Journal of National Fisheries University 58 (3) 191－198 (2010)

データ分析用簡易言語の実装

楫取和明†，瓜倉　茂，青木邦匡

Impementing a Simple Platform for Data Analysis

Kazuaki Kajitori†, Shigeru Urikura and Kunimasa Aoki

Abstract : In our previous works, we have constructed fishery economics database systems. But, since data mining does not end with just databases, the next work for us would be providing efficient methods for further analyzing data from the databases by various techniques of data analysis. However although there are many data analysis applications, we noticed that they are very difficult to learn because of their eager pursuits for versatilities. The versatarities may be achieved by combining many functionalities from applications or independent libraries. So we propose a very simple language for data analysis named Analyze in which many methods of the data analysis applications or libraries are executed in a simple unified fashion.

Key words : Data, Data processing, Fishery data

はじめに

著者たちは，漁業センサスをはじめとする水産関連の電子化データをデータベース化[2,3,4]するとともに，それらを使っての研究[5,6]も行ってきた。

しかし，これらのデータベースのデータ分析機能は独自の拡張SQLに基づく検索機能が主であり，重回帰分析，主成分分析，散布図描画の機能があるものの，網羅的なデータ分析の機能は持ち合わせていない。本格的なデータ分析は別のシステムでと考えたからである。そこで，これらのデータベースから引き出したデータ（やその他一般のデータ）に対して，様々なデータ分析を施す際に便利なように，以下のようなシステムを考案した。

- いろいろなデータ分析用アプリケーションや単機能ライブラリの分析手法が，独自のデータ分析用簡易言語のもとで統一的な方法で使える。

このようなシステムがあれば，一つのデータに対しいろいろな分析を試みることが容易なので，データベースシステムの評価にも適していると考えられる。

一般的なデータ分析を考えても，分析手法は多種多様であり，それをコンピュータ上で実行するアプリケーションも単機能ライブラリもたくさんある。データ分析は往々にしてやってみないとわからないから，いろいろな手法を試してみることができることが望ましい。

しかし各アプリケーションを使うには，それぞれの使用法を知らなければならないが，これが各アプリケーション毎にバラバラであるので，一つでも覚えるのが大変なアプリケーションをそんなに多くは覚えられない。したがって，どれか一つに決めてそれに習熟して何でもそれでやろうとするのだが，やってみると各アプリケーションには得手不得手があって，この用途にはこれ，別の用途にはあれというふうに使い分けたくなるのだ。

また，教育の場において授業で詳しくアプリケーションの説明をする余裕はないから，データ分析用簡易言語が覚え易ければ必要なデータ分析を教える授業で使うことも考えられる。

2009年９月９日受付．Received September 9, 2009.

＊　水産大学校水産流通経営学科（Department of Fisheries Distribution and Management）

†　別刷り請求先（corresponding author）：kajitori@fish-u.ac.jp

つまり，一般のデータ分析において分析用のアプリケーションやライブラリのいろいろな機能を統一的な枠組みの中で使える上記のようなシステムがあればよいと考えられる。そこで，そのようなシステムを実際に設計して実装し使ってみるのが本論の目的である。

なお，システムはLinux上でPerl言語を用いて構築され，またシステムのユーザインターフェイスにはコマンドラインを採用している。同様のシステムはWindows上でも（Strawberry Perl[1]を用いて）構築可能であることを確認している。

システムの設計と実装

簡易言語

我々の提案するデータ分析用の簡易言語をAnalyzeと呼ぶことにする。Analyzeの文の書式とその例を以下に示す。

書式：apply 分析手法名 of アプリケーション名 to データ情報［withパラメータ情報］

ここで「アプリケーション名」には，ライブラリ名も含まれるとする。例文を挙げる：

例：apply principal_component of R to 'data. txt'

これは，フリーの統計アプリケーションであるR[7]の主成分分析機能をデータファイル 'data. txt' にデフォルトのオプション（パラメータなし，つまりwith句省略）で適用する文である。

例：apply bayes_net of weka to 'iris. arff' with 'local. K2 -- -P 2 -S BAYES : SimpleEstimator -- -A 1.0'

