3.2 Practical: logistic regression

Benjamin Rosenbaum

November 4, 2021

Statistical model

Suppose we want to measure the effect of a continuous variable on presence / absence data.

E.g. presence of a species in different locations: $y = 1$ (presence) or $y = 0$ (absence), x = temperature in location.

Statistical model:

$$\begin{aligned} y_i &\sim \text{bernoulli}(p_i) \\ \text{logit}(p_i) &= a + b \cdot x_i \\ \text{or, equivalently:} \quad p_i &= \text{inv.logit}(a + b \cdot x_i) \end{aligned}$$

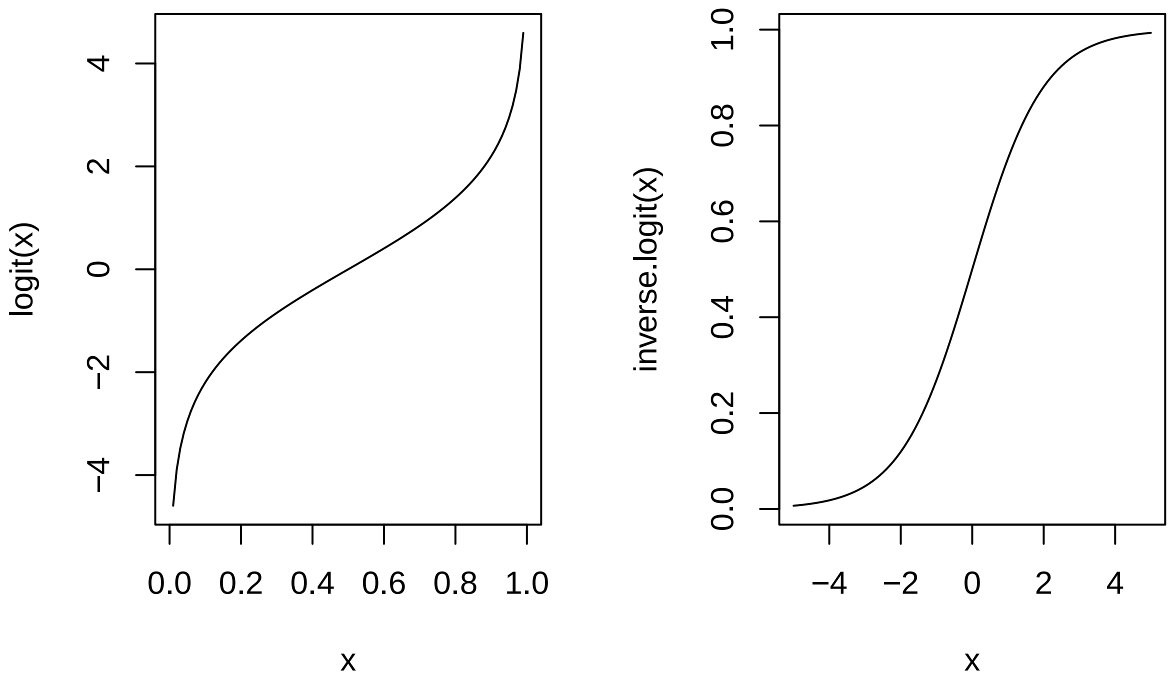
p_i is the probability of presence of the species, the distribution of y_i is bernoulli (unfair coin-flip with probability p for “1”).

We assume a linear relationship of logit(p) with temperature. (Realistically, a hump-shaped relationship would make more sense.)

inv.logit() transforms the values of the whole axis to the interval (0,1)

Research question: is there a positive effect of the predictor to the presence of the species?

```r
par(mfrow=c(1,2))
curve(qlogis, from=0, to=1, ylab="logit(x)")  # qlogis=logit
curve(plogis, from=-5, to=5, ylab="inverse.logit(x)") # plogis=inverse.logit
```



Research question: is there a positive effect of the predictor to the presence of the species?

