Graphics with R

How to create and export plots

Dr Simon R. White

MRC Biostatistics Unit, Cambridge

2012

Part I

Case study: Graphics in R

R graphics

- R has an advanced set of tools for generating high quality graphics
- Many additional packages that generate specific types of plots
- Here, we focus on a few examples

Generating clear graphics takes care and effort; it is important to highlight features of complex data

Sometimes a table of figures is more useful

Length over area

As a general rule, people are better at comparing length rather than area

Pie charts are considered a poor graphic choice for statistical information

Unnecessary three-dimensional effects add a confusing dimension to an image that conveys no additional information. Avoid 3D!

General steps

- Remember to read the help files – they often have example code
- There are many tutorials and guides on the internet `addictedtor.free.fr/graphiques/`
- Break the task into smaller parts that you can later combine
- Try to make code re-useable

Managing plots

- The term **device** describes where produced plots are sent
- Devices include the plot window (the default) and files

`dev.new()`

Opens a new device and selects it

`dev.list()`

List all open devices

`dev.cur()`

Returns the name of the current device (a window or file)

`dev.set("device-name")`

Set the active device

`dev.off()`

Closes a device (active device by default)

Saving plots

Telling R to output to a different device – specifically a file

`postscript("filename.ps")`

Readies a file (device) for outputting graphs to

`pdf("filename.pdf")`

Readies a file (device) for outputting graphs to

`png("filename.png")`

Readies a file (device) for outputting graphs to

`jpeg("filename.jpeg")`

Readies a file (device) for outputting graphs to

Remember to close the device with `dev.off()`

Advanced topic

Comparison of file formats

Postscript	Very good quality, accepted by nearly all journals, can be imported into Microsoft Word documents
PDF	Good quality image and widely viewable, can have very large file size
PNG	Not recommended for publication quality images
JPEG	Not recommended for publication quality images

R has no erase or undo

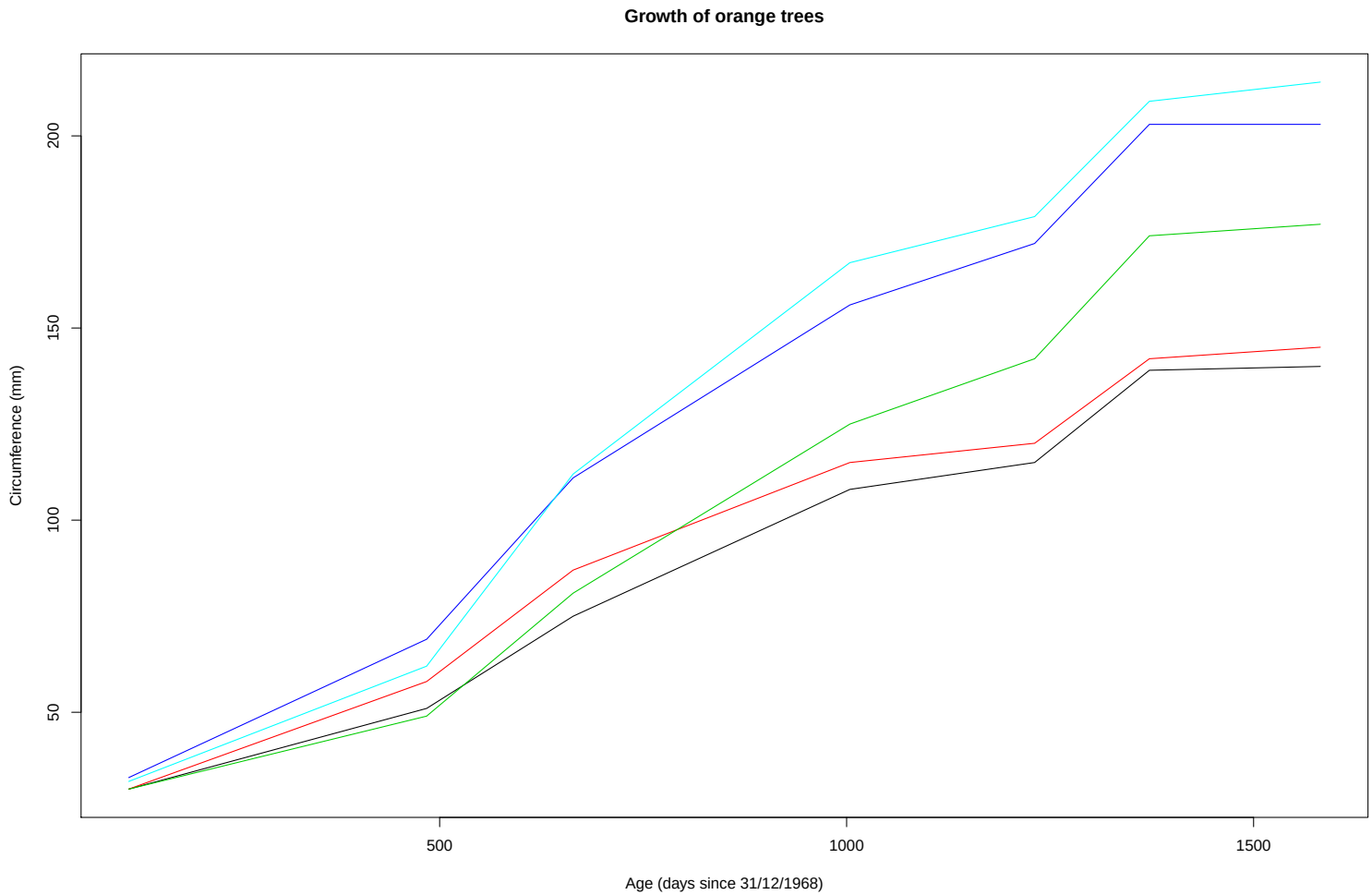
Imagine drawering your plot on a piece of paper in permanent ink

