

## iPlots: Motivation

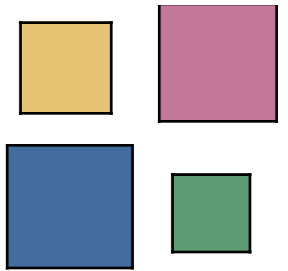
- Viele – teilweise extrem – interaktive Programme:
  - MANET
  - CASSATT
  - Klimt
  - Mondrian
- **Problem**

Oft ist eine Unterstützung durch parametrische Methoden sinnvoll  
Eine “Re-Implementierung” dieser Algorithmen ist aber nicht(!) empfehlenswert
- **Ansätze**

Verwende R um diese Berechnungen auszuführen:

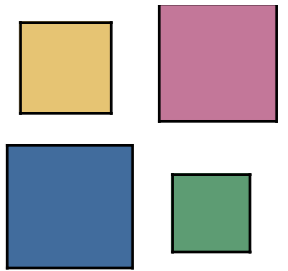
  - rpart-Bäume für Klimt
  - Scatterplot Smoother und Dichte Schätzer für Mondrian
  - ...
- **Alternative**

Ergänzung von R durch interaktive Graphiken



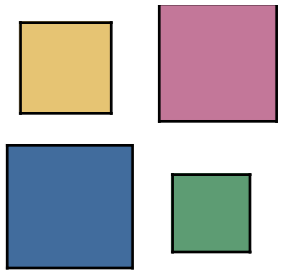
## iPlots: Einführung

- **R** Graphik ist absolut statisch: “Ink on Paper”-Konzept
- Interaktive Ergänzungen in **S** und **S-Plus** wie **spin(...)** und **brush(...)** wurden nicht für **R** portiert.
- **locator(...)** und **identify(...)** sind die einzigen “interaktiven” Funktionen in **R** Graphiken – daher ist eine Implementierung von ISG in R direkt nicht sinnvoll lösbar!
- **iPlots** bringen **Interaktive Statistische Graphik** für R.
- **iPlots** werden als R Paket angeboten und laufen als separater Java-Prozess parallel neben **R**.

