SaeNer-Anexo 3.3: Obtención de Versiones adicionales

-

Los siguientes códigos tienen como fin ejemplificar el uso de las funciones construidas para nuestro SAE-NER, no se proveen bases de datos para el mismo.

[ ]:
```
library(data.table)
    setDTthreads(percent = 100)
library(stringr)
library(magrittr)

#Para paralelizacion
library(doSNOW)

#para lmm
library(lme4)
```

1 Código necesario

[ ]:
```
to_formula_lmm<-function(y,X,dom){
  .X = paste(X, collapse = '+')
  .f = paste0(y,'~',.X, ' + (1|', dom, ')')
  .f = .f %>% as.formula()
  return(.f)
}
```

2 Del mejor modelo, se generan 4 versiones adicionales

Dado que el primero y segundo mejor resultado no se diferencian en Coeficiente de Variación de la estimación EPPC, se utilizará el que presente menor cantidad de variables independientes.

[ ]:
```
#Vector de characters con grupo de variables del mejor modelo
Xs_MejorModelo = Xs_MejorModelo
```

[ ]:
```
#Se carga información muestral
dat = readRDS("dat.rds")
```

2.1 Modelo v2: ningún p-valor mayor a 0.1

[ ]:
```
mx0 = lme4::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)
mx = lmerTest::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)
```

[ ]:
```
mx0_dt = mx0  %>% summary  %>% .[["coefficients"]]  %>% data.frame
mx0_dt[,"x_var"] = mx0_dt  %>% rownames
setDT(mx0_dt)
setnames(mx0_dt,"Estimate", "beta")
mx0_dt = mx0_dt[, .(x_var, beta)]
mx0_dt[, signo := sign(beta)]
```

[ ]:
```
mx_dat = mx  %>% summary  %>% .[["coefficients"]]
mx_dat  %<>% data.frame
mx_dat[, "x_var"] = (mx_dat  %>% rownames)
setDT(mx_dat)
setnames(mx_dat, 'Pr...t..', 'p_value')
mx_dat[,p_value:= round(p_value, 6)]

mx_dat_f = copy(mx_dat)
mx_dat_f = mx_dat_f[order(-p_value)]
Xs_MejorModelo_vx = Xs_MejorModelo
total_no_sig = 1
while (total_no_sig !=0 ){
    total_no_sig = (mx_dat_f[, p_value] > 0.1)  %>% sum(., na.rm = T)
    print(total_no_sig)
    mx_dat_f = mx_dat_f[order(-p_value)]

    Xs_MejorModelo_vx = Xs_MejorModelo_vx  %>% str_subset(mx_dat_f[1, x_var], T)
    mx = lmerTest::lmer(to_formula_lmm(y="y",X = Xs_MejorModelo_vx ,dom = "id_dominio"), data = dat)

    mx_dat = mx  %>% summary  %>% .[["coefficients"]]
    mx_dat  %<>% data.frame
    mx_dat[, "x_var"] = (mx_dat  %>% rownames)
    setDT(mx_dat)
    setnames(mx_dat, 'Pr...t..', 'p_value')
    mx_dat[,p_value:= round(p_value, 6)]

    mx_dat_f = copy(mx_dat)
    mx_dat_f = mx_dat_f[order(-p_value)]
    mx_dat_f[, signo_x := sign(Estimate)]
    mx_dat_f = merge(x = mx_dat_f, y = mx0_dt,
                     by = "x_var", all = T)
    mx_dat_f[, signo_dif := signo_x!= signo ]
    tot_signo_dif = (mx_dat_f[, signo_dif]  %>% sum(., na.rm = T))
    if (tot_signo_dif!=0){ total_no_sig = 0 }
}
```

[ ]:
```
#Información para modelo v2
mx_v2_muestra <<- mx_dat_f[order(-p_value)][!is.na(p_value)]
```

2.2 Modelo v3: ningún p-valor mayor a 0.05

[ ]:
```
Xs_MejorModelo = res_muestra[nombre_modelo_v0 == nombre_mejor_modelo,todas_xvars]
Xs_MejorModelo = Xs_MejorModelo  %>% str_split("~")  %>% unlist

mx0 = lme4::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)
mx = lmerTest::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)

mx0_dt = mx0  %>% summary  %>% .[["coefficients"]]  %>% data.frame
mx0_dt[,"x_var"] = mx0_dt  %>% rownames
setDT(mx0_dt)
setnames(mx0_dt,"Estimate", "beta")
mx0_dt = mx0_dt[, .(x_var, beta)]
mx0_dt[, signo := sign(beta)]

mx_dat = mx  %>% summary  %>% .[["coefficients"]]
mx_dat  %<>% data.frame
mx_dat[, "x_var"] = (mx_dat  %>% rownames)
setDT(mx_dat)
```

```
setnames(mx_dat, 'Pr...t..', 'p_value')
mx_dat[,p_value:= round(p_value, 6)]

mx_dat_f = copy(mx_dat)
mx_dat_f = mx_dat_f[order(-p_value)]
Xs_MejorModelo_vx = Xs_MejorModelo
total_no_sig = 1
while (total_no_sig !=0 ){
    total_no_sig = (mx_dat_f[, p_value] > 0.05)  %>% sum(., na.rm = T)
    print(total_no_sig)
    mx_dat_f = mx_dat_f[order(-p_value)]

    Xs_MejorModelo_vx = Xs_MejorModelo_vx  %>% str_subset(mx_dat_f[1, x_var], T)
    mx = lmerTest::lmer(to_formula_lmm(y="y",X = Xs_MejorModelo_vx ,dom = "id_dominio"), data = dat)

    mx_dat = mx  %>% summary  %>% .[["coefficients"]]
    mx_dat  %<>% data.frame
    mx_dat[, "x_var"] = (mx_dat  %>% rownames)
    setDT(mx_dat)
    setnames(mx_dat, 'Pr...t..', 'p_value')
    mx_dat[,p_value:= round(p_value, 6)]

    mx_dat_f = copy(mx_dat)
    mx_dat_f = mx_dat_f[order(-p_value)]
    mx_dat_f[, signo_x := sign(Estimate)]
    mx_dat_f = merge(x = mx_dat_f, y = mx0_dt,
                     by = "x_var", all = T)
    mx_dat_f[, signo_dif := signo_x!= signo ]
    tot_signo_dif = (mx_dat_f[, signo_dif]  %>% sum(., na.rm = T))
    if (tot_signo_dif!=0){ total_no_sig = 0 }
}
```

[ ]:
```
#Información para modelo v3
mx_v3_muestra <<- mx_dat_f[order(-p_value)][!is.na(p_value)]
```

2.3 Modelo v4: signos contraintuitivos

[ ]:
```
Xs_MejorModelo = res_muestra[nombre_modelo_v0 == nombre_mejor_modelo,todas_xvars]
Xs_MejorModelo = Xs_MejorModelo  %>% str_split("~")  %>% unlist
```

[ ]:
```
signo_esperado = data.table("x_var" = Xs_MejorModelo)
```

[ ]:
```
for (.x in Xs_MejorModelo){
    .x.res = dat[, weighted.mean(dcronica, fexp), by = .x]
    .x.res = .x.res[order(-get(.x))]
    nrow.x = .x.res  %>% nrow
    potential_beta = .x.res[1, V1] - .x.res[nrow(.x.res), V1]
    if (nrow.x>5){
        potential_beta = cor(dat[, dcronica], dat[, get( .x)])
    }
    signo_esperado[x_var == .x, beta_potencial := potential_beta]
}
signo_esperado[, signo_pot := sign(beta_potencial)]
```

[ ]:
```
mx0 = lme4::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)
mx = lmerTest::lmer(to_formula_lmm(y="y",X =Xs_MejorModelo ,dom = "id_dominio"), data = dat)

mx0_dt = mx0  %>% summary  %>% .[["coefficients"]]  %>% data.frame
mx0_dt[,"x_var"] = mx0_dt  %>% rownames
setDT(mx0_dt)
setnames(mx0_dt,"Estimate", "beta")
mx0_dt = mx0_dt[, .(x_var, beta)]
mx0_dt[, signo := sign(beta)*-1]
```

[ ]:
```
mx0_dt = merge(x = mx0_dt, y = signo_esperado,
               by = "x_var", all = T)
```

[ ]:
```
dat_VarLab = readRDS("muestra_varlab.rds")
setDT(dat_VarLab)
dat_VarLab[, V1 := as.character(V1)]
dat_VarLab[, V2 := as.character(V2)]

setnames(dat_VarLab, c("V1","V2"), c("x_var","x_VarLab"))
```

[ ]:
```
#Información para modelo v4
mx_v4_muestra <<- mx0_dt[signo!= signo_pot,]
```

[ ]:
```
#Modelos v5 y v6 son derivaciones simples de los modelos v2 a v4
```