損保アクチュアリーのためのＲによる計算保険数理
＜ASTIN 関連研究会＞

あいおいニッセイ同和損害　　渡辺　重男　君

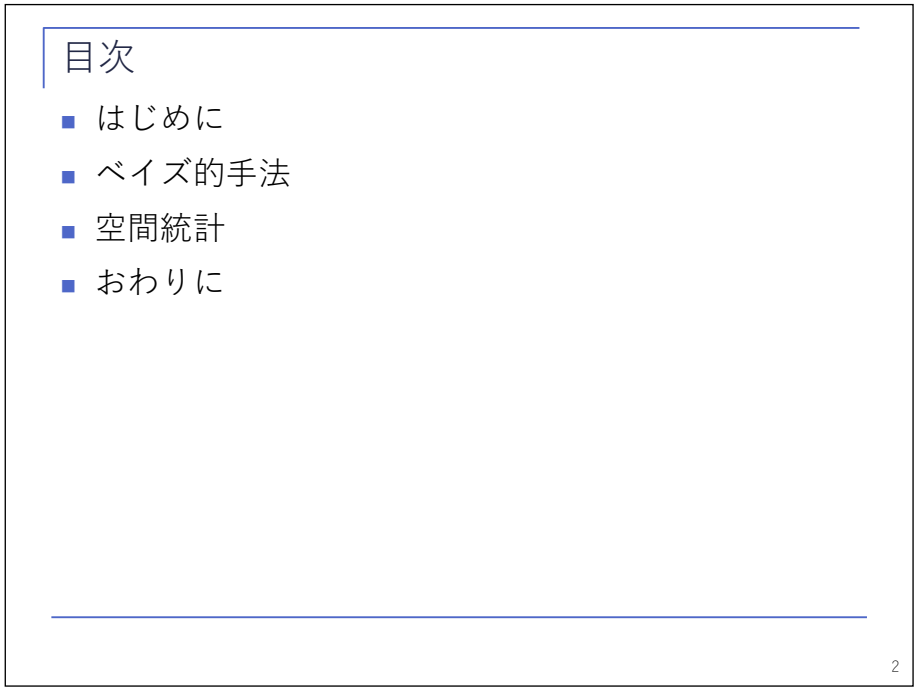
共栄火災海上保険　　佐野　誠一郎　君

【司会】　時間となりましたので、セッション A-1、ASTIN 関連研究会による「損保アクチュアリーのためのＲによる計算保険数理」を開始します。発表者は、あいおいニッセイ同和の渡辺重男さん、共栄火災の佐野誠一郎さんのお二人です。

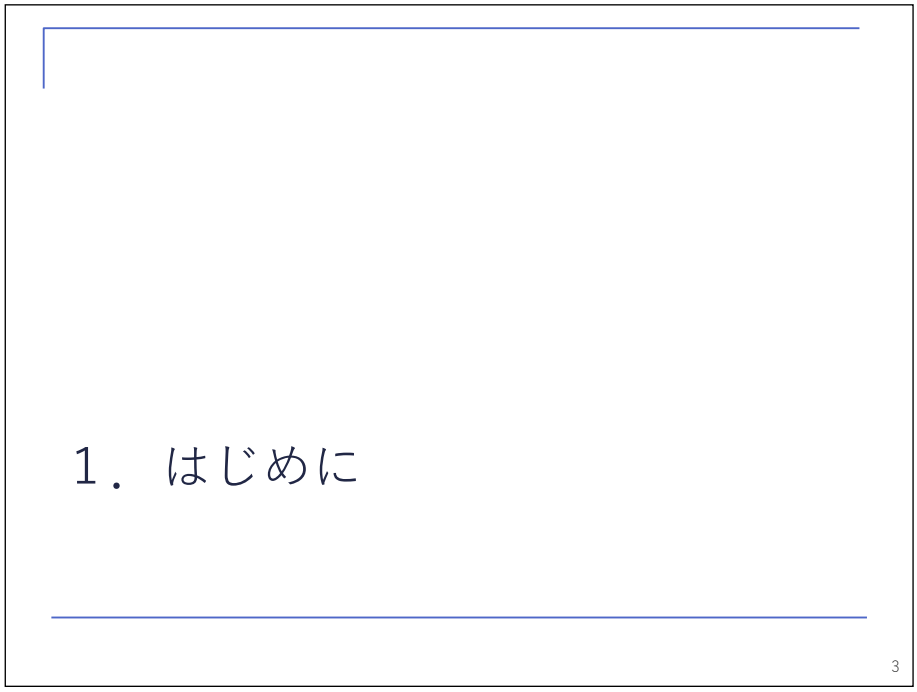
なお、質疑応答の時間は、お二人が発表した後に、まとめて取らせていただきます。また、Slido に投稿された質問に対しても、その際に回答することとしています。リモートで視聴されている方で質問のある方は、Slido への投稿をお願いします。

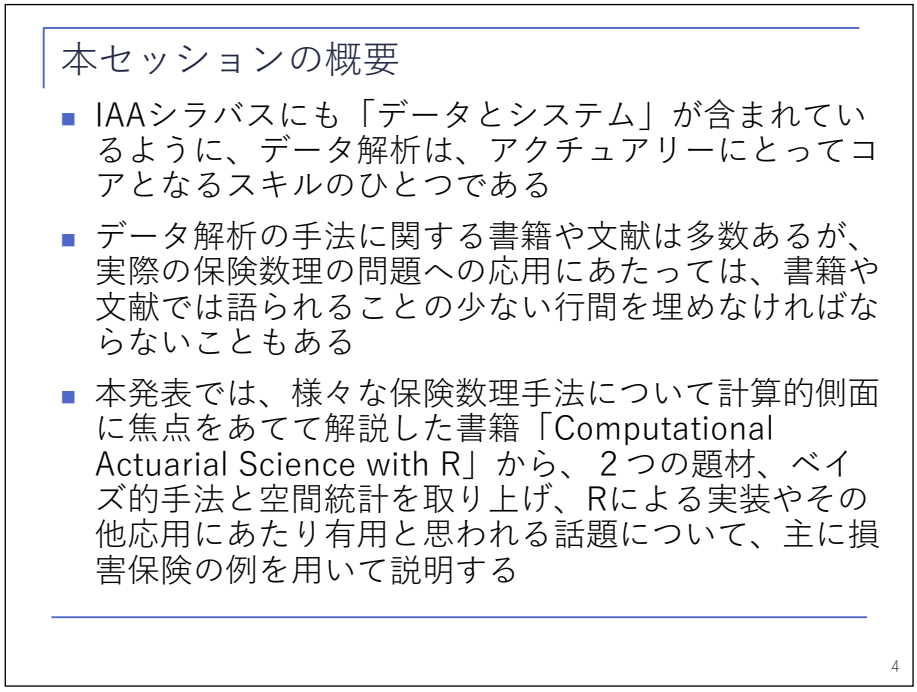
それでは、渡辺さん、よろしくお願いします。

【渡辺】　はい。ASTIN 関連研の渡辺です。このセッションでは、「損保アクチュアリーのためのＲによる計算保険数理」というタイトルで、ASTIN 関連研の活動として輪読を行った書籍の内容をベースにして、お話をしたいと思います。

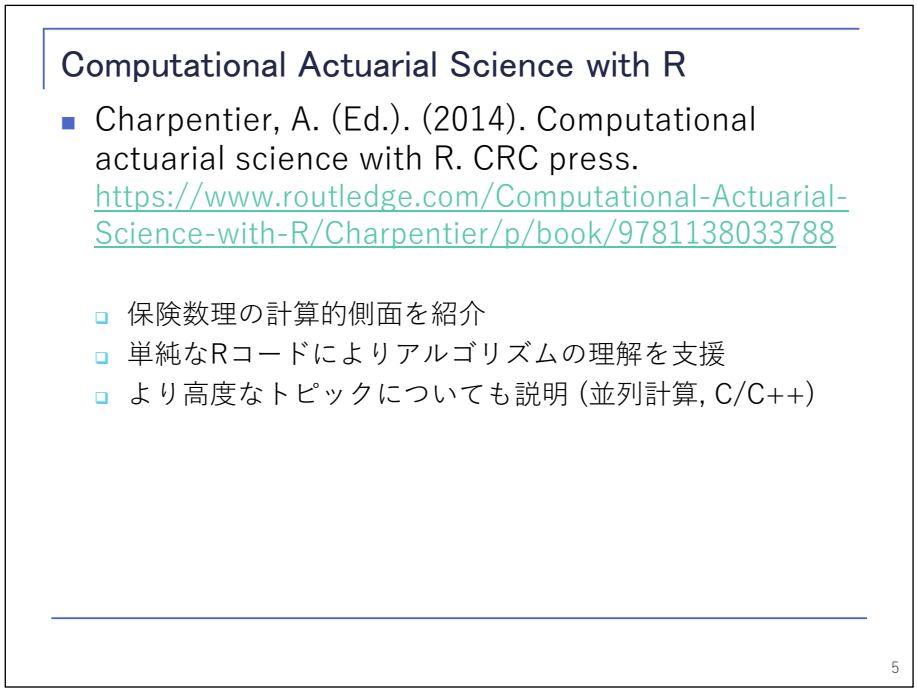


本日の発表内容は、スライドの通りです。初めに、本書について簡単にご紹介しまして、その中から二つのテーマ、ベイズ的手法と空間統計を選び、本書の内容をベースにお話をします。前半を私から、後半は共栄火災の佐野さんからお話しします。





こちらは、本セッションの概要です。プログラムに記載の通りです。



今日ご紹介する本はこちらの書籍です。この本は、「保険数理の計算的側面について紹介する」ことを目的としています。保険数理を学ぶにあたって数式を追って紙と鉛筆で理解するということ、それはそれで尊いことですが、本書ではこれとは違うアプローチを提案しています。

本書では、既存のモデルを理解してその改善に取り組むことで、より創造的な活動、例えば論文執筆などにつなげるというアプローチが教育上有用であるというように考えて、保険数理における新しい分野として計算保険数理（Computational actuarial science）という用語を使っています。

このため、本書の中で保険数理に関する複雑な計算がＲを使って簡単にできるということを示そうとして、

アルゴリズムの理解を助けるために簡単なコードを使って例を幾つか示しています。

また、更に進んだ内容として計算上重要な、例えば並列計算や、C 言語を使った高速化など、より高度なトピックも紹介しています。

Computational Actuarial Science with R (続)

	Chapter 1 Introduction
Part I Methodology	Chapter 2 Standard Statistical Inference
	Chapter 3 Bayesian Philosophy
	Chapter 4 Statistical Learning
	Chapter 5 Spatial Analysis
	Chapter 6 Reinsurance and Extremal Events
Part II Life Insurance	Chapter 7 Life Contingencies
	Chapter 8 Prospective Life Tables
	Chapter 9 Prospective Mortality Tables and Portfolio Experience
	Chapter 10 Survival Analysis
Part III Finance	Chapter 11 Stock Prices and Time Series
	Chapter 12 Yield Curves and Interest Rates Models
	Chapter 13 Portfolio Allocation
Part IV Non-Life Insurance	Chapter 14 General Insurance Pricing
	Chapter 15 Longitudinal Data and Experience Rating
	Chapter 16 Claims Reserving and IBNR
	Chapter 17 Bibliography

こちらはこの本の目次です。全体で四つのパートに分かれていまして、パート I では、統計やモデリングの手法についての説明がされています。今日取り上げる二つのテーマもこの中で説明されています。それからパート II、III、IV が、それぞれ、生保、ファイナンス、損保についての内容になっています。

Computational Actuarial Science with R (続)

- 計算に必要なコードやデータは利用可能
  - コードは本書に掲載
  - データはRのパッケージCASdatasetsとして提供

```
install.packages("CASdatasets",
    repos = "http://cas.uqam.ca/pub/",
    type="source")
```

※事前にパッケージsp, xtsをインストールしておく必要がある

- 「計算保険数理を学ぶ最良の方法は、計算保険数理を実践することである。そして、計算保険数理を実践する最良の方法の一つは、おそらく既存のモデルから始め、これを使って実験することである。」

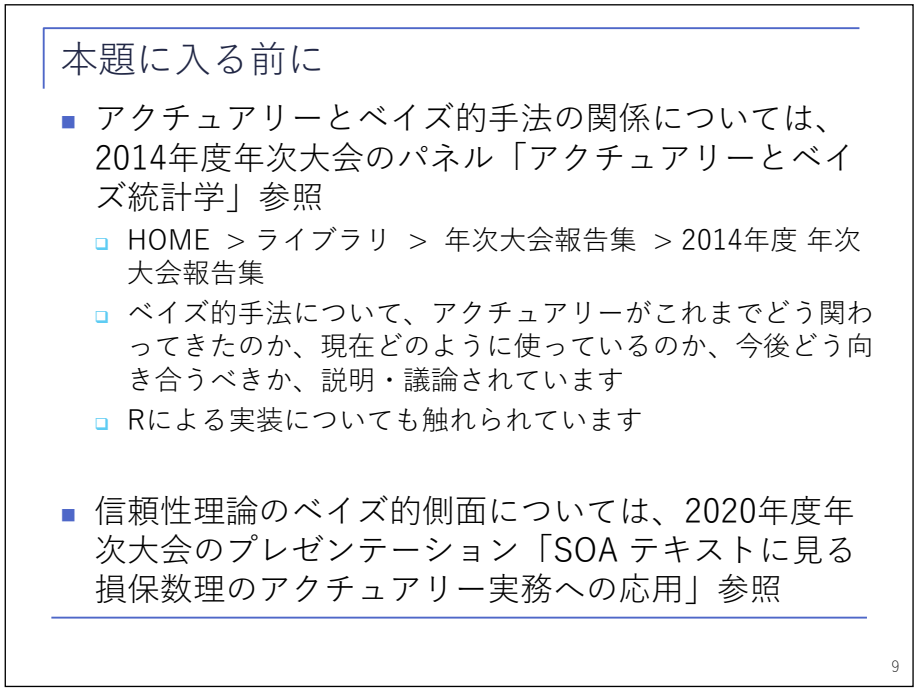
7

この本の特徴としまして、計算に必要なコードやデータが全部利用できるというところがあります。コードは本に掲載されていますし、データは R のパッケージとして提供されています。残念ながら、CRAN のパッ

