

# A multinomial study (model, regression, and mixed-effects)

Henrique Laureano

<http://leg.ufpr.br/~henrique>

Computational Methods for Statistical Inference  
Final Project

December 6, 2019



# The problem

Consider a random variable with **multinomial distribution** with three categories. In this case the probability function is given by

$$\mathbb{P}[X = x] = \frac{n!}{x_1!x_2!x_3!} p_1^{x_1} p_2^{x_2} p_3^{x_3}, \quad \text{with} \quad \sum_{i=1}^3 p_i = 1.$$

Consider a situation with just one trial, made in  $n$  independent subjects.

---

Three situations:

- » **Without** covariates, modeling directly the probabilities;
- » **With** covariates (regression);
- » With **repeated measures** (dependency structure/random effect).



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



# At first, getting some data

$n = 100$  independent subjects, one trial,  $k = 3$  categories.

```
probs <- c(.2, .6, .2) # probabilities for each category k
library(Matrix)
data_generator <- function(n, k = 3, p) {
  data <- t(rmultinom(n, 1, prob = p))
  colnames(data) <- paste0("p", seq(k))
  return(Matrix(data))
}
set.seed(1993)
data <- data_generator(n = 100, p = probs)
data[seq(5), ]
```

```
## 5 x 3 sparse Matrix of class "dgCMatrix"
##      p1 p2 p3
## [1,]  .  1  .
## [2,]  .  1  .
## [3,]  .  1  .
## [4,]  .  1  .
## [5,]  1  .  .
```



# Maximum Likelihood Estimators

First, we write the **log-likelihood** function for a single subject  $j$

$$L(\mathbf{p}) = f(x_j; \mathbf{p}) = \begin{cases} \frac{n!}{\prod_{i=1}^k x_i!} \prod_{i=1}^k p_i^{x_i}, & \text{when } \sum_{i=1}^k x_i = n, x_i \geq 0. \\ 0 & \text{otherwise,} \end{cases}$$

$$l(\mathbf{p}) = \log L(\mathbf{p}) = \log n! - \sum_{i=1}^k \log x_i! + \sum_{i=1}^k x_i \log p_i. \quad (1)$$

Second, we **code** it (next slide)



## MLEs: mostrando o pau

```
multinom_lkl <- function(par, xs) {  
  N <- nrow(xs) ; k <- ncol(xs)  
  ## k-1, since the last parameter is taken as the complementary  
  for (i in 1:(k - 1)) assign(paste0("p", i), par[i])  
  p <- unlist(mget(c(ls(pattern = glob2rx("~p\\d")))))  
  ps <- c(p, 1 - sum(p))  
  lkl_p <- rep(1, N) %*% xs %*% log(ps)  
  ## normalizing constant  
  lkl_c1 <- sum(lfactorial(rowSums(xs)))  
  lkl_c2 <- sum(lfactorial(xs))  
  lkl <- lkl_c1 - lkl_c2 + lkl_p  
  ## returning the negative of it  
  return(-as.numeric(lkl))  
}
```

$$\text{i.e., } \text{lkl} <- \sum_{j=1}^N \log n_j! - \sum_{j=1}^N \sum_{i=1}^k \log x_{ij}! + \sum_{j=1}^N \sum_{i=1}^k x_{ij} \log p_{ij}.$$



# MLEs: matando o gato

```
initial_values <- c(1/3, 1/3)
library(bbmle)
names(initial_values) <- parnames(multinom_lkl) <- c("p1", "p2")
model_fit <- mle2(multinom_lkl,
                  start = initial_values, vecpar = TRUE,
                  data = list(xs = data))
## estimated probabilities
c(model_fit@coef, p3 = 1 - sum(model_fit@coef))
```

```
##      p1      p2      p3
## 0.2300425 0.5899757 0.1799818
```

Equal to the MLEs,  $x_i/n$ . Behold

```
colSums(data)/nrow(data)
```

```
##   p1   p2   p3
## 0.23 0.59 0.18
```



## Some confidence intervals for the MLEs

First, we write the deviance

$$\text{deviance} \equiv D(\mathbf{p}) = -2 \{l(\mathbf{p}) - l(\hat{\mathbf{p}})\}.$$

