Scale Stability and Rank Drift in Consecutive Prime-Gap Frequencies

An Exploratory Computational Note

Dieter Eigler

August 2026

Computational disclosure. All numerical results in this note were generated by explicit prime sieving and gap counting. Early exploratory work used Python/NumPy. The largest shell was counted with an optimized compiled implementation of the same odd-only segmented Eratosthenes procedure; the standalone Python/NumPy listing in Appendix B is an exact reference implementation of that procedure and is intended for transparent independent reproduction and validation. AI systems assisted with exploratory analysis, code development, visualization, and manuscript preparation. The note is descriptive and does not claim a new theorem about prime gaps.

Abstract

This note studies the frequency ranking of gaps between consecutive primes under changes of numerical scale. Instead of considering only the usual cumulative count up to x, the computation separates the primes into disjoint decade shells and asks whether the ranking of the most frequent gaps becomes locally stable. Across the tested shells from 10^3–10^4 through 10^9–10^{10}, the same top-ten candidate set recurs across the larger shells before the exact rank order becomes comparably persistent. In the shell 10^9–10^{10} the leading sequence is 6, 12, 18, 10, 2, 4, 8, 24, 30, 14. A further computation over 10^{10}–10^{11} contains 3,663,002,302 gaps and gives 6, 12, 18, 10, 2, 4, 30, 24, 8, 14. Dividing this shell into ten equal-width subintervals shows the top four gaps 6, 12, 18, 10 in exactly the same order in every subinterval, while lower ranks still exchange places. Relative-frequency ratios to gap 6 reveal a second phenomenon: some ratios are nearly stationary, while 18, 24, and especially 30 continue to gain relative weight. These results are presented as empirical scale-stability and rank-drift observations. They are consistent with, but do not attempt to rederive, the established heuristic framework surrounding Hardy–Littlewood singular series and jumping champions.

Keywords: prime gaps; consecutive primes; jumping champions; rank stability; logarithmic shells; computational number theory

1. Introduction

Let p_n denote the nth prime and $g_n = p_{n+1} - p_n$ the corresponding consecutive prime gap. A classical way to summarize gap frequencies is cumulative: for a bound x, one counts how often each d occurs among consecutive primes not exceeding x and asks which d is most frequent. The most frequent d is called a jumping champion. The terminology was popularized by Conway, and the modern heuristic and computational framework was developed by Odlyzko, Rubinstein, and Wolf (1999).

The present note asks a slightly different empirical question. Cumulative counts carry a strong memory of all smaller scales. If one wants to know whether the local hierarchy of common gaps itself becomes stable as the scale increases, it is natural to remove that memory and count gaps in disjoint shells. The central observable here is therefore not only the cumulative champion but the ordered list of the most frequent gaps inside a prescribed interval of starting primes.

This distinction matters. A fixed gap can remain the cumulative champion even while the relative ordering of several other gaps changes locally. Conversely, a stable local ranking over several large shells may indicate a form of empirical scale regularity without implying any asymptotic theorem. The aim is deliberately narrow: to document how much of the rank ordering stabilizes, how much continues to drift, and which additional numerical patterns accompany that transition.

2. Background: prime-pair heuristics and jumping champions

Hardy and Littlewood (1923) formulated prime-pair and prime-tuple heuristics in terms of a singular series. For a fixed even difference d, the singular series rewards divisibility of d by small odd primes; in particular, the factors entering the singular series are larger when d contains more small prime divisors. This mechanism is one reason primorial gaps are prominent in the jumping-champion literature. For even d, write

$$\mathfrak{S}(d) = 2\,C_2 \prod_{p \mid d,\, p>2} (p-1)/(p-2), \quad C_2 = \prod_{p>2} (1 - 1/(p-1)^2) = 0.66016\ldots$$

Odlyzko, Rubinstein, and Wolf (1999) gave heuristic and computational evidence for the conjecture that, beyond the exceptional value 4, the jumping champions are the primorials 2, 6, 30, 210, 2310, … . Earlier, Erdős and Straus (1980) proved, conditional on the Hardy–Littlewood prime-pair conjecture, that jumping champions tend to infinity. Goldston and Ledoan (2011) extended this method to show, under the same prime-pair assumption, that every fixed prime eventually divides all sufficiently large jumping champions. Goldston and Ledoan (2015) then showed, under appropriate Hardy–Littlewood pair-and-triple assumptions, that sufficiently large jumping champions are primorials and that sufficiently large primorials serve as jumping champions over long ranges.

The present computations do not test those asymptotic results directly. The interval up to 10^{11} is tiny compared with the conjectural transition scale for the cumulative champion from 6 to 30: Goldston and Ledoan (2015), summarizing earlier computations and the heuristics of Odlyzko, Rubinstein, and Wolf, report estimates near 1.7×10^{35} and 1.7427×10^{35} respectively. The contribution here is instead organizational and empirical: gap frequencies are ranked inside disjoint shells, allowing local finite-range rank stability and local rank drift to be inspected separately from cumulative dominance.

