UNIVERSIDAD POLITÉCNICA DE MADRID

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, 7 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
AM	Masa de aire
AO	Adelanto oficial durante el horario de invierno
B	Radiación directa
$B_0(0)$	Irradiancia extra-atmosférica o extra-terrestre en el plano horizontal
$B_{0d}(0)$	Irradiación diaria extra-atmosférica en el plano horizontal
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_{min}	Distancia mínima entre hileras de un generador para evitar el sombreado
d_n	Día del año
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
ψ_s	Ángulo acimutal solar
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].

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 Estrategia 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. Soluciones actuales y sus carencias

Como se mencionó en el capítulo 1, existen una serie de herramientas que permiten el cálculo y la simulación de instalaciones fotovoltaicas. Todas ellas presentan una serie de ventajas específicas, a cambio de una serie de limitaciones. Estas son:

1. **PVsyst - Photovoltaic Software** [PVS24] Este software es probablemente el más conocido dentro del ámbito del estudio y la estimación de instalaciones fotovoltaicas. Destaca

[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.

por la personalización detallada de los componentes de la instalación (módulos, inversores, sombreado, etc.), lo que ermite una simulación precisa a través de datos meteorológicos y parámetros detallados del sistema. Su uso está extendido en proyectos de gran escala y estuidos avanzados de eficiencia.

- Ventajas:
  - **Completo y profesional**: PVsyst es altamente detallado, permitiendo análisis avanzados para proyectos pequeños y grandes.
  - **Base de datos meteorológicos**: Integra datos climáticos de fuentes como Meteonorm [JG20], lo que mejora la precisión de las simulaciones.
  - **Simulaciones avanzadas**: Permite modelar la energía producida por una planta fotovoltaica y calcular las pérdidas debidas a sombreamiento, inclinación, orientaciones y resistencias internas de los módulos.
  - **Herramientas de dimensionamiento**: Ofrece módulos específicos para diseñar la configuración de inversores y módulos solares.
- Limitaciones:
  - **Costo**: Es un software comercial , con licencias que pueden ser costosas para proyectos pequeños.
  - **Curva de aprendizaje**: Su interfaz puede resultar compleja para usuarios nuevos, lo que implica una curva de aprendizaje considerable.
  - **Enfoque técnico**: Está más orientado a ingenieros y técnicos, por lo que carece de accedsibilidad para usuarios no especializados.

2. **SISIFO** [Sis24] Herramienta web diseñada por el **Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar de la Universidad Politécnica de Madrid**. Está diseñada para ser accesible y fácil de usar, enfocándose en una audiencia más general, incluyenco ingenieros, pero también técnicos y académicos.

   - Ventajas:
     - **Facilidad de uso**: Tiene una interfaz amigable y fácil de utilizar, lo que lohace accesible para usuarios con distintos niveles de experiencia.
     - **Open-source**: Al ser de código abierto, permite a los desarrolladores modificar y adaptar el software a sus necesidades específicas.
     - **Simulación integrada**: Ofrece la posibilidad de realizar simulaciones basadas en datos meteorológicos, aunque con un nivel de detalle inferior a PVsyst.
     - **Soporte comunitario**: Al ser de código abierto, cuenta con una comunidad activa de usuarios y desarrrolladores que colaboran en mejoras y actualizaciones.
   - Limitaciones:
     - **Menos preciso**: Al compararse con otras herramientas, su precisión puede ser menor en cuanto a modelado y simulación de pérdidas, ya que simplifica varios aspectos del sistema.
     - **Limitaciones en grandes proyectos**: No está tan bien adaptado para grandes instalaciones o análisis financieros avanzados.

3. **PVGIS** [PVG24] Aplicación web desarrolada por el **European Commission Joint Research Center** desde 2001. Está diseñada para proporcionar estimaciones de producción de energía solar en función de la ubicación geográfica y condiciones meteorológicas históricas.

   - Ventajas:

- **Gratuito y accesible**: Esta herramienta es completamente gratuira y accesible a través de una interfaz web, lo que facilita el uso por parte de cualquier persona.
- **Datos meteorológicos precisos**: Proporciona acceso a datos meteorológicos satelitales y de estaciones terrestres, lo que permite obtener estimaciones razonables de producción de energía.
- **Estudios rápidos**: Es ideal para obtener estimaciones preliminares y estudios de viabilidad de sistemas fotovoltaicos.

4. **System Advisor Model** [SAM24] Desarrollado por el **Laboratorio Nacional de Energías Renovables**, perteneciente al Departamento de energía del gobierno de EE.UU. Está orientada a la modelación tanto técnica como económica de sistemas de energía renovable, incluyendo fotovoltaicos.

   - Ventajas:
     - **Modelo económico avanzado**: Integra análisis detalados sobre la viabilidad económica, lo que permite evaluar tanto la producción energética como los costos y beneficios a lo largo de la vida útil del proyecto.
     - **Acceso a múltiples tecnologías**: Además de fotovoltaicos, permite modelar otras tecnologías de energía renovable, lo que lo hace más flexible para estuidos multidisciplinares.
     - **Integración de bases de datos**: Utiliza datos meteorológicos detallados, lo que mejora la precisión de las simulaciones.
   - Limitaciones:
     - **Complejidad**: Aunque gratuito, SAM es bastante complejo y técnico, lo que puede limitar su uso a usuarios con experiencia en el modelado de sistemas energéticos.
     - **Interfaz no tan intuitiva**: Comparado con otras herramientas, requiere un mayor tiempo de familiarización debido a su enfoque integral y detalle en las simulaciones.

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`. Su funcionamiento se basa, al igual que `solaR2`, en una serie de funciones constructoras que calculan objetos relacionados con cada paso de la simulación de un sistema fotovoltaico. Podemos dividir su funcionamiento en los siguientes grupos:

1. **Cálculo de la geometría solar**: calcula el movimiento aparente diario (con `fSolD`) e intradiario (con `fSolI`) del Sol desde la Tierra. Para ello se vale de la función `calcSol` la cual devuelve un objeto de clase `Sol` que contiene todos los ángulos necesarios.

2. **Almacenamiento de datos meteorológicos**: se define la clase `Meteo`, la cual, se construye mediante una serie de funciones (`readBD`, `readG0dm`, `zoo2Meteo`, `df2Meteo`...). Estas funciones toman los datos meteorológicos provenientes de distintas vias (un `data.frame`, un objeto `zoo`, un fichero...) y los adapta para que puedan ser manipulados por el resto de funciones del paquete.

3. **Cálculo de radiación en un plano horizontal**: tomando los objetos anteriores, es capaz de calcular (si no vienen ya dadas) las componentes de la irradiación (con `fCompD`) y de la irradiancia (con `fCompI`). La función `calcG0` devuelve un objeto `G0` que contiene las anteriores componentes y añade medias mensuales de valores diarios y sumas anuales de la irradiación.

4. **Cálculo de radiación en el plano del generador**: toma un objeto `G0` y lo transforma en un objeto `Gef` mediante la función `calcGef`, la cual utilizando las funciones `fTheta` y `fInclin` determinan la irradiación y la radiación efectiva al igual que las medias mensuales de la irradiación diaria y sumas anuales.

5. **Simulación de sistemas fotovoltaicos conectados a red**: con un objeto `Gef` y con los parametros del sistema, la función `prodGCPV`, tomando los resultado de la función `fProd`, cálcula la producción energética de un SFCR. Devuleve un objeto de clase `ProdGCPV` que incluye valores de potencias instantaneas y energías diarias, medias mensuales y sumas anuales.

6. **Simulación de sistemas fotovoltaicos de bombeo**: toma un objeto `Gef` y con los paremetros del sistema y de la bomba, la función `prodPVPS`, tomando los resultados de la función `fPump`, cálcula la producción energética de un SFB-

7. **Optimización de distancias**: es capaz de optimizar las distancias de un sfcr mediante la función `optimShd`, la cual devuelve un objeto `Shade` el cual contiene multiples combinaciones de distancias para que el usuario pueda decidir la mejor.

8. **Métodos de visualización**: para cada uno de los objetos mencionados existen métodos de visualización gráfica para ayudar a comprender los resultados obtenidos.

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

- **Modularidad**: el paquete `solaR` contiene funciones que realizan muchas operaciones, esto deja poco lugar al usuario para que pueda entender cada componente independientemente.
- **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.
- **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.
- **Manipulación de datos**: `zoo` es adecuado para manejar series temporales básicas, pero carece de las capacidades avanzadas de manipulación de datos que ofrecen otros paquetes.

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

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. Radiación solar

3.1.1. Geometría Sol y Tierra

Como es sabido, el movimiento terrestre se compone de una tralación alrededor del Sol y un giro sobre su eje. En el movimiento de traslación la Tierra se desplaza alrededor del Sol siguiendo una elipse de baja excentricidad en la que el Sol ocupa uno de los focos. La duración de este movimiento define un año. La corrección debida a la excentricidad de la elipse[1] se calcula con:

$$\epsilon_0 = 1 + 0{,}033 \cdot \cos(\frac{2\pi d_n}{365}) \tag{3.1}$$

[1] Correspondiente a la función **eccentricity**.

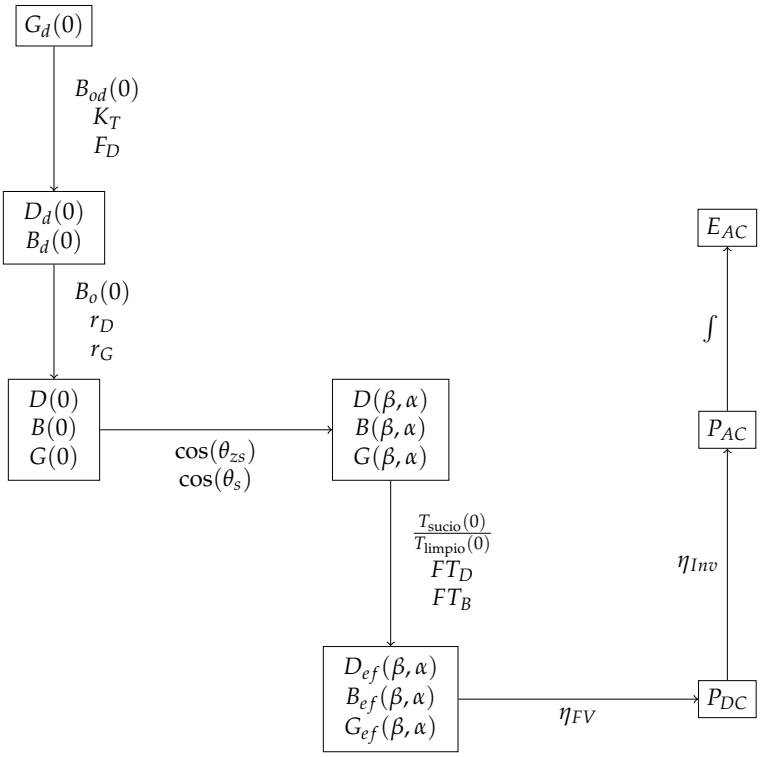


Figura 3.1: *Procedimiento de cálculo. Tomando la irradiación global en el plano horizontal, se pueden calcular las componentes directa y difusa mediante índices de claridad y fracciones de difusa. Con todas las componentes de irradiación, se pueden estimar los valores de irradiancia. A continuación, se trasladan estos valores al plano inclinado para después poder aplicar los facotres por suciedad y sombras y obtener así los valores de irradiancia efectiva. Por ultimo, con la eficiencia del sistema fotovoltaico se puede obtener la potencia en corriente continua, la cual, al pasar por un inversor, se convierte en corriente alterna. Integrando esta potencia se consigue la energía. Figura modificada de la figura 3.3 del libro ESF [Per23].*

donde d_n es el número de día del año (siendo $d_n = 1$ el 1 de Enero).

El movimiento que describe la Tierra al girar alrededor del Sol, está contenido en un plano conocido como *plano de la eclíptica*, el cual, entre este y el eje polar (línea imaginaria que uno los dos polos de la Tierra) se forma un ángulo conocido como declinación[2], el cual se puede aproximar de forma sencilla de la siguiente manera[3]:

$$\delta = 23{,}45° \cdot \sin\left(\frac{2\pi \cdot (d_n + 284)}{365}\right) \tag{3.2}$$

3.1.2. Movimiento aparente del sol

Las variaciones en la declinación hacen que los días fluctuen su duración. Esta duración se puede calcular mediante el ángulo del amanecer[4] (ω_s), el cúal determina la diferencia angular entre el mediodía solar y el momento en el que amanece. Por lo tango un día durará $2 \cdot |\omega_s|$

[2] Correspondiente a la función `declination`

[3] Por razones de economización del espacio, se va a optar por utilizar las ecuaciones de Cooper [Coo69] por su sencillez. Sin embargo, la función `fSolD` (como se verá en el capítulo 4) puede seleccionar entre 4 tipos de ecuaciones expuestas por diferentes autores (Strous [Str11], Spencer [Spe71] y Michalsky [Mic88]).

[4] Correspondiente a la función `sunrise`.

(ya que es el ángulo entre el amanecer y el mediodía, y el mediodía y el anochecer). Se puede calcular el ángulo del amanecer de la siguiente manera:

$$\omega_s = \begin{cases} -\arccos(-\tan\delta\tan\phi) & \text{si } |\tan\delta\tan\phi| < 1 \\ -\pi & \text{si } -\tan\delta\tan\phi < -1 \\ 0 & \text{si } -\tan\delta\tan\phi > 1 \end{cases} \tag{3.3}$$

donde, ϕ corresponde a la latitud (positiva al norte y negativa al sur).

Sin embargo, cada localización tiene una hora oficial, TO, la cual no se corresponde con el ángulo que forma el Sol a lo largo del día, por lo que resulta necesario calcular esta hora solar verdadera (ω)[5]:

$$\omega = 15 \cdot (TO - AO - 12) + \Delta\lambda + \frac{EoT}{4} \tag{3.4}$$

donde:

- TO es la hora oficial.
- AO es el adelanto oficial durante el horario de invierno.
- $\Delta\lambda$ es la diferencia entre la longitud local y la longitud del huso horario.
- EoT es la ecuación del tiempo[6], se calcula de la siguiente manera:

$$EoT = 229,18 \cdot (-0,0334 \cdot sin(\frac{2\pi}{365,24} \cdot dn) + 0,04184 \cdot sin(2 \cdot \frac{2\pi}{365,24} \cdot dn + 3,5884)) \tag{3.5}$$

3.1.3. Radiación fuera de la atmósfera terrestre

La radiación extra-atmosférica es la radiación directa del Sol que alcanza la superficie de la atmósfera. Ya que la relación entre la distancia con el Sol y tamaño de nuestro planeta, es razonable asumir que su valor es constante en toda la superficie exterior de la atmósfera. Se define la constante solar, B_0 como el valor de irradiancia solar incidente en un plano normal al vector Sol-Tierra. Se acepta como representativo el valor promedio de $B_0 = 1367W/m^2$.

Para calcular la irradiancia incidente en una superficie tangente a la atmósfera en una latitud determinada ($B_0(0)$)[7], se toma la siguiente ecuación:

$$B_0(0) = B_0 \cdot \epsilon_0 \cdot \cos(\theta_{zs}) \tag{3.6}$$

donde, $\cos(\theta_{zs})$[8] es el coseno del ángulo cenital solar[9], y se calcula de la siguiente manera:

$$cos(\theta_{zs}) = cos(\delta)cos(\omega)cos(\phi) + sin(\delta) + sin(\phi) \tag{3.7}$$

Es importate añadir, que con el ángulo cenital, se puede obtener otro ángulo relevante en la geometría solar, se trata del ángulo azimutal, ψ_s[10], el cual, es el ángulo formado por el meridiano

[5] Correspondiente a la función **`sunHour`**.

[6] Correspondiente a la función **`eot`**.

[7] Este cálculo, junto con el de la hora solar, ω, el coseno del ángulo cenital, $theta_{zs}$ y el ángulo azimutal, ψ_s, se pueden calcular mediante la función **`fSolI`**, haciendo uso de sus respectivas funciones.

[8] Correspondiente a la función **`zenith`**.

[9] El ángulo cenital, $theta_{zs}$, es el complementario de la áltura solar, $gamma_s$.

[10] Correspondiente a la función **`azimuth`**.

solar y el meridiano del lugar (Sur en el hemisferio Norte y viceversa).

$$\cos(\psi_s) = signo(\phi) \cdot \frac{\cos(\delta)\cos(\omega)\cos(\phi) - \cos(\phi)\sin(\delta)}{\sin(\theta_{zs})} \tag{3.8}$$

Para calcular la irradiación diaria extra-atmoférica[11], $B_{0d}(0)$[12], se puede integrar la ecuación 3.6:

$$B_{0d}(0) = -\frac{24}{\pi} B_0 \epsilon_0 (\omega_s sin\phi sin\delta + cos\phi cos\delta sin\omega_s) \tag{3.9}$$

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)

Todas estas ecuaciones, están recogidas en la función `calcSol`, la cual computa todos los ángulos solares, diarios e intradiarios, necesarios para la simulación de sistemas fotovoltaicos.

3.1.4. Influencia de la atmósfera

La radiación solar, al atravesar la atmósfera, es afectada por reflexión, atenuación y difusión, lo que altera sus características. La reflexión en las nubes reduce la radiación que llega a la Tierra, mientras que la absorción por vapor de agua, ozono y CO_2 cambia el espectro de la radiación. Además, la dispersión por partículas afecta la distribución espacial. Existen tres tipos de difusión según el tamaño de las partículas en interacción:

- **Difusión de Rayleigh**: La longitud de onda esmayor que el tamaño de la partícula, ocurre en las capas altas y causa el color azul del cielo.
- **Difusión de Mie**: La longitud de onda es similar al tamaño de la partícula, ocurre en las capas bajas.
- **Difusión no selectiva**: La longitud de onda es menor que el tamaño de la partícula.

Resulta útil definir la masa de aire (AM, *air mass*) como la relación entre el camino recorrido por los rayos directos del Sol a través de la atmósfera hasta la superficie receptora y el que recorerían en caso de incidencia vertical. Se puede aproximar de la siguiente manera:

$$AM = 1/\cos(\theta_{zs}) \tag{3.10}$$

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

[11] Correspondiente a la función `boOd`.

[12] Este cálculo, junto con la declinación, δ, la corrección por excentricidad, ϵ_0, la ecuación el tiempo, EoT, y el ángulo del amanecer, ω_s, se pueden calcular mediante la función `fSolD`, haciendo uso de sus respectivas funciones.

Para el cálculo de la irradiancia solar que finalmente incide en una superficie arbitraria localizada en corteza terrestre será útil distinguir tres contribuciones diferentes. Estas contribuciones, comúnmente denominadas componentes, son:

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

$$G = B + D + R \tag{3.11}$$

3.1.5. 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:

$$K_{Td} = \frac{G_d(0)}{B_{0d}(0)} \tag{3.12}$$

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

$$K_{Tm} = \frac{G_{d,m}(0)}{B_{0d,m}(0)} \tag{3.13}$$

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]:

$$F_{Dm} = 1 - 1,13 \cdot K_{Tm} \tag{3.14}$$

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[13]:

$$D_d(0) = F_D \cdot G_d(0) \tag{3.15}$$

$$B_d(0) = G_d(0) - D_d(0) \tag{3.16}$$

Estas expresiones, se recogen en la función `calcG0`, la cual, calcula las componentes de la irradiancia intradiaria y la irradiación diaria (además de medias mensuales de estas y sumas anuales). Se vale de las funciones `fCompD`, para la irradiación, y `fCompI`, para la irradiancia.

[13] Correpondiente a la familia de funciones `corrFdKt`.

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] (3.17):

$$r_D = \frac{D(0)}{D_d(0)} = \frac{B_0(0)}{B_{0d}(0)} \tag{3.17}$$

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 3.6 y 3.9:

$$\frac{B_0(0)}{B_{0d}(0)} = \frac{\pi}{T} \cdot \frac{cos(\omega) - cos(\omega_s)}{\omega_s \cdot cos(\omega_s) - sin(\omega_s)} = r_D \tag{3.18}$$

el mismo análisis mostró una relación entre la irradiancia e irradiación global asimilable a una función dependiente de la hora solar (3.19):

$$r_G = \frac{G(0)}{G_d(0)} = r_D \cdot (a + b \cdot cos(w)) \tag{3.19}$$

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).

$$B(\beta, \alpha) = B(0) \cdot \frac{max(0, cos(\theta_s))}{cos(\theta_{zs})} \tag{3.20}$$

  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*.

$$D(\beta, \alpha) = D^I(\beta, \alpha) + D^C(\beta, \alpha) \tag{3.21}$$

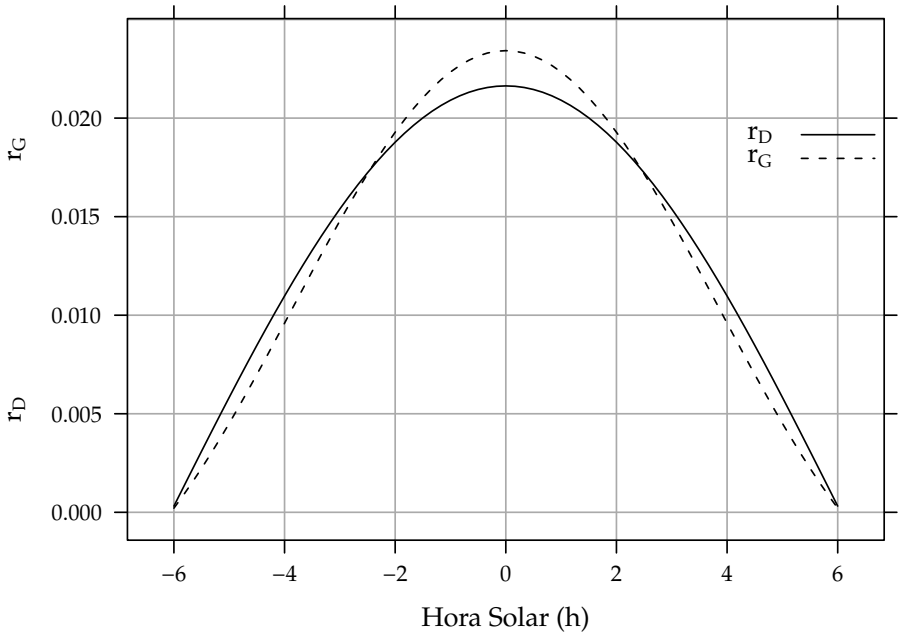


FIGURA 3.2: *Perfil de irradiancia difusa y global obtenido a partir del generador empírico de [CR79] para valores de irradiancia tomadas cada 10 minutos. Figura 3.4 del libro ESF [Per23].*

$$D^I(\beta,\alpha) = D(0)(1-k_1)\cdot\frac{1+cos(\beta)}{2} \tag{3.22}$$

$$D^C(\beta,\alpha) = D(0)\cdot k_1\cdot\frac{max(0,cos(\theta_s))}{cos(\theta_{zs})} \tag{3.23}$$

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 3.24, se utiliza el factor $\frac{1-cos(\beta)}{2}$, complemetario al factor de visión de la difusa isotrópica (figura 3.3)

$$R(\beta,\alpha) = \rho\cdot G(0)\cdot\frac{1-cos(\beta)}{2} \tag{3.24}$$

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

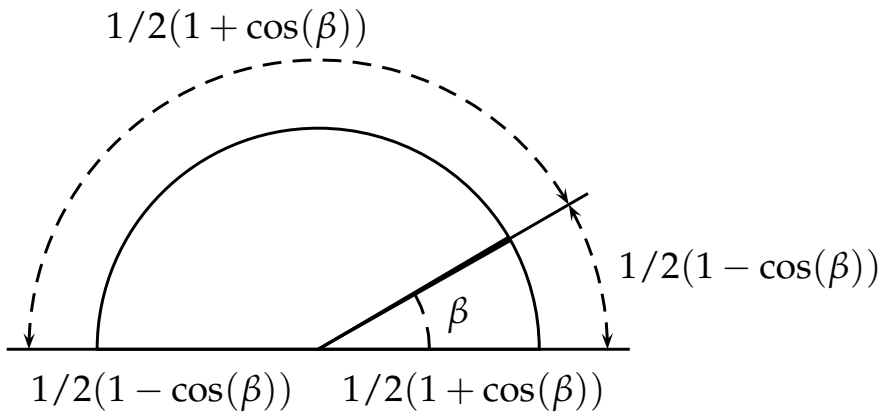


Figura 3.3: *Ángulo de visión del cielo. Figura 3.5 del libro ESF [Per23].*

para radiación directa se expresa en la ecuación 3.25:

$$B_{ef}(\beta,\alpha) = B(\beta,\alpha)\cdot[\frac{T_{sucio}(0)}{T_{limpio}(0)}]\cdot(1-FTB(\theta_s)) \tag{3.25}$$

donde $FTB(\theta_s)$ es el factor de pérdidas angulares, que se calcula con la ecuación[14] 3.26:

$$FTB(\theta_s) = \frac{exp(-\frac{cos(\theta_s)}{a_r}) - exp(-\frac{1}{a_r})}{1-exp(-\frac{1}{a_r})} \tag{3.26}$$

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[15] 3.27 y 3.2.2:

$$\text{FTD}(\beta) \approx exp[-\frac{1}{a_r}\cdot(c_1\cdot(sin\beta + \frac{\pi-\beta-sin\beta}{1+cos\beta}) + c_2\cdot(sin\beta+\frac{\pi-\beta-sin\beta}{1+cos\beta})^2)] \tag{3.27}$$

$$\text{FTR}(\beta) \approx exp[-\frac{1}{a_r}\cdot(c_1\cdot(sin\beta + \frac{\beta-sin\beta}{1-cos\beta}) + c_2\cdot(sin\beta+\frac{\beta-sin\beta}{1-cos\beta})^2)] \tag{3.28}$$

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)

[14] Implementada en la función `fInclin`.
[15] Implementadas en la función `fInclin`.

