

ソフトウェア開発プロジェクト データの統計解析②

——予測・見積もりの技術——

門田 暁人

EASE プロジェクト(<http://empirical.jp>)

奈良先端科学技術大学院大学 情報科学研究科

Copyright © 2007 Nara Institute of Science and Technology

本日のお話

- データ間の関係を調べる.
 - 視覚的に
 - ◆ 散布図, ヒストグラム, 箱ひげ図, 平行座標プロットなど
 - 定量的(統計的)に
 - ◆ t検定, カイ二乗検定, 分散分析, 無相関検定など
 - ◆ 相関係数, 寄与率, クラメールのV, 回帰曲線など
- データの予測(見積もり)を行う.
 - モデルベース手法, メモリベース手法
- 大量のデータの中から隠された関係を発見する.
 - アソシエーション分析(相関ルール分析)

Copyright © 2007 Nara Institute of Science and Technology

2

ソフトウェア開発データリポジトリの例

(注) このデータは架空のものです。

プロジェクトID	開発種別	業種	アーキテクチャ	開発言語(第1言語)	OS
1	a: 新規開発	a: 銀行	a: クライアントサーバ	d: VISUAL BASIC	g: WINDOWS NT
2	a: 新規開発	a: 銀行	b: スタンドアロン	f: PL/I	c: MVS
3	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
4	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
5	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
6	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
7	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
8	a: 新規開発	a: 銀行	c: 混合	d: VISUAL BASIC	c: MVS
...
73	b: 改修・保守	c: 混合	c: 混合	c: COBOL	...

要求仕様_明確度合	開発期間(月数)	ピーク要員数	FP計測手法	規模(FP)	開発工数(人時)
	15	15	a: IFPUG	556	24690
c: ややあいまい	8	6	a: IFPUG	80	825
	6	1	a: IFPUG	77	758
a: 非常に明確	4	6	a: IFPUG	255	2119
	6	0	a: IFPUG	349	2741
d: 非常にあいまい	1	3	a: IFPUG	69	1090
b: かなり明確	4	11	a: IFPUG	375	1855
	6		a: IFPUG	271	1747
b: かなり明確	12	4	a: IFPUG	439	2007

Copyright © 2007 Nara Institute of Science and Technology

3

ソフトウェア開発リポジトリに基づく予測の例

	設計工数 (人月)	基本設計 バグ数	詳細設計 バグ数	試験工数
現行プロジェクトX	50	23	10	予測結果 40.25

詳細設計までの開発が終了した時点で、試験工数を予測したい。

モデル式:

$$\text{試験工数} = \text{設計工数} \times 0.2 + \text{基本設計バグ数} \times 0.1 + \text{詳細設計バグ数} \times 0.05$$

ソフトウェア開発データリポジトリ

↑ 重回帰分析

	設計工数 (人月)	基本設計 バグ数	詳細設計 バグ数	試験工数
過去プロジェクトA	45	20	19	36
過去プロジェクトB	55	33	11	44
過去プロジェクトC	10	14	15	30

Copyright © 2007 Nara Institute of Science and Technology

4

何を予測するのか？

- 開発工数（人月もしくは人時）
 - 総工数, テスト工数, ...
 - ◆ 開発期間, 開発要員数, 生産性
- 品質（バグ数, バグ密度）
 - モジュール単位, 機能単位
- プロジェクトの成否（失敗, 成功）
 - コスト超過, 納期超過

ソフトウェア開発工数見積もり手法

- 積算法
- 計算式等のモデルによる算出（モデルベース）
- 類似プロジェクトからの類推（メモリベース）



ソフトウェア開発データリポジトリを
利用可能

モデルベース手法とメモリベース手法

- モデルベース手法
 - 定義済みモデルを使う
 - ◆COCOMO, COCOMO II, Agile COCOMO
 - 過去のデータからモデルを作る
 - ◆重回帰分析, CoBRA法, ニューラルネット, . . .
- メモリベース手法(類推)
 - ◆Analogy-based法(CBR法)
 - ◆協調フィルタリング法
- ハイブリッドな手法
 - OSR法

