Introduction to solaR

Oscar Perpiñán Lamigueiro

23 March 2011

1 Introduction

The solaR package includes a set of functions which calculate the solar radiation incident on a photovoltaic generator and simulate the performance of several applications of the photovoltaic energy [8]. This package performs the whole calculation from both *daily* and *intra-daily* global horizontal irradiation to the final productivity of grid connected PV systems and water pumping PV systems. Besides, the package includes several visualization methods based on the lattice and latticeExtra packages , and tools for the statistical analysis of the performance of a large PV plant composed of several systems.

The package is constructed with S4 classes and methods. The time series are constructed with the zoo package [9].

2 Solar Geometry

The apparent movement of the Sun is defined with some equations included in the functions fSolD and fSolI. fSolD computes the daily apparent movement of the Sun from the Earth. This movement is mainly described (for the simulation of photovoltaic systems) by the declination angle, the sunset angle and the daily extra-atmospheric irradiation. On the other hand, fSolI computes the angles which describe the intra-daily apparent movement of the Sun from the Earth.

The next example shows these calculations for a certain day:

```
> BTd = fBTd(mode = "serie")
> lat = 37.2
> SolD <- fSolD(lat, BTd[100])
> SolI <- fSolI(SolD, sample = "hour", keep.night = FALSE)
> head(SolI)

                          w aman cosThzS     AlS     AzS    Bo0      rd
2011-04-10 06:00:00 -1.5708    1 0.07927 0.07935 -1.6758  107.8 0.01130
2011-04-10 07:00:00 -1.3090    1 0.28365 0.28760 -1.5179  385.8 0.04044
2011-04-10 08:00:00 -1.0472    1 0.47410 0.49394 -1.3472  644.9 0.06759
2011-04-10 09:00:00 -0.7854    1 0.63764 0.69143 -1.1433  867.3 0.09091
2011-04-10 10:00:00 -0.5236    1 0.76313 0.86814 -0.8742 1038.0 0.10880
2011-04-10 11:00:00 -0.2618    1 0.84202 1.00101 -0.4957 1145.3 0.12005
                          rg
2011-04-10 06:00:00 0.007935
2011-04-10 07:00:00 0.032395
2011-04-10 08:00:00 0.060379
2011-04-10 09:00:00 0.088405
2011-04-10 10:00:00 0.112414
2011-04-10 11:00:00 0.128619
```

and for a set of days:

```
> SolD <- fSolD(lat, BTd[c(10, 50, 100)])
> print(SolD)

              decl    eo     ws BoOd       EoT
2011-01-10 -0.3847 1.033 -1.258 4497 -0.035464
2011-02-19 -0.2082 1.022 -1.410 6327 -0.059933
2011-04-10  0.1315 0.995 -1.671 9541 -0.004637
attr(,"lat")
[1] 37.2
```

With the function fBTd it is possible to get time bases with different structures. Thus, the calculations for the so called "average days" need the next piece of code, with the result displayed in the figure 1.

```
> lat = 37.2
> SolD <- fSolD(lat, BTd = fBTd(mode = "prom"))
> SolI <- fSolI(SolD, sample = "10 min", keep.night = FALSE)
```

```
> mon = month.abb
> p <- xyplot(r2d(AlS) ~ r2d(AzS), groups = month, data = SolI,
+     type = "l", col = "black", xlab = expression(psi[s]), ylab = expression(gamma[s]))
> plab = p + glayer(panel.text(0, y[x == 0], mon[group.value],
+     pos = 4, cex = 0.8))
> print(plab)
```

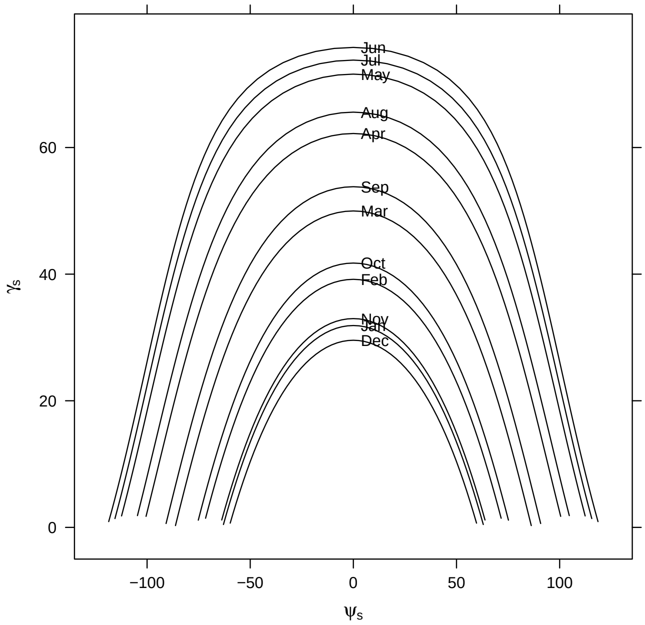

Figure 1: Azimuth and height solar angles during the "average days".

These calculations can also be carried out for the whole year (figure 2).

```
> BTd = fBTd(mode = "serie")
> solD <- fSolD(lat, BTd)
> summary(solD)

     Index                decl                 eo               ws
 Min.   :2011-01-01   Min.   :-4.09e-01   Min.   :0.967   Min.   :-1.91
 1st Qu.:2011-04-02   1st Qu.:-2.89e-01   1st Qu.:0.977   1st Qu.:-1.80
 Median :2011-07-02   Median : 2.63e-16   Median :1.000   Median :-1.57
 Mean   :2011-07-02   Mean   : 9.31e-18   Mean   :1.000   Mean   :-1.57
 3rd Qu.:2011-10-01   3rd Qu.: 2.89e-01   3rd Qu.:1.023   3rd Qu.:-1.34
 Max.   :2011-12-31   Max.   : 4.09e-01   Max.   :1.033   Max.   :-1.24
      Bo0d             EoT
 Min.   : 4235   Min.   :-6.18e-02
 1st Qu.: 5472   1st Qu.:-2.59e-02
 Median : 8302   Median :-2.48e-03
 Mean   : 8116   Mean   : 1.24e-05
 3rd Qu.:10742   3rd Qu.: 2.16e-02
 Max.   :11607   Max.   : 7.09e-02
```

These two functions are included in a function, `calcSol`. This function constructs an object of class `Sol` containing in its slots the `zoo` objects created by `fSolD` and `fSolI`. This class owns methods for getting and displaying information (for example, `as.zooD, as.zooI, xyplot`).

3 Solar Radiation

Values of global horizontal irradiation are commonly available either as monthly averages of daily values or as a time series of daily during one or several years. The analysis of the performance of a PV system starts from the transformation of the global horizontal irradiation to global, diffuse and direct horizontal irradiance and irradiation, and then irradiance and irradiation on the generator surface.

3.1 Irradiation and irradiance on the horizontal plane

The function `fCompD` extracts the diffuse and direct components from the daily global irradiation on a horizontal surface by means of regressions between the clearness index and the diffuse fraction parameters. This function need the

```
> p <- xyplot(solD$decl)
> print(p)
```

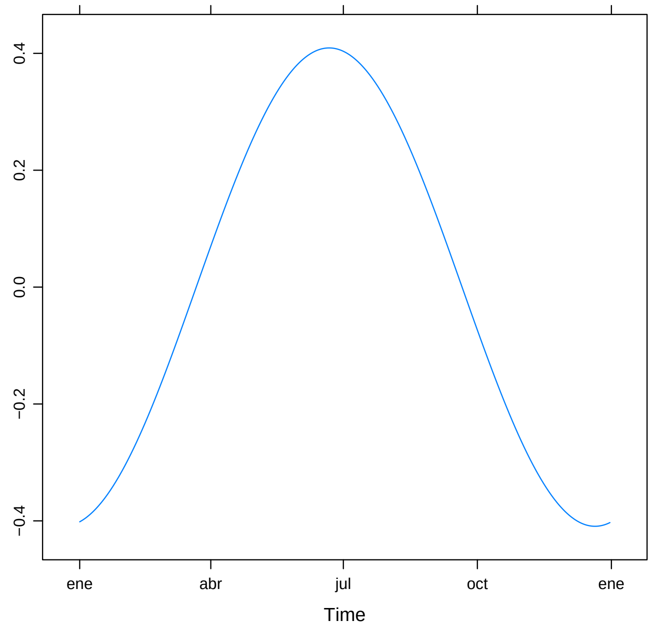

Figure 2: Declination throughout the year

results from `fSolD`, a set of values of global horizontal irradiation ($^{Wh}/_{m^2}$), and the correlation between the clearness index and the diffuse fraction. The current version of `solaR` includes several correlations (type `help(corrFdKt)` for details). Besides, the user may define a particular correlation through the argument `f`. Once again for a certain day:

```
> BTd = fBTd(mode = "serie")
> SolD <- fSolD(lat, BTd[100])
> SolI <- fSolI(SolD, sample = "hour")
> G0d = zoo(5000, index(SolD))
> fCompD(SolD, G0d, corr = "Page")

               Fd    Ktd  G0d  D0d  B0d
2011-04-10 0.4078 0.5241 5000 2039 2961

> fCompD(SolD, G0d, corr = "CPR")

               Fd    Ktd  G0d  D0d  B0d
2011-04-10 0.5582 0.5241 5000 2791 2209
```

and for the "average days":

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

               Fd    Ktd  G0d    D0d  B0d
2011-01-17 0.3354 0.5882 2766  927.6 1838
2011-02-14 0.3452 0.5794 3491 1205.2 2286
2011-03-15 0.3573 0.5687 4494 1605.9 2888
2011-04-15 0.3195 0.6022 5912 1888.9 4023
2011-05-15 0.2871 0.6309 6989 2006.5 4982
2011-06-10 0.2437 0.6693 7742 1886.8 5855
2011-07-18 0.2070 0.7018 7919 1639.0 6280
2011-08-18 0.2209 0.6894 7027 1552.4 5475
2011-09-18 0.2804 0.6368 5369 1505.6 3863
2011-10-19 0.3728 0.5550 3562 1328.1 2234
2011-11-18 0.3475 0.5775 2814  977.8 1836
2011-12-13 0.4233 0.5104 2179  922.3 1257
```

Let's use `corr='user'` to define a function with the correlation of Collares Pereira and Rabl [2]. Obviously, we shall obtain the same result as with `corr='CPR'`.