ケージではないので、ここにあるようなコードを使って自分でインストールする必要があるのですけれども、このようにして利用することができます。

コードの下に、本書からの引用を並べています。いろいろ書いてありますけれども、「まずは手を動かしてみましょう」ということです。そのために必要なデータやコードは全部提供されていますので、この後ご紹介する二つのテーマについても、ぜひ、皆さん、実際に手を動かしてみてもらえればと思っております。

続いてベイズ統計の話に入ります。

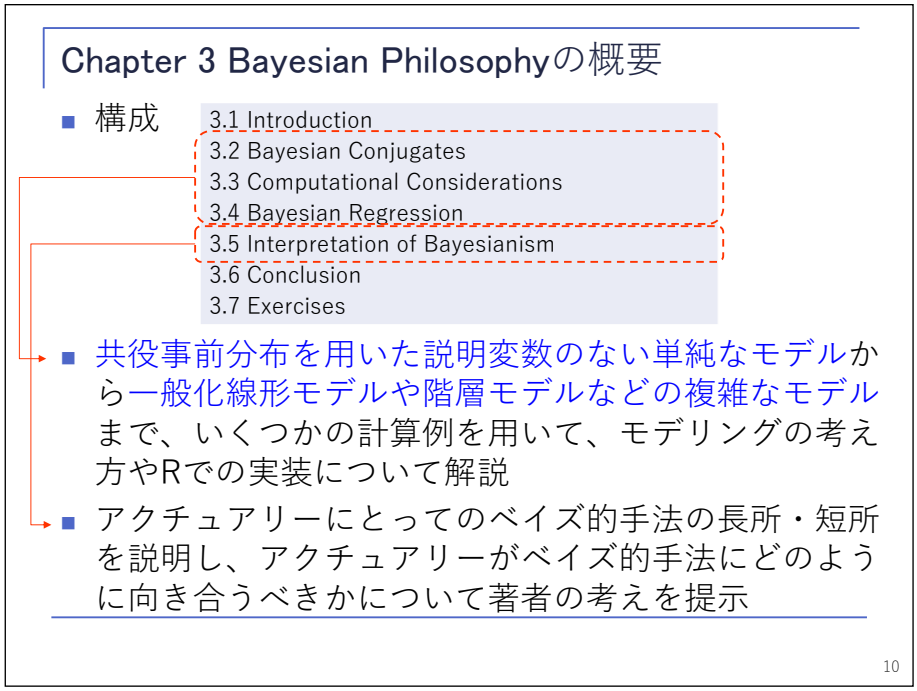


まず、本題に入る前に、これまでの ASTIN 関連研による年次大会の発表でベイズに関連するものを二つご紹介します。

一つは、2014 年の年次大会のパネル「アクチュアリーとベイズ統計学」です。基本的な考え方から始まって応用まで、幅広い内容を 3 人のパネリストがカバーしています。

「なぜベイズ的手法を使うのか」といったような基本的なところは、今日のプレゼンテーションで触れませんので、ぜひ関心のある方は、こちらをご覧いただければと思います。

あともう一つ、2020 年のプレゼンテーションの中で、信頼性理論のベイズ的側面について扱っていますので、関心のある方はご覧いただければと思います。



本章のなかでベイズを扱っているところは、第 3 章の Bayesian Philosophy というところです。

前半、最初の三つのセクションで、解析的に計算できる簡単なモデルから始まって、段々複雑なモデルに進んで、R による分析の例が示されています。

後半、Interpretation of Bayesianism のセクションでベイズの長所・短所について説明した上で、「アクチュアリーは、ベイズ的手法をどう使っていけばいいのか」というところについて著者の考えが説明されています。

計算例一覧

節	概要	尤度/事前分布	事前分布の種類	計算
3.2.1	男女別の出生数から男児が生まれる確率を推定	$Y \sim Bin(n,p)$ $p \sim \mathrm{B}(a,b)$	無情報（一様分布), 主観確率	解析的に計算
3.2.2	5年間の損害率から翌年度の損害率を予測	$\log Y \sim N(\mu,\sigma)$ $\mu \sim N(\mu_0,\sigma_0)$	主観確率	解析的に計算
3.2.3	5年間無事故という情報から2機の飛行機が衝突する確率を予測	$Y \sim Be(p)$ $p \sim \mathrm{B}(a,b)$	無情報（一様分布, Jeffreyの無情報事前分布）	解析的に計算
3.2.5	契約別実績クレーム件数からクレーム件数を予測	$Y \sim Be(p)$ $p \sim \mathrm{B}(a,b)$	経験事前分布	Bühlmann モデル
3.3.4	(3.2.2と同じ)	$\log Y \sim N(\mu,\sigma)$ $\mu \sim N(\mu_0,\sigma_0)$ $\sigma \sim InvGamma(k_0,\theta_0)$	主観確率	MCMC (Rで実装)
3.3.5				MCMC (JAGS, Stan)

11

こちらが、前半部分で取り上げられている計算例の一覧です。最初の方は、共役事前分布を使って解析的に計算できる、よくベイズの初級者向けの本に書いてあるような例が並んでいますが、段々複雑なモデルになっていきます。

計算例一覧 (続)

節	概要	尤度/事前分布	事前分布の種類	計算
3.4.1	個人別の性別・体重から身長を予測	$Y = \mathbf{X}\boldsymbol{\beta} + \epsilon, \epsilon \sim N(0,\sigma^2)$ $\beta \vert \sigma^2 \sim \mathcal{N}(\beta_0, \sigma^2 M_0^{-1})$ $\sigma^2 \sim InvGamma(a_0,b_0)$	主観確率	解析的に計算
3.4.2	クレームデータから弁護士介入の有無を予測	$Y \sim Be(\pi(X)), \pi(X) = \Phi(\mathbf{X}\boldsymbol{\beta})$ $\beta_i \sim U(-\infty,\infty)$	無情報（非正則一様分布）	MCMC (Rで実装)

- 以下、
  「5年間の損害率から翌年度の損害率を予測する」
  という単純な問題を例に、Rによる実装に焦点をあてて説明する

12

次のページにも続きまして、このページの 1 行目の例は、説明変数が入っているようなモデルの例です。2 行目の例は、ベルヌーイ分布のパラメータがリンク関数Φを介して線形予測子につながっているモデルで、要は GLM です。

計算例一覧

節	概要	尤度/事前分布	事前分布の種類	計算
3.2.1	男女別の出生数から男児が生まれる確率を推定	$Y \sim Bin(n,p)$ $p \sim \mathrm{B}(a,b)$	無情報（一様分布），主観確率	解析的に計算
3.2.2	5年間の損害率から翌年度の損害率を予測	$\log Y \sim N(\mu,\sigma)$ $\mu \sim N(\mu_0,\sigma_0)$	主観確率	解析的に計算
3.2.3	5年間無事故という情報から2機の飛行機が衝突する確率を予測	$Y \sim Be(p)$ $p \sim \mathrm{B}(a,b)$	無情報（一様分布，Jeffreyの無情報事前分布）	解析的に計算
3.2.5	契約別実績クレーム件数からクレーム件数を予測	$Y \sim Be(p)$ $p \sim \mathrm{B}(a,b)$	経験事前分布	Bühlmann モデル
3.3.4	(3.2.2と同じ)	$\log Y \sim N(\mu,\sigma)$ $\mu \sim N(\mu_0,\sigma_0)$ $\sigma \sim InvGamma(k_0,\theta_0)$	主観確率	MCMC（Rで実装）
3.3.5				MCMC（JAGS, Stan）

11

以下、前のページの 2 行目と、あと一番下で扱っている「5 年間の損害率から翌年度の損害率を予測」する例を取り上げ、R による実装に焦点を当ててご説明していきます。

損害率の分布の分析

- 5 年前に販売開始した商品について、翌年度の損害率 Lの分布を推定
  - 5 年間の実績損害率 L_i（トレンド、保険料水準調整後）

    95.8%　61.4%　97.7%　96.1%　75.6%

  - 損害率は対数正規分布に従うと仮定

    $$\log L \sim N(\mu,\sigma^2)$$

13

こちらが、その具体的な例です。5 年分の損害率がこのように与えられているときに、これを基に、損害率の分布に対数正規分布を仮定して、翌年度の損害率の分布を予測するモデルを考えてみよう、という例です。

損害率の分布の分析（非ベイズ的手法）

- パラメータの推定

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \log(L_i)$$

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n} (\log(L_i) - \hat{\mu})^2$$

```
Li <- c(0.958, 0.614, 0.977, 0.961, 0.756)
Li.meanlog <- mean(log(Li))
Li.sdlog <- sd(log(Li))
Li.CI.l <- qlnorm(0.025, meanlog=Li.meanlog, sdlog=Li.sdlog)
Li.CI.u <- qlnorm(0.975, meanlog=Li.meanlog, sdlog=Li.sdlog)
```

- 計算結果（パラメータ、Lの95%信頼区間）

```
> c(Li.meanlog, Li.sdlog)
[1] -0.1746863  0.2046574
> c(Li.CI.l, Li.CI.u)
[1] 0.5622507 1.2541209
```

14

まず、おさらいとして、ベイズを使わないやり方でやってみます。損害率の対数の平均と標準偏差を計算して、これを対数正規分布のパラメータとするという、ごく簡単な方法です。

計算結果は一番下にある通りの結果になっています。

損害率の分布の分析（ベイズ的手法）

- ベイズ的手法
  - 損害率 L の分布のパラメータ$\phi = (\mu, \sigma)$の分布を考える
  - データを観測する前の知識に基づくϕの分布（事前分布）を$p(\phi)$とする
  - データe（この場合は実績損害率）が与えられたとき、ϕの分布（事後分布）は以下のとおり

$$p(\phi|e) = \frac{p(e|\phi)p(\phi)}{\int p(e|\phi)p(\phi)d\phi}$$

$p(e|\phi)$：データeの尤度

$p(\phi)$：ϕの事前分布

  - 事後分布を用いて、Lの予測分布も計算できる

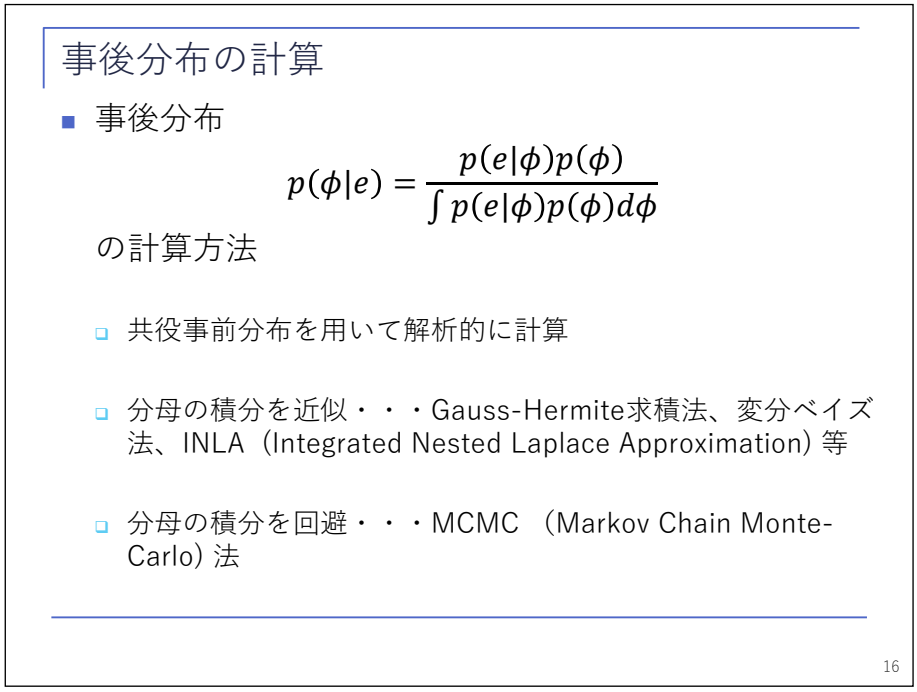
15

これに対して、ベイズ的手法をどのように考えるかということですが、分布のパラメータを確率変数と考えて、その事前分布を考えます。事前分布とは、データを観測する前の知識、事前の信念というように言われたりもしますけれども、これに基づくパラメータの分布を表しています。

この事前分布と尤度、すなわち、あるパラメータの下での損害率の確率分布を仮定すれば、ベイズの定理に従って損害率の観測値を得たときのパラメータの分布、事後分布が得られると、このような仕組みになっ

ています。

また、スライドの最後にあるように、パラメータの事後分布にあるパラメータの下での損害率の分布を掛け合わせることで、翌年の損害率の分布を得ることができます。



問題は、この事後分布をどのように計算するか、というところで、特にこの分母の積分の計算が厄介です。幾つかやり方はありますが、一つは共役事前分布を使って解析的に計算するという方法です。これは、できるケースが限られています。二つ目は、分母の積分を何とか近似してやろうということで、本書の中では Gauss-Hermite 求積法が挙げられていますが、それ以外にも、変分ベイスや INLA など、他の手法もあります。

