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

TÌtulo: 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 Universidad 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

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

1.1. Objetivos

El objetivo principal de este proyecto es el desarrollo de un paquete en R[R C23] con el cual poder realizar estimaciones y representaciones gráficas de la 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 CALCG0].
  - Los componentes de radiación eficaz en el plano inclinado [CITAR CALCGEF].
  - La producción de sistemas fotovoltaicos conectados a red [CITAR PRODGCPV] y sistemas fotovoltaivos 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 prepador de textos LaTeX(partiendo de un archivo .org, se puede exportar a un archivo .tex para posteriormente exportar un pdf).
- Obtener conocimientos teóricos acerca de la radiación solar y de la producción de energía solar mediante sistemas fotovoltaicos y sus diversos tipos. Para ello se ha usado en mayor medida el libro “Energía Solar Fotovoltaica” [Per23].

1.2. Análisis previo de soluciones

Este proyecto, como ya se ha comentado, es el heredero del paquete `solaR` desarrollado por Oscar Perpiñán. La filosofía de ambos paquetes es la misma y los resultados que dan son muy similares. Sin embargo, lo que les diferencia es 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 diseañda por el **Grupo de Sistemas Fotovoltaicos del Instituto de Energía Solar de la Universidad Politécnica de Madrid**.

3. **PVGIS**
   Aplicación web desarrolada 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 [ref:sec:ejemplos] 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.

CAPÍTULO 2

Estado del arte

2.1. Situación actual de la generación fotovoltaica

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

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

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

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

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

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

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.

---

[1] UNEF: Unión Española Fotovoltaica.

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

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

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

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

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

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

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

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 reindustralización de la economía, aprovechando las medidas del REPower Plan, la Estrategia Solar y la Alianza de al 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

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

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

[5]PIB: Producto Interior Bruto.

CAPÍTULO 3

Parte teórica y desarrollo del código

...

CAPÍTULO 4

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

CAPÍTULO 5

Detalles de la programación

...

APÉNDICE A

Código completo

Todo el código que se muestra a continuación está disponible…

A.1. Constructores

calcSol

```
calcSol <- function(lat, BTd,
                    sample = 'hour', BTi,
                    EoT = TRUE,
                    keep.night = TRUE,
                    method = 'michalsky')
{
    if(missing(BTd)) BTd <- truncDay(BTi)
    solD <- fSolD(lat, BTd, method = method) #daily values
    solI <- fSolI(solD = solD, sample = sample, #intradaily values
                  BTi = BTi, keep.night = keep.night,
                  EoT = EoT, method = method)

    if(!missing(BTi)){
        sample <- solI$Dates[2]-solI$Dates[1]
        sample <- format(sample)
    }

    solD[, lat := NULL]
    solI[, lat := NULL]
    result <- new('Sol',
                  lat = lat,
                  solD = solD,
                  solI = solI,
                  sample = sample,
                  method = method)
    return(result)
}
```

calcG0

```
calcG0 <- function(lat,
```

```
                   modeRad='prom',
                   dataRad,
                   sample='hour',
                   keep.night=TRUE,
                   sunGeometry='michalsky',
                   corr, f, ...)
{

    if (missing(lat)) stop('lat missing. You must provide a latitude value.')

    stopifnot(modeRad %in% c('prom', 'aguiar','bd', 'bdI'))


###Datos de Radiacion
    if (missing(corr)){
        corr = switch(modeRad,
                      bd = 'CPR', #Correlation between Fd and Kt for daily values
                      aguiar = 'CPR', #Correlation between Fd and Kt for daily values
                      prom = 'Page',  #Correlation between Fd and Kt for monthly averages
                      bdI = 'BRL'     #Correlation between fd and kt for intraday values
                      )
    }

    if(is(dataRad, 'Meteo')){BD <- dataRad}
    else{
    BD <- switch(modeRad,
                 bd = {
                         if (!is.list(dataRad)) dataRad <- list(file=dataRad)
                         switch(class(dataRad$file)[1],
                                character={
                                    bd.default=list(file='', lat=lat)
                                    bd=modifyList(bd.default, dataRad)
                                    res <- do.call('readBDd', bd)
                                    res
                                },
                                data.table= ,
                                data.frame={
                                    bd.default=list(file='', lat=lat)
                                    bd=modifyList(bd.default, dataRad)
                                    res <- do.call('dt2Meteo', bd)
                                    res
                                },
                                zoo={
                                    bd.default=list(file='', lat=lat, source='')
                                    bd=modifyList(bd.default, dataRad)
                                    res <- do.call('zoo2Meteo', bd)
                                    res
                                })
                     }, #End of bd
                 prom = {
                     if (!is.list(dataRad)) dataRad <- list(G0dm=dataRad)
                     prom.default <- list(G0dm=numeric(), lat=lat)
                     prom = modifyList(prom.default, dataRad)
```

```
                     res <- do.call('readG0dm', prom)
                 }, #End of prom
                 aguiar = {
                     if (is.list(dataRad)) dataRad <- dataRad$G0dm
                     BTd <- fBTd(mode='serie')
                     solD <- fSolD(lat, BTd)
                     G0d <- markovG0(dataRad, solD)
                     res <- dt2Meteo(G0d, lat=lat, source='aguiar')
                 }, #End of aguiar
                 bdI = {
                     if (!is.list(dataRad)) dataRad <- list(file=dataRad)
                     switch(class(dataRad$file)[1],
                            character = {
                                bdI.default <- list(file='', lat=lat)
                                bdI <- modifyList(bdI.default, dataRad)
                                res <- do.call('readBDi', bdI)
                                res
                            },
                            data.table = ,
                            data.frame = {
                                bdI.default <- list(file='', lat=lat)
                                bdI <- modifyList(bdI.default, dataRad)
                                res <- do.call('dt2Meteo', bdI)
                                res
                            },
                            zoo = {
                                bdI.default <- list(file='', lat=lat, source='')
                                bdI <- modifyList(bdI.default, dataRad)
                                res <- do.call('zoo2Meteo', bdI)
                                res
                            },
                            stop('dataRad$file should be a character, a data.table, a data.frame or a zoo.')
                            )} #End of btI
                 )             #End of general switch
    }


### Angulos solares y componentes de irradiancia
    if (modeRad=='bdI') {
        sol <- calcSol(lat, sample = sample,
                       BTi = indexD(BD), keep.night=keep.night, method=sunGeometry)
        compI <- fCompI(sol=sol, G0I=BD, corr=corr, f=f, ...)
        compD <- compI[, lapply(.SD, P2E, sol@sample),
                       .SDcols = c('G0', 'D0', 'B0'),
                       by = truncDay(Dates)]
        names(compD)[1] <- 'Dates'
        names(compD)[-1] <- paste(names(compD)[-1], 'd', sep = '')
        compD$Fd <- compD$D0d/compD$G0d
        compD$Kt <- compD$G0d/sol@solD$Bo0d
    } else { ##modeRad!='bdI'
        sol <- calcSol(lat, indexD(BD), sample = sample,
                       keep.night = keep.night, method = sunGeometry)
        compD<-fCompD(sol=sol, G0d=BD, corr=corr, f, ...)
        compI<-fCompI(sol=sol, compD=compD, ...)
```

```
    }

###Temperature

    Ta=switch(modeRad,
              bd={
                  if (all(c("TempMax","TempMin") %in% names(BD@data))) {
                      fTemp(sol, BD)
                  } else {
                      if ("Ta" %in% names(BD@data)) {
                          data.table(Dates = indexD(sol),
                                     Ta =BD@data$Ta)
                      } else {
                          warning('No temperature information available!')
                      }
                  }
              },
              bdI={
                  if ("Ta" %in% names(BD@data)) {
                      data.table(Dates = indexI(sol),
                                 Ta = BD@data$Ta)
                  } else {
                      warning('No temperature information available!')
                  }
              },
              prom={
                  if ("Ta" %in% names(BD@data)) {
                      data.table(Dates = indexD(sol),
                                 Ta = BD@data$Ta)
                  } else {
                      warning('No temperature information available!')
                  }
              },
              aguiar={
                  data.table(Dates = indexI(sol),
                             Ta = BD@data$Ta)
              }
              )

###Medias mensuales y anuales
    nms <- c('G0d', 'D0d', 'B0d')
    G0dm <- compD[, lapply(.SD/1000, mean, na.rm = TRUE),
                  .SDcols = nms,
                  by = .(month(Dates), year(Dates))]

    if(modeRad == 'prom'){
        G0dm[, DayOfMonth := DOM(G0dm)]
        G0y <- G0dm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                    .SDcols = nms,
                    by = .(Dates = year)]
        G0dm[, DayOfMonth := NULL]
    } else{
        G0y <- compD[, lapply(.SD/1000, sum, na.rm = TRUE),
                     .SDcols = nms,
                     by = .(Dates = year(Dates))]
    }
```

```
    G0dm[, Dates := paste(month.abb[month], year, sep = '. ')]
    G0dm[, c('month', 'year') := NULL]
    setcolorder(G0dm, 'Dates')

###Result
    result <- new(Class='G0',
                  BD,          #G0 contains "Meteo"
                  sol,         #G0 contains 'Sol'
                  G0D=compD, #results of fCompD
                  G0dm=G0dm, #monthly means
                  G0y=G0y,   #yearly values
                  G0I=compI, #results of fCompD
                  Ta=Ta      #ambient temperature
                  )
    return(result)
}
```

calcGef

```
calcGef<-function(lat,
                  modeTrk='fixed',      #c('two','horiz','fixed')
                  modeRad='prom',
                  dataRad,
                  sample='hour',
                  keep.night=TRUE,
                  sunGeometry='michalsky',
                  corr, f,
                  betaLim=90, beta=abs(lat)-10, alfa=0,
                  iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
                  modeShd='',    #modeShd=c('area','bt','prom')
                  struct=list(), #list(W=23.11, L=9.8, Nrow=2, Ncol=8),
                  distances=data.frame(),#data.table(Lew=40, Lns=30, H=0)){
                  ...){

    stopifnot(is.list(struct), is.data.frame(distances))

    if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
        modeShd[which(modeShd=='bt')]='area'
        warning('backtracking is only implemented for modeTrk=horiz')}

    if (modeRad!='prev'){ #not use a prev calculation
        radHoriz <- calcG0(lat=lat, modeRad=modeRad,
                           dataRad=dataRad,
                           sample=sample, keep.night=keep.night,
                           sunGeometry=sunGeometry,
                           corr=corr, f=f, ...)
    } else {                        #use a prev calculation
        radHoriz <- as(dataRad, 'G0')
    }

### Inclined and effective radiation
    BT=("bt" %in% modeShd)
    angGen <- fTheta(radHoriz, beta, alfa, modeTrk, betaLim, BT, struct,
    distances)
    inclin <- fInclin(radHoriz, angGen, iS, alb, horizBright, HCPV)
```

```

### Daily, monthly and yearly values
    by <- radHoriz@sample
    nms <- c('Bo', 'Bn', 'G', 'D', 'B', 'Gef', 'Def', 'Bef')
    nmsd <- paste(nms, 'd', sep = '')


    if(radHoriz@type == 'prom'){
        Gefdm <- inclin[, lapply(.SD/1000, P2E, by),
                        .SDcols = nms,
                        by = .(month(Dates), year(Dates))]
        names(Gefdm)[-c(1,2)] <- nmsd
        GefD <- Gefdm[, .SD*1000,
                      .SDcols = nmsd,
                      by = .(Dates = indexD(radHoriz))]

        Gefdm[, DayOfMonth := DOM(Gefdm)]
        Gefy <- Gefdm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                      .SDcols = nmsd,
                      by = .(Dates = year)]
        Gefdm[, DayOfMonth := NULL]
    } else{
        GefD <- inclin[, lapply(.SD, P2E, by),
                       .SDcols = nms,
                       by = .(Dates = truncDay(Dates))]
        names(GefD)[-1] <- nmsd

        Gefdm <- GefD[, lapply(.SD/1000, mean, na.rm = TRUE),
                      .SDcols = nmsd,
                      by = .(month(indexD(radHoriz)), year(indexD(radHoriz)))]
        Gefy <- GefD[, lapply(.SD/1000, sum, na.rm = TRUE),
                     .SDcols = nmsd,
                     by = .(Dates = year(indexD(radHoriz)))]
    }

    Gefdm[, Dates := paste(month.abb[month], year, sep = '. ')]
    Gefdm[, c('month', 'year') := NULL]
    setcolorder(Gefdm, 'Dates')

###Resultado antes de sombras
    result0=new('Gef',
                radHoriz,                           #Gef contains 'G0'
                Theta=angGen,
                GefD=GefD,
                Gefdm=Gefdm,
                Gefy=Gefy,
                GefI=inclin,
                iS=iS,
                alb=alb,
                modeTrk=modeTrk,
                modeShd=modeShd,
                angGen=list(alfa=alfa, beta=beta, betaLim=betaLim),
                struct=struct,
                distances=distances
                )
###Shadows
```

```
    if (isTRUE(modeShd == "") ||        #If modeShd=='' there is no shadow
    calculation
        ('bt' %in% modeShd)) {           #nor if there is backtracking
        return(result0)
    } else {
        result <- calcShd(result0, modeTrk, modeShd, struct, distances)
        return(result)
    }
}
```

prodGCPV

```
prodGCPV<-function(lat,
                   modeTrk='fixed',
                   modeRad='prom',
                   dataRad,
                   sample='hour',
                   keep.night=TRUE,
                   sunGeometry='michalsky',
                   corr, f,
                   betaLim=90, beta=abs(lat)-10, alfa=0,
                   iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
                   module=list(),
                   generator=list(),
                   inverter=list(),
                   effSys=list(),
                   modeShd='',
                   struct=list(),
                   distances=data.table(),
                   ...){

    stopifnot(is.list(module),
              is.list(generator),
              is.list(inverter),
              is.list(effSys),
              is.list(struct),
              is.data.table(distances))

  if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
      modeShd[which(modeShd=='bt')]='area'
      warning('backtracking is only implemented for modeTrk=horiz')}

    if (modeRad!='prev'){ #We do not use a previous calculation

    radEf<-calcGef(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
                   dataRad=dataRad,
                   sample=sample, keep.night=keep.night,
                   sunGeometry=sunGeometry,
                   corr=corr, f=f,
                   betaLim=betaLim, beta=beta, alfa=alfa,
                   iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
                   modeShd=modeShd, struct=struct, distances=distances, ...)

  } else { #We use a previous calcG0, calcGef or prodGCPV calculation.

```

```
    stopifnot(class(dataRad) %in% c('G0', 'Gef', 'ProdGCPV'))
    radEf <- switch(class(dataRad),
                    G0=calcGef(lat=lat,
                               modeTrk=modeTrk, modeRad='prev',
                               dataRad=dataRad,
                               betaLim=betaLim, beta=beta, alfa=alfa,
                               iS=iS, alb=alb, horizBright=horizBright, HCPV=
  HCPV,
                               modeShd=modeShd, struct=struct, distances=
  distances, ...),
                    Gef=dataRad,
                    ProdGCPV=as(dataRad, 'Gef')
                    )
}


  ##Production
  prodI<-fProd(radEf,module,generator,inverter,effSys)
  module=attr(prodI, 'module')
  generator=attr(prodI, 'generator')
  inverter=attr(prodI, 'inverter')
  effSys=attr(prodI, 'effSys')

  ##Calculation of daily, monthly and annual values
  Pg=generator$Pg #Wp

  by <- radEf@sample
  nms1 <- c('Pac', 'Pdc')
  nms2 <- c('Eac', 'Edc', 'Yf')


  if(radEf@type == 'prom'){
      prodDm <- prodI[, lapply(.SD/1000, P2E, by),
                      .SDcols = nms1,
                      by = .(month(Dates), year(Dates))]
      names(prodDm)[-c(1,2)] <- nms2[-3]
      prodDm[, Yf := Eac/(Pg/1000)]
      prodD <- prodDm[, .SD*1000,
                      .SDcols = nms2,
                      by = .(Dates = indexD(radEf))]
      prodD[, Yf := Yf/1000]

      prodDm[, DayOfMonth := DOM(prodDm)]
      prody <- prodDm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                      .SDcols = nms2,
                      by = .(Dates = year)]
      prodDm[, DayOfMonth := NULL]
  } else {
      prodD <- prodI[, lapply(.SD, P2E, by),
                     .SDcols = nms1,
                     by = .(Dates = truncDay(Dates))]
      names(prodD)[-1] <- nms2[-3]
      prodD[, Yf := Eac/Pg]

      prodDm <- prodD[, lapply(.SD/1000, mean, na.rm = TRUE),
                      .SDcols = nms2,
```

```
                         by = .(month(Dates), year(Dates))]
        prodDm[, Yf := Yf * 1000]
        prody <- prodD[, lapply(.SD/1000, sum, na.rm = TRUE),
                       .SDcols = nms2,
                       by = .(Dates = year(Dates))]
        prody[, Yf := Yf * 1000]
    }

    prodDm[, Dates := paste(month.abb[month], year, sep = '. ')]
    prodDm[, c('month', 'year') := NULL]
    setcolorder(prodDm, 'Dates')

    result <- new('ProdGCPV',
                  radEf,                    #contains 'Gef'
                  prodD=prodD,
                  prodDm=prodDm,
                  prody=prody,
                  prodI=prodI,
                  module=module,
                  generator=generator,
                  inverter=inverter,
                  effSys=effSys
                  )
}
```