```
> fKTd = function(x) {
+     (0.99 * (x <= 0.17)) + (x > 0.17) * (1.188 - 2.272 * x +
+         9.473 * x^2 - 21.856 * x^3 + 14.648 * x^4)
+ }
> fCompD(SolD, G0d, corr = "user", f = fKTd)

               Fd    Ktd  G0d  D0d  B0d
2011-04-10 0.5582 0.5241 5000 2791 2209
```

The daily profile of irradiance is obtained with the function `fCompI`. This function needs the information provided by `fCompD` and `fSolI` or `calcSol`. For example, the profiles for the "average days" are obtained with the next code (fig. 3).

```
> lat = 37.2
> sol <- calcSol(lat, fBTd(mode = "prom"), sample = "hour", keep.night = FALSE)
> G0dm = c(2.766, 3.491, 4.494, 5.912, 6.989, 7.742, 7.919, 7.027,
+     5.369, 3.562, 2.814, 2.179) * 1000
> Ta = c(10, 14.1, 15.6, 17.2, 19.3, 21.2, 28.4, 29.9, 24.3, 18.2,
+     17.2, 15.2)
> BD <- readG0dm(G0dm = G0dm, Ta = Ta, lat = 37.2)
> compD <- fCompD(sol, BD, corr = "Page")
> compI <- fCompI(sol, compD)
```

3.1.1 Meteorological data

There are several functions to construct a `Meteo` object with radiation and temperature data. For daily data, if it is stored in a local file or a `data.frame`, the functions `readBD` and `df2Meteo` are recommended, while `readG0dm` is indicated when only 12 monthly means are available. For intradaily data the correspondent functions are `readBDi` and `dfI2Meteo`. Besides, `zoo2Meteo` can construct a `Meteo` object from a `zoo` object both for daily and intradaily data.

For example, the `helios` dataset included in the package, obtained from http://helios.ies-def.upm.es, can be converted to a `Meteo` object with the next code:

```
> data(helios)
> names(helios) = c("date", "G0", "TempMax", "TempMin")
> bd = df2Meteo(helios, dates.col = "date", lat = 41, source = "helios-IES",
+     format = "%Y/%m/%d")
> summary(getData(bd))

     Index                          G0            TempMax          TempMin
 Min.   :2009-01-01 00:00:00   Min.   :  326   Min.   : 1.41   Min.   :-37.50
 1st Qu.:2009-04-08 12:00:00   1st Qu.: 2523   1st Qu.:14.41   1st Qu.:  1.95
 Median :2009-07-07 00:00:00   Median : 4746   Median :23.16   Median :  7.91
 Mean   :2009-07-04 21:29:54   Mean   : 4812   Mean   :22.59   Mean   :  5.32
 3rd Qu.:2009-10-03 12:00:00   3rd Qu.: 7140   3rd Qu.:31.06   3rd Qu.: 15.11
 Max.   :2009-12-31 00:00:00   Max.   :11254   Max.   :38.04   Max.   : 24.80
```

On the other hand, the function `readMAPA` is able to download the meteorological data available at www.mapa.es/siar. This webpage provides daily measurements from a set of agroclimatic stations located in Spain. This function needs the code of the station and its province, and the start and end date. The codes of stations and provinces are stored at the dataset `RedEstaciones`. For example, there are several stations in Madrid:

```
> data(RedEstaciones)
> Madrid <- subset(RedEstaciones, NomProv == "Madrid")
> print(Madrid)

     Provincia Estacion NomProv                     NomEst
P209        28        1  Madrid Center:_Finca_experimental
P210        28        2  Madrid                    Arganda
P211        28        3  Madrid                   Aranjuez
P212        28        4  Madrid        Fuentiduena_de_Tajo
P213        28        5  Madrid      San_Martin_de_la_Vega
P214        28        6  Madrid                   Chinchon
P215        28      102  Madrid              Villa_del_Prado
```

`readMAPA` constructs an object of class `Meteo`. The data is obtained with the method `getData`. If only the irradiation series is needed, the method `getG0` is recommended.

For example, let's obtain the 2009 data from the station at Aranjuez (fig. 4). It is important to note that the radiation measurements available at the webpage are in $^{MJ}/_{m^2}$, but `readMAPA` converts the values to $^{Wh}/_{m^2}$:

```
> Aranjuez <- readMAPA(28, 3, "01/01/2009", "31/12/2009")

Downloading data from www.mapa.es/siar...
```

This database includes information of maximum and minimum values of temperature. The function `fTemp` calculates a profile of the ambient temperature with this information following the method proposed in [3]. The evolution of this synthetic temperature during March is displayed in the figure 5.

```
> p <- xyplot(G0 + B0 + D0 ~ w | month, data = compI, type = "l",
+     auto.key = list(space = "right"))
> print(p)
```

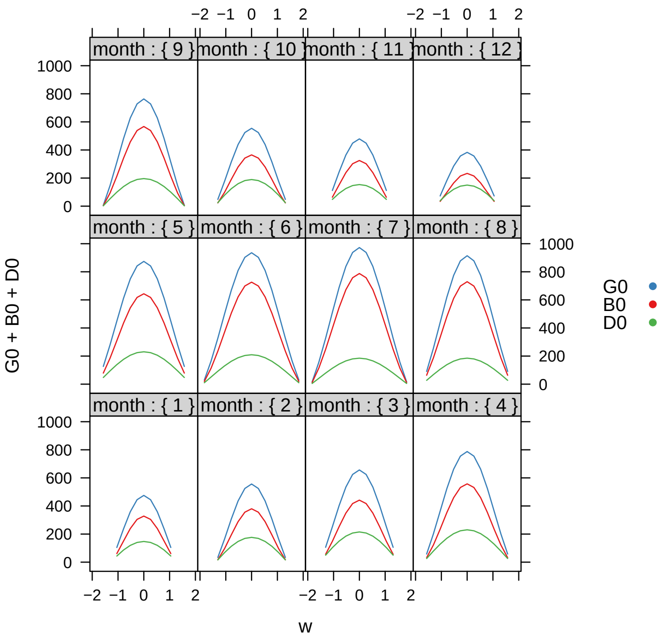

Figure 3: Global, diffuse, and direct irradiance during the "average days".

```
> p = xyplot(G0 ~ TempMedia | month, data = Aranjuez, type = c("p",
+     "r"))
> print(p)
```

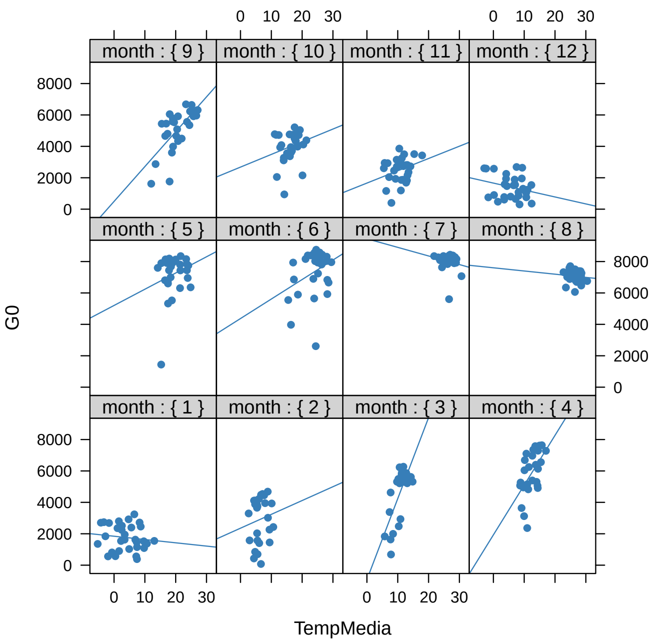

Figure 4: Daily irradiation and mean temperature in the station of Aranjuez.

```
> lat = 41
> sol = calcSol(lat, BTd = indexD(Aranjuez), sample = "hour")
> Temp <- fTemp(sol, Aranjuez)
```

3.1.2 The function `calcG0`

The previous steps are included in the function `calcG0`. For example, with the next code, the components of horizontal irradiation and irradiance are obtained from the measurements of the meteorological station of Aranjuez (figure 6).

```
> g0 <- calcG0(lat = 37.2, modeRad = "mapa", mapa = list(prov = 28,
+     est = 3, start = "01/01/2009", end = "31/12/2009"))

Downloading data from www.mapa.es/siar...

> print(g0)

Object of class  G0

Source of meteorological information: mapa-Est: 3 Prov: 28

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

Monthly averages:
           G0d    D0d    B0d
ene 2009 1.764 1.1461 0.6176
feb 2009 2.916 1.2915 1.6244
mar 2009 4.725 1.5877 3.1371
abr 2009 5.819 2.2890 3.5303
may 2009 7.198 2.2475 4.9510
jun 2009 7.354 2.4525 4.9013
jul 2009 8.002 2.0457 5.9566
ago 2009 7.061 1.9446 5.1160
sep 2009 5.168 1.8897 3.2782
oct 2009 3.993 1.4684 2.5244
nov 2009 2.510 1.3175 1.1920
dic 2009 1.397 0.9773 0.4192

Yearly values:
      G0d   D0d  B0d
2009 1730 614.7 1115
```

`solaR` accepts intradaily irradiation data sources. For example, the Measurement and Instrumentation Data Center of the NREL (NREL-MIDC) provides meteorological data from a variety of stations. We will try the *La Ola - Lanai* station at Hawaii (`http://www.nrel.gov/midc/la_ola_lanai/`).

```
> file = "http://www.nrel.gov/midc/apps/plot.pl?site=LANAI&start=20090722&edy=19&emo=11&eyr=2010&zenloc=19&year=2010&month=11&day=1&endyear=2010&
> dat <- read.table(file, header = TRUE, sep = ",")
> lat = 20.77
> lon = -156.9339
```

First, we have to change the names of the columns and calculate the horizontal direct irradiation, since only the normal direct irradiation is included in the file.

```
> names(dat) <- c("date", "hour", "G0", "B", "D0", "Ta")
> dat$B0 <- dat$G0 - dat$D0
```

The datalogger program runs using Greenwich Mean Time (GMT), and data is converted to Hawaiian Standard Time (HST) after data collection. With `local2Solar` we can calculate the Mean Solar Time of the index.

