Logistische Regression

Ein kurzer Einführungstext

Holger Sennhenn-Reulen
Department of Growth and Yield,
Northwest German Forest Research Institute.

March 3, 2021

Contents

1 Organise R Session

```
rm(list = ls())
library("viridis")
library("lme4")
```

2 Introduction

In der Regressionsanalyse geht es allgemein darum, die Auswirkungen von einer oder mehreren Einflußgrößen $x_1, x_2, \ldots$ auf abhängige Zufallsvariable Y abzuschätzen. Im Fall der linearen Einfachregression wird dies über die Beziehung:

$$Y_i = \beta_0 + \beta_1 x1, i + \epsilon_i \tag{1}$$

vollzogen, wobei Index i die Zugehörigkeit zu Beobachtungseinheit i bezeichet, Parameter β_0 den Wert der Regressionsgerade $\beta_0 + \beta_1 x1, i$ für $x_{1,i} = 0$ bezeichnet, Parameter β_1 die Veränderung dieser Regressionsgeraden wenn sich $x_{1,i}$ um eine Einheit vergrößert, und ϵ_i ist ein sogenannter Residualterm, an welchen wir in der linearen Einfachregression die Annahme stellen dass dieser einer Normalverteilung folgt, sowie dass alle ϵ_i derselben Normalverteilung abstammen und unabhängig daraus resultieren, kurz: ϵ_i wird angenommen als unabhängig und identisch verteilt bezüglich einer Normalverteilung mit Erwwartungswert 0 und Varianz σ^2, oder noch kürzer:

$$\epsilon_i \overset{\text{u.i.v}}{\sim} \mathrm{N}\left(0, \sigma^2\right) \tag{2}$$

Für ein erstes Beispiel simulieren wir $N = 25$, $i = 1, \ldots, N$, Werte aus der Normalverteilung mit Erwartungswert 0 und Varianz 0.5^2:

```
N <- 25
set.seed(123456789) ## Setzen eines Startpunktes zur Reproduktion der Simulation.
epsilon <- rnorm(n = N, mean = 0, sd = .5)
round(epsilon, 5)
```

```
 [1]  0.25244  0.19794  0.70777 -0.36116 -0.30918 -0.78131  0.06398 -0.07848
 [9] -0.75767  0.58080 -0.53083  0.52629 -0.54516 -0.47522  0.09445 -0.65346
[17] -0.54648  0.58371  0.59906 -0.98130  0.36336  0.53352 -0.90729  0.00440
[25] -0.16350
```

Wir bilden die Werte einer Einflussgröße x als eine Sequenz der Länge 25 mit gleichen Abständen zwischen -1 und 1:

```
x <- seq(-1, 1, length.out = N)
```

Für den Parameter β_0 nehmen wir den Wert 0.75 an, für den Parameter β_1 den Wert -1:

```
eta <- 0.75 + -1 * x
```

Wir nennen diese Größe η, welche hier die Werte der Regressionsgeraden an den Werten von x abbildet, allgemein auch den 'linearen Prädiktor':

$$\eta_i = \beta_0 + \beta_1 x_i$$

In der linearen Regression werden nun die Werte y der abhängigen Variablen Y ohne (in der Regel) eine weitere Transformation als Addition von linearem Prädiktor und Residualterm gebildet:

```
y <- eta + epsilon
```

Durch diese Form der Addition, sowie der Eigenschaft, dass wir für ϵ den Erwartungswert 0 angenommen haben, ergibt sich für den Erwartungswert von Y in Abhängigkeit (man sagt 'bedingt auf') von x:

$$\mathrm{E}\left(Y_i \mid x_i\right) = \eta_i + 0 = \eta_i.$$

Wir sprechen hier vom bedingten Erwartungswert von Y, und erhalten weiterhin für die volle Verteilung von Y:

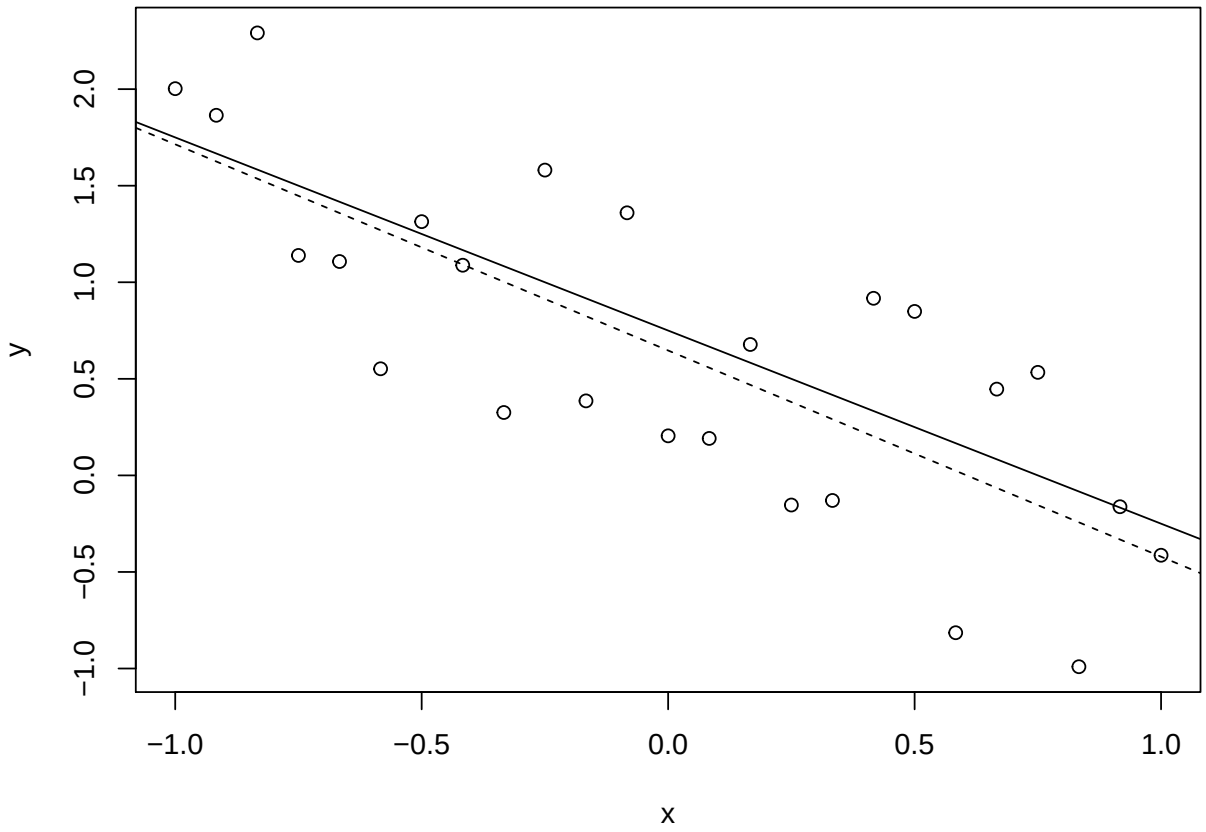
$$Y \sim \text{Normal}\left(\eta_i, \sigma^2\right).$$

Wir stellen dies einmal grafisch dar:

```
plot(x, y)
abline(a = .75, b = -1)
m <- lm(y ~ x)
coef(m)
```

```
(Intercept)           x
 0.6466671  -1.0676225
```

```
abline(a = coef(m)[1], b = coef(m)[2], lty = 2)
```

