8. Gráficos con R (II)

8.1 Introducción a ggplot2

Basado en el libro The Grammar of Graphics [Wil05].

8.1.1 Nubes de puntos

Ejercicio 8.1 Gráfica de nube de puntos

```
> if(!is.installed('ggplot2'))
+   install.packages('ggplot2')
> library(ggplot2)
> # Nube de puntos
> qplot(Sepal.Length, Sepal.Width, data = iris,
+       colour = Species, size = Petal.Width)
```

8.1.2 Gráficas de líneas

Ejercicio 8.2 Gráfica de líneas

```
> qplot(Petal.Length, Sepal.Length, data=iris, color=Species) +
+   geom_line()
```

8.1.3 Añadir curva de regresión

Ejercicio 8.3 Nube de puntos con regresión entre ancho y alto de pétalo por cada especie

```
> qplot(Petal.Length, Petal.Width, data = iris, color = Species) +
+   geom_point() + geom_smooth()
> qplot(elevation, slope, data =
```

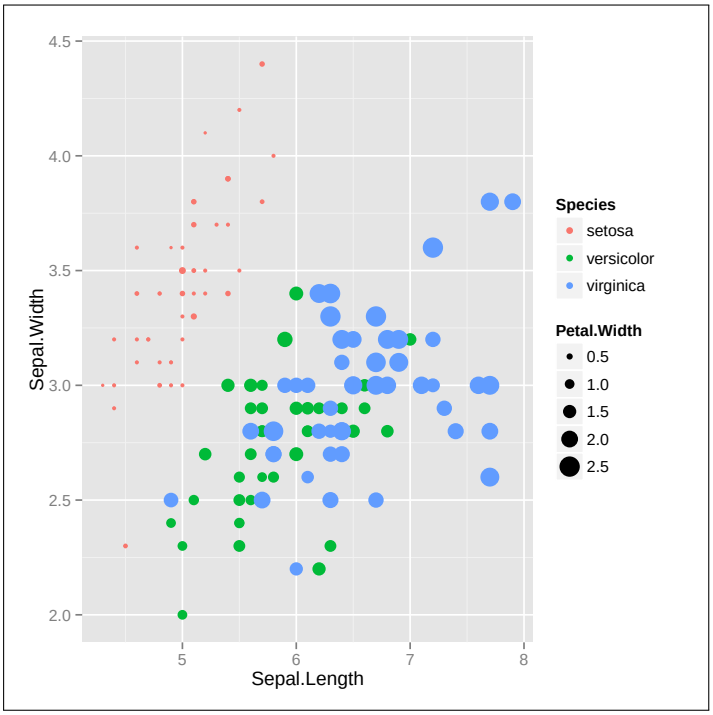

Figura 8.1: Nube de puntos mostrando cinco variables

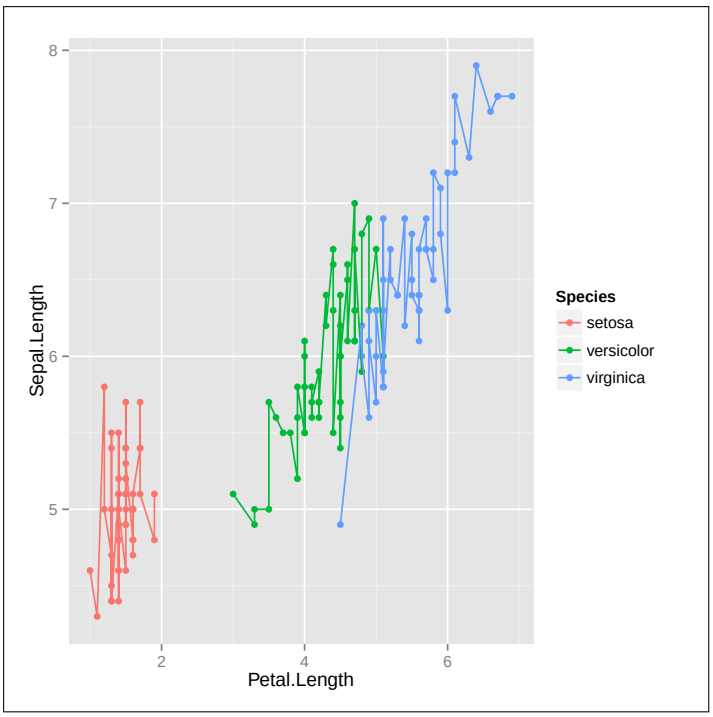

Figura 8.2: Gráfica de líneas

```
+         covertype[sample(1:nrow(covertype), 500),],
+       geom = c("point", "smooth"), color = wilderness_area) +
+   theme_bw()
```

