Using `approximate`

Method-of-moments approximations to an aggregate distribution and their error in the tail

Stephen J. Mildenhall

2026-10-30

A **method-of-moments approximation** replaces an aggregate distribution by a continuous distribution from a named family, with parameters chosen to match its mean, coefficient of variation (CV), and, for a shifted family, its skewness. The approximation is quick to state and to evaluate, and it suits a unit with many claims, whose distribution is close to normal. Its weakness is the tail: two distributions with the same three moments can have quite different 1 in 1,000 year losses. The library fits five families, compares each with the FFT answer, and can use any of them in place of the convolution.

With mean m, standard deviation σ, CV $\nu = \sigma/m$, and skewness $\gamma > 0$, the shifted gamma `sgamma` is $s + G$, where G is gamma with shape α and scale θ, and

$$\alpha = \frac{4}{\gamma^2}, \qquad \theta = \frac{\sigma\gamma}{2}, \qquad s = m - \frac{2\sigma}{\gamma}.$$

The shifted lognormal `slognorm` is $s+e^{\mu+\tau Z}$, with Z standard normal and τ solving $\gamma = (\omega+2)\sqrt{\omega-1}$, $\omega = e^{\tau^2}$. The families `gamma` and `lognorm` have no shift and match the mean and CV only, so their skewness is fixed by the CV, at 2ν and $3\nu + \nu^3$. The family `norm` matches the mean and CV with zero skewness. A left-skewed distribution is fitted by reflection.

The DecL clause `approximate KIND`, or `approx KIND`, follows the frequency and any aggregate reinsurance in an `agg` program, where KIND is one of the five families or `exact`, the default. At construction the clause replaces the compound by the fitted distribution, a single severity on a fixed count of one claim, fitted to the theoretical moments of the compound. The attribute `approximation` holds the kind, or `''` for an exact aggregate. On a built `Aggregate` or `Portfolio`, the method `approximate(approx_type, output)` returns a fit to the computed distribution, and the property `approximation_df` compares all five fits with the FFT answer.

```
agg Book 100 claims sev lognorm 100 cv 2 poisson approximate sgamma
agg Book 100 claims sev lognorm 100 cv 2 poisson approx slognorm
agg Book 100 claims sev lognorm 100 cv 2 poisson aggregate net of 5000 xs 12000 approximate sgamma
approximate norm | gamma | lognorm | sgamma | slognorm | exact     # the six kinds
```

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

```
1.43.1
```

A shifted gamma in place of the convolution

The example book has 100 Poisson claims from a lognormal severity with mean 100 and CV $\nu_X = 2$, for an aggregate mean of 10,000. Its variance is $100 \times 100^2 \times (1 + 2^2) = 5{,}000{,}000$, so $\sigma = 2{,}236$, and for a lognormal severity the skewness is $(1 + \nu_X^2)^{3/2}/\sqrt{100} = 5^{3/2}/10 = 1.118$. The shifted gamma therefore has shape $\alpha = 3.2$, scale $\theta = 1{,}250$, and shift $s = 6{,}000$, the parameters shown in the note of the approximated aggregate. In the qd description the frequency is the fixed count of one, and the severity and the aggregate are the same distribution.

```
book = 'agg {} 100 claims sev lognorm 100 cv 2 poisson {}'
s = build(book.format('BookSG', 'approximate sgamma'))
qd(s)
print(repr(s.approximation))
wprint(s.note)
```

```
Aggregate object: BookSG. Frequency distribution fixed. Severity gamma,
[0, inf), exponential right tail. Updated with bucket size 2 and log2 =
16. Validation: not unreasonable.

      Mean     SD      CV  Skew  P01 Median   P99
X
Freq     1      0       0
Sev  10000 2236.1 0.22361 1.118 6634   9592 16928
Agg  10000 2236.1 0.22361 1.118 6634   9592 16928
'sgamma'
agg BookSG
  100 claims
  sev lognorm 100 cv 2
  poisson
  approximate sgamma  approximated by sgamma: gamma(loc=6000, scale=1250,
                      a=3.2), m=10000 cv=0.223607 skew=1.11803
```