特に INLA については、後半の発表を担当する佐野さんが論文を書かれていまして、本年度優秀論文でも表彰されていますので、是非皆さん、ご覧ください。

三つ目の方法は、分母の積分をそもそもやらないで済む方法はないかということで、これが、この後ご紹介する MCMC、Markov 連鎖 Monte-Carlo 法ということになります。この後続いてご説明をします。

MCMC

- MCMC =Markov Chain Monte-Carlo

求める確率分布を均衡分布として持つマルコフ連鎖を作成することによって確率分布のサンプリングを行う種々のアルゴリズムの総称
(Wikipediaより)

- $\int p(e|\phi)p(\phi)d\phi$を計算せずに事後分布$p(\phi|e)$からのサンプリングを行う
  - 適当な初期値を起点にマルコフ連鎖を反復しサンプリングを行うことで、$p(\phi|e)$からのサンプルが得られる
  - 開始直後のサンプルは初期値の影響を受けるため除外する必要がある（burn-in, warm-up）
  - 収束の確認が必要、自己相関に留意
- アルゴリズム
  - メトロポリス・ヘイスティング法（本書で説明）、ギブスサンプリング法、ハミルトニアンモンテカルロ法、等

17

「MCMC とは何か」ということで、ウィキペディアの定義をそのまま持ってきていますが、要は、どの状態からスタートしてもマルコフ連鎖をつなげていけば最終的には同じ分布に落ち着くという性質がありますので、これを利用したやり方です。これを利用すれば、積分の計算をせずに事後分布からのサンプリングができるということです。目的とする事後分布が得られるようなマルコフ連鎖を設計してやって、どんどんサンプリングを繰り返していけばいいというようなやり方です。

スライドに注意しないといけないところを並べています。まずは、いずれ同じ分布に落ち着くといっても、開始直後のサンプルは初期値の影響を受けますので、それを除いてやる必要があります。これを burn-in や warm-up などといいます。

2 点目は、マルコフ連鎖なのでサンプリングは独立ではないという点です。サンプルには自己相関があります。MCMC を使う上ではこの点に注意する必要があります。

具体的な計算方法については、いろいろなアルゴリズムがありますが、今日は時間もないので説明を省略します。

本書の中では、いろいろなアルゴリズムがある中でも、特にメトロポリス法を取り上げて、R での実装も紹介されていますが、一方で、間違いの基になるので自前でプログラムを組むことはあまり推奨しないというようなことも言っています。

それよりは、計算のための汎用ツールが幾つもあるので、これを使うのが一般的だと勧めています。主なものをここに挙げていますが、本書は、この中でも JAGS と Stan を取り上げて、R で使うときのモデリングの方法などを紹介しています。

これらのツールはRとは別のプログラムで､別途Rから使うためのインターフェイスも開発されています。括弧の中に書いてあるのがそれで、JAGS であれば「runjags」、Stan であれば「rstan」を使います。

これらのツールの使い分けについて、本書の中では「JAGS の方が、より機能が豊富だ」と書かれています。例えば､「Stan ではパラメータが離散的な場合には、うまく扱えない」ということだそうです。一方で､「Stan の方が、より高速だ」ということも書かれています。

説明を補足すると、JAGS も Stan も MCMC を計算するためのコードを自分で書くのですが、Stan ではこれをコンパイルする必要があります。最初に計算するときにはコンパイルの時間がかかるので、Stan の方が少し長くかかることもありますが､同じコードを使って 2 回目以降データを変えて計算をするときには Stan の方が早くなるというような関係にあります。

また、スピード以外の利点として、Stan の方がサンプルの自己相関が小さいというようなことも挙げられています。

rstan

- 以下、rstanについて説明
  - 前提とする環境：Windows 11, R-4.2
- 公式サイト
  - https://mc-stan.org/rstan/
- インストール
  - CRANのrstanはR-4.2では動かない
  - rtools42のインストール：https://cran.r-project.org/bin/windows/Rtools/rtools42/rtools.html
  - rstanのインストール

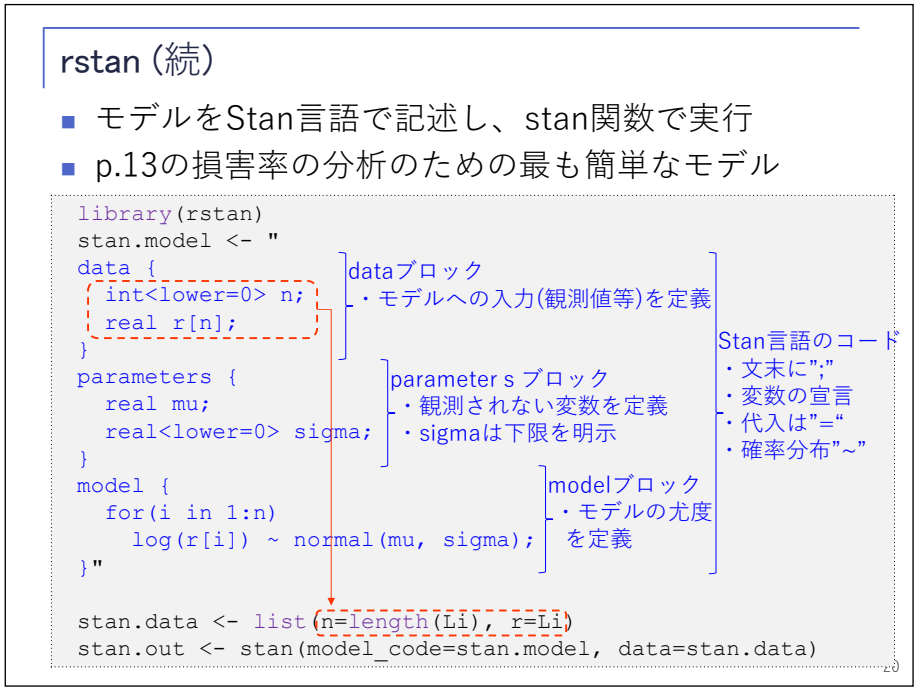
```
> install.packages("StanHeaders", repos = c("https://mc-
  stan.org/r-packages/", getOption("repos")))
> install.packages("rstan", repos = c("https://mc-
  stan.org/r-packages/", getOption("repos")))
```

19

以下、これから Stan を使った計算について説明します。まず、rstan をインストールするのですが、ここに引っかかるポイントがあります。

rstan は CRAN に登録されていますが、そこに登録されているバージョンが最新ではないため、最新バージョンの R では動かないと問題があります。この問題を回避するために、公式サイトから直接インストールするということが必要になってきます。

あともう一つ、先ほど、Stan はコードをコンパイルする必要があるとお話ししましたが、コンパイルするために rtools というツールを使いますので、これもインストールする必要があります。これは CRAN のサイトからダウンロードできます。ダウンロードしたインストーラを起動して、指示に従ってインストールすれば終わりです。その上で、最後、このスライドに書いてあるこのコードを R 上で入力すれば、rstan がインストールできます。



インストールができたら、今度はコードを実行します。Stan 言語という、MCMC の計算を実行するための言語でモデルを記述して、これを R のオブジェクトにテキストデータとして突っ込んでやる必要があります。

このスライドは、スライド p. 13 の損害率の例について計算するためのコードです。青字部分が Stan 言語のコードです。これをテキストデータとして stan. model というオブジェクトに代入しています。

そして下から 2 行目で、分析に使うデータを stan. data というオブジェクトに代入しています。

一番下の行で、Stan 関数を使ってこのモデルとデータを与えて stan. out というオブジェクトに結果を代入しています。

最低限必要なことはこれだけで、JAGS などと比べてシンプルなコードかと思います。

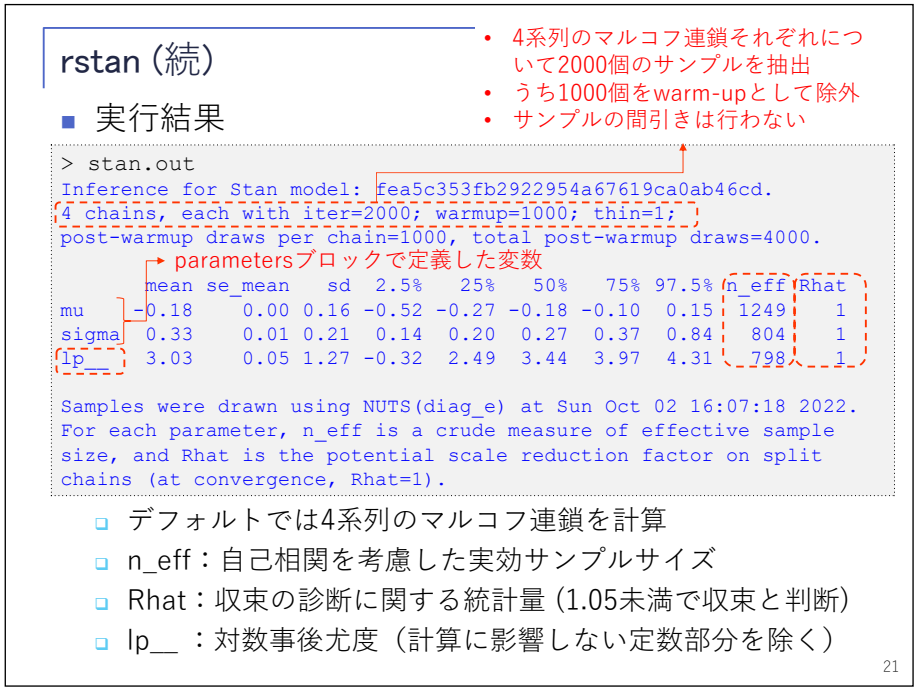
青字部分は、R とは違う言語ですが、割と似たような書き方になっています。違うところとして注意しないといけない部分は、スライドの右端のとおりです。文末の「;」は、割と落としがちです。あと、変数の宣言が必要になるというところと、代入に「=」を使うというところ、この辺りが主に R との違いとして注意しないといけないところです。あと、確率分布を表現するときにチルダ（~）の記号を使えるというところも Stan 言語の特徴です。

コードは、三つのブロックに分かれています。一番上は data ブロックで、観測値などモデルに対して外部から与えるデータの形式を定義しています。この例ですと、まず n が 0 以上の整数値を取る変数です。それから、もう一つ、r が n 次元の実数値の配列です。このようにデータの形式を定義したうえで、実際に Stan 関数に与えるデータも、同じ形式のリストとして定義してやる必要があります。

次のブロックが parameters ブロックです。data ブロックで定義した入力データ以外の変数を定義しており、実行結果の出力にはここで定義した変数に関する情報が含まれます。ここでは、事前分布のパラメータの mu と sigma を定義しています。

最後が、model ブロックで、モデルの尤度を定義するブロックです。

このコードでは、n 個の観測値それぞれについて、損害率の対数がパラメータ mu, sigma の正規分布に従うものとして尤度を計算し、これを全部足し合わせています。



次のスライドは、実行結果です。青字の部分が R の出力です。青字の 2 行目を見ると、どのような計算をしているかということが書かれています。具体的には、四つのマルコフ連鎖があり、それぞれについて 2,000 個のサンプルを計算しています。そのうち 1,000 個を warm-up として除外しています。自己相関を軽減するためのサンプルの間引きは行いません。

1 行空けて 4 行目から下に、事後分布の平均や標準偏差、分位点が並んでいます。表示されている変数を見ると、mu と sigma は先ほどの parameters ブロックで定義した変数です。一番下に lp で始まる見慣れない変数が入っていますが、これは対数事後尤度を表しています。ただし、対数尤度の中でも計算に直結しないような定数部分を省略していますので、本来の値とは違っています。本来の値を計算する方法もあるのですが、ここでは説明を省略します。

右端の 2 列に、n_eff と Rhat という二つの変数があります。n_eff は実効サンプルサイズで、左から 3 列目にある平均の標準誤差の計算に影響します。自己相関があるので、本来のサンプル数よりも小さい値になっています。

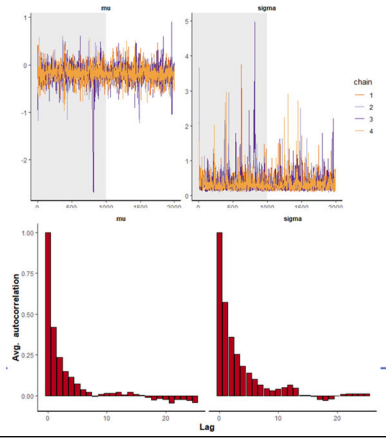
右端の Rhat は収束の診断に関する統計量で、青字の一番下の行に、収束していれば 1 になるということが書かれています。今回は全部 1 になっています。閾値としては 1.1 や 1.05, 1.01 というように、いろいろ言われていますが、今回の計算上は 1.05 未満であれば収束と考えていきます。

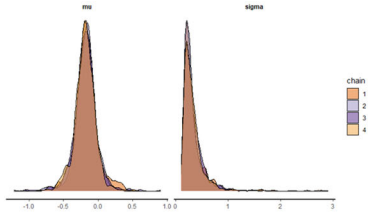
結果をグラフで見ることもできます。

rstan (続)

- 実行結果 (続)

```
> stan_trace(stan.out, inc_warmup=T)
> stan_dens(stan.out, separate_chains=T)
> stan_ac(stan.out)
```