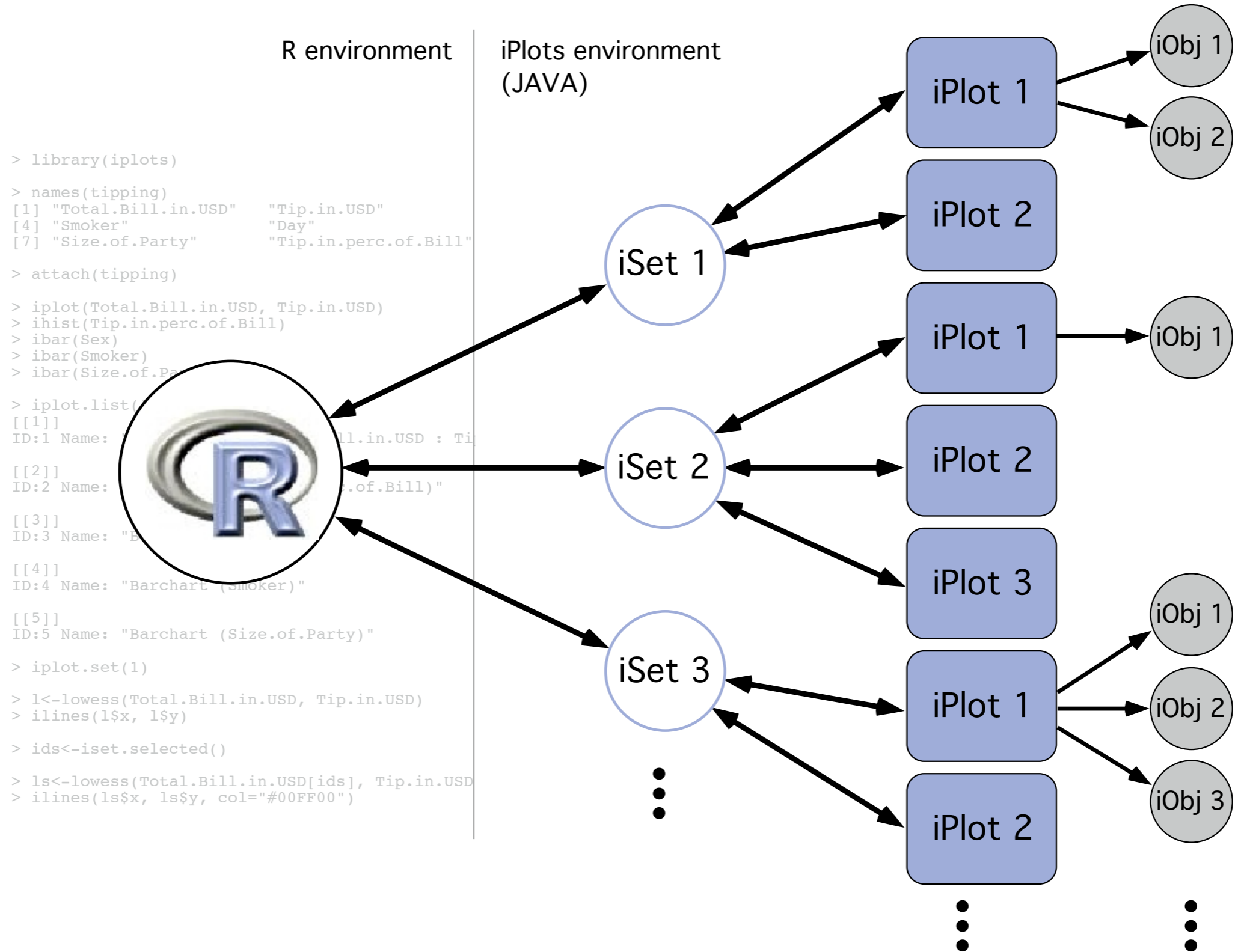


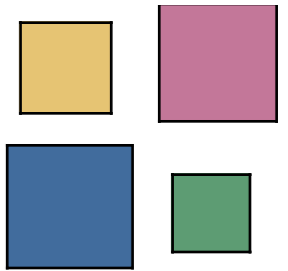
## iPlots: Design

- Grundidee in **iPlots**:  
Syntax und Handhabung soll sich so nah wie möglich und sinnvoll an den vorhandenen statischen Graphiken in R orientieren!
  - (i) Namen der Plots werden mit dem Präfix **i** versehen
  - (ii) Parameter sollen möglichst kompatibel bleiben
  - (iii) Plot Fenster werden über das Device Handling gesteuert
- Das Datenhandling in R erlaubt kein Hot-Linking, daher werden die in iPlots verwendeten Daten von iPlots selber verwaltet ⇒ **iSets**
- iPlots müssen – wie die statischen Graphiken in R – mit weiteren Komponenten (Linien, Punkte, ... ) versehen werden können.



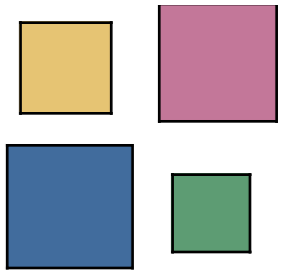
# iPlots: Design





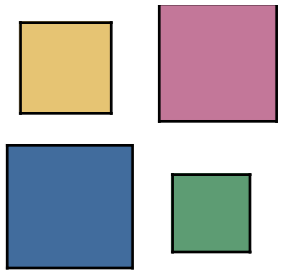
## iPlots: Kompatibilität

- Plots:
  - `plot` ⇒ `iplot` Scatterplot (nicht generisch wie `plot`!)
  - `hist` ⇒ `ihist` Histogramm
  - `barplot` ⇒ `ibar` Barchart (benötigt im Gegensatz zu `barplot` nicht `table()`)
- Fenster/Devices:
  - Wenn mehr als ein iPlot geöffnet ist, muss man zwischen den verschiedenen Fenstern (aus R heraus) wechseln können.
  - Die Verwaltung von iPlots Fenstern geschieht dabei wie die von Devices
    - `iplot.list()` liefert alle aktuell existierenden iPlot Fenster
    - `iplot.cur()` liefert den aktiven iPlot (per Default der letzte geöffnete)
    - `iplot.next(...)` und `iplot.prev(...)` liefern die Nummer des nächsten oder vorherigen iPlots relativ zum aktiven.
    - `iplot.set(...)` setzt den aktiven iPlot.
    - `iplot.off()` schließt das aktive Fenster.
    - `iplot.data()` liefert die Daten aus diesem iPlot