prodPVPS

```
prodPVPS<-function(lat,
                   modeTrk='fixed',
                   modeRad='prom',
                   dataRad,
                   sample='hour',
                   keep.night=TRUE,
                   sunGeometry='michalsky',
                   corr, f,
                   betaLim=90, beta=abs(lat)-10, alfa=0,
                   iS=2, alb=0.2, horizBright=TRUE, HCPV=FALSE,
                   pump , H,
                   Pg, converter= list(), #Pnom=Pg, Ki=c(0.01,0.025,0.05)),
                   effSys=list(),
                   ...){

    stopifnot(is.list(converter),
              is.list(effSys))

    if (modeRad!='prev'){ #We do not use a previous calculation

        radEf<-calcGef(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
                       dataRad=dataRad,
                       sample=sample, keep.night=keep.night,
                       sunGeometry=sunGeometry,
                       corr=corr, f=f,
                       betaLim=betaLim, beta=beta, alfa=alfa,
                       iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
                   modeShd='', ...)
```

```
    } else { #We use a previous calculation of calcG0, calcGef or prodPVPS
        stopifnot(class(dataRad) %in% c('G0', 'Gef', 'ProdPVPS'))
        radEf <- switch(class(dataRad),
                        G0=calcGef(lat=lat,
                                   modeTrk=modeTrk, modeRad='prev',
                      dataRad=dataRad,
                      betaLim=betaLim, beta=beta, alfa=alfa,
                      iS=iS, alb=alb, horizBright=horizBright, HCPV=HCPV,
                      modeShd='', ...),
                      Gef=dataRad,
                      ProdPVPS=as(dataRad, 'Gef')
                      )
    }

###Electric production
    converter.default=list(Ki = c(0.01,0.025,0.05), Pnom=Pg)
    converter=modifyList(converter.default, converter)

    effSys.default=list(ModQual=3,ModDisp=2,OhmDC=1.5,OhmAC=1.5,MPP=1,TrafoMT=1, Disp=0.5)
    effSys=modifyList(effSys.default, effSys)

    TONC=47
    Ct=(TONC-20)/800
    lambda=0.0045
    Gef=radEf@GefI$Gef
    night=radEf@solI$night
    Ta=radEf@Ta$Ta

    Tc=Ta+Ct*Gef
    Pdc=Pg*Gef/1000*(1-lambda*(Tc-25))
    Pdc[is.na(Pdc)]=0 #Necessary for the functions provided by fPump
    PdcN=with(effSys,
              Pdc/converter$Pnom*(1-ModQual/100)*(1-ModDisp/100)*(1-OhmDC/100)
              )
    PacN=with(converter,{
        A=Ki[3]
        B=Ki[2]+1
        C=Ki[1]-(PdcN)
        ##AC power normalized to the inverter
        result=(-B+sqrt(B^2-4*A*C))/(2*A)
    })
    PacN[PacN<0]<-0

    Pac=with(converter,
             PacN*Pnom*(1-effSys$OhmAC/100))
    Pdc=PdcN*converter$Pnom*(Pac>0)


###Pump
    fun<-fPump(pump=pump, H=H)
    ##I limit power to the pump operating range.
    rango=with(fun,Pac>=lim[1] & Pac<=lim[2])
    Pac[!rango]<-0
    Pdc[!rango]<-0
```

```
    prodI=data.table(Pac=Pac,Pdc=Pdc,Q=0,Pb=0,Ph=0,f=0)
    prodI=within(prodI,{
        Q[rango]<-fun$fQ(Pac[rango])
        Pb[rango]<-fun$fPb(Pac[rango])
        Ph[rango]<-fun$fPh(Pac[rango])
        f[rango]<-fun$fFreq(Pac[rango])
        etam=Pb/Pac
        etab=Ph/Pb
    })

    prodI[night,]<-NA
    prodI[, Dates := indexI(radEf)]
    setcolorder(prodI, c('Dates', names(prodI)[-length(prodI)]))

###daily, monthly and yearly values

    by <- radEf@sample

    if(radEf@type == 'prom'){
        prodDm <- prodI[, .(Eac = P2E(Pac, by)/1000,
                            Qd = P2E(Q, by)),
                        by = .(month(Dates), year(Dates))]
        prodDm[, Yf := Eac/(Pg/1000)]

        prodD <- prodDm[, .(Eac = Eac*1000,
                            Qd,
                            Yf),
                        by = .(Dates = indexD(radEf))]

        prodDm[, DayOfMonth := DOM(prodDm)]

        prody <- prodDm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                        .SDcols = c('Eac', 'Qd', 'Yf'),
                        by = .(Dates = year)]
        prodDm[, DayOfMonth := NULL]
    } else {
        prodD <- prodI[, .(Eac = P2E(Pac, by)/1000,
                           Qd = P2E(Q, by)),
                       by = .(Dates = truncDay(Dates))]
        prodD[, Yf := Eac/Pg*1000]

        prodDm <- prodD[, lapply(.SD, mean, na.rm = TRUE),
                        .SDcols = c('Eac','Qd', 'Yf'),
                        by = .(month(Dates), year(Dates))]
        prody <- prodD[, lapply(.SD, sum, na.rm = TRUE),
                       .SDcols = c('Eac', 'Qd', 'Yf'),
                       by = .(Dates = year(Dates))]

    }

    prodDm[, Dates := paste(month.abb[month], year, sep = '. ')]
    prodDm[, c('month', 'year') := NULL]
    setcolorder(prodDm, 'Dates')

    result <- new('ProdPVPS',
                  radEf,                  #contains 'Gef'
```

```
                  prodD=prodD,
                  prodDm=prodDm,
                  prody=prody,
                  prodI=prodI,
                  pump=pump,
                  H=H,
                  Pg=Pg,
                  converter=converter,
                  effSys=effSys
                )
}
```

calcShd

```
calcShd<-function(radEf,##class='Gef'
                  modeTrk='fixed',      #c('two','horiz','fixed')
                  modeShd='prom',       #modeShd=c('area','bt','prom')
                  struct=list(), #list(W=23.11, L=9.8, Nrow=2, Ncol=8),
                  distances=data.frame() #data.table(Lew=40, Lns=30, H=0)){
                  )
{
    stopifnot(is.list(struct), is.data.frame(distances))

    ##For now I only use modeShd = 'area'
    ##With different modeShd (to be defined) I will be able to calculate Gef in a
     different way
    ##See macagnan thesis
    prom=("prom"  %in%  modeShd)
    prev <- as.data.tableI(radEf, complete=TRUE)
    ## shadow calculations
    sol <- data.table(AzS = prev$AzS,
                      AlS = prev$AlS)
    theta <- radEf@Theta
    AngGen <- data.table(theta, sol)
    FS <- fSombra(AngGen, distances, struct, modeTrk, prom)
    ## irradiance calculation
    gef0 <- radEf@GefI
    Bef0 <- gef0$Bef
    Dcef0 <- gef0$Dcef
    Gef0 <- gef0$Gef
    Dief0 <- gef0$Dief
    Ref0 <- gef0$Ref
    ## calculation
    Bef <- Bef0*(1-FS)
    Dcef <- Dcef0*(1-FS)
    Def <- Dief0+Dcef
    Gef <- Dief0+Ref0+Bef+Dcef #Including shadows
    ##Change names
    nms <- c('Gef', 'Def', 'Dcef', 'Bef')
    nmsIndex <- which(names(gef0) %in% nms)
    names(gef0)[nmsIndex]<- paste(names(gef0)[nmsIndex], '0', sep='')
    GefShd <- gef0
    GefShd[, c(nms, 'FS') := .(Gef, Def, Dcef, Bef, FS)]

    ## daily, monthly and yearly values
```

```
    by <- radEf@sample
    nms <- c('Gef0', 'Def0', 'Bef0', 'G', 'D', 'B', 'Gef', 'Def', 'Bef')
    nmsd <- paste(nms, 'd', sep = '')

    Gefdm <- GefShd[, lapply(.SD/1000, P2E, by),
                    by = .(month(truncDay(Dates)), year(truncDay(Dates))),
                    .SDcols = nms]
    names(Gefdm)[-c(1, 2)] <- nmsd

    if(radEf@type == 'prom'){
        GefD <- Gefdm[, .SD[, -c(1, 2)] * 1000,
                      .SDcols = nmsd,
                      by = .(Dates = indexD(radEf))]

        Gefdm[, DayOfMonth := DOM(Gefdm)]

        Gefy <- Gefdm[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                      .SDcols = nmsd,
                      by = .(Dates = year)]
        Gefdm[, DayOfMonth := NULL]
    } else{
        GefD <- GefShd[, lapply(.SD/1000, P2E, by),
                       .SDcols = nms,
                       by = .(Dates = truncDay(Dates))]
        names(GefD)[-1] <- nmsd

        Gefy <- GefD[, lapply(.SD[, -1], sum, na.rm = TRUE),
                     .SDcols = nmsd,
                     by = .(Dates = year(Dates))]
    }

    Gefdm[, Dates := paste(month.abb[month], year, sep = '. ')]
    Gefdm[, c('month', 'year') := NULL]
    setcolorder(Gefdm, c('Dates', names(Gefdm)[-length(Gefdm)]))

    ## Object of class Gef
    ## modifying the 'modeShd', 'GefI', 'GefD', 'Gefdm', and 'Gefy' slots
    ## from the original radEf object
    radEf@modeShd=modeShd
    radEf@GefI=GefShd
    radEf@GefD=GefD
    radEf@Gefdm=Gefdm
    radEf@Gefy=Gefy
    return(radEf)
}
```

optimShd

```
optimShd<-function(lat,
                   modeTrk='fixed',
                   modeRad='prom',
                   dataRad,
                   sample='hour',
                   keep.night=TRUE,
                   sunGeometry='michalsky',
```

```
                     betaLim=90, beta=abs(lat)-10, alfa=0,
                     iS=2, alb=0.2, HCPV=FALSE,
                     module=list(),
                     generator=list(),
                     inverter=list(),
                     effSys=list(),
                     modeShd='',
                     struct=list(),
                     distances=data.table(),
                     res=2,      #resolution, distance spacing
                     prog=TRUE){ #Drawing progress bar

    if (('bt' %in% modeShd) & (modeTrk!='horiz')) {
        modeShd[which(modeShd=='bt')]='area'
        warning('backtracking is only implemented for modeTrk=horiz')}

    ##I save function arguments for later use

    listArgs<-list(lat=lat, modeTrk=modeTrk, modeRad=modeRad,
                   dataRad=dataRad,
                   sample=sample, keep.night=keep.night,
                   sunGeometry=sunGeometry,
                   betaLim=betaLim, beta=beta, alfa=alfa,
                   iS=iS, alb=alb, HCPV=HCPV,
                   module=module, generator=generator,
                   inverter=inverter, effSys=effSys,
                   modeShd=modeShd, struct=struct,
                   distances=data.table(Lew=NA, Lns=NA, D=NA))


    ##I think network on which I will do the calculations
    Red=switch(modeTrk,
               horiz=with(distances,
                          data.table(Lew=seq(Lew[1],Lew[2],by=res),
                                     H=0)),
               two=with(distances,
                        data.table(
                        expand.grid(Lew=seq(Lew[1],Lew[2],by=res),
                                    Lns=seq(Lns[1],Lns[2],by=res),
                                    H=0))),
               fixed=with(distances,
                          data.table(D=seq(D[1],D[2],by=res),
                                     H=0))
    )

    casos<-dim(Red)[1] #Number of possibilities to study

    ##I prepare the progress bar
    if (prog) {pb <- txtProgressBar(min = 0, max = casos+1, style = 3)
        setTxtProgressBar(pb, 0)}

###Calculations
    ##Reference: No shadows
    listArgs0 <- modifyList(listArgs,
                            list(modeShd='', struct=NULL, distances=NULL) )
    Prod0<-do.call(prodGCPV, listArgs0)
```

```
    YfAnual0=mean(Prod0@prody$Yf) #I use mean in case there are several years
    if (prog) {setTxtProgressBar(pb, 1)}

    ##The loop begins

    ##I create an empty vector of the same length as the cases to be studied
    YfAnual<-numeric(casos)

    BT=('bt' %in% modeShd)
    if (BT) { ##There is backtracking, then I must start from horizontal
    radiation.
        RadBT <- as(Prod0, 'G0')
        for (i in seq_len(casos)){
            listArgsBT <- modifyList(listArgs,
                                     list(modeRad='prev', dataRad=RadBT,
                                          distances=Red[i,]))
            prod.i <- do.call(prodGCPV, listArgsBT)
            YfAnual[i]=mean(prod.i@prody$Yf)
            if (prog) {setTxtProgressBar(pb, i+1)}
        }
    } else {
        prom=('prom' %in% modeShd)
        for (i in seq_len(casos)){
            Gef0=as(Prod0, 'Gef')
            GefShd=calcShd(Gef0, modeTrk=modeTrk, modeShd=modeShd,
                           struct=struct, distances=Red[i,])
            listArgsShd <- modifyList(listArgs,
                                      list(modeRad='prev', dataRad=GefShd)
                                      )
            prod.i <- do.call(prodGCPV, listArgsShd)
            YfAnual[i]=mean(prod.i@prody$Yf)
            if (prog) {setTxtProgressBar(pb, i+1)}
        }
    }
    if (prog) {close(pb)}


###Results
    FS=1-YfAnual/YfAnual0
    GRR=switch(modeTrk,
               two=with(Red,Lew*Lns)/with(struct,L*W),
               fixed=Red$D/struct$L,
               horiz=Red$Lew/struct$L)
    SombraDF=data.table(Red,GRR,FS,Yf=YfAnual)
    FS.loess=switch(modeTrk,
                    two=loess(FS~Lew*Lns,data=SombraDF),
                    horiz=loess(FS~Lew,data=SombraDF),
                    fixed=loess(FS~D,data=SombraDF))
    Yf.loess=switch(modeTrk,
                    two=loess(Yf~Lew*Lns,data=SombraDF),
                    horiz=loess(Yf~Lew,data=SombraDF),
                    fixed=loess(Yf~D,data=SombraDF))
    result <- new('Shade',
                  Prod0, ##contains ProdGCPV
                  FS=FS,
                  GRR=GRR,
```

```
                  Yf=YfAnual,
                  FS.loess=FS.loess,
                  Yf.loess=Yf.loess,
                  modeShd=modeShd,
                  struct=struct,
                  distances=Red,
                  res=res
                  )
    result
}
```

