



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO
INDUSTRIAL

Grado en Ingeniería Eléctrica

TRABAJO FIN DE GRADO

TÍTULO DEL TRABAJO

Francisco Delgado López

Tutor: Óscar Perpiñán Lamigueiro

Departamento: Departamento de Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

Madrid, Septiembre, 2024

Copyright ©2024. Francisco Delgado López

Esta obra está licenciada bajo la licencia Creative Commons Atribución-No Comercial-Sin Derivadas 3.0 Unported (CC BY-NC-ND 3.0). Para ver una copia de esta licencia, visite

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es> o envíe una carta a Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, EE.UU.

Todas las opiniones aquí expresadas son del autor, y no reflejan necesariamente las opiniones de la Universidad Politécnica de Madrid.

Título: título del trabajo

Autor: Francisco Delgado López

Tutor: Óscar Perpiñán Lamigueiro

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día de
de ... en, en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Uni-
versidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO

PRESIDENTE

Agradecimientos

Agradezco a ...

Resumen

Este proyecto se resume en

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

Abstract

In this project.....

Keywords: solar geometry, solar radiation, solar energy, photovoltaic, visualitation methods, temporal series, space-time data, S4

Índice general

| | |
|---|-------------|
| Agradecimientos | VII |
| Resumen | IX |
| Abstract | XI |
| Índice general | XII |
| Índice de figuras | XIII |
| Índice de tablas | XIV |
| Nomenclatura | XV |
| 1 Introducción | 1 |
| 1.1. Objetivos | 1 |
| 1.2. Análisis previo de soluciones | 2 |
| 1.3. Aspectos técnicos | 3 |
| 2 Estado del arte | 5 |
| 2.1. Situación actual de la generación fotovoltaica | 5 |
| 2.2. Soluciones existentes y sus carencias | 6 |
| 3 Parte teórica y desarrollo del código | 7 |
| 4 Ejemplo práctico de aplicación | 11 |
| 4.1. solaR2 | 11 |
| 4.2. solaR | 11 |
| 4.3. PVsyst | 11 |
| 4.4. Comparación entre los tres | 11 |
| 5 Detalles de la programación | 13 |
| A Código completo | 15 |
| A.1. Constructores | 15 |
| A.2. Clases | 38 |
| A.3. Funciones | 40 |
| A.4. Métodos | 64 |
| A.5. Conjunto de datos | 84 |
| Bibliografía | 87 |

Índice de figuras

| | |
|---|---|
| 3.1. Procedimiento de cálculo | 8 |
| 3.2. Proceso de cálculo de las funciones de <code>solaR2</code> | 8 |

Índice de tablas

Nomenclatura

| | |
|--------------|---|
| α | Ángulo azimutal de la superficie |
| B_0 | Irradiancia extra-atmosférica o extra-terrestre |
| B | Irradiancia directa |
| β | Ángulo de inclinación de la superficie |
| D | Irradiancia difusa |
| δ | Declinación |
| EoT | Ecuación del tiempo |
| ϵ_0 | Corrección debida a la excentricidad de la elipse de la trayectoria terrestre alrededor del sol |
| F_D | Fracción de difusa |
| FT | Factor de pérdidas angulares |
| G | Irradiancia global |
| γ_s | Altura solar |
| K_T | Índice de claridad |
| ω | Hora solar o tiempo solar verdadero |
| ω_s | Ángulo del amanecer |
| ψ_s | Ángulo azimutal solar |
| R | Irradiancia de albedo |
| θ_s | Ángulo de incidencia o ángulo entre el vector solar y el vector director de una superficie |

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 posible generación de una instalación fotovoltaica.

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 podrá 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 eficaz 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 estas producciones 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 2012. Por la antigüedad del código 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 alojados 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 [CITAR CÓDIGO]
- Tratamiento de datos meteorológicos (en especial de radiación), procedentes de datos ofrecidos del usuario y de la red de estaciones SIAR [Min23] [CITAR CÓDIGO]
- Una vez calculado lo anterior, se pueden hacer estimaciones de:

- Los componentes de radiación horizontal [CITAR CALCGO].
- Los componentes de radiación eficaz en el plano inclinado [CITAR CALCGEF].
- La producción de sistemas fotovoltaicos conectados a red [CITAR PRODGCPV] y sistemas fotovoltaicos de bombeo [CITAR PRODPVPS].

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 preprocesador de textos L^AT_EX (partiendo de un archivo .org, se puede exportar a un archivo .tex para posteriormente exportar un pdf).
- Obtener conocimientos teóricos acerca de la radiación solar y de la producción de energía solar mediante sistemas fotovoltaicos y sus diversos tipos. Para ello se ha usado en mayor medida el libro “Energía Solar Fotovoltaica” [Per23].

1.2. Análisis previo de soluciones

Este proyecto, como ya se ha comentado, es el heredero del paquete `solar` desarrollado por Oscar Perpiñán. La filosofía de ambos paquetes es la misma y los resultados que dan son muy similares. Sin embargo, lo que les diferencia es el paquete sobre el que construyen sus datos. Mientras que `solar` basa sus clases y métodos en el paquete `zoo`, `solar2` en el paquete `data.table`. Los dos paquetes pueden trabajar con series temporales, pero, mientras que `zoo` es más eficaz trabajando con series temporales, `data.table` es más eficiente a la hora de trabajar con una cantidad grande de datos, lo cual a la hora de realizar estimaciones muy precisas es beneficioso. Por otro lado, existen otras soluciones fuera de R:

1. PVsyst - Photovoltaic Software

Este software es probablemente el más conocido dentro del ámbito del estudio y la estimación de instalaciones fotovoltaicas. Permite una gran personalización de todos los componentes de la instalación.

2. SISIFO

Herramienta web diseñada por el **Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar de la Universidad Politécnica de Madrid**.

3. PVGIS

Aplicación web desarrollada por el **European Commission Joint Research Center** desde 2001.

4. System Advisor Model

Desarrollado por el **Laboratorio Nacional de Energías Renovables**, perteneciente al Departamento de energía del gobierno de EE.UU.

En el apartado [4] se realizará un ejemplo práctico que compare los resultados entre **PVsyst**, **solaR** y **solaR2**

1.3. Aspectos técnicos

Para elaborar un paquete en R se deben aportar una serie de ficheros:

- **R**: Fichero que contiene todos los archivos .R que se van a ejecutar en la instalación del paquete. Esto incluye funciones, clases y métodos.
- **data**: Aquí se incluyen los datos externos que el paquete necesita para funcionar.
- **DESCRIPTION**: Contiene metadatos sobre el paquete, como el nombre, la versión, el autor, etc.
- **NAMESPACE**: Especifica qué funciones y datos se exportan y se importan.
- **inst**: Se usa para almacenar archivos importantes pero que no se almacenan en el resto de ficheros.
- **tests**: Se utiliza para almacenar scripts de pruebas que aseguran que el código del paquete funcione correctamente.
- **man**: Donde se alojan los ficheros .Rd relacionados con el manual de uso del paquete. En estos se almacenan la información de funciones, métodos, clases y datos.

Una vez se tienen todos estos ficheros, el paquete se construye y se prueba.

Estado del arte

2.1. Situación actual de la generación fotovoltaica

Según el informe anual de 2023 de la UNEF¹[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, convirtiéndose 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 REPowerEU²[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.

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

¹UNEF: Unión Española Fotovoltaica.

²Plan 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 transición hacia una energía limpia.

³Plan + 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.

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 PERTE⁴, 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 PIB⁵, un 51 % más que el año anterior, y generando una huella económica total de 15.656 millones de euros. En términos de empleo, el sector involucró a 197.383 trabajadores, de los cuales 40.683 fueron 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 componentes 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.

UNEF promueve la transformación industrial para que España se convierta en un hub industrial fotovoltaico. Se destaca la necesidad de proteger la industria existente, garantizar un crecimiento constante de la capacidad y ofrecer condiciones de financiamiento favorables. Además se propone implementar una Estrategia Industrial Fotovoltaica para contribuir significativamente a la reindustrialización de la economía, aprovechando las medidas del REPower Plan, la Estrategia Solar y la Alianza de la Industria Solar Fotovoltaica.

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 existentes y sus carencias

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

⁵PIB: Producto Interior Bruto.

Parte teórica y desarrollo del código

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. Obtener la irradiación global diaria en el plano horizontal
2. A partir de la irradiación global, obtener las componentes de difusa y directa.
3. Se trasladan estos valores de irradiación a valores de irradiancia.
4. Con estos valores se pueden obtener los valores correspondientes en el plano del generador
 - a) Sin los efectos de la suciedad de los módulos y las sombras que se generan unos con otros.
 - b) Con estos efectos
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.

En la figura 3.2, 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. Se calcula la geometría que definen la posición de la Tierra frente al Sol.
 - a) Mediante la función `fSolD`¹, se calcula:
 - El ángulo de declinación de la Tierra (δ).
 - La corrección debida a la excentricidad de la elipse de la trayectoria terrestre alrededor del sol (ϵ_0).

¹Toda función mencionada en este capítulo, está descrita en el anexo A

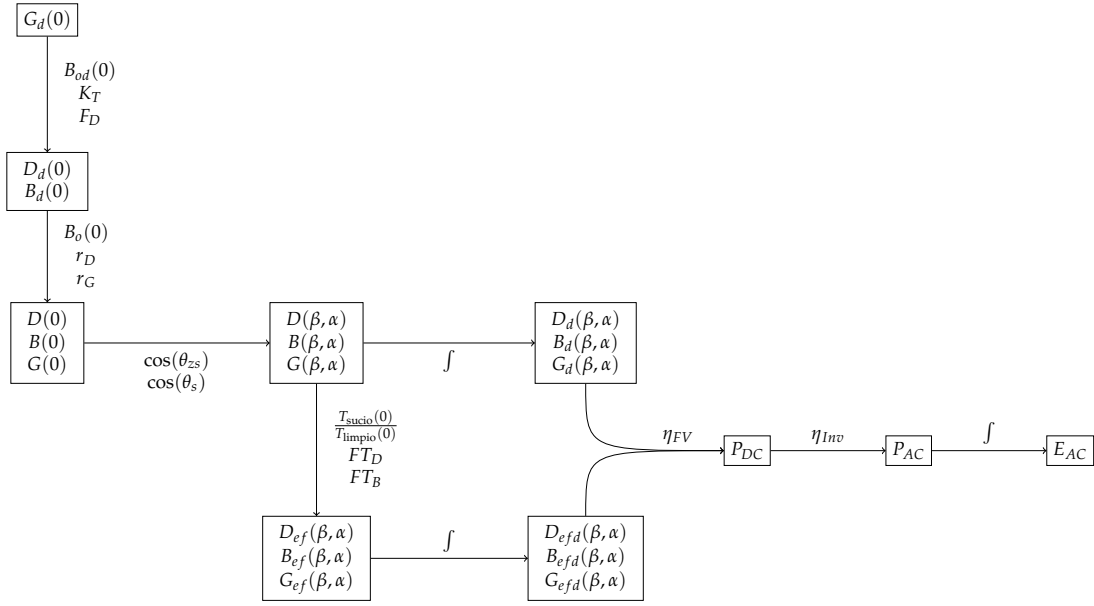


Figura 3.1: Procedimiento de cálculo

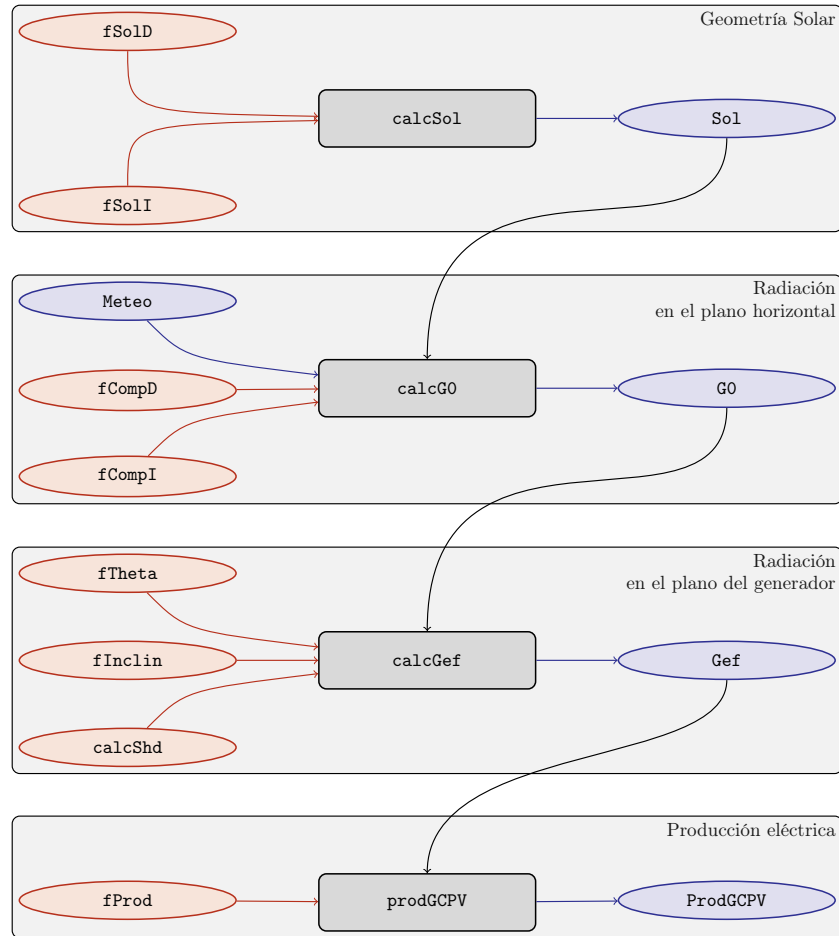


Figura 3.2: Proceso de cálculo de las funciones de solar2

- La ecuación del tiempo (EoT).
 - El ángulo del amanecer (ω_s).
- b) Mediante la función `fSolI`, se calcula:
- La hora solar (ω).
 - El momento del día en el que es de noche.
 - El ángulo cenital solar (θ_{zs}) {Ángulo cenital solar}.
 - El ángulo de altura solar (γ_s).
 - El ángulo azimutal solar (ψ_s).
 - La irradiancia extra-terrestre en el plano horizontal ($B_0(0)$).
- c) El resultado de ambas funciones se juntan en un solo objeto de clase `Sol` mediante la función `calcSol`.
2. Se estima la radiación en el plano horizontal.
- a) La información de irradiación en el plano horizontal (en todos sus componentes o, en su defecto, solo la global($G_d(0)$)) y temperatura viene dada en un objeto de clase `Meteo`.
- b) Mediante la función `fCompD`, se calcula:
- La fracción de radiación difusa diaria (F_{Dd}).
 - El índice de claridad diario (K_{Td}).
 - Si solo se tienen datos de la componente global de irradiación:
 - La irradiación directa en el plano horizontal ($B_d(0)$).
 - La irradiación difusa en el plano horizontal ($D_d(0)$).
- c) Mediante la función `fCompI`, se calcula:
- La fracción de radiación difusa (F_D).
 - El índice de claridad (K_T).
 - Si solo se tienen datos de la componenete global de irradiancia ($G(0)$):
 - La irradiancia directa en el plano horizontal ($B(0)$).
 - La irradiancia difusa en el plano horizontal ($D(0)$).
- d) El resultado de ambas funciones junto a medias mensuales y valores anuales se consolidan en un solo objeto de clase `G0` (que incluye los objetos `Sol` y `Meteo` de los que parte) mediante la función `calcG0`.
3. Se estima la radiación en el plano del generador.
- a) La información de radiación puede venir dada en forma de un objeto de clase `Meteo` o un objeto de clase `G0` (ya que es este último el que se necesita para estimar la radiación en el plano del generador).
- b) Mediante la función `fTheta`, se calcula:
- Ángulo de inclinación de la superficie del módulo (β).
 - Ángulo azimutal de la superficie del módulo (α).
 - Ángulo de incidencia de la irradiancia solar en la superficie del módulo (θ_s).
- c) Mediante la función `fInclin`, se calcula:
- La irradiancia extra-terrestre en la superficie inclinada ($B_0(\beta, \alpha)$).
 - La irradiancia directa normal ($B(n)$).
 - Las irradiancias global ($G(\beta, \alpha)$), directa ($B(\beta, \alpha)$), difusa ($D(\beta, \alpha)$) (total, isotrópica y anisotrópica) y de albedo ($R(\beta, \alpha)$) sobre una superficie inclinada.

- Las irradiancias efectivas global ($G_{ef}(\beta, \alpha)$), directa ($B_{ef}(\beta, \alpha)$), difusa ($D_{ef}(\beta, \alpha)$) (total, isotrópica y anisotrópica) y de albedo ($R_{ef}(\beta, \alpha)$) sobre una superficie inclinada.
 - Los factores de pérdidas angulares para las componentes directa (FT), difusa (FT_D), y de albedo (FT_R).
- d) Mediante la función `calcShd`, se puede calcular:
- La irradiancia e irradiación incluyendo sombras para seguidores a dos ejes y horizontales y paneles fijos mediante la función `fSombra`.
- e) El resultado de estas funciones junto a medias mensuales y valores anuales se consolidan en un solo objeto de clase **Gef** (que incluye el objeto **G0** del que parte) mediante la función `calcGef`.
4. Se estima la producción eléctrica.
- a) Mediante la función `fProd`, se calcula:
- La potencia en corriente continua (P_{DC}).
 - La potencia en corriente alterna (P_{AC}).
- b) Estos resultados, llevados a valores diarios, mensuales y anuales, se pueden convertir en valores de energía (E_{DC} y E_{AC}) y de productividad del sistema (Y_f), los cuales se consolidan en un solo objeto de clase **ProdGCPV** (que incluye el objeto **Gef** del que parte) mediante la función `prodGCPV`.

Ejemplo práctico de aplicación

Como demostración se va a realizar un caso práctico...

4.1. solaR2

...

4.2. solaR

...

4.3. PVsyst

...

4.4. Comparación entre los tres

Detalles de la programación

...

Código completo

Todo el código que se muestra a continuación está disponible...

