Lista 3, Verossimilhança (parte II)

Henrique Ap. Laureano

laureano@ufpr.br ∧ www.leg.ufpr.br/~henrique/

/UFPR/DEST/LEG/

October 25, 2019

```
# packages ====================================================================
library(bbmle)
library(ggplot2)
library(gridExtra)
library(fishualize)
library(mvtnorm)
```

Contents

Exercise 1: dist. Gaussiana com dados censurados/intervalares

Foram tomadas as seguintes observações independentes de uma v.a. $X \sim \text{Normal}(\mu, \sigma^2)$.

- 56.4, 54.2, 65.3, 49.2, 50.1, 56.9, 58.9, 62.5, 70, 61.
- Sabe-se que outras 5 observações são menores que 50.
- Sabe-se que outras 3 observações são maiores que 65.

(a) Escrever a função de verossimilhança.

The likelihood can be expressed as

$$
\begin{aligned}
L(\theta) &= \left(\prod_{i=1}^{10} f(x_i)\right) \times (F(50))^5 \times (1 - F(65))^3 \\
&= \left(\prod_{i=1}^{10} \phi\left(\frac{y_i - \mu}{\sigma}\right)\right) \times \left(\Phi\left(\frac{50 - \mu}{\sigma}\right)\right)^5 \times \left(1 - \Phi\left(\frac{65 - \mu}{\sigma}\right)\right)^3 \\
l(\theta) &= \sum_{i=1}^{10} \log \phi\left(\frac{y_i - \mu}{\sigma}\right) + 5 \log \Phi\left(\frac{50 - \mu}{\sigma}\right) + 3 \log\left(1 - \Phi\left(\frac{65 - \mu}{\sigma}\right)\right).
\end{aligned}
$$

<r code>

```r
data_1 <- c(56.4, 54.2, 65.3, 49.2, 50.1, 56.9, 58.9, 62.5, 70, 61)
intervAussian_lkl <- function(mu, sigma) {
    component_1 <- sum(dnorm(data_1, mean = mu, sd = sigma, log = TRUE))
    component_2 <- 5 * log(pnorm(50, mean = mu, sd = sigma))
    component_3 <- 3 * log(1 - pnorm(65, mean = mu, sd = sigma))
    lkl <- component_1 + component_2 + component_3
    return(-lkl) ## minus, since the optim routine performs a minimization
}
```

(b) Obter as estimativas de máxima verossimilhança.

<r code>

```r
model_fit <- mle2(intervAussian_lkl,
                  start = list("mu" = mean(data_1), "sigma" = sd(data_1)))
(par_est <- model_fit@coef)
```

```
      mu    sigma
56.33051 11.13274
```

FYI: In the ten points sample we have $\bar{x} = 58.45$ and $s^2 = 6.527$.

(c) Obter as verossimilhanças perfilhadas.

To obtain the likelihood profile for μ we need to create a grid of μ's and for each value of this grid find the value, $\hat{\sigma}_\mu$, that maximizes the profile likelihood. We do this for μ and σ, the results can be seen in Figure 1.

<r code>

```
intervAussian_profile <- function(grid, fun, par_est, xlab, main) {
    ggplot() +
        geom_line(aes(x = grid, y = sapply(grid, fun)), size = 1.5) +
        geom_vline(xintercept = par_est, linetype = "dashed") +
        labs(x = xlab, y = NULL, title = main) +
        theme_minimal()
}
grid.arrange(
    intervAussian_profile(
        grid = seq(32, 77, .5),
        fun = function(i) {
            logLik(mle2(intervAussian_lkl,
                        start = list("sigma" = sd(data_1)),
                        data = list("mu" =  i)))
        }, par_est = par_est[1], xlab = expression(mu),
        main = expression(
            "logLik for the optimal" ~ sigma ~ "given a fixed" ~ mu)),
    intervAussian_profile(
        grid = seq(5, 50, .5),
        fun = function(i) {
            logLik(mle2(intervAussian_lkl,
                        start = list("mu" = mean(data_1)),
                        data = list("sigma" =  i)))
        }, par_est = par_est[2], xlab = expression(sigma),
        main = expression(
            "logLik for the optimal" ~ mu ~ "given a fixed" ~ sigma)),
    ncol = 2)
```

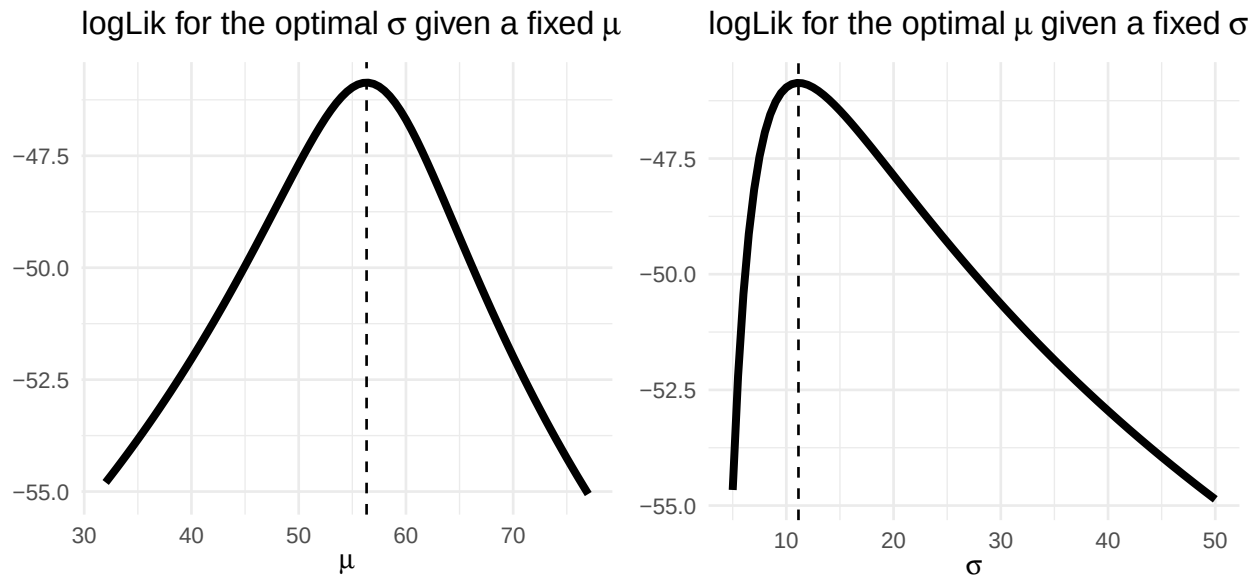

Figure 1: Profile log-likelihoods for μ and σ in a Gaussian sample made of interval/censored data. In dashed, the respective MLE's.

(d) Obter os erro-padrão das estimativas.

<r code>

```
sqrt(diag(model_fit@vcov))
```

```
      mu    sigma
2.794350 2.791622
```

(e) Obter a aproximação quadrática da verossimilhança.

<r code>

```
intervAussian_deviance <- function(mu_grid, sigma_grid) {
    lkl_grid <- intervAussian_lkl(mu_grid, sigma_grid)
    lkl_est <- intervAussian_lkl(par_est[1], par_est[2])
    return(2 * (lkl_grid - lkl_est))
}
intervAussian_deviance <- Vectorize(intervAussian_deviance,
                                    c("mu_grid", "sigma_grid"))
intervAussian_taylor <- function(mu, sigma) {
    par_grid <- c(mu, sigma)
```

```
    half <- (par_grid - model_fit@coef)
    Ie <- solve(model_fit@vcov)
    taylor <- logLik(model_fit) - .5 * half %*% Ie %*% half
    return(taylor)
}
taylor_deviance <- function(mu_grid, sigma_grid) {
    taylor_grid <- intervAussian_taylor(mu_grid, sigma_grid)
    taylor_est <- intervAussian_taylor(par_est[1], par_est[2])
    return(-2 * (taylor_grid - taylor_est))
}
taylor_deviance <- Vectorize(taylor_deviance, c("mu_grid", "sigma_grid"))
mu_grid <- seq(40, 70, length.out = 50)
sigma_grid <- seq(0, 30, length.out = 50)
deviance_grid <- outer(mu_grid, sigma_grid, FUN = intervAussian_deviance)
taylor_grid <- outer(mu_grid, sigma_grid, FUN = taylor_deviance)

rownames(deviance_grid) <- rownames(taylor_grid) <- mu_grid
colnames(deviance_grid) <- colnames(taylor_grid) <- sigma_grid

deviance_grid <- reshape2::melt(deviance_grid)
taylor_grid <- reshape2::melt(taylor_grid)
contour_levels <- c(.99, .95, .9, .8, .7, .5, .3, .1, .05, .01)
ggplot() +
    theme_minimal() +
    stat_contour(data = deviance_grid,
                 aes(Var1, Var2, z = value, fill = ..level..),
                 geom = "polygon",
                 breaks = qchisq(contour_levels, df = 2), alpha = .5) +
    stat_contour(data = taylor_grid,
                 aes(Var1, Var2, z = value, fill = ..level..),
                 geom = "polygon",
                 breaks = qchisq(contour_levels, df = 2), alpha = .25) +
    guides(fill = "none") +
    scale_fill_fish(option = "Trimma_lantana", direction = -1) +
    labs(x = expression(mu), y = expression(sigma),
         title = "Deviance contours", subtitle = "MLE's, in white.") +
    geom_vline(xintercept = par_est[1], col = "white") +
    geom_hline(yintercept = par_est[2], col = "white")
```

