Using `tweedie`

Tweedie parameter conversions and named laws with the tweedie statement

Stephen J. Mildenhall

2026-10-30

A Tweedie distribution with power variance function $V(\mu) = \mu^p$, $1 < p < 2$, has two common parameterizations. A GLM uses the reproductive form $\mathrm{Tw}_p(\mu, \phi)$, with mean μ, dispersion ϕ and variance $\phi\mu^p$. An actuary uses the compound Poisson form: an expected claim count λ and a gamma severity with mean m and coefficient of variation (CV) ν_X. The two are linked by

$$\lambda = \frac{\mu^{2-p}}{(2-p)\phi}, \qquad \alpha = \frac{2-p}{p-1}, \qquad m = \mu/\lambda, \qquad \nu_X = \alpha^{-1/2},$$

where α is the gamma shape. The Tweedie CV is $\nu = \sqrt{\phi}\,\mu^{p/2-1}$, and the probability of no loss is $e^{-\lambda}$.

The DecL statement `tweedie`[1] converts between these parameterizations. It accepts three forms and returns a `TweedieConversion`. It is used to find the claim count and severity implied by a fitted Tweedie GLM. Without a name the statement stores nothing. With a name, as in `tweedie t0 1.5 100 2`, it stores the law in the recipe base, and the reference `tweedie.t0` returns it, as described in Using `stable` names and arithmetic.

```
tweedie 1.5 100 2                       # p, mean, dispersion
tweedie 1.5 100 cv 0.447214             # p, mean, CV
tweedie 10 claims sev gamma 10 cv 1     # Poisson count and gamma severity
tweedie t0 1.5 100 2                    # the first law, stored as tweedie.t0
```

```
from aggregate import build, qd, TweedieConversion, __version__
from style import wprint, refuse, figures
figures()
print(__version__)
```

```
1.43.1
```

Converting between Tweedie parameterizations

Each of the three forms of the `tweedie` statement returns a `TweedieConversion`. Printed, it shows the distribution in all three forms, joined by =, followed by the probability of no loss, to six significant figures. Each form is valid DecL and can be pasted back into a program.

[1] From aggregate 1.22.0

```
t = build('tweedie 1.5 100 2')
wprint(t)
```

```
tweedie 1.5 100 2 = tweedie 1.5 100 cv 0.447214 = tweedie 10 claims sev gamma 10
cv 1, Pr(X=0) 4.53999e-05
```

A `TweedieConversion` is a named tuple of full-precision floats. Its fields are available for further calculation, and are not rounded as the printed form is.

```
t
```

```
TweedieConversion(p=1.5, mean=100.0, dispersion=2.0, cv=0.447213595499958, frequency=10.0,
sev_m=10.0, sev_cv=1.0, p0=4.5399929762484854e-05)
```

The four programs below describe the same distribution, and each returns an expected claim count of 10. The compound form parses its severity with the same `sev` clause as an `agg`, so any gamma severity an `agg` accepts is accepted here, including `sev 10 * gamma 1`. The CV form reproduces the parameters only to the six figures given, and its count is 9.99998.

```
for program in ['tweedie 1.5 100 2',
                'tweedie 1.5 100 cv 0.447214',
                'tweedie 10 claims sev gamma 10 cv 1',
                'tweedie 10 claims sev 10 * gamma 1']:
    print(f'{program:40s}', build(program).frequency)
```

```
tweedie 1.5 100 2                        10.0
tweedie 1.5 100 cv 0.447214              9.999981910232723
tweedie 10 claims sev gamma 10 cv 1      10.0
tweedie 10 claims sev 10 * gamma 1       10.0
```

Frequency and severity implied by a Tweedie GLM

A fitted Tweedie GLM gives the power p and the dispersion ϕ, and the linear predictor gives the mean μ for each risk. The `tweedie` statement converts these to the expected claim count and gamma severity they imply, shown here for three risks.

```
for mean in [50, 250, 1000]:
    wprint(build(f'tweedie 1.65 {mean} 30'))
```

```
tweedie 1.65 50 30 = tweedie 1.65 50 cv 2.76211 = tweedie 0.374499 claims sev
gamma 133.512 cv 1.36277, Pr(X=0) 0.687634
tweedie 1.65 250 30 = tweedie 1.65 250 cv 2.08411 = tweedie 0.657794 claims sev
gamma 380.058 cv 1.36277, Pr(X=0) 0.517993
tweedie 1.65 1000 30 = tweedie 1.65 1000 cv 1.63516 = tweedie 1.06859 claims sev
gamma 935.813 cv 1.36277, Pr(X=0) 0.343493
```

With p and ϕ fixed, the severity CV depends on p alone, $\nu_X^2 = (p-1)/(2-p)$. A larger mean comes from both more claims and larger claims, with $\lambda \propto \mu^{2-p}$ and $m \propto \mu^{p-1}$.

