Models

Ülo Maiväli

Bayesian model – an unthinking machine that generates stochastic counterfactual data

- data generating process → data → data generating model
- by understanding the model we hope to understand the process

- models are tools – and tools do not have truth values.
- stochastic models generate data stochastically to model randomness in the data generating process & and to model our uncertainty about this process

2 kinds of the unknown

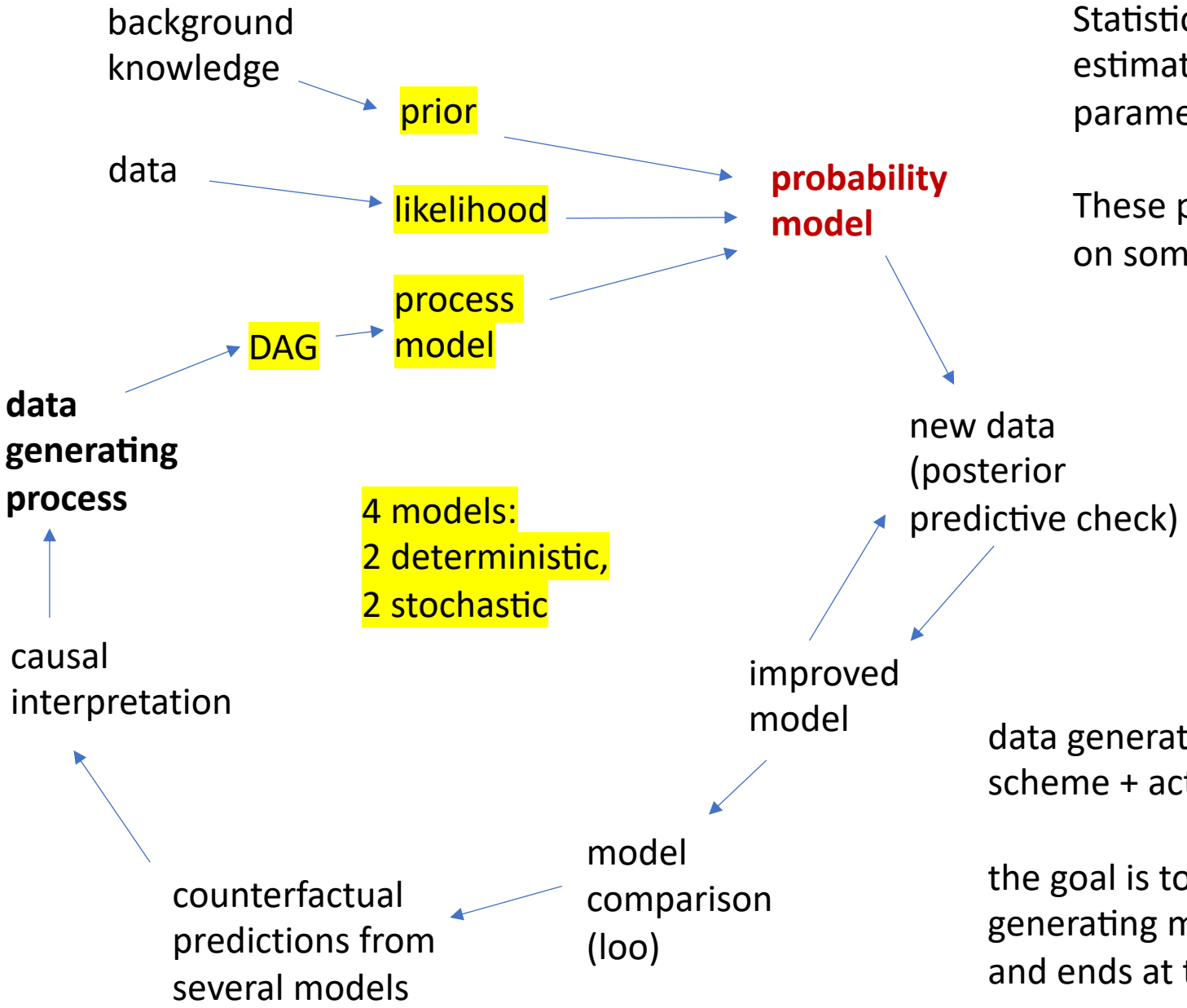
- aleatory – a random process
- epistemic – uncertain knowledge

- in the casino a single game is completely random, but in the long run there is no uncertainty about probabilities. In the long run the predictions are exact.
- in more life-like situations we model biological variation and uncertainty together in the same model.

regression: predict mean y values from x values

$$y_i = a + bx_i + error \quad for\ i = 1, \ldots, n$$

- ***stochastic error model*** (information about y, that is not in x, goes into error)
- ***deterministic process model***
- y – predicted variable (dependent var).
- x – predictor variable (independent var), regressor
- i – index or row of the table (denotes i-th measurement)
- a – intercept
- b – slope
- a, b – parameters. Fitted parameters are coefficients
- fitting the model: fixing parameter values on data (we condition the model on data).



Statistical inference is a set of operations on data that yield estimates and uncertainty statements about predictions and parameters of some underlying process or population.

These probabilistic uncertainty statements are derived based on some assumed probability model for observed data.

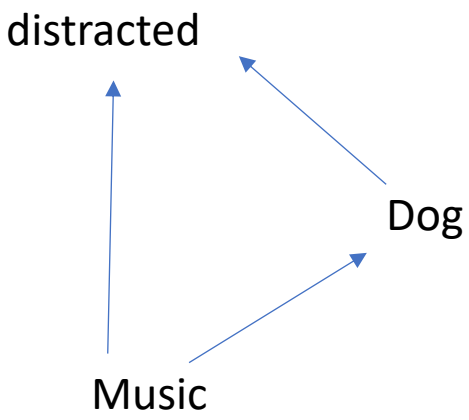
data generating process: scientific hypothesis + data collection scheme + actual process of data collection/measurement

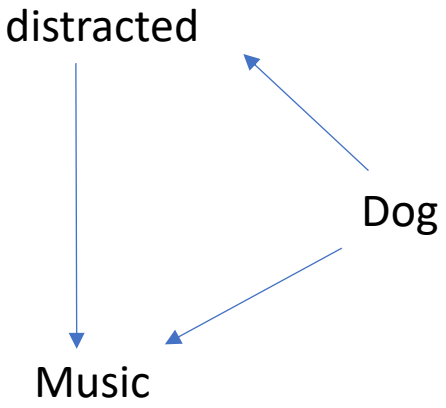
the goal is to infer stuff about the scientific part of the data generating mechanism based on data & model. Everything starts and ends at the data generating process.

