Jリーグの攻撃プレーにおける ゲーム・パフォーマンス・ビッグデータの因子構造 － 探索的因子分析における実用的プログラムを適用して －

徐広孝・中西健一郎・青木優[1)]

Factor Structure of Game Performance Big Data in Offensive Play of Japan Professional Football League: Applying a Practical Program in Exploratory Factor Analysis

JO Hirotaka[1)], NAKANISHI Kenichiro[1)], and AOKI Masaru[1)]

Abstract

Previous studies on skill evaluation and scale construction using game performance data of soccer are based on the premise that the data reflect the skills required for soccer offense. However, no studies have verified whether Japanese professional level (J.League) game performance data reflects the skills required for soccer offense. Although Factor analysis can be used as a method to confirm this problem, there is a problem that it is difficult to properly use hyperparameters such as factor extraction method, rotation method, and number of factors in exploratory factor analysis. Therefore, in practice, inefficient analysis work is repeated. Against this background, in this study, we first developed a program (PAHFA: a Program that Analyzes all Hyperparameters in Factor Analysis) that is practically effective in exploratory factor analysis, and verified its practicality. Then, using the game performance big data (45 variables, 147,032 plays) of offense in J.League referenced to the previous research, the purpose was to elucidate the factor structure by exploratory factor analysis. As a result, it became clear that PAHFA has practicality. In addition, the game performance big data in offense of J.League was composed of 14 variables and 6 factors, and its content validity was confirmed. However, this factor structure is not a completely simple structure, and a task remains to elucidate a more robust factor structure.

Keywords： Soccer, Game performance, big data, Offensive play, Exploratory factor analysis, Program development

Ⅰ 背景

サッカーにおけるプロレベルの指導者であっても，試合内容を記憶できる割合は60%に満たないと指摘されている[1)]．そのため，指導の質を高めるために，近年はゲーム・パフォーマンス・データの分析結果を補助資料として活用している[2)]．ゲーム・パフォーマンス・データの分析は，1950年代から始まり，「頻度による量的分析」，「ゲーム状況の専門的評価による量的分析」，「ランニング，距離，スプリント，パスなどの生体と技術の分析」，「動的な戦術分析」という4つのフェーズを経て現在に至っているが[3)]，動的な戦術分析の成果を上げるためには，アナリストのスキルはもちろんのこと，それに応えるだけのデータの質と量が必要である．現在では，測定器機の発展によって，ボールタッチ・データとトラッキング・データの質，量ともに大

1）静岡産業大学スポーツ科学部
〒438-0043 静岡県磐田市大原1572－1

1）*Faculty of Sport Science, Shizuoka Sangyo University*
1572-1 Owara, Iwata, Shizuoka, 438-0043, Japan.

きく向上し，スポーツのビッグデータ[4]と呼ばれるほどになった．スポーツのビッグデータは，従来のデータでは不可能だった多様な分析を可能とするが，競技スポーツ現場のアナリストにとっては，高度なスキルがなければ，その管理と分析は容易ではないという問題も含んでいる．

そのため学術的水準では，有益な理論や分析手法等を開発し，競技スポーツ現場に還元することが求められる．この目的を達成するために，すでにサッカーのゲーム・パフォーマンス・データを用いた攻撃技能評価尺度[5]や，機械学習を適用した攻撃技能の測定項目[6]，守備技能の測定項目[7]などが開発されている．これらの先行研究の共通点は，いずれもゲーム・パフォーマンス・データを使用していることである．スポーツにおける競技力は，個人の潜在的な能力であるため，直接的に測定することができない．それゆえ，能力が発揮されることで顕在化するパフォーマンス[8]を測定し，実測のパフォーマンスから能力を推定するという方法が用いられている．ところが，ゲーム・パフォーマンス・データの使用に関しては，測定の誤差を無にすることは不可能であるため，パフォーマンスから真の能力を完璧に推定することはできないという，避けて通れない課題が存在する．言い換えると，パフォーマンス・データが，選手に内在する潜在的能力を正しく反映しているとは限らないという問題を完全に排除することはできないのである．こうしたパフォーマンスと能力の誤差問題を抱えたデータを使って技能尺度や測定項目を構成した研究[5,6,7]が行われているが，それらは「パフォーマンス・データが，潜在的なサッカー技能の発揮によって顕在化し，それらの間には一定の関係性がある」という前提に基づいて行われている．もし，この前提が成立しなかったとしたら，構成された尺度や測定項目は，サッカー技能を測っていることを保証できなくなる．

パフォーマンス・データからサッカー技能を推定できるかどうかを検証する方法として，因子分析による一因子性の確認があり，これまでにサッカー技能の測定項目[7]や，ビデオ映像テストによるサッカー戦術技能評価尺度[9]において用いられている．しかし，サッカー技能には様々な下位因子が存在するはずである．例えば，パスの技能，ドリブルの技能，縦への突破力，ギャップを作る能力などが挙げられる．こうした細分化されたサッカー技能は，一因子性の分析では検証することができないため，複数のベクトルを仮定した因子分析モデルにて検証しなければならない．複数の下位因子による構成概念妥当性が確認された場合，そのゲーム・パフォーマンス・データは，単なる一因子モデルに比べて，サッカー技能をより詳細に反映していることが明らかになる．サッカーの小規模なパフォーマンス・データを用いて因子構造を解明した研究[10,11]は存在するが，日本のプロサッカーリーグ（以下Ｊリーグ）において，１シーズンの全試合規模のビッグデータを対象にした研究は，今のところ実践例がない．ゆえに，Ｊリーグのゲーム・パフォーマンス・ビッグデータに因子分析を適用し，因子構造を解明することは，学術的に大きな意味がある．

因子分析[12]とは，観測データを用いて直接的に観測できない内面の構造を探索する手法であり，スポーツ科学分野においても，選手の体力や運動能力，専門的技能の推定に活用されている．現在では様々な方法（注１）を用いることができるが，多次元の因子ベクトルモデル[13]が登場して以降，「分析者の経験則に基づく任意性が介入して，ハイパーパラメータの決定や変数の取捨選択を行う」というプロセスは，いつの時代も変わっていない．特に探索的因子分析においては，内容的に妥当な因子構造が見つかるまで，その都度，分析者が因子負荷量や単純構造の出来具合をみて，変数の取捨選択を何度も繰り返さなければならない．そのため，ほとんどの場合，さらに妥当な因子構造が潜んでいる可能性があるにもかかわらず，ある程度納得のいく結果が出た時点で分析を打ち切ってしまう．このような変数の取捨選択の問題に対する方法として，ステップワイズ変数選択法[14]が考案されている．この方法は，従来の経験則に基づく変数選択とは異なり，α係数やω係数によ

る適合度指標に基づいて，排除すべき変数の候補を提示してくれるという点で，分析者の変数選択の負担を軽減してくれる．しかし，適合度指標は専門家の経験則とは質的に全く異なるものであり，ステップワイズ変数選択法を用いたとしても内容的に妥当な因子構造が自動的に抽出されるわけではない．したがって，どのような手法や基準で変数選択を行うかは，測定したデータや分析者の専門性に基づいて総合的に判断すべきである．

もう一点，探索的因子分析を実行する際に分析者が行わなければならないことは，回転法，因子抽出法，因子数などのハイパーパラメータの指定である．統計解析ソフトおよび計算速度の向上により，近年では斜交回転が多く用いられているが，斜交回転には，プロマックス回転，オブリミン回転，シンプリマックス回転，バイクォーティミン回転，斜交ジオミン回転などが存在する．各回転法を使い分けることは容易ではなく，実務的には「自分のデータをうまく説明できる解を求めればよく，いろいろな回転を試して，一番合っているものを見つけだす」という方法で分析されることが多い[15]．しかし，主要な統計アプリケーションのデフォルトがプロマックス回転になっていることも影響して，斜交回転を用いた論文の多くはプロマックス回転を適用する傾向がある[16]．例えばオブリミン回転など，他にも優れた回転方法があるにも関わらずこのような傾向があるということは，他の回転方法を試していない可能性が推察される．因子抽出法には最小二乗法，最尤法，主因子法，重みづけ最小二乗法など，様々な方法があり，それぞれに特徴がある．一般的には最尤法が優れた結果を導くことが多いとされているが，回転法と同様に，自分のデータに最適な方法を選ぶことは困難である[15]．因子数を決める方法は，ガットマン基準，スクリー基準，VSS 基準などがある．これらは，いずれもそれぞれの統計量とアルゴリズムに基づいて最適な因子数を推定するが，必ずしも分析者が想定する内容的に妥当な因子構造の因子数と一致するとは限らない．また，変数の取捨選択を行って変数の数が変われば，各基準の算出結果も変わるため，因子数もまた，分析者の任意性を排除することはできない．

このように，因子分析を実行する前に決定しなければならない回転法，因子抽出法，因子数は，いずれも分析者の考えや判断が必要となることから，その判断を誤れば最適な因子構造を逃してしまうことになりかねない．機械学習の分野では，モデルの性能を高めるために，様々なハイパーパラメータを試して最も性能の良い値を採用するというチューニングが行われるが，因子分析においてもこの考え方を適用させて，因子抽出法や回転法，因子数のすべての組み合わせを実行し，もっとも優良な結果を採用するという方法を使うことができると考える．このような処理を実行するプログラムがあれば，短時間でより多くのハイパーパラメータ・パターンから最適な因子構造を探ることができるようになる．

II　目的

本研究は，Jリーグのゲーム･パフォーマンス･ビッグデータが，サッカーに要求される技能を反映しているかどうかを検証するために，その因子構造を明らかにすることを目的とした．また，内容的に最も妥当な因子構造を見つけるために，因子抽出法，回転法，因子数のハイパーパラメータにおいてすべての組み合わせを実行するプログラムを開発し,その実用性もあわせて検証することとした．本研究における仮説は次の２点とした．

仮説１：因子分析におけるハイパーパラメータ（回転法，因子抽出法，因子数）のすべての組み合わせを分析し，その中から優れた結果を出力するプログラムは実用的である．

仮説２：Ｊリーグの攻撃におけるゲーム・パフォーマンス・ビッグデータには，サッカーの攻撃技能を構成する複数の下位因子が潜在し，因子分析によって内容的に妥当な因子構造が抽出される．

III　方法

1.　対象

2011年のJリーグ・ディヴィジョン1（以下J1）と，ディヴィジョン2（以下J2）の全686試合における，攻撃プレーのゲーム・パフォーマンス・データを分析対象とした．この年のJリーグは，J1は18クラブで34節，J2は20クラブで38節であった．本研究で使用したゲーム・パフォーマンス・データは，ボールを持った選手がドリブルやパスなどの動作を行うごとに1行記録されるボールタッチ・データであり，同等のデータを用いた先行研究[17]では，これを「アクションデータ」と称した．測定員は，データスタジアム株式会社にて一定の訓練を受けているため，測定値の信頼性は十分である．アクションデータは，1行が1プレーを表す「プレーデータ」に変換することで，プレーを対象とした分析が可能となるため[5]，先行研究[6]の手続きに準拠して，アクションデータをプレーデータに変換した．ただし，セットプレーは特殊な攻撃パターンが発揮されることがあり，因子分析の解の収束に影響を与える可能性が考えられるため，コーナーキック，ペナルティキック，直接フリーキック，間接フリーキックで始まるプレーを除くことにした．最終的に，1,312,117アクションを147,032プレーに変換し，これを分析対象とした．これほど大規模のパフォーマンス・データを用いた研究は極めて稀であり，サッカーにおけるゲーム・パフォーマンス分析の今後の発展性を鑑みて，本研究ではこれをビッグデータであると捉え，「ゲーム・パフォーマンス・ビッグデータ」と称した．

2.　変数

ゲーム・パフォーマンス・ビッグデータの因子構造を検証するためには，プレーデータを構築する時点で用意される変数が極めて重要な要因である．サッカーのゲーム・パフォーマンス・ビッグデータに勾配ブースティング決定木を適用し，プレー単位でシュートの有無を予測するモデルを構築した先行研究[6]は，本研究における貴重な知見を呈している．なぜなら，実際の試合ではサッカーの攻撃技能が備わっていなければシュートまでたどり着くことができず，シュートするか否かを機械学習モデルで予測できるということは，そのモデルで使用されている項目が攻撃技能を反映している可能性が高いと考えられるためである．そこで本研究では，その先行研究[6]で使用されている攻撃プレーの45項目（表1）を使用することとした．

3.　分析プログラムの実装

因子分析のハイパーパラメータを自動的にすべて試すことができれば，分析者は変数の取捨選択を効率的に行うことができるようになる．そこで，統計解析に長けたR言語（バージョン4.1）を用いて，引数に指定された因子抽出法，回転法，因子数のすべての組み合わせで因子分析を実行するプログラムを開発し，PAHFA（a Program that Analyzes all Hyperparameters in Factor Analysis）と命名した．

因子分析は相関行列をもとに計算されるため，統計学における異なる変数尺度が混在しているデータの場合は，尺度の組み合わせに応じた相関係数を選んで相関行列を作成することが推奨されている[18]．本研究で使用するプレーデータには，距離や回数などの間隔尺度変数と，エリアに進入した・しなかったなどを示す二値の名義尺度変数が混在していた．そこで，Rにおけるpolycorパッケージ[19]のhetcor関数を用いて，間隔尺度同士はピアソンの積率相関係数，間隔尺度と名義尺度の組み合わせはポリシリアル相関係数，名義尺度同士はテトラコリック相関係数により相関行列を作成し，その相関行列をpsychパッケージのfa関数に渡すようにした．

Thurston[20]に端を発する単純構造の考え方は，分かりやすい因子構造を見出すために有用な指標となる．単純構造とは，各変数の因子負荷量がひとつの因子に対してのみ大きく，その他の因子に対しては小さくなるような構造を指す．実際の分析の際は，因子負荷量の採用基準値を定め，すべての変数において，基準値を超える因子がひとつになるこ

表１　プレーデータの変数と定義（文献６より引用）

項目番号	測定項目（変数）	定義
1	シュート	そのプレーがシュートで終わったかどうか。
2	ドリブル	1プレー内のドリブル回数。
3	パス本数	1プレー内のパス回数（失敗を含む）。
4	パス成功数	1プレー内のパスの成功数。
5	パス成功率	1プレー内のパスの成功率。
6	ダイレクトプレー回数	1プレー内のダイレクトパス（トラップを行わないでつなぐパス）の回数。
7	ダイレクトプレー最大連続回数	1プレー内の連続したダイレクトパスの回数。
8	スルーパス	1プレー内のスルーパスの回数。
9	スルーパス成功数	1プレー内のスルーパスの成功数。
10	クロス	1プレー内のクロスボールの回数。
11	クロス成功数	1プレー内のクロスボールの成功数。
12	トラップ	1プレー内のトラップ（ボールを受ける動作）の回数。
13	こぼれ球獲得数	1プレー内でこぼれ球を奪取した回数。
14	フリックオン	1プレー内のフリックオン（ボールに軽く触れて軌道を反らす動作）の回数。
15	スローイン	そのプレーがスローインで始まったかどうか。
16	フィード	そのプレーがゴールキーパーからのフィードで始まったかどうか。
17	攻撃アクション総回数	1プレー内の攻撃アクション（動作）の合計回数。
18	攻撃時間（秒）	プレーが始まって終わるまでの秒数。
19	アクション平均時間	そのプレー内のアクションの平均所要時間（秒）。
20	プレー関与人数	そのプレー内で攻撃に加わった選手の人数。
21	総移動距離	1プレー内のアクションの2点間距離の合計。
22	縦総移動距離	1プレー内のアクションの2点間距離における縦方向距離の合計。
23	横総移動距離	1プレー内のアクションの2点間距離における横方向距離の合計。
24	移動距離	プレーが始まった位置から終わった位置までの距離。
25	縦移動距離	プレーが始まった位置から終わった位置までの縦方向距離。
26	横移動距離	プレーが始まって位置から終わった位置までの横方向距離。
27	縦最大移動距離	そのプレー内のアクションの座標から計算される縦方向の最大距離。
28	横最大移動距離	そのプレー内のアクションの座標から計算される横方向の最大距離。
29	パス距離平均	1プレー内のパスにおける2点間距離の平均値。
30	パス距離標準偏差	1プレー内のパスにおける2点間距離の標準偏差。
31	移動面積	1プレー内の連続する3つのアクションから構成される三角形の面積の合計。
32	前方推進力	そのプレーが前方に進む力を表す指標。
33	左右推進力	そのプレーが横方向に展開する力を表す指標。
34	前方移動比率	1プレー内の各アクションの進行方向において、前方に進んだ割合。
35	後方移動比率	1プレー内の各アクションの進行方向において、後方に進んだ割合。
36	右方移動比率	1プレー内の各アクションの進行方向において、右方に進んだ割合。
37	左方移動比率	1プレー内の各アクションの進行方向において、左方に進んだ割合。
38	切り返し	1プレー内において、連続する3つのアクションから成る角度が60度以下の回数。
39	方向転換	1プレー内において、連続する3つのアクションから成る角度が60度より大きく120度以下の回数。
40	同一方向展開	1プレー内において、連続する3つのアクションから成る角度が120度より大きく180度以下の回数。
41	2倍速	1プレー内において、ひとつ前のアクションの動作速度よりも2倍以上速くなった回数。
42	ペナルティエリア進入	そのプレーがペナルティーエリアに進入できたかどうか。
43	ペナルティ脇進入	そのプレーがペナルティーエリアの横に進入できたかどうか。
44	30ライン進入	そのプレーが30mラインより前に進入できたかどうか。
45	バイタルエリア進入	そのプレーがバイタルエリアに進入できたかどうか。

とを目指して因子分析を繰り返すことが多い[15]．必ずしも単純構造でなければならないという決まりはないが，解釈が明快になることから単純構造が好まれるケースがほとんどである．そこで，単純構造の度合いを測る指標として単純構造指数（Simple Structure Index, 式①）を定義し，PAHFA では計算された複数の因子分析結果の中から，単純構造指数の高い結果を優先して表示させるアルゴリズムを実装した．

$$SSI = \frac{V}{N} \quad \cdots \quad 式①$$

ここで，Nは変数の数，Vは採用基準値を超える因子負荷量がひとつの因子にのみ認められる変数の数である．すなわち，単純構造指数は，全変数に対して，ひとつの因子にのみ強い関係を持つ変数の割合である．

4. 因子分析

線形従属が認められる変数を除き，37個の変数でPAHFAの1回目を実行した．その際，因子分析のハイパーパラメータのうち，因子抽出法は最小二乗法，重みづけ最小二乗法，一般化重みづけ最小二乗法，主因子法，最尤法の5種類，回転法はプロマックス回転，オブリミン回転，シンプリマックス回転，斜交ジオミン回転，クラスター回転の5種類，因子数は2～10因子とした．その後は，PAHFAの結果を踏まえて，他の変数との相関が低い変数や，いずれの因子にも関係を持たない変数などの削除を経て，最終的に単純構造指数が1.0かつ内容的に妥当な因子構造となるまで，PAHFAを繰り返した．

5. 倫理的配慮

本研究で使用したJリーグのゲーム・パフォーマンス・データは，データスタジアム株式会社の許可を得たうえで使用した．

Ⅳ 結果

1. PAHFAの実装

PAHFAは，引数として受け取った因子抽出法，回転法，因子数のすべての組み合わせで因子分析を実行し，単純構造指数の高い結果を優先的に返す関数である．引数（ハイパーパラメータ）の指定からPAHFAの処理までの一連の流れ（アルゴリズム）は次の通りであった（図1）．

①使用したい因子抽出法，回転法，因子数をそれぞれ配列に保存する．
②データフレームの中から，使用する変数のみを選ぶ．
③相関行列を求める．
④因子抽出法，回転法，因子数の三重ループを開始する．
⑤相関行列から因子分析を行う．
⑥因子分析の結果において，基準（1.0）を超える因子負荷量がある場合は，推定不良とみなして結果を破棄する．
⑦基準を超える因子負荷量がない場合は，単純構造指数を求める．
⑧単純構造指数が基準（0.9）を超える場合は，因子分析の結果を出力用のテーブルに保存する．
⑨ループをすべて終えたら結果を返す．

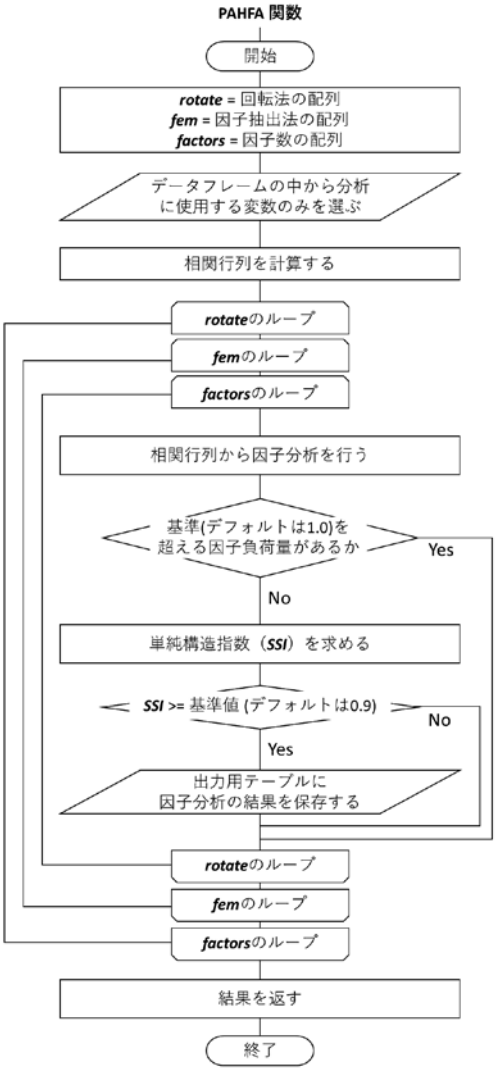

図1 PAHFAを適用した因子分析の基本的なアルゴリズム

一般的な高性能パソコン（HP 社製，EliteDesk 800 G2, Intel Core i7 CPU 3.4GHz, 8GB メモリ）を用いて，37 変数× 147,032 行のプレーデータに，5 種類の因子抽出法，5 種類の回転法，9 種類の因子数，合計 225 通りの因子分析を行ったところ，相関行列の算出に 6.12 秒，225 通りの因子分析を完了し，結果を返すまでに 46.23 秒要した．

なお，実際のプログラムと PAHFA 関数の使い方は，本論文の付録に添えることとした．

2. 因子構造

37 変数を使用した 1 回目の PAHFA を実行した結果，単純構造指数が 1.0 の結果が見つかった（表 2）．しかしながら，各因子に関係を持つ変数を見ても整合性がなく，内容的に妥当な因子構造であるとは言えなかった．その後，変数の取捨選択と PAHFA の再実行を繰り返し，15 回目で，ある程度，内容的に妥当な因子構造となった（表 3）．しかしながら，15 回目の分析結果も，完全な単純構造ではなかった．因子負荷量の基準値を 0.52 以上とした場合，第 1 因子から第 6 因子までは単純構造を示したが，第 7 因子には 0.52 を超える因子負荷量をもつ変数がなく，「縦最大距離」と「横最大距離」が，それぞれ 0.25 と 0.22 という弱い関係を示すのみであった．また「縦総距離」は，内容的には「縦移動」因子に含まれてよいと考えられるが，因子負荷量は 0.26 と低い値であった．このように完全な単純構造ではなかったことから，「縦最大距離」，「横最大距離」，「縦総距離」を削除して 16 回目の PAHFA を実行した．その結果，因子構造は大きく変化し，第一因子に 8 つの変数が集まった（表 4）．その後，変数の取捨選択を繰り返したが優れた結果は見いだせず，結果として，15 回目の結果が最も内容的に妥当であると判断した．最終結果の因子分析においては，オブリミン回転，最尤法が用いられた．

最終結果における各因子と，それに強く関係する変数を確認し，第一因子を「横移動」，第二因子を「繋ぎの動き」，第三因子を「縦移動」，第四因子を「サイド攻撃」，第五因子を「エリア進入」，第六因子を「パス」と命名した．第七因子は縦最大距離と横最大距離が弱い関係を示しているが，因子負荷量の基準値である 0.52 には届いていなかった．そのため，第七因子の命名はしないこととした．第 6 因子までの累積寄与率は 79% であり，良好な値を示した．

表２　37 変数を用いた１回目の PAHFA の分析結果

	因子1	因子2	因子3	因子4	因子5	因子6	因子7	因子8	因子9	共通性	独自性
トラップ	0.98	-0.05	-0.09	-0.01	-0.03	0.02	-0.07	0.01	-0.01	0.81	0.20
パス本数	0.97	-0.05	-0.06	0.04	0.01	0.00	0.03	-0.01	-0.02	0.84	0.16
攻撃AC総回数	0.97	0.01	-0.07	0.01	-0.02	0.01	-0.01	0.02	-0.01	0.87	0.13
展開	0.91	-0.03	-0.08	-0.01	-0.02	0.01	-0.07	-0.01	-0.01	0.71	0.29
左右推進力	0.90	-0.01	0.03	-0.01	-0.02	0.05	-0.03	-0.01	0.04	0.83	0.17
横総距離	0.90	0.00	0.03	-0.02	-0.02	0.05	-0.03	-0.01	0.03	0.85	0.15
方向転換	0.89	0.01	-0.07	-0.01	-0.02	0.01	-0.02	-0.01	0.00	0.75	0.25
切り返し	0.89	-0.09	-0.12	0.05	-0.02	-0.01	0.08	-0.02	-0.04	0.64	0.36
攻撃時間	0.84	0.06	0.05	-0.06	-0.03	0.02	0.01	0.05	-0.01	0.82	0.18
プレー関与人数	0.79	0.04	0.09	-0.01	0.03	0.02	0.03	0.03	0.02	0.79	0.21
縦総距離	0.78	0.13	0.08	-0.07	0.01	-0.06	0.03	0.00	-0.05	0.84	0.16
ダイレクト回数	0.67	-0.03	-0.07	0.02	0.06	-0.04	0.24	-0.04	0.01	0.53	0.47
面積	0.66	0.16	0.09	-0.06	-0.02	0.01	0.03	-0.01	0.01	0.69	0.31
横最大距離	0.45	-0.03	0.31	0.31	0.05	0.02	-0.05	-0.01	0.03	0.47	0.53
ペナ	-0.07	0.85	0.04	0.06	0.02	0.10	-0.01	0.01	0.05	0.70	0.30
スルーパス成功数	-0.22	0.85	-0.25	-0.07	0.06	0.24	-0.01	-0.07	-0.01	0.59	0.41
クロス	-0.09	0.78	0.01	0.23	-0.11	0.00	0.07	0.10	-0.11	0.58	0.42
ライン30	0.24	0.68	0.08	0.02	-0.02	-0.07	-0.04	-0.01	-0.01	0.75	0.25
ペナ脇	0.07	0.68	0.07	0.32	-0.09	-0.07	0.03	-0.01	-0.10	0.59	0.41
バイタル	0.17	0.66	0.02	0.02	0.03	0.07	-0.06	0.04	0.07	0.62	0.38
前方推進力	0.10	0.54	0.13	-0.22	0.05	-0.21	-0.13	-0.01	-0.01	0.65	0.35
縦距離	0.10	0.50	0.15	-0.27	0.03	-0.18	-0.13	-0.02	-0.01	0.63	0.38
ドリブル	-0.12	0.42	0.13	0.13	-0.19	-0.21	0.06	0.08	0.06	0.25	0.75
横距離	0.13	-0.07	0.62	0.06	0.02	0.25	0.00	-0.03	0.21	0.51	0.49
パス距離M	-0.07	0.12	0.59	-0.13	0.09	-0.05	0.03	-0.11	-0.02	0.46	0.55
パス距離SD	0.14	0.14	0.49	-0.08	0.05	0.09	0.21	-0.05	0.01	0.48	0.52
AC平均時間	-0.23	0.08	0.44	-0.30	-0.08	0.07	0.10	0.10	-0.02	0.20	0.80
縦最大距離	0.20	0.12	0.37	0.00	0.02	0.00	-0.03	0.14	-0.02	0.35	0.65
スローイン	0.06	0.06	-0.05	0.76	0.09	-0.04	-0.06	-0.27	-0.01	0.55	0.45
フィード	-0.07	-0.08	0.22	-0.62	-0.12	0.05	-0.07	-0.18	-0.12	0.42	0.58
フリックオン	-0.03	-0.14	0.06	0.22	0.96	-0.05	-0.01	0.10	-0.06	0.83	0.17
右方比率	0.06	-0.05	0.20	-0.14	0.05	0.63	-0.09	0.00	-0.19	0.36	0.64
前方比率	-0.02	0.24	-0.09	-0.02	0.12	-0.54	-0.09	-0.02	-0.17	0.45	0.55
倍速x2	0.08	0.04	0.11	-0.08	0.07	-0.11	0.73	-0.09	0.07	0.64	0.36
後方比率	0.09	-0.22	0.15	0.08	-0.07	0.11	0.47	0.12	-0.30	0.33	0.67
こぼれ球G数	0.00	0.07	0.03	-0.09	0.10	0.01	0.02	0.86	0.07	0.73	0.27
左方比率	0.00	-0.03	0.10	0.12	-0.06	-0.02	-0.17	0.07	0.74	0.48	0.52
寄与	10.36	4.51	1.82	1.42	0.97	0.95	0.92	0.91	0.71		
寄与率	0.28	0.12	0.05	0.04	0.03	0.03	0.03	0.03	0.02		
累積寄与率	0.28	0.40	0.45	0.49	0.52	0.54	0.57	0.59	0.61		

※ R では変数名の先頭に数字を使えないため、「2 倍速」は「倍速 x2」、「30 mライン」は「ライン 30」と置き換えた。

表3　PAHFA による 15 回目の分析結果

	因子1 横移動	因子2 繋ぎの動き	因子3 縦移動	因子4 サイド攻撃	因子5 エリア進入	因子6 パス	因子7	共通性	独自性
左右推進力	0.95	0.05	0.00	-0.01	0.00	0.01	-0.01	1.00	0.00
横総距離	0.91	0.07	0.01	-0.01	0.01	0.03	-0.01	1.00	0.00
横距離	0.59	-0.15	-0.01	0.06	0.01	-0.01	0.03	0.22	0.78
トラップ	0.07	0.93	0.03	0.02	0.05	-0.04	0.01	1.00	0.01
展開	0.20	0.65	0.06	0.00	0.01	0.08	-0.02	0.82	0.19
方向転換	0.23	0.52	0.03	0.02	0.09	0.17	0.00	0.83	0.18
前方推進力	-0.02	-0.01	1.00	0.02	-0.01	-0.01	0.01	0.98	0.02
縦距離	0.00	0.02	0.91	-0.02	0.02	0.00	-0.03	0.83	0.17
ペナ脇	0.03	0.00	0.07	0.99	-0.03	0.01	0.19	1.00	0.01
クロス	-0.02	0.04	-0.04	0.84	0.10	0.00	-0.33	1.00	0.01
バイタル	-0.03	0.10	0.00	-0.03	0.96	0.04	0.14	0.95	0.05
ペナ	0.10	-0.11	0.12	0.11	0.78	-0.01	-0.27	1.00	0.01
ダイレクト回数	0.09	-0.08	0.02	-0.02	0.05	0.77	0.00	0.65	0.36
パス本数	0.04	0.51	0.02	0.08	-0.01	0.53	-0.02	1.00	0.01
縦最大距離	0.13	0.01	0.13	0.12	0.28	0.03	0.25	0.31	0.69
横最大距離	0.35	0.02	-0.09	0.15	0.17	0.07	0.22	0.32	0.68
縦総距離	0.30	0.29	0.26	0.06	0.08	0.21	0.08	0.88	0.12
寄与	3.15	2.69	2.15	1.93	1.99	1.47	0.38		
寄与率	0.19	0.16	0.13	0.11	0.12	0.09	0.02		
累積寄与率	0.19	0.34	0.47	0.58	0.70	0.79	0.81		

表4　16 回目の PAHFA の分析結果

	因子1	因子3	因子4	因子2	因子6	因子5	共通性	独自性
左右推進力	1.02	0.00	0.00	0.01	0.04	-0.16	1.00	0.00
横総距離	1.01	0.02	-0.01	0.01	0.04	-0.14	1.00	0.00
トラップ	0.89	0.01	0.02	0.03	-0.10	0.04	0.87	0.13
パス本数	0.89	0.00	0.06	-0.01	-0.02	0.28	1.00	0.01
展開	0.85	0.05	0.00	-0.02	-0.02	0.05	0.77	0.23
方向転換	0.84	0.02	0.01	0.07	-0.02	0.08	0.81	0.19
ダイレクト回数	0.59	0.02	-0.06	0.04	0.08	0.33	0.55	0.45
横距離	0.43	-0.02	0.07	0.07	-0.06	-0.13	0.23	0.77
前方推進力	-0.01	1.01	0.01	-0.02	-0.01	0.00	1.00	0.01
縦距離	0.03	0.89	-0.02	0.02	0.01	0.00	0.82	0.18
ペナ脇	0.03	0.08	0.94	0.01	-0.23	-0.01	0.94	0.06
クロス	0.01	-0.04	0.92	0.04	0.25	0.01	1.00	0.01
バイタル	0.04	0.00	-0.03	1.01	-0.10	0.01	1.00	0.01
ペナ	-0.01	0.14	0.17	0.68	0.36	-0.03	1.00	0.01
寄与	5.78	1.95	1.88	1.72	0.32	0.31		
寄与率	0.41	0.14	0.13	0.12	0.02	0.02		
累積寄与率	0.41	0.55	0.69	0.81	0.83	0.85		

V　考察

本研究は，日本のプロレベルのサッカーにおける攻撃のゲーム・パフォーマンス・ビッグデータが，サッカーに要求される攻撃技能を反映しているかどうかを検証することを目的とし，その方法として探索的因子分析を用いた．因子分析は従来，心理学分野で誕生した手法であり，潜在的な心理量を推定することを目的としていた．しかし，サッカーにおける各種の技能は，心理量と同じく直接測定することができず，それゆえ試合中に発揮されるパフォーマンスを測定し，実測値から能力を推定する．そういった意味では，スポーツにおける能力推定に因子分析を用いることは妥当であると言える．しかし，因子分析には様々な因子抽出法と回転法があり，手元のデータに応じて，どの因子抽出法と回転法を用いるか，そして因子数をいくつに想定するのかを判断しなければならない．この手続きは決して容易ではないことから，分析の実務においては，「いろいろやって当てはまりのよい手法を選ぶ」という傾向がある[15]．そのため，機械学習のチューニングの考え方を応用し，因子分析においても，因子抽出法，回転法，因子数といったハイパーパラメータをすべての組み合わせで分析し，単純構造が明確な結果を優先して出力させるプログラムが実用的である．本研究において実装したプログラム（PAHFA）は，分析者が因子抽出法，回転法，因子数，および単純構造指数の採用基準等を指定して分析できることから，こうした実務的需要に応えるものである．一般的に高性能とされるコンピュータを使用し，37変数× 147,032行の大規模なデータに対して，225通りのハイパーパラメータを指定して分析した結果，分析結果の終了までに要した時間は46.23秒であり，極めて短時間で処理できることが明らかになった．これは，手作業でハイパーパラメータを変えて分析を繰り返す方法に比べると，作業時間の大幅な短縮になり，分析者は変数の取捨選択に注力することができる．

本研究では，次にPAHFAを用いて，Ｊリーグにおける攻撃のゲーム・パフォーマンス・ビッグデータの因子構造を明らかにすることを試みた．その結果，最終的にオブリミン回転，最尤法による７因子が抽出された．しかしながら，この因子構造は完全な単純構造ではなく，第７因子に強い関係を持つ変数はなかった．また，「縦総距離」の変数は，どの因子とも強い関係を持っていなかった．一方，第１～６因子においては，それぞれの因子と関係の強い変数を見ると，内容的に妥当であると判断された．その理由を以下に示す．

「左右推進力」は，そのプレーにおけるボールを左右に展開する力の程度であり，「前方推進力」は，ボールを前へ運ぶ力の程度である[6]．また，「横総距離」，「横距離」，「縦距離」はボールの移動距離の実測値である．横および縦へのボールの移動は，サッカー指導の現場のみならず，攻撃パフォーマンスの数量化においても重要な変数であることが指摘されている[21]．こうした変数が「横移動」因子と「縦移動」因子を構成していることから，両因子はサッカーの攻撃を構成する下位技能であると判断できる．「パス」因子は「ダイレクト回数」と「パス本数」で構成される．パスはサッカーにおける中核的な動作であり，パス本数が多い方が一回の攻撃における得点率が高いことが分かっている[22]．また，Ｊリーグのチームを対象にした分析では，相手の隙を突くダイレクトプレーが有効であると報告されている[23]．ボールを受ける動きである「トラップ」は，次の動きに繋げるために必要な動作であり，トラップした後に同じ方向へ移動すればボールを大きく展開することができる．さらに方向転換もサッカー選手にとって欠かせない能力である[24,25]．その他，バイタルエリア[25]やペナルティエリアを攻略することは，シュートするための最終的な課題であり，ペナルティ脇からクロスボールをあげることは，それらのエリアに進入するための有効な手段であると考えられる．

以上のことから，完全な単純構造を見出すことはできなかったものの，６つの因子において内容的な妥当性があることから，Ｊリーグにおける攻撃のゲーム・パフォーマンス・

ビッグデータは，サッカーの攻撃に要求される技能を反映していると考えられる．

VI　まとめ

日本のプロレベルのサッカーにおけるゲーム・パフォーマンス・ビッグデータが，サッカーの攻撃に要求される技能を反映しているかどうかを検証することには，学術的に大きな意義がある．それを解明する方法として用いられる探索的因子分析は，因子抽出法，回転法，因子数を分析者が指定しなければならず，データの特性に応じた最適なハイパーパラメータを選ぶことが困難であるという実情がある．これらの背景を受けて，本研究では，複数指定されたハイパーパラメータのすべての組み合わせで探索的因子分析を行い，単純構造指数に基づいて優れた結果を優先的に出力する関数（PAHFA）を実装し，先行研究の測定項目を用いてＪリーグの攻撃におけるゲーム・パフォーマンス・ビッグデータの因子構造を明らかにすることを試みた．その結果，以下の知見を得た．

① PAHFA は，大規模なデータかつ数百パターンのハイパーパラメータを短時間で処理することができ，探索的因子分析における実務において実用的なプログラムであることが明らかとなった．

② Ｊリーグの攻撃におけるゲーム・パフォーマンス・ビッグデータを用いた探索的因子分析の結果，14 変数で 6 因子が構成され，内容的に妥当であることが明らかになった．しかしながら，完全な単純構造は見出されなかった．

VII　本研究の限界

本研究で使用したゲーム・パフォーマンス・ビッグデータは，ボール保持者が起こしたアクションを測定したボールタッチ・データに分類される．ボール保持者の周囲の選手の動き（オフ・ザ・ボールの動き）はデータ化されてないため，本研究の結果は，ボールタッチ・データの範囲内で解釈する必要がある．

VIII　今後の課題

探索的因子分析の結果，14 変数 6 因子の構造が明らかになったが，ゲーム・パフォーマンス・データのさらなる分析への応用や発展性を鑑みて，より多くの変数を含んだ因子構造や，より頑健な単純構造を解明することが望まれる．また，因子間の因果関係を解明することも，指導現場への還元および学術的研究の発展において重要な意味を持つ．

IX　謝辞

本研究は，静岡産業大学特別研究支援経費の支援を受けて行われた．また，分析に使用したデータは，データスタジアム株式会社より提供（有償）された．ともに感謝の意を表する．

注１　ここでいう方法とは，因子抽出法や回転法のことである．

【引用参考文献】

1) Laird P., Waters L. Eye-witness recollection of sports coaches. Int. J. Perform. Anal. Sport, 8(1): 76-84, 2008.
2) Carling C., Williams M., Reilly T. The handbook of soccer match analysis. Routledge, Abingdon, UK, 2005.
3) Memmert D., Raabe D. Revolution im Profifußball: Mit Big Data zur Spielanalyse 4.0, Springer, Berlin, 2017.
4) Dmonte R., Dmello A. Big data in sports: Leverage big data in sports: An insight using SAP HANA. IJERT, 6: 380-383, 2017.
5) 徐広孝，横尾智治，安藤梢．Ｊリーグにおける選手とチームの攻撃力指標．統計数理研究所共同研究リポート 314, 1:21-26, 2014.
6) Jo H., Nishijima T., Construction of Offensive Play Measurement Items and Shot Prediction Model Applying Machine Learning in

Japan Professional Football League, Football Science, 19: 1-21, 2022.
7) Matsuoka H., Tahara Y., Ando K. Nishijima T. Development of Criterion-referenced Measurement Items for Soccer Defensive Tactical Play from Tracking Data. Football Science, 17:29-40, 2020.
8) 日本体育学会（監）. 運動パフォーマンス, 『最新スポーツ科学辞典』, 大修館書店, 東京, 2006. p.40.
9) Ando K., Mishio S., Nishijima T. Analysis of Items and Characteristics of a Criterion-referenced Self-administered Test utilizing Video Images for the Development of a Computerized Adaptive Test for Tactical Skills in Soccer, Football Science, 15:26-37, 2018.
10) 鈴木宏哉, 西嶋尚彦. サッカーゲームにおける攻撃技能の因果構造. 体育学研究 47: 547-567, 2002.
11) 武者尚志, 山本光, 清水優菜. サッカーにおける攻撃戦術尺度の作成と妥当性の検討：レアルマドリードを対象とした分析. 教育デザイン研究, 9: 100-108, 2018.
12) Spearman C. General intelligence objectively determined and measured. American Journal of Psychology, 15: 201-293, 1904.
13) Thurstone L. The vectors of mind. University of Chicago Press, Chicago, USA, 1935.
14) Kano Y., Harada A. Stepwise variable selection in factor analysis. Psychometorica, 65(1): 7-22, 2000.
15) 松尾太加志, 中村知靖. 2章 因子分析を自分でする. 誰も教えてくれなかった因子分析, 北大路書房, 2008. pp31-126.
16) 柳井晴夫. 因子分析法の利用をめぐる問題点を中心にして, 教育心理学年報, 39: 96-108, 2000
17) 徐広孝, 大澤啓亮, 見汐翔太, 安藤梢, 鈴木宏哉, 西嶋尚彦. サッカーの攻撃におけるプレーの最適化アルゴリズムの開発, 統計数理, 65(2):309-321, 2017.
18) 豊田秀樹, 第3章 質的因子分析, 因子分析入門 ~R で学ぶ最新データ解析 ~, 東京図書, 2012.
19) John F. polycor: Polychoric and Polyserial Correlations. R package version 0.7-10. https://CRAN.R-project.org/package=polycor, 2019.
20) Thurstone L. Multiple factor analysis, University of Chicago Press, Chicago, USA, 1947.
21) 大江淳悟, 磨井祥夫, 沖原謙, 塩川満久, 菅輝, 梶山俊仁, 黒川隆志. サッカーゲームにおける攻撃パフォーマンスの数量化. スポーツ方法学研究, 20(1): 1-14, 2007.
22) Hughes M., Franks I. Analysis of passing sequences, shots and goals in soccer. Journal of Sports Sciences, 23: 509-514, 2005.
23) 後藤泰則, サッカーにおける「ボール保持率」と「勝利」との関係性について, 新潟経営大学紀要, 24: 67-75, 2018.
24) 笹木正悟 金子聡, 福林徹, サッカー選手における後方への方向転換能力に関する研究, スポーツ科学研究, 5:45-57, 2008.
25) Ando K., Mishio S., Nishijima T. Validity and Reliability of Computerized Adaptive Test of Soccer Tactical Skill. Football Science, 15:38-81, 2018.

付録：PAHFA の使用方法と関数定義

1. 使用の準備
   ① R 言語を使用し、psych パッケージ，polycor パッケージをインストールする．
   ② pahfa 関数，pahfa.view 関数，fa.loadings.best 関数（次頁 5「関数定義」参照）を R にコピー&ペーストし，実行して関数オブジェクトを作成する．

2. 使用方法
   ① 探索的因子分析を実行したいデータフレームを用意する．
   ② pahfa 関数を実行する．その際，引数は次の通りに指定する．
   - data
     データフレームを指定する．各変数において，間隔尺度は integer または numeric，順序尺度は ordered，名義尺度は factor に設定する．
   - use
     欠損値の処理の方法を指定する．相関行列を作成する際，"pairwise.complete.obs"はペアワイズ，"complete.obs"はリストワイズにて処理する．デフォルトは"pairwise.complete.obs"．省略可．
   - use.items
     分析に使用したい変数名を配列で使用する．例えば，データフレーム内に V1, V2, V3, V4, V5 の 5 変数があり，その中から V1, V4, V5 のみを使いたい場合は，この引数に c(“V1”,”V4”,”V5”)と指定する．データフレーム内のすべての変数を使用する場合は省略または NULL を指定する．
   - cor.method
     相関行列の作成方法を指定する．”pearson”はすべての変数においてピアソンの積率相関係数を用いる．”hetcor”は polycor パッケージの hetcor 関数を用いて，変数尺度に応じた相関係数を選んで相関行列を作成する．デフォルトは”hetcor”．省略可．
   - cor.std.err
     cor.method に”hetcor”を指定した場合，この引数に TRUE を指定すると，相関係数の算出時に標準誤差を計算する．デフォルトは FALSE．省略可．
   - cor.ML
     cor.method に”hetcor”を指定した場合，この引数に TRUE を指定すると，相関係数の算出時に最尤推定を用いる．デフォルトは FALSE．省略可．最尤推定は処理時間が長くなることに注意が必要である。
   - rotate
     因子分析の回転法を配列で指定する．例えば，プロマックス回転とオブリミン回転を用いる場合は，この引数に c(“promax”,”oblimin”)と指定する．使用できる回転法は psych パッケージのヘルプを参照．
   - fem
     因子抽出法を配列で指定する．例えば，最尤法と主因子法を用いる場合は，この引数に c(“ml”,”pa”)と指定する．使用できる抽出法は psych パッケージのヘルプを参照．
   - factors
     因子数を配列で指定する．例えば，3～6 因子で分析する場合は，この引数に 3:6 または c(3,4,5,6)と指定する．
   - exclude.loadings.standard
     因子負荷量の推定不良とみなす基準値を指定する．例えば，この引数に 1.0 を指定すると，1.0 を超える因子負荷量が含まれている分析結果は保存されない．デフォルトは 1.0．省略可．
   - exclude.simple.standard
     単純構造指数の採用基準値を指定する．例えば，この引数に 0.9 を指定すると，単純構造指数が 0.9 未満の分析結果は保存されない．デフォルトは 0.9．省略可．

   ③ pahfa 関数の結果（戻り値）には，次のオブジェクトが含まれる．
   - df
     各因子分析の結果における主要な情報をまとめた表．
   - fa
     保存された因子分析の結果（psych パッケージの fa 関数の戻り値）のリスト．
   - cor
     計算された相関行列．

   ④ pahfa.view 関数を使用して，因子分析の結果を確認する．
   ⑤ 確認した結果を考察して，変数の取捨選択を行い，内容的に妥当な因子構造が見つかるまで PAHFA 関数を再実行する．

3. ライセンス

pahfa 関数，pahfa.view 関数，fa.loadings.best 関数に対して使用範囲の制限は設けないが、発表にあたっては本論文の引用を明記すること。また、著作権は執筆者にある。

4. 使用例

df という名前のデータフレームに保存されている V1，V4，V5，V7，V10，V12 の 6 変数を使って，プロマックス回転とオブリミン回転，最尤法と主因子法，因子数 2～3 で探索的因子分析を実行する場合，以下のコードを用いる．

```
#使用変数とハイパーパラメータの配列を作成する
used.items<-c(”V1”,”V4”,”V5”,”V7”,”V10”,”V12”)
vector.rotate<-c("promax","oblimin")
vector.fem<-c("ml","pa")
vector.factors<-2:3
#PAHFA を実行する（データの大きさによって異なるが、数秒～数分要する）
result.fa<-pahfa(data=df, use.items=used.items, rotate=vector.rotate, fem=vector.fem,
                 factors=vector.factors)
#結果の一覧表を表示する
View(result.fa$df)
#相関行列を表示する
View(result.fa$cor$correlations)
#相関係数の種類を表示する
View(result.fa$cor$type)
#最も単純構造指数の高い結果を表示する
print(pahfa.view(result.fa$fa[[1]]), quote=F)
#次に単純構造指数の高い結果を表示する
print(pahfa.view(result.fa$fa[[2]]), quote=F)

#補足：View は別ウィンドウに，print はコンソールに出力するコマンドである．どちらを使ってもよい．
```

5. pahfa 関数，pahfa.view 関数，fa.loadings.best 関数の定義

```
pahfa<-function(data, use = "pairwise.complete.obs", use.items=NULL, cor.method = "hetcor",
                cor.std.err = F, cor.ML = F, rotate, fem, factors,
                exclude.loadings.standard = 1.0, exclude.simple.standard = 0.9 ){
 #関数内サブルーチンの定義
 vector.join<-function(v,splitter=","){
   # v = ベクトル
   # splitter = 結合文字
   if(!is.vector(v)){
     stop("指定された変数がベクトルではありません．")
   }
   if(length(v)==1){
     return(v)
   }else{
     t<-NULL
     for(i in 1:(length(v)-1)){
       t<-paste(t,v[i],splitter,sep="")
     }
     t<-paste(t,v[length(v)],sep="")
     return(t)
   }
 }
 #引数等の確認
 if(length(which(installed.packages()[,"Package"]=="psych"))==0){
   stop("「psych」パッケージがインストールされていません．インストールしてから実行してください．")
 }
 if(length(which(installed.packages()[,"Package"]=="polycor"))==0){
   stop("「polycor」パッケージがインストールされていません．インストールしてから実行してください．")
 }
 if(!(use=="pairwise.complete.obs" || use=="complete.obs")){
   stop("引数「use」には pairwise.complete.obs または complete.obs を指定してください．")
 }
 if(!(cor.method=="pearson" || cor.method=="hetcor")){
   stop("引数「cor.method」には pearson または hetcor を指定してください．")
```

```
  }
  if(!is.numeric(exclude.loadings.standard)){
    stop("引数「exclude.loadings.standard」には０より大きい数値を指定してください．")
  }else if(!is.numeric(exclude.simple.standard)){
    stop("引数「exclude.simple.standard」には０以上１未満の数値を指定してください．")
  }
  err<-NULL
  if(length(use.items)>0){
    for(i in 1:length(use.items)){
      if(length(which(colnames(data)==use.items[i]))==0){
        err<-c(err, use.items[i])
      }
    }
  }
  if(!is.null(err)){
    stop(paste("引数「use.items」に指定された", vector.join(err, "，"), "は，データフレームの中に存在しません．
"))
  }
  #使用する変数のみを抽出する
  df<-data
  if(!is.null(use.items)){
    df<-df[,use.items]
  }
  #出力保存用変数を定義する
  output.df<-data.frame()
  output.fa<-list()
  message("相関行列を計算しています...")
  #相関行列を求める
  cormat<-NULL
  if(cor.method=="pearson"){
    #積率相関係数の場合
    df.num<-df
    for(i in 1:ncol(df.num)){
      df.num[,i]<-as.numeric(df.num[,i])
    }
    cormat<-cor(df.num, method="pearson", use=use)
    cor.result<-list(correlations=cormat)
  }else if(cor.method=="hetcor"){
    #尺度に応じた混合の場合
    cor.result<-polycor::hetcor(df, ML=cor.ML, std.err = cor.std.err, use=use)
    cormat<-cor.result$correlations
  }
  #プログレスバーを初期化する
  message("指定されたハイパーパラメータで因子分析を分析します．")
  pb.max<-length(rotate) * length(fem) * length(factors)
  #各種のカウンター変数を宣言する
  n.count <- 0 #分析数
  n.destruction <- 0 #破棄された結果の数
  n.save <- 0 #保存された結果の数
  n.not.solve <- 0 #解が収束しなかった分析数
  for(i1 in 1:length(rotate)){
    for(i2 in 1:length(fem)){
      for(i3 in 1:length(factors)){
        n.count <- n.count + 1
        msg<-paste(n.count, "/" , pb.max, " (", round(n.count / pb.max * 100, 1), "%)",
                   " 回転法=",rotate[i1], " 推定法=",fem[i2], " 因子数=",factors[i3],
                   " で分析しています...", sep="")
        cat(msg)
        fa.result<-NULL
        suppressMessages(suppressWarnings(
            #相関行列を使って因子分析を行う
            try(fa.result<-psych::fa(r = cormat, n.obs = NA, nfactors = factors[i3], rotate = rotate[i1],
                                     fm = fem[i2], scores = "tenBerge"))
        ))
        if(class(fa.result)[[1]]!="try-error" & !is.null(fa.result)){
          #解が収束した場合
          lds<-fa.result$loadings
          n.over.loadings<-0
          #因子負荷量が基準値を超えている数を調べる
          for(ir in 1:nrow(lds)){
            for(ic in 1:ncol(lds)){
              if(lds[ir,ic]>=exclude.loadings.standard){
                n.over.loadings <- n.over.loadings + 1
              }
```

```
        }
      }
      #累積寄与率を保存する
      if(length(which(rownames(fa.result$Vaccounted)=="Cumulative Var"))>0){
        suppressMessages(suppressWarnings(
            cum.var<-try(round(fa.result$Vaccounted["Cumulative Var",factors[i3]]*100,3))
        ))
        if(class(cum.var)=="try-error"){
          cum.var<-"エラー"
        }
      }else{
        cum.var<-"未算出"
      }
      #因子数を保存する
      n.fac<-factors[i3]
      #単純構造指数等を求める
      std.best<-fa.loadings.best(fa.result)
      loadings.std<-std.best[1,"基準値"]
      n.item.simple<-std.best[1,"単独の因子に係る項目数"]
      n.item.multi<-std.best[1,"複数の因子に係る項目数"]
      n.item.none<-std.best[1,"どの因子にも係らない項目数"]
      n.variables<-ncol(df)
      index.simple<-std.best[1,"単純構造指数"]
      index.multi<-std.best[1,"複雑構造指数"]
      index.none<-std.best[1,"非構造指数"]
      n.item.var<-std.best[1,"項目数の分散"]
      #破棄するか保存するかを判断する
      is.exclude = F
      if(n.over.loadings>0){
        is.exclude = T
      }
      if(index.simple < exclude.simple.standard){
        is.exclude = T
      }
      if(is.exclude){
        #結果を破棄する
        n.destruction <- n.destruction + 1
          message("破棄しました．")
      }else{
        #結果を保存する
        n.save <- n.save + 1
        output.df<-rbind(output.df,c(factors[i3], rotate[i1], fem[i2], n.over.loadings, cum.var,
                                     loadings.std, n.item.simple, n.item.multi, n.item.none,
                                     n.variables, round(index.simple,3), round(index.multi,3),
                                     round(index.none,3), round(n.item.var,3)))
        if(n.save==1){
          for(id in 1:ncol(output.df)){
            output.df[,id]<-as.character(output.df[,id])
          }
        }
        output.fa<-c(output.fa,list(fa.result))
        message("保存しました．")
      }
    }else{
      #計算結果がエラーの場合
      n.not.solve <- n.not.solve + 1
      n.destruction =  n.destruction + 1
      message("解が収束しませんでした．")
    }
  }
 }
}
if(nrow(output.df)>0){
  colnames(output.df)<-c("因子数","回転法","因子推定法","因子負荷量超過の数","累積寄与率",
                         "因子負荷量基準値","単独の因子に係る項目数","複数の因子に係る項目数",
                         "どの因子にも係らない項目数","項目数", "単純構造指数","複雑構造指数",
                         "非構造指数","項目数の分散")
}
message(paste("合計", n.count, "個の因子分析を行いました．"))
message(paste("保存された分析数 =", n.save))
message(paste("破棄された分析数 =", n.destruction))
message(paste("  うち，解が未集束 =", n.not.solve, "個"))
#単純構造指数の高い順に並べ替える
if(nrow(output.df)>0){
```

```
    index<-order(output.df$単純構造指数, decreasing=T)
    output.df<-output.df[index,]
    output.fa<-output.fa[index]
  }
  return(list(df=output.df, fa=output.fa, cor=cor.result))
}#PAHFA 関数の終わり

#因子負荷量・共通性・独自性・寄与率等のテーブルを作成します.
pahfa.view<-function(fa, fl="best", mark="(#)"){
  #== サブルーチン ==
  matrix.add<-function(tb,v){
    if(is.vector(v)){
      v<-matrix(v,nrow=1,ncol=length(v))
    }
    if(is.matrix(v)){
      n<-(ncol(tb)-ncol(v))
      if(n<0){
        tbrow<-matrix(NA,nrow(tb),-n)
        tb<-cbind(tb,tbrow)
      }else if(n>0){
        tbcol<-matrix(NA,nrow(v),n)
        v<-cbind(v,tbcol)
      }
      tb<-rbind(tb,v)
    }
    return(tb)
  }
  matrix.insert<-function(index,tb,v){
    if(is.vector(v)){
      v<-matrix(v,nrow=1,ncol=length(v))
    }
    if(is.matrix(v)){
      if((index<1)||(index>nrow(tb)+1)){
        warning("引数 index が正しくありません. ")
      }else{
        n<-(ncol(tb)-ncol(v))
        if(n<0){
          tbcol<-matrix(NA,nrow(tb),-n)
          tb<-cbind(tb,tbcol)
        }else if(n>0){
          tbcol<-matrix(NA,nrow(v),n)
          v<-cbind(v,tbcol)
        }
        if(index==1){
          tb<-rbind(v,tb)
        }else if(index==nrow(tb)+1){
          tb<-rbind(tb,v)
        }else{
          tb.a<-tb[1:(index-1),]
          tb.b<-tb[index:nrow(tb),]
          tb<-rbind(tb.a,v,tb.b)
        }
      }
    }
    return(tb)
  }
  matrix.add.col<-function(tb,v){
    if(is.vector(v)){
      v<-matrix(v,nrow=length(v),ncol=1)
    }
    if(is.matrix(v)){
      n<-(nrow(tb)-nrow(v))
      if(n<0){
        tbrow<-matrix(NA,-n,ncol(tb))
        tb<-rbind(tb,tbrow)
      }else if(n>0){
        tbrow<-matrix(NA,n,ncol(v))
        v<-rbind(v,tbrow)
      }
      tb<-cbind(tb,v)
    }
    return(tb)
  }
  matrix.insert.col<-function(index,tb,v){
```

```
  if(is.vector(v)){
    v<-matrix(v,nrow=length(v),ncol=1)
  }
  if(is.matrix(v)){
    if((index<1)||(index>ncol(tb)+1)){
      warning("引数 index が正しくありません. ")
    }else{
      n<-(nrow(tb)-nrow(v))
      if(n<0){
        tbrow<-matrix(NA,-n,ncol(tb))
        tb<-rbind(tb,tbrow)
      }else if(n>0){
        tbrow<-matrix(NA,n,ncol(v))
        v<-rbind(v,tbrow)
      }
      if(index==1){
        tb<-cbind(v,tb)
      }else if(index==ncol(tb)+1){
        tb<-cbind(tb,v)
      }else{
        tb.a<-tb[,1:(index-1)]
        tb.b<-tb[,index:ncol(tb)]
        tb<-cbind(tb.a,v,tb.b)
      }
    }
  }
  return(tb)
}
#== /サブルーチン/ ==
#推定法と回転法を保存
tb<-matrix(NA,nrow=6,ncol=2)
tb[1,1]<-"因子分析"
tb[2,1]<-"項目数"
tb[2,2]<-length(fa$values)
tb[3,1]<-"度数"
tb[3,2]<-fa$n.obs
tb[4,1]<-"推定法"
tb[4,2]<-fa$fm
tb[5,1]<-"回転法"
tb[5,2]<-fa$rotation
tb[6,1]<-"因子スコア推定法"
tb[6,2]<-fa$method
tb<-matrix.add(tb,NA)  #空行を追加
tb2<-fa$loadings  #因子負荷量を保存
tb2<-matrix.add.col(tb2,round(fa$communality,3))  #共通性を結合
tb2<-matrix.add.col(tb2,round(fa$uniquenesses,3))  #独自性を結合
if(fl=="best"){
  fl<-fa.loadings.best(fa)[[1]]
}
if(!is.null(fl)){
  #因子の選別をする
  itn<-rownames(fa$loadings)
  item.names<-NULL
  lds<-tb2
  tb2<-NULL
  for(f in 1:(ncol(lds)-2)){
    #因子負荷量を降順で並べ替え
    indices<-order(abs(lds[,f]),decreasing = T)
    lds<-lds[indices,]
    itn<-itn[indices]
    #基準値未満の行を探す
    i.item<-1
    while(abs(lds[i.item,f])>=fl){
      i.item <- i.item + 1
      if(i.item>nrow(lds)){
        break
      }
    }
    if(i.item > nrow(lds)){
      i.item <- nrow(lds)
    }
    if(i.item>=2){
      tb2<-rbind(tb2,lds[1:(i.item-1),])
      lds<-lds[i.item:nrow(lds),]
```

```
        item.names<-c(item.names,itn[1:(i.item-1)])
        itn<-itn[i.item:length(itn)]
      }
      if(is.null(nrow(lds))){
        break
      }
    }
    tb2<-rbind(tb2,lds)
    tb2<-round(tb2,3)
    item.names<-c(item.names,itn)
    if(!is.null(mark)){
      #基準値以上の場合に印を付与する
      for(f in 1:(ncol(tb2)-2)){
        for(i.item in 1:nrow(tb2)){
          if(abs(as.numeric(tb2[i.item,f]))>=fl){
            tb2[i.item,f]<-paste(tb2[i.item,f],mark,sep="")
          }else if(as.character(tb2[i.item,f])==as.character(fl)){
            #文字列として等しいかどうかを判定する
            tb2[i.item,f]<-paste(tb2[i.item,f],mark,sep="")
          }
        }
      }
    }
  }else{
    item.names<-rownames(fa$loadings)
  }
  #因子寄与を結合
  tb2<-matrix.add(tb2,round(fa$Vaccounted,3))
  #因子数に応じて列ラベルを作成しておく
  fac.colnames<-colnames(fa$loadings)
  if(length(fac.colnames)>0){
    for(i in 1:length(fac.colnames)){
      fac.no<-substring(fac.colnames[i],nchar(fac.colnames[i]),nchar(fac.colnames[i]))
      fac.colnames[i]<-paste("因子",fac.no, sep="")
    }
  }else{
    for(i in 1:ncol(fa$loadings)){
      fac.colnames<-c(fac.colnames, paste("因子",i,sep=""))
    }
  }
  #列ラベル（因子 x，因子 x，…，共通性，独自性）を挿入
  tb2<-matrix.insert(1,tb2,c(fac.colnames,"共通性","独自性"))
  #因子寄与率の行ラベルを作成しておく
  fa.vacc<-fa$Vaccounted
  if(nrow(fa.vacc)==5){
    rownames(fa.vacc)<-c("寄与","寄与率","累積寄与率","説明率","累積説明率")
  }else if(nrow(fa.vacc)==2){
    rownames(fa.vacc)<-c("寄与","寄与率")
  }
  #行ラベル（項目名，寄与，寄与率，他）を挿入
  tb2<-matrix.insert.col(1,tb2,c(NA,item.names,rownames(fa.vacc)))
  rownames(tb2)<-NULL
  colnames(tb2)<-NULL
  #ふたつの表を結合
  tb<-matrix.add(tb,tb2)
  #因子負荷量の基準値を入力
  if(!is.null(fl)){
    tb[6,ncol(tb)-1]<-"因子負荷量の基準値="
    tb[6,ncol(tb)]<-fl
  }
  rm(tb2,fac.colnames,fa.vacc)  #不要な変数の削除
  tb[is.na(tb)]<-""  #NA を""に変換
  return(tb) #リターン
} #pahfa.view 関数の終わり

#因子の採用基準となる因子負荷量の最適値を探します.
fa.loadings.best<-function(fa,min=0.2,max=0.9,step=0.01,full.return=F){
  df.result<-data.frame()
  lds<-fa$loadings
  n.items<-nrow(lds) #項目数
  std<-min
  item.over<-vector(length=n.items) #同一変数で基準値以上の数を保存するベクトル
  repeat{
    #同一変数で基準値を超える因子数を数える
```

```
    for(ir in 1:n.items){
      n.item<-0
      for(ic in 1:ncol(lds)){
        if(abs(lds[ir,ic])>=std){
          n.item<-n.item+1
        }
      }
      item.over[ir]<-n.item
    }
    #各因子の基準値以上の項目数を数える
    sd.items<-NULL
    for(ic in 1:ncol(lds)){
      sd.items<-c(sd.items, length(lds[abs(lds[,ic])>=std,ic]))
    }
    n.one.fac<-length(item.over[item.over==1]) #単独の因子に係る項目数を保存する
    n.multi.fac<-length(item.over[item.over>=2]) #複数の因子に係る項目数を保存する
    n.none.fac<-length(item.over[item.over==0]) #どの因子にも係らない項目数を保存する
    prob.simple<-n.one.fac/n.items #単純構造指数（単独の因子に係る項目数の割合）を計算する
    prob.complex<-n.multi.fac/n.items #複雑構造指数（複数の因子に係る項目数の割合）を計算する
    prob.non.structure<-n.none.fac/n.items #非構造指数（どの因子にも係らない項目数の割合）を計算する
    var.items<-var(sd.items) #項目数の分散
    #行に追加する
    df.result<-rbind(df.result,c(std, n.one.fac, n.multi.fac, n.none.fac, n.items,
                                 prob.simple, prob.complex, prob.non.structure,var.items))
    std <- std + step
    if(std > max){
      break
    }
  }
  colnames(df.result)<-c("基準値","単独の因子に係る項目数","複数の因子に係る項目数","どの因子にも係らない項目数",
                         "項目数", "単純構造指数","複雑構造指数","非構造指数","項目数の分散")
  if(full.return){
    return(df.result)
  }else{
    lds.max<-max(df.result[,"単独の因子に係る項目数"])
    df.result<-df.result[df.result[,"単独の因子に係る項目数"]==lds.max,]
    return(df.result[nrow(df.result),])
  }
} # fa.loadings.best 関数の終わり
```