```
> idxLocal <- with(dat, as.POSIXct(paste(date, hour), format = "%m/%d/%Y %H:%M",
+     tz = "HST"))
> idx <- local2Solar(idxLocal, lon = lon)
```

Therefore, the object `Meteo` is obtained with (figure 7):

```
> z <- zoo(dat[, c("G0", "D0", "B0", "Ta")], idx)
> NRELMeteo <- zoo2Meteo(z, lat = lat)
```

With this data, a `G0` object can be calculated. First, the direct and diffuse components of the data are used (`corr='none'`):

```
> g0NREL <- calcG0(lat = lat, modeRad = "bdI", bdI = NRELMeteo,
+     corr = "none")
```

If these components were not available, a fd-kt hourly correlation is needed (figure 8). For example:

```
> g0BRL <- calcG0(lat = lat, modeRad = "bdI", bdI = NRELMeteo,
+     corr = "BRL")
```

```
> wTemp = window(Temp, start = as.POSIXct("2009-03-01"), end = as.POSIXct("2009-03-31"))
> p = xyplot(wTemp, col = "black", ylab = "T") + layer_(panel.xblocks(x,
+     DoY, col = c("lightgray", "white")))
> print(p)
```

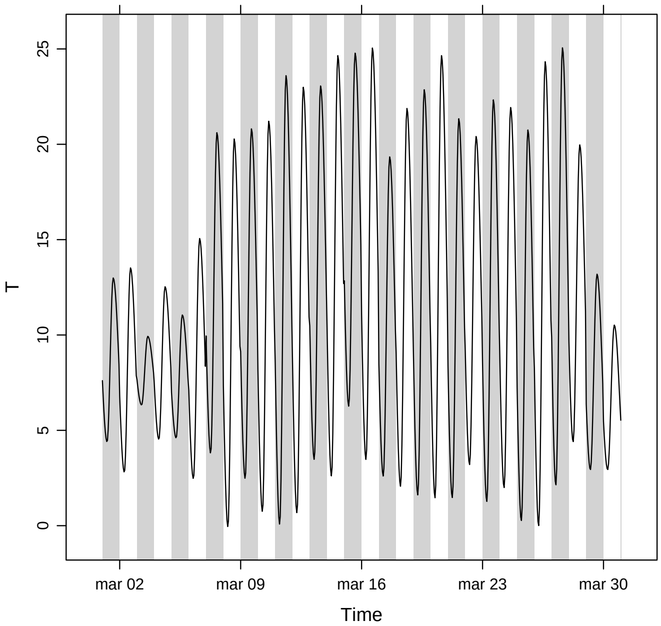

Figure 5: Evolution of the ambiente temperature during March 2009 in Aranjuez.

```
> p = xyplot(g0)
> print(p)
```

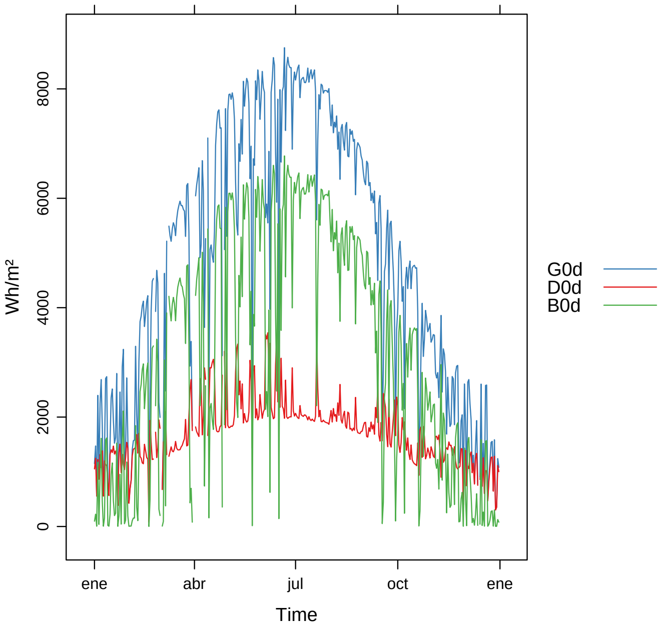

Figure 6: Components of horizontal irradiation calculated with `calcG0`.

```
> p <- xyplot(NRELMeteo)
> print(p)
```

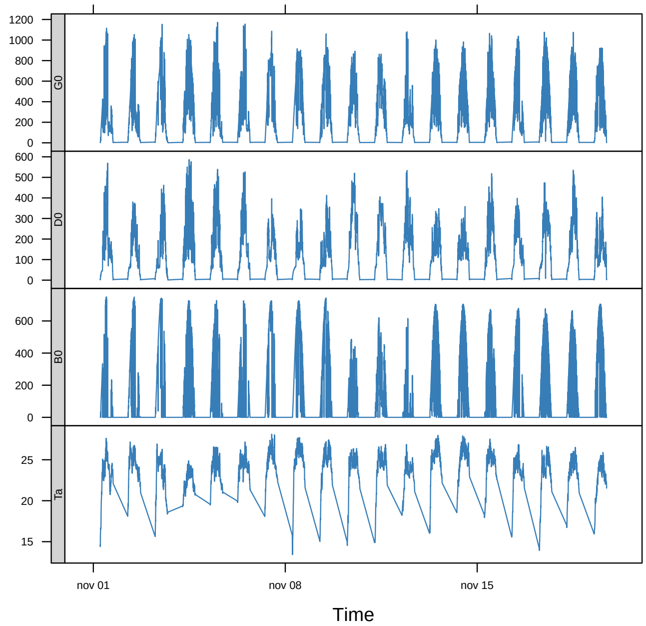

Figure 7: 1-min irradiation data from NREL-MIDC

```
> p <- xyplot(fd ~ kt, data = g0BRL, pch = 19, alpha = 0.3, cex = 0.5)
> print(p)
```

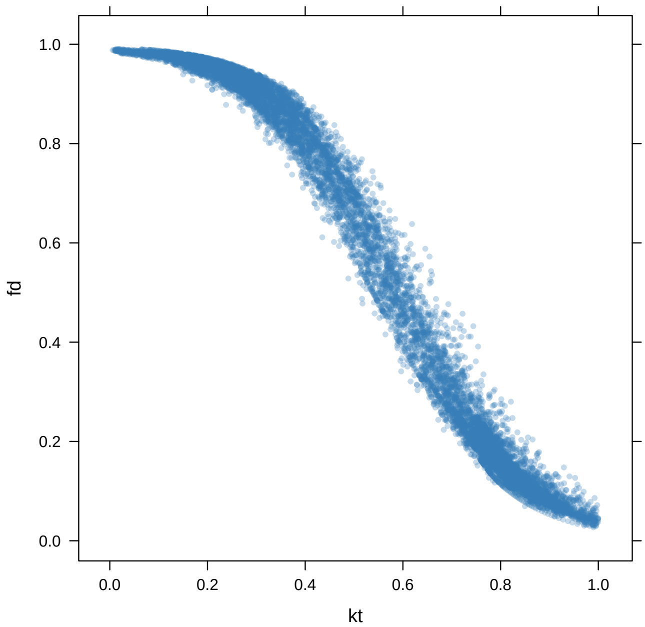

Figure 8: Diffuse fraction and clearness index correlation following the BRL model, with data from NREL-MIDC

3.2 Irradiation and irradiance on the generator plane

The solar irradiance incident on an inclined surface can be calculated from the direct and diffuse irradiance on a horizontal surface, and from the evolution of the angles of the Sun and the surface. The transformation of the direct radiation is straightforward since only geometric considerations are needed. However, the treatment of the diffuse irradiance is more complex since it involves the modelling of the atmosphere. There are several models for the estimation of diffuse irradiance on an inclined surface. The one which combines simplicity and acceptable results is the proposal of Hay and McKay. This model divides the diffuse component in isotropic and anisotropic whose values depends on a anisotropy index. On the other hand, the effective irradiance, the fraction of the incident irradiance that reaches the cells inside a PV module, is calculated with the losses due to the angle of incidence and dirtiness. This behaviour can be simulated with a model proposed by Martin and Ruiz requiring information about the angles of the surface and the level of dirtiness [4].

The orientation, azimuth and incidence angle are calculated from the results of `fSolI` or `calcSol` with the functions `fTheta` and `fInclin`. These functions can calculate the movement and irradiance for fixed systems, and two-axis and horizontal N-S trackers. Besides, the movement of a horizontal NS tracker due to the backtracking strategy [5] can be calculated with information about the tracker and the distance between the trackers included in the system.

Both functions are integrated in `calcGef`, which construct an object of class `Gef`. Once again, this class owns methods for obtaining and displaying information.

For example, with the previous results, we can calculate the irradiance and irradiation on a fixed surface. The figure 9 shows the relation between the effective and incident irradiance versus the cosine of the angle of incidence for this system.

```
> gef <- calcGef(lat = 37.2, modeRad = "prev", prev = g0, beta = 30)
> print(gef)

Object of class  Gef

Source of meteorological information: mapa-Est: 3 Prov: 28

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

Monthly averages:
            Bod   Bnd     Gd     Dd     Bd   Gefd   Defd   Befd
ene 2009  8.720 1.539 1.4310 0.3001 1.1073 1.3643 0.2874 1.0600
feb 2009  9.801 3.425 2.9691 0.5219 2.4096 2.8140 0.4964 2.2907
mar 2009 10.289 5.156 4.3809 0.6827 3.6411 4.1610 0.6507 3.4693
abr 2009 10.428 5.113 4.2134 0.7136 3.4297 3.9956 0.6799 3.2654
may 2009 10.225 7.615 5.7124 0.9206 4.6953 5.3871 0.8719 4.4461
jun 2009 10.025 7.529 5.3273 0.8591 4.3697 5.0087 0.8116 4.1265
jul 2009 10.080 9.328 6.5313 0.9719 5.4522 6.1470 0.9185 5.1518
ago 2009 10.281 7.991 6.2995 0.9809 5.2240 5.9580 0.9311 4.9591
sep 2009 10.270 5.682 4.7969 0.8227 3.9050 4.5570 0.7846 3.7228
oct 2009  9.894 5.210 4.5310 0.7836 3.6939 4.2974 0.7456 3.5135
nov 2009  8.977 2.916 2.6178 0.5253 2.0589 2.4896 0.5015 1.9641
dic 2009  8.484 1.064 0.9878 0.2035 0.7662 0.9405 0.1948 0.7328

Yearly values:
      Bod  Bnd   Gd    Dd   Bd Gefd  Defd Befd
2009 3573 1908 1518 252.4 1242 1436 239.8 1180
-----------------
Mode of tracking:  fixed
    Inclination:  30
    Orientation:  0
```

The next lines of code calculate the movement of a N-S horizontal axis tracker with *backtracking* (`modeShd='bt'`) and whose inclination angle is limited to 60° (`betaLim=60`). The evolution of the inclination angle is displayed in the figure 10. The meaning of the `distances` and `struct` arguments will be detailed in the 4.2 section.