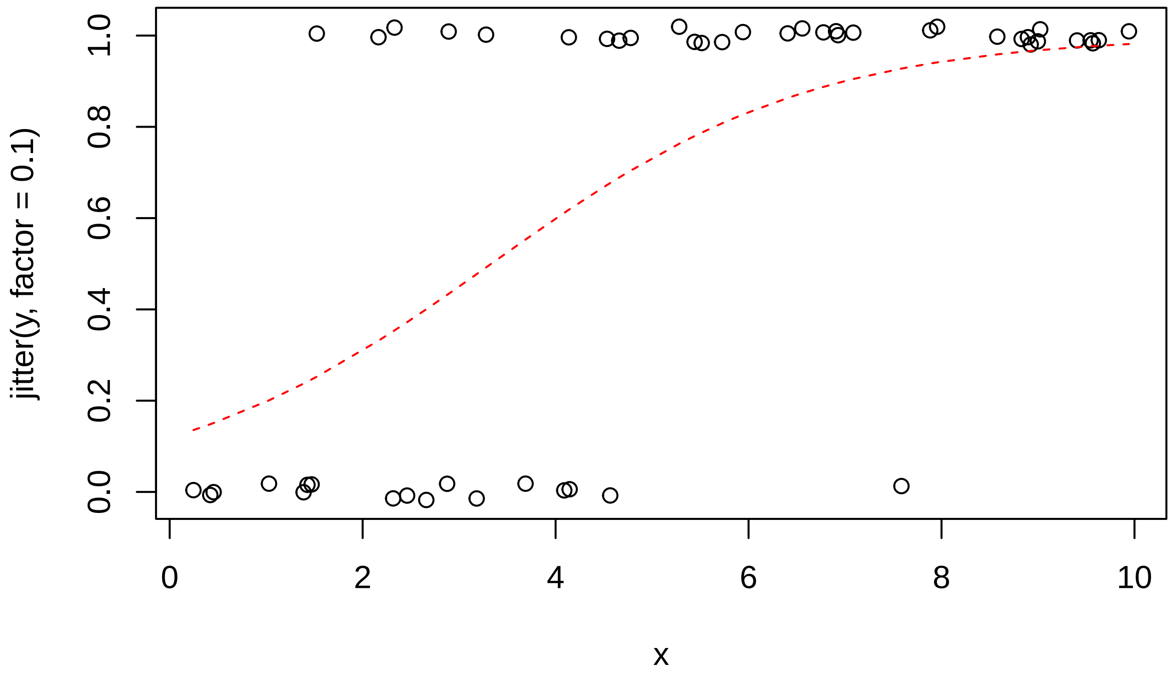
Setup

```
rm(list=ls())
library(rstan)
library(coda)
library(BayesianTools)
library(brms)
setwd("~/Nextcloud/teaching Bayes 2021")

rstan_options(auto_write = TRUE)
options(mc.cores = 3)
```

Generate data

```
set.seed(123) # initiate random number generator for reproducability

n = 50
x = sort(runif(n, 0, 10))

a = -2
b = 0.6

p = plogis(a+b*x)

y = rbinom(n=n, size=1, prob=p)

par(mfrow=c(1,1))
plot(x, jitter(y, factor=0.1))
```

```
lines(x,p, lty=2, col="red")
```



Stan code and fitting

```
data = list(n=n,
            x=x,
            y=y)

stan_code = '
data {
  int  n;
  real x[n];
  int  y[n];
}
parameters {
  real a;
  real b;
}
model {
  real p[n];
  // priors
  a ~ normal(0, 10);
  b ~ normal(0, 10);
  // likelihood
  for(i in 1:n){
    p[i] = inv_logit(a+b*x[i]);
    y[i] ~ bernoulli(p[i]);
  }
  // alternative vector notation:
  // y ~ bernoulli(inv_logit(a+b*x));
}
```

```
'
```

```
stan_model = stan_model(model_code=stan_code)
fit  = sampling(stan_model, data=data)
```

```
print(fit, digits=3, probs=c(0.025, 0.975))
```

```
## Inference for Stan model: d0c5be6f5ff6213b7f34126e22825ce5.
## 4 chains, each with iter=2000; warmup=1000; thin=1; 
## post-warmup draws per chain=1000, total post-warmup draws=4000.
## 
##         mean se_mean    sd    2.5%   97.5% n_eff  Rhat
## a     -2.856   0.036 0.952  -4.991  -1.221   700 1.004
## b      0.825   0.009 0.228   0.448   1.339   716 1.004
## lp__ -20.403   0.030 1.014 -23.187 -19.401  1128 1.003
## 
## Samples were drawn using NUTS(diag_e) at Thu Oct  7 10:59:17 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at 
## convergence, Rhat=1).
```

