Physio-climatic characteristic dataset for India

Dataset v1.0

Ankit Deshmukh & Riddhi Singh

February 28, 2020

Contents

1 Introduction **6**

2 Catchment delineation **7**
2.1 Running *arcpy* outside ArcGIS environment 8
2.2 Catchment selection for database 8

3 Physio-climatic dataset **10**
3.1 Climate 10
3.1.1 Precipitation 10
3.1.2 Temperature 10
3.1.3 Evapotranspiration 10
3.2 Geology 11
3.3 Hydrology 12
3.3.1 Setting-up GRASS 13
3.3.2 Stream order processing 14
3.4 Land cover 15
3.5 Land use 16
3.6 Socioeconomic 17
3.7 Soil 17
3.7.1 Soil fractions 17
3.7.2 Porosity and Permeability 18
3.8 Topography 18

4 Other datasets **20**
4.1 Discharge dataset 20
4.2 Water quality dataset 20

5 Code blocks for data processing (Listenings) **21**

List of Figures

1 Basic approach of data extraction 6
2 27 STRM digital elevation model (DEM) tiles are used for watershed delineation. 7
3 Boundaries for selected catchment are show in the figure. Each circle represents the pour point of a catchment. Colors of pour point and corresponding catchment boundary are same. Histogram represents the area's of catchments. The color gradient sky blue to red represent increasing area of catchments. 9
4 Daily precipitation layer rasterize then shapefile is overlapped to extract the values of precipitation 11
5 Rasterised temperature overlapped with shapefiles to extract the daily temperature. 12
6 UTM projection zones used for India 13
7 Extraction of stream order from raster. 14
8 Dataset for land cover 15
9 Dataset for land use 17
10 Porosity map for India 18
11 Topographic index for a catchment. 19

List of Tables

1 The land cover categories exists in the dataset. 16
2 Land use categories and the corresponding pixel values in the dataset. . . 16
3 Water quality indices are available in the dataset. 20

Listings

1 Catchment delineation process with arcpy 21
2 Plotting: Gauges stations boundaries selected for database. Histogram shows variation of catchment area. 23
3 Downscale gridded precipitation data to catchment scale 24
4 Convert daily precipitation data to monthly data 26
5 Compute precipitation based characteristics 27
6 Downscale gridded temperature data to catchment scale 30
7 Convert daily temperature data to monthly data 32
8 Compute temperature based characteristics 33
9 A function to compute leap year 35
10 A function to compute solar radiation 35
11 Compute evapotranspiration using Hamon equation 36
12 Compute evapo transpiration using hargreaves equation 37
13 Code I use to get geological characteristics 38
14 To process DEM to get stream order and topographical properties 38
15 Extract hydrological characteristics using Grass GIS 42
16 Land cover extraction 43
17 Land use extraction 45
18 Socio economic extracted with the following steps 46
19 Various soil layers for extraction 48
20 Porosity and permeability dataset extraction 50
21 Topographical characteristics extraction 51

1 Introduction

A catchment is the most fundamental hydrological unit of naturally flowing water in the land. Each catchment is unique and has specific characteristics. We found there is no such dataset available by which we can characterize the catchment completely. For the United States, a catchment characteristics dataset Falcone [Falcone, 2011] provides information of more than 6000 catchments. Falcone data used in numerous studies and proven to be very useful. We are unable to find such kind of database for India. This dataset will be valuable for the future research that require physio-climatic characteristics for India catchments. We try to keep dataset processing and collection steps as transparent as possible. Methods and algorithms are used to develop this dataset, are carefully presented in "./Documentation/DatabaseDocumentation.pdf" document.

The dataset is called *Physio-climatic characteristics for India*[1]. We present the dataset into the following sub-categories,(1) Climate, (2) Geology, (3) Hydrologic, (4) Land cover, (5) Land use, (6) Socioeconomic, and (7) Topographic. Each sub-category has several characteristics and the summary of each characteristic is shown in the Index sheet of the dataset. Index sheet has fields like Name, description, method of data preparation, data units, data source citation. We want to add as many characteristics we are able to find in the Falcone dataset. Our basic approach to get the data is to crop a raster with shape file and then take a weighted mean of the characteristics. Basic working of data extraction is shown in the Figure 1.

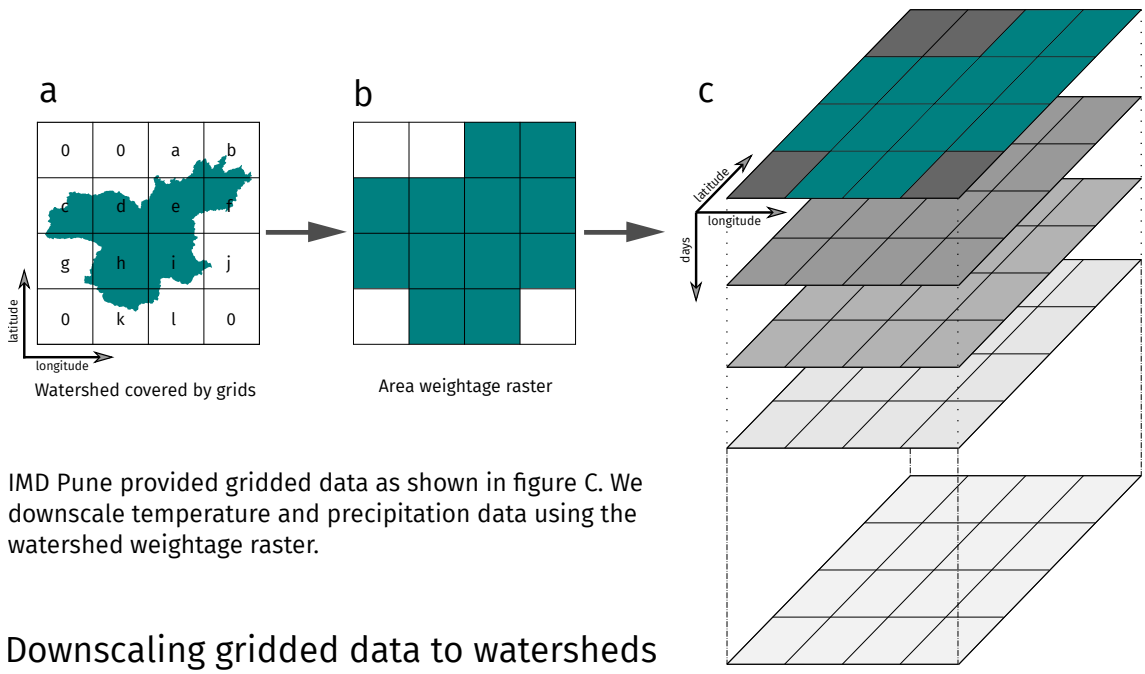

Figure 1: *Basic approach of data extraction*

This documentation also explains how we extract each characteristic with working code blocks. We primarily use **R** and **python** for data processing. All the code blocks can be accessed from the section 5.

[1]Citation: Deshmukh, A., Singh, R. (2020). PCCDI: Physio-Climatic Characteristics Dataset for India. *Unpublished*, 00(0), 00–00.

2 Catchment delineation

Building a database required the hydrological boundaries of water(i.e. catchments). We delineate catchment with pour point as WRIS India gauge stations. As we are unable to find any source of catchments shapefiles. WRIS India portal has 953 hand-operated (HO) gauging stations information. We are able to find 950 catchment information (location and gauge station status) to delineate catchments. Most of the gauge station has HO activity which includes the measurement of gauge discharge and sediment. We use 27 90m digital elevation model (DEM) geoTIFF for catchment delineation process, shown in the Figure 2.

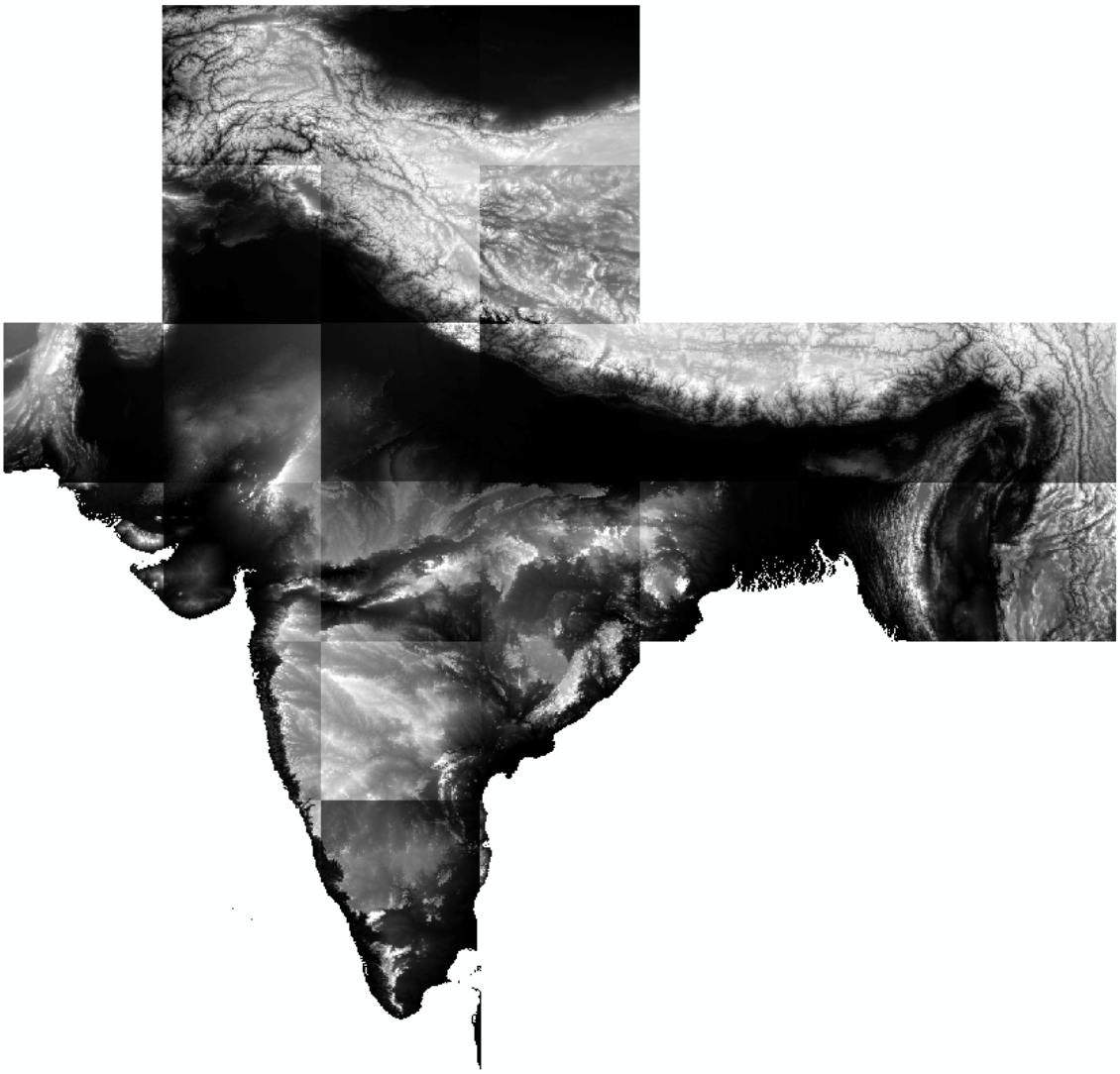

Figure 2: *27 STRM digital elevation model (DEM) tiles are used for watershed delineation.*

We perform following operation to delineate the catchment (a working code presented in the Listening 1):

- Remove sink and holes in DEM
- Calculate flow direction raster
- Calculate flow accumulation raster

- Project pour point and using a buffer zone around to identify highest flow accumulation point
- Calculate upslope-area raster
- Convert upslope-area raster to shape files and calculate catchment area

We format the names of delineation catchments, discharge data files, and water quality data files the same. We remove underscore and standardize the catchment names i.e. A. K_Bridge → AKBridge. All the information is kept in the dataset sheet *PourPoint950*.

2.1 Running *arcpy* outside ArcGIS environment

Python scripting is essential for repetitive works like delineating catchment for several pour points. We use `arcpy`, python library of ArcGIS to automate the catchment delineation process. We use `anaconda` python-2.x distribution, `anaconda` also incorporates `spyder`, a Matlab like python GUI, which we use to write the python scripts for catchment delineation and database manipulation.
System Requirement for Windows operating system.

- ArcGIS installation (We use v10.3)
- Anaconda 32-bit installer

Installation automatic add the variables into system path. Copy all files from your ArcGIS directory `C:\Python27\ArcGISx6410.1 \Lib\site-packages` to Anaconda site package directory i.g. `C:\Anaconda\Libs\site-packages`. Go to Anadonda root folder i.g. `C:\Anaconda` find the `python.exe` and `pythonw.exe` run one by one as administrator. Now you should be able to load `arcpy` in your python console. To check this go to spyder console and type `import arcpy`, if you didn't get any error message, you successfully embedded `arcpy` to anaconda python distribution.

If the above process doesn't work still you can run a python script (*.py) inside ArcMap python console. Go to : Geoprocessing → Python → (Right Click) load script.

2.2 Catchment selection for database

We delineate 950 catchments but we found a lot of them are overlapping. As we select a pour point in the downstream it will delineate a large upslope area and overlap the catchments delineated with the pour points situated on upstream of the same stream. We delineate the catchments with an accumulation threshold of 2000. WRIS does not provide shapefiles of catchment but gives the area of the catchment in the unit of km^2. We found that 126 out of 950 catchments do not have catchment area information from WRIS so we discard these catchments. To verify the accuracy of our delineated catchments first we compute the area bias with WRIS catchment and discard the catchment which has a bias more than $\pm 20\%$. Furthermore, we visually inspect remaining catchments and we finalize 567 catchments. The catchment shapefiles have a field *area* denotes the area of the catchment in km^2. The projection of catchment in geographic coordinate system with the datum WGS1984: `+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0`. The range of catchment in the dataset varies from 40.67 to 804015.10 km^2. Outline of catchments are shown in Figure 3, the gradient in the figure blue to red shows the increasing catchment area. The gauge station location of the catchment are shown with the colored circle, A histogram in the figures shows the area range of catchment in the database.

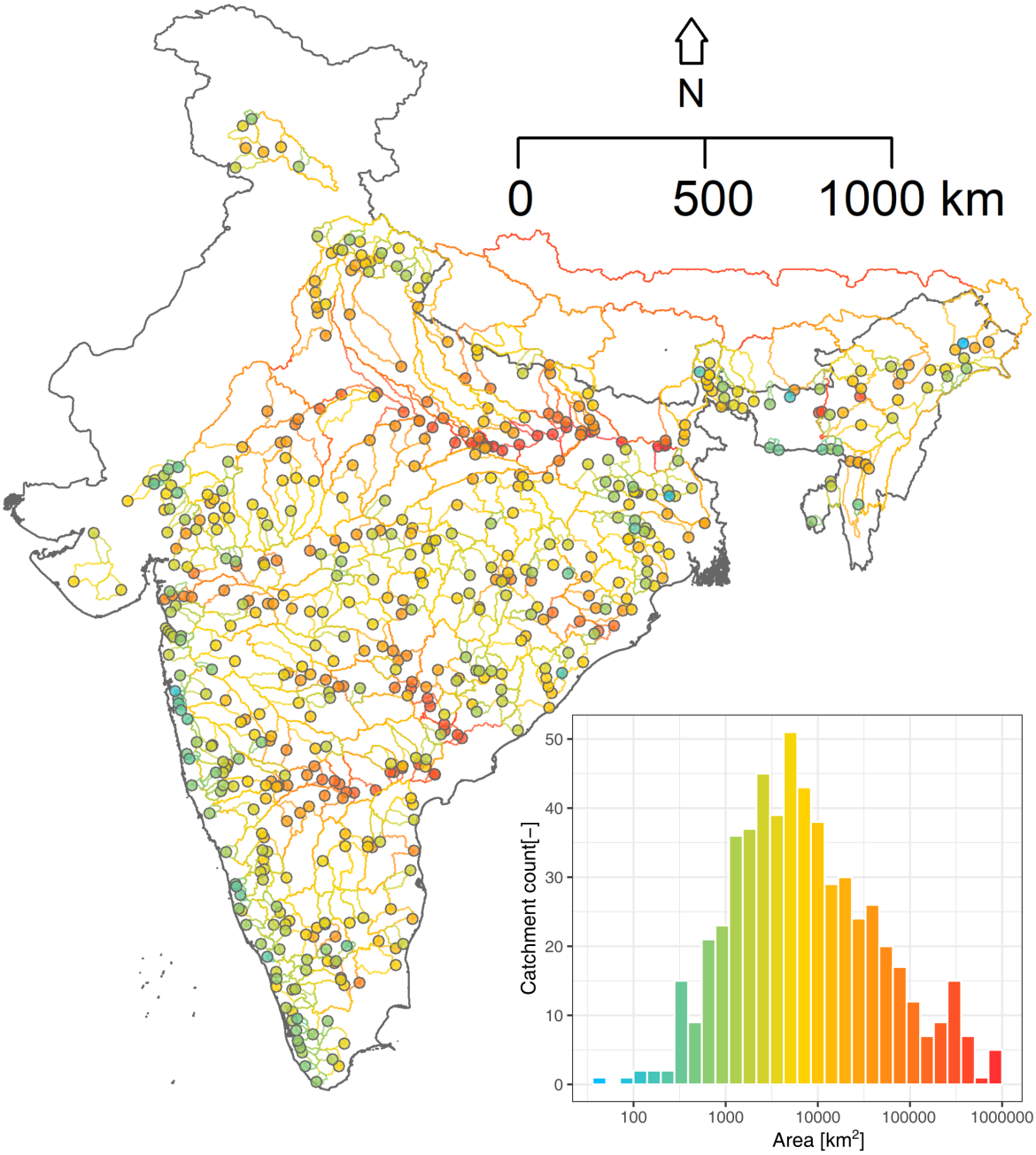

Figure 3: *Boundaries for selected catchment are show in the figure. Each circle represents the pour point of a catchment. Colors of pour point and corresponding catchment boundary are same. Histogram represents the area's of catchments. The color gradient sky blue to red represent increasing area of catchments.*

3 Physio-climatic dataset

Mainly this document deals with the physio-climatic dataset, but we also collected the streamflow datasets for 412 catchments and water quality dataset for 366 gauge stations. The physio-climatic dataset divided into 7 categories named climate, geology, hydrology, land cover, land use, socioeconomic and topography. The climate category subdivided into 3 categories i.e. precipitation, temperature, and Evapotranspiration.

