Introduction to package `shadow`

Michael Dorman, Adi Vulkan, Evyatar Erell, Itai Kloog
Ben-Gurion University of the Negev

2016-11-30

Contents

1 Introduction **1**

1.1 Calculation 2

2 Functions **3**

3 Examples **3**

3.1 Shade height 3

3.2 Calculating shade height for each point 5

3.3 Shade footprint 7

4 References **11**

1 Introduction

Spatial analysis of the urban environment frequently requires estimating *shading*. For example -

- The amount of time a given point / roof / facade is shaded may determine the utility of installing Photo-Voltaic cells for electricity production
- The Sky View Factor (SVF) of a given street is associated with its microclimate
- Calculating shade footprint on a green park helps urban planners provide guidlenes for maximal height of a new building

These types of calculations (and more advanced ones) are usually restricted to proprietary software dealing with 3D models, such as ESRI's ArcScene. Terrain-based solutions (i.e. Digital Elevation Model, DEM) are more common, in both open-source (GRASS GIS) as well as proprietary (ArcGIS) software. The `insol` R packge gives such capability in R. However terrain-based approaches may not be appropriate for an urban environment, for two reasons.

First, a continuous elevation surface at the necessary resolution for the urban context (e.g. LIDAR) may not be available and is expensive to produce. Second, the DEMs cannot adequately represent individual urban elements such as *building facades*, thus limiting the interpretability of results. The `shadow` package aims at addressing these limitations. The `shadow` package operates on a *vector layer* of building outlines along with their heights, rather than a DEM. Such data are generally much more available, either from local municipalities or from global datasets such as OpenStreetMap. Therefore the resulting shade estimates correspond to urban environment such as individual buildings or facade. It should be noted that the approach assumes a flat terrain and no obstacles (e.g. trees) other than the buildings. which may be inappropriate in certain situations (e.g. a montaneous urban area).

1.1 Calculation

The functions currently included `shadow` are based on the trigonometric in the triangle defined by the sun rays, the ground (or plane parallel to the ground) and an obstacle. For example, as shown in Figure 1, shade height (h_{shade}) at a given location (*observer*) can be calculated based on the sun elevation angle (α) in case there is one obstacle with a known height (h_{build}) and at a known distance ($dist$) -

$h_{shade} = h_{build} - dist * tan(\alpha)$

This approach can be explanded for the general case of multiple obstacles, in which case we must take the *maximum* value of all potential shade heights caused by the obstacles.

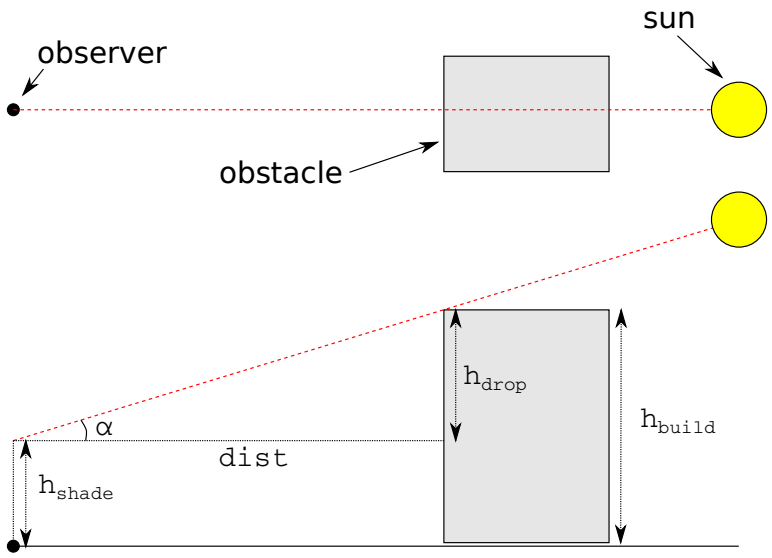

Figure 1: Shade height calculation

2 Functions

The `shadow` package currently contains three main functions, as well as several helper functions. The main functions give three distinct aspects of shading -

- The `shadeHeight` function calculates shade height at a single point; the wrapper `shadePropWall` gives the shaded fraction of a facade, by repeatedly calling `shadeHeight` on sample points along its length.
- The `shadeFootprint` function calculates the shade footprint on the ground, given the layer of obstacles and sun position.
- The `SVF` function calculates the Sky View Factor (SVF), which is a measure of sky proportion not obstructed to the viewer at a given point on the ground.