meteoReaders

```
#### monthly means of irradiation ####
readG0dm <- function(G0dm, Ta = 25, lat = 0,
                     year = as.POSIXlt(Sys.Date())$year + 1900,
                     promDays = c(17, 14, 15, 15, 15, 10, 18, 18, 18, 19, 18, 13)
    ,
                     source = '')
{
    if(missing(lat)){lat <- 0}
    Dates <- as.IDate(paste(year, 1:12, promDays, sep = '-'), tz = 'UTC')
    G0dm.dt <- data.table(Dates = Dates,
                          G0d = G0dm,
                          Ta = Ta)
    setkey(G0dm.dt, 'Dates')
    results <- new(Class = 'Meteo',
                   latm = lat,
                   data = G0dm.dt,
                   type = 'prom',
                   source = source)
}

#### file to Meteo (daily) ####
readBDd <- function(file, lat,
                    format = "%d/%m/%Y",header = TRUE,
                    fill = TRUE, dec = '.', sep = ';',
                    dates.col = 'Dates', ta.col = 'Ta',
                    g0.col = 'G0', keep.cols = FALSE)
{
    #stops if the arguments are not characters or numerics
    stopifnot(is.character(dates.col) || is.numeric(dates.col))
    stopifnot(is.character(ta.col) || is.numeric(ta.col))
    stopifnot(is.character(g0.col) || is.numeric(g0.col))

    #read from file and set it in a data.table
    bd <- fread(file, header = header, fill = fill, dec = dec, sep = sep)

    #check the columns
    if(!(dates.col %in% names(bd))) stop(paste('The column', dates.col, 'is not
    in the file'))
    if(!(g0.col %in% names(bd))) stop(paste('The column', g0.col, 'is not in the
     file'))
    if(!(ta.col %in% names(bd))) stop(paste('The column', ta.col, 'is not in the
     file'))
```

```

    #name the dates column by Dates
    Dates <- bd[[dates.col]]
    bd[,(dates.col) := NULL]
    bd[, Dates := as.IDate(Dates, format = format)]

    #name the g0 column by G0
    G0 <- bd[[g0.col]]
    bd[, (g0.col) := NULL]
    bd[, G0 := as.numeric(G0)]

    #name the ta column by Ta
    Ta <- bd[[ta.col]]
    bd[, (ta.col) := NULL]
    bd[, Ta := as.numeric(Ta)]

    names0 <- NULL
    if(all(c('D0', 'B0') %in% names(bd))){
        names0 <- c(names0, 'D0', 'B0')
    }

    names0 <- c(names0, 'Ta')

    if(all(c('TempMin', 'TempMax') %in% names(bd))){
        names0 <- c(names0, 'TempMin', 'TempMax')
    }
    if(keep.cols)
    {
        #keep the rest of the columns but reorder the columns
        setcolorder(bd, c('Dates', 'G0', names0))
    }
    else
    {
        #erase the rest of the columns
        cols <- c('Dates', 'G0', names0)
        bd <- bd[, ..cols]
    }

    setkey(bd, 'Dates')
    result <- new(Class = 'Meteo',
                  latm = lat,
                  data = bd,
                  type = 'bd',
                  source = file)
}

#### file to Meteo (intradaily) ####
readBDi <- function(file, lat,
                    format = "%d/%m/%Y %H:%M:%S",
                    header = TRUE, fill = TRUE, dec = '.',
                    sep = ';', dates.col = 'dates', times.col,
                    ta.col = 'Ta', g0.col = 'G0', keep.cols = FALSE)
{
    #stops if the arguments are not characters or numerics
    stopifnot(is.character(dates.col) || is.numeric(dates.col))
    stopifnot(is.character(ta.col) || is.numeric(ta.col))
```

```r
    stopifnot(is.character(g0.col) || is.numeric(g0.col))

    #read from file and set it in a data.table
    bd <- fread(file, header = header, fill = fill, dec = dec, sep = sep)

    #check the columns
    if(!(dates.col %in% names(bd))) stop(paste('The column', dates.col, 'is not in the file'))
    if(!(g0.col %in% names(bd))) stop(paste('The column', g0.col, 'is not in the file'))
    if(!(ta.col %in% names(bd))) stop(paste('The column', ta.col, 'is not in the file'))

    if(!missing(times.col)){
        stopifnot(is.character(times.col) || is.numeric(times.col))
        if(!(times.col %in% names(bd))) stop(paste('The column', times.col, 'is not in the file'))

        #name the dates column by Dates
        format <- strsplit(format, ' ')
        dd <- as.IDate(bd[[dates.col]], format = format[[1]][1])
        tt <- as.ITime(bd[[times.col]], format = format[[1]][2])
        bd[,(dates.col) := NULL]
        bd[,(times.col) := NULL]
        bd[, Dates := as.POSIXct(dd, tt, tz = 'UTC')]
    }

    else
    {
        dd <- as.POSIXct(bd[[dates.col]], format = format, tz = 'UTC')
        bd[, (dates.col) := NULL]
        bd[, Dates := dd]
    }

    #name the g0 column by G0
    G0 <- bd[[g0.col]]
    bd[, (g0.col) := NULL]
    bd[, G0 := as.numeric(G0)]

    #name the ta column by Ta
    Ta <- bd[[ta.col]]
    bd[, (ta.col) := NULL]
    bd[, Ta := as.numeric(Ta)]

    names0 <- NULL
    if(all(c('D0', 'B0') %in% names(bd))){
        names0 <- c(names0, 'D0', 'B0')
    }

    names0 <- c(names0, 'Ta')

    if(keep.cols)
    {
        #keep the rest of the columns but reorder the columns
        setcolorder(bd, c('Dates', 'G0', names0))
    }
```

```
    else
    {
        #erase the rest of the columns
        cols <- c('Dates', 'G0', names0)
        bd <- bd[, ..cols]
    }

    setkey(bd, 'Dates')
    result <- new(Class = 'Meteo',
                  latm = lat,
                  data = bd,
                  type = 'bdI',
                  source = file)
}


dt2Meteo <- function(file, lat, source = '', type){
    ## Make sure its a data.table
    bd <- data.table(file)

    ## Dates is an as.POSIX element
    bd[, Dates := as.POSIXct(Dates, tz = 'UTC')]

    ## type
    if(missing(type)){
        sample <- median(diff(file$Dates))
        IsDaily <- as.numeric(sample, units = 'days')
        if(is.na(IsDaily)) IsDaily <- ifelse('G0d' %in% names(bd),
                                             1, 0)
        if(IsDaily >= 30) type <- 'prom'
        else{
            type <- ifelse(IsDaily >= 1, 'bd', 'bdI')
        }

    }
    if(!('Ta' %in% names(bd))){
        if(all(c('Tempmin', 'TempMax') %in% names(bd)))
            bd[, Ta := mean(c(Tempmin, TempMax))]
        else bd[, Ta := 25]
            }

    ## Columns of the data.table
    nms0 <- switch(type,
                   bd = ,
                   prom = {
                       nms0 <- 'G0d'
                       if(all(c('D0d', 'B0d') %in% names(bd))){
                           nms0 <- c(nms0, 'D0d', 'B0d')
                       }
                       nms0 <- c(nms0, 'Ta')
                       if(all(c('TempMin', 'TempMax') %in% names(bd))){
                           nms0 <- c(nms0, 'TempMin', 'TempMax')
                       }
                       nms0
                   },
                   bdI = {
```

```
                        nms0 <- 'G0'
                        if(all(c('D0', 'B0') %in% names(bd))){
                            nms0 <- c(nms0, 'D0', 'B0')
                        }
                        if('Ta' %in% names(bd)){
                            nms0 <- c(nms0, 'Ta')
                        }
                        nms0
                    })
    ## Columns order and set key
    setcolorder(bd, c('Dates', nms0))
    setkey(bd, 'Dates')
    ## Result
    result <- new(Class = 'Meteo',
                  latm = lat,
                  data = bd,
                  type = type,
                  source = source)
}

#### Liu and Jordan, Collares-Pereira and Rabl proposals ####
collper <- function(sol, compD)
{
    Dates <- indexI(sol)
    x <- as.Date(Dates)
    ind.rep <- cumsum(c(1, diff(x) != 0))
    solI <- as.data.tableI(sol, complete = T)
    ws <- solI$ws
    w <- solI$w

    a <- 0.409-0.5016*sin(ws+pi/3)
    b <- 0.6609+0.4767*sin(ws+pi/3)

    rd <- solI[, Bo0/Bo0d]
    rg <- rd * (a + b * cos(w))

    # Daily irradiation components
    G0d <- compD$G0d[ind.rep]
    B0d <- compD$B0d[ind.rep]
    D0d <- compD$D0d[ind.rep]

    # Daily profile
    G0 <- G0d * rg
    D0 <- D0d * rd

    # This method may produce diffuse irradiance higher than
    # global irradiance
    G0 <- pmax(G0, D0, na.rm = TRUE)
    B0 <- G0 - D0

    # Negative values are set to NA
    neg <- (B0 < 0) | (D0 < 0) | (G0 < 0)
    is.na(G0) <- neg
    is.na(B0) <- neg
    is.na(D0) <- neg

```

```
    # Daily profiles are scaled to keep daily irradiation values
    day <- truncDay(indexI(sol))
    sample <- sol@sample

    G0dCP <- ave(G0, day, FUN=function(x) P2E(x, sample))
    B0dCP <- ave(B0, day, FUN=function(x) P2E(x, sample))
    D0dCP <- ave(D0, day, FUN=function(x) P2E(x, sample))

    G0 <- G0 * G0d/G0dCP
    B0 <- B0 * B0d/B0dCP
    D0 <- D0 * D0d/D0dCP

    res <- data.table(G0, B0, D0)
    return(res)
}


#### intradaily Meteo to daily Meteo ####
Meteoi2Meteod <- function(G0i)
{
    lat <- G0i@latm
    source <- G0i@source

    dt0 <- getData(G0i)
    dt <- dt0[, lapply(.SD, sum),
              .SDcols = names(dt0)[!names(dt0) %in% c('Dates', 'Ta')],
              by = .(Dates = as.IDate(Dates))]
    if('Ta' %in% names(dt0)){
        Ta <- dt0[, .(Ta = mean(Ta),
                      TempMin = min(Ta),
                      TempMax = max(Ta)),
                  by = .(Dates = as.IDate(Dates))]
        if(all(Ta$Ta == c(Ta$TempMin, Ta$TempMax))) Ta[, c('TempMin', 'TempMax')
    := NULL]
        dt <- merge(dt, Ta)
    }
    if('G0' %in% names(dt)){
        names(dt)[names(dt) == 'G0'] <- 'G0d'
    }
    if('D0' %in% names(dt)){
        names(dt)[names(dt) == 'D0'] <- 'D0d'
    }
    if('B0' %in% names(dt)){
        names(dt)[names(dt) == 'B0'] <- 'B0d'
    }
    G0d <- dt2Meteo(dt, lat, source, type = 'bd')
    return(G0d)
}

#### daily Meteo to monthly Meteo ####
Meteod2Meteom <- function(G0d)
{
    lat <- G0d@latm
    source <- G0d@source

    dt <- getData(G0d)
```

```
    nms <- names(dt)[-1]
    dt <- dt[, lapply(.SD, mean),
             .SDcols = nms,
             by = .(month(Dates), year(Dates))]
    dt[, Dates := fBTd()]
    dt <- dt[, c('month', 'year') := NULL]

    setcolorder(dt, 'Dates')

    G0m <- dt2Meteo(dt, lat, source, type = 'prom')
    return(G0m)
}

zoo2Meteo <- function(file, lat, source = '')
{
    sample <- median(diff(index(file)))
    IsDaily <- as.numeric(sample, units = 'days')>=1
    type <- ifelse(IsDaily, 'bd', 'bdI')
    result <- new(Class = 'Meteo',
                  latm = lat,
                  data = file,
                  type = type,
                  source = source)
}

siarGET <- function(id, inicio, final, tipo = 'Mensuales', ambito = 'Estacion'){
    if(!(tipo %in% c('Horarios', 'Diarios', 'Semanales', 'Mensuales'))){
        stop('argument \'tipo\' must be: Horarios, Diarios, Semanales or
    Mensuales')
    }
    if(!(ambito %in% c('CCAA', 'Provincia', 'Estacion'))){
        stop('argument \'ambito\' must be: CCAA, Provincia or Estacion')
    }

    mainURL <- "https://servicio.mapama.gob.es"

    path <- paste('/apisiar/API/v1/Datos', tipo, ambito, sep = '/')

    ## prepare the APIsiar
    req <- request(mainURL) |>
        req_url_path(path) |>
        req_url_query(Id = id,
                      FechaInicial = inicio,
                      FechaFinal = final,
                      ClaveAPI = '_Q8L_niYFBBmBs-
    vB3UomUqdUYy98FTRX1aYbrZ8n2FXuHYGTV')
    ## execute it
    resp <- req_perform(req)

    ##JSON to R
    respJSON <- resp_body_json(resp, simplifyVector = TRUE)

    if(!is.null(respJSON$MensajeRespuesta)){
        stop(respJSON$MensajeRespuesta)
    }

```

```
    res0 <- data.table(respJSON$Datos)

    res <- switch(tipo,
                  Horarios = {
                      res0[, HoraMin := as.ITime(sprintf('%04d', HoraMin),
                                                 format = '%H%M')]
                      res0[, Fecha := as.IDate(Fecha, format = '%Y-%m-%d')]
                      res0[, Fecha := as.IDate(ifelse(HoraMin == as.ITime(0),
                                                      Fecha+1, Fecha))]
                      res0[, Dates := as.POSIXct(HoraMin, Fecha,
                                                 tz = 'Europe/Madrid')]
                      res0 <- res0[, .(Dates,
                                       G0 = Radiacion,
                                       Ta = TempMedia)]
                      return(res0)
                  },
                  Diarios = {
                      res0[, Dates := as.IDate(Fecha)]
                      res0 <- res0[, .(Dates,
                                       G0d = Radiacion * 277.78,
                                       Ta = TempMedia,
                                       TempMin,
                                       TempMax)]
                      return(res0)
                  },
                  Semanales = res0,
                  Mensuales = {
                      promDays<-c(17,14,15,15,15,10,18,18,18,19,18,13)
                      names(res0)[1] <- 'Year'
                      res0[, Dates := as.IDate(paste(Year, Mes,
                                                     promDays[Mes],
                                                     sep = '-'))]
                      res0 <- res0[, .(Dates,
                                       G0d = Radiacion * 277.78,
                                       Ta = TempMedia,
                                       TempMin,
                                       TempMax)]
                  })

    return(res)
}

haversine <- function(lat1, lon1, lat2, lon2) {
    R <- 6371  # Radius of the Earth in kilometers
    dLat <- (lat2 - lat1) * pi / 180
    dLon <- (lon2 - lon1) * pi / 180
    a <- sin(dLat / 2) * sin(dLat / 2) + cos(lat1 * pi / 180) *
        cos(lat2 * pi / 180) * sin(dLon / 2) * sin(dLon / 2)
    c <- 2 * atan2(sqrt(a), sqrt(1 - a))
    d <- R * c
    return(d)
}

readSIAR <- function(Lon = 0, Lat = 0,
                     inicio = paste(year(Sys.Date())-1, '01-01', sep = '-'),
                     final = paste(year(Sys.Date())-1, '12-31', sep = '-'),
```

```
                    tipo = 'Mensuales', n_est = 3){
    inicio <- as.Date(inicio)
    final <- as.Date(final)

    n_reg <- switch(tipo,
                    Horarios = {
                        tt <- difftime(final, inicio, units = 'days')
                        tt <- (as.numeric(tt)+1)*48
                        tt <- tt*n_est
                        tt
                    },
                    Diarios = {
                        tt <- difftime(final, inicio, units = 'days')
                        tt <- as.numeric(tt)+1
                        tt <- tt*n_est
                        tt
                    },
                    Semanales = {
                        tt <- difftime(final, inicio, units = 'weeks')
                        tt <- as.numeric(tt)
                        tt <- tt*n_est
                        tt
                    },
                    Mensuales = {
                        tt <- difftime(final, inicio, units = 'weeks')
                        tt <- as.numeric(tt)/4.34524
                        tt <- ceiling(tt)
                        tt <- tt*n_est
                        tt
                    })
    if(n_reg > 100) stop(paste('Number of requested records (', n_reg,
                             ') exceeds the maximum allowed (100)', sep = ''))
    ## Obtain the nearest stations
    siar <- est_SIAR[
        Fecha_Instalacion <= final & (is.na(Fecha_Baja) | Fecha_Baja >= inicio)
    ]

    ## Weigths for the interpolation
    siar[, dist := haversine(Latitud, Longitud, Lat, Lon)]
    siar <- siar[order(dist)][1:n_est]
    siar[, peso := 1/dist]
    siar[, peso := peso/sum(peso)]
    ## Is the given location within the polygon formed by the stations?
    siar <- siar[, .(Estacion, Codigo, dist, peso)]

    ## List for the data.tables of siarGET
    siar_list <- list()
    for(codigo in siar$Codigo){
        siar_list[[codigo]] <- siarGET(id = codigo,
                                       inicio = as.character(inicio),
                                       final = as.character(final),
                                       tipo = tipo)
        siar_list[[codigo]]$peso <- siar[Codigo == codigo, peso]
    }

    ## Bind the data.tables
```

```
    s_comb <- rbindlist(siar_list, use.names = TRUE, fill = TRUE)

    nms <- names(s_comb)
    nms <- nms[-c(1, length(nms))]

    ## Interpole
    res <- s_comb[, lapply(.SD * peso, sum, na.rm = TRUE),
                  .SDcols = nms,
                  by = Dates]

    ## Source
    mainURL <- "https://servicio.mapama.gob.es"
    Estaciones <- siar[, paste(Estacion, '(', Codigo, ')', sep = '')]
    Estaciones <- paste(Estaciones, collapse = ', ')
    source <- paste(mainURL, '\n  -Estaciones:', Estaciones, sep = ' ')

    res <- switch(tipo,
                  Horarios = {dt2Meteo(res, lat = Lat, source = mainURL, type = 'bdI')},
                  Diarios = {dt2Meteo(res, lat = Lat, source = mainURL, type = 'bd')},
                  Semanales = {res},
                  Mensuales = {dt2Meteo(res, lat = Lat, source = source, type = 'prom')})
    return(res)
}
```

A.2. Clases

Sol

```
setClass(
         Class='Sol', ##Solar angles
         slots = c(
             lat='numeric',#latitud in degrees, >0 if North
             solD='data.table',#daily angles
             solI='data.table',#intradaily angles
             sample='character',#sample of time
             method='character'#method used for geometry calculations
         ),
    validity=function(object) {return(TRUE)}
)
```

Meteo

```
setClass(
    Class = 'Meteo', ##radiation and temperature data
    slots = c(
        latm='numeric',#latitud in degrees, >0 if North
        data='data.table',#data, incluying G (Wh/m2) and Ta (ºC)
        type='character',#choose between 'prom', 'bd' and 'bdI'
        source='character'#origin of the data
    ),
    validity=function(object) {return(TRUE)}
```

```
)
```

G0

```
setClass(
    Class = 'G0',
    slots = c(
        G0D = 'data.table',  #result of fCompD
        G0dm = 'data.table', #monthly means
        G0y = 'data.table',  #yearly values
        G0I = 'data.table',  #result of fCompI
        Ta = 'data.table'    #Ambient temperature
    ),
    contains = c('Sol', 'Meteo'),
    validity = function(object) {return(TRUE)}
)

```

Gef

```
setClass(
         Class='Gef',
         slots = c(
           GefD='data.table',  #daily values
           Gefdm='data.table', #monthly means
           Gefy='data.table',  #yearly values
           GefI='data.table',  #result of fInclin
           Theta='data.table', #result of fTheta
           iS='numeric',       #dirt index
           alb='numeric',      #albedo
           modeTrk='character',   #tracking mode
           modeShd='character',   #shadow mode
           angGen='list',         #includes alpha, beta and betaLim
           struct='list',         #structure dimensions
           distances='data.frame' #distances between structures
           ),
         contains='G0',
         validity=function(object) {return(TRUE)}
)
```

ProdGCPV

```
setClass(
         Class='ProdGCPV',
         slots = c(
           prodD='data.table',  #daily values
           prodDm='data.table', #monthly means
           prody='data.table',  #yearly values
           prodI='data.table',  #results of fProd
           module='list',       #module characteristics
           generator='list',    #generator characteristics
           inverter='list',     #inverter characteristics
           effSys='list'        #efficiency values of the system
```

```
          ),
        contains='Gef',
        validity=function(object) {return(TRUE)}
)
```

