
データ統合と視覚化によるデータ分析ツール
PADOCの提案
2018/07/28
mabonaki0725

目次

1. はじめに(1)

- データ分析とは
  - 論語の温故知新を数理的に実現すること
  - 20世紀後半まではデータが貴重で少ない
    - 分析目的に合わせてデータは記録から作り出した → 結果が有意か検定が難解
  - ネット社会になりデータが莫大 → 未だに分析手法が不定
    - データが運用にあわせて分散されて蓄積
    - 問題の徴候を示すデータが容易に見つからない
    - 業務知識のある人(ビジネスアナリスト)とデータ分析者の協業が必要
    - 外部データ(気候、市場)を使って分析する場合もある
- データ分析するには業務知識に従ってデータを収集・統合する前処理が必要
- データ分析の前処理工程が7割以上占める

1. はじめに(2)

- 無償のデータ分析言語Rやpythonは習熟が難しい
  - それ故RやPythonの前処理用の「前処理大全」が最近出版
- 前処理が容易で視覚的な環境を導入したデータ分析ツールをPADOCを提案

PADOC(Process Analysis by Data Oriented Composition)　Window 7 8 10で稼動

2. データ分析先行ツール

- SAS（有償）
  - 1976年　データが貴重な時代に出現し、結果の検定機能は充実
  - 貧弱な計算資源でも動作する様にデーター行毎に同じ処理を適応
  - グラフィカルな表現機能が劣る。直近のモデルの提供が遅い
- SPSS（有償）
  - コマンドベースと視覚的なGUIによる操作を導入
  - 習熟してくるとGUIは使わずコマンドベースとなる
- S-PLAS（有償）
  - 検定よりグラフィカルな表示を重視　Rに発展
- Matlab（有償）
  - 行列演算を得意とするが　Pythonが高機能になり差別化が難しい
- R（無償）
  - 関数型の記述を採用　大規模な処理は不得意
- Python（無償）
  - オブジェクト指向型の言語　簡潔な表現（メモリー管理や変数型宣言が不要）
- Hadoop（無償）
  - PCの連結による並列処理で大規模データの前処理用に使われる
- SQL(無償)
  - RDBのデータの編集加工ツール　分析機能はない 前処理で最も多く使われる

3. 提案ツールPADOCの機能説明

- データ分析の定義
  - Kaggle(分析精度を競うサイト)でのデータ分析の定義
    - 分析の精度を競うことで数理モデルの実証や習熟できる意義あるサイト
    - 業務知識を使ったデータ選別や収集等の前処理がない
  - 実用的なデータ分析の定義

    総務省の「高度ICT 利活用人材育成プログラム」の定義
    - 実務的なデータ分析を定義として以下の機能を提示

      全体把握　比較検討　仮説検証　知識発見
- 提案ツールの機能説明
  - 実用的なデータ分析として提案ツールの機能を説明

    1. データ前処理
    2. 全体把握
    3. 比較検討
    4. 仮説検定
    5. 知識発見

4. データの前処理

- データ前処理の専門書「前処理大全」は次の区分で記述されている
- データの前処理には業務知識と統計知識が混在する

(1) 抽出

一般に大規模なデータは重複しない様に分散されて蓄積されている
問題の徴候を示すと想定されるデータを探し出す必要がある

(2) 集約

一般にデータレコードは互いに独立である必要がある。
明細レコードの多寡が分析結果に影響する　→　明細を集約するる
データの蓄積時によってリスクに晒されている期間が異なる　→　期間平均化する

(3) 結合

分散されたデータはレコードID（キー）で結合する必要がある
結合後のデータは分散時のデータ定義から離れるので、誤解釈が起き易くなる

(4) 分割

データを結合するとデータの由来より欠損が発生する
（例　財務がある法人と財務が乏しい個人商店を結合した場合）
欠損が出来るだけ少なくなる様に、データ分割して分析する必要がある

(5) 生成　(6) 展開

内部データだけではなく外部データ（市場、気候）を使う場合もある
データを正規分布に近づける様に、ハズレ値の除去や正規化、対数化を図る場合がある

4.データの前処理

- RやPythonでデータ前処理は行えるのか？
  - 課題　　結論
    - データ編集ツールとして簡単　　NO
    - 業務知識がある人でも理解できる　　NO
    - 全体把握するツールがある　　NO

- 「前処理大全」の項目 について
  Python記述と提案ツールPADOCとの比較

  (1)抽出 (2)集約 (3)分割 (4)結合 (5)生成 展開

4. 全体把握(データ前処理) (1)抽出

前処理(抽出)でのPythonと提案ツールPADOCとの記述の比較

項目を選択して欠損のないデータを抽出する

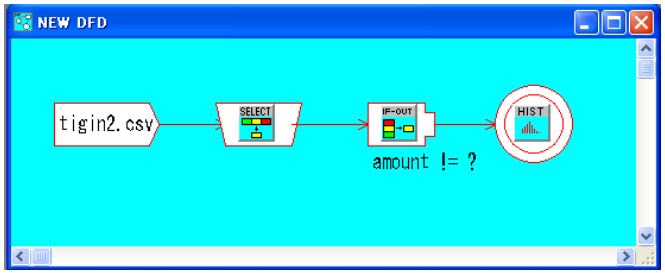
Python

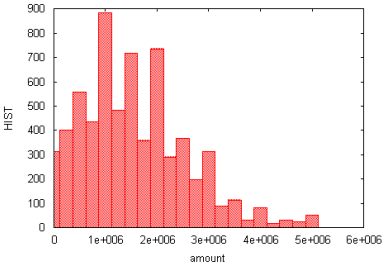
```
#データの呼出し
tigin=pd.read_csv('tigin2.csv')
#項目選択メソッドの使用
select_tb = tigin[['home','amount','job',]]
#欠損削除メソッドの使用
select_tb['amount'].dropna()
```

オブジェクト毎のメソッドを覚える必要がある

PADOC

```
/* データ呼出し*/
get tigin2.csv@;
/* 項目選択*/
select home amount job;
/* 欠損レコードの削除*/
if(amount == ?) delrec;
```





4. 全体把握(データの前処理) (2)集約

前処理（集約)でのPythonと提案ツールPADOCとの記述の比較

区分別に平均値を計算する

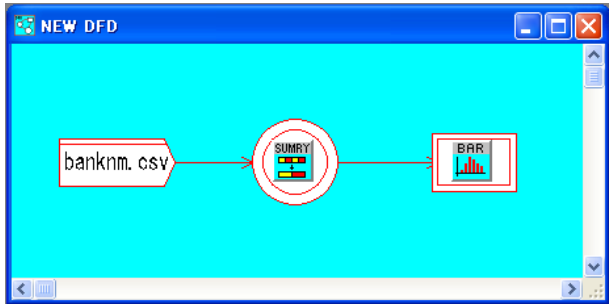
Python

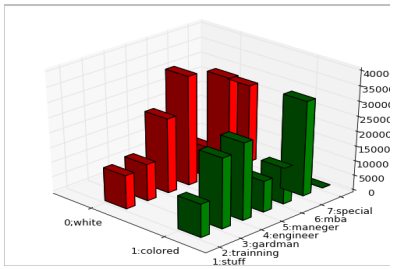
```
#データの呼出し
bankr=pd.read_csv('bankr.csv')
#サマリー 平均
reult = bankr.groupby(['jobcat','minority'])['salnow'].mean().reset_idex()
```

メッソッドの連鎖
記述が長すぎる

PADOC

```
/* データ呼出し*/
get bankr.csv@;
/* サマリー 平均 */
sumup salnow by  jobcat minority/
   method=average
;
```





4. 全体把握 (4)分割

前処理(分割)でのPythonと提案ツールPADOCとの記述の比較

データを均等に4分割する

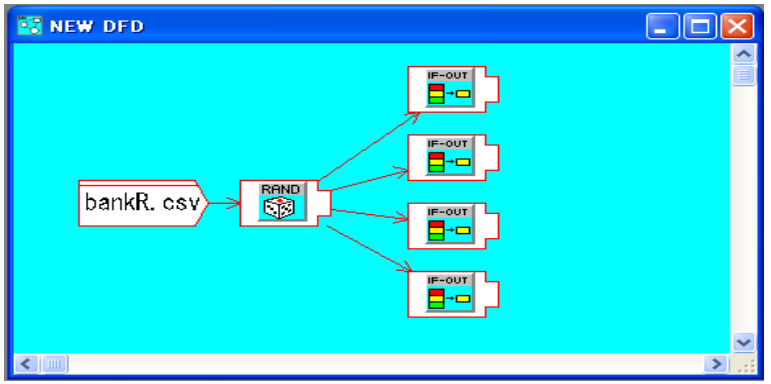
Python

```
#データの呼出し
bankr=pd.read_csv('bankr.csv')
row_no = list(range(len(bankr)))
#4分割指定
k_fold = KFold(n_splits=4,shuffle=True)
#4 分割
for train_cv_no in k_fold.split(row_no) :
  bank = train_data.iloc[train_cv_no,:]
```

特別なオブジェクトを生成する必要がある

PADOC

```
/* データ呼出し*/
get bankr.csv@;
rnd = random; /* 一様乱数付与*/
/* 4 分割*/
      if(rnd <= 1/4) outrec bank1;
else if(rnd <= 2/4) outrec bank2;
else if(rnd <= 3/4) outrec bank3;
else outrec bank4;
```



4. 全体把握(データの前処理) (5)生成

前処理（生成)でのPythonと提案ツールPADOCとの記述の比較

データを対数化する

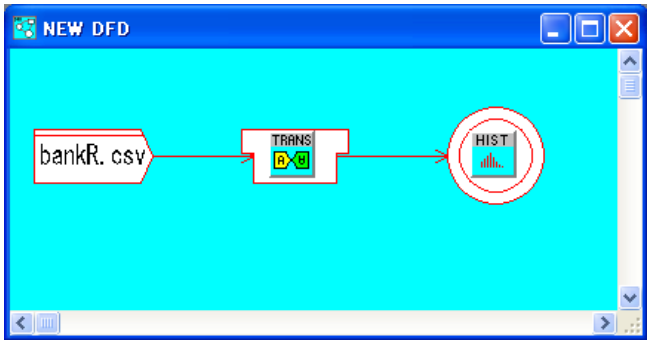
Python

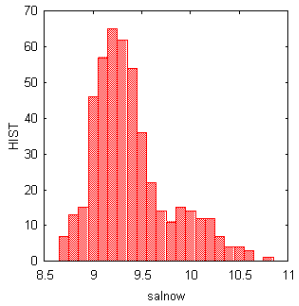
```
#データの呼出し
bankr=pd.read_csv('reserve_tb.csv')
#対数化
reserve_tb[`total_price_log'] = reserve_tb[`total_price']. apply(lambda x:np.log(x/1000+1))
```

記述が長すぎる

PADOC

```
/* データ呼出し*/
get reserve_tb.csv@;
/* 対数化 */
togal_price_log = log(total_price/1000+1);
```





4. 全体把握(データの前処理) (3)結合

前処理(結合)でのPythonと提案ツールPADOCとの記述の比較

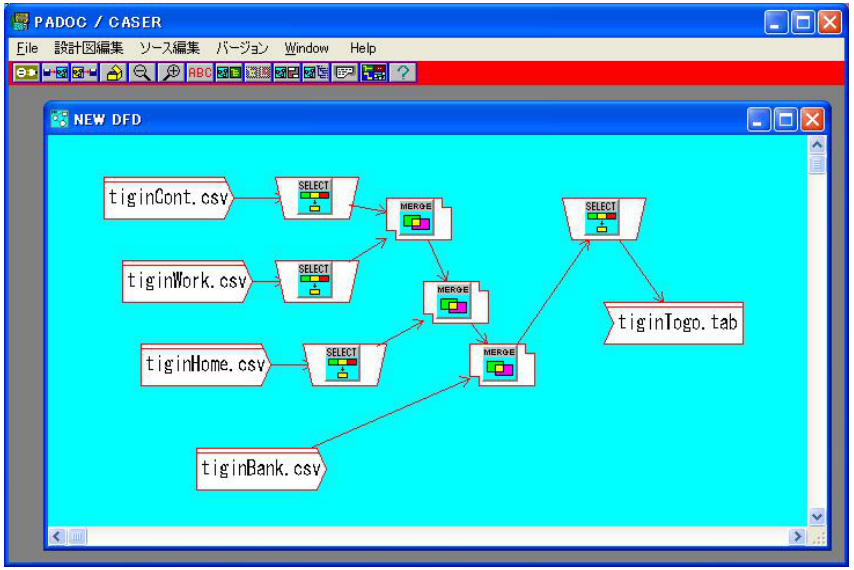
データを結合する

Python

```
#契約情報読込
tiginCont=pd.read_csv('tiginCont.csv')
tiginCont=tiginCont[['id','cont','mon','amount','payrate','bonus']]

#個人情報読込
tiginPerson=pd.read_csv('tiginPerson.csv')
tiginPerson=tiginPerson[['id','sex','old','income','work','job','workspan','workold','homeid']]

#顧客IDでマージ
togo=pd.merge(tiginCont,tiginPerson,on='id',how='outer')

#家族情報読込
tiginHome=pd.read_csv('tiginHome.csv')
tiginHome=tiginHome[['homeid','home','homespan','family','marrige']]

#世帯IDでマージ
togo=pd.merge(togo,tiginHome,on='homeid',how='outer')

#延滞情報読込
tiginBank=pd.read_csv('tiginBank.csv')
tiginBank=tiginBank[['cont','openen','def']]

#契約IDでマージ
togo=pd.merge(togo,tiginBank,on='cont',how='outer')

#結合結果
togo.to_csv('tiginTogo.csv')
```

