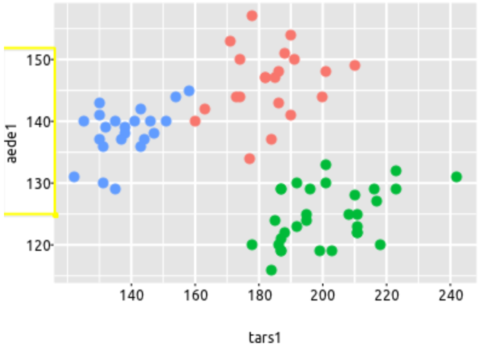
cranvas: Building from Plumbing and Painting

Yihui Xie

Department of Statistics, Iowa State University

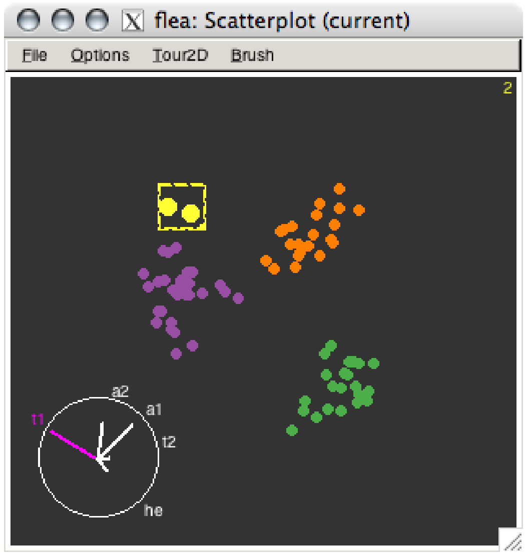


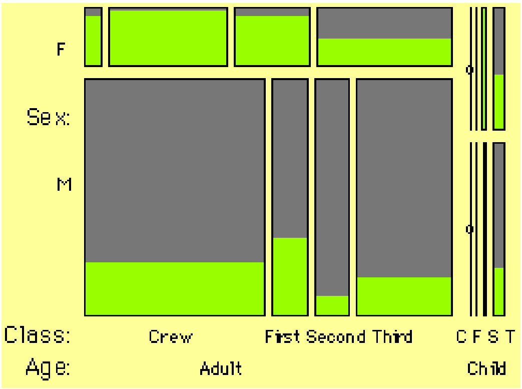
Demo

- old problems (with new look) in cranvas

Looking back

- standalone software packages: GGobi [Swayne et al., 2003], MANET [Unwin et al., 1996] and Mondrian [Theus, 2002], etc





- R itself has poor support for interactivity (ancient *locate()*? *getGraphicsEvent()*? no kidding! there is a **fun** package [Xie et al., 2011], and also **sudoku**)

- R packages: **iplots** [Urbanek and Wichtrey, 2011], **rggobi**, **playwith**, **rgl**...

Overview of cranvas

- the name (CRAN + canvas ⇒ cranvas)
- next generation of GGobi (and more)
  - interactive statistical graphics in R
  - focus on speed, flexibility and elegance
- infrastructures (details later):
  - painting (drawing): **qtbase** & **qtpaint** (connect R with Qt)

  - data structure: **plumbr** & **objectSignals** (mutaframes, reference classes, signals)
  - aesthetics: **scales** (and other **ggplot2**'s friends)

- all available at `http://github.com/ggobi` (most are on CRAN)
  - refer to the wiki for installation under Linux & Mac
  - Windows version not available at the moment

- usage
  - create a data object: `mf <- qdata(df)`
  - make a series of plots based on it: `qhist(x1, data = mf); qscatter(x2, x3, data = mf)`

- cranvas is still under active development
  - currently it is like an interactive version of R base graphics

