Creating Surface plots in R

Ralph Mansson

Introduction

A level plot is a type of graph used to display a surface, i.e. when the data has three dimensions, and displays the surface as if we were looking straight down from above. It is an alternative to a contour plot. In a contour plot lines are used to identify regions of different heights and the level plot uses coloured regions to produce a similar effect. This type of display is useful when considering data with a spatial feature such as terrain heights or counts of events at a given location.

To illustrate this type of graph we will consider surface elevation data that is available in the `geoR` package via the **R** software. The data set is called `elevation` and has elevation height in feet (multiples of ten) for a grid of x and y coordinates (recorded as multiples of 50 feet). To access this data we load the `geoR` pacakage and then use the `data` function:

```
require(geoR)
data(elevation)
```

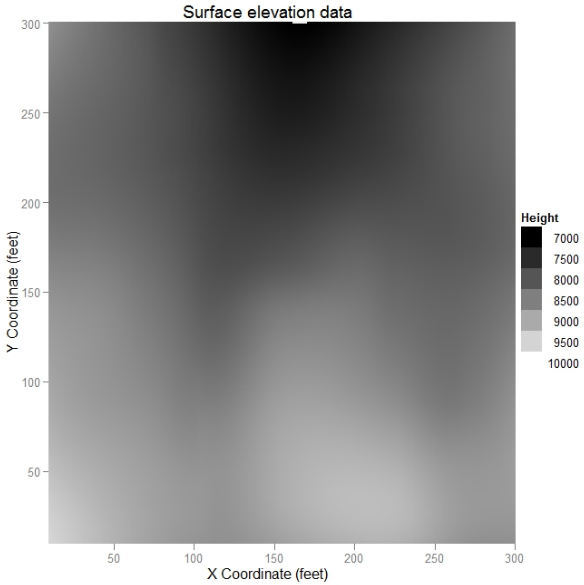
We make a copy of this data and save it in a data frame for use in creating the level plot:

```
elevation.df = data.frame(x = 50 *
elevation$coords[,"x"], y = 50 *
elevation$coords[,"y"], z = 10 *
elevation$data)
```

Level plots

To create the level plot we can fit a local trend surface via the `loess` function - a quadratic surface is estimated using weighted least squares:

tiles to cover the whole grid region covering up the default gray background. The graph that is produced is shown here:



The graph from **ggplot2** is visually as impressive as the other graphs - there is more smoothing between the colours which blurs some of the lines on the other graphs because of the type of colour gradient that was selected.

```
elevation.loess = loess(z ~ x*y, data =
elevation.df, degree = 2, span = 0.25)
```

The next step is to create an array of x and y co-ordinates covering the region of interest and then to calculate the height of the fitted local trend surface at these points. These values will then be used by the plotting functions to create the level plots.

```
elevation.fit = expand.grid(list(x = seq(10,
300, 1), y = seq(10, 300, 1)))
z = predict(elevation.loess, newdata =
elevation.fit)
elevation.fit$Height = as.numeric(z)
```

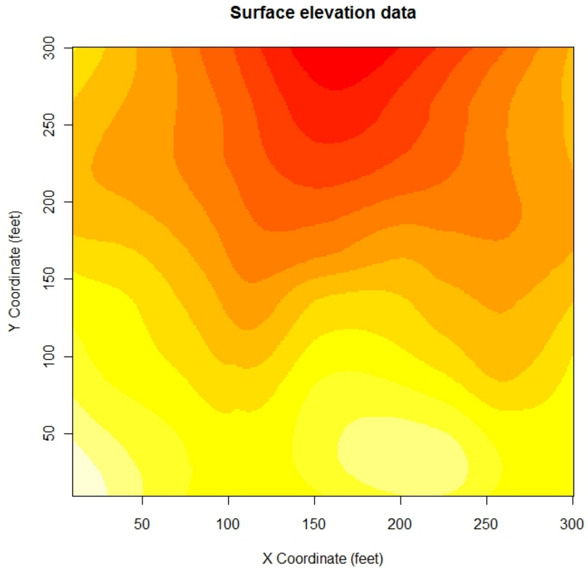
The function `expand.grid` creates a data frame based on the ranges of x and y specified above. The `predict` function then uses the fitted model object to estimate the height of the surface and this is saved in an object `z` as the different graph functions need the data in different formats. The fitted surface heights are converted to a numeric vector and attached as an additional column to the object that was used to create the predictions.

