Vector and bitmap images

Workshop on Data Visualization in R

Markus Ringnér

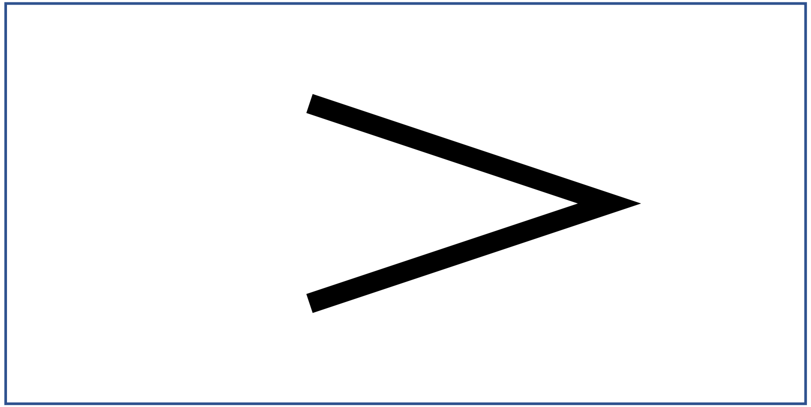
Outline

- Once you have designed and produced your plot, how do you make sure it can be used in publication quality figures?
- There will be some R and ggplot at the end.

What is an image?

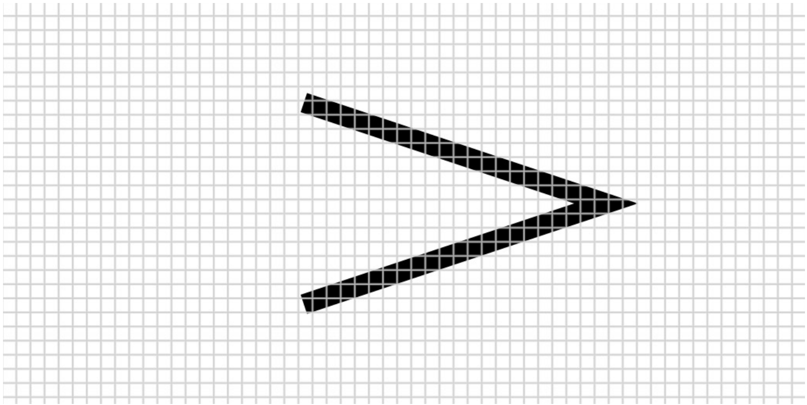
How are images stored on computers?

This is an image!



Now it is art!

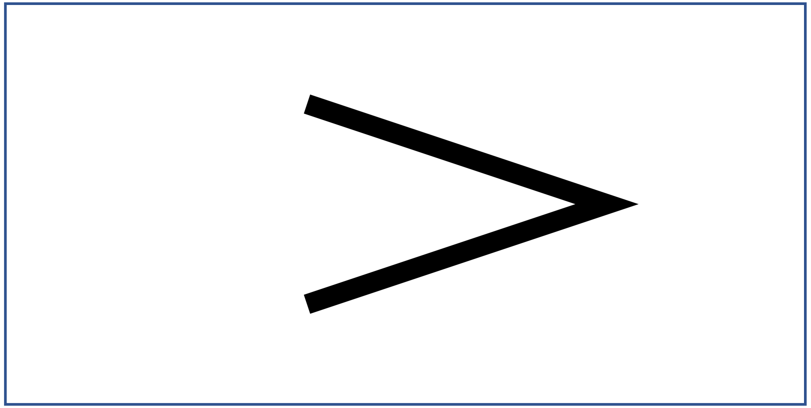
Bitmap (or raster) image



- File size depends on resolution
- Number of pixels * 1 bit (black/white).

