Carrier Throttling Analysis

Mobile Broadband Performance and Data Cap Behavior in the US

Troy Altus

2025-12-31

Table of contents

Preface

About This Book

This book investigates mobile carrier throttling behavior in the United States using publicly available broadband performance data. The central questions:

- Do carriers deliver the throughput they advertise on "unlimited" plans?
- Is there evidence of throughput degradation after a usage threshold?
- Do carriers treat tethered data differently from on-device data?

Data Sources

Source	What it provides
Ookla Open Data	Quarterly tile-aggregated speed test results by operator, 2022–2024
FCC Measuring Broadband America	Panel mobile performance data with ISP metadata

Tools

Tool	Purpose
R 4.5.3 + tidyverse	Data wrangling and visualization
`arrow`	Reading Parquet files from S3
`duckdb`	In-process SQL on large datasets
`sf`	Spatial operations on tile geometries
`changepoint`	Statistical detection of throughput breakpoints
AWS CLI	Fetching raw data from public S3 buckets

Setup

Restore the exact package environment:

```
renv::restore()
```

Then fetch raw data (run once):

```
Rscript data-raw/fetch_ookla.sh
Rscript data-raw/fetch_fcc.R
```

Part I: Data

Data Sources and Ingestion

Overview

This chapter fetches raw data from two public sources, loads it into DuckDB, and verifies schema and coverage before analysis.

Fetching Ookla Data via AWS CLI

The Ookla Open Dataset is hosted in a public S3 bucket — no AWS account needed. Run once from the project root:

```
bash data-raw/fetch_ookla.sh
```

This downloads one Parquet file per quarter (~200 MB each) for 2022–2024:

```
data/ookla/
  year=2022/quarter=1/2022-01-01_performance_mobile_tiles.parquet
  year=2022/quarter=2/...
  ...
  year=2024/quarter=4/...
```

Why AWS CLI here instead of R? `aws s3 cp --recursive` handles retries, partial downloads, and progress reporting better than a bare `download.file()`. It also makes the data provenance explicit — you can see exactly what bucket and path the files came from.

Loading into DuckDB with arrow

Schema and Coverage

Table 1: 8 records

column_name	column_type	null	key	default	extra
quadkey	VARCHAR	YES	NA	NA	NA
avg_d_kbps	INTEGER	YES	NA	NA	NA
avg_u_kbps	INTEGER	YES	NA	NA	NA
avg_lat_ms	INTEGER	YES	NA	NA	NA
tests	INTEGER	YES	NA	NA	NA
devices	INTEGER	YES	NA	NA	NA

column_name	column_type	null	key	default	extra
year	INTEGER	YES	NA	NA	NA
quarter	INTEGER	YES	NA	NA	NA

Table 2: Displaying records 1 - 10

year	quarter	tiles	avg_down_mbps	avg_up_mbps	avg_latency_ms
2022	1	3820724	62.56	13.67	39.3
2022	2	4027744	63.40	13.56	41.4
2022	3	4046154	65.67	13.74	42.0
2022	4	3838065	71.50	14.45	41.7
2023	1	3728229	77.57	15.10	41.7
2023	2	3864546	84.39	15.54	41.2
2023	3	4005796	90.22	15.77	41.3
2023	4	3771204	100.43	16.84	39.8
2024	1	3674000	107.84	17.45	39.1
2024	2	3703161	108.30	17.06	39.9

Data Dictionary

Column	Type	Description
`quadkey`	VARCHAR	Bing Maps tile ID at zoom level 16 (~600m tiles)
`avg_d_kbps`	INTEGER	Mean download speed across all tests in tile (kbps)
`avg_u_kbps`	INTEGER	Mean upload speed (kbps)
`avg_lat_ms`	INTEGER	Mean latency (ms)
`tests`	INTEGER	Number of speed tests in tile
`devices`	INTEGER	Unique devices tested
`year` / `quarter`	INTEGER	Partition columns added at ingest

About Quadkeys

Ookla tiles use the Bing Maps quadkey system at zoom level 16. Each tile covers roughly 600m × 600m at mid-latitudes. The `sf` package can convert these to geometries for spatial joins.