ProdPVPS

```
setClass(
        Class='ProdPVPS',
        slots = c(
          prodD='data.table',  #daily values
          prodDm='data.table', #monthly means
          prody='data.table',  #yearly values
          prodI='data.table',  #results of fPump
          Pg='numeric',        #generator power
          H='numeric',         #manometric head
          pump='list',         #parameters of the pump
          converter='list',    #inverter characteristics
          effSys='list'        #efficiency values of the system
          ),
        contains='Gef',
        validity=function(object) {return(TRUE)}
)
```

Shade

```
setClass(
        Class='Shade',
        slots = c(
          FS='numeric',  #shadows factor values
          GRR='numeric', #Ground Requirement Ratio
          Yf='numeric',  #final productivity
          FS.loess='loess', #local fitting of FS with loess
          Yf.loess='loess', #local fitting of Yf with loess
          modeShd='character', #mode of shadow
          struct='list',       #dimensions of the structures
          distances='data.frame', #distances between structures
          res='numeric'           #difference between the different steps of the calculations
          ),
        contains='ProdGCPV',##Resultado de prodGCPV sin sombras (Prod0)
        validity=function(object) {return(TRUE)}
)
```

A.3. Funciones

corrFdKt

```
#### monthly Kt ####
Ktm <- function(sol, G0dm){
    solf <- sol@solD[, .(Dates, Bo0d)]
    solf[, c('month', 'year') := .(month(Dates), year(Dates))]
```

```
    solf[,Bo0m := mean(Bo0d), by = .(month, year)]
    G0df <- G0dm@data[, .(Dates, G0d)]
    G0df[, c('month', 'year') := .(month(Dates), year(Dates))]
    G0df[, G0d := mean(G0d), by = .(month, year)]
    Ktm <- G0df$G0d/solf$Bo0m
    return(Ktm)
}

#### daily Kt ####
Ktd <- function(sol, G0d){
    Bo0d <- sol@solD$Bo0d
    G0d <- getG0(G0d)
    Ktd <- G0d/Bo0d
    return(Ktd)
}

### intradaily
Kti <- function(sol, G0i){
    Bo0 <- sol@solI$Bo0
    G0i <- getG0(G0i)
    Kti <- G0i/Bo0
    return(Kti)
}


#### monthly correlations ####

### Page ###
FdKtPage <- function(sol, G0dm){
    Kt <- Ktm(sol, G0dm)
    Fd=1-1.13*Kt
    return(data.table(Fd, Kt))
}

### Liu and Jordan ###
FdKtLJ <- function(sol, G0dm){
    Kt <- Ktm(sol, G0dm)
    Fd=(Kt<0.3)*0.595774 +
        (Kt>=0.3 & Kt<=0.7)*(1.39-4.027*Kt+5.531*Kt^2-3.108*Kt^3)+
        (Kt>0.7)*0.215246
    return(data.table(Fd, Kt))
}


#### daily correlations ####

### Collares-Pereira and Rabl
FdKtCPR <- function(sol, G0d){
    Kt <- Ktd(sol, G0d)
    Fd=(0.99*(Kt<=0.17))+(Kt>0.17 & Kt<0.8)*
        (1.188-2.272*Kt+9.473*Kt^2-21.856*Kt^3+14.648*Kt^4)+
        (Kt>=0.8)*0.2426688
    return(data.table(Fd, Kt))
}

### Erbs, Klein and Duffie ###
```

```
FdKtEKDd <- function(sol, G0d){
    ws <- sol@solD$ws
    Kt <- Ktd(sol, G0d)

    WS1=(abs(ws)<1.4208)
    Fd=WS1*((Kt<0.715)*(1-0.2727*Kt+2.4495*Kt^2-11.9514*Kt^3+9.3879*Kt^4)+
            (Kt>=0.715)*(0.143))+
        !WS1*((Kt<0.722)*(1+0.2832*Kt-2.5557*Kt^2+0.8448*Kt^3)+
              (Kt>=0.722)*(0.175))
  return(data.table(Fd, Kt))
}

### CLIMED1 ###
FdKtCLIMEDd <- function(sol, G0d){
    Kt <- Ktd(sol, G0d)
    Fd=(Kt<=0.13)*(0.952)+
    (Kt>0.13 & Kt<=0.8)*(0.868+1.335*Kt-5.782*Kt^2+3.721*Kt^3)+
      (Kt>0.8)*0.141
  return(data.table(Fd, Kt))
}

#### intradaily correlations ####

### intradaily EKD ###
FdKtEKDh <- function(sol, G0i){
    Kt <- Kti(sol, G0i)
    Fd=(Kt<=0.22)*(1-0.09*Kt)+
    (Kt>0.22 & Kt<=0.8)*(0.9511-0.1604*Kt+4.388*Kt^2-16.638*Kt^3+12.336*Kt^4)+
      (Kt>0.8)*0.165
  return(data.table(Fd, Kt))
}

### intradaily CLIMED
FdKtCLIMEDh <- function(sol, G0i){
    Kt <- Kti(sol, G0i)
    Fd=(Kt<=0.21)*(0.995-0.081*Kt)+
        (Kt>0.21 & Kt<=0.76)*(0.724+2.738*Kt-8.32*Kt^2+4.967*Kt^3)+
        (Kt>0.76)*0.180
    return(data.table(Fd, Kt))
}

### intradaily Boland, Ridley and Lauret ###
FdKtBRL <- function(sol, G0i){
    Kt <- Kti(sol, G0i)
    sample <- sol@sample

    solI <- as.data.tableI(sol, complete = TRUE)
    w <- solI$w
    night <- solI$night
    AlS <- solI$AlS

    G0d <- Meteoi2Meteod(G0i)
    ktd <- Ktd(sol, G0d)

    ##persistence
    pers <- persistence(sol, ktd)
```

```

    ##indexRep for ktd and pers
    Dates <- indexI(sol)
    x <- as.Date(Dates)
    ind.rep <- cumsum(c(1, diff(x) != 0))
    ktd <- ktd[ind.rep]
    pers <- pers[ind.rep]

    ##fd calculation
    Fd=(1+exp(-5.38+6.63*Kt+0.006*r2h(w)-0.007*r2d(AlS)+1.75*ktd+1.31*pers))^(-1)

    return(data.table(Fd, Kt))
}

persistence <- function(sol, Ktd){
    kt <- data.table(indexD(sol), Ktd)
    ktNA <- na.omit(kt)
    iDay <- truncDay(ktNA[[1]])

    x <- rle(as.numeric(iDay))$lengths
    xLast <- cumsum(x)

    lag1 <- shift(ktNA$Ktd, -1, fill = NA)
    for (i in xLast){
        if ((i-1) != 0){lag1[i] <- ktNA$Ktd[i-1]}
    }

    lag2 <- shift(ktNA$Ktd, 1, fill = NA)
    for (i in xLast){
        if ((i+1) <= length(ktNA$Ktd)){lag2[i] <- ktNA$Ktd[i+1]}
    }
    pers <- data.table(lag1, lag2)
    pers[, mean := 1/2 * (lag1+lag2)]
    pers[, mean]
}
```

fBTd

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

fBTi

```
intervalo <- function(day, sample){
    intervalo <- seq.POSIXt(from = as.POSIXct(paste(day, '00:00:00'), tz = 'UTC')
    ,
                    to = as.POSIXct(paste(day, '23:59:59'), tz = 'UTC'),
                    by = sample)
    return(intervalo)
}

fBTi <- function(d, sample = 'hour'){
    BTi <- lapply(d, intervalo, sample)
    BTi <- do.call(c, BTi)
    return(BTi)
}
```

fCompD

```
fCompD <- function(sol, G0d, corr = 'CPR', f)
{
    if(!(corr %in% c('CPR', 'Page', 'LJ', 'EKDd', 'CLIMEDd', 'user', 'none'))){
        warning('Wrong descriptor of correlation Fd-Ktd. Set CPR.')
        corr <- 'CPR'
    }
    if(class(sol)[1] != 'Sol'){
        sol <- sol[, calcSol(lat = unique(lat), BTi = Dates)]
    }
    if(class(G0d)[1] != 'Meteo'){
        dt <- copy(data.table(G0d))
        if(!('Dates' %in% names(dt))){
            dt[, Dates := indexD(sol)]
            setcolorder(dt, 'Dates')
            setkey(dt, 'Dates')
        }
        if('lat' %in% names(dt)){
            latg <- unique(dt$lat)
            dt[, lat := NULL]
        }else{latg <- getLat(sol)}
        G0d <- dt2Meteo(dt, latg)
    }

    stopifnot(indexD(sol) == indexD(G0d))
    Bo0d <- sol@solD$Bo0d
    G0 <- getData(G0d)$G0

    is.na(G0) <- (G0>Bo0d)

    ### the Direct and Difuse data is not given
    if(corr != 'none'){
        Fd <- switch(corr,
                    CPR = FdKtCPR(sol, G0d),
                    Page = FdKtPage(sol, G0d),
                    LJ = FdKtLJ(sol, G0d),
                    CLIMEDd = FdKtCLIMEDd(sol, G0d),
                    user = f(sol, G0d))
```

```
        Kt <- Fd$Kt
        Fd <- Fd$Fd
        D0d <- Fd * G0
        B0d <- G0 - D0d
    }
    ### the Direct and Difuse data is given
    else {
        G0 <- getData(G0d)$G0
        D0d <- getData(G0d)[['D0']]
        B0d <- getData(G0d)[['B0']]
        Fd <- D0d/G0
        Kt <- G0/Bo0d
    }

    result <- data.table(Dates = indexD(sol), Fd, Kt, G0d = G0, D0d, B0d)
    setkey(result, 'Dates')
    result
}
```

fCompI

```
fCompI <- function(sol, compD, G0I,
                   corr = 'EKDh', f,
                   filterG0 = TRUE){
    if(!(corr %in% c('EKDh', 'CLIMEDh', 'BRL', 'user', 'none'))){
        warning('Wrong descriptor of correlation Fd-Ktd. Set EKDh.')
        corr <- 'EKDh'
    }

    if(class(sol)[1] != 'Sol'){
        sol <- sol[, calcSol(lat = unique(lat), BTi = Dates)]
    }

    lat <- sol@lat
    sample <- sol@sample
    night <- sol@solI$night
    Bo0 <- sol@solI$Bo0
    Dates <- indexI(sol)

    ## If instantaneous values are not provided, compD is used instead.
    if (missing(G0I)) {

        G0I <- collper(sol, compD)
        G0 <- G0I$G0
        B0 <- G0I$B0
        D0 <- G0I$D0

        Fd <- D0/G0
        Kt <- G0/Bo0

    } else { ## Use instantaneous values if provided through G0I

        if(class(G0I)[1] != 'Meteo'){
            dt <- copy(G0I)
            if(!('Dates' %in% names(G0I))){
```

```
                dt[, Dates := indexI(sol)]
                setcolorder(dt, 'Dates')
                setkey(dt, 'Dates')
            }
            if('lat' %in% names(G0I)){latg <- unique(G0I$lat)}
            else{latg <- lat}
            G0I <- dt2Meteo(dt, latg)
        }

        if (corr!='none'){
            G0 <- getG0(G0I)
            ## Filter values: surface irradiation must be lower than
            ## extraterrestial;
            if (filterG0) {is.na(G0) <- (G0 > Bo0)}

            ## Fd-Kt correlation
            Fd <- switch(corr,
                         EKDh = FdKtEKDh(sol, G0I),
                         CLIMEDh = FdKtCLIMEDh(sol, G0I),
                         BRL = FdKtBRL(sol, G0I),
                         user = f(sol, G0I))

            Kt <- Fd$Kt
            Fd <- Fd$Fd
            D0 <- Fd * G0
            B0 <- G0 - D0

        } else {
            G0 <- getG0(G0I)
            D0 <- getData(G0I)[['D0']]
            B0 <- getData(G0I)[['B0']]
            ## Filter values: surface irradiation must be lower than
            ## extraterrestial;
            if (isTRUE(filterG0)) is.na(G0) <- is.na(D0) <- is.na(B0) <- (G0 > Bo0)

            Fd <- D0/G0
            Kt <- G0/Bo0
        }
    }
    ## Values outside sunrise-sunset are set to zero
    G0[night] <- D0[night] <- B0[night] <- Kt[night] <- Fd[night] <- 0

    result <- data.table(Dates, Fd, Kt, G0, D0, B0)
    setkey(result, 'Dates')
    result
}
```

fInclin

```
fInclin <- function(compI, angGen, iS = 2, alb = 0.2, horizBright = TRUE, HCPV = FALSE){
    ##compI es class='G0'

    ##Arguments
```

```
    stopifnot(iS %in% 1:4)
    Beta <- angGen$Beta
    Alfa <- angGen$Alfa
    cosTheta <- angGen$cosTheta

    comp <- as.data.tableI(compI, complete=TRUE)
    night <- comp$night
    B0 <- comp$B0
    Bo0 <- comp$Bo0
    D0 <- comp$D0
    G0 <- comp$G0
    cosThzS <- comp$cosThzS
    is.na(cosThzS) <- night

    ##N.Martin method for dirt and non-perpendicular incidence
    Suc <- rbind(c(1, 0.17, -0.069),
                 c(0.98,.2,-0.054),
                 c(0.97,0.21,-0.049),
                 c(0.92,0.27,-0.023))
    FTb <- (exp(-cosTheta/Suc[iS,2]) - exp(-1/Suc[iS,2]))/(1 - exp(-1/Suc[iS,2]))
    FTd <- exp(-1/Suc[iS,2] * (4/(3*pi) * (sin(Beta) + (pi - Beta - sin(Beta))/(1 + cos(Beta))) +
                           Suc[iS,3] * (sin(Beta) + (pi - Beta - sin(Beta))/(1 + cos(Beta)))^2))
    FTr <- exp(-1/Suc[iS,2] * (4/(3*pi) * (sin(Beta) + (Beta - sin(Beta))/(1 - cos(Beta))) +
                           Suc[iS,3] * (sin(Beta) + (Beta - sin(Beta))/(1 - cos(Beta)))^2))

    ##Hay and Davies method for diffuse treatment
    B <- B0 * cosTheta/cosThzS * (cosThzS>0.007) #The factor cosThzS>0.007 is needed to eliminate erroneous results near dawn
    k1 <- B0/(Bo0)
    Di <- D0 * (1-k1) * (1+cos(Beta))/2
    if (horizBright) Di <- Di * (1+sqrt(B0/G0) * sin(Beta/2)^3)
    Dc <- D0 * k1 * cosTheta/cosThzS * (cosThzS>0.007)
    R <- alb * G0 * (1-cos(Beta))/2
    D <- (Di + Dc)
    ##Extraterrestrial irradiance on the inclined plane
    Bo <- Bo0 * cosTheta/cosThzS * (cosThzS>0.007)
    ##Normal direct irradiance (DNI)
    Bn <- B0/cosThzS
    ##Sum of components
    G <- B + D + R
    Ref <- R * Suc[iS,1] * (1-FTr) * (!HCPV)
    Ref[is.nan(FTr)] <- 0 #When cos(Beta)=1, FTr=NaN. Cancel Ref.
    Dief <- Di * Suc[iS,1] * (1 - FTd) * (!HCPV)
    Dcef <- Dc * Suc[iS,1] * (1 - FTb) * (!HCPV)
    Def <- Dief + Dcef
    Bef <- B * Suc[iS,1] * (1 - FTb)
    Gef <- Bef + Def + Ref

    result <- data.table(Bo, Bn,
                         G, D, Di, Dc, B, R,
                         FTb, FTd, FTr,
                         Dief, Dcef, Gef, Def, Bef, Ref)
```

```

    ## Use 0 instead of NA for irradiance values
    result[night] <- 0
    result[, Dates := indexI(compI)]
    result[, .SD, by = Dates]
    setcolorder(result, c('Dates', names(result)[-length(result)]))
    result
}
```

fProd

```
## voc, isc, vmpp, impp : *cell* values
## Voc, Isc, Vmpp, Impp: *module/generator* values

## Compute Current - Voltage characteristic of a solar *cell* with Gef
## and Ta
iv <- function(vocn, iscn, vmn, imn,
               TONC, CoefVT = 2.3e-3,
               Ta, Gef,
               vmin = NULL, vmax = NULL)
{
    ##Cell Constants
    Gstc <- 1000
    Ct <- (TONC - 20) / 800
    Vtn <- 0.025 * (273 + 25) / 300
    m <- 1.3

    ##Cell temperature
    Tc <- Ta + Ct * Gef
    Vt <- 0.025 * (Tc + 273)/300

    ## Series resistance
    Rs <- (vocn - vmn + m * Vtn * log(1 - imn/iscn)) / imn

    ## Voc and Isc at ambient conditions
    voc <- vocn - CoefVT * (Tc - 25)
    isc <- iscn * Gef/Gstc

    ## Ruiz method for computing voltage and current characteristic of a *cell*
    rs <- Rs * isc/voc
    koc <- voc/(m * Vt)

    ## Maximum Power Point
    Dm0 <- (koc - 1)/(koc - log(koc))
    Dm <- Dm0 + 2 * rs * Dm0^2

    impp <- isc * (1 - Dm/koc)
    vmpp <- voc * (1 - log(koc/Dm)/koc - rs * (1 - Dm/koc))

    vdc <- vmpp
    idc <- impp

    ## When the MPP is below/above the inverter voltage limits, it
    ## sets the voltage point at the corresponding limit.