3. Definitions and computational protocol

For consecutive primes $p_n < p_{n+1}$, define $g_n = p_{n+1} - p_n$. For an interval I = [A,B), define the shell count

$$N\_I(d) = \#\{\, n : A \le p_n < B \text{ and } p_{n+1} - p_n = d \,\}.$$

A gap is assigned to the interval containing its starting prime p_n. Thus a gap crossing the upper boundary B is still counted in I if $p_n < B$. The shell ranking R_I is obtained by ordering gap values d by decreasing N_I(d), with smaller d used only as a deterministic tie-breaker when counts are exactly equal.

The computations use exact prime generation by sieving. Small ranges were checked with direct and segmented Eratosthenes sieves. For the large shell 10^{10}–10^{11}, a segmented sieve counts gap frequencies without storing all primes or all gaps simultaneously. The full shell contains 3,663,002,302 consecutive-prime gaps; the first prime in the shell is 10,000,000,019 and the last prime below 10^11 is 99,999,999,977. The next prime is 100,000,000,003, so the final boundary-crossing gap is included under the stated convention.

No probabilistic model or significance test is imposed on the rankings. The counts are exact and deterministic; nevertheless, finite-range fluctuations may remain. The relevant distinction is therefore descriptive: exact stability of ranks over the tested intervals, stability of the set of leading candidates over the tested intervals, and systematic changes in relative frequencies. The decade-shell design, the ten-way subdivision, and the selected ratio diagnostics arose during exploratory analysis and are not preregistered confirmatory specifications.

4. Rankings in disjoint decade shells

Table 1 reports the ten most frequent gaps in seven disjoint decade shells. The ranking changes markedly at first, then becomes increasingly constrained. Gap 6 is ranked first in every shell examined. Gap 12 reaches rank 2 by 10^5–10^6 and remains there through 10^9–10^{10}. Gap 18 rises more slowly and reaches rank 3 in the final shell of this table.

Shell	Gaps	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
10^3–10^4	1,061	6	2	4	10	12	8	14	18	16	22
10^4–10^5	8,363	6	2	4	12	10	8	18	14	16	20

Shell	Gaps	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
10^5–10^6	68,906	6	12	2	4	10	8	18	14	16	24
10^6–10^7	586,081	6	12	2	4	10	18	8	14	24	16
10^7–10^8	5,096,876	6	12	4	2	10	18	8	14	24	30
10^8–10^9	45,086,079	6	12	10	4	2	18	8	14	24	30
10^9–10^10	404,204,976	6	12	18	10	2	4	8	24	30	14

Table 1. Frequency ranking of consecutive prime gaps in disjoint decade shells. Gaps are assigned by the location of the starting prime.

A useful separation emerges. The exact order continues to change, but the candidate set becomes unchanged across the tested shells earlier. From 10^7–10^8 onward, the same top-ten set occurs in every tested decade: {2, 4, 6, 8, 10, 12, 14, 18, 24, 30}. Within that finite-range set stability, however, rank drift remains visible. An additional empirical observation within the present computation is therefore that candidate-set stability over the tested range precedes order stability over the tested range.

