R Graphics: What's in it for You?

Paul Murrell

The University of Auckland

CSIRO October 2011

Overview

- R can draw plots ...
- ... but so can lots of other software.

Overview

- R can draw plots ...
- ... but so can lots of other software.

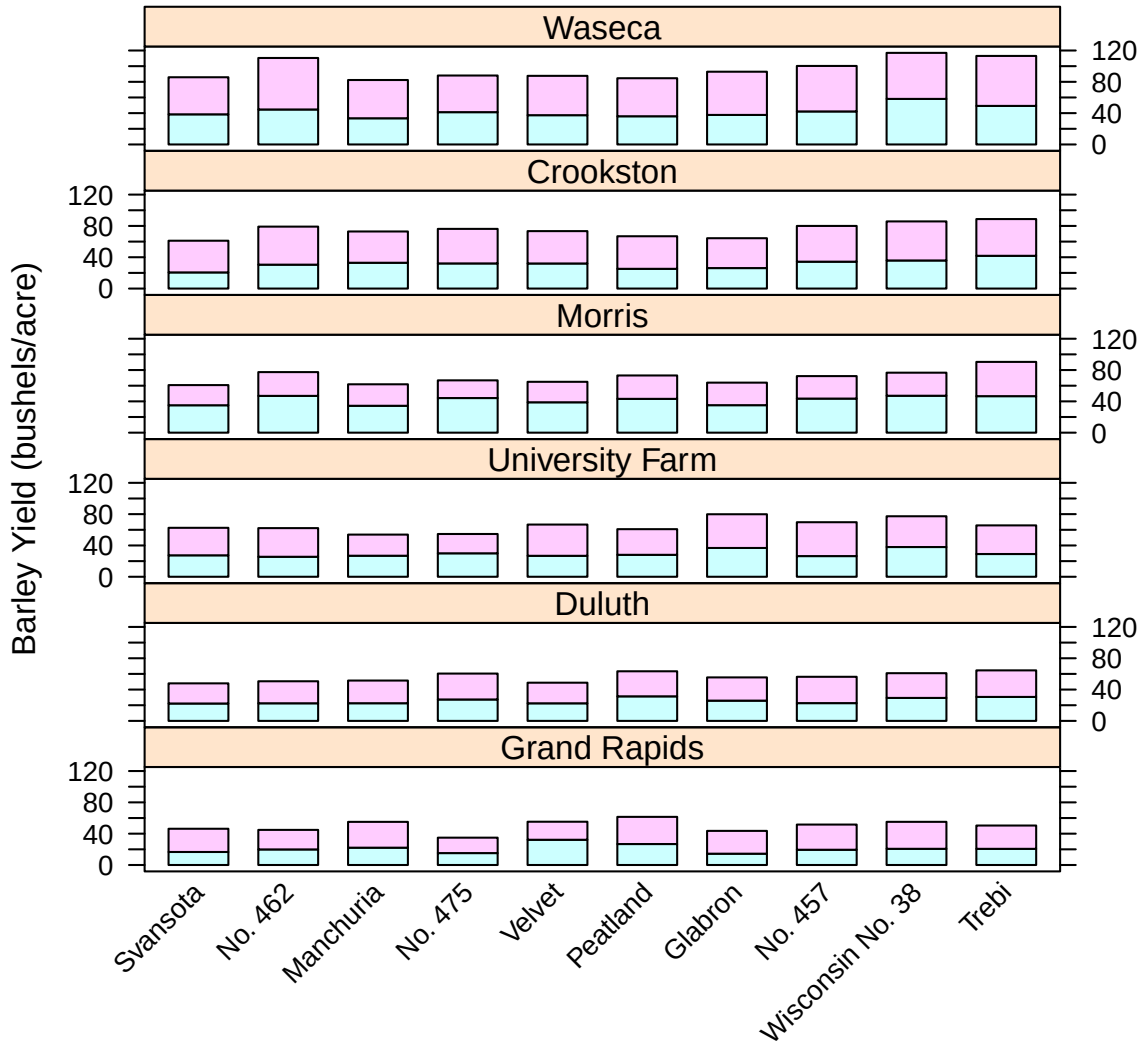
- So why use R?

Overview

- R can draw plots ...
- ... but so can lots of other software.
- So why use R?
- **Customisable, extensible, programmable** graphics

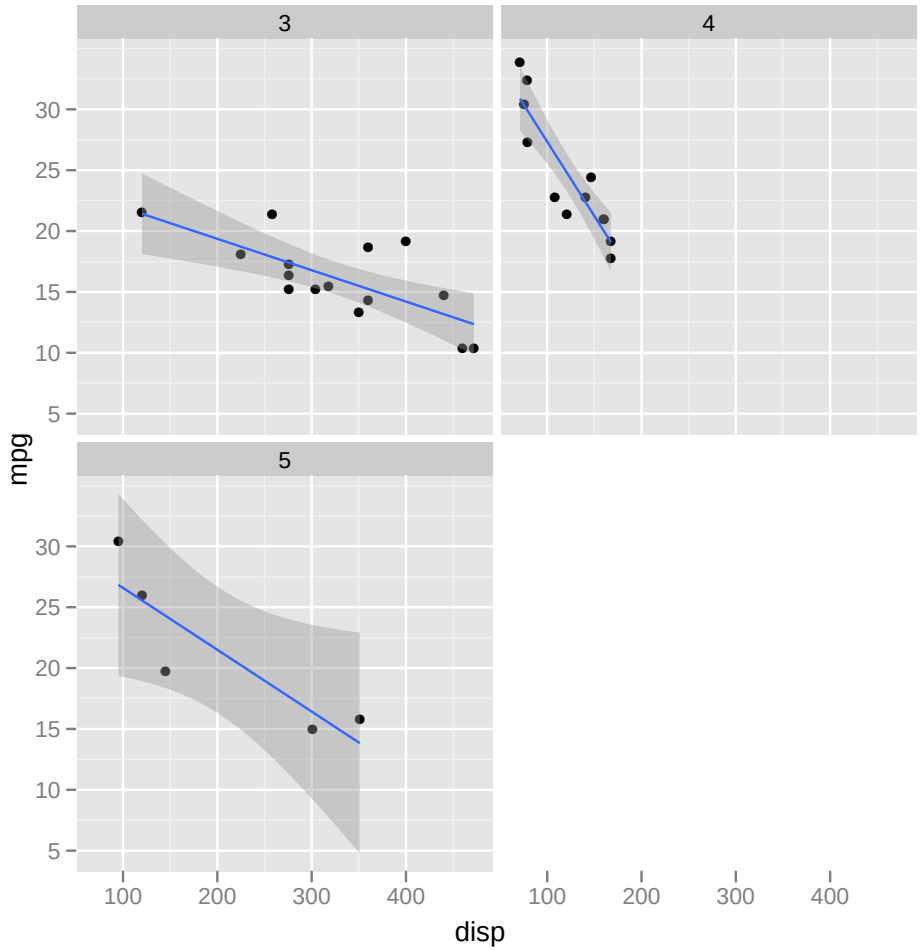
R Plots

- R can draw a wide range of plots.



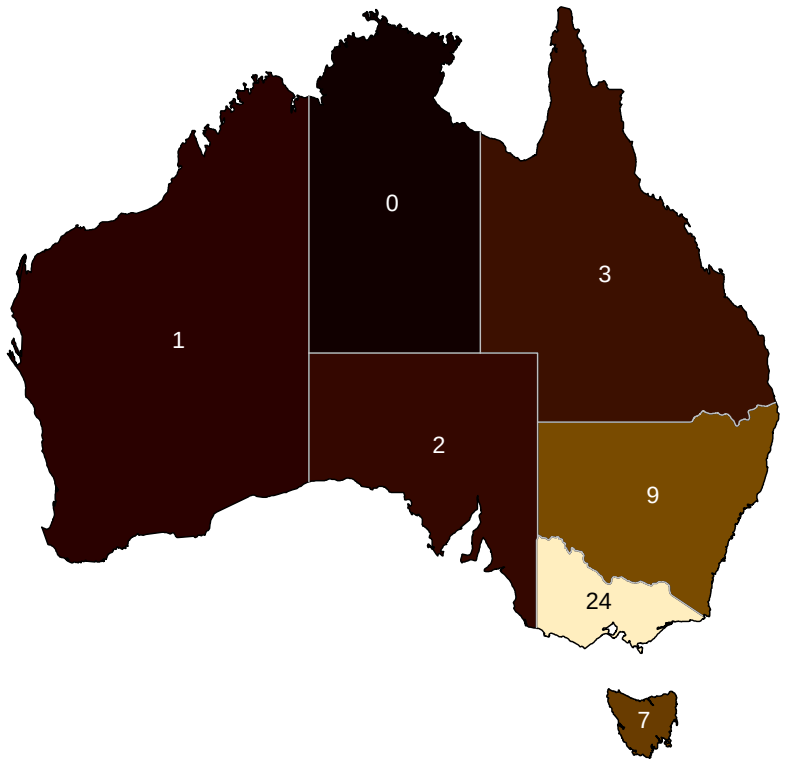
R Plots

- R can draw a wide range of plots.



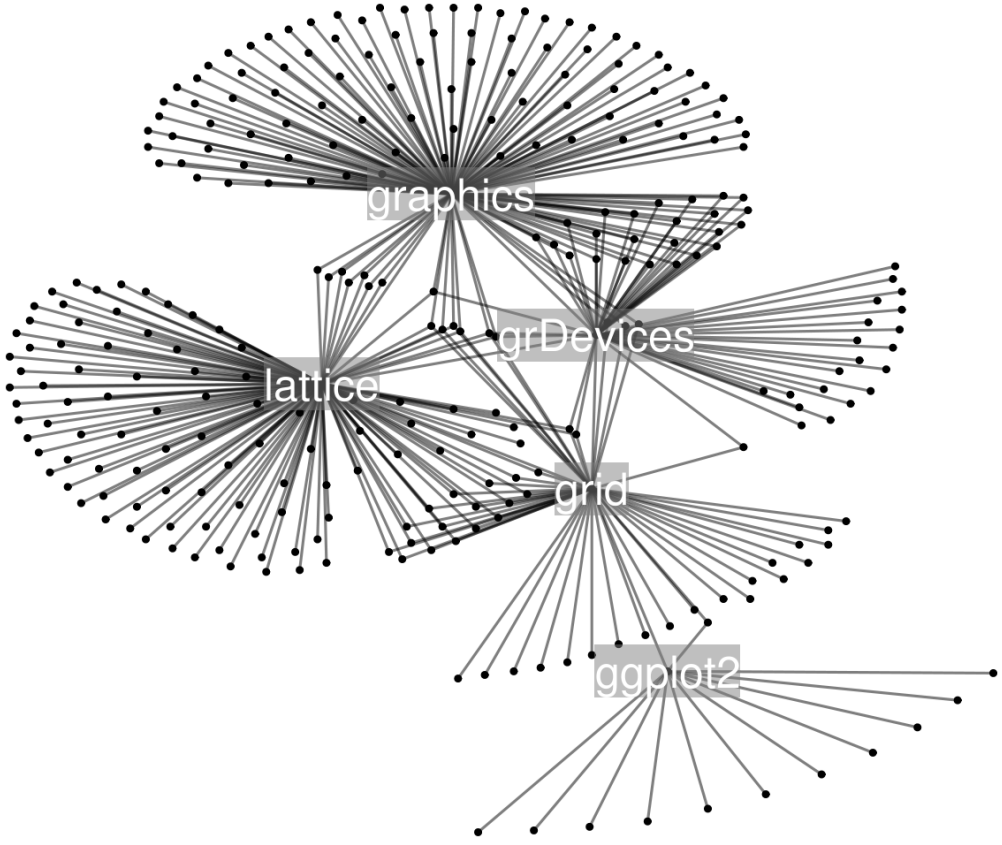
R Plots

- R can draw a wide range of plots.