```

```

## Auxiliary functions for computing the current at a defined
## voltage.
ilimit <- function(v, koc, rs)
{
    if (is.na(koc))
        result <- NA
    else
    {
        ## The IV characteristic is an implicit equation. The starting
        ## point is the voltage of the cell (imposed by the inverter
        ## limit).

        izero <- function(i , v, koc, rs)
        {
            vp <- v + i * rs
            Is <- 1/(1 - exp(-koc * (1 - rs)))
            result <- i - (1 - Is * (exp(-koc * (1 - vp)) - exp(-koc * (1 -
rs))))
        }

        result <- uniroot(f = izero,
                          interval = c(0,1),
                          v = v,
                          koc = koc,
                          rs = rs)$root
    }
    result
}
## Inverter minimum voltage
if (!is.null(vmin))
{
    if (any(vmpp < vmin, na.rm = TRUE))
    {
        indMIN <- which(vmpp < vmin)
        imin <- sapply(indMIN, function(i)
        {
            vocMIN <- voc[i]
            kocMIN <- koc[i]
            rsMIN <- rs[i]
            vmin <- vmin/vocMIN
            ##v debe estar entre 0 y 1
            vmin[vmin < 0] <- 0
            vmin[vmin > 1] <- 1
            ilimit(vmin, kocMIN, rsMIN)
        })
        iscMIN <- isc[indMIN]
        idc[indMIN] <- imin * iscMIN
        vdc[indMIN] <- vmin
        warning('Minimum MPP voltage of the inverter has been reached')}
}

if (!is.null(vmax))
{
    if (any(vmpp > vmax, na.rm = TRUE))
    {
```

```
            indMAX <- which(vmpp > vmax)
            imax <- sapply(indMAX, function(i)
            {
                vocMAX <- voc[i]
                kocMAX <- koc[i]
                rsMAX <- rs[i]
                vmax <- vmax / vocMAX
                ##v debe estar entre 0 y 1
                vmax[vmax < 0] <- 0
                vmax[vmax > 1] <- 1
                ilimit(vmax, kocMAX, rsMAX)
            })
            iscMAX <- isc[indMAX]
            idc[indMAX] <- imax * iscMAX
            vdc[indMAX] <- vmax
            warning('Maximum MPP voltage of the inverter has been reached')
        }
    }
    data.table(Ta, Tc, Gef, voc, isc, vmpp, impp, vdc, idc)
}

fProd <- function(inclin,
                  module=list(),
                  generator=list(),
                  inverter=list(),
                  effSys=list()
                  )
{

    stopifnot(is.list(module),
              is.list(generator),
              is.list(inverter),
              is.list(effSys)
              )
    ## Extract data from objects
    if (class(inclin)[1]=='Gef') {
        indInclin <- indexI(inclin)
        gefI <- as.data.tableI(inclin, complete = TRUE)
        Gef <- gefI$Gef
        Ta <- gefI$Ta
    } else {
        Gef <- inclin$Gef
        Ta <- inclin$Ta
    }

    ## Module, generator, and inverter parameters
    module.default <- list(Vocn = 57.6,
                           Iscn = 4.7,
                           Vmn = 46.08,
                           Imn = 4.35,
                           Ncs = 96,
                           Ncp = 1,
                           CoefVT = 0.0023,
                           TONC = 47)
    module <- modifyList(module.default, module)
    ## Make these parameters visible because they will be used often.
```

```
    Ncs <- module$Ncs
    Ncp <- module$Ncp

    generator.default <- list(Nms = 12,
                              Nmp = 11)
    generator <- modifyList(generator.default, generator)
    generator$Pg <- (module$Vmn * generator$Nms) *
        (module$Imn * generator$Nmp)
    Nms <- generator$Nms
    Nmp <- generator$Nmp

    inverter.default <- list(Ki = c(0.01,0.025,0.05),
                             Pinv = 25000,
                             Vmin = 420,
                             Vmax = 750,
                             Gumb = 20)
    inverter <- modifyList(inverter.default, inverter)
    Pinv <- inverter$Pinv

    effSys.default <- list(ModQual = 3,
                           ModDisp = 2,
                           OhmDC = 1.5,
                           OhmAC = 1.5,
                           MPP = 1,
                           TrafoMT = 1,
                           Disp = 0.5)
    effSys <- modifyList(effSys.default, effSys)

    ## Solar Cell i-v
    vocn <- with(module, Vocn / Ncs)
    iscn <- with(module, Iscn/ Ncp)
    vmn <- with(module, Vmn / Ncs)
    imn <- with(module, Imn / Ncp)
    vmin <- with(inverter, Vmin / (Ncs * Nms))
    vmax <- with(inverter, Vmax / (Ncs * Nms))

    cell <- iv(vocn, iscn,
               vmn, imn,
               module$TONC, module$CoefVT,
               Ta, Gef,
               vmin, vmax)

    ## Generator voltage and current
    Idc <- Nmp * Ncp * cell$idc
    Isc <- Nmp * Ncp * cell$isc
    Impp <- Nmp * Ncp * cell$impp
    Vdc <- Nms * Ncs * cell$vdc
    Voc <- Nms * Ncs * cell$voc
    Vmpp <- Nms * Ncs * cell$vmpp

    ##DC power (normalization with nominal power of inverter)
    ##including losses
    PdcN <- with(effSys, (Idc * Vdc) / Pinv *
                         (1 - ModQual / 100) *
                         (1 - ModDisp / 100) *
                         (1 - MPP / 100) *
```

```
                        (1 - OhmDC / 100)
                )

    ##Normalized AC power to the inverter
    Ki <- inverter$Ki
    if (is.matrix(Ki)) { #Ki is a matrix of nine coefficients-->dependence with
    tension
        VP <- cbind(Vdc, PdcN)
        PacN <- apply(VP, 1, solvePac, Ki)
    } else { #Ki is a vector of three coefficients-->without dependence on
    voltage
        A <- Ki[3]
        B <- Ki[2] + 1
        C <- Ki[1] - (PdcN)
        PacN <- (-B + sqrt(B^2 - 4 * A * C))/(2 * A)
    }
    EffI <- PacN / PdcN
    pacNeg <- PacN <= 0
    PacN[pacNeg] <- PdcN[pacNeg] <- EffI[pacNeg] <- 0


    ##AC and DC power without normalization
    Pac <- with(effSys, PacN * Pinv *
                        (Gef > inverter$Gumb) *
                        (1 - OhmAC / 100) *
                        (1 - TrafoMT / 100) *
                        (1 - Disp / 100))
    Pdc <- PdcN * Pinv * (Pac > 0)


    ## Result
    resProd <- data.table(Tc = cell$Tc,
                         Voc, Isc,
                         Vmpp, Impp,
                         Vdc, Idc,
                         Pac, Pdc,
                         EffI)
    if (class(inclin)[1] %in% 'Gef'){
        result <- resProd[, .SD,
                          by=.(Dates = indInclin)]
        attr(result, 'generator') <- generator
        attr(result, 'module') <- module
        attr(result, 'inverter') <- inverter
        attr(result, 'effSys') <- effSys
        return(result)
    } else {
        result <- cbind(inclin, resProd)
        return(result)
    }
}
```

fPump

```
fPump <- function(pump, H){

```

```
    w1=3000 ##synchronous rpm frequency
    wm=2870 ##rpm frequency with slip when applying voltage at 50 Hz
    s=(w1-wm)/w1
    fen=50 ##Nominal electrical frequency
    fmin=sqrt(H/pump$a)
    fmax=with(pump, (-b*Qmax+sqrt(b^2*Qmax^2-4*a*(c*Qmax^2-H)))/(2*a))
    ##fb is rotation frequency (Hz) of the pump,
    ##fe is the electrical frequency applied to the motor
    ##which makes it rotate at a frequency fb (and therefore also the pump).
    fb=seq(fmin,min(60,fmax),length=1000) #The maximum frequency is 60
    fe=fb/(1-s)

###Flow
    Q=with(pump, (-b*fb-sqrt(b^2*fb^2-4*c*(a*fb^2-H)))/(2*c))
    Qmin=0.1*pump$Qn*fb/50
    Q=Q+(Qmin-Q)*(Q<Qmin)

###Hydraulic power
    Ph=2.725*Q*H

###Mechanical power
    Q50=50*Q/fb
    H50=H*(50/fb)^2
    etab=with(pump, j*Q50^2+k*Q50+l)
    Pb50=2.725*H50*Q50/etab
    Pb=Pb50*(fb/50)^3

###Electrical power
    Pbc=Pb*50/fe
    etam=with(pump, g*(Pbc/Pmn)^2+h*(Pbc/Pmn)+i)
    Pmc=Pbc/etam
    Pm=Pmc*fe/50
    Pac=Pm
    ##Pdc=Pm/(etac*(1-cab))

###I build functions for flow, frequency and powers
###to adjust the AC power.
    fQ<-splinefun(Pac,Q)
    fFreq<-splinefun(Pac,fe)
    fPb<-splinefun(Pac,Pb)
    fPh<-splinefun(Pac,Ph)
    lim=c(min(Pac),max(Pac))
    ##lim marks the operating range of the pump
    result<-list(lim = lim,
                 fQ = fQ,
                 fPb = fPb,
                 fPh = fPh,
                 fFreq = fFreq)
}
```

fSolD

```
fSolD <- function(lat, BTd, method = 'michalsky'){
    if (abs(lat) > 90){
        lat <- sign(lat) * 90
```

```
        warning(paste('Latitude outside acceptable values. Set to', lat))
    }
    sun <- data.table(Dates = unique(as.IDate(BTd)),
                      lat = lat)

    #### solarAngles ####

    ##Declination
    sun[, decl := declination(Dates, method = method)]
    ##Eccentricity
    sun[, eo := eccentricity(Dates, method = method)]
    ##Equation of time
    sun[, EoT := eot(Dates)]
    ##Solar time
    sun[, ws := sunrise(Dates, lat, method = method,
                        decl = decl)]
    ##Extraterrestrial irradiance
    sun[, Bo0d := bo0d(Dates, lat, method = method,
                       decl = decl,
                       eo = eo,
                       ws = ws
                       )]
    setkey(sun, Dates)
    return(sun)
}
```

fSolI

```
fSolI <- function(solD, sample = 'hour', BTi,
                  EoT = TRUE, keep.night = TRUE, method = 'michalsky')
{
    #Solar constant
    Bo <- 1367

    if(missing(BTi)){
        d <- solD$Dates
        BTi <- fBTi(d, sample)
    }
    sun <- data.table(Dates = as.IDate(BTi),
                      Times = as.ITime(BTi))
    sun <- merge(solD, sun, by = 'Dates')
    sun[, eqtime := EoT]
    sun[, EoT := NULL]

    #sun hour angle
    sun[, w := sunHour(Dates, BTi, EoT = EoT, method = method, eqtime = eqtime)]

    #classify night elements
    sun[, night := abs(w) >= abs(ws)]

    #zenith angle
    sun[, cosThzS := zenith(Dates, lat, BTi,
                            method = method,
                            decl = decl,
                            w = w
```

```
                            )]

    #solar altitude angle
    sun[, AlS := asin(cosThzS)]

    #azimuth
    sun[, AzS := azimuth(Dates, lat, BTi, sample,
                         method = method,
                         decl = decl,
                         w = w,
                         cosThzS = cosThzS)]

    #Extraterrestrial irradiance
    sun[, Bo0 := Bo * eo * cosThzS]

    #When it is night there is no irradiance
    sun[night == TRUE, Bo0 := 0]

    #Erase columns that are in solD
    sun[, decl := NULL]
    sun[, eo := NULL]
    sun[, eqtime := NULL]
    sun[, ws := NULL]
    sun[, Bo0d := NULL]

    #Column Dates with Times
    sun[, Dates := as.POSIXct(Dates, Times, tz = 'UTC')]
    sun[, Times := NULL]

    #keep night
    if(!keep.night){
        sun <- sun[night == FALSE]
    }

    return(sun)
}
```

fSombra

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

    stopifnot(modeTrk %in% c('two','horiz','fixed'))
    res=switch(modeTrk,
               two={fSombra6(angGen, distances, struct, prom)},
               horiz={fSombraHoriz(angGen, distances, struct)},
               fixed= {fSombraEst(angGen, distances, struct)}
               )
    return(res)
}
```

```
fSombra2X<-function(angGen,distances,struct)
{
    stopifnot(is.list(struct),is.data.frame(distances))
    ##I prepare starting data
    P=with(struct,distances/W)
```

```
    b=with(struct,L/W)
    AzS=angGen$AzS
    Beta=angGen$Beta
    AlS=angGen$AlS

    d1=abs(P$Lew*cos(AzS)-P$Lns*sin(AzS))
    d2=abs(P$Lew*sin(AzS)+P$Lns*cos(AzS))
    FC=sin(AlS)/sin(Beta+AlS)
    s=b*cos(Beta)+(b*sin(Beta)+P$H)/tan(AlS)
    FS1=1-d1
    FS2=s-d2
    SombraCond=(FS1>0)*(FS2>0)*(P$Lew*AzS>=0)
    SombraCond[is.na(SombraCond)]<-FALSE #NAs are of no use to me in a logical
    vector. I replace them with FALSE
    ## Result
    FS=SombraCond*(FS1*FS2*FC)/b
    FS[FS>1]<-1
    return(FS)
}
```

```
fSombra6<-function(angGen, distances, struct, prom=TRUE)
{
    stopifnot(is.list(struct),
              is.data.frame(distances))
    ##distances only has three distances, so I generate a grid
    if (dim(distances)[1]==1){
        Red <- distances[, .(Lew = c(-Lew, 0, Lew, -Lew, Lew),
                             Lns = c(Lns, Lns, Lns, 0, 0),
                             H=H)]
    } else { #distances is an array, so there is no need to generate the grid
        Red<-distances[1:5,]} #I only need the first 5 rows...necessary in case a
     wrong data.frame is delivered

    ## I calculate the shadow due to each of the 5 followers
    SombraGrupo<-matrix(ncol=5,nrow=dim(angGen)[1]) ###VECTORIZE
    for (i in 1:5) {SombraGrupo[,i]<-fSombra2X(angGen,Red[i,],struct)}
    ##To calculate the Average Shadow, I need the number of followers in each
    position (distrib)
    distrib=with(struct,c(1,Ncol-2,1,Nrow-1,(Ncol-2)*(Nrow-1),Nrow-1))
    vProm=c(sum(distrib[c(5,6)]),
            sum(distrib[c(4,5,6)]),
            sum(distrib[c(4,5)]),
            sum(distrib[c(2,3,5,6)]),
            sum(distrib[c(1,2,4,5)]))
    Nseg=sum(distrib) ##Total number of followers
    ##With the SWEEP function I multiply the Shadow Factor of each type (
    ShadowGroup columns) by the vProm result

    if (prom==TRUE){
        ## Average Shadow Factor in the group of SIX followers taking into
    account distribution
        FS=rowSums(sweep(SombraGrupo,2,vProm,'*'))/Nseg
        FS[FS>1]<-1
    } else {
        ## Shadow factor on follower #5 due to the other 5 followers
        FS=rowSums(SombraGrupo)
```

```
        FS[FS>1]<-1}
    return(FS)
}
```

```
fSombraEst<-function(angGen, distances, struct)
{
    stopifnot(is.list(struct),is.data.frame(distances))
    ## I prepare starting data
    dist <- with(struct, distances/L)
    Alfa <- angGen$Alfa
    Beta <- angGen$Beta
    AlS <- angGen$AlS
    AzS <- angGen$AzS
    cosTheta <- angGen$cosTheta
    h <- dist$H #It must be previously normalized
    d <- dist$D
    ## Calculations
    s=cos(Beta)+cos(Alfa-AzS)*(sin(Beta)+h)/tan(AlS)
    FC=sin(AlS)/sin(Beta+AlS)
    SombraCond=(s-d>0)
    FS=(s-d)*SombraCond*FC*(cosTheta>0)
    ## Result
    FS=FS*(FS>0)
    FS[FS>1]<-1
    return(FS)
}
```

```
fSombraHoriz<-function(angGen, distances, struct)
{
    stopifnot(is.list(struct),is.data.frame(distances))
    ## I prepare starting data
    d <- with(struct, distances/L)
    AzS <- angGen$AzS
    AlS <- angGen$AlS
    Beta <- angGen$Beta
    lew <- d$Lew #It must be previously normalized
    ## Calculations
    Beta0=atan(abs(sin(AzS)/tan(AlS)))
    FS=1-lew*cos(Beta0)/cos(Beta-Beta0)
    SombraCond=(FS>0)
    ## Result
    FS=FS*SombraCond
    FS[FS>1]<-1
    return(FS)
}
```

fTemp

```
fTemp<-function(sol, BD)
{
    ##sol is an object with class='Sol'
    ##BD is an object with class='Meteo', whose 'data' slot contains two columns
    called "TempMax" and "TempMin"

```

```
    stopifnot(class(sol)=='Sol')
    stopifnot(class(BD)=='Meteo')

    checkIndexD(indexD(sol), indexD(BD))

    Dates<-indexI(sol)
    x <- as.Date(Dates)
    ind.rep <- cumsum(c(1, diff(x) != 0))

    TempMax <-  BD@data$TempMax[ind.rep]
    TempMin <-  BD@data$TempMin[ind.rep]
    ws <- sol@solD$ws[ind.rep]
    w <- sol@solI$w

