Chapter 3

Result of function

In this study we created 10 functions, including:

The *create.map*() function is the main function to create a map. The user can specify the color and the grid line for the map. Figure 3.1 shows all sub-districts of Mueang Pattani district, Pattani province. The panel on the left shows the result on R Console, as command line and code for each region and the panel on the right shows a white map and without grid lines. Figure 3.2 shows a green map and dark gray grid lines. Figure 3.3 shows a complex region of a Na Thap canal in Na Thap sub-district, Chana district, Songkhla province, which has two regions in one place.

```
> map<-read.table("file.xy",h=T)
> create.map(flexy=map,
+ wh=3,ww=4,xylabel=F,xyline=F)
[1] simple regions are
    plcid
1  940101
2  940102
3  940103
4  940104
5  940105
6  940106
7  940107
8  940108
9  940109
10 940110
11 940111
12 940112
13 940113
>
```

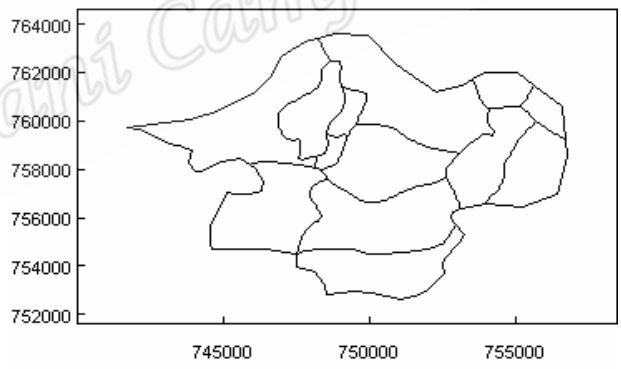

Figure 3.1: The results of *create.map*() function for a simple region

```
> map<-read.table("file.xy",h=T)
> create.map(flexy=map,
+ xscl=c(740000,760000),
+ yscl=c(750000,765000),
+ wh=3,ww=4,
+ header.text="Muang pattani",
+ map.col="lightgreen")
[1] simple regions are
    plcid
1  940101
2  940102
3  940103
4  940104
5  940105
6  940106
7  940107
8  940108
9  940109
10 940110
11 940111
12 940112
13 940113
>
```

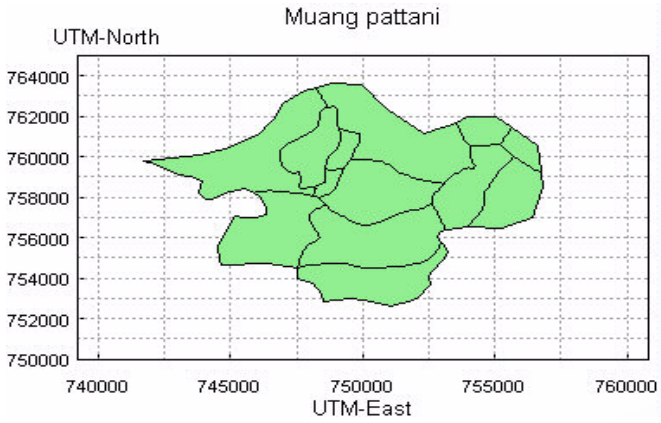

Figure 3.2: The green map display for a simple region

```
> map<-read.table("naThapRiver1.xy",h=T)
> create.map(flexy=map,wh=3,ww=4)
[1] complex regions are
   plcid
1 208184
>
```

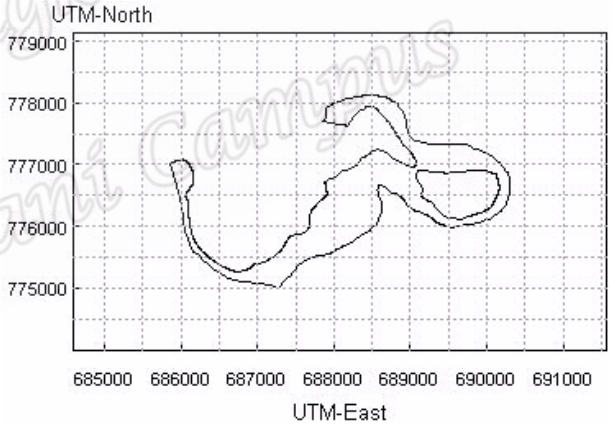

Figure 3.3: The results of *create.map*() function for a complex region

The *setcol.map*() function is a function for specifying color of each region. If the user does not identify color for each region, this function generates the color automatically. But this function cannot be used with a complex region. Figure 3.4 shows different colors in each region and shows the result on R Console.

```
> setcol.map(flexy=map)
[1] simple regions
    plcid         colour
1  940101     aquamarine
2  940102 cornflowerblue
3  940103         bisque
4  940104      burlywood
5  940105          coral
6  940106      cadetblue
7  940107   darkseagreen
8  940108           gold
9  940109          linen
10 940110         violet
11 940111 lightslategray
12 940112      lightpink
13 940113   lightskyblue
>
```

