Using severity parameters

Shape parameters, scale and location, the mean and CV form, and the _ placeholder in a mixture of severity families

Stephen J. Mildenhall

2026-11-01

A continuous severity $X = cZ + b$ is a standard law Z with a scale c and a shift b. The law of Z depends only on its family and its **shape parameters**. A normal, exponential, or uniform Z has no shape parameter, a gamma, lognormal, or Pareto has one, and a beta or Burr has two.

In DecL the numbers after the family word are the shape parameters, in the order `scipy.stats` lists them. Scale and shift are arithmetic around the law, `c * family shapes + b`. A second form, `family mean cv v`, declares a mean and coefficient of variation (CV) and solves for the rest. These forms are used to declare a continuous `scipy.stats` family with up to two shape parameters as a severity, and to mix different families in one severity, where the placeholder `_`[1] holds the place of a family with no shape to give.

```
agg A 10 claims sev 30 * norm + 100 poisson                # normal, mean 100, sd 30
agg B 10 claims sev 25 * gamma 4 poisson                   # gamma, shape 4, scale 25
agg C 10 claims sev gamma 100 cv 0.5 poisson               # gamma, mean 100, CV 0.5
agg D 10 claims sev norm 100 cv 0.3 poisson                # the same law as A
agg E 10 claims sev [100 30] * [lognorm norm] [0.5 _] + [0 100] wts [0.6 0.4] poisson
```

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

```
1.60.0
```

A helper tabulates the mean, standard deviation, and CV of the frozen `scipy.stats` law behind a severity clause, one row for each family in the clause. The frozen law is the attribute `fz` of a severity.

```
def laws(*clauses):
    rows = {}
    for clause in clauses:
        sevs = build(f'agg T 1 claim sev {clause} fixed', update=False).sevs
        for sev in sevs:
            m, v = map(float, sev.fz.stats(moments='mv'))
            sd = v ** 0.5
            label = clause if len(sevs) == 1 else f'{clause}: {sev.sev_name}'
            rows[label] = [m, sd, sd / m if m else float('nan')]
    return pd.DataFrame.from_dict(rows, orient='index', columns=['mean', 'sd', 'cv']
                                  ).rename_axis('clause')
```

[1] From aggregate 1.51.0

Shape parameters, scale, and shift of a severity

The next table shows laws with none, one, and two shape parameters, scaled and shifted by arithmetic. A multiplier scales the mean and the standard deviation, and a shift moves the mean alone.

```
GT(laws('norm', '30 * norm + 100', '5 * expon', '10 * uniform', 'gamma 4',
        '25 * gamma 4', '25 * gamma 4 + 10', '100 * beta 2 3'))
```

clause	mean	sd	cv
norm	0	1.0000	—
30 * norm + 100	100	30.0000	0.3000
5 * expon	5	5.0000	1.0000
10 * uniform	5	2.8868	0.5774
gamma 4	4	2.0000	0.5000
25 * gamma 4	100	50.0000	0.5000
25 * gamma 4 + 10	110	50.0000	0.4545
100 * beta 2 3	40	20.0000	0.5000

The mean and CV form of a severity

The form `family mean cv v` sets the mean μ and CV υ directly. For a family with one shape parameter, the shape is solved from the CV and then the scale from the mean. The two shapes of a beta are solved the same way, on the interval from 0 to the scale. A family with no shape parameter, whose standard law has mean m_0 and standard deviation s_0, takes scale $\upsilon\mu/s_0$ and location $\mu - \text{scale} \cdot m_0$.

The table shows the declared values for each kind of family. A multiplier in front scales the declared mean, and a shift is added afterwards. An exponential with CV 1 needs no location, and with CV 0.5 it must be shifted by half its mean.

```
GT(laws('gamma 100 cv 0.5', '100 * beta 0.5 cv 0.2', 'norm 100 cv 0.3', '2 * norm 100 cv 0.3',
        'expon 10 cv 1', 'expon 10 cv 0.5', 'expon 10 cv 0.5 + 3', 'uniform 10 cv 0.2'))
```

clause	mean	sd	cv
gamma 100 cv 0.5	100	50.0000	0.5000
100 * beta 0.5 cv 0.2	50	10.0000	0.2000
norm 100 cv 0.3	100	30.0000	0.3000
2 * norm 100 cv 0.3	200	60.0000	0.3000
expon 10 cv 1	10	10.0000	1.0000
expon 10 cv 0.5	10	5.0000	0.5000
expon 10 cv 0.5 + 3	13	5.0000	0.3846
uniform 10 cv 0.2	10	2.0000	0.2000

