# Using validation

**The validation verdict on a computed object, reinsured views, and the warning a failing build emits**

Stephen J. Mildenhall

2026-11-01

A distribution computed on a grid can be wrong in ways that are hard to see. A bucket that misses the atoms of a severity moves probability between outcomes, and the mean can still come out right. The **validation verdict** compares what the grid holds with what the model implies and reports the result in one line. It is used to decide whether to rely on a computed object and, if not, what to change.

The verdict compares the mean, coefficient of variation (CV), and skewness of the severity and of the aggregate on the grid with their analytic values. A relative error above $\epsilon =$ `validation_eps` ($10^{-4}$ by default) fails a mean, above $10\epsilon$ a CV, and above $100\epsilon$ a skewness. A pass reads `not unreasonable`, because agreement of three moments cannot prove a distribution right. The attribute `valid` holds the flags, and `valid.passes` is true when nothing failed. The words are the strings `validation_description` and `validation_explanation` (Using explanations).

The examples use these programs.

```
agg L 3 claims 1500 xs 250 dsev [1000 2000 3000] fixed
agg X 1 claims dsev [1000 2000 3000] occurrence net of 1500 xs 250 fixed
agg D 100 claims dsev [999 1000 1001] [0.333333 0.222222 0.444444] fixed
agg ZeroMean 1 claim dsev [-1 1] fixed
```

```
import warnings
from aggregate import build, __version__
from aggregate.constants import ValidationWarning
from style import wprint, figures
figures()
print(__version__)
```

```
1.58.0
```

## The warning from a failing build

The warning `ValidationWarning`[1] reports a build whose verdict fails, with the verdict as its text. The aggregate `L` has three claims from a discrete severity limited to 1500 xs 250, which puts the outcomes on multiples of 250. Pinned to a bucket of 1000, the outcomes fall between grid points and the mean is wrong. Left to the sizer, the bucket lands on the outcomes and the same program passes.

[1] From aggregate 1.58.0

```
program = 'agg L 3 claims 1500 xs 250 dsev [1000 2000 3000] fixed'
with warnings.catch_warnings(record=True) as caught:
    warnings.simplefilter('always')
    a = build(program, bs=1000, log2=8)
    b = build(program, bs=1000, log2=8)
print(len(caught))
wprint(str(caught[0].message))
print(a.actual_m, a.est_m, a.valid.passes)
c = build(program)
print(c.bs, c.log2, c.est_m, c.validation_description)
```

```
1
L: fails sev mean, agg mean. See validation_explanation. Further occurrences
this session are suppressed; each object records its own verdict in validation.
3750.0 3000.0000000000105 False
50.0 7 3750.0 not unreasonable
```

The warning appears once in a session for a given name and failure, so a sweep of one program reports once, and each object keeps its own verdict. A `Portfolio` warns once for itself and its units. A build the library makes on the way to something else, such as a cell of the probe in Using `sharpen` or the inner object of `sev agg.NAME`, does not warn. The warning can be turned off with `silence_warnings(ValidationWarning)`.

### Validating a reinsured view

An object with reinsurance reports a net or ceded view, and its verdict has two clauses. The first compares the gross subject with its analytic moments. The second compares the reported view with the exact image of the subject on the grid: the net or ceded amount of each bucket, before that amount is placed back on the grid. The difference is the error of placing it there. The word in the second clause is `net` or `ceded`, or `output` when the occurrence and aggregate covers report different views.

The aggregate `X` has one claim of 1000, 2000, or 3000, and its net outcomes under 1500 xs 250 are 250, 500, and 1500. On a bucket of 1000 the subject is exact and the net mean of 750 is right, but the net CV is 0.96 against the true 0.72. On the bucket the sizer selects, both clauses pass.

```
program = 'agg X 1 claims dsev [1000 2000 3000] occurrence net of 1500 xs 250 fixed'
with warnings.catch_warnings():
    warnings.simplefilter('ignore', ValidationWarning)
    coarse = build(program, bs=1000, log2=8)
wprint(coarse.validation_description)
print(coarse.est_m, round(coarse.est_cv, 4), coarse.valid.passes)
fine = build(program)
wprint(fine.validation_description)
print(fine.bs, fine.est_m, round(fine.est_cv, 4), fine.valid.passes)
```

```
reinsurance; subject not unreasonable; net fails sev cv, agg cv
750.0 0.9623 False
reinsurance; subject not unreasonable; net not unreasonable
50.0 750.0 0.7201 True
```

Either clause failing fails the object. A failing subject with a clean view still means the gross density is wrong, along with anything priced on it. For a continuous severity the image on the grid is the best reference available rather than the exact law, so the second clause measures only the placement.

### Skewness when the mean is large against the spread

The skewness is computed from the raw moments through the third central moment, $\mathsf{P}X^3 - 3\,\mathsf{P}X\,\mathsf{P}X^2 + 2(\mathsf{P}X)^3$, which subtracts nearly equal numbers when the mean is large against the spread. The grid's skewness then carries a rounding error that no bucket removes, on an exact law as much as on a wrong one. When the third central moment is below $10^{-12}$ of the terms subtracted, the skewness cannot be held to the 1% the test requires, so it is reported and not tested.

The aggregate `D` is exact on its grid, with a mean of 100,000 and a CV of $9\times10^{-5}$. Its analytic skewness is -0.021, close to the true -0.022. The skewness on the grid is printed second and is well away from both, and it varies with the numerical library in use.

```
d = build('agg D 100 claims dsev [999 1000 1001] [0.333333 0.222222 0.444444] fixed')
st = d.stats_df.loc[('agg', 'skew')]
print(round(st['mixed'], 4), round(st['empirical'], 4), d.validation_description)
```

```
-0.0213 -0.0164 not unreasonable
```

### Moments that do not exist

A CV or skewness can be `nan` because the law has no such moment: a CV at a zero mean, a skewness at a zero variance, and either one after an infinite moment. The verdict skips those comparisons. A `nan` the library failed to compute is different, from a negative variance or a `nan` raw moment. That verdict opens with `analytic cv, skew undefined`, and `passes` is false, since nothing has checked those moments.

```
z = build('agg ZeroMean 1 claim dsev [-1 1] fixed')
print(z.stats_df.loc[('agg', 'cv'), 'mixed'], z.validation_description, z.valid.passes)
```

```
nan not unreasonable True
```

A law computed from its characteristic function uses the same words, with its own checks on the inversion (Using `AggregateCHF`). On a stable law with a mean, the verdict compares the mean on the window plus its closed-form tail with the theoretical mean (Using `AggregateCHF` with stable laws).