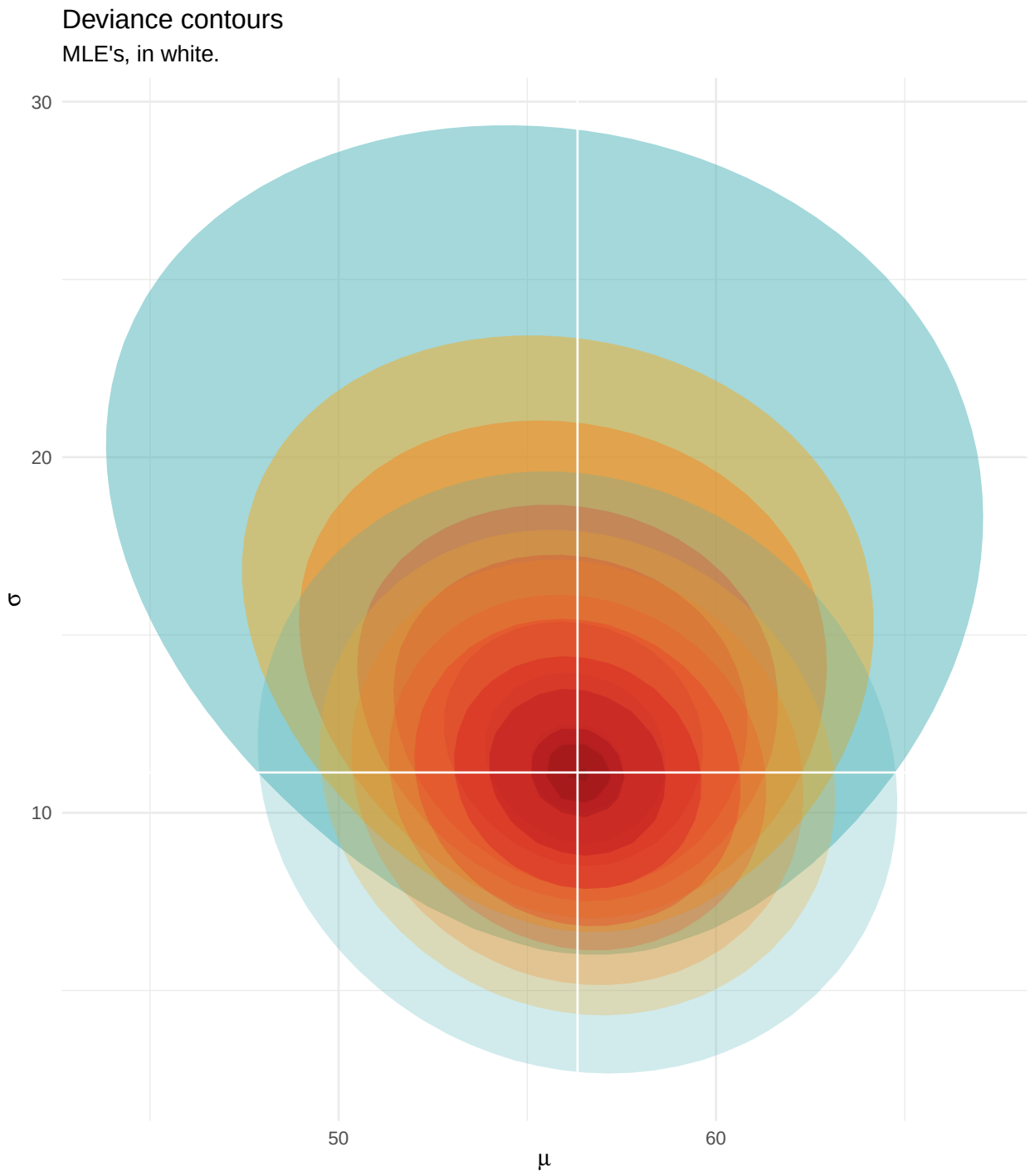

Figure 2: Deviances for the censored/interval Gaussian data. The bigger deviance is the true one, based in the log-likelihood. The smaller one, well-behaved, is based in the quadratic approximation (2nd order taylor approximation) of the log-likelihood.

(f) Obter intervalos de confiança (verossimilhança e Wald) para os parâmetros.

In Figure 3 we see that the regions obtained from the two approaches are quite similar.

<r code>

```r
mu_grid <- seq(55, 57.5, length.out = 50)
sigma_grid <- seq(10, 12.5, length.out = 50)

deviance_grid <- outer(mu_grid, sigma_grid, FUN = intervAussian_deviance)
taylor_grid <- outer(mu_grid, sigma_grid, FUN = taylor_deviance)

rownames(deviance_grid) <- rownames(taylor_grid) <- mu_grid
colnames(deviance_grid) <- colnames(taylor_grid) <- sigma_grid

deviance_grid <- reshape2::melt(deviance_grid)
taylor_grid <- reshape2::melt(taylor_grid)

contour_levels <- c(.95, .05, .01)
ggplot() +
    theme_minimal() +
    stat_contour(data = deviance_grid,
                 aes(Var1, Var2, z = value, fill = ..level..),
                 geom = "polygon",
                 breaks = qchisq(contour_levels, df = 2),
                 alpha = .5) +
    stat_contour(data = taylor_grid,
                 aes(Var1, Var2, z = value, fill = ..level..),
                 geom = "polygon",
                 breaks = qchisq(contour_levels, df = 2),
                 alpha = .25) +
    guides(fill = "none") +
    scale_fill_fish(option = "Cirrhilabrus_solorensis") +
    labs(x = expression(mu), y = expression(sigma),
         title = "99 and 95% deviance contours",
         subtitle = "MLE's, in white.") +
    geom_vline(xintercept = par_est[1], col = "white") +
    geom_hline(yintercept = par_est[2], col = "white")
```

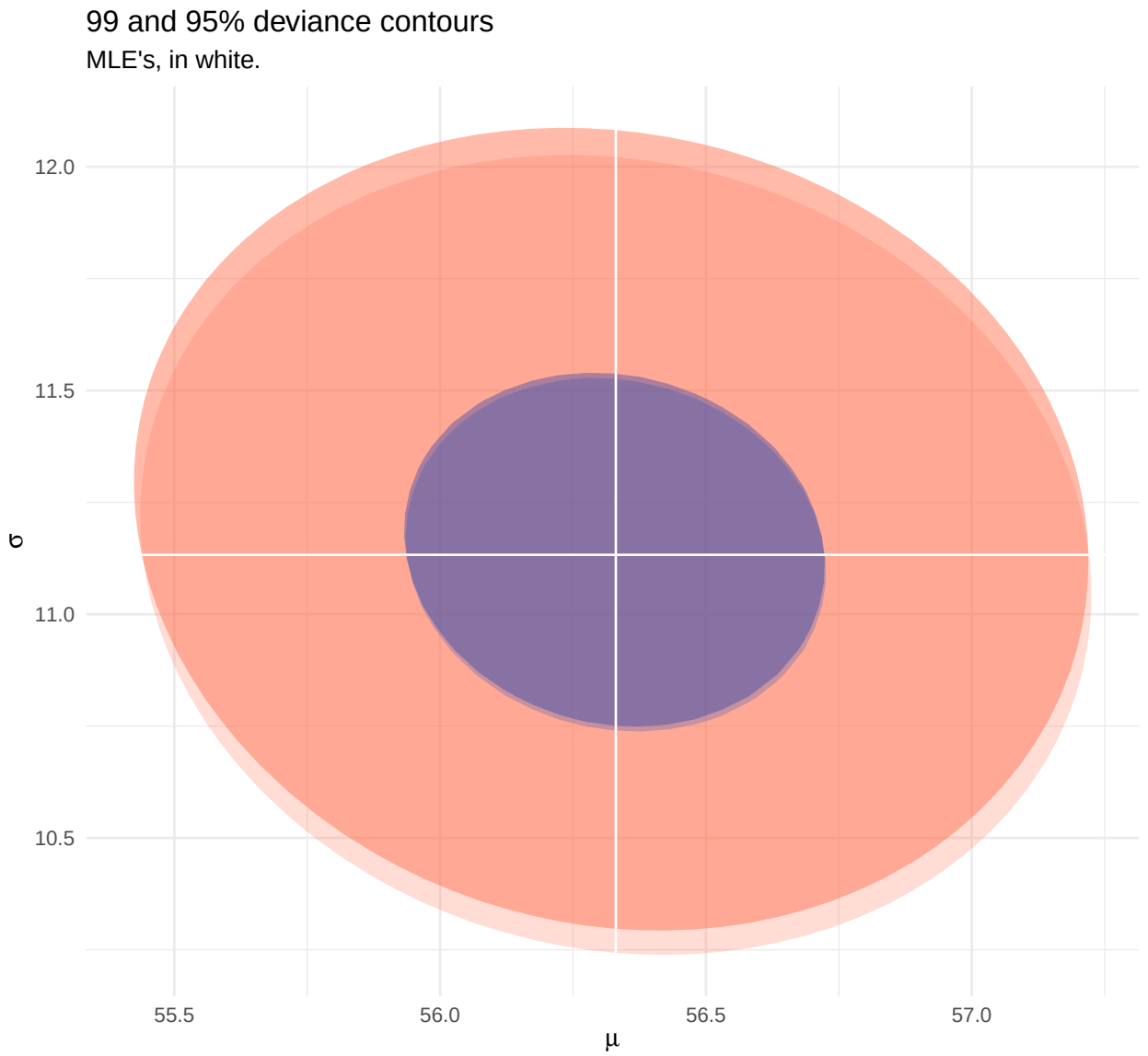

Figure 3: 99 and 95% confidence regions based on cut in the deviances. The stronger one is the likelihood-based, the other is the quadratic approximation (a.k.a., Wald based-interval).

Exercise 2: dist. Gaussiana bivariada

Seja $(x_i, y_i), \ldots, (x_n, y_n)$ uma amostra aleatória de uma distribuição normal bivariada com média zero e matriz de covariância:

$$\Sigma = \sigma^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

(a) Mostrar que a matriz de informação de Fisher é dada por

$$\begin{bmatrix} \frac{n}{\sigma^4} & -\frac{n\rho}{\sigma^2(1-\rho^2)} \\ -\frac{n\rho}{\sigma^2(1-\rho^2)} & \frac{n(1+\rho^2)}{1-\rho^2} \end{bmatrix}.$$

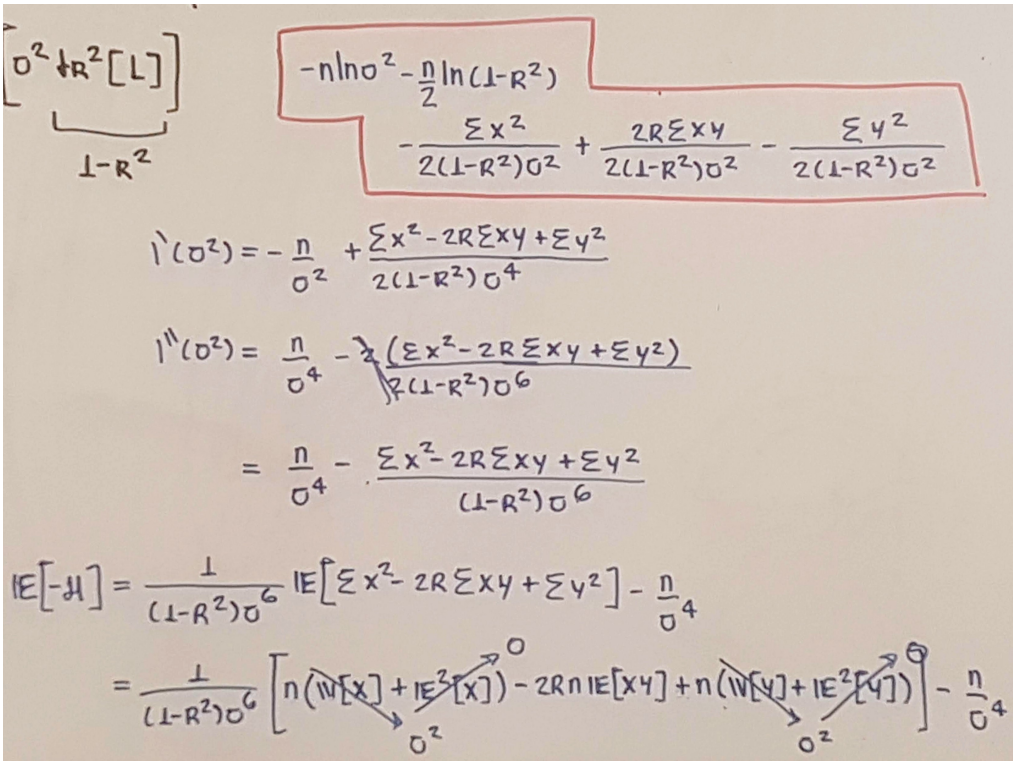

Figure 4: Verossimilhança, já em log, apenas com os componentes relevantes. Derivadas em relação a σ^2 e início do cálculo de seu componente na Informação de Fisher.

