Supplemental Information

Nicolas Gauthier

Last knit on: 28 January, 2019

Contents

Terrain Analysis: Least Cost Distances 1

Digital Elevation Model 2

Hiking Speeds 2

Least-cost Distances 3

```
library(raster)
library(tidyverse)
library(gdistance)
library(tidygraph)
library(ggraph)
library(sf)
library(smoothr)
```

Terrain Analysis: Least Cost Distances

Next we calculate the impact of rugged terrain on the potential flow of people and information between the sites in the SWSN database. For this we need to use the SRTM digital elevation model. From the height data in the DEM, we calculate slope, accounting for cognitive biases people have when assessing the steepness of high slopes (people tend to exaggerate slopes above a certain threshold). From this map of perceived slope, we calculate "perceived walking speeds", using Tobler's hiking function. We uses these perceived walking speeds as a measure of the perceived, symmetric costs of traveling between two locations on the landscape, and from these estimate the least cost paths from every site to every site. From there, we extract the 15 nearest neighbors of each site in each time period, and use this nearest neighbor network as a spatial network, from which to estimate new travel costs (limited to paths along this nearest neighbor network).

We focus on the perceived time costs of travel. People will consistently overestimate slopes, so the perceived cognitive costs of moving across the landscape will be greater. In rugged, mountainous terrain, slope is the key factor in influencing ease of travel. We focus specifically on paths, or the physical routes that people traveled. A good path connecting two places has to optimize for ease of travel in both

directions. The resulting least-cost network will be used as a proxy measure for the constraints on moving both people and bulk goods across the landscape, and thus the topographic affordances for social exchange.

Digital Elevation Model

First import the elevation dataset, crop to the study area, and resample.

```
bbox <- extent(c(-113, -107, 31, 37.5)) # define our study area

elev <- raster('~/SRTM_W_250m_TIF/SRTM_W_250m.tif') %>% # import the SRTM DEM
  crop(bbox) %>% # crop to study area
  aggregate(fact = 2) # aggregate to ~500m resolution
```

Hiking Speeds

Now we use the package *gdistance* to calculate the time it would take for a hiker to traverse this landscape on foot. Tobler's hiking function allows us to convert terrain slope to expected walking speed. We modify this function to account for cognitive bias in the perception of slope, which causes people to overestimate the steepness of steeper slopes.

The maximum walking speed of 5 assumes on path movement.

```
tobler_adjusted <- function(x) {
    4 * exp(-3.5 * 2.15 * x)/3.6  # 3.6 turns km/h into m/s
}
```

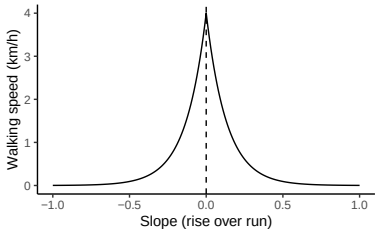

Figure 1: Tobler's hiking function

This package uses a transition matrix approach, whereby we work with matrices that contain the costs or transmissiveness of travel from each cell to its immediate 16 neighbor cells. This allows for a sparse matrix representation, because there is no possible connection to non neighboring cells, which considerably reduces the computational burden.

We're representing paths, not necessarily routes, and so focus on symmetric costs. This is an additional modification to tobler.

Calculate the absolute difference in elevation between each cell and its 16 neighbors.

```
altDiff <- function(x) {
    abs(x[2] - x[1])
}
hd <- transition(elev, altDiff, 16, symm = TRUE)
```

Divide the height differences by the horizontal distances between cells, resulting in slopes.

```
slope_c <- geoCorrection(hd, type = "c")
```

Figure out which cells are adjacent to one another, queen's case.

```
adj <- adjacent(elev, cells = 1:ncell(elev), directions = 16)
```

Use Tobler's hiking function to calculate walking speed from *cognitive* slope.

```
speed_c <- slope_c
speed_c[adj] <- tobler_adjusted(slope_c[adj])
```

Divide by intercell distance again, resulting in the conductance matrix.

```
conductance_c <- geoCorrection(speed_c, type = "c")
```

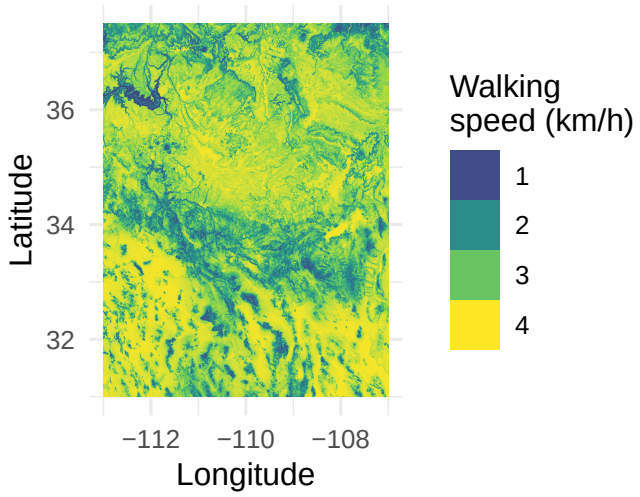

Figure 2: Conductance map.

```
## Saving 5 x 2.5 in image
```

Least-cost Distances

Using the conductance matrix, calculate the full pairwise least cost distance matrix. The points of sites are originally in utm zone 12, so reproject to latlong

```
swsn <- readRDS("output/swsn")
patches <- swsn %N>% as_tibble %>% st_as_sf
```

```
distances <- patches %>%
  st_coordinates %>%
  costDistance(conductance_c, .) %>% # least cost distances
  as.matrix %>%
  `colnames<-`(patches$name) %>%
  as_tibble %>%
  mutate(from_patch = patches$name) %>%
  gather(to_patch, distance, -from_patch) %>%
  mutate(distance = distance / 3600)

write_csv(distances, "output/distances.csv")

knitr::knit_exit()
```