Using `clash`

Clash covers on two units hit by the same events with the clash statement

Stephen J. Mildenhall

2026-10-30

A **clash** model describes two units hit by the same events, such as two lines of business exposed to the same storms. An event can reach unit A, unit B, both, or neither, and one that reaches both is a **clash event**. The DecL statement `clash` specifies the model by three expected counts a year, n_a events that reach A only, n_b that reach B only, and n_c that reach both, and a severity for each unit. It returns a `BivariateAggregate`, the joint distribution of the annual losses X and Y of the two units, described in Using `bivariate`.

Within an event the two units are reached independently, with probabilities p_a and p_b. Independence fixes the fourth cell of the two by two table of expected counts, the events that reach neither unit, through $n_c n_0 = n_a n_b$. Hence

$$n_0 = \frac{n_a n_b}{n_c}, \qquad n = n_a + n_b + n_c + n_0, \qquad p_a = \frac{n_a + n_c}{n}, \qquad p_b = \frac{n_b + n_c}{n},$$

where n is the expected shared event count. Each unit becomes an `agg` with the Bernoulli frequency `dfreq [0 1] [1-p p]`, and the two are coupled by the independent copula on the shared count. All the dependence between X and Y comes from shared events. For unit severity means m_A and m_B and a gamma mixed Poisson count with mixing coefficient of variation c,

$$\operatorname{cov}(X, Y) = \big(n_c + c^2(n_a + n_c)(n_b + n_c)\big)\, m_A m_B,$$

where the first term is the clash events and the second the common shock of the mixing; under a Poisson count, $c = 0$.

Each unit is given by an optional limit clause and a severity clause, and the shared frequency follows, Poisson by default. A **clash cover** pays per event on the combined loss of both units, in excess of a retention. The `clash` statement computes the annual pair rather than the combined loss of each event, so the cover is built on this page from the per-event components.

```
clash Cat 8 5 2 claims sev lognorm 50 cv 0.6 sev lognorm 60 cv 0.8 mixed gamma 0.2
clash Cat 8 5 2 claims 500 xs 0 sev lognorm 50 cv 0.6 500 xs 0 sev lognorm 60 cv 0.8 poisson
```

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

```
1.43.1
```

Shared event count and per-event triggers

The example is the entry `BivariateClash` of the recipe library (see Using recipes): 8 events a year reach A only, 5 reach B only, and 2 reach both, with lognormal severities of means 50 and 60 and coefficients of variation 0.6 and 0.8, and a gamma mixed count with mixing coefficient of variation 0.2. The attribute `clash` holds the solved model. There are $8 \times 5/2 = 20$ events a year that reach neither unit, the shared count has mean 35, and the triggers are $p_a = 10/35 = 0.286$ and $p_b = 7/35 = 0.2$.

```
sevs = 'sev lognorm 50 cv 0.6 sev lognorm 60 cv 0.8 mixed gamma 0.2'
c = build(f'clash Cat 8 5 2 claims {sevs}')
print(c.en)
wprint(c.clash)
```

```
35.0
{'na': 8.0, 'nb': 5.0, 'nc': 2.0, 'n0': 20.0, 'pa': 0.2857142857142857, 'pb':
0.2}
```

The components are named `Cat.A` and `Cat.B`. Unit A is reached by $8 + 2 = 10$ events a year, so its mean annual loss is $10 \times 50 = 500$, and unit B by 7, with mean $7 \times 60 = 420$.

```
qd(c)
```

```
BivariateAggregate(name='Cat', units=['Cat.A', 'Cat.B'],
Copula('independent'), shape=(1024, 1024), corr=0.3120)

       Mean     SD      CV    Skew  P01  Median  P99
unit
Cat.A   500 209.79 0.41959 0.60508  112     480 1076
Cat.B   420    220  0.5238 0.84458   45     390 1060
total   920 348.15 0.37843 0.59124  265     885 1875
```

Dependence from shared events

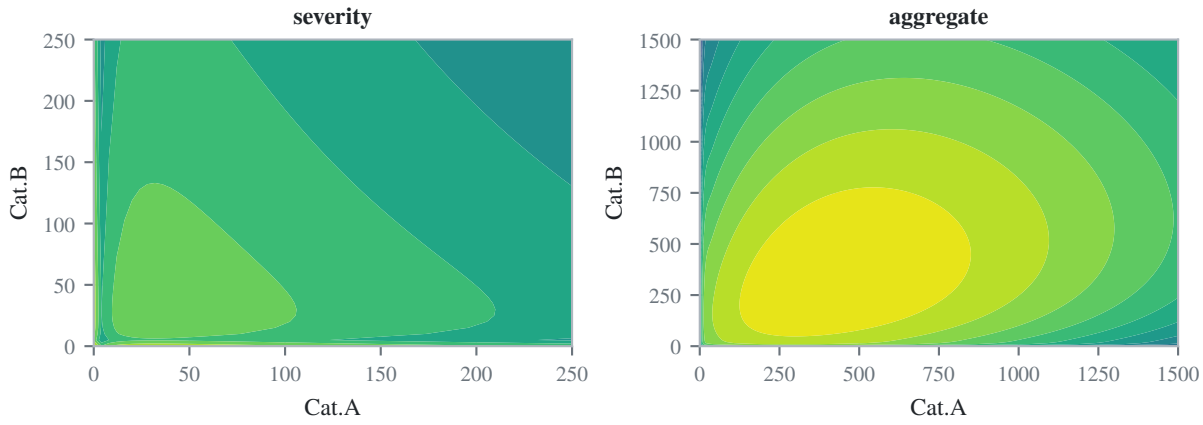
Within an event the two units are independent, so the per-event correlation is zero, while the annual correlation is 0.312. The covariance formula gives $(2 + 0.2^2 \times 10 \times 7) \times 50 \times 60 = 14{,}400$: 6,000 from the two clash events and 8,400 from the mixing.

```
d = c.dependency_df
print(round(d.loc['Sev', 'corr'], 6) + 0, d.loc['Agg', 'corr'])
print(d.loc['Agg', 'cov'], (2 + 0.2**2 * 10 * 7) * 50 * 60)
```

```
0.0 0.31199931266263525
14399.983119090437 14400.000000000002
```

