7th China R Conf (Beijing), 2014-05-25

Interactive Visualization with R

王亮博 (亮亮)

Esc to overview

← → to navigate

Online slide on http://ccwang002.gitcafe.com/ChinaRConf-Interactive-Vis/

About Me

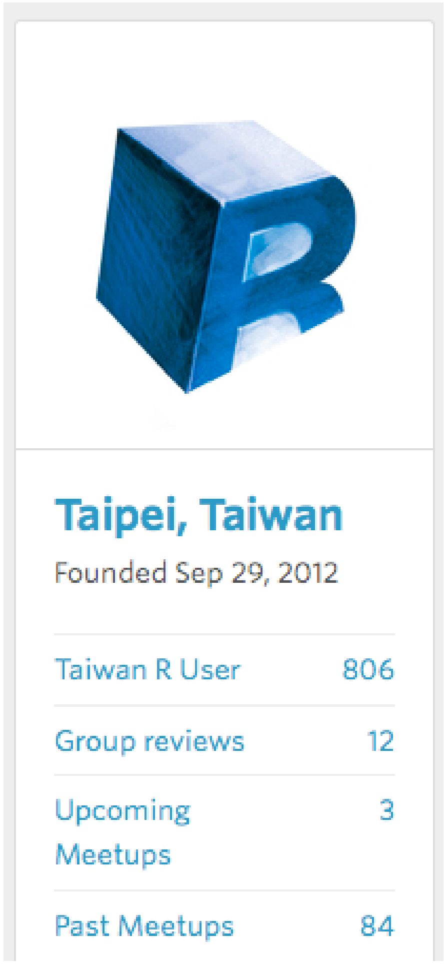
- Master student at

  Bioinfo & Biostat Core Lab, NTU CGM
- R / Python. Learning to speak DNA
- Taiwan R (MLDM) co-organizer
- PyCon APAC 2014 staff and speaker of
  - Statistics in Python with R
  - Handy Parallel(Distributed)

    Computing in Python

About Taiwan R User Group

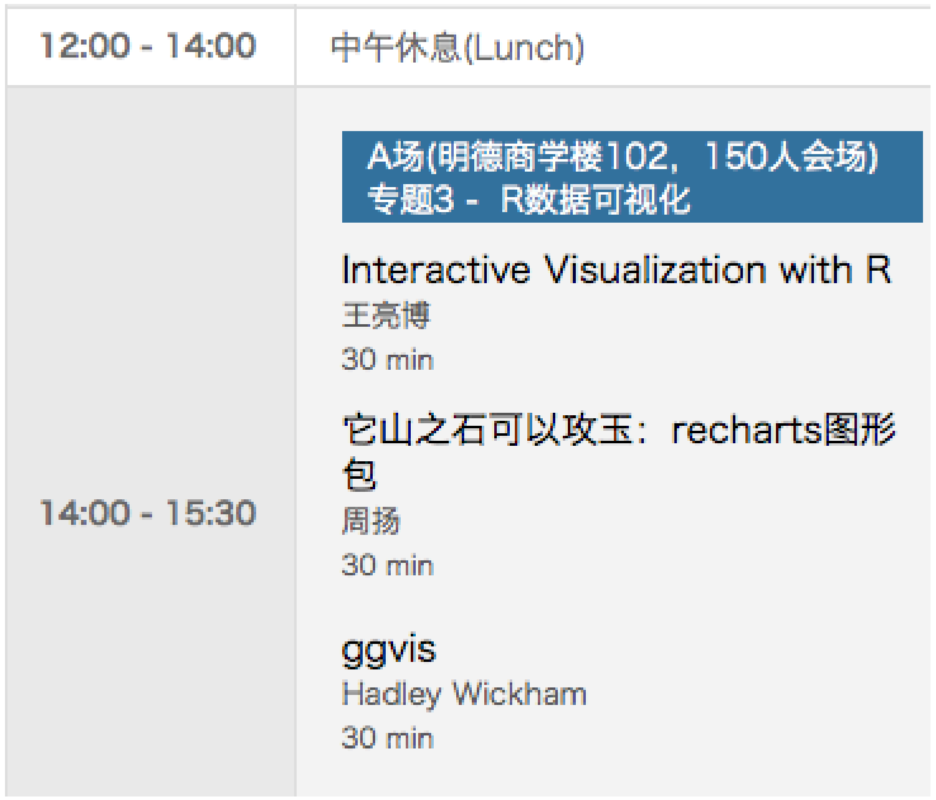
- More known a weekly meetup MLDM Monday (Machine Learning and Data Mining Monday)
- Topics ranges from
  - R lang: basic tutorial, Rcpp, quantmod, ggplot2, slidify, knitr, googleVis
  - Statistics, ML/DM: survival analysis, neural network, SVM, regression, nonparam. stat
  - Big Data: Hadoop, MPI
  - PyData: Numpy, Scikit-learn, pandas



