**MRes mini project: Task 9:**

*Create a document that lists each of the stages involved in generating and summarising the accessibility surface.*

*Put in a table with headings 'Task', 'R' and 'GEE'. For each task note whether it's possible in R and/or Google Earth Engine (GEE) and make notes of packages, functions we're using to undertake the task.*

**Need to fill in ‘packages’ columns**

**Need to add info as to what is being done**

**1. Setup (to do)**

**2. Download/install packages**

Downloads, installs and loads all packages needed to generate and summarise accessibility surface (including those potentially missing from client computers).

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s) to install** | **Function(s)** | **R/GEE** |
| Download/install packages (all available from CRAN) | sf,  mapview,  googledrive,  osmdata,  ggplot2,  raster,  gdistance,  fasterize,  remotes,  rgdal,  stars,  geojsonio | # Create vector list of packages needed  list.of.packages <-  c(list\_packages)  # Identify which packages aren’t currently installed on client computers and store in object  new.packages <-  list.of.packages[!(list.of.packages  %in% installed.packages()[  ,”Package”]))  # Install any missing packages  if(length(new.packages))  install.packages(new.packages)  # Load all required packages  lapply(list.of.packages, library,  character.only = TRUE) | Executed within R,  Does **not** require GEE |

**3. rgee: Download, install, connect to and initialise rgee package**

Write some info about this process.

More information on this process can be found here:

**Now available from CRAN: Change!**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Download/install rgee package |  | remotes::install\_github("r-spatial/rgee") | Executed within R,  Does **not** require GEE |
| Load rgee package |  | library(rgee) | Executed within R,  Does **not** require GEE |
| Connect to GEE using rgee  (only need to run once) | rgee | ee\_install(rgee) | Executed within R,  **Performed using GEE** |
| Initialise rgee package | rgee,  googledrive | ee\_initialize(drive = TRUE) | Executed within R,  **Performed using GEE** |

**4. Define area of interest (aoi) and create aoi polygon**

Area of interest defined using ‘WGS84’ projection.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Define area of interest (aoi) using coordinates (WGS84 projection)  Maximum and minimum values for LAT/LONG; four coordinates used to make square: (bbxmin, bbxmax, bbymin, bbymax) |  | bbxmin <- co-ordinate\_1\_(WGS84)  bbxmax <- co-ordinate\_2\_(WGS84)  bbymin <- co-ordinate\_3\_(WGS84)  bbymax <- co-ordinate\_4\_(WGS84) | Executed within R,  Does **not** require GEE |
| Transform aoi into polygon | rgee | aoi <-  ee$Geometry$Polygon(  cords = list (c(bbxmin, bbymax),  c(bbxmax, bbymax),  c(bbxmax, bbymin),  c(bbxmin, bbymin))) | Executed within R,  **Performed using GEE** |

**5. Landsat 8 data: Read in Landsat 8 (LS8) Tier 1 dataset and filter by area of interest and collection date**

Not sure how data was accessed?

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Read in Landsat 8 (LS8) Tier 1 dataset | rgee | ls8 <-  ee$ImageCollection(landsat\_data) | Executed within R,  **Performed using GEE** |
| Filter LS8 data by aoi |  | spatialFiltered <-  ls8$filterBounds(aoi) | Executed within R,  **Performed using GEE** |
| Filter LS8 data by collection date |  | temporalFiltered <-  spatialFiltered$filterDate(  date(s)\_of\_interest) | Executed within R,  **Performed using GEE** |

**6. Create and apply a cloud mask to filtered LS8 data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Create cloud mask  (generate function) | rgee | ndvilowcloud <-  function(image) {  # Get a cloud score in [0, 100]  cloud <-  ee$Algorithims$landset$simpleCloudScore(  image)$select(‘cloud’)  # Create a mask of cloudy pixels from an arbitrary threshold (20%)  mask <- cloud$lte(20)  # Compute NDVI using inbuilt functions  nvdi <-  image$normalizedDifference(  c(‘B5’, ‘B4’))$rename(‘NDVI’)  # Return the masked image with an NDVI band  image$addBands(ndvi)$updateMask(mask)  } | Executed within R,  **Performed using GEE** |
| Apply cloud mask |  | cloudlessNDVI =  temporalFiltered$map(ndvilowcloud) | Executed within R,  Does **not** require GEE |