3.1 Climate

To develop climatic characteristics we purchase daily precipitation and temperature data from IMD Pune. The gridded precipitation data has a spatial resolution of 0.25 degree $\times$ 0.25 degree and the data available for January 01, 1901 to 31 December 31, 2015. The temperature dataset has a resolution of 1 degree $\times$ 1 degree. It is much courser dataset then the precipitation data. We have 63 years of temperature data dated form January 01, 1951 to December 31, 2013. In the procession stage, we use a common temporal resolution of both the data(precipitation and temperature) which is January 01, 1951 to December 31, 2013. The unit of precipitation and temperature data is mm/day and °C respectively.

3.1.1 Precipitation

IMD Pune provides the precipitation data in plain text format. Each file contains one year of daily data stacked by the days. Rows and columns denote latitude and longitude respectively. Our approach is to get the slice of each day and convert into a raster layer. Then using the shape a shapefile of each watershed we can crop the overlapped precipitation grid for that day. Rasterize daily precipitation layer with shapefile overlapped data for one catchment is shown in Figure 4. Then we use a weighted average of cropped pixels to get the mean daily precipitation for that catchment.

The process for precipitation data downscaling is shown in the Listening 3. Later we use the Listening 4 to convert monthly mean data for daily data. After that we computed several climatic characteristics form daily and monthly precipitation time series in with following code (Listening 5.

3.1.2 Temperature

The temperature data provided by IMD Pune has grid and ASCII text files for the period of 1951- 2013. The data has daily maximum, daily minimum and daily mean attributes in °C. We follow a similar approach as precipitation data to extract mean daily data for a catchment. The steps taken to obtain daily temperature are shown in Listening 6, see Figure 5 to understand the temperature extraction method. After that we convert the daily to monthly temperature and steps required are shown in the Listening 7.

We use daily and monthly average temperature to get climatic characteristics listed in the database sheet *Index*. A few characteristics are evapotranspiration, which we also compute with daily temperature using Hamon and Hargreaves equation. Discussed in detailed in Section 3.1.3.

3.1.3 Evapotranspiration

We generated daily evapotranspiration (unit mm/day) for catchments using Hamon and Hargreaves equations. The equation uses daily maximum, mean and minimum temperature as an input. It also uses the latitude of the centroid of the catchment to compute

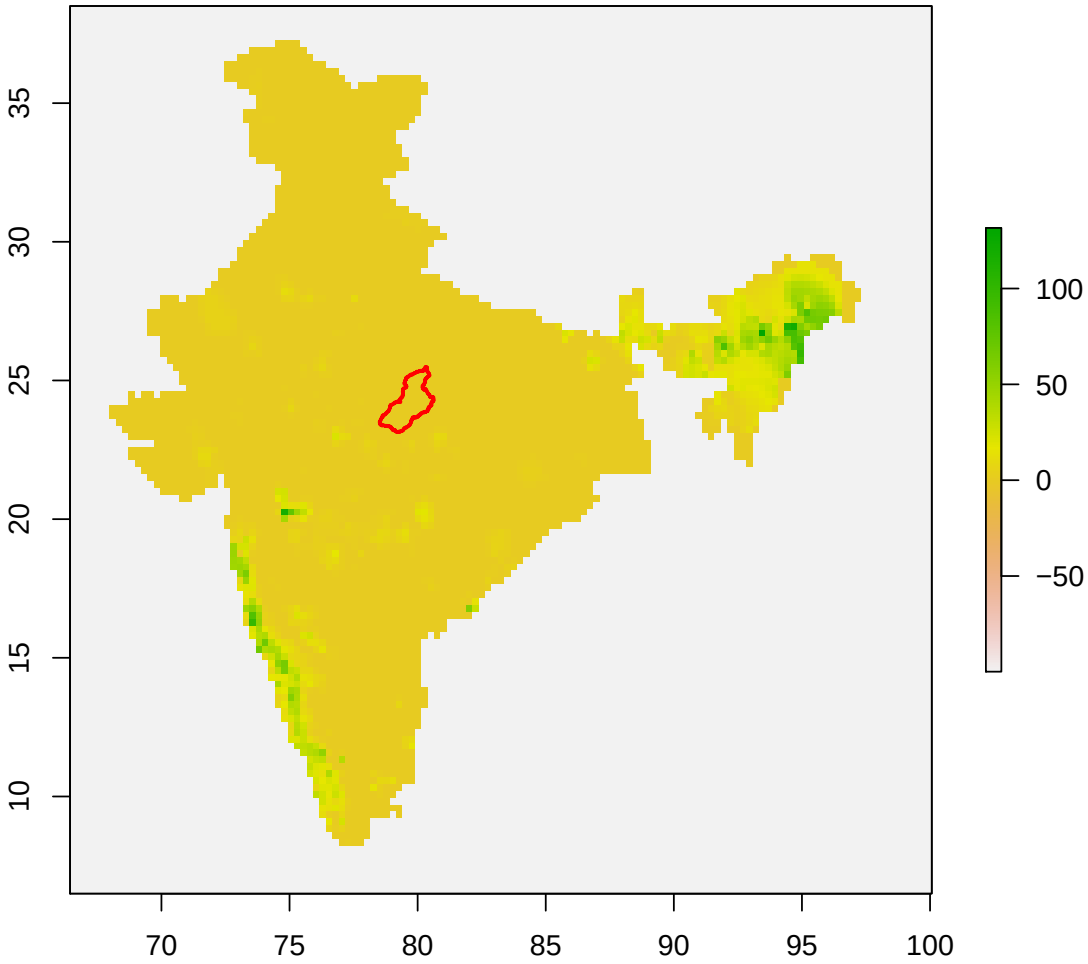

Figure 4: *Daily precipitation layer rasterize then shapefile is overlapped to extract the values of precipitation*

evapotranspiration. The following function we required to compute daily evapotranspiration. **a.** Leap year (Listening 9): to know a year is a leap year. **b.** Solar radiation (Listening 10): to calculate solar radiation based on Julian data and latitude of the centroid of the catchment. **c.** Hamon (Listening 11): to compute daily evapotranspiration with Hamon equation. **d.** Hargreaves (Listening 12): to compute daily evapotranspiration with Hargreaves equation.

3.2 Geology

Geology is the study of earth's surface and the consisting material. To understand the formation of earth's surface we use Global Lithological Map (GLiM) database [Hartmann and Moosdorf, 2012]. Lithology shows us the geochemical, mineralogical, and physical characteristics of rocks. This dataset consists of surface and subsurface lithological formation of earth in 16 categories. The categories are shown in the database documentation Index sheet. We extract % cover each geological formation with a weighted average of

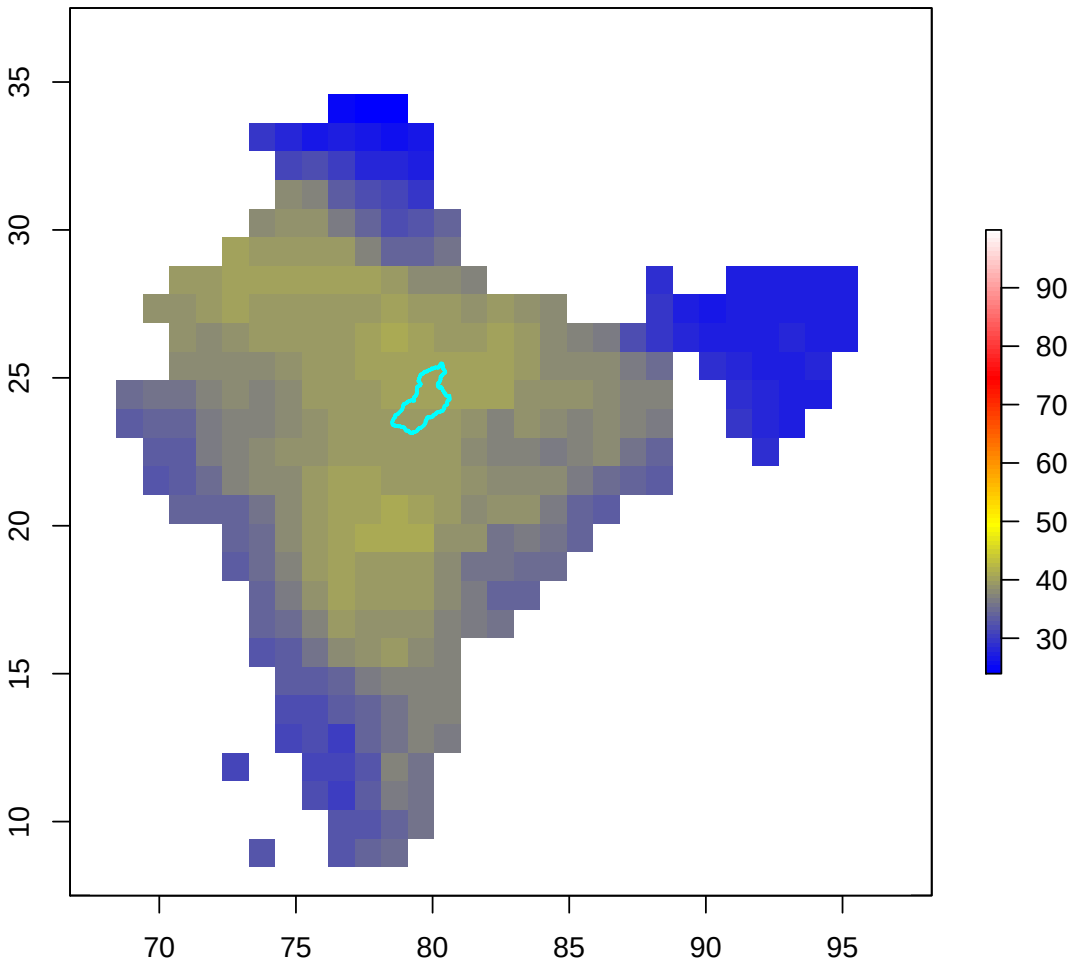

Figure 5: *Rasterised temperature overlapped with shapefiles to extract the daily temperature.*

specific categories for each catchment. The steps taken are shown in the Listening 13

3.3 Hydrology

Catchments' hydrologic characteristics consist of stream density, stream orders, etc. To compute hydrologic characteristics, we need a DEM with a projected coordinate system, initially, it was a geographic coordinate system. We use the Universal Transverse Mercator coordinate system for projection because the cylindrical projection preserves Area and minimizes distortion of distance for small extent. We use 6- UTM zones (UTM-42N or EPSG:32642 to UTM-47N or EPSG:32647). The Figure 6 shows various UTM zones covers India.

We use **R** in most of our dataset development but we found generating streams from DEM are much easier using GRASS (Geographic Resources Analysis Support System). GRASS is an open-source Geographic Information System (GIS) software suite which is a powerful tool for geospatial data management, analysis and image procession. To

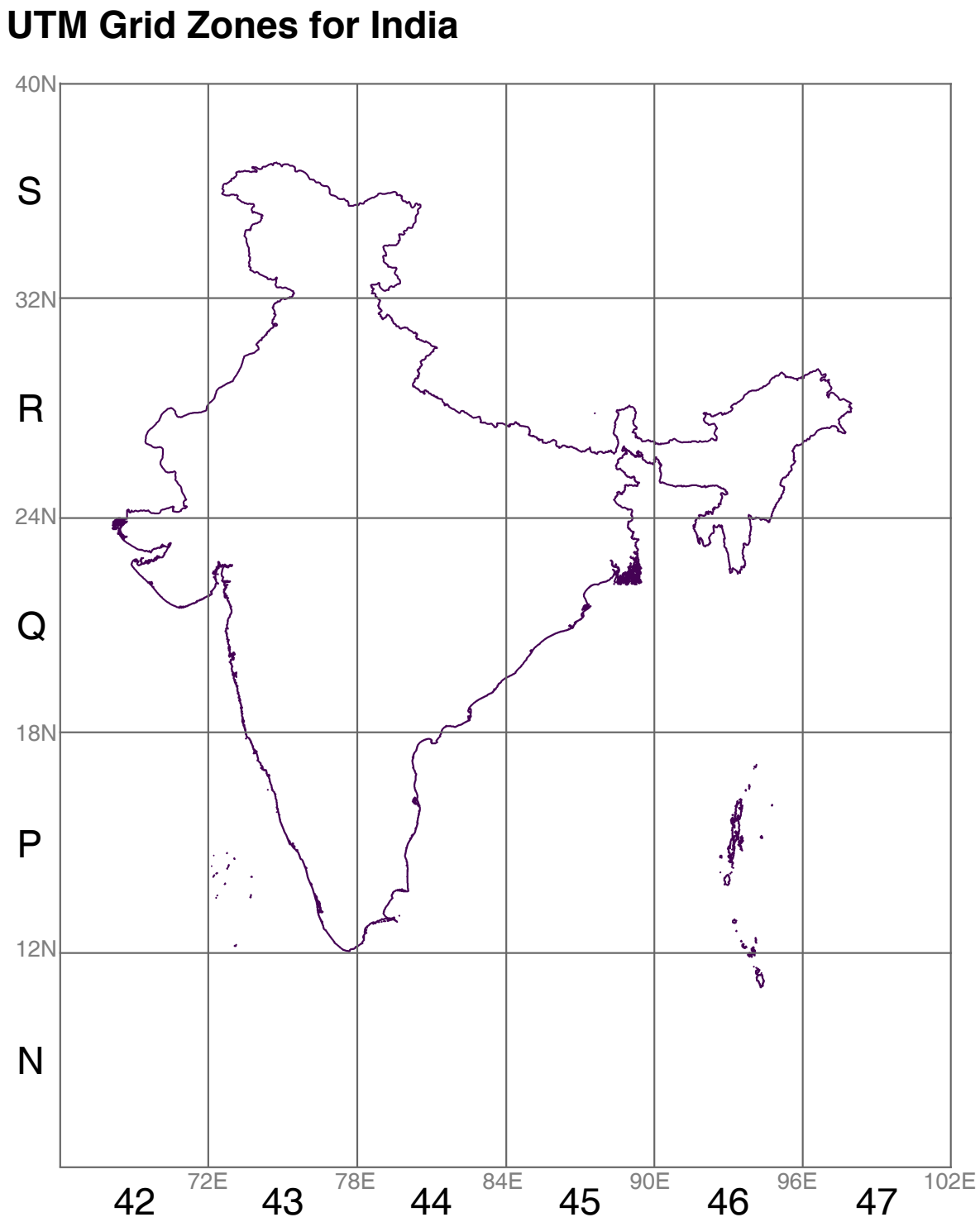

Figure 6: *UTM projection zones used for India*

make analysis reproducible and automated we use rgrass7 package which is an interpreted interface between **GRASS** and **R**. We are explaining the process to run GRASS from R using rgrass7 package in the next Section 3.3.1.