Taipei, Taiwan

Founded Sep 29, 2012

Taiwan R User	806
Group reviews	12
Upcoming Meetups	3
Past Meetups	84



12:00 - 14:00	中午休息(Lunch)
14:00 - 15:30	A场(明德商学楼102，150人会场) 专题3 - R数据可视化 Interactive Visualization with R 王亮博 30 min 它山之石可以攻玉：recharts图形包 周扬 30 min ggvis Hadley Wickham 30 min

My honor to be the first in this section. This is an introductory talk.

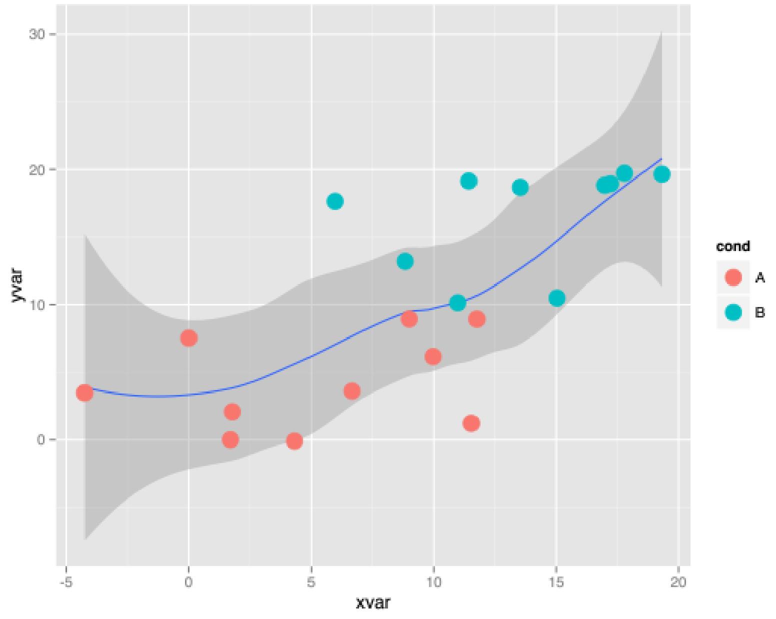
Topics

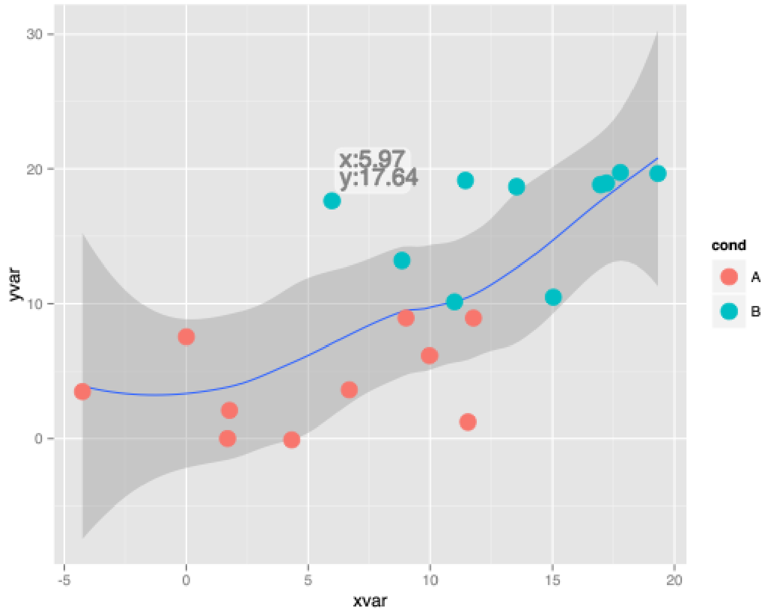
- Why interactive?
- How interactive in R?
- *SVG* intro
- Architecture of R graphics system intro
- R packages *grid* intro
- *gridSVG* intro
- Summary and limitation

Why and how?

