**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.

**1. Setup (to do)**

**2. Download/install packages**

Downloads and installs all packages needed to generate and summarise accessibility surface.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |  | Executed within R,  Does **not** require GEE |

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

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) |  |  | Executed within R,  Does **not** require GEE |
| Transform aoi into polygon | rgee | 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 | ee$ImageCollection(landsat\_data) | Executed within R,  **Performed using GEE** |
| Filter LS8 data by aoi |  | ls8$filterBounds(aoi) | Executed within R,  **Performed using GEE** |
| Filter LS8 data by collection date |  | 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**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |  | medianimage <-  medianimage$clip(aoi) | Executed within R,  **Performed using GEE** |

**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\_file\_path”,  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** |
|  |  |  | Executed within R,  Does **not** require GEE |
|  |  |  | Executed within R,  Does **not** require GEE |