# Dynamic and Interactive R Graphics for the Web

Paul Murrell

The University of Auckland

July 2011

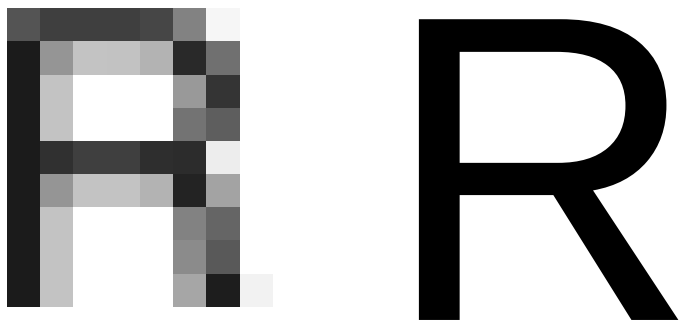
# Overview

The aim is to produce dynamic and interactive R graphics for the web

- R can produce SVG for the web, but it's static
- Several packages can do interaction, but NOT R plots and NOT for the Web
- Some packages can do graphics for the web, but NOT R plots
- **SVGAnnotation** does some cool stuff with R plots for the web, but it's black magic
- **gridSVG** holds hope for more transparent approach

# R Graphics for the Web

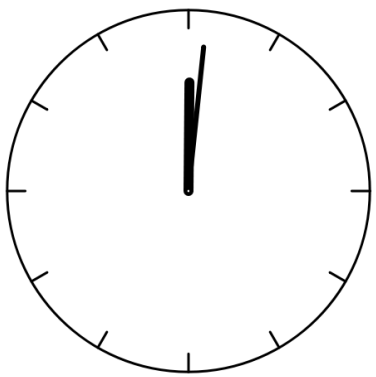
- `png()` produces images suitable for any browser
- SVG, via `svg()` function or **Cairo** package, is better ...
  - SVG is vector graphics so it scales
  - SVG allows text search (in theory)



... but it is only **static** R graphics

# Dynamic and Interactive Graphics

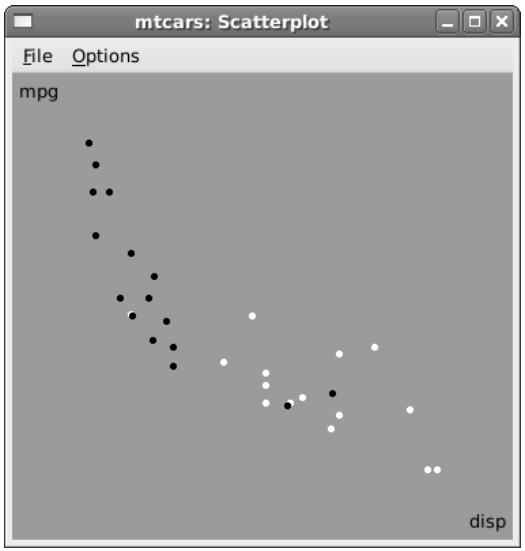
- The core graphics facilities are focused on static plots
  - You can draw lots of static plots and "stitch" them together (e.g., the **animation** package)
  - There are `locator()` and `getGraphicsEvent()` functions

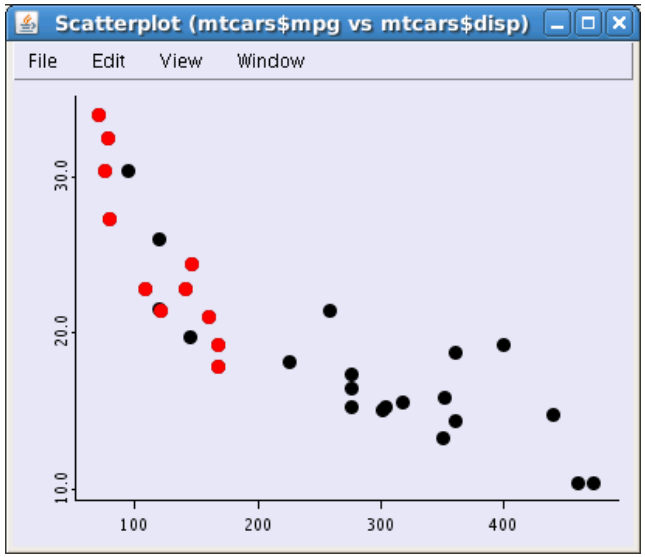


... which does give you R graphics, but the animation is "stop motion" and the interaction is crude (blocks the command line, only pixel coordinates for mouse events)

# Dynamic and Interactive Graphics

- There are stand-alone interactive systems ...
  - The **rggobi** package provides an interface to GGobi
  - The **iplots** package is a self-contained system based on Java
  - The `qtinterfaces` suite of packages is a self-contained system based on Qt

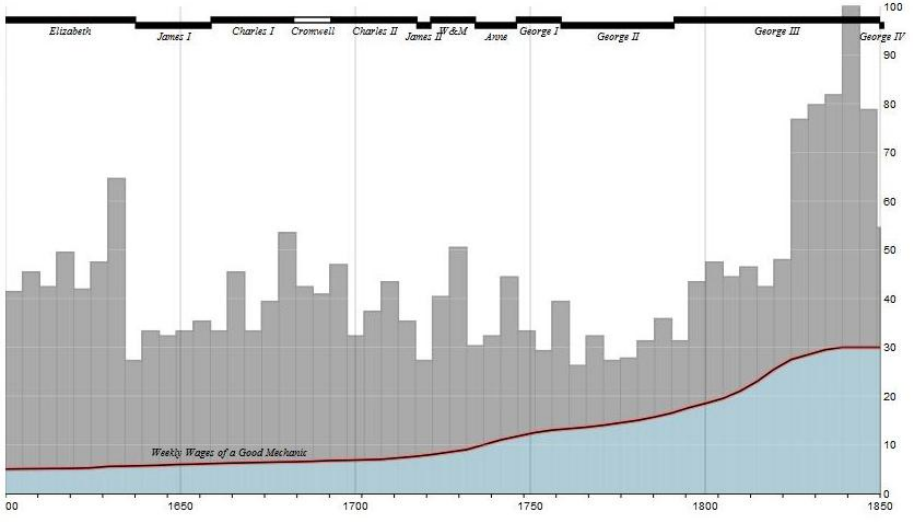



... but they are not R graphics and they are not for the web.