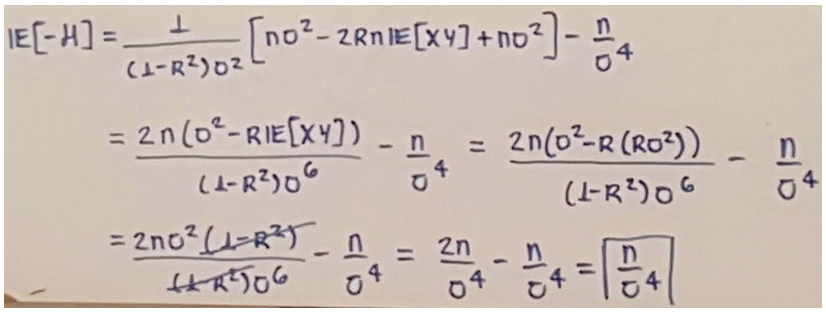

Figure 5: Componente 11 da Informação de Fisher de ordem 2x2.

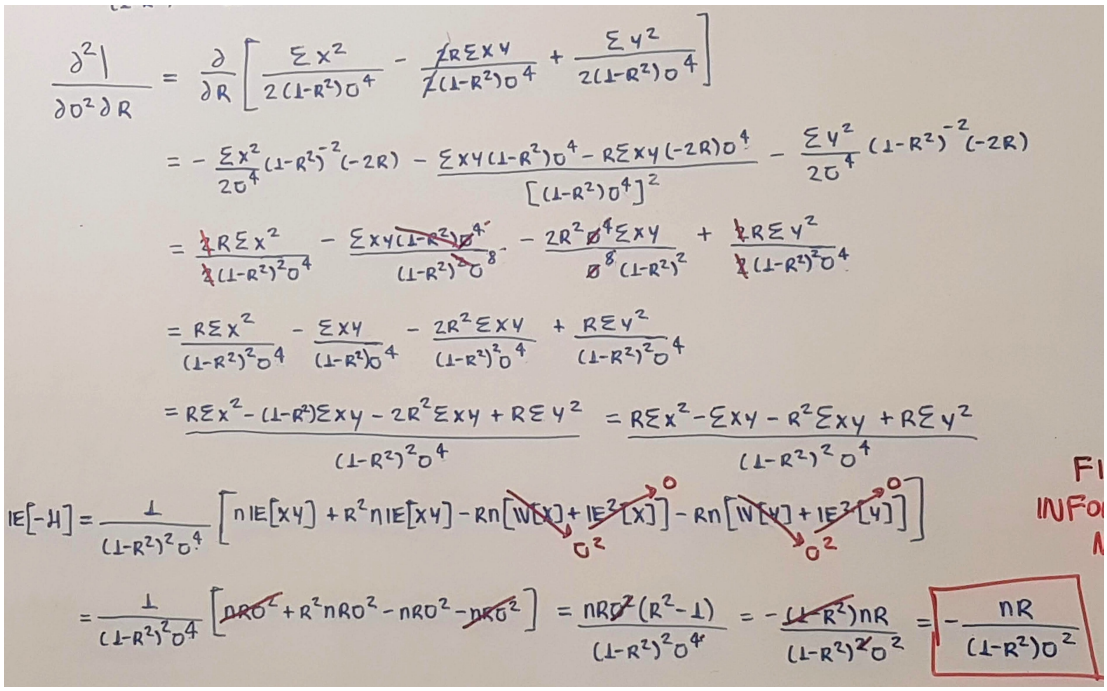

Figure 6: Componente *off-diagonal* da Informação de Fisher de ordem 2x2.

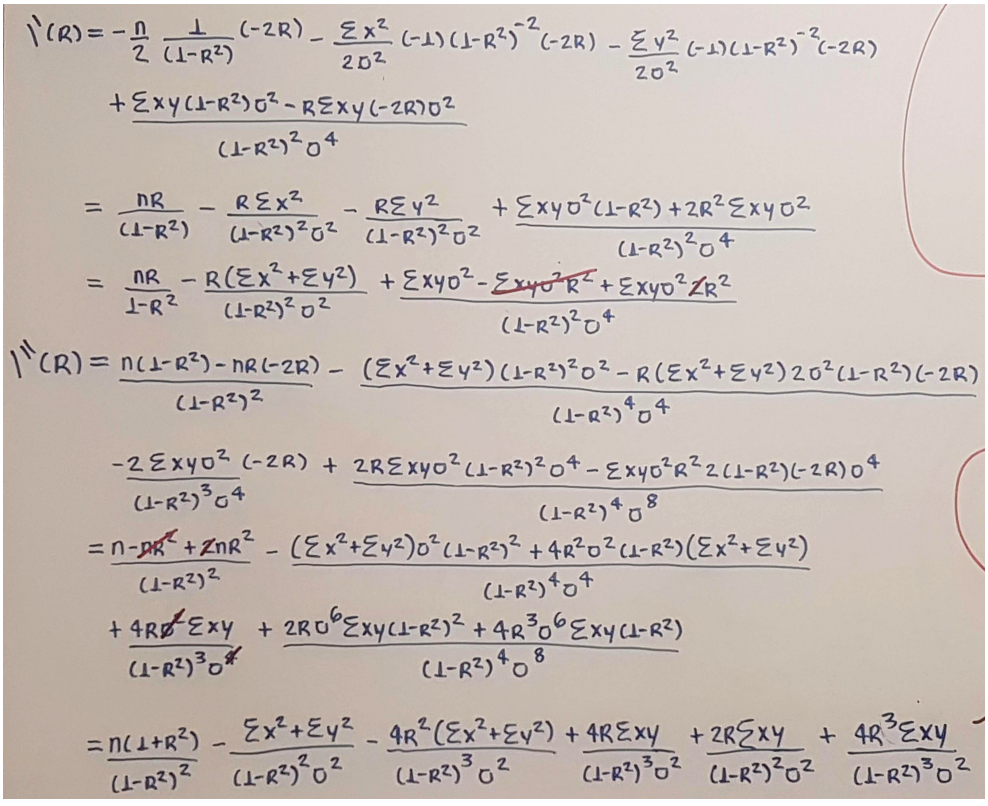

Figure 7: Derivadas em relação a ρ.

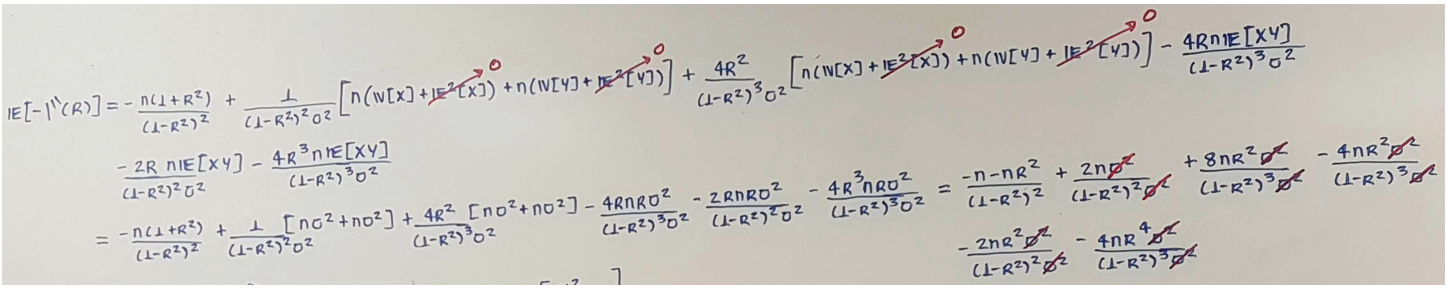

Figure 8: Início do cálculo do componente 22, em relação a ρ, da Informação de Fisher de ordem 2x2.

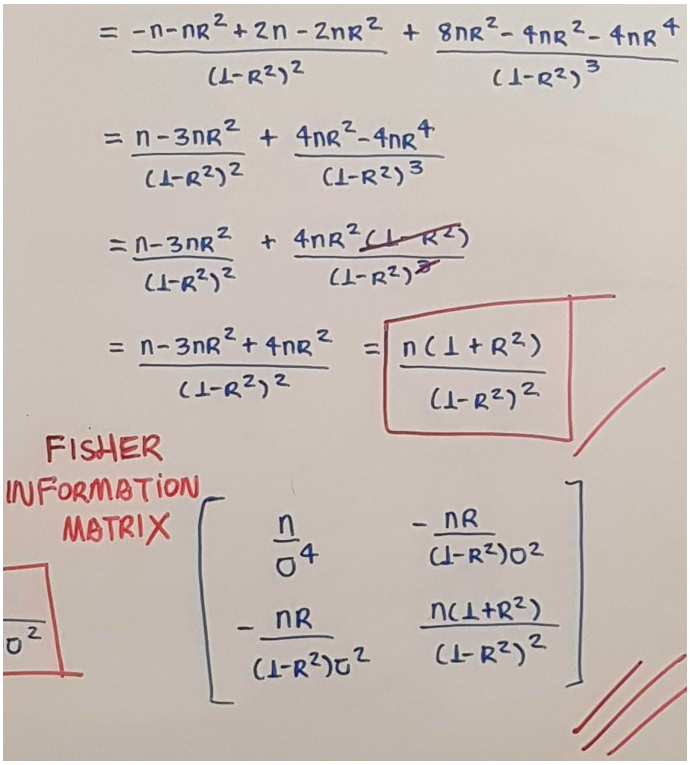

Figure 9: Componente 22 e matriz, completa, de Informação de Fisher.

(b) Considere os dados a seguir assumindo que a média é conhecida no valor observado de forma que os valores podem ser centrados. Obtenha a verossimilhança conjunta de (σ^2, ρ). Compare I_{22} com I_{22}^{-1} e comente os resultados.