Data structure: augmented data

```r
library(cranvas)
data(flea, package = "tourr")
head(flea)  # first 6 rows of flea
```

```
  tars1 tars2 head aede1 aede2 aede3  species
1   191   131   53   150    15   104 Concinna
2   185   134   50   147    13   105 Concinna
3   200   137   52   144    14   102 Concinna
4   173   127   50   144    16    97 Concinna
5   171   118   49   153    13   106 Concinna
6   160   118   47   140    15    99 Concinna
```

```r
str(flea)
```

```
'data.frame':   74 obs. of  7 variables:
 $ tars1  : int  191 185 200 173 171 160 188 186 174
   163 ...
 $ tars2  : int  131 134 137 127 118 118 134 129 131
   115 ...
 $ head   : int  53 50 52 50 49 47 54 51 52 47 ...
 $ aede1  : int  150 147 144 144 153 140 151 143 144
   142 ...
 $ aede2  : int  15 13 14 16 13 15 14 14 14 15 ...
 $ aede3  : int  104 105 102 97 106 99 98 110 116 95
   ...
 $ species: Factor w/ 3 levels "Concinna ","Heikert.
   ",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
qflea ← qdata(flea, color = species)
head(qflea)  # what is the difference?
```

```
  tars1 tars2 head aede1 aede2 aede3   species
1   191   131   53   150    15   104 Concinna
2   185   134   50   147    13   105 Concinna
3   200   137   52   144    14   102 Concinna
4   173   127   50   144    16    97 Concinna
5   171   118   49   153    13   106 Concinna
6   160   118   47   140    15    99 Concinna
  .brushed .visible  .color .border .size
1    FALSE     TRUE #F8766D #F8766D     4
2    FALSE     TRUE #F8766D #F8766D     4
3    FALSE     TRUE #F8766D #F8766D     4
4    FALSE     TRUE #F8766D #F8766D     4
5    FALSE     TRUE #F8766D #F8766D     4
6    FALSE     TRUE #F8766D #F8766D     4
```

- the augmented data contains information about each observation (row)
  - `.brushed`: logical vector – whether a row is brushed or not
  - `.visible`: similar – visible or not?
  - `.color`, `.border`, `.size` – aesthetics of graphical elements

Data structure: mutaframe

- what we created by *qdata()* was a mutaframe (using **plumbr**)
- a mutaframe is like a data frame, but is mutable (technically, it is an environment)

```r
str(qflea, max.level = 1)
```

```
Classes 'mutaframe', 'environment' <environment: 0
  x629d630>
 - attr(*, "col.names")= chr [1:12] "tars1" "tars2"
   "head" "aede1" ...
```

```
- attr(*, "row.names")= chr [1:74] "1" "2" "3" "4"
  ...
- attr(*, "changed")=Reference class 'Signal' [
  package "objectSignals"] with 7 fields
 ..and 18 methods, of which 7 are possibly relevant
- attr(*, "Brush")=Reference class 'BRUSH' [package
   "cranvas"] with 80 fields
 ..and 11 methods, - attr(*, "Link")=Classes '
   mutalist', 'environment' <environment: 0x6c87358
   >
 ..- attr(*, "envNames")= chr [1:3] "linkid" "
   linkvar" "type"
 ..- attr(*, "userNames")= chr [1:3] "linkid" "
   linkvar" "type"
- attr(*, "Scales")=Reference class 'Scales_meta' [
  package "cranvas"] with 15 fields
  ..and 11 methods, - attr(*, "Generator")= chr "
    d38bbe46dae5fa45758f3609f5dc1a0a"
```

- modification to a mutaframe is always “global”, which enables us to pass and modify a mutaframe anywhere in functions
  - application to tour (projections always changing, and the plot needs to know this)

```
a ← 10  # ordinary R object
b ← a
b ← 5
a  # certainly a is not modified
```

```
[1] 10
```

```
## but mutaframes are different
x ← qflea  # assign qflea to x
qflea$tars1[1]  # first element of tars1
```

```
[1] 191
```

```
x$tars1[1]  # same element in x
```

```
[1] 191
```

```
x$tars1[1] ← 300  # modify it
x$tars1[1]
```

```
[1] 300
```

```
qflea$tars1[1]  # original copy also modified!
```

```
[1] 300
```

Data structure: listeners

- how does interaction work?
  - interaction does not modify the plot directly
  - instead, it modifies data and triggers listeners on data, which do the job of updating the plot
  - it makes developers' life a lot easier
- what is a listener?
  - it is a function attached to a mutaframe

- (roughly speaking) whenever the mutaframe is modified, the function is executed