That is how R plotting works. Once you have added something to a plot there is no way to remove it (unless you cover it up with something else)

Thus, it is essential that you use script files so that small changes can be made and the graphs re-plotted with the minimum of effort

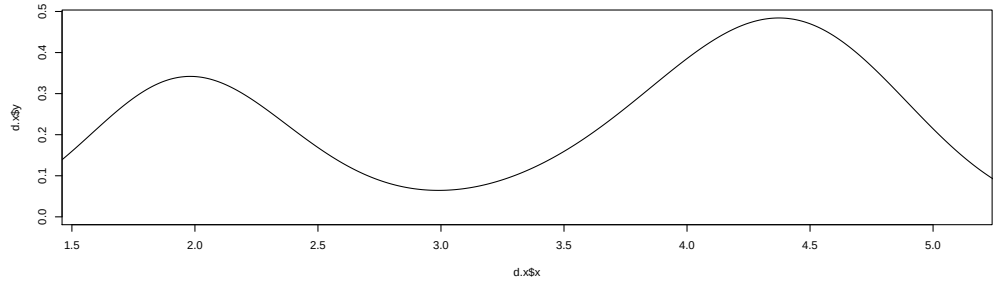
Task 1 recreate this graphic

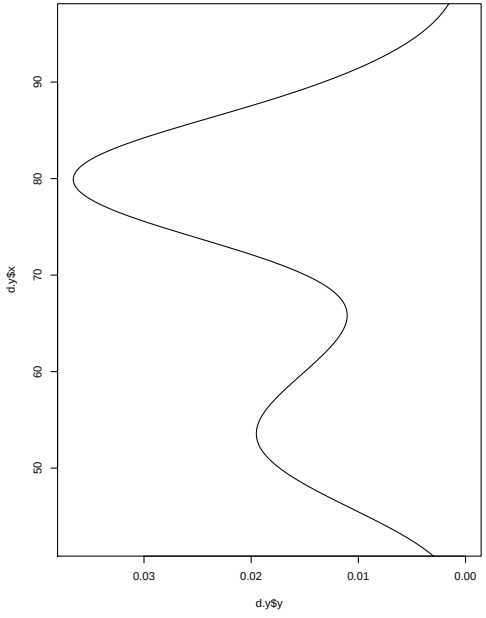
Available using data(Orange)

