Modellazione multilevel

Daniele Zago

2021-04-28

```
library(rstanarm)
library(bayesplot)
library(tidybayes)
library(ggplot2)
library(magrittr)
library(lme4)
library(dplyr)
library(gtools)
library(tidyverse)
```

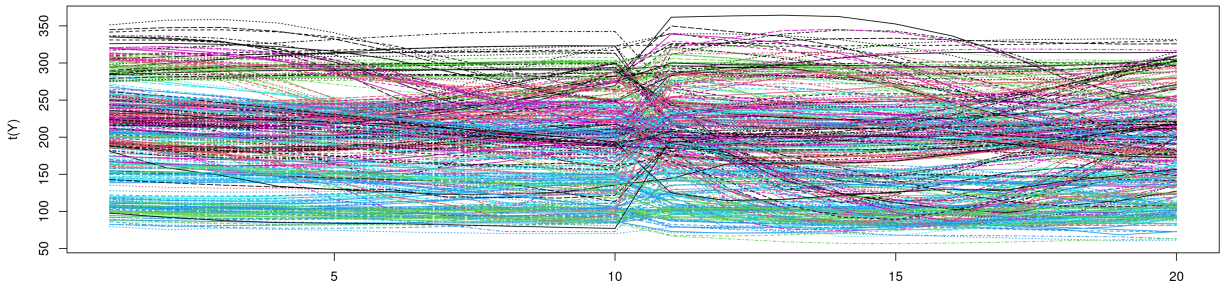
Preprocessing

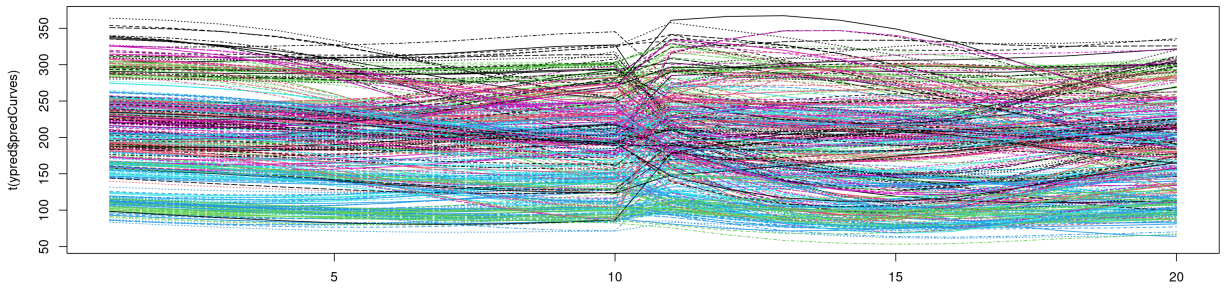
```
mandarino_temp = mandarino[ , -6 ]
mandarino_wide = spread(mandarino_temp, time, "f0")
Y = as.matrix(mandarino_wide[ , 5:(NCOL(mandarino_wide)) ])     # Funzioni osservate
```

Analisi con componenti principali funzionali (fPCA)

