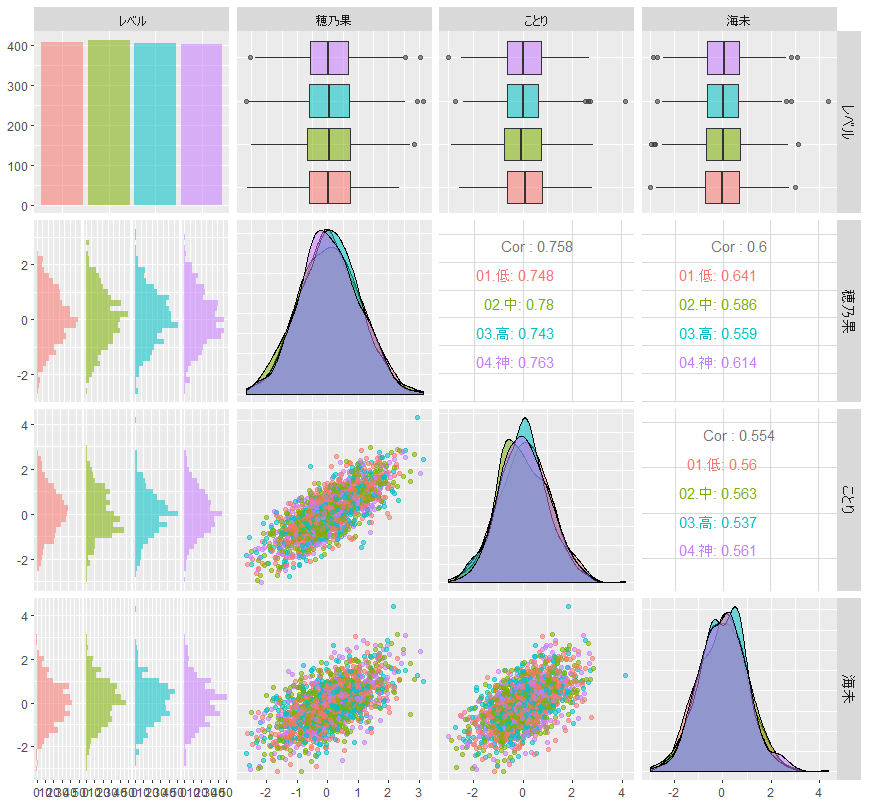
***ラブライブ！*とうけいもでりんぐ**

****

**ラブライブ！キャラクター人気投票の数理**

**1. はじめに**

　政治、スポーツ、ビジネスなどあらゆる世界でランキングをつける習慣が根付いています。アニメの世界でも例外ではなく、どのキャラクターが人気があるかなどを評価して、Web上での話題のネタになったり、対立の種になったりしています。

ラブライブ！の世界でも例外ではなく、定例的に実施される総選挙だけでなく、特定のシチュエーションを想定した人気投票が多数実施されています。

これらの多くはたいてい多数決によって1位を決めること、つまりある選択集合のうち1つだけを選びその得票数が1番多かった者にno1の称号を与えるということが自然と受け入れられています。この自然と受け入れられいる多数決による投票に疑問符を投げかけ、統計的アプローチにより精度の高いランキングづけすることが本稿の目的です。

　このような多数決による人気投票の仕組みをもっとも公平で民主主義的な投票制度とするためには以下の4つの条件が満たされている必要があります。

1. **定義域の非原性**すべての投票者は候補者を望みのままの順番でランク付けできなければならない。
2. **パレート原則**候補Aと候補Bを比較した場合、すべての投票者が候補Aを選ぶならば、つねにAがBより上にランク付けされなければならない。
3. **非独裁性**投票者はだれひとりとして、ランキングを左右するような権限を持っていてはいけない。  
   これは例えば、最後の1票でランキングが決まるような状態になっているということでしょうか。
4. **無関係な選択肢からの独立性(I.I.A)**例えば2人の候補者AとBがそれぞれ60％と40％の得票率を得られると仮定した場合、AがBより上位であるが、2人以外のCの候補者が選択候補に入り、A、B、Cそれぞれの候補者の得票率が30％、40％、30％と変化した場合、AとBの順位は逆転した状態となっている。この状態をI.I.A特性を満たさない状態という。  
   つまりI.I.A特性を満たす条件というのはCの候補者が乱入しても、A、Bの相対的な順位は変化してはならないということです。

経済学者のケネス・アローはこの4つの条件を同時に満たす”完璧”な投票制度は存在し得ないと1951年の論文で証明しました。なので、実質的に完璧な人気投票やランキングづけを考えることは無意味なことのように思えますが、今ある多数決の投票よりよいアプローチでランキングづけを考えることは可能です。

ここでは特に自分自身が興味のある④の無関係な選択肢からの独立性(I.I.A特性)の緩和を目指してラブライブキャラクターのランキングづけを提案してみようと思います。

**2. 無関係な選択肢からの独立性とは？**　前節でI.I.A特性について少し説明しましたが、統計的なアプローチを用いて具体的に説明したいと思います。統計的なアプローチである「離散選択モデル」で投票のシミュレーションをしたいと思います。離散選択モデルとは選択集合からある選択肢の集合から1つあるいは複数を選択する行動を統計モデルとして表現する数理モデルの1つです。離散選択モデルはある選択集合に対しての相対的な効用によって選択結果が変わると仮定します。効用とはその選択肢に持っているスコアであり、例えばにこ、真姫、ことりの3人の選択肢から1番好きなキャラクターを1人だけを選ぶ場合、3人に対する効用の値がそれぞれ50点であるとき、選択確率は3人とも33％になります。

このように3人に対する効用の値が同じであれば3人の選択確率が同じになる場合を「無関係な選択肢からの独立性を満たす」と言います。

これを満たさないパターンを考えるためににこと真姫が相関しているケースを考えます。これはにこが好きなら真姫も好きだ。ということを意味しています。この効用パターンを数理的に表すために効用は以下の多変量正規分布に従うと仮定します。

この複雑な数式のΣは分散共分散行列(あるいは相関行列を)をｘは効用の実現値をμは効用の平均構造を表しています。以下は相関行列の例です。

この分散共分散行列の値により、効用値が同じ場合でも選択確率が大幅に変化してしまうことをシミュレーションにより示します。

具体的にはにこ、真姫、ことりの効用の平均構造を3人とも0に固定し、にこと真姫の間の相関係数をにこと真姫の相関係数を-1～1まで変化させ(上記の相関行列の上の行列の2行1列目と1行2列目がにこと真姫の相関パラメータを表しており、この0.5の値の部分を変化させます。)、多変量正規分布から指定したパラメータに基づく乱数を大量に発生したときにそれぞれの乱数で最大の効用値が出たキャラを選択結果として、選択確率の変化の仕方を観察してみます。以下のグラフに示すように選択確率が大幅に変化しているのが見て取れます。

図1. にこまきに相関がある場合の3人の選択確率の変化

シミュレーション結果を見てみると相関の絶対値が大きいほど相関が0の場合(つまりI.I.A特性が満たされている場合)と比較して、選択確率が大きく歪んでいることが見て取れます。負の相関がある場合、にこまきの選択確率はことりちゃんと比較して有利に正の相関がある場合は不利になるというパターンがあります。その確率の変動は大きく、ことりの選択確率は25％から50％まで大幅に変化しています。(このシミュレーションを実行するRコードはAppendixを参照してください。)