```
> structHoriz = list(L = 4.83)
> distHoriz = data.frame(Lew = structHoriz$L * 4, H = 0)
> gefBT = calcGef(lat = 37.2, prom = prom, sample = "10 min", modeTrk = "horiz",
+     modeShd = "bt", betaLim = 60, distances = distHoriz, struct = structHoriz)
```

4 Productivity of a Grid Connected PV System

From the previous irradiance calculations, the function `fProd` simulates the performance of a Grid Connected PV (GCPV) system paying attention to some parameters of the system (characteristics of the PV module and the inverter, the electrical arrangement of the PV generator, and the losses of the system).

For example, the electrical power, voltage and current of a certain PV system is calculated below.

```
> p <- xyplot(Gef/G ~ cosTheta | month, data = gef, type = c("p",
+     "smooth"), cex = 0.4, alpha = 0.5)
> print(p)
```

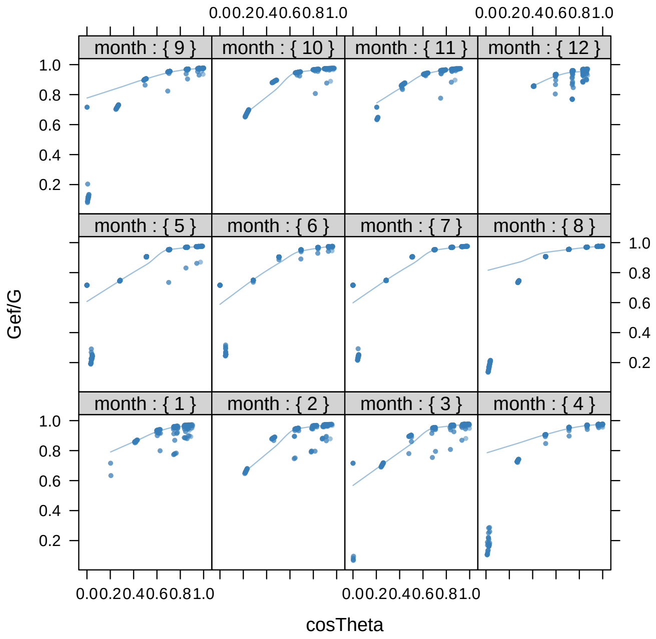

Figure 9: Relation between the effective and incident irradiance versus the cosine of the angle of incidence for a fixed system.

```
> p <- xyplot(r2d(Beta) ~ r2d(w), data = gefBT, type = "l", xlab = expression(omega),
+     ylab = expression(beta))
> print(p)
```

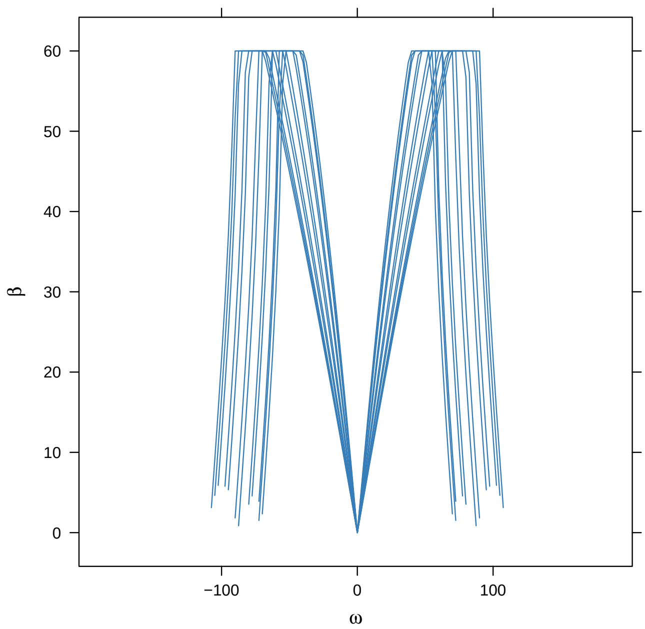

Figure 10: Evolution of the angle of inclination of a NS horizontal axis tracker with *backtracking* and limitation of angle.

```
> inclin = data.frame(Gef = c(200, 400, 600, 800, 1000), Ta = 25)
> fProd(inclin)

   Gef Ta    Tc   Voc   Isc  Vmpp   Impp   Vdc    Idc   Pac   Pdc   EffI
1  200 25 31.75 673.3 10.34 533.1  9.586 533.1  9.586  4212  4737 0.9164
2  400 25 38.50 655.4 20.68 516.3 19.090 516.3 19.090  8275  9137 0.9334
3  600 25 45.25 637.5 31.02 499.6 28.506 499.6 28.506 11972 13202 0.9346
4  800 25 52.00 619.7 41.36 483.0 37.824 483.0 37.824 15323 16936 0.9325
5 1000 25 58.75 601.8 51.70 466.5 47.037 466.5 47.037 18342 20342 0.9293
```

First, `fProd` computes the Maximum Power Point (MPP) of the generator (`Vmpp` and `Impp`) at the irradiance and ambient temperature conditions contained in `Inclin`. Next, it checks that this point is inside the MPP window of the inverter, as defined by `inverter$Vmin` and `inverter$Vmax`. If the MPP value is outside this range, the function asigns the limit value to the voltage, and calculates the correspondent current value with a warning.

Anyway, the inverter input voltage and current are `Vdc` e `Idc`. With the next piece of code, the `Vdc` value is set to `Vmin` (the minimum value of the MPP window of the inverter), 420 V, since `Vmpp` is below this value.

```
> inclin = data.frame(Gef = 800, Ta = 30)
> gen1 = list(Nms = 10, Nmp = 11)
> inv1 = list(Ki = c(0.01, 0.025, 0.05), Pinv = 25000, Vmin = 420,
+     Vmax = 750, Gumb = 20)
> prod = fProd(inclin, generator = gen1, inverter = inv1)
> print(prod)

  Gef Ta Tc   Voc   Isc  Vmpp  Impp Vdc   Idc   Pac   Pdc   EffI
1 800 30 57 505.3 41.36 392.3 37.68 420 33.83 11943 13169 0.9346
```

For this configuration, the losses due to the voltage limitation are:

```
> with(prod, Vdc * Idc/(Vmpp * Impp))

[1] 0.961
```

The function `prodGCPV` integrates the calculation procedure of irradiation, irradiance and simulation of the GCPV system. It constructs an object of class `ProdGCPV`.

The next code computes the productivity of the previous GCPV system working as fixed, NS horizontal axis tracking and two-axis tracking systems. The parameters of the generator, module, inverter and rest of the system are those by default in `prodGCPV`. The comparative of the intradaily power time series is shown at the figure 11. Later on, the `compare` and `compareLosses` methods will be shown. They are useful for comparisons of *yearly* values.

```
> ProdFixed <- prodGCPV(lat = lat, prom = prom, keep.night = FALSE)
> Prod2x <- prodGCPV(lat = lat, prom = prom, modeTrk = "two", keep.night = FALSE)
> ProdHoriz <- prodGCPV(lat = lat, prom = prom, modeTrk = "horiz",
+     keep.night = FALSE)
```

4.1 Using `mergesolaR`

The `mergesolaR` method is designed to merge *daily* time series of several `solaR` objects.

For example, we can obtain the daily irradiation of the whole set of meteorological stations of Madrid (Spain) and use this information to calculate the productivity of a grid connected PV system. It is possible to complete this process with the `lapply` function. Therefore we obtain a list of `ProdGCPV` objects:

```
> EstMadrid <- subset(RedEstaciones, NomProv == "Madrid")
> nEstMadrid <- nrow(EstMadrid)
> namesMadrid <- EstMadrid$NomEst
> prodMadrid <- lapply(1:nEstMadrid, function(x) {
+     try(prodGCPV(lat = 41, modeRad = "mapa", mapa = list(prov = 28,
+         est = x, start = "01/01/2009", end = "31/12/2010")))
+ })

Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...
Downloading data from www.mapa.es/siar...

> names(prodMadrid) <- namesMadrid
> okMadrid <- lapply(prodMadrid, class) != "try-error"
> prodMadrid <- prodMadrid[okMadrid]
```

In order to prevent from the erroneous behaviour of some stations, the code includes the use of `try`. Now it's time for `mergesolaR`. Since we have a list of objects, `do.call` can solve the problem:

```
> ComparePac <- CBIND(two = as.zooI(Prod2x)$Pac, horiz = as.zooI(ProdHoriz)$Pac,
+     fixed = as.zooI(ProdFixed)$Pac)
> AngSol = as.zooI(as(ProdFixed, "Sol"))
> ComparePac = CBIND(AngSol, ComparePac)
> mon = month(index(ComparePac))
> p = xyplot(two + horiz + fixed ~ AzS | mon, data = ComparePac,
+     type = "l", auto.key = list(space = "right", lines = TRUE,
+         points = FALSE), ylab = "Pac")
> print(p)
```

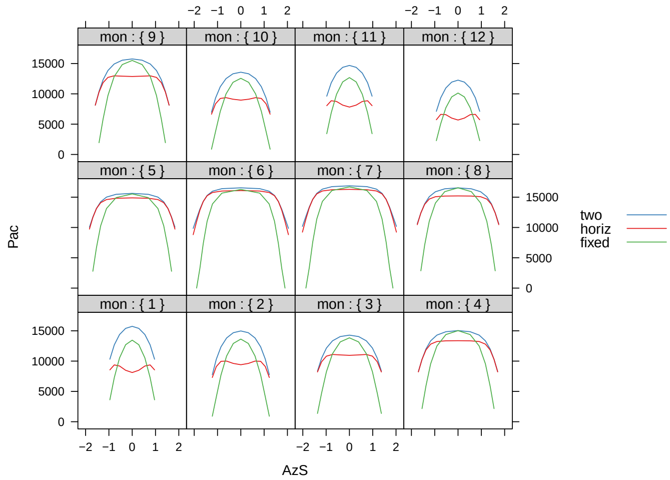

Figure 11: Comparative of intradaily power between tracker strategies.

