R による GIS データの利用について

鑓　田　　亨

概　要

近年、エリアマーケティングの分野では GIS(地理情報システム、Geographic Information System) が用いられている。本稿では、オープンソースソフトとして広く利用可能な R を使い、GIS データを扱う方法についてまとめる。

流通業における GIS データの R での扱いについては、山内 (2004) が論じている。しかし山内で使用されているパッケージは CRAN に収録されていない Rmap パッケージであり、データの操作も Excel によるものに偏っている。そこで本稿では maptools パッケージを用い、山内のデータの再計算を行なう。

1 はじめに

近年、エリアマーケティングの分野では GIS（地理情報システム、Geographic Information System） が用いられている。

企業の利用している GIS データを分析したいとする。データを入手できたとしても、商用の GIS アプリケーションは高価であり、個人での購入は難しい。その点 R の場合には、地図の作成機能などは商用ソフトに劣るものの、オープンソースソフトとして自由に利用できる。また、通常の GIS アプリケーションでは、地図情報に関係づけられたデータに対して適用できる統計分析の手法が限られている。これに対して R なら、データに対して空間統計学を含むさまざまな統計分析の手法を適用できる。

R の maptools パッケージでは、米国の ESRI 社の GIS 標準データフォーマット形式である Shape ファイルを扱うことができる。MapInfo 社の TAB や MIF/MID ファイルなど他のフォーマットについても、GDAL（Geospatial Data Abstraction Library）[1] に含まれている ogr2ogr によって Shape ファイルに変換することができる。

流通業における GIS データの R での扱いについては、山内（2004） が論じている。しかし山内で使用されているパッケージは CRAN に収録されていない Rmap パッケージであり、データの操作も Excel によるものに偏っている。そこで本稿では maptools パッケージを用い、山内のデータの再計算を行なう。

なお、筆者の使用するコンピュータの OS は Debian GNU/Linux 5.0.3 （lenny） である。R は version 2.9.2 （2009-08-24） を使用した。

2　利用するデータについて

地域販売データおよび郵便番号・JIS コード[2] 変換テーブルについては、山内の用いた岡田

[1] http://www.gdal.org/

[2] 正確には「全国地方公共団体コード」という名称である。日本工業規格では、規格番号「JISX0402」、規格名称「市区町村コード」となっている。

表 1　japan ver62.dbf の属性データ項目

フィールド名	内容	備考
PREF	都道府県名	
CITY1	市町村名	政令指定都市の場合は区名
CITY2	政令指定都市の市名	
TOWN1	郡名（町村部のみ）	
TOWN2	県市区町村名	
JCODE	市町村コード	5 桁の JIS コード
P-NUM	人口	
H-NUM	世帯数	
FLAG1	陸／水フラグ	陸部：1、湖沼：2
FLAG2	更新時のデータ変更フラグ	変更無：0、ver3.0：1、ver5.0：3

（2004）付録の CD-ROM のファイルを利用した。ただし Excel ファイルの内容を、文字コードを UTF-8 とした CSV ファイルに変換し、ファイル名をそれぞれ hanbai2.csv, jis.csv とした。

地図データについては、ESRI Japan が提供している全国市区町村界データ（シェープファイル）を利用した[3]。

以下、データファイルはすべてカレントディレクトリにあるものとする。

3　シェープファイルの読み込み

まず maptools パッケージをロードする。その後、read.shape( )で、japan_ver62. shp を Map オブジェクトとして読み込み、jpn に付値する[4]。

```
> library(maptools)
> jpn <- read.shape("./japan_ver62.shp")
```

jpn は Shape と、DBF ファイルのデータを格納したデータフレームである att. data からなるリストとなる。plot(jpn)と入力することで日本地図が描画される[5]。地図の特定の部分を抽出するためには、Map オブジェクトのポリゴンデータを、Map2poly( )によって polylist オブジェクトに変換しなければならない。ついでに後の作業のために、get. Pcent( )で市町村の中心点を取り出しておこう。

```
> jpn.poly <- Map2poly(jpn)
> jpn.xy <- get.Pcent(jpn)
```

japan_ver62.dbf の文字コードがシフト JIS のため、jpn$att.data の文字コードもシフト JIS となっている。そこで jpn$att.data の文字コードを UTF-8 に変換する。そのためには次のような関数 sjis2utf( )を定義しておくと良いだろう。以下では変換したデータを jpn.df に付値した。