（改行は表示の便宜のためのもので，本来は改行無し。以下同様）これは，フリーのデータマイニングアプリケーションであるweka[8]のベイジアンネットワーク学習機能をその固有のパラメータ指定'local. K2 -- -P2 -S BAYES : SimpleEstimator -- -A 1.0' のもとにデータファイル 'iris.arff' に適用する文である。

これらの例で分かるとおり，Analyzeの文は，分析手法名とアプリケーション名とデータ名を指定して，その他の細かい指定は「with パラメータ情報」で指定するという単純構造である。

データ分析用にこのような簡易言語を考案した理由をまとめておく：

- データ分析用のアプリケーション（R，SPLUS，MATLAB，Mathematicaなど）は使い方が難しい。独自の言語，独自のインターフェイス，独自のデータ構造，オブジェクト指向などなど，高機能，汎用性を目指すあまり複雑怪奇である。これで，一つのアプリケーションで何でもできればいいが，それは難しいのが現状である。
- wekaは，Java言語で書かれたデータマイニングに特化したアプリケーションで，GUIもあるがコマンドラインで動かすことをベストの使い方としている。本システムAnalyzeのコンセプトに少し似ているが，分析手法のインプリメントは簡単ではない。
- そこで，これらのアプリケーションやデータ分析用ライブラリをなるべく統一的な方法で簡単に使いたいという要望が自然に出てくる。Analyze言語は，すべての分析を

  apply 手法名 of アプリケーション名 to データ情報 with パラメータ情報

  という枠にはめて実行するのでわかりやすい。また，分析手法をAnalyzeに実装するのも簡単である。
- 汎用スクリプト言語pythonをデータ分析アプリケーションの言語として使うというコンセプトのデータマイニングアプリケーションorange[9]があるが，これはpythonという汎用言語の知識に依存している。Analyze言語は汎用言語に比べ著しく単純である。

ユーザインターフェイス

ユーザインターフェイスは，Analyze言語を直接使う形にしたいので，コマンドラインとする。

Analyzeの文は，Linuxシェル（かそれと同等のもの）のコマンドラインで実行できるが，

```
$ ./analyze. pl "apply..."
```

という形になるのが冗長なので，直接apply文が書けるように，perlでAnalyze用のシェルを書くことにした。Linuxシェル上で

```
$./analyze. pl
```

を実行すると，

```
Analyze>
```

というプロンプトが出るので，

```
Analyze> apply...
```

と書いてEnterすれば実行される。

Analyzeはコマンドラインインターフェイスにはお馴染みのhistory機能を持っているので，上矢印キーで以前入

力した文を呼び戻すことができる。これは入力に関して大変助けになる機能である。

言語の解釈実行

Analyzeの文の解釈実行には，perlのyappを用いた。これはC用のyaccのPerl版である。yappは，字句解析から構文解析を行う中でRやwekaなどのアプリケーションを実行する方法をyappの書式に従って書いたファイル（analyze. yp）を処理して，実際にそれらを行うperlのプログラム（analyze. pm）を生成する。

analyze. pmでは，apply文で指定したアプリケーション名，手法名，データ情報，パラメータ情報を読み取ってアプリケーションを実行するが，各アプリケーション毎にR. pmなどのファイルを作ってこれらの中にperlからのアプリケーションの実行のしかたを手法毎に書いている。

perlからのアプリケーションの実行のしかたは，アプリケーションのAPIに依存するが，いずれのアプリケーションもバッチ（スクリプト）処理には気を配っているので，使えない機能が出てくることは少ない（むしろGUIよりできることが多い場合もある）。

分析結果は，分析手法によって結果を標準化できる場合（主成分分析の場合やXML標準フォーマットがある場合など）は，標準化して結果を表示する。標準化できない場合でもともかく（アプリケーションの吐く）結果を表示する。

結果の保存と継続実行

一つの分析はそれだけで完結するとは限らない。その結果をグラフにしたり，他の分析に使ったりすることが多い。一つのアプリケーション内で分析を実行していれば，結果はそのまま使えるので問題はないが，本システムでは一度アプリケーションの実行を止めるので，分析の継続には結果の保存と継承が必要である。