R Plots

- R can draw a wide range of plots.



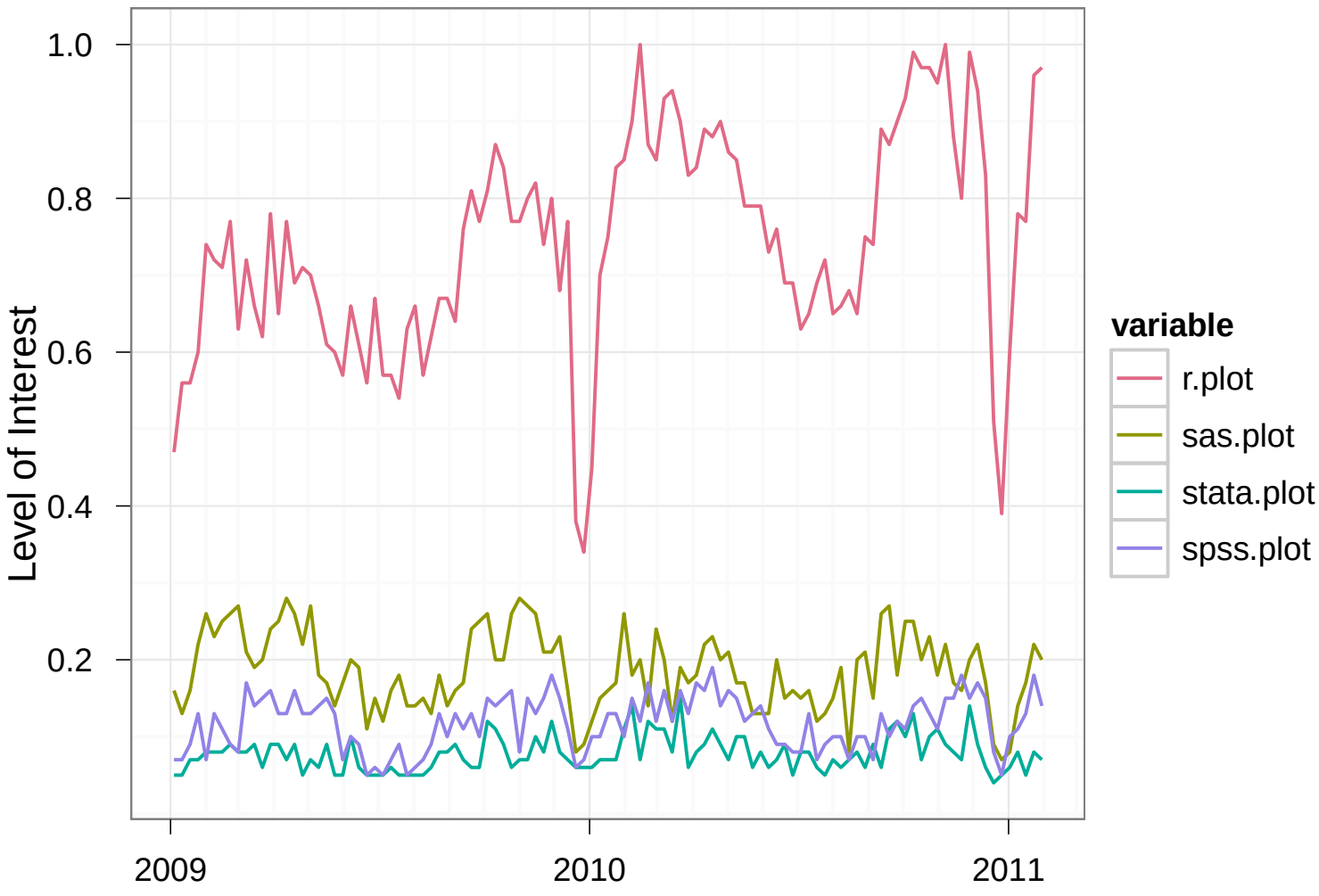
R Plots

- We could argue about which software package has the widest range of plots, but that's just a spitting contest.
- A more interesting question is this:

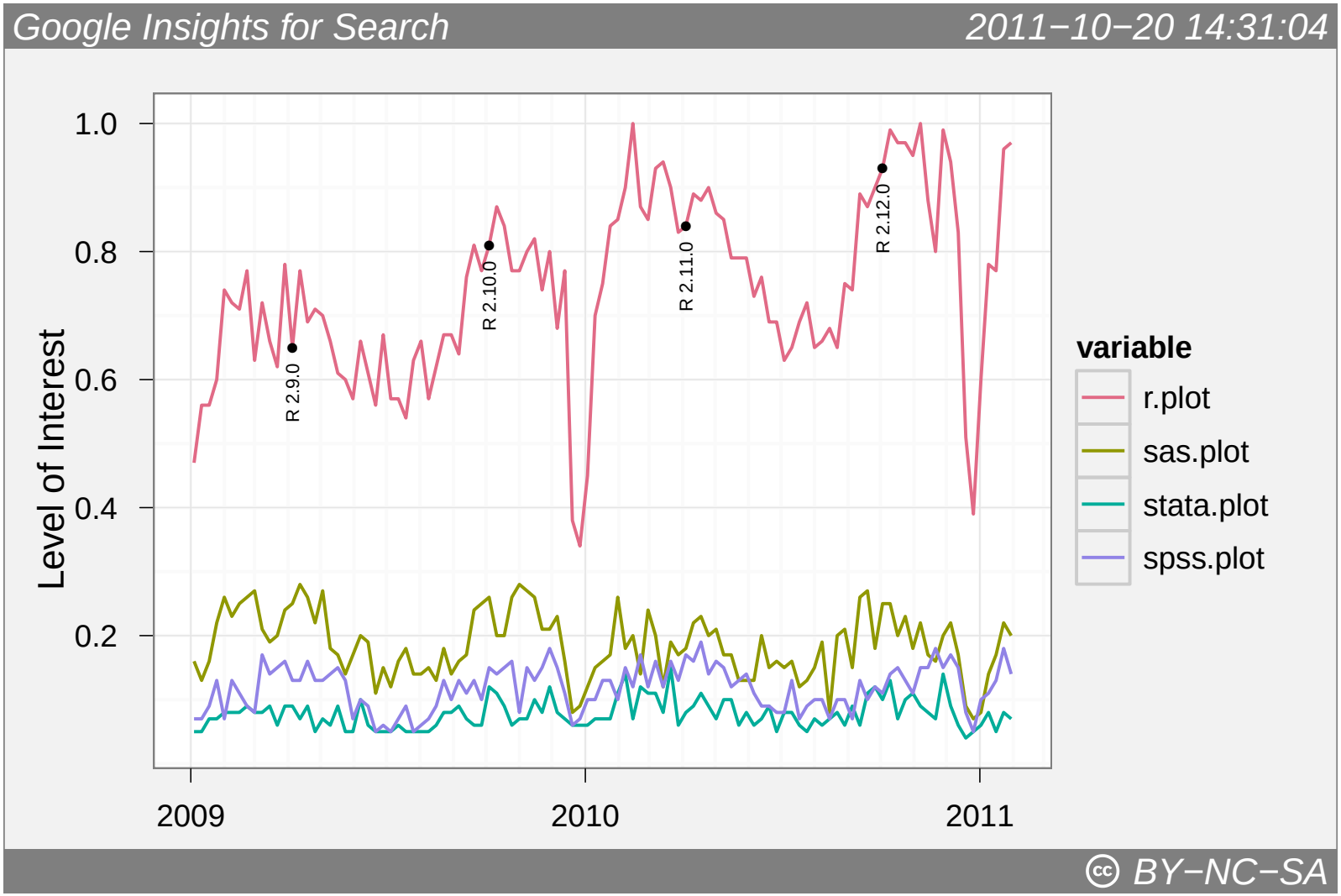
  If the software cannot already create the plot that you want, what are your options?

- One way to characterise R graphics is that it aims to provide you with **lots** of options.
- R graphics is **permissive**.

An Example


Level of Interest
1.0
0.8
0.6
0.4
0.2
2009
2010
2011
variable
r.plot
sas.plot
stata.plot
spss.plot

An Example

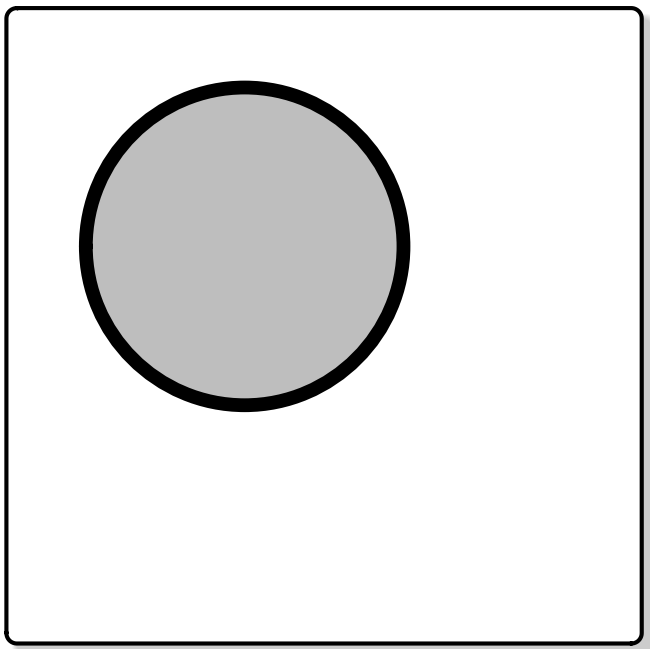

Google Insights for Search
2011−10−20 14:31:04
Level of Interest
1.0
0.8
0.6
0.4
0.2
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot
BY−NC−SA

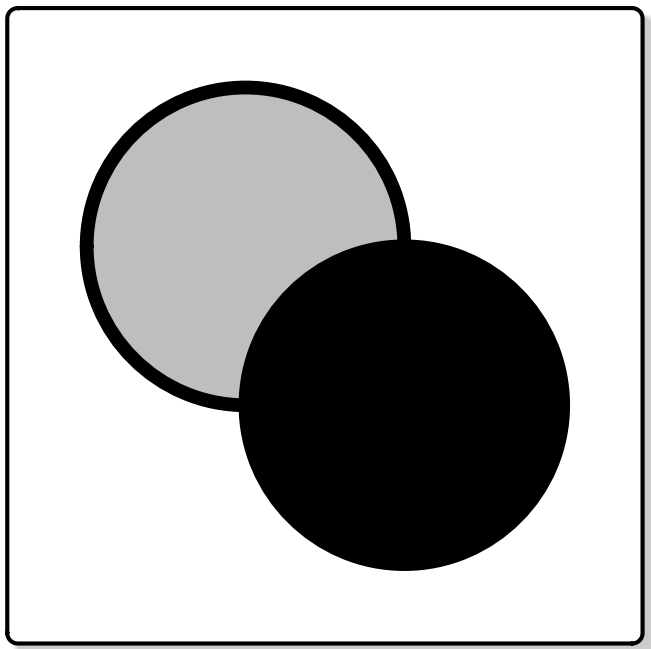
Why Use R?