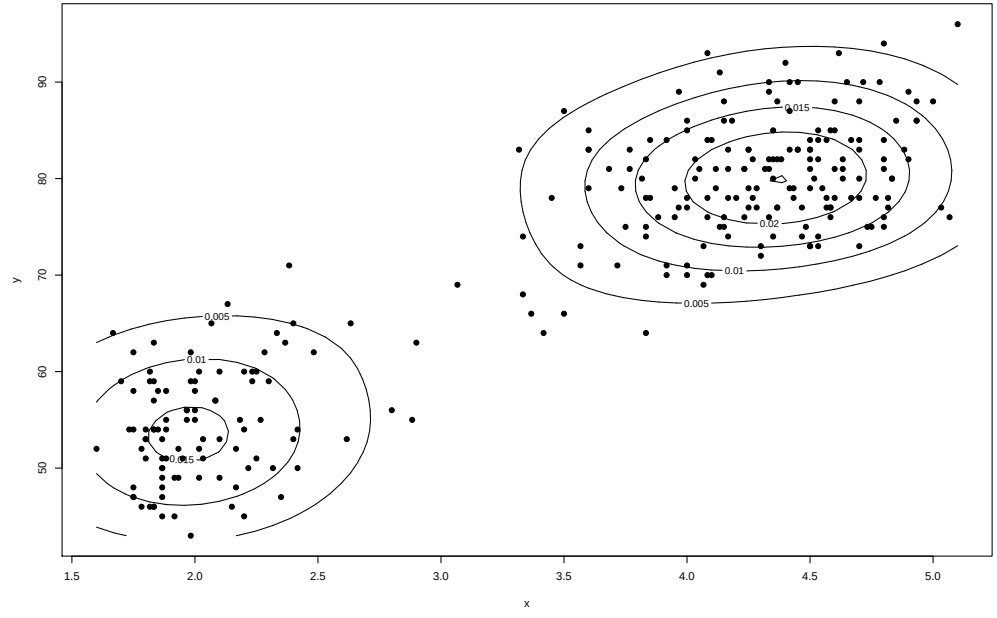


Task 2: recreate this graphic

Available using data(faithful)

