

Excelの高度な数式はどこで学ぶのか

Microsoft Community Hubを入口に、検索・読解・検証をつなぐ学習ガイド

Copyright © 2026 LWP 山中 一弘

本資料は、出典を明記いただければ、商用・非商用を問わず、ご自由に複製・改変・再配布していただけます。なお、著作権表示は改変せず、そのまま記載してご利用くださいますようお願いいたします。

記事要約

Excelの高度な数式を学ぶには、関数の説明を読むことと、その関数を組み合わせて問題を解く過程を読むことの両方が必要です。公式資料で構文と制約を確かめ、コミュニティで応用の発想を見つけ、自分の小さな例で条件を確認する。この往復が、見つけた式を自分の知識へ変える土台になります。

Microsoft Community HubのExcelの議論には、LAMBDA、動的配列、REDUCE、MAKEARRAYなどを組み合わせた実験が蓄積されています。完成した回答だけでなく、質問の前提、うまくいかなかった理由、別解、訂正まで読むと、どこに設計上の分かれ道があるかが見えてきます。

本記事では、BYROWとThunkの議論を具体的な入口にして、探し方と読み方を説明します。Microsoftのサイトにある個人の回答を公式仕様と混同せず、再現条件を記録しながら学ぶ方法を扱います。

本記事の対象とゴール

対象は、通常のExcel関数を使っており、LETやLAMBDA、動的配列を使った応用へ進みたい方です。すべての関数を理解してから読む必要はありません。知らない関数を見つけたら公式説明へ戻り、議論での役割を確認する読み方を想定しています。

読後のゴールは、自分の疑問を検索語へ変換し、有用なスレッドを見つけ、その方法を採用できる条件と未確認の条件を書き分けられることです。ThunkやBYROWの代替関数を完成実装する教程ではなく、そうした技法を調べるための学習ガイドです。機能の対応版は関数ごとに確認してください。

第1章 公式ヘルプから、組合せの議論へ進む

関数の使い方を知ることと、問題を解くこと

公式ヘルプを読めば、関数が何を受け取り、何を返すかを調べられます。しかし、実際の疑問は一つの関数に収まるとは限りません。「各行を処理したい。しかも行ごとの結果を横に複数返したい」という要望では、行処理、出力の形、結果の結合をまとめて考える必要があります。

このようなとき、関数名だけを覚え直すより、同じ制約にぶつかった人の議論を読む方が、次の調査先を見つけやすくなります。公式資料は部品の性質を確認するために使い、応用の議論は部品の組み合わせ方を検討するために使います。

Microsoft Community Hubの位置づけ

入口は、Microsoft Community Hubの[Excelのディスカッション](https://techcommunity.microsoft.com)です。URLにはtechcommunity.microsoft.com が使われています。元ネタや過去のリンクにMicrosoft Tech Communityという名称があっても、URLと対象スレッドを手掛かりにたどれます。

読む際は、サイトの運営者と文章の書き手を分けます。Microsoftのドメインで公開されていても、利用者による回答が製品の公式仕様になるわけではありません。一方、個人の回答だから役に立たないということでもありません。入力例と理由が明確な実験は、応用を考えるための有力な材料です。

調べたいこと	主な参照先	読むときの焦点
引数・戻り値・制約	Microsoft Support / Learn	対象版、構文、エラー条件
機能の発表と背景	Excel Blogの製品チーム記事	筆者、公開日、提供段階
関数を組み合わせる発想	Community Hubの議論	質問の条件、別解、訂正
特定の問題の解法	Stack Overflow	最小例、回答の条件、コメント
自分の仕事への適用	自分のExcelでの確認	入力、期待結果、環境、性能

この表はサイトの順位表ではありません。一つの説明のうち、仕様は公式資料、応用方法は原投稿、再現結果は自分の記録というように、主張ごとに根拠を分けます。

第2章 疑問を、検索できる言葉へ変える

「高度なExcel」より、関数名と困りごと

広い言葉で検索すると、今の疑問と関係の薄い入門記事も多く見つかります。まず「何を使って」「何をしたい」「何が起きたか」を一文で書きます。例えば「BYROWの各行から複数の値を返そうとすると、#CALC!になる」です。