Positive b means increase in probability p with temperature x.

log odds increase by 0.82 for increase of 1 degree in temperature x (on average, this is still associated with uncertainty, sd=0.23).

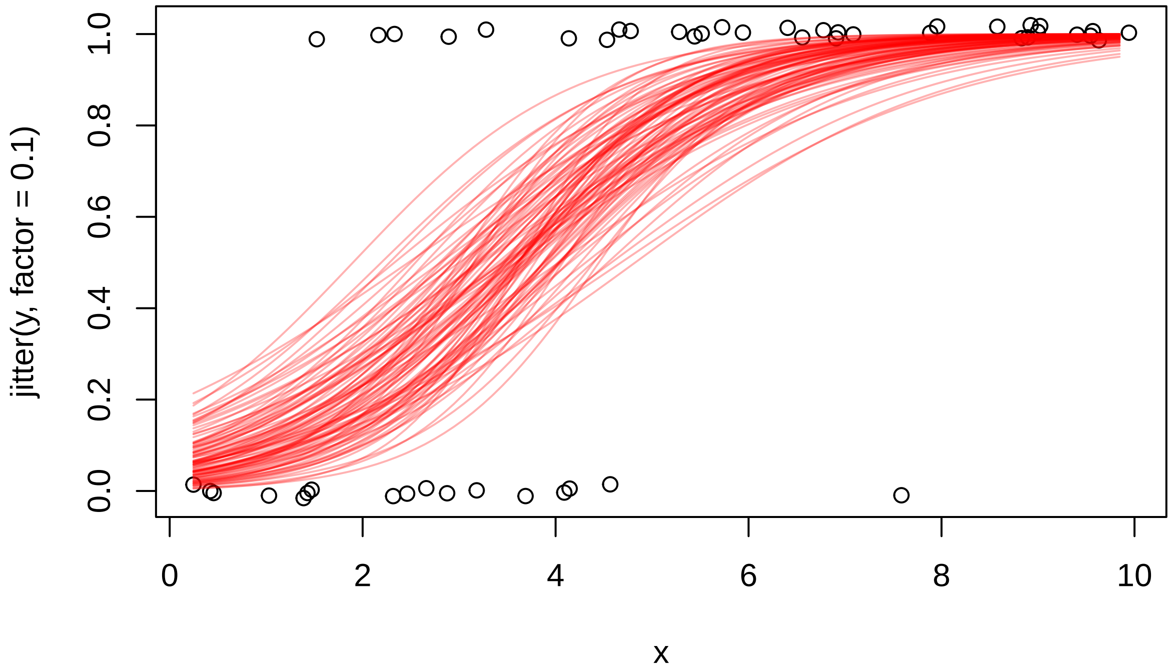
odds increase by exp(0.82)=2.27 multiplicatively, e.g. from 1:1 (=1/(1+1)=50% chance of species present) to (2.27*1):1 (=2.27/(2.27+1)=69%) when x increases by 1 degree. Or from 2:1 (=2/(2+1)=66%) to (2*2.27):1 (=4.54/(4.54+1)=82%).

Predictions

We generate credible intervals for the determinstic model (for p), 90%, but choose as you like.

