Script para ajuste do modelo de Clutter pelo metodo dos Minimos Quarados em 2 Estagios utilizando o R

Sollano Rabelo Braga
Ana Carolina Araujo

Outubro, 2016

Contents

1) Carregar os pacotes e dados

Primeiro carrega-se as bibliotecas; Systemfit para o ajuste do modelo utilizando o metodo dos Minimos Quadrados em 2 Estagios, e tidyverse para manipulacao de dados e graficos.

```
library(systemfit)
```

```
## Loading required package: Matrix
```

```
## Loading required package: car
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## expand(): tidyr, Matrix
## filter(): dplyr, stats
## lag():    dplyr, stats
## recode(): dplyr, car
## some():   purrr, car
```

Carrega-se os dados em csv:

```
dados <- read.csv2("dados_clutter.csv")
head(dados, 10)
```

```
##    Parcela Idade   HD    N     V    B
## 1        1  26.4 12.4 1020  19.7  5.7
## 2        1  38.4 17.2 1020  60.8  9.8
## 3        1  51.6 19.1 1020 103.4 13.9
## 4        1  63.6 21.8 1020 136.5 15.3
## 5        2  26.4 15.0  900  27.3  6.0
## 6        2  38.4 20.3  900  80.0 10.5
## 7        2  50.4 22.0  900 111.0 13.3
## 8        2  62.4 24.4  900 148.3 15.0
## 9        3  26.4 16.0 1040  35.4  7.1
## 10       3  38.4 19.5 1040  69.0 10.5
```

2) Estimar site

Primeiro ajusta-se o modelo em funcao do inverso da idade; Para isso calcula-se o inverso da idade e ln(HD) separadamente:

```
dados$LN_HD <- log(dados$HD)
dados$I_IDADE <- 1/dados$Idade
```

Em seguida visualiza-se o ajuste, e salva-se os coeficientes em objetos separados:

```
summary(lm(LN_HD ~ I_IDADE, dados))
```

```
##
## Call:
## lm(formula = LN_HD ~ I_IDADE, data = dados)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30498 -0.06746  0.01493  0.09491  0.21761
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.51240    0.03248  108.14   <2e-16 ***
## I_IDADE     -19.11291    1.31017  -14.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1198 on 137 degrees of freedom
## Multiple R-squared:  0.6084, Adjusted R-squared:  0.6055
## F-statistic: 212.8 on 1 and 137 DF,  p-value: < 2.2e-16
```

```
b0 <- lm(LN_HD ~ I_IDADE, dados)[[1]][[1]]
b1 <- lm(LN_HD ~ I_IDADE, dados)[[1]][[2]]
b0
```

```
## [1] 3.512401
```

```
b1
```

```
## [1] -19.11291
```

Remover variaveis criadas:

```
dados[c("LN_HD", "I_IDADE")] <- NULL
```

Utilizando uma idade indice de 64, calcula-se o site:

```
Idade_I <- 64

dados$Site <- exp(log(dados$HD) - b1 * (1/dados$Idade -
    1/Idade_I))
head(dados)
```

```
##   Parcela Idade   HD    N     V    B     Site
## 1       1  26.4 12.4 1020  19.7  5.7 18.97327
## 2       1  38.4 17.2 1020  60.8  9.8 20.98908
## 3       1  51.6 19.1 1020 103.4 13.9 20.52111
## 4       1  63.6 21.8 1020 136.5 15.3 21.84098
## 5       2  26.4 15.0  900  27.3  6.0 22.95153
```

```
## 6       2  38.4 20.3  900  80.0 10.5 24.77199
```

3) Preparacao dos dados

Para se fazer o ajuste e necessario organizar os dados de uma certa forma. Separando os dados em idade 1 e 2, altura dominante 1 e 2, area basal 1 e 2, e volume 1 e 2. Para isso pode-se utilizar um loop for, ou o pacote dplyr.

3.1) Preparacao dos dados - rbase

```
list <- vector("list", nrow(dados))

for (i in 1:nrow(dados)) {

    if (dados$Parcela[i] == dados$Parcela[i + 1] && !is.na(dados$Idade[i +
        1])) {
        list[[i]] <- data.frame(Parcela = dados$Parcela[i],
            I1 = dados$Idade[i], I2 = dados$Idade[i + 1],
            HD1 = dados$HD[i], HD2 = dados$HD[i + 1], B1 = dados$B[i],
            B2 = dados$B[i + 1], V1 = dados$V[i], V2 = dados$V[i +
                1], Site = dados$Site[i])
    }

}
dados_prep <- do.call(rbind, list)
head(dados_prep, 10)
```

```
##    Parcela   I1   I2  HD1  HD2   B1   B2    V1    V2     Site
## 1        1 26.4 38.4 12.4 17.2  5.7  9.8  19.7  60.8 18.97327
## 2        1 38.4 51.6 17.2 19.1  9.8 13.9  60.8 103.4 20.98908
## 3        1 51.6 63.6 19.1 21.8 13.9 15.3 103.4 136.5 20.52111
## 4        2 26.4 38.4 15.0 20.3  6.0 10.5  27.3  80.0 22.95153
## 5        2 38.4 50.4 20.3 22.0 10.5 13.3  80.0 111.0 24.77199
## 6        2 50.4 62.4 22.0 24.4 13.3 15.0 111.0 148.3 23.84627
## 7        3 26.4 38.4 16.0 19.5  7.1 10.5  35.4  69.0 24.48164
## 8        3 38.4 51.6 19.5 21.9 10.5 12.0  69.0  97.3 23.79576
## 9        3 51.6 64.8 21.9 24.4 12.0 12.6  97.3 114.5 23.52945
## 10       4 32.4 43.2 20.8 22.7 11.1 14.9  79.6 128.0 27.83289
```

3.2) Preparacao dos dados - dplyr

Utilizando o dplyr, agrupa-se os dados por parcela, e utilizando a funcao lead, cria-se variaveis novas que chama o dados seguinte da linha. Lead funciona de forma similar a "i+1", utilizando no topico 3.1. obs: na omit remove as linhas com NA.

```
dados_prep <- dados %>% group_by(Parcela) %>% transmute(I1 = Idade,
    I2 = lead(Idade), HD1 = HD, HD2 = lead(HD), B1 = B,
    B2 = lead(B), V1 = V, V2 = lead(V), Site = Site) %>%
    na.omit
```

```
## Adding missing grouping variables: `Parcela`
```

```
dados_prep
```

```
## Source: local data frame [104 x 10]
## Groups: Parcela [35]
##
##    Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##      <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1        1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
## 2        1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3        1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
## 4        2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5        2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6        2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
## 7        3  26.4  38.4  16.0  19.5   7.1  10.5  35.4  69.0 24.48164
## 8        3  38.4  51.6  19.5  21.9  10.5  12.0  69.0  97.3 23.79576
## 9        3  51.6  64.8  21.9  24.4  12.0  12.6  97.3 114.5 23.52945
## 10       4  32.4  43.2  20.8  22.7  11.1  14.9  79.6 128.0 27.83289
## # ... with 94 more rows
```

4) Converter dados para forma estrutural

Para realizar o ajuste e necessario que os dados sejam convertidos para a forma estrutural do modelo: Agora sera ofeito o ajuste do modelo de Clutter:

Na sua forma estrutural, ele pode ser representado da seguinte forma:

$$Y_2 = \beta_0 + \beta_1 X_4 + \beta_2 X_5 + \beta_3 Y_1 + Ln(\varepsilon_1)$$

$$Y_1 = X_1 + \alpha_0 X_2 + \alpha_1 X_3 + ln(\varepsilon_2)$$

4.1) Converter dados para forma estrutural - rbase

```
dados_form_est <- dados_prep
dados_form_est$Y1 <- log(dados_prep$B2)
dados_form_est$Y2 <- log(dados_prep$V2)
dados_form_est$X1 <- log(dados_prep$B1) * (dados_prep$I1/dados_prep$I2)
dados_form_est$X2 <- 1 - dados_prep$I1/dados_prep$I2
dados_form_est$X3 <- (1 - dados_prep$I1/dados_prep$I2) *
    dados_prep$Site
dados_form_est$X4 <- 1/dados_prep$I2
dados_form_est$X5 <- dados_prep$Site

head(dados_form_est, 10)
```

```
## Source: local data frame [10 x 17]
## Groups: Parcela [4]
##
##    Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##      <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1        1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
```

```
## 2        1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3        1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
## 4        2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5        2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6        2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
## 7        3  26.4  38.4  16.0  19.5   7.1  10.5  35.4  69.0 24.48164
## 8        3  38.4  51.6  19.5  21.9  10.5  12.0  69.0  97.3 23.79576
## 9        3  51.6  64.8  21.9  24.4  12.0  12.6  97.3 114.5 23.52945
## 10       4  32.4  43.2  20.8  22.7  11.1  14.9  79.6 128.0 27.83289
##          Y1       Y2       X1        X2       X3         X4       X5
##       <dbl>    <dbl>    <dbl>     <dbl>    <dbl>      <dbl>    <dbl>
## 1  2.282382 4.107590 1.196570 0.3125000 5.929146 0.02604167 18.97327
## 2  2.631889 4.638605 1.698517 0.2558140 5.369299 0.01937984 20.98908
## 3  2.727853 4.916325 2.135306 0.1886792 3.871908 0.01572327 20.52111
## 4  2.351375 4.382027 1.231835 0.3125000 7.172355 0.02604167 22.95153
## 5  2.587764 4.709530 1.791524 0.2380952 5.898093 0.01984127 24.77199
## 6  2.708050 4.999237 2.090117 0.1923077 4.585820 0.01602564 23.84627
## 7  2.351375 4.234107 1.347565 0.3125000 7.650512 0.02604167 24.48164
## 8  2.484907 4.577799 1.749861 0.2558140 6.087286 0.01937984 23.79576
## 9  2.533697 4.740575 1.978722 0.2037037 4.793035 0.01543210 23.52945
## 10 2.701361 4.852030 1.805209 0.2500000 6.958222 0.02314815 27.83289
```

4.2) Converter dados para forma estrutural - dplyr

```
dados_form_est2 <- dados_prep %>% mutate(Y1 = log(B2), Y2 = log(V2),
    X1 = log(B1) * (I1/I2), X2 = 1 - I1/I2, X3 = (1 - I1/I2) *
        Site, X4 = 1/I2, X5 = Site)
dados_form_est2
```

```
## Source: local data frame [104 x 17]
## Groups: Parcela [35]
##
##    Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##      <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1        1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
## 2        1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3        1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
## 4        2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5        2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6        2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
## 7        3  26.4  38.4  16.0  19.5   7.1  10.5  35.4  69.0 24.48164
## 8        3  38.4  51.6  19.5  21.9  10.5  12.0  69.0  97.3 23.79576
## 9        3  51.6  64.8  21.9  24.4  12.0  12.6  97.3 114.5 23.52945
## 10       4  32.4  43.2  20.8  22.7  11.1  14.9  79.6 128.0 27.83289
##          Y1       Y2       X1        X2       X3         X4       X5
##       <dbl>    <dbl>    <dbl>     <dbl>    <dbl>      <dbl>    <dbl>
## 1  2.282382 4.107590 1.196570 0.3125000 5.929146 0.02604167 18.97327
## 2  2.631889 4.638605 1.698517 0.2558140 5.369299 0.01937984 20.98908
## 3  2.727853 4.916325 2.135306 0.1886792 3.871908 0.01572327 20.52111
## 4  2.351375 4.382027 1.231835 0.3125000 7.172355 0.02604167 22.95153
## 5  2.587764 4.709530 1.791524 0.2380952 5.898093 0.01984127 24.77199
## 6  2.708050 4.999237 2.090117 0.1923077 4.585820 0.01602564 23.84627
## 7  2.351375 4.234107 1.347565 0.3125000 7.650512 0.02604167 24.48164
```

```
## 8  2.484907 4.577799 1.749861 0.2558140 6.087286 0.01937984 23.79576
## 9  2.533697 4.740575 1.978722 0.2037037 4.793035 0.01543210 23.52945
## 10 2.701361 4.852030 1.805209 0.2500000 6.958222 0.02314815 27.83289
## # ... with 94 more rows
```

5) Ajuste do modelo de Clutter pelo metodo MQ2E (2SLS)

Agora sera feito o ajuste do modelo de Clutter:

$$Ln(B_2) = LnB_1\left(\frac{I_1}{I_2}\right) + \alpha_0\left(1 - \frac{I_1}{I_2}\right) + \alpha_1\left(1 - \frac{I_1}{I_2}\right)S + ln(\varepsilon_2)$$

$$Ln(V_2) = \beta_0 + \beta_1\left(\frac{1}{I_2}\right) + \beta_2 S + \beta_3 Ln(B_2) + Ln(\varepsilon_1)$$

Na sua forma estrutural, ele pode ser representado da seguinte forma:

$$Y_1 = X_1 + \alpha_0 X_2 + \alpha_1 X_3 + ln(\varepsilon_2)$$

$$Y_2 = \beta_0 + \beta_1 X_4 + \beta_2 X_5 + \beta_3 Y_1 + Ln(\varepsilon_1)$$

O ajuste pode ser feito tanto utilizando o pacote systemfit, tanto quanto utilizando o rbase, porem, utilizando o pacote, o ajuste e mais preciso.

5.1) Ajuste do modelo pelo metodo MQ2E (2SLS) - rbase

O modelo de area basal nao possui intercepto, e o coeficiente de X1 possui valor 1. Para gerar este resultado, deve-se seguir os seguintes passos:

Primeiro gera-se uma copia dos dados:

```
dados_form_est_aux <- dados_form_est
head(dados_form_est_aux)
```

```
## Source: local data frame [6 x 17]
## Groups: Parcela [2]
##
##   Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1       1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
## 2       1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3       1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
## 4       2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5       2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6       2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
##         Y1       Y2       X1        X2       X3         X4       X5
##      <dbl>    <dbl>    <dbl>     <dbl>    <dbl>      <dbl>    <dbl>
## 1 2.282382 4.107590 1.196570 0.3125000 5.929146 0.02604167 18.97327
## 2 2.631889 4.638605 1.698517 0.2558140 5.369299 0.01937984 20.98908
## 3 2.727853 4.916325 2.135306 0.1886792 3.871908 0.01572327 20.52111
## 4 2.351375 4.382027 1.231835 0.3125000 7.172355 0.02604167 22.95153
## 5 2.587764 4.709530 1.791524 0.2380952 5.898093 0.01984127 24.77199
## 6 2.708050 4.999237 2.090117 0.1923077 4.585820 0.01602564 23.84627
```

Calcula-se o inverso da idade:

```
dados_form_est_aux$INV_I1 <- 1/dados_form_est_aux$I1
```

Ajusta-se o sistema na sua forma reduzida, de forma linear:

```
reg_B2_linear <- lm(Y1 ~ INV_I1 + Site + X1 + X2 + X3, dados_form_est_aux)

summary(reg_B2_linear)
```

```
##
## Call:
## lm(formula = Y1 ~ INV_I1 + Site + X1 + X2 + X3, data = dados_form_est_aux)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50808 -0.02129  0.00163  0.03451  0.13408
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.29442    0.34364  -3.767 0.000282 ***
## INV_I1      14.08940    3.62964   3.882 0.000188 ***
## Site         0.01042    0.01401   0.744 0.458592
## X1           1.33181    0.06288  21.180  < 2e-16 ***
## X2           5.24795    1.57377   3.335 0.001207 **
## X3          -0.07075    0.05406  -1.309 0.193699
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07333 on 98 degrees of freedom
## Multiple R-squared:  0.955,  Adjusted R-squared:  0.9527
## F-statistic: 415.6 on 5 and 98 DF,  p-value: < 2.2e-16
```

```
tab_coef_B2_linear <- data.frame(b0 = coef(reg_B2_linear)[1],
    b1 = coef(reg_B2_linear)[2], b2 = coef(reg_B2_linear)[3],
    b3 = coef(reg_B2_linear)[4], b4 = coef(reg_B2_linear)[5],
    b5 = coef(reg_B2_linear)[6], row.names = NULL)
tab_coef_B2_linear
```

```
##          b0      b1         b2       b3       b4          b5
## 1 -1.294421 14.0894 0.01042306 1.331811 5.247955 -0.07074907
```