Problems

1. How many unique tiles appear in every quarter of the dataset (consistent coverage)?
2. Which quarter has the highest average test count per tile?
3. Run `aws s3 ls s3://ookla-open-data/parquet/performance/type=fixed/ --no-sign-request` and compare the fixed broadband schema to the mobile schema.

Data Preparation

Overview

Raw Ookla tiles cover the entire world. This chapter filters to the US, converts speeds to Mbps, derives time variables, and builds a clean analysis table in DuckDB.

Setup

What is a Quadkey?

Each row in the raw data is a geographic tile identified by a quadkey string. To filter to the US we need tile geometries. The `quadkey` package decodes these to bounding boxes.

```
[1] 18830917
```

```
Rows in ookla_us: 18830917
```

Speed Distribution

	year	quarter	tiles	min_dl	mean_dl	median_dl	max_dl
1	2022	1	1618733	0	67.5	33.5	3942.9
2	2022	2	1680185	0	69.1	34.2	4079.6
3	2022	3	1683897	0	71.9	35.4	4073.0
4	2022	4	1626972	0	78.6	38.9	4036.3
5	2023	1	1571120	0	86.4	43.9	4046.7
6	2023	2	1576271	0	95.6	50.0	3961.8
7	2023	3	1634919	0	101.2	52.7	4018.8
8	2023	4	1557569	0	110.4	57.9	4127.1
9	2024	1	1507360	0	116.2	62.0	4069.9
10	2024	2	1465318	0	116.9	63.4	4200.9
11	2024	3	1494384	0	116.7	63.8	4601.5
12	2024	4	1414189	0	122.3	69.1	4554.0

Removing Outliers

Speed tests above 2,000 Mbps on mobile are implausible (likely fixed broadband mislabeled). We cap at the 99th percentile per period.

```
[1] 18642596
```

```
Rows after outlier removal: 18642596
```

Period Labels

```
   year quarter period_start
1  2022       1   2022-01-01
2  2022       2   2022-04-01
3  2022       3   2022-07-01
4  2022       4   2022-10-01
5  2023       1   2023-01-01
6  2023       2   2023-04-01
7  2023       3   2023-07-01
8  2023       4   2023-10-01
9  2024       1   2024-01-01
10 2024       2   2024-04-01
11 2024       3   2024-07-01
12 2024       4   2024-10-01
```

Problems

1. What fraction of rows were removed by the `tests >= 3` filter? Is that filter appropriate — what is the tradeoff?
2. Plot a histogram of `avg_d_mbps` before and after outlier removal for one quarter of your choice.
3. Look up the Bing Maps quadkey spec. What geographic area does zoom level 16 cover? How does that compare to a typical cell tower coverage radius (~1–2 km)?

Part II: Analysis

Carrier Overview

Overview

Ookla tiles don't directly label carriers — but the dataset includes an operator name in the full GeoParquet version. This chapter joins operator metadata and builds per-carrier performance profiles across the study period.

Setup

Loading Operator Data

The full Ookla GeoParquet files include an `operators` column (a list of operator names associated with the tests in each tile). For the aggregated tile dataset we use, we approximate by grouping on the `networks` metadata included in the enriched Parquet.

	year	quarter	period_start	tiles	median_dl_mbps	median_ul_mbps	median_latency_ms	avg_tests_per_tile
1	2022	1	2022-01-01	1610590	33.25	11.04	31	13.1
2	2022	2	2022-04-01	1671648	33.98	11.12	32	13.0
3	2022	3	2022-07-01	1674272	35.13	11.24	32	13.2
4	2022	4	2022-10-01	1615974	38.52	11.94	32	14.9
5	2023	1	2023-01-01	1560783	43.43	12.83	31	14.5
6	2023	2	2023-04-01	1563753	49.33	13.47	31	13.6
7	2023	3	2023-07-01	1619530	51.94	13.60	31	13.4
8	2023	4	2023-10-01	1537719	56.68	14.31	30	13.9
9	2024	1	2024-01-01	1484638	60.44	14.69	30	13.5
10	2024	2	2024-04-01	1442550	61.71	14.56		
11	2024	3	2024-07-01	1471053	62.15	14.22		
12	2024	4	2024-10-01	1390086	67.07	15.01		