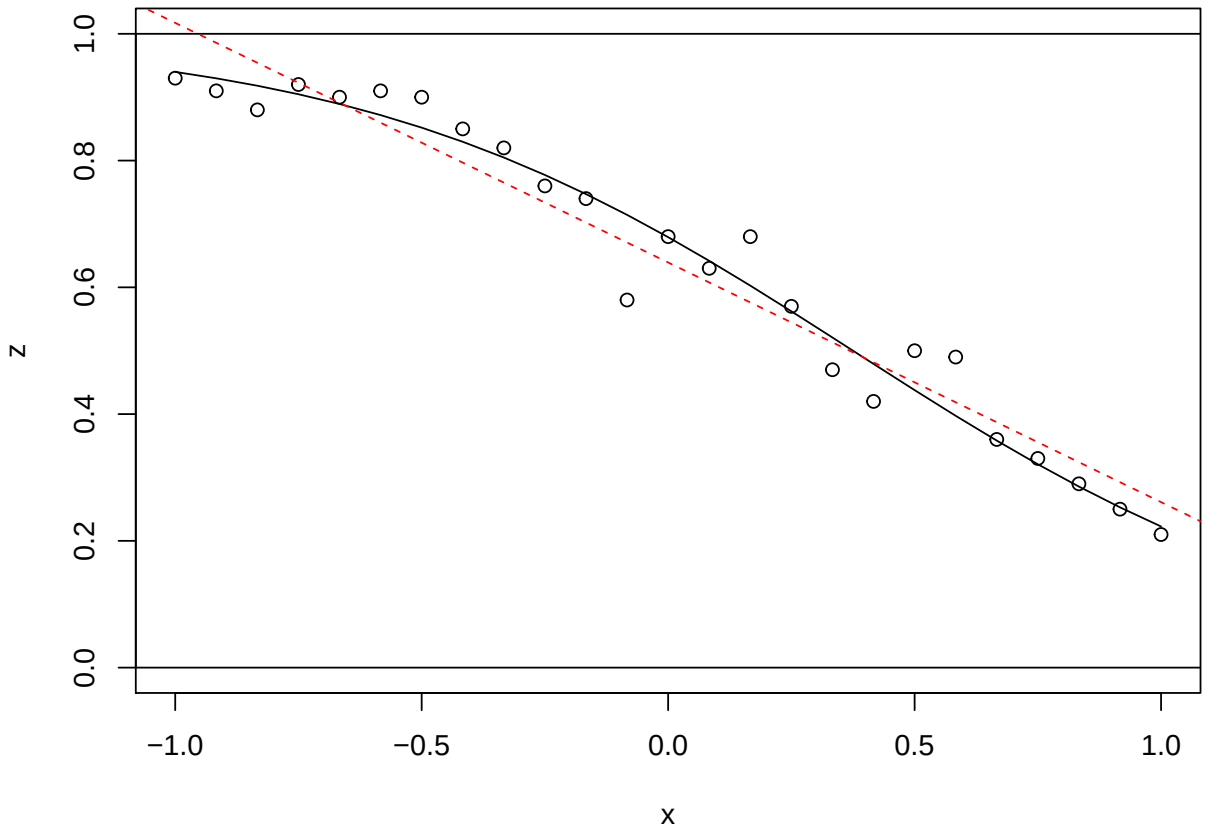


Wir konnten hier mit der Funktion `lm` den von uns zugrundegelegten Daten-generierenden Prozess ganz gut nachbilden.
In der praxis können wir jedoch nicht ein solch einfaches Modell nutzen, da meist durch unser Vorwissen über den Daten-generierenden Prozess die ein oder andere Annahme hier nicht gehalten werden kann. Zum Beispiel könnte unsere abhängige Variable ein prozentualer Anteil zwischen 0 und 1 sein. Nach unserer obigen Gleichung 2 würde für die lineare Einfachregression aber eine Verletzung dieses Wertebereichs für aus dem Modell prädiktierte Werte nicht ausgeschlossen sein. Ich verdeutliche dies ohne viele begleitende Worte einmal auf Basis der folgenden Simulation:

```
z <- rbinom(n = N, size = 100, prob = plogis(.75 - 2 * x)) / 100
plot(x, z, ylim = c(0, 1))
lines(x, plogis(.75 - 2 * x))
abline(h = c(0, 1))
m <- lm(z ~ x)
coef(m)
```

```
(Intercept)           x
 0.6392000  -0.3780923
```

```
abline(a = coef(m)[1], b = coef(m)[2], lty = 2, col = 2)
```