モデルベース予測

	設計工数 (人月)	基本設計 バグ数	詳細設計 バグ数	試験工数
現行プロジェクトX	50	23	10	予測結果 40.25

詳細設計までの開発が終了した時点で、試験工数を予測したい。

モデル式:

$$\text{試験工数} = \text{設計工数} \times 0.2 + \text{基本設計バグ数} \times 0.1 + \text{詳細設計バグ数} \times 0.05$$

ソフトウェア開発データリポジトリ

↑ 重回帰分析

	設計工数 (人月)	基本設計 バグ数	詳細設計 バグ数	試験工数
過去プロジェクトA	45	20	19	36
過去プロジェクトB	55	33	11	44
過去プロジェクトC	10	14	15	30

重回帰分析(線形回帰モデル, 重回帰モデル)

■モデル式

$$\hat{Y} = a_1N_1 + a_2N_2 + \dots + a_kN_k + C$$

\hat{Y} : 従属変数(目的変数)の予測値

N_j : 独立変数(説明変数)

a_j : 係数(偏回帰係数)

C : 定数項

実測値 Y と予測値 \hat{Y} の差を残差と呼ぶ。
残差の2乗和が最小となるように a_j と C を定める。

仮定1: 各説明変数は, 互いに独立である。

仮定2: 目的変数は, 正規分布に従う。

仮定3: 各説明変数と目的変数は直線相関関係にある

重回帰モデルは工数予測モデルとして妥当か？

■モデル式

$$\hat{Y} = a_1N_1 + a_2N_2 + \dots + a_kN_k + C$$

仮定1: 各説明変数は, 互いに独立である。

→ 規模(FP), 工期, 開発要員数など, 独立とはいえない。

→ 多重共線性がある。

重回帰モデルの説明変数間に強い関連が存在することにより,
モデル式が構築できなかったり, 予測結果に信頼が置けなくなる現象。

解決策

→ **変数選択**を行う。(もしくは, 変数の合成を行う)

変数選択

- 目的
 - 多重共線性を避ける.
 - 予測に効いてない説明変数を除去する.
- 2つの手法
 - Filter手法: 予測アルゴリズムとは独立に, 変数間の関係のみに着目して選択する変数を決定する.
 - ◆ 例: 互いに相関の高い変数は, 一方を除去する.
 - Wrapper手法: 予測アルゴリズムを実績データのサンプルに繰り返し適用・評価することで, 選択する変数を探索的に決定する.
 - ◆ 全探索, ステップワイズ法, 山登り法
 - 予測アルゴリズムに応じた統計量を利用する. (尤度比検定量, Wald統計量, AIC, など)

重回帰モデルは工数予測モデルとして妥当か？

■モデル式

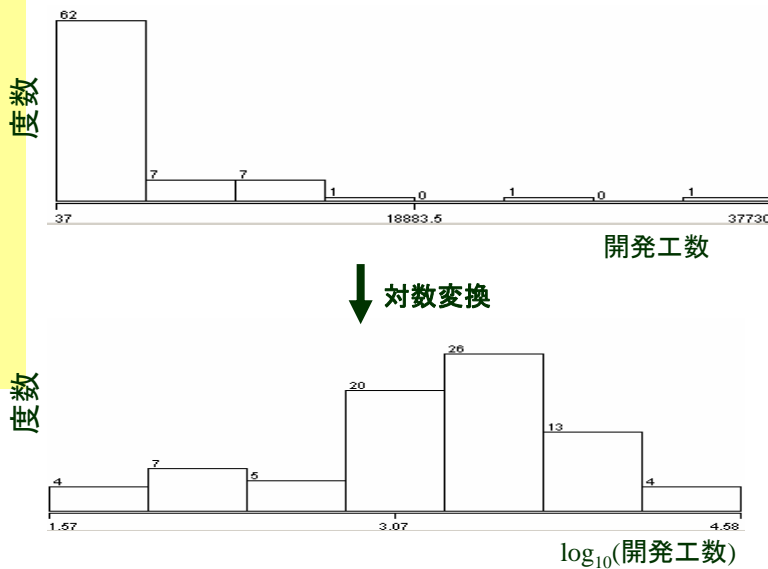
$$\hat{Y} = a_1 N_1 + a_2 N_2 + \cdots + a_k N_k + C$$