Scientific hypotheses are causal, thus modelling has to be causal, but causal knowledge is not in the data

science moves through successive simplifications

- the world is complicated
- scientific hypothesis is a reductive simplification (isolates a part of the world and offers a non-mathematical explanation)
- DAG is a non-quantitative simplification of the causal part of the hypothesis
- process model translates implications of the DAG into the model structure. This translation is an imperfect model itself.
- prior translates non-mathematical knowledge about possible parameter values into a probability distribution
- likelihood translates compatibility of data with the possible parameter values into a probability distribution





Are you distracted from doing your homework by your dog?

what is the right regression equation (the process model) for this question?

*distracted = a + b1 * dog + error*

or

*distracted = a + b1 * dog + b2 * music + error*

or

*distracted = a + b1 * dog + b2 * music + b3 * dog * music + error*

or something else?

simple probability model

y ~ N(mu, sigma) –likelihood, or y data model

mu ~ N(.,.) - prior for the mean value of y

sigma ~exp(.) - prior for the data level variation (SD)

- mu and sigma are parameters, whose values will be estimated on data.
- the model is generative – we can generate new data from the fitted model.

the full regression model

- y_i ~ N(mu, sigma) - likelihood
- mu = a + bx_i - process model
- a ~ N(.,.) - prior for the intercept
- b ~ N(.,.) - prior for the slope
- sigma ~ exp(.) - prior for data level standard deviation

- y ~ Binomial(n, p)
- p = a + bx_i
- a ~ N(.,.)
- b ~ N(.,.)

continuous predicted var (y)

Cholesterol measurement	Treatment (T=0 placebo)
18.3	1

1. we divide data into 4 tables (one for each sex and treatment level): $y_{S0\_T1} \sim N(mu, sigma)$ etc. then we get effect size as $posterior_{S0\_T1} - posterior_{S0\_T0}$ and so on.

2. we run the model

continuous y – linear model

Normal likelihood

Y ~ N(mu, sigma)

mu = a + bx

a ~ N()

b ~ N()

sigma ~ t()

students t likelihood

y ~ t(nu, mu, sigma)

mu = a + bx

a ~ N()

b ~ N()

sigma ~ t()

nu ~ N()

model mean and variation conditional on x

$Y \sim N(mu, sigma)$

$mu = a_1 + b_1 x$

$sigma = a_{sd} + b_{sd} x$

$a_1 \sim N()$

$b_2 \sim N()$

$a_{sd} \sim N()$

$b_{sd} \sim N()$

brm(bf(y~x, sigma ~ x))

Adding predictors

We add additional predictors when we believe (i) that all predictors ($x_1, ..., x_i$) influence the predicted variable (y), and (ii) that those influences are independent of each other. Independence leads to an additive process model.

$$y \sim normal(\mu, \sigma)$$

$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i$$

Now the slope of x_1 tells us, how much mean y changes, if we change x_1 by 1 unit and x_2 by 0 units (keep x_2 constant). Equivalently the slope of x_2 tells us, how much mean y changes, if we change x_2 by 1 unit and keep x_1 constant.

There is another way of interpreting multiple regression betas: β_1 tells us, what extra information on predicting the y value we get from x_1, if we already know how x_2 influences y. And conversely, what extra value in predicting y do we get from x_2, if we already know how x_1 influences y. For instance, we can predict someones height from the length of his right foot, but if we already know the prediction from the left foot, the right foot gives us (almost) nothing. And since including it in the model leaves very little covariation with height that is not also covariation of the left foot with height (and vice versa), then most of the predictive information that is included in the R^2, is not included in the respective betas (which consequently have extremely wide CI-s). This is collinearity.

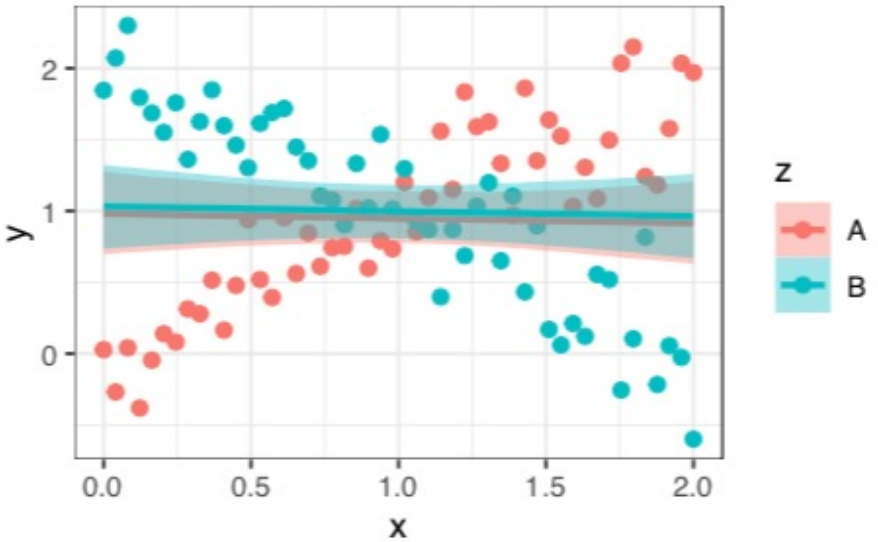


Figure 20: An additive model y = a + b1 * x + b2 * z, where z is a two-level categorical variable leads to fixed parallel slopes.

The slopes of the A and B experiments are restricted to be parallel (fixed) by the additive model structure. The regression lines cross the *mean*($y|x = A$) and *mean*($y|x = B$) points, which in this simulation is the same value (1).

- there is no reason to think that if we compare models $\mu = \alpha + \beta x_1$ and $\mu = \alpha + \beta_1 x_1 + \beta_2 x_2$, then β and β_1 would be similar, or even have the same sign. The first is the total effect of x_1 on y, and the second is, in causal interpretation, the direct effect of x_1 on y.

- if you have several predictors in your regression, then for easier comparison of the betas, it makes sense to standardize them. Then the predictors that have betas farther from zero, have more effect on y.