Wir finden hier ein geeigneteres statistisches Verfahren in dem wir näher am Daten-generierenden Prozess modellieren.
Dies ist in diesem Fall ein *logistisches Regressionsmodell* für binomial-verteilte abhängige Variablen, ein Modell aus der Klasse der generalisiertes linearen Modelle (GLM). Wie die Bezeichnung bereits sagt, nehmen wir nun nicht mehr eine Normalverteilung der abhängige Variablen, sondern eine Binomialverteilung an. Die Binomialverteilung kann definiert werden als die sich ergebende Summe, wenn man die Ergebnisse mehrerer unabhängig durchgeführter Bernoulli-Experimente addiert. Im Spezialfall nur eines Bernoulli-Experiments ergibt sich die Bernoulli-Verteilung. Die Binomial-Verteilung hat zwei Parameter, die *Anzahl der Trials*, n – das ist also die Anzahl der unabhängig durchgeführten Bernoulli-Experimente –, und die Wahrscheinlichkeit p, dass in einem Bernoulli-Experiment das Ergebnis erscheint welches wir mit der Zahl 1 kodieren. Kurz zum Bernoulli-Experiment: Das ist jedes Zufallsexperiment bei dem nur zwei versch. Ereignisse als Ergebnis auftreten können, z.B. ob irgendein zufällig ausgewählter Baum ein Laub- oder Nadelbaum ist, ob in einer Falle eine Maus gefangen wurde, oder nicht, ob es morgen regnet, oder nicht, ….
Der Erwartungswert einer binomal-verteilten Zufallsvriablen Y ist dann:

$$\mathrm{E}(Y) = n \cdot p, \tag{3}$$

die Varianz ist:

$$\mathrm{Var}(Y) = \frac{p(1-p)}{n}, \tag{4}$$

In einem logistischen Regressionsmodell für binomial-verteilte abhängige Variablen modellieren wir nun diesen Erwartungswert dadurch dass wir den Parameter p_i durch Kovariableneinflüsse variieren lassen, vergleichbar dazu wie wir in der linearen Einfachregression den Parameter η_i durch Kovariableneinflüsse haben variieren lassen. Nun müssen wir hier jedoch beachten, dass der Parameter p_i in Abhängigkeit von x_i nur im Intervall zwischen 0 und 1 variieren kann. Dafür nutzen wir eine

Transformation-Funktion für den linearen Prädiktor die dies sicherstellt, und die für die Regressionsmodellierung binomial-verteilter Zielvariablen am häufigsten genutzte Funktion ist hier die logistische Verteilungs-Funktion:

$$g(a) = \frac{\exp(a)}{1 + \exp(a)}.$$

Wir erhalten für einen linearen Prädiktor mit Einfluss größe x_i:

$$\mathrm{E}(Y \mid x_i) = n_i p_i = n_i \frac{\exp(\eta_i)}{1 + \exp(\eta_i)} = n_i \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}. \tag{5}$$

Durch die Erhebung einzelner Bernoulli-Experimente an einem Ort – bezeichnet durch Index $j = 1, \ldots, J$; J ist also die Anzahl der Orte / Flächen / Gruppen / …– kann diese Gleichung jedoch fehlspezifiziert sein, insofern als dass durch standörtliche Eigenschaften die Ergebnisse der Experimente nicht mehr allein durch x erklärt werden können. Sind solch standörtliche Eigenschaften nicht durch weitere Kovariablen greifbar, so bietet es sich an eine jede Abweichung eines Standorts durch einen eigenen Indikator γ_j zu modellieren:

$$\eta_{i,j} = \beta_0 + \beta_1 x_{1,i} + \gamma_j \tag{6}$$

Da in der Regel hier eine größere Menge an Indikatoren – J viele – zu schätzen sind, ist es meist effektiver alle Indikatoren über eine gemeinsame Verteilung zu schätzen – als Erklärung hierfür dient z.B. Efron & Morris (1977): *Stein's Paradox in Statistics*, Scientific American 236(5):119-127 –:

$$\gamma_j \overset{\text{u.i.v}}{\sim} \mathrm{N}\left(0, \sigma_\gamma^2\right) \tag{7}$$