仮定2: 目的変数は, 正規分布に従う.
→ 工数は値の小さい部分に偏っている. **正規分布とはいえない.**

解決策

→ **対数変換**を行う.
工数の代わりに $\log_{10}(\text{工数})$ を使う

対数変換の例 - 開発工数(人時)のヒストグラム



Copyright © 2007 Nara Institute of Science and Technology

13

重回帰モデルは工数予測モデルとして妥当か？

■モデル式

$$\hat{Y} = a_1 N_1 + a_2 N_2 + \dots + a_k N_k + C$$

仮定3: 各説明変数と目的変数は直線相関関係にある。

→直線相関関係にあるとはいえない。

一般に、規模が大きくなると工数や要員数は指数的に増大する。

解決策

→指数曲線回帰を使う。

$$Y = C N_1^{a_1} N_2^{a_2} \dots N_k^{a_k}$$

両辺対数取ると

→説明変数, 目的変数共に対数変換してから
重回帰分析する。

$$\log \hat{Y} = b_1 \log N_1 + b_2 \log N_2 + \dots + b_k \log N_k + C_0$$

これを「対数線形モデル」と呼ぶ。

Copyright © 2007 Nara Institute of Science and Technology

14

モデルに対する言い訳

■工数が

$$\hat{Y} = a_1 N_1 + a_2 N_2 + \dots + a_k N_k + C \quad \text{もしくは}$$

$$\log \hat{Y} = b_1 \log N_1 + b_2 \log N_2 + \dots + b_k \log N_k + C_0$$

のような式で表現できるという理論的根拠はない。

■有名な言葉:

- All models are wrong. Some of them are **useful**.
- 予測精度が高ければ役に立つ。

■人間が介在する以上、厳密なモデル化は難しい。
現実的には、「簡潔さ」「計測の容易さ」「高い予測精度」が求められる。

例題1

■ある海外の企業で収集された77プロジェクトの開発データリポジトリ

□目的変数

◆ ActualEffort (開発工数)

□説明変数

◆ Duration, ExpEquip, ExpProjMan, Adj FPs, Dev Env

■として重回帰モデルを作成する。

■ツールの例: JavaScript による重回帰分析

□群馬大学社会情報学部の青木繁伸教授による

<http://aoki2.si.gunma-u.ac.jp/JavaScript/mreg.html>

例題1の結果

変数	偏回帰係数	標準誤差	t値	P値	標準化偏回帰係数
Var01	146.1377	63.42289	2.30418	0.02414	0.2367945
Var02	-269.9088	261.6748	1.03147	0.30582	-0.08561447
Var03	290.0082	226.9117	1.27807	0.20539	0.1052341
Var04	13.06223	2.310043	5.65454	0.00000	0.5812246
Var05	-1894.870	441.4975	4.29192	0.00006	-0.3240460
定数項	2299.255	971.3267	2.36713	0.02066	