- If you have only continuous predictors, use this `x_st = (x - mean(x))/sd(x)`. Then all your predictors have mean = 0 and sd = 1. Then the intercept is the value of y, when all predictors are at their mean values (which is always 0), and each slope gives the change in mean y, if the relevant predictor changes by one standard deviation, and all other predictors remain unchanged.

- If you have both continuous and binary predictors, then you will get approximate comparability of betas with the following transformation: `x_st2 = (x - mean(x))/(2*sd(x))`. Now the meaning of the slope changes - it is the change in mean y if x changes by 2 standard deviations (that is from the 2.5th quantile to the 97.5th quantile, which is a change that covers almost the range of x variation). As before, all other X-s are kept constant, and intercept is connected to mean values of all X-s.

interaction model

the likelihood is the usual y ~N(mu, sigma)

the process model:

$$mu = \alpha + \beta_1 x + \beta_2 z + \beta_3 xz$$

or

$$mu = \alpha + \gamma x + \beta_2 z$$

$$\gamma = \beta_1 + \beta_3 z$$

or

$$mu = \alpha + \gamma z + \beta_1 x$$

$$\gamma = \beta_2 + \beta_3 x$$

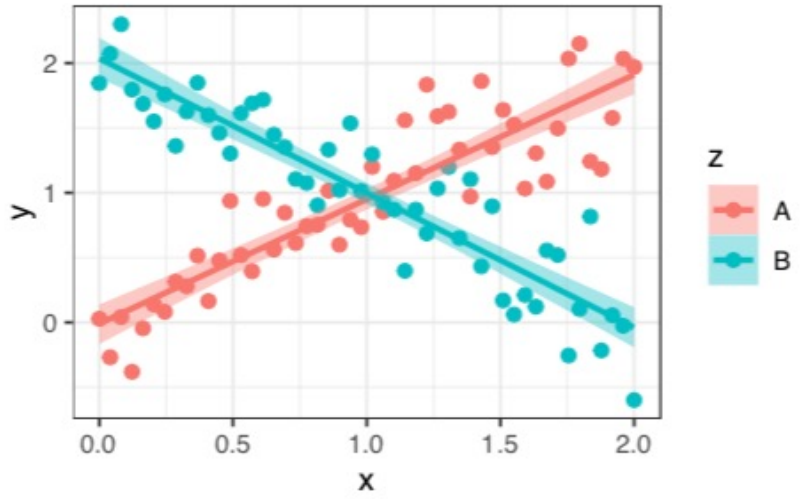


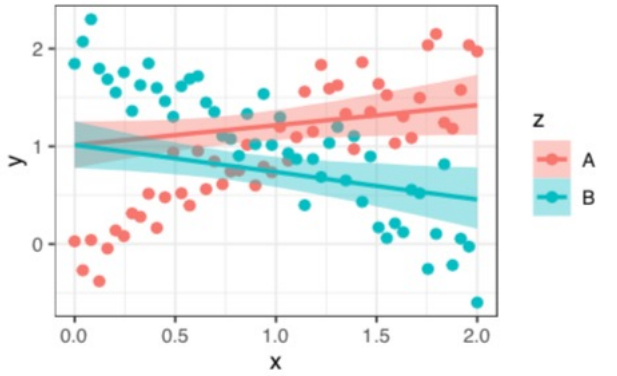
Figure 21: An interaction model y = a + b1 * x + b2 * z + b3 * x * z leads to free slopes.

```
##                Estimate
## b_Intercept -0.0110400
## b_x          0.9610941
## b_zB         2.0473411
## b_x:zB      -1.9991338
## sigma        0.2908296
```

- if z is a continuous variable the interaction model gives a separate x slope for every z value, and conversely, it gives a seprate z slope for each x value.
- This means that the model expects that the relationship between x and y, e.g. how much a unit increase in x predicts the mean y value at this x value, to depend on the value of z.
  - And vice versa, it symmetrically expects that the relationship between z and y, e.g. how much a unit increase in z predicts the mean y value at this z value, to depend on the value of x.
- The model neither knows nor cares, which variable, x or z, is of main interest to you, or which (if any) of them causally affects y.
- But it does know (assume) that the variability of x is identical at all levels of z, and vice versa.

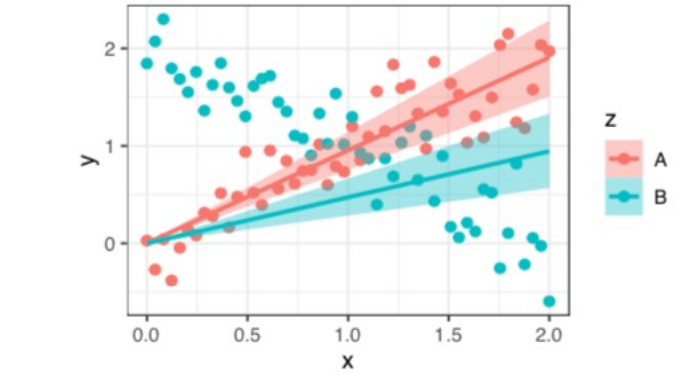
- when we omit the additive part of the model: $y = \alpha + \beta xz$

  Now the intercept is at the mean values of $x|z = A$ and $x|z = B$, which in both cases is 1. The slopes are 0.2 and -0.3 for A and B.



- an even simpler model $y = \beta xz$, where intercept = 0.

  Here the slopes are 1 and 0.5, for A and B, respectively.



More generally, for a sample space with n members (n mutually exclusive and exhaustive hypotheses) $\Omega = \{H_1, H_2, ...H_n\}$ and

$$P(d) = P(H_1)P(d|H_1) + P(H_2)P(d|H_2) + ... + P(H_n)P(d|H_n)$$

This is the formula for total, a.k.a. marginal, probability.
So the Bayes theorem becomes

$$P(H_1|d) = \frac{P(H_1)P(d|H_1)}{P(H_1)P(d|H_1) + P(H_2)P(d|H_2) + ... + P(H_n)P(d|H_n)}$$

- the formula that allows, in the logically best way, to combine evidence that is present in data (as modelled in the likelihood) with data-independent beliefs (as modeled in the prior), and end up with a posterior probability of hypothesis.
  - In other words, we can use the Bayes theorem to combine disparate strands of information into a single posterior probability distribution.
- The Bayes theorem is not a truth machine – it merely combines information that we put into it.
- Bayes theorem works iteratively and there is no lower limit of data that you can put into it: N=1 is fine.
  - it is easy to incorporate new data, as you can simply redefine your posterior from the previous iteration as the new prior, add new data as likelihood, and calculate the new posterior, which before long will become a prior.
  - Bayesian inference recapitulates the piecemeal nature of scientific inference.
  - Unlike in frequentist statistics, there are no stopping rules, meaning that you can safely calculate the posterior after receiving each new datum.