Estima-se a area basal com os coeficientes:

```
dados_form_est_aux$Y1_EST <- tab_coef_B2_linear$b0 + tab_coef_B2_linear$b1 *
    dados_form_est_aux$INV_I1 + tab_coef_B2_linear$b2 *
    dados_form_est_aux$Site + tab_coef_B2_linear$b3 * dados_form_est_aux$X1 +
    tab_coef_B2_linear$b4 * dados_form_est_aux$X2 + tab_coef_B2_linear$b5 *
    dados_form_est_aux$X3
head(dados_form_est_aux)
```

```
## Source: local data frame [6 x 19]
## Groups: Parcela [2]
##
##   Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1       1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
```

```
## 2       1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3       1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
## 4       2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5       2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6       2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
##         Y1       Y2       X1        X2       X3         X4       X5
##      <dbl>    <dbl>    <dbl>     <dbl>    <dbl>      <dbl>    <dbl>
## 1 2.282382 4.107590 1.196570 0.3125000 5.929146 0.02604167 18.97327
## 2 2.631889 4.638605 1.698517 0.2558140 5.369299 0.01937984 20.98908
## 3 2.727853 4.916325 2.135306 0.1886792 3.871908 0.01572327 20.52111
## 4 2.351375 4.382027 1.231835 0.3125000 7.172355 0.02604167 22.95153
## 5 2.587764 4.709530 1.791524 0.2380952 5.898093 0.01984127 24.77199
## 6 2.708050 4.999237 2.090117 0.1923077 4.585820 0.01602564 23.84627
##       INV_I1   Y1_EST
##        <dbl>    <dbl>
## 1 0.03787879 2.251138
## 2 0.02604167 2.515992
## 3 0.01937984 2.752593
## 4 0.03787879 2.251613
## 5 0.02604167 2.548891
## 6 0.01984127 2.702103
```

Agora ajusta-se o modelo de volume, utilizando Y1 estimado:

```
reg_V2 <- lm(Y2 ~ X4 + X5 + Y1_EST, dados_form_est_aux)

summary(reg_V2)
```

```
##
## Call:
## lm(formula = Y2 ~ X4 + X5 + Y1_EST, data = dados_form_est_aux)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52130 -0.05393  0.00947  0.06634  0.24089
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.266261   0.125587  10.083  < 2e-16 ***
## X4          -22.367325   3.364847  -6.647 1.59e-09 ***
## X5            0.030622   0.006664   4.595 1.26e-05 ***
## Y1_EST        1.223357   0.063888  19.148  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1013 on 100 degrees of freedom
## Multiple R-squared:  0.9635, Adjusted R-squared:  0.9624
## F-statistic:   880 on 3 and 100 DF,  p-value: < 2.2e-16
```

```
tab_coef_V2 <- data.frame(b0 = coef(reg_V2)[1], b1 = coef(reg_V2)[2],
    b2 = coef(reg_V2)[3], b3 = coef(reg_V2)[4], row.names = NULL)
tab_coef_V2
```

```
##         b0        b1         b2       b3
## 1 1.266261 -22.36733 0.03062226 1.223357
```

Agora ja e possivel ajustar o modelo de area basal; faz-se o ajuste de forma nao linear, para que se possa restringir os coeficientes do intercepto e do X1:

```
reg_B2_n_linear <- nls(Y1 ~ X1 + a0 * X2 + a1 * X3, dados_form_est_aux)
```

```
## Warning in nls(Y1 ~ X1 + a0 * X2 + a1 * X3, dados_form_est_aux): No starting values specified for so
## Initializing 'a0', 'a1' to '1.'.
## Consider specifying 'start' or using a selfStart model
```

```
summary(reg_B2_n_linear)
```

```
##
## Formula: Y1 ~ X1 + a0 * X2 + a1 * X3
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a0  1.69920    0.31267   5.434 3.76e-07 ***
## a1  0.06491    0.01239   5.240 8.70e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08629 on 102 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 3.798e-08
```

```
tab_coef_B2_n_linear <- data.frame(a0 = coef(reg_B2_n_linear)[1],
    a1 = coef(reg_B2_n_linear)[2], row.names = NULL)
tab_coef_B2_n_linear
```

```
##         a0         a1
## 1 1.699197 0.06490728
```

Agora une-se as duas tabelas de coeficientes, obtendo-se a tabela de coeficientes do modelo de Clutter:

```
tab_coef_clut_rbase <- cbind(tab_coef_V2, tab_coef_B2_n_linear)
tab_coef_clut_rbase
```

```
##         b0        b1         b2       b3       a0         a1
## 1 1.266261 -22.36733 0.03062226 1.223357 1.699197 0.06490728
```

Obs: Caso o objetivo seja ajustar o modelo modificado de Clutter, basta realziar as alterações na hora de realizar o ajuste.

5.2) Ajuste do modelo pelo metodo MQ2E (2SLS) - systemfit

Primeiro visualiza-se os dados que serao utilizados:

```
head(dados_form_est)
```

```
## Source: local data frame [6 x 17]
## Groups: Parcela [2]
##
##   Parcela    I1    I2   HD1   HD2    B1    B2    V1    V2     Site
##     <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1       1  26.4  38.4  12.4  17.2   5.7   9.8  19.7  60.8 18.97327
## 2       1  38.4  51.6  17.2  19.1   9.8  13.9  60.8 103.4 20.98908
## 3       1  51.6  63.6  19.1  21.8  13.9  15.3 103.4 136.5 20.52111
```

```
## 4       2  26.4  38.4  15.0  20.3   6.0  10.5  27.3  80.0 22.95153
## 5       2  38.4  50.4  20.3  22.0  10.5  13.3  80.0 111.0 24.77199
## 6       2  50.4  62.4  22.0  24.4  13.3  15.0 111.0 148.3 23.84627
##         Y1       Y2       X1        X2       X3         X4       X5
##      <dbl>    <dbl>    <dbl>     <dbl>    <dbl>      <dbl>    <dbl>
## 1 2.282382 4.107590 1.196570 0.3125000 5.929146 0.02604167 18.97327
## 2 2.631889 4.638605 1.698517 0.2558140 5.369299 0.01937984 20.98908
## 3 2.727853 4.916325 2.135306 0.1886792 3.871908 0.01572327 20.52111
## 4 2.351375 4.382027 1.231835 0.3125000 7.172355 0.02604167 22.95153
## 5 2.587764 4.709530 1.791524 0.2380952 5.898093 0.01984127 24.77199
## 6 2.708050 4.999237 2.090117 0.1923077 4.585820 0.01602564 23.84627
```

Para se realizar este ajuste, sera utilizada a funcao systemfit, do pacote de mesmo nome. Nela deve-se definir o sistema de equacoes que sera utilizado no ajuste, e os instrumentos. Para definir o sistema, cria-se dois objetos representando as duas equacoes, e salva-se em uma lista:

```
eq1 <- Y2 ~ X4 + X5 + Y1
eq2 <- Y1 ~ X1 + X2 + X3
system <- list(Volume = eq1, AreaBasal = eq2)
```

Da mesma forma, os instrumentos sao salvos em um objeto separado:

```
inst <- ~X4 + X5 + X1 + X2 + X3
```

Com estes objetos definidos, e possivel se ajustar o modelo:

```
systemfit(system, "2SLS", inst = inst, data = dados_form_est)
```

```
##
## systemfit results
## method: 2SLS
##
## Coefficients:
##    Volume_(Intercept)             Volume_X4             Volume_X5
##             1.2610748           -22.1719700             0.0303653
##             Volume_Y1 AreaBasal_(Intercept)          AreaBasal_X1
##             1.2262434             -0.8642016             1.2518915
##          AreaBasal_X2          AreaBasal_X3
##             5.2216742             -0.0116815
```

Porem, observa-se que foram gerados 8 coeficientes, quando o objetivo seriam apenas 6. Isto porque e preciso especificar que deseja-se zerar o coeficiente do intercepto, e fazer com que o coeficiente de X1 seja 1.