From publication to *manipulation*

Demo from http://timelyportfolio.github.io/gridSVG_intro/





Why interactive?

- Re-train your model (change the parameters): *Shiny*
- Provide more details of your data: tooltip
- Data are collected in real time
- Provide different view point of your data
- We *just* want to be fancy

With today's method, we can reveal more details, provide different view points, and can be fancier :)

How interactive?

- In the past, one might first think of using GUI framework (QT, Gtk): *iPlot*
- Real pain for developers,
  - inexperience in GUI application devel
  - embedded other non-interactive information is hard
- Also a pain for users,
  - no need for another GUI application
  - users now mainly from internet

← Hope SVG here someday

Web and browsers dominates our front-end world.

Almost every PC and mobile have a modern browser today.

How interactive in R?

- Put everything on web (in the cloud)
- Use SVG to plot
- Mainly two ways:
  - Usual R plots → parse R plot object → output SVG → add interactivity on SVG
  - Use R lang to generate SVG directly

We take the first way in this talk.

Why SVG?

> *Scalable Vector Graphics (SVG) is an XML markup language for describing two-dimensional vector graphics.*
>
> ***Mozilla Developer Network***

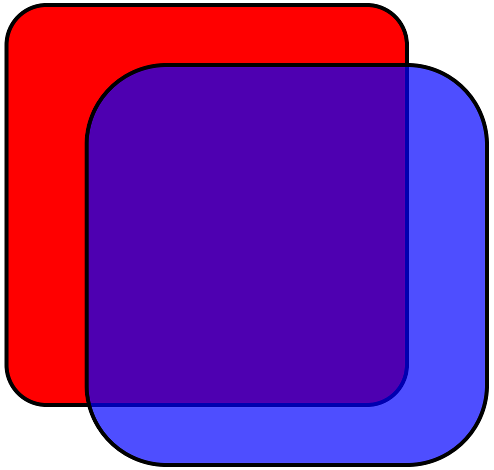
- Web standard widely supported by both desktop and mobile browsers
- Manipulate SVG elements by javascript and CSS; Animation is possible
- (not in R) most graphic tools can export to SVG

SVG Intro

SVG Intro

Full intro can be found on Mozilla Developer Network.

```
<svg version="1.1" width="300" height="300"
  xmlns="http://www.w3.org/2000/svg">
  <rect x="40" y="30" width="200" height="200" rx="20"
    fill="red" stroke="black" stroke-width="2" />
  <rect x="80" y="60" width="200" height="200" rx="40"
    fill="blue" stroke="black" stroke-width="2"
    fill-opacity="0.7" />
</svg>
```



SVG Basic Elements

- Rectangles: *rect*
- Circle: *circle*
- Ellipse: *ellipse*
- Line: *line*
- Polyline: *polyline*
- Polygon: *polygon*
- Path: *path*
- Group: *g*

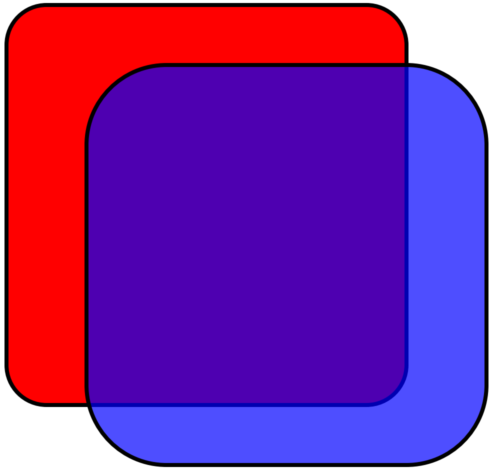
Attributes

- Position (0, 0) at topleft: *x y*
- Size: *width height*
- Stroke (color): *stroke stroke-width stroke-opacity*
- Fill (color): *fill fill-opacity*

But specify each element one by one is hard.