本システムでは，

- 分析手法によって結果を標準化できる場合（主成分分析など）は，標準化して結果をテキストにダンプする。標準化できない場合でもともかく結果をテキストにダンプする。
- アプリケーションによっては（Rなど）一部結果をバイナリで保存できるので，その場合は同じアプリケーションでの継続を考慮してバイナリでも保存する。

apply文のあと，あらかじめ決めてあるデフォルトの結果を表示するが，それ以外の結果を表示したいときにshow文を用意している：

show 結果名

show文の使用例およびAnalyzeの細かい仕様は次節で示す。

使 用 例

Analyze言語を実行するシステムもAnalyzeと呼ぶことにする。本節のAnalyze使用例の実行環境は，OSはLinux（Fedora 10），perlは5.10.0，Rは2.9.1，LIVSVM（この節の例２で使う）は2.89, wekaは3.6.0である。

例１：各地主要都市の水産物消費に対する主成分分析

まず，家計調査年報の都道府県庁所在市（＋４都市）における，生鮮魚（まぐろ，かつお，さけ，たい，ぶり）の一世帯平均年間消費金額データ[10,11]に，主成分分析を適用する例を示す。これは，標準的な手法であるので，統計アプリケーションならどれでも可能であるしできることに大差はない。よってここでは，パラメータ情報も結果表示も標準化して統一してある（どのアプリケーションを使う場合でもパラメータ指定のしかたが同じ，結果表示項目が同じということ）

まず，

$./analyze. pl

で，Analyzeのシェルに入る。そして，次のapply文を実行する。

Analyze>apply principal_component of R to 'suisan_shohi. tab' with 'scale=TRUE, center=TRUE'

すると，つぎの結果が上のapply文のすぐ下に表示される。

eigen_values

2.415336　1.122466　0.610280　0.511916　0.340004

eigen_vectors

まぐろ

0.550089　－0.120712　0.067064　－0.335819　－0.752036

かつお

0.168102　－0.833904　－0.456168　－0.075854　0.250007

さけ

0.375685　0.538136　－0.708869　－0.164986　0.198882

たい

－0.521329　－0.018394　－0.530587　0.337670　－0.576484

ぶり

−0.506182　0.010224　−0.058265　−0.860366　0.007101

デフォルトの結果表示は，固有値と固有ベクトルとbiplotである。biplotはRplot.epsというファイルで出力される。この例のRplot.epsは次図のようである。

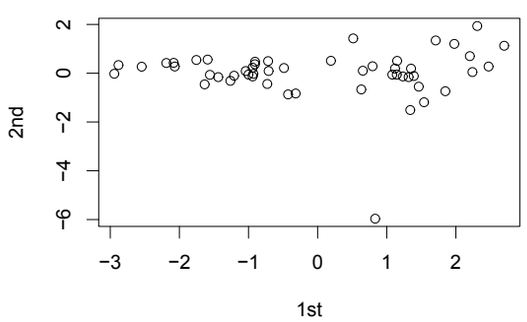

Fig. 1. Biplot of 1 st and 2 nd principal components.

Rのように著名なプラットフォームには，同一手法のインプリメントが複数あるのはよくあることであるが，主成分分析の場合もprincompとprcompがある。prcompの方が高機能であるのでこの例ではprcompを使用している。

'scale=TRUE, center=TRUE'は（prcompのオプション指定と同じで），各変量を分散１にスケーリングする，平均０に中心化するという意味である。デフォルトではどちらもTRUEとしておけば（デフォルトの指定は簡単）この場合with句は省略できる。

上のapply文の実行によって実際に行われることは，次のようになる。まず，apply文がanalyze. pm（analyze. ypをyappで変換した構文解析プログラム）で解釈される。その過程でフックがかかりR.pmの関数principal_componentが引数（'suisan_shohi. tab', 'scale=TRUE, center=TRUE'）で実行されるが，この関数の中でRの関数prcompがオプション 'scale=TRUE, center=TRUE' で実行されて，デフォルトの結果が表示される。

