Stephen P. Kaluzny
Silvia C. Vega
Tamre P. Cardoso
Alice A. Shelly

# S+SPATIALSTATS

## ユーザーズマニュアル

株式会社　数理システム

(URL) http://www.msi.co.jp/splus/

F00/02

# 目次

# 本書の使い方

*S+SPATIALSTATS ユーザーズ・マニュアル*は、S-PLUS の空間統計モジュールである S+SPATIALSTATS の使い方と、S+SPATIALSTATS の重要な関数の解説について書かれたものである。

この本では、S+SPATIALSTATS に関する以下の項目について学習する：

- UNIX システムまたは Windows システムへのインストール。
- S-PLUS と S+SPATIALSTATS の関数を使った空間データの探索的データ解析。
- hexagonal binning を用いた空間データの集約。
- 地理統計データに対するバリオグラム推定とクリギング。
- 格子データに対する様々な空間的相関の尺度の計算。
- 自己回帰モデルと移動平均モデルを用いた格子データのモデリング。
- 空間点パターンに対する完全ランダム性の検定。
- 空間点パターンに対する強度や K 関数の推定。
- 空間データのシミュレーション。

## 対象

本書は S+SPATIALSTATS 同様、地理学者、科学者、その他空間データを解析する必要のあるデータ解析者を対象にしている。本書は、*Gentle Introduction to S-PLUS* や *Crash Course in S-PLUS* を読んで得られるような、S-PLUS の操作に関する知識を前提としている。また、ユーザは統計に関する基礎的な知識、特に空間統計に関する知識があることを前提としている。本書は空間統計の教科書となるように書かれているわけではない。空間統計に関して推奨できるいくつかの参考文献については、この序説の後半の「関連書物」を見ていただきたい。

# ユーザーズ・マニュアルの構成

本書は 6 つの章に大別される。第 1 章では、空間データと S+SPATIALSTATSに関する導入が行われる。また、S+SPATIALSTATS が提供する基本的な空間解析ツールの定義と概要の紹介も行われる。第 2 章も導入であるが、ここでは S+SPATIALSTATSの起動と終了、S-PLUS での作業ディレクトリの作り方、外部との空間データのやり取りに焦点を絞る。残りの章では、空間データを大まかに分類した以下の 3 カテゴリに対する空間データ解析について述べる：地理統計データ、格子データ、空間点パターン(定義については 1.1 節と 1.3 節を見よ)。第 3 章では、S-PLUS と S+SPATIALSTATS を使って、探索的データ解析と空間データの視覚化の手法を紹介する。3 タイプの空間データの実例も紹介する。第 4 章から第 6 章では、それぞれ S+SPATIALSTATS関数を使った地理統計データ、格子データ、空間点パターンの解析を行う。また、本書では触れていないが、本書の原著である *S+SPATIALSTATS User's Manual*（英語）の第 7 章は ARC/INFO ユーザ向けであり、ARC/INFO が扱う coverage と、それらに見合う適切な空間データ解析の手法を定義する。いくつかのタイプの coverage に対する空間データ解析の実例も紹介する。

本書の原著である *S+SPATIALSTATS User's Manual*（英語）には 5 つの付録がある：付録 A には S+SPATIALSTATSのインストールに関する解説；付録 B にはカテゴリ別に書かれた S+SPATIALSTATS関数のリスト；付録 C には S+SPATIALSTATS が提供するサンプルデータセットについて書かれたヘルプファイル；付録 D には S+SPATIALSTATS関数に対する個々のヘルプファイル；付録 E には空間統計で良く使われる単語集。

# 表記について

本書では、表記の方法について以下の約束をする：

- ***斜体***（*italic font*）は強調、または UNIX、DOS、S-PLUS のコマンド中でユーザが定義した変数名に用いられる。
- **太字**（**bold font**）は、UNIX や DOS のコマンドやファイル名、章や節の見出しに用いられる。例えば、

**setenv S_PRINT_ORIENTATION portrait**

**SET_SHOME=C:¥SPLUS**

このフォントで「“」と「”」の両者は、キーボードでは 2 重引用符「〃」で表わされるものである。

- **タイプライター体**（typewriter font）は、S-PLUS 関数や S-PLUS セッションの例に用いられる。例えば以下の様なものである：

```
> plot(bramble)
```

S-PLUS コマンドの表示は、S-PLUS プロンプトの「>」に続くものである。コマンドが 1 行以上にわたる場合は、S-PLUS の継続を表わすプロンプト「+」あるいは「Continue string:」に続いて表わされる。

- 四角で囲んだ語句は、キーボード上のキーかマウスボタンのいずれかを表わす。例えば以下のようなものである：

  文字を消去したければ、Delete キーを押しなさい。

**警告：** 左マークの後にある**警告**の語句は、S-PLUS の動作に関する警告を与えるためのものである。これらの警告は注意して読んでほしい。

⇨ **ヒント：** この矢印の後の**ヒント**は、S-PLUS のより高度な使い方について解説しているものである。

**注意：** 警告でもなく、ヒントでもないような興味の対象は、**注意**のあとに続けることにする。

# 関連書物

S-PLUS の操作に詳しいユーザにとって、*S+SPATIALSTATS User's Manual* は S+SPATIALSTATS を様々な作業に使い始めるに当たって必要な、ほとんどすべての情報を含んでいる。S-PLUS に**詳しくない**ユーザは以下のマニュアルのいずれかを読んで、S-PLUS の操作に関する知識を得るのが先決である：

- *A Gentle Introduction to S-PLUS* は、S-PLUS の基本的な操作のほとんどを丁寧に解説している。この本は、S-PLUS を使い始めるに当たって、オペレーティングシステムとデータ解析ソフトウェアの双方に詳しくないユーザを対象に書かれている。

- *A Crash Course in S-PLUS*は、S-PLUS の機能を概括して解説している。他のデータ解析パッケージの操作が出来て、S-PLUS の構文をすばやく理解し、特定の S-PLUS 関数を使えるようになりたいユーザ向けである。
- *S-PLUS User's Manual*は、グラフィック操作やカスタマイズ、データの入出力といった S-PLUS の基本的な操作の手順をきちんと解説している。

その他の情報源としては、*S-PLUS Guide to Statistical and Mathematical Analysis* がある．このマニュアルには、様々な統計的、数学的手法を用いたデータ解析の方法が記述してあり、時系列解析、線形回帰、ANOVA モデル、一般化線形モデル、一般化加法モデル、局所回帰モデル、非線型回帰、回帰樹、階層樹などの古典的な統計モデルについて知識を得ることができる。

空間統計分野一般に関する参考書としては、以下のものをお薦めする：

- Cressie, Noel A.C. (1993). *Statistics for Spatial Data*, Revised Edition. John Wiley and Sons, New York.
- Ripley, B.D. (1981). *Spatial Statistics*. Wiley, New York.

空間統計の特定分野に関する参考書については以下のものをお薦めする：

地理統計データ：

- Isaaks, E.H. and Srivastava, R.M. (1989). *An Introduction to Applied Geostatistics*. Oxford University Press New York.
- Journel A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press, London.

格子データ：

- Cliff, A. D., and J.K. Ord (1981). *Spatial Processes: Models and Applications*, Pion Limited, London.
- Haining, Robert (1983). *Spatial Data Analysis in the Social and Environmental Sciences*, Cambridge University Press, Cambridge.

空間点パターン：

- Diggle, Peter J. (1983). *Statistical Analysis of Spatial Point Patterns*, Academic Press Inc., New York.

これらの他にも空間統計に関する図書、論文は多数ある。巻末の「参考文献」を参照されるとよい。

## S-PLUS テクニカルサポート

S+SPATIALSTATS のサポート契約を購入されている方は、製品のインストールまたは使用に関して問題が生じた場合に、S-PLUS テクニカルサポートを以下のいずれかの方法によって受けることができます：

**日本にお住まいの方：**

- **電子メール：** 以下のアドレスにご質問をお送り下さい：

  **splus-support@msi.co.jp**

- **ファックス：** **03 (3358) 1727** までご質問をお送り下さい。
- **電話：**電話でのサポートは受け付けておりません。

**海外にお住まいの方：** 購入元にお問い合わせ下さい。

## コメント?

私たちは、弊社の資料をより使いやすい、またお客様のニーズにより正確に答えるものにしたいと願っております。この本もしくは S-PLUS 関連の書物に何かコメントがございましたら、以下のアドレスまで電子メールをお送り下さい：

**splus-support@msi.co.jp**

お便りを心からお待ちしております。

# 謝辞

S+SPATIALSTATSモジュールは Stephen Kaluzny 氏と Silvia Vega 氏によって開発された。

本マニュアルは TerraStat Consulting の Tamre Cardoso、Alice Shell 両氏、Stephen Kaluzny 氏と Silvia Vega 氏によって執筆された。Richard Calaway 氏には技術編集をして頂いた。John Minarde 氏には編集とフォーマットを支援して頂いた。

研究と開発には一部 NASA の商用地球観測支援プログラム（EOCAP 93, Earth Observation Commercial Applications Program）契約番号 NAS13-623 に援助して頂いた。空間線形モデル、空間相関、最近接近傍、そしてそれらに関連する S-PLUS 関数のソフトウェアとヘルプファイルは、National Institutes of Health Small Business Innovative Research（NIH SBIR）第 1 期認可番号 1R43CA65340-01 の一部として、Douglas B. Clarkson 氏（主任研究員）と Hubert Jin 氏（研究員）によって開発された。

モジュールの設計と開発は、主に以下の方々に監修して頂いた：

- Brian Ripley 氏（オックスフォード大学）には、このモジュールと解説文書の初期バージョンの設計や全体を通した監修に関して貴重なご意見を頂いた。
- Dan Carr 氏（ジョージ･メイソン大学）には、hexagonal binning に関する関数を提供して頂いた。

Bill Dunlap 氏には、設計の補佐とプログラミングのやっかいな問題に関して貴重な助言をして頂いた。

Miles Logsdon 氏と、NASA 支援によるワシントン大学の学際科学チームである EOS アマゾンモデリングプロジェクトには、月次降雨データを提供して頂いた。Peter Guttorp 氏（ワシントン大学）には地震データセットを提供して頂いた。S+SPATIALSTATS の試作版をテスト使用して下さった方々には、たくさんの有益なコメントを頂いた。

# 1 空間データと S+SPATIALSTATS の紹介

S+SPATIALSTATS は、S-PLUS のアドオンモジュールである。S+SPATIALSTATS は、空間データの統計解析のために設計された、理解しやすく使いやすいツール群を提供する。この章では以下のことを紹介する：

- 空間データの種類(1.1 節)
- 空間データの解析(1.2 節)
- 空間データのための確率モデル(1.3 節)
- S+SPATIALSTATS で利用できるツール群(1.4 節)
- S+SPATIALSTATS の制約(1.5 節)

## 1.1 空間データの種類

空間データはある地域内の特定された位置で観測された測定値により構成される。様々な属性を持った観測対象の値に加えて、空間データセットは位置あるいは相対位置をあわせ持つ。位置は、**点**（*points*）によって与えられることもあれば**領域**（*areal*）によって与えられることもある。特定の固定された位置において観測された値は前者の例である。これらの観測値は、その緯度と経度を特定することで参照することができる。後者は、領域を特定することでその観測値を参照することのできるデータである。センサス区域ごとの夜盗の発生件数などがその例である。ここでセンサス区域のそれぞれは領域である。どちらの場合においても、位置は規則的な場合と、不規則な場合がある。位置が点で与えられる場合、それらの点は

グリッド上規則的に並ぶこともあるし、点どうしの距離もまちまちな、不規則に配置されることもある。位置が領域で与えられる場合、それらは農業用データのように等面積のブロックが連なっていることもあれば、ある郡内の市町村の境界線のように面積も形状も異なることもある。空間データは、採集された岩石中の鉱物の含有量のような連続量であったり、郡によって公表された麻疹（はしか）患者の数のような離散量であったりする。さらに、位置が、採石場の位置をあらわす点のように空間的に連続な場合と、州内の郡のように離散的な場合がある。

S+SPATIALSTATS は空間データの３つのタイプを解析するためのツールを提供する：地理統計データ、格子データ、そして空間点パターンである。

### 1.1.1　地理統計データ

地理統計データ（確率場データとも呼ばれる）は固定された位置で観測された値である。位置の空間は一般に連続である。連続な地理統計データの例には、鉱山内のテストサイトで測定された鉱物の含有率、測候所で記録された降雨量、観測地点における汚染物質の濃度、ある河川の流域のサンプル地点における土壌の浸透率などがある。離散的な地理統計データの例には、ある沖合の固定したサンプルサイト群におけるホタテ貝の収穫数のような計数データがある。

地理統計データは局所変動性を持つことが多く、空間的相関としてモデリングされる。空間的変動はサンプルサイトどうしの距離の関数としてモデリングされる。空間上でサイトどうしが接近していれば一般に値どうしも似てくる、という関係である。地理統計データの探索や分析に使われる方法とツールの記述については第 3 章と第 4 章を見ていただきたい。

### 1.1.2　格子データ

格子データは領域に関連付けられた観測値である。領域は規則的あるいは不規則に配置されている。領域は面積を持つものであれば何でもよく、グリッド上に制限されるものではない。一般に、それぞれの領域に与えられた近傍情報が利用できる。規則的な格子データの例は、人工衛星からのリモートセンシングによって得られた情報である。地表は小さな矩形（ピクセル）に分割され、データは $\mathbb{R}^2$ 上の規則的な格子として受信される。不規則な格子データの例は、一つの州内の郡それぞれに対応するガンの発生

率である。

数学的には、格子は頂点と辺の集合として定義される。サイトが頂点になり、近傍のサイトどうしは辺で連結される。格子データは領域に対して定義されるものであるから、サイトを参照する方法を決定する必要がある（サイトは、その領域の中心によって参照されることが多い）。図 1.1 はスコットランドのグラスゴーにおける地域医療エリア（CMA, community medicine areas）の格子（Haining (1990)表 7.10 より）を図示した例である。

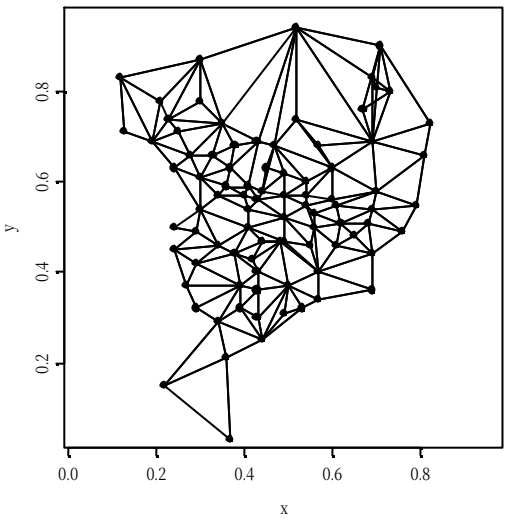


**図 1.1．グラスゴーの格子。**

格子は、各サイトのインデックスに、近傍のサイトのインデックスを付加して形成される。グラスゴーのデータでは、サイトの集合は CMA で構成されている（1, 2, ..., 87）。各サイトは CMA センターの位置で代表され、図 1.1 では頂点で表わされている。近傍集合は距離、共有する境界線、その他データの相関構造に依存する性質によって定義される。図 1.1 の表現において、辺で連結されたサイトは近傍であることを表わしている。近傍関係には重みをつけることもできる（共有する境界線の長さや、サイト間の距離を基にした重みが例として挙げられる）。格子データの探索と解析に使われる方法とツールの記述については第 3 章と第 5 章も見ていただきたい。

### 1.1.3 空間点パターン

空間点パターンは、位置自体が興味ある変量であるようなデータである。空間点パターンは、領域内の有限個の点で構成されている。空間的なランダム性、クラスタ化、規則性、などが最初に行われる解析である。森林地帯におけるある種の木の位置のデータ、震源地の位置などが空間点パターンの例である。空間点パターンの例を図 1.2 に示す。

**マーク付き**（$marked$）空間点パターンには、それぞれの位置に関連する量が付加されている。付加的な量はマーク変量と呼ばれ、点パターンの解析をさらに洗練するために用いられる。ランシングウッズのデータはマーク付き点パターンである；位置に加え、木の種類が記録されているからである。

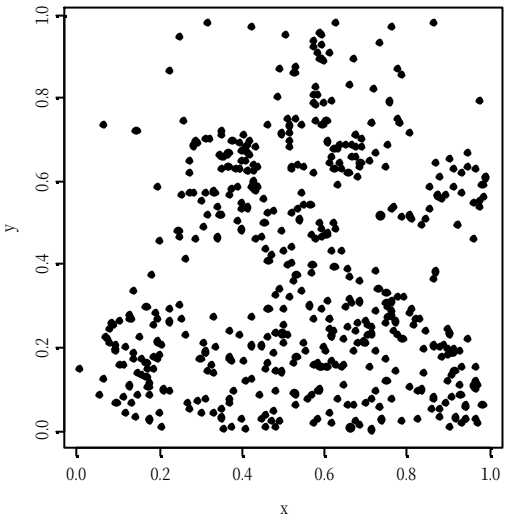


**図 1.2.** ミシガン州ランシングウッズのある森林地帯におけるカエデの木の位置。

## 1.2 空間データの解析

空間データの解析は、そのモデリングと予測に空間情報を盛り込む点で典型的なデータ解析と異なる。データは空間的な相関を持つことが多い；空間的な構造は測定誤差、空間的な異種混合を含む連続効果、空間に依存するプロセスやメカニズムといった様々な原因から生じる（Haining, 1990）。それらの相関または共分散構造はモデリングの精度を挙げたり予測の効

果を挙げたりする際に評価され、用いられる。

### 1.2.1 データのモデル

空間データは以下のように２つの大きな変動成分に分解されることが多い：

$$\text{データ} = \text{大域変動} + \text{局所変動}$$

**大域変動**は個々の点や領域を越えた性質により構成され、全域的なトレンドや勾配を表わす。また、依存関係の局所パターンとしてモデリングされる場合もある。**局所変動**は誤差項と考えることもでき、測定誤差、領域内の変動、サイトに固有な変動によるものとして特徴づけられる（Haining, 1990）。空間的モデリングは、どちらの成分に対しても行うことができる。

### 1.2.2 データ収集と精度

大抵のデータ同様、空間データにも測定誤差は存在し、精度には限界がある。しかし、空間データに特有な誤差として、位置の記録や領域の境界の表現に伴い生じるポテンシャルエラーがある。特に、領域には様々な大きさや形がある；データベース上で境界を表現するのに用いられる方法や解像度が、そのポテンシャルエラーを生じさせる重要な原因となっている。

空間依存モデルは領域依存（area dependent）である。 それらは、局所的な外れ値の存在によることはもちろんのこと、サンプリングに偏りがあるといったようなことも原因となる。空間依存性のほとんどは 2 次元的なものであるから、空間データの境界条件（エッジ効果）はより複雑なものになる。空間データセットの境界は領域の周囲に広がっており、時系列解析のような 1 次元的な解析よりもはるかに多数のサイトが影響を及ぼしているのである。

### 1.2.3 定常性

多くの場合、空間データセットはある確率過程の 1 つの実現値として表現される。そのような場合、データに対する推論を行うためにある程度の**定常性**（*stationarity*）が仮定される必要が生じる。定常性とは、データのある程度の**位置の普遍性**（*location invariance*）のことである。位置の普遍性とは、空間のどんな部分集合内の点どうしの関係も、それらが空間の

どこにあるかに関わらず一定であることを示す。定常性は平均と分散の両方に適応される。よく用いられる定常性は以下の 3 種類である：

1. 強定常 — 分布関数が平行移動と回転に関して等しくなければならない。
2. 弱定常 — 一定平均と位置に無関係な分散を持たなければならない。共分散は 2 点間の距離（と場合によっては方向）のみに依存する。正の有限分散を仮定する。
3. 増分定常 — **増分**（*increments*）の分散が位置に無関係である必要がある。増分は 2 点間の 1 回差分で定義する。

過程の定常性は**局所的**（*local*）にも**大域的**（*global*）にも定義できる。過程が大域的には非定常であっても、局所的な部分集合内では定常性を示すことも有り得る。大域的に定常であるかを確かめるのは非常に難しいが、局所定常性を確かめるのは比較的容易である。そして、データ解析には局所定常性が示されれば十分である。

### 1.2.4 等方性

空間データのほとんどは 2 次元なので、どんな空間過程にも**等方性**（*isotropy*）は重要である。この場合の等方性とは、「空間過程がどの方向にも同じように展開している」、ということである。相関係数や共分散が方向によって異なるような空間過程は**異方的**（*anisotropic*）であるという。空間データの解析手法の多くは、空間的な相関が等方的であることを仮定している。

### 1.2.5 縮尺

空間データの解析において、縮尺あるいは空間の解像度は重要である。空間データの大域的なパターンは、縮尺が異なると異なった過程から生じたものになるかもしれないからである。探索的データ解析は様々な縮尺におけるパターンを検出する手助けになる。しかし、データから求められたパターンはデータが採集された時点の解像度と相関を持つ。

## 1.3 空間データの確率モデル

数学的には、空間過程の多くが 1 つのシンプルな確率モデルによって記述される（Cressie, 1993）。$\mathbf{s} \in \mathbb{R}^d$ を、$d$ 次元ユークリッド空間内の一般的な位置を表わすものとしよう。空間上の位置 $\mathbf{s}$ における潜在的なデータ量 $\mathbf{Z}(\mathbf{s})$ を確率的な量として定義する。

$\mathbf{s}$ を $D \subset \mathbb{R}^d$ 内で変化させることで、多変量確率場（RF）

$$\{\mathbf{Z}(\mathbf{s}) : \mathbf{s} \in D\},$$

が生成される。実際には、与えられる空間データは、元になる確率過程の唯一の実現値であることが多い。$Z(\boldsymbol{s})$ の実現値を $\{z(\boldsymbol{s}): \boldsymbol{s} \in D\}$ で表わす。

すでに紹介した特徴的な 3 つのタイプの空間データは、以下に示すように一般的な確率モデルの特別なケースとして考えられる：

- 地理統計データ — $D$ は $\mathbb{R}^d$ の固定した部分集合で、正の体積を持った $d$ 次元立方体を含んでいる。$Z(\boldsymbol{s})$ は位置 $\boldsymbol{s} \in D$ におけるランダムベクトルで、$\boldsymbol{s}$ は $D$ の部分集合内を連続的に変化することができる。これは地理統計データを格子データや点パターンデータと区別する一般的な性質である。

- 格子データ — $D$ は $\mathbb{R}^d$ 内の固定した、可算個のサイトの集合である。格子はサイトのインデックスによって定義され、近傍情報が付置されている。

- 空間点パターン — $D$ は $\mathbb{R}^d$ もしくは $\mathbb{R}^d$ の部分集合の点パターンである；$D$ のインデックス集合がランダムな事象の集まり、すなわち空間点パターンなのである。一般的な空間点パターンでは、$Z$ は特定されないか、すべての $\boldsymbol{s} \in D$ においてスカラー量 $Z(\boldsymbol{s}) \equiv 1$ と考えられる。マーク付き点パターンでは、$Z(\boldsymbol{s})$ は位置 $\boldsymbol{s} \in D$ におけるランダムベクトルである。

## 1.4 S+SPATIALSTATS の空間解析ツール

S+SPATIALSTATS は様々なタイプの空間データの空間解析を行うための関数を提供する。加えて、S-PLUS にも空間データの解析、特に探索的データ解析に有効なツールは多く存在する。探索的データ解析における個々の S-PLUS 関数の応用のしかたについては、第 3 章を見ていただきたい。

本書においては、S+SPATIALSTATS の関数のほとんどを 1.1 節で定義した空間データの 3 つのカテゴリに分類する。これらのカテゴリは排他的なものではなく、多くの関数は異なるタイプのデータに対しても用いることができる。例えば、空間回帰モデルは典型的には格子データに適応されるものであるが、地理統計データにも用いることができる。同様に、バリオグラム推定のような、地理統計データに対して行うようなタイプの解析を格子データにも行うことが可能である。

地理統計データの解析には、S+SPATIALSTATS は以下のことを行うツールを用意している：

- 標準的または頑健な、全方向のあるいは一方向のバリオグラムの推定と表示。
- 経験バリオグラムのモデリング；経験データに理論バリオグラムモデルを当てはめる。
- 通常クリギングを行い、観測値の得られていない位置での値と、そのクリギング予測分散を点推定する。
- 普遍クリギングを行い、予測と大域的なトレンドをモデリングする。

格子データに対しては、S＋SPATIALSTATS は以下のことを行う関数を供給する：

- 距離や共有する境界線をもとに、最近接近傍や近傍集合を求める。
- Moran や Geary の相関係数を用いた空間的自己相関の算出と検定。
- 条件付き自己回帰、同時自己回帰、移動平均共分散構造モデルを使った空間回帰。

点パターンデータの解析には、S+SPATIALSTATS は以下のことを行うような関数を供給する：

- 空間的な歪みを取り除いた正確な位置のプロット。
- 最近接近傍法を用いた完全ランダム性の探索。
- カーネル、局所回帰、ガウス法を用いた強度の推定。
- Riplay のK関数を用いた２次定常性の検定。

S+SPATIALSTATS には 3 つの基本タイプの空間データをシミュレーションする機能もある。

## 1.5 制限

S+SPATIALSTATS Version 1.0 は2 次元の空間データを解析するように設計された。このバージョンには 3 次元空間データを扱う手法は含まれていない。しかし、多くの手法は拡張可能である。また、S+SPATIALSTATS には空間時系列データを解析する手法は含まれていない。

S+SPATIALSTATS は画像解析用パッケージではない。画像解析は通常、データタイプが特殊であり（例えば 1 バイトデータ）、大きなサイズのデータを扱う必要がある。S-PLUS と S+SPATIALSTATS は一般に、この方面の応用を考えて作られてはいない。

最後に、S-PLUS のシステムは、インタプリタ型プログラミング言語である。その柔軟性はコンピュータのメモリの大量消費につながる。典型的なデータ解析では、ユーザが所有する最も大きなデータオブジェクトの 6、7 倍のメモリが要求される。このことによって解析可能なデータのサイズに上限が生じる。

# 2 S+SPATIALSTATS の手始め

以前に S-PLUS を使ったことがある人ならば、S+SPATIALSTATS の起動は易しい。単純に S-PLUS を起動し, S+SPATIALSTATS モジュールを付加し、S+SPATIALSTATS の関数（空間データを表示し、解析する S-PLUS 関数）の使用法の学習に着手すればよいのである。この章では、S+SPATIALSTATS を動かすのにために必要な、以下の作業について説明する。

- S+SPATIALSTATS の起動と終了。S-PLUS の起動と同時に S+SPATIALSTATS を開始するための環境設定。
- Windows 環境で S+SPATIALSTATS のヘルプを得る方法（UNIX 環境では、S+SPATIALSTATS 関数のヘルプは S-PLUS 関数のヘルプと全く同じ方法で見ることができる）。
- Trellis グラフィックスを S+SPATIALSTATS 内で用いて、空間データを視覚化する方法。
- 外部の空間データを S-PLUS に取り込んで S+SPATIALSTATS で使用する方法。
- S+SPATIALSTATS に含まれるサンプルデータ集を使用する方法。

本書は、S-PLUS の操作に関する知識を前提にしている。

この章に登場する手続きの中には、S+SPATIALSTATS を Windows で利用しているか UNIX で利用しているかによって異なるものもいくつかある。オペレーティングシステムの違いによってコマンドや手続きが異なる場合は、この章では両方を記述する。

# 2.1 S+SPATIALSTATS の起動と終了

S-PLUS を起動し、S-PLUS のセッションに S+SPATIALSTATS 関数を付加することで S+SPATIALSTATS が起動する。UNIX システムで S-PLUS を起動するには、シェルプロンプトで Splus と打てばよい。Windows システムで S-PLUS を起動するには、スタートメニューから「S-PLUS for Windows」を選択する。

S-PLUS セッションに S+SPATIALSTATS 関数を付加するには、以下のコマンドを使う：

```
> module(spatial)
```

**警告：**S+SPATIALSTATS モジュールは、S-PLUS 関数のいくつかをマスクしてしまう。マスクされた関数は、S+SPATIALSTATS 関数の使用に合わせて修正されてはいるものの、S-PLUS の通常使用には影響を及ぼさない。

S-PLUS セッションで S+SPATIALSTATS モジュールを取り外す場合は、以下のコマンドを用いる：

```
> module(spatial, unload=T)
```

S+SPATIALSTATS を常に使いたいユーザは、起動時に自動的に S+SPATIALSTATS モジュールを付加するようにカスタマイズしたいと思うかもしれない。このことは、関数.First に module(spatial)の 1 行を追加するだけで簡単に行なえる。関数.First をまだ作成していない場合は、以下のようにして作成するとよい:

```
> .First <- function(){module(spatial)}
```

S+SPATIALSTATS で解析したデータを見失わないようにするには、プロジェクトごとにディレクトリを作るとよい。プロジェクトごとのディレクトリそれぞれには、S-PLUS 用のサブディレクトリ、**.Data**（Windows では**_data**）を作っておく必要がある。起動時に S+SPATIALSTATS を付加するようにしたい場合には、**.Data**（Windows では**_data**）それぞれに上で定義した.First を定義しておく。これで、あるプロジェクトの作業を行ないたい場合は、そのプロジェクト用に作ったディレクトリで S-PLUS を起動するだけでよくなる。データディレクトリの編成とその使用法はオペレーティングシステムが UNIX であるか Windows である

かによって異なる。以下の節では、それぞれのシステムで行なう作業について説明する。

### 2.1.1 UNIXでの作業データの編成

UNIX 上で S+SPATIALSTATS プロジェクトのためにディレクトリ *dir* を作成し、使用するには、以下のコマンドを用いる（%はシェルプロンプトを表わす。コマンドではないのでタイプしなくてよい）：

% **mkdir** *dir dir*/**.Data**
% **cd** *dir*
% **Splus**

このプロジェクト用のディレクトリで最初に作業するときのみ、module 関数を直接使用して S+SPATIALSTATS 関数を付加し、次からの作業では自動的に S+SPATIALSTATS モジュールが付加されるように関数.First を定義する：

```
> module(spatial)
> .First <- function(){module(spatial)}
```

## 2.2 Windows 上で S+SPATIALSTATS 関数のヘルプを得る

S-PLUS は内蔵するほとんどすべての関数に対してオンラインヘルプを提供している（内部呼び出しのみの関数の中にはヘルプファイルがないものも存在する）。S-PLUSでは、またUNIX上の S+SPATIALSTATS では、関数 *fun* のヘルプは関数 help を使って以下のようにして参照できる。

```
> help(fun)
```

しかし、S-PLUS for Windows で *fun* が S+SPATIALSTATS 関数の場合に同様のことを行なうと、S-PLUS for Windows は Windows ヘルプの検索ダイアログを表示する。これは、S+SPATIALSTATS 関数 *fun* が S-PLUS for Windows 標準のヘルプファイルに含まれていないからである。S-PLUS for Windows で S+SPATIALSTATS関数のヘルプを得るためには、関数のヘルプを要求する際に、その関数が所属するモジュールを特定しなければならない。したがって、関数 variogram のヘルプを得たい場合は、help を次のように用いる：

```
> help(variogram, module="spatial")
```

S+SPATIALSTATS ヘルプファイルのインデックスを表示させる場合には、関数名を省略する：

```
> help(module="spatial")
```

現われた目次の中からヘルプの必要な項目を選択すればよい。

## 2.3 グラフィックデバイス

S-PLUS でグラフィックスを使うには、一つ以上のグラフィックスデバイスドライバを開始しなければならない。グラフィックデバイスには画面上のウィンドウ、物理的なプリンタやプロッタ、物理的なグラフィックデバイスを制御するソフトウェアなどがある。このマニュアルでは、描画コマンドを実行する際にはいつでもグラフィックスデバイスは開かれており、アクティブであることを仮定している。ほとんどの場合、アプリケーションに応じて `motif` か `trellis.device` のいずれかを使用する。Windows を使用する場合、`win.graph` か `trellis.device` を使う必要がある（S-PLUS for Windows 4.0 以上の場合は特別な場合を除いて必要ない）。グラフィック用ウィンドウは `q()` を使って S-PLUS を終了する際、自動的に閉じられる。セッション中にグラフィックデバイスを閉じる場合、`graphics.off` か `dev.off` を使う。S-PLUS のグラフィックデバイスに関するより詳しい情報は S-PLUS ユーザーズガイドを見よ。

## 2.4 S+SPATIALSTATS で Trellis グラフィックスを使う

S+SPATIALSTATS ユーザーズマニュアルのいたるところに現われる多くの例は、空間データを表示するのに、標準 S-PLUS グラフィックスと併用して Trellis グラフィックスを用いて描いてある。加えて、S+SPATIALSTATS 関数の中には結果を表示するために、Trellis 関数を直接呼び出すものもある。Trellis によるグラフは、1 枚または数枚のグラフが行、列、ページにわたって規則的な格子のように配置される。それぞれのグラフはデータの部分集合を表わしており、それらの部分集合は**与えられた**（*given*）変数の値によって決定される。

Trellisグラフィックスは2次元表示や3次元表示の様々なバリエーションを含む十分に一般的なものである：ヒストグラム、散布図、点プロット、等高線プロット、3次元プロット、3次元点プロット等。しかし、Trellisグラフィックスに現われるすべてのグラフは類似していなければならない。データの部分集合は規則的に選ばれ、データの連続的なまたは離散的な変数に関して条件付けされ、高次元データのグラフの配列となって現われる。Trellisソフトウェアは軸や縦横比を柔軟に制御できるようになっており、またデータによって縦横比を変えるための"banking"の計算も行なう。

Trellisライブラリは S+SPATIALSTATSモジュールが付加されると同時に自動的に付加され、Trellis関数のすべてが使用可能になる。Trellisグラフィックスの使用法に関する詳細は *S-PLUS Trellis Displays User's Manual* や *S-PLUS Programmer's Guide* を参照。

## 2.5 空間データのインポートとエクスポート

本書中の例に使われる空間データは、S-PLUS のデータフレームとして含まれているか、もしくはS+SPATIALSTATS関数を用いて生成されたものである。S+SPATIALSTATSモジュールを実際に用いる際、自分のデータを S-PLUSの中に読み込む必要がある。フィールド数が固定されたテキストファイルとして保存されているデータを S-PLUS 関数 scan や `read.table` を使って読み込むだけでなく、S+SPATIALSTATSは特殊なタイプの空間データを読み込む関数を持っている。この節では、空間近傍情報を持った、フィールド数が固定されないようなテキストデータファイルを読み込む方法と、GEO-EASや ARC/INFOとのデータのやり取りの方法について解説する。

### 2.5.1 空間近傍情報を含むテキストファイルの読み込み

格子データの解析は、格子の要素それぞれに定義された空間近傍の利用を伴う。一般に、近傍の数は領域によって異なる。よって、近傍を定義したファイルはフィールドの数が変化する。S+SPATIALSTATS の関数 `read.neighbor` を、このようなデータを読み込む際に利用することができる。この関数の使用方法の詳細は5.1節を見よ。

### 2.5.2 GEO-EAS データのインポートとエクスポート

GEO-EAS は、空間に分布するデータを 2 次元の地理統計学的に解析するための対話的ツールを含む DOS アプリケーションである（Englund and Sparks, 1992）。GEO-EAS はテキストデータを特殊なフォーマットで読み込む。空間解析を行う他のソフトウェアの中にはこの GEO-EAS ファイルフォーマットのデータを使うものもいくつか存在する。GEO-EAS によって生成されたデータは、S+SPATIALSTATS関数 `read.geoeas` を以下のように使って取り込むことができる：

```
> read.geoeas(file)
```

ここで *file* は読み込みたい GEO-EAS データファイルの名前を含む文字列である。ファイル内の変数の数と同じ列数を持ったデータフレームが作られる。タイトル行や変数名とともに並んだ測定単位は無視される。

S+SPATIALSTATS にはまた、S-PLUS のデータフレームを GEO-EAS フォーマットのテキストファイルに書き出す関数 `write.geoeas` も存在する。オプションとして、ファイル名や各変数の単位、改行文字を指定することもできる。この関数に関する詳細はヘルプファイルを参照。

### 2.5.3 ARC/INFO データのインポートとエクスポート

S-PLUS 製品 S+GISLINKは、S-PLUS と ARC/INFO の間でデータのやり取りを行なうツールを提供している。S+GISLINK は coverage と auxiliary info table として保存されているデータを変換する関数を提供している。詳しくは *S+GISLINK User's Manual* を参照。

## 2.6 S+SPATIALSTATS で利用可能なサンプルデータ集

本書全体を通して例の中で使われる空間データの多くは S-PLUS オブジェクトとして S+SPATIALSTATS の中に含まれている。これらのデータセットのヘルプファイルはオンラインヘルプでも見ることができる。

# 3 空間データの視覚化

この章では、S-PLUS と S+SPATIALSTATS による空間データの探索的データ解析（EDA, exploratory data analysis）と視覚化の方法について紹介する。EDA とは、データとその構造を記述することで仮説を定式化したり、仮説の有効性をチェックしたりするのに役立てる方法のことである。一般に、データの分布や、局所定常性または大域的定常性（第 1 章で定義済み）からのずれ ―トレンドや外れ値を含む― に着目する。本来、解析者は、その空間解析とモデリングの初めから最後まで EDA の手法を用いるのであるが、この章では、解析が試みられる前の初歩的な探索に的を絞る。空間データのそれぞれのタイプ ―地理統計データ、格子データ、空間点パターン― に特有な解析の手法は、それぞれ第 4 章、第 5 章、第 6 章で議論する。

この章では以下の事柄について学習する:

- S-PLUS を使って EDA の基本ツールを学ぶ（3.1 節）
- EDA の基本的な手法を地理統計データに適用する（3.2 節）
- EDA の基本的な手法を格子データに適用する（3.3 節）
- EDA の基本的な手法を点パターンデータに適用する（3.4 節）
- hexagonal binning を適用する（3.5 節）

## 3.1 EDA ツール

この節に登場する手法のほとんどは通常の S-PLUS パッケージで利用可能である。よって、S-PLUS に詳しい読者は以降の章に登場する例まで読み飛ばしたいと思われるかも知れない。この節で紹介する 3 つの主な EDA の手法は：

- 記述統計量（3.1.1 節）
- プロットとグラフィックス（3.1.2 節）
- 分類とクラスタ法（3.1.3 節）

### 3.1.1 記述統計量

データの持つ性質を一目で見ることができるのが記述統計量である。これらの中には外れ値を発見したり、正規性からのずれを見るシンプルな方法が含まれている。S-PLUS 関数 summary を使うと要約統計量を簡単に得ることができる。データフレーム aquifer で例を示すと：

```
> summary(aquifer)
     easting            northing               head
Min.    :-145.20   Min.    :  9.414   Min.   :1024
1st Qu. : -21.30   1st Qu. : 33.680   1st Qu.:1548
Median  :  11.66   Median  : 59.160   Median :1797
Mean    :  16.89   Mean    : 79.360   Mean   :2002
3rd Qu. :  70.90   3rd Qu. :131.800   3rd Qu.:2540
Max.    : 112.80   Max.    :184.800   Max.   :3571
```

データセット aquifer には、西テキサスのウルフキャンプ浸水層の水頭高データ（海抜、フィート）[（Cressie, 1989, p.212）、（Harper and Furr, 1986）]が入っている。データフレームの easting と northing の列は位置を示すものである。（位置のデータを外して）水頭高データだけの要約統計量を見たければ、次のコマンドを用いる：

```
> summary(aquifer$head)
 Min. 1st Qu.  Median Mean 3rd Qu. Max.
 1024    1548    1797 2002    2540 3571
```

平均と中央値のみに興味がある場合は、関数 mean か median を用いる。水頭高データの平均と中央値の間には大きな差があり、分布が歪んでいるか、無視できない外れ値が存在することを示唆している。

データの 1 次元分布を見るシンプルな方法に、茎葉図を使う方法がある：

```
> stem(aquifer$head, twodig=T)
N = 85   Median = 1797   Quartiles = 1548, 2540
```

```
Decimal point is 2 places to the right of the colon
  10 : 24,30,38,89,92
  11 : 61
  12 : 31
  13 : 06,32,64,76,84,86
  14 : 02,08,15,37,64,66,76
  15 : 27,48,79,91
  16 : 06,11,38,74,80,82
  17 : 02,14,22,25,29,35,36,39,56,57,71,77,97
  18 : 05,06,28,65,68
  19 : 99
  20 : 03
  21 : 18,58
  22 : 00,38
  23 : 00,52,86
  24 : 00,32,55,68
  25 : 28,33,40,44,53,53,60,75,94
  26 : 46,48,50,91
  27 : 28,29,36,66,98
  28 : 11
  29 : 46
  30 :
  31 : 36
  32 :
  33 :
  34 : 90
  35 : 10,71
```

茎葉図は横向きのヒストグラムであるが、より多くの情報を持っており、出力から直接個々の値を読むことができる。個々の値は**葉**（*leaf*）であり、等間隔に区切られて並んだ**茎**（*stem*）によって分類される。コロンより左の値は水頭高データの最初の２桁であり、コロンより右の値は最後の２桁である。省略可能な引数 `twodig=T` が指定されると、４桁すべてが表示される。例えば、21:で始まる行は、2096 と 2195 の間には２つのデータが存在することを示している；2118 と 2158 である。

水頭高データの分布は、幾つかの大きな外れ値（3490、3510、3571）と異常に小さな値のグループ（1024、1030、1038、1089、1092）を除くと、２つの山があるようである。以降の解析は、この特異な性質を念頭において行うべきである。

### 3.1.2 プロットとグラフィックス

データセットの頻度の分布をプロットする最もよく知られた方法がヒストグラムである。図 3.1 を表示させるには以下のようにする：

```
> motif()
> par(mfrow=c(2,1))
 > hist(aquifer$head)
> hist(aquifer$head, nclass=20, xlim=c(1000,4000))
```

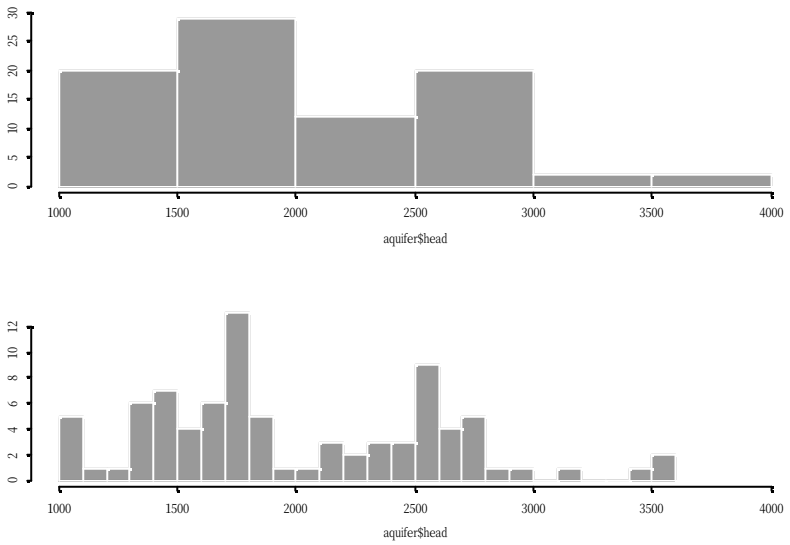


**図 3.1.** ウルフキャンプ水頭データのヒストグラム。

S-PLUS でプロットを行うには、まずグラフィックデバイスを開かなければならない（この例では `motif` を選んだ）。他にも `openlook`、`trellis.device`、Windows 版には `win.graph`、`graphsheet`（バージョン 4.0 以降）などがある。

関数 `par` はグラフィックデバイスの初期パラメータを変更するのに用いられる。この場合、２つのプロットを同時に見たい（ページを 2 行 1 列に分割したい）ので、パラメータ `mfrow` を変更する。

図 3.1 の最初のヒストグラムは、デフォルトの階級（棒）数を使っているので、茎葉図で明らかになった 2 つの山と外れ値があらわれていない。2 番目のヒストグラムでは、省略可能な引数 `nclass=20` を加えて、棒の数を増やしている。同時に、省略可能な引数 `xlim=c(1000,4000)` で x 軸の範囲を指定し、2 つのヒストグラムが比較できるようにしている（y 軸についても `ylim=` で同様のことができる）。この図では水頭高データの分布に 2 つの山と極めて大きい外れ値と思われる値が現われているのが分かる。

データセットが幾つかの層に別れるような変動を示す場合、Trellis ライブラリの関数 `histogram` を使うことでそれぞれの層のヒストグラムを一度に表示することができる。これを行うために、まず水頭高データを、頻度分布を見て決めた自然な分割に基づいて 4 グループに分割する。

```
> head.groups <- cut(aquifer$head,
+      breaks=c(0,1100,2000,3000,3600))
> trellis.device(color=F)
> histogram(~ aquifer$head|head.groups)
```

