PRACTICAL

0 Generalized Linear Model

In this practical we will:

- Simulate non-Gaussian data
- Learn how to fit a generalised linear model with `inlabru`
- Generate predictions from the model

A generalised linear model allows for the data likelihood to be non-Gaussian. In this example we have a discrete response variable which we model using a Poisson distribution. Thus, we assume that our data

$$y_i \sim \text{Poisson}(\lambda_i)$$

with rate parameter λ_i which, using a log link, has associated predictor

$$\eta_i = \log \lambda_i = u_0 + u_1 x_i$$

with parameters u_0 and u_1, and covariate x. This is identical in form to the predictor in **?@sec-linmodel**. The only difference is now we must specify a different data likelihood.

0.1.1 Simulate example data

This code generates 100 samples of covariate `x` and data `y`.

```
set.seed(123)
n = 100
beta = c(1,1)
x = rnorm(n)
lambda = exp(beta[1] + beta[2] * x)
y = rpois(n, lambda  = lambda)
df = data.frame(y = y, x = x)
```

0.1.2 Define model components and likelihood

Since the predictor is the same as **?@sec-linmodel**, we can use the same component definition:

```
cmp =  ~ Intercept(1) + x_effect(x, model = "linear")
```

However, when building the observation model likelihood we must now specify the Poisson likelihood using the `family` argument (the default link function for this family is the log link).

```
lik =  bru_obs(formula = y ~.,
               family = "poisson",
               data = df)
```

0.1.3 Fit the model

Once the likelihood object is constructed, fitting the model is exactly the same process as in **?@sec-linmodel**.

```
fit_glm = bru(cmp, lik)
```

And model summaries can be viewed using

```
summary(fit_glm)
```

```
inlabru version: 2.12.0
INLA version: 25.02.10
Components:
Intercept: main = linear(1), group = exchangeable(1L), replicate = iid(1L), NULL
x_effect: main = linear(x), group = exchangeable(1L), replicate = iid(1L), NULL
Likelihoods:
  Family: 'poisson'
    Tag: ''
    Data class: 'data.frame'
    Response class: 'integer'
    Predictor: y ~ .
    Used components: effects[Intercept, x_effect], latent[]
Time used:
    Pre = 0.437, Running = 0.167, Post = 0.0383, Total = 0.642
Fixed effects:
           mean    sd 0.025quant 0.5quant 0.975quant  mode kld
Intercept 0.915 0.071      0.775    0.915      1.054 0.915   0
x_effect  1.048 0.056      0.938    1.048      1.157 1.048   0

Deviance Information Criterion (DIC) ...............: 386.39
Deviance Information Criterion (DIC, saturated) ....: 120.67
Effective number of parameters .....................: 2.00

Watanabe-Akaike information criterion (WAIC) ...: 387.33
Effective number of parameters .................: 2.73

Marginal log-Likelihood:  -204.02
 is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

0.1.3.1 Generate model predictions

To generate new predictions we must provide a data frame that contains the covariate values for x at which we want to predict.

This code block generates predictions for the data we used to fit the model (contained in `df$x`) as well as 10 new covariate values sampled from a uniform distribution `runif(10)`.

```
# Define new data, set to NA the values for prediction

new_data = data.frame(x = c(df$x, runif(10)),
```

```
                    y = c(df$y, rep(NA,10)))

# Define predictor formula
pred_fml <- ~ exp(Intercept + x_effect)

# Generate predictions
pred_glm <- predict(fit_glm, new_data, pred_fml)
```

Since we used a log link (which is the default for `family = "poisson"`), we want to predict the exponential of the predictor. We specify this using a general `R` expression using the formula syntax.

> **Note**
>
> Note that the predict function will call the component names (i.e. the "labels") that were decided when defining the model.

Since the component definition is looking for a covariate named x, all we need to provide is a data frame that contains one, and the software does the rest.

0 Plot

```
Warning: Removed 10 rows containing missing values or values outside the scale range
(`geom_point()`).
```

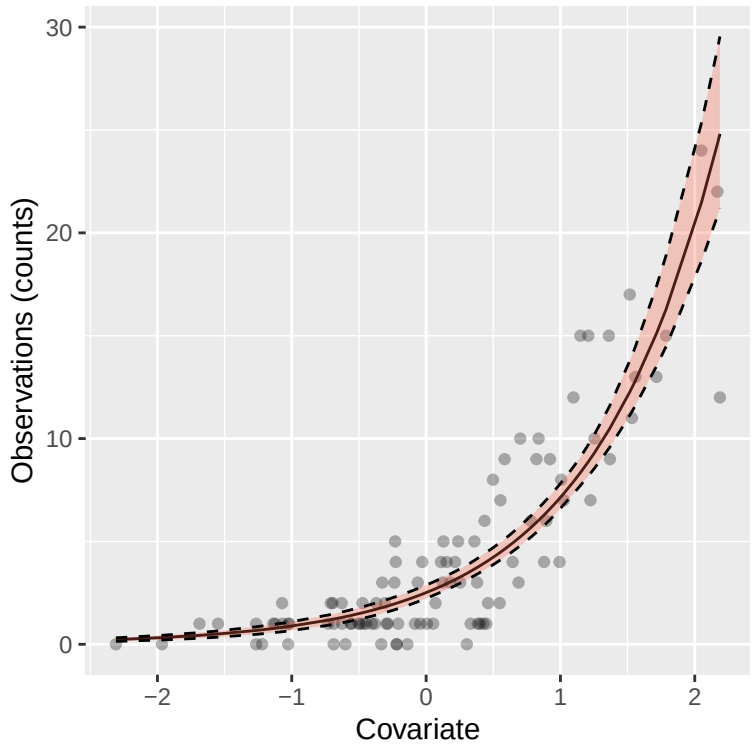


Figure 1: Data and 95% credible intervals

0 R Code