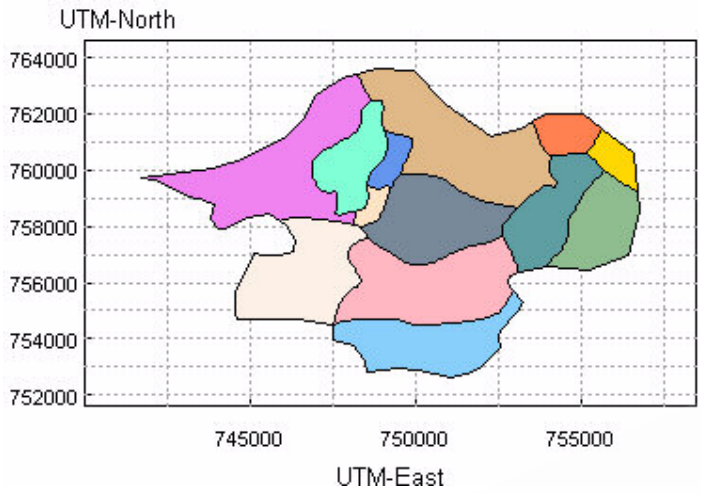

Figure 3.4: The results of *setcol.map*() function

The *setcol.cmap*() function is a function for specifying color for a complex region. The user must identify color of region in order from the largest region to the smallest region. Figure 3.5 shows two regions of a complex region, the first region is white color and the second region is gold color.

```
> setcol.cmap(flexy=map,
+ plcid=208184,reg=c(1,2),
+ mcol=c("white","gold"))
  region colour
1      1  white
2      2   gold
>
```

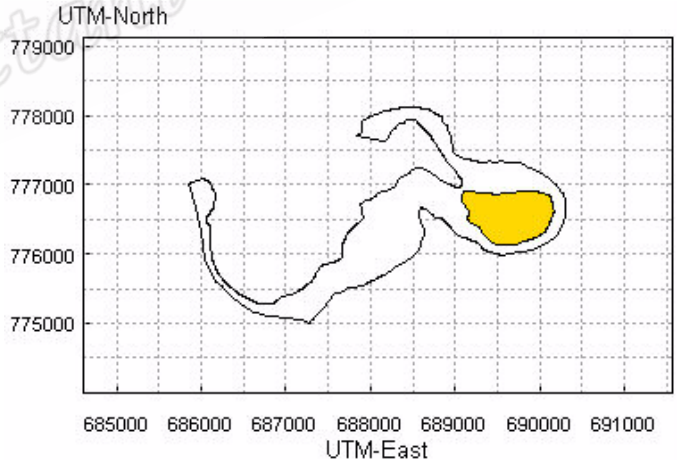

Figure 3.5: The results of *setcol.cmap*() function

The *setnme.map*() function displays a name on each region. If the user does not identify position of *x* and *y*, this function will display the name on the center of each region. For a complex region, the name displays on the center of the largest region.

The function automatically generates name from 1 to *n*, when *n* is number of region, if the name is not identified by the user. This function also can display a frame for the name. For example, figure 3.6 shows the name, which is generated by the function. The left panel shows the name of each region on R Console and the right panel shows the name of each region without a frame. Figure 3.7 shows the name with yellow frame. Figure 3.8 shows the name on the center of the largest region in a complex region.

```
> setnme.map(flexy=map)
[1] there are simple regions
[1] -------------------------
    plcid name
1  940101    1
2  940102    2
3  940103    3
4  940104    4
5  940105    5
6  940106    6
7  940107    7
8  940108    8
9  940109    9
10 940110   10
11 940111   11
12 940112   12
13 940113   13
>
```