記述が詳細すぎる

PADOC 処理フローによる結合



5. 全体把握

PADOCには全体把握ツールAIC表がある

(例)
ローン破綻と関係が高い項目のランキングと分布表示

home(持ち家状態)
amount(ローン金額)
mon(貸出し期間)

持ち家状態(home)の分布では
賃貸や借家などの流動性が高い先の破綻率が高いことが示されている

	name	AIC	band	bad	SHR0
1	home	-68.4238	その他	18	0.054545
2			一戸建借家	28	0.177215
3			家族持家	46	0.046843
4			公営ｱﾊﾟｰﾄ	12	0.085714
5			社宅寮	6	0.070588
6			貸間	1	0.142857
7			賃貸ﾏﾝｼｮﾝ	47	0.155629
8			本人持家	105	0.049482
9			民間ｱﾊﾟｰﾄ	55	0.1134
10	amount	-66.337743	C0:50000 -	66	0.035831
11			C1:1030000 -	141	0.076547
12			C2:2350000 -	111	0.119741
13	mon	-38.503975	C0:2 - 36	30	0.028846
14			C1:36 - 60	157	0.087612
15			C2:60 - 121	131	0.073637
16	job	-25.764680	その他	43	0.066052
17			サービス業	55	0.076816
18			飲食	12	0.0458
19			運送	33	0.095652
20			金融	0	0
21			建設土木	104	0.101365
22			娯楽関連	2	0.0909
23			小売卸売	30	0.049751

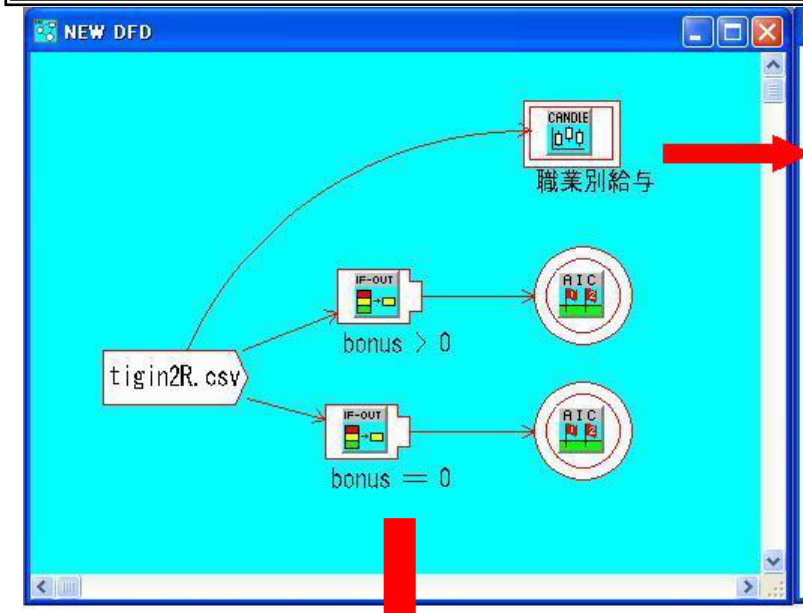
提案ツールPADOCの機能説明

- 提案ツールの機能説明
  - 実用的なデータ分析として提案ツールの機能を説明
    1. データの前処理
       - Pythonに比べて簡潔な記述
       - データ結合の処理フロー編集
    2. 全体把握
       - 全体把握ツール
    3. 比較検討
    4. 仮説検定
    5. 知識発見

5. 比較検討

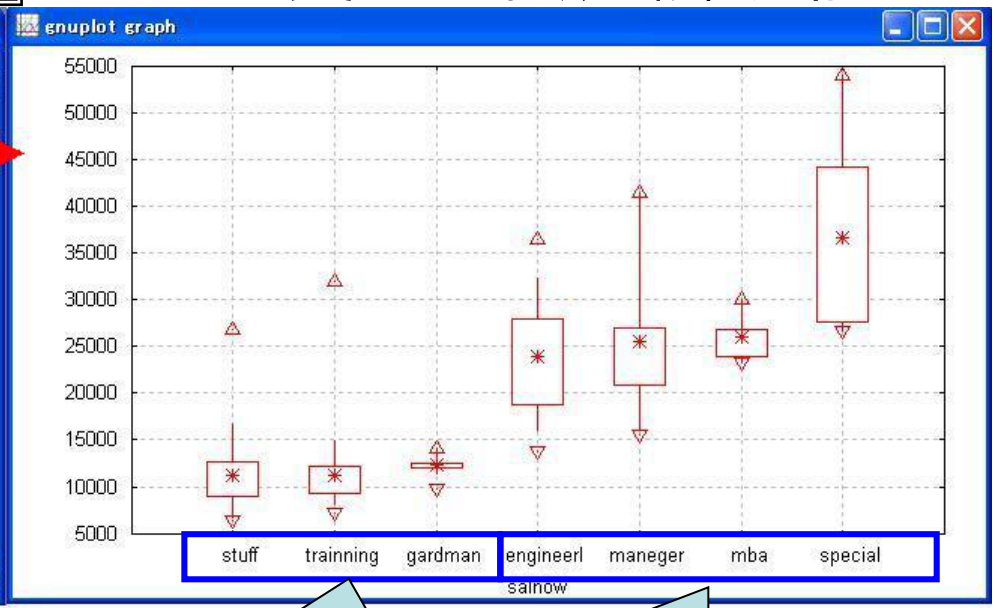
PADOC 区分による比較検討(例1)

区分毎に分布を比較する機能がある



ボーナス払い有無で分割
（次ページ）

ろうそく図　米銀の職種別給与



stuff　一般従業員
traininng 研修中
gardman 警備

engineer　技術者
manager　管理職
rba　RBA
special　特別職

5. 比較検討(例2)

PADOC ファイル分割による比較検討(例2)

データを分割すると全項目で比較できる

ボーナス払い有（従業員）

PADOC VIEWR

	name	AIC	band	bad	SHR0
1	amount	-10.233582	C0:120000 -	6	0.007958
2			C1:1600000	17	0.022849
3			C2:2600000	16	0.041026
4	mon	-8.770352	C0:6 - 48	7	0.009524
5			C1:48 - 60	15	0.044510
6			C2:60 - 120	17	0.020833
7	sex	-2.598178	?	1	0.5
8			女	1	0.0074
9			男	37	0.021131
10	family	-2.199381	C0:?	2	0.005263
11			C1:1 - 3	16	0.029091
12			C2:3 - 4	8	0.023810
13			C3:4 - 8	13	0.0209
14	workspan	-1.999518	C0:0 - 6	10	0.014430
15			C1:6 - 18	24	0.030038

Exit

ボーナス払い無（経営者）

PADOC VIEWR

	name	AIC	band	bad	SHR0
1	home	-68.4238	その他	18	0.054545
2			一戸建借家	28	0.177215
3			家族持家	46	0.046843
4			公営ｱﾊﾟｰﾄ	12	0.085714
5			社宅寮	6	0.070588
6			賃間	1	0.142857
7			賃貸ﾏﾝｼｮﾝ	47	0.155629
8			本人持家	105	0.049482
9			民間ｱﾊﾟｰﾄ	55	0.1134
10	amount	-66.337743	C0:50000 -	66	0.035831
11			C1:1030000	141	0.076547
12			C2:2350000	111	0.119741
13	mon	-38.503975	C0:2 - 36	30	0.028846
14			C1:36 - 60	157	0.087612
15			C2:60 - 121	131	0.073637

Exit

ローンの破綻リスク：従業員は借入金額　　経営者は居住形態が最も高い

提案ツールPADOCの機能説明

- 提案ツールの機能説明
  - 実用的なデータ分析として提案ツールの機能を説明

1.データの前処理
- Pythonに比べて簡潔な記述
- データ結合の処理フロー編集

2.全体把握
- 全体把握ツール

3.比較検討
- 区分による分布比較
- データ分割による全比較

4.仮説検定

5.知識発見

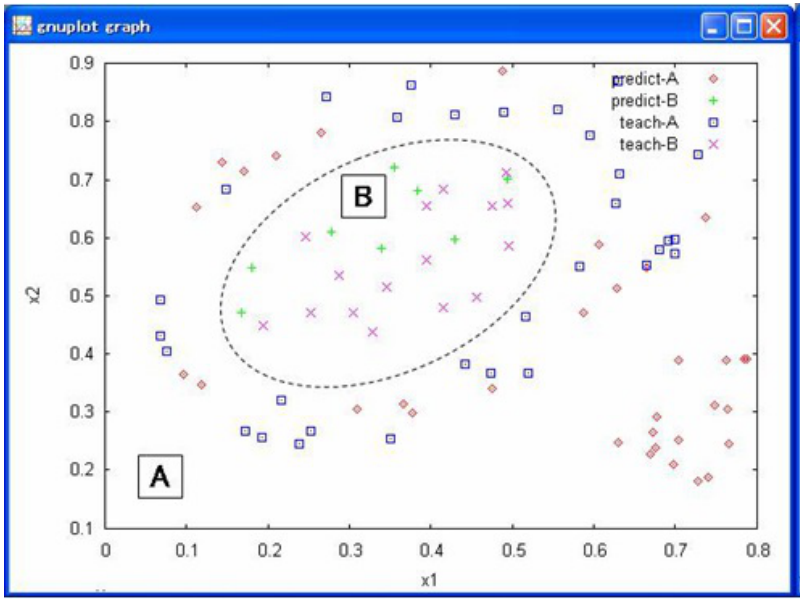
6. 仮説検証

- 分析対象を他の項目で説明する仮説モデルの検証が多い
- 分析対象合う様に予測するので**教師付モデル**と云う
- 検証は予測値と実測値の差で評価する
  - **精度**　：予測値と実測値の差が少ない
  - **頑健性**：運用データでも精度が高い
- 実務的には**説明性**も大事(運用時の問合せに答えるため)
- 精度と頑健性はトレードオフの関係がある
  - 学習データで精度を向上しても、試験データでは劣化する（過学習）
  - これは学習データのノイズで説明するモデルになっているため