3.3.1 Setting-up GRASS

We process the DEM to compute hydrology characteristics with the following steps shown in the Listening 14. Before running any GRASS module we need to set up the GRASS environment for R. First we need to download GRASS GIS setup. We use GRASS GIS

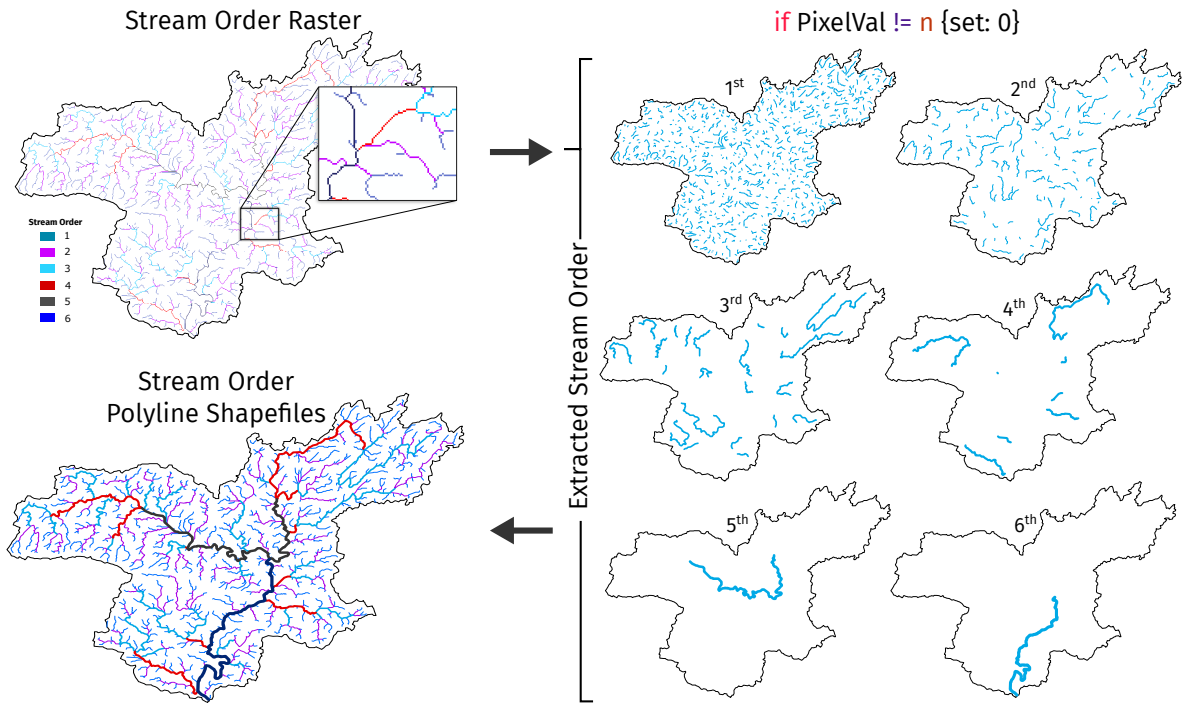

Figure 7: *Extraction of stream order from raster.*

v7.6. GRASS GIS is installed in the location of `C:/Program Files/GRASS GIS 7.6`. Then we need 'rgrass7' package for R to initiate GRASS session. The following piece of code required to set up the GRASS session.

```
# Initiate GRASS session
if(.Platform$OS.type == "windows"){gisbase = "C:/GRASS GIS 7.6"}
initGRASS(gisBase = gisbase,
          home = tempfile(),
          override = TRUE)
```

3.3.2 Stream order processing

With GRASS we perform the following step to get the stream and order shapefiles for each catchment.

1. Load the DEM to conditioning, which required remove sink and fill, using GRASS `r.hydrodem` module.
2. Then conditioned DEM used to compute flow accumulation raster in the module `r.watershed`
3. After that we use `r.stream.extract` to generate stream and direction raster.
4. The stream raster used in `r.stream.order` to develope stream order map.

The working code is presented in the Listening 14. After getting stream order we extract/compute hydrological characteristic using the process presented in the Listening 15

3.4 Land cover

We use the land cover map for south-east Asia from the year of 2000. The source of the dataset is Stibig et al. [2003]. The data is 1km resolution raster dataset. It contains 47 classes where each pixel can hold a value between 0 to 46. Some of the categories like tropical evergreen subtropical evergreen, temperate broad-leaved, tropical montane, tropical semi-evergreen, temperate conifer, etc.(see Figure 8). We clip land cover class data from raster using extract function form **R** and based on a weighted average on pixel we calculate the % of land cover in covered by each category for each catchment.

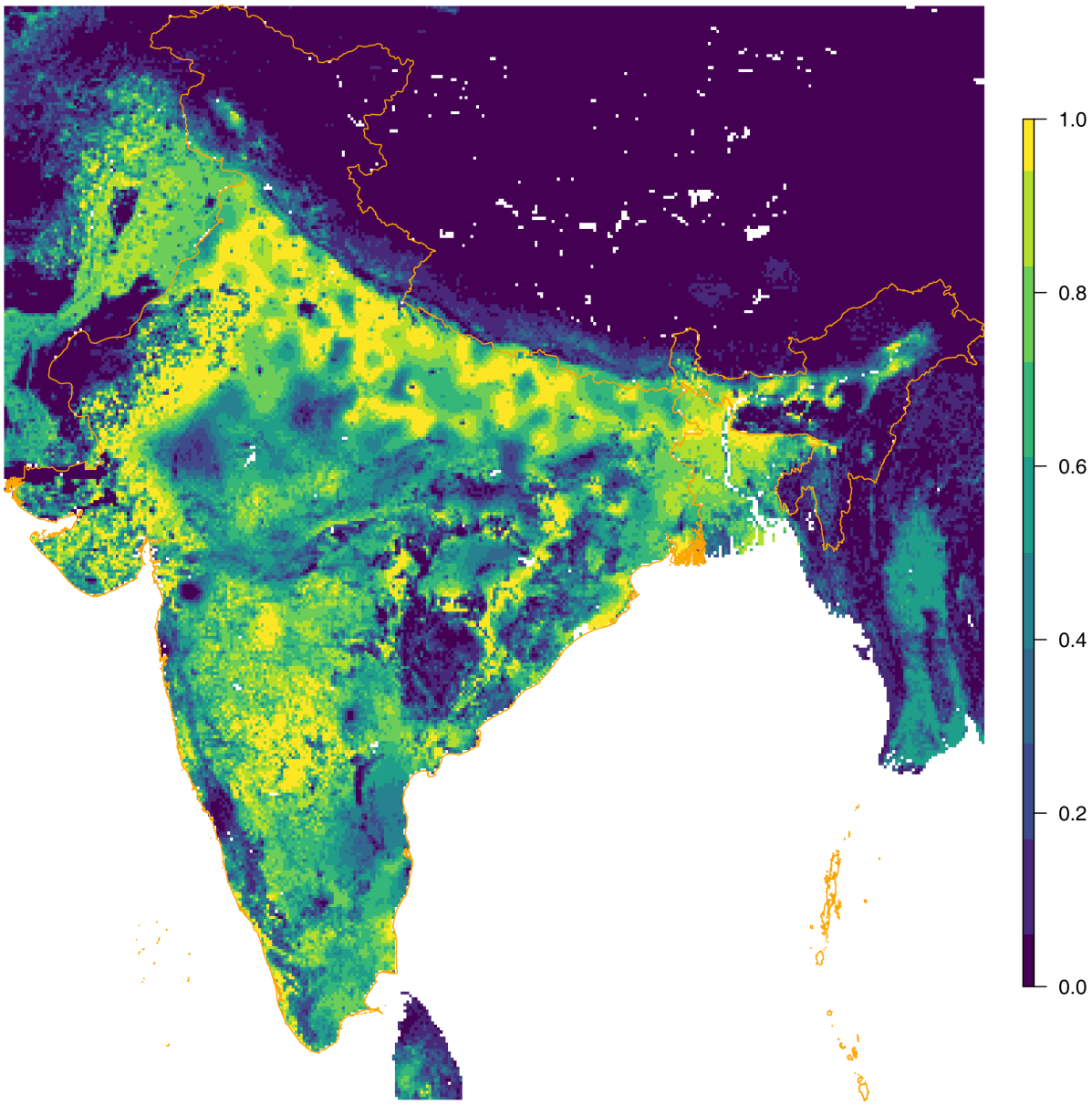

Figure 8: *Dataset for land cover*

The table of land cover categories are shown in the Table **??**.

We add CROP_LAND field in the dataset from Global Croplands in 2000 map. It shows the proportion of each 5 minutes (10km) grid cell land area that is under cropland [Ramankutty et al., 2010]. We use a similar data extraction technique as land cover to get the mean cropland cover for each catchment.

SN	Category	SN	Category
1	Sea	25	Desert Grasslands
2	Tropical Evergreen	26	Alpine Meadow
3	Subtropical Evergreen	27	Alpine Grasslands
4	Temperate Broadleaved	28	Sparse vegetation (cold)
5	Tropical Montane	29	Sparse vegetation (hot)
6	Tropical Semievergreen	30	Gobi
7	Temperate Conifer	31	Desert (cold)
8	Subtropical Conifer	32	Thorn Scrub / Desert (hot)
9	Tropical Moist Deciduous	33	Irrigated Intensive Agriculture
10	Tropical Dry Deciduous	34	Irrigated Agriculture
11	Junipers	35	Slope Agriculture
12	Mangroves	36	Rainfed Agriculture
13	Degraded Forest	37	Current Jhum
14	Dry Woodland	38	Swamp
15	Thorn Forest/Scrub (Northern)	39	Coral reef
16	Thorn Forest/Scrub (Southern)	40	Water Bodies
17	Shrubs	41	Snow
18	Abandoned Jhum	42	Barren
19	Sparse woods	43	Bare Rock
20	Bush	44	Salt Pans
21	Coastal vegetation	45	Mud Flats
22	Savannah	46	Settlement
23	Plain Grasslands	47	No Data
24	Slope Grasslands	48	Cropland field

Table 1: *The land cover categories exists in the dataset.*

3.5 Land use

To develop the land use dataset we use the anthropogenic biome of the world map for the year 2000. The map has 19 categories. Provided data is in raster grid (GeoTIFF) format and each pixel has a value that is associated with a land use category. The land use category is shown below in Table 2.

SN	Pixel	Category	SN	Pixel	Category
1	11	Urban	11	41	Residential rangelands
2	12	Mixed settlements	12	42	Populated rangelands
3	21	Rice villages	13	43	Remote rangelands
4	22	Irrigated villages	14	51	Residential woodlands
5	23	Rainfed villages	15	52	Populated woodlands
6	24	Pastoral villages	16	53	Remote woodlands
7	31	Residential irrigated croplands	17	54	Inhabited treeless and barren lands
8	32	Residential rainfed croplands	18	61	Wild woodlands
9	33	Populated croplands	19	62	Wild treeless and barren lands
10	34	Remote croplands			

Table 2: *Land use categories and the corresponding pixel values in the dataset.*

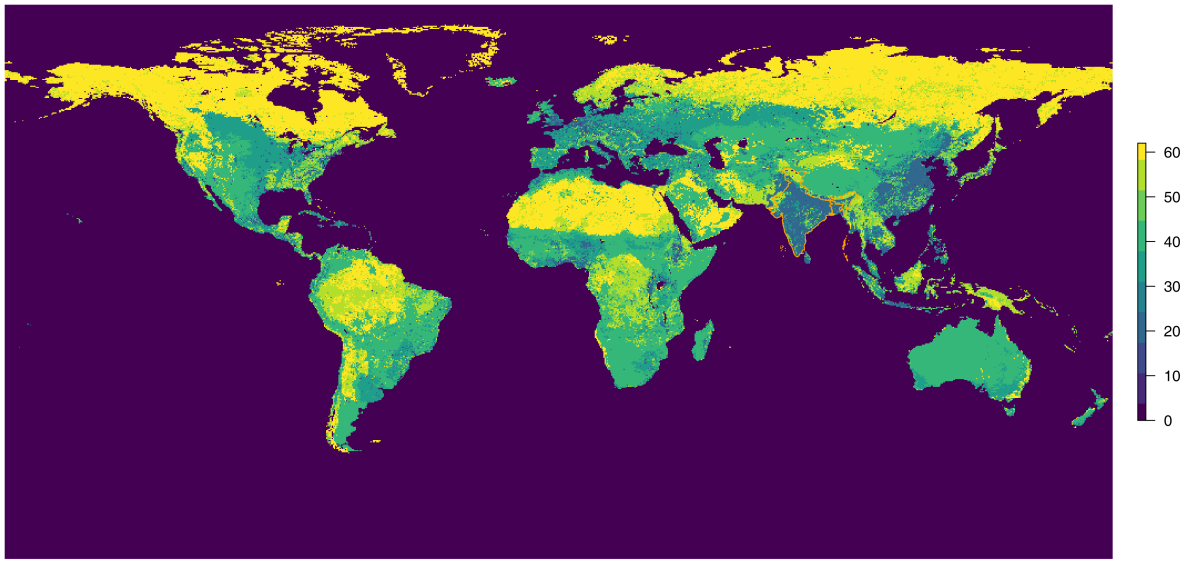

Figure 9: *Dataset for land use*

3.6 Socioeconomic

The socioeconomic dataset is the collection of characteristics represents human influence in the catchment. As socioeconomics is the study of how economic activity affects the social process, we use a socioeconomic dataset to quantify the anthropogenic effect on the catchment. The socioeconomic data set consists of a characteristic such as population density, human footprint index, suitability for agriculture, total nitrogen consumption, total nitrogen consumption, total nitrogen consumption, gross domestic production per capita (purchasing power parity), and human development index. The detail discretion and citation can be found in the database documentation. In the future, we will add other classes of socioeconomic data.

All the data is available in the raster grid format, we process it by cropping the raster with shapefile and the compute weight age average for each pixel to get mean characteristic values for each catchment.

3.7 Soil

We acquire soil characteristics from SoilGrids1km — Global Soil Information Based on Automated Mapping. Soil grid data is available in 6 layers (0-5cm, 5-15cm, 15-30cm, 30-60cm, 60-100cm, 100-200cm) and it consists soil, organic carbon (g kg^{-1}), soil pH, sand, silt and clay fractions (%), bulk density (kg m^{-3}), cation-exchange capacity (cmol+/kg), coarse fragments (%), soil organic carbon stock (t ha^{-1}). Hengl et al. [2014] develop the dataset and can be obtained freely from Soilgrid. We also gather permeability and porosity data form GLobal HYdrogeology MaPS (GLHYMPS) of permeability and porosity [Gleeson et al., 2014]

3.7.1 Soil fractions

We extract soil fraction and chemical characteristics form SoilGrids1km dataset. We use layers of the mean depth of available soil products such as for soil organic carbon we use a layer of soil at 2.5 cm (mean of 0 to 5cm). The dataset provided in the raster grid. We extract with mean values of characteristics with weighted pixel values covered by catchment shapefiles.

3.7.2 Porosity and Permeability

We extract porosity and permeability of the soil, from Gleeson et al. [2014]. It is a global porosity and permeability database available in spatial polygons shapefiles. Each polygon has properties fields of porosity, permeability without permafrost, permeability with permafrost. We consider permeability without permafrost as permeability for our database. We rasterize the polygon layer (shapefiles) with field with porosity and permeability into 2 separate raster gird. Then we crop and extract mean values or characteristics for each catchment. We are demonstrating the polygon layer into Figure 10.