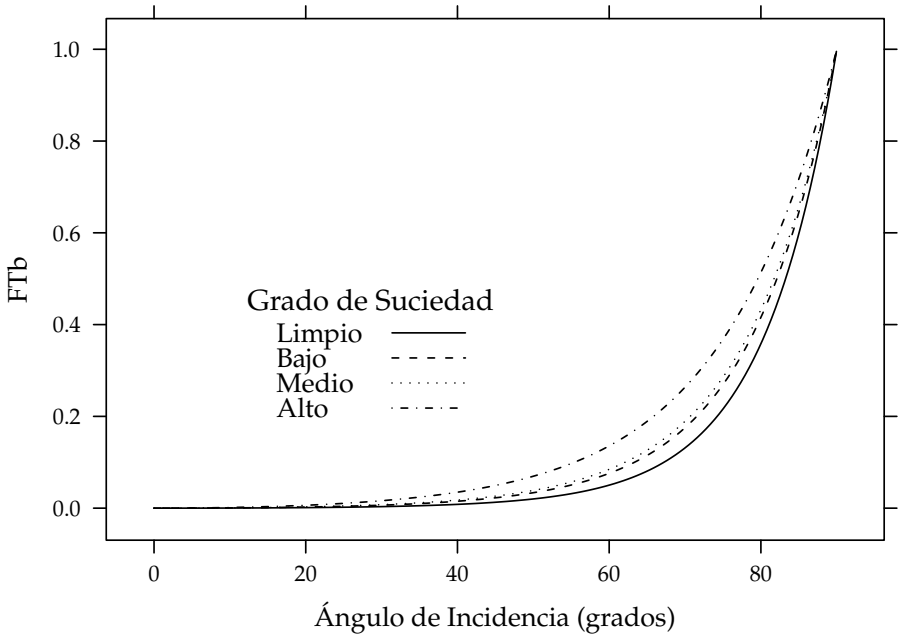


FIGURA 3.4: *Pérdidas angulares de un módulo fotovoltaico para diferentes grados de suciedad en función del ángulo de incidencia. Figura 3.7 del libro ESF [Per23].*

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

Para estas componenetes el cálculo de irradiancia efectiva es similar al de la irradiancia directa (ecuaciones 3.29 y 3.31). Para la componente difusa circunsolar emplearemos el factor de pérdidas angulares de la irradiancia efectiva(ecuacion 3.30):

$$D^I_{ef}(\beta,\alpha) = D^I(\beta,\alpha)\cdot[\frac{T_{sucio}(0)}{T_{limpio}(0)}]\cdot(1-FT_D(\beta)) \tag{3.29}$$

$$D^C_{ef}(\beta,\alpha) = D^C(\beta,\alpha)\cdot[\frac{T_{sucio}(0)}{T_{limpio}(0)}]\cdot(1-FT_B(\theta_s)) \tag{3.30}$$

$$R_{ef}(\beta,\alpha) = R(\beta,\alpha)\cdot[\frac{T_{sucio}(0)}{T_{limpio}(0)}]\cdot(1-FT_R(\beta)) \tag{3.31}$$

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[16]. Comparando la irradiación global incidente con la irradición efectiva, se puede evaluar el impacto de la suciedad y el desajuste del ángulo en perídoods prolongados.

[16]Correpondiente a la función `calcGef`.

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[17] 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).

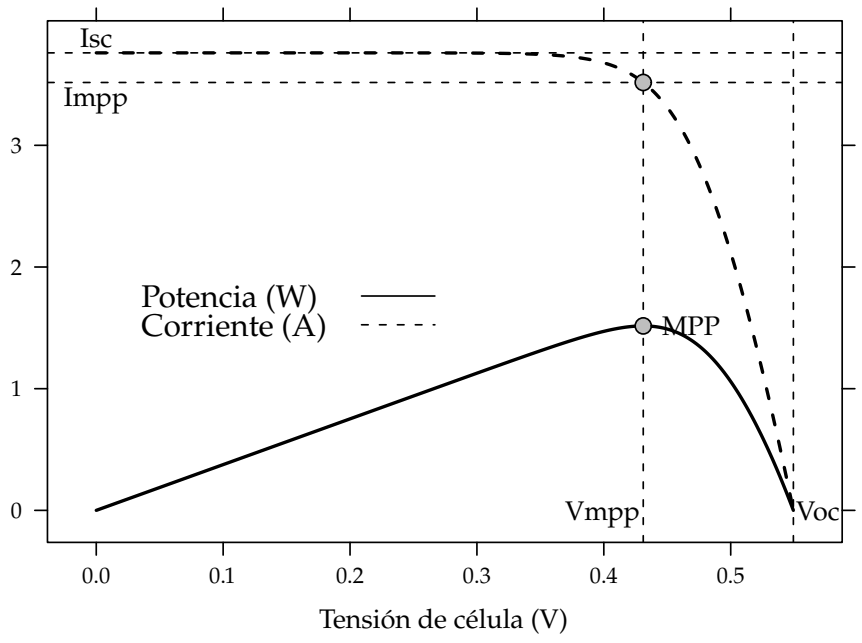


Figura 3.5: *Curvas corriente-tensión(línea discontinua) y potencia-tensión(línea continua) de una célula solar ($T_a = 20°C$ y $G = 800W/m^2$). Figura 4.6 del libro ESF [Per23].*

[17] Los cálculos de este apartado, quedan recogidos en la función `fProd`.

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:

$$FF = \frac{I_{mpp} \cdot V_{mpp}}{I_{sc} \cdot V_{oc}} \tag{3.32}$$

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 3.33).

$$\eta = \frac{I_{mpp} \cdot V_{mpp}}{P_L} \tag{3.33}$$

donde $P_L = A_c \cdot G_{ef}$ representa la potencia luminosa que incide en la célula.Como es evidente de la ecuación 3.33, 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.

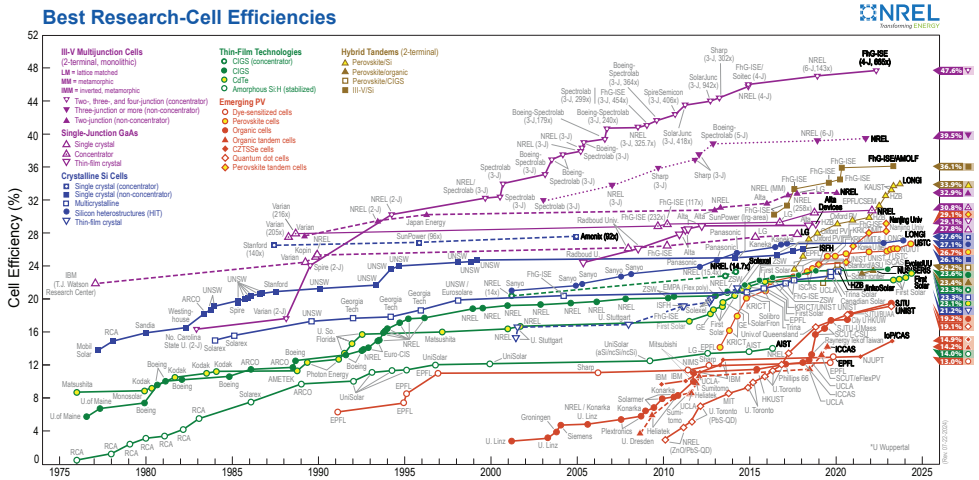


FIGURA 3.6: *Evolución de la eficiencia de células según la tecnología (según el National Renewable Energy Laboratory [Nat24] (EEUU)).*

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:

- Irradiancia: $G_{stc} = 1000W/m^2$ con incidencia normal.
- Temperatura de célula: $T_c^* = 25°C$.
- Masa de aire: $AM = 1,5$.[18]

Frecuentemente los fabricantes informan de los valores de las tensiones V_{oc}^* y V_{mpp}^* y las corrientes I_{sc}^* y I_{mpp}^*[19]. 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 que definen el comportamiento de un módulo fotovoltaico[20] 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.

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.

[18] 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 ($AM = 1/cos\theta_{zs}$).

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

[20] Los cálculos de este apartado, quedan recogidos en la función `fProd`.

- Temperatura *ambiente*: $T_a = 20°C$.
- Velocidad de viento: $v_v = 1m/s$.

La ecuación 3.34 expresa una aproximación aceptable del comportamiento térmico de una célula integrada en un módulo en base a las consideraciones previas:

$$T_c = T_a + G_{ef} \cdot \frac{NOCT - 20}{800} \tag{3.34}$$

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 3.35. Y, por último, mediante la ecuación 3.36 se puede calcular I_{sc}.

$$V_{oc}(T_c) = V_{oc}^* + (T_c - T_c^*) \cdot \frac{dV_{oc}}{dT_c} \cdot N_{cs} \tag{3.35}$$

$$I_{sc} = G_{ef} \cdot \frac{I_{sc}^*}{G^*} \tag{3.36}$$

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 3.32 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

$$FF = FF^* \tag{3.37}$$

entonces

$$\frac{I_{mpp} \cdot V_{vmpp}}{I_{sc} \cdot V_{oc}} = \frac{I_{mpp}^* \cdot V_{vmpp}^*}{I_{sc}^* \cdot V_{oc}^*} \tag{3.38}$$

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:

$$V_{tn} = \frac{V_t \cdot (273 + 25)}{300} \tag{3.39}$$

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

$$R_s^* = \frac{\frac{V_{oc}^*}{N_{cs}} - \frac{V_{mpp}^*}{N_{cs}} + m \cdot V_{tn} \cdot ln(1 - \frac{I_{mpp}^*}{I_{sc}^*})}{\frac{I_{mpp}^*}{N_{cp}}} \tag{3.40}$$

- **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.

$$r_s = R_s^* \cdot (\frac{N_{cs}}{N_{cp}} \cdot \frac{I_{sc}}{V_{oc}}) \tag{3.41}$$

- **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:

$$k_{oc} = \frac{V_{oc}/N_{cs}}{m \cdot V_t \cdot \frac{T_c+273}{300}} \tag{3.42}$$

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

$$i_{mpp} = 1 - \frac{D_M}{k_{oc}} \tag{3.43}$$

$$v_{mpp} = 1 - \frac{ln(k_{oc}/D_M)}{k_{oc}} - r_s \cdot i_{mpp} \tag{3.44}$$

donde:

$$D_M = D_{M0} + 2 \cdot r_s \cdot D_{M0}^2 \tag{3.45}$$

$$D_{M0} = \frac{k_{oc} - 1}{k_{oc} - lnk_{oc}} \tag{3.46}$$

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:

$$P_{mpp} = I_{mpp} \cdot V_{mpp} \tag{3.47}$$

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[21].

$$P_g^* = N_s \cdot N_p \cdot P_m^* \tag{3.48}$$

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:

$$p_i = \frac{P_{DC}}{P_{inv}} \tag{3.49}$$

[21] Los cálculos de este apartado, quedan recogidos en las funciones `fProd` y `prodCGPV`.

$$p_o = \frac{P_{AC}}{P_{inv}} \tag{3.50}$$

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

$$\eta_{inv} = \frac{p_o}{p_o + k_0 + k_1 p_o + k_2 p_o^2} \tag{3.51}$$

De las dos ecuaciones anteriores se puede deducir:

$$p_i = p_o + k_0 + k_1 p_o + k_2 p_o^2 \tag{3.52}$$

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

$$k_2 p_o^2 + (k_1 + 1) p_o + (k_0 - p_i) = 0 \tag{3.53}$$

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

$$P_{AC} = p_o \cdot P_{inv} \tag{3.54}$$

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

$$E_{AC} = \int_T P_{AC}\, dt \tag{3.55}$$

y la productividad:

$$Y_f = \frac{E_{ac}}{P_g^*} \tag{3.56}$$

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

$$P_H = g \cdot \rho \cdot Q \cdot H_V \tag{3.57}$$

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

$$P_H = 2{,}725 \cdot Q \cdot H_v \tag{3.58}$$

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:

$$H_T = H_v + H_f \tag{3.59}$$

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

$$P_{el} = \frac{P_H + P_f}{\eta_{mp}} \tag{3.60}$$

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:

$$P_{el} = P_g^* \cdot \frac{G}{G_{stc}} \frac{\eta_g}{\eta_g^*} \cdot \eta_{inv} \tag{3.61}$$

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:

$$Q_d = \int_d \frac{G}{G^*} \cdot P_g^* \cdot \eta_g \cdot \frac{\eta_{ig}}{\eta_{ig}^*} \cdot \eta_{inv} \cdot \eta_{mp}\, dt \tag{3.62}$$

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.

$$Q_d = \frac{P_g^*}{2{,}725 \cdot G^* \cdot H_{TE}} \int \left(\frac{G}{\eta_{ig}^* \eta_m^* \eta_{inv} \eta_{mp}} \right) dt \tag{3.63}$$

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:

$$H_{TE} = H_{ot} + H_{st} + \left(\frac{H_{dt} - H_{st}}{Q_T} \right) Q_{AP} + H_f(Q_{AP}) \tag{3.64}$$

donde:

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

$$P_g^* = \frac{10 \cdot HTE \cdot Q_d}{\frac{G_d}{G_{stc}}} \tag{3.65}$$

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 ordenador[22]. 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:
$$H = a \cdot f^2 + b \cdot f \cdot Q + c \cdot Q^2 \tag{3.66}$$
   - Donde a, b, y c son coeficientes característicos de la bomba y f es la frecuencia.
2. Relaciones de semejanza para bombas centrífugas:
$$\frac{f_1}{f_2} = \frac{Q_1}{Q_2} = \left(\frac{H_1}{H_2}\right)^{1/2} = \left(\frac{P_1}{P_2}\right)^{1/3} \tag{3.67}$$
3. Cálculo de caudal y altura a frecuencia nominal (50 Hz):
$$Q_{50} = \frac{50 \cdot Q}{f} \tag{3.68}$$
$$H_{50} = H \cdot \left(\frac{50}{f}\right)^2 \tag{3.69}$$
4. Ecuación de potencia hidráulica:
$$P_{h,50} = 2{,}725 \cdot Q_{50} \cdot H_{50} \tag{3.70}$$
5. Potencia mecánica en el eje de la bomba a 50 Hz:
$$P_{b,50} = \frac{P_{h,50}}{\eta_b} \tag{3.71}$$
6. Potencia mecánica a frecuencia f:
$$P_b = P_{b,50} \cdot \left(\frac{f}{50}\right)^3 \tag{3.72}$$
7. Potencia eléctrica demandada por el motor:
$$P_{bc} = P_b \cdot \frac{50}{f} \tag{3.73}$$
$$P_{e,50} = \frac{P_{bc}}{\eta_m} \tag{3.74}$$
$$P_e = P_{e,50} \cdot \frac{f}{50} \tag{3.75}$$

[22] Correspondientes a la función `fPump`.

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

$$G = G_{max} \cdot \cos\left(\frac{t}{t_0} \cdot \frac{\pi}{2}\right) \cdot \left[1 + s \cdot \left(1 - \cos\left(\frac{t}{t_0} \cdot \frac{\pi}{2}\right)\right)\right] \tag{3.76}$$

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:

$$s = \frac{d \cdot \frac{\pi}{2} - 1}{1 - \frac{\pi}{4}} \tag{3.77}$$

siendo d el factor de conjunto de datos calculado con:

$$d = \frac{G_d}{G_{max} \cdot h} \tag{3.78}$$

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[23].

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).

$$GCR = \frac{A_G}{A_T} \tag{3.79}$$

- 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.

$$ROT = \frac{A_T}{A_G} \tag{3.80}$$

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:

$$d = \frac{h}{tan\gamma_s} \tag{3.81}$$

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:

$$d_{min} = \frac{h}{tan(61° - |\phi|)} \tag{3.82}$$

[23] Correpondiente a las funciones `calcShd` (cálculo de sombras) y `optimShd` (optimización de distancias).

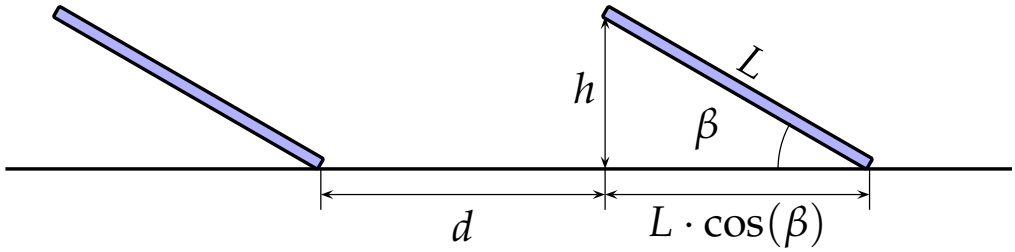


FIGURA 3.7: *Dimensiones y distancias entre filas de un sistema estático. Figura 6.10 del libro ESF [Per23].*

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:

$$ROT = \frac{L_{ns} \cdot L_{eo}}{L \cdot W} \tag{3.83}$$

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

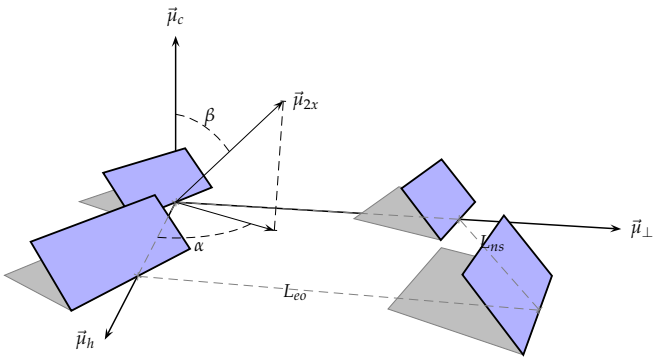


FIGURA 3.8: *Sombras mutuas en un conjunto de cuatro seguidores. Figura 6.11 del libro ESF [Per23].*

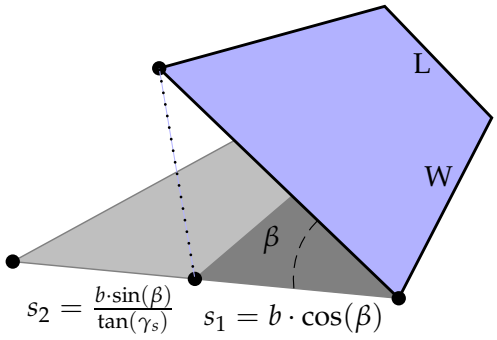


Figura 3.9: *Dimensiones de un seguidor a doble eje y longitud de su sombra arrojada. Figura 6.12 del libro ESF [Per23].*

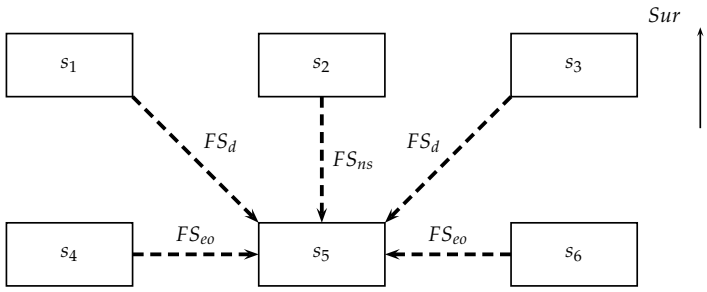


Figura 3.10: *Posibles sombras en un conjunto de seis seguidores. Figura 6.13 del libro ESF [Per23].*

(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 $l_{ns} = \frac{L_{ns}}{W}$:

$$\begin{matrix} |l_{eo} \cdot \cos(\psi_s)| < 1 \\ |l_{eo} \cdot \sin(\psi_s)| < s \end{matrix} \Rightarrow FS_{eo} = \frac{(1 - |l_{eo} \cos(\psi_s)|) \cdot (s - |l_{eo} \sin(\psi_s)|)}{s} \tag{3.84}$$

$$\begin{matrix} |l_{ns} \cdot \cos(\psi_s)| < s \\ |l_{ns} \cdot \sin(\psi_s)| < 1 \end{matrix} \Rightarrow FS_{ns} = \frac{(s - |l_{ns} \cos(\psi_s)|) \cdot (1 - |l_{ns} \sin(\psi_s)|)}{s} \tag{3.85}$$

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

$$FS_d = \frac{[s - (|l_{eo} \cdot \sin(\psi_s)| + |l_{ns} \cos(\psi_s)|)] \cdot [1 - (|l_{eo} \cdot \cos(\psi_s)| - |l_{ns} \sin(\psi_s)|)]}{s} \tag{3.86}$$

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:

$$s = s_1 + s_2 \tag{3.87}$$

$$s_1 = b \cdot \cos(\beta) \tag{3.88}$$

$$s_2 = \frac{b \cdot \sin(\beta)}{|\tan(\gamma_s)|} \tag{3.89}$$

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:

$$FS_{eo} = \frac{(1 - l_{eo}\cos(\psi_s)) \cdot (s - l_{eo}\sin(\psi_s))}{s} \tag{3.90}$$

$$FS_{ns} = \frac{(s - l_{ns}\cos(\psi_s)) \cdot (1 - l_{ns}\sin(\psi_s))}{s} \tag{3.91}$$

$$FS_{d} = \frac{[s - (l_{eo} \cdot \sin(\psi_s) + l_{ns}\cos(\psi_s))] \cdot [1 - (l_{eo} \cdot \cos(\psi_s) - l_{ns}\sin(\psi_s))]}{s} \tag{3.92}$$

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[24] 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}).
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:

$$\begin{aligned} FS_{eo} &= \frac{s - l_{eo}}{s} \\ &= 1 - l_{eo} \cdot \cos(\beta) \\ &= 1 - l_{eo} \cdot \frac{\sin(\omega)}{\sqrt{\sin^2(\omega) + (\cos(\omega)\cos(\phi) + \tan(\delta)\sin(\phi))^2}} \end{aligned} \tag{3.93}$$

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.