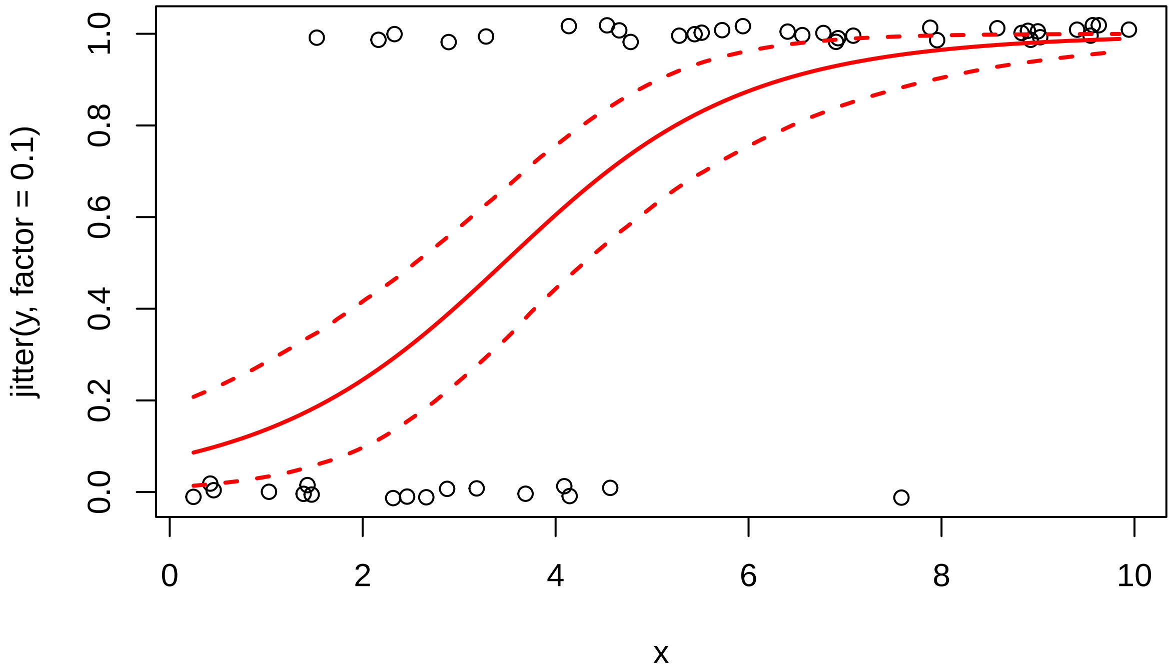
First, we plot some regression lines (for the first 100 posterior samples).

```
posterior=as.matrix(fit)

x.pred = seq(from=min(data$x), to=max(data$x), by=0.1)
p.pred = matrix(0, nrow=nrow(posterior), ncol=length(x.pred))

for(i in 1:nrow(posterior)){
  p.pred[i, ] = plogis(posterior[i,"a"] + posterior[i,"b"]*x.pred)
}

plot(x, jitter(y, factor=0.1))
for(i in 1:100){
  lines(x.pred, p.pred[i, ], col=adjustcolor("red", alpha.f=0.3))
}
```



Next, credible intervals (quantiles) for predictions are computed.

```
plot(x, jitter(y, factor=0.1))

p.pred.mean = apply(p.pred, 2, function(x) mean(x))
lines(x.pred, p.pred.mean, col="red", lwd=2)

p.pred.q05 = apply(p.pred, 2, function(x) quantile(x, probs=0.05))
lines(x.pred, p.pred.q05, col="red", lwd=2, lty=2)

p.pred.q95 = apply(p.pred, 2, function(x) quantile(x, probs=0.95))
lines(x.pred, p.pred.q95, col="red", lwd=2, lty=2)
```



Observed vs predicted plots are not that useful in logistic regression, since the response y is either 0 or 1.

brms fit

Frequentist solition: glm

```
summary( glm(y~x, family=binomial(link="logit")) )
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial(link = "logit"))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4985  -0.6016   0.1844   0.5721   1.8230
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.5901     0.8925  -2.902 0.003709 **
## x             0.7471     0.2108   3.543 0.000396 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64.104  on 49  degrees of freedom
## Residual deviance: 38.682  on 48  degrees of freedom
## AIC: 42.682
##
## Number of Fisher Scoring iterations: 5
```