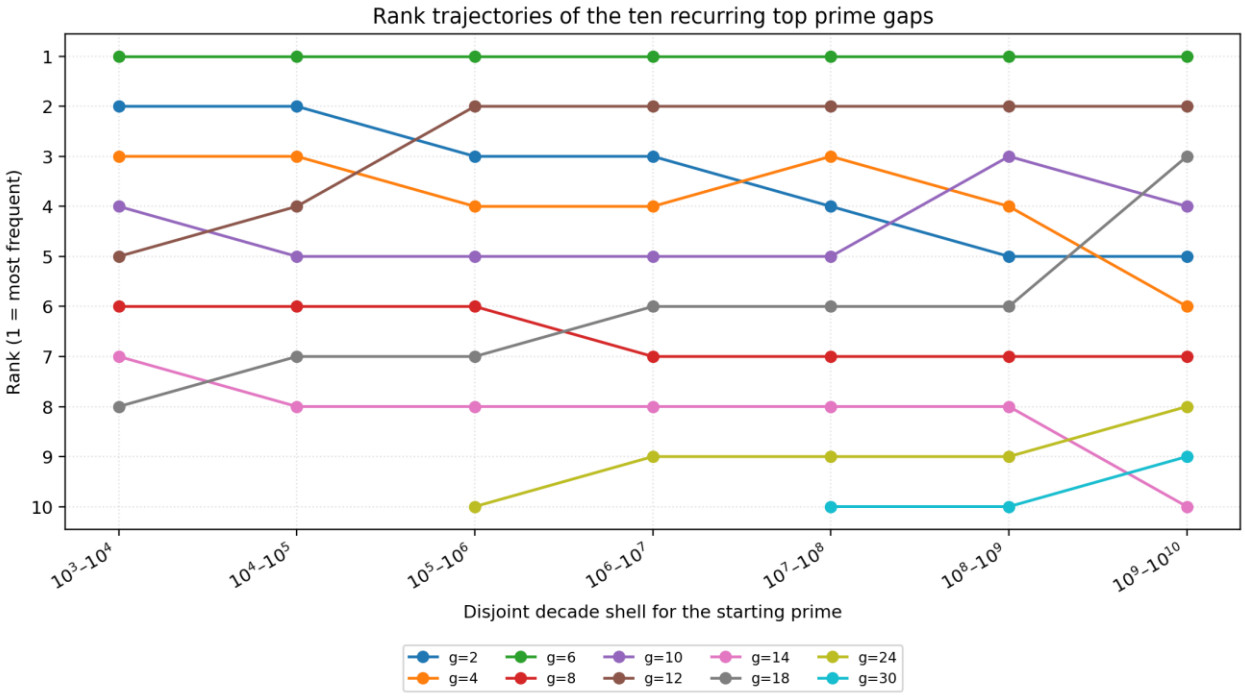

Figure 1. Rank trajectories of the ten gap values that constitute the recurring top-ten set. Lower numerical rank means greater frequency; a gap is omitted in shells where it lies outside that shell's top ten.

5. Extension to the shell 10^{10}–10^{11}

The next disjoint shell was computed separately to test whether the ordering seen at 10^9–10^{10} persists. The shell contains 3,663,002,302 gaps. Its leading fifteen frequencies are shown in Table 2.

Rank	Gap d	Frequency N(d)
1	6	359,157,555
2	12	282,384,248

Rank	Gap d	Frequency N(d)
3	18	226,182,006
4	10	212,534,126
5	2	196,963,369
6	4	196,963,162
7	30	173,315,067
8	24	172,715,371
9	8	163,241,302
10	14	158,774,047
11	20	129,061,729
12	16	121,826,825
13	22	110,105,362
14	36	100,524,519
15	28	92,290,521

Table 2. Top fifteen consecutive-prime gaps in the disjoint shell 10^{10}–10^{11}.

The top six positions are unchanged from the preceding shell 10^9–10^{10}: 6, 12, 18, 10, 2, 4. Below them, the order is still moving: gap 30 rises above 24 and 8. A particularly striking finite-range near-tie occurs between gaps 2 and 4. Their shell counts are 196,963,369 and 196,963,162 respectively, a difference of only 207 out of roughly 197 million occurrences each.

This additional empirical observation sharpens the meaning of “stability.” A long stable prefix over adjacent finite shells can coexist with unresolved lower-rank competition. The empirical hierarchy at this scale is therefore neither fully frozen nor freely fluctuating.

6. Ten-way subdivision of 10^{10}–10^{11}

To check whether the shell-wide ordering is an artifact of pooling a very broad interval, 10^{10}–10^{11} was divided into ten equal-width subintervals, each of width 9×10^9. Each subinterval contains approximately 356–385 million gaps. Table 3 lists the top ten in each subinterval.

Part	Interval (×10^9)	Gaps (M)	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
1	10–19	384.948	6	12	18	10	2	4	24	30	8	14
2	19–28	376.988	6	12	18	10	4	2	24	30	8	14
3	28–37	371.885	6	12	18	10	4	2	24	30	8	14
4	37–46	368.145	6	12	18	10	2	4	30	24	8	14
5	46–55	365.205	6	12	18	10	2	4	30	24	8	14
6	55–64	362.777	6	12	18	10	4	2	30	24	8	14
7	64–73	360.730	6	12	18	10	4	2	30	24	8	14
8	73–82	358.953	6	12	18	10	2	4	30	24	8	14
9	82–91	357.384	6	12	18	10	2	4	30	24	8	14
10	91–100	355.988	6	12	18	10	4	2	30	24	8	14

Table 3. Top-ten gap ranking in ten equal-width subintervals of 10^{10}–10^{11}.

The result is unusually rigid at the top: every one of the ten subintervals has exactly the same first four gaps, in the same order, 6 > 12 > 18 > 10. Ranks 5 and 6 alternate between 2 and 4. Ranks 7 and 8 transition from 24 > 30 in the first three subintervals to 30 > 24 from the fourth onward. Ranks 9 and 10 are consistently 8 > 14. Thus the full top-ten set is stable throughout all ten subintervals, and six positions are effectively fixed; only two close pairs exchange order.

This additional empirical observation is stronger than decade-by-decade stability. It shows that the leading hierarchy is not merely a consequence of averaging across an order of magnitude: the top four survive a much finer tested partition without a single reversal.

7. Relative-frequency stability and drift

Rank order discards information about how far apart the competitors are. To distinguish true numerical stability from mere rank stability, the frequencies of selected gaps were normalized by N(6) within each of the ten subintervals. The ratios reveal two behaviors.

- Near-stationary ratios: N2/N6, N4/N6, N8/N6, and N10/N6 vary by about one percent or less from the first to the tenth subinterval, except for small local fluctuations.
- Systematic upward drift: N12/N6 rises by about 2.1%, N18/N6 by 4.1%, N24/N6 by 6.7%, and N30/N6 by 9.4% across the same sequence of subintervals.