A simple example of Bayesian inference

Mortality of a disease is 50% and we have 3 patients. How many of them are likely to die? We have 2 pieces of data (mortality rate p = 0.5 and N = 3) and a 4-member sample space (0 dead, 1 dead, 2 dead, and 3 dead). We start by enumerating all logically possible scenarios (d - dead, a - alive)

d d d
d a d
d d a
a d a
a a d
a a a
a d d
d a a

Kui P(dead) = 0.5, then by simple counting we get that:

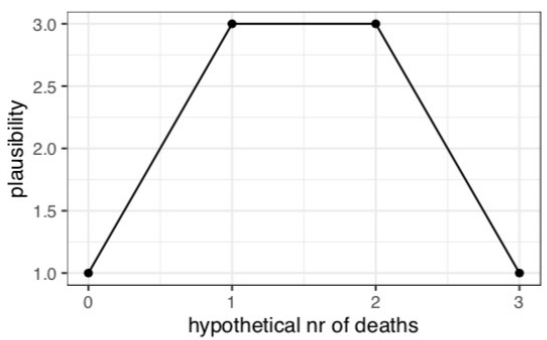
- 0 dead - 1,
- 1 dead - 3,
- 2 dead - 3,
- 3 dead - 1

Thus we have 1 + 3 + 3 + 1 = 8 possible futures that divide between the 4 hypotheses. This means that for each member of the parameter space we know the probability of its realization (P(1 death) = 3/8, etc.). It is this knowledge we now turn into a likelihood function.

```r
# Parameter space: all possible futures
x <- seq(from = 0, to = 3)
```



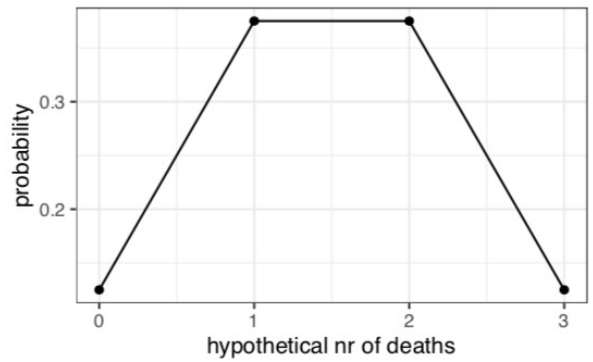
```r
# Likelihoods for each x value, or P(deaths I x)
y <- c(1, 3, 3, 1)

ggplot(data = NULL, aes(x, y)) +
geom_point() + geom_line() +
xlab("hypothetical nr of deaths") +
ylab("plausibility") + theme_bw()
```



One death and two deaths are equilikely and one death is three times as likely as no deaths (or three deaths). Likelihood simply tells for each number of deaths, how likely is this outcome, given the mortality figure that we state.

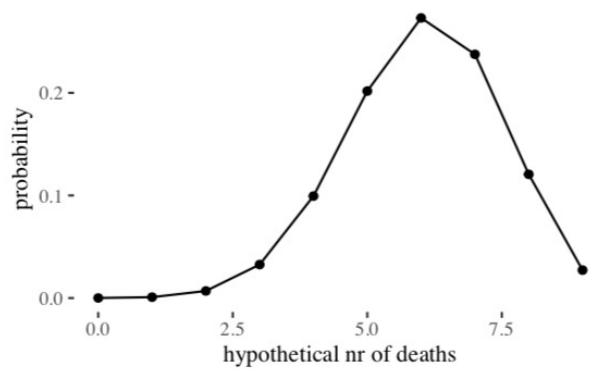
Now we demonstrate the same using the binomial distribution model. The only difference is that now we get on the y-axis normalized probabilities.

```
y <- dbinom(x, 3, 0.5)

ggplot(data = NULL, aes(x, y)) +
geom_point() + geom_line() +
xlab("hypothetical nr of deaths") +
ylab("probability") + theme_bw()
```



How many dead when we have 9 patients and mortality rate is 67 percent?

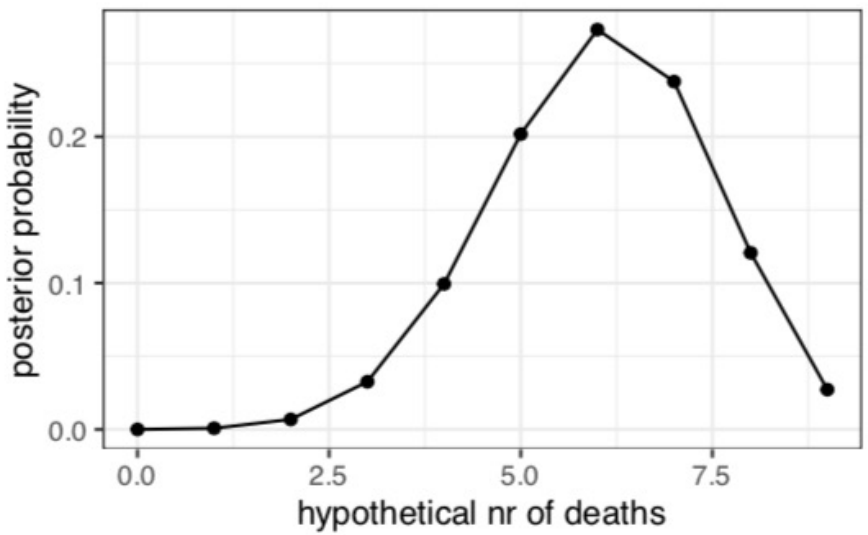
```
x <- seq(from = 0, to = 9)
y <- dbinom(x, 9, 0.67)

ggplot(data = NULL, aes(x, y)) +
geom_point() +
geom_line() +
xlab("hypothetical nr of deaths") +
ylab("probability") +
ggthemes::theme_tufte()
```



Next we add to the likelihood a flat prior and use the Bayes theorem.

