R package Least Cost Path: formulating least cost path analyses

Joseph Lewis

2019-11-28

1. Introduction

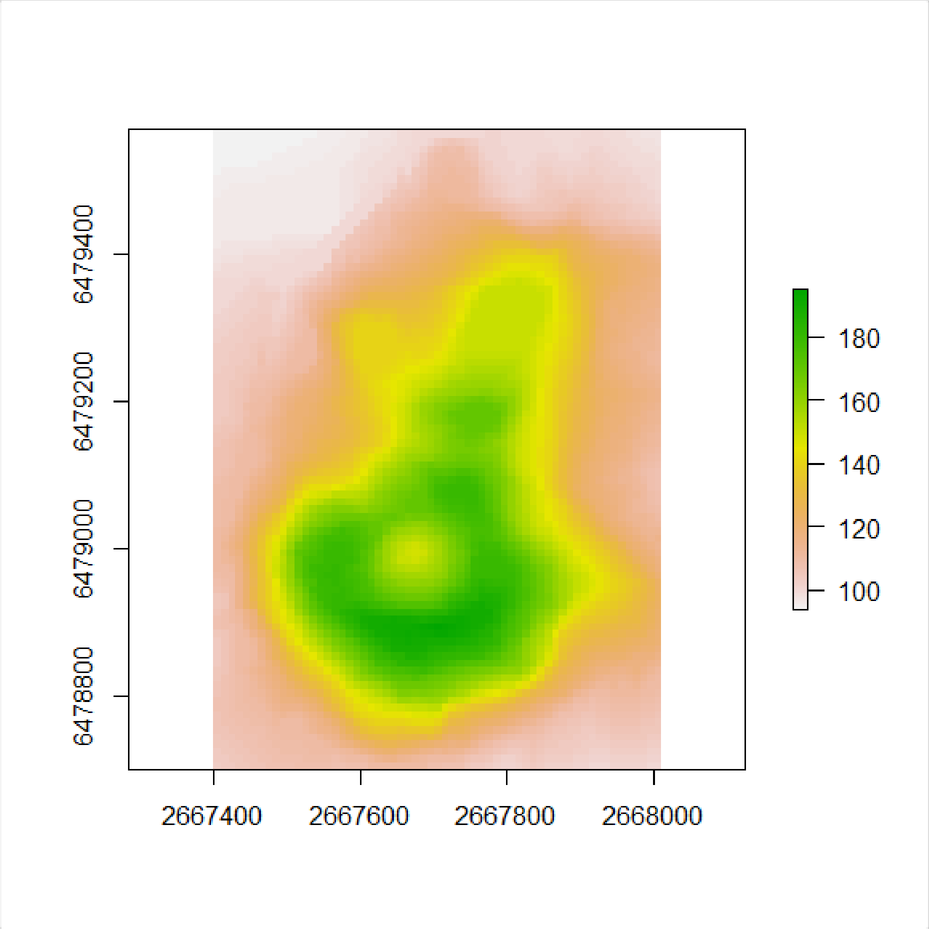
This vignette describes **leastcostpath**, a package written for use in the R environment (R Core Team, 2016). It provides functionality to calculate Least Cost Paths using multiple cost functions that approximate the difficulty of moving across a landscape, taking into account obstacles and local fricion (e.g. slope). Furthermore, this package allows for the incorporation of cost when traversing across slope, as well as other factors such as landscape feature attraction.

2. Example 1: Least Cost Path Analysis (Slope only)

```
library(rgdal)
library(rgeos)
library(sp)
library(raster)
library(gdistance)
library(leastcostpath)
```

```
r <- raster::raster(system.file('external/maungawhau.grd', package = 'gdistance'))

plot(r)
```