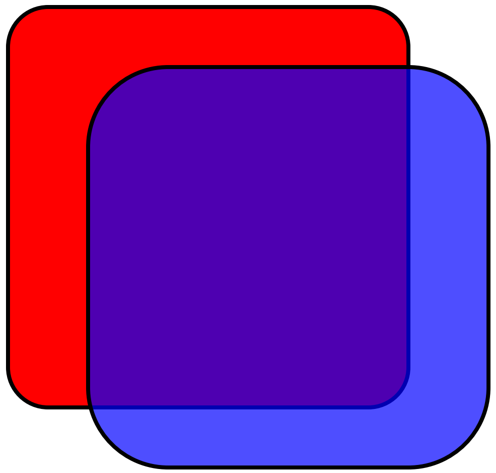
SVG style can be specified by CSS

```
<svg>
  <rect class="myrect" ... />
  <rect class="myrect" id="upper" ... />
</svg><style>
  .myrect {
    fill: red;
    stroke: black; stroke-width: 2px;
  }
  #upper {
    fill: blue; fill-opacity: 0.7;
  }
</style>
```



Use CSS3 interaction ability

```
<style>
  .myrect:hover {
    fill: white;
    stroke: green;
    stroke-width: 10;
    transition: 0.75s;
  }
</style>
```



So you get an interactive SVG!

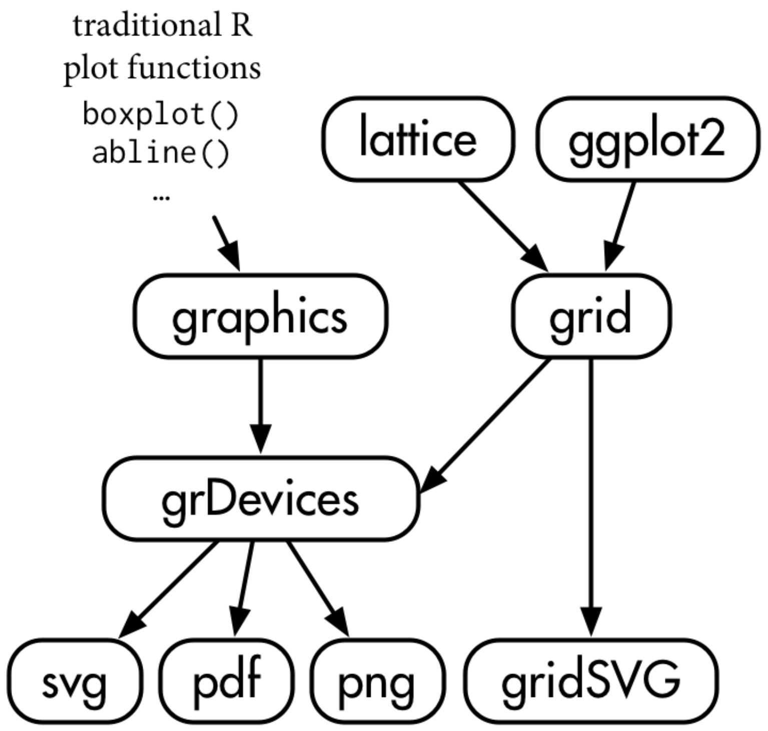
SVG Interaction Summary

- Build SVG plot from basic elements
- Label elements with class and id name
- Use CSS and JS to provide interaction or manipulation
- Embed your figure into a web page. *Done*
- We don't even need D3.js or other 3rd party tools here!
- For embedding problem, see comparison here

grid Intro

R Graphics Toolchain

- Adapted from [gridSVG project page]
- Today we focus on ggplot2 here
- ggplot2 builds on top of *grid*
- To export to SVG, one can through either
  - native *grDevices*
  - *gridSVG*