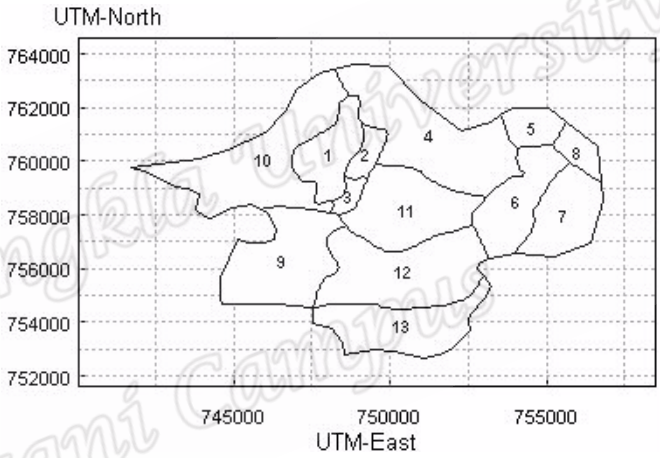

Figure 3.6: Map display of name of each region without a frame

```
> setnme.map(flexy=map,
+ frm="circle",sfrm=3,
+ colfrm="yellow",
+ sfont=0.6)
[1] there are simple regions
[1] -------------------------
    plcid name
1  940101    1
2  940102    2
3  940103    3
4  940104    4
5  940105    5
6  940106    6
7  940107    7
8  940108    8
9  940109    9
10 940110   10
11 940111   11
12 940112   12
13 940113   13
>
```

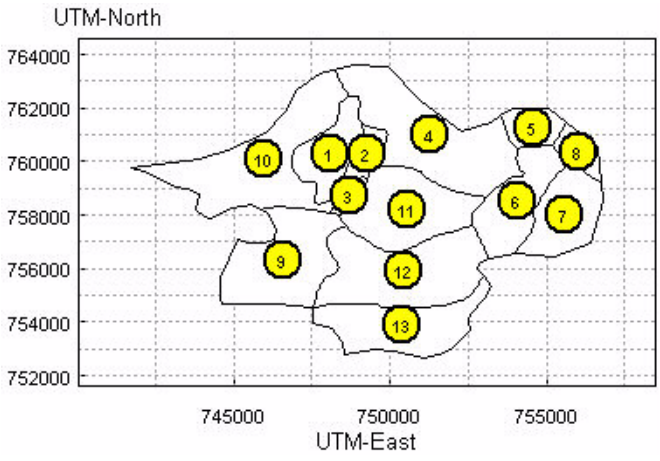

Figure 3.7: A map displays of name of each region with yellow frame

```
> setnme.map(flexy=map,
+ frm="circle",sfrm=3,
+ colfrm="lightpink",
+ sfont=0.6)
[1] there are complex regions
[1] -------------------------
   plcid name
1 208184    1
>
```

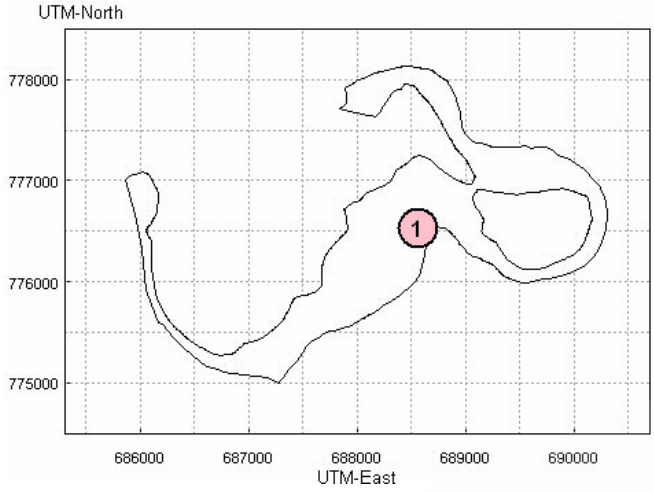

Figure 3.8: Map display name for complex region

The *combine.map*() function allows users to combine different regions into one region. For example, as shown in figure 3.9, figure 3.10 and figure 3.11, the map has 13 regions, which we will call 1 until 13. Suppose the user wants to combine regions 2, 3 and 11 and use a yellow color for the final combined region. Also, the user wishes to combine regions 9, 12 and 13 and use a gold color for the final combined region. Figure 3.9 shows the common boundaries of each of the three regions as dotted lines. Figure 3.10 shows the final combined region with removed lines. Figure 3.11 shows the final region with the solid lines.

```
> combine.map(flexy=map,
+ plcid=c(940109,940112,940113),
+ mline="13",mcol="gold")
>
> combine.map(flexy=map,
+ plcid=c(940111,940104,940102),
+ mline="13",mcol="yellow")
>
```

