Groceries に関する R 言語を使用した頻出パターン抽出及び相関ルール分析

文理学部情報科学科
5419045 高林 秀

2021 年 7 月 26 日

概要

本稿では、今年度データ科学 2 で学習した「頻出パターン抽出」及び「相関ルール分析」の手法を使用して、R 言語のライブラリである arules に付属しているデータ Groceries を対象とした頻出パターン抽出、相関ルール分析を行うものである。

1 目的

本稿では実際に、R 言語を使用しライブラリ arules 付属のデータである Groceries の頻出パターン抽出、相関ルール分析を行うことで、本年度データ科学 2 で学習した頻出パターン、相関ルール分析の手法への理解を深め、その定着を図ることを目的とする。また、1 年次に学習した latex を用いた PDF 作成の復習も兼ねるものである。

2 理論説明

今回の実験で用いた、計算理論をそれぞれ説明する。

2.1 バスケット分析

初めに、頻出パターン抽出、相関ルール分析を説明する前に「バスケット分析」について説明する。

バスケット分析とは、データマイニングにおける代表的な手法の 1 つで、「顧客の購買記録をデータ化し分析を行うことで、顧客に共通するルールや傾向を導く」データ分析のことである。すなわち「一緒に買われやすい商品の組み合わせを見つける」ということである。

顧客の買い物データを分析しその結果を企業の販促活動などのマーケティングに関わる施策に適用するのが目的である。なお、バスケット分析はアソシエーション分析[*1]の一つとされ、マーケットバスケット分析とも呼ばれる。

[*1] データマイニングにおけるデータ間の関連性を見つける手法のこと。「もし A ならば B である」といった法則を見つけ出し、主に購買記録などから顧客の購買行動の関連性を見つけ出すのに利用される。

2.2　頻出パターン・相関ルール分析の概要

身の回りで頻出パターン抽出と相関ルール分析が使われている代表例としては、先に述べたバスケット分析をはじめ、オンラインショッピングサイト等のレコメンドシステム等が挙げられるだろう。頻出パターン抽出と相関ルール分析はこれらのシステムの基本的な原理である。この章では、頻出パターン抽出と相関ルール分析とはなにか概要を説明する。

2.2.1　頻出パターン

まず、頻出パターン (頻出アイテム集合) とはなにか説明する。頻出パターンとは「データベース中に高頻度で現れる組み合わせ、集合のこと」であり、頻出パターン抽出 (頻出パターンマイニング) とはその集合を発見するための手法である。またこの集合のことを頻出アイテム集合と呼ぶ。頻出アイテム集合か、そうでないかを判断するための基準として後述する支持度 (同時確率) と呼ばれる数値を計算し、その数値が、あらかじめ設定した閾値を超えるかどうかで判定する。頻出パターン抽出は「どの商品が一緒に購入されているか」を見るので、得られた結果は、商品の陳列場所の改善や、販促キャンペーン、店舗レイアウト等を考える際に利用することができる。

頻出パターン抽出は、後述する相関ルール問題の部分問題として広く認知されている。

2.2.2　相関ルール

次に、相関ルールとはなにか説明する。相関ルールとはアソシエーション・ルールとも呼ばれ、「頻出パターン間の関係性」のことを示す。相関ルール分析・抽出はこの関係性すなわちルールを見つける目的で行われる。例えば、「あるアイテム集合 $I1$ が生起するとき、別のアイテム集合である $I2$ も同時に生起する」といったようなものが相関ルールとなる。このとき、記号で「$\{I1\} \Rightarrow \{I2\}$」といった形で記述する。導いた相関ルールを評価する評価基準として、後述する確信度と呼ばれるものが存在する。具体的な計算法は後述するが、確信度とは一言で言えば「ルールの強さ」を示す指標で、左辺のアイテム集合が生起したときの右辺のアイテム集合の生起確率である。。加えて、支持度も利用される。相関ルールにおける評価指標としての支持度は「ルールの汎用性」を示すものとして利用される。

相関ルール抽出問題とは、あらかじめ設定する「最小支持度」「最小確信度」を閾値として、この閾値を超える相関ルールをデータベース上から見つけることを目的とした問題である。

2.2.3　トランザクションデータベース

頻出パターン・相関ルール分析は後述するように、形式的な定義のもとで、入力をうけその出力として頻出パターン・相関ルールを返す。このとき、入力として「トランザクションデータベース」が与えられる。

トランザクション (英名：transaction) とは、商取引、議事録、売買等の意味があり、情報処理用語としては一連の処理をひとつにまとめたものという意味をもつ。トランザクションデータベースとは、データの更新処理を一つにまとめているようなデータベースのことである。

2.3 計算法

この章では、実際に頻出パターンや相関ルールがどのように計算されているのかについて説明する。その前に、概要の部分で登場した「支持度」と「確信度 (信頼度)」について説明する。以下の説明で使用する数式記号について、

$$D = t_1, t_2, t_3, ..., t_n : n \text{ 個のトランザクションを含むデータベース } D$$
$$I = \cup_{t_i \in D} t_i : \text{全アイテムの集合}$$
$$t_i \subseteq I : i \text{ 番目のトランザクション}$$
$$min\_sup(0 < min_sup \leq 1) : \text{最小支持度}$$
$$min\_conf(0 \leq min_conf \leq 1) : \text{最小確信度}$$