The method `plot` draws the per-event law on the left and the annual joint law on the right, here with a log color scale. On the left, events that reach one unit lie along the axes and clash events fill the interior.

```
fig, (ax0, ax1) = figure('full', cols=2)
c.plot(axs=(ax0, ax1), log=True)
ax0.set(xlim=[0, 250], ylim=[0, 250])
ax1.set(xlim=[0, 1500], ylim=[0, 1500]);
```



A clash cover on the combined event loss

A clash cover `200 xs 150` pays $\min((A+B-150)^+, 200)$ on each event, where A and B are the losses of the two units from that event, either of which may be zero. The per-event combined loss $A+B$ is the total of a portfolio of the two Bernoulli components, with the solved triggers, so an `agg` with the shared count and `sev port.Event` compounds it, and an `occurrence` clause applies the cover. A severity reference needs its source to fix its own grid, hence the `hints` clause. The function below builds the clash, the gross compound and the cover for a set of counts.

```
def clash_cover(counts, layer='200 xs 150'):
    c = build(f'clash Cat {counts} claims {sevs}')
    pa, pb = c.clash['pa'], c.clash['pb']
    build(f'''port Event hints{{log2=12; bs=1/2}}
      agg Event.A dfreq [0 1] [{1 - pa} {pa}] sev lognorm 50 cv 0.6
      agg Event.B dfreq [0 1] [{1 - pb} {pb}] sev lognorm 60 cv 0.8''')
    gross = build(f'agg Gross {c.en} claims sev port.Event mixed gamma 0.2')
    cover = build(f'agg Cover {c.en} claims sev port.Event '
                  f'occurrence ceded to {layer} mixed gamma 0.2')
    return c, gross, cover

results = {counts: clash_cover(counts) for counts in ['8 5 2', '9.5 6.5 0.5']}
```

Without the cover, the compound of the combined event loss is the annual total $X+Y$ of the clash: both have mean 920 and 99th percentile 1,875, to the grid.

```
c, gross, cover = results['8 5 2']
print(c.total.mean(), gross.est_m)
print(c.total.q(0.99), gross.q(0.99))
```

```
919.9995144451415 919.9993396539279
1875.0 1875.5
```

The second set of counts, 9.5, 6.5, and 0.5, keeps each unit's events at 10 and 7, so the two marginals are unchanged, but has a quarter of the clash events. The table shows the annual correlation and the mean, standard deviation, and probability of no recovery of the cover. Fewer clash events lower the correlation from 0.312 to 0.214 and the cost of the cover from 34.99 to 26.04.

```
df = pd.DataFrame.from_dict(
    {k: [c.clash['nc'], c.en, c.corr, cover.est_m, cover.est_sd,
         cover.density_df.p_total.iloc[0]] for k, (c, gross, cover) in results.items()},
    orient='index', columns=['clash events', 'events', 'corr', 'cover mean',
                             'cover SD', 'Pr(no recovery)'])
df.index.name = 'counts'
GT(df, float_format=',.3f')
```

counts	clash events	events	corr	cover mean	cover SD	Pr(no recovery)
8 5 2	2.000	35	0.312	34.993	59.160	0.502
9.5 6.5 0.5	0.500	140	0.214	26.035	51.246	0.598

The same layer bought on each unit separately applies to the loss of one unit in each event, in 10 events a year for A and 7 for B. Together the two pay 23.05 a year, which is what the clash cover would cost with no clash events. Each expected clash event adds $(34.993 - 23.050)/2 = 5.97$ to the cost, and the second set of counts gives the same, $(26.035 - 23.050)/0.5 = 5.97$.

```
per_unit = [build(f'agg {u} {n} claims sev lognorm {m} cv {v} '
                  'occurrence ceded to 200 xs 150 mixed gamma 0.2').est_m
            for u, n, m, v in [('A', 10, 50, 0.6), ('B', 7, 60, 0.8)]]
print(*per_unit, sum(per_unit))
for k, (c, gross, cover) in results.items():
    print(k, (cover.est_m - sum(per_unit)) / c.clash['nc'])
```

```
4.120041844422314 18.929654408650748 23.049696253073062
8 5 2 5.971820469938377
9.5 6.5 0.5 5.971586027564847
```

Refused programs

The clash count must be positive, because $n_0 = n_a n_b / n_c$; two units with no clash events are two separate aggregates. The triggers are independent within an event, so a copula clause is a parse error.

```
refuse(f'clash Cat 10 7 0 claims {sevs}',
       'clash Cat 8 5 2 claims sev lognorm 50 cv 0.6 sev lognorm 60 cv 0.8 copula gumbel 0.4')
```

```
clash: nc (the clash count) must be positive.
DecL parse error at line 1, column 68: Unexpected 'copula'.
```