The declared moments are those of the law before it is applied to losses. A normal or a widely spread shifted law can reach below zero, where `sev` clamps it to zero and `ssev` keeps it, as described in Using `ssev`. The normal with mean 100 and CV 1 has 16% of its mass below zero, so under `sev` the severity has a higher mean and a lower CV than were declared.

```
rows = {}
for keyword in ['sev', 'ssev']:
    s = build(f'agg T 1 claim {keyword} norm 100 cv 1 fixed', update=False).sevs[0]
    rows[keyword] = [s.actual_m, s.actual_sd, s.actual_cv]
GT(pd.DataFrame.from_dict(rows, orient='index', columns=['mean', 'sd', 'cv']
                          ).rename_axis('keyword'))
```

keyword	mean	sd	cv
sev	108.3315	86.6653	0.8000
ssev	100.0000	100.0000	1.0000

Mixtures of different severity families

A vector of families shares one list of numbers, and entry i of the list goes to family i. The placeholder _ holds the place of a family with no shape to give. In a mixture of a lognormal and a normal the normal has no shape, so the list is `[0.5 _]`. The scales and shifts are vectors in the same order. The next severity mixes a lognormal with $\sigma = 0.5$ scaled by 100 and a normal with mean 100 and standard deviation 30, and the table shows the law of each family.

```
mix = '[100 30] * [lognorm norm] [0.5 _] + [0 100] wts [0.6 0.4]'
GT(laws(mix))
```

clause	mean	sd	cv
[100 30] * [lognorm norm] [0.5 _] + [0 100] wts [0.6 0.4]: lognorm	113.3148	60.3901	0.5329
[100 30] * [lognorm norm] [0.5 _] + [0 100] wts [0.6 0.4]: norm	100.0000	30.0000	0.3000

A family with two shape parameters needs a second list, and _ holds a place there too. In a mixture of a lognormal and a beta, the lognormal's one shape goes in the first list, and _ holds its place in the second. The table shows the two laws.

```
GT(laws('[lognorm beta] [0.5 2] [_ 3] wts=2'))
```

clause	mean	sd	cv
[lognorm beta] [0.5 2] [_ 3] wts=2: lognorm	1.1331	0.6039	0.5329
[lognorm beta] [0.5 2] [_ 3] wts=2: beta	0.4000	0.2000	0.5000

The mixture is an ordinary severity for an aggregate, and the function `qd` displays the statistics of the aggregate.

```
qd(build(f'agg Mix 10 claims sev {mix} poisson'))
```

```
Aggregate object: Mix. Frequency distribution poisson. Mixed severity
with 2 components. Updated with bucket size 1/8 and log2 = 16.
Validation: not unreasonable.

       Mean      SD       CV     Skew     P01 Median     P99
```

```
X
Freq     10 3.1623 0.31623 0.31623
Sev  107.99 50.896  0.4713  1.9534     31    100 289.75
Agg  1079.9 377.52 0.34959  0.4379 329.38 1052.4 2074.9
```

Refused severity clauses

Numbers a family cannot take are refused and never dropped, in a `sev` clause as in a `chfagg` statement, and the error message gives the form to use where there is one. A placeholder must leave each family with the shapes it takes. A mean and CV the family cannot reach are refused too, and so is a family that does not exist, with the closest names. The class `Severity` refuses the same mistakes where the values show them.

```
refuse(*[f'agg T 1 claim sev {clause} fixed' for clause in [
    'norm 100 30', 'expon 5', 'gamma 4 25', 'beta 2', 'gamma',
    'burr 100 cv 0.5', 'cauchy 10 cv 1',
    '[lognorm norm] [0.5 17]', '[expon pareto] [1 3]', '[lognorm lognorm] [0.5 _]',
    'norm -100 cv 0.3', 'beta 2 cv 0.2', 'lognorml 0.5']],
    lambda: Severity('gamma', sev_a=4, sev_b=25),
    lambda: Severity('norm', sev_a=100))
```

```
sev: norm has no shape parameter; for location 100 and scale 30 write '30 * norm
+ 100', or 'norm 100 cv 0.3'.
sev: expon has no shape parameter; for scale 5 write '5 * expon', for a shift
'expon + 5'.
sev: gamma takes one shape parameter; for shape 4 and scale 25 write '25 * gamma
4', and if 25 is a cv, 'gamma 4 cv 25'.
sev: beta takes two shape parameters, got one. Or give a mean and cv: beta m cv
v.
sev: gamma takes one shape parameter, got none. Or give a mean and cv: gamma m
cv v.
sev: burr has no mean cv form; give its two shapes.
sev: cauchy has no finite mean, so no mean cv form.
sev: norm has no shape parameter, and in a vector of families '_' holds its
place: write '[lognorm norm] [0.5 _]'.
sev: expon has no shape parameter, and in a vector of families '_' holds its
place: write '[expon pareto] [_ 3]'.
sev: lognorm takes one shape parameter, got none in this vector of families.
norm: a mean and cv form needs both positive, got mean -100.0 and cv 0.3.
Otherwise give a scale and shift: c * norm + b.
beta has no law with mean 2.0 and cv 0.2 on [0, 1.0]: the mean over the scale,
m, must be below 1 and the cv squared below (1 - m) / m.
DecL parse error at line 1, column 19: Unknown severity family 'lognorml'. Did
you mean: lognorm, loguniform, powerlognorm?
gamma takes one shape parameter, got sev_b=25; use sev_scale.
norm has no shape parameter, got sev_a=100; use sev_scale and sev_loc.
```