<r code>
```r
data_2 <- data.frame(x = c(109, 96, 109, 114, 85, 113, 107, 101,
                           88, 96, 116, 96, 100, 117, 104, 81),
                     y = c(116, 95, 113, 109, 91, 115, 100, 95,
                           77, 79, 122, 94, 88, 119, 115, 90))
```

<r code>
```r
bigaussian <- function(sigma2, rho, z = FALSE,
                       logsigma2 = FALSE, logisticrho = FALSE) {
    if (z) rho <- (exp(2 * rho) - 1)/(1 + exp(2 * rho))
    if (logsigma2) sigma2 <- exp(sigma2)
    if (logisticrho) rho <- qlogis(rho)
    lkl <- sum(dmvnorm(data_2,
                       mean = apply(data_2, 2, mean),
                       sigma = sigma2 * matrix(c(1, rho, rho, 1), 2),
                       log = TRUE))
    return(-lkl)
}
model_fit <- mle2(bigaussian, start = list("sigma2" = 150, "rho" = .5))
(par_est <- model_fit@coef)
```

```
    sigma2        rho
150.008070   0.792046
```

```r
bigaussian_deviance <- function(sigma2_grid, rho_grid, z = FALSE,
                                logsigma2 = FALSE, logisticrho = FALSE) {
    lkl_grid <- bigaussian(sigma2_grid, rho_grid, z, logsigma2, logisticrho)
    lkl_est <- bigaussian(par_est[1], par_est[2], z, logsigma2, logisticrho)
    return(2 * (lkl_grid - lkl_est))
}
bigaussian_deviance <- Vectorize(bigaussian_deviance,
                                 c("sigma2_grid", "rho_grid"))

sigma2_grid <- seq(70, 535, length.out = 50)
rho_grid <- seq(.25, .95, length.out = 50)

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance)
rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid
```

```r
deviance_grid <- reshape2::melt(deviance_grid)

contour_levels <- c(.99, .95, .9, .8, .7, .5, .3, .1, .05, .01)
ggplot() +
    theme_minimal() +
    stat_contour(data = deviance_grid,
                 aes(Var1, Var2, z = value, fill = ..level..),
                 geom = "polygon",
                 breaks = qchisq(contour_levels, df = 2)) +
    guides(fill = "none") +
    scale_fill_fish(option = "Trimma_lantana", direction = -1) +
    labs(x = expression(sigma^2), y = expression(rho),
         title = "Deviance contour", subtitle = "MLE's, in white.") +
    geom_vline(xintercept = par_est[1], col = "white") +
    geom_hline(yintercept = par_est[2], col = "white")
```

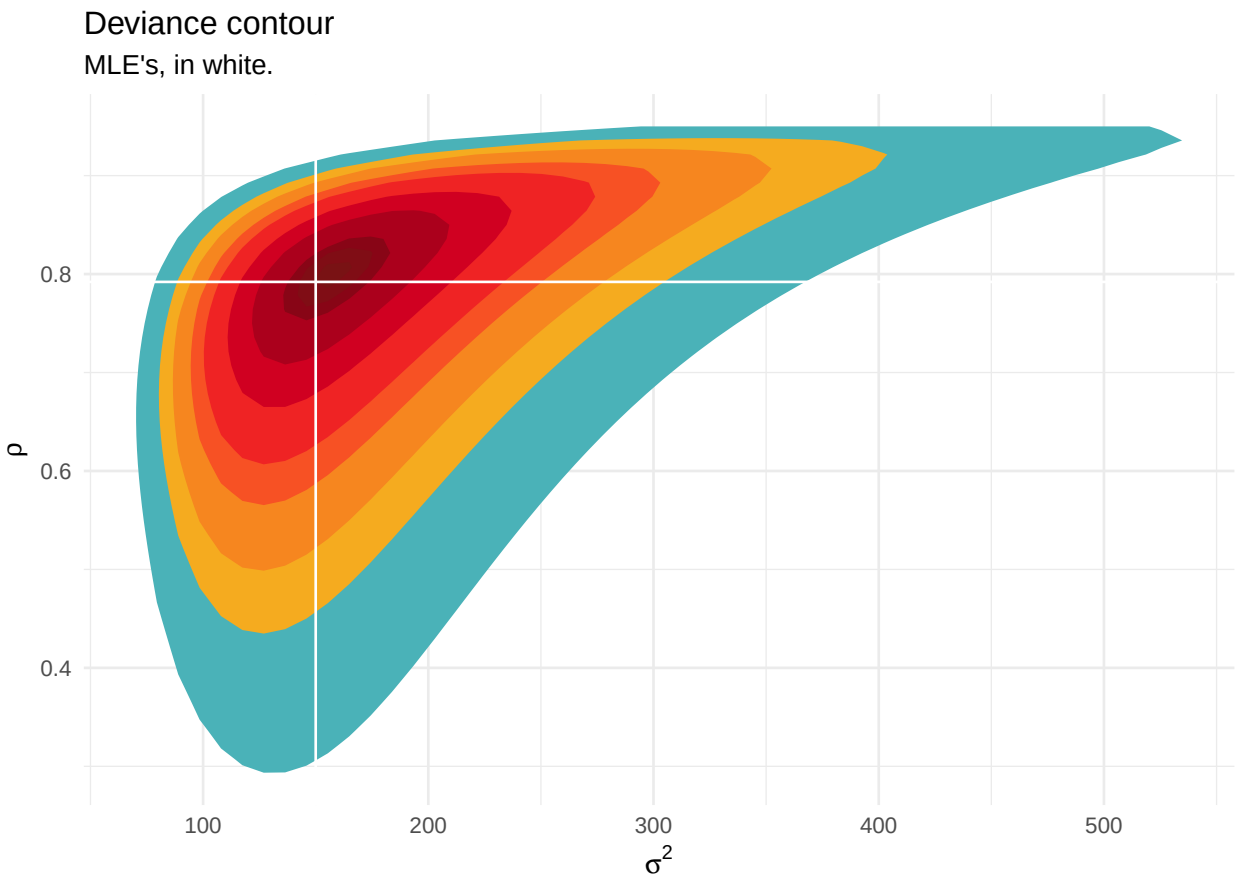

Figure 10: Deviance contour of a bivariate Gaussian of zero mean.

(c) Investigue como a aproximação quadrática se comporta neste problema. Tente transformações de ρ e σ^2 para obter uma função de verossimilhança mais regular.

<r code>

```r
bigaussian_taylor <- function(sigma2, rho) {
    par_grid <- c(sigma2, rho)
    half <- (par_grid - par_est)
    Ie <- solve(model_fit@vcov)
    taylor <- logLik(model_fit) - .5 * half %*% Ie %*% half
    return(taylor)
}
taylor_deviance  <- function(sigma2_grid, rho_grid) {
    taylor_grid <- bigaussian_taylor(sigma2_grid, rho_grid)
    taylor_est <- bigaussian_taylor(par_est[1], par_est[2])
    return(-2 * (taylor_grid - taylor_est))
}
taylor_deviance <- Vectorize(taylor_deviance, c("sigma2_grid", "rho_grid"))
sigma2_grid <- seq(10, 290, length.out = 50)
rho_grid <- seq(.5, 1.1, length.out = 50)
taylor_grid <- outer(sigma2_grid, rho_grid, FUN = taylor_deviance)
rownames(taylor_grid) <- sigma2_grid
colnames(taylor_grid) <- rho_grid
taylor_grid <- reshape2::melt(taylor_grid)
grid.arrange(
    ggplot() +
    theme_minimal() +
    stat_contour(data = taylor_grid, geom = "polygon",
                 aes(Var1, Var2, z = value, fill = ..level..),
                 breaks = qchisq(contour_levels, df = 2)) +
    guides(fill = "none") +
    scale_fill_fish(option = "Trimma_lantana", direction = -1) +
    labs(x = expression(sigma^2), y = expression(rho),
         title = "Deviance of the quadratic approx.",
         subtitle = "MLE's, in white.", tag = "A") +
    geom_vline(xintercept = par_est[1], col = "white") +
    geom_hline(yintercept = par_est[2], col = "white"),
    ggplot() +
```

```
theme_minimal() +
stat_contour(data = deviance_grid, geom = "polygon",
             aes(Var1, Var2, z = value, fill = ..level..),
             breaks = qchisq(contour_levels, df = 2), alpha = .5) +
stat_contour(data = taylor_grid, geom = "polygon",
             aes(Var1, Var2, z = value, fill = ..level..),
             breaks = qchisq(contour_levels, df = 2), alpha = .75) +
guides(fill = "none") +
scale_fill_fish(option = "Trimma_lantana", direction = -1) +
labs(x = expression(sigma^2), y = expression(rho),
     title = "Deviance contours",
     subtitle = "MLE's, in white.", tag = "B") +
geom_vline(xintercept = par_est[1], col = "white") +
geom_hline(yintercept = par_est[2], col = "white"), ncol = 2)
```

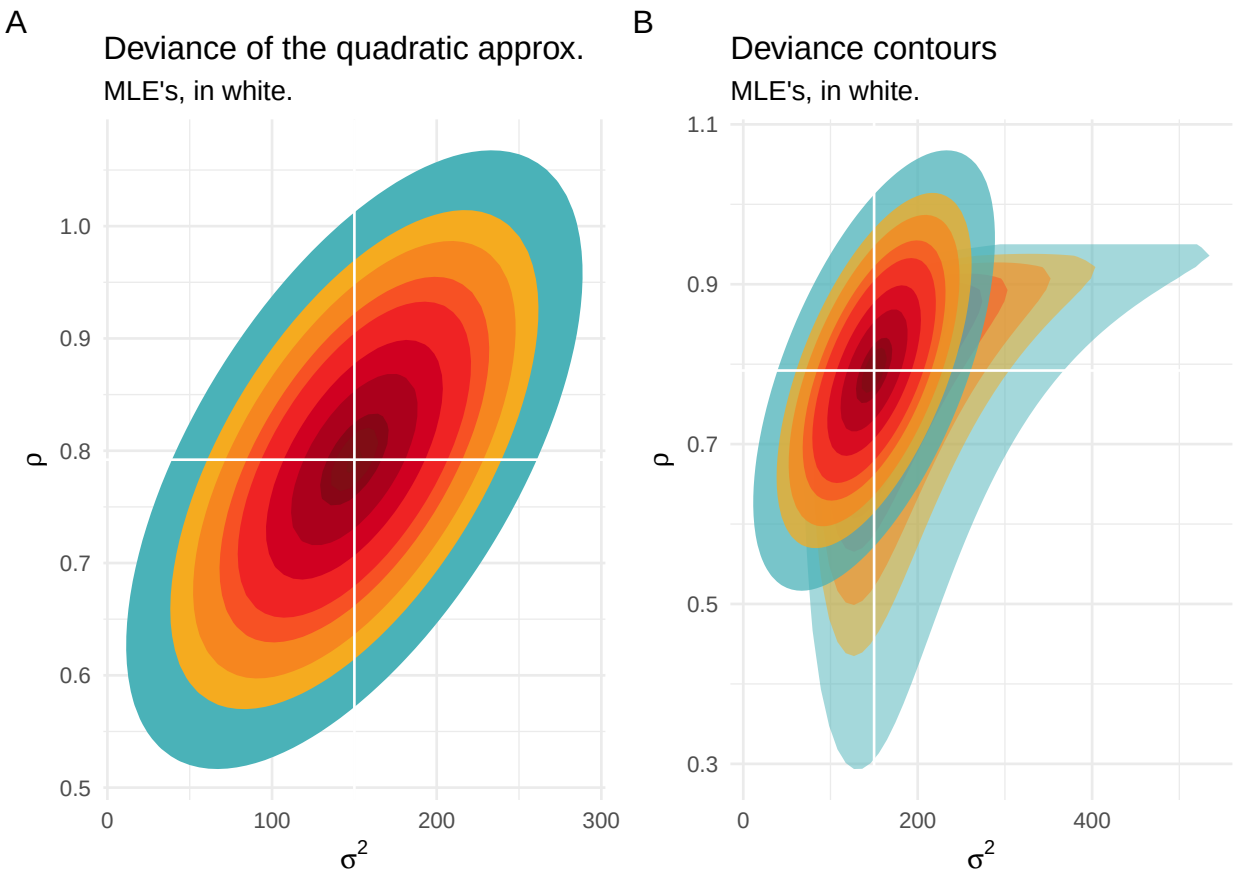

Figure 11: A: Deviance contour of the quadritic approximation for a bivariate Gaussian with zero mean; B: Quadratic approx. added into the original likelihood-based deviance contour.