- What are the features that R graphics provides that make this sort of customisation possible?
- In order to do it (in a simple, yet flexible and sophisticated and rational and reproducible fashion) we need **six** important features:
  1. The ability to write graphics **code**.
  2. The ability to draw simple shapes (**low-level graphics**).
  3. The ability to draw complex shapes (**modern graphics**).
  4. Access to multiple **coordinate systems**.
  5. The ability to arrange graphical components in a **layout**.
  6. The ability to **query** graphical **objects**.

2 Low-level Graphics

- R follows a "painters model" so that you can always add more ink to the page (on top of what is already there).





- A plot is just a whole lot of ink all at once.

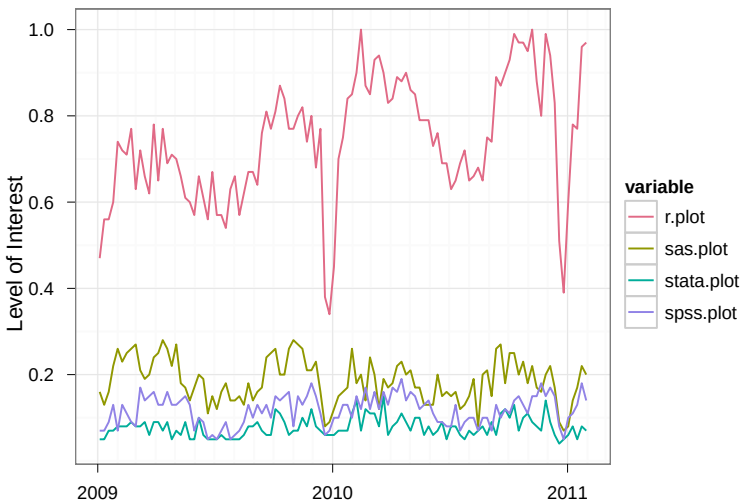
2 Low-level Graphics

- R follows a "painters model" so that you can always add more ink to the page (on top of what is already there).

```
> xyplot( ... )
> grid.text( ... )
```

- A plot is just a starting point.

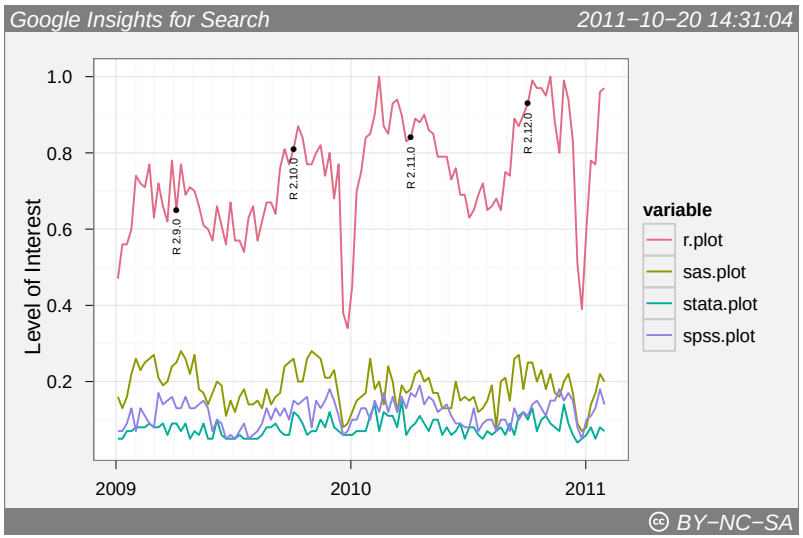
An Example


1.0
0.8
0.6
0.4
0.2
Level of Interest
2009
2010
2011
variable
r.plot
sas.plot
stata.plot
spss.plot


1.0
0.8
0.6
0.4
0.2
Level of Interest
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot

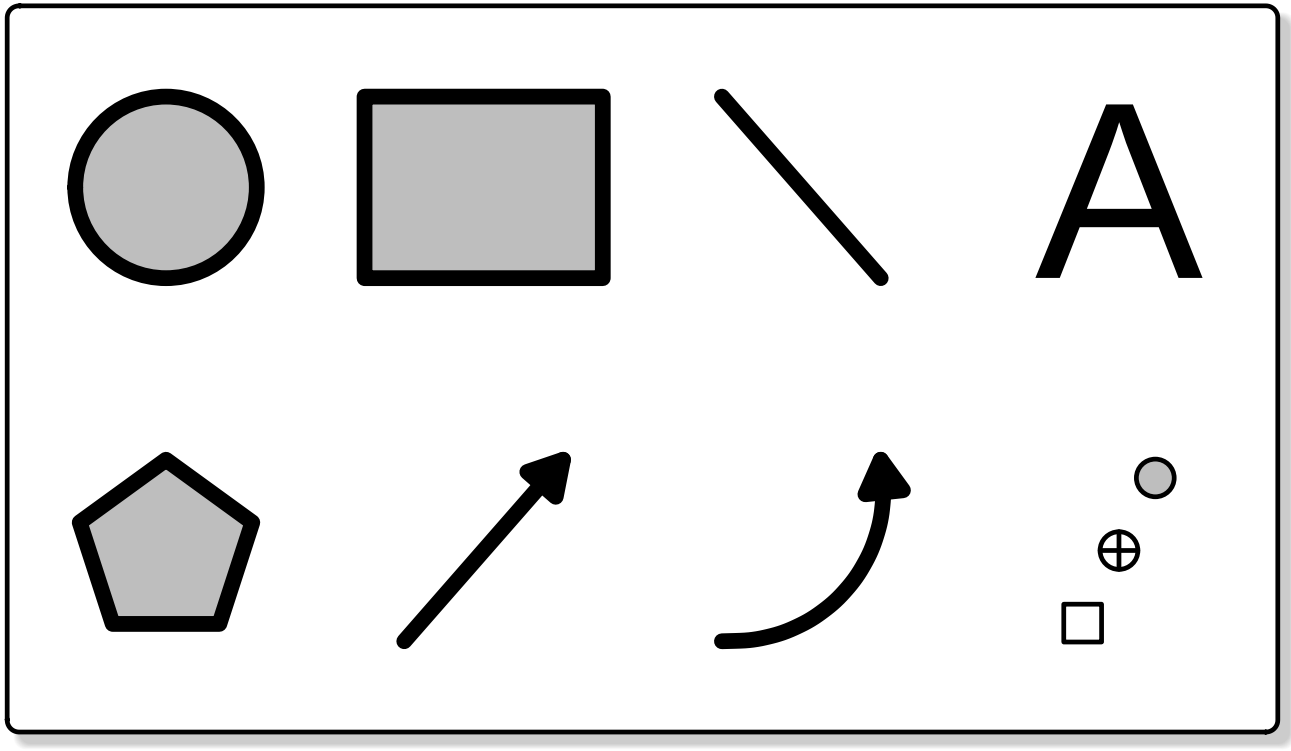
An Example


Google Insights for Search
2011−10−20 14:31:05
BY−NC−SA


Google Insights for Search
2011−10−20 14:31:04
Level of Interest
1.0
0.8
0.6
0.4
0.2
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot
BY−NC−SA

2 Low-level Graphics

- R provides basic drawing primitives.



- A plot is just a whole lot of basic primitives (neatly arranged).

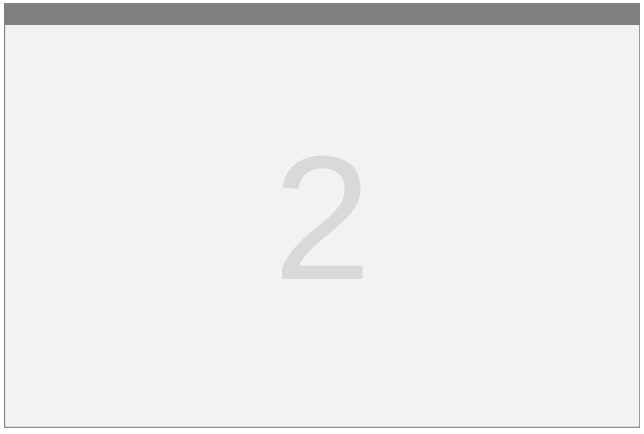
2 Low-level Graphics

- R provides basic drawing primitives.

```
> grid.circle(x = 0.5,
              y = 0.5,
              r = 0.1,
              gp=gpar(col = "black",
                      lwd = 10,
                      fill = "grey"))
```