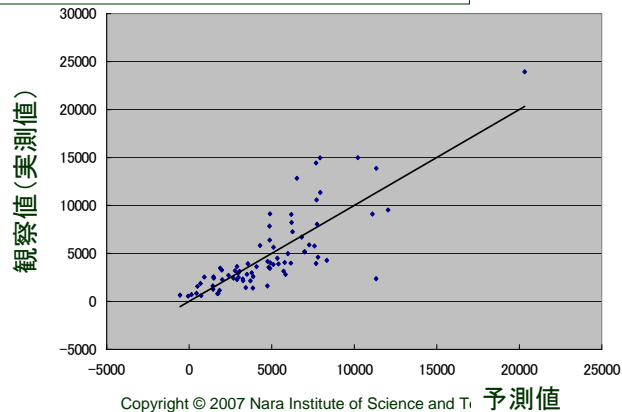
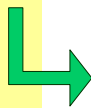
↓ 重回帰モデル式

ActualEffort =

$$146.1377 * \text{Duration} - 269.9088 * \text{ExpEquip} + 290.0082 * \text{ExpProjMan} + 13.06223 * \text{Adj FPs} - 1894.870 * \text{Dev Env} + 2299.255$$

例題1の結果

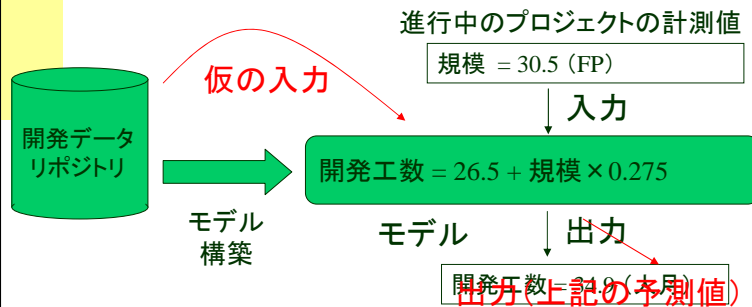
予測結果	番号	観察値	予測値	残差	標準化残差
	1	5152.000000	6992.955093	-1840.955093	-0.751814
	2	5635.000000	5103.539044	531.460956	0.220733
	3	805.000000	1715.085434	-910.085434	-0.375773
				



例題1の結果 注意点

予測結果				
番号	観察値	予測値	残差	標準化残差
1	5152.000000	6992.955093	-1840.955093	-0.751814
2	5635.000000	5103.539044	531.460956	0.220733
3	805.000000	1715.085434	-910.085434	-0.375773
4	3829.000000	5092.929905	-1263.929905	-0.521641
.....				

■この「予測値」は、「予測」により得られた値ではない！



19

例題1の結果 モデルの性能

予測結果				
番号	観察値	予測値	残差	標準化残差
1	5152.000000	6992.955093	-1840.955093	-0.751814
2	5635.000000	5103.539044	531.460956	0.220733
3	805.000000	1715.085434	-910.085434	-0.375773
4	3829.000000	5092.929905	-1263.929905	-0.521641
.....				

■「誤差」ではなく「残差」

$$\text{相対残差} = \frac{|\text{残差}|}{\text{実測値}}$$

$$\text{相対残差の平均} = 0.498$$

(精度はよくない)

20

例題2

- 例題1と同じデータセットを用い、対数線形モデルを作成する。
 - 目的変数、および、説明変数を対数変換してから重回帰モデルを作成する。
 - ただし、ゼロが含まれる変数 (ExpEquip, ExpProjMan) は対数変換できない。これらについては+1してから対数変換する。
 - また、結果の評価に使う「実測値(観測値)」、「予測値」、「残差」は、対数変換前の値を算出する。

例題2の結果

変数	偏回帰係数	標準誤差	t値	P値	標準化偏回帰係数
Var01	0.3849775	0.1163379	3.30913	0.00147	0.2886319
Var02	-0.07782952	0.1455016	0.53490	0.59439	-0.04487862
Var03	0.1502131	0.1414241	1.06215	0.29177	0.0896984
Var04	0.6866678	0.1122953	6.11484	0.00000	0.5186158
Var05	-0.8407526	0.1429008	5.88347	0.00000	-0.4341687
定数項	1.630145	0.2228920	7.31361	0.00000	



重回帰モデル式

$\log(\text{ActualEffort}) =$

$$\begin{aligned}
 & 0.3849775 * \log(\text{Duration}) - 0.07782952 * \log(\text{ExpEquip}) \\
 & + 0.1502131 * \log(\text{ExpProjMan}) + 0.6866678 * \log(\text{Adj FPs}) \\
 & - 0.8407526 * \log(\text{Dev Env}) + 1.630145
 \end{aligned}$$

例題2の結果

■ 対数変換前の値に戻してから評価する.