3 Examples

Before going into the examples, we load the `shadow` package as well as packages `sp` (loaded automatically), `raster` and `rgeos` -

```
library(shadow)
library(raster)
library(rgeos)
```

In the examples we will use a polygonal layer representing four buildings (`build`) along having a height attribute (`BLDG_HT`) as shown on Figure 2 -

```
plot(build)
text(gCentroid(build, byid = TRUE), build$BLDG_HT)
```

3.1 Shade height

The `shadeHeight` function calculates shade height at a given point location. For example, to calculate shade height at the centroid of the layer (`location`), on `2004-12-24 13:30:00` we first need to determine the sun elevation and azimuth at that time. This can be done with function `solarpos` from package `maptools` -

```
location = gCentroid(build)
time = as.POSIXct("2004-12-24 13:30:00", tz = "Asia/Jerusalem")
location_geo = spTransform(location, "+proj=longlat +datum=WGS84")
solar_pos = maptools::solarpos(location_geo, time)
```

Figure 2: Sample buildings and heights

```
solar_pos
#>          [,1]     [,2]
#> [1,] 208.7333 28.79944
```

Now we know the sun azimuth (208.7) and elevation (28.8). Given sun position, the layer of obstacles and queried location, shade height can be calculated with `shadeHeight` -

```
h = shadeHeight(location, build, "BLDG_HT", solar_pos)
#> Assuming BLDG_HT given in m
h
#> [1] 19.86451
```

Shade height at the queried point is 19.86 meters. Note the warning regarding the units of the `BLDG_HT` attribute. The function has no way of knowing the height dimension units are the same as the Coordinate Reference System (CRS) spatial distance units. It is up to the user to make sure.

The following code and subsequent Figure 3 illustrate how the calculation is carried out. First, a line of sight `ray` is drawn between the point of interest `location` and the sun position based on its azimuth `sun_az`. Potential intersections `inter` are then detected. Finally, the shade height induced by each intersection is calculated based on the distance to intersection, sun elevation `sun_elev` and building height. The final result is the maximum value of these potential heights.

```
sun = shadow:::.sunLocation(
  location = location,
  sun_az = solar_pos[1, 1],
  sun_elev = solar_pos[1, 2]
```

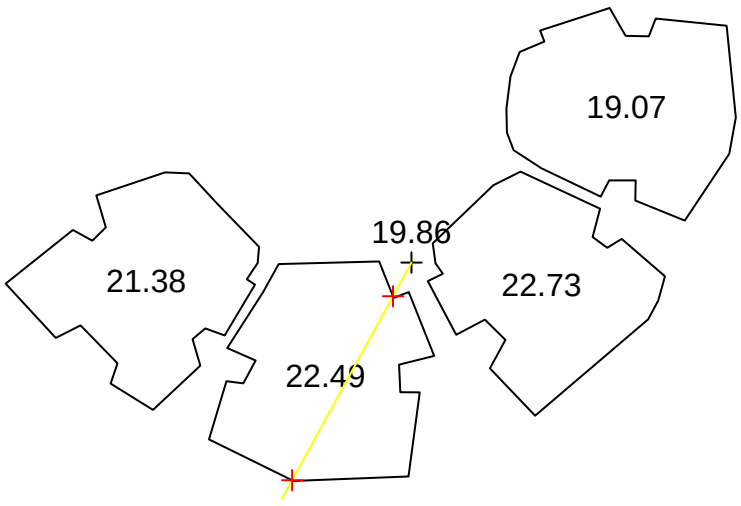

Figure 3: Shade height at a single location

```
  )
sun_ray = ray(from = location, to = sun)
build_outline = as(build, "SpatialLinesDataFrame")
inter = gIntersection(build_outline, sun_ray)
plot(build)
text(gCentroid(build, byid = TRUE), build$BLDG_HT)
plot(location, add = TRUE)
text(
  location,
  round(shadeHeight(location, build, "BLDG_HT", solar_pos), 2),
  pos = 3
  )
plot(sun_ray, add = TRUE, col = "yellow")
plot(inter, add = TRUE, col = "red")
```

3.2 Calculating shade height for each point

The procedure can be readily expanded to calculate a continuous surface of shade heights. To make it faster, we can use `mclapply` from package `parallel` or any other parallelization solution.