この一文から、関数名、返したいデータの形、エラーを抜き出します。検索語には英語も使います。これは英語で高度な説明を書くためではなく、原投稿と同じ語に合わせるためです。

検索欄へ貼り付ける例です。Excelのセルへ入力する式ではありません。

```
site:techcommunity.microsoft.com BYROW LAMBDA array  
site:techcommunity.microsoft.com "Nested arrays are not supported"  
site:techcommunity.microsoft.com Excel thunk lambda  
site:stackoverflow.com/questions Excel BYROW array
```

最初から条件を増やしすぎないことも大切です。結果が少なければサイト指定や一語を外し、多すぎれば関数名やエラーの語を足します。完全一致検索で見つからない場合は、引用符を外して表現の違いを拾います。

見つかった語を、次の検索へ持ち込む

検索は一回で終わる必要がありません。最初の回答で「array of arrays」という説明を見つけたら、それを次の検索語にします。Thunkという方法に出会ったら、関数名との組合せで調べます。検索語の更新自体が、問題の構造を学ぶ過程です。

良い説明を見つけたときは、投稿者名も補助になります。例えば後述のスレッドで登場するPeterBartholomew1やPatrick2788の名前に、LAMBDAやMAKEARRAYを組み合わせで探せます。ただし、その人が書いたという理由だけで方法を採用せず、各投稿の条件を読み直します。

検索結果の短い抜粋は、読むページを選ぶ手掛かりです。答えの確定には使いません。原投稿を開いて質問、回答、返信の日付を確認してから、自分の疑問との対応を判断します。

第3章 Thunkの議論を、小さな入口にする

最初に、質問者のほしい結果を読む

具体例として、2022年に始まった[Can SCAN\(\) and BYROW\(\) be combined?](#)を見てみましょう。質問の入口は、行ごとに文字を順に連結し、その途中結果を横へ返したいというものです。

この議論を読むときは、長い式の理解に入る前に、「一行から最後の結果だけがほしいのか、途中結果も並べたいのか」を押さえます。この違いを落とすと、別の問題を解いた式を同じ解法だと受け取ってしまいます。

PeterBartholomew1の回答には、配列を返す処理をThunkとして包み、必要なところで取り出して展開する考え方が示されています。原回答で示される短い定義は次のとおりです。

```
=LAMBDA(x, LAMBDA(x))
```

これは原回答の読解用の引用です。名前はThunk λ とされており、Excelの組込み関数名ではありません。式だけをセルへ貼れば完成結果が出るという意味でもありません。

包むことと、表示することを分ける

このパターンは、値や配列そのものをすぐに最終出力へ置く代わりに、それを返す引数なしの関数として扱う、という発想で読むと理解しやすくなります。中身を取り出す呼出しと、取り出した結果をシートに並べる処理は別です。

ここでは「配列を返す関数として包む」という説明までを入口にします。あらゆる数式の評価時点を自由に遅らせる、すべての配列関数がThunkを同じように扱える、といった一般化はしません。何を保持し、いつ呼び出し、何行何列へ展開するかは、採用する式で確認する事項です。

原スレッドにはその後の別解や性能に関するやり取りもあります。最初の回答を読んだら、同じ条件で何が改善され、どんな制約が残るのかを追ってください。「珍しい式を覚える」より、「なぜその回り道が必要なのか」を説明できることが学習上の到達点になります。

第4章 公式仕様と、投稿者の実験を照合する

BYROWの制約は、正確な単位で読む

Microsoftの[BYROW関数の説明](#)では、BYROWは行ごとにLAMBDAを適用し、結果の配列を返します。一方、そのLAMBDAには一つの結果を返すことが求められています。

したがって、「BYROWは配列を返せない」とだけ覚えると不正確です。BYROW全体は複数行の結果を配列として返します。問題になるのは、各行の処理からさらに複数要素の配列を返そうとする場面です。関数全体の結果と、内側のLAMBDAの戻り値を分けて読みます。

また、[#CALC!の公式説明](#)には、入れ子配列以外にも複数の原因が記載されています。#CALC!が出たという情報だけで、原因をThunkで解決すべき問題だと決めつけないことが大切です。

根拠を三つの文章に分ける

調査メモでは、次のように書き分けると判断が安定します。

1. 公式説明: BYROWの各行のLAMBDAは、一つの結果を返す。
2. 投稿者の方法: 結果を関数で包み、別の処理で取り出して展開する案がある。
3. 自分の確認: 採用した式が、自分の入力とExcel環境で期待どおりになるかを確かめる。

三つ目をまだ試していなければ、空欄にせず「未実行」と残します。公式仕様を読んだことも、他人の成功例があることも、自分の環境で動いたという証拠にはなりません。

Excel Blogは機能の発表時の背景を読む入口になります。ただし、公開時点の提供チャネルや機能の状態で、今の自分のExcelの状態は分けて確認します。古い記事の掲載日を見ずに、現在も同じ条件で使えると判断しないようにします。

第5章 肩書きと解決済み表示を、入口として使う

信頼は、主張と証拠の組合せで判断する

Microsoft社員やMVPなどの表示、投稿数、回答への評価は、読み進める対象を選ぶ参考になります。しかし、特定の式が自分の問題に適合するかは、表示とは別に確かめる必要があります。この記事では、投稿者名に確認していない肩書きを補いません。

特に注意したいのは、説明が詳しい回答を見つけると、入力条件まで自分のケースと同じだと思い込みやすいことです。固定された列数、空白のないデータ、文字列だけの入力など、答えが成立する条件を拾ってから比較します。

Solvedでも、返信を読む

SolvedやMarked as Solutionは、そのスレッドの解決策を探す有用な目印です。Microsoftによる一般的な動作保証や、あらゆる条件での最適性を表す認証としては扱いません。前章までの原スレッドにも、解決済みの表示と、その後続く別案があります。

読む順番は、質問、解決策とされた回答、その回答への補足や訂正、別解の順を基本にすると整理しやすくなります。返信が新しい順に表示されるページでは、表示順と議論の時系列を混同しないよう、日付を確認します。

採用前には「同じ入力か」「説明と結果が対応しているか」「後の訂正がないか」「自分の条件で確認できるか」の四点を見ます。反論がないことだけを正しさの証拠にはせず、検証に使える説明があるかを評価します。

第6章 別解とライブラリから、設計を学ぶ

別の場所で、同じ問題の切り分けを見る

Stack Overflowの[Alternative of BYROW for LAMBDA results of array](#)には、Tigregalisによる2024年2月24日の回答があります。行番号を使って各行を取り出し、REDUCEとVSTACKで結果を積み上げる考え方です。

この回答からは、標準BYROWの内側へ処理を押し込む以外に、自分で反復と結合を組み立てる選択肢が見つかります。ただし、同じ引数構成を持つことと、標準関数と同じすべての性質を持つことは別です。返す配列の幅、空入力、初期値の除去などを採用時に確認します。本記事では、その完成式の転載や動作保証は行いません。

Community Hubを継続した議論、Stack Overflowを質問と回答の対応が見える資料として読み比べるのは、有効な学習方法です。これは本記事での使い方の提案であり、両サイトのすべての投稿を分類する規則ではありません。

BigSpillを、関数の数ではなく構成から読む

Patrick2788が2026年8月9日に紹介した[BigSpill](#)も、応用を調べる入口です。紹介時点では92関数・10カテゴリとされ、同じスレッドには9月4日の追加告知があります。紹介時の数字を、常に最新の総数として覚える必要はありません。

[作者のGitHubリポジトリ](#)にはREADME、説明資料、導入用ブックへの案内があります。ここでは、関数名を大量に覚えるより、自分の課題に近い関数の一つを選び、引数、結果の形、依存する補助関数を順に読むことを勧めます。

例えば「この引数は利用者が変えたい条件なのか」「中間結果にどんな名前を付けているか」「一つの関数へ何を任せているか」という問いを持つと、ライブラリは設計の教材になります。対応環境は作者の説明と利用版を確認し、宣伝上の性能表現を自分の測定結果に置き換えないようにします。

添付ブックと、実務の話題の使い方

添付ブックがある場合は、式だけでなく入力表と結果と一緒に確認できます。まず元の例を理解し、次に入力を一箇所だけ変えると、何が結果を変えたかを追いやすくなります。原本は残し、試すコピーに利用日や変更点を記録しておくと、後で元の条件へ戻れます。

[Redditのr/excel](#)なども、仕事上の困りごとを探す入口として使えます。ここで得た着想は、採用する技術の原典や公式説明へつなげて確認します。本記事では、個別のReddit投稿を技術仕様の根拠にはしていません。

第7章 翻訳を使い、数式と重要な条件は原文へ戻る

全文を英語で精読する必要はない

英語の議論を読むときは、最初から全文を正確に訳そうとせず、ページ翻訳で話題と結論の位置をつかむ方法があります。FirefoxとChromeの翻訳機能は、それぞれの[Mozilla公式ヘルプ](#)と[Google公式ヘルプ](#)で使い方を確認できます。

次に、採用判断に関わる一文だけ原文へ戻ります。とくに「できる」「できない」「この場合だけ」「まだ試していない」の区別は落とせません。数式、名前定義、エラー文も原文を確認し、翻訳された説明に合わせて識別子を勝手に変えないようにします。

数式は文章より共有しやすい材料ですが、言語・地域設定によって関数の表示や区切り記号などが異なる場合があります。「数式だから、どの環境でもそのまま貼れる」とは考えず、使うExcelの構文に照らします。

用語は、次に検索できる形で残す

訳語だけで記録すると、後から英語の議論へ戻れなくなることがあります。次のように英語を併記します。意味はこの学習に必要な範囲の短い説明です。

英語	読むための手掛かり
dynamic array / spill	動的配列／結果が周囲のセルへ展開されること
nested array / array of arrays	配列の要素として、さらに配列を持つ構造
thunk	後で呼び出して中身を得る、引数なしの関数として扱う仕掛け
closure	関数が作られた環境の値を参照できる仕組み
higher-order function	関数を引数や戻り値として扱う関数
accumulator	累積処理の途中結果
recursion	再帰。処理の中で自分自身を呼び出すこと
fold / reduce	値を順に取り込み、累積結果を更新する考え方
workaround	制約を回避するための代替手段
helper function / custom LAMBDA	補助関数／自作のLAMBDA

array、range、reference、valueも、すべて同じ「データ」と訳して済ませないことが大切です。配列の値なのか、セル範囲への参照なのかによって、使える関数や説明の意味が変わります。用語を正確に残すことは、翻訳の補助であると同時に、次の検索の準備でもあります。

第8章 小さく再現し、正しさと性能を分けて確かめる

最初に期待結果を決める

式を試す前に、手で結果を求められる小さな入力を作ります。各行の集計なら二、三行から、配列を返す処理なら行数と列数を紙に書ける大きさから始めます。結果の値だけでなく、並び順、配列の向き、空白やエラーの扱いも期待結果に含めます。

最初の目的は、大量データで速く動かすことではありません。「この式は、どんな入力に対して、何を返すものか」を明らかにすることです。動いた結果を見てから期待結果を合わせると、誤りを見逃しやすくなります。

本記事で提案する確認順は次のとおりです。

1. 投稿の入力と期待結果を対応づける。名前定義があれば、その定義も確認する。
2. 自分の小さな入力で同じ考え方を試し、値と配列の形を比べる。
3. 空白、数値と文字列の混在、行数や列数の変化を、一条件ずつ加える。
4. 必要な条件で正しさを確認してから、実際に近いデータ量へ増やす。
5. 採用した式、環境、結果、残った制約を記録する。

すべての方法で同じ試験項目が必要なわけではありません。固定幅の表しか扱わないなら、その範囲を明記して確認します。可変幅まで一般化する必要が生じたときに、確認範囲を広げます。

「速かった」には、条件を付ける

高度な式では、正しい結果が出ることと、実用的な時間で計算できることを分けて考えます。ある投稿でREDUCE、MAKEARRAY、再帰、Thunkの方式が速かったとしても、自分の表の形やデータ量で同じ順位になるとは限りません。

比較するなら、同じ入力、同じ出力、同じExcel環境で測ります。行数だけでなく列数、文字列の長さ、数式の配置、再計算の方法も記録します。初回と再計算を混ぜず、複数回の結果を残せば、一回の偶然を一般化しにくくなります。

短い学習メモなら、次の項目で十分に始められます。

項目	記録する内容
出典	投稿URL、回答者、参照日
環境	Excelの版・ビルド、OS、必要なら更新チャンネル
入力と期待結果	行列数、データ例、値と出力形
使用式	名前定義、補助関数、原文からの変更点
結果	一致・不一致、エラー、未実行の区別
性能と制約	測定条件、所要時間、未対応の入力

本記事はこの確認方法を提案するもので、掲載したコミュニティの式やBigSpillを実機で検証した報告ではありません。性能の測定値も提示していません。

第9章 答えよりも、判断の過程を持ち帰る

一つの疑問を、一周させる

学習を始めるときは、LAMBDA関連のすべてを理解しようとせず、今の疑問を一つだけ選びます。検索し、公式説明と原投稿を読み、小さく確認して、残った疑問を次の検索へつなぐ。その一周を終えることを目標にします。

BYROWの例なら、まず内側のLAMBDAの出力条件を確かめます。次に、Thunkで包む案や、反復と結合を自分で組み立てる案を見つけます。そして、ほしい結果に必要な方法を選び、自分の条件で確かめます。すべての別解を実装することが、学習の完成条件ではありません。

読んだスレッドを保存するときは、URLだけにせず「何が制約だったか」「どの条件で方法が変わるか」を一、二文で添えます。完成した式だけの保存より、後で似た問題に出会ったときに判断を再利用しやすくなります。

使える知識として残す

難しい式を使うこと自体を目標にすると、保守しづらい仕組みを選びやすくなります。式を引き継ぐ人が理解できるか、補助列で分けた方が確認しやすくないか、更新処理なら別の道具が適さないかも、自分の用途に照らして考えます。

Microsoft Community Hubの価値は、答えを得られることに加え、他の利用者がどこで困り、何を試し、どの条件で考え直したかを読めることにあります。公式資料で仕様を確かめ、議論から設計の選択肢を学び、自分の確認で適用範囲を定める。この往復を続けることで、検索した知識を自分の判断に使えるようになります。

出典メモ

確認日: 2026年9月17日。以下の外部資料は、公式説明と投稿者本人の回答・公開資料を区別して参照しています。他者の画像や長いコードは転載せず、Thunkの短い定義だけを出典付きで引用しました。冒頭の利用許諾は本記事の独自執筆部分に適用し、外部資料の権利を変更するものではありません。

- 元ネタ: ほえほえ指定「260913_S2_整理済_Excel_Community_Hub_高度数式_学習ガイド_URL付き_記事元ネタ.md」。学習経路、検索語、情報源の区別を出発点に再構成しました。
- [Microsoft Support: BYROW関数](#) — 第4章の戻り値と制約を確認。
- [Microsoft Support: #CALC!の修正方法](#) — 入れ子配列を含む複数の原因を確認。
- [Can SCAN\(\) and BYROW\(\) be combined?](#) — TheDubの質問、PeterBartholomew1の2022年3月11日回答、後続の別解を参照。Thunk定義の引用元。
- [Alternative of BYROW for LAMBDA results of array](#) — Tigregalisの2024年2月24日回答を、反復と結合という考え方の参照に使用。
- [BigSpill紹介スレッド](#) — Patrick2788の2026年8月9日の紹介と9月4日の追加告知を確認。
- [BigSpill作者リポジトリ](#) — READMEと説明資料・導入ブックへの案内を確認。収録関数の全数調査、添付ブックの実行、対応環境の実測は未実施。
- [Firefoxページ全体の翻訳](#)、[Chromeのページ翻訳](#) — 各社の公式操作案内を確認。

追加の探索先は、[Excel Blog](#)、[excel-formulaタグ](#)、[excel-lambdaタグ](#)です。これらの一覧全件や、個々の投稿の正しさを検査したという意味ではありません。

元ネタにあった「Announcing LAMBDA Helper Functions」の個別ページは、今回の取得では本文を確認できなかったため、そこに書かれたとされる詳細は根拠に採用していません。数式の実行と性能比較は本記事の確認範囲外です。検索・読解・検証の順序と学習メモの様式は、本記事による実務上の提案です。