# Dynamic and Interactive Graphics for the Web

- There are several packages that interface to web systems (mostly based on javascript)
  - The **webvis** package provides an R interface to the protovis system (javascript + SVG)
  - The **googleVis** package provides an interface to the Google Visualisation API





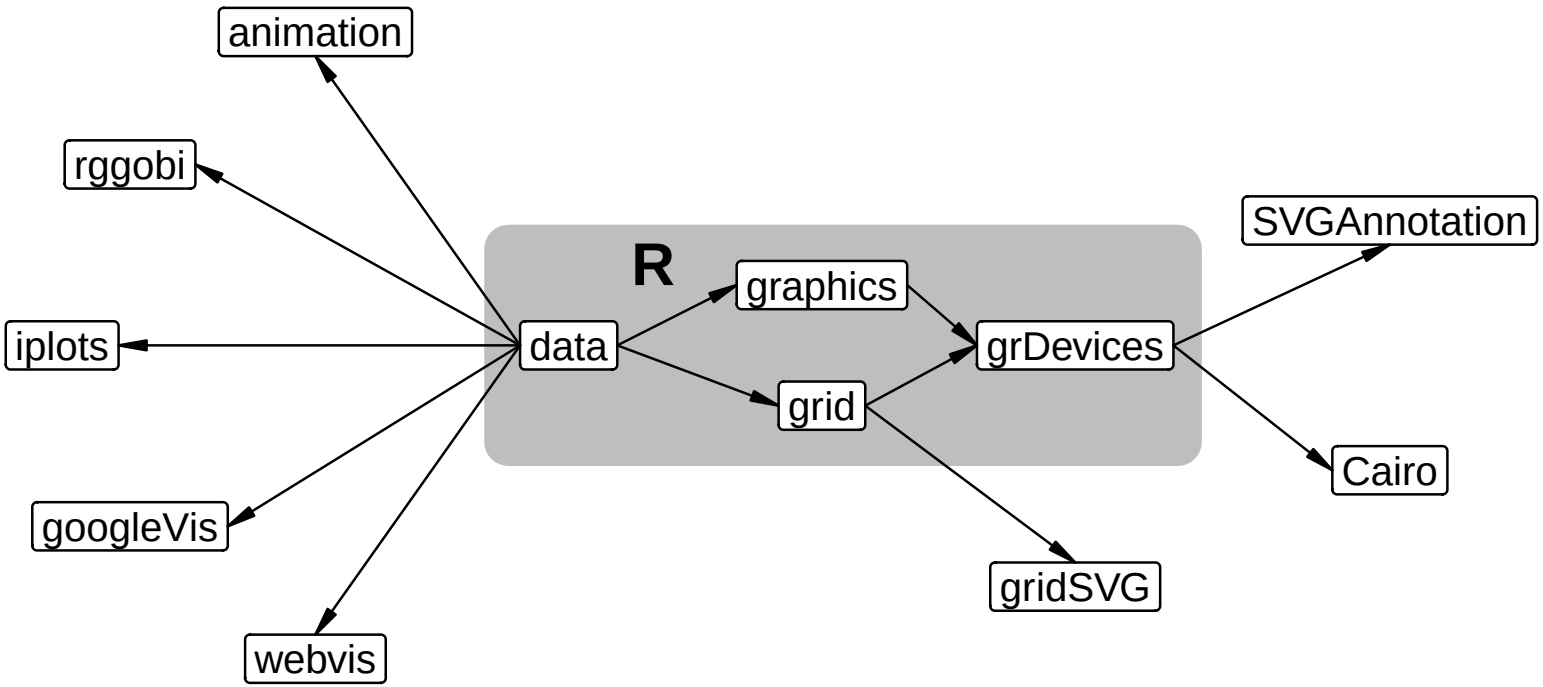
... but they are not R graphics.

# Dynamic and Interactive Graphics for the Web

- The **SVGAnnotation** package exports R graphics as SVG and provides functions for adding interactivity (with javascript) to the exported SVG ...
  - Any R graphics output can be used
  - Includes animation, tooltips, and even linked plots

  ... but adding animation and interactivity and linking plots is "black magic"

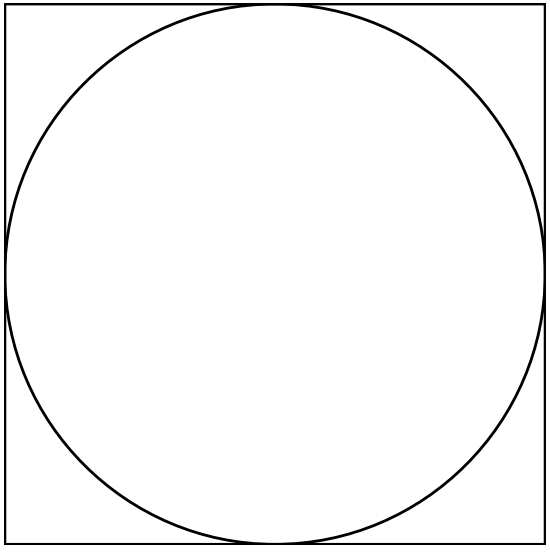
# Dynamic and Interactive Graphics for the Web



# The **gridSVG** package: Introduction

- The **grid** package creates **grobs** - objects that contain information about what to draw - and records them on a **display list**.

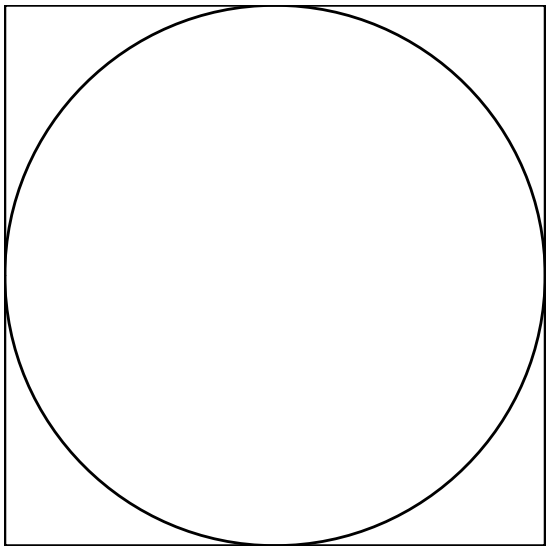
```
> library(grid)

> grid.rect()
> grid.circle()

> grid.ls()

GRID.rect.3
GRID.circle.4
```



# The **gridSVG** package: Introduction

- **grid** grobs can be **named**.