If we code the same model with brms, we receive a complaint that we should use bernoulli (which is binomial with n=1 trials).

```
fit.b1 = brm(y ~ x,
            family=binomial(link="logit"),
            data = data)
```

```
fit.b1 = brm(y ~ x,
            family=bernoulli(link="logit"),
            data = data)
```

```
fit.b1
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: y ~ x
##    Data: data (Number of observations: 50)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -2.86      0.93    -4.82    -1.17 1.00     2066     2263
## x             0.81      0.22     0.42     1.28 1.00     1345     1673
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

We can also specify a prior for the effect of temperature. But it's the effect size for the log odds, which is not that intuitive. We use a very wide prior.

```
priors = c(prior(normal(0,10), class=b))
```

```
fit.b2 = brm(y ~ x,
            family=bernoulli(link="logit"),
            data = data,
            prior = priors)
```

```
fit.b2
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: y ~ x
##    Data: data (Number of observations: 50)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -2.85      0.94    -4.84    -1.16 1.00     2105     2389
## x             0.81      0.22     0.43     1.30 1.00     1398     1649
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

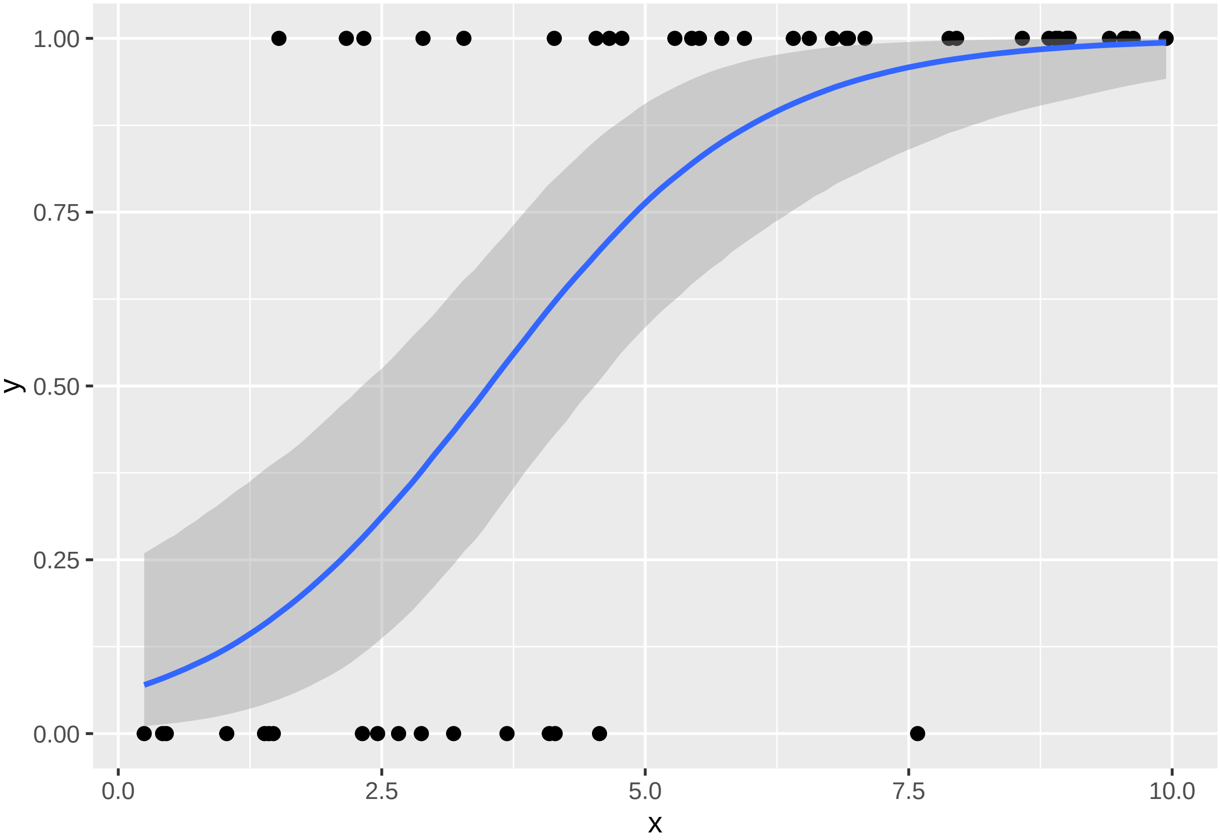
The effect of temperature on species occurrence is positive.

```
hypothesis(fit.b2, "x>0", class="b")
```

```
## Hypothesis Tests for class b:
##   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1    (x) > 0     0.81      0.22     0.47      1.2        Inf         1    *
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

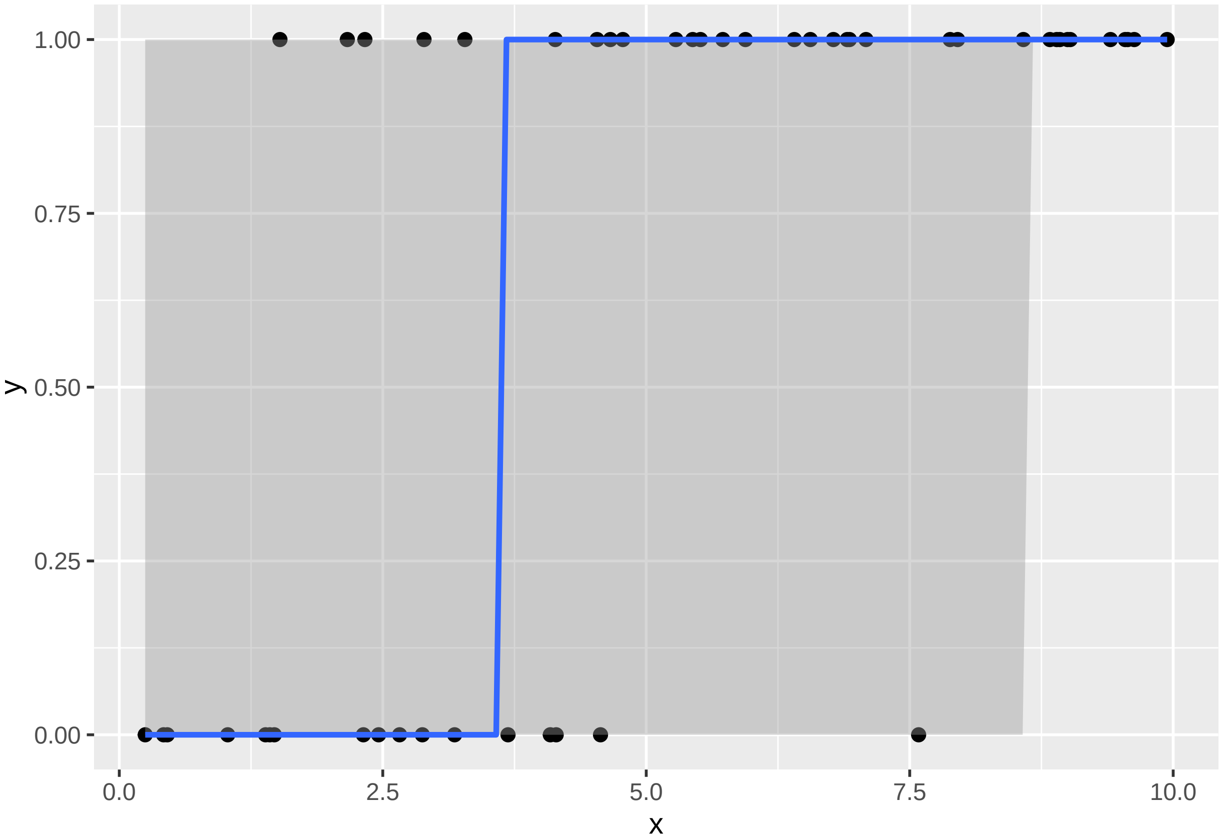
`conditional_effects()` automatically plots predictions for the deterministic model, which is the occurrence probability `p=inv.logit(a+b*x)`. The default is `method="posterior_epred"`.