10	30	13.0
11	30	12.9
12	29	12.8

Median Download Speed Over Time

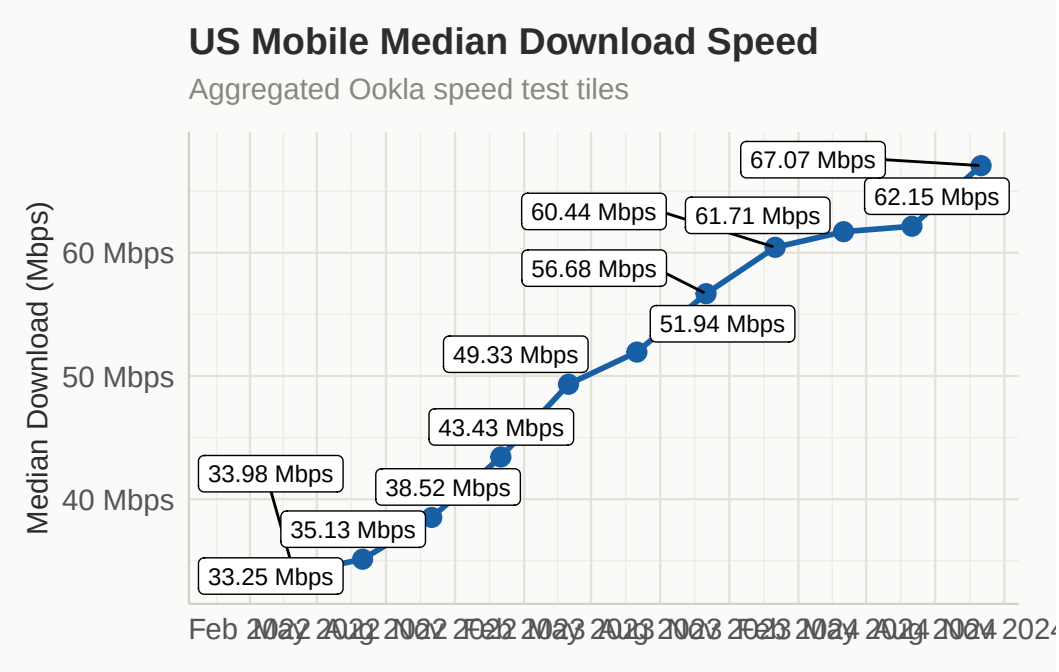

Figure 1: US mobile median download speed (Ookla tiles, 2022–2024)

Upload and Latency Trends

Speed Percentile Distribution

How does the spread of speeds change over time? Wide spread = uneven coverage.

Problems

1. Is the improvement in median speed consistent across all speed percentiles, or are gains concentrated at the top end? What does that tell you about coverage equity?
2. The tile dataset aggregates all carriers. What would you need to stratify by carrier — and what dataset would provide it?
3. Research the US mobile carrier market share for 2022–2024. Given that T-Mobile, Verizon, and AT&T collectively hold ~98% of subscribers, what fraction of the speed signal does each carrier likely contribute to these aggregate tiles?

