iPlots eXtreme

Next-generation interactive graphics for analysis of large data

Simon Urbanek

AT&T Labs
Statistics Research

Overview

- About interactive graphics
- iPlots: next generation - why and how?
- New approaches
- Design and implementation (more at DSC)
- Example
- Summary

About iPlots

DEMO - using new iPlots eXtreme

- iPlots = Interactive Graphics for R
  - selection, highlighting, brushing ...
  - interactive change of plot parameters
  - queries
  - all essential plots (scatterplots, barcharts, histograms, parallel coordinate plots, mosaic plots, boxplots ..)
- Extensible framework
  - add your own objects (points, lines, text, polygons, ...)
  - create custom plots (define statistical objects in R)

(Java implementation available since 2003 - www.iplots.org)

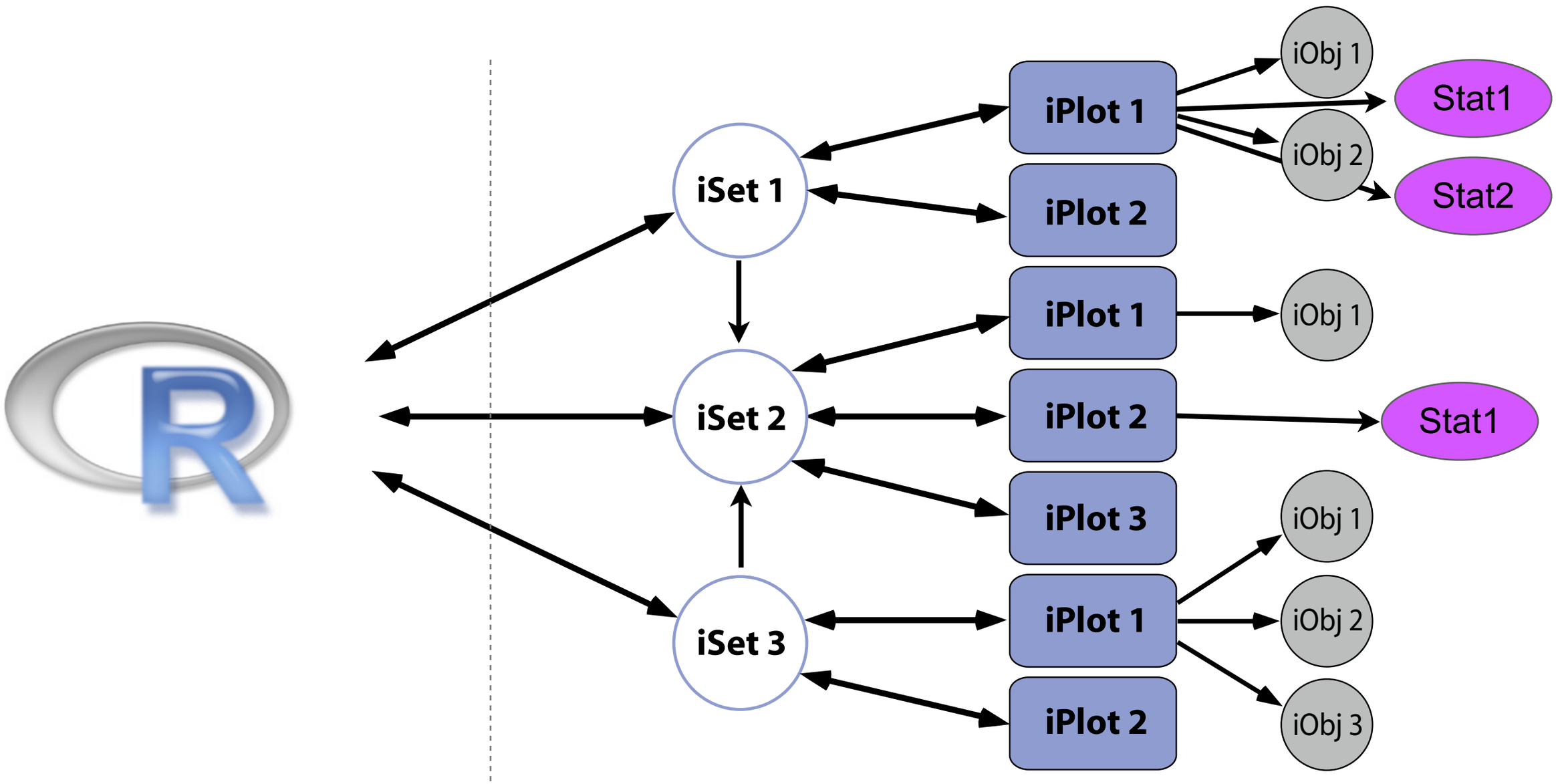
Next generation: Goals

- Support for large data
  - fast rendering (leverage modern GPUs via OpenGL)
  - native data structures (no copying from R)
  - fastest code possible (C++ subset, aggressively optimizing compilers)
- Integration
  - seamless integration in R GUIs
  - direct callback interface with R
- Clean user interface
  - learn from “clunkiness” of old iPlots and other IGs