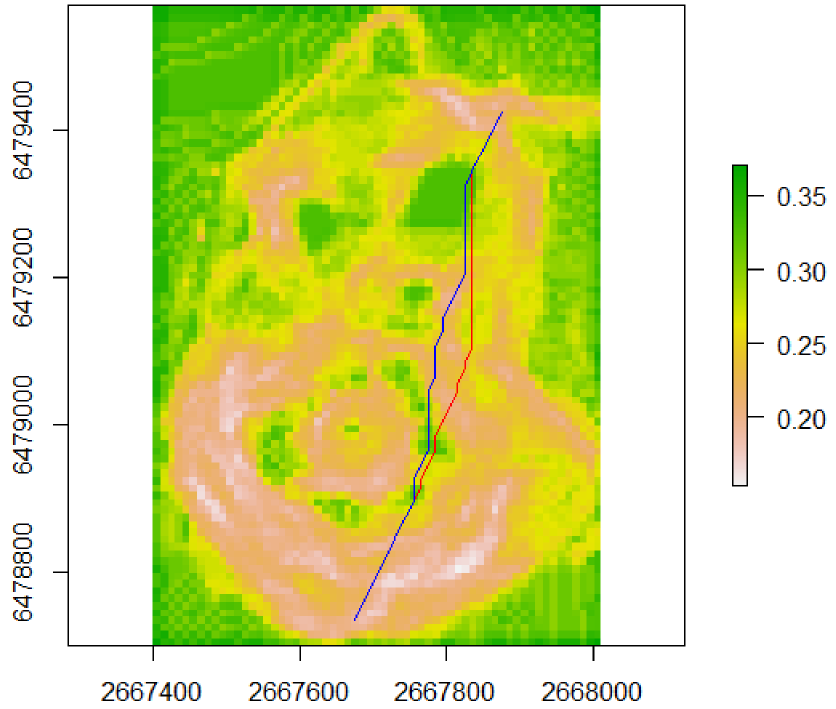
```
cs <- create_slope_cs(dem = r, cost_function = 'tobler', neighbours = 16)
```

```
loc1 = cbind(2667876, 6479424)
loc1 = sp::SpatialPoints(loc1)

loc2 = cbind(2667677, 6478737)
loc2 = sp::SpatialPoints(loc2)

lcp <- create_lcp(cost_surface = cs, origin = loc1, destination = loc2, directional = FALSE)

plot(raster(cs))
plot(lcp[[1]], add = T, col = "red")
plot(lcp[[2]], add = T, col = "blue")
```