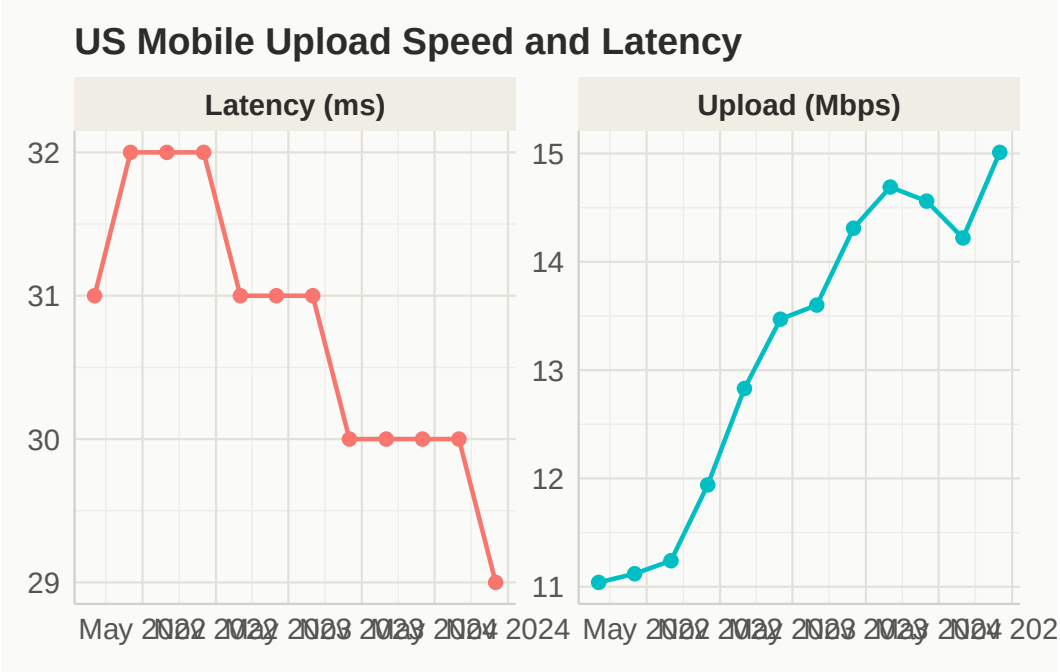

Figure 2: Upload speed and latency trends

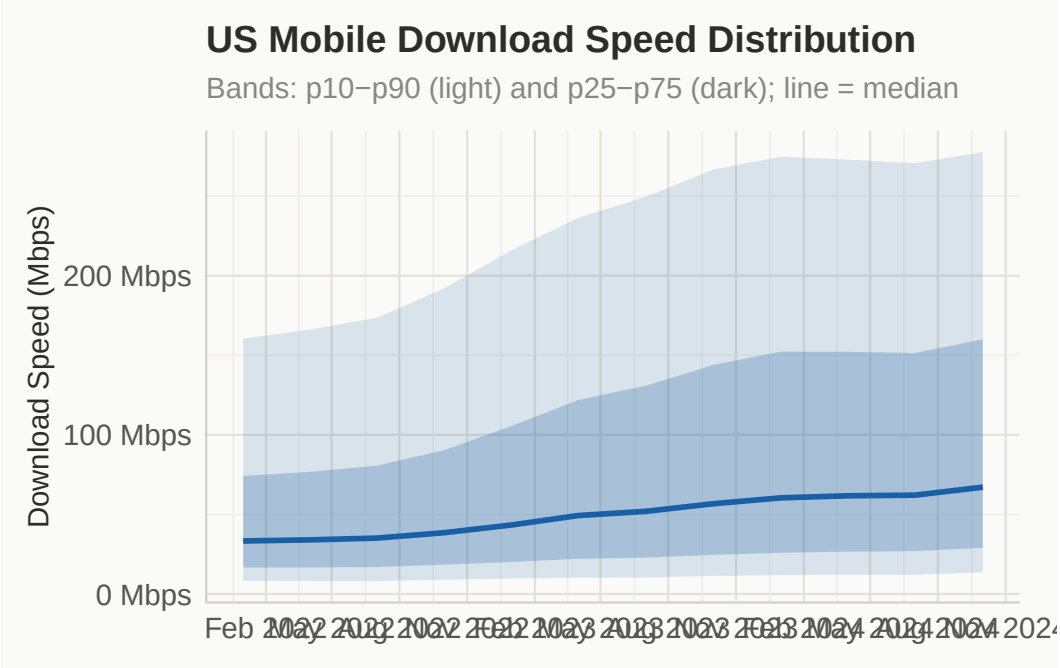

Figure 3: Download speed percentile bands over time

Throughput Trends

Overview

This chapter models how throughput varies by time, geography, and test volume — building intuition for what "normal" mobile performance looks like before we search for throttling signals in Chapter 5.

