統計数理 (2001)
第 49 巻 第 2 号 261–275

統計グラフライブラリの開発とWebへの応用

佐藤 健一[1] ・大瀧 慈[1]

（受付 2001 年 2 月 21 日；改訂 2001 年 4 月 26 日）

要　　旨

Windows OS の普及と共にその開発環境も整備されてきた．これにより，従来のコマンドライン形式のアプリケーションだけでなく，グラフィカルなユーザーインターフェイスを持つものも容易に作成できるようになった．本稿では，こうした状況において汎用的に利用され得る統計グラフライブラリについて議論し，実装されたライブラリの仕様について説明する．また，その利用例として，Web 上のグラフ描画システムを取り上げる．

キーワード：Windows OS, DELPHI, データベース，グラフライブラリ．

1. はじめに

近年，Windows OS 上のソフトウェアの開発環境はめざましい発展を遂げてきた．様々な種類のアプリケーションの開発を容易にすると同時に，新しい高度な技術の利用をも可能にした．本稿では，開発環境 DELPHI（ボーランド社）において統計グラフライブラリを開発し，これを用いた Web 上の描画システムへの応用について議論する．

DELPHI は ALGOL から派生した教育用言語 PASCAL を用いた統合開発環境である．他の言語と比較するとマイクロソフト社の VISUAL BASIC と同様に優れたユーザーインターフェイスおよびビジュアルなコンポーネントを提供し，C++と同等の高速な実行ファイルを生成する．さらに，JAVA と同様にオブジェクト指向言語であり，カプセル化，継承，多様性，およびインターフェイス（多重継承，多様性）などの概念を持つ．また，厳格な文法によって高速なコンパイルを実現するなど，アプリケーションの作成において高い生産性を誇る．現在，DELPHI は Windows OS だけでなく，Linux OS において KyLix（カイリックス）として移植され，Mac OS (ver 10) での利用も可能となりつつある．これにより，JAVA と同様に OS に依存しない開発環境として利用されることが期待される．

我々の提供する統計グラフライブラリをアプリケーションから利用する手順は図 1 のようにまとめることができる．この流れに従って, 第 2 章ではグラフのもとになるデータファイルについて説明し, 第 3 章ではグラフを定義するグラフファイルについてその具体的な記述も含めて説明する．第 4 章では, グラフライブラリを構成する関数を紹介し, 第 5 章においてライブラリを利用するシステムの実例としてユーザーと対話的な Web 上のアプリケーションを取り上げる．尚, 本稿で紹介するライブラリ等はホームページ, http://apollo.rbm.hiroshima-u.ac.jp/satoh/ において配布されている．

[1]広島大学 原爆放射能医学研究所：〒734–8553 広島市南区霞 1–2–3

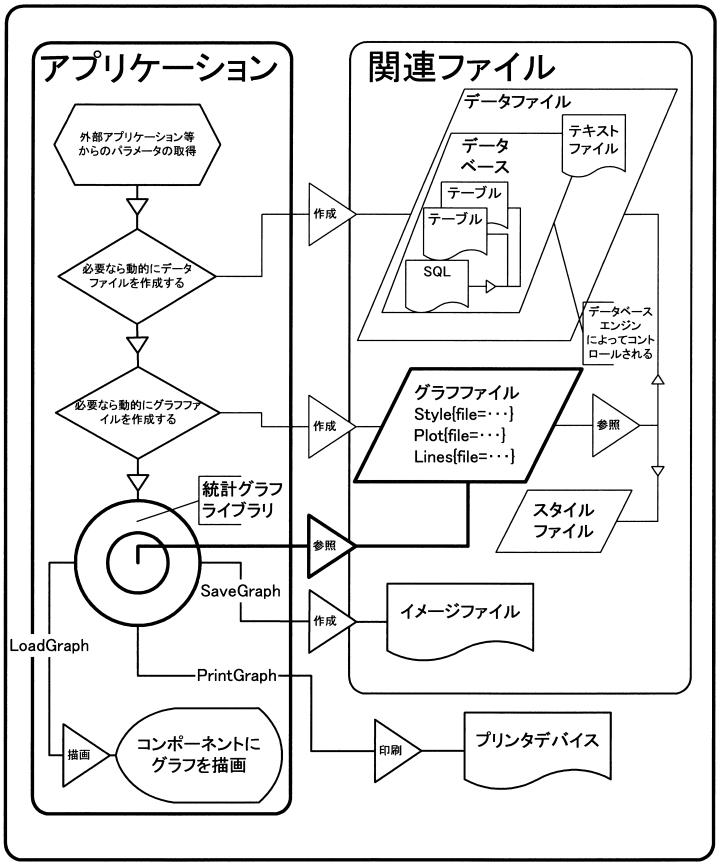

図 1. 統計グラフライブラリのアプリケーションでの利用とその関連ファイル. 一つのアプリケーションの中で閉じた利用ができ，他のアプリケーションで利用できるイメージファイルの出力も可能である.