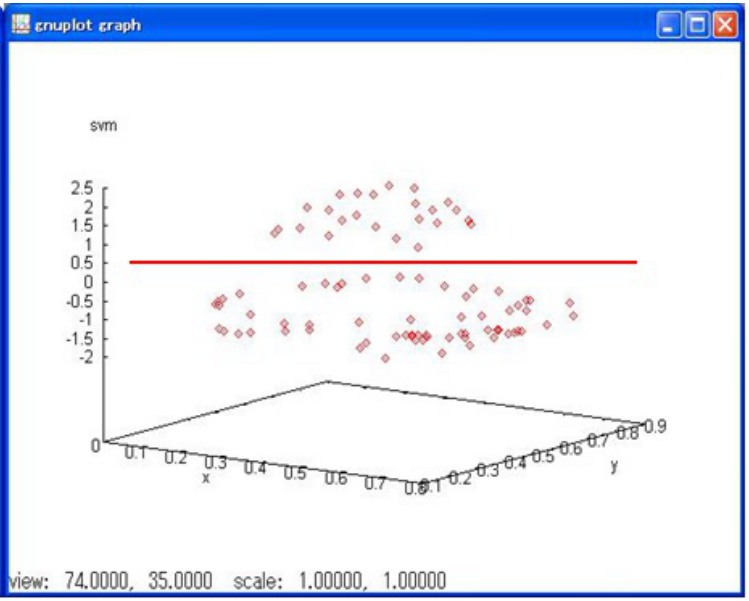
6. 仮説検証

6.1 分析対象項目が2値の場合

- 説明変数が数値の場合　ロジット回帰モデル
- 説明数にカテゴリ値がある場合　判別ツリー
- 非線形的な分離がある場合　高次元に写像し分離　SVM



線形分離不能

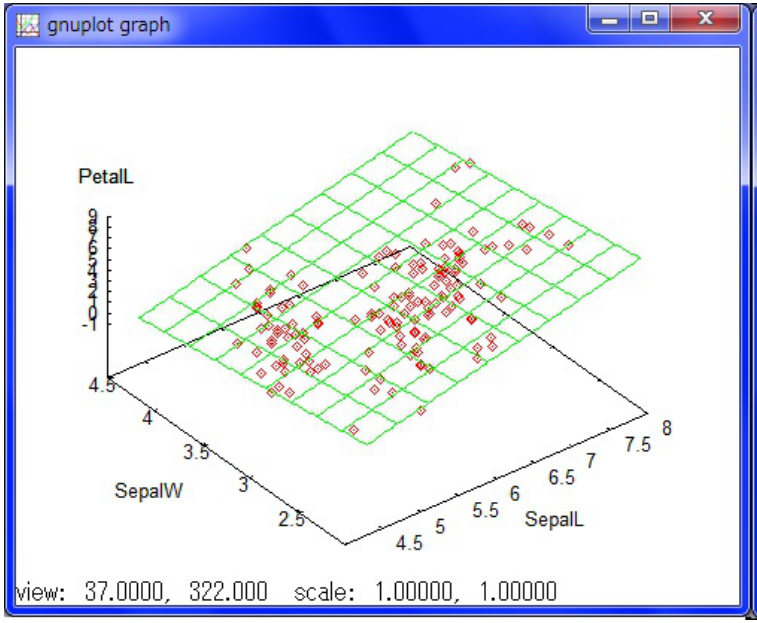


SVMは高次元に写像し線形分離

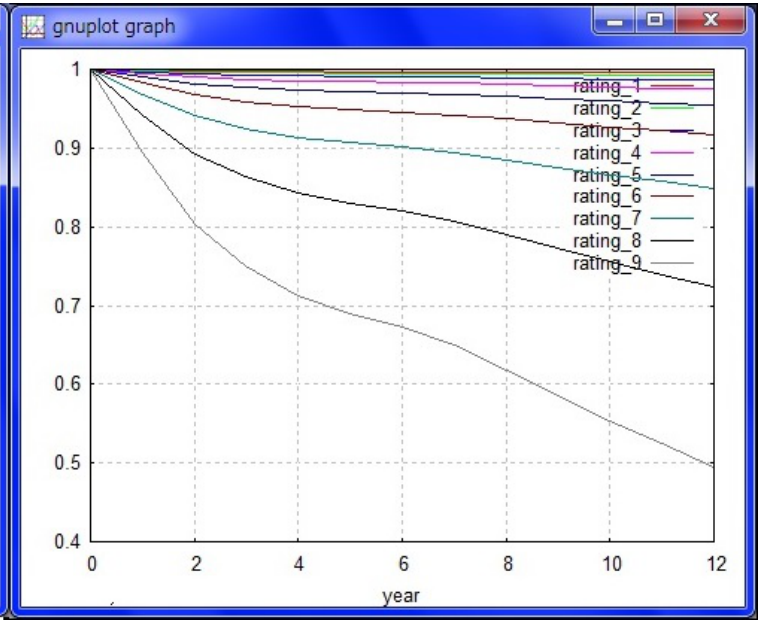
6. 仮説検証

6.2 分析対象項目が実数の場合

- 説明変数が数値の場合　　重回帰モデル
- 分析対象が時間経で劣化するなら　ハザードモデル



3Dの重回帰結果



Cox Hazard モデル

6. 仮説検証

6.2 分析対象項目が実数の場合

- 説明変数にカテゴリ値がある場合　回帰木モデル

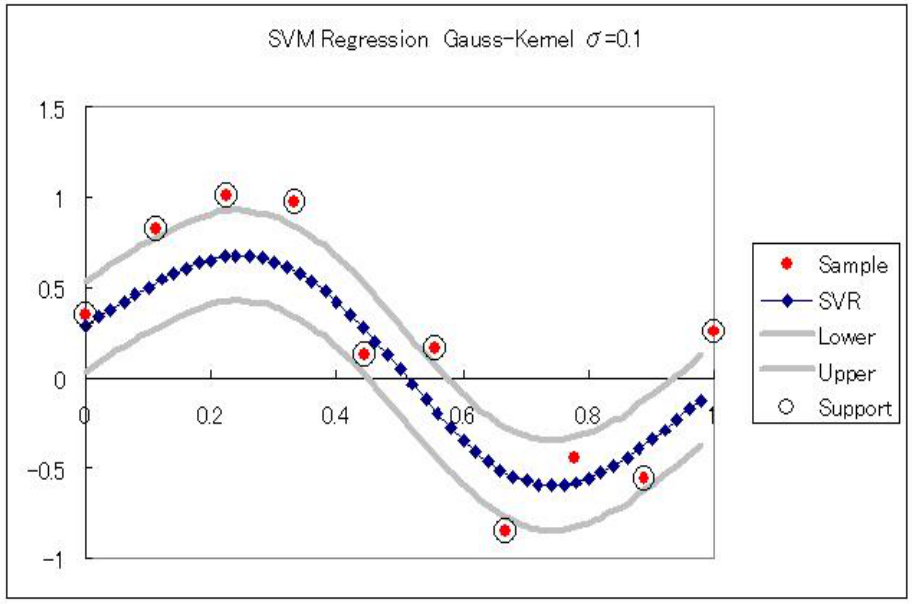
```
TOP:(500.0/SS=31171.1)=9.032599
  ├─住居=一戸建:(99.0/SS=16897.8)=14.343435
  │  ├─勤続年>6:(67.0/SS=15150.6)=16.119404
  │  │  ├─勤続年<=14:(34.0/SS=12614.4)=18.629412
  │  │  └─勤続年>14:(33.0/SS=2536.2)=13.533333
  │  └─勤続年<=6:(32.0/SS=1747.3)=10.625000
  └─住居_in_{その他,アパート,公団,社宅,社宅2,借家,賃貸マンション,分譲マンション,寮}:(401.0/SS=14273.3)=7.721447
     ├─年齢>23:(326.0/SS=13628.9)=8.582210
     │  ├─勤続年>3:(184.0/SS=9250.5)=9.876086
     │  │  ├─仕事_in_{サービス,営業,営業2,教師,事務,自営}:(114.0/SS=6496.0)=11.104384
     │  │  │  ├─月収>240000:(81.0/SS=5422.6)=12.577779
     │  │  │  │  ├─入居年<=6:(47.0/SS=4152.0)=13.802129
     │  │  │  │  └─入居年>6:(34.0/SS=1270.6)=10.885295
     │  │  │  └─月収<=240000:(33.0/SS=1073.4)=7.487879
     │  │  └─仕事_in_{運転手,技術,労務}:(70.0/SS=2754.5)=7.875715
     │  └─勤続年<=3:(142.0/SS=4378.4)=6.905634
     │     ├─住居_in_{公団,社宅,借家,分譲マンション,寮}:(69.0/SS=2755.5)=7.943478
     │     │  ├─仕事_in_{営業,営業2,事務}:(32.0/SS=1813.4)=9.671878
     │     │  └─仕事_in_{サービス,運転手,技術,教師,自営,労務}:(37.0/SS=942.2)=6.448649
     │     └─住居_in_{その他,アパート,社宅2,賃貸マンション}:(73.0/SS=1622.9)=5.924657
     │        ├─世帯人数>2:(42.0/SS=1203.5)=6.969048
     │        └─世帯人数<=2:(31.0/SS=419.3)=4.509677
     └─年齢<=23:(75.0/SS=644.4)=3.980000
        ├─勤続年>1:(38.0/SS=381.4)=4.784211
        └─勤続年<=1:(37.0/SS=263.0)=3.154054
```

世帯プロファイル別のマンション購入希望価格(百万円)

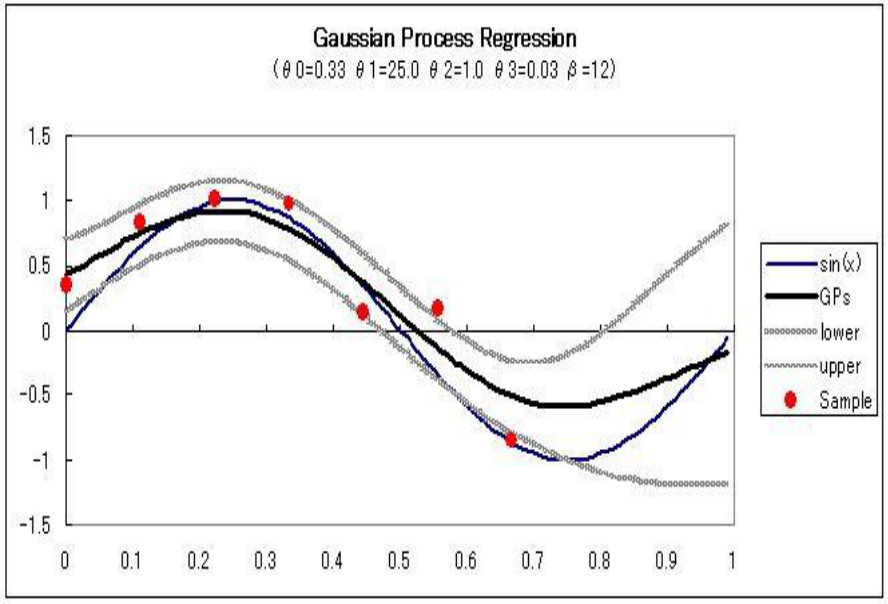
6. 仮説検証

6.2 分析対象項目が実数の場合(小規模データの場合)