他に，寄与率を表示したければ，上のapply文を実行した直後に

Analyze> show proportion

のようにshow文を実行する。その結果表示は

Kiyoritsu

0.483067　0.224493　0.122056　0.102383　0.068001

上のapply文で，suisan_shohi.tabはデータファイルである。多くの分析手法で共通なのは，データファイルは同一カラム数の行（レコード）が並んだものであることで，ここでもそれを仮定している。しかし，このデータ例でいえば，生鮮魚の名前である「列ラベル」の行の有無，同一行内で列を区切る「列区切り記号」，都市名などの「行ラベル」の有無などは，データファイル毎に異なっているので，一般にはなんらかの指定が必要である。

データファイルのこうしたオプション指定を説明する。

データファイルオプションは，-dに続く '-row 1 -col 2 -sep tab' の形式で書く。これは列ラベルは１行目にあり（データは次の行から始まると仮定），データ行の各行の最初の２列は行ラベルであり，区切り記号はtabであることを示している。列ラベルの行が２行目以下にあることを許すことで，最初の数行はコメントなどの情報を盛り込むことができる。

'-row 0 - col 0' は列ラベル，行ラベルともなしを意味する。-d オプションのデフォルトは '-row 1 -col 1 -sep tab' としてあるが，これは簡単に変えられる。

データファイルオプション -d が指定されているときは，分析手法のオプションは -a を前につけて区別して指定する。例えば，上の例で-d '-row 2 -col 2 -sep comma' を指定するなら，

Analyze> apply principal_component of R to 'suisan_shohi. tab'

with -d '-row 2 -col 2 -sep comma' -a 'scale=TRUE, center=TRUE'

となる。

アプリケーション独自のフォーマット（バイナリ含む），XMLなどは，-d オプション無しで臨み，プログラム側で識別子等で区別する。

例２：サポートベクトルマシン

第２例では，従来の方法に比べ新しいとされる手法の一つサポートベクトルマシン（以下，SVM）を使ってみる。SVMにはいくつか広く使われているコマンドベースのライブラリがあるが，ここではその一つであるLIBSVM[13] を取り上げる。

LIBSVMは，Rなどいろいろなデータ分析用のアプリケーションやPerlからも使えるようになっているが，ここではLIBSVMのコマンドラインインターフェイスを直接使う形で使ってみる。

対象とするデータは，11次漁業サンサス[14,15] から2111漁業経営体の「漁業経営体の基本構成」（属性は次頁表）データである（全漁業経営体2177レコードから「基本構成」データが公開されていない漁業経営体レコードを除い

たもの)。

Table 1. fundamental structure of fishery management unit

漁業経営体の基本構成											
漁業経営体数	無動力船	船外機付船隻数	動力船隻数	動力船トン数	動力船馬力数	最盛期の海上作業従事者数	陸上作業のみの最多従事者				1経営体平均漁獲金額
							家族		雇用者		
							男	女	男	女	

このデータに，SVMのone class ν-SVMを適用してみる。これは，分類と言うより全体の中の外れ値を分出する手法である。以下の一連のAnalyze文を実行する。（途中のAnalyze>プロンプトなしの行は直前のapply文の結果表示である。）

```
Analyze> apply libsvm_format of libsvm to 'gyos.tab' with -d '-row 1 -col 2'
Analyze> apply svmscale of libsvm to 'libsvm. format'
Analyze> apply svmtrain of libsvm to 'libsvm. scale' with '-s 2 -n 0.01'
*
optimization finished, #iter = 23
obj = 118.439739, rho = 12.151561
nSV = 23, nBSV = 21
Analyze> apply svmpredict of libsvm to 'libsvm. scale'
Accuracy = 95.1682% (2009/2111) (classification)
Analyze> apply merge_predict of libsvm to 'gyos.tab'
```

第１文は，上のデータがタブ区切りデータgyos. tabに入ってるのをLIBSVMのデータフォーマットのファイルlibsvm.formatに変換している。gyos.tabの１行目は列名が入っていて，各行の第１，２列は都道府県支庁名，漁業地区名入っているので，これらを除くオプションが'with -row 1 -col 2'である。