Ratio	Mean	Minimum	Maximum	Part 1 → 10 trend
N12/N6	0.786468	0.775961	0.792381	+2.12%
N18/N6	0.630112	0.613944	0.639163	+4.11%
N10/N6	0.591840	0.588060	0.593982	+1.01%
N2/N6	0.548344	0.546764	0.551190	-0.80%
N4/N6	0.548342	0.546867	0.551083	-0.76%
N30/N6	0.483168	0.455574	0.498599	+9.44%
N24/N6	0.481323	0.461645	0.492360	+6.65%
N8/N6	0.454560	0.452333	0.455862	+0.78%
N14/N6	0.442236	0.434729	0.446205	+2.64%

Table 4. Stability and drift of selected relative-frequency ratios within 10^{10}–10^{11}.

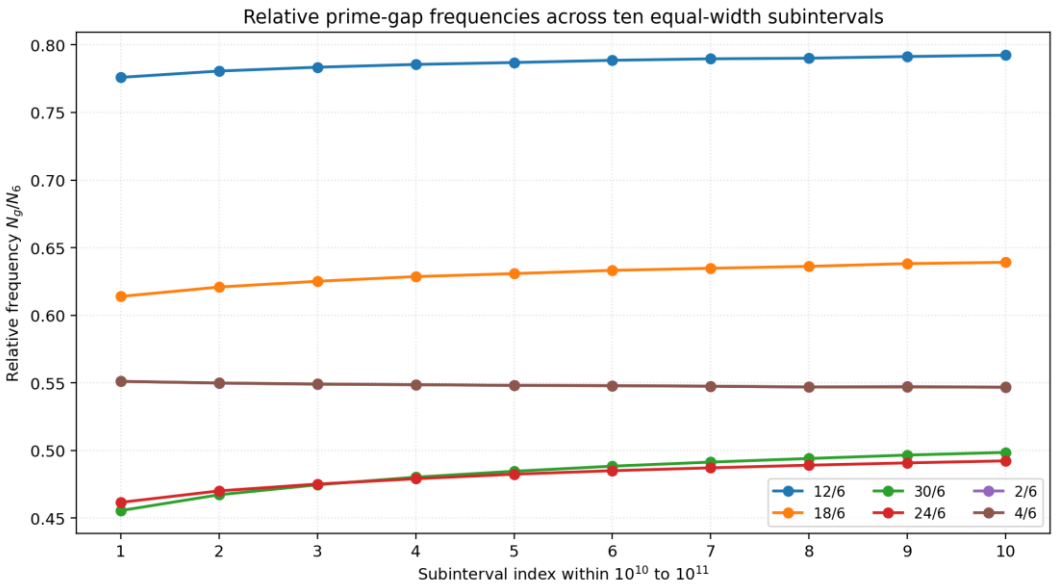

Figure 2. Selected ratios N_g/N_6 across the ten equal-width subintervals. The near coincidence of gaps 2 and 4 and the stronger upward drift of 24 and 30 are visible.

The ratio analysis adds an important qualification to the phrase “scale invariant.” The observed hierarchy has a stable backbone, but it is not fully scale invariant in amplitude. Gap 30, in particular, is gaining relative weight even while gap 6 remains the clear leader. The empirical pattern is therefore better described as stable ordering with structured rank pressure than as a frozen distribution.

8. Relation to the jumping-champion framework

The dominance of 6 and the increasing competitiveness of 30 should not be presented as unexplained anomalies in the broader literature. The Hardy–Littlewood singular series favors even differences divisible by small odd primes, and Odlyzko, Rubinstein, and Wolf explicitly connected this mechanism to the primorial sequence. Goldston and Ledoan developed the corresponding conditional theory. In that sense, the present numerical observations are qualitatively compatible with known heuristics.

At the same time, the specific object studied here is different from the standard cumulative jumping champion. This note ranks all common gaps locally inside disjoint shells and then asks how that local ranking changes with scale. Values such as 12, 18, and 24 can therefore occupy high local ranks without themselves being predicted eventual cumulative champions. Their presence is a feature of the finite local hierarchy, not a contradiction of the primorial jumping-champion conjecture.

The calculation also indicates why the twin-prime gap 2 should not be confused with the most frequent gap. The twin-prime conjecture concerns whether gap 2 occurs infinitely often; frequency dominance is a different question. In the shell 10^{10}–10^{11}, gap 2 is only fifth, while gap 6 occurs approximately 1.82 times as often. This distinction is elementary once stated, but it is visually and numerically striking.

A further, deliberately cautious observation is that gap 2 shows a downward drift in local frequency rank across the tested decade shells: it is second in the two smallest shells, later falls to third and fourth, and is fifth from 10^8–10^9 through 10^{10}–10^{11}. This finite-range tendency does not bear on the twin prime conjecture itself. It does, however, make the distinction particularly visible: a gap may become relatively less dominant among common local gaps while the separate question of whether it occurs infinitely often remains unresolved.

9. Additional exploratory findings

The initial exploration also produced several side observations that are not needed for the central rank-stability claim but help characterize the data.