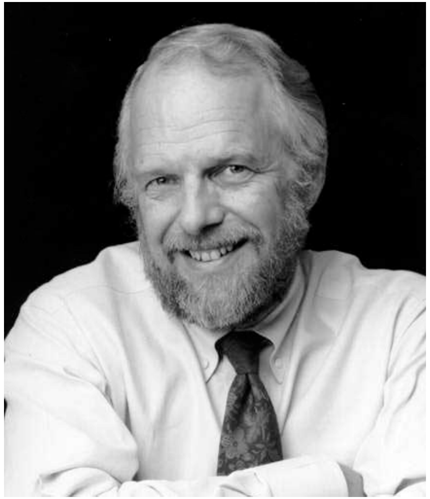
Vector image in (encapsulated) postscript

ps_example1.eps



```
%!PS
%%Creator: Markus Ringnér
%%BoundingBox: 0 0 400 200
%%End Comments
newpath
150 50 moveto
300 100 lineto
150 150 lineto
10 setlinewidth
stroke
```

John Warnock



- A Hidden Surface Algorithm for Computer Generated Half-Tone Pictures (1969).
- Inventor of postscript, pdf, …

Key idea was to describe all of the content of pages for printing not as collections of spots, but at a much more abstract level – as geometry.

File sizes of images

```
$ wc ps_example1.eps
      10      24     149 ps_example1.eps
```

- Vector image file size: 149 characters.
- Image size (bounding box): 400 * 200 points. 1 pt = 1/72 inches.
- Bitmap image at 300 dpi gives:
  (300*400/72)*(300*200/72) = 1,388,889 pixels
- 32-bit tiff with no compression:
  1388889*32/(8*1024*1024) = 5.3 Megabyte

```
$ ls -lh ps_example1.*
-rw-r--r--  1 markus  staff   149B Nov  2 16:41 ps_example1.eps
-rw-r--r--@ 1 markus  staff   5.3M Nov  2 17:20 ps_example1.tiff
```

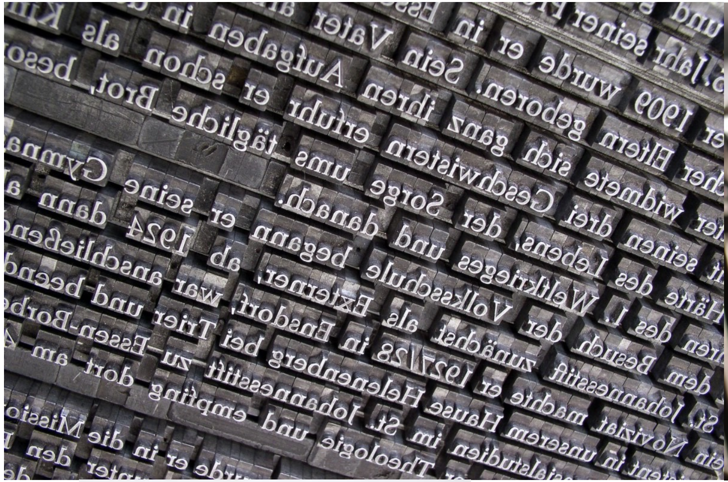
Lossless compression of bitmap image

```
$ ls -lh ps_example1.*
-rw-r--r--  1 markus  staff   149B Nov  2 16:41 ps_example1.eps
-rw-r--r--@ 1 markus  staff   5.3M Nov  2 17:20 ps_example1.tiff
-rw-r--r--@ 1 markus  staff    33K Nov  2 17:21 ps_example1.png
```

- 5.3M / 149B = 5.3*1024*1024/149 ≈ 37000
- 33K / 149B = 33*1024/149 ≈ 200

```
$ du -h ps_example1.*
4.0K ps_example1.eps
5.3M ps_example1.tiff
 36K ps_example1.png
```

Fonts (bitmap)

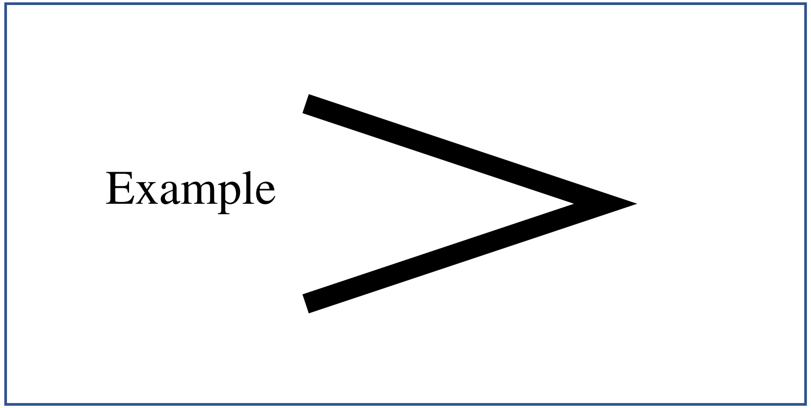


Before the 1990s there were typicallt only bitmap fonts on computers and printers; raster images of glyphs only available in certain optimized sizes (Axis).

