1. OVERVIEW

Air temperature measurements are of great importance as their variations are often related to human morbidity and mortality. However, spatial coverage of air temperature (Ta) by weather stations are often limited. By building a model between available Land Surface Temperature (LST) derived from satellite images and corresponding air temperature, we can therefore predict air temperature for later epidemiological analysis. The methodology was to:

1) build 10km grid points across the entire span of Texas;
2) process LST, wind speed, elevation, Normalized Difference Vegetation Index (NDVI), Percent of Impervious Surfaces on the grid point-based buffers
3) built a mixed regression model for Ta~ (all above variables) based on grids that have complete information
4) predict Ta where Ta are missing

2. METHODOLOGY

2.1 Build Buffer Files in ArcGIS

2.1.1 10km* 10km Grid Points File

File path: U:\Research\Kai_group\Jie\Buffers_10\gridpts\GridPtsWCoor.shp

Description: Total 6845 center points of 6845 10km*10km grids covering whole Texas. For contents in the attribute table, ORIG_FID is the variable to identify each grid point based on its geological location (which is essential for all the following variable processing), X is the longitude in decimal degrees, and Y is the Latitude in decimal degrees.

GridPtsWCoor

FID	Shape *	X	Y	ORIG_FID
0	Point	-97.430777	25.884896	0
1	Point	-97.531207	25.97457	1
2	Point	-97.432267	25.975991	2
3	Point	-97.333323	25.977316	3
4	Point	-97.234377	25.978547	4
5	Point	-97.829913	26.060805	5
6	Point	-97.73088	26.062511	6
7	Point	-97.631844	26.064122	7
8	Point	-97.532803	26.065638	8
9	Point	-97.43376	26.067059	9
10	Point	-97.334713	26.068386	10
11	Point	-98.32744	26.141882	11
12	Point	-98.228325	26.144064	12
13	Point	-98.129206	26.146151	13
14	Point	-98.030083	26.148143	14
15	Point	-97.930955	26.15004	15
16	Point	-97.831822	26.151843	16
17	Point	-97.732686	26.15355	17
18	Point	-97.633546	26.155162	18
19	Point	-97.534403	26.15668	19
20	Point	-97.435256	26.158102	20
21	Point	-97.336106	26.159429	21
22	Point	-98.52829	26.228234	22
23	Point	-98.429082	26.230608	23
24	[illegible]	[illegible]	[illegible]	[illegible]

0 (0 out of 6845 Selected)

2.1.2 Grid Points-based Buffer Files

(Tentatively) all needed buffer radii are listed in the table below:

Category	Variables	Units	Buffer radii
Land use	Source: Imperviousness 2011	%	50, 100, 150, 300, 400, 500, 750, 1000, 1500, 3000, 5000m
NDVI	Source: NASA MOD13A3 product		
	Monthly NDVI 2011	NDVI	50, 100, 150, 300, 400, 500, 750, 1000, 1500, 3000, 5000m

Process:

All needed buffer files are generated majorly using arcpy.Buffer_analysis.

Detailed python code is attached in Appendix 1.

Final outputs: A list of generated buffer files could be found at:

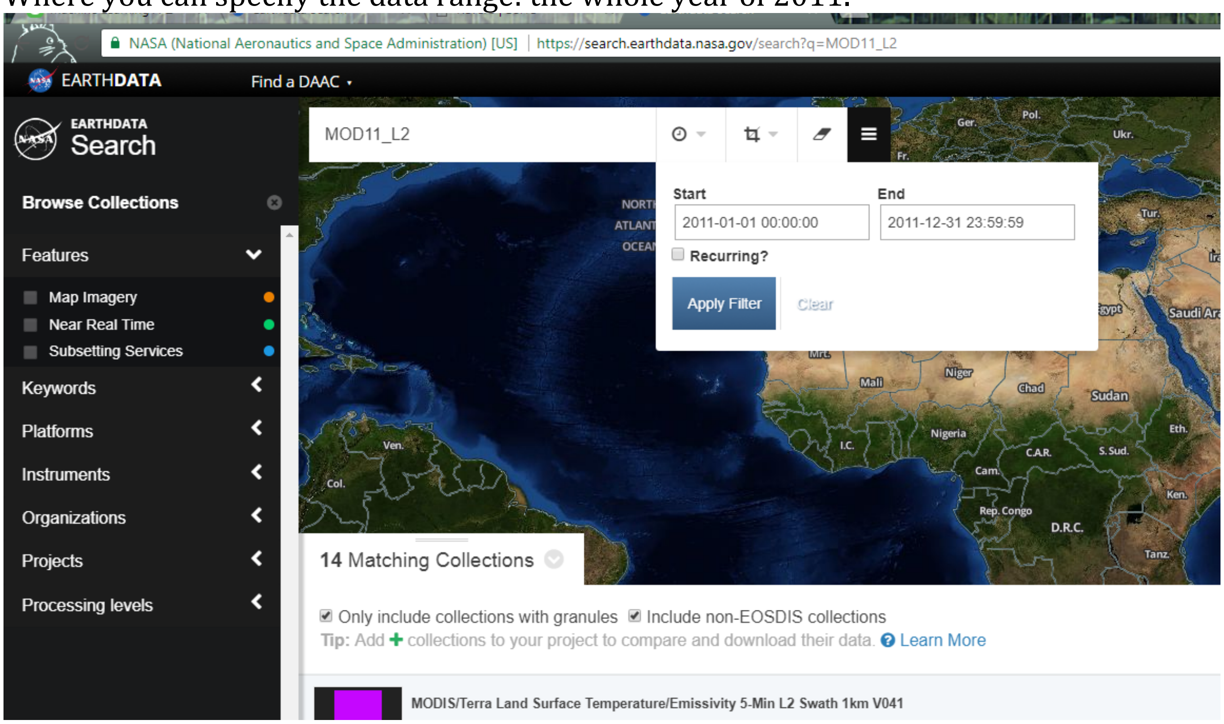
U:\Research\Kai_group\Jie\Buffers_10

File names indicate the radius.

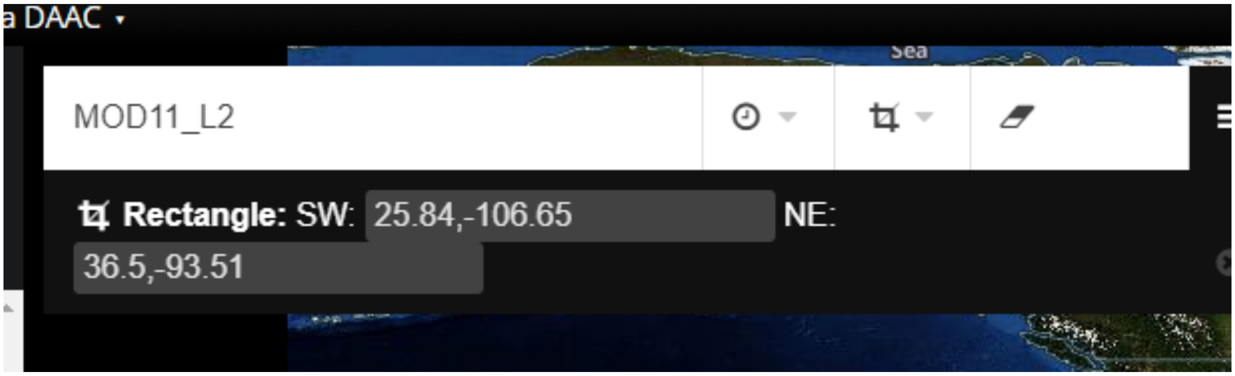
2.2 Pre-process LST Satellite Image Files