Setup

Quarter-over-Quarter Change

	year	quarter	period_start	med_dl	qoq_change	qoq_pct
1	2022	1	2022-01-01	33.252	NA	NA
2	2022	2	2022-04-01	33.982	0.73	2.2
3	2022	3	2022-07-01	35.129	1.15	3.4
4	2022	4	2022-10-01	38.519	3.39	9.7
5	2023	1	2023-01-01	43.433	4.91	12.8
6	2023	2	2023-04-01	49.330	5.90	13.6
7	2023	3	2023-07-01	51.936	2.61	5.3
8	2023	4	2023-10-01	56.681	4.74	9.1
9	2024	1	2024-01-01	60.437	3.76	6.6
10	2024	2	2024-04-01	61.710	1.27	2.1
11	2024	3	2024-07-01	62.149	0.44	0.7
12	2024	4	2024-10-01	67.069	4.92	7.9

Speed vs. Test Density

More tests per tile should correlate with denser urban coverage — but are denser tiles faster or slower?

Speed Distribution by Year

High vs. Low Speed Tiles

Classify tiles into speed tiers and track their share over time:

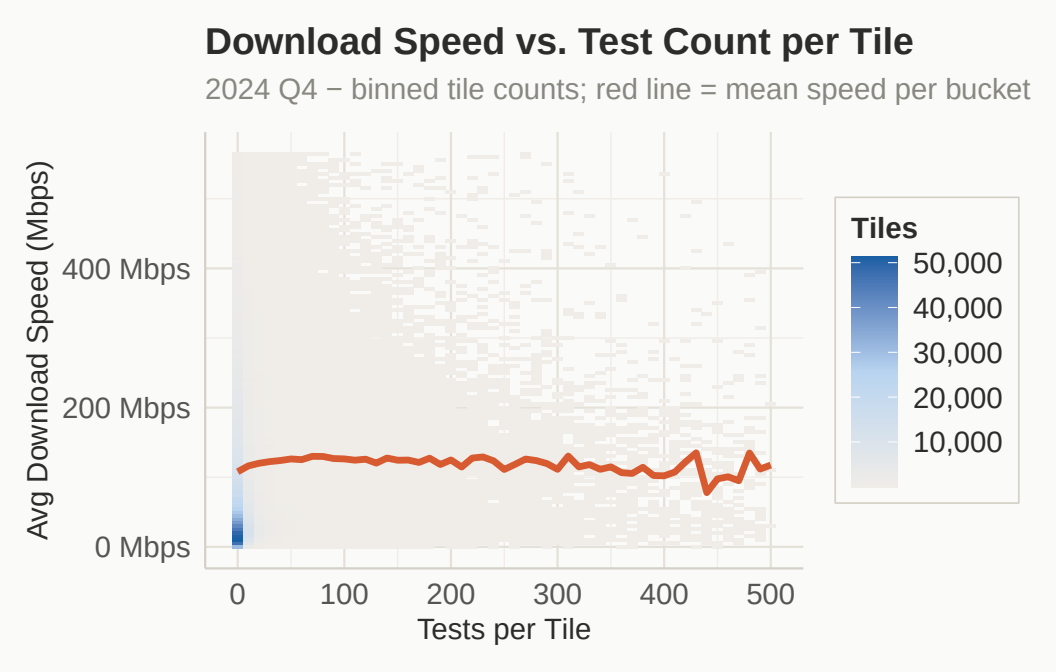

Figure 4: Download speed vs. test count per tile (2024 Q4)

