Is image() Everything?

Paul Murrell

The University of Auckland

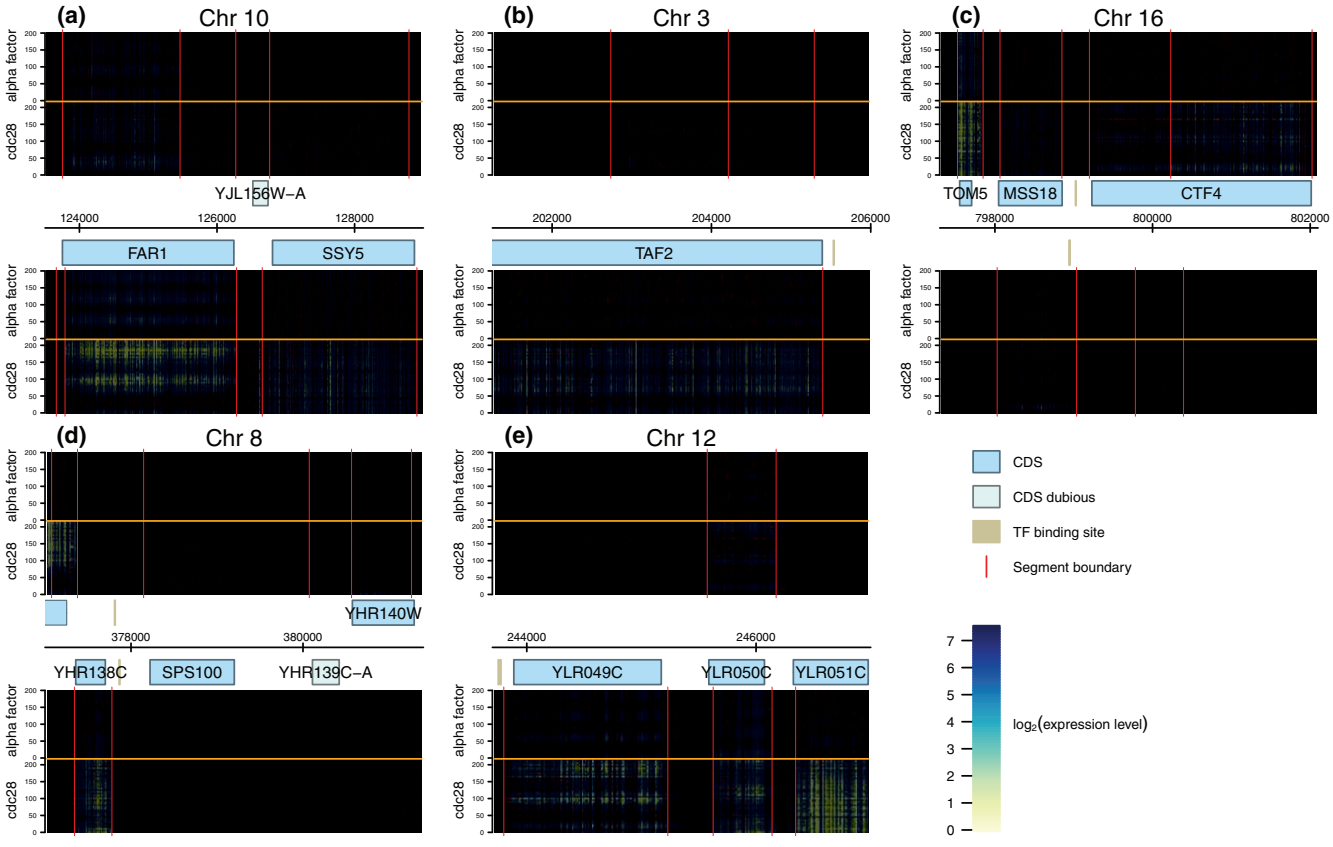
July 2011

Raster support in R graphics

- From version 2.11.0 R has native support for **raster images**.
  - `as.raster()`
  - `rasterImage()`
  - `grid.raster()`
  - prior to that, raster images were drawn as a matrix of rectangles.
- So what?

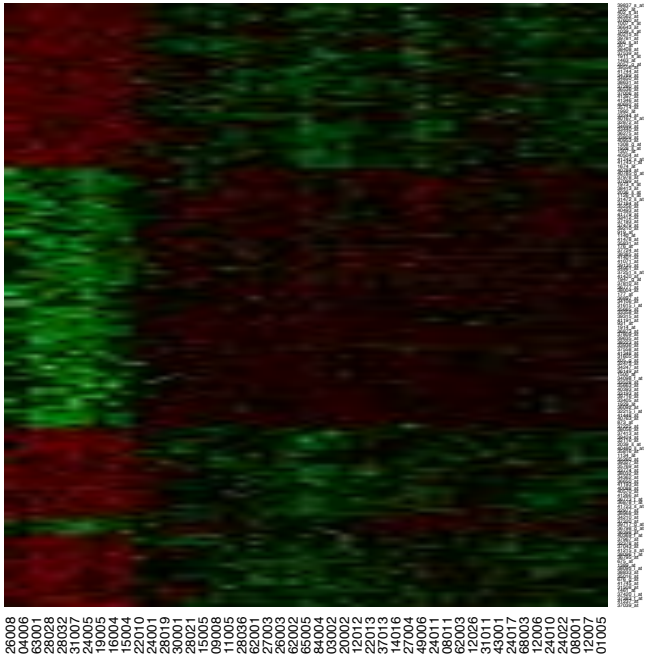
The original reason

- Do raster graphical elements the right way.
  - Some sorts of plots, e.g., heatmaps, contain graphical elements that are naturally raster in nature.



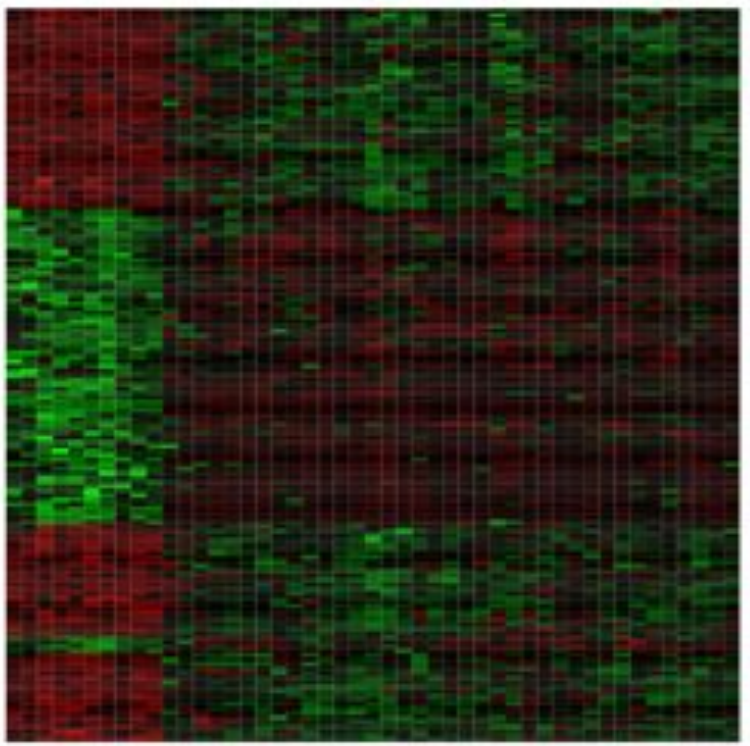
The original reason

- Practical considerations:
  - Smaller files.
    For example, the PDF versions of the heatmap below using rectangles vs raster are 45k vs 13k.
  - Faster drawing.
    Not just when R produces the graphic but when viewing software renders the graphic.