Next we add to the likelihood a flat prior and use the Bayes theorem.

```
x <- seq(from = 0, to = 9)
#parameter space: nr of deaths
prior <- rep(1, 10)
# flat prior
likelihood <- dbinom(x, size = 9, prob = 0.67)
# Compute likelihood at each value in grid
unstd.posterior <- likelihood * prior
# Compute product of likelihood and prior
posterior <- unstd.posterior/sum(unstd.posterior)
# Normalize the posterior, so that it sums to 1
```




```
ggplot(data = NULL, aes(x, posterior)) +
geom_point() + geom_line() +
xlab("hypothetical nr of deaths") +
ylab("posterior probability") + theme_bw()
```

a shift of perspective: estimate the true mortality.

- data: 6 dead out of 9 patients.
- parameter to estimate: the mortality rate (p).
- sample space Ω: real numbers between 0 and 1.

<u>dbinom() has 3 arguments:</u>

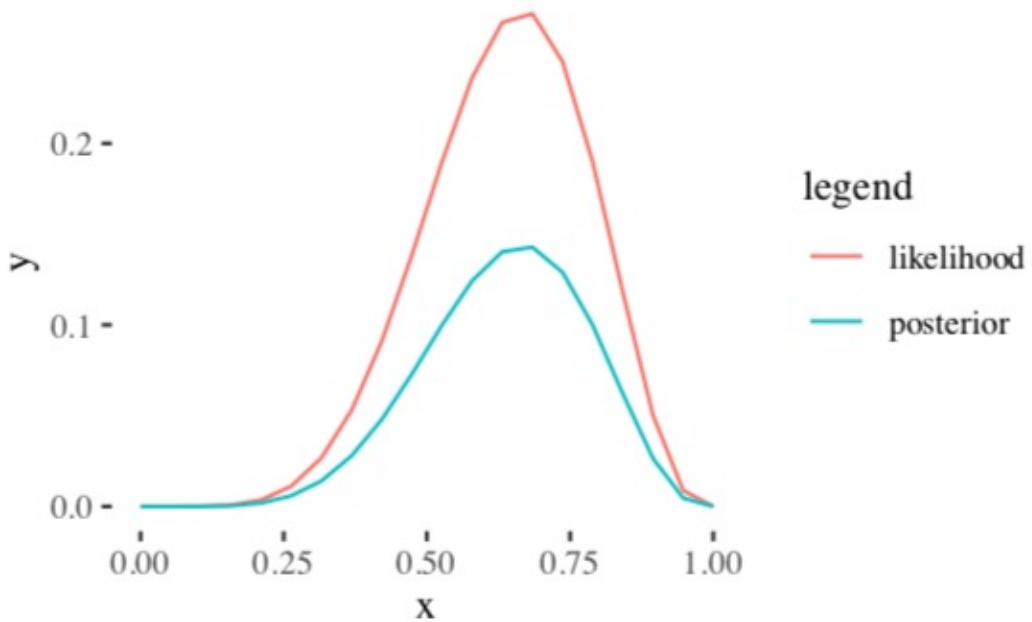
1. nr of events,
2. nr of tries and
3. probability of events.

Again, we have 2 of them as data and 1 as the parameter to estimate.

We use the flat prior.

As there are Inf number of members in Ω, we cheat a little and calculate for the grid of 20 evenly spaced parameter values.

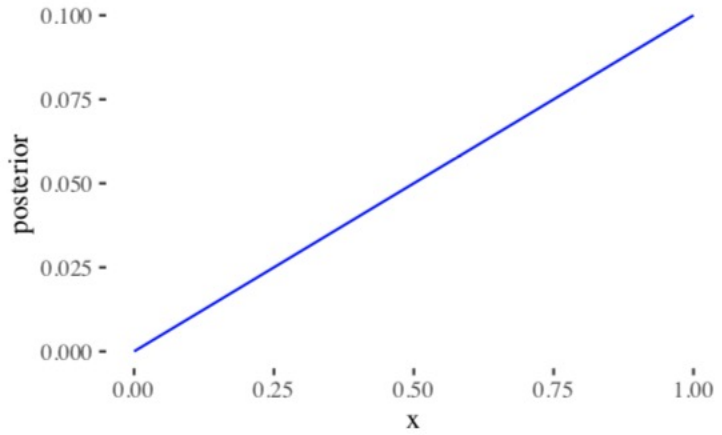
```r
# grid: mortality at 20 evenly spaced probabilities from 0 to 1
x <- seq(from = 0 , to = 1, length.out = 20)

# prior
prior <- rep(1 , 20)

# likelihood at each value in grid
likelihood <- dbinom(6, size = 9 , prob = x)

posterior <- likelihood * prior / sum(likelihood * prior)
```



Our posterior sums to 1, but thanks to flat prior its shape is the same as the likelihood.

Lets do this datapoint-by datapoint, so that n = 1. First datapoint is that the 1st patient died.

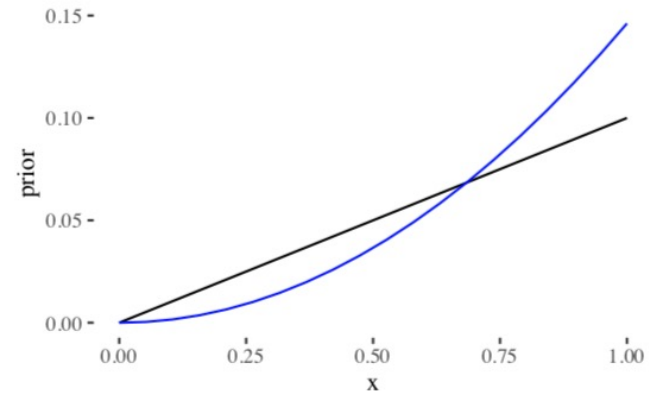
```
likelihood <- dbinom(1, size = 1, prob = x)
posterior <- likelihood * prior/sum(likelihood * prior)
ggplot(data = NULL) +
geom_line(aes(x, posterior), color = "blue") +
ggthemes::theme_tufte()
```



Zero mortality is now logically impossible and 100 percent mortality is the hypothesis best supported by the data.

Second datapoint is that the 2nd patient died. Our previous posterior is now the prior.