First, we will create a grid covering the examnined area with a spatial resolution of 2 meters -

```
ext = as(extent(build) + 50, "SpatialPolygons")
r = raster(ext, res = 2)
proj4string(r) = proj4string(build)
grid = rasterToPoints(r, spatial = TRUE)
```

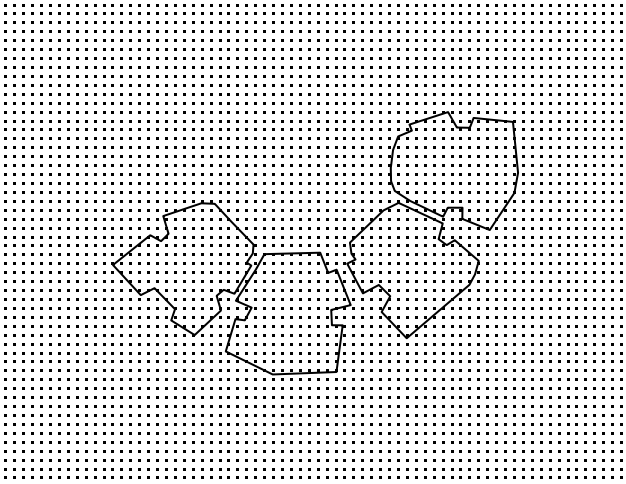

Figure 4: Input grid for creating a shade heights surface

```
grid = SpatialPointsDataFrame(
  grid,
  data.frame(grid_id = 1:length(grid))
  )
```

The following code section plots the resulting grid (Figure 4).

```
plot(grid, pch = ".")
plot(build, add = TRUE)
```

Calculating shade height at each point, using package `parallel` -

```
library(parallel)
shade_heights = mclapply(
  split(grid, grid$grid_id),
  shadeHeight,
  build, "BLDG_HT", solar_pos,
  mc.cores = 3
  )
grid$shade_height = simplify2array(shade_heights)
```

The calculation of shade height at 3780 grid points took 18 seconds on a laptop with an Intel® Core™ i5-6200U CPU @ 2.30GHz × 4 processor using the above 3-core parallel process. The resulting grid can be converted to a `RasterLayer` object of shade heights and plotted with following code section (Figure 5). Note the partial shade on the roof of the 19.07-m building which is caused by the slightly taller 22.73-m building.