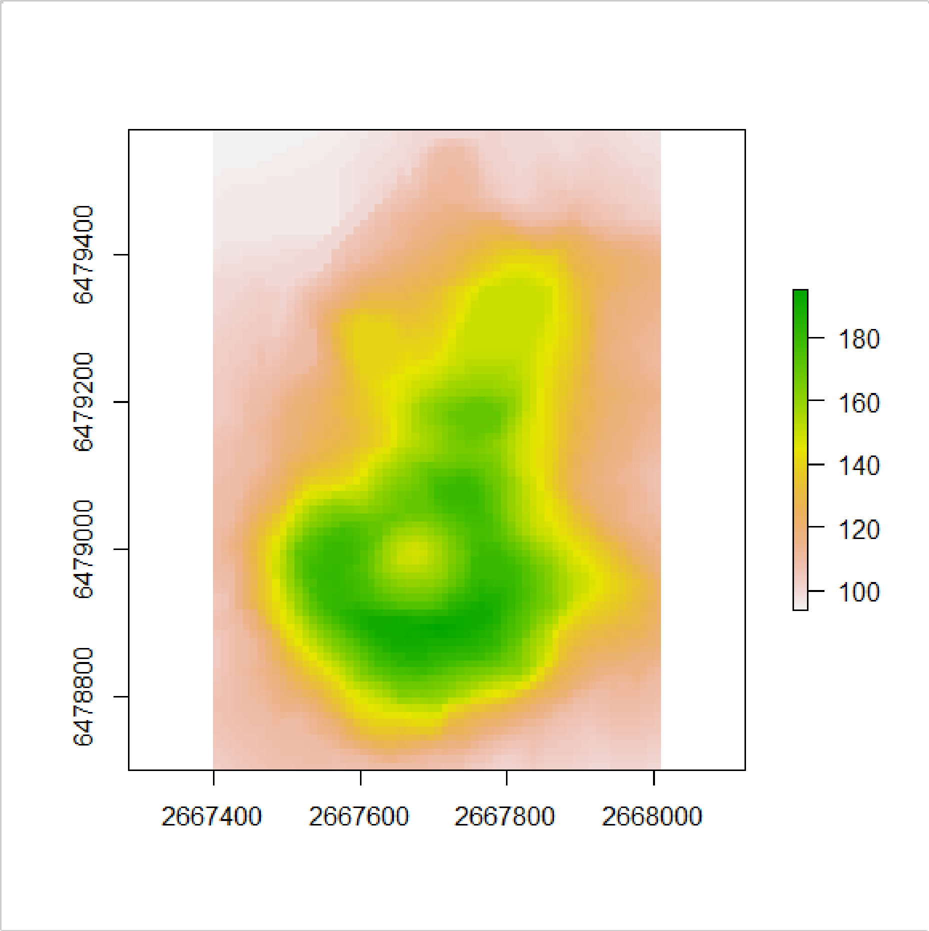


3. Example 2: Least Cost Path Analysis (Slope and Traversal Across Slope)

```
library(rgdal)
library(rgeos)
library(sp)
library(raster)
library(gdistance)
library(leastcostpath)
```

```
r <- raster::raster(system.file('external/maungawhau.grd', package = 'gdistance'))

plot(r)
```



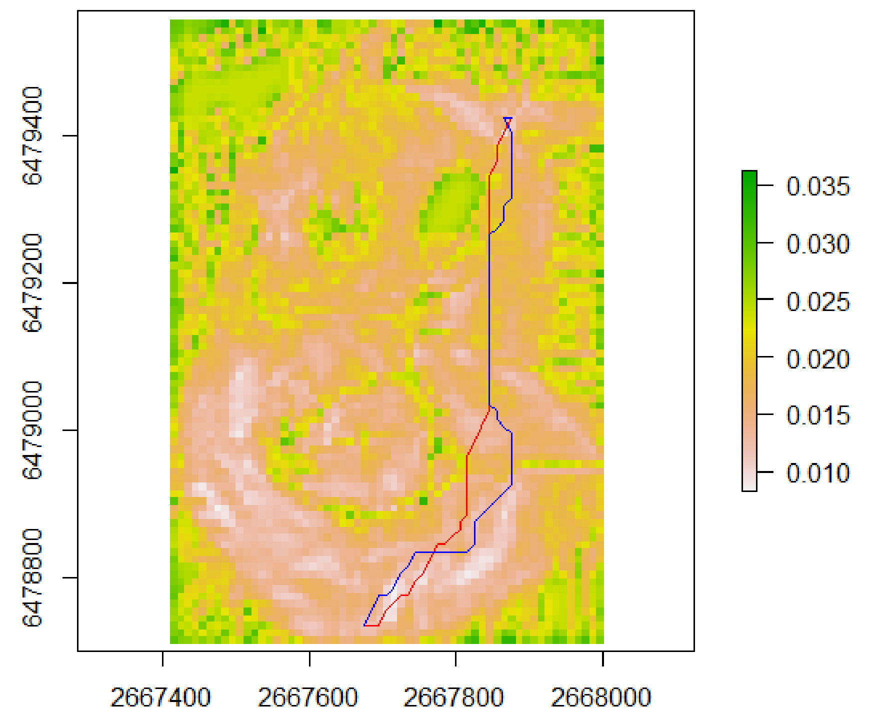
```
cs <- create_slope_cs(dem = r, cost_function = 'tobler', neighbours = 16) %>%
  "*" (create_traversal_cs(dem = r, neighbours = 16))
```

```
loc1 = cbind(2667876, 6479424)
loc1 = sp::SpatialPoints(loc1)

loc2 = cbind(2667677, 6478737)
loc2 = sp::SpatialPoints(loc2)

lcp <- create_lcp(cost_surface = cs, origin = loc1, destination = loc2, directional = FALSE)

plot(raster(cs))
plot(lcp[[1]], add = T, col = "red")
plot(lcp[[2]], add = T, col = "blue")
```