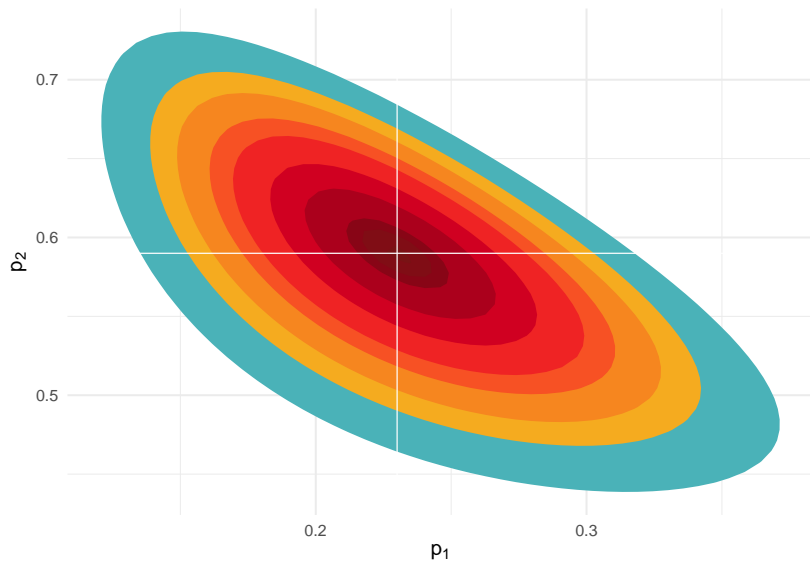
It is much more simple and pretty if we work here with it, instead of the log-likelihood function itself.

```
multinom_deviance <- function(p1, p2) {  
  par <- c(p1, p2)  
  log_ratio <- multinom_lkl(par, xs = data) - model_fit@min  
  deviance <- 2 * (log_ratio)  
  return(deviance)  
}  
multinom_deviance <- Vectorize(multinom_deviance, c("p1", "p2"))
```





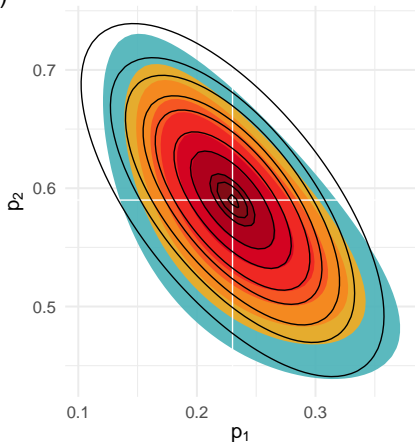
# Deviance contour



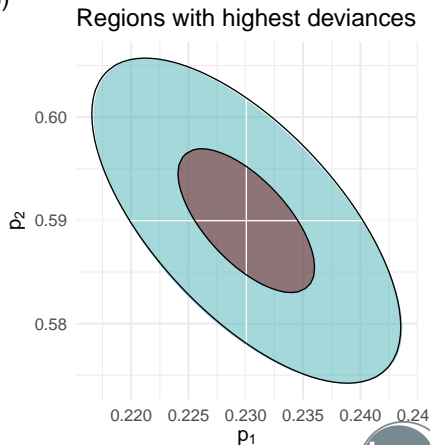
# Deviance contour + quadratic approximation

Goal: **confidence regions** for the MLEs.

(B)



(S)



(B)ig picture and (S)mall picture.



Ok, but I also want some **confidence intervals**.

First, we need some stuff.

Key component: **Fisher information** matrix

$$I_{\text{Expected}}[\mathbf{p}] = \mathbb{E}\{I_{\text{Observed}}[\mathbf{p}]\} = \mathbb{E}\{-H_{\text{essian}}[\mathbf{p}]\} = \begin{bmatrix} \frac{n}{p_1} + \frac{n}{p_3} & \frac{n}{p_3} \\ \frac{n}{p_3} & \frac{n}{p_2} + \frac{n}{p_3} \end{bmatrix},$$

since  $\mathbb{E}\{x_i\} = np_i$ .

Thus, the asymptotic **variance-covariance** matrix is

$$I_{\text{Expected}}[\mathbf{p}]^{-1} = \begin{bmatrix} \frac{(p_2+p_3)(p_1p_2+p_1p_3+p_2p_3)}{np_2p_3} & -\frac{(p_1p_2+p_1p_3+p_2p_3)}{np_3} \\ -\frac{(p_1p_2+p_1p_3+p_2p_3)}{np_3} & \frac{(p_1+p_3)(p_1p_2+p_1p_3+p_2p_3)}{np_1p_3} \end{bmatrix}.$$

And the correlation between the estimates  $\hat{p}_1$  and  $\hat{p}_2$ ,  $\hat{\rho}$ , is given by

$$\hat{\rho}_{\hat{p}_1\hat{p}_2} = -\frac{\sqrt{\hat{p}_1\hat{p}_2}}{\sqrt{\hat{p}_1 + \hat{p}_3}\sqrt{\hat{p}_2 + \hat{p}_3}}.$$



# The intervals,

A 95% Wald interval

$$\hat{p}_1 = 0.23 : \quad \hat{p}_1 + c(\text{qnorm}(0.025), \text{qnorm}(0.975)) I_{\text{Expected}}[\hat{\mathbf{p}}]_{1,1}^{-1/2}$$

```
## [1] 0.1475502 0.3125348
```

$$\hat{p}_2 = 0.59 : \quad \hat{p}_2 + c(\text{qnorm}(0.025), \text{qnorm}(0.975)) I_{\text{Expected}}[\hat{\mathbf{p}}]_{2,2}^{-1/2}$$

```
## [1] 0.4935764 0.6863750
```

