統計数理(2007)
第 55 巻 第 1 号 27–45
ⓒ2007 統計数理研究所

特集「統計データの可視化」
［統計ソフトウェア］

デザインパターンを用いた統計グラフのための Java ライブラリ

山本 由和[1] ・中野 純司[2,3] ・本多 啓介[3]

(受付 2006 年 8 月 2 日；改訂 2007 年 3 月 6 日)

要　　旨

本稿では，現代的な統計グラフライブラリに必要な条件を考察するとともに，それらを満足させる実装としてわれわれが開発している統計グラフライブラリ Jasplot (**Ja**va **s**tatistical **plot**) の詳細を説明する．Jasplot は，マウスによる対話的な操作を実現しており，基本的なグラフやグラフを構成する部品を組み合わせることによって新しいグラフを作ることを可能にしている．また，R や Jasp のような既存のシステムから利用しやすくなっている．このような機能を実現するために，プログラムの設計においてデザインパターンの考え方を積極的に利用したことがその特徴となっている．

キーワード： 対話的な操作，統計グラフライブラリ，Java，linked views.

1. はじめに

最近では，多くの統計グラフは計算機によって描かれている．そして，計算機技術の進歩によって，多くの色を使用した美しいグラフや対話的なグラフ，さらに，3 次元表示のグラフなどもパーソナルコンピュータを利用して簡単に描けるようになっている．例えば，Microsoft Excel は，シートに保存されているデータを簡単に美しいグラフとして表示することができる．また，Gnuplot (`http://www.gnuplot.info/`) は，ファイルに保存されている数値データを簡単にグラフ表示することができる．

現代の統計解析ソフトウエアは，そのようなグラフ表示機能を基本機能の 1 つとして持っている．世界中で広く使われている統計解析ソフトウエアの 1 つである R は，特に高度なデータ視覚化関数を持っている (Murrell, 2005)．

ただ，そのような視覚化関数でも，まだ不十分な場合も多くなってきている．近年，さまざまな分野において計算機を利用した自動的なデータ収集システムが利用されるようになったために，解析すべきデータ量が飛躍的に増加しており，そのようなデータについてはこれまでのグラフではその特徴をうまく表示できないことが多い．例えば，すべての観測値を散布図や平行座標プロットで描いた場合，点や線が重なるために，各観測値を区別することが難しくなる．このような問題を解決する方法として，focusing，brushing，zooming，linked views のような対話的な操作が提案されている (Symanzik, 2004) が，これらは R の標準的な関数の中には含ま

[1] 徳島文理大学 工学部情報システム工学科：〒769–2193 香川県さぬき市志度 1314–1
[2] 統計数理研究所：〒106–8569 東京都港区南麻布 4–6–7
[3] 総合研究大学院大学 複合科学研究科統計科学専攻：〒106–8569 東京都港区南麻布 4–6–7

れない．ただし，R のパッケージとして配布されている iPlots (http://rosuda.org/iPlots/) を利用することによって対話的なグラフを表示できる．iPlots に含まれる関数によって，マウスによる操作や linked highlighting などの機能を利用できるグラフを描いたり，グラフに線や文字を追加することなどができる (Urbanek and Theus, 2003)．iPlots は，Java 言語によって実装されているが，Java 言語のレベルでの変更や新しいグラフの定義や機能を追加することは容易なことではない．

R のような統計解析用のプログラミング言語では実現が難しいような新しいグラフや複雑な対話機能を利用したい場合や既存のシステムに対話機能を持ったグラフ表示機能を追加したい場合には，C++ や Java のような一般的なプログラミング言語を利用することが多い．そこで，このような対話的な処理を実現し，また，新しいグラフを簡単に構成できるようなライブラリが必要となってくる．

対話機能を統計グラフ上で実現するには，統計の数値計算を行うプログラムを書く場合よりも，オペレーティングシステムや表示デバイスなどの計算機に関する知識をより多く必要とする．そして，ユーザがグラフに対して行ったマウス操作を処理するためには，マウス操作をイベントとして受取り，その操作に対してグラフの再描画を適切に行うプログラムを書かなければならない．これは簡単な作業ではない．そこで，このような処理を比較的容易にするようなライブラリが利用できれば，統計グラフのためのプログラミング作業を軽減することができる．例えば，JFreeChart (http://www.jfree.org/jfreechart/) は，このような目的のための Java クラスライブラリである．JFreeChart を利用すれば，用意されているグラフに対するオプションの設定などによって，さまざまな美しいグラフを描くためのアプリケーションプログラムや Web ブラウザ上にグラフを表示する Web アプリケーションなどを開発できる．しかし，マウスで指定した領域を拡大するような対話的な操作は簡単に実現できるが，データを選択したり，Java のクラスを組み合わせたり新しくクラスを定義したりして新しいグラフを開発するようなことは容易ではない．

そこで，われわれは，汎用的な統計グラフライブラリ Jasplot (**Ja**va **s**tatistical **plot**; http://jasp.ism.ac.jp/Jasplot/) を開発した．Jasplot は，マウスによる対話的な操作をサポートしており，R や Jasp (http://jasp.ism.ac.jp/; Nakano et al., 2000, 2004) のような既存のシステムに組み込みやすく，基本的なグラフやグラフを構成する部品を組み合わせて新しいグラフを作りやすいという特徴を持っている．これらの特徴は，デザインパターンの考え方 (Gamma et al., 1995; Martin, 2003) を積極的に取り入れることで効果的に実現できる．

Jasplot の設計の際に考慮した統計グラフライブラリのための基本的な考え方については，次章で説明する．これらの考え方を利用した Jasplot の設計と実装についての詳細を 3 章で説明する．4 章では，簡単な利用例を紹介する．

2. 統計グラフライブラリに必要な要件

現代の統計グラフライブラリは，以下のようなことを実現していることが望ましい．

2.1 対話的な操作

データが多量にある場合には，静的なグラフだけでは不十分で，対話的な操作が必要不可欠である．例えば，データのある特徴を知るために，特定の観測値だけに注目して，それを強調表示したいことがある．これには，近くに位置する観測値のグループを選択するための brushing 技術が利用される．その他にもいくつかの対話的な操作が提案されている (Symanzik, 2004)．

現在，そのような対話的な操作にはマウスを利用するのが一般的である．さらに，他の機能からも対話的な操作を簡単に利用できなければならない．そのためグラフライブラリは，基本

的な機能としてマウス操作を持たなければならない．しかし，このような機能を一般的なプログラミング言語によって実装することは容易ではない．そのため，統計グラフライブラリを使用することによって，できるだけ簡単に実装できることが望ましい．

2.2 基本的な統計グラフ

統計学では，多くのグラフが提案されているが，それらすべては，紙やスクリーンというような 2 次元デバイス上に描かれる．最も基本的な 2 次元グラフは，散布図である．これは，連続 2 変数をもつ観測値を 2 次元デカルト座標面上に点として表示したものである．観測値を表す点を線分で結んだり，それらの線分で指定される特定の領域をある色で塗りつぶしたりすることもある．グラフは，点や線分や多角形で描かれるので，形式的にはすべてのグラフを散布図の拡張形と考えることもできる．連続 1 変数データに対しては，ヒストグラムが最も基本的である．また，箱ひげ図も 1 変数毎に分布の特徴を表示することができる．

