STAT 585X - Final Project - Draft Report
Changes in cultivated cropland soil in Iowa

Andreea Erciulescu

April 6, 2014

1 Introduction

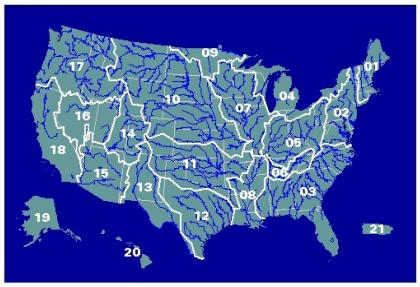
1.1 Question of interest

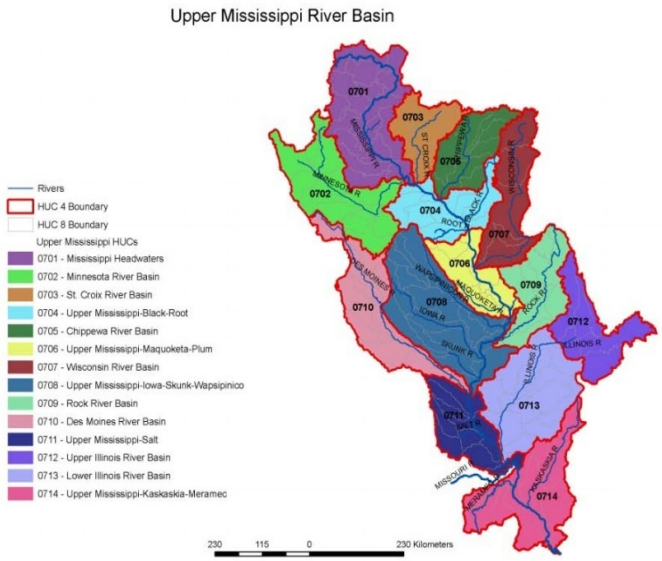
Do soil characteristics changes over time match the Conservation Effects Assessement Project (CEAP) survey design and data collection?

1.2 Motivation

National Resources Inventory (NRI) is an annual survey conducted collaboratively by USDA NRCS (Natural Resources Conservation Services) and ISU Center for Survey Statistics and Methodology (CSSM) to provide status and trend estimates for natural resources on nonfederal lands in US. Example of such estimates are soil erosion estimates in relation to land characteristics and programs.

Conservation Effects Assessment Project (CEAP) is a series of surveys intended to quantify environmental effects of conservation prctices and programs by hydrologic unit codes (HUCs). The following image illustrates the division of the United States territory into 2-digits HUCs and the sudivision of the Upper Mississippi region into 4-digit HUCs.





The CEAP sample is a subset of the NRI points classified as cultivated cropland. The data was collected using farmers interviews and NRCS hydrologic, climate and soil databases. The data was then

filtered through an APEX model (black box) and the result is a set of observations for usable points for 19 response variables. The final goal is to produce erosion estimates for 8-digit HUCs.

CEAP is run at both national and regional levels. Among the regional levels, the Des Moines River Watershed (HUC 0710) is a region of interest in this project, that is subdivided into 9 8-digit HUCs.

For analysis, data from the Soil Survey is considered, collected over the 2003 to 2006 time period. Hence, the survey frame, consisting of NRI cultivated cropland points, is set in 2003. The information for these eligible points is collected in the following years, 2004-2007.

1.3 Data

The CEAP sample data for the Des Moines River Watershed (HUC 0710) region is not publicly available, so in the class project I will consider the following sources of information:

- Crop Data Layer (CDL)

  The CDL data is available at http://nassgeodata.gmu.edu/CropScape/ in the form of Tagged Image File (.tif) Format. We are interested in the state of Iowa data, available for the years of 2003-2007. The information consists of pixel counts and acreage values for different category of cropland data. Each of the category has an associated value (code), for example 1 stands for Corn and 5 stands for Soybean. A complete list of category codes, class names and colors for the USDA NASS CDL is available at http://www.nass.usda.gov/research/Cropland/docs/CDL$_2$013$_c$*rosswalk.htm*.

- Census Topologically Integrated Geographic Encoding and Referencing database (Tigerweb)

  The Census Tigerweb data is available at http://tigerweb.geo.census.gov/tigerwebmain for both national and regional leveles. Also, data for the hydrologic levels is available at

  http://tigerweb.geo.census.gov/tigerwebmain/Files/tigerweb$_t$*ab*10$_h$*ydro*$_p$*oly*$_i$*a.html*. We are interested in the state of Iowa data, as well as the Des Moines River data. In particular, we are interested in the points coordinates.

- Public Land Survey System (PLSS)

  The PLSS data can be found on http://http://www.geocommunicator.gov/GeoComm/lsis$_h$*ome/home/index.htm* in the form of shapefiles. Information is available at both state and county levels.

- GIS data on hydrologic basins

  The GIS data can be found at ftp://ftp.igsb.uiowa.edu/gis$_l$*ibrary/basins/* in the form of shapefiles. Information is available for the entire Des Moines River basin.

- Atlas of historical countyu boundaries (AtlasHCB)

  The AtlasHCB data is available at http://publications.newberry.org/ahcbp/pages/Iowa.html in the form of shapefiles. Information os available for the entire state of Iowa.

2 Data Collection and Processing Steps

We are going to explore all these different data sources and decide on the most useful one. We are interested in a very specific region in the state of Iowa so we would have to be careful at utilizing the data. Also, the files have very large dimensions, so we would have to be careful at selecting the amount of useful information.

2.1 CDL data

We first download the data for years 2003-2007 from the website http://nassgeodata.gmu.edu/CropScape/. We have five .tif files, one for each year of interest. These are raster graphics images, spatial data structures that divide the US teritory into pixels that store crop information. This type of data are referred to as a 'grid,' contrasted with 'vector' data is used to represent points, lines, polygons. The dimensions of files are very large [1].