2. データファイル

一般的に，グラフライブラリの開発にあたっては，その元となるデータの扱いが重要な要素となる．本ライブラリでは『データはファイルとして与えられている』ことを前提とした．グラフソフトの中には数式を与えることでグラフを描画するものがあるが，内部的には数値データが生成されていることに注意して頂きたい．一方で，データを配列としてサブルーチンに渡す場合も考えられる．このようなデータの利用は，アプリケーションの内部で生成されるデータを扱う場合には高速かつ効率的であるが，アプリケーションの外部から提供されるデータに対しては扱いが難しい．

データが記述されるファイルとして最も単純な構造を持つものは表 1 (坂元 他 (1983)) で与えられるようなテキストファイルであろう．通常, テキストファイルの 1 行目には各データ列に対する列名が記述され，2 行目以降に実際のデータが記述される．各データ列はカンマ，タブおよび空白で区切られ，対応する拡張子はそれぞれ，csv, txt および prn とされる．例外的に，空白やカンマを含む文字列を扱う場合にはダブルクオーテーションマーク『”』で囲むこと

表 1. 理科年表から編集された日本の 20 都市における 1 月の日最低気温の月平均値 (1941 年–1970 年) のデータ（坂元 他（1983), p. 138 より引用）データファイル名は “Temperature.txt” とした.

都市名	気温	緯度	経度	標高
稚内	-8.0	45.42	141.68	2.8
旭川	-13.6	43.77	142.37	111.9
札幌	-9.5	43.05	141.33	17.2
青森	-5.4	40.82	140.78	3.0
盛岡	-6.7	39.70	141.17	155.2
仙台	-3.2	38.27	140.90	38.9
金沢	-0.1	36.55	136.65	26.1
長野	-5.5	36.67	138.20	418.2
高山	-7.6	36.15	137.25	560.2
軽井沢	-10.0	36.33	138.55	999.1
名古屋	-0.9	35.17	136.97	51.1
飯田	-4.7	35.52	137.83	481.8
東京	-0.4	35.68	139.77	5.3
鳥取	0.5	35.48	134.23	7.1
京都	-0.6	35.02	135.73	41.4
広島	0.2	34.37	132.43	29.3
福岡	1.5	33.58	130.38	2.5
鹿児島	2.0	31.57	130.55	4.3
高知	0.1	33.55	133.53	1.9
那覇	13.5	26.23	127.68	34.9

がある．このようなテキストファイルはほとんどの表計算ソフトウェアおよび統計ソフトウェアにおいて利用することができ汎用性が高い．

しかしながら，テキストファイルの場合，レコード数が多くなると IO（ハードディスクへの読み書き）に時間がかかるという欠点がある．また，グラフを描く場合，しばしばデータの一部のレコードだけを抜き出すことがあり，テキストファイルを用いると，その都度，表計算ソフトなどに読み込ませて編集をする作業が必要となる．これらの問題に対する一つの答えとして，データベースの利用が考えられる．データベースはいくつかのテーブルファイルとそれをまとめて操作する SQL ファイルから成る．テーブルファイルは基本的にテキストファイルと等価なものである．付加的な情報としては，各列のデータ型，データの一意性および NULL の許可等が挙げられる．また，IO の高速化および検索の高速化のための仕様を持つ．SQL ファイルは複数のテーブルファイルを統括的に扱うことができる Structured Query Language を記述したスクリプトファイルである．例えば，共通のデータ列を持つテーブルファイルがあれば，その列をキーとしてデータを結合させ，仮想的に 1 つのテーブルを抽出することができる．実際，表 2 に示すように，その抽出結果である仮想的なテーブルはデータとして利用できる．そこで，本ライブラリでは SQL ファイルを 1 つのデータファイルとして扱うものとした．SQL の実行などデータベースに関する処理は，通常データベースエンジンと呼ばれる外部モジュールによってコントロールされる．ここでは，DELPHI Enterprise 版に付属する無償配布可能な BDE (Boland Database Engine) を採用した．

表 2. SQL ファイル “Temperature.sql” によって抽出されたレコードのデータ．SQL ファイルには “Select * from Temperature.db where 標高 < 10” と記述し，その意味は，“Temperature.db” から標高が 10 未満のレコードを全てのフィールドについて抜き出しなさい，である．ただし，“Temperature.db” は表 1 で示された “Temperature.txt” をテーブルファイルに変換したものである．

都市名	気温	緯度	経度	標高
稚内	-8.0	45.42	141.68	2.8
青森	-5.4	40.82	140.78	3.0
東京	-0.4	35.68	139.77	5.3
鳥取	0.5	35.48	134.23	7.1
福岡	1.5	33.58	130.38	2.5
鹿児島	2.0	31.57	130.55	4.3
高知	0.1	33.55	133.53	1.9

3. グラフファイル

グラフの定義ファイル（以下，グラフファイルと呼ぶ）は編集が容易なテキストファイルとし，グラフの記述形式は，統計解析ソフトウェア *S-PLUS* (Becker et al. (1991)) を参考にした．その特徴は，端的に言えば，必要最小限のスクリプトでもっともらしいグラフを描くことにある．その意味でデフォルトの設定は重要である．本ライブラリではグラフとして散布図，箱ひげ図，ヒストグラム，散布図行列および簡単な表組みを扱うことができるが，ここでは，散布図について説明する．はじめに，表 1 のデータを用いて各都市の位置を散布図で表す．このとき，グラフファイル “Temperature.log” は以下のようになる．