統計グラフライブラリには，少なくともこのような基本的なグラフは含まれていなければならない．そして，それらは簡単なプログラムで利用できなければならない．

2.3 新しいグラフの作成

基本的なグラフの組合わせによって，新しいグラフを作成しなければならないことがある．例えば，多変量連続データに対して散布図を行列のように配置した散布図行列は，有用なグラフである．なお，散布図行列では，対角線上にヒストグラムが描かれる場合もある．

統計グラフライブラリは，より基本的な部品である軸や文字列などを細かく組み合わせて，新しいグラフをできるだけ容易に構築できる必要がある．

2.4 Linked views

1 つのデータに対して，複数の統計グラフを描いてその特徴を発見しようとすることは，一般的な方法である．この場合，対話的な操作がそれらの統計グラフで連携できれば特に有効である．この機能は，linked views といわれる．その簡単な例は散布図行列において，1 つの散布図である観測値のグループを強調すると，他の散布図においても同じ観測値が強調される機能である．この機能を linked highlighting と呼ぶ．さらに強調するデータを連続的に変化させることを linked brushing と呼ぶ．

統計グラフライブラリは，このような複数グラフの連携をサポートするべきである．

2.5 データアクセスと出力形態

統計グラフライブラリでは，データをライブラリに渡さなければグラフを描くことはできない．そして，統計グラフライブラリの簡単な利用法は，ファイルに保存されているデータを読み込んでグラフ表示するようなプログラムを書くか，既存のシステム内のデータをグラフ表示するためにそのシステムに組み込むかであろう．

あるフォーマットに従って，ファイルに保存されているデータを読み込むためには，ライブラリはそのフォーマットに対応していなければならない．よく利用される CSV（Comma Separated Values）形式や Microsoft Excel 形式のファイルを読み込むための機能は，ライブラリが持っていることが望ましい．さらに，これら以外の保存形式にも柔軟に対応できる拡張性も必要である．

既存のシステムに組み込む場合には，そのシステムにおけるデータとグラフライブラリのデータの間の双方向の変換が必要である．システムのデータ構造は多様であるため，ライブラリはどのようなデータ構造にも柔軟に対応できなければならない．

さらに，その出力形態として，コンピュータネットワーク環境において利用する Web アプリケーションなども考えられる．統計グラフライブラリは，さまざまな環境で結果を出力でき

ることが望ましい.

3. Jasplot の設計と実装

Jasplot は，汎用的な統計解析ソフトウエア Jasp（**Ja**va **s**tatistical **p**rocessor）のために開発が開始された．その後，Jasplot は，Jasp に依存していたバージョンから大幅に機能が拡張され，対話的な統計グラフを描くための一般的な Java ライブラリとして開発されている．現在では，簡単な Java プログラムによってファイルからデータを読み込んでグラフを表示するために利用できるとともに，Jasp だけではなく他の Java 環境，例えば，rJava（`http://rosuda.org/rJava/`）を利用した R などに組み込むことができる．

Jasplot は，基本的なグラフとして，散布図（図 1），ヒストグラム，箱ひげ図（図 2）を描くことができる．このような基本グラフを組み合わせることによって，新しいグラフを作ること

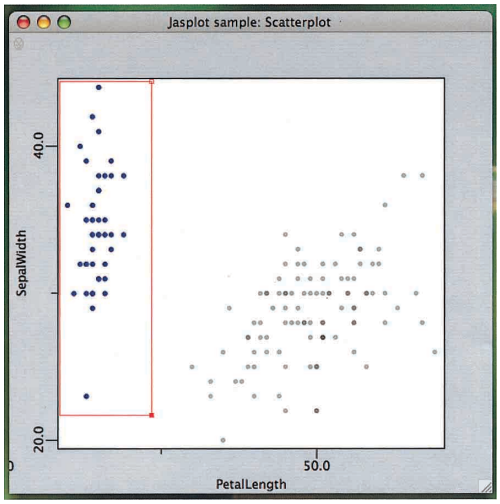

図 1. 散布図.

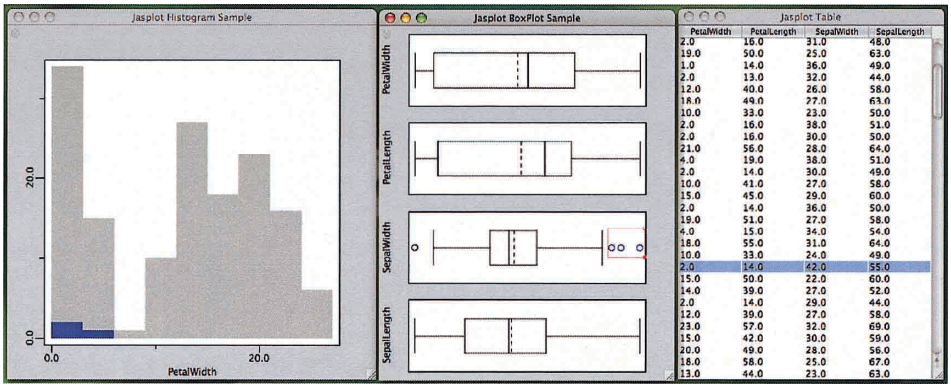

図 2. ヒストグラムと箱ひげ図の例.

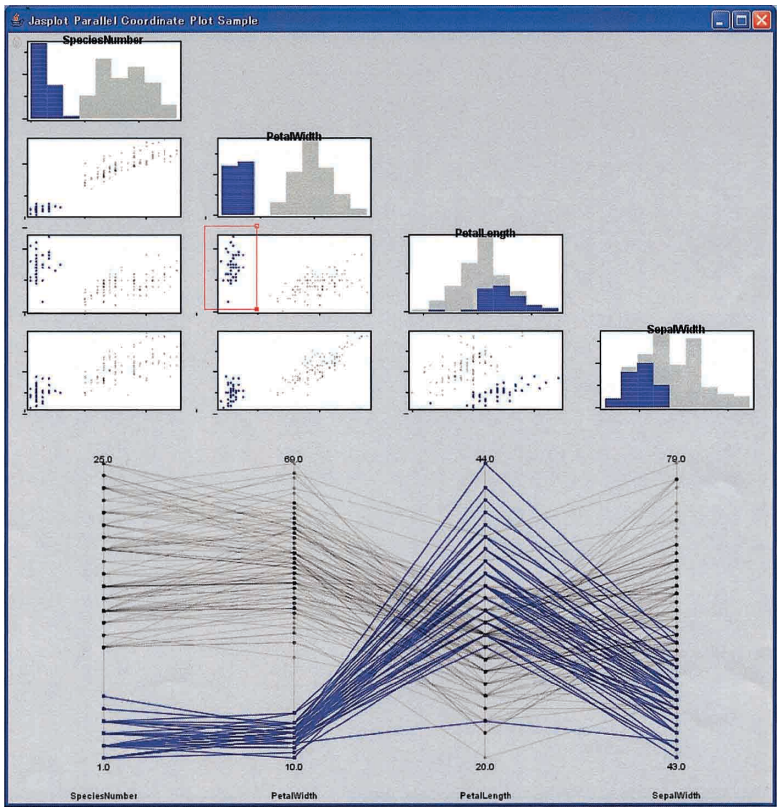

図 3. 散布図行列と平行座標プロット.