Fonts (scalable)

```
%!PS
%%Creator: Markus Ringnér
%%BoundingBox: 0 0 400 200
%%End Comments
newpath
150 50 moveto
300 100 lineto
150 150 lineto
10 setlinewidth
stroke

/Times-Roman findfont
24 scalefont
setfont
newpath
50 100 moveto
(Example) show
```



- Special facilities in the PostScript language:
  - Characters from fonts
- Apple LaserWriter (1985 with postscript)
- Can make your own fonts.
  - Programming language, even recursive functions.

Scaling images (bitmap)

Scaling images (vector)

Exa

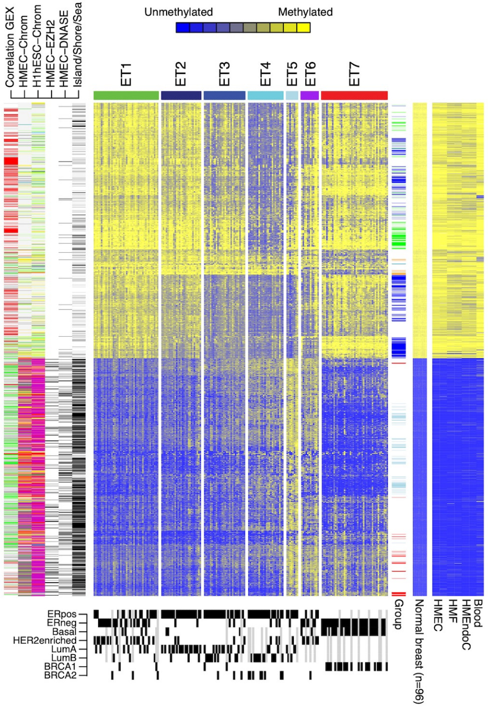
- Also scaling to small sizes. For example gene names in dense plots.

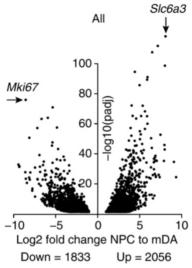
Vector images vs bitmap images

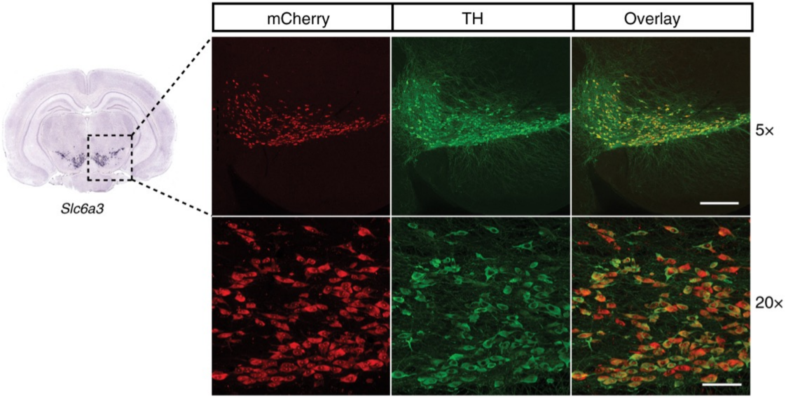
- In terms of file size, vector images are typically much smaller than the corresponding bitmap.
- Vector images are scalable (redrawn to compensate for scale changes). Bitmap graphics are affected by resolution.
- Vector images are simple to edit (Adobe Illustrator, Affinity Designer, …)
- Bitmap fonts can be faster to draw/print (not requiring computer processing).
- Vector graphics are not suited for photographs.
- Drawing vs Painting.

Painting vs Drawing

Mixing bitmap and vector graphics







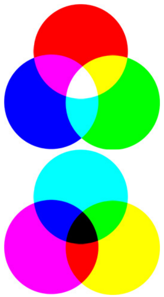
- File size of vector image larger than of corresponding bitmap? “Photograph”?
- Convert to bitmap as late as possible and to the requested size and resolution.
- Have “code” to regenerate your plots for new sizes and resolutions.