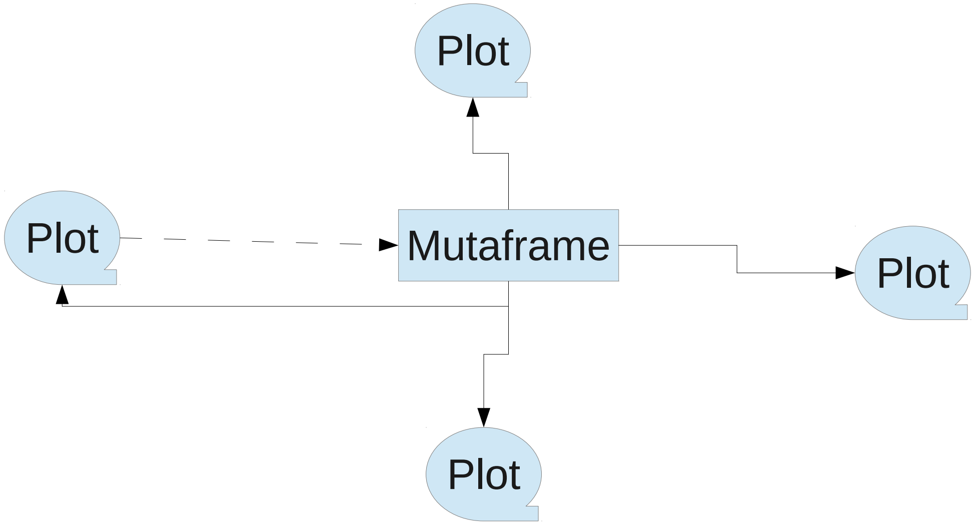
- which part of the mutaframe is modified?
  - we should not abuse the listeners, hence comes the concept of the "data pipeline"
  - `.brushed` changed? update the brush layer!
  - `.visible` changed? update the main plot layer!
  - original data changed? go back to the beginning and re-calculate everything, then update all the layers!

- each time we create a plot, a listener is attached to the mutaframe

  - so the `.brushed` column can update several plots at the same time

Data structure: a summary

a data object is sitting behind the scene; interactions change the data; listeners on the data update the plot(s)



Interaction: brushing or selection

- mouse interactions
  - left click and move the brush; right click to resize the brush; middle button to toggle between brush and selection
  - brush mode and identify mode (use ? to toggle)
- selection mode
  - `AND`, `OR`, `XOR`, `NOT`, `COMPLEMENT` (use the initial letters)
- related keyboard interactions

– `Delete` makes elements invisible; `F5`: all visible

Interaction: linking

- one-to-one linking
  - brush one observation in this plot, highlight the same observation in other plots
- categorical linking
  - ..., all observations in the same category are brushed too
- kNN linking

– ..., k nearest neighbors are brushed

Interaction: misc

- + and - changes the alpha transparency
- `PageDown` and `PageUp` navigates through brush history
- ...

Painting

- qtpaint
  - low-level plotting functions, e.g. qdrawCircle(), qdrawPolygon(), ...
  - graphics layers

Examples: plots in cranvas now, and what's new

- histogram (and spine plot): change binwidth with ↑ and ↓; shift bins with ← and →
- density plot: similar to histogram
- bar plot: application to missing value plot
- scatter plot: change size of points with ↑ and ↓

- boxplot: also show small boxplots for brushed observations inside original boxplots (not actually new)

- mosaic plot: dynamically change variables in the plot and layout

- parallel coordinates plot: rearrange order of variables (not new); show data ranges

- map: also cartograms

- time series plot: many, many new features (as a result of GSoC project)

Examples

- US 2012 presidential election

Limitations

- suffers from Qt flaws and bugs
  - mysterious clipping (aggressive: points clipped into halves; imprecise: no clipping even when points reach beyond boundary)
- Qt is big (in size) and installation of **qtbase** / **qtpaint** under Windows (64bit) can be difficult

Future plans

- automatic and *interactive* legend
- conditioning (faceting) like Trellis
- more types of plots (e.g. hexagons)
- a lot to learn from **ggplot2** and **iplots**

Acknowledgements

- Heike Hofmann, Di Cook, Xiaoyue Cheng, Tengfei Yin (current developers)
- Barret Schloerke, Marie Vendettuoli (former developers)
- Hadley Wickham, Michael Lawrence (all the hard work on infrastructure)
- Deborah Swayne

Questions & Comments?

- Testing and bug reports (`https://github.com/ggobi/cranvas/issues`) are welcome
- Thanks!

References

[Swayne et al., 2003] Swayne, D. F., Temple Lang, D., Buja, A., and Cook, D. (2003). GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43:423–444.

[Theus, 2002] Theus, M. (2002). Interactive data visualization using Mondrian. *Journal of Statistical Software*, 7(11):1–9.

[Unwin et al., 1996] Unwin, A. R., Hawkins, G., Hofmann, H., and Siegl, B. (1996). Interactive graphics for data sets with missing values – MANET. *Journal of Computational and Graphical Statistics*, 5(2):113–122.

[Urbanek and Wichtrey, 2011] Urbanek, S. and Wichtrey, T. (2011). *iplots: iPlots - interactive graphics for R*. R package version 1.1-4.

[Xie et al., 2011] Xie, Y., Wei, T., and Qiu, Y. (2011). *fun: Use R for Fun*. R package version 0.1-0.