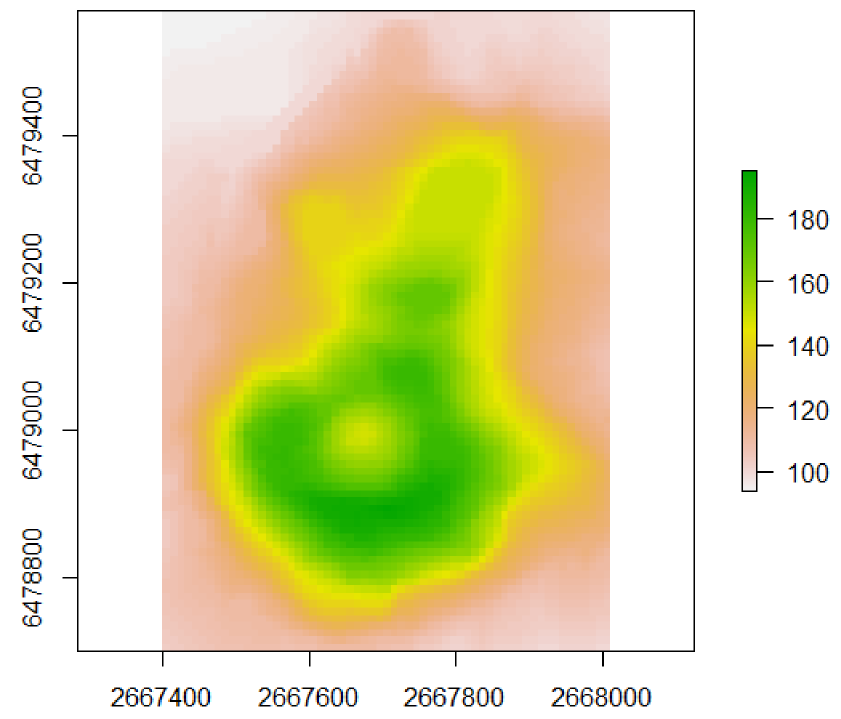


4. Example 3: Least Cost Path Analysis (Slope, Traversal Across Slope, and Landscape Feature Attraction)

```
library(rgdal)
library(rgeos)
library(sp)
library(raster)
library(gdistance)
library(leastcostpath)
```

```
r <- raster::raster(system.file('external/maungawhau.grd', package = 'gdistance'))

plot(r)
```



```
feature_loc = cbind(2667652, 6478997)
feature_loc = sp::SpatialPoints(feature_loc)

cs <- create_slope_cs(dem = r, cost_function = 'tobler', neighbours = 16) %>%
  "*" (create_traversal_cs(dem = r, neighbours = 16)) %>%
  "*" (create_feature_attraction(dem = r, locs = feature_loc, max_attraction = 5000, distance
= 200))
```

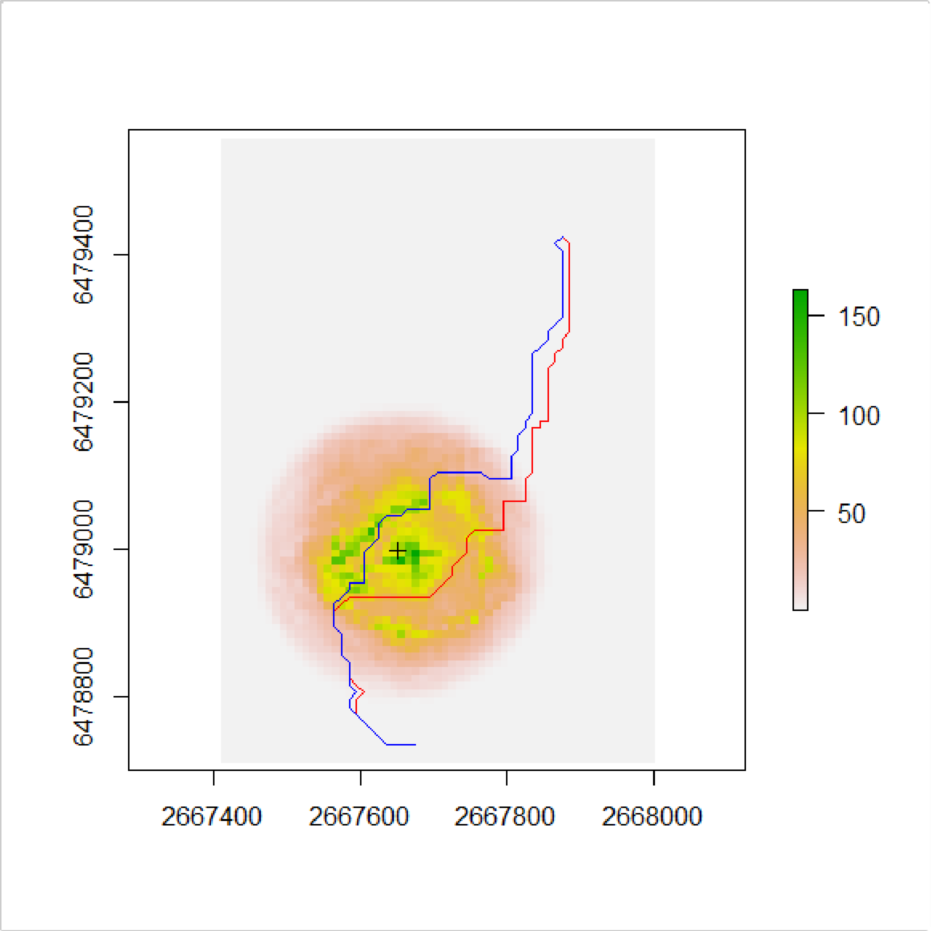
```
## note: create_feature_attraction expects planar coordinates
```

```
loc1 = cbind(2667876, 6479424)
loc1 = sp::SpatialPoints(loc1)

loc2 = cbind(2667677, 6478737)
loc2 = sp::SpatialPoints(loc2)

lcp <- create_lcp(cost_surface = cs, origin = loc1, destination = loc2, directional = FALSE)

plot(raster(cs))
plot(feature_loc, add = T, col = "black")
plot(lcp[[1]], add = T, col = "red")
plot(lcp[[2]], add = T, col = "blue")
```