スクリプト 1. **Plot**{ **file**=“Temperature.txt”; **xcol**=“経度”; **ycol**=“緯度” }

このグラフファイルに対応するグラフイメージは図 2 である．ここで，各都市を表すマークは “+” に設定されているが, このようなデフォルトの設定やグラフの論理構造以外の視覚的な設定はグラフファイルとは別にスタイルファイルによって記述できるようにした．これはホームページを記述する HTML とその視覚的な要素を指定するカスケードスタイルシートの関係を参考にしたものである．

次に，本ライブラリが持つ強力な機能について紹介する．図 2 を見ても分かるように散布図は 2 次元の情報を表現するに過ぎない. しかしながら，マークのプロパティを利用することによって 3 次元以上の情報を視覚化することができる．確かに，このような工夫をしなくても，多くのソフトで直接高次元のデータを立体的なグラフとして表現することができるかも知れない．しかしながら，統計で扱うような一般的なデータは大域的な変動を持つ関数曲線と異なり，局所的な変動が大きいために，立体的なグラフを描いても得られる情報が多いとは言い難い．むしろ，2 次元の散布図で表現する方が『一目でその傾向を理解する』のに適していると思われる．マークの大きさで 3 次元目の情報を表現する散布図はバブルチャートと呼ばれる．本ライブラリではこのようなグラフを簡単な記述によって実現できる．まず，はじめに，3 次元目の情報としてデータ列の値を直接表示することを考える．例えば，図 2 のグラフにおいてマークの代わりにその “都市名” を表示させるには以下のように記述する（参照：図 3).

スクリプト 2. **Plot**{ **file**=“Temperature.txt”; **xcol**=“経度”; **ycol**=“緯度”;
txt=**ref**(“都市名”) }

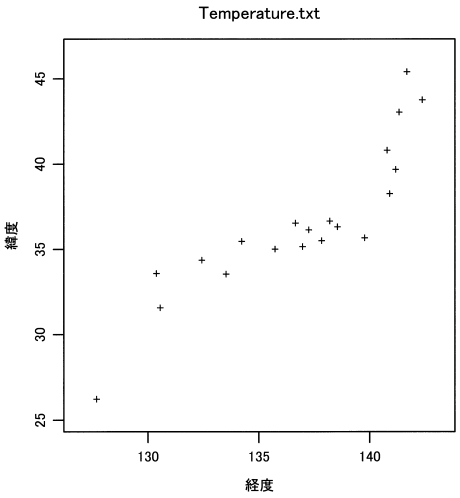

図 2. グラフファイル “Temperature.log” によって描かれた各都市の位置を示す散布図．デフォルトの設定では，グラフ上部のタイトルとしてデータファイル名，X 軸のラベルとして X 軸に割り当てたデータ列名，Y 軸のラベルとして Y 軸に割り当てたデータ列名が表示される．

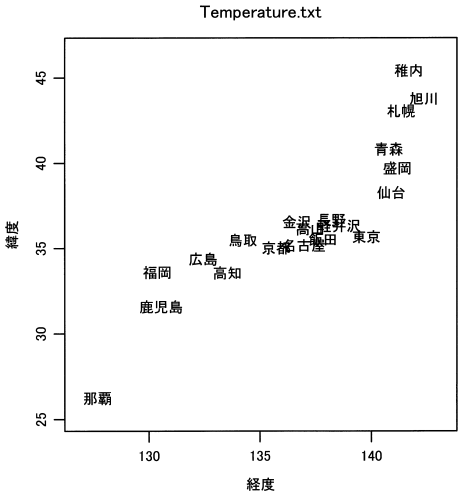

図 3. マークとしてデータファイル “Temperature.txt” のデータ列 “都市名” のデータを文字列として参照した散布図．

データファイルの列として 3 次元目の情報になるデータ列が存在すればこの方法は有効である．しかし，データファイルが含む情報の大部分は数値であり，これを直接表示させても有効でない場合が多い．そこで，数値のデータを一度，離散化し，これにテキスト情報を割り当てる機能を実装した．ここでは，“気温” 列のデータをテキスト化することを考えよう．例えば，その手順は，(1) “気温”列のデータの分布をもとに -5 と 0 に仕切りを入れる，(2) 2 つの仕切りによりデータは 3 つの区間，$[-\infty, -5)$, $[-5, 0)$ および $[0, +\infty)$ に分割される，(3) 3 つの分割それぞれに対して “低”, “中” および “高” のテキストを割り当てる，となる．グラフファイル内の記述としては “txt” プロパティを次のように変更するだけである（参照：図 4).

スクリプト 3. **Plot**{ **file**=“Temperature.txt”; **xcol**=“経度”; **ycol**=“緯度”;
txt=**ref**(“気温”, “−5 0”, “低 中 高”)}

本ライブラリではマークとしてテキスト文字を利用しているが，一般的な言語環境においては視覚化の手段として適切でない場合も考えられる．そのような場合には，グラフも画像の一種であることから，そのマークとして画像を利用することが有効であると思われる．実際，本ライブラリでは Bitmap ファイル形式の画像ファイルをテキストの代わりに用いることが可能である．例えば，(1)txt=“+” の代わりに txt=“A.bmp”，(2) スクリプト 2 で利用した “都市名” のデータ列において都市名の代わりに Bitmap ファイル名，(3) スクリプト 3 におけるテキスト “低”, “中” および “高” の代わりに “A.bmp”, “B.bmp” および “C.bmp”, のようにして利用できる．この他にも, 基本的な図形：○，△，□等についてはテキスト番号（“# 番号”）によって描画することができる．また，詳しくは第 5 章にて解説するが，他のデータ列を参照した表現はテキストだけでなく，マークの色および大きさについても同様の処理が適用できる．