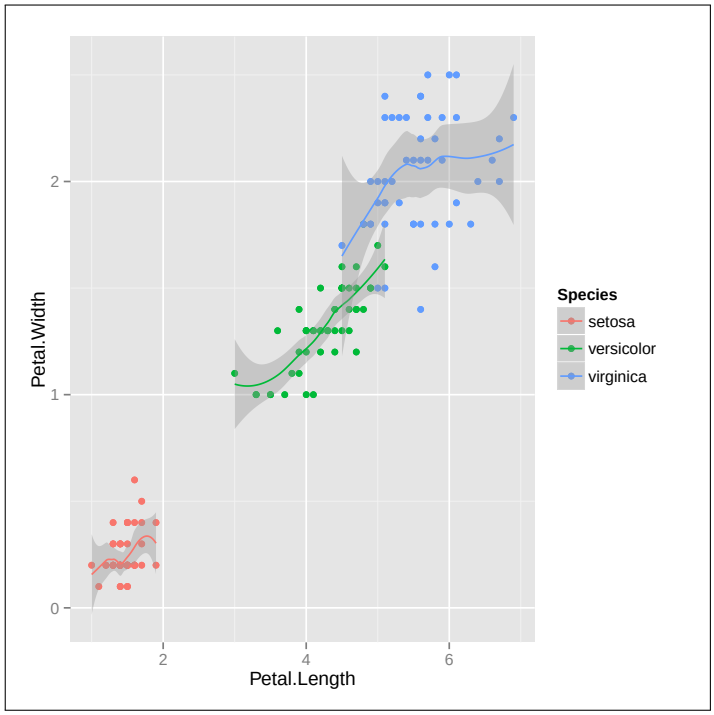

Figura 8.3: Nube de puntos con regresión entre ancho y alto de pétalo por cada especie

8.1.4 Curva de densidad

Ejercicio 8.4 Curva de densidad

```
> qplot(elevation, data = covertype, geom = "density",
+       fill = wilderness_area)
```

8.1.5 Composición de múltiples gráficas

Ejercicio 8.5 Facets: Representar 5 variables en una gráfica

```
> qplot(ClosePrice, sellerRating, data =
+         ebay[ebay$endDay %in% c('Wed','Thu','Fri') &
+                ebay$currency != 'US',],
+       facets = currency ~ endDay, color=Duration)
```

Ejercicio 8.6 Facets: Precio de cierre por moneda y día de finalización

```
> qplot(currency, ClosePrice, data=ebay[ebay$endDay != 'Wed',],
+       fill = currency) + geom_bar(stat = 'identity') +
+   facet_wrap(~endDay)
```