A.1. Constructores

calcSol

```

1 calcSol <- function(lat, BTd,
2                     sample = 'hour', BTi,
3                     EoT = TRUE,
4                     keep.night = TRUE,
5                     method = 'michalsky')
6 {
7   if(missing(BTd)) BTd <- truncDay(BTi)
8   solD <- fSolD(lat, BTd, method = method) #daily values
9   solI <- fSolI(solD = solD, sample = sample, #intradaily values
10              BTi = BTi, keep.night = keep.night,
11              EoT = EoT, method = method)
12
13   if(!missing(BTi)){
14     sample <- solI$Dates[2]-solI$Dates[1]
15     sample <- format(sample)
16   }
17
18   solD[, lat := NULL]
19   solI[, lat := NULL]
20   result <- new('Sol',
21               lat = lat,
22               solD = solD,
23               solI = solI,
24               sample = sample,
25               method = method)
26   return(result)
27 }
```

calcG0

```

1 calcG0 <- function(lat,
2                   modeRad='prom',
3                   dataRad,
```

```

4         sample='hour',
5         keep.night=TRUE,
6         sunGeometry='michalsky',
7         corr, f, ...)
8     {
9
10        if (missing(lat)) stop('lat missing. You must provide a latitude value.')
11
12        stopifnot(modeRad %in% c('prom', 'aguiar', 'bd', 'bdI'))
13
14
15    ###Datos de Radiacion
16        if (missing(corr)){
17            corr = switch(modeRad,
18                bd = 'CPR', #Correlation between Fd and Kt for daily values
19                aguiar = 'CPR', #Correlation between Fd and Kt for daily values
20                prom = 'Page', #Correlation between Fd and Kt for monthly
21                averages
22                values
23                bdI = 'BRL' #Correlation between fd and kt for intraday
24                )
25        }
26
27        if(is(dataRad, 'Meteo')){BD <- dataRad}
28        else{
29            BD <- switch(modeRad,
30                bd = {
31                    if (!is.list(dataRad)) dataRad <- list(file=dataRad)
32                    switch(class(dataRad$file)[1],
33                        character={
34                            bd.default=list(file='', lat=lat)
35                            bd=modifyList(bd.default, dataRad)
36                            res <- do.call('readBDd', bd)
37                            res
38                        },
39                        data.table= ,
40                        data.frame={
41                            bd.default=list(file='', lat=lat)
42                            bd=modifyList(bd.default, dataRad)
43                            res <- do.call('dt2Meteo', bd)
44                            res
45                        },
46                        zoo={
47                            bd.default=list(file='', lat=lat, source='')
48                            bd=modifyList(bd.default, dataRad)
49                            res <- do.call('zoo2Meteo', bd)
50                            res
51                        })
52                }, #End of bd
53                prom = {
54                    if (!is.list(dataRad)) dataRad <- list(G0dm=dataRad)
55                    prom.default <- list(G0dm=numeric(), lat=lat)
56                    prom = modifyList(prom.default, dataRad)
57                    res <- do.call('readG0dm', prom)
58                }, #End of prom
59                aguiar = {
60                    if (is.list(dataRad)) dataRad <- dataRad$G0dm
61                    BTd <- fBTd(mode='serie')

```



```

60         sold <- fSold(lat, BTd)
61         G0d <- markovG0(dataRad, sold)
62         res <- dt2Meteo(G0d, lat=lat, source='aguilar')
63     }, #End of aguilar
64     bdI = {
65         if (!is.list(dataRad)) dataRad <- list(file=dataRad)
66         switch(class(dataRad$file)[1],
67             character = {
68                 bdI.default <- list(file='', lat=lat)
69                 bdI <- modifyList(bdI.default, dataRad)
70                 res <- do.call('readBDi', bdI)
71                 res
72             },
73             data.table = ,
74             data.frame = {
75                 bdI.default <- list(file='', lat=lat)
76                 bdI <- modifyList(bdI.default, dataRad)
77                 res <- do.call('dt2Meteo', bdI)
78                 res
79             },
80             zoo = {
81                 bdI.default <- list(file='', lat=lat, source='')
82                 bdI <- modifyList(bdI.default, dataRad)
83                 res <- do.call('zoo2Meteo', bdI)
84                 res
85             },
86             stop('dataRad$file should be a character, a data.table, a
data.frame or a zoo.'))
87     }) #End of btI
88     #End of general switch
89 }
90
91
92 ### Angulos solares y componentes de irradiancia
93 if (modeRad=='bdI') {
94     sol <- calcSol(lat, sample = sample,
95                   BTi = indexD(BD), keep.night=keep.night, method=sunGeometry)
96     compI <- fCompI(sol=sol, G0I=BD, corr=corr, f=f, ...)
97     compD <- compI[, lapply(.SD, P2E, sol@sample),
98                     .SDcols = c('G0', 'D0', 'B0'),
99                     by = truncDay(Dates)]
100     names(compD)[1] <- 'Dates'
101     names(compD)[-1] <- paste(names(compD)[-1], 'd', sep = '')
102     compD$F0 <- compD$D0d/compD$G0d
103     compD$Kt <- compD$G0d/sol@solD$Bo0d
104 } else { ##modeRad!='bdI'
105     sol <- calcSol(lat, indexD(BD), sample = sample,
106                   keep.night = keep.night, method = sunGeometry)
107     compD<-fCompD(sol=sol, G0d=BD, corr=corr, f, ...)
108     compI<-fCompI(sol=sol, compD=compD, ...)
109 }
110
111 ###Temperature
112
113 Ta=switch(modeRad,
114           bd={
115             if (all(c("TempMax", "TempMin") %in% names(BD@data))) {
116                 fTemp(sol, BD)

```

```

117     } else {
118         if ("Ta" %in% names(BD@data)) {
119             data.table(Dates = indexD(sol),
120                         Ta = BD@data$Ta)
121         } else {
122             warning('No temperature information available!')
123         }
124     }
125 },
126 bdI={
127     if ("Ta" %in% names(BD@data)) {
128         data.table(Dates = indexI(sol),
129                     Ta = BD@data$Ta)
130     } else {
131         warning('No temperature information available!')
132     }
133 },
134 prom={
135     if ("Ta" %in% names(BD@data)) {
136         data.table(Dates = indexD(sol),
137                     Ta = BD@data$Ta)
138     } else {
139         warning('No temperature information available!')
140     }
141 },
142 aguiar={
143     data.table(Dates = indexI(sol),
144                 Ta = BD@data$Ta)
145 }
146 )
147
148 ###Medias mensuales y anuales
149 nms <- c('G0d', 'D0d', 'B0d')
150 G0dm <- compD[, lapply(.SD/1000, mean, na.rm = TRUE),
151                 .SDcols = nms,
152                 by = .(month(Dates), year(Dates))]
153
154 if(modeRad == 'prom'){
155     G0dm[, DayOfMonth := DOM(G0dm)]
156     G0y <- G0dm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
157                 .SDcols = nms,
158                 by = .(Dates = year)]
159     G0dm[, DayOfMonth := NULL]
160 } else{
161     G0y <- compD[, lapply(.SD/1000, sum, na.rm = TRUE),
162                 .SDcols = nms,
163                 by = .(Dates = year(Dates))]
164 }
165 G0dm[, Dates := paste(month.abb[month], year, sep = '. ')]
166 G0dm[, c('month', 'year') := NULL]
167 setcolorder(G0dm, 'Dates')
168
169 ###Result
170 result <- new(Class='GO',
171               BD,          #GO contains "Meteo"
172               sol,         #GO contains 'Sol'
173               GOD=compD,   #results of fCompD
174               G0dm=G0dm,   #monthly means

```

```

175         GOy=GOy,      #yearly values
176         GOI=compI,    #results of fCompD
177         Ta=Ta         #ambient temperature
178     )
179     return(result)
180 }

```

calcGef

```

1  calcGef<-function(lat,
2      modeTrk='fixed',      #c('two','horiz','fixed')
3      modeRad='prom',
4      dataRad,
5      sample='hour',
6      keep.night=TRUE,
7      sunGeometry='michalsky',
8      corr, f,
9      betaLim=90, beta=abs(lat)-10, alfa=0,
10     iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
11     modeShd='',          #modeShd=c('area','bt','prom')
12     struct=list(), #list(W=23.11, L=9.8, Nrow=2, Ncol=8),
13     distances=data.frame(),#data.table(Lew=40, Lns=30, H=0)){
14     ...){
15
16     stopifnot(is.list(struct), is.data.frame(distances))
17
18     if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
19         modeShd[which(modeShd=='bt')]='area'
20         warning('backtracking is only implemented for modeTrk=horiz')}
21
22     if (modeRad!='prev'){ #not use a prev calculation
23         radHoriz <- calcG0(lat=lat, modeRad=modeRad,
24             dataRad=dataRad,
25             sample=sample, keep.night=keep.night,
26             sunGeometry=sunGeometry,
27             corr=corr, f=f, ...)
28     } else {
29         radHoriz <- as(dataRad, 'G0')
30     }
31
32     ### Inclined and effective radiation
33     BT=("bt" %in% modeShd)
34     angGen <- fTheta(radHoriz, beta, alfa, modeTrk, betaLim, BT, struct, distances)
35     inclin <- fInclin(radHoriz, angGen, iS, alb, horizBright, HCPV)
36
37     ### Daily, monthly and yearly values
38     by <- radHoriz@sample
39     nms <- c('Bo', 'Bn', 'G', 'D', 'B', 'Gef', 'Def', 'Bef')
40     nmsd <- paste(nms, 'd', sep = '')
41
42     if(radHoriz@type == 'prom'){
43         Gefdm <- inclin[, lapply(.SD/1000, P2E, by),
44             .SDcols = nms,
45             by = .(month(Dates), year(Dates))]
46         names(Gefdm)[-c(1,2)] <- nmsd
47         GefD <- Gefdm[, .SD*1000,

```

```

49         .SDcols = nmsd,
50         by = .(Dates = indexD(radHoriz))]
51
52     Gefdm[, DayOfMonth := DOM(Gefdm)]
53     Gefy <- Gefdm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
54                     .SDcols = nmsd,
55                     by = .(Dates = year)]
56     Gefdm[, DayOfMonth := NULL]
57 } else{
58     GefD <- inclin[, lapply(.SD, P2E, by),
59                     .SDcols = nms,
60                     by = .(Dates = truncDay(Dates))]
61     names(GefD)[-1] <- nmsd
62
63     Gefdm <- GefD[, lapply(.SD/1000, mean, na.rm = TRUE),
64                   .SDcols = nmsd,
65                   by = .(month(indexD(radHoriz)), year(indexD(radHoriz)))]
66     Gefy <- GefD[, lapply(.SD/1000, sum, na.rm = TRUE),
67                   .SDcols = nmsd,
68                   by = .(Dates = year(indexD(radHoriz)))]
69 }
70
71 Gefdm[, Dates := paste(month.abb[month], year, sep = '. ')]
72 Gefdm[, c('month', 'year') := NULL]
73 setcolorder(Gefdm, 'Dates')
74
75 ###Resultado antes de sombras
76 result0=new('Gef',
77             radHoriz,                #Gef contains 'GO'
78             Theta=angGen,
79             GefD=GefD,
80             Gefdm=Gefdm,
81             Gefy=Gefy,
82             GefI=inclin,
83             iS=iS,
84             alb=alb,
85             modeTrk=modeTrk,
86             modeShd=modeShd,
87             angGen=list(alfa=alfa, beta=beta, betaLim=betaLim),
88             struct=struct,
89             distances=distances
90             )
91 ###Shadows
92 if (isTRUE(modeShd == "") ||          #If modeShd==' ' there is no shadow calculation
93     ('bt' %in% modeShd)) {            #nor if there is backtracking
94     return(result0)
95 } else {
96     result <- calcShd(result0, modeTrk, modeShd, struct, distances)
97     return(result)
98 }
99 }

```

prodGCPV

```

1 prodGCPV<-function(lat,
2                     modeTrk='fixed',
3                     modeRad='prom',

```

```

4         dataRad,
5         sample='hour',
6         keep.night=TRUE,
7         sunGeometry='michalsky',
8         corr, f,
9         betaLim=90, beta=abs(lat)-10, alfa=0,
10        iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
11        module=list(),
12        generator=list(),
13        inverter=list(),
14        effSys=list(),
15        modeShd='',
16        struct=list(),
17        distances=data.table(),
18        ...){
19
20    stopifnot(is.list(module),
21              is.list(generator),
22              is.list(inverter),
23              is.list(effSys),
24              is.list(struct),
25              is.data.table(distances))
26
27    if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
28      modeShd[which(modeShd=='bt')]='area'
29      warning('backtracking is only implemented for modeTrk=horiz')}
30
31    if (modeRad!='prev'){ #We do not use a previous calculation
32
33      radEf<-calcGef(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
34                    dataRad=dataRad,
35                    sample=sample, keep.night=keep.night,
36                    sunGeometry=sunGeometry,
37                    corr=corr, f=f,
38                    betaLim=betaLim, beta=beta, alfa=alfa,
39                    iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
40                    modeShd=modeShd, struct=struct, distances=distances, ...)
41
42    } else { #We use a previous calcG0, calcGef or prodGCPV calculation.
43
44      stopifnot(class(dataRad) %in% c('G0', 'Gef', 'ProdGCPV'))
45      radEf <- switch(class(dataRad),
46                      G0=calcGef(lat=lat,
47                                modeTrk=modeTrk, modeRad='prev',
48                                dataRad=dataRad,
49                                betaLim=betaLim, beta=beta, alfa=alfa,
50                                iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
51                                modeShd=modeShd, struct=struct, distances=distances,
52                                ...),
53                      Gef=dataRad,
54                      ProdGCPV=as(dataRad, 'Gef')
55                      )
56    }
57
58    ##Production
59    prodI<-fProd(radEf,module,generator,inverter,effSys)
60    module=attr(prodI, 'module')

```

```

61 generator=attr(prodI, 'generator')
62 inverter=attr(prodI, 'inverter')
63 effSys=attr(prodI, 'effSys')
64
65 ##Calculation of daily, monthly and annual values
66 Pg=generator$Pg #Wp
67
68 by <- radEf@sample
69 nms1 <- c('Pac', 'Pdc')
70 nms2 <- c('Eac', 'Edc', 'Yf')
71
72
73 if(radEf@type == 'prom'){
74   prodDm <- prodI[, lapply(.SD/1000, P2E, by),
75                       .SDcols = nms1,
76                       by = .(month(Dates), year(Dates))]
77   names(prodDm)[-c(1,2)] <- nms2[-3]
78   prodDm[, Yf := Eac/(Pg/1000)]
79   prodD <- prodDm[, .SD*1000,
80                   .SDcols = nms2,
81                   by = .(Dates = indexD(radEf))]
82   prodD[, Yf := Yf/1000]
83
84   prodDm[, DayOfMonth := DOM(prodDm)]
85   prody <- prodDm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
86                   .SDcols = nms2,
87                   by = .(Dates = year)]
88   prodDm[, DayOfMonth := NULL]
89 } else {
90   prodD <- prodI[, lapply(.SD, P2E, by),
91                   .SDcols = nms1,
92                   by = .(Dates = truncDay(Dates))]
93   names(prodD)[-1] <- nms2[-3]
94   prodD[, Yf := Eac/Pg]
95
96   prodDm <- prodD[, lapply(.SD/1000, mean, na.rm = TRUE),
97                   .SDcols = nms2,
98                   by = .(month(Dates), year(Dates))]
99   prodDm[, Yf := Yf * 1000]
100   prody <- prodD[, lapply(.SD/1000, sum, na.rm = TRUE),
101                   .SDcols = nms2,
102                   by = .(Dates = year(Dates))]
103   prody[, Yf := Yf * 1000]
104 }
105
106 prodDm[, Dates := paste(month.abb[month], year, sep = '. ')]
107 prodDm[, c('month', 'year') := NULL]
108 setcolorder(prodDm, 'Dates')
109
110 result <- new('ProdGCPV',
111             radEf,                                #contains 'Gef'
112             prodD=prodD,
113             prodDm=prodDm,
114             prody=prody,
115             prodI=prodI,
116             module=module,
117             generator=generator,
118             inverter=inverter,

```

```

119         effSys=effSys
120     )
121 }

```

prodPVPS

```

1 prodPVPS<-function(lat,
2     modeTrk='fixed',
3     modeRad='prom',
4     dataRad,
5     sample='hour',
6     keep.night=TRUE,
7     sunGeometry='michalsky',
8     corr, f,
9     betaLim=90, beta=abs(lat)-10, alfa=0,
10    iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
11    pump , H,
12    Pg, converter= list(), #Pnom=Pg, Ki=c(0.01,0.025,0.05)),
13    effSys=list(),
14    ...){
15
16    stopifnot(is.list(converter),
17              is.list(effSys))
18
19    if (modeRad!='prev'){ #We do not use a previous calculation
20
21        radEf<-calcGef(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
22                      dataRad=dataRad,
23                      sample=sample, keep.night=keep.night,
24                      sunGeometry=sunGeometry,
25                      corr=corr, f=f,
26                      betaLim=betaLim, beta=beta, alfa=alfa,
27                      iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
28                      modeShd='', ...)
29
30    } else { #We use a previous calculation of calcG0, calcGef or prodPVPS
31        stopifnot(class(dataRad) %in% c('G0', 'Gef', 'ProdPVPS'))
32        radEf <- switch(class(dataRad),
33                      G0=calcGef(lat=lat,
34                                modeTrk=modeTrk, modeRad='prev',
35                                dataRad=dataRad,
36                                betaLim=betaLim, beta=beta, alfa=alfa,
37                                iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
38                                modeShd='', ...),
39                      Gef=dataRad,
40                      ProdPVPS=as(dataRad, 'Gef')
41                      )
42    }
43
44    ###Electric production
45    converter.default=list(Ki = c(0.01,0.025,0.05), Pnom=Pg)
46    converter=modifyList(converter.default, converter)
47
48    effSys.default=list(ModQual=3,ModDisp=2,OhmDC=1.5,OhmAC=1.5,MPP=1,TrafoMT=1,Disp
49    =0.5)
50    effSys=modifyList(effSys.default, effSys)

```

```

51  TONC=47
52  Ct=(TONC-20)/800
53  lambda=0.0045
54  Gef=radEf@GefI$Gef
55  night=radEf@solI$night
56  Ta=radEf@Ta$Ta
57
58  Tc=Ta+Ct*Gef
59  Pdc=Pg*Gef/1000*(1-lambda*(Tc-25))
60  Pdc[is.na(Pdc)]=0 #Necessary for the functions provided by fPump
61  PdcN=with(effSys,
62           Pdc/converter$Pnom*(1-ModQual/100)*(1-ModDisp/100)*(1-OhmDC/100)
63           )
64  PacN=with(converter,{
65           A=Ki[3]
66           B=Ki[2]+1
67           C=Ki[1]-(PdcN)
68           ##AC power normalized to the inverter
69           result=(-B+sqrt(B^2-4*A*C))/(2*A)
70  })
71  PacN[PacN<0]<-0
72
73  Pac=with(converter,
74           PacN*Pnom*(1-effSys$OhmAC/100))
75  Pdc=PdcN*converter$Pnom*(Pac>0)
76
77
78  ###Pump
79  fun<-fPump(pump=pump, H=H)
80  ##I limit power to the pump operating range.
81  rango=with(fun,Pac>=lim[1] & Pac<=lim[2])
82  Pac[!rango]<-0
83  Pdc[!rango]<-0
84  prodI=data.table(Pac=Pac,Pdc=Pdc,Q=0,Pb=0,Ph=0,f=0)
85  prodI=within(prodI,{
86           Q[rango]<-fun$fQ(Pac[rango])
87           Pb[rango]<-fun$fPb(Pac[rango])
88           Ph[rango]<-fun$fPh(Pac[rango])
89           f[rango]<-fun$fFreq(Pac[rango])
90           etam=Pb/Pac
91           etab=Ph/Pb
92  })
93
94  prodI[night,]<-NA
95  prodI[, Dates := indexI(radEf)]
96  setcolorder(prodI, c('Dates', names(prodI)[-length(prodI)]))
97
98  ###daily, monthly and yearly values
99
100  by <- radEf@sample
101
102  if(radEf@type == 'prom'){
103      prodDm <- prodI[, .(Eac = P2E(Pac, by)/1000,
104                          Qd = P2E(Q, by)),
105                      by = .(month(Dates), year(Dates))]
106      prodDm[, Yf := Eac/(Pg/1000)]
107
108      prodD <- prodDm[, .(Eac = Eac*1000,

```



```

109         Qd,
110         Yf),
111         by = .(Dates = indexD(radEf))])
112
113     prodDm[, DayOfMonth := DOM(prodDm)]
114
115     prody <- prodDm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
116                          .SDcols = c('Eac', 'Qd', 'Yf'),
117                          by = .(Dates = year)]
118     prodDm[, DayOfMonth := NULL]
119 } else {
120     prodD <- prodI[, .(Eac = P2E(Pac, by)/1000,
121                       Qd = P2E(Q, by)),
122                     by = .(Dates = truncDay(Dates))])
123     prodD[, Yf := Eac/Pg*1000]
124
125     prodDm <- prodD[, lapply(.SD, mean, na.rm = TRUE),
126                        .SDcols = c('Eac', 'Qd', 'Yf'),
127                        by = .(month(Dates), year(Dates))])
128     prody <- prodD[, lapply(.SD, sum, na.rm = TRUE),
129                      .SDcols = c('Eac', 'Qd', 'Yf'),
130                      by = .(Dates = year(Dates))])
131
132 }
133
134 prodDm[, Dates := paste(month.abb[month], year, sep = '. ')]
135 prodDm[, c('month', 'year') := NULL]
136 setcolorder(prodDm, 'Dates')
137
138 result <- new('ProdPVPS',
139              radEf,                      #contains 'Gef'
140              prodD=prodD,
141              prodDm=prodDm,
142              prody=prody,
143              prodI=prodI,
144              pump=pump,
145              H=H,
146              Pg=Pg,
147              converter=converter,
148              effSys=effSys
149              )
150 }

```

calcShd

```

1 calcShd<-function(radEf, ##class='Gef'
2                   modeTrk='fixed',      #c('two','horiz','fixed')
3                   modeShd='prom',       #modeShd=c('area','bt','prom')
4                   struct=list(), #list(W=23.11, L=9.8, Nrow=2, Ncol=8),
5                   distances=data.frame() #data.table(Lew=40, Lns=30, H=0)){
6
7 {
8     stopifnot(is.list(struct), is.data.frame(distances))
9
10    ##For now I only use modeShd = 'area'
11    ##With different modeShd (to be defined) I will be able to calculate Gef in a
    different way