An useful R package to read and manipulate these data is the *Raster* package. This package allows us to read the raster values from the files and to convert the cell numbers to coordinates and back.

```
setwd("U:/classes/585X")
library("raster")

## Loading required package:  sp

# ?raster ####S4 methd

## read the data
cdl.ia03 <- raster("data/cdl/CDL_2003_19.tif")

## rgdal:  version:  0.8-14, (SVN revision 496)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime:  GDAL 1.10.1, released 2013/08/26
## Path to GDAL shared files:  C:/Program Files/R/R-3.0.2/library/rgdal/gdal
## GDAL does not use iconv for recoding strings.
## Loaded PROJ.4 runtime:  Rel.  4.8.0, 6 March 2012, [PJ_VERSION: 480]
## Path to PROJ.4 shared files:  C:/Program Files/R/R-3.0.2/library/rgdal/proj

cdl.ia04 <- raster("data/cdl/CDL_2004_19.tif")
cdl.ia05 <- raster("data/cdl/CDL_2005_19.tif")
cdl.ia06 <- raster("data/cdl/CDL_2006_19.tif")
cdl.ia07 <- raster("data/cdl/CDL_2007_19.tif")

cdl.ia03

## class       : RasterLayer
## dimensions  : 11672, 17796, 207714912  (nrow, ncol, ncell)
## resolution  : 30, 30  (x, y)
## extent      : -52065, 481815, 1938165, 2288325  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96 +x_0=0 +y_0=0 +ellps=GRS80
## data source : U:\classes\585X\data\cdl\CDL_2003_19.tif
## names       : CDL_2003_19
## values      : 0, 255  (min, max)
```

Notice the attributes of the raster objects. One of the most important ones for us is the coordinate reference system (CRS), or the map projection. We would need this in order to carefully manipulate the future data on the region of interest to get the matching coordinate reference (more details follow). The pjcets we have so far are 'skeletons,' because they only contain attributes of the data, not the actual values stored. We are going to read the values for the region of interest using cell numbers

[1] After saving the data to my U drive I received an e-mail from the university alerting me that I have filled 99% of my Cyfiles. Hence, I opened a Cybox account and started to move other folders over.

and coordinates (xy) in the extraction method from the *cellFromXY*. For this, we need the coordinates for the Des Moines River Watershed and surrounding area.

2.2 GIS, PLSS, AtlasHCB data

We first download the data, very small dimensions in coparison the the raster data. We are going to read in these data using the *maptools* library in R, as we did in class. We extract the polygons information from these shapefiles using functions presented in class. Also, we could use the function developed in one of the labs in order to read in the polygon information for all the counties in Iowa (as we did with the district in Australia).

GIS

```
setwd("U:/classes/585X")

library(ggplot2)
library(maps)
library(plyr)

xx <- readShapeSpatial("data/igsb/basin.shp")
xxx <- thinnedSpatialPoly(as(xx, "SpatialPolygons"), tolerance = 0.1, minarea = 0.001, topologyPres
class(xxx)
slotNames(xxx)

polys <- xxx@polygons[[1]]

length(polys@Polygons)

this.poly <- polys@Polygons[[1]]

# gets the coords for this polygon
poly.coords <- as.data.frame(this.poly@coords)

# plot the area
qplot(x, y, data = poly.coords, geom = "polygon")

# convert the coordinates to the matching raster system
loc.newcoords <- project(as.matrix(poly.coords), proj = "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=2

### convert universal transverse mercador (UTM) coordinates to longitude-latitude coordinates
SP <- SpatialPoints(cbind(poly.coords[, 2], poly.coords[, 1]), proj4string = CRS("+proj=utm +zone=1
loc.newcoords <- spTransform(SP, CRS("+proj=longlat"))

loc.newcoords <- as.data.frame(loc.newcoords@coords)
names(loc.newcoords) <- c("x", "y")
names(loc.newcoords) <- c("y", "x")

loc.newcoords <- project(cbind(loc.newcoords[, 1], loc.newcoords[, 2]), proj = "+proj=aea +lat_1=29

qplot(x, y, data = loc.newcoords, geom = "polygon")

# gets the values of the pixels
```

```
cdl.ia13 <- raster("data/cdl/CDL_2013_19.tif")
cdl.pts <- cdl.ia03[cellFromXY(cdl.ia03, loc.newcoords)]
table(cdl.pts)  ##gives NAs

apply(poly.coords, 2, min)
apply(poly.coords, 2, max)
```

The region includes too much. The polygon coordinates are fine, but I can't get them to match the coordinate system needed for raster. It seems that the polygon data is in the universal transverse mercador (UTM) and we need to convert it to the longitude-latitude, then to the CRS with the appropriate raster characteristics.