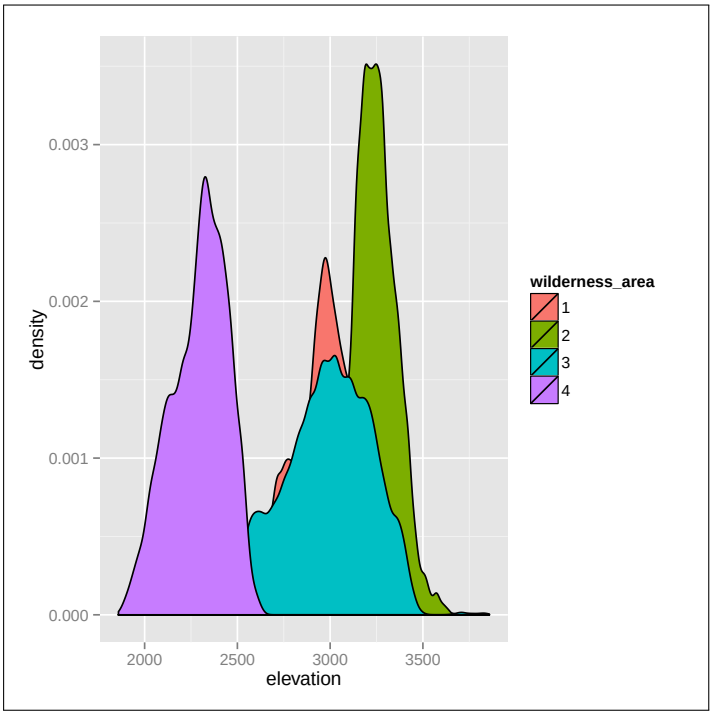

Figura 8.4: Curva de densidad

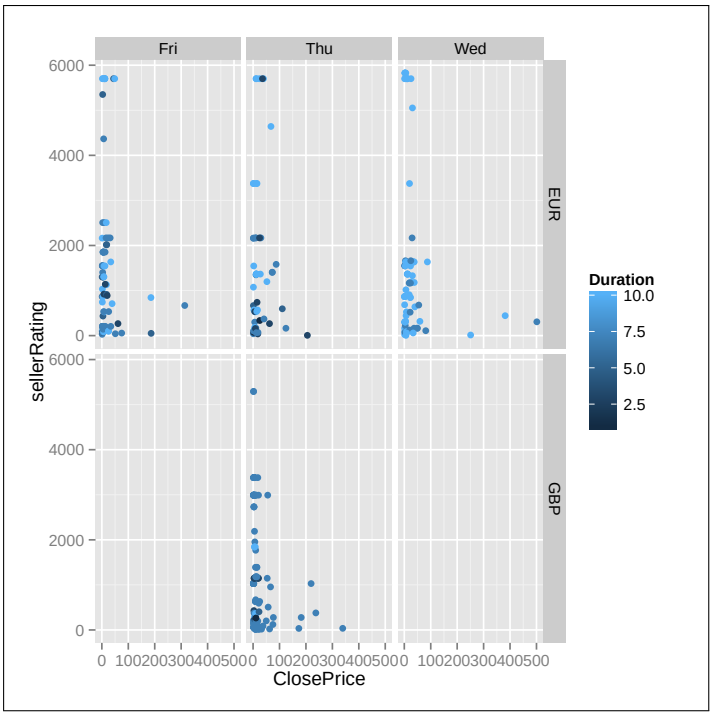

Figura 8.5: Facets: Representar 5 variables en una gráfica

8.2 Otras posibilidades gráficas

8.2.1 Dibujo de funciones y polinomios

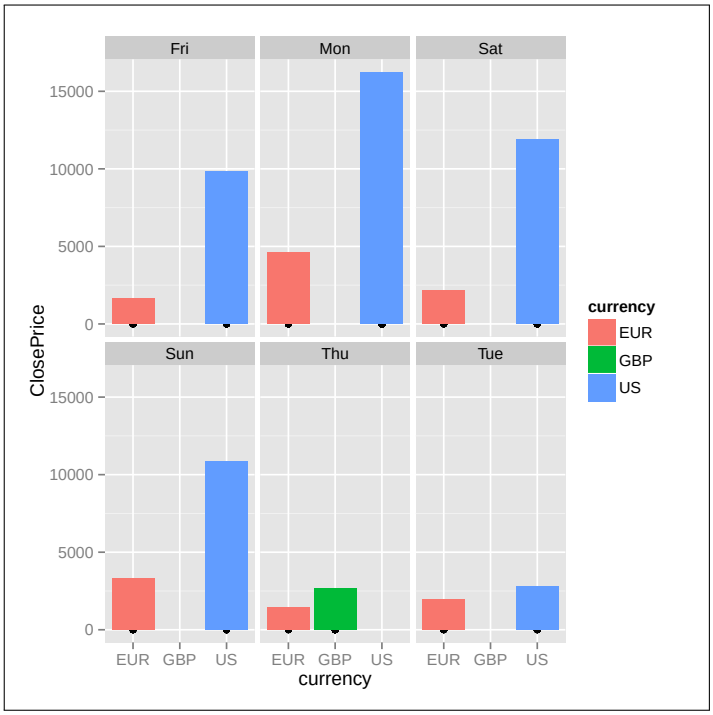

Figura 8.6: Facets: Precio de cierre por moneda y día de finalización

Ejercicio 8.7 Dibujo de funciones seno y coseno

```
> curve(sin, from = -4, to = 4, col = 'blue',
+       lty = 'dotted', lwd = 2, ylab = 'sin(x) vs cos(x)')
> curve(cos, from = -4, to = 4, col = 'cyan',
+       lty = 'dashed', lwd = 2, add = T)
> legend("topleft", c("sin(x)", "cos(x)"),
+        col = c('blue', 'cyan'), lty = c('dotted','dashed'),
+        lwd = 2, ncol = 2)
```

Ejercicio 8.8 Dibujo de un polinomio

```
> curve(x^3-x+1, lty = 2, from = -10, to = 10)
```

8.2.2 circlize

Gráficos tipo 'circos', muy usados en genética pero que pueden tener otras aplicaciones, por ejemplo para mostrar interacciones de todo tipo (migracion-emigración, por ejemplo)

Ejercicio 8.9 Gráfica tipo 'circos'