- カーネル回帰モデル(信頼区間を表示できる)
  - SVM回帰
  - ガウス過程回帰



カーネル回帰



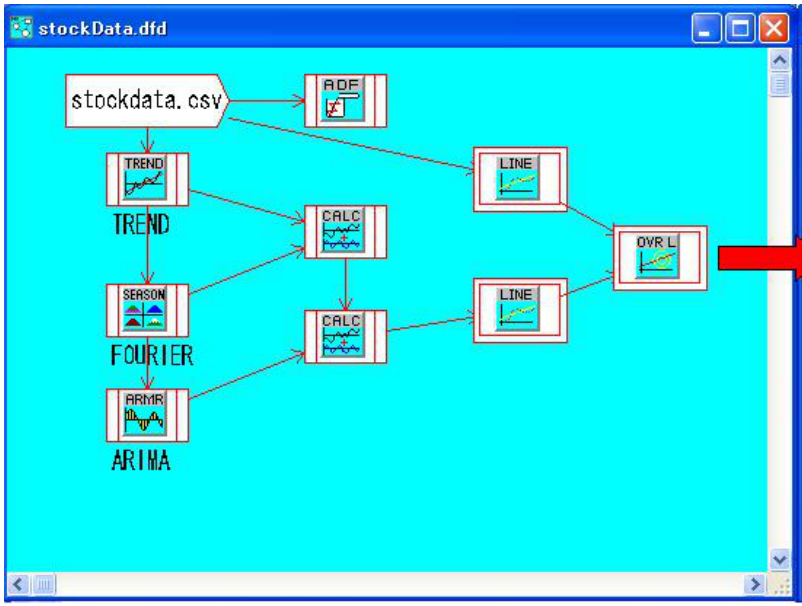
ガウス過程回帰

6. 仮説検証

6.3 分析対象項目が時系列の場合

- 定常波　　ARMAモデル
- 非定常波　カルマンフィルター

下図は処理フローの編集でトレンド成分と低周波（季節）成分を除去し非定常波を定常波にしてARMAモデルで予測した結果



非定常波→定常波　処理フロー図

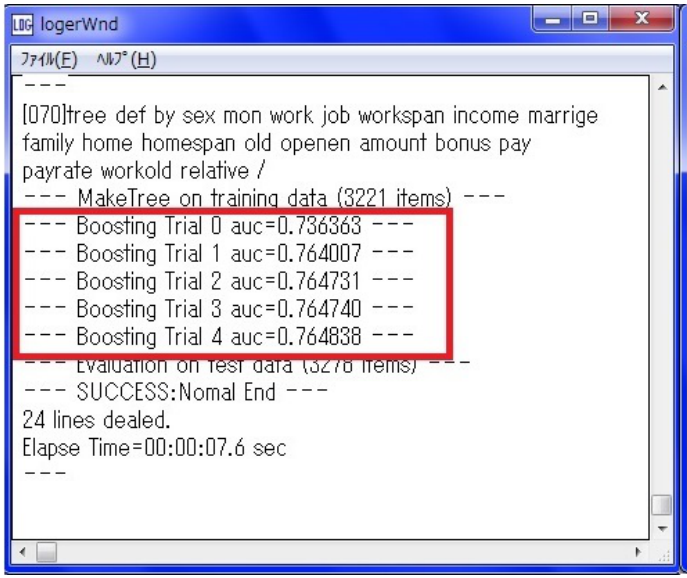
ARIMAによる予測（赤線）

6. 仮説検証

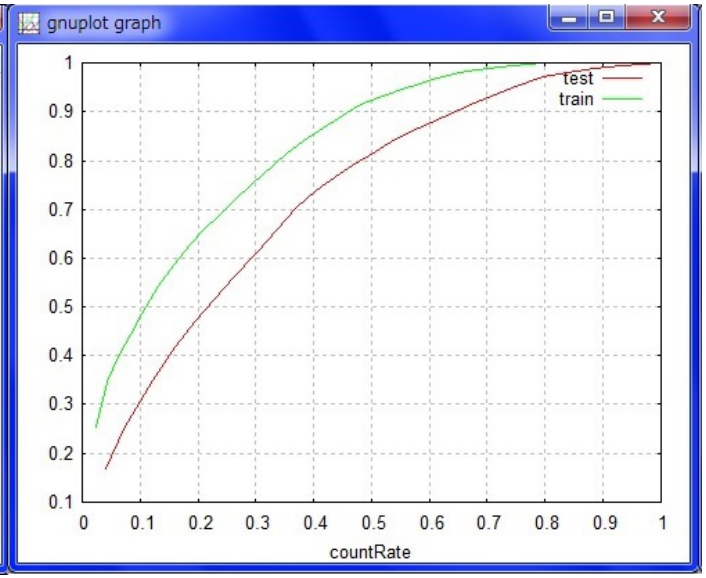
6.4 頑健性の検証

- PADOCではBoostingにより学習データの精度を向上させ試験データで精度比較する機能がある

※Boosting　誤判別の割合に応じて学習データを増やし精度改善する仕組み



Boostingにより精度改善を繰返す

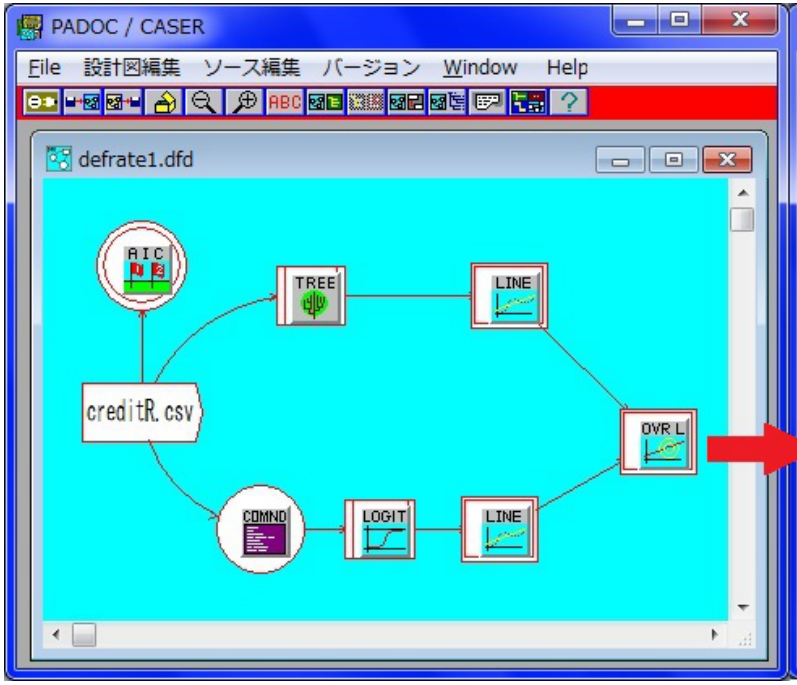


学習と試験データとの精度比較

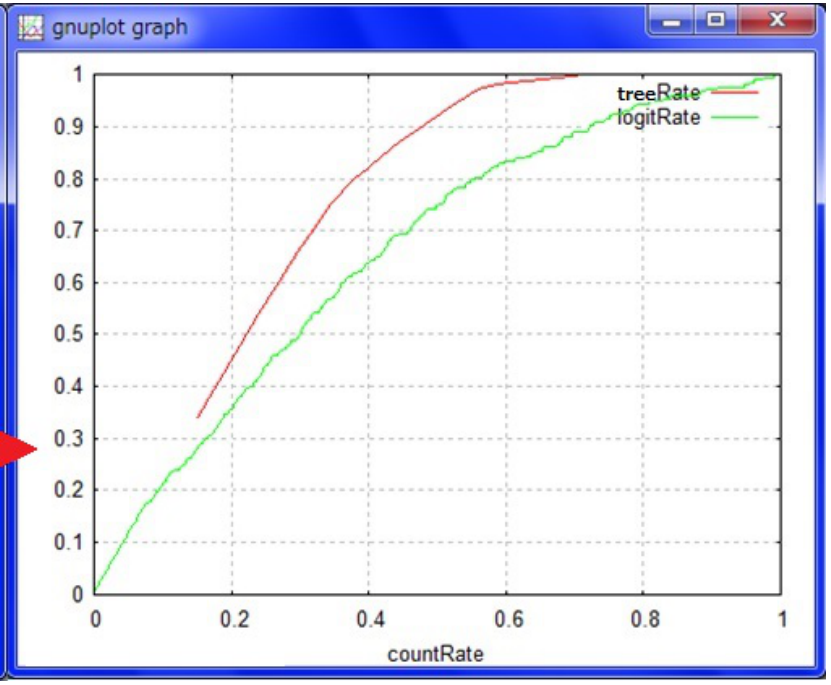
6. 仮説検証

6.5 他モデルとの比較

- PADOCでは処理フローを編集して
  他モデルとの精度比較を視覚的に行える



判別ツリーとロジットの処理フロー図



判別ツリーとロジットの精度比較

提案ツールPADOCの機能説明

- 提案ツールの機能説明
  - 実用的なデータ分析として提案ツールの機能を説明
    1. 全体把握（データの前処理）
       - Pythonに比べて簡潔な記述
       - データ結合の処理フロー編集
       - 全体把握ツール
    2. 比較検討
       - 区分による分布比較
       - データ分割による全比較
    3. 仮説検証
       - 教師付モデル
       - 精度と頑健性の問題の検証
    4. 知識発見

7. 知識発見

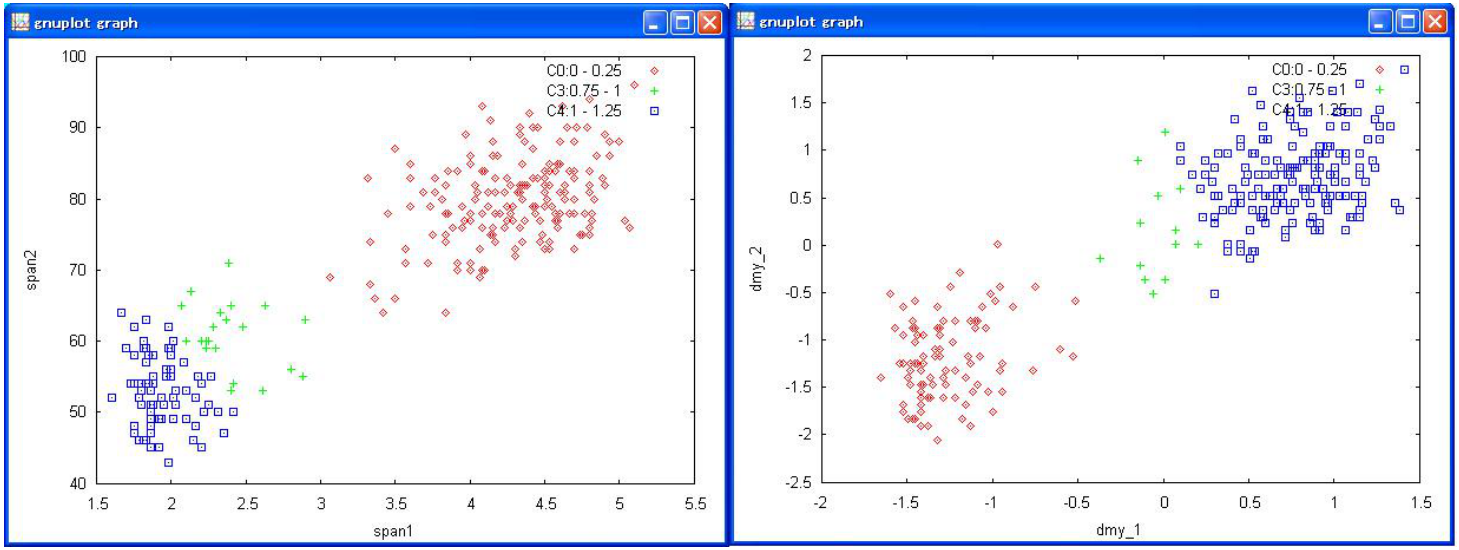
- 一般に知識発見は教師データがない非教師モデル
- データから隠れた要因を推定するモデルが主流
- 結果は図やグラフで視覚的に説明される場合が多い
- PADOCが提供する知識発見のモデル
  - 隠れ変数推定ベイズモデル
  - グラフィカルモデル
  - 最適計画問題
  - 強化学習