Man nennt ein solches Modell ein gemischtes Modell und die Parameter γ_j werden als Random Intercept-Koeffizienten bezeichnet. In R bietet sich in einer nicht-bayesianischen Modellierung für solch ein gemischtes logistisches Modell die Funktion `glmer` aus dem `lme4`-Paket. Ein Beispiel hierfür nun im Folgenden.

3 Datenmanagement

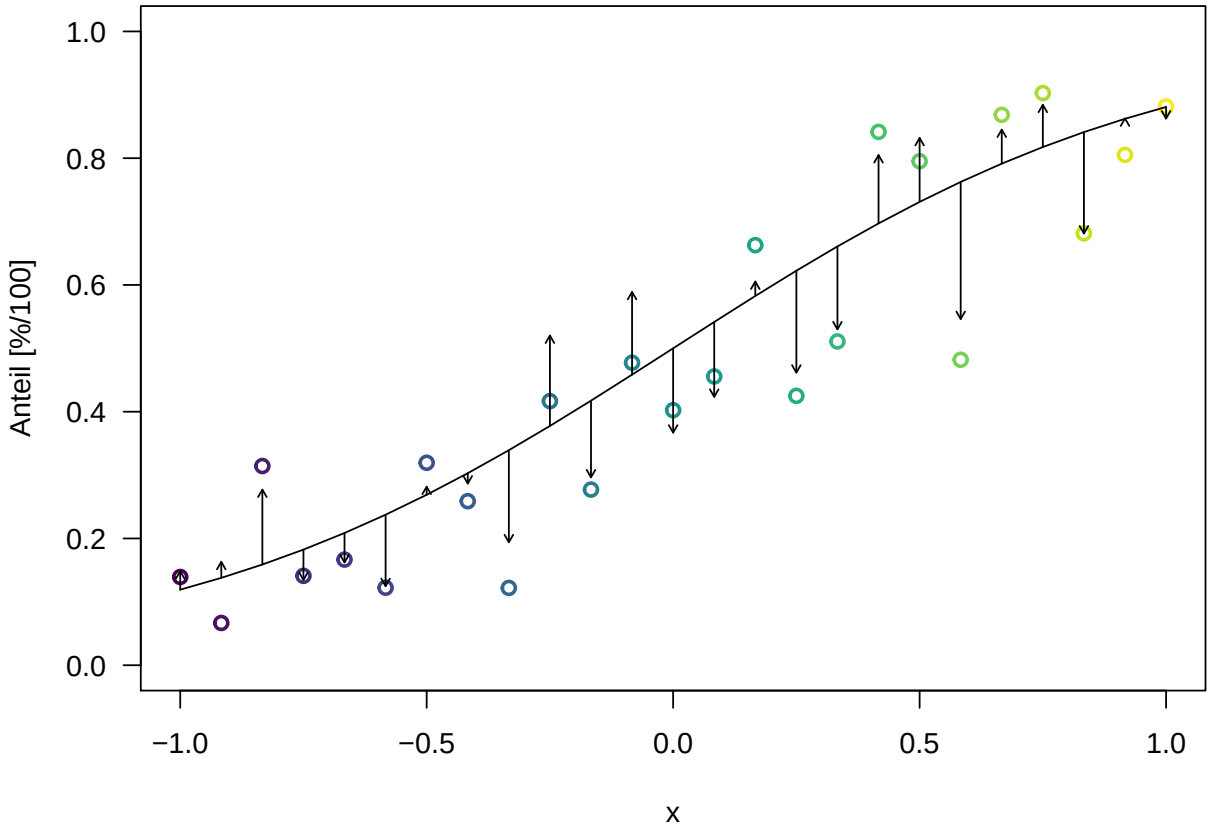
Simulation:

```
N <- 25
set.seed(123456789)
df <- data.frame(x = round(seq(-1, 1, length.out = N), 5),
                 gamma = rnorm(N, sd = .5),
                 n = sample(70:95, size = N, replace = TRUE))
df$y <- rbinom(n = N, size = df$n, prob = plogis(2 * df$x + df$gamma))
```

