (default value 45).</p>
generator	<p>list of numeric values with information about the generator,</p> <p>Nms number of modules in series (default value 22)</p> <p>Nmp number of modules in parallel (default value 130)</p>
inverter	<p>list of numeric values with information about the DC/AC inverter,</p> <p>Ki vector of three values, coefficients of the efficiency curve of the inverter (default c(0.002, 0.005, 0.008)), or a matrix of nine values (3x3) if there is dependence with the voltage (see references).</p> <p>Pinv nominal inverter power (W) (default value 1.5e6 watts.)</p> <p>Vmin, Vmax minimum and maximum voltages of the MPP range of the inverter (default values 822 and 1300 volts)</p> <p>Gumb minimum irradiance for the inverter to start (W/m²) (default value 20 W/m²)</p>
effSys	<p>list of numeric values with information about the system losses,</p> <p>ModQual average tolerance of the set of modules (%), default value is 3</p> <p>ModDisp module parameter dispersion losses (%), default value is 2</p> <p>OhmDC Joule losses due to the DC wiring (%), default value is 1.5</p> <p>OhmAC Joule losses due to the AC wiring (%), default value is 1.5</p> <p>MPP average error of the MPP algorithm of the inverter (%), default value is 1</p> <p>TrafoMT losses due to the MT transformer (%), default value is 1</p> <p>Disp losses due to stops of the system (%), default value is 0.5</p>

Value

If `inclin` is `data.table` or `Gef` object, the result is a `data.table` object with these components:

Tc	cell temperature, °C.
Voc, Isc, Vmpp, Impp	open circuit voltage, short circuit current, MPP voltage and current, respectively, in the conditions of irradiance and temperature provided by <code>Inclin</code>
Vdc, Idc	voltage and current at the input of the inverter. If no voltage limitation occurs (according to the values of <code>inverter\$Vmax</code> and <code>inverter\$Vmin</code>), their values are identical to <code>Vmpp</code> and <code>Impp</code> . If the limit values are reached a warning is produced
Pdc	power at the input of the inverter, W
Pac	power at the output of the inverter, W
EffI	efficiency of the inverter

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Jansch, M., Schmidt, H. y Schmid, J.: Results on the concerted action on power conditioning and control. 11th European photovoltaic Solar Energy Conference, 1992.
- Baumgartner, F. P., Schmidt, H., Burger, B., Bründlinger, R., Haeberlin, H. and Zehner, M.: Status and Relevance of the DC Voltage Dependency of the Inverter Efficiency. 22nd European Photovoltaic Solar Energy Conference, 2007.
- Alonso Garcia, M. C.: Caracterización y modelado de asociaciones de dispositivos fotovoltaicos. PhD Thesis, CIEMAT, 2005.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fInclin](#), [prodGCPV](#), [fTemp](#).

Examples

```
library("data.table")

inclin <- data.table(Gef = c(200,400,600,800,1000),Ta = 25)

#using default values
fProd(inclin)

#Using a matrix for Ki (voltage dependence)
inv1 <- list(Ki = rbind(c(-0.00019917, 7.513e-06, -5.4183e-09),
c(0.00806, -4.161e-06, 2.859e-08),
c(0.02118, 3.4002e-05, -4.8967e-08)))

fProd(inclin, inverter = inv1)

#Voltage limits of the inverter
inclin <- data.table(Gef = 800,Ta = 30)
gen1 <- list(Nms = 10, Nmp = 11)

prod <- fProd(inclin,generator = gen1)
print(prod)

with(prod, Vdc * Idc / (Vmpp * Imp))
```

C_fPump

*Performance of a centrifugal pump***Description**

Compute the performance of the different parts of a centrifugal pump fed by a frequency converter following the affinity laws.

Usage

```
fPump(pump, H)
```

Arguments

pump	list containing the parameters of the pump to be simulated. It may be a row of pumpCoef .
H	Total manometric head (m).

Value

lim	Range of values of electrical power input
fQ	Function constructed with <code>splinefun</code> relating flow and electrical power
fPb	Function constructed with <code>splinefun</code> relating pump shaft power and electrical power of the motor
fPh	Function constructed with <code>splinefun</code> relating hydraulical power and electrical power of the motor
fFreq	Function constructed with <code>splinefun</code> relating frequency and electrical power of the motor

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Abella, M. A., Lorenzo, E. y Chenlo, F.: PV water pumping systems based on standard frequency converters. *Progress in Photovoltaics: Research and Applications*, 11(3):179–191, 2003, ISSN 1099-159X.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[NmgPVPS](#), [prodPVPS](#), [pumpCoef](#), [splinefun](#).

Examples

```

library("data.table")

data(pumpCoef)
CoefSP8A44 <- subset(pumpCoef, Qn == 8 & stages == 44)

fSP8A44 <- fPump(pump = CoefSP8A44,H = 40)
SP8A44 = with(fSP8A44,{
  Pac = seq(lim[1],lim[2],by = 100)
  Pb = fPb(Pac)
  etam = Pb/Pac
  Ph = fPh(Pac)
  etab = Ph/Pb
  f = fFreq(Pac)
  Q = fQ(Pac)
  result = data.frame(Q,Pac,Pb,Ph,etam,etab,f)})

#Efficiency of the motor, pump and the motor-pump
library("latticeExtra")

SP8A44$etamb = with(SP8A44,etab*etam)
lab = c(expression(eta[motor]), expression(eta[pump]), expression(eta[mp]))
p <- xyplot(etam + etab + etamb ~ Pac,data = SP8A44,type = 'l', ylab = 'Efficiency')

p + glayer(panel.text(x[1], y[1], lab[group.number], pos = 3))

#Mechanical, hydraulic and electrical power
lab <- c(expression(P[pump]), expression(P[hyd]))
p <- xyplot(Pb + Ph ~ Pac,data = SP8A44,type = 'l', ylab = 'Power (W)', xlab = 'AC Power (W)')

p + glayer(panel.text(x[length(x)], y[length(x)], lab[group.number], pos = 3))

#Flow and electrical power
xyplot(Q ~ Pac,data = SP8A44,type = 'l')

```

Description

Compute the daily apparent movement of the Sun from the Earth. This movement is mainly described (for the simulation of photovoltaic systems) by the declination angle, the sunrise angle and the daily extra-atmospheric irradiation.

Usage

```
fSolD(lat, BTd, method = 'michalsky')
```

Arguments

lat	Latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
BTd	Daily temporal base, a POSIXct object which may be the result of <code>fBTd</code> .
method	character, method for the sun geometry calculations to be chosen from 'cooper', 'spencer', 'michalsky' and 'strous'. See references for details.

Value

A `data.table` object with these components:

lat	Latitude (degrees)
decl	Declination angle (radians) for each day of year in dn or BTd
eo	Factor of correction due the eccentricity of orbit of the Earth around the Sun.
ws	Sunrise angle (in radians) for each day of year. Due to the convention which considers that the solar hour angle is negative before midday, this angle is negative.
Bo0d	Extra-atmospheric daily irradiation (watt-hour per squared meter) incident on a horizontal surface
EoT	Equation of Time.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Cooper, P.I., Solar Energy, 12, 3 (1969). "The Absorption of Solar Radiation in Solar Stills"
- Spencer, Search 2 (5), 172, <https://www.mail-archive.com/sundial@uni-koeln.de/msg01050.html>
- Strous: <https://www.aa.quae.nl/en/reken/zonpositie.html>
- Michalsky, J., 1988: The Astronomical Almanac's algorithm for approximate solar position (1950-2050), Solar Energy 40, 227-235
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

Examples

```
library("data.table")

BTd <- fBTd(mode = 'serie')

lat <- 37.2
(fSolD(lat, BTd[100]))
```

```

(fSolD(lat, BTd[100], method = 'strous'))
(fSolD(lat, BTd[100], method = 'spencer'))
(fSolD(lat, BTd[100], method = 'cooper'))

lat <- -37.2
(fSolD(lat, BTd[283]))

#Solar angles along the year
Sold <- fSolD(lat, BTd = fBTd())

xyplot(Sold)

#Calculation of the daylength for several latitudes
library("latticeExtra")

Lats <- c(-60, -40, -20, 0, 20, 40, 60)
NomLats <- ifelse(Lats > 0, paste(Lats,'N', sep = ''),
                 paste(abs(Lats), 'S', sep = ''))
NomLats[Lats == 0] <- '0'

BTd <- fBTd(mode = 'serie')
mat <- matrix(nrow = length(BTd), ncol = length(Lats))
colnames(mat) <- NomLats
WsZ <- data.table(Dates = BTd, mat)

for (i in seq_along(Lats)){
  Soldaux <- fSolD(lat = Lats[i], BTd = fBTd(mode = 'serie'));
  WsZ[,i+1] <- r2h(2*abs(Soldaux$ws))}

p = xyplot(`60S` + `40S` + `20S` + `0` + `20N` + `40N` + `60N` ~ Dates, data = WsZ, type = "l",
           ylab = expression(omega[s] * (h)))
plab = p+glayer(panel.text(x[1], y[1], NomLats[group.number], pos = 2))
print(plab)

```

C_fSolI

Instantaneous apparent movement of the Sun from the Earth

Description

Compute the angles which describe the intraday apparent movement of the Sun from the Earth.

