UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Eléctrica

TRABAJO DE FIN DE GRADO

Desarrollo de una herramienta software para la simulación de sistemas fotovoltaicos con R

Autor: Francisco Delgado López

Tutor: Oscar Perpiñán Lamigueiro

Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física aplicada

Madrid, 5 de septiembre de 2024

Agradecimientos

Agradezco a …

Resumen

El presente proyecto se enfoca en el desarrollo de un paquete de software estadístico en R, denominado solaR2, diseñado para estimar la productividad de sistemas fotovoltaicos a partir de datos de irradiación solar. solaR2 es una evolución del paquete solaR, con mejoras significativas en modularidad y eficiencia. A diferencia de solaR, que utilizaba el paquete zoo para la gestión de series temporales, solaR2 se basa en data.table, lo que optimiza la manipulación de grandes volúmenes de datos y acelera el procesamiento. Este avance es crucial para un análisis más detallado y eficiente en el campo de la energía solar fotovoltaica.

solaR2 ofrece herramientas avanzadas para simular el rendimiento de sistemas conectados a la red y sistemas de bombeo de agua con energía solar. Incluye clases, métodos y funciones que permiten calcular la geometría solar y la radiación solar incidente, así como estimar la productividad final de estos sistemas a partir de la irradiación global horizontal diaria e intradía.

El diseño modular basado en clases S4 facilita el manejo de series temporales multivariantes y proporciona métodos de visualización avanzados para el análisis de rendimiento en plantas fotovoltaicas a gran escala. La implementación con data.table mejora la eficiencia en la manipulación de datos, permitiendo análisis más rápidos y precisos. Entre sus funcionalidades destacadas están el cálculo de radiación solar en diferentes planos, la estimación de rendimiento de sistemas fotovoltaicos y de bombeo, y la evaluación de sombras.

Además, solaR2 ofrece herramientas avanzadas para la visualización estadística del rendimiento, compatible con otros paquetes de R para manipulación de series temporales y análisis espacial. Esto la convierte en una herramienta útil para investigadores y profesionales en el diseño y optimización de sistemas fotovoltaicos, permitiendo un análisis detallado bajo diversas condiciones. En resumen, solaR2 representa una mejora significativa en el análisis y simulación de sistemas solares, proporcionando una herramienta flexible y reproducible para mejorar la eficiencia energética y la rentabilidad de las instalaciones solares.

Palabras clave: geometría solar, radiación solar, energía solar, fotovoltaica, métodos de visualización, series temporales, datos espacio-temporales, S4

Abstract

This project focuses on the development of a statistical software package in R, called solaR2, designed to estimate the productivity of photovoltaic systems based on solar irradiation data. solaR2 represents an evolution of the existing solaR package, featuring significant improvements in modularity and efficiency. Unlike solaR, which relied on the zoo package for time series management, solaR2 uses data.table, optimizing the handling of large data volumes and speeding up processing. This advancement is crucial for more detailed and efficient analysis in the field of photovoltaic solar energy.

solaR2 provides advanced tools for simulating the performance of grid-connected systems and solar-powered water pumping systems. It includes classes, methods, and functions for calculating solar geometry and incident solar radiation, as well as estimating the final productivity of these systems from global horizontal irradiation on a daily and intraday basis.

The modular design based on S4 classes facilitates the management of multivariate time series and provides advanced visualization methods for performance analysis in large-scale photovoltaic plants. The use of data.table enhances data handling efficiency, allowing for faster and more precise analyses. Key functionalities include calculating solar radiation on different planes, estimating the performance of photovoltaic and pumping systems, and evaluating shading.

Additionally, solaR2 offers advanced statistical visualization tools, compatible with other R packages for time series manipulation and spatial analysis. This makes it a valuable tool for researchers and professionals involved in the design and optimization of photovoltaic systems, enabling detailed analysis under various conditions. In summary, solaR2 represents a significant improvement in the analysis and simulation of solar systems, providing a flexible and reproducible tool to enhance energy efficiency and the profitability of solar installations.

Keywords: solar geometry, solar radiation, solar energy, photovoltaic, visualization methods, time series, spatiotemporal data, S4

Índice general

Índice de figuras

Nomenclatura

A_c	Área de una célula
A_G	Área de un generador fotovoltáico
α	Ángulo de orientación de un sistema fotovoltaico
AM	Masa de aire
AO	Adelanto oficial durante el horario de verano
$B_0(0)$	irradiancia extra-atmósferica o extra-terrestre en el plano horizontal
B_0	Constante solar o irradiancia solar incidente en un plano normal al vector solar en el límite superior de la atmósfera terrestre
B	Radiación directa
β	Ángulo de inclinación de un sistema fotovoltaico
D	Radiación difusa
D^C	Radiación difusa circunsolar
δ	Declinación
$\Delta\lambda$	Diferencia entre la longitud local y la longitud del huso horario
D^I	Radiación difusa isotrópica
d_n	Día del año
d_{min}	Distancia mínima entre hileras de un generador para evitar el sombreado
EoT	Ecuación del tiempo
ϵ_0	Corrección debida a la excentricidad de la elipse de la trayectoria terrestre alrededor del sol
η_{mp}	Eficiencia de una motobomba
F_D	Fracción de difusa
F_{Dd}	Fracción de difusa diaria
F_{Dm}	Fracción de difusa mensual
FT_B	Factor de pérdidas angulares para irradiancia directa
FT_R	Factor de pérdidas angulares para irradiancia de albedo

FT_D	Factor de pérdidas angulares para irradiancia difusa
g	Aceleración de la gravedad
G	Radiación global
GCR	Ground coverage ratio
G_{STC}	Irradiancia incidente en condiciones estandar de medida
H_{dt}	Nivel dinámico de un pozo
H_f	Altura asociada a las pérdidas de frición en una tubería
H_{OT}	Diferencia de cotas entre la salida de agua y la entrada en el depósito
H_{st}	Nivel estático de un pozo
H_T	Altura total incluyendo las pérdidas de fricción de la tubería
H_{TE}	Altura total equivalente de un sistema de bombeo
H_v	Altura vertical aparente
I_{mpp}	Corriente de una célula en el punto de máxima potencia
I_{sc}	Corriente de cortocircuito de una célula
K_T	Índice de claridad
K_{Td}	Índice de claridad diario
K_{Tm}	Índice de claridad mensual
L_{eo}	Separación entre seguidores en sentido Este-Oeste
L_{ns}	Separación entre seguidores en sentido Norte-Sur
MPP	Punto de máxima potencia de un dispositivo fotovoltaico
ω	Hora solar o tiempo solar verdadero
ω_s	Ángulo del amanecer
P_{el}	Potencia eléctrica necesaria en la entrada de una motobomba
P_f	Pérdidas de frición en la tubería de un sistema de bombeo
P_H	Potencia hidraúlica necesaria en un sistema de bombeo de agua
P_H	Potencia hidráulica
ϕ	Latitud
P_{inv}	Potencia nominal de un inversor
Q	Caudal de agua
Q_{max}	caudal máximo del pozo
Q_t	Caudal de ensayo de un pozo

R	Radiación del albedo
r_D	Relación entre la irradiancia y la irradiación difusa en el plano horizontal
ρ	Densidad del agua
ρ	Coeficiente de reflexión del terreno para la irradiancia de albedo
ROT	Ratio de ocupación del terreno
STC	Condiciones estándar de medida de un dispositivo fotovoltaico
T_c^*	Temperatura de célula en condiciones estándar de medida
T_c	Temperatura de célula
θ_s	Ángulo de incidencia o ángulo entre el vector solar y el vector director de una superficie
θ_{zs}	Ángulo cenital solar
TO	Hora oficial
$TONC$	Temperatura de operación nominal de célula
V_{mpp}	Tensión de una célula en el punto de máxima potencia
V_{oc}	Tensión de circuito abierto de una célula

CAPÍTULO 1

Introducción

1.1. Objetivos

El objetivo principal de este proyecto es el desarrollo de un paquete en R [R C23] con el cual poder realizar estimaciones y representaciones gráficas de la geometría solar, radiación solar en el plano horizontal y del generador, y el funcionamiento de sistemas fotovoltaicos de conexión a red y de bombeo de agua.

Durante el resto del documento, si fuera necesario, se hará referencia al paquete desarrollado en este proyecto con el nombre solaR2 [CITAR SOLAR2].

El usuario puede colocar los datos que considere convenientes (desde una base de datos oficial, una base de datos propia... etc.) en cada una de las funciones que ofrece el paquete pudiendo así obtener resultados de la geometría solar, de la radiación horizontal, de la efectiva y hasta de la producción de diferentes tipos de sistemas fotovoltaicos.

El paquete también incluye una serie de funciones que permiten hacer representaciones gráficas de estos resultados con el fin de poder apreciar con más detalle las diferencias entre sistemas y contemplar cual es la mejor opción para el emplazamiento elegido.

Este proyecto toma su origen en el paquete ya existente `solaR` [Per12] el cual desarrolló el tutor de este proyecto en 2010. Esta versión, la 0.14, ha tenido una serie de actualizaciones, siendo la más reciente la 0.46 (en el 2021). Sin embargo, al ser versiones de un software antiguo se propuso la idea de renovarlo teniendo en cuenta el paquete en el que basa su funcionamiento. El paquete `solaR` basó su funcionamiento en el paquete `zoo` [ZG05] el cual proporciona una sólida base para trabajar con series temporales. Sin embargo, como base de `solaR2` se optó por el paquete `data.table` [Bar+24]. Este paquete ofrece una extensión de los clásicos `data.frame` de R en los `data.table`, los cuales pueden trabajar rápidamente con enormes cantidades de datos (por ejemplo, 100 GB de RAM).

La clave de ambos proyectos es que al estar basados en R, cualquier usuario puede acceder a ellos de forma gratuita, tan solo necesitas tener instalado R en tu dispositivo.

Para alojar este proyecto se toman dos vías:

- **`Github`** [Wan+23]: Donde se aloja la versión de desarrollo del paquete.

- **CRAN**: Acrónimo de Comprehensive R Archive Network, es el repositorio donde se alojan las versiones definitivas de los paquetes y desde el cual se descargan a la sesión de R.

El paquete **solaR2** permite realizar las siguientes operaciones:

- Cálculo de toda la geometría que caracteriza a la radiación procedente del Sol (**??**).
- Tratamiento de datos meteorológicos (en especial de radiación), procedentes de datos ofrecidos del usuario y de la red de estaciones SIAR [Min23] (**??**).
- Una vez calculado lo anterior, se pueden hacer estimaciones de:
  - Los componentes de radiación horizontal (**??**).
  - Los componentes de radiación eficaz en el plano inclinado (**??**).
  - La producción de sistemas fotovoltaicos conectados a red (**??**) y sistemas fotovoltaivos de bombeo (**??**).

Este proyecto ha tenido a su vez una serie de objetivos secundarios:

- Uso y manejo de GNU Emacs [Sta85] en el que se realizaron todos los archivos que componen este documento (utilizando el modo Org [Dom+03]) y el paquete descrito (empleando ESS [Pro24])
- Dominio de diferentes paquetes de R:
  - **zoo** [ZG05]: Paquete que proporciona un conjunto de clases y métodos en S3 para trabajar con series temporales regulares e irregulares. Usado en el paquete **solaR** como pilar central.
  - **data.table** [Bar+24]: Otorga una extensión a los datos de tipo data.frame que permite una alta eficiencia especialmente con conjuntos de datos muy grandes. Se ha utilizado en el paquete **solaR2** en sustitución del paquete **zoo** como tipo de dato principal en el cual se construyen las clases y métodos de este paquete.
  - **microbenchmark** [Mer+23]: Proporciona infraestructura para medir y comparar con precisión el tiempo de ejecución de expresiones en R. Usado para comparar los tiempos de ejecución de ambos paquetes.
  - **profvis** [Wic+24]: Crea una interfaz gráfica donde explorar los datos de rendimiento de una expresión dada. Aplicada junto con **microbenchmark** para detectar y corregir cuellos de botella en el paquete **solaR2**
  - **lattice** [Sar08]: Proporciona diversas funciones con las que representar datos. El paquete **solaR2** utiliza este paquete para representar de forma visual los datos obtenidos en las estimaciones.
- Junto con el modo Org, se ha utilizado el prepador de textos LaTeX (partiendo de un archivo .org, se puede exportar a un archivo .tex para posteriormente exportar un pdf).
- Obtener conocimientos teóricos acerca de la radiación solar y de la producción de energía solar mediante sistemas fotovoltaicos y sus diversos tipos. Para ello se ha usado en mayor medida el libro “Energía Solar Fotovoltaica” [Per23].

1.2. Análisis previo de soluciones

Este proyecto, como ya se ha comentado, es el heredero del paquete **solaR** desarrollado por Oscar Perpiñán. La filosofía de ambos paquetes es la misma y los resultados que dan son muy similares. Sin embargo, lo que les diferencia es que **solaR2** es más modular, es decir, tiene muchas funciones autónomas que permiten realizar cálculos específicos, en especial de geometría y radiación, y, el paquete sobre el que construyen sus datos.

Mientras que **solaR** basa sus clases y métodos en el paquete **zoo**, **solaR2** en el paquete **data.table**. Los dos paquetes pueden trabajar con series temporales, pero, mientras que **zoo** es más eficaz trabajando con series temporales, **data.table** es más eficiente a la hora de trabajar con una cantidad grande de datos, lo cual a la hora de realizar estimaciones muy precisas es beneficioso.

Por otro lado, existen otras soluciones fuera de R:

1. **PVsyst - Photovoltaic Software** [PVS24] Este software es probablemente el más conocido dentro del ámbito del estudio y la estimación de instalaciones fotovoltaicas. Permite una gran personalización de todos los componentes de la instalación.

2. **SISIFO** [Sis24] Herramienta web diseñada por el **Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar de la Universidad Politécnica de Madrid**.

3. **PVGIS** [PVG24] Aplicación web desarrolada por el **European Commission Joint Research Center** desde 2001.

4. **System Advisor Model** [SAM24] Desarrollado por el **Laboratorio Nacional de Energías Renovables**, perteneciente al Departamento de energía del gobierno de EE.UU.

En el capitulo 5 se realizará un ejemplo práctico que compare los resultados entre **PVsyst**, **solaR** y **solaR2**

1.3. Aspectos técnicos

Las fuentes de un paquete de **R** están contenidas en un directorio que contiene al menos:

- Los ficheros **DESCRIPTION** y **NAMESPACE**
- Los subdirectorios:
  - **R**: código en ficheros .R
  - **man**: páginas de ayuda de las funciones, métodos y clases contenidas en el paquete.

Esta estructura puede ser generada con **package.skeleton**

1.3.1. DESCRIPTION

El fichero DESCRIPTION contiene la información básica:

```
Package: pkgname
Version: 0.5-1
Date: 2004-01-01
Title: My First Collection of Functions
Authors@R: c(person("Joe", "Developer", role = c("aut", "cre"),
                    email = "Joe.Developer@some.domain.net"),
             person("Pat", "Developer", role = "aut"),
             person("A.", "User", role = "ctb",
                    email = "A.User@whereever.net"))
Author: Joe Developer and Pat Developer, with contributions from A. User
Maintainer: Joe Developer <Joe.Developer@some.domain.net>
Depends: R (>= 1.8.0), nlme
Suggests: MASS
Description: A short (one paragraph) description of what
  the package does and why it may be useful.
License: GPL (>= 2)
URL: http://www.r-project.org, http://www.another.url
```

- Los campos **`Package`**, **`Version`**, **`License`**, **`Title`**, **`Autor`** y **`Maintainer`** son obligatorios.
- Si usa métodos **`S4`** debe incluir **`Depends: methods`**.

1.3.2. NAMESPACE

`R` usa un sistema de gestión de **espacio de nombres** que permite al autor del paquete especificar:

- Las **variables** del paquete que se **exportan** (y son, por tanto, accesibles a los usuarios).
- Las **variables** que se **importan** de otros paquetes.
- Las **clases y métodos `S3`** y **`S4`** que deben registrarse.

El **`NAMESPACE`** controla la estrategia de búsqueda de variables que utilizan las funciones del paquete:

- En primer lugar, busca entre las creadas localmente (por el código de la carpeta **`R/`**).
- En segundo lugar, busca entre las variables importadas explícitamente de otros paquetes.
- En tercer lugar, busca en el **`NAMESPACE`** del paquete **`base`**.
- Por último, busca siguiendo el camino habitual (usando **`search()`**).

```r
search()
```

```
[1] ".GlobalEnv"        "ESSR"              "package:stats"     "package:graphics"
[5] "package:grDevices" "package:utils"     "package:datasets"  "package:methods"
[9] "Autoloads"         "package:base"
```

Manejo de variables

- Exportar variables:

```
export(f, g)
```

- Importar **todas** las variables de un paquete:

```
import(pkgExt)
```

- Importar variables **concretas** de un paquete:

```
importFrom(pkgExt, var1, var2)
```

Manejo de clases y métodos

- Para registrar un **método** para una **clase** determinada:

```
S3method(print, myClass)
```

- Para usar clases y métodos `S4`:

```
import("methods")
```

- Para registrar clases `S4`:

```
exportClasses(class1, class2)
```

- Para registrar métodos `S4`:

```
exportMethods(method1, method2)
```

- Para importar métodos y clases `S4` de otro paquete:

```
importClassesFrom(package, ...)
importMethodsFrom(package, ...)
```

1.3.3. Documentación

Las páginas de ayuda de los objetos `R` se escriben usando el formato “R documentation” (Rd), un lenguaje similar a LaTeX.

```
\name{load}
\alias{load}
\title{Reload Saved Datasets}
\description{
  Reload the datasets written to a file with the function
  \code{save}.
}
\usage{
  load(file, envir = parent.frame())
}
\arguments{
\item{file}{a connection or a character string giving the
    name of the file to load.}
\item{envir}{the environment where the data should be
    loaded.}
}
\seealso{
  \code{\link{save}}.
}
\examples{
  ## save all data
  save(list = ls(), file= "all.RData")

  ## restore the saved values to the current environment
  load("all.RData")

  ## restore the saved values to the workspace
  load("all.RData", .GlobalEnv)
}
\keyword{file}
```

CAPÍTULO 2

Estado del arte

2.1. Situación actual de la generación fotovoltaica

Según el informe anual de 2023 de la UNEF[1] [UNE23] en 2022 la fotovoltaica se posicionó como la tecnología con más crecimiento a nivel internacional, tanto entre las renovables como entre las no renovables. Se instalaron 240 GWp de nueva capacidad fotovoltaica a nivel mundial, suponiendo esto un incremento del 137 % con respecto a 2021.

A pesar de las diversas crisis internacionales, la energía solar fotovoltaica alcanzó a superar los 1185 GWp instalados. Como otros años, las cifras indican que China continuó siendo el primer actor mundial, superando los 106 GWp de potencia instalada en el año. La Unión Europea se situó en el segundo puesto, duplicando la potencia instalada en 2021, y alcanzando un nuevo record con 41 GWp instalados en 2022.

La producción energía fotovoltaica a nivel mundial representó el 31 % de la capacidad de generación renovable, convirtiendose así en la segunda fuente de generación, solo por detrás de la energía hidráulica. En 2022 se añadió 3 veces más de energía solar que de energía eólica en todo el mundo.

Por otro lado, la Unión Europea superó a EE.UU. como el segundo mayor actor mundial en desarrollo fotovoltaico, instalando un 47 % más que en 2021 y alcanzando una potencia acumulada de más de 208 GWp. España lideró el mercado europeo con 8,6 GWp instalados en 2022, superando a Alemania.

El año 2022 fue significativo en términos legislativos con el lanzamiento del Plan REPowerEU[2] [Eur22]. Dentro de este plan, se lanzó la Estrategía de Energía Solar con el objetivo de alcanzar 400 GWp (320 GW) para 2030, incluyendo medidas para desarrollar tejados solares, impulsar la industria fotovoltaica y apoyar la formación de profesionales en el sector.

En 2022, España vivió un auge en el desarrollo fotovoltaico, instalando 5.641 MWp en plantas en suelo, un 30 % más que en 2021, y aumentando el autoconsumo en un 108 %, alcanzando 3.008 MWp. El sector industrial de autoconsumo creció notablemente, representando el 47 % del autoconsumo total.

[1] UNEF: Unión Española Fotovoltaica.

[2] Plan REPowerEU: Proyecto por el cual la Unión Europea quiere poner fin a su dependencia de los combustibles fósiles rusos ahorrando energía, diversificando los suministros y acelerando la transción hacia una energía limpia.

España implementó varias iniciativas legislativas para enfrentar la volatilidad de precios de la energía y la dependencia del gas, destacando el RD-ley 6/2022 [BOE22b] y el RD 10/2022 [BOE22a], que han modificado mecanismos de precios y estableciendo límites al precio del gas.

El Plan SE+[3] [dem22] incluye medidas fiscales y administrativas para apoyar las renovables y el autoconsumo. En 2022, se realizaron subastas de energía renovable, asignando 140 MW a solar fotovoltaica en la tercera subasta y 1.800MW en la cuarta, aunque esta última quedó desierta por precios de reserva bajos.

Se adjudicaron 1.200 MW del nudo de transición justa de Andorra a Enel Green Power España, con planes para instalar plantas de hidrógeno verde y agrovoltaica. la actividad en hidrógeno verde y almacenamiento también creció, con fondos adicionales y exenciones de cargos.

El autoconsumo, apoyado por diversas regulaciones y altos precios de la electricidad, registró un crecimiento significativo, alcanzado 2.504 MW de nueva potencia en 2022. Las comunidades energéticas también avanzaron gracias a ayudas específicas, a pesar de la falta de un marco regulatorio definido.

2022 estuvo marcado por los programas financiados por la Unión Europea, especialmente el Mecanismo de Recuperación y Resiliencia [Hac22] que canaliza los fondos NextGenerationEU [Uni20]. El PERTE[4], aprobado en diciembre de 2021, espera crear más de 280.000 empleos, con ayudas que se ejecutarán hasta 2026. En 2023 se solicitó a Bruselas una adenda para segunda fase del PERTE, obteniendo 2.700 millones de euros adicionales.

La contribución del sector fotovoltaico a la economía española en 2022 fue significativa, aportando 7.014 millones de euros al PIB[5], un 51 % más que el año anterior, y generando una huella econóimca total de 15.656 millones de euros. En términos de empleo, el sector involucró a 197.383 trabajadores, de los cuales 40.683 fueros directos, 97.600 indirectos y 59.100 inducidos.

El sector industrial fotovoltaico nacional tiene una fuerte presencia en España, con hasta un 65 % de los componenetes manufacturados localmente. Empresas españolas se encuentran entre los principales fabricantes mundiales de inversores y seguidores solares. Además, España es un importante exportador de estructuras fotovoltaicas y cuenta con iniciativas prometedoras para la fabricación de módulos solares.

En definitiva, la fotovoltaica es una tecnología en auge y con perspectivas para ser el pilar de la transición ecológica. Por ello, surge la necesidad de encontrar herramientas que permitan estimar el desempeño que estos sistemas pueden tener a la hora de realizar estudios de viabilidad económica.

2.2. Solución actual y sus carencias

Como se mencionó en el capitulo 1 este proyecto toma su base en el paquete **`solaR`** [Per12], el cual es una herramienta robusta para el cálculo de la radiación solar y el rendimiento de sistemas fotvoltaicos. Este paquete está diseñado utilizando clases `S4` en `R`, y su núcleo se basa en series temporales multivariantes almacenadas en objetos de la clase **`zoo`**. El paquete permite realizar investigaciones reproducibles sobre el rendimiento de sistemas fotovoltaicos y la radiación solar, proporcionando métodos para calcular la geometría solar, la radiación incidente sobre un

[3]Plan + Seguridad Energética: Se trata de un plan con medidas de rápido impacto dirigidas al invierno 2022/2023, junto con medidas que contribuyen a un refuerzo estructural de esa seguridad energética.

[4]PERTE: Proyecto Estratégico para la Recuperación y Transformación Económica.

[5]PIB: Producto Interior Bruto.

generador fotovoltaico, y simular el rendimiento de sistemas fotovoltaicos tanto conectados a la red como de bombeo de agua.

Pese a ser un herramienta muy capaz, `solaR` presenta una serie de carencias relativas al paquete `zoo`:

- **Eficiencia y rendimiento**: el paquete `solaR` utiliza `zoo` para manejar series temporales, lo cual es adecuado para volúmenes de datos moderados. Sin embargo, `zoo` no está optimizado para operaciones de alta eficiencia en datasets grandes. Por otro lado, `data.table` está diseñado específicamente para manejar grandes volúmenes de datos de manera eficiente, ofreciendo un rendimiento superior en operaciones de lectura, escritura y manipulación masiva de datos.
- **Escalabilidad**: `solaR` puede experimentar problemas de escalabilidad al trabajar con datasets extensos, ya que `zoo` no es tan eficiente en operaciones que requieren manipulación compleja o paralelización. Sin embargo, `data.table` supera esta limitación al proporcionar una infraestructura altamente optimizada para operaciones en paralelo y manejo de grandes conjuntos de datos, permitiendo que las aplicaciones escalen mejor en entornos de datos intensivos.
- **Manipulación de datos**: `zoo` es adecuado para manejar series temporales básicas, pero carece de las capacidades avanzadas de manipulación de datos que ofrece `data.table`, como la indexación rápida, las uniones eficientes, y la capacidad de realizar operaciones complejas de agrupamiento y agregación. Estas características de `data.table` permiten un manejo de datos más flexible y potente, lo cual es esencial en análisis de datos complejo y en tiempo real.
- **Consumo de memoria**: `zoo` puede consumir más memoria en comparación con `data.table` cuando se trabaja con grandes conjuntos de datos. Por otro lado, `data.table` está optimizado para operaciones en memoria, lo que permite manejar datasets más grandes sin requerir un incremento proporcionla en el uso de recursos, haciendo que las operaciones sean más sostenibles en términos de memoria.

Por lo tanto, al adoptar `data.table` en `solaR2`, se abordarían esta limitaciones, proporcionando un paquete más robusto y capaz de manejar los desafíos actuales en el análisis de datos de radiación solar y de producción de sistemas fotovoltaicos.

CAPÍTULO 3

Marco teórico

El paquete **solaR2** toma como marco teórico el libro de Oscar Perpiñán, tutor de este trabajo, Energía Solar Fotovoltaica [Per23] para cada una de las operaciones de cálculo que realizan cada una de las funciones. En la figura 3.1, se muestra un diagrama que resume los pasos que se siguen a la hora de calcular la producción de sistemas fotovoltaicos.

Estos pasos son:

1. Calcular la geometría que define la posición relativa del Sol desde la Tierra.
2. Obtener la irradiación global diaria en el plano horizontal
3. A partir de la irradiación global, obtener las componentes de difusa y directa.
4. Se trasladan estos valores de irradiación a valores de irradiancia.
5. Integrando estos valores se pueden obtener las estimaciones irradiación diaria difusa, directa y global
6. El generador fotovoltaico produce una potencia en corriente continua dependiente del rendimiento del mismo.
7. Se transforma en potencia en corriente alterna mediante un inversor que tiene una eficiencia asociada.
8. Integrando esta potencia se puede obtener la energía que produce el generador en un tiempo determinado.

3.1. Naturaleza de la radiación solar

Para el cálculo de la radiación solar que incide en una superficie se deben distinguir tres componentes diferenciados:

- **Radiación Directa**, B: fracción de radiación que procede en línea recta desde el Sol.
- **Radiación Difusa**, D: fracción de radiación que procede de todo el cielo, excepto del Sol. Son todos aquellos rayos que dispersa la atmósfera.

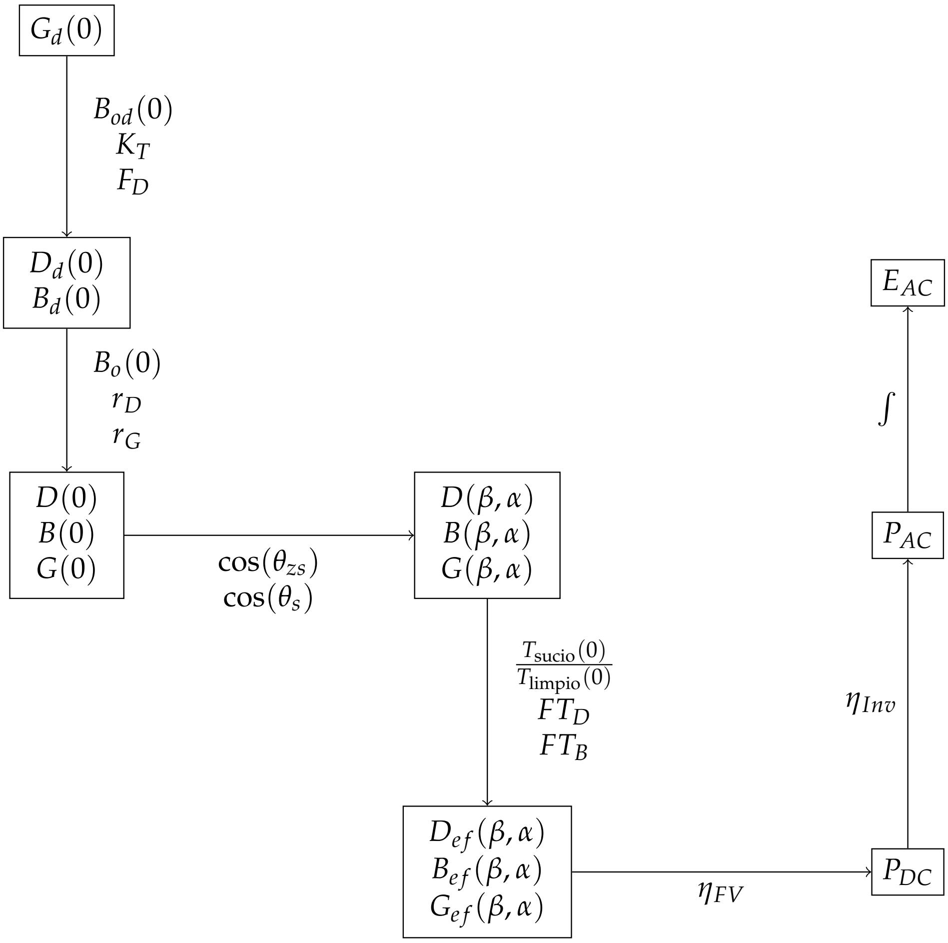


Figura 3.1: *Procedimiento de cálculo*

- **Radiación del albedo**, R: parte de la radiación procedente de la reflexión con el suelo.

La suma de las tres componentes constituye la denominada radiación global:

$$G = B + D + R \tag{3.1}$$

Tomando como base el libro antes mencionado [Per23], se describirá el proceso que se ha de seguir para obtener una estimación de las componenetes directa y difusa a partir del dato de radiación global, dado que es el que comúnmente se puede obtener de una localización determinada.

3.1.1. Radiación fuera de la atmósfera terrestre

Lo primero que se menciona en dicho proceso es la obtención de la irradiancia denominda extra-terrestre o extra-atmosférica, que es la radiación que llega a la atmósfera, directamente desde el Sol, que no sufre ninguna pérdida por interaccionar con algún medio. Como la relación entre el tamaño de nuesto planeta y la distancia entre el Sol y la Tierra es muy reducida, es posible asumir que el valor de dicha irradiancia es constante, siendo este valor $B_0 = 1367\frac{W}{m^2}$, según varias mediciones. Como la órbita que describe la Tierra alrededor del Sol no es totalmente circular, sino que tiene forma de elipse, para calcular la irradiancia incidente en una superficie tangente a la atmosfera en ua latitud concreta, debemos aplicar un factor de correción de la excentricidad de la elipse:

$$B_0(0) = B_0\epsilon_0 cos\theta_{zs} \tag{3.2}$$

Siendo cada componente:

- Constante solar: $B_0 = 1367\frac{W}{m^2}$
- Factor de corrección por excentricidad: $\epsilon_0 = (\frac{r_0}{r})^2 = 1 + 0{,}033 \cdot cos(\frac{2\pi d_n}{365})$[1]
- Ángulo zenital solar: $cos(\theta_{zs}) = cos(\delta)cos(\omega)cos(\phi) + sin(\delta) + sin(\phi)$[2] Donde:
  - Declinación: $\delta = 23,45° \cdot sin(\frac{2\pi \cdot (d_n+284)}{365})$ donde d_n es el dia del año.
  - Latitud: ϕ
  - Hora solar o tiempo solar verdadero: $\omega = 15 \cdot (TO - AO - 12) + \Delta\lambda + \frac{EoT}{4}$
    Donde:
    - Hora oficial: TO
    - Adelanto oficial durante el horario de verano: AO
    - Diferencia entre la longitud local y la longitud del huso horario: $\Delta\lambda$
    - Ecuación del tiempo: $EoT = 229,18 \cdot (-0,0334 \cdot sin(\frac{2\pi}{365{,}24} \cdot dn) + 0,04184 \cdot sin(2 \cdot \frac{2\pi}{365{,}24} \cdot dn + 3,5884))$

Esta irradiancia extra-terrestre solo tiene componentes geométicas. De modo que, si integramos la ecuación 3.2, se obtiene la irradiación diaria extra-terrestre:

$$B_{0d}(0) = -\frac{T}{\pi} B_0 \epsilon_0 (\omega_s sin\phi sin\delta + cos\phi cos\delta sin\omega_s) \tag{3.3}$$

Siendo:

- Ángulo del amananecer:

$$\omega_s = \begin{cases} -\arccos(-\tan\delta\tan\phi) & \text{si } |\tan\delta\tan\phi| < 1 \\ -\pi & \text{si } -\tan\delta\tan\phi < -1 \\ 0 & \text{si } -\tan\delta\tan\phi > 1 \end{cases}$$

Es posible demostrar que el promedio mensual de esta irradiación diaria coincide numéricamente con el valor de irradiación diaria correspondiente a los denominados "días promedios", días en los que la declinación correpondiente coincide con el promedio mensual (tabla 3.1)

Tabla 3.1: *Valor d_n correspondiente a los doce días promedio.*

Mes	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
d_n	17	45	74	105	135	161	199	230	261	292	322	347

[1] Para las ecuaciones de este apartado se va a optar por poner la ecuación más simple posible. Sin embargo, el paquete `solaR2` otorga la posibilidad de realizar los cálculos de utilizando las ecuaciones propuestas por 4 autores diferentes.

[2] Se van a utilizar las ecuaciones propuestas por P.I. Cooper [Coo69] por su simpleza.

3.1.2. Cálculo de componentes de radiación solar

Para caracterizar la radiación solar en un lugar, Liu y Jordan [LJ60] propusieron el índice de claridad, K_T. Este índice es la relación entre la radiación global y la radiación extra-atmosférica, ambas en el plano horizontal. El índice de claridad diario es la relación entre los valores diarios de irradiación:

$$K_{Td} = \frac{G_d(0)}{B_{0d}(0)} \tag{3.4}$$

mientras que el índice de claridad mensual es la relación entre las medias mensuales de la irradiación diaria:

$$K_{Tm} = \frac{G_{d,m}(0)}{B_{0d,m}(0)} \tag{3.5}$$

Una vez se tiene el índice de claridad, se puede calcular la fracción de radiación difusa en el plano horizontal. En el caso de medias mensuales [Pag61]:

$$F_{Dm} = 1 - 1,13 \cdot K_{Tm} \tag{3.6}$$

Donde:

- Fracción de radiación difusa: $F_D = \frac{D(0)}{G(0)}$

Al tener la fracción de radiación difusa, se pueden obtener los valores de la radiación directa y difusa en el plano horizontal:

$$D_d(0) = F_D \cdot G_d(0) \tag{3.7}$$

$$B_d(0) = G_d(0) - D_d(0) \tag{3.8}$$

3.2. Radiación en superficies inclinadas

Dados los valores de irradiación diaria difusa, directa y global en el plano horizontal se puede realizar la transformación al plano inclinado. Para ello, es necesario estimar el perfil de irradiancia correspondiente a cada valor de irradiación. dado que la variación solar durante una hora es baja, podemos suponer que el valor medio de la irradiancia durante esa hora coincide numéricamente con la irradiación horaria. Por otra parte, el análisis de valores *medios* en *largas* series temporales ha mostrado que la relación entre la irradiancia y la irradiación extra-atmosférica [CR79] (3.9):

$$r_D = \frac{D(0)}{D_d(0)} = \frac{B_0(0)}{B_{0d}(0)} \tag{3.9}$$

Este factor r_D es calculable directamente sabiendo que la relación entre irradiancia e irradiación extra-atmosférica es deducible teóricamente a partir de las ecuaciones 3.2 3.3:

$$\frac{B_0(0)}{B_{0d}(0)} = \frac{\pi}{T} \cdot \frac{cos(\omega) - cos(\omega_s)}{\omega_s \cdot cos(\omega_s) - sin(\omega_s)} = r_D \tag{3.10}$$

el mismo análisis mostró una relación entre la irradiancia e irradiación global asimilable a una función dependiente de la hora solar (3.11):

$$r_G = \frac{G(0)}{G_d(0)} = r_D \cdot (a + b \cdot cos(w)) \tag{3.11}$$

Donde:

- $a = 0,409 - 0,5016 \cdot sin(\omega_s + \frac{\pi}{3})$
- $b = 0,6609 + 0,4767 \cdot sin(\omega_s + \frac{\pi}{3})$

Es importante resaltar que estos perfiles proceden de medias sobre largos períodos, y de ahí que, como es observable en la figura 3.2, las fluctuaciones propias del movimiento de nubes a lo largo del día queden atenuadas y se obtenga una curva sin alteraciones.