Base Graphics

The function `image` is the **base** graphics function for creating a level plot. This function requires a list of x and y values covering the grid of vertical values. The heights are specified as a table of values which was saved as the object `z` during the calculations. The text on the axis labels are specified by the `xlab` and `ylab` function arguments and the `main` argument determines the overall title for the graph.

```
image(seq(10, 300, 1), seq(10, 300, 1),
z, xlab = "X Coordinate (feet)", ylab = "Y
Coordinate (feet)", main = "Surface elevation
data")
box()
```

The function `box` is used to superimpose a box around the plotted surface.



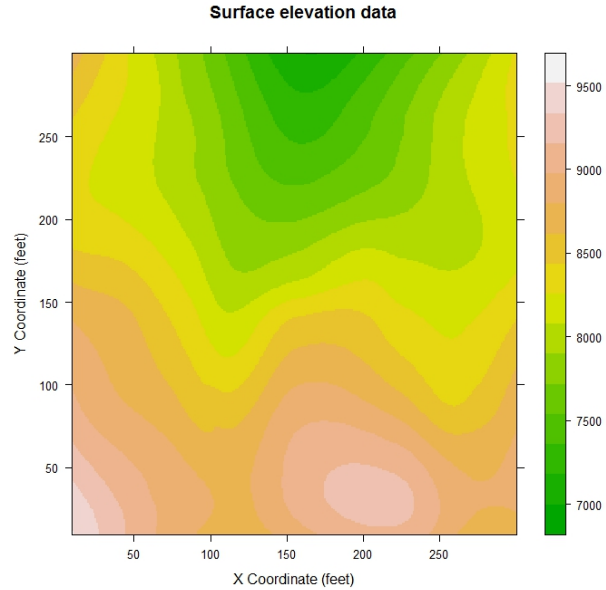
The default colour scheme produces an attractive graph where we can easily see the variation in height across the grid region displayed.

Lattice Graphics

The **lattice** graphics package has a function `levelplot` and we use the data in the object `elevation.fit` to create the graph.

```
levelplot(Height ~ x*y, data =
elevation.fit, xlab = "X Coordinate (feet)",
ylab = "Y Coordinate (feet)", main =
"Surface elevation data", col.regions =
terrain.colors(100))
```

The formula is used to specify which variable to use for the three axes and a data frame where the values are stored. The axes labels and title are specified in the same way as the **base** graphics. The range of colours used in the level plot can be specified as a vector of colours to the `col.regions` argument of the function. We make use of the `terrian.colors` function to create this vector which a range of 100 colours.



This graph is also visually appealing and easy to use but the default aspect ratio differs from the **base** graphics version.

ggplot2 Graphics

The **ggplot2** package also provides facilities for creating a level plot making use of the tile geom to create the desired graph. The function ggplot forms the basis of the graph and various other options are used to customise the graph.

```
ggplot(elevation.fit, aes(x, y, fill =
Height)) + geom_tile() + xlab("X Coordinate
(feet)") + ylab("Y Coordinate (feet)") +
opts(title = "Surface elevation data")
+ scale_fill_gradient(limits = c(7000,
10000), low = "black", high = "white")
+ scale_x_continuous(expand = c(0,0)) +
scale_y_continuous(expand = c(0,0))
```

The choice of colours used on graph is selected using the `scale_fill_gradient` function with colours ranging from black to white. The `scale_x_continuous` and `scale_y_continuous` options are used to stretch the