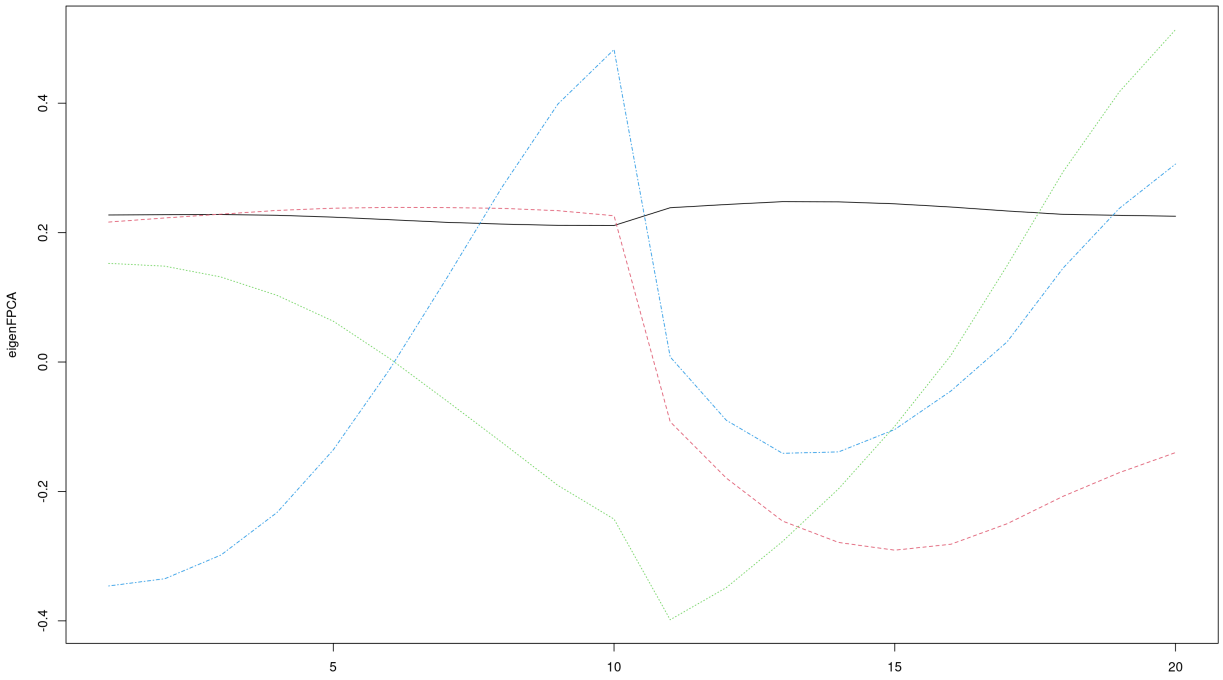
Stima delle funzioni principali che spieghino il 95% della varianza funzionale:

```
library(fdapace)

Y_list = lapply(seq_len(NROW(Y)), function(i) Y[i,])
t_list = lapply(seq_len(NROW(Y)), function(i) 1:20)

fpca = FPCA(Ly = Y_list, Lt = t_list, optns = list(FVEthreshold = 0.99))

ypred = predict(fpca, Y_list, t_list)
{
    # Confronto fPCA univariate con MFPCA
    par(mfrow = c(2,1))
    matplot(t(Y), type = "l")
    matplot(t(ypred$predCurves), type = "l")
    par(mfrow = c(1,1))
}
```






```
K = fpca$selectK
eigenFPCA = fpca$phi                                    # Estimated eigenfunctions

matplot(eigenFPCA, type = "l")
```

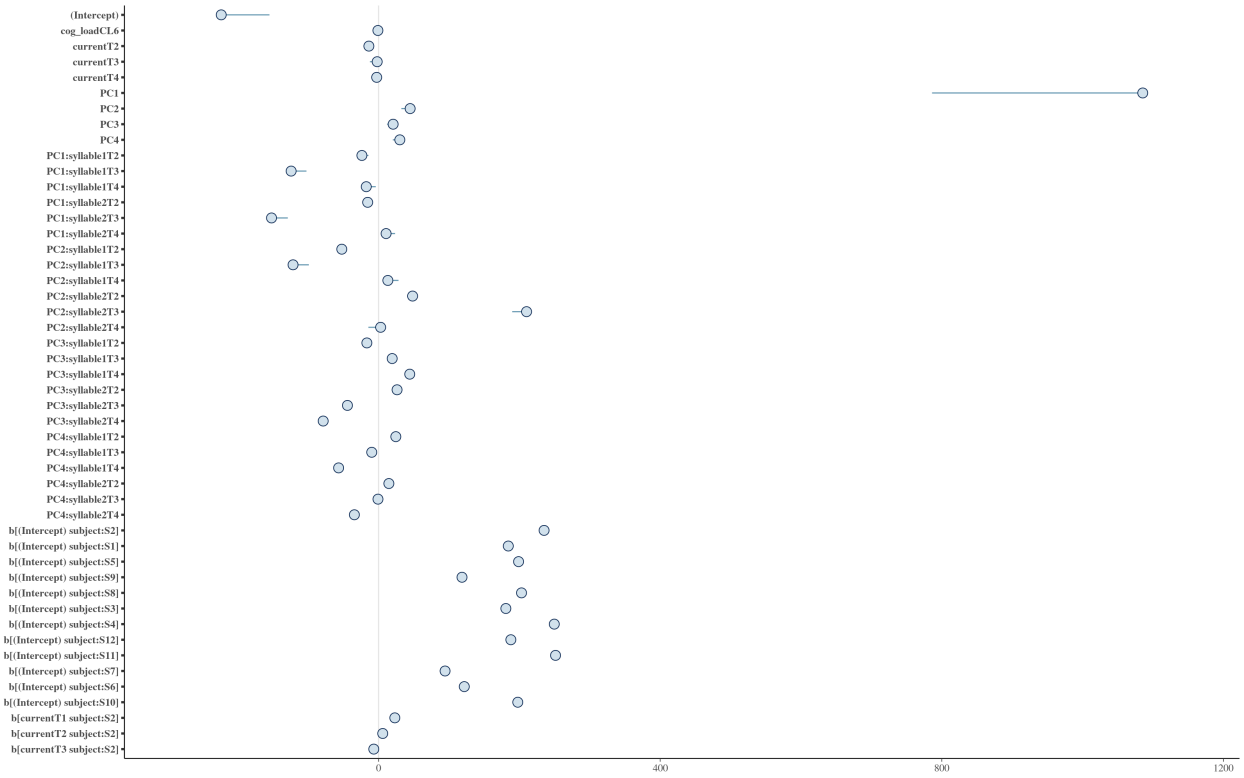



```
# Add eigenfunctions as covariates
eigenColumns = NULL
for(i in 1:NROW(Y)){
    eigenColumns = rbind(eigenColumns, eigenFPCA)
}

mandarino_fpca = cbind(mandarino, eigenColumns)
colnames(mandarino_fpca) = c(colnames(mandarino), paste0("PC", 1:K))
```