- マルコフ連鎖でありサンプル間に自己相関がある
- 各系列2000個のサンプルのうちwarm-upを除く1000個×4を並び替えて利用する

22

左上は、マルコフ連鎖のステップごとのサンプルの値を示しています。

右上は事後分布の経験密度関数で、左下はサンプルの自己相関です。先ほどお話ししたように、やや自己相関があります。Stan では、自己相関の影響を和らげるために、四つの Chain からの結果をランダムに並べ替えてつなげたものを使って計算をすることにしています。

rstan (続)

- 実行結果 (続)
  - 4系列の各1000個のサンプルをランダムに並び替えた結果

```
> extract(stan.out)$mu

   [1] -0.221512815 -0.254758324  0.179910435 -0.173946512 -0.258703907 -0.196681321 -0.124393153
   [8] -0.348538661 -0.179554801 -0.346560938 -0.161057392 -0.113770549 -0.028899127  0.007769661
  [15]  0.341774279 -0.047770273 -0.162741454  0.032653586 -0.207700255 -0.238295585 -0.160724906
...
[3991] -0.172279990 -0.306120655 -0.156889246 -0.125090609 -0.114032893 -0.720310491 -0.249227106
[3998] -0.283448939 -0.064863311 -0.439628097
```

  - 並び替える前の結果

```
> extract(stan.out, permuted=FALSE)[, , "mu"]
          chains
iterations    chain:1     chain:2    chain:3    chain:4
      [1,] -0.2278730 -0.02215098 -0.2632654 -0.2190651
      [2,] -0.2379709 -0.31418250 -0.3132228 -0.1505139
      [3,] -0.1145457 -0.06052351 -0.2213526 -0.1958901
      [4,] -0.2874225 -0.33657972 -0.1091642 -0.2009928
      ...
   [1000,] -0.1912756 -0.29527330 -0.2080481 -0.0816280
```

23

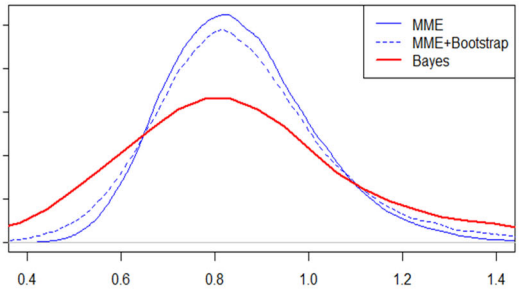
mu について、具体的な数値をスライドに示しています。上に表示しているのは、実際に使用する値で、四つの Chain からのそれぞれ 1000 個のサンプルをランダムに並べ替えたものです。下の値は、並び替える前のサンプルをそのまま表示したものです。これを並び替えたものが上の値になるということです。

予測分布の計算

- 翌年度の損害率Lの分布の計算
  - モデルの末尾にgenerated_quantitiesブロックを追加

```
generated quantities {          (modelブロックまでは同一であり省略)
  real r_new;
  r_new = exp(normal_rng(mu, sigma));
}
```

L (r_new) の分布



Lの信頼区間

	2.5%点	97.5%点
モーメント法	56.2%	125.4%
〃 +Bootstrap	53.5%	132.3%
ベイズ	31.5%	188.6%

24

次に、パラメータの mu, sigma の事後分布に基づく予測分布を計算します。スライド p. 20 のコードの最後に generated quantities ブロックとしてここに示したコードを追加して実行すれば計算できます。これによって、新しく定義した r_new という変数の中に予測損害率のサンプルが入ってきます。

得られた予測損害率の分布を、最初に示したベイズを使わないやり方と比べてみると、下の図のようになっています。赤い線が Stan の実行結果で、青い実線の方が、モーメント法で計算したパラメータによる結果です。青い点線は、これに Bootstrap を使ってパラメータリスクを反映した結果です。これを見ると、どうもベイズを使うと幅が広くなるようです。

事前分布

- p.20のモデルでは、事前分布を明示していない
  - 明示しない場合は一様分布となる（非正則事前分布）

$$\mu \sim U(-\infty, +\infty), \quad \sigma \sim U(0, +\infty)$$

- 事前分布を明示的に指定（弱情報事前分布）
  - 例：$\mu \sim N(0, 25^2)$, $\sigma \sim hCauchy(0, 25)$

```
model {
  mu ~ normal(0, 25);
  sigma ~ cauchy(0, 25);
  for(i in 1:n)
    log(r[i]) ~ normal(mu, sigma);
}
```

sigmaは、下限を0としているため0未満の部分が切り捨てられてHalf-Cauchy分布となる

  - 別の例：$\sigma^2 \sim InvGamma(0.001, 0.001)$
    ただしStan Development Teamは推奨していない*1
    （確率の大半が妥当と思われる範囲の外にあるため、事後分布を歪める可能性がある）

*1 Stan モデリング言語: ユーザーガイド・リファレンスマニュアル https://stan-ja.github.io/gh-pages-html/

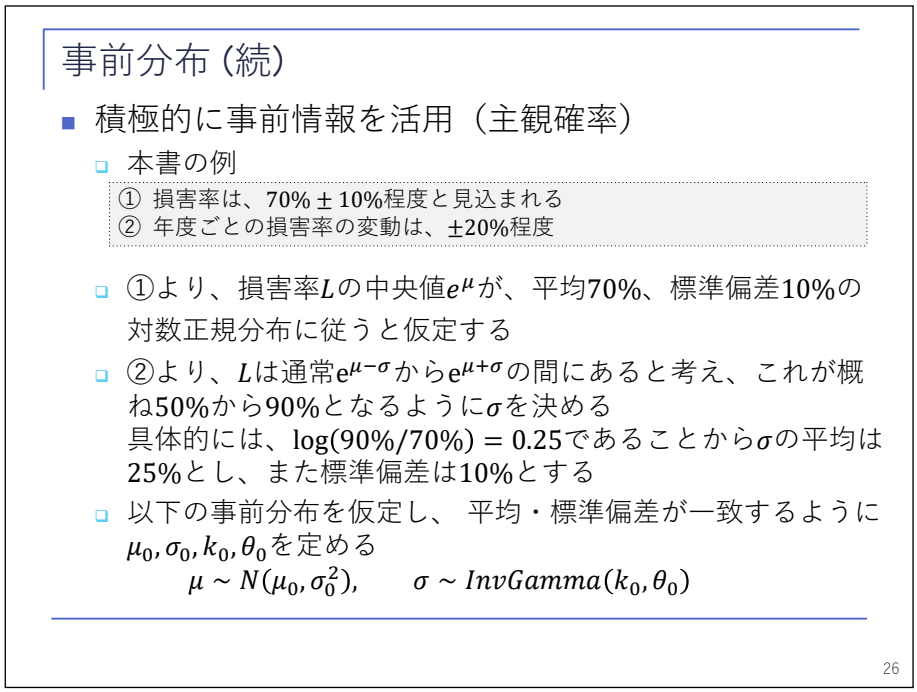
25

理由としては、スライド p. 20 のモデルは、事前分布を特に明示していなかったことが考えられます。事前

分布を明示しない場合、Stan では一様分布として扱われます。事前分布を明示的に与えることももちろん可能で、例えば、このスライドのコードであれば mu については正規分布で、sigma については Half-Cauchy 分布を使っています。

例えば、正規分布は標準偏差を 25 としており、損害率が通常取り得る範囲と比べると相当ばらつきが大きい分布になっています。事実上一様分布に近いものだと考えられるということで、このようにしています。

sigma の分布を Half-Cauchy 分布とする代わりに、sigma の 2 乗の分布を逆ガンマ分布とし、そのパラメータを非常に 0 に近い値にすることもあります。よく使われているようですが、実は、Stan の開発チームはこれを推奨していません。



ここまでは、あまり事前の情報を活用しない、「何も知らないよ」という体の分析だったのですが、「積極的に事前の情報を活用していこう」というような立場もあります。これも本書で説明されており、例えば、この例であれば、「損害率の平均について大体 70%±10%ぐらいの範囲にあるだろう」とか、あるいは「年度ごとの損害率の変動は 20%前後だろう」というような事前の知識があるという前提の下に、平均は標準正規分布、標準偏差は逆ガンマ分布に従うという前提を置いて、事前分布のパラメータをスライドにあるように推定するというようなことが紹介されています。

事前分布 (続)

- 無情報事前分布、弱情報事前分布
  - 一様分布
    - Stanにおけるデフォルトの選択
    - 採りうる範囲を決めないと$\int p(\phi)d\phi = 1$にならない（非正則）
    - パラメータ変換のもとで不変ではない
  - 分散の大きい分布
    - $\mu \sim N(0, 1000^2)$、$\sigma \sim InvGamma(0.001, 0.001)$など
    - Stan Development Teamの推奨
      回帰係数…正規分布, コーシー分布、標準偏差…半コーシー分布
  - Jeffreysの事前分布
    - $p(\phi) \propto \sqrt{\det I(\phi)}$　$\mathrm{I}(\phi) = -\mathbb{E}\left(\frac{\partial^2 \log f(x|\phi)}{\partial \phi^2}\right)$:フィッシャー情報行列
    - 単位体積あたりの確率がパラメータ変換の影響を受けない
- より強い情報を持つ事前分布
  - 事前に得られた別のデータ、メタアナリシス、専門家の情報

27

事前分布の設定について、このスライドにまとめています。まず、なるべく分析対象のデータ以外の情報を使わない立場として、無情報事前分布や弱情報事前分布といったようなものがあります。一様分布がその例で、Stan ではデフォルトの設定になっています。ただ、パラメータの取り得る範囲が有限でない場合には積分しても 1 になりませんので、厳密には確率分布ではなくなってしまいます。計算上、特にこれでも問題ないのですが、それを嫌って正規分布などの分布で、先ほど示したように幅が広い分布を使うというやり方もあります。Stan のマニュアルには、開発チームが推奨する事前分布の設定なども書かれています。

もう一つ、一様分布の問題として、パラメータ変換の下で不変ではないという性質があります。例えば、正規分布について、一般には平均と標準偏差をパラメータにするかと思いますが、標準偏差の代わりに分散の逆数を「精度」と呼んで、これをパラメータにするという方法もあります。

実際、先ほどご紹介した MCMC エンジンの中でも、Stan 以外はそのようなパラメータ表現を使っています。すると、精度について事前の知識がないということで一様分布を仮定しても、実は標準偏差の観点から見ると一様ではないということになってしまいます。これを避けるために、パラメータ変換の影響を受けないような事前分布として、Jeffreys の事前分布などを使うということもあります。

一方で、前のスライドでお示しした、より強い情報を使うというような立場もあります。この場合には、例えば、分析対象とは違うデータを使ったり、同じ題材についての先行研究を見たり、専門的な他の情報を活用したり、というようなことがされます。

事前分布 (続)

- 様々な事前分布による計算結果

手法	事前分布	2.5%点	97.5%点
モーメント法		56.2%	125.4%
〃 +Bootstrap		53.5%	132.3%
ベイズ	① $\mu \sim U(-\infty, +\infty), \sigma \sim U(0, +\infty)$	31.5%	188.6%
	② $\mu \sim N(0, 25^2), \sigma \sim InvGamma(0.001, 0.001)$	44.8%	153.4%
	③ $\mu \sim N(0, 25^2), \sigma \sim InvGamma(0.1, 0.1)$	44.8%	152.5%
	④ $\mu \sim N(0, 25^2), \sigma \sim hCauchy(0, 25)$	36.9%	190.5%
	⑤ $\mu \sim N(0, 25^2), \sigma \sim hCauchy(0, 2500)$	39.3%	186.1%
	⑥ $\mu \sim N(0, 25^2), \sigma \sim Beta(1, 1)$	40.7%	169.3%
	⑦ $\mu \sim U(-\infty, +\infty), \sigma \sim InvGamma(2, 0.001)$	53.9%	128.6%
	⑧ $\mu \sim N(\mu_0, \sigma_0^2), \sigma \sim InvGamma(k_0, \theta_0)$	47.3%	126.1%

- 結局どう選べばよいのか

28

実際に計算した結果はこのスライドの通りです。①から⑧はそれぞれ違う事前分布を使った結果ですが、相当ばらついています。

結局、どう選べばよいのかが問題です。この問いに対して、この本の著者の立場としては、「人間は、必ずしもデータを見る前に事前分布の形で信念を持っているわけではない。あくまでも、ベイズというのは、そのような事前分布があるという仮定の下に理想化されたモデルであり、実際には、事前分布を選択するにあたっては、分析結果を見て理解する必要がある」と、そのような立場を取っています。

一旦本書の内容からは離れますが、具体的な方法についてお話しします。

モデルの評価

- ホールドアウト検証、交差検証、情報量規準…
- DIC (Deviance Information Criterion)
  - WinBUGS, JAGS等の標準の出力に含まれる
- WAIC (Widely Applicable Information Criterion)*1
  - $W_n = T_n + V_n/n$

    事後分布についての期待値

    $T_n = -\frac{1}{n}\sum_{i=1}^{n} \log \mathbb{E}_{\phi}[p(X_i|\phi)]$,

    $V_n = \sum_{i=1}^{n}\{\mathbb{E}_{\phi}[(\log p(X_i|\phi))^2] - \mathbb{E}_{\phi}[\log p(X_i|\phi)]^2\}$
  - X_iが独立かつ同一分布に従うとき

    $\mathbb{E}[G_n] = \mathbb{E}[W_n] + o(1/n^2)$

    $G_n = -\int q(x) \log p(x) dx$ ：汎化損失

    $= \int q(x) \log \frac{q(x)}{p(x)} dx - \int q(x) \log q(x) dx$

    予測分布に依存しない量

    真の分布$q(x)$と予測分布$p(x)$のK-L距離

*1 Watanabe, S., & Opper, M. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. Journal of machine learning research, 11(12).