この結果は必ずしも、前述の効用値がもっとも高いキャラクターが1位になることが限らないことを示し、キャラクター間の相関構造が選択結果に大きく影響を及ぼしていることがわかります。

I.I.A特性は特に政治の世界ではかなりの問題で2000年アメリカ大統領選挙のアルゴア(民主党)とブッシュ(共和党)の戦いでしたが、民主党に近いラルフ・ネーダーが大統領選挙に加わったため、アルゴアの投票の一部を奪ってしまい、ブッシュが勝利したという事例があります。

**3. スクフェスのデータからラブライブキャラの人気と相関構造を探る**

　この節では実際のデータを用いて、ラブライブキャラの人気とキャラ間の相関構造を推定したいと思います。解析するデータはスクフェスの７月下期イベント「第1回なかよしマッチ」のscoreランキングの上位に入ったユーザーから1628件をサンプルデータとして取得しました。ユーザーデータには現在パートナーに設定しているキャラクターと累積スコアが高いキャラクター上位3位までを取得することができるのですが、今回解析に用いるのは累積スコア上位3位のデータを取得しています。

抽出条件として、scoreランキングを選んだのは累積ポイントと比較してイベント対象キャラに依存しにくいと考えたからです。また累積スコア上位3位のキャラクターをデータとして選んだのは、8月1日に穂乃果ちゃんが誕生日だったのと集めたサンプル数ではパートナーデータから相関構造が推定できなかったためです。集めたデータレイアウトの一部を以下に示します。

図2 ラブライブランキングデータ



データの単純集計は以下の通りです。Aqoursで選択回数が少ないキャラクターはAqoursとして一つにまとめています。すみません。

　このスクフェスデータによるとことりがもっとも1位に選ばれた回数が多いようです。(このランキング結果は正直予想していませんでした。またAqoursは不利なことに注意してください。)

図3 メンバーごとの選択順位およびパートナーでの選択数の集計(n=1628)

このデータから人気と相関構造を探るために離散選択モデルを用います。離散選択モデルでは以下のような効用関数を用います

… ①

… ②

Uは効用値を表し、UはV(効用の確定項)とε(確率的変動項)の和として表現されます。さらにVはZ(説明変数)とβ(パラメータ)の線形結合として分解できます。βは説明変数に対する影響度を表すパラメータでこの中の切片項のパラメータ(β0jの部分)の高低を人気度の指標として利用します。(マーケティングサイエンスの分野ではこの切片が価格やプロモーション要素を取り除いた時のブランド力と定義することが多いです。)

離散選択モデルを扱う実務でもっとも使われるのは以下の数式で表される多項ロジットモデルです。

　… ③

このモデルはある選択肢の選択確率Prが0～1の間に、Prの全選択肢の確率の和が1になることを満たしています。このモデルの欠点は①で表したεがお互いの選択肢で独立と仮定しており、相関が表現できません。つまり相関が0に固定されていますので、I.I.A特性が成り立たたない選択を表現できません。また今回のデータのように1位2位3位のような複数選ぶ選択に対してはこのモデルでは無力です。このモデルでこのようなランキング集合をモデル化しようと思えば、以下のモデルのように表現することが出来ます。例えば選択肢の集合ににこ、真姫、ことり、穂乃果がある場合に1位にこ、2位真姫、3位ことり、4位穂乃果を選んだ場合は

となります。

　このタイプの数式で表された離散選択モデルの①で表された確率変動項が極値分布という確率分布に従うと仮定しています。このタイプのモデル(確率変動項が極値分布に従うロジット型モデル)で選択肢間に相関を持たせるために様々なモデルが例えば提案されています。例えばNested Logit model、Paired Combinatorial Logit model、Cross Nested Logit model、Generalized Nested Logit model、Mixed Logit modelなどがあります。

もう1つのタイプの離散選択モデルとして①の効用関数の確率変動項が極値分布ではなく正規分布に従うと仮定したプロビットモデルが考えられます。今回の解析ではロジットタイプのモデルではなくこのプロビットタイプのモデルを推定に用います。

　多肢選択の場合、確率変動項が以下の多変量正規分布に従います。

… ④

多変量正規分布に従う場合、2節で示した通りΣの分散共分散行列で選択肢間の様々な相関関係を柔軟に表現することができます。ただしこのモデルで選択確率を表現すれば、以下のような多重積分を解かなければなりません。(積分数は選択肢数-1次元になり、今回のラブライブ解析の場合、15次元の多重積分となります。)

… ⑤

は②で表される効用の確定項で、uは効用をΣは相関構造を表しています。また積分記号の右下についている積分領域Rは以下の条件を示します。

… ⑥

要するにこの条件は、選ばれた選択肢jの効用がすべての選択集合の中でもっとも大きくなければならないという条件を表しています。なおプロビットモデルでは選択肢Jを基準選択肢として、他の選択肢jとJの相対的な効用値を効用関数として用います。

⑥で表される条件は選択集合の中から1つだけを選ぶ際の条件なので、これをランキングデータに適用する際には積分領域Rは以下のような条件に変わります。例えば、3位までランク付けをした場合

… ⑦

となります。この条件は選択肢に対する効用値が選ばれたランキングの順序どおりになることを要請しています。例えば図2のデータの場合、no1のユーザーは1位真姫、2位花陽、3位穂乃果を選んでいるので、効用の順位は真姫>花陽>穂乃果という順になります。

　スクフェスランキングデータを⑤のモデルを⑦の条件で解くためには多重積分が必要ですが、それは困難なので、データ拡大法に基づくマルコフ連鎖モンテカルロ法(MCMC)でベイズ推定を行うことで推定します。データ拡大法と呼ばれる手法は選択結果と整合的な⑦の条件に基づく潜在効用を多変量正規分布の乱数から大量に発生させることで多重積分を回避します。このデータ拡大法を用いることで効用uに対する積分をuの乱数に置き換えることで連続変数になる、積分を回避して推定すべきパラメータβおよびΣは多変量線形回帰モデルを解くことに帰着することになります。今回のこのモデルを多項型のランクプロビットと呼ぶことにします。

　さらに今回のモデルでは発生させた潜在効用uを用いて、メンバー間の関係を因子分析モデルにより表現するモデルを付け加えています。(ただ今回はこの因子分析モデルはMCMCでベイズ推定がうまくいかず、ラ４ンクプロビットを推定した後に事後的に推定されたΣを用いて最尤法で因子分析を実施しています。)

なお推定に用いたRコードはAppendixを参照してください。

**4. モデルに用いた説明変数**

今回のランクプロビットモデルのデータには図2のスクフェスランキングデータを用います。選択集合となるメンバーは以下の15名+モブとなります。

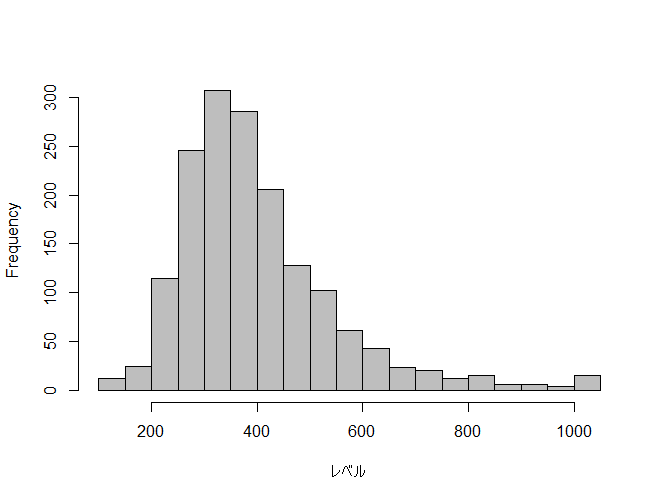
穂乃果、ことり、海未、凛、花陽、真姫、にこ、希、絵里、千歌、曜、花丸、善子(ヨハネ)、Aqours