- Gap 6 is the most frequent gap in every disjoint decade shell examined from 10^3–10^4 through 10^{10}–10^{11}.
- The top-ten candidate set becomes unchanged across the tested shells before the exact order does. From 10^7–10^8 onward the same ten values recur in every tested decade, while 18 and 30 continue to move upward within that set.
- Gaps 2 and 4 behave as an exceptionally close pair at large scales. In 10^{10}–10^{11} their total counts differ by only 207, and their local ranks swap repeatedly across the ten subintervals.
- The ratio N30/N6 rises monotonically across the ten tested subintervals from approximately 0.456 to 0.499, whereas N2/N6 and N4/N6 are nearly flat. This suggests that "rank stability" and "relative-frequency stationarity" should be treated as separate empirical notions.

These side findings are best viewed as prompts for future work rather than as separate claims requiring explanation in the present note. In particular, a focused paper need not pursue the detailed arithmetic mechanism behind every local rank movement; the existing jumping-champion literature already provides the natural theoretical context.

10. Limitations and scope

Several limitations should be explicit. First, all conclusions are finite-range computational statements; stability over the tested shells does not imply eventual stability. Second, the partition into decade shells and ten equal-width subintervals is natural but not unique. Other logarithmic or adaptive partitions could reveal additional transitions. Third, the shell structure, the ten-way subdivision, and the selected ratio diagnostics were chosen during exploratory analysis rather than preregistered. No formal change-point or multiple-comparison procedure has been applied. The observed rank changes are deterministic facts about the computed data, but any stochastic interpretation would require a separate model.

Fourth, this note does not claim priority for the observation that 6 is extraordinarily frequent or that 30 gains importance; these phenomena belong to the established theory and computation of jumping champions. The potentially useful contribution is the local-shell presentation: it separates candidate-set stability, rank-prefix stability, close-rank swapping, and relative-frequency drift in one simple finite-scale framework.

11. Conclusion

Across the tested disjoint numerical scales, consecutive prime-gap frequencies exhibit a mixture of rigidity and drift. Gap 6 remains first throughout every tested decade shell. From 10^7–10^8 onward, the same top-ten candidate set is observed in every tested decade, while the internal ordering continues to evolve. In 10^9–10^{10} and 10^{10}–10^{11} the first six ranks are identical, and a ten-way subdivision of 10^{10}–10^{11} preserves the top four 6, 12, 18, 10 in every tested subinterval.

The relative-frequency analysis shows why the result should not be summarized as complete scale invariance. Some gap ratios are nearly stationary, while 18, 24, and especially 30 gain relative weight. The empirical picture is therefore one of a stable local hierarchy with structured internal drift. This finite-scale perspective complements the classical cumulative jumping-champion question and provides a compact computational object for further comparison, replication, or theoretical interpretation.

Appendix A. Reproducibility specification

The numerical object is the consecutive-prime gap $g_n = p_{n+1} - p_n$. A gap belongs to the interval containing its starting prime p_n, even if p_{n+1} crosses the upper boundary. Rankings are by decreasing frequency, with smaller gap used only as a deterministic tie-breaker for exact equal counts. The main shells are 10^3–10^4, 10^4–10^5, 10^5–10^6, 10^6–10^7, 10^7–10^8, 10^8–10^9, 10^9–10^{10}, and 10^{10}–10^{11}. The fine partition uses ten equal-width intervals of length 9×10^9 covering 10^{10}–10^{11}.

The computation uses exact Eratosthenes sieving and, for larger ranges, an odd-only segmented sieve. The production count for 10^{10}–10^{11} used an optimized compiled implementation of this procedure; Appendix B gives a self-contained Python/NumPy reference implementation, and Appendix C gives a compiled C++17/OpenMP reproduction implementation. Neither implementation needs to store all primes or all gaps: only one segment, boundary primes, and integer gap counters are required. For integer endpoints A < B, the number of counted gaps whose starting prime lies in [A,B) is exactly the number of primes in that half-open interval, namely π(B−1) − π(A−1). For the power-of-ten endpoints used here, A and B are composite, so this quantity equals π(B) − π(A). The reference π(10^k) values used below are the standard tabulated values in OEIS A006880. For example, $\pi(10^4) - \pi(10^3) = 1{,}061$, $\pi(10^5) - \pi(10^4) = 8{,}363$, $\pi(10^6) - \pi(10^5) = 68{,}906$, and $\pi(10^{11}) - \pi(10^{10}) = 3{,}663{,}002{,}302$. These equal the gap totals reported in the corresponding shells.

Requirements for the reference implementation: Python 3.10 or later and NumPy 1.24 or later. The Python standard-library modules collections, math, and time are also used. NumPy can be installed with “python -m pip install numpy”. No SciPy, pandas, or plotting library is required. The same reference script reproduces small and large shells by changing only the parameter block: set A and B to the desired interval endpoints, PARTS = 1 for a whole shell, or PARTS = 10 with A = 10**10 and B = 10**11 to reproduce the ten-way subdivision. The algorithm is exact and memory-bounded, but runtime depends strongly on hardware. The small checkpoints are practical direct tests of the printed code; the 10^{10}–10^{11} production count was obtained with the optimized compiled implementation described above.