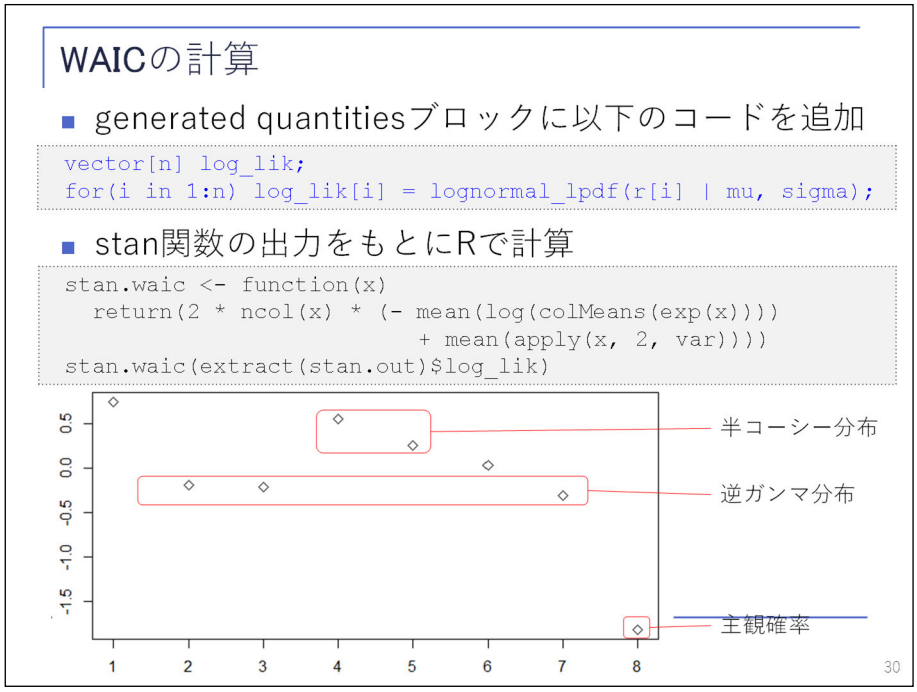
29

今回取り上げた例では、翌年度の損害率を予測するということを目的にしていましたが、このような予測

の観点からモデルの選択を行う方法としては、例えば、スライドに書いてあるような、ホールドアウト検証、交差検証、情報量規準に基づく選択、というようなものが使われます。今回の例では、データが 5 個しかないので、情報量規準による選択を採用します。

情報量規準にも幾つかあるのですが、よく使われるものとして DIC があります。DIC は、先ほど紹介した MCMC エンジンの中でも Stan 以外のものでは標準の出力に含まれており、そのようなところもあってよく使われているのではないかと思います。

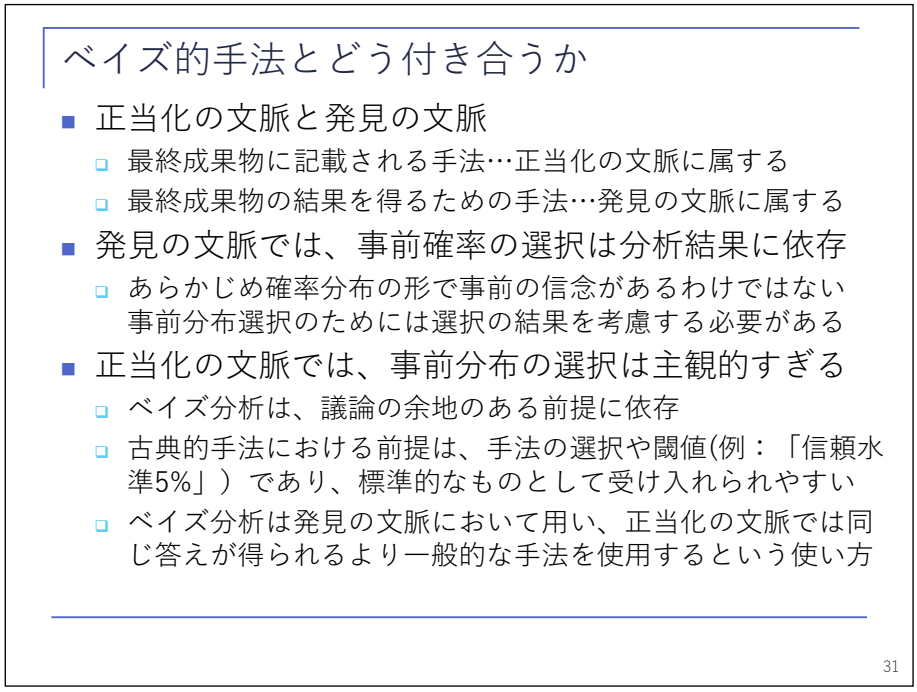
今回使うのは、DIC ではなく WAIC で、東工大の渡辺澄夫先生が提案されている情報量基準です。詳細は割愛しますが、これを使うことによって、WAIC が小さくなるようなモデルを選んでいけば、平均的には真の分布と予測分布の距離が小さいようなモデルを選択することができます。



Stan では、generated quantities ブロックの中にスライドの青字のコードを追加することで計算できます。これで出力として log_lik という変数が得られますので、これを使って R 上でスライドの黒字のコードのように計算していけば WAIC が計算できます。

結果はスライドの図の通りで、縦軸が WAIC、横軸の 1 から 8 はスライド p.28 の①から⑧のモデルに対応します。これを見ると、⑧の主観確率を使ったモデルが一番低くなっています。それ以外では、②③⑦の三つが低いのですが、これらは全部、事前分布に逆ガンマ分布を使った例です。中でも、差は微妙ですが、⑦が一番低くなっています。⑦は、逆ガンマ分布の最初のパラメータを 2 としたものです。

⑦、⑧はどちらもベイズ的手法を使わなかった場合と大体近い結果になっており、この辺りが大体妥当な結果なのだろうと思われます。



ここまで、計算の仕方について説明してきましたが、最後に、ベイズ的手法への向き合い方について、本書の著者の考え方をお話ししたいと思います。

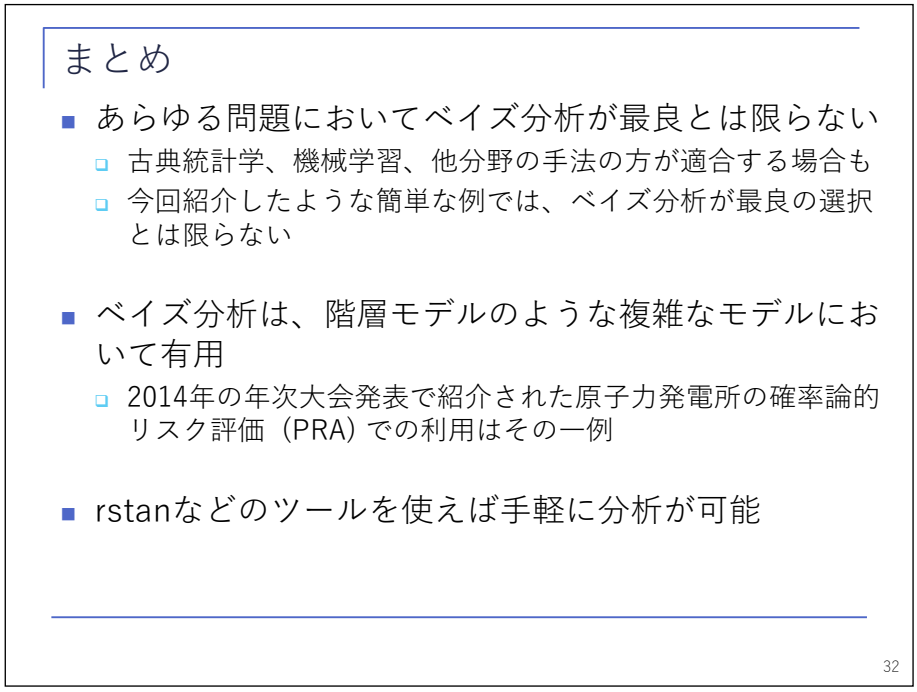
説明にあたっては、「発見の文脈」「正当化の文脈」という言葉が使われています。最終成果物、例えばアクチュアリーであれば商品認可の申請書類などがこれにあたりますが、このような最終成果物に記載される説明に使用される手法は「正当化の文脈」に属するもの、これに対し、結果としてその手法を得るに至る過程についての説明は「発見の文脈」に属するものということです。

事前分布の設定に関しては、ひとは事前の信念を確率分布の形で持っているわけではないので、設定にあたり何らかの判断が必要になります。判断にあたってはその判断の帰結を考慮する必要があります。つまり、事前分布の設定の際には結果として得られる予測の妥当性を考慮する必要があるということです。ただし、これは発見の文脈での説明であり、正当化の文脈では、事前分布は分析プロセスへのインプットとして扱われます。

正当化の文脈では、事前分布の選択は主観的であるという批判を受けやすい懸念があります。明示的に示された事前分布の仮定は、いかにも突っ込みやすいポイントです。実際には、非ベイズ的手法であっても、手法や閾値の選択で主観の入る要素はありますが、一般に用いられているということで比較的受け入れられやすいかもしれません。

著者としては、ベイズ的手法は発見の文脈に限定し、正当化の文脈では同様の結果が得られるより一般的な手法を用いるという対応を、称賛に値するものとして挙げています。

一方で、前のスライドでお話ししたように、情報量規準等を用いて予測の観点から事前分布を選択することで、正当化の文脈においても評価に耐えうる説明ができるかもしれません。



前半を終えるにあたり強調しておきたい点をまとめます。

まず、全ての問題についてベイズが最良とは限らないということです。特に今回お示ししたような簡単な例では、ベイズでない方法を使ったほうがいいかもしれません。

ただ、複雑な問題、例えば 2014 年の年次大会で紹介した原子力発電所の確率論的リスク評価のような複雑な問題については、ベイズが威力を発揮します。今日お話ししたようなやり方で rstan のようなツールを使えば、手軽に分析もできるようになっていますので、ぜひ、皆さん、実際に簡単なモデルでやってみてください。うまくいったら今度は複雑なモデルについてもどんどん試してみて、使えるものがあれば、論文発表などしていただければと思います。これが、本書が提案する計算保険数理のアプローチではないかと思っています。

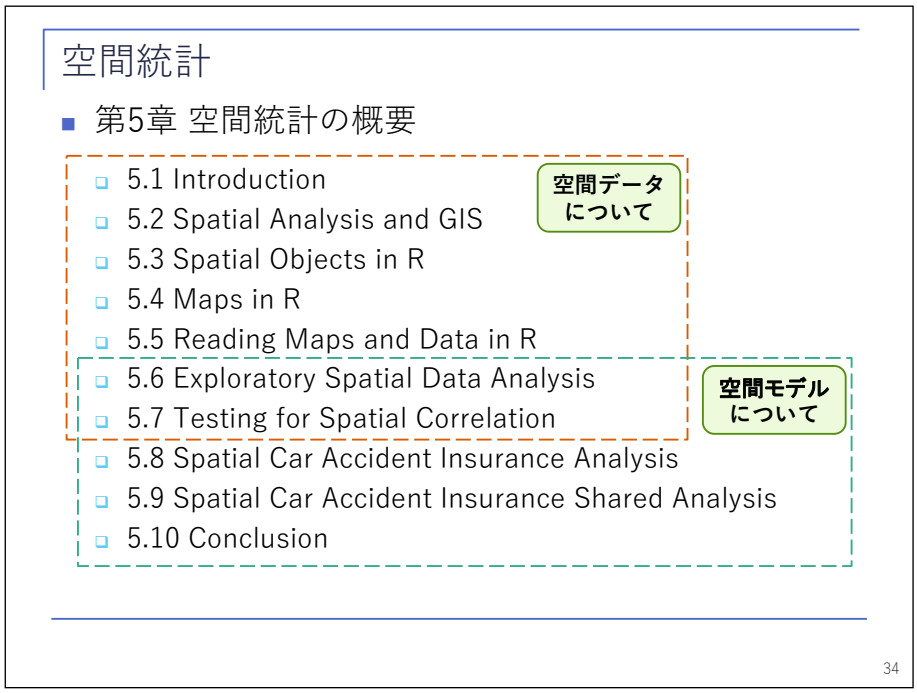
以上で前半の発表を終わります。続いて佐野さんから、空間統計についてお話しします。

【司会】　渡辺さん、ありがとうございます。引き続き、佐野さんから発表していただきます。

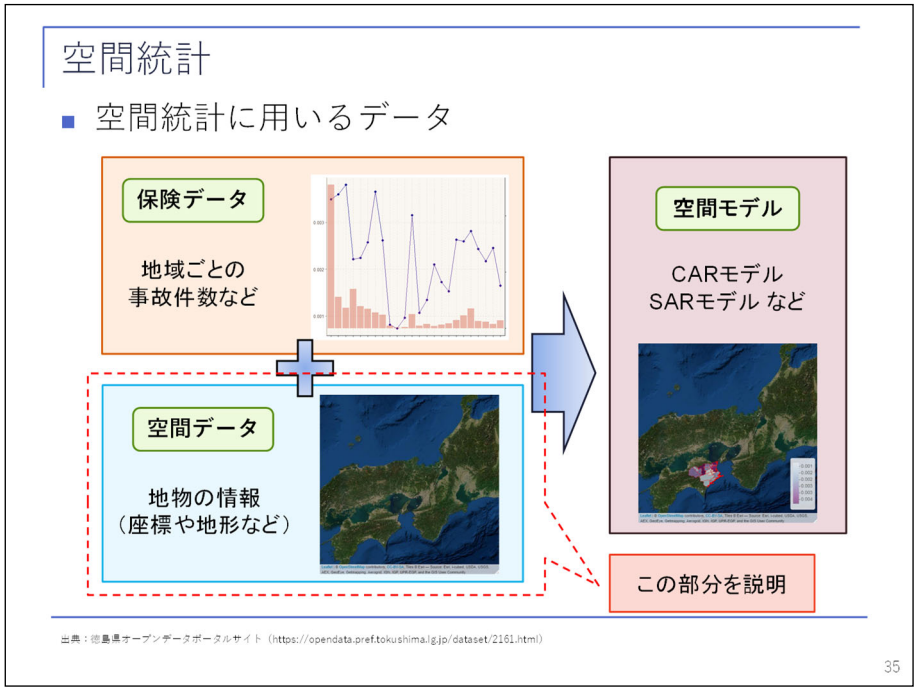
では、佐野さん、よろしくお願いします。

【佐野】　ASTIN 関連研究会の佐野と申します。

私からは本の第 5 章、空間統計についての話をします。



第 5 章の構成はここにある通りです。初めにひとしきり空間データの話をした後、最後に保険データを用いた空間モデルの実例という流れになっています。この章でほとんど話していることは、空間データについてです。