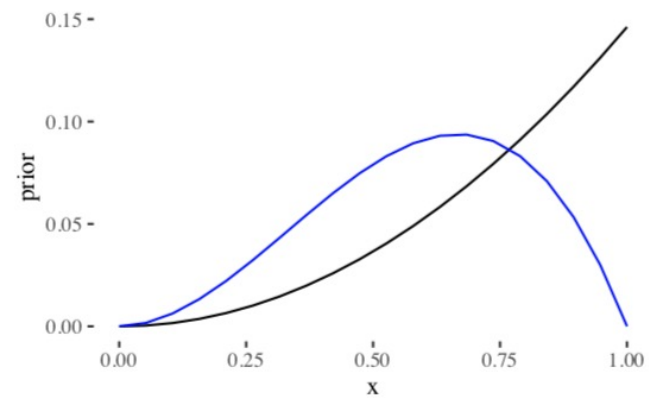
```
prior <- posterior
likelihood <- dbinom(1, size = 1, prob = x)
posterior1 <- likelihood * prior/sum(likelihood * prior)
ggplot(data = NULL) +
geom_line(aes(x, prior)) +
geom_line(aes(x, posterior1), color = "blue") +
ggthemes::theme_tufte()
```



The posterior is no longer a straight line. 100 percent mortality is still the most likely mortality estimate.

Third datapoint is that the 3d patient survived.

```
prior <- posterior1
likelihood <- dbinom(0, size = 1, prob = x)
posterior2 <- likelihood * prior/sum(likelihood * prior)

ggplot(data = NULL) +
geom_line(aes(x, prior)) +
geom_line(aes(x, posterior2), color = "blue") +
ggthemes::theme_tufte()
```

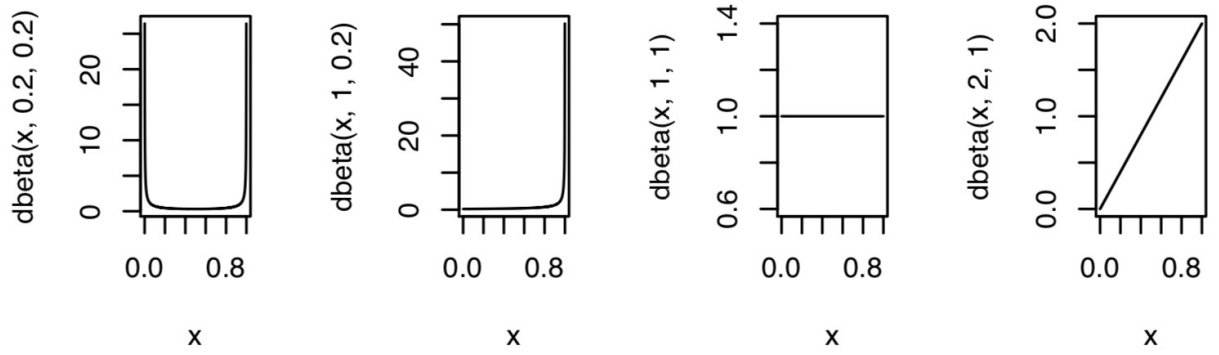


Now zero mortality and 100 percent mortality are both logically impossible. The most likely value for mortality is 2/3 or 75 percent.

model a proportion (p)

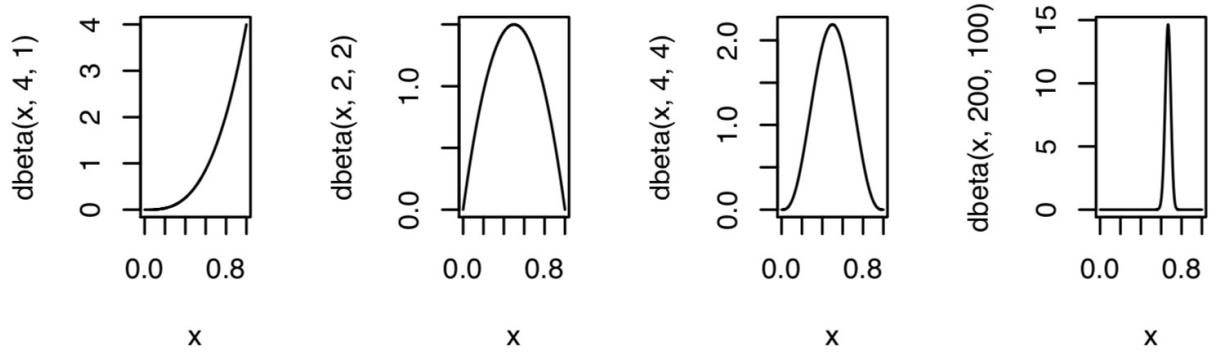
y ~ Binomial(n, p) or y I trials(n) ~ Binomial(p)

p ~ Beta(1,1)



brm(y I trials(n)~1, family="binomial", link = "identity")

brm(y ~1, family="bernoulli", link = "identity")



adding a predictor to binomial model

y I trials(n) ~ Binomial(p) Successes come stochastically from binomial distribution with probability p

p = a + bx **p is redefined as regression equation**

a ~ N(.,.)

b ~ N(.,.)

This model means that p can be >1 or >0!

brm(y I trials(n) ~ x, family="binomial, link = "identity")

adding a logit link

y I trials(n) ~ Binomial(p)

logit(p) = a + bx

a ~ N(.,.)

b ~ N(.,.)

now p is bounded by 0 and 1

brm(y I trials(n) ~ x, family="binomial, link = "logit")

logistic regression

- we are not modelling y = {0 , 1}, but instead the Pr that Y = 1 conditional on X, or P(Y = 1 | X)
- Furthermore, we are not modelling P(Y = 1 | X) directly, but use logistic transformation of the process model.