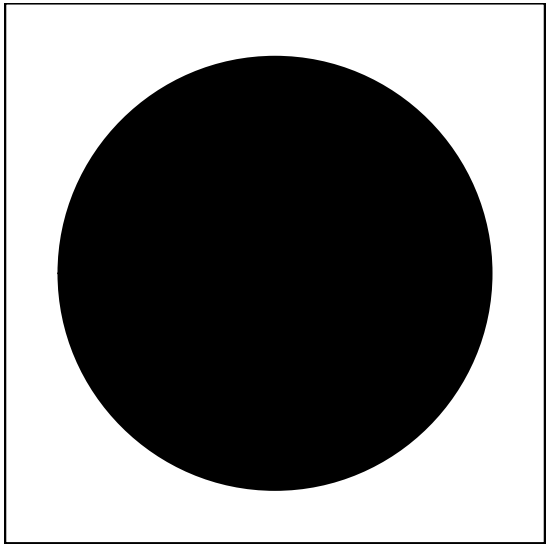
```
> grid.rect(name="myrect")
> grid.circle(name="mycircle")

> grid.ls()

myrect
mycircle
```



# The **gridSVG** package: Introduction

- Grobs on the display list can be accessed **by name** and **modified**.

```
> grid.edit("mycircle",
            r=unit(.4, "npc"),
            gp=gpar(fill="black"))
```



# The **gridSVG** package: Introduction

- The **gridSVG** package converts grobs into SVG elements.
- Grob names map to SVG `id` attributes.

```
> library(gridSVG)

> grid.rect(name="myrect")
> grid.circle(name="mycircle")
> gridToSVG("grobs.svg")
```

```
...
<rect id="myrect.1" x="0" y="0"
      width="689" height="689.40415704388" />
...
<circle id="mycircle.1"
        cx="344.5" cy="344.70207852194"
        r="344.5" />
...
```

# The **gridSVG** package: Animation

- The **gridSVG** package provides functions to modify grobs in special ways.

```
> grid.rect(name="myrect")
> grid.circle(name="mycircle")
> grid.animate("mycircle",
               r=c(seq(.5, .1, -.1), seq(.1, .5, .1)),
               duration=2, rep=TRUE)
> class(grid.get("mycircle"))

[1] "animated.grob" "circle"
[3] "grob"          "gDesc"

> names(grid.get("mycircle"))

[1] "x"          "y"          "r"
[4] "name"       "gp"         "vp"
[7] "animations"
```

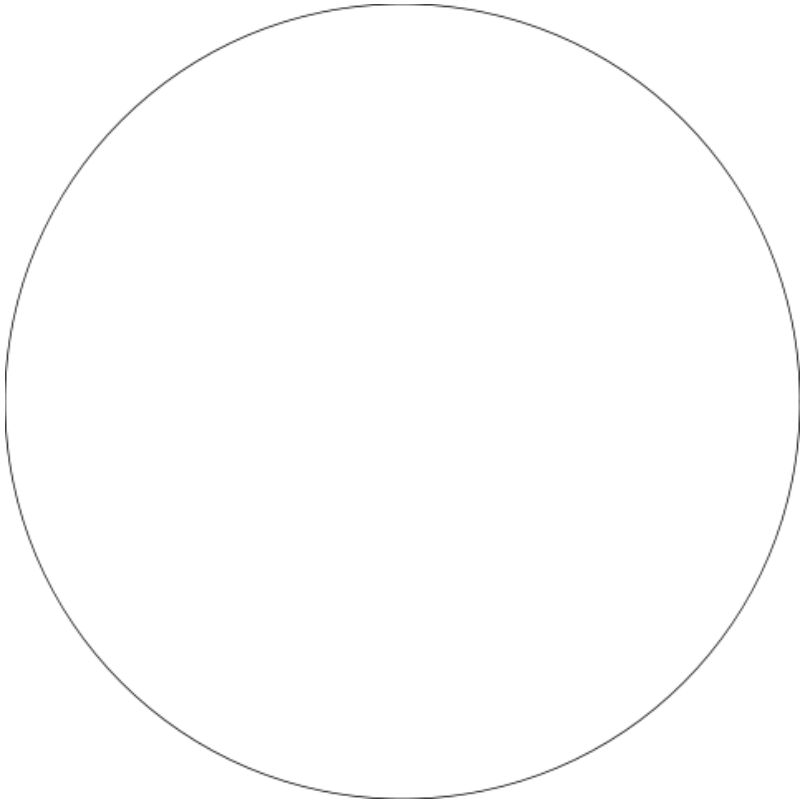
# The **gridSVG** package: Animation

- When the modified grob is exported, additional SVG **elements** are produced.

```
> gridToSVG("animate.svg")
```

```
<animate xlink:href="#mycircle.1"
         attributeName="r" dur="2s"
         values="453.6;403.2;352.8;302.4;..."
         repeatCount="indefinite" fill="freeze" />
...
<circle id="mycircle.1" cx="252" cy="252" r="252" />
```

# The **gridSVG** package: Animation



# The **gridSVG** package: Garnishing

- The **gridSVG** package provides functions to modify grobs in special ways.

```
> grid.rect(name="myrect")
> grid.circle(name="mycircle")
> grid.garnish("mycircle",
                onmouseover="godark()")
> class(grid.get("mycircle"))

[1] "svg.grob" "circle"   "grob"     "gDesc"

> names(grid.get("mycircle"))

[1] "x"          "y"          "r"
[4] "name"       "gp"         "vp"
[7] "attributes"
```

# The **gridSVG** package: Garnishing

- When the modified grob is exported, additional SVG **attributes** are produced.

```
> gridToSVG("garnish.svg")
```

```
<g id="mycircle" onmouseover="godark()" >
<circle id="mycircle.1" cx="252" cy="252" r="252" />
</g>
```

# The **gridSVG** package: Scripting

- The **gridSVG** package provides a function that creates a new sort of grob.

```
> grid.rect(name="myrect")
> grid.circle(name="mycircle")
> grid.garnish("mycircle",
                onmouseover="godark()")
> grid.script('
    godark = function() {
        circle = document.getElementById("mycircle.1");
        circle.setAttribute("style", "fill:black");
    }')
> grid.ls()

myrect
mycircle
GRID.script.18
```

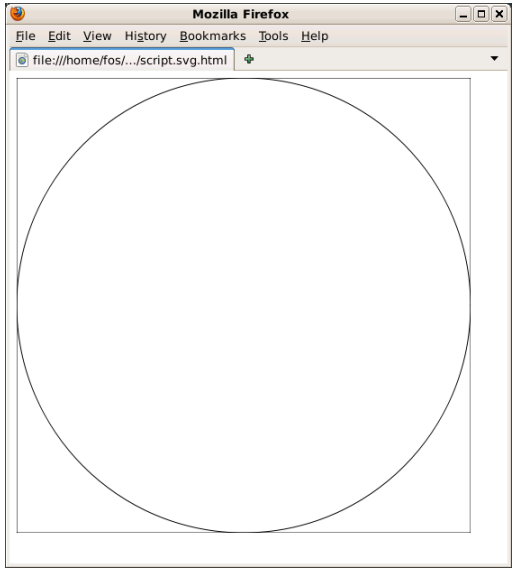
# The **gridSVG** package: Scripting

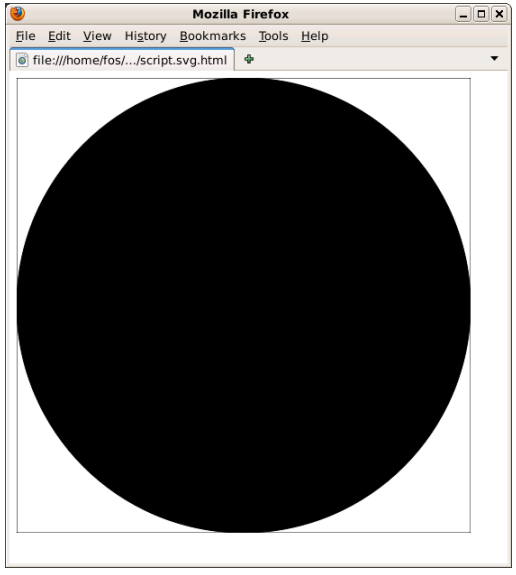
- The javascript code can take advantage of the `id` attributes that have come from the grob names.

```
> gridToSVG("script.svg")
```

```
<script type="text/ecmascript" id="4" >
<![CDATA[
  godark = function() {
      circle = document.getElementById("mycircle.1");
      circle.setAttribute("style", "fill:black");
  }
  ]]>
</script>
```

