Find prime real estate using satellites images

Motivation

Bangalore (https://en.wikipedia.org/wiki/Bangalore) has been touted as the Sillicon Valley of India. The city has seen rapid growth in the last decade. The booming tech ecosystem has influenced real estate changes in various parts of the city. Identifying such changes presciently can yield great financial returns.

Using Landsat 8 satellite images of Bangalore city I looked at each pincode area and tried to identify urbanization signals. Below is my approach and results.

Data Source

Landsat 8 (http://landsat.usgs.gov/landsat8.php) images were used; Specifically images belonging to Path 144 and Row 51 which encompasses Bangalore city

Bangalore Shape Files (http://openbangalore.org/available-data/) were utilized to clip each Landsat image to a particular pincode area

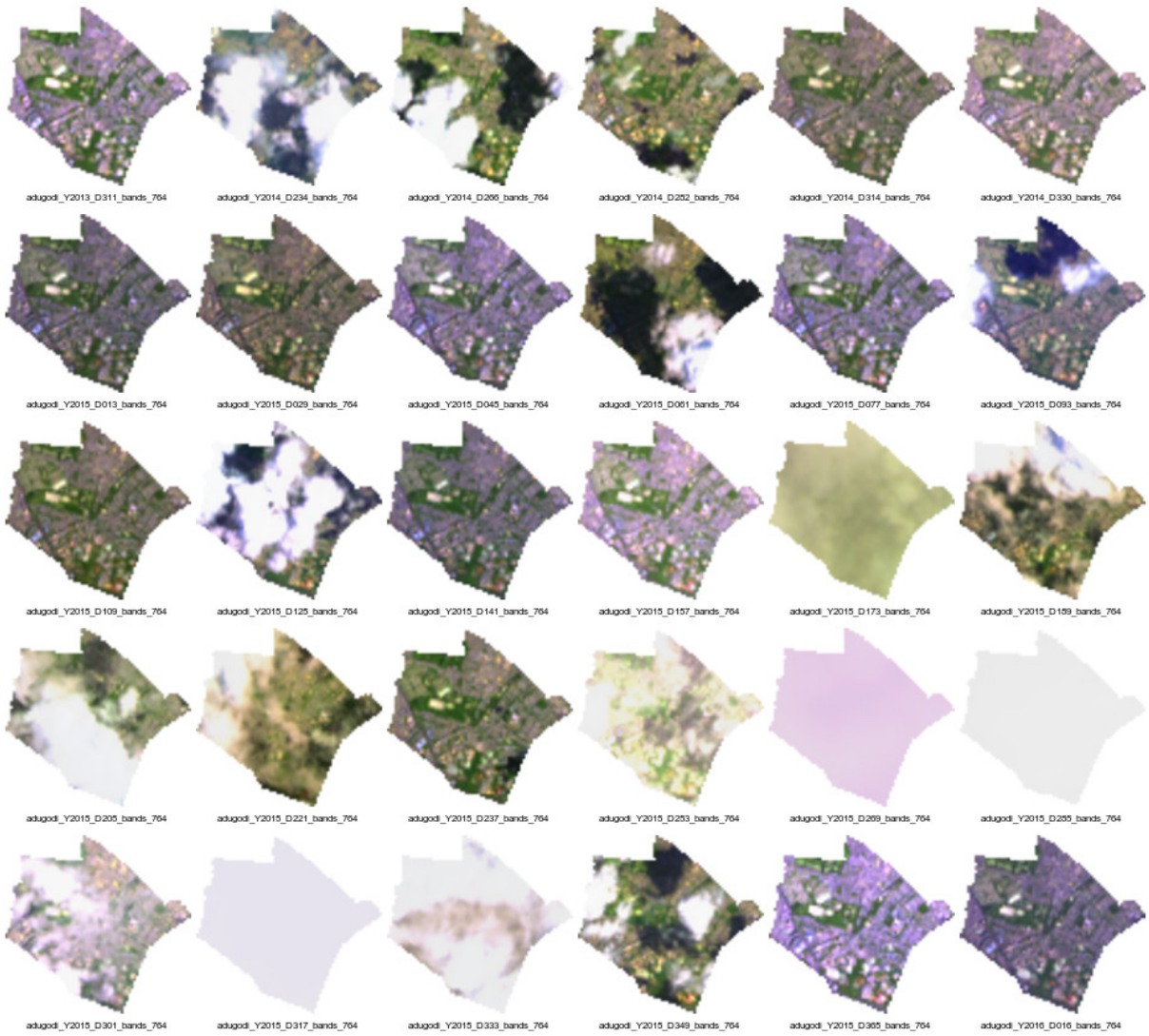
Pre-processing

- Landsat 8 images were obtained from aws (https://aws.amazon.com/public-data-sets/landsat/)
- I found that bands 7 6 4 are good for detecting urbanization patterns (http://www.exelisvis.com/Home/NewsUpdates/TabId/170/ArtMID/735/ArticleID/14305/The-Many-Band-Combinations-of-Landsat-8.aspx)
- ogr2ogr (http://www.gdal.org/ogr2ogr.html) to convert json files to .shp files.
- gdalwarp (http://www.gdal.org/gdalwarp.html) to clip a large satellite image and extract only the area of interest.
- landsat-util (https://github.com/developmentseed/landsat-util) was used to process the varios Landsat 8 bands into a single image. This docker image (https://hub.docker.com/r/developmentseed/landsat-util/) really helped.