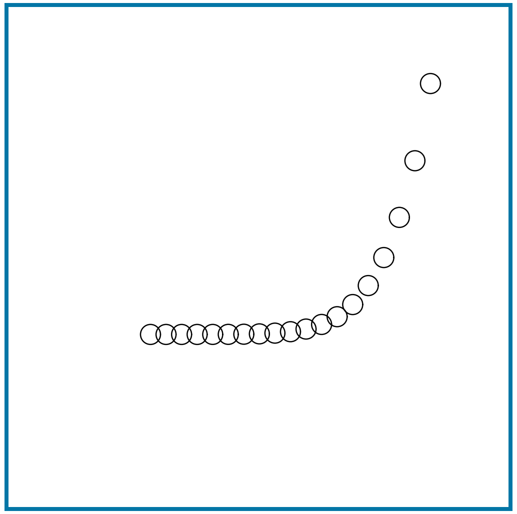
What's *grid*

- We stress two main component today:
  - Viewpoint (somewhat like *g* of SVG)
  - Plotting elements (e.g., points, rect, text)
- Every viewpoint has its coordinate system
- Viewpoint is buttom-up but g is top-down

Show the concept by a quick demo.

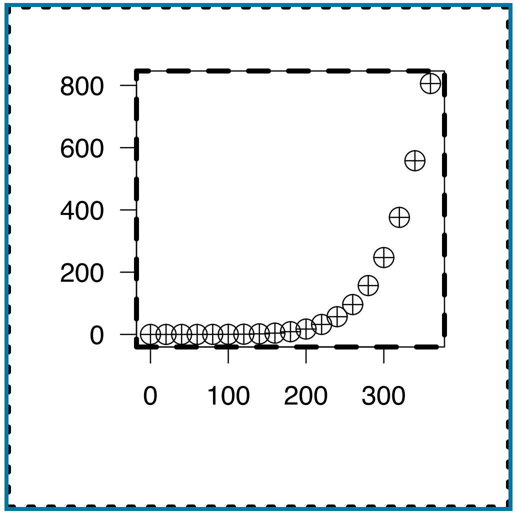
```r
library(grid)
grid.newpage()
pushViewport(plotViewport(c(5, 4, 2, 2)))
pushViewport(dataViewport(
  pressure$temperature, pressure$pressure,
  name="plotRegion"
))
grid.points(
  pressure$temperature, pressure$pressure,
  name="dataSymbols"
)  # upper figure

grid.rect(gp=gpar(fill=0))
grid.xaxis()
grid.yaxis()  # lower figure
```



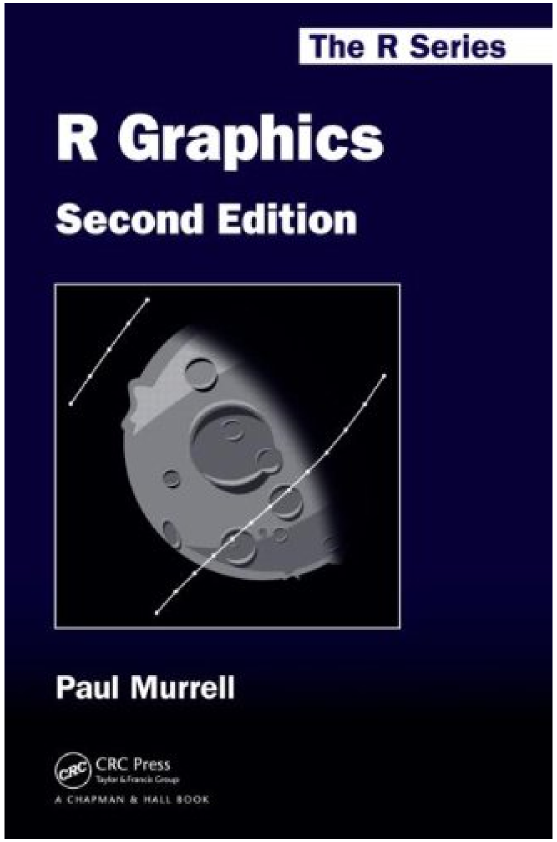
```
grid.edit("dataSymbols", pch=10)
upViewport(1)  # inner
grid.rect(gp=gpar(lty="dashed", fill=0))
upViewport(1)  # outer
grid.rect(gp=gpar(lty="dotted", fill=0))
# upper plot

downViewport("plotRegion")
grid.text(
  "Pressure (mm Hg)\nversus\nTemperature (Celsius)",
  just="right",
  x=unit(250, "native"), y=unit(600, "native")
)
# lower plot
```