# The **gridSVG** package: Scripting



# The **gridSVG** package: Summary

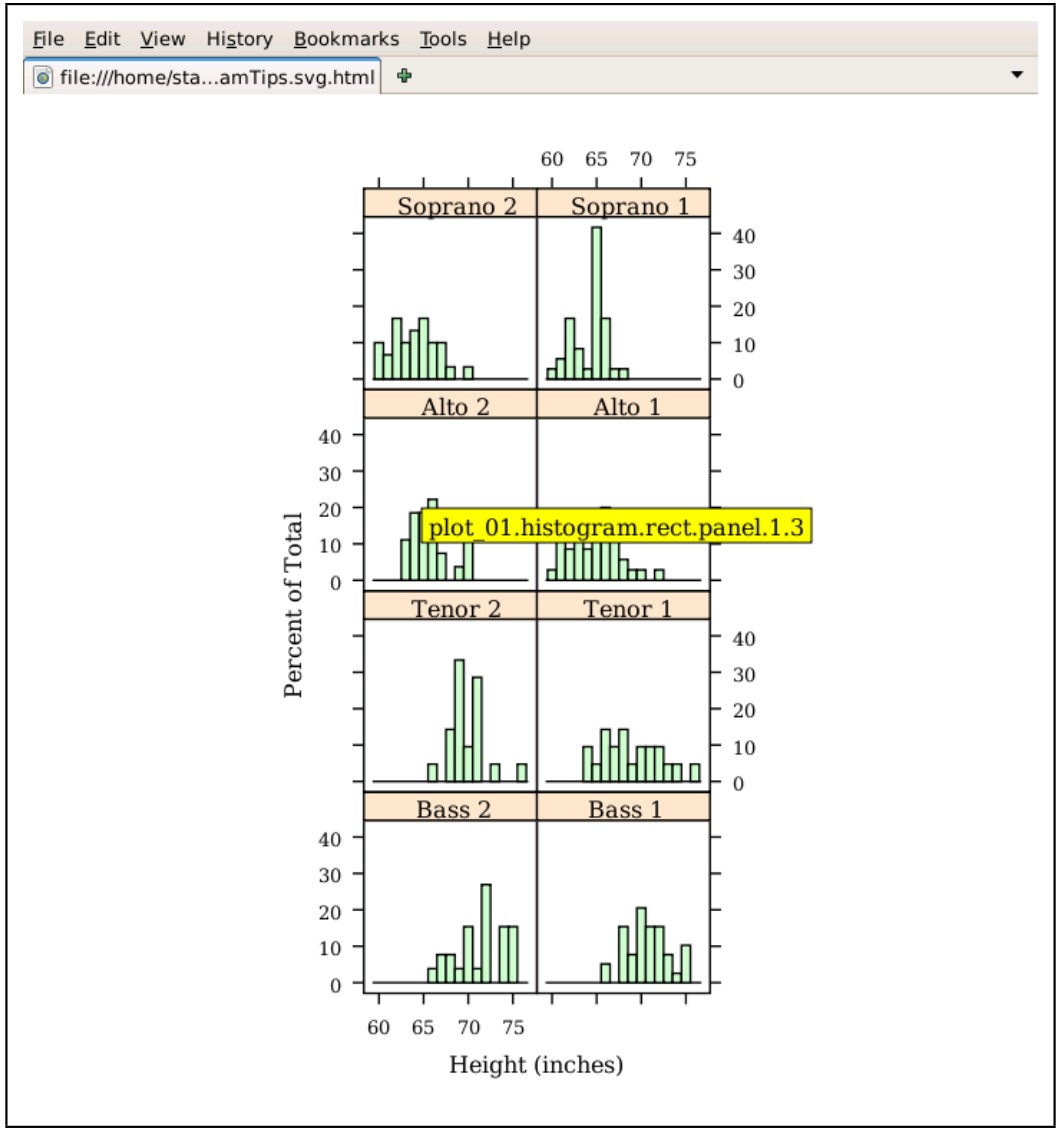
| **R code** | **R object** | **SVG code** |
|---|---|---|
| `xyplot()` | grob (`"points"`) | `<path id="points" ...>` |
| `grid.animate("points")` | animated.grob | `<animate href="#points" ...>` |
| `grid.garnish("points")` | svg.grob | `<path id="points"`<br>`onmouseover=showTip()`<br>`...>` |
| `grid.script()` | script.grob | `<script>`<br>`showTip() <- function() {`<br>`...`<br>`}`<br>`</script>` |

# The **gridSVG** package: Examples

- An object browser for **grid** scenes.

```
garnishAllGrobs <- function(elt) {
  if (inherits(elt, "grob")) {
    garnishGrob(elt,
                onmousemove=paste("showTooltip(evt, '",
                  gsub("\n", " ", elt$name), "')",
                  sep=""),
                onmouseout="hideTooltip()")
  } else {
    elt
  }
}

addTooltips <- function() {
  grid.DLapply(garnishAllGrobs)
  grid.script(filename="tooltip.js")
}

histogram( ~ height | voice.part, data = singer,
          xlab = "Height (inches)")
addTooltips()
gridToSVG("tooltips.svg")
```