```

```

12  ##See macagnan thesis
13  prom=("prom" %in% modeShd)
14  prev <- as.data.tableI(radEf, complete=TRUE)
15  ## shadow calculations
16  sol <- data.table(AzS = prev$AzS,
17                  AlS = prev$AlS)
18  theta <- radEf@Theta
19  AngGen <- data.table(theta, sol)
20  FS <- fSombra(AngGen, distances, struct, modeTrk, prom)
21  ## irradiance calculation
22  gef0 <- radEf@GefI
23  Bef0 <- gef0$Bef
24  Dcef0 <- gef0$Dcef
25  Gef0 <- gef0$Gef
26  Dief0 <- gef0$Dief
27  Ref0 <- gef0$Ref
28  ## calculation
29  Bef <- Bef0*(1-FS)
30  Dcef <- Dcef0*(1-FS)
31  Def <- Dief0+Dcef
32  Gef <- Dief0+Ref0+Bef+Dcef #Including shadows
33  ##Change names
34  nms <- c('Gef', 'Def', 'Dcef', 'Bef')
35  nmsIndex <- which(names(gef0) %in% nms)
36  names(gef0)[nmsIndex] <- paste(names(gef0)[nmsIndex], '0', sep='')
37  GefShd <- gef0
38  GefShd[, c(nms, 'FS') := .(Gef, Def, Dcef, Bef, FS)]
39
40  ## daily, monthly and yearly values
41  by <- radEf@sample
42  nms <- c('Gef0', 'Def0', 'Bef0', 'G', 'D', 'B', 'Gef', 'Def', 'Bef')
43  nmsd <- paste(nms, 'd', sep = '')
44
45  Gefdm <- GefShd[, lapply(.SD/1000, P2E, by),
46                  by = .(month(truncDay(Dates)), year(truncDay(Dates))),
47                  .SDcols = nms]
48  names(Gefdm)[-c(1, 2)] <- nmsd
49
50  if(radEf@type == 'prom'){
51    GefD <- Gefdm[, .SD[, -c(1, 2)] * 1000,
52                  .SDcols = nmsd,
53                  by = .(Dates = indexD(radEf))]
54
55    Gefdm[, DayOfMonth := DOM(Gefdm)]
56
57    Gefy <- Gefdm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
58                  .SDcols = nmsd,
59                  by = .(Dates = year)]
60    Gefdm[, DayOfMonth := NULL]
61  } else{
62    GefD <- GefShd[, lapply(.SD/1000, P2E, by),
63                  .SDcols = nms,
64                  by = .(Dates = truncDay(Dates))]
65    names(GefD)[-1] <- nmsd
66
67    Gefy <- GefD[, lapply(.SD[, -1], sum, na.rm = TRUE),
68                  .SDcols = nmsd,
69                  by = .(Dates = year(Dates))]

```

```

70 }
71
72 Gefdm[, Dates := paste(month.abb[month], year, sep = '. ')]
73 Gefdm[, c('month', 'year') := NULL]
74 setcolorder(Gefdm, c('Dates', names(Gefdm)[-length(Gefdm)]))
75
76 ## Object of class Gef
77 ## modifying the 'modeShd', 'GefI', 'GefD', 'Gefdm', and 'Gefy' slots
78 ## from the original radEf object
79 radEf@modeShd=modeShd
80 radEf@GefI=GefShd
81 radEf@GefD=GefD
82 radEf@Gefdm=Gefdm
83 radEf@Gefy=Gefy
84 return(radEf)
85 }

```

optimShd

```

1  optimShd<-function(lat,
2      modeTrk='fixed',
3      modeRad='prom',
4      dataRad,
5      sample='hour',
6      keep.night=TRUE,
7      sunGeometry='michalsky',
8      betaLim=90, beta=abs(lat)-10, alfa=0,
9      iS=2, alb=0.2, HCPV=FALSE,
10     module=list(),
11     generator=list(),
12     inverter=list(),
13     effSys=list(),
14     modeShd='',
15     struct=list(),
16     distances=data.table(),
17     res=2,      #resolution, distance spacing
18     prog=TRUE){ #Drawing progress bar
19
20     if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
21         modeShd[which(modeShd=='bt')]='area'
22         warning('backtracking is only implemented for modeTrk=horiz')}
23
24     ##I save function arguments for later use
25
26     listArgs<-list(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
27         dataRad=dataRad,
28         sample=sample, keep.night=keep.night,
29         sunGeometry=sunGeometry,
30         betaLim=betaLim, beta=beta, alfa=alfa,
31         iS=iS, alb=alb, HCPV=HCPV,
32         module=module, generator=generator,
33         inverter=inverter, effSys=effSys,
34         modeShd=modeShd, struct=struct,
35         distances=data.table(Lew=NA, Lns=NA, D=NA))
36
37
38     ##I think network on which I will do the calculations

```

```

39 Red=switch(modeTrk,
40     horiz=with(distances,
41         data.table(Lew=seq(Lew[1],Lew[2],by=res),
42             H=0)),
43     two=with(distances,
44         data.table(
45             expand.grid(Lew=seq(Lew[1],Lew[2],by=res),
46                 Lns=seq(Lns[1],Lns[2],by=res),
47                 H=0))),
48     fixed=with(distances,
49         data.table(D=seq(D[1],D[2],by=res),
50             H=0))
51 )
52
53 casos<-dim(Red)[1] #Number of possibilities to study
54
55 ##I prepare the progress bar
56 if (prog) {pb <- txtProgressBar(min = 0, max = casos+1, style = 3)
57     setTxtProgressBar(pb, 0)}
58
59 ###Calculations
60 ##Reference: No shadows
61 listArgs0 <- modifyList(listArgs,
62     list(modeShd='', struct=NULL, distances=NULL) )
63 Prod0<-do.call(prodGCPV, listArgs0)
64 YfAnual0=mean(Prod0@prody$Yf) #I use mean in case there are several years
65 if (prog) {setTxtProgressBar(pb, 1)}
66
67 ##The loop begins
68
69 ##I create an empty vector of the same length as the cases to be studied
70 YfAnual<-numeric(casos)
71
72 BT=('bt' %in% modeShd)
73 if (BT) { ##There is backtracking, then I must start from horizontal radiation.
74     RadBT <- as(Prod0, 'G0')
75     for (i in seq_len(casos)){
76         listArgsBT <- modifyList(listArgs,
77             list(modeRad='prev', dataRad=RadBT,
78                 distances=Red[i,]))
79         prod.i <- do.call(prodGCPV, listArgsBT)
80         YfAnual[i]=mean(prod.i@prody$Yf)
81         if (prog) {setTxtProgressBar(pb, i+1)}
82     }
83 } else {
84     prom=('prom' %in% modeShd)
85     for (i in seq_len(casos)){
86         Gef0=as(Prod0, 'Gef')
87         GefShd=calcShd(Gef0, modeTrk=modeTrk, modeShd=modeShd,
88             struct=struct, distances=Red[i,])
89         listArgsShd <- modifyList(listArgs,
90             list(modeRad='prev', dataRad=GefShd)
91             )
92         prod.i <- do.call(prodGCPV, listArgsShd)
93         YfAnual[i]=mean(prod.i@prody$Yf)
94         if (prog) {setTxtProgressBar(pb, i+1)}
95     }
96 }

```



```

23         fill = TRUE, dec = '.', sep = ';',
24         dates.col = 'Dates', ta.col = 'Ta',
25         g0.col = 'G0', keep.cols = FALSE)
26 {
27     #stops if the arguments are not characters or numerics
28     stopifnot(is.character(dates.col) || is.numeric(dates.col))
29     stopifnot(is.character(ta.col) || is.numeric(ta.col))
30     stopifnot(is.character(g0.col) || is.numeric(g0.col))
31
32     #read from file and set it in a data.table
33     bd <- fread(file, header = header, fill = fill, dec = dec, sep = sep)
34
35     #check the columns
36     if(!(dates.col %in% names(bd))) stop(paste('The column', dates.col, 'is not in
the file'))
37     if(!(g0.col %in% names(bd))) stop(paste('The column', g0.col, 'is not in the file
'))
38     if(!(ta.col %in% names(bd))) stop(paste('The column', ta.col, 'is not in the file
'))
39
40     #name the dates column by Dates
41     Dates <- bd[[dates.col]]
42     bd[, (dates.col) := NULL]
43     bd[, Dates := as.IDate(Dates, format = format)]
44
45     #name the g0 column by G0
46     G0 <- bd[[g0.col]]
47     bd[, (g0.col) := NULL]
48     bd[, G0 := as.numeric(G0)]
49
50     #name the ta column by Ta
51     Ta <- bd[[ta.col]]
52     bd[, (ta.col) := NULL]
53     bd[, Ta := as.numeric(Ta)]
54
55     names0 <- NULL
56     if(all(c('D0', 'B0') %in% names(bd))){
57         names0 <- c(names0, 'D0', 'B0')
58     }
59
60     names0 <- c(names0, 'Ta')
61
62     if(all(c('TempMin', 'TempMax') %in% names(bd))){
63         names0 <- c(names0, 'TempMin', 'TempMax')
64     }
65     if(keep.cols)
66     {
67         #keep the rest of the columns but reorder the columns
68         setcolorder(bd, c('Dates', 'G0', names0))
69     }
70     else
71     {
72         #erase the rest of the columns
73         cols <- c('Dates', 'G0', names0)
74         bd <- bd[, ..cols]
75     }
76
77     setkey(bd, 'Dates')

```

```

78   result <- new(Class = 'Meteo',
79                 latm = lat,
80                 data = bd,
81                 type = 'bd',
82                 source = file)
83 }
84
85 ##### file to Meteo (intradaily) #####
86 readBDi <- function(file, lat,
87                     format = "%d/%m/%Y %H:%M:%S",
88                     header = TRUE, fill = TRUE, dec = '.',
89                     sep = ';', dates.col = 'dates', times.col,
90                     ta.col = 'Ta', g0.col = 'G0', keep.cols = FALSE)
91 {
92   #stops if the arguments are not characters or numerics
93   stopifnot(is.character(dates.col) || is.numeric(dates.col))
94   stopifnot(is.character(ta.col) || is.numeric(ta.col))
95   stopifnot(is.character(g0.col) || is.numeric(g0.col))
96
97   #read from file and set it in a data.table
98   bd <- fread(file, header = header, fill = fill, dec = dec, sep = sep)
99
100  #check the columns
101  if(!(dates.col %in% names(bd))) stop(paste('The column', dates.col, 'is not in
the file'))
102  if(!(g0.col %in% names(bd))) stop(paste('The column', g0.col, 'is not in the file
'))
103  if(!(ta.col %in% names(bd))) stop(paste('The column', ta.col, 'is not in the file
'))
104
105  if(!missing(times.col)){
106    stopifnot(is.character(times.col) || is.numeric(times.col))
107    if(!(times.col %in% names(bd))) stop(paste('The column', times.col, 'is not
in the file'))
108
109    #name the dates column by Dates
110    format <- strsplit(format, ' ')
111    dd <- as.IDate(bd[[dates.col]], format = format[[1]][1])
112    tt <- as.ITime(bd[[times.col]], format = format[[1]][2])
113    bd[, (dates.col) := NULL]
114    bd[, (times.col) := NULL]
115    bd[, Dates := as.POSIXct(dd, tt, tz = 'UTC')]
116  }
117
118  else
119  {
120    dd <- as.POSIXct(bd[[dates.col]], format = format, tz = 'UTC')
121    bd[, (dates.col) := NULL]
122    bd[, Dates := dd]
123  }
124
125  #name the g0 column by G0
126  G0 <- bd[[g0.col]]
127  bd[, (g0.col) := NULL]
128  bd[, G0 := as.numeric(G0)]
129
130  #name the ta column by Ta
131  Ta <- bd[[ta.col]]

```

```

132   bd[, (ta.col) := NULL]
133   bd[, Ta := as.numeric(Ta)]
134
135   names0 <- NULL
136   if(all(c('D0', 'B0') %in% names(bd))){
137     names0 <- c(names0, 'D0', 'B0')
138   }
139
140   names0 <- c(names0, 'Ta')
141
142   if(keep.cols)
143   {
144     #keep the rest of the columns but reorder the columns
145     setcolorder(bd, c('Dates', 'G0', names0))
146   }
147   else
148   {
149     #erase the rest of the columns
150     cols <- c('Dates', 'G0', names0)
151     bd <- bd[, ..cols]
152   }
153
154   setkey(bd, 'Dates')
155   result <- new(Class = 'Meteo',
156                 latm = lat,
157                 data = bd,
158                 type = 'bdI',
159                 source = file)
160 }
161
162
163 dt2Meteo <- function(file, lat, source = '', type){
164   ## Make sure its a data.table
165   bd <- data.table(file)
166
167   ## Dates is an as.POSIX element
168   bd[, Dates := as.POSIXct(Dates, tz = 'UTC')]
169
170   ## type
171   if(missing(type)){
172     sample <- median(diff(file$Dates))
173     IsDaily <- as.numeric(sample, units = 'days')
174     if(is.na(IsDaily)) IsDaily <- ifelse('G0d' %in% names(bd),
175                                         1, 0)
176     if(IsDaily >= 30) type <- 'prom'
177     else{
178       type <- ifelse(IsDaily >= 1, 'bd', 'bdI')
179     }
180   }
181
182   if(!('Ta' %in% names(bd))){
183     if(all(c('Tempmin', 'TempMax') %in% names(bd))){
184       bd[, Ta := mean(c(Tempmin, TempMax))]
185     } else bd[, Ta := 25]
186   }
187
188   ## Columns of the data.table
189   nms0 <- switch(type,

```



```

190         bd = ,
191         prom = {
192             nms0 <- 'G0d'
193             if(all(c('D0d', 'B0d') %in% names(bd))){
194                 nms0 <- c(nms0, 'D0d', 'B0d')
195             }
196             nms0 <- c(nms0, 'Ta')
197             if(all(c('TempMin', 'TempMax') %in% names(bd))){
198                 nms0 <- c(nms0, 'TempMin', 'TempMax')
199             }
200             nms0
201         },
202         bdI = {
203             nms0 <- 'G0'
204             if(all(c('D0', 'B0') %in% names(bd))){
205                 nms0 <- c(nms0, 'D0', 'B0')
206             }
207             if('Ta' %in% names(bd)){
208                 nms0 <- c(nms0, 'Ta')
209             }
210             nms0
211         })
212     ## Columns order and set key
213     setcolorder(bd, c('Dates', nms0))
214     setkey(bd, 'Dates')
215     ## Result
216     result <- new(Class = 'Meteo',
217                   latm = lat,
218                   data = bd,
219                   type = type,
220                   source = source)
221 }
222
223 ##### Liu and Jordan, Collares-Pereira and Rabl proposals #####
224 collper <- function(sol, compD)
225 {
226     Dates <- indexI(sol)
227     x <- as.Date(Dates)
228     ind.rep <- cumsum(c(1, diff(x) != 0))
229     solI <- as.data.tableI(sol, complete = T)
230     ws <- solI$ws
231     w <- solI$w
232
233     a <- 0.409-0.5016*sin(ws+pi/3)
234     b <- 0.6609+0.4767*sin(ws+pi/3)
235
236     rd <- solI[, Bo0/Bo0d]
237     rg <- rd * (a + b * cos(w))
238
239     # Daily irradiation components
240     G0d <- compD$G0d[ind.rep]
241     B0d <- compD$B0d[ind.rep]
242     D0d <- compD$D0d[ind.rep]
243
244     # Daily profile
245     G0 <- G0d * rg
246     D0 <- D0d * rd
247

```

```

248 # This method may produce diffuse irradiance higher than
249 # global irradiance
250 GO <- pmax(GO, DO, na.rm = TRUE)
251 BO <- GO - DO
252
253 # Negative values are set to NA
254 neg <- (BO < 0) | (DO < 0) | (GO < 0)
255 is.na(GO) <- neg
256 is.na(BO) <- neg
257 is.na(DO) <- neg
258
259 # Daily profiles are scaled to keep daily irradiation values
260 day <- truncDay(indexI(sol))
261 sample <- sol@sample
262
263 G0dCP <- ave(GO, day, FUN=function(x) P2E(x, sample))
264 B0dCP <- ave(BO, day, FUN=function(x) P2E(x, sample))
265 D0dCP <- ave(DO, day, FUN=function(x) P2E(x, sample))
266
267 GO <- GO * G0d/G0dCP
268 BO <- BO * B0d/B0dCP
269 DO <- DO * D0d/D0dCP
270
271 res <- data.table(GO, BO, DO)
272 return(res)
273 }
274
275
276 ##### intradaily Meteo to daily Meteo #####
277 MeteoI2Meteod <- function(G0i)
278 {
279   lat <- G0i@latm
280   source <- G0i@source
281
282   dt0 <- getData(G0i)
283   dt <- dt0[, lapply(.SD, sum),
284     .SDcols = names(dt0)[!names(dt0) %in% c('Dates', 'Ta')],
285     by = .(Dates = as.IDate(Dates))]
286   if('Ta' %in% names(dt0)){
287     Ta <- dt0[, .(Ta = mean(Ta),
288       TempMin = min(Ta),
289       TempMax = max(Ta)),
290     by = .(Dates = as.IDate(Dates))]
291     if(all(Ta$Ta == c(Ta$TempMin, Ta$TempMax))) Ta[, c('TempMin', 'TempMax') :=
292       NULL]
293     dt <- merge(dt, Ta)
294   }
295   if('GO' %in% names(dt)){
296     names(dt)[names(dt) == 'GO'] <- 'G0d'
297   }
298   if('DO' %in% names(dt)){
299     names(dt)[names(dt) == 'DO'] <- 'D0d'
300   }
301   if('BO' %in% names(dt)){
302     names(dt)[names(dt) == 'BO'] <- 'B0d'
303   }
304   G0d <- dt2Meteo(dt, lat, source, type = 'bd')
305   return(G0d)

```

```

305 }
306
307 ##### daily Meteo to monthly Meteo #####
308 Meteod2Meteom <- function(G0d)
309 {
310     lat <- G0d@latm
311     source <- G0d@source
312
313     dt <- getData(G0d)
314     nms <- names(dt)[-1]
315     dt <- dt[, lapply(.SD, mean),
316                     .SDcols = nms,
317                     by = .(month(Dates), year(Dates))]
318     dt[, Dates := fBTd()]
319     dt <- dt[, c('month', 'year') := NULL]
320
321     setcolororder(dt, 'Dates')
322
323     G0m <- dt2Meteo(dt, lat, source, type = 'prom')
324     return(G0m)
325 }
326
327 zoo2Meteo <- function(file, lat, source = '')
328 {
329     sample <- median(diff(index(file)))
330     IsDaily <- as.numeric(sample, units = 'days')>=1
331     type <- ifelse(IsDaily, 'bd', 'bdI')
332     result <- new(Class = 'Meteo',
333                   latm = lat,
334                   data = file,
335                   type = type,
336                   source = source)
337 }
338
339 siarGET <- function(id, inicio, final, tipo = 'Mensuales', ambito = 'Estacion'){
340     if(!(tipo %in% c('Horarios', 'Diarios', 'Semanales', 'Mensuales'))){
341         stop('argument \'tipo\' must be: Horarios, Diarios, Semanales or Mensuales')
342     }
343     if(!(ambito %in% c('CCAA', 'Provincia', 'Estacion'))){
344         stop('argument \'ambito\' must be: CCAA, Provincia or Estacion')
345     }
346
347     mainURL <- "https://servicio.mapama.gob.es"
348
349     path <- paste('/apisiar/API/v1/Datos', tipo, ambito, sep = '/')
350
351     ## prepare the APIsiar
352     req <- request(mainURL) |>
353         req_url_path(path) |>
354         req_url_query(Id = id,
355                       FechaInicial = inicio,
356                       FechaFinal = final,
357                       ClaveAPI = '_Q8L_niYFBBmBs-vB3UomUqdUYy98FTRX1aYbrZ8n2FXuHYGTV')
358     ## execute it
359     resp <- req_perform(req)
360
361     ##JSON to R
362     respJSON <- resp_body_json(resp, simplifyVector = TRUE)

```

```

363
364   if(!is.null(respJSON$MensajeRespuesta)){
365       stop(respJSON$MensajeRespuesta)
366   }
367
368   res0 <- data.table(respJSON$Datos)
369
370   res <- switch(tipo,
371       Horarios = {
372           res0[, HoraMin := as.ITime(sprintf('%04d', HoraMin),
373               format = '%H%M')]
374           res0[, Fecha := as.IDate(Fecha, format = '%Y-%m-%d')]
375           res0[, Fecha := as.IDate(ifelse(HoraMin == as.ITime(0),
376               Fecha+1, Fecha))]
377           res0[, Dates := as.POSIXct(HoraMin, Fecha,
378               tz = 'Europe/Madrid')]
379           res0 <- res0[, .(Dates,
380               GO = Radiacion,
381               Ta = TempMedia)]
382           return(res0)
383       },
384       Diarios = {
385           res0[, Dates := as.IDate(Fecha)]
386           res0 <- res0[, .(Dates,
387               GOd = Radiacion * 277.78,
388               Ta = TempMedia,
389               TempMin,
390               TempMax)]
391           return(res0)
392       },
393       Semanales = res0,
394       Mensuales = {
395           promDays<-c(17,14,15,15,15,10,18,18,18,19,18,13)
396           names(res0)[1] <- 'Year'
397           res0[, Dates := as.IDate(paste(Year, Mes,
398               promDays[Mes],
399               sep = '-'))]
400           res0 <- res0[, .(Dates,
401               GOd = Radiacion * 277.78,
402               Ta = TempMedia,
403               TempMin,
404               TempMax)]
405       })
406
407   return(res)
408 }
409
410 haversine <- function(lat1, lon1, lat2, lon2) {
411     R <- 6371 # Radius of the Earth in kilometers
412     dLat <- (lat2 - lat1) * pi / 180
413     dLon <- (lon2 - lon1) * pi / 180
414     a <- sin(dLat / 2) * sin(dLat / 2) + cos(lat1 * pi / 180) *
415         cos(lat2 * pi / 180) * sin(dLon / 2) * sin(dLon / 2)
416     c <- 2 * atan2(sqrt(a), sqrt(1 - a))
417     d <- R * c
418     return(d)
419 }
420