<r code>

```r
bigaussian_trans <- function(dataset, xlab, ylab, tag, par_est) {
    ggplot() +
        theme_minimal() +
        stat_contour(data = dataset,
                     aes(Var1, Var2, z = value, fill = ..level..),
                     geom = "polygon",
                     breaks = qchisq(contour_levels, df = 2)) +
        guides(fill = "none") +
        scale_fill_fish(option = "Trimma_lantana", direction = -1) +
        labs(x = xlab, y = ylab, tag = tag) +
        geom_vline(xintercept = par_est[1], col = "white") +
        geom_hline(yintercept = par_est[2], col = "white")
}
## transforming only sigma2 ================================================
par_est <- mle2(bigaussian,
                start = list("sigma2" = log(150), "rho" = .5),
                data = list(logsigma2 = TRUE))@coef

sigma2_grid <- seq(log(70), log(535), length.out = 50)
rho_grid <- seq(.25, .95, length.out = 50)

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance,
                       logsigma = TRUE)

rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid

deviance_grid <- reshape2::melt(deviance_grid)

p1 <- bigaussian_trans(dataset = deviance_grid,
                       xlab = expression(log(sigma^2)),
                       ylab = expression(rho),
                       tag = "A", par_est = par_est)
## transforming only rho ===================================================
par_est <- mle2(bigaussian,
                start = list("sigma2" = 150, "rho" = plogis(.5)),
                data = list(logisticrho = TRUE))@coef
```

```r
sigma2_grid <- seq(70, 535, length.out = 50)
rho_grid <- seq(plogis(.25), plogis(.95), length.out = 50)

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance,
                       logisticrho = TRUE)

rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid

deviance_grid <- reshape2::melt(deviance_grid)

p2 <- bigaussian_trans(dataset = deviance_grid,
                       xlab = expression(sigma^2),
                       ylab = expression(logistic(rho)),
                       tag = "B", par_est = par_est)
## transforming both parameters ========================================
par_est <- mle2(bigaussian,
                start = list("sigma2" = log(150), "rho" = plogis(.5)),
                data = list(logsigma2 = TRUE, logisticrho = TRUE))@coef

sigma2_grid <- seq(log(70), log(535), length.out = 50)
rho_grid <- seq(plogis(.25), plogis(.95), length.out = 50)

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance,
                       logsigma2 = TRUE, logisticrho = TRUE)

rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid

deviance_grid <- reshape2::melt(deviance_grid)

p3 <- bigaussian_trans(dataset = deviance_grid,
                       xlab = expression(log(sigma^2)),
                       ylab = expression(logistic(rho)),
                       tag = "C", par_est = par_est)
## plotting everything =================================================
grid.arrange(p1, p2, p3, ncol = 3)
```

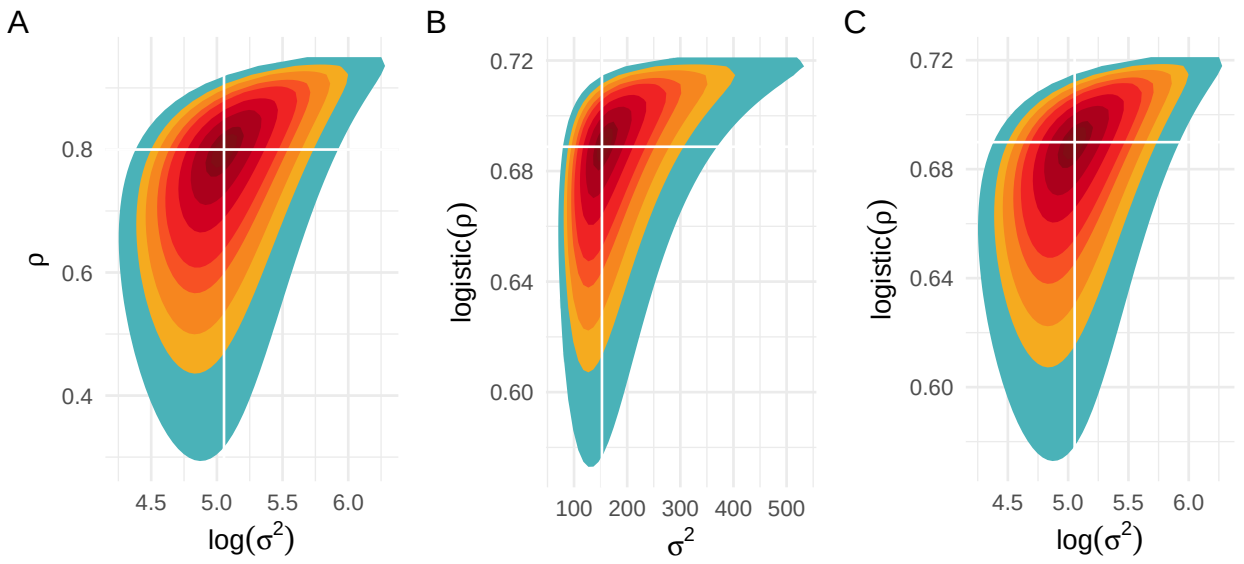

Figure 12: Deviance contours for different parameter transformations. Since here the parameter ρ is positive, we applied the logistic function. In white, the MLE's.

(d) A chamada transformação z de Fisher da correlação amostral é dada por

$$z = \frac{1}{2} \log \frac{1+\hat{\rho}}{1-\hat{\rho}}.$$

Compare a verossimilhança de ρ baseada na transformação z de Fisher com a verossimilhança perfilhada.

<r code>

```r
## original scale ======================================================
model_fit <- mle2(bigaussian, start = list("sigma2" = 150, "rho" = .5))
par_est <- model_fit@coef

sigma2_grid <- seq(70, 535, length.out = 50)
rho_grid <- seq(.25, 1, length.out = 50)

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance)
rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid
deviance_grid <- reshape2::melt(deviance_grid)
```

```r
p1 <- bigaussian_trans(dataset = deviance_grid,
                       xlab = expression(sigma^2),
                       ylab = expression(rho),
                       par_est = par_est, tag = "A.1") +
    labs(title = "Original Deviance contour")

data_profile <- as.data.frame(profile(model_fit, which = "rho"))

my_profile <- function(dataset, xlab, main, tag) {
    dataset <- subset(dataset, abs(z) < 3)
    ggplot(dataset, aes(x = focal, y = abs(z))) +
        geom_line(size = 1) + geom_point(size = 2) + theme_minimal() +
        labs(x = xlab, title = main, tag = tag)
}
p3 <- my_profile(data_profile, xlab = expression(rho), tag = "A.2",
                 main = expression(rho ~ "likelihood profile"))
## fisher z-transformation ==========================================
model_fit <- mle2(bigaussian,
                  start = list("sigma2" = 150,
                               "rho" = .5 * (log(1 + .5) - log(1 - .5))),
                  data = list(z = TRUE))
par_est <- model_fit@coef

ztrans <- function(rho) .5 * (log(1 + rho) - log(1 - rho))
rho_grid <- sapply(seq(-1, 1, length.out = 50), function(i) ztrans(i))

deviance_grid <- outer(sigma2_grid, rho_grid, FUN = bigaussian_deviance,
                       z = TRUE)
rownames(deviance_grid) <- sigma2_grid
colnames(deviance_grid) <- rho_grid
deviance_grid <- reshape2::melt(deviance_grid)

p2 <- bigaussian_trans(dataset = deviance_grid,
                       xlab = expression(sigma^2),
                       ylab = expression(z(rho)), par_est = par_est,
                       tag = "B.1") +
    labs(title = expression("with Fisher z-transformation in" ~ rho))
```

```
data_profile <- as.data.frame(profile(model_fit, which = "rho"))

p4 <- my_profile(data_profile, xlab = expression(z(rho)), tag = "B.2",
                 main = expression(z(rho) ~ "likelihood profile"))
## plotting ================================================================
grid.arrange(p1, p2, p3, p4, nrow = 2, ncol = 2)
```

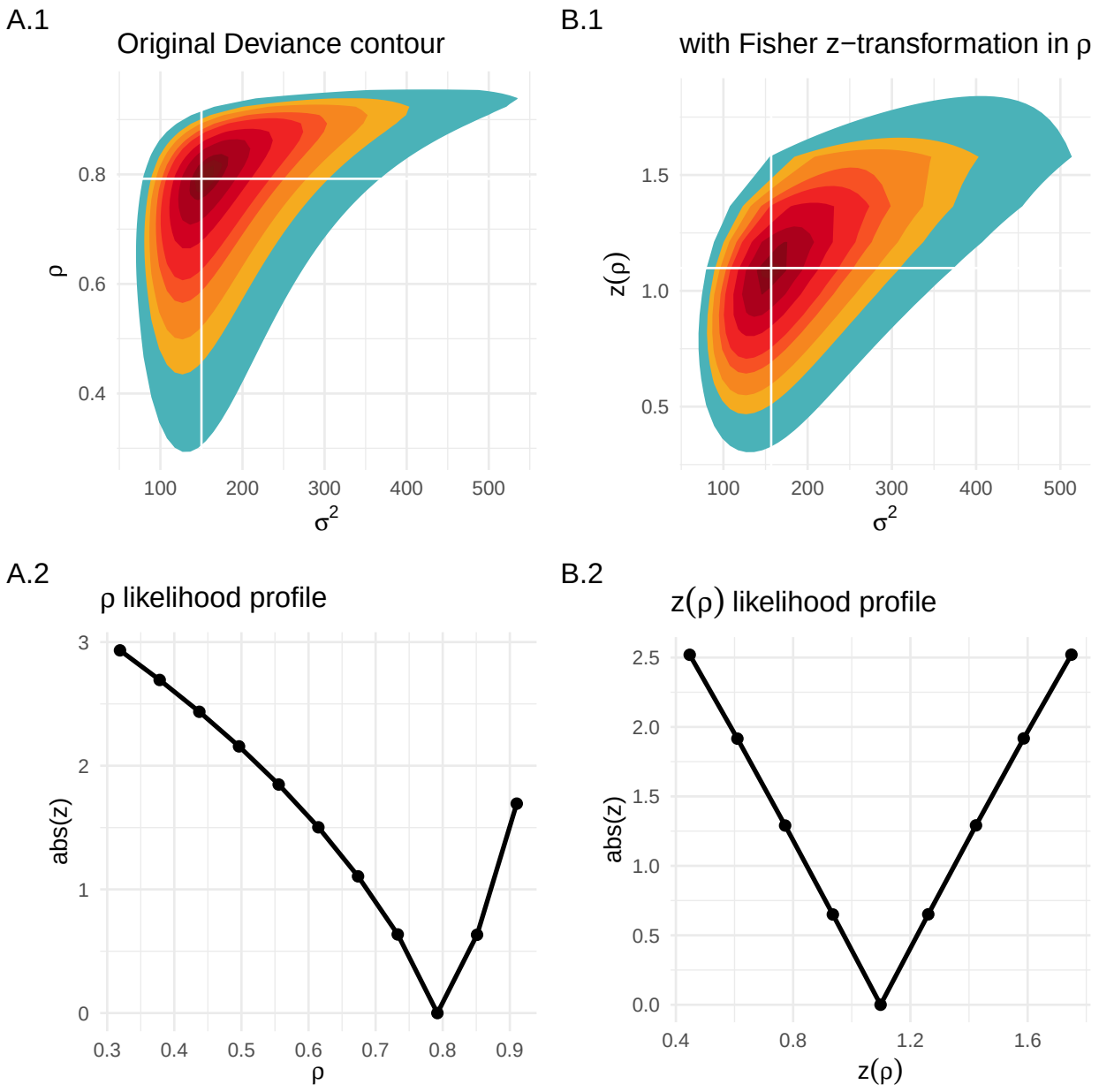

Figure 13: A: original likelihood. A.1 is the deviance contour, A.2 is the likelihood profile for ρ; B: applying the Fisher z-transformation in ρ. B.1 is the resulting deviance contour, B.2 is the likelihood profile for $z(\rho)$.