# The **gridSVG** package: Examples



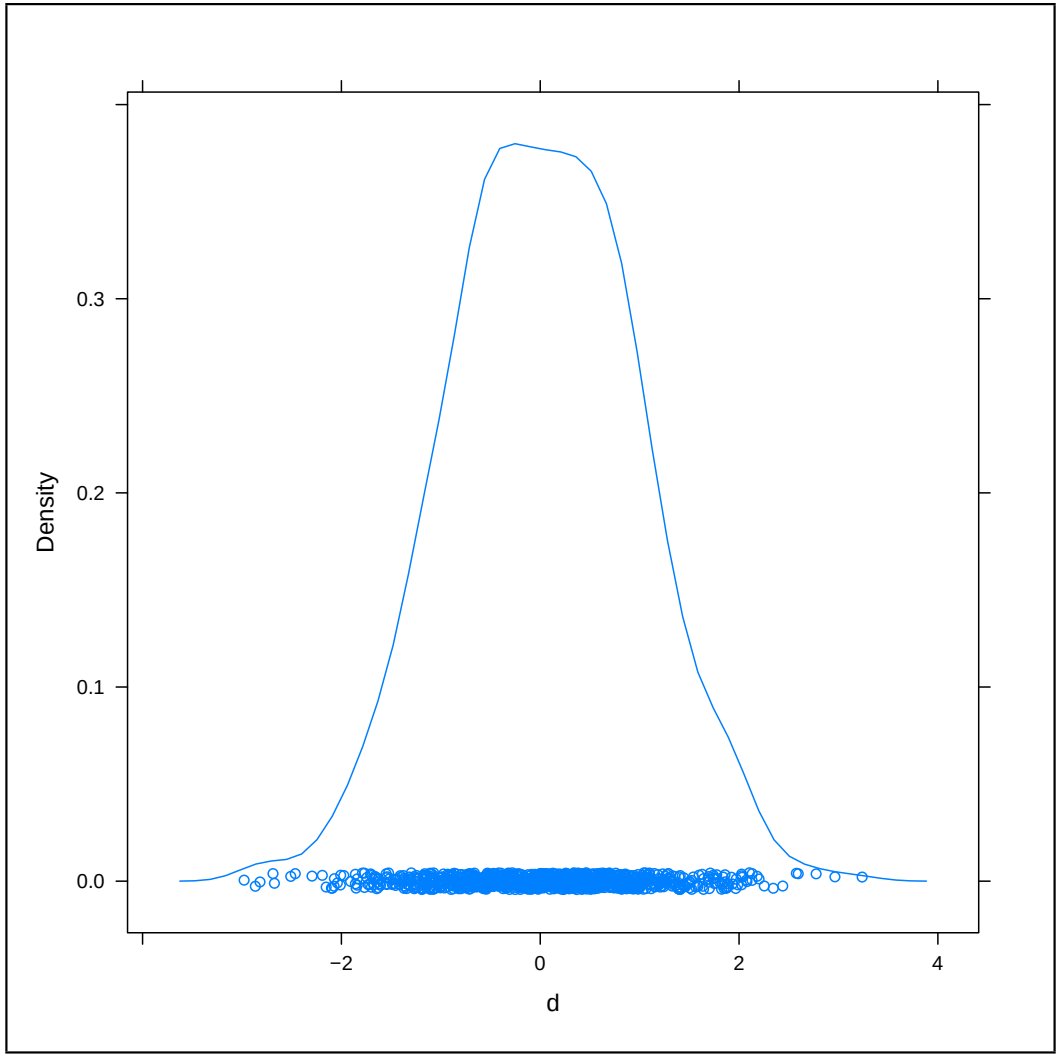

http://www.stat.auckland.ac.nz/~paul/Genentech2011/tooltips.svg.html

# The **gridSVG** package: Examples

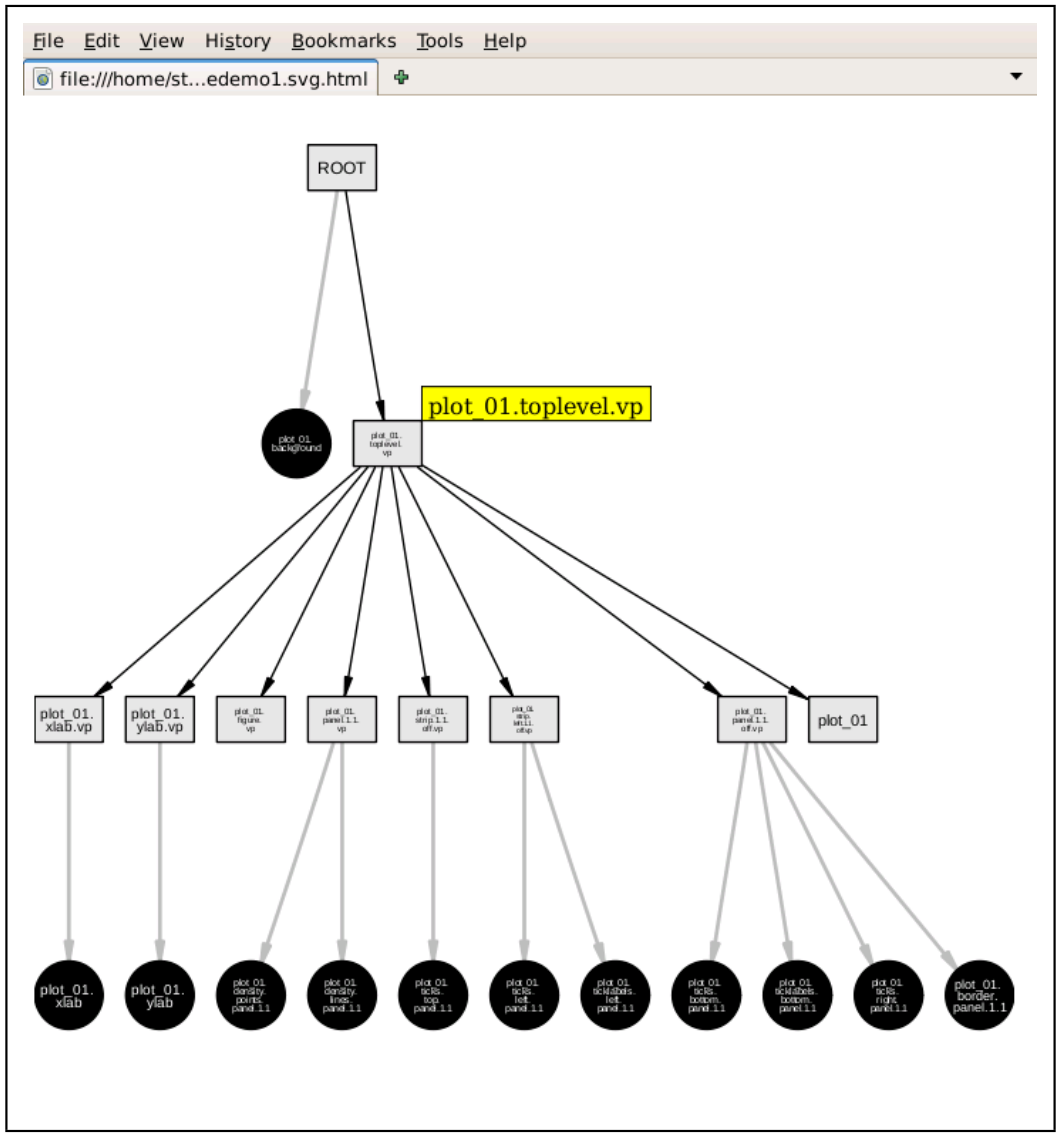
- A scene graph browser for **grid** scenes.

```
library(gridDebug)

library(lattice)
d <- rnorm(1000)
densityplot(~d)

gridTree(grid=TRUE)
addTooltips("gridtree.svg")
```