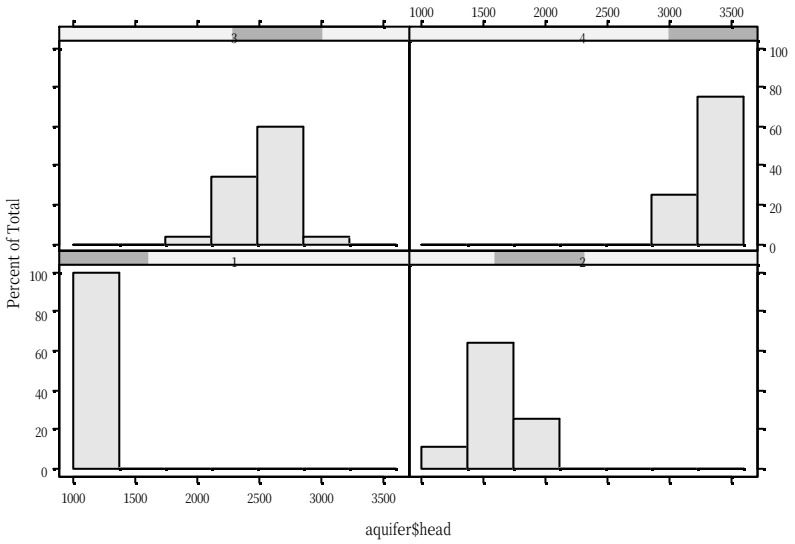


**図 3.2.** Trellis グラフィックスを使った水頭の層別ヒストグラム。

図 3.2 はそれぞれのグループのヒストグラムを示している。Trellis グラフィックスを表示させるには、グラフィックウィンドウ `trellis.device` が最適である。ディスプレイがカラーでない場合、引数 `color=F` を使わなければならない。Trellis グラフィックスに関する詳

しい情報は *S-PLUS Trellis Displays* ユーザーズマニュアルまたは *S-PLUS Programmer's Guide* を見よ。

水頭高データには２つの山があることをすでに見てきた。非正規性をさらに調べるためには、標準正規分布に対する水頭高データの正規 QQ プロットを行う：

```
> qqnorm(aquifer$head)
> qqline(aquifer$head)
```

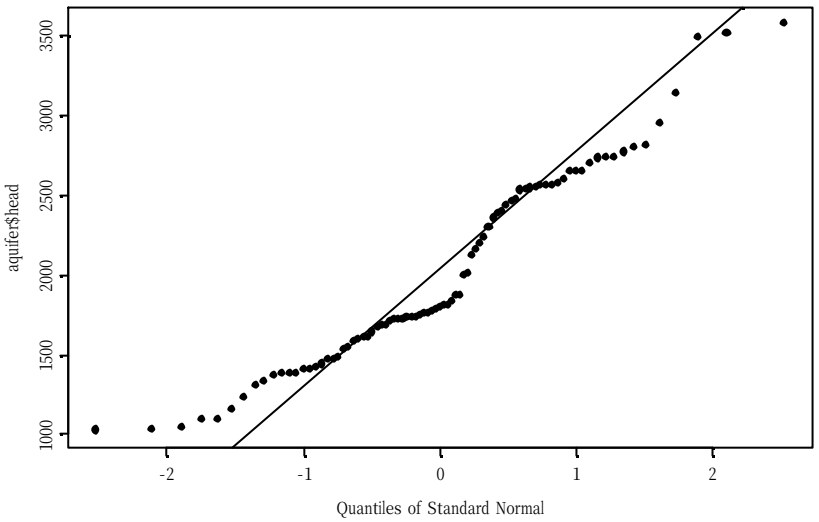


**図 3.3.** 水頭データの正規 QQ プロット。

もし正規分布から得られたデータであれば、`qqnorm` によって得られた図 3.3 のような点はほぼ直線になる。関数 `qqline` を使えば、比較のために `qqnorm` の点にあてはまる直線を引くことができる。点がシグモイドな（sigmoidal）パターンを示していることから、水頭高データが正規分布に従う傾向にないことは明らかである。

データの分布に関する性質を見たところで、さらに探索を進めて空間的な情報を得ようと思う。最初は単に位置をプロットしてみる。水頭高データの収集位置に対する図 3.4 のような散布図を得るには：

```
> scaled.plot(aquifer$easting,aquifer$northing)
```

関数 `scaled.plot` は空間データの縦横の比を正確に保ってプロットする。これは空間データの探索に重要である。

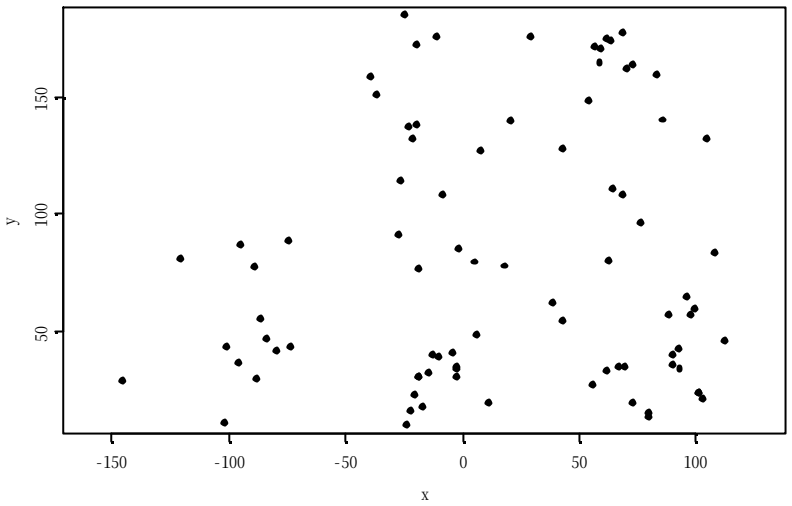


**図 3.4．ウルフキャンプのデータ収集地。**

**水頭高データのヒストグラム（図 3.1）に見られたデータグループが空間的にどう分布しているのかを調べるために、グループごとに異なる記号を使ってプロットを行う：**

```
> scaled.plot(aquifer$easting,aquifer$northing,
+      type="n")
> text(aquifer$easting, aquifer$northing,
+      labels=head.groups)
```

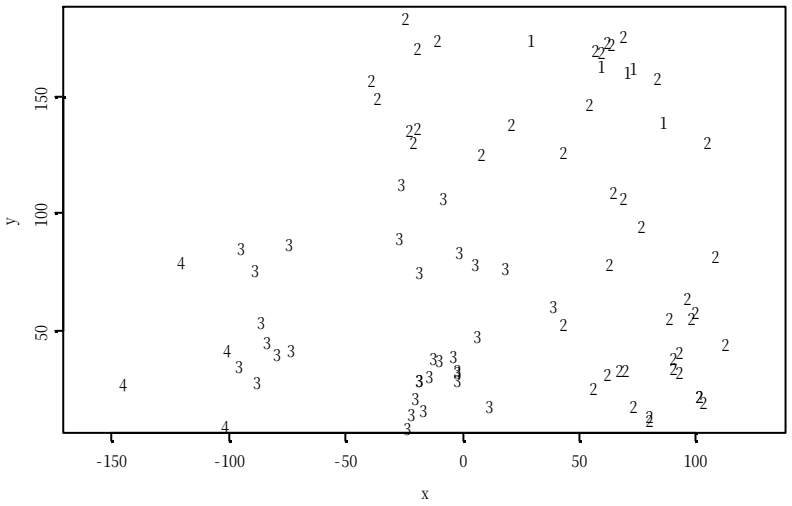


**図 3.5．値の 4 グループを表示したウルフキャンプの位置。**

関数 plot の引数 type="n"は縦横の比が正確な軸のみを出力する（これは関数 text を使って、点に対応するテキストラベルを付けるのに必要な準備である）。図 3.5 には明らかな空間トレンドが見られる。値が最も高いグループ（グループ 4）が左下に片寄っている。

図 3.5 で見たトレンドを確認するため、2 次元トレンドを視覚化するツールを使って図 3.6 を描く。その手順は次のように行う：

```
> par(pty="s", mfrow=c(1,2))
> attach(aquifer)
> int.aq <- interp(easting, northing, head/1000)
> contour(int.aq)
> points(easting,northing)
> image(int.aq)
> detach("aquifer")
```

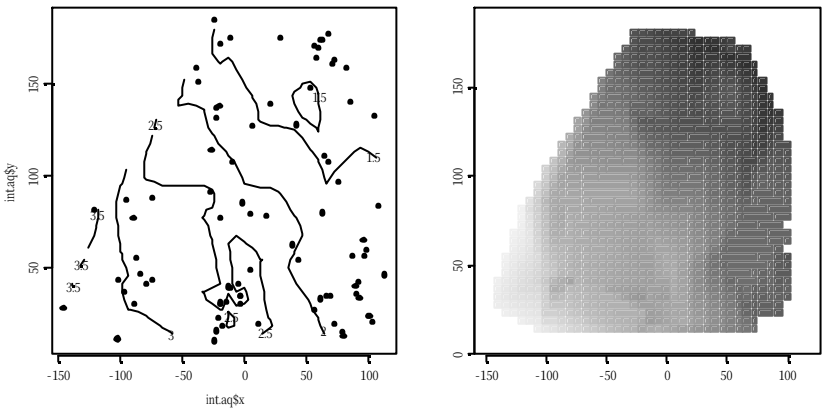


**図 3.6.** ウルフキャンプの水頭データの空間トレンドを見る 2 つの方法：等高線図（左）；イメージ図（右）。

コマンド par(pty="s")は正方形のプロット領域を作成する。こうすると、位置データを表示するのに都合のよいことが多い。S-PLUS のデフォルトのプロット領域は pty="m"、使用できる最大限の領域であり、縦横の比率が変化するため、グラフが変形するおそれがある。関数 attach はデータフレーム aquifer を検索リストの中で 2 番目に置き、以下の関数の中でデータフレームの各列を呼び出す際、その名前だけで使えるよう

にしている。

**注意：** attach に渡したデータフレームは、使わなくなったときには関数 detach を用いて検索リストから消去しておくことをお薦めする。

関数 interp は水頭高値を、緯度経度が等間隔のグリッド上に乗るように補間する。値は 1000 で割ってプロットの際に読み易いようにする。でき上がったリストは、x 軸 y 軸とそれに相当する z の値の行列で、等高線プロットやイメージプロットを行うのに必要なデータ構造をしている。図 3.6 の両方が空間トレンドを示している。

トレンドを表示するのに有効な、3 次元プロットも幾つかある。水頭高データに対して：

```
> par(mfrow=c(1,1))
> persp(int.aq, zlab="Z/1000", eye=c(300,-1500,15))
> trellis.device(color=F)
> cloud(head ~ easting*northing, data=aquifer)
```

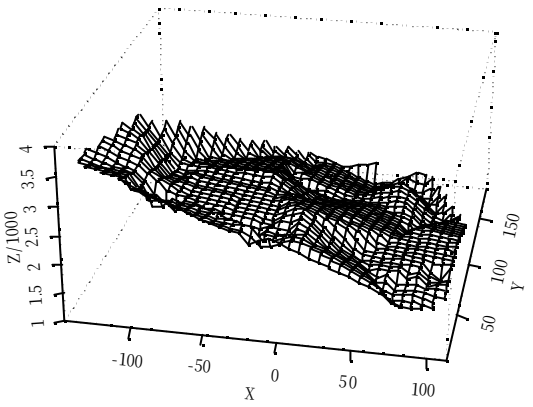


**図 3.7.** 水頭高データの 3 次元鳥瞰図。

図 3.7 は関数 persp によって描かれた鳥瞰図である。画像を回転させるのに省略可能な引数 eye を用いた。図 3.8 は Trellis ライブラリの関数 cloud を使って描いた 3 次元散布図である。3 次元図形にも空間トレンドは現われている。

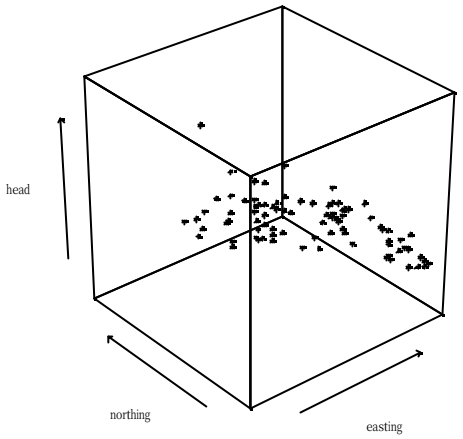


**図 3.8.** 水頭高データの 3 次元散布図。

### 3.1.3 分類とクラスタ法

多変量データに対する標準的な分類とクラスタ法が S-PLUS で利用可能である。この節では、回帰樹とモデルに基づいたクラスタ法について簡単に説明する。階層クラスタリング（`hclust`）、因子分析（`factanal`）、判別分析（`discr`）、主成分分析（`princomp`）など、他の多変量解析法に関する情報はそれぞれの関数のヘルプファイル、*S-PLUS Guide to Statistics*、参考図書に挙げた本などで得ることができる；例えば（Venables and Ripley, 1994）など。

回帰樹は、多変量データの構造を見つけるための探索的手法として使われるが、空間データ解析では、データが変化する局所的な近傍を発見する手助けになる。S-PLUS の関数 `tree` は、反応変量を予測変量に基づいて分類する。よって水頭高データを位置によって分割すれば、前に見たようなグループが現われることが期待される。

```
> aqtree <- tree(head~easting+northing, data=aquifer,
+      mindev=.001)
> plot(aqtree)
```

引数 `mindev` は各ノード間の尤離度の下限を設定する。かなり大きな樹から始めるために、尤離度を小さくしておく。コマンド `plot` は樹構造を表示する（図 3.9）。デフォルトでは、枝の長さは分離条件の重要度に応じて決定される。

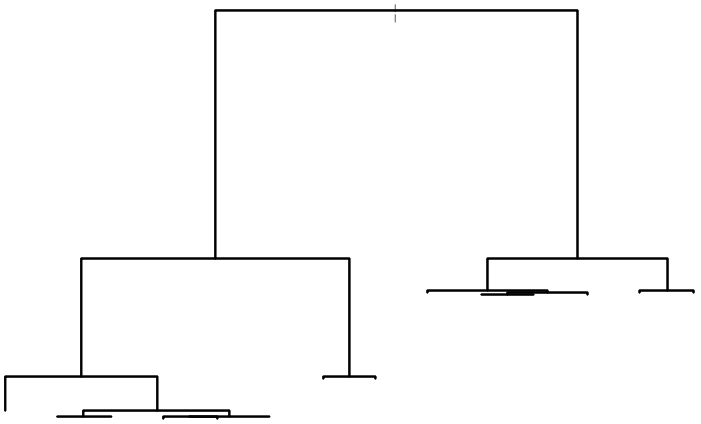

**図 3.9.** 水頭高データに対する回帰樹。

図 3.9 における下部の枝のいくつかは、あまり重要でないように思われる。S-PLUS 関数 `prune.tree` は最適な大きさまで枝を剪定する。まず、尤離度のプロットを見て適当と思われる大きさを選ぶ：

```
> plot(prune.tree(aqtree))
```

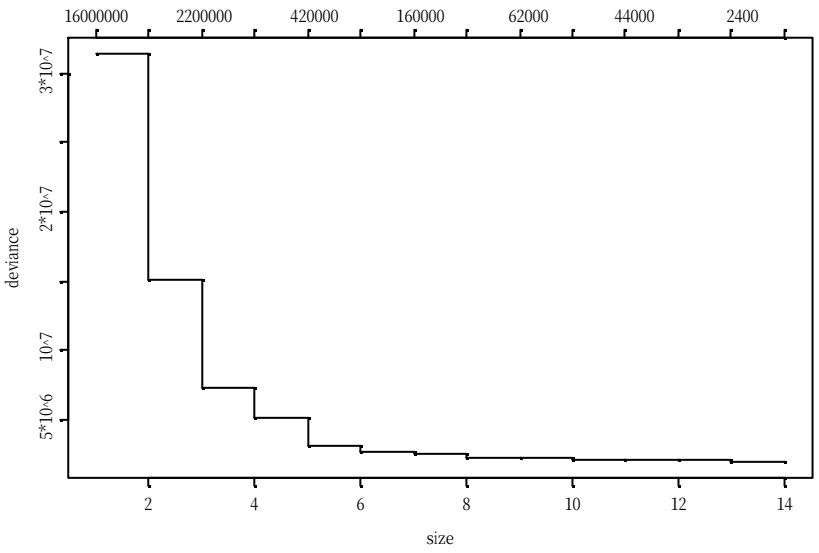


**図 3.10.** 水頭高データに対する回帰樹の、サイズ増加に伴って減少する尤離度の様子。

図 3.10 は、5 番目以降のノードの尤離度が実質上の分割に寄与していないことを示している。樹を重要度の高い 5 つのノードを残して剪定し、結果を表示するには以下のようにする：

```
> aqtree2 <- prune.tree(aqtree, best=5)
> par(mfrow=c(1,2))
> plot(aqtree2, type="u")
> text(aqtree2, srt=90)
```

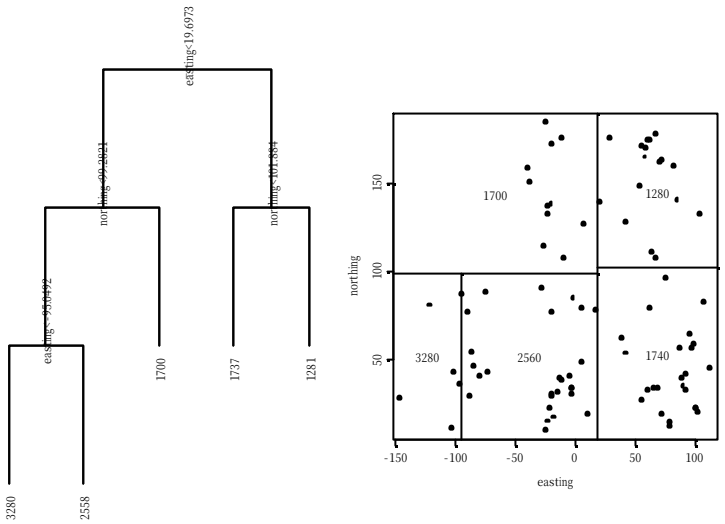


**図 3.11.** 剪定された水頭高データの回帰樹と分割のプロット。

図 3.11 の左側は剪定された回帰樹である。コマンド `plot` 内で引数 `type="u"`を使うことで、枝の長さを一様にできる。コマンド `text` が樹オブジェクトに対して用いられると、ノードそれぞれに情報を書き入れる。省略可能な引数 `str=90` は字句を回転させて互いに重ならないようにしている。回帰樹の端点のノードに書かれた数値は、分割後にそのノードに残ったデータの平均である。

回帰樹のノードの、空間における位置を調べるには以下のようにする:

```
> par(pty="s")
> partition.tree(aqtree2)
> points(aquifer$easting, aquifer$northing)
```

関数 `partition.tree` は関数 `tree` によって生成された空間分割をプロットする。図 3.11 の右側がその結果である。グループの大きさを判断できるように点を添えた。これらの分割によっても図 3.6 の空間トレンドは

示された。

樹に関連する S-PLUS 関数は他にも回帰樹の縮小（shrink.tree）、ブラウジング（browser）、切り捨て（snip.tree）などがある。詳しくは *S-PLUS Guide to Statistics* のそれぞれのトピックを見よ。

クラスタ分析は多変量データの構造を探るためのもう 1 つの方法である。S-PLUS 関数 mclust は 6 つのモデルまたは 5 つの発見的判断基準による凝集型階層クラスタリングを行う。mclust を使ったときに可能な判定基準の種類については *S-PLUS Guide to Statistics* を見よ。凝集型階層クラスタリングはそれぞれのオブジェクトを異なったグループとした状態からスタートし、ある距離尺度に基づいてオブジェクトをまとめ、次第に大きなクラスタを生成してゆく方法である。他のクラスタ法、例えば iterative relocation は、初期分類からスタートし、それをより良い分類になるように繰り返し改善する方法である。iterative relocation clustering は S-PLUS 関数 kmeans を使って行なえる。

クラスタリングの手法を用いた水頭高データの探索的データ解析は以下のようにして行なわれる：

```
> mclust.aq <- mclust(data.matrix(aquifer))
> xy <- plclust(mclust.aq$tree,labels=F)
> lab.aq <- as.character(aquifer$head)
> labclust(xy, lab.aq, cex=.75)
```

関数 mclust は行列型のデータを要求するため、最初に水頭高データフレームに関数 data.matrix を使っている。関数 plclust によって描かれる図 3.12 の樹構造は水頭高データのクラスタを表わしている。これにはデフォルトのクラスタ判定基準（S*）が使われている。各クラスタには labclust を使ってデータ値をラベルとして書き入れた。省略可能な引数 cex=.75 は文字の高さをデフォルトの高さの 75 パーセントにしている。

**警告：** mclust によって生成されるクラスタは、データのスケールに依存しない。空間に関する単位は 1 つではないので、この手法は注意して用いなければならない。

mclust が生成した上位 5 つのクラスタが空間的にどのように位置するかを見るために，図 3.13 のようなクラスタのプロットを行う：

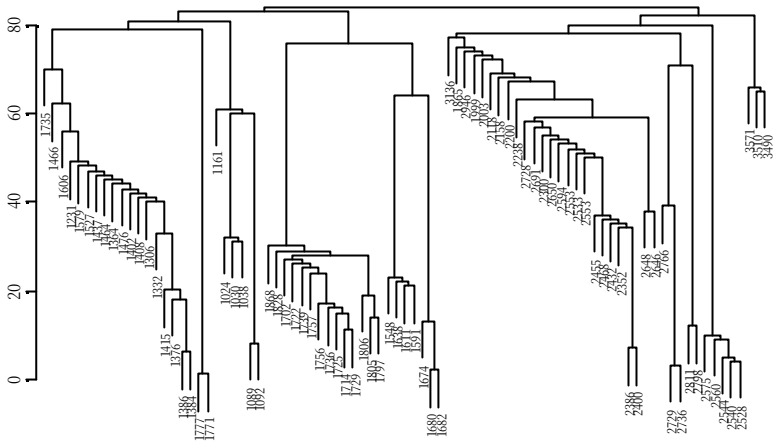

**図 3.12. 水頭高データに、モデルに基づくクラスタリングを行なって生成した樹。**

```
> aquifer.5 <- mclass(mclust.aq, 5)
> scaled.plot(aquifer$easting, aquifer$northing,
+ type=”n”)
> text(aquifer$easting, aquifer$northing,
+ aquifer.5$class)
```

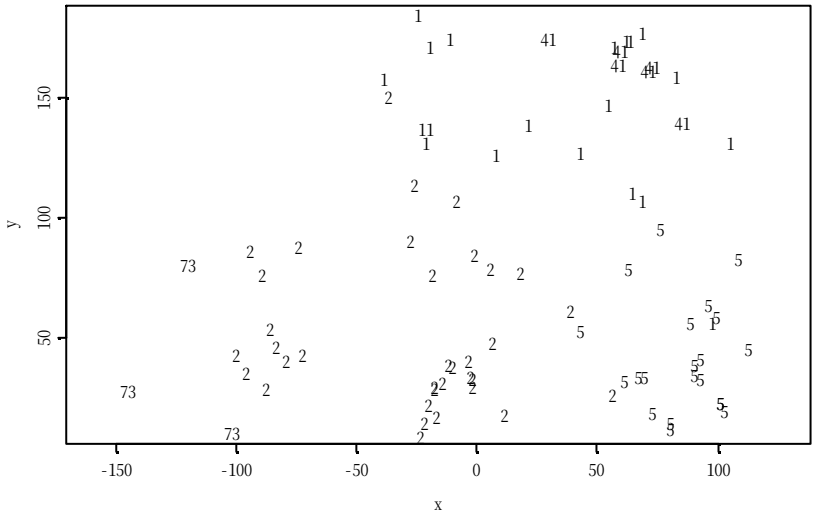

**図 3.13. 水頭高データのモデルに基づくクラスタリングにより生成された 5 つのクラスタの地図上での位置。**

関数 mclass は mclust が生成したオブジェクトを引数に取り、元データをクラスタに分類したベクトルを返す。デフォルトでは各クラスタには、図 3.13 で示すように、そのクラスタに含まれる最初の要素のデータフレームにおける行番号がラベルとして付加される。例えば、「73」のラベルがついたクラスタの要素で、データフレーム上で最初に登場する要素は 73 番目である。図 3.5 では、ヒストグラムで見た 2 つの山をもとに分類したグループの空間上の位置をプロットした。それと図 3.13 を比較してほしい。分類のされ方は類似しているが、以前には気付かなかった南東の角に新たなクラスタ（「5」のラベル）が加えられている。

## 3.2 EDA の地理統計データへの応用

この節では、EDA を**地理統計データ** ―連続的な空間上で採集されたデータ（より正確な定義は第 1 章を参照）― に用いた例を紹介する。等間隔なグリッド上の位置で採集されたデータに対して EDA を行う例として、データフレーム coal.ash を用いる（3.2.1 節）。グリッド上で採集されていないデータに対して EDA を行う例には、データフレーム scallops を用いる（3.2.2 節）。

### 3.2.1 例：グリッドデータ

データフレーム coal.ash はペンシルバニア州グリーン郡ロベナ鉱山のピッツバーグ炭層のデータである［（Cressie, 1993, p.32）、（Gomez and Hazen, 1970）］。データフレームには、xy 平面座標で与えられるグリッド上で採集された 208 のコアサンプルの石炭含有率が入っている。図 3.14 のようにグリッドの位置をプロットするには以下のようにする：

```
> plot(coal.ash$x, coal.ash$y, type="n", lab=c(16,23,7))
> text(coal.ash$x, coal.ash$y,
+      format(round(coal.ash$coal,1)))
```

図 3.14 には採集位置にその点におけるデータが表示されている。作図パラメータ lab は関数 text 内で使われると、各軸の目盛の刻み幅と軸ラベルの長さを指定する（作図パラメータについてのより詳しい解説はヘルプファイルを参照）。データは関数 format と関数 round によって小数点第 1 位以下に丸めてある。石炭含有率の分布に関する情報を得る

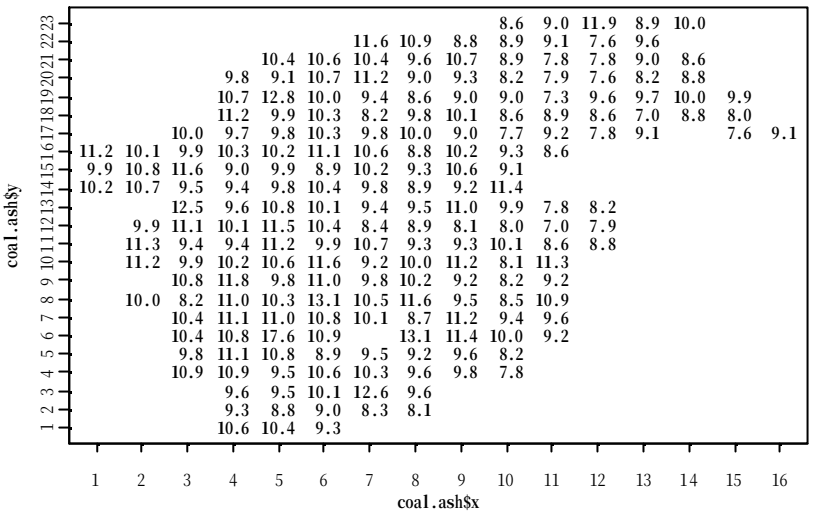


**図 3.14. サンプル位置におけるコア中の石炭含有量（%）。**

には、以下のようにして記述統計量を見ればよい：

```
> summary(coal.ash$coal)
  Min.  1st Qu.   Median    Mean  3rd Qu.    Max.
     7     8.96    9.785   9.779    10.57   17.61
> stem(coal.ash$coal)

N = 208   Median = 9.785
Quartiles = 8.96, 10.575

Decimal point is at the colon

   7 : 003
   7 : 666788888999
   8 : 0111122222234
   8 : 56666666778888899999999
   9 : 000000001111111222223333333344444
   9 : 55555566666666677888888888999999999
  10 : 00000000111111122222233333444444444
  10 : 55666677777888888999999
  11 : 00000111112222223344
  11 : 5666679
```

```
  12 :
  12 : 568
  13 : 11

High: 17.61
```

要約統計量と茎葉図によると、データはほぼ正規分布で、外れ値が数個あるようである。関数 stem の出力 High: 17.61 より、少なくとも 1 点 17.61 は大きすぎて茎葉図に入らなかったようである（省略可能な引数 fence を指定して、stem の呼び出し内で外れ値の定義を変えてみよ）。

これまでは、**大域的な**（*global*）領域内での外れ値やトレンドのみを見てきた。空間データ解析では、**局所的な**（*local*）近傍の点内で変則的な観測値も興味の対象になる。データはグリッド上にあるので、グリッドの行や列に関する局所定常性からの逸脱を発見するのは容易である。これを行うには図 3.15 や図 3.16 のような箱ひげ図を用いる。まず、外れ値と思われる値を除去し、図の縮尺に悪影響を及ぼさないようにする：

```
> coal.ash[coal.ash$coal==max(coal.ash$coal),]
    x  y   coal
50  5  6  17.61
> trellis.device(color=F)
> bwplot(y~coal, data=coal.ash, subset=-50,
+      main="Row Summaries")
> bwplot(x~coal, data=coal.ash, subset=-50,
+      main="Column Summaries")
```

Trellis ライブラリの関数 bwplot を使って、石炭データを行で分割したものと列で分割したものの箱ひげ図を描く（図 3.15 と図 3.16）。どちらのグラフにもいくつかの外れ値が見られる。これらの点はもっと詳しく調べる必要がある。また、列に関してはトレンドがあるように思われる。

Cressie (1993)によって提案された、局所定常性を見る他の方法に、行や列ごとに平均や中央値をプロットする方法がある。S-PLUS の関数 tapply は、mean や median といった関数をデータフレームの部分集合に当てはめる。図 3.17 を描くには以下のようにする：

1. ４つのグラフを一度に表示させるように画面を分割する。

```
> par(mfrow=c(2,2))
```

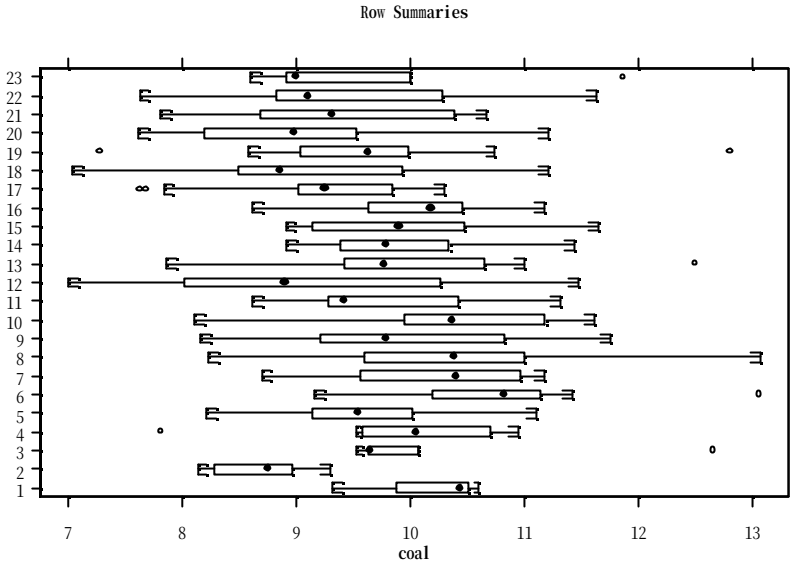


**図 3.15.** 行によって分割した石炭データの箱ひげ図。

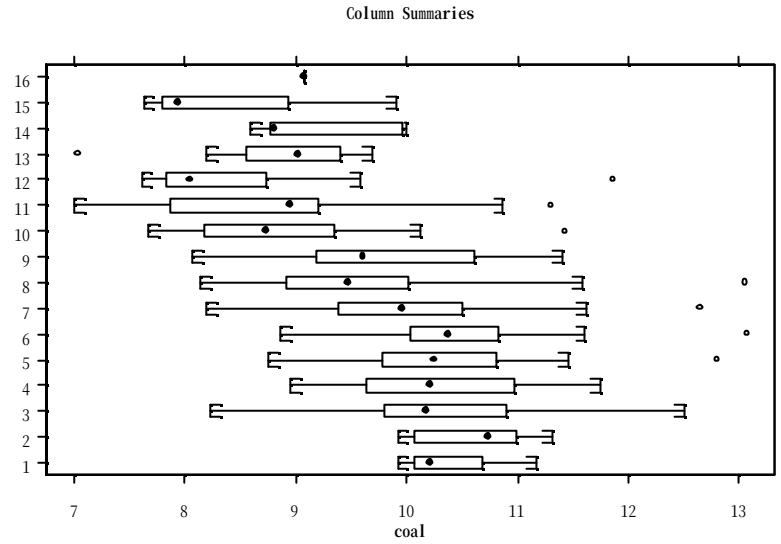


**図 3.16.** 列で分割した石炭データの箱ひげ図。

2. **軸やラベルを除いた点の位置だけをプロットする。**

```
> Plot(coal.ash$x, coal.ash$y, axes=F, xlab="",
+      ylab="")
```

3. **行ごとの中央値をプロットし、次に行平均をプロットする。**

```
> Plot(tapply(coal.ash$coal, coal.ash$y, median),
+     1:23, xlab="% coal ash", ylab="Rows",
+     xlim=c(8,12), pch="o")
```

```
> points(tapply(coal.ash$coal, coal.ash$y, mean),
+      1:23, pch="x")
```

4. 列ごとの中央値をプロットし、次に列平均をプロットする。

```
> Plot(tapply(coal.ash$coal, coal.ash$x, median),
+      1:23, xlab="% coal ash", ylab="Columns",
+      ylim=c(7,11), pch="o")
> points(1:16, tapply(coal.ash$coal,coal.ash$x,
+         mean), pch="x")
```

5. 凡例を描く。

```
> Plot(coal.ash$x, coal.ash$y, axes=F, type="n",
+      xlab="", ylab="")
> text(1,22, "o = Median % coal ash", adj=0)
> text(1,19, "x = Mean % coal ash", adj=0)
```

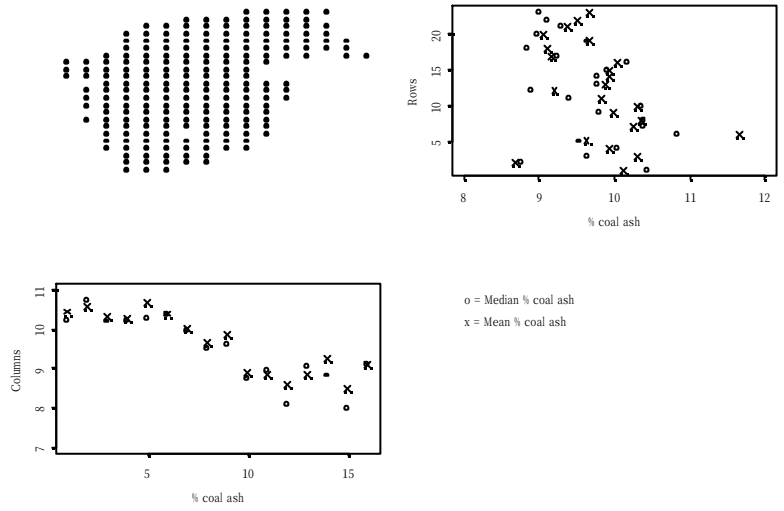


**図 3.17. 石炭データの要約（右上が行ごと、左下が列ごと）。**

箱ひげ図で見たような列に関するトレンドが図 3.17 でも明らかに見られる。同じ行または列の中での外れ値や分布の歪みがあるときは、平均値と中央値の間に大きな差が見られる。

局所定常性からのズレを検出するもうひとつの方法が、空間ラグプロット（各点とその点のある方向の近傍との二次元散布図を描く）を見る方法である。これを行うには以下のようにする：

1. x を列に、y を行に、石炭含有量を値に持つ行列を作る。

```
> grid.mat <- tapply(coal.ash$coal,
+       list(factor(coal.ash$x), factor(coal.ash$y)),
+       function(x)x)
```

2. 2 列目から 23 列目対 1 列目から 22 列目をプロットし、興味ある点を識別する。

```
> par(pty="s")
> plot(grid.mat[,-1],grid.mat[,-23],xlab="coal ash% in
+       column Z",ylab="coal ash% in column Z+1")
> identify(grid.mat[,-1], grid.mat[,-23],
+       label=grid.mat[,-1])
[1] 179 277  72  23
```

3. 2 行目から 16 行目対 1 行目から 15 行目をプロットし、興味ある点を識別する。

```
> plot(grid.mat[-1,],grid.mat[-16,],
+       xlab="coal ash% in row Z",
+       ylab="coal ash% in row Z+1")
> identify(grid.mat[-1,], grid.mat[-16,],
+      label=grid.mat[-1,])
[1]  79 110
```

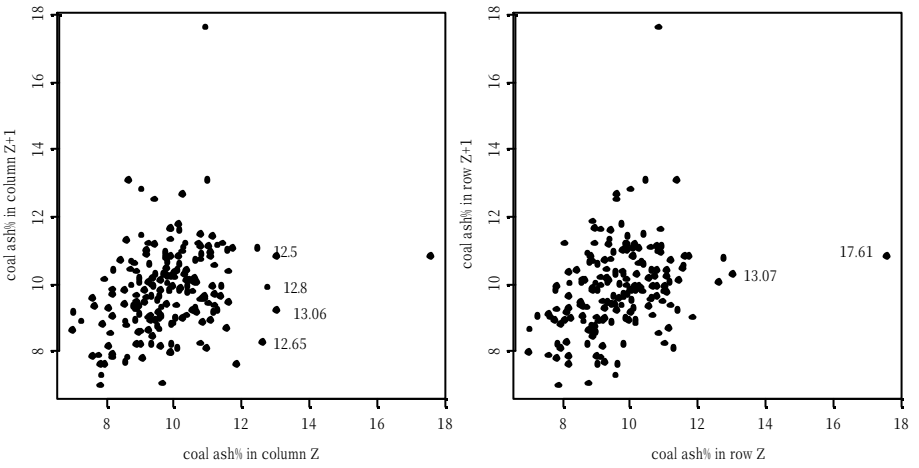


**図 3.18.** 各点の石炭含有量対その次の行または列にある点の石炭含有量の 2 変量散布図（左が列、右が行）。

図 3.18 の左側が列に関するラグプロット、右側が行に関するラグプロットである。ここでは S-PLUS 関数 identify を使って、外れ値と思われる点に対してラベルを付けている。以前に外れ値と判断した z[5,6]=17.61 はどちらのプロットでも顕著である。z[x,y]は第 5 行第 6 列における値を意味する。これ以外に、近傍の値と比較して変則的であると思われる点は Cressie (1993)によると、z[7,3]=12.65、z[8,6]=13.06、z[6,8]=13.07、z[3,13]=12.5、z[5,19]=12.8 などである。この探索の結果、異常な観測値を発見することができた。これらはバリオグラムモデルを当てはめる際に再評価する必要がある。

### 3.2.2 例：グリッド上にないデータ

アメリカ北東部の大西洋沿岸におけるホタテ漁獲量のサンプルデータセットは、グリッド上にない地理統計データの例である。データは米国海洋水産サービスの北東水産科学センターが層別無作為抽出法（Ecker and Heltshe, 1994）によって採集したものである。データフレーム scallops は 7 列で構成されている：

```
> summary(scallops)
       strata           sample             lat
 6310    :24       Min.    :   1.0   Min.    :38.60
 6270    :17       1st Qu. : 106.8   1st Qu. :39.46
 6230    :16       Median  : 147.0   Median  :39.98
 6340    :14       Mean    : 131.8   Mean    :39.91
 6300    :14       3rd Qu. : 185.2   3rd Qu. :40.41
 6260    :12       Max.    : 224.0   Max.    :40.92
(Other)  :51
        long             tcatch             prerec
 Min.    :-73.70   Min.    :   0.0   Min.    :    0.00
 1st Qu. :-73.14   1st Qu. :   8.0   1st Qu. :    1.00
 Median  :-72.74   Median  :  30.0   Median  :    8.00
 Mean    :-72.72   Mean    : 274.6   Mean    :  156.50
 3rd Qu. :-72.31   3rd Qu. : 115.2   3rd Qu. :   48.25
 Max.    :-71.52   Max.    :7084.0   Max.    : 4487.00
      recruits
 Min.    :  0.00
 1st Qu. :  5.00
 Median  : 21.50
```

```
 Mean    : 118.10
 3rd Qu. :  73.75
Max.     :2597.00
```

補充数（recruits）と補充前の数（prerec）の和が全捕獲数（tcatch）になる。

要約統計量を一目見ると、平均と中央値がかなり離れており、データがかなり歪んでいることが分かる。正規近似を行うために tcatch を対数変換する：

```
> scallops[,"lgcatch"] <- log(scallops$tcatch+1)
> summary(scallops$lgcatch)
  Min. 1st Qu.  Median   Mean 3rd Qu.   Max.
     0   2.197   3.434 3.483   4.756 8.866
```

ホタテデータの空間的特性を、散布図と等高線図によって見てみよう：

```
> library(maps)
Warning messages:
 Warning in library(maps): The functions and datasets in
library section maps are not supported by MathSoft.
> map("usa", xlim=c(-74,-71), ylim=c(38.2,41.5))
> points(scallops$long, scallops$lat, cex=.75)
> int.scp <- interp(scallops$long, scallops$lat,
+     scallops$lgcatch)
> contour(int.scp, add=T)
```

図 3.19 は捕獲位置と大西洋岸の海岸線の関係を表したものである。等高線を見ると、北東から南西の方向に向かって捕獲数の多い位置が尾根のように現れていることが分かる。また、捕獲数の急激な落ち込みも見られるが、これは水深が深すぎてホタテ貝が生息できないからであろう。この空間トレンドはこの先このデータのモデリングに影響を与えるので、もっと注意深く観察すべきである。

グリッド上にないデータのトレンドを発見するひとつの方法に、S-PLUS の線形回帰関数 lm によってデータを緯度と経度の多項式を使ったパラメトリックなトレンド曲面としてモデリングする方法がある。探索的な解析により適した方法は、データを緯度と経度の平滑化関数としてモデリングする方法である。S-PLUS でこれを行うには、関数 glm を使って、対数

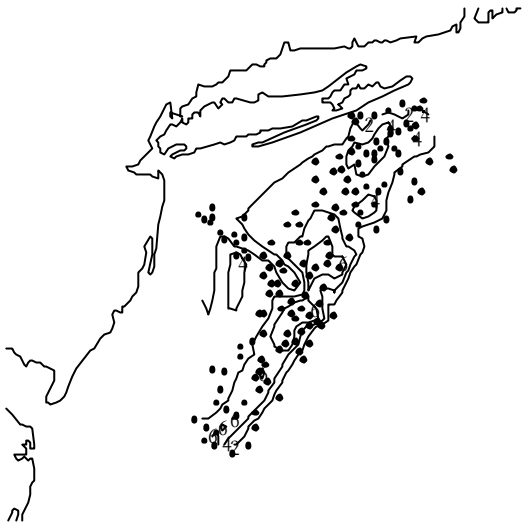

**図 3.19.** 米国北東部沿岸のホタテ貝捕獲量調査の位置と捕獲量の等高線。

をとった捕獲数データに、緯度と経度を予測変量に取った一般加法モデル（GAM, generalized additive model）を当てはめればよい。このモデルは独立性を仮定しているので、結果は注意深く扱う必要がある。

```
> gam.scp <- gam(lgcatch~lo(long)+lo(lat),
+                  data=scallops)
> par(mfrow=c(2,1))
> plot(gam.scp, residuals=T, rug=F)
```

関数 gam の中で使われている lo は、当てはめに局所回帰（loess）を使用することを指定している。図 3.20 には北東から南西にかけて走るトレンドが現れていない。当てはめられた曲線は平らではなく、何か強い結論を出すには残差が大きすぎる。

**警告：**この局所回帰曲線を残差抜きでプロットするとトレンドが顕著に表れるかもしれない。

緯度経度で与えられる平面座標系は唯一のものではない。図 3.19 で、海岸線にほぼ垂直に走るトレンドを発見した。このトレンドをはっきりと観測する（もしくは除去する）には、緯度経度の軸を回転させてみるとよいかもしれない。軸の回転を行う S-PLUS の関数を書くには以下のようにする：

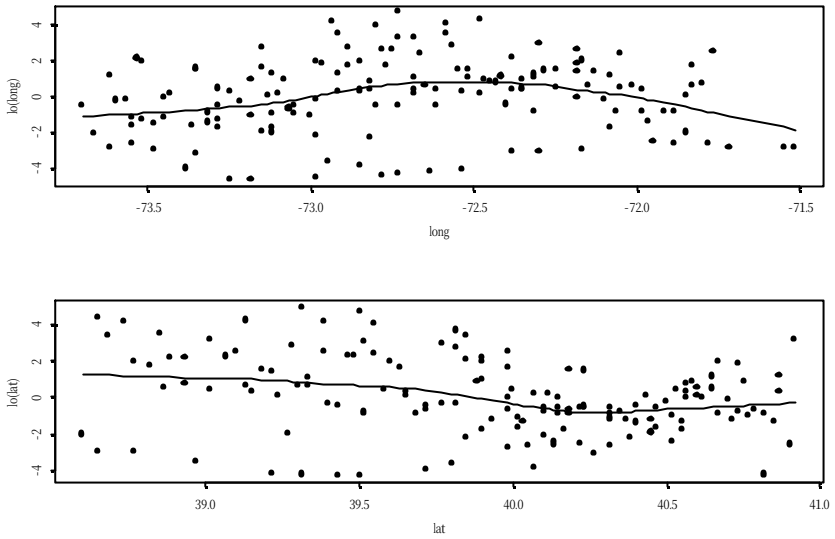


**図 3.20.** 緯度（上段）または経度（下段）に関する平滑化トレンドモデルのプロット。

```
> rotate.axis <- function(xy, theta)
+ {
+ # xy は n×2 の座標の行列
+ # theta は回転したい角度
+ # theta は負でも可
+        pimult <- (theta * 2 * pi)/360
+        newx <- c(cos(pimult), sin(pimult))
+        newy <- c( - sin(pimult), cos(pimult))
+        XY <- as.matrix(xy) %*% cbind(newx, newy)
+  as.data.frame(XY)
+ }
```

この関数を数回テストした結果、52 度（45 度と 60 度の中間）が適当であるように思われる。52 度回転した結果のプロット（図 3.21）と新たに gam を当てはめる（図 3.22）には以下のようにする：