第２文は，各項目を−１から１までの数値に揃えるなどのスケーリングを行っている（LIBSVMの機能の一つ)。

第３文は，ν=0.01でone class SVM（-s 2 で指定）の学習を行っている。

第４文は，同じデータに学習結果を適用して，各レコードの決定関数値を求めている。

第５文は，元データgyos.tabに決定関数値を最終列に書き込んだファイルlibsvm_resultを作成している。

作成したlibsvm_resultから，例外とされる地区を取り出し，それらの例外地区からまた特徴的な地区を取り出したりする作業には，SQLが手軽である。そこで，SQLITE[12]（簡易的なRDB）を利用してAnalyzeからSQLで分析結果表を検索できるようにした。

```
Analyze> apply sql_create of sqlite to 'libsvm_result' with -d ' -row 1 -col 2'
```

これで，libsvm_resultをRDBの同名の表libsvm_resultとして格納した。

```
Analyze> apply sql_select of sqlite to 'libsvm_result' with 'c15 = -1'
```

（結果出力は省略）

これで，例外とされる地区のレコードが表示される。この結果はtmpというタブ区切りファイルに保存されるので，上の文の実行直後にtmpを表計算で見れば，「例外」地区は合計102地区あり，うち漁業経営体数が０の地区が相当数あることがわかる。この数は以下の文で確かめられる。

```
Analyze> apply sql_select of sqlite to 'libsvm_result' with 'count(*) : c15 = -1'
84
```

残り18地区を表示するには，

```
Analyze> apply sql_select of sqlite to 'libsvm_result' with 'not c3 = 0 and c15 = -1'
```

（結果出力は下の表に示す）

Table 2. Output for 18 areas

都道府県	漁業地区	漁業経営体数	無動力船	船外機付船隻数	動力船隻数	動力船トン数	動力船馬力数	最盛期の海上作業従事者数	陸上作業のみの最多従事者				1経営体平均漁獲金額
									家族		雇用者		
									男	女	男	女	
北海道	稚内	290	14	707	110	5795.36	30594	872	35	255	371	789	3636
北海道	雄武	165	2	248	77	986.04	12888	702	28	148	439	749	2008
北海道	歯舞	467	35	874	116	1155.43	17303	1199	101	615	170	306	1579
北海道	羅臼	410	39	688	240	2806.85	31437	1565	83	396	570	671	2516
北海道	厚岸	513	6	823	123	1592.75	11707	1304	54	357	106	390	1235
北海道	えりも	489	8	654	59	510.36	6764	1543	30	109	331	455	901
北海道	鹿部	368	1	239	321	1652.94	20227	817	33	411	155	804	752
青森県	八戸	156	2	48	197	21853.57	72399	1713	136	99	171	49	12130
青森県	大間	390	3	215	324	1365.72	21772	672	492	280	15	5	527
青森県	白糠	477	16	97	85	518.71	8412	890	452	390	45	37	168
宮城県	気仙沼	37	1	5	112	34569.64	50150	2180	2	2	76	34	64110
千葉県	金田	418	12	723	35	140.75	1203	882	6	2	1	0	377
東京都	千代田	3	0	0	15	12837.89	19110	460	0	0	8	1	161063
東京都	中央	9	0	5	38	12966.63	17945	506	0	3	2	0	135457
静岡県	焼津	111	1	3	167	19863.5	47407	1514	4	14	74	19	34416
兵庫県	坊勢	363	0	64	636	3499.5	27538	784	1	22	15	54	1665
佐賀県	南川副	164	594	271	197	892.25	9525	610	13	84	52	140	1956
鹿児島県	東町	413	18	186	720	3242.09	44081	1512	1	3	39	56	2833

例３：ベイジアンネットワーク分析

単一の分析手法の実行において複数のアプリケーションをAnalyzeから使う例を挙げる。

ベイジアンネットワーク分析は，まずデータの各変数間の確率的な依存構造を方向付き非循環グラフ（DAG）に表すベイジアンネットワークを学習する過程と，得られたベイジアンネットワークに問い合わせをする過程に分けられる。Rにおけるベイジアンネットワーク分析のパッケージにはgRainがあるが，このパッケージのベイジアンネットワーク学習関数gRainはオプションが少ないので，この部分はwekaのBayesNetを使うことにする。しかし，問い合わせに関してはwekaはわかりにくいのに対しgRainは明解かつ十分な機能が揃っているのでgRainを使うことにする。