Para isso, esta funcao possui a entrada de matrizes de restricao linear, que permitem ao usuario criar restricoes ao ajuste. A matriz deve ser composta de j x k, onde j representa o numero de restricoes, e k o numero de coeficientes. Neste caso, iremos realizar 2 restricoes, e temos 8 coeficientes, portanto, cria-se uma matriz preenchida por zeros, de 2x8:

```
restrict <- matrix(0, nrow = 2, ncol = 8)
```

Zeros significam que o coeficiente nao sera alterado. Portanto altera-se as posicoes 1x5 e 2x6 para 1, para que sejam impostas restricoes nos coeficientes 5 e 6, como desejado:

```
restrict[1, 5] <- 1
restrict[2, 6] <- 1

restrict
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    0    0    0    1    0    0    0
```

```
## [2,]    0    0    0    0    0    1    0    0
```

Deve ser definido tambem um vetor que contem j elementos; por padrao ele e consistido por zeros. Neste caso, ele deve ser composto por 0 e 1, pois deseja-se que o segundo coeficiente seja 1. (nao tenho certeza)

```
restrict.rhs <- c(0, 1)
restrict.rhs
```

```
## [1] 0 1
```

Agora ja e possivel realizar o ajuste:

```
reg_clutter <- systemfit(system, "2SLS", inst = inst, data = dados_form_est,
    restrict.matrix = restrict, restrict.rhs = restrict.rhs)
```

Observa-se que o intercepto da equacao de area basal tem seu valor bem proximo de zero, e X1 tem valor 1, assim como desejado:

```
summary(reg_clutter)
```

```
##
## systemfit results
## method: 2SLS
##
##          N  DF     SSR detRCov   OLS-R2 McElroy-R2
## system 208 202 1.10954 2.6e-05 0.972154   0.980009
##
##             N  DF      SSR      MSE     RMSE       R2   Adj R2
## Volume    104 100 0.350097 0.003501 0.059169 0.987560 0.987187
## AreaBasal 104 100 0.759446 0.007446 0.086288 0.935101 0.934464
##
## The covariance matrix of the residuals
##                 Volume    AreaBasal
## Volume     0.003500967 -0.000329963
## AreaBasal -0.000329963  0.007445546
##
## The correlations of the residuals
##               Volume  AreaBasal
## Volume     1.0000000 -0.0646284
## AreaBasal -0.0646284  1.0000000
##
##
## 2SLS estimates for 'Volume' (equation 1)
## Model Formula: Y2 ~ X4 + X5 + Y1
## <environment: 0x00000000132c0168>
## Instruments: ~X4 + X5 + X1 + X2 + X3
## <environment: 0x00000000132c0168>
##
##                 Estimate   Std. Error  t value   Pr(>|t|)
## (Intercept)   1.26107480   0.09198602 13.70942 < 2.22e-16 ***
## X4          -22.17197001   2.46668986 -8.98855 2.2204e-16 ***
## X5            0.03036532   0.00488441  6.21678 2.8588e-09 ***
## Y1            1.22624339   0.04687084 26.16218 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.059169 on 100 degrees of freedom
```

```
## Number of observations: 104 Degrees of Freedom: 100
## SSR: 0.350097 MSE: 0.003501 Root MSE: 0.059169
## Multiple R-Squared: 0.98756 Adjusted R-Squared: 0.987187
##
##
## 2SLS estimates for 'AreaBasal' (equation 2)
## Model Formula: Y1 ~ X1 + X2 + X3
## <environment: 0x00000000132c0168>
## Instruments: ~X4 + X5 + X1 + X2 + X3
## <environment: 0x00000000132c0168>
##
##                Estimate  Std. Error t value   Pr(>|t|)
## (Intercept) 3.60057e-16 0.00000e+00     Inf < 2.22e-16 ***
## X1          1.00000e+00 0.00000e+00     Inf < 2.22e-16 ***
## X2          1.69920e+00 2.68559e-01 6.32709 1.5799e-09 ***
## X3          6.49073e-02 1.06395e-02 6.10062 5.2997e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.086288 on 100 degrees of freedom
## Number of observations: 104 Degrees of Freedom: 100
## SSR: 0.759446 MSE: 0.007446 Root MSE: 0.086288
## Multiple R-Squared: 0.935101 Adjusted R-Squared: 0.934464
```

E interessante se realizar o teste de Durbin-Watson ele testa a hiptose H0 de que nao ha autocorrelacao entre os erros:

```
dwtest(eq1, data = dados_form_est)
```

```
##
##  Durbin-Watson test
##
## data:  eq1
## DW = 1.7411, p-value = 0.07794
## alternative hypothesis: true autocorrelation is greater than 0
```

```
dwtest(eq2, data = dados_form_est)
```

```
##
##  Durbin-Watson test
##
## data:  eq2
## DW = 1.5409, p-value = 0.007216
## alternative hypothesis: true autocorrelation is greater than 0
```

Agora basta criar uma tabela de coeficientes, denomimando-os de acordo; serao selecionados todos os coeficientes, menos 5 e 6, ja que foram restringidos no ajuste:

```
tab_coef_clut <- data.frame(b0 = coef(reg_clutter)[1], b1 = coef(reg_clutter)[2],
    b2 = coef(reg_clutter)[3], b3 = coef(reg_clutter)[4],
    a0 = coef(reg_clutter)[7], a1 = coef(reg_clutter)[8],
    row.names = NULL)

tab_coef_clut
```

```
##         b0        b1         b2       b3       a0         a1
## 1 1.261075 -22.17197 0.03036532 1.226243 1.699198 0.06490727
```

Obs: Caso o objetivo seja ajustar o modelo modificado de Clutter, além de se realizar as alterações nos modelos, deve-se alterar o número de colunas da matriz de restrição.

6) Definicao das classes

O primeiro passo e calcular o site medio, e anexa-lo aos dados originais; feito isso, arredonda-se os dados para 4 casas decimais, e organiza-se os dados de menor para maior com base no site medio.

```
dados_class <- dados %>% group_by(Parcela) %>% summarise(Site_medio = mean(Site)) %>%
    left_join(dados, ., by = "Parcela") %>% round(4) %>%
    arrange(Site_medio)
head(dados)
```

```
##   Parcela Idade   HD    N     V    B     Site
## 1       1  26.4 12.4 1020  19.7  5.7 18.97327
## 2       1  38.4 17.2 1020  60.8  9.8 20.98908
## 3       1  51.6 19.1 1020 103.4 13.9 20.52111
## 4       1  63.6 21.8 1020 136.5 15.3 21.84098
## 5       2  26.4 15.0  900  27.3  6.0 22.95153
## 6       2  38.4 20.3  900  80.0 10.5 24.77199
```

