Using deductibles

Franchise and disappearing deductibles in the DecL layers clause

Stephen J. Mildenhall

2026-10-30

An ordinary deductible `y xs d1` retains the first d_1 of every loss. A franchise deductible retains a loss of at most d_1 and nothing of a larger one. A disappearing deductible sits between them: it retains all of d_1 at a loss of d_1, less as the loss grows, and nothing once the loss reaches the **retention end** d_2. The three are one family of retentions,

$$r(x) = \begin{cases} d_1 & x \le d_1 \\ d_1 \dfrac{d_2 - x}{d_2 - d_1} & d_1 < x \le d_2 \\ 0 & x > d_2, \end{cases}$$

and the layer pays $g(x) = \min((x - r(x))^+, y)$: deductible first, limit second, so y is the largest payment in every form. The ordinary deductible is $d_2 = \infty$, with $g(x) = \min((x - d_1)^+, y)$, and the franchise is $d_2 = d_1$, with $g(x) = \min(x, y)$ for $x > d_1$. In between, g rises with slope $k = d_2/(d_2 - d_1)$ from zero at d_1 to the whole loss at d_2.

The layers clause takes `y xs d1 franchise`[1] and `y xs d1 to d2 disappearing`[2]. The description of the severity shows the form in the layer phrase, as `80 xs 20 franchise`. As for an ordinary layer, the claim count is the count of losses above d_1 and the severity is conditional on $X > d_1$; a `!` after the severity makes it ground up.

```
agg Ordinary 10 claims 80 xs 20 sev 100 * uniform poisson
80 xs 20 franchise                          # in place of 80 xs 20
80 xs 20 to 60 disappearing                 # no retention above 60
[40 80] xs [10 20] to [30 60] disappearing  # one end per layer
```

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

```
1.43.1
```

[1] From aggregate 1.7.0
[2] From aggregate 1.11.0

Three deductible forms on a uniform loss

Ten claims a year above 20, from a ground-up loss uniform on $[0, 100]$, under an 80 xs 20 layer, make each deductible form easy to follow. The three aggregates differ only in the layers clause, and the severity in the `qd` description shows the form, here `80 xs 20 franchise`.

```
forms = {'Ordinary': '', 'Franchise': 'franchise',
         'Disappearing': 'to 60 disappearing'}
book = 'agg {} 10 claims 80 xs 20 {} sev 100 * uniform {} poisson'
books = {k: build(book.format(k, v, '')) for k, v in forms.items()}
qd(books['Franchise'])
```

```
Aggregate object: Franchise. Frequency distribution poisson. Severity
uniform, [0, 80], bounded; 80 xs 20 franchise. Updated with bucket size
1/16 and log2 = 16. Validation: not unreasonable.

      Mean     SD      CV     Skew    P01 Median    P99
X
Freq    10 3.1623 0.31623  0.31623
Sev   57.5 19.843  0.3451 -0.32397 20.812     60     80
Agg    575 192.35 0.33453  0.35899 182.44 563.44 1070.4
```

The same layer phrase appears as the declared support in the `tail_explanation` of each severity. The ordinary layer shows `80 xs 20`, with no suffix.

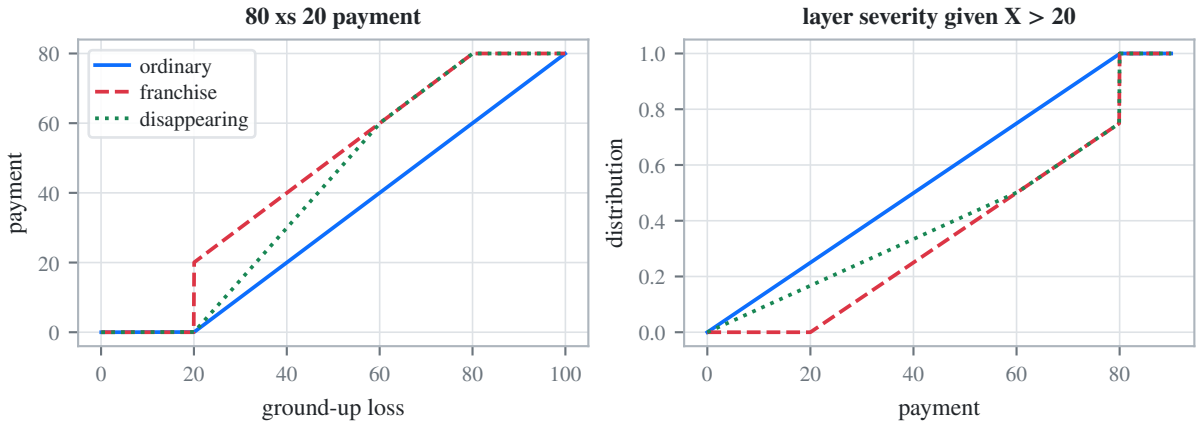
```
for k, a in books.items():
    wprint(a.sevs[0].tail_explanation)
```

```
Uniform severity [0, 80], bounded. Declared support: 80 xs 20. Support is
bounded, so every moment is finite.
Uniform severity [0, 80], bounded. Declared support: 80 xs 20 franchise. Support
is bounded, so every moment is finite.
Uniform severity [0, 80], bounded. Declared support: 80 xs 20 to 60
disappearing. Support is bounded, so every moment is finite.
```