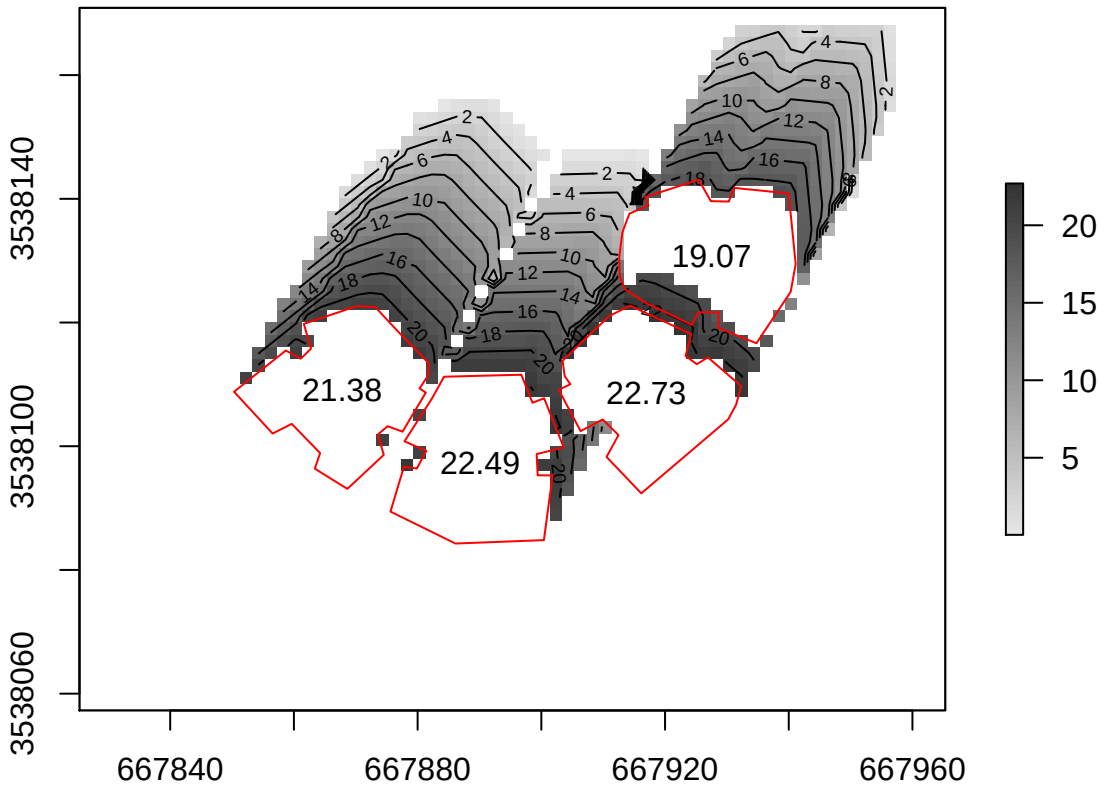

Figure 5: Shade height (m) grid

```
shade = as(grid, "SpatialPixelsDataFrame")
shade = raster(shade, layer = "shade_height")
plot(shade, col = grey(seq(0.9, 0.2, -0.01)))
plot(shade, col = grey(seq(0.9, 0.2, -0.01)))
contour(shade, add = TRUE)
plot(build, add = TRUE, border = "red")
text(gCentroid(build, byid = TRUE), build$BLDG_HT)
```

3.3 Shade footprint

The `shadeFootprint` function calculates the geometry of shade projection on the ground, rather than its height at discrete sampling points. The resulting layer can be used to calculate the proportion of shaded surface out of a defined area, to examine which building shades a given element, etc.

For example, suppose a playground is being built in vicinity to the four buildings in `build`, as shown on Figure 6.

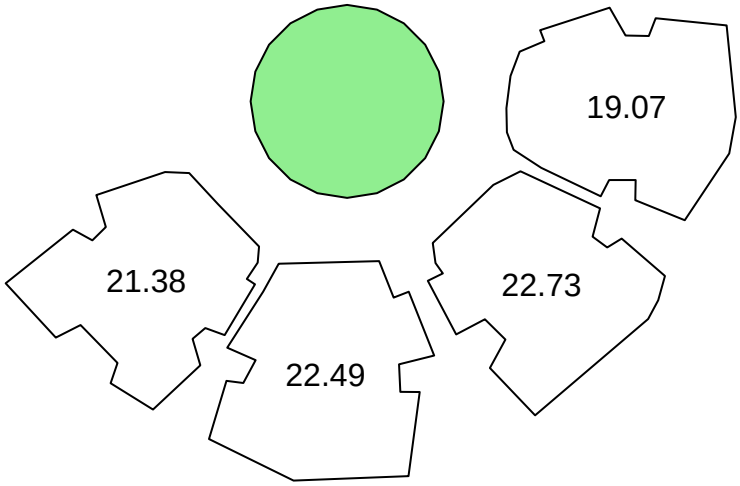

Figure 6: Park location

```
park_location = raster::shift(location, y = 20, x = -8)
park = rgeos::gBuffer(park_location, width = 12)
plot(build)
text(gCentroid(build, byid = TRUE), build$BLDG_HT)
plot(park, col = "lightgreen", add = TRUE)
```

