Using tail classes

Tail thickness classes and the FFT grid they size

Stephen J. Mildenhall

2026-10-30

An aggregate is computed on $n = 2^\ell$ buckets of size b, with $b =$ `bs` and $\ell =$ `log2`, which for a non-negative law cover the window $[0, nb)$. For a given ℓ a smaller bucket gives finer resolution and a shorter window (Using FFT windows). How far the window must reach depends on how fast the right tail decays, and `aggregate` assigns every tail a **tail class** before it sizes the grid.

Write $H(x) = -\log \Pr(X > x)$ for the cumulative hazard. An unbounded tail is super-exponential when $H(x)/x \to \infty$, exponential when $H(x)/x \to \lambda \in (0, \infty)$, subexponential when $H(x)/x \to 0$ and $H(x)/\log x \to \infty$, and power-law when $H(x)/\log x \to \alpha$. Together with bounded, these are the five tail classes, ordered from lightest to heaviest. The aggregate class is the heavier of the frequency and severity classes. For a subexponential or heavier severity the aggregate tail follows the single big jump, $\Pr(S > x) \sim \mathsf{P}N \Pr(X > x)$: one large claim on an otherwise typical total.

The tail class is determined by the frequency and severity families and any bound a limit imposes, never by a computed density, so it is available before `update()`. The sizer starts from the window of a three-moment fit holding $1 - 10^{-12}$ of the aggregate. For a thick right tail (subexponential or power-law) it also computes the single big jump reach $\mathsf{P}S - \mathsf{P}X + q_X(p^{**})$, where q_X is the severity quantile function and $p^{**} = 1 - 10^{-12}/\mathsf{P}N$. The sizer extends the window to that reach when it fits in 2^ℓ buckets of the same size, and otherwise keeps the window and records the probability it clips. For a thin tail the reach is not used.

Tail classes[1] are computed for an `Aggregate` and for a `Portfolio`, with no DecL of their own. The examples are three aggregates of 20 Poisson claims whose gamma, lognormal and Lomax severities share mean 100 and CV 1.5.

```
agg Gamma 20 claims sev gamma 100 cv 1.5 poisson
agg Lognorm 20 claims sev lognorm 100 cv 1.5 poisson
agg Lomax 20 claims sev 260 * lomax 3.6 poisson
```

```
import pandas as pd
from aggregate import build, qd, __version__
from style import wprint, GT, figures
figures()
print(__version__)
```

```
1.43.1
```

[1] From aggregate 1.0

Tail classes before update

The attribute `tail_class` is the triple of frequency, severity and aggregate tail classes. The aggregates are built with `update=False`, so there is no grid yet, and `bs_window_df` and the return period table `tail_df` are `None`.

```
programs = {'gamma': 'agg Gamma 20 claims sev gamma 100 cv 1.5 poisson',
            'lognorm': 'agg Lognorm 20 claims sev lognorm 100 cv 1.5 poisson',
            'lomax': 'agg Lomax 20 claims sev 260 * lomax 3.6 poisson'}
books = {k: build(v, update=False) for k, v in programs.items()}
print(books['gamma'].bs_window_df, books['gamma'].tail_df)
GT(pd.DataFrame({k: [str(c) for c in a.tail_class] for k, a in books.items()},
                index=pd.Index(['freq', 'sev', 'agg'], name='component')))
```

```
None None
```

component	gamma	lognorm	lomax
freq	super-exponential	super-exponential	super-exponential
sev	exponential	subexponential	power-law
agg	exponential	subexponential	power-law

The table `tail_behavior_df` has a row for the frequency, one for each severity component, and one for the aggregate. Its columns are the support (`min`, `max`), the class of each tail (`left_tail`, `right_tail`), and for the aggregate a `concentrated` flag (CV under 0.1) and the CV; a few are shown.

```
GT(books['lomax'].tail_behavior_df[['family', 'right_tail', 'cv', 'note']])
```

component	family	right_tail	cv	note
frequency	poisson	super-exponential	—	claim count
comp0	lomax	power-law	—	power-law, alpha=3.6
aggregate	severity-driven	power-law	0.4031	power-law, alpha=3.6

The string `tail_description` gives the same information in three aligned lines, and `tail_explanation` gives it in prose, with the mechanism behind the aggregate tail. For the Lomax the mechanism is the single big jump, and the aggregate has the same power-law index as the severity, $\alpha = 3.6$.

```
wprint(books['lomax'].tail_description)
wprint(books['lomax'].tail_explanation)
```

```
frequency tail          poisson, count [0, inf), super-exponential right tail
severity tail           lomax, [0, inf), power-law right tail
aggregate tail          [0, inf), power-law right tail; not concentrated
                        (cv=0.403) [power-law, alpha=3.6]
Poisson frequency on counts [0, inf) (super-exponential right tail). Lomax
severity [0, inf), power-law right tail. The aggregate is [0, inf), power-law
right tail. Its right tail follows the severity by the single big jump P(S>x) ~
E[N]*P(X>x). Power-law, alpha=3.6. It is not concentrated: cv ~ 0.403.
```