7. 知識発見

7.1 PADOC 隠れ変数推定モデル (その1)

1. MCMC　隠れ変数をマルコフ連鎖でサンプリングして最尤値を探索する
2. EM法　　隠れ変数の平均が最尤値を得る様に修正を繰返す
3. 変分ベイズ法　隠れ変数で仮定した変分の下界を最大化して推定する

(例) Oldfaithfull間歇泉の噴射間隔と噴射高のプロット点の所属率を推定



EM法　　変分ベイズ

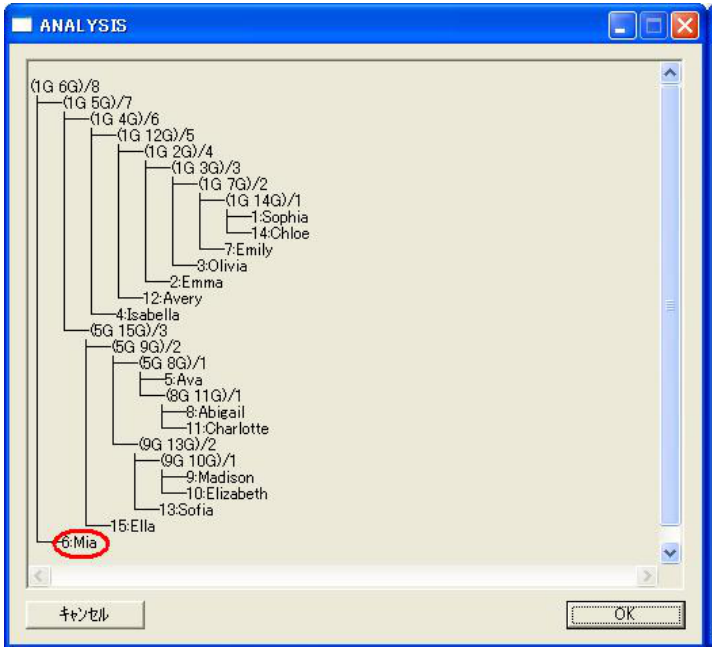
7. 知識発見

7.1 PADOC 隠れ変数推定モデル(その2)

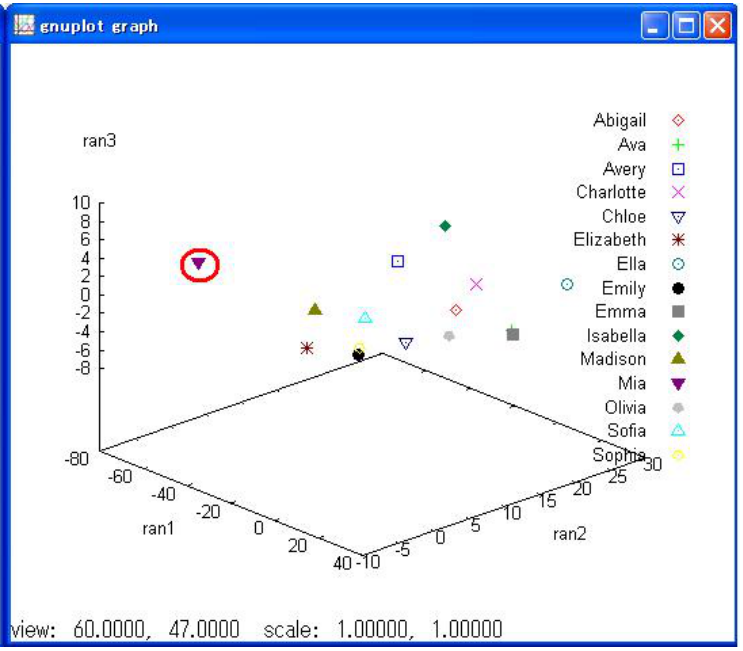
4. 樹系図　近傍点間を樹系図で表示

5. LDA　　言語の類似状態(トピック) を推定する

(例)学生14人の3教科の成績　両図ともMiaがハズレ値を示す



樹系図



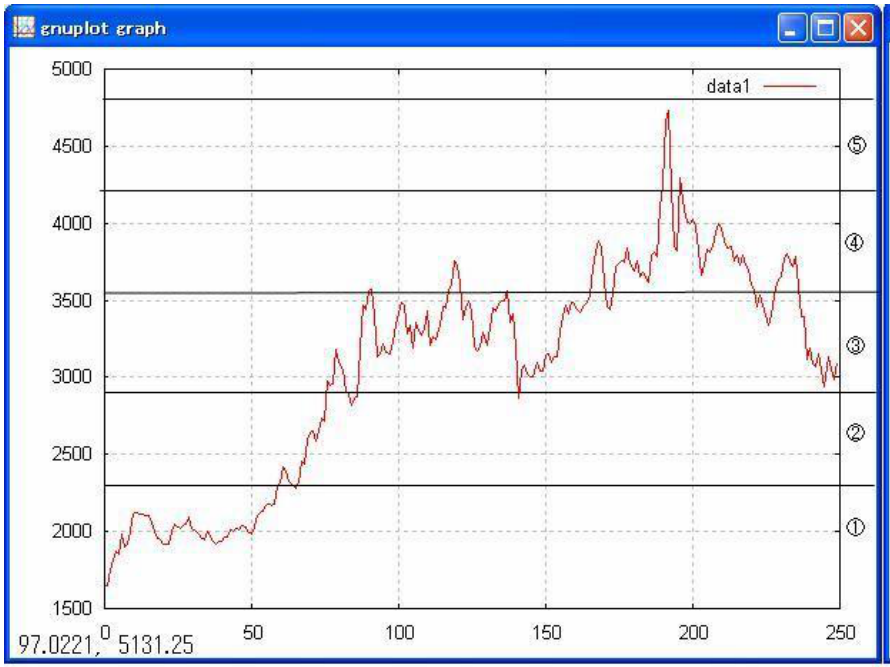
3Dプロット

7. 知識発見

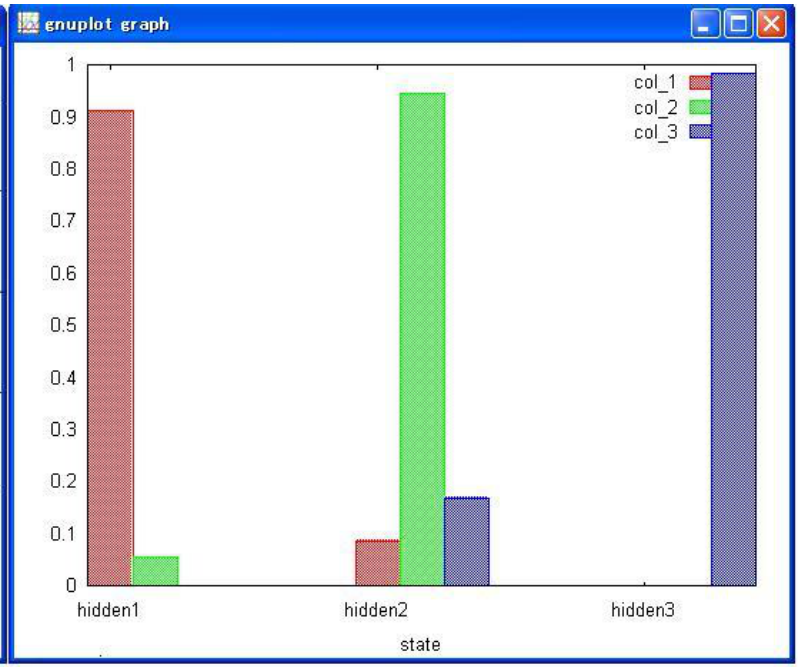
7.1 PADOC 隠れ変数推定モデル（その3）

6. 隠れマルコフ　時系列の隠れモードを推定する

(例) 富士通の株価推移を隠れマルコフで隠れモードを推定



富士通の株価推移



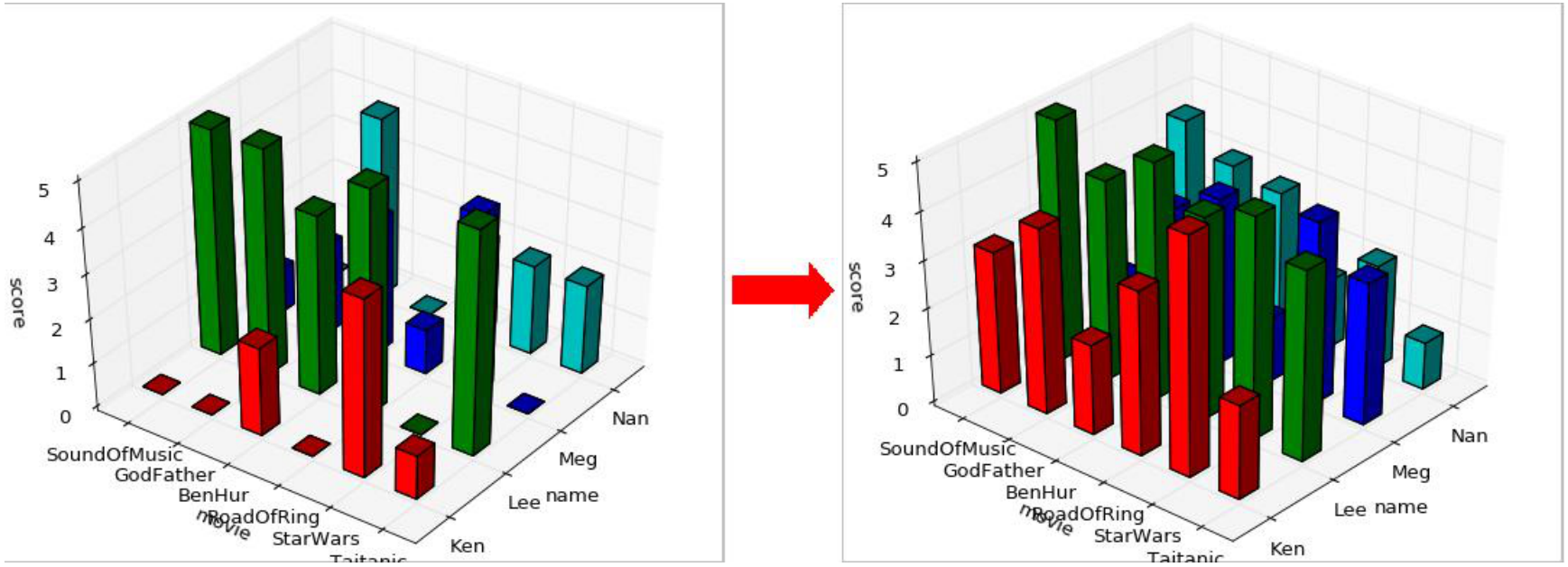
隠れマルコフ　低中高のモードがあり遷移が起り難い

7. 知識発見

7.1 PADOC 隠れ変数推定モデル（その6）

8. Group Lense　未使用の評点を人と商品の類似により推定する

(例) 未鑑賞の映画の評点を類似により推定する　未鑑賞の推定評点が高ければ推奨できる



4人の5本の映画の評点（未鑑賞がある）

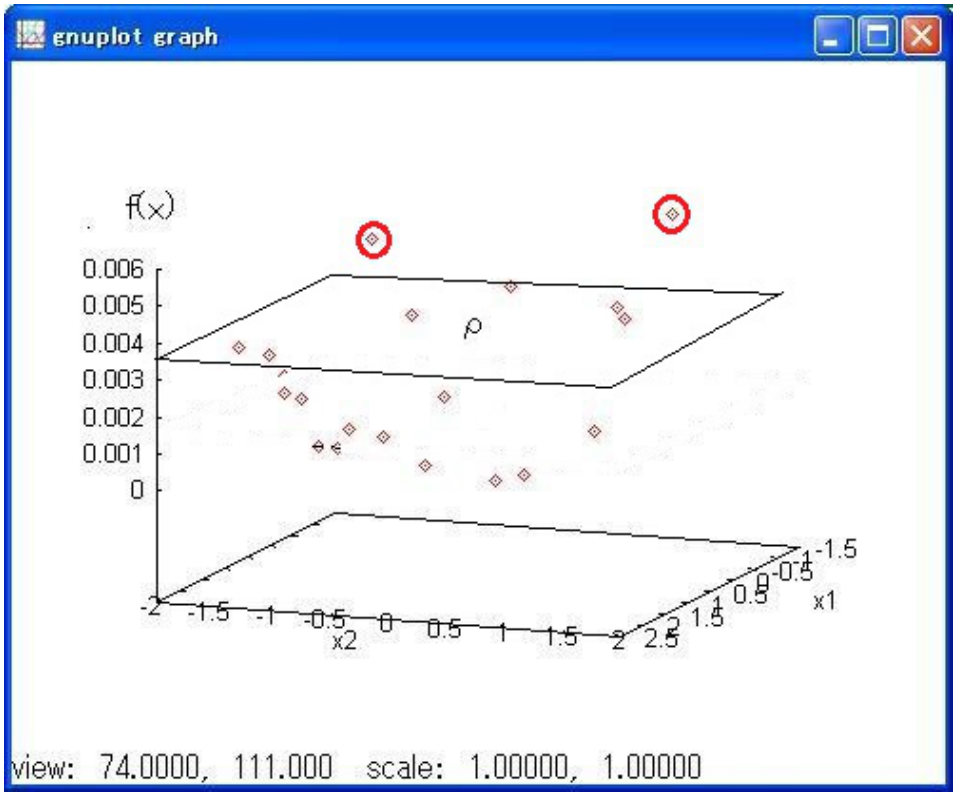
未鑑賞の映画の評点を類似により推定する

7. 知識発見

7.1 PADOC 隠れ変数推定モデル（その5）

7. One Class SVM　ハズレ値を推定する

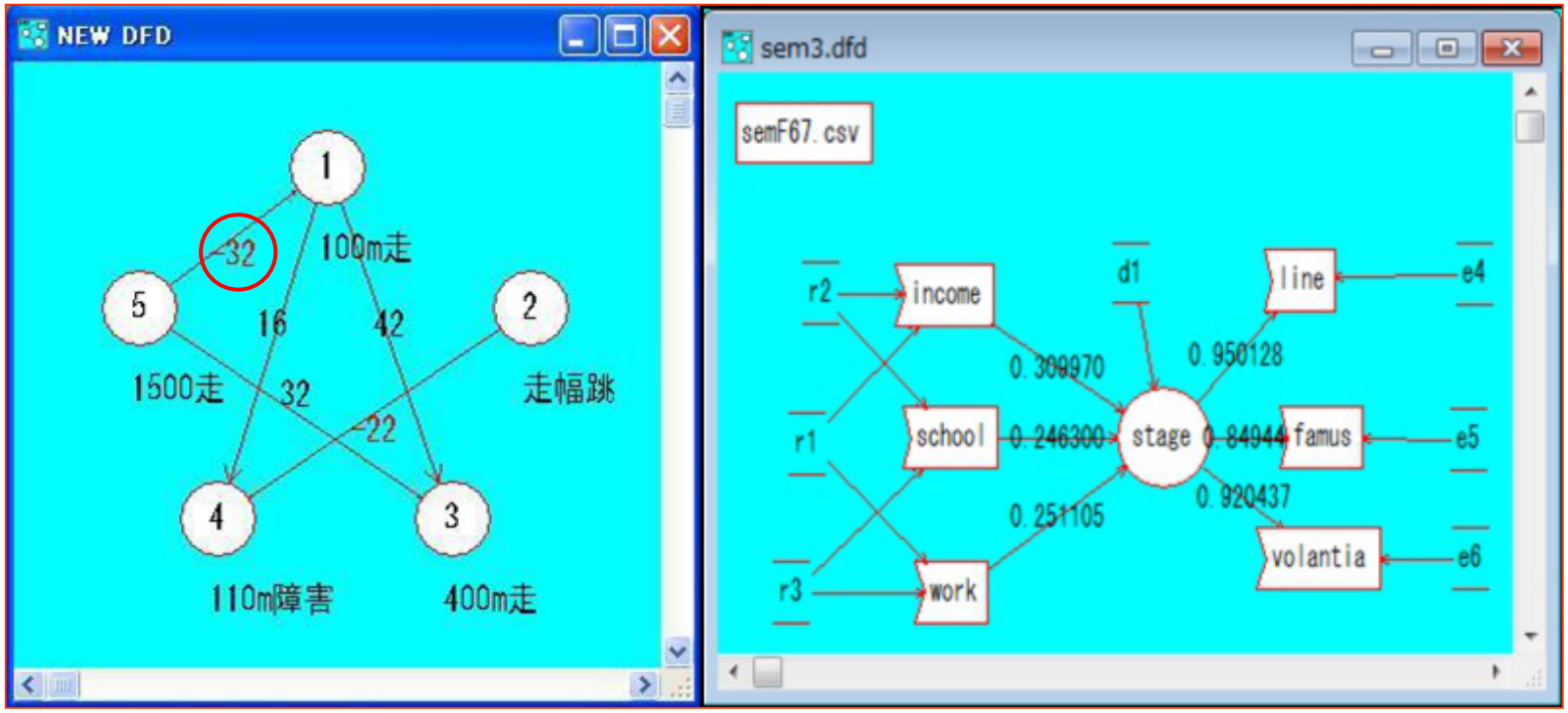
SVM 原点から最大マージンを採る様に高次元に写像し，ハズレ値を摘出する



7. 知識発見

7.2 PADOC グラフィカルモデル (その1)

1. ガウシアン･グラフィカルモデル(GGM)　見かけ上の相関を排除した関連図
2. 共分散構造 モデル(SEM)　項目間の関係を隠れた因子で説明する図



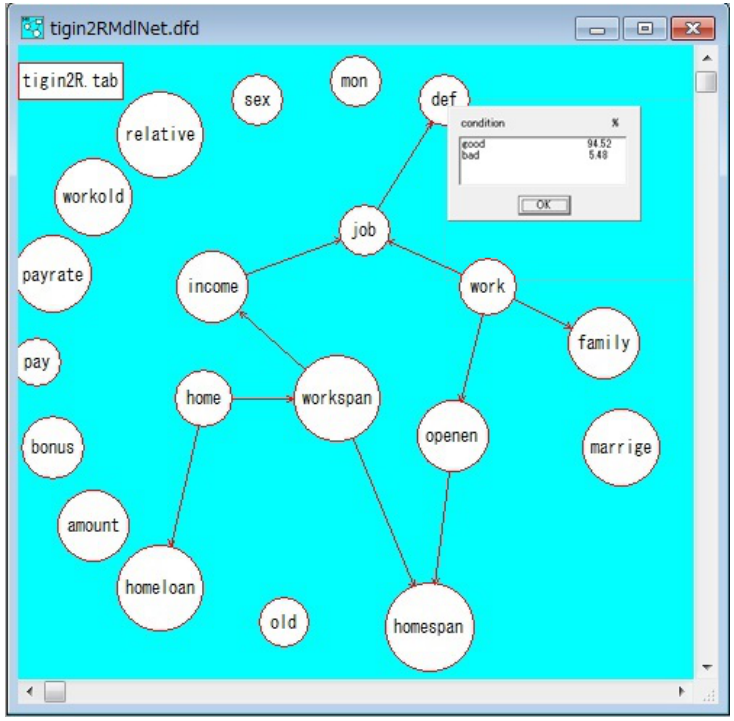
GGM 陸上近代5種競技の関連図　SEM　社会活動データをstage(地位)で説明