# The **gridSVG** package: Examples



# The **gridSVG** package: Examples



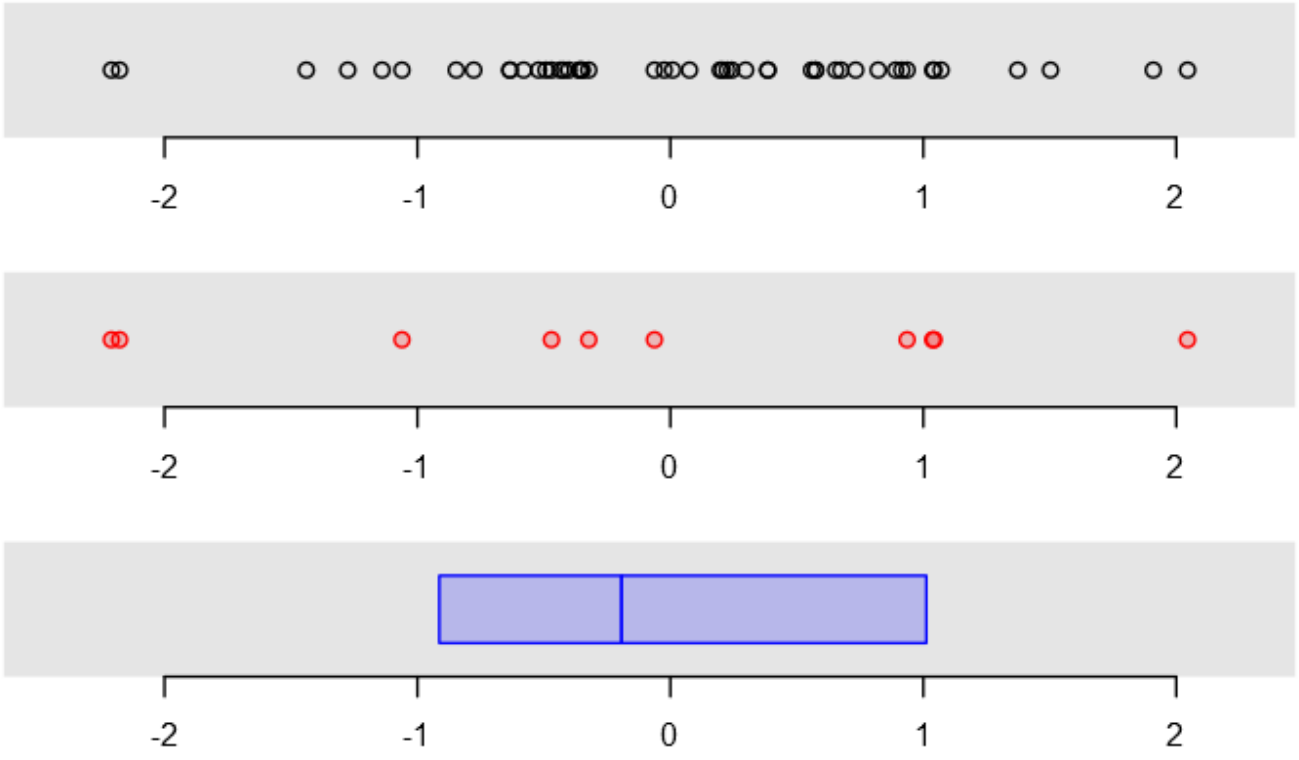

http://www.stat.auckland.ac.nz/~paul/Genentech2011/gridtree.svg.html

# The **gridSVG** package: Examples

- Animations for teaching.

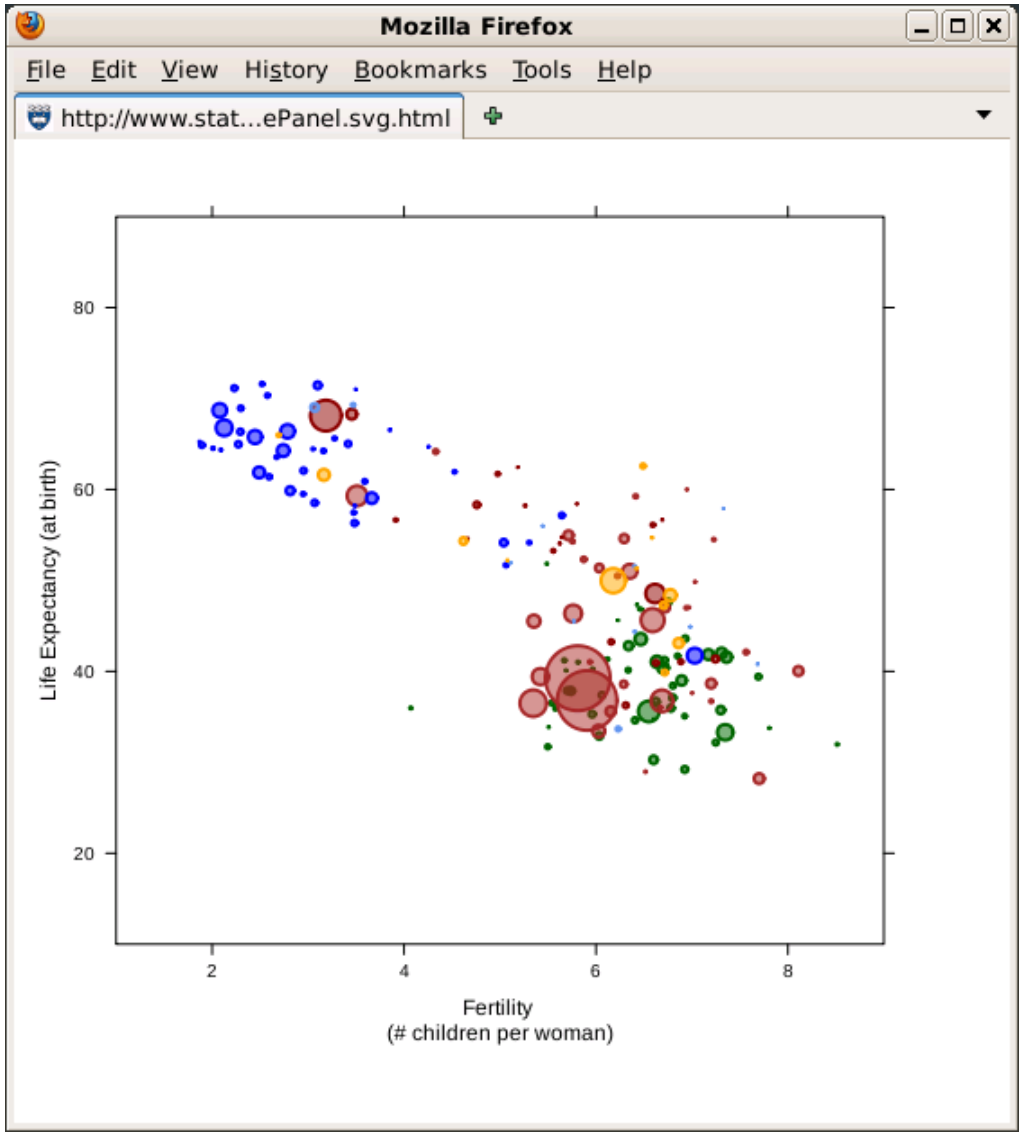
```
for (i in seq_along(sample)) {
  cg <- circleGrob(unit(x[sample[i]], "native"),
                   unit(.75, "npc"), r=unit(1, "mm"),
                   gp=gpar(col="red", fill=rgb(1, 0, 0, .2)))
  gcg <- garnishGrob(cg, visibility="hidden")
  acg <- animateGrob(gcg,
                     y=rep(c(.75, .55), c(i, Nsample - i + 1)),
                     visibility=rep(c("hidden", "visible"),
                       c(i - 1, Nsample - i + 2)),
                     duration=10)
  grid.draw(acg)
}
```