2.2.1 Downloading data:

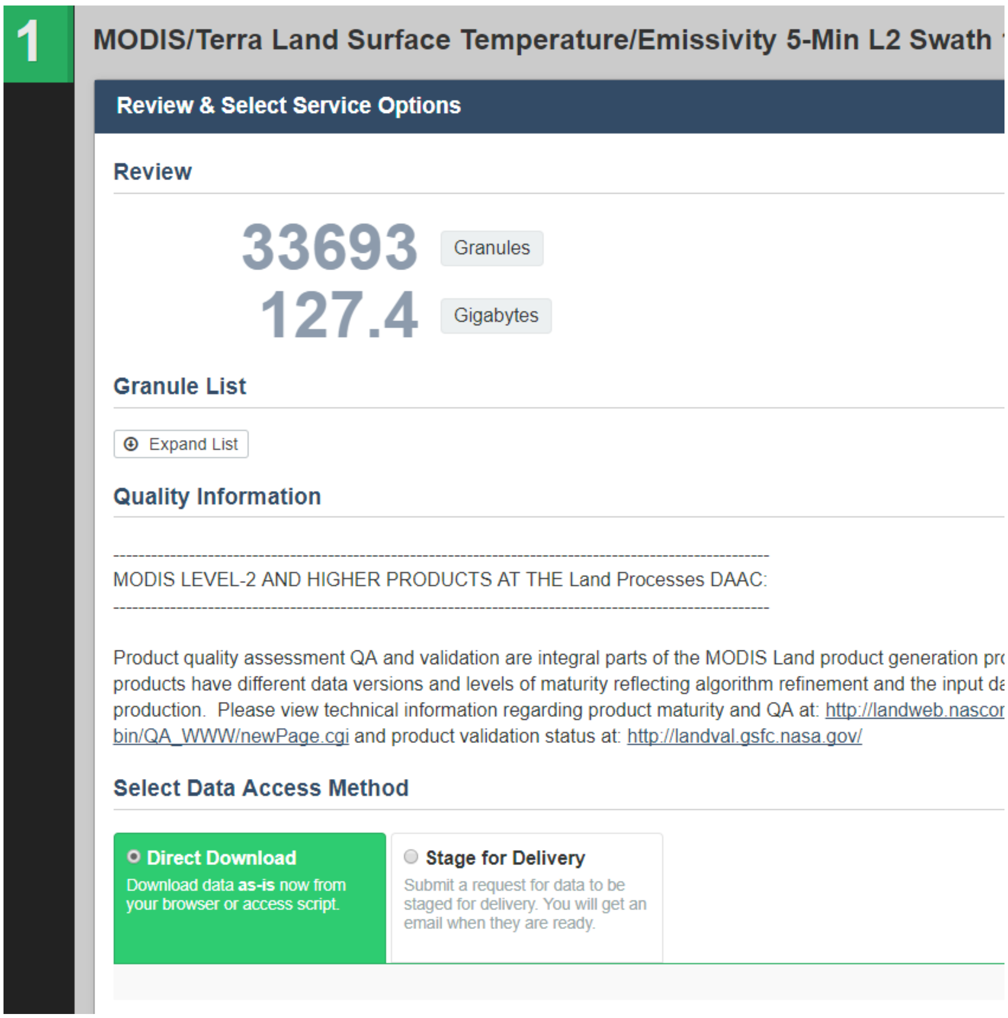
The data source is MODIS/Terra Land Surface Temperature and Emissivity 5-Minute L2 Swath 1 km V005.
Data description could be found at
https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod11_l2
Downloading link:
https://search.earthdata.nasa.gov/search?q=MOD11_L2
Where you can specify the data range: the whole year of 2011.