7. 知識発見

7.2 PADOC グラフィカルモデル (その2)

3. ベイジアンネットワークモデル　データからベイジアンネットを自動生成する

矢印に沿って確率伝播するので、上流の条件を変更すると確率が変わる

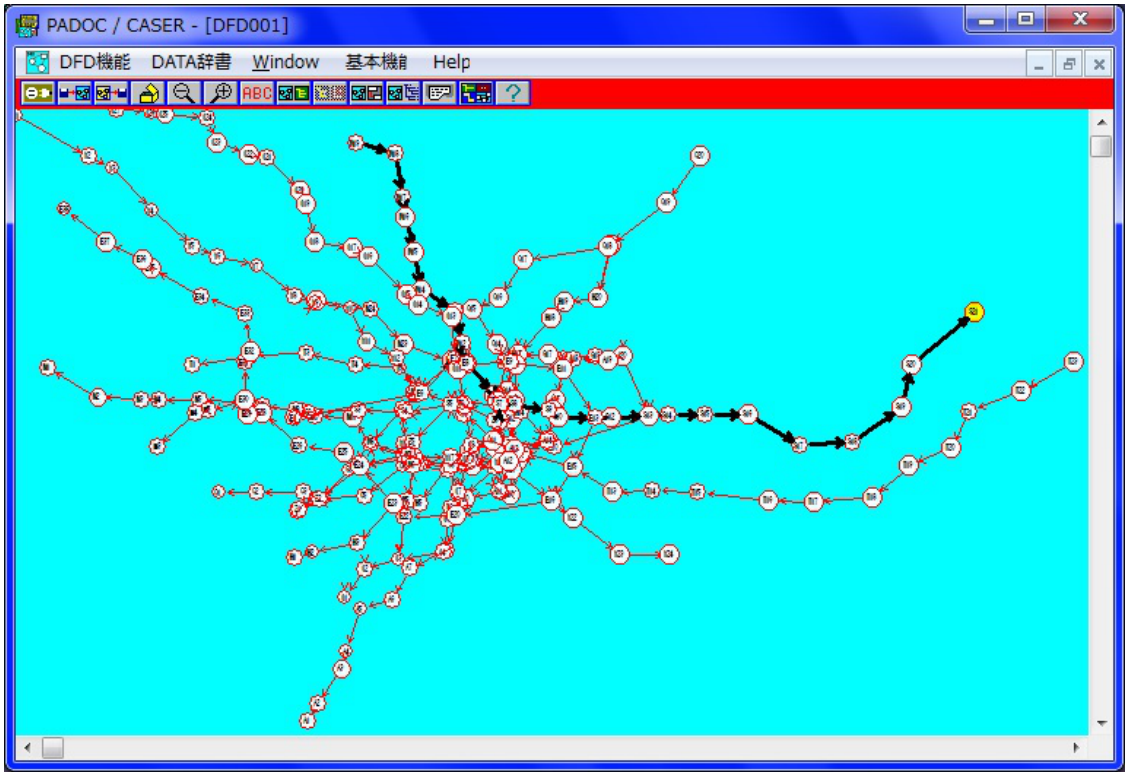


ベイジアンネットワーク：ローン破綻(def)の確率を表示

7. 知識発見

7.2 PADOC グラフィカルモデル (その3)

4. 最短経路を表示する　ダイクストラ法

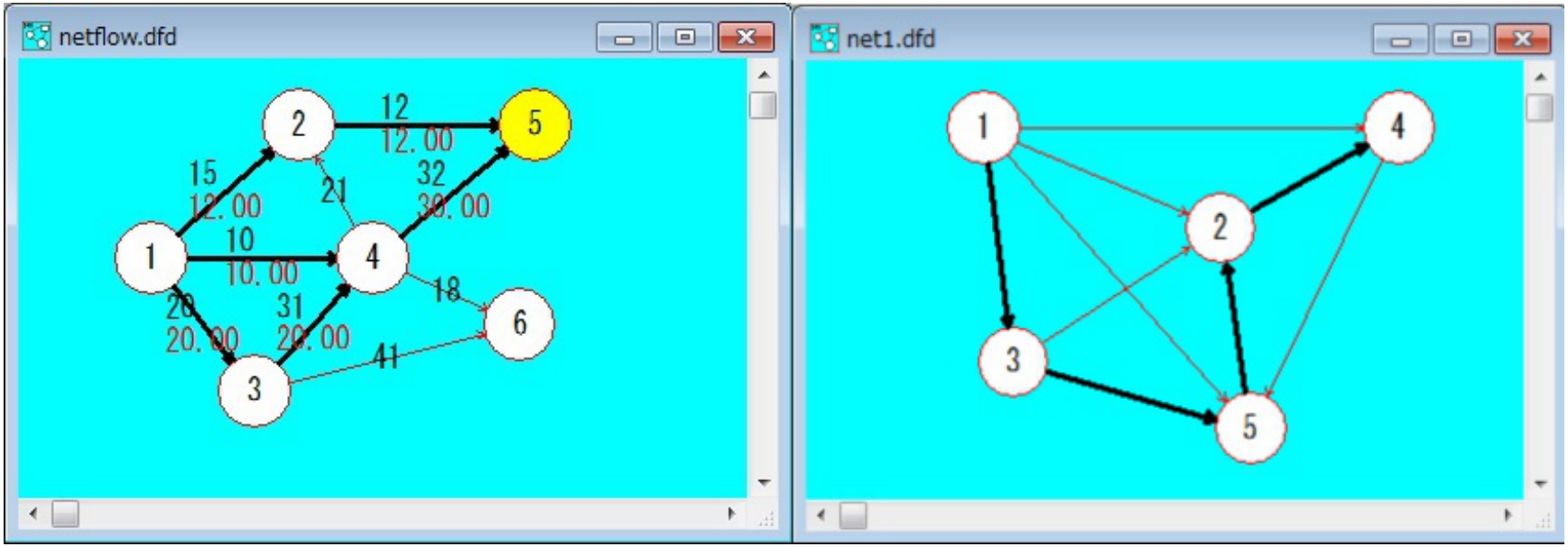


東京地下鉄の位置を読込み　始点と終点を指定すると最短経路を表示する

7. 知識発見

7.2 PADOC グラフィカルモデル (その3)

5. 最大流入(流出)問題　配管網の各配管の容量内で最大流入量を計算
6. 最大張り木　全ノードに送電する最短送電経路を表示する

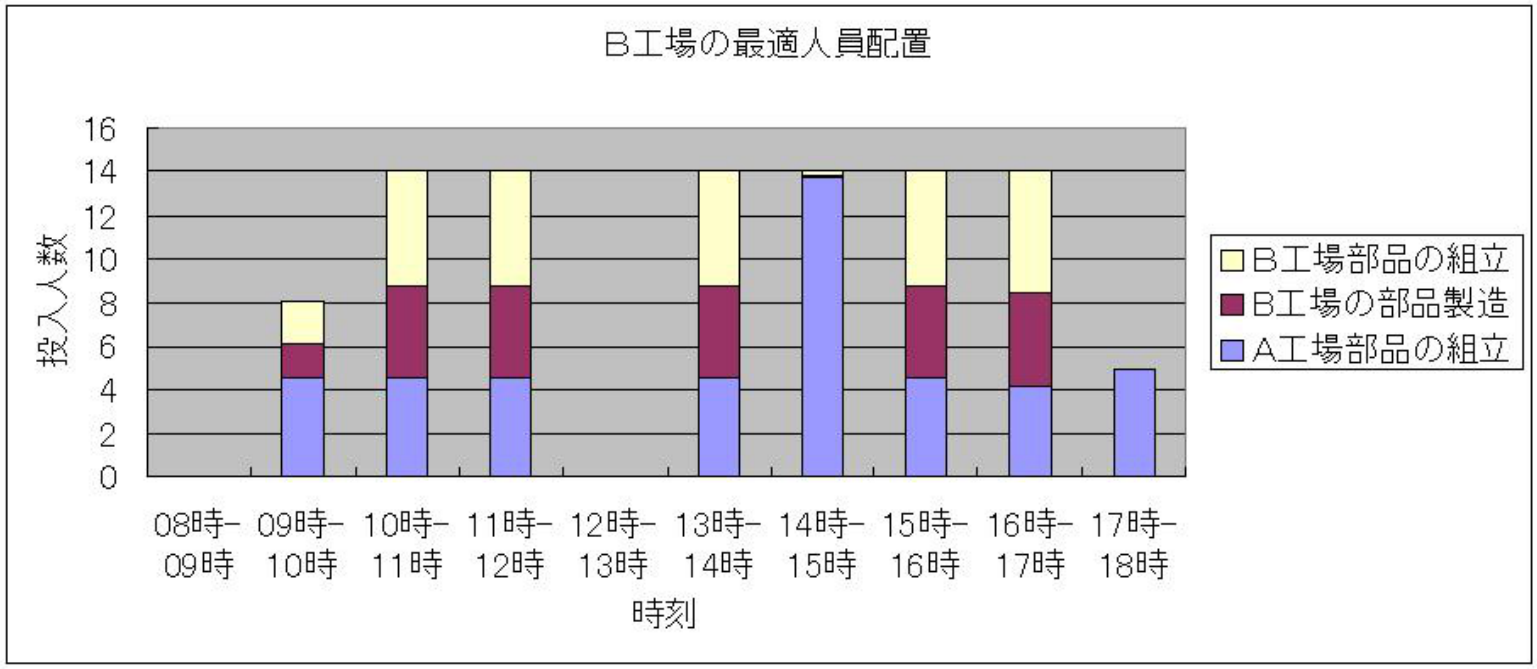


左図:最大流入図　①が流入点　⑤が流出点　黒字:配管容量　赤字:流入量

7.3 PADOC 最適問題 (その1)

1. 線形計画法

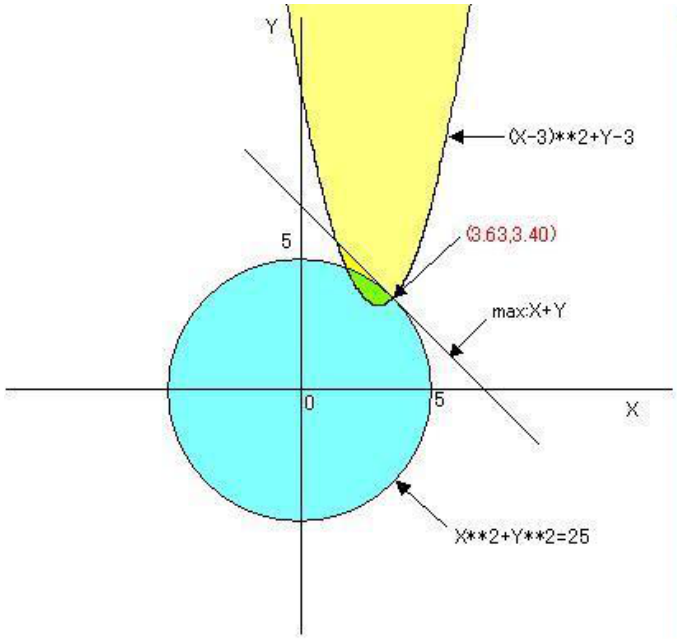
(実用例) A工場は部品製造専門　B工場は部品製造とAとB工場の部品の組立て行う
B工場の最適人員割当を求める