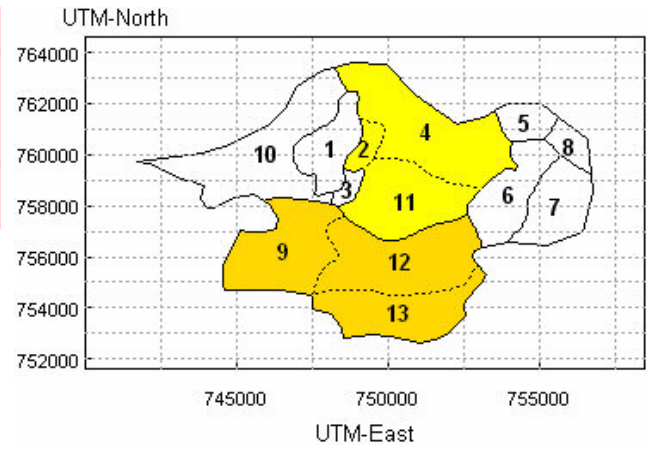

Figure 3.9: Example of combined regions with dotted line

```
> combine.map(flexy=map,
+ plcid=c(940109,940112,940113),
+ mline="blank",mcol="gold")
>
> combine.map(flexy=map,
+ plcid=c(940111,940104,940102),
+ mline="blank",mcol="yellow")
>
```

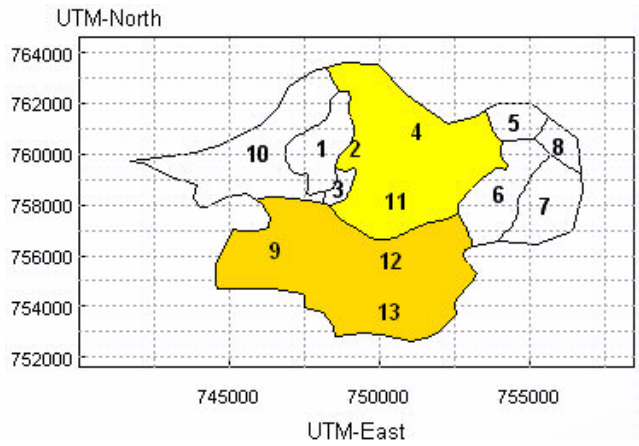

Figure 3.10: Example of combined regions with removed line

```
> combine.map(flexy=map,
+ plcid=c(940109,940112,940113),
+ mcol="gold")
>
> combine.map(flexy=map,
+ plcid=c(940111,940104,940102),
+ mcol="yellow")
>
```

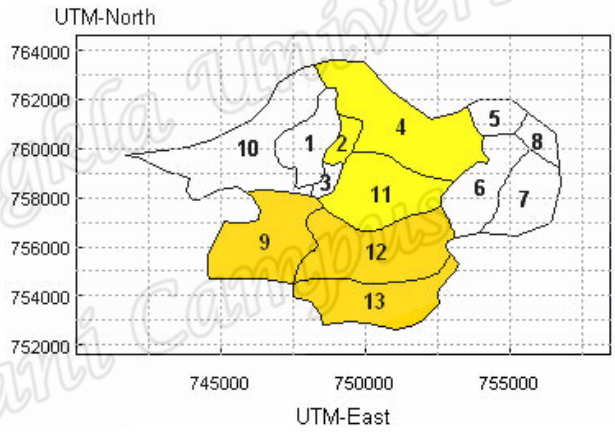

Figure 3.11: Example of combined regions with solid line

The colstat.*map*() function displays statistical data on the map. Users can supply a continuous or categorical variable for showing the contrasting region colors. If the user does not identify the color for each group of categorical variable, this function generates the color automatically. For continuous variables, user can specify the number of distinct categories, and the function will automatically divide the data into groups based on equal ranges. If the number of categories is not specified by the user, the function will determine the most appropriate number based on the total number of regions. The user can choose a type of line for each group such as dotted line, solid

line and removed line. This function also shows a legend of each group, but the user can choose to show or not show a legend. The name of legend and color of each group also will show on R Console.