**7. Normalised difference vegetation index (NDVI): Calculate median NDVI per pixel and clip to area of interest (and check output)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Calculate median NDVI per pixel |  | medianimage <-  cloudlessNDVI$median()$select(‘NDVI’) | Executed within R,  **Performed using GEE** |
| Clip to aoi |  | medNDVIaoi <-  medianimage$clip(aoi) | Executed within R,  **Performed using GEE** |
| View output |  | Map$centerObject(aoi)  Map$addLayer(  eeObject = medNDVIaoi,  viaParam = list(min = -1,  max = 1,  palette = c(‘blue’,  ‘white’,  ‘green’)),  name = “Median NDVI”) | Executed within R,  **Performed using GEE** |

Should produce map of aoi

**8. Convert image to raster and download it using Google Drive (drive) or Google Cloud Storage (GCS)**

These data are saved as an image within GEE.

Convert data to raster and download using drive or GCS.

Raster is stored as .tif file in a temporary local folder, which can then be written to our data folder.

More information on this process can be found here: <https://r-spatial.github.io/rgee/reference/ee_as_raster.html>

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Convert data to raster (within GEE) and download/store in temporary folder |  | med\_ndvi <-  ee\_as\_raster(  image = medNDVIaoi,  region = aoi,  scale = 30  via = ‘drive’) | Executed within R,  **Performed using GEE** |
| Write raster (.tif) to local folder |  | writeRaster(  med\_ndvi,  “local\_filepath”,  Format = ‘GTiff”,  Overwrite = TRUE) | Executed within R,  **Performed using GEE** |

**9. OpenStreetMap (OSM) data: Download OSM road network data for our aoi within R.**

To detail travel speeds within the area of interest, an open source road network which is publicly compiled and hosted by OpenStreetMaps (OSM) is used.

Hosted here: [www.openstreetmap.org](http://www.openstreetmap.org)

OSM road data from the aoi can be directly downloaded within R.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Define bounding box |  | aoi\_bbox <-  c(bbxmin,  bbymin,  bbxmax,  bbymax) | Executed within R,  Does **not** require GEE |
| Obtain road data | osmdata | q <-  opq(bbox = aoi\_bbox) %>%  add\_osm\_feature(key = ‘highway’) %>%  osmdata\_sf() | Executed within R,  Does **not** require GEE |
| Plot road data to check | ggplot  osmdata | ggplot(q$osm\_lines) + geom\_sf() |  |

**Should produce map of road data**

**10a. Assign speeds to NDVI pixels by walking speed**

Need to document how these speeds are determined and how they vary by season and location.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Temporarily read in NDVIexample data from folder (downloaded from GEE) and save as raster object  This may be replaced with med\_ndvi if rgee continues to be reliable. |  | ndvipath <- “NDVIexample.tif”  ndvi <- raster(ndvipath) | Executed within R,  Does **not** require GEE |
| Reclassify raster so that:  **< 0.35** = impassable (cannot traverse)  **0.35 – 0.6** = 3.5 km p/hour  **0.6 – 0.7** = 2.48 km p/hour  **> 0.7** = 1.49 km p/hour |  | # Generate ndvi\_walk\_kph vector object  ndviwalk\_kph <- c(0.1,  3.5,  2.48,  1.49)  # Convert ndvi\_walk\_kph vector to metres p/second  ndviwalk\_mps <- ndviwalk\_kph/3.6  # Convert to crossing time in seconds (assumes travel along hypotenuse and pixel size is 30 m2)  nvdiwalk\_secs <- 42.43 / ndviwalk\_mps  ## Convert km p/hour to metres p/second using matrix  # Create matrix  ndviwalk\_vec <-  c(-1, 0.35, nvdiwalk\_secs[1],  0.35, 0.6, nvdiwalk\_secs[2],  0.6, 0.7, nvdiwalk\_secs[3],  0.7, 1, nvdiwalk\_secs[4])  ndviwalk\_mat <-  matrix(ndviwalk\_vec,  ncol = 3,  byrow = TRUE)  # Reclassify ndvi raster  ndvi\_assigned <- ndvi  ndvi\_assigned <-  reclassify(ndvi\_assigned,  ndviwalk\_mat) | Executed within R,  Does **not** require GEE |

**10b. Assign speeds by motor vehicle to OSM road data**

Need to document how these speeds are determined and how they vary by season and location.