Exercise 3: regressão Gaussiana

Considere os dados da tabela a seguir (adaptados/modificados de Montgomery & Runger, 1994) aos quais deseja-se ajustar um modelo de regressão linear simples relacionando a variável resposta Y (pureza em %) a uma variável explicativa X (nível de hidrocarbonetos).

X	0.99	1.02	1.15	1.29	1.46	1.36	0.87	1.23	1.55	1.40
Y	99.01	89.05	91.43	93.74	96.73	94.45	87.59	91.77	99.42	93.65
X	1.19	1.15	0.98	1.01	1.11	1.20	1.26	1.32	1.43	0.95
Y	93.54	92.52	90.56	89.54	89.85	90.39	93.25	93.41	94.98	87.33

<r code>

```r
data_3 <- data.frame(
    x = c(.99, 1.02, 1.15, 1.29, 1.46, 1.36, .87, 1.23, 1.55, 1.4,
          1.19, 1.15, .98, 1.01, 1.11, 1.2, 1.26, 1.32, 1.43, .95),
    y = c(99.01, 89.05, 91.43, 93.74, 96.73, 94.45, 87.59, 91.77, 99.42,
          93.65, 93.54, 92.52, 90.56, 89.54, 89.85, 90.39, 93.25, 93.41,
          94.98, 87.33))
```

(a) Encontre a função de verossimilhança.

Assuming a Normal distribution for $Y|X$, we have the following log-likelihood, in matricial form

$$\begin{aligned} l(\theta; Y) &= -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2}\|Y - X\beta\|^2 \\ &= \text{constant} - n\log\sigma - \frac{1}{2\sigma^2}\|Y - X\beta\|^2, \end{aligned}$$

were $\theta = [\beta_0\ \beta_1\ \sigma]^\top$, $\beta = [\beta_0\ \beta_1]^\top$, and $\mu_i = \beta_0 + \beta_1 x_i$.

<r code>

```r
gaussian3_reg  <- function(b0, b1, sigma) {
    mu <- b0 + b1 * data_3$x
    lkl <- sum(dnorm(data_3$y, mean = mu, sd = sigma, log = TRUE))
    return(-lkl)
}
```

(b) Encontre as estimativas de máxima verossimilhança.

<r code>
```
model_fit <- mle2(gaussian3_reg,
                  start = list(b0 = min(data_3$y), b1 = 1.5,
                               sigma = sd(data_3$y)))
(par_est <- model_fit@coef)
```

```
       b0        b1     sigma
77.988429 12.225489  2.343799
```

Just checking...

<r code>
```
lm(y ~ x, data_3)$coef # beta_0 and beta_1
```

```
(Intercept)           x
   77.98996    12.22453
```

```
par_est[[3]]**2 # sigma^2
```

```
[1] 5.493392
```

```
(summary(lm(y ~ x, data_3))$sigma**2) * (nrow(data_3) - 2) / nrow(data_3)
```

```
[1] 5.493386
```

Checked, ✓.

(c) Obtenha a verossimilhança conjunta para os parâmetros β_0 e β_1:

Considerando σ fixo com valor igual à sua estimativa,

<r code>
```
gaussian3_deviance <- function(b0_grid, b1_grid, theta_est) {
    b0_est <- theta_est[[1]]
    b1_est <- theta_est[[2]]
    sigma_est <- theta_est[[3]]
    lkl_grid <- gaussian3_reg(b0_grid, b1_grid, sigma_est)
    lkl_est <- gaussian3_reg(b0_est, b1_est, sigma_est)
```

```
    return(2 * (lkl_grid - lkl_est))
}
gaussian3_deviance <- Vectorize(gaussian3_deviance,
                                c("b0_grid", "b1_grid"))
b0_grid <- seq(67, 89, length.out = 50)
b1_grid <- seq(3, 21, length.out = 50)
deviance_grid <- outer(b0_grid, b1_grid, FUN = gaussian3_deviance,
                       theta_est = par_est)
rownames(deviance_grid) <- b0_grid
colnames(deviance_grid) <- b1_grid
deviance_grid <- reshape2::melt(deviance_grid)
bigaussian_trans(deviance_grid,
                 xlab = expression(beta[0]), ylab = expression(beta[1]),
                 tag = NULL, par_est = par_est) +
    labs(title = "Deviance contour", subtitle = "MLE's, in white.")
```

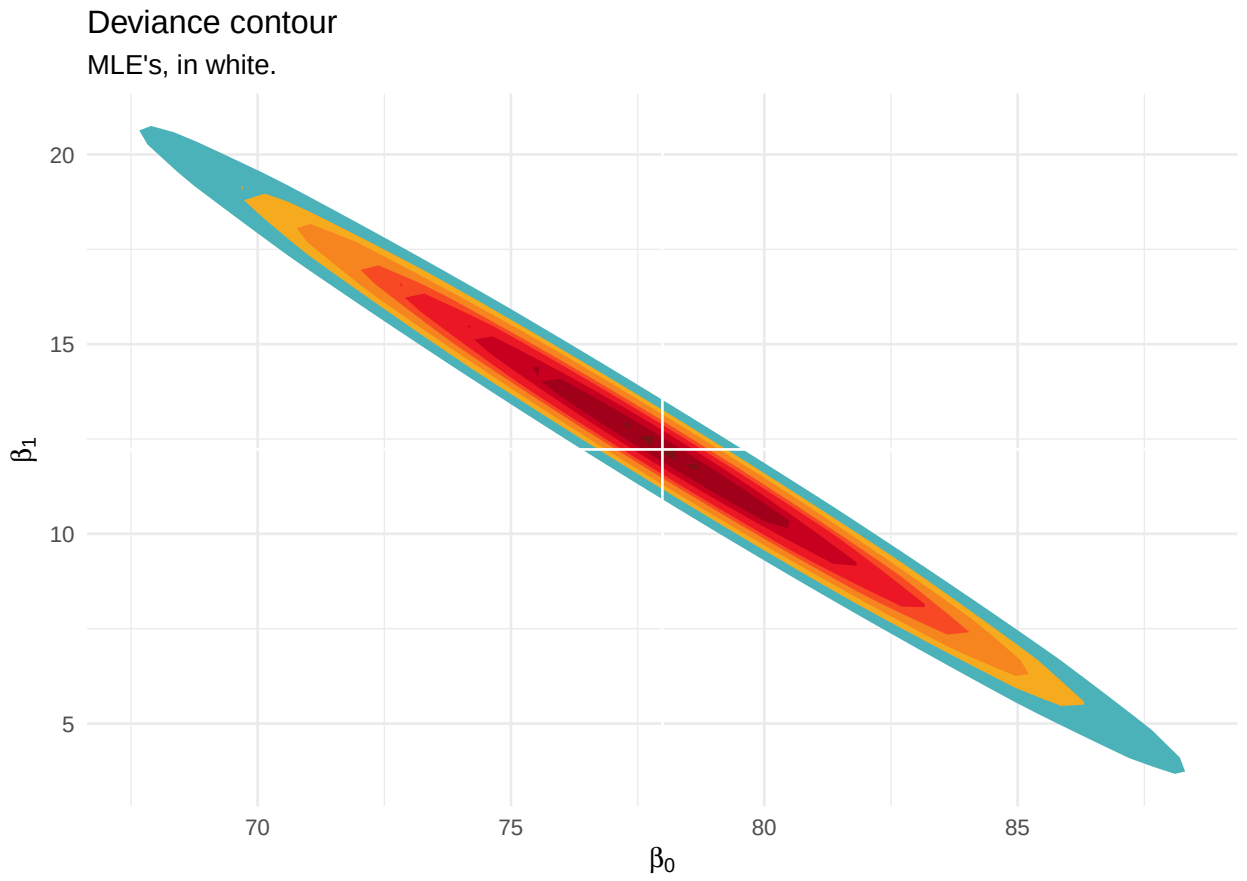

Figure 14: Deviance contour for the β's of a simple Gaussian linear regression with the standard deviation fixed in it's MLE.

Obtendo a verossimilhança perfilhada em relação a σ.

<r code>
```r
sigma_grid <- seq(1, 15, length.out = 50)
beta_perf <- matrix(0, nrow = length(sigma_grid), ncol = 3)
for (i in seq(sigma_grid)) {
    beta_perf[i, ] <- unlist(
        mle2(gaussian3_reg, start = list(b0 = min(data_3$y), b1 = 1.5),
             data = list(sigma = sigma_grid[i]))@details[1:2])
}
ggplot() + theme_minimal() +
    geom_line(aes(x = sigma_grid, y = -beta_perf[ , 3]), size = 1.5) +
    geom_vline(xintercept = par_est[3], linetype = "dashed") +
    labs(x = expression(sigma), y = NULL,
         title = expression(sigma ~ "profile likelihood"),
         subtitle = "MLE, in dashed.")
```

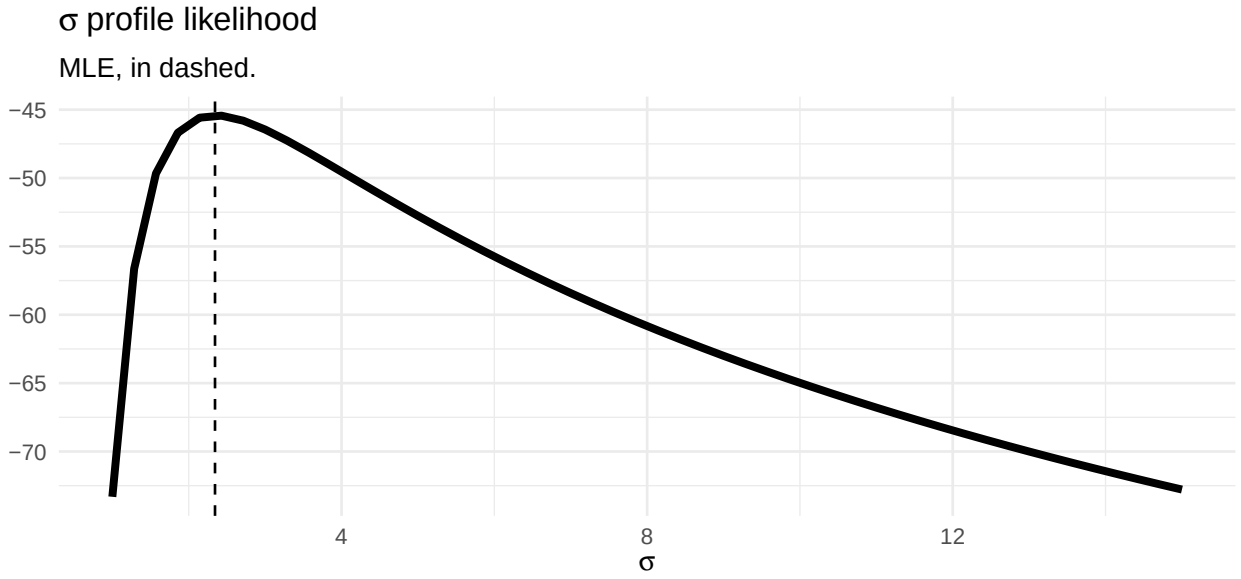

Figure 15: σ profile likelihood in a simple Gaussian linear regression. In dashed, the MLE.