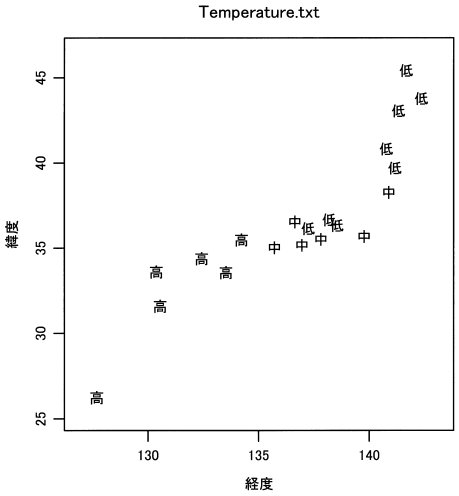

図 4. マークとしてデータファイル “Temperature.txt” のデータ列 “気温” を参照し，データを 3 区間に分割して，それぞれに文字列 “低”，“中” および “高” を割り当てた散布図.

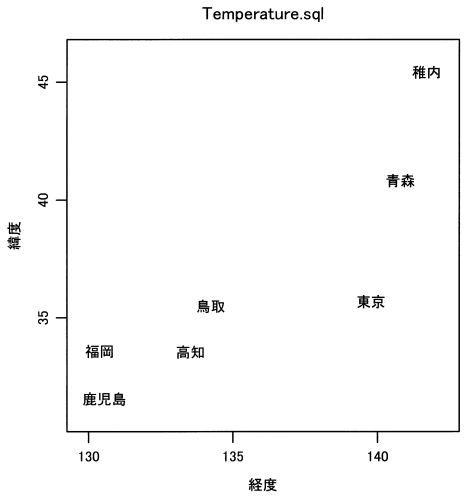

図 5. 表 2 の抽出結果を持つ SQL ファイル “Temperature.sql”をデータファイルとして利用した散布図.

最後に，データとして SQL ファイルを用いることについて説明する．記述方法としてはスクリプト 1 での “file” プロパティの値をテキストファイル “Temperature.txt” から SQL ファイル “Temperature.sql” に変更するだけでよい．内部的にデータベースエンジンが SQL を実行し，該当するレコードからなるデータをデータファイルとして参照する（参照：図 5）．また，SQL の性質から，表 2 での “Temperature.sql” において where 文を取り除くと，抽出結果はテーブルファイル “Temperature.txt” と等価となり，その結果，図 5 は図 2 と同じグラフになる．その意味では，全てのテーブルファイルは SQL ファイルを通して参照できる．しかし，参照する際に，データを抽出するというプロセスが必要となるので多用は避けたい．

4. 統計グラフライブラリ

データファイルとグラフファイルの導入により我々の開発した統計グラフライブラリ *StatGraph.pas* は単純化されている．ライブラリを構成するサブルーチン（DELPHI ではプロシージャという）は，*LoadGraph*, *SaveGraph* および *PrintGraph* の 3 つのみである．LoadGraph は抽象的な描画スペース *TCanvas* に対してそのメソッド，プロパティを利用してグラフを描画する関数である．その描画スペースとしてイメージファイルを対象としたものが SaveGraph, プリンタデバイスを対象としたものが PrintGraph である．まず，各関数の引数の説明と使用例を与え，最後にライブラリを利用した簡単なアプリケーションについて述べる．はじめに，LoadGraph について説明する．

プロシージャ 1. **LoadGraph**(

const	**FileName**:	String;
const	**Canvas**:	TCanvas;
const	**CanvasRect**:	TRect

)

ただし，引数 FileName はグラフファイル名，Canvas はグラフイメージを描画する TCanvas 型のインスタンス，CanvasRect は Canvas 上の描画領域をあらわす．各引数の前に記述された const はその引数で渡された変数が関数内で変更されないことを保証する．引数 CanvasRect を見ても分かるように，グラフファイル自体にはグラフの大きさの情報はなく，描画する際に直接指定するようになっている．このことにより，異なる描画領域を持つ Canvas（イメージファイル，プリンタ等）に対して同一のグラフファイルの使用が可能となっている．また，関数内での描画処理は Windows OS が提供する API ライブラリを直接使用することを極力押さえ，DELPHI の提供するインターフェイス，すなわち，Canvas プロパティの持つプロパティおよびメソッドを通して行っている．このため, 将来的には少ない修正で異なる OS 上でも利用できる可能性を持っている．以下にアプリケーション内での使用例について述べる．Form1 が持つ Canvas に，グラフファイル “Temperature.log” に記述されたグラフを描画するのであれば，

使用例 1. StatGraph.LoadGraph(‘Temperature.log’, Form1.Canvas, Form1.ClientRect)

とコーディングすれば良い．次に，グラフイメージをイメージファイルとして出力したい場合に有効な関数 SaveGraph を説明する．

プロシージャ 2. **SaveGraph**(

const	**FileName**:	String;
const	**ImageFileName**:	String;
const	**Width, Height**:	Integer;
const	**Progressive**:	Boolean