# The **gridSVG** package: Examples



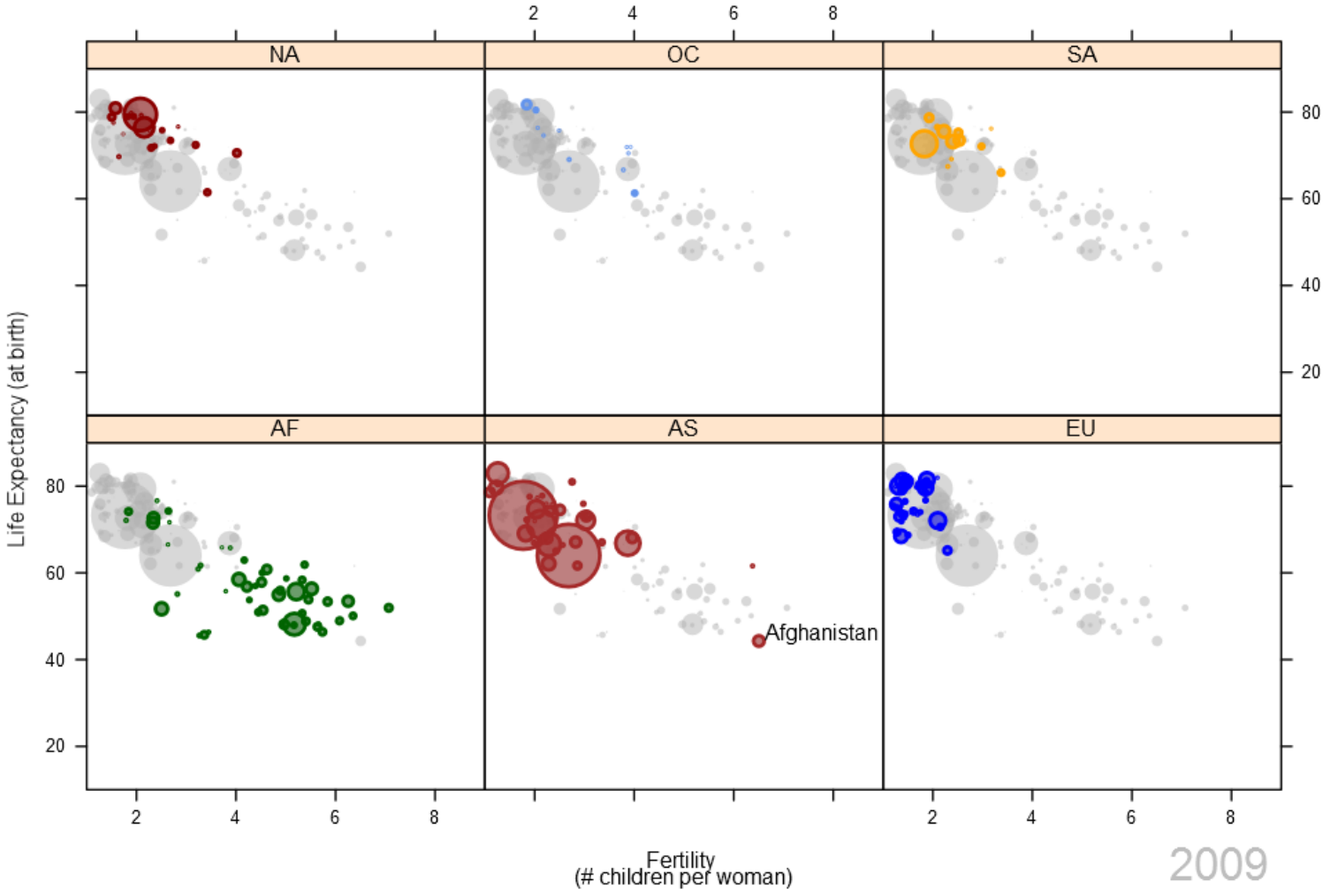

http://www.stat.auckland.ac.nz/~paul/Genentech2011/wildanim.svg.html

# The **gridSVG** package: Examples




http://www.stat.auckland.ac.nz/~paul/Velvet/gapminderOnePanel.svg.html

# The **gridSVG** package: Examples




http://www.stat.auckland.ac.nz/~paul/Velvet/gapminderMultiPanel.svg.html

# The **gridSVG** package: Examples

- Interactive scatterplots.