External count-validation checkpoints

A	B	π(B) − π(A)	Counted gaps	Leading ranks
10^3	10^4	1,061	1,061	6, 2, 4, 10, 12
10^4	10^5	8,363	8,363	6, 2, 4, 12, 10
10^5	10^6	68,906	68,906	6, 12, 2, 4, 10
10^{10}	10^{11}	3,663,002,302	3,663,002,302	6, 12, 18, 10, 2

Table A1. External count-validation checkpoints used by the reference script. In general the shell gap count is π(B−1) − π(A−1); for the composite power-of-ten endpoints used here this equals π(B) − π(A). Reference values are from OEIS A006880.

Appendix B. Standalone Python reproduction script

The following script is self-contained apart from NumPy. With the default parameter block it reproduces the first row of Table 1 in seconds. To reproduce another decade shell, replace A and B and leave PARTS = 1. To reproduce Table 3 and the ratio diagnostics, set A = 10**10, B = 10**11, and PARTS = 10. The program prints the gap count, a separately accumulated internal consistency check between starting-prime and gap counts, external reference checks when available, the requested ranking, and ratios to N(6). For very large shells this reference implementation is intentionally transparent rather than optimized; the reported production result used the compiled implementation in Appendix C.

```
"""Reproduce local rankings of consecutive prime gaps.

Requirements:
    Python >= 3.10
    NumPy >= 1.24

Install:
    python -m pip install numpy

Change only A, B, PARTS and TOP_K in the parameter block below.
For Table 1 use PARTS = 1 and the decade-shell endpoints.
For Table 3 use A = 10**10, B = 10**11, PARTS = 10.
"""

from collections import Counter
from math import isqrt
import time
import numpy as np

# ---------------- user parameters ----------------
A = 10**3
B = 10**4
PARTS = 1
TOP_K = 15
SEGMENT_ODDS = 5_000_000
# -------------------------------------------------

# Exact reference values used only as validation checks when available.
REFERENCE_PI = {
    10**3: 168,
    10**4: 1_229,
    10**5: 9_592,
    10**6: 78_498,
    10**7: 664_579,
    10**8: 5_761_455,
    10**9: 50_847_534,
    10**10: 455_052_511,
    10**11: 4_118_054_813,
}

RATIO_GAPS = (12, 18, 10, 2, 4, 30, 24, 8, 14)


def base_primes(limit):
    """All primes <= limit by an ordinary sieve."""
    sieve = np.ones(limit + 1, dtype=np.bool_)
    sieve[:2] = False
    for p in range(2, isqrt(limit) + 1):
        if sieve[p]:
            sieve[p * p : limit + 1 : p] = False
    return np.flatnonzero(sieve)


def primes_in_segment(lo, hi, small_primes):
    """Return primes in [lo, hi), using an odd-only segmented sieve."""
    out = []
    if lo <= 2 < hi:
        out.append(2)

    odd_lo = max(lo, 3)
    if odd_lo % 2 == 0:
        odd_lo += 1
    if odd_lo >= hi:
        return np.asarray(out, dtype=np.int64)

    n = (hi - odd_lo + 1) // 2
    sieve = np.ones(n, dtype=np.bool_)

    for p0 in small_primes:
```

```
        p = int(p0)
        if p == 2:
            continue
        if p * p >= hi and p > isqrt(hi - 1):
            break
        start = max(p * p, ((odd_lo + p - 1) // p) * p)
        if start % 2 == 0:
            start += p
        if start < hi:
            sieve[(start - odd_lo) // 2 :: p] = False

    odds = odd_lo + 2 * np.flatnonzero(sieve)
    if out:
        return np.concatenate((np.asarray(out, dtype=np.int64), odds))
    return odds.astype(np.int64, copy=False)


def part_index(p):
    """Part containing starting prime p, for PARTS equal-width pieces."""
    return min(PARTS - 1, ((p - A) * PARTS) // (B - A))


def rank(counter):
    return sorted(counter.items(), key=lambda kv: (-kv[1], kv[0]))


def main():
    if not (0 <= A < B and PARTS >= 1):
        raise ValueError("Require 0 <= A < B and PARTS >= 1")

    # Search one ordinary segment beyond B so that the final prime in [A,B)
    # has a successor. Increase automatically if an unusually long gap occurs.
    segment_span = 2 * SEGMENT_ODDS
    small = base_primes(isqrt(B + segment_span) + 2)

    counts = [Counter() for _ in range(PARTS)]
    starts_per_part = [0] * PARTS
    shell_prime_count = 0
    first_prime = None
    last_start_prime = None
    next_prime_after_B = None
    prev = None
    processed = A
    t0 = time.time()

    while next_prime_after_B is None:
        lo = processed
        hi = min(processed + segment_span, B + segment_span)
        ps = primes_in_segment(lo, hi, small)

        for q0 in ps:
            q = int(q0)
            if first_prime is None and A <= q < B:
                first_prime = q
            if A <= q < B:
                shell_prime_count += 1

            if prev is not None and A <= prev < B:
                g = q - prev
                j = part_index(prev)
                counts[j][g] += 1
                starts_per_part[j] += 1
                last_start_prime = prev

            if q >= B:
                next_prime_after_B = q
                break
            prev = q

        processed = hi
        if processed >= B + segment_span and next_prime_after_B is None:
            # Rare safeguard: extend the search and rebuild base primes.
            segment_span *= 2
            small = base_primes(isqrt(B + segment_span) + 2)

    elapsed = time.time() - t0
    total_gaps = sum(starts_per_part)
    shell_primes = shell_prime_count

    print(f"interval = [{A:,}, {B:,})")
    print(f"parts = {PARTS}")
    print(f"first prime in shell = {first_prime:,}")
    print(f"last starting prime = {last_start_prime:,}")
```

```
    print(f"next prime after B = {next_prime_after_B:,}")
    print(f"counted gaps = {total_gaps:,}")
    print(f"starting primes = {shell_primes:,}")
    print(f"internal consistency: gaps == separately counted starts -> {total_gaps == shell_primes}")

    if A in REFERENCE_PI and B in REFERENCE_PI:
        expected = REFERENCE_PI[B] - REFERENCE_PI[A]
        print(f"reference pi(B)-pi(A) for composite endpoints = {expected:,}")
        print(f"reference validation -> {expected == total_gaps}")

    for j, counter in enumerate(counts):
        lo = A + (B - A) * j // PARTS
        hi = A + (B - A) * (j + 1) // PARTS
        ordered = rank(counter)
        print(f"\npart {j + 1}: [{lo:,}, {hi:,})")
        print(f"gaps = {starts_per_part[j]:,}")
        print("rank  gap  frequency")
        for r, (g, n) in enumerate(ordered[:TOP_K], start=1):
            print(f"{r:>4} {g:>4} {n:>12,}")

        if counter[6]:
            print("ratios to N(6):")
            for g in RATIO_GAPS:
                print(f"N({g})/N(6) = {counter[g] / counter[6]:.6f}")

    print(f"\nelapsed seconds = {elapsed:.3f}")


if __name__ == "__main__":
    main()
```

