# **A Compendium of Esoteric & Foundational Crates for the Rust Ecosystem**

## **Forging New Wands for the Rust Ecosystem**

The Rust programming language has achieved remarkable maturity in a relatively short period. Its ecosystem, centered around crates.io, boasts powerful, production-ready libraries for domains such as web development, embedded systems, and databases.1 A survey of popular resources like the

awesome-rust lists reveals a vibrant community building complex, high-performance applications. However, a deeper analysis of the available crates, particularly in categories like "Algorithms" and "Computation," reveals a subtle maturity gap when compared to ecosystems like Python or Go.3 While foundational libraries certainly exist, the rich layer of fine-grained, specialized, and highly ergonomic computational utilities is less developed.

This gap does not represent a deficiency but rather a significant opportunity. The comprehensive utility libraries found in other languages—such as Python's itertools and collections modules, Go's slices package, or JavaScript's lodash library—are the product of decades of developers identifying and solving recurring, small-scale algorithmic problems.5 These battle-tested collections of functions are not mere conveniences; they form an "iterator algebra" and a vocabulary of data manipulation that accelerates development, reduces bugs, and promotes efficient, idiomatic code.9 Rust's relative youth means that many of these "solved problems" from other ecosystems represent greenfield opportunities for high-impact contributions. The user's query is therefore exceptionally well-timed, targeting the creation of small, focused crates that can fill these niches and provide immense quality-of-life improvements for developers engaged in data-intensive or algorithmic work.

Rust is a uniquely suitable proving ground for these new libraries for several key reasons. Firstly, its commitment to zero-cost abstractions and fine-grained memory control makes it the ideal language for implementing CPU-level bit-twiddling algorithms and memory-frugal compact data structures, tasks that would be inefficient or impossible in higher-level managed languages.10 Secondly, the language's powerful ownership model and type system provide a fortress of safety, preventing entire classes of common C++ bugs like iterator invalidation or data races, which are particularly pernicious in complex geometric or concurrent algorithms. This allows for the creation of "next-generation" implementations of classic algorithms that are not only fast but also fundamentally more robust and secure. Finally, Rust's modern tooling, particularly its seamless interoperability with WebAssembly (WASM) and its straightforward Foreign Function Interface (FFI), means that a high-performance mathematical library written in Rust can have an outsized impact, serving not only the native Rust community but also the web and as a performant backend for languages like Python and Ruby.

This report presents a curated list of over 100 such opportunities. Each idea is a proposal for a small, focused crate of under 300 lines of code, designed to be both mathematically rigorous and highly useful. The ideas are drawn from a deep survey of academic literature in computer science and mathematics, as well as a comparative analysis of the most successful utility libraries in other major programming languages.

### **Methodology and Scoring**

To provide a pragmatic filter for prioritizing effort, each idea in the following compendium is evaluated against two key metrics, each on a 1-to-5 scale:

* **Probability of Product-Market Fit (PMF):** This score estimates the likelihood that the proposed crate will be adopted and used by the community.
  + **5:** A direct port of a universally used, fundamental function from a major language's standard library (e.g., Python's itertools.combinations). Its utility is already proven beyond doubt.
  + **4:** A widely-used utility from a popular third-party library (e.g., lodash, more-itertools) or a very common algorithm (e.g., a standard statistical test).
  + **3:** A more niche but powerful algorithm from academic literature that solves a well-known, recurring problem (e.g., a specific type of sequence alignment or a compact data structure).
  + **2:** A useful but highly specialized algorithm that may only appeal to a small sub-community (e.g., specific cache-oblivious structures or esoteric bit hacks).
  + **1:** A speculative or highly theoretical idea that is academically interesting but has an unproven practical application.
* **Ease of Success-Testing:** This score assesses the difficulty of verifying the correctness of an implementation.
  + **5:** The function is pure, with simple, discrete inputs and outputs that are easily verifiable against a known specification (e.g., sorting, bit manipulation).
  + **4:** The logic is straightforward, but may involve iterators or state, requiring more comprehensive test cases.
  + **3:** The algorithm is complex or has numerous edge cases that must be tested. May require validation against established reference implementations in other languages (e.g., statistical tests).
  + **2:** The algorithm involves difficult edge cases related to floating-point arithmetic or complex geometric configurations, making robust testing challenging.
  + **1:** The correctness is difficult to prove automatically and may rely on visual inspection, statistical validation of distributions, or formal methods.

## **The Grand Compendium of Esoteric & Foundational Crates**

The following table is a curated roadmap of high-impact open-source opportunities. It is organized thematically to guide exploration through different domains of computer science and mathematics, from battle-tested utilities proven in other ecosystems to cutting-edge algorithms from academic research. Each entry represents a potential crate that is small in scope but large in potential value to the Rust community.