3.2.1. Transformación al plano del generador

Una vez otenidos los valores de irradiancia en el plano horizontal, se traspone al plano del generador:

- **Irradiancia Directa** $B(\beta, \alpha)$: Ecuación basada en geometría solar (ángulo zenital) y del generador (ángulo de incidencia).

$$B(\beta, \alpha) = B(0) \cdot \frac{max(0, cos(\theta_s))}{cos(\theta_{zs})} \tag{3.12}$$

  donde:

  - Ángulo de inclinación: β.
  - Ángulo de orientación: α.

- **Irradiancia Difusa** $D(\beta, \alpha)$: Utilizando el modelo de cielo anisotrópico [Per23], se distinguen dos componentes de la irradiancia difusa, denominados *circunsolar* e *isotrópica*.

$$D(\beta, \alpha) = D^I(\beta, \alpha) + D^C(\beta, \alpha) \tag{3.13}$$

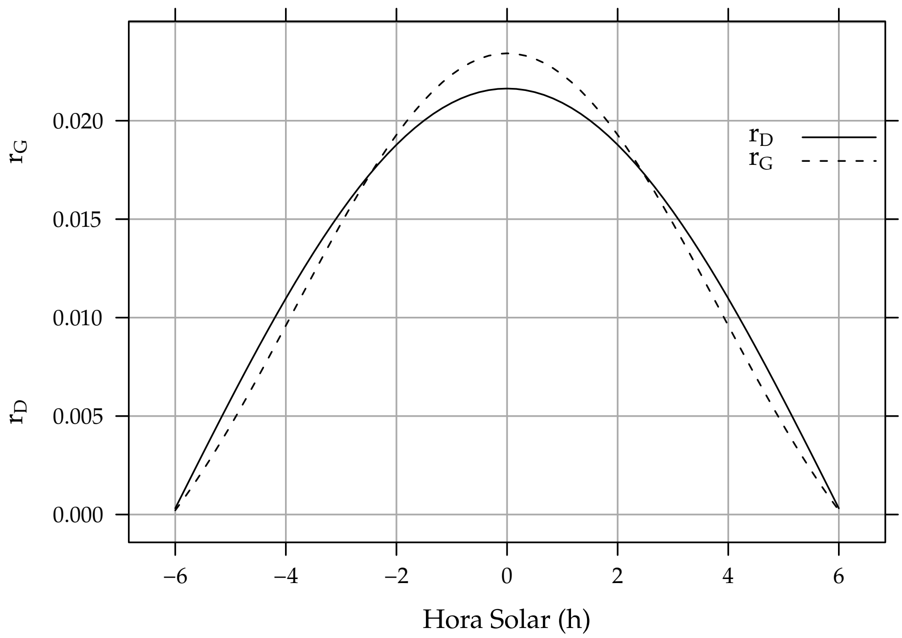


Figura 3.2: *Perfil de irradiancia difusa y global obtenido a partir del generador empírico de [CR79] para valores de irradiancia tomadas cada 10 minutos*

$$D^I(\beta,\alpha) = D(0)(1-k_1)\cdot\frac{1+cos(\beta)}{2} \tag{3.14}$$

$$D^C(\beta,\alpha) = D(0)\cdot k_1\cdot\frac{max(0,cos(\theta_s))}{cos(\theta_{zs})} \tag{3.15}$$

Donde:

- $k_1 = \frac{B(n)}{B_0\cdot\epsilon_0} = \frac{B(0)}{B_0(0)}$

- **Irradiancia de albedo** $R(\beta,\alpha)$: Se considera isotrópica debido a su baja contribución a la radiación global. Se calcula a partir de la irradiancia global en el plano horizontal usando un coeficiente de reflexión, ρ, que depende del terreno. En la ecuación 3.16, se utiliza el factor $\frac{1-cos(\beta)}{2}$, complemetario al factor de visión de la difusa isotrópica (figura 3.3)

$$R(\beta,\alpha) = \rho\cdot G(0)\cdot\frac{1-cos(\beta)}{2} \tag{3.16}$$

3.2.2. Ángulo de incidencia y suciedad

En un módulo fotovoltaico, la radiación incidente generalmente no es perpendicular a la superficie del módulo, lo que provoca pérdidas por reflexión o pérdidas angulares, cuantificadas por el ángulo de incidencia θ_s. La suciedad acumulada en la superficie del módulo también reduce la transmitancia del vidrio (representada por $T_{limpio}(0)$), disminuyendo la irradiancia efectiva, es decir, la radiación que realmente puede ser aprovechada por el módulo. La irradiancia efectiva para radiación directa se expresa en la ecuación 3.17:

$$B_{ef}(\beta,\alpha) = B(\beta,\alpha)\cdot[\frac{T_{sucio}(0)}{T_{limpio}(0)}]\cdot(1-FTB(\theta_s)) \tag{3.17}$$

donde $FTB(\theta_s)$ es el factor de pérdidas angulares, que se calcula con la ecuación 3.18:

$$FTB(\theta_s) = \frac{exp(-\frac{cos(\theta_s)}{a_r}) - exp(-\frac{1}{a_r})}{1-exp(-\frac{1}{a_r})} \tag{3.18}$$

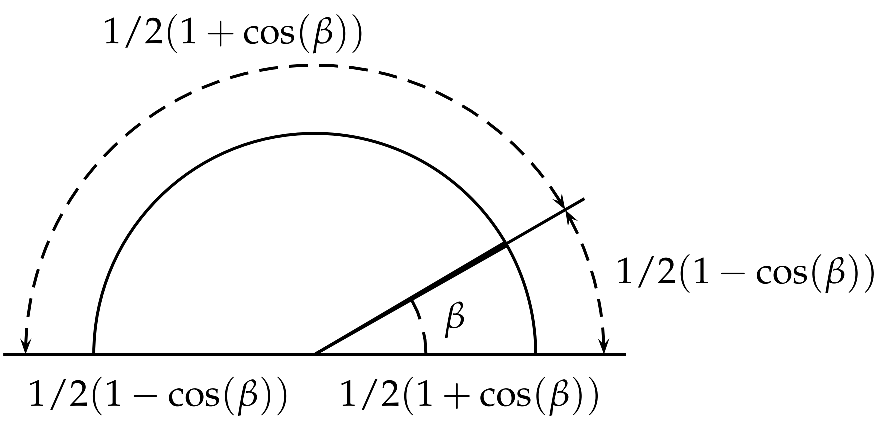


Figura 3.3: *Ángulo de visión del cielo*

Este factor depende el ángulo de incidencia $theta_s$ y del coeficiente de pérdidas angulares a_r. Cuando la radiación es perpendicular a la superficie ($\theta_s = 0$), FTB es cero. En la figura 3.4 se puede observar que las pérdidas angulares son más significativas cuando θ_s supera los 60º, y se acentúan con mayor suciedad.

Para calcular las componente de radiación difusa isotrópica y de albedo se utilizan las ecuaciones 3.19 y 3.2.2:

$$\mathrm{FTD}(\beta) \approx exp[-\frac{1}{a_r} \cdot (c_1 \cdot (sin\beta + \frac{\pi - \beta - sin\beta}{1 + cos\beta}) + c_2 \cdot (sin\beta + \frac{\pi - \beta - sin\beta}{1 + cos\beta})^2)] \tag{3.19}$$

$$\mathrm{FTR}(\beta) \approx exp[-\frac{1}{a_r} \cdot (c_1 \cdot (sin\beta + \frac{\beta - sin\beta}{1 - cos\beta}) + c_2 \cdot (sin\beta + \frac{\beta - sin\beta}{1 - cos\beta})^2)] \tag{3.20}$$

Donde:

- Ángulo de inclinación del generador (en radianes): β
- Coeficiente de pérdidas angulares: a_r
- Coeficientes de ajuste: c_1 y c_2 (en la tabla 3.2 se recogen algunos valores característicos de un módulo de silicio monocristalino convencional para diferentes grados de suciedad)

Para estas componenetes el cálculo de irradiancia efectiva es similar al de la irradiancia directa (ecuaciones 3.21 y 3.23). Para la componente difusa circunsolar emplearemos el factor de pérdidas angulares de la irradiancia efectiva(ecuacion 3.22):

$$D_{ef}^I(\beta, \alpha) = D^I(\beta, \alpha) \cdot [\frac{T_{sucio}(0)}{T_{limpio}(0)}] \cdot (1 - FT_D(\beta)) \tag{3.21}$$

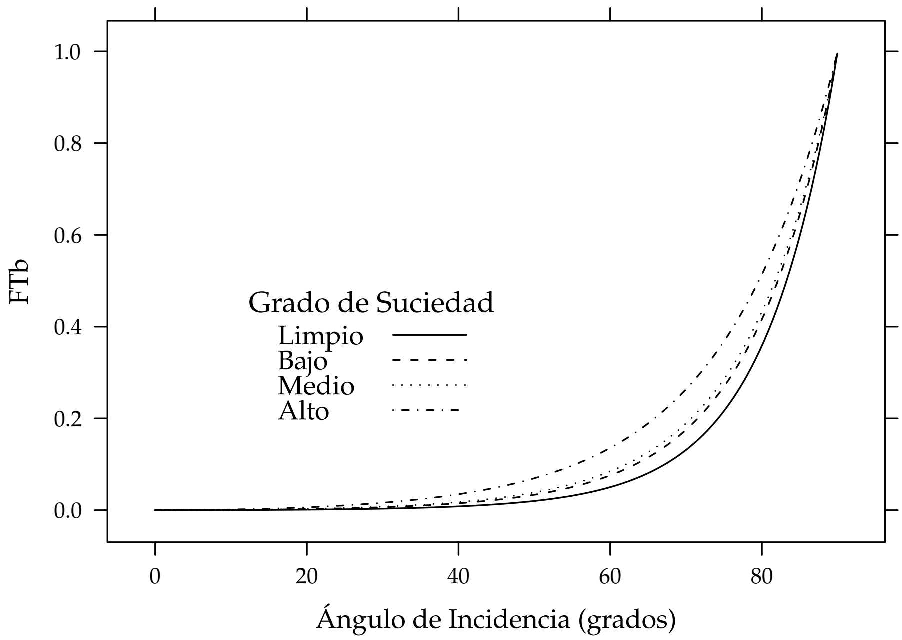


Figura 3.4: *Pérdidas angulares de un módulo fotovoltaico para diferentes grados de suciedad en función del ángulo de incidencia.*

Tabla 3.2: *Valores del coeficiente de pérdidas angulares y transmitancia relativa en incidencia normal para diferentes tipos de suciedad.*

Grado de suciedad	$\frac{T_{sucio}(0)}{T_{limpio}(0)}$	a_r	c_2
Limpio	1	0.17	-0.069
Bajo	0.98	0.20	-0.054
Medio	0.97	0.21	-0.049
Alto	0.92	0.27	-0.023

$$D_{ef}^C(\beta,\alpha) = D^C(\beta,\alpha) \cdot [\frac{T_{sucio}(0)}{T_{limpio}(0)}] \cdot (1 - FT_B(\theta_s)) \tag{3.22}$$

$$R_{ef}(\beta,\alpha) = R(\beta,\alpha) \cdot [\frac{T_{sucio}(0)}{T_{limpio}(0)}] \cdot (1 - FT_R(\beta)) \tag{3.23}$$

Siguiendo el esquema de la figura 3.1, a partir de estas irradiancias efectivas se puede calcular la irradiación global efectiva diaria, mensual y anual. Comparando la irradiación global incidente con la irradición efectiva, se puede evaluar el impacto de la suciedad y el desajuste del ángulo en períoods prolongados.

3.3. Cálculo de la energía producida por un generador fotovoltaico

3.3.1. Funcionamiento de una célula solar

Para calcular la energía producida por un generador fotovoltaico, se deben tener en cuenta la influencia de factores tales como la radiación o la temperatura en una célula solar y en los valores de tensión y corriente que se alcanzan en dichas condiciones.

Para definir una célula solar, se tomar 4 variables:

- La corriente de cortocircuito: I_{sc}
- La tensión de circuito abierto: V_{oc}
- La corriente en el punto de máxima potencia: I_{mpp}
- La tensión en el punto de máxima potencia: V_{mpp}

Punto de máxima potencia

El punto de máxima potencia es aquel situado en la curva de funcionamiento del generador donde, como su propio nombre indica, los valores de tensión y corriente son tales que la potencia que entrega es máxima (figura 3.5).

Factor de forma y eficiencia

El área encerrada por el rectángulo definido por el producto $I_{mpp} \cdot V_{mpp}$ es, como e observable en la figura 3.5, inferiro a la respresentada por el producto $I_{sc} \cdot V_{oc}$. La relación entre estad dos superficies se cuantifica con el factor de forma:

$$FF = \frac{I_{mpp} \cdot V_{mpp}}{I_{sc} \cdot V_{oc}} \tag{3.24}$$

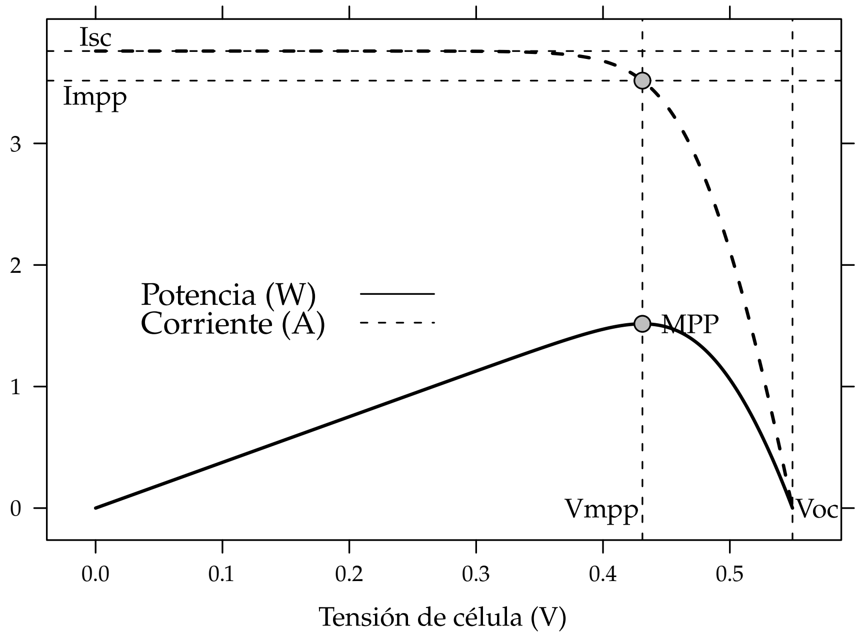


FIGURA 3.5: *Curvas corriente-tensión(línea discontinua) y potencia-tensión(línea continua) de una célula solar ($T_a = 20°C$ y $G = 800W/m^2$)*

Conociendo los valores de I_{sc} y V_{oc} es posible calcular la potencia en el punto de máxima potencia, dado que $P_{mpp} = FF \cdot I_{sc} \cdot V_{oc}$.

Por otra parte, la calidad de una célula se puede cuantificar con la eficiencia de conversión (ecuación 3.25).

$$\eta = \frac{I_{mpp} \cdot V_{mpp}}{P_L} \tag{3.25}$$

donde $P_L = A_c \cdot G_{ef}$ representa la potencia luminosa que incide en la célula.Como es evidente de la ecuación 3.25, este valor de eficiencia se corresponde al caso en el que el acoplamiento entre la carga y la célula permite a ésta trabajar en el punto de máxima potencia. En la figura 3.6 se muestra la evolución temporal del valor de eficiencia de célula de laboratorio para diferentes tecnologías.

Influencia de la temperatura y la radiación

La temperatura y la radiación son factores cruciales en el funcionamiento de una célula solar. El aumento de la temperatura ambiente reduce la tensión de circuito abierto según la relación dV_{oc}/dT_c, , que para células de silicio cristalino es de$-2,3\frac{mV}{°C}$. Además, disminuye la eficiencia de la célula solar con $\frac{d\eta}{dT_c} = -0,4\,\%/°C$.

En cuanto a la iluminación, la fotocorriente y la tensíon de circuito abierto son proporcionales a la irradiancia incidente.

Tomando en cuanta estas influencias, se definen una condiciones de funcionamiento, denominadas condiciones estándar de medida(STC), válidas para caracterizar una célula en el entorno de un laboratorio. Estas condiciones vienen determinadas por:

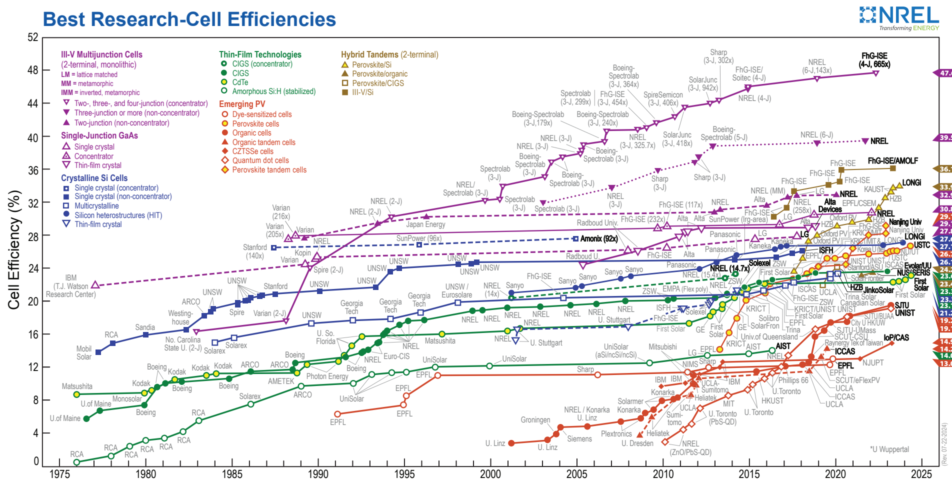


Figura 3.6: *Evolución de la eficiencia de células según la tecnología (según el National Renewable Energy Laboratory [Nat24] (EEUU)).*

- Irradiancia: $G_{stc} = 1000 W/m^2$ con incidencia normal.
- Temperatura de célula: $T_c^* = 25°C$.
- Masa de aire: $AM = 1,5$.[3]

Frecuentemente los fabricantes informan de los valores de las tensiones V_{oc}^* y V_{mpp}^* y las corrientes I_{sc}^* y I_{mpp}^*[4]. A partir de estos valores es posible referir a estas condiciones:

- La potencia: $P_{mpp}^* = I_{mpp}^* \cdot V_{mpp}^*$
- El factor de forma: $FF^* = \frac{P_{mpp}^*}{I_{sc}^* \cdot V_{oc}^*}$
- La eficiencia: $\eta^* = \frac{I_{mpp}^* \cdot V_{mpp}^*}{A_c \cdot G_{stc}}$

3.3.2. Funcionamiento de un módulo fotovoltaico

Comportamiento térmico de un módulo

La mayoría de las ecuaciones ue definen el comportamiento de un módulo fotovoltaico se establecen en lo que se conocen como condiciones estándar de funcionamiento. En estas condiciones, la temperatura de la célula es de $25°C$. Sin embargo, la temperatura de operación de la célula es diferente y depende directamente de la radiación que recibe el módulo en cada momento.

El módulo recibe una cantidad de radiación dada, absorbiendo la fracción de ésta que no se refleja al exterior. De dicha fracción, parte de ella es transformada en energía eléctrica mientras que el resto se entrega en forma de calor al entorno.

[3] Relación entre el camino recorrido por los rayos directos del Sol a través de la atmósfera hasta la superficie receptora y el que recorrerían en caso de incidencia vertical ($AM = 1/cos\theta_{zs}$).

[4] Es de uso común añadir un asterisco como superíndice para denotar aquellos parámetros medidos en estas condiciones.

Para simplificar, se puede asumir que el incremento de la temperatura de la célula respecto de la temperatura ambiente depende linealmente de la irradiancia incidente en ésta. El coeficiente de proporcionalidad depende de muchos factores, tales como el modo de instalación del módulo, la velocidad del viento, la humedad ambiente y las características constructivas del laminado.

Estos factores quedan recogidos en un valor único representado por la temperatura de operación nominal de célula (NOCT o TONC), definida como aquella que alcanza una *célula* cuando su *módulo* trabaja en las siguientes condiciones:

- Irradiancia: $G = 800W/m^2$.
- Masa de aire: $AM = 1,5$.
- Irradiancia normal.
- Temperatura *ambiente*: $T_a = 20°C$.
- Velocidad de viento: $v_v = 1m/s$.

La ecuación 3.26 expresa una aproximación aceptable del comportamiento térmico de una célula integrada en un módulo en base a las consideraciones previas:

$$T_c = T_a + G_{ef} \cdot \frac{NOCT - 20}{800} \tag{3.26}$$

Para la simulación del funcionmaiento de un módulo fotovoltaico en condiciones de operación real, es necesario contar con secuencias de valores de temperatura ambiente. Si no se dispone de información detallada, se puede asumir un valor constante de $T_a = 25°C$ para simulaciones anuales. Sin embargo, si se conocen los valores máximos y mínimos diarios de la temperatura ambiente, se puede generar una secuencia intradiaria usando una combinación de funciones coseno.

Cálculo de V_{oc} y I_{sc}

Conociendo ya los valores horarios de temperatura de la célula, se puede calcular V_{oc} utilizando la ecuación 3.27. Y, por último, mediante la ecuación 3.28 se puede calcular I_{sc}.

$$V_{oc}(T_c) = V_{oc}^* + (T_c - T_c^*) \cdot \frac{dV_{oc}}{dT_c} \cdot N_{cs} \tag{3.27}$$

$$I_{sc} = G_{ef} \cdot \frac{I_{sc}^*}{G^*} \tag{3.28}$$

Factor de forma variable

Una vez obtenidos los valores de V_{oc} y I_{sc}, el siguiente paso ha de ser calcular los valores de tensión y corriente en el punto de máxima potencia, pues es donde el generador estará entregando su máxima potencia, como su propio nombre indica, y por tanto es un punto de interés para el cálculo.

Existen dos metodologías de cálculo de dicho punto, uno de ellos significantemente más sencillo que el otro. Éste consiste en suponer que el Factor de Forma, definido en la expresión 3.24 es constante.

Si suponemos que FF es constante, se podrían extraer los valores de tensión y corriente en el punto de máxima potencia ya que si

$$FF = FF^* \tag{3.29}$$

entonces

$$\frac{I_{mpp} \cdot V_{vmpp}}{I_{sc} \cdot V_{oc}} = \frac{I^*_{mpp} \cdot V^*_{vmpp}}{I^*_{sc} \cdot V^*_{oc}} \tag{3.30}$$

pudiendo así obtener los valores de I_{mpp} y V_{vmmp}.

Sin embargo, este suposición da resultados alejados a una estimación acertada. Por ello, se tendrá en cuenta la variación del factor de forma:

- **Cálculo de la tensión termica, V_t, a temperatura de la célula**: Se calculará el valor de V_t a 25ºC con la expresión:

$$V_{tn} = \frac{V_t \cdot (273 + 25)}{300} \tag{3.31}$$

- **Cálculo de R^*_s**: El segundo paso consiste en calcular el valor de resistencia en serie con los valores STC:

$$R^*_s = \frac{\frac{V^*_{oc}}{N_{cs}} - \frac{V^*_{mpp}}{N_{cs}} + m \cdot V_{tn} \cdot ln(1 - \frac{I^*_{mpp}}{I^*_{sc}})}{\frac{I^*_{mpp}}{N_{cp}}} \tag{3.32}$$

- **Cálculo de r_s**: Utilizando el valors de R^*_s calculado en el paso anterior junto con los valores de V_{oc} y I_{sc} podemos calcular r_s que se utilizará más adelante en el proceso.

$$r_s = R^*_s \cdot (\frac{N_{cs}}{N_{cp}} \cdot \frac{I_{sc}}{V_{oc}}) \tag{3.33}$$

- **Cálculo de k_{oc}**: A continuación, utilizando los valores de temperatura ambiente obtenidos con anterioridad junto con la tensión de circuito abierto, se calcula k_{oc} mediante la expresión:

$$k_{oc} = \frac{V_{oc}/N_{cs}}{m \cdot V_t \cdot \frac{T_c+273}{300}} \tag{3.34}$$

Con éstos cálculos previos, éste método propone localizar el punto de máxima potencia de forma aprodimada mediante la ecuaciones:

$$i_{mpp} = 1 - \frac{D_M}{k_{oc}} \tag{3.35}$$

$$v_{mpp} = 1 - \frac{ln(k_{oc}/D_M)}{k_{oc}} - r_s \cdot i_{mpp} \tag{3.36}$$

donde:

$$D_M = D_{M0} + 2 \cdot r_s \cdot D^2_{M0} \tag{3.37}$$

$$D_{M0} = \frac{k_{oc} - 1}{k_{oc} - lnk_{oc}} \tag{3.38}$$

Por último, multiplicando los valores de i_{mpp} y v_{mpp} por I_{sc} y V_{oc} respectivamente, se obtienen los valores de I_{mpp} y V_{mpp} que serán los que se utilicen para calcular la potencia entregada por el generador en el punto de máxima potencia.

Teniendo estos valores se puede obtener:

$$P_{mpp} = I_{mpp} \cdot V_{mpp} \tag{3.39}$$

3.3.3. Cálculo de potencias y energías de un sistema fotovoltaico conectado a la red

La potencia obtenida en el paso anterior es la de un solo módulo. Para conocer la potencia que va a ser capaz de entregar un sfcr, se debe tener en cuenta su configuración de módulos en serie y en paralelo.

$$P_g^* = N_s \cdot N_p \cdot P_m^* \tag{3.40}$$

Con este paso se obtiene la potencia horaria entregada por el generador fotovoltaico. El siguiente paso será pasar esa potencia a través del inversor y calcular la potencia a la salida de este.

Primero, se esteblecen las expresiones de las potencias normalizadas. Siendo P_{inv} la potencia nominal del inversor:

$$p_i = \frac{P_{DC}}{P_{inv}} \tag{3.41}$$

$$p_o = \frac{P_{AC}}{P_{inv}} \tag{3.42}$$

Por otro lado, el rendimiento de un inversor fotovoltaico se puede modelizar de la siguiente manera:

$$\eta_{inv} = \frac{p_o}{p_o + k_0 + k_1 p_o + k_2 p_o^2} \tag{3.43}$$

De las dos ecuaciones anteriores se puede deducir:

$$p_i = p_o + k_0 + k_1 p_o + k_2 p_o^2 \tag{3.44}$$

Desarrollando esta ecuación, se puede obtener una ecuación de segundo grado con p_o como incógnita:

$$k_2 p_o^2 + (k_1 + 1) p_o + (k_0 - p_i) = 0 \tag{3.45}$$

Por último, volviendo a las primeras expresiones se puede obtener la potencia en corriente alterna:

$$P_{AC} = p_o \cdot P_{inv} \tag{3.46}$$

Con esta potencia, integrando en función del tiempo se puede obtener la energía que genera el sistema

$$E_{AC} = \int_T P_{AC}\, dt \tag{3.47}$$

y la productividad:

$$Y_f = \frac{E_{ac}}{P_g^*} \tag{3.48}$$

3.3.4. Cálculo de potencias y energías de un sistema fovoltaico de bombeo

Potencia hidráulica

La potencia hidráulica, P_H , necesaria para bombear agua es función de,

- La altura vertical aparente, H_v
- El caudal de agua, Q

$$P_H = g \cdot \rho \cdot Q \cdot H_V \tag{3.49}$$

Expresando P_H en watios, H_v en metros y Q en m^3/h la ecuación resulta en:

$$P_H = 2{,}725 \cdot Q \cdot H_v \tag{3.50}$$

Asumiendo que el agua bombeado sale por el coducto a baja velocidad, la potencia de salida de la bomba necesita satisfacer P_H más las pérdidas de fricción en la tubería, P_f . Este valor se asimila a una altura equivalente H_f asociado a un caudal determinado:

$$H_T = H_v + H_f \tag{3.51}$$

La potencia eléctrica a la entrada de la motobomba, P_{el}, es:

$$P_{el} = \frac{P_H + P_f}{\eta_{mp}} \tag{3.52}$$

donde η_{mp} es la eficiencia de la motobomba. La potencia eléctrica requerida por la motobomba es entregada por un generador FV y acondicionador de potencia:

$$P_{el} = P_g^* \cdot \frac{G}{G_{stc}} \frac{\eta_g}{\eta_g^*} \cdot \eta_{inv} \tag{3.53}$$

siendo G la irradiancia en el plano del generador, eta_{inv} la eficiencia del equipo de acondicionamiento de potencia y $\frac{\eta_g}{\eta_g^*}$ modela el comportamiento del generador con la temperatura.

Caudal diario

El caudal diario bombeado por este conjunto es:

$$Q_d = \int_d \frac{G}{G^*} \cdot P_g^* \cdot \eta_g \cdot \frac{\eta_{ig}}{\eta_{ig}^*} \cdot \eta_{inv} \cdot \eta_{mp}\, dt \tag{3.54}$$

Altura

Se puede definir una altura total equivalente, H_{TE}, con las siguientes suposiciones:

- Las pérdidas de fricción en tubería son despreciables ($H_f < 0{,}05 \cdot H_T$).
- El nivel del agua dentro del pozo se mantiene constante.

$$Q_d = \frac{P_g^*}{2{,}725 \cdot G^* \cdot H_{TE}} \int \left(\frac{G}{\eta_{ig}^* \eta_m^* \eta_{inv} \eta_{mp}} \right) dt \tag{3.55}$$

Ahora el cálculo en la integral sólo depende de la radiación, temperatura, y equipos.

Para calcular esta altura total equivalente, se debe suponer que:

- El pozo está caracterizado con tres parámetros:
  - Nivel estático, H_{st}.
  - Nivel dinámico, H_{dt}.
  - Caudal de ensayo, Q_t.

- Que se ha realizado el ensayo de bombeo para caracterizar los pozos con bomba portátil empleando el caudal máximo del pozo, Q_{max} ($Q_t = Q_{max}$).

Con estas suposiciones se puede llegar a la expresión:

$$H_{TE} = H_{ot} + H_{st} + \left(\frac{H_{dt} - H_{st}}{Q_T}\right) Q_{AP} + H_f(Q_{AP}) \tag{3.56}$$

donde:

- H_{OT}, es la altura desde la salida de agua hasta el suelo.
- Nivel estático, H_{st}.
- Nivel dinámico, H_{dt}.
- Caudal aparente, $Q_{AP} = \alpha \cdot Q_d$ ($\alpha = 1/24 = 0{,}0416h^{-1}$).
- $H_f(Q_{AP})$, pérdidas en la tubería al caudal aparente.

Potencia del generador

Como primera aproximación, se consideran constantes a lo largo del tiempo las eficiencias de los componentes del sistema con la elección de ciertos valores adecuados ($\frac{\eta_g}{\eta_g^*} = 0{,}85, \eta_{mp} = 0{,}35, \eta_{inv} = 0{,}9$). Así, es posible calcular de forma aproximada la potencia nominal del generador necesaria para bombear un caudal diario Q_d a una altura total equivalente H_{TE} a partir de la ecuación:

$$P_g^* = \frac{10 \cdot HTE \cdot Q_d}{\frac{G_d}{G_{stc}}} \tag{3.57}$$

Simulación de sistemas fotovoltaicos de bombeo

Debido a la complicación del cálculo del dimensionamiento de los sistemas fotovoltaicos de bombeo, se puede recurrir a métodos de simulación asistidos por ordenaor. El algoritmo a seguir es:

1. Curva característica de la bomba que relaciona la altura, H, y el caudal, Q, a la frecuencia nominal de la bomba:
$$H = a \cdot f^2 + b \cdot f \cdot Q + c \cdot Q^2 \tag{3.58}$$
    - Donde a, b, y c son coeficientes característicos de la bomba y f es la frecuencia.
2. Relaciones de semejanza para bombas centrífugas:
$$\frac{f_1}{f_2} = \frac{Q_1}{Q_2} = \left(\frac{H_1}{H_2}\right)^{1/2} = \left(\frac{P_1}{P_2}\right)^{1/3} \tag{3.59}$$
3. Cálculo de caudal y altura a frecuencia nominal (50 Hz):
$$Q_{50} = \frac{50 \cdot Q}{f} \tag{3.60}$$
$$H_{50} = H \cdot \left(\frac{50}{f}\right)^2 \tag{3.61}$$

4. Ecuación de potencia hidráulica:

$$P_{h,50} = 2{,}725 \cdot Q_{50} \cdot H_{50} \tag{3.62}$$

5. Potencia mecánica en el eje de la bomba a 50 Hz:

$$P_{b,50} = \frac{P_{h,50}}{\eta_b} \tag{3.63}$$

6. Potencia mecánica a frecuencia f:

$$P_b = P_{b,50} \cdot \left(\frac{f}{50}\right)^3 \tag{3.64}$$

7. Potencia eléctrica demandada por el motor:

$$P_{bc} = P_b \cdot \frac{50}{f} \tag{3.65}$$

$$P_{e,50} = \frac{P_{bc}}{\eta_m} \tag{3.66}$$

$$P_e = P_{e,50} \cdot \frac{f}{50} \tag{3.67}$$

8. Perfil de irradiancia diaria (según IEC 61725):

$$G = G_{max} \cdot \cos\left(\frac{t}{t_0} \cdot \frac{\pi}{2}\right) \cdot \left[1 + s \cdot \left(1 - \cos\left(\frac{t}{t_0} \cdot \frac{\pi}{2}\right)\right)\right] \tag{3.68}$$

donde G es la irradiancia (W/m^2) en la hora t, G_{max} es el valor máximo de irradiancia (W/m^2) dureante el día en cuestión, y s es el facotor de forma definido por:

$$s = \frac{d \cdot \frac{\pi}{2} - 1}{1 - \frac{\pi}{4}} \tag{3.69}$$

siendo d el factor de conjunto de datos calculado con:

$$d = \frac{G_d}{G_{max} \cdot h} \tag{3.70}$$

3.4. Sombras y ocupación de terreno

Al diseñar una central fotovoltaica se debe decidir la ubicación de las diferentes partes del generador resolviendo un compromiso entre la mejor ocupación del terreno disponible y la minimización del impacto de sombras mutuas arrojadas entre los módulos.

Este factor de sombras implica un nivel de ocupación de terreno que depende del modo de seguimiento del generador. La ocupación del terreno se puede medir con dos métricas:

- Relación de ocupación del terreno (*Ground Coverage Ratio*, GCR): es la relación entre el área del generador, A_G, y el área del terreno ocupado, A_T (por tanto, siempre será GCR <1).

$$GCR = \frac{A_G}{A_T} \tag{3.71}$$

- Ratio de ocupación del terreno (ROT, o *Ground Requirement Ratio*, GRR): es el inverso del GCR, la relación entre el área de terreno ocupado, A_T, y el área del generador, A_G.

$$ROT = \frac{A_T}{A_G} \tag{3.72}$$

3.4.1. Sistemas estáticos

Las filas que componen el generador arojan sombras unas sobre otras en determinados momentos del días y año. Como recomendación general, es de uso común respetar un mínimo de 4 horas de sol en torno al mediodía del solsticio de invierno libres de sombra. La longitud de la sombra de un obstáculo se mide con:

$$d = \frac{h}{tan\gamma_s} \tag{3.73}$$

siendo h la altura de la fila adyacente, $h = L \cdot sin(\beta)$ y L la longitud del generador, según se indica en la figura 3.7.

En el mediodía del solsticio de invierno la altura solar es $\gamma_s = 90° - 23{,}45° - |\phi| \simeq 67° - |\phi|$. Por tanto, la distancia mínima que permite 4 horas libres de sombra alrededor del mediodía es:

$$d_{min} = \frac{h}{tan(61° - |\phi|)} \tag{3.74}$$

3.4.2. Sistemas de seguimiento a doble eje

El diseño de un sistema de seguimiento solar a doble eje busca optimizar la ubicación de los seguidores para minimizar las pérdidas de radiciación por sombras, utilizando eficientemente el terreno. Para esto, se simula el sistema en diferentes configuraciones y se elige la más eficiente en términos de productividad y ROT, que se calcucula con la fórmula:

$$ROT = \frac{L_{ns} \cdot L_{eo}}{L \cdot W} \tag{3.75}$$

donde (figuras 3.8 y 3.9):

- L_{ns} es la separación entre seguidores en la dirección Norte-Sur.
- L_{eo} es la separación en la dirección Este-Oeste.
- L es la longitud del seguidor.
- W es la anchura del seguidor.

