ヴァリデーション

初めに

S-PLUS のアルゴリズムと関数の正確さを関数 validate によりチェックすることが可能です。validate は一般に広まっている典型的なデータ例と非常に極端なデータ例について、いくつかの統計解析に対して検証する（ヴァリデーション：validation）を目的として書かれた関数です。テンプレートファイル形式で行いたいテストを記述し、ユーザー自身の検証用例について、validate を使って確認することも可能です。

この章では、すでに提供されている対応可能な例、validate の文法、出力例を説明しています。この章の最後では、自分自身の例を作る方法に言及しています。

ヴァリデーションのアウトライン

表 1 は組み込みヴァリデーションテストの一覧です。表の左列は解析のグループを、中央列はテストされる S-PLUS 関数を、右列はどのようなテストを行うかを表しています。すでにあるテストを拡張したり、他の S-PLUS 関数についてのテストを作ることも可能です（ユーザー定義関数も含め）。

表 1 組み込みのテスト一覧

解析	関数名	テスト内容
分散分析	aov	・繰り返しがあり、釣り合いの取れた 1 元分散分析 ・繰り返しのない 2 元分散分析 ・繰り返しのある 2 元分散分析 ・完全な釣り合い型ブロック実験、2 次の交互作用項を含む場合と含まない場合 ・完全な釣り合いのとれないブロック実験、交互作用項なし ・2^3 実験 ・split-plot 実験 ・繰り返し計測

	manova	・単純 ・繰り返し計測
記述統計	mean median var standard dev.	何種類かの数値ベクトルに対する記述等計量、大変大きい値や小さい値を含む
	cor	2 つのベクトルの間の相関
検定	binom.test	両側検定 片側検定（上側、下側）
	prop.test	2 群の検定、両側検定
	chisq.test	2x2 分割表（連続補正の有無） 4x5 分割表（連続補正の有無）
	t.test	1 群、両側検定、片側検定（上側、下側） 2 群、両側検定、片側検定（下側） 対のある検定
	wilcox.test	1 群の符号付順位検定、片側検定(上側) 2 群の符号和検定、大きいサンプルを用い、連続相関が仮定できない場合
	fisher.test	2x2 分割表の正確検定、両側検定
	mantelhaen.test	2x2x3 分割表、連続補正あり
	mcnemar.test	2x2 分割表
	kruskal.test	1 変数 3 グループ、両側検定
	friedman.test	2 変数繰り返しなし
	var.test	分散比の検定、両側、片側(下側)検定
	ks.gof	1 群の検定、両側と片側
	chisq.gof	パラメータ μ と σ を指定した連続量の正規分布との比較、期待値が指定された分位から計算される
多変量解析	princomp	相関と共分散に基づく成分の計算
	factanal	バリマックス回転による主成分の抽出、バリマックス回転による最大尤度因子の計算
回帰	lm	単純な線形回帰、仮説検定を含む。 仮説検定を含む多重回帰 多項式回帰 切片のない多重回帰、仮説検定を含む

	glm	ロジスティック回帰、カイ二乗適合度を含む 対数回帰 正規線形モデル
	nls	パラメータ 4 個の Michaelis-Menten モデル
	lme	切片がランダムなモデル 誤差項に AR(1)を持つ、切片がランダムなモデル ランダムな因子 誤差項に AR(1)を持つ、ランダムな因子
統計分布	pnorm, qnorm	-3.0902 から 3.0902 の間の z 値
	pchisq	0.001 から 0.995 の確率 ; 自由度は 1 から 25 までの 5 刻みと、30 から 100 までの 10 刻み
生存時間解析	survfit	2 つのグループのカプランマイヤー推定
	survdiff	Log ランクテスト
	coxph	グループによってモデリングされる単純な生存時間データ 層でモデル化される、死亡が複数表れるデータ 死亡が複数表れるデータに Andersen-Gill の当てはめ
	survreg	正規、ワイブル、指数モデル

テストの実行

関数 validate を用いてのヴァリデーションテストの実行には、単純に次のように呼び出すだけです。

```
> validate()
```

この呼び出しは S-PLUS のホームディレクトリにある、**/splus/lib/validate** にあるすべての可能なテストを実行します。validate はすべてのテストが成功すれば T、それ以外では F を返します。すべてのテストが成功したとすると、出力は次のようなサマリーで終わります。

```
VALIDATION TEST SUMMARY:
…
All tests PASSED
```