And how the β's fluctuates across the grid of σ?

<r code>
```r
summary(beta_perf[ , 1]) # beta_0
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  77.99   77.99   78.02   78.04   78.07   78.20
```

```r
summary(beta_perf[ , 2]) # beta_1
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.05   12.15   12.20   12.19   12.22   12.23
```

Very few, as expected. Since the estimation of β doesn't depend on σ.

(d) Obtenha a verossimilhança perfilhada para os parâmetros β_0 e β_1 individualmente.

<r code>

```r
coef_profile <- function(grid, coef = c("b0", "b1")) {
    n <- length(grid)
    prof <- numeric(n)
    for (i in seq(n)) {
        prof[i] <-
            switch(coef,
                   "b0" = mle2(gaussian3_reg,
                               start = list(b1 = 1.5),
                               data = list(b0 = grid[i],
                                           sigma = par_est[[3]])
                               )@details$value,
                   "b1" = mle2(gaussian3_reg,
                               start = list(b0 = min(data_3$y)),
                               data = list(b1 = grid[i],
                                           sigma = par_est[[3]])
                               )@details$value)
    }
    return(-prof)
}
## beta_0 ======================================================================
b0_grid <- seq(56, 100, length.out = 50)
p1 <- ggplot() + theme_minimal() +
    geom_line(aes(x = b0_grid, y = coef_profile(b0_grid, "b0")),
              size = 1.5) +
    geom_vline(xintercept = par_est[1], linetype = "dashed") +
    labs(x = expression(beta[0]), y = NULL,
         title = expression(beta[0] ~ "profile likelihood"),
```

```
         subtitle = "MLE, in dashed.", tag = "A")
## beta_1 ============================================================
b1_grid <- seq(-6, 30, length.out = 50)
p2 <- ggplot() + theme_minimal() +
    geom_line(aes(x = b1_grid, y = coef_profile(b1_grid, "b1")),
              size = 1.5) +
    geom_vline(xintercept = par_est[2], linetype = "dashed") +
    labs(x = expression(beta[1]), y = NULL,
         title = expression(beta[1] ~ "profile likelihood"),
         subtitle = "MLE, in dashed.", tag = "B")
## plotting ==========================================================
grid.arrange(p1, p2, ncol = 2)
```

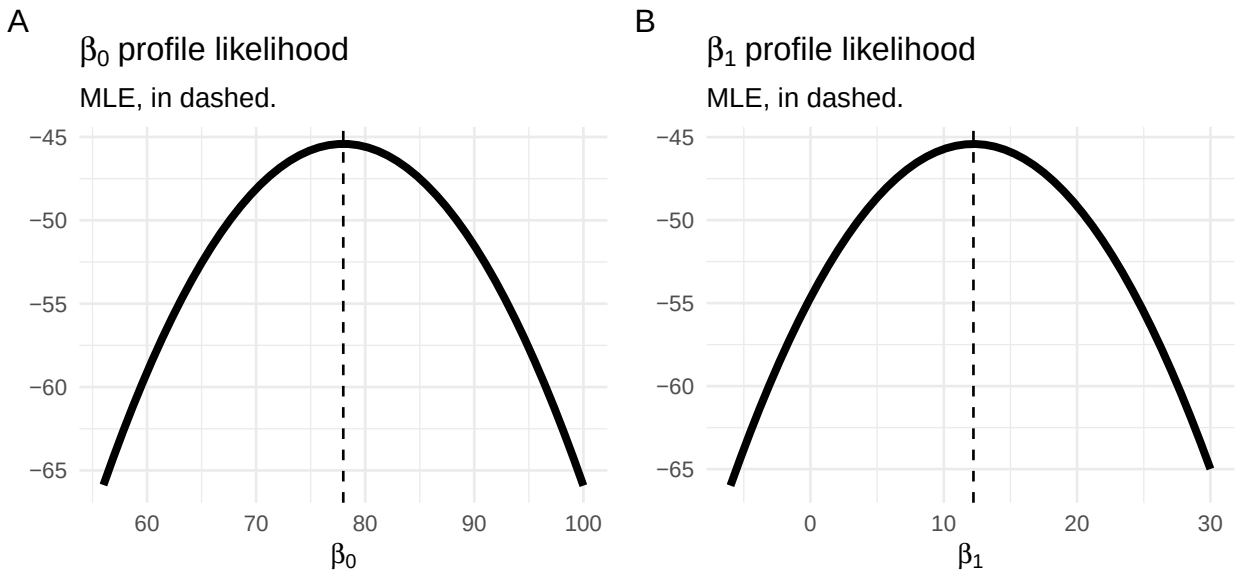

Figure 16: Profile likelihoods for β_0 and β_1 in a simple Gaussian linear regression. In dashed, the MLE's.

Exercise 4: regressão Gaussiana censurada

Repita o exercício anterior porém considerando os dados modificados como indicado na da tabela a seguir.

X	0.99	1.02	1.15	1.29	1.46	1.36	0.87	1.23	1.55	1.40
Y	99.01	≤ 90	91.43	$[91, 95]$	96.73	94.45	87.59	91.77	$[96, 100]$	93.65
X	1.19	1.15	0.98	1.01	1.11	1.20	1.26	1.32	1.43	0.95
Y	≥ 92.5	92.52	90.56	89.54	89.85	$[89, 93]$	93.25	93.41	≥ 93	87.33

(a) Encontre a função de verossimilhança.

<r code>

```r
data_4 <- data.frame(
    x = c(.99, 1.15, 1.46, 1.36, .87, 1.23, 1.4,
          1.15, .98, 1.01, 1.11, 1.26, 1.32, .95),
    y = c(99.01, 91.43, 96.73, 94.45, 87.59, 91.77, 93.65,
          92.52, 90.56, 89.54, 89.85, 93.25, 93.41, 87.33))

gaussian4_reg <- function(b0, b1, sigma) {
    mu <- b0 + b1 * data_4$x
    lkl_exact <- sum(dnorm(data_4$y, mean = mu, sd = sigma, log = TRUE))
    ##-------------------------------------------------------------------
    lkl_1 <- log(pnorm(90, mean = b0 + b1 * 1.02, sd = sigma))
    ##-------------------------------------------------------------------
    lkl_2 <- log(pnorm(95, mean = b0 + b1 * 1.29, sd = sigma) -
                 pnorm(91, mean = b0 + b1 * 1.29, sd = sigma))
    ##-------------------------------------------------------------------
    lkl_3 <- log(pnorm(100, mean = b0 + b1 * 1.55, sd = sigma) -
                 pnorm(96, mean = b0 + b1 * 1.55, sd = sigma))
    ##-------------------------------------------------------------------
    lkl_4 <- log(1 - pnorm(92.5, mean = b0 + b1 * 1.19, sd = sigma))
    ##-------------------------------------------------------------------
    lkl_5 <- log(pnorm(93, mean = b0 + b1 * 1.20, sd = sigma) -
                 pnorm(89, mean = b0 + b1 * 1.20, sd = sigma))
    ##-------------------------------------------------------------------
    lkl_6 <- log(1 - pnorm(93, mean = b0 + b1 * 1.43, sd = sigma))
    return(-(lkl_exact + lkl_1 + lkl_2 + lkl_3 + lkl_4 + lkl_5 + lkl_6))
}
```

(b) Encontre as estimativas de máxima verossimilhança.

<r code>
```r
model_fit <- mle2(gaussian4_reg,
                  start = list(b0 = min(data_4$y), b1 = 1.5,
                               sigma = sd(data_4$y)))
(par_est <- model_fit@coef)
```

```
       b0        b1     sigma
78.410145 11.869453  2.432607
```

Considering only the exact observations, we get these β's

<r code>
```r
lm(y ~ x, data_4)
```

```

Call:
lm(formula = y ~ x, data = data_4)

Coefficients:
(Intercept)            x
      80.58        10.04
```

(c) Obtenha a verossimilhança conjunta para os parâmetros β_0 e β_1:

Considerando σ fixo com valor igual à sua estimativa,

<r code>
```r
gaussian4_deviance <- function(b0_grid, b1_grid, theta_est) {
    b0_est <- theta_est[[1]]
    b1_est <- theta_est[[2]]
    sigma_est <- theta_est[[3]]
    lkl_grid <- gaussian4_reg(b0_grid, b1_grid, sigma_est)
    lkl_est <- gaussian4_reg(b0_est, b1_est, sigma_est)
    return(2 * (lkl_grid - lkl_est))
}
gaussian4_deviance <- Vectorize(gaussian4_deviance,
                                c("b0_grid", "b1_grid"))
b0_grid <- seq(67, 90, length.out = 50)
b1_grid <- seq(2, 22, length.out = 50)
```

```r
deviance_grid <- outer(b0_grid, b1_grid, FUN = gaussian4_deviance,
                       theta_est = par_est)
rownames(deviance_grid) <- b0_grid
colnames(deviance_grid) <- b1_grid
deviance_grid <- reshape2::melt(deviance_grid)
bigaussian_trans(deviance_grid,
                 xlab = expression(beta[0]), ylab = expression(beta[1]),
                 tag = NULL, par_est = par_est) +
    labs(title = "Deviance contour", subtitle = "MLE's, in white.")
```