El sistema se modela como un grupo de seis seguidores en una matriz de dos filas en dirección Norte-Sur (figura 3.10), representando tres situaciones de sombra: lateral (Este-Oeste), frontal

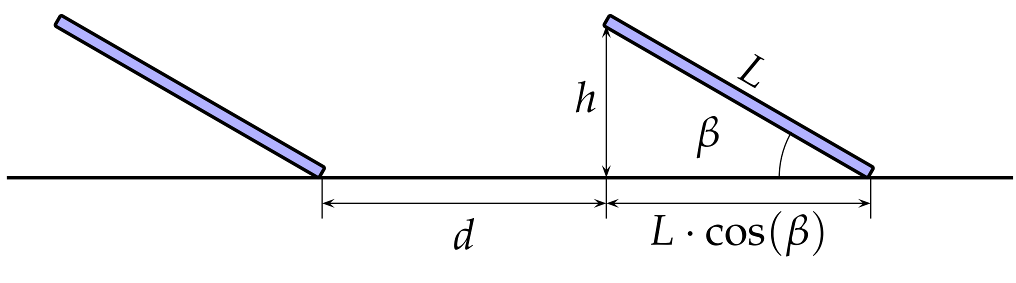


Figura 3.7: *Dimensiones y distancias entre filas de un sistema estático.*

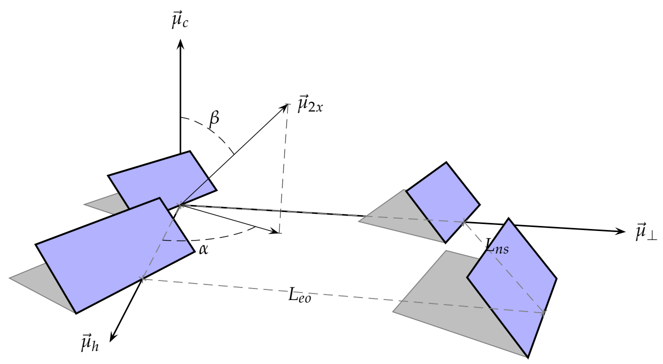


Figura 3.8: *Sombras mutuas en un conjunto de cuatro seguidores.*

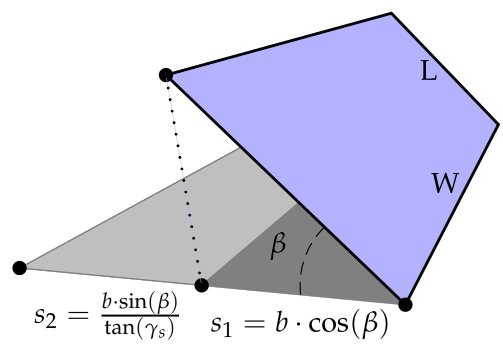


Figura 3.9: *Dimensiones de un seguidor a doble eje y longitud de su sombra arrojada.*

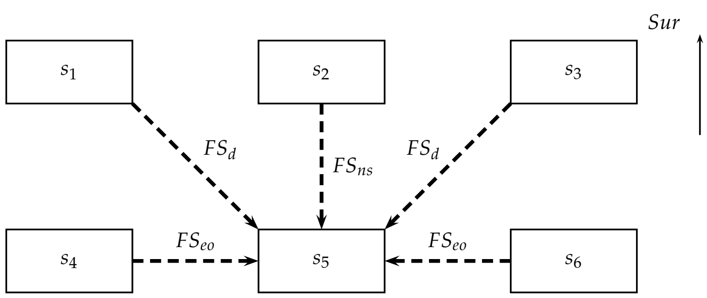


Figura 3.10: *Posibles sombras en un conjunto de seis seguidores.*

(Norte-Sur) y diagonal, caracterizados por los factores de sombra FS_{xx}, definidos como la relación entre el área sombreada y el área total del generador. Las ecuaciones para estos factores son, en las que se emplean los valores normalizados de las distancias, $l_{eo} = \frac{L_{eo}}{W}$ y $l_{ns} = \frac{L_{ns}}{W}$:

$$\begin{array}{r} |l_{eo} \cdot \cos(\psi_s)| < 1 \\ |l_{eo} \cdot \sin(\psi_s)| < s \end{array} \Rightarrow FS_{eo} = \frac{(1 - |l_{eo} \cos(\psi_s)|) \cdot (s - |l_{eo} \sin(\psi_s)|)}{s} \tag{3.76}$$

$$\begin{array}{r} |l_{ns} \cdot \cos(\psi_s)| < s \\ |l_{ns} \cdot \sin(\psi_s)| < 1 \end{array} \Rightarrow FS_{ns} = \frac{(s - |l_{ns} \cos(\psi_s)|) \cdot (1 - |l_{ns} \sin(\psi_s)|)}{s} \tag{3.77}$$

$$\begin{array}{r} s > |l_{ns} \cdot \cos(\psi_s)| + |l_{eo} \sin(\psi_s)| \\ 1 > |l_{eo} \cdot \cos(\psi_s)| - |l_{ns} \cdot \sin(\psi_s)| \end{array} \Rightarrow$$

$$FS_d = \frac{[s - (|l_{eo} \cdot \sin(\psi_s)| + |l_{ns} \cos(\psi_s)|)] \cdot [1 - (|l_{eo} \cdot \cos(\psi_s)| - |l_{ns} \sin(\psi_s)|)]}{s} \tag{3.78}$$

siendo ψ_s el acimut solar y γ_s la altura solar y donde la longitud de sombra (normalizada con la anchura del seguidor) se calcula con:

$$s = s_1 + s_2 \tag{3.79}$$

$$s_1 = b \cdot \cos(\beta) \tag{3.80}$$

$$s_2 = \frac{b \cdot \sin(\beta)}{|\tan(\gamma_s)|} \tag{3.81}$$

El factor $\frac{\sin(\gamma_s)}{\sin(\gamma_s+\beta)}$ representa la proyección de sombra existente en el suelo sobre el plano del generador, y por tanto, el porcentaje de área sombreada que debe ser eliminado de la radiación directa. Desarrollando este factor se obtiene una formulación alternativa que puede facilitar el cálculo de los tres factores:

$$FS_{eo} = \frac{(1 - l_{eo} \cos(\psi_s)) \cdot (s - l_{eo} \sin(\psi_s))}{s} \tag{3.82}$$

$$FS_{ns} = \frac{(s - l_{ns} \cos(\psi_s)) \cdot (1 - l_{ns} \sin(\psi_s))}{s} \tag{3.83}$$

$$FS_d = \frac{[s - (l_{eo} \cdot \sin(\psi_s) + l_{ns} \cos(\psi_s))] \cdot [1 - (l_{eo} \cdot \cos(\psi_s) - l_{ns} \sin(\psi_s))]}{s} \tag{3.84}$$

Realizando la simulación de este sistema incluyendo el cálculo de sombras, y repitiendo la simulación para varias combinaciones (Lns, Leo) pueden elaborarse gráficos de nivel como el de la figura 3.11, donde se recoge el ratio entre la energía anual producida por un seguidor *promedio* incluyendo el efecto de por sombras mutuas[5] y la energía anual producida por un seguidor sin sombreado.

3.4.3. Sistemas de seguimiento de eje horizontal

Se considera que los seguidores son de longitud infinita en sentido Norte-Sur (se desprecia el efecto de borde). Así, los parámetros que determinan el diseño de este tipo de sistema son (figura 3.12):

1. La inclinación del generador fotovoltaico, β, (coincidente con el ángulo ψ_{ns}).

[5] En el cálculo de la producción del seguidor afectado por sombras mutuas se considera que la reducción en potencia está exclusivamente relacionada con el área sombreada, por tanto, no se tienen en cuenta las conexiones eléctricas entre módulos.

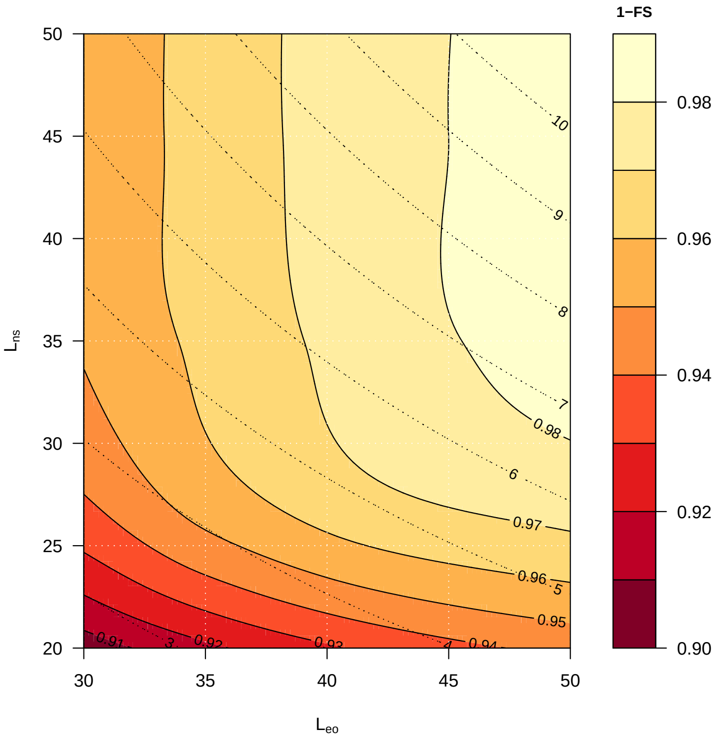


Figura 3.11: *Ábaco para planta de seguimiento a doble eje. Recoge el ratio entre la energía anual producida por un seguidor afectado por sombras mutuas (E_{acS}) y la producida por un seguidor sin sombreado (E_{ac0}). Las curvas de color negro representan la fracción de energía no afectada por sombras. Las curvas de puntos representan el valor del ROT.*

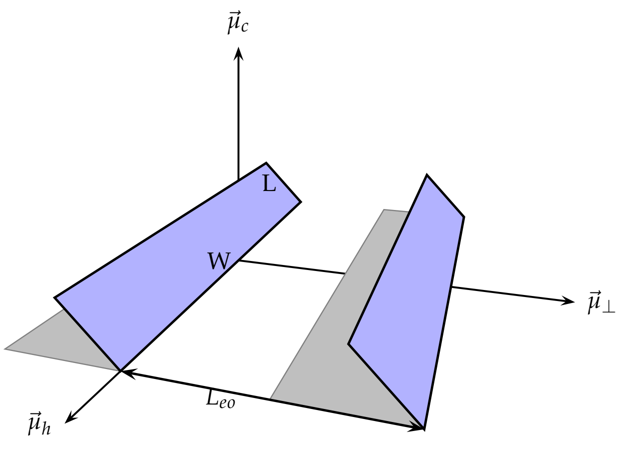


Figura 3.12: *Dimensiones básicas en sistemas con seguidores de eje horizontal.*

2. La dimensión en sentido Este-Oeste del campo generador, L.
3. La separación entre los diferente seguidores en la dirección Este-Oeste, L_{eo}. Por tanto, $ROT = \frac{L_{eo}}{L}$.

Para caracterizar numéricamente el sombreado, se empreará el factor FS_{eo}. Mediante consideraciones geométricas, utilizando la distancia normalizada $l_{eo} = \frac{L_{eo}}{L}$, es posible escribir:

$$\begin{aligned} FS_{eo} &= \frac{s - l_{eo}}{s} \\ &= 1 - l_{eo} \cdot \cos(\beta) \\ &= 1 - l_{eo} \cdot \frac{\sin(\omega)}{\sqrt{\sin^2(\omega) + (\cos(\omega)\cos(\phi) + \tan(\delta)\sin(\phi))^2}} \end{aligned} \tag{3.85}$$

Limitación de ángulo y retroseguimeitno

En seguidores de eje horizontal se puede evitar la incidencia de sombras en cualquier en cualquier instante mediante algoritmos de *backtracking* o retroseguimiento [Pan+91]. Esta técnia provoca el desvío del seguidor de su posición óptima en los instantes en los que se produce la sombra entre seguidores, evitando el impacto de sombras pero con la consiguiente reducción en energía producida pro alejamiento del apuntamiento óptimo.

Para evitar la aparición de sombras, el ángulo de inclinación de los seguidores debe ser tal que la longitud de la sombra sea igual a la distancia entre seguidores. Siendo, β el ángulo de inclinación con retroseguimiento, y, β_0 el ángulo de inclinación original, de la ecuación 3.85 se deduce que sólo será necesario aplicar esta técnica cuando $l_{eo} \cdot cos(\beta_0) \leq 1$. El triángulo definido por el rayo solar, el seguidro y la sombra debe cumplir la siguiente condición, basada en el teorema de los senos:

$$\frac{l_{eo}}{\cos(\beta_0 - \beta)} = \frac{1}{\cos\beta_0} \tag{3.86}$$

Por tanto, el ángulo de inclinación que grantiza la ausencia de sombras a costa de apartarse de la condición de seguimiento es:

$$\beta = \beta_0 - \arccos(l_{eo} \cdot \cos\beta_0) \tag{3.87}$$

ecuación que debe aplicarse sólo cuando $l_{eo} \cdot \cos(\beta_0) \leq 1$. En caso contrario $\beta = \beta_0$.

CAPÍTULO 4

Desarrollo del código

En la figura 4.1, se muestra el proceso de cálculo que sigue el paquete a la hora de obtener la estimación de la producción del sistema fotovoltaico. A la hora de estimar la producción, el programa sigue los siguientes procesos:

4.1. Geometría solar

Para calcular la geometría que definen las posiciones de la Tierra y el Sol, **solaR2** se vale de una función constructora, **calcSol** [??], la cual mediante las funciones **fSolD** [??] y **fSolI** [??] cálcula todos los ángulos y componentes que caracterizan la geometría solar.

Como se puede ver en la figura 4.2, **calcSol** funcia gracias a las siguientes funciones:

- **fSolD**: la cual, a partir de la latitud (ϕ), calcula la geometría a nivel diario, es decir, los ángulos y componentes que se pueden calcular en cada día independiente.

  Estas son:

  - Declinación (δ): calculada a partir de la función **declination**[1].
  - Excentricidad (ϵ_o): obtenida mediante la función **eccentricity**.
  - Ecuación del tiempo (EoT): obtenida mediante la función **eot**.
  - Ángulo del amanecer (ω_s): calculada a partir de la función **sunrise**.
  - Irradiancia diaria extra-atmosférica ($B_{0d}(0)$): obtenida a paritr de la función **bo0d**.

```r
lat <- 37.2
BTd <- fBTd(mode = 'prom')
solD <- fSolD(lat = lat, BTd = BTd)
show(solD)
```

```
Key: <Dates>
        Dates   lat        decl        eo           EoT        ws     Bo0d
       <IDat> <num>       <num>     <num>         <num>     <num>    <num>
 1: 2024-01-17  37.2 -0.36271754 1.0340422 -0.0455346238 -1.278593  4738.993
 2: 2024-02-14  37.2 -0.22850166 1.0259717 -0.0614793356 -1.393341  6137.388
```

[1] Todas las funciones mencionadas en este punto, se encuentran en el apartado ??.

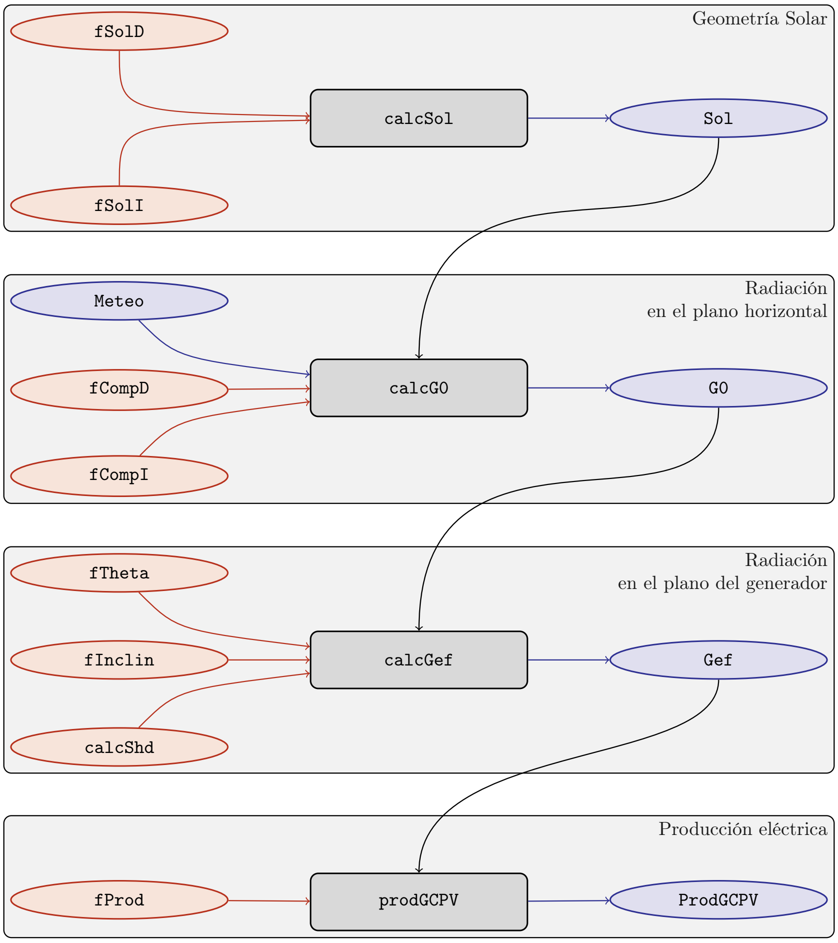


Figura 4.1: *Proceso de cálculo de las funciones de* ***solaR2***

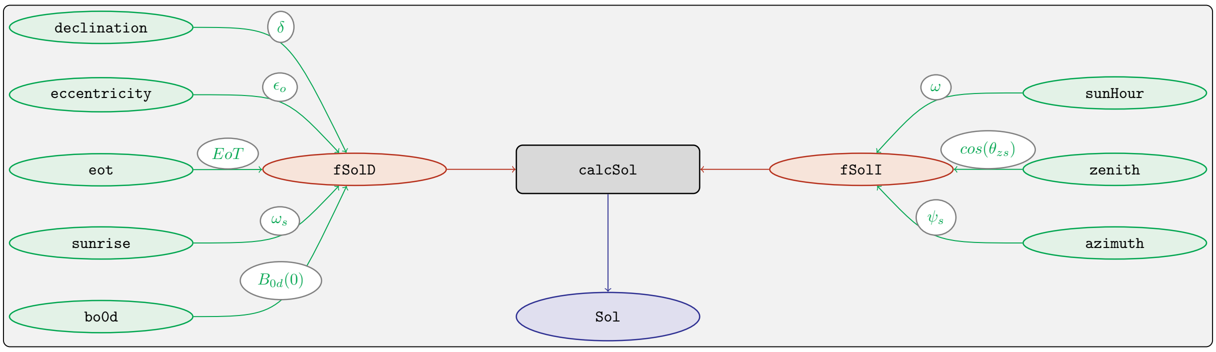


Figura 4.2: *Cálculo de la geometría solar mediante la función* ***calcSol****, la cual unifica las funciones* ***fSolD*** *y* ***fSolI*** *resultando en un objeto clase* ***Sol*** *el cual contiene toda la información geométrica necesaria para realizar las siguientes estimaciones.*

```
 3: 2024-03-15  37.2 -0.03191616 1.0107943 -0.0368674274 -1.546560  8086.323
 4: 2024-04-15  37.2  0.17531794 0.9926547  0.0017482721 -1.705659  9921.357
 5: 2024-05-15  37.2  0.33246485 0.9775162  0.0143055938 -1.835976 11115.619
 6: 2024-06-10  37.2  0.40257826 0.9691480 -0.0007378952 -1.899934 11573.907
 7: 2024-07-18  37.2  0.36439367 0.9675489 -0.0263454380 -1.864521 11257.133
 8: 2024-08-18  37.2  0.22407398 0.9758022 -0.0111761118 -1.744657 10183.208
 9: 2024-09-18  37.2  0.02730595 0.9907919  0.0342189964 -1.591529  8508.642
10: 2024-10-19  37.2 -0.17900474 1.0088406  0.0689613044 -1.433019  6554.218
11: 2024-11-18  37.2 -0.33862399 1.0245012  0.0575423573 -1.300179  4951.750
12: 2024-12-13  37.2 -0.40478283 1.0328516  0.0158622941 -1.239567  4284.472
```

Además, **fSolD** permite seleccionar el método de cáculo entre los propuestos por 4 autores diferentes (**cooper** [Coo69], **spencer** [Spe71], **strous** [Str11], **michalsky** [Mic88])(el valor por defecto es **michalsky**):

```r
solD_cooper <- fSolD(lat = lat, BTd = BTd, method = 'cooper')
show(solD_cooper)
```

```
Key: <Dates>
         Dates   lat        decl        eo           EoT        ws      Bo0d
        <IDat> <num>       <num>     <num>         <num>     <num>     <num>
 1: 2024-01-17  37.2 -0.36506987 1.0315970 -0.0455346238 -1.276457  4702.617
 2: 2024-02-14  37.2 -0.23770977 1.0235842 -0.0614793356 -1.385835  6024.833
 3: 2024-03-15  37.2 -0.04219743 1.0091112 -0.0368674274 -1.538742  7968.679
 4: 2024-04-15  37.2  0.17074888 0.9917107  0.0017482721 -1.702053  9870.335
 5: 2024-05-15  37.2  0.33214647 0.9770196  0.0143055938 -1.835696 11107.378
 6: 2024-06-10  37.2  0.40292516 0.9690335 -0.0007378952 -1.900263 11575.213
 7: 2024-07-18  37.2  0.36346384 0.9684861 -0.0263454380 -1.863677 11260.684
 8: 2024-08-18  37.2  0.21721704 0.9778484 -0.0111761118 -1.739110 10144.635
 9: 2024-09-18  37.2  0.01056696 0.9933706  0.0342189964 -1.578817  8367.014
10: 2024-10-19  37.2 -0.19902155 1.0107363  0.0689613044 -1.417100  6356.454
11: 2024-11-18  37.2 -0.34965673 1.0247443  0.0575423573 -1.290358  4835.353
12: 2024-12-13  37.2 -0.40651987 1.0315970  0.0158622941 -1.237915  4260.830
```

```r
solD_spencer <- fSolD(lat = lat, BTd = BTd, method = 'spencer')
show(solD_spencer)
```

```
Key: <Dates>
         Dates   lat        decl        eo           EoT        ws      Bo0d
        <IDat> <num>       <num>     <num>         <num>     <num>     <num>
 1: 2024-01-17  37.2 -0.36483670 1.0340422 -0.0455346238 -1.276669  4716.264
 2: 2024-02-14  37.2 -0.23199205 1.0259717 -0.0614793356 -1.390501  6100.057
 3: 2024-03-15  37.2 -0.03563921 1.0107943 -0.0368674274 -1.543730  8048.574
 4: 2024-04-15  37.2  0.17171286 0.9926547  0.0017482721 -1.702813  9888.522
 5: 2024-05-15  37.2  0.33007088 0.9775162  0.0143055938 -1.833871 11096.093
 6: 2024-06-10  37.2  0.40208757 0.9691480 -0.0007378952 -1.899469 11570.124
 7: 2024-07-18  37.2  0.36657157 0.9675489 -0.0263454380 -1.866501 11274.319
 8: 2024-08-18  37.2  0.22748717 0.9758022 -0.0111761118 -1.747427 10212.886
 9: 2024-09-18  37.2  0.03143967 0.9907919  0.0342189964 -1.594670  8548.821
10: 2024-10-19  37.2 -0.17549393 1.0088406  0.0689613044 -1.435795  6590.939
11: 2024-11-18  37.2 -0.33679169 1.0245012  0.0575423573 -1.301800  4971.285
12: 2024-12-13  37.2 -0.40419949 1.0328516  0.0158622941 -1.240121  4290.674
```

```r
solD_strous <- fSolD(lat = lat, BTd = BTd, method = 'cooper')
show(solD_strous)
```

```
Key: <Dates>
         Dates   lat        decl        eo           EoT        ws      Bo0d
        <IDat> <num>       <num>     <num>         <num>     <num>     <num>
 1: 2024-01-17  37.2 -0.36506987 1.0315970 -0.0455346238 -1.276457  4702.617
 2: 2024-02-14  37.2 -0.23770977 1.0235842 -0.0614793356 -1.385835  6024.833
 3: 2024-03-15  37.2 -0.04219743 1.0091112 -0.0368674274 -1.538742  7968.679
 4: 2024-04-15  37.2  0.17074888 0.9917107  0.0017482721 -1.702053  9870.335
 5: 2024-05-15  37.2  0.33214647 0.9770196  0.0143055938 -1.835696 11107.378
 6: 2024-06-10  37.2  0.40292516 0.9690335 -0.0007378952 -1.900263 11575.213
 7: 2024-07-18  37.2  0.36346384 0.9684861 -0.0263454380 -1.863677 11260.684
 8: 2024-08-18  37.2  0.21721704 0.9778484 -0.0111761118 -1.739110 10144.635
 9: 2024-09-18  37.2  0.01056696 0.9933706  0.0342189964 -1.578817  8367.014
10: 2024-10-19  37.2 -0.19902155 1.0107363  0.0689613044 -1.417100  6356.454
11: 2024-11-18  37.2 -0.34965673 1.0247443  0.0575423573 -1.290358  4835.353
12: 2024-12-13  37.2 -0.40651987 1.0315970  0.0158622941 -1.237915  4260.830
```

- **fSolI**: toma los resultados obtenidos en **fSolD** y calcula la geometría a nivel intradiario, es decir, aquella que se puede calcular en unidades de tiempo menores a los días. Estas son:
  - La hora solar o tiempo solar verdadero (ω): calculada a partir de la función **sunHour**.
  - Los momentos del día en los que es de noche ($night$): calculada a partir del resultado anterior y de el ángulo del amanecer (cálculada en **fSolD**)[2].
  - El coseno del ángulo cenital solar ($cos(\theta_{zs})$): obtenida a partir de la función **zenith**.
  - La altura solar (γ_s): obtenida a partir del resultado anterior[3].
  - El ángulo acimutal solar (θ_{zs}): calculada mediante la función **azimuth**.
  - La irradiancia extra-atmosférica ($B_0(0)$): calculada mediante el coseno del ángulo cenital, la constante solar (B_0) y la excentridad (cálculada en **fSolD**) [ecuación 3.2].

```r
solI <- fSolI(solD = solD[1], sample = 'hour') #Computo solo un día a fin
    mejorar la visualización
show(solI)
```

```
Index: <night>
                  Dates   lat           w  night       cosThzS          AlS         AzS       Bo0
                 <POSc> <num>       <num> <lgcl>         <num>        <num>       <num>     <num>
 1: 2024-01-17 00:00:00  37.2  3.09905026   TRUE -0.958552332 -1.281876984  3.00157749   0.00000
 2: 2024-01-17 01:00:00  37.2 -2.92239722   TRUE -0.941407376 -1.226779122 -2.49462689   0.00000
 3: 2024-01-17 02:00:00  37.2 -2.66065932   TRUE -0.874749489 -1.064918604 -2.03862388   0.00000
 4: 2024-01-17 03:00:00  37.2 -2.39892132   TRUE -0.763119126 -0.868125900 -1.77932134   0.00000
 5: 2024-01-17 04:00:00  37.2 -2.13718324   TRUE -0.614120126 -0.661270606 -1.59701536   0.00000
 6: 2024-01-17 05:00:00  37.2 -1.87544507   TRUE -0.437901763 -0.453263434 -1.44469585   0.00000
 7: 2024-01-17 06:00:00  37.2 -1.61370681   TRUE -0.246467423 -0.249033534 -1.30093496   0.00000
 8: 2024-01-17 07:00:00  37.2 -1.35196846   TRUE -0.052856976 -0.052881619 -1.15283370   0.00000
 9: 2024-01-17 08:00:00  37.2 -1.09023003  FALSE  0.129741461  0.130108233 -0.99014548 183.39419
10: 2024-01-17 09:00:00  37.2 -0.82849151  FALSE  0.288889848  0.293067041 -0.80329847 408.35612
11: 2024-01-17 10:00:00  37.2 -0.56675290  FALSE  0.413747472  0.426566560 -0.58400587 584.84684
12: 2024-01-17 11:00:00  37.2 -0.30501420  FALSE  0.495809380  0.518766586 -0.32921922 700.84427
13: 2024-01-17 12:00:00  37.2 -0.04327541  FALSE  0.529485721  0.557994217 -0.04769723 748.44699
14: 2024-01-17 13:00:00  37.2  0.21846346  FALSE  0.512482515  0.538073327  0.23821864 724.41235
15: 2024-01-17 14:00:00  37.2  0.48020243  FALSE  0.445957919  0.462244212  0.50355560 630.37745
16: 2024-01-17 15:00:00  37.2  0.74194148  FALSE  0.334443348  0.341014503  0.73469016 472.74762
```

[2] Cuando la hora solar verdadera excede los ángulos en los que amanece y anochece ($|\omega| >= |\omega_s|$), el Sol queda por debajo de la línea del horizonte, por lo que es de noche.

[3] $\gamma_s = asin(cos(\theta_s))$.

```
17: 2024-01-17 16:00:00  37.2  1.00368062  FALSE  0.185534810  0.186616094  0.93148844 262.26008
18: 2024-01-17 17:00:00  37.2  1.26541985  FALSE  0.009375501  0.009375638  1.10112996  13.25261
19: 2024-01-17 18:00:00  37.2  1.52715917   TRUE -0.182035120 -0.183055757  1.25297092   0.00000
20: 2024-01-17 19:00:00  37.2  1.78889857   TRUE -0.375658695 -0.385107424  1.39694027   0.00000
21: 2024-01-17 20:00:00  37.2  2.05063807   TRUE -0.558306105 -0.592342658  1.54466726   0.00000
22: 2024-01-17 21:00:00  37.2  2.31237766   TRUE -0.717535874 -0.800258081  1.71368519   0.00000
23: 2024-01-17 22:00:00  37.2  2.57411733   TRUE -0.842501657 -1.001910427  1.93928567   0.00000
24: 2024-01-17 23:00:00  37.2  2.83585709   TRUE -0.924691065 -1.180223341  2.30977400   0.00000
                  Dates   lat           w  night       cosThzS          AlS         AzS        Bo0
```

Además, como los datos nocturnos aportan poco a los cálculos que atañen a este proyecto, **`fSolI`** presenta la posibilidad de eliminar estos datos con el argumento **`keep.night`**.

```
solI_nigth <- fSolI(solD = solD[1], sample = 'hour', keep.night = FALSE)
show(solI_nigth)
```

```
                  Dates   lat           w  night     cosThzS          AlS         AzS        Bo0
                 <POSc> <num>       <num> <lgcl>       <num>        <num>       <num>      <num>
 1: 2024-01-17 08:00:00  37.2 -1.09023003  FALSE 0.129741461 0.130108233 -0.99014548 183.39419
 2: 2024-01-17 09:00:00  37.2 -0.82849151  FALSE 0.288889848 0.293067041 -0.80329847 408.35612
 3: 2024-01-17 10:00:00  37.2 -0.56675290  FALSE 0.413747472 0.426566560 -0.58400587 584.84684
 4: 2024-01-17 11:00:00  37.2 -0.30501420  FALSE 0.495809380 0.518766586 -0.32921922 700.84427
 5: 2024-01-17 12:00:00  37.2 -0.04327541  FALSE 0.529485721 0.557994217 -0.04769723 748.44699
 6: 2024-01-17 13:00:00  37.2  0.21846346  FALSE 0.512482515 0.538073327  0.23821864 724.41235
 7: 2024-01-17 14:00:00  37.2  0.48020243  FALSE 0.445957919 0.462244212  0.50355560 630.37745
 8: 2024-01-17 15:00:00  37.2  0.74194148  FALSE 0.334443348 0.341014503  0.73469016 472.74762
 9: 2024-01-17 16:00:00  37.2  1.00368062  FALSE 0.185534810 0.186616094  0.93148844 262.26008
10: 2024-01-17 17:00:00  37.2  1.26541985  FALSE 0.009375501 0.009375638  1.10112996  13.25261
```

Aparte, en vez de identificar el intervalo intradiario (con el argumento **`sample`**), se puede dar directamente la base temporal intradiaria.

```
BTi <- fBTi(BTd, sample = 'hour')
solI_BTi <- fSolI(solD, BTi = BTi)
show(solI_BTi)
```

```
Index: <night>
                   Dates   lat         w  night    cosThzS        AlS        AzS   Bo0
                  <POSc> <num>     <num> <lgcl>      <num>      <num>      <num> <num>
  1: 2024-01-17 00:00:00  37.2  3.099050   TRUE -0.9585523 -1.2818770  3.001577     0
  2: 2024-01-17 01:00:00  37.2 -2.922397   TRUE -0.9414074 -1.2267791 -2.494627     0
  3: 2024-01-17 02:00:00  37.2 -2.660659   TRUE -0.8747495 -1.0649186 -2.038624     0
  4: 2024-01-17 03:00:00  37.2 -2.398921   TRUE -0.7631191 -0.8681259 -1.779321     0
  5: 2024-01-17 04:00:00  37.2 -2.137183   TRUE -0.6141201 -0.6612706 -1.597015     0
 ---
284: 2024-12-13 19:00:00  37.2  1.856445   TRUE -0.4444110 -0.4605166  1.394524     0
285: 2024-12-13 20:00:00  37.2  2.118158   TRUE -0.6191456 -0.6676542  1.539641     0
286: 2024-12-13 21:00:00  37.2  2.379871   TRUE -0.7679298 -0.8756029  1.709361     0
287: 2024-12-13 22:00:00  37.2  2.641583   TRUE -0.8806309 -1.0771921  1.946876     0
288: 2024-12-13 23:00:00  37.2  2.903296   TRUE -0.9495736 -1.2518732  2.377338     0
```

También, se puede indicar que no realice las correcciones de la ecuación del tiempo.

```
solI_EoT <- fSolI(solD = solD, BTi = BTi, EoT = FALSE)
show(solI_EoT)
```

```
Index: <night>
                   Dates   lat          w  night    cosThzS        AlS       AzS   Bo0
                  <POSc> <num>      <num> <lgcl>      <num>      <num>     <num> <num>
  1: 2024-01-17 00:00:00  37.2   3.099050   TRUE -0.9585523 -1.2818770  3.001577     0
  2: 2024-01-17 01:00:00  37.2 -2.922397   TRUE -0.9414074 -1.2267791 -2.494627     0
  3: 2024-01-17 02:00:00  37.2 -2.660659   TRUE -0.8747495 -1.0649186 -2.038624     0
  4: 2024-01-17 03:00:00  37.2 -2.398921   TRUE -0.7631191 -0.8681259 -1.779321     0
  5: 2024-01-17 04:00:00  37.2 -2.137183   TRUE -0.6141201 -0.6612706 -1.597015     0
 ---
284: 2024-12-13 19:00:00  37.2  1.856445   TRUE -0.4444110 -0.4605166  1.394524     0
285: 2024-12-13 20:00:00  37.2  2.118158   TRUE -0.6191456 -0.6676542  1.539641     0
286: 2024-12-13 21:00:00  37.2  2.379871   TRUE -0.7679298 -0.8756029  1.709361     0
287: 2024-12-13 22:00:00  37.2  2.641583   TRUE -0.8806309 -1.0771921  1.946876     0
288: 2024-12-13 23:00:00  37.2  2.903296   TRUE -0.9495736 -1.2518732  2.377338     0
```

Finalmente, estas dos funciones, como se muestra en la figura 4.2, convergen en la función **calcSol**, dando como resultado un objeto de clase **Sol**. Este objeto muestra un resumen de ambos elementos junto con la latitud de los cálculos.

```
sol <- calcSol(lat = lat, BTd = BTd, sample = 'hour')
show(sol)
```

```
Object of class Sol

Latitude:  37.2 degrees

Daily values:
     Dates                decl                 eo               EoT                    ws
 Min.   :2024-01-17   Min.   :-0.404783   Min.   :0.9675   Min.   :-0.0614793   Min.   :-1.900
 1st Qu.:2024-04-07   1st Qu.:-0.256032   1st Qu.:0.9771   1st Qu.:-0.0289759   1st Qu.:-1.767
 Median :2024-06-29   Median :-0.002305   Median :1.0007   Median : 0.0005052   Median :-1.569
 Mean   :2024-07-01   Mean   :-0.001618   Mean   :1.0009   Mean   : 0.0008748   Mean   :-1.569
 3rd Qu.:2024-09-25   3rd Qu.: 0.251172   3rd Qu.:1.0249   3rd Qu.: 0.0204515   3rd Qu.:-1.370
 Max.   :2024-12-13   Max.   : 0.402578   Max.   :1.0340   Max.   : 0.0689613   Max.   :-1.240
      Bo0d
 Min.   : 4284
 1st Qu.: 5841
 Median : 8297
 Mean   : 8109
 3rd Qu.:10416
 Max.   :11574

Intradaily values:
     Dates                            w                   night              cosThzS
 Min.   :2024-01-17 00:00:00   Min.   :-3.1393050   Mode :logical   Min.   :-0.9700256
 1st Qu.:2024-04-07 11:45:00   1st Qu.:-1.5692285   FALSE:145       1st Qu.:-0.5004531
 Median :2024-06-29 11:30:00   Median : 0.0010871   TRUE :143       Median : 0.0062923
 Mean   :2024-07-01 15:30:00   Mean   : 0.0009975                   Mean   :-0.0009523
 3rd Qu.:2024-09-26 11:15:00   3rd Qu.: 1.5716412                   3rd Qu.: 0.5007129
 Max.   :2024-12-13 23:00:00   Max.   : 3.1413972                   Max.   : 0.9697262
      AlS                 AzS                 Bo0
 Min.   :-1.325336   Min.   :-3.139169   Min.   :   0.000
 1st Qu.:-0.524130   1st Qu.:-1.570722   1st Qu.:   0.000
 Median : 0.006292   Median : 0.003834   Median :   8.748
 Mean   :-0.001202   Mean   : 0.001011   Mean   : 337.752
 3rd Qu.: 0.524433   3rd Qu.: 1.555342   3rd Qu.: 698.153
 Max.   : 1.324107   Max.   : 3.141331   Max.   :1284.718
```

4.2. Datos meteorológicos

Para el procesamiento de datos meteorologicos, **solaR2** provee una serie de funciones[4] que son capaces de leer todo tipo de datos. Estos datos se procesan y se almacenan en un objeto de tipo **Meteo** tal y como se ve en la figura 4.3. Estas funciones son:

- **readG0dm**: Esta función construye un objeto **Meteo** a partir de 12 valores de medias mensuales de irradiación.

```r
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)
show(BD)
```

```
Object of class  Meteo

Source of meteorological information: prom-
Latitude of source:  37.2 degrees

Meteorological Data:
     Dates                  G0d            Ta
 Min.   :2024-01-17   Min.   :2179   Min.   :10.00
 1st Qu.:2024-04-07   1st Qu.:3322   1st Qu.:15.50
 Median :2024-06-29   Median :4932   Median :17.70
 Mean   :2024-07-01   Mean   :5022   Mean   :19.22
 3rd Qu.:2024-09-25   3rd Qu.:6998   3rd Qu.:21.98
 Max.   :2024-12-13   Max.   :7919   Max.   :29.90
```

- **readBD**: Esta familia de funciones puede leer ficheros de datos y transformarlos en un objeto de clase **Meteo**. Se dividen en:
  - **readBDd**: Procesa datos meteorológicos de tipo diarios.

```r
## Se utiliza un archivo alojado en el
## github del tutor de este proyecto
myURL <-"https://raw.githubusercontent.com/oscarperpinan/R/master/data/
    aranjuez.csv"
```

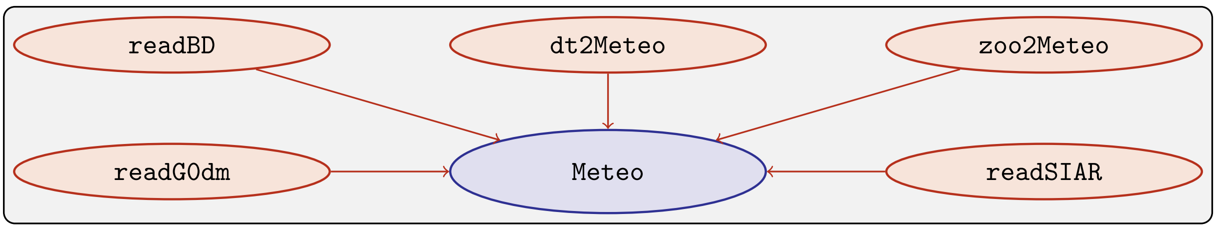


FIGURA 4.3: *Los datos meteorologicas se pueden leer mediante las funciones* **readG0dm**, **readBD**, **dt2Meteo**, **zoo2Meteo** *y* **readSIAR** *las cuales procesan estos datos y los almacenan en un objeto de clase* **Meteo**.