```
> YfMadrid <- do.call(mergesolaR, prodMadrid)
> summary(YfMadrid)

     Index                        Center:_Finca_experimental     Arganda
 Min.   :2009-01-01 00:00:00   Min.   :0.00                Min.   :0.00
 1st Qu.:2009-07-02 06:00:00   1st Qu.:1.38                1st Qu.:1.63
 Median :2009-12-31 12:00:00   Median :3.88                Median :4.12
 Mean   :2009-12-31 12:00:00   Mean   :3.15                Mean   :3.29
 3rd Qu.:2010-07-01 18:00:00   3rd Qu.:4.72                3rd Qu.:4.77
 Max.   :2010-12-31 00:00:00   Max.   :5.63                Max.   :5.57
                               NA's   :3.00                NA's   :3.00
    Aranjuez     Fuentiduena_de_Tajo San_Martin_de_la_Vega    Chinchon
 Min.   :0.00   Min.   :0.00        Min.   :0.00          Min.   :0.00
 1st Qu.:1.43   1st Qu.:1.65        1st Qu.:1.54          1st Qu.:1.66
 Median :4.04   Median :4.17        Median :3.95          Median :4.24
 Mean   :3.17   Mean   :3.32        Mean   :3.20          Mean   :3.35
 3rd Qu.:4.66   3rd Qu.:4.84        3rd Qu.:4.76          3rd Qu.:4.88
 Max.   :5.64   Max.   :5.57        Max.   :5.71          Max.   :5.62
 NA's   :3.00   NA's   :3.00        NA's   :3.00          NA's   :3.00
```

The `mergesolaR` for a set of `ProdGCPV` objects merges the daily time series of the `Yf` variable of each object. The result is a multivariate `zoo` object where each column is the daily productivity with the radiation data of each meteorological station. It can be displayed (for example) with the `horizonplot` function (figure 12). This result will be revisited with the `Target Diagram` tool (figure 27).

4.2 Shadows

The shadows on PV generators alter the performance of the PV generators and reduce their productivity [6]. This package includes functions for the estimation of mutual shadows between generators from a same system. `fSombra2X`, `fSombraHoriz`, `fSombraEst`, calculate the shadows in two-axis, horizontal axis and fixed systems, respectively. The function `fSombra6` is indicated for groups of 6 two-axis trackers. Finally, `fSombra` is a wrapper to the previous functions.

For example, the shadows factor of a tracker surrounded by five trackers is calculated in the next code box. The dimensions of the tracker structure and the configuration (rows and columns) of the plant are defined by `struct`, while the distances between the trackers are defined by `distances`. The figure 13 shows the evolution of the shadows factor during the day (X axis) and year (Y axis).

Since the `data.frame distances` does only have one row, the function `fSombra6` builds a symmetric grid around the point (0,0,0), which is the affected tracker. This grid can also be constructed with:

```
> print(horizonplot(YfMadrid - rowMeans(YfMadrid), origin = 0,
+     scales = list(y = list(relation = "same")), colorkey = TRUE))
```

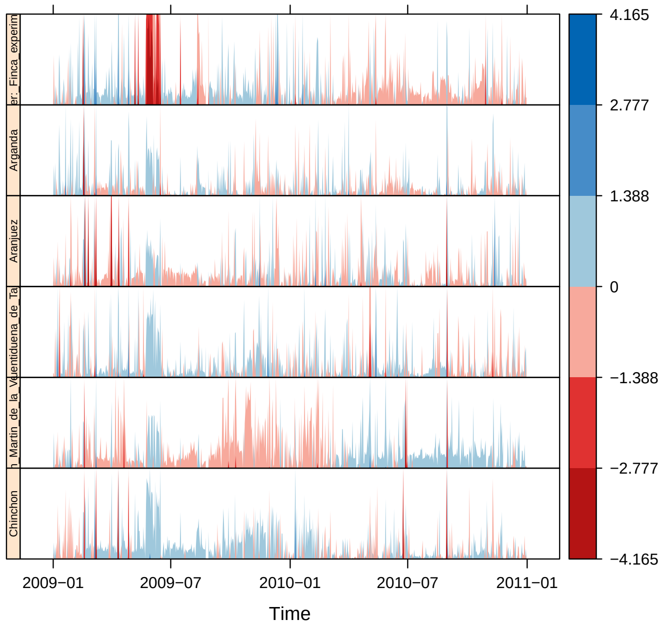

Figure 12: `Horizonplot` of the result of a `mergesolaR` call. Previously, the row mean is substracted from each column in order to show the deviation of each meteorological station from the daily mean of the set.

```
> p <- levelplot(FS ~ w * day, data = Angles, par.settings = custom.theme(region = brewer.pal("YlOrBr",
+     n = 9)))
> print(p)
```

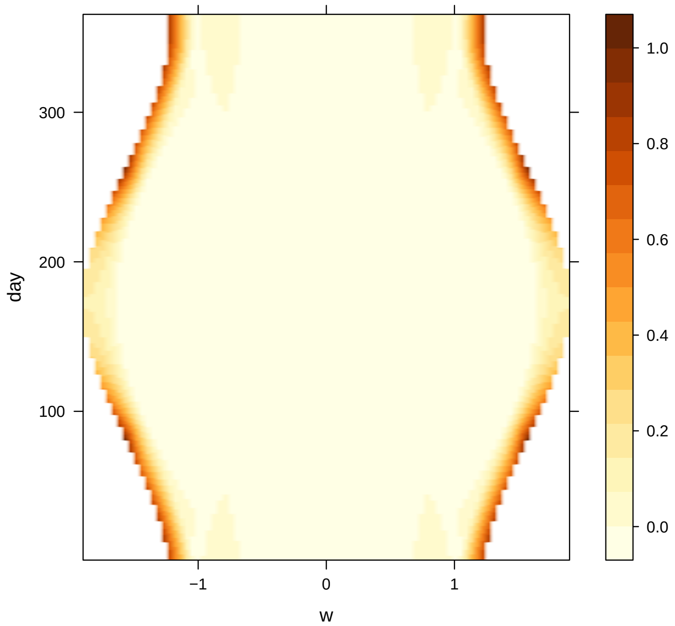

Figure 13: Shadows in a PV plant with two-axis trackers.

```
> distances = data.frame(Lew = c(-40, 0, 40, -40, 40), Lns = c(30,
+     30, 30, 0, 0), H = 0)
> ShdFactor2 <- fSombra6(Angles, distances, struct, prom = FALSE)
> identical(coredata(ShdFactor), coredata(ShdFactor2))

[1] TRUE
```

Besides, `distances` can define a irregular grid around the affected tracker. Since this tracker is situated at (0,0,0), `distances` must have five rows. When `prom=TRUE`, `fSombra6` provides a weighted averaged of the shadows in the whole set of trackers, whose distribution in the PV plant is defined by `Nrow` and `Ncol`.

These functions are integrated in `calcShd`, `calcGef` and `prodGCPV`, as these examples show.

First, a two-axis tracking system.

```
> struct2x = list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 8)
> dist2x = data.frame(Lew = 40, Lns = 30, H = 0)
> prod2xShd <- prodGCPV(lat = lat, prom = prom, modeTrk = "two",
+     modeShd = "area", struct = struct2x, distances = dist2x)
```

Then, a N-S horizontal axis tracking system without backtracking,

```
> structHoriz = list(L = 4.83)
> distHoriz = data.frame(Lew = structHoriz$L * 4, H = 0)
> prodHorizShd <- prodGCPV(lat = lat, prom = prom, sample = "10 min",
+     modeTrk = "horiz", modeShd = "area", betaLim = 60, distances = distHoriz,
+     struct = structHoriz)
```

and a N-S horizontal axis tracking system with backtracking,

```
> prodHorizBT <- prodGCPV(lat = lat, prom = prom, sample = "10 min",
+     modeTrk = "horiz", modeShd = "bt", betaLim = 60, distances = distHoriz,
+     struct = structHoriz)
```

Finally, we can compare the *yearly* performance of these systems with the method `compare` (fig. 14), and calculate and compare their *yearly* losses with the methods `losses` and `compareLosses` (fig. 15), respectively.

4.3 Position of trackers in a PV plant

The optimum distance between trackers or static structures of a PV grid connected plant depends on two main factors: the ground requirement ratio (defined as the ratio of the total ground area to the PV generator area), and the productivity of the system including shadow losses. Therefore, the optimum separation may be the one which achieves the highest productivity with the lowest ground requirement ratio (GRR). However, this definition is not complete since the terrain characteristics and the costs of wiring or civil works could alter the decision.

The function `optimShd` is a help for choosing this distance: it computes the productivity for a set of combinations of distances between the elements of the plant [6]. The designer should adopt the decision from these results with the adequate economical translations.

Let's analyse the configuration of a PV plant with NS horizontal axis trackers, without *backtracking*, and a height of 4,83 m. We are interested in a range of separations of 2 and 5 times this dimension. Besides, the analysis will be carried out with a limitation in the angle of inclination:

```
> structHoriz = list(L = 4.83)
> distHoriz = list(Lew = structHoriz$L * c(2, 5))
> Shd12Horiz <- optimShd(lat = lat, prom = prom, modeTrk = "horiz",
+     betaLim = 60, distances = distHoriz, res = 2, struct = structHoriz,
+     modeShd = "area", prog = FALSE)
```

The function `optimShd` constructs an object of class `Shade` This class owns a S4 method of `plot` for displaying the results (figure 16).

Now, for a fixed system (figure 17):

```
> structFixed = list(L = 5)
> distFixed = list(D = structFixed$L * c(1, 3))
> Shd12Fixed <- optimShd(lat = lat, prom = prom, modeTrk = "fixed",
+     distances = distFixed, res = 1, struct = structFixed, modeShd = "area",
+     prog = FALSE)
```

Last, we are interested in a two-axis tracker whose dimensions are 23,11 m width and 9,8 m height. We will try to design a PV plant with a grid of trackers of 2 rows and 8 columns.

```
> struct2x = list(W = 23.11, L = 9.8, Nrow = 2, Ncol = 8)
```

We will try the separations between 30 m and 50 m for the E-O direction and between 20 m and 50 m for the N-S direction.