File formats

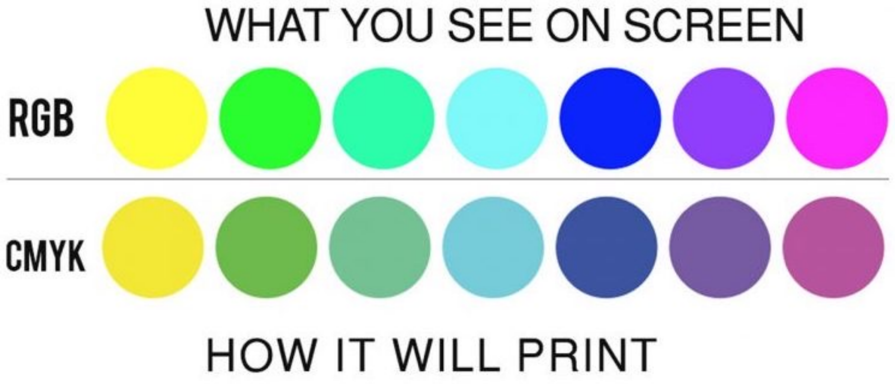
- Vector graphics: pdf, eps, svg, …
  (compound formats)

- Bitmap file formats: jpg, png, tiff, …
  (lossy or lossless data compression)

RGB and CMYK



- Primary colors are arbitrary, but …
- RGB: Red-Green-Blue – additive type of color mode
- CMYK: Cyan-Magenta-Yellow-Black – subtractive type of color mode
- Cyan, magenta, and yellow are lighter than red, green, and blue.



- If you are going to print: CMYK
- If only to be seen digitally: RGB
- Most modern printers will convert automatically, but …

Generate figures with consistent sizes and font sizes

- Explicitly specify font sizes.
- Explicitly specify the width and height of the plot. Easier in inches/millimeters than pixels + dpi.
- Export the plot with ggsave, and not by copying or exporting from R/Rstudio window (that can be resized arbitrarily), to ensure a controlled and reproducible process.

- If you print on paper to test, remember to set scale to 100% or similar (not fit to page …).

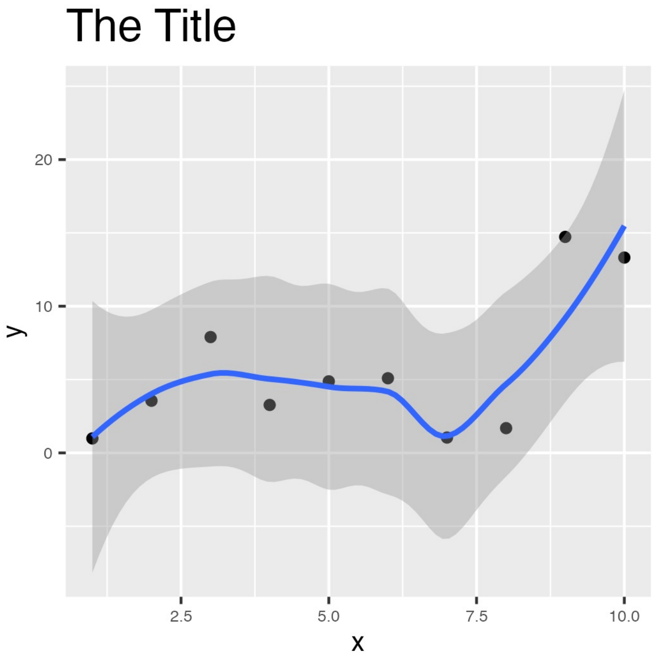
Example plot – 90 mm wide, min font size 6

```
library(ggplot2)

x <- 1:10
y <- x*abs(rnorm(10))

p1 <- ggplot(data.frame(x,y), mapping=aes(x=x,y=y)) +
geom_point() + geom_smooth() + ggtitle("The Title") +
theme(title=element_text(size=14, hjust=0.5),
axis.title=element_text(size=10), axis.text =
element_text(size=6))
```

```
ggsave(filename="gg_example.png",plot=p1, device="png",
height=90, width=90, units="mm", dpi=300)
```




```
ggsave(filename="gg_example.pdf",plot=p1, device="pdf",
height=90, width=90, units="mm", dpi=300)
```

Conclusions

- Hopefully this has provided some helpful initial thoughts on how to produce publication quality figures.

Thank you. Questions?