)

ここで，FileName は LoadGraph と同様にグラフファイル名，ImageFileName はグラフを保存するイメージファイル名，Width, Height はイメージファイルの幅と高さを表す．イメージファイルの種類は Bitmap と Jpeg をサポートしており，ImageFileName の拡張子で決定する．また，Jpeg ファイルの場合には Progressive を用いて段階的に表示させるかどうかを指定できる．例えば，“Temperature.log” で記述されるグラフイメージを 400×400 ピクセルの Bitmap ファイル “Temperature.bmp” として保存したければ以下のようにコーディングすれば良い．

使用例 2. StatGraph.SaveGraph (‘Temperature.log’, ‘Temperature.bmp’, 400, 400, false)

前述したように，同様の作業は直接 LoadGraph を用いても可能である．

使用例 3.

```
var
  Bitmap:TBitmap;
begin
  Bitmap:=TBitmap.Create;
  try
    with Bitmap do begin
      Width:=400; Height:=400;
```

```
      StatGraph.LoadGraph('Temperature.log', Canvas,
        Rect(0, 0, Width, Height));
      SaveToFile('Temperature.bmp');
    end;
  finally
    Bitmap.Free;
  end;
end;
```

次に，グラフイメージをプリンタデバイスに出力するための PrintGraph について説明する．

プロシージャ 3. **PrintGraph**(

```
const   FileName:          String;
const   Width, Height:     Integer;
    )
```

ここで，FileName は印刷したいグラフファイル名，Width, Height は直感的には印刷時のイメージの幅と高さを表す．実際，グラフイメージは印刷時の印刷方向にしたがって，その比を保ったままプリンタの持つ描画領域に応じた大きさで印刷される．プリンタデバイスの持つ Canvas の大きさは 4000×3000 ピクセル程度あり，通常のディスプレイよりも 10 倍程度広い．したがって，グラフを高品位に印刷するには画面のハードコピーでなく，直接描画することが不可欠である．さらに，ユーザーの利用するプリンタデバイスは様々であるから，その環境において最良の状態で印刷されるように，このような工夫が必要となる．しかし，本ライブラリによる印刷は非常に簡素化されており，たとえば，“Temperature.log” で記述されるグラフイメージを正方形で印刷したければ，

使用例 4. StatGraph.PrintGraph ('Temperature.log', 400, 400)

とするだけでよい．また，印刷を実行するプリンタデバイスの細かい設定（印刷方向，部数など）が必要な場合は，この関数を呼び出す前にプリンタダイアログなどを表示させて設定すればよい．

ライブラリを構築することの最大の利点は，『生産性の向上』にある．我々の開発した統計グラフライブラリはビジュアルなコンポーネントを使用していないため，多くの種類のアプリケーション（コンソールアプリケーション，通常のフォームアプリケーション，Active X フォームアプリケーション，CGI アプリケーション等）に組み込むことが可能である．ここではその簡単な例としてコンソールアプリケーションを取り上げる．プログラム 1 にその全ソースコードを示す．図 1 にしたがってその機能を説明すると，(1) グラフファイル名をコマンドラインで受け取る，(2) 統計グラフライブラリを用いて，そのイメージファイルを作成する，となる．尚，このアプリケーションは DELPHI 以外で作成されたアプリケーションから呼び出すことも可能である．

プログラム 1. コマンドラインからグラフファイル名を取得し，グラフをイメージファイルとして保存するプログラム．

```
program Project1;
{$ APPTYPE CONSOLE }
// StatGraph をライブラリとして使用する
uses
  SysUtils, StatGraph;
var
  FileName: String; // グラフファイル名
  i: Integer;
begin
  // グラフファイル名の取得
  FileName:=‘ ’;
  for i:=1 to ParamCount do begin
    FileName:=FileName+ParamStr(i);
  end;
  // グラフファイルが存在しなければ終了
  if not FileExists(FileName) then exit;
  // グラフをイメージファイルとして保存
  StatGraph.SaveGraph(FileName,
    ChangeFileExt(FileName, ‘.jpg’), 400, 400, false);
end.
```

5. Web への応用

この章では，統計グラフライブラリを Web に応用して “市区町村情報視覚化システム” を構築することを考える (参照：図 6).