予測結果

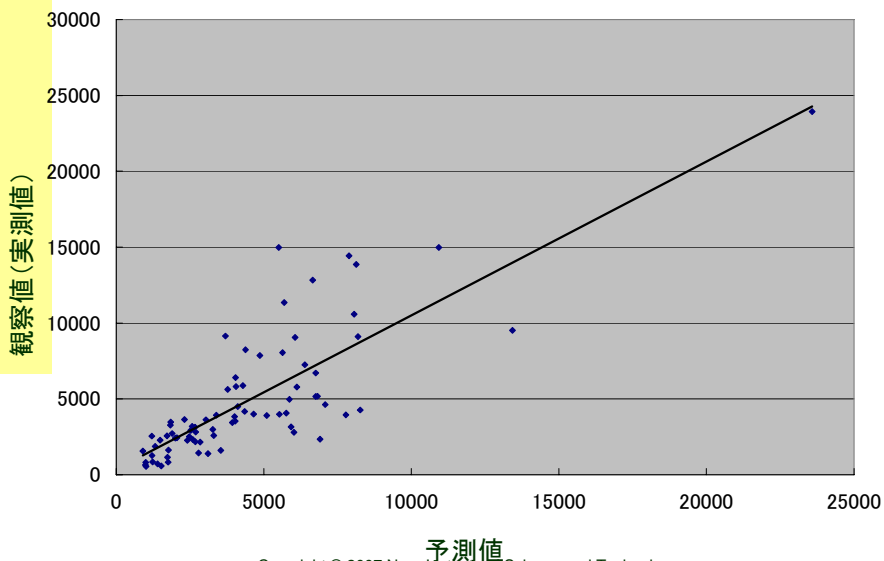
番号	観察値	予測値	残差	標準化残差
1	3.711976	3.830112	-0.118136	-0.596390
2	3.750894	3.577434	0.173460	0.907528
3	2.905796	2.998508	-0.092712	-0.528460
...				

10^{観察値} 10^{予測値} (10^{観察値} - 10^{予測値})



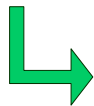
番号	観察値	予測値	残差
1	5152.001728	6762.573526	-1610.571798
2	5635.001033	3779.496957	1855.504076
3	805.0002217	996.5704375	-191.5702157
...			

例題2の結果



例題2の結果

番号	観察値	予測値	残差
1	5152.001728	6762.573526	-1610.571798
2	5635.001033	3779.496957	1855.504076
3	805.0002217	996.5704375	-191.5702157
...			



相対残差の平均=0.409

(例題1と比べて少し精度が向上した)

より現実的な予測

次のようなりポジトリで同様の予測が行えるか？ 説明変数？

プロジェクトID	開発種別	業種	アーキテクチャ	開発言語(第1言語)	OS
1	a: 新規開発	a: 銀行	a: クライアントサーバ	d: VISUAL BASIC	g: WINDOWS NT
2	a: 新規開発	a: 銀行	b: スタンドアロン	f: PL/I	c: MVS
3	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
4	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: MVS
5	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: COBOL
6	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: COBOL
7	a: 新規開発	a: 銀行	b: スタンドアロン	c: COBOL	c: COBOL
8	a: 新規開発	a: 銀行	c: 混合	d: VISUAL BASIC	c: MVS

質的データ (名義尺度)