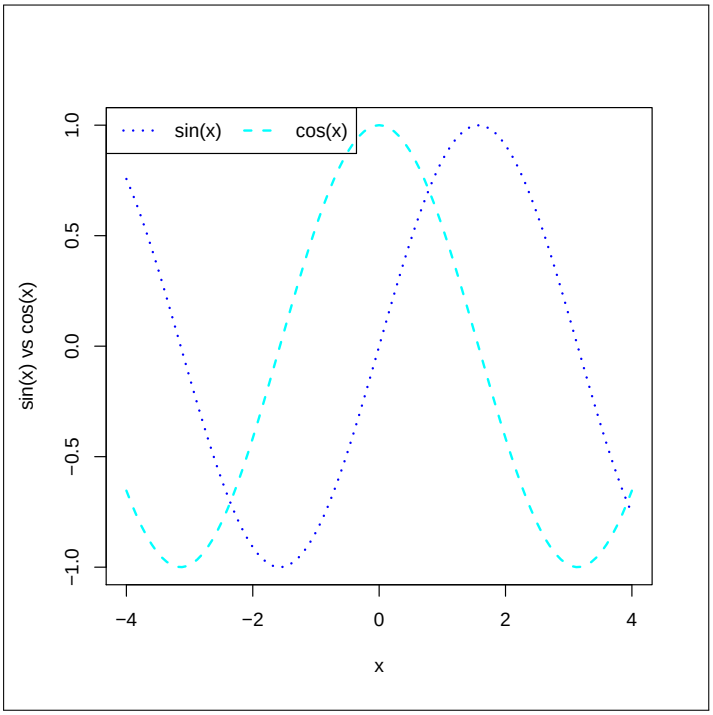

Figura 8.7: Dibujo de funciones seno y coseno

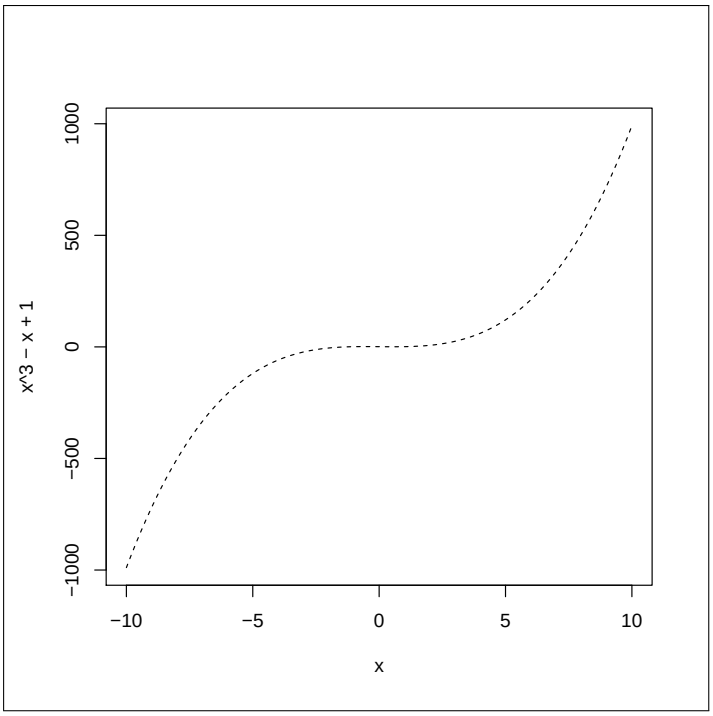

Figura 8.8: Dibujo de un polinomio

```
> if (!is.installed("circlize")) install.packages("circlize")
> library(circlize)
> mat = matrix(sample(1:100, 18, replace = TRUE), 3,
+     6)
> rownames(mat) = letters[1:3]
```

```
> colnames(mat) = LETTERS[1:6]
> rn = rownames(mat)
> cn = colnames(mat)
> par(mar = c(1, 1, 1, 1))
> circos.par(gap.degree = c(rep(2, nrow(mat) - 1),
+     10, rep(2, ncol(mat) - 1), 10))
> chordDiagram(mat, annotationTrack = "grid", transparency = 0.5,
+     preAllocateTracks = list(track.height = 0.1))
> for (si in get.all.sector.index()) {
+     circos.axis(h = "top", labels.cex = 0.3, sector.index = si,
+         track.index = 2)
+ }
> circos.trackPlotRegion(track.index = 1, panel.fun = function(x,
+     y) {
+     xlim = get.cell.meta.data("xlim")
+     ylim = get.cell.meta.data("ylim")
+     sector.name = get.cell.meta.data("sector.index")