Some bonuses

- No viewer artifacts.
  (This is svg() output viewed in Firefox 3)



Some bonuses

- Interpolation

Infographics in R

- Start to conceive of doing this sort of thing in R (A GOOD Blog post)

Infographics in R

- It becomes simpler to think about incorporating raster images as part of an R graphics image.
- The raster image can not only be raster data, but also an external raster image, such as a digital photo.

```
> library(png)
> bg <- readPNG("AfterTheBombs.png")
> library(grid)
> grid.raster(bg, height=1)

> grid.polygon(x, y,
                gp=gpar(col=NA,
                        fill=rgb(.67, 0, .11, .7)))
```

Infographics in R

Image processing in R

- Once we have raster images in R, they are essentially just matrices.
- We can perform simple image processing via matrix manipulation.

```
> png("mask.png")
> grid.polygon(x, y, gp=gpar(col=NA, fill="black"))
> dev.off()
> mask <- readPNG("mask.png")
> maskRaster <- as.raster(mask)
> bgRaster <- as.raster(bg)
> bgMask <- bgRaster[maskRaster == "#000000"]
```

Image processing in R

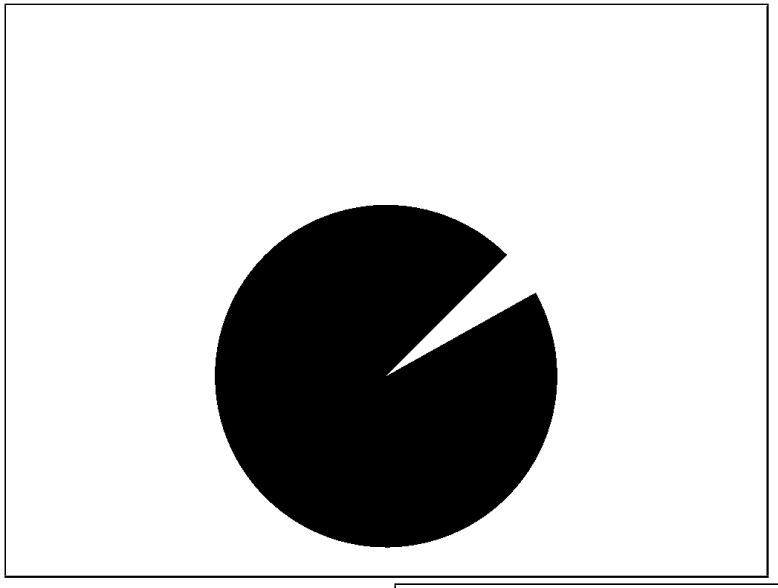


Image processing in R

```
> bgMaskRGB <- col2rgb(bgMask)
> bgRed <- rgb((0.3*bgMaskRGB[1, ] +
                0.59*bgMaskRGB[2, ] +
                0.11*bgMaskRGB[3, ]),
               0, 0, max=255)
> bgRaster[maskRaster == "#000000"] <- bgRed
```

Image processing in R


That's 96 Percent
The Department of Defense is unable to account for the use of $8.7 billion of the $9.1 billion it spent on reconstruction in Iraq
UNACCOUNTED FOR
ACCOUNTED FOR

Image processing in R

- There is always a danger that people will get carried away with this sort of feature ...

Image processing in plots

- ... but there are also some serious applications.
  For example, a plot with a legend that has a "blank" background.

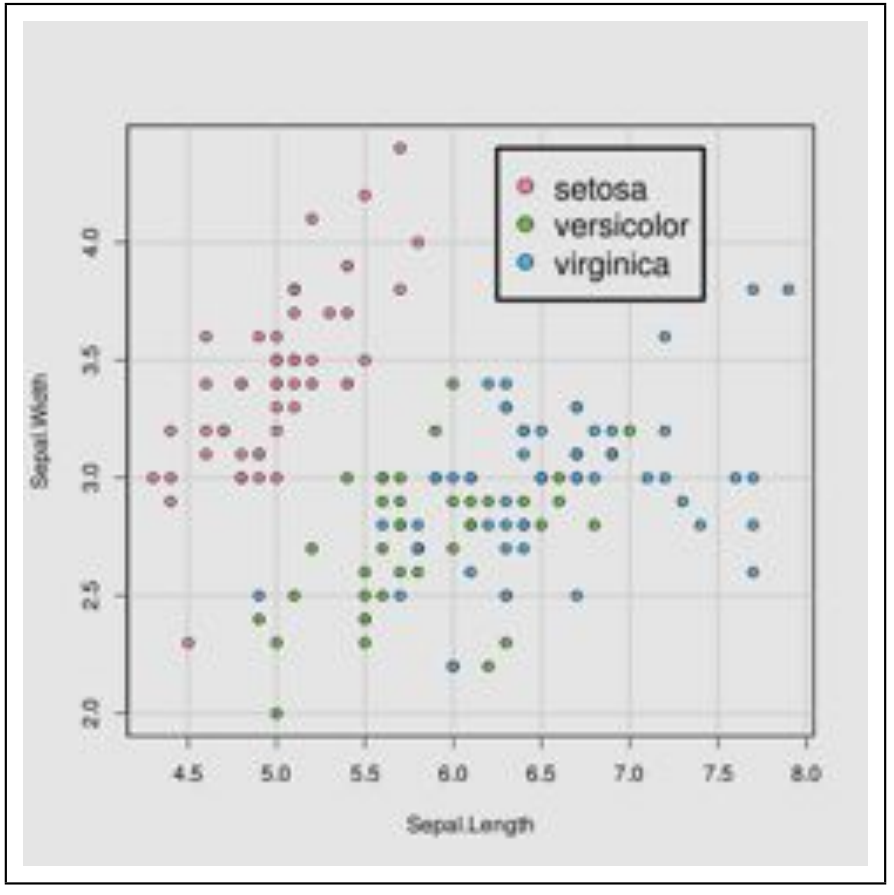


Image processing in plots

- A transparent background for the legend is not sufficient.

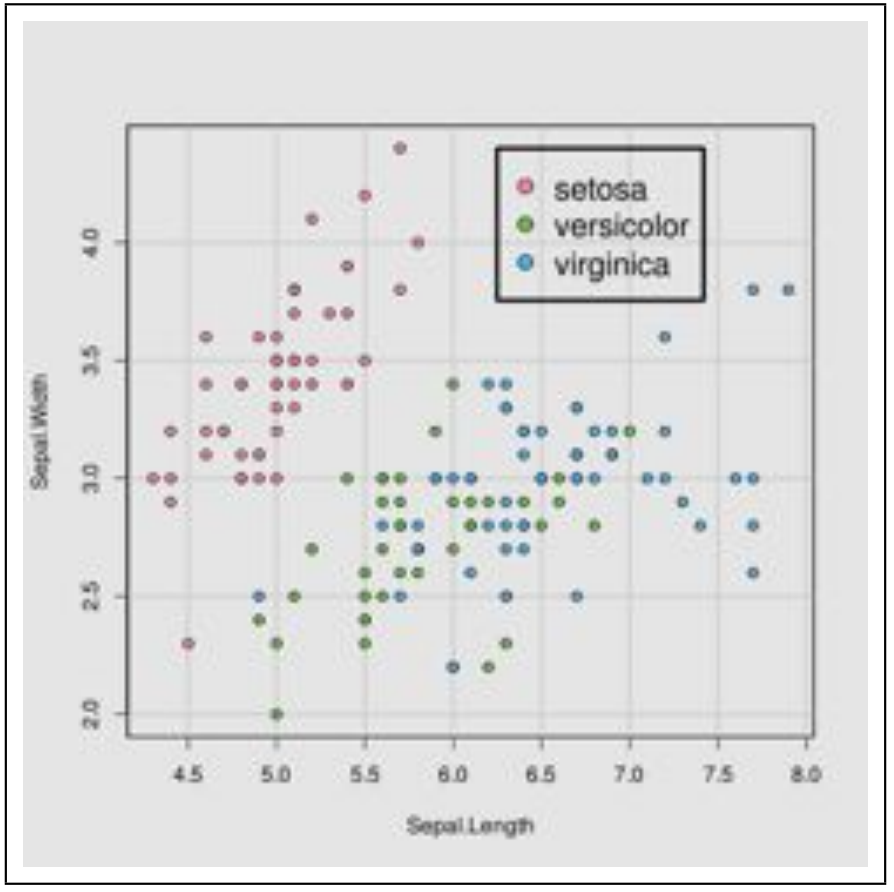


Image processing in plots

- A solid colour background for the legend is not a good general solution.