```
plot(conditional_effects(fit.b2),
     points=TRUE)
```



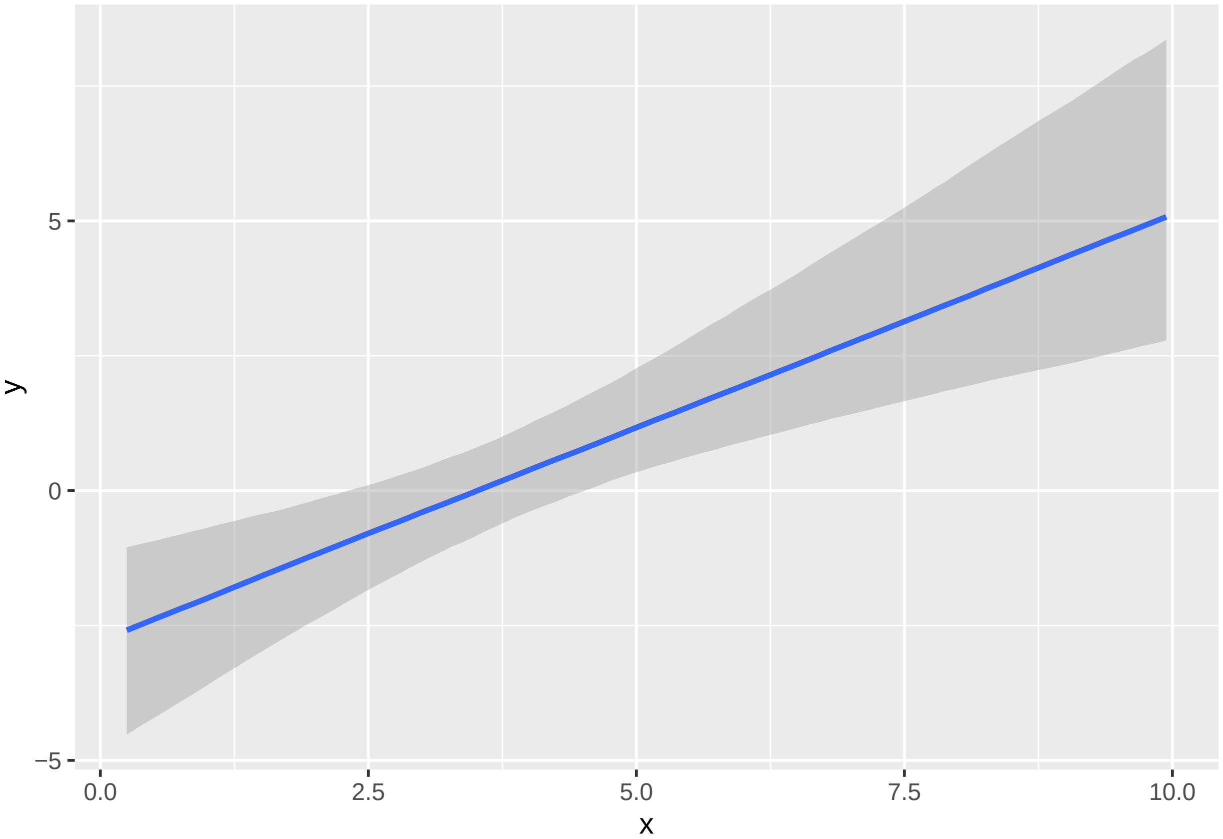
Using predictions including the stochastic part, i.e. predicting datapoints will generate only zeros and ones: `y~bernoulli(p)`. The blue line is the median (not the mean), which like all quantiles can also be only 0 or 1 !