    ##Generate temperature sequence from database Maxima and Minima

    Tm=(TempMin+TempMax)/2
    Tr=(TempMax-TempMin)/2

    wp=pi/4

    a1=pi*12*(ws-w)/(21*pi+12*ws)
    a2=pi*(3*pi-12*w)/(3*pi-12*ws)
    a3=pi*(24*pi+12*(ws-w))/(21*pi+12*ws)

    T1=Tm-Tr*cos(a1)
    T2=Tm+Tr*cos(a2)
    T3=Tm-Tr*cos(a3)

    Ta=T1*(w<=ws)+T2*(w>ws&w<=wp)+T3*(w>wp)

    ##Result
    result<-data.table(Dates, Ta)
}
```

fTheta

```
fTheta<-function(sol, beta, alfa=0, modeTrk='fixed', betaLim=90,
                 BT=FALSE, struct, dist)
{
    stopifnot(modeTrk %in% c('two','horiz','fixed'))
    if (!missing(struct)) {stopifnot(is.list(struct))}
    if (!missing(dist)) {stopifnot(is.data.frame(dist))}

    betaLim=d2r(betaLim)
    lat=getLat(sol, 'rad')
    signLat=ifelse(sign(lat)==0, 1, sign(lat)) ##When lat=0, sign(lat)=0. I
    change it to sign(lat)=1

    solI<-as.data.tableI(sol, complete=TRUE, day = TRUE)
    AlS=solI$AlS
    AzS=solI$AzS
    decl=solI$decl
    w<-solI$w

```

```
    night<-solI$night

    Beta<-switch(modeTrk,
                 two = {Beta2x=pi/2-AlS
                     Beta=Beta2x+(betaLim-Beta2x)*(Beta2x>betaLim)},
                 fixed = rep(d2r(beta), length(w)),
                 horiz={BetaHoriz0=atan(abs(sin(AzS)/tan(AlS)))
                     if (BT){lew=dist$Lew/struct$L
                         Longitud=lew*cos(BetaHoriz0)
                         Cond=(Longitud>=1)
                         Longitud[Cond]=1
                         ## When Cond==TRUE Length=1
                         ## and therefore asin(Length)=pi/2,
                         ## so that BetaHoriz=BetaHoriz0
                         BetaHoriz=BetaHoriz0+asin(Longitud)-pi/2
                     } else {
                         BetaHoriz=BetaHoriz0
                         rm(BetaHoriz0)}
                     Beta=ifelse(BetaHoriz>betaLim,betaLim,BetaHoriz)}
                 )
    is.na(Beta) <- night

    Alfa<-switch(modeTrk,
                 two = AzS,
                 fixed = rep(d2r(alfa), length(w)),
                 horiz=pi/2*sign(AzS))
    is.na(Alfa) <- night

    cosTheta<-switch(modeTrk,
                     two=cos(Beta-(pi/2-AlS)),
                     horiz={
                         t1=sin(decl)*sin(lat)*cos(Beta)
                         t2=cos(decl)*cos(w)*cos(lat)*cos(Beta)
                         t3=cos(decl)*abs(sin(w))*sin(Beta)
                         cosTheta=t1+t2+t3
                         rm(t1,t2,t3)
                         cosTheta
                     },
                     fixed={
                         t1=sin(decl)*sin(lat)*cos(Beta)
                         t2=-signLat*sin(decl)*cos(lat)*sin(Beta)*cos(Alfa)
                         t3=cos(decl)*cos(w)*cos(lat)*cos(Beta)
                         t4=signLat*cos(decl)*cos(w)*sin(lat)*sin(Beta)*cos(Alfa)
                         t5=cos(decl)*sin(w)*sin(Alfa)*sin(Beta)
                         cosTheta=t1+t2+t3+t4+t5
                         rm(t1,t2,t3,t4,t5)
                         cosTheta
                     }
                     )
    is.na(cosTheta) <- night
    cosTheta=cosTheta*(cosTheta>0) #when cosTheta<0, Theta is greater than 90º,
   and therefore the Sun is behind the panel.

    result <- data.table(Dates = indexI(sol),
                         Beta, Alfa, cosTheta)
    return(result)
```

```
}
```

HQCurve

```
## HQCurve: no visible binding for global variable ‘fb'
## HQCurve: no visible binding for global variable ‘Q'
## HQCurve: no visible binding for global variable ‘x'
## HQCurve: no visible binding for global variable ‘y'
## HQCurve: no visible binding for global variable ‘group.value'

if(getRversion() >= "2.15.1") globalVariables(c('fb', 'Q', 'x', 'y', 'group.value'))

HQCurve<-function(pump){
  w1=3000 #synchronous rpm frequency
  wm=2870 #rpm frequency with slip when applying voltage at 50 Hz
  s=(w1-wm)/w1
  fen=50 #Nominal electrical frequency

  f=seq(35,50,by=5)
  Hn=with(pump,a*50^2+b*50*Qn+c*Qn^2) #height corresponding to flow rate and nominal frequency

  kiso=Hn/pump$Qn^2 #To paint the isoyield curve I take into account the laws of similarity
  Qiso=with(pump,seq(0.1*Qn,Qmax,l=10))
  Hiso=kiso*Qiso^2 #Isoperformance curve

  Curva<-expand.grid(fb=f,Q=Qiso)

  Curva<-within(Curva,{
    fe=fb/(1-s)
    H=with(pump,a*fb^2+b*fb*Q+c*Q^2)

    is.na(H) <- (H<0)
    Q50=50*Q/fb
    H50=H*(50/fb)^2
    etab=with(pump,j*Q50^2+k*Q50+l)
    Pb50=2.725*H50*Q50/etab
    Pb=Pb50*(fb/50)^3

    Pbc=Pb*50/fe
    etam=with(pump,g*(Pbc/Pmn)^2+h*(Pbc/Pmn)+i)
    Pmc=Pbc/etam
    Pm=Pmc*fe/50

    etac=0.95 #Variable frequency drive performance
    cab=0.05  #Cable losses
    Pdc=Pm/(etac*(1-cab))
    rm(etac,cab,Pmc,Pbc,Pb50,Q50,H50)
  })

###H-Q curve at different frequencies
  ##I check if I have the lattice package available, which should have been loaded in .First.lib
```

```
  lattice.disp<-("lattice" %in% .packages())
  latticeExtra.disp<-("latticeExtra" %in% .packages())
  if (lattice.disp && latticeExtra.disp) {
    p<-xyplot(H~Q,groups=factor(fb),data=Curva, type='l',
              par.settings=custom.theme.2(),
              panel=function(x,y,groups,...){
                panel.superpose(x,y,groups,...)
                panel.xyplot(Qiso,Hiso,col='black',...)
                panel.text(Qiso[1], Hiso[1], 'ISO', pos=3)}
              )
    p=p+glayer(panel.text(x[1], y[1], group.value, pos=3))
    print(p)
    result<-list(result=Curva, plot=p)
  } else {
    warning('lattice and/or latticeExtra packages are not available. Thus, the
    plot could not be created')
    result<-Curva}
}
```

local2Solar

```
local2Solar <- function(x, lon=NULL){
  tz=attr(x, 'tzone')
  if (tz=='' || is.null(tz)) {tz='UTC'}
  ##Daylight savings time
  AO=3600*dst(x)
  AOneg=(AO<0)
  if (any(AOneg)) {
    AO[AOneg]=0
    warning('Some Daylight Savings Time unknown. Set to zero.')
  }
  ##Difference between local longitude and time zone longitude LH
  LH=lonHH(tz)
  if (is.null(lon))
    {deltaL=0
   } else
  {deltaL=d2r(lon)-LH
 }
  ##Local time corrected to UTC
  tt <- format(x, tz=tz)
  result <- as.POSIXct(tt, tz='UTC')-AO+r2sec(deltaL)
  result
}
```

markovG0

```
## Objects loaded at startup from data/MTM.RData
if(getRversion() >= "2.15.1") globalVariables(c(
                  'MTM', ## Markov Transition Matrices
                  'Ktmtm', ## Kt limits to choose a matrix from MTM
                  'Ktlim' ## Daily kt range of each matrix.
                  ))

markovG0 <- function(G0dm, solD){
```

```
    solD <- copy(solD)
    timeIndex <- solD$Dates
    Bo0d <- solD$Bo0d
    Bo0dm <- solD[, mean(Bo0d), by = .(month(Dates), year(Dates))][[3]]
    ktm <- G0dm/Bo0dm

    ##Calculates which matrix to work with for each month
    whichMatrix <- findInterval(ktm, Ktmtm, all.inside = TRUE)

    ktd <- state <- numeric(length(timeIndex))
    state[1] <- 1
    ktd[1] <- ktm[state[1]]
    for (i in 2:length(timeIndex)){
        iMonth <- month(timeIndex[i])
        colMonth <- whichMatrix[iMonth]
        rng <- Ktlim[, colMonth]
        classes <- seq(rng[1], rng[2], length=11)
        matMonth <- MTM[(10*colMonth-9):(10*colMonth),]
        ## http://www-rohan.sdsu.edu/~babailey/stat575/mcsim.r
        state[i] <- sample(1:10, size=1, prob=matMonth[state[i-1],])
        ktd[i] <- runif(1, min=classes[state[i]], max=classes[state[i]+1])
    }
    G0dmMarkov <- data.table(ktd, Bo0d)
    G0dmMarkov <- G0dmMarkov[, mean(ktd*Bo0d), by = .(month(timeIndex), year(
   timeIndex))][[3]]
    fix <- G0dm/G0dmMarkov
    indRep <- month(timeIndex)
    fix <- fix[indRep]
    G0d <- data.table(Dates = timeIndex, G0d = ktd * Bo0d * fix)
    G0d
}
```

NmgPVPS

```
## NmgPVPS: no visible binding for global variable ‘Pnom'
## NmgPVPS: no visible binding for global variable ‘group.value'

if(getRversion() >= "2.15.1") globalVariables(c('Pnom', 'group.value'))

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

    ##I build the type day by IEC procedure
    t=seq(-t0,t0,l=2*t0*Nm);
    d=Gd/(Gmax*2*t0)
    s=(d*pi/2-1)/(1-pi/4)
    G=Gmax*cos(t/t0*pi/2)*(1+s*(1-cos(t/t0*pi/2)))
    G[G<0]<-0
    G=G/(sum(G,na.rm=1)/Nm)*Gd
    Red<-expand.grid(G=G,Pnom=Pg,H=H,Ta=Ta)
    Red<-within(Red,{Tcm<-Ta+G*(TONC-20)/800
                     Pdc=Pnom*G/1000*(1-lambda*(Tcm-25)) #Available DC power
                     Pac=Pdc*eta})                       #Inverter yield
```

```

    res=data.table(Red,Q=0)

    for (i in seq_along(H)){
        fun=fPump(pump, H[i])
        Cond=res$H==H[i]
        x=res$Pac[Cond]
        z=res$Pdc[Cond]
        rango=with(fun,x>=lim[1] & x<=lim[2]) #I limit the power to the operating range of the pump.
        x[!rango]<-0
        z[!rango]<-0
        y=res$Q[Cond]
        y[rango]<-fun$fQ(x[rango])
        res$Q[Cond]=y
        res$Pac[Cond]=x
        res$Pdc[Cond]=z
    }

    resumen <- res[, lapply(.SD, function(x)sum(x, na.rm = 1)/Nm),
                   by = .(Pnom, H)]
    param=list(pump=pump, Pg=Pg, H=H, Gd=Gd, Ta=Ta,
               lambda=lambda, TONC=TONC, eta=eta,
               Gmax=Gmax, t0=t0, Nm=Nm)


###Abacus with common X-axes

    ##I check if I have the lattice package available, which should have been loaded in .First.lib
    lattice.disp<-("lattice" %in% .packages())
    latticeExtra.disp<-("latticeExtra" %in% .packages())
    if (lattice.disp && latticeExtra.disp){
        tema<-theme
        tema1 <- modifyList(tema, list(layout.width = list(panel=1,
                                       ylab = 2, axis.left=1.0,
                                       left.padding=1, ylab.axis.padding=1,
                                       axis.panel=1)))
        tema2 <- modifyList(tema, list(layout.width = list(panel=1,
                                       ylab = 2, axis.left=1.0, left.padding=1,
                                       ylab.axis.padding=1, axis.panel=1)))
        temaT <- modifyList(tema, list(layout.heights = list(panel = c(1, 1))))
        p1 <- xyplot(Q~Pdc, groups=H, data=resumen,
                     ylab="Qd (m\u00b3/d)",type=c('l','g'),
                     par.settings = tema1)

        p1lab<-p1+glayer(panel.text(x[1], y[1], group.value, pos=2, cex=0.7))

        ##I paint the linear regression because Pnom~Pdc depends on the height.
        p2 <- xyplot(Pnom~Pdc, groups=H, data=resumen,
                     ylab="Pg",type=c('l','g'), #type=c('smooth','g'),
                     par.settings = tema2)
        p2lab<-p2+glayer(panel.text(x[1], y[1], group.value, pos=2, cex=0.7))

        p<-update(c(p1lab, p2lab, x.same = TRUE),
                  main=paste(title, '\nSP', pump$Qn, 'A', pump$stages, ' ',
```

```
                  'Gd ', Gd/1000," kWh/m\u00b2",sep=''),
                  layout = c(1, 2),
                  scales=list(x=list(draw=FALSE)),
                  xlab='',
                  ylab = list(c("Qd (m\u00b3/d)","Pg (Wp)"), y = c(1/4, 3/4)),
                  par.settings = temaT
                  )
        print(p)
        result<-list(I=res,D=resumen, plot=p, param=param)
    } else {
        warning('lattice, latticeExtra packages are not all available. Thus, the
    plot could not be created')
        result<-list(I=res,D=resumen, param=param)
    }
}
```

utils-angle

```
#degrees to radians
d2r<-function(x){x*pi/180}

#radians to degrees
r2d<-function(x){x*180/pi}

#hours to radians
h2r<-function(x){x*pi/12}

#hours to degrees
h2d<-function(x){x*180/12}

#radians to hours
r2h<-function(x){x*12/pi}

#degrees to hours
d2h<-function(x){x*12/180}

#radians to seconds
r2sec<-function(x){x*12/pi*3600}

#radians to minutes
r2min<-function(x){x*12/pi*60}
```

utils-time

```
#complete time to hours
t2h <- function(x)
{
    hour(x)+minute(x)/60+second(x)/3600
}

#hours minutes and seconds to hours
hms <- function(x)
{
    hour(x)+minute(x)/60+second(x)/3600
```

```
}

#day of the year
doy <- function(x){
  as.numeric(format(x, '%j'))
}

#day of the month
dom <- function(x){
  as.numeric(format(x, '%d'))
}

#trunc days
truncDay <- function(x){as.POSIXct(trunc(x, units='days'))}
```

A.4. Métodos

as.data.tableI

as.data.tableD

```
setGeneric('as.data.tableD', function(object, complete=FALSE, day=FALSE){
    standardGeneric('as.data.tableD')})

setMethod('as.data.tableD',
          signature=(object='Sol'),
          definition=function(object, complete=FALSE, day=FALSE){
              sol <- copy(object)
              solD <- sol@solD
              data <- solD
              if(day){
                  ind <- indexD(object)
                  data[, day := doy(ind)]
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )

setMethod('as.data.tableD',
          signature = (object='G0'),
          definition = function(object, complete=FALSE, day=FALSE){
              g0 <- copy(object)
              G0D <- g0@G0D
              solD <- g0@solD
              if(complete){
                  data <- data.table(G0D, solD[, Dates := NULL])
              } else {
                  G0D[, Fd := NULL]
                  G0D[, Kt := NULL]
                  data <- G0D
              }
              if(day){
                  ind <- indexD(object)
```

```
                data[, day := doy(ind)]
                data[, month := month(ind)]
                data[, year := year(ind)]
            }
            return(data)
        })

setMethod('as.data.tableD',
          signature = (object='Gef'),
          definition = function(object, complete=FALSE, day=FALSE){
              gef <- copy(object)
              GefD <- gef@GefD
              GOD <- gef@GOD
              solD <- gef@solD
              if(complete){
                  data <- data.table(GefD,
                                     GOD[, Dates := NULL],
                                     solD[, Dates := NULL])
              } else {data <- GefD[, c('Dates', 'Gefd',
                                       'Defd', 'Befd')]}
              if(day){
                  ind <- indexD(object)
                  data[, day := doy(ind)]
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )

setMethod('as.data.tableD',
          signature = (object='ProdGCPV'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodgcpv <- copy(object)
              prodD <- prodgcpv@prodD
              GefD <- prodgcpv@GefD
              GOD <- prodgcpv@GOD
              solD <- prodgcpv@solD
              if(complete){
                  data <- data.table(prodD,
                                     GefD[, Dates := NULL],
                                     GOD[, Dates := NULL],
                                     solD[, Dates := NULL]
                                     )
              } else { data <- prodD[, c('Dates', 'Eac',
                                         'Edc', 'Yf')]}
              if(day){
                  ind <- indexD(object)
                  data[, day := doy(ind)]
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )
```