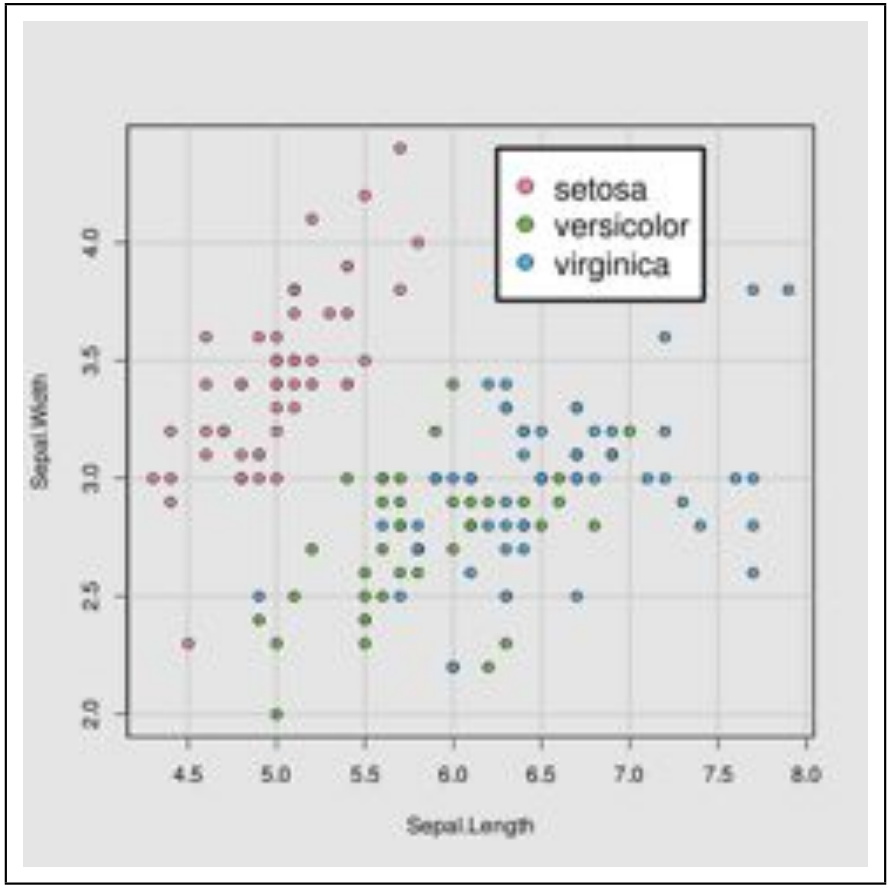


Image processing in plots

```
plot(Sepal.Length, Sepal.Width)
```

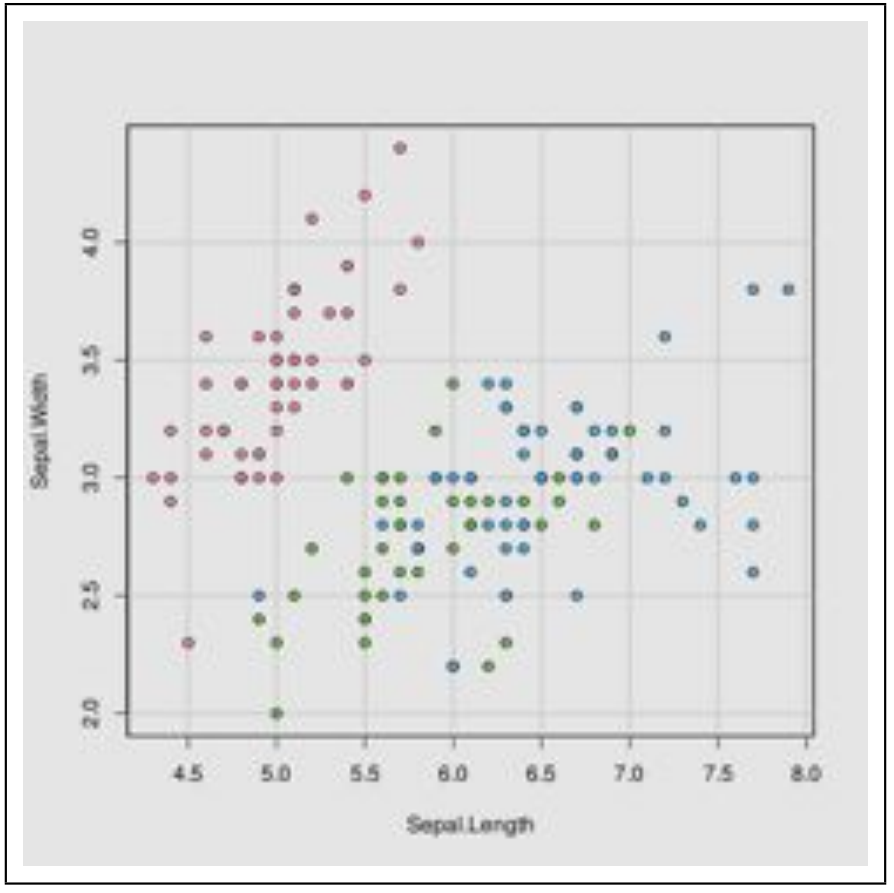


Image processing in plots

```
plot.new()
legend(bg="transparent")
```

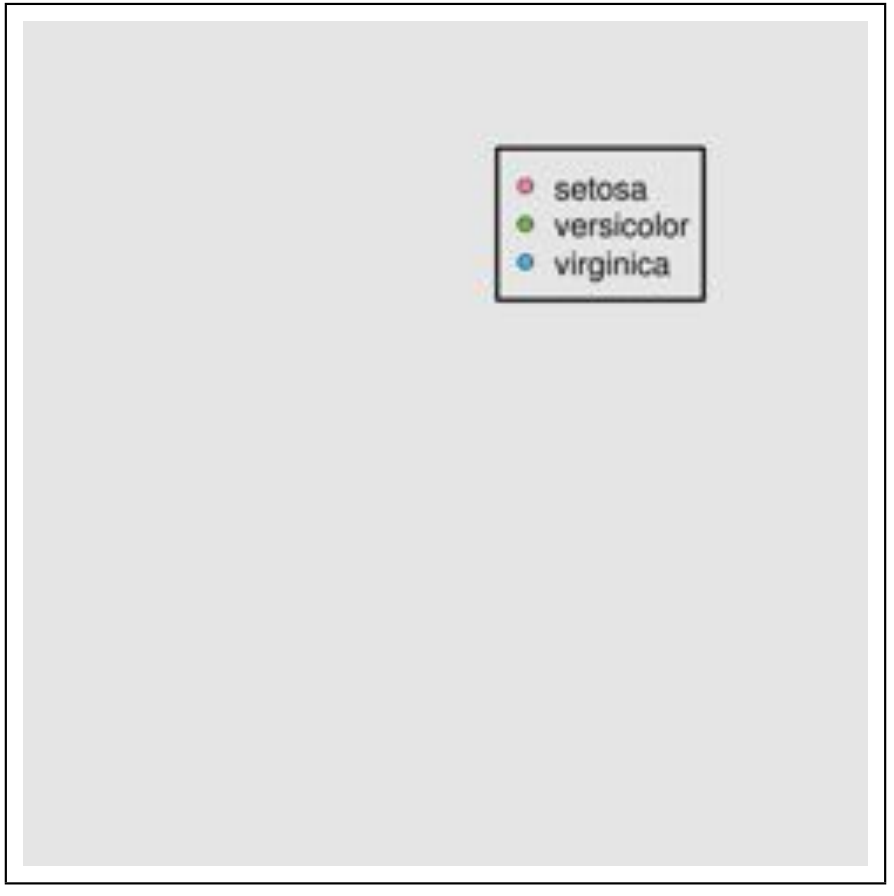