1. 軸を回転し、結果をプロットする。

```
> xy <- scallops[c("long", "lat")]
> scall.rot <- cbind(rotate.axis(xy,52),
+      lgcatch=scallops$lgcatch)
> plot(scall.rot$newx,scall.rot$newy)
```

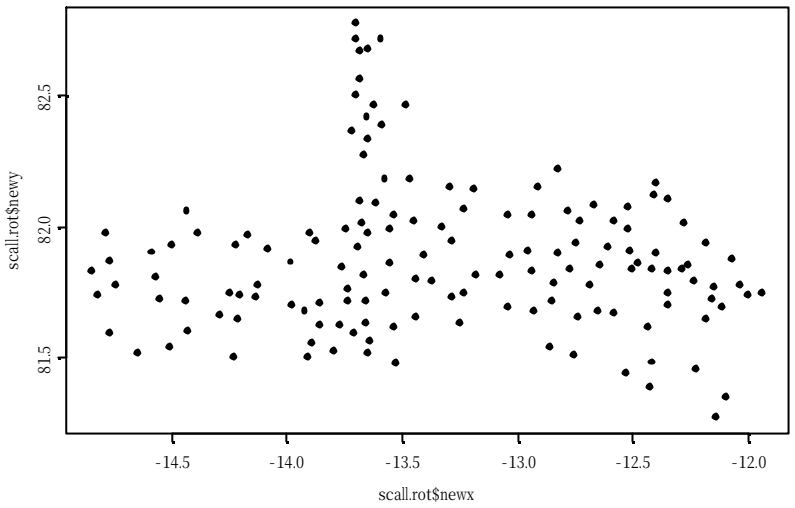


**図 3.21. 回転後のホタテ貝捕獲位置のプロット。北はプロットの右上になる。**

2. **回転したデータに** GAM **を当てはめ、結果をプロットする。**

```
> gam.scprot <- gam(lgcatch~lo(newx)+lo(newy),
+     data=scall.rot)
> par(mfrow=c(2,1))
> plot(gam.scprot, residuals=T)
```

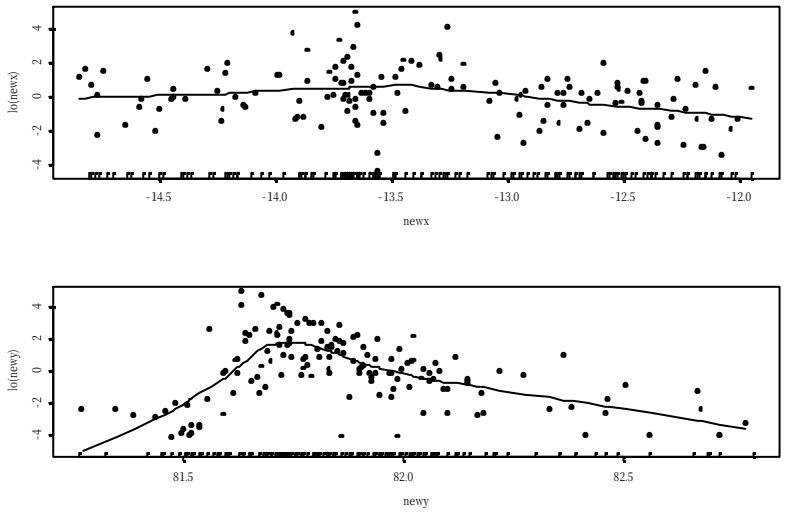


**図 3.22. 回転後の経度（上段）と緯度（下段）に平滑化トレンドモデルを当てはめた結果のプロット。**

図 3.22 の上段から、海岸線に沿った南西から北東への方向のトレンドは平坦であることがわかる。この方向にははっきりとしたトレンドは見られない。沖から陸へ向かう方向のトレンドは、南東から北西に向かう回転後の緯度によっておおよそ近似され、これは図 3.22 の下段に示されている。このトレンドは予想どおり水深がホタテの生息に適するところで急激に増加し,沖に行くにしたがって少しずつ減少している。

上で用いた加法モデルは 2 方向のトレンドしか見ることが出来なかった。トレンド曲面すべてをモデリングするには、関数 `loess` を以下のように用いる：

```
> loess.scp < - loess( lgcatch ~ newx*newy, data=scall.rot,
+      normalize=F, span=.25)
```

関数 `loess` はデータに局所回帰モデルを当てはめ、クラス"`loess`"のオブジェクトを返す。ここでは省略可能な引数 `normalize=F` を使用して、予測値の正規化を行わないようにした。この場合の予測値は任意の位置だからである。また、平滑化パラメータを設定する引数 `span` を.25 に設定した。この値を大きくするとよりスムーズな予測を行ない、小さくすると逆の効果が得られる。このモデルからの残差は第 4 章のクリギング予測を行うのに用いられる。局所回帰モデルについての詳細は *S-PLUS Guide to Statistics* か Cleveland, et al. (1992)を参照。

局所回帰モデルによって推定されたトレンド曲面を見るには以下のようにする:

1. 局所回帰モデルによる予測値を、*newx* と *newy* の範囲のグリッド上になるように編成する。

```
> range(scall.rot$newx)
[1] -14.84007 -11.94120
> range(scall.rot$newy)
[1] 81.26967 82.77920
> lo.grid <- expand.grid(
+      newx = seq(-14.8,-11.9,length=50),
+      newy = seq(81.3,82.8,length=50))
> scp.pred <- predict(loess.scp, lo.grid)
```

上で用いた関数 `expand.grid` は *newx* と *newy* のすべての組合わせを含むような 50×50 のデータフレームを生成する。`loess` オブジェクトに対

して一般的な関数 predict を使ったときに呼び出される関数 predict.loess は、グリッド上の局所回帰予測値を求める。

2. 予測範囲を絞り、ホタテ貝捕獲量データが採集された範囲に合うようにする。

```
> scall.chull <- chull(scall.rot$newx, scall.rot$newy)
> scall.poly <- list(x=scall.rot$newx[scall.chull],
+     y=scall.rot$newy[scall.chull])
> inside <- points.in.poly(lo.grid$newx, lo.grid$newy,
+     scall.poly)
> scp.pred[!inside] <- NA
```

関数 chull はサンプル採集が行なわれたサイトの凸包を形成する点のインデックスを返す。関数 points.in.poly は、グリッド上の各点について、chull によって作られる多角形に含まれるかどうかを示す論理ベクトルを返す。

3. 予測値を見る。

```
> persp(scp.pred)
```

局所回帰予測値のプロットが図 3.23 である。

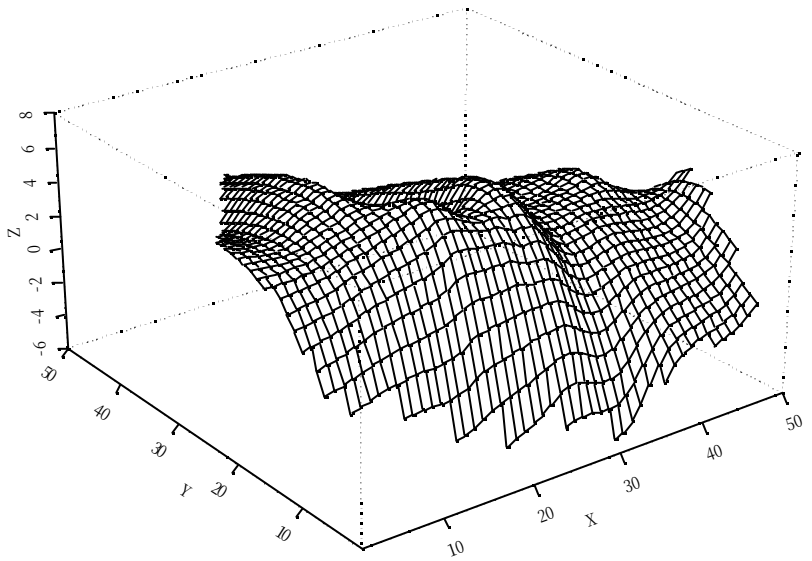


**図 3.23.** 対数をとったホタテ捕獲数データの局所回帰モデルによる予測値の鳥瞰図。

ここまで、データ scallop に対しては大域的定常性に焦点を当てていた。各点の局所近傍についてはどうだろうか。グリッド上に並んだデータセットに対しては行ごとのあるいは列ごとの要約統計量を見たり、空間ラグプロットを行ってそれを検証することができた。グリッド上にないデータに対してこれを行うには、点をグループ化して人工的なグリッドデータにするしか方法がない。

S-PLUS で各点の局所近傍を観察するひとつの方法は、Trellis ライブラリの関数 xyplot を用いる方法である。ホタテ捕獲数データを近傍または「かたまり」（*shingles*）に分割し、xyplot を用いて以下のようにプロットを行う：

```
> y.shing <- equal.count(scall.rot$newy, number=6,
+      overlap=.25)
> xyplot(lgcatch~newx|y.shing, data=scall.root)
```

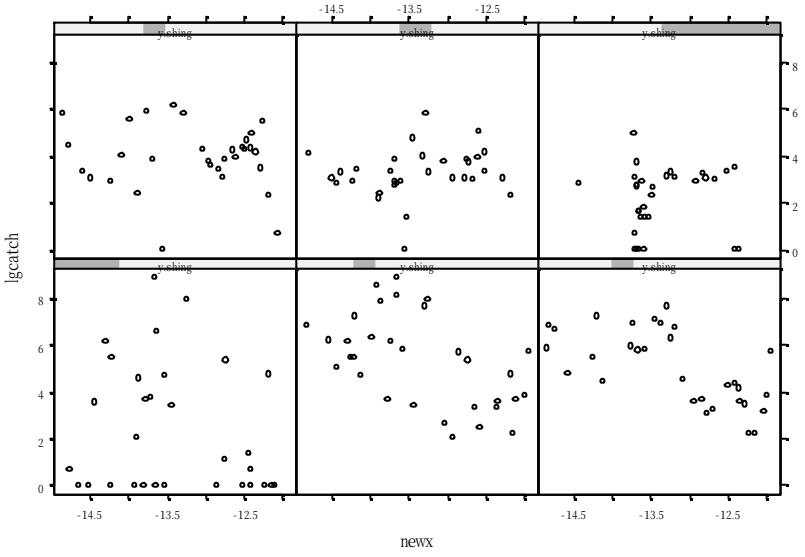


**図 3.24.** 対数を取ったホタテ捕獲数データを、*newy* 軸によって分割したものの散布図。

関数 equal.count は「かたまり」を作る（*newy* 軸の値をもとに、データを 25%の重複を許して 6 つの等しい大きさのグループに分割する）。図 3.24 はホタテ捕獲数を、回転後の軸 *newy* の値に応じて分割したものの散布図である。数個の局所的な外れ値（0 に近い値も含む）が 4 番目（上段左）と 5 番目（上段中央）の「かたまり」に見られる。

ホタテ捕獲数データの要素 strata には、米国海洋水産サービスが分類

した位置の階層が収められており、これは局所的な近傍と考えることができる。よって、階層ごとに捕獲数の箱ひげ図を描いてその分布を見ることができる。観測値が 6 つ以下の層における「外れ値」は対象外にすることにして、これらのデータをプロット前に除去する。

```
> attach(scallops)
> table(strata)
6220  6230  6240  6250  6260  6270  6280  6290  6300
   8    16     5     3    12    17    10     5    14

6310  6330  6340  6350
  24    10    14    10
> scp.y <- lgcatch[strata!= 6240 & strata!= 6250
+      & strata!= 6290]
> scp.x <- strata[strata!= 6240 & strata!= 6250
+      & strata!= 6290]
> bwplot(scp.x ~ scp.y)
```

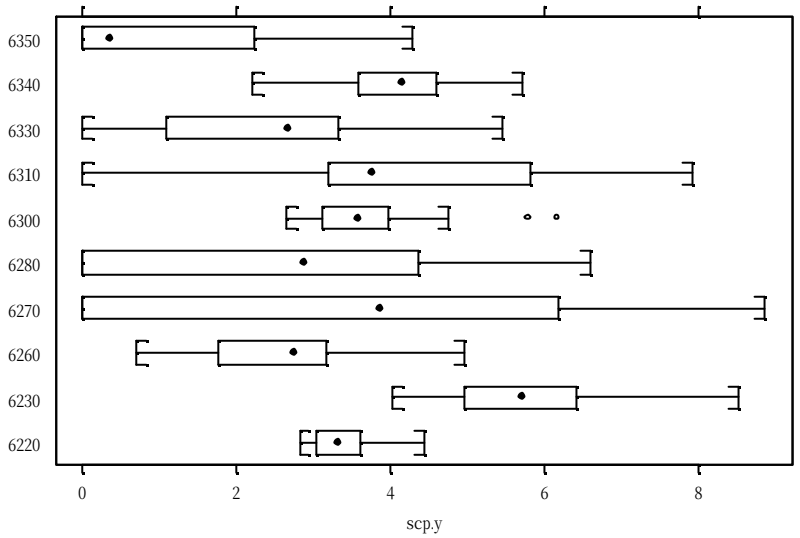


**図 3.25. 対数を取ったホタテ捕獲数データの、階層ごとの箱ひげ図。**

関数 table は、それぞれの階層にいくつの観測値があるかを数える。図 3.25 によると、階層 6300 に外れ値と思われる点が 2 つある：観測番号 165 と 167（92 行目と 94 行目）。

## 3.3 EDA の格子データへの応用

サンプルデータフレーム sids は**格子**（*lattice*）上で採集された空間データである。格子データの定義は第 1 章を参照。データはノースカロライナ州内の各郡で収集されており、1974 年から 1978 年の間に乳幼児突然死症候群（SIDS）によって死亡した乳幼児の数である（Cressie and Chan, 1989）。SIDS データフレームの成分は以下の通りである：

```
> names(sids)
[1] "id"          "easting"     "northing"
[4] "sid"         "birth"       "nwbirths"
[7] "group"       "sid.ft"      "nwbirth.ft"
```

1979 年から 1984 年までの同様のデータが sids2 に収められている。個々の変数の詳細は sids のヘルプファイルを参照。

空間格子を作るには、位置のデータに加えて**近傍**（*neighborhood*）情報が伴っていなければならない。SIDS データの位置は easting と northing に収められている。近傍情報は典型的には近傍行列に収められている。もしこの行列の $i, j$ 成分がゼロでなければ、領域 $i$ と領域 $j$ は近傍どうしである。S+SPATIALSTATS では、近傍情報はクラス"spatial.neighbor"のオブジェクトとして収められる。このオブジェクトは近傍行列を疎な行列として表現したものである。S-PLUS オブジェクト sids.neighbor には SIDS データの近傍情報が含まれている：

```
> sids.neighbor[1:15,]
Total number of spatial units =  100
(Matrix was NOT defined as symmetric)
   row.id  col.id     weights  matrix
 2      1      17  0.04351368       1
 3      1      19  0.04862620       1
 4      1      32  0.10268062       1
 5      1      41  0.20782813       1
 6      1      68  0.11500900       1
 8      2      14  0.17520402       1
 9      2      18  0.27140700       1
10      2      49  0.20882988       1
11      2      97  0.22700297       1
```

```
13      3       5  0.16797229       1
14      3      86  0.25458569       1
15      3      97  0.22660765       1
17      4      62  0.06865403       1
18      4      77  0.15401887       1
19      4      84  0.09565681       1
```

列 row.id と col.id は近傍の対を表わしており、近傍行列において 0 でない成分の行番号と列番号に相当する。例えば、領域 1 の近傍は領域 17、19、32、41、68 である。近傍の対の関係の強さは列 weights で示される。重み付き近傍行列の第 1 行第 17 列成分の値は、0.0435 である。近傍の重みと空間近傍に関する詳細は 5.1 節を参照。

SIDS データでは、Cressie (1993)に従い、郡庁どうしの距離が 30 マイル以内であることを近傍の定義とした。SIDS の格子をプロットするには以下のようにする：

```
> attach(sids)
> plot(easting, northing)
> segments(easting[sids.neighbor$row.id],
+      northing[sids.neighbor$row.id],
+      easting[sids.neighbor$col.id],
+      northing[sids.neighbor$col.id])
> detach("sids")
```

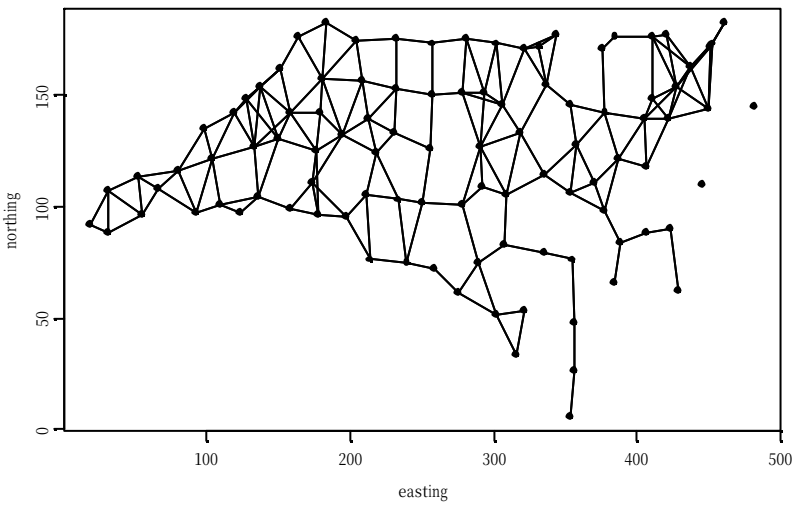


**図 3.26.** SIDS 格子のプロット。

関数 segments は近傍どうしを結ぶ線を引くのに用いた。図 3.26 によれば、近傍を持たない領域が 2 つあり、1 つあるいは 2 つしか近傍を持たない領域もいくつか存在する。

変量 sids$sid は離散量であり、1974 年から 1978 年の期間に各郡で SIDS により死亡した乳幼児数が収められている。ノースカロライナ州における SIDS 死者数の空間的な分布を正確に調べるために、まず各郡の全出生数（births）により補正を行う必要がある。Cressie (1993)に従い、正規化された SIDS 死亡率を生成するには以下のようにする：

```
> sids.w <- 1000*(sids$sid+1)/sids$births
> hist(sids.w)
```

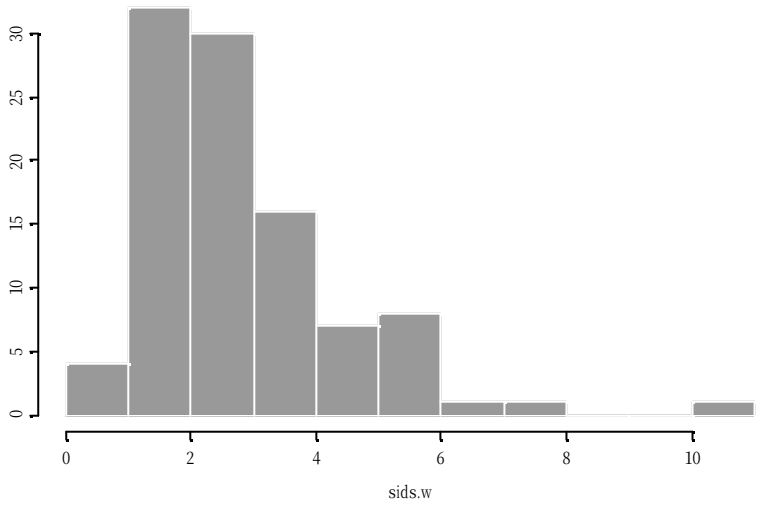


**図 3.27.** SIDS 死亡率のヒストグラム。

全郡の SIDS 死者数に 1 を加えて、SIDS による死者がゼロの郡の中で区別できるようにした。図 3.27 のヒストグラムを見ると、歪んだ分布と大きい外れ値と思われる点が検出できる。

空間上のパターンを見るために、地図上の各郡を死亡率に応じた色で塗り分けてみる。SIDS 死亡率は平均一定のポアソン分布に従うと仮定して確率の地図を作り、分布の裾にある郡を表示させるようにする：

1. SIDS の共通確率 $\hat{p}$ を求める。

```
> attach(sids)
> sids.phat <- sum(sid)/sum(births)
```

2. 各郡の SIDS の期待値 $\hat{l}$ を求める。

```
> sids.lambdahat <- births*sids.phat
```

3. 関数 ppois を使って、実際の SIDS 死亡率に対する累積確率 $d_i$ を計算する。

```
> sids.di <- ppois(sid, sids.lambdahat)
> detach("sids")
```

4. 地図の色を定義する：SIDS 死亡率が異常に高い郡は色番号 1、異常に低い郡は色番号 4。

```
> sids.dcol <- rep(NA,100)
> sids.dcol[sids.di > .95] <- 1
> sids.dcol[sids.di < .05] <- 4
```

5. 結果を地図にする[1]。

```
> library(maps)
Warning messages:
  The functions and datasets in library section maps are
not  supported by StatSci. in: library(maps)
> map("county", "north carolina", fill=T,
+      color=sids.dcol)
> map("county", "north carolina", add=T)
> legend(locator(1), legend
+      c("Prob > .95","Prob < .05"), fill=c(1,4))
```

図 3.28 によると北東部と南部に値の高いクラスタがあり、また中心部には値の小さなクラスタがある可能性がある。この種の地図は探索的技法には有益であるが、ポアソンモデルでは近隣の州との空間的自己相関が考慮されていない。

ポアソン確率がよく当てはまるかどうかに関係なく、計数データは平均に比例する分散を持つことが多く、図 3.28 に間違いを生じさせる場合がある。出生率の高い郡はその SIDS 死者数の年毎の変動が出生率の低い郡の死者数の変動よりも小さくなる傾向にある。この問題は、図 3.27 で見た

[1] **訳注：**S-PLUS for Windows Ver.4.0（日本語版）のライブラリ maps には残念ながらアメリカ合衆国の州境データしか含まれておらず、図 3.28 を描くことはできない。

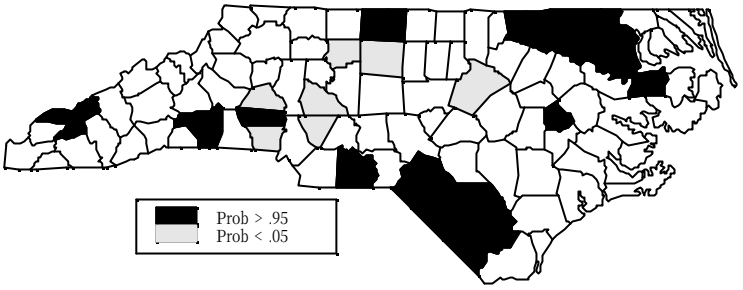


**図 3.28.** SIDS 死亡率の確率地図。確率とはポアソン分布の累積確率。

ような分布のゆがみと同様、データの変換によって解決する。SIDS 死者数の分布に二項分布を仮定すると、平方根をとる変換を行なうことで平均と分散の従属関係と分布の歪みが解消されることが多い。しかし、二項分布モデルに空間依存性が加わると、もっと複雑な変換が必要となる。Cressie and Read(1989)は Freeman-Tukey 平方根変換を推奨している：

$$Y_i = \sqrt{1000}\left(\sqrt{S_i / n_i} + \sqrt{(S_i + 1)/ n_i}\right)$$

ここで $S_i$ は郡 $i$ における SIDS 死亡者数で、$n_i$ は郡 $i$ における出生者数を表わす。この変換を行ったデータが `sids$sid.ft` に収められている。変換後の SIDS 死亡率の茎葉図は以下の通りである：

```
> stem(sids$sid.ft)
N = 100   Median = 2.892998
Quartiles = 2.222682, 3.391945
Decimal point is at the colon

  0 : 9
  1 : 111244
  1 : 567789999
  2 : 0011111222334444
  2 : 55555666677778999999999
  3 : 000111122333333344444444
  3 : 5568999
```

```
  4 : 013344
  4 : 555557
  5 : 2

High: 6.283325
```

変換されたデータの茎葉図はより対称的になっており、外れ値は 4 番目の郡のみとなっている。分散は依然として出生数に依存しているが、Cressie and Read(1989)はこの変換により分散が安定化されることを示している。それゆえ、郡 $i$ における出生者数を $n_i$ とし、郡 $i$ における変換後の SIDS 死亡率を $Y_i$ とすると、$n_i * Y_i$ は分散がほぼ一定になるはずである。この情報は 5.3 節の空間回帰の際に有効である。

3.2.1 節では、グリッド上の地理統計データに対して局所近傍内の外れ値を検出する為に空間ラグプロットを使った（図 3.18 参照）。この手法はグリッド上の（あるいは規則的な）格子データに対しても適用できる。不規則格子については、Cressie(1993)のようにグリッド上に強制的に変換しない限りこの手法は使えない。Haining(1990)では別の方法が提案されている：近傍の値の平均値に対して散布図を描く方法である。

変換された SIDS 死亡率のデータから、その近傍の平均のベクトルを生成するには以下のようにする：

1. 各領域にいくつ近傍があるかを関数 `tabulate` を使って数える。

```
> sidstable <- tabulate(sids.neighbor$row.id)
> sidstable
  [1] 5 4 3 4 3 6 3 4 2 1 6 6 4 7 4 1 4 6 7 2 8 3 3 2 2
 [26] 3 2 0 4 5 2 6 5 5 5 4 3 4 5 4 4 2 5 5 4 5 4 0 7 4
 [51] 4 3 3 4 5 7 4 3 6 4 5 4 4 3 2 2 1 5 2 5 2 7 4 5 2
 [76] 3 2 4 3 5 5 2 3 4 3 4 4 4 5 3 3 5 3 6 4 4 7 5 5 5
```

2. 重みが各領域の `row.id` の数の逆数であるような新たな空間近傍を作る。

```
> sids.nhbr <- sids.neighbor
> sids.nhbr$weights <- 1/sidstable[sids.nhbr$row.id]
```

3. `spatial.multiply` を使って近傍の平均のベクトルを作る。

```
> sids.Ny <- spatial.multiply(sids.nhbr, sids$sid.ft)
```

関数 spatial.multiply は疎な行列の掛け算を行なう；各領域（行）の近傍の重みと、相当する sids$sid.ft の SIDS 死亡率を掛け合わせ、足し合わせることで算術平均を求める。オブジェクト sids.Ny は、各行が近傍の平均値をあらわす（100×1）行列となる。

プロットを行なう前に、（上で示したように）sidstable には 0 が 2 つあることに注意せよ。これは近傍を持たない郡が 2 つ存在することを意味する。これらに相当する sids.Ny の要素をプロットしないようにすべきである：

```
> par(pty="s")
> plot(sids.Ny[-48][-28], sids$sid.ft[-48][-28],
+      xlim=c(1,6))
```

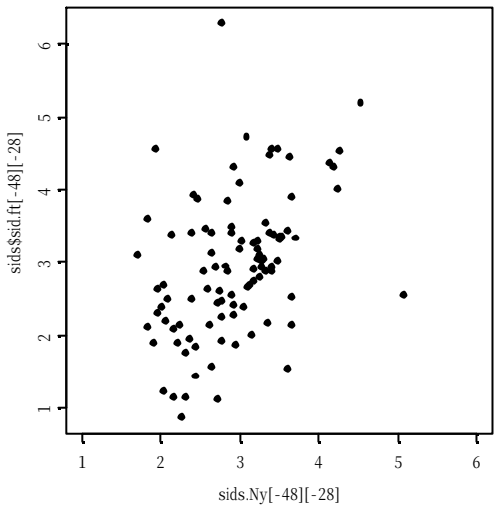


**図 3.29.** 各郡の SIDS 死亡率（変換後）に対して、その近傍の平均値を散布したもの。

外れ値として明らかなのは、郡 4 と郡 4 を含む近傍の平均のみである。この時点で、郡 4 を除外したほうが後の解析にとって妥当であるように思われる。

大域的トレンドの探索をさらに続けて、3.2.2 節でグリッド上にない地理統計データに対して行なったような、$x$ 軸や $y$ 軸に直行する方向のトレンドを見ることにする。重ねて言うが、このモデルは独立性を仮定しているので、このモデルはデータの探索のためだけに使用する。

```
> gam.sids <- gam(sid.ft ~ lo(easting) + lo(northing),
```

```
+      data=sids[-4,])
> par(mfrow=c(2,1))
> plot(gam.sids, residual=T, rug=F)
```

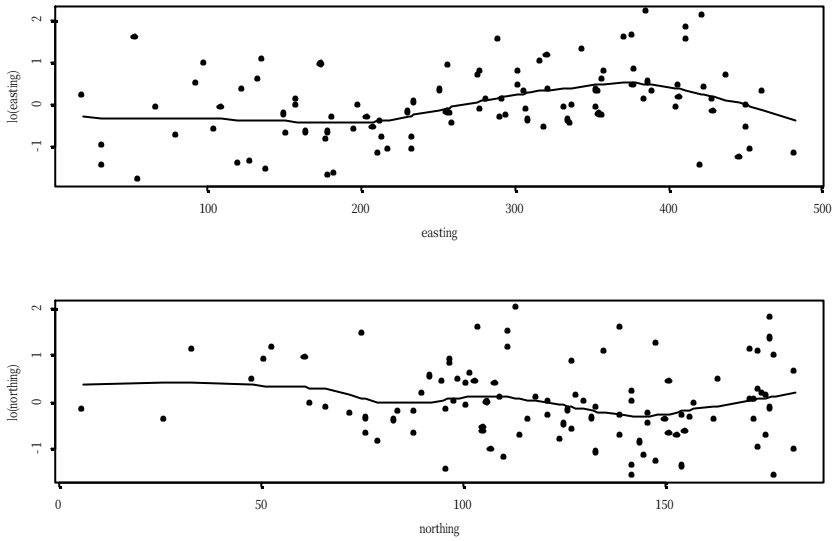


**図 3.30.** SIDS データの、座標軸による平滑化トレンドモデルのプロット。

図 3.30 を見ると、図 3.28 の地図上で見たような一般的なトレンドが現われている。しかし関係は線形ではなく、残差もきわめて大きい。よって `easting` と `northing` は SIDS データのトレンドを線形推定する予測因子としては不十分なのかもしれない。

このデータに対する過去の分析によると、人種が SIDS 死亡数と大きい相関を持つことが示唆されている［（Cressie, 1993, p.550）；（Cressie and Read, 1985）；（Cressie and Chan, 1989）；（Symons et al., 1983）］。1974 年から 1978 年の間の、各郡の有色人種の出生率を Freeman-Tukey 変換したものが変数 `sids$nwbirths.ft` に収められている。この出生率を変換後の SIDS 死亡率に対してプロットし、関係を見てみる：

```
> plot(sids$nwbirths.ft[-4], sids$sid.ft[-4])
> abline(lm(sids$sid.ft[-4] ~ sids$nwbirth.ft[-4]))
```

外れ値である郡 4 は図 3.31 にはプロットされていない。この散布図から、有色人種の出生率と SIDS 死亡率との間には線形に近い関係があることが分かる。関数 `abline` によって書き加えられた直線は、線形最小 2 乗回帰直線であり、関数 `lm` により求められたものである。もし有色人種の出生率と郡の位置の間に相関があれば、有色人種の出生率とをトレンドモ

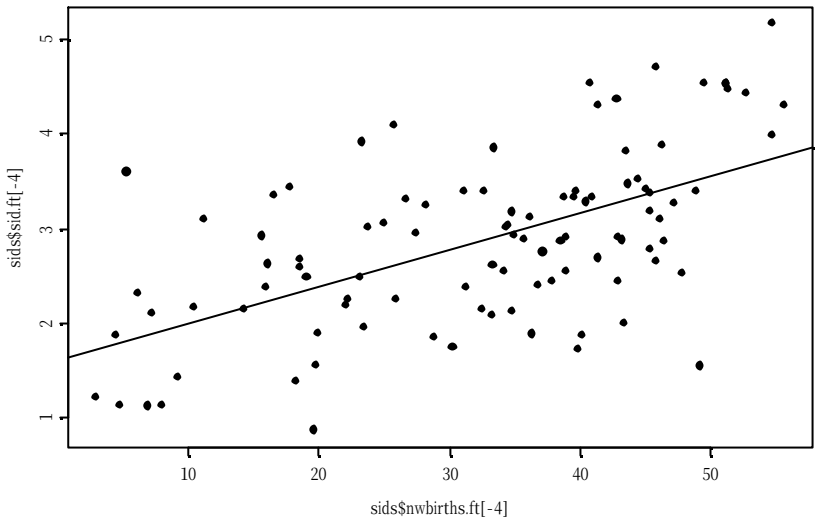


**図 3.31.** 変換後の有色人種出生率の、変換後の SIDS 死亡率に対する散布図。

デルの線形予測因子として用いることで、図 3.30 のような位置によるトレンドを説明できるかもしれない。easting を有色人種の出生率に対してプロットして、正の相関があるかどうかを見てみる：

```
> attach(sids)
> plot(easting[-4], nwbirths.ft[-4], pch = 16)
> abline(lm(nwbirths.ft[-4] ~ easting[-4]))
> detach("sids")
```

図 3.32 によると、有色人種の出生率と東西の位置との間には正の相関がありそうだが、線形関係には見えない。

有色人種の出生率は、ノースキャロライナ州の SIDS 死亡率を線形モデルとして表わす場合の予測因子の 1 つに過ぎない。最終的なモデルを構築する前に、他の様々な変量を考慮する必要があるのは明らかなことである。

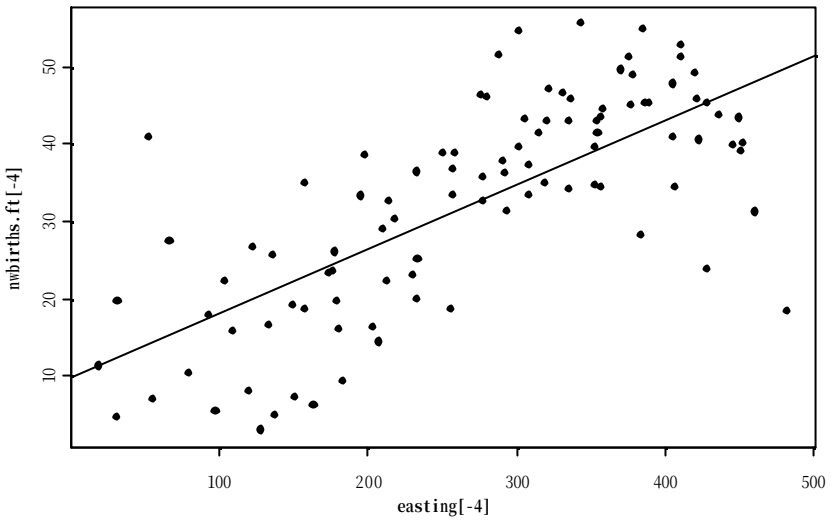


**図 3.32.** 変換後の有色人種の出生率に対する東西の位置の散布図。

## 3.4 EDA の点パターンへの適用

空間点パターンは、空間のある領域内に位置する点、あるいはイベント（何かが発生した位置）の集合である。データの位置あるいは点はランダムに散らばっていることもあれば寄り合う傾向にあることもあるし、規則的に配置されることもある。典型的なデータ解析は、まず**完全ランダム性**（CSR, *complete spatial randomness*）の検定に始まり、ランダム性の欠落をモデリングしようと試みることへと続く。この節では、空間点パターンの探索をいかにして始めるかということについて説明する。空間点パターンの CSR のより正式な確かめ方とモデリングの技法については第 6 章で述べる。

S+SPATIALSTATS では、空間点パターンとして解析される位置のデータはクラス"spp"のオブジェクトとして収められている。点パターンオブジェクトの 2 つの列（典型的には最初の 2 列）は各点の位置であり、それ以外の列はあってもなくてもよい。これらのオブジェクトは以下のように、S-PLUS に取り込んだデータでも、既存の S-PLUS データフレームでも、位置を表わす２本のベクトルでもよい：

```
> pines <- spp(matrix(scan("pine.dat"), byrow=T, ncol=2))
```

```
> bramble.spp <- as.spp(bramble)
> random.spp <- spp(x=runif(100), y=runif(100))
```

例えば、**pine.dat** はテキストファイル（S+SPATIALSTATS に含まれているのではない）であり、$x$ と $y$ のペアがデータとして入っている。bramble は S-PLUS のデータフレームであり、最初の 2 列がデータの位置を表わしている。関数 runif は[0,1]上の一様分布に従う乱数を発生させる。結果として生成された pines、bramble.spp、random.spp の 3 つはクラス"spp"のオブジェクトになる。

S+SPATIALSTATS に含まれるデータフレーム bramble は、9×9 メートルの小区画に新たに生えたキイチゴの茎 359 本の位置のデータである[(Diggle, 1983, p.83), (Hutchings, 1979)]。

位置は、データが単位正方形内におさまるようにスケーリングされている。位置を単にプロットすることからキイチゴデータの探索を開始しよう：

```
> is.spp(bramble.spp)
[1] T
> par(pty="s")
> plot(bramble.spp)
```

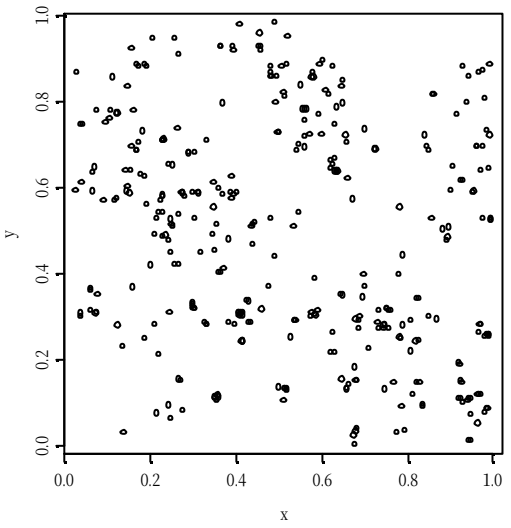


**図 3.33. キイチゴデータの散布図。**

関数 is.spp は bramble.spp が点パターンオブジェクトであるかを確かめるためのものである。空間点パターンをプロットする際は、軸の縮尺

が幾何学的に正確になるようにする。S+SPATIALSTATS の解析ツールは、空間点パターンの境界を長方形に設定している。もし spp の中で特定しなければ、この境界は関数 bbox で求められる正方形となる：

```
> bbox(bramble.spp)
$x:
[1] 0.026 0.026 0.997 0.997
$y:
[1] 0.987 0.001 0.001 0.987
```

このリストはクラス"spp"のオブジェクトの属性 boundary に収められている：

```
> attributes(bramble.spp)$boundary
$x:
[1] 0.026 0.026 0.997 0.997
$y:
[1] 0.987 0.001 0.001 0.987
```

キイチゴデータは、単位正方形にスケーリングされた正方区画の中で採集されたデータである。境界は以下のようにすると単位正方形に再定義できる：

```
> bramble.spp <- spp(bramble, boundary=bbox(x=c(0,1),
+      y=c(0,1)))
```

この他、境界は凸多角形のこともあるだろう。凸多角形は S+SPATIALSTATS では各頂点の $x$、$y$ 座標のリストで表現される。例えば、すべての点を内側に含むような最小の凸多角形、凸包を見つけるには、以下のように関数 chull を用いる：

```
> hull <- chull(bramble.spp)
> hull
 [1]   1  72 175 119 165 305 308 117 290 283 235
[12] 48   3  32
> bramb.poly <- list(x=bramble$x[hull],
+                        y=bramble$y[hull])
```

この多角形は関数 spp の引数 boundary として使うことができる。キイチゴデータを比較のために 2 種類の境界とともにプロットするには以下のようにする：

```
> plot(bramble.spp, boundary=T)
> polygon(bramb.poly, density=0)
```

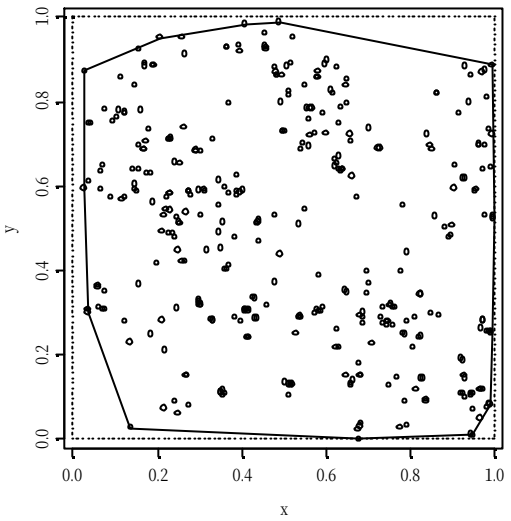


**図 3.34.** キイチゴデータの散布図をその境界と凸包とともに示したもの。

図 3.34 の点線は単位正方形の境界を示し、実線は関数 `polygon` によって描かれた凸包である。境界を少し拡げて、すべての点が完全に内側に含まれるようにしたければ、関数 `poly.expand` を使うとよい。

次に、図 3.34 の凸包に対して全体の**強度**（*intensity*）、つまり単位面積辺りの点の数を推定することができる。単にすべての点の数を、関数 `poly.area` を使って求めた多角形の面積で割ればよいのである：

```
> poly.area(bramb.poly)
[1] 0.8789835
> 359/.879
[1] 408.4187
```

この凸包が境界として正しければ、強度の推定値は 408.4 となる。この強度の推定は定常過程を仮定している；つまりこの境界内で強度は一定であるということである。

⟹ **ヒント：** 強度は S+SPATIALSTATS 関数 `intensity` を使って推定することもできる。この関数は第 6 章で紹介する。

## 3.5 Hexagonal Binning

Hexagonal binning とはデータをまとめる手法であり、データ数の多いデータの空間構造を明らかにするためによく用いられる。散布図をより大きな単位にまとめ、データ数を減らし、かつデータの密度は維持する方法と解釈することができる。グループまたは bin は、その密度に応じて色や大きさを変えて描く hexagonal mosaic maps を作るのに使われる。図 3.6、3.7 の濃淡画像や等高線図や鳥瞰図のように、画像処理の応用に対しては矩形または正方格子が使われることが多い。しかし、6 角形のほうが視覚的にも表現の正確さに関しても好ましい（Carr et al., 1992）。Hexagonal binning は地理統計データを空間回帰モデリングするために格子に変換するときにも使うことができる。

データフレーム quakes.bay には、サンフランシスコの湾岸地域で 1962 年から 1981 年の間に発生した地震の震源地の位置情報が含まれている。S+SPATIALSTATS では、hexagonal binning はクラス"hexbin"のオブジェクトに対して行われる。関数 hexbin を用いて地震データを hexbin オブジェクトにするには以下のようにする：

```
> quakes.bin <- hexbin(quakes.bay$longitude,
+     quakes.bay$latitude)
> summary(quakes.bin)
Call:
hexbin(x=quakes.bay$longitude, y=quakes.bay$latitude)
Total Grid Extent:  36 by 31
       cell             count              xcenter
Min.    :  17.0   Min.    :   1.000   Min.    :-123.3
1st Qu. : 239.0   1st Qu. :   1.000   1st Qu. :-122.0
Median  : 419.0   Median  :   3.000   Median  :-121.6
Mean    : 467.9   Mean    :   7.505   Mean    :-121.5
3rd Qu. : 696.0   3rd Qu. :   5.000   3rd Qu. :-121.0
Max.    :1091.0   Max.    : 144.000   Max.    :-119.8

       ycenter
Min.    :36.01
1st Qu. :36.51
```

```
Median   :36.94
Mean     :37.06
3rd Qu.  :37.59
Max.     :38.50
```

関数 summary は hexbin オブジェクトの 4 成分とそれらの分布を表示している。cell で識別される 6 角形は、count を値に持ち、(xcenter, ycenter)を中心に持つ。hexbin はデフォルトでは $x$ の範囲をおおよそ 30 個の正 6 角形に分割するように設定されている。最も有効な bin の大きさはデータ数に依存するので、繰り返しによって選ばれる。hexagonal bins をプロットするには以下のようにする：

```
> trellis.device(color=F)
> at.quakes <- c(0,10,20,30,40,50,150)
> plot(quakes.bin, border=T, col.regions=80:15,
+      at=at.quakes)
```

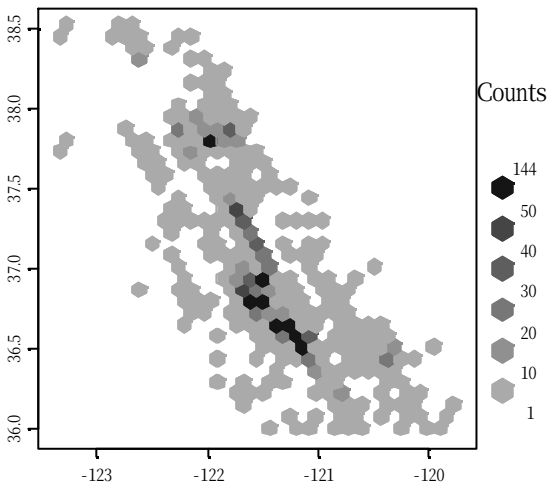


**図 3.35.** 1962 年から 1981 年の間にサンフランシスコ湾岸地域で発生した地震の hexagonal bins のプロット。

Trellis グラフィックデバイスは、hexagonal binning に最も適したカラー画像または濃淡画像を作成する。関数 plot.hexbin はデフォルトでは hexagonal bins をモザイクにしてプロットする。モザイクの一つ一つは大きさの等しい 6 角形で、count の値に応じて分類された色がつけられている。この分類はデフォルトではすべて等間隔に区切られている。しかし、quakes.bin$count の分布は（上の summary の出力で見たように）

歪んでいるため、at.quakes のように定義した分類を使った。図 3.35 によると、地震はサンアンドレアス断層に沿った尾根状の部分で頻繁に発生している。

図 3.35 で用いたデフォルトの濃淡画像の他に、count の値によって 6 角形の大きさを変化させる 4 スタイルのプロット方法が利用できる。地震データの hexbin オブジェクトを、6 角形の大きさを変化させてプロットするには以下のようにする：

```
> plot(quakes.bin, style="centroids", cuts=6)
```