---

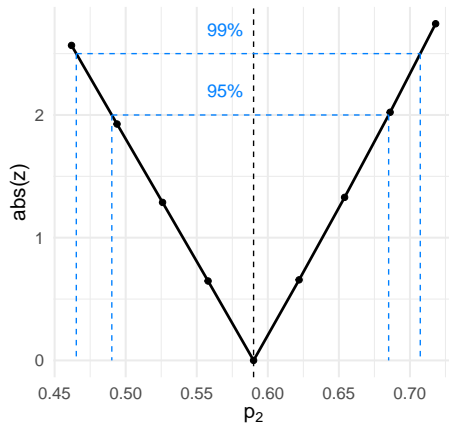
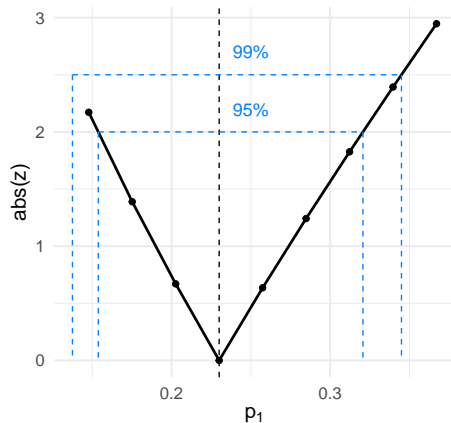
And for  $\hat{p}_3 = g(\hat{\mathbf{p}}) = 1 - \sum_{i=1}^2 \hat{p}_i = 0.18$ , we can get a 95% CI through the multiparameter **Delta Method**

$$g(\hat{\mathbf{p}}) \pm \text{qnorm}(0.975) \sqrt{(\nabla g(\hat{\mathbf{p}}))^{\top} I_{\text{Expected}}[\hat{\mathbf{p}}]^{-1} \nabla g(\hat{\mathbf{p}})}$$

```
## [1] 0.1046884 0.2552752
```



And to finish this part, some **likelihood profiles**



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



# The model

$$l(\beta) = \sum_{j=1}^N \log n_j! - \sum_{j=1}^N \sum_{i=1}^k \log y_{ij}! + \sum_{j=1}^N \sum_{i=1}^k y_{ij} \log p_{ij},$$

$$\text{with } p_{ij} = \frac{\exp\{\mathbf{x}_j^\top \beta_i\}}{1 + \sum_{i=1}^{k-1} \exp\{\mathbf{x}_j^\top \beta_i\}}.$$

- »  $y$  represents the outcome for the subject  $j$ ;
- »  $i$  represents the multinomial category/level;
- »  $p_{ij}$  is the probability of the subject  $j$  be classified in the category  $i$ , the so called **inverse multinomial logit (logistic) function**;
- »  $\beta_i$  is the vector of coefficients for category  $i$ ;
- » and  $\mathbf{x}_j$  is the row vector of covariates of the subject  $j$ .



# Inverse multinomial **logit** function (the **heart** of everything)

```
inv_logit <- function(coefs, preds, data) {  
  ## building objects  
  ind_pred <- seq(preds) ; n_pred <- length(preds) ; k <- n_pred + 1  
  betas <- split(coefs, substr(names(coefs), start = 3, stop = 3))  
  list_X <- lapply(preds, model.matrix, data = data)  
  ## computing the important stuff  
  exp_xb <- sapply(seq(list_X),  
                    function(i) exp(list_X[[i]] %*% betas[[i]]))  
  link_denominator <- 1 + rowSums(exp_xb)  
  ps <- sapply(ind_pred, function(i) exp_xb[ , i]/link_denominator)  
  ps <- cbind(ps, 1 - rowSums(ps))  
  colnames(ps) <- paste0("p", seq(k))  
  return(list(y = data[ , seq(k)], ps = ps))  
}
```





## Now, simulating some probabilities

```
n <- 100 ; k <- 3 # defining sample size and number of categories
naive_data <- function(n, k) {
  data <- as.data.frame(matrix(0, nrow = n, ncol = k))
  names(data) <- paste0("y", 1:k)
  return(data)
}
data <- naive_data(n, k)
set.seed(0080)
data$x1 <- rnorm(n, mean = 5, sd = 1) # covariates
data$x2 <- rnorm(n, mean = 1, sd = .5)
linear_pred <- list(y1 ~ x1 + x2, y2 ~ x1 + x2)
initial_values <- c("b01" = .4, "b11" = .1, "b21" = -.3,
                   "b02" = .2, "b12" = .5, "b22" = -.6)
probs <- inv_logit(initial_values, preds = linear_pred, data = data)$ps
```