説明変数? b: 改修・保守

欠損値

要求仕様 明確度合	関係	員数	FP計測手法	規模(FP)	開発工数(人時)
c: ややあいまい	15	15	a: IFPUG	556	24690
a: 非常に明確	8	6	a: IFPUG	80	825
d: 非常にあいまい	6	1	a: IFPUG	77	758
b: かなり明確	4	6	a: IFPUG	255	2119
b: かなり明確	6	0	a: IFPUG	349	2741
b: かなり明確	1	3	a: IFPUG	69	1090
b: かなり明確	4	11	a: IFPUG	375	1855
b: かなり明確	6	1	a: IFPUG	271	1747
b: かなり明確	12	4	a: IFPUG	439	2007

目的変数

より現実的な予測の手順

1. 予測を行う開発工程の決定
 - 例:「詳細設計工程」の完了時
 - 説明変数の候補が決まる.
2. 欠損値のないデータセットの作成
 - いくつかのプロジェクト, 変数の削除
 - 欠損値の補完
3. 尺度の変換
 - 質的データ(カテゴリ変数)→量的データ(数値変数)
4. モデルの構築
5. モデルの評価

予測を行う開発工程の決定

- 工程の例:
 - システム化計画完了時
 - 要件定義完了時
 - 基本設計完了時
 - 詳細設計完了時
 - コーディング完了時

システム化計画完了時

- システム化計画 実績工数(人時)
- 開発期間(月数)計画値
- 開発プロジェクト種別(新規開発, 改修・保守, 再開発, 拡張)
- 開発プロジェクトの形態(商用パッケージ開発, 受託開発, インハウスなど)
- 開発作業の場所(顧客先, 自社)
- ビジネス分野, 業務の種類
- 開発ライフサイクルモデル(ウォーターフォール, 反復型)
- 類似プロジェクトの有無

[参考] 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター, “ソフトウェア開発データ白書2006”, 日経BP社, 東京, 2006.

29

要件定義完了時

- | | |
|-------------------|----------------|
| ■利用形態 | ■業務パッケージ_利用有無 |
| ■利用拠点数 | ■達成目標_優先度_明確度合 |
| ■システム種別 | ■法的規制有無 |
| ■処理形態 | ■要件定義書_文書量 |
| ■要求仕様_明確度合 | ■月数(実績)要件定義 |
| ■ユーザ担当者_要求仕様関与 | ■要件定義_実績工数(人時) |
| ■要求レベル_信頼性 | |
| ■要求レベル_使用性 | |
| ■要求レベル_性能・効率性 | |
| ■要求レベル_保守性 | |
| ■要求レベル_移植性 | |
| ■要求レベル_ランニングコスト要求 | |
| ■要求レベル_セキュリティ | |

[参考] 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター, “ソフトウェア開発データ白書2006”, 日経BP社, 東京, 2006.