Image processing in plots

```
plot.new()
legend(bg="black")
```

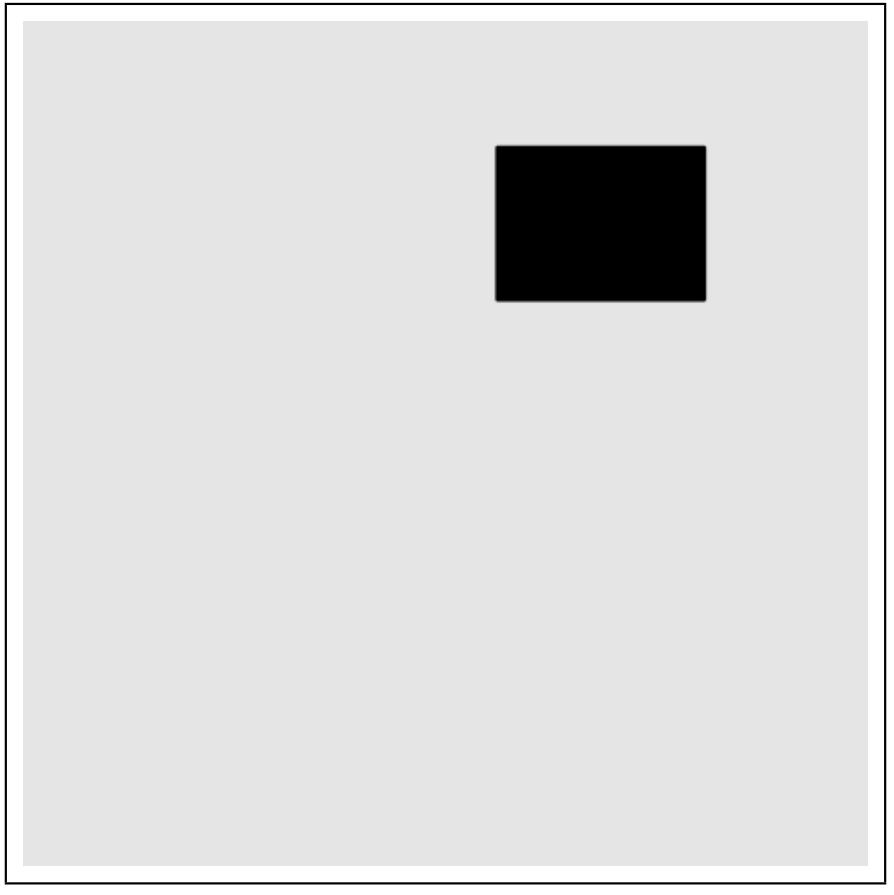


Image processing in plots

```
plot[legendMask == "black"] <- "transparent"
```

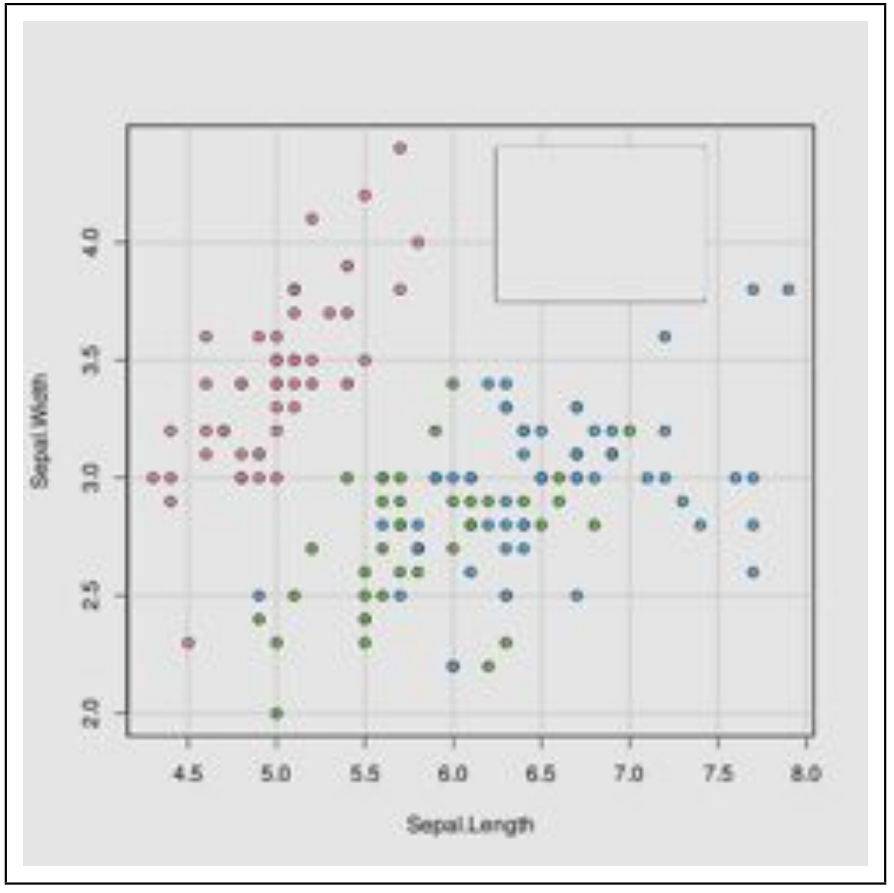
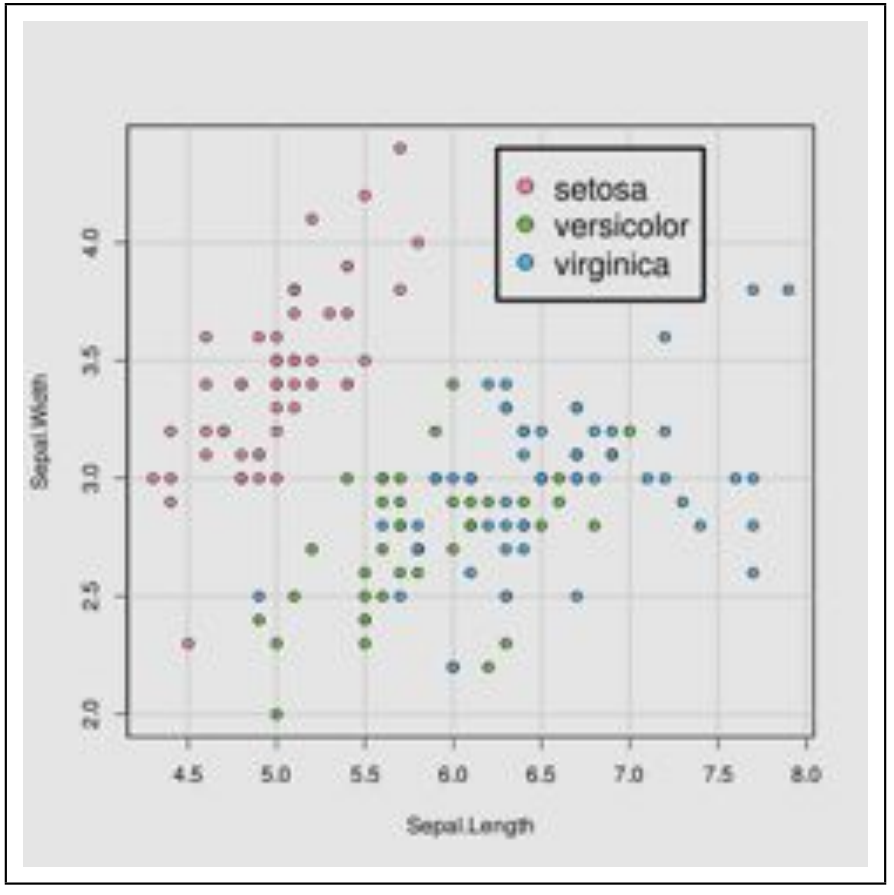