### **The Itertools & Collections Grimoire (Inspired by Python, Go, & JS)**

This section details small, powerful iterator adaptors, combinatoric functions, and specialized collection utilities ported from mature ecosystems. These ideas have an inherently high probability of product-market fit, as their utility has been demonstrated over years of widespread use in languages like Python, Go, and JavaScript. Rust's lazy, composable Iterator trait provides a perfect foundation for this "iterator algebra," making these additions feel native to the language's design philosophy.9

| Crate Name (Harry Potter Themed) | Description & Reasoning | Probability of PMF (1-5) | Ease of Success-Testing (1-5) | Relevant Artifacts (Links) |
| --- | --- | --- | --- | --- |
| **The Goblet of Fire** | An iterator adaptor tee(n) that splits a single iterator into n independent iterators. Each new iterator yields the same items as the original. Essential for algorithms that need to perform multiple independent passes over a sequence without collecting it into a Vec. This is a direct port of Python's highly useful itertools.tee. | 5 | 3 | 9 |
| **The Time-Turner** | An iterator adaptor cycle() that endlessly repeats the items of the underlying iterator. This is fundamental for tasks involving repeating patterns, circular buffers, or generating cyclical data streams. A core utility in Python's itertools. | 5 | 5 | 5 |
| **The Room of Requirement** | A DefaultHashMap and DefaultBTreeMap struct that mimics Python's collections.defaultdict. When a key is accessed that does not exist, a user-provided closure is called to generate a default value, which is then inserted and returned. This drastically simplifies code for grouping and counting tasks. | 5 | 4 | 14 |
| **Gringotts' Vault** | A Counter struct, essentially a HashMap<T, isize> wrapper, that implements multiset arithmetic. It supports efficient element counting and operations like addition, subtraction, union (`), and intersection (&) between counters. This is a powerful and surprisingly missing data structure from Python's collections` module. | 5 | 4 |  |
| **The Marauder's Map** | An iterator adaptor chain\_from\_iterable() that takes an iterator of iterators (e.g., &) and flattens it by one level, yielding the items of the inner iterators sequentially. This is the equivalent of Python's itertools.chain.from\_iterable and is more ergonomic than flatten() for certain types. | 5 | 5 | 9 |
| **The Sorting Hat** | An iterator adaptor groupby() that groups consecutive elements from a sorted iterator that share the same key, as determined by a key function. Yields (key, group\_iterator) pairs. A direct and powerful port of Python's itertools.groupby. | 4 | 3 | 9 |
| **The Pensieve** | A peekable iterator extension that allows looking ahead more than one item without consuming the iterator (e.g., peek\_nth(n)). The standard library's Peekable only allows looking one item ahead. Inspired by Python's more-itertools spy or peekable concepts. | 4 | 4 | 18 |
| **The Howler** | An iterator adaptor repeat(element, n) that yields a given element n times (or infinitely if n is omitted). A simple but fundamental building block from Python's itertools. | 5 | 5 | 13 |
| **The Triwizard Tournament** | A collection of combinatoric iterators: permutations(r), combinations(r), and combinations\_with\_replacement(r). These are mathematically pure, highly useful for solving algorithmic problems, and are a cornerstone of Python's itertools library. | 5 | 5 | 12 |
| **The Portkey** | An iterator adaptor batched(n) or chunked(n) that yields non-overlapping chunks (slices or Vecs) of size n from the source iterator. The last chunk may be smaller. Essential for batch processing of data. Inspired by Python's itertools.batched and Go's slices.Chunk. | 5 | 4 | 7 |
| **The Knight Bus** | An iterator adaptor windowed(n) that returns a sliding window of size n over the iterator. Unlike chunked, the windows overlap. For an input and `n=3`, it would yield, then ``. From Python's more-itertools. | 5 | 4 | 18 |
| **The Mirror of Erised** | A pairwise() iterator adaptor that yields overlapping pairs of items. For an input ``, it yields (1, 2), (2, 3), (3, 4). A special case of windowed(2) but common enough to warrant its own utility. From Python's itertools. | 5 | 5 | 9 |
| **The Invisibility Cloak** | An iterator adaptor filter\_false(predicate) which is the opposite of filter(). It yields only the items for which the predicate returns false. A direct port from Python's itertools. | 4 | 5 | 9 |
| **The Philosopher's Stone** | An iterator adaptor accumulate(func) that produces a running total or other accumulated result. For example, .accumulate(+) would yield 1, 3, 6, 10. From Python's itertools. | 4 | 4 | 12 |
| **The Daily Prophet** | A ChainMap struct that groups multiple HashMaps or other mappings into a single, updateable view. Lookups search the maps in order, while writes only affect the first map. Excellent for managing configuration overrides or simulating nested scopes. From Python's collections. | 4 | 3 | 14 |
| **The Whomping Willow** | A deque (e.g., VecDeque) wrapper that provides a maxlen constructor. When a bounded deque is full, adding an item to one end automatically discards an item from the opposite end. A core feature of Python's collections.deque. | 4 | 4 | 6 |
| **The Golden Snitch** | A slice utility compact() that removes consecutive runs of equal elements in-place, returning the modified (and possibly shorter) slice. A direct port of Go's slices.Compact. | 4 | 5 | 7 |
| **The Quaffle** | A slice utility union() that returns a new slice containing the unique elements from a set of input slices. Inspired by lodash.union. | 4 | 4 | 20 |
| **The Bludger** | A slice utility xor() that returns the symmetric difference of a set of input slices (elements that appear in an odd number of the input slices). Inspired by lodash.xor. | 3 | 4 | 20 |
| **The Deluminator** | An iterator adaptor dropwhile(predicate) that drops elements from the start of an iterator while the predicate is true, then yields all subsequent elements. From Python's itertools. | 4 | 4 | 9 |
| **The Remembrall** | An iterator adaptor takewhile(predicate) that yields elements from the start of an iterator as long as the predicate is true, then stops. The opposite of dropwhile. From Python's itertools. | 4 | 4 | 9 |
| **The Sword of Gryffindor** | An iterator adaptor compress(selectors) that filters an iterator, yielding only the elements where the corresponding element in a boolean selectors iterator is true. From Python's itertools. | 4 | 5 | 5 |
| **The Elder Wand** | A starmap(func) iterator adaptor that takes an iterator of tuples and applies a function func to the unpacked elements of each tuple. E.g., [(2, 5), (3, 2)].starmap(pow) yields 32, 9. From Python's itertools. | 4 | 5 | 9 |
| **The Resurrection Stone** | A zip\_longest(fill\_value) iterator that works like the standard zip but continues until the longest iterator is exhausted, filling in missing values from shorter iterators with a fill\_value. From Python's itertools. | 5 | 4 | 9 |
| **The Diadem of Ravenclaw** | A slice utility sorted\_index(value) that performs a binary search on a sorted slice to find the index where value should be inserted to maintain order. A port of lodash.sortedIndex. | 4 | 5 | 20 |
| **The Locket of Slytherin** | A slice utility intersection() that computes the set intersection of multiple slices, returning a new slice with the common elements. A port of lodash.intersection. | 4 | 4 | 20 |
| **The Cup of Hufflepuff** | A partition(predicate) function for iterators that consumes the iterator and returns two collections: one with elements for which the predicate was true, and one for which it was false. A port of lodash.partition. | 5 | 4 | 20 |
| **The Boggart** | A groupBy(key\_func) function for iterators that consumes the iterator and returns a HashMap where keys are derived from the key\_func and values are Vecs of the grouped items. A port of lodash.groupBy. | 5 | 4 | 20 |
| **The Dementor's Kiss** | An unzip() function that reverses the operation of zip. Given an iterator of tuples like [(a1, b1), (a2, b2)], it returns two collections: [a1, a2] and [b1, b2]. A port of lodash.unzip. | 4 | 4 | 20 |
| **The Patronus Charm** | A reject(predicate) function for iterators that is the opposite of filter. It returns a new collection containing only the elements for which the predicate returns false. A port of lodash.reject. | 4 | 5 | 20 |

### **The Hacker's Delight Spellbook (Bit Manipulation & Low-Level Tricks)**

This section focuses on algorithms that operate at the bit and byte level, drawing heavily from the canonical text "Hacker's Delight".10 These functions are ideal for Rust, which can provide safe, high-performance, and portable abstractions over low-level CPU intrinsics. A key value proposition for these crates is encapsulating the complexity of platform-specific

unsafe code (e.g., using std::arch for SIMD or population count instructions) behind a simple, safe, and tested API that falls back to a pure Rust implementation when intrinsics are unavailable.21

| Crate Name (Harry Potter Themed) | Description & Reasoning | Probability of PMF (1-5) | Ease of Success-Testing (1-5) | Relevant Artifacts (Links) |
| --- | --- | --- | --- | --- |
| **Lumos Maxima** | A function nlz(n) (Number of Leading Zeros) or clz for all integer primitives. This is a fundamental operation for integer logarithms and floating-point manipulation. The crate would use the leading\_zeros intrinsic and provide a consistent API. | 5 | 5 | 21 |
| **Nox** | A function ntz(n) (Number of Trailing Zeros) or ctz for all integer primitives. The counterpart to nlz, useful for finding the lowest set bit. It would use the trailing\_zeros intrinsic. | 5 | 5 | 23 |
| **Geminio** | A function popcount(n) (Population Count) that counts the number of set (1) bits in an integer. This is extremely useful in cryptography, error-correcting codes, and bioinformatics. It would use the popcnt CPU intrinsic where available. | 5 | 5 | 23 |
| **Finite Incantatem** | A function is\_power\_of\_two(n) that uses the bitwise trick n > 0 && (n & (n - 1) == 0) to determine if a number is a power of two. This is far more efficient than loops or logarithms. | 5 | 5 | 21 |
| **Wingardium Leviosa** | A function next\_power\_of\_two(n) that finds the smallest power of two greater than or equal to n. A common requirement in graphics and memory allocation for aligning data to power-of-two boundaries. | 4 | 5 | 23 |
| **Reducto** | A function floor\_log2(n) that computes floor(log2(n)) for an integer n using nlz. This is an O(1) operation on modern CPUs, vastly outperforming floating-point conversions. | 4 | 5 | 23 |
| **Engorgio** | A function ceil\_log2(n) that computes ceil(log2(n)) for an integer n, also using bit manipulation. | 4 | 5 | 23 |
| **Impervius** | A function parity(n) that determines if the number of set bits in n is even or odd. Can be implemented efficiently with XOR folding or by checking the last bit of popcount(n). Used in error checking. | 3 | 5 | 23 |
| **Confundo** | A function bit\_reverse(n) that reverses the order of bits in an integer. For 0b11010000, it would return 0b00001011. Essential for some FFT algorithms and cryptographic routines. Would use the bitreverse intrinsic. | 4 | 5 | 23 |
| **Muffliato** | A function byte\_swap(n) that reverses the byte order (endianness) of an integer. For 0x12345678, it would return 0x78563412. Would use the bswap intrinsic. | 5 | 5 | 23 |
| **Sectumsempra** | A function extract\_bits(n, pos, len) that extracts a field of len bits starting at bit pos from n. Can be implemented with shifts and masks. | 4 | 5 | 21 |
| **Episkey** | A function deposit\_bits(n, field, pos, len) that deposits the bits from field into n at bit pos with a mask of len. The inverse of extract\_bits. | 4 | 5 | 21 |
| **Oppugno** | A function pdep(source, mask) (Parallel Deposit), which scatters bits from source to the positions indicated by set bits in mask. A powerful permutation primitive available in modern CPUs (BMI2). | 3 | 4 | 21 |
| **Carpe Retractum** | A function pext(source, mask) (Parallel Extract), which gathers bits from source at the positions indicated by set bits in mask and packs them into the low-order bits of the result. The inverse of pdep. | 3 | 4 | 21 |
| **The Unbreakable Vow** | A function sign\_extend(x, b) that extends the sign of an integer x from b bits to the full width of the type. For example, extending a 4-bit signed -1 (0b1111) to 8 bits (0b11111111). | 3 | 5 | 23 |
| **The Dark Mark** | A function isolate\_rightmost\_one(n) which returns a number with only the lowest set bit of n remaining. The trick is n & -n. Useful in Fenwick trees and other algorithms. | 4 | 5 | 24 |
| **Priori Incantatem** | A function clear\_rightmost\_one(n) which clears the lowest set bit of n. The trick is n & (n - 1). Used in Brian Kernighan's popcount algorithm. | 4 | 5 | 25 |
| **The Trace** | A function average\_floor(a, b) that computes (a + b) / 2 without overflowing. The trick is (a & b) + ((a ^ b) >> 1). | 3 | 5 | 23 |
| **The Fidelius Charm** | A collection of functions for division by a known constant using multiplication and shifts (magic numbers). For a given constant D, pre-calculates a "magic number" M and a shift s such that (n \* M) >> s equals n / D. Extremely fast when the divisor is fixed. | 3 | 4 | 23 |
| **The Vanishing Cabinet** | An implementation of the "compress" or "generalized extract" operation, which packs selected bits from a word to one side. For example, compressing 0b10110101 with mask 0b11001100 might yield 0b00001001. | 2 | 4 | 23 |
| **The Hand of Glory** | An implementation of the "expand" or "generalized insert" operation, the inverse of compress. It unpacks bits from one side of a word into positions specified by a mask. | 2 | 4 | 23 |
| **The Gray Code Grimoire** | Functions to convert an integer to its Gray code representation (n ^ (n >> 1)) and back. Gray codes are useful in error correction and state machines, as only one bit changes between consecutive values. | 3 | 5 | 23 |
| **The Hilbert Curve Hex** | Functions to convert 2D coordinates (x, y) to a distance d along a Hilbert space-filling curve, and back. This preserves locality, mapping nearby 2D points to nearby 1D distances. Useful in spatial indexing and databases. | 2 | 2 | 27 |
| **The Hamming Code Charm** | Functions to encode data with a simple Hamming code for single-error correction and double-error detection (SEC-DED). A classic algorithm from information theory. | 2 | 3 | 10 |
| **The CRC Spellbook** | A no\_std library for calculating various Cyclic Redundancy Check (CRC) values (e.g., CRC-32, CRC-64). While crc crates exist, a minimal, no\_std, zero-dependency implementation focused on the core bitwise algorithms would be valuable for embedded systems. | 3 | 4 | 10 |

### **The Restricted Section of the Library (Algorithms from CS Research)**

This section translates theoretical computer science concepts into practical, high-value Rust crates. These are areas where Rust's performance and memory model are not just beneficial but essential. The academic world for these algorithms is dominated by C++ implementations, which are often complex and memory-unsafe. Rust offers a unique opportunity to create next-generation implementations that are just as fast but are also safe, ergonomic, and easy to integrate via Cargo.28

| Crate Name (Harry Potter Themed) | Description & Reasoning | Probability of PMF (1-5) | Ease of Success-Testing (1-5) | Relevant Artifacts (Links) |
| --- | --- | --- | --- | --- |
| **Hermione's Beaded Bag** | A RankSelectBitvector struct. This is a bit vector that supports O(1) rank (count bits up to an index) and select (find the index of the k-th bit) queries. It is the fundamental building block of nearly all modern succinct data structures and is a glaring omission in the Rust ecosystem. | 4 | 3 | 30 |
| **The Prophet's Orb** | A WaveletTree struct. A compact data structure that represents a sequence of symbols and supports rank, select, and access queries in O(log σ) time, where σ is the alphabet size. It has major applications in text processing, bioinformatics, and computational geometry. | 3 | 2 | 28 |
| **The Hogwarts Castle** | A K2Tree struct. A compact representation of a cardinal tree (and by extension, quadtrees or sparse grids) using bit vectors. It offers significant space savings for clustered data and is used in graph databases and spatial indexing. | 2 | 2 | 30 |
| **Protego Totalum** | A function convex\_hull(points) that implements an efficient 2D convex hull algorithm, such as Graham Scan or Chan's Algorithm. This is one of the most fundamental algorithms in computational geometry. | 4 | 2 | 31 |
| **The Four-Point Spell** | A function point\_in\_polygon(point, polygon) that implements a robust point-in-polygon test, likely using the winding number or ray casting algorithm. A core primitive for geospatial and graphics applications. | 5 | 2 | 29 |
| **The Spider's Web** | An implementation of the Bentley-Ottmann algorithm, a plane-sweep algorithm for finding all intersection points among a set of line segments. It is an output-sensitive algorithm, making it efficient when intersections are sparse. | 3 | 2 | 29 |
| **The Delaunay Charm** | A function delaunay\_triangulation(points) that implements the randomized incremental Delaunay triangulation algorithm. This structure is dual to the Voronoi diagram and is fundamental in mesh generation, interpolation, and surface reconstruction. | 3 | 2 | 31 |
| **The Voronoi Diagram** | A function voronoi\_diagram(points) that computes the Voronoi diagram for a set of points, likely using Fortune's plane-sweep algorithm. Each region contains points closer to one site than any other. | 3 | 1 | 31 |
| **The Half-Blood Prince's Potion Book** | A function half\_plane\_intersection(planes) that computes the intersection of a set of half-planes, resulting in a convex polygon. This is the dual problem to convex hull and is used in linear programming and geometric modeling. | 3 | 2 | 32 |
| **The Liquid Luck** | A function linear\_program\_2d(constraints, objective) that solves a 2D linear programming problem in O(n) expected time using a randomized incremental algorithm. | 3 | 3 | 32 |
| **The Parseltongue** | An implementation of the Needleman-Wunsch algorithm for optimal global sequence alignment. A cornerstone of bioinformatics, it uses dynamic programming to align two DNA or protein sequences from end to end. | 3 | 3 | 33 |
| **The Serpent's Fang** | An implementation of the Smith-Waterman algorithm for optimal local sequence alignment. Unlike Needleman-Wunsch, it finds the most similar sub-regions between two sequences, making it better for divergent sequences. | 3 | 3 | 33 |
| **The Memory Charm** | A cache-oblivious matrix transposition algorithm. This recursive algorithm partitions the matrix into sub-problems, achieving near-optimal cache performance on any memory hierarchy without needing to be tuned with cache-size parameters. | 2 | 3 | 36 |
| **The Sorting Spell** | A cache-oblivious sorting algorithm, like the Funnelsort or the cache-oblivious distribution sort. These algorithms are designed to be asymptotically optimal in terms of cache misses on a multi-level memory hierarchy. | 2 | 2 | 36 |
| **The Fountain of Fair Fortune** | A streaming algorithm for Count-Min Sketch. This is a probabilistic data structure that serves as a frequency table of events in a stream of data. It uses a sub-linear amount of space at the cost of a small overestimation error. | 3 | 3 | 39 |
| **The Well of Wondrous Wit** | A streaming algorithm for HyperLogLog. This is another probabilistic algorithm for the count-distinct problem, which approximates the number of distinct elements in a multiset. It is extremely space-efficient. | 3 | 3 | 39 |
| **The Tale of the Three Brothers** | An implementation of Hirschberg's algorithm for sequence alignment in linear space. It is a divide-and-conquer optimization of the Needleman-Wunsch algorithm, crucial for aligning very long sequences where quadratic space is prohibitive. | 2 | 2 | 40 |
| **The Doubly-Connected Edge List** | A robust implementation of the Doubly-Connected Edge List (DCEL) or Half-edge data structure. This is not an algorithm itself, but a fundamental data structure for representing planar subdivisions, required by many other geometry algorithms. A safe, ergonomic Rust implementation would be a huge boon. | 4 | 2 | 42 |
| **The Rotating Calipers** | An algorithm that, given a convex polygon, can find properties like its diameter, width, or minimum-area bounding box in linear time. It works by "rotating" a pair of parallel supporting lines around the polygon. | 2 | 2 | 31 |
| **The Shoelace Formula** | A simple function to calculate the area of a polygon given its vertices' coordinates using the shoelace (or surveyor's) formula. Mathematically trivial but a very common need. | 5 | 4 | 29 |
| **The Gift-Wrapping Algorithm** | An implementation of the Jarvis march (gift wrapping) algorithm for finding the convex hull. While asymptotically slower (O(nh), where h is the number of hull vertices) than Graham Scan, it is simpler to implement and can be faster for small h. | 3 | 3 | 29 |
| **The Quickhull Algorithm** | An implementation of the Quickhull algorithm for finding the convex hull. It uses a divide-and-conquer approach analogous to Quicksort and has an average time complexity of O(n log n). | 3 | 2 | 29 |
| **The Nussinov Algorithm** | A dynamic programming algorithm for predicting the secondary structure of an RNA molecule by finding the maximum number of base pairs. A classic bioinformatics algorithm. | 2 | 2 | 40 |
| **The Baum-Welch Algorithm** | An implementation of the Baum-Welch algorithm, which uses the expectation-maximization algorithm to find the unknown parameters of a Hidden Markov Model (HMM). Essential for gene prediction and sequence analysis. | 2 | 1 | 40 |
| **The Ukkonen's Algorithm** | An online algorithm for constructing a suffix tree in linear time. Suffix trees are powerful data structures for string searching and related problems. | 2 | 1 | 40 |