+     circos.lines(xlim, c(mean(ylim), mean(ylim)),
+         lty = 3)
+     for (p in seq(0, 1, by = 0.25)) {
+         circos.text(p * (xlim[2] - xlim[1]) + xlim[1],
+             mean(ylim), p, cex = 0.4, adj = c(0.5,
+                 -0.2), niceFacing = TRUE)
+     }
+     circos.text(mean(xlim), 1.4, sector.name, niceFacing = TRUE)
+ }, bg.border = NA)
> circos.clear()
```

8.2.3 radarchart

Ejercicio 8.10 Gráfica tipo 'spider' o 'radar'

```
> if (!is.installed("fmsb")) install.packages("fmsb")
> library("fmsb")
> set.seed(4242)
> maxmin <- data.frame(emotions = c(1, 0), corel = c(1,
+     0), scene = c(1, 0), yeast = c(1, 0), ebay = c(1,
+     0))
> dat <- data.frame(emotions = runif(3, 0, 1), corel = runif(3,
+     0, 1), scene = runif(3, 0, 1), yeast = runif(3,
+     0, 1), ebay = runif(3, 0, 1))
> dat <- rbind(maxmin, dat)
> radarchart(dat, axistype = 2, plty = 1:3, plwd = 2,
+     vlabels = names(dat))
```

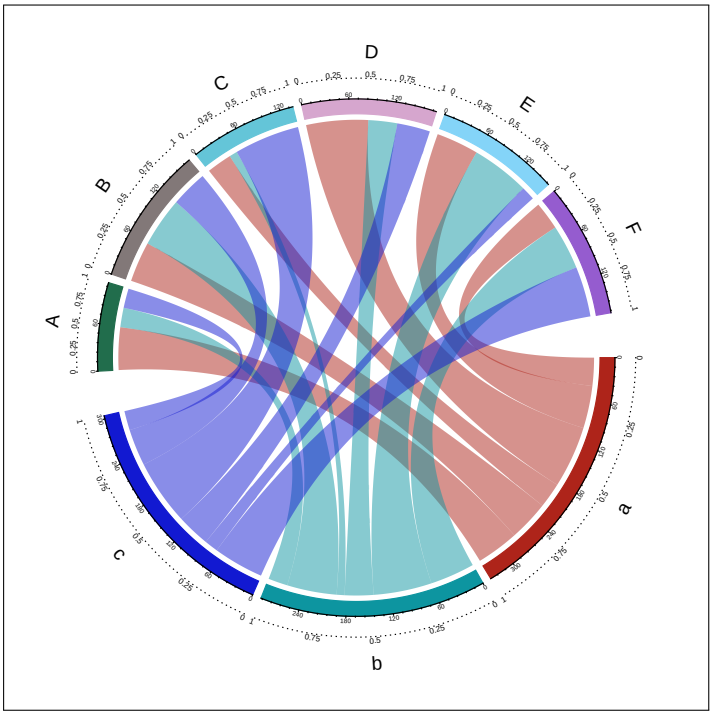

Figura 8.9: Gráfica tipo 'circos'