Deskriptive Darstellung:

- Die Linie stellt den zugrundeliegenden Einfluß der Größe x dar.
- Die von der Linie vertikal abgehenden Pfeile stellen die verschiedung dar die sich durch Abweichungen einer Gruppe `id` ergibt.
- Die Punkte sind dann der Anteil an *Cases*, $y = 1$, aller n *Trials* einer `id`. Da es sich hierbei um eine Zufallsstichprobe von jeweils Umfang n handelt, weichen die y-Koordinaten von den Spitzen der Pfeile ab. Wenn man für eine `id` einen immer höheren Sampling-Aufwand betreiben würde, so würden diesen Pseudo-Replikation ein immer repräsentativeres Bild der `id` abgeben, jedoch würde hier auch die Unabhängigkeit jeder einzelnen Messung von allen anderen Messungen an einem Ort immer weiter zurückgehen, das Sampling also immer weniger effizient werden.

```
plot(df$x, df$y / df$n, las = 1, col = viridis(n = nrow(df)), cex = 1, lwd = 2,
     ylim = c(0, 1), xlab = "x", ylab = "Anteil [%/100]")
lines(df$x, plogis(2 * df$x))
for (i in 1:nrow(df)) {
  arrows(x0 = df$x[i], x1 = df$x[i],
         y0 = plogis(2 * df$x[i]),
         y1 = plogis(2 * df$x[i] + df$gamma[i]), length = .05)
}
```



4 Modell ohne Berücksichtigung der Gruppierung

```
m <- glm(y/n ~ x, weights = n, data = df, family = binomial(link = "logit"))
summary(m)
```

```
Call:
glm(formula = y/n ~ x, family = binomial(link = "logit"), data = df,
    weights = n)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.3914  -1.7503  -0.3565   1.6471   4.0204

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.22002    0.05094   -4.32 1.56e-05 ***
x            1.92935    0.09718   19.85  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 636.88  on 24  degrees of freedom
Residual deviance: 128.47  on 23  degrees of freedom
AIC: 243.54

Number of Fisher Scoring iterations: 4
```

5 Modell mit Berücksichtigung der Gruppierung

```
dfl <- data.frame(id = rep(paste0("id", 1:nrow(df)), df$n),
                  x = rep(df$x, df$n))
print(table(dfl$id))
```

```
 id1 id10 id11 id12 id13 id14 id15 id16 id17 id18 id19  id2 id20 id21 id22 id23
  79   84   83   88   82   90   86   80   92   82   83   75   83   76   72   91
id24 id25  id3  id4  id5  id6  id7  id8  id9
  77   76   70   78   84   90   72   85   82
```

```
print(table(dfl$x))
```

```
      -1 -0.91667 -0.83333    -0.75 -0.66667 -0.58333     -0.5 -0.41667
      79       75       70       78       84       90       72       85
-0.33333    -0.25 -0.16667 -0.08333        0  0.08333  0.16667     0.25
      82       84       83       88       82       90       86       80
 0.33333  0.41667      0.5  0.58333  0.66667     0.75  0.83333  0.91667
      92       82       83       83       76       72       91       77
       1
      76
```

```
dfl$y <- 0
for (i in 1:nrow(df)) {
  index <- which(dfl$id == paste0("id", i))
  dfl$y[index[1:df$y[i]]] <- 1
}
xtabs(~ y + x, data = dfl)
```

```
   x
y   -1 -0.91667 -0.83333 -0.75 -0.66667 -0.58333 -0.5 -0.41667 -0.33333 -0.25
  0 68       70       48    67       70       79   49       63       72    49
  1 11        5       22    11       14       11   23       22       10    35
   x
y   -0.16667 -0.08333  0 0.08333 0.16667 0.25 0.33333 0.41667 0.5 0.58333
  0       60       46 49      49      29   46      45      13  17      43
  1       23       42 33      41      57   34      47      69  66      40
   x
y   0.66667 0.75 0.83333 0.91667  1
  0      10    7      29      15  9
  1      66   65      62      62 67
```

```
m <- glmer(y ~ x + (1 | id), data = dfl, family = binomial(link = "logit"))
summary(m)
```