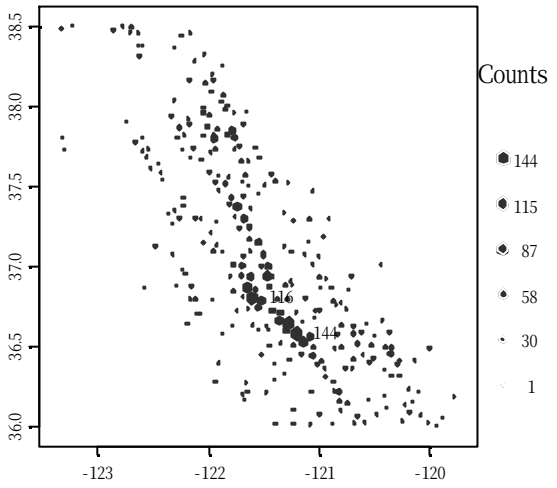


**図 3.36.** サンフランシスコ湾岸地域の地震データの hexagonal bins を、"centroid"スタイルでプロットしたもの。

図 3.36 の"centroid"スタイルは count に応じて 6 角形の大きさを設定し、それらを中心が xcenter と ycenter によって定められるようにプロットするものである。引数 cuts = 6 によって 6 角形の大きさを 6 段階に変化させるようにしている。他に、2 種類の入れ子スタイル（"nested.lattice"と"nested.centroids"、ここでは例示しない）のプロットがあるが、カラースクリーンにプロットした場合に視覚的な深さが得られるような図を描く。

図 3.36 には、より詳しい調査が必要とされるような大きな bin が幾つかある。hexagonal bin プロット上の点を対話的に識別するのに、関数 identify を使うことができる。2 つの大きな bin を識別するには以下の

ようにする：

```
> quake.par <- plot(quakes.bin, style="centroids",
+ cuts=6)
> oldpar <- par(quake.par)
> identify(quakes.bin, use.par = quake.par, offset=1)
[1] 114  79
> par(oldpar)
```

最初に、hexagonal bin のプロットに使用した作図パラメータを保存する必要がある。関数 identify を入力し、グラフィックウィンドウ上の興味ある点を左クリックする。クリックした場所から一番近いセルの count の値がグラフィックウィンドウに表示される。表示される値を読みやすくするために、省略可能な引数 offset を使用した（図 3.36）。2 つの点の認識が終わったら、グラフィックウインドウ内でマウスの中央ボタンもしくは右ボタンをクリックする。コマンドラインには、上のように識別した点のインデックスが表示される。その後、関数 par を使って作図パラメータを元に戻す。

S+SPATIALSTATS の関数 rayplot は、各位置における興味ある変量の大きさを、方向を持った線で表現する。データ数が少ない場合は各位置に線や他の記号を描くこともできるが、データ数が多い場合は先に hexbin で bin してからの方が値の大きさやトレンドを見やすくできる。以下の例では S-PLUS のデータセット ozone を使っている：

1. オゾンデータから、x 軸方向を 8 つの bin に分けた hexbin オブジェクトをつくる。

```
> ozone.bin <- hexbin(ozone.xy$x, ozone.xy$y, xbin=8)
```

2. 関数 xy2cell を使って元データの(x,y)の各組を 6 角形セルに対応させる。

```
> ozone.cells <- xy2cell(ozone.xy$x, ozone.xy$y,
+     xbins=8)
```

3. 関数 tapply を使って各セルの中央値を求め、これらの値を rayplot の角度にする。

```
> ozone.angle <- tapply(ozone.median,ozone.cells,
+     median)
> library(maps)
```

```
Warning messages:
 The functions and datasets in library section maps are
not supported by MathSoft. in: library(maps)
> map(region=c("new york","new jersey","conn","mass"),
+      lty=2)
> rayplot(ozone.bin$xcenter,ozone.bin$ycenter,
+     ozone.angle)
```

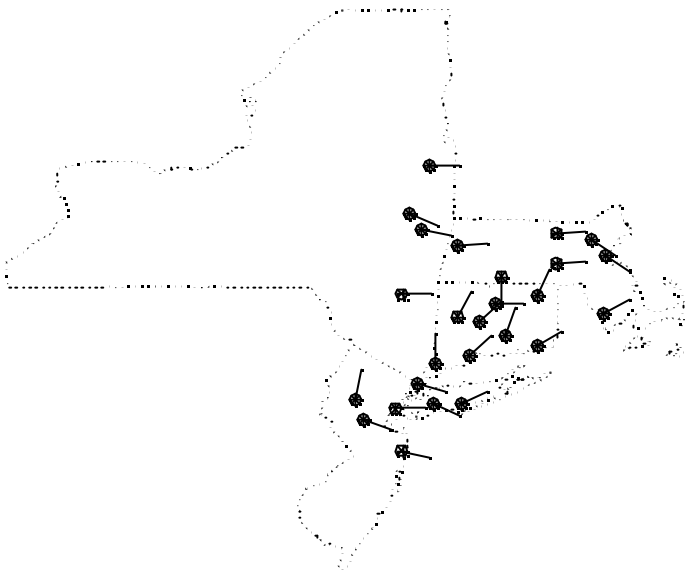

**図 3.37．オゾン放出量の中央値の** rayplot。

**図** 3.37 **は各** hexagonal bin **内のサイトのオゾン放出量の中央値を表わしている。**ray **は各** bin **の中央に位置し、各中央値は**$-\pi/2$**（最小値）から**$\pi/2$**（最大値）までの角度にスケーリングされている。オゾンの放出量が最も多いのはコネティカット州であるように思われる。**rayplot **には他にも信頼区間や第** 2 **の変量を加えたりすることもできる。また、線の長さや太さ、**8 **角形の大きさなども変えることができる。詳しくは** rayplot **のヘルプファイルを参照。**

# 4 地理統計データの解析

この章では、地理統計データの解析に使うことのできる S-PLUS または S+SPATIALSTATS の関数を紹介する。地理統計データは確率場データとも呼ばれ、固定された各地点で得られた測定値で構成される。地理統計データの正確な記述については第 1 章を参照。この章では特に、バリオグラム解析やクリギングに関連した手法について議論する。バリオグラムの推定やクリギングはそもそも地理統計的手法を鉱山学に応用したものとして導入された。近年、これらの手法は気象学、森林学、農業、地図作成、気候学、水産学など多くの学問分野に応用されている。

この章では、以下のことについて学習する：

- バリオグラムの推定（4.1 節）
- 理論バリオグラムモデルの当てはめ（4.2 節）
- 通常クリギングと普遍クリギングを行う（4.3 節）
- 地理統計データのシミュレーション（4.4 節）

## 4.1 バリオグラムの推定

地理統計データは局所変動性を持つのが典型的であり、これを空間自己相関としてモデリングしたり、推定の手続きに組み込んだりすることもある。バリオグラムは、サンプルデータが距離と方向にどのように関係しているか、ということを通して空間的相関の度合いを測るものである。一般に、近く隣り合ったデータどうしは、遠く離れたものどうしよりも似通った値になる傾向にある。他にも空間的相関を、距離をもとに測る尺度として、コレログラム関数やコバリオグラム関数がある。

この節では、経験バリオグラムを生成するのに用いるツールを見ながら、

地理統計データの探索的データ解析を引き続き行う。バリオグラムの推定は探索的であり、そのプロセスは多段階であることが多い；バリオグラムモデルは、最終的にはそのデータに影響を与えている（生成している）プロセスについての知識と利用可能なツールのカスタマイズによって構築される。

S+SPATIALSTATS はバリオグラムの推定に必要な様々な関数を提供する。以下の小節では、経験バリオグラムの計算、バリオグラム雲の作成、トレンドの判定と除去、異方性の探索と修正などを行うツールについて説明する。バリオグラムの解析は以下のトピックの順番で行う必要はない；最終的な結論は、おそらく可能なツールを組み合わせて繰り返し解析を行った結果に基づいて出されることになるだろう。バリオグラムの解析をどこから始めるか、またどの順番で行うかということは、解析者がどの程度データについて知識があるかに依存する。例えば、解析を行おうとするデータにはトレンドがあることを解析者が知っていれば、そのトレンドをモデリングすることから解析が始まるだろう（4.1.3 節参照）。

### 4.1.1 経験バリオグラム

経験バリオグラムは、データが距離とどう関連しているか（どういう相関を持つか）を記述するものである。セミバリオグラム関数 $\gamma(h)$は、距離 $h$ だけ離れた 2 点間の差の平均の 2 分の 1 として Matheron(1963)が定義したのが最初である。セミバリオグラムは

$$g(h) = \frac{1}{2|N(h)|}\sum_{N(h)}(z_i - z_j)^2$$

と書ける。ここで $N(h)$はユークリッド距離 $i - j = h$ となるすべてのペアの集合で、$|N(h)|$ は $N(h)$の要素数、$z_i$ と $z_j$ はそれぞれ位置 $i$ と $j$ でのデータ値である。この式での距離 $h$ は、大きさのみの尺度である。ときには距離の他に方向を考慮したほうが好ましい場合もある。そのような場合、$h$ は大きさと方向を持ったベクトル $\boldsymbol{h}$ で表現される。

**注意：**セミバリオグラムとバリオグラムはよく混同して使われる。定義上は $\gamma(h)$がセミバリオグラムで、$2\gamma(h)$がバリオグラムなのであるが、便宜上、本マニュアルでは $\gamma(h)$をバリオグラムと呼ぶことにする。

バリオグラム解析の最終目的は、データの根底にある確率過程の自己相関構造を最も良く推定するバリオグラムを構築することである。ほとんどの

バリオグラムは、いくつかのパラメータによって定義される；**ナゲット効果**（*nugget effect*）、**シル**（*sill*）、**レンジ**（*range*）である。これらのパラメータを一般的なバリオグラムに合わせて描いたものが図 4.1 である。また、これらは以下のように定義される：

- **ナゲット効果** — 微視規模変動あるいは測定誤差を表わす。経験バリオラム $\gamma(h)$の、$h = 0$ における値で推定する。
- **シル** — $\lim_{h\to\infty} g(h)$ で、確率場の分散を表わす。
- **レンジ** — 自己相関が無くなる距離（もしあれば）。

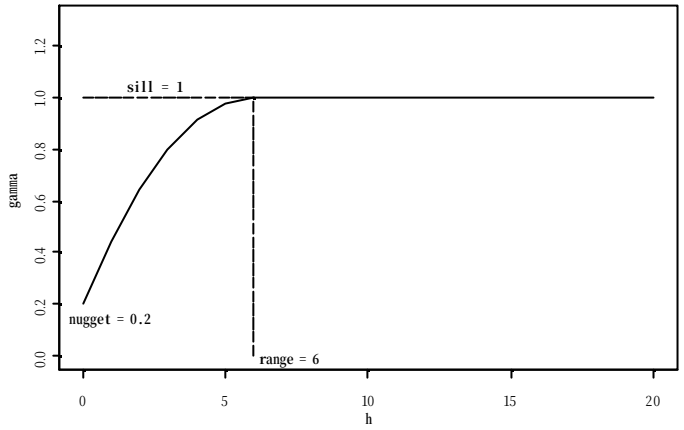


**図 4.1. ナゲット効果、シル、レンジ**の各パラメータを描き込んだ一般的なバリオグラム。

バリオグラムを構築するには、以下のことを考慮する必要がある：

- $h$ のラグ増分（*lag increment*）の適切な取り方；
- ラグ増分の範囲（*tolerance*）；
- バリオグラムを求めるラグの数（*number of lags*）

**ラグ増分**とは、バリオグラムが計算される距離のことである。**範囲**はラグ増分の距離の幅を設定し、空間的に一様でないデータに対処する。**ラグの数**は、ラグ増分の大きさに関連して、全体でどういう範囲の距離でバリオグラムを計算するかということを設定する。

ラグ増分とラグの数を設定する際に考慮すべき 2 つの実用的な規則

（Journel and Huijbregts, 1978）がある：

- 経験バリオグラムは、ペアの数が 30 以上になる距離 $h$ のみで計算すべきである。
- 経験バリオグラムの信頼できる距離は $h < D/2$ である。ここで、$D$ はデータの 2 点間の距離のうちで最大になる距離である。

### 全方向バリオグラム（*Omnidirectional Variograms*）

バリオグラム解析の手始めとして、全方向バリオグラムを求めて見るのも 1 つの方法である。全方向バリオグラムは、方向を考慮せずに計算されるバリオグラムである；1 つのバリオグラム値を求める際にすべての方角の値を用いるのである。石炭埋蔵量のデータの全方向バリオグラムを求めるには、S+SPATIALSTATS 関数 variogram を用いる：

```
> coal.var1 <- variogram(coal ~ loc(x,y), data=coal.ash)
```

関数 variogram は空間位置に対する反応変量を定義した式を必要とする。関数 loc は位置変量を統合して予測因子にするために使用した。この処理は、通常の S-PLUS でモデリングする際の予測変量と区別することにもなる。関数 variogram にはバリオグラムをカスタマイズするための様々な引数がある。例えば lag、nlag、tol.lag、maxdist、minpairs などである。デフォルトでは、variogram は maxdist が上の信頼できる距離になるような全方向バリオグラムを求める。ラグの数 nlag は 20 にしてある。ラグ増分 lag は自動的に maxdist/nlag に設定される。ラグの許容範囲 tol.lag のデフォルト値は lag/2 である。

関数 variogram はクラス"variogram"のオブジェクトを返す。coal.var1 の成分は distance、$\gamma(h)$（gamma）、ペアの数（np）、azimuth[1]である：

```
> coal.var1
     distance      gamma    np  azimuth
  1  1.201634   1.202911   719        0
```

[1] **訳注：**「方角」の意味。

```
 2   2.000000   1.172307   331        0
 3   2.236068   1.321759   644        0
 4   3.036036   1.314383  1170        0
 5   3.605551   1.297816   545        0
 6   4.234139   1.398687  1518        0
 7   5.039712   1.531598  1142        0
 8   5.472889   1.537340   638        0
 9   6.063483   1.536710  1382        0
10   6.552482   1.624369   719        0
11   7.147503   1.478895  1307        0
12   7.894487   1.491406  1269        0
13   8.459312   1.654917   882        0
14   9.094676   1.714728  1012        0
15   9.624306   1.804034   685        0
16  10.174499   1.656996   995        0
17  10.843929   1.705829   682        0
18  11.400797   1.880180   860        0
```

各距離は、その距離区間に入った点のペアの平均距離である。方角は北からの時計周りの角度（°）で、バリオグラムを計算する角度を定義している。全方向バリオグラムは、デフォルトでは`azimuth=0`をもとにしている[2]。`variogram`が呼び出された場合の要約を見てみる：

```
> summary(coal.var1)
Call:
variogram.formula(formula = coal ~ loc(x, y),
                  data = coal.ash)
       lag  nlag    maxdist
 0.6041523    20   12.08305
     distance          gamma               np
Min.    : 1.202   Min.    : 1.172   Min.    :  331.0
1st Qu. : 3.763   1st Qu. : 1.341   1st Qu. :  682.8
Median  : 6.308   Median  : 1.534   Median  :  871.0
Mean    : 6.338   Mean    : 1.518   Mean    :  916.7
```

[2] **訳注：**関数`variogram`は角度に対してデフォルトで引数`azimuth=0`（方向：北）、`tol.azimuth=90`（角度に対する許容範囲±90°）を指定する。つまり、全方向である。

```
3rd Qu.: 8.936   3rd Qu.: 1.656   3rd Qu. : 1163.0
Max.    :11.400   Max.    : 1.880   Max.    : 1518.0
     azimuth
      0:18
```

要約にはvariogramの呼び出し、いくつかの引数に対して計算されたデフォルト値、返された値に対する記述統計量が含まれる。

石炭データの全方向バリオグラムを図 4.2 のようにプロットするには、S-PLUS関数plotを用いて以下のように行う：

```
> trellis.device()
> plot(coal.var1)
```

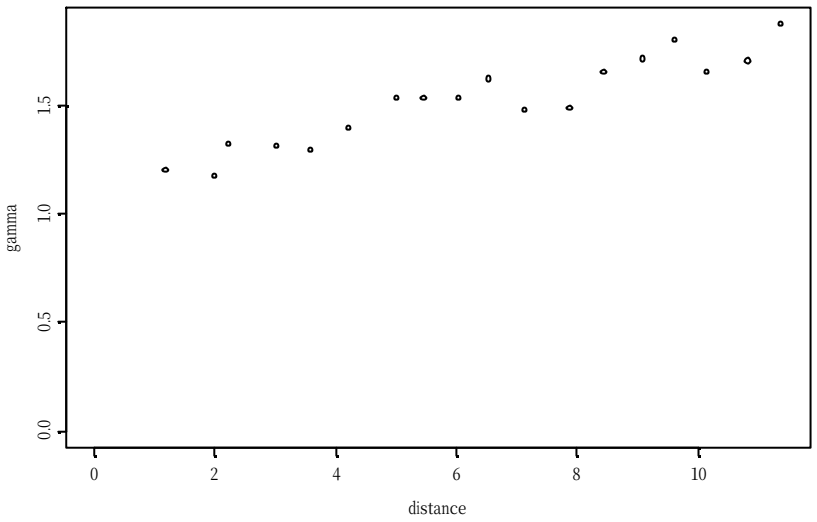


**図 4.2.** 石炭データの全方向経験バリオグラム。

クラス"variogram"のオブジェクトに対して関数plotが使われると、関数plot.variogramによる作図が行われる。単一のバリオグラムに対しては、距離に対するガンマの値の標準的なプロットが行われる。軸は必ず(0,0)が含まれるように設定される。ユーザが引数xlimやylimを設定することもできる。

石炭データの全方向バリオグラムは一般に増加傾向にある。これは大域的なトレンドが存在すること、もしくは根底にある確率過程が非定常であることを示すものかもしれない。3.2.1節で石炭データに対して行ったEDAの結果、東西方向には明らかなトレンドが見られたことを思い出そう。石

炭データにバリオグラムモデルを導入する前に、より深い解析が必要である。

S+SPATIALSTATS は関数 covariogram と関数 correlogram も提供している。コバリオグラムは以下のように定義される：

$$\mathrm{cov}(Z(i+h), Z(i)) = C(h), \quad \text{for all } i, i+h \in D$$

コレログラム$\rho(h)$は共分散の比で、以下のようにして求められる。

$$r(h) = \frac{C(h)}{C(0)} = 1 - \frac{g(h)}{C(0)}$$

ここで$C(h)$はユークリッド距離が$h$離れた2点どうしの共分散(コバリオグラム）で、$C(0)$は確率場の有限分散、$\gamma(h)$はバリオグラムに相当する。これらの定義は等方的な場合のものであり、$h$はスカラーである；これらは、$\boldsymbol{h}$が大きさと方向を持ったベクトルである場合に拡張することができる。

石炭データの経験コバリオグラムと経験コレログラムを求めて図4.3のようにプロットするには以下のようにする：

```
# プロットウィンドウを1×2に設定する
> par(mfrow=c(1,2))
> coal.cov1 < - covariogram(coal~loc(x,y),data=coal.ash)
> plot(coal.cov1)
> coal.cor1 < - correlogram(coal~loc(x,y),data= coal.ash)
> plot(coal.cor1)
```

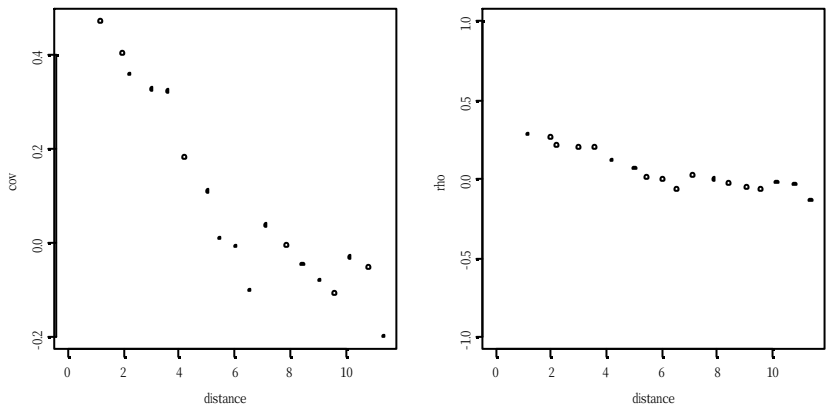


**図 4.3.** 石炭データの全方向経験コバリオグラム（左）とコレログラム（右）

関数 covariogram はクラス"covariogram"のオブジェクトを返し、関数 correlogram はクラス"correlogram"のオブジェクトを返す。これらの関数から返されたオブジェクトは、関数 variogram が返すオブジェクトと類似している。距離、ペアの数（np）、方角などは同じ量である。各距離区間の $cov(h)$あるいは $\gamma(h)$が返される。"variogram"のときと同じく、これらのオブジェクトが関数 plot に使われると、関数 plot.covariogram または関数 plot.correlogram による作図が行われる。コバリオグラムの y 軸の下限はデフォルトで、0 か $min(\mathrm{cov})$の小さいほうである。コレログラムの $y$ 軸の範囲はデフォルトで(-1,1)である。$x$ 軸の下限はどちらの関数でも 0 である。

### 一方向バリオグラム（*Directional Variogram*）

関数 variogram は**一方向バリオグラム**を求めるために使うこともできる。この場合、$\gamma(h)$は大きさと方向を持った $\boldsymbol{h}$ の関数である。一方向バリオグラムを生成するには、引数 azimuth を求めたい方向（北からの相対角度）に設定すればよい。いくつかの方角をベクトルで指定すれば、多方向のバリオグラムを求めることができる。石炭データの一方向バリオグラムをいくつかの方角に対して求め、プロットするには以下のようにする：

```
> az <- c(0, 22.5, 45, 67.5, 90, 112.5)
> coal.var2 <- variogram(coal ~ loc(x, y), data = coal.ash,
    azimuth = az, tol.azimuth = 11.25)
> plot(coal.var2)
```

多方向のバリオグラムに対して、メソッド plot.variogram は、Trellis グラフィックスの関数 xyplot を使ってマルチパネル表示を行う。各パネルには特定の方角のバリオグラムが距離に対してプロットされている。両軸が点 (0,0) を含む範囲に設定されている。この作図は、trellis.device によって起動されるデバイスによって最も良く表示される。

求められた一方向バリオグラムを azimuth に応じてプロットしたものが図 4.4 である。引数 tol.azimuth を 11.25 に設定することで、各方角±11.25 ° の範囲に入る点のペアをもとにバリオグラムを計算している。

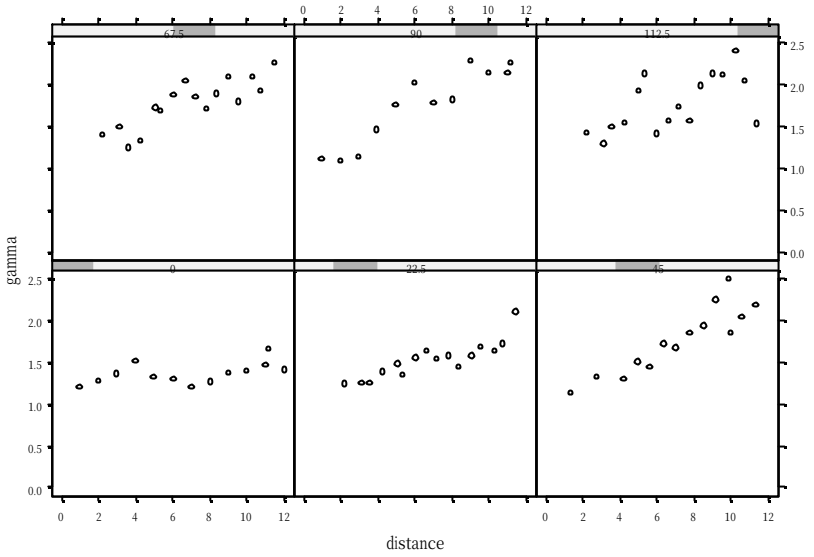


**図 4.4.** 石炭データの一方向経験バリオグラム。

図 4.4 によると、南北（az=0）方向と東西（az=90）方向のバリオグラムには違いが見られる。南北方向は基本的に平坦であり、自己相関がわずか、もしくは無いことを示している。他の方向のバリオグラムは一般的に増加傾向にあり、トレンドと異方性の両方もしくは片方が存在するか、またはプロセスが非定常であることによって引き起こされた可能性がある。トレンドと異方性の検出と修正に関する詳細は 4.1.3 節、4.1.4 節を参照。

### バリオグラムの頑健推定

これまでのバリオグラムは、Matheron(1963)による古典的な公式によって求められるものだった。関数variogramはオプションとして、Cressie and Hawkins(1980)によって開発されたバリオグラムの頑健推定量を計算することができる。頑健推定は、差の絶対値の平方根の 4 乗に基づくもので、以下の式で与えられる：

$$\bar{g}(h) = \frac{\left\{ \frac{1}{2\left|N(h)\right|} \sum_{N(h)} \left| z_i - z_j \right|^{1/2} \right\}^4}{0.457 + 0.494 / \left|N(h)\right|}$$

ここで $N(h)$はユークリッド距離 $i - j = h$ となるすべてのペアの集合で、$|N(h)|$ は $N(h)$の要素数、$z_i$ と $z_j$ はそれぞれ位置 $i$ と $j$ でのデータ値である。

頑健推定量の利点は、特定の観測点をデータセットから除外しなくても外れ値の影響を少なくできる点である。頑健推定を行うには、引数`method`に`"robust"`を与えればよい：

```
> coal.var3 <- variogram(coal ~ loc(x, y), data = coal.ash,
      azimuth = az, tol.azimuth = 11.25, method = "robust")
> plot(coal.var3)
```

頑健推定量を使って計算した一方向バリオグラムが図 4.5 である。頑健推定量は、従来の推定量によって求めたバリオグラム（図 4.4）に見られた変動をいくらかスムースにしたように見える。

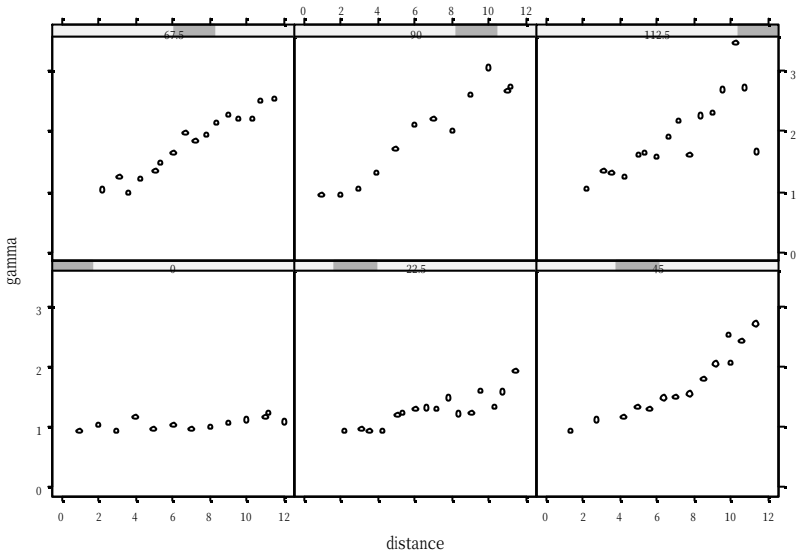


**図 4.5.** 頑健推定を行った、石炭データの一方向経験バリオグラム。

### 4.1.2 バリオグラム雲

**バリオグラム雲**（*variogram cloud*）は箱ひげ図と同様に外れ値だと思われる点やトレンドを検出したり、距離が離れるにしたがって値の変動はどうなるかといったことを評価するのに用いることができる診断ツールである。距離が近いにも関わらず非類似性が高い（$\gamma$ の値が大きい）場合は、変則的な値があるか、または領域内で値が均質でないと判断できる。

バリオグラム雲は、すべての可能な距離 $h$ だけ離れたすべてのペアの分散の分布である。S+SPATIALSTATS の関数 `variogram.cloud` は、分散の算出に使う関数を設定することができる；最もよく使われるのは差の 2 乗または平方根である。差の 2 乗の雲（デフォルト）は古典的なバリオグ

ラム推定量の元になる分布を生成し、$h$ に対して$\left(Z_{i+h}-Z_i\right)^2/2$をプロットする。差の平方根の雲は $h$ に対する$\sqrt{\left(\left|Z_{i+h}-Z_i\right|\right)}/2$をプロットする。

ホタテデータに対してデフォルトの全方向バリオグラム雲を求め、結果をプロットするには以下のようにする：

```
> scallops.vcloud1 <- variogram.cloud(log(tcatch+1)
+      ~ loc(lat,long), data=scallops)
> # プロットウィンドウを1×1に戻す
> par(mfrow = c(1,1))
> # プロット領域を最大にする
> par(pty="m")
> plot(scallops.vcloud1)
```

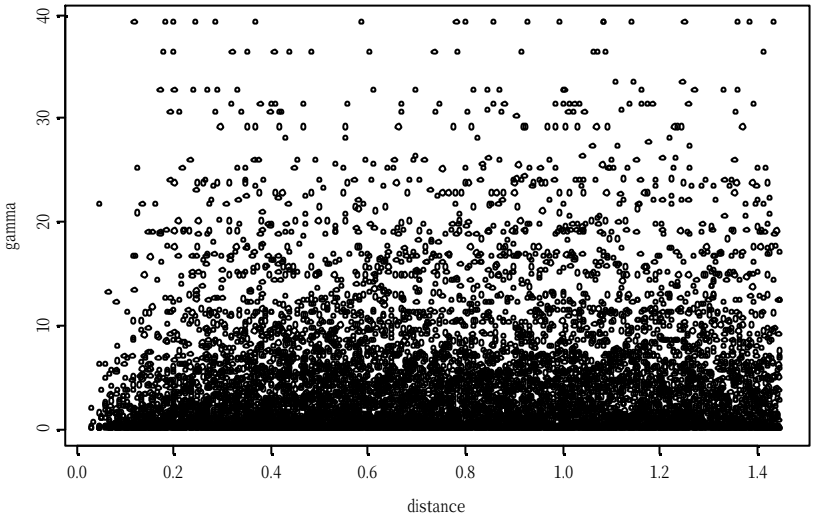


**図 4.6.** ホタテデータに対する差の 2 乗のバリオグラム雲。

差の 2 乗のバリオグラム雲をプロットしたものが図 4.6 である。極度に密集しているバリオグラム雲は理解しづらいかもしれない。分散を計算する最大距離を減らすことによって出力結果の密度を下げることができる。省略可能な引数 maxdist はデフォルトでは信頼できる距離（ホタテデータでは 1.5）である。以下のようにして、見やすい図になるまで maxdist を減少させる：

```
> par(mfrow=c(2,2))
> scallops.vcloud3 <- variogram.cloud(log(tcatch+1)
+      ~ loc(lat,long), data=scallops, maxdist=.5)
```

```
> plot(scallops.vcloud3)
> scallops.vcloud4 <- update(scallops.vcloud3,
+ maxdist=.25)
> plot(scallops.vcloud4)
> scallops.vcloud5 <- update(scallops.vcloud3,
+     maxdist=.125)
> plot(scallops.vcloud5)
> scallops.vcloud6 <- update(scallops.vcloud3,
+     maxdist=.0625)
> plot(scallops.vcloud6)
```

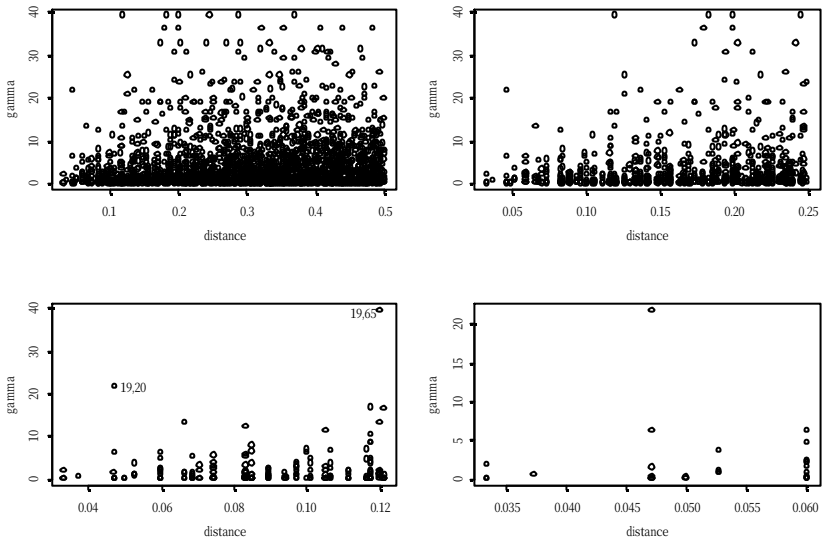


**図 4.7. 最大距離を変化させたときのホタテデータのバリオグラム雲。**

関数 `update` を図 4.7 の一連のバリオグラム雲を生成するのに用いた；`update` は最初の `variogram.cloud` の呼び出しを修正して、`maxdist` を新しい値に置き換える。`update` は `variogram` の呼び出しを修正するときにも使うことができる。

興味を持ったペアは、`identify` を使って対話的に識別することができる。図 4.7 の異質な点のペア(19,20)と(19,65)は以下のようにして求めた：

```
> identify(scallops.vcloud5)
```

`identify` を実行すると、マウスの左ボタンを興味ある点の上でクリック

することで何回でも点を識別できる。終了するには、グラフィックウィンドウ内でマウスの中央ボタン（Windows では右ボタン）をクリックするとよい。デフォルトでは、識別された点のペアがグラフィックスに書き込まれ、S-PLUS コマンドウィンドウにリストアップされる。それらは以下のようにすると S-PLUS オブジェクトとして保存される：

```
> scall.prs <- identify(scallops.vcloud5)
```

**注意：**この例では、関数 `identify` はオブジェクト `scallops.vcloud6` をプロットする前に実行されている。

要約を見ながら外れ値であると思われる点を探し出すには、図 4.8 のように、バリオグラム雲の箱ひげ図を描いてみるとよい：

```
> boxplot(scallops.vcloud1)
```

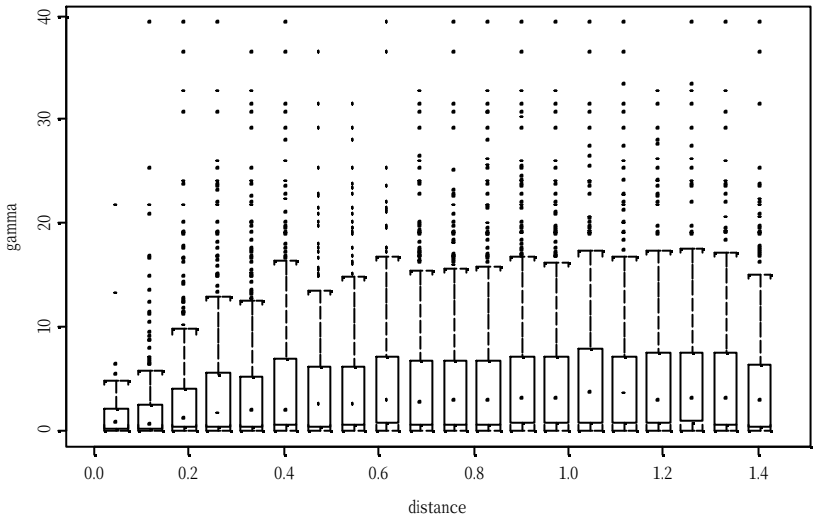


**図 4.8.** ホタテデータに対する差の 2 乗のバリオグラム雲の箱ひげ図。

デフォルトでは、データは 20 の bin に分割される。多くの点がヒゲの外に出ており、これらが外れ値であることが示されている。しかし、これらが外れ値と判定されたのは、バリオグラム雲の分布が歪んでいるためで、観測値が異常なのではないと思われる。この事を確認するために、以下のように差の平方根のバリオグラム雲に対して箱ひげ図を描いてみる：

```
> scallops.vcloud2 <- variogram.cloud(log(tcatch+1)
+       ~ loc(lat,long), data=scallops,
+       fun=function(zi,zj) sqrt(abs(zi-zj))/2)
```

```
> boxplot(scallops.vcloud2, mean=T, pch.mean="o")
```

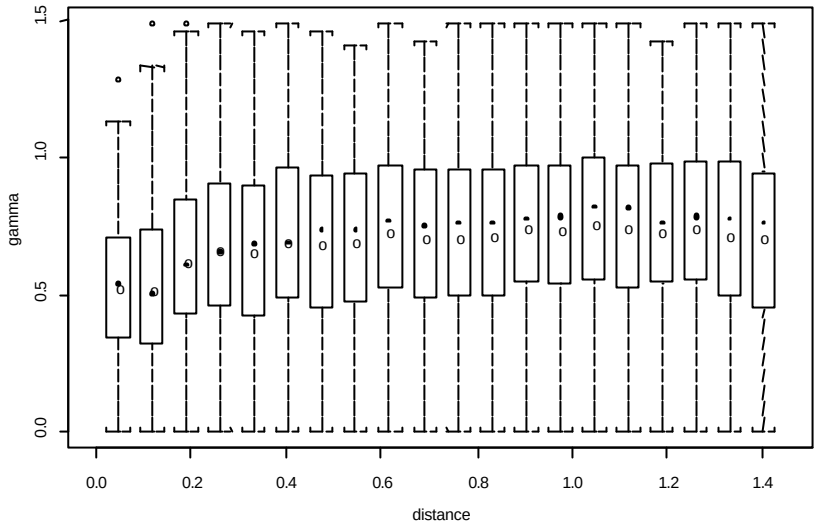


**図 4.9．ホタテデータに対する差の平方根のバリオグラム雲の箱ひげ図。**

**省略可能な引数** `fun` **は差の平方根を計算するのに使用された。引数** `mean` **と** `pch.mean` **は箱ひげ図に、中央値以外に平均値もプロットするために使われた。差の平方根のバリオグラムに対する箱ひげ図を描いたのが図** 4.9 **である。外れ値として残ったのは** 3 **点のみであり、これらはいずれも距離が** 0.2 **以下のところで発生している。**

### 4.1.3 トレンドの検出と除去

**バリオグラムが存在するためには、確率関数の本質的定常性（*intrinsic stationarity*）が仮定されていなければならない。これは一階差分に対して以下のように定義される：**

$$E\big(Z(i+h)-Z(i)\big)=0,\quad \text{for all } i, i+h\in D,$$

**かつ**

$$\operatorname{var}\big(Z(i+h)-Z(i)\big)=2g(h).$$

**基本的には、本質的定常性は過程が一定平均と、$h$ の大きさのみに依存する分散を持つことを示している。**

**しかし、地理統計データのモデルは、大域的トレンドもしくはドリフトと局所的ランダム変動の双方を組み合わせることが多い。この場合、確率場 $Z(x)$ は一定平均を持たず、$Z(x)$ のもととなるバリオグラムは必要な仮定を**

満たさないことになる。

トレンドの存在は、探索的データ解析によって明らかになることが多い。トレンドの存在を説明する作図方法は 3.2.1 節と 3.2.2 節でいくつか紹介した。本章の 4.1.1 節では、トレンドの存在を検出する方法として一方向バリオグラムを用いた。トレンドが検出されるかどうかに関わらず、目標は適切なモデリングとトレンドの除去であり、背後にある確率過程のバリオグラムを推定することである。

### Median Polishing

median polishing はグリッドデータのトレンドを除去するのに用いられる resistant な方法で、加法分解をもとにした方法である。つまり、

$$\text{データ} = \text{グランド}(grand) + \text{行}(row) + \text{列}(column) + \text{残差}(residuals)$$

median polishing は加法的なトレンドを仮定しているため、行と列が互いに影響し合うようなトレンドモデルには適当でない。このアルゴリズムは、各行から順に中央値を引き去った後、各列に対しても同様のことを行い、それらを行（*row*）、列（*column*）、グランド（*grand*）成分に集約する。残差はデータからそれらを引き去ったものである。

median polishing はグリッド上にデータが揃っていることを必要とするため、グリッドデータに対しては自然な方法である；グリッド上にないデータは、グリッドデータに変換したときにのみこの方法を用いることができる。関数 `twoway` を使って石炭データに median polish を用いると以下のようになる：

```
> coal.mp <- twoway(coal~x+y,data=coal.ash)
```

デフォルトでは、`twoway` は石炭データの各行、各列から中央値を引き去る。関数 `twoway` は、省略可能な引数 `trim` を変えることで他にも様々な加工を行うことができる。median polishing はそのバリエーションの 1 つである。

**注意：**ここで用いたのは S+SPATIALSTATS に含まれる総称関数 `twoway` である。内部で `twoway.fomula` が呼び出されている。

`twoway` の呼び出しにより、グランド、行、列、残差の各成分が返される。

以下のように、残差は元データと併用することでトレンドを見るのに利用できる：

1. 元データから median polish による残差を引き去り、シグナルを求める。

```
> coal.signal <- coal.ash$coal-coal.mp$residuals
```

2. 元データとシグナルのベクトルを行列に変換して、関数 image に使えるようにする。

```
> coal.mat <- tapply(coal.ash$coal,list(
+       factor(coal.ash$x),factor(coal.ash$y)),
+       function(x)x)
> coalsig.mat <- tapply(coal.signal,list(
+       factor(coal.ash$x),factor(coal.ash$y)),
+       function(x)x)
```

3. 作図装置と $z$の値の範囲を設定する。

```
> motif()  # UNIX 版の場合
> par(mfrow=c(1,2))
> # プロット領域を正方形にする
> par(pty="s")
> zmin <- min(coal.mat[!is.na(coal.mat)],
+       coalsig.mat[!is.na(coalsig.mat)])
> zmax <- max(coal.mat[!is.na(coal.mat)],
+       coalsig.mat[!is.na(coalsig.mat)])
```

4. 元データとシグナルを、共通の濃淡スケールを使ってプロットする。

```
> image(coal.mat, zlim=c(zmin,zmax))
> image(coalsig.mat, zlim=c(zmin,zmax))
```

出力結果は図 4.10 に示す通りである。シグナルを image によってプロットすることで、東西方向のトレンドが明らかになった（右側）。

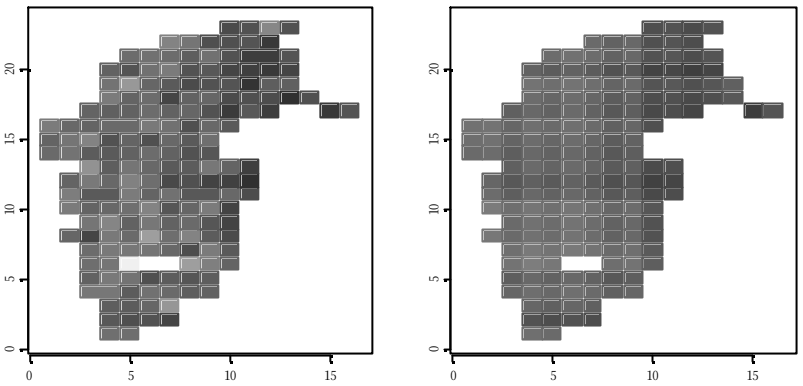


**図 4.10.** 石炭データの median polishing の結果。元データ（左）とシグナル(右)の濃淡プロット。東西方向にはっきりしたトレンドが見られる。

median polish によるトレンド除去の効果は、残差の東西方向のバリオグラムと、元データの東西方向のバリオグラムを比較することで確かめられる：

1. median polish による残差と元データの東西方向のバリオグラムを、共通の y 軸スケールを用いて作成する。

```
> coal.var5 <- variogram(coal.mp$residuals
+       ~ loc(x,y),data=coal.ash,
+       azimuth=90, tol.azimuth=11.25)
> coal.var6 <- variogram(coal~loc(x,y),data=coal.ash,
+       azimuth=90, tol.azimuth=11.25)
```

2. 東西方向のバリオグラムをプロットする。

```
> par(mfrow=c(2,1))
> ymax <- max(coal.var5$gamma,coal.var6$gamma)
> plot(coal.var5,ylim=c(0,ymax))
> plot(coal.var6,ylim=c(0,ymax))
```

median polish の残差と元データの東西方向のバリオグラムを図 4.11 に示す。これらを見る限り、東西方向に見られた相関のほとんどはトレンドによるものであると思われる。

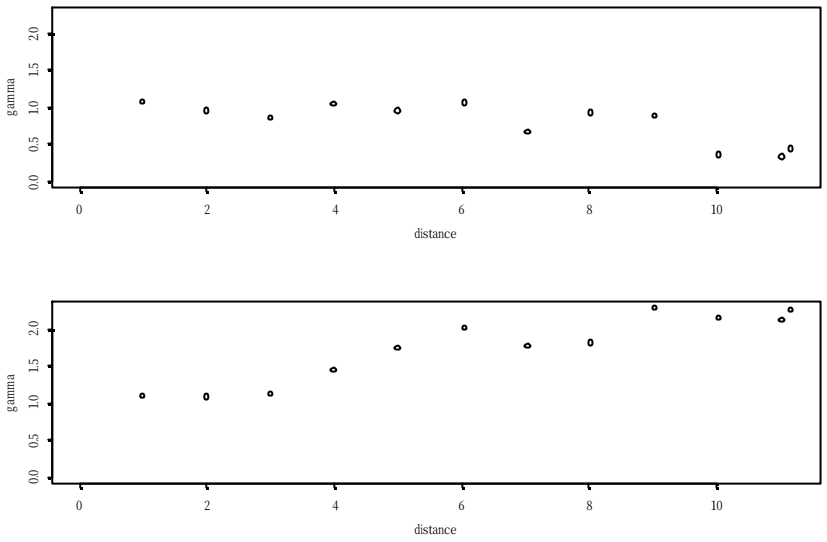


**図 4.11.** median polish の残差（上）と元データ（下）の東西方向のバリオグラム。

**その他の方法**

3.2.2 節では、ホタテデータに対して緯度と経度を予測変量とした平滑関数を用いた一般加法モデルを当てはめた。その結果得られた平滑関数はトレンドの存在を視覚化するのに役立った。トレンドの存在が確認されると、データに緯度と経度を予測変量とした空間局所回帰モデルを当てはめた。局所回帰モデルからの残差は、バリオグラム解析に使うことが出きる.