AtlasHCB

```
setwd("U:/classes/585X")
library(ggplot2)
library(maps)
library(plyr)
states <- map_data("state")
head(states)


library(maptools)
xx <- readShapeSpatial("data/IA_AtlasHCB/IA_Historical_Counties/IA_Historical_Counties.shp")
xxx <- thinnedSpatialPoly(as(xx, "SpatialPolygons"), tolerance = 0.1, minarea = 0.001, topologyPres
class(xxx)
slotNames(xxx)

polys <- xxx@polygons

length(polys@Polygons)




library(maptools)
xx <- readShapeSpatial("data/cdl/IAcdl.shp")
xxx <- thinnedSpatialPoly(as(xx, "SpatialPolygons"), tolerance = 0.1, minarea = 0.001, topologyPres
class(xxx)
slotNames(xxx)

polys <- xxx@polygons
length(polys)
# Iowa CDL shapefile, county level Extract the polygons and store the additional
# information , using the lab function

district <- xxx@polygons[[1]]

slotNames(district)
length(district@Polygons)


extractPolygons <- function(xxx) {
```

```
# Keeps track of the order
current.order <- 1

# Keeps track of the polygon group
current.group <- 1

# Keeps track of the county
current.county <- 1

# gets the number of county
n.countys <- length(xxx@polygons)

# creates a place to store the coordinates
out.coordinates <- NULL
# loops through countys to get county poly infor
for (ii in 1:n.countys) {

    # gets information on this county
    this.county <- xxx@polygons[[ii]]

    # gets the number of polygons for this county
    n.polys <- length(this.county@Polygons)

    # loops through all of the polygons in this county
    for (jj in 1:n.polys) {

        # gets the polygon that we're working on
        this.poly <- this.county@Polygons[[jj]]

        # gets the coords for this polygon
        poly.coords <- as.data.frame(this.poly@coords)

        # adds order to the polygon coords
        poly.coords$order <- c(current.order:(current.order + nrow(poly.coords) - 1))

        # updates the current.order
        current.order <- current.order + nrow(poly.coords)

        # adds group
        poly.coords$group <- current.group

        # updates the current group
        current.group <- current.group + 1

        # adds county
        poly.coords$county <- current.county

        # appends out.coordinates
        out.coordinates <- rbind(out.coordinates, poly.coords)
    }

    # updates the current county
    current.county <- current.county + 1
```

```
    }

    # standardizes the first two names
    names(out.coordinates)[1:2] <- c("x", "y")

    # Returns the coordiantes data frame
    out.coordinates

}

oz <- extractPolygons(xxx)
qplot(x, y, order = order, group = group, data = oz, geom = "polygon")

ia <- extractPolygons(readShapeSpatial("data/cdl/IAcdl.shp"))
qplot(x, y, order = order, group = group, data = ia, geom = "polygon")

dat <- xx@data


poly.coords2 <- oz[, 1:2]
loc.newcoords <- project(as.matrix(poly.coords2), proj = "+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=

cdl.pts <- cdl.ia[cellFromXY(cdl.ia, loc.newcoords)]


table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100
```

PLSS
Basically similar code, these files take also too much space. I am going to get rid of them and use the census data, pulling from the web will help me save some space on the cyfiles for my other projects.

2.3 Census data

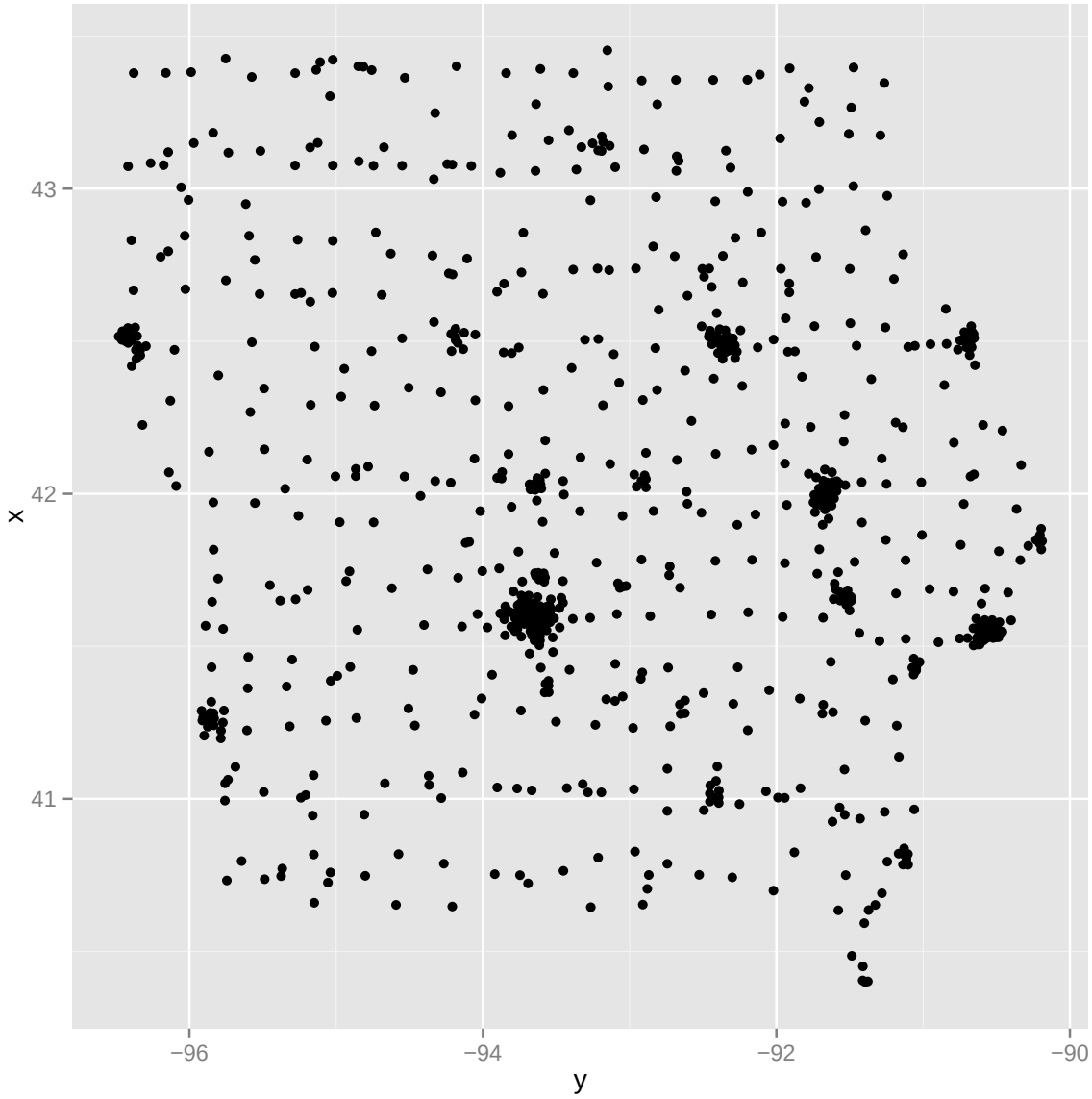
We are going to pull the Iowa data and the Des Moines River data from the web using the XML library in R, as we learnt in class. We will have to pull the coordinates for all the points and do some data processing to get the information in desired format, such as a dataframe containing the numeric values for the points coordinates. The data for the Des Moines River will have to be pulled from the full dataframe on the hydrologic levels in Iowa. Again, we would like to have a final product in the form of a dataframe, with numeric values for the point coordinates.