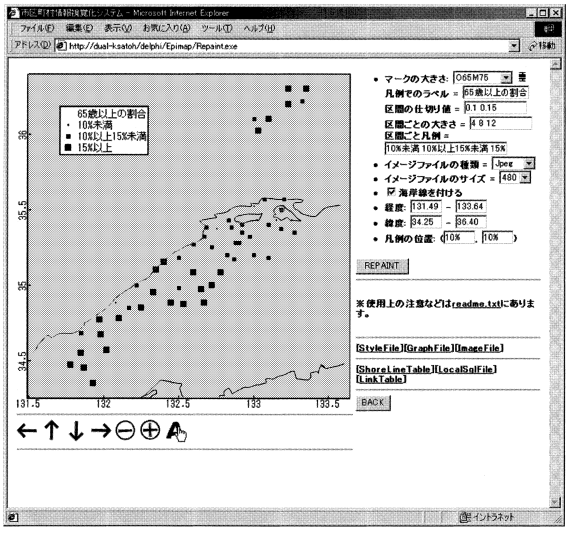

図 6. “市区町村情報視覚化システム” による統計地図の描画．グラフィカルなユーザーインターフェイスを用いてユーザーと対話的に描画範囲などを指定できる．

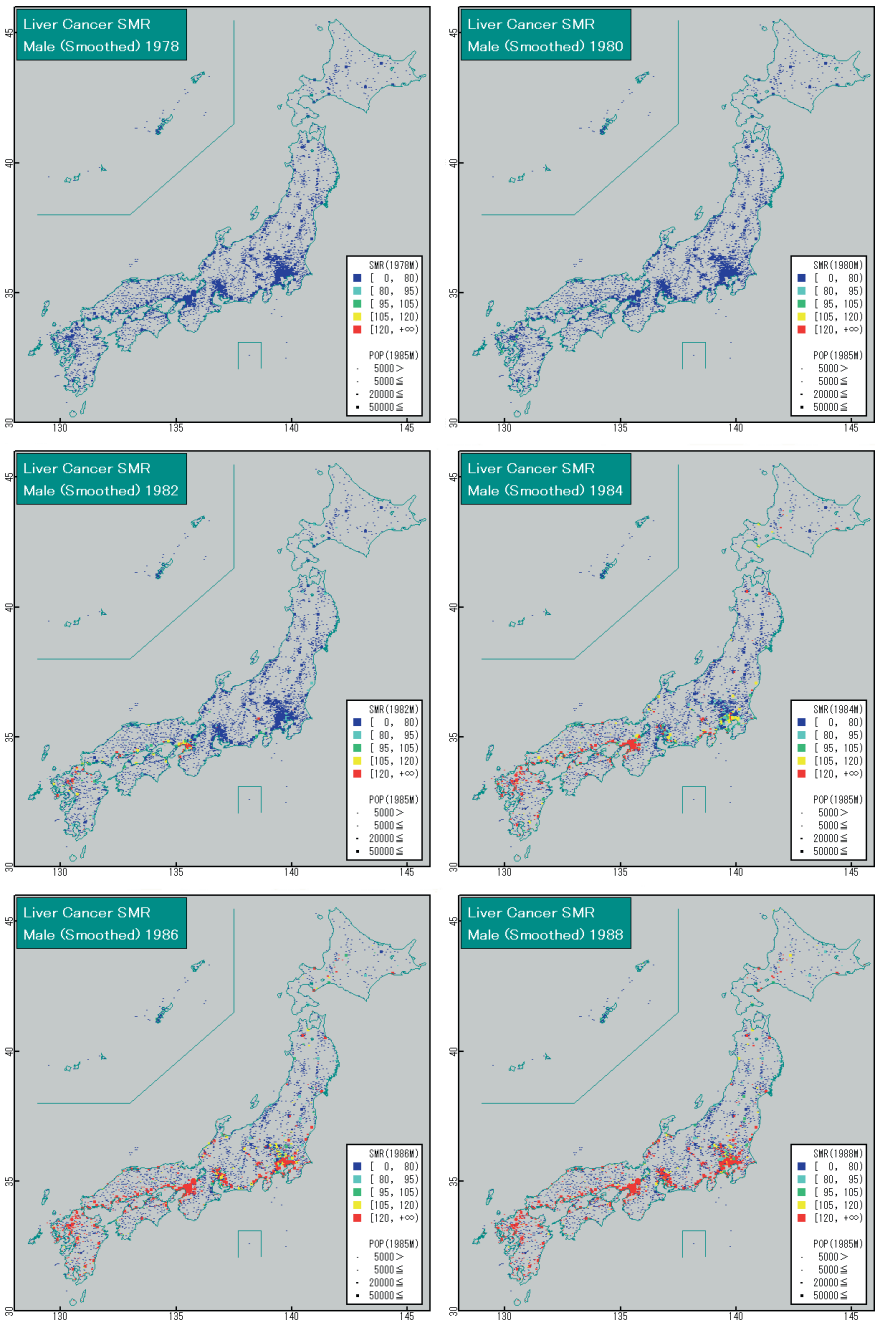

図 7. 本システムによって描画された実際の統計地図．1974 年から 1994 年までの 20 年間における全国の市区町村における肝臓ガンの標準化死亡比 (SMR) データを対象として，SMR の大きさをマークの色で，市区町村の人口をマークの大きさで 2 次元的に表現した．肝臓ガンの SMR が経時的かつ空間的に変化する様子が読み取れる．