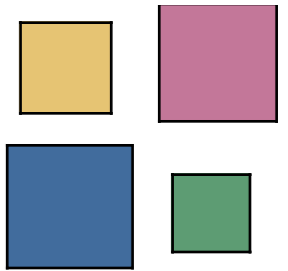
## iPlots: iSets

- R hat einige Defizite bei der Datenhandhabung:
  - Es gibt nur ein loses Konzept eines Datensatzes, für Linking benötigt man jedoch ein sehr strenges Konzept um Konsistenz zu gewährleisten.
  - Bei jeder Änderung von Daten in **R** wird **immer** eine Kopie der Daten erzeugt!
- Daher werden die Daten für **iPlots** auf Seiten der Java VM in sog. **iSets** gespeichert.
- Plots werden immer eindeutig einem **iSet** zugeordnet
- Alles Plots die zum gleichen **iSet** gehören sind gelinkt
- **iSets** werden automatisch generiert, immer wenn ein iPlot erzeugt wird, der nicht zum aktuellen iSet gelinkt werden kann.
  - es existiert noch kein iSet
  - die Anzahl der Daten passt nicht zum aktuellen iSet
  - der Benutzer erzeugt explizit eine neues iSet



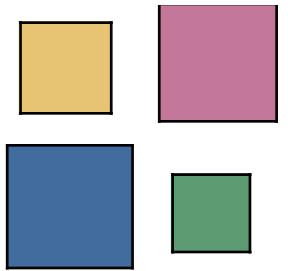
## iPlots: iSets

- Auch **iSets** können über die üblichen Verwaltungsfunktionen bearbeitet werden:
  - `iset.cur()` Aktives iSet
  - `iset.list()` Liste aller iSets
  - `iset.new(...)` Erstellen eines neuen iSets
  - `iset.next(...)` Nächstes iSet
  - `iset.prev(...)` Vorheriges iSet
  - `iset.set(...)` Bestimmung des aktiven iSets
- Wichtig sind die Funktionen zur Verwaltung der selektierten Daten:
  - `iset.select(...)` explizites Setzen der Selektion
  - `iset.selected()` Menge der selektierten Fälle (Integer Vektor)
  - `iset.sel.changed()` **true** oder **false** falls sich die Selection geändert hat
- Color Brushing wird auch über iSet Funktionen realisiert
  - `iset.col(...)` Definition der Farben über die Integerwerte eines Faktors
  - `iset.brush(...)` Synonym (?)



## iPlots: iObjects

- Alle iPlots können mit (statischen) graphischen Objekten versehen werden (im Augenblick):
  - `iabline(...)` fügt eine gerade Linie in den iPlot ein
  - `ilines(...)` fügt einen Polygonzug in den iPlot ein
  - `itext(...)` fügt Text in den iPlot ein
- Auch hier gelten die üblichen Verwaltungsfunktionen
  - `iobj.cur()` Aktives Objekt
  - `iobj.get(...)` Liefert das spezifizierte Objekt
  - `iobj.list()` Liste aller Objekte
  - `iobj.next(...)` Nächstes Objekt in der Liste
  - `iobj.opt(...)` Setzt Eigenschaften des Objekts
  - `iobj.prev(...)` Vorheriges Objekt
  - `iobj.rm(...)` Löschen des aktiven Objekts
  - `iobj.set(...)` Setzen des aktiven Objekts



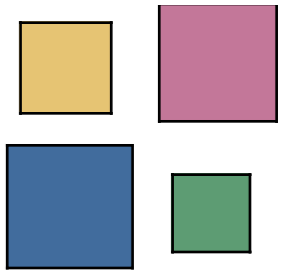
## iPlots: Plots in v0.2

In der aktuellen Version sind die folgenden Plots implementiert:

- **ibar** – interaktiver Barchart  
Verwendet als Input keine Tabelle wie **barplot(...)**, sondern die Daten direkt  
Wahlweise Darstellung als Spineplot
- **ihist** – interaktives Histogramm  
Binbreite und Ankerpunkt können interaktiv geändert werden.
- **iplot** – interaktive Scatterplot

Bis Ende des Jahre werden ergänzt:

- **imosaic** – interaktiver Mosaic Plot
- **iparcoord** – interaktive Parallele Koordinaten Plots



## iPlots: Ausblick

- Das aktuelle Interface zu R bietet “nur” fertige, abgeschlossene Plots an
- “Echte”, interaktive Objekte wie
  - Punkte
  - Boxen, und
  - Linienerlauben das Erstellen von neuen, frei definierbaren interaktiven Plots
- Jedes dieser interaktiven Elemente bezieht sich auf eine Menge von Punkten im iSet, und kann
  - selektiert, und
  - gehighlighted werden
- Ein Histogramm ist dann “nur noch” eine Ansammlung von interaktiven Boxen, die linear aufgelistet werden.