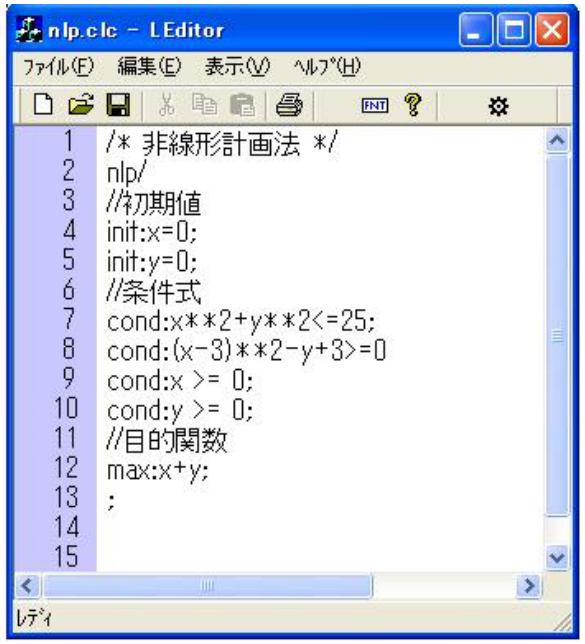
7. 知識発見

7.3 PADOC 最適問題 (その2)

2. 整数計画法（ナップザック問題） 最適解が整数　分岐制限法を採用
3. 非線形計画法　SUMT法採用



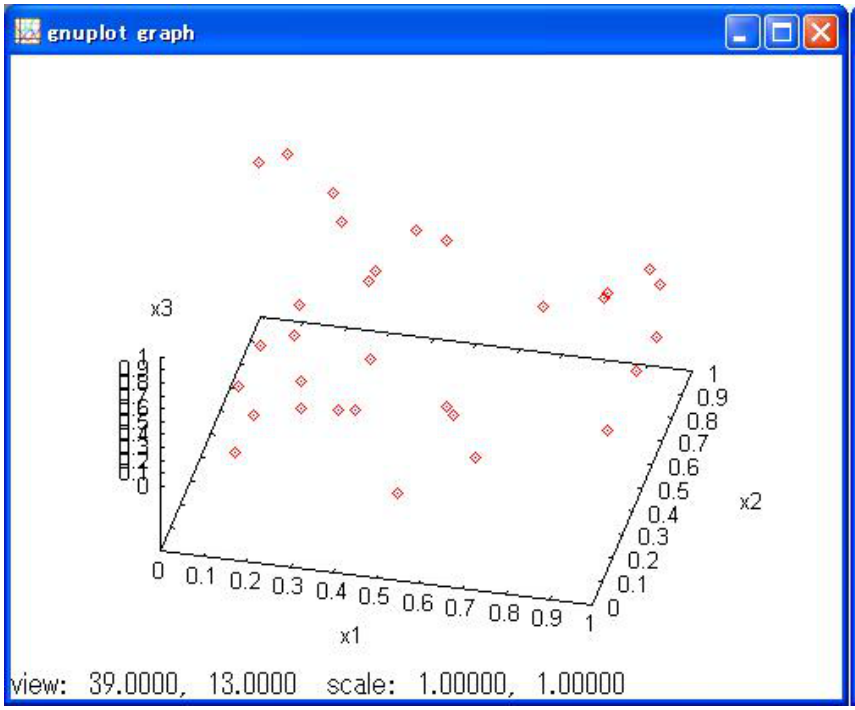
最適問題グラフ図



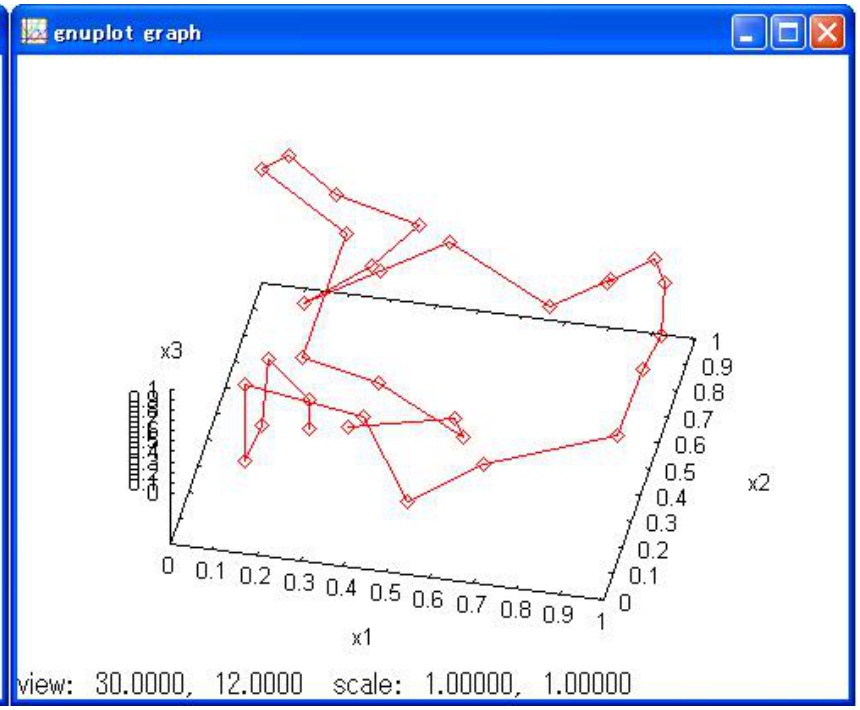
左図の最適問題の定義記述

7.3 PADOC 最適問題 (その3)

4. 最適順回路問題　やきなまし法を採用



3Dのランダムな点を生成

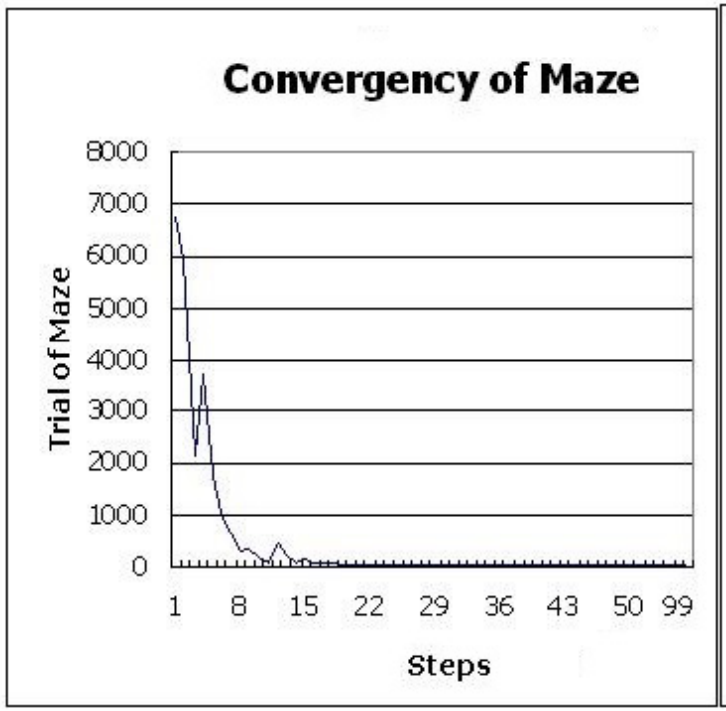


ランダムな点を巡回する最適経路を表示

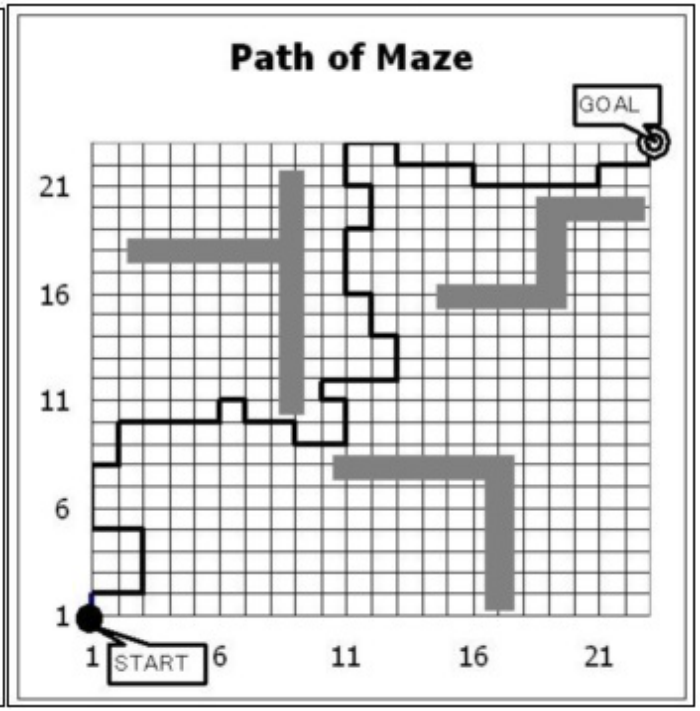
7.4 PADOC 強化学習

1. 強化学習　SARSA法

(例)迷路での最適経路の学習　最初は7000回だが学習後は60回移動でゴールに達する



移動回数の学習状態



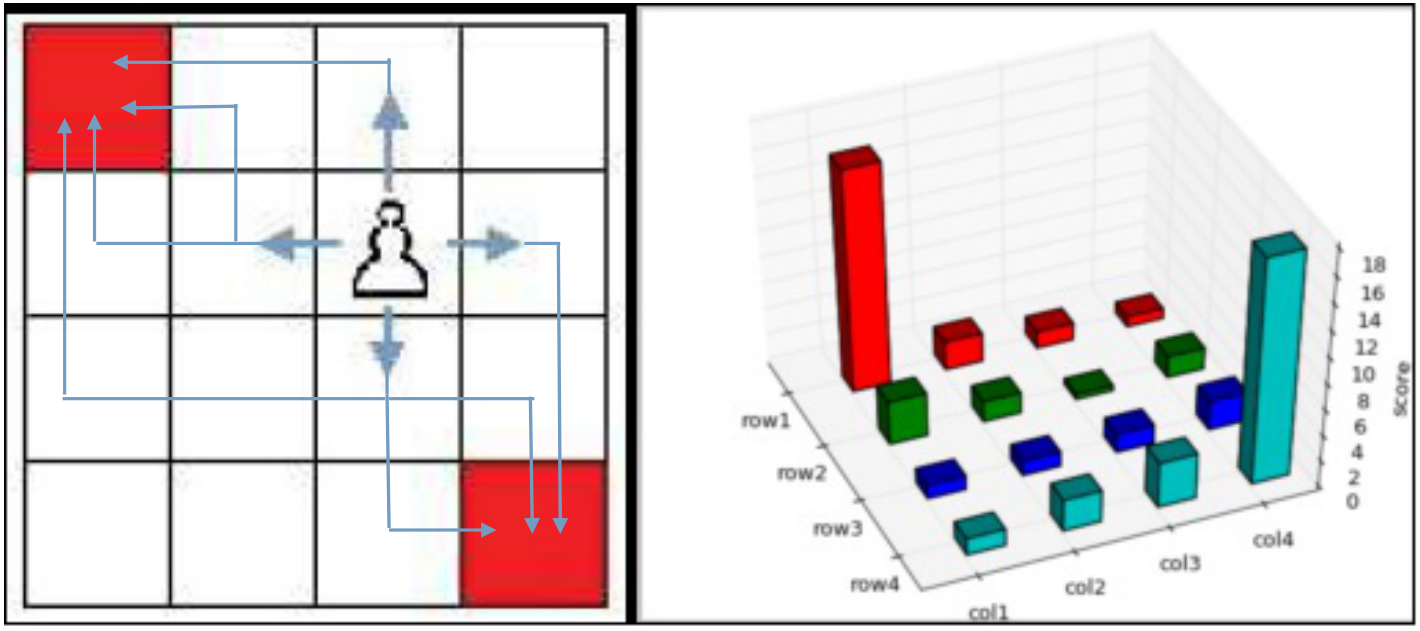
学習後の移動経路

7. 知識発見

7.4 PADOC 強化学習

2. 逆強化学習　行動データより擬似報酬を計算する　最大エントリィ法採用

(例)報酬を得られる様にポーンの経路データより逆強化学習より擬似報酬を計算する



左：ポーンがグリッドを移動して左上隅と右下隅の報酬を得る試行を繰返す
右：ポーンの移動データより逆強化学習で報酬を計算した結果

提案ツールPADOCの機能説明

- 提案ツールの機能説明
  - 実用的なデータ分析として提案ツールの機能を説明
    1. 全体把握（データの前処理）
       - Pythonに比べて簡潔な記述
       - データ結合の処理フロー編集
       - 全体把握ツール
    2. 比較検討
       - 区分による分布比較
       - データ分割による全比較
    3. 仮説検証
       - 教師付モデル
       - 精度と頑健性の問題の検証
    4. 知識発見
       - 隠れ変数推定ベイズモデル
       - グラフィカルモデル
       1. 最適計画問題
       2. 強化学習