For example, figure 3.12 shows the categorical variable. User does not specify color for each group. This function generates blue color for the first group, dark gold for the second group and green color for the third group. The color of each group will show on R Console, in the panel on the left. Figure 3.13 shows the categorical variable. User specifies green color for the first group, yellow color for the second group and red color for third group, and also uses a dotted line. Figure 3.14 shows the continuous variable. User does not specify number of categories and the function manages the data into 10 groups. Figure 3.15 shows the continuous variable with two groups. User specifies yellow color for the first group and red color for the second group.

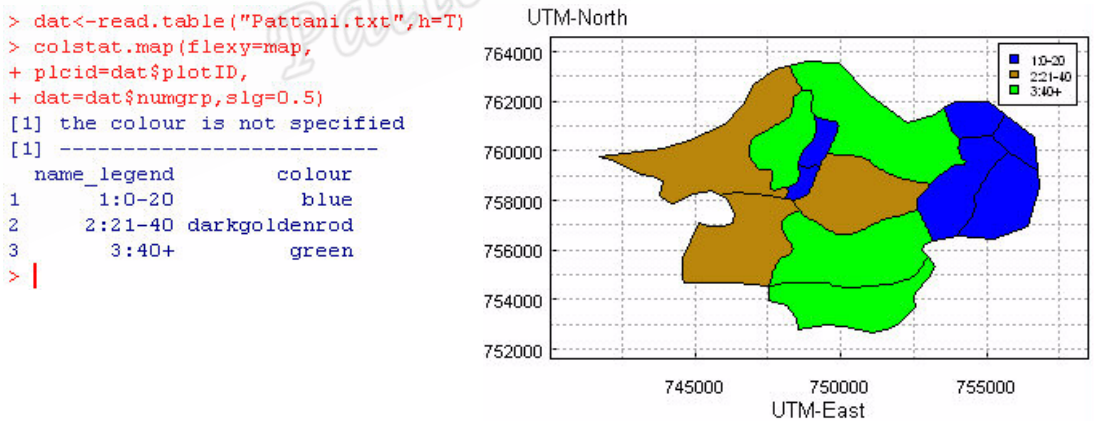

Figure 3.12: Categorical variable not specifying color

```
> dat<-read.table("Pattani.txt",h=T)
> colstat.map(flexy=map,
+ plcid=dat$plotID,
+ dat=dat$numgrp,
+ mcol=c("green","yellow","red"),
+ slg=0.5)
  name_legend colour
1      1:0-20  green
2     2:21-40 yellow
3       3:40+    red
>
```

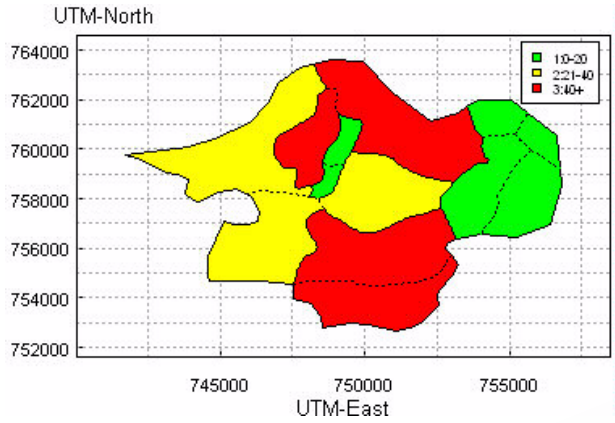

Figure 3.13: Categorical variable specifying color for each group and use of dotted line

```
> dat<-read.table("Pattani.txt",h=T)
> colstat.map(flexy=map,
+ plcid=dat$plotID,dat=dat$num,
+ pslg="br",slg=0.4,ncollg=2)
    name_legend        colour
1      1: 4 - 9          blue
2   2 : 10 - 15 darkgoldenrod
3   3 : 16 - 21         green
4   4 : 22 - 27    darkorchid
5   5 : 28 - 33           red
6   6 : 34 - 39    darksalmon
7   7 : 40 - 45      lavender
8   8 : 46 - 51      deeppink
9   9 : 52 - 57     limegreen
10 10 : 58 - 63         linen
>
```