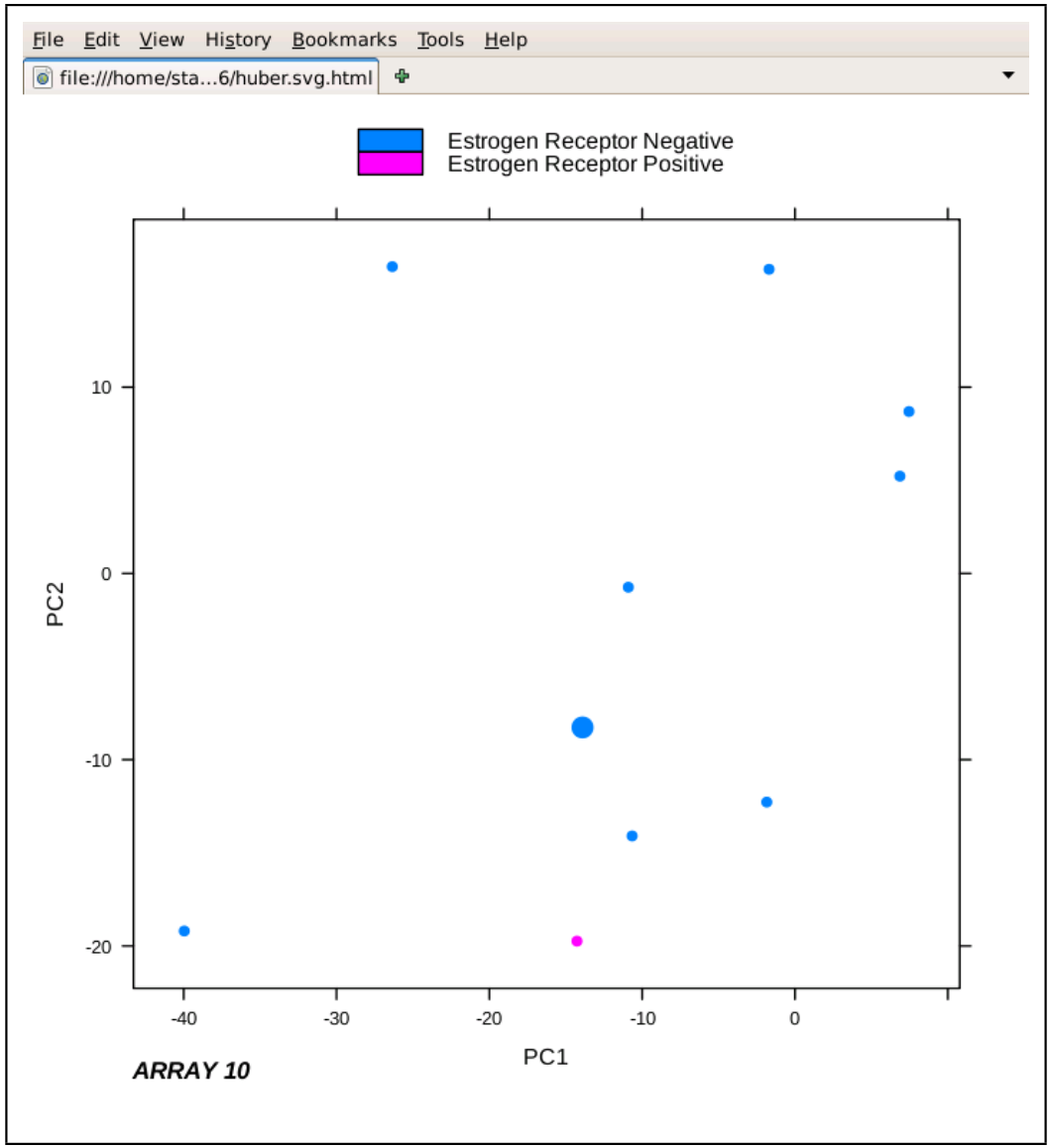
```
for (i in 1:10) {
  grid.garnish(paste("point", i, sep="."),
               onmouseover=paste('highlight(', i, '.1)',
                                 sep=""),
               onmouseout=paste('dim(', i, '.1)',
                                sep=""))
  grid.garnish(paste("label", i, sep="."),
               visibility="hidden")
}
```

# The **gridSVG** package: Examples

- Interactive scatterplots.

```
highlight = function(i) {
  var point = document.getElementById("point." + i);
  var label = document.getElementById("label." + i);
  point.setAttribute("r", point.getAttribute("r")*2);
  label.setAttribute("visibility", "visible");
}

dim = function(i) {
  var point = document.getElementById("point." + i);
  var label = document.getElementById("label." + i);
  point.setAttribute("r", point.getAttribute("r")/2);
  label.setAttribute("visibility", "hidden");
}
```

# The **gridSVG** package: Examples




http://www.stat.auckland.ac.nz/~paul/Genentech2011/huber.svg.html

# The **gridSVG** package: Examples

There are infinitely many other possibilities ...

- Integration with HTML 5 (e.g., HTML widgets as controllers).
- Post-processing via the **XML** package.
- R on the server.

# The **gridSVG** package vs **SVGAnnotation**

R graphics $\longrightarrow$ SVG $\longrightarrow$ SVGAnnotation $\longrightarrow$ SVG

R graphics $\longrightarrow$ gridSVG $\longrightarrow$ SVG

- With **gridSVG** the mapping to SVG is determined by **gridSVG**.
- With **SVGAnnotation** the mapping to SVG is determined by **Cairo**.
- With **gridSVG**, grob names map to SVG `id` attributes.
- With **gridSVG**, can target **groups** of objects (gTrees).

# The **gridSVG** package vs **SVGAnnotation**

```
> svg()
> grid.rect(name="myrect")
> grid.circle(name="mycircle")
> dev.off()
```

```
<path style="fill: rgb(100%,100%,100%);"
  d="M 0 504 L 504 504 L 504 0 L 0 0 Z M 0 504 "/>
<path style="stroke: rgb(0%,0%,0%);
  d="M 0 504 L 504 504 L 504 0 L 0 0 Z M 0 504 "/>
<path style="fill: rgb(100%,100%,100%);"
  d="M 504 252
     C 504 391.175751 391.175751 504 252 504
     C 112.824249 504 0 391.175751 0 252
     C 0 112.824249 112.824249 0 252 0
     C 391.175751 0 504 112.824249 504 252 "/>
<path style="stroke: rgb(0%,0%,0%);"
  d=" ... "/>
```

# The **gridSVG** package: Downsides

- Drawing the original image **and** exporting it will be slow (because R graphics and particularly **grid** graphics are slow)
- The package only works with graphics drawn using **grid** (includes **lattice** and **ggplot2**, but excludes a LOT of other stuff)
- The package by-passes the core graphics engine and simply emulates the normal behaviour; but it does NOT emulate the plotmath feature (yet)

# The **gridSVG** package: Future Directions

- The single-grob to multiple-elements problem
  - Animation targets multiple elements; garnishing targets the parent group; ideally, they would both target both.
- NOTE that lattice now has sensible names.
- Expand the animation interface.
- Javascript library (possible synergy with **SVGAnnotation**).

# Summary

- **gridSVG** is useful because ...
  - it produces SVG for the web (like **Cairo**) **and**
  - it allows addition of interaction (like **SVGAnnotation**) **and**
  - you can see how it works, so it can be extended (in more ways, by more people).

# References

- The packages **animation**, **rggobi**, **iplots**, **gWidgets**, **webvis**, and **gWidgetsWWW** can be found on CRAN
- The qtinterfaces suite of packages
  `http://qtinterfaces.r-forge.r-project.org/`
- The **googleVis** package
  `http://code.google.com/p/google-motion-charts-with-r/`
- The **SVGAnnotation** package
  `http://www.omegahat.org/SVGAnnotation/`
- The **gridSVG** package
  `https://r-forge.r-project.org/projects/gridsvg/`

# Acknowledgements


- Simon Potter added many new features to **gridSVG** version 0.7 during his BScHons project.

- The teaching animation is loosely based on teaching animations produced by Chris Wild.
- Two animation examples made use of data from the GapMinder Project `http://www.gapminder.org/data/`
- The interactive plot example is loosely based on output produced by Audrey Kauffmann and Wolfgang Huber's **arrayQualityMetrics** package.