[4]Las funciones comentadas en este apartado, se recogen en la sección **??**

```
download.file(myURL, 'data/aranjuez.csv', quiet = TRUE)
BDd <- readBDd(file = 'data/aranjuez.csv', lat = lat,
               format = '%Y-%m-%d', header = TRUE,
               fill = TRUE, dec = '.', sep = ',', dates.col = '',
               ta.col = 'TempAvg', g0.col = 'Radiation', keep.cols = TRUE)
show(BDd)
```

```
Object of class  Meteo

Source of meteorological information: bd-data/aranjuez.csv
Latitude of source:  37.2 degrees

Meteorological Data:
     Dates                      G0                Ta              TempMin            TempMax
 Min.   :2004-01-01   Min.   : 0.277   Min.   :-5.309   Min.   :-12.980   Min.   :-2.362
 1st Qu.:2005-12-29   1st Qu.: 9.370   1st Qu.: 7.692   1st Qu.:  1.515   1st Qu.:14.530
 Median :2008-01-09   Median :16.660   Median :13.810   Median :  7.170   Median :21.670
 Mean   :2008-01-03   Mean   :16.742   Mean   :14.405   Mean   :  6.888   Mean   :22.531
 3rd Qu.:2010-01-02   3rd Qu.:24.650   3rd Qu.:21.615   3rd Qu.: 12.590   3rd Qu.:30.875
 Max.   :2011-12-31   Max.   :32.740   Max.   :30.680   Max.   : 22.710   Max.   :41.910
                      NA's   :13                        NA's   :4
    HumidAvg          HumidMax          WindAvg          WindMax             Rain              ET
 Min.   : 19.89   Min.   : 35.88   Min.   :0.251   Min.   : 0.000   Min.   : 0.000   Min.   :0.000
 1st Qu.: 47.04   1st Qu.: 81.60   1st Qu.:0.667   1st Qu.: 3.783   1st Qu.: 0.000   1st Qu.:1.168
 Median : 62.58   Median : 90.90   Median :0.920   Median : 5.027   Median : 0.000   Median :2.758
 Mean   : 62.16   Mean   : 87.22   Mean   :1.174   Mean   : 5.208   Mean   : 1.094   Mean   :3.091
 3rd Qu.: 77.38   3rd Qu.: 94.90   3rd Qu.:1.431   3rd Qu.: 6.537   3rd Qu.: 0.200   3rd Qu.:4.926
 Max.   :100.00   Max.   :100.00   Max.   :8.260   Max.   :10.000   Max.   :49.730   Max.   :8.564
                  NA's   :13       NA's   :8       NA's   :128      NA's   :4        NA's   :18
```

- **`readBDi`**: Procesa datos meteorológicos de tipo intradiarios.

```
myURL <- "https://raw.githubusercontent.com/oscarperpinan/R/master/data/
    NREL-Hawaii.csv"
download.file(myURL, 'data/NREL-Hawaii.csv', quiet = TRUE)
BDi <- readBDi(file = 'data/NREL-Hawaii.csv', lat = 19,
               format = "%d/%m/%Y %H:%M", header = TRUE,
               fill = TRUE, dec = '.', sep = ',',
               dates.col = 'DATE', times.col = 'HST',
               ta.col = 'Air Temperature [deg C]',
               g0.col = 'Global Horizontal [W/m^2]',
               keep.cols = TRUE)
show(BDi)
```

```
Object of class  Meteo

Source of meteorological information: bdI-data/NREL-Hawaii.csv
Latitude of source:  19 degrees

Meteorological Data:
     Dates                                G0                  Ta           Direct Normal [W/m^2]
 Min.   :2010-01-11 06:32:00.00   Min.   :   0.4769   Min.   :13.42   Min.   :  0.0
 1st Qu.:2010-03-11 17:37:45.00   1st Qu.: 147.4328   1st Qu.:22.76   1st Qu.:  0.0
 Median :2010-06-11 17:32:30.00   Median : 300.6510   Median :24.15   Median :270.3
 Mean   :2010-06-26 11:55:22.63   Mean   : 370.5293   Mean   :23.64   Mean   :356.6
 3rd Qu.:2010-09-11 17:34:15.00   3rd Qu.: 585.7402   3rd Qu.:25.24   3rd Qu.:715.2
 Max.   :2010-12-11 17:46:00.00   Max.   :1172.3000   Max.   :28.12   Max.   :943.0
 NA's   :4660
 Diffuse Horizontal [W/m^2]
 Min.   :  0.4769
 1st Qu.: 78.4636
 Median :152.9320
 Mean   :171.7706
```

```
3rd Qu.:246.3193
Max.   :586.3600
```

- **dt2Meteo**: Transforma un **data.table** o **data.frame** en un objeto de clase **Meteo**.

```
data(helios)
names(helios) <- c('Dates', 'G0d', 'TempMax', 'TempMin')
helios_meteo <- dt2Meteo(file = helios, lat = 40, type = 'bd')
show(helios_meteo)
```

```
Object of class  Meteo

Source of meteorological information: bd-data.frame
Latitude of source:  40 degrees

Meteorological Data:
     Dates                          G0d              TempMin            TempMax
 Min.   :2009-01-01 00:00:00.00   Min.   :  325.6   Min.   :-37.500   Min.   : 1.41
 1st Qu.:2009-04-08 12:00:00.00   1st Qu.: 2523.2   1st Qu.:  1.950   1st Qu.:14.41
 Median :2009-07-07 00:00:00.00   Median : 4745.7   Median :  7.910   Median :23.16
 Mean   :2009-07-04 21:29:54.93   Mean   : 4812.0   Mean   :  5.323   Mean   :22.59
 3rd Qu.:2009-10-03 12:00:00.00   3rd Qu.: 7139.5   3rd Qu.: 15.105   3rd Qu.:31.06
 Max.   :2009-12-31 00:00:00.00   Max.   :11253.9   Max.   : 24.800   Max.   :38.04
       Ta
 Min.   :-23.049
 1st Qu.:  7.008
 Median : 12.055
 Mean   : 10.944
 3rd Qu.: 19.472
 Max.   : 28.619
```

- **zoo2Meteo**: Transforma un objeto de clase **zoo**[5] en un objeto de clase **Meteo**.

```
library(zoo)
bd_zoo <- read.csv.zoo('data/aranjuez.csv')
BD_zoo <- zoo2Meteo(file = bd_zoo, lat = 40)
show(BD_zoo)
```

```
Object of class  Meteo

Source of meteorological information: bd-zoo-bd_zoo
Latitude of source:  40 degrees

Meteorological Data:
    TempAvg          TempMax          TempMin           HumidAvg         HumidMax          WindAvg
 Min.   :-5.309   Min.   :-2.362   Min.   :-12.980   Min.   : 19.89   Min.   : 35.88   Min.   :0.251
 1st Qu.: 7.692   1st Qu.:14.530   1st Qu.:  1.515   1st Qu.: 47.04   1st Qu.: 81.60   1st Qu.:0.667
 Median :13.810   Median :21.670   Median :  7.170   Median : 62.58   Median : 90.90   Median :0.920
 Mean   :14.405   Mean   :22.531   Mean   :  6.888   Mean   : 62.16   Mean   : 87.22   Mean   :1.174
 3rd Qu.:21.615   3rd Qu.:30.875   3rd Qu.: 12.590   3rd Qu.: 77.38   3rd Qu.: 94.90   3rd Qu.:1.431
 Max.   :30.680   Max.   :41.910   Max.   : 22.710   Max.   :100.00   Max.   :100.00   Max.   :8.260
                                   NA's   :4                           NA's   :13       NA's   :8
    WindMax            Rain           Radiation            ET
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.277   Min.   :0.000
```

[5]Pese a que este proyecto trate de "desligarse" del paquete **zoo**, sigue siendo un paquete muy extendido. Por lo que es interesante tener una función así para que los usuarios tengan una mayor flexibilidad.

```
1st Qu.: 3.783   1st Qu.: 0.000   1st Qu.: 9.370   1st Qu.:1.168
Median : 5.027   Median : 0.000   Median :16.660   Median :2.758
Mean   : 5.208   Mean   : 1.094   Mean   :16.742   Mean   :3.091
3rd Qu.: 6.537   3rd Qu.: 0.200   3rd Qu.:24.650   3rd Qu.:4.926
Max.   :10.000   Max.   :49.730   Max.   :32.740   Max.   :8.564
NA's   :128      NA's   :4        NA's   :13       NA's   :18
```

- **readSIAR**: Esta función es capaz de extraer información de la red SIAR y transformarlo en un objeto de clase **Meteo**.

```
library(httr2)
library(jsonlite)
bd_SIAR <- readSIAR(Lat = 40.40596822621351, Lon = -3.70038308516172,
                    ## Ubicación de la Escuela Técnica Superior
                    ## de Ingeniería y Diseño Industrial (ETSIDI)
                    inicio = '2023-09-01', final = '2024-08-01',
                    tipo = 'Mensuales', n_est = 3)
show(bd_SIAR)
```

```
Object of class  Meteo

Source of meteorological information: prom-https://servicio.mapama.gob.es
  -Estaciones: Center: Finca experimental(M01), Arganda(M02), San Martín de la Vega(M05)
Latitude of source:  40.4 degrees

Meteorological Data:
     Dates                         G0d            Ta              TempMin            TempMax
 Min.   :2023-09-18 00:00:00   Min.   :1860   Min.   : 5.318   Min.   :-4.6513   Min.   :15.34
 1st Qu.:2023-12-06 18:00:00   1st Qu.:2744   1st Qu.: 9.857   1st Qu.:-2.1466   1st Qu.:21.12
 Median :2024-02-29 00:00:00   Median :4052   Median :14.890   Median : 0.3663   Median :31.01
 Mean   :2024-03-01 04:00:00   Mean   :4502   Mean   :15.307   Mean   : 2.4225   Mean   :29.41
 3rd Qu.:2024-05-21 12:00:00   3rd Qu.:6549   3rd Qu.:20.047   3rd Qu.: 7.1506   3rd Qu.:35.47
 Max.   :2024-08-18 00:00:00   Max.   :7608   Max.   :27.069   Max.   :12.6082   Max.   :40.70
```

Esta función tiene dos argumentos importantes:

- **tipo**: La API SIAR[6] permite tener 4 tipos de registros: **Mensuales**, **Semanales**, **Diarios** y **Horarios**.
- **n_est**: Con este argumento, la función es capaz de localizar el número seleccionado de estaciones más proximas a la ubicación dada, y obtener los datos individuales de cada una de ellas. Una vez obtenidos estos datos realiza una interpolación de distancia inversa ponderada (IDW) y entrega un solo resultado. Es importante añadir que la API SIAR tiene una limitación a la solicitud de registros que se le hace cada minuto, por lo que esta función cuenta con un comprobante para impedir que el usuario exceda este límite.

4.3. Radiación en el plano horizontal

Una vez se ha calculado la geometría solar (sección 4.1) y se han procesado los datos meteorológicos (sección 4.2), es necesario calcular la radiación en el plano horizontal. Para ello,

[6] La API (Interfaz de Programación de Aplicaciones) que se usa para la función **readSIAR** está proporcionada por la propia red SIAR [Min23].

solaR2 cuenta con la función **calcG0** [??] la cual mediante las funciones **fCompD** [??] y **fCompI** [??] procesan los objetos de clase **Sol** y clase **Meteo** para dar un objeto de tipo **G0**.

Como se puede ver en la figura 4.4, **calcG0** funciona gracias a las siguientes funciones:

- **fCompD**: La cual calcula todas las componentes de la irradiación diaria en una superficie horizontal mediante regresiones entre los parámetros del índice de claridad y la fracción difusa. Para ello se pueden usar varias correlaciones dependiendo del tipo de datos:
    - Mensuales:

```r
lat <- 37.2
BTd <- fBTd(mode = 'prom')
solD <- fSolD(lat, BTd)
G0d <- 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
compD_page <- fCompD(sol = solD, G0d = G0d, corr = "Page")
compD_page
```

```
Key: <Dates>
         Dates        Fd        Kt   G0d      D0d      B0d
        <POSc>     <num>     <num> <num>    <num>    <num>
 1: 2024-01-17 0.3404548 0.5836683  2766  941.698 1824.302
 2: 2024-02-14 0.3572461 0.5688088  3491 1247.146 2243.854
 3: 2024-03-15 0.3719989 0.5557532  4494 1671.763 2822.237
 4: 2024-04-15 0.3266485 0.5958862  5912 1931.146 3980.854
 5: 2024-05-15 0.2895069 0.6287549  6989 2023.364 4965.636
 6: 2024-06-10 0.2441221 0.6689185  7742 1889.994 5852.006
 7: 2024-07-18 0.2050844 0.7034651  7919 1624.064 6294.936
 8: 2024-08-18 0.2202349 0.6900576  7027 1547.591 5479.409
 9: 2024-09-18 0.2869638 0.6310055  5369 1540.708 3828.292
10: 2024-10-19 0.3858825 0.5434669  3562 1374.513 2187.487
11: 2024-11-18 0.3578392 0.5682839  2814 1006.959 1807.041
12: 2024-12-13 0.4253038 0.5085807  2179  926.737 1252.263
```

```r
compD_lj <- fCompD(sol = solD, G0d = G0d, corr = "LJ")
compD_lj
```

```
Key: <Dates>
         Dates        Fd        Kt   G0d       D0d      B0d
        <POSc>     <num>     <num> <num>     <num>    <num>
 1: 2024-01-17 0.3058193 0.5836683  2766  845.8961 1920.104
```

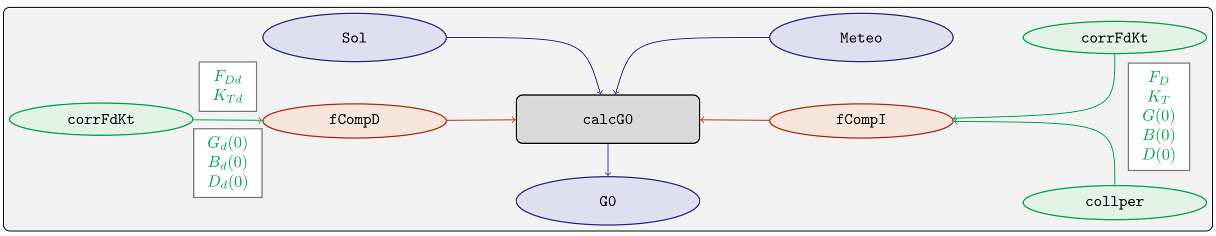


Figura 4.4: *Cálculo de la radiación incidente en el plano horizontal mediante la función **calcG0**, la cual procesa un objeto clase **Sol** y otro clase **Meteo** mediante las funciones **fCompD** y **fCompI** resultando en un objeto clase **G0**. :*

```
 2: 2024-02-14 0.3169470 0.5688088  3491 1106.4621 2384.538
 3: 2024-03-15 0.3268047 0.5557532  4494 1468.6603 3025.340
 4: 2024-04-15 0.2967018 0.5958862  5912 1754.1011 4157.899
 5: 2024-05-15 0.2720419 0.6287549  6989 1901.3006 5087.699
 6: 2024-06-10 0.2408700 0.6689185  7742 1864.8154 5877.185
 7: 2024-07-18 0.2152460 0.7034651  7919 1704.5331 6214.467
 8: 2024-08-18 0.2236251 0.6900576  7027 1571.4138 5455.586
 9: 2024-09-18 0.2703347 0.6310055  5369 1451.4268 3917.573
10: 2024-10-19 0.3361895 0.5434669  3562 1197.5071 2364.493
11: 2024-11-18 0.3173415 0.5682839  2814  892.9990 1921.001
12: 2024-12-13 0.3637158 0.5085807  2179  792.5367 1386.463
```

- Diarios:

```r
G0d <- readSIAR(Lat = 40.40596822621351, Lon =-3.70038308516172,
                inicio = '2024-07-15', final = '2024-08-01',
                tipo = 'Diarios', n_est = 3)
sol <- calcSol(lat, BTd = indexD(G0d))
compD_cpr <- fCompD(sol = sol, G0d = G0d, corr = "CPR")
compD_cpr
```

```
Key: <Dates>
         Dates        Fd        Kt      G0d      D0d      B0d
        <POSc>     <num>     <num>    <num>    <num>    <num>
 1: 2024-07-15 0.2833125 0.6798139 7697.945 2180.924 5517.021
 2: 2024-07-16 0.2597185 0.7000272 7911.858 2054.856 5857.002
 3: 2024-07-17 0.2815044 0.6812283 7684.293 2163.163 5521.131
 4: 2024-07-18 0.6627754 0.4674993 5262.702 3487.989 1774.713
 5: 2024-07-19 0.2595844 0.7001561 7865.166 2041.675 5823.491
 6: 2024-07-20 0.2594075 0.7003266 7849.961 2036.339 5813.622
 7: 2024-07-21 0.2315068 0.7365959 8237.938 1907.138 6330.799
 8: 2024-07-22 0.2269337 0.7493438 8361.056 1897.406 6463.650
 9: 2024-07-23 0.2451723 0.7156288 7965.753 1952.982 6012.771
10: 2024-07-24 0.2620008 0.6978638 7748.845 2030.204 5718.641
11: 2024-07-25 0.2746548 0.6867564 7606.140 2089.063 5517.077
12: 2024-07-26 0.3320728 0.6462270 7138.548 2370.518 4768.030
13: 2024-07-27 0.3186769 0.6547900 7213.697 2298.839 4914.858
14: 2024-07-28 0.2767163 0.6850625 7526.355 2082.665 5443.689
15: 2024-07-29 0.6566999 0.4709412 5159.260 3388.086 1771.174
16: 2024-07-30 0.3185533 0.6548709 7153.359 2278.726 4874.633
17: 2024-07-31 0.2503814 0.7096003 7728.034 1934.956 5793.078
18: 2024-08-01 0.2428514 0.7185406 7801.435 1894.589 5906.846
```

```r
compD_ekdd <- fCompD(sol = sol, G0d = G0d, corr = 'EKDd')
compD_ekdd
```

```
Key: <Dates>
         Dates    Fd        Kt      G0d      D0d   B0d
        <POSc> <num>     <num>    <num>    <num> <num>
 1: 2024-07-15     1 0.6798139 7697.945 7697.945     0
 2: 2024-07-16     1 0.7000272 7911.858 7911.858     0
 3: 2024-07-17     1 0.6812283 7684.293 7684.293     0
 4: 2024-07-18     1 0.4674993 5262.702 5262.702     0
 5: 2024-07-19     1 0.7001561 7865.166 7865.166     0
 6: 2024-07-20     1 0.7003266 7849.961 7849.961     0
 7: 2024-07-21     1 0.7365959 8237.938 8237.938     0
 8: 2024-07-22     1 0.7493438 8361.056 8361.056     0
 9: 2024-07-23     1 0.7156288 7965.753 7965.753     0
10: 2024-07-24     1 0.6978638 7748.845 7748.845     0
11: 2024-07-25     1 0.6867564 7606.140 7606.140     0
12: 2024-07-26     1 0.6462270 7138.548 7138.548     0
13: 2024-07-27     1 0.6547900 7213.697 7213.697     0
14: 2024-07-28     1 0.6850625 7526.355 7526.355     0
```

```
15: 2024-07-29     1 0.4709412 5159.260 5159.260     0
16: 2024-07-30     1 0.6548709 7153.359 7153.359     0
17: 2024-07-31     1 0.7096003 7728.034 7728.034     0
18: 2024-08-01     1 0.7185406 7801.435 7801.435     0
```

```
compD_climedd <- fCompD(sol = sol, G0d = G0d, corr = 'CLIMEDd')
compD_climedd
```

```
Key: <Dates>
         Dates        Fd        Kt      G0d      D0d      B0d
        <POSc>     <num>     <num>    <num>    <num>    <num>
 1: 2024-07-15 0.2724591 0.6798139 7697.945 2097.375 5600.570
 2: 2024-07-16 0.2455880 0.7000272 7911.858 1943.057 5968.801
 3: 2024-07-17 0.2705287 0.6812283 7684.293 2078.822 5605.472
 4: 2024-07-18 0.6086148 0.4674993 5262.702 3202.958 2059.744
 5: 2024-07-19 0.2454217 0.7001561 7865.166 1930.282 5934.884
 6: 2024-07-20 0.2452020 0.7003266 7849.961 1924.826 5925.135
 7: 2024-07-21 0.2013208 0.7365959 8237.938 1658.468 6579.470
 8: 2024-07-22 0.1873678 0.7493438 8361.056 1566.592 6794.463
 9: 2024-07-23 0.2259736 0.7156288 7965.753 1800.050 6165.703
10: 2024-07-24 0.2483878 0.6978638 7748.845 1924.718 5824.126
11: 2024-07-25 0.2630540 0.6867564 7606.140 2000.826 5605.314
12: 2024-07-26 0.3202837 0.6462270 7138.548 2286.361 4852.187
13: 2024-07-27 0.3077503 0.6547900 7213.697 2220.018 4993.679
14: 2024-07-28 0.2653324 0.6850625 7526.355 1996.986 5529.369
15: 2024-07-29 0.6029930 0.4709412 5159.260 3110.998 2048.263
16: 2024-07-30 0.3076331 0.6548709 7153.359 2200.610 4952.749
17: 2024-07-31 0.2334298 0.7096003 7728.034 1803.954 5924.080
18: 2024-08-01 0.2224291 0.7185406 7801.435 1735.266 6066.168
```

También, se puede aportar una función de correlación propia.

```
f_corrd <- function(sol, G0d){
  ## Función CLIMEDd
    Kt <- Ktd(sol, G0d)
    Fd=(Kt<=0.13)*(0.952)+
    (Kt>0.13 & Kt<=0.8)*(0.868+1.335*Kt-5.782*Kt^2+3.721*Kt^3)+
      (Kt>0.8)*0.141
  return(data.table(Fd, Kt))
}
compD_user <- fCompD(sol = sol, G0d = G0d, corr = 'user', f = f_corrd)
compD_user
```

```
Key: <Dates>
         Dates        Fd        Kt      G0d      D0d      B0d
        <POSc>     <num>     <num>    <num>    <num>    <num>
 1: 2024-07-15 0.2724591 0.6798139 7697.945 2097.375 5600.570
 2: 2024-07-16 0.2455880 0.7000272 7911.858 1943.057 5968.801
 3: 2024-07-17 0.2705287 0.6812283 7684.293 2078.822 5605.472
 4: 2024-07-18 0.6086148 0.4674993 5262.702 3202.958 2059.744
 5: 2024-07-19 0.2454217 0.7001561 7865.166 1930.282 5934.884
 6: 2024-07-20 0.2452020 0.7003266 7849.961 1924.826 5925.135
 7: 2024-07-21 0.2013208 0.7365959 8237.938 1658.468 6579.470
 8: 2024-07-22 0.1873678 0.7493438 8361.056 1566.592 6794.463
 9: 2024-07-23 0.2259736 0.7156288 7965.753 1800.050 6165.703
10: 2024-07-24 0.2483878 0.6978638 7748.845 1924.718 5824.126
11: 2024-07-25 0.2630540 0.6867564 7606.140 2000.826 5605.314
12: 2024-07-26 0.3202837 0.6462270 7138.548 2286.361 4852.187
13: 2024-07-27 0.3077503 0.6547900 7213.697 2220.018 4993.679
14: 2024-07-28 0.2653324 0.6850625 7526.355 1996.986 5529.369
15: 2024-07-29 0.6029930 0.4709412 5159.260 3110.998 2048.263
16: 2024-07-30 0.3076331 0.6548709 7153.359 2200.610 4952.749
17: 2024-07-31 0.2334298 0.7096003 7728.034 1803.954 5924.080
18: 2024-08-01 0.2224291 0.7185406 7801.435 1735.266 6066.168
```

Por último, si **G0d** ya contiene todos los componentes, se puede especifica que no haga ninguna correlación.

```
compD_none <- fCompD(sol = sol, G0d = compD_user, corr = 'none')
compD_none
```

```
Key: <Dates>
         Dates        Fd        Kt      G0d      D0d      B0d
        <POSc>     <num>     <num>    <num>    <num>    <num>
 1: 2024-07-15 0.2724591 0.6798139 7697.945 2097.375 5600.570
 2: 2024-07-16 0.2455880 0.7000272 7911.858 1943.057 5968.801
 3: 2024-07-17 0.2705287 0.6812283 7684.293 2078.822 5605.472
 4: 2024-07-18 0.6086148 0.4674993 5262.702 3202.958 2059.744
 5: 2024-07-19 0.2454217 0.7001561 7865.166 1930.282 5934.884
 6: 2024-07-20 0.2452020 0.7003266 7849.961 1924.826 5925.135
 7: 2024-07-21 0.2013208 0.7365959 8237.938 1658.468 6579.470
 8: 2024-07-22 0.1873678 0.7493438 8361.056 1566.592 6794.463
 9: 2024-07-23 0.2259736 0.7156288 7965.753 1800.050 6165.703
10: 2024-07-24 0.2483878 0.6978638 7748.845 1924.718 5824.126
11: 2024-07-25 0.2630540 0.6867564 7606.140 2000.826 5605.314
12: 2024-07-26 0.3202837 0.6462270 7138.548 2286.361 4852.187
13: 2024-07-27 0.3077503 0.6547900 7213.697 2220.018 4993.679
14: 2024-07-28 0.2653324 0.6850625 7526.355 1996.986 5529.369
15: 2024-07-29 0.6029930 0.4709412 5159.260 3110.998 2048.263
16: 2024-07-30 0.3076331 0.6548709 7153.359 2200.610 4952.749
17: 2024-07-31 0.2334298 0.7096003 7728.034 1803.954 5924.080
18: 2024-08-01 0.2224291 0.7185406 7801.435 1735.266 6066.168
```

- **fCompI**: calcula, en base a los valores de irradiación diaria, todas las componentes de irradiancia. Se vale de dos procedimientos en base al tipo de argumentos que toma:
    - **compD**: Si recibe un **data.table** resultado de **fCompD**, calcula las relaciones entre las componentes de irradiancia e irradiación de las componentes de difusa y global, obteniendo con ellas un perfil de irradiancias [3.2] (las irradiancias global y difusa salen de estas relaciones, mientras que la directa surge por diferencia entre las dos).

```
sol <- calcSol(lat = 37.2, BTd = fBTd(mode = 'prom'),
               sample = 'hour', keep.night = FALSE)
G0d <- 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
compD <- fCompD(sol = sol, G0d = G0d, corr = 'CPR')
compI <- fCompI(sol = sol, compD = compD)
show(compI)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.5656199 0.4583592  84.06042  47.54625  36.40399
  2: 2024-01-17 09:00:00 0.4912826 0.5277148 215.49558 105.86922 109.51548
  3: 2024-01-17 10:00:00 0.4453619 0.5821268 340.45500 151.62569 188.82159
  4: 2024-01-17 11:00:00 0.4195854 0.6178887 433.04376 181.69885 251.45464
  5: 2024-01-17 12:00:00 0.4098508 0.6325646 473.44106 194.04019 279.57020
 ---
141: 2024-12-13 12:00:00 0.5437347 0.5488870 382.71443 208.09513 174.85828
142: 2024-12-13 13:00:00 0.5556284 0.5371376 352.10710 195.64071 156.62669
143: 2024-12-13 14:00:00 0.5893861 0.5063725 276.60890 163.02945 113.57257
144: 2024-12-13 15:00:00 0.6506594 0.4586869 172.87432 112.48231  60.23704
145: 2024-12-13 16:00:00 0.7511394 0.3973283  63.15968  47.44173  15.57107
```

    - **G0I**: Este argumento recibe datos de irradiancia, para después, poder aplicar las correcciones indicadas en el argumento **corr**.

```
G0I <- compI$G0
compI_ekdh <- fCompI(sol = sol, G0I = G0I, corr = 'EKDh')
show(compI_ekdh)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.7417600 0.4583592  84.06042  62.35265  21.70776
  2: 2024-01-17 09:00:00 0.6000150 0.5277148 215.49558 129.30057  86.19500
  3: 2024-01-17 10:00:00 0.4791716 0.5821268 340.45500 163.13636 177.31865
  4: 2024-01-17 11:00:00 0.4004462 0.6178887 433.04376 173.41074 259.63302
  5: 2024-01-17 12:00:00 0.3692679 0.6325646 473.44106 174.82659 298.61447
 ---
141: 2024-12-13 12:00:00 0.5533972 0.5488870 382.71443 211.79307 170.92135
142: 2024-12-13 13:00:00 0.5793829 0.5371376 352.10710 204.00484 148.10226
143: 2024-12-13 14:00:00 0.6457949 0.5063725 276.60890 178.63262  97.97628
144: 2024-12-13 15:00:00 0.7411461 0.4586869 172.87432 128.12512  44.74920
145: 2024-12-13 16:00:00 0.8439123 0.3973283  63.15968  53.30123   9.85845
```

```
compI_brl <- fCompI(sol = sol, G0I = G0I, corr = 'BRL')
show(compI_brl)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.6689300 0.4583592  84.06042  56.23053  27.82988
  2: 2024-01-17 09:00:00 0.5775367 0.5277148 215.49558 124.45660  91.03897
  3: 2024-01-17 10:00:00 0.4826595 0.5821268 340.45500 164.32384 176.13116
  4: 2024-01-17 11:00:00 0.4204896 0.6178887 433.04376 182.09040 250.95337
  5: 2024-01-17 12:00:00 0.3948666 0.6325646 473.44106 186.94604 286.49502
 ---
141: 2024-12-13 12:00:00 0.5872522 0.5488870 382.71443 224.74989 157.96454
142: 2024-12-13 13:00:00 0.6048894 0.5371376 352.10710 212.98583 139.12126
143: 2024-12-13 14:00:00 0.6521416 0.5063725 276.60890 180.38818  96.22073
144: 2024-12-13 15:00:00 0.7207149 0.4586869 172.87432 124.59311  48.28121
145: 2024-12-13 16:00:00 0.7818945 0.3973283  63.15968  49.38421  13.77547
```

```
compI_climedh <- fCompI(sol = sol, G0I = G0I, corr = 'CLIMEDh')
show(compI_climedh)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.7093252 0.4583592  84.06042  59.62617  24.43424
  2: 2024-01-17 09:00:00 0.5818534 0.5277148 215.49558 125.38683  90.10875
  3: 2024-01-17 10:00:00 0.4782729 0.5821268 340.45500 162.83039 177.62462
  4: 2024-01-17 11:00:00 0.4110389 0.6178887 433.04376 177.99784 255.04592
  5: 2024-01-17 12:00:00 0.3840268 0.6325646 473.44106 181.81406 291.62701
 ---
141: 2024-12-13 12:00:00 0.5416063 0.5488870 382.71443 207.28055 175.43387
142: 2024-12-13 13:00:00 0.5639749 0.5371376 352.10710 198.57956 153.52754
143: 2024-12-13 14:00:00 0.6220088 0.5063725 276.60890 172.05317 104.55573
144: 2024-12-13 15:00:00 0.7087489 0.4586869 172.87432 122.52448  50.34984
145: 2024-12-13 16:00:00 0.8099691 0.3973283  63.15968  51.15739  12.00229
```

Como con **fCompD**, se puede añadir una función correctora propia.

```
f_corri <- function(sol, G0i){
  ## Función CLIMEDh
  Kt <- Kti(sol, G0i)
```

```
  Fd=(Kt<=0.21)*(0.995-0.081*Kt)+
    (Kt>0.21 & Kt<=0.76)*(0.724+2.738*Kt-8.32*Kt^2+4.967*Kt^3)+
    (Kt>0.76)*0.180
  return(data.table(Fd, Kt))
}
compI_user <- fCompI(sol = sol, G0I = G0I, corr = 'user', f = f_corri)
show(compI_user)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.7093252 0.4583592  84.06042  59.62617  24.43424
  2: 2024-01-17 09:00:00 0.5818534 0.5277148 215.49558 125.38683  90.10875
  3: 2024-01-17 10:00:00 0.4782729 0.5821268 340.45500 162.83039 177.62462
  4: 2024-01-17 11:00:00 0.4110389 0.6178887 433.04376 177.99784 255.04592
  5: 2024-01-17 12:00:00 0.3840268 0.6325646 473.44106 181.81406 291.62701
 ---
141: 2024-12-13 12:00:00 0.5416063 0.5488870 382.71443 207.28055 175.43387
142: 2024-12-13 13:00:00 0.5639749 0.5371376 352.10710 198.57956 153.52754
143: 2024-12-13 14:00:00 0.6220088 0.5063725 276.60890 172.05317 104.55573
144: 2024-12-13 15:00:00 0.7087489 0.4586869 172.87432 122.52448  50.34984
145: 2024-12-13 16:00:00 0.8099691 0.3973283  63.15968  51.15739  12.00229
```

Y además, se puede no añadir correlación.

```
G0I <- compI_user
compI_none <- fCompI(sol = sol, G0I = G0I, corr = 'none')
show(compI_none)
```

```
Key: <Dates>
                   Dates        Fd        Kt        G0        D0        B0
                  <POSc>     <num>     <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00 0.7093252 0.4583592  84.06042  59.62617  24.43424
  2: 2024-01-17 09:00:00 0.5818534 0.5277148 215.49558 125.38683  90.10875
  3: 2024-01-17 10:00:00 0.4782729 0.5821268 340.45500 162.83039 177.62462
  4: 2024-01-17 11:00:00 0.4110389 0.6178887 433.04376 177.99784 255.04592
  5: 2024-01-17 12:00:00 0.3840268 0.6325646 473.44106 181.81406 291.62701
 ---
141: 2024-12-13 12:00:00 0.5416063 0.5488870 382.71443 207.28055 175.43387
142: 2024-12-13 13:00:00 0.5639749 0.5371376 352.10710 198.57956 153.52754
143: 2024-12-13 14:00:00 0.6220088 0.5063725 276.60890 172.05317 104.55573
144: 2024-12-13 15:00:00 0.7087489 0.4586869 172.87432 122.52448  50.34984
145: 2024-12-13 16:00:00 0.8099691 0.3973283  63.15968  51.15739  12.00229
```

Por útlimo, esta función incluye un argumento extra, **`filterG0`** que cuando su valor es **`TRUE`**, elimina todos aquellos valores de irradiancia que son mayores que la irradiancia extra-atmosfércia (ya que es incoherente que la irradiancia terrestre sea mayor que la extra-terrestre)

Estas dos funciones, como se muestra en la figura 4.4, convergen en la función constructora **`calcG0`**, dando como resultado un objeto de clase **`G0`**. Este objeto muestra la media mensual de la irradiación diaria y la irradiación anual. Aparte, incluye los resultados de **`fCompD`** y **`fCompI`** y los objetos **`Sol`** y **`Meteo`** de los que parte.