$$P(Y = 1|X) = \frac{exp(a + bx)}{1 + exp(a + bx)}$$

- the reverse function is logit transformation of P(Y = 1 | X), which gives us logarithm of odds

$$odds = \frac{P(Y = 1 \mid X)}{1 - P(Y = 1 \mid X)} = exp(a + bx)$$

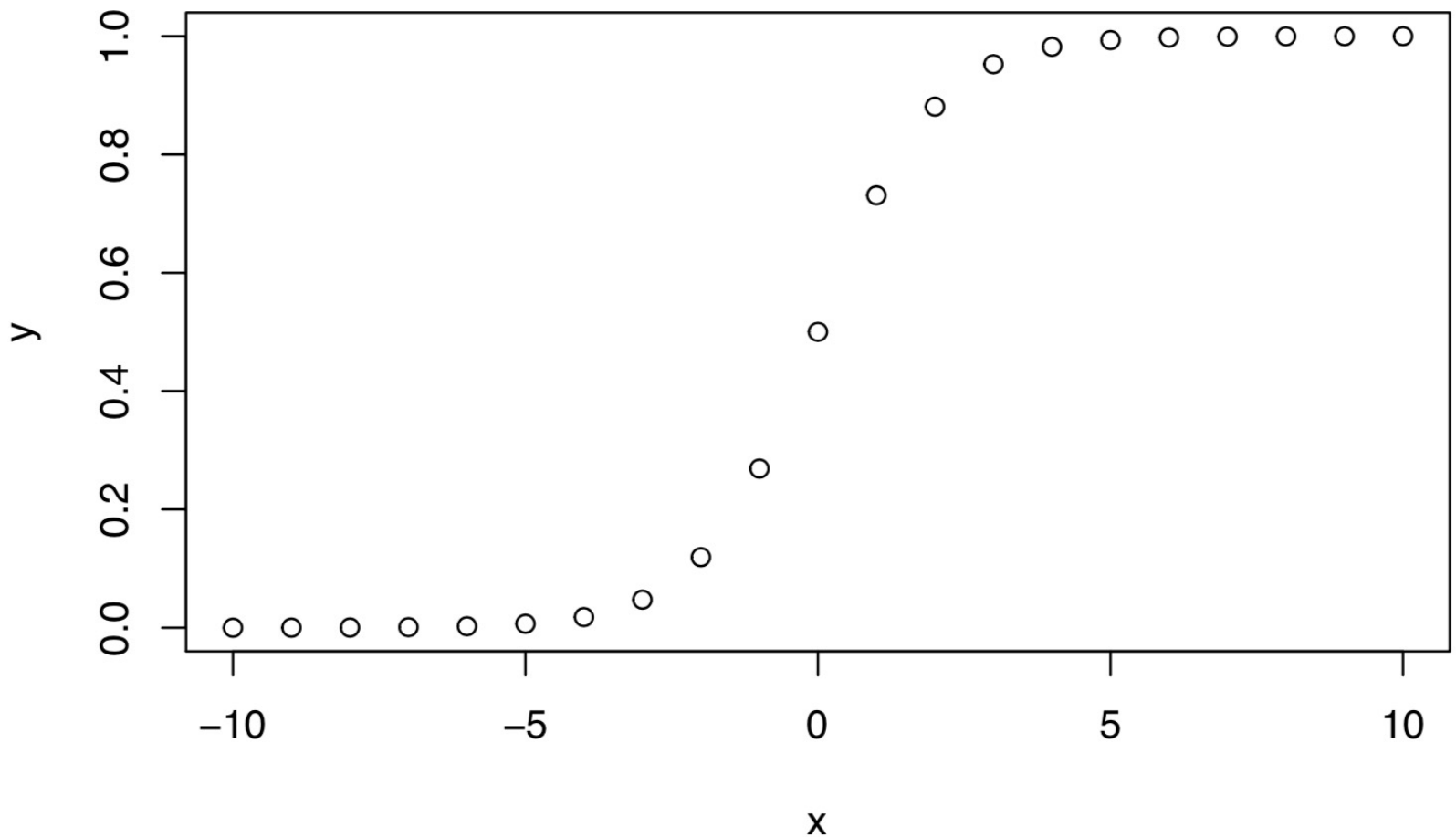
ja ekvivalentselt

$$log(odds) = logit(p) = a + bx$$

Suvalise arvu α logistiline funktsioon on logiti põõrdväärtus:

$$logit^{-1}\alpha = logistic(\alpha) = \frac{exp(\alpha)}{1 + exp(\alpha)}$$

```
x <- -10:10
y <- exp(x) / (1+exp(x))
plot(y~x)
```



- if we fit a logistic regression model $y = a + bx$ and change x value by 1 unit, then log-odds will change by b units, or odds will change by $exp(b)$ units.
- b does not say, how much $P(Y = 1|X)$ will change if x changes by 1 unit. This change depends on the value of X.
- but as long as $b > 0$, any increase in x will lead to some increase in $P(y=1)$
- The difference of logits of 2 probabilities = log(Odds Ratio)

$$log(OR) = logit(p_1) - logit(p_2)$$

Odds-ratio

Kui meil on 2 katsetingimust (ravim/platseebo) ning 2 väljundit (näit elus/surnud), siis

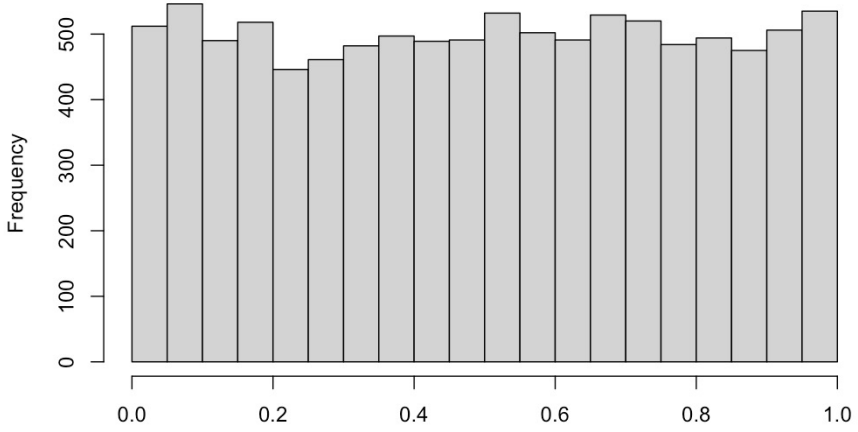
- a - ravim/elus juhutude arv,
- b - ravim/surnud juhutude arv,
- c - platseebo/elus juhutude arv,
- d - platseebo/surnud juhutude arv.