Usage

```
fSolI(sold, sample = 'hour', BTi, EoT = TRUE, keep.night = TRUE, method = 'michalsky')
```

Arguments

`sold` A data.table object with the result of fSolD

sample	Increment of the intradaily sequence. It is a character string, containing one of "sec", "min", "hour". This can optionally be preceded by a (positive or negative) integer and a space, or followed by "s". It is used by seq.POSIXt . It is not considered when BTi is provided.
BTi	Intradaily time base, a POSIXct object. It could be the index of the G0I argument to calcG0 . fSolI will produce results only for those days contained both in sold and in BTi.
EoT	logical, if TRUE (default) the Equation of Time is used.
keep.night	logical, if TRUE (default) the night is included in the time series.
method	character, method for the sun geometry calculations to be chosen from 'cooper', 'spencer', 'michalsky' and 'strous'. See references for details.

Value

A data.table object is returned with these components:

lat	numeric, latitude (degrees)
w	numeric, solar hour angle (radians)
aman	logical, TRUE when Sun is above the horizon
cosThzS	numeric, cosine of the solar zenith angle
AzS	numeric, solar acimuth angle (radians)
AlS	numeric, solar elevation angle (radians)
Bo0	numeric, extra-atmospheric irradiance (W/m2)

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Cooper, P.I., Solar Energy, 12, 3 (1969). "The Absorption of Solar Radiation in Solar Stills"
- Spencer, Search 2 (5), 172, <https://www.mail-archive.com/sundial@uni-koeln.de/msg01050.html>
- Strous: <https://www.aa.quae.nl/en/reken/zonpositie.html>
- Michalsky, J., 1988: The Astronomical Almanac's algorithm for approximate solar position (1950-2050), Solar Energy 40, 227-235
- Collares-Pereira, M. y Rabl, A., The average distribution of solar radiation: correlations between diffuse and hemispherical and between daily and hourly insolation values. Solar Energy, 22:155–164, 1979.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fSold](#)

Examples

```

library("data.table")

###Angles for one day
BTd <- fBTd(mode = 'serie')

#North hemisphere
lat <- 37.2
solD <- fSolD(lat,BTd[100])
solI <- fSolI(solD, sample = 'hour')
print(solI)

#South hemisphere
lat <- -37.2;
solDs <- fSolD(lat,BTd[283])
solIs <- fSolI(solDs, sample = 'hour')
print(solIs)

###Angles for the 12 average days
lat <- 37.2;
solD <- fSolD(lat,BTd = fBTd(mode = 'prom'))
solI <- fSolI(solD, sample = '10 min', keep.night = FALSE)

###Solar elevation angle vs. azimuth.
#This kind of graphics is useful for shadows calculations
library("latticeExtra")

mon <- month.abb
p <- xyplot(r2d(AIS)~r2d(AzS),
  groups = month(Dates),
  data = solI, type = 'l', col = 'black',
  xlab = expression(psi[s]),ylab = expression(gamma[s]))

plab <- p + glayer({
  idx <- round(length(x)/2+1)
  panel.text(x[idx], y[idx], mon[group.value], pos = 3, offset = 0.2, cex = 0.8)})

print(plab)

```

Description

Compute the shadows factor for two-axis and horizontal N-S axis trackers and fixed surfaces.

Usage

```
fSombra(angGen, distances, struct, modeTrk = 'fixed', prom = TRUE)
```

```
fSombra6(angGen, distances, struct, prom = TRUE)
```

```
fSombra2X(angGen, distances, struct)
```

```
fSombraHoriz(angGen, distances, struct)
```

```
fSombraEst(angGen, distances, struct)
```

Arguments

angGen	A data.table object, including at least variables named Beta, Alpha, AzS, ALS and cosTheta.
distances	data.frame, with a component named Lew, being the distance (meters) between horizontal NS and two-axis trackers along the East-West direction, a component named Lns for two-axis trackers or a component named D for static surfaces. An additional component named H can be included with the relative height (meters) between surfaces. When modeTrk = 'two' (or when fSombra6 is used) this data.frame may have five rows. Each of these rows defines the distances of a tracker in a set of six ones.
struct	list. When modeTrk = 'fixed' or modeTrk = 'horiz' only a component named L, which is the height (meters) of the tracker, is needed. For two-axis trackers (modeTrk = 'two'), an additional component named W, the width of the tracker, is required. Moreover, two components named Nrow and Ncol are included under this list. These components define, respectively, the number of rows and columns of the whole set of trackers in the PV plant.
modeTrk	character, to be chosen from 'fixed', 'two' or 'horiz'. When modeTrk = 'fixed' the surface is fixed (inclination and azimuth angles are constant). The performance of a two-axis tracker is calculated with modeTrk = 'two', and modeTrk = 'horiz' is the option for an horizontal N-S tracker. Its default value is modeTrk = 'fixed'
prom	logical, only needed for two-axis tracker mode. If TRUE the shadows are averaged between the set of trackers defined by struct\$Nrow and struct\$Ncol

Details

fSombra is only a wrapper for fSombra6 (two-axis trackers), fSombraEst (fixed systems) and fSombraHoriz (horizontal N-S axis trackers). Depending on the value of modeTrk the corresponding function is selected. fSombra6 calculates the shadows factor in a set of six two-axis trackers. If distances has only one row, this function constructs a symmetric grid around a tracker located at (0,0,0). These five trackers are located at (-Lew, Lns, H), (0, Lns, H), (Lew, Lns, H), (-Lew, 0, H) and (Lns, 0, H). It is possible to define a irregular grid around (0,0,0) including five rows in distances. When prom = TRUE the shadows factor for each of the six trackers is calculated. Then, according to the distribution of trackers in the plant defined by struct\$Nrow and struct\$Ncol, a

weighted average of the shadows factors is the result. It is important to note that the distances are defined between axis for trackers and between similar points of the structure for fixed surfaces.

Value

data.table including angGen and a variable named FS, which is the shadows factor. This factor is the ratio between the area of the generator affected by shadows and the total area. Therefore its value is 1 when the PV generator is completely shadowed.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O.: Grandes Centrales Fotovoltaicas: producción, seguimiento y ciclo de vida. PhD Thesis, UNED, 2008. https://www.researchgate.net/publication/39419806_Grandes_Centrales_Fotovoltaicas_Produccion_Seguimiento_y_Ciclo_de_Vida.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[calcShd](#), [optimShd](#), [fTheta](#), [calcSol](#)

Examples

```
library("data.table")

lat <- 37.2;
sol <- calcSol(lat, fBTd(mode = 'prom'), sample = '10 min', keep.night = FALSE)
angGen <- fTheta(sol, beta = 35);
Angles <- merge(as.data.tableI(sol), angGen)

###Two-axis tracker
#Symmetric grid
distances = data.table(Lew = 40,Lns = 30,H = 0)
struct = list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 8)

ShdFactor <- fSombra6(Angles, distances, struct, prom = FALSE)

Angles$FS = ShdFactor
xyplot(FS ~ w, groups = month(Dates), data = Angles,
       type = 'l',
       auto.key = list(space = 'right',
                       lines = TRUE,
                       points = FALSE))

#Symmetric grid defined with a five rows data.frame
```

```

distances = data.table(Lew = c(-40,0,40,-40,40),
                       Lns = c(30,30,30,0,0),
                       H = 0)
ShdFactor2 <- fSombra6(Angles, distances, struct,prom = FALSE)

#of course, with the same result
identical(ShdFactor, ShdFactor2)

```

C_fTemp

Intradaily evolution of ambient temperature

Description

From the maximum and minimum daily values of ambient temperature, its evolution is calculated through a combination of cosine functions (ESRA method)

Usage

```
fTemp(sol, BD)
```

Arguments

sol	A <code>Sol</code> object. It may be the result of the <code>calcSol</code> function.
BD	A <code>Meteo</code> object, as provided by the <code>readBDd</code> function. It must include information about TempMax and TempMin.

Details

The ESRA method estimates the dependence of the temperature on the time of the day (given as the local solar time) from only two inputs: minimum and maximum daily temperatures. It assumes that the temperature daily profile can be described using three piecewise cosine functions, dividing the day into three periods: from midnight to sunrise, from sunrise to the time of peak temperature (3 hours after midday), and to midnight.

Value

A `data.table` object with the profile of the ambient temperature.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Huld, T., Suri, M., Dunlop, E. D., and Micale F., Estimating average daytime and daily temperature profiles within Europe, *Environmental Modelling & Software* 21 (2006) 1650-1661.
- Perpiñán, O., *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[calcSol](#), [readBDd](#).

C_fTheta

Angle of incidence of solar irradiation on a inclined surface

Description

The orientation, azimuth and incidence angle are calculated from the results of `fSolI` or `calcSol` and from the information supplied by the arguments `beta` and `alpha` when the surface is fixed (`modeTrk = 'fixed'`) or the movement equations when a tracking surface is chosen (`modeTrk = 'horiz'` or `modeTrk = 'two'`). Besides, the modified movement of a horizontal NS tracker due to the backtracking strategy is calculated if `BT = TRUE` with information about the tracker and the distance between the trackers included in the system.