An Example


1

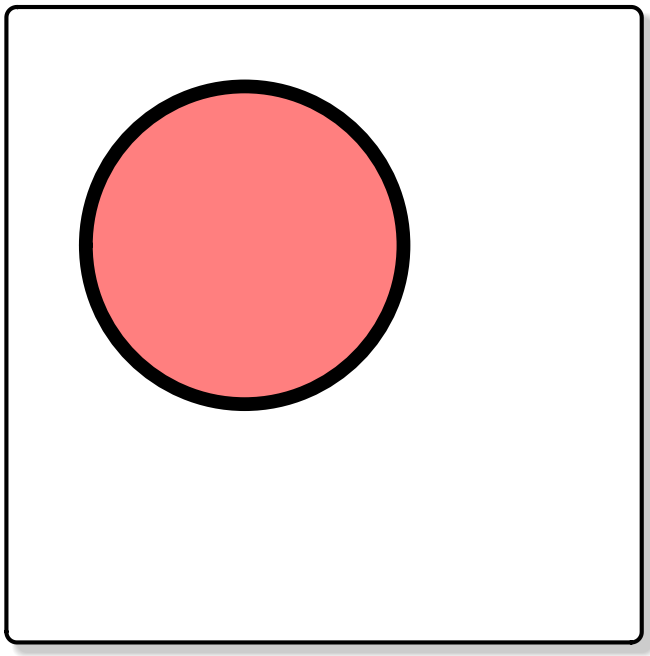

2

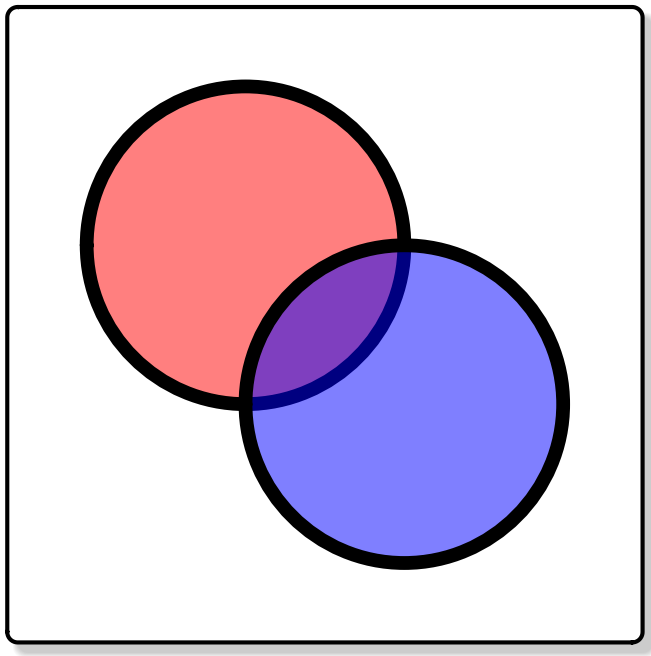

Google Insights for Search
2011−10−20 14:31:05
3


Google Insights for Search
2011−10−20 14:31:05
4
BY−NC−SA

3 Modern Graphics

- R provides features for general graphics (not just for plots) like alpha channels, complex paths, and raster images.





3 Modern Graphics

- R provides features for general graphics (not just for plots) like alpha channels, complex paths, and raster images.

```
> grid.circle(x = 0.5,
              y = 0.6,
              r = 0.2,
              gp=gpar(lwd = 10,
                      rgb(red = 1, green = 0, blue = 0,
                          alpha = 0.5)))
```

3 Modern Graphics

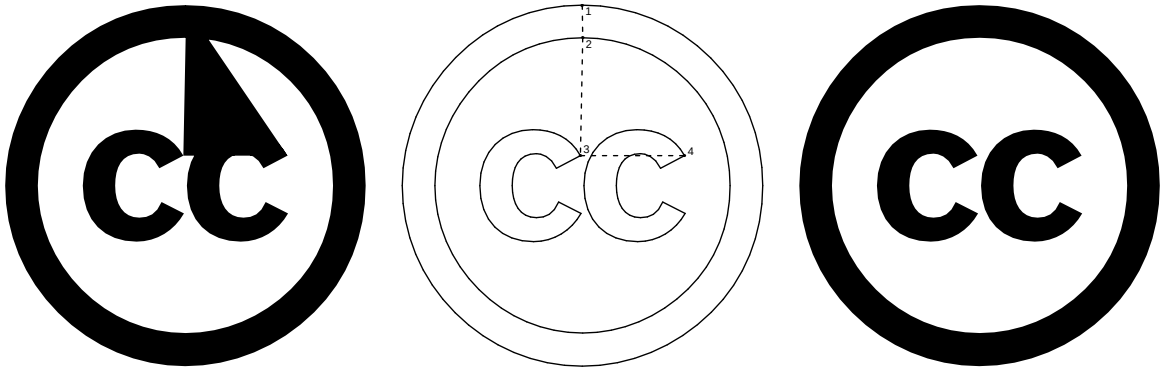
- R provides features for general graphics (not just for plots) like alpha channels, complex paths, and raster images.

3 Modern Graphics

- R provides features for general graphics (not just for plots) like alpha channels, complex paths, and raster images.

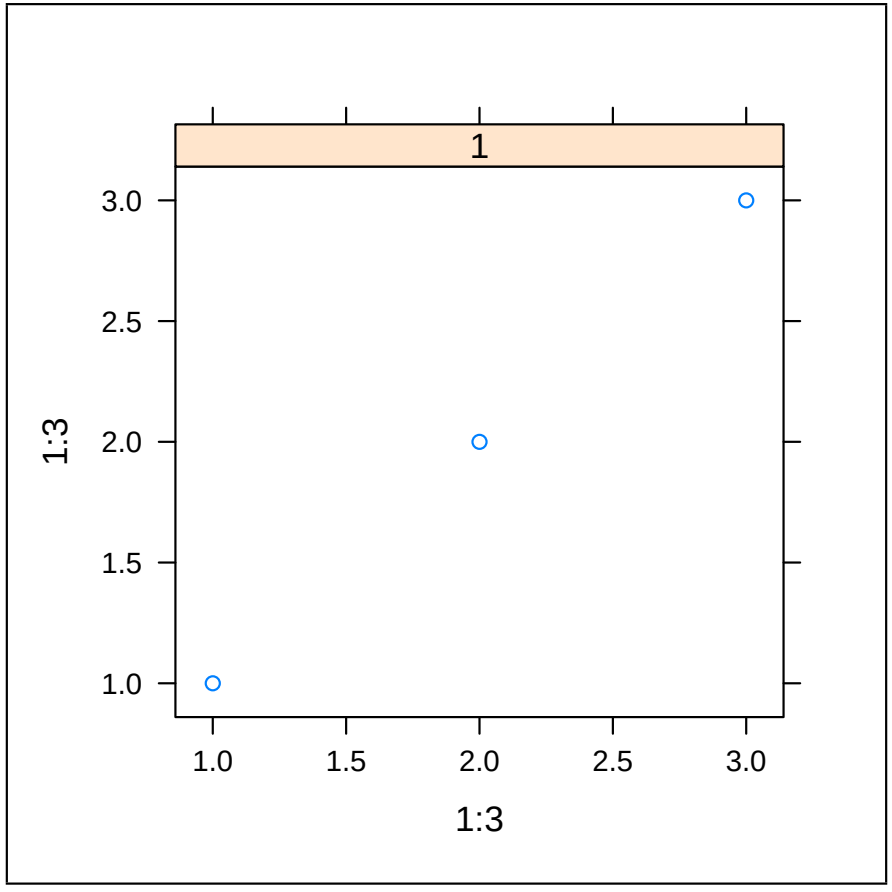
```
> library("png")
> Rlogo <- readPNG(system.file("img", "Rlogo.png",
                                package = "png"))
> grid.raster(Rlogo,
              x = 0.5, y = 0.5,
              width = 0.8)
```

An Example



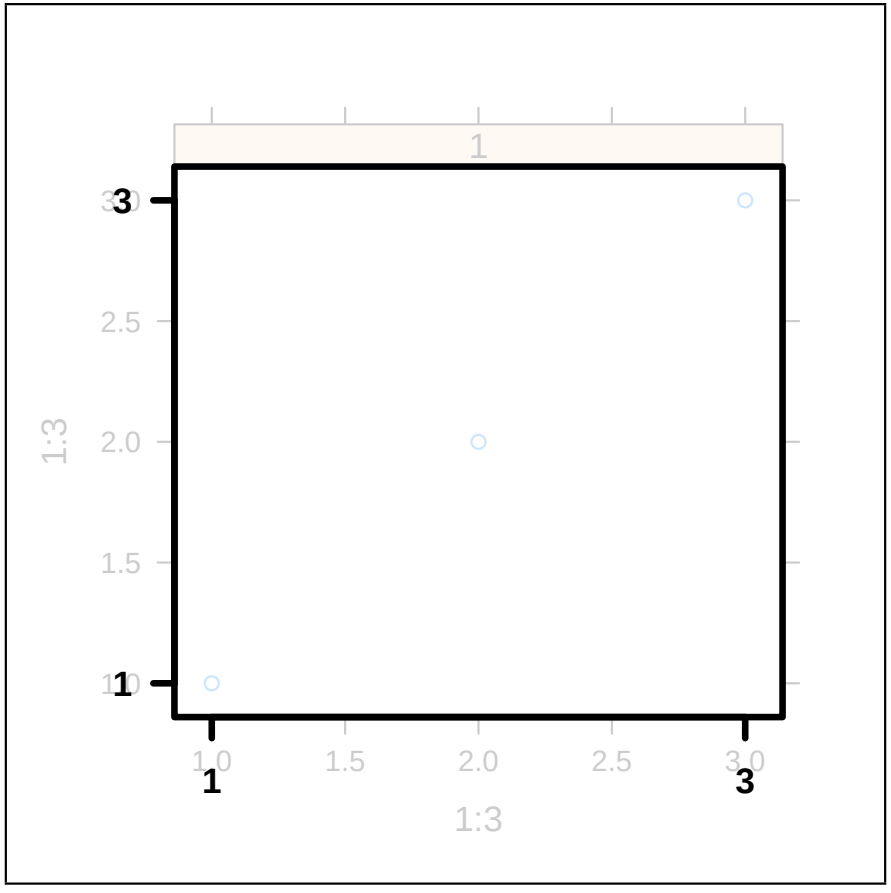
4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.