[3] http://www.esrij.com/products/gis data/japanshp/files/japan ver62.zip
[4] 以下、ワーニングメッセージなどは省略している。
[5] ただし plot(jpn$Shape) としてもエラーとなる。

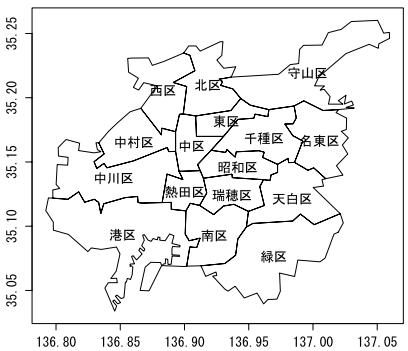
```
> sjis2utf <- function(df){
+     df2 <- df
+     for (i in 1:ncol(df)){
+         if  (is.factor(df[, i])){
+             df2 [, i] <- as.factor(iconv(df[, i], "shift-jis", "utf-8"))
+             df2 [, i] [df2 [, i] == "NA"] <- NA ## NA も因子になってしまうので
+         }
+     }
+      df2
+ }
>   jpn.df <- sjis2utf(jpn$att.data)
>  names(jpn.df) ## jpn.df のもつ変数名を表示
 [1] "OBJECTID"   "PREF"       "CITY1"      "CITY2"      "TOWN1"
 [6] "TOWN2"      "JCODE"      "P_NUM"      "H_NUM"      "FLAG1"
[11] "FLAG2"      "Shape_Leng" "Shape_Area"
```

jpn.df の *n* 番目の行のデータは、polylist オブジェクト jpn.poly の *n* 番目のデータに対応している。したがってデータフレーム jpn.df に新たな列を加えたり、列の順序を変更することはできるが、行の順序を変えてはならない。

関数 subset( ) を用いて特定の条件に合致する polylist オブジェクトを抽出することができる[6]。例えば、名古屋市の地図を描画するためには、以下のように入力する（結果は図 1）。

```
> nagoya.poly <- subset(jpn.poly, jpn.df$CITY2==" 名古屋市 ")
> nagoya.df <- subset(jpn.df, jpn.df$CITY2==" 名古屋市 ")
> nagoya.xy <- subset(jpn.xy, jpn.df$CITY2==" 名古屋市 ")
> plot(nagoya.poly)
> text(nagoya.xy, as.character(nagoya.df$CITY1))## 区名の表示
## 結果を nagoya.eps に保存
> dev.copy2eps(file="./nagoya.eps", family="Japan1GothicBBB")
```

図 1　名古屋市の地図



[6] この場合、内部的に subset.polylist( ) が使用される。

4　地域販売データの加工

地域販売データと郵便番号・JIS コード変換テーブルを読み込み、それぞれ hanbai2, jis と名前を付ける。

```
> hanbai2 <- read.csv("./hanbai2.csv")
> head(hanbai2, 3) ## 始めの 3 行を表示
yubin jyusyo kingaku
1 600062   札幌市中央区       27139
2 30802    札幌市中央区        1406
3 640953   札幌市中央区        6881
> jis <- read.csv("./jis.csv")
> jis <- unique(jis)## 重複行の除去
> head(jis, 3)## 始めの 3 行を表示
  JIS1 yubin1
1 1101 600000
2 1101 640941
3 1101 600041
```

hanbai2 は個々の取引についての、取引先の郵便番号、住所、取引金額からなるデータである。これを地図データと結びつけたいのだが、jpn.df が持つ地域コードは、JIS コードである。そこで、各郵便番号に対応する JIS 地名コードを割り当て、同じ JIS 地名コードを持つ取引ごとに取引金額の合計を計算する。

jis は、郵便番号とそれに対応する JIS 地名コードからなるデータである。日本郵便のデータ[7] から作成したものと思われるが、日本郵便のデータは、全角となっている町域部分の文字数が 38 文字を越える場合、また半角となっているフリガナ部分の文字数が 76 文字を越える場合は、複数レコードに分割している。そのためか、jis にも重複行がある。そのままにしておくと以下の処理で問題となるので、unique( ) によって重複行を取り除いておく。