```
pred_glm %>% ggplot() +
  geom_point(aes(x,y), alpha = 0.3) +
  geom_line(aes(x,mean)) +
    geom_ribbon(aes(x = x, ymax = q0.975, ymin = q0.025),fill = "tomato", alpha = 0.3)+
  xlab("Covariate") + ylab("Observations (counts)")
```

::: {.callout-warning icon="false"} ## Task

Suppose a binary response such that

$$y_i \sim \text{Bernoulli}(\psi_i)$$
$$\eta_i = \text{logit}(\psi_i) = \alpha_0 + \alpha_1 \times w_i$$

Using the following simulated data, use `inlabru`to fit the logistic regression above. Then, plot the predictions for 10 new observations.

```
set.seed(123)
n = 100
alpha = c(0.5,1.5)
w = rnorm(n)
psi = plogis(alpha[1] + alpha[2] * w)
y = rbinom(n = n, size = 1, prob =  psi) # set size = 1 to draw binary observations
df_logis = data.frame(y = y, w = w)
```

Here we use the logit link function $\text{logit}(x) = \log\left(\frac{x}{1-x}\right)$ (`plogis()` function in R) to link the linear predictor to the probabilities ψ.

Take hint

You can set `family = "binomial"` for binary responses and the $plogis()$ function for computing the predicts probabilities at new observations.

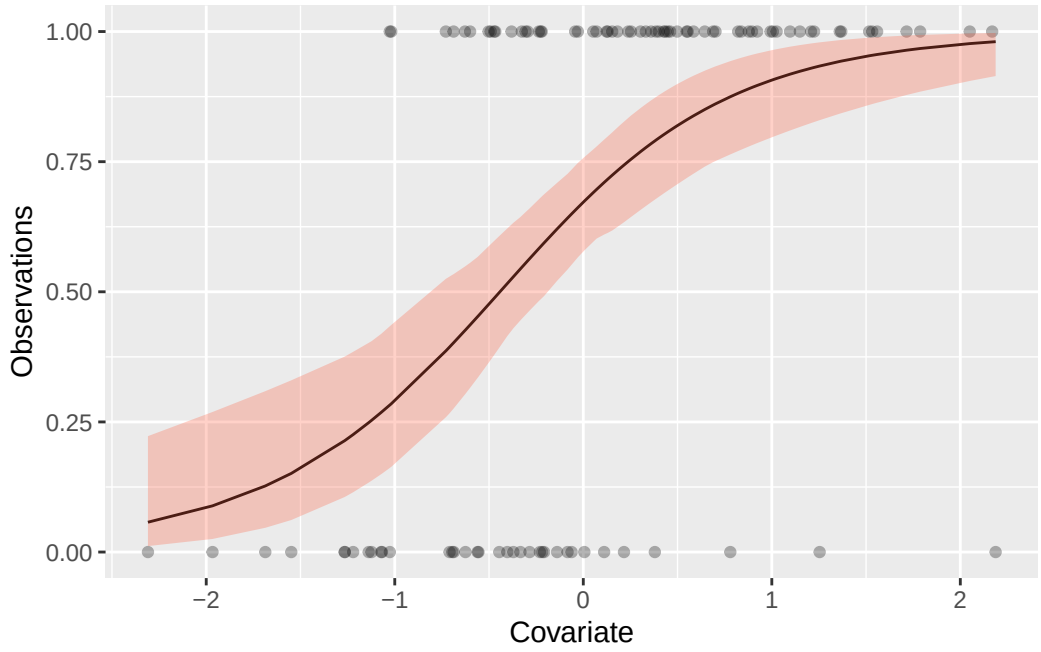
> **Note**
>
> The default distribution is $\text{Binomial}(1, \psi)$. However, if you have proportional data (e.g. $\frac{\text{no. successes}}{\text{no. of trials}}$) you can specify the number of events as your response and then the number of trials via the $Ntrials = n$ option in the `bru_obs`function.

Click here to see the solution

```
# Model components
cmp_logis =  ~ Intercept(1) + w_effect(w, model = "linear")
# Model likelihood
lik_logis =  bru_obs(formula = y ~.,
             family = "binomial",
             data = df_logis)
# fit the model
fit_logis <- bru(cmp_logis,lik_logis)

# Define data for prediction
new_data = data.frame(w = c(df_logis$w, runif(10)),
```

```
                      y = c(df_logis$y, rep(NA,10)))
# Define predictor formula
pred_fml <- ~ plogis(Intercept + w_effect)

# Generate predictions
pred_logis <- predict(fit_logis, new_data, pred_fml)

# Plot predictions
pred_logis %>% ggplot() +
  geom_point(aes(w,y), alpha = 0.3) +
  geom_line(aes(w,mean)) +
    geom_ribbon(aes(x = w, ymax = q0.975, ymin = q0.025),fill = "tomato", alpha = 0.3)+
  xlab("Covariate") + ylab("Observations")
```

```
Warning: Removed 10 rows containing missing values or values outside the scale range
(`geom_point()`).
```



:::