After simulating the probabilities based in the covariates, we simulate the **outcomes**

```
summary(probs)
```

```
##           p1           p2           p3
## Min.      :0.05503   Min.      :0.4361   Min.      :0.01842
## 1st Qu.:0.13533   1st Qu.:0.6810   1st Qu.:0.06418
## Median :0.16719   Median :0.7436   Median :0.09363
## Mean     :0.16954   Mean     :0.7323   Mean     :0.09813
## 3rd Qu.:0.20092   3rd Qu.:0.8002   3rd Qu.:0.12448
## Max.     :0.32329   Max.     :0.9247   Max.     :0.24065
```

```
library(mc2d) # vectorized versions of {r, d}multinom
## finally, simulating the multinomial data
set.seed(1101)
data[ , seq(k)] <- rmultinomial(n, 1, prob = probs)
colSums(data[ , seq(k)]) # as expected, close to the ``probs`` means
```

```
## y1 y2 y3
## 17 76 7
```



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



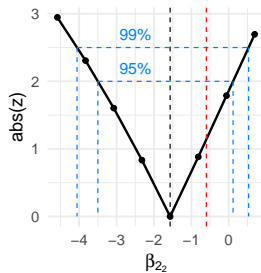
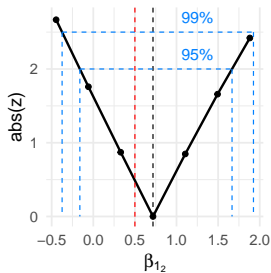
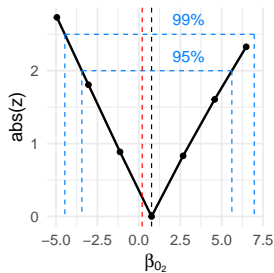
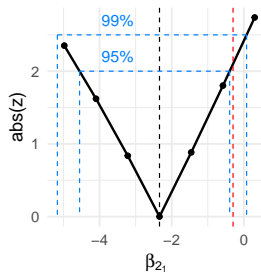
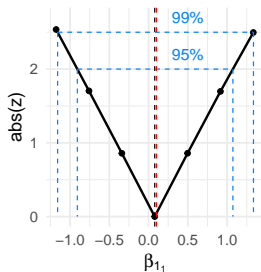
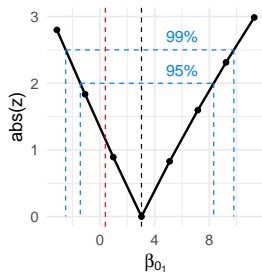
# Maximum Likelihood Estimators

```
multi_lkl <- function(initial_values, linear_pred, data) {  
  ilogit <- inv_logit(initial_values, preds = linear_pred, data = data)  
  lkl_c1 <- with(ilogit, sum(lfactorial(rowSums(y))))  
  lkl_c2 <- with(ilogit, sum(lfactorial(y)))  
  lkl_p <- with(ilogit, sum(y * log(ps)))  
  lkl <- lkl_c1 - lkl_c2 + lkl_p  
  ## lkl <- sum(dmultinomial(as.matrix(ilogit$y), size = 1,  
  ##                          prob = ilogit$ps, log = TRUE))  
  return(-lkl)  
}  
  
parnames(multi_lkl) <- names(initial_values) # mle2 exigency  
model_fit <- mle2(multi_lkl, start = initial_values,  
                  data = list(linear_pred = linear_pred, data = data))  
round(model_fit@coef, 6)
```

##	b01	b11	b21	b02	b12	b22
##	3.040304	0.078060	-2.338833	0.767001	0.717383	-1.568099



# $\beta$ 's profile



# Checking

```
model_fit@coef
```

```
##           b01           b11           b21           b02           b12           b22
## 3.04030427 0.07805955 -2.33883295 0.76700112 0.71738285 -1.56809948
```

```
fit_nnet <- nnet::multinom(y ~ x1 + x2, data_long)
```

```
coef(fit_nnet)
```

```
##      (Intercept)      x1      x2
## y1 3.0412131 0.07785772 -2.338862
## y2 0.7677984 0.71720357 -1.568073
```

```
fit_mlogit <- mlogit::mlogit(y ~ 0 | x1 + x2,
                             refllevel = 3, shape = "wide", data_long)
matrix(fit_mlogit$coef, nrow = 2, ncol = 3)
```

```
##           [,1]      [,2]      [,3]
## [1,] 3.0403108 0.07805888 -2.338837
## [2,] 0.7670852 0.71736537 -1.568101
```



Now, behold the **estimated probabilities**

```
results <- lapply(
  seq(k),
  function(i) cbind(inv_logit(
    model_fit@coef, linear_pred, data)$ps[ , i],
    probs[ , i]))
results <- do.call(cbind, results)
colnames(results) <- c(sapply(seq(k),
  function(i) c(paste0("p", i, "_est"),
    paste0("p", i, "_true"))))
head(round(results, 6), 8)
```

##		p1_est	p1_true	p2_est	p2_true	p3_est	p3_true
##	[1,]	0.044708	0.214565	0.615056	0.607281	0.340236	0.178154
##	[2,]	0.069857	0.142905	0.871300	0.774907	0.058844	0.082188
##	[3,]	0.123282	0.155231	0.830405	0.760373	0.046313	0.084396
##	[4,]	0.182925	0.125786	0.804101	0.819901	0.012974	0.054314
##	[5,]	0.108773	0.162534	0.828098	0.743764	0.063129	0.093702
##	[6,]	0.149994	0.190772	0.764336	0.693429	0.085670	0.115800
##	[7,]	0.229920	0.144030	0.753823	0.791296	0.016257	0.064674
##	[8,]	0.088643	0.200580	0.740493	0.658897	0.170864	0.140523



# Hypothesis tests

Some  $\beta$  is significantly **different from zero**? Testing for  $\beta_{1_1} = 0.078$ .