データのトレンドをモデリングし、除去するために、通常の線形モデルあるいは一般化線形モデルを用いてもよい。

### 4.1.4 異方性

**異方性**（*anisotropy*）はプロセスの空間的自己相関が方向によって変化する場合に出現する；背景にあるプロセスが空間上に一様に広がっていないのである。**等方的**（*isotropic*）なプロセスのバリオグラムと異なり、異方的なプロセスのバリオグラムは単なる距離 $h$ の関数ではなく、大きさと方向を持つ $\boldsymbol{h}$ の関数である。

異方性には 2 種類ある：バリオグラムのシルは一定だが、レンジが方向によって変動する場合に生じる**幾何異方性**（*geometric anisotropy*）；バリオグラムのシルが方向によって変動する**地域異方性**（*zonal anisotropy*）。

異方性の検出と修正は重要である。なぜなら、クリギングに使用される理論バリオグラムは等方的なモデルを元にするからである。幾何異方性は一般に、空間を線形変換することで等方的なモデルに修正できる。地域異方性はデータのトレンドを適切に判定し除去するか、入れ子（*nested*）バリオグラムモデルを選択することで修正できる。入れ子モデルの 1 成分を、地域異方性を示す方向に当てはめ、それを等方的なモデルとして扱うのである；その他の成分は幾何異方的なバリオグラムとなる。

### 異方性の検出

`variogram` によって生成される一方向バリオグラムは異方性の検出に使用することが出きる。3.2.2 節で最初に導入した、回転後のホタテデータを継続して使用する。図 4.12 のような、いくつかの方向に対するバリオグラムを描いて見る：

```
> scallops.dvar1 <- variogram(lgcatch ~ loc(newx,newy),
+       data=scall.rot, azimuth=c(0,45,90,135),
+       tol.azimuth=11.25)
> plot(scallops.dvar1)
```

45 ° と 135 ° の方向のバリオグラムは類似しているが、0 ° と 90 ° の方向にははっきりとした異方性を見ることができる。回転後の空間において、0 ° と 90 ° の方向はそれぞれ海岸線に対してほぼ垂直と平行な方向に相当する。

0 ° の方向のバリオグラムは、最後の 4 点を除くと（ペアの数が少ないという理由から）上限のない増加バリオグラムである。これはおそらく、一般化加法モデルを使った探索的データ解析によって検出されたトレンド（3.2.2 節）によるものである。

90 ° の方向のバリオグラムには幾何異方性があるようである。シルはほぼ 5 であり、45 ° と 135 ° のバリオグラムのシルと似通っている；しかし、レンジ（約 1）の方は 45 ° と 135 ° のバリオグラムのおおよそ 2 倍であ

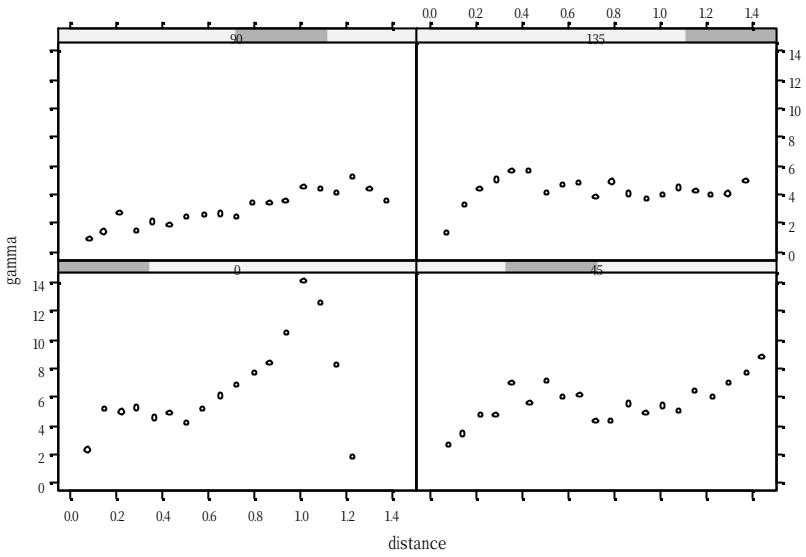


**図 4.12.** 回転後のホタテデータに対する一方向バリオグラム。0 ° と 90 ° の方向に異方性が見られる。

る。よって明らかに、海岸線に平行な方向の自己相関は他のどの方向の自己相関よりも大きい。これは予想できる結果である。なぜなら、海岸線に平行な方向には垂直な方向に比べて類似した環境条件が出現しやすいからである。

### 異方性の修正

図 4.12 における 0 ° のバリオグラムの異方性がトレンドによるものならば、トレンドを除去したデータのバリオグラムを描くことにより明らかになる。図 4.13 は、回転後のホタテデータに空間局所回帰モデル（局所回帰モデルについては 3.2.2 節を参照。）を当てはめて得られた残差について同じバリオグラムを求めたものである。

```
> scall.res <- scall.rot$lgcatch – predict(loess.scp)
> scallops.dvar2 <- variogram(scall.res
+       ~ loc(scall.rot$newx, scall.rot$newy),
+       azimuth=c(0,45,90,135), tol.azimuth=11.25,
+       method=”robust”)
```

```
> plot(scallops.dvar2)
```

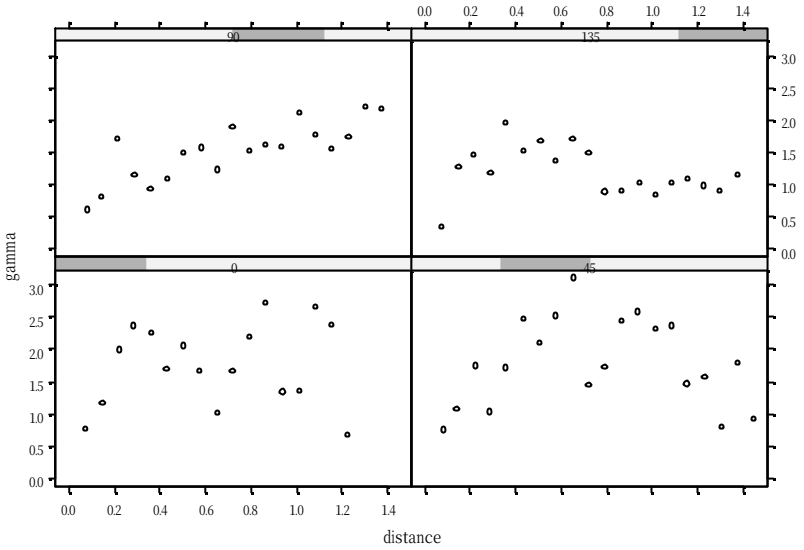


**図 4.13. ホタテデータに局所回帰モデルを当てはめて得られた残差に対する一方向バリオグラム。**

0°のバリオグラムにはもはや増加傾向は見られず、レンジもシルも45°のバリオグラムに類似している。90°のバリオグラムは未だ他の方向のバリオグラムより大きいレンジを持っている。135°のバリオグラムにはナゲット効果があるが、空間的相関はごく僅か、もしくは無いと言ってよい.シルはどのバリオグラムでもほぼ同じで、元の一方向バリオグラムよりも減少している。これはトレンドを除去したことに矛盾しない。

幾何異方性をより明確にするためには、方向によってレンジがどう変化しているかを記述する必要がある。これは、様々な距離と方向に対するγの等高線を描いて評価することができる。以下はこれを行なう1つの方法である：

1. あるγの値に対する距離を計算する関数（panel function）を作る。バリオグラム関数の近似には局所平滑化を使い、局所平滑曲線上のある値に対する距離を決定するのには関数approxを使う。

```
> panel.gamma0 <-
+ function(x, y, gamma0 = gamma0, span = 2/3, ...)
+ {
+     lofit <- loess.smooth(x, y, span = span)
```

```
+     panel.xyplot(x, y, ...)
+     panel.xyplot(lofit$x, lofit$y, type="l")
+     dist0 <- approx(lofit$y, lofit$x,
+         xout = gamma0)$y
+     segments(0, gamma0, dist0, gamma0)
+     segments(dist0, 0, dist0, gamma0)
+     parusr <- par()$usr
+     text(parusr[2] – 0.05 * diff(parusr[1:2]),
+         parusr[3] + 0.05 * diff(parusr[3:4]),
+         paste("d0=", format(round(dist0, 4)),
+         sep = ""), adj = 1)
+ }
```

手間を省くため、この関数は S+SPATIALSTATS に含まれている；しかしヘルプファイルはない。

2. ペアの数が少ないため、scallops.dvar2 の 0° と 45° の方向の、距離 0.9 以内の点のみを含むような部分集合を作る。

```
> scallops.aniso <- scallops.dvar2[
+      (scallops.dvar2$distance < 0.9 &
+      scallops.dvar2$azimuth == 0) |
+      (scallops.dvar2$distance < 0.9 &
+    scallops.dvar2$azimuth == 45) |
+    scallops.dvar2$azimuth == 90 |
+    scallops.dvar2$azimuth == 135,]
```

3. xyplot と関数 panel.gamma0 を使ってバリオグラムを描き、γ = 1.4 に対する内挿距離（interpolated distance）を書き入れる。

```
> xyplot(gamma = ~ distance | azimuth,
+       data=scallops.aniso, panel=panel.gamma0,
+       gamma0=1.4)
```

図 4.14 はγ = 1.4 に対する内挿距離を書き込んだものである。

内挿距離 $h$ は 45° と 135° の方向では類似している。最も顕著な違いは 90° の方向に出ている。

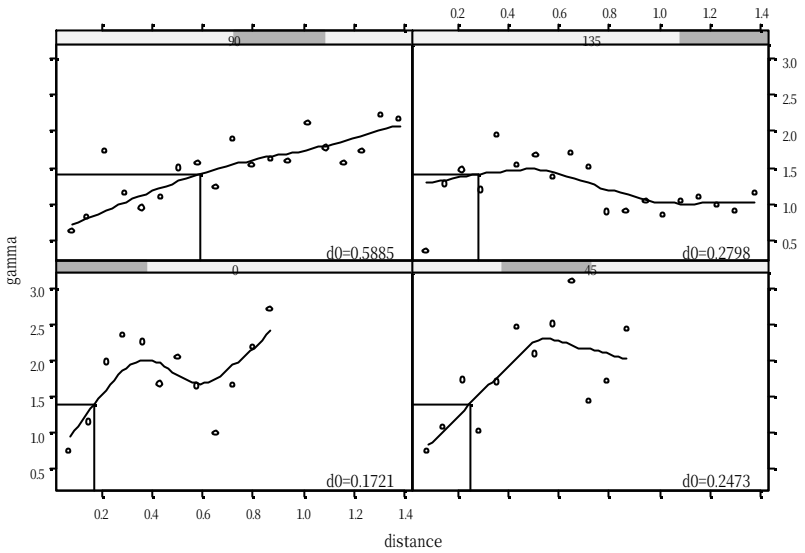


**図 4.14.** ホタテデータに局所回帰モデルを当てはめて得られた残差に対する一方向バリオグラムに $\gamma = 1.4$ に対する内挿距離を書き込んだもの。

**注意：**局所平滑化の際には独立性の仮定が必要になるが、バリオグラムの値はこれを満たさない。ここでは 1 次近似を得るためだけに使用したが、異方性を測定するにはこれで十分なように思われる。

この内挿距離は、異方性を検出・視覚化するもう 1 つの方法である rose diagram (Isaaks and Srivastava, 1989)の作成にも使用できる。rose diagram は、$\gamma$ の等高線図をもとにいくつかの方向に対して内挿距離を表示するものである。図 4.15 の rose diagram を作成するには以下のようにする：

1. 内挿距離 $h$ を、最大値に対する相対距離に変換する。この場合、0° から 135° の相対距離はそれぞれ、.292、.420、1.000、.475：

```
> dscale <- .5885
> d0 <- .1721/dscale
> d45 <- .2473/dscale
> d90 <- 1
> d135 <- .2798/dscale
```

2. 各方向の相対距離を単位 $xy$ 平面上で表現する場合の各点の位置を決

定する。0°と 90°の方向の各点は(0,.292)と(1,0)。45°の方向に対しては、点の位置は $r$ を相対距離として、円の方程式 $x^2 + y^2 = r^2$ を解いて求める。角度 45°のとき、$x = y$ であるから、計算は $x^2 + x^2 = r^2$ に簡略化される。45°に対する点は(.297,.297)となる。135°に対する点は(.336,.336)になる。

3. 関数 `plot` と `segments` を使って rose diagram を作成する：

```
> plot(0,0,type="n", axes=F, xlim=c(-1,1),
+      ylim=c(-1,1),
+      xlab="along scall.rot$newx",
+      ylab="along scall.rot$newy")
> segments(0,0,1,0)
> segments(0,0,0,.292)
> segments(0,0,.297,.297)
> segments(0,0,.336,-.336)
> segments(0,0,-1,0)
> segments(0,0,0,-.292)
> segments(0,0,-.297,-.297)
> segments(0,0,-.336,.336)
```

rose diagram は対称的である（バリオグラムが対称的だから）；線分は 180°反対側にも伸ばしてある。

rose diagram の形はおおよそ、90°の方向を長軸とする楕円である。このことから、幾何異方性の形がより明らかになった。これは線形変換で修正することができる。

S+SPATIALSTATS には、データの線形変換を行ない、結果として生じる等方的バリオグラムをプロットする関数 `anisotropy.plot` がある。ホタテデータの幾何異方性を修正するには以下の手続きを踏む：

```
> anisotropy.plot(scall.res ~ loc(scall.rot$newx,
+      scall.rot$newy), angle=90,
+      ratio=c(2.25,2.5,2.75,3.0,3.25,3.5),
+      method="robust", layout=c(2,3))
```

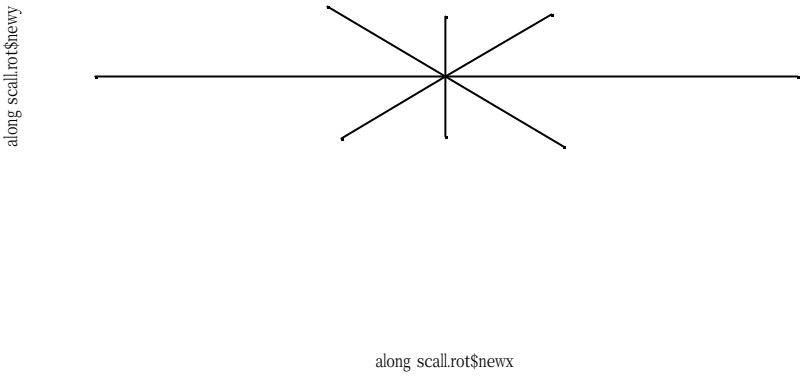


**図 4.15.** ホタテデータに局所回帰モデルを当てはめて得られた残差の、γ = 1.4 に対する内挿距離をもとにして作成した rose diagram。

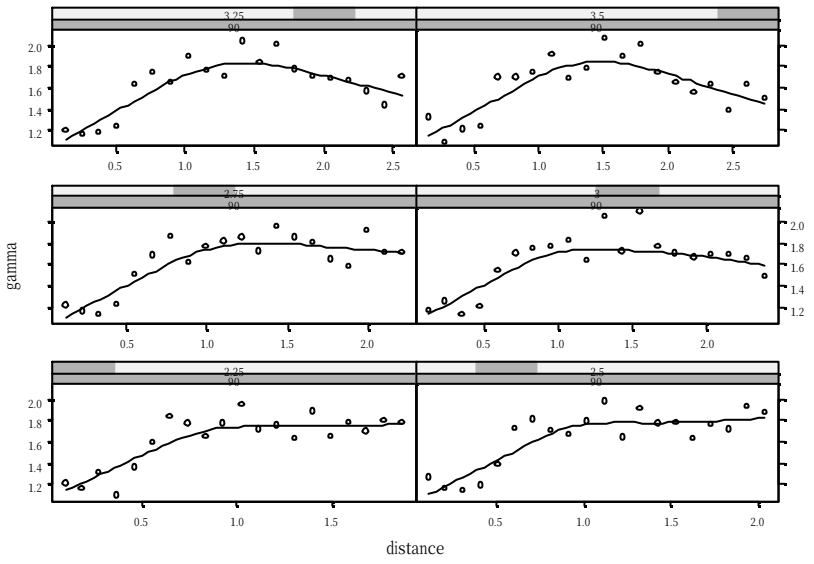


**図 4.16.** トレンドと幾何異方性を修正したホタテデータのバリオグラム。

省略可能な引数 `angle` は時計回りの回転角（°）で、*y* 軸が rose diagram の楕円の長軸に平行になる様に回転させる角度である。省略可能な引数 `ratio` は楕円の長軸と短軸の長さの比である。ホタテデータは、軸を 90° 回転させなければならないので、`angle` を 90° に設定した。γ = 1.4 と

なる距離から、楕円の長軸と短軸の比は約 3 である。よって比較のために 3 の周辺の値を ratio に設定した。図 4.16 は線形変換によって得られたバリオグラムを表わしている。

幾何異方性とトレンドを修正したバリオグラムは、どれも似通っている。モデルの当てはめにどのバリオグラムを選択するかは解析者に委ねられる。

## 4.2 経験バリオグラムのモデリング

4.3 節では、バリオグラムをクリギング方程式に用いて予測やクリギング予測分散を求める方法について説明する。分散や予測値を、現実味を帯びたものにするために、経験バリオグラムを理論バリオグラム関数に置き換えなければならない。

### 4.2.1 理論バリオグラムモデル

S+SPATIALSTATS は理論バリオグラムとしてよく利用される関数を提供している。有界バリオグラム関数には指数型（exponential）モデル、球型（spherical）モデル、ガウス型（gaussian）モデルを、非有界バリオグラム関数には線形（linear）モデル、ベキ乗（power）モデルを用意した。図 4.17 のような理論バリオグラムの例をプロットするには以下のようにする：

```
> par(mfrow=c(3,2))
> vdist <- 1:15
> vrange <- 7
> plot(vdist,exp.vgram(distance=vdist, range=vrange),
+       type=”l”)
> plot(vdist,spher.vgram(distance=vdist, range=vrange),
+       type=”l”)
> plot(vdist,gauss.vgram(distance=vdist, range=vrange),
+       type=”l”)
> plot(vdist,linear.vgram(distance=vdist, slope=.3),
+       type=”l”)
> plot(vdist,power.vgram(distance=vdist, slope=.3,
```

```
+       range=.5), type="l")
```

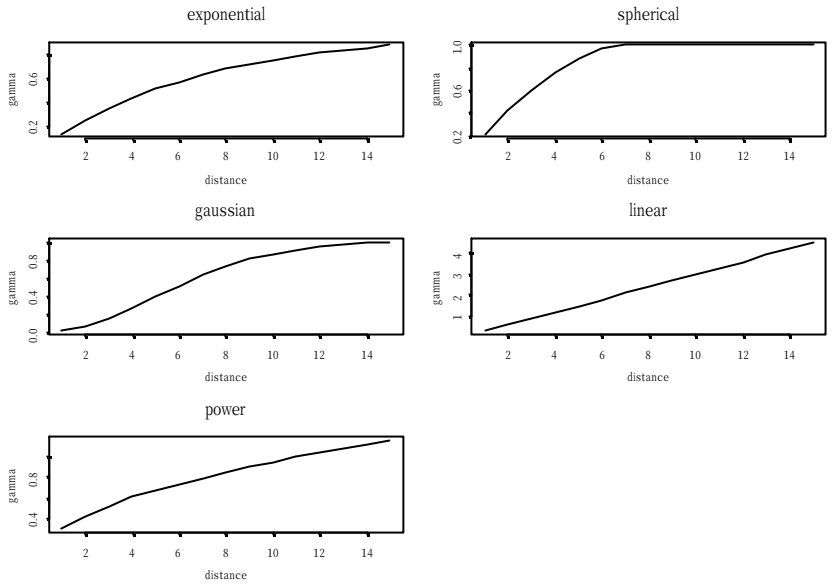


**図 4.17.** S+SPATIALSTATS の理論バリオグラムモデル。

ベクトル distance はどの理論バリオグラム関数でも指定しなければならない。指数型、球型、ガウス型の各モデルには range も必要である。線形、巾乗の各モデルには slope が必要となる。省略可能な引数 nugget はすべてのモデルで指定でき、デフォルトは 0 である。有界関数型には省略可能な引数 sill を指定することもできる（デフォルトは 1）。巾乗モデルには引数 range が必要である。

exp.vgram のレンジは見かけの（*apparent*）レンジの約 3 分の 1 と定義されている。ここで見かけのレンジとは、データがもはや相関を持たないとみなされた距離 $h$ のことである。球型、ガウス型モデルの引数 range は見かけのレンジである。ベキ乗型モデルの引数 range は距離の指数である[3]。引数 sill は**絶対**（*absolute*）シルを表わす；絶対シルは経験バリオグラムのシルから任意のナゲット効果を引いたもので推定する。

### 4.2.2 理論バリオグラムの当てはめ

経験バリオグラムを理論バリオグラムに当てはめる作業は、多くは解析者の目によって行われる。計算機による当てはめルーチンを使用する場合で

[3] **訳注：** $c_0$、$c_1$ をある定数、range を $c_2$ とするとベキ乗型モデルは $g(h) = c_0 + c_1 h^{c_2}$ 。

も、初期値は解析者が目で見て決定する方法が有効である。はじめにどのモデルを選択するかということも、目で見た際の経験バリオグラムの形状と、背景にある過程の性質を元にした解析者の信念によって決定される。レンジ、シル、ナゲット効果の各パラメータに対する初期値も経験バリオグラムから決定される。理論バリオグラムモデルの当てはめは、S+SPATIALSTATS関数の`model.variogram`によって対話的に行うことができる。`model.variogram` の呼び出しの中で各パラメータの値は更新され、最終的に満足ゆく当てはめができるまで何度でも更新は繰り返される。

幾何異方性を修正したホタテデータのバリオグラム（4.1.4 節）に理論バリオグラムモデルを当てはめる方法は以下の通り：

1. 幾何異方性を適切に修正した（変換した）データを元に経験バリオグラムを計算する。図 4.16 より、90 ° に対する変換の比が 2.25 のときのバリオグラムが妥当である。変換済みバリオグラムを以下のようにして作成する：

```
> scallops.finalvr <- variogram(scall.res
+       ~loc(scall.rot$newx, scall.rot$newy,
+       angle=90, ratio=2.25), method="robust")
```

2. 関数の型とパラメータの初期値を選ぶ。経験バリオグラムの一般的な形状（図 4.16 下段左）を元に、レンジ = .8、シル = 1.75、ナゲット効果 = .5 の球型モデルを初期値とする。

3. 上の初期値を元に`model.variogram`を呼び出す。

```
> model.variogram(scallops.finalvar,
+ fun=spher.vgram,range=.8, sill=1.75-.5, nugget=.5)
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.4848
1: range - current value: 0.8
2: sill - current value: 1.25
3: nugget - current value: 0.5
Selection:
```

関数`model.variogram`は初期値を代入した理論バリオグラム及び経験バリオグラム（図 4.18 上段左）と、現在のモデルに対するパラメータの変更を促す対話メニューを表示する。球型バリオグラムの

パラメータ、レンジ、シル、ナゲット効果のいずれかを修正するには 1、2、3 のいずれかを選ぶ。修正が行われるごとに修正後の理論バリオグラムが自動的にプロットされる。関数 `model.variogram` には、省略可能な引数 `objective.fun` によるモデルの当てはまり具合の尺度を測る機能もある。デフォルトでは、この objective function は理論バリオグラムと経験バリオグラムの残差平方和である。パラメータが修正される毎にこの objective value も計算され、コマンドウィンドウと作図ウィンドウに表示される。ユーザが objective function を定義してもよい。プロンプト `Selection:` で `0` を入力すると `model.variogram` を終了できる。

4. パラメータの値を修正して当てはめを改善する。例えば、ナゲット効果を増加させて、（シルを適当に修正した後に）レンジを減少させた方が良いように思われる。これは以下のようにして行う：

```
Selection: 3
New nugget : .7
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.9706
1: range - current value: 0.8
2: sill - current value: 1.25
3: nugget - current value: 0.7
Selection: 2
New sill : 1.05
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.3385
1: range - current value: 0.8
2: sill - current value: 1.05
3: nugget - current value: 0.7
Selection: 1
New range : .75
Select a number to change a parameter (or 0 to exit):
 Current objective = 0.3556
1: range - current value: 0.75
2: sill - current value: 1.05
3: nugget - current value: 0.7
```

```
Selection: 0
```

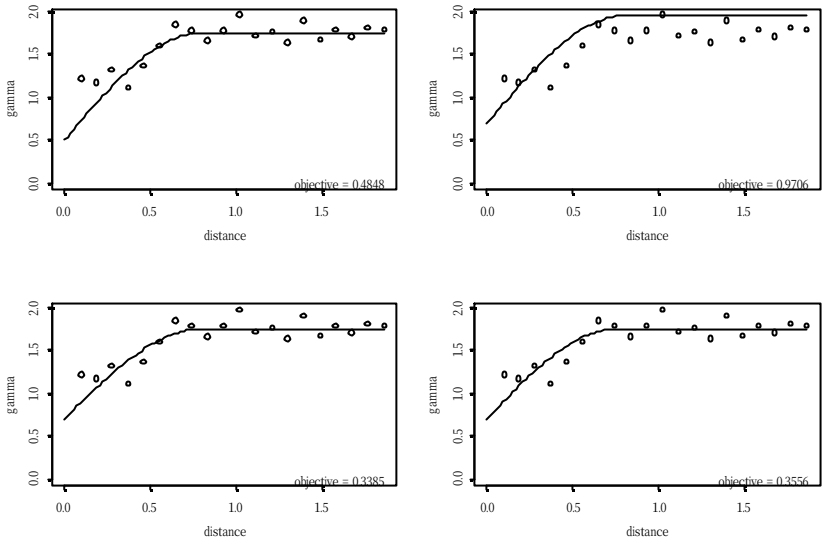


**図 4.18.** ホタテデータの経験バリオグラムに球型モデルの当てはめを行った経過を示したもの。

モデルを当てはめた結果が図 4.18 である。対話メニューを使って、最初にナゲット効果を 0.7 に増加させた（上段右）；ナゲット効果の増加を補正するために、シルを 1.05 に減少させた（下段左）；レンジを 0.75 に減少させた。残差平方和は、レンジを増加させる前までは各回とも減少していた。最終的にはレンジ ＝ 0.8、シル = 1.75、ナゲット効果 = 0.7 の球型バリオグラムをモデルとして選択した。

関数 `model.variogram` は、コバリオグラムやコレログラムに対する当てはめを対話的に行う際にも使用できる。

### 非線形最小 2 乗法による当てはめ

バリオグラムモデルの当てはめには、最適化の技法をを使うこともできる。その典型的な例が非線形最小 2 乗法である。非線形回帰モデルを用いる際に通常課せられる仮定は、バリオグラムの当てはめには有効ではない。各ラグにおけるバリオグラム値は独立ではないからである。Cressie (1985) はバリオグラム値に固有な構造を説明するため、重み付き最小 2 乗法あるいは一般化最小 2 乗法を行っている。Zimmerman and Zimmerman (1991)は数種類の推定法を比較し、通常の非線形最小 2 乗法または重み付

き非線形最小 2 乗法のいくつかは、より複雑で計算量の多い他の方法の多くと同等に良い当てはめを行うと結論している。

石炭データのバリオグラムのモデリングに非線形最小 2 乗法を使用してみることにする。以前（3.2.1 節）、このデータには東西方向にトレンドがあると断定したので、以下では南北方向のみのバリオグラムに球型バリオグラムモデルを当てはめることにする：

1. 南北方向のバリオグラムを求め、プロットする。

```
> coal.varns <- variogram(coal ~ loc(x, y),
+      data=coal.ash, azimuth = 0,
+      tol.azimuth = .01, lag = 1)
> plot(coal.varns)
```

求められたバリオグラムは図 4.19 に示した。

2. 平方和を最小化する際に使用する残差を定義する関数を書く。球型バリオグラム関数 `sper.vgram` を使用する。

```
> spher.fun <- function(gamma, distance, range,
+       sill, nugget)
+      gamma - spher.vgram(distance, range = range,
+      sill = sill, nugget = nugget)
```

3. 初期値を設定する。`coal.varns` のバリオグラムのプロットより、レンジ ＝ 4.0、シル ＝ 0.2、ナゲット効果 ＝ 0.8 を初期値とする。

4. 上で定義した `spher.fun` を使って、関数 `nls` を呼び出す。

```
> coal.nl1 <- nls( ~ spher.fun(gamma, distance,
+    range, sill, nugget), data = coal.varns,
+    start = list(range = 4, sill = 0.2, nugget=.8))
> coef(coal.nl1)
     range       sill     nugget
 3.443336   0.240059   1.093492
```

5. 経験バリオグラムに当てはめたモデルを描き加える（図 4.19）。

```
> lines(coal.varns$dist,
+     spher.vgram(coal.varns$dist,
+     range=3.443336,sill=.240059,nugget=1.093492))
```

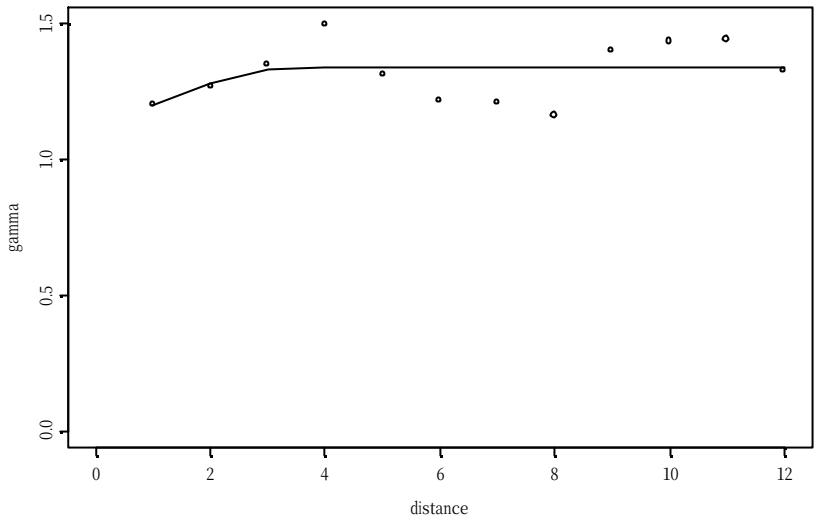


**図 4.19.** 石炭データの経験バリオグラム（白丸）と非線形最小 2 乗法によって推定した球型バリオグラムモデル（実線）。

Cressie (1985)は以下の重み付き 2 乗和の最小化を提案している：

$$\sum_{j=1}^{K} |N(h(j))| \left\{ \frac{g(h(j))}{g(h(j); \boldsymbol{?})} - 1 \right\}^2,$$

ここで、$|N(h(j))|$はラグが $j$ となるペアの数、$K$ は経験バリオグラムにおけるラグの数、$\gamma(h(j))$はラグ $j$ における経験バリオグラム値、$\gamma(h(j); \boldsymbol{\theta})$は仮定したある理論バリオグラムモデルのラグ $j$ における値。パラメータ $\boldsymbol{\theta}$ は未知。

残差関数の中でこの重み関数を定義することで、`nls` の中でこれを使うことができる：

```
> sper.wfun <-
+ function(gamma, distance, np, range, sill, nugget)
+ {
+ gammahat <- spher.vgram(distance, range = range,
+ sill = sill, nugget = nugget)
+    sqrt(np) * (gamma/gammahat - 1)
+ }
```

以下では、上で定義した重み付き関数を使って石炭データの理論バリオグラムのパラメータを推定する：

```
> coal.nl2 <- nls( ~ sper.wfun(gamma, distance,
+       np, range, sill, nugget), data = coal.varns,
+       start = list(range = 4, sill = 0.2, nugget=.8))
> coef(coal.nl2)
     range      sill    nugget
 3.658897 0.2427316 1.099064
```

この例では、重み付き非線形最小 2 乗法による係数と重み付きでない場合の係数との間に大差はない。

非線形の当てはめを行う他のS-PLUS関数にはmsやnlminbなどがある。これらの関数の詳細については個々のヘルプファイルを参照。Bates and Chambers (1992)や Venables and Ripley (1994)にも、S-PLUS による非線形モデルの当てはめに関する記述がある。

## 4.3 クリギング

クリギングは、観測の行われた地点の値を用いてランダム関数の未知の値を予測する補間法である。未知の値を予測する際、クリギングはランダム関数の共分散モデルを用いる。S+SPATIALSTATS は 2 種類のクリギングを行う関数を提供する：**通常**（*ordinary*）クリギングと**普遍**（*universal*）クリギングである。通常クリギングは、空間相関を持ったランダム関数モデルを使って、値の得られていない位置での値を、得られている位置の値の重み付き線形和で予測するものである。重みは、平均誤差が 0 で当てはめ誤差が最小になるように選ばれる（Isaaks and Srivastava, 1989）。普遍クリギングはトレンドが存在するもとでの局所的な推定にも、またトレンド自体の推定にも使うことができる。平均が一定である場合の普遍クリギングは通常クリギングに等しい。

### 4.3.1 通常クリギング

2 次元におけるクリギングは、S+SPATIALSTATS の関数 krige と predict.krige を使って行う：krige が予測のためのクリギング行列を生成するには、クリギング反応変量、位置、理論共分散関数を必要とする；predict.krige は krige の出力を用いて、ユーザが指定した値の得られていない位置のクリギング予測値と標準誤差を計算する。理論共分

散関数は理論バリオグラムを元に計算され、指数型、球型、ガウス型の各モデルに対して定義される。線形とベキ乗型モデルは有界でないため、これらに相当する共分散モデルはない。しかし、線形バリオグラムモデルの値を大きな値（予測がなされる中でもっとも離れた距離における $\gamma$ の値）で置き換えることで線形共分散モデルを計算することもできる。

ホタテデータに対して通常クリギングを行ってみる：

```
> scallops.krige <- krige(scall.res~loc(newx,newy,
+      90,2.25), data=scall.rot, covfun=spher.cov,
+      range=.8, sill=(1.75-.7), nugget=.7)
```

クリギング変量 `scall.res` は空間局所回帰モデルの残差である。位置は回転した座標系 `newx` と `newy` における位置である。空間相関は経験バリオグラム（図 4.18 左下）に当てはめた球型バリオグラムモデルをもとにした球型共分散としてモデリングされている。`spher.vgram` と同様、`spher.cov` でも引数 `sill` はシルからナゲット効果を引いたものが用いられる。

関数 `krige` はクラス`"krige"`のオブジェクトを返すが、これには関数呼び出しに対する要約や求められた係数が含まれている：

```
> scallops.krige
Call:
krige(formula = scall.res ~ loc(newx, newy, 90, 2.25),
     data = scall.rot, covfun = spher.cov, range
      = 0.8, sill = (1.75 - 0.7), nugget = 0.7)

Coefficients:
   constant
 -0.1873672

Number of observations: 148
```

ここで `predict.krige` を使って、値の得られていない位置でのクリギング予測を行うことができる。予測したい位置を定義するのには 2 通りの方法がある。1 つ目は引数 `newdata` を指定する方法である；`newdata` は予測したい位置のデータフレームもしくはリストである。2 つ目は、引数 `grid` を使って点のグリッドを生成する方法である；`grid` は 2 つのベクトルのリストであり（各ベクトルが 1 つの軸に対応）、それぞれ最小値、

最大値、点の個数を指定する。newdata と grid のどちらの場合でも、軸の名前は krige の呼び出しに用いた軸の名前と同じでなければならない。デフォルトでは予測される位置は、値の得られている位置の座標の最大値と最小値で定義される 30×30 のグリッドになる。ホタテデータの残差のクリギング予測をデフォルトの位置に対して行ってみる：

```
> scallops.pkrige1 <- predict(scallops.krige)
```

総称関数である predict がクラス"krige"のオブジェクトに対して呼び出されると、メソッド predict.krige が使用される。predict の呼び出しによって返ってくるデータフレームは 4 列で構成される。予測値の位置 $x$、$y$、予測値、予測値の標準誤差の 4 つである。以下に、予測された位置を値の得られている位置とともにプロットする：

```
> plot(scall.rot$newx, scall.rot$newy,
+      xlab="newx", ylab="newy", pch=16)
> points(scallops.pkrige1$newx, scallops.pkrige1$newy,
+      pch="+")
```

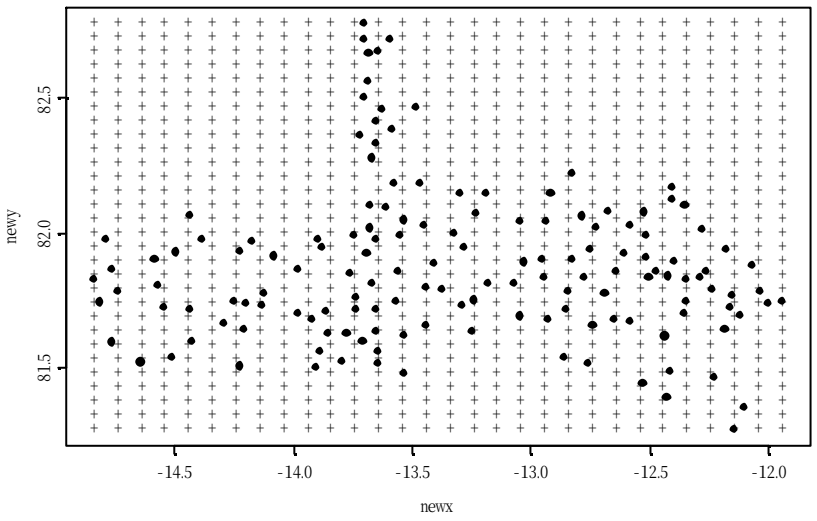


**図 4.20.** 元のホタテ採集位置（黒丸）とデフォルトの予測位置（＋）。

図 4.20 は、値の得られている位置と、predict が返したデフォルトの予測値の位置を示したものである。

予測したこれらの位置は、ホタテデータにとって満足のいかないものかも知れない。海岸に近かったり、採集の行われた位置から遠かったりする点があまりに多いからである；デフォルトの予測位置の中には陸地のところ

もある。引数 grid を使うのであれば、主要な採集領域（長方形）の座標の最大値と最小値を使う必要がある。これに代わる方法として、S+SPATIALSTATS や S-PLUS には多角形を境界にもつ予測領域を生成できる関数がいくつかある。関数 chull は採集領域の凸包を求める；locator は採集領域の周囲あるいは内部にユーザ定義の多角形を対話的に作ることができる。図 4.21 のような多角形境界を生成するには以下のようにする：

```
> par(mfrow=c(1,2))
> par(pty="s")
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> scallops.chull<- chull(scall.rot$newx,scall.rot$newy)
> polygon(scall.rot$newx[scallops.chull],
+     scall.rot$newy[scallops.chull], density=0)
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> scallops.poly <- locator(type = "l")
```

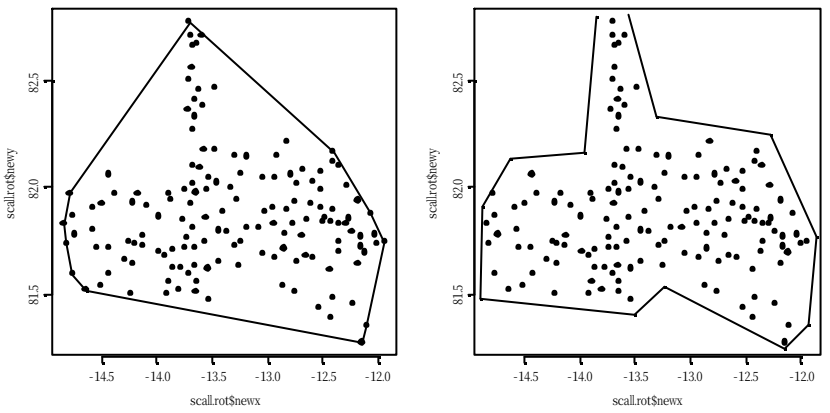


**図 4.21.** ホタテデータの採集地を元にした凸包（左）とユーザ定義の多角形（右）。

chull は凸包の頂点に相当する位置ベクトルのインデックスを返す。関数 polygon は現在のプロット図に凸包を描き加える。デフォルトでは、polygon は凸包内部を塗りつぶしてしまう；省略可能な引数 density

を 0 にして輪郭だけを描くようにする。関数 locator は省略可能な引数 type を"l"にすると、ユーザが選んだ点をもとに多角形を描く。多角形の境界をマウスの左ボタンで選んでゆく。locator を終了するには作図ウィンドウ内で UNIX では中央ボタンを、Windows では右ボタンをクリックする。ユーザ定義の多角形は閉じている必要はない。

poly.grid を使えば、凸包やユーザ定義の多角形を境界に持つ予測領域を定義することができる。図 4.22 のような予測領域を生成するには以下のようにする：

```
> par(mfrow=c(1,2))
> par(pty="s")
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> predict.loc1 <- poly.grid(cbind(
+      scall.rot$newx[scallops.chull],
+      scall.rot$newy[scallops.chull]), nx=20, ny=20)
> points(predict.loc1,pch="+")
>
> plot(scall.rot$newx, scall.rot$newy, pch=16)
> predict.loc2 <- (poly.grid(scallops.poly,
+      size=c(0.08,0.05)))
> points(predict.loc2, pch="+")
```

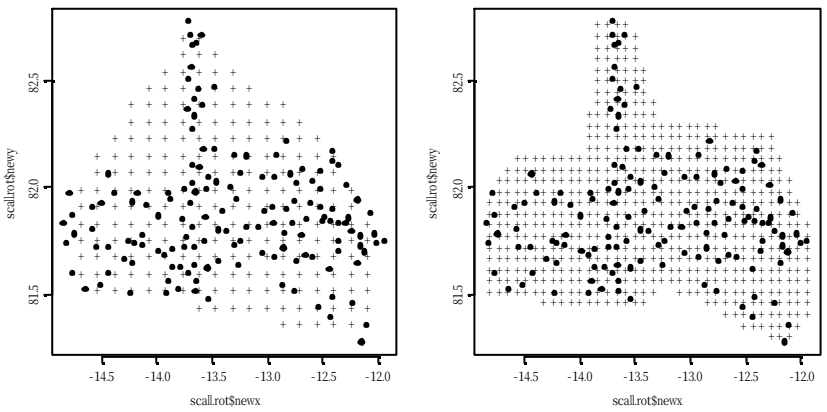


**図 4.22.** 凸包(左側の＋)とユーザ定義の多角形(右側の＋)の予測領域。

左側の図（図 4.22）の予測領域は、poly.grid の引数 nx と ny を指定して結果が $nx \times ny$ のグリッドになるようにした。右側の図は省略可能な引数 size を指定して、$x$ 方向には 0.08 間隔、$y$ 方向には 0.05 間隔になるようにした。

ホタテデータの残差に対して、図 4.22 のユーザ定義の予測範囲内で通常クリギングを行うには以下のようにする：

1. poly.grid の出力結果である predict.loc2 をデータフレームに変換する。予測領域の軸の名前を変更して、krige の呼び出しの際に用いたものと同じ物にする。

   **注意：** 関数 predict.krige がデータフレームを必要とするのではない。後に predict.loess を用いる際に必要になる。

   ```
   > predict.loc2  <-
   +     as.data.frame(predict.loc2)[c(1,2)]
   > names(predict.loc2) <- c("newx","newy")
   ```

2. 予測を行う。

   ```
   > scallops.pkrige2 <- predict(scallops.krige,
   +       newdata=predict.loc2)
   ```

3. 予測値の要約を見る。

   ```
   > summary(scallops.pkrige2)
         newx             newy              fit
   Min.    :-14.82   Min.    :81.29   Min.    :-2.99300
   1st Qu. :-13.92   1st Qu. :81.65   1st Qu. :-0.35030
   Median  :-13.42   Median  :81.87   Median  :-0.09489
   Mean    :-13.35   Mean    :81.88   Mean    :-0.23070
   3rd Qu. :-12.68   3rd Qu. :82.08   3rd Qu. : 0.19680
   Max.    :-11.94   Max.    :82.76   Max.    : 2.13800
        se.fit
   Min.    :0.4019
   1st Qu. :1.0020
   Median  :1.0360
   Mean    :1.0510
   3rd Qu. :1.0860
   Max.    :1.2430
   ```

この場合、newx と newy が予測範囲の軸、fit が残差予測量、se.fit がクリギング予測標準誤差である。以下では、ホタテデータの予測曲面を生成するために、空間局所回帰モデルにより予測したトレンド曲面を、クリギング残差予測量に加える：

```
> scall.lo <- predict(loess.scp, predict.loc2)
> Scall.pred <- scallops.pkrige2$fit + scall.lo
```

予測曲面と相対標準誤差の等高線図を作るには以下のようにする：

1. ホタテデータの予測量とクリギング予測標準誤差を predict.loc2 の範囲の行列に変換する。

```
> xmat <- sort(unique(predict.loc2$newx))
> ymat <- sort(unique(predict.loc2$newy))
> scall.predmat < - matrix(NA, length(xmat),
+     length(ymat))
> scall.predmat[cbind(match(predict.loc2$newx,
+     xmat), match(predict.loc2$newy, ymat))] <-
+     scall.pred - 1
> scall.semat < - matrix(NA, length(xmat),
+     length(ymat))
> scall.semat[cbind(match(predict.loc2$newx,
+     xmat), match(predict.loc2$newy, ymat))] <-
+     scallops.pkrig2$se.fit
```