Como argumento más importante está **`modeRad`**, el cual selecciona el tipo de datos que introduce el usuario en el argumento **`dataRad`**. Estos son:

- Medias mensuales.

```
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 <- data.table(G0dm, Ta)
g0_prom <- calcG0(lat, modeRad = 'prom', dataRad = prom)
show(g0_prom)
```

```
Object of class  G0

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates   G0d      D0d      B0d
       <char> <num>    <num>    <num>
 1: Jan. 2024 2.766 0.941698 1.824302
 2: Feb. 2024 3.491 1.247146 2.243854
 3: Mar. 2024 4.494 1.671763 2.822237
 4: Apr. 2024 5.912 1.931146 3.980854
 5: May. 2024 6.989 2.023364 4.965636
 6: Jun. 2024 7.742 1.889994 5.852006
 7: Jul. 2024 7.919 1.624064 6.294936
 8: Aug. 2024 7.027 1.547591 5.479409
 9: Sep. 2024 5.369 1.540708 3.828292
10: Oct. 2024 3.562 1.374513 2.187487
11: Nov. 2024 2.814 1.006959 1.807041
12: Dec. 2024 2.179 0.926737 1.252263

Yearly values:
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 540.6331 1298.732
```

- Generación de secuencias diarias mediante matrices de transición de Markov.

```
g0_aguiar <- calcG0(lat, modeRad = 'aguiar', dataRad = prom)
show(g0_aguiar)
```

```
Object of class  G0

Source of meteorological information: bd-aguiar

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates   G0d      D0d      B0d
       <char> <num>    <num>    <num>
 1: Jan. 2024 2.766 1.018399 1.747601
 2: Feb. 2024 3.491 1.539128 1.951872
 3: Mar. 2024 4.494 2.019729 2.474271
 4: Apr. 2024 5.912 2.212283 3.699717
 5: May. 2024 6.989 2.443056 4.545944
 6: Jun. 2024 7.742 2.469004 5.272996
 7: Jul. 2024 7.919 2.202626 5.716374
 8: Aug. 2024 7.027 2.245755 4.781245
 9: Sep. 2024 5.369 1.714299 3.654701
10: Oct. 2024 3.562 1.791424 1.770576
11: Nov. 2024 2.814 1.148719 1.665281
12: Dec. 2024 2.179 1.161355 1.017645

Yearly values:
```

```
Key: <Dates>
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 670.3165 1169.049
```

- Diarios.

```r
bd <- as.data.tableD(g0_aguiar)
g0_bd <- calcG0(lat, modeRad = 'bd', dataRad = bd)
show(g0_bd)
```

```
Object of class  G0

Source of meteorological information: bd-data.table

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates   G0d      D0d      B0d
       <char> <num>    <num>    <num>
 1: Jan. 2024 2.766 1.018399 1.747601
 2: Feb. 2024 3.491 1.539128 1.951872
 3: Mar. 2024 4.494 2.019729 2.474271
 4: Apr. 2024 5.912 2.212283 3.699717
 5: May. 2024 6.989 2.443056 4.545944
 6: Jun. 2024 7.742 2.469004 5.272996
 7: Jul. 2024 7.919 2.202626 5.716374
 8: Aug. 2024 7.027 2.245755 4.781245
 9: Sep. 2024 5.369 1.714299 3.654701
10: Oct. 2024 3.562 1.791424 1.770576
11: Nov. 2024 2.814 1.148719 1.665281
12: Dec. 2024 2.179 1.161355 1.017645

Yearly values:
Key: <Dates>
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 670.3165 1169.049
```

- Intradiarios

```r
bdI <- as.data.tableI(g0_aguiar)
g0_bdI <- calcG0(lat, modeRad = 'bdI', dataRad = bdI)
show(g0_bdI)
```

```
Object of class  G0

Source of meteorological information: bdI-data.table

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      G0d       D0d       B0d
       <char>    <num>     <num>     <num>
 1: Jan. 2024 2.766000 0.9625472 1.8034528
 2: Feb. 2024 3.491000 1.5898320 1.9011680
 3: Mar. 2024 4.494000 2.0981954 2.3958046
 4: Apr. 2024 5.912000 2.1497145 3.7622855
 5: May. 2024 6.989000 2.3990824 4.5899176
 6: Jun. 2024 7.742000 2.3030247 5.4389753
 7: Jul. 2024 7.919000 1.7067936 6.2122064
 8: Aug. 2024 6.483769 1.6494109 4.8343585
 9: Sep. 2024 5.369000 1.6544602 3.7145398
```

```
10: Oct. 2024 3.562000 1.9052417 1.6567583
11: Nov. 2024 2.659222 0.9870476 1.6721742
12: Dec. 2024 2.179000 1.2360451 0.9429549

Yearly values:
Key: <Dates>
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1817.882 629.6093 1188.272
```

4.4. Radiación efectiva en el plano del generador

Teniendo la radiación incidente en plano horizontal (sección 4.3), se puede calcular la radiación efectiva incidente en el plano del generador. Para ello, **solaR2** cuenta con la función **calcGef** [??] la cual mediante las funciones **fInclin** y **calcShd** procesa un objeto de clase **G0** para obtener un objeto **Gef**.

Como se puede ver en la figura 4.5, **calcGef** funciona gracias a las siguientes funciones:

- **fTheta**: la cual, partiendo del ángulo de inclinación (β) y la orientación (α), calcula el ángulo de inclinación en cada instante (β), el ángulo azimutal (ψ_s) y el coseno del ángulo de incidencia de la radiación solar en la superficie ($cos(\theta_s)$). Como principal argumento tiene **modeTrk**, el cual determina el sistema de seguimiento que tiene el sistema:
  - **fixed**: para sistemas estáticos.

```r
BTd <- fBTd(mode = 'prom')[6]
sol <- calcSol(lat, BTd = BTd, keep.night = FALSE)
beta <- lat - 10
alfa <- 0
angGen_fixed <- fTheta(sol = sol, beta = beta, alfa = alfa,
                       modeTrk = 'fixed')
show(angGen_fixed)
```

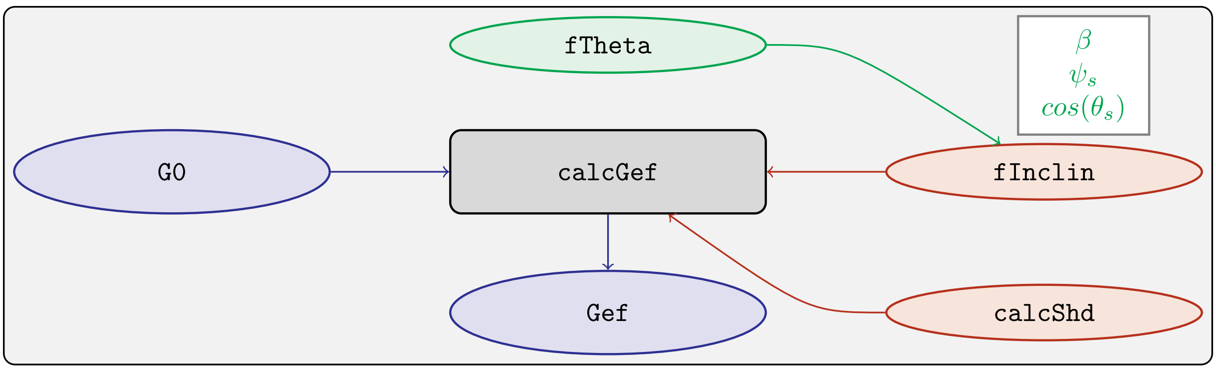


Figura 4.5: *Cálculo de la radiación efectiva incidente en el plano del generador mediante la función* ***calcGef****, la cual emplea la función* ***fInclin*** *para el computo de las componentes efectivas, la función* ***fTheta*** *que provee a la función anterior los ángulos necesarios para su computo y la función* ***calcShd*** *que reprocesa el objeto de clase* ***Gef*** *resultante, añadiendole el efecto de las sombras producidas entres módulos.*

```
                   Dates      Beta  Alfa   cosTheta
                  <POSc>     <num> <num>      <num>
 1: 2024-06-10 05:00:00 0.4747296     0 0.00000000
 2: 2024-06-10 06:00:00 0.4747296     0 0.06990810
 3: 2024-06-10 07:00:00 0.4747296     0 0.30432148
 4: 2024-06-10 08:00:00 0.4747296     0 0.52263672
 5: 2024-06-10 09:00:00 0.4747296     0 0.70998013
 6: 2024-06-10 10:00:00 0.4747296     0 0.85358815
 7: 2024-06-10 11:00:00 0.4747296     0 0.94367686
 8: 2024-06-10 12:00:00 0.4747296     0 0.97410861
 9: 2024-06-10 13:00:00 0.4747296     0 0.94281011
10: 2024-06-10 14:00:00 0.4747296     0 0.85191372
11: 2024-06-10 15:00:00 0.4747296     0 0.70761218
12: 2024-06-10 16:00:00 0.4747296     0 0.51973665
13: 2024-06-10 17:00:00 0.4747296     0 0.30108697
14: 2024-06-10 18:00:00 0.4747296     0 0.06655958
15: 2024-06-10 19:00:00 0.4747296     0 0.00000000
```

- **two**: para sistemas de seguimiento de doble eje.

```r
angGen_two <- fTheta(sol = sol, beta = beta, alfa = alfa,
                     modeTrk = 'two')
show(angGen_two)
```

```
                   Dates      Beta         Alfa cosTheta
                  <POSc>     <num>        <num>    <num>
 1: 2024-06-10 05:00:00 1.5220852 -2.043678875        1
 2: 2024-06-10 06:00:00 1.3300857 -1.896688029        1
 3: 2024-06-10 07:00:00 1.1285281 -1.756655282        1
 4: 2024-06-10 08:00:00 0.9215732 -1.612213267        1
 5: 2024-06-10 09:00:00 0.7134716 -1.445120762        1
 6: 2024-06-10 10:00:00 0.5110180 -1.215351693        1
 7: 2024-06-10 11:00:00 0.3328578 -0.809087856        1
 8: 2024-06-10 12:00:00 0.2466893  0.006963841        1
 9: 2024-06-10 13:00:00 0.3349967  0.817155564        1
10: 2024-06-10 14:00:00 0.5137803  1.219398208        1
11: 2024-06-10 15:00:00 0.7163931  1.447776194        1
12: 2024-06-10 16:00:00 0.9245147  1.614353339        1
13: 2024-06-10 17:00:00 1.1314208  1.758631827        1
14: 2024-06-10 18:00:00 1.3328735  1.898691776        1
15: 2024-06-10 19:00:00 1.5247042  2.045849315        1
```

- **horiz**: para sistemas de seguimiento horizontal Norte-Sur.

```r
angGen_horiz <- fTheta(sol = sol, beta = beta, alfa = alfa,
                       modeTrk = 'horiz')
show(angGen_horiz)
```

```
                   Dates        Beta      Alfa  cosTheta
                  <POSc>       <num>     <num>     <num>
 1: 2024-06-10 05:00:00 1.516091993 -1.570796 0.8905353
 2: 2024-06-10 06:00:00 1.317263961 -1.570796 0.9504350
 3: 2024-06-10 07:00:00 1.121771495 -1.570796 0.9859551
 4: 2024-06-10 08:00:00 0.921160041 -1.570796 0.9994560
 5: 2024-06-10 09:00:00 0.709555740 -1.570796 0.9966296
 6: 2024-06-10 10:00:00 0.483954771 -1.570796 0.9854098
 7: 2024-06-10 11:00:00 0.245151627 -1.570796 0.9742418
 8: 2024-06-10 12:00:00 0.001753607  1.570796 0.9697277
 9: 2024-06-10 13:00:00 0.248597042  1.570796 0.9743648
10: 2024-06-10 14:00:00 0.487239436  1.570796 0.9855868
11: 2024-06-10 15:00:00 0.712638107  1.570796 0.9967482
```

```
12: 2024-06-10 16:00:00 0.924058412  1.570796 0.9993956
13: 2024-06-10 17:00:00 1.124550569  1.570796 0.9856166
14: 2024-06-10 18:00:00 1.320024608  1.570796 0.9497600
15: 2024-06-10 19:00:00 1.518974473  1.570796 0.8895182
```

También, tiene un argumento **BT** que indica cuando se usa la técnica de backtracking para un sistema horizontal Norte-Sur. Para funcionar, necesita de los argumentos **struct**, el cual presenta una lista con la altura de los módulos, y **dist**, el cual presenta un **data.frame** (o **data.table**) con la distancia que separa los módulos en la dirección Este-Oeste.

```
struct <- list(L = 1)
distances <- data.table(Lew = 2)
angGen_BT <- fTheta(sol = sol, beta = beta, alfa = alfa,
                    modeTrk = 'horiz', BT = TRUE,
                    struct = struct, dist = distances)
show(angGen_BT)
```

```
                  Dates        Beta      Alfa   cosTheta
                 <POSc>       <num>     <num>      <num>
 1: 2024-06-10 05:00:00 0.054868903 -1.570796 0.09738369
 2: 2024-06-10 06:00:00 0.271972628 -1.570796 0.47678565
 3: 2024-06-10 07:00:00 0.602487004 -1.570796 0.85598103
 4: 2024-06-10 08:00:00 0.921160041 -1.570796 0.99945597
 5: 2024-06-10 09:00:00 0.709555740 -1.570796 0.99662956
 6: 2024-06-10 10:00:00 0.483954771 -1.570796 0.98540983
 7: 2024-06-10 11:00:00 0.245151627 -1.570796 0.97424175
 8: 2024-06-10 12:00:00 0.001753607  1.570796 0.96972767
 9: 2024-06-10 13:00:00 0.248597042  1.570796 0.97436477
10: 2024-06-10 14:00:00 0.487239436  1.570796 0.98558683
11: 2024-06-10 15:00:00 0.712638107  1.570796 0.99674816
12: 2024-06-10 16:00:00 0.924058412  1.570796 0.99939563
13: 2024-06-10 17:00:00 0.595256963  1.570796 0.85074877
14: 2024-06-10 18:00:00 0.268563625  1.570796 0.47136897
15: 2024-06-10 19:00:00 0.051961679  1.570796 0.09215170
```

- **fInclin**: la cual, partiendo del resultado de **fTheta** y de un objeto de clase **G0**, cálcula la irradiancia solar incidente en una superficie inclinada junto con los efectos del ángulo de incidencia y la suciedad para obtener la irradiancia efectiva. Como argumentos principales están:
  - **iS**: permite seleccionar entre 4 valores del 1 al 4 correspondientes al grado de suciedad del módulo. Siendo 1 limpio y 4 alto y basandose en los valores de la tabla 3.2 calcula la irradiancia efectiva. Por defecto tiene valor 2 (grado de suciedad bajo).

```
compI <- calcG0(lat, dataRad = prom, keep.night = FALSE)
sol <- calcSol(lat, BTi = indexI(compI))
angGen <- fTheta(sol = sol, beta = beta, alfa = alfa)
inclin_limpio <- fInclin(compI = compI, angGen = angGen, iS = 1)
show(inclin_limpio)
```

```
                   Dates        Bo       Bn        G         D       Di        Dc         B         R
                  <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00  514.5612 365.8727 186.4590  52.34286 25.82073  26.52212 133.18653 0.9295706
  2: 2024-01-17 09:00:00  792.6980 464.2106 366.6704 103.96230 52.12242  51.83988 260.32510 2.3830282
  3: 2024-01-17 10:00:00 1010.9063 541.3602 536.6247 145.69981 68.60264  77.09717 387.15997 3.7648749
  4: 2024-01-17 11:00:00 1154.3223 592.0663 662.0048 173.72247 77.44190  96.28057 483.49354 4.7887550
  5: 2024-01-17 12:00:00 1213.1770 612.8750 716.5974 185.35767 80.61172 104.74595 526.00427 5.2354830
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 578.4583 180.82966 95.85462  84.97504 393.39650 4.2321949
```

```
142: 2024-12-13 13:00:00 1129.5610 453.5904 536.8668 170.08970 91.70559  78.38411 362.88341 3.8937280
143: 2024-12-13 14:00:00  994.4636 409.9651 434.0673 142.25355 79.88147  62.37208 288.75488 3.0588416
144: 2024-12-13 15:00:00  785.0640 342.3463 292.1950  99.92831 58.81096  41.11735 190.35496 1.9117069
145: 2024-12-13 16:00:00  515.6229 255.3390 140.8937  46.94651 26.80445  20.14206  93.24874 0.6984426
             FTb        FTd       FTr     Dief      Dcef      Gef       Def       Bef       Ref
           <num>      <num>     <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 0.115032290 0.05043622 0.2503398 24.51843  23.47122 166.5523  47.98966 117.86578 0.6968621
  2: 0.034235799 0.05043622 0.2503398 49.49356  50.06510 352.7578  99.55866 251.41266 1.7864615
  3: 0.012139104 0.05043622 0.2503398 65.14258  76.16128 526.5864 141.30386 382.46020 2.8223770
  4: 0.005426675 0.05043622 0.2503398 73.53602  95.75809 653.7538 169.29411 480.86978 3.5899392
  5: 0.003640433 0.05043622 0.2503398 76.54597 104.36463 708.9248 180.91060 524.08939 3.9248333
 ---
141: 0.004516349 0.05043622 0.2503398 91.02007  84.59127 570.4038 175.61134 391.61978 3.1727082
142: 0.006269898 0.05043622 0.2503398 87.08031  77.89265 528.5001 164.97296 360.60816 2.9189730
143: 0.013120704 0.05043622 0.2503398 75.85255  61.55372 424.6656 137.40626 284.96622 2.2930919
144: 0.035287438 0.05043622 0.2503398 55.84476  39.66642 280.5821  95.51118 183.63782 1.4331306
145: 0.114223038 0.05043622 0.2503398 25.45254  17.84137 126.4151  43.29391  82.59758 0.5235947
```

```r
inclin_sucio <- fInclin(compI = compI, angGen = angGen, iS = 4)
show(inclin_sucio)
```

```
                   Dates        Bo       Bn        G         D       Di        Dc         B         R
                  <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00  514.5612 365.8727 186.4590  52.34286 25.82073  26.52212 133.18653 0.9295706
  2: 2024-01-17 09:00:00  792.6980 464.2106 366.6704 103.96230 52.12242  51.83988 260.32510 2.3830282
  3: 2024-01-17 10:00:00 1010.9063 541.3602 536.6247 145.69981 68.60264  77.09717 387.15997 3.7648749
  4: 2024-01-17 11:00:00 1154.3223 592.0663 662.0048 173.72247 77.44190  96.28057 483.49354 4.7887550
  5: 2024-01-17 12:00:00 1213.1770 612.8750 716.5974 185.35767 80.61172 104.74595 526.00427 5.2354830
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 578.4583 180.82966 95.85462  84.97504 393.39650 4.2321949
142: 2024-12-13 13:00:00 1129.5610 453.5904 536.8668 170.08970 91.70559  78.38411 362.88341 3.8937280
143: 2024-12-13 14:00:00  994.4636 409.9651 434.0673 142.25355 79.88147  62.37208 288.75488 3.0588416
144: 2024-12-13 15:00:00  785.0640 342.3463 292.1950  99.92831 58.81096  41.11735 190.35496 1.9117069
145: 2024-12-13 16:00:00  515.6229 255.3390 140.8937  46.94651 26.80445  20.14206  93.24874 0.6984426
            FTb        FTd       FTr     Dief     Dcef      Gef       Def       Bef       Ref
          <num>      <num>     <num>    <num>    <num>    <num>     <num>     <num>     <num>
  1: 0.24100175 0.09714708 0.3918962 21.44734 18.51982 133.4885  39.96716  93.00127 0.5200533
  2: 0.10321543 0.09714708 0.3918962 43.29416 42.77007 302.1765  86.06424 214.77909 1.3331982
  3: 0.04727214 0.09714708 0.3918962 56.98305 67.57641 466.0152 124.55946 339.34944 2.1062799
  4: 0.02455379 0.09714708 0.3918962 64.32515 86.40320 587.2996 150.72835 433.89218 2.6790952
  5: 0.01743586 0.09714708 0.3918962 66.95809 94.68605 640.0594 161.64413 475.48630 2.9290196
 ---
141: 0.02100686 0.09714708 0.3918962 79.61921 76.53478 512.8436 156.15400 354.32187 2.3677246
142: 0.02771140 0.09714708 0.3918962 76.17293 70.11502 473.0675 146.28795 324.60121 2.1783674
143: 0.05023795 0.09714708 0.3918962 66.35152 54.49955 374.8709 120.85106 252.30856 1.7112857
144: 0.10550059 0.09714708 0.3918962 48.84983 33.83709 240.4070  82.68692 156.65061 1.0695149
145: 0.23984890 0.09714708 0.3918962 22.26444 14.08613 101.9538  36.35057  65.21248 0.3907476
```

- **alb** Correspondiente al coeficiente de reflexión del terreno para la irradiancia de albedo. Por defecto tiene un valor de 0,2 (valor aceptable para un terreno normal).

```r
inclin_alb0 <- fInclin(compI = compI, angGen = angGen, alb = 0)
show(inclin_alb0)
```

```
                   Dates        Bo       Bn        G         D       Di        Dc         B     R
                  <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num> <num>
  1: 2024-01-17 08:00:00  514.5612 365.8727 185.5294  52.34286 25.82073  26.52212 133.18653     0
  2: 2024-01-17 09:00:00  792.6980 464.2106 364.2874 103.96230 52.12242  51.83988 260.32510     0
  3: 2024-01-17 10:00:00 1010.9063 541.3602 532.8598 145.69981 68.60264  77.09717 387.15997     0
  4: 2024-01-17 11:00:00 1154.3223 592.0663 657.2160 173.72247 77.44190  96.28057 483.49354     0
  5: 2024-01-17 12:00:00 1213.1770 612.8750 711.3619 185.35767 80.61172 104.74595 526.00427     0
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 574.2262 180.82966 95.85462  84.97504 393.39650     0
142: 2024-12-13 13:00:00 1129.5610 453.5904 532.9731 170.08970 91.70559  78.38411 362.88341     0
```

```
143: 2024-12-13 14:00:00  994.4636 409.9651 431.0084 142.25355 79.88147  62.37208 288.75488     0
144: 2024-12-13 15:00:00  785.0640 342.3463 290.2833  99.92831 58.81096  41.11735 190.35496     0
145: 2024-12-13 16:00:00  515.6229 255.3390 140.1953  46.94651 26.80445  20.14206  93.24874     0
             FTb         FTd        FTr      Dief       Dcef      Gef       Def       Bef   Ref
           <num>       <num>      <num>     <num>      <num>    <num>     <num>     <num> <num>
  1: 0.156321477 0.06473603 0.2994808 23.66622  21.92862 155.7141  45.59484 110.11928     0
  2: 0.054197292 0.06473603 0.2994808 47.77325  48.04970 337.1148  95.82295 241.29186     0
  3: 0.021399057 0.06473603 0.2994808 62.87835  73.93841 508.1144 136.81676 371.29761     0
  4: 0.010185772 0.06473603 0.2994808 70.98005  93.39388 633.3713 164.37393 468.99741     0
  5: 0.006996517 0.06473603 0.2994808 73.88537 101.93283 687.6958 175.81821 511.87759     0
 ---
141: 0.008575046 0.06473603 0.2994808 87.85638  82.56145 552.6405 170.41783 382.22264     0
142: 0.011653979 0.06473603 0.2994808 84.05356  75.92121 511.4560 159.97477 351.48128     0
143: 0.022965930 0.06473603 0.2994808 73.21605  59.72086 409.4178 132.93691 276.48089     0
144: 0.055666181 0.06473603 0.2994808 53.90370  38.05193 268.1191  91.95563 176.16345     0
145: 0.155368802 0.06473603 0.2994808 24.56786  16.67236 118.4258  41.24021  77.18558     0
```

```
inclin_alb1 <- fInclin(compI = compI, angGen = angGen, alb = 1)
show(inclin_alb1)
```

```
                    Dates        Bo       Bn        G         D       Di        Dc         B         R
                   <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00  514.5612 365.8727 190.1772  52.34286 25.82073  26.52212 133.18653  4.647853
  2: 2024-01-17 09:00:00  792.6980 464.2106 376.2025 103.96230 52.12242  51.83988 260.32510 11.915141
  3: 2024-01-17 10:00:00 1010.9063 541.3602 551.6842 145.69981 68.60264  77.09717 387.15997 18.824375
  4: 2024-01-17 11:00:00 1154.3223 592.0663 681.1598 173.72247 77.44190  96.28057 483.49354 23.943775
  5: 2024-01-17 12:00:00 1213.1770 612.8750 737.5394 185.35767 80.61172 104.74595 526.00427 26.177415
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 595.3871 180.82966 95.85462  84.97504 393.39650 21.160975
142: 2024-12-13 13:00:00 1129.5610 453.5904 552.4417 170.08970 91.70559  78.38411 362.88341 19.468640
143: 2024-12-13 14:00:00  994.4636 409.9651 446.3026 142.25355 79.88147  62.37208 288.75488 15.294208
144: 2024-12-13 15:00:00  785.0640 342.3463 299.8418  99.92831 58.81096  41.11735 190.35496  9.558535
145: 2024-12-13 16:00:00  515.6229 255.3390 143.6875  46.94651 26.80445  20.14206  93.24874  3.492213
             FTb         FTd        FTr      Dief       Dcef      Gef       Def       Bef       Ref
           <num>       <num>      <num>     <num>      <num>    <num>     <num>     <num>     <num>
  1: 0.156321477 0.06473603 0.2994808 23.66622  21.92862 158.9049  45.59484 110.11928  3.190792
  2: 0.054197292 0.06473603 0.2994808 47.77325  48.04970 345.2947  95.82295 241.29186  8.179849
  3: 0.021399057 0.06473603 0.2994808 62.87835  73.93841 521.0375 136.81676 371.29761 12.923098
  4: 0.010185772 0.06473603 0.2994808 70.98005  93.39388 649.8089 164.37393 468.99741 16.437612
  5: 0.006996517 0.06473603 0.2994808 73.88537 101.93283 705.6668 175.81821 511.87759 17.971025
 ---
141: 0.008575046 0.06473603 0.2994808 87.85638  82.56145 567.1677 170.41783 382.22264 14.527195
142: 0.011653979 0.06473603 0.2994808 84.05356  75.92121 524.8214 159.97477 351.48128 13.365392
143: 0.022965930 0.06473603 0.2994808 73.21605  59.72086 419.9174 132.93691 276.48089 10.499608
144: 0.055666181 0.06473603 0.2994808 53.90370  38.05193 274.6811  91.95563 176.16345  6.562018
145: 0.155368802 0.06473603 0.2994808 24.56786  16.67236 120.8232  41.24021  77.18558  2.397435
```

Además, cuenta con dos argumentos adicionales, **`horizBright`**, el cual, cuando su valor es **`TRUE`** (el que tiene por defecto), realiza una corrección de la radiación difusa [RBD90], y **`HCPV`**, es el acrónimo de **High Concentration PV system**[7] (sistema fotovoltaico de alta concentración) que cuando su valor es **`TRUE`** (por defecto está puesto en **`FALSE`**), anula los valores de radiación difusa y de albedo.

```
inclin_horizBright <- fInclin(compI = compI, angGen = angGen,
                                  horizBright = FALSE)
show(inclin_horizBright)
```

```
                    Dates        Bo       Bn        G         D       Di        Dc         B         R
                   <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num>     <num>
```

[7] la tencología de concentración fotovoltaica funciona gracias a unos dispositivos ópticos que permiten concentrar la radiación solar sobre una célula fotovoltaica de tamaño reducido pero con una eficiencia muy superior alas células tradicionales. Con ello se consigue emplear menor cantidad de semiconductores reduciendo los costes.

```
  1: 2024-01-17 08:00:00  514.5612 365.8727 186.2091  52.09303 25.57090  26.52212 133.18653 0.9295706
  2: 2024-01-17 09:00:00  792.6980 464.2106 366.1413 103.43314 51.59325  51.83988 260.32510 2.3830282
  3: 2024-01-17 10:00:00 1010.9063 541.3602 535.9087 144.98390 67.88673  77.09717 387.15997 3.7648749
  4: 2024-01-17 11:00:00 1154.3223 592.0663 661.1846 172.90227 76.62170  96.28057 483.49354 4.7887550
  5: 2024-01-17 12:00:00 1213.1770 612.8750 715.7390 184.49921 79.75326 104.74595 526.00427 5.2354830
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 577.4973 179.86860 94.89356  84.97504 393.39650 4.2321949
142: 2024-12-13 13:00:00 1129.5610 453.5904 535.9539 169.17679 90.79268  78.38411 362.88341 3.8937280
143: 2024-12-13 14:00:00  994.4636 409.9651 433.2885 141.47476 79.10268  62.37208 288.75488 3.0588416
144: 2024-12-13 15:00:00  785.0640 342.3463 291.6442  99.37758 58.26023  41.11735 190.35496 1.9117069
145: 2024-12-13 16:00:00  515.6229 255.3390 140.6606  46.71344 26.57138  20.14206  93.24874 0.6984426
             FTb        FTd       FTr     Dief      Dcef      Gef       Def       Bef       Ref
           <num>      <num>     <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 0.156321477 0.06473603 0.2994808 23.43723  21.92862 156.1233  45.36586 110.11928 0.6381583
  2: 0.054197292 0.06473603 0.2994808 47.28824  48.04970 338.2658  95.33794 241.29186 1.6359698
  3: 0.021399057 0.06473603 0.2994808 62.22217  73.93841 510.0428 136.16059 371.29761 2.5846197
  4: 0.010185772 0.06473603 0.2994808 70.22829  93.39388 635.9071 163.62217 468.99741 3.2875223
  5: 0.006996517 0.06473603 0.2994808 73.09855 101.93283 690.5032 175.03138 511.87759 3.5942050
 ---
141: 0.008575046 0.06473603 0.2994808 86.97552  82.56145 554.6650 169.53697 382.22264 2.9054390
142: 0.011653979 0.06473603 0.2994808 83.21682  75.92121 513.2924 159.13803 351.48128 2.6730784
143: 0.022965930 0.06473603 0.2994808 72.50225  59.72086 410.8039 132.22311 276.48089 2.0999216
144: 0.055666181 0.06473603 0.2994808 53.39892  38.05193 268.9267  91.45086 176.16345 1.3124036
145: 0.155368802 0.06473603 0.2994808 24.35423  16.67236 118.6917  41.02659  77.18558 0.4794870
```

```r
inclin_HCPV <- fInclin(compI = compI, angGen = angGen,
                       HCPV = TRUE)
show(inclin_HCPV)
```

```
                   Dates        Bo       Bn        G         D       Di        Dc         B         R
                  <POSc>     <num>    <num>    <num>     <num>    <num>     <num>     <num>     <num>
  1: 2024-01-17 08:00:00  514.5612 365.8727 186.4590  52.34286 25.82073  26.52212 133.18653 0.9295706
  2: 2024-01-17 09:00:00  792.6980 464.2106 366.6704 103.96230 52.12242  51.83988 260.32510 2.3830282
  3: 2024-01-17 10:00:00 1010.9063 541.3602 536.6247 145.69981 68.60264  77.09717 387.15997 3.7648749
  4: 2024-01-17 11:00:00 1154.3223 592.0663 662.0048 173.72247 77.44190  96.28057 483.49354 4.7887550
  5: 2024-01-17 12:00:00 1213.1770 612.8750 716.5974 185.35767 80.61172 104.74595 526.00427 5.2354830
 ---
141: 2024-12-13 12:00:00 1181.1554 470.2512 578.4583 180.82966 95.85462  84.97504 393.39650 4.2321949
142: 2024-12-13 13:00:00 1129.5610 453.5904 536.8668 170.08970 91.70559  78.38411 362.88341 3.8937280
143: 2024-12-13 14:00:00  994.4636 409.9651 434.0673 142.25355 79.88147  62.37208 288.75488 3.0588416
144: 2024-12-13 15:00:00  785.0640 342.3463 292.1950  99.92831 58.81096  41.11735 190.35496 1.9117069
145: 2024-12-13 16:00:00  515.6229 255.3390 140.8937  46.94651 26.80445  20.14206  93.24874 0.6984426
             FTb        FTd       FTr  Dief  Dcef       Gef   Def       Bef   Ref
           <num>      <num>     <num> <num> <num>     <num> <num>     <num> <num>
  1: 0.156321477 0.06473603 0.2994808     0     0 110.11928     0 110.11928     0
  2: 0.054197292 0.06473603 0.2994808     0     0 241.29186     0 241.29186     0
  3: 0.021399057 0.06473603 0.2994808     0     0 371.29761     0 371.29761     0
  4: 0.010185772 0.06473603 0.2994808     0     0 468.99741     0 468.99741     0
  5: 0.006996517 0.06473603 0.2994808     0     0 511.87759     0 511.87759     0
 ---
141: 0.008575046 0.06473603 0.2994808     0     0 382.22264     0 382.22264     0
142: 0.011653979 0.06473603 0.2994808     0     0 351.48128     0 351.48128     0
143: 0.022965930 0.06473603 0.2994808     0     0 276.48089     0 276.48089     0
144: 0.055666181 0.06473603 0.2994808     0     0 176.16345     0 176.16345     0
145: 0.155368802 0.06473603 0.2994808     0     0  77.18558     0  77.18558     0
```