This function is used by the `calcGef` function.

Usage

```
fTheta(sol, beta, alpha = 0, modeTrk = "fixed", betaLim = 90,
       BT = FALSE, struct, dist)
```

Arguments

<code>sol</code>	Sol object as provided by <code>calcSol</code> .
<code>beta</code>	numeric, inclination angle of the surface (degrees). It is only needed when <code>modeTrk = 'fixed'</code> .
<code>alpha</code>	numeric, azimuth angle of the surface (degrees). It is measured from the south (<code>alpha = 0</code>), and it is negative to the east and positive to the west. It is only needed when <code>modeTrk = 'fixed'</code> . Its default value is <code>alpha = 0</code> (surface facing to the south).
<code>modeTrk</code>	character, to be chosen from <code>'fixed'</code> , <code>'two'</code> or <code>'horiz'</code> . When <code>modeTrk = 'fixed'</code> the surface is fixed (inclination and azimuth angles are constant). The performance of a two-axis tracker is calculated with <code>modeTrk = 'two'</code> , and <code>modeTrk = 'horiz'</code> is the option for an horizontal N-S tracker. Its default value is <code>modeTrk = 'fixed'</code>
<code>betaLim</code>	numeric, maximum value of the inclination angle for a tracking surface. Its default value is 90 (no limitation)
<code>BT</code>	logical, TRUE when the backtracking technique is to be used with a horizontal NS tracker, as described by Panico et al. (see References). The default value is FALSE. In future versions of this package this technique will be available for two-axis trackers.
<code>struct</code>	Only needed when <code>BT = TRUE</code> . A list, with a component named <code>L</code> , which is the height (meters) of the tracker. In future versions the backtracking technique will be used in conjunction with two-axis trackers, and a additional component named <code>W</code> will be needed.

`dist` Only needed when `BT = TRUE`. A `data.frame`, with a component named `Lew`, being the distance between the horizontal NS trackers along the East-West direction. In future versions an additional component named `Lns` will be needed for two-axis trackers with backtracking.

Value

A `data.table` object with these components:

`Beta` numeric, inclination angle of the surface (radians). When `modeTrk = 'fixed'` it is the value of the argument `beta` converted from degrees to radians.

`Alpha` numeric, azimuth angle of the surface (radians). When `modeTrk = 'fixed'` it is the value of the argument `alpha` converted from degrees to radians.

`cosTheta` numeric, cosine of the incidence angle of the solar irradiance on the surface

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Panico, D., Garvison, P., Wenger, H. J., Shugar, D., Backtracking: a novel strategy for tracking PV systems, Photovoltaic Specialists Conference, 668-673, 1991
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fInclin](#), [fSombra](#), [calcGef](#).

C_HQCurve

H-Q curves of a centrifugal pump

Description

Compute and display the H-Q curves of a centrifugal pump fed working at several frequencies, and the iso-efficiency curve as a reference.

Usage

`HQCurve(pump)`

Arguments

`pump` list containing the parameters of the pump to be simulated. It may be a row of [pumpCoef](#).

Value

result	A data.frame with the result of the simulation. It contains several columns with values of manometric height (H), frequency (fe and fb), mechanical power (Pb), AC electrical power (Pm), DC electrical power (Pdc) and efficiency of the pump (etab) and motor (etam).
plot	The plot with several curves labelled with the correspondent frequencies, and the isoefficiency curve (named "ISO").

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Abella, M. A., Lorenzo, E. y Chenlo, F.: PV water pumping systems based on standard frequency converters. Progress in Photovoltaics: Research and Applications, 11(3):179–191, 2003, ISSN 1099-159X.
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[NmgPVPS](#), [prodPVPS](#), [pumpCoef](#).

Examples

```
library("data.table")

data(pumpCoef)

CoefSP8A44 <- subset(pumpCoef, Qn == 8 & stages == 44)
CurvaSP8A44 <- HQCurve(pump = CoefSP8A44)
```

C_local2Solar

Local time, mean solar time and UTC time zone.

Description

The function `local2Solar` converts the time zone of a `POSIXct` object to the mean solar time and set its time zone to UTC as a synonym of mean solar time. It includes two corrections: the difference of longitudes between the location and the time zone, and the daylight saving time.

The function `lonHH` calculates the longitude (radians) of a time zone.

Usage

```
local2Solar(x, lon = NULL)
lonHH(tz)
```

Arguments

x	a POSIXct object
lon	A numeric value of the longitude (degrees) of the location. If lon = NULL (default), this value is assumed to be equal to the longitude of the time zone of x, so only the daylight saving time correction (if needed) is included.
tz	A character, a time zone as documented in https://en.wikipedia.org/wiki/List_of_tz_database_time_zones .

Details

Since the result of `local2Solar` is the mean solar time, the Equation of Time correction is not calculated with this function. The `eot` function includes this correction if desired.

Value

The function `local2Solar` produces a POSIXct object with its time zone set to UTC.

The function `lonHH` gives a numeric value.

Note

It is important to note that the `solaR2` package sets the system time zone to UTC with `Sys.setenv(TZ = 'UTC')`.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

Examples

```
library("data.table")

t.local <- as.POSIXct("2006-01-08 10:07:52", tz = 'Europe/Madrid')

##The local time zone and the location have the same longitude (15 degrees)
local2Solar(t.local)
##But Madrid is at lon = -3
local2Solar(t.local, lon = -3)
```



```
##Daylight saving time
t.local.dst <- as.POSIXct("2006-07-08 10:07:52", tz = 'Europe/Madrid')

local2Solar(t.local.dst)
local2Solar(t.local.dst, lon = -3)
```

C_NmgPVPS

*Nomogram of a photovoltaic pumping system***Description**

This function simulate the performance of a water pump fed by a frequency converter with several PV generators of different size during a day. The result is plotted as a nomogram which relates the nominal power of the PV generator, the total water flow and the total manometric head.

Usage

```
NmgPVPS(pump, Pg, H, Gd, Ta = 30,
         lambda = 0.0045, TONC = 47, eta = 0.95,
         Gmax = 1200, t0 = 6, Nm = 6,
         title = '', theme = custom.theme.2())
```

Arguments

pump	A list extracted from pumpCoef
Pg	Sequence of values of the nominal power of the PV generator (Wp))
H	Sequence of values of the total manometric head (m)
Gd	Global irradiance incident on the generator (Wh/m ²)
Ta	Ambient temperature (°C).
lambda	Power losses factor due to temperature
TONC	Nominal operational cell temperature (°C).
eta	Average efficiency of the frequency converter
Gmax	Maximum value of irradiance (parameter of the IEC 61725)
t0	Hours from midday to sunset (parameter of the IEC 61725)
Nm	Number of samples per hour
title	Main title of the plot.
theme	Theme of the lattice plot.

Details

This function computes the irradiance profile according to the IEC 61725 "Analytical Expression for Daily Solar Profiles", which is a common reference in the official documents regarding PV pumping systems. At this version only pumps from the manufacturer Grundfos are included in [pumpCoef](#).

Value

I	list with the results of irradiance, power and flow of the system.
D	list with the results of total irradiation, electrical energy and flow for every nominal power of the generator.
param	list with the arguments used in the call to the function.
plot	trellis object containing the nomogram.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Abella, M. A., Lorenzo, E. y Chenlo, F.: PV water pumping systems based on standard frequency converters. *Progress in Photovoltaics: Research and Applications*, 11(3):179–191, 2003, ISSN 1099-159X.
- Perpiñán, O, *Energía Solar Fotovoltaica*, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", *Journal of Statistical Software*, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fPump](#), [prodPVPS](#), [pumpCoef](#)

Examples

```
library("data.table")

Pg = seq(4000, 8000, by = 100);
H = seq(120, 150, by = 5);

data(pumpCoef)

CoefSP8A44 <- subset(pumpCoef, Qn == 8 & stages == 44)

NmgSP8A44 <- NmgPVPS(pump = CoefSP8A44, Pg = Pg, H = H, Gd = 5000,
  title = 'Choice of Pump', theme = custom.theme())
```

Description

diff2Hours converts a difftime object into its numeric value with units = 'hours'.

char2diff converts a character description into a difftime object, following the code of [seq.POSIXt](#).

sample2Hours calculates the sampling time in hours described by a character or a difftime.

P2E (power to energy) sums a series of power values (for example, irradiance) to obtain energy aggregation (for example, irradiation) using sample2Hours for the units conversion.

Usage

```
diff2Hours(by)
char2diff(by)
sample2Hours(by)
P2E(x, by)
```

Arguments

by	A character for char2diff, sample2Hours and P2E, or a difftime for diff2Hours, sample2Hours and P2E.
x	A numeric vector.

Value

A numeric value or a difftime object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[Sol](#)

Examples

```
library("data.table")

char2diff('min')
char2diff('2 s')

sample2Hours('s')
sample2Hours('30 m')

by1 <- char2diff('10 min')
sample2Hours(by1)
```

C_solarAngles	<i>Solar angles</i>
---------------	---------------------

Description

A set of functions that compute the apparent movement of the Sun from the Earth.