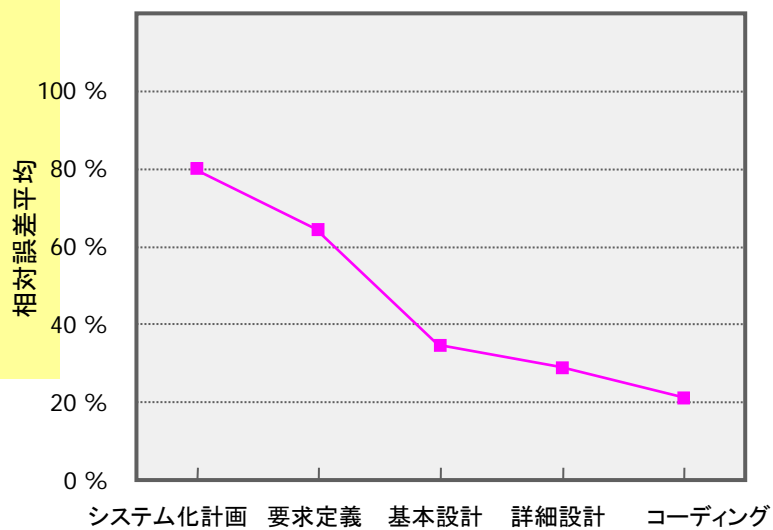
30

基本設計完了時

- 開発言語
- データベース
- 開発支援ツール
 - プロジェクト管理ツール
 - 設計支援ツール
 - ドキュメント作成ツール
 - デバッグ_テストツール
 - 上流CASEツール
 - コードジェネレータ
- ファンクションポイント
- 設計書文書量基本設計書
- 月数(実績)基本設計
- 基本設計書レビュー指摘件数
- 基本設計 実績工数(人時)
- アーキテクチャ
- 開発対象プラットフォーム
- 利用するWeb技術
- 開発方法論

[参考] 独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター, “ソフトウェア開発データ白書2006”, 日経BP社, 東京, 2006.

予測の時期と誤差の関係(例)



欠損値のないデータセットの作成

■ 欠損値処理法

- 平均値挿入法: 欠損値に対して当該変数の平均値を挿入する.
- リストワイズ除去法: 欠損値を一つでも含むケースを削除する.
- 他に, ペアワイズ除去法, ホットデック法などがある.

■ 現実的には,


【手順1】 必要不可欠な変数が欠損しているプロジェクトを削除する.

例: 規模(FP)が欠損しているプロジェクト

【手順2】 欠損率の高い変数, プロジェクトを除去する(値が30%以上欠損している変数やプロジェクトをなくす).

【手順3】 欠損値を補完する.

変数の変換

プロジェクト ID	業種		業種 = 銀行	業種 = 製造業	業種 = 公共
PRO-01	銀行	 ダミー変数化 (2値化)	1	0	0
PRO-02	製造業		0	1	0
PRO-03	銀行		1	0	0
PRO-04	銀行		1	0	0
PRO-05	製造業		0	1	0
PRO-06	銀行		1	0	0
PRO-07	銀行		1	0	0
PRO-08	公共		0	0	1

質的データ
(カテゴリ変数)

ダミー変数
便宜上, 量的データ
とみなす.

変数の変換2

プロジェクト ID	要求仕様 明確度合い
PRO-01	やや明確
PRO-02	非常に明確
PRO-03	やや曖昧
PRO-04	やや曖昧
PRO-05	やや明確
PRO-06	非常に曖昧
PRO-07	やや曖昧
PRO-08	非常に明確



数値化

要求仕様 明確度合い
3
4
2
2
3
1
2
4

順序尺度

便宜上, 量的データ
とみなす.

より現実的な予測の手順

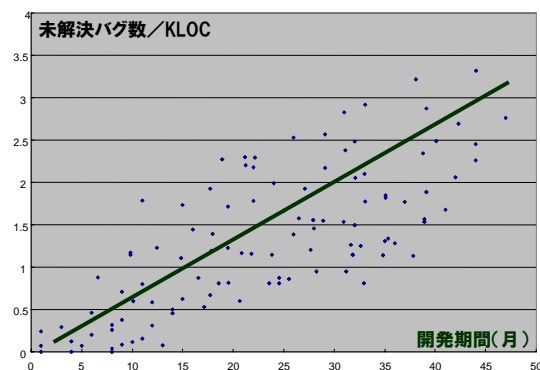
1. 予測を行う開発工程の決定
 - 例:「詳細設計工程」の完了時
 - 説明変数の候補が決まる.
2. 欠損値のないデータセットの作成
 - いくつかのプロジェクト, 変数の削除
 - 欠損値の補完
3. 尺度の変換
 - 質的データ(カテゴリ変数)→量的データ(数値変数)
4. モデルの構築
5. モデルの評価