Below is a montage of images for the pincode area of Adugodi (selected randomly) in Bangalore city. Pink pixels denote urbanization. For each row and path, 30 images are available in Landsat 8 archive. Most of the images below seem unusable due to excess cloud cover on some and complete blanks on others.

In [5]:
```
from IPython.display import Image
Image(filename='/Users/hsinghal/workspace/ERRATA/LANDSAT/shape_files/adugodi/montage_adugodi.jpg')
```

Out[5]:



The files obtained after clipping the larger Landsat image with the pincode area shape files consists of 4 layers. The first 3 are red, green and blue. These have to be resolved to a single number to help with subsequent analysis.

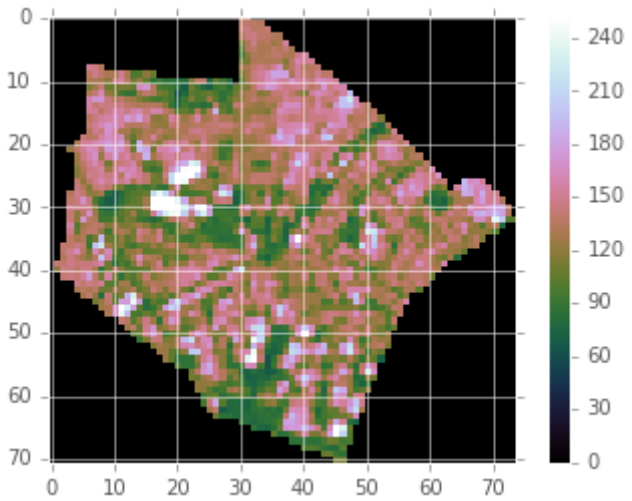
For each image (corresponding to a particular date) the formula ***0.2126xRed + 0.7152xGreen + 0.0722xBlue*** was used to extract values of each pixel.

In [23]:
```
%matplotlib inline
import rasterio
import numpy as np
from skimage import io
import matplotlib
matplotlib.style.use('ggplot')

def getLuminance(rgb):
    """
    Ref:
    http://stackoverflow.com/questions/596216/formula-to-determine-brighness-of-rgb-color
    """
    return (0.2126* rgb[0]) + (0.7152*rgb[1]) + (0.0722*rgb[2])

src = rasterio.open('/Users/hsinghal/workspace/ERRATA/LANDSAT/shape_files/adugodi/adugodi_Y2013_D311_bands_764.TIF')
r, g, b, rnd = src.read()
processedPixels = np.zeros(r.shape)
for (x, y), value in np.ndenumerate(r):
    processedPixels[x,y] = getLuminance([r[x,y], g[x,y], b[x,y]])

io.imshow(processedPixels)
io.show()
```



After visually inspecting many such images I chose pixel values falling betwenn 120 to 180 as those that identify "urban" pixels.