のように定義する。

2.3.1 支持度 (support)

支持度とは「ルールの汎用性 (一般性) の尺度」であり、集合 $I1, I2$ を例にしたベン図で示すと以下のようになる。これは、同時確率とみなすことができる。つまり、あるパターン X の支持度とは X 中のアイテムが同時に出現する確率ということができる。これを式で示すと以下のようになる。

※ $|a|$: 集合 a の要素数

$$sup_D(X) = \frac{|t \in D | X \subseteq t|}{|D|}$$

- $t \in D$:アイテム集合 X を含むデータベース中のトランザクション

つまり、アイテム集合 X の支持度は、X を含むトランザクションの割合、すなわち X 中のアイテムがすべて出現するときの確率という意味で、その値は「X を含むデータベー上のトランザクション」を「データベース全体のトランザクション数」で除算した値である。

支持度が低いとは、そのパターンがごく少数の事例にのみ関係するパターンであり、いかに特徴的、すなわちその集合が他の部分集合を包含していて、少ないトランザクションに出現するということであり、データベースの一般的な傾向とはみなされていない、ということになる。反対に、支持度が高いとは、そのパターンがデータベース内で一般的、すなわち他の部分集合に包含されていて、多くのトランザクションに出現している、ということを意味している。

2.3.2 確信度 (confidence)

確信度とは、「そのルールの確からしさの尺度」であり、データマイニングにおける相関ルールの重要度を示す指標である。別名、信頼度とも呼ばれる。

前章の部分でも述べたが、あるアイテム集合 $I1$ が生起するとき同時に $I2$ も生起するという現象、ルールは $I1 \Rightarrow I2$ で表記される。確信度とは、$I1 \Rightarrow I2$ のルールの強さを示す指標と言える。$I1 \Rightarrow I2$ の確信度を

$conf_D(I1, I2)$ と示すと、確信度は以下のように計算される。

$$conf_D(I1, I2) = \frac{|\{t \in D | I1 \subseteq t, I2 \subseteq t|\}}{\{|t \in D | I1 \subseteq t|\}}$$
$$= \frac{|\{t \in D | (I1 \cup I2) \subseteq t\}|}{|\{t \in D | I1 \subseteq t\}|}$$

このとき、分母の式 $|\{t \in D | I1 \subseteq t\}|$ は $I1$ の出現回数を示している。また分子の式 $|\{t \in D | (I1 \cup I2) \subseteq t\}|$ は、$I1, I2$ の同時出現回数を示している。つまり確信度とは、$\frac{I1 \text{ の出現回数}}{I1 \text{ と } I2 \text{ が同時に出現する回数}}$ の値ということになり、これは条件付き確率と同じになる。

確信度が低いときとは、そのルールが不正確であることを示している。反対に確信度が高いときとは、そのルールが正確なルールであることを示していることになる。

2.3.3 頻出パターン抽出の計算法

頻出パターン抽出の際は、入力としてトランザクションデータベースを受け取り、すべての頻出アイテム集合を出力する。このとき、頻出パターン F は次の式で示すことができる。

$$F = \{X | X \subseteq I, X \neq \phi, sup_D(X) \geq min\_sup\}$$

■頻出パターン抽出の計算量　上記式で示すとおり、頻出パターンに選ばれた集合 X の支持度は、予め定めた支持度の下限値すなわち最小支持度以上である必要がある。ということは、頻出パターンを見つけるには単純に、すべてのアイテム集合 $ $ I に属する、すべての部分集合の支持度を 1 つずつ計算していき、その値が最小支持度を超えるかどうか判定すれば良いことになる。

しかし、現実の場合ではそうもいかない。ご存知の通り、我々が普段使うコンピュータの計算資源は無限ではなく有限である。したがって、あまりにも計算量が大きすぎる問題に関してはそもそも計算リソースが足らず、答えを求めることができない。

現実の現場 (小売店やその他店舗) 等で使用されるデータベースは膨大な数のデータを扱うことがほとんどである。そうなれば当然すべてのアイテム集合 $ $ I の数も膨大である。以下に、すべてのアイテム集合 I の数を $|I|$ としたときの冪集合[*2]の個数を示す。

$$\text{冪集合の個数 } N = 2^{|I|} - 1$$

表 1　$|I|$ の値による冪集合の個数

$\|I\|$	冪集合の個数
$\|I\| = 10$	1024
$\|I\| = 16$	65536
$\|I\| = 50$	約 1.1 京 ※ 1 京 $= 10^{16}$
$\|I\| = 100$	約 1267 穣 ※ 1 穣 $= 10^{28}$
$\|I\| = 200$	約 1.6 那由他 ※ 1 那由他 $= 10^{60}$, 一説では 10^{72}

[*2] ある集合の部分集合全体の集合。単にべき集合とも記される。

上記の表からも分かるとおり、現実の現場ではすべての冪集合を求めることは不可能に近い。したがってより効率的に頻出パターンを計算するアルゴリズム・手法が必要となる。そのような代表例として、後述する「バックトラック法」や「アプリオリアルゴリズム」が存在する。1