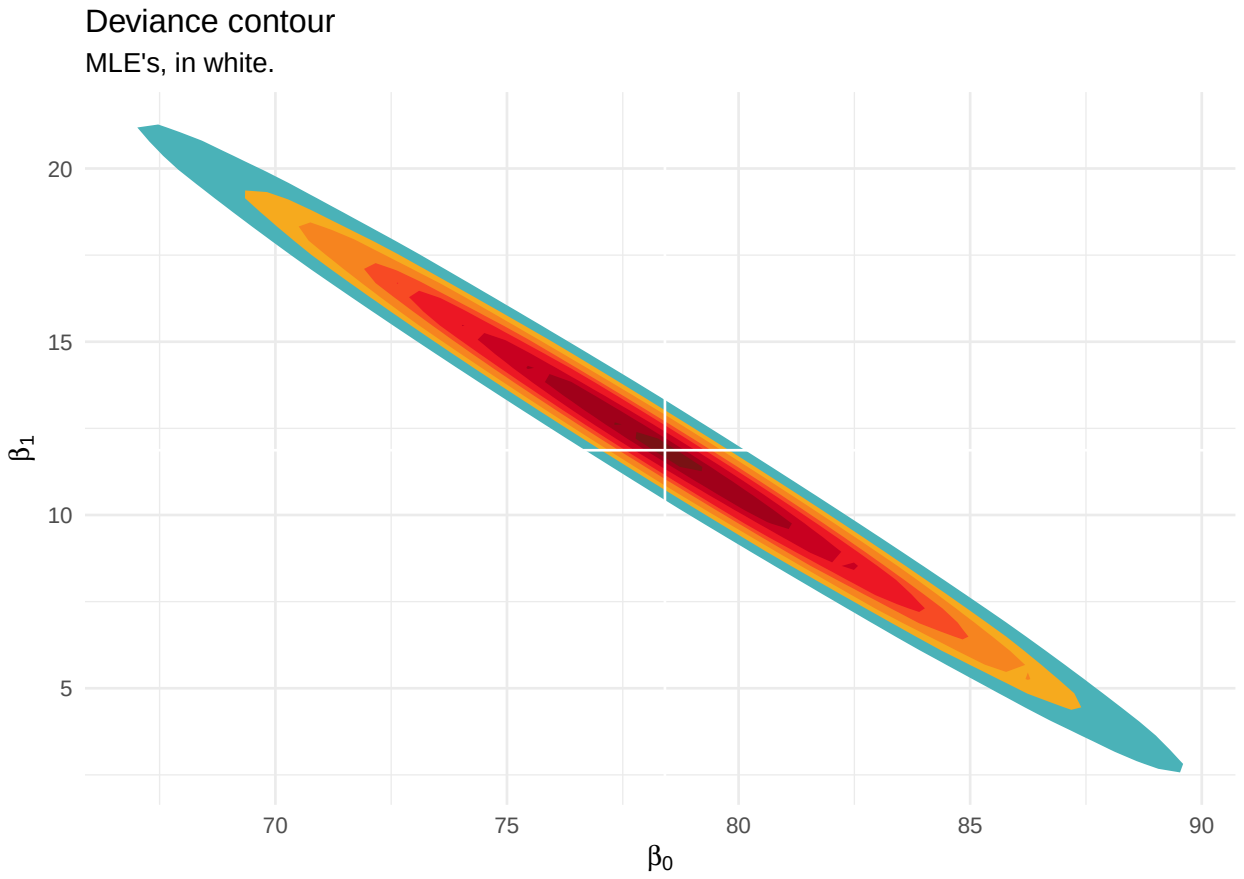

Figure 17: Deviance contour for the β's of a Gaussian linear regression with some censored observations and with the standard deviation fixed in it's MLE.

Obtendo a verossimilhança perfilhada em relação a σ.

<r code>

```r
sigma_grid <- seq(1, 17.5, length.out = 50)
beta_perf <- matrix(0, nrow = length(sigma_grid), ncol = 3)
for (i in seq(sigma_grid)) {
    beta_perf[i, ] <- unlist(
        mle2(gaussian4_reg, start = list(b0 = min(data_4$y), b1 = 1.5),
             data = list(sigma = sigma_grid[i]))@details[1:2])
}
ggplot() + theme_minimal() +
    geom_line(aes(x = sigma_grid, y = -beta_perf[ , 3]), size = 1.5) +
    geom_vline(xintercept = par_est[3], linetype = "dashed") +
    labs(x = expression(sigma), y = NULL,
         title = expression(sigma ~ "profile likelihood"),
         subtitle = "MLE, in dashed.")
```

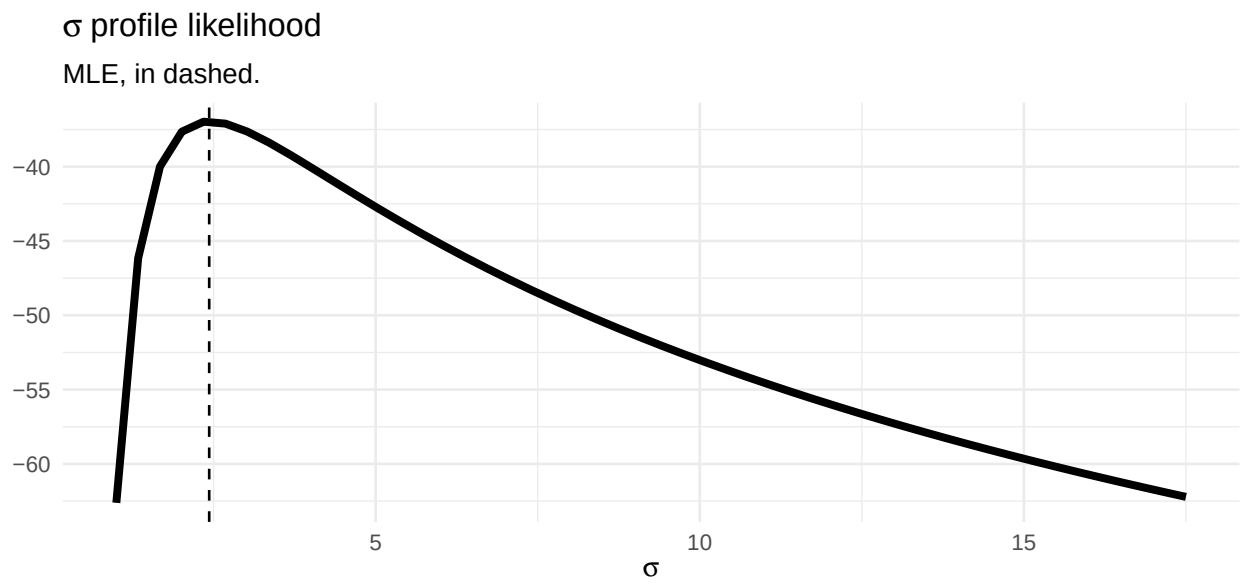

Figure 18: σ profile likelihood in a Gaussian linear regression with some censored observations. In dashed, the MLE.

And how the β's fluctuates across the grid of σ?

<r code>

```r
summary(beta_perf[ , 1]) # beta_0
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  70.40   72.35   74.70   74.68   76.84   79.54
```

```r
summary(beta_perf[ , 2]) # beta_1
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.90   13.27   15.21   15.23   17.31   19.09
```

Very few, as expected. Since the estimation of β doesn't depend on σ.

(d) Obtenha a verossimilhança perfilhada para os parâmetros β_0 e β_1 individualmente.

<r code>

```r
coef_profile <- function(grid, coef = c("b0", "b1")) {
    n <- length(grid)
    prof <- numeric(n)
    for (i in seq(n)) {
        prof[i] <-
            switch(coef,
                   "b0" = mle2(gaussian4_reg,
                               start = list(b1 = 5),
                               data = list(b0 = grid[i],
                                           sigma = par_est[[3]])
                               )@details$value,
                   "b1" = mle2(gaussian4_reg,
                               start = list(b0 = min(data_4$y)),
                               data = list(b1 = grid[i],
                                           sigma = par_est[[3]])
                               )@details$value)
    }
    return(-prof)
}
## beta_0 ======================================================================
b0_grid <- seq(68.1, 89, length.out = 50)
p1 <- ggplot() + theme_minimal() +
    geom_line(aes(x = b0_grid, y = coef_profile(b0_grid, "b0")),
              size = 1.5) +
    geom_vline(xintercept = par_est[1], linetype = "dashed") +
    labs(x = expression(beta[0]), y = NULL,
         title = expression(beta[0] ~ "profile likelihood"),
```

```
        subtitle = "MLE, in dashed.", tag = "A")
## beta_1 ============================================================
b1_grid <- seq(-7, 31, length.out = 50)
p2 <- ggplot() + theme_minimal() +
    geom_line(aes(x = b1_grid, y = coef_profile(b1_grid, "b1")),
              size = 1.5) +
    geom_vline(xintercept = par_est[2], linetype = "dashed") +
    labs(x = expression(beta[1]), y = NULL,
         title = expression(beta[1] ~ "profile likelihood"),
         subtitle = "MLE, in dashed.", tag = "B")
## plotting ==========================================================
grid.arrange(p1, p2, ncol = 2)
```

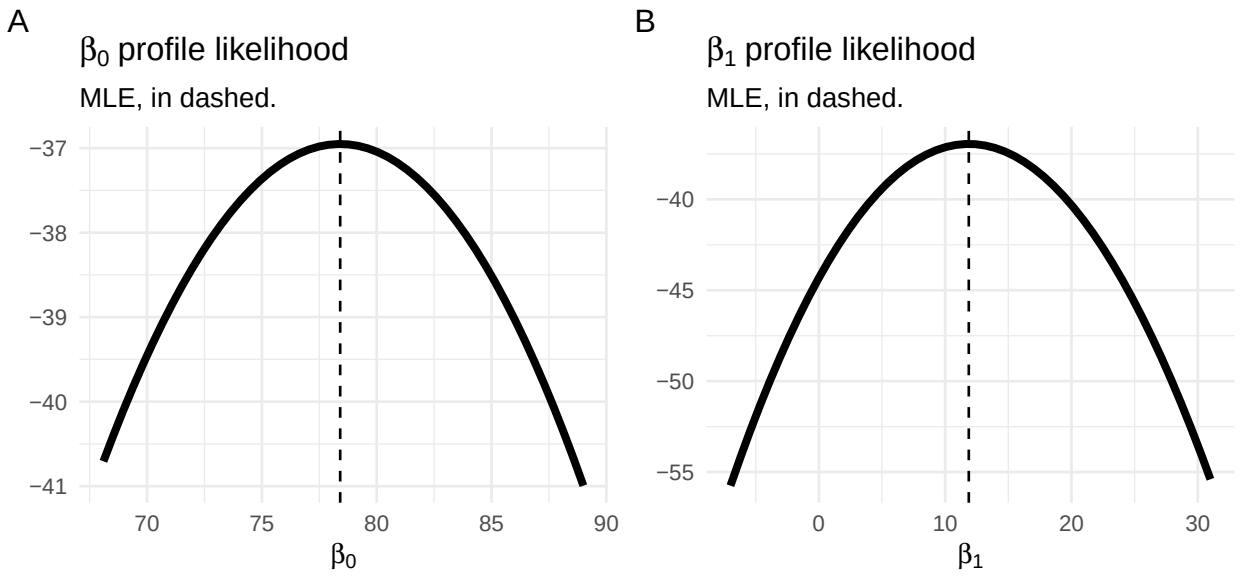

Figure 19: Profile likelihoods for β_0 and β_1 in a Gaussian linear regression with some censored observations. In dashed, the MLE's.

Last modification on ...

```
[1] "2019-10-25 20:35:51 -03"
```