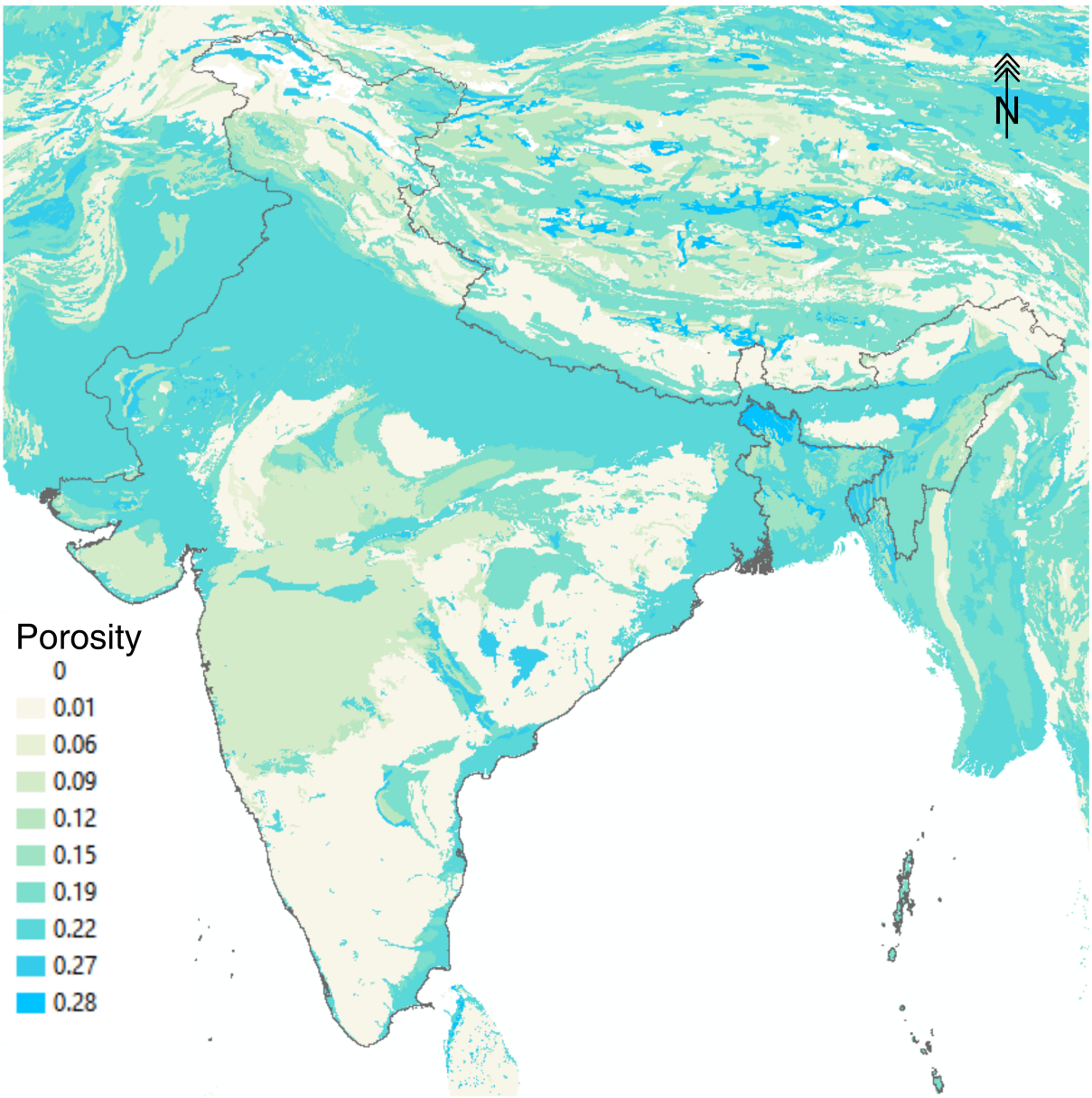

Figure 10: *Porosity map for India*

3.8 Topography

To understand the physical features of our catchment we compute various topographical characteristics such as mean watershed elevation, mean watershed aspect, aspect-northness, relief ratio, tangential curvature, mean watershed slope, etc. A full list of topographical characteristics is available in the database. Most of the characteristics

are derived from DEM. STRM DEM layers are obtained from the Consultative Group for International Agricultural Research (CGIAR). The steps to compute topographical characteristics are presented in the Listening 21.

The topographic wetness index is commonly used to quantify topographic control on hydrological processes. It is defined as: $\ln \frac{a}{tan(b)}$. where a a is the local upslope area draining through a certain point per unit contour length and $\tan(b)$ is the local slope in radians. **R** does not have any module to compute the Topographical wetness index(TWI). So we use GRASS with rgrass7 package to get the TWI map for each catchment. One crucial step to project DEM map into UTM otherwise GRASS yields negative TWI. The computation of the TWI map is shown in the listening 14. A sample map for TWI is also presented in Figure 11

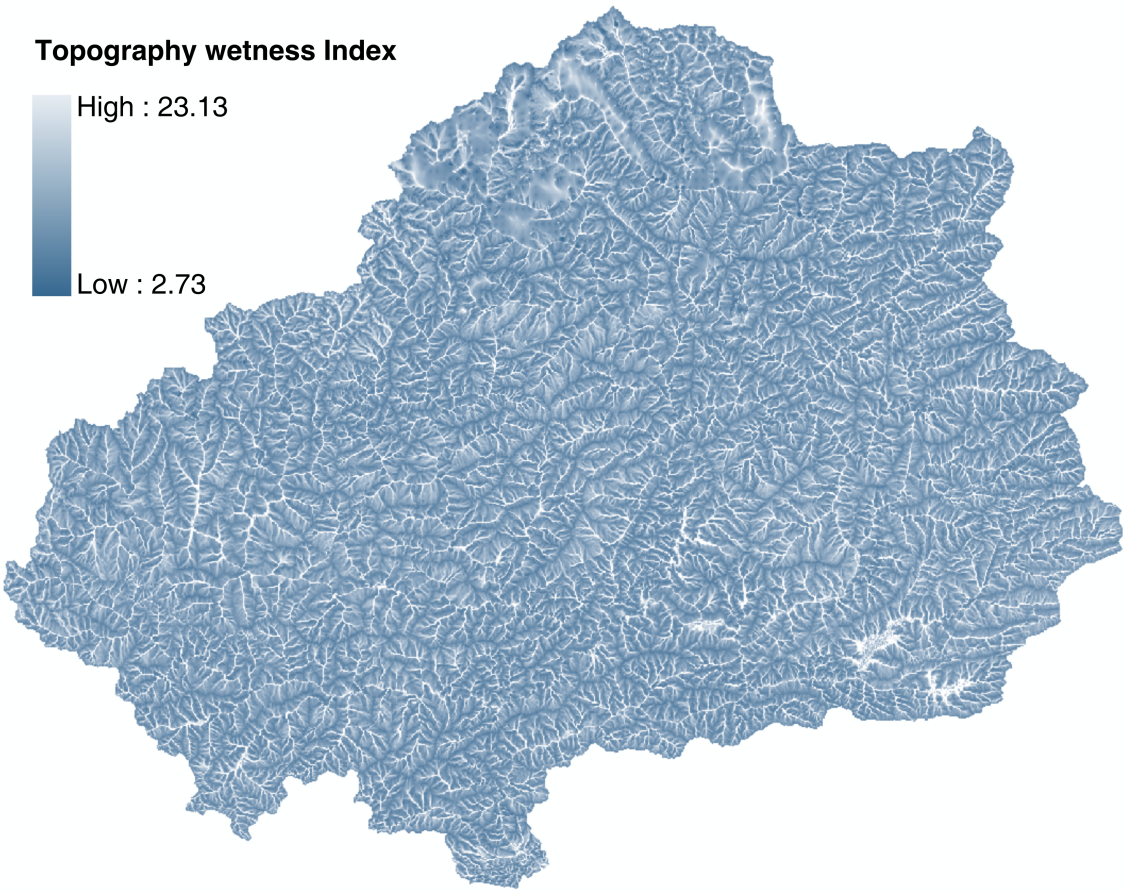

Figure 11: *Topographic index for a catchment.*

4 Other datasets

4.1 Discharge dataset

We obtain discharge data from WRIS India for 412 catchments. We do not have any gauge discharge data for Ganga, Indus, and Brahmaputra river basins. Discharge data provided in the daily time stamp but data availability vary for each gauge station. The data has the following columns 1. Day: Date of the observation, 2. Data Type: shows the measurement with respect to absolute gauge(HZS) or mean sea level(HHS), 3. Mean Gauge: gauge height in meter, 4. Discharge: runoff measured in cubic meter/sec (cumecs), and 5. Observed/Computed: data collection method observed(O) or computed(C).

4.2 Water quality dataset

Water quality data obtained from WRIS it has 366 stations monthly dataset with 33 water quality indicators. Data availability varies with gauge stations. The processing of water quality is not presented in this document. A list of water quality indicators are shown in Table 3

SN	Indices	SN	Indices	SN	Indices
1	pH FLD	12	NO2	23	Ca Major Ions
2	EC FLD	13	Total Nutrients	24	Mg
3	DO	14	SiO2	25	Na
4	Water	15	BOD	26	K
5	Color	16	COD	27	Cl
6	Odour	17	Phen	28	SO_4
7	pH GEN	18	Total Alkalinity	29	CO_3
8	EC GEN	19	Total Hardness	30	HCO_3
9	TDS	20	Ca Hardness	31	Total Coliforms
10	SS	21	F	32	Faecal
11	NH3	22	B	33	Chlorophyll

Table 3: *Water quality indices are available in the dataset.*

5 Code blocks for data processing (Listenings)

Listing 1: *Catchment delineation process with arcpy*

```
# -*- coding: utf-8 -*- ##############################################
"""
Created Thu Jun 30 17:37:50 2016
Last edited Fri Sep 16 15:01:38 2016
@author: Ankit Deshmukh
"""
## Spatial analysis with DEM ##########################################

import arcpy
arcpy.env.workspace = r"C:\TempWork"
arcpy.env.overwriteOutput = True

# input dem:
originalDEM = "IndiaDEM.tif"
arcpy.AddMessage("Elevation File: " + originalDEM+ "\n")

outFill = arcpy.sa.Fill(originalDEM)
outFill.save(r"D:\AnkitWork\20160603_rasterProc\DEM\shp\arcpyFiles\fill.tif")
arcpy.AddMessage("Fill DEM complete.\n")

#run flow direction tool and save raster
outFlowD = arcpy.sa.FlowDirection(outFill)
outFlowD.save('DEM_FlowD')
arcpy.AddMessage("Flow direction raster saved.\n")
outFlowD = r"D:\AnkitWork\20160603_rasterProc\DEM\shp\arcpyFiles\dem_flowd"

#run flow accumulation tool and save raster
outFlowA = arcpy.sa.FlowAccumulation(outFlowD)
outFlowA.save('DEM_FlowA')
arcpy.AddMessage("Flow accumulation raster saved.\n")

outFlowD = r"C:\TempWork\dem_flowd"
outFlowA = r"C:\TempWork\dem_flowa"

#Build a stream network and save raster
threshold = "2000" #(defalt = 2000)
arcpy.AddMessage("Stream network threshold:" + threshold+ "\n")
streamNetwork = arcpy.sa.Con(outFlowA,1,"","Value > "+threshold,)
streamNetwork.save("streamNetwork")

#use stream link tool and save raster
streamLink = arcpy.sa.StreamLink(streamNetwork,outFlowD)
streamLink.save("streamLink")

#use stream order tool and save raster
orderMethod = "STRAHLER" # specify the method ("SHREVE")
streamOrder = arcpy.sa.StreamOrder(streamNetwork,outFlowD,orderMethod)
streamOrder.save("streamOrder")

#Convert Stream to Feature
simplify = "SIMPLIFY" # option 2 : simplify = "NO_SIMPLIFY"
arcpy.sa.StreamToFeature(streamNetwork,outFlowD,"streams.shp")

#Run Snap Pour Point tooland save raster
arcpy.AddMessage("Running Snap Pour Point tool\n")

```

```
## read CSV file for input coords.
import csv
ptFile = open(r"C:\TempWork\PointData.csv")
ptReader = csv.reader(ptFile)
ptData = list(ptReader) # reading all data in once
shapeArea = []

for row in ptData:
   ptList = [float(row[3]), float(row[4])] # pour point one at a time
   wsName = row[1]+"_"+row[2]+ ".shp"
   wsNameNil = row[1]+"_"+row[2]

# Creating a point shape file;
   pt = arcpy.Point()
   ptGeoms = []
   pt.Y = ptList[0]
   pt.X = ptList[1]
   ptGeoms.append(arcpy.PointGeometry(pt))

#Writing pourpoint shapefile
   pourPoints = arcpy.CopyFeatures_management(ptGeoms,
      arcpy.env.workspace+ "\point.shp")
   tolerance = 0.01
   arcpy.AddMessage("Tolerance is fixed at %r\n" %tolerance)

#set env settings so union of inputs are used
   arcpy.env.extent = "MAXOF"
   snapPts = arcpy.sa.SnapPourPoint(pourPoints,outFlowA,tolerance)
   snapPts.save("SnapPourPts")

#run watershed tool and save raster
   arcpy.AddMessage("Running Watershed tool...\n")
   watershed = arcpy.sa.Watershed(outFlowD,snapPts)
   watershed.save("Watershed")

# converting raster to polygon shapefile:
   arcpy.AddMessage("Converting raster to shapefile...\n")
   arcpy.RasterToPolygon_conversion("Watershed", "temp.shp") # wsName

# dessloving shpapefiles for only single polygon
   fc = "temp.shp" # Temporary shapefiles
   fc_dis= arcpy.Dissolve_management(fc, wsName)

# Calulating area of each shapefile and saving into a csv.
   arcpy.AddField_management(fc_dis,"area","Double")
   expression1 = "{0}".format("!SHAPE.area@SQUAREKILOMETERS!") #SQL
      expression!.
   arcpy.CalculateField_management(fc_dis, "area", expression1, "PYTHON")

# reading area from shapefiles and
   featureClass = wsName
   searchCol = ["area"]
   with arcpy.da.SearchCursor(featureClass,searchCol) as sc:
      for row in sc:
         k = float(row[0])
         print k
         shapeArea.append((wsNameNil,k))

# saving area with shapefile name
with open("Shapefile_area.csv", "wb") as f:
   writer = csv.writer(f)
```

```
    writer.writerows(shapeArea)
```

Listing 2: *Plotting: Gauges stations boundaries selected for database. Histogram shows variation of catchment area.*

```
## Title :: Plotting the Indian catchments with gauge points colored with area
# Author :: Ankit Deshmukh
# DOC   :: 26 June 2019
# DOLE  :: 26 June 2019
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101/")
## Load required libraries ##################################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(raster)){install.packages("raster");library(raster)}

## Main code ##########################################################
gaugeData <- readxl::read_excel(path = "./CatchmentDelineation/CatchmentDelineation.xlsx", sheet = "PourPoint950")
Index = which(gaugeData$IndexForDB != 0)
Cats = length(Index)

Area = log10(gaugeData$CatAreaCalc_Km2[Index])
Order = order(Area,decreasing = TRUE)

#Read whole India shapefiles
India <- readOGR("./CatchmentDelineation/India_SHP_plotting", "All_India",verbose = FALSE)

#Read all the shapefiles
load("./CatchmentDelineation/AllShpFile567.RData")

#Transform coordinates into spatial point for plotting
d <- data.frame(lon=gaugeData$Long[Index], lat=gaugeData$Lat[Index])
coordinates(d) <- c("lon", "lat")
d@proj4string <- CRS(India@proj4string@projargs)

#Color based on area of watershed
colorRampAlpha <- function(..., n, alpha) {
  colors <- colorRampPalette(...)(n)
  paste(colors, sprintf("%x", ceiling(255*alpha)), sep="")
}
colVec <- c("deepskyblue","gold1","firebrick1")
Col <- colorRampAlpha(colVec, n=Cats, alpha=0.75)
Col = Col[as.numeric(cut(Area,breaks = Cats))]

#pdf(file = 'Figure01_StudyArea.pdf')
png(file = 'CatPourPlot.png', width = 3600, height = 3600, res = 600)
plot(India, border = 'gray40', lwd = 0.75)

# plotting order, such that biggest catchment poltted first.
for(cat in seq_along(Index)){#
  printIndex = Order[cat]
  plot(AllShpFile[[printIndex]],
```

```
      border =Col[printIndex],lwd = 0.5, add = T)

  points(gaugeData$Long[Index[printIndex]],
        gaugeData$Lat[Index[printIndex]],
        pch = 21, cex = 0.5, col = 'gray40',
        bg = Col[printIndex], lwd = 0.5)

  cat(cat,"\ ")
}

rng <- par("usr")
# insert north arrow
if(!require(GISTools)){install.packages("GISTools");library(GISTools)}
north.arrow(xb=88,yb=35.5,len=0.3,lab="N",cex=0.80)
detach('package:GISTools')

# insert scale
if(!require(maps)){install.packages("maps");library(maps)}
map.scale(x=83, y=33.5, ratio=FALSE, relwidth=0.2,sfcol = "red")
detach('package:maps')
dev.off()

#Plotting the histogram
AreaDB = data.frame(Area = Area) # log10 transformed
nbins = 30 # did some interation; but can be automated
Col   = colorRampPalette(colVec, alpha = 0.75)

ggplot(data = AreaDB, aes(x = Area))+
  geom_histogram(bins = nbins,color="white", fill= Col(nbins))+
  scale_x_continuous(name = "Area [scale::log10]",
                breaks =
                    seq(floor(min(AreaDB$Area)),ceiling(max(AreaDB$Area)),by=1),
                labels = 10^(1:6))+ylab('Catchment count[-]') +theme_bw()
ggsave(filename = "HistAreaComp_log.pdf",width = 4, height = 3.5, units =
    "in")

## Later use inkscape to merge figures into one.
```