共通する変数の列をもとに、2 つのデータフレームを結合するには merge( ) を用いる。ここでは、hanbai2 の yubin と jis の yubin1 が共通する変数となる。hanbai2 の行はすべて用いられなければならないが、hanbai2 に含まれない郵便番号が追加されても意味がないので、引数 all.x=TRUE, all.y=FALSE を指定する。結果を mapdata に付値した。

```
> id <- 1:nrow(hanbai2)
> hanbai2 <- cbind(id, hanbai2) ## hanbai2 のレコードに id 番号をつける
> mapdata <- merge(hanbai2, jis, by.x="yubin", by.y="yubin1",
+                          all.x=TRUE, all.y=FALSE)
> mapdata <- mapdata [order(mapdata$id), ] ## id をもとにソート
```

sum(is.na(mapdata$JIS1)) と入力すると、計算結果は 31 である。これは hanbai2 の変数 yubin のなかに、jis の yubin1 に含まれないものが 31 件あることを意味する。おそらく地域販売データの郵便番号に間違いがあるのだろう。日本郵便の郵便番号データにも存在しない郵便

[7] http://www.post.japanpost.jp/zipcode/download.html

番号がある。仕方がないので、NA となっているデータを確認し、以下のように手動で適当な JIS コードと思われるものを付値した[8]。

```
> mapdata [mapdata$yubin==608646, ] $JIS1 <- 1103
> mapdata [mapdata$jyusyo==" 札幌市北区 ", ] $JIS1 <- 1102
> mapdata [mapdata$jyusyo==" 札幌市厚別区 ", ] $JIS1 <- 1108
> mapdata [mapdata$jyusyo==" 旭川市 ", ] $JIS1 <- 1204
> mapdata [mapdata$jyusyo==" 空知郡中富良野町 ", ] $JIS1 <- 1461
```

JIS コードごとに、取引金額の合計を計算するには tapply( ) を用いる。結果をデータフレームとして、mapdata1 に付値する。

```
> jis.kingaku <- tapply(mapdata$kingaku, mapdata$JIS1, sum)
> mapdata1 <- data.frame(jiscode=names(jis.kingaku),
+                        kingaku=as.numeric(jis.kingaku),
+                        row.names=NULL)
```

さらに、この mapdata1 を、DBF ファイルのデータを格納している jpn.df と結合させる。結果を jpn2.df に付値した。jpn2.df$OBJECTID で、行の順序が変化していないか確認しておいた方が良いだろう。

```
> jpn2.df <- merge(jpn.df, mapdata1, by.x="JCODE", by.y="jiscode",
+                  all.x=TRUE)
```

5 地図の出力

以上のように計算した、地域ごとの金額データの大きさによって色を変えて札幌市の地図を描画する。まず札幌市のデータを抽出しよう。

```
> sapporo.poly <- subset(jpn.poly, jpn2.df$CITY2==" 札幌市 ")
> sapporo.xy <- subset(jpn.xy, jpn2.df$CITY2==" 札幌市 ")
> sapporo.df <- subset(jpn2.df, jpn2.df$CITY2==" 札幌市 ")
```

5.1 色の指定

plot( )は、polylist オブジェクトを順に描画していく。plot( )のオプション引数 col に、polylist オブジェクトと同じ長さの、色を表わすベクトルを渡せば、polylist オブジェクトをその色で描画する。

色を表わすベクトルをつくるための基本的な考え方を説明しよう。まず 5 つの標本が属する

[8]郵便番号 608646 は jis のなかにはない。hanbai2 で郵便番号 608646 に対応する jyusyo は「札幌市中央区」となっているが、山内 (2004) の計算結果をおさめたファイルを確認すると「札幌市東区」の JIS コードである 1103 が振られていた。ここでは山内にしたがうことにする。

階級（1 ～ 3）からなるベクトル cl があったとする。cp を白、グレー、黒からなるカラーパレットとする。階級 1 には白、2 にはグレー、3 には黒が対応するとしよう。cp[cl] によって、階級番号に対応したパレットの色からなるベクトルが生成される。

```
> cl <- c(2, 3, 1, 3, 2, 1) ## 5 つの標本の属する階級（1 ～ 3）
> cp <- c("white", "gray", "black") ## カラーパレット
> cp[cl]
[1] "gray"  "black" "white" "black" "gray"  "white"
```

さまざまなカラーパレットが用意されているが、ここでは RColorBrewer パッケージを利用して、5 色からなるカラーパレットを作成する。

```
> library(RColorBrewer) ## パッケージのロード
> cp <- brewer.pal(5, "Greys")
> barplot(rep(1, 5), col=cp)## 色の確認
```