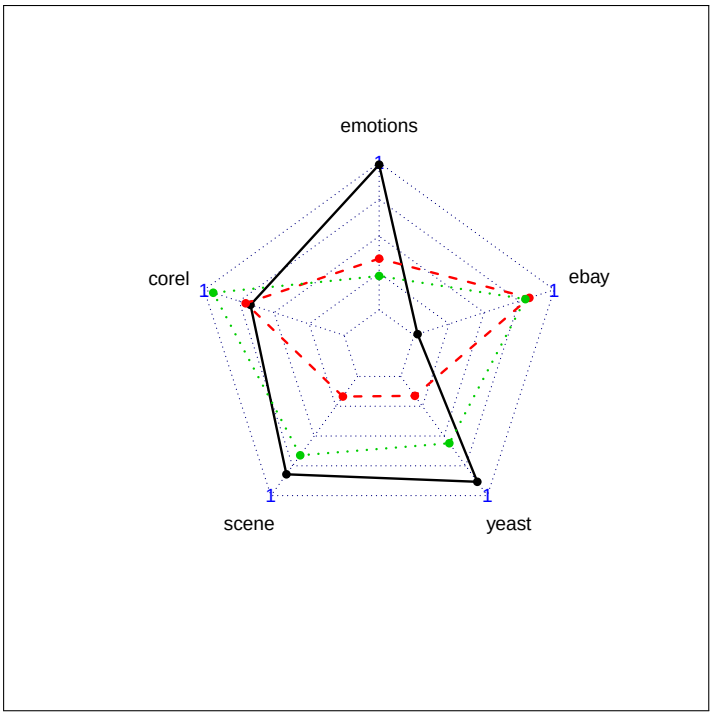

Figura 8.10: Gráfica tipo 'spider' o 'radar'

8.2.4 Gráficas 3D

scatterplot3d

Ejercicio 8.11 Gráfica tridimensional con `scatterplot3d()`

```
> if(!is.installed('scatterplot3d'))
+   install.packages('scatterplot3d')
> library('scatterplot3d')
> z <- seq(-10, 10, 0.01)
> x <- cos(z)
> y <- sin(z)
> scatterplot3d(x, y, z, highlight.3d = TRUE, col.axis = "blue",
+ col.grid = "lightblue", main = "Helix", pch = 20)
```