Image processing in plots

```
plot[legendMask == "black"] <- legend[legendMask == "black"]
```



Arranging images in R

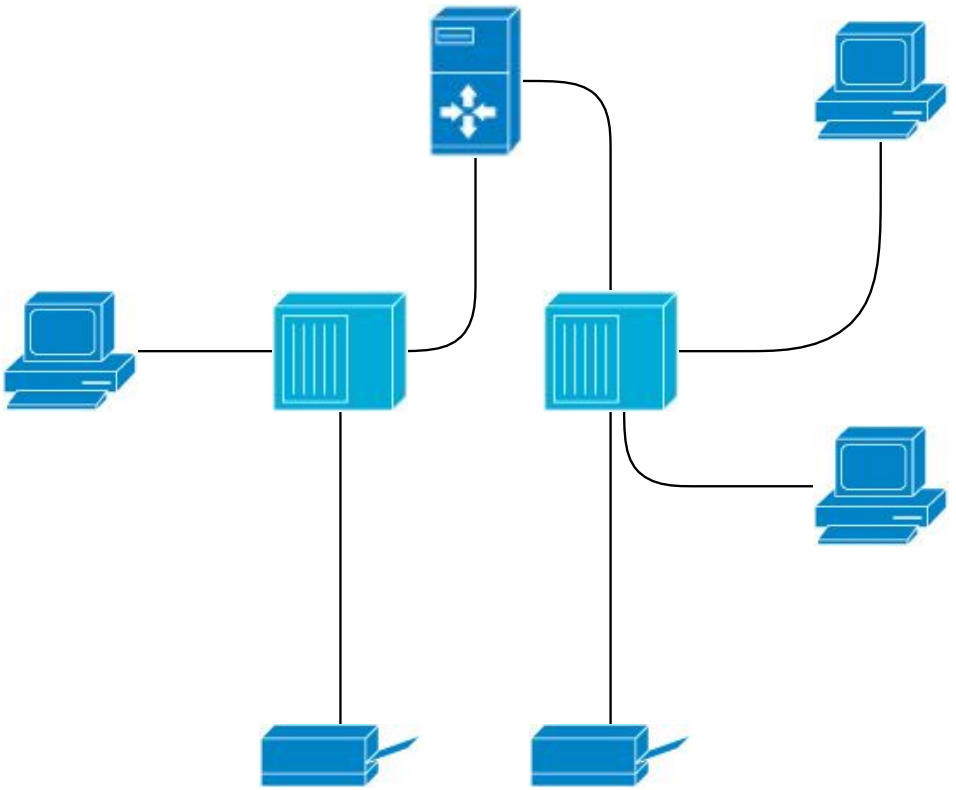
- It can be useful to arrange images using R graphics facilities.
  - For example, arrange images using **grid** coordinate systems.

```
grid.raster(server, y=.7,
            width=unit(7, "mm"))
grid.raster(hub, x=c(.4, .6),
            width=unit(1, "cm"))
grid.raster(pc, x=c(.2, .8, .8), y=c(.5, .7, .4),
            width=unit(1, "cm"))
grid.raster(printer, x=c(.4, .6), y=.2,
            width=unit(12, "mm"))
```

Arranging images in R

- Cisco Network Icons.

Arranging images in R

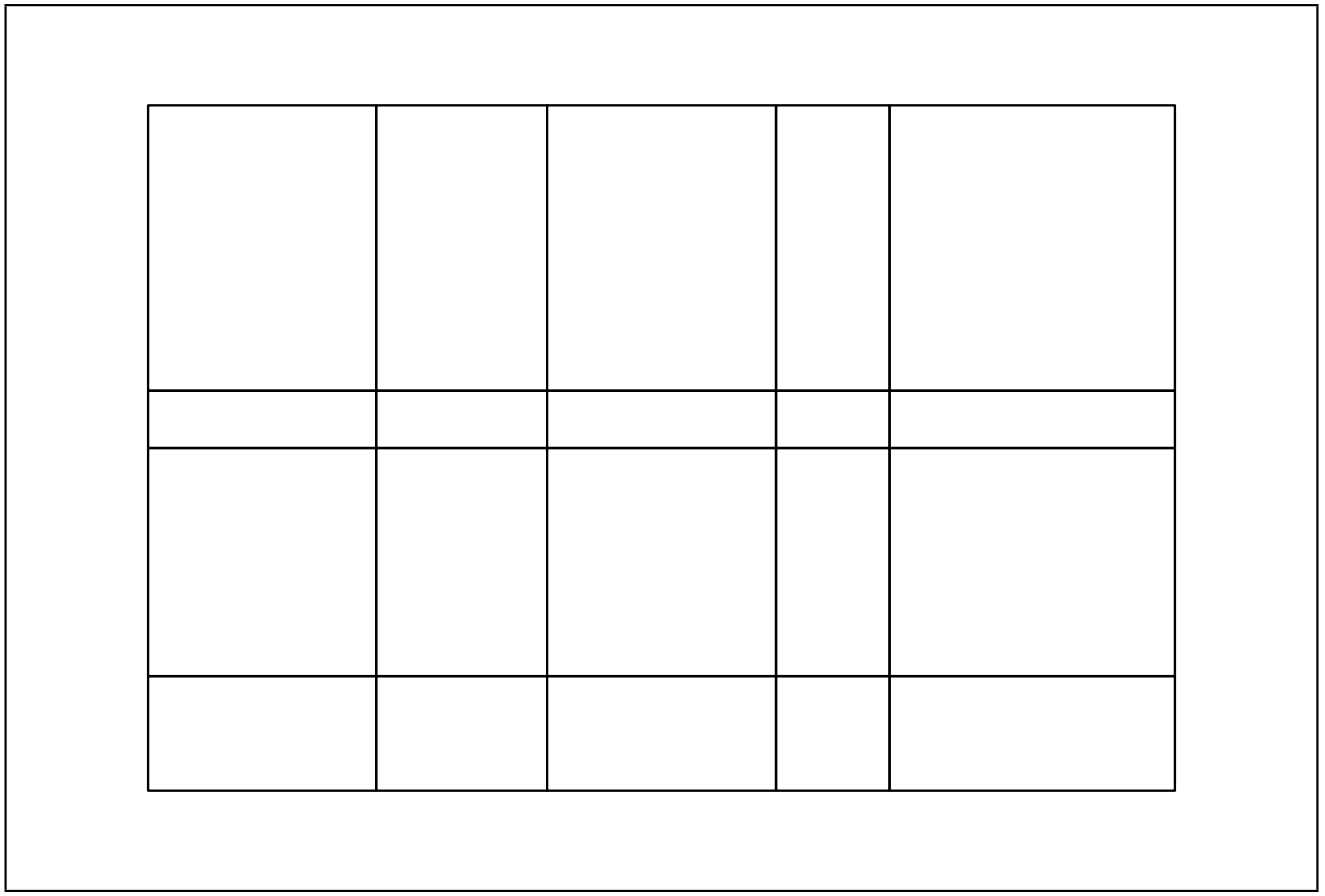


Arranging images in R

- It can be useful to arrange images using R graphics facilities.
  - For example, arrange images using **grid** viewports and layouts.

```
> imageLayout <-
      grid.layout(4, 5,
                  widths=c(2, 1.5, 2, 1, 2.5),
                  heights=c(2.5, .5, 2, 1),
                  respect=TRUE)
```

Arranging images in R



Arranging images in R

- Holiday snaps.