Regression Analysis

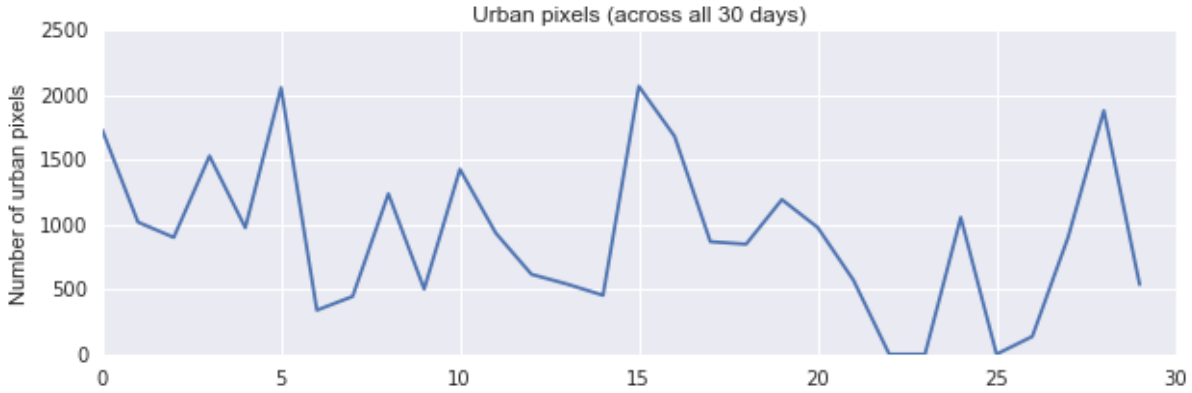
- For each pincode area and date, count number of "urban" pixels
- Consider only those dates for which the count of "urban" pixels are within 30% of the earliest available date. This is to filter out images which have excessive cloud cover or are empty.

In [62]:
```
# Below are the number of urban pixels counted for pincode area Adugodi.
import pandas as pd
adugodiUrbanPixels_df = pd.read_table('/Users/hsinghal/workspace/ERRATA/
LANDSAT/shape_files/adugodi/UrbanPixelsCount.txt')
adugodiUrbanPixels_df.ix[:,[0,1]]
```

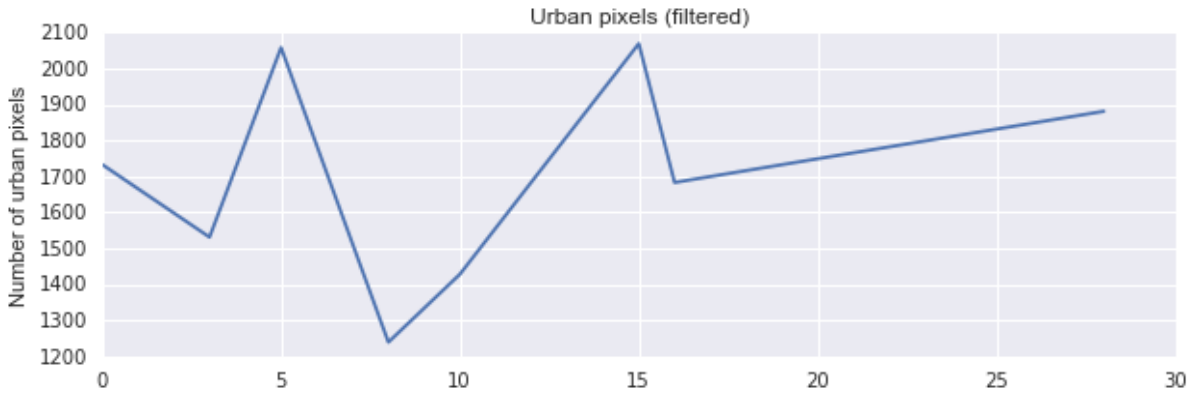
Out[62]:

	YearJulianDay	NumberPixelsUrban
0	2013311	1733
1	2014234	1020
2	2014266	902
3	2014282	1531
4	2014314	977
5	2014330	2058
6	2015013	340
7	2015029	446
8	2015045	1239
9	2015061	503
10	2015077	1428
11	2015093	936
12	2015109	617
13	2015125	541
14	2015141	455
15	2015157	2069
16	2015173	1683
17	2015189	869
18	2015205	850
19	2015221	1194
20	2015237	979
21	2015253	572
22	2015269	0
23	2015285	0
24	2015301	1056
25	2015317	0
26	2015333	137
27	2015349	903
28	2015365	1881
29	2016016	538

In [59]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,3))
sns.set_style("darkgrid")
plt.plot(adugodiUrbanPixels_df.NumberPixelsUrban)
plt.ylabel("Number of urban pixels")
plt.title("Urban pixels (across all 30 days)")
plt.show()
```



In [61]:
```
# Filter out days that are not within +/- 30% of the first days' value
adugodiFiltered = adugodiUrbanPixels_df[(adugodiUrbanPixels_df.NumberPix
elsUrban/1733 <= 1.3) & (adugodiUrbanPixels_df.NumberPixelsUrban/1733 >=
0.7)]
plt.figure(figsize=(10,3))
sns.set_style("darkgrid")
plt.plot(adugodiFiltered.NumberPixelsUrban)
plt.ylabel("Number of urban pixels")
plt.title("Urban pixels (filtered)")
plt.show()
```



Very few data points remain due to the filtering process above and fitting a regression model on a small number of data points is not a good idea.

Nevertheless, I'm using linear regression here to "indicate" if a pincode area exhibits increased urbanization.

NumberOfUrbanPixels ~ time

In [75]:
```
pinCodeTrends  = pd.read_csv("/Users/hsinghal/workspace/ERRATA/LANDSAT/shape_files/allLinearModels.csv")
# Shown only those pincodes that exhibit a positive trend and less than
 0.30 prob. of significance (chosen arbitrarily)
pinCodeTrends[(pinCodeTrends.trendProb <= 0.30) & (pinCodeTrends.trendCoeff > 0) ]
```

Out[75]:

	pinCodeName	InterceptProb	trendProb	intercept	trendCoeff	numPoints
7	ashoknagar	0.005377	0.224292	2563.500000	119.800000	4
8	vyalikaval_extn	0.027388	0.241805	1714.800000	186.400000	5
12	bangalore_air_port	0.000003	0.272956	5913.362637	98.923077	14

Images used (after filtering) for *vyalikaval_extn*

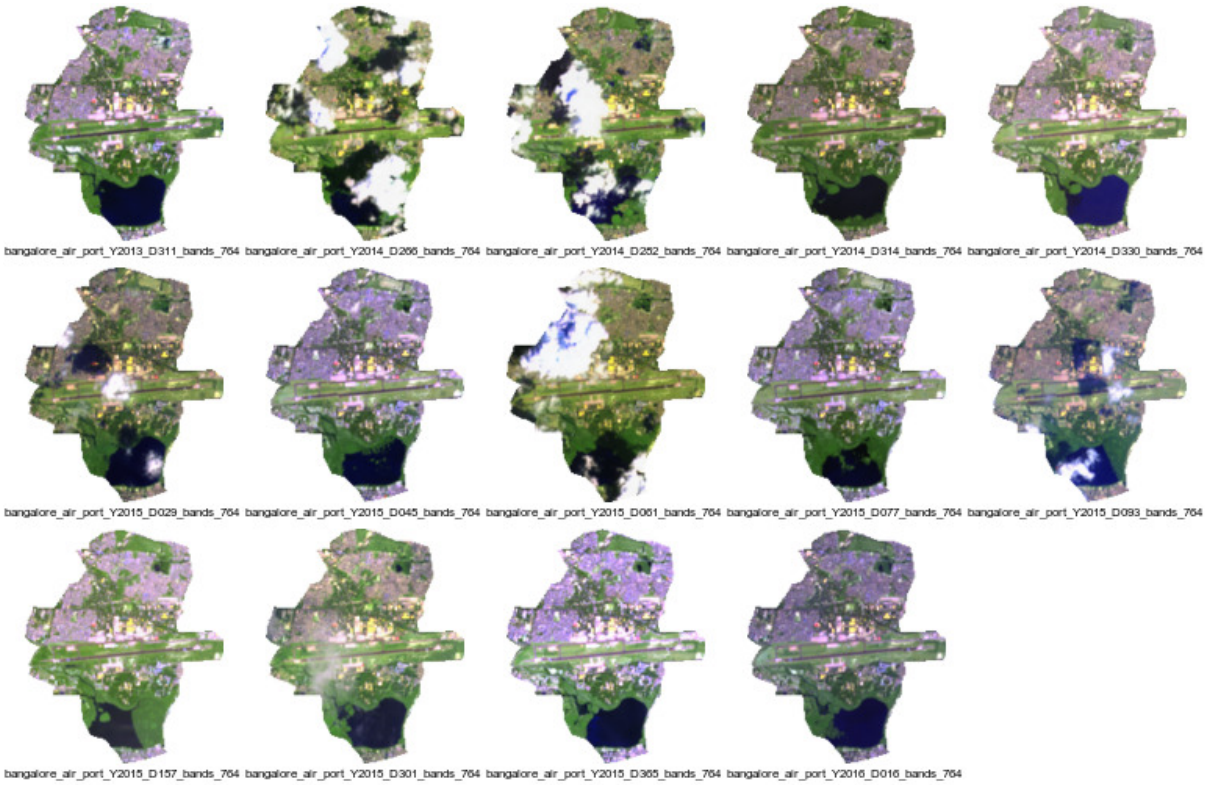
In [76]:
```
from IPython.display import Image