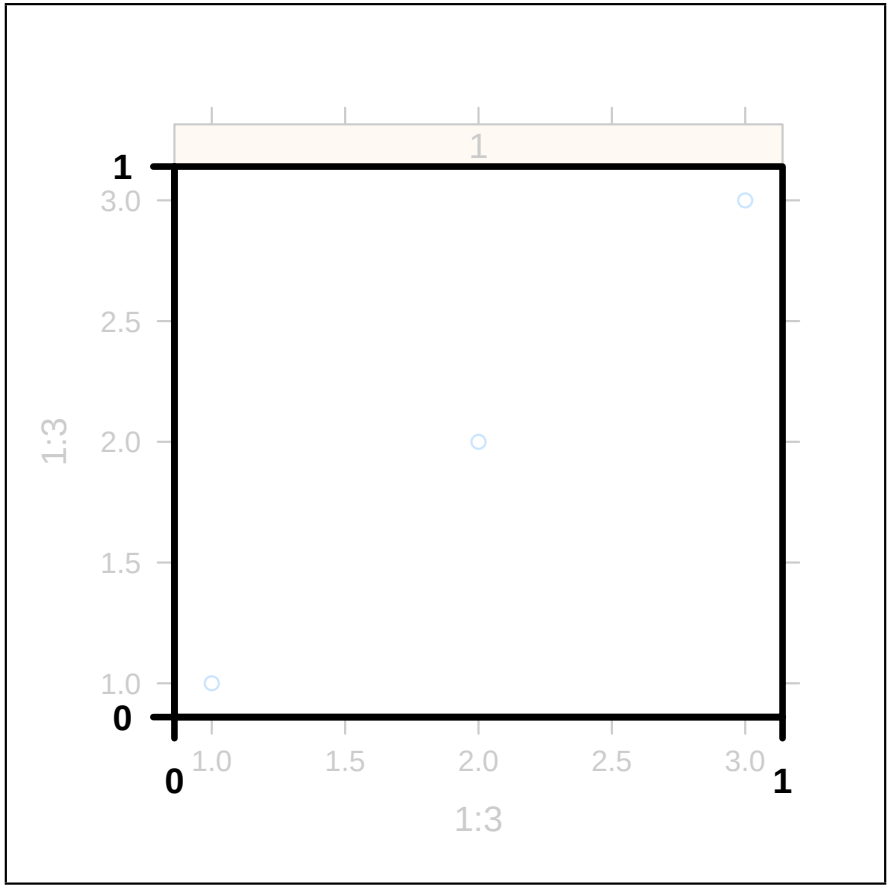
4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.



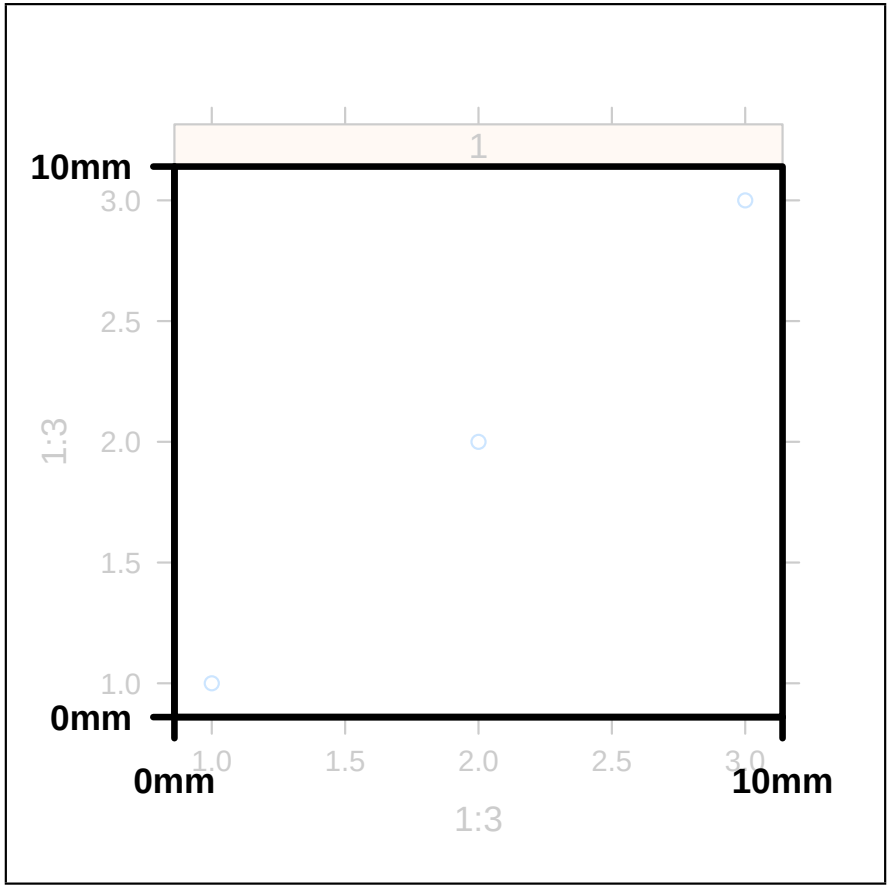
4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.



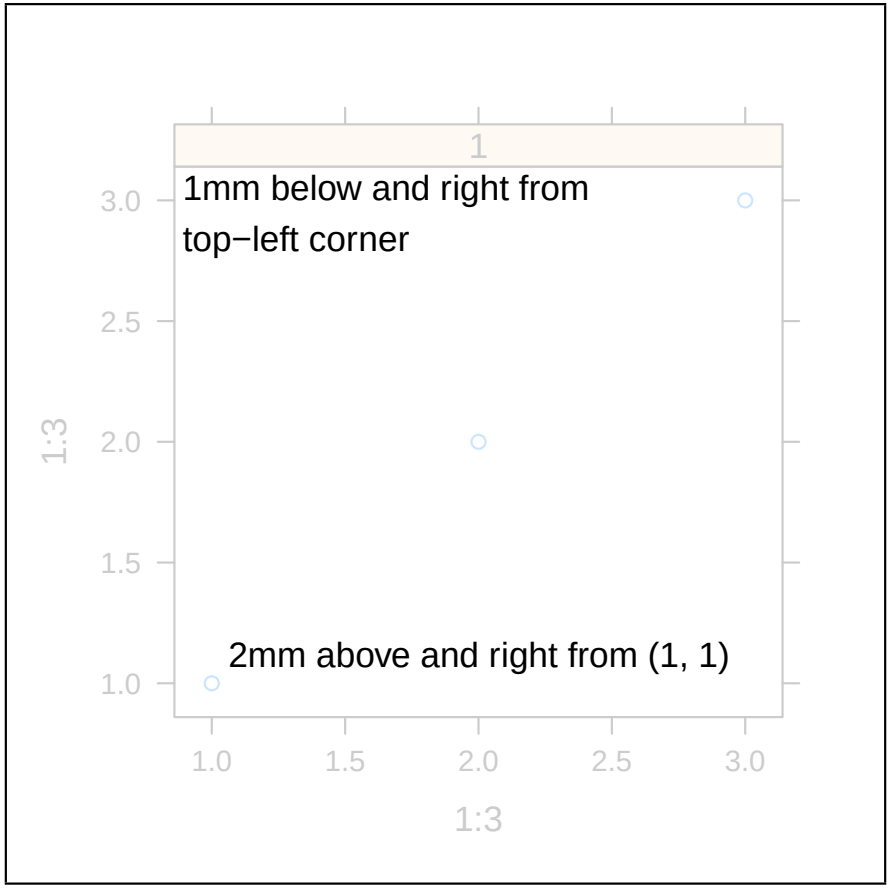
4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.



4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.



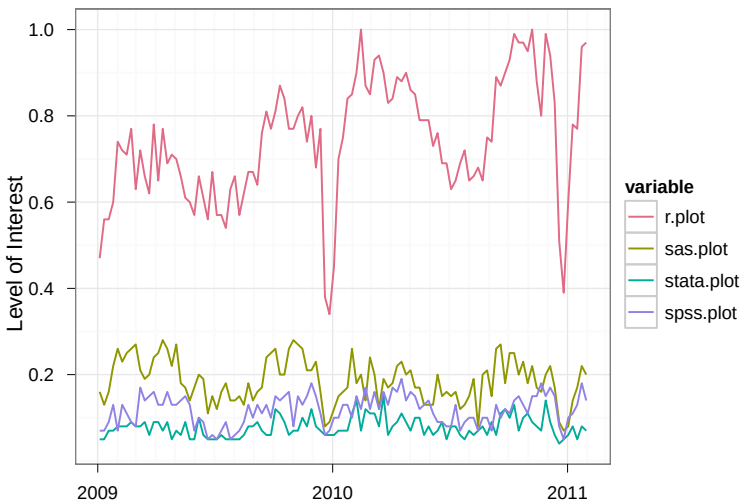
4 Coordinate Systems

- Drawing a plot requires working in multiple coordinate systems.

```
> grid.text("1mm below and right from top-left corner",
            x = unit(0, "npc") + unit(1, "mm"),
            y = unit(1, "npc") - unit(1, "mm"),
            just = c("left", "top"))

> grid.text("2mm above and right from (1, 1)",
            x = unit(1, "native") + unit(2, "mm"),
            y = unit(1, "native") + unit(2, "mm"),
            just = c("left", "bottom"))
```

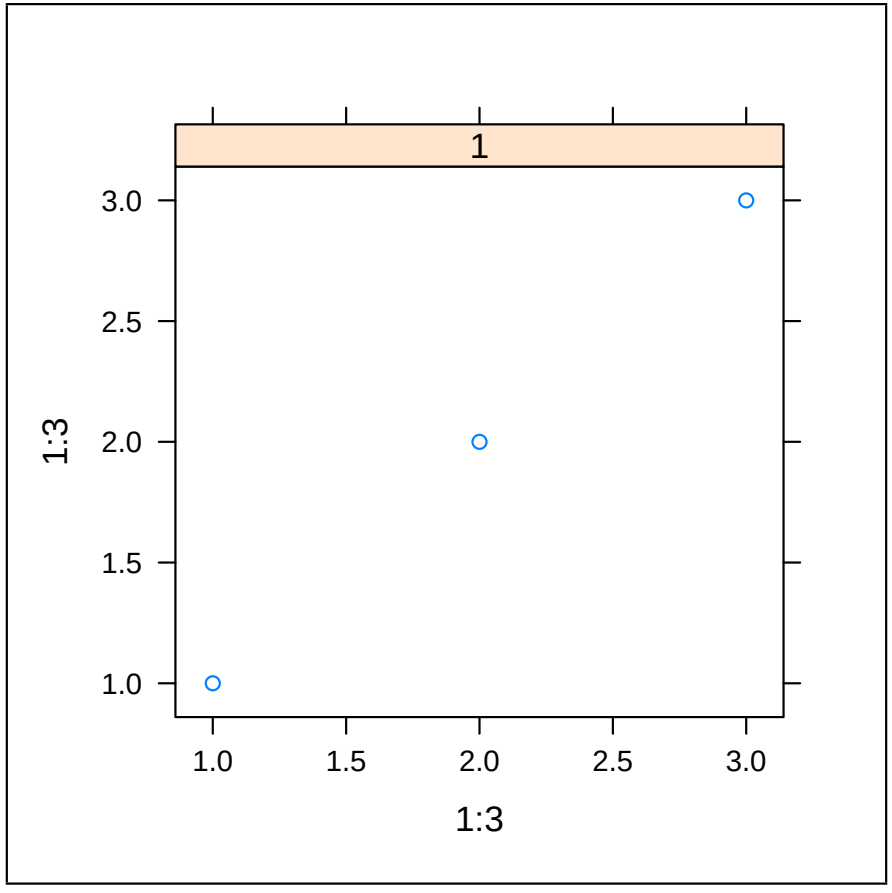
An Example


1.0
0.8
0.6
0.4
0.2
Level of Interest
2009
2010
2011
variable
r.plot
sas.plot
stata.plot
spss.plot