Finalmente, esta función le otorga estos datos a la función **calcGef** para que produzca un objeto de clase **Gef** como resultado. Esta función tiene como argumentos principales los mismos que los que tiene **calcG0** 4.3, es decir, **modeRad** y **dataRad**. Y además, como es lógico, con todos los argumentos mencionados con anterioridad en **fTheta** y **fInclin**.

```r
gef_prom <- calcGef(lat = lat, modeTrk = 'two', modeRad = 'prom',
                    dataRad = prom,
                    beta = lat-10, alfa = 0,
```

```r
                    iS = 2, alb = 0.2,
                    horizBright = TRUE, HCPV = FALSE)
show(gef_prom)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Bod       Bnd       Gd       Dd       Bd      Gefd     Defd     Befd
       <char>    <num>     <num>     <num>    <num>    <num>     <num>    <num>    <num>
 1: Jan. 2024 14.13536  4.924221  6.522313 1.440413 4.924221  6.348801 1.384087 4.825736
 2: Feb. 2024 15.42754  5.034287  6.875052 1.672079 5.034287  6.680139 1.599929 4.933601
 3: Mar. 2024 16.58107  5.163713  7.329138 1.998110 5.163713  7.104641 1.902356 5.060439
 4: Apr. 2024 17.64047  6.408617  8.843422 2.265896 6.408617  8.578222 2.158071 6.280444
 5: May. 2024 18.70771  7.617499 10.178196 2.394606 7.617499  9.885240 2.284334 7.465149
 6: Jun. 2024 19.87238  9.102430 11.606533 2.329653 9.102430 11.293417 2.230338 8.920381
 7: Jul. 2024 18.51695 10.037233 11.801533 2.029150 9.589205 11.495648 1.948530 9.397421
 8: Aug. 2024 17.34098  8.640959 10.777404 1.947410 8.640959 10.493150 1.869393 8.468140
 9: Sep. 2024 16.25295  6.698488  8.831006 1.948075 6.698488  8.584604 1.864962 6.564518
10: Oct. 2024 15.16994  4.546024  6.418653 1.711039 4.546024  6.226290 1.631551 4.455104
11: Nov. 2024 14.00493  4.638289  6.247341 1.452953 4.638289  6.076159 1.393353 4.545523
12: Dec. 2024 12.70717  3.439788  4.825181 1.254616 3.439788  4.685547 1.198824 3.370992

Yearly values:
   Dates      Bod      Bnd       Gd       Dd       Bd     Gefd    Defd     Befd
   <int>    <num>    <num>    <num>    <num>    <num>    <num>   <num>    <num>
1:  2024 5988.455 2326.882 3058.651 684.4232 2312.993 2973.115 654.591 2266.733
-----------------
Mode of tracking:  two
    Inclination limit: 90
```

Sin embargo, como argumento importante está **modeShd**, el cual permite incluir el efecto de las sombras entre módulos al objeto **Gef** mediante el uso de la función **calcShd**. Esta opción añade las variables **Gef0**, **Def0** y **Bef0** las cuales son las componentes de radiación efectiva previas a aplicar el efecto de las sombras con el fin de poder comparar.

```r
struct <- list(W=23.11, L=9.8, Nrow=2, Ncol=8)
distances <- data.table(Lew=40, Lns=30, H=0)
gef_shd <- calcShd(radEf = gef_prom, modeShd = 'prom',
                   struct = struct, distances = distances)
show(gef_shd)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates    Gef0d    Def0d    Bef0d       Gd       Dd       Bd      Gefd     Defd     Befd
       <char>    <num>    <num>    <num>    <num>    <num>    <num>     <num>    <num>    <num>
 1: Jan. 2024  6.348801 1.384087 4.825736  6.522313 1.440413 4.924221  6.104126 1.343455 4.621693
 2: Feb. 2024  6.680139 1.599929 4.933601  6.875052 1.672079 5.034287  6.406274 1.553670 4.705996
 3: Mar. 2024  7.104641 1.902356 5.060439  7.329138 1.998110 5.163713  6.788630 1.848127 4.798657
 4: Apr. 2024  8.578222 2.158071 6.280444  8.843422 2.265896 6.408617  8.295340 2.112064 6.043569
 5: May. 2024  9.885240 2.284334 7.465149 10.178196 2.394606 7.617499  9.688308 2.253942 7.298609
 6: Jun. 2024 11.293417 2.230338 8.920381 11.606533 2.329653 9.102430 11.115054 2.205314 8.767042
```

```
 7: Jul. 2024 11.495648 1.948530 9.397421 11.801533 2.029150 9.589205 11.308971 1.924962 9.234312
 8: Aug. 2024 10.493150 1.869393 8.468140 10.777404 1.947410 8.640959 10.196758 1.830334 8.210807
 9: Sep. 2024  8.584604 1.864962 6.564518  8.831006 1.948075 6.698488  8.228309 1.810198 6.262986
10: Oct. 2024  6.226290 1.631551 4.455104  6.418653 1.711039 4.546024  6.018374 1.595528 4.283212
11: Nov. 2024  6.076159 1.393353 4.545523  6.247341 1.452953 4.638289  5.875732 1.359514 4.378935
12: Dec. 2024  4.685547 1.198824 3.370992  4.825181 1.254616 3.439788  4.575893 1.179346 3.280817

Yearly values:
   Dates   Gef0d   Def0d    Bef0d       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>   <num>   <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 2973.115 654.591 2266.733 3058.651 684.4232 2312.993 2886.328 640.9157 2193.621
-----------------
Mode of tracking:  two
    Inclination limit: 90
```

```r
gef_shd2 <- calcGef(lat = lat, modeTrk = 'two', dataRad = prom,
                    modeShd = 'prom', struct = struct, distances = distances)
show(gef_shd2)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
         Dates     Gef0d    Def0d    Bef0d        Gd       Dd       Bd      Gefd     Defd     Befd
        <char>     <num>    <num>    <num>     <num>    <num>    <num>     <num>    <num>    <num>
 1: Jan. 2024  6.348801 1.384087 4.825736  6.522313 1.440413 4.924221  6.104126 1.343455 4.621693
 2: Feb. 2024  6.680139 1.599929 4.933601  6.875052 1.672079 5.034287  6.406274 1.553670 4.705996
 3: Mar. 2024  7.104641 1.902356 5.060439  7.329138 1.998110 5.163713  6.788630 1.848127 4.798657
 4: Apr. 2024  8.578222 2.158071 6.280444  8.843422 2.265896 6.408617  8.295340 2.112064 6.043569
 5: May. 2024  9.885240 2.284334 7.465149 10.178196 2.394606 7.617499  9.688308 2.253942 7.298609
 6: Jun. 2024 11.293417 2.230338 8.920381 11.606533 2.329653 9.102430 11.115054 2.205314 8.767042
 7: Jul. 2024 11.495648 1.948530 9.397421 11.801533 2.029150 9.589205 11.308971 1.924962 9.234312
 8: Aug. 2024 10.493150 1.869393 8.468140 10.777404 1.947410 8.640959 10.196758 1.830334 8.210807
 9: Sep. 2024  8.584604 1.864962 6.564518  8.831006 1.948075 6.698488  8.228309 1.810198 6.262986
10: Oct. 2024  6.226290 1.631551 4.455104  6.418653 1.711039 4.546024  6.018374 1.595528 4.283212
11: Nov. 2024  6.076159 1.393353 4.545523  6.247341 1.452953 4.638289  5.875732 1.359514 4.378935
12: Dec. 2024  4.685547 1.198824 3.370992  4.825181 1.254616 3.439788  4.575893 1.179346 3.280817

Yearly values:
   Dates   Gef0d   Def0d    Bef0d       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>   <num>   <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 2973.115 654.591 2266.733 3058.651 684.4232 2312.993 2886.328 640.9157 2193.621
-----------------
Mode of tracking:  two
    Inclination limit: 90
```

El argumento **modeShd** puede ser de distintas maneras:

- **area**: el efecto de las sombras se calcula como una reducción proporcional de las irradiancias difusa circunsolar y directa.

```r
gef_shdarea <- calcGef(lat, modeTrk = 'two', dataRad = prom,
                       modeShd = 'area',
                       struct = struct, distances = distances)
show(gef_shdarea)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates    Gef0d    Def0d    Bef0d        Gd       Dd       Bd      Gefd     Defd     Befd
       <char>    <num>    <num>    <num>     <num>    <num>    <num>     <num>    <num>    <num>
 1: Jan. 2024  6.348801 1.384087 4.825736  6.522313 1.440413 4.924221  5.877879 1.305883 4.433019
 2: Feb. 2024  6.680139 1.599929 4.933601  6.875052 1.672079 5.034287  6.291348 1.534257 4.610483
 3: Mar. 2024  7.104641 1.902356 5.060439  7.329138 1.998110 5.163713  6.743478 1.840379 4.761253
 4: Apr. 2024  8.578222 2.158071 6.280444  8.843422 2.265896 6.408617  8.254928 2.105491 6.009730
 5: May. 2024  9.885240 2.284334 7.465149 10.178196 2.394606 7.617499  9.660175 2.249601 7.274817
 6: Jun. 2024 11.293417 2.230338 8.920381 11.606533 2.329653 9.102430 11.089573 2.201739 8.745137
 7: Jul. 2024 11.495648 1.948530 9.397421 11.801533 2.029150 9.589205 11.282303 1.921596 9.211011
 8: Aug. 2024 10.493150 1.869393 8.468140 10.777404 1.947410 8.640959 10.154416 1.824754 8.174045
 9: Sep. 2024  8.584604 1.864962 6.564518  8.831006 1.948075 6.698488  8.177410 1.802375 6.219910
10: Oct. 2024  6.226290 1.631551 4.455104  6.418653 1.711039 4.546024  5.950189 1.583714 4.226840
11: Nov. 2024  6.076159 1.393353 4.545523  6.247341 1.452953 4.638289  5.705306 1.330740 4.237284
12: Dec. 2024  4.685547 1.198824 3.370992  4.825181 1.254616 3.439788  4.440179 1.155239 3.169210

Yearly values:
   Dates    Gef0d   Def0d    Bef0d       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>    <num>   <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 2973.115 654.591 2266.733 3058.651 684.4232 2312.993 2856.633 636.0199 2168.822
-----------------
Mode of tracking:  two
    Inclination limit: 90
```

- **`prom`**: cuando **`modeTrk`** es **`two`**, se puede calcular el efecto de las sombras de un seguidor promedio.

```r
gef_shdprom <- calcGef(lat, modeTrk = 'two', dataRad = prom,
                              modeShd = c('area', 'prom'),
                              struct = struct, distances = distances)
show(gef_shdprom)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates    Gef0d    Def0d    Bef0d        Gd       Dd       Bd      Gefd     Defd     Befd
       <char>    <num>    <num>    <num>     <num>    <num>    <num>     <num>    <num>    <num>
 1: Jan. 2024  6.348801 1.384087 4.825736  6.522313 1.440413 4.924221  6.104126 1.343455 4.621693
 2: Feb. 2024  6.680139 1.599929 4.933601  6.875052 1.672079 5.034287  6.406274 1.553670 4.705996
 3: Mar. 2024  7.104641 1.902356 5.060439  7.329138 1.998110 5.163713  6.788630 1.848127 4.798657
 4: Apr. 2024  8.578222 2.158071 6.280444  8.843422 2.265896 6.408617  8.295340 2.112064 6.043569
 5: May. 2024  9.885240 2.284334 7.465149 10.178196 2.394606 7.617499  9.688308 2.253942 7.298609
 6: Jun. 2024 11.293417 2.230338 8.920381 11.606533 2.329653 9.102430 11.115054 2.205314 8.767042
 7: Jul. 2024 11.495648 1.948530 9.397421 11.801533 2.029150 9.589205 11.308971 1.924962 9.234312
 8: Aug. 2024 10.493150 1.869393 8.468140 10.777404 1.947410 8.640959 10.196758 1.830334 8.210807
 9: Sep. 2024  8.584604 1.864962 6.564518  8.831006 1.948075 6.698488  8.228309 1.810198 6.262986
10: Oct. 2024  6.226290 1.631551 4.455104  6.418653 1.711039 4.546024  6.018374 1.595528 4.283212
11: Nov. 2024  6.076159 1.393353 4.545523  6.247341 1.452953 4.638289  5.875732 1.359514 4.378935
12: Dec. 2024  4.685547 1.198824 3.370992  4.825181 1.254616 3.439788  4.575893 1.179346 3.280817

Yearly values:
```

```
   Dates    Gef0d   Def0d    Bef0d       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>    <num>   <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 2973.115 654.591 2266.733 3058.651 684.4232 2312.993 2886.328 640.9157 2193.621
-----------------
Mode of tracking:  two
    Inclination limit: 90
```

- **bt**: cuando **modeTrk** es **horiz**, se puede calcular el efecto del *backtracking* en las sombras.

```
gef_shdhoriz <- calcGef(lat, modeTrk = 'horiz', dataRad = prom,
                        modeShd = 'area',
                        struct = struct, distances = distances)
show(gef_shdhoriz)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates     Gef0d     Def0d    Bef0d        Gd       Dd       Bd      Gefd      Defd     Befd
       <char>     <num>     <num>    <num>     <num>    <num>    <num>     <num>     <num>    <num>
 1: Jan. 2024  4.274445 1.0909303 3.118987  4.528022 1.166334 3.285391  3.826940 1.0166151 2.745797
 2: Feb. 2024  5.173537 1.3974587 3.699745  5.414413 1.484046 3.839622  4.709780 1.3191237 3.314324
 3: Mar. 2024  6.270377 1.8008592 4.379272  6.512568 1.906181 4.498391  5.856407 1.7298195 4.036342
 4: Apr. 2024  8.160354 2.1103041 5.938446  8.429640 2.222836 6.072611  7.744288 2.0426359 5.590049
 5: May. 2024  9.639011 2.2544315 7.260788  9.932830 2.366831 7.416258  9.158384 2.1802588 6.854334
 6: Jun. 2024 11.005388 2.1942042 8.675874 11.320680 2.294944 8.861907 10.355140 2.1029750 8.116855
 7: Jul. 2024 11.220872 1.9183453 9.163290 11.527430 2.000253 9.358648 10.747413 1.8585724 8.749603
 8: Aug. 2024 10.066277 1.8239013 8.112148 10.352216 1.904515 8.290847  9.601132 1.7626031 7.708301
 9: Sep. 2024  7.732062 1.7621525 5.864625  7.991813 1.852070 6.013507  7.317424 1.6984219 5.513717
10: Oct. 2024  5.023316 1.4757157 3.471271  5.250215 1.568278 3.591050  4.691499 1.4182254 3.196944
11: Nov. 2024  4.211801 1.1318865 3.014748  4.452659 1.209397 3.166130  3.846165 1.0701542 2.710845
12: Dec. 2024  3.024846 0.9640813 2.008270  3.237139 1.039367 2.135901  2.849995 0.9330218 1.864479

Yearly values:
   Dates    Gef0d    Def0d    Bef0d       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 2618.414 607.6589 1975.038 2714.415 640.9193 2030.645 2463.159 583.5528 1843.889
-----------------
Mode of tracking:  horiz
    Inclination limit: 90
```

```
gef_shdbt <- calcGef(lat, modeTrk = 'horiz', dataRad = prom,
                        modeShd = c('area', 'bt'),
                        struct = struct, distances = distances)
show(gef_shdbt)
```

```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates       Bod       Bnd        Gd       Dd       Bd      Gefd      Defd     Befd
```

```
      <char>     <num>     <num>     <num>    <num>    <num>     <num>     <num>    <num>
 1: Jan. 2024  8.071623  4.924221  4.069604 1.101792 2.902196  3.802336 1.0232875 2.724604
 2: Feb. 2024 10.170791  5.034287  4.943127 1.417056 3.445443  4.680459 1.3258434 3.287780
 3: Mar. 2024 12.816149  5.163713  6.094523 1.850253 4.148386  5.841685 1.7419635 4.020914
 4: Apr. 2024 15.326568  6.408617  8.007438 2.166491 5.716983  7.711198 2.0485357 5.560571
 5: May. 2024 16.624320  7.617499  9.439815 2.303156 7.000336  9.132906 2.1878882 6.833933
 6: Jun. 2024 17.408383  9.102430 10.652929 2.206022 8.288629 10.286974 2.0977541 8.059004
 7: Jul. 2024 16.861601 10.037233 11.038213 1.944739 8.935057 10.701158 1.8585291 8.712900
 8: Aug. 2024 15.551202  8.640959  9.872463 1.850828 7.878525  9.562356 1.7662720 7.678732
 9: Sep. 2024 13.422796  6.698488  7.568105 1.795358 5.655421  7.285297 1.7012821 5.487114
10: Oct. 2024 10.764846  4.546024  4.915408 1.521915 3.310678  4.666904 1.4246602 3.173452
11: Nov. 2024  8.434950  4.638289  4.079866 1.156410 2.854293  3.813241 1.0737415 2.681776
12: Dec. 2024  7.370928  3.439788  3.062505 1.023011 1.987550  2.836653 0.9441838 1.849321

Yearly values:
   Dates      Bod      Bnd       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 4662.615 2326.882 2555.869 620.2896 1896.422 2451.499 585.4392 1833.809
-----------------
Mode of tracking:  horiz
    Inclination limit: 90
```

4.5. Producción eléctrica de un SFCR

Con la radiación efectiva, se puede estimar la producción eléctrica que va a tener un sistema fotovoltaico conectado a red. Esta estimación, se puede calcular mediante la función **`prodGCPV`** [??] la cual mediante la función **`fProd`** [??] procesa un objeto de clase **`Gef`** y obtiene un objeto **`ProdGCPV`**.

Como se puede ver en la figura 4.6, **`prodGCPV`** funciona gracias a la siguiente función:

- **`fProd`**: simula el comportamiento de un sistema fotovoltaico conectado a red bajo diferentes condiciones de temperatura e irradiancia. Tiene los siguientes argumentos:
  - **`inclin`**: puede ser tanto un objeto de clase **`Gef`** como un **`data.frame`** (o **`data.table`**). Sin embargo, si es un **`data.frame`**, debe contener como mínimo una columna para **`Gef`** y otra para **`Ta`**
  - **`module`**: una lista de valores numéricos con la información sobre el módulo fotovoltaico:
    - **`Vocn`**: tensión de circuito abierto en STC (V^*_{oc})(condiciones estandar de médida). Por defecto, tiene un valor de 57,2V.

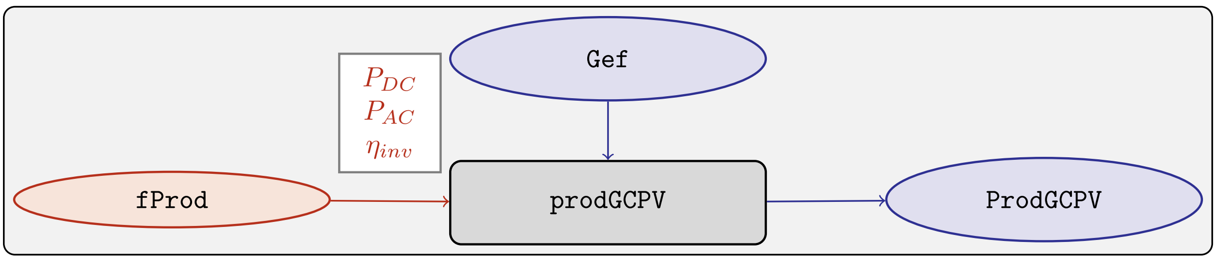


Figura 4.6: *Estimación de la producción eléctrica de un SFCR mediante la función* **`prodGCPV`**, *la cual emplea la función* **`fProd`** *para el computo de la potencia a la entrada (P_{DC}), a la salida (P_{AC}) y el rendimiento (η_{inv}) del inversor.*

- `Iscn`: corriente de cortocircuito en STC (I^*_{sc}). Por defecto, tiene un valor de $4{,}7A$.
- `Vmn`: tensión en el punto de máxima potencia en STC (I^*_{MPP}). Por defecto, tiene un valor de $46{,}08V$.
- `Imn`: corriente de cortocircuito en STC (I^*_{MPP}). Por defecto, tiene un valor de $4{,}35A$).
- `Ncs`: número de células en serie dentro del módulo. Por defecto, tiene un valor de 96.
- `Ncp`: número de células en paralelo dentro del módulo. Por defecto, tiene un valor de 1.
- `CoefVT`: coeficiente de disminución de la tensión de cada célula con la temperatura (dV_{oc}/dT_c). Por defecto, tiene un valor de $-0{,}0023V/°C$.
- `TONC`: temperatura de operación nominal de célula ($TONC$). Por defecto, tiene un valor de $47°C$.

* `generator`: lista de valores numéricos con la información sobre el generador:
  - `Nms`: número de módulos en serie. Por defecto, tiene un valor de 12.
  - `Nmp`: número de módulos en paralelo. Por defecto, tiene un valor de 11.
* `inverter`: lista de valores númericos con la información del inversor DC/AC.
  - `Ki`: coeficientes de la curva de eficiencia del inversor. Se puede presentar en un vector de 3 valores (por defecto, `c(0.01, 0.025, 0.05)`) o una matriz de 9 valores (si tiene dependencia del voltage).
  - `Pinv`: potencia nominal del inversor. Por defecto, tiene un valor de $25000W$.
  - `Vmin`: mínima tensión del rango MPP del inversor. Por defecto, tiene un valor de $420V$.
  - `Vmax`: máxima tensión del rango MPP del inversor. Por defecto, tiene un valor de $750V$.
  - `Gumb`: irradiancia umbral de funcionamienot del inversor. Por defecto, tiene un valor de $20W/m^2$.
* `effSys`: una lista de valores numéricos con la información sobre las pérdidas del sistema.
  - `ModQual`: tolerancia media del set de módulos (%). Por defecto, tiene un valor de 3.
  - `ModDisp`: pérdidas por dispersión en los módulos (%). Por defecto, tiene un valor de 2.
  - `OhmDC`: pérdidas por efecto Joule en el cableado de DC (%). Por defecto, tiene un valor de 1.5.
  - `OhmAC`: pérdidas por efecto Joule en el cableado de AC (%). Por defecto, tiene un valor de 1.5.
  - `MPP`: error promedio del algoritmo de búsqueda del MPP del inversor (%). Por defecto, tiene un valor de 1.
  - `TrafoMT`: pérdidas por el transformador MT (%). Por defecto, tiene un valor de 1.
  - `Disp`: pérdidas por las paradas del sistema (%). Por defecto, tiene un valor de 0.5.

```r
inclin <- calcGef(lat, dataRad = prom, keep.night = FALSE)
module <- list(Vocn=57.6, Iscn=4.7, Vmn=46.08, Imn=4.35,
               Ncs=96, Ncp=1, CoefVT=0.0023, TONC=47)
```

```
generator <- list(Nms=12, Nmp=11)
inverter <- list(Ki=c(0.01, 0.025, 0.05), Pinv=25000,
                 Vmin=420, Vmax=750, Gumb=20)
effSys <- list(ModQual=3, ModDisp=2, OhmDC=1.5, OhmAC=1.5,
               MPP=1, TrafoMT=1, Disp=0.5)
prod <- fProd(inclin = inclin, module = module,
              generator = generator, inverter = inverter,
              effSys = effSys)
show(prod)
```

```
                   Dates       Tc      Voc       Isc     Vmpp      Impp      Vdc       Idc       Pac
                  <POSc>    <num>    <num>     <num>    <num>     <num>    <num>     <num>     <num>
  1: 2024-01-17 08:00:00 15.27689 716.9624  8.083413 607.4640  7.620135 607.4640  7.620135  3796.209
  2: 2024-01-17 09:00:00 21.43284 700.6516 17.513415 583.9663 16.433741 583.9663 16.433741  8053.912
  3: 2024-01-17 10:00:00 27.23609 685.2753 26.403138 562.0190 24.658263 562.0190 24.658263 11650.920
  4: 2024-01-17 11:00:00 31.48724 674.0114 32.915263 546.0746 30.625265 546.0746 30.625265 14041.629
  5: 2024-01-17 12:00:00 33.33104 669.1261 35.739693 539.1958 33.196772 539.1958 33.196772 15016.481
 ---
141: 2024-12-13 12:00:00 33.94967 667.4869 28.721724 542.4718 26.706186 542.4718 26.706186 12177.570
142: 2024-12-13 13:00:00 32.55186 671.1906 26.580476 547.6944 24.746716 547.6944 24.746716 11395.331
143: 2024-12-13 14:00:00 29.08872 680.3665 21.275466 560.6878 19.868077 560.6878 19.868077  9362.088
144: 2024-12-13 15:00:00 24.29331 693.0724 13.929608 578.8034 13.059814 578.8034 13.059814  6316.091
145: 2024-12-13 16:00:00 19.21305 706.5331  6.147403 598.1441  5.786102 598.1441  5.786102  2784.663
           Pdc      EffI
         <num>     <num>
  1:  4290.940 0.9118076
  2:  8895.974 0.9330800
  3: 12846.437 0.9347232
  4: 15502.477 0.9335163
  5: 16592.492 0.9327431
 ---
141: 13429.451 0.9345615
142: 12563.918 0.9347755
143: 10326.335 0.9343983
144:  7007.083 0.9290019
145:  3208.198 0.8945754
```

Esta función brinda estos datos a la función **prodGCPV** para que produzca un objeto de clase **ProdGCPV** como resultado. Esta función tiene como argumentos principales los mismo que **calcGef**, ya que parte de un objeto tipo **Gef**, y los argumentos de la función **fProd**.

```
prodFixed <- prodGCPV(lat, modeTrk = 'fixed', dataRad = prom)
show(prodFixed)
```

```
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
       Dates       Eac       Edc       Yf
      <char>     <num>     <num>    <num>
 1: Jan. 2024  95.36291 105.62767 3.604158
 2: Feb. 2024 101.50809 112.56166 3.836410
 3: Mar. 2024 110.26945 122.11835 4.167538
 4: Apr. 2024 124.53728 138.29836 4.706778
 5: May. 2024 131.48629 145.91065 4.969410
 6: Jun. 2024 135.89421 150.78725 5.136003
 7: Jul. 2024 134.98501 149.81246 5.101641
 8: Aug. 2024 130.25804 144.39951 4.922989
```

```
 9: Sep. 2024 119.91911 132.77648 4.532238
10: Oct. 2024  96.49455 106.99182 3.646928
11: Nov. 2024  90.17737  99.88152 3.408175
12: Dec. 2024  73.89289  81.80967 2.792718

Yearly values:
   Dates     Eac      Edc       Yf
   <int>   <num>    <num>    <num>
1:  2024 41014.8 45473.37 1550.119
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5
```

```
prod2x <- prodGCPV(lat, modeTrk = 'two', dataRad = prom)
show(prod2x)
```

```
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Eac      Edc       Yf
       <char>    <num>    <num>    <num>
 1: Jan. 2024 138.6806 153.2566 5.241314
 2: Feb. 2024 143.4987 158.5247 5.423408
 3: Mar. 2024 151.8477 167.7311 5.738952
 4: Apr. 2024 178.6717 197.4274 6.752741
 5: May. 2024 200.8888 222.0523 7.592419
 6: Jun. 2024 223.9959 247.6903 8.465728
 7: Jul. 2024 214.2749 236.9628 8.098332
 8: Aug. 2024 194.6043 215.1439 7.354902
 9: Sep. 2024 168.9824 186.7349 6.386542
10: Oct. 2024 132.2995 146.0747 5.000145
11: Nov. 2024 128.5783 141.9871 4.859507
12: Dec. 2024 102.9116 113.5613 3.889454

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2024 60369.04 66710.67 2281.595
-----------------
Mode of tracking:  two
    Inclination limit: 90
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5
```

```
prodHoriz <- prodGCPV(lat, modeTrk = 'horiz', dataRad = prom)
show(prodHoriz)
```

```
Object of class  ProdGCPV
```

```
Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates       Eac       Edc       Yf
       <char>     <num>     <num>    <num>
 1: Jan. 2024  99.43006 109.66074 3.757873
 2: Feb. 2024 116.24796 128.22238 4.393490
 3: Mar. 2024 137.39485 151.61074 5.192719
 4: Apr. 2024 172.03044 189.97488 6.501741
 5: May. 2024 196.91337 217.61396 7.442169
 6: Jun. 2024 219.15566 242.31468 8.282797
 7: Jul. 2024 210.33644 232.56087 7.949482
 8: Aug. 2024 189.03576 208.87993 7.144442
 9: Sep. 2024 156.22909 172.44519 5.904542
10: Oct. 2024 110.69482 122.11859 4.183614
11: Nov. 2024  94.40734 104.14723 3.568043
12: Dec. 2024  69.94550  77.30532 2.643529

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2024 54052.14 59697.16 2042.854
-----------------
Mode of tracking:  horiz
    Inclination limit: 90
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5
```

4.6. Producción eléctrica de un SFB

De igual forma que en el apartado anterior, se puede estimar la producción eléctrica de un sistema fotovoltaico de bombeo.

Como se puede ver en la figura 4.7, **prodPVPS** funciona gracias a la siguiente función:

- **fPump**: calcula el rendimiento de las diferentes partes de una bomba centrífuga alimentada por un convertidor de frecuencia siguiendo las leyes de afinidad. Tiene solo dos argumentos:
  - **pump**: lista que contiene los parametros de la bomba que va a ser simulada. Puede ser una fila de **pumpCoef**:

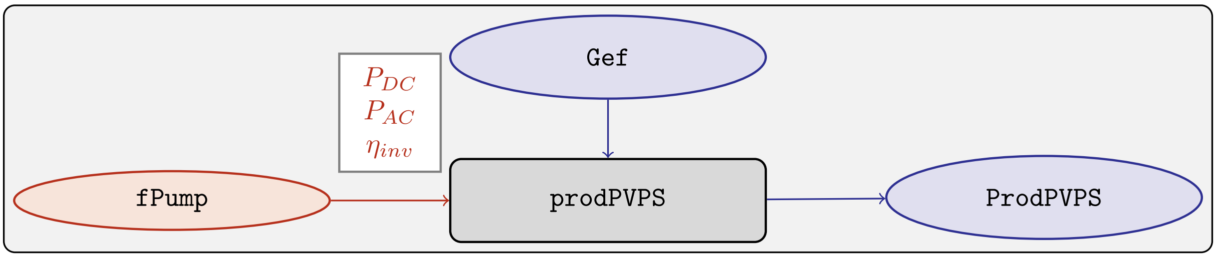


FIGURA 4.7: *Estimación de la producción eléctrica de un SFB mediante la función* ***prodPVPS****, la cual emplea la función* ***fPump*** *para el computo del rendimiento de las diferentes parte de una bomba centrífuga alimentada por un convertidor de frecuencia.*

```
CoefSP8A44 <- pumpCoef[Qn == 8 & stages == 44]
show(CoefSP8A44)
```

```
      Qn stages  Qmax   Pmn         a         b      c     g     h     i       j     k      l
   <int>  <int> <num> <int>     <num>     <num>  <num> <num> <num> <num>   <num> <num>  <num>
1:     8     44    12  7500 0.1043011 -0.101288 -0.726 -0.24  0.42  0.64 -0.0058 0.095 0.2013
```

- `H`: el salto manometrico total.

```
fSP8A44 <- fPump(pump = CoefSP8A44, H = 40)
```

Obtiene como resultado los siguientes valores y funciones:

- `lim`: rango de valores de la potencia eléctrica de salida.

```
show(fSP8A44$lim)
```

```
[1]  190.100 4084.218
```

- `fQ`: función que relaciona el caudal con la potencia eléctrica.

```
show(fSP8A44$fQ)
```

```
function (x, deriv = 0L)
{
    deriv <- as.integer(deriv)
    if (deriv < 0L || deriv > 3L)
        stop("'deriv' must be between 0 and 3")
    if (deriv > 0L) {
        z0 <- double(z$n)
        z[c("y", "b", "c")] <- switch(deriv, list(y = z$b, b = 2 *
            z$c, c = 3 * z$d), list(y = 2 * z$c, b = 6 * z$d,
            c = z0), list(y = 6 * z$d, b = z0, c = z0))
        z[["d"]] <- z0
    }
    res <- .splinefun(x, z)
    if (deriv > 0 && z$method == 2 && any(ind <- x <= z$x[1L]))
        res[ind] <- ifelse(deriv == 1, z$y[1L], 0)
    res
}
<bytecode: 0x00000241cd298748>
<environment: 0x00000241cd29cda0>
```

- `fPb`: función que relaciona la potencia del eje de la bomba con la potencia eléctrica del motor.

```
show(fSP8A44$fPb)
```

```
function (x, deriv = 0L)
{
    deriv <- as.integer(deriv)
    if (deriv < 0L || deriv > 3L)
        stop("'deriv' must be between 0 and 3")
    if (deriv > 0L) {
        z0 <- double(z$n)
        z[c("y", "b", "c")] <- switch(deriv, list(y = z$b, b = 2 *
            z$c, c = 3 * z$d), list(y = 2 * z$c, b = 6 * z$d,
            c = z0), list(y = 6 * z$d, b = z0, c = z0))
        z[["d"]] <- z0
    }
    res <- .splinefun(x, z)
    if (deriv > 0 && z$method == 2 && any(ind <- x <= z$x[1L]))
```

```
        res[ind] <- ifelse(deriv == 1, z$y[1L], 0)
    res
}
<bytecode: 0x00000241cd298748>
<environment: 0x00000241cd311e08>
```

- **fPh**: función que relaciona la potencia hidráulica con la potencia eléctrica del motor.

```
show(fSP8A44$fPh)
```

```
function (x, deriv = 0L)
{
    deriv <- as.integer(deriv)
    if (deriv < 0L || deriv > 3L)
        stop("'deriv' must be between 0 and 3")
    if (deriv > 0L) {
        z0 <- double(z$n)
        z[c("y", "b", "c")] <- switch(deriv, list(y = z$b, b = 2 *
            z$c, c = 3 * z$d), list(y = 2 * z$c, b = 6 * z$d,
            c = z0), list(y = 6 * z$d, b = z0, c = z0))
        z[["d"]] <- z0
    }
    res <- .splinefun(x, z)
    if (deriv > 0 && z$method == 2 && any(ind <- x <= z$x[1L]))
        res[ind] <- ifelse(deriv == 1, z$y[1L], 0)
    res
}
<bytecode: 0x00000241cd298748>
<environment: 0x00000241cd32cd30>
```

- **fFreq**: función que relaciona la frecuencia con la potencia eléctrica del motor.

```
show(fSP8A44$fFreq)
```

```
function (x, deriv = 0L)
{
    deriv <- as.integer(deriv)
    if (deriv < 0L || deriv > 3L)
        stop("'deriv' must be between 0 and 3")
    if (deriv > 0L) {
        z0 <- double(z$n)
        z[c("y", "b", "c")] <- switch(deriv, list(y = z$b, b = 2 *
            z$c, c = 3 * z$d), list(y = 2 * z$c, b = 6 * z$d,
            c = z0), list(y = 6 * z$d, b = z0, c = z0))
        z[["d"]] <- z0
    }
    res <- .splinefun(x, z)
    if (deriv > 0 && z$method == 2 && any(ind <- x <= z$x[1L]))
        res[ind] <- ifelse(deriv == 1, z$y[1L], 0)
    res
}
<bytecode: 0x00000241cd298748>
<environment: 0x00000241cd2be128>
```

Se pueden realizar operaciones con este objeto:

```
SP8A44 = with(fSP8A44,{
  Pac = seq(lim[1],lim[2],l=10)
  Pb = fPb(Pac)
  etam = Pb/Pac
  Ph = fPh(Pac)
  etab = Ph/Pb
  f = fFreq(Pac)
  Q = fQ(Pac)
  result = data.table(Q,Pac,Pb,Ph,etam,etab,f)})
show(SP8A44)
```

```
             Q       Pac        Pb         Ph      etam      etab        f
         <num>     <num>     <num>      <num>     <num>     <num>    <num>
 1:  0.3133325  190.1000  124.8346   34.15325 0.6566786 0.2735880 20.47033
 2:  2.0718468  622.7798  429.6728  225.83130 0.6899274 0.5255890 22.33036
 3:  4.0764128 1055.4595  752.8970  444.32900 0.7133358 0.5901591 25.51459
 4:  5.6406747 1488.1393 1087.3665  614.83354 0.7306887 0.5654336 28.73213
 5:  6.9474993 1920.8190 1429.7984  757.27743 0.7443692 0.5296393 31.78514
 6:  8.1028841 2353.4988 1778.0156  883.21437 0.7554776 0.4967416 34.69527
 7:  9.1607296 2786.1786 2130.4683  998.51953 0.7646560 0.4686855 37.49608
 8: 10.1514390 3218.8583 2486.0213 1106.50685 0.7723301 0.4450915 40.21428
 9: 11.0937480 3651.5381 2843.8295 1209.21854 0.7788032 0.4252078 42.86977
10: 12.0000000 4084.2179 3203.2578 1308.00000 0.7843014 0.4083343 45.47737
```

Está función entrega todos estos resultados a **`prodPVPS`** la cual calcula los resultados en base a la potencia del generador a simular, y devuleve un objeto de clase **`ProdPVPS`**.

```r
prodsfb <- prodPVPS(lat, modeTrk = 'fixed', dataRad = prom,
                    pump = CoefSP8A44, H = 40, Pg = SP8A44$Pac[10])
show(prodsfb)
```

```
Object of class  ProdPVPS

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Eac       Qd       Yf
       <char>    <num>    <num>    <num>
 1: Jan. 2024 14.07129 50.46621 3.445284
 2: Feb. 2024 15.43701 54.71213 3.779675
 3: Mar. 2024 17.00102 59.68995 4.162613
 4: Apr. 2024 19.39135 67.24260 4.747874
 5: May. 2024 20.65046 71.34195 5.056160
 6: Jun. 2024 21.63947 74.27359 5.298315
 7: Jul. 2024 22.62915 76.77927 5.540633
 8: Aug. 2024 22.17136 75.07166 5.428546
 9: Sep. 2024 19.61622 67.34348 4.802932
10: Oct. 2024 14.92078 53.24853 3.653277
11: Nov. 2024 13.75298 49.50040 3.367348
12: Dec. 2024 11.21349 40.90244 2.745567

Yearly values:
   Dates      Eac       Qd       Yf
   <int>    <num>    <num>    <num>
1:  2024 6482.059 22589.95 1587.099
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Pump:
    Qn:  8
    Stages:  44
Height (m):  40
Generator (Wp):  4084.218
```

4.7. Optimización de distancias

Por último, el paquete **`solaR2`** contiene una función que permite calcular un conjunto de combinaciones de distancias entre los elementos de un sistema fotovoltaico conectado a red, con el fin de que el usuario posteriormente pueda optar cual es la opción mas rentable en base a los precios del cableado y de la ocupación del terreno.