```
setMethod('as.data.tableD',
          signature = (object='ProdPVPS'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodpvps <- copy(object)
              prodD <- prodpvps@prodD
              GefD <- prodpvps@GefD
              G0D <- prodpvps@G0D
              solD <- prodpvps@solD
              if(complete){
                  data <- data.table(prodD,
                                     GefD[, Dates := NULL],
                                     G0D[, Dates := NULL],
                                     solD[, Dates := NULL]
                                     )
              } else { data <- prodD[, c('Dates', 'Eac',
                                         'Qd', 'Yf')]}
              if(day){
                  ind <- indexD(object)
                  data[, day := doy(ind)]
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )
```

as.data.tableM

```
setGeneric('as.data.tableM', function(object, complete = FALSE, day=FALSE){
    standardGeneric('as.data.tableM')})

setMethod('as.data.tableM',
          signature=(object='G0'),
          definition=function(object, complete=FALSE, day=FALSE){
              g0 <- copy(object)
              G0dm <- g0@G0dm
              data <- G0dm
              if(day){
                  ind <- indexD(object)
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )

setMethod('as.data.tableM',
          signature=(object='Gef'),
          definition = function(object, complete=FALSE, day=FALSE){
              gef <- copy(object)
              Gefdm <- gef@Gefdm
              G0dm <- gef@G0dm
              if(complete){
                  data <- data.table(Gefdm, G0dm[, Dates := NULL])
              } else {data <- Gefdm}
```

```
            if(day){
                ind <- indexD(object)
                data[, month := month(ind)]
                data[, year := year(ind)]
            }
            return(data)
        }
        )

setMethod('as.data.tableM',
          signature = (object='ProdGCPV'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodgcpv <- copy(object)
              prodDm <- prodgcpv@prodDm
              Gefdm <- prodgcpv@Gefdm
              G0dm <- prodgcpv@G0dm
              if(complete){
                  data <- data.table(prodDm,
                                     Gefdm[, Dates := NULL],
                                     G0dm[, Dates := NULL])
              } else {data <- prodDm}
              if(day){
                  ind <- indexD(object)
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )

setMethod('as.data.tableM',
          signature = (object='ProdPVPS'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodpvps <- copy(object)
              prodDm <- prodpvps@prodDm
              Gefdm <- prodpvps@Gefdm
              G0dm <- prodpvps@G0dm
              if(complete){
                  data <- data.table(prodDm,
                                     Gefdm[, Dates := NULL],
                                     G0dm[, Dates := NULL])
              } else {data <- prodDm}
              if(day){
                  ind <- indexD(object)
                  data[, month := month(ind)]
                  data[, year := year(ind)]
              }
              return(data)
          }
          )
```

as.data.tableY

```
setGeneric('as.data.tableY', function(object, complete=FALSE, day=FALSE){
    standardGeneric('as.data.tableY')})
```

```

setMethod('as.data.tableY',
          signature=(object='G0'),
          definition=function(object, complete=FALSE, day=FALSE){
              g0 <- copy(object)
              G0y <- g0@G0y
              data <- G0y
              if(day){data[, year := Dates]}
              return(data)
          }
          )

setMethod('as.data.tableY',
          signature = (object='Gef'),
          definition = function(object, complete=FALSE, day=FALSE){
              gef <- copy(object)
              Gefy <- gef@Gefy
              G0y <- gef@G0y
              if(complete){
                  data <- data.table(Gefy, G0y[, Dates := NULL])
              } else {data <- Gefy}
              if(day){data[, year := Dates]}
              return(data)
          }
          )

setMethod('as.data.tableY',
          signature = (object='ProdGCPV'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodgcpv <- copy(object)
              prody <- prodgcpv@prody
              Gefy <- prodgcpv@Gefy
              G0y <- prodgcpv@G0y
              if(complete){
                  data <- data.table(prody,
                                     Gefy[, Dates := NULL],
                                     G0y[, Dates := NULL])
              } else {data <- prody}
              if(day){data[, year := Dates]}
              return(data)
          }
          )

setMethod('as.data.tableY',
          signature = (object='ProdPVPS'),
          definition = function(object, complete=FALSE, day=FALSE){
              prodpvps <- copy(object)
              prody <- prodpvps@prody
              Gefy <- prodpvps@Gefy
              G0y <- prodpvps@G0y
              if(complete){
                  data <- data.table(prody,
                                     Gefy[, Dates := NULL],
                                     G0y[, Dates := NULL])
              } else {data <- prody}
              if(day){data[, year := Dates]}
```

```
            return(data)
          }
          )
```

compare

```
## compareFunction: no visible binding for global variable ‘name'
## compareFunction: no visible binding for global variable ‘x'
## compareFunction: no visible binding for global variable ‘y'
## compareFunction: no visible binding for global variable ‘group.value'

if(getRversion() >= "2.15.1") globalVariables(c('name', 'x', 'y', 'group.value'))

setGeneric('compare', signature='...', function(...){standardGeneric('compare')})

compareFunction <- function(..., vars){
    dots <- list(...)
    nms0 <- substitute(list(...))
    if (!is.null(names(nms0))){ ##in do.call
        nms <- names(nms0[-1])
    } else {
        nms <- as.character(nms0[-1])
    }
    foo <- function(object, label){
        yY <- colMeans(as.data.tableY(object, complete = TRUE)[, ..vars])
        yY <- cbind(stack(yY), name=label)
        yY
    }
    cdata <- mapply(FUN=foo, dots, nms, SIMPLIFY=FALSE)
    z <- do.call(rbind, cdata)
    z$ind <- ordered(z$ind, levels=vars)
    p <- dotplot(ind~values, groups=name, data=z, type='b',
                 par.settings=solaR.theme)
    print(p+glayer(panel.text(x[length(x)], y[length(x)],
                              label=group.value, cex=0.7, pos=3, srt=45)))
    return(z)
}


setMethod('compare',
          signature='G0',
          definition=function(...){
            vars <- c('D0d', 'B0d', 'G0d')
            res <- compareFunction(..., vars=vars)
            return(res)
          }
          )

setMethod('compare',
          signature='Gef',
          definition=function(...){
            vars <- c('Defd', 'Befd', 'Gefd')
            res <- compareFunction(..., vars=vars)
            return(res)
          }
```

```
          )

setMethod('compare',
          signature='ProdGCPV',
          definition=function(...){
            vars <- c('G0d', 'Gefd', 'Yf')
            res <- compareFunction(..., vars=vars)
            return(res)
          }
          )
```

getData

```
## extracts the data for class Meteo ##
setGeneric('getData', function(object){standardGeneric('getData')})

### getData ####
setMethod('getData',
          signature = (object = 'Meteo'),
          definition = function(object){
              result <- object@data
              return(result)
          })
```

getG0

```
## extracts the global irradiance for class Meteo ##
setGeneric('getG0', function(object){standardGeneric('getG0')})

### getG0 ###
setMethod('getG0',
          signature = (object = 'Meteo'),
          definition = function(object){
              result <- getData(object)
              return(result$G0)
          })
```

getLat

```
## extracts the latitude from the objects ##
setGeneric('getLat', function(object, units = 'rad')
{standardGeneric('getLat')})

## extracts the latitude from the objects ##
setGeneric('getLat', function(object, units = 'rad')
{standardGeneric('getLat')})

setMethod('getLat',
          signature = (object = 'Meteo'),
          definition = function(object, units = 'rad'){
              stopifnot(units %in% c('deg', 'rad'))
              result = switch(units,
                              rad = d2r(object@latm),
```

```
                              deg = object@latm)
              return(result)
          })
```

indexD

```
## extract the index of the daily data ##
setGeneric('indexD', function(object){standardGeneric('indexD')})
### indexD ###
setMethod('indexD',
          signature = (object = 'Sol'),
          definition = function(object){as.POSIXct(object@solD$Dates)
          })

setMethod('indexD',
          signature = (object = 'Meteo'),
          definition = function(object){as.POSIXct(getData(object)$Dates)})
```

indexI

```
## extract the index of the intradaily data ##
setGeneric('indexI', function(object){standardGeneric('indexI')})
### indexI ###
setMethod('indexI',
          signature = (object = 'Sol'),
          definition = function(object){as.POSIXct(object@solI$Dates)
          })
```

levelplot

```
setGeneric('levelplot')

setMethod('levelplot',
          signature=c(x='formula', data='Meteo'),
          definition=function(x, data,
                              par.settings = solaR.theme,
                              panel = panel.levelplot.raster, interpolate = TRUE,
                              xscale.components = xscale.solar,
                              yscale.components = yscale.solar,
                              ...){
              data0=getData(data)
              ind=data0$Dates
              data0$day=doy(ind)
              data0$month=month(ind)
              data0$year=year(ind)
              data0$w=h2r(hms(ind)-12)
              levelplot(x, data0,
                        par.settings = par.settings,
                        xscale.components = xscale.components,
                        yscale.components = yscale.components,
                        panel = panel, interpolate = interpolate,
                        ...)
          }
```

```
          )

setMethod('levelplot',
          signature=c(x='formula', data='Sol'),
          definition=function(x, data,
                              par.settings = solaR.theme,
                              panel = panel.levelplot.raster, interpolate = TRUE,
                              xscale.components = xscale.solar,
                              yscale.components = yscale.solar,
                              ...){
              data0=as.data.tableI(data, complete=TRUE, day=TRUE)
              ind=data0$Dates
              data0$day=doy(ind)
              data0$month=month(ind)
              data0$year=year(ind)
              levelplot(x, data0,
                        par.settings = par.settings,
                        xscale.components = xscale.components,
                        yscale.components = yscale.components,
                        panel = panel, interpolate = interpolate,
                        ...)
          }
          )

setMethod('levelplot',
          signature=c(x='formula', data='G0'),
          definition=function(x, data,
                              par.settings = solaR.theme,
                              panel = panel.levelplot.raster, interpolate = TRUE,
                              xscale.components = xscale.solar,
                              yscale.components = yscale.solar,
                              ...){
              data0=as.data.tableI(data, complete=TRUE, day=TRUE)
              ind=data0$Dates
              data0$day=doy(ind)
              data0$month=month(ind)
              data0$year=year(ind)
              levelplot(x, data0,
                        par.settings = par.settings,
                        xscale.components = xscale.components,
                        yscale.components = yscale.components,
                        panel = panel, interpolate = interpolate,
                        ...)
          }
          )
```

losses

```
setGeneric('losses', function(object){standardGeneric('losses')})

setMethod('losses',
          signature=(object='Gef'),
          definition=function(object){
            dat <- as.data.tableY(object, complete=TRUE)
            isShd=('Gef0d' %in% names(dat)) ##is there shadows?
```

```
            if (isShd) {
              shd <- with(dat, mean(1-Gefd/Gef0d))
              eff <- with(dat, mean(1-Gef0d/Gd))
            } else {
              shd <- 0
              eff <- with(dat, mean(1-Gefd/Gd))
            }
            result <- data.table(Shadows = shd, AoI = eff)
            result
          }
          )

setMethod('losses',
          signature=(object='ProdGCPV'),
          definition=function(object){
              datY <- as.data.tableY(object, complete=TRUE)
              module0=object@module
              module0$CoefVT=0 ##No losses with temperature
              Pg=object@generator$Pg
              Nm=1/sample2Hours(object@sample)
              datI <- as.data.tableI(object, complete=TRUE)
              if (object@type=='prom'){
                  datI[, DayOfMonth := DOM(datI)]
                  YfDC0 <- datI[, sum(Vmpp*Impp/Pg*DayOfMonth, na.rm = TRUE),
                                by = month(Dates)][[2]]
                  YfDC0 <- sum(YfDC0, na.rm = TRUE)
                  YfAC0 <- datI[, sum(Pdc*EffI/Pg*DayOfMonth, na.rm = TRUE),
                                by = month(Dates)][[2]]
                  YfAC0 <- sum(YfAC0, na.rm = TRUE)
              } else {
                  datI[, DayOfMonth := DOM(datI)]
                  YfDC0 <- datI[, sum(Vmpp*Impp/Pg*DayOfMonth, na.rm = TRUE),
                                by = year(Dates)][[2]]
                  YfAC0 <- datI[, sum(Pdc*EffI/Pg*DayOfMonth, na.rm = TRUE),
                                by = year(Dates)][[2]]
              }
              gen <- mean(1-YfDC0/datY$Gefd)
              YfDC <- datY$Edc/Pg*1000
              DC=mean(1-YfDC/YfDC0)
              inv=mean(1-YfAC0/YfDC)
              AC=mean(1-datY$Yf/YfAC0)
              result0 <- losses(as(object, 'Gef'))
              result1 <- data.table(Generator = gen,
                                    DC = DC,
                                    Inverter = inv,
                                    AC = AC)
              result <- data.table(result0, result1)
              result
          }
          )

###compareLosses

## compareLosses,ProdGCPV: no visible binding for global variable 'name'
if(getRversion() >= "2.15.1") globalVariables(c('name'))
```

```
setGeneric('compareLosses', signature='...', function(...){standardGeneric('
    compareLosses')})

setMethod('compareLosses', 'ProdGCPV',
          definition=function(...){
            dots <- list(...)
            nms0 <- substitute(list(...))
            if (!is.null(names(nms0))){ ##do.call
              nms <- names(nms0[-1])
            } else {
              nms <- as.character(nms0[-1])
            }
            foo <- function(object, label){
              yY <- losses(object)
              yY <- cbind(yY, name=label)
              yY
            }
            cdata <- mapply(FUN=foo, dots, nms, SIMPLIFY=FALSE)
            z <- do.call(rbind, cdata)
            z <- melt(z, id.vars = 'name')
            p <- dotplot(variable~value*100, groups=name, data=z,
                         par.settings=solaR.theme, type='b',
                         auto.key=list(corner=c(0.95,0.2), cex=0.7), xlab='Losses
     (%)')
            print(p)
            return(z)
          }
          )
```

mergeSolar

```
setGeneric('mergesolaR', signature='...', function(...){standardGeneric('
    mergesolaR')})

fooMeteo <- function(object, var){yY <- getData(object)[, .SD,
                                                        by = Dates,
                                                        .SDcols = var]}

fooG0 <- function(object, var){yY <- as.data.tableD(object)[, .SD,
                                                            by = Dates,
                                                            .SDcols = var]}

mergeFunction <- function(..., foo, var){
    dots <- list(...)
    dots <- lapply(dots, as, class(dots[[1]])) ##the first element is the one
    that dictates the class to everyone
    nms0 <- substitute(list(...))
    if (!is.null(names(nms0))){ ##do.call
        nms <- names(nms0[-1])
    } else {
        nms <- as.character(nms0[-1])
    }
    cdata <- sapply(dots, FUN=foo, var, simplify=FALSE)
    z <- cdata[[1]]
    for (i in 2:length(cdata)){
```

```
        z <- merge(z, cdata[[i]], by = 'Dates', suffixes = c("", paste0('.', i)))
    }
    names(z)[-1] <- nms
    z
}

setMethod('mergesolaR',
          signature='Meteo',
          definition=function(...){
            res <- mergeFunction(..., foo=fooMeteo, var='G0')
            res
          }
          )

setMethod('mergesolaR',
          signature='G0',
          definition=function(...){
            res <- mergeFunction(..., foo=fooG0, var='G0d')
            res
          }
          )

setMethod('mergesolaR',
          signature='Gef',
          definition=function(...){
            res <- mergeFunction(..., foo=fooG0, var='Gefd')
            res
          }
          )

setMethod('mergesolaR',
          signature='ProdGCPV',
          definition=function(...){
            res <- mergeFunction(..., foo=fooG0, var='Yf')
            res
          }
          )

setMethod('mergesolaR',
          signature='ProdPVPS',
          definition=function(...){
            res <- mergeFunction(..., foo=fooG0, var='Yf')
            res
          }
          )
```

shadeplot

```
setGeneric('shadeplot', function(x, ...)standardGeneric('shadeplot'))

setMethod('shadeplot', signature(x='Shade'),
          function(x,
                   main='',
                   xlab=expression(L[ew]),
                   ylab=expression(L[ns]),
```

```
               n=9, ...){
          red=x@distances
          FS.loess=x@FS.loess
          Yf.loess=x@Yf.loess
          struct=x@struct
          mode=x@modeTrk
          if (mode=='two'){
              Lew=seq(min(red$Lew),max(red$Lew),length=100)
              Lns=seq(min(red$Lns),max(red$Lns),length=100)
              Red=expand.grid(Lew=Lew,Lns=Lns)
              FS=predict(FS.loess,Red)
              Red$FS=as.numeric(FS)
              AreaG=with(struct,L*W)
              GRR=Red$Lew*Red$Lns/AreaG
              Red$GRR=GRR
              FS.m<-matrix(1-FS,
                           nrow=length(Lew),
                           ncol=length(Lns))
              GRR.m<-matrix(GRR,
                            nrow=length(Lew),
                            ncol=length(Lns))
              niveles=signif(seq(min(FS.m),max(FS.m),l=n+1),3)
              pruebaCB<-("RColorBrewer" %in% .packages())
              if (pruebaCB) {
                  paleta=rev(brewer.pal(n, 'YlOrRd'))
              } else {
                  paleta=rev(heat.colors(n))}
              par(mar=c(4.1,4.1,2.1,2.1))
              filled.contour(x=Lew,y=Lns,z=FS.m,#...,
                             col=paleta, #levels=niveles,
                             nlevels=n,
                             plot.title=title(xlab=xlab,
                                              ylab=ylab, main=main),
                             plot.axes={
                                 axis(1);axis(2);
                                 contour(Lew, Lns, FS.m,
                                         nlevels=n, #levels=niveles,
                                         col="black", labcex=.8,  add=TRUE)
                                 contour(Lew, Lns, GRR.m,
                                         col="black", lty=3, labcex=.8, add=TRUE)
                                 grid(col="white",lty=3)},
                             key.title=title("1-FS",cex.main=.8))
          }
          if (mode=='horiz') {
              Lew=seq(min(red$Lew),max(red$Lew),length=100)
              FS=predict(FS.loess,Lew)
              GRR=Lew/struct$L
              plot(GRR,1-FS,main=main,type='l',...)
              grid()    }
          if (mode=='fixed'){
              D=seq(min(red$D),max(red$D),length=100)
              FS=predict(FS.loess,D)
              GRR=D/struct$L
              plot(GRR,1-FS,main=main,type='l',...)
              grid()    }
```