### **The Potions Cabinet (Numerical & Statistical Methods)**

This section proposes small, single-purpose crates for common numerical and statistical tasks. Large scientific computing libraries like Python's scipy bundle hundreds of functions into a single, monolithic dependency. The Rust ecosystem, however, favors a more "unbundled" approach with smaller, focused libraries that have minimal dependency trees. This presents a clear opportunity: create a micro-crate for each major statistical test, allowing users to select only the specific "potion" they need for their analysis.43

| Crate Name (Harry Potter Themed) | Description & Reasoning | Probability of PMF (1-5) | Ease of Success-Testing (1-5) | Relevant Artifacts (Links) |
| --- | --- | --- | --- | --- |
| **Veritaserum** | A function for the one-sample and two-sample Student's t-test. This is one of the most fundamental statistical tests for comparing the means of one or two groups. | 5 | 3 | 43 |
| **The Draught of Peace** | A function for Welch's t-test, an adaptation of the t-test that is more reliable when the two samples have unequal variances. | 4 | 3 | 45 |
| **Felix Felicis** | A function to calculate the Pearson correlation coefficient between two sets of data, measuring the linear relationship between them. | 5 | 4 | 43 |
| **The Polyjuice Potion** | A function to calculate Spearman's rank correlation coefficient. This is a non-parametric measure of rank correlation, assessing how well the relationship between two variables can be described using a monotonic function. | 4 | 3 | 47 |
| **The Draught of Living Death** | A function for the Mann-Whitney U test (or Wilcoxon rank-sum test). A non-parametric test used to determine whether two independent samples were selected from populations having the same distribution. | 4 | 3 | 43 |
| **The Elixir of Life** | A function for the Wilcoxon signed-rank test. A non-parametric test used for paired samples to assess whether their population mean ranks differ. | 4 | 3 | 47 |
| **The Wolfsbane Potion** | A function for the Chi-squared test, used for goodness of fit or to test the independence of two categorical variables in a contingency table. | 5 | 3 | 43 |
| **The Babbling Beverage** | A function for the Kruskal-Wallis H test. A non-parametric test for determining if there are statistically significant differences between two or more groups of an independent variable on a continuous or ordinal dependent variable. It is the non-parametric equivalent of ANOVA. | 3 | 3 | 47 |
| **The Shrinking Solution** | A function for Fisher's exact test, used for analyzing contingency tables when sample sizes are small. | 3 | 3 | 45 |
| **The Befuddlement Draught** | A one-way Analysis of Variance (ANOVA) function to compare the means of three or more groups. | 4 | 2 | 43 |
| **The Levene Test** | A function for Levene's test, an inferential statistic used to assess the equality of variances for a variable calculated for two or more groups. It is a prerequisite for some comparison tests like ANOVA. | 3 | 3 | 45 |
| **The Shapiro-Wilk Test** | A function for the Shapiro-Wilk test, a test of normality in frequentist statistics. It is a common prerequisite for using parametric tests. | 3 | 3 | 48 |
| **The G-Test of Goodness-of-Fit** | An implementation of the G-test, which is a likelihood-ratio or maximum likelihood statistical significance test that is increasingly used in situations where chi-squared tests were previously recommended. | 2 | 3 | 45 |
| **The Kolmogorov-Smirnov Test** | A function for the one-sample and two-sample Kolmogorov-Smirnov (K-S) test. It is a non-parametric test of the equality of continuous, one-dimensional probability distributions. | 3 | 2 | 45 |
| **The Binomial Test** | A function for the binomial test of statistical significance. It is used when an experiment has two possible outcomes (success and failure) and you have an idea about what the probability of success is. | 4 | 4 | 45 |
| **The F-Test of Equality of Variances** | A function to perform an F-test to check if two population variances are equal. The test statistic is the ratio of the two sample variances. | 3 | 3 | 45 |
| **The Kendall's Tau** | A function to calculate Kendall's rank correlation coefficient (Tau), which is a statistic used to measure the ordinal association between two measured quantities. | 3 | 3 | 49 |
| **The Confusion Concoction** | A small library to generate confusion matrices and calculate related metrics (accuracy, precision, recall, F1-score) from predicted and actual labels. While more ML-oriented, the core calculations are simple and universally needed. | 5 | 5 | 50 |
| **The Root of Mandrake** | An implementation of the integer square root algorithm, which computes floor(sqrt(n)) for an integer n without using floating-point arithmetic. Can be done efficiently with bit manipulation. | 3 | 5 | 23 |
| **The Bezoar** | A small crate for robustly calculating basic descriptive statistics (mean, median, mode, variance, standard deviation) on an iterator of numbers, with careful handling of floating-point precision issues. | 5 | 3 | 47 |