2.3.4 パターン空間について

パターン空間とは、「考えうるすべてのアイテム集合を列挙したもの」である。すべてのアイテム集合 I の部分集合、すなわち頻出パターンの候補を列挙し、その部分集合 P, Q に対し $P \subseteq Qand|P| = |Q| - 1$ のとき線で結ぶ、という動作をする。このような操作で集合を図示、列挙すると「束 (Lattice)」と呼ばれる、任意の 2 点間に上限と下限の存在する (半) 順序集合ができあがる。すなわち、要素全てに共通する上限と下限が存在することを意味する。アイテム集合 I の冪集合を対象にすると、下図に示すとおり上限は空集合 $\{\phi\}$、下限は I となる。

■支持度の逆単調性 アイテム集合 P とその部分集合 Q、$P \subseteq Q$ であるとき、P の支持度はその部分集合 Q の支持度以上になることを「支持度の逆単調性」と呼ぶ。

$$sup_D(P) \geq sup_D(Q)$$

このことから次のことが導き出せる。

1. $P \subseteq Q$ であるとき $sup_D(P) \geq sup_D(Q)$
2. P の支持度が最小支持度未満であるとき、P のすべての上位集合の支持度は最小支持度未満になる。
3. 上記より、最小支持度未満となる集合の上位集合は必ず頻出パターンになることはない。よって計算する必要がなくなる。

$$sup_D(P) < min\_sup \rightarrow \forall Q_{\supseteq P}[sup_D(Q) < min\_sup]$$
$$\forall Q_{\supseteq P} : Q \text{ は } P \text{ の上位集合}$$

また、この性質は「アプリオリ特性」とも呼ばれている。頻出パターンの発見は、言い換えると考えうるパターン空間から、最小支持度以上を満たす頻出アイテム集合を探すということになる。ただし先に述べたように、すべてを探索し切るのは困難なので前述した「支持度に関する逆単調性」を利用して探索範囲を限定 (枝刈り) することが求められる。

■（補足）深さ優先探索・幅優先探索 木構造をとるデータ構造におけるデータの探索手法として深さ優先探索 (depth-first search) と幅優先探索 (breath-first search) が存在する。

深さ優先探索とは下図に示すとおり、探索の順序を「木の深さ (レベル) を大きくするように」探索する手法で、進めるところまで進んでこれ以上進めなくなったら一度上の深さまで戻ってまた探索をする、といったような動作をする。

反対に幅優先探索は、探索の順序を「同じ深さに属するデータから順番に見るように」探索する手法で、探索の出発点から横に近い順番で探索をする、といったような動作をする。

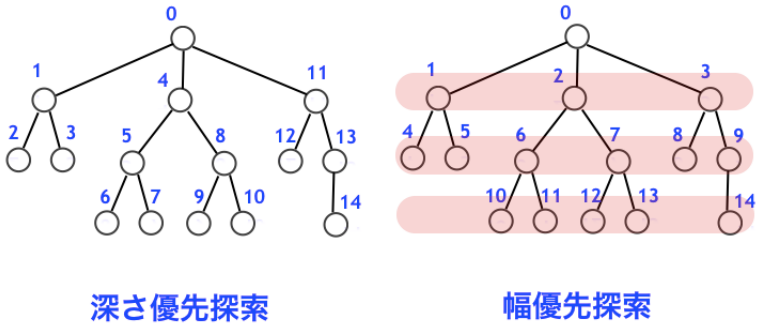

図 1　深さ優先探索と幅優先探索

出典：https://qiita.com/drken/items/4a7869c5e304883f539b

表 2　深さ優先探索と幅優先探索の相違点

深さ優先探索	幅優先探索
探索開始点から近い順に深いほうへ探索	探索開始点から近い順に同じ深さのデータを探索
一般にメモリへの負担が小さい	一般にメモリの使用量は多め
一般に答えまでの所要時間が不定	一般に答えにたどり着くまでの時間は短い

2.3.5　バックトラック法

前述したように、パターン空間から考えうるすべての冪集合の支持度を計算し頻出アイテム集合を決定するのは計算量と効率の面から非現実的である。したがって、より効率の良い計算方法のとして「バックトラック法」と「アプリオリアルゴリズム」を先に例示した。ここではバックトラック法についての具体的な説明を行う。

まず、バックトラック法とはなにか。バックトラック法とは「考えうるすべてパターンを系統的に探索し答えを得る」手法で、探索時の頻出アイテム集合の候補数をできるだけ少なくすることで探索時の効率を上げるというものである。バックトラック法は支持度の逆単調性を利用して探索時の頻出アイテム集合の候補数を限定している。そのため、すべてのアイテム集合を探索しないとはいえ頻出アイテム集合を逃すことはない。

バックトラック法は、パターン空間内の集合に「親」を設定することによってグラフから木構造へ変形する。このとき、親とは包含するアイテム集合（そのアイテム集合のパターン空間における 1 つ下にあるアイテム集合）から「最大のアイテムを削除した集合」ということなる。また、親の子は親の集合を得る逆の操作をすれば良いので、その集合の要素より大きなアイテムを 1 つ追加した集合となる。

この親子関係が結ばれる集合同士を線で結ぶとき、あるアイテム集合の親は必ず 1 つに定まる。したがって、そのアイテム集合は自身の親からのみ探索することができるので入力が 1 つに決まる。これを図示すると下図のようになり、木構造が出来上がる。このときの木構造を「集合列挙木」と呼ぶ。

上図からも分かるとおり、各子集合は親集合からのみ探索することができるので探索時に重複することはない。バックトラック法はこの集合列挙木を作成しながら、アイテム集合の支持度を計算し、最小支持度以下ならば、支持度の逆単調性より、それより深いアイテム集合の探索を打ち切る、すなわち枝刈りを行うことで探索の効率を高めている。このとき、あるアイテム集合の支持度が最小支持度未満であるときそれより先のアイ

テム集合の探索を打ち切り、親に戻って (バックトラック) また探索をするのでバックトラック法は、深さ優先探索であると言うことができる。

以下にバックトラック法の擬似的な python コードを示す。

$def \quad backtrack(D, min\_sup):$
$\quad I = \cup_{t \in D} t \quad$ #全アイテム集合を取得し I に代入
$\quad dfs(\phi, I, D, min\_sup) \quad$ #深さ優先探索 $(depth - first - search)$

$def \quad dfs(P, I, D, min\_sup):$
$\quad for \quad i \quad in \quad I:$
$\quad\quad Q.append(P \cup i)$#子アイテム集合 Q にアイテム i を追加
$\quad\quad if \quad sup_D(Q) \geq min\_sup:$
$\quad\quad\quad print(Q) \quad$ #Q を出力
$\quad\quad\quad dfs(Q, I, D, min\_sup) \quad$ #再帰呼び出し
$\quad\quad else:$
$\quad\quad\quad pass$

2.3.6 アプリオリアルゴリズム

次に挙げられる方法として、「アプリオリアルゴリズム」が存在する。先述したアプリオリ特性を利用しているアルゴリズムなので、アプリオリアルゴリズムと呼ばれる。アプリオリアルゴリズムはバックトラック法とは異なり、支持度の逆単調性を利用した幅優先探索である。バックトラック法では、親すなわち一つの部分集合の支持度のみを計算していたが、アプリオリアルゴリズムではすべての部分集合に対して支持度を計算する。したがって、探索の仕方が階層的、横に進むようになることから幅優先探索と言える。

アプリオリアルゴリズムには大きく分けて以下の 2 ステップがある。

- ジョインステップ (Join Step)
- プルーンステップ (Prune Step)

このアルゴリズムでは、頻出アイテム集合を、集合の要素数の大きい方から順番に 1 $ K まで求めていく。サイズ K の頻出アイテム集合 (以下 $K - itemset$) を求めるため 1 つ要素数が大きい $K + 1 - itemset$ を利用する。

■ジョインステップ JoinStep ジョインステップは、結合ステップとも呼ばれる。

このステップでは、2 つの $K - itemset$ を利用して $K + 1 - itemset$ の候補を生成する。

C_K : サイズ K の候補集合
L_K : サイズ K の頻出アイテム集合

とするとき、ジョインステップでは下記の計算が行われる。

$$C_{K+1}^{join} = \{X \cup Y \mid \begin{array}{l} X, Y \in L_K \\ X \setminus \{tail(X)\} = Y \setminus \{tail(Y)\}, \\ tail(X) < tail(Y) \end{array}$$

平たく言えば、すべての頻出アイテム集合を抽出する工程がジョインステップであり、$K = 1, 2, 3, ...$ アイテム集合に対して頻出か否か調べる。このとき、K を増やしていくときの枝刈り時には、アプリオリ特性を利用し、探索の必要がない集合の事前削除を行っている。

ソート済みの $K - itemset$ から、一番右 (昇順ソート時の最も大きい要素) のだけが異なる $k - itemset$ のペアを見つけそのペアから $k + 1 - itemset$ 候補を生成する。例えば、$4 - itemset$ である{□, □, □, ○}と{□, □, □, △}の集合から、$5 - itemset$ である{□, □, □, ○, △}を生成するといった感じである。この時、生成した候補集合が頻出アイテム集合となるためには、候補集合の親である{□, □, □, ○}と{□, □, □, △}がともに頻出アイテム集合でなければならない。この時点で、アプリオリ特性から頻出アイテム集合となることができない $K + 1 - itemset$ は候補集合とは見なさない。より厳密に言えば、あるアイテム集合 $Q(X \cup Y)$ が頻出であるためには、その両親となる以下の 2 集合が頻出アイテム集合となる必要がある。

1. Q から最大要素を除いたアイテム集合 X
2. Q から 2 番目に大きい要素を取り除いた集合 Y

また、上記の 2 集合が頻出であれば、それらは $L_{\{|Q|-1\}}$ に含まれる。

■プルーンステップ PuruneStep　ジョインステップで生成される候補から、頻出となりえないアイテム集合を削除する工程がプルーンステップである。プルーンステップでは、$K + 1$ アイテム集合候補の K 要素、すなわちその $K + 1$ アイテム集合の親集合の要素の各部分集合が頻出であるか否かを計算する。

$$C_{K+1} = Q \in C_{K+1}^{join} | P \subset Q || P | = K \subseteq L_K$$

このとき、ジョインステップの出力から要素を選別しそれをアイテム集合 Q とする。ジョインステップでその集合が頻出アイテム集合であるか否かはわかっているので、プルーンステップで生成される、サイズ K の Q の部分集合も頻出アイテム集合となる。

ここまでの説明をまとめると、アプリオリアルゴリズムは以下のような手順となる。

1. ジョインステップ
   (a) 2 つの $k - itemset$ を利用し $K + 1 - pitemset$ の候補集合を生成する。
2. プルーンステップ
   (a) ジョインステップの結果である各候補集合を対象に、サイズ K である部分集合が頻出アイテム集合か否か計算する。
3. プルーンステップの結果である各頻出アイテム集合候補の支持度を計算し、min_sup 最小支持度を満たす集合候補のみが $(K + 1) - itemset$ となる。

以上がアプリオリアルゴリズムの概要である。次は、この頻出パターン抽出によって得られた結果を元に、相関ルールを導出する方法について説明する。

2.3.7　相関ルール抽出の計算法

頻出アイテム集合の抽出を終えた次に行われるのが相関ルール抽出である。先に述べたように{I1}が起こる時{I2}も同時に起こるといったルール・法則は $\{I1\} \Rightarrow \{I2\}$ という記述方式で示される。この章では、得られた頻出アイテム集合から上記の様な相関ルールを抽出する計算法を説明する。

相関ルール抽出の主たる考えは「1 つの頻出アイテム集合を排他的な 2 つの集合に分解することによってルールを生成し、その各ルールの確信度を計算する」である。例えば、ある頻出アイテム集合 $\{a, b, c\}$ を以下の様な各集合に分解するパターンをを考えてほしい。

$$\{a\}, \{b, c\} \tag{1}$$
$$\{b\}, \{a, c\} \tag{2}$$
$$\{c\}, \{a, b\} \tag{3}$$
$$\tag{4}$$

このとき、各組み合わせ (1),(2),(3) からは次の様なルールが考えられる。

$$(1) : \{a\} \Rightarrow \{b, c\} \text{ と } \{b, c\} \Rightarrow \{a\}$$
$$(2) : \{b\} \Rightarrow \{a, c\} \text{ と } \{a, c\} \Rightarrow \{b\}$$
$$(3) : \{c\} \Rightarrow \{a, b\} \text{ と } \{a, b\} \Rightarrow \{c\}$$

このように導かれた各ルールについて確信度を計算し、予め設定している、最小確信度を超えるか否かで相関ルール抽出を行う。

このとき、支持度を計算せずよいのかと思われるが、既出の通り、「相関ルールの支持度＝頻出 K アイテム集合の支持度」であるので、すべての相関ルールは最小支持度を満たしている。したがって、相関ルール抽出において計算するのは確信度のみとなる。

より具体的な式は次に示す通り。前述している各文字の定義をこの章にも記す。

$$D = t_1, t_2, t_3, ..., t_n : n \text{ 個のトランザクションを含むデータベース } D$$
$$I = \cup_{t_i \in D} t_i : \text{全アイテムの集合}$$
$$t_i \subseteq I : i \text{ 番目のトランザクション}$$
$$min\_sup(0 < min_sup \leq 1) : \text{最小支持度}$$
$$min\_conf(0 \leq min_conf \leq 1) : \text{最小確信度}$$

このとき、

$$\text{相関ルール } R = \{X \Rightarrow Y\}$$
$$X \Rightarrow Y | \begin{cases} X \subseteq I, X \neq \phi, Y \subseteq I, Y \neq \phi \\ X \cap Y = \phi \\ sup_D(X \cup Y) \geq min\_sup \quad \text{※ } X, Y \text{ 両方がもつトランザクションの割合} \\ conf_D(X, Y) \geq min\_conf \quad \text{※ルール「} X \Rightarrow Y \text{」の確信度} \end{cases}$$

確信度 $conf_D(X, Y)$ は、

$$conf_D(X, Y) = \frac{|\{t \in D | X \subseteq t, Y \subseteq t\}|}{|\{t \in D | X \subseteq t\}|} = \frac{|\{t \in D | (X \cup Y) \subseteq t\}|}{|\{t \in D | X \subseteq t\}|}$$

頻出アイテム集合の要素数が多いとき、考えられる相関ルールの組み合わせも非常に多くなるため組み合わせ爆発[*3]が発生する可能性がある。考えられる相関ルールの総数は $\Sigma_{k=2}^{|I|} \frac{|I|}{K} \times (2^K - 2)$ となるため、アイテム集合の要素が多いと相関ルールの数も大きくなるのが分かるであろう。したがって、事前に確信度を計算することなく、相関ルールから除外する必要が出てくる。ここで、頻出パターン抽出で支持度の逆単調性の利用と同様に「確信度の逆単調性」を利用する。

■確信度の逆単調性

$$\forall Z \subset X[conf_D(X,Y) \geq conf_D(X \setminus Z, Y \cup Z)]$$
$$conf_D < min\_conf \rightarrow \forall Z \subset X[conf_D(X \setminus Z, Y \cup Z)]$$

つまり、X, Y の確信度が最小確信度未満であるとき、そのすべての部分集合 Z の確信度も最小確信度未満である、という性質である。

なお、アプリオリ特性から $\Rightarrow$ の方向へ進む、すなわち右辺に要素が加わると、ルールの確信度は減少する。此の性質からもルールの枝刈りを行うことができる。

■相関ルールの導出　ある頻出アイテム集合 Q を分割することで得ることので来る相関ルール $X \Rightarrow Y$ を考える。相関ルール抽出には、頻出パターン抽出と同じように、考えられるすべての相関ルールを列挙し、包含関係にあるルールを線で結んだパターン空間を考える。このとき、$X \Rightarrow Y$ である相関ルールに関して、帰結部 (Y) に対し $Y \subseteq Y' and |Y| = |Y'| - 1$ のとき線で結ぶ。また、相関ルール抽出では、頻出パターン抽出と同様にバックトラック法やアプリオリアルゴリズムを使用して、パターン空間の探索を行う。

例えば頻出アイテム集合 a, b, c, d が与えられているとき、第 1 段階で考えられる相関ルールは以下の 4 つになる。

$$\{a\} \Rightarrow b, c, d(1)$$
$$\{b\} \Rightarrow a, c, d(2)$$
$$\{d\} \Rightarrow a, b, c(3)$$
$$\{c\} \Rightarrow a, b, d(4)$$

このとき、各ルールの確信度を計算したとして、ルール (1) が最小確信度未満である場合、そのルールとそのルールの先で考えられる新たなルール (そのルールの先にある全部分集合) は相関ルールから除外され、以降確信度の計算は行われない。

このようにして、実際には確信度を計算しながらパターン空間を生成し最終的な相関ルールの抽出を行っている。なお、バックトラック法とアプリオリアルゴリズムの詳しい説明に関しては頻出パターン抽出のときと同じなので、ご一読いただきたい。

2.3.8　相関ルール分析の評価基準

もとめた相関ルールの良し悪しを評価する基準が存在する。代表的な例はこれまで示した、支持度と確信度であるがそれ以外にも Lift 値や confrict 値なるものが存在する

[*3]

■支持度と確信度の弱点　相関ルールの評価において、単に支持度と確信度のみでそのルールの良し悪しを判断することは危険である。以下の例に示すように、統計的な数値、すなわち相関係数やそのルールに合致する事例の総数など、が求めた確信度や支持度で比較した場合とで逆転してしまうことがあり得るからだ。例えば次のような例を考える。ある変数 X, Y, Z の値がそれぞれ以下のように与えられているとする。

表 3　X, Y, Z の数値

X	1	1	1	1	0	0	0	0
Y	1	1	0	0	0	0	0	0
Z	0	1	1	1	1	1	1	1

このとき、次のことが分かる。

- X, Y の一致数：6/8 個、相関係数；0.58
- X, Z の一致数：3/8 個、相関係数；-0.38

このときの相関ルールが次のように導出されたとする。

1. $X = 1 \Rightarrow Y = 1$:支持度 25%, 確信度 50%
2. $X = 1 \Rightarrow Z = 1$:支持度 37.5%, 確信度 75%

このルールでは、2 のルールのほうが 1 よりも支持度も確信度も高いので、2 のルールは 1 のルールより強く良いように見えてしまう。しかし、表では 1 のほうが 6 個も一致しており 2 よりも一致している数が多い上、相関係数は (1) のほうが性の値を取っている。むしろ (2) の相関係数は負の値を取っていることから、X と Z の性質は得られたルール $X = 1 \Rightarrow Z = 1$ とは正反対である。このように、単に支持度と確信度が大きいからと言って、そのルールの良し悪しを決めることはよろしくない。

ここまでの話をまとめると、支持度と確信度が高い場合でも意味のないルールを導いていしまう恐れがある、ということだ。

■Lift 値　上記に示した弱点を補うために、相関ルール評価のための指標の 1 つとして Lift 値が存在する。これは、X と Y が独立であると仮定した場合の比となる。具体的な式は下記の通り。

※ $P(A) : A$ の生起確率

$$lift(X \Rightarrow Y) = \frac{P(X \cup Y)}{P(X) \times P(Y)} = \frac{P(Y|X) \times P(X)}{P(X) \times P(Y)} = \frac{P(Y|X)}{P(Y)}$$

このとき、分母である $P(Y)$ は Y の支持度である。また分子はルール $X \Rightarrow Y$ の確信度である。このことから、Lift 値は $\frac{X \Rightarrow Y \text{ の確信度}}{Y \text{ の支持度}}$ と言える。ただし、このときの分母は Y の事前確率によって正規化されなければならない。