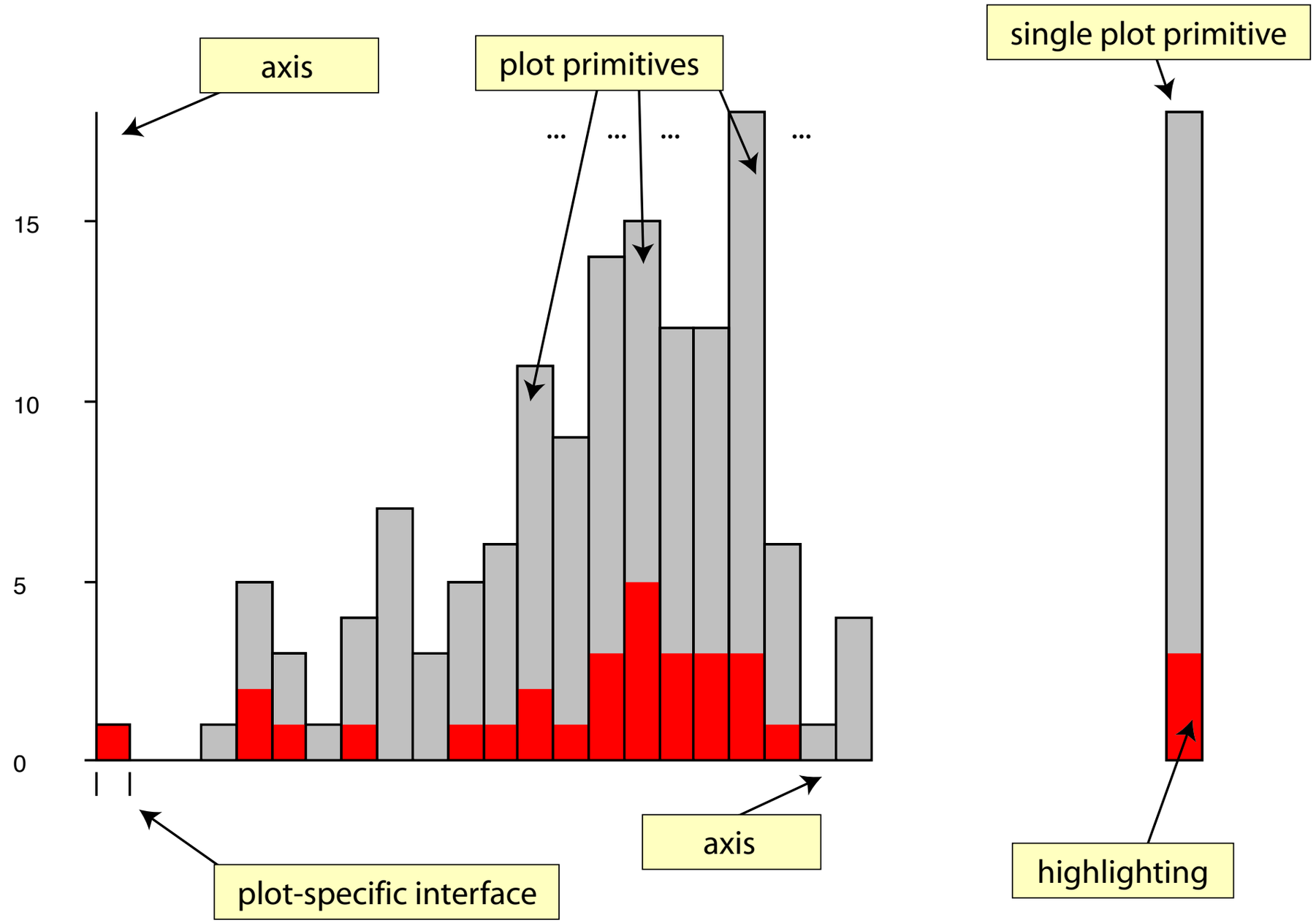
New research ideas *(in progress...)*

- Combine models and plots interactively
  - p = iplot(x, y) + lm(y ~ x)
    - creates a visual representation of the model in the plot
    - the representation is fully interactive:
      - supports queries, interprets selection
      - allows change of model parameters interactively
  - functional approach (the method is a function of plot type and model class) allows generalization and extensibility
- more Exploratory Model Analysis (EMA)