[24] 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.

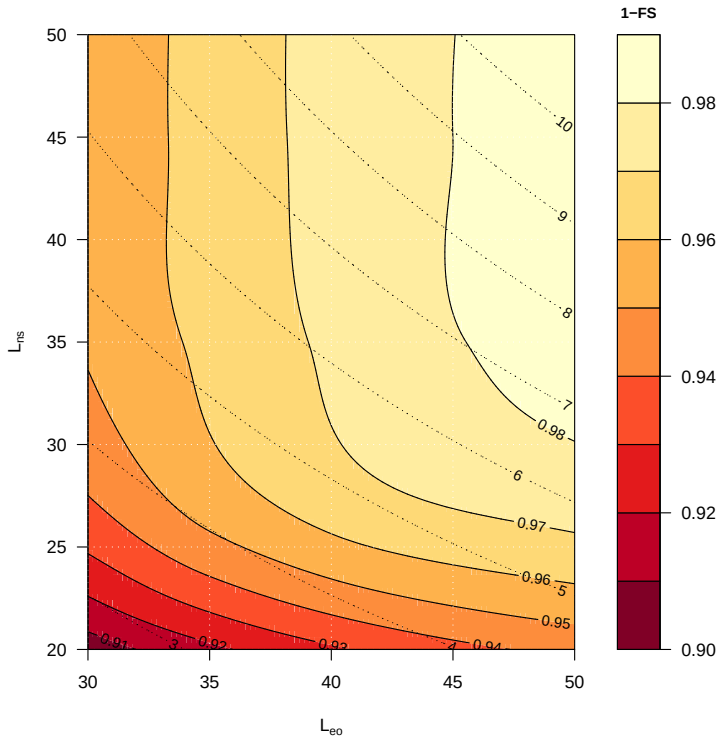


Figura 3.11: *Ábaco para planta de seguimiento a doble eje. Recoge el ratio entre la energía anual producida por un seguidor afectado por sombras mutuas (E_{acS}) y la producida por un seguidor sin sombreado (E_{ac0}). Las curvas de color negro representan la fracción de energía no afectada por sombras. Las curvas de puntos representan el valor del ROT. Figura 6.14 del libro ESF [Per23].*

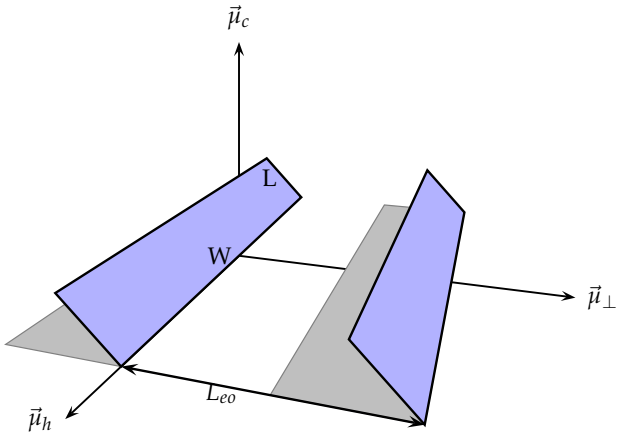


Figura 3.12: *Dimensiones básicas en sistemas con seguidores de eje horizontal. Figura 6.16 del libro ESF [Per23].*

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 3.93 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:

$$\frac{l_{eo}}{\cos(\beta_0 - \beta)} = \frac{1}{\cos \beta_0} \tag{3.94}$$

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

$$\beta = \beta_0 - \arccos(l_{eo} \cdot \cos \beta_0) \tag{3.95}$$

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[1]:

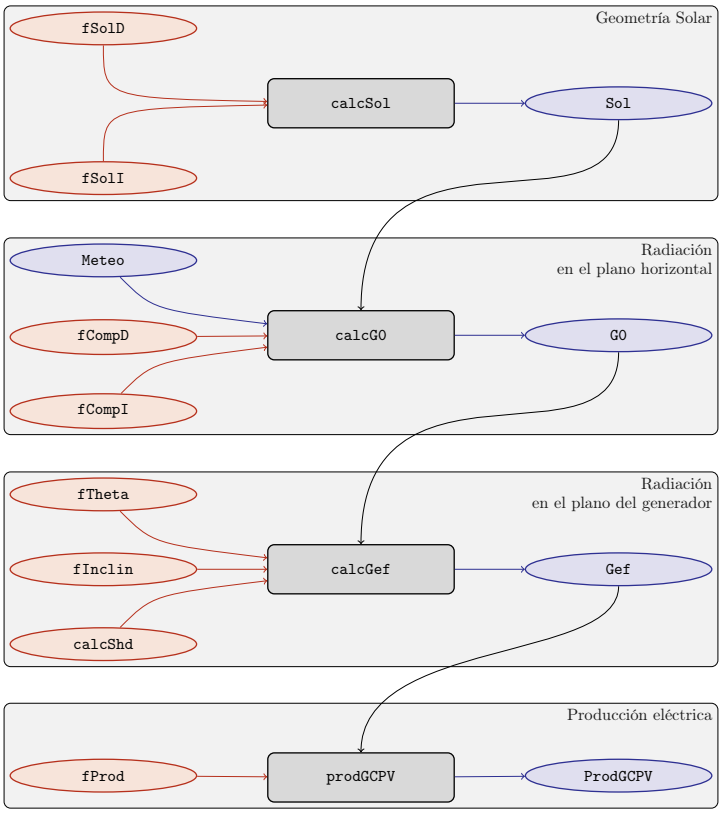


FIGURA 4.1: *Proceso de cálculo de las funciones de* **`solaR2`**

[1]Todas las funciones recogidas en este capítulo, están descritas en el manual de uso del paquete **`solaR2`**, el cual, está disponible en el apendice A de este documento.

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**.
  - 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**.

```
1 lat <- 37.2
2 BTd <- fBTd(mode = 'prom')
3 solD <- fSolD(lat = lat, BTd = BTd)
4 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
 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
```

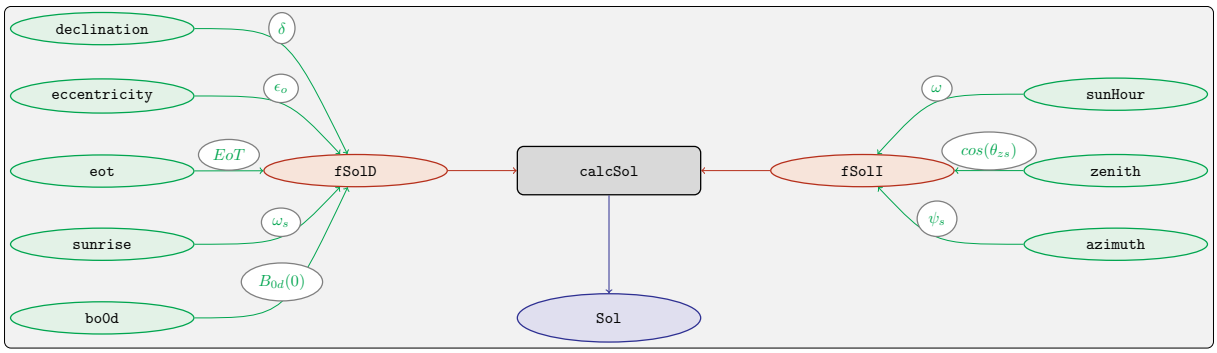
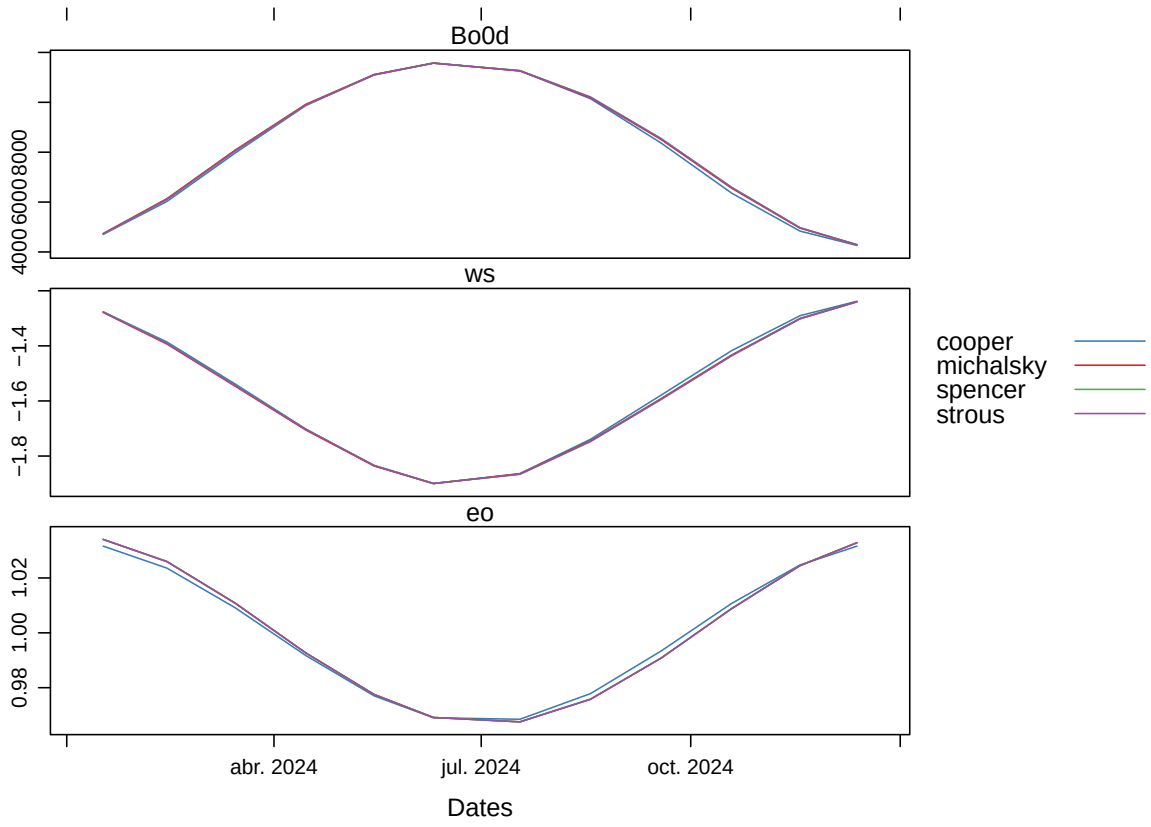


Figura 4.2: *Cálculo de la geometría solar mediante la función* ***calcSol****, la cual unifica las funciones* ***fSolD*** *y* ***fSolI*** *resultando en un objeto clase* ***Sol*** *el cual contiene toda la información geométrica necesaria para realizar las siguientes estimaciones.*

Tiene los siguientes argumentos:

- **lat**: para introducir la latitud en grados.
- **BTd**: para introducir la base temporal **diaria** sobre la que se harán los cálculos.
- **method**: para elegir el método de cálculo. Se puede elegir entre: **michalsky**, **cooper**, **spencer** y **strous**

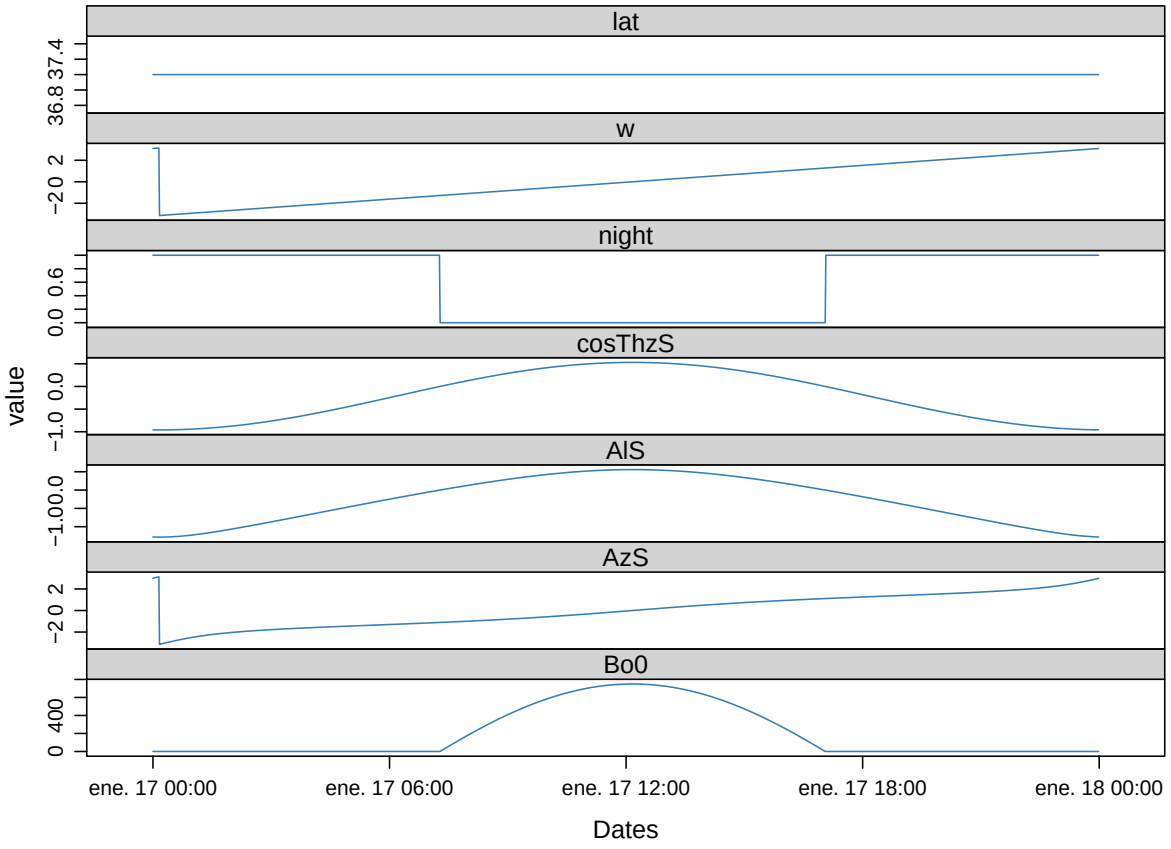
```
BTd <- fBTd(mode = 'prom')
solD_michalsky <- fSolD(lat, BTd, method = 'michalsky')
solD_michalsky[, group := 'michalsky']
solD_cooper <- fSolD(lat, BTd, method = 'cooper')
solD_cooper[, group := 'cooper']
solD_spencer <- fSolD(lat, BTd, method = 'spencer')
solD_spencer[, group := 'spencer']
solD_strous <- fSolD(lat, BTd, method = 'strous')
solD_strous[, group := 'strous']
solD_all <- rbind(solD_michalsky, solD_cooper,
                  solD_spencer, solD_strous)
xyplot(eo + ws + Bo0d ~ Dates , solD_all,
       groups = group, type = 'l', auto.key = TRUE,
       par.settings = solaR.theme, scales = list(y = 'free'),
       ylab = '', layout = c(1, 3))
```



- **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 3.6].

  Su argumento principal es **sample** con el cual se puede determinar el intervalo intradiario de cálculos.

```
1 solD <- solD[1] #Computo solo un día a fin mejorar la visualización
2 solI <- fSolI(solD = solD, sample = 'min')
3 xyplot(solI)
```

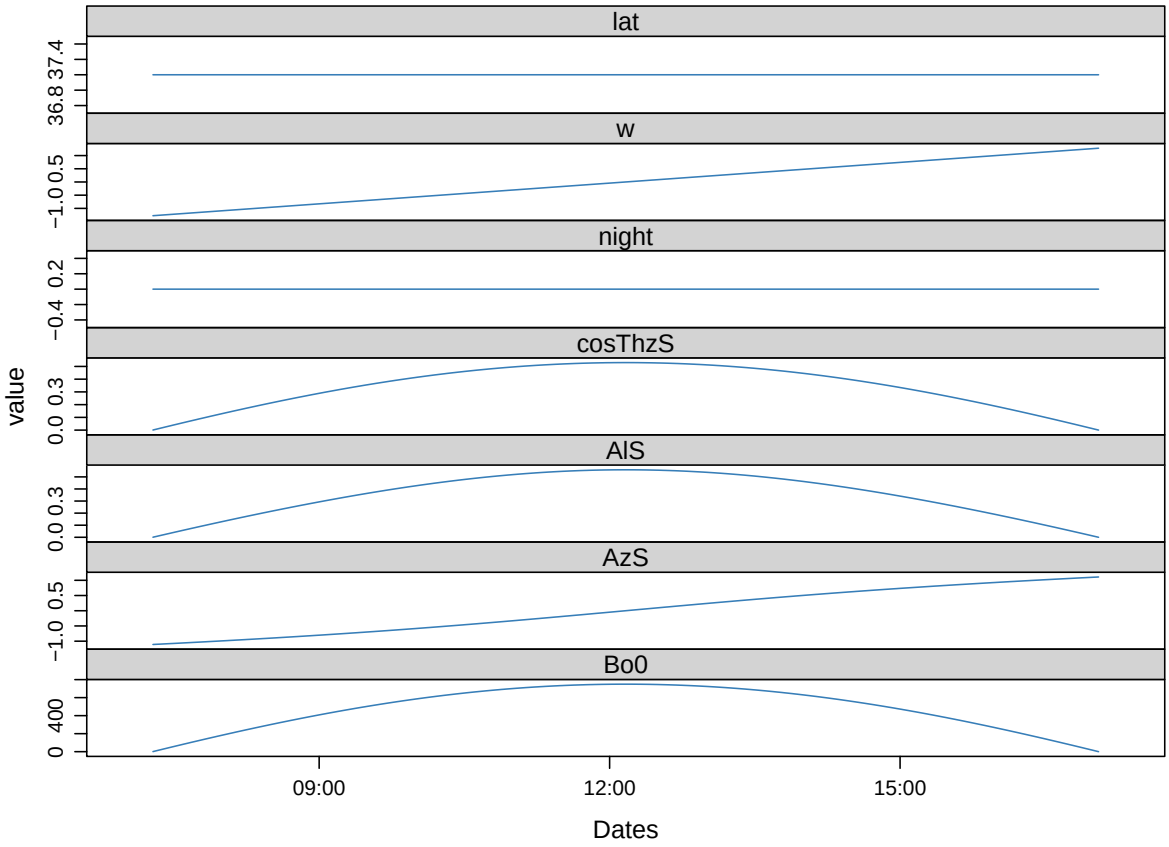


[2]Cuando la hora solar verdadera excede los ángulos en los que amanece y anochece ($|\omega| >= |\omega_s|$), el Sol queda por debajo de la línea del horizonte, por lo que es de noche.