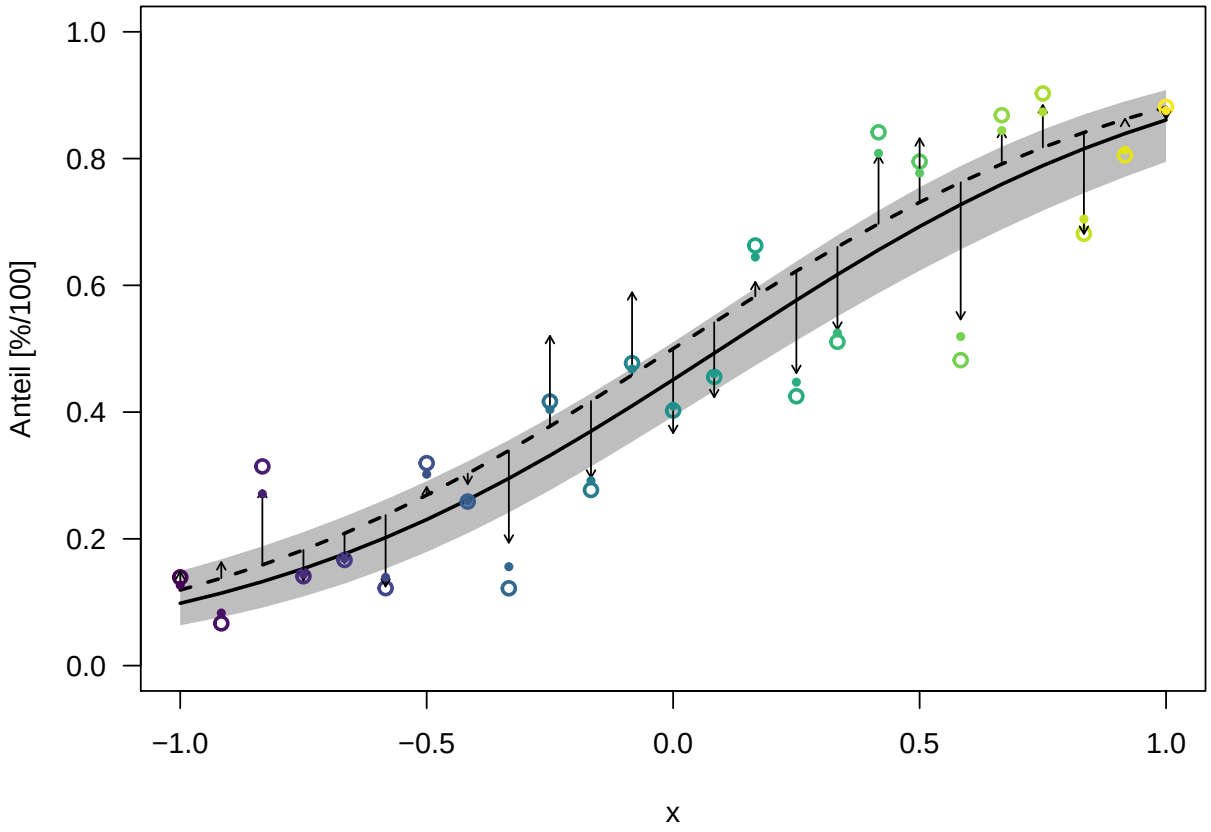
```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: y ~ x + (1 | id)
   Data: dfl

      AIC      BIC   logLik deviance df.resid
  2249.5   2266.3  -1121.7   2243.5     2037

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.6547 -0.6418 -0.3811  0.6476  3.3259

Random effects:
 Groups Name        Variance Std.Dev.
 id     (Intercept) 0.2923   0.5406
Number of obs: 2040, groups:  id, 25

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.1964     0.1213  -1.619    0.105
x             2.0193     0.2077   9.720   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation of Fixed Effects:
  (Intr)
x -0.015
```

```
b <- fixef(m)
V <- vcov(m)
v <- V[1, 1] + df$x^2 * V[2, 2] + 2 * df$x * V[1, 2]
l <- plogis(b[1] + b[2] * df$x + qnorm(p = .025) * sqrt(v))
```

```
u <- plogis(b[1] + b[2] * df$x + qnorm(p = .975) * sqrt(v))
plot(df$x, df$y / df$n, lwd = 2, type = "n", ylim = c(0, 1), las = 1,
     xlab = "x", ylab = "Anteil [%/100]")
## Punktweises 95%-Konfidenzintervall fuer den bedingten Erwartungswert
##   im Zentrum der Verschiebungen durch Gruppierung, Pr(y = 0 | x, gamma = 0).
polygon(c(df$x, rev(df$x)), c(l, rev(u)), col = "grey", border = NA)
lines(df$x, plogis(b[1] + b[2] * df$x), lwd = 2, lty = 1)
lines(df$x, plogis(0 + 2 * df$x), lwd = 2, lty = 2)
points(df$x, df$y / df$n, las = 1, col = viridis(n = nrow(df)), cex = 1, lwd = 2)
for (i in 1:nrow(df)) {
  arrows(x0 = df$x[i], x1 = df$x[i],