Serao utilizadas 3 classes, inferior media e superior (1, 2 e 3). Entao define-se o numero de classes que sera utilizado:

```
nc <- 3
```

cria-se uma lista com o numero de elementos correspondente ao numero de classes

```
list <- vector("list", nc)
```

define-se o intervalo que sera utilizado no calculo

```
intervalo <- (max(dados_class$Site_medio) - min(dados_class$Site_medio))/nc
```

cria-se o primeiro intervalo da tabela de classe; a tabela de classe possui 3 colunas: primeiro o intervalo inferior, segundo o intervalo inferior, terceiro o centro de classe.

```
list[[1]] <- c(min(dados_class$Site_medio), min(dados_class$Site_medio) +
    intervalo, mean(c(min(dados_class$Site_medio), min(dados_class$Site_medio) +
    intervalo)), 1)
```

com o loop for cria-se os demais intervalos; o loop vai da segunda linha da lista, ate a ultima linha, que corresponde ao numero de classes utilizado. o loop cria as 3 colunas citadas anteriormente, seguindo o padrao: primeiro seleciona-se o intervalo superior da classe anterior, ou seja, linha anterior [i-1] posicao 2 [2]; segundo seleciona-se o mesmo item selecionado anteriormente, e adiciona-se o intervalo de classe; terceiro cria-se o centro de classe.

```
for (i in 2:nc) {

    list[[i]] <- c(list[[i - 1]][[2]], list[[i - 1]][[2]] +
        intervalo, mean(c(list[[i - 1]][[2]], list[[i -
        1]][[2]] + intervalo)), i)

}
```

transformacao da lista em matriz e em seguida em data frame

```
classe <- data.frame(do.call(rbind, list))
```

renomear

```
names(classe) <- c("inf", "sup", "cc", "categoria")
```

Criar as classes com funcao composta:

```
classe <- class_table(dados_class, "Site_medio", 3)
```

Cria-se uma coluna adicional, com a classificacao em forma de caractere:

```
classe$categoria_ <- c("Inferior", "Media", "Superior")
classe
```

```
##        inf      sup       cc categoria categoria_
## 1 20.13570 23.38947 21.76258         1   Inferior
## 2 23.38947 26.64323 25.01635         2      Media
## 3 26.64323 29.89700 28.27012         3   Superior
```

Para se anexar as classes aos dados originais, compara-se o site medio com os intervalos superiores das classes. Para isso, sera utilizado o loop while, em conjunto com o loop for.

Primeiro, cria-se duas listas com o tamanho do dataframe:

```
list1 <- vector("list", nrow(dados_class))
list2 <- vector("list", nrow(dados_class))
```

Serao utilizadas duas listas, pois serao gerados dados de classes diferentes, como eles serao gerados em vetores, nao podem ser misturados. Se numeros e caracteres sao utilizados no mesmo vetor, todos viram caracteres. Em seguida inicia-se o vetor do loop while em 1:

```
i <- 1
```

Agora, os loops:

```
# Para (loop for) cada classe (1, 2 e 3), faz-se o loop
# while.
for (j in 1:nrow(classe)) {

    # enquanto a classe j for maior ou igual que o site
    # medio i,
    while (classe$sup[j] >= dados_class$Site_medio[i]) {

        # insere-se o intervalo j e a categoria j em um
        # elementos i das listas

        list1[[i]] <- classe$sup[j]

        list2[[i]] <- c(classe$categoria[j], classe$categoria_[j])

        # aumentar i, ou seja, passar para o proximo site medio
        i <- i + 1

        # parar quando acabar o dataframe
        if (is.na(dados_class$Site_medio[i]))
            break

    }
```

```
}
```

Agora converte-se as listas em data frames:

```
aux1 <- data.frame(do.call(rbind, list1))
aux2 <- data.frame(do.call(rbind, list2))
```

O ultimo passo e unir os dois dataframes gerados em um novo, dar nome as variaveis, e adicionar os resultados aos dados:

```
aux3 <- cbind(aux1, aux2)
names(aux3) <- c("Intervalo", "Categoria", "Categoria_")

dados_class <- cbind(dados_class, aux3)

head(dados_class)
```

```
##   Parcela Idade   HD    N    V    B    Site Site_medio Intervalo Categoria
## 1      24  30.0 13.5 1040 24.3  6.0 18.9376    20.1357  23.38947         1
## 2      24  40.8 17.5 1040 54.8  8.9 20.7390    20.1357  23.38947         1
## 3      24  52.8 19.0 1040 76.6 10.9 20.2425    20.1357  23.38947         1
## 4      24  64.8 20.7 1040 98.2 12.1 20.6238    20.1357  23.38947         1
## 5      25  28.8 16.1 1080 25.5  6.1 23.1924    20.3347  23.38947         1
## 6      25  39.6 17.3 1060 51.8  8.8 20.7951    20.3347  23.38947         1
##   Categoria_
## 1   Inferior
## 2   Inferior
## 3   Inferior
## 4   Inferior
## 5   Inferior
## 6   Inferior
```

7) Estimar B1, B2, V2, ICM, IMM e ITC

O primeiro passo e estimar o site medio e a area basal media para cada categoria (caso seja utilizado o passo 7.2):

```
tab_site_medio <- dados_class %>% group_by(Categoria_) %>%
    summarise(Site = mean(Site_medio), B_MEDIO = mean(B))
tab_site_medio
```

```
## # A tibble: 3 × 3
##   Categoria_     Site   B_MEDIO
##       <fctr>    <dbl>     <dbl>
## 1   Inferior 21.39981  9.389744
## 2      Media 25.25361 13.501786
## 3   Superior 28.01135 18.804545
```

Utilizando merge, adiciona um vetor que vai de 20:125, representando a idade, para cada site medio; Em seguida, organiza-se os dados de acordo com a categora:

```
dados_est_ <- merge(tab_site_medio, data.frame(Idade = 20:125)) %>%
    arrange(Categoria_)
head(dados_est_)
```

```
##   Categoria_     Site  B_MEDIO Idade
## 1   Inferior 21.39981 9.389744    20
## 2   Inferior 21.39981 9.389744    21
## 3   Inferior 21.39981 9.389744    22
## 4   Inferior 21.39981 9.389744    23
## 5   Inferior 21.39981 9.389744    24
## 6   Inferior 21.39981 9.389744    25
```

7.1) Definicao da area basal 1 e 2 - b1 modelo

Para se estimar a area basal utilizando a equacao do sistema de Clutter, precisa-se de um valor de area basal inicial, ja que na equacao utiliza-se de B1 e B2 nos calculos. Entao primeiro estima-se uma area basal com um modelo baseado apenas no site, e dai pra frente se estima utilizando a equacao de Clutter. Primeiro cria-se uma tabela com os coeficientes para se estimar o primeiro valor de B2. Utiliza-se os dados originais:

```
dados$Site_quad <- dados$Site^2
reg_B2_inicial <- lm(B ~ Site + Site_quad, dados)
summary(reg_B2_inicial)
```

```
##
## Call:
## lm(formula = B ~ Site + Site_quad, data = dados)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.559  -2.482   0.028   2.918   9.503
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.66045   21.81648   0.993   0.3225
## Site        -1.94522    1.78635  -1.089   0.2781
## Site_quad    0.06464    0.03623   1.784   0.0766 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.001 on 136 degrees of freedom
## Multiple R-squared:  0.4554, Adjusted R-squared:  0.4473
## F-statistic: 56.85 on 2 and 136 DF,  p-value: < 2.2e-16
```

```
tab_coef_B2 <- data.frame(b0 = coef(reg_B2_inicial)[[1]],
    b1 = coef(reg_B2_inicial)[[2]], b2 = coef(reg_B2_inicial)[[3]])
```

O processo de estimacao sera feito utilizando um loop for. Sendo assim, o primeiro passo e criar uma lista fora do loop:

```
list2 <- vector("list", length = nrow(dados_est_))
```

Em seguida, estima-se o primeiro valor de area basal, utilizando a equcao que se baseia no site:

```
list2[[1]] <- log(tab_coef_B2$b0 + tab_coef_B2$b1 * dados_est_$Site[1] +
    tab_coef_B2$b2 * (dados_est_$Site[1]^2))
```

e feito um log do resultado, pois no modelo de Clutter a variavel utilizada e Ln(B1).