Image(filename='/Users/hsinghal/workspace/ERRATA/LANDSAT/shape_files/vyalikaval_extn/montage.jpg')
```

Out[76]:

Images used (after filtering) for *bangalore_air_port*

In [77]:
```
from IPython.display import Image
Image(filename='/Users/hsinghal/workspace/ERRATA/LANDSAT/shape_files/bangalore_air_port//montage.jpg')
```

Out[77]:



Conclusion

I chose Bangalore becuase of my familiarity with the city. The final results are promising and the pincode areas identified are indeed areas where urbanization has seen an uptick. The "bangalore_air_port" area especially has seen a lot of construction of hotels and business parks around the airport facility.

Issues

- Landsat 8 images are low-res and are not amenable to reliably identifying features such as buildings or structures that could improve detection of land development/urbanization
- Inclusion of the panchromatic band with bands 432 improves resolution to 15 m but is still not enough
- Landsat 8 images are available only since late 2013 and many of them are unusable due to cloud cover or other technical glitches that render them blank. This results in very little data for analysis.

Future Work

- I ranked the pincode areas based on the intercept value (descending). The top areas are associated with pincodes that are geographically large and the low ranked pincodes are associated with very small geographical areas. A suitable normalization needs to be applied based on physical area. Some of the low ranked ones are staff housing complexes of large public sector companies characterized by a few buildings and ample lawns and gardens
- Identify growth areas by clustering on urbanization markers and not rely on pre-defined pincode boundaries. Track cluster centers over time. In high density cities like Bangalore, new areas attract rapid growth due to saturation of former urban hotspots.
- Acquire more granual imagery (DigitalGlobe API beta program ?) and identify buildings to further classify into housing, commercial (high-rise, large parking lot), etc. Train Deep Learning models on lebelled image data.

Appendix

Python code to extract pincode area shape files and clip the larger Landsat 8 file to the pincode shape