8. 考察

- Pythonは難易度が高い
  - オブジェクト指向型の言語
  - データ前処理に使うには難易度が高い
    - データアナリストは前処理はSQL(無償)やSAS(有償)、SPSS(有償)を使う場合が多い
    - ライブラリィが豊富で数理モデル（機械学習や深層学習）を構築するには適している
  - 処理フローを編集してモデルを構築するツールは存在しない
    - 深層学習モデル構築用のTensolbordは存在する
- PADOCは実用的なデータ分析で使い易い
  - データ前処理を簡便な記述で行える
  - グラフィカルな環境が提供されている
  - 比較検討、仮説検証、知識発見で示す様に十分な機能を提供している

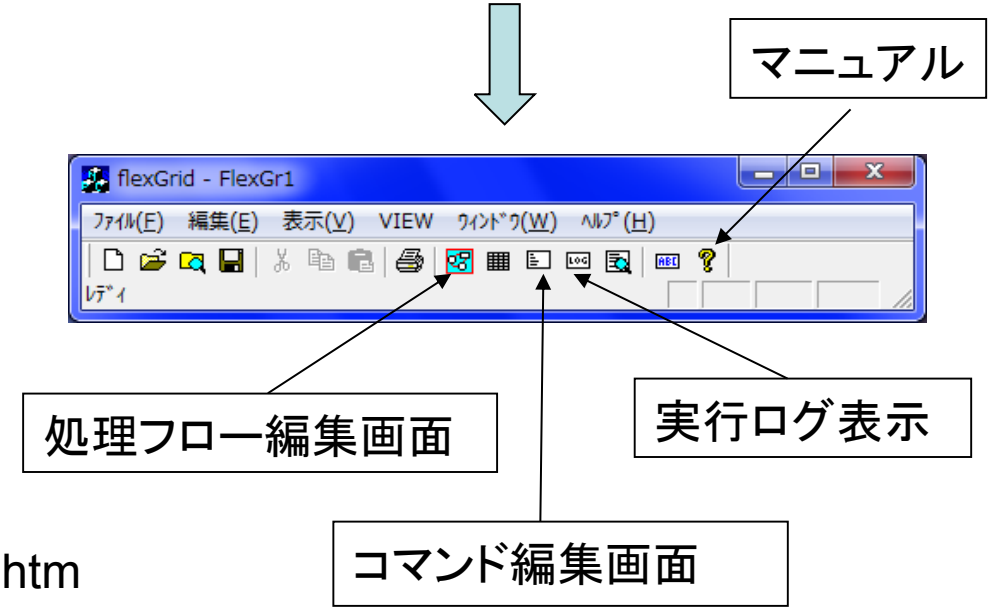
9. まとめと今後

- まとめ
  - 前処理大全 の記述に沿ってデータ編集ツールとして優位性を示した
  - 仮説検定，比較検討，知識発見について，提案ツール PADOC を検討し、十分な機能があることを示した
- 今後
  - Python上では驚異的なモデルが続々と発表されている
    - AlphaGoZeroの結論の記述
      「人類が数千年かけて習熟した囲碁の技を我々は数日で達成した」
  - Python で作られたライブラリィを取り込むAPIを公開して多くの人がこの環境をカスタマイズを可能にする

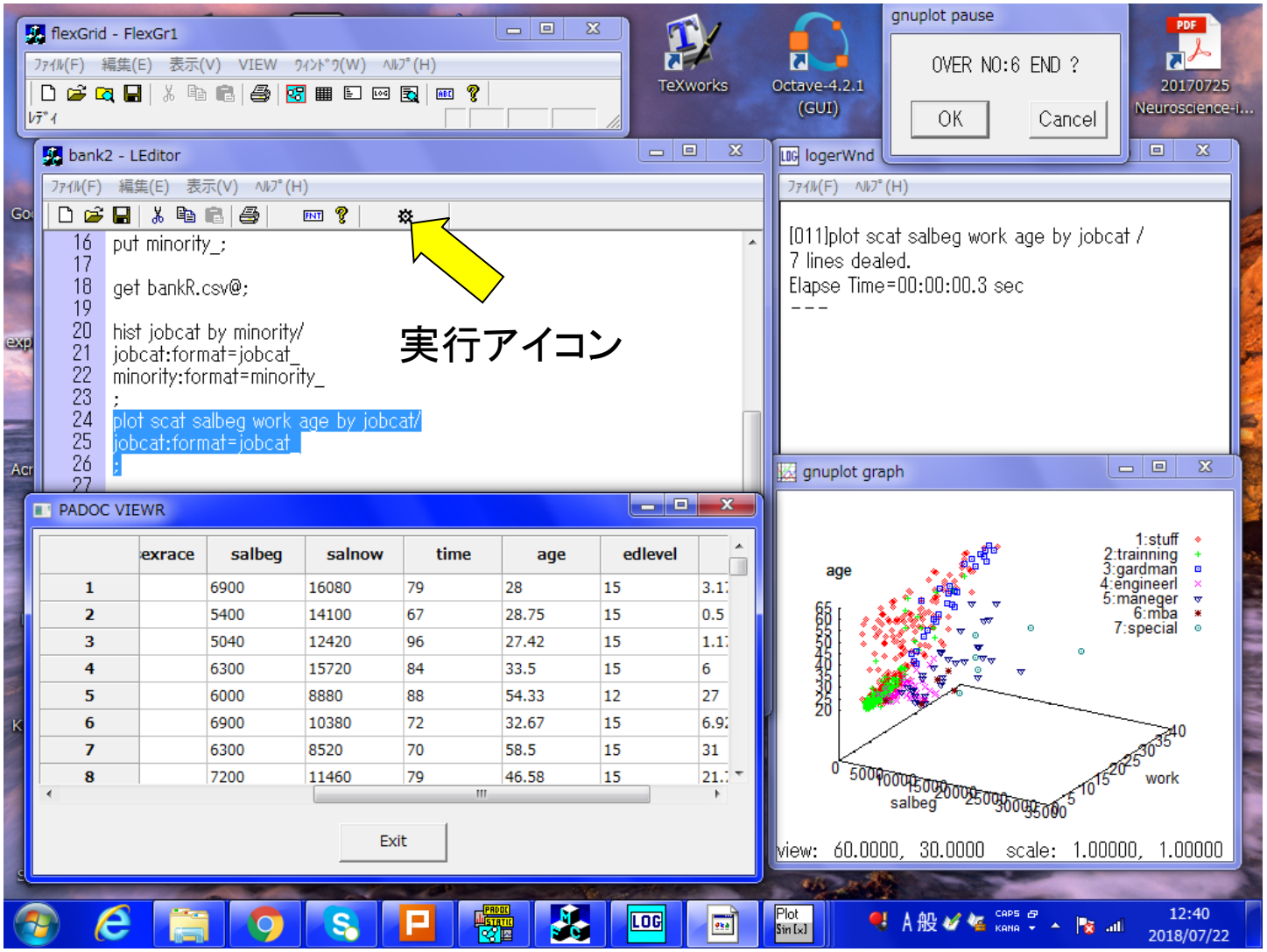
付録1　インストール方法

http://www.geocities.jp/mabonakai/index.htm

実行アイコン



付録2　コマンドモード



	exrace	salbeg	salnow	time	age	edlevel	
1		6900	16080	79	28	15	3.1
2		5400	14100	67	28.75	15	0.5
3		5040	12420	96	27.42	15	1.1
4		6300	15720	84	33.5	15	6
5		6000	8880	88	54.33	12	27
6		6900	10380	72	32.67	15	6.9
7		6300	8520	70	58.5	15	31
8		7200	11460	79	46.58	15	21.1

付録3　データフローモード