1.0
0.8
0.6
0.4
0.2
Level of Interest
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot

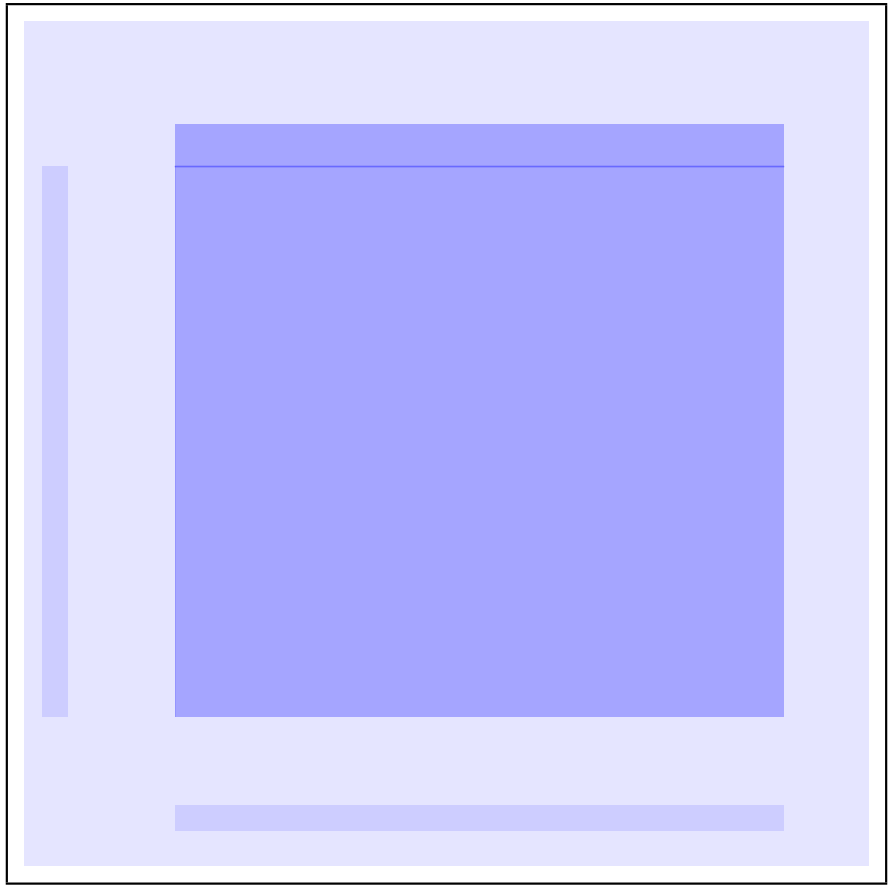
5 Layout

- Drawing a plot requires being able to arrange multiple coordinate systems.



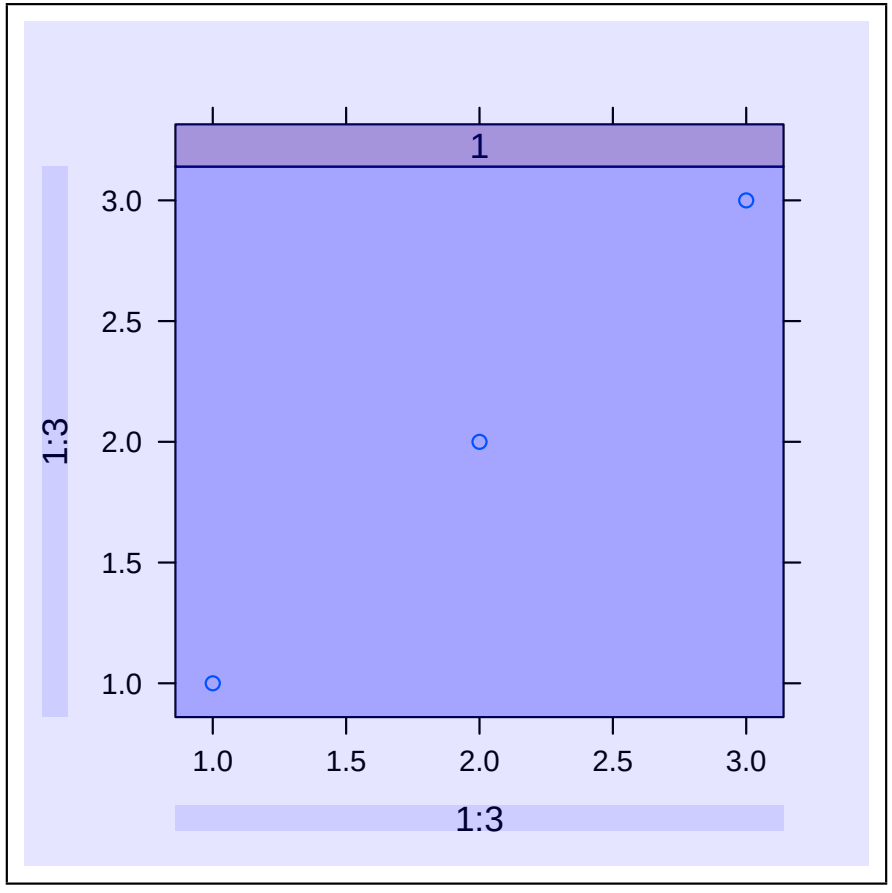
5 Layout

- Drawing a plot requires being able to arrange multiple coordinate systems.



5 Layout

- Drawing a plot requires being able to arrange multiple coordinate systems.

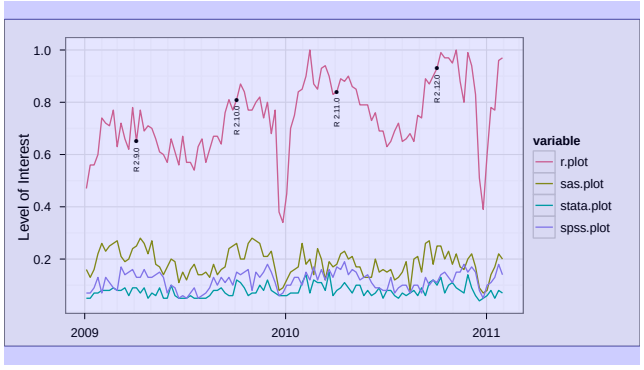


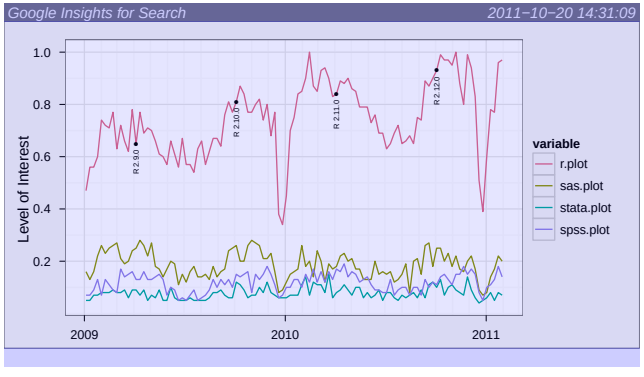
5 Layout

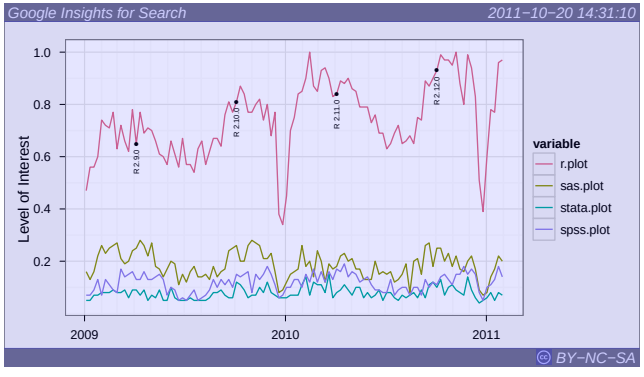
- Drawing a plot requires being able to arrange multiple coordinate systems.

```
> pushViewport(viewport(width = 0.5, height = 0.5,
                        name = "plotvp"))
> upViewport(1)
> downViewport("plotvp")
```

An Example


1.0
0.8
0.6
0.4
0.2
Level of Interest
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot

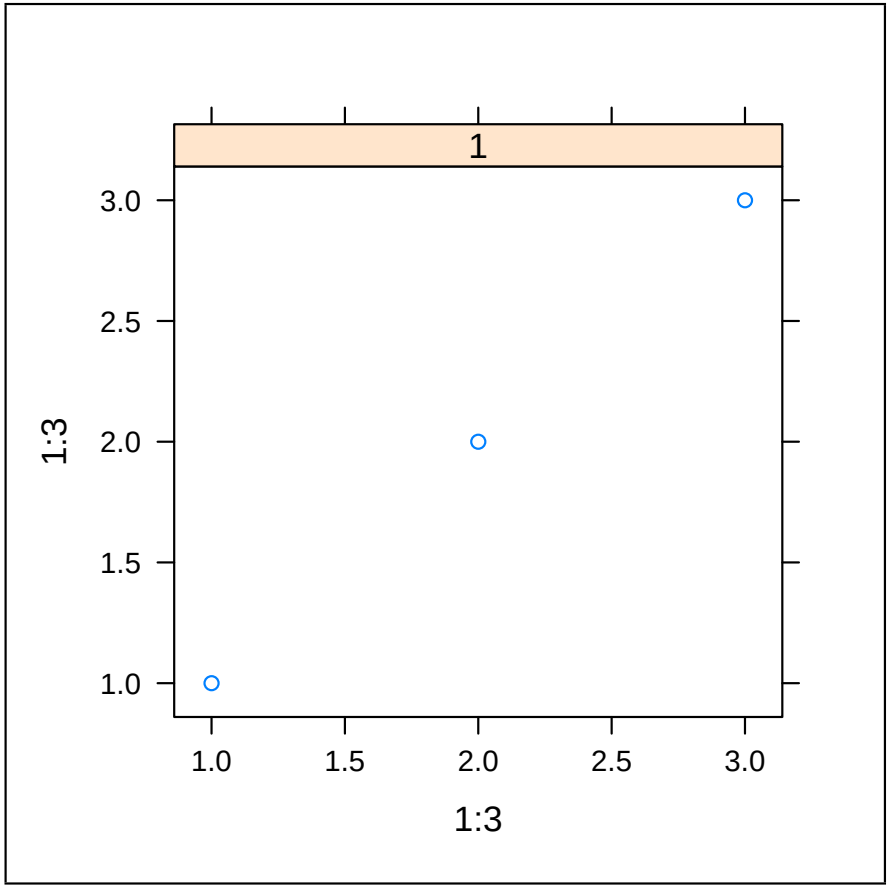

Google Insights for Search
2011−10−20 14:31:09
1.0
0.8
0.6
0.4
0.2
Level of interest
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot


Google Insights for Search
2011−10−20 14:31:10
1.0
0.8
0.6
0.4
0.2
Level of interest
2009
2010
2011
R 2.9.0
R 2.10.0
R 2.11.0
R 2.12.0
variable
r.plot
sas.plot
stata.plot
spss.plot
BY−NC−SA

6 Graphical Objects

- In R graphics, it is possible to access, query and modify components of a plot.

```
plot_01.background
plot_01.xlab
plot_01.ylab
plot_01.ticks.top.panel.1.1
plot_01.ticks.left.panel.1.1
plot_01.ticklabels.left.panel.1.1
plot_01.ticks.bottom.panel.1.1
plot_01.ticklabels.bottom.panel.1.1
plot_01.ticks.right.panel.1.1
plot_01.xyplot.points.panel.1.1
plot_01.border.panel.1.1
plot_01.bg.strip.1.1
plot_01.fg.strip.1.1
plot_01.textl.strip.1.1
plot_01.border.strip.1.1
```



6 Graphical Objects

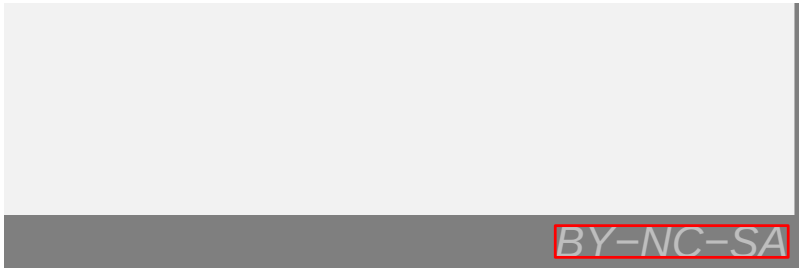
- In R graphics, it is possible to access, query and modify components of a plot.

```
> grid.text("BY-NC-SA",
            x = unit(1, "npc") - unit(1, "mm"),
            just = c("right"),
            name = "cclabel")

> ccx <- unit(1, "npc") - unit(1, "mm") -
         grobWidth("cclabel")

> grid.edit("cclabel", gp=gpar(fontface="bold"))
```

An Example



A Bit of Razzmatazz

Dynamic and Interactive Demo

1 Code for Graphics

- We create an image in R using **code**.

1 Code for Graphics

- We create an image in R using **code**.

```
library(EBImage)
trim <- function(col) {
    if (any(col < 0 | col > 1)) {
        col[col < 0] <- 0
        col[col > 1] <- 1
    }
    col
}
as.raster.Image <- function(x) {
    # For now, bail if there is more than one image
    if (getNumberOfFrames(x, "render") > 1)
        stop("Cannot handle multiple frames")
    image <- imageData(x)
    # Either a grey scale or a color image
    if (colorMode(x) == 0) { # Grayscale
        r <- grey(image)
        dim(r) <- dim(image)
    } else { # Color
        r <- rgb(aperm(trim(image[,,1])),
                 aperm(trim(image[,,2])),
                 aperm(trim(image[,,3])))
        dim(r) <- dim(image)[2:1]
    }
    r
}
tree <- as.raster(readImage("/scratch/TealBay/tree.jpg"))
rock <- as.raster(readImage("/scratch/TealBay/rock.jpg"))
rope <- as.raster(readImage("/scratch/TealBay/rope.jpg"))
sand <- as.raster(readImage("/scratch/TealBay/sand.jpg"))
gull <- as.raster(readImage("/scratch/TealBay/gull.jpg"))
hill <- as.raster(readImage("/scratch/TealBay/hill.jpg"))
sign <- as.raster(readImage("/scratch/TealBay/sign.jpg"))
pohu <- as.raster(readImage("/scratch/TealBay/pohu.jpg"))

library(grid)
imageLayout <- grid.layout(4, 5,
                           widths=c(2, 1.5, 2, 1, 2.5),
                           heights=c(2.5, .5, 2, 1),
                           respect=TRUE)
margin <- unit(20, "mm")
drawImage <- function(c, r, img=NULL, rot=FALSE, width=NULL, height=NULL,
                      ...) {
    pushViewport(viewport(layout.pos.col=c,
                          layout.pos.row=r),
                 viewport(width=unit(1, "npc") - margin,
                          height=unit(1, "npc") - margin))
    if (is.null(img)) {
        grid.rect()
    } else {
        if (rot) {
            pushViewport(viewport(width=convertUnit(unit(1, "npc"),
                                    "npc", "y", "dimension", "x", "dimension"),
                                  height=convertUnit(unit(1, "npc"),
                                    "npc", "x", "dimension", "y", "dimension"),
                                  angle=90,
                                  clip=TRUE))
        } else {
            pushViewport(viewport(clip=TRUE))
        }
        grid.raster(img, width=width, height=height, ...)
        # grid.rect()
        popViewport()
    }
    popViewport(2)
}
```

```
jpeg("tealbay.jpg", width=3600, height=2400)
grid.newpage()
grid.rect(gp=gpar(col=NA, fill="white"))
pushViewport(viewport(width=unit(1, "npc") - margin,
                      height=unit(1, "npc") - margin,
                      layout=imageLayout))
pushViewport(viewport(layout.pos.col=1:4))
grid.rect(width=unit(1, "npc") + margin,
          height=unit(1, "npc") + margin,
          gp=gpar(col=NA, fill="white"))
popViewport()
drawImage(1, 1, tree, width=2)
drawImage(1, 2:3, rock, height=1, y=1, just="top")
drawImage(2, 1:3, rope, TRUE, width=1, y=.45)
drawImage(1:3, 4, sand, width=1, height=1.5, y=0, just="bottom")
drawImage(3, 2:3, gull, width=2, y=0, just="bottom")
drawImage(3:5, 1, hill, width=1.5, y=1, vjust=.85)
drawImage(4:5, 2, sign, width=2, vjust=.605)
drawImage(4:5, 3:4, pohu, height=1, x=1, just="right")
popViewport()
dev.off()
```

1 Code for Graphics

- This means that ...
  - we have a record of what we did.
  - we can easily repeat what we did.
  - we can create a similar image by slightly modifying what we did.
  - we can create lots of images easily (batch jobs).
  - we can easily and accurately show someone else what we did (and they can do exactly what we did, or something similar).
  - we can generate plot content on-the-fly (e.g., time stamps).
  - we can express complex ideas (edit all graphical objects that match a pattern).

1 Code for Graphics

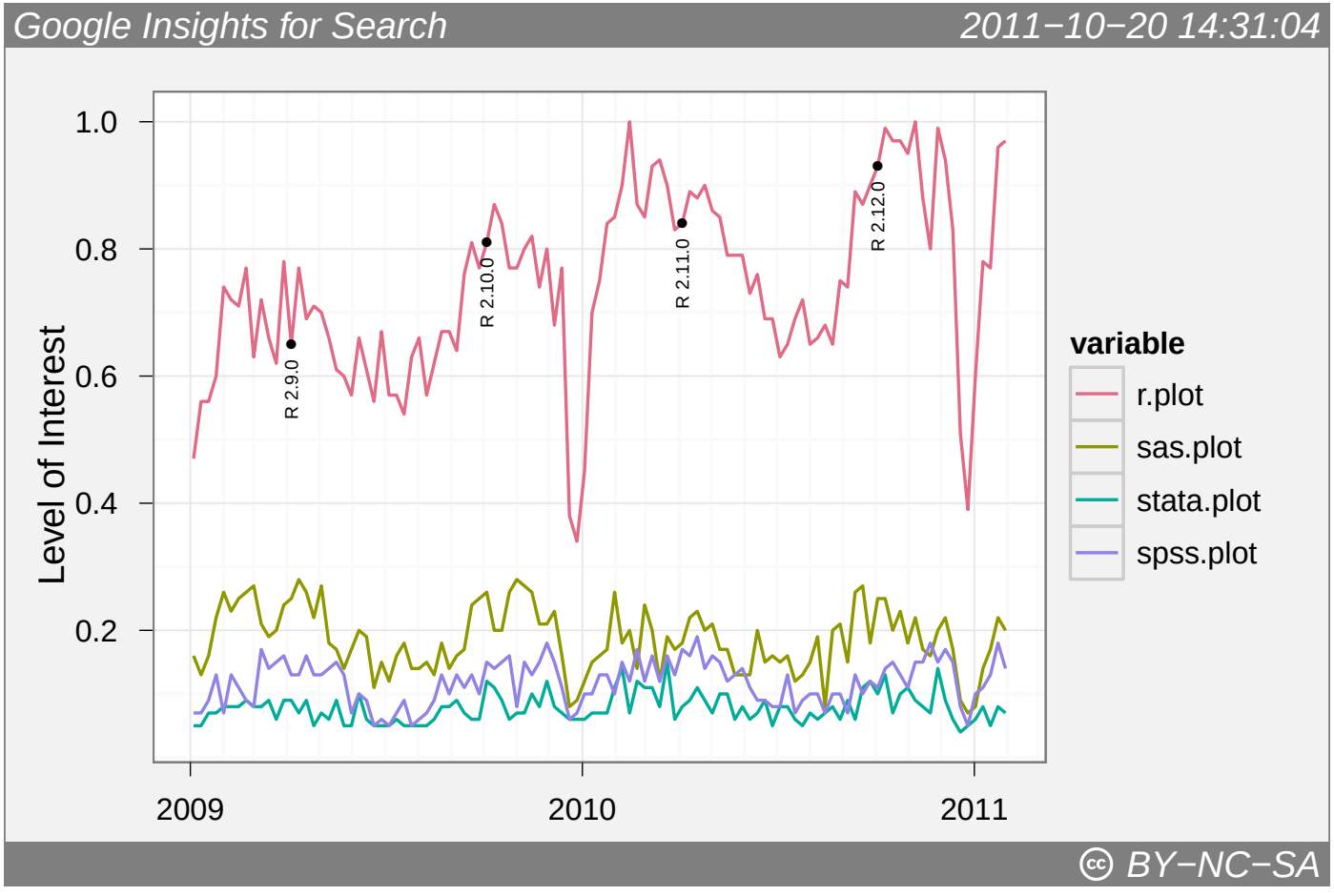
- Because R is a programming language, we can encapsulate code in functions or even packages.
- It also doesn't hurt that R is a **statistical** programming language.
  - We can add a least-squares line to a plot.
- It hurts even less that R is **open source**.
  - We can easily share and build on each other's code.