For primary and major roads, national speed limits are used as a maximum.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Create dataframe (to be converted to raster) |  | # Create road-type vector  road\_vector <- c(“primary”,  “secondary”,  “motorway”,  “trunk”)  # Not really sure of this step?  q$osm\_lines$motorspeedkph <-  ifelse(q$osm\_lines$highway %in%  road\_vector, 80, 20)  # Convert to metres p/second  q$osm\_lines$motorspeedmps <-  q$osm\_lines$motorspeedkph / 3.6  # Assume a 30 m resolution cell  q$osm\_lines$time\_secs <-  42.43 / q$osm\_lines$motorspeedmps | Executed within R,  Does **not** require GEE |
| Convert to raster, matching the NDVI raster resolution and extent | fasterise | # fasterise function only works with polygons, so a road buffer of ~30 m is added  roads.poly <-  st\_buffer(q$osm\_line, 0.00015)  # Convert to raster  osm\_road\_raster <-  fasterise(roads\_poly,  ndvi\_assigned,  “time\_secs”,  fun = ‘min’) | Executed within R,  Does **not** require GEE |

**11. Merge ‘NVDI’ raster (ndvi\_assigned) and road-data raster (osm\_road\_raster)**

This step will merge data for off-road and on-road travel to create one cohesive friction surface.

In areas where off-road and on-road cells overlap, road values will be retained as these will be associated with the lowest cost (i.e., quickest speed).

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Merge ndvi\_assigned and osm\_road\_raster. Maintain the minimum value, i.e., the quickest cell crossing time. |  | friction\_surface\_motor <-  mosaic(osm\_road\_raster,  ndvi\_assigned,  fun = min,  tolerance = 1) | Executed within R,  Does **not** require GEE |
| Save friction\_surface\_motor raster as .tif file |  | writeraster(friction\_surface\_motor,  “local\_filepath”,  format = “GTiff”,  overwrite = TRUE) | Executed within R,  Does **not** require GEE |

**11. Read in health facility location data**

Will be revised in future to use AfriMapR?

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Read in health facility data and store as object |  | healthfac <-  st\_read(“health\_facility\_data.shp”) | Executed within R,  Does **not** require GEE |

**12. Calculate shortest paths**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Calculate transition matrix |  | trans\_motor <-  transition(friction\_surface\_motor,  transitionFunction =  function(x){1/mean(x)},  directions = 8) | Executed within R,  Does **not** require GEE |
| Calculate cumulative cost |  | leastcost\_motor <-  accost(trans\_motor,  as\_Spatial(healthfac)) | Executed within R,  Does **not** require GEE |
| Save leastcost\_motor raster as .tif file |  | writeRaster(leastcost\_motor,  “local\_filepath”,  format = “GTiff”,  overwrite = TRUE) | Executed within R,  Does **not** require GEE |

**12. Create plot to visualise leastcost\_motor raster data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
| Store leastcost\_motor as dataframe |  | lcm\_df <-  as.data.frame(leastocost\_motor,  xy = TRUE) |  |
| Add ‘mins’ column by dividing ‘Layer’ column (shortest path time in seconds) by 60 |  | Lcm\_df$mins <-  lcm\_df$Layer / 60 |  |
| Create plot to visualise raster data | ggplot | ggplot()+  geom\_raster(  data = lcm\_df,  aes(x = x,  y = y,  fill = cut(mins,  c(0,30,  60, 120  180, 240,  300,  max(mins)))))+  scale\_fill\_brewer(  palette = “YlGnBu”)+  geom\_sf(  data = q$osm\_lines,  colour = “darkgrey”,  alpha = 0.3)+  geom\_sf(  data = healthfac,  size = 2,  colour = “red”)+  guides(fill= guide\_legend(  title= “Time (mins)”)) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Package(s)** | **Function(s)** | **R/GEE** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |