FlatPPL 0.1 / BEGINNER

Parameters and likelihoods

CONDITIONAL MEAN

$$\mu_i = \alpha + \beta x_i$$

COMPLETE EXAMPLE

Linear regression

```
%%%
Linear regression with an optional prior.
%%%
flatppl_compat = "0.1"
x_data = [0.0, 1.0, 2.0, 3.0]
y_data = [0.8, 2.1, 2.9, 4.2]

# Parameter domains.
alpha = elementof(reals)
beta = elementof(reals)
sigma = elementof(posreals)

# One mean for each observation.
mu = alpha .+ beta .* x_data
y ~ Normal.(mu, sigma)
K = kernelof(record(y = y))
L = likelihoodof(K, record(y = y_data))

# Optional prior on the named parameters.
a ~ Normal(0, 5)
b ~ Normal(0, 2)
s ~ LogNormal(0, 0.5)
prior = lawof(record(alpha = a,
    beta = b, sigma = s))
posterior = bayesupdate(L, prior)

point = record(alpha = 1.0,
    beta = 1.0, sigma = 0.5)
log_L = logdensityof(L, point)

f_mean = functionof(mu, alpha, beta)
fitted = f_mean(alpha = 1.0, beta = 1.0)
```

Starting from prior draws

```
alpha ~ Normal(0, 5)
beta ~ Normal(0, 2)
sigma ~ LogNormal(0, 0.5)
prior = lawof(record(alpha = alpha,
    beta = beta, sigma = sigma))
K = kernelof(record(y = y), alpha, beta, sigma)
```

Use these draws on lines 9–11. Keep the mean and response on lines 14–15. Use this `prior` and `K` with the original `L` and `posterior`.

Bounds on the noise scale

```
domain = cartprod(alpha = reals, beta = reals,
    sigma = interval(0.1, 2))
```

This set contains parameter records with $0.1 \leq \sigma \leq 2$.

ANNOTATED LINES

A Declare the parameter domains 9–11

```
alpha = elementof(reals)
beta = elementof(reals)
sigma = elementof(posreals)
```

Declare a free parameter on `S` with `elementof(S)`. The regression permits real coefficients and requires a positive scale. Each can vary before a prior is specified.

B From a mean to a likelihood 14–17

```
mu = alpha .+ beta .* x_data
y ~ Normal.(mu, sigma)
K = kernelof(record(y = y))
L = likelihoodof(K, record(y = y_data))
```

The dotted arithmetic gives one mean per reading. `Normal.(mu, sigma)` pairs each mean with the shared noise scale. `kernelof` infers the three free inputs. Bind the four readings to its `y` field.

C A prior on the named parameters 20–25

```
a ~ Normal(0, 5)
b ~ Normal(0, 2)
s ~ LogNormal(0, 0.5)
prior = lawof(record(alpha = a,
    beta = b, sigma = s))
posterior = bayesupdate(L, prior)
```

Give the prior draws `a`, `b` and `s` the field names used by the likelihood. `lawof` returns their joint law. `bayesupdate` weights it by `L` to give an unnormalized posterior.

D Log likelihood and fitted means 27–32

```
point = record(alpha = 1.0,
    beta = 1.0, sigma = 0.5)
log_L = logdensityof(L, point)

f_mean = functionof(mu, alpha, beta)
fitted = f_mean(alpha = 1.0, beta = 1.0)
```

Pass all three parameters to `logdensityof` in one record. The mean depends only on the two coefficients. `functionof` therefore takes two inputs and returns the four fitted means at the original predictors.

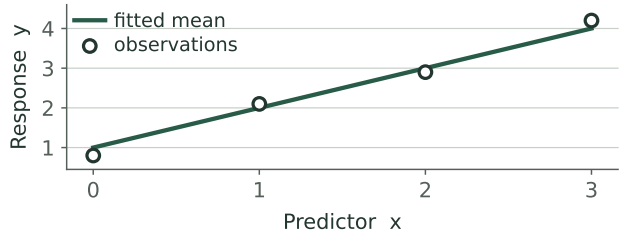
Predict at a new input

```
mean_at(x, a, b) = a + b*x
pred = mean_at(5, point.alpha, point.beta)
```

`f_mean` uses the four fixed `x_data` values. This helper also takes `x` as an argument and gives a mean of 6 at `x = 5`.

PREDICTIONS AND VARIANTS

Prediction at a named point



At `point`, `fitted` gives `[1, 2, 3, 4]`. The four residuals give `log_L` ≈ -1.10317 for $\sigma = 0.5$.

Change the observation family

```
rate = exp.(mu)
y_count ~ Poisson.(rate)
p = invlogit.(mu)
y_binary ~ Bernoulli.(p)
```

For counts, exponentiate the linear predictor to get a positive mean. For a binary response, `invlogit` converts the predictor to a success probability.

Common and observation-specific means

```
ys ~ iid(Normal(alpha, sigma), 4)
zs ~ Normal.(mu, sigma)
```

All entries of `ys` have the same mean. Each entry of `zs` uses its corresponding mean in `mu`. In each case, the observations are conditionally independent.

Field names and indices

```
r = record(a = 1, b = 2)
u = r.a
v = [1, 2, 3]
w = v[1]
```

Record fields have names. Array indices start at one. Each binding has a single definition within a module.

Supply fixed inputs

```
n = external(posintegers)
```

Supply `n` when loading the module. The suffix `_data` denotes a naming convention. Bind the observations with `likelihoodof`.

Use `#` for a comment, `%` for documentation on a binding, and paired `%%%` lines for a documentation block.

Jessica Cox · bcox@mpp.mpg.de