なおルビィ、鞠莉、果南、ダイヤは出現数が少なく、推定が難しいと感じられたため、Aqoursとしてひとつにまとめています。

今回の目的の一つに②のβ0jで示されるメンバーごとの切片項がどのような順序になっているかを知ることです。その選択結果の集計グラフは以下に示します。(n=1628)

図4 選択結果のランキング集計(n=1628)

説明変数としてもう1つレベル図2にあるlv(レベル)を用います。おそらくスクフェスのレベルはスクフェスの開始時期と関連があるので、レベルを開始時期の代理変数として用います。この説明変数のパラメータβの大小はスクフェス加入時期により、好むメンバーは異なるのか？を説明します。なおこのレベルの変数はスケールが大きいので標準化を行います。

図5 スクフェスプレイヤーの上位1628位までのレベル分布

標準化したレベルと元のレベルの対応関係は以下の通りです。これを見るとレベルが平均値より低い人が半数以上を占めていることが分かります。

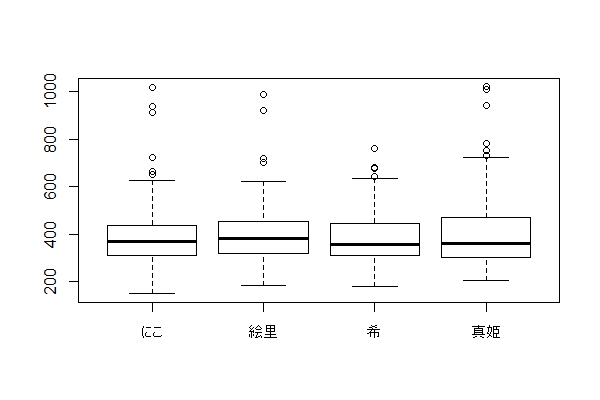
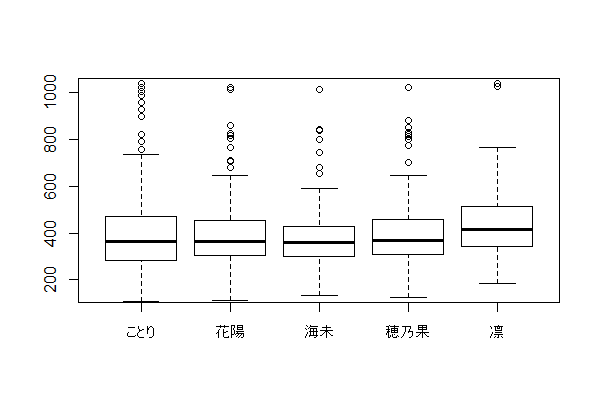
またメンバーごとのレベルと1位選択の関係を箱ひげ図として図7に示します。これを見ると凛、モブあたりがレベルが高いと選ばれる傾向があるように見られます。



図6 レベルと標準化レベルの対応表



図2 再掲



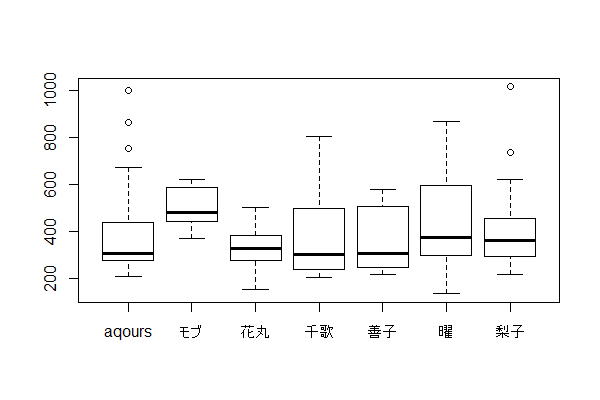


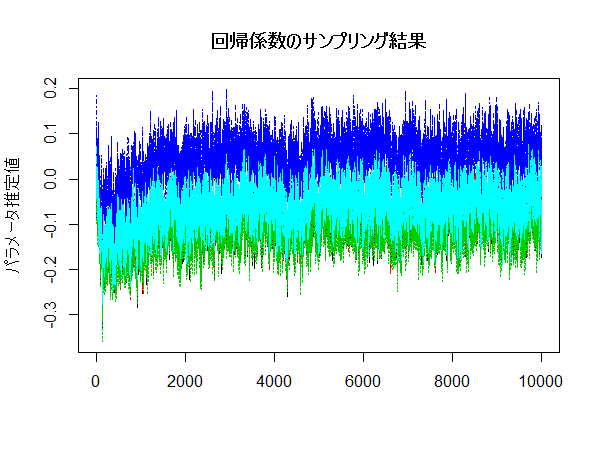
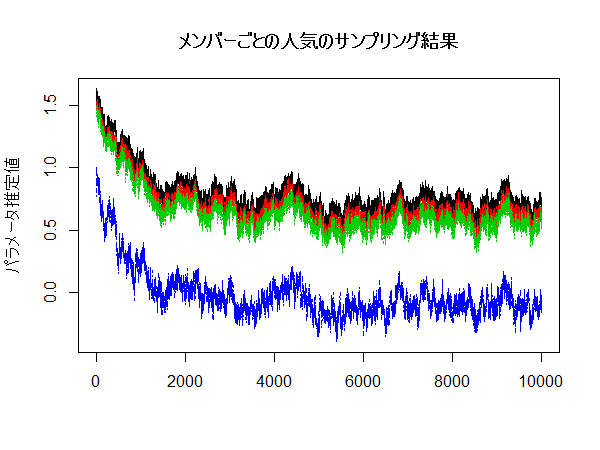
図7 メンバーごとの1位選択とレベルの関係(y軸=レベル、x軸=メンバー)

以上の説明変数を用い、あるユーザーiの効用関数Uをメンバーごとに行列表現で以下のように表現します。

… ⑧

**5. モデルの推定結果と解釈**

　ランクプロビットモデルをマルコフ連鎖モンテカルロ法でパラメータを40,000回サンプリングし、10,000回までをburnin期間として捨て、残りの30,000回のサンプリングで推定結果としました。なお、サンプリングは4回に1回だけ抽出しました。そのサンプリング結果のプロットの一部を以下に示します。グラフのy軸はパラメータ推定値、x軸はサンプリング回数を表します。