よって，wekaで学習したベイジアンネットワークをセーブしてRに読ませなくてはならないが，こうした目的のためにXMLBIFというベイジアンネットワークのためのXMLのフォーマットがあってwekaはこのフォーマットでベイジアンネットワークを保存できるので，これをperlのlibXMLモジュールで読んでRで同等のベイジアンネットワークを生成させることにする。

使用データは，wekaに付属しているサンプルデータcontact-lenses.arffで，コンタクトレンズに関する事例データである。本システムは水産関連データ以外にも当然使えることと，ベイジアンネットワーク分析に適した水産関連データが（例１と例２で用いた漁業センサスと家計調査以外に）手元になかったことで，このデータを使用した。このデータのフォーマットであるARFF形式はwekaのデータファイルフォーマットでありデータ部分はCSVで@DATAと書かれた行の次行から始まるようになっており，属性などデータ以外の情報も書けるようになっている。

contact-lenses.arffの属性は，age（年代），spectacle_prescription（眼鏡処方箋），astigmatism（乱視），tear_drop_production（涙量），contact_lenses（コンタクトの有無，種類）であり，それぞれの値は，age：young（若者），pre_presbyotic（前期老眼期），presbyotic（老眼期），spectacle_prescription：myope（近視），hypermetrope（遠視），astigmatism：no, yes, tear_production_rate：reduced, normal, contact_lenses：soft, hard,noneである。

事例は24個ある。

まず，wekaのBayesNetでcontact-lenses.arffを学習させ，できたネットワークをbayesnet.xmlに保存する。（with句のオプションの意味は省略する。）

```
Analyze> apply learn_bayesnet_xml of weka to 'contact_lenses.arff'
  with 'local.K2 -- -P2 -S ENTROPY : SimpleEstimator -- -A 0.5'
```

（結果表示は省略）

これを，Rで読んで，ageとcontact_lensesの結合（同時）確率を表示する。

```
Analyze> apply infer_bayesnet of R to 'bayesnet.xml'
  with "nodes = c ('age', 'contact_lenses'), type = 'joint'"
  contact_lenses
age                    soft       hard       none
  young          0.08295626 0.08021390 0.1657754
  pre_presbyopic 0.08295626 0.04812834 0.2026144
  presbyopic     0.04977376 0.04812834 0.2394534
```

ここで，tear_production_rateの値をnormalにするという既定事実（evidence）を設定する。

```
Analyze> apply set_evidence of R to 'bayesnet. xml'
  with "nodes = c('tear_prod_rate'), state = 'normal'"
```

evidenceを設定した結果のネットワークは，Rのバイナリ gRain.Rdataに保存される。その上で，再びageとcontact_lensesの結合（同時）確率を表示する。

```
Analyze> apply infer_bayesnet of R to 'gRain. Rdata'
  with "nodes = c('age', 'contact_lenses'), type = 'joint'"
  contact_lenses
age                    soft       hard       none
  young          0.111496309 0.10781048 0.2005275
  pre_presbyopic 0.111496309 0.05821766 0.2450892
  presbyopic     0.008027734 0.01940589 0.1379290
```

これ以外に，テストデータに対する予測なども可能である。

いくつかのアプリケーションの分析手法をAnalyzeから実行できるようにしてみて気づいたことは，Rやweka（やその他データ分析用ライブラリ，アプリケーション）における分析手法の実装には，実装が中途半端でドキュメントが不足しているものが多いことである。

例２で使用したSVMのライブラリLIBSVMには，README，チュートリアル，論文，FAQと，一通りの情報が得られるドキュメントとともに，C（C++）のソースもついているので情報提供は完璧である。Analyzeへの実装のためだけに限らず，分析手法の実装にはLIBSVMのような情報完備が望まれる。