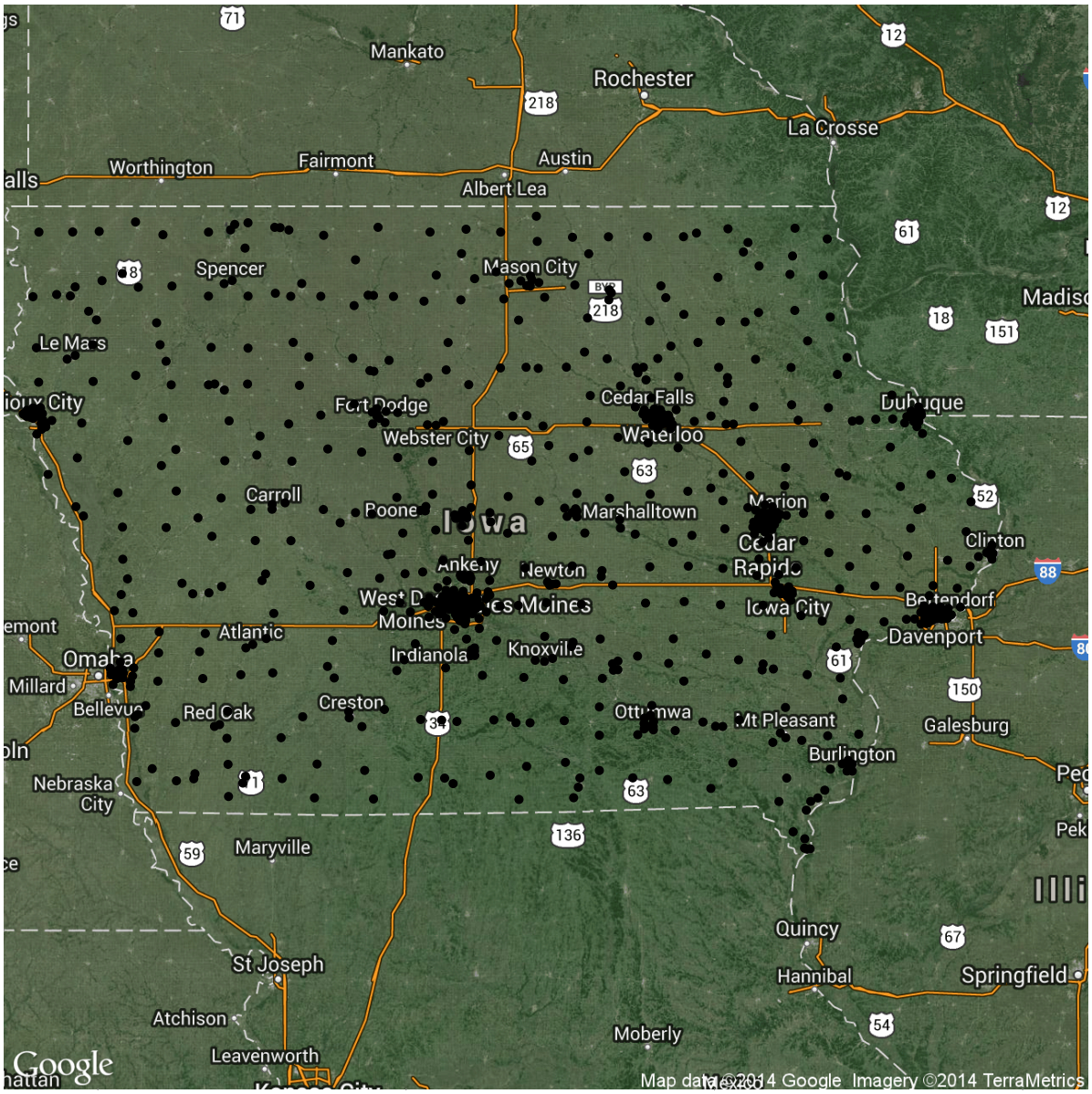
The plyr package in R with come handy at creating the region of interest around the Des Moiner River. This is necessary because the census data on the river contains only a few points, but CEAP was run at more points in the entire watershed. We will have to replicate a region to give us the necessary information, representative for the Des Moines River watershed.