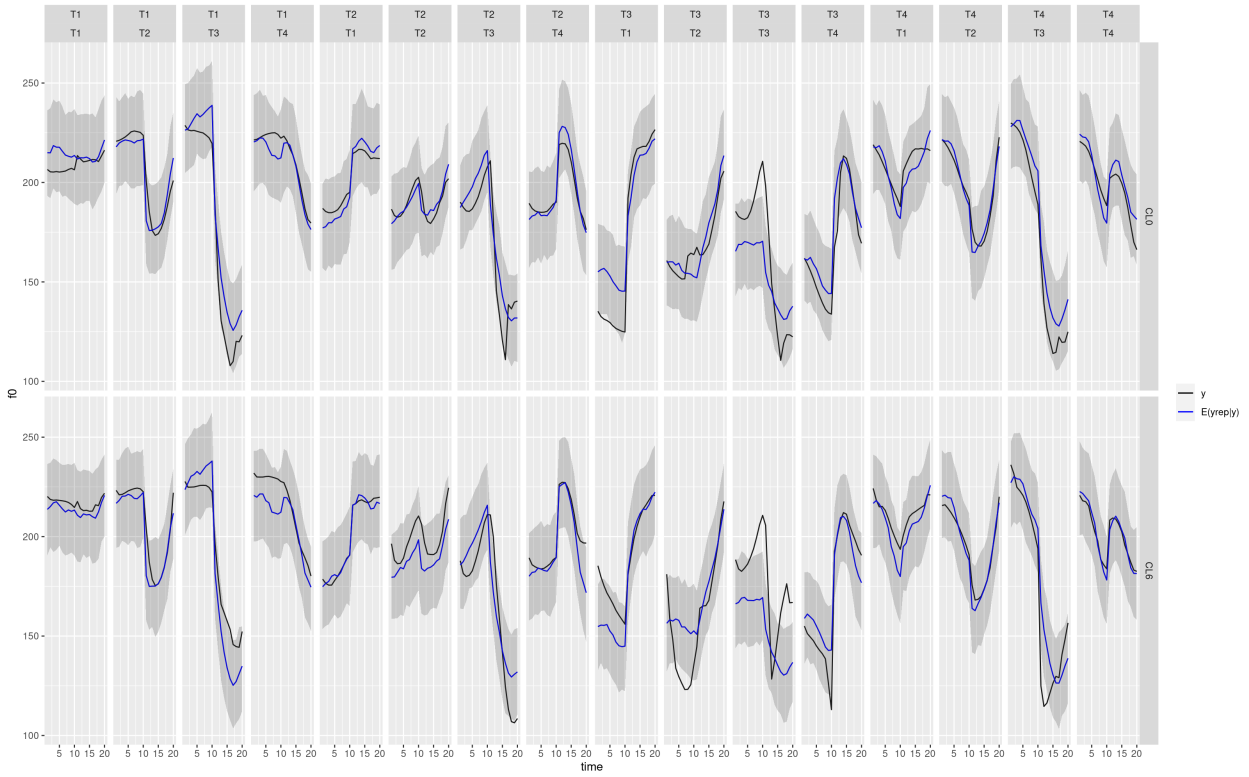
```
if(file.exists("fitStanLmer2.Rdata")){
    load("fitStanLmer2.Rdata")
} else{
    fitStanLmer2 = stan_glmer(f0 ~ cog_load + current +
                              PC1 + PC2 + PC3 + PC4 +
                              PC1:syllable1 + PC1:syllable2 +
                              PC2:syllable1 + PC2:syllable2 +
                              PC3:syllable1 + PC3:syllable2 +
                              PC4:syllable1 + PC4:syllable2 +
                              (1 | subject) +
                              (0 + current|subject) +
                              (0 + PC1 + PC2 + PC3 + PC4|subject) +
                              (0 + PC1:syllable1 | subject) +
                              (0 + PC1:syllable2 | subject) +
                              (0 + PC2:syllable1 | subject) +
                              (0 + PC2:syllable2 | subject) +
                              (0 + PC3:syllable1 | subject) +
                              (0 + PC3:syllable2 | subject) +
                              (0 + PC4:syllable1 | subject) +
                              (0 + PC4:syllable2 | subject)
```

```
                          ,
                     data = mandarino_fpca,
                algorithm = "meanfield",
                prior = hs(),
                iter = 20000,
                QR = TRUE)
    save(fitStanLmer2, file="fitStanLmer2.Rdata")
}
```

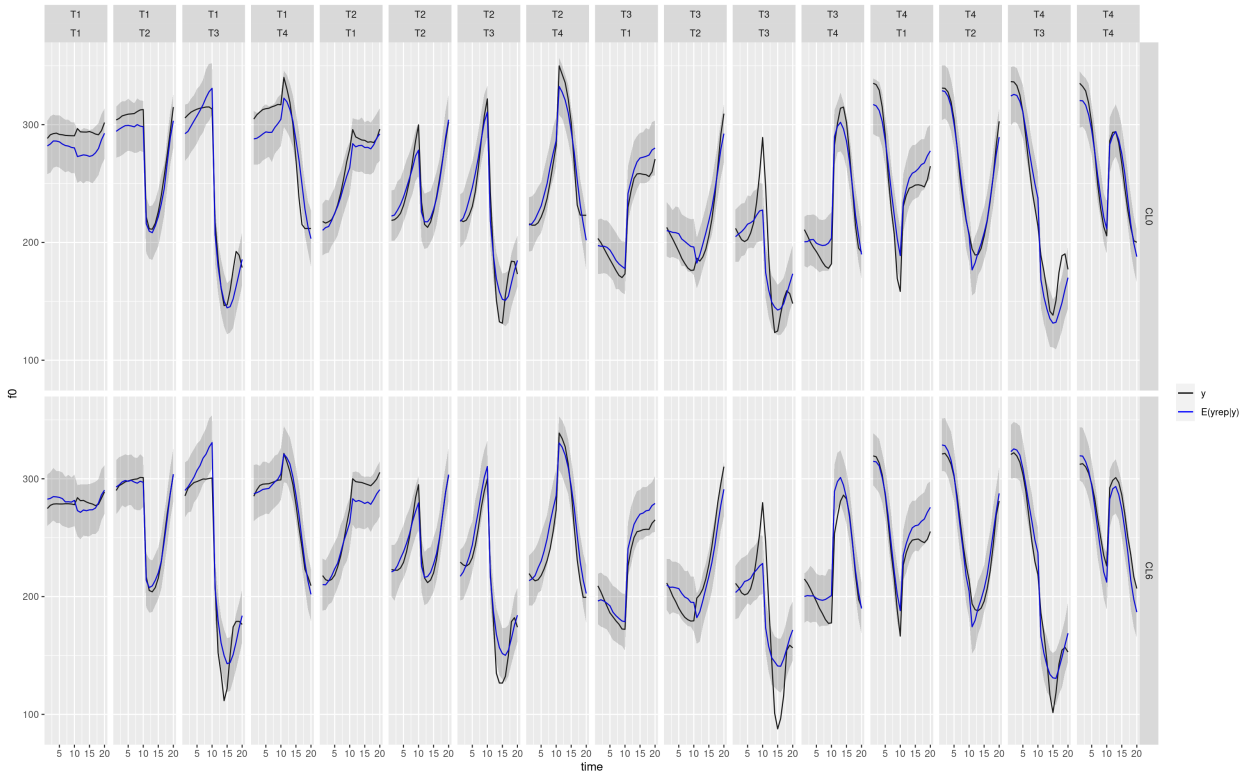
```
plot(fitStanLmer2, pars=names(fitStanLmer2$coefficients)[1:48])
```