組み込みのテストのうち、特定のものを実行するには、引数 file で指定します。選択可能なものは、表 1 組み込みのテスト一覧のいずれかで、順に、anova, descstat, hypotest, multivar, regress, sdistrib, survival です。例えば、次のコマンドは分散分析と、仮説検定を行います。引数 verbose を TRUE にすると、各テストの詳細が validate によって返されます。

```
> validate(file = c("anova", "hypotest"), verbose = T)
--------------- Analysis of Variance ---------------
{
cat("-------------- Analysis of Variance --------------¥n")
T
}
test:
{
#Functions: aov, summary.aov
#Data: Sokal and Rohlf, Box 9.4 p. 220
#Reference: Sokal, R. and F. J. Rohlf. 1981.
#Biometry, 2nd edition.
#W. H. Freeman and Company
#Description: 1-way balanced layout with 5 treatments,
#10 replicates/trtm;
#check df's, sum of squares and mean squares; check F value
#and p-value using a different tolerance
tol1 <- 0.005
tol2 <- 0.04
y <- c(75, 67, 70, 75, 65, 71, 67, 67, 76, 68,
57, 58, 60, 59, 62, 60, 60, 57, 59,
61, 58, 61, 56, 58, 57, 56, 61, 60,
57, 58, 58, 59, 58, 61, 57, 56, 58,
57, 57, 59, 62, 66, 65, 63, 64, 62,
65, 65, 62, 67)
y.treat <- factor(rep(1:5,
c(10, 10, 10, 10, 10)))
y.df <- data.frame(y, y.treat)
y.aov <- aov(y ~ y.treat, data = y.df)
a.tab <- summary(y.aov)
all(c(a.tab$Df == c(4, 45), abs(a.tab$
"Sum of Sq" - c(1077.32, 245.5)) <
tol1, abs(a.tab$"Mean Sq" - c(269.33,
5.46)) < tol1, abs(a.tab$"F Value"[1] -
49.33) < tol2, a.tab$"Pr(F)"[1] <
0.001))
}
. . .
All tests PASSED
VALIDATION TEST SUMMARY:
Test Directory:C:/splus6¥splus¥lib¥validate
File splus6¥splus¥lib¥validate¥anova: All tests PASSED
File splus6¥splus¥lib¥validate¥hypotest: All tests PASSED
```

カスタマイズしたテストを実行するには、まず、テスト内容のファイルを作ります。テストファイル生成については、「自分自身のテストを作る」を参照してください。自分自身のテストファイルを validate で用いるには、引数 file にファイル名を指定し、test.loc にファイルを置いているディレクトリを指定します。例えば、anova1 と hypotest1 というファイルを **c:¥Spluswork¥valdir** に作った場合、validate では次のように呼び出します。

```
> validate(file = c("anova1", "hypotest1"),
+ test.loc = "C:¥¥Spluswork¥¥valdir")
```

もし、テストのいずれかが不成功なら、次のような表示と共に、失敗したテストの詳細が返されます。

```
[1] "***** Test FAILED *****"
1 test(s) FAILED
VALIDATION TEST SUMMARY:
Test Directory:C:¥Spluswork¥valdir
File Spluswork¥valdir¥anova1: All tests PASSED
File Spluswork¥valdir¥hypotest1: 1 test(s) FAILED
```

自分自身のテストを作る

組み込みのヴァリデーションテストはループテストです。ループテストはかっこで括られた一連の S-PLUS コマンドで構成され、各テストは結果に応じて、TRUE か FALSE が返ります。ヴァリデーションテストは 1 つ以上のこのループ式を含みます。例えば、組み込みの記述統計量のヴァリデーションテストにおける最初のループは次のような内容です。このコードは S-PLUS 関数 mean のテストになっています。

```
{
# Function: mean
# Data: test.mat; a test data set suggested by Leland
# Wilkinson in Statistic's Statistics Quiz (1985).
# Reference(s): Sawitzki, G. 1993. Numerical Reliability of
# Data Analysis Systems. submitted for publication in
# Computational Statistics and Data Analysis.
# Description: test mean for numeric data
tol <- 1e-6
test.mat <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0, 0, 0,
0, 0, 0, 0, 0,
99999991, 99999992, 99999993, 99999994, 99999995, 99999996,
99999997, 99999998, 99999999,
0.99999991, 0.99999992, 0.99999993, 0.99999994, 0.99999995,
```

```
0.99999996, 0.99999997, 0.99999998, 0.99999999,
1e+12, 2e+12, 3e+12, 4e+12, 5e+12, 6e+12, 7e+12, 8e+12,
9e+12, 1e-12, 2e-12, 3e-12, 4e-12, 5e-12, 6e-12, 7e-12,
8e-12, 9e-12,
0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5),
ncol=7, dimnames = list(NULL, c(
"X", "Zero", "Big", "Little", "Huge", "Tiny", "Round")))
test.mean <- matrix(0,1,7)
test.mean[1] <- mean(test.mat[,1])
test.mean[2] <- mean(test.mat[,2])
test.mean[3] <- mean(test.mat[,3])
test.mean[4] <- mean(test.mat[,4])
test.mean[5] <- mean(test.mat[,5])
test.mean[6] <- mean(test.mat[,6])
test.mean[7] <- mean(test.mat[,7])
all(c(test.mean[1] == 5,
test.mean[2] == 0,
test.mean[3] == 99999995,
test.mean[4] == 0.99999995,
test.mean[5] == 5.0e+12,
test.mean[6] == 5.0e-12,
test.mean[7] == 4.5))
}
```

カスタムテストファイルは 1 から作ることも、既存のファイルをテンプレートにして作ることもできます。S-PLUS にすでに存在するテストファイルは、**SHOME/splus/lib/validate** にあります。上のコードはディレクトリにある descstat の最初のカッコです。SHOME が環境変数で指定されていなければ、S-PLUS 内部で getenv("SHOME")で返るパスを用いてください。

基本的なテストファイルの構成は{}で括られた一連のコマンドで定義される式を含みます。関数 validate は最初の行がコメント行であることを求めます。もし、1 行目がテストでも、サマリーには記述されません。各式は完結しており、TRUE か FALSE に評価される式になっています。一般的には、ヴァリデーションテストでは、データを作って、1 つ以上の S-PLUS 関数を呼び出します。関数呼び出しの結果はあらかじめ想定された結果と比較され、その差が許容値（tolarance)内かどうかを調べられます。S-PLUS の関数 all や any が比較のために用いられ、結果は TRUE か FALSE になります。