#### Works cited

1. rust-unofficial/awesome-rust: A curated list of Rust code and resources. - GitHub, accessed on August 15, 2025, <https://github.com/rust-unofficial/awesome-rust>
2. Awesome Rust - GitHub, accessed on August 15, 2025, <https://github.com/awesome-rust-com/awesome-rust>
3. Categories - crates.io: Rust Package Registry, accessed on August 15, 2025, <https://crates.io/categories>
4. Category Slugs - Rust Package Registry - Crates.io, accessed on August 15, 2025, <https://crates.io/category_slugs>
5. What are itertools in Python? - Educative.io, accessed on August 15, 2025, <https://www.educative.io/answers/what-are-itertools-in-python>
6. Python's collections: A Buffet of Specialized Data Types - Real Python, accessed on August 15, 2025, <https://realpython.com/python-collections-module/>
7. slices package - slices - Go Packages, accessed on August 15, 2025, <https://pkg.go.dev/slices>
8. Lodash Quick Guide - Tutorialspoint, accessed on August 15, 2025, <https://www.tutorialspoint.com/lodash/lodash_quick_guide.htm>
9. itertools — Functions creating iterators for efficient looping — Python 3.10.17 documentation, accessed on August 15, 2025, <https://docs.python.org/3.10/library/itertools.html?highlight=pairwise>
10. Hackers Delight 2nd Edition | Powell's Books, accessed on August 15, 2025, <https://www.powells.com/book/hackers-delight-2nd-edition-9780321842688>
11. Compact Data Structures: A Practical Approach, accessed on August 15, 2025, <https://api.pageplace.de/preview/DT0400.9781316792926_A27740717/preview-9781316792926_A27740717.pdf>
12. Python Itertools Module, accessed on August 15, 2025, <https://www.pythoncheatsheet.org/modules/itertools-module>
13. itertools — Functions creating iterators for efficient looping — Python ..., accessed on August 15, 2025, <https://docs.python.org/3/library/itertools.html>
14. Python Collections Module - GeeksforGeeks, accessed on August 15, 2025, <https://www.geeksforgeeks.org/python/python-collections-module/>
15. collections — Container datatypes — Python 3.13.7 documentation, accessed on August 15, 2025, <https://docs.python.org/3/library/collections.html>
16. 10 Most Useful Itertools Methods (63/100 Days of Python) | by Martin Mirakyan - Medium, accessed on August 15, 2025, <https://martinxpn.medium.com/10-most-useful-itertools-methods-63-100-days-of-python-b7e7a93639c3>
17. More Itertools - more-itertools 10.7.0 documentation, accessed on August 15, 2025, <https://more-itertools.readthedocs.io/>
18. slices package - golang.org/x/exp/slices - Go Packages, accessed on August 15, 2025, <https://pkg.go.dev/golang.org/x/exp/slices>
19. Lodash Documentation, accessed on August 15, 2025, <https://lodash.com/docs/4.17.15>
20. Bit manipulation - Wikipedia, accessed on August 15, 2025, <https://en.wikipedia.org/wiki/Bit_manipulation>
21. Bitwise Algorithms - GeeksforGeeks, accessed on August 15, 2025, <https://www.geeksforgeeks.org/dsa/bitwise-algorithms/>
22. Hacker's Delight by Henry Warren, Hardcover | Barnes & Noble®, accessed on August 15, 2025, <https://www.barnesandnoble.com/w/hackers-delight-henry-warren/1117355434>
23. Basics of Bit Manipulation Tutorials & Notes | Basic Programming - HackerEarth, accessed on August 15, 2025, <https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/tutorial/>
24. Bit manipulation - Algorithms for Competitive Programming, accessed on August 15, 2025, <https://cp-algorithms.com/algebra/bit-manipulation.html>
25. The basics of bit manipulation - Educative.io, accessed on August 15, 2025, <https://www.educative.io/blog/bit-manipulation-algorithms>
26. Hacker's Delight, accessed on August 15, 2025, <https://www.pearson.de/media/muster/toc/toc_9780133085006.pdf>