Usage

```

## Declination
declination(d, method = 'michalsky')

## Eccentricity
eccentricity(d, method = 'michalsky')

## Equation of time
eot(d)

## Solar time
sunrise(d, lat, method = 'michalsky',
        decl = declination(d, method = method))

## Extraterrestrial irradiation
bo0d(d, lat, method = 'michalsky',
     decl = declination(d, method = method),
     eo = eccentricity(d, method = method),
     ws = sunrise(d, lat, method = method))

## Sun hour angle
sunHour(d, BTi, sample = 'hour', EoT = TRUE,
        method = 'michalsky',
        eqtime = eot(d))

## Cosine of the zenith angle
zenith(d, lat, BTi, sample = 'hour', method = 'michalsky',
      decl = declination(d, method = method),
      w = sunHour(d, BTi, sample, method = method))

## Azimuth angle
azimuth(d, lat, BTi, sample = 'hour', method = 'michalsky',
      decl = declination(d, method = method),
      w = sunHour(d, BTi, sample, method = method),
      cosThzS = zenith(d, lat, BTi, sample,
                       method = method,
                       decl = decl,
                       w = w))

```

Arguments

d	Date, a daily time base, it may be the result of <code>fBTd</code>
method	character, method for the sun geometry calculations, to be chosen from 'cooper', 'spencer', 'michalsky' and 'strous'. See references for details.
lat	numeric, latitude (degrees) of the point of the Earth where calculations are needed. It is positive for locations above the Equator.
sample	Character, increment of the intradaily sequence.
BTi	POSIXct, intradaily time base, it may the result of <code>fBTi</code> .
EoT	logical, if EoT=TRUE (default value), the function sunHour use the Equation of time
decl, eo, ws, eqtime, w, cosThzS	Arguments that compute the variables they reference (default value). It can be replaced with previously calculated values to avoid calculating the same variable twice.

Value

A vector with the calculated elements. Its size varies depending on whether the calculations are daily or intradaily.

Author(s)

Francisco Delgado López, Oscar Perpiñán Lamigueiro.

References

- Cooper, P.I., Solar Energy, 12, 3 (1969). "The Absorption of Solar Radiation in Solar Stills"
- Spencer, Search 2 (5), 172, <https://www.mail-archive.com/sundial@uni-koeln.de/msg01050.html>
- Strous: <https://www.aa.quae.nl/en/reken/zonpositie.html>
- Michalsky, J., 1988: The Astronomical Almanac's algorithm for approximate solar position (1950-2050), Solar Energy 40, 227-235
- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

`fSold`, `fSolI`, `calcSol`

Examples

```
library("data.table")
```

```
d = fBTd(mode = 'serie')[100]
```

```
decl = declination(d, method = 'michalsky')
decl

w = sunHour(d, sample = 'hour', method = 'michalsky')
w

cosThzS = zenith(d, lat = 37.2, sample = 'hour',
                 method = 'michalsky',
                 decl = decl,
                 w = w)
cosThzS
```

C_utils-angle

Conversion between angle units.

Description

Several small functions to convert angle units.

Usage

```
d2r(x)
r2d(x)
h2r(x)
h2d(x)
r2h(x)
d2h(x)
r2sec(x)
```

Arguments

x A numeric value.

Value

A numeric value:

d2r: Degrees to radians.

r2d: Radians to degrees.

h2r: Hours to radians.

r2h: Radians to hours.

h2d: Hours to degrees.

d2h: Degrees to hours.

r2sec: Radians to seconds.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

C_utils-time

Utilities for time indexes.

Description

Several small functions to extract information from POSIXct indexes.

Usage

hms(x)
doy(x)
dom(x)
dst(x)
truncDay(x)

Arguments

x A POSIXct vector.

Value

doy and dom provide the (numeric) day of year and day of month, respectively.

hms gives the numeric value

$\text{hour}(x) + \text{minute}(x)/60 + \text{second}(x)/3600$

dst is +1 if the Daylight Savings Time flag is in force, zero if not, -1 if unknown ([DateTimeClasses](#)).

truncDay truncates the POSIXct object towards the day.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

as.POSIXct

D_as.data.tableD-methods

Methods for Function as.data.tableD

Description

Convert a Sol, G0, Gef, ProdGCPV or ProdPVPS object into a data.table object with daily values.

Usage

```
## S4 method for signature 'Sol'
as.data.tableD(object, complete=FALSE, day=FALSE)
```

Arguments

object	A Sol object (or extended.)
complete	A logical.
day	A logical.

Methods

signature(object = "Sol") Conversion to a data.table object with the content of the sold slot. If day=TRUE (default is FALSE), the result includes three columns named month, day (day of the year) and year.

signature(object = "G0") If complete=FALSE (default) the result includes only the columns of G0d, D0d and B0d from the G0D slot. If complete=TRUE it returns the contents of the slots sold and G0D.

signature(object = "Gef") If complete=FALSE (default) the result includes only the columns of Gefd, Defd and Befd from the GefD slot. If complete=TRUE it returns the contents of the slots sold, G0D and GefD

signature(object = "ProdGCPV") If complete=FALSE (default) the result includes only the columns of Eac, Edc and Yf from the prodD slot. If complete=TRUE it returns the contents of the slots sold, G0D, GefD and prodD.

signature(object = "ProdPVPS") If complete=FALSE (default) the result includes only the columns of Eac, Qd and Yf from the prodD slot. If complete=TRUE it returns the contents of the slots sold, G0D, GefD and prodD.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```

library("data.table")

lat = 37.2
BTd = fBTd(mode = 'prom')
sol = calcSol(lat, BTd)
solD = as.data.tableD(sol)
solD

solD2 = as.data.tableD(sol, day = TRUE)
solD2

G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562,
          2814, 2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
        17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)
prodfixed = prodGCPV(lat, dataRad = prom)
prodD = as.data.tableD(prodfixed, complete = TRUE, day = TRUE)
prodD

```

D_as.data.tableI-methods

Methods for Function as.data.tableI

Description

Convert a Sol, G0, Gef, ProdGCPV or ProdPVPS object into a data.table object with daily values.

Usage

```

## S4 method for signature 'Sol'
as.data.tableI(object, complete=FALSE, day=FALSE)

```

Arguments

object	A Sol object (or extended.)
complete	A logical.
day	A logical.

Methods

signature(object = "Sol") If complete=FALSE and day=FALSE (default) the result includes only the content of the solI slot. If complete=TRUE the contents of the solD slots are included.

signature(object = "G0") If complete=FALSE and day=FALSE (default) the result includes only the columns of G0, D0 and B0 of the G0I slot. If complete=TRUE it returns the contents of the slots G0I and solI. If day=TRUE the daily values (slots G0D and solD) are also included.)

signature(object = "Gef") If complete=FALSE and day=FALSE (default) the result includes only the columns of Gef, Def and Bef of the GefI slot. If complete=TRUE it returns the contents of the slots GefI, G0I and solI. If day=TRUE the daily values (slots GefD, G0D and solD) are also included.)

signature(object = "ProdGCPV") If complete=FALSE and day=FALSE (default) the result includes only the columns of Pac and Pdc of the prodI slot. If complete=TRUE it returns the contents of the slots prodI, GefI, G0I and solI. If day=TRUE the daily values (slots prodD, GefD, G0D and solD) are also included.)

signature(object = "ProdPVPS") If complete=FALSE and day=FALSE (default) the result includes only the columns of Pac and Q of the prodI slot. If complete=TRUE it returns the contents of the slots prodI, GefI, G0I and solI. If day=TRUE the daily values (slots prodD, GefD, G0D and solD) are also included.)

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```
library("data.table")

lat = 37.2
BTd = fBTd(mode = 'prom')[1]
sol = calcSol(lat, BTd, keep.night = FALSE)
solI = as.data.tableI(sol)
solI

solI2 = as.data.tableI(sol, day = TRUE)
solI2

G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562,
          2814, 2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
        17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)
prodfixed = prodGCPV(lat, dataRad = prom)
prodI = as.data.tableI(prodfixed, complete = TRUE, day = TRUE)
prodI
```

D_as.data.tableM-methods

Methods for Function as.data.tableM

Description

Convert a G0, Gef, ProdGCPV or ProdPVPS object into a as.data.table object with monthly average of daily values.