Windows for three tail classes

The table `bs_window_df` has a row for each sizing method that applied and a `used` row for the realized window (Using FFT windows). The `selected` column marks the method whose bucket was taken, `W` is the width of each window, and `coverage` the probability it is sized to hold, adjusted by p^{**} for `sbj` (shown as `E[N]-adj`). The `clipped` column estimates the probability beyond the top of the window.

```
for a in books.values():
    a.update()
cols = ['selected', 'W', 'bs', 'log2', 'coverage', 'clipped']
GT(pd.concat({k: a.bs_window_df[cols] for k, a in books.items()},
             names=['severity', 'method']), float_format=',.6g')
```

severity	*method*	selected	W	bs	log2	coverage	clipped
gamma	moment	True	15,038.9	0.25	16	1-1e-12	—
	sbj	False	30,971.4	0.5	16	E[N]-adj 1-1e-12	—
	used	False	16,384	0.25	16	1-1e-12	—
lognorm	moment	True	29,492.3	0.5	16	1-1e-12	—
	sbj	False	180,738	4	16	E[N]-adj 1-1e-12	—
	used	False	32,768	0.5	16	1-1e-12	5.78511e-08
lomax	moment	True	39,913.1	1	16	1-1e-12	—
	sbj	False	271,476	5	16	E[N]-adj 1-1e-12	—
	used	False	65,536	1	16	1-1e-12	4.95711e-08

All three take the `moment` window. The single big jump reach of the gamma is twice its window and is not used, because the tail is exponential. For the lognormal and the Lomax the reach is six and seven times the window, too far to fit in 2^{16} buckets of the same size, so the window stays and the clipped probability, about 5×10^{-8}, is reported. The one-line `bs_description` gives the window and the `log2` that would hold the reach at the same bucket, and `validation_description` gives the result of the moment check.

```
for k, a in books.items():
    wprint(f'{k}: {a.validation_description}')
    wprint(' ', a.bs_description)
```

```
gamma: not unreasonable
  moment grid: bs=1/4, log2=16, window [0, 16,384] of width 16,384
lognorm: not unreasonable
  moment grid: bs=1/2, log2=16, window [0, 32,768] of width 32,768; clips
  ~5.79e-08 of the tail (raise log2 to 19)
lomax: fails sev skew, agg skew
  moment grid: bs=1, log2=16, window [0, 65,536] of width 65,536; clips
  ~4.96e-08 of the tail (raise log2 to 19)
```

The Lomax fails the skewness check. Its third moment is finite, but the part beyond x, $\mathsf{P}(X^3; X > x)$, decays like $x^{3-\alpha} = x^{-0.6}$, so beyond the top of the window, 65,536, it is still about 9 percent of the whole.

When the reach does fit, the window grows to it. A lognormal with CV 0.75 has a reach that fits at the same bucket, so the `moment` row is floored up to the `sbj` row: the two share `x_max`, and the `note` column ends `sbj floor`.

```
b = build('agg Lognorm2 20 claims sev lognorm 100 cv 0.75 poisson')
GT(b.bs_window_df[['x_max', 'W', 'bs', 'log2', 'note']], float_format=',.6g')
```

method	x_max	W	bs	log2	note
moment	13,433	13,433	0.25	16	3-moment MoM window; sbj floor
sbj	13,433	13,433	0.25	16	single big jump: ES - mu_X + q_X(p**)
used	16,384	16,384	0.25	16	realized grid (moment)

Capturing a clipped tail

The `log2` suggested by `bs_description`, 19 for the lognormal, assumes the same bucket. Passing `log2=19` alone makes the sizer refine the bucket to 1/16 over the same window, and the clip remains. Fixing the bucket as well, with `bs=1/2`, extends the window to 262,144, past the reach of 180,738.

```
for kw in [dict(log2=19), dict(bs=1/2, log2=19)]:
    wprint(kw, build(programs['lognorm'], **kw).bs_description)
```

```
{'log2': 19} moment grid: bs=1/16, log2=19, window [0, 32,768] of width 32,768;
clips ~5.79e-08 of the tail (raise log2 to 22)
{'bs': 0.5, 'log2': 19} moment grid: bs=1/2, log2=19, window [0, 262,144] of
width 262,144
```

Return periods for both tails

The return period table `tail_df` is indexed by the non-exceedance probability P and is symmetric about the median: each period T gives a row at $1/T$ and one at $1 - 1/T$. The method `tail_periods_df(periods)` computes the table for another list of periods. The columns `VaR` and `TVaR` are the value at risk (the quantile) and the tail value at risk, `xsVaR` is `VaR` less the mean, and `VaR/Mean` is the ratio of `VaR` to the mean.

```
GT(books['lognorm'].tail_periods_df([2, 10, 100, 1000]),
   formatters={'VaR': 0, 'TVaR': 0, 'xsVaR': 0, 'VaR/Mean': 3})
```

P	T	VaR	TVaR	xsVaR	VaR/Mean
0.0010	1,000	426	2,002	-1,574	0.213
0.0100	100	666	2,015	-1,334	0.333
0.1000	10	1,107	2,122	-893	0.554
0.5000	2	1,880	2,606	-120	0.940
0.9000	10	3,028	3,671	1,028	1.514
0.9900	100	4,488	5,271	2,488	2.244
0.9990	1,000	6,294	7,467	4,294	3.147