And the Texas region bounding box:



Then click on download:



Then a list of available downloadable links will appear:
https://search.earthdata.nasa.gov/granules/download.html?project=8374630052&collection=C108956785-LPDAAC_ECS
All downloaded data is stored at path: (Portable drive) D:\2011LST

The naming convention for the downloaded swath data:

PPPPPPPP.AYYYYDDD.HHMM.VVV.YYYYDDDHHMMSS.hdf

PPPPPPPP= Product Short Name

AYYYYDDD = Julian Date of Acquisition

HHMM = Hours & Minutes of Acquisition

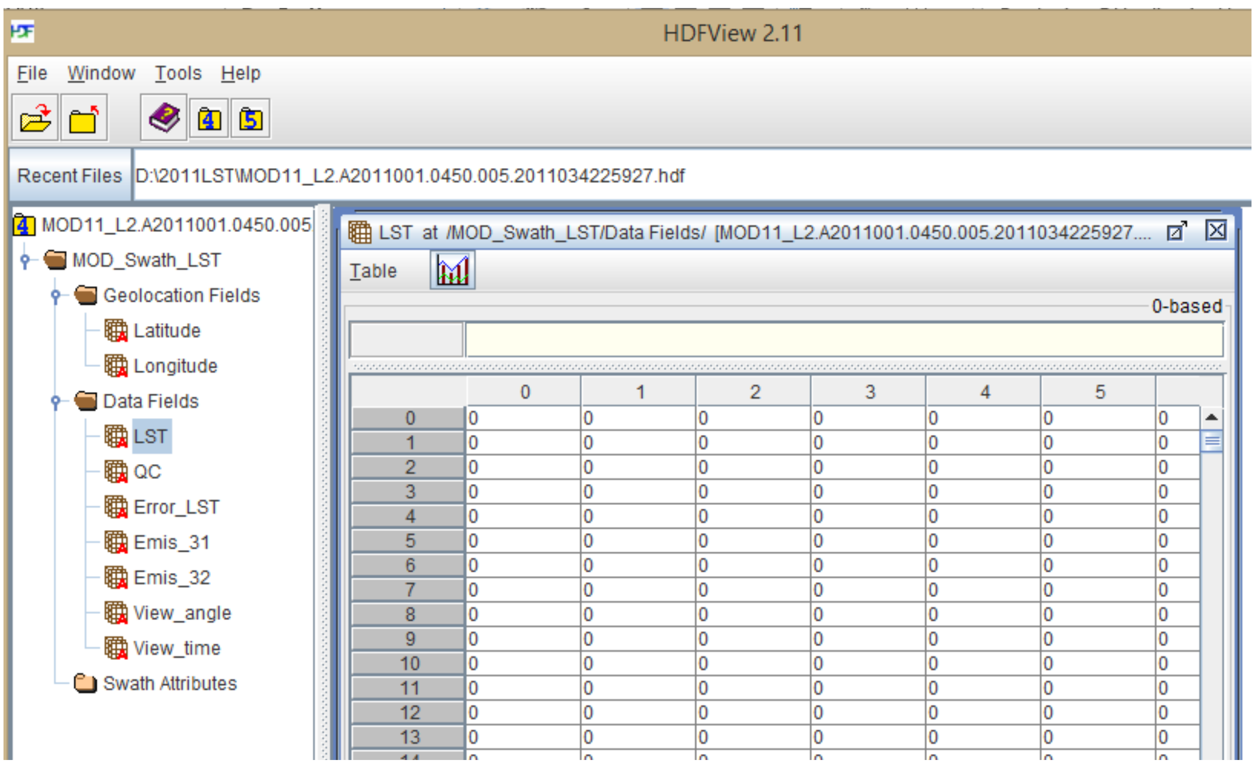
VVV = Version Number

YYYYDDDHHMMSS = Julian Date of Production

hdf = Data Format

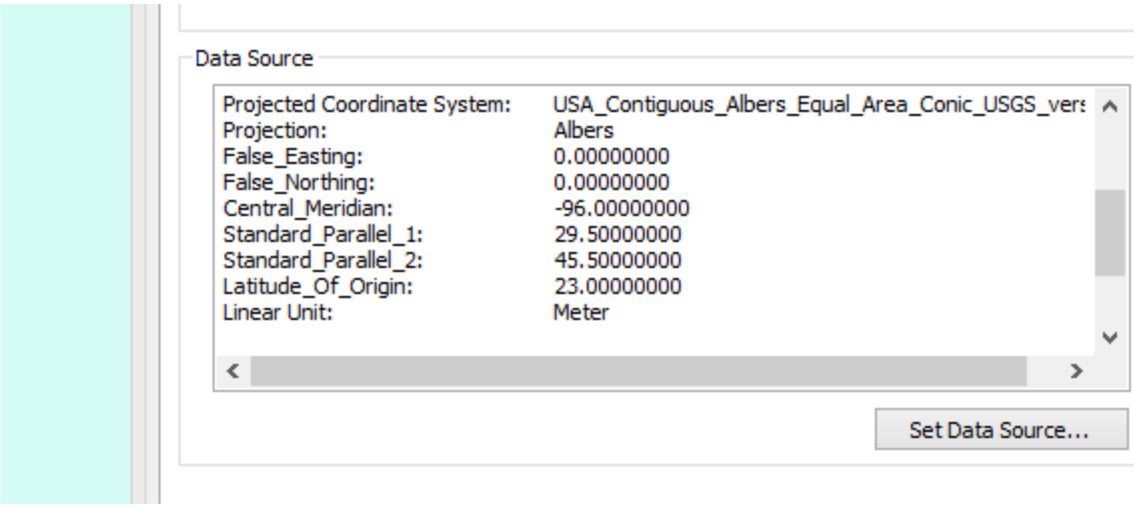
2.2.2 Extracting Geographical information with ENVI/IDL

The satellite data downloaded are in the format of HDF4, which is a library and multi-object hierarchical file format. We can use HDFView 2.11 software to look into different layers of information that files have, and we can get the data structure shown as below:



The problem with HDF4 file is that the hierarchical file cannot be directly utilized by ArcGIS, and by nature although the coordinates were recorded in the file, the file itself is un-projected. MCTK plug-in within ENVI/IDL can re-project satellite products into raster files with geographic information one file at a time, which can be further processed in ArcGIS. Based on supplemental information for MCTK at https://github.com/dawhite/MCTK, I'm

using USA Albers Equal Area Conic projection (parameters as detailed in the screenshot below) to do batch transformations of HDF4 files into georeferenced .dat files. More reference on how to definite different projections could be found at