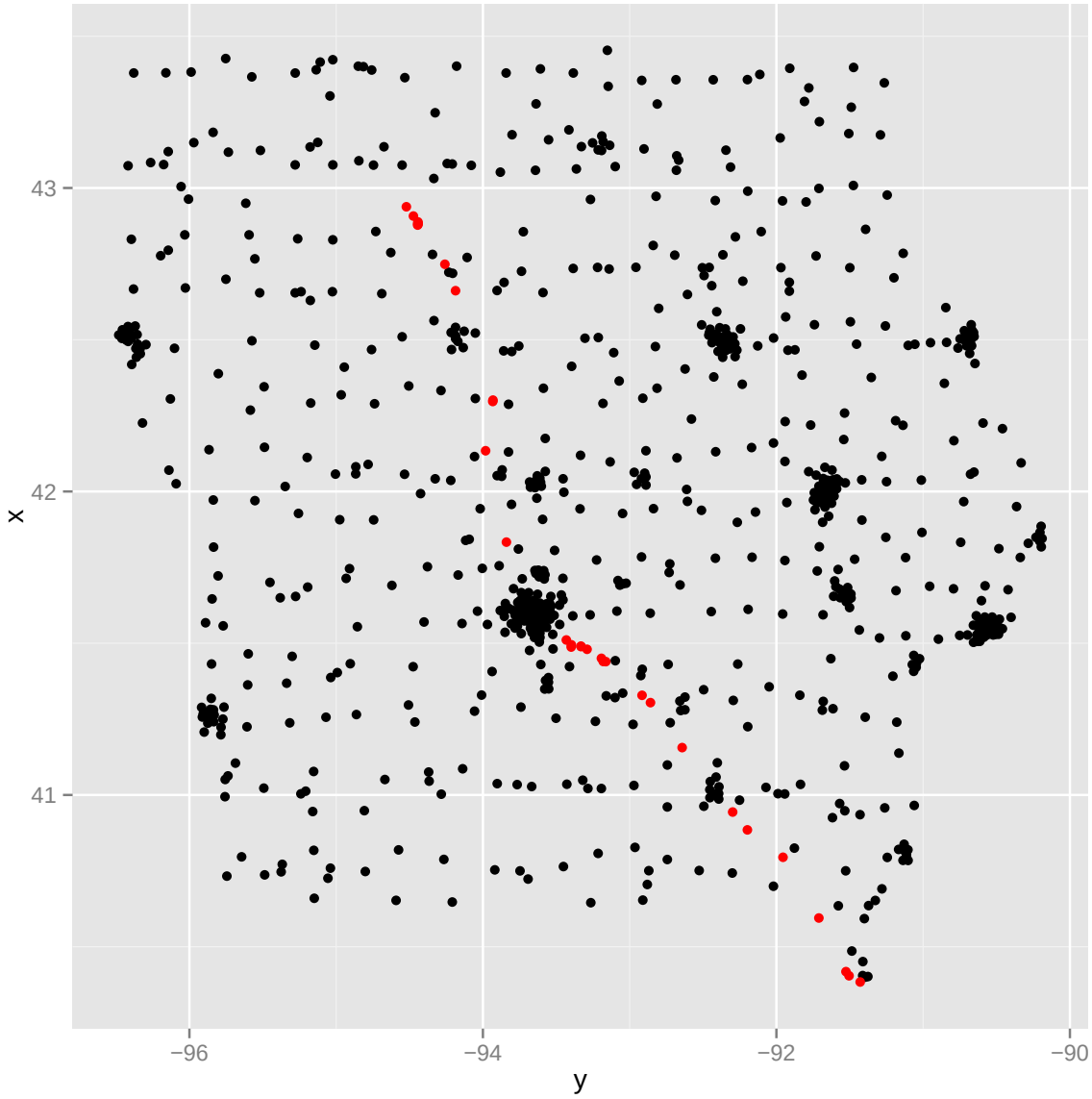
```
setwd("U:/classes/585X")
library(XML)

### pull the iowa census data from the web
src <- "http://tigerweb.geo.census.gov/tigerwebmain/Files/tigerweb_acs13_tract_ia.html"
tables <- readHTMLTable(src)
```

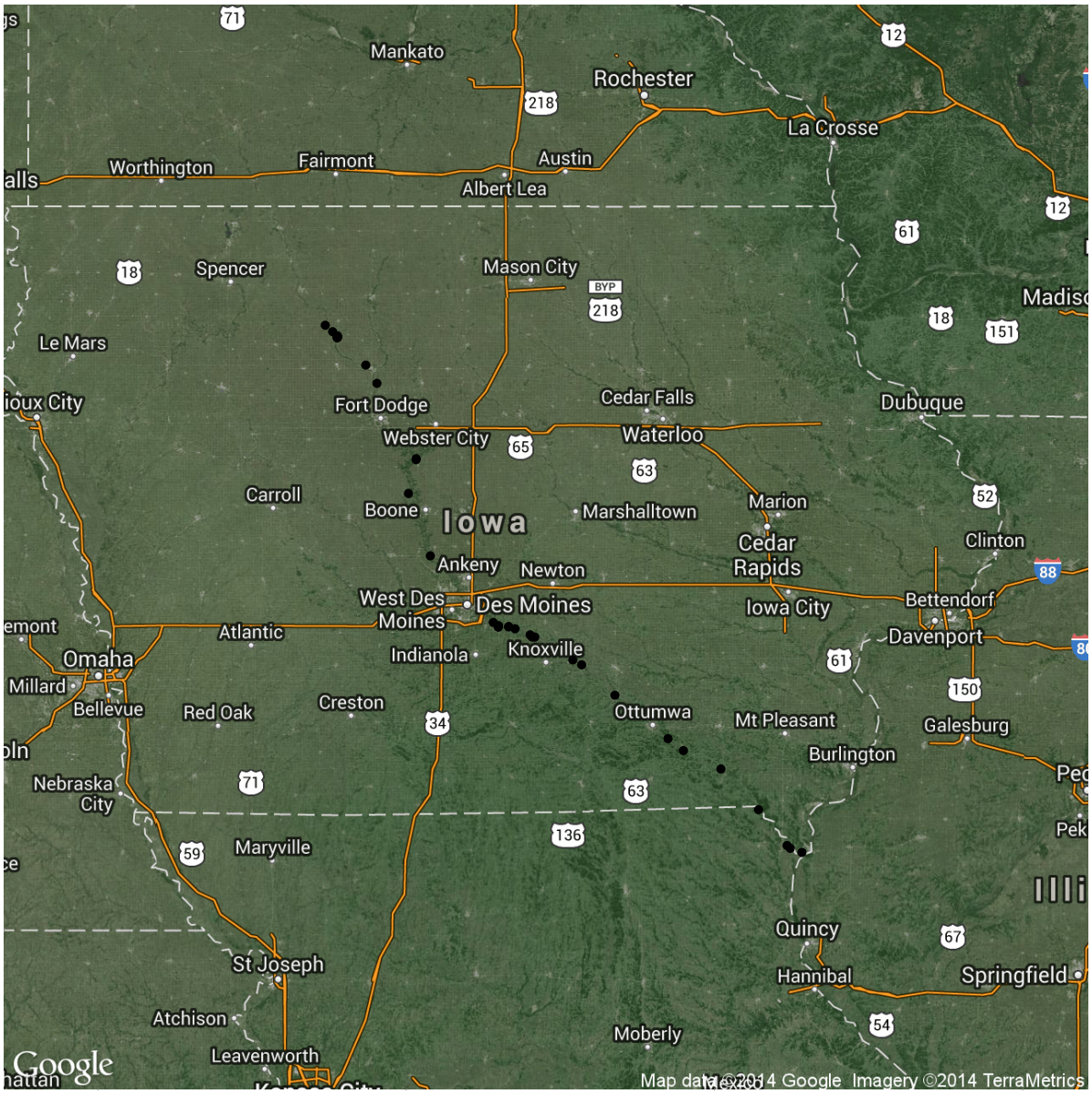
```
tigeria <- tables[[1]]
dim(tigeria)

## [1] 825  17

### transform the data to the desired format

## first we need to pull the coordinates
dsmriv <- tigeria
poly.coords.ia <- as.data.frame(dsmriv[, c("INTPTLAT", "INTPTLON")])
row.names(poly.coords.ia) <- NULL
names(poly.coords.ia) <- c("x", "y")
dim(poly.coords.ia)

## [1] 825   2

poly.coords.ia[, 1] <- as.numeric(as.vector(poly.coords.ia[, 1]))
poly.coords.ia[, 2] <- as.numeric(as.vector(poly.coords.ia[, 2]))

## plot the area
library(ggplot2)

## Warning:  package 'ggplot2' was built under R version 3.0.3

qplot(y, x, data = poly.coords.ia)


library(maps)
library(ggmap)

qmap(location = "iowa", zoom = 7, maptype = "hybrid") + geom_point(data = poly.coords.ia, mapping =
    y = x), size = 2)

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=iowa&zoom=7&size=%20640x640
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=iowa&sensor=fals
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms

### always remember, longitude first, then latitude!!!!!!


### pull the hydrologic iowa data from the web

src <- "http://tigerweb.geo.census.gov/tigerwebmain/Files/tigerweb_tab10_hydro_poly_ia.html"
tables <- readHTMLTable(src)
tigeria <- tables[[1]]
dim(tigeria)

## [1] 16448    15

### transform the data to the desired format and pull the information of interest
mode(tigeria$NAME)
```

