Counting experiments

EXPECTED YIELD IN CHANNEL J

$$\lambda_j = T_j\left(b\rho_j + \varepsilon_j S\right)$$

COMPLETE EXAMPLE

Two channels, one background scale

```
%%%
Two counting channels with a shared nuisance.
%%%
flatppl_compat = "0.1"
channels = table(
    exposure = [1.0, 2.0],
    efficiency = [0.8, 0.7],
    background_rate = [3.0, 4.0],
    counts = [5, 10])

# Signal production and detector nuisance.
S ~ Gamma(shape = 2, rate = 1)
b_scale ~ LogNormal(0, 0.2)
background = b_scale .* channels.background_rate
signal = channels.efficiency .* S
expected = channels.exposure .* (
    background .+ signal)
counts ~ Poisson.(expected)

# One nuisance is shared by both channels.
prior = lawof(record(S = S, b_scale = b_scale))
K = kernelof(record(counts = counts),
    S, b_scale)
L = likelihoodof(K,
    record(counts = channels.counts))
posterior = bayesupdate(L, prior)

# Independent check point: [4.6, 10.8].
point = record(S = 2.0, b_scale = 1.0)
log_L = logdensityof(L, point)
expected_fn = functionof(expected, S, b_scale)
expected_at_point = expected_fn(point)
```

Independent channel uncertainties

```
b_scales ~ iid(LogNormal(0, 0.2), 2)
```

For independent background scales draw a two-element vector. Replace the scalar in the prediction, prior, kernel inputs and parameter record.

A histogram with two templates

```
means = 4 .* [0.3, 0.7] .+ 11 .* [0.4, 0.6]
bins ~ Poisson.(means)
```

Each template sums to one. The signal yield 4 and background yield 11 give bin means 5.6 and 9.4.

ANNOTATED LINES

A Channel data as a table 5–9

```
channels = table(
    exposure = [1.0, 2.0],
    efficiency = [0.8, 0.7],
    background_rate = [3.0, 4.0],
    counts = [5, 10])
```

Each row describes one channel. `channels.exposure` returns its exposure column. The expected signal is production times exposure times efficiency.

B Broadcast the yield equation 14–18

```
background = b_scale .* channels.background_rate
signal = channels.efficiency .* S
expected = channels.exposure .* (
    background .+ signal)
counts ~ Poisson.(expected)
```

Both background rates share `b_scale`. Efficiency acts on the produced signal `S`. Given these parameters, `Poisson.` draws the two channel counts independently.

C Observed counts and parameter names 21–26

```
prior = lawof(record(S = S, b_scale = b_scale))
K = kernelof(record(counts = counts),
    S, b_scale)
L = likelihoodof(K,
    record(counts = channels.counts))
posterior = bayesupdate(L, prior)
```

Name both `S` and `b_scale` in the prior and the kernel inputs. The observation record needs one field, `counts`, containing the vector of channel counts.

D The two-channel log likelihood 29–32

```
point = record(S = 2.0, b_scale = 1.0)
log_L = logdensityof(L, point)
expected_fn = functionof(expected, S, b_scale)
expected_at_point = expected_fn(point)
```

Evaluate the expected counts with `functionof` at the parameter point used by `logdensityof`. The sum of the two Poisson log masses gives `log_L` ≈ -3.86616.

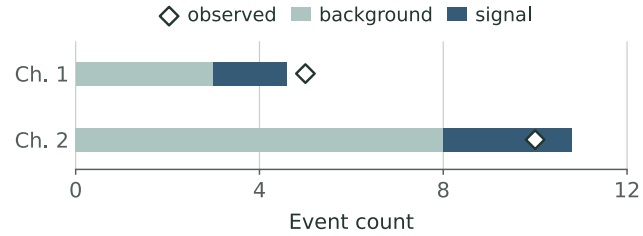
A calibration constraint

```
cal(S, b_scale) = Normal(b_scale, 0.1)
L_cal = likelihoodof(cal, 1.05)
L_all = joint_likelihood(L, L_cal)
```

Only `b_scale` enters the calibration. The combined log likelihood at `point` is −2.60751.

COUNTS AND SPECTRA

Expected and observed counts



For `S = 2` and `b_scale = 1` the means are 4.6 and 10.8. The diamonds mark the synthetic counts supplied in `channels.counts`.

Define the observation window

```
window = interval(0, 10)
peak = normalize(truncate(
    Normal(5, 0.2), window))
continuum = Uniform(window)
```

Normalize the shapes on the same window when their coefficients give the expected yields there.

Unbinned event likelihood

```
I = superpose(weighted(4, peak),
    weighted(11, continuum))
events ~ PoissonProcess(I)
```

The total intensity gives the expected count. The event array specifies both the energies and their number. Its likelihood therefore contains the count term.

Condition on the event count

```
shape = normalize(I)
energy ~ iid(shape, 15)
```

Fixing the count at 15 leaves the energy shape. For counts in two energy bins, use `BinnedPoissonProcess([0,5,10], I)`.

Fixed trial totals

```
passed ~ Binomial(20, channels.efficiency[1])
```

Use a Binomial law when the trial total is known. A Poisson yield describes a random count over a fixed exposure.

Jessica Cox · bcox@mpp.mpg.de