For the default A = 10**3 and B = 10**4, the reference script reports 1,061 counted gaps, separately accumulates 1,061 starting primes, validates the result against $\pi(10^4) - \pi(10^3) = 1{,}061$, and begins the ranking 6, 2, 4, 10, 12. For A = 10**10, B = 10**11, PARTS = 1, the externally validated shell count is 3,663,002,302 and the leading sequence is 6, 12, 18, 10, 2, 4, 30, 24, 8, 14. These values are direct checkpoints for independent implementations; the large production count reported in the main text was obtained with the compiled segmented-sieve implementation reproduced in Appendix C.

Appendix C. Compiled C++17/OpenMP reproduction implementation

For the large shell, a compiled implementation is substantially faster than the transparent Python reference code. The implementation below uses the same half-open starting-prime convention, an odd-only segmented Eratosthenes sieve, parallel segment processing, and explicit treatment of gaps crossing segment boundaries and the upper shell boundary. It requires a C++17 compiler with OpenMP support; for GCC, a typical command is “g++ -O3 -march=native -fopenmp prime_gap_shell.cpp -o prime_gap_shell”. The constants A, B and SEG may be changed at the top of the program. With A = 10^10 and B = 10^11 it reproduces the shell frequencies reported in Table 2. The Python code remains the simpler reference specification, while this listing provides a directly auditable compiled route to the large computation.

```
#include <algorithm>
#include <array>
#include <cmath>
#include <cstdint>
#include <iostream>
#include <vector>
#include <omp.h>

using u64 = unsigned long long;
using i64 = long long;

static std::vector<int> base_primes(int n) {
    std::vector<bool> is_prime(n + 1, true);
    is_prime[0] = is_prime[1] = false;
    for (int p = 2; 1LL * p * p <= n; ++p)
        if (is_prime[p])
            for (i64 j = 1LL * p * p; j <= n; j += p)
                is_prime[(size_t)j] = false;
    std::vector<int> out;
    for (int i = 2; i <= n; ++i)
        if (is_prime[i]) out.push_back(i);
    return out;
}

int main() {
    const i64 A = 10000000000LL;
    const i64 B = 100000000000LL;      // shell [A,B)
    const i64 SEG = 100000000LL;        // segment width
    const int MAX_GAP = 2048;

    const int nseg = int((B - A + SEG - 1) / SEG);
    const int lim = int(std::sqrt((long double)(B + SEG))) + 2;
    const auto small = base_primes(lim);

    std::vector<std::array<u64, MAX_GAP>> local(nseg);
    std::vector<i64> first(nseg, -1), last(nseg, -1);
    for (auto &a : local) a.fill(0);

    #pragma omp parallel for schedule(dynamic,1)
    for (int s = 0; s < nseg; ++s) {
        const i64 raw_lo = A + i64(s) * SEG;
        const i64 hi = std::min(B, raw_lo + SEG);
        i64 lo = raw_lo;
        if ((lo & 1LL) == 0) ++lo;
        const i64 n_odd = (hi - lo + 1) / 2;
        std::vector<unsigned char> is_prime((size_t)n_odd, 1);

        for (int p : small) {
            if (p == 2) continue;
            const i64 pp = 1LL * p * p;
            if (pp >= hi && p > std::sqrt((long double)(hi - 1))) break;
            i64 start = std::max(pp, ((lo + p - 1) / p) * 1LL * p);
            if ((start & 1LL) == 0) start += p;
            if (start >= hi) continue;
            for (i64 j = (start - lo) / 2; j < n_odd; j += p)
                is_prime[(size_t)j] = 0;
        }

        i64 prev = -1;
        for (i64 i = 0; i < n_odd; ++i) if (is_prime[(size_t)i]) {
            const i64 q = lo + 2 * i;
            if (first[s] < 0) first[s] = q;
            if (prev > 0) {
                const i64 g = q - prev;
                if (g >= MAX_GAP) { std::cerr << "Increase MAX_GAP\n"; std::abort(); }
                local[s][(size_t)g]++;