Agora ja e possivel estimar utilizando o modelo Clutter, dentro do loop. Existem 3 condicionais no loop:

1. Primeiro: Inserir NA caso o proximo dado nao exista, ou quando se trocar de classe. Isso evita que os dados de classes diferentes nao sejam misturados na hora do calculo.
2. A primeira area basal de cada classe deve ser calculada utilizando o modelo baseado no site, para isso utiliza-se a segunda condicao. Isso e verificado checando se a elemento anterior e NA; Se ele for NA, quer dizer que houve troca de Categoria, ou seja, este e o primeiro dado da Categoria, portanto, deve-se fazer o calculo com esse modelo. Caso contrario, ou seja, se os dados forem da mesma categoria, realizar o calculo utilizando o modelo de Clutter para Area Basal.

```
for (i in 2:nrow(dados_est_)) {


    if (is.na(dados_est_$Categoria_[i + 1]) | dados_est_$Categoria_[i] !=
        dados_est_$Categoria_[i + 1]) {

        list2[[i]] <- NA


    } else if (is.na(list2[[i - 1]])) {

        list2[[i]] <- log(tab_coef_B2$b0 + tab_coef_B2$b1 *
            dados_est_$Site[i] + tab_coef_B2$b2 * (dados_est_$Site[i]^2))

    } else {

        list2[[i]] <- list2[[i - 1]] * (dados_est_$Idade[i]/dados_est_$Idade[i +
            1]) + tab_coef_clut$a0 * (1 - (dados_est_$Idade[i]/dados_est_$Idade[i +
            1])) + tab_coef_clut$a1 * (1 - (dados_est_$Idade[i]/dados_est_$Idade[i +
            1])) * dados_est_$Site[i]



    }

}
```

Agora converte-se a lista em um vetor, e salva-se o vetor como uma variavel no dataframe:

```
dados_est_$LN_B2_EST <- as.vector(do.call(rbind, list2))
head(dados_est_)
```

```
##   Categoria_     Site  B_MEDIO Idade LN_B2_EST
## 1   Inferior 21.39981 9.389744    20  2.265323
## 2   Inferior 21.39981 9.389744    21  2.302726
## 3   Inferior 21.39981 9.389744    22  2.336877
## 4   Inferior 21.39981 9.389744    23  2.368183
## 5   Inferior 21.39981 9.389744    24  2.396983
## 6   Inferior 21.39981 9.389744    25  2.423569
```

7.2) Definicao da area basal 1 e 2 - B1 media

Para se estimar a area basal utilizando a equacao do sistema de Clutter, precisa-se de um valor de area basal inicial, ja que na equacao utiliza-se de B1 e B2 nos calculos. Neste caso sera utilizada a area basal media calculada no passo 7: O processo de classificacao sera feito utilizando um loop for. Sendo assim, o primeiro passo e criar uma lista fora do loop:

```
list2 <- vector("list", length = nrow(dados_est_))
```

Em seguida, insere-se o primeiro valor de area basal, utilizando a area basal media. como ela se repete ao longo de toda a classe, qualquer linha (desde que seja daquela classe) tera o seu valor. No caso incia-se com a classe baixa, entao a primeira linha ira conter o seu valor:

```
head(dados_est_)
```

```
##   Categoria_     Site  B_MEDIO Idade LN_B2_EST
## 1   Inferior 21.39981 9.389744    20  2.265323
## 2   Inferior 21.39981 9.389744    21  2.302726
## 3   Inferior 21.39981 9.389744    22  2.336877
## 4   Inferior 21.39981 9.389744    23  2.368183
## 5   Inferior 21.39981 9.389744    24  2.396983
## 6   Inferior 21.39981 9.389744    25  2.423569
```

```
list2[[1]] <- log(dados_est_$B_MEDIO[1])
```

e feito um log do resultado, pois no modelo de Clutter a variavel utilizada e Ln(B1).

Agora ja e possivel estimar utilizando o modelo Clutter, dentro do loop. Existem 3 condicionais no loop:

1. Primeiro: Inserir NA caso o proximo dado nao exista, ou quando se trocar de classe. Isso evita que os dados de classes diferentes nao sejam misturados na hora do calculo.
2. A primeira area basal de cada classe deve ser calculada utilizando o modelo baseado no site, para isso utiliza-se a segunda condicao. Isso e verificado checando se a elemento anterior e NA; Se ele for NA, quer dizer que houve troca de Categoria, ou seja, este e o primeiro dado da Categoria, portanto, deve-se utilizar a area basal media daquela classe. Caso contrario, ou seja, se os dados forem da mesma categoria, realizar o calculo utilizando o modelo de Clutter para Area Basal.

```
for (i in 2:nrow(dados_est_)) {


    if (is.na(dados_est_$Categoria_[i + 1]) | dados_est_$Categoria_[i] !=
        dados_est_$Categoria_[i + 1]) {

        list2[[i]] <- NA


    } else if (is.na(list2[[i - 1]])) {

        list2[[i]] <- log(dados_est_$B_MEDIO[i])

    } else {

        list2[[i]] <- list2[[i - 1]] * (dados_est_$Idade[i]/dados_est_$Idade[i +
            1]) + tab_coef_clut$a0 * (1 - (dados_est_$Idade[i]/dados_est_$Idade[i +
            1])) + tab_coef_clut$a1 * (1 - (dados_est_$Idade[i]/dados_est_$Idade[i +
            1])) * dados_est_$Site[i]

    }

}
```

Agora converte-se a lista em um vetor, e salva-se o vetor como uma variavel no dataframe:

```
dados_est_$LN_B2_EST <- as.vector(do.call(rbind, list2))
head(dados_est_)
```

```
##   Categoria_     Site  B_MEDIO Idade LN_B2_EST
## 1   Inferior 21.39981 9.389744    20  2.239618
## 2   Inferior 21.39981 9.389744    21  2.278190
## 3   Inferior 21.39981 9.389744    22  2.313408
## 4   Inferior 21.39981 9.389744    23  2.345691
## 5   Inferior 21.39981 9.389744    24  2.375391
## 6   Inferior 21.39981 9.389744    25  2.402807
```