```
> comp <- compare(ProdFixed, Prod2x, ProdHoriz, prod2xShd, prodHorizShd,
+     prodHorizBT)
> head(comp)

  values  ind      name
1   1836  G0d ProdFixed
2   1719 Gefd ProdFixed
3   1329   Yf ProdFixed
4   1836  G0d    Prod2x
5   2747 Gefd    Prod2x
6   2093   Yf    Prod2x
```

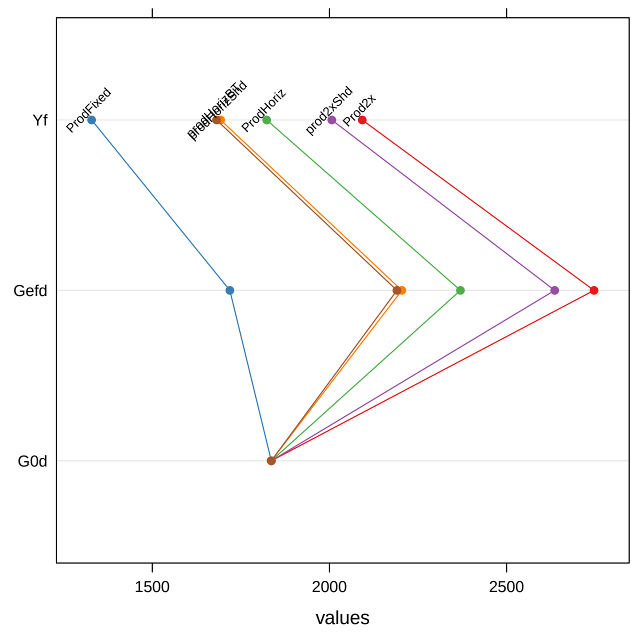

Figure 14: Comparison of several `ProdGCPV` objects.

```
> compL <- compareLosses(ProdFixed, Prod2x, ProdHoriz, prod2xShd,
+     prodHorizShd, prodHorizBT)
> head(compL)

         id  values      name
1   Shadows 0.00000 ProdFixed
2       AoI 0.05419 ProdFixed
3 Generator 0.07473 ProdFixed
4        DC 0.07435 ProdFixed
5  Inverter 0.06979 ProdFixed
6        AC 0.02973 ProdFixed
```

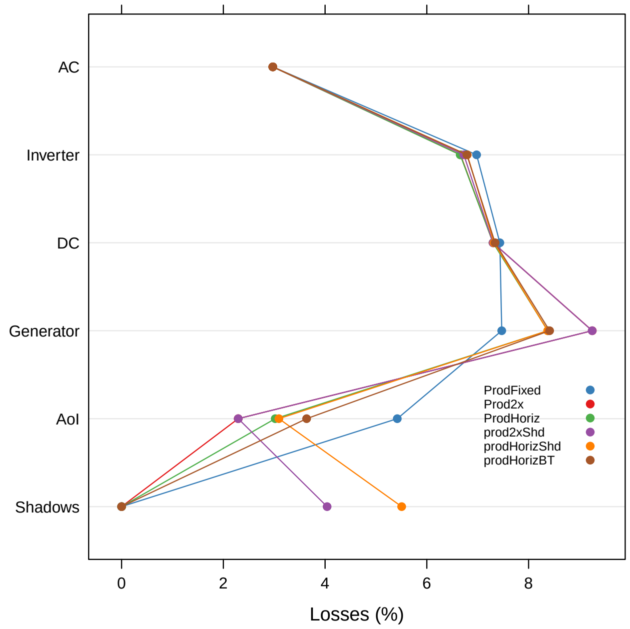

Figure 15: Comparison of the losses of several `ProdGCPV` objects.

```
> shadeplot(Shd12Horiz)
```

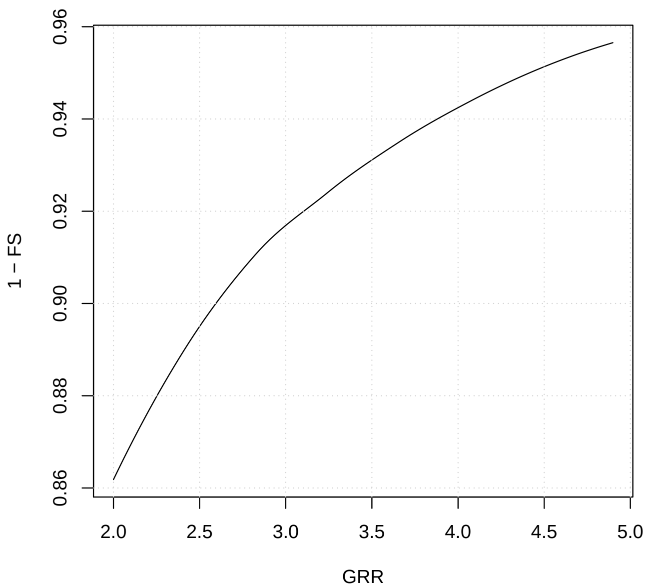

Figure 16: Mutual shadows in a NS horizontal axis tracking PV system.

```
> shadeplot(Shd12Fixed)
```

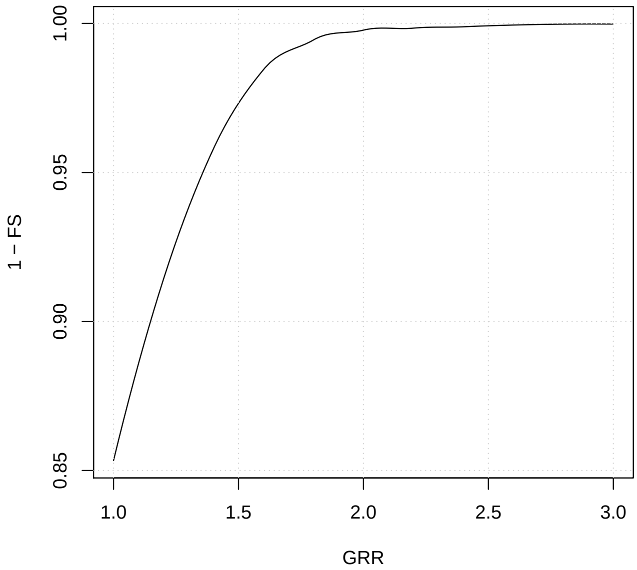

Figure 17: Mutual shadows in a PV plant with fixed structures.

```
> dist2x = list(Lew = c(30, 50), Lns = c(20, 50))
```

optimShd constructs a sequence from the minimum to the maximum value of distances, with res as the increment, in meters, of the sequence. In this example, res=5.

```
> ShdM2x <- optimShd(lat = lat, prom = prom, modeTrk = "two", modeShd = c("area",
+     "prom"), distances = dist2x, struct = struct2x, res = 5,
+     prog = FALSE)
```

Besides, the Shade object includes the local fitting of the sequence of Yf and FS values (slots named Yf.loess and FS.loess). The predict method is used with these loess slots inside the shadeplot method of the Shade class (figure 18).

5 PV pumping systems

5.1 Simulation of centrifugal pumps

The first step for the simulation of the performance of a PV pumping system (PVPS) is the characterization of the pump under the supposition of constant manometric height [1]. The function fPump computes the performance of the different parts of a centrifugal pump fed by a frequency converter following the affinity laws.

For example, we can characterize the performance of the SP8A44 pump (http://net.grundfos.com/Appl/WebCAPS/InitCtrl?mode=1) working with $H = 40$ m. The information of this pump is stored in the dataset pumpCoef.

```
> data(pumpCoef)
> CoefSP8A44 <- subset(pumpCoef, Qn == 8 & stages == 44)
> fSP8A44 <- fPump(pump = CoefSP8A44, H = 40)
```

The result of fPump is a set of functions which relate the electrical power and the flow, hydraulical and mechanical power, and frequency. These functions allow the calculation of the performance for any electrical power inside the range of the pump (figures 19 and 20):

```
> shadeplot(ShdM2x)
```

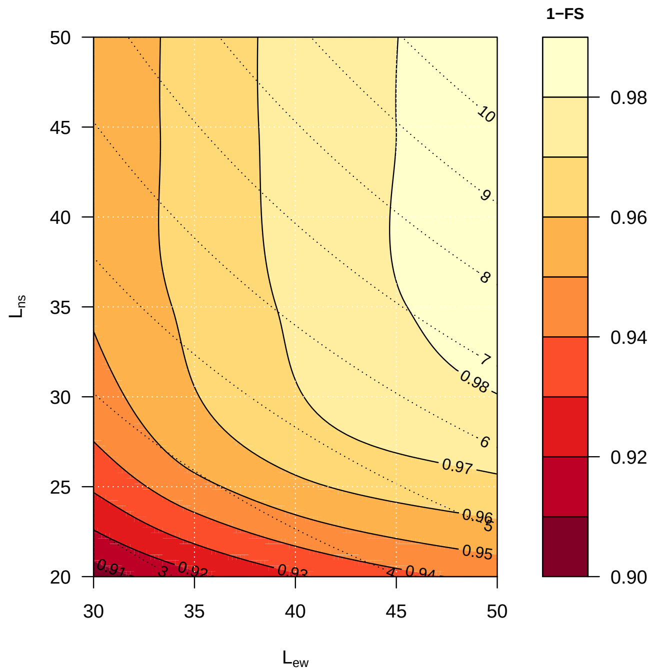

Figure 18: Mutual shadows in a two-axis tracking PV system for a combination of separations between trackers.

```
> SP8A44 = with(fSP8A44, {
+     Pac = seq(lim[1], lim[2], by = 100)
+     Pb = fPb(Pac)
+     etam = Pb/Pac
+     Ph = fPh(Pac)
+     etab = Ph/Pb
+     f = fFreq(Pac)
+     Q = fQ(Pac)
+     result = data.frame(Q, Pac, Pb, Ph, etam, etab, f)
+ })
> SP8A44$etamb = with(SP8A44, etab * etam)
```

5.2 Nomograms of PVPS

The international standard IEC 61725 is of common usage in public licitations of PVPS. This standard proposes a equation of the irradiance profile with several parameters such as the length of the day, the daily irradiation and the maximum value of the irradiance. With this profile, the performance of a PVPS can be calculated for several manometric heights and nominal PV power values. A nomogram can display the set of combinations. This graphical tool can help to choose the best combination of pump and PV generator for certain conditions of irradiation and height [1].

This kind of graphics is provided by the function `NmgPVPS`. For example, the figure 21 is a nomogram for the SP8A44 pump working in a range of heights from 50 to 80 meters, with different PV generators. The peculiar shape of the curve of 50 meters shows that this pump does not work correctly with this height.

5.3 Productivity of PVPS

A different approach is to simulate the performance of the PVPS following the same procedure as the one described for the GCPV systems. The function `prodPVPS` is the equivalent to the function `prodGCPV`. The inputs are very similar between them, although there are some changes due to the different composition of the system. This function does not allow for the calculation of shadows.

Once again with the SP8A44 pump, we compute the flow to be produced by this pump with a PV generator of 5500 Wp and a manometric height of 50 meters. The relation between flow and effective irradiance is displayed in the figure 22.