Esta función es **optimShd**, la cual en base a una resolución (determinada por el argumento **res**, el cual, indica el incremento de la secuencia de distancias) obtiene la producción de cada combinación y la plasma en un objeto de clase **Shade**.

```r
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, dataRad = prom, modeTrk = 'two',
                   modeShd = c('area', 'prom'),
                   distances = dist2x, struct = struct2x,
                   res = 5,
                   prog = FALSE) #Se quita la barra de progreso
show(ShdM2x)
```

```
Object of class  Shade

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
Dimensions of structure:
$W
[1] 23.11

$L
[1] 9.8

$Nrow
[1] 2

$Ncol
[1] 3

Shade calculation mode:
[1] "area" "prom"
Productivity without shadows:
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Eac      Edc       Yf
       <char>    <num>    <num>    <num>
 1: Jan. 2024 138.6806 153.2566 5.241314
 2: Feb. 2024 143.4987 158.5247 5.423408
 3: Mar. 2024 151.8477 167.7311 5.738952
 4: Apr. 2024 178.6717 197.4274 6.752741
 5: May. 2024 200.8888 222.0523 7.592419
 6: Jun. 2024 223.9959 247.6903 8.465728
 7: Jul. 2024 214.2749 236.9628 8.098332
 8: Aug. 2024 194.6043 215.1439 7.354902
 9: Sep. 2024 168.9824 186.7349 6.386542
10: Oct. 2024 132.2995 146.0747 5.000145
11: Nov. 2024 128.5783 141.9871 4.859507
12: Dec. 2024 102.9116 113.5613 3.889454

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2024 60369.04 66710.67 2281.595
```

```
-----------------
Mode of tracking:  two
    Inclination limit: 90
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5

Summary of results:
      Lew              Lns          H           FS                GRR              Yf
 Min.   :30.00   Min.   :20   Min.   :0   Min.   :0.01509   Min.   :2.649   Min.   :2104
 1st Qu.:33.75   1st Qu.:25   1st Qu.:0   1st Qu.:0.02223   1st Qu.:3.946   1st Qu.:2192
 Median :37.50   Median :30   Median :0   Median :0.02870   Median :4.802   Median :2216
 Mean   :37.50   Mean   :30   Mean   :0   Mean   :0.03463   Mean   :4.967   Mean   :2203
 3rd Qu.:41.25   3rd Qu.:35   3rd Qu.:0   3rd Qu.:0.03945   3rd Qu.:6.016   3rd Qu.:2231
 Max.   :45.00   Max.   :40   Max.   :0   Max.   :0.07769   Max.   :7.948   Max.   :2247
```

```r
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',
                         prog = FALSE) #Se quita la barra de progreso
show(Shd12HorizBT)
```

```
Object of class  Shade

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
Dimensions of structure:
$L
[1] 4.83

Shade calculation mode:
[1] "bt"
Productivity without shadows:
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates       Eac       Edc       Yf
       <char>     <num>     <num>    <num>
 1: Jan. 2024  97.48365 107.53823 3.684309
 2: Feb. 2024 114.31569 126.11751 4.320462
 3: Mar. 2024 135.67629 149.74056 5.127767
 4: Apr. 2024 170.28530 188.06424 6.435785
 5: May. 2024 194.83600 215.33865 7.363657
 6: Jun. 2024 216.37522 239.26256 8.177713
 7: Jul. 2024 208.20413 230.21074 7.868894
 8: Aug. 2024 187.19428 206.84745 7.074845
 9: Sep. 2024 154.37402 170.41035 5.834432
10: Oct. 2024 109.27362 120.57435 4.129901
11: Nov. 2024  92.82584 102.42576 3.508272
12: Dec. 2024  69.13228  76.42401 2.612794
```

```
Yearly values:
   Dates      Eac      Edc       Yf
   <int>     <num>    <num>    <num>
1:  2024 53386.77 58969.19 2017.707
-----------------
Mode of tracking:  horiz
    Inclination limit: 60
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5

Summary of results:
      Lew              H            FS                GRR               Yf
 Min.   : 9.66   Min.   :0   Min.   :0.04804   Min.   :2.000   Min.   :1736
 1st Qu.:13.16   1st Qu.:0   1st Qu.:0.05727   1st Qu.:2.725   1st Qu.:1824
 Median :16.66   Median :0   Median :0.07295   Median :3.449   Median :1871
 Mean   :16.66   Mean   :0   Mean   :0.08078   Mean   :3.449   Mean   :1855
 3rd Qu.:20.16   3rd Qu.:0   3rd Qu.:0.09598   3rd Qu.:4.174   3rd Qu.:1902
 Max.   :23.66   Max.   :0   Max.   :0.13968   Max.   :4.899   Max.   :1921
```

```r
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',
                              prog = FALSE) #Se quita la barra de progreso
show(Shd12Fixed)
```

```
Object of class  Shade

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
Dimensions of structure:
$L
[1] 5

Shade calculation mode:
[1] "area"
Productivity without shadows:
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates        Eac        Edc        Yf
       <char>      <num>      <num>     <num>
 1: Jan. 2024   95.36291 105.62767 3.604158
 2: Feb. 2024  101.50809 112.56166 3.836410
 3: Mar. 2024  110.26945 122.11835 4.167538
 4: Apr. 2024  124.53728 138.29836 4.706778
 5: May. 2024  131.48629 145.91065 4.969410
 6: Jun. 2024  135.89421 150.78725 5.136003
 7: Jul. 2024  134.98501 149.81246 5.101641
 8: Aug. 2024  130.25804 144.39951 4.922989
```

```
 9: Sep. 2024 119.91911 132.77648 4.532238
10: Oct. 2024  96.49455 106.99182 3.646928
11: Nov. 2024  90.17737  99.88152 3.408175
12: Dec. 2024  73.89289  81.80967 2.792718

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2024 41014.8 45473.37 1550.119
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5

Summary of results:
       D               H            FS                  GRR            Yf
 Min.   : 5.0   Min.   :0   Min.   :0.0008477   Min.   :1.0   Min.   :1364
 1st Qu.: 7.5   1st Qu.:0   1st Qu.:0.0015710   1st Qu.:1.5   1st Qu.:1511
 Median :10.0   Median :0   Median :0.0038992   Median :2.0   Median :1544
 Mean   :10.0   Mean   :0   Mean   :0.0269608   Mean   :2.0   Mean   :1508
 3rd Qu.:12.5   3rd Qu.:0   3rd Qu.:0.0252790   3rd Qu.:2.5   3rd Qu.:1548
 Max.   :15.0   Max.   :0   Max.   :0.1199180   Max.   :3.0   Max.   :1549
```

4.8. Métodos de visualización

Una vez creados todos los objetos, para mejorar la visualización de los mismos, **solaR2** cuanta con una serie de métodos que ayudan a la compresión de los datos obtenidos.

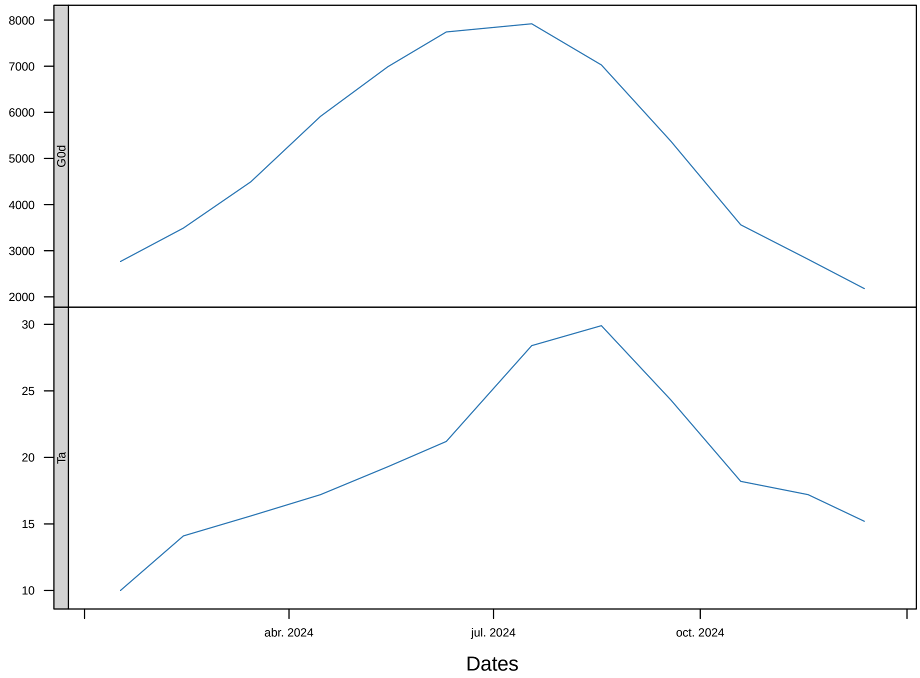
4.8.1. Datos meteorológicos

La clase **Meteo** cuenta con un método para **xyplot**.

```
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;
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)
show(BD)
```

```
Object of class  Meteo

Source of meteorological information: prom-
Latitude of source:  37.2 degrees

Meteorological Data:
     Dates                 G0d             Ta
 Min.   :2024-01-17   Min.   :2179   Min.   :10.00
 1st Qu.:2024-04-07   1st Qu.:3322   1st Qu.:15.50
 Median :2024-06-29   Median :4932   Median :17.70
 Mean   :2024-07-01   Mean   :5022   Mean   :19.22
 3rd Qu.:2024-09-25   3rd Qu.:6998   3rd Qu.:21.98
 Max.   :2024-12-13   Max.   :7919   Max.   :29.90
```

```
xyplot(BD)
```



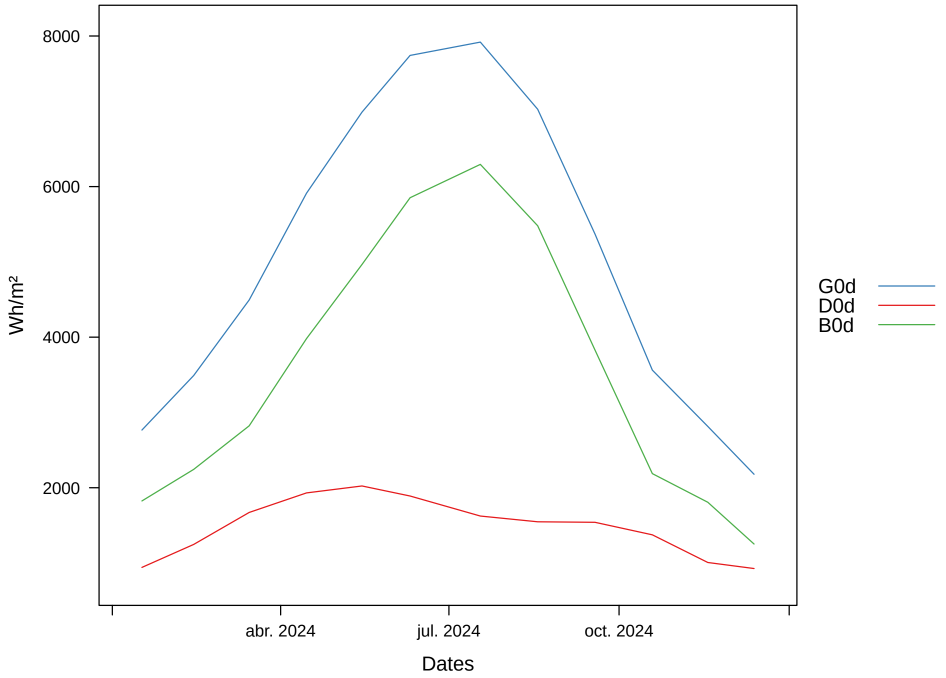
4.8.2. Radiación en el plano horizontal

La clase **G0** cuenta con un método para **xyplot**.

```
g0 <- calcG0(lat, dataRad = BD)
show(g0)
```

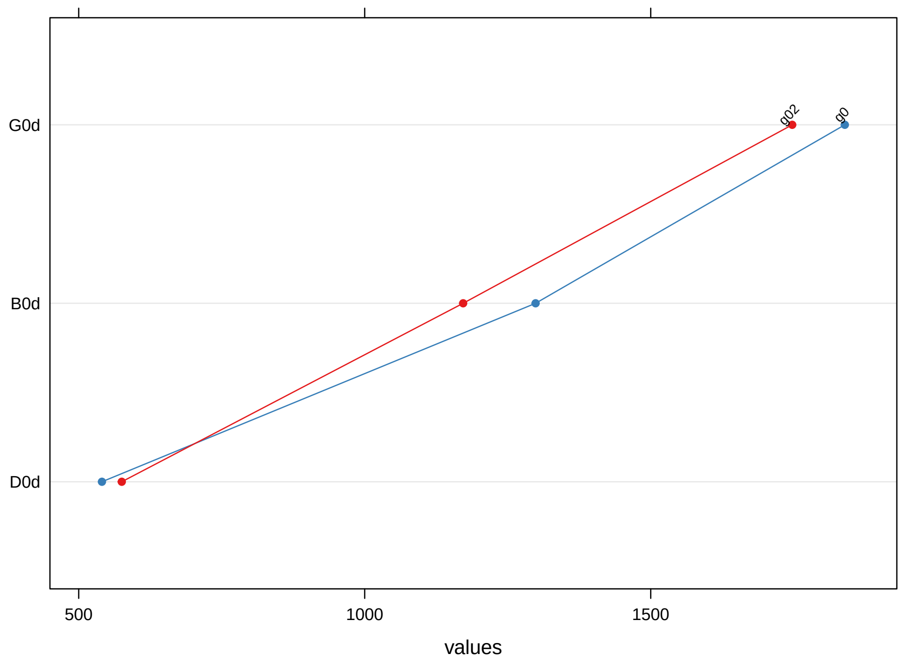
```
Object of class  G0

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates   G0d      D0d      B0d
       <char> <num>    <num>    <num>
 1: Jan. 2024 2.766 0.941698 1.824302
 2: Feb. 2024 3.491 1.247146 2.243854
 3: Mar. 2024 4.494 1.671763 2.822237
 4: Apr. 2024 5.912 1.931146 3.980854
 5: May. 2024 6.989 2.023364 4.965636
 6: Jun. 2024 7.742 1.889994 5.852006
 7: Jul. 2024 7.919 1.624064 6.294936
 8: Aug. 2024 7.027 1.547591 5.479409
 9: Sep. 2024 5.369 1.540708 3.828292
10: Oct. 2024 3.562 1.374513 2.187487
11: Nov. 2024 2.814 1.006959 1.807041
12: Dec. 2024 2.179 0.926737 1.252263

Yearly values:
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 540.6331 1298.732
```

```
xyplot(g0)
```



Y con un método para **`compare`**.

```
g02 <- calcG0(lat, dataRad = list(G0dm = G0dm*0.95, Ta = Ta))
compare(g0, g02)
```



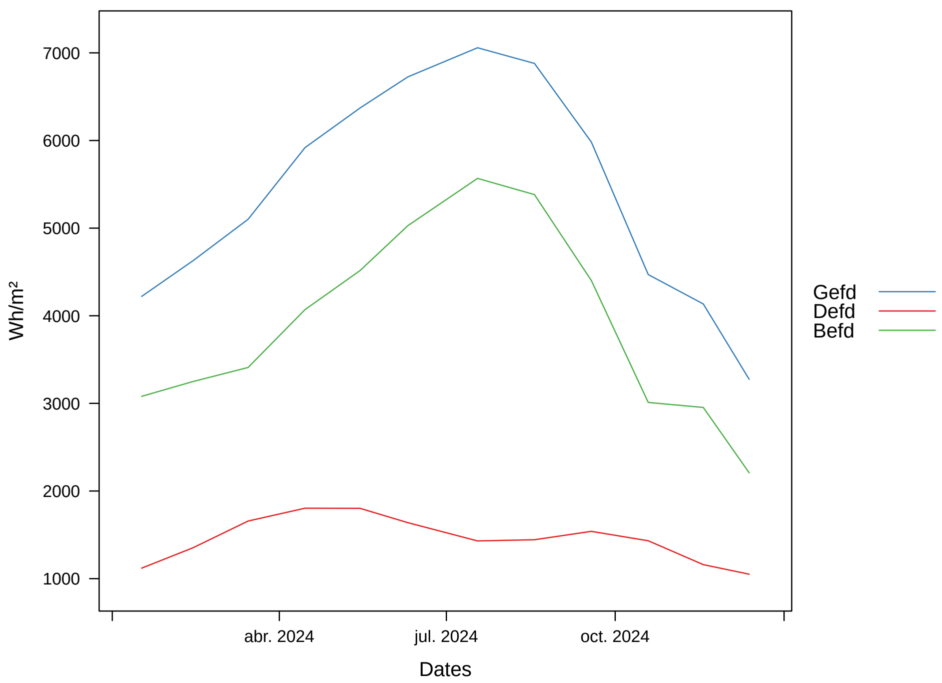
4.8.3. Radiación efectiva en el plano del generador

La clase **`Gef`** cuenta con un método para **`xyplot`**.

```
gef <- calcGef(lat, dataRad = BD)
show(gef)
```

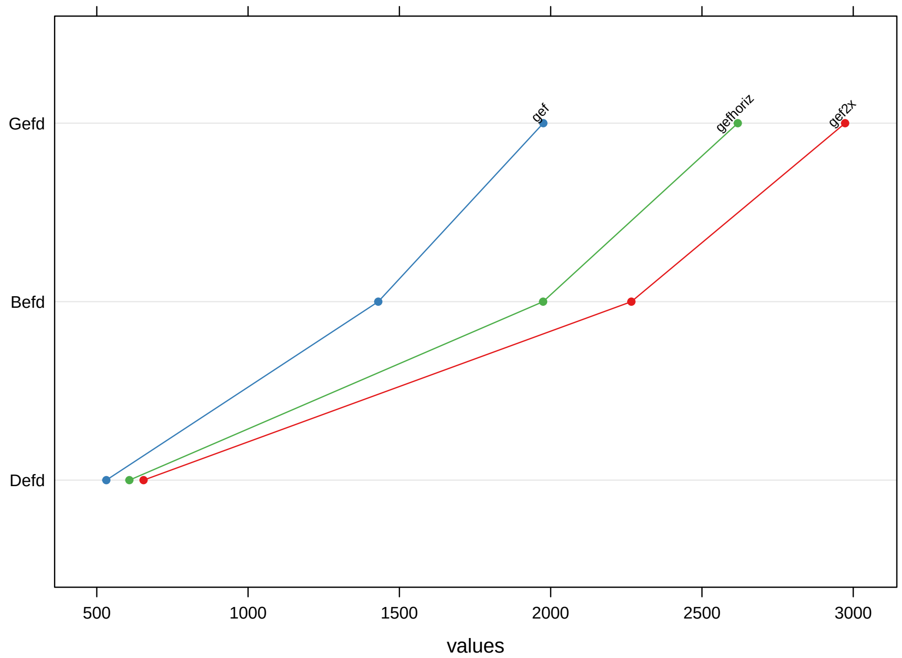
```
Object of class  Gef

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates       Bod       Bnd       Gd       Dd       Bd     Gefd     Defd     Befd
       <char>     <num>     <num>    <num>    <num>    <num>    <num>    <num>    <num>
 1: Jan. 2024  8.724907  4.924221 4.489744 1.200992 3.258164 4.220907 1.119517 3.080392
 2: Feb. 2024  9.592013  5.034287 4.919206 1.451954 3.428647 4.628492 1.352529 3.249460
 3: Mar. 2024 10.281308  5.163713 5.413543 1.779951 3.583896 5.101556 1.657369 3.410070
 4: Apr. 2024 10.527227  6.408617 6.282631 1.936897 4.280357 5.918787 1.803811 4.070094
 5: May. 2024 10.431853  7.617499 6.784202 1.937331 4.769584 6.371295 1.802060 4.516177
 6: Jun. 2024 10.291163  9.102430 7.173475 1.762326 5.325535 6.725684 1.639192 5.027718
 7: Jul. 2024 10.305302 10.037233 7.511733 1.533887 5.890275 7.058263 1.430322 5.567823
 8: Aug. 2024 10.394682  8.640959 7.295543 1.545089 5.672747 6.879777 1.443952 5.382478
 9: Sep. 2024 10.233884  6.698488 6.335591 1.647975 4.628244 5.982520 1.539552 4.402209
10: Oct. 2024  9.659077  4.546024 4.746760 1.538325 3.169044 4.470026 1.432213 3.010771
11: Nov. 2024  8.798687  4.638289 4.393712 1.244217 3.118376 4.134590 1.159756 2.953471
12: Dec. 2024  8.176298  3.439788 3.478125 1.128381 2.325648 3.274677 1.050626 2.207509

Yearly values:
   Dates      Bod      Bnd       Gd       Dd       Bd     Gefd     Defd     Befd
   <int>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>
1:  2024 3580.873 2326.882 2099.528 570.4317 1508.756 1975.745 531.5105 1430.271
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
```

```
xyplot(gef)
```



Y con un método para **compare**.

```
gef2x <- calcGef(lat, modeTrk = 'two', dataRad = BD)
gefhoriz <- calcGef(lat, modeTrk = 'horiz', dataRad = BD)
compare(gef, gef2x, gefhoriz)
```



4.8.4. Producción eléctrica de un SFCR

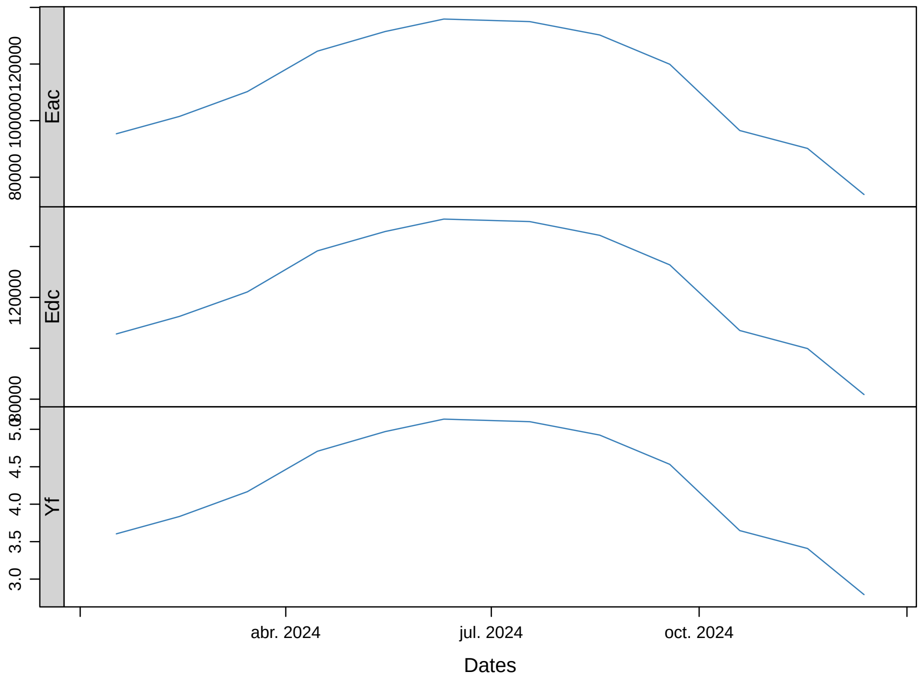
La clase **ProdGCPV** cuenta con un método para **xyplot**.

```
prodFixed <- prodGCPV(lat, modeTrk = 'fixed', dataRad = BD)
show(prodFixed)
```

```
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates       Eac       Edc       Yf
       <char>     <num>     <num>    <num>
 1: Jan. 2024  95.36291 105.62767 3.604158
 2: Feb. 2024 101.50809 112.56166 3.836410
 3: Mar. 2024 110.26945 122.11835 4.167538
 4: Apr. 2024 124.53728 138.29836 4.706778
 5: May. 2024 131.48629 145.91065 4.969410
 6: Jun. 2024 135.89421 150.78725 5.136003
 7: Jul. 2024 134.98501 149.81246 5.101641
 8: Aug. 2024 130.25804 144.39951 4.922989
 9: Sep. 2024 119.91911 132.77648 4.532238
10: Oct. 2024  96.49455 106.99182 3.646928
11: Nov. 2024  90.17737  99.88152 3.408175
12: Dec. 2024  73.89289  81.80967 2.792718
```

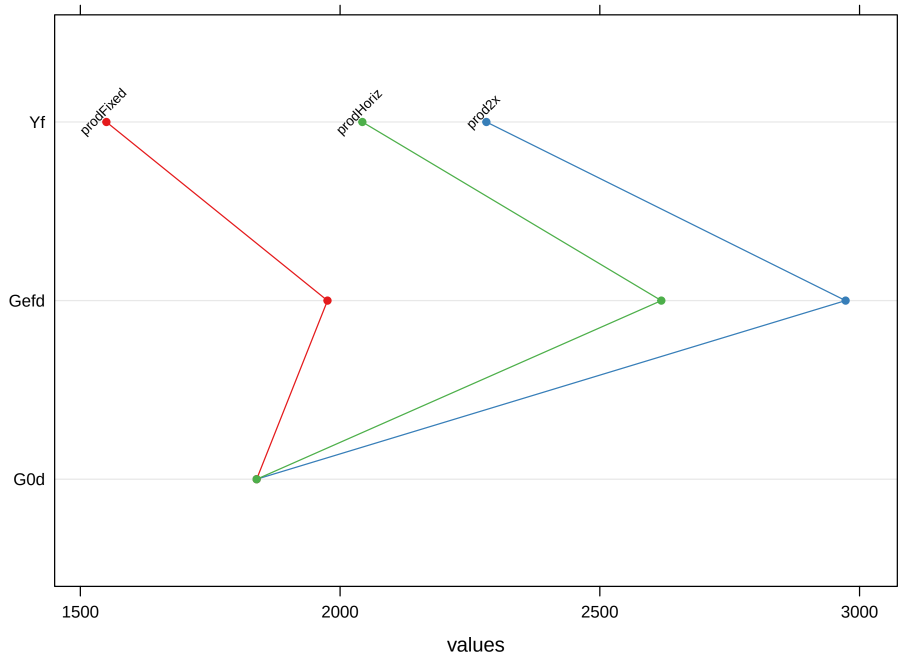
```
Yearly values:
   Dates      Eac       Edc        Yf
   <int>    <num>     <num>     <num>
1:  2024 41014.8 45473.37 1550.119
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5
```

```r
xyplot(prodFixed)
```



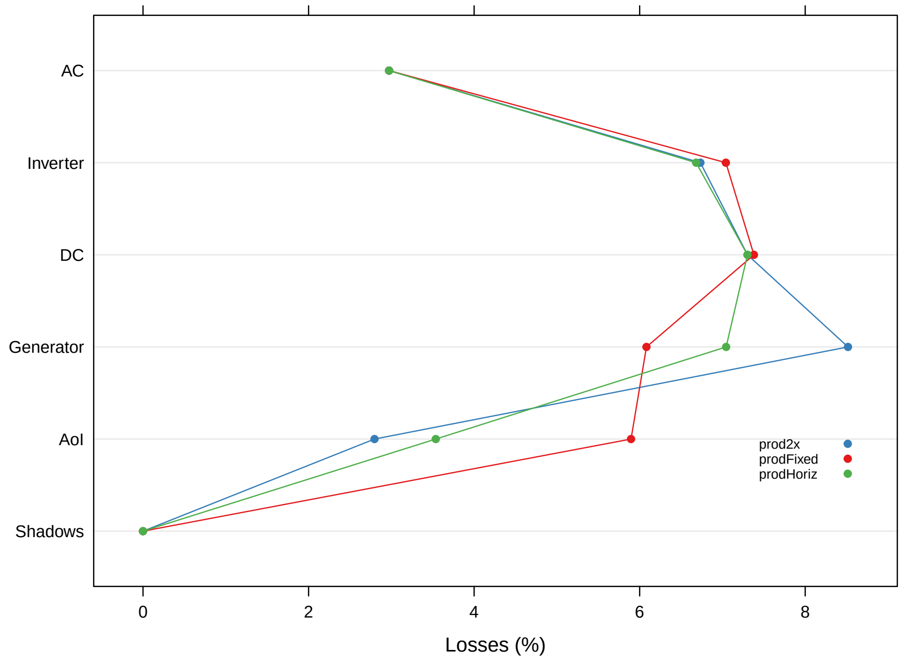
Un método para **compare**.

```r
prod2x <- prodGCPV(lat, modeTrk = 'two', dataRad = BD)
prodHoriz <- prodGCPV(lat, modeTrk = 'horiz', dataRad = BD)
compare(prodFixed, prod2x, prodHoriz)
```



Y un método para **compareLosses**.

```
compareLosses(prodFixed, prod2x, prodHoriz)
```



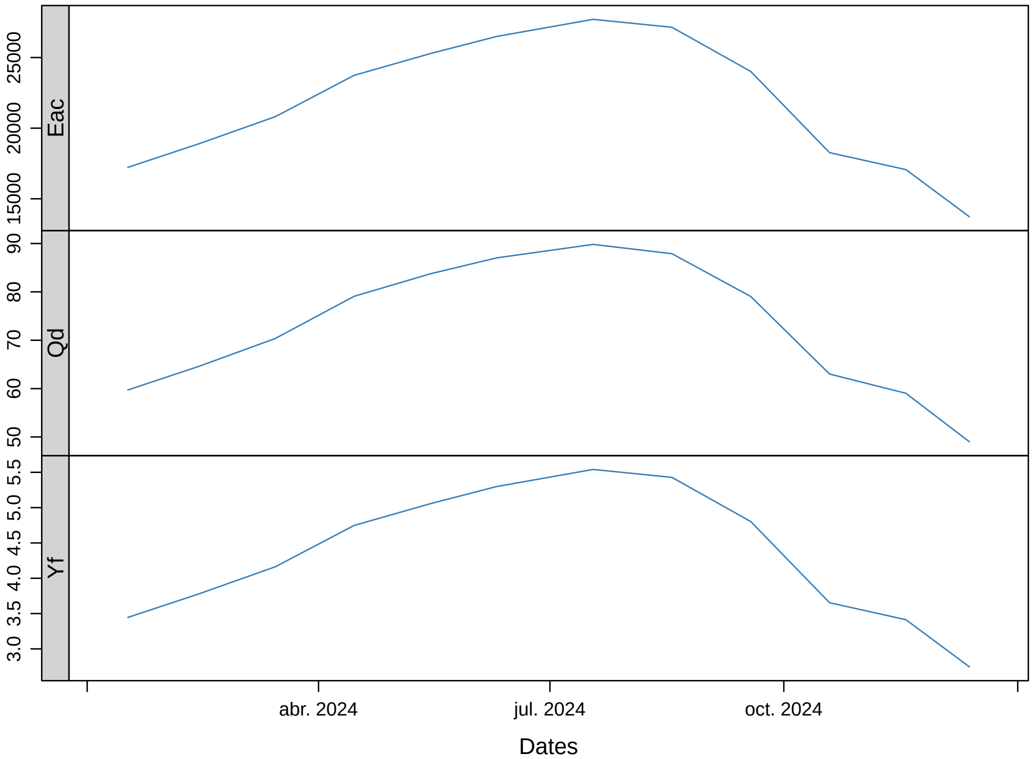
4.8.5. Producción electrica de un SFB

La clase **ProdPVPS** cuenta con un método para **xyplot**.

```
pump <- prodPVPS(lat, dataRad = BD, pump = CoefSP8A44, H = 40, Pg = 5000)
show(pump)
```

```
Object of class  ProdPVPS

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Eac        Qd       Yf
       <char>    <num>     <num>    <num>
 1: Jan. 2024 17.22642 59.71506 3.445284
 2: Feb. 2024 18.89837 64.60949 3.779675
 3: Mar. 2024 20.81307 70.36542 4.162613
 4: Apr. 2024 23.73937 79.08382 4.747874
 5: May. 2024 25.28080 83.74003 5.056160
 6: Jun. 2024 26.49158 87.02474 5.298315
 7: Jul. 2024 27.70317 89.81648 5.540633
 8: Aug. 2024 27.14273 87.89528 5.428546
 9: Sep. 2024 24.01466 79.04010 4.802932
10: Oct. 2024 18.26638 63.00860 3.653277
11: Nov. 2024 17.06794 59.03182 3.413588
12: Dec. 2024 13.72784 48.99686 2.745567

Yearly values:
   Dates      Eac       Qd       Yf
   <int>    <num>    <num>    <num>
1:  2024 7942.432 26608.76 1588.486
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Pump:
    Qn:  8
    Stages:  44
Height (m):  40
Generator (Wp):  5000
```

```
xyplot(pump)
```



4.8.6. Optimización de distancias

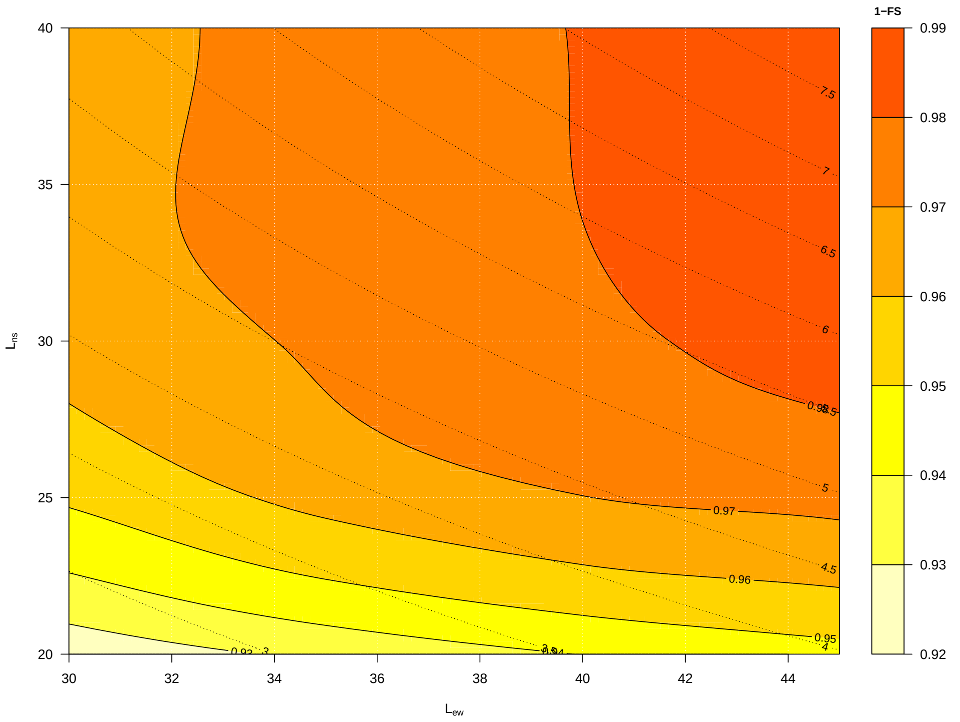
La clase **Shade** cuenta con un método para **shadeplot**.

```
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, prog = FALSE)
show(ShdM2x)
```

```
Object of class  Shade

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
Dimensions of structure:
$W
[1] 23.11

$L
[1] 9.8

$Nrow
[1] 2

$Ncol
[1] 3

Shade calculation mode:
[1] "area" "prom"
```

```
Productivity without shadows:
Object of class  ProdGCPV

Source of meteorological information: prom-

Latitude of source:  37.2 degrees
Latitude for calculations:  37.2 degrees

Monthly avarages:
        Dates      Eac      Edc       Yf
       <char>    <num>    <num>    <num>
 1: Jan. 2024 138.6806 153.2566 5.241314
 2: Feb. 2024 143.4987 158.5247 5.423408
 3: Mar. 2024 151.8477 167.7311 5.738952
 4: Apr. 2024 178.6717 197.4274 6.752741
 5: May. 2024 200.8888 222.0523 7.592419
 6: Jun. 2024 223.9959 247.6903 8.465728
 7: Jul. 2024 214.2749 236.9628 8.098332
 8: Aug. 2024 194.6043 215.1439 7.354902
 9: Sep. 2024 168.9824 186.7349 6.386542
10: Oct. 2024 132.2995 146.0747 5.000145
11: Nov. 2024 128.5783 141.9871 4.859507
12: Dec. 2024 102.9116 113.5613 3.889454

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2024 60369.04 66710.67 2281.595
-----------------
Mode of tracking:  two
    Inclination limit: 90
-----------------
Generator:
    Modules in series:  12
    Modules in parallel:  11
    Nominal power (kWp):  26.5

Summary of results:
      Lew             Lns           H           FS                GRR              Yf
 Min.   :30.00   Min.   :20   Min.   :0   Min.   :0.01509   Min.   :2.649   Min.   :2104
 1st Qu.:33.75   1st Qu.:25   1st Qu.:0   1st Qu.:0.02223   1st Qu.:3.946   1st Qu.:2192
 Median :37.50   Median :30   Median :0   Median :0.02870   Median :4.802   Median :2216
 Mean   :37.50   Mean   :30   Mean   :0   Mean   :0.03463   Mean   :4.967   Mean   :2203
 3rd Qu.:41.25   3rd Qu.:35   3rd Qu.:0   3rd Qu.:0.03945   3rd Qu.:6.016   3rd Qu.:2231
 Max.   :45.00   Max.   :40   Max.   :0   Max.   :0.07769   Max.   :7.948   Max.   :2247
```

```
shadeplot(ShdM2x)
```


1−FS
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
L_{ns}
L_{ew}
0.97
0.96
0.95
7.5
7
6.5
6
5.5
5
4.5
4
3
0.93
30
32
34
36
38
40
42
44
20
25
30
35
40

CAPÍTULO 5

Ejemplo práctico de aplicación

Una vez explicado como funciona el paquete, se puede realizar una demostración práctica tomando como ejemplo los módulos fotovoltaicos que tiene en su azotea la Escuela Técnica Superior de Ingeniería y Diseño Industrial (en adelante la ETSIDI).