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

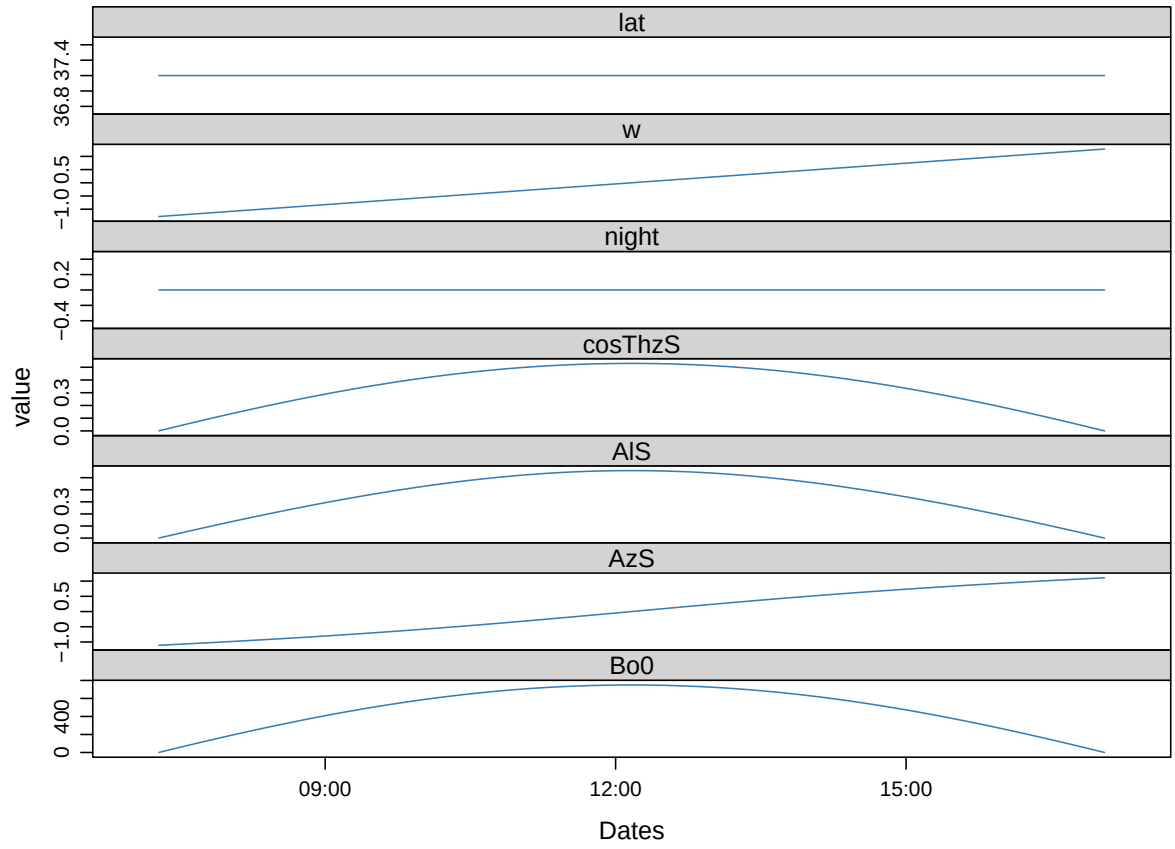
Además, como los datos nocturnos aportan poco a los cálculos que atañen a este proyecto, **fSolI** presenta la posibilidad de eliminar estos datos con el argumento **keep.night**.

```
solI_nigth <- fSolI(solD = solD, sample = 'min', keep.night = FALSE)
xyplot(solI_nigth)
```



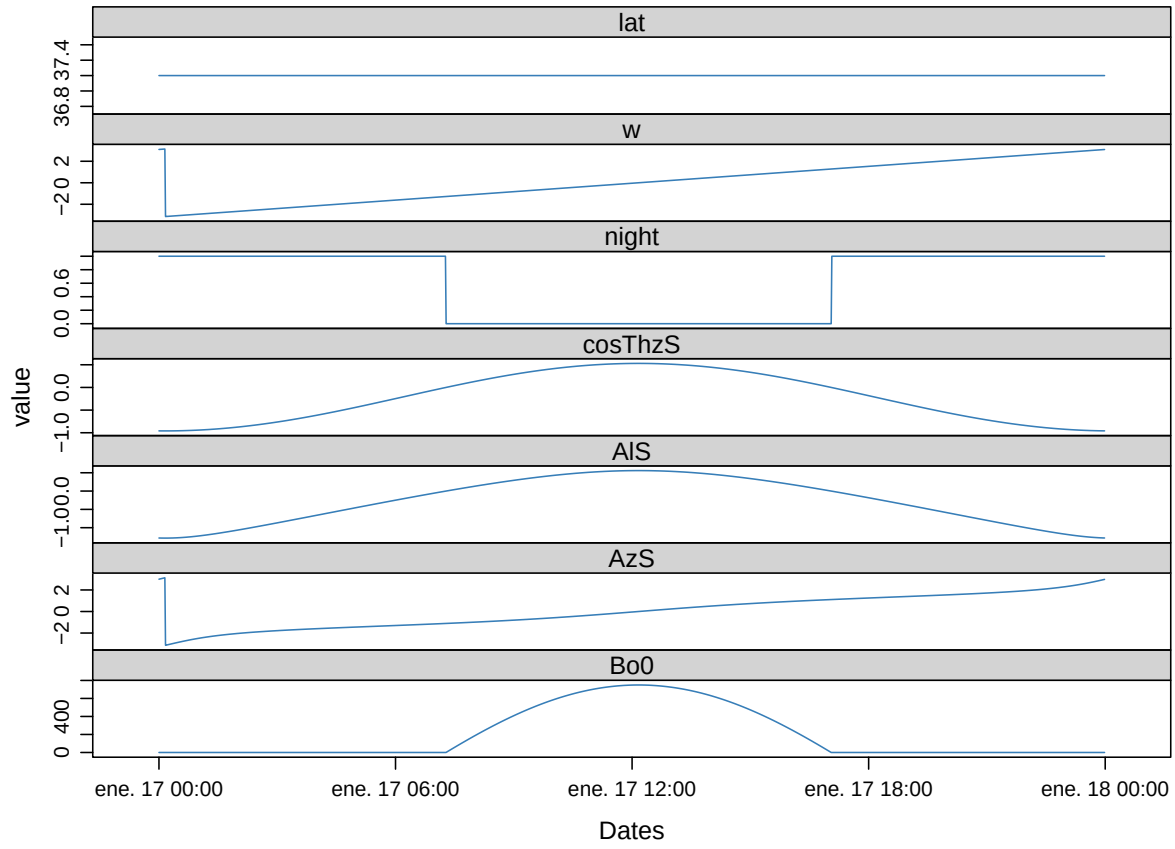
Aparte, en vez de identificar el intervalo intradiario (con el argumento **sample**), se puede dar directamente la base temporal intradiaria.

```
BTd <- solD$Dates
BTi <- fBTi(BTd, sample = 'min')
solI_BTi <- fSolI(solD, BTi = BTi, keep.night = FALSE)
xyplot(solI_BTi)
```



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

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



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. Los argumentos de **calcSol** son las argumentos de las funciones anteriores

```
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.   :-0.3627   Min.   :1.034   Min.   :-0.04553   Min.   :-1.279
 1st Qu.:2024-01-17   1st Qu.:-0.3627   1st Qu.:1.034   1st Qu.:-0.04553   1st Qu.:-1.279
 Median :2024-01-17   Median :-0.3627   Median :1.034   Median :-0.04553   Median :-1.279
 Mean   :2024-01-17   Mean   :-0.3627   Mean   :1.034   Mean   :-0.04553   Mean   :-1.279
 3rd Qu.:2024-01-17   3rd Qu.:-0.3627   3rd Qu.:1.034   3rd Qu.:-0.04553   3rd Qu.:-1.279
 Max.   :2024-01-17   Max.   :-0.3627   Max.   :1.034   Max.   :-0.04553   Max.   :-1.279
      Bo0d
 Min.   :4739
 1st Qu.:4739
 Median :4739
 Mean   :4739
 3rd Qu.:4739
 Max.   :4739

Intradaily values:
     Dates                          w              night           cosThzS             AlS
 Min.   :2024-01-17 00:00:00   Min.   :-2.92240   Mode :logical   Min.   :-0.9586   Min.   :-1.2819
```

```
1st Qu.:2024-01-17 05:45:00   1st Qu.:-1.41740   FALSE:10        1st Qu.:-0.7289   1st Qu.:-0.8172
Median :2024-01-17 11:30:00   Median : 0.08759   TRUE :14        Median :-0.2143   Median :-0.2160
Mean   :2024-01-17 11:30:00   Mean   : 0.08766                   Mean   :-0.2144   Mean   :-0.2724
3rd Qu.:2024-01-17 17:15:00   3rd Qu.: 1.59259                   3rd Qu.: 0.3003   3rd Qu.: 0.3051
Max.   :2024-01-17 23:00:00   Max.   : 3.09905                   Max.   : 0.5295   Max.   : 0.5580
     AzS                Bo0
Min.   :-2.49463   Min.   :  0.0
1st Qu.:-1.18986   1st Qu.:  0.0
Median : 0.09526   Median :  0.0
Mean   : 0.08773   Mean   :197.0
3rd Qu.: 1.28896   3rd Qu.:424.5
Max.   : 3.00158   Max.   :748.4
```

4.2. Datos meteorológicos

Para el procesamiento de datos meteorologicos, **solaR2** provee una serie de funciones 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. Como argumentos tiene:
  - **G0dm**: vector con medias mensuales de irradiación global horizontal.
  - **Ta**: vector con la temperatura ambiente.
  - **lat**: latitud en grados.
  - **year**: año de los datos. Por defecto presenta el año actual.
  - **source**: información de la fuente de los datos.

```
1 G0dm = c(2.766,3.491,4.494,5.912,6.989,7.742,
2          7.919,7.027,5.369,3.562,2.814,2.179) * 1000;
3 Ta = c(10, 14.1, 15.6, 17.2, 19.3, 21.2,
4        28.4, 29.9, 24.3, 18.2, 17.2, 15.2)
5 BD <- readG0dm(G0dm = G0dm, Ta = Ta, lat = 37.2)
6 xyplot(BD)
```

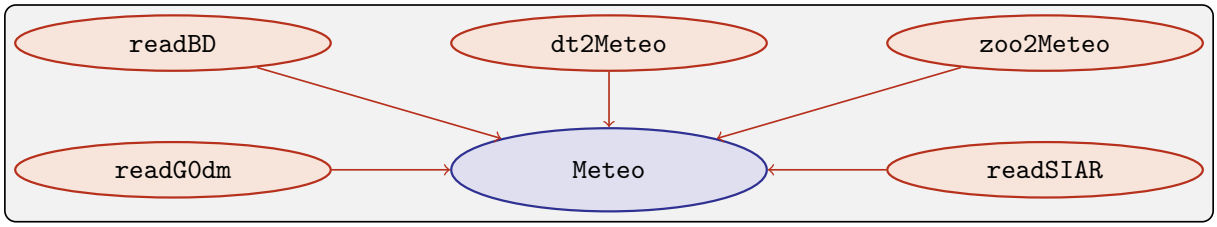
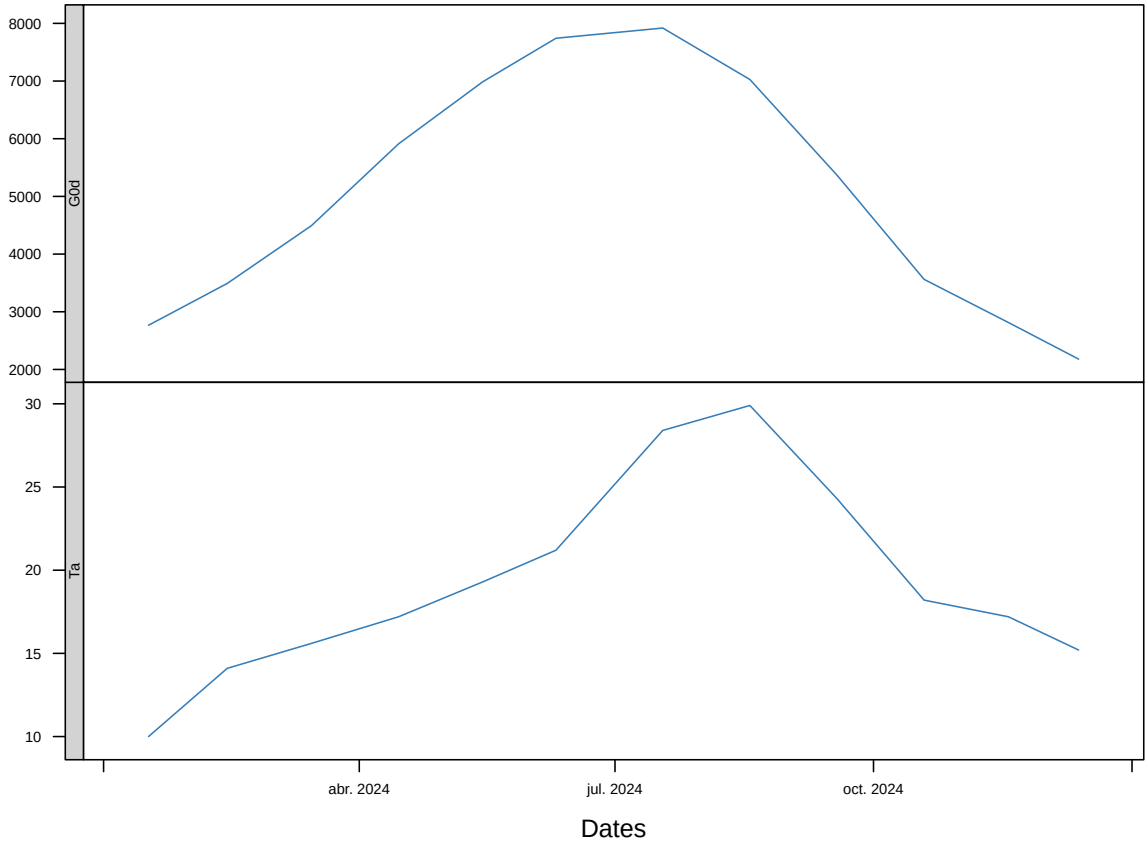


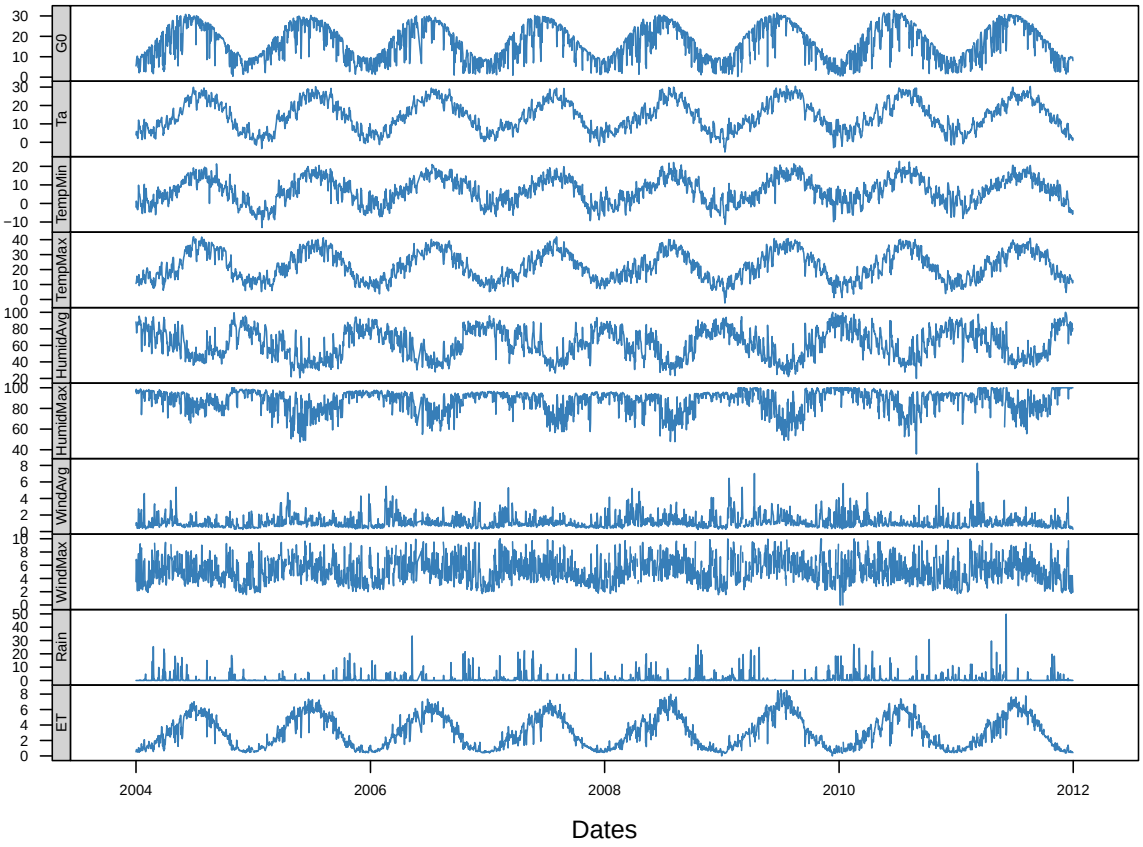
Figura 4.3: *Los datos meteorologicas se pueden leer mediante las funciones* ***readG0dm***, ***readBD***, ***dt2Meteo***, ***zoo2Meteo*** *y* ***readSIAR*** *las cuales procesan estos datos y los almacenan en un objeto de clase* **Meteo**.



- **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. Como argumentos tiene:
    - **file**: nombre del archivo que contiene los datos.
    - **lat**: latitud en grados.
    - **format**: formato de los datos de fechas.
    - **header**, **fill**, **dec**, **sep**: argumentos para **fread**.
    - **dates.col**: nombre de la columna que contiene los datos de fechas. Por defecto es 'Dates'.
    - **ta.col**: nombre de la columna que contiene los datos de temperatura. Por defecto es 'Ta'.
    - **g0.col**: nombre de la columna que contiene los datos de irradiación. Por defecto es 'G0'.
    - **keep.cols**: si su valor es **TRUE**, mantiene las columnas que no sean importantes para el resto de operaciones.

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

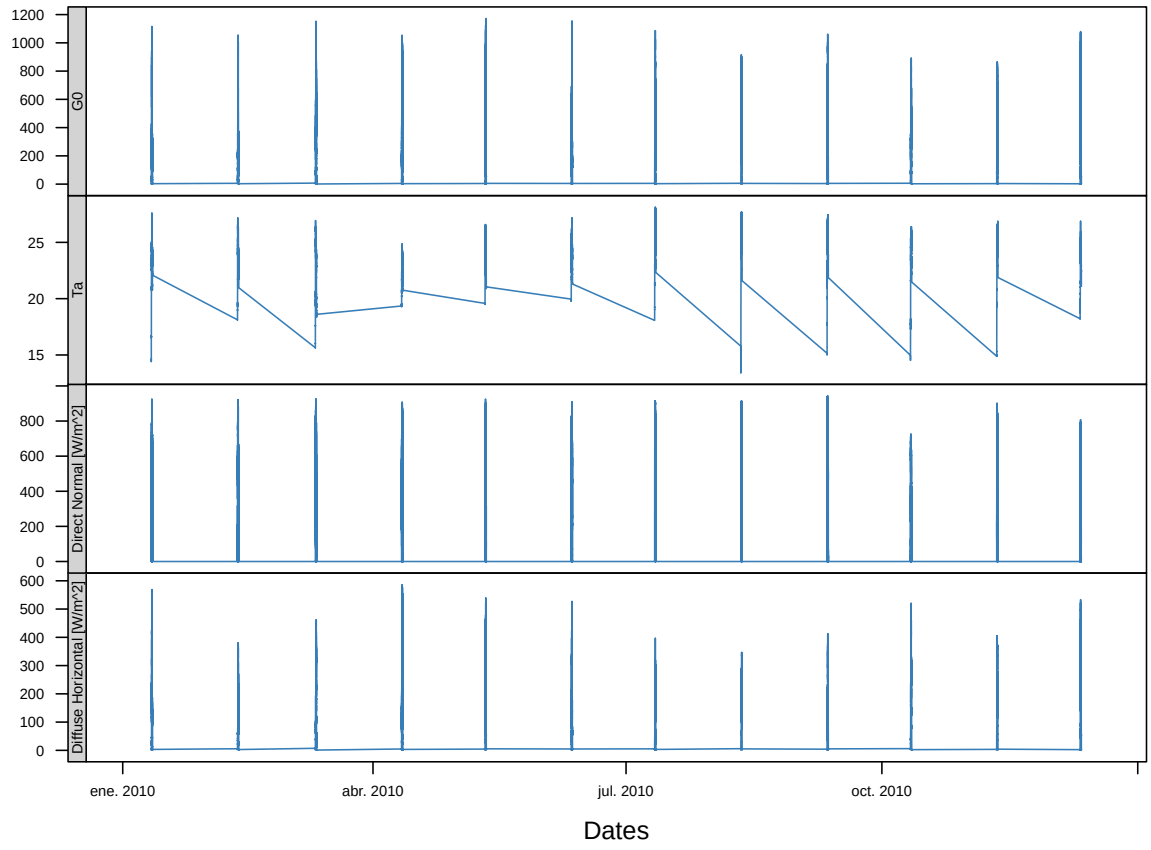
```
                ta.col = 'TempAvg', g0.col = 'Radiation', keep.cols = TRUE)
xyplot(BDd)
```



- **readBDi**: Procesa datos meteorológicos de tipo intradiarios. Como argumentos tiene:
  - **file**: nombre del archivo que contiene los datos.
  - **lat**: latitud en grados.
  - **format**: formato de los datos de fechas.
  - **header**, **fill**, **dec**, **sep**: argumentos para **fread**.
  - **dates.col**: nombre de la columna que contiene los datos de fechas y/o tiempos. Por defecto es 'Dates'.
  - **times.col**: nombre de la columna que contiene los datos de tiempos en caso de que **dates.col** no los incluyera.
  - **ta.col**: nombre de la columna que contiene los datos de temperatura. Por defecto es 'Ta'.
  - **g0.col**: nombre de la columna que contiene los datos de irradiancia. Por defecto es 'G0'.
  - **keep.cols**: si su valor es **TRUE**, mantiene las columnas que no sean importantes para el resto de operaciones.

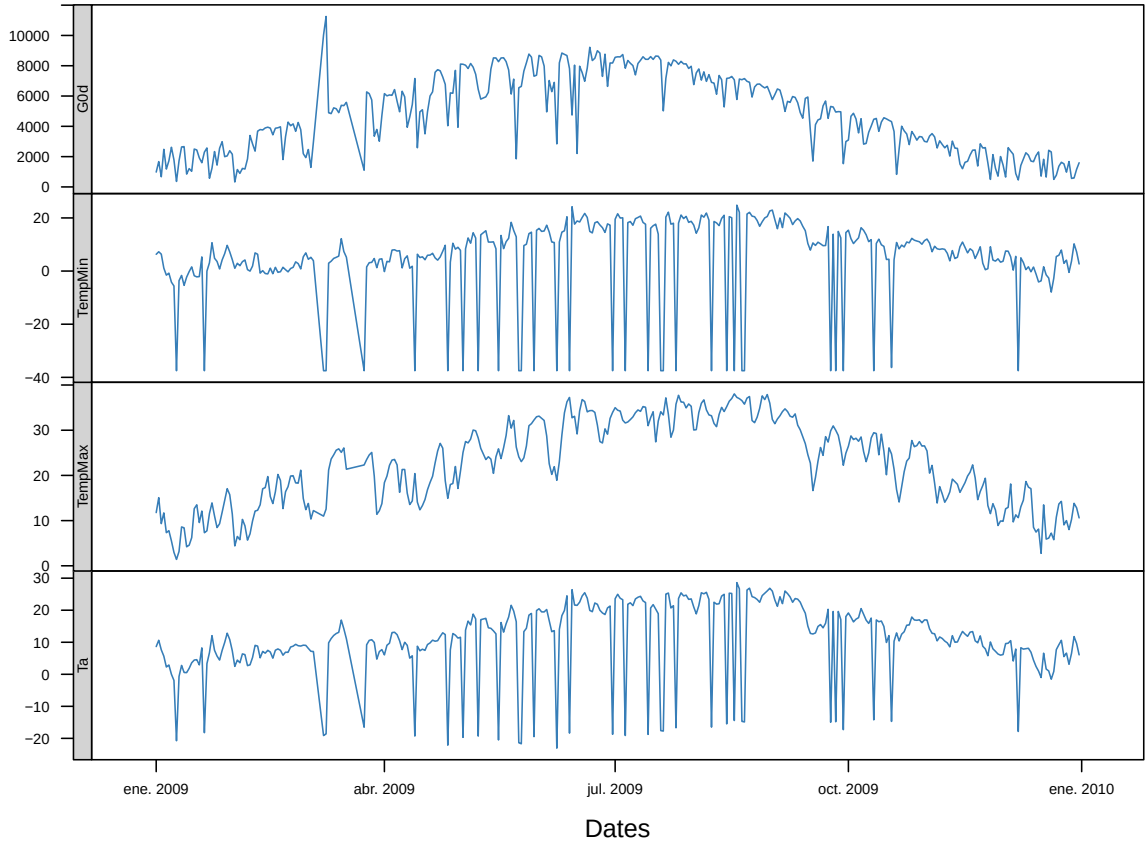
```
myURL <- "https://raw.githubusercontent.com/oscarperpinan/R/master/data/
    NREL-Hawaii.csv"
download.file(myURL, 'data/NREL-Hawaii.csv', quiet = TRUE)
BDi <- readBDi(file = 'data/NREL-Hawaii.csv', lat = 19,
```

```
                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)
xyplot(BDi)
```



- **dt2Meteo**: Transforma un **data.table** o **data.frame** en un objeto de clase **Meteo**. Como argumentos tiene:
  - **file**: **data.table** que contiene los datos.
  - **lat**: latitud en grados.
  - **source**: información sobre la fuente de los datos.
  - **type**: tipo de datos. A elegir entre **bdI** (intradiarios), **bd** (diarios) y **prom** (medias mensuales). Si no viene dado, lo calcula por su cuenta.

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



- **zoo2Meteo**: Transforma un objeto de clase **zoo**[4] en un objeto de clase **Meteo**. Como argumentos tiene:
  - **file**: **data.table** que contiene los datos.
  - **lat**: latitud en grados.
  - **source**: información sobre la fuente de los datos.

```
1 library(zoo)
2 bd_zoo <- read.csv.zoo('data/aranjuez.csv')
3 BD_zoo <- zoo2Meteo(file = bd_zoo, lat = 40)
4 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
```

[4]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.

```
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
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[5] y transformarlo en un objeto de clase **Meteo**. Como argumentos tiene:
  - **Lon**: Longitud en grados.
  - **Lat**: Latitud en grados.
  - **inicio**: Primer día de los registros.
  - **final**: Último día de los registros.
  - **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[7]) 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.

```r
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)
xyplot(bd_SIAR)
```

[5]La red SiAR (Sistema de Información Agroclimática para el Regadio) es una infraestructura que captura, registra y divulga los datos climáticos necesarios para el cálculo de la demanda hídrica en las zonas de riego [Min23].

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

[7]La interpolación IDW es un método de interpolación que estima el valor de un punto desconocido basado en los valores conocidos de puntos cercanos. Los puntos más cercanos tienen más peso en la estimación que los más lejanos, utilizando una relación inversa con la distancia.

HTTP 403 Forbidden.

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, **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. Tiene los siguientes argumentos:

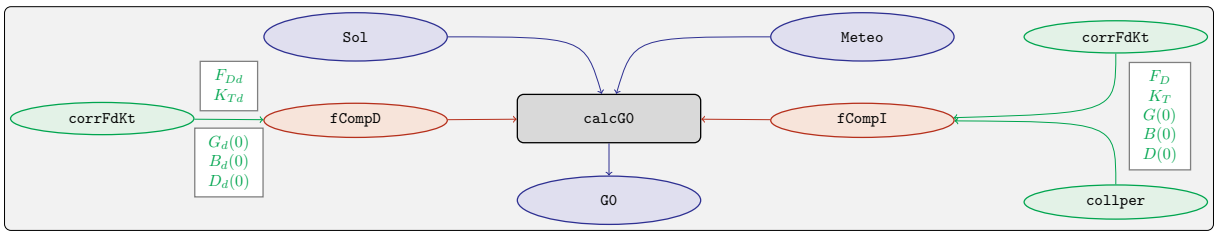
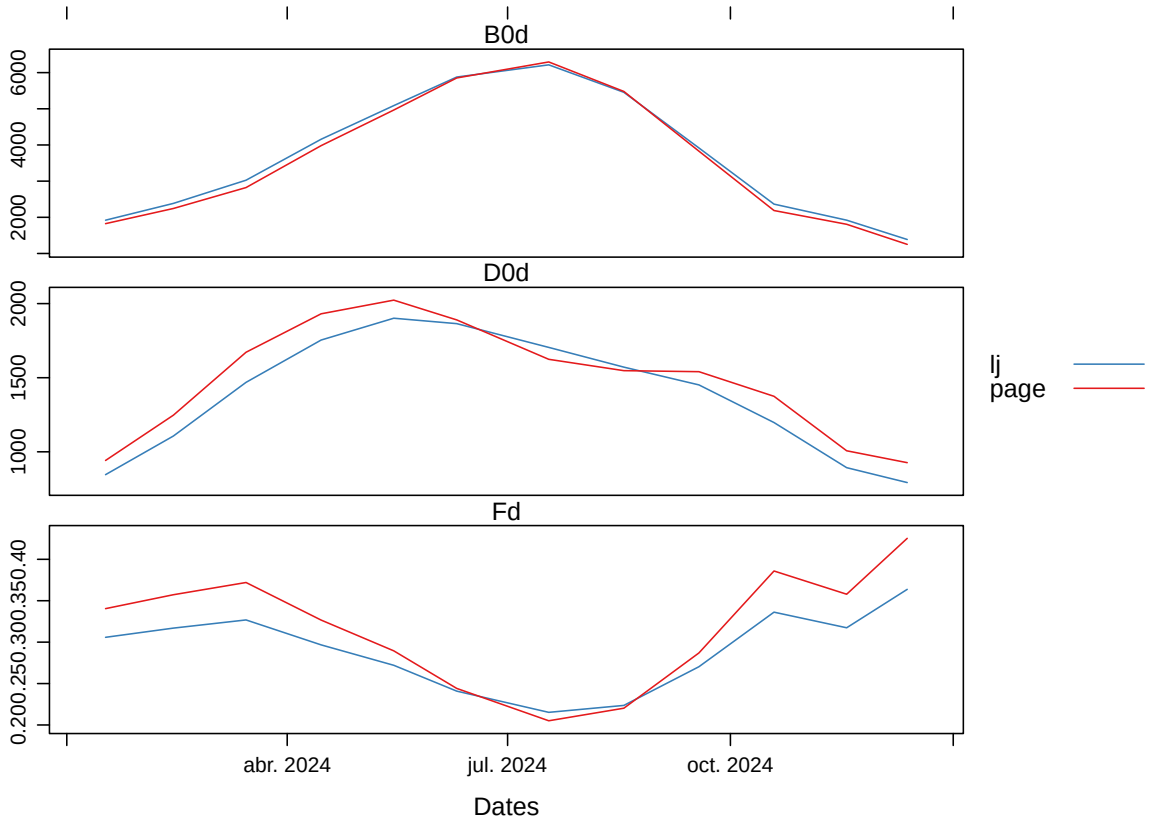


Figura 4.4: *Cálculo de la radiación incidente en el plano horizontal mediante la función* ***calcG0****, la cual procesa un objeto clase* ***Sol*** *y otro clase* ***Meteo*** *mediante las funciones* ***fCompD*** *y* ***fCompI*** *resultando en un objeto clase* ***G0****. :*

- **sol**: Un objeto de clase **Sol**.
- **G0d**: Un objeto clase **Meteo** o un **data.table** con datos de irradiación diaria en una superficie horizontal.
- **corr**: A elegir el tipo de correlación entre la fracción de difusa y el índice de claridad. Dependiendo del tipo de datos:
  - Mensuales:

```
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[, group := 'page']
compD_lj <- fCompD(sol = solD, G0d = G0d, corr = "LJ")
compD_lj[, group := 'lj']
compD_dia <- rbind(compD_page, compD_lj)
xyplot(Fd + D0d + B0d ~ Dates, compD_dia,
       groups = group, type = 'l', auto.key = TRUE,
       par.settings = solaR.theme, scales = list(y = 'free'),
       ylab = '', layout = c(1, 3))

```



  - 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[, group := 'cpr']
compD_ekdd <- fCompD(sol = sol, G0d = G0d, corr = 'EKDd')
compD_ekdd[, group := 'ekdd']
compD_climedd <- fCompD(sol = sol, G0d = G0d, corr = 'CLIMEDd')
compD_climedd[, group := 'climedd']
compD_mes <- rbind(compD_cpr, compD_ekdd, compD_climedd)
xyplot(Fd + D0d + B0d ~ Dates, compD_mes,
       groups = group, type = 'l',auto.key = TRUE,
       par.settings = solaR.theme, scales = list(y = 'free'),
       ylab = '', layout = c(1, 3))
```

También, se puede aportar una función de correlación propia con el argumento **f**.

```r
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)
```

```
10 xyplot(compD_user)
```

argument specifying columns received non−existing column(s): cols[2]='G0'

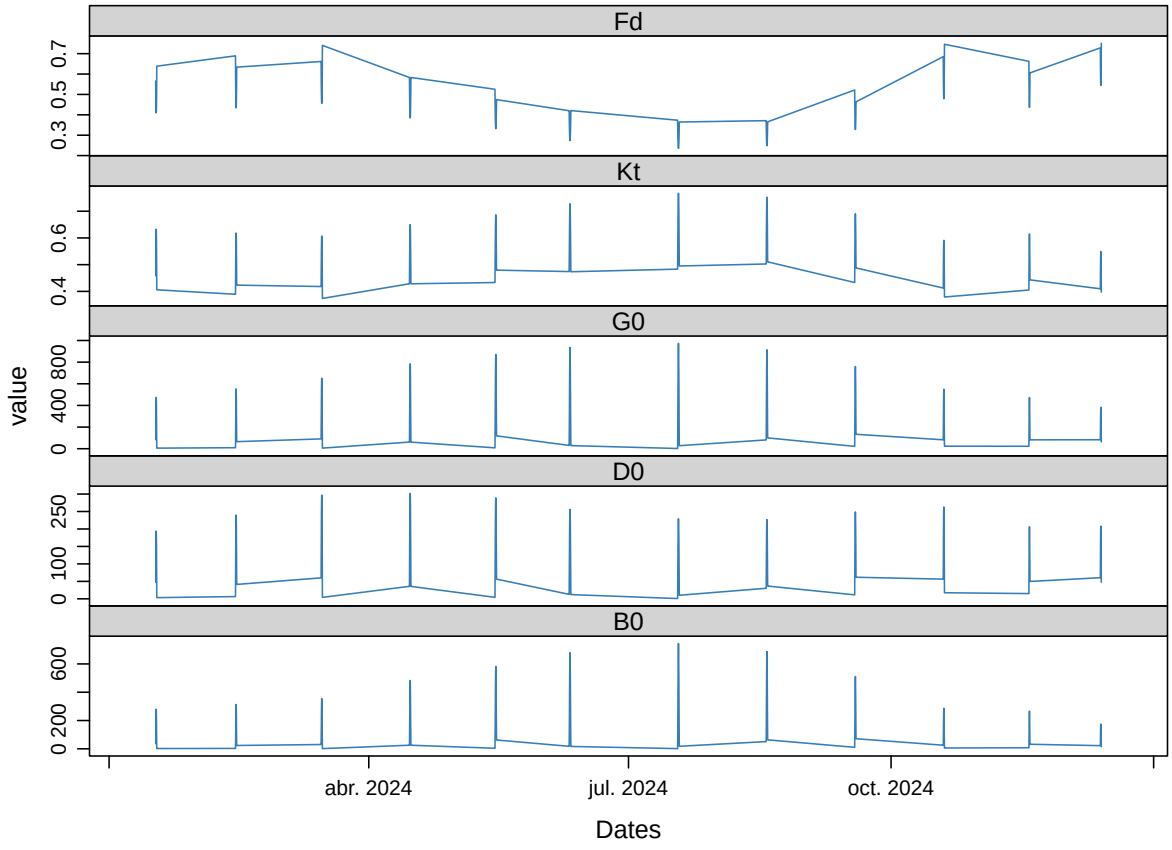
Por último, si **G0d** ya contiene todos los componentes, se puede especifica que no haga ninguna correlación.

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

```
Error en fCompD(sol = sol, G0d = compD_user, corr = "none"):
  indexD(sol) == indexD(G0d) are not all TRUE
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)
xyplot(compI)
```



  - **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')
compI_ekdh[, group := 'ekdh']
compI_brl <- fCompI(sol = sol, G0I = G0I, corr = 'BRL')
compI_brl[, group := 'brl']
compI_climedh <- fCompI(sol = sol, G0I = G0I, corr = 'CLIMEDh')
compI_climedh[, group := 'climedh']
```

```
compI_all <- rbind(compI_ekdh, compI_ekdh, compI_climedh)
xyplot(Fd+ D0 +B0 ~ Dates, compI_all,
       groups= group, type = 'l',auto.key= TRUE,
       par.settings= solaR.theme,scales =list(y = 'free'),
       ylab = '', layout= c(1,3))
```

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 1.1180668 1.647933
```

```
 2: Feb. 2024 3.491 1.5046009 1.986399
 3: Mar. 2024 4.494 2.0762128 2.417787
 4: Apr. 2024 5.912 2.3080442 3.603956
 5: May. 2024 6.989 2.2254279 4.763572
 6: Jun. 2024 7.742 2.5449492 5.197051
 7: Jul. 2024 7.919 2.3044799 5.614520
 8: Aug. 2024 7.027 2.0284400 4.998560
 9: Sep. 2024 5.369 1.7184590 3.650541
10: Oct. 2024 3.562 1.5762501 1.985750
11: Nov. 2024 2.814 1.3162794 1.497721
12: Dec. 2024 2.179 0.9316858 1.247314

Yearly values:
Key: <Dates>
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 660.3428 1179.022
```

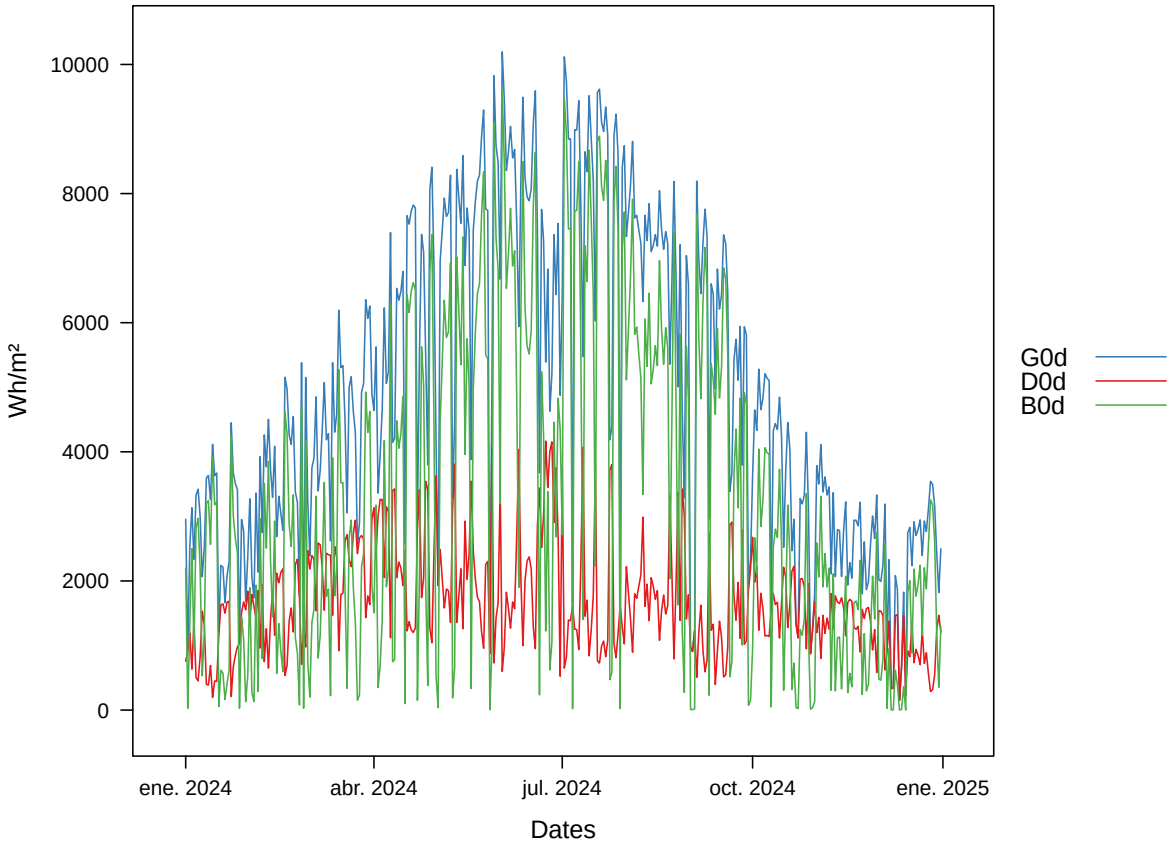
- Diarios.

```
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 1.1180668 1.647933
 2: Feb. 2024 3.491 1.5046009 1.986399
 3: Mar. 2024 4.494 2.0762128 2.417787
 4: Apr. 2024 5.912 2.3080442 3.603956
 5: May. 2024 6.989 2.2254279 4.763572
 6: Jun. 2024 7.742 2.5449492 5.197051
 7: Jul. 2024 7.919 2.3044799 5.614520
 8: Aug. 2024 7.027 2.0284400 4.998560
 9: Sep. 2024 5.369 1.7184590 3.650541
10: Oct. 2024 3.562 1.5762501 1.985750
11: Nov. 2024 2.814 1.3162794 1.497721
12: Dec. 2024 2.179 0.9316858 1.247314

Yearly values:
Key: <Dates>
   Dates      G0d      D0d      B0d
   <int>    <num>    <num>    <num>
1:  2024 1839.365 660.3428 1179.022
```

- Intradiarios

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



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.

```
BTd <- fBTd(mode = 'serie')[1:10]
sol <- calcSol(lat, BTd = BTd, keep.night = FALSE)
beta <- lat - 10
alpha <- 0
angGen_fixed <- fTheta(sol = sol, beta = beta, alpha = alpha,
                       modeTrk = 'fixed')
xyplot(angGen_fixed)
```

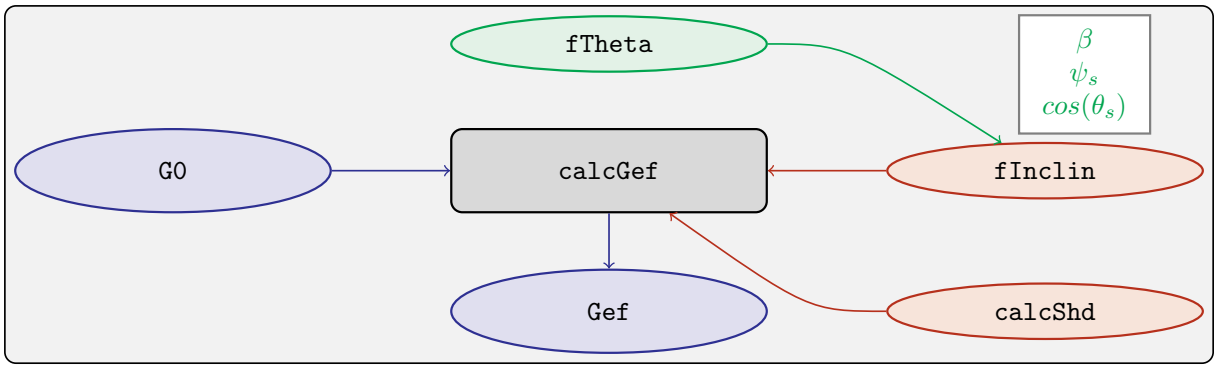
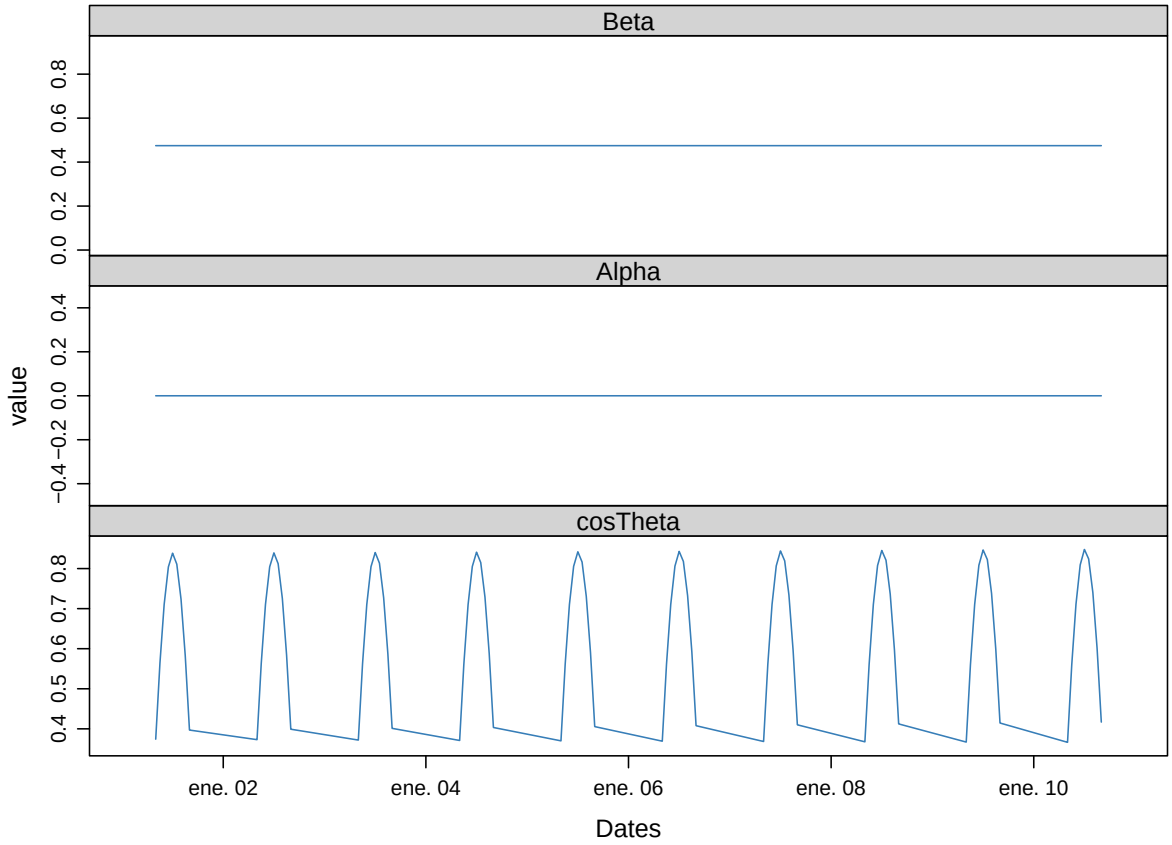
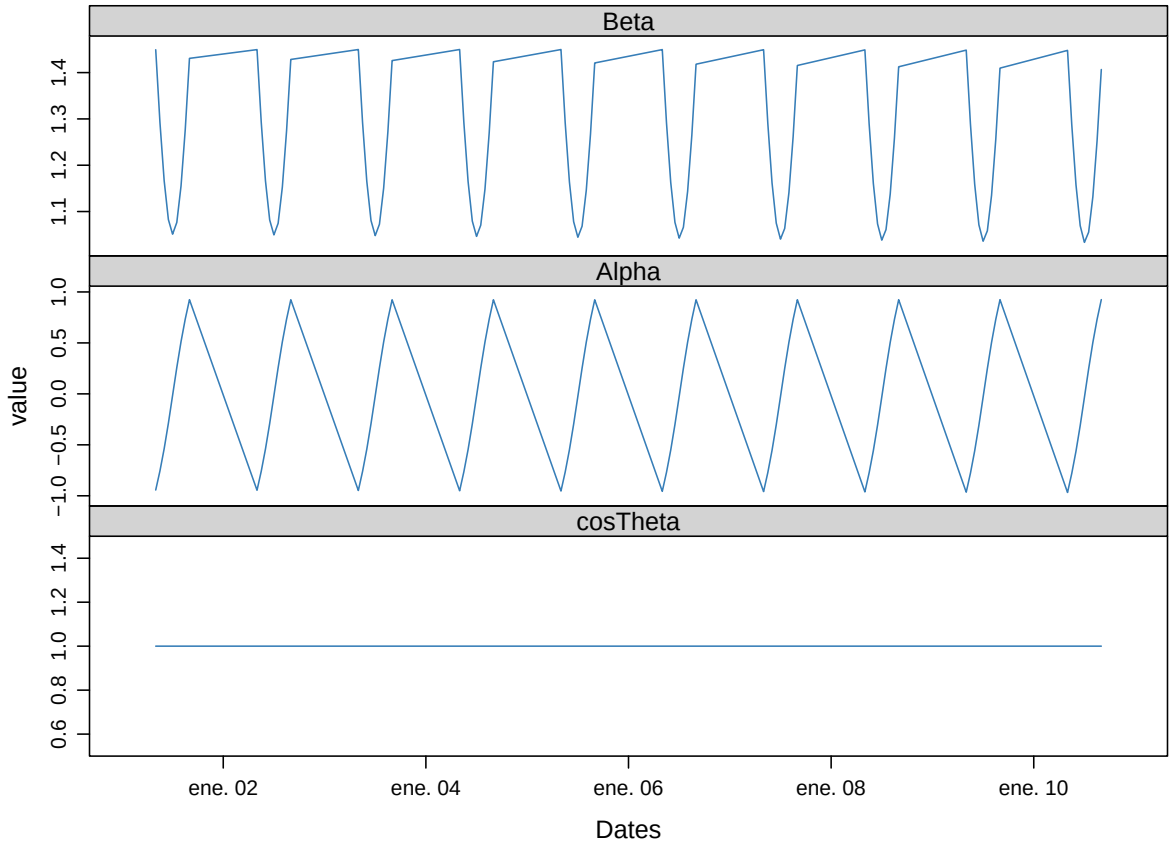