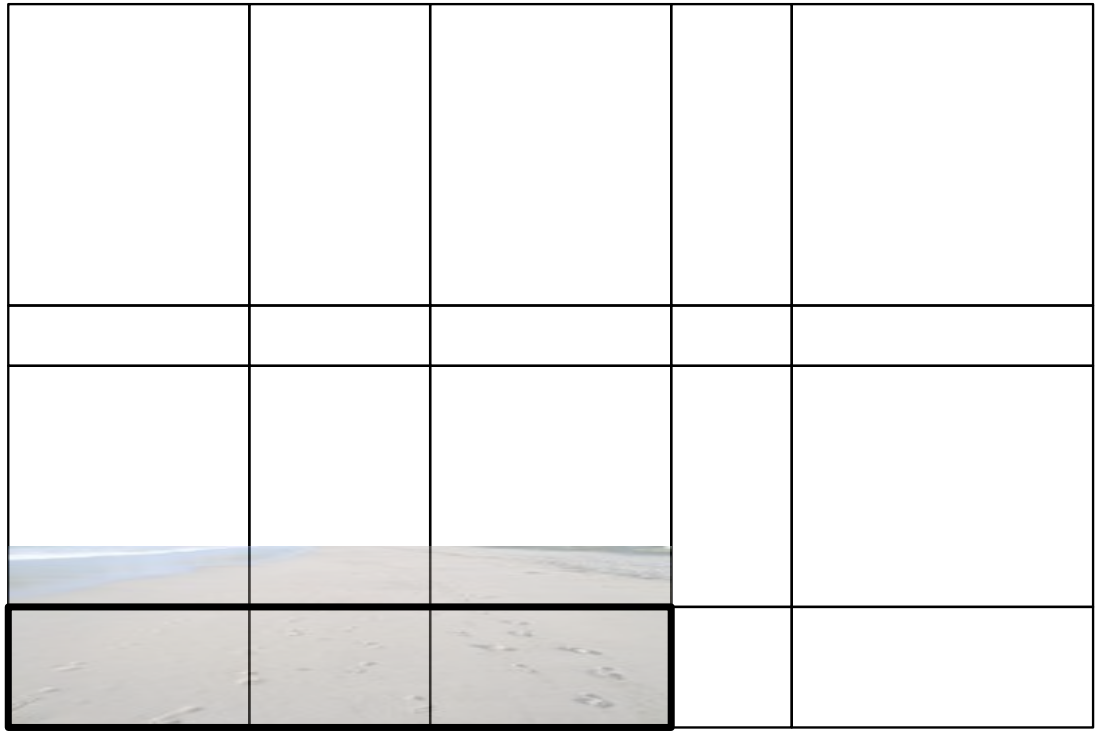


Arranging images in R

```
> pushViewport(viewport(layout.pos.col=1,
                        layout.pos.row=1))
> grid.raster(treeMatrix, width=2)
```

Arranging images in R

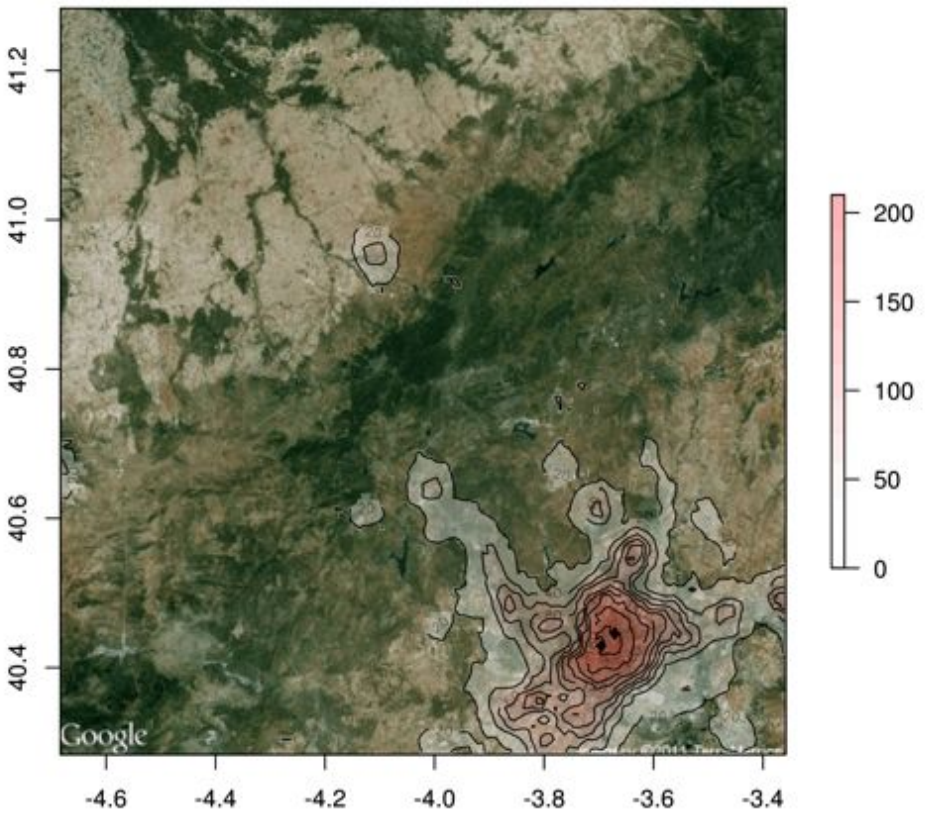
```
> pushViewport(viewport(layout.pos.col=1:3,
                        layout.pos.row=4))
> grid.raster(sandMatrix, width=1, height=1.5,
              y=0, just="bottom")
```



Arranging images in R

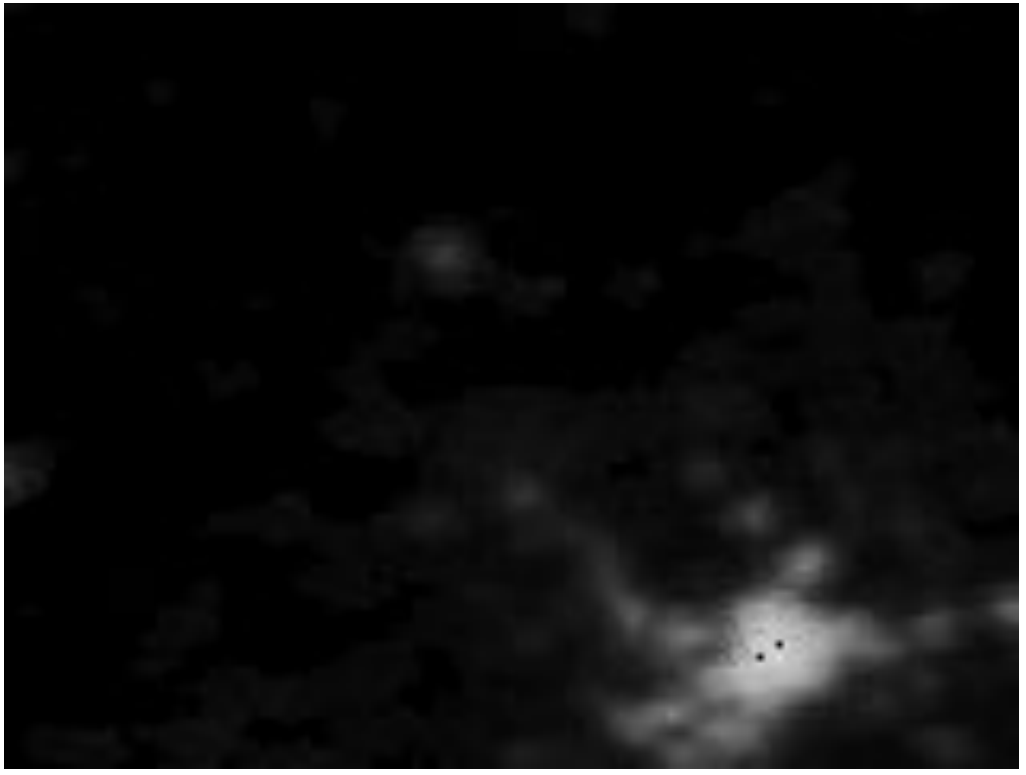
Arranging images in plots

- Again, there are more serious applications.