```
> lab = c(expression(eta[motor]), expression(eta[pump]), expression(eta[mp]))
> p <- xyplot(etam + etab + etamb ~ Pac, data = SP8A44, type = "l",
+     ylab = "Eficiencia")
> print(p + glayer(panel.text(x[1], y[1], lab[group.number], pos = 3)))
```

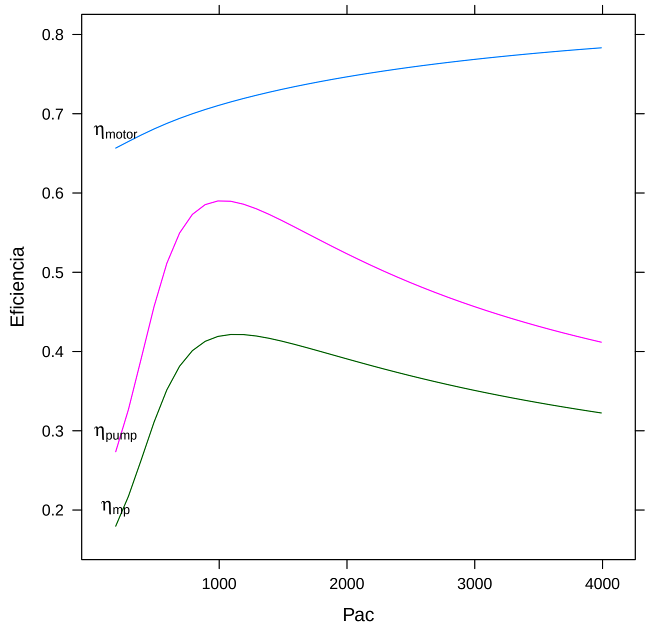

Figure 19: Efficiency of the motor and pump for several values of electrical power of a SP8A44 pump with $H = 40\,\text{m}$

```
> lab = c(expression(P[pump]), expression(P[hyd]))
> p <- xyplot(Pb + Ph ~ Pac, data = SP8A44, type = "l", ylab = "Power (W)",
+     xlab = "AC power (W)")
> print(p + glayer(panel.text(x[length(x)], y[length(x)], lab[group.number],
+     pos = 3)))
```

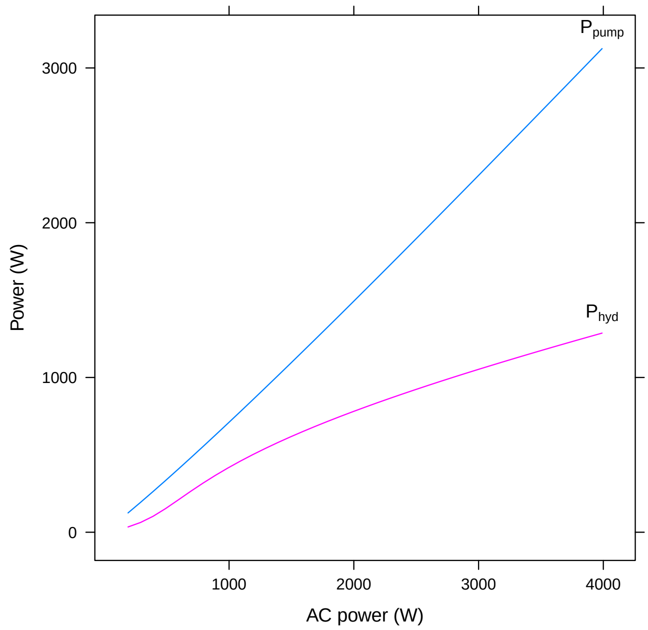

Figure 20: Mechanical and hydraulical power versus electrical power of a SP8A44 pump with $H = 40\,\text{m}$.

```
> Pg = seq(3000, 5500, by = 500)
> H = seq(50, 80, by = 5)
> NmgSP8A44 <- NmgPVPS(pump = CoefSP8A44, Pg = Pg, H = H, Gd = 6000,
+     title = "Selection of Pumps", theme = custom.theme())
> print(NmgSP8A44$plot)
```

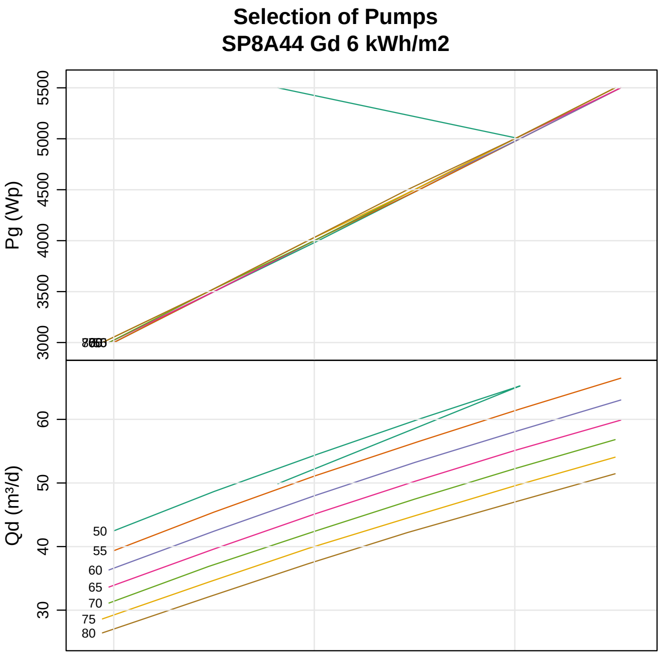

Figure 21: Nomogram for the SP8A44 pump working in a range of heights from 50 to 80 meters, with different PV generators.

```
> prodSP8A44 <- prodPVPS(lat = 41, modeRad = "mapa", mapa = list(prov = 28,
+     est = 3, start = "01/01/2009", end = "31/12/2009"), pump = CoefSP8A44,
+     Pg = 5500, H = 50)

Downloading data from www.mapa.es/siar...

> as.zooY(prodSP8A44)

      Eac    Qd   Yf
2009 6757 19233 1229
```

Let's try to obtain more water with this pump using a larger PV generator of 7000 Wp. However, we can check that this is not a correct decision since the productivity has decreased. The figure 23 shows that during the central months of the year, during the maximum irradiance periods, the pump reaches its limits of flow and frequency, and so the frequency converter stops the system. Finally, the figure 24 shows the evolution of the daily productivity of these two configurations.

```
> prodSP8A44Lim <- prodPVPS(lat, modeRad = "prev", prev = prodSP8A44,
+     pump = CoefSP8A44, H = 50, Pg = 7000)
> as.zooY(prodSP8A44Lim)

      Eac    Qd   Yf
2009 7527 20770 1075
```

6 Statistical analysis of PV plants

In a PV plant, the individual systems are theoretically identical and their performance along the time should be the same. Due to their practical differences –power tolerance, dispersion losses, dust–, the individual performance of each system will deviate from the average behaviour. However, when a system is performing correctly, these deviations are constrained inside a range and should not be regarded as a sign of malfunctioning.

If these common deviations are assumed as a random process, a statistical analysis of the performance of the whole set of systems can identify a faulty system as the one that departs significantly from the mean behaviour.

The functions `analyzeData` and `Target Diagram` compare the daily performance of each system with a reference (for example, the median of the whole set) during a time period of N days preceding the current day. They calculate

```
> p = xyplot(Q ~ Gef | month, data = prodSP8A44, cex = 0.5, type = c("p",
+     "smooth"), col.symbol = "gray", col.line = "black")
> print(p)
```

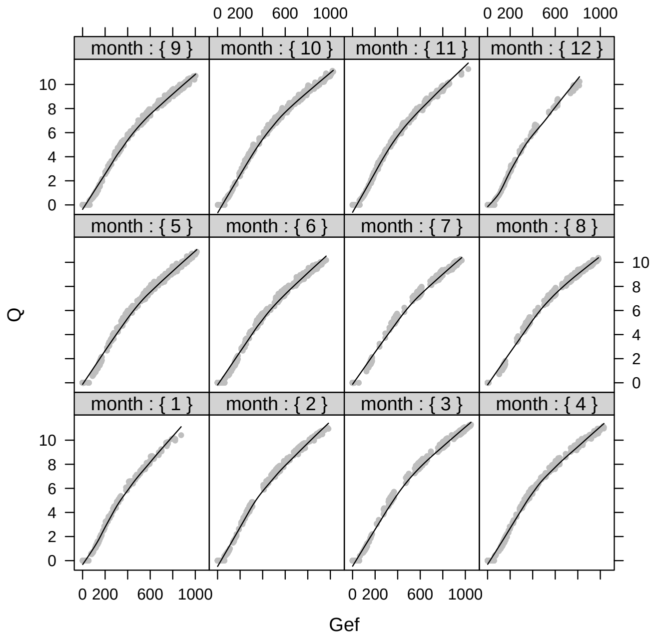

Figure 22: Flow versus irradiance of a PVPS with a SP8A44 pump and a PV generator with a nominal power of 5500 Wp and a manometric height of 50 meters.

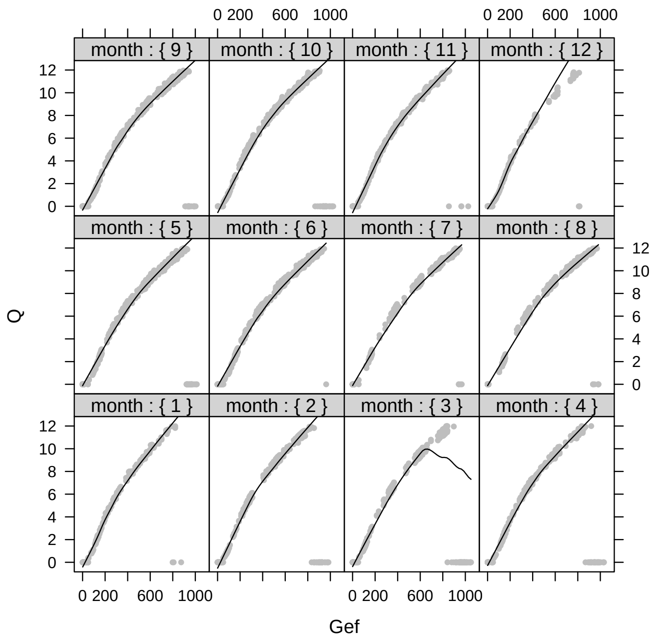

Figure 23: Water flow versus irradiance of a PVPS system with a SP8A44 pump and a generator of 7000 Wp with a manometric height of 50 meters.