```
import json
import re
import subprocess
import string
import numpy
import os
l8nameFormat = 'LXSPPPRRRYYYYDDDGSIVV'
landsat_images = ['LC81440512013311LGN00',
'LC81440512014234LGN00','LC81440512014266LGN00','LC81440512014282LGN00','LC
81440512014314LGN00',
'LC81440512014330LGN00','LC81440512015013LGN00','LC81440512015029LGN00','LC
81440512015045LGN00',
'LC81440512015077LGN00','LC81440512015093LGN00','LC81440512015109LGN00','LC
81440512015125LGN00',
'LC81440512015141LGN00','LC81440512015157LGN00','LC81440512015173LGN00','LC
81440512015189LGN00',
'LC81440512015205LGN00','LC81440512015221LGN00','LC81440512015237LGN00','LC
81440512015253LGN00',
'LC81440512015269LGN00','LC81440512015285LGN00','LC81440512015301LGN00','LC
81440512015317LGN00',
'LC81440512015333LGN00','LC81440512015349LGN00','LC81440512015365LGN00','LC
81440512016016LGN00',
'LO81440512015061LGN00']

with open('bangalore_pincode.json') as data_file:
    all_bangalore_pincodes = json.load(data_file)

finalResFileOut = open('finalResultsFile.txt','a')
all_subdirs = list(os.walk('.'))[0][1]
# 1. For each pin code, create json file
# 2. Use json file to create shape files
# 3. clip the bangalore large file to the shape files for all landsat image
s
for eachPinCode in all_bangalore_pincodes['features']:
    newDict = {}
    newDict['type'] = all_bangalore_pincodes['type']
    newDict['features'] = [eachPinCode]
    pin_area_name = eachPinCode['properties']['name']
    cleaned_name = ''.join([k.lower() for k in pin_area_name if k not in st
ring.punctuation])
    # name of the area
    cleanedPinCodeName= re.sub("\\s+","_",cleaned_name)
    newDirCreate = cleanedPinCodeName
    createDir = "mkdir " + newDirCreate
    if newDirCreate not in  all_subdirs:
        subprocess.call(createDir, shell=True)
        # Write to json file
        print "writing json file"
        shapePinCode_json = newDirCreate + '/' + cleanedPinCodeName + '-sha
```

```
pe.json'
        print shapePinCode_json
        openJsonFile = open(shapePinCode_json, 'w')
        json.dump(newDict, openJsonFile)
        openJsonFile.close()
        # Take each json file and convert each file to its shape files
        shapePinCode_shpfile = newDirCreate + '/' + cleanedPinCodeName + '-
shape.shp'
        runJsonToShapeFileConverter_Cmd = 'ogr2ogr -nlt POLYGON -skipfailur
es ' + shapePinCode_shpfile + ' ' + shapePinCode_json + ' ' + 'OGRGeoJSON -
overwrite'
        print runJsonToShapeFileConverter_Cmd
        subprocess.call(runJsonToShapeFileConverter_Cmd, shell=True)
        # Clip for this particular pincode all the files
        # Take each band file and extract pincode
        for eachImage in landsat_images:
            yrval = ''.join([k[1] for k in zip(l8nameFormat,eachImage) if k
[0] =='Y'])
            juliandayval = ''.join([k[1] for k in zip(l8nameFormat,eachImag
e) if k[0] =='D'])
            cmdToRun = 'gdalwarp -cutline ' + shapePinCode_shpfile + ' -cro
p_to_cutline -dstalpha ../bands764_banfalore_tifs/processed/' + eachImage+
'/' + eachImage + '_bands_764.TIF ' + newDirCreate + '/' + cleanedPinCodeNa
me + '_Y'+ yrval + '_D' + juliandayval+'_bands_764.TIF'
            print cmdToRun
            subprocess.call(cmdToRun, shell=True)
```

Python code to extract pixel values for each pincode area