2. 元の採集位置をプロットする。これに、クリギング予測と空間局所回帰モデルを元に作成した等高線図を上描きする。相対標準誤差の等高線図をプロットする。

```
> plot(scall.rot$newx, scall.rot$newy)
> contour(xmat,ymat,scall.predmat,
+     levels=c(-6,-4,-2,0,2,3,4,5,6),
+     add=T,lty=3)
> plot(scall.rot$newx, scall.rot$newy)
> contour(xymt,ymat,scall.semat)
```

出力結果は図 4.23 と図 4.24 に示した。

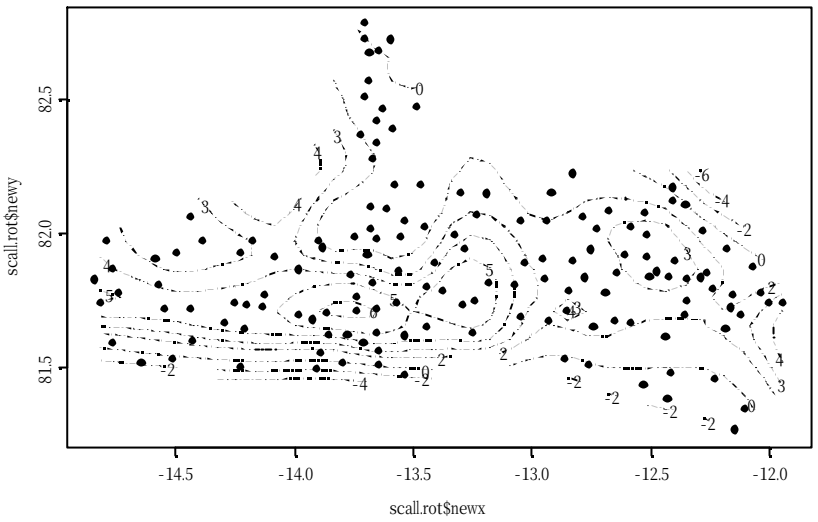


**図 4.23.** 対数をとったホタテ貝の総捕獲数に対してクリギング予測を行った結果の等高線図。

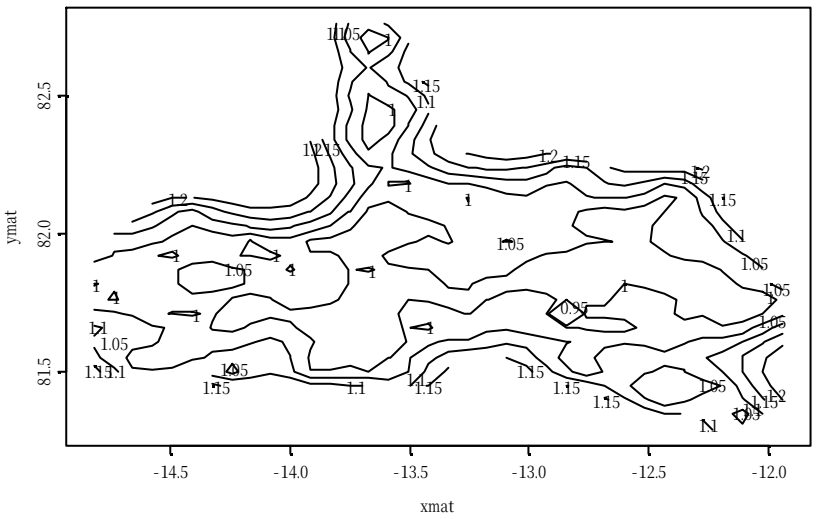


**図 4.24.** 対数を取ったホタテ貝の総捕獲数に対してクリギング予測を行なった際に生じた相対標準誤差の等高線図。

### 4.3.2 普遍クリギング

普遍クリギングは S+SPATIALSTATS では通常クリギングと同じ関数を使って行なうことができる；つまり、krige と predict.krige である。違いは krige のモデル式の指定をするところにある。普遍クリギングはデータに対して同時にトレンドモデルを当てはめるので、モデル式には多項式トレンド曲面を含んでいる必要がある。

普遍クリギングを行なう際には、データのトレンドモデルと、バリオグラムモデルまたは共分散モデルの両方に対する知識が必要になる。トレンドが存在していればバリオグラムを正確に推定することは出来ないし、空間的な相関があれば標準的なトレンド推定法はトレンドを正確に推定できないからである。よって、繰り返しを用いた方法を用いることが多い；共分散とトレンドモデルに初期値を与え、普遍クリギングを行なう。トレンド曲面からの残差（resid(obj)、ここで obj は krige が返すオブジェクト）を調べることで、トレンドが正確にモデリングされているかどうかを確かめることができる。この残差のバリオグラムもまた共分散モデルを改良するために用いることができる。そして再び普遍クリギングをこれらの改良されたモデルに対して実行するのである。

状況によっては、追加の情報により、データの一部を使ったバリオグラムの推定が可能である。適当な仮定をおけば、このモデルはデータ全体の普遍クリギングに使用することができる。このことは以下の例で述べる。

普遍クリギングの実例には3.2.1節で最初に登場した石炭データを用いる。探索的データ解析により、このデータは東西方向にトレンドをもつことが示されている。この章では以前このデータのトレンドを median polishing によって取り除いた。その結果得られた 1 方向バリオグラムによれば、東西方向の空間的相関は基本的に残っていなかった。よって、空間的相関は南北方向の経験バリオグラムに球型バリオグラムを当てはめてモデリングした。各パラメータは 4.2.2 節で非線形最小 2 乗法によって推定した。トレンドモデルは東西方向に見られたトレンドを元に指定する。石炭データに対する普遍クリギングを以下のようにして行なう：

```
> coal.krige <- krige(coal ~ loc(x, y) + x + x^2,
+       data = coal.ash, covfun = spher.cov,
+       range = 4.31, sill=0.14, nugget=0.89)
> coal.predict <- predict(coal.krige)
```

coal.krige オブジェクトの要約を見る：

```
>  coal.krige
Call:
krige(formula = coal ~ loc(x, y) + x + x^2,
         data = coal.ash, covfun = spher.cov,
         range = 4.31, sill = 0.14, nugget = 0.89)

Coefficients:
 constant        x          x^2
 9.633667 -1.30365  -0.1383046

Number of observations: 208
```

呼び出しの要約に加えて、推定されたトレンド曲面の係数が表示される。クリギング予測によるトレンド曲面を Trellis グラフィックスの関数 wireframe を用いてプロットする：

```
> wireframe(fit ~ x * y, data=coal.predict,
+       screen = list(z = 300, x = -60, y = 0),
+      drape=T)
```

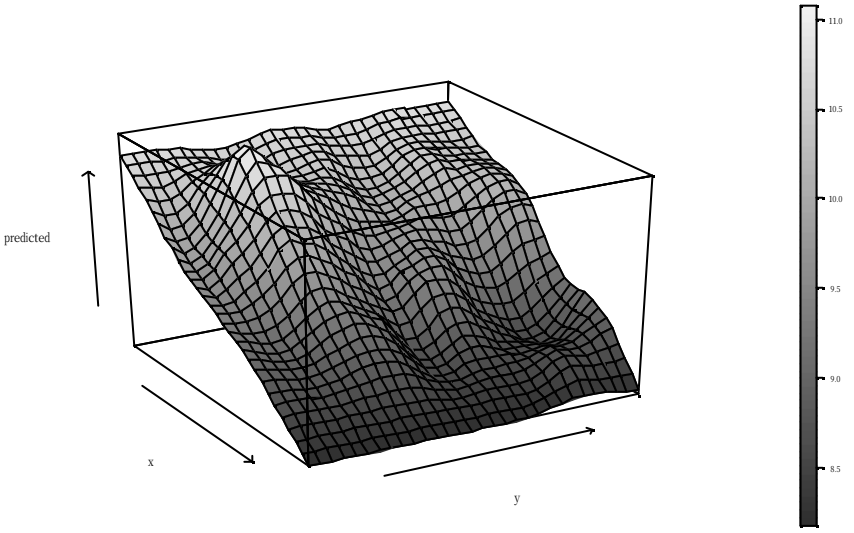


**図 4.25. 石炭含有率（%）の普遍クリギング予測による曲面。**

実行結果は 4.25 に示した。同様にしてクリギング標準予測誤差 se.fit も 3 次元曲面プロットすることができる。これは図 4.26 に示した。

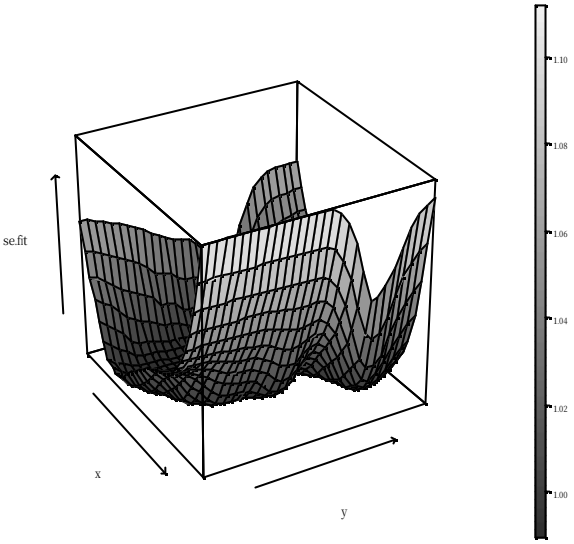


**図 4.26.** 普遍クリギング予測標準誤差の曲面プロット。

予測曲面にはモデリングした東西方向のトレンドがはっきり出ている。予測誤差の曲面には採集領域の形状が反映されている；予測を行なったグリッドの境界部分は採集地域から距離が離れているため、標準誤差が大きくなっている。4.3.1 節のホタテデータのクリギングで議論したように、異なる予測領域でもクリギングを行なってみて欲しい。

## 4.4 地理統計データのシミュレーション

シミュレーションデータはサンプリング方法を確かめたり、予測手法を評価したりする際に有益である。

S+SPATIALSTATS では関数 `rfsim` を使って、自己相関を与えた地理統計データを生成することができる。例えば、20×20 のグリッド上の確率場によるシミュレーションデータを発生させるには以下のようにする：

```
> xy20 <- expand.grid(x=seq(0.5,10,len=20),
+       y=seq(0.5,10,len=20))
> z.exp <- rfsim(xy20,covfun=exp.cov,range=2)
```

関数 `expand.grid` はグリッド上の位置を 400×2 のデータフレームにして返す。これらの位置は共分散構造と共に `rfsim` に入力される。デフォ

ルトでは、出力は正規確率場になる。*z.exp* の値の自己相関構造は指数型である。図 4.27 は以下のようにして得られたシミュレーションデータの 3 次元曲面プロットと自己相関構造を図示したものである：

```
> par(mfrow=c(2,1))
> persp(unique(xy20$x),unique(xy20$y),
+      matrix(z.exp,20))
> z.var <- variogram(z.exp~loc(xy20$x,xy20$y))
> plot(z.var,ylim=c(0,1.1))
> lines(z.var$distance,exp.vgram(
+      z.var$distance,range=2))
```

このシミュレーションデータは指数型の自己相関構造を持つ正規確率場の 1 つの実現値である。

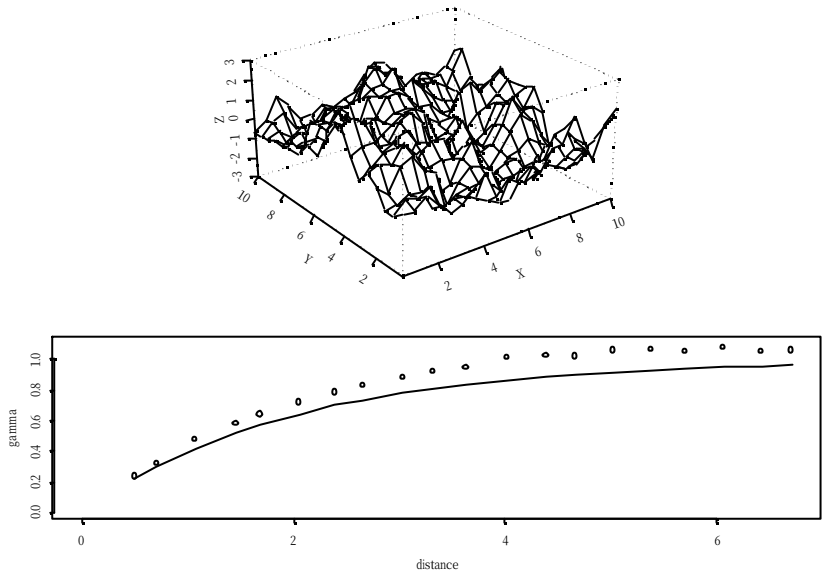


**図 4.27.** シミュレーションデータの鳥瞰図（上段）と、その経験バリオグラム（下段白丸）にシミュレーションを行なった際に使用した理論自己相関構造（下段実線）を描き加えたもの。

# 5 格子データの解析

この章では、格子データの解析とモデリングに使用される手続きについて紹介する。格子データは、加算個で近傍構造を持った空間領域の集合における確率過程から得られた実現値である。観測位置は規則的（等間隔のグリッド）な場合もあるし、不規則な場合もある。ある特定の点におけるデータはその領域全体を代表するものである。各サイトにおける観測値は連続値の場合も離散値の場合もある。例えば、サンプルデータフレーム sids には 1974 年から 1978 年の間にノースカロライナ州の各郡で乳幼児突然死症候群（SIDS）で死亡した乳幼児の数（Cressie and Chan, 1989）が収められている。各サイトの位置は郡庁（county seats）の座標である。このデータは不規則格子上の離散データであり、各サイトが郡全体を表わしている。格子データのより厳密な定義は第 1 章を参照。

格子データの空間的相関を持つ成分を S+SPATIALSTATS によってモデリングする前に、我々はデータに、定常性（定義は用語集を参照）と多変量正規性を仮定する。これはトレンドが除去されなければならないこと、そして分散を安定化させる、あるいは正規近似を行なう（もしくはその両方）ためには何らかの変換が必要になるかもしれないということを意味する。3.3 節で、これらの仮定が SIDS データに当てはまるかどうかを基本的な探索的データ解析（EDA）によって確かめた。本章では常にこの解析結果を使用し、さらなる解析を展開する。

本章では S+SPATIALSTATS による以下の手法について学習する：

- 空間近傍を定義する（5.1 節）。
- 格子データの空間的自己相関に対する検定（5.2 節）。
- 空間回帰を用いた格子データのモデリング（5.3 節）。
- 格子データのシミュレーション（5.4 節）。

# 5.1 空間近傍

格子データのモデリングは時系列データのモデリングの空間版である。時系列モデルは各時点の実現値を、過去の 1 時点または数時点との関係（自己相関の列）を元に予測するものである。空間過程は各領域の実現値を、近隣のもしくは近傍の（*neighboring*）領域との従属関係を一部用いて予測するモデルである。もし 2 つの領域が近傍（関係）にあれば、これらの領域で測定される確率過程は空間的な相関を持つことになる。**近傍**（*neighborhood*）構造を構築することが、格子データの解析の第 1 段階である。この結果が共分散構造を決定し、より一般的な線形回帰モデルの空間的相関を持つ成分のために使われることになる。

近傍は、境界線を共有している領域どうし、ある距離以内にある領域どうし、といった感じに定義される。例えば図 5.1 では、中心の郡は隣接した郡（影をつけた）領域だけを近傍としている。相関構造が距離に関して定義されるのであれば、薄い影をつけた郡も近傍集合に含まれることになる。

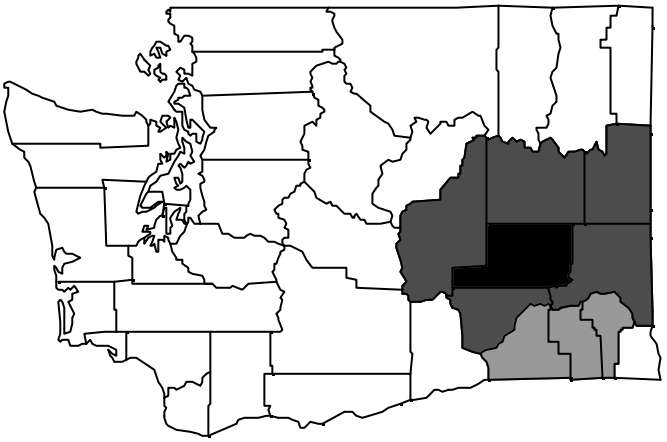

**図 5.1.** 考えられる近傍構造の例示に用いたワシントン州の地図。

近傍関係は対称的である必要はない。例えば、背景にあるプロセスが一方向にのみ流れるものである、あるいは非常に大きい領域は影響力があるけれど、小さい領域から影響を受けることはない、といった場合である。近

傍構造は格子データの共分散モデルにとって基本的な構造であるから、空間近傍を注意深く選択することは、解析にとって重要なステップである。

### 5.1.1 クラス "spatial.neighbor" のオブジェクト

S+SPATIALSTATS で格子データをモデリングする際、近傍情報はクラス"spatial.neighbor"のオブジェクトとして保存されている必要がある。S-PLUS オブジェクト sids.neighbor は SIDS データの近傍情報を含んでいる：

```
> sids.neighbor[1:15,]
Total number of spatial units =  100
(Matrix was NOT defined as symmetric)
   row.id col.id    weights matrix
 2      1     17 0.04351368      1
 3      1     19 0.04862620      1
 4      1     32 0.10268062      1
 5      1     41 0.20782813      1
 6      1     68 0.11500900      1
 8      2     14 0.17520402      1
 9      2     18 0.27140700      1
10      2     49 0.20882988      1
11      2     97 0.22700297      1
13      3      5 0.16797229      1
14      3     86 0.25458569      1
15      3     97 0.22660765      1
17      4     62 0.06865403      1
18      4     77 0.15401887      1
19      4     84 0.09565681      1
```

クラス"spatial.neighbor"のオブジェクトの最初の 2 列は、近傍関係にある領域のペアを表示している。これらは空間近傍（近接）行列の疎行列 (sparse matrix) 表現である。空間近傍行列は［行，列］（[row.id,col.id]）の組合わせが近傍のペアになるとき、列 weights で与えられる零でない値を持ち、その他の要素はすべて零になる行列である。例えば、インデックスが 1 である領域と 17 である領域は近傍の重み 0.0435 を持つ近傍である。近傍の重みの与え方は 5.1.4 節で議論する。列名 matrix は、1 つの空間近傍の中で複数の近傍や近傍の重みを指定す

る際に有効になる。2 つの近傍行列を使用する例については 5.1.5 節を参照。

クラス"spatial.neighbor"のオブジェクトは 2 つの属性を持つ。疎行列を扱うルーチンには空間領域の総数を表わす属性 nregion が必要である。近傍を 1 つも持たない領域もあるからである。属性 symmetry は近傍行列が対称的であるかどうかを表わす論理値である。オブジェクト sids.neighbor は非対称的と定義されており、この情報は行列計算を行う際に考慮される。対称性の議論は 5.1.3 節を参照。

### 5.1.2 テキストファイルから近傍情報を読み込む

近傍情報が S-PLUS の外部に保存されている場合、それを取り込むには関数 read.neighbor が必要になるだろう。近傍の数は各領域ごとに異なるので、一般に S-PLUS 関数 scan は使用できない。例えば、データがテキストファイル"neighbor.dat"に以下のように保存されているとしよう：

```
1 2 3 5
2 1 3 4 5
3 2 5 6 1
4 2 5
5 2 3 4 6 1
6 3 5
```

ここで、各行は領域とその近傍を表わしているものとしよう。例えば、領域 1 は領域 2、3、5 を近傍に持つ、など。各行は 1 つの固定した要素あるいは変量、領域番号と、長さに制約のない近傍のリストを持つ。このファイルを S-PLUS に読み込むには以下のようにする：

```
> readnhbr.test < - read.neighbor("neighbor.dat", keep=F)
> readnhbr.test
Total number of spatial units =  6
(Matrix was NOT defined as symmetric)
   row.id col.id weights matrix
 1      1      2       1      1
 2      1      3       1      1
 3      1      5       1      1
 4      2      1       1      1
```

```
 5     2     3     1     1
 6     2     4     1     1
 7     2     5     1     1
 8     3     2     1     1
 9     3     5     1     1
10     3     6     1     1
11     3     1     1     1
12     4     2     1     1
13     4     5     1     1
14     5     2     1     1
15     5     3     1     1
16     5     4     1     1
17     5     6     1     1
18     5     1     1     1
19     6     3     1     1
20     6     5     1     1
```

keep=F は領域番号だけを含む余計な成分を作るのを防ぐためのものである。今の場合、各領域は単に 1 から n（領域の総数）の番号が付けられているため、この操作は妥当である。しかし、領域番号がそのように付けられていない場合、近傍を修正する必要がある。領域番号をそのままにしたい場合は keep=T を用いる。領域番号の付け替えについては 5.1.3 節を参照。

read.neighbor が返す weights と matrix の列はデフォルトですべて値が 1 である；これらはモデルを計算する再に修正する必要がある場合もある。

近傍情報を含むファイルに、それ以外のサイトに固有なデータが含まれている場合でも read.neighbor を使うことができる。例えば、テキストファイル"sids.dat"の最初の 5 列には各サイトに固有なデータが入っており、残りのフィールドに各サイトの近傍が入っているとする：

```
1   278   151    4672    13     17 19 32 41 68
2   179   142    1333     0     14 18 49 97
3   183   182     487     0     5 86 97
4   240    75    1570    15     62 77 84 90
5   164   176    1091     1     3 95 97
```

```
6   138   154     781     0     12 14 56 61 95 100
7   406   118    2692     7     59 74 94
8   411   148    1324     6     21 46 59 94
.
.
100 120   142     770     0     6 11 56 58 61
```

**注意：このデータは** S+SPATIALSTATS **には含まれていない。**

**このファイルを以下のようにして読み込む：**

```
> sids.test <- read.neighbor("sids.dat",
+      field.name=c("id","easting", "northing",
+      "births", "sid"), region.id=1)
> sids.test$data[1:8,]
  id easting northing births sid
1  1     278      151    4672  13
2  2     179      142    1333   0
3  3     183      182     487   0
4  4     240       75    1570  15
5  5     164      176    1091   1
6  6     138      154     781   0
7  7     406      118    2692   7
8  8     411      148    1324   6
> sids.test$ngbr[1:10,]
Total number of spatial units =  100
(Matrix was NOT defined as symmetric)
   row.id col.id weights matrix
 1  1      17     1      1
 2  1      19     1      1
 3  1      32     1      1
 4  1      41     1      1
 5  1      68     1      1
 6  2      14     1      1
 7  2      18     1      1
 8  2      49     1      1
 9  2      97     1      1
10  3       5     1      1
```

このファイルは近傍情報の他に付加的なデータを含んでいるので、デフォルトである keep=T により、付加的なデータはデータフレーム$data に保存される。領域番号以外の固定長レコードがある場合は引数 field.names が必要である。可変長の近傍のフィールドはこの引数に入っていないことに注意せよ。引数 region.id は field.names の中で領域番号を表わしている列を明示するためのものである。デフォルトでは field.names の最後の列（固定長変量のうち最後の変量）が領域番号になる。

read.neighbor はデフォルトではすべてのフィールドは数値であり、可変長の近傍のリストは、各行の最後に現われると仮定している。引数 all.numeric、char、first.neighbor、を使用するとこれらの仮定を変更することができる。read.neighbor の詳細と使用例はヘルプファイルを参照。

### 5.1.3 空間近傍を見つける

データを S-PLUS に取り込む際に空間近傍をまだ定義していない場合、それを行うための関数がいくつか用意されている。データが規則的な格子状であれば、neighbor.grid を使ってクラス"spatial.neighbor"のオブジェクトを生成することができる。不規則な格子状であれば、代わりに関数 find.neighbor を使えばよい。

3 × 3 のグリッドに、1 次の近傍関係を入れたクラス "spatial.neighbor"のオブジェクトを生成するには以下のようにする：

```
> ng <- neighbor.grid(nrow=3, ncol=3,
+      neighbor.type="first.order")
> ng[1:10,]
Total number of spatial units =  9
(Matrix was NOT defined as symmetric)
   row.id col.id weights matrix
 1  1      2      1      1
 2  1      4      1      1
 3  2      1      1      1
 4  2      3      1      1
 5  2      5      1      1
```

```
 6   3      2      1      1
 7   3      6      1      1
 8   4      1      1      1
 9   4      5      1      1
10   4      7      1      1
```

グリッドは列にそって下へと順番に番号付けされており、"first.order"は近傍を各領域の上下左右とする手法である（*rook* パターンと呼ばれることもある）。

neighbor.grid には他にも 4 つの標準的な近傍のタイプが利用できる：1 次の近傍に斜めの方向を入れた"second.order"（*queen* パターン）；斜めの方向**のみ**を近傍とする"diagonal"（*bishop* パターン）；6 角グリッドに対する"hexagonal.in"と"hexagonal.out"。これらのパターンの詳細についてはヘルプファイルを参照。その他にも、ユーザがパターンと重みを独自のものに定義することも可能である。以下の例では、次のパターンと重みを与えた空間近傍を定義している：

```
0   .5   0
1    X   1
0   .5   0
```

1. まず、位置(row1,col1)と(row2,col2)との間の重みを返す関数を書く：

```
> my.weight.fun <- function(row1,col1,row2,col2){
+   if(abs(row1 - row2) == 1)
+          if(col1 == col2)
+                 return(0.5)
+          else return(0)
+   else if(row1 == row2)
+          if(abs(col1 - col2) == 1)
+                 return(1)
+          else return(0)
+   else return(0)
+ }
```

2. neighbor.grid の中で neighbor.type="user" とし、weight.fun にステップ 1 で作成した関数を与える：

```
> ng2 <- neighbor.grid(nrow=5, ncol=5,
```

```
+      neighbor.type="user", weight.fun=my.weight.fun,
+      max.horiz.dist=1, max.vert.dist=1)
> ng2[1:10,]
Total number of spatial units =  25
(Matrix was NOT defined as symmetric)
   row.id col.id weights matrix
 1  1      2      0.5    1
 2  1      6      1.0    1
 3  2      1      0.5    1
 4  2      3      0.5    1
 5  2      7      1.0    1
 6  3      2      0.5    1
 7  3      4      0.5    1
 8  3      8      1.0    1
 9  4      3      0.5    1
10  4      5      0.5    1
```

省略可能な引数 max.horiz.dist と max.vert.dist は近傍の候補を探す範囲を限定し、大きなグリッドに対する計算量を減らすために用いる。

上で生成した ng と ng2 はいずれも対称的な近傍関係を持っていた。しかし、どちらの空間近傍の属性 symmetry もデフォルトで FALSE に設定されている（"Matrix was NOT defined as symmetric"と表示される）。以下では，ng2 に対して S+SPATIALSTATS 関数 spatial.condense を使用してこの属性を変更し、冗長な情報を消去している：

```
> ng2 <- spatial.condense(ng2, symmetry=T)
> ng2[1:10,]
Total number of spatial units =  25
(Matrix defined as symmetric)
   row.id col.id weights matrix
 3  2      1      0.5    1
 6  3      2      0.5    1
 9  4      3      0.5    1
12  5      4      0.5    1
14  6      1      1.0    1
17  7      2      1.0    1
```

```
18  7      6      0.5    1
21  8      3      1.0    1
22  8      7      0.5    1
25  9      4      1.0    1
```

**警告：** spatial.condense を使う際、近傍行列が実際には**対称的でない**にも関わらず symmetric=T を指定した場合、モデルの計算がうまく行かない場合がある。

不規則な格子上で、k 個の最近接近傍やある与えられた距離内にあるすべての近傍を見つける場合、find.neighbor を使用する。find.neighbor を使用するには、初めに関数 quad.tree を使って四分樹(quad tree)を作る必要がある。四分樹は並べ替えられた行列で、最近接近傍を探索するのに最も効果的な順序を与えるものである。この手続きを、SIDS データに対して行ってみる。Cressie (1993, p. 385)によるこのデータセットの解析では、互いの郡庁が 30 マイル以内にあるものを近傍と考えている。互いが 30 マイル以内にある近傍を探すには以下のようにする：

```
> sids.place <- cbind(sids$easting,sids$northing)
> sids.quad <- quad.tree(sids.place)
> sids.nhbr <- find.neighbor(x=sids.place,
+      quadtree=sids.quad, max.dist=30)
> sids.nhbr[1:10,]
   index1 index2   distances
 1      1      1    0.00000
 2      1     41   20.02498
 3      1     17   24.18677
 4      1     68   16.00000
 5      1     32   28.44293
 6      1     19   27.29469
 7      2      2    0.00000
 8      2     97   15.13275
 9      2     49   18.86796
10      2     14   21.00000
```

**注意：**この結果には、Cressie [p.244, 1993]で使われ、空間近傍 spatial.neighbor に含まれているものより近傍のペアが 1 組多く含まれている。領域 74 と 98 との距離は 29.83 であり、近傍に含まれるべきである。しかし、Cressie の結果と一貫性を持たせるためにこのペアは除

外した。

find.neighbor の結果得られるものは近傍のペアとそれらの距離の 3 列からなるデータフレームである。この情報は、5.1.3 節でクラス "spatial.neighbor"のオブジェクトを作るのに用いられる。

今のままでは、自分自身を領域の一部と見なしている。次に進む前にこの無駄な情報を消去しておく：

```
> sids.nhbr <- sids.nhbr[sids.nhbr[,3] != 0,]
```

### クラス"spatial.neighbor"のオブジェクトの作成

S+SPATIALSTATS で格子データの解析を行うには、近傍情報はクラス "spatial.neighbor"のオブジェクトとして保存されていなければならない。S-PLUS に保存された近傍情報からこのオブジェクトを作成するには、関数 spatial.neighbor を用いて以下のようにする:

```
> sids.snhbr <- spatial.neighbor(row.id=sids.nhbr[,1],
+     col.id=sids.nhbr[,2])
> sids.snhbr[1:10,]
Total number of spatial units =  100
(Matrix was NOT defined as symmetric)
   row.id col.id weights matrix
 2   1      41     1      1
 3   1      17     1      1
 4   1      68     1      1
 5   1      32     1      1
 6   1      19     1      1
 8   2      97     1      1
 9   2      49     1      1
10   2      14     1      1
11   2      18     1      1
13   3      97     1      1
```

引数 row.id と col.id は、find.neighbor の出力のように近傍行列の行と列の番号でなければならない。重複を除いたベクトル row.id の成分が、1 から空間領域の数までの番号に**等しくない**場合は、以下の 2 つ

のうちいずれかに該当する：

1. データセットに孤島、つまりデータはあるが近傍を持たない領域が存在する。

2. 近傍行列として適当でない行番号と列番号を領域番号に使用している。

空間近傍の属性 nregion が近傍行列の領域数と異なる場合は後者の場合である（この属性は上の出力結果において、空間領域の数として表示される）。この場合、関数 check.islands を引数 remap=T を指定して使用し、空間近傍を修正する。check.islands に関する詳細はヘルプファイルを参照。

weights の列はデフォルトで 1（すべての近傍に対して等しい重み）になっている。しかし、これらの重みの定義は注意深く行うことを薦める。近傍の重みの選び方は 5.1.4 節で議論する。

spatial.neighbor を呼び出した際に対称性について特定しなかった場合、対称性は仮定されない（どちらの場合も計算は正しく行われる）。しかし、対称的な近傍関係を持つ大きな行列の場合、spatial.neighbor の呼び出しの際に任意属性 symmetric=T を与えると、冗長性を省き、モデリングの手続き中の計算速度を向上させることができる。この場合、対称なペアのうちどちらか片方だけを入力するだけでよい。

**警告：**空間近傍オブジェクトを作成する際、近傍行列が実際には**対称的でない**にも関わらず symmetric=T を指定した場合、モデルの計算がうまく行かない場合がある。

もし近傍行列が対称的かどうか分からない場合は、以下のように S-PLUS 関数 is.Hermitian を使用する：

```
> is.Hermitian(spatial.weights(sids.snhbr), tol=0)

[1] T
```

S+SPATIALSTATS 関数 spatial.weights はクラス "spatial.neighbor"のオブジェクトを数値型の行列に変換する。対称性を確かめるためには引数 tol=0 が必要である。sids.snhbr に入って

いるノースカロライナ州の各郡の近傍関係は対称的である。これは近傍の重みを加えたときに変化する（5.1.4 節参照）。

関数"spatial.neighbor"は疎行列表現でない近傍行列にも使用できる。例えば：

```
> n.mat <- matrix(data=c(0,1,.5,0,1,0,.5,0,0,.5,0,1,
+     0,.5,1,0), nrow=4)
> n.mat
     [,1] [,2] [,3] [,4]
[1,]  0.0  1.0  0.0  0.0
[2,]  1.0  0.0  0.5  0.5
[3,]  0.5  0.5  0.0  1.0
[4,]  0.0  0.0  1.0  0.0
> spatial.neighbor(neighbor.matrix=n.mat)
Total number of spatial units =  4
(Matrix was NOT defined as symmetric)
  row.id col.id weights matrix
1   2      1      1.0    1
2   3      1      0.5    1
3   1      2      1.0    1
4   3      2      0.5    1
5   2      3      0.5    1
6   4      3      1.0    1
7   2      4      0.5    1
8   3      4      1.0    1
```

### 5.1.4 近傍の重み

クラス"spatial.neighbor"の weights 成分は近傍どうしの相関の強さを決定する。データのタイプに応じて重みの与え方は様々である。重みを与えるには、領域どうしの空間的共分散の**範囲**（*range*）や**強さ**（*intensity*）に対する**事前**（*a priori*）知識が必要になる（Griffith, 1995）。よく用いられる手法には、行標準化、共通境界の長さ、距離関数などがある。**行標準化**（*row-standardization*）は、共分散を各領域の近傍数に応じて均等に分配する方法である。ある領域（行）が 5 つの近傍を持つとすると、各近傍の組（零でない列）の重みは 1/5 となる。各領域（行）の重みは他の領域とは関係なく分配されるため、近傍関係は非対称になる。

**警告：負の重みを与えることが望ましい場合もあるかも知れないが、S+SPATIALSTATSのこの版では負の重みを与えることはできない。**

**近傍間の重みを与える作業は格子データの解析にとって重要なステップである。**Griffith (1995)**によれば、重みの与え方を誤ると、特にサンプル数の少ない場合、モデルの標準誤差を増大させ、相関の推定に偏りを生じさせることになる。相関の影響範囲の指定を誤ることは相関関係の関数の型に違うものを与えるよりももっと深刻である。また、近傍行列を少なく見積もる（近傍の数を少なく見積もる）ほうが、多く見積もるよりもよい（**Griffith, 1995**）。近傍行列の変化がモデリング結果にいかに敏感に反映されるかを見るために、複数の重みを与えてみることを薦める。**

S-PLUS **オブジェクト** `sids.neighbor` **の近傍の重みは** Cressie (1993, p.557)**で使用された、距離に応じて減少する相関関数によって与えられている。近傍のペアどうしの相関は以下のように推定されている：**

$$c_{ij} = \begin{cases} r\left(d_{ij}^{-k} / C(k)\right)\left(n_j / n_i\right)^{1/2} & j \in N_i \\ 0 & \text{otherwise.} \end{cases}$$

**ここで** ρ **は未知の定数、**$d_{ij}$ **は近傍間の距離、**$n_i$ **は郡** $i$ **における出生数、**$C(k) = \max\{d_{ij}^{-k} : j \in N_i; i = 1,\cdots,n\}$**（スケーリング因子）、**$N_i$ **は郡** $i$ **の近傍集合**

$$g_{ij} = \begin{cases} \left(\min\left\{d_{ij} : i = 1,\ldots,n\right\} / d_{ij}\right)\left(n_j / n_i\right)^{1/2} & j \in N_i \\ 0 & \text{otherwise.} \end{cases} \qquad (5.1)$$

**である。近傍の重みは** $c_{ij}$ ρ **で与える。**Cressie **は様々な** $k$ **に対してこれを試し、**$k = 1$**が最もよい結果出すモデルであることを発見した。ゆえに、**`sids.neighbor` **には以下の重みが収録されている：**

**この重みを** S-PLUS **で定義する：**

```
> dij <- sids.nhbr[,3]        #  近傍どうしの距離
> n <- sids$births  # 各郡の出生数
> el1 <- min(dij)/dij
> el2<- sqrt(n[sids.snhbr$col.id]/n[sids.snhbr$row.id])
> sids.snhbr$weights <- el1*el2
> sids.snhbr[1:10,]
Total number of spatial units =  100
(Matrix was NOT defined as symmetric)
```

```
   row.id col.id       weights matrix
 2      1     41     0.20782813      1
 3      1     17     0.04351368      1
 4      1     68     0.11500900      1
 5      1     32     0.10268062      1
 6      1     19     0.04862620      1
 8      2     97     0.22700297      1
 9      2     49     0.20882988      1
10      2     14     0.17520402      1
11      2     18     0.27140700      1
13      3     97     0.22660765      1
```

sids.neighbor には上で作成したオブジェクト sids.snhbr と同じ情報が含まれている。sids.neighbor の対称性をチェックするには以下のようにする：

```
> is.Hermitian(spatial.weights(sids.neighbor),tol=0)
[1] F
```

この重みの与え方は非対称な近傍行列を生成している。

### 5.1.5 複数の近傍行列を使う

クラス"spatial.neighbor"のオブジェクトの matrix 成分は、タイプの異なる近傍モデルを並行して使う場合に用いられる。例えば、0 から 20 マイル離れた近傍と、20 から 40 マイル離れた近傍の相関には、異なる水準を期待するだろう。この差異を考慮した新しい空間近傍を SIDS データに作成する：

1. 上記の範囲に入る近傍を探す：

```
> sids.20nhbr <- find.neighbor(x = sids.place,
+     quadtree=sids.quad, max.dist=20)
> sids.40nhbr <- find.neighbor(x = sids.place,
+     quadtree=sids.quad, max.dist=40)
```

2. 自分自身と冗長な情報を近傍から除去する：

```
> sids.20.nhbr <- sids.20nhbr[sids.20nhbr[,3] !=0,]
> sids.40.nhbr <- sids.40nhbr[sids.40nhbr[,3] >20,]
```

3. 2 つの近傍のリストを結合する：

```
> row.id2040 <- c(sids.20nhbr[,1],sids.40nhbr[,1])
> col.id2040 <- c(sids.20nhbr[,2],sids.40nhbr[,2])
```

4. 0 から 20 マイルの近傍が matrix.id=1、20 から 40 マイルの近傍が matrix.id=2 となるような matrix.id 列を作る：

```
> dim(sids.20nhbr)
[1] 140 3
> dim(sids.40nhbr)
[1] 590 3
> matrix.id <- c(rep(1,140),rep(2,590))
```

5. 空間近傍オブジェクトを生成する：

```
> sids.2nhbr <- spatial.neighbor(row.id=row.id2040,
+       col.id=col.id2040, matrix.id=matrix.id)
```

6. 各近傍に距離に応じて減少する重みを与える：

```
> dij2 <- c(sids.20nhbr[,3], sids.40nhbr[,3])
> elem12 <- min(dij2)/dij2
> nvec <- sids$births
> col.id <- sids.2nhbr$col.id
> row.id <- sids.2nhbr$row.id
> elem22 <- sqrt(nvec[col.id]/nvec[row.id])
> sids.2nhbr$weights <- elem12*elem22
```

この空間近傍は 5.2 節では空間的自己相関の検定に、5.3.2 節では空間回帰モデルの当てはめに用いる。

## 5.2 空間的自己相関

プロセスに空間的自己相関がある場合、空間モデリングが必要になるかもしれない。空間的自己相関の検定は、空間モデリングを行うかどうかを決定する探索的な技法としても使用することができる。標準的な重回帰モデルあるいはトレンド局面モデルの残差に対する検定としても用いることはできるが、あまり勧められない（Ripley, 1981, p.99）。帰無仮説は自己相関がないことであり、対立仮説は重み付き近傍行列によって定義される。よって、近傍と重みの選択は検定にも大きな影響を与える。いくつか

の異なるシナリオを用意し、検定を行うのが望ましい。空間的自己相関の計算には一定平均と一定分散が仮定されている。プロセスにトレンドや異なる分散が含まれている場合、導かれた結論は慎重に用いる必要がある。

S+SPATIALSTATS 関数 `spatial.cor` は空間的自己相関の 2 つの代表的尺度ある Moran 統計量と Geary 統計量を計算する。他に、ユーザが自己相関の尺度を定義するオプションもある。これらの尺度に対する式と詳しい定義は Cliff and Ord (1981, p.17)やヘルプを参照。

SIDS の発生は分散一定ではなさそうである。出生数の低い郡ほど、分散が高くなる傾向にありそうだからである。発生数は Freedman-Tukey 平方根変換（`sids$sid.ft`）を使って変換された（3.3 節参照）。Cressie (1993)は、出生数の平方根にこの変換された値を掛けた値がほぼ一定分散を持つと見なした。この変換はポアソン分布に従う確率変数の平方根に近似できるため、妥当な仮定である。

1974 年から 1978 年にかけての SIDS 死亡者数の自己相関を、Moran 統計量と `sids.neighbor` を使って検定してみる：

```
> sids.cor1 <- spatial.cor(
+       sids$sid.ft*sqrt(sids$births),
+       neighbor=sids.neighbor, statistic="moran",
+       sampling="free", npermutes=1000)
```

分散は属性 `sampling` に応じた 2 種類のうち、いずれかの方法で計算される。メソッド"`free`"を使用すると正規性を仮定する―データは単一の、または複数の正規分布からの独立な観測値であるという仮定である。メソッド"`nonfree`"を使用すると、ランダム化を行って分散を計算する。分布の型は仮定しない。これは復元抽出（"`free`"）と非復元抽出（"`nonfree`"）に相当する。Freedman-Tukey 変換を行った SIDS データの分布をチェックした際（3.3 節）、正規性を仮定できない材料は何もなかったため、SIDS データにはメソッド"`free`"が妥当である。

省略可能な引数 `npermutes` は標本自己相関を計算するのに用いるデータベクトルのランダムな順列の数を指定するために使用する。この順列の分布によって、計算された係数が有意かどうかを検定する。`npermutes` が与えられない場合、デフォルトは 0 である。

結果は以下の通りである：

```
> sids.cor1
Spatial Correlation Estimate

Statistic = "moran" Sampling = "free"

Correlation  =  0.2287
Variance     =  0.00772
Std. Error   =  0.08786

Normal statistic =  2.717
Normal p-value (2-sided) =  0.006579

Null Hypothesis:  No spatial autocorrelation

Summary of the permutation-correlations :
    Min.  1st Qu.   Median     Mean 3rd Qu.   Max.
 -0.3114 -0.06815 -0.01771 -0.01301 0.04112 0.2669

permutation p-value = 0.002
```

ほとんどの場合、Moran 統計量は帰無仮説のもとでは、漸近的に平均が -1/($n$ - 1)の正規分布に従う。ここで $n$ は領域の数。統計量の値の範囲と分散は近傍の重みの関数である。相関係数が有意であるかどうかを評価するには、上の出力結果における正規分布の両側確率点や順列の確率点のそれと比較すれば良い。どちらの方法でも SIDS データは有意である。

Geary の相関係数を使った結果との比較を行う：

```
> sids.cor2<-spatial.cor(
+       sids$sid.ft*sqrt(sids$births),
+       neighbor=sids.neighbor, sampling = "free",
+       statistic = "geary", npermutes=1000)
> sids.cor2

Spatial Correlation Estimate

Statistic = "geary" Sampling = "free"
```

```
Correlation  =  0.7439
Variance     =  0.0129
Std. Error   =  0.1136

Normal statistic =  -2.255
Normal p-value (2-sided) =  0.02414

Null Hypothesis:  No spatial autocorrelation

Summary of the permutation-correlations :
   Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
 0.6756   0.9186  0.9941   1.003    1.072   1.677

permutation p-value = 0.011
```

Geary統計量もまた漸近的に正規分布に従う。範囲や分散は近傍の重みに依存する。しかし、Moran 統計量とは重要な違いがある：Geary 統計量の平均は帰無仮説のもとでは 1 であり、決して負にならない。また、平均が**低い**値（0 から 1 の間）を取ることは**正の**空間的自己相関があることを示す。従って、**正の**自己相関の存在が有意であることが Geary 統計量によっても示された。

spatial.cor は複数の近傍行列を持つ空間近傍に対しても用いることができる。5.1.5 節で作成した、2 つの近傍行列を持つ空間近傍に対する Moran 統計量を以下のようにして求める。

```
> spatial.cor(sids$sid.ft*sqrt(sids$births),
+       neighbor=sids.2nhbr, sampling="free",
+       statistic="moran", npermute=1000)
     Spatial Correlation Estimate

Statistic = "moran" Sampling = "free"

Correlation =  0.1995 0.2285
Variance    =  0.02396  0.003349
Std. Error  =  0.1548  0.05787
```

```
Normal statistic =  1.354 4.122
Normal p-value (2-sided) =  0.1758   3.753e-5

Null Hypothesis:  No spatial autocorrelation

Summary of the permutation-correlations :
              x1.1       x1.2
   Min.   -0.66700   -0.19910
1st Qu.   -0.10720   -0.05476
 Median   -0.01507   -0.01509
   Mean   -0.01468   -0.01316
3rd Qu.    0.07788    0.02744
   Max.    0.71910    0.19270

permutation p-value = 0.073 0.000
```

互いの距離が 20 から 40 マイル離れた郡どうしを近傍とした 2 番目の近傍行列は有意な正の自己相関を持っている。0 から 20 マイル離れた郡どうしで定義した近傍行列はきわどいが有意な自己相関を持つという結果になった。これは 0 から 20 マイルの近傍行列の近傍が比較的少ない（140）ことが影響しているのかもしれない。