```
> compPVPS <- mergesolaR(prodSP8A44, prodSP8A44Lim)
> print(xyplot(compPVPS, superpose = TRUE, ylab = "Yf"))
```

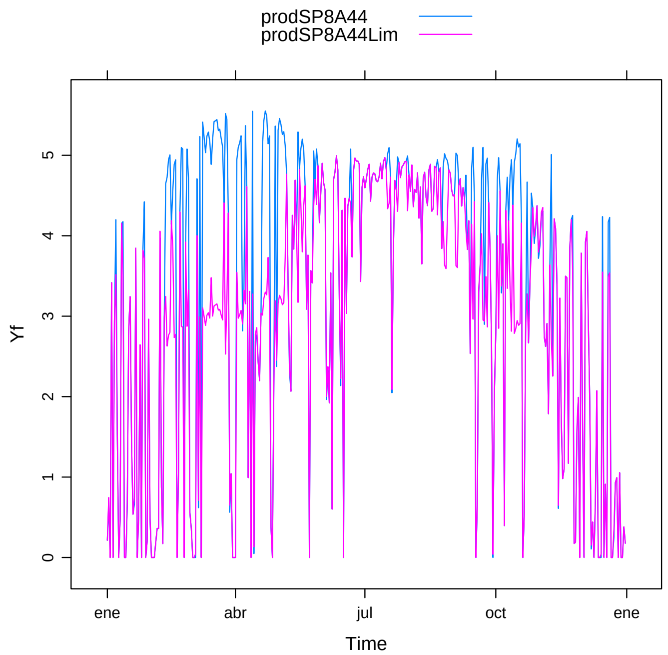

Figure 24: Comparison of the daily productivity of two pumping PV systems.

a set of statistics of the performance of the PV plant as a whole, and another set of the comparison with the reference. This statistical analysis can be summarised with a graphical tool named "Target Diagram", which plots together the root mean square difference, the average difference and the standard deviation of the difference. Besides, this diagram includes the sign of the difference of the standard deviations of the system and the reference [7].

The next example uses a dataset of productivity from a PV plant composed of 22 systems (`data(prodEx)`). It is clear that the system no.20 is not working correctly during these periods (horizonplot of figure 25 and target diagram of figure 26).

```
> data(prodEx)
> prodStat <- analyzeData(prodEx)
```

Let's remember the example devoted to `mergesolaR`, with the result displayed in the figure 12. The function `TargetDiagram` is an alternative tool to show the behaviour of the set of meteorological stations (figure 27).

7 Changes

solaR 0.22

- A new `mergesolaR` method has been defined for merging `solaR` objects.
- The calculation of the sunset time has been improved.
- The voltage dependency of the efficiency curve of the inverter is now included in `fProd` and `calcGCPV`.
- The default values of the `module`, `generator` and `inverter` of both `fProd`, `calcGCPV` and `optimShd` is now documented.
- The help page of optimShd now explains correctly the concept of GRR.
- The plot method for Shade has been renamed to shadeplot.
- The `as.data.frame` method of the `Shade` class is now exported.

```
> dif <- prodEx - prodStat$stat$Median
> day = as.Date("2008-8-29")
> p <- horizonplot(window(dif, start = day - 60, end = day), origin = 0,
+      layout = c(1, 22), colorkey = TRUE, colorkey.digits = 1,
+      scales = list(y = list(relation = "same")))
> print(p)
```

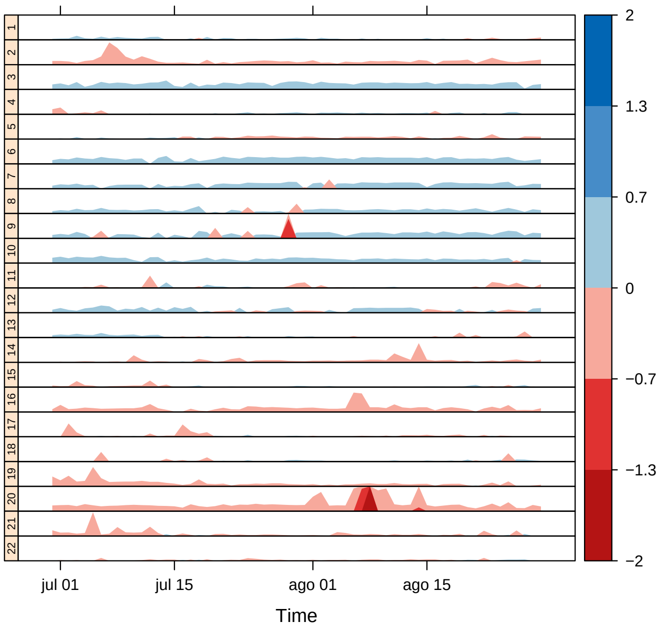

Figure 25: Horizonplot of the differences of productivity of a set of 22 PV systems.

```
> ndays = c(5, 10, 15, 20)
> palette = brewer.pal(n = length(ndays), name = "Set1")
> TDColor <- TargetDiagram(prodEx, end = day, ndays = ndays, color = palette)
> print(TDColor$plot)
```

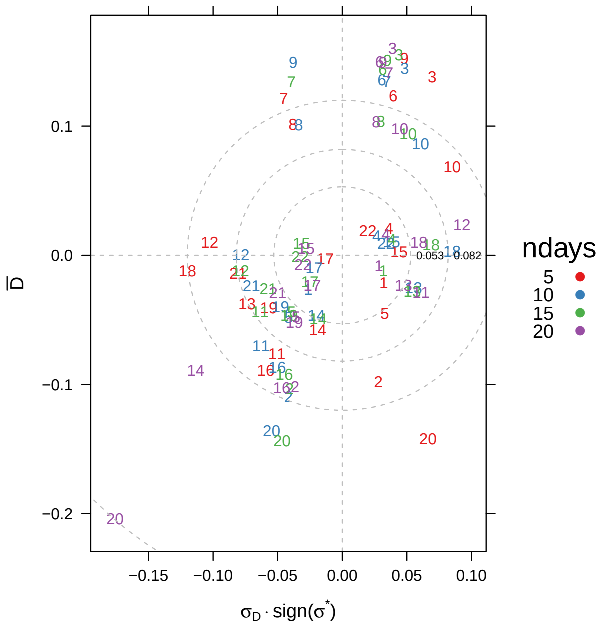

Figure 26: "Target Diagram" of the statistical analysis of a set of 22 systems during various time periods.

```
> TDMadrid <- TargetDiagram(YfMadrid, end = as.POSIXct("2010-12-31"),
+     ndays = c(10, 20, 30, 40, 50, 60), cex = 0.5)
> print(TDMadrid$plot)
```

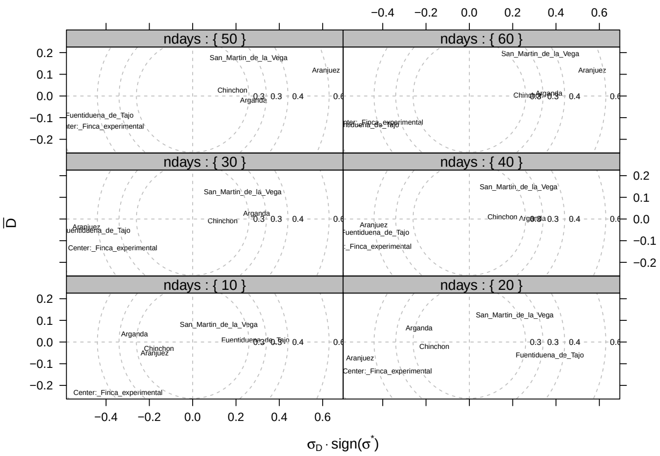

Figure 27: Target Diagram of the result of the mergesolaR example

solaR 0.21

solaR is now able to calculate from both daily and sub-daily irradiation values. Besides,

- `calcSol` and `fSolI` gain a `"BTi"` argument for intradaily time bases.
- `fCompI` gains a `"G0I"` argument for intradaily irradiation series.
- `fCompI` gains both `"corr"` and `"f"`.
- `calcG0`, `calcGef`, `prodGCPV` and `prodPVPS` gain a new `"bdI"` argument for intradaily irradiation, and the `"corr"`, `"f"` arguments.
- The `"bd"` (and the new `"bdI"`) argument of `"calcG0"` can be now a `"Meteo"` object. The `"file"` component of this argument can be now a `"zoo"` object.
- New methods (`"losses"`, `"compareLosses"` and `"compare"`) are available for `"Gef"` and `"ProdGCPV"` classes.
- The `"corr"` argument of `"fCompD"` (and `"fCompI"`) can be now `"corr=none"`.
- The correlations between the diffuse fraction and the clearness index are now coded outside `"fCompD"` as separate functions. Several new correlations have been included, both for monthly/daily values and for intradaily values.
- New small functions for `difftime` objects have been included.

solaR 0.20

- The package is now almost entirely designed with S4 classes and methods.
- The time series object are constructed with the 'zoo' package.
- Most of the functions and arguments have been renamed in order to ease the understanding by international users.
- Two new functions have been included for the statistical analysis of a PV plant composed of several systems.
- The package dependencies have been optimized.
- Several new small functions for date-time calculations are now available.

References

[1] M.~Alonso Abella, E.~Lorenzo, and F.~Chenlo. Pv water pumping systems based on standard frequency converters. *Progress in Photovoltaics: Research and Applications*, 11(3):179–191, 2003.

[2] M.~Collares-Pereira and Ari Rabl. The average distribution of solar radiation: correlations between diffuse and hemispherical and between daily and hourly insolation values. *Solar Energy*, 22:155–164, 1979.

[3] Thomas~A. Huld, Marcel Súri, Ewan~D. Dunlop, and Fabio Micale. Estimating average daytime and daily temperature profiles within europe. *Environmental Modelling & Software*, 21(12):1650 – 1661, 2006.

[4] N.~Martin and J.~M. Ruíz. Calculation of the pv modules angular losses under field conditions by means of an analytical model. *Solar Energy Materials & Solar Cells*, 70:25–38, 2001.

[5] D.~Panico, P.~Garvison, H.~J. Wenger, and D.~Shugar. Backtracking: a novel strategy for tracking pv systems. In *IEEE Photovoltaic Specialists Conference*, pages 668–673, 1991.

[6] O.~Perpiñán. *Grandes Centrales Fotovoltaicas: producción, seguimiento y ciclo de vida*. PhD thesis, UNED, 2008.

[7] O.~Perpiñán. Statistical analysis of the performance and simulation of a two-axis tracking pv system. *Solar Energy*, 83(11):2074–2085, 2009.

[8] O.~Perpiñán. *Energía Solar Fotovoltaica*. 2011.

[9] A.~Zeileis and G.~Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005.