Usage

```
## S4 method for signature 'G0'
as.data.tableM(object, complete=FALSE, day=FALSE)
```

Arguments

object	A G0 object (or extended.)
complete	A logical.
day	A logical

Methods

signature(object = "G0") The result is the G0dm slot. If day=TRUE (default is FALSE), the result includes two columns names month and year.

signature(object = "Gef") If complete=FALSE (default) the result is the slot Gefdm. If complete=TRUE it returns the slot G0dm.

signature(object = "ProdGCPV") If complete=FALSE (default) the result is the prodDm slot. If complete=TRUE the result includes the slots G0dm and Gefdm.

signature(object = "ProdPVPS") If complete=FALSE (default) the result is the prodDm slot. If complete=TRUE the result includes the slots G0dm and Gefdm.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```
library("data.table")

lat <- 37.2
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562,
         2814, 2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
       17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)
prodfixed <- prodGCPV(lat, dataRad = prom)
prodM <- as.data.tableM(prodfixed, complete = TRUE, day = TRUE)
prodM
```

D_as.data.tableY-methods

Methods for Function as.data.tableY

Description

Convert a G0, Gef, ProdGCPV or ProdPVPS object into a data.table object with yearly values.

Usage

```
## S4 method for signature 'G0'
as.data.tableY(object, complete=FALSE, day=FALSE)
```

Arguments

object	A G0 object (or extended.)
complete	A logical.
day	A logical.

Methods

signature(object = "G0") The result is the G0y slot. If day = TRUE (default is FALSE), the result includes a column named year.

signature(object = "Gef") If complete=FALSE (default) the result is the slot Gefy. If complete=TRUE it returns the slot G0y.

signature(object = "ProdGCPV") If complete=FALSE (default) the result is the prody slot. If complete=TRUE the result includes the slots G0y and Gefy.

signature(object = "ProdPVPS") If complete=FALSE (default) the result is the prody slot. If complete=TRUE the result includes the slots G0y and Gefy.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```
library("data.table")

lat <- 37.2
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562,
         2814, 2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
        17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)
prodfixed = prodGCPV(lat, dataRad = prom)
prodY = as.data.tableY(prodfixed, complete = TRUE, day = TRUE)
prodY
```

D_compare-methods *Compare G0, Gef and ProdGCPV objects*

Description

Compare and plot the yearly values of several objects.

Usage

```
## S4 method for signature 'G0'  
compare(...)
```

Arguments

... A list of objects to be compared.

Methods

The class of the first element of ... is used to determine the suitable method. The result is plotted with [dotplot](#):

```
signature(... = "G0") yearly values of G0d, B0d and D0d.  
signature(... = "Gef") yearly values of Gefd, Befd and Defd.  
signature(... = "ProdGCPV") yearly values of Yf, Gefd and G0d.
```

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[dotplot](#)

Examples

```
library("data.table")  
  
lat <- 37.2;  
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562, 2814,  
2179)  
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)  
prom <- list(G0dm = G0dm, Ta = Ta)  
  
###Comparison of different tracker methods  
ProdFixed <- prodGCPV(lat = lat, dataRad = prom, keep.night = FALSE)  
Prod2x <- prodGCPV(lat = lat, dataRad = prom, modeTrk = 'two', keep.night = FALSE)  
ProdHoriz <- prodGCPV(lat = lat, dataRad = prom, modeTrk = 'horiz', keep.night = FALSE)
```

```
compare(ProdFixed, Prod2x, ProdHoriz)

##The first element rules the method
GefFixed <- as(ProdFixed, 'Gef')
compare(GefFixed, Prod2x, ProdHoriz)
```

D_getData-methods *Methods for function getData*

Description

Meteorological source data of a Meteo (or extended) object.

Methods

signature(object = "Meteo") returns the meteorological source data of the slot data of the object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_getG0-methods *Methods for function getG0*

Description

Global irradiation source data of a Meteo (or extended) object.

Methods

signature(object = "Meteo") returns the global irradiation values stored in a Meteo object.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_getLat-methods *Methods for Function getLat*

Description

Latitude angle of solar objects.

Usage

```
getLat(object, units='rad')
```

Arguments

object	A Sol or Meteo object (or extended.)
units	A character, 'rad' or 'deg'.

Methods

This function returns the latitude angle in radians (units='rad', default) or degrees (units='deg').

signature(object = "Meteo") Value of the latData slot, which is defined by the argument lat of the readG0dm and readBDd functions, or by the lat component of the dataRad object passed to calcG0 (or equivalent). It is the latitude of the meteorological station (or equivalent) which provided the irradiation source data. It may be different from the value used for the calculation procedure.

signature(object = "Sol") Value of the lat slot, which is defined by the argument lat of the calcSol function. It is the value used through the calculation procedure.

signature(object = "G0") same as for the Sol class.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_indexD-methods *Methods for Function indexD*

Description

Daily time index of solar objects.

Methods

signature(object = "Meteo") returns the index of the data slot (a data.table object.)

signature(object = "Sol") returns the index of the solD slot (a data.table object.)

signature(object = "G0") same as for object='Sol'

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_indexI-methods

Methods for Function indexI

Description

Intra-daily time index of solar objects.

Methods

signature(object = "Sol") returns the index of the slot solI (a data.table object).

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_levelplot-methods

Methods for function levelplot.

Description

Methods for function levelplot and zoo and solar objects.

Methods

signature(x = "formula", data = "Meteo"): The Meteo object is converted into a data.table object, and the previous method is used.

signature(x = "formula", data = "Sol"): idem

signature(x = "formula", data = "G0"): idem

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Description

The function `losses` calculates the yearly losses from a `Gef` or a `ProdGCPV` object. The function `compareLosses` compares the losses from several `ProdGCPV` objects and plots the result with `dotplot`.

Usage

```
compareLosses(...)  
losses(object)
```

Arguments

`...` A list of `ProdGCPV` objects to be compared.
`object` An object of `Gef` or `ProdGCPV` class..

Methods

`signature(... = "Gef")` shadows and angle of incidence (AoI) losses.
`signature(... = "ProdGCPV")` shadows, AoI, generator (mainly temperature), DC and AC system (as detailed in `effSys` of `fProd`) and inverter losses.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

References

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

See Also

[fInclin](#), [fProd](#)

Examples

```
library("data.table")  
setDTthreads(2)  
  
lat <- 37.2;  
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562, 2814,  
2179)  
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
```

```

prom <- list(G0dm = G0dm, Ta = Ta)

###Comparison of different tracker methods
ProdFixed <- prodGCPV(lat = lat,dataRad = prom, keep.night = FALSE)
Prod2x <- prodGCPV(lat = lat, dataRad = prom, modeTrk = 'two', keep.night = FALSE)
ProdHoriz <- prodGCPV(lat = lat,dataRad = prom, modeTrk = 'horiz', keep.night = FALSE)

losses(ProdFixed)
losses(as(ProdFixed, 'Gef'))

compareLosses(ProdFixed, Prod2x, ProdHoriz)

```

D_mergesolaR-methods *Merge solaR objects*

Description

Merge the daily time series of solaR objects

Usage

```

## S4 method for signature 'G0'
mergesolaR(...)

```

Arguments

... A list of objects to be merged.

Methods

The class of the first element of ... is used to determine the suitable method. Only the most important daily variable is merged, depending on the class of the objects:

```

signature(... = "Meteo") G0
signature(... = "G0") G0d
signature(... = "Gef") Gefd
signature(... = "ProdGCPV") Yf
signature(... = "ProdPVPS") Yf

```

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

Examples

```
library("data.table")

lat <- 37.2;
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562, 2814,
2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)

###Different tracker methods
ProdFixed <- prodGCPV(lat = lat,dataRad = prom, keep.night = FALSE)
Prod2x <- prodGCPV(lat = lat, dataRad = prom, modeTrk = 'two', keep.night = FALSE)
ProdHoriz <- prodGCPV(lat = lat,dataRad = prom, modeTrk = 'horiz', keep.night = FALSE)

prod <- mergesolaR(ProdFixed, Prod2x, ProdHoriz)
head(prod)
```

D_shadeplot-methods *Methods for Function shadeplot*

Description

Visualization of the content of a [Shade](#) object.

Methods

`signature(x = "Shade")` display the results of the iteration with a level plot for the two-axis tracking, or with conventional plot for horizontal tracking and fixed systems.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

D_window-methods *Methods for extracting a time window*

Description

Method for extracting the subset of a `solaR` object whose daily time index ([indexD](#)) is comprised between the times `i` and `j`.

Usage

```
## S4 method for signature 'Meteo'
x[i, j, ..., drop = TRUE]
## S4 method for signature 'Sol'
x[i, j, ..., drop = TRUE]
## S4 method for signature 'G0'
x[i, j, ..., drop = TRUE]
## S4 method for signature 'Gef'
x[i, j, ..., drop = TRUE]
## S4 method for signature 'ProdGCPV'
x[i, j, ..., drop = TRUE]
## S4 method for signature 'ProdPVPS'
x[i, j, ..., drop = TRUE]
```

Arguments

x	A Meteo, Sol, etc. object.
i	an index/time value (Date or POSIXct classes) defining the start of the time window.
j	an index/time value (Date or POSIXct classes) defining the end of the time window.
..., drop	Additional arguments for window.zoo

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[indexD](#)

Examples

```
library("data.table")

lat <- 37.2
sol <- calcSol(lat, BTd = fBTd(mode = 'serie'))
range(indexD(sol))

start <- as.Date(indexD(sol)[1])
end <- start + 30

solWindow <- sol[start, end]
range(indexD(solWindow))
```

Description

Exports the results of the solaR functions as text files using [write.table](#)

Usage

```
## S4 method for signature 'Sol'
writeSolar(object, file, complete = FALSE,
           day = FALSE, timeScales = c('i', 'd', 'm', 'y'), sep = ',', ...)
```

Arguments

object	A Sol object (or extended.)
file	A character with the name of the file.
complete	A logical. Should all the variables be exported?
day	A logical. Should be daily values included in the intradaily file?
timeScales	A character. Use 'i' to export intradaily values, 'd' for daily values, 'm' for monthly values and 'y' for yearly values. A different file will be created for each choice.
sep	The field separator character.
...	Additional arguments for write.table

Methods

`signature(object = "Sol")` This function exports the slots with results using [write.table](#). If `complete = FALSE` and `day = FALSE` (default) the result includes only the content of the `solI` slot. If `day = TRUE` the contents of the `solD` slot are included.