Further Reading

- Cannot not fully cover *grid* today
- Follow ***R Graphics*** *2nd, Paul Murrel*
- Detailed illustration about traditional R plotting functions, grid system, lattice and ggplot2

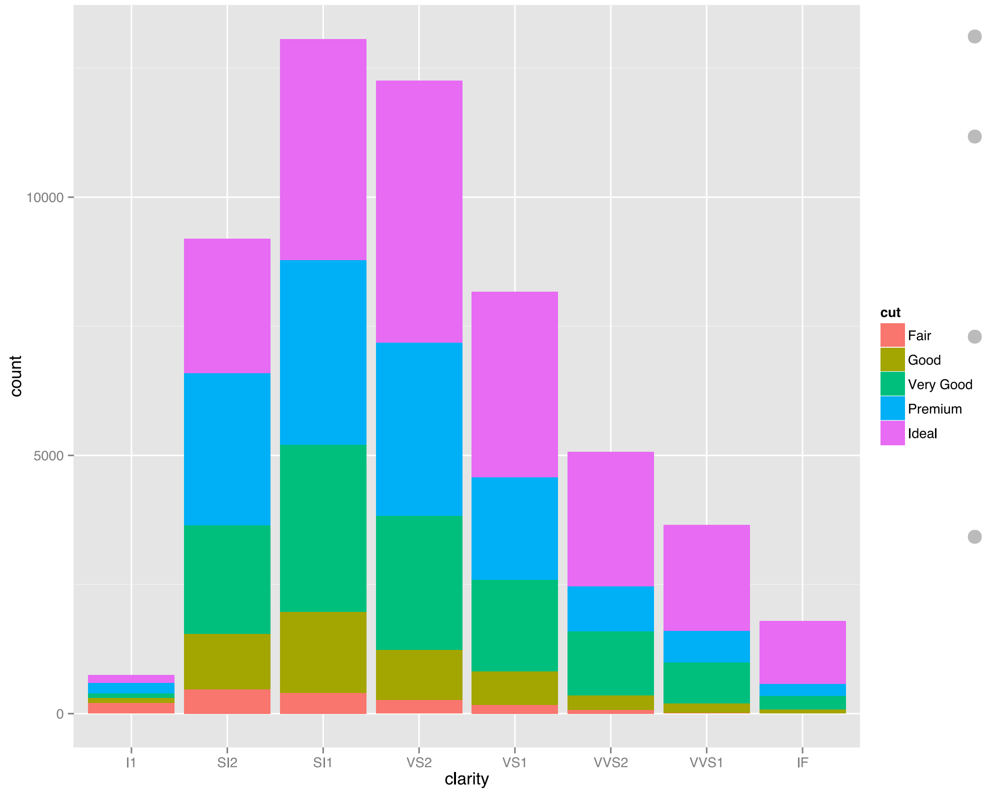


ggplot2 to SVG

Using *grid* to export to SVG is just a filename away.

```
require("ggplot2")
g <- qplot(clarity, data=diamonds, fill=cut, geom="bar")
ggsave(file="ggplot2_direct.svg",
       plot=g, width=10, height=8)
```