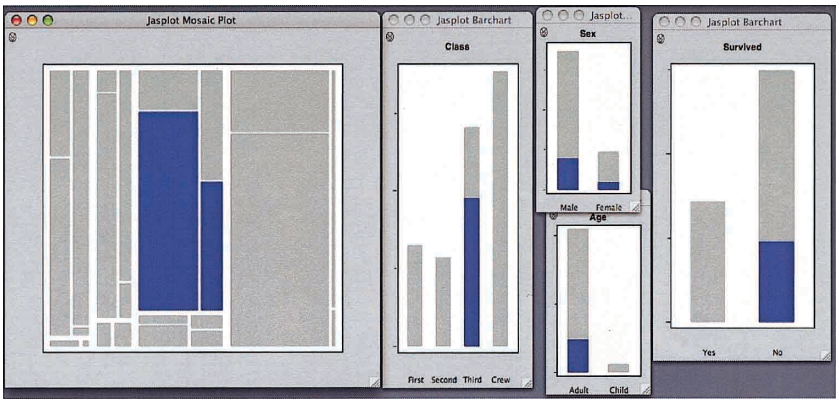

図 4. モザイクプロットと棒グラフ.

もできる．例えば，Jasplot の散布図行列や平行座標プロット(図 3)は，この機能を利用している．また，モザイクプロットや棒グラフ（図 4)も実装している．これらのグラフにおいて focusing，zooming，linked brushing などの対話的な操作が可能である．

3.1 デザインパターン

われわれは Jasplot の再利用性と拡張性を実現するために，プログラム設計において，GoF デザインパターンと MVC（Model-View-Controller）デザインパターンを利用している．

GoF デザインパターンとは，“Gang of Four（GoF）” と呼ばれる 4 人の著者による書籍(Gamma et al., 1995)でまとめられた良いオブジェクト指向プログラミングのためのガイドラインである．これらは 23 種類のパターンであり，生成に関するパターン，構造に関するパターン，振る舞いに関するパターン，という 3 種類のカテゴリに分類されている．これらの中で Jasplot の設計において特に有効であったのは，以下の 7 種類のパターンである．

Adapter パターンは，インタフェースをクライアントが期待する別のインタフェースに変換するパターンである．すなわち，互換性の無いインタフェースのために，そのままでは連携できないクラスを連携させることができる．Decorator パターンは，元のクラスに新しい機能を追加し，元のクラスの変更していないすべてのメソッドを新しいクラスに与えるものである．Factory method パターンでは，オブジェクトを生成するためのインタフェースを定義するが，どのクラスのインスタンスを生成するかはサブクラスが決定する仕組みを提供する．Mediator パターンは，複数のオブジェクトがどのように互いに影響しあうかをカプセル化するオブジェクトを定義するものである．つまり，Mediator パターンの利用によって，オブジェクトの結びつきを緩めて，相互作用を独立したものにすることができる．Observer パターンは，オブジェクト間の 1 対多の依存関係を定義し，1 つのオブジェクトの状態が変化した場合に，依存するすべてのオブジェクトに更新を連絡する仕組みである．これを実現するために，他のオブジェクト（“Subject”）からのイベントを受け取り，他の “Subject” に通知するオブジェクト（“Observer”）を作成する．State パターンは，状態を表す抽象クラスを定義し，そのクラスから派生したクラスとして異なる状態を表すものである．そして，適切な状態を表す派生クラスの中に振る舞いを定義する．そのために，状態の参照先の変更によって，その振る舞いを変更することができる．Strategy パターンは，アルゴリズムのグループを定義し，それぞれをカプセル化して，交換可能なものにするパターンである．これによって，クライアントからアルゴリズムを独立して交換できるようになる．

MVC（Model-View-Controller）デザインパターンは，GoF デザインパターンには含まれていない．しかし，グラフィックプログラミングのために非常に重要なデザインパターン，または，フレームワークである．これは，Samlltalk（Krasner and Pope, 1988）の GUI（Graphical User Interface）の開発のために提案されたものであり，多くのシステムで広く利用されている．MVC パターンでは，“Model”，“View”，“Controller” という 3 種類のオブジェクトのコミュニケーションによって GUI を構築する．

3.2 基本的なインタフェースとクラスの概要

Jasplot の設計では，2 章で述べた条件を実装するために，デザインパターンを考慮して，基本的なクラスを以下のように分類した．

- データのためのクラス
  `DataModel`，`RealDataModel`，`CSVDataModel`，`ExcelDataModel`，
  `DataTableDataModel`，`RDataModel`
- グラフの情報のためのクラス

PlotModel, BasicPlotModel, ScatterPlotModel, HistogramPlotModel, BoxPlotModel, MultiPlotModel, MosaicPlotModel, BarPlotModel

- グラフを描くためのクラス
  Plotter, BasicPlotter, ScatterPlotter, HistogramPlotter, BoxPlotter, MultiPlotter, MosaicPlotter, BarPlotter
- グラフを出力するためのクラス
  JasplotPanel, JasplotPanelPaletteLayer, JasplotPanelPaletteLayerUI
- 対話的な操作のためのクラス
  - マウスイベントを取得するためのクラス
    JasplotPanelDragLayer, JasplotPanelDragLayerUI, JasplotPanelPopupLayer, JasplotPanelPopupLayerUI
  - マウスイベントを解釈するためのクラス
    DragLayerState, BrushingState, DragGestureState, DraggingState, ManipulatingState, SelectingState, ToolTipState, ZoomingState
  - 観測値を選択するためのクラス
    Selector, RectangleSelector, PointSelector, LineSelector
  - linked views のためのクラス
    PlotModelEvent, PlotModelHandler, PlotModelListener, DataModelEvent, DataModelHandler, DataModelListener
- データを表形式で表示するためのクラス
  TableModel, TableDisplayer

さらに，デザインパターンにおいて推奨されているように，できるだけインタフェースを利用したプログラミングを行っている．具体的な機能は，それぞれのインタフェースを実装するクラスの中で記述されている．図 5 は，これらのクラスの関係を表した UML (Unified Modeling Language) のクラスダイヤグラム (Hunt, 2003) である．これ以外の機能が必要な場合には，これらのクラスの拡張やインタフェースの実装などによって機能の追加を容易にすることが考慮されている．

Jasplot の基本クラスである DataModel, PlotModel, Plotter, JasplotPanel には，MVC デザインパターンを適用している．DataModel インタフェースを実装したデータのクラスが “Model” の役割を果たす．それぞれのグラフの実体が “View” であり，2 種類のクラス (PlotModel インタフェースを実装したものと Plotter インタフェースを実装したもの) によって構成される．グラフを描いたり，マウスイベントを処理するための JasplotPanel クラスが “Controller” である．