`signature(object = "G0")` If `complete = FALSE` and `day = FALSE` (default) the result includes only the columns of `G0`, `D0` and `B0` of the `G0I` slot. If `complete = TRUE` it returns the contents of the slots `G0I` and `solI`. If `day = TRUE` the daily values (slots `G0D` and `solD`) are also included.

`signature(object = "Gef")` If `complete = FALSE` and `day = FALSE` (default) the result includes only the columns of `Gef`, `Def` and `Bef` of the `GefI` slot. If `complete = TRUE` it returns the contents of the slots `GefI`, `G0I` and `solI`. If `day = TRUE` the daily values (slots `GefD`, `G0D` and `solD`) are also included.

`signature(object = "ProdGCPV")` If `complete = FALSE` and `day = FALSE` (default) the result includes only the columns of `Pac` and `Pdc` of the `prodI` slot. If `complete = TRUE` it returns the contents of the slots `prodI`, `GefI`, `G0I` and `solI`. If `day = TRUE` the daily values (slots `prodD`, `GefD`, `G0D` and `solD`) are also included.

`signature(object = "ProdPVPS")` If `complete = FALSE` and `day = FALSE` (default) the result includes only the columns of `Pac` and `Q` of the `prodI` slot. If `complete = TRUE` it returns the contents of the slots `prodI`, `GefI`, `G0I` and `solI`. If `day = TRUE` the daily values (slots `prodD`, `GefD`, `G0D` and `solD`) are also included.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

See Also

[write.table](#), [fread](#), [as.data.tableI](#), [as.data.tableD](#), [as.data.tableM](#), [as.data.tableY](#)

Examples

```
library("data.table")

lat <- 37.2;
G0dm <- c(2766, 3491, 4494, 5912, 6989, 7742, 7919, 7027, 5369, 3562, 2814, 2179)
Ta <- c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
prom <- list(G0dm = G0dm, Ta = Ta)

prodFixed <- prodGCPV(lat = lat, dataRad = prom, modeRad = 'aguiar', keep.night = FALSE)

old <- setwd(tempdir())

writeSolar(prodFixed, 'prodFixed.csv')

dir()

zI <- fread("prodFixed.csv",
            header = TRUE, sep = ",")
zI

zD <- fread("prodFixed.D.csv",
            header = TRUE, sep = ",")
zD

zM <- fread("prodFixed.M.csv",
            header = TRUE, sep = ",")
zM

zY <- fread("prodFixed.Y.csv",
            header = TRUE, sep = ",")
zY

setwd(old)
```

Description

Methods for function xyplot in Package 'solaR'

Methods

`signature(x = "data.table", data = "missing")`: This method creates an XY plot for objects of class `data.table` without specifying a data argument. It must contain a column named `Dates` with the time information.

`signature(x = "formula", data = "Meteo")`: The `Meteo` object is converted into a `data.table` object with `getData(x)` and displayed with the method for `data.table`.

`signature(x = "formula", data = "Sol")`: The `Sol` object is converted into a `data.table` object with `as.data.tableI(x, complete = TRUE, day = TRUE)` and displayed with the method for `data.table`.

`signature(x = "formula", data = "G0")`: Idem.

`signature(x = "Meteo", data = "missing")`: The `Meteo` object is converted into a `data.table` object with `getData(data)`. This `data.table` is the `x` argument for a call to `xyplot`, using the `S4` method for `signature(x = "data.table", data = "missing")`.

`signature(x = "G0", data = "missing")`: The `G0` object is converted into a `data.table` object with `indexD(data)`. This `data.table` is the `x` argument for a call to `xyplot`, using the `S4` method for `signature(x = 'data.table', data = 'missing')`.

`signature(x = "ProdGCPV", data = "missing")`: Idem, but the variables are not superposed.

`signature(x = "ProdPVPS", data = "missing")`: Idem.

`signature(x = "formula", data = "Shade")`: Idem.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

E_aguiar

Markov Transition Matrices for the Aguiar et al. procedure

Description

Markov Transition Matrices and auxiliary data for generating sequences of daily radiation values.

Usage

`data(MTM)`

Format

`MTM` is a `data.frame` with the collection of Markov Transition Matrices defined in the paper "Simple procedure for generating sequences of daily radiation values using a library of Markov transition matrices", Aguiar et al., Solar Energy, 1998. `Ktlim` (matrix) and `Ktmtm` (vector) are auxiliary data to choose the correspondent matrix of the collection.

Author(s)

Oscar Perpiñán Lamigueiro, Francisco Delgado López.

E_helios	<i>Daily irradiation and ambient temperature from the Helios-IES database</i>
----------	---

Description

A year of irradiation, maximum and minimum ambient temperature from the HELIOS-IES database.

Usage

```
data(helios)
```

Format

A data frame with 355 observations on the following 4 variables:

yyyy.mm.dd a factor: year, month and day.

G.0. a numeric vector, daily global horizontal irradiation.

TambMax a numeric vector, maximum ambient temperature.

TambMin a numeric vector, minimum ambient temperature.

Source

<http://helios.ies-def.upm.es/consulta.aspx>

E_prodEx	<i>Productivity of a set of PV systems of a PV plant.</i>
----------	---

Description

A data.table object with the time evolution of the final productivity of a set of 22 systems of a large PV plant.

Usage

```
data(prodEx)
```

References

O. Perpiñán, Statistical analysis of the performance and simulation of a two-axis tracking PV system, Solar Energy, 83:11(2074–2085), 2009.https://oa.upm.es/1843/1/PERPINAN_ART2009_01.pdf

E_pumpCoef

*Coefficients of centrifugal pumps.***Description**

Coefficients of centrifugal pumps

Usage

data(pumpCoef)

Format

A data.table with 13 columns:

Qn rated flux**stages** number of stages**Qmax** maximum flux**Pmn** rated motor power**a, b, c** Coefficients of the equation $H = a \cdot f^2 + b \cdot f \cdot Q + c \cdot Q^2$.**g, h, i** Coefficients of the efficiency curve of the motor (50 Hz): $\eta_m = g \cdot (\%P_{mn})^2 + h \cdot (\%P_{mn}) + i$.**j, k, l** Coefficients of the efficiency curve of the pump (50 Hz): $\eta_b = j \cdot Q^2 + k \cdot Q + l$.**Details**

With this version only pumps from the manufacturer Grundfos are included.

Source<https://product-selection.grundfos.com/>**References**

- Perpiñán, O, Energía Solar Fotovoltaica, 2015. (<https://oscarperpinan.github.io/esf/>)
- Perpiñán, O. (2012), "solaR: Solar Radiation and Photovoltaic Systems with R", Journal of Statistical Software, 50(9), 1-32, [doi:10.18637/jss.v050.i09](https://doi.org/10.18637/jss.v050.i09)

E_SIAR

Data on the stations that make up the SIAR network

Description

Information about the location and operational status of the stations that make up the SIAR network

Usage

data(SIAR)

Format

est_SIAR is a data.table with 625 estations containing the following information:

Estacion character, name of the station.

Codigo character, code of the station.

Longitud numeric, longitude of the station in degrees (negative is for locations in the west).

Latitud numeric, latitud of the station in degrees.

Altitud integer, altitude of the station in meters.

Fecha_Instalacion Date, day the station was installed, and therefore, the start of its records.

Fecha_Baja Date, day the station was decommissioned, and therefore, the end of its records (if its value is NA, it means it is still operational).