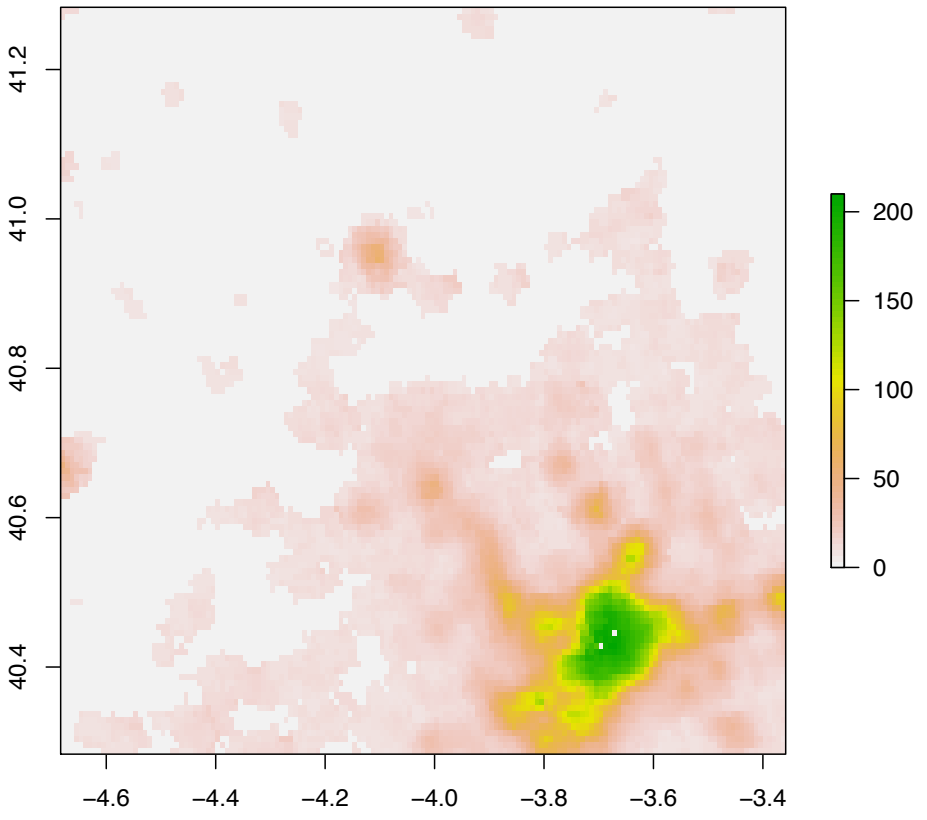
Arranging images in plots

- We want to combine this image (light source data from NASA) ...



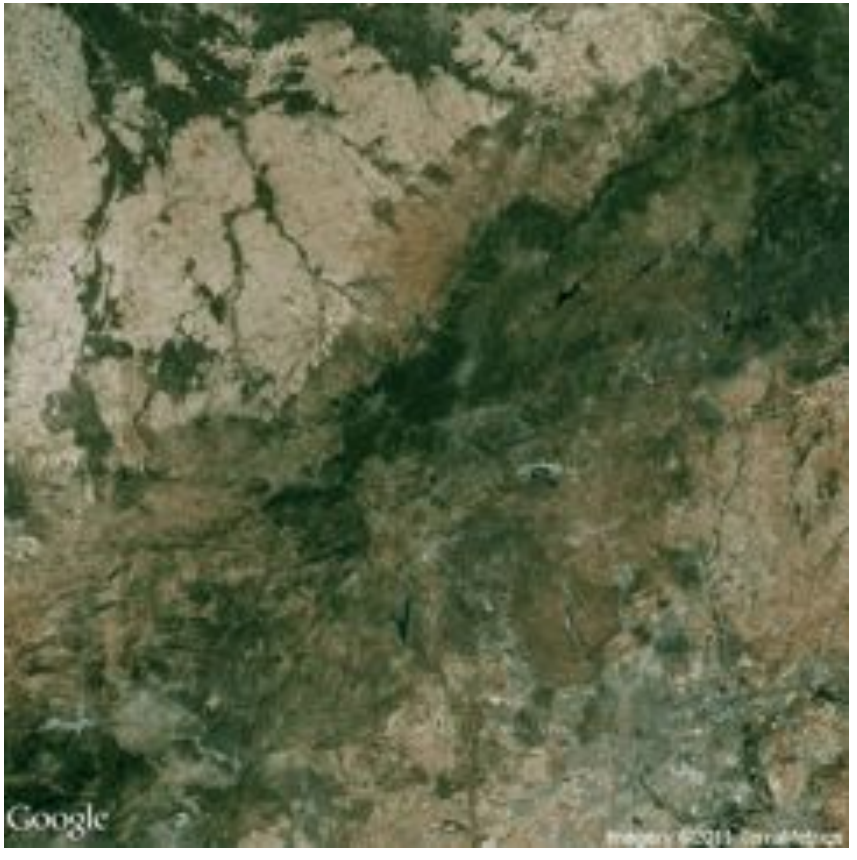
Arranging images in plots

- ... as plotted by R (the **raster** package) ...



Arranging images in plots

- ... with this image (Google Map tile).



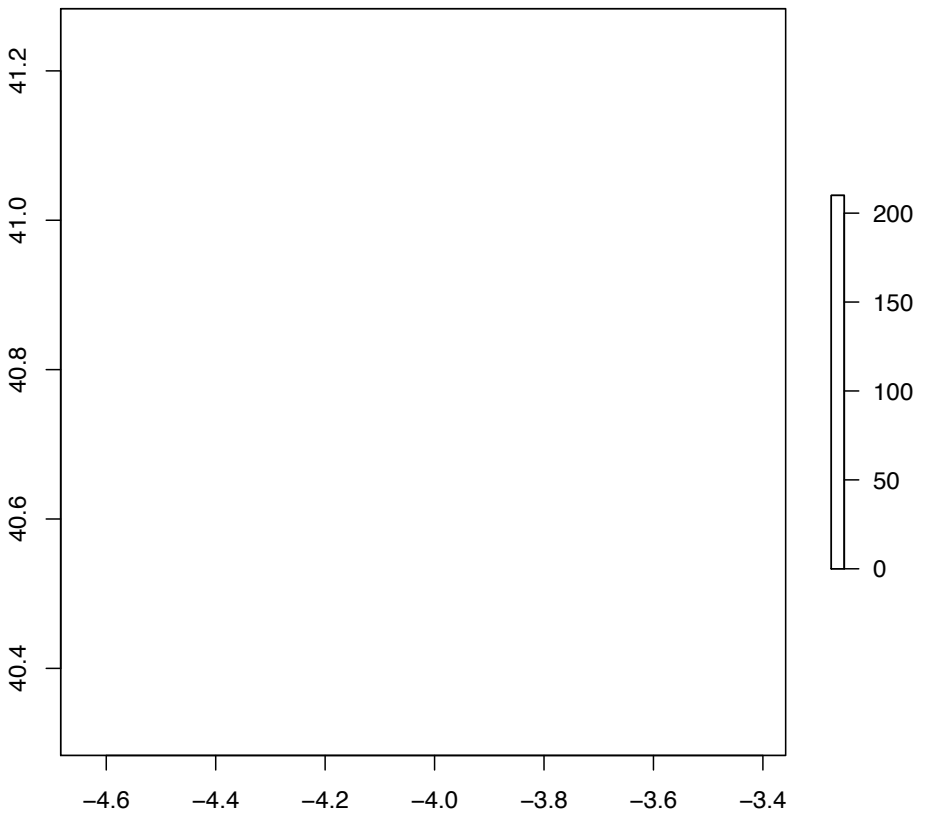
Arranging images in plots

- The important bit is getting the coordinate systems lined up.

```
> library(raster)
> light <- raster("light.tif")

> plot(light, maxpixels=(640*640),
       col="transparent")
```