3.3 データのためのクラス

Jasplot は，統計データを表現するために DataModel インタフェースによる汎用的なデータモデルを提供する．DataModel は単純な 2 次元の表であり，行と列がそれぞれ観測値と変数を表す．

DataModel には，観測値の値を実数値または文字列として取得したり観測値や変数の数を取得したりするメソッド，観測値の最小値，最大値，中央値の値を取得するメソッドなどが定義されている．

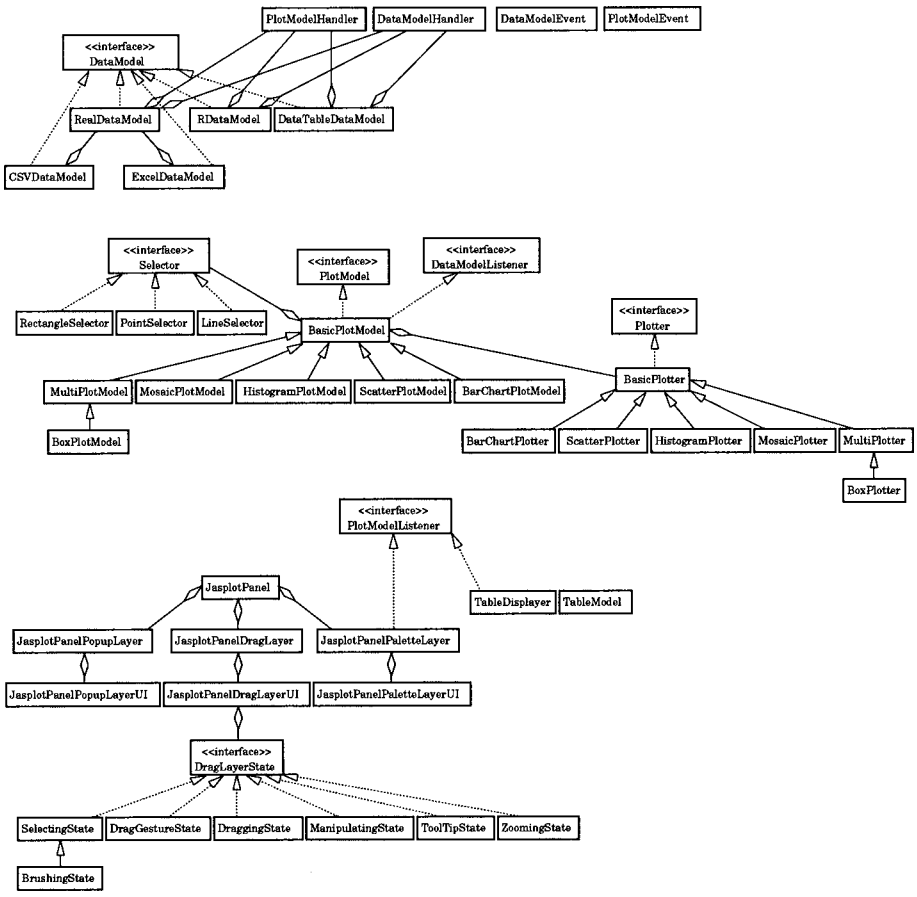

図 5. 基本的なクラスの関係.

Jasplot では，DataModel を実装するクラスを作ることによって様々なデータを扱うことができる．この例として，以下の 2 種類のクラスがある．

- Java プログラムでデータを読み込むためのクラス
  RealDataModel，CSVDataModel，ExcelDataModel
- 既存のシステムに組み込んでデータを読み込むためのクラス
  DataTableDataModel，RDataModel

RealDataModel は，実数配列に格納されたデータのためのクラスである．CSVDataModel は，CSV 形式でファイルに保存されているデータを読み込むためのクラスである．ExcelDataModel は，Microsoft Excel 形式のファイルを読むことができる．これらのクラスの実装のために Decorator パターンのコンポジション機能を使用している．例えば，CSVDataModel の setReal() は，指定された CSV ファイルからデータを読み込み，2 次元の実数配列にセットする．そして，RealDataModel の setReal() を用いて指定された実数データを保持する．これ以外の CSVDataModel に定義されているメソッドは，RealDataModel に委譲する．

DataTableDataModel は，Jasp に組み込んでデータのやり取りをするためのクラスであり，DataModel に定義されているメソッドと Jasp のデータ構造である DataTable のメソッドとの対応関係を定義している．RDataModel は，R のデータを Jasplot で利用するためのクラスである．これらのクラスの実装のために Adapter パターンを使用している．Jasplot では，DataModel が “Adapter” である．例えば，DataModel を実装する DataTableDataModel は，Jasp のデータ構造である DataTable を Jasplot から利用できるようにしている．例えば，観測値の値を実数として取得するメソッドが DataTableDataModel に送られた場合には，DataTable からその観測値のデータを取得して実数値として返す．

3.4 グラフの情報のためのクラス

PlotModel は，グラフの情報をセットしたり，取得したりするためのインタフェースである．そのために，データやグラフの情報を扱うためのメソッド，対話的な操作のためのメソッド，グラフに情報を追加するためのメソッドなどが定義されている．

データを扱うためのメソッドは，DataModel を実装するクラスのオブジェクトの指定とデータの取得を行う．グラフの情報を扱うためのメソッドは，色情報や軸のラベルなどの指定と取得を行う．対話的な操作のためのメソッドは，マウス操作に対する処理を定義しているオブジェクトやデータの選択状況を管理しているオブジェクトを扱う．グラフに情報を追加するためのメソッドは，グラフ上の指定した位置に文字などを描く．このとき，PlotModel は絶対的な位置情報を持たないため，相対的な位置を指定する．例えば，縦方向がグラフの下から 10%，横方向が左から 50% の位置，のように指定する．

PlotModel は，インタフェースなので，PlotModel を実装したクラスに実際のグラフのためのプログラムを書くことになる．しかし，すべてのグラフが PlotModel に定義されているすべてのメソッドを実装することは冗長なので，PlotModel を実装した抽象クラスである BasicPlotModel で基本的なメソッドの実装と Plotter や Selector やタイトルのような，重要なオブジェクトを保持するためのフィールドが定義されている．そのため，Jasplot に含まれている基本グラフのクラスは，BasicPlotModel を継承している．例えば，2 次元の散布図を描くためのクラスは，BasicPlotModel を継承した ScatterPlotModel である．

MultiPlotModel は，複数のグラフを並べたり，重ねたりして 1 つのグラフとして扱うためのクラスである．このクラスも BasicPlotModel を継承しているため，基本グラフと同じようにマウスによる対話的な操作を行うことができる．また，MultiPlotModel によって構成されるグラフをさらに MultiPlotModel によって配置することもできる (図 3)．これについての詳しい説明は，4 章で行う．

3.5 グラフを描くためのクラス

Plotter は，描画のためのインタフェースである．そのため，データを表す点や線，グラフの軸やラベルなどを描くためのメソッドが定義されている．これらのメソッドは，対応する PlotModel を実装するクラスからデータや色などの情報を取得して，Java のグラフィックスコンテキスト (java.awt.Graphics) に描く．