空間モデルを使おうとか勉強しようというとき、手元に保険データはもちろんあるとしてよく見ますし、空間モデルも本に書いてあるからよく見ますが、そのときに必要な空間データに関する知識についてはあまり書かれていません。空間データのことばかり書いている本もありますが、そちらだと情報が多すぎて、自分にとってどれが必要なのかよく分かりません。

その点、この章は保険データを使った空間モデルというゴールに対して、必要な空間データの話ばかりしています。空間モデルを使うときに必要なことをこういう書き方をしているものはあまりないので、有益だと思って今回ピックアップしています。

そしてこの章の考え方に基づいて、空間モデルの話は一切せずに、空間データの話だけをします。

大きい話からしていくと、まず空間モデルに使われる空間データは、主に 4 種類あります。一つ目が Point Pattern、事故の発生場所などのイベントがどこで発生したかを示すデータです。二つ目が Random Surface、事故の発生場所それぞれが量を持つ雨量計の雨量や気温などのデータです。次が空間相互作用、点と点をつないだ路線図や航路などの線データです。最後が地域データ、一定の形に区分された各地域が人口などの量を持つデータです。今回は地域データをゴールとします。

持っているデータを分析に使うためには、そのためのツールが必要です。地理情報システム、GIS がそのためのシステムです。データを使ってモデル化するアプローチは、主に二つあります。一つが Geo-objects、点や線、ポリゴンによるベクトルデータです。もう一つが fields、均等に区切られたグリッドやピクセルが量を持つデータです。今回は Geo-objects を扱います。

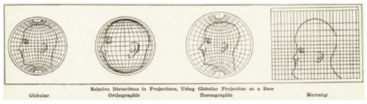
そして R を GIS として活用するために、今回は sp パッケージの話をします。

空間統計

- spパッケージ
  - 空間データとその属性を扱うクラスを持つ
  - Spatialクラス

```
Spatial(proj4string = CRS(as.character(NA)), bbox = NULL)
```

  - 2つのスロットを持つ
    - 境界線（bbox）
    - 投影法（proj4string）



```
bb <- matrix(c(-10, -10, 10, 10), ncol = 2, dimnames = list(NULL, c("min", "max")))
crs <- CRS(projargs = "+proj=longlat")
sobj <- Spatial(bbox = bb, proj4string = crs)
> sobj
class       : Spatial
features    : 1
extent      : -10, 10, -10, 10  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
```

出典：Deetz, C.H., & Adams, O.S. Elements of Map Projection: with Applications to Map and Chart Construction.

38

sp パッケージとは、空間データを扱うためのパッケージです。空間データを扱うためのデータ形式をクラスといいますが、sp パッケージで扱うクラスを知ることで、R での空間データの扱い方を学べます。

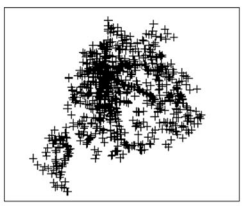
最も基本的なものが Spatial クラス。ここでは二つ決めます。一つが境界線、プロットするときの範囲です。もう一つが投影法、3 次元のものをどのような形で地図として投影するかというものです。

Spatial クラスは最も基本的なクラスですが、実際にデータを入れて使うのは、この下の階層のサブクラスです。

空間統計

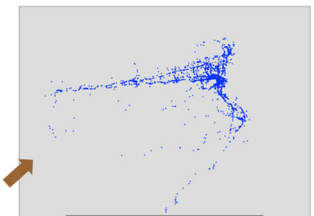
- spパッケージ
  - SpatialPointサブクラス
    - Spatialクラスの拡張版
    - 地理的な点の空間座標をcoordsスロットに格納する

```
SpatialPoints(coords, proj4string = CRS(as.character(NA)), bbox = NULL)
```



ペロオリゾンテで2011年2月に発生した交通事故の発生場所

latitude	longitude
34.053501	134.572708
34.020863	134.551952
34.067621	134.547293
34.030796	134.552054
・・・	



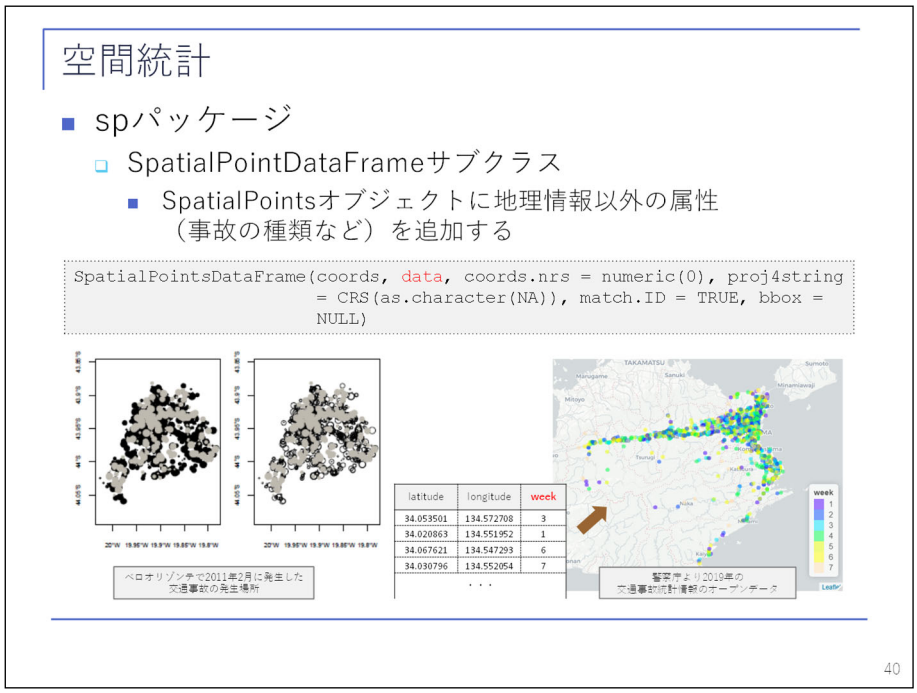
警察庁より2019年の交通事故統計情報のオープンデータ

出典：警察庁ウェブサイト（https://www.npa.go.jp/publications/statistics/koutsuu/opendata/index_opendata.html）

39

一つ目が SpatialPoint サブクラス、点データを扱うサブクラスです。coords スロットに緯度・経度のデータ入れることで、点データを扱うことができます。

左下にあるのが、ベロオリゾンテでの交通事故の発生場所を示したプロットです。この章に出てくる実例は、初めから最後までブラジルです。ブラジルの例を見てどうだと言われてもちょっとよく分からないので、分かりやすさの観点で日本での例を付け加えています。右下にあるのが、警察庁のオープンデータにある交通事故の発生場所のプロットです。



もし点データを扱う場合であっても、保険データであれば、変数としてそれ以外の情報が必要となります。そのために使うのが SpatialPointDateFrame サブクラスです。これはデータの中に、例えば事故の発生場所であればその事故の種類などの属性を入れたサブクラスです。

左下は本にあるプロットです。右下のプロットは、警察庁のオープンデータに事故の発生が何曜日かという情報が付いていましたので、曜日ごとに 7 色に色分けしています。

データは coords スロットに緯度・経度を入れて、data に属性を入れます。データがひとまとめになっている場合は、coords.nrs で緯度・経度を表す列を指定します。データは別々だがお互いが共通のインデックスを持っている場合は、match.ID を使うことでマッチマージできます。

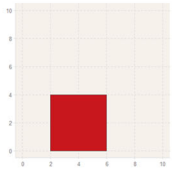
ここまでが点データの話です。普通は点の次といえば線ですが、線データを保険で扱うことはあまりないので、次はポリゴンに行きます。

空間統計

- spパッケージ
  - SpatialPolygonsサブクラス
    - Polygonサブクラス

```
Polygon(coords, hole=as.logical(NA))
```

```
p1 <- rbind(c(2,0), c(6,0), c(6,4), c(2,4), c(2,0))
pl1 <- Polygon(p1)
> str(pl1)
Formal class 'Polygon' [package "sp"] with 5 slots
  ..@ labpt  : num [1:2] 4 2
  ..@ area   : num 16
  ..@ hole   : logi TRUE
  ..@ ringDir: int -1
  ..@ coords : num [1:5, 1:2] 2 6 6 2 2 0 0 4 4 0
```



- labpt：ポリゴンの重心座標（頂点の座標の平均）
- area：ポリゴンの面積
- hole：ポリゴンが穴であるかどうかを示す値
- ringDir：ポリゴンの描画方向
- coords：頂点の座標

41

ポリゴンとは、始点と終点が同じである点の集合を線でつないでできる図形です。一番の基本が Polygon サブクラスです。p1 は始点と終点が同じベクトルで、これをつないで Polygon サブクラスのオブジェクトとしています。右にある図が出来上がりです。

中を見るといろいろと決まりごとがあって、重心や面積などがあり、その中に hole というものがあります。これはポリゴンを作るときに、TRUE か FALSE で指定します。使い方は後で説明します。

空間統計

- spパッケージ
  - SpatialPolygonsサブクラス
    - Polygonsサブクラス
      - 複数のpolygonオブジェクトを結合して、一つの地域を表すオブジェクトを作る

```
Polygons(srl, ID)
```

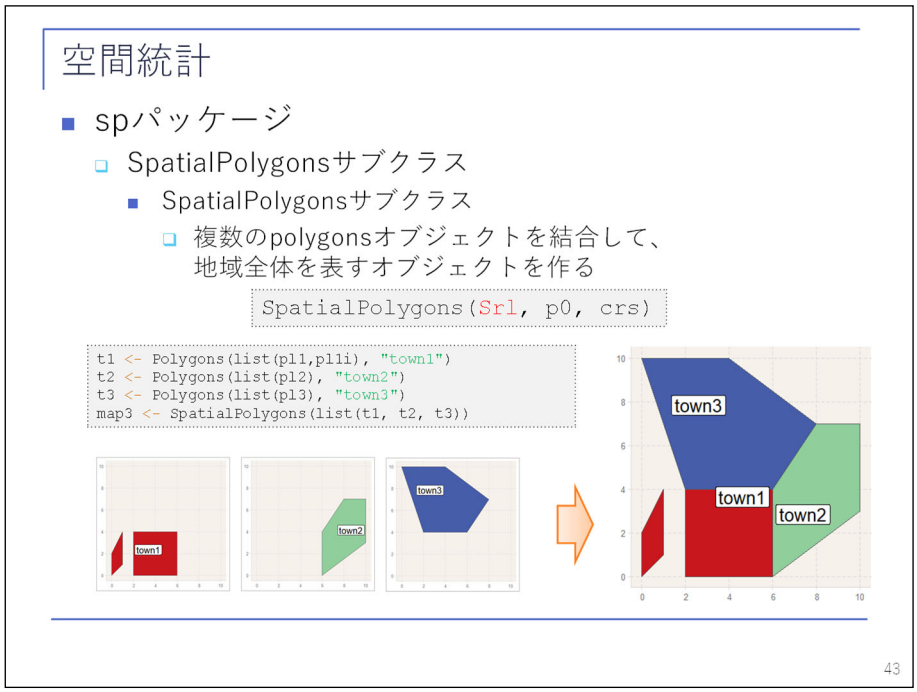
```
p1i <- rbind(c(0,0), c(1,1), c(1,4), c(0,2), c(0,0))
p2 <- rbind(p1[2,], c(10,3), c(10,7), c(8,7), p1[3:2,])
p3 <- rbind(p1[4:3,], p2[4,], c(4,10), c(0,10), p1[4,])
# polygon object
pl1 <- Polygon(p1); pl1i <- Polygon(p1i);
pl2 <- Polygon(p2)
pl3 <- Polygon(p3)

t1 <- Polygons(list(pl1,pl1i), "town1")
t2 <- Polygons(list(pl2), "town2")
t3 <- Polygons(list(pl3), "town3")
```