Arranging images in plots



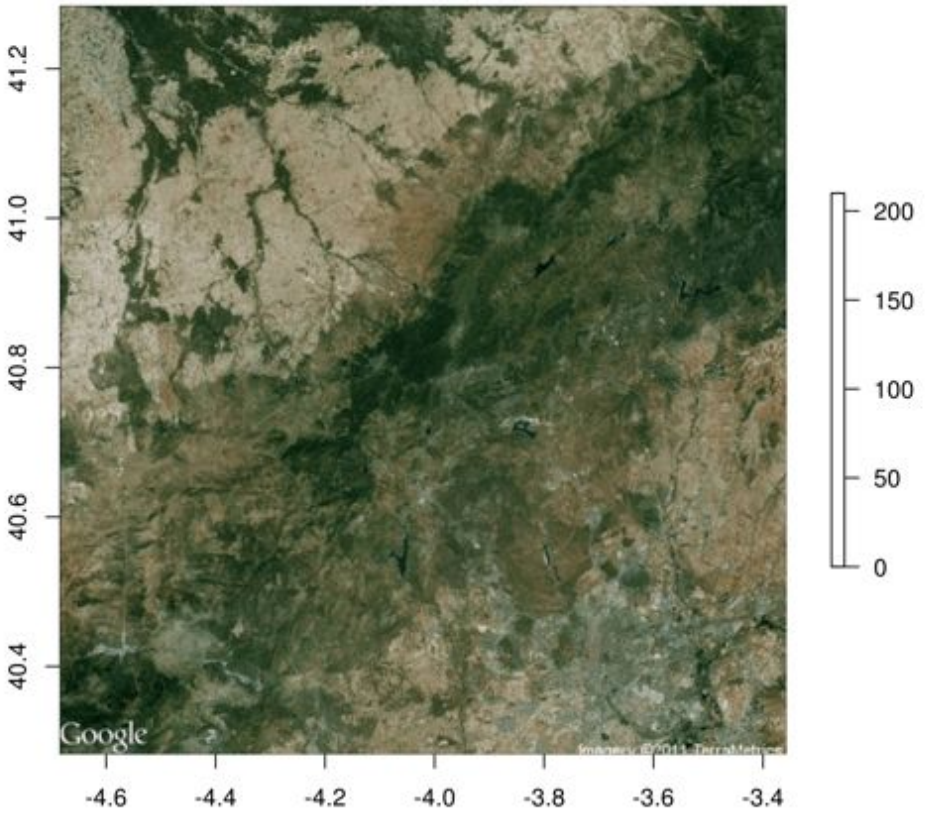
Arranging images in plots

- The important bit is getting the coordinate systems lined up.

```
> gmap <- GetMap(c(40.78, -4.02),
                 zoom=9, size=c(485, 485),
                 maptype="satellite", format="png32",
                 destfile="gmaptile.png")
> gmaptile <- readPNG("gmaptile.png")
> gmaprange <- XY2LatLon(gmap,
                         c(-485/2, 485/2),
                         c(-485/2, 485/2))

> rasterImage(gmaptile,
              gmaprange[1, 2], gmaprange[1, 1],
              gmaprange[2, 2], gmaprange[2, 1])
```

Arranging images in plots

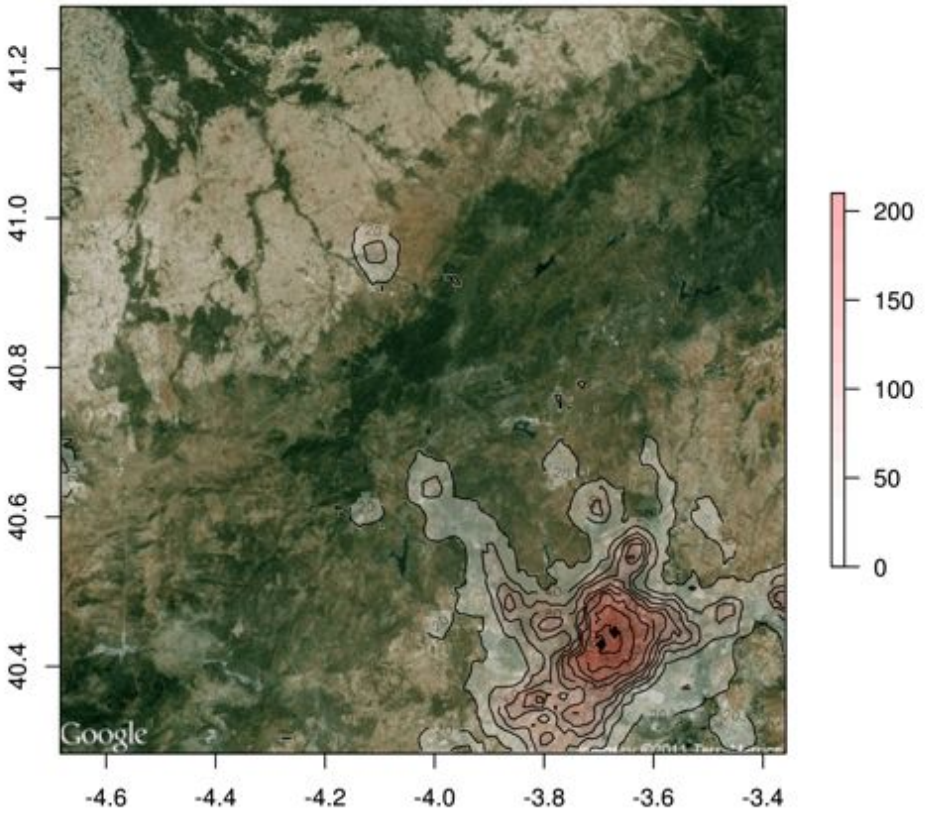


Arranging images in plots

- The final result requires the original plot to be redrawn. Contour lines are also added.

```
> par(new=TRUE)
> colorPalette <-
      colorRampPalette(c("white", "red"))
> plot(light, maxpixels=(640*640),
       col=c("transparent",
             adjustcolor(colorPalette(10),
                         alpha.f=.3)))
> contour(light, add=TRUE)
```

Arranging images in plots



Summary

- Native support for raster images leads to faster drawing and smaller graphics files.
- We can start to think about R graphics as image manipulation software, which leads to things like infographics.
- Can use R graphics features to arrange graphical images, which is particularly useful for incorporating raster images within plots.

Acknowledgements

- The opening raster image is from an open access article at Biomed Central (provided by Wolfgang Huber).
- The original blog post was by Morgan Clendaniel `http://www.good.is/post/infographic-where-did-the-money-to-rebuild-iraq-go/`
- The background image used in the remake is "After the Bombs" by Adam Henning `http://www.flickr.com/photos/adamhenning/66822173/` CC BY-NC-SA
- The network icons are from Cisco `http://www.cisco.com/web/about/ac50/ac47/2.html` "You may use them freely, but you may not alter them."
- The face animation is by and of Mark Holmes.
- The NASA light data is from Steve Mosher.
- The Google Map tile is copyright 2011 Google, Map Data and copyright 2011 Tele Atlas (valid for use in blogs).