Plotter も PlotModel と同じようにインタフェースであるため，これを実装した抽象クラス BasicPlotter がある．例えば，散布図を描くためのクラスは，BasicPlotter を継承した ScatterPlotter である．これらのクラスの生成は，Factory method パターンによって実装した．Plotter を実装するクラスのオブジェクトは，JasplotPanel のオブジェクトが PlotModel を実装するクラスのオブジェクトに createPlotter メソッドを送ることによって生成される．createPlotter は，対応する Plotter を実装するクラスのオブジェクトが自動的に作られる．この createPlotter メソッドが “Factory method” である．

3.6 グラフを出力するためのクラス

JasplotPanel は，PlotModel と Plotter によって定義されたグラフを表示するためのクラスであり，表示したい PlotModel を実装するクラスのオブジェクトを指定するためのものである．

グラフに対するすべてのイベントは，JasplotPanel が処理する．例えば，ウインドウが表示されたり，大きさが変更された場合には，そのサイズを Plotter を実装するクラスのオブジェクトに送り，再描画する．また，マウス操作によるイベントは，指定されているオブジェクトに送られたり，ポップアップメニューの表示を開始したりする．

JasplotPanel は Swing の JLayeredPanel を継承している．そのため，最近のほとんどの対話型グラフィカルアプリケーションのように，複数のレイヤを利用できる．具体的には，3 つのレイヤを持っている．JasplotPanelPaletteLayer は，グラフを構成するための点や線や文字を描くものである．

3.7 対話的な操作のためのクラス

最近では，アプリケーションに対する操作を指定するには，マウスを利用することが多い．そのために，アプリケーションは，マウスボタンが押されたり離されたりする操作のイベントを検出しなければならない．そして，ユーザがマウス操作によって行いたい処理を解釈しなければならない．そこで，限られたマウス操作によってシステムに多くの命令を指示するために，1 つのマウス操作が状況に応じていくつかの意味を持つようになっている．例えば，グラフ上で四角形の領域を指定した場合，その領域に含まれる観測値を選択したい場合と，その領域を拡大したい場合とがある．また，異なるグラフ上では異なる選択方法も必要である．散布図での観測値選択では，四角形の領域による選択が自然であろう．すなわち，四角形の選択領域を描いて，その領域内の観測値を選択すればよい．そして，選択された観測値の変更のためにその領域を直接ドラッグできることが望ましい．ヒストグラムでの選択は，マウスボタンのクリックによって 1 つの階級を選択することが自然である．隣接する複数の階級の選択には，四角形の領域を指定することが適しているかもしれない．さらに，統計グラフのための対話的な操作では，linked views の機能は非常に重要である．

3.7.1 マウスイベントを取得するためのクラス

イベントには，キーやマウスボタンを押すような操作も含まれるので，対話的なマウス操作はイベント駆動型プログラミングによって効率的に実装できる．イベント駆動型プログラミングは，特定のイベントが発生した場合に指示された処理を行うために，常にイベント発生を監視する．

Jasplot では，対話的な操作のために JasplotPanelDragLayer のユーザインタフェースクラスである JasplotPanelDragLayerUI において Swing の MouseInputListener と DragGestureListener に定義されているメソッドである mousePressed, mouseMoved, mouseDragged, mouseReleased, dragGestureRecognized を実装することによってマウスイベントを処理する．ポップアップメニューの表示のためには，JasplotPanelPopupLayer のユーザインタフェースクラスである JasplotPanelPopupLayerUI クラスにおいて，mousePressed を実装している．

3.7.2 マウスイベントを解釈するためのクラス

状態に応じてマウスイベントの解釈を変更しなければならないが，これは，State パターンを適用することによって効果的に実装できる．Jasplot では，状態を表すクラスのインタフェースが DragLayerState である．このインタフェースは，振る舞いを決定するための 4 種類のメソッド(mousePressed, mouseMoved, mouseDragged, mouseReleased)を宣言している．具

体的な状態の実装は，BrushingState，DragGestureState，DragLayerState，DraggingState，ManipulatingState，SelectingState，ZoomingState などのクラスによって行われている．これらのうちの 1 つが状態として利用される．

これらの操作以外に，さらに特別な操作を実現したい場合には，JasplotPanelDragLayer のオブジェクトに addMouseMotionListener や addMouseListener メソッドを利用して，マウス操作のためのオブジェクトを追加することも可能である．

選択の操作は動的であるので，操作対象のグラフも動的に変化しなければならない．その際，再描画のタイミングについては注意が必要である．なぜなら，観測値の数が非常に多ければ，再描画にはかなりの時間が必要になるからである．Jasplot では，2 種類の再描画のタイミングを選択できる．1 つは，頻繁に再描画を行うことである．これは，BrushingState によって実装されている．観測値の選択状況が変わると直ちにグラフに反映されるので，直感的にわかりやすい．ただ，再描画には時間がかかることが多く，特に観測値数が多い場合には非常に遅くなる．もう 1 つの再描画のタイミングは，マウスボタンが離されたときである．これにより，再描画の回数を激減させることができる．これは，SelectingState によって実装されている．

DraggingState は，セレクタをドラッグすることによって，選択されたデータを移動する．この操作では，データの値が変化する．変化した値は，Reset メソッドを実行することによって移動したデータをもとの値に戻すことが可能である．

ManipulatingState は，グラフを操作するためのクラスである．このクラスは，グラフに枠を描き，その枠を操作することによって，x 軸，y 軸を反転させることができる．また，MultiPlotModel によって配置されたグラフに対しては，この枠のドラッグによってグラフを入れ替えることができる．平行座標プロットでは，この操作が基本操作の 1 つになっている．

ZoomingState は，mouseReleased によって指定された領域の拡大を行うためのクラスである．拡大したグラフは，Reset メソッドを実行することによってもとに戻すことができる．

Jasplot のグラフ上に描かれるすべての文字は，ツールチップの表示とドラッグを行える．長い変数名を描いた場合には，他の文字と重なって見にくいことがあるが，Plotter が描く文字の上にマウスカーソルを置くと，ツールチップが表示されるようになっている．また，グラフ上の文字列に対しては，DragGestureState によるドラッグ & ドロップができる．

3.7.3 観測値を選択するためのクラス

観測値を選択するために，グラフ上で観測値を表現するオブジェクトの選択が必要である．そのようなオブジェクトとしては，散布図における点，ヒストグラムにおける階級を示す四角形，平行座標プロットにおける折れ線などがある．

RectangleSelector は，オブジェクトを選択するためのクラスの 1 つであり，四角形の箱のような選択オブジェクトを指定し，選択オブジェクトの内側のオブジェクトを選択する．そして，マウスドラッグによるセレクタの移動と選択される観測値の動的な変更が可能である．

ヒストグラムの 1 つの階級は，マウスのクリックによって選択することが望まれる．これは，PointSelector によって実現されている．このクラスはまた，マウスボタンを押し，マウスカーソルをドラッグし，マウスボタンを離すという操作によって指定された四角形の領域内のオブジェクトを選択するためにも使用される．この四角形の領域は，マウスボタンを離した後，見えなくなる．