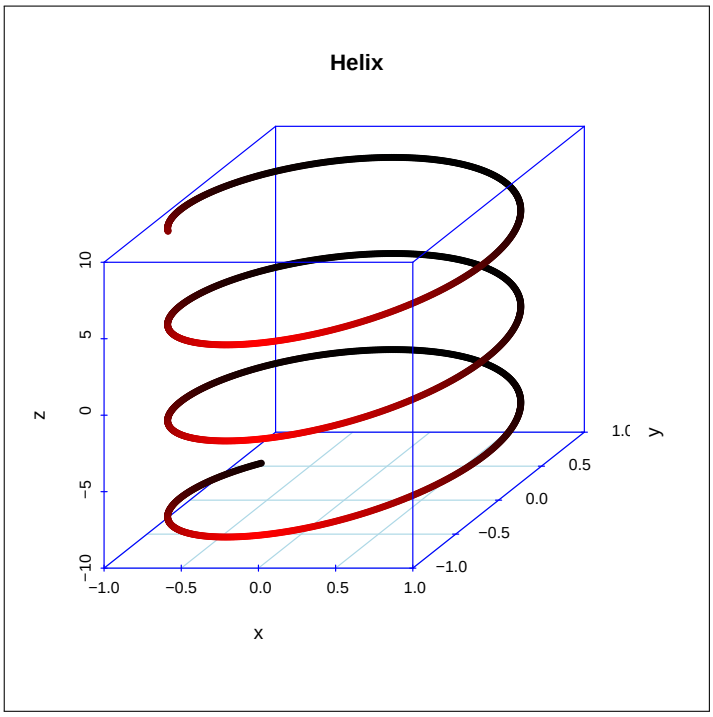

Figura 8.11: Gráfica tridimensional con `scatterplot3d()`

lattice

Ejercicio 8.12 Gráfica tridimensional con `lattice`

```
> if (!is.installed("lattice")) install.packages("lattice")
> library("lattice")
> z <- matrix(rnorm(625) + 574, nrow = 25)
> z <- z + seq(50, 1, length = 25)
> persp(z, phi = 30, theta = 30, zlim = c(550, 650),
+     xlab = "X", ylab = "Y", zlab = "Z", main = "Elevación del terreno")
```

8.3 Gráficos de tortuga

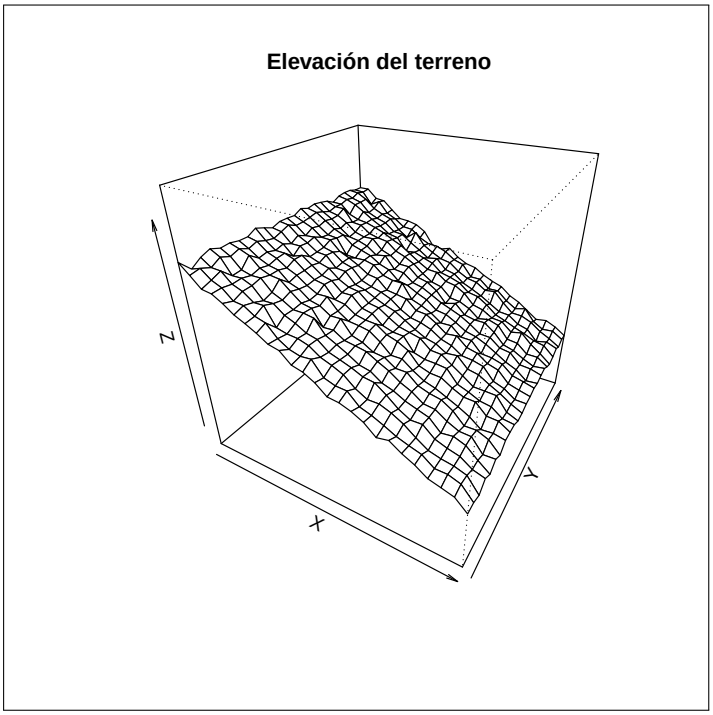

Figura 8.12: Gráfica tridimensional con lattice

Ejercicio 8.13 Gráficos de tortuga: Triángulo de Sierpinski

```
> if(!is.installed('TurtleGraphics'))
+   install.packages('TurtleGraphics')
> library('TurtleGraphics')
> drawTriangle<- function(points){
+   turtle_setpos(points[1,1],points[1,2])
+   turtle_goto(points[2,1],points[2,2])
+   turtle_goto(points[3,1],points[3,2])
+   turtle_goto(points[1,1],points[1,2])
+   }
>   getMid<- function(p1,p2) c((p1[1]+p2[1])/2, c(p1[2]+p2[2])/2)
>   sierpinski <- function(points, degree){
+   drawTriangle(points)
+   if (degree  > 0){
+   p1 <- matrix(c(points[1,], getMid(points[1,], points[2,]),
+   getMid(points[1,], points[3,])), nrow=3, byrow=TRUE)
+
+   sierpinski(p1, degree-1)
+   p2 <- matrix(c(points[2,], getMid(points[1,], points[2,]),
+   getMid(points[2,], points[3,])), nrow=3, byrow=TRUE)
+
+   sierpinski(p2, degree-1)
+   p3 <- matrix(c(points[3,], getMid(points[3,], points[2,]),
+   getMid(points[1,], points[3,])), nrow=3, byrow=TRUE)
+   sierpinski(p3, degree-1)
+   }
```

```
+   invisible(NULL)
+   }
>   turtle_init(520, 500, "clip")
>   p <- matrix(c(10, 10, 510, 10, 250, 448), nrow=3, byrow=TRUE)
>   turtle_col("red")
>   turtle_do(sierpinski(p, 6))
>   turtle_setpos(250, 448)
```