FIGURA 4.5: *Cálculo de la radiación efectiva incidente en el plano del generador mediante la función* **`calcGef`**, *la cual emplea la función* **`fInclin`** *para el computo de las componentes efectivas, la función* **`fTheta`** *que provee a la función anterior los ángulos necesarios para su computo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



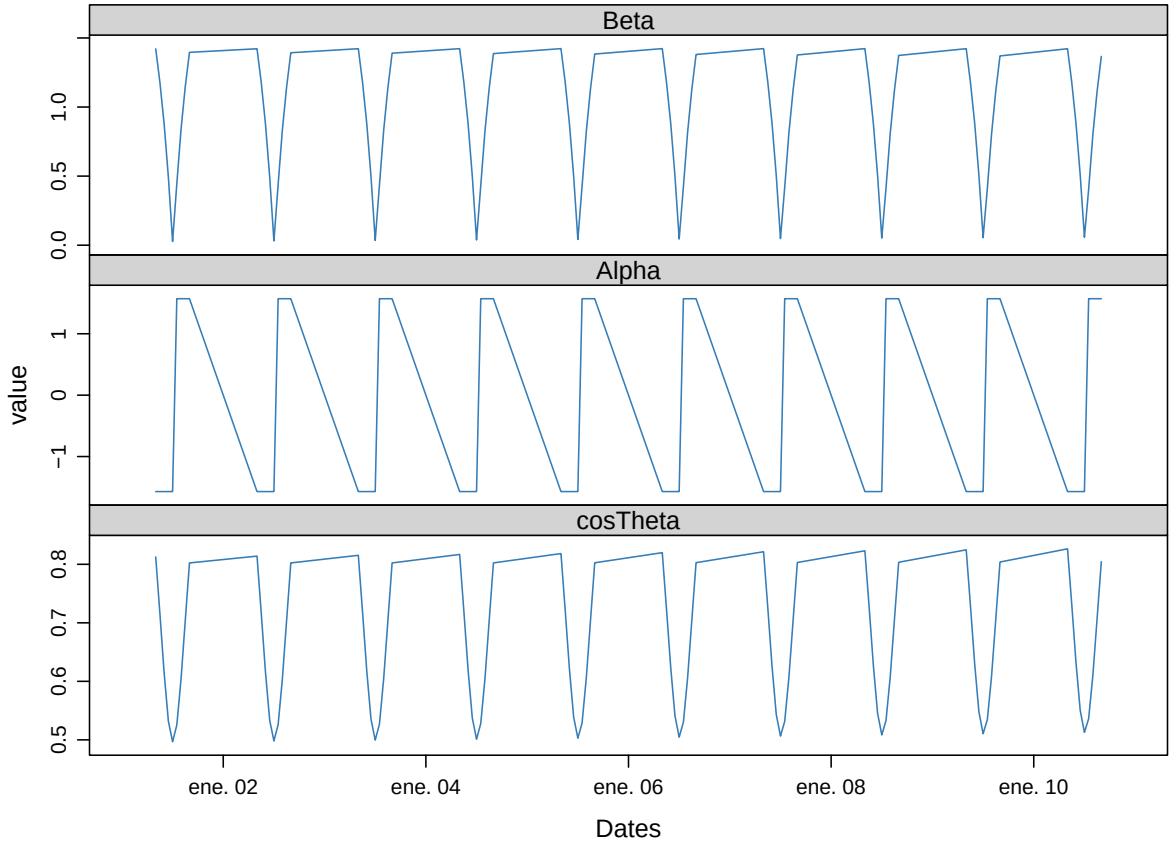
- **`two`**: para sistemas de seguimiento de doble eje.

```
angGen_two <- fTheta(sol = sol, beta = beta, alpha = alpha,
                     modeTrk = 'two')
xyplot(angGen_two)
```



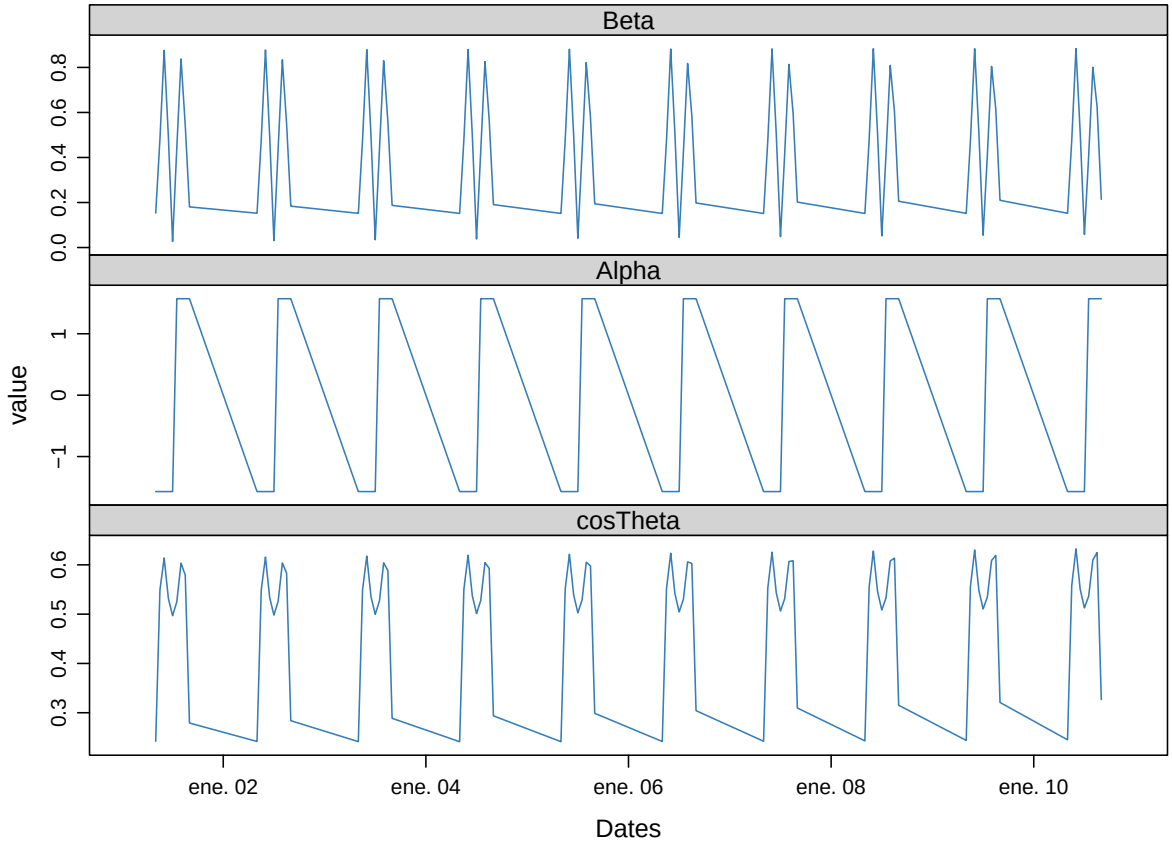
- **horiz**: para sistemas de seguimiento horizontal Norte-Sur.

```
angGen_horiz <- fTheta(sol = sol, beta = beta, alpha = alpha,
                       modeTrk = 'horiz')
xyplot(angGen_horiz)
```



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.

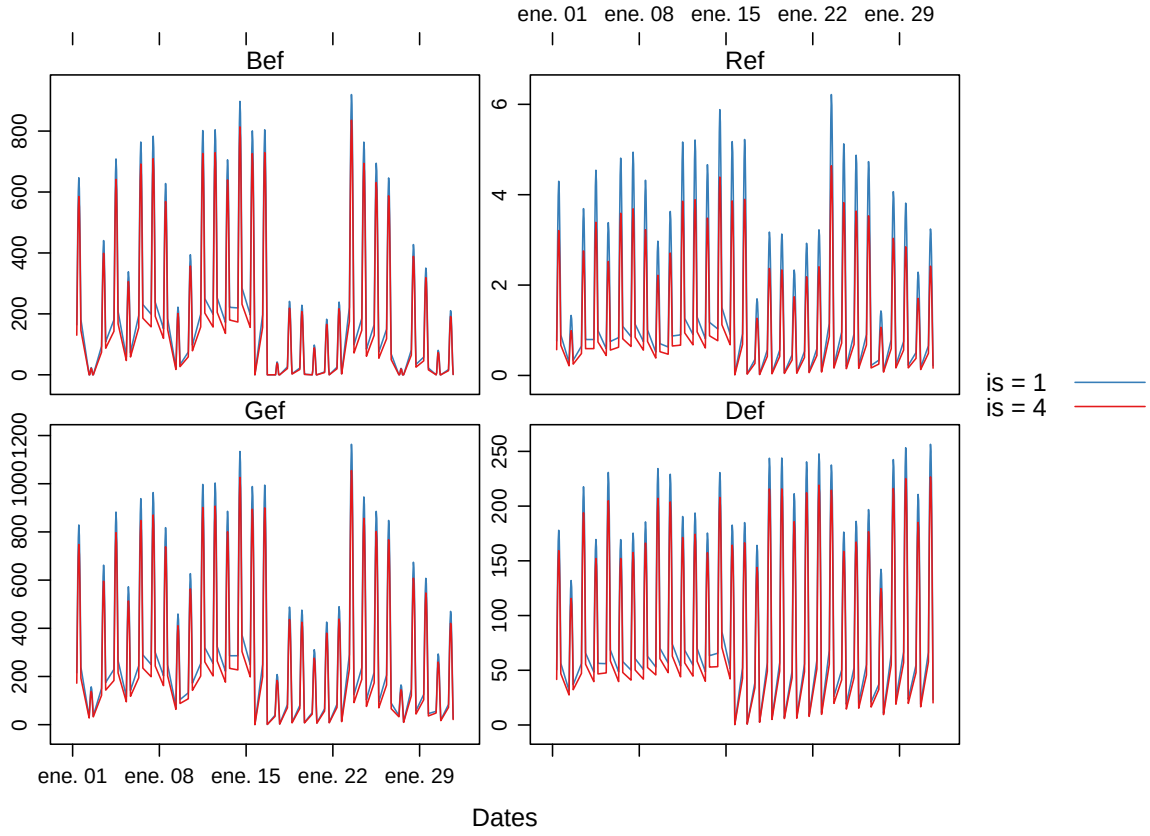
```
struct <- list(L = 1)
distances <- data.table(Lew = 2)
angGen_BT <- fTheta(sol = sol, beta = beta, alpha = alpha,
                    modeTrk = 'horiz', BT = TRUE,
                    struct = struct, dist = distances)
xyplot(angGen_BT)
```



- **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:

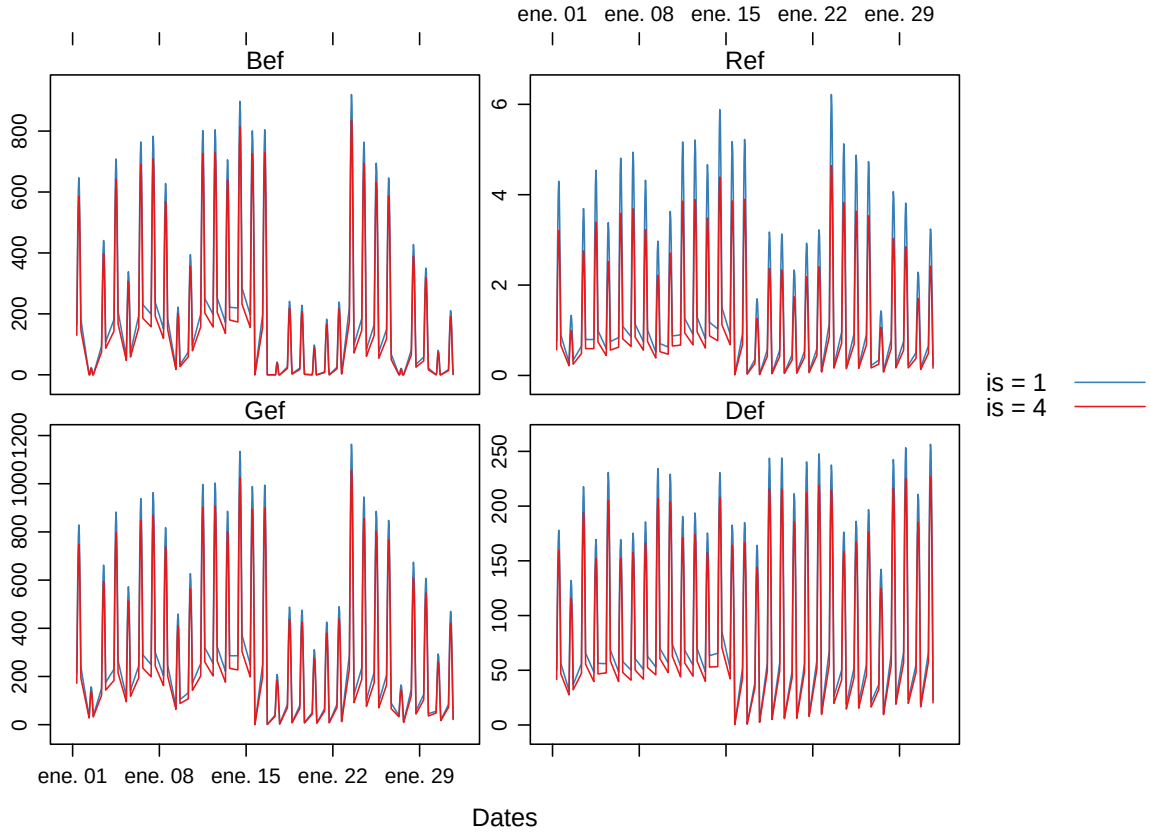
  - **iS**: permite seleccionar entre 4 valores del 1 al 4 correspondientes al grado de suciedad del módulo. Siendo 1 limpio y 4 alto y basandose en los valores de la tabla 3.2 calcula la irradiancia efectiva. Por defecto tiene valor 2 (grado de suciedad bajo).

```
compI <- calcG0(lat, modeRad = 'bd', dataRad = bd[1:31], keep.night = FALSE
    )
sol <- calcSol(lat, BTi = indexI(compI))
angGen <- fTheta(sol = sol, beta = beta, alpha = alpha)
inclin_limpio <- fInclin(compI = compI, angGen = angGen, iS = 1)
inclin_limpio[, group := 'is = 1']
inclin_sucio <- fInclin(compI = compI, angGen = angGen, iS = 4)
inclin_sucio[, group := 'is = 4']
inclin_is <- rbind(inclin_limpio, inclin_sucio)
xyplot(Gef + Def + Bef + Ref ~ Dates, inclin_is,
       groups = group, type = 'l', auto.key = TRUE,
       par.settings = solaR.theme, scales = list(y = 'free'),
       ylab = '', layout = c(2, 2))
```



- **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).

```
inclin_alb0 <- fInclin(compI = compI, angGen = angGen, alb = 0)
inclin_alb0[, group := 'alb = 0']
inclin_alb1 <- fInclin(compI = compI, angGen = angGen, alb = 1)
inclin_alb1[, group := 'alb = 1']
inclin_alb <- rbind(inclin_alb0, inclin_alb1)
xyplot(Gef + Def + Bef + Ref ~ Dates, inclin_is,
       groups = group, type = 'l', auto.key = TRUE,
       par.settings = solaR.theme, scales = list(y = 'free'),
       ylab = '', layout = c(2, 2))
```



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**[8] (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)
summary(inclin_horizBright)
```

```
    Dates                          Bo               Bn               G
Min.   :2024-01-01 08:00:00.00   Min.   :   0.0   Min.   :   0.0   Min.   :   0.4769
1st Qu.:2024-01-09 09:45:00.00   1st Qu.: 642.2   1st Qu.: 157.8   1st Qu.: 212.7325
Median :2024-01-17 09:30:00.00   Median :1005.1   Median : 457.2   Median : 413.8913
Mean   :2024-01-16 21:56:08.92   Mean   : 898.5   Mean   : 451.6   Mean   : 449.1240
3rd Qu.:2024-01-24 13:15:00.00   3rd Qu.:1152.8   3rd Qu.: 744.3   3rd Qu.: 664.7594
Max.   :2024-01-31 17:00:00.00   Max.   :1248.5   Max.   :1061.8   Max.   :1170.9366
      D                Di               Dc               B               R
Min.   :  0.4519   Min.   :  0.4519   Min.   :  0.00   Min.   :  0.00   Min.   :0.02496
1st Qu.: 87.4382   1st Qu.: 38.5447   1st Qu.: 22.91   1st Qu.: 82.65   1st Qu.:1.42466
Median :144.6206   Median : 52.3344   Median : 58.94   Median :230.00   Median :3.05507
Mean   :140.3699   Mean   : 76.3276   Mean   : 64.04   Mean   :305.55   Mean   :3.20341
3rd Qu.:187.2631   3rd Qu.:120.0449   3rd Qu.:102.29   3rd Qu.:496.65   3rd Qu.:4.66460
Max.   :265.2378   Max.   :199.3722   Max.   :200.37   Max.   :922.89   Max.   :8.29293
     FTb                FTd               FTr              Dief              Dcef
Min.   :0.005218   Min.   :0.06474   Min.   :0.2995   Min.   :  0.4142   Min.   :  0.00
1st Qu.:0.010325   1st Qu.:0.06474   1st Qu.:0.2995   1st Qu.: 35.3285   1st Qu.: 20.09
```

[8] 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.

```
Median :0.022076   Median :0.06474   Median :0.2995   Median : 47.9676   Median : 56.15
Mean   :0.062300   Mean   :0.06474   Mean   :0.2995   Mean   : 69.9587   Mean   : 60.85
3rd Qu.:0.096613   3rd Qu.:0.06474   3rd Qu.:0.2995   3rd Qu.:110.0282   3rd Qu.: 98.82
Max.   :0.344085   Max.   :0.06474   Max.   :0.2995   Max.   :182.7363   Max.   :195.14
     Gef                Def               Bef              Ref
Min.   :   0.4314   Min.   :  0.4142   Min.   :  0.00   Min.   :0.01713
1st Qu.: 192.2193   1st Qu.: 79.6995   1st Qu.: 73.57   1st Qu.:0.97804
Median : 385.0118   Median :134.6125   Median :213.50   Median :2.09733
Mean   : 423.1706   Mean   :130.8085   Mean   :290.16   Mean   :2.19917
3rd Qu.: 635.3122   3rd Qu.:177.6651   3rd Qu.:476.41   3rd Qu.:3.20229
Max.   :1135.6920   Max.   :246.9478   Max.   :898.77   Max.   :5.69317
```

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

```
    Dates                          Bo              Bn              G
Min.   :2024-01-01 08:00:00.00   Min.   :   0.0   Min.   :   0.0   Min.   :   0.4817
1st Qu.:2024-01-09 09:45:00.00   1st Qu.: 642.2   1st Qu.: 157.8   1st Qu.: 213.4170
Median :2024-01-17 09:30:00.00   Median :1005.1   Median : 457.2   Median : 415.0283
Mean   :2024-01-16 21:56:08.92   Mean   : 898.5   Mean   : 451.6   Mean   : 449.7021
3rd Qu.:2024-01-24 13:15:00.00   3rd Qu.:1152.8   3rd Qu.: 744.3   3rd Qu.: 665.3801
Max.   :2024-01-31 17:00:00.00   Max.   :1248.5   Max.   :1061.8   Max.   :1171.3877
      D                Di               Dc              B               R
Min.   :  0.4567   Min.   :  0.4567   Min.   :  0.00   Min.   :  0.00   Min.   :0.02496
1st Qu.: 87.8068   1st Qu.: 38.8853   1st Qu.: 22.91   1st Qu.: 82.65   1st Qu.:1.42466
Median :145.2683   Median : 52.6465   Median : 58.94   Median :230.00   Median :3.05507
Mean   :140.9480   Mean   : 76.9057   Mean   : 64.04   Mean   :305.55   Mean   :3.20341
3rd Qu.:188.0437   3rd Qu.:121.1258   3rd Qu.:102.29   3rd Qu.:496.65   3rd Qu.:4.66460
Max.   :266.7754   Max.   :200.9098   Max.   :200.37   Max.   :922.89   Max.   :8.29293
     FTb               FTd               FTr              Dief        Dcef        Gef
Min.   :0.005218   Min.   :0.06474   Min.   :0.2995   Min.   :0   Min.   :0   Min.   :  0.00
1st Qu.:0.010325   1st Qu.:0.06474   1st Qu.:0.2995   1st Qu.:0   1st Qu.:0   1st Qu.: 73.57
Median :0.022076   Median :0.06474   Median :0.2995   Median :0   Median :0   Median :213.50
Mean   :0.062300   Mean   :0.06474   Mean   :0.2995   Mean   :0   Mean   :0   Mean   :290.16
3rd Qu.:0.096613   3rd Qu.:0.06474   3rd Qu.:0.2995   3rd Qu.:0   3rd Qu.:0   3rd Qu.:476.41
Max.   :0.344085   Max.   :0.06474   Max.   :0.2995   Max.   :0   Max.   :0   Max.   :898.77
     Def         Bef              Ref
Min.   :0   Min.   :  0.00   Min.   :0
1st Qu.:0   1st Qu.: 73.57   1st Qu.:0
Median :0   Median :213.50   Median :0
Mean   :0   Mean   :290.16   Mean   :0
3rd Qu.:0   3rd Qu.:476.41   3rd Qu.:0
Max.   :0   Max.   :898.77   Max.   :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**.

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

```
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
```

```
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.

```
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.