LineSelector は，グラフ上で線分を選択するための垂直の線分状のセレクタであり，それと交差する線分を選択できる．

これらのセレクタの交換は，Strategy パターンによって実装できる．Jasplot では，getRegion，mouseReleased などのすべてのセレクタのための共通のメソッドを定義するために Selector イン

タフェースを使用する．RectangleSelector, PointSelector, LineSelector は, Selector を実装している．例えば, HistogramPlotModel は, PointSelector を使用するが RectangleSelector を使用するように変更することも簡単である．これは，Strategy パターンの利点である．

3.7.4 linked views のためのクラス

これまでは，1 つの統計グラフ上での観測値の選択の仕組みを説明した．linked views の実現のためには，そのようなグラフが互いに連携しなければならない．そのためには，データの選択情報を共有しなければならない．さらに，選択されているデータが変更されたことを通知することも必要である．これは，Jasplot の PlotModel と PlotModelListener を実装した JasplotPanelPaletteLayer のオブジェクトに Observer パターンと Mediator パターンを適用することによって実現できる．PlotModel は Observer パターンの “Observer”, JasplotPanelPaletteLayer は “Subject” である．つまり，ある JasplotPanelPaletteLayer のオブジェクトの変化は，PlotModel のオブジェクトによって，他の JasplotPanelPaletteLayer のオブジェクトに配信される．PlotModel のオブジェクトは，Mediator パターンの “Mediator” であるので，それぞれ JasplotPanelPaletteLayer のオブジェクトは，配信先を知っている必要がない．

Jasplot では，PlotModelHandler のオブジェクトは，DataModel のオブジェクトと同時に生成される．そして，PlotModelListener インタフェースを実装した JasplotPanelPaletteLayer のオブジェクトは，PlotModelHandler の addPlotModelListener メソッドによって登録する．

マウス操作によって選択された観測値が変化した状況を考える．SelectorState オブジェクトは，各観測値が選択されているかどうかの情報を保持するための PlotModelEvent オブジェクトを生成する．PlotModelHandler の setSelectedObservation メソッドを利用して，観測値の選択状況を変更する．そして，登録された JasplotPanelPalletLayer のオブジェクトに PlotModelEvent を通知する．これによって，JasplotPanelPalletLayer の observationSelect ed (PlotModelEvent e) が実行される．このメソッドでは，repaint を実行しているので，再描画される．そのため，各 JasplotPanelPalletLayer オブジェクトは，新しい情報が表示されることになる．

4. Jasplot の利用

ここで，Jasplot を利用した簡単な例を示す．

Jasplot のソースコードとクラスファイルは，http://jasp.ism.ac.jp/Jasplot/ からダウンロードできる．ソースコードからコンパイルするためには，Apache Ant (http://ant.apache.org/) が必要である．ここで示す例を含むいくつかのデモは，ダウンロードした Jasplot に含まれている jasplot-demos.jar を実行することによって確認できる．

4.1 散布図の表示

Jasplot の最も簡単な例として，iris データの 2 つの変数の散布図を表示する以下のような Java プログラムを考える．

```
import javax.swing.JFrame;
import jp.jasp.jasplot.CSVDataModel;
import jp.jasp.jasplot.DataModel;
import jp.jasp.jasplot.JasplotPanel;
import jp.jasp.jasplot.ScatterPlotModel;
```

```
public class Sample {
    public Sample() {
        DataModel dataModel = new CSVDataModel("data/iris.csv");
        ScatterPlotModel model = new ScatterPlotModel(dataModel, 1, 2);
        JasplotPanel jasplot = new JasplotPanel(model);

        JFrame jFrame = new JFrame("Jasplot Scatter Plot Sample");
        jFrame.getContentPane().add(jasplot);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jFrame.setSize(500, 500);
        jFrame.setVisible(true);
    }

    public static void main(String[] args) {
        Sample sample = new Sample();
    }
}
```

一般的に，Jasplot を利用してグラフを描くためには，以下の手順が必要である．

(1) DataModel のオブジェクトを生成する．このプログラムでは，実際のオブジェクトは，DataModel を実装する CSVDataModel である．CSVDataModel のオブジェクトがファイル (data/iris.csv) からデータを読み込む．
(2) PlotModel のオブジェクトを生成する．ここでは，散布図を描くので，ScatterPlotModel のオブジェクトを生成している．
(3) JasplotPanel のオブジェクトを生成するために，表示する PlotModel のオブジェクトを指定する．このプログラムでは，JasplotPanel のオブジェクトを生成するときに ScatterPlotModel のオブジェクトを指定している．
(4) JasplotPanel を表示するための java.awt.Container のオブジェクトを生成する．ここでは，Container を継承する JFrame のオブジェクトを生成している．このオブジェクトの add メソッドによって JasplotPanel のオブジェクトを指定している．

このプログラムは，Jasplot の demo/ScatterPlot/Sample.java に含まれている．例えば，UNIX では，コンパイルと実行のために demo のディレクトリにおいて以下のコマンドを実行する．

```
javac -classpath ../dist/Jasplot.jar:. ScatterPlot/Sample.java
java -classpath ../dist/Jasplot.jar:. ScatterPlot.Sample
```

図 1 は，このプログラムの実行結果である．ここでは，四角形の領域でいくつかの観測値を選択できる．

4.2 平行座標プロットの実装

Jasplot では，MultiPlotModel を用いて ScatterPlotModel を配置して，それぞれの観測値を線で結ぶことによって平行座標プロットを実装している．以下は，これを実現するためのプログラムの一部である．このプログラムは，Jasplot の demo/ParallelCoordinatePlot/Sample.java にある．

```
MultiPlotModel multiPlotModel = new MultiPlotModel(dataModel);
multiPlotModel.setRowColumn(1, dataModel.getVariableNumber());
multiPlotModel.setConnect(true);
multiPlotModel.setSelector(new LineSelector());

for (int i = 0; i < dataModel.getVariableNumber(); i++) {
    ScatterPlotModel plotModel =
        new ScatterPlotModel(dataModel, PlotModel.NULL, i);
    plotModel.setDrawYTick(false);
    plotModel.setDrawXTick(false);

    Limits yLimits = plotModel.getYLimits();
    yLimits.setMin(dataModel.getMin(i));
    yLimits.setMax(dataModel.getMax(i));
    plotModel.setYLimits(yLimits);

    plotModel.drawString(String.valueOf(dataModel.getMin(i)),
                         0.5, 0.1, DrawString.CENTER, null);
    plotModel.drawString(String.valueOf(dataModel.getMax(i)),
                         0.5, 0.9, DrawString.CENTER, null);

    plotModel.setXAxisLabel(plotModel.getYAxisLabel());
    plotModel.setDrawYAxisLabel(false);
    plotModel.setDrawFrame(false);
    plotModel.setDrawXZeroLine(true);
    multiPlotModel.setPlotModel(plotModel);
}
```