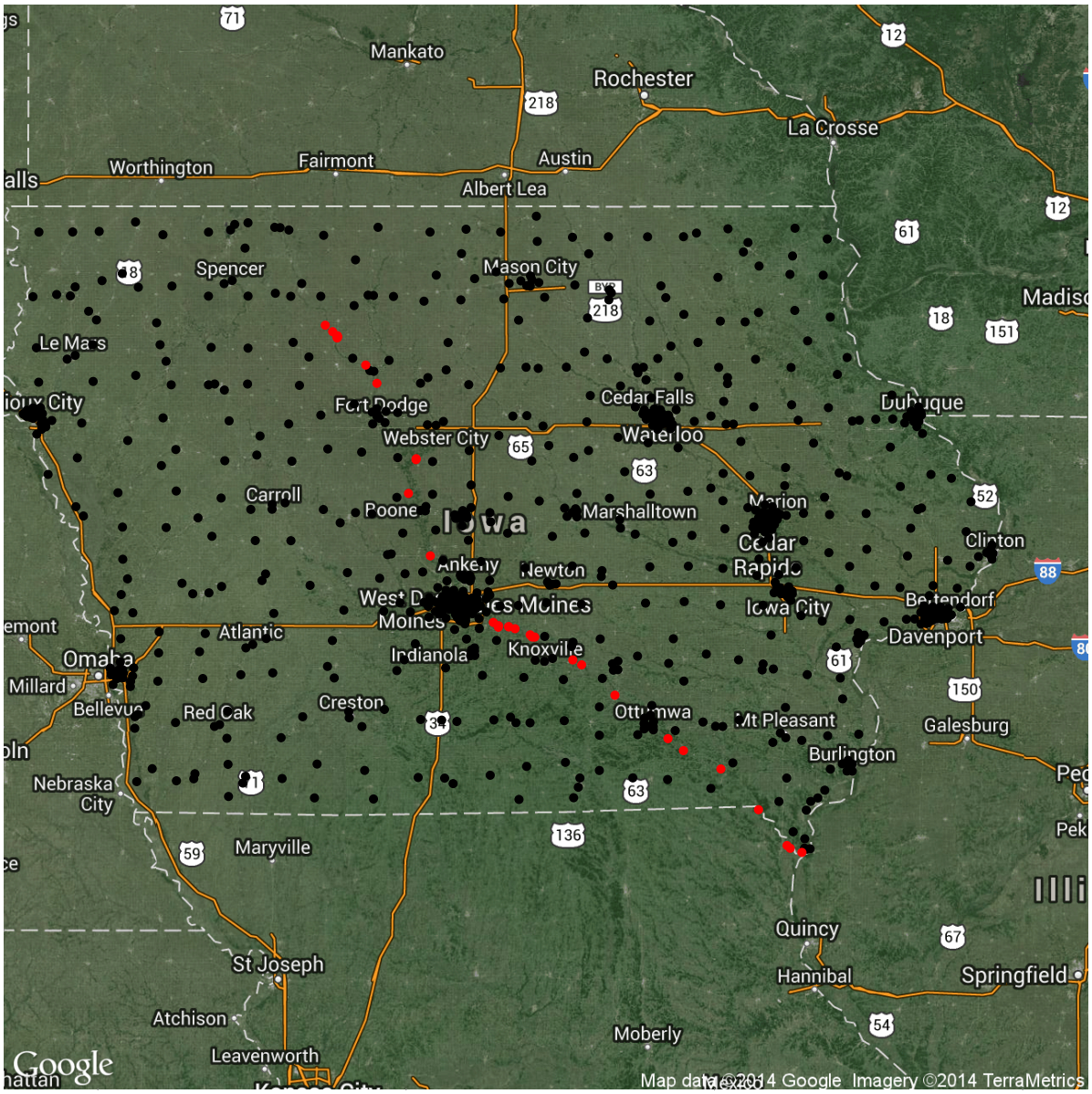
```
## [1] "numeric"

tigeria$NAME <- as.character(tigeria$NAME)

## pull the des moines river data
dsmriv <- tigeria[tigeria$NAME == "Des Moines Riv", ]
dim(dsmriv)

## [1] 32 15

poly.coords <- as.data.frame(dsmriv[, c("INTPTLAT", "INTPTLON")])
row.names(poly.coords) <- NULL
names(poly.coords) <- c("x", "y")
dim(poly.coords)

## [1] 32  2

poly.coords[, 1] <- as.numeric(as.vector(poly.coords[, 1]))
poly.coords[, 2] <- as.numeric(as.vector(poly.coords[, 2]))

## overlay the des moines river area over the iowa plot

qplot(y, x, data = poly.coords.ia) + geom_point(data = poly.coords, aes(x = y, y = x), color = "red


qmap(location = "iowa", zoom = 7, maptype = "hybrid") + geom_point(data = poly.coords, mapping = ae
    y = x), size = 2)

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=iowa&zoom=7&size=%20640x640
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=iowa&sensor=fals
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms

qmap(location = "iowa", zoom = 7, maptype = "hybrid") + geom_point(data = poly.coords.ia, mapping =
    y = x), size = 2) + geom_point(data = poly.coords, mapping = aes(x = y, y = x), size = 2,
    color = "red")

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=iowa&zoom=7&size=%20640x640
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=iowa&sensor=fals
## Google Maps API Terms of Service :  http://developers.google.com/maps/terms

## create a region of interest around the des moines river
library("plyr")

## Warning:  package 'plyr' was built under R version 3.0.3

add.poly.coords <- as.data.frame(t(ldply(llply(poly.coords, function(x) ldply(x, function(y) jitter
    7), 0.075))), unlist)[, -1]))
names(add.poly.coords) <- c("x", "y")

qplot(y, x, data = poly.coords.ia) + geom_point(data = poly.coords, aes(x = y, y = x), color = "red
    geom_point(data = add.poly.coords, aes(x = y, y = x), color = "green")

# gets the values of the pixels
loc.newcoords <- project(cbind(add.poly.coords[, 2], add.poly.coords[, 1]), proj = "+proj=aea +lat_
```