空間的自己相関の検定結果は注意して扱うべきである。第 1 の理由は、近傍とその重みの選択が Moran、Geary の両統計量の値を決定するからである。有意でない結果が出たということは、仮定した近傍構造において有意な自己相関がない、ということを意味する。Griffith (1995)は、近傍の重みの与え方を誤ると、空間的自己相関の検定における統計的な検出力が低下し、そのような場合、Moran 統計量の検出力がわずかに上回る、と報告している。近傍関係として他にも可能なものがあるならば、異なったスケールにある見えないパターンを見落とさないために、それらに対しても自己相関の検定を行う。第 2 の理由は、データにトレンドが含まれていることにより正の自己相関が有意となることがあり得るからである。3.3 節の探索的データ解析で作成した塗り分け地図により、SIDS データには空間的なトレンドがあることが明らかになっている。この場合、位置に関する回帰モデルに自己相関共分散モデルを適用するのが適当である。

自己相関統計量とサンプリング法の選択に関する詳しい議論は Haining

（1990）か Cliff and Ord （1981）を参照。

## 5.3 空間回帰モデル

格子データをモデリングする際、我々は 2 つのレベルの変動（位置や他の説明変数による平均の大域変動と、近傍どうしの相互作用による局所変動）に着目する。S+SPATIALSTATS には以下の形の空間回帰モデルを当てはめるための関数群が含まれている：

$$Z_i = m_i + d$$

ここで $Z_i$ はサイト $i$ における確率過程である；$\mu_i$ はサイト $i$ の平均で、一定か、互いに相関を持つ線形モデルである；$\delta \sim N(0,\Sigma)$；$\Sigma$ はすべてのサイトの確率変数の共分散行列である。一定でないような $\mu$ は、S+SPATIALSTATS の空間モデリングでは、線形モデルとしてモデリングすることが可能である。局所変動は、$\Sigma$ に自己回帰モデルや移動平均モデルを当てはめてモデリングする。これら 2 つのモデルののパラメータは相互に影響し合うため、モデリングは対話的に行われる。

### 5.3.1 共分散モデル族

S+SPATIALSTATS では共分散構造には 3 つの選択が可能である。それぞれ条件付き空間自己回帰（CAR, conditional spatial autoregression）モデル、同時空間自己回帰（SAR, simultaneous spatial autoregression）モデル、移動平均（MA, moving average）モデルである。これらのモデルはすべて多変量正規性を仮定しており、違いは分散共分散行列の選択に現れる。

分散共分散行列の 3 つのモデルは以下の通り：

$$CAR: \quad \Sigma = (I - rN)^{-1} D s^2$$
$$SAR: \quad \Sigma = \left[(I - rN)^T D^{-1} (I - rN)\right]^{-1} s^2$$
$$MA: \quad \Sigma = (I + rN) D (I + rN)^T s^2$$

ここで $\rho$ と $\sigma$ は空間回帰で推定するスカラーパラメータ、$N$ は重み付き近

傍行列、$D$ は周辺分布の不均一分散を考慮するための対角行列である。

**注意：**行列 $D$ は重み行列と呼ばれることがある。これは上の $N$ の要素である近傍の重みと混同してはならない。本書では、$N$ の要素を近傍の重みと呼ぶことにする。

CAR モデルと SAR モデルが時系列の自己回帰モデルに相当する。SAR モデルの残差は近傍のデータ値と相関を持ち、パラメータの最小 2 乗推定量が一致性を持たない（Cressie, 1993, p. 408）。CAR モデルにこの問題は生じないため、分散共分散行列が対称行列である限りは（この事が条件なのであるが）こちらのほうが望ましい。SIDS データで見たように、分散共分散行列を対称化するために行列 $D$ を用いる場合がある。MA モデルは時系列の移動平均モデルと同じである。CAR、SAR、MA の各モデルの使用に関する詳細は Cliff and Ord (1981)、Cressie (1993)、Haining (1990)を参照。

### 5.3.2 空間線形モデルの当てはめ

S+SPATIALSTATS 関数 `slm` は、一般化最小 2 乗回帰を使って空間に依存した線形モデルを当てはめる。S-PLUS 関数の `lm` や `glm` に似た働きをするが、これらに加えて空間的共分散のモデリングも行う。

`slm` の大域成分もしくは線形モデル成分は単純な線形モデルであったり、データの位置を基にした多項式を用いてトレンドをモデリングするトレンド曲面モデル（*trend surface model*）に含まれたりする。例えば、$(x_i, y_i)$ によって表わされるデータの位置に関する 2 次のトレンド表面モデルは以下の形で表わされる：

$$m_i = b_{10}x_i + b_{20}x_i^2 + b_{11}x_i y_i + b_{01}y_i + b_{02}y_i^2,$$

ここで各 β は推定する係数である。`slm` では、位置ベクトルの適切な関数を説明変数としてモデル式の中で用いることでこれを指定する。

`slm` の中で μ のモデリングを行うのではなく、S-PLUS の他の関数で平均をモデリングした後、相関のある誤差を `slm` でモデリングしてもよい。例えば、この方法は空間ラグモデルを当てはめる際に必要となる。空間ラグモデルは、トレンドあるいは各位置の平均値が近傍領域の平均値の加重平均であるようなモデルである。説明変数は、興味対象の変量に行標準化した重み近傍行列を掛けたものになる。興味対象の変量は従属変量そのも

のの場合もあれば、プロセスに影響を与える他の独立変量である場合もある。

SIDS データセットを例にとって、Freeman-Tukey 変換を行った SIDS 死亡者数の大域変動に対する 4 つの簡単なモデルについて見てゆく：**トレンドなし**（*null*）モデル（一定平均を仮定）；平均値に線形予測因子（有色人種の出生数）を使った**人種**（*race*）モデル；領域のクラスタごとに 12 種類の平均値を与える**グループ**（*group*）モデル；人種モデルに 5.1.5 節の 2 つの近傍行列を併用するモデル、の 4 つである。当てはまりのよさの比較は 5.3.3 節で行う。

共分散に関しては CAR モデルを用いることにしたいのだが、我々が定めた近傍の重みは対照的ではなかった（5.1.4 節）。出生数の高い郡は、近傍の出生数の低い郡に、より強い影響を及ぼす。3.3 節の様に、SIDS 死亡者数の分散もまた出生数と相関を持つことを思い出すと：

$$\begin{aligned}\mathrm{var}\left(Z_i \mid Z_j : j \in N_i\right) &= s_i^2 \\ &= s^2 / n_i .\end{aligned}$$

もし対角行列 $D$ を $d_i = 1 / n_i$ として用いるならば、分散を補正し、共分散行列を対称化できる。共分散行列が対称であることを見るために、最初に式 5.1 を見てみる：

$$g_{ij} * n_i = g_{ji} * n_j$$

このことは行列 $(D^{-1} * N)$ が対称的であることを示しており、CAR モデルの共分散行列も対称的になる。

1974 年から 1978 年の SIDS データに対してトレンドなし CAR モデルを仮定した場合のパラメータを推定してみる：

```
> sids.nullslm <- slm(sid.ft ~ 1, cov.family = CAR,
+       data = sids, subset= -4, spatial.arglist=
+       list(neighbor=sids.neighbor, region.id = 1:100,
+       weights = 1/sids$births))
```

`slm` を使うには、まず線形モデル式を与えなければならない。もし、すでに平均を引き去っている場合はモデル式を $y \sim -1$ として平均を 0 に固定する。次に、共分散モデル族を選択し、共分散モデルに必要となる引数のリスト `spatial.arglist` を与える。最も基本的な `spatial.arglist` は空間近傍だけを含むものである。省略可能な引数 `subset` を与えると、

モデルの当てはめに悪影響を及ぼすと思われる外れ値を取り除くことができる。SIDS データに対しては、3.3節で郡 4 を外れ値とした。subset を用いる時は、spatial.arglist の中で引数 region.id を指定して領域の総数を与える必要がある。省略可能な引数 weight は以前議論した対角行列 $D$ である。

当てはめの結果を表示させるには以下のようにする：

```
> sids.nullslm
Call:
slm(formula = sid.ft ~ 1, cov.family = CAR, data = sids,
    subset = -4, spatial.arglist = list(neighbor =
    sids.neighbor, region.id = 1:100, weights = 1/sids$
    births))

Coefficients:
 (Intercept)
   2.839042

Degrees of freedom: 99 total; 97 residual

sigma^2 =  1457.617
rho =  0.8334194

Iterations =  12
Gradient norm =  0.00001014913
Log-likelihood =  -211.8542
Convergence:  RELATIVE FUNCTION CONVERGENCE
```

係数の推定値（一定平均）は 2.839 である。各パラメータの推定値は $\sigma^2$ が 1457.617、$\rho$ が 0.8334 である。このモデルの対数尤度は-211.8542 である。

3.3 節で我々は、ノースカロライナ州において有色人種の出生数と SIDS 死亡者数の間に相関があることを確かめた（図 3.31 参照）。これらの間には正の相関が見られたため、Freeman-Tukey 変換後の有色人種の出生数を線形説明変数とした空間回帰モデルを当てはめてみる：

```
> sids.raceslm <- slm(sid.ft ~ nwbirths.ft,
```

```
+      cov.family = CAR, data = sids, subset = -4,
+      spatial.arglist = list(neighbor = sids.neighbor,
+      region.id = 1:100, weights = 1/sids$births))
> summary(sids.raceslm)
Call:
slm(formula = sid.ft ~ nwbirths.ft, cov.family = CAR,
data = sids, subset = -4, spatial.arglist =
list(neighbor = sids.neighbor, region.id = 1:100,
        weights = 1/sids$births))
Residuals:
  Min     1Q Median    3Q   Max
 -106 -18.79   7.01 26.27 77.8

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept) 1.6456  0.2385     6.8990   0.0000
nwbirths.ft 0.0345  0.0066     5.2068   0.0000

Residual standard error: 34.525 on 96 degrees of freedom

Variance-Covariance Matrix of Coefficients
            (Intercept)     nwbirths.ft
(Intercept)  0.056896258 -0.00151596506
nwbirths.ft -0.001515965  0.00004402731

Correlation of Coefficient Estimates
            (Intercept) nwbirths.ft
(Intercept)   1.0000000  -0.9578252
nwbirths.ft  -0.9578252   1.0000000

rho =  0.6454

Iterations =  9
Gradient norm =  4.431e-7
Log-likelihood =  -200.4
```

```
Convergence:  RELATIVE FUNCTION CONVERGENCE
```

係数の推定値に関する詳細な情報を得るために総称関数 summary を用いた。線形成分の係数の推定値は 1.646 と 0.035 である。各係数の標準誤差とその t 統計値が与えられている。与えられた両側確率点により、どちらの係数も 0 ではないという仮説に対して有意である。残差の分布は、基準化された残差の分布である。これについては 5.3.3 節で述べる。パラメータ ρ の推定値が減少しており、トレンドなしモデルに比べて対数尤度が増加している。

Cressie (1993)による SIDS データの解析では、SIDS 死亡者数の類似をもとに全体を 12 のクラスタに分類している。我々は SIDS 死亡者数の大域的トレンドを、因子 sids$group によるグループごとに 12 の異なる平均を与えてモデリングする：

```
> sids.gpslm <- slm(sid.ft ~ group-1, cov.family=CAR,
+       data = sids, subset= -4, spatial.arglist=
+       list(neighbor=sids.neighbor, region.id = 1:100,
+       weights = 1/sids$births))
> sids.gpslm
Call:
slm(formula = sid.ft ~ group - 1, cov.family = CAR,
data = sids, subset = -4, spatial.arglist =
list(neighbor = sids.neighbor, region.id = 1:100,
         weights = 1/sids$births))

Coefficients:
    group1   group2   group3   group4   group5  group6
 2.054726 2.869104 4.253104 2.476255 2.148905 2.63754
    group7   group8   group9  group10  group11  group12
 3.276925 3.110909 2.678222 2.836351 3.178222 3.685853

Degrees of freedom: 99 total; 86 residual

sigma^2 =  983.9988
rho =  0.7093103

Iterations =  11
```

```
Gradient norm =  2.063091e-006
Log-likelihood =  -185.7641
Convergence:  RELATIVE FUNCTION CONVERGENCE
```

係数の推定値は 12 グループの影響の大きさを表わしている。

5.1.5 節で我々は、近傍行列を 2 つ持つ空間近傍を作成した。互いの距離が 20 マイル以内のものを近傍とする行列（$N_1$）と、20 マイルから 40 マイルのものを近傍とする行列（$N_2$）である。5.2 節で、我々はこの空間近傍の 2 つの Moran 空間的自己相関統計量を求めた共分 ($\hat{r}_1$と$\hat{r}_2$)。散行列を $\rho_1 N_1 + \rho_2 N_2$ とする人種モデルを以下のようにして当てはめる：

```
> sids.race2slm <- slm(sid.ft~nwbirths.ft,
+      cov.family=CAR, data = sids, subset = -4,
+      spatial.arglist = list(neighbor = sids.2nhbr,
+      region.id = 1:100, weights = 1/sids$births))
> summary(sids.race2slm)
Call:
slm(formula = sid.ft ~ nwbirths.ft, cov.family = CAR,
    data = sids, subset = -4, spatial.arglist =
    list(neighbor = sids.2nhbr, region.id = 1:100,
    weights = 1/sids$births))
Residuals:
    Min     1Q Median     3Q    Max
 -105.1 -18.52  4.616 27.18 83.49

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept) 1.7207 0.2718      6.3298   0.0000
nwbirths.ft 0.0324 0.0074      4.3663   0.0000

Residual standard error: 34.162 on 95 degrees of freedom

Variance-Covariance Matrix of Coefficients
             (Intercept)    nwbirths.ft
(Intercept)  0.073896230 -0.00191830774
nwbirths.ft -0.001918308  0.00005491929
```

```
Correlation of Coefficient Estimates
            (Intercept) nwbirths.ft
(Intercept)   1.0000000   -0.9522362
nwbirths.ft  -0.9522362    1.0000000

rho =  0.3445 1.036

Iterations =  6
Gradient norm =  8.274e-7
Log-likelihood =  -199.9

Convergence:  RELATIVE FUNCTION CONVERGENCE
```

線形係数の推定値が変化し、今回はそれぞれの近傍行列に対するρの 2 つの推定値が与えられる。

### 5.3.3 モデル選択

モデリングを行なうほとんどの場合と同様、SIDS データにも複数の可能なモデルが存在する。空間的共分散の成分により、可能なモデルはさらに増加する。説明変数を見落としたり、誤って与えたりするといった通常の問題に加えて、近傍の選択、近傍の重み、共分散モデルなどもモデルに大きく影響を与える。この節では、いくつかのモデルの比較と、S-PLUS と S+SPATIALSTATS で利用可能な残差診断の技法について簡単に述べる。

5.3.2 節で当てはめたモデルの比較を尤度比検定によって行なう。Cressie (1993, p. 562)による検定統計量は：

$$U^2 = 2 * \{(n - p - r)/n\} * (L_p - L_{p+r})$$

ここで $n$ は標本数、$p$ はパラメータが少ない方のモデルのパラメータ数、$r$ は推定パラメータが多い方のモデルに必要な付加パラメータ数[1]、$L_p$ はパラメータの少ないモデルの対数尤度の符号を変えたもの、$L_{p+q}$ はパラメータの多いモデルの対数尤度の符号を変えたものである。この統計量は漸近的に自由度 $r$ の $\chi^2$ 分布に従う。トレンド無しモデル対人種モデルに対し

[1] パラメータが多い方のモデルのパラメータ数は合計 $p+q$。

てこれを計算する：

```
> sids.nullslm$objective
[1] 211.8542
> sids.raceslm$objective
[1] 200.402
> U2 <- ((2 * (99 - 3 - 1))/99) * (211.854 - 200.402)
> 1 - pchisq(U2, df = 1)
[1] 2.757096e-006
```

これより、説明変数に有色人種の出生数を用いた線形モデルは、トレンド無しモデルの有意な改善を行なっている。

モデルの係数に対しては、`slm` の要約の中で与えられる t 統計量による検定が行なえる。あるいは、関数 `lrt` を使って、ある一定値に対する係数やパラメータの仮説検定をどのモデルに対しても行なうことができる。例えば、人種モデルの線形係数が 0 かどうかの仮説検定は：

```
> coef(sids.raceslm)
 (Intercept) nwbirths.ft
    1.645617   0.03454844
> lrt(sids.raceslm, coefficients = c(nwbirths.ft = 0))
Likelihood Ratio Test

Chisquare statistic = 22.90435, df =1,
p.value = 1.702664e-006

parameters:
    param1
 0.8334194

coefficients:
 (Intercept) nwbirths.ft(fixed)
    2.839042                  0
```

関数 `coef` は、与えられたモデルの係数名と係数の推定値を返す。`lrt` 内の引数 `coefficients` は `nwbirths.ft` という変数の係数が 0 であるという帰無仮説に対する検定を行ないたいことを示している。検定結果は有意である。`lrt` の出力中の係数は帰無仮説に対するものである。

同様に、20 マイルから 40 マイル離れたものを近傍とした近傍行列に対する ρ の推定値が 0 かどうかの仮説検定を行なうことができる：

```
> lrt(sids.race2slm, parameters = c(NA,0))
Likelihood Ratio Test

Chisquare statistic = 2.77897, df =1, p.value = 0.0955096

parameters:
     param1 param2(fixed)
 0.4635494              0

coefficients:
 (Intercept) nwbirths.ft
   1.602641   0.03580331
```

推定されたパラメータは 0 でないという検定において（有意水準 0.05 に対して）有意でない。パラメータ数を減らしたモデルのパラメータと係数の推定値が上で出力されている。

### 5.3.4 モデル診断

slm による当てはめからの残差は、共分散モデルによって様々に導かれるが（CAR、SAR、MA のヘルプファイル参照）、どの場合でもおおよそ、分散一定の正規分布に従う独立なものになる。SIDS データの場合、分散を安定化するため対角重み行列を用い、残差をこれらの重みによってスケーリングした。正規性のチェックから診断を開始しよう：

```
> par(mfrow=c(1,2))
> par(pty=”s”)
> hist(sids.raceslm$residuals)
> qqnorm(sids.raceslm$residuals)
```

```
> qqline(sids.raceslm$residuals)
```

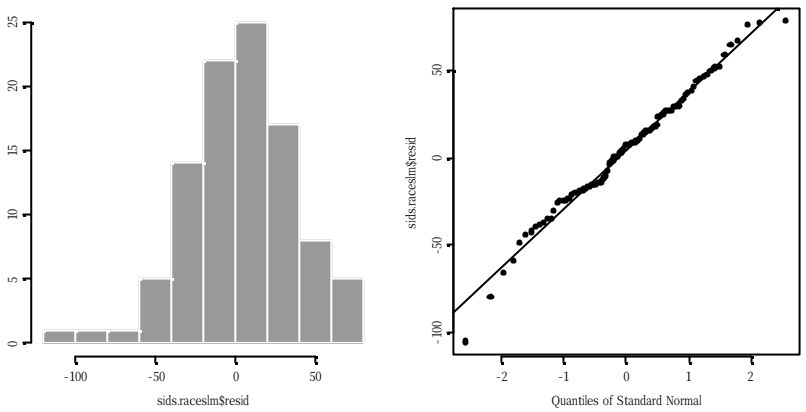


**図 5.2**. 人種モデルからの残差のヒストグラムと QQ プロット。

図 5.2 を見ると、負の方の大きな数個の値により正規性がそこなわれているのがわかる。

残差を当てはめ値に対してプロットし、均質性と外れ値をチェックする：

```
> par(mfrow=c(1,1))
> plot(fitted(sids.raceslm), resid(sids.raceslm))
> abline(h=0)
> identify(fitted(sids.raceslm), resid(sids.raceslm),
+     label=sids$id[-4])
[1] 91 33 40
```

関数 `fitted` が返す値は線形モデルのみを当てはめた場合の結果である。関数 `resid` が返す値は `sids.raceslm$residuals` に保存されている残差である。図 5.3 の中で関数 `identify` を使って、負の方向に大きな残差に領域番号のラベルをつけた。領域 4 がモデルに含まれていないことを思い出すと、残差、当てはめ値ともに 99 個しかないが、`sids` データフレームは 100 のレコードを持つ。これらの領域のデータを見るには以下のようにする：

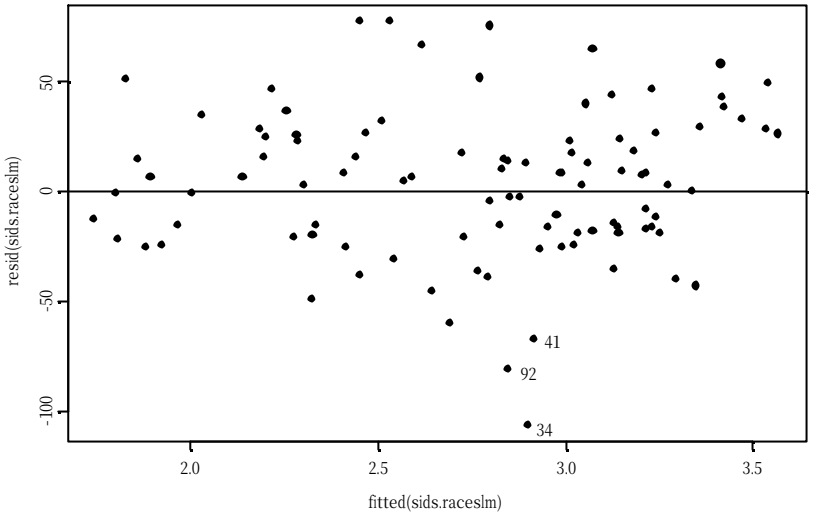


**図 5.3. 人種モデルの残差診断：残差 対 当てはめ値。**

```
> sids[c(34,41,92),]
          id easting northing  sid  births nwbirths
 Forsyth 34     233      153   10   11858     3919
Guilford 41     258      150   23   16184     5483
    Wake 92     319      133   16   14484     4397
          group   sid.ft nwbirths.ft
 Forsyth      1 1.881463    36.36132
Guilford      2 2.409886    36.81425
    Wake      6 2.134410    34.84887
> summary(sids$births)
 Min. 1st Qu. Median Mean 3rd Qu.  Max.
  248    1077     2180 3300    3936 21590
```

これらの領域は共通して出生数が高い。これらの残差が負の方に大きい値になったのはスケーリングによるものかもしれない：残差は `1/sids$births` によって割り算される。しかし、出生数が 10,000 を越す郡が他にも 3 つあるが、これらの残差はそれほど大きくない。

これらの領域の近傍のデータを見てみる：

```
> sids[sids.neighbor$col.id[sids.neighbor$row.id==34],
+      ]
          id  easting northing  sid  births nwbirths
Davidson 29      231      133    8    5509      736
```

```
   Davie 30      213      139    1    1207      148
Guilford 41      258      150   23   16184     5483
  Stokes 85      233      175    1    1612      160
  Yadkin 99      208      156    1    1269       65
         group   sid.ft nwbirths.ft
 Davidson     5 2.483219    23.12491
    Davie     5 2.197465    22.18395
 Guilford     2 2.409886    36.81425
   Stokes     1 1.901486    19.95650
   Yadkin     1 2.143112    14.36867
> sids[sids.neighbor$col.id[sids.neighbor$row.id ==41],
+     ]
           id easting northing  sid  births nwbirths
  Alamance  1     278      151   13    4672     1243
   Forsyth 34     233      153   10   11858     3919
  Randolph 76     256      126    7    4456      384
Rockingham 79     257      173   16    4449     1243
           group   sid.ft nwbirths.ft
  Alamance     2 3.399155    32.62883
   Forsyth     1 1.881463    36.36132
  Randolph     6 2.593262    18.57828
Rockingham     2 3.851154    33.43657
> sids[sids.neighbor$col.id[sids.neighbor$row.id ==92],
+     ]
         id easting northing  sid  births nwbirths
 Chatham 19     291      127    2    1646      591
  Durham 32     306      146   16    7970     3732
Franklin 35     337      155    2    1399      736
 Harnett 43     309      105    6    3776     1051
Johnston 51     335      114    6    3999     1165
         group    sid.ft nwbirths.ft
 Chatham     6  2.452338    37.91337
  Durham     2  2.877352    43.28134
Franklin     2  2.660029    45.88888
 Harnett     6  2.622097    33.37480
Johnston     6  2.547939    34.14369
```

これらの残差に対しては、さらに調査する必要がある。

次に、残差を sids$group に対してプロットし、グループ毎に何か相関がないか見てみる：

```
> plot(sisd$group[-4],resid(sids.raceslm))
> abline(h=0)
```

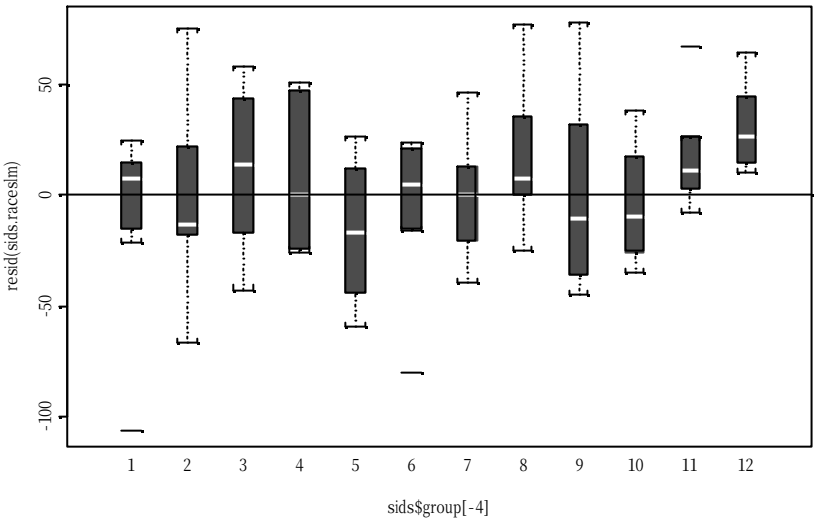


**図 5.4**. 人種モデルの残差診断：残差 対 グループ。

図 5.4 から、関数の形で書けるような関係は発見できない。しかし、グループ 11 と 12（おそらく 8 も）は期待される残差より大きいように思われる。要因の有無を確かめるために各グループの位置をプロットする：

```
> plot(sids$easting,sids$northing,type="n")
> text(sids$easting,sids$northing,sids$group)
```

図 5.5 を見ると、グループ 3 は州の境界周辺に位置している―人為的な境界である。おそらく、これはモデルの残差に対する境界効果によるものと思われる。

人種モデルでは SIDS データの変動に対して、3 つの成分でモデリングを行った：線形成分（fitted(sids.raceslm)）、近傍共分散成分（シグナルと呼ばれる（Haining, 1990, p. 258））、残差変動もしくはノイズである。人種モデルのシグナル成分を求める最も容易な方法は、元データから当てはめ値とノイズを引き去る方法である：

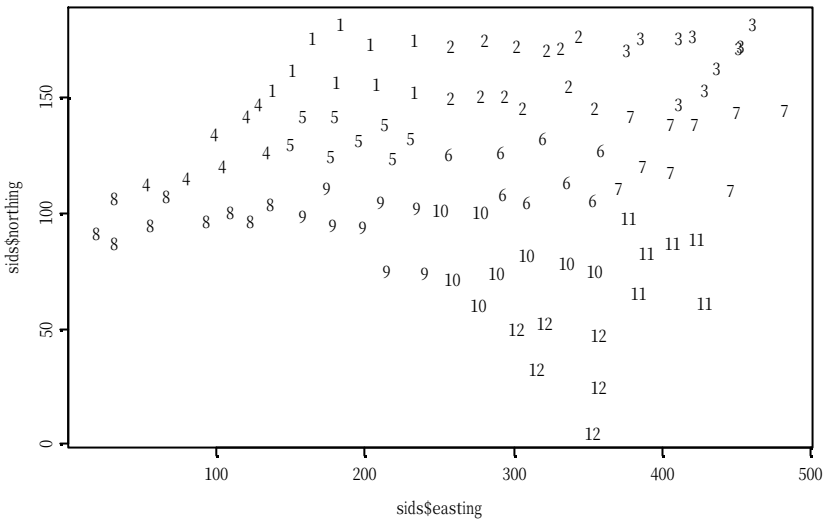


**図 5.5.** group の位置。

```
> noise <-(1/sqrt(sids$births[-4]))*resid(sids.raceslm)
> signal <- sids$sid.ft[-4]-fitted(sids.raceslm)-noise
```

データに加える成分とするためには、ノイズはスケーリングされていないノイズでなければならない。シグナルは直接計算することもできる。CAR モデルでは、シグナルは $\rho N(Y - X\beta)$ である：

```
> nhbr2 <- spatial.subset(sids.neighbor,
+     region.id=c(1:3,5:100), subset=c(1:3,5:100))
> rhoN <- spatial.weights(nhbr2$neighbor,
+     parameter=0.6454, region.id=c(1:3,5:100))
> signal2 <- rhoN %*%
+     (sids$sids.ft[-4]-fitted(sids.raceslm))
> summary(signal-signal2)
        Min.      1st Qu.       Median       Mean
-0.00001617  -2.968e-006  -1.092e-007  0.001053
    3rd Qu.      Max.
 2.083e-006  0.04893
```

領域 4 を除くために関数 spatial.subset を使用し、近傍の相関の行列を求めるのに関数 spatial.weights を使用した。シグナルの 2 つの計算結果はよく一致している。

モデルによる予測値は当てはめ値＋シグナルによって計算することがで

きる。人種モデルの当てはまり具合を評価するために、予測値を実際の SIDS 死亡者数に対してプロットしてみる：

```
> plot(sids$sid.ft[-4], fitted(sids.raceslm) + signal,
+      ylim=c(1,5.2), xilm=c(1,5.2))
> abline(0,1)
```

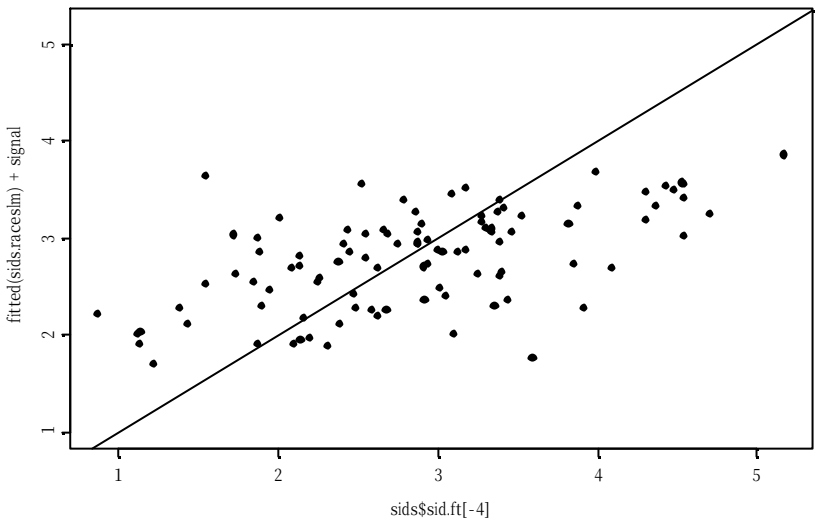


**図 5.6.** Freeman-Tukey 変換後の SIDS データに対する人種モデルによる予測値の散布図。

人種モデルでは SIDS 死亡者数の変動がつかみ切れていないことを図 5.6 は示している。

データ対予測値を地図の塗り分けで表現することでモデルの視覚的解釈が可能になる。これは以下の手順で行う：

1. 塗り分けのために、SIDS データ、当てはめ値、モデルによる予測値を 4 グループに分割する：

```
> breaks.sids <- c(-.001, 2.2, 3.0, 3.5, 7)
> sids.y <- c(sids$sid.ft[1:26]
+     rep(sids$sid.ft[27], 3), sids$sid.ft[28:100])
> sids.ygrp <- cut(sids.y, breaks.sids)
> fit2 <- fitted(sids.raceslm)
> fit2 <- c(fit2[1:3], NA, fit2[4:25],
+       rep(fit2[26], 3), fit2[27:99])
```

```
> sids.fgrp <- cut(fit2, breaks.sids)
> pred2 <- signal + fitted(sids.raceslm)
> pred2 <- c(pred2[1:3], NA, pred2[4:25],
+      rep(pred2[26], 3), pred2[27:99])
> sids.pgrp <- cut(pred2, breaks.sids)
```

州内地図の多角形の数と予測値の数を同じにする必要がある（27 番目の郡 Currituck は隣接する 3 つの多角形を持つ）。

2. 州の塗り分け地図を作成する：

```
> library(maps)
Warning messages:
  The functions and datasets in library section maps
   are not supported by MathSoft. in: library(maps)
> par(mfrow=c(3,1))
> map("county", "north carolina", fill=T,
+     color=sids.ygrp)
> map("county", "north carolina", add=T)
> title(main="Acutual Transformed SIDS Rates")
> legend(locator(1), legend=c("< 2.2", "2.2-3.0",
+     "3.0-3.5", "> 3.5"), fill=c(1:4))
> map("county", "north carolina", fill=T,
+     color=sids.fgrp)
> map("county", "north carolina", add=T)
> title(main="Fitted Values")
> legend(locator(1), legend=c("< 2.2", "2.2-3.0",
+     "3.0-3.5", "> 3.5"), fill=c(1:4))
> map("county", "north carolina", fill=T,
+     color=sids.pgrp)
> map("county", "north carolina", add=T)
> title(main="Predictions")
> legend(locator(1), legend=c("< 2.2", "2.2-3.0",
+     "3.0-3.5", "> 3.5"), fill=c(1:4))
```

図 5.7 の上段は Freeman-Tukey 変換を行った SIDS 死亡者数の塗り分け地図である。これは中段と下段の予測値の塗り分け図と視覚的な比較ができるようになっている。下段は SIDS 死亡者数を空間近傍の効果を含めて予測したものである。人種モデルはある程度の当てはまりを見せているが、

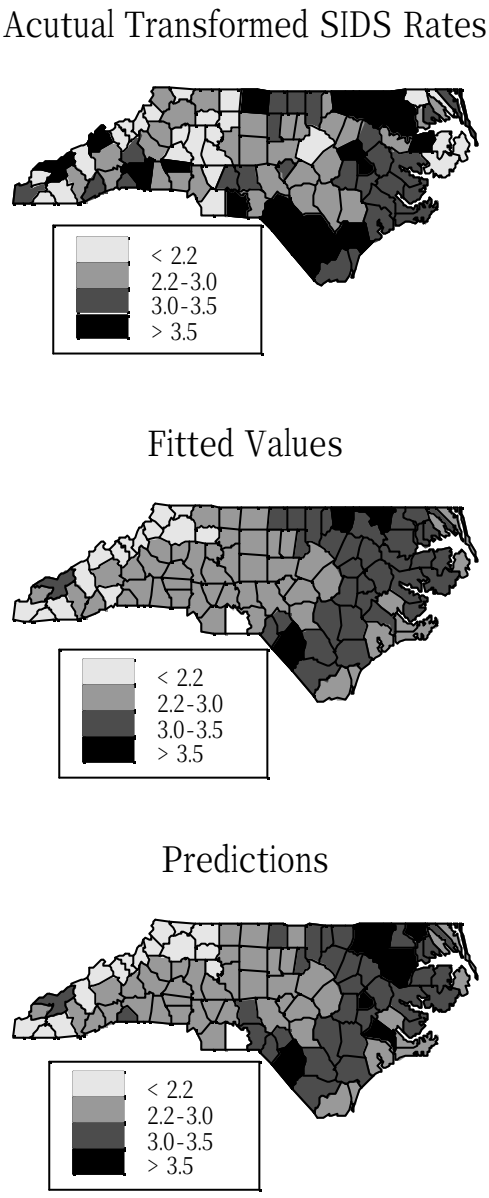


**図 5.8.** モデルによる SIDS 死亡者数の予測値の塗りわけ地図。中段は線形人種モデルのみ。下段は空間成分を含む。

完全ではないことがここでも明らかである。よって、他の線形モデルを考案したり、他の近傍や近傍の重みを定義することが必要である。

## 5.4 格子データのシミュレーション

シミュレーションデータは、この章で構築した回帰モデルの特性を確認・評価するのに役立つ。この方法は、データに当てはまるモデルが複数あることの多い空間データのモデリングにとって特に重要である。

最初に、平均 0、共分散 Σ となるデータベクトルが必要となる。このベクトルは $y = Le$、ここで $LL^T = \Sigma$、$e \sim N(0,1)$である。L は Σ をコレスキー分解して得られる下三角行列である（Haining, 1990, p. 116）。CAR 共分散行列を持つSIDS 人種モデルの 1 つの実現値を以下のようにして生成する：

1. 対角重み行列、単位行列、$\rho N$ を生成する：

```
> sids.diag <- diag(1/sids$births)
> Id <- diag(rep(1,100))
> rhoN <- spatial.weights(sids.neighbor,
+      parameters=c(0.6453))
```

2. CAR モデルの共分散行列 Σ を計算する：

```
>            sids.carcov                          <-
solve(Id-rhoN)%*%sids.diag*1167.897
```

3. Σ をコレスキー分解する：

```
> sids.carcovL  <-
+      chol((sids.carcov+t(sids.carcov))/2)
```

4. 空間的相関を持つランダムベクトルを生成する：

```
> e.norm <- rnorm(100)
> sids.carsim <- t(sids.carcovL) %*% e.norm
```

関数 `chol` は上三角コレスキー行列を返す。下三角行列を得るために転置を行った。

共分散構造に SAR モデルを入れた SIDS 人種モデル（例は示していない）のパラメータを使った SAR モデルのシミュレーションベクトルを以下のようにして生成する：

```
> sids.wN2 <- spatial.weights(sids.neighbor,
```

```
+       parameters=c(0.4738))
> sids.sarcov <- solve(t(Id-sids.wN2)
+        %*%solve(sids.diag)%*% (Id-sids.wN2))*1159.813
> sids.sarcovL <- chol((sids.sarcov + t(sids.sarcov))/2)
> sids.sarsim <- t(sids.sarcovL)%*%e.norm
```

共分散構造に MA モデルを入れた SIDS 人種モデル（例は示していない）のパラメータを使った MA モデルのシミュレーションベクトルを以下のようにして生成する：

```
> sids.wN3 <- spatial.weights(sids.neighbor,
+       parameters=c(0.6337))
> sids.macov <- (Id+sids.wN3)
+        %*% sids.diag %*% t(Id+sids.wN3)*1198.683
> sids.macovL <- chol((sids.macov + t(sids.macov))/2)
> sids.masim <- sids.macovL%*%e.norm
```

3 種類の共分散モデルに対して、シミュレーションを行って得られたこれらのベクトルを適当な平均ベクトルに加えることで、各モデルでの予測を行うことができる。

# 6 空間点パターンの解析

この章では、平面上の空間点パターンの解析とモデリングに利用可能な S+SPATIALSTATS の手続きを紹介する。空間点パターンは空間の有界領域内に不規則に位置する点の集合である。点は、地震や植物の発芽などの自然現象が発生する位置を表わしたり、小都市やある疾病の発生位置といった社会的な現象を表わしたりする。データセットは、位置だけから成る場合と、位置に付随したデータ値（マーク）を持つマーク（$mark$）付き点過程の場合がある。マーク付き点過程の例としては、森林中における樹木の位置の集合に、胸の高さでの幹の直径を付置したものが考えられる。

3.4 節では、キイチゴの茎のデータに対して探索的データ解析を行なった。この章では 2 番目のデータセットとして、ミシガン州クリントン郡のランシングウッズにおける 19.6 エーカーの土地の、カエデとヒッコリーの木の位置に対する解析を主に行なう [(Diggle, 1983, p.27), (Gerrard, 1969)]。データは単位正方形内に収まるようスケーリングが施されているが、これは S+SPATIALSTATS の解析で必ずしも必要ではない。

本章では S+SPATIALSTATS による以下の作業について学習する：

- 点パターンデータの完全ランダム性を調べる（6.2 節）。
- 空間点パターンの強度を推定する（6.3.1 節）。
- Ripley の K 関数の計算（6.3.2 節）。
- 空間点パターンのシミュレーション（6.4 節）。

## 6.1 クラス"spp"のオブジェクト

S+SPATIALSTATS のこのリリースにおける空間点パターンとは、平面上のパターンである。この平面上のパターンを表現する座標の集合はクラ

ス"spp"のオブジェクトとして保存される。点パターンオブジェクトは、2 つの列（典型的には最初の 2 列）に点の位置が含まれているデータフレームである。それ以上の列はあってもなくてもよい。これらの座標軸は緯度・経度、東西距・南北距、ユーザが定義した座標軸などいろいろなものが考えられる。データフレーム lansing から点パターンオブジェクトを生成するには以下のようにする：

```
> summary(lansing)
       x                y            species
 Min.   :0.0010   Min.   :0.0000   hickory:703
 1st Qu.:0.2480   1st Qu.:0.2410   maple:514
 Median :0.5130   Median :0.5220
 Mean   :0.5094   Mean   :0.5027
 3rd Qu.:0.7720   3rd Qu.:0.7370
 Max.   :1.0000   Max.   :0.9910

> lansing.spp <- spp(lansing)
```

関数 spp は位置座標 $x$ と $y$ を含む行列またはデータフレームからクラス"spp"のオブジェクトを生成する。点パターンオブジェクトに要約メソッドを適用すると、点の総数と各軸の範囲が表示される：

```
> summary(lansing.spp)
Total number of points: 1217
Coordinate extents :
 x :  0.001 , 1.000
 y :  0.000 , 0.991
Number of boundary vertices : 4
Other covariates :
    species
 hickory:703
 maple  :514
```

点パターンオブジェクトに描画メソッドを適用すると、縦横の縮尺が幾何的に正確なプロットが行われる。ランシングデータのプロットを以下のように行う：

```
> par(pty="s")
> plot(lansing.spp, boundary=T)
```

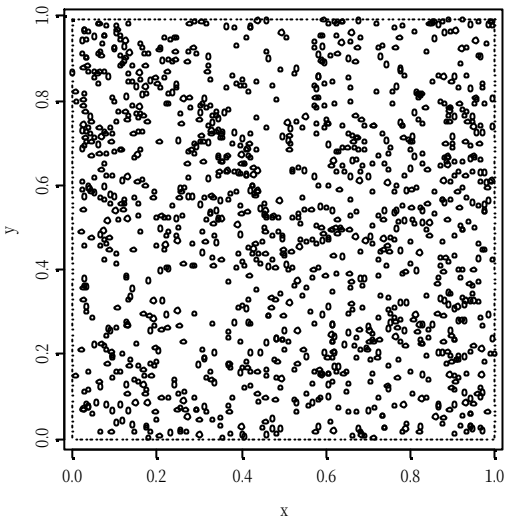


**図 6.1. ランシングウッズの樹木の位置の散布図。**

図 6.1 では省略可能な引数 boundary=T を指定して、ランシングデータの属性 boundary で与えられる矩形をプロットした。境界は spp の呼び出し内で特定していなかったため、デフォルトである 2 つの軸の範囲が与えられた。これは S-PLUS 関数 bbox を用いて求められる：

```
> bbox(lansing)
$x:
[1] 0.001 0.001 1.000 1.000

$y:
[1] 0.991 0.000 0.000 0.991

> attributes(lansing.spp)$boundary
$x:
[1] 0.001 0.001 1.000 1.000

$y:
[1] 0.991 0.000 0.000 0.991
```

調査区域が単位正方形であるため、spp の呼び出しの中で正確な境界を与えたいこともある：

```
> lansing.spp <- spp(lansing, boundary=bbox(x=c(0,1),
```

```
+      y=c(0,1)))
```

ここで bbox は $x$ と $y$ の範囲で与えられる境界枠を計算している。

一般に、点パターンオブジェクトの境界には任意の凸多角形が可能である。3.4 節では、キイチゴ点パターンを含む最小の凸多角形を生成するために chull を用いた例を紹介した。そのほか、関数 locator を用いて多角形を対話的に定義する方法がある。ランシングウッズのカエデの木に対して多角形の境界を生成するには以下のようにする：

1. 余白を多く取った平面にカエデの木の位置をプロットする：

```
> attach(lansing)
> scaled.plot(x[species=="maple"],
+      y[species=="maple"], xlim=c(-.2,1.2),
+      ylim=c(-.2,1.2))
```

2. locator を使って境界を選択する。グラフィックウィンドウ上でマウスの左ボタンをクリックして位置を指定する。指定が終わったらグラフィックウィンドウ上で中央ボタンまたは右ボタンを押して終了する：

```
> maple.poly <- locator(type="line")
```

多角形が閉じている必要はないことに注意。

3. 多角形が凸であるかどうかをチェックする：

```
> is.convex.poly(maple.poly)
[1] T
```

関数 locator はグラフィックウィンドウ上で指定した点の座標を返す。境界が凸であるかどうかを確かめるには関数 is.convex.poly を使う。多角形 maple.poly は spp の呼び出しの中で boundary として与えることができる。

## 6.2 空間的ランダムネスの尺度

この節では空間点パターンの解析の手始めに、完全ランダムネス（CSR, complete spatial randomness）に対する検定を行う。我々は CSR を以