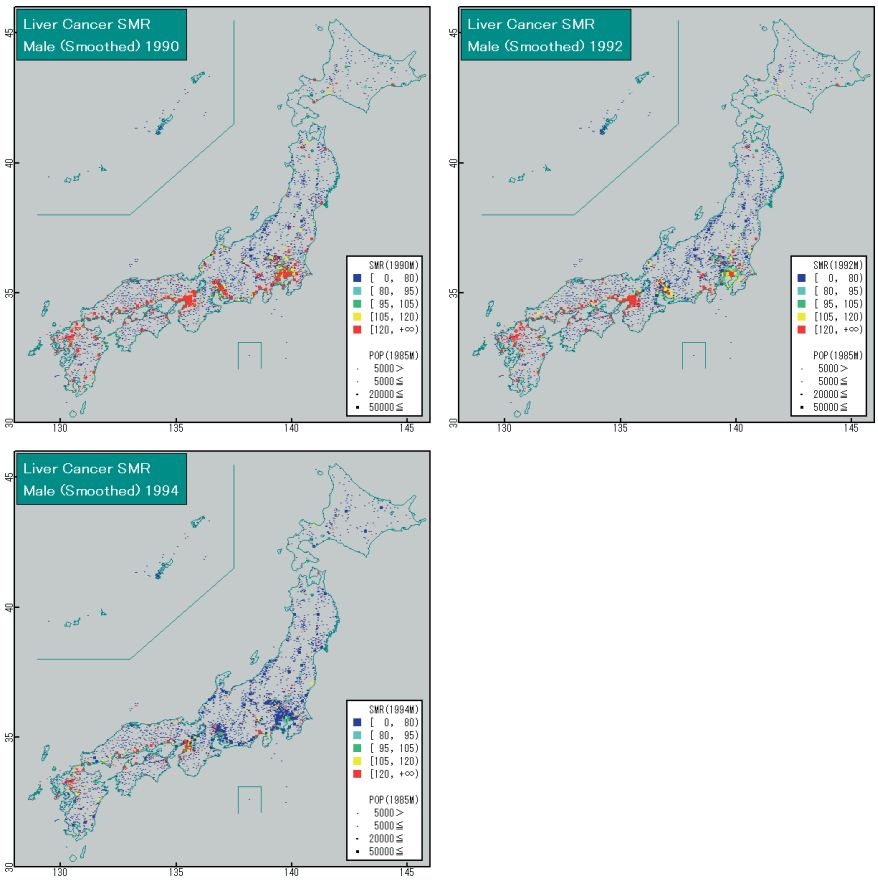

図 7. （つづき）

このシステムは，市区町村ごとに得られるデータを日本地図上に視覚化するものである．このようなグラフは疫学分野において統計地図（参照：福富・橋本（1995））と呼ばれ，対象となるデータの大域的傾向，地域集積性および孤立した異常値を観察するために利用されている．既存の静的な地図作成システムとしては，青木 (1998) による Web 上のサービスが挙げられる．しかしながら，広範囲な多数の市区町村に対してデータが与えられている場合には，描画範囲の移動，拡大縮小，中心補正など，よりインタラクティブな描画システムが求められる．

ここで作成する統計地図は，市区町村に付随する情報と，その役場所在地（緯度，経度）から作られる散布図である．また，各点のマークの大きさと色によって 2 つの情報を表現するものとした．システムで利用するデータには『市区町村番号（全国の市区町村が持つ一意的な 5 桁の整数）の列がある』ことのみを想定し，汎用的なデータについて利用できるように配慮した．

実際にこのシステムによって描画された複数の統計地図，図 7 をここで紹介する．対象となったデータは厚生省から目的外使用を許可された，1974 年から 1994 年までの 20 年間における全国の市区町村における肝臓ガンの標準化死亡比：SMR である．基準死亡率としては 20 年間の全国性別 5 歳階級別死亡率を用いた．より詳しくは（大瀧 他（2000））を参照されたい．

図 8. 市区町村情報データの Web ブラウザによる送信．送信するデータには市区町村番号（全国の市区町村に割り当てられた 5 桁の整数）の列が必要となる．この例では 1 列目の “CityId” が該当し，上 2 桁の 32 より島根県のデータであることが分かる．

地図は左から右，上から下の順に 1978 年から 2 年おきに 1994 年までの男性のものである．マークの色で SMR の大きさが表現され，マークの大きさで町の人口が表現されている．1980 年を境に九州地方から肝臓ガンの SMR が大きくなり，その後，1998 年まで東北地方に向かって北上し，波が引き返すように小さくなっていく様子が読み取れる．疫学的な解説は本稿の範疇を超えるので言及は避けることにするが，このように実際には 4 次元のデータではあるがマークの色と大きさを用いることにより的確な情報を保った地図が描画されていることに注意して頂きたい．

以下に，本システムの概要を利用時の流れにしたがって述べる．特に，ライブラリが使われているステップ 6 については，CGI アプリケーション内の流れを図 1 に基づいて言及した．

ステップ 1. クライアント：市区町村番号の列を含むテキストデータをインターネットブラウザに貼り付け，サーバーにアップロードする（参照：図 8）.

ステップ 2. サーバー：送られたテキストデータをサーバー上のデータベーステーブルファイル（DBASE 形式）として保存する．

ステップ 3. クライアント：市区町村番号の列をコンボボックスにより選択し，サーバーに

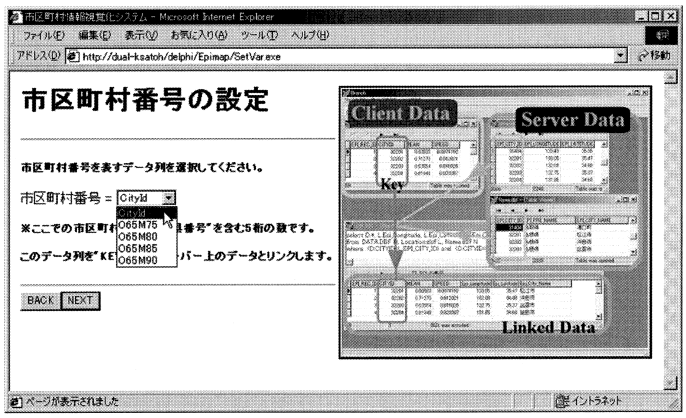

図 9. “市区町村番号” を表すデータ列の選択．市区町村番号を結合キーとして，ユーザーからアップロードされたデータファイルとサーバー上のデータベーステーブル（位置，市区町村名など）をリンクする．

送信する (参照：図 9).