Direct SVG Result



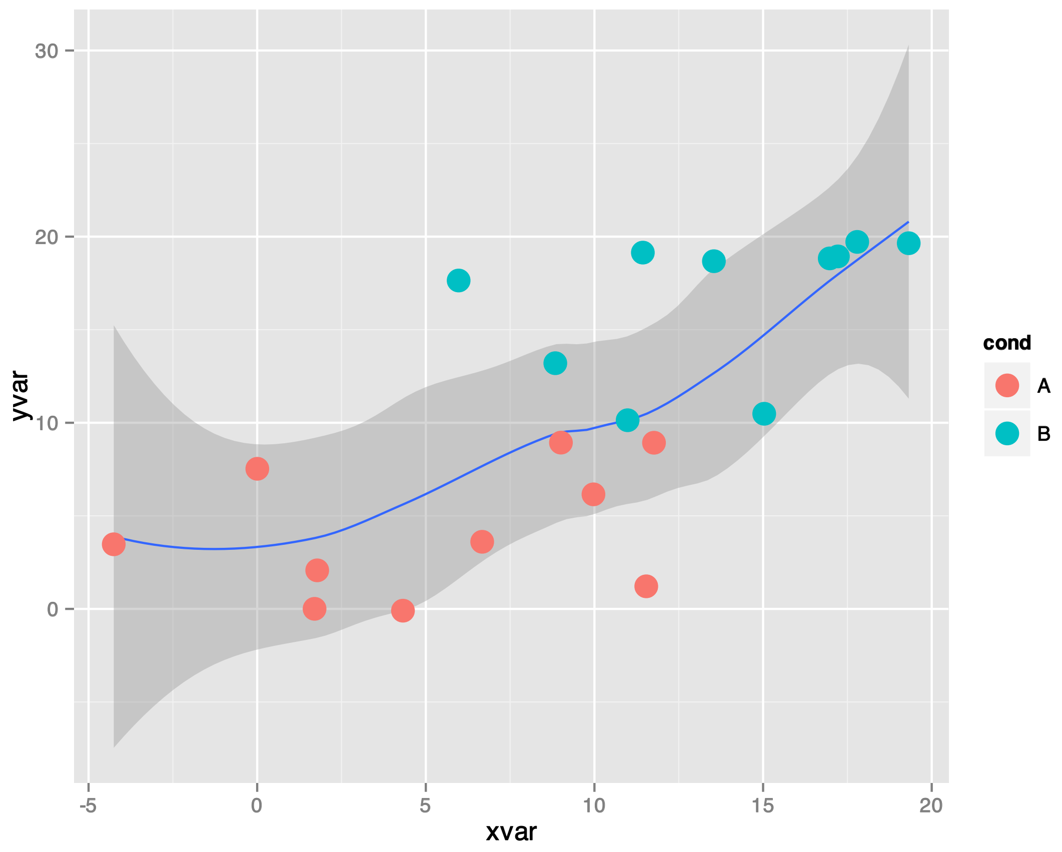
- Work like a charm
- Convert all words into outline
- Generated SVG loses original grid structure
- Hacking this SVG is slightly harder

girdSVG Intro

- gridSVG parse the grid structure then export to SVG directly
- Also, it provides some helper function to create animation
- Demo from gridSVG intro
- More examples hosted on Shiny by timelyportfolio

```r
g <- ggplot(...) + ...  # plot your ggplot2 here
g.svg <- grid.export("demo.svg", addClasses=TRUE)
```

(The text is selectable)



Summary

Our try on interactive visualization is ...

- Make SVG in R the hard way :)
- Utilize ordinal R(ggplot2, lattice) plots
- And make SVG interactive by hand adding custom CSS and JS
- Pretty much based on our knowledge about CSS and JS

Like this approach?

Limitation of this approach

- We are dealing with the *front-end*. Not R itself
- It is *hard*. Harder when you are dealing with chinese text and complicated coordinate system
- However, getting our hands dirty, we learn some fundamental architecture for R graphics
- What's *next*?

We mentioned two ways about interactive visualization in R.

Generate an interactive R plot directly

Package-dependent implementation.

All generates R plots through HTML, SVG, JS, CSS.

- rCharts: provide lattice-like interface
- googleVis: communicate with Google Chart API
- recharts: R interface to ECharts for data visualization
- ggvis: next-generation ggplot2 based on JS lib *Vega*, also facilitates HTML5 Canvas

Take home message

- Do interactive visualization on web
- How to write SVG on our own
- Get some insight about R graphics ecosystem
- Turn current *grid*-based R plots into SVG
- Future

Thank You!