42

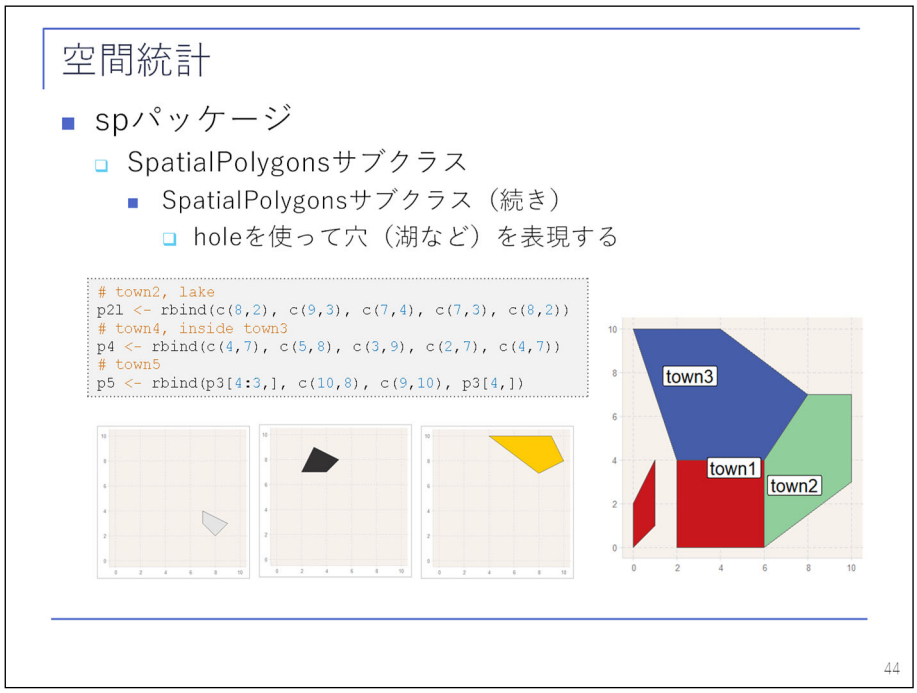
実際に地図を作るとき、淡路島が兵庫県だというように、複数の図形でひとまとめの地域として扱うことがあります。そのために Polygons サブクラス、複数形になっているものを使います。

複数の Polygon サブクラスのオブジェクトをリスト化して、まとめて一つの地域にします。例えば、先ほ

どの図形 p1 にプラスして p1i を作り、town1 という Polygons サブクラスのオブジェクトとしてまとめています。他に town2、town3 もついでに作ります。



SpatialPolygons サブクラスとは、複数の地域をひとまとめにして、その地域全体を表すサブクラスです。town1 から town3 をリスト化して、三つ合わせて合体して、一つの地域全体を表すポリゴンのオブジェクトを作っています。

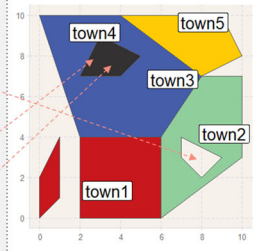


これで基本的なポリゴンの作り方は分かったので、次は、先ほどの hole を使ってポリゴンに穴を掘ります。まず p21 という town2 の中に入るポリゴン、p4 として town3 の中に入るポリゴン、ついでに p5 を作ります。

空間統計

- spパッケージ
  - SpatialPolygonsサブクラス
    - SpatialPolygonsサブクラス（続き）
      - holeを使って穴（湖など）を表現する

```
pls5 <- list()
pls5[[1]] <- Polygons(list(Polygon(p1, hole = FALSE),
                           Polygon(p1i, hole = FALSE)),
                      "town1")
pls5[[2]] <- Polygons(list(Polygon(p2, hole = FALSE),
                           Polygon(p2i, hole = TRUE)),
                      "town2")
pls5[[3]] <- Polygons(list(Polygon(p3, hole = FALSE),
                           Polygon(p4, hole = TRUE)),
                      "town3")
pls5[[4]] <- Polygons(list(Polygon(p4, hole = FALSE)),
                      "town4")
pls5[[5]] <- Polygons(list(Polygon(p5, hole = FALSE)),
                      "town5")
map5 <- SpatialPolygons(pls5)
```



45

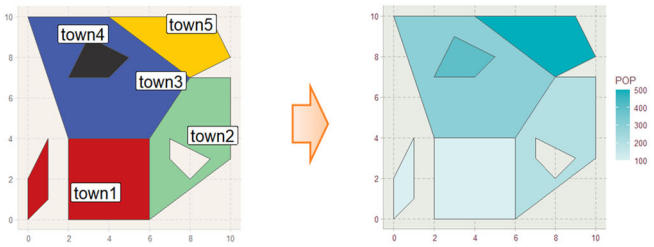
これらを town1 から town3 で作った地域全体のポリゴンに更に合わせるのですが、ポリゴンとして各地域を指定するとき、p2l は town2 として指定するときに hole を TRUE とします。そうすると town2 に穴を開けることができるので、自分で作ることはあまりないですが、琵琶湖を作ることができます。さらに town3 を作るとき p4 の hole を TRUE にして穴を開けて、town4 として hole を FALSE にすることで、今度はバチカンを作ることもできます。

空間統計

- spパッケージ
  - SpatialPolygonsDataFrameサブクラス
    - SpatialPolygonsオブジェクトに属性を追加する

```
SpatialPolygonsDataFrame(Sr, data, match.ID = TRUE)
```

```
POP <- data.frame(POP = seq(100, 500, by = 100)) # the number from 100 to 500
map5d <- SpatialPolygonsDataFrame(map5, POP, match.ID = FALSE)
```



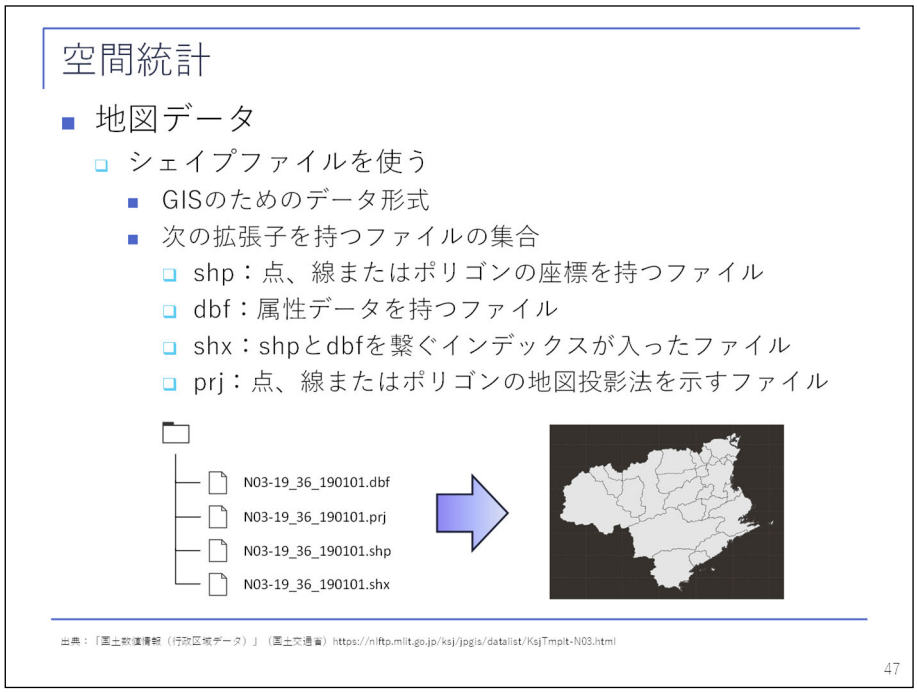
46

これで SpatialPolygons のオブジェクトの作り方が分かりました。点データでいえば SpatialPoints の次が SpatialPointsDataFlame サブクラスであるのと同じように、ポリゴンの場合も SpatialPolygonsDataFlame サブクラスがあります。これも同じように属性を入れたものです。

空間モデルの本を読んで言われるがままにデータを入れると、おそらく一番初めに見るのがSpatialPolygonsDataFlame サブクラスです。とても長い名前のリストが入っているので「何だこれは」となりますが、今回の話を聞いていただければ「SpatialPolygons の DataFlame のサブクラスだ」と理解しやすいのではないかと思います。

今回は town1 から town5 に 100 から 500 の値を入れて、それで色分けしてプロットしています。

ただし、SpatialPolygonsDataFlame サブクラスまで理解してポリゴンの作り方が分かったからといって、すぐに空間モデルとはなりません。自分でポリゴンを作ることはなく、実際はどこかから地図データを取ってくる必要があるからです。



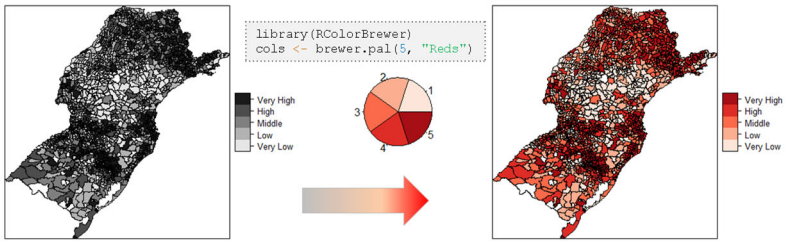
そのときに使うのがシェイプファイルです。シェイプファイルとは GIS のためのデータ形式で、中身はここにあるような拡張子を持ったデータファイルの集合です。

例えば国土地理院に行政区域データがあって、都道府県別や市町村別ポリゴンのシェイプファイルがあるので、それを R に取り込んでプロットすれば、市町村別のプロットを作ることができます。

空間統計

- 地図データ
  - SpatialPolygonsDataFrameオブジェクトをプロットする
  - spplot関数：属性の値ごとに地域（ポリゴン）を色分ける

```
library(CASdatasets)
data(brgeomunicins)
cols <- rev(gray(seq(0.1, 0.9, length = 5)))
spplot(brgeomunicins, "HDIcity00", col.regions = cols, cuts = length(cols)-1)
```



48

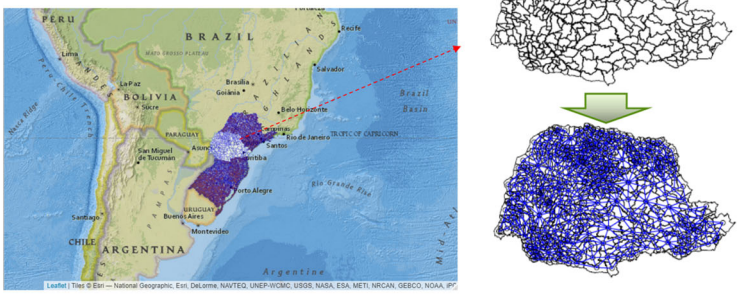
地図データを取ってきたら、次は保険データを分析します。分析するときに地域別で持っている事故の発生件数や人口などを色分けしてプロットしたいときには、spplot 関数を使います。

またブラジルの例ですが、先ほど説明があった CASdatasets ライブラリの中に brgeomunicins というデータがあって、これを読み込むと本の再現ができるのでプロットしています。これは人口データです。色分けしてプロットしたものが左にあるもので、brewer パッケージを使えば、更にこれをカラフルに色分けできます。

空間統計

- 隣接関係
  - 地域（ポリゴン）同士の近傍関係を設定する
  - spdepパッケージのpoly2nb関数を使う

```
pos <- which(brgeomunicins@data$State == "Parana")
prshape <- brgeomunicins[pos,]
pr.nb <- poly2nb(prshape)
```

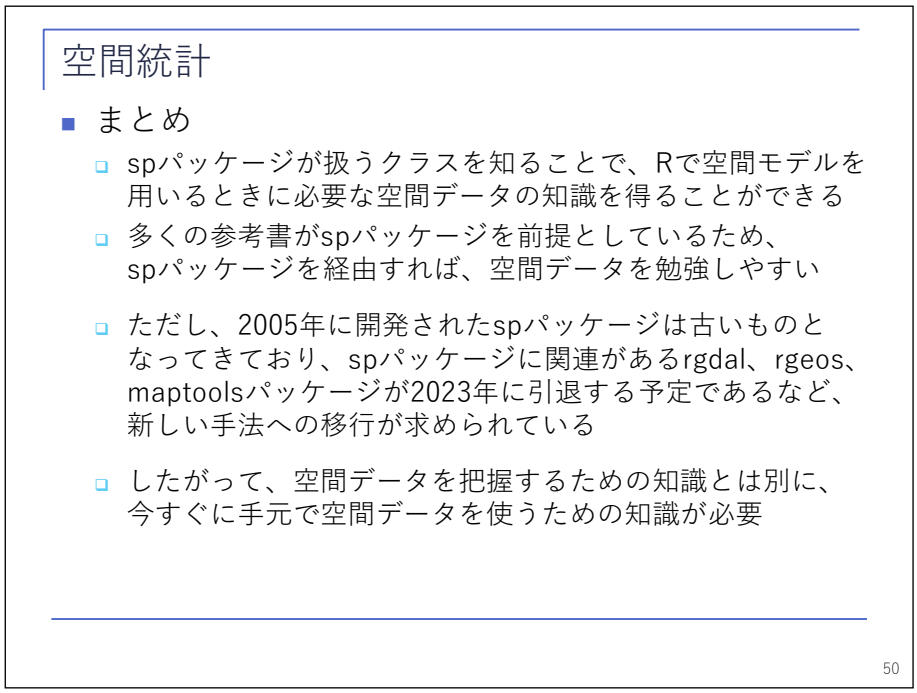


49

分析が済んで次に空間モデルにいくためには、空間データを使って地域別の近傍関係、どの地域とどの地域が近いのかを設定する必要があります。そのために spdep パッケージの poly2nb 関数を使います。

ここでは、先ほどのブラジル南部の地域からパラナ州のデータを取ってきて、隣接関係のリストを作っています。プロットすると分かりやすいのかどうか分からないですが、図のような隣接関係を作ることができます。