         y0 = plogis(2 * df$x[i]),
         y1 = plogis(2 * df$x[i] + df$gamma[i]), length = .05)
}
gamma_hat <- ranef(m)[[1]][rep(paste0("id", 1:nrow(df))), ]
cbind(ranef(m)[[1]], gamma_hat, df$gamma)
```

```
      (Intercept)   gamma_hat     df$gamma
id1   0.28647152  0.28647152  0.252436155
id10  0.31221559 -0.35601547  0.197937909
id11 -0.35412556  0.88881071  0.707768884
id12  0.23678635 -0.07057226 -0.361162154
id13 -0.16954873 -0.05171471 -0.309178478
id14 -0.13015550 -0.44735952 -0.781310179
id15  0.45554790  0.36806375  0.063979386
id16 -0.52019197 -0.01180782 -0.078476037
id17 -0.37701794 -0.81810996 -0.757668140
id18  0.79420544  0.31221559  0.580800816
id19  0.43619742 -0.35412556 -0.530831706
id2  -0.35601547  0.23678635  0.526285251
id20 -0.90458080 -0.16954873 -0.545158561
id21  0.53951984 -0.13015550 -0.475218176
id22  0.61478468  0.45554790  0.094447118
id23 -0.61734723 -0.52019197 -0.653456396
id24 -0.18286850 -0.37701794 -0.546483021
id25  0.12978606  0.79420544  0.583713767
id3   0.88881071  0.43619742  0.599064829
id4  -0.07057226 -0.90458080 -0.981299756
id5  -0.05171471  0.53951984  0.363355304
id6  -0.44735952  0.61478468  0.533523642
id7   0.36806375 -0.61734723 -0.907292618
id8  -0.01180782 -0.18286850  0.004398461
id9  -0.81810996  0.12978606 -0.163499350
```

```
points(df$x, plogis(b[1] + b[2] * df$x + gamma_hat), col = viridis(n = nrow(df)), pch = 16, cex = .7)
```


1.0
0.8
0.6
0.4
0.2
0.0
Anteil [%/100]
−1.0
−0.5
0.0
0.5
1.0
x