プロジェクト特性データに基づく見積もり

- 定義済みモデルに基づく見積もり
 - COCOMO, COCOMO II, Agile COCOMO
- 過去のデータに基づく見積もり
 - モデルベース手法
 - ◆ 重回帰分析, CoBRA法, ニューラルネット, . . .
 - メモリベース手法
 - ◆ Analogy-based法, 協調フィルタリング法, OSR法

モデルベース手法の問題点(1)

- 多様なソフトウェア開発プロジェクトを一つのモデルで表現することは難しい。



モデルベース手法の問題点(2)

- データ欠損に対して脆弱である.
 - データ欠損を補う方法は開発されているが、欠損率が30%を超えると、予測精度は著しく低下する.

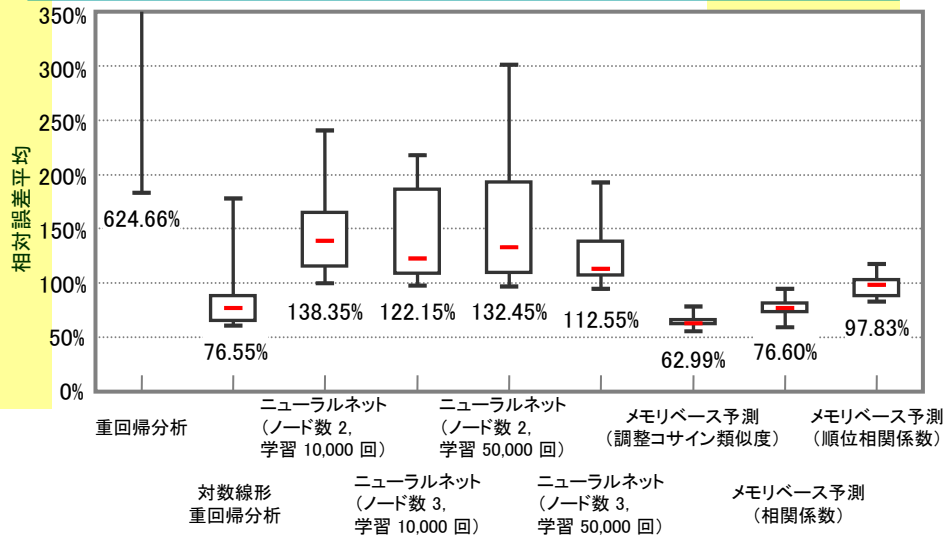
Kromrey, J., and Hines, C.: "Non-randomly missing data in multiple regression: An empirical comparison of common missing-data treatments," *Educational and Psychological Measurement*, 54, 3, pp.573-593 (1994).
- データの欠損は避けられない.
 - 分析目的や組織が異なれば、収集データも異なる.
 - 開発過程のデータは、取り直しがきかない.

メモリベース予測

- ステップ1: 類似度計算
 - 説明変数の値に基づいて、現行プロジェクトXと過去プロジェクトそれぞれ(A, B, C, ...)との間で類似度を計算する.
- ステップ2: 予測値計算
 - 現行プロジェクトXと類似度の高い k個の過去プロジェクトの工数を類似度で加重平均して、現行プロジェクトXの工数の予測値とする.

	設計工数	製造工数	基本設計 欠陥数	予測結果
現行プロジェクトX	50	20	3	40.0
過去プロジェクト 類似度: 0.91	45	18	2	36
過去プロジェクト 類似度: 0.92 (損値)		22	3	44
過去プロジェクト 類似度: 0.69	10	10	(欠損値)	30

メモリベース予測の評価事例



出典: 大杉 他, “企業横断的収集データに基づくソフトウェア開発プロジェクトの工数見積もり,”
SEC journal, No.5, pp.16-25, February 2006.

まとめ

- データの予測(見積もり)を行う手法
 - モデルベース手法
 - ◆ 重回帰分析
 - ◆ 対数線形モデル
 - メモリベース手法