Listing 3: *Downscale gridded precipitation data to catchment scale*

```
## Title :: Precipitation data processing daily and monthly products
# Author :: Ankit Deshmukh
# DOC   :: 2019-06-27 15:25:12
# DOLE  :: 2019-06-27
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database 101")
## Load required libraries ############################################
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(reader)){install.packages("reader");library(reader)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(reshape2)){install.packages("reshape2");library(reshape2)}
if(!require(doParallel)){install.packages("doParallel");library(doParallel)}
## Main code ##########################################################
Extent = data.frame(Long = c(66.50, 100.00), # West to East
               Lat = c(6.50, 38.50)) # South to Norths
Res     = 0.25
naVal   = -99.9
```

```r
startYear = 1951
endYear = 2013
DateName <- paste0("R",gsub("-","",seq(as.Date("1951-01-01"), as.Date("2013-12-31"), by="days")))

kk = 1
for(Year in 1:(endYear- startYear+1)){
  startData = paste0((startYear+Year-1),"-01-01")
  endData = paste0((startYear+Year-1),"-12-31")
  Date = gsub("-","",seq(as.Date(startData), as.Date(endData), by="days"))
  fileName = paste0("./Precipiation/IMDPrecipData/IND",(startYear+Year-1),"_rfP25.TXT")
  allLines = readLines(fileName)
  indexLine = substr(allLines[1], start = 9, stop = nchar(allLines[1])) #nchar(20190101)+1 = 9

  indexRow = c()
  for(ii in seq_along(Date)){
    indexRow[ii] <- which(allLines %in% paste0(Date[ii],indexLine))
  }

  Rows = (indexRow[2]-indexRow[1])-1
  for(Day in seq_along(indexRow)){
    Data = read.table(fileName,header = FALSE,skip = indexRow[Day],
                  nrows = Rows)

    Data = Data[,2:dim(Data)[2]] #remove latitude column.
    rownames(Data) <- seq(Extent$Lat[1],Extent$Lat[2], by = Res)
    colnames(Data) <- seq(Extent$Long[1],Extent$Long[2], by = Res)
    #image(as.matrix(Data))
    Data = apply(t(as.matrix(Data)),1,rev)
    #image(as.matrix(Data))
    Raster = raster(nrows = dim(Data)[1], ncols = dim(Data)[2],
                xmn = Extent$Long[1], xmx = Extent$Long[2],
                ymn = Extent$Lat[1], ymx = Extent$Lat[2],
                crs = CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"), #proj4string(catShp)
                vals = Data)

    #graphics.off();plot(indRaster);plot(catShp,add = T)
    print(paste(kk,Day))
    save(Raster,file = paste0("C:/Temp/RasterData/",DateName[kk],".RData"))
    kk = kk+1
  }
}

gaugeData <- readxl::read_excel("./CatchmentDelineation/CatchmentDelineation.xlsx",sheet = "PourPoint950")
gaugeName <- paste(gaugeData$`River Basins`,gaugeData$GaugeStations,sep = "_")
gaugeName <- gaugeName[gaugeData$IndexForDB ==1]

# Read the shapefile
for(shp in 1:length(gaugeName)){
  a= Sys.time()
  precipDaily = data.frame(Date = rep(NA,length(DateName)),
                      Precipitation = matrix(NA,nrow = length(DateName),ncol = 1))
  catShp <- readOGR("./CatchmentDelineation/Shapefiles 950",gaugeName[shp])

```

```r
  #making loop parellel
  cl <- makeCluster(24)
  registerDoParallel(cl)
  precip = unlist(foreach(Day = 1:length(rasterData))%dopar%{
    if(!require(raster)){install.packages("raster");library(raster)}
    # load the image data file and extract the value
    load(paste0("C:/Temp/RasterData/",DateName[Day],".RData"))
    temp = extract(x        = Raster,
                   y        = catShp,
                   weights      = TRUE,
                   normalizeWeights = TRUE,
                   df           = TRUE)
    rmIndex = temp$layer != naVal
    tempePrecip = sum(temp$layer[rmIndex]*temp$weight[rmIndex],na.rm = T)
  }
  )
  stopCluster(cl)
  precipDaily[,1] <- gsub("R","",DateName)
  precipDaily[,2] <- precip;rm(precip)
  write.csv(precipDaily,file =
      paste0("IMD_precip_daily_",gaugeName[shp],".csv"),row.names = FALSE)
  print(shp);print(Sys.time()-a)
}
```

Listing 4: *Convert daily precipitation data to monthly data*

```r
## Title :: Convert Daily Data to monthly data
# Author :: Ankit Deshmukh
# DOC   :: 2019-06-30 14:27:42
# DOLE  :: 2019-06-30
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database 101/")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(readxl)){install.packages("readxl");library(readxl)}
if(!require(lubridate)){install.packages("lubridate");library(lubridate)}

## Main code ##########################################################
gaugeData <-
    readxl::read_excel("./CatchmentDelineation/CatchmentDelineation.xlsx",sheet
    = "PourPoint950")
gaugeName <- paste(gaugeData$`River Basins`,gaugeData$GaugeStations,sep =
    "_")
gaugeName <- gaugeName[gaugeData$IndexForDB ==1]

Len = seq(ymd(19510101),ymd(20131231),by = "days")
precipAllShp <- matrix(NA, nrow = length(Len), ncol =
    (length(gaugeName)+1)) #One for Date
colnames(precipAllShp) <- c("Date",gaugeName)

for(shp in seq_along(gaugeName)){
  precipData =
      read.csv(paste0("./Precipiation/CatPrecipDaily/IMD_precip_daily_",gaugeName[shp],".
      = TRUE)
  Index = seq(which(precipData$Date == "19510101"), which(precipData$Date
      == "20131231"),by = 1)
  precipAllShp[,1] <- precipData$Date[Index]
```

```
  precipAllShp[,shp+1] <- precipData$Precipitation[Index]
  rm(precipData)
  print(shp)
}
write.csv(precipAllShp,file = "precipAllShp.csv",row.names = FALSE)

# get the monthly data based on month.
Month = month(ymd(precipAllShp[,1]))
Year = year(ymd(precipAllShp[,1]))

precipAllShpMonth <- matrix(NA,nrow = 63*12,(length(gaugeName)+2))
colnames(precipAllShpMonth) <- c("Month","Year",gaugeName)

for(shp in seq_along(gaugeName)){
  df = data.frame(Precip = precipAllShp[,(shp+1)],
              Month = Month,
              Year  = Year)
  Temp = aggregate(Precip~ Month + Year ,df,mean)
  precipAllShpMonth[,1] <- Temp$Month
  precipAllShpMonth[,2] <- Temp$Year
  precipAllShpMonth[,(shp+2)] <- Temp$Precip
  rm(df,Temp)
}

write.csv(precipAllShpMonth,file = "precipAllShpMonth.csv",row.names =
    FALSE)
```

Listing 5: *Compute precipitation based characteristics*

```
## Title :: Precipitation derivatives
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-02 12:25:37
# DOLE  :: 2019-07-02
# Remarks ::
## ClearUp and dir ######################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ##############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(lubridate)){install.packages("lubridate");library(lubridate)}

## Main code ############################################################
#load the precipitation dataset
precipDaily <- read.csv("./Precipiation/precipAllShp.csv",header = T)
gaugeCount <- dim(precipDaily)[2]-1

Month = month(ymd(precipDaily[,1]))
Year = year(ymd(precipDaily[,1]))

## 1. Mean annual precip (mm) for the watershed.
# computing mean of 63 year of daily precipitation data for all 567
    catchments
PPTAVG_BASIN = colMeans(precipDaily[,2:dim(precipDaily)[2]]) # Col:1 ==
    Date coloumn.

## 2. Watershed average of maximum monthly precipitation (mm).
## 3. Watershed average of minimum monthly precipitation (mm).
# idenfity maximum values of each month and average it

PPTMAX_BASIN <- PPTMIN_BASIN <- rep(NA,gaugeCount)
```

```r
for (cat in 1:gaugeCount) {
  df = data.frame(Precip = precipDaily[,cat+1],
                  Month = Month,
                  Year  = Year)
  PPTMAX_BASIN[cat] = mean(aggregate(Precip ~ Month+Year,df,max)$Precip)
  PPTMIN_BASIN[cat] = mean(aggregate(Precip ~ Month+Year,df,min)$Precip)
  print(cat)
}
#plot(1:length(maxMonthlyPrecip),maxMonthlyPrecip,"l",lwd = 1,col = "red")
#lines(1:length(minMonthlyPrecip),minMonthlyPrecip,"l",lwd = 1,col = "#33eeff66")

## 4. Watershed average of annual number of days (days) of measurable precipitation.
WD_BASIN <- rep(NA,gaugeCount)
for(cat in 1:gaugeCount){
  precipDayIndex <- precipDaily[,cat+1] > 0
  mesurePrecip <- precipDaily[,cat+1][precipDayIndex]
  mesureYear  <- Year[precipDayIndex]
  df <- data.frame(Precip = mesurePrecip,
                   Year  = mesureYear)
  WD_BASIN[cat] <- mean(aggregate(Precip ~ Year,df,function(x){length(x)})$Precip)
}

# 5. Watershed average of monthly maximum number of days (days) of measurable precipitation.
# 6. Watershed average of monthly minimum number of days (days) of measurable precipitation.
WDMAX_BASIN <- WDMIN_BASIN <- rep(NA,gaugeCount)
for(cat in 1:gaugeCount){
  precipDayIndex <- precipDaily[,cat+1] > 0
  mesurePrecip <- precipDaily[,cat+1][precipDayIndex]
  mesureMonth <- Month[precipDayIndex]
  mesureYear  <- Year[precipDayIndex]
  df <- data.frame(Precip = mesurePrecip,
                   Month = mesureMonth,
                   Year  = mesureYear)
  df <- aggregate(Precip ~ Year+Month,df,function(x){length(x)})

  WDMAX_BASIN[cat] <- mean(aggregate(Precip ~ Month,df,max)$Precip)
  WDMIN_BASIN[cat] <- mean(aggregate(Precip ~ Month,df,min)$Precip)
}
# 7. Precipitation seasonality index (Walsh and Lawler (1981)). Index of how much annual precipitation falls seasonally (high values) or spread out over the year (low values). Based on monthly precip values from 30 year (1971-2000) PRISM. Range is 0 (precip spread out exactly evenly in each month) to 1 (all precip falls in a single month).
PRECIP_SEAS_IND <- PRECIP_SEAS_IND_CLS <- rep(NA,gaugeCount)
for(cat in 1:gaugeCount){
  df <- data.frame(Precip = precipDaily[,cat+1],
                   Month = Month,
                   Year  = Year)
  catPrecipMonthly <- aggregate(Precip ~ Month+Year,df,sum)

  yrLn <- length(unique(catPrecipMonthly$Year))
  SI <- rep(NA,yrLn)
  k = 0
  for(ii in 1:yrLn){
    monthPrecipSI <- catPrecipMonthly$Precip[(k+1):(k+12)]
    SI[ii] <- sum(abs((monthPrecipSI-mean(monthPrecipSI,na.rm =
```

```
        TRUE))/(mean(monthPrecipSI,na.rm = TRUE)*12)),na.rm = TRUE)
    k= k+12
  }
  PRECIP_SEAS_IND[cat] <- mean(SI)
  print(cat)
}
# ----------------------------------------------------------------|
# Seasonality Index classes (when using mean monthly data) |
# ----------------------------------------------------------------|
# Class | Class definition              | Ranges   |
# ----------------------------------------------------------------|
#    1 | Very equable                   | <= 0.19  |
#    2 | Equable but with a definite wetter season | 0.20 - 0.39 |
#    3 | Rather seasonal with a short drier season | 0.40 - 0.59 |
#    4 | Seasonal                       | 0.60 - 0.79 |
#    5 | Markedly seasonal with a long drier season| 0.80 - 0.99 |
#    6 | Most rain in 3 months or less  | 1.00 - 1.19 |
#    7 | Extreme, almost all rain in 1 - 2 months | >= 1.20 |
# ----------------------------------------------------------------|

PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND <= 0.19] <- 1
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 0.19 & PRECIP_SEAS_IND <= 0.39] <- 2
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 0.39 & PRECIP_SEAS_IND <= 0.59] <- 3
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 0.59 & PRECIP_SEAS_IND <= 0.79] <- 4
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 0.79 & PRECIP_SEAS_IND <= 0.99] <- 5
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 0.99 & PRECIP_SEAS_IND <= 1.19] <- 6
PRECIP_SEAS_IND_CLS[PRECIP_SEAS_IND > 1.19] <- 7

# 8. Mean January precip (mm) for the watershed.
# 9. Mean February precip (mm) for the watershed.
# 10. Mean March precip (mm) for the watershed.
# 11. Mean April precip (mm) for the watershed.
# 12. Mean May precip (mm) for the watershed.
# 13. Mean June precip (mm) for the watershed.
# 14. Mean July precip (mm) for the watershed.
# 16. Mean August precip (mm) for the watershed.
# 17. Mean September precip (mm) for the watershed.
# 18. Mean October precip (mm) for the watershed.
# 19. Mean November precip (mm) for the watershed.
# 20. Mean December precip (mm) for the watershed.
## 4. Watershed average of annual number of days (days) of measurable
    precipitation.
meanMonthlyPrecip <- matrix(NA, nrow = gaugeCount, ncol =
    length(month.name))
colnames(meanMonthlyPrecip) <- month.name
for(cat in 1:gaugeCount){
  df <- data.frame(Precip = precipDaily[,cat+1],
                   Month = Month)
  meanMonthlyPrecip[cat,] <- aggregate(Precip ~ Month,df,mean)$Precip
}

## Exporting to csv
PrecipExp <- data.frame(PPTAVG_BASIN = PPTAVG_BASIN,
                        PPTMAX_BASIN   = PPTMAX_BASIN,
                        PPTMIN_BASIN   = PPTMIN_BASIN,
                        WD_BASIN       = WD_BASIN,
                        WDMAX_BASIN    = WDMAX_BASIN,
                        WDMIN_BASIN    = WDMIN_BASIN,
                        PRECIP_SEAS_IND = PRECIP_SEAS_IND,
                        PRECIP_SEAS_IND_CLS = PRECIP_SEAS_IND_CLS,
                        JAN_PPT7100_CM = meanMonthlyPrecip[,1],
```

```
                    FEB_PPT7100_CM = meanMonthlyPrecip[,2],
                    MAR_PPT7100_CM = meanMonthlyPrecip[,3],
                    APR_PPT7100_CM = meanMonthlyPrecip[,4],
                    MAY_PPT7100_CM = meanMonthlyPrecip[,5],
                    JUN_PPT7100_CM = meanMonthlyPrecip[,6],
                    JUL_PPT7100_CM = meanMonthlyPrecip[,7],
                    AUG_PPT7100_CM = meanMonthlyPrecip[,8],
                    SEP_PPT7100_CM = meanMonthlyPrecip[,9],
                    OCT_PPT7100_CM = meanMonthlyPrecip[,10],
                    NOV_PPT7100_CM = meanMonthlyPrecip[,11],
                    DEC_PPT7100_CM = meanMonthlyPrecip[,12])

write.csv(PrecipExp,file = "./Precipiation/Climate.csv",row.names = FALSE)
```