The simplest way is via the **Wald test**.

```
wald <- function(par, value, alpha) {  
  mle <- as.numeric(model_fit@coef[par])  
  test <- (mle - value)/sqrt(model_fit@vcov[par, par])  
  critic <- qnorm(1 - alpha/2)  
  print(ifelse(test <= critic, "Accept H0", "Reject H0"))  
  return(c("Test" = test, "Critical_value" = critic))  
}  
wald(par = "b11", value = 0, alpha = .05)
```

```
## [1] "Accept H0"
```

```
##           Test Critical_value  
##      0.1606135      1.9599640
```

Other options are the **LRT** and the **Score test**.





# Likelihood Ratio Test (LRT)

```
lrt <- function(model_h0, model_h1, alpha) {  
  test <- 2 * (model_h0@min - model_h1@min)  
  critic <- qchisq(1 - alpha, df = 1)  
  print(ifelse(test <= critic, "Accept H0", "Reject H0"))  
  return(c("Test" = test, "Critical_value" = critic))  
}
```

```
test_values <- c("b01" = .4,          "b21" = - .3,  
                "b02" = .2, "b12" = .5, "b22" = - .6)
```

```
lrt(model_h0 = mle2(multi_lkl2, start = test_values,  
                    data = list(  
                      linear_pred = list(y1 ~ x2, y2 ~ x1 + x2),  
                      data = data, "b11" = 0)),  
    model_h1 = model_fit, alpha = .05)
```

```
## [1] "Accept H0"
```

```
##           Test Critical_value  
## 0.02578236    3.84145882
```



# Score test

```
score <- function(par, ordered_Ie, alpha) {  
  parpos <- which(colnames(ordered_Ie) == par)  
  n <- ncol(ordered_Ie)  
  I12 <- ordered_Ie[parpos, (parpos + 1):n]  
  Vc <- ordered_Ie[parpos, parpos] -  
    I12 %*% solve(ordered_Ie[-parpos, -parpos]) %*% I12  
  test <- as.numeric(U[par] * 1/Vc * U[par])  
  critic <- qchisq(1 - alpha, df = 1)  
  print(ifelse(test <= critic, "Accept H0", "Reject H0"))  
  return(c("Test" = test, "Critical_value" = critic))  
}  
score(par = "b11", ordered_Ie, alpha = .05)
```

```
## [1] "Accept H0"
```

```
##           Test Critical_value  
## 0.02584964      3.84145882
```



# On the Agenda

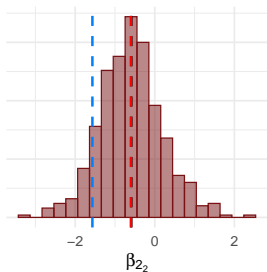
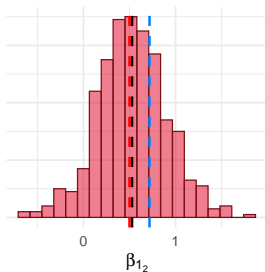
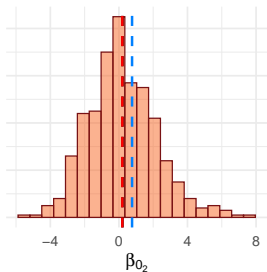
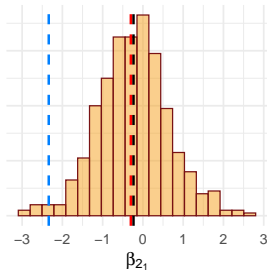
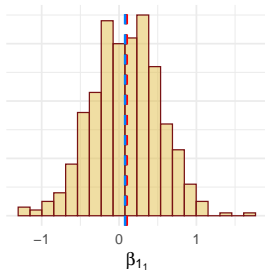
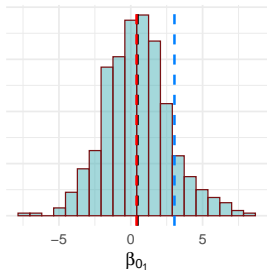
- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