Five approximations against the FFT answer

The property `approximation_df` fits all five families to an aggregate computed by FFT and sets each against the `exact` column, the FFT answer. Its rows form four blocks: `meta`, the fitted shape, shift (`loc`) and scale; `stats`, the mean, CV and skewness of each fit, and `ks`, its Kolmogorov distance $\sup_x |F(x) - G(x)|$ from the FFT distribution; `quantiles`, at probabilities from 0.001 to 0.999; and `rel err`, each quantile as a relative error against the FFT quantile. Every family matches the mean and CV. The fits use the moments of the computed distribution, whose skewness, 1.1108, is a little below the theoretical 1.118 because the grid cuts off the far tail of the lognormal severity.

```
a = build(book.format('Book', ''))
df = a.approximation_df
GT(df.loc['stats'])
```

measure	exact	norm	gamma	lognorm	sgamma	slognorm
mean	10000.0175	10000.0192	10000.0175	10000.0175	10000.0175	10000.0175
cv	0.2236	0.2236	0.2236	0.2236	0.2236	0.2236
skew	1.1108	0.0000	0.4472	0.6820	1.1108	1.1108
ks	0.0000	0.0509	0.0234	0.0114	0.0301	0.0206

In the right tail the two-parameter fits are too light, the more so the further out: at 0.999 the normal is 20.8% below the FFT quantile, the gamma 14.1%, and the lognormal 9.6%. Matching the skewness helps most far out: at 0.999 the shifted fits are 4.1% and 2.1% low, while at 0.99 they are 2.1% and 2.2% high, a little further off than the lognormal. The left tail behaves differently, because the shift bounds each shifted fit below, and their 0.01 quantiles are 11.5% and 7.1% too high. The lognormal, which does not match the skewness, has the smallest Kolmogorov distance, 0.011: matching the third moment improves the right tail but does not improve every measure of fit.

```
GT(df.loc['rel err'].loc[[0.01, 0.1, 0.5, 0.9, 0.99, 0.999]], ratio_cols='all')
```

measure	exact	norm	gamma	lognorm	sgamma	slognorm
0.01	0.0%	-19.2%	-6.7%	-1.8%	11.5%	7.1%
0.1	0.0%	-4.1%	-2.4%	-1.2%	1.2%	1.1%
0.5	0.0%	2.6%	0.9%	0.2%	-1.5%	-1.1%
0.9	0.0%	0.2%	0.9%	0.9%	1.3%	0.7%
0.99	0.0%	-8.3%	-3.9%	-1.6%	2.1%	2.2%
0.999	0.0%	-20.8%	-14.1%	-9.6%	-4.1%	-2.1%

Approximation error and the expected claim count

The skewness of a compound Poisson distribution falls as $1/\sqrt{\lambda}$ in the expected claim count λ, so an approximation improves as the count grows. The table repeats the relative errors at 0.99 and 0.999 for the same severity at 10, 100, 1,000, and 10,000 expected claims. At 10,000 claims every family is within 0.4%. At 10 claims the normal is 49% low at 0.999, and the shifted gamma is 15% high at 0.99. With skewness 3.53 its shape is $4/3.53^2 = 0.32$, and a gamma with shape below one has a density that is infinite at the shift, a poor model of a book of ten claims.

```
families = ['norm', 'gamma', 'lognorm', 'sgamma', 'slognorm']
rows = {}
for n in [10, 100, 1000, 10000]:
    d = build(f'agg N{n} {n} claims sev lognorm 100 cv 2 poisson').approximation_df
    for p in [0.99, 0.999]:
        rows[(n, p)] = [d.loc[('stats', 'skew'), 'exact'],
                        *d.loc[('rel err', p), families]]
err = pd.DataFrame.from_dict(rows, orient='index', columns=['skew', *families])
err.index = pd.MultiIndex.from_tuples(err.index, names=['claims', 'p'])
GT(err, ratio_cols=families, formatters={'skew': '.3f'})
```

claims	*p*	skew	norm	gamma	lognorm	sgamma	slognorm
10	0.9900	3.531	-23.8%	-4.4%	3.4%	15.0%	7.4%
	0.9990	3.531	-48.8%	-25.8%	-6.1%	4.2%	5.9%
100	0.9900	1.111	-8.3%	-3.9%	-1.6%	2.1%	2.2%
	0.9990	1.111	-20.8%	-14.1%	-9.6%	-4.1%	-2.1%
1,000	0.9900	0.353	-1.5%	-0.9%	-0.6%	0.0%	0.1%
	0.9990	0.353	-3.8%	-2.7%	-2.0%	-1.0%	-0.8%
10,000	0.9900	0.112	-0.2%	-0.1%	-0.1%	-0.0%	-0.0%
	0.9990	0.112	-0.4%	-0.3%	-0.2%	-0.1%	-0.1%