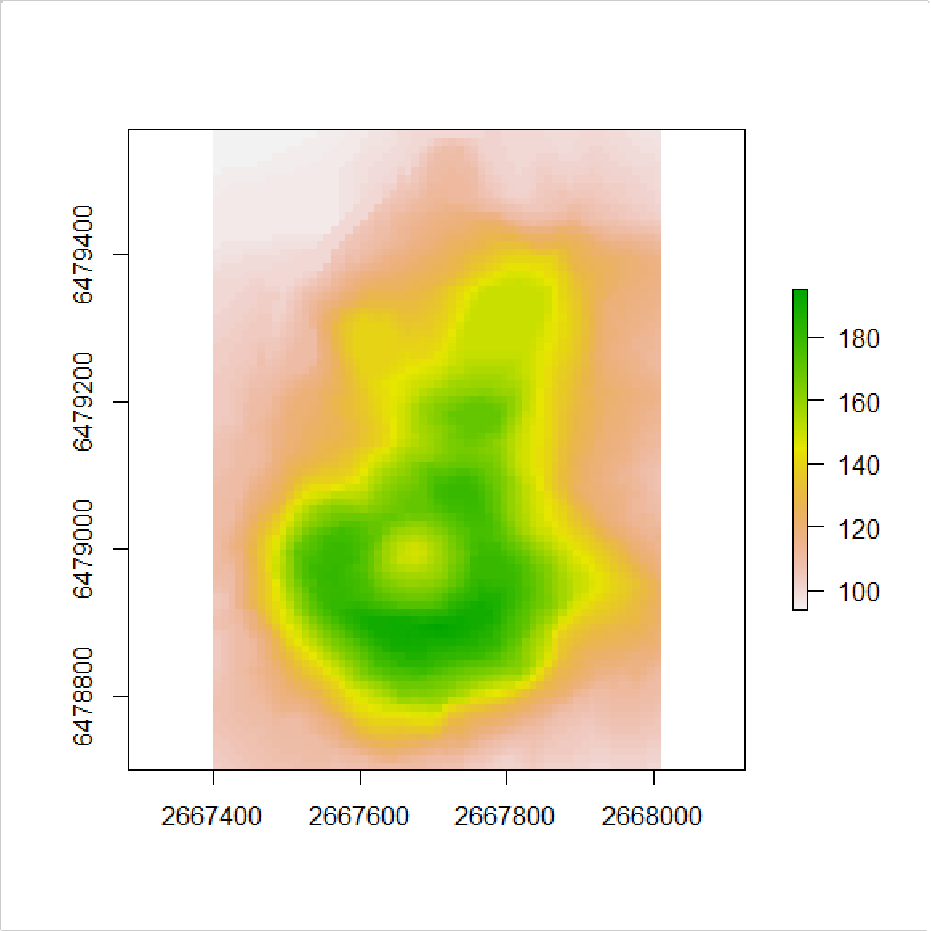


5. Example 4: Least Cost Path Corridor (Slope, Traversal Across Slope, and Landscape Feature Attraction)

```
library(rgdal)
library(rgeos)
library(sp)
library(raster)
library(gdistance)
library(leastcostpath)
```

```
r <- raster::raster(system.file('external/maungawhau.grd', package = 'gdistance'))

plot(r)
```



```
feature_loc = cbind(2667652, 6478997)
feature_loc = sp::SpatialPoints(feature_loc)

cs <- create_slope_cs(dem = r, cost_function = 'tobler', neighbours = 16) %>%