# Running 500 simulations

```
library(furrr)
plan(multiprocess)
parallel2run <- function(nsimu) {
  coefs <- Matrix(0, nrow = nsimu, ncol = 6)
  confint_profile <- confint_quad <- Matrix(0, nrow = nsimu, ncol = 12)
  data[ , seq(k)] <- rmultinomial(n, 1, prob = probs)
  fit <- try(mle2(multi_lkl, start = initial_values,
                 data = list(linear_pred = linear_pred, data = data)))
  if (class(fit) != "try-error") {
    coefs[nsimu, ] <- as.numeric(fit@coef)
    confint_profile[nsimu, ] <- c(confint(fit))
    confint_quad[nsimu, ] <- c(confint(fit, method = "quad"))
  }
  return(list(coefs = coefs, confint_profile = confint_profile,
             confint_quad = confint_quad))
}
nsimu <- 500 ; simu <- vector("list", nsimu)
simu <- future_map(rep(1, nsimu), parallel2run)
```

# MLEs distribution



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



## Coverage rate: Deviance and Wald intervals

```
coverage <- function(method) {  
  coverage <- Matrix(0, nrow = 6, ncol = nsimu)  
  for (i in seq(6))  
    for (j in seq(nsimu))  
      coverage[i, j] <-  
        initial_values[i] >= simu[[j]][[method]][i] &  
        initial_values[i] <= simu[[j]][[method]][i + 6]  
  means <- rowMeans(coverage, na.rm = TRUE)  
  names(means) <- names(initial_values)  
  return(round(means, 3))}  
coverage(method = 2) # deviance interval
```

```
##    b01    b11    b21    b02    b12    b22  
## 0.940 0.958 0.944 0.948 0.942 0.936
```

```
coverage(method = 3) # wald interval
```

```
##    b01    b11    b21    b02    b12    b22  
## 0.952 0.966 0.954 0.950 0.954 0.958
```



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.





For a subject  $i$ , with measurements  $j$ , we have

$$y_{ij} \mid \alpha_{1i}, \alpha_{2i} \sim \text{Multinomial}(p_{1ij}, p_{2ij}, p_{3ij}),$$

$$\text{with } \begin{bmatrix} \alpha_{1i} \\ \alpha_{2i} \end{bmatrix} \sim \mathcal{N} \left[ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_{\alpha_1}^2 & \sigma_{\alpha_1} \sigma_{\alpha_2} \rho \\ \sigma_{\alpha_1} \sigma_{\alpha_2} \rho & \sigma_{\alpha_2}^2 \end{bmatrix} \right],$$

$$\text{and } p_{kij} = \frac{\exp\{\mathbf{x}_{kij}\beta_{kj} + \alpha_{ki}\}}{1 + \sum_{k=1}^{K-1} \exp\{\mathbf{x}_{kij}\beta_{kj} + \alpha_{ki}\}}.$$

The **likelihood** for a random sample, and  $\boldsymbol{\theta} = [\boldsymbol{\beta}, \rho, \sigma_{\alpha_1}^2, \sigma_{\alpha_2}^2]^\top$ , is given by

$$L(\boldsymbol{\theta}; y) = \prod_{i=1}^N \int_{\mathbb{R}} \int_{\mathbb{R}} \prod_{j=1}^{n_i} \left[ \binom{n_{ij}}{y_{1ij}, y_{2ij}, y_{3ij}} \prod_{k=1}^K p_{kij}^{y_{kij}} \right] \times \\ \frac{\exp \left\{ -\frac{1}{2(1-\rho^2)} \left( \frac{\alpha_{1i}^2}{\sigma_{\alpha_1}^2} + \frac{\alpha_{2i}^2}{\sigma_{\alpha_2}^2} - \frac{2\alpha_{1i}\alpha_{2i}\rho}{\sigma_{\alpha_1}\sigma_{\alpha_2}} \right) \right\}}{2\pi\sigma_{\alpha_1}\sigma_{\alpha_2}\sqrt{1-\rho^2}} d\alpha_{1i} d\alpha_{2i}.$$

$$\text{i.e., } L(\boldsymbol{\theta}; y_i) = \int_{\mathbb{R}} \int_{\mathbb{R}} f(y_i \mid \alpha_{1i}, \alpha_{2i}) f(\alpha_{1i}, \alpha_{2i}) d\alpha_{1i} d\alpha_{2i}.$$



# On the Agenda

- 1 Find the MLEs. Provide some confidence intervals.
- 2 Consider two covariates for each subject and propose a regression model.
- 3 Do the maximum likelihood estimation for the proposed model.
- 4 Do a simulation study to evaluate the MLEs distribution.
- 5 Evaluate the coverage rate for the Wald and deviance intervals.
- 6 Propose a mixed model for multiple observations from the same subject.
- 7 Do the maximum likelihood estimation for the proposed model.