Se tomará de base un estudio realizado por profesores de la escuela [Adr+17], en el cual, comparan la producción energética de seis tipos de tecnologías fotovoltaicas.

En este ejemplo se realizará el mismo análisis tomando tres herramientas distintas: **solaR**, para poder tomar como referencia el paquete del que sale para poder apreciar las mejoras del programa, **PVSyst**, ya que es uno de los softwares más usados en el ámbito de la fotovoltaica y puede servir como punto de referencia, y por último **solaR2**.

5.1. solaR

Se empieza inicilizando el paquete:

```r
library(solaR)
```

```
Cargando paquete requerido: zoo

Adjuntando el paquete: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric

Cargando paquete requerido: lattice
Cargando paquete requerido: latticeExtra
Time Zone set to UTC.
```

En el estudio anterior, se recopilaron datos intradiarios de irradiación los cuales fueron almacenados en archivos.

```r
enemar13 <- readBDi(file = 'TFG/data/ETSIDI/etsidi/EneMar2013_1.csv',
                    lat = 40.4, time.col = 'Fecha')
show(enemar13)
```

```
Object of class  Meteo

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv
Latitude of source:  40.4 degrees

Meteorological Data:
     Index                              G0              Ta
 Min.   :2013-01-24 00:15:00.00   Min.   :  0.0   Min.   :-33.80
 1st Qu.:2013-02-09 18:00:00.00   1st Qu.: 13.0   1st Qu.:  9.50
 Median :2013-02-26 11:15:00.00   Median : 13.0   Median : 12.20
 Mean   :2013-02-26 11:19:20.84   Mean   :113.2   Mean   : 11.97
 3rd Qu.:2013-03-15 04:30:00.00   3rd Qu.:135.0   3rd Qu.: 14.40
 Max.   :2013-03-31 23:45:00.00   Max.   :755.0   Max.   : 64.50
```

Una vez se tienen estos datos, se puede calcular la producción que van a tener los diferentes sistemas fotovoltaicos.

Para ello, se necesitan los parámetros de los diferentes sistemas. En la tabla 5.1 se pueden ver los distintos parámetros de los módulos fotovoltaicos.

Se almacena esta información en listas con la información de cada módulo.

```r
## mc-Si
module1 <- list(Vocn = 37.1,
                Iscn = 8.76,
                Vmn = 29.9,
                Imn = 8.37,
                Ncs = 60,
                Ncp = 1,
                CoefVT = 0.00338,
                TONC = 43.7)
## pc-Si
module2 <- list(Vocn = 36.5,
                Iscn = 8.15,
                Vmn = 29,
                Imn = 7.59,
                Ncs = 60,
```

Tabla 5.1: *Parámetros técnicos de diferentes tipos de células solares.*

Parámetros Técnicos	**mc-Si**	**pc-Si**
Potencia se salida (Wp)	250	220
Voltaje en P_{max} (Vmp)	29.9	29.0
Corriente en P_{max} (Imp)	8.37	7.59
Voltaje en circuito abierto (Voc)	37.1	36.5
Corriente en cortocircuito (Isc)	8.76	8.15
Eficiencia del módulo (%)	15.5	14.4
α_{Isc} (%/K)	0.0043	0.06
β_{Voc} (%/K)	-0.338	-0.37
γ_{Pmpp} (%/K)	-0.469	-0.45
Temperatura NOC (ºC)	43.7	46

```
                 Ncp = 1,
                 CoefVT = 0.0037,
                 TONC = 46)
```

Una vez se tiene la información de cada tipo de módulo, en la tabla 5.2 se pueden ver la información de la agrupación de cada sistema.

De la misma manera, se almacenará esta información en listas.

```
## mc-Si
generator1 <- list(Nms = 5, Nmp = 1)
## pc-Si
generator2 <- list(Nms = 5, Nmp = 1)
```

Una vez se tienen todos los parémtros del sistema fotovoltaico, se requieren los parámetros del inversor que tienen estos sistemas. Para facilitar el estudio, en el artículo explican que se usa el mismo inversor para todos los sistemas. Los parámetros de este se pueden ver en la tabla 5.3.

Se almacena esta información en otra lista:

```
inverter <- list(Pinv = 1200,
                 Vmin = 100,
```

TABLA 5.2: *Sistemas fotovoltaicos.*

Sistema	**Tecnología**	**Año de Fabricación**	**Módulos en Serie**	**Módulos en Paralelo**	**Potencia del Sistema STC (Wp_{STC})**	**Tamaño (m^2)**
1	mc-Si	2012	5	1	1250	8
2	pc-Si	2009	5	1	1100	8.2

TABLA 5.3: *Carácteristicas del inversor.*

Inversor	**SMA Sunny Boy-1200**
Potencia máxima DC	1320 W
Corriente máxima DC	12.6 A
Tensión máxima DC	400 V
Rango de tensión fotovoltaica (mpp)	100-320 V
Potencia máxima DC	1320 W
Potencia nominal de salida	1200 W
Maxima potencia aparente	1200 VA
Corriente máxima AC	6.1 A
Eficiencia	92.1 %

```
                    Vmax = 320)
```

Una vez recopilada toda la información (la información que falta se deja sin añadir para que el propio paquete añada sus valores por defecto), se puede calcular la producción que tuvieron los sistemas:

```r
prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = enemar13,
                  beta = 30, alfa = -19, iS = 1,
                  module = module1, generator = generator1,
                  inverter = inverter)
show(prod1)
```

```
Object of class  ProdGCPV

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv

Latitude of source:  40.4 degrees
Latitude for calculations:  40.4 degrees

Monthly averages:
               Eac      Edc       Yf
ene. 2013 2.288657 2.544214 1.829001
feb. 2013 2.912867 3.246235 2.327844
mar. 2013 2.642931 2.958194 2.112123

Yearly values:
          Eac      Edc       Yf
2013 181.8004 202.9523 145.2875
-----------------
Mode of tracking:  fixed
    Inclination:  30
    Orientation:  -19
-----------------
Generator:
    Modules in series:  5
    Modules in parallel:  1
    Nominal power (kWp):  1.3
```

```r
prod2 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = enemar13,
                  beta = 30, alfa = -19, iS = 1,
                  module = module2, generator = generator2,
                  inverter = inverter)
show(prod2)
```

```
Object of class  ProdGCPV

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv

Latitude of source:  40.4 degrees
Latitude for calculations:  40.4 degrees

Monthly averages:
               Eac      Edc       Yf
ene. 2013 1.995563 2.219924 1.813242
feb. 2013 2.546910 2.840829 2.314216
mar. 2013 2.324995 2.608686 2.112576

Yearly values:
```

```
          Eac       Edc        Yf
2013 159.3528 178.1718 144.7938
-----------------
Mode of tracking:  fixed
    Inclination:  30
    Orientation:  -19
-----------------
Generator:
    Modules in series:  5
    Modules in parallel:  1
    Nominal power (kWp):  1.1
```

5.2. PVsyst

Con la herramienta **PVsyst**, se ha generado un año promedio de datos de irradiación en la localización y con estos datos se han obtenido dos informes (uno por cada sistema).

Por comodidad, en este documento se van a extraer solo unas tablas con los resultados principales, sin embargo los informes completos están disponibles en el github del documento.

En las tablas 5.4 y 5.5 se tienen los resultados de la simulación de los sistemas.

5.3. solaR2

Con los datos obtenidos en la sección 5.1, hacemos la misma operación pero con el paquete **solaR2**.

```r
library(solaR2)
```

```
Cargando paquete requerido: data.table
data.table 1.15.4 using 6 threads (see ?getDTthreads).  Latest news: r-datatable.com
Cargando paquete requerido: lattice
Cargando paquete requerido: latticeExtra
Time Zone set to UTC.
```

TABLA 5.4: *Energía media mensual estimada por* **PVSyst** *en KWh del sistema 1.*

Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic	Total
3,7	4,0	5,6	5,3	6,7	6,7	7,9	7,2	6,4	4,8	3,5	3,6	1941,1

TABLA 5.5: *Energía media mensual estimada por* **PVSyst** *en KWh del sistema 2.*

Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic	Total
4,3	4,6	6,4	6,1	7,3	7,3	8,3	7,7	6,9	5,4	4,1	4,4	2213,7

Para ello importamos de la misma manera los datos de radiación.

```
enemar13 <- readBDi(file = 'TFG/data/ETSIDI/etsidi/EneMar2013_1.csv',
                    lat = 40.4, dates.col = 'Fecha')
show(enemar13)
```

```
Object of class  Meteo

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv
Latitude of source:  40.4 degrees

Meteorological Data:
     Dates                              G0             Ta
 Min.   :2013-01-24 00:15:00.00   Min.   :  0.0   Min.   :-33.80
 1st Qu.:2013-02-09 18:00:00.00   1st Qu.: 13.0   1st Qu.:  9.50
 Median :2013-02-26 11:15:00.00   Median : 13.0   Median : 12.20
 Mean   :2013-02-26 11:19:20.84   Mean   :113.2   Mean   : 11.97
 3rd Qu.:2013-03-15 04:30:00.00   3rd Qu.:135.0   3rd Qu.: 14.40
 Max.   :2013-03-31 23:45:00.00   Max.   :755.0   Max.   : 64.50
```

Con estos datos se procede al cálculo de la producción (los datos de los componentes del sistema son los mismos que los realizados en la sección 5.1).

```
prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = enemar13,
                  beta = 30, alfa = -19, iS = 1,
                  module = module1, generator = generator1,
                  inverter = inverter)
show(prod1)
```

```
Object of class  ProdGCPV

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv

Latitude of source:  40.4 degrees
Latitude for calculations:  40.4 degrees

Monthly avarages:
       Dates      Eac      Edc       Yf
      <char>    <num>    <num>    <num>
1: Jan. 2013 2.288657 2.544214 1.829001
2: Feb. 2013 2.912867 3.246235 2.327844
3: Mar. 2013 2.642931 2.958194 2.112123

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2013 181.8004 202.9523 145.2875
-----------------
Mode of tracking:  fixed
    Inclination:  30
    Orientation:  -19
-----------------
Generator:
    Modules in series:  5
    Modules in parallel:  1
    Nominal power (kWp):  1.3
```

```
prod2 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = enemar13,
```

```
                        beta = 30, alfa = -19, iS = 1,
                        module = module2, generator = generator2,
                        inverter = inverter)
show(prod2)
```

```
Object of class  ProdGCPV

Source of meteorological information: bdI-TFG/data/ETSIDI/etsidi/EneMar2013_1.csv

Latitude of source:  40.4 degrees
Latitude for calculations:  40.4 degrees

Monthly avarages:
       Dates      Eac      Edc       Yf
      <char>    <num>    <num>    <num>
1: Jan. 2013 1.995563 2.219924 1.813242
2: Feb. 2013 2.546910 2.840829 2.314216
3: Mar. 2013 2.324995 2.608686 2.112576

Yearly values:
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2013 159.3528 178.1718 144.7938
-----------------
Mode of tracking:  fixed
    Inclination:  30
    Orientation:  -19
-----------------
Generator:
    Modules in series:  5
    Modules in parallel:  1
    Nominal power (kWp):  1.1
```

5.4. Comparación y conclusiones

Como se puede observar en las secciones anteriores, tanto el paquete **solaR** como el paquete **solaR2** ofrecen los mismos resultados ya que toman las mismas referencias y estudios para realizar los cáculos. Sin embargo, el paquete **solaR2**, a parte de la corrección de algunos erores, presenta unas claras ventajas frente a su antecesor. Estas son:

- **Eficiencia**: al estar basado en **data.table**, el paquete gana eficiencia en operaciones complejas. Para mostrar esto vamos a utilizar el paquete **microbenchmark**.

```
## Con el paquete solaR
library(microbenchmark)
prodGCPVcustom <- function(){
  prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                    dataRad = enemar13, beta = 30, alfa =-19,
                    iS = 1, module = module1,
                    generator = generator1, inverter = inverter)
}
microbenchmark(prodGCPVcustom(), times = 50)
```

```
Unit: milliseconds
             expr      min       lq     mean   median       uq      max neval
 prodGCPVcustom() 529.5188 543.3399 554.9077 548.5941 558.3901 626.5719    50
```

```r
## Con el paquete solaR2
library(microbenchmark)
prodGCPVcustom <- function(){
  prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                    dataRad = enemar13, beta = 30, alfa =-19,
                    iS = 1, module = module1,
                    generator = generator1, inverter = inverter)
}
microbenchmark(prodGCPVcustom(), times = 50)
```

```
Unit: milliseconds
             expr      min       lq     mean   median       uq      max neval
 prodGCPVcustom() 333.9282 338.6178 345.8622 341.9009 345.6647 402.8548    50
```

Aquí se puede ver que la eficiencia mejora. Sin embargo, suponiendo que en vez de un sistema fijo, tuvieramos un sistema de seguimiento de doble eje y quisieramos optener la mejor combinación de distancias, se podría utilizar la función **`optimShd`**, la cual al ser una tarea muy exigente se aprecia con más detalle las virtudes del paquete **`solaR2`** gracias al uso de **`data.table`**.

```r
## Con el paquete solaR
struct2x <- list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 3)
dist2x <- list(Lew = c(30, 45),Lns = c(20, 40))
optimShdcustom <- function(){
  optim <- optimShd(lat = 40.4, modeTrk = 'two', modeRad = 'bdI',
                    dataRad = enemar13, beta = 30, alfa =-19,
                    iS = 1, module = module1,
                    generator = generator1, inverter = inverter,
                    modeShd = c('area', 'prom'),
                    distances = dist2x, struct = struct2x,
                    res = 5, prog = FALSE)
}
microbenchmark(optimShdcustom(), times = 20)
```

```
Unit: seconds
             expr     min       lq     mean   median      uq      max neval
 optimShdcustom() 6.30659 6.387046 6.447254 6.453184 6.48896 6.610802    20
```

```r
## Con el paquete solaR2
struct2x <- list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 3)
dist2x <- list(Lew = c(30, 45),Lns = c(20, 40))
optimShdcustom <- function(){
  optim <- optimShd(lat = 40.4, modeTrk = 'two', modeRad = 'bdI',
                    dataRad = enemar13, beta = 30, alfa =-19,
                    iS = 1, module = module1,
                    generator = generator1, inverter = inverter,
                    modeShd = c('area', 'prom'),
                    distances = dist2x, struct = struct2x,
                    res = 5, prog = FALSE)
}
microbenchmark(optimShdcustom(), times = 20)
```

```
Unit: seconds
            expr      min       lq     mean   median       uq      max neval
 optimShdcustom() 5.121113 5.154294 5.175936 5.171816 5.182229 5.249169    20
```

APÉNDICE A

Manual de referencia de `solaR2`

En este apéndice se incluye el manual de referencia del paquete solaR2. Este manual se genera en base a los archivos de documentación (`.Rd`) propios de un paquete de R, y en el cual se recoge la información de todas las funciones, objetos y set de datos que contiene el paquete.

Se distribuye siguiendo la siguiente nomenclatura:

- **Constructores**: se trata de funciones que devuelven un objeto de una clase propia del paquete. Como identificador, se añade la letra **A** antes del nombre.
- **Clases**: la definición de las clases de los objetos definidos por este paquete. Como identificador, se añade la letra **B** antes del nombre.
- **Utilidades**: funciones que sirven de apoyo a los cálculos de las funciones constructoras. Como identificador, se añade la letra **C** antes del nombre.
- **Métodos**: métodos para los objetos definidos en el paquete. Como identificador, se añade la letra **D** antes del nombre.

Package ‘solaR2’

September 5, 2024

Type Package

Title Radiation and Photovoltaic Systems

Version 0.10

Encoding UTF-8

Description Calculation methods of solar radiation and performance of photovoltaic systems from daily and intradaily irradiation data sources.

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

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

License GPL-3

LazyData yes

Depends R (>= 4.0.0), data.table, lattice, latticeExtra

Imports RColorBrewer, graphics, grDevices, stats, methods, utils

Suggests zoo, sp, raster, rasterVis, tdr, meteoForecast, httr2, jsonlite, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Oscar Perpiñán-Lamigueiro [cre, aut] (<https://orcid.org/0000-0002-4134-7196>), Francisco Delgado-López [aut]

Maintainer Oscar Perpiñán-Lamigueiro <oscar.perpinan@upm.es>

Contents

solaR2-package	*Solar Radiation and Photovoltaic Systems with R version 2*

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 and 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 `fBTd`. 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 `seq.POSIXt`. It is not considered if `BTi` is provided.
`BTi`	Intradaily time base, a `POSIXct` object to be used by `fSolI`. It may be the result of `fBTi`.
`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.

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

Examples

```
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	A character string, describes the kind of source data of the global irradiation and ambient temperature. 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. 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 etal. (see reference below). If modeRad = 'bd' the information of *daily* 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). If modeRad = 'bdI' the information of *intradaily* 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).
dataRad	• 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 data.table (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 data.table (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	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' while the FdKtCPR with corr = 'CPR'. If corr = 'user' the use of a correlation defined by a function f is possible. 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'.
f	A function defininig a correlation between the fraction of diffuse irradiation and the clearness index. It is only neccessary when corr = 'user'
...	Additional arguments for fCompD or fCompI

Value

A G0 object.

Author(s)

Oscar Perpiñán Lamigueiro.

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
- 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

```
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 *effective* 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.

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

See Also

calcG0, fTheta, fInclin, calcShd.

Examples

```
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/m²) (default value 20 W/m²)

`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

`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

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

`fProd`, `calcGef`, `calcShd`, `calcG0`, `compare`, `compareLosses`, `mergesolaR`

Examples

```
library(lattice)
library(latticeExtra)

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.

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

See Also

`NmgPVPS`, `fPump`, `pumpCoef`

`A6_calcShd` *Shadows on PV systems.*

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.

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

`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)

`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/m²) (default value 20 W/m²)

`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

`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` 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.

`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 setof trackers in the PV plant.

`distances` `list`, whose three components are vectors of length 2:

`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.

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

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

See Also

prodGCPV, calcShd

Examples

```
library(lattice)
library(latticeExtra)

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
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)
```

A8_Meteo2Meteo	*Transformation of intradaily meteorological data into daily and daily into monthly data.*

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

See Also

readBDd, readG0dm, readSIAR

Examples

```
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 *Daily or intradaily values of global horizontal irradiation and ambient temperature from a local file or a data.frame.*

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` — The name of the file (`readBDd` and `readBDi`), `data.frame` (or `data.table`) (`dt2Meteo`) or `zoo` (`zoo2Meteo`) which the data are to be read from. It should contain a column `G0d` with *daily* (`readBDd`) or `G0` with *intradaily* (`readBDi`) values of global horizontal irradiation (Wh/m²). It should also include a column named `Ta` with values of ambient temperature. However, if the object is only a vector with irradiation values, it will converted to a `data.table` with two columns named `G0` and `Ta` (filled with constant values)

  If the `Meteo` object is to be used with `calcG0` (or `fCompD`, `fCompI`) and the option `corr = 'none'`, the file/data.frame **must** include three columns named `G0`, `B0` and `D0` with values of global, direct and diffuse irradiation on the horizontal plane.

  Only for daily data: if the ambient temperature is not available, the file should include two columns named `TempMax` and `TempMin` with daily values of maximum and minimum ambient temperature, respectively (see `fTemp` for details).
- `header, fill, dec, sep` — See `fread`
- `format` — character string with the format of the dates or time index. (Default for daily time bases:`%d/%m/%Y`). (Default for intradaily time bases: `%d/%m/%Y %H:%M:%S`)
- `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.

See Also

fread, readG0dm.

Examples

```
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/m²).
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.

See Also

readBDd

Examples

```
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 fromm 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, tipe 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

See Also

`readG0dm`, `readBDd`

Examples

```
library(httr2)
library(jsonlite)

SIAR = readSIAR(Lon = -3.603, Lat = 40.033,
## Aranjuez, Comunity 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 (`G0`, Wh/m²), the ambient temperature (`Ta`) 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.

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.

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.

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, 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, FTr:** 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 degreesto 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.

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 *Class "ProdGCPV": performance of a grid connected PV system.*

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.

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`, `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, (m^3/h)
- **Pb, Ph:** Pump shaft power and hydraulical 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 (m^3) and productivity of the system.

`prodDm`: A `data.table` object with monthly means of daily values of AC energy (kWh), flow (m^3) and productivity of the system.

`prody`: A `data.table` object with yearly sums of AC energy (kWh), flow (m^3) 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.

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

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.

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

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 intradaily 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; 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

```
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

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

fSolD, as.POSIXct, seq.POSIXt.

Examples

```
#Average days
fBTd(mode = 'prom')

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

`C_fBTi` *Intra-daily time base*

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 `fSolI`.

Value

This function returns a `POSIXct` object.

Author(s)

Oscar Perpiñán Lamigueiro

Examples

```
#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 Bo0d, 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/m²) 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 defininig a correlation between the fraction of diffuse irradiation and the clearness index. It is only neccessary 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/m²)
D0d	numeric, the diffuse irradiation on a horizontal surface (Wh/m²)
B0d	numeric, the direct irradiation on a horizontal surface (Wh/m²)

Author(s)

Oscar Perpiñán Lamigueiro

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

```
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 calcSol or a data.table object as provided by fSolI.
compD	A data.table object as provided by fCompD. It is not considered if G0I is provided.

G0I	A Meteo object from readBDi, dt2Meteo or zoo2Meteo, or a data.table object containing *intradaily* global irradiance (W/m²) on a horizontal surface. See below for corr = 'none'.
corr	A character, the correlation between the the fraction of intradaily diffuse irradiation and the clearness index to be used. It is ignored if G0I is not provided. With this version several correlations are available, as described in corrFdKt. You should choose one of *intradaily* proposals. For example, the FdKtCLIMEDh is selected with corr = 'CLIMEDh'. If corr = 'user' the use of a correlation defined by a function f is possible. If corr = 'none' the G0I object must include information about global, diffuse and direct intradaily irradiation with columns named G0, D0 and B0, respectively.
f	A function defininig a correlation between the fraction of diffuse irradiation and the clearness index. It is only neccessary when corr = 'user'
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/m²)
D0	numeric, diffuse irradiance on a horizontal surface, (W/m²)
B0	numeric, direct irradiance on a horizontal surface, (W/m²)

Author(s)

Oscar Perpiñán Lamigueiro.

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

```
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)
```

`C_fInclin` *Solar irradiance on an inclined surface*

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

`compI`	A `G0` object. It may be the result of `calcG0`.
`angGen`	A `data.table` object, including at least three variables named `Beta`, `Alpha` and `cosTheta`. It may be the result of `fTheta`.
`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 choice for a dirty surface. Its default value is 2
`alb`	numeric, albedo reflection coefficient. Its default value is 0.2

`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 *effective* 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:

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

Author(s)

Oscar Perpiñán Lamigueiro.

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

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 `data.table` object. In case of being `data.table` it must include a component named `Gef` (effective irradiance, W/m²) and another named `Ta` (ambient temperature, °C).

`module` list of numeric values with information about the PV module,

- `Vocn` open-circuit voltage of the module at Standard Test Conditions (default value 51.91 volts.)
- `Iscn` short circuit current of the module at Standard Test Conditions (default value 14.07 amperes.)
- `Vmn` maximum power point voltage of the module at Standard Test Conditions (default value 43.76 volts.)
- `Imn` Maximum power current of the module at Standard Test Conditions (default value 13.03 amperes.)
- `Ncs` number of cells in series inside the module (default value 24)
- `Ncp` number of cells in parallel inside the module (default value 6)
- `CoefVT` coefficient of decrement of voltage of each cell with the temperature (default value 0.1194 volts per celsius degree)
- `TONC` nominal operational cell temperature, celsius degree (default value 20).

`generator` list of numeric values with information about the generator,

- `Nms` number of modules in series (default value 22)
- `Nmp` number of modules in parallel (default value 130)

`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 1500 watts.)
- `Vmin, Vmax` minimum and maximum voltages of the MPP range of the inverter (default values 822 and 1300 volts)
- `Gumb` minimum irradiance for the inverter to start (W/m²) (default value 200 W/m²)

`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

`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

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 `Inclin`

`Vdc, Idc` voltage and current at the input of the inverter. If no voltage limitation occurs (according to the values of `inverter$Vmax` and `inverter$Vmin`), their values are identical to `Vmpp` and `Impp`. 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

References

- Jantsch, 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

See Also

`fInclin`, `prodGCPV`, `fTemp`.

Examples

```
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 * Impp))
```

`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 `splinefun` relating flow and electrical power
`fPb`	Function constructed with `splinefun` relating pump shaft power and electrical power of the motor
`fPh`	Function constructed with `splinefun` relating hydraulical power and electrical power of the motor
`fFreq`	Function constructed with `splinefun` relating frequency and electrical power of the motor

Author(s)

Oscar Perpiñán Lamigueiro.

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

See Also

NmgPVPS, prodPVPS, pumpCoef, splinefun.

Examples

```
library(latticeExtra)

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
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')
```

C_fSolD *Daily apparent movement of the Sun from the Earth*

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 `fBTd`.
`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.

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

Examples

```
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())

library(lattice)
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 intradaily 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.

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

See Also

`fSolD`

Examples

```
###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)

library(lattice)
library(latticeExtra)

###Solar elevation angle vs. azimuth.
#This kind of graphics is useful for shadows calculations
mon = month.abb
p <- xyplot(r2d(AlS)~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)
```

C_fSombra *Shadows on PV systems*

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.

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

See Also

calcShd, optimShd, fTheta, calcSol

Examples

```
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 its calculated through a combination of cosine functions (ESRA method)

Usage

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

Arguments

sol	A Sol object. It may be the result of the calcSol function.
BD	A Meteo object, as provided by the readBDd 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.

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

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 back-tracking 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

`sol`	`Sol` object as provided by `calcSol`.
`beta`	numeric, inclination angle of the surface (degrees). It is only needed when `modeTrk = 'fixed'`.
`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` (surface facing to the south).
`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'`
`betaLim`	numeric, maximum value of the inclination angle for a tracking surface. Its default value is 90 (no limitation))
`BT`	logical, `TRUE` when the bactracking 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.
`struct`	Only needed when `BT = TRUE`. A `list`, with a component named `L`, which is the height (meters) of the tracker. In future versions the backtracking technique will be used in conjuction with two-axis trackers, and a additional component named `W` 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.

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

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.

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

See Also

NmgPVPS, prodPVPS, pumpCoef.

Examples

```
library(lattice)
library(latticeExtra)

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.

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

Examples

```
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 irradiation incident on the generator (Wh/m²)
`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.

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

See Also

`fPump`, `prodPVPS`, `pumpCoef`

Examples

```
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())
```

C_sample2Diff *Small utilities for difftime objects.*

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

See Also

Sol

Examples

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

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

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

C_solarAngles *Solar angles*

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 fBTd

`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`, intradily time base, it may the result of `fBTi`.

`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.

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

See Also

`fSolD`, `fSolI`, `calcSol`

Examples

```
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.

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

`hour(x)+minute(x)/60+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.

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

Examples

```
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

Examples

```
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

Examples

```
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

Examples

```
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

See Also

`dotplot`

Examples

```
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

`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

`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

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

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

`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

`D_Losses-methods` *Losses of a GCPV system*

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

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

fInclin, fProd

Examples

```
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

Examples

```
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

`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

See Also

indexD

Examples

```
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))
```

D_writeSolar-methods *Exporter of solaR results*

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. It `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

See Also

`write.table`, `fread`, `as.data.tableI`, `as.data.tableD`, `as.data.tableM`, `as.data.tableY`

Examples

```
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)
```

D_xyplot-methods *Methods for function xyplot in Package 'solaR'*

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

`E_aguiar` *Markov Transition Matrices for the Aguiar etal. 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.

`E_helios` *Daily irradiation and ambient temperature from the Helios-IES database*

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 *Productivity of a set of PV systems of a PV plant.*

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): $\eta_m = g \cdot (\%P_{mn})^2 + h \cdot (\%Pmn) + i$.

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

`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 decommisioned, 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, 'YlOrRd'))`
- `strip.background$col = 'lightgray'`
- `strip.shingle$col = 'transparent'`

Index

Bibliografía

[LJ60] B. Y. H. Liu y R. C. Jordan. "The interrelationship and characteristic distribution of direct, diffuse, and total solar radiation". En: *Solar Energy* 4 (1960), págs. 1-19.

[Pag61] J. K. Page. "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. 98. 1961, págs. 378-390.

[Coo69] P.I. Cooper. "The Absorption of Solar Radiation in Solar Stills". En: *Solar Energy* 12 (1969).

[Spe71] J.W. Spencer. "Fourier Series Representation of the Position of the Sun". En: 2 (1971). URL: http://www.mail-archive.com/sundial@uni-koeln.de/msg01050.html.

[CR79] M. Collares-Pereira y Ari Rabl. "The average distribution of solar radiation: correlations between diffuse and hemispherical and between daily and hourly insolation values". En: *Solar Energy* 22 (1979), págs. 155-164.

[Sta85] Richard Stallman. *GNU Emacs*. Un editor de texto extensible, personalizable, autodocumentado y en tiempo real. 1985. URL: https://www.gnu.org/software/emacs/.

[Mic88] Joseph J. Michalsky. "The Astronomical Almanac's algorithm for approximate solar position (1950-2050)". En: *Solar Energy* 40.3 (1988), págs. 227-235. ISSN: 0038-092X. DOI: DOI:10.1016/0038-092X(88)90045-X.

[RBD90] D.T. Reindl, W.A. Beckman y J.A. Duffie. "Evaluation of hourly tilted surface radiation models". En: *Solar Energy* 45.1 (1990), págs. 9-17. ISSN: 0038-092X. DOI: https://doi.org/10.1016/0038-092X(90)90061-G. URL: https://www.sciencedirect.com/science/article/pii/0038092X9090061G.

[Pan+91] D. Panico et al. "Backtracking: a novel strategy for tracking PV systems". En: *IEEE Photovoltaic Specialists Conference*. 1991, págs. 668-673.

[Dom+03] Carsten Dominik et al. *Org Mode*. Un sistema de organización de notas, planificación de proyectos y autoría de documentos con una interfaz de texto plano. 2003. URL: https://orgmode.org.

[ZG05] Achim Zeileis y Gabor Grothendieck. "zoo: S3 Infrastructure for Regular and Irregular Time Series". En: *Journal of Statistical Software* 14.6 (2005), págs. 1-27. DOI: 10.18637/jss.v014.i06.

[Sar08] Deepayan Sarkar. *Lattice: Multivariate Data Visualization with R*. New York: Springer, 2008. ISBN: 978-0-387-75968-5. URL: http://lmdvr.r-forge.r-project.org.

[Str11] L. Strous. *Position of the Sun*. 2011. URL: http://aa.quae.nl/en/reken/zonpositie.html.

[Per12] Oscar Perpiñán. "solaR: Solar Radiation and Photovoltaic Systems with R". En: *Journal of Statistical Software* 50.9 (2012), págs. 1-32. DOI: 10.18637/jss.v050.i09.

[Adr+17] T. Adrada Guerra et al. "Comparative Energy Performance Analysis of Six Primary Photovoltaic Technologies in Madrid (Spain)". En: *Energies* 10.6 (2017), pág. 772. DOI: 10.3390/en10060772. URL: https://doi.org/10.3390/en10060772.

[Uni20] European Union. *NextGenerationEU*. 2020. URL: https://next-generation-eu.europa.eu/index_es.

[BOE22a] BOE. *Real Decreto-ley 10/2022, de 13 de mayo, por el que se establece con carácter temporal un mecanismo de ajuste de costes de producción para la reducción del precio de la electricidad en el mercado mayorista*. 2022. URL: https://www.boe.es/buscar/act.php?id=BOE-A-2022-7843.

[BOE22b] BOE. *Real Decreto-ley 6/2022, de 29 de marzo, por el que se adoptan medidas urgentes en el marco del Plan Nacional de respuesta a las consecuencias económicas y sociales de la guerra en Ucrania*. 2022. URL: https://www.boe.es/buscar/doc.php?id=BOE-A-2022-4972.

[dem22] Ministerio para transción ecológica y el reto demográfico. *Plan + Seguridad Energética*. 2022. URL: https://www.miteco.gob.es/es/ministerio/planes-estrategias/seguridad-energetica.html#planSE.

[Eur22] Consejo Europeo. *REPowerEU*. 2022. URL: https://www.consilium.europa.eu/es/policies/eu-recovery-plan/repowereu/.

[Hac22] Ministerio de Hacienda. *Mecanismo de Recuperación y Resiliencia*. 2022. URL: https://www.hacienda.gob.es/es-ES/CDI/Paginas/FondosEuropeos/Fondos-relacionados-COVID/MRR.aspx.

[Mer+23] Olaf Mersmann et al. *microbenchmark: Accurate Timing Functions*. Proporciona infraestructura para medir y comparar con precisión el tiempo de ejecución de las expresiones de R. 2023. URL: https://github.com/joshuaulrich/microbenchmark.

[Min23] pesca y alimentación Ministerio de agricultura. *Sistema de Información Agroclimática para el Regadío*. 2023. URL: https://servicio.mapa.gob.es/websiar/.

[Per23] O. Perpiñán. *Energía Solar Fotovoltaica*. 2023. URL: https://oscarperpinan.github.io/esf/.

[R C23] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2023. URL: https://www.R-project.org/.

[UNE23] UNEF. "Fomentando la biodiversidad y el crecimiento sostenible". En: *Informe anual UNEF* (2023). URL: https://www.unef.es/es/recursos-informes?idMultimediaCategoria=18.

[Wan+23] Chris Wanstrath et al. *GitHub*. 2023. URL: https://github.com/.

[SAM24] System Advisor Model (SAM). *SAM: System Advisor Model*. https://sam.nrel.gov/. 2024.

[Bar+24] Tyson Barrett et al. *data.table: Extension of 'data.frame'*. R package version 1.15.99, https://Rdatatable.gitlab.io/data.table, https://github.com/Rdatatable/data.table. 2024. URL: https://r-datatable.com.

[Nat24] National Renewable Energy Laboratory. *Best Research-Cell Efficiency Chart*. https://www.nrel.gov/pv/cell-efficiency.html. 2024.

[Pro24] ESS Project. *Emacs Speaks Statistics (ESS)*. Un paquete adicional para GNU Emacs diseñado para apoyar la edición de scripts y la interacción con varios programas de análisis estadístico. 2024. URL: https://ess.r-project.org/.

[PVG24] PVGIS. *PVGIS: Photovoltaic Geographical Information System*. https://ec.europa.eu/jrc/en/pvgis. 2024.

[PVS24] PVSyst. *PVSyst: Software for Photovoltaic Systems*. https://www.pvsyst.com/. 2024.

[Sis24] Sisifo. *Sisifo: Solar Energy Simulation Software*. https://www.sisifo.org/. 2024.

[Wic+24] H. Wickham et al. *profvis: Interactive Visualizations for Profiling R Code*. R package version 0.3.8.9000. 2024. URL: https://github.com/rstudio/profvis.