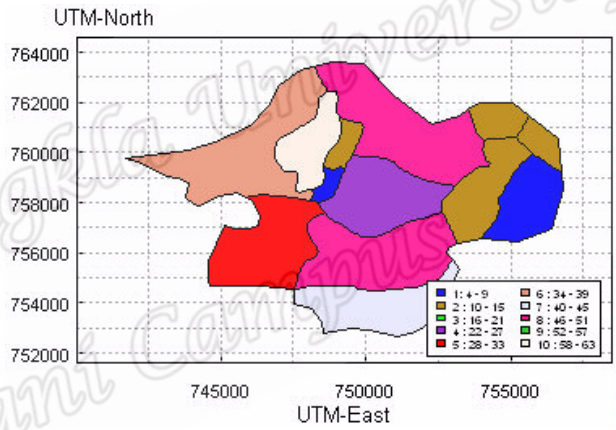

Figure 3.14: Continuous variable not specifying group

```
> dat<-read.table("Pattani.txt",h=T)
> colstat.map(flexy=map,
+ plcid=dat$plotID,dat=dat$num,
+ mcol=c("yellow","red"),grp=2,
+ slg=0.6)
  name_legend colour
1   1: 4 - 32 yellow
2 2 : 33 - 61    red
>
```

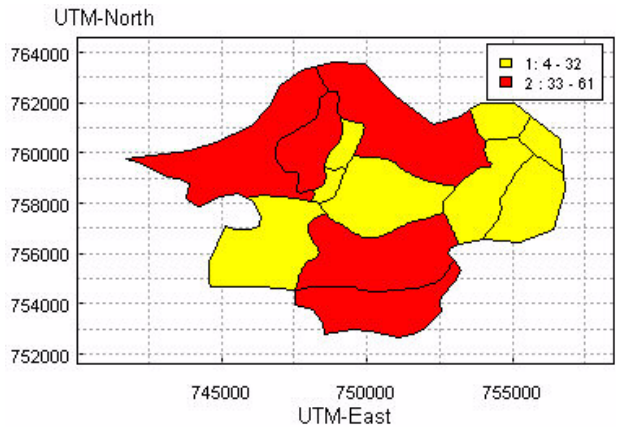

Figure 3.15: Continuous variable specifying group and color

The *piestat.map*() function displays the statistical data on the map like the *colstat.map*() function does, but this function also shows the circle sign on the map. The size of circle replaces statistical data in each group. The method for managing the variable is the same as with *colstat.map*() function. With this function the user can specify both the color of the circle and the size of the circle. The name of legend and size of circle are shown on R Console. For example, figure 3.16 shows three groups of categorical variable. User uses red color circle and starts size of circle with 0.5. Figure 3.17 shows the continuous variable. User creates map with blue color. The user specifies two groups, red color circle and start size of circle with 1.