27. Compact Data Structures Meet Databases - DROPS, accessed on August 15, 2025, <https://drops.dagstuhl.de/storage/00lipics/lipics-vol255-icdt2023/LIPIcs.ICDT.2023.2/LIPIcs.ICDT.2023.2.pdf>
28. Computational geometry - Wikipedia, accessed on August 15, 2025, <https://en.wikipedia.org/wiki/Computational_geometry>
29. Compact Data Structures Meet Databases - DCC UChile, accessed on August 15, 2025, <https://users.dcc.uchile.cl/~gnavarro/ps/icdt23.pdf>
30. Computational Geometry - Algorithms for Geometry - GeeksforGeeks, accessed on August 15, 2025, <https://www.geeksforgeeks.org/dsa/what-is-computational-geometry-and-how-is-it-applied-in-solving-geometric-problems/>
31. Computational Geometry Algorithms and Applications, accessed on August 15, 2025, <https://www.cs.cmu.edu/afs/cs/academic/class/15456-s14/Handouts/BKOS.pdf>
32. Sequence alignment - Wikipedia, accessed on August 15, 2025, <https://en.wikipedia.org/wiki/Sequence_alignment>
33. Sequence Alignment- Definition, Types, Methods, Uses, accessed on August 15, 2025, <https://microbenotes.com/local-global-multiple-sequence-alignment/>
34. Job Dispatcher < EMBL-EBI, accessed on August 15, 2025, <https://www.ebi.ac.uk/jdispatcher/psa>
35. (PDF) Cache-oblivious algorithms - ResearchGate, accessed on August 15, 2025, <https://www.researchgate.net/publication/3832223_Cache-oblivious_algorithms>
36. Cache-Oblivious Algorithms, accessed on August 15, 2025, <https://ocw.mit.edu/courses/6-895-theory-of-parallel-systems-sma-5509-fall-2003/6dc7de52dcf13b53cebf2fe10ae6752a_cach_oblvs_thsis.pdf>
37. Cache-Oblivious and Cache-Aware Algorithms - DROPS - Schloss Dagstuhl, accessed on August 15, 2025, <https://drops.dagstuhl.de/storage/16dagstuhl-seminar-proceedings/dsp-vol04301/DagSemProc.04301.1/DagSemProc.04301.1.pdf>
38. Graph Stream Algorithms: A Survey∗, accessed on August 15, 2025, <https://people.cs.umass.edu/~mcgregor/papers/13-graphsurvey.pdf>
39. Category:Bioinformatics algorithms - Wikipedia, accessed on August 15, 2025, <https://en.wikipedia.org/wiki/Category:Bioinformatics_algorithms>
40. Bioinformatic Algorithms - Department of Computer Science and Technology |, accessed on August 15, 2025, <https://www.cl.cam.ac.uk/~pl219/Bioinformatics2015.pdf>
41. Computational Geometry - Algorithms and Applications, accessed on August 15, 2025, <https://cimec.org.ar/foswiki/pub/Main/Cimec/GeometriaComputacional/DeBerg_-_Computational_Geometry_-_Algorithms_and_Applications_2e.pdf>
42. Commonly Used Statistical Tests in Data Science - StrataScratch, accessed on August 15, 2025, <https://www.stratascratch.com/blog/commonly-used-statistical-tests-in-data-science/>
43. Essential Statistical Tests for Data Scientists - DASCA certification, accessed on August 15, 2025, <https://www.dasca.org/world-of-data-science/article/essential-statistical-tests-for-data-scientists>
44. Category:Statistical tests - Wikipedia, accessed on August 15, 2025, <https://en.wikipedia.org/wiki/Category:Statistical_tests>
45. The three most common statistical tests you should deeply understand - APA Divisions, accessed on August 15, 2025, <https://www.apadivisions.org/division-5/publications/score/2024/10/common-statistical-tests>
46. Choosing the Right Statistical Test | Types & Examples - Scribbr, accessed on August 15, 2025, <https://www.scribbr.com/statistics/statistical-tests/>
47. 18 Simple statistical tests - The Epidemiologist R Handbook, accessed on August 15, 2025, <https://epirhandbook.com/en/new_pages/stat_tests.html>
48. List of Statistical Procedures - NCSS, accessed on August 15, 2025, <https://www.ncss.com/software/ncss/procedures/>
49. Awesome Rust Mentors - github, accessed on August 15, 2025, <https://rustbeginners.github.io/awesome-rust-mentors/>