Lift 値は、ルール「$X \Rightarrow Y$」における Y の支持度が大きい時に、計算上そのルール自体の確信度が大きくなってしまう問題を、Y の生起確率を考慮することによって防いでいる。ルールの帰結部 Y の支持度が大きいなら、Y の支持度で除算してしまおうというシンプルな考えである。ただし、Lift 値には大きな欠点があ

る。それは「ルールの方向」を無視している点にある。つまり、2 つのルール $X \Rightarrow Y$ と $Y \Rightarrow X$ のリフト値が全く同じになるということである。

Lift 値は、X, Y を独立と仮定したときの比であるので、値が 1 を越えるか否かでそのルールの良し悪しを判定することができる。例えば、$lift(X \Rightarrow Y) > 1$ のとき、正の相関があり、$lift(X \Rightarrow Y) < 1$ のとき、負の相関があると言える。

■conviction 値　もう一つの評価基準として conviction 値が存在する。conviction 値では Lift 値の弱点である「ルールの方向を失っている」という点を補うような評価基準である。具体的な計算式は下記に示すとおり。

$$conv(X \Rightarrow Y) = \frac{P(X)P(\neg Y)}{P(X) \cup \neg Y} = \frac{1 - sup(Y)}{1 - conf(X \Rightarrow Y)}$$

中間式の分子は、X が生起する事と Y が生起しない事を独立と見なしているときの確率で、分子は X が生起する確率と Y が生起しない確率の同時確率を意味する。また、最終部分の分母 $1 - conf(X \Rightarrow Y)$ から Lift 値とは異なりルールの方向性が残っていることが分かる。

3　計算機実験

3.1　実験準備

3.1.1　実験環境

今回の実験は仮想マシン上で R 言語を起動し行った。下記に実験時の環境を示す。

- ホスト OS：Window10 Home Ver.20H2
- 仮想 OS：Ubuntu 20.04.2 LTS
- CPU：Intel(R)Core(TM)i7-9700K @ 3.6GHz
- GPU：Nvidia Geforce RTX2070 OC @ 8GB
- ホスト RAM：16GB
- 仮想 RAM：4GB

3.1.2　実験データ

今回の実験で使用するデータは、R 言語のライブラリ arules に付属している Groceries と呼ばれるデータを使用する。以下にこのデータの概要を示す。

- 行数：9835, 列数：169
- 主要なアイテム：milk, vegetables, rolls/buns, soba, yogurut
- 先頭 5 行分データ

```
> head(as(Groceries, "data.frame"),5)
                                                      items
1              {citrus fruit,semi-finished bread,margarine,ready soups}
2                                   {tropical fruit,yogurt,coffee}
3                                                {whole milk}
4                        {pip fruit,yogurt,cream cheese ,meat spreads}
```

```
5 {other vegetables,whole milk,condensed milk,long life bakery product}
```

なお基本統計量は下記。

- 最小値：1.000
- 第一四分位数：2.000
- 中央値：3.000
- 第三四分位数：6.000
- 最大値：32.000

なお平均値は 4.409 である。また、出現回数上位 5 アイテムとその出現回数は次の通り。

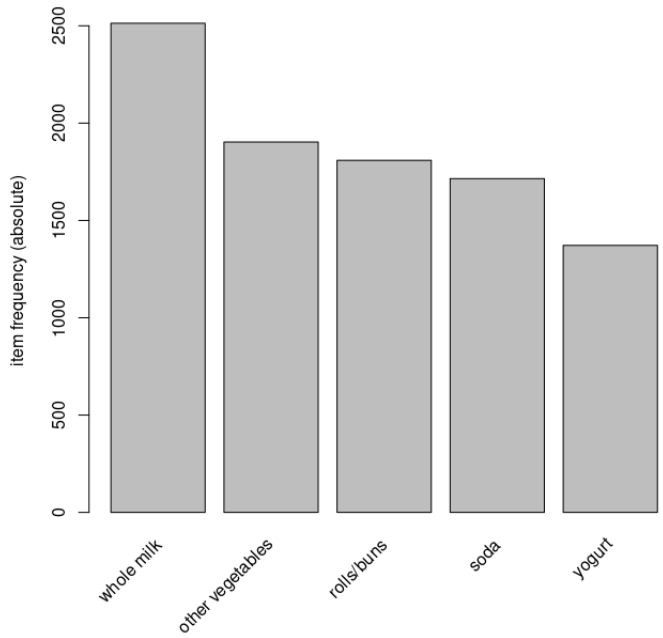

図 2　出現するアイテムとその頻度

3.1.3　R 言語での頻出パターン・相関ルール分析の手法

■頻度アイテム集合の抽出　頻出アイテム集合の抽出と相関ルール抽出には、先に示したように深さ優先探索的な手法と、幅優先探索的な手法の 2 種類が存在する。それらの方法はそれぞれ R 言語では「eclat」「Apriori」と呼ばれる関数を使用することで実現できる。

eclat 関数は頻出パターン抽出を行う関数である。以下説明する。 変数 <- eclat(データ, parameter= list(support, minlen, maxlen,target)) このとき、support は最小支持度でデフォルト値は 0.02、minlen は頻出アイテム集合の最小サイズでデフォルト値は 1、maxlen は頻出アイテム集合の最大サイズでデフォルト