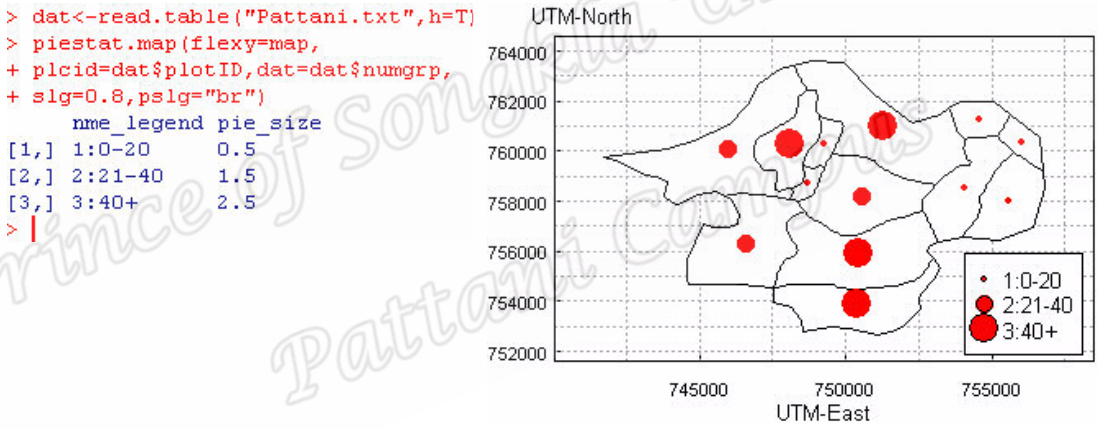

Figure 3.16: Example of a categorical variable

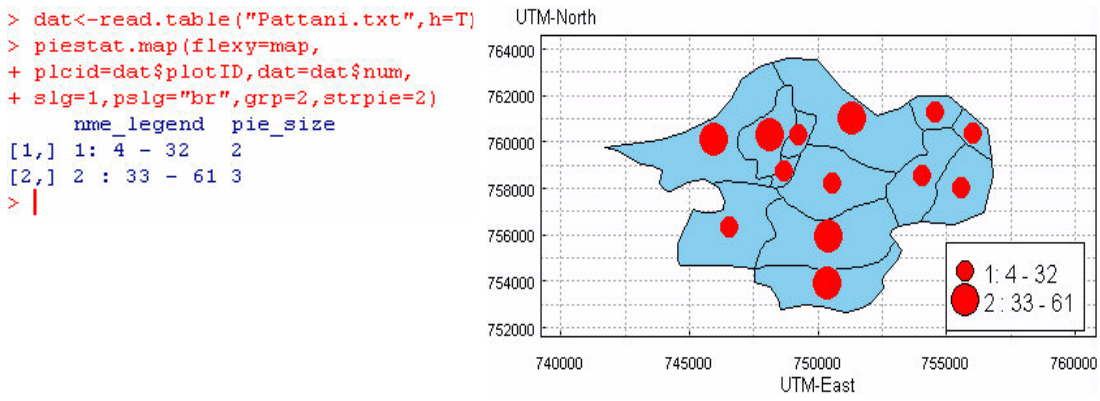

Figure 3.17: Example of a continuous variable

The *area.map*() function computes the area of each region. User can identify many regions to compute at one time and this function will show the result on R Console. For a complex region, this function computes the area of each region. User can specify to show the area of each region on a map, but this function defaults to 'do not show'. Figure 3.18 shows the area of each region for a simple region. The panel on the left shows the area of each region on R Console and the panel on the right shows on a map the area of each region. Figure 3.19 shows the area of a complex region.

```
> area.map(flexy=map,mshow=T)
    plcid              area
1  940101 4877344.84515381
2  940102 1483320.15002441
3  940103  864789.54510498
4  940104 13766829.6346436
5  940105 2360723.35992432
6  940106 6368714.45532227
7  940107 5865490.57489014
8  940108 1698088.70483398
9  940109 10127790.4850464
10 940110 13580488.7798462
11 940111 9108047.69042969
12 940112 12207051.3297729
13 940113 8342325.94506836
14  total  90651005.500061
>
```

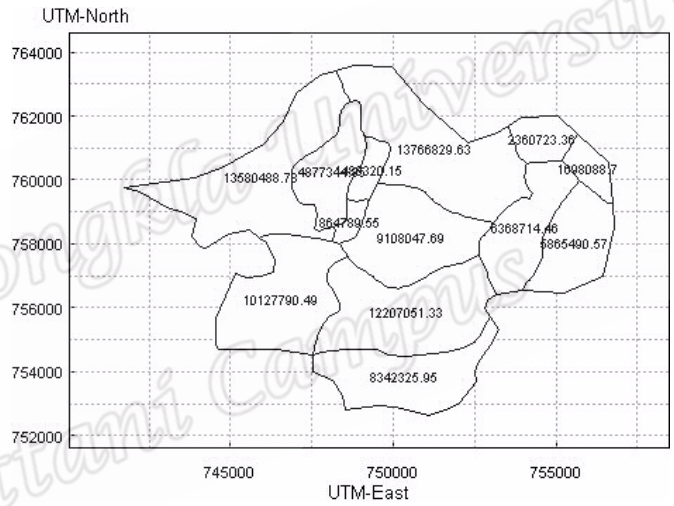

Figure 3.18: The area of a simple region

```
> area.map(flexy=map,mshow=T)
[1] the area of complex region are
[1] 208184
     region      area
[1,]      1 3102885.4
[2,]      2  652757.1
>
```