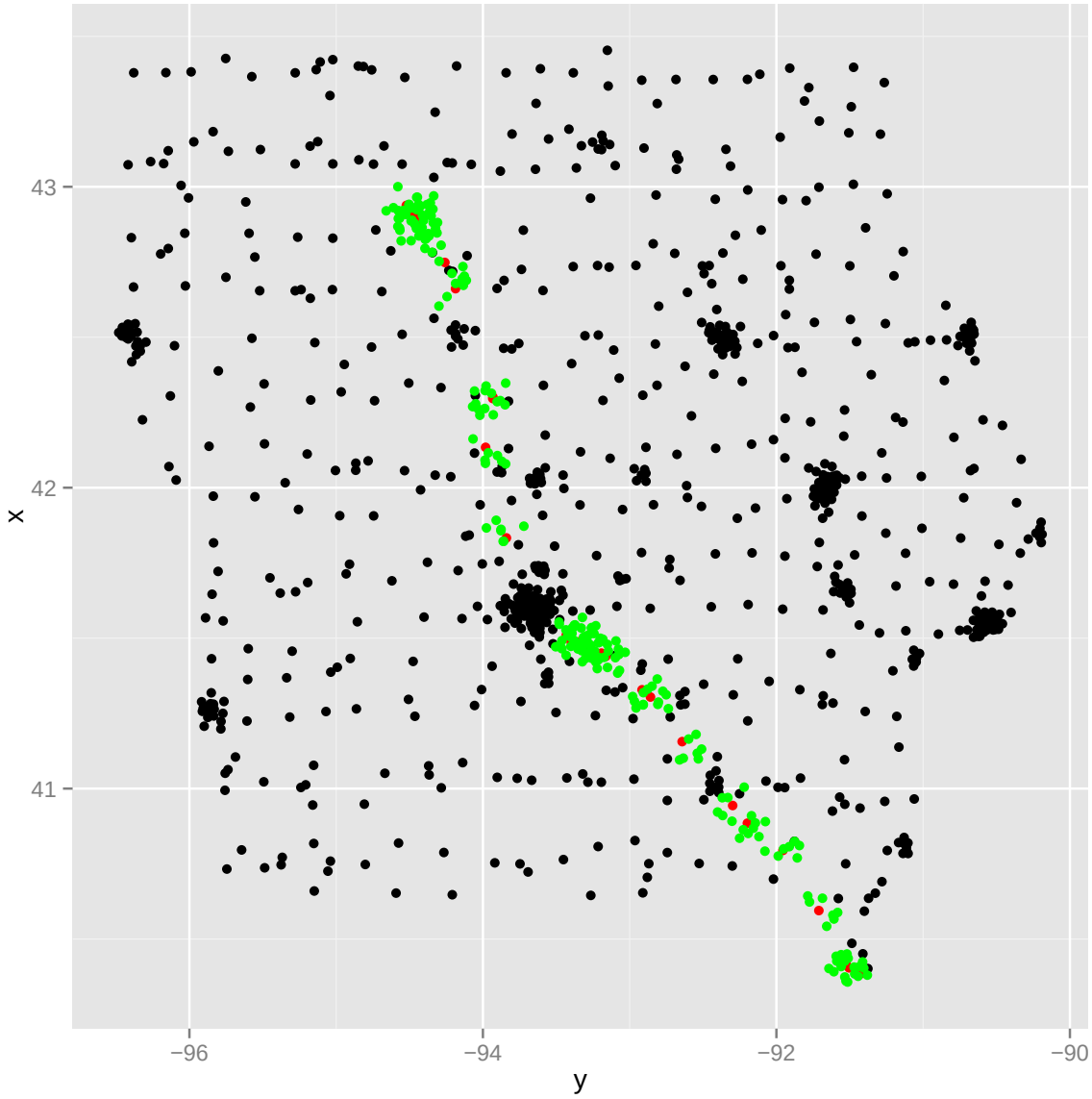

43
42
41
x
−96
−94
−92
−90
y


Mankato
Rochester
La Crosse
Worthington
Fairmont
Austin
Albert Lea
Spencer
Mason City
Le Mars
Cedar Falls
Dubuque
Fort Dodge
Webster City
Waterloo
Carroll
Marshalltown
Iowa
Cedar Rapids
Clinton
Newton
Iowa City
Bettendorf
Davenport
Atlantic
Indianola
Knoxville
Omaha
Millard
Bellevue
Red Oak
Creston
Ottumwa
Mt Pleasant
Galesburg
Burlington
Nebraska City
Maryville
Quincy
St Joseph
Hannibal
Springfield
Atchison
Moberly
Leavenworth
Google
Map data ©2014 Google Imagery ©2014 TerraMetrics


43
42
41
x
−96
−94
−92
−90
y


Mankato
Rochester
La Crosse
Worthington
Fairmont
Austin
Albert Lea
Spencer
Mason City
Le Mars
Fort Dodge
Cedar Falls
Dubuque
Webster City
Waterloo
Carroll
Boone
Iowa
Marshalltown
Marion
Cedar Rapids
Clinton
Ankeny
Newton
West Des Moines
Des Moines
Iowa City
Bettendorf
Atlantic
Davenport
Omaha
Indianola
Knoxville
Millard
Bellevue
Red Oak
Creston
Ottumwa
Mt Pleasant
Galesburg
Burlington
Nebraska City
Maryville
Quincy
St Joseph
Hannibal
Springfield
Atchison
Moberly
Leavenworth
Google
Map data ©2014 Google Imagery ©2014 TerraMetrics


Mankato
Rochester
La Crosse
Worthington
Fairmont
Austin
Albert Lea
Spencer
Mason City
Le Mars
Fort Dodge
Cedar Falls
Dubuque
Webster City
Waterloo
Carroll
Poone
Iowa
Marshalltown
Marion
Cedar Rapids
Clinton
Ankeny
Newton
West Des Moines
Iowa City
Bettendorf
Atlantic
Davenport
Omaha
Indianola
Knoxville
Millard
Bellevue
Red Oak
Creston
Ottumwa
Mt Pleasant
Galesburg
Burlington
Nebraska City
Maryville
Quincy
St Joseph
Hannibal
Springfield
Atchison
Moberly
Leavenworth
Google
Map data ©2014 Google Imagery ©2014 TerraMetrics



3 Results

We will have to extract the pixel count information available in the CDL data for the points with coordinates in the Census data. By doing this matching, we will construct summaries for all the categories (corn, soybean, etc) for the points in the region of interest. We will do this for the years 2003-2007 and we will report the tables for each year. The final comments would be on the changes in crop categories over this time period and the effect on the future analysis done with the CEAP data.

```
# ?cellFromXY cellFromXY

cdl.pts <- cdl.ia03[cellFromXY(cdl.ia03, loc.newcoords)]

## Warning:  some indices are invalid (NA returned)
```

```
table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100

## cdl.pts
##       1       5      25      28      36      42      44      61      63      82      83
## 26.6667 30.0000  0.9524  0.4762  4.2857  0.4762  0.4762  5.7143  9.0476  1.4286  3.3333
##     176
## 17.1429

cdl.pts <- cdl.ia04[cellFromXY(cdl.ia04, loc.newcoords)]

## Warning:  some indices are invalid (NA returned)

table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100

## cdl.pts
##       1       5      36      61      63      70      81      82      83      88     176
## 31.9048 21.9048  1.4286  4.2857 10.0000  0.9524  0.9524  3.3333  3.8095  1.4286 20.0000

cdl.pts <- cdl.ia05[cellFromXY(cdl.ia05, loc.newcoords)]

## Warning:  some indices are invalid (NA returned)

table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100

## cdl.pts
##       1       5      36      61      63      81      82      83      88     176
## 24.2857 26.6667  0.9524  2.3810 18.5714  0.4762  3.8095  3.3333  3.8095 15.7143

cdl.pts <- cdl.ia06[cellFromXY(cdl.ia06, loc.newcoords)]

## Warning:  some indices are invalid (NA returned)

table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100

## cdl.pts
##       1       5      28      36      61     111     121     124     141     142     143
## 30.4762 20.9524  0.9524  0.4762  1.4286  2.3810  3.8095  0.4762 13.8095  0.4762  0.4762
##     152     176     190     195
##  1.4286 19.0476  1.4286  2.3810

cdl.pts <- cdl.ia07[cellFromXY(cdl.ia07, loc.newcoords)]

## Warning:  some indices are invalid (NA returned)

table(cdl.pts)/length(cdl.pts[-which(is.na(cdl.pts))]) * 100

## cdl.pts
##       1       5      28      36      87     111     121     124     141     176     190
## 26.1905 17.6190  0.4762  0.4762  0.4762  2.8571  6.1905  0.4762 16.6667 24.7619  1.9048
##     195
##  1.9048
```