An Example

```
> plotframe("Title goes here")
```

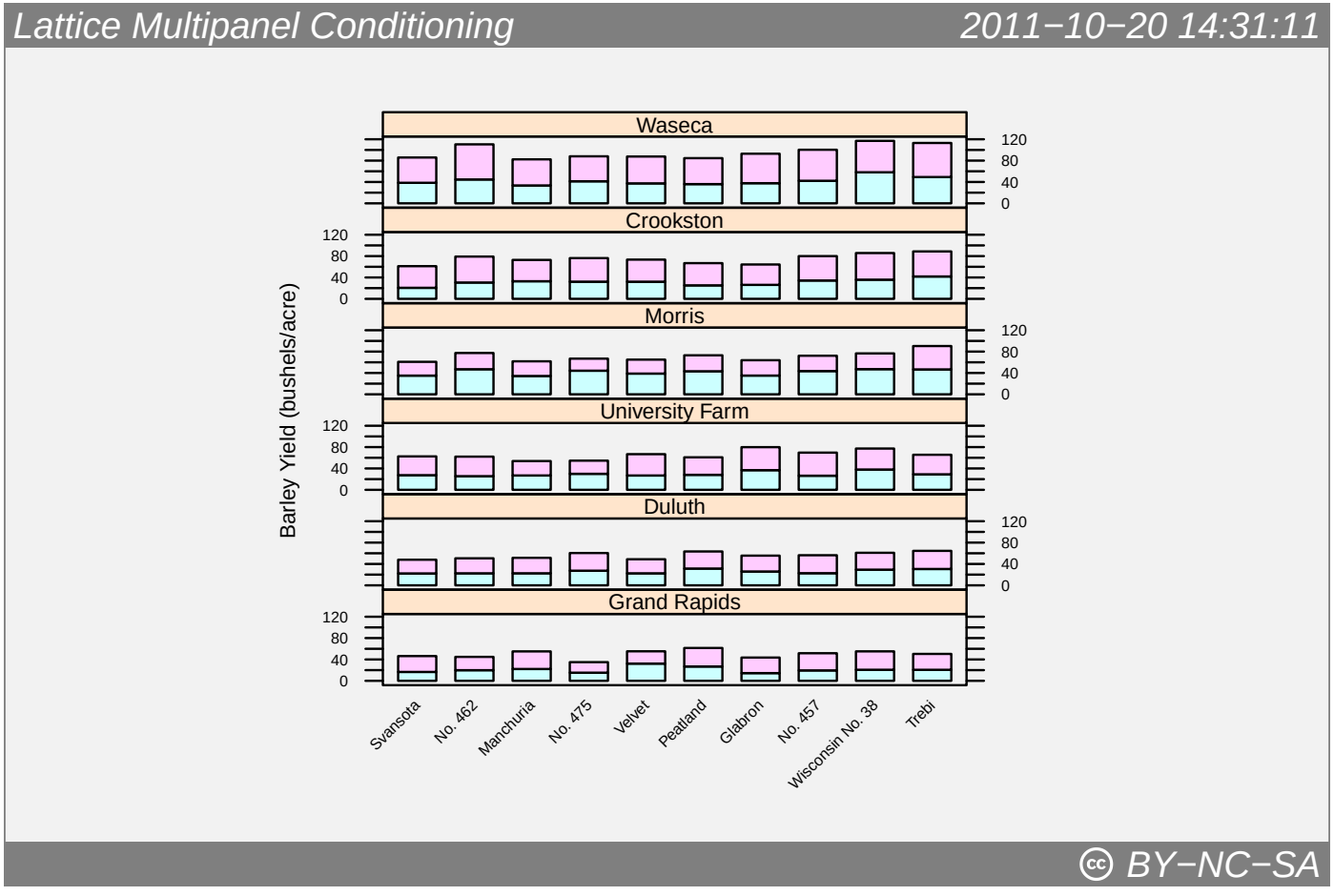
An Example

```
> plotframe("Google Insights for Search")
> ggplot( ... )
```



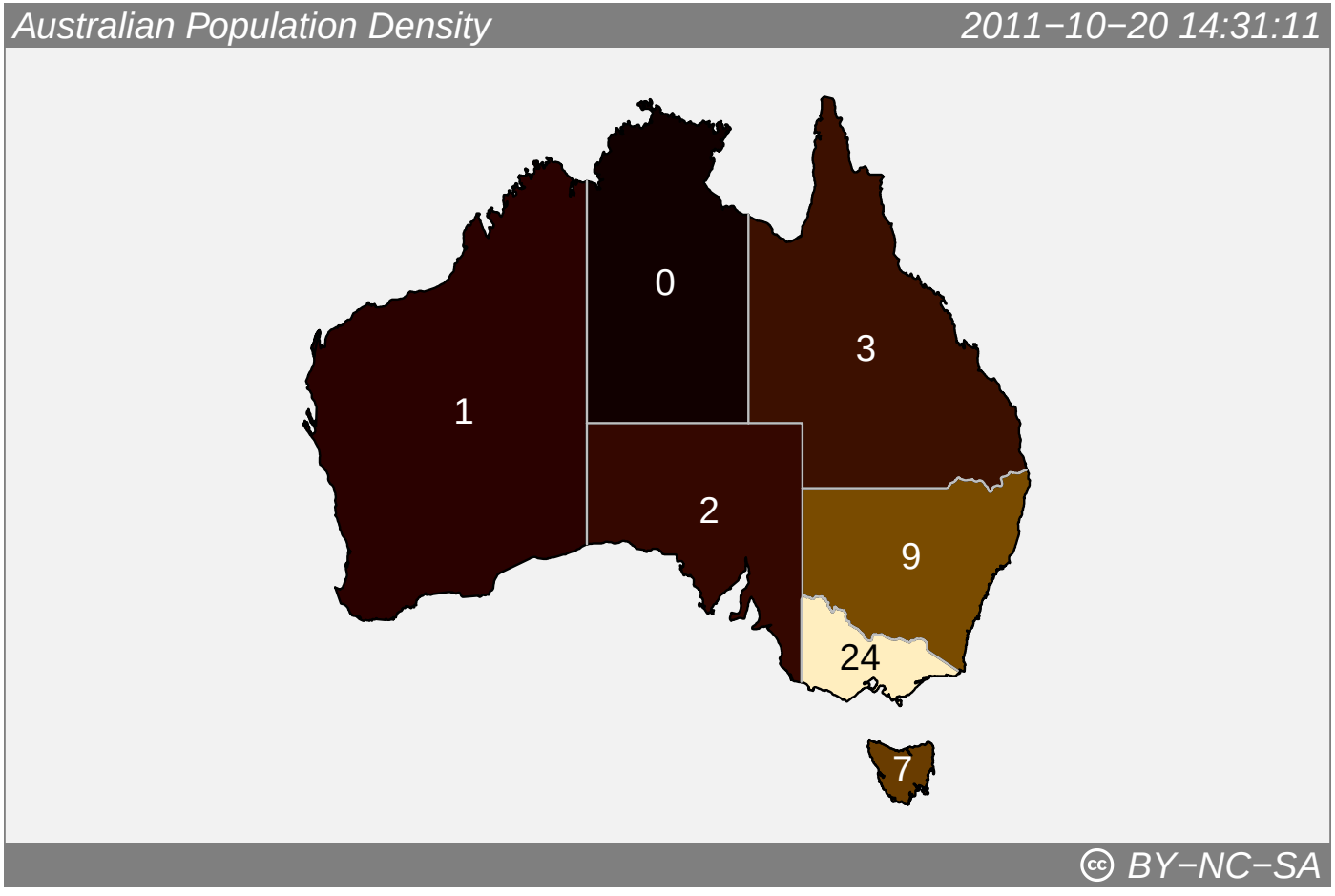
An Example

```
> plotframe("Lattice Multipanel Conditioning")
> xyplot( ... )
```



An Example

```
> plotframe("Australian Population Density")
> oz( ... )
```

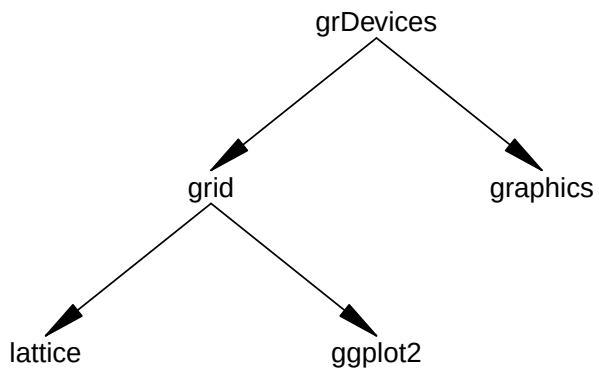


Summary

- Why use R?
  - Because if R can't already do what you want, it gives you lots of tools so you can do it yourself.
  - Customisability:
    - Because of the painters model.
    - Because you can draw basic shapes.
    - Because you have access to coordinate systems.
    - Because you have access to graphical objects.
  - Extensibility:
    - Because you can write code.
    - Because you can see others' code.
    - Because you have access to the same tools that others use.
  - Programmability:
    - Because you write code to draw stuff.
    - Because R is a programming language.
    - Because R is a graphics language.

Caveats

- Although it is possible with R to produce 3D plots and animated plots and interactive plots, those are typically provided by connecting R to external graphics systems.
- The **core** R graphics system is focused on **static 2D** plots.



- Most of the examples have been **grid**-based graphics.
- (Almost) all of these examples can be achieved (with a little more effort) in "traditional" graphics.

Further Reading

- "R Graphics"
  http://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html
- "R Graphics, Second Edition"
  http://www.stat.auckland.ac.nz/~paul/RG2e/
- "Lattice: Multivariate Data Visualization with R"
  http://lmdvr.r-forge.r-project.org/
- "ggplot2: Elegant Graphics for Data Analysis"
  http://had.co.nz/ggplot2/book/

- The Graphics CRAN Task View
  http://cran.r-project.org/web/views/Graphics.html
- The R Graphics Gallery
  http://addictedtor.free.fr/graphiques/
- The R Graphical Manual
  http://www.oga-lab.net/RGM2/

Acknowledgements

- The Creative Commons logo is available from `http://creativecommons.org/about/downloads` under a Creative Commons Attribution license.
- The GNU logo is by Aurélio A. Heckert and is available from `http://commons.wikimedia.org/wiki/Image:Heckert_GNU_white.svg` under a Free Art Licence.
- The Google Search data are from `http://www.google.com/insights/search/`

Acknowledgements

The following R packages were used in the production of these slides:

- **ggplot2** by Hadley Wickham.
- **graph** (maintained) by Seth Falcon (Bioconductor).
- **gridBase** by Paul Murrell.
- **gridSVG** by Paul Murrell and Simon Potter.
- **grImport** by Paul Murrell and Richard Walton.
- **lattice** by Deepayan Sarkar.
- **maps** (maintained) by Ray Brownrigg.
- **oz** (maintained) by Kurt Hornik.
- **png** by Simon Urbanek.
- **reshape** by Haldey Wickham.
- **Rgraphviz** (maintained) by Kasper Hansen (Bioconductor).