Source

<https://servicio.mapa.gob.es/websiar/>

E_solaR.theme

solaR theme

Description

A customized theme for lattice. It is based on the custom.theme.2 function of the latticeExtra package with the next values:

- pch = 19
- cex = 0.7
- region = rev(brewer.pal(9, 'Yl0rRd'))
- strip.background\$col = 'lightgray'
- strip.shingle\$col = 'transparent'

Index

* classes

- B1_Meteo-class, [30](#)
- B2_Sol-class, [31](#)
- B3_G0-class, [32](#)
- B4_Gef-class, [33](#)
- B5_ProdGCPV-class, [35](#)
- B6_ProdPVPS-class, [36](#)
- B7_Shade-class, [38](#)

* constructors

- A1_calcSol, [5](#)
- A2_calcG0, [7](#)
- A3_calcGef, [10](#)
- A4_prodGCPV, [12](#)
- A5_prodPVPS, [17](#)
- A6_calcShd, [19](#)
- A7_optimShd, [20](#)
- A8_readBD, [26](#)
- A8_readG0dm, [28](#)
- A8_readSIAR, [29](#)

* datasets

- E_aguiar, [87](#)
- E_helios, [88](#)
- E_prodEx, [88](#)
- E_pumpCoef, [89](#)
- E_SIAR, [90](#)
- E_solaR.theme, [90](#)

* methods

- D_as.data.tableD-methods, [72](#)
- D_as.data.tableI-methods, [73](#)
- D_as.data.tableM-methods, [74](#)
- D_as.data.tableY-methods, [76](#)
- D_compare-methods, [77](#)
- D_getData-methods, [78](#)
- D_getG0-methods, [78](#)
- D_getLat-methods, [79](#)
- D_indexD-methods, [79](#)
- D_indexI-methods, [80](#)
- D_levelplot-methods, [80](#)
- D_Losses-methods, [81](#)

- D_mergesolaR-methods, [82](#)
- D_shadeplot-methods, [83](#)
- D_window-methods, [83](#)
- D_writeSolar-methods, [85](#)
- D_xyplot-methods, [86](#)

* utilities

- C_corrFdKt, [39](#)
- C_fBTd, [41](#)
- C_fBTi, [43](#)
- C_fCompD, [44](#)
- C_fCompI, [45](#)
- C_fInclin, [47](#)
- C_fProd, [49](#)
- C_fPump, [52](#)
- C_fSolD, [53](#)
- C_fSolI, [55](#)
- C_fSombra, [57](#)
- C_fTemp, [60](#)
- C_fTheta, [61](#)
- C_HQCurve, [62](#)
- C_local2Solar, [63](#)
- C_NmgPVPS, [65](#)
- C_sample2Diff, [66](#)
- C_solarAngles, [68](#)
- C_utils-angle, [70](#)
- C_utils-time, [71](#)

- [,G0,ANY,ANY-method (D_window-methods), [83](#)

- [,G0-method (D_window-methods), [83](#)

- [,Gef,ANY,ANY-method (D_window-methods), [83](#)

- [,Gef-method (D_window-methods), [83](#)

- [,Meteo,ANY,ANY-method (D_window-methods), [83](#)

- [,Meteo-method (D_window-methods), [83](#)

- [,ProdGCPV,ANY,ANY-method (D_window-methods), [83](#)

- [,ProdGCPV-method (D_window-methods), [83](#)

- [,ProdPVPS,ANY,ANY-method

- (D_window-methods), 83
- [,ProdPVPS-method (D_window-methods), 83
- [,Sol,ANY,ANY-method
 - (D_window-methods), 83
- [,Sol-method (D_window-methods), 83

- A1_calcSol, 5
- A2_calcG0, 7
- A3_calcGef, 10
- A4_prodGCPV, 12
- A5_prodPVPS, 17
- A6_calcShd, 19
- A7_optimShd, 20
- A8_Meteo2Meteo, 24
- A8_readBD, 26
- A8_readG0dm, 28
- A8_readSIAR, 29
- aguiar (E_aguiar), 87
- as.data.frame,Shade-method
 - (B7_Shade-class), 38
- as.data.tableD, 86
- as.data.tableD
 - (D_as.data.tableD-methods), 72
- as.data.tableD,G0-method
 - (D_as.data.tableD-methods), 72
- as.data.tableD,Gef-method
 - (D_as.data.tableD-methods), 72
- as.data.tableD,ProdGCPV-method
 - (D_as.data.tableD-methods), 72
- as.data.tableD,ProdPVPS-method
 - (D_as.data.tableD-methods), 72
- as.data.tableD,Sol-method
 - (D_as.data.tableD-methods), 72
- as.data.tableD-methods
 - (D_as.data.tableD-methods), 72
- as.data.tableI, 86
- as.data.tableI
 - (D_as.data.tableI-methods), 73
- as.data.tableI,G0-method
 - (D_as.data.tableI-methods), 73
- as.data.tableI,Gef-method
 - (D_as.data.tableI-methods), 73
- as.data.tableI,ProdGCPV-method
 - (D_as.data.tableI-methods), 73
- as.data.tableI,ProdPVPS-method
 - (D_as.data.tableI-methods), 73
- as.data.tableI,Sol-method
 - (D_as.data.tableI-methods), 73

- as.data.tableI-methods
 - (D_as.data.tableI-methods), 73
- as.data.tableM, 86
- as.data.tableM
 - (D_as.data.tableM-methods), 74
- as.data.tableM,G0-method
 - (D_as.data.tableM-methods), 74
- as.data.tableM,Gef-method
 - (D_as.data.tableM-methods), 74
- as.data.tableM,ProdGCPV-method
 - (D_as.data.tableM-methods), 74
- as.data.tableM,ProdPVPS-method
 - (D_as.data.tableM-methods), 74
- as.data.tableM-methods
 - (D_as.data.tableM-methods), 74
- as.data.tableY, 86
- as.data.tableY
 - (D_as.data.tableY-methods), 76
- as.data.tableY,G0-method
 - (D_as.data.tableY-methods), 76
- as.data.tableY,Gef-method
 - (D_as.data.tableY-methods), 76
- as.data.tableY,ProdGCPV-method
 - (D_as.data.tableY-methods), 76
- as.data.tableY,ProdPVPS-method
 - (D_as.data.tableY-methods), 76
- as.data.tableY-methods
 - (D_as.data.tableY-methods), 76
- as.POSIXct, 42
- azimuth (C_solarAngles), 68

- B1_Meteo-class, 30
- B2_Sol-class, 31
- B3_G0-class, 32
- B4_Gef-class, 33
- B5_ProdGCPV-class, 35
- B6_ProdPVPS-class, 36
- B7_Shade-class, 38
- bo0d (C_solarAngles), 68

- C_corrFdKt, 39
- C_fBTd, 41
- C_fBTi, 43
- C_fCompD, 44
- C_fCompI, 45
- C_fInclin, 47
- C_fProd, 49
- C_fPump, 52
- C_fSold, 53

- C_fSolI, 55
- C_fSombra, 57
- C_fTemp, 60
- C_fTheta, 61
- C_HQCurve, 62
- C_local2Solar, 63
- C_NmgPVPS, 65
- C_sample2Diff, 66
- C_solarAngles, 68
- C_utils-angle, 70
- C_utils-time, 71
- calcG0, 10, 11, 13, 14, 17, 18, 20, 21, 26, 32, 48, 56
- calcG0 (A2_calcG0), 7
- calcGef, 13, 14, 17–19, 21, 33, 48, 49, 61, 62
- calcGef (A3_calcGef), 10
- calcShd, 10, 11, 14, 20, 23, 59
- calcShd (A6_calcShd), 19
- calcSol, 7–11, 13, 17, 18, 21, 31, 40, 44, 46, 47, 59–61, 69
- calcSol (A1_calcSol), 5
- char2diff (C_sample2Diff), 66
- compare, 14
- compare (D_compare-methods), 77
- compare, G0-method (D_compare-methods), 77
- compare, Gef-method (D_compare-methods), 77
- compare, ProdGCPV-method (D_compare-methods), 77
- compare-methods (D_compare-methods), 77
- compareLosses, 14
- compareLosses (D_Losses-methods), 81
- compareLosses, ProdGCPV-method (D_Losses-methods), 81
- compareLosses-methods (D_Losses-methods), 81
- corrFdKt, 8, 9, 44, 46, 47
- corrFdKt (C_corrFdKt), 39
- d2h (C_utils-angle), 70
- d2r (C_utils-angle), 70
- D_as.data.tableD-methods, 72
- D_as.data.tableI-methods, 73
- D_as.data.tableM-methods, 74
- D_as.data.tableY-methods, 76
- D_compare-methods, 77
- D_getData-methods, 78
- D_getG0-methods, 78
- D_getLat-methods, 79
- D_indexD-methods, 79
- D_indexI-methods, 80
- D_levelplot-methods, 80
- D_Losses-methods, 81
- D_mergesolaR-methods, 82
- D_shadeplot-methods, 83
- D_window-methods, 83
- D_writeSolar-methods, 85
- D_xyplot-methods, 86
- DateTimeClasses, 71
- declination (C_solarAngles), 68
- diff2Hours (C_sample2Diff), 66
- dom (C_utils-time), 71
- dotplot, 77, 81
- doy (C_utils-time), 71
- dst (C_utils-time), 71
- dt2Meteo, 7–9, 30, 46
- dt2Meteo (A8_readBD), 26
- E_aguiar, 87
- E_helios, 88
- E_prodEx, 88
- E_pumpCoef, 89
- E_SIAR, 90
- E_solaR.theme, 90
- eccentricity (C_solarAngles), 68
- eot, 64
- eot (C_solarAngles), 68
- est_SIAR (E_SIAR), 90
- fBTd, 6, 7, 54, 69
- fBTd (C_fBTd), 41
- fBTi, 6, 69
- fBTi (C_fBTi), 43
- fCompD, 7–9, 32, 39, 40, 47
- fCompD (C_fCompD), 44
- fCompI, 7–9, 32, 39, 40, 45, 49
- fCompI (C_fCompI), 45
- FdKtBRL (C_corrFdKt), 39
- FdKtCLIMEd (C_corrFdKt), 39
- FdKtCLIMEdh, 46
- FdKtCLIMEdh (C_corrFdKt), 39
- FdKtCPR, 8, 44
- FdKtCPR (C_corrFdKt), 39
- FdKtEKDd (C_corrFdKt), 39
- FdKtEKDh (C_corrFdKt), 39
- FdKtLJ (C_corrFdKt), 39
- FdKtPage, 8, 44