このプログラムでは，setRowColumn メソッドによって，グラフの横方向を変数の数と同じ列数だけの区画に分割する．次に，setConnect(true) メソッドによって，配置するグラフの同じ観測値のデータを線で結ぶように指定する．そして，setSelector(new LineSelector()) によって，線分によるセレクタに変更する．この後の for 文による繰り返しによって散布図を配置している．散布図のオブジェクトは，new ScatterPlotModel(dataModel, PlotModel.NULL, i) によって生成している．これによって，x 軸に対応するデータは無く，y 軸に対応するデータは dataModel の i 列目のデータを表示する散布図となる．このオブジェクトに対して，setDrawYTick(false) は y 軸を描かないことを，setDrawXTick(false) は x 軸を描かないことを，setXAxisLabel(plotModel.getYAxisLabel()) は x 軸の変数を描く場所に y 軸で指定した変数名を描くことを，setDrawYAxisLabel(false) は y 軸の変数名を描かないことを，setDrawFrame(false) は散布図の枠を描かないことを，setDrawXZeroLine(true) は x 軸の 0 の位置に線を描くことを指定する．setPlotModel(plotModel) によって，引数で指定されたオブジェクトが左から順番に配置される．変数の数が多い場合には，グラフが横方向に長くなるので JasplotPanel オブジェクトを Swing の JScrollPanel 上に描くようにするとスクロールバーを利用することができる．

上記のプログラムを MultiPlotModel を継承するクラスにすると，そのクラスをさらに

MultiPlotModel によって配置することができる．例えば，図 3 は，MultiPlotModel によって作られている散布図行列と平行座標プロットを MultiPlotModel によって 1 つのグラフとして表示している．

4.3 Jasp からの利用

図 6 は，Jasp から Jasplot を利用している例であり，散布図行列と平行座標プロットと箱ひげ図を 1 つのグラフとして表示した．この表示を行うための Jasp プログラムは，以下のものである．このプログラムは，Jasp の examples/jasplot.jasp に含まれている．

```
iris = read("iris.jdt")
data = DataTableDataModel(iris)
mpm = MultiPlotModel()
mpm.setRowColumn(3, 1)
model = ScatterPlotMatrixModel(data)
mpm.setPlotModel(model)
model = ParallelCoordinatePlotModel(data)
mpm.setPlotModel(model);
model = BoxPlotModel()
model.setChangeAxis(true)
model.setDataModel(data)
mpm.setPlotModel(model)
jasplot = JasplotPanel(mpm)
jasplot
```

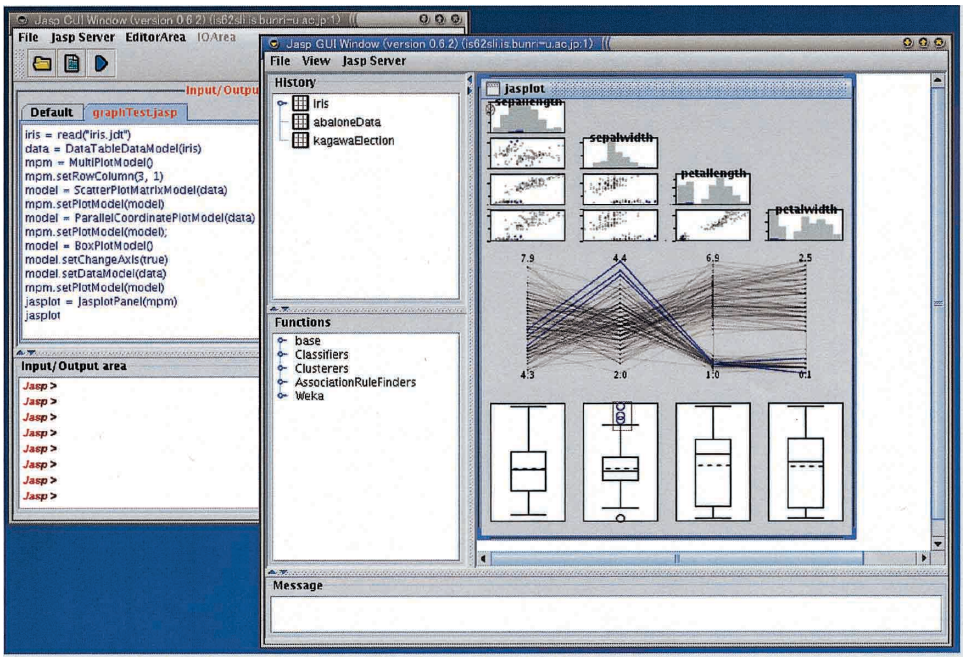

図 6. Jasp からの利用.

まず，iris = read("iris.jdt") によって iris.jdt を読み込んでいる．次に DataTableDataModel のオブジェクトを生成して，変数 data に代入する．この data を指定してグラフを描く．ここでは，3 種類のグラフを MultiPlotModel によって 1 つのグラフとして表示する．そのために生成された MultiPlotModel のオブジェクトに対して，setRowColumn(3, 1) というメソッドを送ることによって 3 行 1 列の区画に分割している．setPlotModel メソッドによって，それぞれの区画に ScatterPlotMatrixModel，ParallelCoordinatePlotModel，BoxPlotModel，のオブジェクトを順番に配置している．

このように，Jasp 言語では Java のクラスを直接利用することができる．もちろん，Jasplot の基本グラフについては，Jasp 言語の関数が準備されているので，Java を知らなくても簡単に利用できる．

4.4 R からの利用

図 7 は，R から Jasplot を利用している例である．このグラフは，以下のような R のコマンドを実行することによって描かれる．

```
library(rJava)
source("jasplot.R")
dataModel <- jasplotDataModel(iris)
jasplotScatterPlotMatrix(dataModel)
jasplotParallelCoordinatePlot(dataModel)
```

Jasplot を R から利用するためには rJava パッケージが必要である．まず，library(rJava) によって rJava を利用可能にしなければならない．次に，jasplot.R を R に読み込む．jasplot.R は，Jasplot の demo/R にあるファイルである．jasplot.R には，Jasplot を rJava から利用できるようにするための指定と Jasplot のクラスを R から利用するための関数が定義されている．

rJava から Jasplot を利用するためには，rJava の関数である .jinit の引数に Jasplot.jar を指定する．Jasplot.jar は，Jasplot の dist にあるファイルである．

この例では，Jasplot のクラスを利用するための関数として，jasplotDataModel，jasplotScatterPlotMatrix, jasplotParallelCoordinatePlot を使用している．jasplotDataModel は，Jasplot の RDataModel のオブジェクトを生成する関数である．この jasplotDataModel によって生成されたオブジェクトを引数として jasplotScatterPlotMatrixModel や jasplotParallelCoordinatePlot を実行することによって Jasplot によって散布図行列や平行座標プロットが描かれる．

R から描かれたグラフに対しても対話的な操作が可能である．さらに，Jasplot のグラフ上で選択された観測値を R に知らせるための関数(getSelectedObservation) も準備されている．グラフを描くための関数に指定した変数をこの関数に与えることによって選択されている観測値は true，選択されていない観測値は false という値のベクトルが返される．