http://gridkr.com/d/ENVI_4_3/online_help/ENVI_PROJ_CREATE.html



The detailed code is attached in Appendix 2.
Final processed files are at the path:
U:\Research\Kai_group\Jie\2011LST_processing
Final files are named using convention: 'MOD11_L2.'+ 'A'+(Date and time of acquisition)+'_Swath_2D_1_georef.dat'

2.2.3 Using python for processing into point shapefiles with LST reading

As converted .dat files are raster files with multiple bands, we want to extract/keep all the bands needed in the converted shapefiles.
First is to clip the .dat raster files to Texas boundary as previously we downloaded data to Texas bounding box, which is basically a rectangle encompassing Texas.
Code is detailed in Appendix 3.
Processed files path:
U:\Research\Kai_group\Jie\2011LST_processing\clippedraster
All the tiles are then converted to point shapefiles using raster to points. Detailed code is attached in Appendix 4. However, as this step was done through multiple computers, the shapefiles are pretty messy in the end, and therefore additional code is needed to clean up and check for completeness.
Code is attached in Appendix 5.
Final processed files path:

U:/Research/Kai_group/Jie/2011LST_processing/shp
After the conversion to shapefiles, I'm using python to again extract all 3 bands (LST, Band 31 Emissivity and Band 32 Emissivity) values from the corresponding raster files to the point features previously created.
Code is detailed in Appendix 6, and final processed file paths are still:
U:/Research/Kai_group/Jie/2011LST_processing/shp
Now the files are point shapefiles, with 3 band values attached to the attribute table.
Band_1, Band_2, Band_3 indicate LST, Band 31 Emissivity and Band 32 Emissivity respectively.