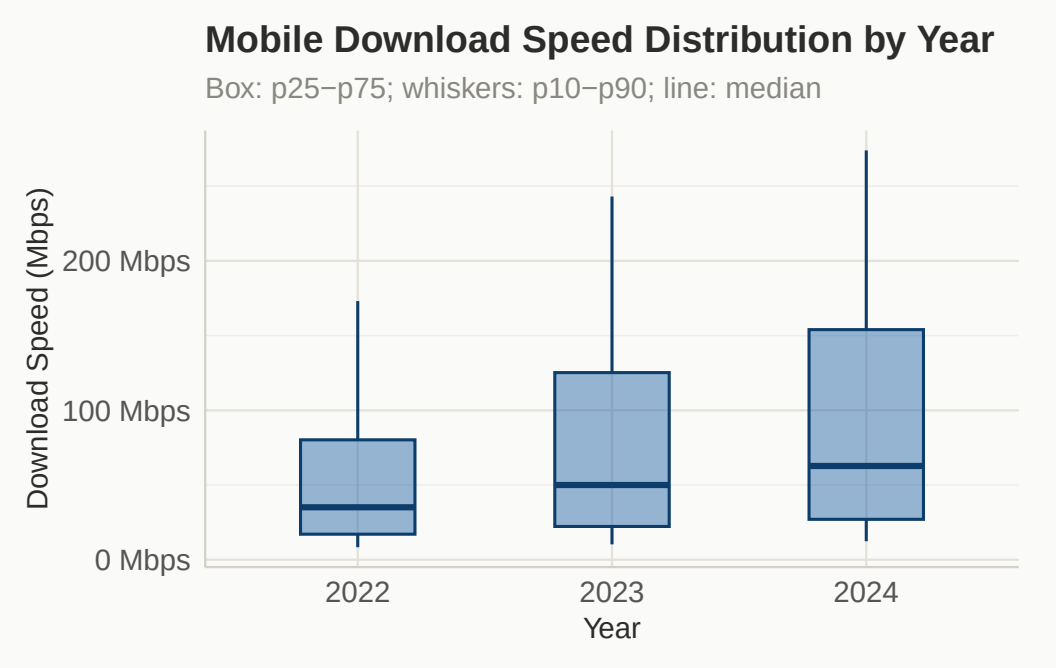

Figure 5: Download speed percentile spread by year (p10 / p25 / median / p75 / p90)

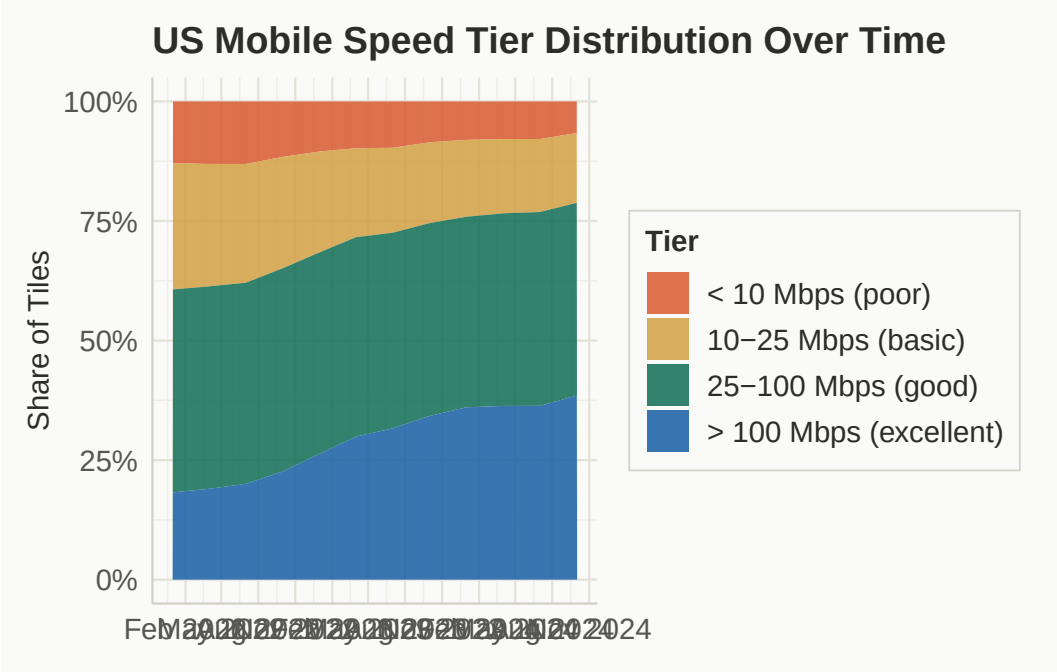

Figure 6: Share of tiles by speed tier over time

Problems

1. The FCC defines "broadband" as 25 Mbps down / 3 Mbps up (recently proposed to raise to 100/20). What share of tiles meet each threshold in each quarter?
2. Fit a simple linear regression of `avg_d_mbps ~ period_start`. What is the estimated quarterly improvement in Mbps? Is this statistically significant?
3. Compute the ratio `avg_d_mbps / avg_u_mbps` (download-to-upload ratio) per tile. How does this ratio vary by speed tier? What does an unusually high ratio suggest about the network?