5.2 階級区分

取引金額のデータが 5 つの階級のどこに属するかを示すベクトルをつくるには cut( ) を用いる。どのような分けるかは、引数 breaks で区間を指定しなければならない。区間の決め方については、いろいろな考え方があるだろう。

```
> br1 <- c(0, 500000, 750000, 1000000, 2000000, 3000000)## 手動
> br2 <- 5  ## 等間隔
> br3 <- quantile(sapporo.df$kingaku,
+                 probs=seq(0, 1, 0.2))## 等サイズ
> br3
      0%      20%      40%      60%      80%     100%
446476.0 593098.8 745175.4 1206107.4 1324731.2 2366430.0
```

今回は、各階級に属するサンプル数が同じになるように区間を決める。

```
> cl <- cut(sapporo.df$kingaku, breaks=br3, include.lowest=TRUE)
```

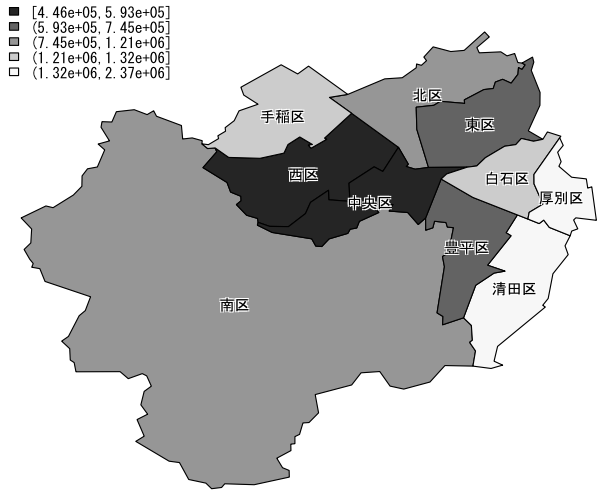
5.3 地図の描画

では、実際に地図を描画してみる。text( ) で札幌市内の区名を表示するが、テキストが黒いと売上の多い地域では文字を識別できない。そこで少しずらした位置に白字でテキストを表示し、その後に黒字でテキストを表示する関数 text.cw を定義しておく。最後に legend( ) で凡例を表示する。結果は図 2 のようになる [9]。

[9] 山内 , 2004, p.208) の図 9.43 と色分けが異なるが、山内の途中の計算結果を見ても、北区の金額が突出し、中央区や西区の金額が少ないということはない。

```
> plot(sapporo.poly, col=cp[cl], axes=FALSE)
> ## 区名の表示
> # 黒字を目立たせるために、部分的に背景を白く
> text.cw <- function(x, y=NULL, labels, ...) {
+     text(t(t(x)+c(0.001, 0.001)), labels, col="white", ...)
+     text(t(t(x)+c(0.001, -0.001)), labels, col="white", ...)
+     text(t(t(x)+c(-0.001, 0.001)), labels, col="white", ...)
+     text(t(t(x)+c(-0.001, -0.001)), labels, col="white", ...)
+     text(x, labels, col="black", ...)
+ }
> text.cw(sapporo.xy, labels=as.character(sapporo.df$CITY1), cex=0.7)
> ## 凡例の表示
> legend(x="topleft", legend=rev(levels(cl)), fill=rev(cp), cex=0.7,
+             bty="n")
> dev.copy2eps(file="./sapporo.eps", family="Japan1GothicBBB")
```

図 2　札幌市内の取引金額



参考文献

Lewin-Koh, Nicholas J., Roger Bivand, contributions by Edzer J. Pebesma, Eric Archer, Adrian Baddeley, Hans-Jorg Bibiko, Stéphane Dray, David Forrest, Patrick Giraudoux, Duncan Golicher, Virgilio Gómez Rubio, Patrick Hausmann, Thomas Jagger, Sebastian P. Luque, Don MacQueen, Andrew Niccolai, and Tom Short（2009）*maptools: Tools for reading and handling spatial objects*. R package version 0.7-26.

Neuwirth, Erich（2007）*RColorBrewer: ColorBrewer palettes*. R package version 1.0-2.

R Development Core Team（2009）*R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

岡田昌史（編）（2004）『The R Book ―データ解析環境 R の活用事例集―』，九天社．

尾野久二（2004）「R で GIS」，岡田昌史（編）『The R Book―データ解析環境 R の活用事例集―』，九天社，第 12 章，264-282 頁．

牧山文彦（2007）「maptools による地図の作成」，*ESTRELA*，第 163 号，54-59 頁，10 月．

山内紀久雄（2004）「流通業におけるデータのビジュアル化」，岡田昌史（編）『The R Book ―データ解析環境 R の活用事例集―』，九天社，第 9 章，179-209 頁．