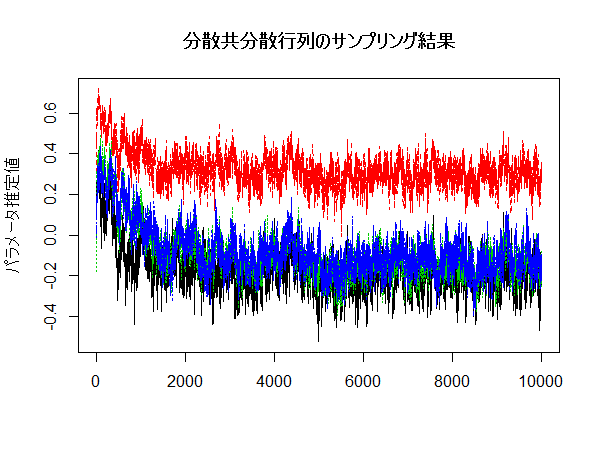
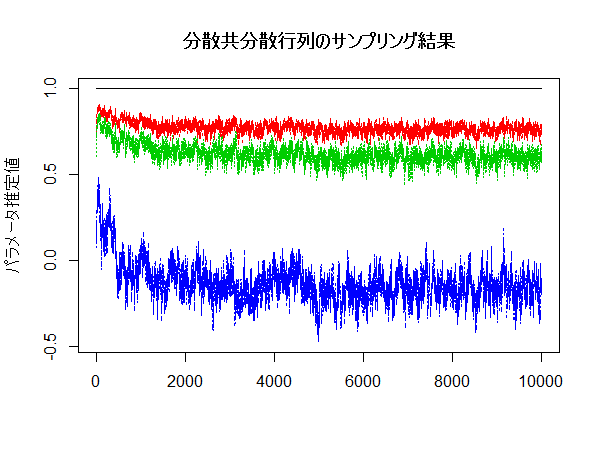


図8 MCMCサンプリング結果のプロット

まずはLVの説明変数のメンバーごとのパラメータ推定値を見てみましょう。結果は以下の通りです。



図9 Lvの説明変数のメンバーごとのパラメータ推定値

　レベルのパラメータの事後平均について見てみると、ほぼ全員のメンバーがマイナスの数値が出ており、レベルが低いほどそのメンバーが選ばれやすいことを示しています。特に値が小さいのはにこ、希、絵里の3年生組とAqoursです。海未、真姫もやや低くなっています。レベルをスクフェス参入時期の代理変数と捉えるならば、より新規のユーザーほどこれらのメンバーを選びやすくなります。

反対にパラメータが正の値であるのは凛のみとなっており、凛を選ぶユーザーは他のメンバーを選ぶユーザーと比較してややレベルが高い傾向にあるようです。

　次にメンバーごとの切片のパラメータを事後平均を見ていきます。推定結果は以下の通りです。



図10 切片(人気度)のメンバーごとのパラメータ推定値

この切片のパラメータの解釈はメンバー選択のレベルの要素を取り除いた時のメンバー選択の影響を表したパラメータと解釈できます。それを踏まえて事後平均の推定値を見てみると、μ’ｓの中で高い順に並べると穂乃果、ことり、海未、真姫、花陽、にこ、絵里、凛、希の順になっています。

　3年生組のパラメータ値が低くなっていますが、3年生組はレベルが低いほど選ばれる確率が高くなる傾向があったので、比較的新規のユーザーに絞って見ると選ばれやすいということが出来ます。2年生組は3人ともに高い推定値になっており、また1位選択数と比較するとパラメータの順序がことりと穂乃果が逆転していることが見て取れ、穂乃果と海未とも差が開いていることが見て取れます。この理由はメンバー間の選択にI.I.A特性が満たされておらず、ラブライブのメンバー選択に何らかの構造をあることを示しています。その構造についての詳細は後述します。

Aqours間の比較では全体的にパラメータの事後平均が低いですが、これはAqoursの参入が遅かったためでμ’ｓより不利になるためです。Aqoursはμ’ｓと比較してメンバー間でパラメータの事後平均がフラットな傾向があります。また、1位選択数がヨハネがもっとも多いですが、パラメータはヨハネよりちかようのほうが高くなっています。

　μ’ｓのメンバー間のパラメータの事後信頼区間を以下にプロットします。このグラフはメンバー間でパラメータ推定値が重なっていなければそのメンバー間の人気度が統計的に見て明確に差があると見ることが出来ます。



図11 切片(人気度)のメンバーごとのパラメータの事後信用区間

最後に相関行列のパラメータの事後平均を以下の表に示します。





図13 μ’ｓメンバーの相関行列の可視化

相関行列を見るといくつかの構造を持っていることが分かります。大きなグループとして、μ’ｓ1年生、2年生、3年生および、Aqoursでは全員がお互いに正の相関を持っています。他に大きなグループとして、Printemps、BiBiが正の相関を持ち、そしてlily whiteは特に相関が出ているわけではないようです。

μ’ｓとAqours間の相関関係を見ると穂乃果とことりおよび海未しか正の相関を持ちません。そしてこの2人は相関の仕方が非常に似ています。

　切片のパラメータ推定値が凛と希が低かった理由がこのメンバー間の相関構造で説明できます。その理由はこの2人は他のメンバーと正の相関の度合いが低いことにあり、このメンバーの効用が高いと他のメンバーの効用が負の相関があるため低くなり、結果的に低い切片値でも1位選択に選ばれる可能性が高くなります。反対に穂乃果、ことり、海未はお互いに正の相関が強いため、つまり3人とも好きとなる可能性が高いため(効用値が3人同時に高くなるため)、お互いに票を食い合うことになり、3人それぞれが1位に選ばれる確率が相関のない場合と比較して低くなってしまいます。

　　以下個人的なカップリングに対する考察です。

|  |
| --- |
| 1. 穂乃果と相関が高いのは海未よりことり。 2. ちかようよりちかりこ。 3. にこまきはカップリングとして比較的妥当。ただ同じ3年生の絵里のほうが相関は高い。 4. 独立傾向にある凛は花陽とはやっぱりカップリングとして妥当性がある。 5. ほのことが同じ似た相関構造をもつならようちかも似た相関構造を持っている。 6. 花陽と曜が弱い正の相関を持っている。 7. Aqoursでもμ’ｓのように今後メンバー内でいくつかのサブグループができる相関構造を持つのだろうか？ |

　次節では切片の推定値が全メンバーで同じ場合でこの相関行列が与えられたらどのメンバーがもっとも選ばれるのかをシミュレートします。

　さらにこの相関行列から因子分析モデルを推定します。因子数はスクリープロットによ5因子を選びました。その結果は以下の通りです。結果を見ると前述したとおりのメンバー間に構造があることが分かります。

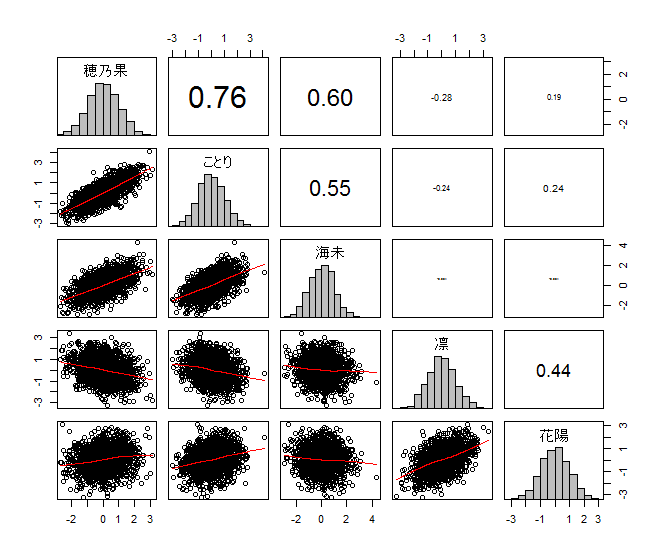
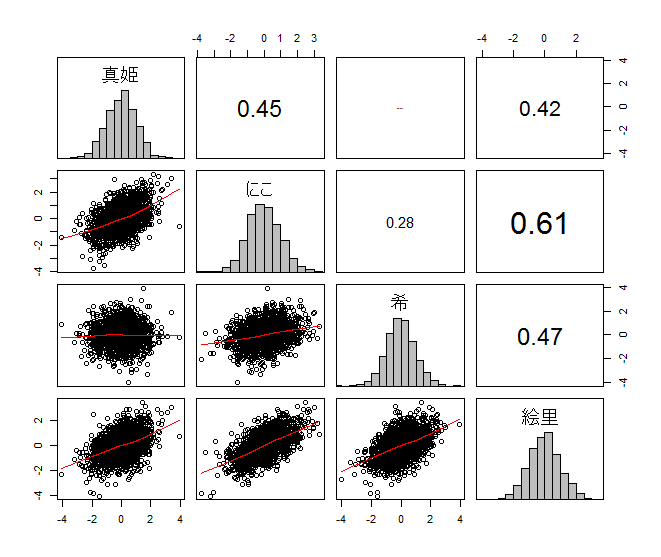