7.3) Estimacao de B2, V2, ICM e IMM

No proximo passo remover-se os NAs, adiciona-se os coeficientes das equacoes de Clutter, e estima-se o volume, seguido do incremento corrente mensal e incremento medio mensal:

```
dados_est <- dados_est_ %>% cbind(tab_coef_clut) %>% group_by(Categoria_) %>%
    mutate(B2_EST = exp(LN_B2_EST), V2_EST = exp(b0 + (b1 *
        1/Idade) + b2 * Site + b3 * LN_B2_EST), ICM = abs(V2_EST -
        lag(V2_EST)), IMM = V2_EST/Idade, ICM_IMM = ICM -
        IMM)
dados_est
```

```
## Source: local data frame [318 x 16]
## Groups: Categoria_ [3]
##
##    Categoria_     Site  B_MEDIO Idade LN_B2_EST       b0        b1
##        <fctr>    <dbl>    <dbl> <int>     <dbl>    <dbl>     <dbl>
## 1    Inferior 21.39981 9.389744    20  2.239618 1.261075 -22.17197
## 2    Inferior 21.39981 9.389744    21  2.278190 1.261075 -22.17197
## 3    Inferior 21.39981 9.389744    22  2.313408 1.261075 -22.17197
## 4    Inferior 21.39981 9.389744    23  2.345691 1.261075 -22.17197
## 5    Inferior 21.39981 9.389744    24  2.375391 1.261075 -22.17197
## 6    Inferior 21.39981 9.389744    25  2.402807 1.261075 -22.17197
## 7    Inferior 21.39981 9.389744    26  2.428192 1.261075 -22.17197
## 8    Inferior 21.39981 9.389744    27  2.451764 1.261075 -22.17197
## 9    Inferior 21.39981 9.389744    28  2.473710 1.261075 -22.17197
## 10   Inferior 21.39981 9.389744    29  2.494193 1.261075 -22.17197
##            b2       b3       a0         a1    B2_EST   V2_EST      ICM
##         <dbl>    <dbl>    <dbl>      <dbl>     <dbl>    <dbl>    <dbl>
## 1  0.03036532 1.226243 1.699198 0.06490727  9.389744 34.76479       NA
## 2  0.03036532 1.226243 1.699198 0.06490727  9.759000 38.42446 3.659665
## 3  0.03036532 1.226243 1.699198 0.06490727 10.108815 42.09257 3.668108
## 4  0.03036532 1.226243 1.699198 0.06490727 10.440483 45.75386 3.661293
## 5  0.03036532 1.226243 1.699198 0.06490727 10.755220 49.39567 3.641806
## 6  0.03036532 1.226243 1.699198 0.06490727 11.054162 53.00755 3.611884
## 7  0.03036532 1.226243 1.699198 0.06490727 11.338363 56.58100 3.573449
## 8  0.03036532 1.226243 1.699198 0.06490727 11.608803 60.10914 3.528142
## 9  0.03036532 1.226243 1.699198 0.06490727 11.866387 63.58650 3.477360
## 10 0.03036532 1.226243 1.699198 0.06490727 12.111953 67.00879 3.422286
##          IMM  ICM_IMM
##       <dbl>    <dbl>
## 1  1.738240       NA
## 2  1.829736 1.829929
```

```
## 3  1.913298 1.754809
## 4  1.989298 1.671995
## 5  2.058153 1.583653
## 6  2.120302 1.491582
## 7  2.176192 1.397256
## 8  2.226264 1.301877
## 9  2.270946 1.206414
## 10 2.310648 1.111639
## # ... with 308 more rows
```

7.4) Idade Tecnica de Corte e tabela de resultados

Para se encontrar a idade tecnica de corte, basta fazer o seguinte:

```
dados_est %>% group_by(Categoria_) %>% filter(round(ICM,
    1) == round(IMM, 1)) %>% select(ITC = Idade, ITC_Y = IMM) %>%
    summarise_all(mean)
```

```
## Adding missing grouping variables: `Categoria_`
```

```
## # A tibble: 3 × 3
##   Categoria_   ITC    ITC_Y
##       <fctr> <dbl>    <dbl>
## 1   Inferior    44 2.519539
## 2      Media    41 4.121480
## 3   Superior    37 6.167924
```

Adiciona-se a ITC aos dados:

```
dados_est <- dados_est %>% filter(round(ICM, 1) == round(IMM,
    1)) %>% select(Categoria_, ITC = Idade, ITC_Y = IMM) %>%
    summarise_all(mean) %>% left_join(dados_est, by = "Categoria_")
dados_est
```

```
## # A tibble: 318 × 18
##    Categoria_   ITC    ITC_Y     Site  B_MEDIO Idade LN_B2_EST       b0
##        <fctr> <dbl>    <dbl>    <dbl>    <dbl> <int>     <dbl>    <dbl>
## 1    Inferior    44 2.519539 21.39981 9.389744    20  2.239618 1.261075
## 2    Inferior    44 2.519539 21.39981 9.389744    21  2.278190 1.261075
## 3    Inferior    44 2.519539 21.39981 9.389744    22  2.313408 1.261075
## 4    Inferior    44 2.519539 21.39981 9.389744    23  2.345691 1.261075
## 5    Inferior    44 2.519539 21.39981 9.389744    24  2.375391 1.261075
## 6    Inferior    44 2.519539 21.39981 9.389744    25  2.402807 1.261075
## 7    Inferior    44 2.519539 21.39981 9.389744    26  2.428192 1.261075
## 8    Inferior    44 2.519539 21.39981 9.389744    27  2.451764 1.261075
## 9    Inferior    44 2.519539 21.39981 9.389744    28  2.473710 1.261075
## 10   Inferior    44 2.519539 21.39981 9.389744    29  2.494193 1.261075
##           b1         b2       b3       a0         a1    B2_EST   V2_EST
##        <dbl>      <dbl>    <dbl>    <dbl>      <dbl>     <dbl>    <dbl>
## 1  -22.17197 0.03036532 1.226243 1.699198 0.06490727  9.389744 34.76479
## 2  -22.17197 0.03036532 1.226243 1.699198 0.06490727  9.759000 38.42446
## 3  -22.17197 0.03036532 1.226243 1.699198 0.06490727 10.108815 42.09257
## 4  -22.17197 0.03036532 1.226243 1.699198 0.06490727 10.440483 45.75386
## 5  -22.17197 0.03036532 1.226243 1.699198 0.06490727 10.755220 49.39567
## 6  -22.17197 0.03036532 1.226243 1.699198 0.06490727 11.054162 53.00755
## 7  -22.17197 0.03036532 1.226243 1.699198 0.06490727 11.338363 56.58100
```

```
## 8  -22.17197 0.03036532 1.226243 1.699198 0.06490727 11.608803 60.10914
## 9  -22.17197 0.03036532 1.226243 1.699198 0.06490727 11.866387 63.58650
## 10 -22.17197 0.03036532 1.226243 1.699198 0.06490727 12.111953 67.00879
##          ICM      IMM  ICM_IMM
##        <dbl>    <dbl>    <dbl>
## 1        NA 1.738240       NA
## 2  3.659665 1.829736 1.829929
## 3  3.668108 1.913298 1.754809
## 4  3.661293 1.989298 1.671995
## 5  3.641806 2.058153 1.583653
## 6  3.611884 2.120302 1.491582
## 7  3.573449 2.176192 1.397256
## 8  3.528142 2.226264 1.301877
## 9  3.477360 2.270946 1.206414
## 10 3.422286 2.310648 1.111639
## # ... with 308 more rows
```

Agora gera-se tabela final com informacoes por classe de area basal, idade tecnica de corte, site, e volume total:

```
B2_Inicial <- dados_est %>% group_by(Categoria_) %>% filter(row_number() ==
    1) %>% select(Categoria_, B2_Inicial = B2_EST)

ITC <- dados_est %>% group_by(Categoria_) %>% filter(round(ICM,
    1) == round(IMM, 1)) %>% summarise(ITC = mean(Idade),
    Site = mean(Site))

sum_V <- dados_est %>% group_by(Categoria_) %>% summarise(V_total = sum(V2_EST,
    na.rm = T))
```

Apos gerar-se as variaveis, basta junta-las com left_join:

```
tab_final <- left_join(B2_Inicial, ITC) %>% left_join(sum_V)
```

```
## Joining, by = "Categoria_"
## Joining, by = "Categoria_"
```

```
tab_final
```

```
## Source: local data frame [3 x 5]
## Groups: Categoria_ [?]
##
##   Categoria_ B2_Inicial   ITC     Site  V_total
##       <fctr>      <dbl> <dbl>    <dbl>    <dbl>
## 1   Inferior   9.389744    44 21.39981 15667.04
## 2      Media  13.501786    41 25.25361 24949.62
## 3   Superior  18.804545    37 28.01135 35756.52
```