Table

0011700

FID	Shape	pointid	grid_code	Band_1	Band_2	Band_3
32178	Point	321783	269.76	-9999	-9999	-9999
32178	Point	321784	269.76	-9999	-9999	-9999
32178	Point	321785	269.76	-9999	-9999	-9999
32178	Point	321786	269.76	-9999	-9999	-9999
32178	Point	321787	269.76	-9999	-9999	-9999
32178	Point	321788	269.76	-9999	-9999	-9999
32178	Point	321789	269.76	-9999	-9999	-9999
32178	Point	321790	269.76	-9999	-9999	-9999
32179	Point	321791	269.76	-9999	-9999	-9999
32179	Point	321792	269.76	-9999	-9999	-9999
32203	Point	322034	269.92	269.92	0.49102	0.49502
32203	Point	322035	269.92	-9999	-9999	-9999
32203	Point	322036	269.92	-9999	-9999	-9999
32203	Point	322037	269.92	-9999	-9999	-9999
32203	Point	322038	269.92	-9999	-9999	-9999
32203	Point	322039	269.92	-9999	-9999	-9999
32203	Point	322040	269.92	-9999	-9999	-9999
32204	Point	322041	269.92	-9999	-9999	-9999
32204	Point	322042	269.92	-9999	-9999	-9999
32204	Point	322043	269.92	-9999	-9999	-9999
32204	Point	322044	269.92	-9999	-9999	-9999
32204	Point	322045	269.92	-9999	-9999	-9999
32204	Point	322046	269.92	-9999	-9999	-9999
32204	Point	322047	269.92	-9999	-9999	-9999
32204	Point	322048	269.92	-9999	-9999	-9999
32214	Point	322147	270.06	270.06	0.49302	0.49702
32214	Point	322148	270.06	-9999	-9999	-9999
32214	Point	322149	270.06	-9999	-9999	-9999

Appendix

1)

```
import arcpy
import os
from arcpy import env

# Set environment settings
env.workspace = "C:/data"

for dis in [50, 100, 150, 300, 400, 500, 750, 1000, 1500, 3000, 5000, 7500, 10000, 15000]:

  fishnet= "U:/Research/Kai_group/Jie/Buffers_10/gridpts/gridpts.shp"
  outpath= "U:/Research/Kai_group/Jie/Buffers_10/"
  Buffer = outpath+str(dis)+"meters"+ ".shp"
  distance = str(dis)+" meters"
  sideType = "FULL"
  endType = "ROUND"
  dissolveType = "NONE"
  arcpy.Buffer_analysis(fishnet, Buffer, distance, sideType, endType, dissolveType)
```