```

```

421 readSIAR <- function(Lon = 0, Lat = 0,
422                       inicio = paste(year(Sys.Date())-1, '01-01', sep = '-'),
423                       final = paste(year(Sys.Date())-1, '12-31', sep = '-'),
424                       tipo = 'Mensuales', n_est = 3){
425   inicio <- as.Date(inicio)
426   final <- as.Date(final)
427
428   n_reg <- switch(tipo,
429     Horarios = {
430       tt <- difftime(final, inicio, units = 'days')
431       tt <- (as.numeric(tt)+1)*48
432       tt <- tt*n_est
433       tt
434     },
435     Diarios = {
436       tt <- difftime(final, inicio, units = 'days')
437       tt <- as.numeric(tt)+1
438       tt <- tt*n_est
439       tt
440     },
441     Semanales = {
442       tt <- difftime(final, inicio, units = 'weeks')
443       tt <- as.numeric(tt)
444       tt <- tt*n_est
445       tt
446     },
447     Mensuales = {
448       tt <- difftime(final, inicio, units = 'weeks')
449       tt <- as.numeric(tt)/4.34524
450       tt <- ceiling(tt)
451       tt <- tt*n_est
452       tt
453     })
454   if(n_reg > 100) stop(paste('Number of requested records (', n_reg,
455                             ') exceeds the maximum allowed (100)', sep = ''))
456   ## Obtain the nearest stations
457   siar <- est_SIAR[
458     Fecha_Instalacion <= final & (is.na(Fecha_Baja) | Fecha_Baja >= inicio)
459   ]
460
461   ## Weights for the interpolation
462   siar[, dist := haversine(Latitud, Longitud, Lat, Lon)]
463   siar <- siar[order(dist)][1:n_est]
464   siar[, peso := 1/dist]
465   siar[, peso := peso/sum(peso)]
466   ## Is the given location within the polygon formed by the stations?
467   siar <- siar[, .(Estacion,Codigo, dist, peso)]
468
469   ## List for the data.tables of siarGET
470   siar_list <- list()
471   for(codigo in siar$Codigo){
472     siar_list[[codigo]] <- siarGET(id = codigo,
473                                   inicio = as.character(inicio),
474                                   final = as.character(final),
475                                   tipo = tipo)
476     siar_list[[codigo]]$peso <- siar[Codigo == codigo, peso]
477   }
478

```

```

479   ## Bind the data.tables
480   s_comb <- rbindlist(siar_list, use.names = TRUE, fill = TRUE)
481
482   nms <- names(s_comb)
483   nms <- nms[-c(1, length(nms))]
484
485   ## Interpole
486   res <- s_comb[, lapply(.SD * peso, sum, na.rm = TRUE),
487                     .SDcols = nms,
488                     by = Dates]
489
490   ## Source
491   mainURL <- "https://servicio.mapama.gob.es"
492   Estaciones <- siar[, paste(Estacion, '(',Codigo, ')', sep = '')]
493   Estaciones <- paste(Estaciones, collapse = ', ')
494   source <- paste(mainURL, '\n -Estaciones:', Estaciones, sep = ' ')
495
496   res <- switch(tipo,
497                 Horarios = {dt2Meteo(res, lat = Lat, source = mainURL, type = 'bdI')}
498                 },
499                 Diarios = {dt2Meteo(res, lat = Lat, source = mainURL, type = 'bd')},
500                 Semanales = {res},
501                 Mensuales = {dt2Meteo(res, lat = Lat, source = source, type = 'prom')}
502   ))
503   return(res)
504 }

```

A.2. Clases

Sol

```

1  setClass(
2    Class='Sol', ##Solar angles
3    slots = c(
4      lat='numeric',#latitud in degrees, >0 if North
5      sold='data.table',#daily angles
6      soli='data.table',#intradaily angles
7      sample='character',#sample of time
8      method='character'#method used for geometry calculations
9    ),
10   validity=function(object) {return(TRUE)}
11 )

```

Meteo

```

1  setClass(
2    Class = 'Meteo', ##radiation and temperature data
3    slots = c(
4      latm='numeric',#latitud in degrees, >0 if North
5      data='data.table',#data, including G (Wh/m2) and Ta (°C)
6      type='character',#choose between 'prom', 'bd' and 'bdI'
7      source='character'#origin of the data
8    ),
9   validity=function(object) {return(TRUE)}
10 )

```

G0

```

1  setClass(
2    Class = 'G0',
3    slots = c(
4      GOD = 'data.table', #result of fCompD
5      G0dm = 'data.table', #monthly means
6      G0y = 'data.table', #yearly values
7      G0I = 'data.table', #result of fCompI
8      Ta = 'data.table'   #Ambient temperature
9    ),
10   contains = c('Sol', 'Meteo'),
11   validity = function(object) {return(TRUE)}
12 )
13

```

Gef

```

1  setClass(
2    Class='Gef',
3    slots = c(
4      GefD='data.table', #daily values
5      Gefdm='data.table', #monthly means
6      Gefy='data.table', #yearly values
7      GefI='data.table', #result of fInclin
8      Theta='data.table', #result of fTheta
9      iS='numeric',      #dirt index
10     alb='numeric',      #albedo
11     modeTrk='character', #tracking mode
12     modeShd='character', #shadow mode
13     angGen='list',       #includes alpha, beta and betaLim
14     struct='list',       #structure dimensions
15     distances='data.frame' #distances between structures
16   ),
17   contains='G0',
18   validity=function(object) {return(TRUE)}
19 )

```

ProdGCPV

```

1  setClass(
2    Class='ProdGCPV',
3    slots = c(
4      prodD='data.table', #daily values
5      prodDm='data.table', #monthly means
6      prody='data.table', #yearly values
7      prodI='data.table', #results of fProd
8      module='list',      #module characteristics
9      generator='list',    #generator characteristics
10     inverter='list',      #inverter characteristics
11     effSys='list'         #efficiency values of the system
12   ),
13   contains='Gef',
14   validity=function(object) {return(TRUE)}
15 )

```

ProdPVPS

```

1 setClass(
2   Class='ProdPVPS',
3   slots = c(
4     prodD='data.table', #daily values
5     prodDm='data.table', #monthly means
6     prody='data.table', #yearly values
7     prodI='data.table', #results of fPump
8     Pg='numeric',       #generator power
9     H='numeric',        #manometric head
10    pump='list',         #parameters of the pump
11    converter='list',    #inverter characteristics
12    effSys='list'        #efficiency values of the system
13  ),
14  contains='Gef',
15  validity=function(object) {return(TRUE)}
16 )

```

Shade

```

1 setClass(
2   Class='Shade',
3   slots = c(
4     FS='numeric', #shadows factor values
5     GRR='numeric', #Ground Requirement Ratio
6     Yf='numeric', #final productivity
7     FS.loess='loess', #local fitting of FS with loess
8     Yf.loess='loess', #local fitting of Yf with loess
9     modeShd='character', #mode of shadow
10    struct='list',        #dimensions of the structures
11    distances='data.frame', #distances between structures
12    res='numeric'         #difference between the different steps of the
    calculations
13  ),
14  contains='ProdGCPV', ##Resultado de prodGCPV sin sombras (Prod0)
15  validity=function(object) {return(TRUE)}
16 )

```

A.3. Funciones

corrFdKt

```

1 ##### monthly Kt #####
2 Ktm <- function(sol, G0dm){
3   solf <- sol@solD[, .(Dates, Bo0d)]
4   solf[, c('month', 'year') := .(month(Dates), year(Dates))]
5   solf[, Bo0m := mean(Bo0d), by = .(month, year)]
6   G0df <- G0dm@data[, .(Dates, G0d)]
7   G0df[, c('month', 'year') := .(month(Dates), year(Dates))]
8   G0df[, G0d := mean(G0d), by = .(month, year)]
9   Ktm <- G0df$G0d/solf$Bo0m
10  return(Ktm)
11 }
12
13 ##### daily Kt #####

```



```

14 Ktd <- function(sol, G0d){
15   Bo0d <- sol@solD$Bo0d
16   G0d <- getG0(G0d)
17   Ktd <- G0d/Bo0d
18   return(Ktd)
19 }
20
21 ### intradaily
22 Kti <- function(sol, G0i){
23   Bo0 <- sol@solI$Bo0
24   G0i <- getG0(G0i)
25   Kti <- G0i/Bo0
26   return(Kti)
27 }
28
29
30 ##### monthly correlations #####
31
32 ### Page ###
33 FdKtPage <- function(sol, G0dm){
34   Kt <- Ktm(sol, G0dm)
35   Fd=1-1.13*Kt
36   return(data.table(Fd, Kt))
37 }
38
39 ### Liu and Jordan ###
40 FdKtLJ <- function(sol, G0dm){
41   Kt <- Ktm(sol, G0dm)
42   Fd=(Kt<0.3)*0.595774 +
43     (Kt>=0.3 & Kt<=0.7)*(1.39-4.027*Kt+5.531*Kt^2-3.108*Kt^3)+
44     (Kt>0.7)*0.215246
45   return(data.table(Fd, Kt))
46 }
47
48
49 ##### daily correlations #####
50
51 ### Collares-Pereira and Rabl
52 FdKtCPR <- function(sol, G0d){
53   Kt <- Ktd(sol, G0d)
54   Fd=(0.99*(Kt<=0.17))+(Kt>0.17 & Kt<0.8)*
55     (1.188-2.272*Kt+9.473*Kt^2-21.856*Kt^3+14.648*Kt^4)+
56     (Kt>=0.8)*0.2426688
57   return(data.table(Fd, Kt))
58 }
59
60 ### Erbs, Klein and Duffie ###
61 FdKtEKDd <- function(sol, G0d){
62   ws <- sol@solD$ws
63   Kt <- Ktd(sol, G0d)
64
65   WS1=(abs(ws)<1.4208)
66   Fd=WS1*((Kt<0.715)*(1-0.2727*Kt+2.4495*Kt^2-11.9514*Kt^3+9.3879*Kt^4)+
67     (Kt>=0.715)*(0.143))+
68     !WS1*((Kt<0.722)*(1+0.2832*Kt-2.5557*Kt^2+0.8448*Kt^3)+
69     (Kt>=0.722)*(0.175))
70   return(data.table(Fd, Kt))
71 }

```

```

72
73 ### CLIMED1 ###
74 FdKtCLIMEDd <- function(sol, G0d){
75   Kt <- Ktd(sol, G0d)
76   Fd=(Kt<=0.13)*(0.952)+
77     (Kt>0.13 & Kt<=0.8)*(0.868+1.335*Kt-5.782*Kt^2+3.721*Kt^3)+
78     (Kt>0.8)*0.141
79   return(data.table(Fd, Kt))
80 }
81
82 ##### intradaily correlations #####
83
84 ### intradaily EKD ###
85 FdKtEKDh <- function(sol, G0i){
86   Kt <- Kti(sol, G0i)
87   Fd=(Kt<=0.22)*(1-0.09*Kt)+
88     (Kt>0.22 & Kt<=0.8)*(0.9511-0.1604*Kt+4.388*Kt^2-16.638*Kt^3+12.336*Kt^4)+
89     (Kt>0.8)*0.165
90   return(data.table(Fd, Kt))
91 }
92
93 ### intradaily CLIMED
94 FdKtCLIMEDh <- function(sol, G0i){
95   Kt <- Kti(sol, G0i)
96   Fd=(Kt<=0.21)*(0.995-0.081*Kt)+
97     (Kt>0.21 & Kt<=0.76)*(0.724+2.738*Kt-8.32*Kt^2+4.967*Kt^3)+
98     (Kt>0.76)*0.180
99   return(data.table(Fd, Kt))
100 }
101
102 ### intradaily Boland, Ridley and Lauret ###
103 FdKtBRL <- function(sol, G0i){
104   Kt <- Kti(sol, G0i)
105   sample <- sol@sample
106
107   solI <- as.data.tableI(sol, complete = TRUE)
108   w <- solI$w
109   night <- solI$night
110   AlS <- solI$AlS
111
112   G0d <- Meteoi2Meteod(G0i)
113   ktd <- Ktd(sol, G0d)
114
115   ##persistence
116   pers <- persistence(sol, ktd)
117
118   ##indexRep for ktd and pers
119   Dates <- indexI(sol)
120   x <- as.Date(Dates)
121   ind.rep <- cumsum(c(1, diff(x) != 0))
122   ktd <- ktd[ind.rep]
123   pers <- pers[ind.rep]
124
125   ##fd calculation
126   Fd=(1+exp(-5.38+6.63*Kt+0.006*r2h(w)-0.007*r2d(AlS)+1.75*ktd+1.31*pers))^-1)
127
128   return(data.table(Fd, Kt))
129 }

```

```

130 persistence <- function(sol, Ktd){
131   kt <- data.table(indexD(sol), Ktd)
132   ktNA <- na.omit(kt)
133   iDay <- truncDay(ktNA[[1]])
134
135   x <- rle(as.numeric(iDay))$lengths
136   xLast <- cumsum(x)
137
138   lag1 <- shift(ktNA$Ktd, -1, fill = NA)
139   for (i in xLast){
140     if ((i-1) != 0){lag1[i] <- ktNA$Ktd[i-1]}
141   }
142
143   lag2 <- shift(ktNA$Ktd, 1, fill = NA)
144   for (i in xLast){
145     if ((i+1) <= length(ktNA$Ktd)){lag2[i] <- ktNA$Ktd[i+1]}
146   }
147   pers <- data.table(lag1, lag2)
148   pers[, mean := 1/2 * (lag1+lag2)]
149   pers[, mean]
150 }
151

```

fBTd

```

1 fBTd<-function(mode='prom',
2   year= as.POSIXlt(Sys.Date())$year+1900,
3   start=paste('01-01-',year,sep=''),
4   end=paste('31-12-',year,sep=''),
5   format='%d-%m-%Y'){
6   promDays<-c(17,14,15,15,15,10,18,18,18,19,18,13)
7   BTd=switch(mode,
8     serie={
9       start.<-as.POSIXct(start, format=format, tz='UTC')
10      end.<-as.POSIXct(end, format=format, tz='UTC')
11      res<-seq(start., end., by="1 day")
12    },
13    prom=as.POSIXct(paste(year, 1:12, promDays, sep='-'), tz='UTC')
14  )
15   BTd
16 }

```

fBTi

```

1 intervalo <- function(day, sample){
2   intervalo <- seq.POSIXt(from = as.POSIXct(paste(day, '00:00:00'), tz = 'UTC'),
3     to = as.POSIXct(paste(day, '23:59:59'), tz = 'UTC'),
4     by = sample)
5   return(intervalo)
6 }
7
8 fBTi <- function(d, sample = 'hour'){
9   BTi <- lapply(d, intervalo, sample)
10  BTi <- do.call(c, BTi)
11  return(BTi)
12 }

```

fCompD

```

1 fCompD <- function(sol, G0d, corr = 'CPR', f)
2 {
3   if(!(corr %in% c('CPR', 'Page', 'LJ', 'EKDd', 'CLIMEDd', 'user', 'none'))){
4     warning('Wrong descriptor of correlation Fd-Ktd. Set CPR.')
5     corr <- 'CPR'
6   }
7   if(class(sol)[1] != 'Sol'){
8     sol <- sol[, calcSol(lat = unique(lat), BTi = Dates)]
9   }
10  if(class(G0d)[1] != 'Meteo'){
11    dt <- copy(data.table(G0d))
12    if(!('Dates' %in% names(dt))){
13      dt[, Dates := indexD(sol)]
14      setcolorder(dt, 'Dates')
15      setkey(dt, 'Dates')
16    }
17    if('lat' %in% names(dt)){
18      latg <- unique(dt$lat)
19      dt[, lat := NULL]
20    }else{latg <- getLat(sol)}
21    G0d <- dt2Meteo(dt, latg)
22  }
23
24  stopifnot(indexD(sol) == indexD(G0d))
25  Bo0d <- sol@solD$Bo0d
26  G0 <- getData(G0d)$G0
27
28  is.na(G0) <- (G0>Bo0d)
29
30  ### the Direct and Difuse data is not given
31  if(corr != 'none'){
32    Fd <- switch(corr,
33      CPR = FdKtCPR(sol, G0d),
34      Page = FdKtPage(sol, G0d),
35      LJ = FdKtLJ(sol, G0d),
36      CLIMEDd = FdKtCLIMEDd(sol, G0d),
37      user = f(sol, G0d))
38    Kt <- Fd$Kt
39    Fd <- Fd$Fd
40    D0d <- Fd * G0
41    B0d <- G0 - D0d
42  }
43  ### the Direct and Difuse data is given
44  else {
45    G0 <- getData(G0d)$G0
46    D0d <- getData(G0d)[['D0']]
47    B0d <- getData(G0d)[['B0']]
48    Fd <- D0d/G0
49    Kt <- G0/B0d
50  }
51
52  result <- data.table(Dates = indexD(sol), Fd, Kt, G0d = G0, D0d, B0d)
53  setkey(result, 'Dates')
54  result
55 }

```

fCompI

```

1 fCompI <- function(sol, compD, GOI,
2     corr = 'EKDh', f,
3     filterG0 = TRUE){
4   if(!(corr %in% c('EKDh', 'CLIMEDh', 'BRL', 'user', 'none'))){
5     warning('Wrong descriptor of correlation Fd-Ktd. Set EKDh.')
6     corr <- 'EKDh'
7   }
8
9   if(class(sol)[1] != 'Sol'){
10     sol <- sol[, calcSol(lat = unique(lat), BTi = Dates)]
11   }
12
13   lat <- sol@lat
14   sample <- sol@sample
15   night <- sol@solI$night
16   Bo0 <- sol@solI$Bo0
17   Dates <- indexI(sol)
18
19   ## If instantaneous values are not provided, compD is used instead.
20   if (missing(GOI)) {
21
22     GOI <- collper(sol, compD)
23     GO <- GOI$GO
24     BO <- GOI$BO
25     DO <- GOI$DO
26
27     Fd <- DO/GO
28     Kt <- GO/Bo0
29
30   } else { ## Use instantaneous values if provided through GOI
31
32     if(class(GOI)[1] != 'Meteo'){
33       dt <- copy(GOI)
34       if(!('Dates' %in% names(GOI))){
35         dt[, Dates := indexI(sol)]
36         setcolorder(dt, 'Dates')
37         setkey(dt, 'Dates')
38       }
39       if('lat' %in% names(GOI)){latg <- unique(GOI$lat)}
40       else{latg <- lat}
41       GOI <- dt2Meteo(dt, latg)
42     }
43
44     if (corr!='none'){
45       GO <- getG0(GOI)
46       ## Filter values: surface irradiation must be lower than
47       ## extraterrestrial;
48       if (filterG0) {is.na(GO) <- (GO > Bo0)}
49
50       ## Fd-Kt correlation
51       Fd <- switch(corr,
52         EKDh = FdKtEKDh(sol, GOI),
53         CLIMEDh = FdKtCLIMEDh(sol, GOI),
54         BRL = FdKtBRL(sol, GOI),
55         user = f(sol, GOI))
56

```

```

57     Kt <- Fd$Kt
58     Fd <- Fd$Fd
59     D0 <- Fd * G0
60     B0 <- G0 - D0
61
62   } else {
63     G0 <- getG0(GOI)
64     D0 <- getData(GOI)[['D0']]
65     B0 <- getData(GOI)[['B0']]
66     ## Filter values: surface irradiation must be lower than
67     ## extraterrestrial;
68     if (isTRUE(filterG0)) is.na(G0) <- is.na(D0) <- is.na(B0) <- (G0 > Bo0)
69
70     Fd <- D0/G0
71     Kt <- G0/Bo0
72   }
73 }
74 ## Values outside sunrise-sunset are set to zero
75 G0[night] <- D0[night] <- B0[night] <- Kt[night] <- Fd[night] <- 0
76
77 result <- data.table(Dates, Fd, Kt, G0, D0, B0)
78 setkey(result, 'Dates')
79 result
80 }

```

fInclin

```

1 fInclin <- function(compI, angGen, iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE)
2   {
3     ##compI es class='G0'
4
5     ##Arguments
6     stopifnot(iS %in% 1:4)
7     Beta <- angGen$Beta
8     Alfa <- angGen$Alfa
9     cosTheta <- angGen$cosTheta
10
11     comp <- as.data.tableI(compI, complete=TRUE)
12     night <- comp$night
13     B0 <- comp$B0
14     Bo0 <- comp$Bo0
15     D0 <- comp$D0
16     G0 <- comp$G0
17     cosThzS <- comp$cosThzS
18     is.na(cosThzS) <- night
19
20     ##N.Martin method for dirt and non-perpendicular incidence
21     Suc <- rbind(c(1, 0.17, -0.069),
22                 c(0.98, .2, -0.054),
23                 c(0.97, 0.21, -0.049),
24                 c(0.92, 0.27, -0.023))
25     FTb <- (exp(-cosTheta/Suc[iS,2]) - exp(-1/Suc[iS,2]))/(1 - exp(-1/Suc[iS,2]))
26     FTd <- exp(-1/Suc[iS,2]) * (4/(3*pi) * (sin(Beta) + (pi - Beta - sin(Beta))/(1 +
27       cos(Beta))) +
28       Suc[iS,3] * (sin(Beta) + (pi - Beta - sin(Beta))/(1 +
29       cos(Beta)))^2))