The approximate method

The method `approximate(approx_type, output)` fits one family to a built aggregate, by default the shifted lognormal, and returns it in the form given by `output`. The default, `'scipy'`, is a frozen `scipy.stats` distribution. The forms `'sev_decl'` and `'agg_decl'` are DecL, a severity clause or a complete program on a fixed count of one claim, and `'agg'` builds that program. With `approx_type='all'` the result is a dictionary with one entry per family. The method fits the moments of the computed distribution, as `approximation_df` does, so the 0.999 quantile of the frozen shifted lognormal equals the `slognorm` entry in the `quantiles` block.

```
fz = a.approximate()
print(f'{fz.ppf(0.999):,.1f}  {df.loc[("quantiles", 0.999), "slognorm"]:,.1f}')
wprint(a.approximate('sgamma', 'agg_decl'))
```

```
20,917.6  20,917.6
agg sgam.Book 1 claim sev 1241.8670417296307 * gamma 3.2417855883847233 +
5974.150788235019 fixed
```

Approximating one unit of a portfolio

A portfolio computes every unit on one grid, and a unit with many small claims can be poorly served by a bucket chosen for the others. In the portfolio below the bucket is 40, which suits the unit `Large`, ten claims with mean 5,000, but not `Small`, 2,000 claims with mean 50. Computed exactly, the severity of `Small` has mean 48.1 on the grid, its aggregate mean is 96,230 rather than 100,000, and validation fails. With `approximate sgamma`, `Small` is one smooth distribution with standard deviation 4,031, which a bucket of 40 resolves, and the portfolio validates.

```
port = '''port {}
  agg Small 2000 claims sev lognorm 50 cv 1.5 poisson {}
  agg Large 10 claims sev lognorm 5000 cv 1 poisson'''
ports = {name: build(port.format(name, clause))
         for name, clause in [('Exact', ''), ('Mixed', 'approximate sgamma')]}
for name, q in ports.items():
    u = q['Small']
    wprint(f'{name}: bucket {q.bs:.0f}; Small severity mean {u.est_sev_m:,.1f}, '
           f'aggregate mean {u.est_m:,.0f}, sd {u.est_sd:,.0f}; '
           f'validation {q.validation_description}')
```

```
Exact: bucket 40; Small severity mean 48.1, aggregate mean 96,230, sd 4,063;
validation fails sev mean, agg mean
Mixed: bucket 40; Small severity mean 100,000.0, aggregate mean 100,000, sd
4,031; validation not unreasonable
```

On a portfolio, the method `approximate` and the property `approximation_df` work on the total, here the sum of a nearly normal unit and a unit of ten claims. The code prints the shifted lognormal fit to the total as DecL, then tabulates the relative errors. The shifted fits are within 1.6% of the FFT quantiles at 0.999, and the shifted lognormal is within 0.4% at all three probabilities shown.

```
mixed = ports['Mixed']
wprint(mixed.approximate('slognorm', 'agg_decl'))
GT(mixed.approximation_df.loc['rel err'].loc[[0.01, 0.99, 0.999]], ratio_cols='all')
```

```
agg slog.Mixed 1 claim sev 78954.16309631686 * lognorm 0.2721911145567305 +
68066.2108631491 fixed
```

measure	exact	norm	gamma	lognorm	sgamma	slognorm
0.01	0.0%	-11.6%	-6.9%	-4.9%	1.5%	0.1%
0.99	0.0%	-6.2%	-3.9%	-2.7%	0.1%	0.2%
0.999	0.0%	-12.6%	-8.7%	-6.3%	-1.5%	-0.4%

Refused programs

An approximated aggregate has no individual claims, so occurrence reinsurance is refused; aggregate reinsurance applies to the fitted distribution and is allowed. A kind outside the six is refused with the list of kinds.

```
lognormal = 'agg R 100 claims sev lognorm 100 cv 2'
refuse(f'{lognormal} occurrence net of 500 xs 500 poisson approximate sgamma',
       f'{lognormal} poisson approximate weibull')
```

```
DecL: approximate is incompatible with occurrence reinsurance (the
method-of-moments fit bypasses the per-occurrence convolution); use aggregate
reinsurance instead.
DecL: approximate 'weibull' is not recognised; use one of exact, norm, lognorm,
gamma, sgamma, slognorm
```