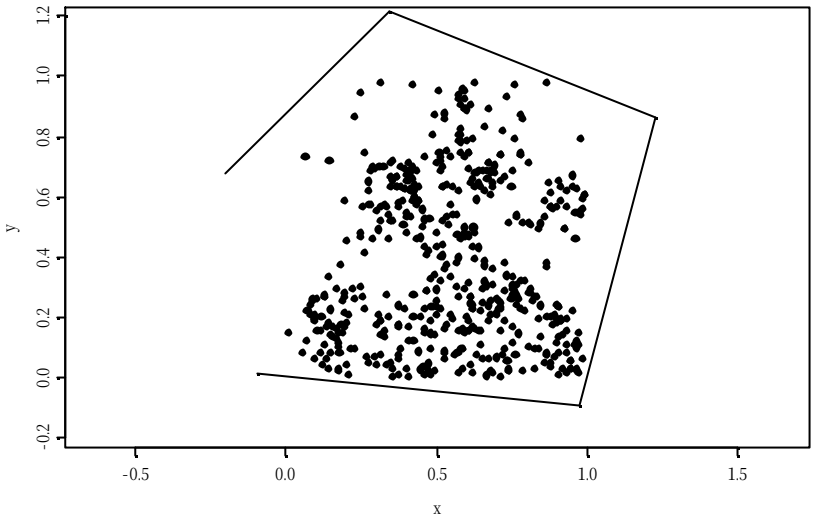


**図 6.2.** ランシングウッズのカエデの散布図。対話的に描いた境界もプロットした。

下の判断基準により定義する（Diggle, 1983, p. 4）：

1. 点パターンの**強度**（*intensity*）（単位面積辺りの点の数）が有界領域 $A$ 内で変化しない。数学的に表現すると、面積 $|A|$ の平面領域 $A$ 内の点の数が均質的に平均 $\lambda|A|$ のポアソン分布に従う。ここで $\lambda$ は一定の強度である。

2. 点どうしの相互作用がない―点は互いに引き付けあうことも退けあうこともしない。数学的には、領域 $A$ 内の $n$ 個の点の位置をベクトル $\boldsymbol{x}$ で表わすとすると、$\boldsymbol{x}$ は $A$ 内の一様分布からの独立なランダム標本である。

以下の節では、S+SPATIALSTATS を用いてこの仮説の有効性を調べる技法について見てゆく。

### 6.2.1 視覚的手法

図 6.3 はランダムでない点パターンの 2 つの例を示している。両者は 6.4 節で述べるシミュレーション法によって発生させたものである。左側の散布図は明らかにクラスタ化しており、右側の散布図には規則性が見られる。

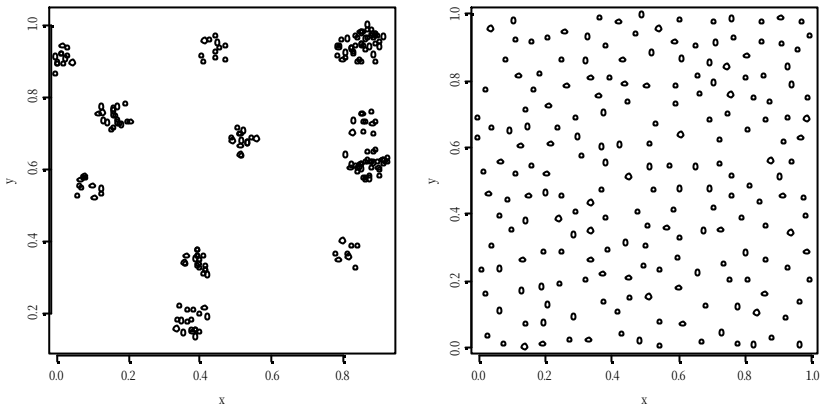


**図 6.3.** 明らかにランダムでない点パターンの 2 つの例。

図 6.1 のランシングウッドのデータはこれらの散布図とは異なり、非常に密集しており、パターンがないことは一目見て明らかである。ランシングウッドのデータは、ヒッコリーとカエデの 2 種類の点が混在する 2 変量点パターンの例である。これらを種ごとにプロットするには以下のようにする：

```
> hick.spp < - spp(lansing[lansing$species=="hickory",],
+      boundary=bbox(x=c(0,1), y=c(0,1)))
> maple.spp <- spp(lansing[lansing$species=="maple",],
+      boundary=bbox(x=c(0,1), y=c(0,1)))
> par(mfrow=c(1,2))
> par(pty="s")
> plot(hick.spp,main="Hickories")
> plot(maple.spp,main="Maples")
```

ヒッコリーとカエデの点パターンオブジェクトを、全データセットの時と同じ単位正方形の境界を用いて生成した。図 6.4 の散布図は 2 種の間に相互作用があることを示している。片方の種の存在が他方の種の存在を妨げているようである。カラーモニタを使用できる場合は、関数 `points` を使って以下のように片方の種の上に他方の種を重ね描きしてみよ（散布図は掲載しない）：

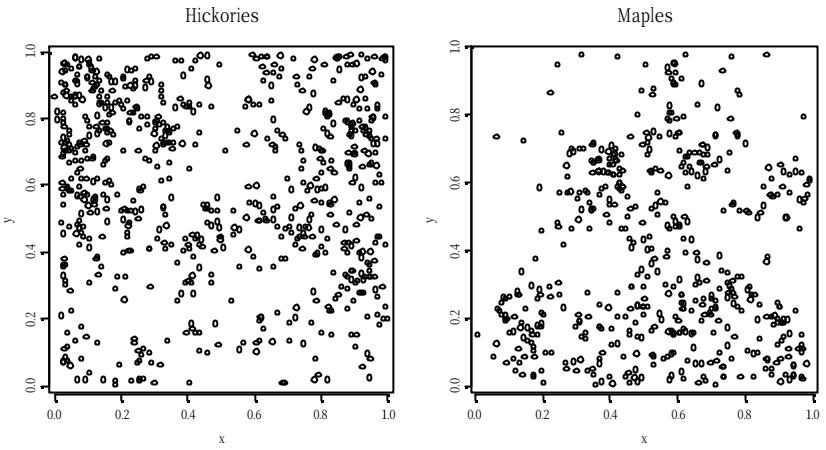


**図 6.4.** ランシングウッズの木の位置を 2 種に分けてプロットしたもの。

```
> plot(hick.spp)
> points(maple.spp, col=3)
```

カラーモニタを使用できない場合は以下のようにするとよい（散布図は掲載しない）：

```
> points(maple.spp, pch="x")
```

全データで考える限りは空間的にランダムに見えても、種に関する知識によってこのデータには興味ある空間的成分があることが明らかになった。知っておく必要のあるすべて（図 6.3 のようなケース）は視覚的技法でも十分伝わるが、CSR のより数学的な検定が求められる。

### 6.2.2 最近接法

最近接距離は、点どうしの局所的な相互作用を見るための客観的な手法である。これらの距離は S+SPATIALSTATS の関数 `find.neighbor` を用いて容易に計算することができる。`find.neighbor` の使い方に関する詳細は 5.1.3 節を参照。以下の節で議論される最近接統計量は、手持ちのデータと、ランダムなパターンとの性質の違いを比較するために用いることができる。

### point-to-point 最近接統計量

最近接距離 $d_i$ を、$i$ 番目の点から有界領域 $A$ の中で最も近い点までの距離とする。これらの **point-to-point 最近接距離**（*point-to-point nearest neighbor distances*）の経験分布関数（EDF）は CSR プロセスとの比較に用いることができる。EDF は：

$$\hat{G}(y) = n^{-1} \sum_{d_i \le y} 1$$

ここで $n$ は $A$ 内の点の数である。

ランシングウッズのデータに対して関数 `Ghat` を使って $\hat{G}$ を求め、プロットするには以下のようにする：

```
> par(mfrow=c(1,2))
> hick.ghat <- Ghat(hick.spp)
> title(main="Hickories")
> maple.ghat <- Ghat(maple.spp)
> title(main="Maples")
```

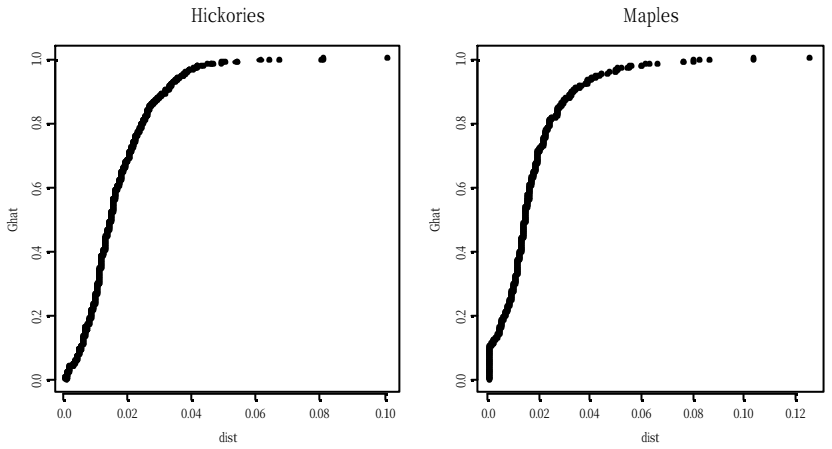


**図 6.5.** ランシングウッドの 2 種に対する point-to-point 最近接距離の EDF をプロットしたもの：ヒッコリー（左）、カエデ（右）。

各 EDF は引数 `plot=F` を指定しない限り `Ghat` により自動的にプロットされる。距離 $y$ は引数 `dist.ghat` によって与えることができる。与えない場合、すべての最近接距離が使われる。図 6.5 は 2 種の $\hat{G}(y)$ をプロッ

トしたものである。データがクラスタ化する場合、距離が小さいところで値が大きくなることが予想される。データが規則的な場合、距離が大きいところで値が大きくなることが予想される。

⟹ **ヒント：**デフォルトの距離を用いるほうが `Ghat` の実行速度は速い。

**注意：**Diggle [1983]にあるように、データフレーム `lansing` にはヒッコリーの 703 本の位置が保存されている。上で生成されたオブジェクト `hick.ghat` には 701 の最近接距離しが存在しないことから、これらの位置の中には距離が等しくなる点が 2 つあることが分かる。

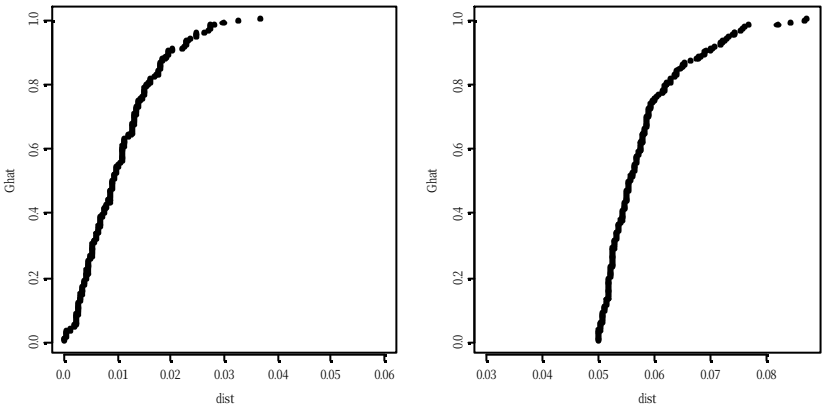


**図 6.6.** 2 つのランダムでない点パターンに対する point-to-point 最近接距離の EDF をプロットしたもの。

ランシングウッズのデータとの比較のために、図 6.3 の 2 つの点パターンに対して$\hat{G}$を求めてプロットしたものが図 6.6 である。左側がクラスタ化している点パターン例の$\hat{G}(y)$であり、右側が規則性を持つ点パターン例の$\hat{G}(y)$である。

$\hat{G}$プロットにより視覚的に判断するのに変わる方法には、理論分布の検定やシミュレーション法などがある。CSR に対する $G(y)$の理論分布は境界効果が無視できる場合にのみ閉じた形で利用できる。この場合については後で議論する。

境界効果を考慮する必要がある場合、CSR という仮説に対してはモンテ

カルロ法を用いて評価することができる。例えば、我々は元の点パターンを含む領域 $A$ 内の CSR プロセスによる $s$ 個の実現値から最近接距離の EDF をシミュレーションによって求めることができる。$s$ 回のシミュレーションによる EDF の平均は参照線としてグラフ上に引かれ、最大値と最小値はシミュレーション・エンベロープ（simulation envelope）となる。もし、データから求められた $\hat{G}$ が、$y$ の小さい範囲または大きい範囲でこのエンベロープ（envelope）からはみ出せば、CSR という仮説に反する証拠になる。

ランシングウッズのデータでは、$A$ はスケーリングが施された単位正方形である。よって、強度の推定値（CSR では一定と仮定）は単に点の総数に等しい。我々は S+SPATIALSTATS の関数 `make.pattern` を用いて、比較対象となるポアソン（CSR）過程をシミュレーションさせることができる。ある矩形内で点の総数が $n$ になる CSR パターンを $s$ 回発生させ、$\hat{G}$ を計算する関数を以下のように作成する：

```
> ghat.env <- function(n, s, dist,
+      boundary=bbox(x=c(0,1),y=c(0,1))){
+ # n は発生させる点の個数
+ # s はシミュレーションを行う回数
+ # dist は Ghat を計算する距離
+     hold <- matrix(0, s, length(dist))
+     for(i in 1:s) {
+          hold[i,  ] <- Ghat(make.pattern(n,
+               boundary=boundary), dist.ghat=dist,
+               plot=F)[,2]
+     }
+ # hold の各列に単位正方形上の CSR のシミュレーションの
+ # 各回で求められた EDF が保存される
+     mn <- apply(hold,2,mean)
+     Up <- apply(hold,2,max)
+     Down <- apply(hold,2,min)
+     return(data.frame(mn, Up, Down))
}
```

関数 `make.pattern` については 6.4 節で詳しく議論する。

ランシングウッズのカエデに対する $\hat{G}$ 統計量のシミュレーション・エン

ベロープを生成し、プロットする：

```
> maple.env <- ghat.env(n=514, s=20,
+       dist=unique(maple.ghat[,1]))
```

ここでは 20 の CSR 点パターンをシミュレーションによって発生させ、ランシングウッズのカエデの $\hat{G}$ と同じ $y$ における EDF を計算した。引数 boundary を指定していないが、これはデフォルトが単位正方形になっているからである。

これら 20 のシミュレーションの平均を母集団値の推定値として用いるならば、これをデータの EDF に対してプロットすると、CSR の仮定のもとではほぼ直線になるはずである。最小値と最大値のシミュレーション・エンベロープとともにこれをプロットするには以下のようにする：

1. maple.env の算出に用いた距離 $y$ は maple.ghat の中で何度も登場する。S-PLUS の関数 tapply を用いて、maple.env をプロットするためのインデックス用ベクトルを作成する。

   ```
   > ind <- tapply(maple.ghat[,1], maple.ghat[,1])
   ```

2. CSR の $G(y)$ の推定値に対して、カエデデータの $\hat{G}(y)$ とエンベロープをプロットする。

   ```
   > par(mfrow=c(1,1))
   > plot(maple.env$mn[ind],maple.ghat[,2],
   +     type="line")
   > lines(maple.env$mn[ind], maple.env$Up[ind],
   +     lty=2)
   > lines(maple.env$mn[ind], maple.env$Down[ind],
   +     lty=2)
   ```

図 6.7 には、カエデの木のクラスタ化に対する強い証拠が示されている。この証拠にはヒッコリーの木からの作用効果が含まれていることを特に注意すべきである。もしこのシミュレーションを、カエデの木だけを含む多角形領域に限って行ったならば、この証拠はおそらく弱まるだろう。

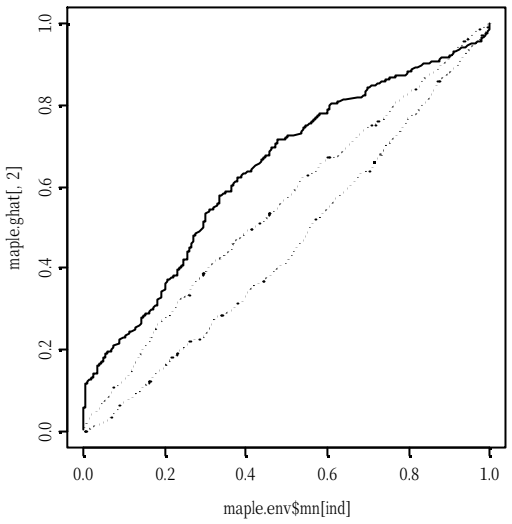


**図 6.7.** ランシングウッズのカエデデータに対する point-point 最近接距離の EDF を、CSR 過程を 20 回シミュレーションして得た上限と下限のシミュレーション・エンベロープとともにプロットしたもの。

**origin-to-point 最近接距離**

$\hat{G}$ に関連した検定統計量を、領域 $A$ 上に $k \times k$ グリッドを重ねてできる $m$ 個の原点とそれから一番近い点との距離を比較することで定義する。$e_i$ を $i$ 番目の原点からデータの $n$ 個の点の中で一番近い点までの距離とする。これらの **origin-to-point 最近接距離**[1]（*origin-to-point nearest neighbor distance*）の EDF は：

$$\hat{F}(x) = m^{-1} \sum_{e_i < x} 1$$

である。関数 `Fhat` を以下のように用いて、ランシングウッズのデータの 2 種に対する $\hat{F}(x)$ を求める：

```
> par(mfrow=c(1,2))
> hick.fhat <- Fhat(hick.spp)
> title(main="Hichories")
> maple.fhat <- Fhat(maple.spp)
> title(main="Maples")
```

[1] **訳注：** Stoyan (1995?) は、球状接触分布（*spherical contact distribution*）という名前で定義した。

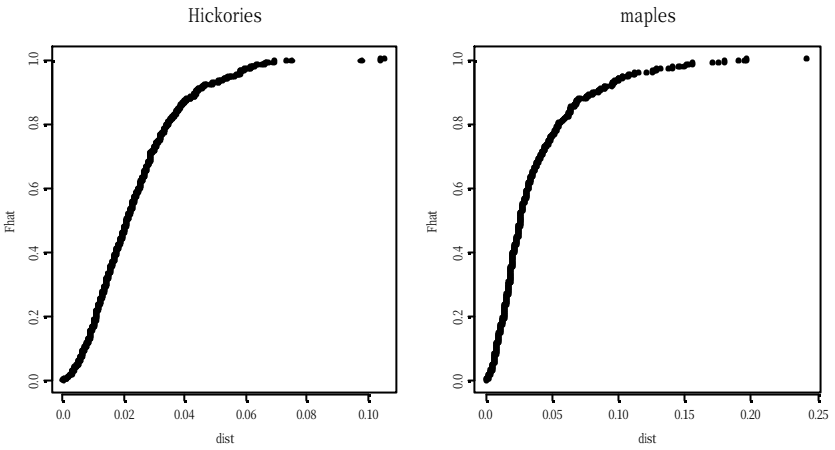


**図 6.8.** ランシングウッズのデータの 2 種に対して origin-to-point 最近接距離の EDF をプロットしたもの：ヒッコリー（左）、カエデ（右）。

ヒッコリーとカエデの $\hat{F}$ 統計量を図6.8にプロットした。$\hat{F}$ のプロットの解釈は $\hat{G}$ のプロットと反対になる。距離の大きいところで値が大きくなるものはクラスタ化していると解釈される。前の場合と同様、この統計量を CSR プロセスのシミュレーション結果と比較して視覚的な解釈を得ることが可能である。この場合も十分な境界効果がないことを仮定し、カエデの $\hat{F}$ を CSR の $F(x)$と比較してみよう。

以前議論したように、境界効果が無視できて、かつ点の数が比較的大きい場合は、理論分布関数 $G(y)$と $F(y)$は閉じた式として表わされる。境界効果は $n$ が小さいとき、または領域が細長い多角形の場合に問題になりがちである。理論分布関数 $G(y)$と $F(y)$は CSR の時、等しくなる：

$$G(y) = F(y) = 1 - \exp\left(-p\, l y^2\right)$$

ここでλは一定の強度である。EDF との比較のため、単位面積当たりの点 の $\hat{l}$ 数で置き換える。次にどちらかの最近接距離を理論分布関数に対して、あるいは他方の距離に対してプロットし、CSR に対する視覚的検定を行う。カエデデータに対して：

```
> maple.edf <- 1 - exp(-514*pi*(maple.fhat[,1]**2))
> plot(maple.edf, maple.fhat[,2])
> abline(0,1)
```

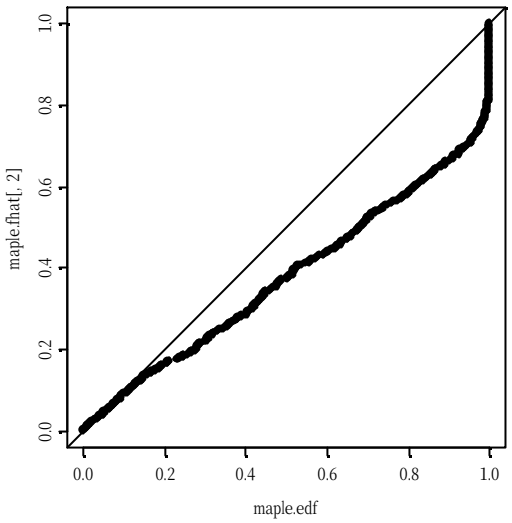


**図 6.9.** CSR の origin-to-point 最近接距離の EDF に対して、ランシングウッズのカエデに対する同様の分布関数をプロットしたもの。

図 6.9 により、ランシングウッズのカエデがクラスタ化していることが再び示された。図 6.10 では比較のため、図 6.3 のクラスタ化と規則的な例に対して同様のプロットを行った。

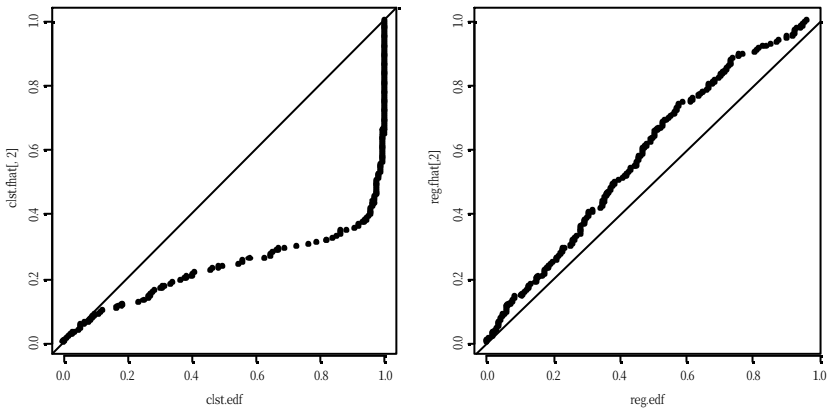


**図 6.10.** CSR の origin-to-point 最近接距離の EDF に対して、クラスタプロセスの同様の EDF（左）と規則的なプロセスの同様の EDF（右）をプロットしたもの。

## 6.3 1次・2次特性値の調査

この節では初めに、空間点パターンを表現するモデルの構築について見てゆくことにする。まず、背景にあるプロセスが**定常**（*stationary*）であるかどうかと、**等方的**（*isotropic*）であるかどうかを判定しなければならない。これらの単語はすでに以前の章で定義済みである。空間点過程においては、プロセスは：

- 強度が一定で、2次の強度が点のペアの方向と距離だけに依存する（絶対位置に依存しない）時、**定常**（*stationary*）であるという。
- 定常であり、かつ2次の強度が点のペアの距離だけに依存し、方向に依存しない時、**等方的**（*isotropic*）であるという。

空間点過程の**2次の強度**（*second-order intensity*）は2点の空間的関係を測る尺度である（Gatrell et al., 1995）。以下の節では、これらの特性量をS+SPATIALSTATSによって求める方法について説明する。

### 6.3.1 強度

空間点過程の1次の特性量は、単位面積当たりの平均的な点の数（強度）が空間内でどう変化するかを記述するものである。定常過程では、強度λは有界領域$A$内で一定であると仮定する。

空間点パターンの強度をS+SPATIALSTATSで推定するには、関数`intensity`を使用する。メソッドは4つある：**ベーシック**（*basic*）、**バイニング**（*binning*）、**核型**（*kernel*）、**2次元正規**（*gauss2d*）。引数`method="basic"`を選ぶと強度の点推定を行なう。すなわち、点の数を有界領域の面積で割るのである。他の3つのメソッドは、領域内で局所的に変化する強度を推定し、平滑化した強度の推定値を含むリストを返す。

これらの変化を視覚化し、点パターンが定常であるという仮説を評価するために関数`image`を使用する。

バイニング（*binning*）メソッドは四角なbinを形成する際、2次元のヒストグラムを用いる。これらのbinに入る点の数は局所平滑法を用いて平滑化される。以下のようにしてカエデデータにバイニングメソッドを使用する：

```
> maple.bing <- intensity(maple.spp, method="binning",
+      nx=30, ny=30, span=.1)
```

引数 nx と ny はマスの数を制御するために用いる。特に指定しなければパターン内の点の総数の平方根がデフォルトになる。引数 span は平滑関数 loess が使用する（span の取り得る値、推奨される値については loess のヘルプファイルを参照）。

関数 intensity は S-PLUS 関数 image を使ってプロットを行なえる $x$、$y$、$z$ を含むリストを返す：

```
> image(maple.bing)
> image(intensity(maple.spp, method="binning", nx=30,
+       ny=30, span=.2))
```

両者の濃淡プロットを図 6.11 に示す。右が大きいスパン 0.2 による結果である。よりスムーズな画像になっている（nx=50、ny=50 も試みよ）。

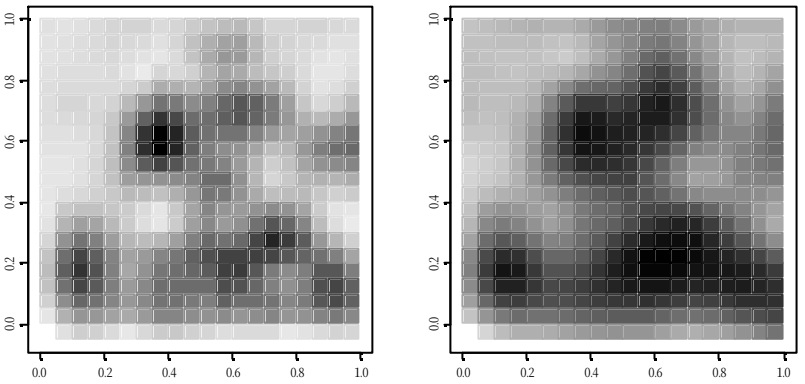


**図 6.11.** ランシングウッズのカエデの強度を濃淡画像にしたもの。span=.1（左）と span=.2(右)の binning メソッドを用いた。

method="kernel"の場合、intensity はパターンの各点における強度を各点の影響範囲（*region of influence*）にある点の重み付き関数を用いて推定する。核型（*kernel*）関数法は空間点パターンの強度推定に、引数

bwで与えられる帯域（影響範囲の半径）を用いた4次核型推定量[2]を用いる。カエデデータに関する4次核型推定量を求める：

```
> maple.kern <- intensity(maple.spp,
+          method="kernel", bw=1)[3]
> image(maple.kern)
```

引数bwは$x$方向と$y$方向で異なってもよい。つまり、長さ2のベクトルで与えることもできる。核型と2次元正規の場合、この引数は必ず与えなければならない。結果の濃淡プロットは図6.12の左である。

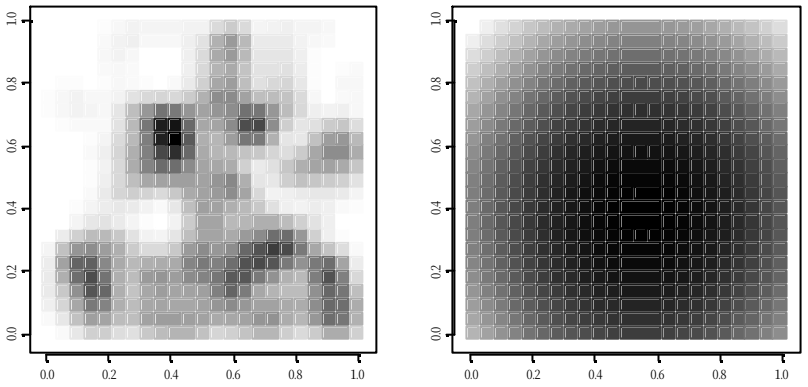


**図 6.12.** ランシングウッズのカエデの強度を濃淡画像にしたもの。推定には4次核型関数（左）、正規核型関数（右）を用いた。

2次元正規(*gauss2d*)メソッドは強度の推定に正規核型推定を使用する。このメソッドを、上と同じ帯域でカエデデータに適用してみる：

```
> maple.gaus <- intensity(maple.spp,
+          method="gauss2d",bw=1)
> image(maple.gaus)
```

---

[2] **訳注：**「$K(x)=0.9375\left(1-x^2\right)^2$ if $|x|\le 1$」を核関数に用いるようである（kern2d、intensityヘルプファイル、オブジェクトkernquartを参照）。

[3] 実際はbw=0.1とした。

結果の濃淡プロットは図 6.12 の右である。

空間点パターンの強度を S+SPATIALSTATS により 3 次元プロットするには、Trellis 作図関数 `wireframe` を以下のように用いる：

```
> trellis.device(color=F)
> mgrid <- expand.grid(x=maple.bing$x, y=maple.bing$y)
> mdf <- data.frame(x=mgrid$x, y=mgrid$y,
+      z=c(maple.bing$z))
> wireframe(z~x*y, data=mdf, drape=T)
```

白黒の作図を行うため、`trellis.device` を `color=F` として用いた。結果は図 6.13 に示す。

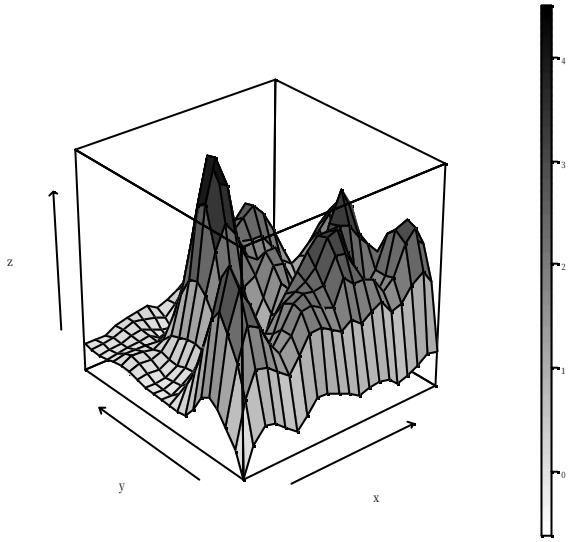


**図 6.13.** バイニングメソッドで推定したランシングウッズのカエデの強度を 3 次元プロットしたもの。

この節で行なったすべての強度推定とその視覚化の技法により、ランシングウッズのカエデの強度はランダムパターンの強度より大きく変化していることが分かった。北の両角にカエデが繁殖していないことが原因と考えられるが、これはヒッコリーとの相互作用によるものと思われる。

### 6.3.2 K関数

空間点過程の2次特性は、2点の相互作用、あるいは空間上の関係が空間内でどのように変化するのかを記述する量である。これらの特性は通常、空間点パターンの2次強度により記述される。2次特性を記述するもう一つの方法がK関数（*K-function*）である：

$$K(d) = \lambda^{-1} E[\text{点の数} \leqq \text{任意の点からの距離 } d]$$

ここで$\lambda$は強度、$E[\ ]$は期待値を表わす。K関数の利点は、$K(d)$の理論値が空間点過程の幾つかの便利なモデルについて分かっていることである。例えば、空間依存の無い均質なプロセスのK関数は$\pi d^2$である。クラスタ化するプロセスでは近距離での点の数が多くなることが予想されるので、小さな$d$に対して$K(d) > \pi d^2$となる。同様に、規則的な点パターンは$K(d) < \pi d^2$となることが予想される。

Ripley [1976]によるK関数の推定量は：

$$\hat{K}(d) = n^{-2}|A|\sum\sum_{i \neq j} w_{i,j}^{-1} I_d(d_{i,j})$$

ここで、$n$は面積が$|A|$の領域$A$にある点の総数、$d_{ij}$は$i$番目の点と$j$番目の点との距離、$w_{ij}$は点$i$を中心とした半径$d_{ij}$の円のうち、領域$A$内に入る部分の面積の、円全体面積に対する割合である。$I_d(d_{ij})$は$d_{ij} < d$のとき1となる指示関数である。この推定量は境界効果を考慮した補正を行なっている。

ランシングウッズのデータに対してRipleyのK関数を、S+SPATIALSTATS関数`Khat`を用いて求めている：

```
> lans.khat <- Khat(lansing.spp)
> lans.d <- lans.khat$values[,"dist"]
> lines(lans.d, pi*lans.d**2)
> maple.khat <- Khat(maple.spp)
> maple.d <- maple.khat$values[,"dist"]
> lines(maple.d, pi*maple.d**2)
> hick.khat <- Khat(hick.spp, plot=F)
```

関数`Khat`は上で定義した$K(d)$の推定値$\hat{K}(d)$を計算し、プロットする。ヒッコリーに対しては引数plot=Fを使用した。`Khat`の返す行列`values`には距離の列と、対応する$\hat{K}$の値が含まれる。上の`Khat`の呼び出しによって描かれた2つのプロットは図6.14に示した。CSRのK関数を参照のために加えた。左はランシングウッズ全データのK関数であり、CSR

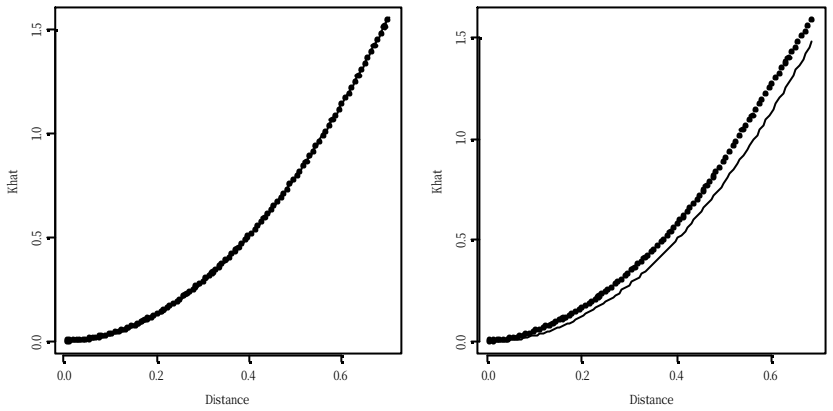


**図 6.14.** ランシングウッズのすべての木の K 関数（左）とカエデだけの K 関数（右）。CSR の K 関数を表わす線を加えた。

の K 関数の期待値に非常に近い。右はカエデが $K(d) > \pi d^2$ となることを示しており、クラスタ化が起こっている。

S+SPATIALSTATS には、分散を安定化する変換 $L(d) = \sqrt{K(d)/p}$ の $\hat{L}(d)$ を計算する関数がある。均質なポアソン点過程の $L(d)$は直線になる。以下でヒッコリーデータのを $\hat{L}(d)$ プロットしてみる：

```
> par(mfrow=c(1,1))
> hick.lhat <- Lhat(hick.spp)
> abline(0,1)
```

関数 abline は現在のプロットに切片 0、傾き 1 の直線を描き加える。図 6.15 は、ヒッコリーデータが短距離でクラスタ化していることを示している。

6.2.2 節で $\hat{G}$ に対して行なったように、クラスタ化が有意であるかを評価するために、K 関数のプロットにシミュレーション・エンベロープと平均を描き加えることができる。関数 Kenv を用いて両種の $\hat{K}$ にエンベロープを描き加える：

1. 評価を明確にするために距離を 0.25 以下に限定し、maple.khat をプロットする：

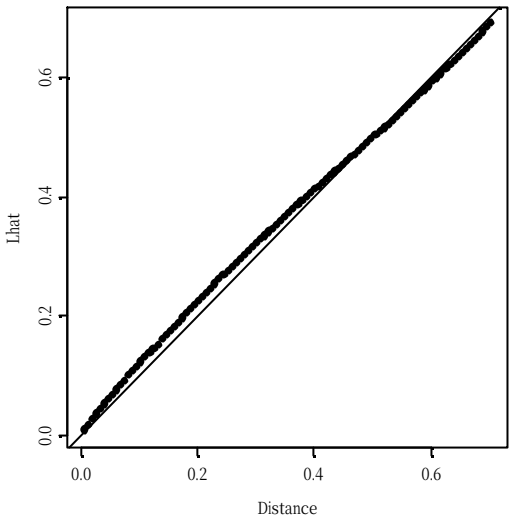


**図 6.15. ランシングウッズのヒッコリーの $L(d)$に CSR 過程の $L(d)$を描き加えたもの。**

```
> par(mfrow=c(1,2))
> plot(maple.khat$values
+      [maple.khat$values[,1] < 0.25,])
```

2. **カエデのシミュレーション・エンベロープを計算し、プロットする：**

```
> maple.kenv <- Kenv(maple.spp, nsims=25, add=F)
> ind <- (maple.kenv$dist < 0.25)
> lines(maple.kenv$dist[ind],maple.kenv$lower[ind])
> lines(maple.kenv$dist[ind],maple.kenv$upper[ind])
```

3. **同じ手続きをヒッコリーに対して行なう：**

```
> plot(
+    hick.khat$values[hick.khat$values[,1]<0.25,])
> hick.kenv <- Kenv(hick.spp, nsims=25, add=F)
> ind2 <- (hick.kenv$dist < 0.25)
> lines(hick.kenv$dist[ind2],hick.kenv$lower[ind2])
> lines(hick.kenv$dist[ind2],hick.kenv$upper[ind2])
```

**図 6.16 は 2 種共にクラスタ化している証拠を示している。**

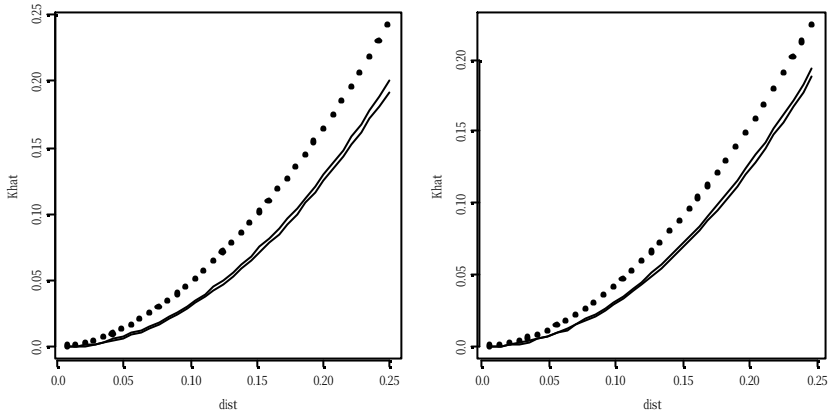


**図 6.16.** ランシングウッズの 2 種に対する K 関数の推定値にシミュレーション・エンベロープを描き加えたもの。

## 6.4 点パターンのシミュレーション

空間点パターンの解析手法の多くはモンテカルロ法が基本になっている。S+SPATIALSTATS では、以下の 5 つの基本的なモデルに従う 2 次元のランダムな点の集合を関数 `make.pattern` によって生成することができる（**ポアソン**（*Poisson*）、**二項**（*binomial*）、**SSI**（*SSI*）、**Strauss**（*Strauss*）、**クラスタ**（*cluster*））。例えば、単位正方形内の CSR 点過程の実現値を発生させ、プロットするには：

```
> par(mfrow=c(1,1))
> plot(make.pattern(n=100))
```

オプション `process="binomial"` が `make.pattern` のデフォルトであり、図 6.17 は点の個数が 100 の二項点過程（CSR 点過程）の実現値である。これは点の数を固定した条件のもとでの均質なポアソン点過程に等しい。CSR 過程は `process="poisson"` とすることでも生成できる。この場合、引数 `lambda` を `n` の代わりに与える必要がある。すなわち、点の数ではなく強度に注目するのである。これらの点パターンは本章でこれまで論じてきた統計量のシミュレーション・エンベロープを作成する際に使

用できる。

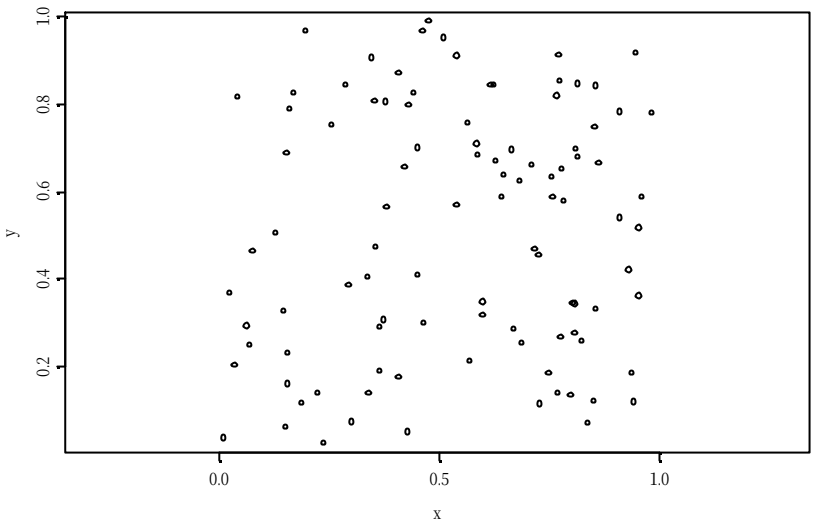


**図 6.17.** 関数 make.pattern により発生させた、完全ランダム（二項）点過程に従う 100 個の点の実現値。

データに規則性を示す根拠が見つかった場合、データのモデリングを行なう際には SSI 点過程、あるいは Strauss 点過程の生成を試みて欲しい。SSI（*Sequential Spatial Inhibition*）点過程はどの 2 点も与えられた**禁止半径**（*radius of inhibition*）以上離れた点パターンである。禁止半径が 0.1 の SSI 点過程に従う点を 100 個発生させ、プロットしてみる：

```
> par(mfrow=c(1,2))
> plot(make.pattern(100, process="SSI", radius=.1,
+      boundary=bbox(x=c(0,1),y=c(0,2))))
```

図 6.18 の左が発生させた SSI 点過程である。make.pattern の中で 1×2 の長方形の有界領域を指定した。

"Strauss"点過程（Ripley, 1981, p.166）は禁止半径内であっても、ある与えられた比率で点の発生を許す点パターンを発生させる。この比率は引数 cpar で指定する。デフォルトは 0 である。以下がその点パターンの生成手順である：

1. 有界領域 $A$ 内で 1 点を発生させる。

2. 次の点を発生させ、確率 $c^s$ でこの点を残す。ここで $c$ は [0,1] 上の

禁止パラメータ、s はその新しい点から半径 $r$ 以内に現存する点の数である。

禁止半径 0.1、禁止パラメータ.5 の Strauss 点パターンを発生させ、プロットしてみる：

```
> plot(make.pattern(100, process="Strauss",radius=.1,
+      cpar=.5, bundary=bbox(x=c(0,1),y=c(0,2))))
```

図 6.18 の右にその結果を示した。

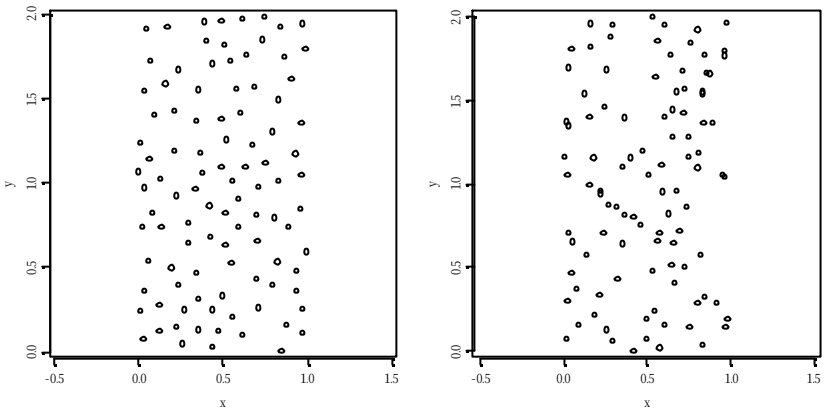


**図 6.18.** 関数 make.patten により発生させた SSI 点パターン（左）と Strauss 点パターン（右）。

**警告：** 禁止半径が大きすぎると指定した数の点を発生させることが不可能になるか、不可能に近くなる場合がある。

オプション process="cluster"は、ポアソン親子点過程を生成する。引数 cpar は親過程の平均として、引数 radius はクラスタの半径として使用される。単位正方形内のポアソンクラスタ点過程を発生させてみる：

```
> plot(make.pattern(100, process="cluster",
+      radius=.05, cpar=15))
> plot(make.pattern(100, process="cluster",
+      radius=.15, cpar=15))
```

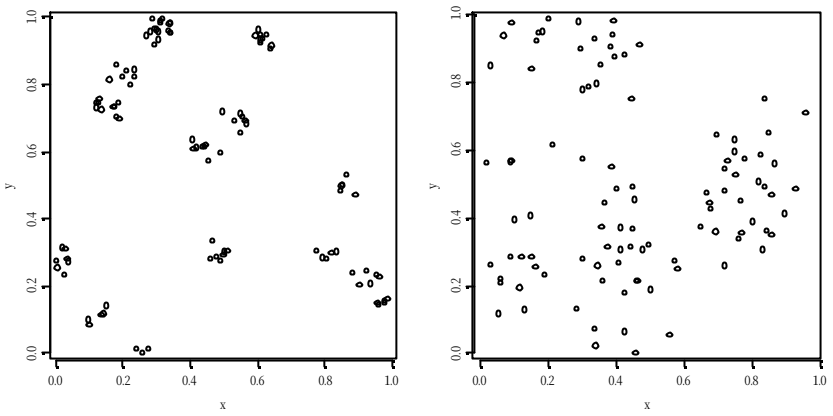


**図 6.19.** 関数 `make.pattern` により発生させたポアソンクラスタ点過程の 2 つの実現値。

図 6.19 はどちらも親ポアソン点過程の平均が 15 である。左の方が子点過程の半径が小さいため、よりはっきりしたクラスタが現われている。