```

```

27   FTr <- exp(-1/Suc[iS,2] * (4/(3*pi) * (sin(Beta) + (Beta - sin(Beta))/(1 - cos(
28   Beta)))) +
29   Suc[iS,3] * (sin(Beta) + (Beta - sin(Beta))/(1 - cos(
30   Beta))))^2))
31   ##Hay and Davies method for diffuse treatment
32   B <- B0 * cosTheta/cosThzS * (cosThzS>0.007) #The factor cosThzS>0.007 is needed
33   to eliminate erroneous results near dawn
34   k1 <- B0/(Bo0)
35   Di <- D0 * (1-k1) * (1+cos(Beta))/2
36   if (horizBright) Di <- Di * (1+sqrt(B0/G0) * sin(Beta/2)^3)
37   Dc <- D0 * k1 * cosTheta/cosThzS * (cosThzS>0.007)
38   R <- alb * G0 * (1-cos(Beta))/2
39   D <- (Di + Dc)
40   ##Extraterrestrial irradiance on the inclined plane
41   Bo <- Bo0 * cosTheta/cosThzS * (cosThzS>0.007)
42   ##Normal direct irradiance (DNI)
43   Bn <- B0/cosThzS
44   ##Sum of components
45   G <- B + D + R
46   Ref <- R * Suc[iS,1] * (1-FTr) * (!HCPV)
47   Ref[is.nan(FTr)] <- 0 #When cos(Beta)=1, FTr=NaN. Cancel Ref.
48   Dief <- Di * Suc[iS,1] * (1 - FTd) * (!HCPV)
49   Dcef <- Dc * Suc[iS,1] * (1 - FTb) * (!HCPV)
50   Def <- Dief + Dcef
51   Bef <- B * Suc[iS,1] * (1 - FTb)
52   Gef <- Bef + Def + Ref
53
54   result <- data.table(Bo, Bn,
55   G, D, Di, Dc, B, R,
56   FTb, FTd, FTr,
57   Dief, Dcef, Gef, Def, Bef, Ref)
58
59   ## Use 0 instead of NA for irradiance values
60   result[night] <- 0
61   result[, Dates := indexI(compI)]
62   result[, .SD, by = Dates]
63   setcolorder(result, c('Dates', names(result)[-length(result)]))
64   result
65 }

```

fProd

```

1  ## voc, isc, vmpp, impv : *cell* values
2  ## Voc, Isc, Vmpp, Impv: *module/generator* values
3
4  ## Compute Current - Voltage characteristic of a solar *cell* with Gef
5  ## and Ta
6  iv <- function(vocn, iscn, vmn, imn,
7  TONC, CoefVT = 2.3e-3,
8  Ta, Gef,
9  vmin = NULL, vmax = NULL)
10 {
11   ##Cell Constants
12   Gstc <- 1000
13   Ct <- (TONC - 20) / 800
14   Vtn <- 0.025 * (273 + 25) / 300

```

```

15   m <- 1.3
16
17   ##Cell temperature
18   Tc <- Ta + Ct * Gef
19   Vt <- 0.025 * (Tc + 273)/300
20
21   ## Series resistance
22   Rs <- (vocn - vmn + m * Vtn * log(1 - imn/iscn)) / imn
23
24   ## Voc and Isc at ambient conditions
25   voc <- vocn - CoefVT * (Tc - 25)
26   isc <- iscn * Gef/Gstc
27
28   ## Ruiz method for computing voltage and current characteristic of a *cell*
29   rs <- Rs * isc/voc
30   koc <- voc/(m * Vt)
31
32   ## Maximum Power Point
33   Dm0 <- (koc - 1)/(koc - log(koc))
34   Dm <- Dm0 + 2 * rs * Dm0^2
35
36   impp <- isc * (1 - Dm/koc)
37   vmpp <- voc * (1 - log(koc/Dm)/koc - rs * (1 - Dm/koc))
38
39   vdc <- vmpp
40   idc <- impp
41
42   ## When the MPP is below/above the inverter voltage limits, it
43   ## sets the voltage point at the corresponding limit.
44
45
46   ## Auxiliary functions for computing the current at a defined
47   ## voltage.
48   ilimit <- function(v, koc, rs)
49   {
50     if (is.na(koc))
51       result <- NA
52     else
53     {
54       ## The IV characteristic is an implicit equation. The starting
55       ## point is the voltage of the cell (imposed by the inverter
56       ## limit).
57
58       izero <- function(i, v, koc, rs)
59       {
60         vp <- v + i * rs
61         Is <- 1/(1 - exp(-koc * (1 - rs)))
62         result <- i - (1 - Is * (exp(-koc * (1 - vp)) - exp(-koc * (1 - rs))))
63       }
64
65       result <- uniroot(f = izero,
66                        interval = c(0,1),
67                        v = v,
68                        koc = koc,
69                        rs = rs)$root
70     }
71   result
72 }

```



```

73  ## Inverter minimum voltage
74  if (!is.null(vmin))
75  {
76      if (any(vmpp < vmin, na.rm = TRUE))
77      {
78          indMIN <- which(vmpp < vmin)
79          imin <- sapply(indMIN, function(i)
80          {
81              vocMIN <- voc[i]
82              kocMIN <- koc[i]
83              rsMIN <- rs[i]
84              vmin <- vmin/vocMIN
85              ##v debe estar entre 0 y 1
86              vmin[vmin < 0] <- 0
87              vmin[vmin > 1] <- 1
88              ilimit(vmin, kocMIN, rsMIN)
89          })
90          iscMIN <- isc[indMIN]
91          idc[indMIN] <- imin * iscMIN
92          vdc[indMIN] <- vmin
93          warning('Minimum MPP voltage of the inverter has been reached')}}
94  }
95
96  if (!is.null(vmax))
97  {
98      if (any(vmpp > vmax, na.rm = TRUE))
99      {
100          indMAX <- which(vmpp > vmax)
101          imax <- sapply(indMAX, function(i)
102          {
103              vocMAX <- voc[i]
104              kocMAX <- koc[i]
105              rsMAX <- rs[i]
106              vmax <- vmax / vocMAX
107              ##v debe estar entre 0 y 1
108              vmax[vmax < 0] <- 0
109              vmax[vmax > 1] <- 1
110              ilimit(vmax, kocMAX, rsMAX)
111          })
112          iscMAX <- isc[indMAX]
113          idc[indMAX] <- imax * iscMAX
114          vdc[indMAX] <- vmax
115          warning('Maximum MPP voltage of the inverter has been reached')}}
116  }
117  }
118  data.table(Ta, Tc, Gef, voc, isc, vmpp, impp, vdc, idc)
119 }
120
121 fProd <- function(inclin,
122                  module=list(),
123                  generator=list(),
124                  inverter=list(),
125                  effSys=list()
126                  )
127 {
128
129     stopifnot(is.list(module),
130              is.list(generator),

```

```

131         is.list(inverter),
132         is.list(effSys)
133     )
134     ## Extract data from objects
135     if (class(inclin)[1]=='Gef') {
136         indInclin <- indexI(inclin)
137         gefI <- as.data.tableI(inclin, complete = TRUE)
138         Gef <- gefI$Gef
139         Ta <- gefI$Ta
140     } else {
141         Gef <- inclin$Gef
142         Ta <- inclin$Ta
143     }
144
145     ## Module, generator, and inverter parameters
146     module.default <- list(Vocn = 57.6,
147                           Iscn = 4.7,
148                           Vmn = 46.08,
149                           Imn = 4.35,
150                           Ncs = 96,
151                           Ncp = 1,
152                           CoefVT = 0.0023,
153                           TONC = 47)
154     module <- modifyList(module.default, module)
155     ## Make these parameters visible because they will be used often.
156     Ncs <- module$Ncs
157     Ncp <- module$Ncp
158
159     generator.default <- list(Nms = 12,
160                              Nmp = 11)
161     generator <- modifyList(generator.default, generator)
162     generator$Pg <- (module$Vmn * generator$Nms) *
163         (module$Imn * generator$Nmp)
164     Nms <- generator$Nms
165     Nmp <- generator$Nmp
166
167     inverter.default <- list(Ki = c(0.01,0.025,0.05),
168                             Pinv = 25000,
169                             Vmin = 420,
170                             Vmax = 750,
171                             Gumb = 20)
172     inverter <- modifyList(inverter.default, inverter)
173     Pinv <- inverter$Pinv
174
175     effSys.default <- list(ModQual = 3,
176                           ModDisp = 2,
177                           OhmDC = 1.5,
178                           OhmAC = 1.5,
179                           MPP = 1,
180                           TrafoMT = 1,
181                           Disp = 0.5)
182     effSys <- modifyList(effSys.default, effSys)
183
184     ## Solar Cell i-v
185     vocn <- with(module, Vocn / Ncs)
186     iscn <- with(module, Iscn / Ncp)
187     vmn <- with(module, Vmn / Ncs)
188     imn <- with(module, Imn / Ncp)

```

```

189 vmin <- with(inverter, Vmin / (Ncs * Nms))
190 vmax <- with(inverter, Vmax / (Ncs * Nms))
191
192 cell <- iv(vocn, iscn,
193           vmn, imn,
194           module$TONC, module$CoefVT,
195           Ta, Gef,
196           vmin, vmax)
197
198 ## Generator voltage and current
199 Idc <- Nmp * Ncp * cell$idc
200 Isc <- Nmp * Ncp * cell$isc
201 Impp <- Nmp * Ncp * cell$impp
202 Vdc <- Nms * Ncs * cell$vdc
203 Voc <- Nms * Ncs * cell$voc
204 Vmpp <- Nms * Ncs * cell$vmpp
205
206 ##DC power (normalization with nominal power of inverter)
207 ##including losses
208 PdcN <- with(effSys, (Idc * Vdc) / Pinv *
209                (1 - ModQual / 100) *
210                (1 - ModDisp / 100) *
211                (1 - MPP / 100) *
212                (1 - OhmDC / 100)
213              )
214
215 ##Normalized AC power to the inverter
216 Ki <- inverter$Ki
217 if (is.matrix(Ki)) { #Ki is a matrix of nine coefficients-->dependence with
218   tension
219   VP <- cbind(Vdc, PdcN)
220   PacN <- apply(VP, 1, solvePac, Ki)
221 } else { #Ki is a vector of three coefficients-->without dependence on voltage
222   A <- Ki[3]
223   B <- Ki[2] + 1
224   C <- Ki[1] - (PdcN)
225   PacN <- (-B + sqrt(B^2 - 4 * A * C)) / (2 * A)
226 }
227 EffI <- PacN / PdcN
228 pacNeg <- PacN <= 0
229 PacN[pacNeg] <- PdcN[pacNeg] <- EffI[pacNeg] <- 0
230
231 ##AC and DC power without normalization
232 Pac <- with(effSys, PacN * Pinv *
233            (Gef > inverter$Gumb) *
234            (1 - OhmAC / 100) *
235            (1 - TrafoMT / 100) *
236            (1 - Disp / 100))
237 Pdc <- PdcN * Pinv * (Pac > 0)
238
239 ## Result
240 resProd <- data.table(Tc = cell$Tc,
241                      Voc, Isc,
242                      Vmpp, Impp,
243                      Vdc, Idc,
244                      Pac, Pdc,

```

```

246         EffI)
247   if (class(inclin)[1] %in% 'Gef'){
248     result <- resProd[, .SD,
249       by=.(Dates = indInclin)]
250     attr(result, 'generator') <- generator
251     attr(result, 'module') <- module
252     attr(result, 'inverter') <- inverter
253     attr(result, 'effSys') <- effSys
254     return(result)
255   } else {
256     result <- cbind(inclin, resProd)
257     return(result)
258   }
259 }

```

fPump

```

1 fPump <- function(pump, H){
2
3   w1=3000 ##synchronous rpm frequency
4   wm=2870 ##rpm frequency with slip when applying voltage at 50 Hz
5   s=(w1-wm)/w1
6   fen=50 ##Nominal electrical frequency
7   fmin=sqrt(H/pump$a)
8   fmax=with(pump, (-b*Qmax+sqrt(b^2*Qmax^2-4*a*(c*Qmax^2-H)))/(2*a))
9   ##fb is rotation frequency (Hz) of the pump,
10  ##fe is the electrical frequency applied to the motor
11  ##which makes it rotate at a frequency fb (and therefore also the pump).
12  fb=seq(fmin,min(60,fmax),length=1000) #The maximum frequency is 60
13  fe=fb/(1-s)
14
15  ###Flow
16  Q=with(pump, (-b*fb-sqrt(b^2*fb^2-4*c*(a*fb^2-H)))/(2*c))
17  Qmin=0.1*pump$Qn*fb/50
18  Q=Q+(Qmin-Q)*(Q<Qmin)
19
20  ###Hydraulic power
21  Ph=2.725*Q*H
22
23  ###Mechanical power
24  Q50=50*Q/fb
25  H50=H*(50/fb)^2
26  etab=with(pump, j*Q50^2+k*Q50+1)
27  Pb50=2.725*H50*Q50/etab
28  Pb=Pb50*(fb/50)^3
29
30  ###Electrical power
31  Pbc=Pb*50/fe
32  etam=with(pump, g*(Pbc/Pmn)^2+h*(Pbc/Pmn)+i)
33  Pmc=Pbc/etam
34  Pm=Pmc*fe/50
35  Pac=Pm
36  ##Pdc=Pm/(etac*(1-cab))
37
38  ###I build functions for flow, frequency and powers
39  ###to adjust the AC power.
40  fQ<-splinefun(Pac,Q)

```

```

41 fFreq<-splinefun(Pac,fe)
42 fPb<-splinefun(Pac,Pb)
43 fPh<-splinefun(Pac,Ph)
44 lim=c(min(Pac),max(Pac))
45 ##lim marks the operating range of the pump
46 result<-list(lim = lim,
47             fQ = fQ,
48             fPb = fPb,
49             fPh = fPh,
50             fFreq = fFreq)
51 }

```

fSolD

```

1 fSolD <- function(lat, BTd, method = 'michalsky'){
2   if (abs(lat) > 90){
3     lat <- sign(lat) * 90
4     warning(paste('Latitude outside acceptable values. Set to', lat))
5   }
6   sun <- data.table(Dates = unique(as.IDate(BTd)),
7                   lat = lat)
8
9   ##### solarAngles #####
10
11   ##Declination
12   sun[, decl := declination(Dates, method = method)]
13   ##Eccentricity
14   sun[, eo := eccentricity(Dates, method = method)]
15   ##Equation of time
16   sun[, EoT := eot(Dates)]
17   ##Solar time
18   sun[, ws := sunrise(Dates, lat, method = method,
19                     decl = decl)]
20   ##Extraterrestrial irradiance
21   sun[, BoOd := boOd(Dates, lat, method = method,
22                     decl = decl,
23                     eo = eo,
24                     ws = ws
25                   )]
26   setkey(sun, Dates)
27   return(sun)
28 }

```

fSolI

```

1 fSolI <- function(solD, sample = 'hour', BTi,
2                 EoT = TRUE, keep.night = TRUE, method = 'michalsky')
3 {
4   #Solar constant
5   Bo <- 1367
6
7   if(missing(BTi)){
8     d <- solD$Dates
9     BTi <- fBTi(d, sample)
10  }
11  sun <- data.table(Dates = as.IDate(BTi),

```

```

12         Times = as.ITime(BTi))
13     sun <- merge(sold, sun, by = 'Dates')
14     sun[, eqtime := EoT]
15     sun[, EoT := NULL]
16
17     #sun hour angle
18     sun[, w := sunHour(Dates, BTi, EoT = EoT, method = method, eqtime = eqtime)]
19
20     #classify night elements
21     sun[, night := abs(w) >= abs(ws)]
22
23     #zenith angle
24     sun[, cosThzS := zenith(Dates, lat, BTi,
25                             method = method,
26                             decl = decl,
27                             w = w
28                             )]
29
30     #solar altitude angle
31     sun[, AlS := asin(cosThzS)]
32
33     #azimuth
34     sun[, AzS := azimuth(Dates, lat, BTi, sample,
35                           method = method,
36                           decl = decl,
37                           w = w,
38                           cosThzS = cosThzS)]
39
40     #Extraterrestrial irradiance
41     sun[, Bo0 := Bo * eo * cosThzS]
42
43     #When it is night there is no irradiance
44     sun[night == TRUE, Bo0 := 0]
45
46     #Erase columns that are in sold
47     sun[, decl := NULL]
48     sun[, eo := NULL]
49     sun[, eqtime := NULL]
50     sun[, ws := NULL]
51     sun[, Bo0d := NULL]
52
53     #Column Dates with Times
54     sun[, Dates := as.POSIXct(Dates, Times, tz = 'UTC')]
55     sun[, Times := NULL]
56
57     #keep night
58     if(!keep.night){
59         sun <- sun[night == FALSE]
60     }
61
62     return(sun)
63 }

```

fSombra

```

1 fSombra<-function(angGen, distances, struct, modeTrk='fixed',prom=TRUE){
2

```

```

3  stopifnot(modeTrk %in% c('two','horiz','fixed'))
4  res=switch(modeTrk,
5             two={fSombra6(angGen, distances, struct, prom)},
6             horiz={fSombraHoriz(angGen, distances, struct)},
7             fixed= {fSombraEst(angGen, distances, struct)}
8             )
9  return(res)
10 }
```

```

1  fSombra2X<-function(angGen,distances,struct)
2  {
3    stopifnot(is.list(struct),is.data.frame(distances))
4    ##I prepare starting data
5    P=with(struct,distances/W)
6    b=with(struct,L/W)
7    AzS=angGen$AzS
8    Beta=angGen$Beta
9    AlS=angGen$AlS
10
11    d1=abs(P$Lew*cos(AzS)-P$Lns*sin(AzS))
12    d2=abs(P$Lew*sin(AzS)+P$Lns*cos(AzS))
13    FC=sin(AlS)/sin(Beta+AlS)
14    s=b*cos(Beta)+(b*sin(Beta)+P$H)/tan(AlS)
15    FS1=1-d1
16    FS2=s-d2
17    SombraCond=(FS1>0)*(FS2>0)*(P$Lew*Azs>=0)
18    SombraCond[is.na(SombraCond)]<-FALSE #NAs are of no use to me in a logical vector.
19    I replace them with FALSE
20    ## Result
21    FS=SombraCond*(FS1*FS2*FC)/b
22    FS[FS>1]<-1
23    return(FS)
24 }
```

```

1  fSombra6<-function(angGen, distances, struct, prom=TRUE)
2  {
3    stopifnot(is.list(struct),
4              is.data.frame(distances))
5    ##distances only has three distances, so I generate a grid
6    if (dim(distances)[1]==1){
7      Red <- distances[, .(Lew = c(-Lew, 0, Lew, -Lew, Lew),
8                                Lns = c(Lns, Lns, Lns, 0, 0),
9                                H=H)]
10   } else { #distances is an array, so there is no need to generate the grid
11     Red<-distances[1:5,] #I only need the first 5 rows...necessary in case a
12     wrong data.frame is delivered
13
14     ## I calculate the shadow due to each of the 5 followers
15     SombraGrupo<-matrix(ncol=5,nrow=dim(angGen)[1]) ###VECTORIZE
16     for (i in 1:5) {SombraGrupo[,i]<-fSombra2X(angGen,Red[i,],struct)}
17     ##To calculate the Average Shadow, I need the number of followers in each position
18     (distrib)
19     distrib=with(struct,c(1,Ncol-2,1,Nrow-1,(Ncol-2)*(Nrow-1),Nrow-1))
20     vProm=c(sum(distrib[c(5,6)]),
21             sum(distrib[c(4,5,6)]),
22             sum(distrib[c(4,5)]),
23             sum(distrib[c(2,3,5,6)]),
24             sum(distrib[c(1,2,4,5)]))
25 }
```

```

23 Nseg=sum(distrib) ##Total number of followers
24 ##With the SWEEP function I multiply the Shadow Factor of each type (ShadowGroup
    columns) by the vProm result
25
26 if (prom==TRUE){
27     ## Average Shadow Factor in the group of SIX followers taking into account
    distribution
28     FS=rowSums(sweep(SombraGrupo,2,vProm,'*'))/Nseg
29     FS[FS>1]<-1
30 } else {
31     ## Shadow factor on follower #5 due to the other 5 followers
32     FS=rowSums(SombraGrupo)
33     FS[FS>1]<-1}
34 return(FS)
35 }