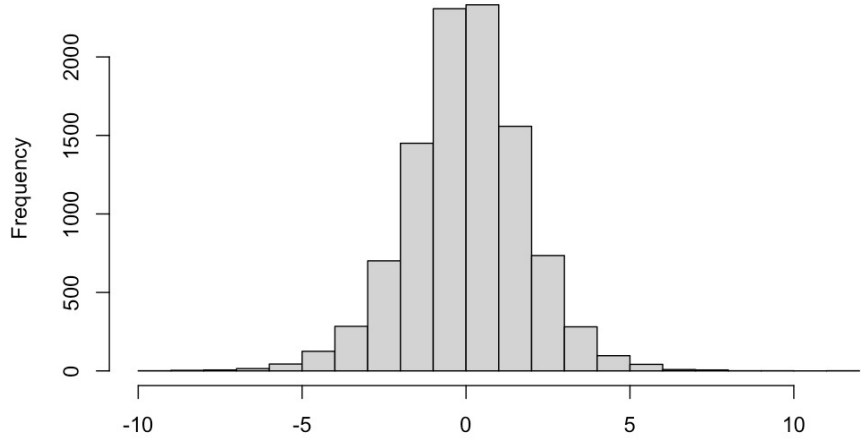
$$OR = \frac{a/b}{c/d}$$

- logistic regression generalizes OR from 2 binary vars. with binary y-var and binary x-var (x_1) pluss other regressors the β_1 is connected with conditional OR.
- exp(β_1) gives the OR between Y and X, if the values of ohter regressors are fixed.
- How to interpret OR as a relative effect depends on the base Pr of the event y=1. If the event is very rare then even OR=10 can mean a very small increase in the probability of event.
- brms::inv_logit_scaled() transforms your coef into probability scale

```{r}
set.seed(1)
rbeta(10000, 1, 1) %>% hist()
rbeta(10000, 1, 1) %>% logit_scaled() %>%  hist()
```

priors must be specified in logit scale!





Poisson likelihood

- there are many chances for an event to occur, but taken one-by-one they are all very unlikely.
- These events are independent of each other and exchangeable, the frequency of events is constant and two events cannot occur in the same time/place.
- The probability of an event is proportional to interval in time/size in space.
- Poisson distribution has a single parameter – mean number of events in an interval/space, or λ.
- SD = sqrt(lambda)

Matemaatiliselt näeb Poissoni jaotus välja niiviisi (k on sündmuste arv intervallis):

$$P(k) = e^{\lambda} \times \frac{\lambda^k}{k!}$$

Poissoni (ja negatiivne binoom) regressioonimudel näeb välja niimoodi:

$$log[E(Y|X)] = \beta X$$

Neg binomial likelihood

- most biol data are over-dispersed for Poisson. Poisson distribution presumes σ^2 scales with μ. The negative binomial distribution relaxes this assumption and presumes "each Poisson count observation has its own rate.

Negatiivne binoomjaotus (ehk gamma-poisson) eeldab, et iga Poissoni sündmus toimub oma sagedusega. Gammajaotus annab nende sageduste jaotuse. Jaotusel on kaks parameetrit, λ ja ϕ.

$$y_i \sim NegBinomial(\lambda_i, \phi)$$

Kui ϕ läheneb nullile, läheneb jaotus sama μ-ga normaaljaotusele. ϕ on alati positiivne ja määrab gamma-poissoni jaotuse dispersiooni nii: $\lambda + \lambda^2/\phi$. Suurem ϕ lähendab seda jaotust puhtale poissonile.

λ-le saab lisada lineaarse mudeli log-linki abil. NB! Bioloogilised protsessid, mida te hataksite mudeldada Poissoni tõepäramudeliga, tasub alati proovida mudeldada ka negbinoom tõepäraga, ja siis mudeleid võrrelda (loo ja pp-checkiga, näiteks).

- A beta-binomial model assumes that each binomial count observation has its own probability of success. The model estimates the *distribution* of probabilities of success across cases, instead of a single probability of success. And predictor variables change the shape of this distribution, instead of directly determining the probability of each success.

$$
\begin{aligned}
\text{admit}_i &\sim \text{BetaBinomial}(n_i, \bar{p}_i, \theta) \\
\text{logit}(\bar{p}_i) &= \alpha \\
\alpha &\sim \text{Normal}(0, 2) \\
\theta &\sim \text{Exponential}(1)
\end{aligned}
$$

custom likelihood for beta binomial in brms

```
beta_binomial2 <-
  custom_family(
    "beta_binomial2", dpars = c("mu", "phi"),
    links = c("logit", "log"), lb = c(NA, 0),
    type = "int", vars = "trials[n]"
  )

stan_funs <- "
  real beta_binomial2_lpmf(int y, real mu, real phi, int T) {
    return beta_binomial_lpmf(y | T, mu * phi, (1 - mu) * phi);
  }
  int beta_binomial2_rng(real mu, real phi, int T) {
    return beta_binomial_rng(T, mu * phi, (1 - mu) * phi);
  }
"

stanvars <-
  stanvar(scode = stan_funs, block = "functions")
```

```
brm(data = d,
    family = beta_binomial2,  # here's our custom likelihood
    admit | trials(applications) ~ 1,
    prior = c(prior(normal(0, 2), class = Intercept),
              prior(exponential(1), class = phi)),
    iter = 4000, warmup = 1000, cores = 2, chains = 2,
    stanvars = stanvars,  # note our `stanvars`
    seed = 11)
```

- Kui me viskame mitut münti, millel igaühel on erinev kulli saamise tõenäosus, siis saame kasutada binoomjaotuste segujaotust, ehk beeta-binoomjaotust.
- Poissoni protsessi korral, kus iga sündmus võib olla pärit erinevast poissoni jaotusest, saame kasutada negatiivset binoomjaoust, ehk gamma-piossoni jaotust (sünonüümid).
- Kui meil on tegu mõne sellise protsessiga, siis oleks parim lahendus konditsioneerida mudel X-muutuja(te)ga, mis viiksid Y-jaotuse lihtsale kujule (binoomjaotus, jne). Kui seda teha ei saa, siis järgmine trikk on kasutada neid eelpoolmainitud nn üle-disperseeritud jaotusmudeleid.
- Need on nn pidevad mixture mudelid, kus iga binoom- või poissoni jaotusega count eeldatakse omavat isiklikku beta- või gammajaotusega edutõenäosust. Pidevad selle pärast, et beta ja gammajaotus on pidevad jaotused.
- Praktikas on sageli parem alternatiiv pidevatele segumudelitele mitme-tasemelised (hierarhilised) mudelid