ここまで作れば、あとは自分が持っている保険データを合わせて、空間モデルに向かっていくだけです。

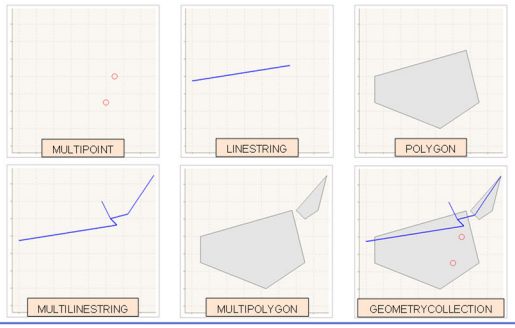


まとめとして、今回は sp パッケージが扱うクラスを学ぶことで、R で空間データをどのように運用するかということを学びました。空間モデルの本は新しいものがどんどん出ているわけではなく、多くの参考書が sp パッケージを前提として話を進めているので、空間モデルを勉強するとき、sp パッケージの知識はまだ現時点では必須と思われます。

思われるのですが、sp パッケージに関連する周辺のパッケージは段々なくなっており「sp パッケージではなくて新しいところに移行しなさい」とアナウンスされています。ですから sp パッケージを知る必要はありますが、今すぐ手元で動かすというときには、また別の知識が必要です。

空間統計

- sfパッケージ
  - Simple Feature（空間情報を定義・格納・表現するための標準的な規格）を扱うパッケージ
  - spパッケージの運用に関するギャップを埋める
  - 主な地物の種類



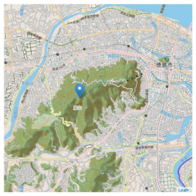
51

そのために、sf パッケージの話を少しします。

sf パッケージとは、Simple Feature という空間情報の定義、格納や表現、そのための標準的な規格を扱うパッケージです。sp パッケージを使うときの「もっとこうできれば」という部分が改善されています。使われるジオメトリも、sp パッケージで使う点、線やポリゴンなどが網羅されています。

空間統計

- sfパッケージ
  - 3つのクラスでSimple Featureを表す
    - sfg（Simple Feature Geometry）クラス
      - 地物（点、線、ポリゴンなど）の集合



```
b_sfg <- st_point(c(134.5165, 34.0607))
> b_sfg
POINT (134.5165 34.0607)
```

    - sfc（Simple Feature geometry list-Column）クラス
      - sfgオブジェクトに地理的情報を付保したもの

```
b_sfc <- st_sfc(b_sfg, crs = 4326)
> b_sfc
Geometry set for 1 feature
geometry type:  POINT
dimension:      XY
bbox:           xmin: 134.5165 ymin: 34.0607 xmax: 134.5165 ymax: 34.0607
geographic CRS: WGS 84
POINT (134.5165 34.0607)
```

52

こちらもクラスがありますので、三つのクラスを簡単に説明します。まずは sfg クラス、点や線、ポリゴンなどの地理的な情報が入ったデータの集合です。それから sfc クラス、地理情報が入ったデータに、更に周辺情報、投影法や有効桁数などの周辺情報を付け足したものです。

空間統計

- sfパッケージ
  - 3つのクラスでSimple Featureを表す
    - sf（Simple Feature）クラス
      - sfcオブジェクトと属性を結合したもの

```
b_sf <- st_sf(name = "Bizan", geometry = b_sfc)
> b_sf
Simple feature collection with 1 feature and 1 field
geometry type:  POINT
dimension:      XY
bbox:           xmin: 134.5165 ymin: 34.0607 xmax: 134.5165 ymax: 34.0607
geographic CRS: WGS 84
   name                 geometry
1 Bizan POINT (134.5165 34.0607)
```

      - data.frameとして格納される

```
> class(b_sf)
[1] "sf"         "data.frame"
```

53

次が sf クラス、地理情報に属性を付加したものです。

sf パッケージで使うオブジェクトの特徴として、sp パッケージのオブジェクトは全部 list ですが、sf パッケージのオブジェクトは data. frame として格納されます。

空間統計

- sfパッケージ
  - sfクラス
    - 地理情報（座標など）はgeometry列に組み込まれている

```
> b_sf
Simple feature collection with 1 feature and 1 field
geometry type:  POINT
dimension:      XY
bbox:           xmin: 134.5165 ymin: 34.0607 xmax: 134.5165 ymax: 34.0607
geographic CRS: WGS 84 (with axis order normalized for visualization)
   name                 geometry
1 Bizan POINT (134.5165 34.0607)
```

    - spパッケージのオブジェクトと互換性をもつ

```
b_sp <- as(b_sf, "Spatial")
> class(b_sp)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
```

```
b_sf <- st_as_sf(b_sp)
> class(b_sf)
[1] "sf"         "data.frame"
```

54

地理情報は、最後に geometry 列ができて、ここに全部入っています。1 オブザベーションごとに地理情報とその属性がセットになって格納されている感じです。

もし sf パッケージを使ってデータを読み込んだあと、本を読んで sp パッケージの使い方は分かるが sf パッケージはよく分からないという場合、sp パッケージのオブジェクトと関数を使って行ったり来たりできます。ですので、sf パッケージで入れて sp パッケージのオブジェクトに変換してから、コピーした関数などで

運用できます。

空間統計

- sfパッケージ
  - sfオブジェクトの取扱い
    - tidyverseパッケージ等と互換性を持つ
      - dplyrパッケージによるデータ操作が可能

```
jpn_sf <- st_read(dsn = ".", layer = "N03-22_220101")
> head(jpn_sf)
Simple feature collection with 6 features and 5 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 139.334 ymin: 37.73383 xmax: 148.8944 ymax: 45.55724
geographic CRS: JGD2011
  N03_001 N03_002 N03_003 N03_004 N03_007                        geometry
1  北海道     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((139.7994 41...
2  青森県     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((139.936 40....
3  岩手県     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((141.7176 38...
4  宮城県     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((141.5056 38...
5  秋田県     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((139.8783 39...
6  山形県     <NA>     <NA>     <NA>     <NA> MULTIPOLYGON (((139.5472 38...
```

```
tp_sf <- jpn_sf %>% filter(str_detect(N03_007, "^36")) %>%
          group_by(N03_004) %>% summarise()
```

出典：「国土数値情報（行政区域データ）」（国土交通省）https://nlftp.mlit.go.jp/ksj/jpgis/datalist/KsjTmplt-N03.html

55

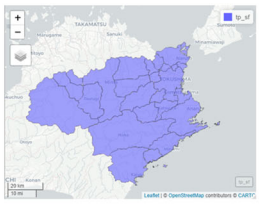
data.frame なので dplyr を使った運用もできます。地理情報を持ったままの運用です。ここでは国土地理院から都道府県のデータを sf パッケージの関数でインポートして、36 で始まるデータのみをフィルタを掛けて抽出しています。

空間統計

- sfパッケージ
  - sfオブジェクトの取扱い
    - tidyverseパッケージ等と互換性を持つ
      - ggplot2・mapview・tmap等のパッケージによるプロットが可能

```
ggplot(tp_sf) +
  geom_sf()
```

```
mapview(tp_sf)
```



```
tm_shape(tp_sf) +
  tm_polygons()
```

56

data.frame として tidyverse が使えますので、ggplot2 も使えます。sp パッケージのオブジェクトだと plot 関数か spplot 関数ぐらいしか使えなくて苛々することがありましたが、こちらだと ggplot2 も使えますし、インタラクティブに使いたかったら mapview を使ってアウトプットを動かしたりもできます。ですので、こちらの方が手元で使う分にはかなり便利ですし、時代は sf パッケージに移っていますので、sp パッケ

ージも先ほど言った通り知識としては必要なのですが、現状は両方の情報を知る必要があります。

4．おわりに

57

参考文献

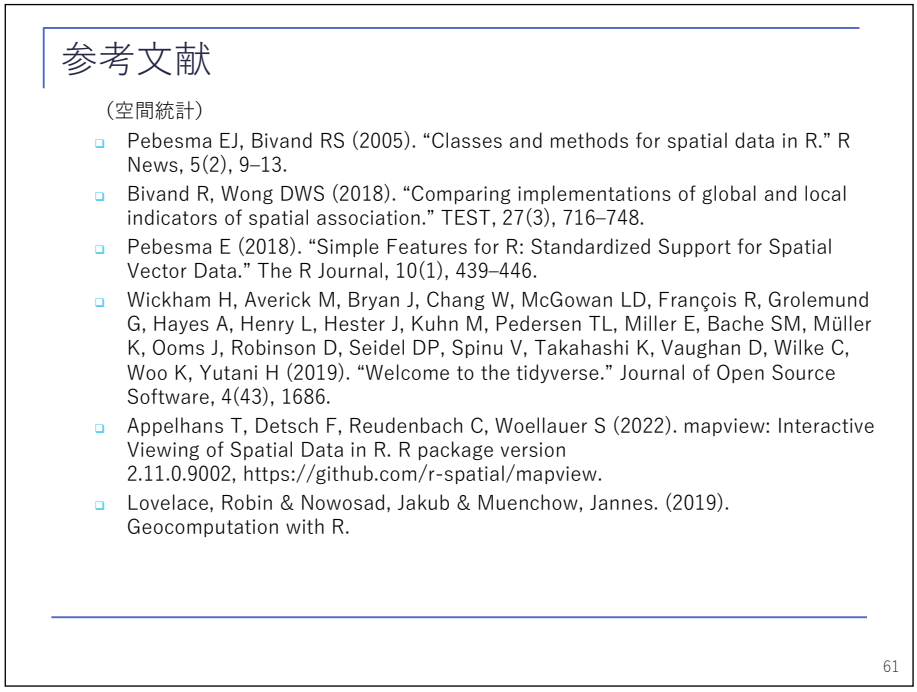
（共通）

- Charpentier, A. (Ed.). (2014). Computational actuarial science with R. CRC press.

（ベイズ的手法）

- 松浦健太郎. (2017). Stan と R でベイズ統計モデリング: Wonderful R 2 (Vol. 2). 共立出版.
- 渡辺澄夫. (2012). ベイズ統計の理論と方法. コロナ社.
- Hooten, M. B., & Hobbs, N. T. (2015). A guide to Bayesian model selection for ecologists. Ecological monographs, 85(1), 3-28.
- Stan Development Team, stan-jaチーム訳, (2020). Stan モデリング言語: ユーザーガイド・リファレンスマニュアル, Stan Version 2.16.0.

60



ということで、最後に参考文献二つ並べて、私からの話は終わりにします。ご清聴ありがとうございました。

【司会】　佐野さん、ありがとうございます。

【大会委員】　Slido から質問を 2 件いただいていますので、読み上げさせていただきます。

1 件目は、渡辺様への質問となります。

「本日、MCMC の実装として主に Stan をご説明いただきましたが、他のツールも含めて、お勧めのツールや使い分けがございましたら、ご教授ください」。

【渡辺】　スライド p. 18 にいくつかツールを挙げていますが、それぞれ特徴、長所、短所があります。

四つのツールを挙げていますが、中でも WinBUGS が一番古いものです。古いということは、それなりに、皆さん使い込んで、割と枯れているという状態です。その代わり、最近は開発が止まっていると聞いています。また、この中では、一番多くのモデルに対応しているのではないかと思います。

その下の OpenBUGS は、WinBUGS と同じような使い方ができます。これら二つと、その下の JAGS と Stan には大きな違いがあります。上の二つは、計算の都度、R とは別のツールが起動します。結果も、その別のツールの方に格納されて、R ではこの結果を利用するための関数がパッケージとして提供されていますが、計算結果の全体は元のツールの方にしかないので、R に慣れた人にとってはやや使いにくいかと思っています。

それに対して JAGS や Stan であれば、結果が丸ごと R 上のオブジェクトとして返ってきますので、普通の R の関数の結果と同じような感覚で扱えるという意味では、使いやすいように思います。

JAGS と Stan の関係については、先ほどお話しした通りではあります。Stan で使えるモデルであれば Stan を使うのが一番使いやすいのではないかと、私としては思っています。ただ、機能が限られているところもありますので、必要に応じて JAGS や BUGS を使っていくという使い方がいいのではないかと思います。

【大会委員】　ありがとうございました。

もう 2 件質問が入っておりまして、いずれも佐野様に、空間情報に関連してですので、併せてご質問させていただきます。

一つが、「空間情報を扱うことができたことで、これまでは見えなかった何かが見えてきたといったことで、分かりやすい例があれば教えていただけるでしょうか」。

もう一つは、「R で GIS を行うことのメリットについて教えてください。他のツールなどでも可能でしょうか」。

ということで、よろしくお願いいたします。

【佐野】　一つ目は、都道府県のデータを扱っていると、日本は縦に長いということがよく分かります。空間モデルについて本を読むと大体の例はアメリカの州で、アメリカの州は平べったい形をしているので使いやすい。それを日本に当てはめて本のとおりに使えるのかというと、日本は細長く地域が分かれているので、なかなか思ったようにはならない。あと島が分かれていて、例えば四国は他と海で分かれていますが、文化などが独立しているのかというと、そういうわけではない。むしろ海を隔てている方が近い文化を持っていたりする。空間情報は事前の情報として自分で与える必要がありますが、そのときにただ本に書いているとおりにやっていたら難しい。実際に動かして分かる部分があるのではないかと思います。

二つ目は完全に主観ですが、自分が以前からRを使っているのであれば、RでできることにプラスしてGISとしてできることが増えるのが一番大きなメリット。前から他のGISを使っているのであれば共感できないかもしれませんが、Rが先にあるのなら、それが一番のメリットかと思います。他にも探せばいろいろありますが、特に「これだ」と思って自分が使っているわけでもありません。

【大会委員】　ありがとうございました。ウェブからの質問は以上になります。

【司会】　以上をもちまして、セッションA-1「損保アクチュアリーのためのRによる計算保険数理」を終了します。

発表されたお二人に拍手をお願いします。