  "*" (create_traversal_cs(dem = r, neighbours = 16)) %>%
  "*" (create_feature_attraction(dem = r, locs = feature_loc, max_attraction = 100, distance =
50))
```

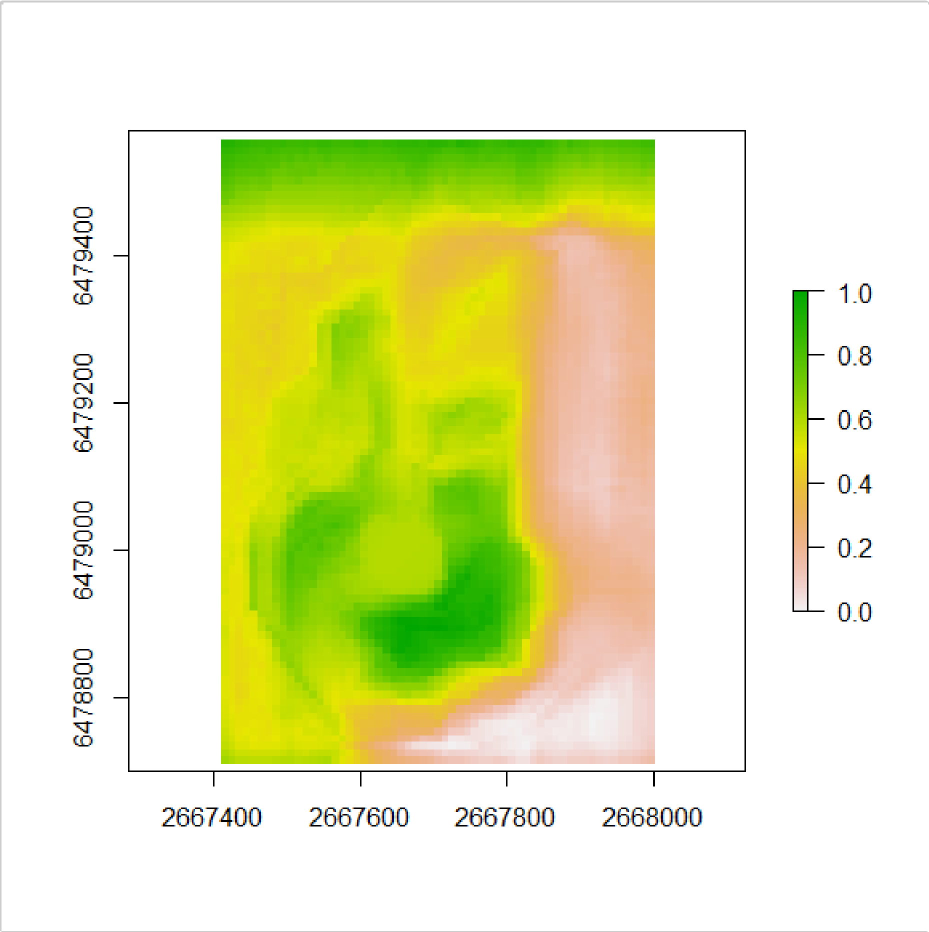
```
## note: create_feature_attraction expects planar coordinates
```

```
loc1 = cbind(2667876, 6479424)
loc1 = sp::SpatialPoints(loc1)

loc2 = cbind(2667677, 6478737)
loc2 = sp::SpatialPoints(loc2)

cc <- create_cost_corridor(cs, loc1, loc2)

plot(cc)
```


6479400
6479200
6479000
6478800
2667400
2667600
2667800
2668000
1.0
0.8
0.6
0.4
0.2
0.0