2)

```
PRO mctk_batch_lst


  COMPILE_OPT IDL2
  catch,Error_status
  if Error_status ne 0 then begin
    void = Dialog_Message(!Error_State.Msg, /error)
    return
  endif

  ; Initialize ENVI
  ; Send all errors and warnings to the log file
  envi, /restore_base_save_files
  envi_batch_init, log_file='reproject_mod11_L2.log'

  ;do not put the bridge creation/destruction code inside a loop
  bridges = mctk_create_bridges()


  cd,'U:\Research\Kai_group\Jie\2011LST\'
```

```
modis_l2_file = FILE_SEARCH("*.hdf")


IF SIZE(modis_l2_file, /N_ELEMENT) GE 1 THEN BEGIN

  FileCount = SIZE(modis_l2_file, /N_ELEMENT)
  PRINT, FileCount,' image(s) will be precessed'



  FOR i = 0, FileCount-1 DO BEGIN
    PRINT, 'START : ',SYSTIME()
    tmp_filename = modis_l2_file[i]
    print, '"'+ tmp_filename+'"'+' is being processed'


    fname = 'U:\Research\Kai_group\Jie\2011LST\'+tmp_filename

    output_location = 'U:\Research\Kai_group\Jie\2011LST_processing\'

    output_rootname = STRMID(tmp_filename,0,22)

    ;Output method schema is:
    ;0 = Standard, 1 = Projected, 2 = Standard and Projected
    out_method = 1

    swath_name = 'MOD_Swath_LST'

    sd_names = ['LST','Emis_31','Emis_32']

    ; =0 Nearest neighbor, =1 Linea,=2 Cubic Convolution interpolation
    interpolation_method = 0


    ; Albers Conical Equal Area projection for conversion
    name=  'Albers Conical Equal Area 84’
    output_datum = 'WGS-84'
    params = [9, 6378137.0, 6356752.3,  $
      23.000000, -96.000000, $
      0.0, 0.0, $
      29.500000, 45.500000]
    output_projection=ENVI_PROJ_CREATE(type=9, $
      name=name, datum=output_datum, params=params)


    convert_modis_data, in_file=fname,$
      out_path=output_location,out_root=output_rootname,$
      swt_name=swath_name, sd_names=sd_names,$
      out_method=out_method,out_proj=output_projection,$
      interp_method=interpolation_method,/no_msg,$
      r_fid_array=r_fid_array, $
      r_fname_array=r_fname_array, bridges=bridges, msg=msg
    PRINT, 'END : ', SYSTIME()
```

```
    ENDFOR
  ENDIF

  mctk_destroy_bridges, bridges

  ;ENVI_BATCH_EXIT
  PRINT, 'Reproject_MOD11_L2 Done!'
  ;return

END
```

3)

```
import arcpy
import os
from arcpy import env
import re

# Set environment settings
arcpy.env.workspace = "C:/data"

#first step just split

path1 = "U:/Research/Kai_group/Jie/2011LST_processing"
path2 = "U:/Research/Kai_group/Jie/2011LST_processing/clippedraster/"
path3 = "U:/Research/Kai_group/Jie/Boundary/Texas_albersNAD.shp"
shp_list = []
tile_id = []
for dirpath, dirnames, files in os.walk(path1):
 for f in files:
  if f.lower().endswith(".dat"):
   fullpath = os.path.join(dirpath, f)
   shp_list.append(fullpath)
   tile_id.append(f[14:17]+f[18:22])

for i in range(0, 2147):
 in_raster = shp_list[i]
 Rectangle = "#"
 arcpy.Clip_management(in_raster, Rectangle, path2+tile_id[i]+".tif", path3, "#", "ClippingGeometry",
"MAINTAIN_EXTENT")
```

4)

```
import arcpy
import os
```

```
from arcpy import env
import re

# Set environment settings
arcpy.env.workspace = "C:/data"

#Convert to point features
shp_list = []
tile_id = []
path2 = "U:/Research/Kai_group/Jie/2011LST_processing/clippedraster/"
output= "U:/Research/Kai_group/Jie/2011LST_processing/shp/"
for dirpath, dirnames, files in os.walk(path2):
 for f in files:
  if f.lower().endswith(".tif"):
   fullpath = os.path.join(dirpath, f)
   shp_list.append(fullpath)
   tile_id.append(f[:-4])

for i in range(0, 2147):
 in_raster = shp_list[i]
 arcpy.RasterToPoint_conversion(shp_list[i], output+tile_id[i]+".shp", "VALUE")
```

5)

```
import arcpy
import os
from arcpy import env

# Set environment settings
arcpy.env.workspace = "C:/data"


path1 = "U:/Research/Kai_group/Jie/2011LST_processing/shp"
path2 = "U:/Research/Kai_group/Jie/2011LST_processing/clippedraster"


shp_list= []
name_list=[]
shp_list2= []
name_list2=[]

for dirpath, dirnames, files in os.walk(path1):
 for f in files:
  if f.lower().endswith(".shp"):
   fullpath = os.path.join(dirpath, f)
```

```
  shp_list.append(fullpath)
  name_list.append(f[:-4])
#length:2025

for dirpath, dirnames, files in os.walk(path2):
 for f in files:
  if f.lower().endswith(".tif"):
   fullpath = os.path.join(dirpath, f)
   shp_list2.append(fullpath)
   name_list2.append(f[:-4])
#length:2146
#shapefile shortage: 2146- 2025= 121


d1_contents = set(name_list)
d2_contents = set(name_list2)
common = list (d1_contents & d2_contents)

missing = []
for f in name_list2:
 if f not in common:
    print f
    missing.append(f)
```