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

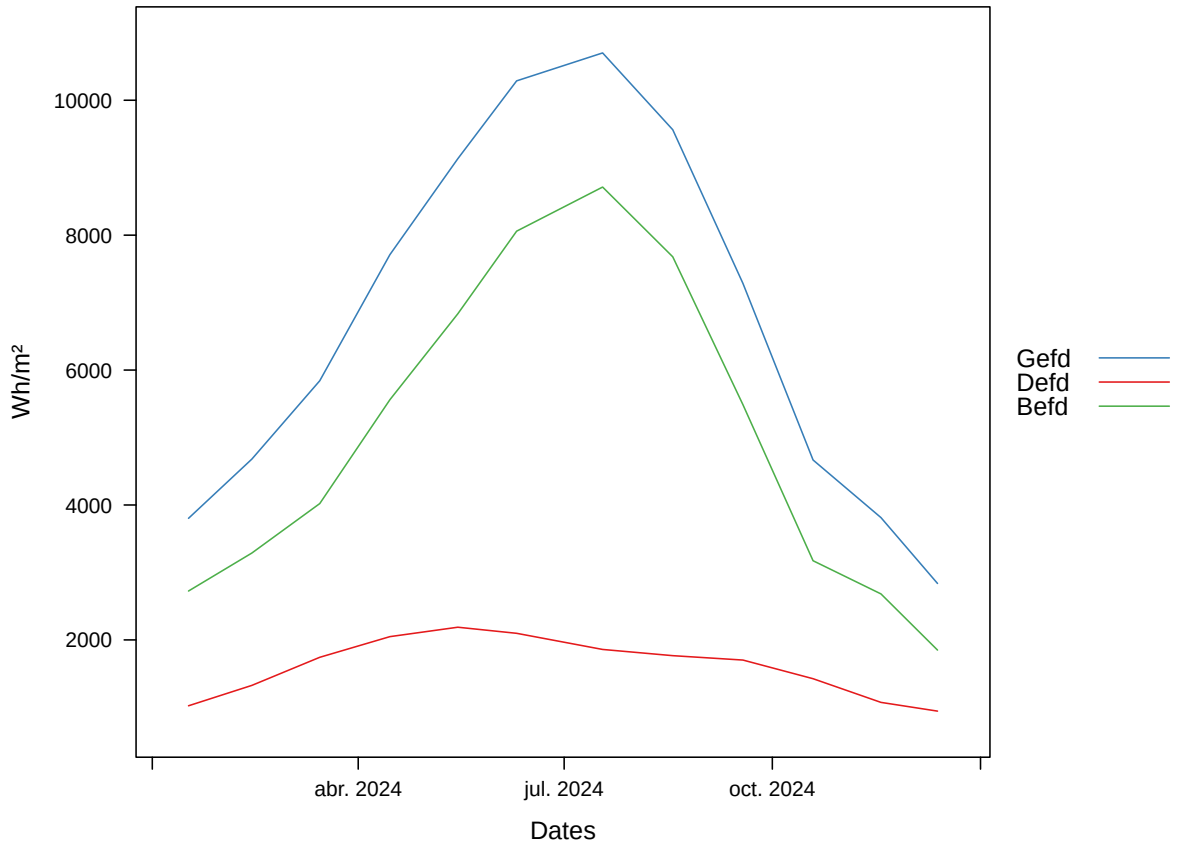
```
Object of class  Gef

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)
xyplot(gef_shdbt)
```



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:

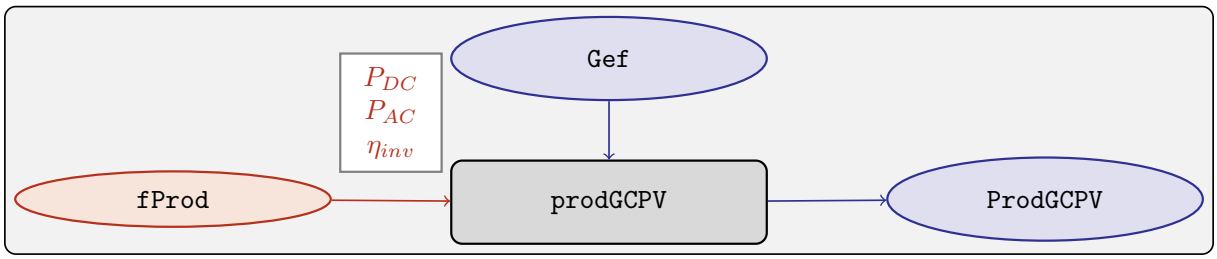


Figura 4.6: *Estimación de la producción eléctrica de un SFCR mediante la función* **`prodGCPV`**, *la cual emplea la función* **`fProd`** *para el computo de la potencia a la entrada (P_{DC}), a la salida (P_{AC}) y el rendimiento (η_{inv}) del inversor.*

- `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.
  - `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.

```
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
```

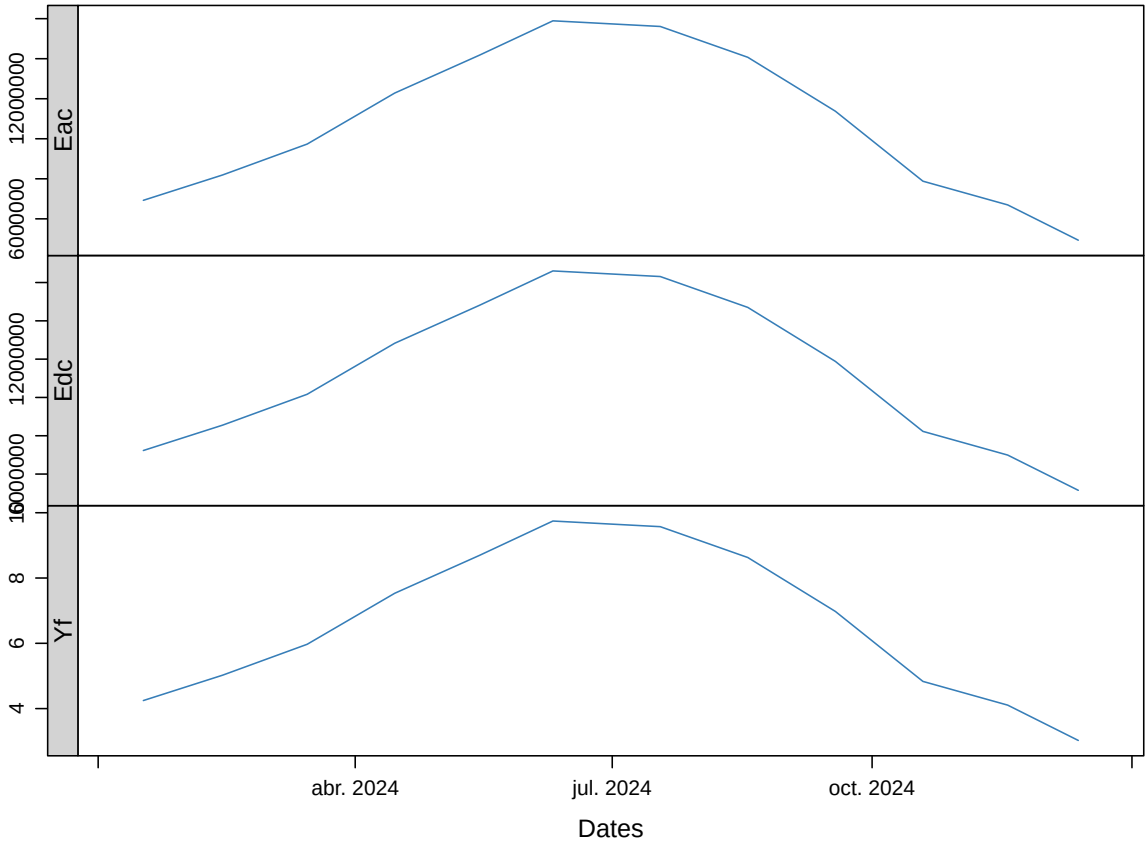
```
Monthly avarages:
        Dates       Eac       Edc       Yf
       <char>     <num>     <num>    <num>
 1: Jan. 2024  6678.640  6978.809 4.095437
 2: Feb. 2024  7194.835  7519.763 4.411976
 3: Mar. 2024  7843.267  8195.794 4.809603
 4: Apr. 2024  8929.962  9336.190 5.475980
 5: May. 2024  9484.132  9914.605 5.815805
 6: Jun. 2024  9862.108 10309.878 6.047585
 7: Jul. 2024 10019.327 10475.108 6.143994
 8: Aug. 2024  9703.467 10142.955 5.950304
 9: Sep. 2024  8757.170  9151.202 5.370022
10: Oct. 2024  6908.624  7220.274 4.236467
11: Nov. 2024  6431.264  6720.101 3.943743
12: Dec. 2024  5229.190  5463.318 3.206614

Yearly values:
   Dates     Eac     Edc       Yf
   <int>   <num>   <num>    <num>
1:  2024 2959931 3093711 1815.072
-----------------
Mode of tracking:  fixed
    Inclination:  27.2
    Orientation:  0
-----------------
Generator:
    Modules in series:  22
    Modules in parallel:  130
    Nominal power (kWp):  1630.8
```

```r
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  9725.125 10159.052 5.963585
 2: Feb. 2024 10163.306 10616.084 6.232284
 3: Mar. 2024 10793.363 11273.933 6.618644
 4: Apr. 2024 12779.088 13349.623 7.836319
 5: May. 2024 14473.608 15121.428 8.875422
 6: Jun. 2024 16252.186 16981.660 9.966072
 7: Jul. 2024 15916.049 16632.343 9.759948
 8: Aug. 2024 14511.287 15163.412 8.898528
 9: Sep. 2024 12350.603 12903.142 7.573565
10: Oct. 2024  9458.655  9878.812 5.800182
11: Nov. 2024  9177.823  9586.046 5.627972
12: Dec. 2024  7284.466  7606.879 4.466938

Yearly values:
   Dates     Eac     Edc       Yf
   <int>   <num>   <num>    <num>
1:  2024 4358566 4553392 2672.735
-----------------
Mode of tracking:  two
    Inclination limit: 90
-----------------
Generator:
    Modules in series:  22
```

```
Modules in parallel:  130
Nominal power (kWp):  1630.8
```

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



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**:

```
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
```

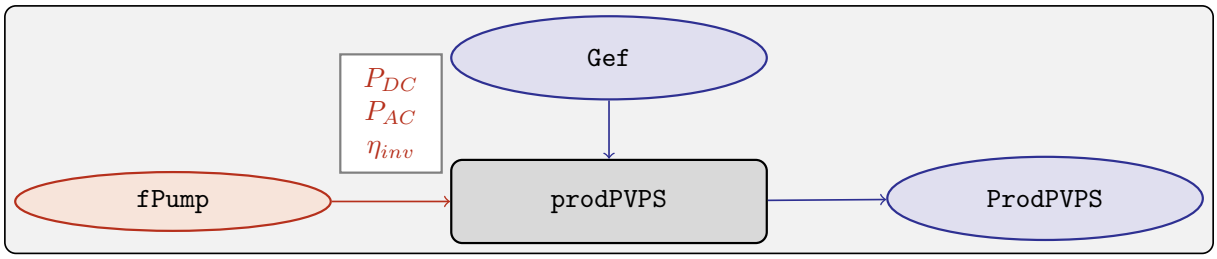


FIGURA 4.7: *Estimación de la producción eléctrica de un SFB mediante la función* ***prodPVPS****, la cual emplea la función* ***fPump*** *para el computo del rendimiento de las diferentes parte de una bomba centrífuga alimentada por un convertidor de frecuencia.*

- **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: 0x0000016a6d7242e8>
<environment: 0x0000016a6cc7c8c8>
```

- **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: 0x0000016a6d7242e8>
<environment: 0x0000016a6cc6cc10>
```

- **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: 0x0000016a6d7242e8>
<environment: 0x0000016a6cc54660>
```

- **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: 0x0000016a6d7242e8>
<environment: 0x0000016a6cc746d0>
```

Se pueden realizar operaciones con este objeto:

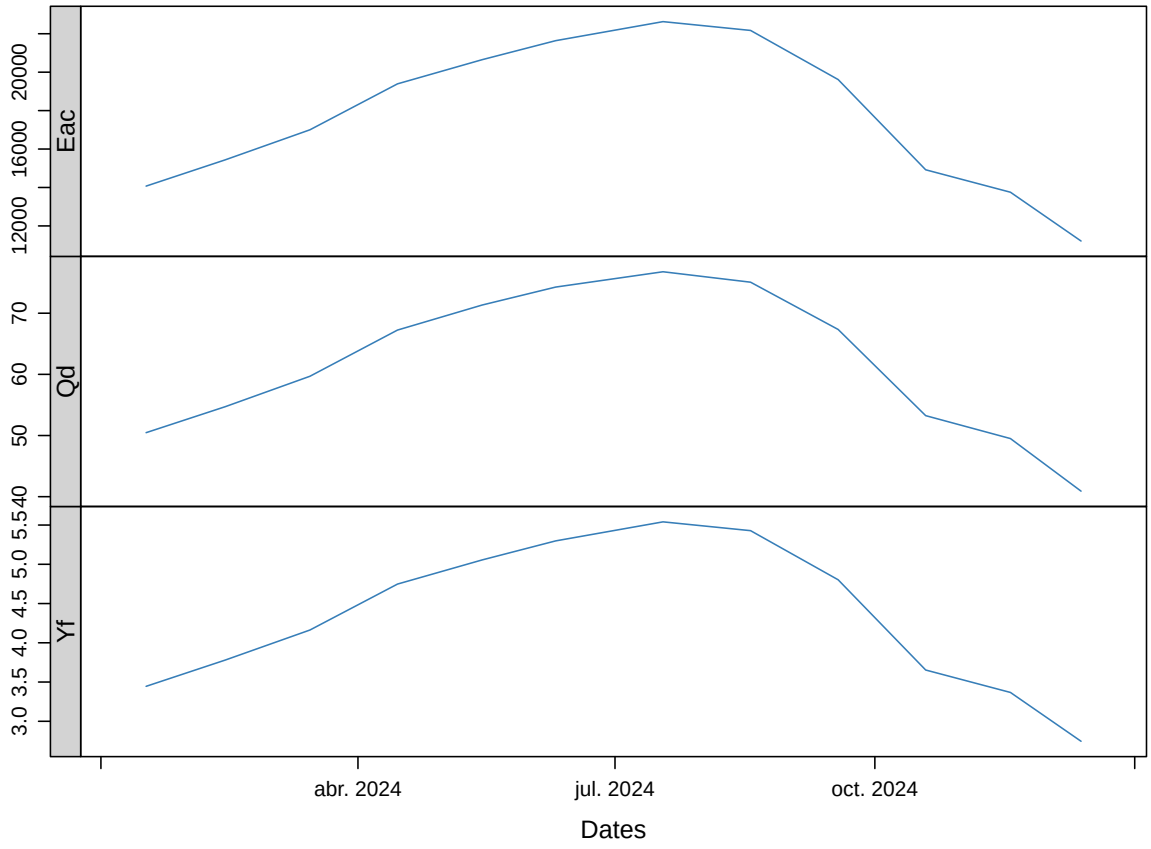
```
SP8A44 = with(fSP8A44,{
  Pac = seq(lim[1],lim[2],l=10)
```

```
  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**.

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

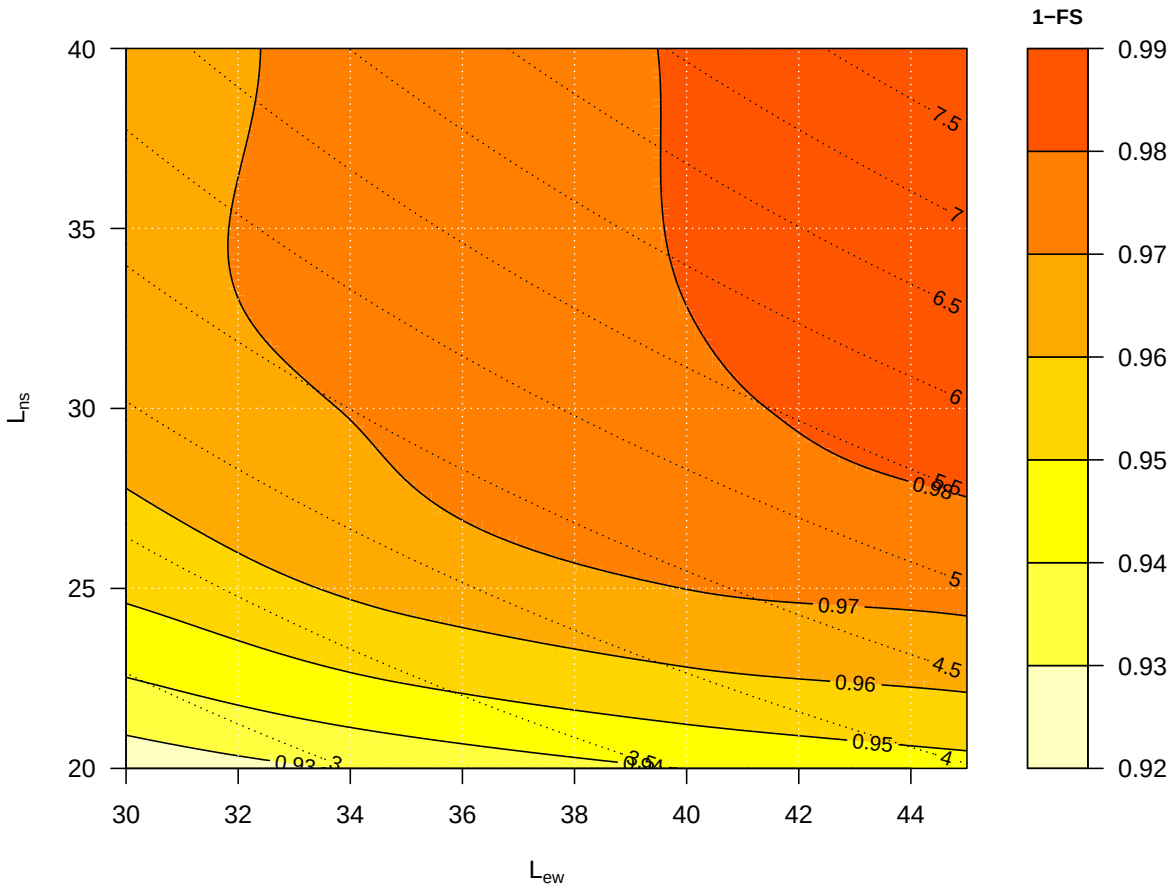


4.7. Optimización de distancias

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.

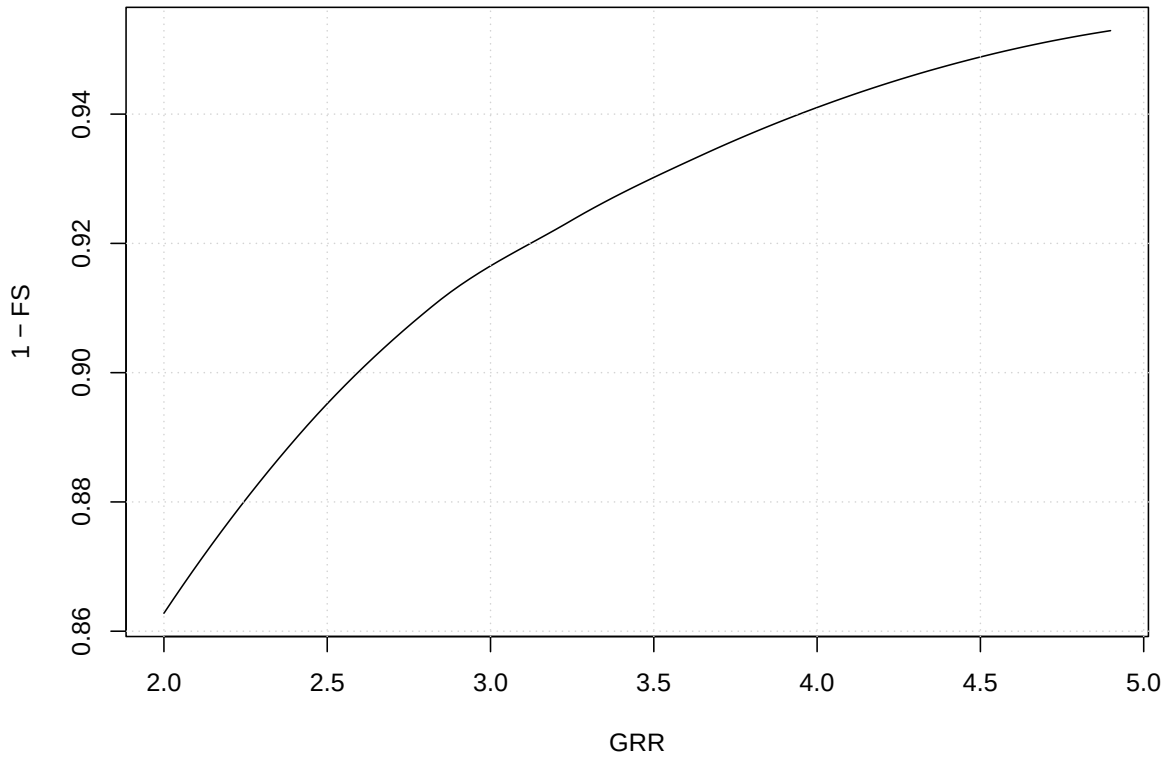
Esta función es **optimShd**, la cual en base a una resolución (determinada por el argumento **res**, el cual, indica el incremento de la secuencia de distancias) obtiene la producción de cada combinación y la plasma en un objeto de clase **Shade**.