図14 因子分析モデルの推定結果

因子に対して名前をつけるとすれば、因子1はAqours因子、因子2はμ’ｓ2年生因子、因子3はBiBi因子、因子4はμ’ｓ1年生因子(花陽型)、因子5はμ’ｓ1年生因子(凛型)というところでしょうか。なお表中の独自性は5つの因子で説明できない部分を表し、共通性は説明できる部分を表しています。独自性が多くのメンバーで高くなっていることから5因子で説明できない構造がまだあるかもしれません。

　Aqoursが1つの因子にまとまりμ’ｓが4つのサブグループに分かれているのが見て取れます。Aqoursが今後μ’ｓのようにこのようなサブグループに分かれていくのを見ていくのも面白いでしょう。また他のアニメにもこの傾向が見られるのかが気になります。

　以下のグラフはMCMCのあるサンプリング回数における発生させた潜在効用間の散布図行列を示しています。見ての通り、元データのランキングデータが潜在効用に置き換わることにより、離散変数が連続変数でなおかつ正規分布になっていることが見て取れます。

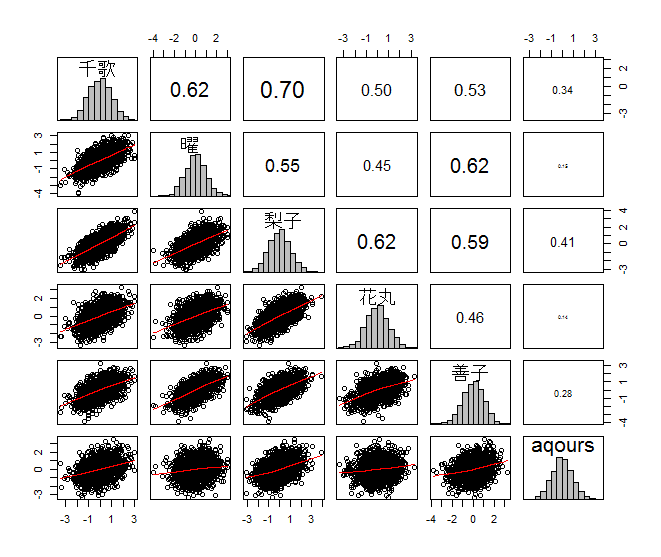


図15 あるサンプリング回数における潜在効用間の散布図行列

**6. 相関行列のシミュレーション**

　この節では5節で与えられた相関行列を用いシミュレーションを行います。パラメータ推定結果から切片とレベルによりメンバーの選択に差があることが分かりました。そこでこれらのパラメータを全メンバーで同じとして、相関行列を与えればどのような選択結果になるのかをシミュレーションで検証します。このシミュレーションの意図として人気度が同じでもI.I.A特性を満たさなければ選択確率は大幅に変わってしまうことをより深く理解するためです。もしI.I.A特性を満たせば、パラメータが同じ場合選択確率はすべてのメンバーで同じになるはずです(つまり全員が1/15=6.67％)。

　具体的なシミュレーション手法として切片とレベルのパラメータを全員0に固定させ、5節で推定された相関行列の事後平均を与えた多変量正規分布から300,000回乱数を発生させ、メンバーの選択確率を見ます。その結果は以下の通りです。



図16 相関行列を与えた場合の選択確率のシミュレーション

　結果を見ると希、凛が選択確率が大幅に向上し、千歌、梨子、穂乃果、曜、ことりが大幅に選択確率が低下していることが見て取れます。特に千歌は実際に実施されている人気投票の結果が芳しくなく、その一因として、他のメンバーと相関係数が高いことが一因になっているかもしれません。

　以上の結果からメンバー間の相関を無視した人気投票は思わぬ結果を生むことが理解できたと思います。

**Appendix**

　今回の解析で用いたRコードを掲載します。

①I.I.A特性を説明するためのシミュレーションコード

|  |
| --- |
| library(MASS)  ##相関の変化による選択確率の変動をシミュレーションで確認  n <- 500000  rank <- c(0.5, 0.5, 0.5)  cc <- seq(-1, 1, length=101)  Rank.ones <- matrix(0, nrow=n, ncol=length(cc))  Rank.agg <- matrix(0, nrow=length(cc), ncol=length(rank))  for(i in 1:length(cc)){  print(i)  CORM <- matrix(c(1, cc[i], 0, cc[i], 1, 0, 0, 0, 1), nrow=3, ncol=3)  U <- mvrnorm(n, rank, CORM)  colnames(U) <- c("にこ", "真姫", "ことり")    first <- apply(U, 1, which.max)  Rank.ones[, i]　<- ifelse(first==1, "にこ", ifelse(first==2, "真姫", "ことり"))  Rank.agg[i, ] <- table(Rank.ones[, i])  }  #変数に名前をつける  colnames(Rank.agg) <- c("ことり", "にこ", "真姫")  rownames(Rank.agg) <- round(cc, 2)  #可視化と結果の確認  matplot(cc, Rank.agg/n, type="l", xlab="相関係数", ylab="選択確率", lwd=2, main="メンバー間の相関による選択確率の変化")  round(Rank.agg/n, 3) |