Listing 6: *Downscale gridded temperature data to catchment scale*

```
## Title :: Temperature data processing daily and monthly products
# Author :: Ankit Deshmukh
# DOC   :: 2019-06-27 15:25:12
# DOLE  :: 2019-06-27
# Remarks :: Min and Max

## ClearUp and dir ##################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database 101")
## Load required libraries ##########################################
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(reader)){install.packages("reader");library(reader)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(stringr)){install.packages("stringr");library(stringr)}
if(!require(reshape2)){install.packages("reshape2");library(reshape2)}
if(!require(doParallel)){install.packages("doParallel");library(doParallel)}
## Main code ########################################################
Extent = data.frame(Long = c(67.50, 97.50), # West to East
               Lat = c(7.50, 37.50)) # South to Norths
Res     = 1.0
naVal   = 99.90
startYear = 1951
endYear = 2013

temp = seq(dmy("01-01-1951"),dmy("31-12-2013"),by = "days")
DateName <-
    paste0("R",str_pad(day(temp),2,pad="0"),str_pad(month(temp),2,pad =
    "0"),year(temp)); rm(temp)

# Choose min, mean or max data procession
flag = c("MaxT","MeanT","MinT")[as.numeric(readline(prompt="Enter 1:Max |
    2:Mean | 3:Min : "))]

kk = 1
for(Year in 1:(endYear- startYear+1)){
  startData = paste0((startYear+Year-1),"-01-01")
  endData = paste0((startYear+Year-1),"-12-31")
  Date    = seq(ymd(startData), ymd(endData), by="days")
  Date    = paste0(str_pad(day(Date),2,pad="0"),str_pad(month(Date),2,pad =
      "0"),year(Date))
  fileName =
      paste0("./Temperature/",flag,"/",flag,"_",(startYear+Year-1),".TXT")
  allLines = readLines(fileName)
  allLines = allLines[-1]
```

```
  indexRow <- (which(allLines %in% allLines[1])) + 1 # one line removed form title

  Rows = (indexRow[2]-indexRow[1]) # -1 removed when used n.readline: "99.9-10.54"
  for(Day in seq_along(indexRow)){
    #Data = read.table(fileName,header = FALSE,skip = indexRow[Day],
    #             nrows = Rows) #
    #Data = Data[,3:dim(Data)[2]] #remove latitude column.
    #total row is 32.
    dtLines = n.readLines(fileName , n = Rows, skip = indexRow[Day]-1)
    listChar = regmatches(dtLines,gregexpr("\\-*\\d+\\.*\\d*",dtLines))
    Data = matrix(NA,nrow=length(dtLines),ncol = 31)
    for(ii in seq_along(listChar)){
      Data[ii,] = as.numeric(unlist(listChar[ii]))[3:33]
    }

    rownames(Data) <- seq(Extent$Lat[1],Extent$Lat[2], by = Res)
    colnames(Data) <- seq(Extent$Long[1],Extent$Long[2], by = Res)
    #image(as.matrix(Data))
    Data = apply(t(as.matrix(Data)),1,rev)
    #image(as.matrix(Data))
    Raster = raster(nrows = dim(Data)[1], ncols = dim(Data)[2],
                xmn = Extent$Long[1], xmx = Extent$Long[2],
                ymn = Extent$Lat[1], ymx = Extent$Lat[2],
                crs = CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"), #proj4string(catShp)
                vals = Data)

    #graphics.off();plot(Raster);plot(catShp,add = T)
    print(paste(kk,Day))
    save(Raster,file = paste0("C:/Temp/RasterData",flag,"/",DateName[kk],".RData"))
    kk = kk+1
  }
}

gaugeData <- readxl::read_excel("./CatchmentDelineation/CatchmentDelineation.xlsx",sheet = "PourPoint950")
gaugeName <- paste(gaugeData$`River Basins`,gaugeData$GaugeStations,sep = "_")
gaugeName <- gaugeName[gaugeData$IndexForDB ==1]

# Read the shapefile
for(shp in 1:length(gaugeName)){ #473
  a= Sys.time()
  temperDaily = data.frame(Date = rep(NA,length(DateName)),
                     Temperature = matrix(NA,nrow = length(DateName),ncol = 1))
  catShp <- readOGR("./CatchmentDelineation/Shapefiles 950",gaugeName[shp])

  #making loop parellel
  cl <- makeCluster(24)
  registerDoParallel(cl)
  temper = unlist(foreach(Day = 1:length(DateName))%dopar%{
    if(!require(raster)){install.packages("raster");library(raster)}
    # load the image data file and extract the value
    load(paste0("C:/Temp/RasterData",flag,"/",DateName[Day],".RData"))
    temp = extract(x        = Raster,
                y           = catShp,
```

```
                weights      = TRUE,
                normalizeWeights = TRUE,
                df           = TRUE)
    rmIndex = temp$layer != naVal
    tempTemp = sum(temp$layer[rmIndex]*temp$weight[rmIndex],na.rm = T)
  }
  )
  stopCluster(cl)
  temperDaily[,1] <- gsub("R","",DateName)
  temperDaily[,2] <- temper;rm(temper)
  write.csv(temperDaily,file =
      paste0("./Temperature/",flag,"/IMD_Temper_",flag,"_daily_",gaugeName[shp],".csv"),r
      = FALSE)
  print(shp);print(Sys.time()-a)
}
```

Listing 7: *Convert daily temperature data to monthly data*

```
## Title :: Convert Daily Data to monthly data for Temperature
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-01 16:38:25
# DOLE  :: 2019-07-01
# Remarks :: Combine daily precipitation to one file and convert into
    monthly data.
## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database 101/")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(readxl)){install.packages("readxl");library(readxl)}
if(!require(lubridate)){install.packages("lubridate");library(lubridate)}

## Main code ##########################################################
gaugeData <-
    readxl::read_excel("./CatchmentDelineation/CatchmentDelineation.xlsx",sheet
    = "PourPoint950")
gaugeName <- paste(gaugeData$`River Basins`,gaugeData$GaugeStations,sep =
    "_")
gaugeName <- gaugeName[gaugeData$IndexForDB ==1]

# use flag to process Max and Min data in a single code. Mean will be avg
    of max and Min.
flag = c("MaxT","MeanT","MinT")[as.numeric(readline(prompt="Enter 1:Max |
    2:Mean | 3:Min : "))]

Len = as.numeric(gsub("-","",seq(ymd(19510101),ymd(20131231),by = "days")))
tempAllShp <- matrix(NA, nrow = length(Len), ncol = (length(gaugeName)+1))
    #One for Date
colnames(tempAllShp) <- c("Date",gaugeName)

for(shp in seq_along(gaugeName)){
  tempData <-
      read.csv(paste0("./Temperature/",flag,"/IMD_Temper_",flag,"_daily_",gaugeName[shp],
      = TRUE)
  tempAllShp[,1] <- Len
  tempAllShp[,shp+1] <- tempData$Temperature
  rm(tempData)
  print(shp)
}

```

```r
write.csv(tempAllShp,file = paste0("tempAllShp_",flag,".csv"),row.names = FALSE)

# get the monthly data based on month.
Month = month(ymd(tempAllShp[,1]))
Year = year(ymd(tempAllShp[,1]))

tempAllShpMonth <- matrix(NA,nrow = (length(unique(Month))*length(unique(Year))),(length(gaugeName)+2))# 2 extra for MM +YYYY
colnames(tempAllShpMonth) <- c("Month","Year",gaugeName)

for(shp in seq_along(gaugeName)){
  df = data.frame(Temper = tempAllShp[,(shp+1)],
                  Month = Month,
                  Year = Year)
  Temp = aggregate(Temper ~ Month + Year ,df,mean)
  tempAllShpMonth[,1] <- Temp$Month
  tempAllShpMonth[,2] <- Temp$Year
  tempAllShpMonth[,(shp+2)] <- Temp$Temper
  rm(df,Temp)
}

write.csv(tempAllShpMonth,file = paste0("tempAllShpMonth_",flag,".csv"),row.names = FALSE)
```

Listing 8: *Compute temperature based characteristics*

```r
## Title :: Temperature Derivatives
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-05 13:22:41
# DOLE  :: 2019-07-05
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(lubridate)){install.packages("lubridate");library(lubridate)}

## Main code ##########################################################
tempDailyMax <- read.csv("./Temperature/tempAllShp_MaxT.csv",header = TRUE)
tempDailyMean <- read.csv("./Temperature/tempAllShp_MeanT.csv",header = TRUE)
tempDailyMin <- read.csv("./Temperature/tempAllShp_MinT.csv",header = TRUE)

Month    <- month(ymd(tempDailyMean[,1]))
Year     <- year(ymd(tempDailyMean[,1]))
JDay     <- yday(ymd(tempDailyMean[,1]))

gaugeCount <- dim(tempDailyMean)[2]-1

# 1.Average annual air temperature for the watershed, degrees C.
T_AVG_BASIN <- colMeans(tempDailyMean[,2:dim(tempDailyMean)[2]])

# 2.Watershed average of maximum monthly air temperature (degrees C).
# 2.Standard deviation of maximum monthly air temperature (degrees C).
T_MAX_BASIN <- T_MAXSTD_BASIN <- rep(NA,gaugeCount)
for(cat in 1:gaugeCount){
  df <- data.frame(Temp = tempDailyMean[,(cat+1)],
```

```
                Month = Month,
                Year = Year)
  TempMon <- aggregate(Temp~Month+Year,df,max)$Temp

  T_MAX_BASIN[cat] <- mean(TempMon)
  T_MAXSTD_BASIN[cat] <- sd(TempMon)
  print(cat)
  rm(df)
}

# 3.Watershed average of minimum monthly air temperature (degrees C.
# 4.Standard deviation of minimum monthly air temperature (degrees C).
T_MIN_BASIN <- T_MINSTD_BASIN <- rep(NA,gaugeCount)
for(cat in 1:gaugeCount){
  df <- data.frame(Temp = tempDailyMean[,(cat+1)],
                Month = Month,
                Year = Year)
  TempMon <- aggregate(Temp~Month+Year,df,min)$Temp

  T_MIN_BASIN[cat] <- mean(TempMon)
  T_MINSTD_BASIN[cat] <- sd(TempMon)
  print(cat)
  rm(df)
}

# 5.Average January air temperature for the watershed, degrees C.
# 6.Average February air temperature for the watershed, degrees C.
# 7.Average March air temperature for the watershed, degrees C
# 8.Average April air temperature for the watershed, degrees C.
# 9.Average May air temperature for the watershed, degrees C.
# 10.Average June air temperature for the watershed, degrees C.
# 11.Average July air temperature for the watershed, degrees C.
# 12.Average August air temperature for the watershed, degrees C.
# 13.Average September air temperature for the watershed, degrees C.
# 14.Average October air temperature for the watershed, degrees C.
# 15.Average November air temperature for the watershed, degrees C.
# 16.Average December air temperature for the watershed, degrees C.
meanMonthlyTemp <- matrix(NA, nrow = gaugeCount, ncol = length(month.name))
colnames(meanMonthlyTemp) <- month.name
for(cat in 1:gaugeCount){
  df <- data.frame(Temp = tempDailyMean[,cat+1],
                Month = Month)
  meanMonthlyTemp[cat,] <- aggregate(Temp ~ Month,df,mean)$Temp
}

# # Visulize the monthly temperature
# for(ii in 1:gaugeCount){
#  plot(meanMonthlyTemp[ii,],
#      type = "b",col= rgb(runif(ii),runif(ii),runif(ii)),
#      lwd = 3,pch = 16,cex = 2)
#  Sys.sleep(0.5)
# }

# 17.Mean-annual potential evapotranspiration (PET), estimated using the
    Hamon (1961) equation.
# 18.Mean-annual potential evapotranspiration (PET), estimated using the
    Hargreves (xxx) equation.
gaugeData <-
    readxl::read_excel("./CatchmentDelineation/CatchmentDelineation.xlsx",sheet
    = "PourPoint950")
Latitude <- gaugeData$Lat[gaugeData$IndexForDB ==1]
```

```r

source("./Temperature/Evapotranspiration/hargreaves.R")
source("./Temperature/Evapotranspiration/hamon.R")

DateStr<- data.frame(Year = Year,
                     Month = Month,
                     JDay = JDay)

PET_HAMON <- PET_HARTG <- rep(NA,gaugeCount)

for(cat in 1:length(Latitude)){
  PET_HAMON[cat] = mean(hamon(tempDailyMean[,(cat+1)],
          Latitude[cat],
          DateStr))

  PET_HARTG[cat] = mean(hargreaves(tempDailyMean[,(cat+1)],
           tempDailyMax[,(cat+1)],
           tempDailyMin[,(cat+1)],
           Latitude[cat],
           DateStr))
  print(cat)
}

TempExp <- data.frame(T_AVG_BASIN = T_AVG_BASIN,
                      T_MAX_BASIN = T_MAX_BASIN,
                      T_MAXSTD_BASIN = T_MAXSTD_BASIN,
                      T_MIN_BASIN = T_MIN_BASIN,
                      T_MINSTD_BASIN = T_MINSTD_BASIN,
                      JAN_TMP7100_DEGC = meanMonthlyTemp[,1],
                      FEB_TMP7100_DEGC = meanMonthlyTemp[,2],
                      MAR_TMP7100_DEGC = meanMonthlyTemp[,3],
                      APR_TMP7100_DEGC = meanMonthlyTemp[,4],
                      MAY_TMP7100_DEGC = meanMonthlyTemp[,5],
                      JUN_TMP7100_DEGC = meanMonthlyTemp[,6],
                      JUL_TMP7100_DEGC = meanMonthlyTemp[,7],
                      AUG_TMP7100_DEGC = meanMonthlyTemp[,8],
                      SEP_TMP7100_DEGC = meanMonthlyTemp[,9],
                      OCT_TMP7100_DEGC = meanMonthlyTemp[,10],
                      NOV_TMP7100_DEGC = meanMonthlyTemp[,11],
                      DEC_TMP7100_DEGC = meanMonthlyTemp[,12],
                      PET_HAMON = PET_HAMON,
                      PET_HARTG = PET_HARTG)

write.csv(TempExp, file = "./Temperature/Temperature.csv", row.names =
    FALSE)
```

Listing 9: *A function to compute leap year*

```r
is.leapyear=function(year){
  return(((year %% 4 == 0) & (year %% 100 != 0)) | (year %% 400 == 0))
}
```

Listing 10: *A function to compute solar radiation*

```r
## Title :: Solar radiation computation with latitude and Jday
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-08 16:42:32
# DOLE  :: 2019-07-08
# Remarks ::

```

```r
## ClearUp and dir ####################################################
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
source("./Temperature/Evapotranspiration/is.leapyear.R")
## Main code ##########################################################
solar_rad <- function(latitude,Date){
  numdays <- length(Date$JDay)
  J     <- rep(0,numdays)
  delta <- rep(0,numdays)
  dr    <- rep(0,numdays)
  omegas <- rep(0,numdays)
  So    <- rep(0,numdays)

  for (day in 1:numdays){
    J[day]    = Date$JDay[day];
    lat       = latitude*pi/180;

    if(is.leapyear(Date$Year[day])){
      delta[day] = 0.4093 * sin((2*pi*J[day]/366) - 1.405)
      dr[day]  = 1+0.0333 * cos(2*pi*J[day]/366)
    }else{
      delta[day] = 0.4093 * sin((2*pi*J[day]/365) - 1.405)
      dr[day]  = 1+0.0333 *cos(2*pi*J[day]/365)
    }
    omegas[day] = acos(-tan(lat)* tan(delta[day]))
    So[day]   =
        15.392*dr[day]*((omegas[day]*sin(lat)*sin(delta[day]))+(cos(lat)*cos(delta[day])*
  }
  return(So)
}
```

Listing 11: *Compute evapotranspiration using Hamon equation*

```r
## Title :: Hamon fuction to compute ET
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-08 17:44:43
# DOLE  :: 2019-07-08
# Remarks ::
#hamon_new: function to calculate the Potential evapotranspiration using
    the Hamon equation
#latitude: numeric with latitute data
#date: a structure array with fields: Year, Month, Day ranging from the
    beginning date all the way to the end dates for the analysis

## ClearUp and dir ####################################################
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
source("./Temperature/Evapotranspiration/solar_rad.R")
source("./Temperature/Evapotranspiration/is.leapyear.R")
## Main code ##########################################################
hamon <- function(tav,latitude,Date){
  numdays = length(tav)
  PET <- rep(0,numdays)
  for(day in 1: numdays){
   J = Date$JDay[day]
   e_a = 0.611*exp(17.3*tav[day]/(tav[day]+237.3))

   if(is.leapyear(Date$Year[day])){
     Gamma = 2*pi*(J-1)/366
   }else{
```

```
    Gamma = 2*pi*(J-1)/365
  }

  delta =
      (0.006918-0.399912*cos(Gamma)+0.070257*sin(Gamma)-0.006758*cos(2*Gamma)+0.000907*s
  #delta should be in radians so no multiplication with 180/pi
  #from Page 602 in Dingman: Physical Hydrology
  omega    =  2*pi/24; #angluar velocity of Earth's rotation

  #from Page 603 in Dingman: Physical Hydrology
  Dy       =  2*acos(-tan(delta)*tan(latitude*pi/180))/omega;

  #from Page 310 in Dingman: Physical Hydrology
  if(tav[day] == -9999){
    PET[day] = -9999
  }else{
     if(tav[day] > 0){
       PET[day] = 29.8*Dy*(e_a/(tav[day]+237.2))
     }else{
       PET[day] = 0
     }
    }
   rm(J,e_a,Gamma,delta,omega,Dy)
 }
 return(PET)
}
```