2730450
2730455
2731655
2731700
2731835
2740355
2740400
2740535
2740540
2741740
2741745
2750440
2750445
2751645
2751650
2751825
2760345
2760350
2760520
2760525
2761730
2770425
2770430
2771630
2771635

2771810
2771815
2780335
2780510
2780515
2781715
2781720
2790415
2790420
2790555
2791620
2791625
3151740
3160435
3160440
3161640
3161645
3161815
3161820
3170340
3170515
3170520
3171720
3171725
3180420
3180425
3180600
3181625
3181630
3181805
3181810
3190330
3190505
3190510
3191710
3191715
3191845
3191850
3200410
3200415
3200545
3200550
3201615
3201620
3201750
3201755
3210450
3210455
3211655

3211700
3211835
3220355
3220400
3220535
3220540
3221740
3221745
3230440
3230445
3231645
3231650
3231825
3240345
3240350
3240520
3240525
3241725
3241730
3250425
3250430
3251630
3251635
3251810
3251815
3260335
3260510
3260515
3261715
3261720
3270415
3270420
3270555
3271620
3271625
3271800
3280325
3280500
3281705
3281840
3281845
3290405
3290540
3290545
3291610
3291745
3291750

```
for dirpath, dirnames, files in os.walk(path2):
 for f in files:
  if f.lower().endswith(".tif"):
   fullpath = os.path.join(dirpath, f)
   shp_list2.append(fullpath)
   name_list2.append(f[:-4])

output= "U:/Research/Kai_group/Jie/2011LST_processing/shp/"
# somehow it doesn't work
for i in range(0, 2147):
 in_raster = shp_list2[i]
 if name_list2[i] in missing:
   arcpy.RasterToPoint_conversion(shp_list2[i], output+name_list2[i]+".shp", "VALUE")
```

6)

```
import arcpy
import os
from arcpy import env
import re
from arcpy import env
from arcpy.sa import *

# Set environment settings
arcpy.env.workspace = "C:/data"

path1 = "U:/Research/Kai_group/Jie/2011LST_processing/shp/"
path2 = "U:/Research/Kai_group/Jie/2011LST_processing/clippedraster/"

shp_list = []
name_list = []
shp_list2 = []
name_list2 = []

for dirpath, dirnames, files in os.walk(path1):
 for f in files:
  if f.lower().endswith(".shp"):
   fullpath = os.path.join(dirpath, f)
   shp_list.append(fullpath)
   name_list.append(f[:-4])

for dirpath, dirnames, files in os.walk(path2):
 for f in files:
```

```
 if f.lower().endswith(".tif"):
  fullpath = os.path.join(dirpath, f)
  shp_list2.append(fullpath)
  name_list2.append(f[:-4])



# Check out the ArcGIS Spatial Analyst extension license
arcpy.CheckOutExtension("Spatial")


#Extract 3 band values from the rasters based on names cuz corresponding shapefiles have the same names as the raster files they were converted from
for i in range(0, 2146):
 inPointFeatures= shp_list[i]
 fname=name_list[i]
 multibandRaster=path2+fname+".tif"
 desc = arcpy.Describe(multibandRaster)
 bands = desc.bandCount
 in_rasters = []
 for band in desc.children:
   bandName = band.name
   #append each band to the in_rasters list
   in_rasters.append(os.path.join(multibandRaster, bandName))
 ExtractMultiValuesToPoints(inPointFeatures, in_rasters)
```