Using `shadeFootprint` we can determine to what extent the palyground is shaded at a given time. For example, sun position at 2004-06-24 09:30:00 is (88.8, 46.7). The corresponding park shade proportion can be calculated by intersecting the shade footprint with park area and claculating the ratio. The result is 34.5%.

```
time2 = as.POSIXct("2004-06-24 09:30:00", tz = "Asia/Jerusalem")
solar_pos2 = maptools::solarpos(location_geo, time2)
solar_pos2
#>          [,1]   [,2]
#> [1,] 88.83113 46.724
footprint = shadeFootprint(build, "BLDG_HT", solar_pos2)
park_shade = gIntersection(park, footprint)
shade_prop = gArea(park_shade) / gArea(park)
shade_prop
#> [1] 0.3447709
```

The following code section graphically demonstrates the resulting shaded proportion (Figure 7).

```
plot(footprint,  col = adjustcolor("lightgrey", alpha.f = 0.5))
plot(build, col = "darkgrey", add = TRUE)
plot(park, col = "lightgreen", add = TRUE)
```

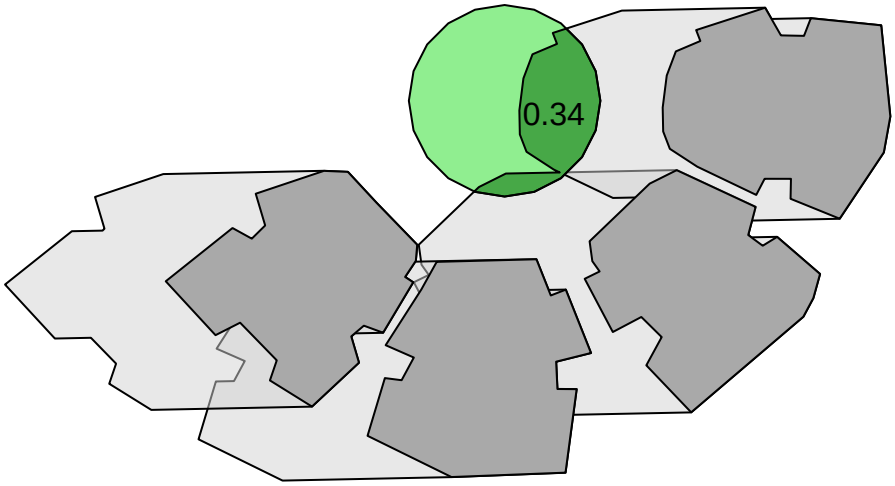

Figure 7: Shade footprint at 2004-06-24 09:30:00

```
plot(park_shade, col = adjustcolor("darkgreen", alpha.f = 0.5), add = TRUE)
text(gCentroid(park_shade), round(shade_prop, 2))
```

The calculation can be repeated for a sequence of times, rather than a single one, to monitor the daily (monthly, annual) course of shade proportions. For example, the following code creates a matrix `solar_pos_seq` which represents the sun positions course on `2004-06-24` at hourly intervals -

```
time_seq = seq(
  from = as.POSIXct("2004-06-24 03:30:00", tz = "Asia/Jerusalem"),
  to = as.POSIXct("2004-06-24 22:30:00", tz = "Asia/Jerusalem"),
  by = "1 hour"
)
solar_pos_seq = maptools::solarpos(location_geo, time_seq)
solar_pos_seq
#>             [,1]       [,2]
#>  [1,]  41.08360 -21.769851
#>  [2,]  51.77580 -12.527348
#>  [3,]  60.65421  -1.806295
#>  [4,]  68.26563   9.597935
#>  [5,]  75.12424  21.620929
#>  [6,]  81.74838  34.061974
#>  [7,]  88.83113  46.723999
#>  [8,]  97.78504  59.410673
#>  [9,] 113.28070  71.684791
#> [10,] 160.14772  80.943168
#> [11,] 233.16186  76.606322
#> [12,] 256.70087  64.978904
```

```
#> [13,] 267.54640  52.385195
#> [14,] 275.20092  39.682139
#> [15,] 281.92087  27.120222
#> [16,] 288.60587  14.879269
#> [17,] 295.81954   3.268622
#> [18,] 304.06676  -7.945221
#> [19,] 313.88346 -17.870062
#> [20,] 325.79830 -26.100500
```