```

```

1 fSombraEst<-function(angGen, distances, struct)
2 {
3     stopifnot(is.list(struct),is.data.frame(distances))
4     ## I prepare starting data
5     dist <- with(struct, distances/L)
6     Alfa <- angGen$Alfa
7     Beta <- angGen$Beta
8     AlS <- angGen$AlS
9     AzS <- angGen$AzS
10    cosTheta <- angGen$cosTheta
11    h <- dist$H #It must be previously normalized
12    d <- dist$D
13    ## Calculations
14    s=cos(Beta)+cos(Alfa-AzS)*(sin(Beta)+h)/tan(AlS)
15    FC=sin(AlS)/sin(Beta+AlS)
16    SombraCond=(s-d>0)
17    FS=(s-d)*SombraCond*FC*(cosTheta>0)
18    ## Result
19    FS=FS*(FS>0)
20    FS[FS>1]<-1
21    return(FS)
22 }

```

```

1 fSombraHoriz<-function(angGen, distances, struct)
2 {
3     stopifnot(is.list(struct),is.data.frame(distances))
4     ## I prepare starting data
5     d <- with(struct, distances/L)
6     AzS <- angGen$AzS
7     AlS <- angGen$AlS
8     Beta <- angGen$Beta
9     lew <- d$Lew #It must be previously normalized
10    ## Calculations
11    Beta0=atan(abs(sin(AzS)/tan(AlS)))
12    FS=1-lew*cos(Beta0)/cos(Beta-Beta0)
13    SombraCond=(FS>0)
14    ## Result
15    FS=FS*SombraCond
16    FS[FS>1]<-1
17    return(FS)
18 }

```


fTemp

```

1 fTemp<-function(sol, BD)
2 {
3   ##sol is an object with class='Sol'
4   ##BD is an object with class='Meteo', whose 'data' slot contains two columns
   called "TempMax" and "TempMin"
5
6   stopifnot(class(sol)=='Sol')
7   stopifnot(class(BD)=='Meteo')
8
9   checkIndexD(indexD(sol), indexD(BD))
10
11   Dates<-indexI(sol)
12   x <- as.Date(Dates)
13   ind.rep <- cumsum(c(1, diff(x) != 0))
14
15   TempMax <- BD@data$TempMax[ind.rep]
16   TempMin <- BD@data$TempMin[ind.rep]
17   ws <- sol@sold$ws[ind.rep]
18   w <- sol@soli$w
19
20   ##Generate temperature sequence from database Maxima and Minima
21
22   Tm=(TempMin+TempMax)/2
23   Tr=(TempMax-TempMin)/2
24
25   wp=pi/4
26
27   a1=pi*12*(ws-w)/(21*pi+12*ws)
28   a2=pi*(3*pi-12*w)/(3*pi-12*ws)
29   a3=pi*(24*pi+12*(ws-w))/(21*pi+12*ws)
30
31   T1=Tm-Tr*cos(a1)
32   T2=Tm+Tr*cos(a2)
33   T3=Tm-Tr*cos(a3)
34
35   Ta=T1*(w<=ws)+T2*(w>ws&w<=wp)+T3*(w>wp)
36
37   ##Result
38   result<-data.table(Dates, Ta)
39 }

```

fTheta

```

1 fTheta<-function(sol, beta, alfa=0, modeTrk='fixed', betaLim=90,
2   BT=FALSE, struct, dist)
3 {
4   stopifnot(modeTrk %in% c('two','horiz','fixed'))
5   if (!missing(struct)) {stopifnot(is.list(struct))}
6   if (!missing(dist)) {stopifnot(is.data.frame(dist))}
7
8   betaLim=d2r(betaLim)
9   lat=getLat(sol, 'rad')
10  signLat=ifelse(sign(lat)==0, 1, sign(lat)) ##When lat=0, sign(lat)=0. I change it
   to sign(lat)=1
11

```

```

12 solI<-as.data.tableI(sol, complete=TRUE, day = TRUE)
13 AlS=solI$AlS
14 AzS=solI$AzS
15 decl=solI$decl
16 w<-solI$w
17
18 night<-solI$night
19
20 Beta<-switch(modeTrk,
21             two = {Beta2x=pi/2-AlS
22                   Beta=Beta2x+(betaLim-Beta2x)*(Beta2x>betaLim)},
23             fixed = rep(d2r(beta), length(w)),
24             horiz={BetaHoriz0=atan(abs(sin(AzS)/tan(AlS)))
25                   if (BT){lew=dist$Lew/struct$L
26                           Longitud=lew*cos(BetaHoriz0)
27                           Cond=(Longitud>=1)
28                           Longitud[Cond]=1
29                           ## When Cond==TRUE Length=1
30                           ## and therefore asin(Length)=pi/2,
31                           ## so that BetaHoriz=BetaHoriz0
32                           BetaHoriz=BetaHoriz0+asin(Longitud)-pi/2
33                   } else {
34                       BetaHoriz=BetaHoriz0
35                       rm(BetaHoriz0)}
36                   Beta=ifelse(BetaHoriz>betaLim,betaLim,BetaHoriz)}
37             )
38 is.na(Beta) <- night
39
40 Alfa<-switch(modeTrk,
41             two = AzS,
42             fixed = rep(d2r(alfa), length(w)),
43             horiz=pi/2*sign(AzS))
44 is.na(Alfa) <- night
45
46 cosTheta<-switch(modeTrk,
47                 two=cos(Beta-(pi/2-AlS)),
48                 horiz={
49                     t1=sin(decl)*sin(lat)*cos(Beta)
50                     t2=cos(decl)*cos(w)*cos(lat)*cos(Beta)
51                     t3=cos(decl)*abs(sin(w))*sin(Beta)
52                     cosTheta=t1+t2+t3
53                     rm(t1,t2,t3)
54                     cosTheta
55                 },
56                 fixed={
57                     t1=sin(decl)*sin(lat)*cos(Beta)
58                     t2=-signLat*sin(decl)*cos(lat)*sin(Beta)*cos(Alfa)
59                     t3=cos(decl)*cos(w)*cos(lat)*cos(Beta)
60                     t4=signLat*cos(decl)*cos(w)*sin(lat)*sin(Beta)*cos(Alfa)
61                     t5=cos(decl)*sin(w)*sin(Alfa)*sin(Beta)
62                     cosTheta=t1+t2+t3+t4+t5
63                     rm(t1,t2,t3,t4,t5)
64                     cosTheta
65                 }
66             )
67 is.na(cosTheta) <- night
68 cosTheta=cosTheta*(cosTheta>0) #when cosTheta<0, Theta is greater than 90°, and
                                therefore the Sun is behind the panel.

```

```

69
70     result <- data.table(Dates = indexI(sol),
71                           Beta, Alfa, cosTheta)
72     return(result)
73 }

```

HQCurve

```

1  ## HQCurve: no visible binding for global variable 'fb'
2  ## HQCurve: no visible binding for global variable 'Q'
3  ## HQCurve: no visible binding for global variable 'x'
4  ## HQCurve: no visible binding for global variable 'y'
5  ## HQCurve: no visible binding for global variable 'group.value'
6
7  if(getRversion() >= "2.15.1") globalVariables(c('fb', 'Q', 'x', 'y', 'group.value'))
8
9  HQCurve<-function(pump){
10     w1=3000 #synchronous rpm frequency
11     wm=2870 #rpm frequency with slip when applying voltage at 50 Hz
12     s=(w1-wm)/w1
13     fen=50 #Nominal electrical frequency
14
15     f=seq(35,50,by=5)
16     Hn=with(pump,a*50^2+b*50*Qn+c*Qn^2) #height corresponding to flow rate and nominal
        frequency
17
18     kiso=Hn/pump$Qn^2 #To paint the isoyield curve I take into account the laws of
        similarity
19     Qiso=with(pump,seq(0.1*Qn,Qmax,l=10))
20     Hiso=kiso*Qiso^2 #Isoperformance curve
21
22     Curva<-expand.grid(fb=f,Q=Qiso)
23
24     Curva<-within(Curva,{
25         fe=fb/(1-s)
26         H=with(pump,a*fb^2+b*fb*Q+c*Q^2)
27
28         is.na(H) <- (H<0)
29         Q50=50*Q/fe
30         H50=H*(50/fe)^2
31         etab=with(pump,j*Q50^2+k*Q50+1)
32         Pb50=2.725*H50*Q50/etab
33         Pb=Pb50*(fb/50)^3
34
35         Pbc=Pb*50/fe
36         etam=with(pump,g*(Pbc/Pmn)^2+h*(Pbc/Pmn)+i)
37         Pmc=Pbc/etam
38         Pm=Pmc*fe/50
39
40         etac=0.95 #Variable frequency drive performance
41         cab=0.05 #Cable losses
42         Pdc=Pm/(etac*(1-cab))
43         rm(etac,cab,Pmc,Pbc,Pb50,Q50,H50)
44     })
45
46     ###H-Q curve at different frequencies

```

```

47  ##I check if I have the lattice package available, which should have been loaded in
    .First.lib
48  lattice.disp<-"lattice" %in% .packages()
49  latticeExtra.disp<-"latticeExtra" %in% .packages()
50  if (lattice.disp && latticeExtra.disp) {
51    p<-xyplot(H~Q,groups=factor(fb),data=Curva, type='l',
52             par.settings=custom.theme.2(),
53             panel=function(x,y,groups,...){
54               panel.superpose(x,y,groups,...)
55               panel.xyplot(Qiso,Hiso,col='black',...)
56               panel.text(Qiso[1], Hiso[1], 'ISO', pos=3)}
57    )
58    p=p+glayer(panel.text(x[1], y[1], group.value, pos=3))
59    print(p)
60    result<-list(result=Curva, plot=p)
61  } else {
62    warning('lattice and/or latticeExtra packages are not available. Thus, the plot
        could not be created')
63    result<-Curva}
64  }

```

local2Solar

```

1  local2Solar <- function(x, lon=NULL){
2    tz=attr(x, 'tzone')
3    if (tz==' ' || is.null(tz)) {tz='UTC'}
4    ##Daylight savings time
5    AO=3600*dst(x)
6    AOneg=(AO<0)
7    if (any(AOneg)) {
8      AO[AOneg]=0
9      warning('Some Daylight Savings Time unknown. Set to zero.')
10   }
11   ##Difference between local longitude and time zone longitude LH
12   LH=lonHH(tz)
13   if (is.null(lon))
14     {deltaL=0
15    } else
16     {deltaL=d2r(lon)-LH
17   }
18   ##Local time corrected to UTC
19   tt <- format(x, tz=tz)
20   result <- as.POSIXct(tt, tz='UTC')-AO+r2sec(deltaL)
21   result
22 }

```

markovG0

```

1  ## Objects loaded at startup from data/MTM.RData
2  if(getRversion() >= "2.15.1") globalVariables(c(
3    'MTM', ## Markov Transition Matrices
4    'Ktmtm', ## Kt limits to choose a matrix from MTM
5    'Ktlim' ## Daily kt range of each matrix.
6  ))
7
8  markovG0 <- function(G0dm, sold){

```

```

9      sold <- copy(sold)
10     timeIndex <- sold$Dates
11     Bo0d <- sold$Bo0d
12     Bo0dm <- sold[, mean(Bo0d), by = .(month(Dates), year(Dates))][[3]]
13     ktm <- G0dm/Bo0dm
14
15     ##Calculates which matrix to work with for each month
16     whichMatrix <- findInterval(ktm, Ktmtm, all.inside = TRUE)
17
18     ktd <- state <- numeric(length(timeIndex))
19     state[1] <- 1
20     ktd[1] <- ktm[state[1]]
21     for (i in 2:length(timeIndex)){
22         iMonth <- month(timeIndex[i])
23         colMonth <- whichMatrix[iMonth]
24         rng <- Ktlim[, colMonth]
25         classes <- seq(rng[1], rng[2], length=11)
26         matMonth <- MTM[(10*colMonth-9):(10*colMonth),]
27         ## http://www-rohan.sdsu.edu/~babailey/stat575/mcsim.r
28         state[i] <- sample(1:10, size=1, prob=matMonth[state[i-1],])
29         ktd[i] <- runif(1, min=classes[state[i]], max=classes[state[i]+1])
30     }
31     G0dmMarkov <- data.table(ktd, Bo0d)
32     G0dmMarkov <- G0dmMarkov[, mean(ktd*Bo0d), by = .(month(timeIndex), year(timeIndex))][[3]]
33     fix <- G0dm/G0dmMarkov
34     indRep <- month(timeIndex)
35     fix <- fix[indRep]
36     G0d <- data.table(Dates = timeIndex, G0d = ktd * Bo0d * fix)
37     G0d
38 }

```

NmgPVPS

```

1  ## NmgPVPS: no visible binding for global variable 'Pnom'
2  ## NmgPVPS: no visible binding for global variable 'group.value'
3
4  if(getRversion() >= "2.15.1") globalVariables(c('Pnom', 'group.value'))
5
6  NmgPVPS <- function(pump, Pg, H, Gd, Ta=30,
7                      lambda=0.0045, TONC=47,
8                      eta=0.95, Gmax=1200, t0=6, Nm=6,
9                      title='', theme=custom.theme.2()){
10
11     ##I build the type day by IEC procedure
12     t=seq(-t0,t0,l=2*t0*Nm);
13     d=Gd/(Gmax*2*t0)
14     s=(d*pi/2-1)/(1-pi/4)
15     G=Gmax*cos(t/t0*pi/2)*(1+s*(1-cos(t/t0*pi/2)))
16     G[G<0]<-0
17     G=G/(sum(G,na.rm=1)/Nm)*Gd
18     Red<-expand.grid(G=G,Pnom=Pg,H=H,Ta=Ta)
19     Red<-within(Red,{Tcm<-Ta+G*(TONC-20)/800
20                     Pdc=Pnom*G/1000*(1-lambda*(Tcm-25)) #Available DC power
21                     Pac=Pdc*eta}) #Inverter yield
22
23     res=data.table(Red,Q=0)

```

```

24
25   for (i in seq_along(H)){
26     fun=fPump(pump, H[i])
27     Cond=res$H==H[i]
28     x=res$Pac[Cond]
29     z=res$Pdc[Cond]
30     rango=with(fun,x>=lim[1] & x<=lim[2]) #I limit the power to the operating
range of the pump.
31     x[!rango]<-0
32     z[!rango]<-0
33     y=res$Q[Cond]
34     y[rango]<-fun$fQ(x[rango])
35     res$Q[Cond]=y
36     res$Pac[Cond]=x
37     res$Pdc[Cond]=z
38   }
39
40   resumen <- res[, lapply(.SD, function(x)sum(x, na.rm = 1)/Nm),
41                     by = .(Pnom, H)]
42   param=list(pump=pump, Pg=Pg, H=H, Gd=Gd, Ta=Ta,
43             lambda=lambda, TONC=TONC, eta=eta,
44             Gmax=Gmax, t0=t0, Nm=Nm)
45
46
47   ###Abacus with common X-axes
48
49   ##I check if I have the lattice package available, which should have been loaded
in .First.lib
50   lattice.disp<-"lattice" %in% .packages()
51   latticeExtra.disp<-"latticeExtra" %in% .packages()
52   if (lattice.disp && latticeExtra.disp){
53     tema<-theme
54     tema1 <- modifyList(tema, list(layout.width = list(panel=1,
55                                                         ylab = 2, axis.left=1.0,
56                                                         left.padding=1, ylab.axis.padding=1,
57                                                         axis.panel=1)))
58     tema2 <- modifyList(tema, list(layout.width = list(panel=1,
59                                                         ylab = 2, axis.left=1.0, left.padding=1,
60                                                         ylab.axis.padding=1, axis.panel=1)))
61     temaT <- modifyList(tema, list(layout.heights = list(panel = c(1, 1))))
62     p1 <- xyplot(Q~Pdc, groups=H, data=resumen,
63                 ylab="Qd (m\u00b3/d)",type=c('l','g'),
64                 par.settings = tema1)
65
66     p1lab<-p1+glayer(panel.text(x[1], y[1], group.value, pos=2, cex=0.7))
67
68     ##I paint the linear regression because Pnom~Pdc depends on the height.
69     p2 <- xyplot(Pnom~Pdc, groups=H, data=resumen,
70                 ylab="Pg",type=c('l','g'), #type=c('smooth','g'),
71                 par.settings = tema2)
72     p2lab<-p2+glayer(panel.text(x[1], y[1], group.value, pos=2, cex=0.7))
73
74     p<-update(c(p1lab, p2lab, x.same = TRUE),
75              main=paste(title, '\nSP', pump$Qn, 'A', pump$stages, ' ',
76                        'Gd ', Gd/1000," kWh/m\u00b2",sep=''),
77              layout = c(1, 2),
78              scales=list(x=list(draw=FALSE)),
79              xlab='',

```

```

80         ylab = list(c("Qd (m\u00b3/d)", "Pg (Wp)"), y = c(1/4, 3/4)),
81         par.settings = temaT
82     )
83     print(p)
84     result<-list(I=res,D=resumen, plot=p, param=param)
85 } else {
86     warning('lattice, latticeExtra packages are not all available. Thus, the plot
87     could not be created')
88     result<-list(I=res,D=resumen, param=param)
89 }

```

utils-angle

```

1  #degrees to radians
2  d2r<-function(x){x*pi/180}
3
4  #radians to degrees
5  r2d<-function(x){x*180/pi}
6
7  #hours to radians
8  h2r<-function(x){x*pi/12}
9
10 #hours to degrees
11 h2d<-function(x){x*180/12}
12
13 #radians to hours
14 r2h<-function(x){x*12/pi}
15
16 #degrees to hours
17 d2h<-function(x){x*12/180}
18
19 #radians to seconds
20 r2sec<-function(x){x*12/pi*3600}
21
22 #radians to minutes
23 r2min<-function(x){x*12/pi*60}

```

utils-time

```

1  #complete time to hours
2  t2h <- function(x)
3  {
4      hour(x)+minute(x)/60+second(x)/3600
5  }
6
7  #hours minutes and seconds to hours
8  hms <- function(x)
9  {
10     hour(x)+minute(x)/60+second(x)/3600
11 }
12
13 #day of the year
14 doy <- function(x){
15     as.numeric(format(x, '%j'))
16 }

```

```

17
18 #day of the month
19 dom <- function(x){
20   as.numeric(format(x, '%d'))
21 }
22
23 #trunc days
24 truncDay <- function(x){as.POSIXct(trunc(x, units='days'))}

```

A.4. Métodos

as.data.tableI

as.data.tableD

```

1 setGeneric('as.data.tableD', function(object, complete=FALSE, day=FALSE){
2   standardGeneric('as.data.tableD')})
3
4 setMethod('as.data.tableD',
5   signature=(object='Sol'),
6   definition=function(object, complete=FALSE, day=FALSE){
7     sol <- copy(object)
8     sold <- sol@sold
9     data <- sold
10    if(day){
11      ind <- indexD(object)
12      data[, day := doy(ind)]
13      data[, month := month(ind)]
14      data[, year := year(ind)]
15    }
16    return(data)
17  })
18
19 setMethod('as.data.tableD',
20   signature = (object='G0'),
21   definition = function(object, complete=FALSE, day=FALSE){
22     g0 <- copy(object)
23     GOD <- g0@GOD
24     sold <- g0@sold
25     if(complete){
26       data <- data.table(GOD, sold[, Dates := NULL])
27     } else {
28       GOD[, Fd := NULL]
29       GOD[, Kt := NULL]
30       data <- GOD
31     }
32     if(day){
33       ind <- indexD(object)
34       data[, day := doy(ind)]
35       data[, month := month(ind)]
36       data[, year := year(ind)]
37     }
38     return(data)
39  })
40
41 setMethod('as.data.tableD',
42   signature = (object='Gef'),

```



```

101         GOD[, Dates := NULL],
102         sold[, Dates := NULL]
103     )
104     } else { data <- prodD[, c('Dates', 'Eac',
105                               'Qd', 'Yf')]}
106     if(day){
107         ind <- indexD(object)
108         data[, day := doy(ind)]
109         data[, month := month(ind)]
110         data[, year := year(ind)]
111     }
112     return(data)
113 }
114 )

```

as.data.tableM

```

1  setGeneric('as.data.tableM', function(object, complete = FALSE, day=FALSE){
2      standardGeneric('as.data.tableM')})
3
4  setMethod('as.data.tableM',
5      signature=(object='G0'),
6      definition=function(object, complete=FALSE, day=FALSE){
7          g0 <- copy(object)
8          G0dm <- g0@G0dm
9          data <- G0dm
10         if(day){
11             ind <- indexD(object)
12             data[, month := month(ind)]
13             data[, year := year(ind)]
14         }
15         return(data)
16     }
17 )
18
19 setMethod('as.data.tableM',
20     signature=(object='Gef'),
21     definition = function(object, complete=FALSE, day=FALSE){
22         gef <- copy(object)
23         Gefdm <- gef@Gefdm
24         G0dm <- gef@G0dm
25         if(complete){
26             data <- data.table(Gefdm, G0dm[, Dates := NULL])
27         } else {data <- Gefdm}
28         if(day){
29             ind <- indexD(object)
30             data[, month := month(ind)]
31             data[, year := year(ind)]
32         }
33         return(data)
34     }
35 )
36
37 setMethod('as.data.tableM',
38     signature = (object='ProdGCPV'),
39     definition = function(object, complete=FALSE, day=FALSE){
40         prodgcpv <- copy(object)