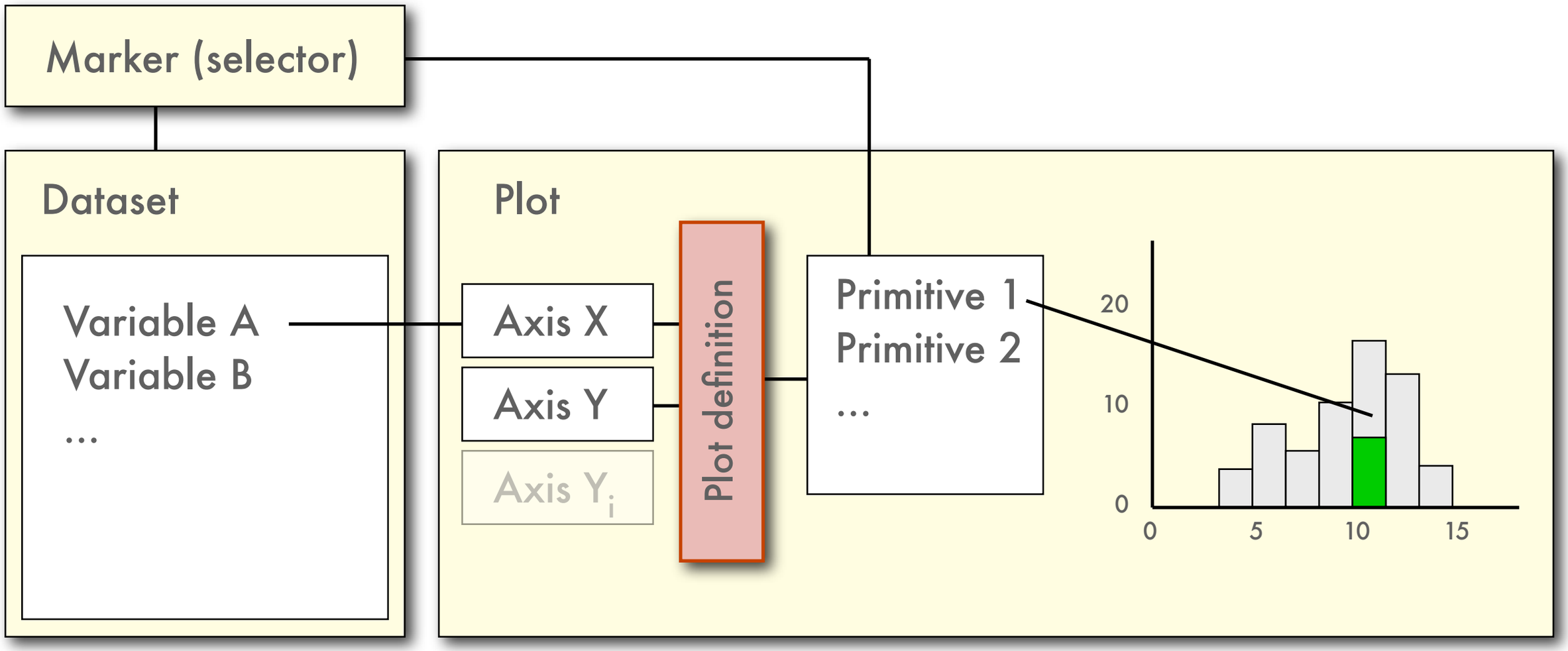
iPlots - Basic Design



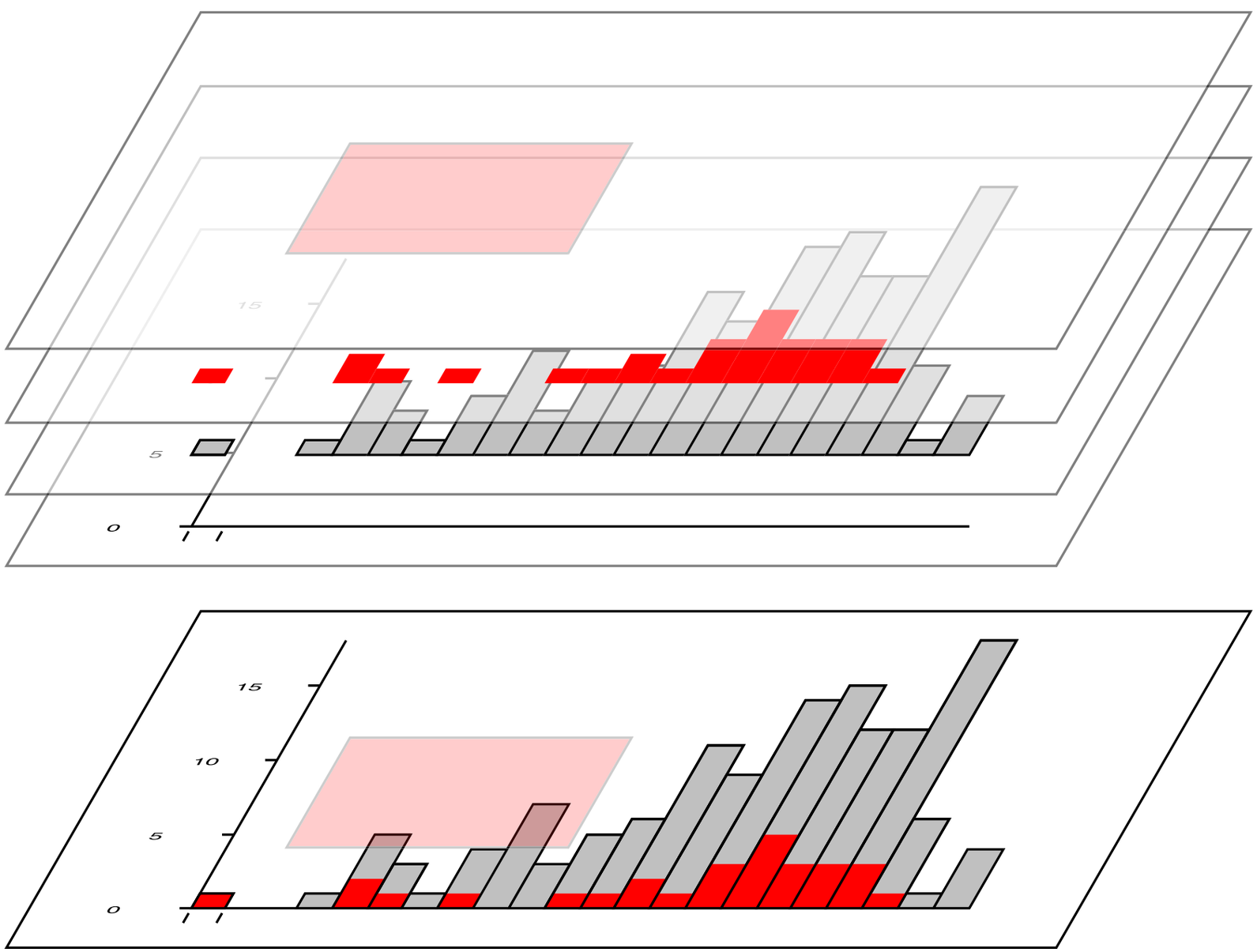
iPlots - Plot Design



iPlots - Design



iPlots - Layers



New in iPlots eXtreme design

- Plot objects can be visual primitives (graphics objects) or statistical primitives (linked to data)
- All primitives can have individual callbacks
- Allows multiple markers (e.g. 1:1, 1:n, m:n linking), no strict distinction between iSets
- R objects can have virtual attributes with direct access into C++ objects (e.g. `line$color = 1`, `histogram$bin.width = 0.1`)
- Reference-semantics storage (e.g. `plot$MyFoo`)

High-performance graphics back-end

- Can be used as R graphics device (very fast!)
- Supports double-buffering, delayed drawing (display when ready) and layers - controlled by R (great for animations)
- Exposes all interactivity to R (from mouse, keyboard level to selection, zoom etc.)
- Flexible layout facility for all components (R graphics, interactive plots, ...)

Implementation

- Complete re-write from scratch
- Uses a strict subset of C++ (no templates, MI, ...)
- Purely self-contained code (no STL, ...)
- Own object model (NeXT-like semantics, reference counting, debug-mode with RTTI)
- Cross-platform (purely OpenGL based + very thin platform-specific layer [Cocoa, Windows, GLUT, ...])
- Does not depend on a toolkit
- Can be used as a stand-alone application or R package or an application linked to libR

DEMO II

Conclusion

- Fast (C++, OpenGL: interactivity on >1 mio points)
- Efficient (no copying, reference semantics)
- Built-in support for interactive visualization of statistical models
- Extensible (custom visuals, statistical objects, plots)
- Combines all worlds in one package:

  Fastest R device, interactive graphics, OpenGL (3D)
- CRAN release: September 2009

  development code publicly available now

  http://RForge.net/Acinonyx/

Contact

- Simon URBANEK
  AT&T Labs-Research

  urbanek@research.att.com

www.iPlots.org