- FdKtPage (C_corrFdKt), 39
- fInclin, 10, 11, 14, 20, 33, 51, 62, 81
- fInclin (C_fInclin), 47
- fProd, 14, 35, 81
- fProd (C_fProd), 49
- fPump, 18, 38, 66
- fPump (C_fPump), 52
- fread, 26, 27, 86
- fSolD, 5, 8, 10, 13, 17, 21, 31, 42, 44, 56, 69
- fSolD (C_fSolD), 53
- fSolI, 5, 6, 8, 10, 13, 17, 21, 31, 46, 47, 69
- fSolI (C_fSolI), 55
- fSombra, 19, 58, 62
- fSombra (C_fSombra), 57
- fSombra2X (C_fSombra), 57
- fSombra6, 19, 22, 58
- fSombra6 (C_fSombra), 57
- fSombraEst, 58
- fSombraEst (C_fSombra), 57
- fSombraHoriz, 58
- fSombraHoriz (C_fSombra), 57
- fTemp, 7, 26, 51
- fTemp (C_fTemp), 60
- fTheta, 10, 11, 14, 20, 34, 48, 49, 59
- fTheta (C_fTheta), 61

- G0, 31, 34–38
- G0-class (B3_G0-class), 32
- Gef, 19, 31, 33, 35–39, 49
- Gef-class (B4_Gef-class), 33
- getData (D_getData-methods), 78
- getData, Meteo-method (D_getData-methods), 78
- getData-methods (D_getData-methods), 78
- getG0 (D_getG0-methods), 78
- getG0, Meteo-method (D_getG0-methods), 78
- getG0-methods (D_getG0-methods), 78
- getLat (D_getLat-methods), 79
- getLat, G0-method (D_getLat-methods), 79
- getLat, Meteo-method (D_getLat-methods), 79
- getLat, Sol-method (D_getLat-methods), 79
- getLat-methods (D_getLat-methods), 79

- h2d (C_utils-angle), 70
- h2r (C_utils-angle), 70
- helios (E_helios), 88
- hms (C_utils-time), 71
- HQCurve (C_HQCurve), 62

- indexD, 83, 84
- indexD (D_indexD-methods), 79
- indexD, G0-method (D_indexD-methods), 79
- indexD, Meteo-method (D_indexD-methods), 79
- indexD, Sol-method (D_indexD-methods), 79
- indexD-methods (D_indexD-methods), 79
- indexI (D_indexI-methods), 80
- indexI, Sol-method (D_indexI-methods), 80
- indexI-methods (D_indexI-methods), 80

- Ktd (C_corrFdKt), 39
- Kti (C_corrFdKt), 39
- Ktlim (E_aguiar), 87
- Ktm (C_corrFdKt), 39
- Ktmtm (E_aguiar), 87

- levelplot, formula, G0-method (D_levelplot-methods), 80
- levelplot, formula, Meteo-method (D_levelplot-methods), 80
- levelplot, formula, Sol-method (D_levelplot-methods), 80
- levelplot, formula, zoo-method (D_levelplot-methods), 80
- levelplot-methods (D_levelplot-methods), 80
- local2Solar (C_local2Solar), 63
- lonHH (C_local2Solar), 63
- losses (D_Losses-methods), 81
- losses, Gef-method (D_Losses-methods), 81
- losses, ProdGCPV-method (D_Losses-methods), 81
- losses-methods (D_Losses-methods), 81

- mergesolaR, 14
- mergesolaR (D_mergesolaR-methods), 82
- mergesolaR, G0-method (D_mergesolaR-methods), 82
- mergesolaR, Gef-method (D_mergesolaR-methods), 82
- mergesolaR, Meteo-method (D_mergesolaR-methods), 82
- mergesolaR, ProdGCPV-method (D_mergesolaR-methods), 82
- mergesolaR, ProdPVPS-method (D_mergesolaR-methods), 82
- mergesolaR-methods (D_mergesolaR-methods), 82

- Meteo, [32](#), [34](#), [35](#), [37](#), [38](#), [40](#), [60](#)
- Meteo-class (B1_Meteo-class), [30](#)
- Meteod2Meteom (A8_Meteo2Meteo), [24](#)
- Meteoi2Meteod (A8_Meteo2Meteo), [24](#)
- MTM (E_aguiar), [87](#)

- NmgPVPS, [18](#), [52](#), [63](#)
- NmgPVPS (C_NmgPVPS), [65](#)

- optimShd, [38](#), [59](#)
- optimShd (A7_optimShd), [20](#)

- P2E (C_sample2Diff), [66](#)
- prodEx (E_prodEx), [88](#)
- ProdGCPV, [38](#), [39](#)
- prodGCPV, [23](#), [35](#), [49](#), [51](#)
- prodGCPV (A4_prodGCPV), [12](#)
- ProdGCPV-class (B5_ProdGCPV-class), [35](#)
- ProdPVPS, [18](#)
- prodPVPS, [36](#), [38](#), [52](#), [63](#), [66](#)
- prodPVPS (A5_prodPVPS), [17](#)
- ProdPVPS-class (B6_ProdPVPS-class), [36](#)
- pumpCoef, [18](#), [37](#), [52](#), [62](#), [63](#), [65](#), [66](#)
- pumpCoef (E_pumpCoef), [89](#)

- r2d (C_utils-angle), [70](#)
- r2h (C_utils-angle), [70](#)
- r2sec (C_utils-angle), [70](#)
- readBDd, [7-9](#), [25](#), [28-30](#), [40](#), [44](#), [60](#), [61](#), [79](#)
- readBDd (A8_readBD), [26](#)
- readBDi, [7-9](#), [30](#), [40](#), [46](#)
- readBDi (A8_readBD), [26](#)
- readG0dm, [7-9](#), [25](#), [27](#), [29](#), [30](#), [40](#), [44](#), [79](#)
- readG0dm (A8_readG0dm), [28](#)
- readSIAR, [25](#)
- readSIAR (A8_readSIAR), [29](#)

- sample2Hours (C_sample2Diff), [66](#)
- seq.POSIXt, [6](#), [8](#), [21](#), [42](#), [56](#), [67](#)
- Shade, [23](#), [36](#), [83](#)
- Shade-class (B7_Shade-class), [38](#)
- shadeplot (D_shadeplot-methods), [83](#)
- shadeplot, Shade-method (D_shadeplot-methods), [83](#)
- shadeplot-methods (D_shadeplot-methods), [83](#)
- show, G0-method (B3_G0-class), [32](#)
- show, Gef-method (B4_Gef-class), [33](#)
- show, Meteo-method (B1_Meteo-class), [30](#)
- show, ProdGCPV-method (B5_ProdGCPV-class), [35](#)
- show, ProdPVPS-method (B6_ProdPVPS-class), [36](#)
- show, Shade-method (B7_Shade-class), [38](#)
- show, Sol-method (B2_Sol-class), [31](#)
- Sol, [32-38](#), [40](#), [60](#), [67](#)
- Sol-class (B2_Sol-class), [31](#)
- solaR.theme (E_solaR.theme), [90](#)
- solaR2 (solaR2-package), [3](#)
- solaR2-package, [3](#)
- sunHour (C_solarAngles), [68](#)
- sunrise (C_solarAngles), [68](#)

- truncDay (C_utils-time), [71](#)

- window (D_window-methods), [83](#)
- window-methods (D_window-methods), [83](#)
- write.table, [85](#), [86](#)
- writeSolar (D_writeSolar-methods), [85](#)
- writeSolar, Sol-method (D_writeSolar-methods), [85](#)
- writeSolar-methods (D_writeSolar-methods), [85](#)

- xyplot, data.table, missing-method (D_xyplot-methods), [86](#)
- xyplot, formula, G0-method (D_xyplot-methods), [86](#)
- xyplot, formula, Meteo-method (D_xyplot-methods), [86](#)
- xyplot, formula, Shade-method (D_xyplot-methods), [86](#)
- xyplot, formula, Sol-method (D_xyplot-methods), [86](#)
- xyplot, G0, missing-method (D_xyplot-methods), [86](#)
- xyplot, Meteo, missing-method (D_xyplot-methods), [86](#)
- xyplot, ProdGCPV, missing-method (D_xyplot-methods), [86](#)
- xyplot, ProdPVPS, missing-method (D_xyplot-methods), [86](#)
- xyplot-methods (D_xyplot-methods), [86](#)

- zenith (C_solarAngles), [68](#)
- zoo2Meteo, [7](#), [8](#), [30](#), [46](#)
- zoo2Meteo (A8_readBD), [26](#)