5. おわりに

本論文では Java による統計グラフライブラリ Jasplot を説明した．Jasplot は，グラフ上でのマウスによる対話的な操作を行うことや複数のグラフが連携することなどを前提として設計されている．そのため，マウスによるデータの選択やそれを他のグラフに反映するようなことを比較的容易に実現できる．これらの機能は，Jasp や R のような既存のアプリケーションに組み込んだ場合や Java プログラムで利用した場合に全く同じように操作できる．Jasplot の設計においては，デザインパターンを考慮した．それにより，実装がある程度簡単になり，拡張

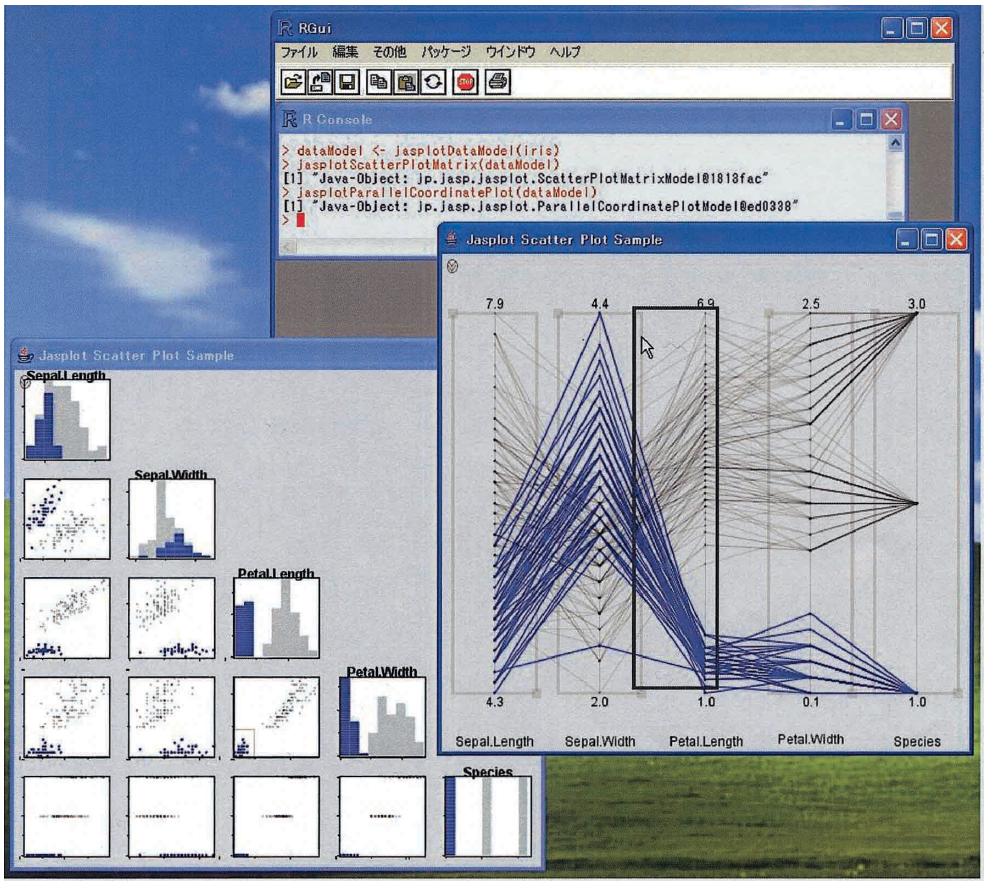

図 7. R からの利用.

性を持つこともできた．特に，デザインパターンの基本原則の一つであるインタフェースによるプログラミングは，既存のアプリケーションに組み込み易くすることや統一的な対話的操作を実現することに有効であった．

ただ，現在の Jasplot には，データ視覚化のためのソフトウエアとして重要な機能の中で欠けているものも多い．例えば，色の指定は不十分である．また，3 次元グラフの機能は実現されていない．これらは，今後改善していく予定である．

参 考 文 献

Gamma, E., Helm, R., Johonson, R. and Vlissides, J. (1995). *Design Patterns*: *Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading. (本位田真一・吉田和樹 監訳 (1999). 『オブジェクト指向における再利用のためのデザインパターン 改訂版』, ソフトバンクパブリッシング, 東京)

Hunt, J. (2003). *Guide to the Unified Process Featuring UML, Java and Design PatternsR Graphics*,

Springer, London, Berlin, Heidelberg.

Krasner, G. and Pope, S. (1988). A description of the Model-View-Controller user interface paradigm in the Smalltalk-80 system, *Journal of Object Oriented Programming*, **1**, 26–49.

Martin, R. (2003). *Agile Software Development*: *Principles, Patterns, and Practices*, Prentice Hall PTR, Upper Saddle River, New Jersey. (瀬谷啓介 訳 (2004).『アジャイルソフトウエア開発の奥義』, ソフトバンククリエイティブ, 東京)

Murrell, P. (2005). *R Graphics*, Chapman & Hall/CRC, Boca Raton.

Nakano, J., Fujiwara, T., Yamamoto, Y. and Kobayashi, I. (2000). A statistical package based on Pnuts, *COMPSTAT2000 Proceedings in Computational Statistics* (eds. J. G. Bethlehem and P. G. M. van der Heijden), 361–366, Physica, Heidelberg.

Nakano, J., Huh, M. Y., Yamamoto, Y., Fujiwara, T. and Kobayashi, I. (2004). Adding visualization functions of DAVIS to Jasp: Mixing two Java-based statistical systems, *Computational Statistics*, **19**, 137–146.

Symanzik, J. (2004). Interactive and dynamic graphics, *Handbook of Computational Statistics* (eds. J. E. Genyle, W. Härdle and Y. Mori), 293–336, Springer, Berlin, Heidelberg.

Urbanek, S. and Theus, M. (2003). iPlots — high interaction graphics for R, *International Workshop on Distributed Statistical Computing* (*DSC 2003*) (eds. K. Hornik, F. Leisch and A. Zeileis) http://www.ci.tuwien.ac.at/Conferences/DSC-2003.

Proceedings of the Institute of Statistical Mathematics Vol. 55, No. 1, 27–45 (2007) 45

A Java Library for Statistical Graphs Using Design Patterns

Yoshikazu Yamamoto[1], Junji Nakano[2,3] and Keisuke Honda[3]

[1]Faculty of Engineering, Tokushima Bunri University
[2]The Institute of Statistical Mathematics
[3]School of Multidisciplinary Sciences, The Graduate University for Advanced Studies

This paper describes the design and the implementation of a Java library for statistical graphs. First, we consider the required functions for a modern interactive statistical graph library. Then we explain our statistical graph library, called Jasplot (**Ja**va **s**tatistical **plot**), as an example. Jasplot has functions for interactive operations on graphs, such as brushing and linked views. We can build new statistical graphs by combining several basic graphs and/or components such as an axis and data input functions. We can easily use Jasplot from other statistical analysis software products such as Jasp and R. We adopt "design pattern" software technology to realize such functions effectively.

Key words: Java, linked views, interactive operation, statistical graph library.