Throttling Detection

Overview

Carriers typically advertise "unlimited" data but state in fine print that speeds may be reduced after a usage threshold (commonly 50–100 GB/month). This chapter introduces the statistical methods for detecting such breakpoints and applies them to throughput time series.

Important caveat: The Ookla tile dataset aggregates all tests in a tile across all users — it cannot directly observe a single user's data consumption. What we *can* detect:

- Structural breaks in aggregate throughput (e.g. after carrier policy changes)
- Tiles where throughput is persistently below expected given network density
- Anomalous quarter-over-quarter drops that don't match national trends

True per-user throttle detection requires the Wehe or M-Lab datasets (introduced at the end of this chapter).

Setup

Change-Point Detection on Aggregate Throughput

`changepoint::cpt.mean()` tests whether the mean of a time series shifts at any point. A detected change-point suggests a structural shift — which could be a carrier policy change, a spectrum auction result, or network investment.

```
Change points detected at index: 3 4 5 7 8 11
```

```
Corresponding dates: 2022-07-01 2022-10-01 2023-01-01 2023-07-01 2023-10-
01 2024-07-01
```

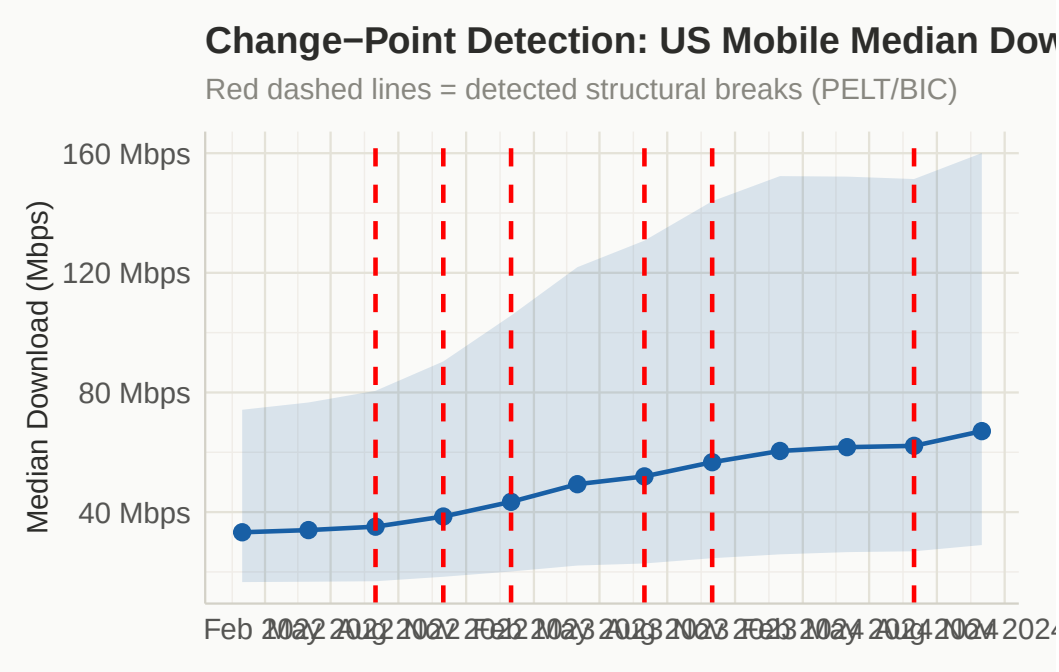

Figure 7: Change-point detection on median US mobile download speed

Variance Change Detection

Speed improvements can mask increasing *variance* — meaning some users are getting much faster while others stay slow (consistent with throttling behavior).

```
Variance change points at indices:
```