```

```

40     prodDm <- prodgcpv@prodDm
41     Gefdm <- prodgcpv@Gefdm
42     G0dm <- prodgcpv@G0dm
43     if(complete){
44         data <- data.table(prodDm,
45                             Gefdm[, Dates := NULL],
46                             G0dm[, Dates := NULL])
47     } else {data <- prodDm}
48     if(day){
49         ind <- indexD(object)
50         data[, month := month(ind)]
51         data[, year := year(ind)]
52     }
53     return(data)
54 }
55 )
56
57 setMethod('as.data.tableM',
58     signature = (object='ProdPVPS'),
59     definition = function(object, complete=FALSE, day=FALSE){
60         prodpvps <- copy(object)
61         prodDm <- prodpvps@prodDm
62         Gefdm <- prodpvps@Gefdm
63         G0dm <- prodpvps@G0dm
64         if(complete){
65             data <- data.table(prodDm,
66                                 Gefdm[, Dates := NULL],
67                                 G0dm[, Dates := NULL])
68         } else {data <- prodDm}
69         if(day){
70             ind <- indexD(object)
71             data[, month := month(ind)]
72             data[, year := year(ind)]
73         }
74         return(data)
75     }
76 )

```

as.data.tableY

```

1  setGeneric('as.data.tableY', function(object, complete=FALSE, day=FALSE){
2      standardGeneric('as.data.tableY')})
3
4  setMethod('as.data.tableY',
5      signature=(object='G0'),
6      definition=function(object, complete=FALSE, day=FALSE){
7          g0 <- copy(object)
8          G0y <- g0@G0y
9          data <- G0y
10         if(day){data[, year := Dates]}
11         return(data)
12     }
13 )
14
15 setMethod('as.data.tableY',
16     signature = (object='Gef'),
17     definition = function(object, complete=FALSE, day=FALSE){

```

```

17     gef <- copy(object)
18     Gefy <- gef@Gefy
19     GOy <- gef@GOy
20     if(complete){
21         data <- data.table(Gefy, GOy[, Dates := NULL])
22     } else {data <- Gefy}
23     if(day){data[, year := Dates]}
24     return(data)
25 }
26 )
27
28 setMethod('as.data.tableY',
29     signature = (object='ProdGCPV'),
30     definition = function(object, complete=FALSE, day=FALSE){
31         prodgcpv <- copy(object)
32         prody <- prodgcpv@prody
33         Gefy <- prodgcpv@Gefy
34         GOy <- prodgcpv@GOy
35         if(complete){
36             data <- data.table(prody,
37                               Gefy[, Dates := NULL],
38                               GOy[, Dates := NULL])
39         } else {data <- prody}
40         if(day){data[, year := Dates]}
41         return(data)
42     }
43 )
44
45 setMethod('as.data.tableY',
46     signature = (object='ProdPVPS'),
47     definition = function(object, complete=FALSE, day=FALSE){
48         prodpvps <- copy(object)
49         prody <- prodpvps@prody
50         Gefy <- prodpvps@Gefy
51         GOy <- prodpvps@GOy
52         if(complete){
53             data <- data.table(prody,
54                               Gefy[, Dates := NULL],
55                               GOy[, Dates := NULL])
56         } else {data <- prody}
57         if(day){data[, year := Dates]}
58         return(data)
59     }
60 )

```

compare

```

1  ## compareFunction: no visible binding for global variable 'name'
2  ## compareFunction: no visible binding for global variable 'x'
3  ## compareFunction: no visible binding for global variable 'y'
4  ## compareFunction: no visible binding for global variable 'group.value'
5
6  if(getRversion() >= "2.15.1") globalVariables(c('name', 'x', 'y', 'group.value'))
7
8  setGeneric('compare', signature='...', function(...){standardGeneric('compare')})
9
10 compareFunction <- function(..., vars){

```

```

11 dots <- list(...)
12 nms0 <- substitute(list(...))
13 if (!is.null(names(nms0))) { ##in do.call
14   nms <- names(nms0[-1])
15 } else {
16   nms <- as.character(nms0[-1])
17 }
18 foo <- function(object, label){
19   yY <- colMeans(as.data.tableY(object, complete = TRUE)[, ..vars])
20   yY <- cbind(stack(yY), name=label)
21   yY
22 }
23 cdata <- mapply(FUN=foo, dots, nms, SIMPLIFY=FALSE)
24 z <- do.call(rbind, cdata)
25 z$ind <- ordered(z$ind, levels=vars)
26 p <- dotplot(ind~values, groups=name, data=z, type='b',
27             par.settings=solaR.theme)
28 print(p+glayer(panel.text(x[length(x)], y[length(x)],
29                           label=group.value, cex=0.7, pos=3, srt=45)))
30 return(z)
31 }
32
33
34 setMethod('compare',
35           signature='G0',
36           definition=function(...){
37             vars <- c('D0d', 'B0d', 'G0d')
38             res <- compareFunction(..., vars=vars)
39             return(res)
40           }
41 )
42
43 setMethod('compare',
44           signature='Gef',
45           definition=function(...){
46             vars <- c('Defd', 'Befd', 'Gefd')
47             res <- compareFunction(..., vars=vars)
48             return(res)
49           }
50 )
51
52 setMethod('compare',
53           signature='ProdGCPV',
54           definition=function(...){
55             vars <- c('G0d', 'Gefd', 'Yf')
56             res <- compareFunction(..., vars=vars)
57             return(res)
58           }
59 )

```

getData

```

1 ## extracts the data for class Meteo ##
2 setGeneric('getData', function(object){standardGeneric('getData')})
3
4 ### getData ####
5 setMethod('getData',

```

```

6     signature = (object = 'Meteo'),
7     definition = function(object){
8         result <- object@data
9         return(result)
10    }

```

getG0

```

1  ## extracts the global irradiance for class Meteo ##
2  setGeneric('getG0', function(object){standardGeneric('getG0')})
3
4  ### getG0 ###
5  setMethod('getG0',
6            signature = (object = 'Meteo'),
7            definition = function(object){
8                result <- getData(object)
9                return(result$G0)
10           })

```

getLat

```

1  ## extracts the latitude from the objects ##
2  setGeneric('getLat', function(object, units = 'rad')
3  {standardGeneric('getLat')})
4
5  ## extracts the latitude from the objects ##
6  setGeneric('getLat', function(object, units = 'rad')
7  {standardGeneric('getLat')})
8
9  setMethod('getLat',
10           signature = (object = 'Meteo'),
11           definition = function(object, units = 'rad'){
12               stopifnot(units %in% c('deg', 'rad'))
13               result = switch(units,
14                               rad = d2r(object@latm),
15                               deg = object@latm)
16               return(result)
17           })

```

indexD

```

1  ## extract the index of the daily data ##
2  setGeneric('indexD', function(object){standardGeneric('indexD')})
3  ### indexD ###
4  setMethod('indexD',
5            signature = (object = 'Sol'),
6            definition = function(object){as.POSIXct(object@solD$Dates)
7            })
8
9  setMethod('indexD',
10           signature = (object = 'Meteo'),
11           definition = function(object){as.POSIXct(getData(object)$Dates)})

```

indexI

```

1  ## extract the index of the intradaily data ##
2  setGeneric('indexI', function(object){standardGeneric('indexI')})
3  ### indexI ###
4  setMethod('indexI',
5            signature = (object = 'Sol'),
6            definition = function(object){as.POSIXct(object@solI$Dates)
7            })

```

levelplot

```

1  setGeneric('levelplot')
2
3  setMethod('levelplot',
4            signature=c(x='formula', data='Meteo'),
5            definition=function(x, data,
6                               par.settings = solaR.theme,
7                               panel = panel.levelplot.raster, interpolate = TRUE,
8                               xscale.components = xscale.solar,
9                               yscale.components = yscale.solar,
10                              ...){
11      data0=getData(data)
12      ind=data0$Dates
13      data0$day=doy(ind)
14      data0$month=month(ind)
15      data0$year=year(ind)
16      data0$w=h2r(hms(ind)-12)
17      levelplot(x, data0,
18               par.settings = par.settings,
19               xscale.components = xscale.components,
20               yscale.components = yscale.components,
21               panel = panel, interpolate = interpolate,
22               ...)
23      }
24      )
25
26  setMethod('levelplot',
27            signature=c(x='formula', data='Sol'),
28            definition=function(x, data,
29                               par.settings = solaR.theme,
30                               panel = panel.levelplot.raster, interpolate = TRUE,
31                               xscale.components = xscale.solar,
32                               yscale.components = yscale.solar,
33                              ...){
34      data0=as.data.tableI(data, complete=TRUE, day=TRUE)
35      ind=data0$Dates
36      data0$day=doy(ind)
37      data0$month=month(ind)
38      data0$year=year(ind)
39      levelplot(x, data0,
40               par.settings = par.settings,
41               xscale.components = xscale.components,
42               yscale.components = yscale.components,
43               panel = panel, interpolate = interpolate,
44               ...)
45      }

```

```

46     )
47
48   setMethod('levelplot',
49     signature=c(x='formula', data='G0'),
50     definition=function(x, data,
51       par.settings = solaR.theme,
52       panel = panel.levelplot.raster, interpolate = TRUE,
53       xscale.components = xscale.solar,
54       yscale.components = yscale.solar,
55       ...){
56       data0=as.data.tableI(data, complete=TRUE, day=TRUE)
57       ind=data0$Dates
58       data0$day=doy(ind)
59       data0$month=month(ind)
60       data0$year=year(ind)
61       levelplot(x, data0,
62         par.settings = par.settings,
63         xscale.components = xscale.components,
64         yscale.components = yscale.components,
65         panel = panel, interpolate = interpolate,
66         ...)
67     }
68   )

```

losses

```

1  setGeneric('losses', function(object){standardGeneric('losses')})
2
3  setMethod('losses',
4    signature=(object='Gef'),
5    definition=function(object){
6      dat <- as.data.tableY(object, complete=TRUE)
7      isShd=('Gef0d' %in% names(dat)) ##is there shadows?
8      if (isShd) {
9        shd <- with(dat, mean(1-Gefd/Gef0d))
10       eff <- with(dat, mean(1-Gef0d/Gd))
11     } else {
12       shd <- 0
13       eff <- with(dat, mean(1-Gefd/Gd))
14     }
15     result <- data.table(Shadows = shd, AoI = eff)
16     result
17   }
18   )
19
20  setMethod('losses',
21    signature=(object='ProdGCPV'),
22    definition=function(object){
23      datY <- as.data.tableY(object, complete=TRUE)
24      module0=object@module
25      module0$CoefVT=0 ##No losses with temperature
26      Pg=object@generator$Pg
27      Nm=1/sample2Hours(object@sample)
28      datI <- as.data.tableI(object, complete=TRUE)
29      if (object@type=='prom'){
30        datI[, DayOfMonth := DOM(datI)]
31        YfDCO <- datI[, sum(Vmpp*Impp/Pg*DayOfMonth, na.rm = TRUE),

```



```

32         by = month(Dates))[[2]]
33     YfDC0 <- sum(YfDC0, na.rm = TRUE)
34     YfACO <- datI[, sum(Pdc*EffI/Pg*DayOfMonth, na.rm = TRUE),
35         by = month(Dates))[[2]]
36     YfACO <- sum(YfACO, na.rm = TRUE)
37 } else {
38     datI[, DayOfMonth := DOM(datI)]
39     YfDC0 <- datI[, sum(Vmpp*ImpP/Pg*DayOfMonth, na.rm = TRUE),
40         by = year(Dates))[[2]]
41     YfACO <- datI[, sum(Pdc*EffI/Pg*DayOfMonth, na.rm = TRUE),
42         by = year(Dates))[[2]]
43 }
44 gen <- mean(1-YfDC0/datY$Gefd)
45 YfDC <- datY$Edc/Pg*1000
46 DC=mean(1-YfDC/YfDC0)
47 inv=mean(1-YfACO/YfDC)
48 AC=mean(1-datY$Yf/YfACO)
49 result0 <- losses(as(object, 'Gef'))
50 result1 <- data.table(Generator = gen,
51     DC = DC,
52     Inverter = inv,
53     AC = AC)
54 result <- data.table(result0, result1)
55 result
56 }
57 )
58
59 ###compareLosses
60
61 ## compareLosses,ProdGCPV: no visible binding for global variable 'name'
62 if(getRversion() >= "2.15.1") globalVariables(c('name'))
63
64 setGeneric('compareLosses', signature='...', function(...){standardGeneric('
65     compareLosses')})
66
67 setMethod('compareLosses', 'ProdGCPV',
68     definition=function(...){
69         dots <- list(...)
70         nms0 <- substitute(list(...))
71         if (!is.null(names(nms0))){ ##do.call
72             nms <- names(nms0[-1])
73         } else {
74             nms <- as.character(nms0[-1])
75         }
76         foo <- function(object, label){
77             yY <- losses(object)
78             yY <- cbind(yY, name=label)
79             yY
80         }
81         cdata <- mapply(FUN=foo, dots, nms, SIMPLIFY=FALSE)
82         z <- do.call(rbind, cdata)
83         z <- melt(z, id.vars = 'name')
84         p <- dotplot(variable~value*100, groups=name, data=z,
85             par.settings=solaR.theme, type='b',
86             auto.key=list(corner=c(0.95,0.2), cex=0.7), xlab='Losses (%)'
87         )
88         print(p)
89         return(z)

```

```

88     }
89   )

```

mergeSolar

```

1  setGeneric('mergesolaR', signature='...', function(...){standardGeneric('mergesolaR')
   })
2
3  fooMeteo <- function(object, var){yY <- getData(object)[, .SD,
4                                     by = Dates,
5                                     .SDcols = var]}
6
7  fooG0 <- function(object, var){yY <- as.data.tableD(object)[, .SD,
8                                     by = Dates,
9                                     .SDcols = var]}
10
11 mergeFunction <- function(..., foo, var){
12   dots <- list(...)
13   dots <- lapply(dots, as, class(dots[[1]])) ##the first element is the one that
   dictates the class to everyone
14   nms0 <- substitute(list(...))
15   if (!is.null(names(nms0))){ ##do.call
16     nms <- names(nms0[-1])
17   } else {
18     nms <- as.character(nms0[-1])
19   }
20   cdata <- sapply(dots, FUN=foo, var, simplify=FALSE)
21   z <- cdata[[1]]
22   for (i in 2:length(cdata)){
23     z <- merge(z, cdata[[i]], by = 'Dates', suffixes = c("", paste0('.', i)))
24   }
25   names(z)[-1] <- nms
26   z
27 }
28
29 setMethod('mergesolaR',
30           signature='Meteo',
31           definition=function(...){
32             res <- mergeFunction(..., foo=fooMeteo, var='G0')
33             res
34           }
35 )
36
37 setMethod('mergesolaR',
38           signature='G0',
39           definition=function(...){
40             res <- mergeFunction(..., foo=fooG0, var='G0d')
41             res
42           }
43 )
44
45 setMethod('mergesolaR',
46           signature='Gef',
47           definition=function(...){
48             res <- mergeFunction(..., foo=fooG0, var='Gefd')
49             res
50           }

```

```

51     )
52
53   setMethod('mergesolaR',
54     signature='ProdGCPV',
55     definition=function(...){
56       res <- mergeFunction(..., foo=fooG0, var='Yf')
57       res
58     }
59   )
60
61   setMethod('mergesolaR',
62     signature='ProdPVPS',
63     definition=function(...){
64       res <- mergeFunction(..., foo=fooG0, var='Yf')
65       res
66     }
67   )

```

shadeplot

```

1  setGeneric('shadeplot', function(x, ...)standardGeneric('shadeplot'))
2
3  setMethod('shadeplot', signature(x='Shade'),
4    function(x,
5      main='',
6      xlab=expression(L[ew]),
7      ylab=expression(L[ns]),
8      n=9, ...){
9    red=x@distances
10    FS.loess=x@FS.loess
11    Yf.loess=x@Yf.loess
12    struct=x@struct
13    mode=x@modeTrk
14    if (mode=='two'){
15      Lew=seq(min(red$Lew),max(red$Lew),length=100)
16      Lns=seq(min(red$Lns),max(red$Lns),length=100)
17      Red=expand.grid(Lew=Lew,Lns=Lns)
18      FS=predict(FS.loess,Red)
19      Red$FS=as.numeric(FS)
20      AreaG=with(struct,L*W)
21      GRR=Red$Lew*Red$Lns/AreaG
22      Red$GRR=GRR
23      FS.m<-matrix(1-FS,
24        nrow=length(Lew),
25        ncol=length(Lns))
26      GRR.m<-matrix(GRR,
27        nrow=length(Lew),
28        ncol=length(Lns))
29      niveles=signif(seq(min(FS.m),max(FS.m),l=n+1),3)
30      pruebaCB<-("RColorBrewer" %in% .packages())
31      if (pruebaCB) {
32        paleta=rev(brewer.pal(n, 'YlOrRd'))
33      } else {
34        paleta=rev(heat.colors(n))}
35      par(mar=c(4.1,4.1,2.1,2.1))
36      filled.contour(x=Lew,y=Lns,z=FS.m,#...,
37        col=paleta, #levels=niveles,

```

```

38         nlevels=n,
39         plot.title=title(xlab=xlab,
40                         ylab=ylab, main=main),
41         plot.axes={
42             axis(1);axis(2);
43             contour(Lew, Lns, FS.m,
44                     nlevels=n, #levels=niveles,
45                     col="black", labcex=.8, add=TRUE)
46             contour(Lew, Lns, GRR.m,
47                     col="black", lty=3, labcex=.8, add=TRUE)
48             grid(col="white",lty=3)},
49         key.title=title("1-FS",cex.main=.8))
50     }
51     if (mode=='horiz') {
52         Lew=seq(min(red$Lew),max(red$Lew),length=100)
53         FS=predict(FS.loess,Lew)
54         GRR=Lew/struct$L
55         plot(GRR,1-FS,main=main,type='l',...)
56         grid()    }
57     if (mode=='fixed'){
58         D=seq(min(red$D),max(red$D),length=100)
59         FS=predict(FS.loess,D)
60         GRR=D/struct$L
61         plot(GRR,1-FS,main=main,type='l',...)
62         grid()    }
63 }
64 )

```

window

```

1  setMethod('[',
2      signature='Meteo',
3      definition=function(x, i, j,...){
4          if (!missing(i)) {
5              i <- truncDay(i)
6          } else {
7              i <- indexD(x)[1]
8          }
9          if (!missing(j)) {
10             j <- truncDay(j)+86400-1 ##The end is the last second of the day
11         } else {
12             nDays <- length(indexD(x))
13             j <- indexD(x)[nDays]+86400-1
14         }
15         stopifnot(j>i)
16         if (!is.null(i)) i <- truncDay(i)
17         if (!is.null(j)) j <- truncDay(j)+86400-1
18         d <- indexD(x)
19         x@data <- x@data[(d >= i & d <= j)]
20         x
21     }
22 )
23
24
25 setMethod('[',
26     signature='Sol',
27     definition=function(x, i, j, ...){

```

```

28     if (!missing(i)) {
29         i <- truncDay(i)
30     } else {
31         i <- indexD(x)[1]
32     }
33     if (!missing(j)) {
34         j <- truncDay(j)+86400-1##The end is the last second of the day
35     } else {
36         nDays <- length(indexD(x))
37         j <- indexD(x)[nDays]+86400-1
38     }
39     stopifnot(j>i)
40     if(!is.null(i)) i <- truncDay(i)
41     if(!is.null(j)) j <- truncDay(j)
42     d1 <- indexD(x)
43     d2 <- indexI(x)
44     x@solD <- x@solD[(d1 >= i & d1 <= j)]
45     x@solI <- x@solI[(d2 >= i & d2 <= j)]
46     x
47 }
48 )
49
50 setMethod('[',
51     signature='G0',
52     definition=function(x, i, j, ...){
53         sol <- as(x, 'Sol')[i=i, j=j, ...] ##Sol method
54         meteo <- as(x, 'Meteo')[i=i, j=j, ...] ##Meteo method
55         i <- indexI(sol)[1]
56         j <- indexI(sol)[length(indexI(sol))]
57         d1 <- indexD(x)
58         d2 <- indexI(x)
59         G0Iw <- x@G0I[(d2 >= i & d2 <= j)]
60         Taw <- x@Ta[(d2 >= i & d2 <= j)]
61         G0dw <- x@G0D[(d1 >= truncDay(i) & d1 <= truncDay(j))]
62         G0dmw <- G0dw[, lapply(.SD/1000, mean, na.rm= TRUE),
63             .SDcols = c('G0d', 'D0d', 'B0d'),
64             by = .(month(Dates), year(Dates))]
65         if (x@type=='prom'){
66             G0dmw[, DayOfMonth := DOM(G0dmw)]
67             G0yw <- G0dmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
68                 .SDcols = c('G0d', 'D0d', 'B0d'),
69                 by = .(Dates = year)]
70             G0dmw[, DayOfMonth := NULL]
71         } else {
72             G0yw <- G0dw[, lapply(.SD/1000, sum, na.rm = TRUE),
73                 .SDcols = c('G0d', 'D0d', 'B0d'),
74                 by = .(Dates = year(unique(truncDay(Dates))))]
75         }
76         G0dmw[, Dates := paste(month.abb[month], year, sep = '. ')]
77         G0dmw[, c('month', 'year') := NULL]
78         setcolorder(G0dmw, 'Dates')
79         result <- new('G0',
80             meteo,
81             sol,
82             G0D=G0dw,
83             G0dm=G0dmw,
84             G0y=G0yw,
85             G0I=G0Iw,