```
import numpy as np
import rasterio
import subprocess
import os
import re

def getLuminance(rgb):
    """
    Ref:
    http://stackoverflow.com/questions/596216/formula-to-determine-brightne
ss-of-rgb-color
    """
    res = (0.2126* rgb[0]) + (0.7152*rgb[1]) + (0.0722*rgb[2])
    return round(res,4)

allSubDirectories = list(os.walk('.'))[0][1]
for eachSubDir in allSubDirectories:
    resultsFile = eachSubDir+'_pixelValues.DATA'
    resultsFileOpen = open(resultsFile, 'w')
    tiffiles = [k for k in os.listdir('./' + eachSubDir) if 'TIF' in k]
    for eachFile in tiffiles:
        with rasterio.drivers():
            with rasterio.open(eachSubDir + '/' + eachFile) as src:
                r, g, b, rnd = src.read()
                # Initialize empty array
                outputArray = np.zeros(r.shape)
                for (x, y), value in np.ndenumerate(r):
                    rgblist = [r[x,y], g[x,y], b[x,y]]
                    outputArray[x,y] = photometric(rgblist) # Scoring funct
ion
        outputFlatten = outputArray.flatten()
        outputFlattenNonZero = outputFlatten[outputFlatten > 0.0]
        resMatch = re.match(".*_Y(\d+){1,}_D(\d+){1,}_.*", eachFile)
        yr = resMatch.group(1)
        dd = resMatch.group(2)
        for eachPixelValue in outputFlattenNonZero:
            resList = [eachSubDir, yr, dd, str(round(eachPixelValue))]
            resultsFileOpen.write('\t'.join(resList) + '\n')
    resultsFileOpen.close()
```

R code to fit a linear regression model - count of urban pixels vs time

```
# There are a total of 100 pincode areas
# Pixel values for all dates per pincode area are saved in a file
# Each pincode file is read and urban pixels are identified and counted
dataFiles <- list.files(".","DATA")
allDf <- NULL
for (eachFile in dataFiles){
  pinCodeName <- gsub("_pixelValues.DATA","",eachFile)
  print(pinCodeName)
  upperUrbanThreshold <- 180
  lowerUrbanThreshold <- 120
  d <- read.csv(eachFile,header=F,sep="\t",stringsAsFactors=FALSE)
  names(d) <- c("area","yr","dd","pixval")
  grpedData <- group_by(d, area, yr, dd)
  summData <- summarise(grpedData,
                numVals= n(),
                nDist = n_distinct(pixval),
                varVal = var(pixval),
                meanVal = mean(pixval),
                medianVal = median(pixval),
                numPinks = sum(pixval >= lowerUrbanThreshold & pixval <= upp
erUrbanThreshold))
  summData <- summData[order(summData$yr_day),]
  # Keep those days where the numPinks are 30% varying from day 1
  upperBound <- summData$numPinks[1] * 1.3
  lowerBound <- summData$numPinks[1] * 0.70
  summData <- summData[summData$numPinks >= lowerBound & summData$numPinks
<= upperBound,]
  # Order and fit lm
  summData <- summData[order(summData$yr_day),]
  summData$nPoints <- 1:nrow(summData)
  lmVal <- summary(lm(numPinks ~ nPoints, data = summData))
  tempDf <- data.frame(areaName =pinCodeName,
                       InterceptProb = lmVal$coefficients[,4][1][[1]],
                       nPointsProb = lmVal$coefficients[,4][2][[1]],
                       InterceptVal =   lmVal$coefficients[,1][1][[1]],
                       nPointsVal =   lmVal$coefficients[,1][2][[1]],
                       numPoints = nrow(summData))
  allDf <- rbind(allDf, tempDf)
}
```