Listing 12: *Compute evapo transpiration using hargreaves equation*

```
## Title :: Hargreaves and Samini PET equation 1985 formulation
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-08 18:25:29
# DOLE  :: 2019-07-08
# Remarks ::

## Load required libraries ##########################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
source("./Temperature/Evapotranspiration/solar_rad.R")
## Main code ########################################################
hargreaves <- function(Tav,Tmax,Tmin,latitude,daterun){
  # Estimating the solar radiation
  So     = solar_rad(latitude,daterun)
  num_runs = length(Tmax)
  PET    = rep(0,num_runs)

  # Based on Hargreeaves and Samani 1985
  for(timestep in 1:num_runs){
    if(Tav[timestep] == -9999 | Tmax[timestep] == -9999 | Tmin[timestep] ==
        -9999){
      PET[timestep] = -9999
    } else{
      if((Tmax[timestep] - Tmin[timestep]) >=0 & Tmax[timestep] >= 0){
        PET[timestep]= 0.0023*So[timestep] *
            (Tmax[timestep]-Tmin[timestep])^0.5 * (Tav[timestep]+17.8)
      }else{
        PET[timestep] = 0
      }
    }
  }
  return(PET)
```

```
}
```

Listing 13: *Code I use to get geological characteristics*

```
## Title :: File Description
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-17 19:57:30
# DOLE  :: 2019-07-17
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(rgeos)){install.packages("rgeos");library(rgeos)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}

## Main code ##########################################################
xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")

LithoLayer <- raster("./Geology/glim_wgs84.asc") # reading the raster file.
crs(LithoLayer) <- ("+init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs
    +ellps=WGS84 +towgs84=0,0,0")

#India <- readOGR("./CatchmentDelineation/India_SHP_plotting","All_India",
    verbose = FALSE)
#plot(LithoLayer);plot(India,add = TRUE);rm(India)
Classes = 16
LithoCount <- matrix(NA, nrow = length(Index), ncol = Classes)
for(cat in seq_along(Index)){
  #now extract the information in the grids
  temp <- extract(x        = LithoLayer,
                  y        = AllShpFile[[cat]],
                  weights  = TRUE,
                  normalizeWeights = TRUE,
                  df       = TRUE)

  LithoExt <- temp$glim_wgs84[!is.na(temp$glim_wgs84)]
  LithoWt  <- temp$weight[!is.na(temp$glim_wgs84)]

  for(Type in 1:Classes) {
    LithoCount[cat,Type] <- sum(LithoWt[LithoExt == Type])*100
  }
  cat(cat,"\n")
}
LithoInfo <- read.csv(file = "./Geology/LithoInfo.csv",header = TRUE)
colnames(LithoCount) <- LithoInfo$CatName
write.csv(LithoCount,file = "./Geology/Geology.csv", row.names = FALSE)
```

Listing 14: *To process DEM to get stream order and topographical properties*

```
## Title :: Raster procession in R using GRASS GIS-7.6 with opneSTARS and
    rgrass7
# Author :: Ankit Deshmukh
```

```r
# DOC   :: 2019-07-11 11:34:02
# DOLE  :: 2019-07-11
# Remarks ::
## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(openSTARS)){install.packages("openSTARS");library(openSTARS)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(readxl)){install.packages("readxl");library(readxl)}
## Main code ##########################################################
xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index = which(xlData$IndexForDB == 1)

# Load files Raster and Vector files ##################################
demFileName <-
    c("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101/CatchmentDelineation/IndiaDEM
indDEM <- raster(demFileName)
for(cat in Index){
  shpFile <- readOGR(paste0(getwd(),
      "/CatchmentDelineation/Shapefiles950"),
                     paste0(xlData$`River
                         Basins`[cat],"_",xlData$GaugeStations[cat]))

  temp = mask(crop(indDEM, shpFile),shpFile)
  if(temp@extent@xmin >= 66 & temp@extent@xmax < 72){CRS =
      "+init=epsg:32642"}
  if(temp@extent@xmin >= 72 & temp@extent@xmax < 78){CRS =
      "+init=epsg:32643"}
  if(temp@extent@xmin >= 78 & temp@extent@xmax < 84){CRS =
      "+init=epsg:32644"}
  if(temp@extent@xmin >= 84 & temp@extent@xmax < 90){CRS =
      "+init=epsg:32645"}
  if(temp@extent@xmin >= 90 & temp@extent@xmax < 96){CRS =
      "+init=epsg:32646"}
  if(temp@extent@xmin >= 96 & temp@extent@xmax < 102){CRS =
      "+init=epsg:32647"}

  temp = projectRaster(temp, crs = CRS)

  # writing the raster
  writeRaster(temp,
              paste0("./Hyderology/CropTiffData/",xlData$`River
                  Basins`[cat],"_",xlData$GaugeStations[cat]),
              format = "GTiff",
              overwrite = TRUE)
  rm(temp)
  print(cat)
}

## Initiate GRASS session ##############################################
if(.Platform$OS.type == "windows"){gisbase = "C:/Program Files/GRASS GIS
    7.6"}
initGRASS(gisBase = gisbase,
          home    = tempfile(),
          override = TRUE)

```

```r
##Grass-R functions ##################################################
r.list <- function(){
  print(execGRASS("g.list",
              parameters = list(type = "rast"),
              intern   = TRUE))
}
v.list <- function(){
  print(execGRASS("g.list",
              parameters = list(type = "vect"),
              intern   = TRUE))
}

rm_grass <- function(Name,R = "r"){
  Type = ifelse(R == "r","raster","vector")
  execGRASS("g.remove",flags = "f",
          parameters = list(name = Name,type = Type),intern = TRUE)
}
#######################################################################
tifName <- paste0(xlData$RiverBasins[Index],"_",xlData$GaugeStations[Index])
tifFile <- paste0(getwd(),"/Hyderology/DEM_Derivatives/",tifName,".tif")
shpFile <- paste0(getwd(),"/CatchmentDelineation/Shapefiles950/",tifName,".shp")

for(cat in 1:length(Index)){
  ## Setting up grass GIS envionment for mapset etc
  setup_grass_environment(dem = tifFile[cat])
  gmeta()

  ## load the raster layers
  execGRASS("r.in.gdal",
          flags = c("overwrite", "quiet"),
          parameters = list(input = tifFile[cat],band = 1,output = tifName[cat]),
          ignore.stderr = T)

  proj_dem <- execGRASS("g.proj", flags = c("j", "f"), parameters = list(georef = tifFile[cat]),
                  intern = TRUE, ignore.stderr = TRUE)

  # ## load the vector layers
  # execGRASS("v.import",flags =c("overwrite"), input = shpFile[cat],
  #         output = tifName[cat], intern = TRUE)

  # execGRASS("r.mask", flags = c("overwrite"),
  #         parameters = list(vector = tifName[cat]),
  #         intern = TRUE)
  #
  # execGRASS("g.region", flags = c("overwrite","p","d"),
  #         parameters = list(raster = "MASK"),
  #         intern = TRUE)
  #
  # execGRASS("r.clip",flags = c("overwrite"),
  #         parameters = list(input = tifName[cat],
  #                       output = paste0(tifName[cat],"_Crop")))

  execGRASS("r.hydrodem",
          flags = c("overwrite"),
          parameters = list(input = tifName[cat],output = paste0(tifName[cat],"_Cond"))
  )
```

```
execGRASS("r.watershed",
        flags = c("overwrite", "quiet"),
        parameters = list(elevation = paste0(tifName[cat],"_Cond"),
                      accumulation = paste0(tifName[cat],"_Accum"))
)

# MiKatt: Solution: set d8cut to total number of cells in g.region.
ncell <- execGRASS("g.region",flags="p",intern=T)
ncell <- as.numeric(unlist(strsplit(ncell[grep("cells",ncell)],split=":"))[2])
execGRASS("r.stream.extract",
        flags = c("overwrite", "quiet"),
        parameters = list(elevation = paste0(tifName[cat],"_Cond"),
                      accumulation = paste0(tifName[cat],"_Accum"),
                      threshold = 700, # use ATRIC to get this value?
                      d8cut = ncell,
                      stream_length = 0,
                      stream_raster = paste0(tifName[cat],"_streams_r"),
                      direction = paste0(tifName[cat],"_dirs"))
)

execGRASS("r.stream.order",
        flags = c("overwrite", "quiet","z","m"),
        parameters = list(stream_rast = paste0(tifName[cat],"_streams_r"), # input
                      direction = paste0(tifName[cat],"_dirs"), # input
                      elevation = paste0(tifName[cat],"_Cond"), # input
                      accumulation = paste0(tifName[cat],"_Accum"), # input
                      stream_vect = paste0(tifName[cat],"_stream_v")), # output
        ignore.stderr=T)

# MiKatt: ESRI shape files must not have column names with more than 10 characters
execGRASS("v.db.renamecolumn", flags = "quiet",
        parameters = list(map = paste0(tifName[cat],"_stream_v"),
                      column = "next_stream,next_str")
)
# to keep column "next_str" next to prev_str
execGRASS("v.db.renamecolumn", flags = "quiet",
        parameters = list(map = paste0(tifName[cat],"_stream_v"),
                      column = "flow_accum, flow_accu")
)

# delete unused columns
execGRASS("v.db.dropcolumn", flags = c("quiet"),
        parameters = list(
          map = paste0(tifName[cat],"_stream_v"),
          columns = c("hack","topo_dim","scheidegger","drwal_old","stright",
                "sinosoid","source_elev","outlet_elev","elev_drop","out_drop","gradi
)
temp = readVECT(paste0(tifName[cat],"_stream_v"))
writeOGR(temp,".",paste0("./Hyderology/StreamOrder/",tifName[cat],"_stream_v"),
       driver = "ESRI Shapefile", overwrite_layer = TRUE);rm(temp)

# Slope, Aspect, slope-NE, slope-SE
execGRASS("r.slope.aspect", flags = c("overwrite"),
        parameters = list(elevation = tifName[cat],
```

```
                        slope = paste0(tifName[cat],"_Slope"),
                        aspect = paste0(tifName[cat],"_Aspect"),
                        dx = paste0(tifName[cat],"_EW_slope"),
                        dy = paste0(tifName[cat],"_NS_slope"),
                        tcurvature = paste0(tifName[cat],"_tCurv"))

  )

  writeRaster(raster(readRAST(paste0(tifName[cat],"_Slope"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_Slope"),
            format = "GTiff", overwrite = TRUE)

  writeRaster(raster(readRAST(paste0(tifName[cat],"_Aspect"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_Aspect"),
            format = "GTiff", overwrite = TRUE)

  writeRaster(raster(readRAST(paste0(tifName[cat],"_EW_slope"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_EW_slope"),
            format = "GTiff", overwrite = TRUE)

  writeRaster(raster(readRAST(paste0(tifName[cat],"_NS_slope"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_NS_slope"),
            format = "GTiff", overwrite = TRUE)

  writeRaster(raster(readRAST(paste0(tifName[cat],"_tCurv"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_tCurv"),
            format = "GTiff", overwrite = TRUE)

  #Topographic wetness index computation
  execGRASS("g.region", flags = c("overwrite","p","d"),
          parameters = list(raster = tifName[cat]),
          intern = TRUE)

  execGRASS("r.topidx", flags = c("overwrite"),
          parameters = list(input = tifName[cat],
                        output = paste0(tifName[cat],"_TWI"))
  )

  writeRaster(raster(readRAST(paste0(tifName[cat],"_TWI"))),
            filename =
               paste0("./Hyderology/DEM_Derivatives/",tifName[cat],"_TWI"),
            format = "GTiff", overwrite = TRUE)

  rm_grass(r.list())
  rm_grass(v.list(),"v")
  cat("\014")
  print(paste0("Catchments completed: ",cat))
}
```

Listing 15: *Extract hydrological characteristics using Grass GIS*

```
## Title :: Hydrologic characteristics extraction
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-23 23:18:46
# DOLE  :: 2019-07-23
```

```r
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ##############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(foreign)){install.packages("foreign");library(foreign)} # to read .dbf files

## Main code ##########################################################
xlData <- readxl::read_excel(path = "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet = "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")
fName <- paste0("./Hyderology/StreamOrder/",xlData$RiverBasins,"_",xlData$GaugeStations,"_str

Hydrology <- matrix(NA,nrow = length(Index), ncol = 9)
colnames(Hydrology) <- c("TOPWET","STREAMS_KM_SQ_KM","STRAHLER_MAX","PCT_1ST_ORDER",
                         "PCT_2ND_ORDER","PCT_3RD_ORDER","PCT_4TH_ORDER",
                         "PCT_5TH_ORDER","PCT_6TH_ORDER_OR_MORE")

## Function files
rm.na.ras <- function(filename,ext){
  temp <- as.vector(raster(paste0(filename,ext)))
  temp <- temp[!is.na(temp)]
  return(temp)
}

for(cat in 1:length(Index)){
  twiFname <- paste0("./Hyderology/DEM_Derivatives/",xlData$RiverBasins[Index[cat]],
                     "_",xlData$GaugeStations[Index[cat]])
  Hydrology[cat,1] <- mean(rm.na.ras(twiFname,"_TWI.tif"))

  #1. Stream density, km of streams per watershed sq km.
  Area         <- AllShpFile[[1]]$area
  shpDbf       <- read.dbf(fName[cat])
  Hydrology[cat,2] <- sum(shpDbf$length)/Area/1000
  Hydrology[cat,3] <- max(shpDbf$strahler)

  for(Order in 1:6){ # above 6th order are merged.
    if(Order <=5){
      Hydrology[cat,3+Order] <- sum(shpDbf$length[shpDbf$strahler %in% Order])/sum(shpDbf$length)
    }else{
      Hydrology[cat,3+Order] <- sum(shpDbf$length[shpDbf$strahler %in% c(6:max(shpDbf$strahler))])/sum(shpDbf$length)
    }
  }
  cat(cat,"\ ")
}
write.csv(Hydrology,file = "./Hyderology/Hydrology.csv",row.names = FALSE)
```

Listing 16: *Land cover extraction*

```r
## Title :: Land cover classification
```

```r
# Author :: Ankit Deshmukh
# DOC    :: 2019-07-17 19:57:30
# DOLE   :: 2019-07-17
# Remarks ::

## ClearUp and dir ####################################################
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(rgeos)){install.packages("rgeos");library(rgeos)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(viridis)){install.packages("viridis");library(viridis)}
## Main code ##########################################################

xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")
LandCover <- raster("./Landcover/SouthAsia_Grid_v4_LC/south_asia_v4")
LandCover@data@names <- "RasterLayer"
Table <- as.data.frame(LandCover@data@attributes)
Table$CLASS_NAMES <- gsub(" ","_", trimws(gsub('[[:punct:] ]+',"
    ",toupper(Table$CLASS_NAMES))))

Classes = unique(Table$ID)
LC_Count <- matrix(NA, nrow = length(Index), ncol = length(Classes))
colnames(LC_Count) <- Table$CLASS_NAMES

## Add Crop percentage 2005
India  <- readOGR(dsn = "./CatchmentDelineation/India_SHP_plotting", layer
    = "All_India")
CropLand <- raster("./Landcover/World cropland/CropLand.tif")
CropLand <- crop(x = CropLand, y = India@bbox )
CropLand@data@names <- "RasterLayer"


pdf("./Landcover/LC.pdf", width = 10, height = 16)
plot(CropLand,col = viridis(10), axes=FALSE, box=FALSE)
plot(India, add = T, border = "orange", lwd = 1)
dev.off()

LC_Count <- cbind(LC_Count,rep(NA,length(Index)))
colnames(LC_Count) <- c(colnames(LC_Count)[1:length(Classes)],"CROP_LAND")

for(cat in seq_along(Index)){
  #now extract the information in the grids
  temp <- extract(x          = LandCover,
                  y          = AllShpFile[[cat]],
                  weights    = TRUE,
                  normalizeWeights = TRUE,
                  df         = TRUE)

  LC_Ext <- temp$RasterLayer[!is.na(temp$RasterLayer)]
  LC_Wt  <- temp$weight[!is.na(temp$RasterLayer)]

  for(Type in 1:length(Classes)) {
    LC_Count[cat,Type] <- sum(LC_Wt[LC_Ext == Classes[Type]])*100
  }
  rm(temp,LC_Ext,LC_Wt)
```

```r

  temp <- extract(x        = CropLand,
                  y        = AllShpFile[[cat]],
                  weights      = TRUE,
                  normalizeWeights = TRUE,
                  df           = TRUE)

  LC_Count[cat,(length(Classes)+1)] <- sum(temp$RasterLayer[!is.na(temp$RasterLayer)]*temp$weight[!is.na(temp$RasterLayer)
  rm(temp)
  cat(cat,"\ ")
}

write.csv(LC_Count,file = "./Landcover/LandCover.csv", row.names = FALSE)
```