Payment function and layer severity

For a loss uniform on $[0, 100]$ the ground-up quantile at p is $100p$, so the quantile function of the ground-up payment (declared with `!`) is the payment function itself: $g(x)$ is `ppf(x / 100)`. The left panel shows g for each form, and the right panel shows the distribution of the payment given $X > 20$, which is the severity the aggregate convolves.

```
ground_up = {k: build(book.format(k + 'GU', v, '!'), update=False)
             for k, v in forms.items()}
x = np.linspace(0.05, 99.95, 1000)
t = np.linspace(0, 90, 1000)
fig, (ax0, ax1) = figure('full', cols=2)
for (k, a), ls in zip(books.items(), ['-', '--', ':']):
    ax0.plot(x, ground_up[k].sevs[0].ppf(x / 100), ls=ls, label=k.lower())
    ax1.plot(t, a.sevs[0].cdf(t), ls=ls, label=k.lower())
ax0.set(xlabel='ground-up loss', ylabel='payment', title='80 xs 20 payment')
ax1.set(xlabel='payment', ylabel='distribution',
        title='layer severity given X > 20')
ax0.legend();
```



With a franchise deductible the layer pays nothing up to 20 and then the whole loss, so no payment lies in $(0, 20)$. With the disappearing deductible it pays $1.5(x - 20)$ up to 60 and then the whole loss, the same as the franchise above 60. Both reach the limit at a loss of 80, the layer's `detachment`, and put mass 1/4 on it; the ordinary layer reaches the limit only at 100.

```
for k, a in books.items():
    s = a.sevs[0]
    print(f'{k:13s} detachment {s.detachment:4.0f}   '
          f'mass at the limit {s.sf(79.999):.4f}')
```

```
Ordinary      detachment  100   mass at the limit 0.0000
Franchise     detachment   80   mass at the limit 0.2500
Disappearing  detachment   80   mass at the limit 0.2500
```

Layer means against a hand calculation

The hand calculation of the layer means starts from the loss uniform on $[0, 100]$, which given $X > 20$ is uniform on $(20, 100)$. The ordinary layer pays $X - 20$, uniform on $(0, 80)$, mean 40. The franchise layer pays X up to 80 and 80 above, mean $\int_{20}^{80} x\,dx/80 + 80/4 = 57.5$. The disappearing deductible layer has mean $\int_{20}^{60} 1.5(x - 20)\,dx/80 + \int_{60}^{80} x\,dx/80 + 20 = 15 + 17.5 + 20 = 52.5$. The ground-up means are $\Pr(X > 20) = 0.8$ times the layer means: 32, 46, and 42.

```
hand = {'Ordinary': 40, 'Franchise': 57.5, 'Disappearing': 52.5}
df = pd.DataFrame.from_dict(
    {k: [hand[k], a.sevs[0].moms()[0], a.est_sev_m,
         ground_up[k].sevs[0].moms()[0], a.est_m] for k, a in books.items()},
    orient='index',
    columns=['hand', 'layer mean', 'grid mean', 'ground up', 'aggregate'])
df.index.name = 'deductible'
GT(df)
```

deductible	hand	layer mean	grid mean	ground up	aggregate
Ordinary	40.0000	40.0000	40.0000	32.0000	400.0000
Franchise	57.5000	57.5000	57.5000	46.0000	575.0000
Disappearing	52.5000	52.5000	52.5000	42.0000	525.0000

The layer mean is exact, from the same partial expectations as an ordinary layer; the grid mean is that of the discretized severity, on a bucket of 1/16. The aggregate mean is ten times the layer mean.

A vector of layers with disappearing deductibles

In a vector of layers, `to [...]` takes one retention end per layer, and a scalar `to d2` applies to every layer. The first layer below, `40 xs 10 to 30 disappearing`, pays $1.5(x-10)$ on $(10, 30]$ and then $\min(x, 40)$, with mean 33.89 given $X > 10$.

```
v = build('agg Vector 10 claims [40 80] xs [10 20] to [30 60] disappearing '
          'sev 100 * uniform poisson')
for s in v.sevs:
    print(s.tail_description, f'mean {s.moms()[0]:.4f}')
```

```
uniform, [0, 40], bounded; 40 xs 10 to 30 disappearing mean 33.8889
uniform, [0, 80], bounded; 80 xs 20 to 60 disappearing mean 52.5000
```

Refused programs

For a disappearing deductible the retention end must be at least the attachment, and a vector `to` must have one entry per layer. A franchise takes no `to`, since its retention end is the attachment, so `to 60 franchise` is a parse error.

```
refuse(*[p + ' sev 100 * uniform poisson' for p in [
    'agg D 10 claims 80 xs 20 to 10 disappearing',
    'agg D 10 claims [40 80] xs [10 20] to [30 60 90] disappearing',
    'agg D 10 claims 80 xs 20 to 60 franchise']])
```

```
Retention end 10 must be at least the attachment 20.
disappearing deductible: to [...] has 3 entries but there are 2 layers; give one
retention end per layer, or a single value for all of them.
DecL parse error at line 1, column 32: Unexpected 'franchise'.
```