ステップ 4. サーバー：市区町村番号を主キーとしてサーバー上の複数のテーブル（位置，市区町村名など)とリンクさせ新しいテーブルとして保存する．このときのリンク状況をレポートする．

ステップ 5. クライアント：地図上でのマークの色，大きさをどのデータ列に基づいて決定するのかをコンボボックスにより指定する．

ステップ 6. サーバー：地図を描画する (参照：図 6).

(1) CGI を通してユーザーからの要求を受け取る．要求はグラフの描画範囲に関係する情報 (中心修正，移動，拡大縮小)，凡例の情報およびイメージファイルの情報など多岐にわたり，これらのパラメータは次の再描画で利用できるように HTML ファイル内に HIDDEN タグとして記述する．

(2) 地図の描画範囲に基づき，その範囲のレコードを抽出する SQL ファイルを作成する．ここで，テーブルファイル自体は更新されないことに注意する．

(3) SQL ファイル等をデータとして，ユーザーからの要求をもとにグラフファイルを作成する．このとき，SQL ファイルを使うことで，描画範囲に該当するレコードからなるデータだけを内部メモリに読み込み, サーバーの負荷を軽減することを実現している．

(4) グラフファイルを参照して内部的に統計グラフライブラリを用いてイメージファイル，すなわち地図を作成する．

ステップ 7. サーバー: ユーザーからの要求に応じてステップ 6 を繰り返す．

このシステムで用いられる技術は，データベースプログラミング，CGI プログラミングおよびグラフ描画に関するプログラミングである．同種の技術を用いたシステムとして，佐藤 (1997) は WINDOWS NT 上の統計処理システムを，ファイル処理に C 言語，計算部分に FORTRAN, グラフ部分に S 言語，を用いて構成し，また，山本・垂水 (1998) は動的な描画システムを，ファイル処理に PERL 言語，グラフ部分に GNUPLOT を用いて構成している．一方, 本システムは DELPHI のみで構築されていることに注意して頂きたい（参照: 村上（1997), 田原（1998)). アプリケーションの構成としてはデータ受信部，データリンク部およびグラフ描画部の 3 部分に分割されており，その一つ一つが外部 DLL を必要としない単独で動作する高速な実行可能

ファイルとなっている．

ここでは，統計ライブラリを Web アプリケーションに応用することについて議論してきた．しかしながら，ライブラリの適用自体は第 4 章で紹介したコンソールアプリケーションを作る場合と基本的には同じである．この例からも分かるように，ビジュアルな開発環境においては，『見える部分』と『見えない部分』を切り離して開発することが生産性向上の上で重要である．

謝　辞

本研究の一部は，平成 12 年度厚生省科学研究費補助金『統計情報高度利用総合研究事業』(代表: 山口直人)，および『生活安全総合研究事業』(代表: 渡辺　昌)により実施されたものである．

参 考 文 献

青木繁信 (1998)．インターネット上での統計解析支援システムの構築，日本行動計量学会 26 回大会発表論文抄録集，177–180.

Becker, R., Chambers, J. and Wilks, A. (1991). 『S 言語 ── データ解析とグラフィックスのためのプログラミング環境』(渋谷政昭，柴田里程　訳), 1–2 巻，共立出版，東京.

福富和夫，橋本修二 (1995). 『保健統計・疫学』, 南山堂，東京.

村上宣寛 (1997). 『やさしい DELPHI』, 日刊工業新聞社，東京.

大瀧　慈，川崎裕美，佐藤健一，柳原宏和，山口直人 (2000)．ノンパラメトリック平滑化処理による市区町村別 SMR 疾病地図アニメーションの作製，第 68 回日本統計学会講演報告集，261–262.

坂元慶行，石黒真木夫，北川源四郎 (1983). 『情報量統計学』，共立出版，東京.

佐藤整尚 (1997)．Web Decomp の紹介 ── WWW 上で行う季節調整システム ──, 統計数理，**45**, 233–243.

田原　孝 (1998). 『DELPHI で作る高速 CGI』, エーアイ出版，東京.

山本義郎，垂水共之 (1998)．Web 上の統計解析システムの構築 ── CGI による統計処理とグラフ描画の実装, 計算機統計学, **11**, 45–50.

Development of a Statistical Graph Library and Its Application to Web

Kenichi Satoh and Megu Ohtaki

(Research Institute for Radiation Biology and Medicine, Hiroshima University)

With the greatly improved development environments on the Windows OS, it is now easy to create various types of applications, such as console applications, applications with graphical user interface and so on. This paper considers how to construct a statistical graph library that can be generally used in these applications. The actual implementation is explained and we show the Web server application in which the library was embedded.

Key words: Windows OS, Delphi, database, graph library.