まとめと考察

Analyzeの利点をまとめれば，

⑴ いろいろなライブラリ，アプリケーションの分析手法を，簡単な構造のapply文で統一的な方法で実行できる。

⑵ Analyze自体の実装，およびAnalyzeへのライブラリ・アプリケーションの分析手法の実装がプログラミングとして簡単である。

一方，Analyzeの問題点を述べれば，

⑴ 分析の入出力フォーマットが標準化されていないと，データファイルの指定方法が統一できず，分析結果の再利用やアプリケーション間の連系も難しい。

⑵ apply文におけるパラメータ指定は，主成分分析などを除きアプリケーションのオプション指定に依存しているケースが多く，Analyze言語のレベルで標準化できていない。

⑶ Perl（あるいはこれに代わる言語）を知らないと，Analyzeに新しい手法を実装できない。

⑴の問題点のうち結果の再利用の問題を解決する一つの方法は，Analyze独自のデータ構造を持ちこれをAnalyze実行の間保持することであろう。しかしこれは，独自アーキテクチャのRやその他アプリケーションのやりかたで，これによる実装者やユーザにかかる負担を減らすためにAnalyzeを考案したのであるから，この方法は採らない。

とすれば，前節の例３でのように複数のアプリケーションを使うところでもアプリケーション間をつなぐXMLフォーマットが効果的に使われた（なければ結果の変換にはかなり手間がかかる）ように，分析結果の加工と再利用に関しては分析結果フォーマットの標準化がなされていないと限界がある。

⑴ の問題点の解決には，XML等で入出力データのフォーマットが標準化されることが望まれる。

⑵ の問題点も分析手法の標準化の問題である。各ライブラリ・アプリケーションの機能を生かす点では，無理に標準化しなくてもいいと考えもしてライブラリ・アプリケーションのオプションをそのまま使うケースが多くなったが，整理する余地はあると思う。

⑶ の問題点は，プログラミングの知識がなくてもAnalyzeに手法を追加して使えるのが理想という観点から問題点として述べた。Analyzeへの実装においてアプリケーション毎にある程度決まった書き方ができるので，この辺もプログラムによらずにできたらいいかもしれない。検討課題である。

最後に，以上をふまえて想定されるAnalyzeの使用形態をまとめておく。

⑴ ドキュメントの整ったデータ分析用ライブラリを直接実装して使うのが理想。

⑵ 複数のライブラリ・アプリケーションを使い分ける場合に便利。また複雑なアプリケーションの使い方を覚えるほど使わないが，ある分析は行いたいという場合にも便利。

⑶ Analyze言語が簡単なので，授業での使用にも向いている。

⑷ 分析パラメータの標準化，データフォーマットの標準化がなされればそれらを積極的に活用することで，Analyzeの汎用性を高めることができる。

以後，Analyzeの使用実績を作り改善していく予定である。

参考文献

1）http://strawberryperl.com/.

2）楫取和明，井元康裕，三輪千年：漁業センサスをはじめとする水産関連電子データの利用方法に関する一考察．農林統計調査，50(8)，19-32（2000）

3）楫取和明，瓜倉　茂，青木邦匡：漁業センサスデータベースの更新について．水大研報，56(4)，349-354（2008）

4）楫取和明，瓜倉　茂，青木邦匡：漁業センサスの電子化におけるXML標準の利用について．水大研報，57(3)，183-190（2009）

5）井元康裕，楫取和明：漁業継承性指数を通じてみる漁業の地域構造．地域漁業研究，42(3)，141-156（2002）

6）井元康裕，楫取和明：沿岸漁船階層の構成割合からみた地域漁業構造の態様．49(1)，175-193（2004）

7）R : http://www.r-project.org/

8）weka : http://www.cs.waikato.ac.nz/ml/weka/

9）Orange:http://www.ailab.si/orange/

10）政府統計の総合窓口「e-Stat」: http://www.e-stat.go.jp/SG1/estat/List.do?lid=000001042133

11）統計情報研究開発センター：平成20年家計調査年報集計結果データ.

12）SQLITE : http://www.sqlite.org/

13）LIBSVM : http://www.csie.ntu.edu.tw/~cjlin/libsvm/

14）農林水産省統計部：2003年（第11次）漁業センサス報告書.

15）農林水産省統計部：2003年（第11次）漁業センサス海面漁業に関する電子統計書.