8) Graficos de Incremento Corrente Mensal e Incremento Medio Mensal

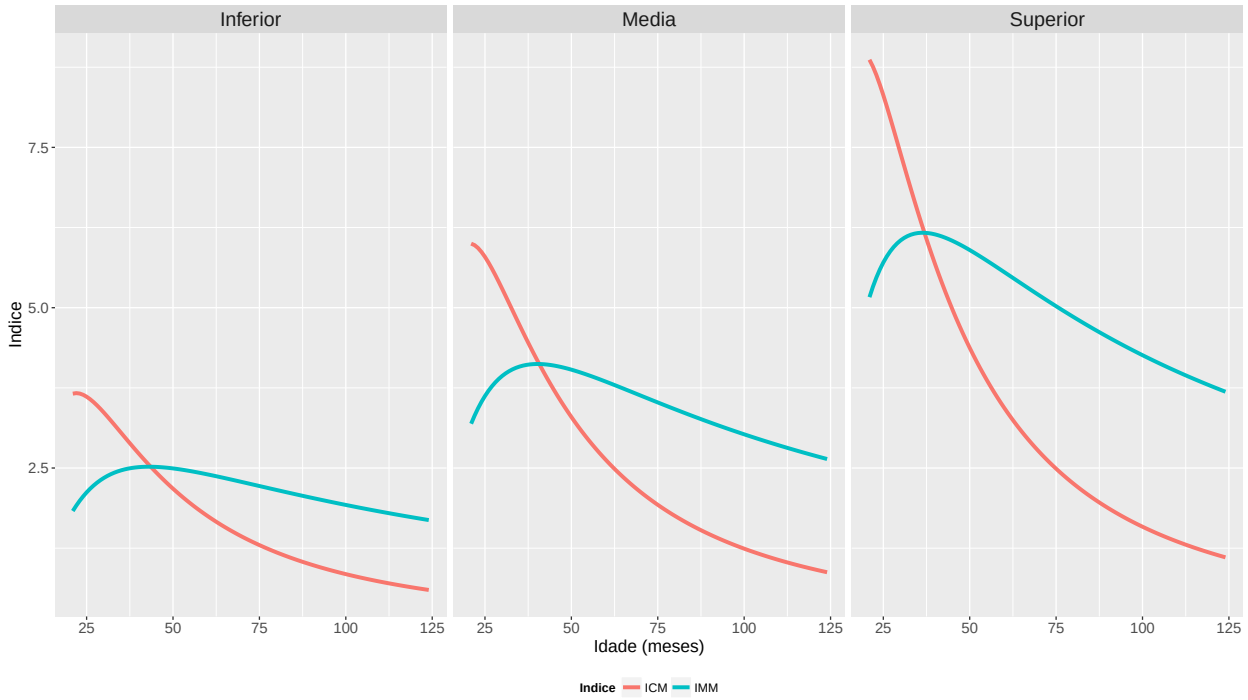
Com os valores de ICM e IMM obtidos, e necessario te-los na mesma coluna, sendo identificados por um fator, para que se possa fazer os graficos de linha. Para isso utiliza-se a funcao gather:

```
dados_graph <- dados_est %>% na.omit() %>% select(Categoria_,
    ICM, IMM, Idade, ITC, ITC_Y) %>% gather(Indice, Valor,
    ICM, IMM)
dados_graph
```

```
## # A tibble: 624 × 6
##    Categoria_ Idade   ITC    ITC_Y Indice    Valor
##        <fctr> <int> <dbl>    <dbl>  <chr>    <dbl>
## 1    Inferior    21    44 2.519539    ICM 3.659665
## 2    Inferior    22    44 2.519539    ICM 3.668108
## 3    Inferior    23    44 2.519539    ICM 3.661293
## 4    Inferior    24    44 2.519539    ICM 3.641806
## 5    Inferior    25    44 2.519539    ICM 3.611884
## 6    Inferior    26    44 2.519539    ICM 3.573449
## 7    Inferior    27    44 2.519539    ICM 3.528142
## 8    Inferior    28    44 2.519539    ICM 3.477360
## 9    Inferior    29    44 2.519539    ICM 3.422286
## 10   Inferior    30    44 2.519539    ICM 3.363920
## # ... with 614 more rows
```

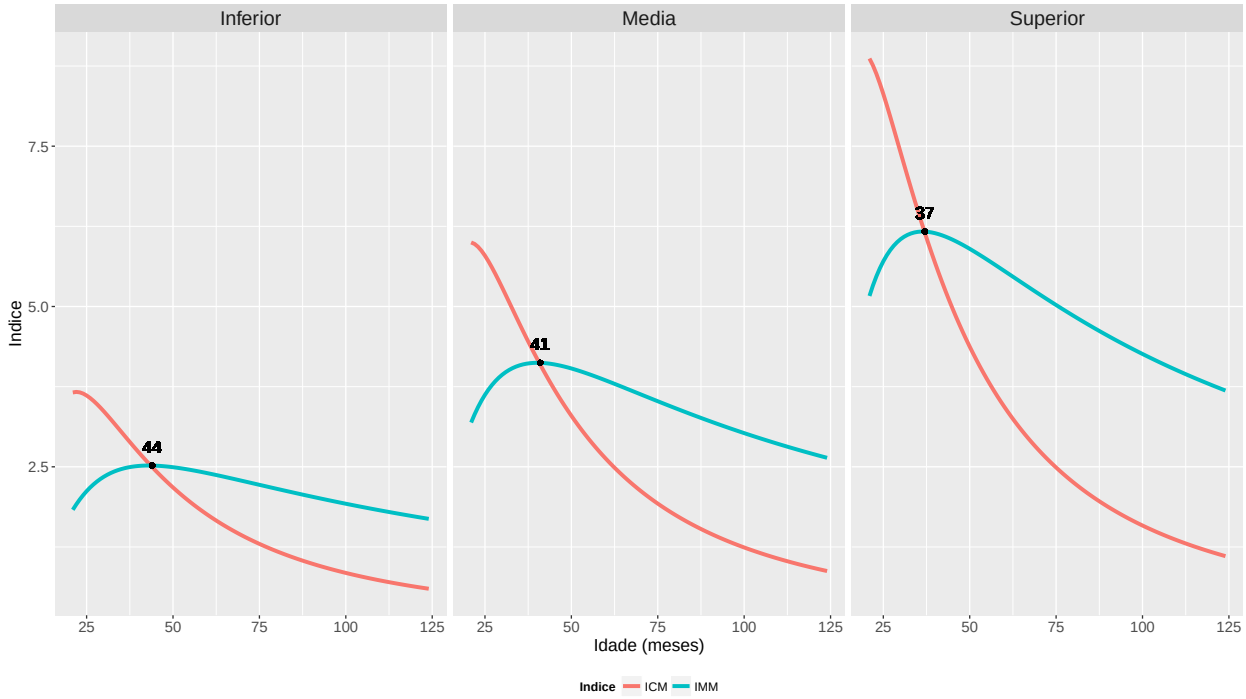
Agora, plota-se o grafico com ggplot, utilizando a variavel Indice como cor, para que se diferencie os indices. Alem disso, utiliza-se facet_wrap para gerar um grafico para cada classe:

```
graph <- ggplot(dados_graph, aes(x = Idade, y = Valor, color = Indice)) +
    facet_wrap(~Categoria_) + geom_line(size = 1.5) + labs(x = "Idade (meses)",
    y = "Indice", color = "Indice") + theme(legend.position = "bottom",
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 10), axis.title = element_text(size = 14),
    axis.text = element_text(size = 12), strip.text.x = element_text(size = 16))
graph
```



Pode-se adicionar a idade tecnica de corte ao grafico:

```
graph2 <- ggplot(dados_graph, aes(x = Idade, color = Indice)) +
    facet_wrap(~Categoria_) + geom_line(aes(y = Valor),
    size = 1.5) + geom_point(aes(x = ITC, y = ITC_Y), color = "black") +
    geom_text(aes(x = ITC, y = ITC_Y, label = ITC), vjust = 0,
        nudge_y = 0.2, color = "black", size = 5) + labs(x = "Idade (meses)",
    y = "Indice", color = "Indice") + theme(legend.position = "bottom",
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 10), axis.title = element_text(size = 14),
    axis.text = element_text(size = 12), strip.text.x = element_text(size = 16))
graph2
```



Exporta-se o grafico com ggsave:

```
ggsave("graph_itc.png", graph, width = 14, height = 8)
ggsave("graph_itc2.png", graph2, width = 14, height = 8)
```