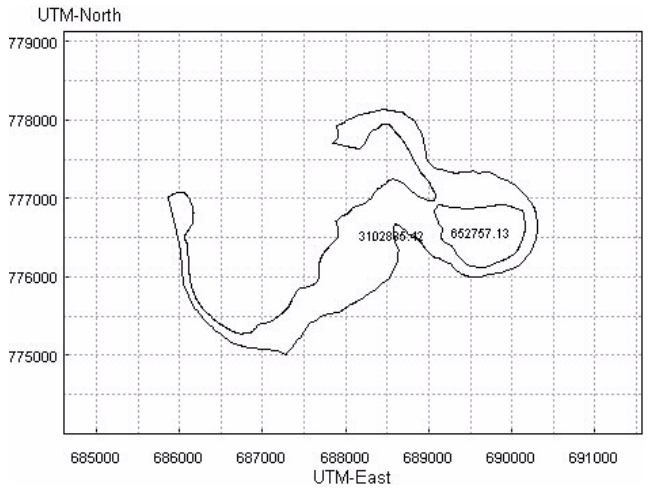

Figure 3.19: The area of a complex region

The *perimeter.map*() function computes the perimeter of each region. The result of this function is similar to the result of the *area.map*() function. An example is shown in figure 3.20 for a simple region and figure 3.21 shows a complex region.

```
> perimeter.map(flexy=map,
+ mshow=T)
    plcid perimeter
1  940101 10492.512
2  940102  5738.660
3  940103  4485.355
4  940104 18782.648
5  940105  5826.944
6  940106 11844.224
7  940107 10068.048
8  940108  5653.656
9  940109 14642.136
10 940110 23044.666
11 940111 12413.659
12 940112 16687.696
13 940113 15048.805
>
```

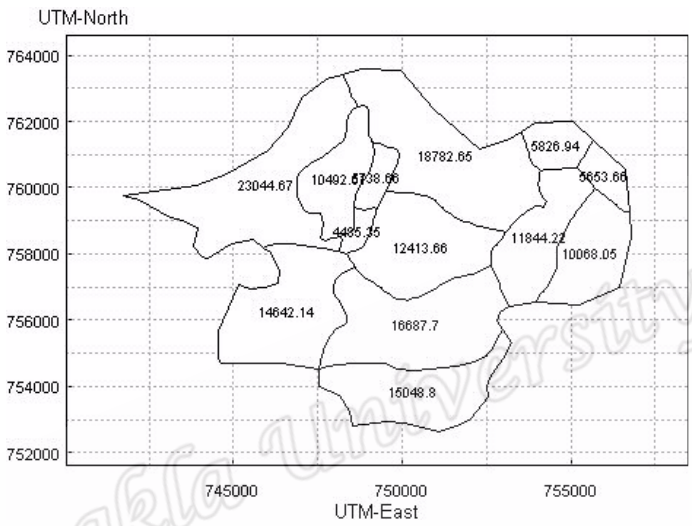

Figure 3.20: The perimeter of each region for simple regions

```
> perimeter.map(flexy=map,mshow=T)
[1] the perimeter of complex region are
[1] 208184
     region perimeter
[1,]      1 19007.909
[2,]      2  3184.411
>
```

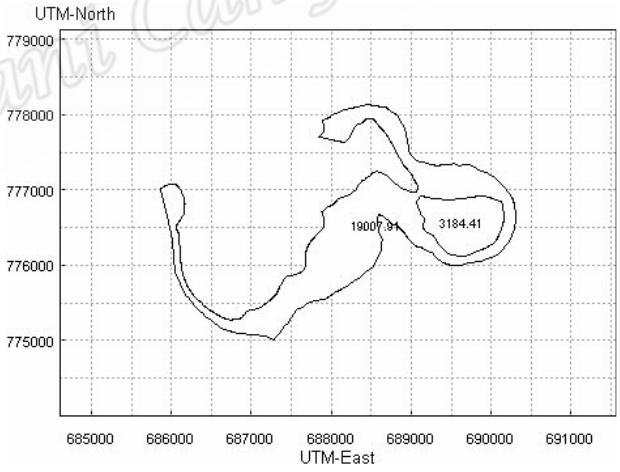

Figure 3.21: The perimeter of each region for a complex region

The *center.map*() function computes the center of each region. The result will show on R Console. An example is shown in figure 3.22. The left panel shows the center of each simple region. The right panel shows the center of each region for a complex region.

```
> center.map(flexy=map)
    plcid x_center y_center
1  940101 748120.5 760304.5
2  940102 749266.1 760312.2
3  940103 748708.0 758705.1
4  940104 751303.6 760984.4
5  940105 754568.3 761294.7
6  940106 754052.2 758509.8
7  940107 755592.2 758025.1
8  940108 756037.2 760377.8
9  940109 746579.2 756285.6
10 940110 745958.4 760120.6
11 940111 750564.8 758220.9
12 940112 750425.2 755895.0
13 940113 750359.4 753872.0
```

```
> center.map(flexy=map)
[1] the area of complex region are
[1] 208184
  region x_center y_center
1      1 688565.5 776532.1
2      2 689643.2 776562.0
>
```

Figure 3.22: The result on R Console shows the center of each region

In summary, the 10 functions are independent each other. The user is not necessary to call them in order. But the first step, user should call the *create.map*() function to create a map before calling other functions.

For next chapter, we will describe the conclusion and suggestion of this study and also ongoing work.