```
plot(conditional_effects(fit.b2, method="posterior_predict"),
     points=TRUE)
```



Finally, we can also plot the linear predictor only `(a+b*x)` without the link function

```
plot(conditional_effects(fit.b2, method="posterior_linpred") )
```



Predictions for deterministic and stochastic model for the data (or specified newdata) are computed with `fitted()` and `predict()`. Note that quantiles for the prediction are 0 or 1, but the column `Estimate` is the mean here.

```
pred.fit.b2 = fitted(fit.b2)
head(pred.fit.b2)
```

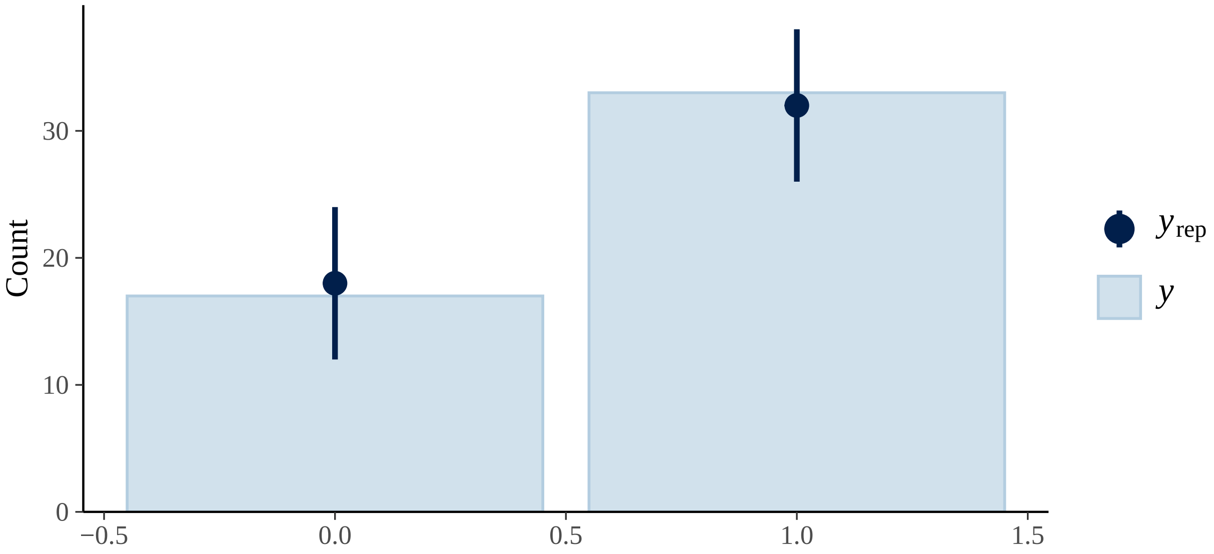
```
##        Estimate  Est.Error       Q2.5     Q97.5
## [1,] 0.08671592 0.06561733 0.01071003 0.2593894
## [2,] 0.09621217 0.06905373 0.01333227 0.2754176
## [3,] 0.09823697 0.06974869 0.01386073 0.2786174
## [4,] 0.13807634 0.08117766 0.02750257 0.3383194
## [5,] 0.17036593 0.08794722 0.04221942 0.3799536
## [6,] 0.17435904 0.08865965 0.04422847 0.3843148
```

```
pred.fit.b2 = predict(fit.b2)
head(pred.fit.b2)
```

```
##      Estimate Est.Error Q2.5 Q97.5
## [1,]  0.08725 0.2822363    0     1
## [2,]  0.09700 0.2959948    0     1
## [3,]  0.09900 0.2986994    0     1
## [4,]  0.13325 0.3398872    0     1
## [5,]  0.17700 0.3817164    0     1
## [6,]  0.17000 0.3756798    0     1
```

Observed vs residuals and PPC (posterior predictive checks) with density plots don't make much sense, since the response y is either 0 or 1. But PPC can be plotted with a bars plot (counting zeroes and ones for data and model predictions)

```r
pp_check(fit.b2, ndraws=100, type="bars")
```


Count
y_{rep}
y