②ランクデータ前処理およびランクプロビットモデル推定のためのRコード

|  |
| --- |
| #####潜在変数ランクプロビットモデル#####  library(MASS)  library(bayesm)  library(MCMCpack)  library(psych)  library(gtools)  library(MNP)  library(reshape2)  library(dplyr)  library(ggplot2)  library(lattice)  library(qgraph)  ####多変量正規乱数を発生させる関数####  ##多変量正規分布からの乱数を発生させる  #任意の相関行列を作る関数を定義  corrM <- function(col, lower, upper){  diag(1, col, col)    rho <- matrix(runif(col^2, lower, upper), col, col)  rho[upper.tri(rho)] <- 0  Sigma <- rho + t(rho)  diag(Sigma) <- 1  Sigma  (X.Sigma <- eigen(Sigma))  (Lambda <- diag(X.Sigma$values))  P <- X.Sigma$vector  P %\*% Lambda %\*% t(P)    #新しい相関行列の定義と対角成分を1にする  (Lambda.modified <- ifelse(Lambda < 0, 10e-6, Lambda))  x.modified <- P %\*% Lambda.modified %\*% t(P)  normalization.factor <- matrix(diag(x.modified),nrow = nrow(x.modified),ncol=1)^0.5  Sigma <- x.modified <- x.modified / (normalization.factor %\*% t(normalization.factor))  eigen(x.modified)  diag(Sigma) <- 1  round(Sigma, digits=3)  return(Sigma)  }  ##相関行列から分散共分散行列を作成する関数を定義  covmatrix <- function(col, corM, lower, upper){  m <- abs(runif(col, lower, upper))  c <- matrix(0, col, col)  for(i in 1:col){  for(j in 1:col){  c[i, j] <- sqrt(m[i]) \* sqrt(m[j])  }  }  diag(c) <- m  cc <- c \* corM  #固有値分解で強制的に正定値行列に修正する  UDU <- eigen(cc)  val <- UDU$values  vec <- UDU$vectors  D <- ifelse(val < 0, val + abs(val) + 0.00001, val)  covM <- vec %\*% diag(D) %\*% t(vec)  data <- list(covM, cc, m)  names(data) <- c("covariance", "cc", "mu")  return(data)  }  ####データクレンジング####  ##データの読み込み  sf\_data <- read.csv("スクフェスデータセット.csv")  hh <- nrow(sf\_data) #サンプル数  #factor型を文字列に変更  sf\_data$X1位 <- as.character(sf\_data$X1位)  sf\_data$X2位 <- as.character(sf\_data$X2位)  sf\_data$X3位 <- as.character(sf\_data$X3位)  r <- c(sf\_data$X1位, sf\_data$X2位, sf\_data$X3位)  table(c(sf\_data$X1位, sf\_data$X2位, sf\_data$X3位))  unique(c(sf\_data$X1位, sf\_data$X2位, sf\_data$X3位))  ##モブとaqoursを1つの変数にまとめる  r1 <- ifelse(r %in% c("いるか", "姫乃", "咲", "咲良", "ココ", "かさね", "遊宇", "理亜", "紗菜", "あきる", "ななか",  "咲夜", "仁美", "千鶴子", "小雪", "瑞希", "優理", "クリスティーナ"), "モブ", r)  r2 <- ifelse(r1 %in% c("ダイヤ", "鞠莉", "果南", "ルビィ"), "aqours", r1)  rank <- matrix(r2, nrow=nrow(sf\_data), ncol=3)  ##モブやaqoursが同時に選ばれているサンプルは除去  #aqoursの重複を削除  index1 <- subset(1:nrow(rank), rowSums(rank=="aqours") > 1)  if(length(index1)==0) {  print("重複なし") } else {  rank <- rank[-index1, ]  sf\_data1 <- sf\_data[-index1, ]  }  #モブの重複を削除  index2 <- subset(1:nrow(rank), rowSums(rank=="モブ") > 1)  if(length(index2)==0) {  print("重複なし") } else {  rank <- rank[-index2, ]  sf\_data1 <- sf\_data[-index2, ]  }  ##メンバーごとに整理番号をつける  t(t(table(rank)))  Rank <- rank #新しいランクデータ  lovelive <- c("穂乃果", "ことり", "海未", "凛", "花陽", "真姫", "にこ", "希", "絵里", "千歌", "曜", "梨子",  "花丸", "善子", "aqours", "モブ")  #メンバーごとに番号をつける  for(i in 1:length(lovelive)){  Rank[Rank==lovelive[i]] <- i  }  Rank <- apply(Rank, 2, as.numeric) #文字列になっている番号を数値型に変更  data.frame(Rank, rank) #データを確認  member <- length(lovelive) #メンバー数  hh <- nrow(Rank) #サンプル数  ##説明変数をベクトル形式のデータフォーマットに変更  #IDを設定  id <- rep(1:hh, rep(member-1, hh))  m <- rep(1:(member-1), hh)  ID <- data.frame(no=1:length(id), id=id, m=m)  #切片の設定  p <- c(1, rep(0, member-1))  Pop <- matrix(p, nrow=hh\*member, ncol=member-1, byrow=T)  Pop <- subset(Pop, rowSums(Pop) > 0)  #レベルと順位の説明変数の設定  LV <- scale(sf\_data1$lv)  SCORE <- -scale(sf\_data1$順位)  #ベクトル形式に変更  LV.v <- matrix(0, hh\*(member-1), ncol=member-1)  SCORE.v <- matrix(0, hh\*(member-1), ncol=member-1)  for(i in 1:hh){  index.v <- ((i-1)\*(member-1)+1):((i-1)\*(member-1)+member-1)  v.lv <- diag(LV[i, ], member-1)  v.score <- diag(SCORE[i, ], member-1)  LV.v[index.v, ] <- v.lv  SCORE.v[index.v, ] <- v.score  }  ##メンバー選択とレベルの関係を可視化  index.muse1 <- subset(1:nrow(rank), rank[, 1] %in% c("穂乃果", "ことり", "海未", "凛", "花陽"))  index.muse2 <- subset(1:nrow(rank), rank[, 1] %in% c("真姫", "にこ", "希", "絵里"))  index.aqours <- subset(1:nrow(rank), rank[, 1] %in% c("千歌", "曜", "梨子", "花丸", "善子", "aqours", "モブ"))  boxplot(sf\_data1$lv[index.muse1] ~ rank[index.muse1, 1], ylim=c(140, 1025))  boxplot(sf\_data1$lv[index.muse2] ~ rank[index.muse2, 1])  boxplot(sf\_data1$lv[index.aqours] ~ rank[index.aqours, 1])  ##データを結合  X <- data.frame(pop=Pop, lv=LV.v)  XM <- as.matrix(X)  ####マルコフ連鎖モンテカルロ法でランクプロビットモデルを推定####  ####MCMC推定のための推定準備####  ##切断正規分布の乱数を発生させる関数  rtnorm <- function(mu, sigma, a, b){  FA <- pnorm(a, mu, sigma)  FB <- pnorm(b, mu, sigma)  return(qnorm(runif(length(mu))\*(FB-FA)+FA, mu, sigma))  }  ##多変量正規分布の条件付き期待値と分散を計算する関数  cdMVN <- function(mu, Cov, dependent, U){    #分散共分散行列のブロック行列を定義  Cov11 <- Cov[dependent, dependent]  Cov12 <- Cov[dependent, -dependent]  Cov21 <- Cov[-dependent, dependent]  Cov22 <- Cov[-dependent, -dependent]    #条件付き分散と条件付き平均を計算  CDinv <- Cov12 %\*% solve(Cov22)  CDmu <- mu[, dependent] + t(CDinv %\*% t(U[, -dependent] - mu[, -dependent])) #条件付き平均を計算  CDvar <- Cov11 - Cov12 %\*% solve(Cov22) %\*% Cov21 #条件付き分散を計算  val <- list(CDmu=CDmu, CDvar=CDvar)  return(val)  }  ##アルゴリズムの設定  R <- 40000  sbeta <- 1.5  keep <- 4  factors <- 3  llike <- array(0, dim=c(R/keep)) #対数尤度の保存用  ##データの設定  #説明変数を多次元配列化  X.array <- array(0, dim=c(member-1, ncol(XM), hh))  for(i in 1:hh){  X.array[, , i] <- XM[ID[, 2]==i, ]  }  YX.array <- array(0, dim=c(member-1, ncol(XM)+1, hh))  #IDの設定  id\_r <- matrix(1:nrow(XM), nrow=hh, ncol=member-1, byrow=T)  ##推定プロセスの格納配列  UM <- matrix(0, nrow=hh, ncol=member-1)  util.M <- matrix(0, nrow=hh, ncol=member-1)  ##事前分布の設定  nu <- member #逆ウィシャート分布の自由度  V <- solve(0.1\*diag(member-1)) #逆ウィシャート分布のパラメータ  Deltabar <- rep(0, ncol(XM)) #回帰係数の平均の事前分布  Adelta <- solve(100 \* diag(rep(1, ncol(XM)))) #回帰係数の事前分布の分散  inv.facov <- solve(diag(100, factors))  alpha\_d <- 1  beta\_d <- 100  ##サンプリング結果の保存用配列  Util <- array(0, dim=c(hh, member-1, R/keep))  BETA <- matrix(0, nrow=R/keep, ncol=ncol(XM))  SIGMA <- matrix(0, nrow=R/keep, ncol=(member-1)^2)  FA.A <- matrix(0, nrow=R/keep, ncol=(member-1)\*factors)  FA.D <- matrix(0, nrow=R/keep, ncol=member-1)  FA.F <- array(0, dim=c(hh, factors, R/keep))  ##初期値の設定  #回帰係数の初期値  oldbeta <- c((table(Rank)/sum(table(Rank))\*10)[-member], runif(member-1, -0.3, 0.3))  #分散共分散行列の初期値  corM.f <- corrM(col=member-1, lower=0, upper=0) #相関行列を作成  Sigma.f <- covmatrix(col=member-1, corM=corM.f, lower=1, upper=1) #分散共分散行列  oldcov <- Sigma.f$covariance  #効用の平均構造の初期値  old.utilm <- matrix(XM %\*% oldbeta, nrow=hh, ncol=member-1, byrow=T)  #効用の初期値  old.util <- old.utilm + mvrnorm(nrow(old.utilm), rep(0, member-1), oldcov)  #因子負荷量と独自因子の初期値  A <- matrix(runif((member-1)\*factors, -1, 1), nrow=member-1, ncol=factors)  D <- diag(runif(member-1, 0, 0.5))  ####マルコフ連鎖モンテカルロ法で潜在変数ランクプロビットモデルを推定####  for(rp in 1:R){    ##順位選択結果と整合的な潜在効用を発生させる  #条件付き期待値と条件付き分散を計算  S <- rep(0, member-1)    for(j in 1:(member-1)){  MVR <- cdMVN(mu=old.utilm, Cov=oldcov, dependent=j, U=old.util)  UM[, j] <- MVR$CDmu #条件付き期待値を取り出す  S[j] <- sqrt(MVR$CDvar) #条件付き分散を取り出す    #潜在変数を発生させる  #切断領域の設定  rank.u <- t(apply(cbind(old.util[, -j], 0), 1, function(x) sort(x, decreasing=TRUE)))[, 1:3]  rank.u <- ifelse(Rank==member, 0, rank.u)    #切断正規分布より潜在変数を発生  old.util[, j] <- ifelse(Rank[, 1]==j, rtnorm(mu=UM[, j], S[j], a=rank.u[, 1], b=150),  ifelse(Rank[, 2]==j, rtnorm(mu=UM[, j], S[j], a=rank.u[, 2],  b=rank.u[, 1]),  ifelse(Rank[, 3]==j, rtnorm(mu=UM[, j], S[j], a=rank.u[, 3],  b=rank.u[, 2]),  rtnorm(mu=UM[, j], sigma=S[j], a=-150, b=rank.u[,  3]))))  }    util.v <- as.numeric(t(old.util)) #発生させた潜在効用をベクトルに置き換える    ##betaの分布のパラメータの計算とmcmcサンプリング  #z.vecとX.vecを結合して多次元配列に変更  YX.bind <- cbind(util.v, XM)  for(i in 1:hh){  YX.array[, , i] <- YX.bind[id\_r[i, ], ]  }    ##回帰モデルのギブスサンプリングでbetaとsigmaを推定  #betaのギブスサンプリング  invcov <- solve(oldcov)  xvx.vec <- rowSums(apply(X.array, 3, function(x) t(x) %\*% invcov %\*% x))  XVX <- matrix(xvx.vec, nrow=ncol(XM), ncol=ncol(XM), byrow=T)  XVY <- rowSums(apply(YX.array, 3, function(x) t(x[, -1]) %\*% invcov %\*% x[, 1]))    #betaの分布の分散共分散行列のパラメータ  inv\_XVX <- solve(XVX + Adelta)    #betaの分布の平均パラメータ  B <- inv\_XVX %\*% (XVY + Adelta %\*% Deltabar) #betaの平均  b1 <- as.numeric(B)    #多変量正規分布から回帰係数をサンプリング  oldbeta <- mvrnorm(1, b1, inv\_XVX)    ##Covの分布のパラメータの計算とmcmcサンプリング  #逆ウィシャート分布のパラメータを計算  R.error <- matrix(util.v - XM %\*% oldbeta, nrow=hh, ncol=member-1, byrow=T)  IW.R <- V + matrix(rowSums(apply(R.error, 1, function(x) x %\*% t(x))), nrow=member-1, ncol=member-1, byrow=T)    #逆ウィシャート分布の自由度を計算  Sn <- nu + hh    #逆ウィシャート分布からCovをサンプリング  Cov\_hat <- rwishart(Sn, solve(IW.R))$IW  oldcov <- cov2cor(Cov\_hat)    ##潜在効用とパラメータを更新  #潜在効用と潜在効用の平均を更新  old.utilm <- matrix(XM %\*% oldbeta, nrow=hh, ncol=member-1, byrow=T)  Z <- old.util - old.utilm  Z <- scale(Z)    ##潜在効用の誤差項から因子分析モデルを推定  #多変量正規分布から潜在変数f(共通因子)をサンプリング  ADA <- t(A) %\*% solve(A %\*% t(A) + D)  F\_mean <- Z %\*% t(ADA) #共通因子の平均  F\_var <- diag(factors) - ADA %\*% A #共通因子の分散共分散行列  Fi <- t(apply(F\_mean, 1, function(x) mvrnorm(1, x, F\_var))) #多変量正規分布から共通因子をサンプリング      #ガンマ分布から独自因子dをサンプリング  Z.error <- Z - Fi %\*% t(A)  Zv.R <- matrix(rowSums(apply(Z.error, 1, function(x) x %\*% t(x))), nrow=member-1, ncol=member-1)    gamma\_alpha <- (hh + alpha\_d)/2 #alphaを計算  gamma\_beta <- (diag(Zv.R) + beta\_d)/2 #betaを計算  D <- diag(rgamma(length(gamma\_beta), gamma\_alpha, gamma\_beta)) #ガンマ分布から独自因子をサンプリング    #多変量正規分布から因子負荷量Aをサンプリング  FF <- t(Fi) %\*% Fi  FZ <- t(Fi) %\*% Z  d\_sigma <- list()    for(i in 1:(member-1)){  d\_sigma[[i]] <- 1/diag(D)[i] \* inv.facov  A\_mu <- solve(d\_sigma[[i]] + FF) %\*% FZ[, i]  A\_cov <- solve(inv.facov + diag(D)[i]\*FF)  A[i, ] <- mvrnorm(1, A\_mu, A\_cov)  }      ##サンプリング結果を保存  if(rp%%keep==0){  print(rp)  mkeep <- rp/keep  Util[, , mkeep] <- old.util  BETA[mkeep, ] <- oldbeta  SIGMA[mkeep, ] <- as.numeric(oldcov)  FA.A[mkeep, ] <- as.numeric(A)  FA.D[mkeep, ] <- diag(D)  FA.F[, , mkeep] <- Fi  print(round(oldcov, 2))  print(round(oldbeta[1:member-1], 2))  print(round(oldbeta[member:length(oldbeta)], 2))  print(round(t(A), 2))  }  }  ####推定結果の要約と適合度の確認####  burnin <- 10000/keep #バーンイン期間  ##サンプリング結果を可視化  #回帰係数のプロット  matplot(BETA[, 1:4], type="l", main="メンバーごとの人気のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 5:9], type="l", main="メンバーごとの人気のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 10:11], type="l", main="回帰係数のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 12:15], type="l", main="回帰係数のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 16:20], type="l", main="回帰係数のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 21:24], type="l", main="回帰係数のサンプリング結果", ylab="パラメータ推定値")  