```
            }
            )
```

window

```
setMethod('[',
          signature='Meteo',
          definition=function(x, i, j,...){
            if (!missing(i)) {
              i <- truncDay(i)
            } else {
              i <- indexD(x)[1]
            }
            if (!missing(j)) {
              j <- truncDay(j)+86400-1 ##The end is the last second of the day
            } else {
              nDays <- length(indexD(x))
              j <- indexD(x)[nDays]+86400-1
            }
            stopifnot(j>i)
            if (!is.null(i)) i <- truncDay(i)
            if (!is.null(j)) j <- truncDay(j)+86400-1
            d <- indexD(x)
            x@data <- x@data[(d >= i & d <= j)]
            x
          }
          )


setMethod('[',
          signature='Sol',
          definition=function(x, i, j, ...){
              if (!missing(i)) {
                  i <- truncDay(i)
              } else {
                  i <- indexD(x)[1]
              }
              if (!missing(j)) {
                  j <- truncDay(j)+86400-1##The end is the last second of the day
              } else {
                  nDays <- length(indexD(x))
                  j <- indexD(x)[nDays]+86400-1
              }
              stopifnot(j>i)
              if(!is.null(i)) i <- truncDay(i)
              if(!is.null(j)) j <- truncDay(j)
              d1 <- indexD(x)
              d2 <- indexI(x)
              x@solD <- x@solD[(d1 >= i & d1 <= j)]
              x@solI <- x@solI[(d2 >= i & d2 <= j)]
              x
          }
          )

setMethod('[',
```

```
          signature='G0',
          definition=function(x, i, j, ...){
              sol <- as(x, 'Sol')[i=i, j=j, ...] ##Sol method
              meteo <- as(x, 'Meteo')[i=i, j=j, ...] ##Meteo method
              i <- indexI(sol)[1]
              j <- indexI(sol)[length(indexI(sol))]
              d1 <- indexD(x)
              d2 <- indexI(x)
              G0Iw <- x@G0I[(d2 >= i & d2 <= j)]
              Taw <- x@Ta[(d2 >= i & d2 <= j)]
              G0dw <- x@G0D[(d1 >= truncDay(i) & d1 <= truncDay(j))]
              G0dmw <- G0dw[, lapply(.SD/1000, mean, na.rm= TRUE),
                            .SDcols = c('G0d', 'D0d', 'B0d'),
                            by = .(month(Dates), year(Dates))]
              if (x@type=='prom'){
                  G0dmw[, DayOfMonth := DOM(G0dmw)]
                  G0yw <- G0dmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                                .SDcols = c('G0d', 'D0d', 'B0d'),
                                by = .(Dates = year)]
                  G0dmw[, DayOfMonth := NULL]
              } else {
                  G0yw <- G0dw[, lapply(.SD/1000, sum, na.rm = TRUE),
                               .SDcols = c('G0d', 'D0d', 'B0d'),
                               by = .(Dates = year(unique(truncDay(Dates))))]
              }
              G0dmw[, Dates := paste(month.abb[month], year, sep = '. ')]
              G0dmw[, c('month', 'year') := NULL]
              setcolorder(G0dmw, 'Dates')
              result <- new('G0',
                            meteo,
                            sol,
                            G0D=G0dw,
                            G0dm=G0dmw,
                            G0y=G0yw,
                            G0I=G0Iw,
                            Ta=Taw)
              result
          }
          )


setMethod('[',
          signature='Gef',
          definition=function(x, i, j, ...){
              g0 <- as(x, 'G0')[i=i, j=j, ...] ##G0 method
              i <- indexI(g0)[1]
              j <- indexI(g0)[length(indexI(g0))]
              d1 <- indexD(x)
              d2 <- indexI(x)
              GefIw <- x@GefI[(d2 >= i & d2 <= j)]
              Thetaw <- x@Theta[(d2 >= i & d2 <= j)]
              Gefdw <- x@GefD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
              nms <- c('Bod', 'Bnd', 'Gd', 'Dd',
                       'Bd', 'Gefd', 'Defd', 'Befd')
              Gefdmw <- Gefdw[, lapply(.SD/1000, mean, na.rm = TRUE),
                              .SDcols = nms,
```

```
                              by = .(month(Dates), year(Dates))]
                if (x@type=='prom'){
                    Gefdmw[, DayOfMonth:= DOM(Gefdmw)]
                    Gefyw <- Gefdmw[, lapply(.SD*DayOfMonth, sum),
                                   .SDcols = nms,
                                   by = .(Dates = year)]
                    Gefdmw[, DayOfMonth := NULL]
                } else {
                    Gefyw <- Gefdw[, lapply(.SD/1000, sum, na.rm = TRUE),
                                  .SDcols = nms,
                                  by = .(Dates = year)]
                }
                Gefdmw[, Dates := paste(month.abb[month], year, sep = '. ')]
                Gefdmw[, c('month', 'year') := NULL]
                setcolorder(Gefdmw, 'Dates')
                result <- new('Gef',
                              g0,
                              GefD=Gefdw,
                              Gefdm=Gefdmw,
                              Gefy=Gefyw,
                              GefI=GefIw,
                              Theta=Thetaw,
                              iS=x@iS,
                              alb=x@alb,
                              modeTrk=x@modeTrk,
                              modeShd=x@modeShd,
                              angGen=x@angGen,
                              struct=x@struct,
                              distances=x@distances
                              )
                result
            }
            )


setMethod('[',
          signature='ProdGCPV',
          definition=function(x, i, j, ...){
              gef <- as(x, 'Gef')[i=i, j=j, ...] ##Gef method
              i <- indexI(gef)[1]
              j <- indexI(gef)[length(indexI(gef))]
              d1 <- indexD(x)
              d2 <- indexI(x)
              prodIw <- x@prodI[(d2 >= i & d2 <= j)]
              prodDw <- x@prodD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
              prodDmw <- prodDw[, lapply(.SD/1000, mean, na.rm = TRUE),
                                .SDcols = c('Eac', 'Edc'),
                                by = .(month(Dates), year(Dates))]
              prodDmw$Yf <- prodDw$Yf
              if (x@type=='prom'){
                  prodDmw[, DayOfMonth := DOM(prodDmw)]
                  prodyw <- prodDmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                                    .SDcols = c('Eac', 'Edc', 'Yf'),
                                    by = .(Dates = year)]
                  prodDmw[, DayOfMonth := NULL]
              } else {
```

```
                prodyw <- prodDw[, lapply(.SD/1000, sum, na.rm = TRUE),
                                 .SDcols = c('Eac', 'Edc', 'Yf'),
                                 by = .(Dates = year)]
            }
              prodDmw[, Dates := paste(month.abb[month], year, sep = '. ')]
              prodDmw[, c('month', 'year') := NULL]
              setcolorder(prodDmw, c('Dates', names(prodDmw)[-length(prodDmw)]))
              result <- new('ProdGCPV',
                            gef,
                            prodD=prodDw,
                            prodDm=prodDmw,
                            prody=prodyw,
                            prodI=prodIw,
                            module=x@module,
                            generator=x@generator,
                            inverter=x@inverter,
                            effSys=x@effSys
                            )
              result
          }
          )

setMethod('[',
          signature='ProdPVPS',
          definition=function(x, i, j, ...){
            gef <- as(x, 'Gef')[i=i, j=j, ...] ##Gef method
            i <- indexI(gef)[1]
            j <- indexI(gef)[length(indexI(gef))]
            d1 <- indexD(x)
            d2 <- indexI(x)
            prodIw <- x@prodI[(d2 >= i & d2 <= j)]
            prodDw <- x@prodD[(d1 >= truncDay(i) & d1 <= truncDay(j))]
            prodDmw <- prodDw[, .(Eac = Eac/1000,
                                  Qd = Qd,
                                  Yf = Yf),
                              by = .(month(Dates), year(Dates))]
            if (x@type=='prom'){
                prodDmw[, DayOfMonth := DOM(prodDmw)]
                prodyw <- prodDmw[, lapply(.SD*DayOfMonth, sum, na.rm = TRUE),
                                  .SDcols = c('Eac', 'Qd', 'Yf'),
                                  by = .(Dates = year)]
                prodDmw[, DayOfMonth := NULL]
            } else {
                prodyw <- prodDw[, .(Eac = sum(Eac, na.rm = TRUE)/1000,
                                     Qd = sum(Qd, na.rm = TRUE),
                                     Yf = sum(Yf, na.rm = TRUE)),
                                 by = .(Dates = year)]
            }
            prodDmw[, Dates := paste(month.abb[month], year, sep = '. ')]
            prodDmw[, c('month', 'year') := NULL]
            setcolorder(prodDmw, c('Dates', names(prodDmw)[-length(prodDmw)]))
            result <- new('ProdPVPS',
                          gef,
                          prodD=prodDw,
                          prodDm=prodDmw,
                          prody=prodyw,
```

```
                                  prodI=prodIw,
                                  pump=x@pump,
                                  H=x@H,
                                  Pg=x@Pg,
                                  converter=x@converter,
                                  effSys=x@effSys
                                  )
                result
              }
              )
```

writeSolar

```
setGeneric('writeSolar', function(object, file,
                                  complete=FALSE, day=FALSE,
                                  timeScales=c('i', 'd', 'm', 'y'), sep=',',
                                  ...){
    standardGeneric('writeSolar')})

setMethod('writeSolar', signature=(object='Sol'),
          definition=function(object, file, complete=FALSE, day=FALSE,
                              timeScales=c('i', 'd', 'm', 'y'), sep=',', ...){
              name <- strsplit(file, '\\.')[[1]][1]
              ext <- strsplit(file, '\\.')[[1]][2]
              timeScales <- match.arg(timeScales, several.ok=TRUE)
              if ('i' %in% timeScales) {
                  zI <- as.data.tableI(object, complete=complete, day=day)
                  write.table(zI,
                              file=file, sep=sep, row.names = FALSE, ...)
              }
              if ('d' %in% timeScales) {
                  zD <- as.data.tableD(object, complete=complete, day = day)
                  write.table(zD,
                            file=paste(name, 'D', ext, sep='.'),
                            sep=sep, row.names = FALSE, ...)
              }
              if ('m' %in% timeScales) {
                  zM <- as.data.tableM(object, complete=complete, day = day)
                  write.table(zM,
                            file=paste(name, 'M', ext, sep='.'),
                            sep=sep, row.names = FALSE, ...)
              }
              if ('y' %in% timeScales) {
                  zY <- as.data.tableY(object, complete=complete, day = day)
                  write.table(zY,
                            file=paste(name, 'Y', ext, sep='.'),
                            sep=sep, row.names = FALSE, ...)
              }
          })
```

xyplot

```
#################################################################
## THEMES
```

```
##################################################################
xscale.solar <- function(...){ans <- xscale.components.default(...); ans$top=
    FALSE; ans}
yscale.solar <- function(...){ans <- yscale.components.default(...); ans$right=
    FALSE; ans}

solaR.theme <- function(pch=19, cex=0.7, region=rev(brewer.pal(9, 'YlOrRd')),
    ...) {
  theme <- custom.theme.2(pch=pch, cex=cex, region=region, ...)
  theme$strip.background$col='transparent'
  theme$strip.shingle$col='transparent'
  theme$strip.border$col='transparent'
  theme
}

solaR.theme.2 <- function(pch=19, cex=0.7, region=rev(brewer.pal(9, 'YlOrRd')),
    ...) {
  theme <- custom.theme.2(pch=pch, cex=cex, region=region, ...)
  theme$strip.background$col='lightgray'
  theme$strip.shingle$col='lightgray'
  theme
}

##################################################################
## XYPLOT
##################################################################
setGeneric('xyplot')

setMethod('xyplot',
          signature = c(x = 'data.frame', data = 'missing'),
          definition = function(x, data,
                                par.settings = solaR.theme.2,
                                xscale.components=xscale.solar,
                                yscale.components=yscale.solar,
                                scales = list(y = 'free'),
                                ...){
              N <- length(x)-1
              x0 <- x[, lapply(.SD, as.numeric), by = Dates]
              x0 <- melt(x0, id.vars = 'Dates')
              x0$variable <- factor(x0$variable,
                                    levels = rev(levels(factor(x0$variable))))
              xyplot(value ~ Dates | variable, x0,
                     par.settings = par.settings,
                     xscale.components = xscale.components,
                     yscale.components = yscale.components,
                     scales = scales,
                     type = 'l', layout = c(1,N),
                     ...)
          })

setMethod('xyplot',
          signature=c(x='formula', data='Meteo'),
          definition=function(x, data,
                              par.settings=solaR.theme,
                              xscale.components=xscale.solar,
                              yscale.components=yscale.solar,
```

```
                              ...){
            data0=getData(data)
            xyplot(x, data0,
                   par.settings = par.settings,
                   xscale.components = xscale.components,
                   yscale.components = yscale.components,
                   strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
          }
          )

setMethod('xyplot',
          signature=c(x='formula', data='Sol'),
          definition=function(x, data,
                              par.settings=solaR.theme,
                              xscale.components=xscale.solar,
                              yscale.components=yscale.solar,
                              ...){
              data0=as.data.tableI(data, complete=TRUE, day=TRUE)
              data0[, w := h2r(hms(Dates)-12)]
              xyplot(x, data0,
                     par.settings = par.settings,
                     xscale.components = xscale.components,
                     yscale.components = yscale.components,
                     strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
          }
          )

setMethod('xyplot',
          signature=c(x='formula', data='G0'),
          definition=function(x, data,
                              par.settings=solaR.theme,
                              xscale.components=xscale.solar,
                              yscale.components=yscale.solar,
                              ...){
            data0=as.data.tableI(data, complete=TRUE, day=TRUE)
            xyplot(x, data0,
                   par.settings = par.settings,
                   xscale.components = xscale.components,
                   yscale.components = yscale.components,
                   strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
          }
          )

setMethod('xyplot',
          signature=c(x='formula', data='Shade'),
          definition=function(x, data,
                              par.settings=solaR.theme,
                              xscale.components=xscale.solar,
                              yscale.components=yscale.solar,
                              ...){
            data0=as.data.table(data)
            xyplot(x, data0,
                   par.settings = par.settings,
                   xscale.components = xscale.components,
                   yscale.components = yscale.components,
                   strip = strip.custom(strip.levels=c(TRUE, TRUE)), ...)
```

```
          }
          )

setMethod('xyplot',
          signature=c(x='Meteo', data='missing'),
          definition=function(x, data,
                              ...){
              x0=getData(x)
              xyplot(x0,
                     scales=list(cex=0.6, rot=0, y='free'),
                     strip=FALSE, strip.left=TRUE,
                     par.strip.text=list(cex=0.6),
                     ylab = '',
                     ...)
          }
          )

setMethod('xyplot',
          signature=c(x='G0', data='missing'),
          definition=function(x, data, ...){
              x0 <- as.data.tableD(x, complete=FALSE)
              x0 <- melt(x0, id.vars = 'Dates')
              xyplot(value~Dates, x0, groups = variable,
                     par.settings=solaR.theme.2,
                     xscale.components=xscale.solar,
                     yscale.components=yscale.solar,
                     superpose=TRUE,
                     auto.key=list(space='right'),
                     ylab='Wh/m\u00b2',
                     type = 'l',
                     ...)
          }
          )

setMethod('xyplot',
          signature=c(x='ProdGCPV', data='missing'),
          definition=function(x, data, ...){
              x0 <- as.data.tableD(x, complete=FALSE)
              xyplot(x0,
                     strip = FALSE, strip.left = TRUE,
                     ylab = '', ...)
          }
          )

setMethod('xyplot',
          signature=c(x='ProdPVPS', data='missing'),
          definition=function(x, data, ...){
              x0 <- as.data.tableD(x, complete=FALSE)
              xyplot(x0,
                     strip = FALSE, strip.left = TRUE,
                     ylab = '', ...)
          }
          )
```

A.5. Conjunto de datos

aguiar

```r
data(MTM)
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
```

```r
Ktmtm
```

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

```r
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

```r
data(SIAR)
head(est_SIAR)
```

```
         Estacion Codigo     Longitud  Latitud Altitud Fecha_Instalacion Fecha_Baja
           <char> <char>        <num>    <num>   <int>            <Date>     <Date>
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

```r
data(helios)
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

```
data(prodEx)
head(prodEx)
```

```
        Dates        1        2        3        4        5        6        7        8        9
       <Date>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <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       11       12       13       14       15       16       17       18       19       20
      <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <num>    <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       22
      <num>    <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

```
data(pumpCoef)
head(pumpCoef)
```

```
      Qn stages  Qmax   Pmn          a        b        c     g     h     i       j      k      l
   <int>  <int> <num> <int>      <num>    <num>    <num> <num> <num> <num>   <num>  <num>  <num>
1:     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, autodocumentado 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.

[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.

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

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

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

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

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

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

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

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

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

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

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

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

[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.