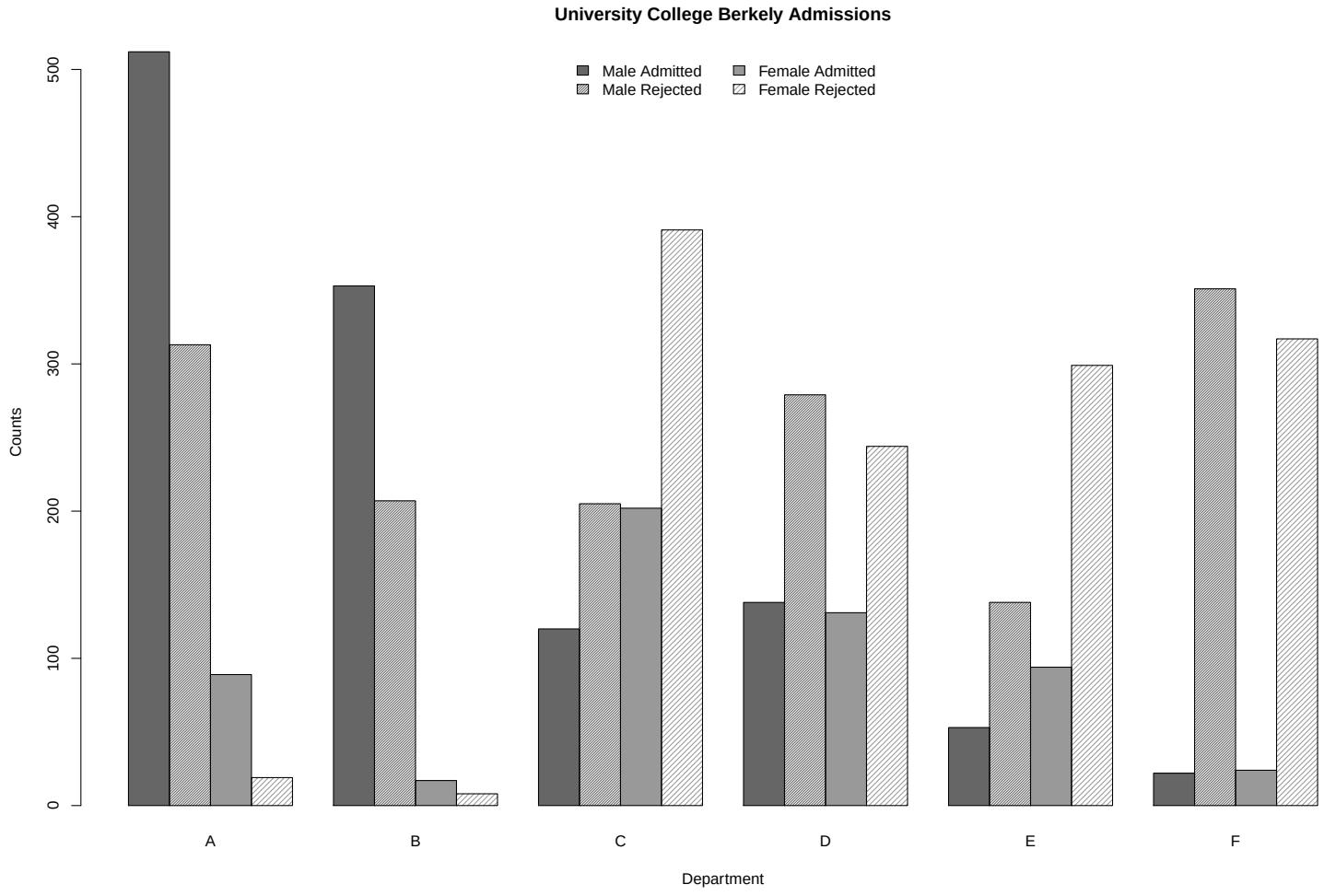






Task 3: experiement with options

Available using library(datasets);data(UCBAdmissions)



Task 1: solution A

```
data(Orange)
pdf("plot_1a.pdf", width=15, height=10)
  Columns <- c("age","circumference")
  plot(0, 0, type="n", xlim=range(Orange$age),
       ylim=range(Orange$circumference),
       main="Growth of orange trees",
       xlab="Age (days since 31/12/1968)",
       ylab="Circumference (mm)" )
  lines( Orange[Orange$Tree=="1",Columns], col=1 )
  lines( Orange[Orange$Tree=="2",Columns], col=2 )
  lines( Orange[Orange$Tree=="3",Columns], col=3 )
  lines( Orange[Orange$Tree=="4",Columns], col=4 )
  lines( Orange[Orange$Tree=="5",Columns], col=5 )
dev.off()
```

Careful when indexing

Ensure you are comparing strings with strings

```
class(Orange$Tree)

Orange$Tree=="5" # correctly comparing string with a string
Orange$Tree==5   # the numeric '5' is sliently converted
Orange$Tree==as.character(5) # the above line actually

levels(Orange$Tree)
levels(Orange$Tree)[3]

Orange$Tree==levels(Orange$Tree)[3]
```

Task 1: solution B

```
data(Orange)
pdf("plot_1b.pdf", width=15, height=10)

  Columns   <- c("age","circumference")
  Num.lines <- length(levels(Orange$Tree))
  Cols      <- c("red","black","blue","green","grey")

  plot(0, 0, type="n", xlim=range(Orange$age),
       ylim=range(Orange$circumference),
       main="Growth of orange trees",
       xlab="Age (days since 31/12/1968)",
       ylab="Circumference (mm)" )

  for( i in 1:Num.lines ) {
    lines( Orange[Orange$Tree==levels(Orange$Tree)[i],Columns],
           col=Cols[i]  )
  }
dev.off()
```

Task 2: solution

```
data(faithful)
library(MASS)
x <- faithful$eruptions
y <- faithful$waiting
pdf("plot_2.pdf", width=15, height=10)
  d.x <- density(x)
  d.y <- density(y)
  den <- kde2d(x,y,n=50)
  layout( matrix( c(0,2,2,1,3,3,1,3,3),ncol=3) )
  plot(d.x$x, d.x$y, xlim=range(x), type="l")
  plot(d.y$y, d.y$x, ylim=range(y),
       xlim=rev(range(d.y$y)), type="l")
  plot(x,y, xlim=range(x), ylim=range(y), pch=19 )
  contour( den, add=TRUE, nlevels=4 )
dev.off()
```

Task 3: solution

```
library(datasets);data(UCBAdmissions)
tab <- ftable(UCBAdmissions, row.vars=c(2,1))
png("plot_3.png",width=1200, height=800)
  names <- apply( rev(expand.grid(rev(attr(tab, "row.vars")))),
                  1, paste, collapse=" ")
  bp.reps <- 2
  bp.title <- "University College Berkely Admissions"
  bp.colours <- c(gray(0.4),gray(0.6))[rep(1:2, each=bp.reps)]
  bp.density <- c(NA,50,NA,25)
  bp <- barplot( tab, beside=TRUE, col=bp.colours,
                 names.arg=attr(tab, "col.vars")[[1]],
                 density=bp.density, ylab="Counts",
                 main=bp.title,
                 xlab=names(attr(tab, "col.vars")) )
  legend( "top", names, fill=bp.colours, density=bp.density,
          bty="n", ncol=2)
dev.off()
```