Using a `for` loop over `solar_pos_seq`, the following code section calculates the vector of park shade proportions over the entire day. Note the two conditional statements: (1) shade proportion is maximal (i.e. 1) when sun is below the horizon and (2) shade proportion is minimal (i.e. 0) when no intersections are detected between the park and the shade footprint.

```
shade_props = rep(NA, nrow(solar_pos_seq))
for(i in 1:nrow(solar_pos_seq)) {
  if(solar_pos_seq[i, 2] < 0)
    shade_props[i] = 1 else {
      footprint =
        shadeFootprint(
          build,
          "BLDG_HT",
          solar_pos_seq[i, , drop = FALSE]
          )
    park_shade = gIntersection(park, footprint)
    if(is.null(park_shade))
      shade_props[i] = 0
    else
      shade_props[i] = gArea(park_shade) / gArea(park)
    }
}
```

Figure 8 summarizes the temporal variation in park shade proportion over the chosen day. The individual value we manually calculated in a previous step (2004-06-24 09:30:00) is highlighted in red.

```
plot(
  time_seq,
  shade_props,
  xlab = "Time",
  ylab = "Shade proportion",
  type = "b"
  )
text(time_seq[7], shade_props[7], round(shade_props[7], 2), pos = 4, col = "red")
```

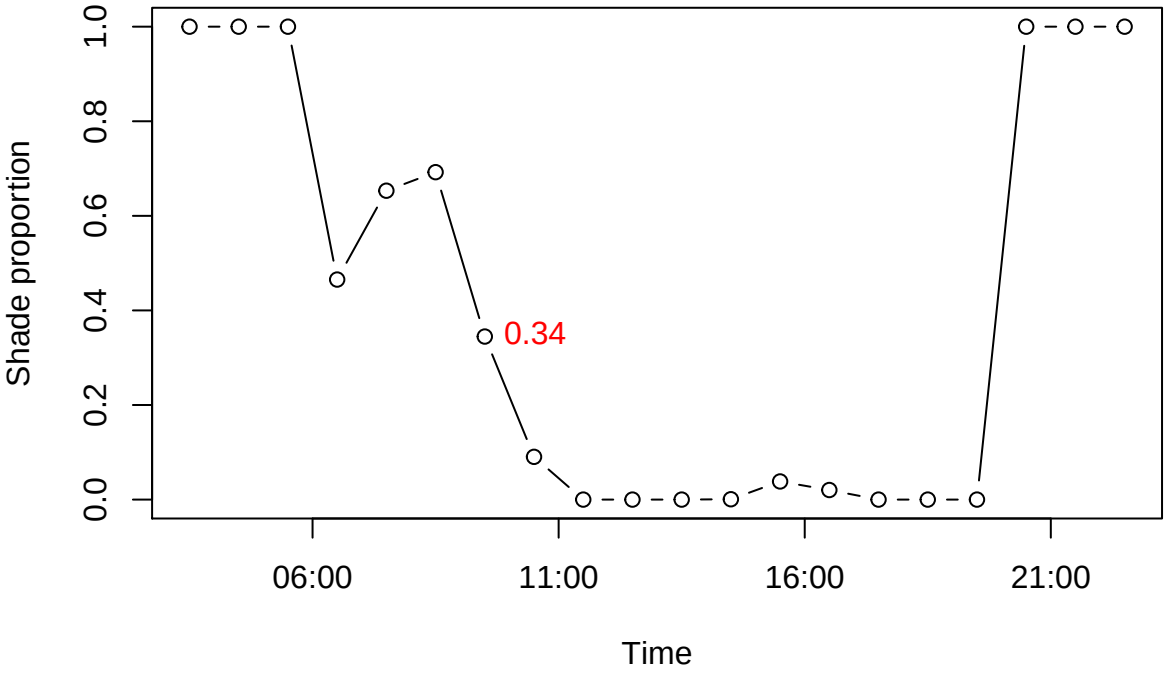

Figure 8: Shaded park proportion on 2004-06-24

4 References