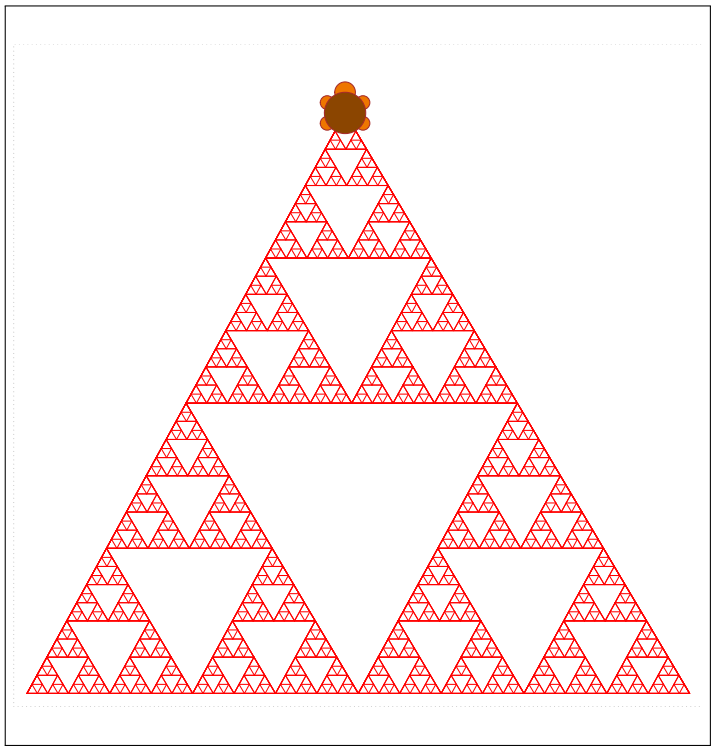

Figura 8.13: Gráficos de tortuga: Triángulo de Sierpinski

Ejercicio 8.14 Gráficos de tortuga: Espiral de hexágonos

```
> turtle_init()
> turtle_do({
+ for(j in 1:45) {
+   for(i in 1:6) {
+     turtle_forward(20)
+     turtle_right(360/6)
+   }
+  turtle_right(360/45)
+ }
+ })
```

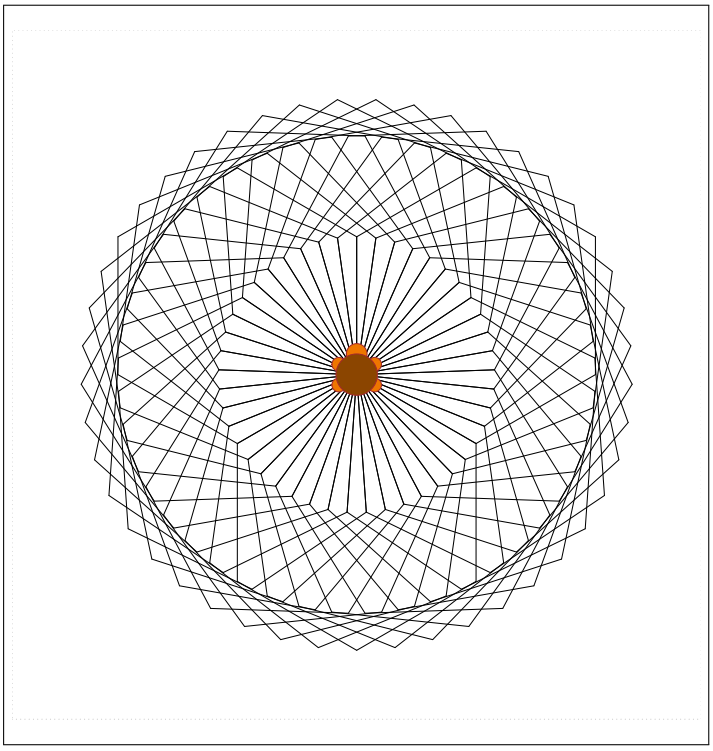

Figura 8.14: Gráficos de tortuga: Espiral de hexágonos