## First of all, let's simulate some data

```
data_setup <- function(N, n, k, rho, var1, var2) {  
  gauss_dim <- k - 1 ; off_diag <- rho * sqrt(var1) * sqrt(var2)  
  alpha <- rmvnorm(N, mean = rep(0, gauss_dim),  
                    sigma = matrix(c(var1, rep(off_diag, 2), var2),  
                                   gauss_dim, gauss_dim))  
  
  data <- as.data.frame(  
    matrix(0, nrow = N * n, ncol = 2 + k + gauss_dim))  
  names(data) <- c("i", "j",  
                  paste0("y", seq(k)), paste0("alpha", seq(gauss_dim)))  
  data$i <- rep(seq(N), each = n) ; data$j <- rep(seq(n), N)  
  for (i in seq(gauss_dim)) {  
    data[, paste0("alpha", i)] <- rep(alpha[, i], each = n)}  
  return(data)  
}  
  
N = 200 ; n = 10 ; k = 3 ## subjects, repeated measures and classes  
bigdata <- data_setup(N, n, k, rho = -.3, var1 = .25, var2 = .5)  
linear_pred <- list(y1 ~ 1, y2 ~ 1)  
initial_values <- c("b01" = .85, "b02" = 1.25)  
probs <- inv_logit(initial_values, linear_pred, bigdata)$ps
```

# The data

```
set.seed(955)
bigdata[, paste0("y", seq(k))] <- rmultinomial(N * n, 1, prob = probs)
head(bigdata, 15)
```

##	i	j	y1	y2	y3	alpha1	alpha2
## 1	1	1	0	1	0	-0.8949553	-1.0148634
## 2	1	2	0	1	0	-0.8949553	-1.0148634
## 3	1	3	0	1	0	-0.8949553	-1.0148634
## 4	1	4	0	1	0	-0.8949553	-1.0148634
## 5	1	5	1	0	0	-0.8949553	-1.0148634
## 6	1	6	1	0	0	-0.8949553	-1.0148634
## 7	1	7	0	0	1	-0.8949553	-1.0148634
## 8	1	8	0	1	0	-0.8949553	-1.0148634
## 9	1	9	1	0	0	-0.8949553	-1.0148634
## 10	1	10	1	0	0	-0.8949553	-1.0148634
## 11	2	1	0	0	1	-0.5140890	-0.9003746
## 12	2	2	1	0	0	-0.5140890	-0.9003746
## 13	2	3	1	0	0	-0.5140890	-0.9003746
## 14	2	4	0	0	1	-0.5140890	-0.9003746
## 15	2	5	0	1	0	-0.5140890	-0.9003746



# Integrating into the random effects

```
library(tidyverse)

integrating <- function(alpha, beta, det_sigma, inv_sigma, preds, data) {
  ind_pred <- seq(preds) ; n_pred <- length(preds) ; k <- n_pred + 1
  beta <- split(beta, substr(names(beta), start = 3, stop = 3))
  list_X <- lapply(preds, model.matrix, data = data)
  exp_xb <- sapply(
    seq(list_X),
    function(i) exp(list_X[[i]] %*% beta[[i]] + alpha[i]))
  link_denominator <- 1 + rowSums(exp_xb)
  ps <- sapply(ind_pred, function(i) exp_xb[ , i]/link_denominator)
  ps <- cbind(ps, 1 - rowSums(ps)) ; colnames(ps) <- paste0("p", seq(k))
  y <- as.matrix(data[ , paste0("y", seq(k))]) ; n <- nrow(y)
  out <- sum(dmultinomial(y, size = 1, prob = ps, log = TRUE)) -
    (n/2) * log(2 * pi) -
    .5 * det_sigma - .5 * alpha %*% inv_sigma %*% alpha
  return(out) }
```



Ok, but **how** we integrate that?... **Laplace** approximation!

```
laplace <- function(lkl, dim, ...) {  
  integral <- -6e+05  
  initial <- numeric(dim)  
  logQ <- try(optim(initial, lkl, ..., method = "BFGS",  
                    control = list(fnscale = -1), hessian = TRUE),  
              silent = TRUE)  
  if (class(logQ) != "try-error") {  
    integral <- logQ$value +  
      (dim/2) * log(2*pi) - .5 * determinant(-logQ$hessian)$modulus  
  }  
  return(integral)  
}
```



## Now, we put everything together

```
multi_mixed <- function(theta, data, until) {  
  out <- -sqrt(.Machine$double.xmax)  
  beta <- theta[str_detect(names(theta), pattern = "~b\\d")]  
  rho <- 2 * exp(theta["rho"])/(1 + exp(theta["rho"])) - 1  
  var1 <- exp(theta["var1"]) ; var2 <- exp(theta["var2"])  
  c1 <- 1 - rho^2 ; off_diag <- -rho/(sqrt(var1) * sqrt(var2) * c1)  
  inv_sigma <-  
    matrix(c(1/(var1 * c1), rep(off_diag, 2), 1/(var2 * c1)), 2, 2)  
  det_sigma <- log(var1 * var2 * c1)[[1]]  
  out <- future_map_dbl(  
    seq(until),  
    function(index) {  
      laplace(lkl = integrating, dim = 2, beta = beta,  
              det_sigma = det_sigma, inv_sigma = inv_sigma,  
              preds = linear_pred,  
              data = data[data$i == index, ]) })  
  return(-sum(out)) }  
}
```