Listing 17: *Land use extraction*

```r
## Title :: Land use classification
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-17 19:57:30
# DOLE  :: 2019-07-17
# Remarks ::

## ClearUp and dir ######################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ##############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(rgeos)){install.packages("rgeos");library(rgeos)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(viridis)){install.packages("viridis");library(viridis)}

## Main code ############################################################
xlData <- readxl::read_excel(path = ./CatchmentDelineation/CatchmentDelineation.xlsx",sheet = "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")

LandUse <- raster("./Landuse/Anthropogenic World Biome/2000/anthro2_a2000.tif")
LandUse@data@names <- "RasterLayer"

if(!require(foreign)){install.packages("foreign");library(foreign)}
AttTbl = read.dbf("./Landuse/Anthropogenic World Biome/2000/anthro2_a2000.tif.vat.dbf")
detach("package:foreign", unload=TRUE)

AttTbl$LABEL <- gsub(" ","_", trimws(gsub('[[:punct:]]+|[[:digit:]]+',"",toupper(AttTbl$LABEL))))

Classes = unique(AttTbl$VALUE)
LU_Count <- matrix(NA, nrow = length(Index), ncol = length(Classes))
colnames(LU_Count) <- AttTbl$LABEL

India  <- readOGR(dsn = "./CatchmentDelineation/India_SHP_plotting", layer = "All_India", verbose = FALSE)

```

```
pdf("./Landuse/LU.pdf", width = 16, height = 8)
plot(LandUse,col = viridis(10), axes=FALSE, box=FALSE)
plot(India, add = T, border = "orange", lwd = 1)
dev.off()

for(cat in seq_along(Index)){
  #now extract the information in the grids
  temp <- extract(x        = LandUse,
                  y        = AllShpFile[[cat]],
                  weights  = TRUE,
                  normalizeWeights = TRUE,
                  df       = TRUE)

  LU_Ext <- temp$RasterLayer[!is.na(temp$RasterLayer)]
  LU_Wt  <- temp$weight[!is.na(temp$RasterLayer)]

  for(Type in 1:length(Classes)) {
    LU_Count[cat,Type] <- sum(LU_Wt[LU_Ext == Classes[Type]])*100
  }
  cat(cat,"\ ")
}

write.csv(LU_Count,file = "./Landuse/LandUse.csv", row.names = FALSE)
```

Listing 18: *Socio economic extracted with the following steps*

```
## Title :: Socio-economic properties extraction
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-22 22:12:34
# DOLE  :: 2019-07-22
# Remarks ::

## ClearUp and dir ##################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ##########################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(viridis)){install.packages("viridis");library(viridis)}
if(!require(ncdf4)){install.packages("ncdf4");library(ncdf4)}

## Main code ########################################################
xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")

pop <-
    raster("./SocioEconomic/WorldPopulation/Population-density-2000.tif")
hfi <-
    raster("./SocioEconomic/HumanFootprintIndex/The-Human-Footprint-Index.tif")
agri <-
    raster("./SocioEconomic/AgricultureSuitability/AgricultureSuitability.tif")
nCon <-
    raster("./SocioEconomic/FertilizerConsumption/Nitrogen_Consumption.tif")
pCon <-
    raster("./SocioEconomic/FertilizerConsumption/Phosphorus_Consumption.tif")
kCon <-
    raster("./SocioEconomic/FertilizerConsumption/Potassium_Consumption.tif")
```

```r
gdp <- "./SocioEconomic/GPP/GDP_PCPTA_PPP_1990_2015.nc"
hdi <- "./SocioEconomic/GPP/HDI_1990_2015.nc"

## Reading the NC file to get band information
NC <- nc_open(gdp);
print(NC)
Years <- ncvar_get(NC,"time")
missing_value <- -9
nc_close(NC);rm(NC)

## Save multiband raster in the memory
gdp <- stack(gdp)
hdi <- stack(hdi)

## Get column name R:576 | C:(6+2*26)
cols <-
    c("POP","HFI","AGRI","NITRO","PHOSF","POTAS",paste0("GDP_",Years),paste0("HDI_",Year
SocioEco_save <- matrix(NA, nrow = length(Index),ncol = length(cols))
colnames(SocioEco_save) <- cols

## Loop through all the catchments
for(cat in 1:length(Index)){ # [1:141] | [142:284] | [285:423] | [424:567]
  ## 1. Gridded population extraction
  pop@data@names <- "RasterLayer"
  temp <- raster::extract(x      = pop,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)

  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,1] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 2. Human Footprint Index extraction
  hfi@data@names <- "RasterLayer"
  temp <- raster::extract(x      = hfi,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)

  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,2] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 3. Agriculture Suitability gridded data extraction
  agri@data@names <- "RasterLayer"
  temp <- raster::extract(x      = agri,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)
  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,3] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 4. Nitrogen Consumption gridded data extraction
  nCon@data@names <- "RasterLayer"
  temp <- raster::extract(x      = nCon,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)
```

```r
  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,4] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 5. Phosphorus consumption gridded data extraction
  pCon@data@names <- "RasterLayer"
  temp <- raster::extract(x      = pCon,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)
  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,5] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 6. Potassium Consumption gridded data extraction
  kCon@data@names <- "RasterLayer"
  temp <- raster::extract(x      = kCon,
                          y      = AllShpFile[[cat]],
                          weights      = TRUE,
                          normalizeWeights = TRUE,
                          df           = TRUE)
  keepIndex <- !is.na(temp$RasterLayer)
  SocioEco_save[cat,6] <- sum((temp$RasterLayer*temp$weight)[keepIndex])

  ## 7. Gridded gross domestic product extraction
  for(Band in seq_along(Years)){
    gdp_slice <- gdp[[Band]]
    gdp_slice@data@names <- "RasterLayer"
    temp <- raster::extract(x      = gdp_slice,
                            y      = AllShpFile[[cat]],
                            weights      = TRUE,
                            normalizeWeights = TRUE,
                            df           = TRUE)

    keepIndex <- temp$RasterLayer != missing_value
    SocioEco_save[cat,(6+Band)] <-
        sum((temp$RasterLayer*temp$weight)[keepIndex])
  }

  ## 8. Human developement index extraction
  for(Band in seq_along(Years)){
    hdi_slice <- hdi[[Band]]
    hdi_slice@data@names <- "RasterLayer"
    temp <- raster::extract(x      = hdi_slice,
                            y      = AllShpFile[[cat]],
                            weights      = TRUE,
                            normalizeWeights = TRUE,
                            df           = TRUE)

    keepIndex <- temp$RasterLayer != missing_value
    SocioEco_save[cat,(6+length(Years)+Band)] <-
        sum((temp$RasterLayer*temp$weight)[keepIndex])
    cat(Band,"\ ")
  }
  cat("\n",cat,"\n")
}

write.csv(SocioEco_save,file = "./SocioEconomic/SocioEconomic.csv",
    row.names = FALSE)
```

Listing 19: *Various soil layers for extraction*

```r
## Title :: Soil data processing value extraction and projection
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-11 11:34:02
# DOLE  :: 2019-07-11
# Remarks ::
## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(readxl)){install.packages("readxl");library(readxl)}
if(!require(doParallel)){install.packages("doParallel");library(doParallel)}
## Main code ##########################################################
xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index = which(xlData$IndexForDB == 1)

# AllShpFile <- list()
# for(cat in seq_along(Index)){
#  AllShpFile[cat] <- readOGR(paste0(getwd(),
    "/CatchmentDelineation/Shapefiles950"),
#
    paste0(xlData$RiverBasins[Index[cat]],"_",xlData$GaugeStations[Index[cat]]),verbose
    = FALSE)
#  print(cat)
# }
# save(AllShpFile,file = "AllShpFile567.R")

load("./CatchmentDelineation/AllShpFile567.RData")

dirName <- c("BLD","CLYPPT","CRFVOL","ORCDRC","SLTPPT","SNDPPT")
mnLayer <- c("D2.5cm","D10cm","D22.5cm","D45cm","D80cm","D150cm")
tileNums <- c(350,386,387,421,422,423,424,458,459)

SoilData <- matrix(NA,nrow = length(Index),ncol =
    length(dirName)*length(mnLayer))

colName <- c()
for(ii in 1: length(dirName)){colName <- append(colName,
    paste0(dirName[ii],"_",mnLayer))}
colnames(SoilData) <- colName; rm(colName)

kk = 1
for(Dir in seq_along(dirName)){
  for(Layers in seq_along(mnLayer)){
    a = Sys.time() # Time computation
    RasterLayer <- list()
    for(Tile in tileNums){
      RasterLayer <- append(RasterLayer,
                        list(raster(paste0("./Soil/",dirName[Dir],
                                    "/",dirName[Dir],
                                    "_sd",Layers,"_M_1km_T",Tile,".tif"))))
    }
    # merging soil tiles for whole India Dir: Dx and Layer Lx
    mergeRaster <- do.call(merge, RasterLayer);rm(RasterLayer)
    Vals <- rep(NA,length(Index))
    for(cat in seq_along(Index)){
```

```
    shpFile <- AllShpFile[[cat]]
    temp <- extract(x          = mergeRaster,
                    y          = shpFile,
                    weights    = TRUE,
                    normalizeWeights = TRUE,
                    df         = TRUE)
    Vals[cat] <- sum(temp$layer*temp$weight, na.rm = TRUE)
    rm(shpFile,temp)
    cat(cat,'\n')
   }
   SoilData[,kk] <- Vals
   kk <- kk +1
   print(Sys.time()-a)
  }
}

write.csv(SoilData,file = "./Soil/Soil.csv",row.names = FALSE)
```

Listing 20: *Porosity and permeability dataset extraction*

```
## Title :: Porosity and permeability crop then covert to raster for extraction
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-24 19:00:07
# DOLE  :: 2019-07-24
# Remarks ::

## ClearUp and dir ##################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ##########################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(rgdal)){install.packages("rgdal");library(rgdal)}

## Main code ########################################################
xlData <- readxl::read_excel(path = "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet = "PourPoint950")
Index = which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")
ppShpData <- readOGR(dsn = "./Soil/PORPER",layer="PorPerWGSTras")

Extent    <- extent(ppShpData)
rr        <- raster(Extent, res=0.01)
Porosity  <- rasterize(ppShpData, rr, field = "Porosity")
Permeability <- rasterize(ppShpData, rr, field = "Permeabili")

PorPer <- matrix(NA,nrow = length(Index), ncol = 2)
colnames(PorPer) <- c("Porosity","Permeability")

Porosity@data@names <- "rasterLayer"
Permeability@data@names <- "rasterLayer"

for(cat in seq_along(Index)){
 shpFile <- AllShpFile[[cat]]
 temp <- extract(x          = Porosity,
                 y          = shpFile,
                 weights    = TRUE,
                 normalizeWeights = TRUE,
```

```
              df            = TRUE)
  PorPer[cat,1] <- sum(temp$rasterLayer*temp$weight, na.rm = T)

  temp <- extract(x         = Permeability,
              y             = shpFile,
              weights       = TRUE,
              normalizeWeights = TRUE,
              df            = TRUE)
  PorPer[cat,2] <- 10^(sum(temp$rasterLayer * temp$weight, na.rm = T)/100)
  cat(cat,'\ ')
}

write.csv(PorPer,file = "./Soil/PorPer.csv",row.names = FALSE)
```

Listing 21: *Topographical characteristics extraction*

```
## Title :: Topologic characteristic extraction
# Author :: Ankit Deshmukh
# DOC   :: 2019-07-23 12:47:43
# DOLE  :: 2019-07-23
# Remarks ::

## ClearUp and dir ####################################################
graphics.off(); rm(list = ls()); cat("\014")
setwd("D:/AnkitDeshmukh/GoogleDrive/Database01/Database101")
## Load required libraries ############################################
if(!require(tidyverse)){install.packages("tidyverse");library(tidyverse)}
if(!require(raster)){install.packages("raster");library(raster)}
if(!require(viridis)){install.packages("viridis");library(viridis)}

## Main code ##########################################################
xlData <- readxl::read_excel(path =
    "./CatchmentDelineation/CatchmentDelineation.xlsx",sheet =
    "PourPoint950")
Index <- which(xlData$IndexForDB == 1)
load("./CatchmentDelineation/AllShpFile567.RData")

fName <-
    paste0("./Hyderology/DEM_Derivatives/",xlData$RiverBasins,"_",xlData$GaugeStations)[

Topography <- matrix(NA,nrow = length(Index), ncol = 14)
colnames(Topography) <-
    c("ELEV_MEAN_M_BASIN","ELEV_MEDIAN_M_BASIN","ELEV_MAX_M_BASIN","ELEV_MIN_M_BASIN",
                    "ELEV_STD_M_BASIN","RRMEAM","RRMEDIAN","ASPECT_DEGREES","ASPECT_NORT
                    "ASPECT_EASTNESS","NS_SLOPE","EW_SLOPE","SLOPE_PCT","TAN_CURVE")

## Function files
rm.na.ras <- function(filename,ext){
  temp <- as.vector(raster(paste0(filename,ext)))
  temp <- temp[!is.na(temp)]
  return(temp)
}

aspectCompFun <- function(angle){
  if(angle == 0){
    compAsp = 0
  }else if(angle <= 90){
    compAsp = (90 - angle)
  }else(compAsp = (360 + 90 - angle))
  return(compAsp)
```

```
}

## Iteration for catchments
for(cat in seq_along(Index)){
  demVect <- rm.na.ras(fName[cat],'.tif')

  Topography[cat,1] <- mean(demVect)
  Topography[cat,2] <- median(demVect)
  Topography[cat,3] <- max(demVect)
  Topography[cat,4] <- min(demVect)
  Topography[cat,5] <- sd(demVect)
  Topography[cat,6] <-
      (mean(demVect)-min(demVect))/(max(demVect)-min(demVect))
  Topography[cat,7] <-
      (median(demVect)-min(demVect))/(max(demVect)-min(demVect))

  ## Aspect products
  meanAspect <- mean(rm.na.ras(fName[cat],"_Aspect.tif"))
  Topography[cat,8] <- meanAspect
  Topography[cat,9] <- cos(aspectCompFun(meanAspect))
  Topography[cat,10] <- sin(aspectCompFun(meanAspect))
  Topography[cat,11] <- mean(rm.na.ras(fName[cat],"_NS_slope.tif"))
  Topography[cat,12] <- mean(rm.na.ras(fName[cat],"_EW_slope.tif"))
  Topography[cat,13] <- mean(rm.na.ras(fName[cat],"_Slope.tif"))
  Topography[cat,14] <- mean(rm.na.ras(fName[cat],"_tCurv.tif"))
  cat(cat,"\n")
}

write.csv(Topography,file = "./Topography/Topography.csv",row.names =
    FALSE)
```

References

Tom Gleeson, Nils Moosdorf, Jens Hartmann, and LPH Van Beek. A glimpse beneath earth's surface: Global hydrogeology maps (glhymps) of permeability and porosity. *Geophysical Research Letters*, 41(11):3891–3898, 2014.

Jens Hartmann and Nils Moosdorf. The new global lithological map database glim: A representation of rock properties at the earth surface. *Geochemistry, Geophysics, Geosystems*, 13(12), 2012.

Tomislav Hengl, Jorge Mendes de Jesus, Robert A MacMillan, Niels H Batjes, Gerard BM Heuvelink, Eloi Ribeiro, Alessandro Samuel-Rosa, Bas Kempen, Johan GB Leenaars, Markus G Walsh, et al. Soilgrids1km—global soil information based on automated mapping. *PloS one*, 9(8):e105992, 2014.

N Ramankutty, AT Evan, C Monfreda, and JA Foley. Global agricultural lands: Croplands, 2000. *Data distributed by the Socioeconomic Data and Applications Center (SEDAC)[online]: http://sedac. ciesin. columbia. edu/data/set/aglands-croplands-2000 (accessed on 16 november 14)*, 2010.

HJ Stibig, R Upik, R Beuchle, Mubareka S Hildanus, and S Mubareka. The land cover map for south-east asia in the year 2000. *European Commision Joint Research Centre: Ispra, Italy) Available at http://www. gvm. jrc. it/glc2000 [Verified November 2006]*, 2003.