```

```

86         Ta=Taw)
87     result
88 }
89 )
90
91
92 setMethod('[',
93     signature='Gef',
94     definition=function(x, i, j, ...){
95         g0 <- as(x, 'GO')[i=i, j=j, ...] ##GO method
96         i <- indexI(g0)[1]
97         j <- indexI(g0)[length(indexI(g0))]
98         d1 <- indexD(x)
99         d2 <- indexI(x)
100         GefIw <- x@GefI[(d2 >= i & d2 <= j)]
101         Thetaw <- x@Theta[(d2 >= i & d2 <= j)]
102         Gefdw <- x@GefD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
103         nms <- c('Bod', 'Bnd', 'Gd', 'Dd',
104                 'Bd', 'Gefd', 'Defd', 'Befd')
105         Gefdmw <- Gefdw[, lapply(.SD/1000, mean, na.rm = TRUE),
106                             .SDcols = nms,
107                             by = .(month(Dates), year(Dates))]
108         if (x@type=='prom'){
109             Gefdmw[, DayOfMonth:= DOM(Gefdmw)]
110             Gefyw <- Gefdmw[, lapply(.SD*DayOfMonth, sum),
111                                 .SDcols = nms,
112                                 by = .(Dates = year)]
113             Gefdmw[, DayOfMonth := NULL]
114         } else {
115             Gefyw <- Gefdw[, lapply(.SD/1000, sum, na.rm = TRUE),
116                                 .SDcols = nms,
117                                 by = .(Dates = year)]
118         }
119         Gefdmw[, Dates := paste(month.abb[month], year, sep = '. ')]
120         Gefdmw[, c('month', 'year') := NULL]
121         setcolorder(Gefdmw, 'Dates')
122         result <- new('Gef',
123             g0,
124             GefD=Gefdw,
125             Gefdm=Gefdmw,
126             Gefy=Gefyw,
127             GefI=GefIw,
128             Theta=Thetaw,
129             iS=x@iS,
130             alb=x@alb,
131             modeTrk=x@modeTrk,
132             modeShd=x@modeShd,
133             angGen=x@angGen,
134             struct=x@struct,
135             distances=x@distances
136         )
137     result
138 }
139 )
140
141
142 setMethod('[',
143     signature='ProdGCPV',

```

```

144     definition=function(x, i, j, ...){
145         gef <- as(x, 'Gef')[i=i, j=j, ...] ##Gef method
146         i <- indexI(gef)[1]
147         j <- indexI(gef)[length(indexI(gef))]
148         d1 <- indexD(x)
149         d2 <- indexI(x)
150         prodIw <- x@prodI[(d2 >= i & d2 <= j)]
151         prodDw <- x@prodD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
152         prodDmw <- prodDw[, lapply(.SD/1000, mean, na.rm = TRUE),
153                                .SDcols = c('Eac', 'Edc'),
154                                by = .(month(Dates), year(Dates))]
155         prodDmw$Yf <- prodDw$Yf
156         if (x@type=='prom'){
157             prodDmw[, DayOfMonth := DOM(prodDmw)]
158             prodyw <- prodDmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
159                                .SDcols = c('Eac', 'Edc', 'Yf'),
160                                by = .(Dates = year)]
161             prodDmw[, DayOfMonth := NULL]
162         } else {
163             prodyw <- prodDw[, lapply(.SD/1000, sum, na.rm = TRUE),
164                                .SDcols = c('Eac', 'Edc', 'Yf'),
165                                by = .(Dates = year)]
166         }
167         prodDmw[, Dates := paste(month.abb[month], year, sep = ' ')]
168         prodDmw[, c('month', 'year') := NULL]
169         setcolorder(prodDmw, c('Dates', names(prodDmw)[-length(prodDmw)]))
170         result <- new('ProdGCPV',
171                      gef,
172                      prodD=prodDw,
173                      prodDm=prodDmw,
174                      prody=prodyw,
175                      prodI=prodIw,
176                      module=x@module,
177                      generator=x@generator,
178                      inverter=x@inverter,
179                      effSys=x@effSys
180                      )
181         result
182     }
183 )
184
185 setMethod('[',
186           signature='ProdPVPS',
187           definition=function(x, i, j, ...){
188             gef <- as(x, 'Gef')[i=i, j=j, ...] ##Gef method
189             i <- indexI(gef)[1]
190             j <- indexI(gef)[length(indexI(gef))]
191             d1 <- indexD(x)
192             d2 <- indexI(x)
193             prodIw <- x@prodI[(d2 >= i & d2 <= j)]
194             prodDw <- x@prodD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
195             prodDmw <- prodDw[, .(Eac = Eac/1000,
196                                Qd = Qd,
197                                Yf = Yf),
198                                by = .(month(Dates), year(Dates))]
199             if (x@type=='prom'){
200                 prodDmw[, DayOfMonth := DOM(prodDmw)]
201                 prodyw <- prodDmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),

```

```

202         .SDcols = c('Eac', 'Qd', 'Yf'),
203         by = .(Dates = year)]
204     prodDmw[, DayOfMonth := NULL]
205 } else {
206     prodyw <- prodDw[, .(Eac = sum(Eac, na.rm = TRUE)/1000,
207         Qd = sum(Qd, na.rm = TRUE),
208         Yf = sum(Yf, na.rm = TRUE)),
209         by = .(Dates = year)]
210 }
211 prodDmw[, Dates := paste(month.abb[month], year, sep = '. ')]
212 prodDmw[, c('month', 'year') := NULL]
213 setcolorder(prodDmw, c('Dates', names(prodDmw)[-length(prodDmw)]))
214 result <- new('ProdPVPS',
215     gef,
216     prodD=prodDw,
217     prodDm=prodDmw,
218     prody=prodyw,
219     prodI=prodIw,
220     pump=x@pump,
221     H=x@H,
222     Pg=x@Pg,
223     converter=x@converter,
224     effSys=x@effSys
225 )
226 result
227 }
228 )

```

writeSolar

```

1  setGeneric('writeSolar', function(object, file,
2      complete=FALSE, day=FALSE,
3      timeScales=c('i', 'd', 'm', 'y'), sep=',',
4      ...){
5      standardGeneric('writeSolar')})
6
7  setMethod('writeSolar', signature=(object='Sol'),
8      definition=function(object, file, complete=FALSE, day=FALSE,
9      timeScales=c('i', 'd', 'm', 'y'), sep=',', ...){
10     name <- strsplit(file, '\\.')[[1]][1]
11     ext <- strsplit(file, '\\.')[[1]][2]
12     timeScales <- match.arg(timeScales, several.ok=TRUE)
13     if ('i' %in% timeScales) {
14         zI <- as.data.tableI(object, complete=complete, day=day)
15         write.table(zI,
16             file=file, sep=sep, row.names = FALSE, ...)
17     }
18     if ('d' %in% timeScales) {
19         zD <- as.data.tableD(object, complete=complete, day = day)
20         write.table(zD,
21             file=paste(name, 'D', ext, sep='.'),
22             sep=sep, row.names = FALSE, ...)
23     }
24     if ('m' %in% timeScales) {
25         zM <- as.data.tableM(object, complete=complete, day = day)
26         write.table(zM,
27             file=paste(name, 'M', ext, sep='.'),

```



```

28         sep=sep, row.names = FALSE, ...)
29     }
30     if ('y' %in% timeScales) {
31         zY <- as.data.tableY(object, complete=complete, day = day)
32         write.table(zY,
33             file=paste(name, 'Y', ext, sep='.'),
34             sep=sep, row.names = FALSE, ...)
35     }
36 })

```

xyplot

```

1 #####
2 ## THEMES
3 #####
4 xscale.solar <- function(...){ans <- xscale.components.default(...); ans$top=FALSE;
5   ans}
6
7 yscale.solar <- function(...){ans <- yscale.components.default(...); ans$right=FALSE;
8   ans}
9
10 solaR.theme <- function(pch=19, cex=0.7, region=rev(brewer.pal(9, 'YlOrRd')), ...) {
11   theme <- custom.theme.2(pch=pch, cex=cex, region=region, ...)
12   theme$strip.background$col='transparent'
13   theme$strip.shingle$col='transparent'
14   theme$strip.border$col='transparent'
15   theme
16 }
17
18 solaR.theme.2 <- function(pch=19, cex=0.7, region=rev(brewer.pal(9, 'YlOrRd')), ...) {
19   theme <- custom.theme.2(pch=pch, cex=cex, region=region, ...)
20   theme$strip.background$col='lightgray'
21   theme$strip.shingle$col='lightgray'
22   theme
23 }
24
25 #####
26 ## XYPLOT
27 #####
28 setGeneric('xyplot')
29
30 setMethod('xyplot',
31   signature = c(x = 'data.frame', data = 'missing'),
32   definition = function(x, data,
33     par.settings = solaR.theme.2,
34     xscale.components=xscale.solar,
35     yscale.components=yscale.solar,
36     scales = list(y = 'free'),
37     ...){
38     N <- length(x)-1
39     x0 <- x[, lapply(.SD, as.numeric), by = Dates]
40     x0 <- melt(x0, id.vars = 'Dates')
41     x0$variable <- factor(x0$variable,
42       levels = rev(levels(factor(x0$variable))))
43     xyplot(value ~ Dates | variable, x0,
44       par.settings = par.settings,
45       xscale.components = xscale.components,
46       yscale.components = yscale.components,

```

```

44         scales = scales,
45         type = 'l', layout = c(1,N),
46         ...)
47     })
48
49 setMethod('xyplot',
50   signature=c(x='formula', data='Meteo'),
51   definition=function(x, data,
52     par.settings=solaR.theme,
53     xscale.components=xscale.solar,
54     yscale.components=yscale.solar,
55     ...){
56     data0=getData(data)
57     xyplot(x, data0,
58       par.settings = par.settings,
59       xscale.components = xscale.components,
60       yscale.components = yscale.components,
61       strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
62   }
63 )
64
65 setMethod('xyplot',
66   signature=c(x='formula', data='Sol'),
67   definition=function(x, data,
68     par.settings=solaR.theme,
69     xscale.components=xscale.solar,
70     yscale.components=yscale.solar,
71     ...){
72     data0=as.data.tableI(data, complete=TRUE, day=TRUE)
73     data0[, w := h2r(hms(Dates)-12)]
74     xyplot(x, data0,
75       par.settings = par.settings,
76       xscale.components = xscale.components,
77       yscale.components = yscale.components,
78       strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
79   }
80 )
81
82 setMethod('xyplot',
83   signature=c(x='formula', data='G0'),
84   definition=function(x, data,
85     par.settings=solaR.theme,
86     xscale.components=xscale.solar,
87     yscale.components=yscale.solar,
88     ...){
89     data0=as.data.tableI(data, complete=TRUE, day=TRUE)
90     xyplot(x, data0,
91       par.settings = par.settings,
92       xscale.components = xscale.components,
93       yscale.components = yscale.components,
94       strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
95   }
96 )
97
98 setMethod('xyplot',
99   signature=c(x='formula', data='Shade'),
100  definition=function(x, data,
101    par.settings=solaR.theme,

```

```

102         xscale.components=xscale.solar,
103         yscale.components=yscale.solar,
104         ...){
105     data0=as.data.table(data)
106     xyplot(x, data0,
107           par.settings = par.settings,
108           xscale.components = xscale.components,
109           yscale.components = yscale.components,
110           strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
111     }
112   )
113
114   setMethod('xyplot',
115     signature=c(x='Meteo', data='missing'),
116     definition=function(x, data,
117       ...){
118       x0=getData(x)
119       xyplot(x0,
120             scales=list(cex=0.6, rot=0, y='free'),
121             strip=FALSE, strip.left=TRUE,
122             par.strip.text=list(cex=0.6),
123             ylab = '',
124             ...)
125     }
126   )
127
128   setMethod('xyplot',
129     signature=c(x='G0', data='missing'),
130     definition=function(x, data, ...){
131       x0 <- as.data.tableD(x, complete=FALSE)
132       x0 <- melt(x0, id.vars = 'Dates')
133       xyplot(value~Dates, x0, groups = variable,
134             par.settings=solaR.theme.2,
135             xscale.components=xscale.solar,
136             yscale.components=yscale.solar,
137             superpose=TRUE,
138             auto.key=list(space='right'),
139             ylab='Wh/m\u00b2',
140             type = 'l',
141             ...)
142     }
143   )
144
145   setMethod('xyplot',
146     signature=c(x='ProdGCPV', data='missing'),
147     definition=function(x, data, ...){
148       x0 <- as.data.tableD(x, complete=FALSE)
149       xyplot(x0,
150             strip = FALSE, strip.left = TRUE,
151             ylab = '', ...)
152     }
153   )
154
155   setMethod('xyplot',
156     signature=c(x='ProdPVPS', data='missing'),
157     definition=function(x, data, ...){
158       x0 <- as.data.tableD(x, complete=FALSE)
159       xyplot(x0,

```

```

160         strip = FALSE, strip.left = TRUE,
161         ylab = '', ...)
162     }
163 )

```

A.5. Conjunto de datos

aguiar

```

1 data(MTM)
2 Ktlim

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 0.031 0.058 0.051 0.052 0.028 0.053 0.044 0.085 0.010 0.319
[2,] 0.705 0.694 0.753 0.753 0.807 0.856 0.818 0.846 0.842 0.865

```

```

1 Ktmtm

[1] 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 1.00

```

```

1 head(MTM)

      V1      V2      V3      V4      V5      V6      V7      V8      V9 V10
1 0.229 0.333 0.208 0.042 0.083 0.042 0.042 0.021 0.000 0
2 0.167 0.319 0.194 0.139 0.097 0.028 0.042 0.000 0.014 0
3 0.250 0.250 0.091 0.136 0.091 0.046 0.046 0.023 0.068 0
4 0.158 0.237 0.158 0.263 0.026 0.053 0.079 0.026 0.000 0
5 0.211 0.053 0.211 0.158 0.053 0.053 0.158 0.105 0.000 0
6 0.125 0.125 0.250 0.188 0.063 0.125 0.000 0.125 0.000 0

```

SIAR

```

1 data(SIAR)
2 head(est_SIAR)

      Estacion Codigo      Longitud  Latitud  Altitud Fecha_Instalacion Fecha_Baja
1:      Villena  A01 -0.884444444 38.67639      519      1999-11-09 2000-03-19
2: Camp de Mirra A02 -0.772777778 38.67917      589      1999-11-09      <NA>
3:  Vila Joiosa A03 -0.256111111 38.52778      73      1999-11-10      <NA>
4:      Ondara  A04 0.006388889 38.81833      38      1999-11-10      <NA>
5:  Dénia Gata  A05 0.082500000 38.79250      86      1999-11-15      <NA>
6:      Pinoso  A06 -1.060555556 38.42722     629      1999-11-14      <NA>

```

helios

```

1 data(helios)
2 head(helios)

      yyyy.mm.dd      G.O. TambMax TambMin
1 2009/01/01 980.14      11.77      6.31
2 2009/01/02 1671.80     15.08      7.27
3 2009/01/03 671.02      9.33      6.36
4 2009/01/04 2482.80     11.71      1.11
5 2009/01/05 1178.19      7.33     -1.54
6 2009/01/06 1722.31      7.77     -0.78

```

prodEx

```
1 data(prodEx)
2 head(prodEx)
```

| | Dates <Date> | 1 <num> | 2 <num> | 3 <num> | 4 <num> | 5 <num> | 6 <num> | 7 <num> | 8 <num> | 9 <num> |
|----|-----------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| 1: | 2007-07-02 | 8.874982 | 8.847533 | 7.173181 | 8.874982 | 8.920729 | 8.975626 | 8.948177 | 8.948177 | 8.948177 |
| 2: | 2007-07-03 | 8.710291 | 8.691992 | 8.655395 | 8.710291 | 8.737740 | 8.792637 | 8.774338 | 8.774338 | 8.746889 |
| 3: | 2007-07-04 | 8.746889 | 8.737740 | 8.865832 | 8.737740 | 8.765188 | 8.838384 | 8.810935 | 8.792637 | 8.801786 |
| 4: | 2007-07-05 | 8.280266 | 8.271117 | 8.408359 | 8.280266 | 8.344313 | 8.380911 | 8.353462 | 8.362612 | 8.316864 |
| 5: | 2007-07-06 | 8.399209 | 8.417508 | 8.509003 | 8.435807 | 8.490704 | 8.490704 | 8.499854 | 8.527302 | 8.472405 |
| 6: | 2007-07-07 | 8.197921 | 8.170473 | 8.335163 | 8.225370 | 8.243669 | 8.307715 | 8.298565 | 8.280266 | 8.243669 |

| | 10 <num> | 11 <num> | 12 <num> | 13 <num> | 14 <num> | 15 <num> | 16 <num> | 17 <num> | 18 <num> | 19 <num> | 20 <num> |
|----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1: | 8.984775 | 8.783487 | 8.865832 | 8.966476 | 8.884131 | 8.774338 | 8.829234 | 8.627946 | 8.911580 | 8.807886 | 6.505270 |
| 2: | 8.801786 | 8.545601 | 8.682843 | 8.774338 | 8.691992 | 8.591348 | 8.646245 | 8.426658 | 8.710291 | 8.563900 | 3.952569 |
| 3: | 8.829234 | 8.545601 | 8.618797 | 8.829234 | 8.719441 | 8.618797 | 8.664544 | 8.426658 | 8.728590 | 8.612697 | 6.331430 |
| 4: | 8.380911 | 8.179622 | 8.271117 | 8.353462 | 8.280266 | 8.207071 | 8.261968 | 8.188772 | 7.950886 | 8.222320 | 5.498829 |
| 5: | 8.509003 | 8.316864 | 8.426658 | 8.490704 | 8.435807 | 8.344313 | 8.408359 | 8.371761 | 8.463256 | 8.332113 | 6.551017 |
| 6: | 8.326014 | 8.152174 | 8.161323 | 8.316864 | 8.234519 | 8.143024 | 8.179622 | 8.170473 | 8.243669 | 8.161323 | 6.669960 |

| | 21 <num> | 22 <num> |
|----|-------------|-------------|
| 1: | 3.742131 | 3.980018 |
| 2: | 4.080662 | 3.238911 |
| 3: | 1.363270 | 1.043039 |
| 4: | 3.998316 | 2.461206 |
| 5: | 5.361587 | 4.959010 |
| 6: | 5.215195 | 4.922413 |

pumpCoef

```
1 data(pumpCoef)
2 head(pumpCoef)
```

| | Qn <int> | stages <int> | Qmax <num> | Pmn <int> | a <num> | b <num> | c <num> | g <num> | h <num> | i <num> | j <num> | k <num> | l <num> |
|----|-------------|-----------------|---------------|--------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| 1: | 2 | 6 | 2.6 | 370 | 0.01409736 | 0.018576 | -3.6324 | -0.32 | 0.74 | 0.22 | -0.1614 | 0.5247 | 0.0694 |
| 2: | 2 | 9 | 2.6 | 370 | 0.02114604 | 0.027864 | -5.4486 | -0.32 | 0.74 | 0.22 | -0.1614 | 0.5247 | 0.0694 |
| 3: | 2 | 13 | 2.6 | 550 | 0.03054428 | 0.040248 | -7.8702 | -0.12 | 0.49 | 0.27 | -0.1614 | 0.5247 | 0.0694 |
| 4: | 2 | 18 | 2.6 | 750 | 0.04229208 | 0.055728 | -10.8972 | -0.16 | 0.42 | 0.47 | -0.1614 | 0.5247 | 0.0694 |
| 5: | 2 | 23 | 2.6 | 1100 | 0.05403988 | 0.071208 | -13.9242 | -0.20 | 0.51 | 0.42 | -0.1614 | 0.5247 | 0.0694 |
| 6: | 2 | 28 | 2.6 | 1500 | 0.06578768 | 0.086688 | -16.9512 | -0.24 | 0.50 | 0.49 | -0.1614 | 0.5247 | 0.0694 |

Bibliografía

- [Sta85] Richard Stallman. *GNU Emacs*. Un editor de texto extensible, personalizable, auto-documentado y en tiempo real. 1985. URL: <https://www.gnu.org/software/emacs/>.
- [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](https://doi.org/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>.
- [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](https://doi.org/10.18637/jss.v050.i09).
- [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 transición 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/>.

- [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>.
- [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/>.
- [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>.