Let's see if all this stuff that I did really works

```
plan(multiprocess)
tic() ; mle2_fit <- mle2(
  multi_mixed, start = theta, method = "BFGS",
  data = list(data = bigdata[ , seq(5)], until = N)) ; toc()
```

```
## 649.112 sec elapsed
```

```
summary(mle2_fit)@coef
```

##		Estimate	Std. Error	z value	Pr(z)
##	b01	0.8479660	0.08598689	9.8615728	6.108492e-23
##	b02	1.3733239	0.09347495	14.6918926	7.265576e-49
##	rho	0.6232604	0.99615488	0.6256661	5.315340e-01
##	var1	-2.8755136	1.78410385	-1.6117412	1.070183e-01
##	var2	-0.6797736	0.31460515	-2.1607198	3.071699e-02





# Compare!

```
thetasss <- theta ; mle2_est <- mle2_fit@coef

mle2_est["rho"] <- 2 * exp(mle2_est["rho"])/(1 + exp(mle2_est["rho"])) - 1
thetasss[c("var1", "var2")] <- exp(thetasss[c("var1", "var2")])
mle2_est[c("var1", "var2")] <- exp(mle2_est[c("var1", "var2")])

rbind(thetasss, mle2_est)
```

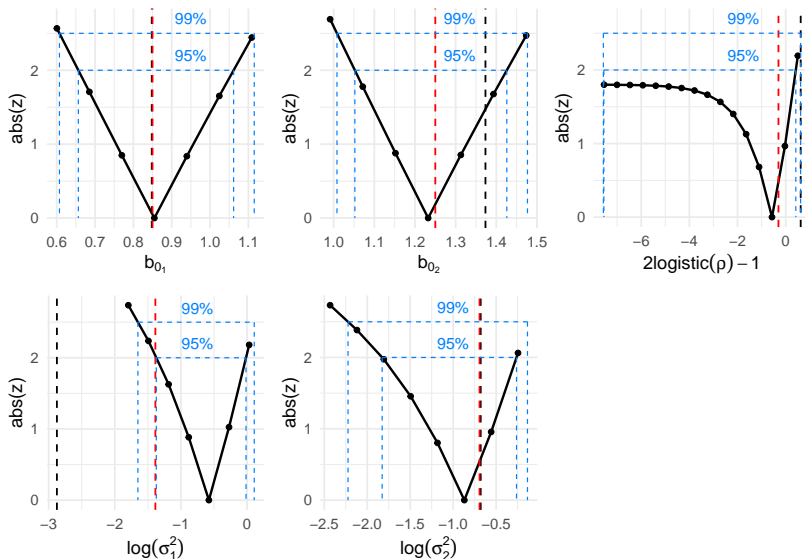
```
##           b01      b02      rho      var1      var2
## thetasss 1.000000 1.500000 0.3000000 0.2500000 0.7500000
## mle2_est 0.847966 1.373324 0.3019194 0.05638717 0.5067317
```

```
## standard errors
c(sqrt(diag(mle2_fit@vcov[1:2, 1:2])), delta_method)
```

```
##           b01      b02      rho      var1      var2
## 0.08598689 0.09347495 0.45267503 0.10060057 0.15942041
```



# Profiling (and waiting 21402.816 sec)



Black dotted: mle; red dotted: true parameter value.



# A deeper look...

```
(optimals_matrix <- rbind(optimals_matrix, mle2_fit@coef))
```

```
##           b01           b02           rho           var1           var2
## [1,] 0.854774 1.232459 -0.5678167 -0.5766362 -0.8699930
## [2,] 0.854774 1.232459 -0.5678167 -0.5766362 -0.8699930
## [3,] 0.854774 1.232459 -0.5678167 -0.5766362 -0.8699930
## [4,] 0.854774 1.232459 -0.5678167 -0.5766362 -0.8699930
## [5,] 0.854774 1.232459 -0.5678167 -0.5766362 -0.8699930
## [6,] 0.847966 1.373324 0.6232604 -2.8755136 -0.6797736
```

```
tic() ; sapply(seq(nrow(optimals_matrix)),
               function(i)
                 multi_mixed(optimals_matrix[i, ],
                             data = bigdata[ , seq(5)], until = N)
               ) ; toc()
```

```
## [1] 3425.798 3425.798 3425.798 3425.798 3425.798 3425.403
```

```
## 47.965 sec elapsed
```



# Next steps for Henrique of the near future

- » See if I'm doing something wrong and consequently generating this [multimodality](#);
- » Integrate the  $\beta$ 's together with the random effects in the Laplace approximation;
- » See how to get back the integrated  $\beta$ 's, hahaha
- » Change the `optim` routine inside the Laplace approx. by a [Newton-Raphson](#) algorithm;
- » Check which parts of the code I can do in a faster way;
- » More covariates and [time-dependent](#) covariates ( $\beta$  varying on time);
- » Insert more [dependency structures](#);
- » Swim in [Bayesian](#) waters;
- » Whatever else [Wagner](#) asks for hahaha
- » ...



and...