```
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
shadeplot(ShdM2x)
```

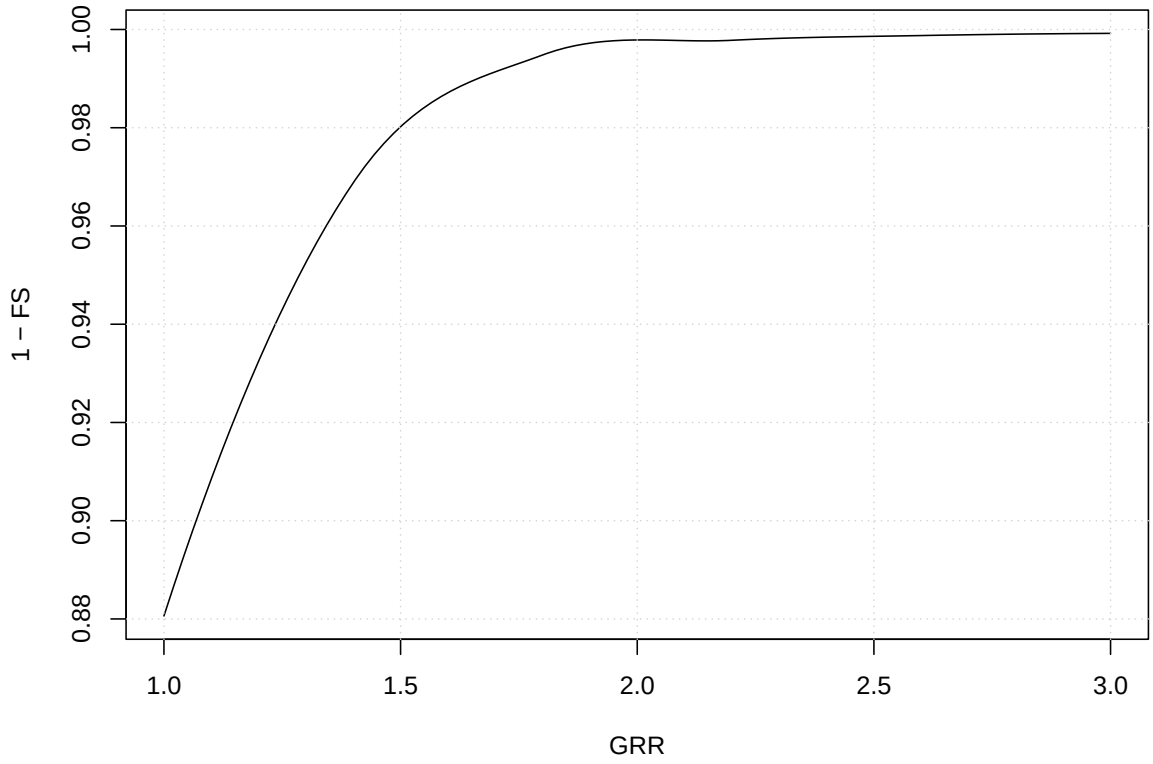



```
structHoriz = list(L = 4.83)
distHoriz = list(Lew = structHoriz$L * c(2,5))
Shd12HorizBT <- optimShd(lat = lat, dataRad = prom,
                         modeTrk = 'horiz',
                         betaLim = 60,
                         distances = distHoriz, res = 2,
```

```
                          struct = structHoriz,
                          modeShd = 'bt',
                          prog = FALSE) #Se quita la barra de progreso
shadeplot(Shd12HorizBT)
```




```
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
shadeplot(Shd12Fixed)
```



4.8. Aspectos técnicos de la elaboración de un paquete en R

4.8.1. Estructura básica del paquete

En la creación de un paquete en `R`, la estructura de los archivos es clave para asegurar un desarrollo organizado y que `R` pueda interactuar correctamente con el código y los datos. Los paquetes de R son esencialmente un conjunto de archivos organizados en un directorio específio. El contenido mínimo requerido incluye:

- Un archivo **DESCRIPTION**, que proporciona la información esencial del paquete.
- Un archivo **NAMESPACE**, que controla qué funciones y objetos son visibles fuera del paquete.
- Subdirectoriso como `R/` y `man/`:
  - `R/`: Contiene los archivos de codigo `.R`, que son las funciones, clases y métodos definidos en el paquete.
  - `man/`: Contiene las páginas de ayuda y documentación para las funciones, métodos y clases del paqeute.

La estructura básica de un paquete puede generarse fácilmente utilizando la función `package.skeleton()`, que crea los archivos y carpetas necesarios para empezar a trabajar en el desarrollo.

4.8.2. DESCRIPTION

El fichero **DESCRIPTION** es fundamental, ya que incluye la información descriptiva y técnica del paquete, como el nombre, la versión, los autores y las dependencias. Un ejemplo típico de este archivo es el siguiente:

```
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 principales de este archivo son:

- **Package**: Nombre del paquete.
- **Version**: Versión del paquete. Generalmente sigue un esquema de numeración semántica (`major.minor-patch`)[9].
- **Title**: Un título breve pero descriptivo de lo que hace el paquete.
- **Authors@R**: Especifica el o los autores con sus respectivos roles, como “aut” (autor) y “cre” (creador principal).
- **Maintainer**: Persona responsable del mantenimiento del paquete, con su correo electrónico.
- **Depends**: Lista de dependencias, es decir, otros paquetes de los que depende el correcto funcionamiento del paquete.
- **Suggests**: Lista de paqeute que no son obligatorios, pero que pueden ser útiles.
- **Description**: Una breve descripción del propósito del paquete.
- **License:** Tipo de licencia bajo la cual se distribuye el paquete (GPL, MIT, etc.).

Este archivo es crucial para que los usuarios y el sistema `R` identifiquen las características y requisitos del paquete

[9] Un esquema de numeración semántica es un sistema de versiones que sigue un patrón específico para asignar números a las versiones de software. Se utiliza para indicar claramente la magnitud de los cambios realizados y su impacto en la compatibilidad. Una versión `major` o mayor se refiere a modificaciones grandes o incompatibles con versiones anteriores, `minor` o menor es una versión que incluye mejoras o nuevas funciones compatibles con versiones anteriores y `patch` o parche es una versión que incluye correcciones menores o mejoras que no afectan a la funcionalidad.

4.8.3. NAMESPACE

El archivo **NAMESPACE** es el encargado de gestionar el espacio de nombres del paquete, permitiendo definir qué funciones y objetos serán visibles (exportados) y cuáles se mantendrán internos. Además, es útil para definir qué funciones o métodos de otros paquetes serán importados para us uso dentro del paquete.

`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()`).

```
1 search()
```

```
 [1] ".GlobalEnv"           "package:jsonlite"     "package:httr2"        "package:zoo"
 [5] "package:solaR2"       "package:latticeExtra" "package:lattice"      "package:data.table"
 [9] "ESSR"                 "package:stats"        "package:graphics"     "package:grDevices"
[13] "package:utils"        "package:datasets"     "package:methods"      "Autoloads"
[17] "package:base"
```

Manejo de variables

- Exportar variables:

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

  Esto asegura que las variables `f` y `g` sean accesibles desde fuera del paquete.

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

```
1 import(pkgExt)
```

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

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

Manejo de clases y métodos

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

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

Esto permite definir cómo se imprimen objetos de la clase **`myClass`**

- Para los paquetes que utilizan clases y métodos **`S4`**, es necesario agregar una dependencia explícita en el archivo **DESCRIPTION**:

```
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, ...)
```

4.8.4. Documentación

La documentación en R sigue un formato específico llamado **`Rd`** (*R documentation*), que está inspirado en LaTex. Cada función, método o clase del paquete debe tener una página de documentación asociada, que generalmente se encuentra en el subdirectorio **`man/`**. Estas páginas incluyen información sobre el uso de la función, argumentos, detalles de la implementación y ejemplos de uso.

```
\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}
```

El formato tiene varios componentes:

- **name**: El nombre de la función.
- **alias**: Nombres alternativos o alias de la función.
- **title**: Título breve que describe la función.
- **description**: Una descripción de lo que hace la función.
- **usage**: La sintaxis de la función de lo que hace la función.
- **arguments**: Explicación de los argumentos que recibe la función.
- **seealso**: Enlaces a funciones relacionadas.
- **examples**: Ejemplos de cómo utilizar la función.

Esta estructura de documentación permite a los usuarios comprender rápidamente cómo utilizar las funciones del paquete y verificar su funcionalidad con ejemplos prácticos.

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**.

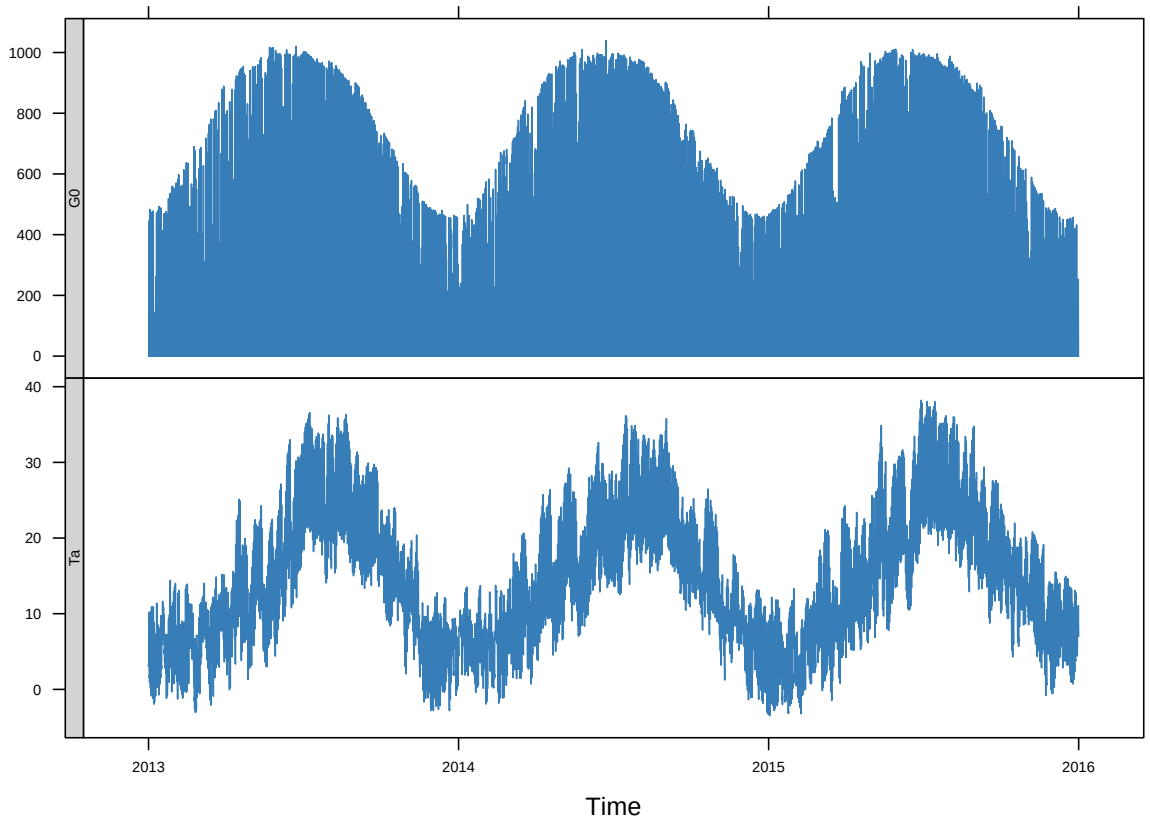
5.1. solaR

Se empieza inicilizando el paquete:

```
1 library(solaR)
```

En el estudio anterior, se recopilaron datos intradiarios de irradiación intradiaria. Sin embargo, estos datos fueron tomados en un plano inclinado, por lo que ni **solaR**, ni **solaR2** son capaces de interpretarlos correctamente. En su lugar, para este ejemplo se van a tomar datos horarios de la plataforma **PVGIS** [PVG24] de los años 2013, 2014 y 2015 en la localización de la ETSIDI.

```
1 etsidi_1315 <- readBDi(file = 'TFG/data/PVGIS_1315.csv',
2                        lat = 40.4, time.col = 'Dates',
3                        format = '%Y-%m-%d %H:%M:%S')
4 xyplot(etsidi_1315)
```



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.

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

```
## 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,
                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,
                 Vmax = 320)
```

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 %

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:

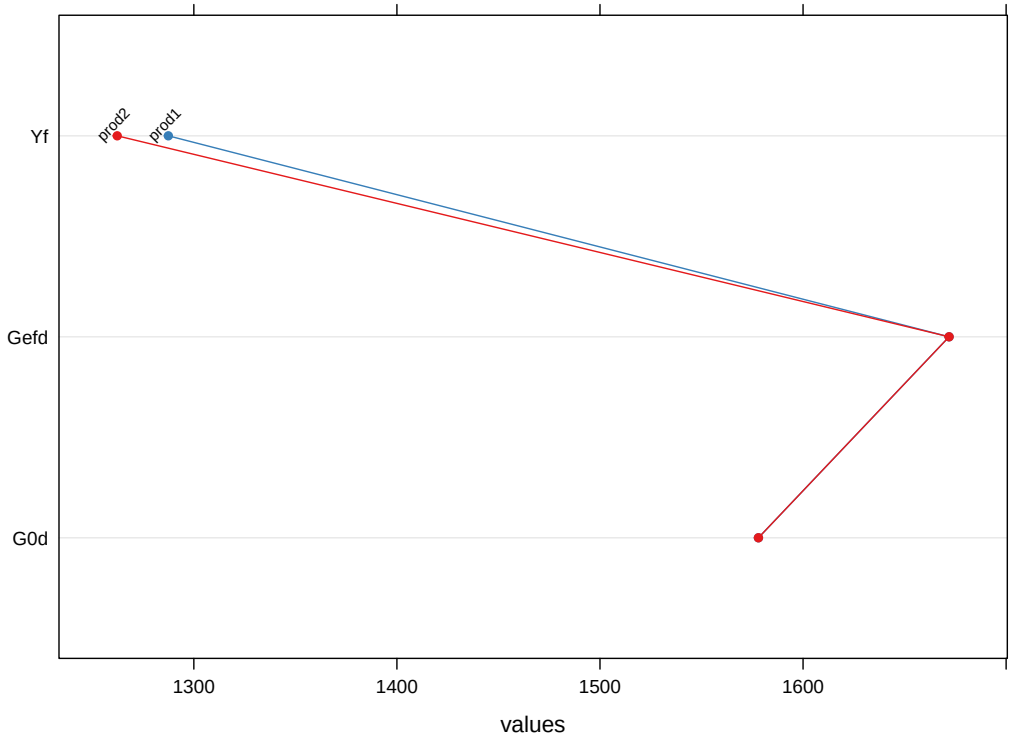
```
prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = etsidi_1315,
                  beta = 30, alfa = -19,
                  module = module1, generator = generator1,
                  inverter = inverter)
prod2 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = etsidi_1315,
                  beta = 30, alfa = -19,
                  module = module2, generator = generator2,
                  inverter = inverter)
compare(prod1, prod2)
```

```
show(as.zooY(prod1))
```

```
          Eac      Edc       Yf
2013 1583.545 1757.235 1265.505
2014 1600.589 1775.426 1279.126
2015 1648.836 1828.569 1317.683
```

```
show(as.zooY(prod2))
```

```
          Eac      Edc       Yf
2013 1367.625 1517.779 1242.674
2014 1379.995 1530.833 1253.914
2015 1419.956 1574.704 1290.224
```



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.

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

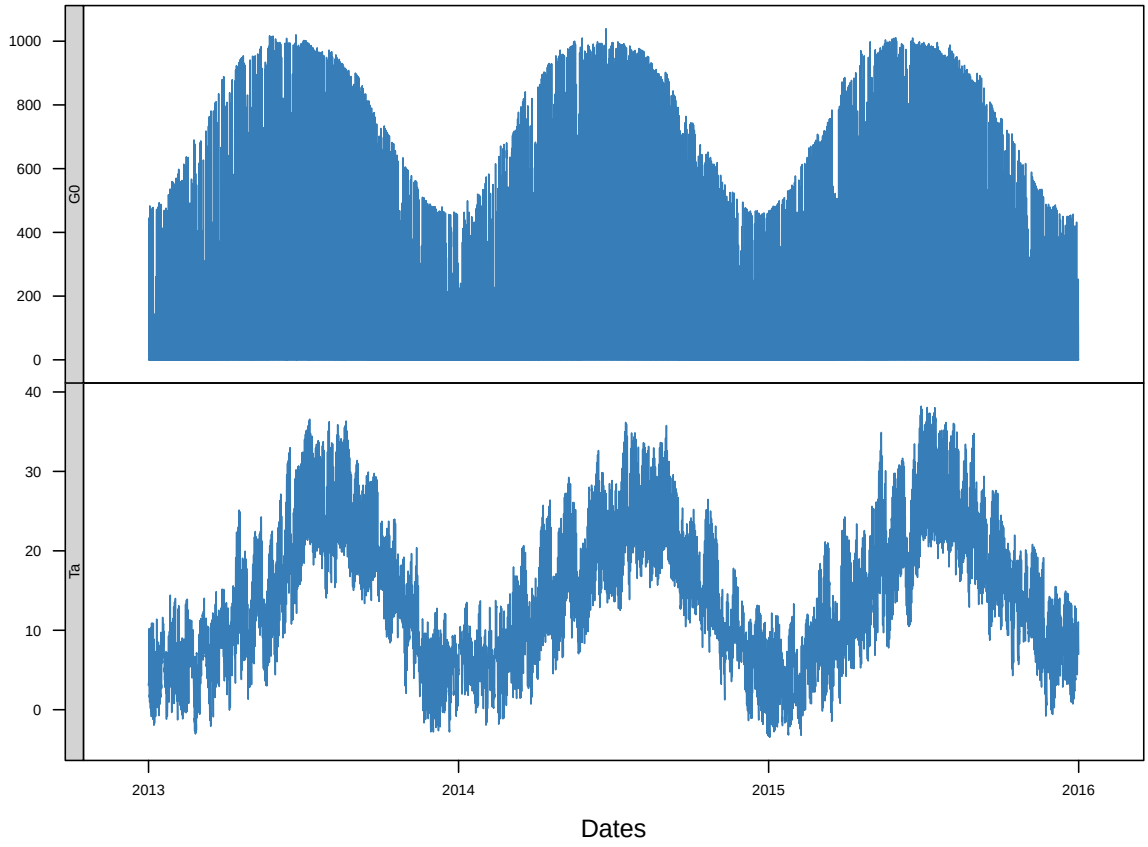
5.3. solaR2

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

```
library(solaR2)
```

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

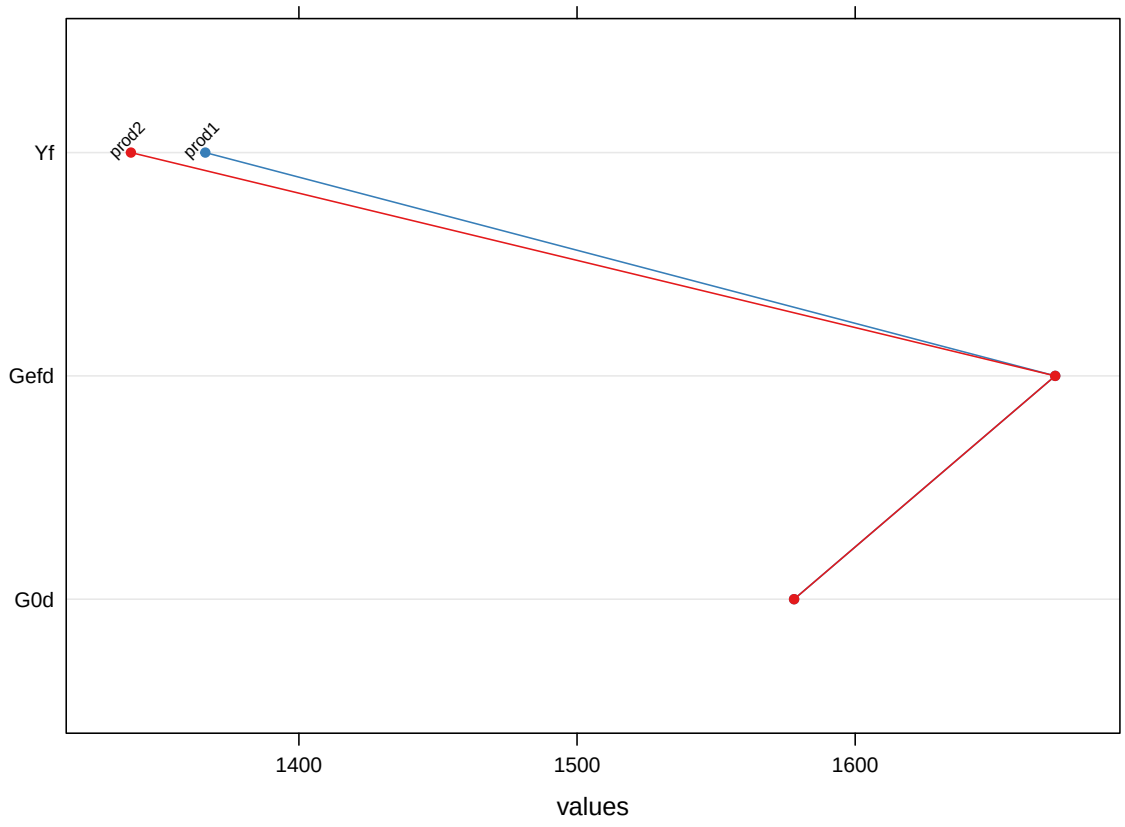
```
etsidi_1315 <- readBDi(file = 'TFG/data/PVGIS_1315.csv',
                       lat = 40.4, dates.col = 'Dates',
                       format = '%Y-%m-%d %H:%M:%S')
xyplot(etsidi_1315)
```



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 = etsidi_1315,
                  beta = 30, alpha = -19,
                  module = module1, generator = generator1,
                  inverter = inverter)
prod2 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                  dataRad = etsidi_1315,
                  beta = 30, alpha = -19,
                  module = module2, generator = generator2,
```

```
                    inverter = inverter)
compare(prod1, prod2)
```




```
show(as.data.tableY(prod1))
```

```
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2013 1681.077 1757.235 1343.449
2:  2014 1698.613 1775.426 1357.463
3:  2015 1749.536 1828.569 1398.158
```

```
show(as.data.tableY(prod2))
```

```
   Dates      Eac      Edc       Yf
   <int>    <num>    <num>    <num>
1:  2013 1451.873 1517.779 1319.225
2:  2014 1464.483 1530.833 1330.683
3:  2015 1506.544 1574.704 1368.901
```

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:

- **Modularidad**: el paquete **solaR2** presenta muchas funciones que son capaces de realizar pequeñas operaciones, al contrario que **solaR**, que no permite esto.
- **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)
## se recortan los datos a un solo año
etsidi_13 <- etsidi_1315[as.Date('2013-01-01'), as.Date('2013-12-31')]
prodGCPVcustom <- function(){
  prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                    dataRad = etsidi_13, beta = 30, alfa =-19,
                    module = module1, generator = generator1,
                    inverter = inverter)
}
microbenchmark(prodGCPVcustom(), times = 20)
```

```
Unit: milliseconds
            expr      min       lq     mean   median       uq     max neval
 prodGCPVcustom() 758.3203 769.8908 777.2625 779.0366 784.3147 791.931    20
```

```
## Con el paquete solaR2
library(microbenchmark)
etsidi_13 <- etsidi_1315[as.Date('2013-01-01'), as.Date('2013-12-31')]
prodGCPVcustom <- function(){
  prod1 <- prodGCPV(lat = 40.4, modeTrk = 'fixed', modeRad = 'bdI',
                    dataRad = etsidi_13, beta = 30, alpha =-19,
                    module = module1, generator = generator1,
                    inverter = inverter)
}
microbenchmark(prodGCPVcustom(), times = 20)
```

```
Unit: milliseconds
            expr      min       lq     mean   median       uq      max neval
 prodGCPVcustom() 520.3954 530.1572 536.8405 534.1607 539.0759 573.3836    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 7, 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 [aut] (<https://orcid.org/0000-0002-4134-7196>), Francisco Delgado-López [aut, cre]

Maintainer Francisco Delgado-López <f.delgadol@alumnos.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, (m³/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 (m³) and productivity of the system.

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

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

`pump` A `list` extracted from `pumpCoef`

`H` Total manometric head (m)

`Pg` Nominal power of the PV generator (Wp)

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

`effSys` list of numeric values with information about the system losses

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

Extends

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

Methods

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

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

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

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

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

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

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

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

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

Author(s)

Oscar Perpiñán Lamigueiro.

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 intraidaily 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 0.0049 volts per celsius degree)
	TONC nominal operational cell temperature, celsius degree (default value 45).
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(0.002, 0.005, 0.008)), or a matrix of nine values (3x3) if there is dependence with the voltage (see references).
	Pinv nominal inverter power (W) (default value 1.5e6 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 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

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.

[JG20] Michael Schmutz Jan Remund Stefan Müller y Pascal Graf. *Meteonorm Version 8*. Versión 8.0. Meteotest. Berna, Suiza, 2020. URL: https://meteonorm.com.

[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.