            }
```

```
            prev = q;
            last[s] = q;
        }
    }

    std::array<u64, MAX_GAP> total{};
    total.fill(0);
    i64 previous_last = -1;
    for (int s = 0; s < nseg; ++s) {
        for (int g = 1; g < MAX_GAP; ++g) total[g] += local[s][g];
        if (previous_last > 0 && first[s] > 0) {
            const i64 g = first[s] - previous_last;
            if (g >= MAX_GAP) { std::cerr << "Increase MAX_GAP\n"; return 2; }
            total[(size_t)g]++;
        }
        if (last[s] > 0) previous_last = last[s];
    }

    // Find the first prime q >= B, so the last starting prime below B
    // receives its successor gap under the paper's convention.
    i64 q_after = -1;
    i64 lo = (B & 1LL) ? B : B + 1;
    for (i64 span = SEG; q_after < 0; span *= 2) {
        const i64 hi = B + span;
        const i64 n_odd = (hi - lo + 1) / 2;
        std::vector<unsigned char> is_prime((size_t)n_odd, 1);
        for (int p : small) {
            if (p == 2) continue;
            const i64 pp = 1LL * p * p;
            i64 start = std::max(pp, ((lo + p - 1) / p) * 1LL * p);
            if ((start & 1LL) == 0) start += p;
            if (start >= hi) continue;
            for (i64 j = (start - lo) / 2; j < n_odd; j += p)
                is_prime[(size_t)j] = 0;
        }
        for (i64 i = 0; i < n_odd; ++i)
            if (is_prime[(size_t)i]) { q_after = lo + 2 * i; break; }
    }

    if (previous_last > 0 && q_after > 0) {
        const i64 g = q_after - previous_last;
        if (g >= MAX_GAP) { std::cerr << "Increase MAX_GAP\n"; return 2; }
        total[(size_t)g]++;
    }

    std::vector<std::pair<u64,int>> ranking;
    u64 gap_count = 0;
    for (int g = 1; g < MAX_GAP; ++g) if (total[g]) {
        ranking.push_back({total[g], g});
        gap_count += total[g];
    }
    std::sort(ranking.begin(), ranking.end(), [](auto a, auto b) {
        return (a.first != b.first) ? a.first > b.first : a.second < b.second;
    });

    std::cout << "interval = [" << A << ", " << B << ")\n";
    std::cout << "counted gaps = " << gap_count << "\n";
    std::cout << "first prime in shell = " << first.front() << "\n";
    std::cout << "last starting prime = " << previous_last << "\n";
    std::cout << "next prime after B = " << q_after << "\n";
    std::cout << "rank  gap  frequency\n";
    for (size_t i = 0; i < std::min<size_t>(15, ranking.size()); ++i)
        std::cout << i + 1 << "  " << ranking[i].second << "  " << ranking[i].first << "\n";
}
```

For the published large-shell checkpoint, the expected total is 3,663,002,302 gaps and the leading sequence is 6, 12, 18, 10, 2, 4, 30, 24, 8, 14. The tabulated π(10^k) values used for external validation are available in OEIS A006880.

References

Erdős, P., & Straus, E. G. (1980). Remarks on the differences between consecutive primes. Elemente der Mathematik, 35(5), 115–118.

Goldston, D. A., & Ledoan, A. H. (2011). Jumping champions and gaps between consecutive primes. International Journal of Number Theory, 7(6), 1413–1421. https://doi.org/10.1142/S179304211100471X

Goldston, D. A., & Ledoan, A. H. (2015). The jumping champion conjecture. Mathematika, 61(3), 719–740. https://doi.org/10.1112/S0025579315000078

Hardy, G. H., & Littlewood, J. E. (1923). Some problems of “Partitio Numerorum”; III: On the expression of a number as a sum of primes. Acta Mathematica, 44, 1–70. https://doi.org/10.1007/BF02403921

Odlyzko, A., Rubinstein, M., & Wolf, M. (1999). Jumping champions. Experimental Mathematics, 8(2), 107–118. https://doi.org/10.1080/10586458.1999.10504393

OEIS Foundation Inc. (2026). A006880: Number of primes < 10^n. The On-Line Encyclopedia of Integer Sequences. https://oeis.org/A006880

Contact

www.dietereigler.at