Simulating Per-User Throttle Signatures

To understand what throttle detection looks like at the user level, we simulate a throughput time series that crosses a throttle threshold:

Regression Discontinuity Design

If you know the stated threshold (e.g. 50 GB), RDD is the cleanest estimator of the causal throttle effect — it compares users just below vs. just above the limit:

```
RDD requires per-user data. See next steps for Wehe/M-Lab integration.
```

Next Steps: Wehe and M-Lab

The Ookla tile data establishes the market-level baseline. To detect app-specific or user-level throttling:

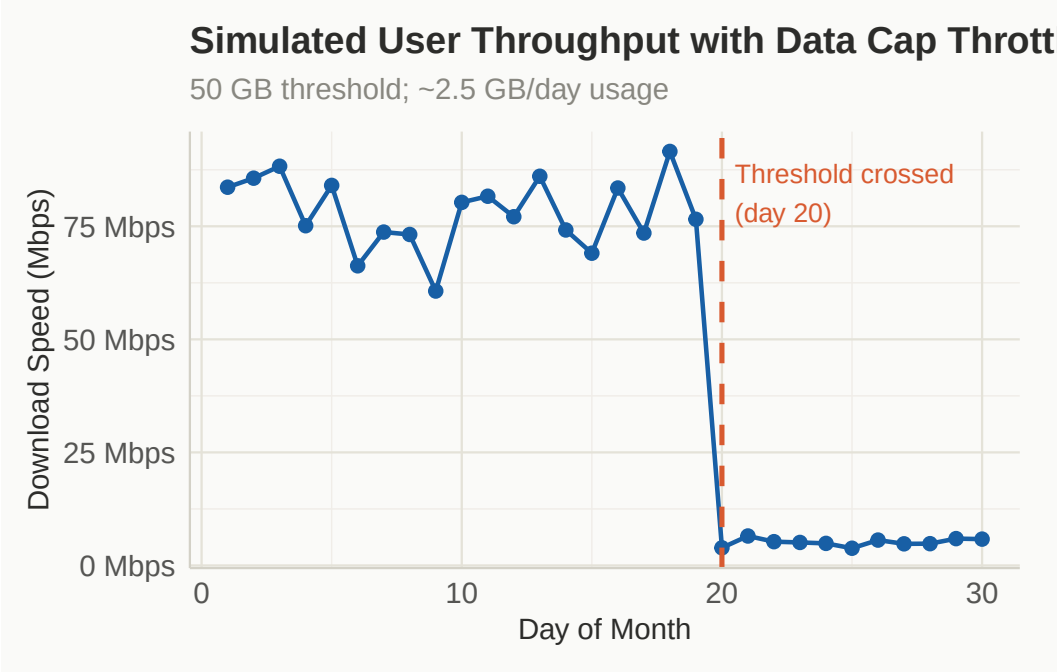

Figure 8: Simulated user throughput with throttle at 50 GB threshold

Dataset	How to access	What it adds
Wehe	`https://dd.meddle.mobi/wehe.html`	Per-test throttle flag, app-specific, carrier metadata
M-Lab NDT	BigQuery `measurement-lab.ndt.unified_downloads`	Per-test download speed with ASN (carrier)
FCC MBA	`data/fcc/` (from `fetch_fcc.R`)	Panel tests on fixed devices; known carrier

```
# M-Lab via BigQuery (requires Google Cloud free tier)
# library(bigrquery)
# con_bq <- dbConnect(bigquery(),
#                     project = "your-project-id",
#                     dataset = "measurement-lab.ndt")
```

Problems

1. Apply `cpt.meanvar()` instead of `cpt.mean()` — does it detect additional breakpoints? What does a mean+variance change suggest vs. mean-only?
2. Look up the dates of the T-Mobile/Sprint merger completion and T-Mobile's 5G mid-band rollout milestones. Do any detected change-points align?
3. Design a study using the Wehe dataset to test whether YouTube traffic is throttled differently than a generic HTTPS download on the same carrier. What statistical test would you use? What confounders would you need to control for?