値は 10、target は頻出アイテム集合の種類（頻出パターン、飽和パターン、極大パターン）を指定する。

■相関ルールの抽出　Apriori 関数は相関ルールの導出を行う関数である。 変数名 <- apriori(データ, parameter=list(support, confidence, maxlen, minlen)) support はルールの最小支持度でデフォルト値は 0.1, confidence はルールの最小確信度でデフォルト値は 0.8, maxlen はルールの前提部と帰結部の合計サイズがこのパラメータ値以下の相関ルールを対象とする、デフォルト値は 5。minlen はルールの前提部と帰結部の合計サイズがこのパラメータ値以上の相関ルールを対象とする。

3.2　実験結果

まず、デフォルト値で頻出パターン抽出を行った結果を以下に示す。

```
> inspect(groce.freq)
    items              support   transIdenticalToItemsets count
[1] {whole milk}       0.2555160 2513                     2513
[2] {other vegetables} 0.1934926 1903                     1903
[3] {rolls/buns}       0.1839349 1809                     1809
[4] {yogurt}           0.1395018 1372                     1372
[5] {soda}             0.1743772 1715                     1715
[6] {root vegetables}  0.1089985 1072                     1072
[7] {tropical fruit}   0.1049314 1032                     1032
[8] {bottled water}    0.1105236 1087                     1087
```

また、パラメータの値を以下のように変更して頻出パターン抽出を行ってみた。

- support=0.05
- minlen = 2
- maxlen = 5

この時の結果は次の通り。

```
> inspect(groce.freq)
    items                          support    transIdenticalToItemsets count
[1] {whole milk,yogurt}            0.05602440 551                      551
[2] {whole milk,rolls/buns}        0.05663447 557                      557
[3] {other vegetables,whole milk} 0.07483477 736                      736
```

次に、相関ルール抽出を apriori 関数を使用した導いた結果を示す。なお、パラメータはデフォルト。

結果は相関ルールとして検出されたルールは 1 つもなかった。

次に、パラメータを次のようにして apriori 関数を実行してみる。

- support=0.05
- confidence = 0.4

その時の結果は下記

```
  > inspect(groce.rules)
    lhs          rhs           support    confidence coverage  lift     count
[1] {yogurt} => {whole milk} 0.0560244 0.4016035  0.1395018 1.571735 551
```

また、パラメータを次のようにして再び apriori 関数を実行してみる。

- support=0.11
- confidence = 0.08

その時の結果は下記

```
  lhs    rhs                 support    confidence coverage lift count
[1] {}  => {bottled water}    0.1105236 0.1105236  1        1    1087
[2] {}  => {soda}             0.1743772 0.1743772  1        1    1715
[3] {}  => {yogurt}           0.1395018 0.1395018  1        1    1372
[4] {}  => {rolls/buns}       0.1839349 0.1839349  1        1    1809
[5] {}  => {other vegetables} 0.1934926 0.1934926  1        1    1903
[6] {}  => {whole milk}       0.2555160 0.2555160  1        1    2513
```

4 考察

一番初めのデフォルト値での apriori 関数の実行結果において、条件にヒットするルールが存在しなかったのか、1 つもルールが抽出されなかった。そのため、次の実験ではパラメータ「最小支持度」と「最小確信度」の値をデフォルト値よりも下げて、再び相関ルール抽出を行ってみたところ、1 つのみヒットした。この相関ルール”{yogurt} => {whole milk}”のリフト値を見てみると、約 1.57 となているのが分かるだろう。つまり、yogurt と whole milk の間には正の相関があると言えるだろう。

次に、パラメータを support=0.11、confidence = 0.08 としたときの相関ルール抽出では、6 件ヒットしたが、左辺がすべて空集合となってしまった。加えてすべての lift 値が 1 になった。これは、パラメータ confidence の値を極端に低く設定し過ぎたのが原因であると考えられる。1 つ前の相関ルール抽出の結果では、左辺は空集合ではなく、要素が入っていた。

以上のことから、相関ルール抽出の際にはユーザが指定するパラメータ support、最小支持度と confidence 最小確信度の設定によって得られる相関ルールに大きな違いが出ることが分かる。今回の実験で、相関ルール抽出時にはデータセットに対し、適切なパラメータを指定しなければ良い相関ルール抽出が行えないと考えられる。

5 まとめ

本稿では、R 言語を使用して、データ Groceries の頻出パターン抽出と相関ルール抽出を行った。結果は、頻出パターン抽出の方は適切に頻出アイテム集合を抽出することができたが、相関ルール抽出では指定するパラメータによって、ルールそのものが得られない場合があるなど、パラメータに値に大きく結果が左右された。

参考文献

[1] 集合について：https://mathlandscape.com/power-set/#:~:text=%E4%B8%80%E8%A8%80%E3%81%A7%E3%81%84%E3%81%86%E3%81%A8,%E3%81%AE%E9%9B%86%E5%90%88%E3%82%92%E6%8C%87%E3%81%97%E3%81%BE%E3%81%99%E3%80%82