matplot(BETA[, 25:29], type="l", main="回帰係数のサンプリング結果", ylab="パラメータ推定値")  #分散供分散行列の可視化  matplot(SIGMA[, 1:4], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 5:9], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 10:13], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 14:18], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 19:22], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 23:27], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 28:31], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 32:36], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 37:40], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 41:45], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 46:49], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 50:54], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 55:58], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 59:63], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 64:67], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 68:72], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 73:76], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(SIGMA[, 77:81], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  #因子負荷量の可視化  matplot(FA.A[, 1:4], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(FA.A[, 5:9], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  matplot(FA.A[, 10:13], type="l", main="分散共分散行列のサンプリング結果", ylab="パラメータ推定値")  ##効用の散布図散布図行列の作成  panel.hist <- function(x, ...){  usr <- par("usr"); on.exit(par(usr))  par(usr = c(usr[1:2], 0, 1.5) )  h <- hist(x, plot = FALSE)  breaks <- h$breaks; nB <- length(breaks)  y <- h$counts; y <- y/max(y)  rect(breaks[-nB], 0, breaks[-1], y, col = "grey", ...)  }  panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...){  usr <- par("usr"); on.exit(par(usr))  par(usr = c(0, 1, 0, 1))  r1 <- cor(x, y)  r2 <- abs(cor(x, y))  txt <- format(c(r1, 0.123456789), digits = digits)[1]  txt <- paste0(prefix, txt)  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)  text(0.5, 0.5, txt, cex = cex.cor \* r2)  }  i <- 8000  Util.Z <- Util[, , i] - matrix(XM %\*% BETA[i, ], nrow=hh, ncol=member-1, byrow=T)  colnames(Util.Z) <- lovelive[-member]  #変数1～5の散布図行列  pairs(as.data.frame(Util.Z[, 1:5]), panel=panel.smooth, bg="lightblue", diag.panel=panel.hist,  upper.panel=panel.cor)  #変数6～9の散布図行列  pairs(as.data.frame(Util.Z[, 6:9]), panel=panel.smooth, bg="lightblue", diag.panel=panel.hist,  upper.panel=panel.cor)  #変数10～12の散布図行列  pairs(as.data.frame(Util.Z[, 10:15]), panel=panel.smooth, bg="lightblue", diag.panel=panel.hist,  upper.panel=panel.cor)  ##推定値の事後平均の比較  #betaの要約統計量  #メンバーの切片の要約推定量  round(beta.m1 <- colMeans(BETA[burnin:nrow(BETA), 1:(member-1)]), 3) #betaの事後平均  round(beta.q1 <- apply(BETA[burnin:nrow(BETA), 1:(member-1)], 2, function(x) quantile(x, 0.05)), 2) #5％分位点  round(beta.q2 <- apply(BETA[burnin:nrow(BETA), 1:(member-1)], 2, function(x) quantile(x, 0.95)), 2) #95％分位点  round(apply(BETA[burnin:nrow(BETA), 1:(member-1)], 2, sd), 2) #事後標準偏差  beta.q <- cbind(beta.q1, beta.q2)  t(t(table(sf\_data$X1位)))  #人気度の信用区間の可視化  plot(beta.q[1, ], rep(9, 2), type="l", xlab="人気", ylab="メンバー", xlim=c(-0.3, 0.9), ylim=c(0, 10), lwd=2, main="μ’ｓの人気の事後信用区間")  lines(beta.q[2, ], rep(8, 2), type="l", lwd=2)  lines(beta.q[3, ], rep(7, 2), type="l", lwd=2)  lines(beta.q[4, ], rep(6, 2), type="l", lwd=2)  lines(beta.q[5, ], rep(5, 2), type="l", lwd=2)  lines(beta.q[6, ], rep(4, 2), type="l", lwd=2)  lines(beta.q[7, ], rep(3, 2), type="l", lwd=2)  lines(beta.q[8, ], rep(2, 2), type="l", lwd=2)  lines(beta.q[9, ], rep(1, 2), type="l", lwd=2)  #説明変数の要約推定量  round(colMeans(BETA[burnin:nrow(BETA), member:ncol(BETA)]), 3) #betaの事後平均  round(apply(BETA[burnin:nrow(BETA), member:ncol(BETA)], 2, function(x) quantile(x, 0.05)), 2) #5％分位点  round(apply(BETA[burnin:nrow(BETA), member:ncol(BETA)], 2, function(x) quantile(x, 0.95)), 2) #95％分位点  round(apply(BETA[burnin:nrow(BETA), member:ncol(BETA)], 2, sd), 2) #事後標準偏差  #sigmaの要約統計量  round(cor.m <- matrix(colMeans(SIGMA[burnin:nrow(SIGMA), ]), nrow=member-1, ncol=member-1), 2) #sigmaの事後平均  round(matrix(apply(SIGMA[burnin:nrow(SIGMA), ], 2, function(x) quantile(x, 0.05)), nrow=member-1, ncol=member-1), 2) #5％分位点  round(matrix(apply(SIGMA[burnin:nrow(SIGMA), ], 2, function(x) quantile(x, 0.95)), nrow=member-1, ncol=member-1), 2) #95％分位点  round(matrix(apply(SIGMA[burnin:nrow(SIGMA), ], 2, sd), nrow=member-1, ncol=member-1), 2) #事後標準偏差  ####因子分析モデルで因子構造を推定####  ##因子数を決定  i <- 20000/keep  plot(1:(member-1), eigen(cor.m)$values,  type="l", ylab="固有値", xlab="変数数")  factors <- 5 #因子数は5  ##MCMCサンプリング結果を用いて因子分析モデルを当てはめる  res <- fa(r=cor.m, nfactors=5, fm="ml", rotate="promax", cor="cor", n.obs=hh)  res  ####相関行列のシミュレーション####  s <- 300000  lovelive <- c("穂乃果", "ことり", "海未", "凛", "花陽", "真姫", "にこ", "希", "絵里", "千歌", "曜", "梨子",  "花丸", "善子", "aqours")  U.simul <- mvrnorm(s, rep(0, member-1), cor.m)  colnames(U.simul) <- lovelive  simul.max <- apply(U.simul, 1, which.max)  simul.res <- table(simul.max)  names(simul.res) <- lovelive  t(t(simul.res/s)) #結果の集計  ####相関行列を可視化####  cor.mus <- cor.m[1:9, 1:9]  colnames(cor.mus) <- lovelive[1:9]  qgraph(cor.mus, edge.labels=T, minimum=.2, layout="circle", palette="pastel") |