The tweedie clause in an aggregate

Inside an `agg` program, the `tweedie` clause takes the dispersion form or the `cv` form. The stored program keeps the form as written, as `pprogram` shows.

```
a = build('agg TW tweedie 1.5 100 cv 0.447214')
qd(a)
print(a.pprogram)
```

```
Aggregate object: TW. Frequency distribution poisson. Severity gamma,
[0, inf), exponential right tail. Updated with bucket size 1/64 and log2
= 16. Validation: not unreasonable.

      Mean     SD      CV    Skew     P01 Median    P99
X
Freq    10 3.1623 0.31623 0.31623
Sev     10     10       1       2 0.09375 6.9375 46.047
Agg    100 44.721 0.44721 0.67082  19.594 94.953 224.94
agg TW
  tweedie 1.5 100 cv 0.447214
```

The aggregate is computed as a compound Poisson gamma, so its variance is $\phi\mu^p$ and its probability of no loss is $e^{-\lambda}$. The first bucket is slightly larger than $e^{-\lambda}$, because it also includes the gamma probability below half a bucket.

```
t = build('tweedie 1.5 100 cv 0.447214')
print(a.density_df.p_total.iloc[0], t.p0)
print(a.est_sd ** 2, t.dispersion * t.mean ** t.p)
```

```
4.575669265299114e-05 4.540075104407702e-05
2000.0038214101514 2000.00361796
```

The method `Aggregate.as_tweedie()` returns the reproductive parameters (p, μ, ϕ) of any compound Poisson gamma aggregate, however it was specified. The constructor `TweedieConversion.from_dispersion` converts them to the full conversion.

```
b = build('agg Longhand 10 claims sev gamma 10 cv 1 poisson')
print(b.as_tweedie())
wprint(TweedieConversion.from_dispersion(*b.as_tweedie()))
```

```
TweedieParameters(p=1.5, mean=100.0, dispersion=2.0)
tweedie 1.5 100 2 = tweedie 1.5 100 cv 0.447214 = tweedie 10 claims sev gamma 10
cv 1, Pr(X=0) 4.53999e-05
```

The TweedieConversion constructors

The class `TweedieConversion` has one constructor for each form of the statement: `from_dispersion`, `from_cv`, and `from_compound`. Each returns the same object as the corresponding DecL statement, so the conversion is available in Python without writing a program.

```
print(TweedieConversion.from_cv(1.5, 100, 0.447214) == build('tweedie 1.5 100 cv 0.447214'))
wprint(TweedieConversion.from_compound(10, 10, 1))
```

```
True
tweedie 1.5 100 2 = tweedie 1.5 100 cv 0.447214 = tweedie 10 claims sev gamma 10
cv 1, Pr(X=0) 4.53999e-05
```

Refused programs

A power outside $1 < p < 2$ is refused, because it has no compound Poisson form, and so is a compound form whose severity is not an unshifted gamma, such as a lognormal or a shifted gamma. Each of these raises a `ValueError` that identifies the form. The other powers are available through a `chfagg`, as described in Using `AggregateCHF`. An `agg` takes the `tweedie` clause but not a reference such as `tweedie.t0`.

```
refuse('tweedie 2.5 100 2',
       'tweedie 10 claims sev lognorm 10 cv 1',
       'tweedie 10 claims sev gamma 10 cv 1 + 5',
       'agg A tweedie.t0')
```

```
tweedie: p must be strictly between 1 and 2, got 2.5. The clause is `tweedie <p>
<mean> <dispersion>` or `tweedie <p> <mean> cv <cv>`, shape parameter first. (It
read `<mean> <p> <dispersion>` before 1.0.0a231, so an older program needs its
first two numbers swapped.)
tweedie: the compound form needs a plain gamma severity, `tweedie <count> claims
sev gamma <mean> cv <cv>`, but found severity 'lognorm'.
tweedie: the compound form needs a plain gamma severity, `tweedie <count> claims
sev gamma <mean> cv <cv>`, but found sev_loc = 5.0.
DecL parse error at line 1, column 7: Unexpected 'tweedie.t0'. Did you mean:
tweedie?
```