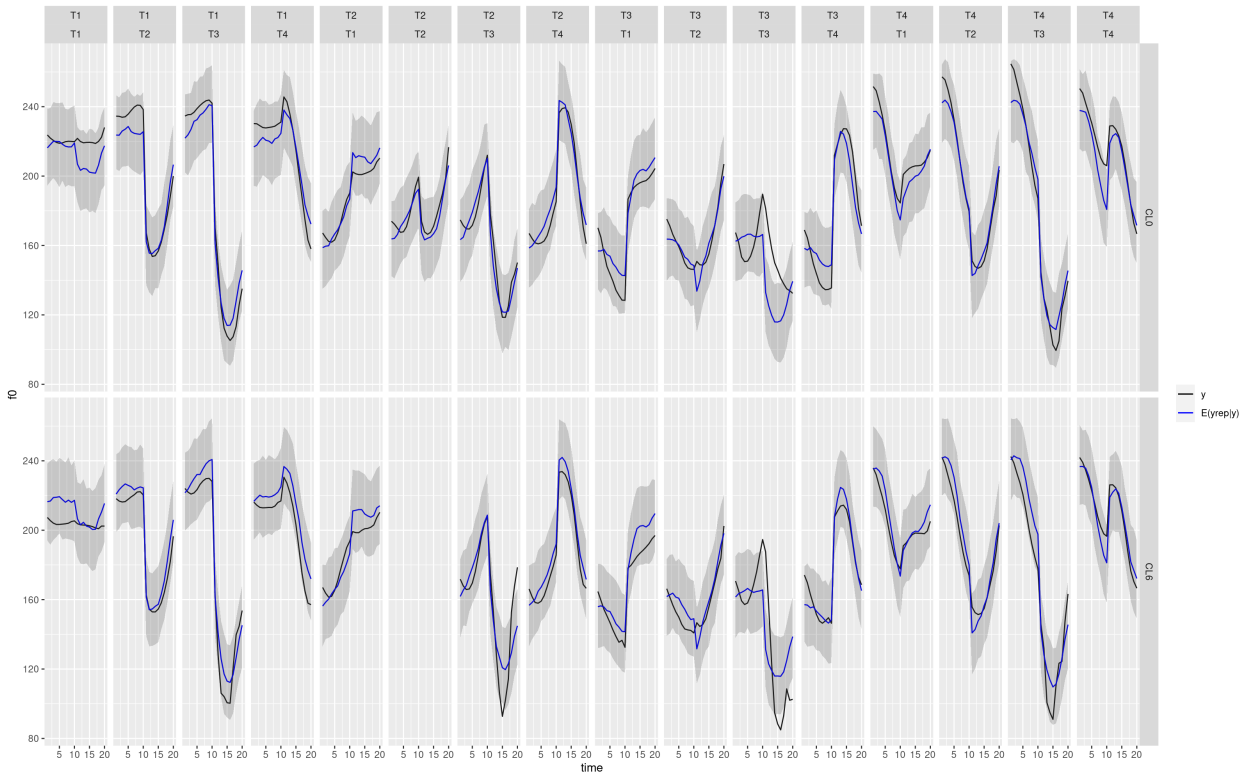

```
ypredStanLmer2 = posterior_predict(fitStanLmer2, draws = 500)
ypred = apply(ypredStanLmer2, 2, mean)
stdev = apply(ypredStanLmer2, 2, sd)
lmin = apply(ypredStanLmer2, 2, function(col) quantile(col, p = 0.05))
lmax = apply(ypredStanLmer2, 2, function(col) quantile(col, p = 0.95))

ppc_subject("S1", mandarino, ypred, lmin, lmax)
```



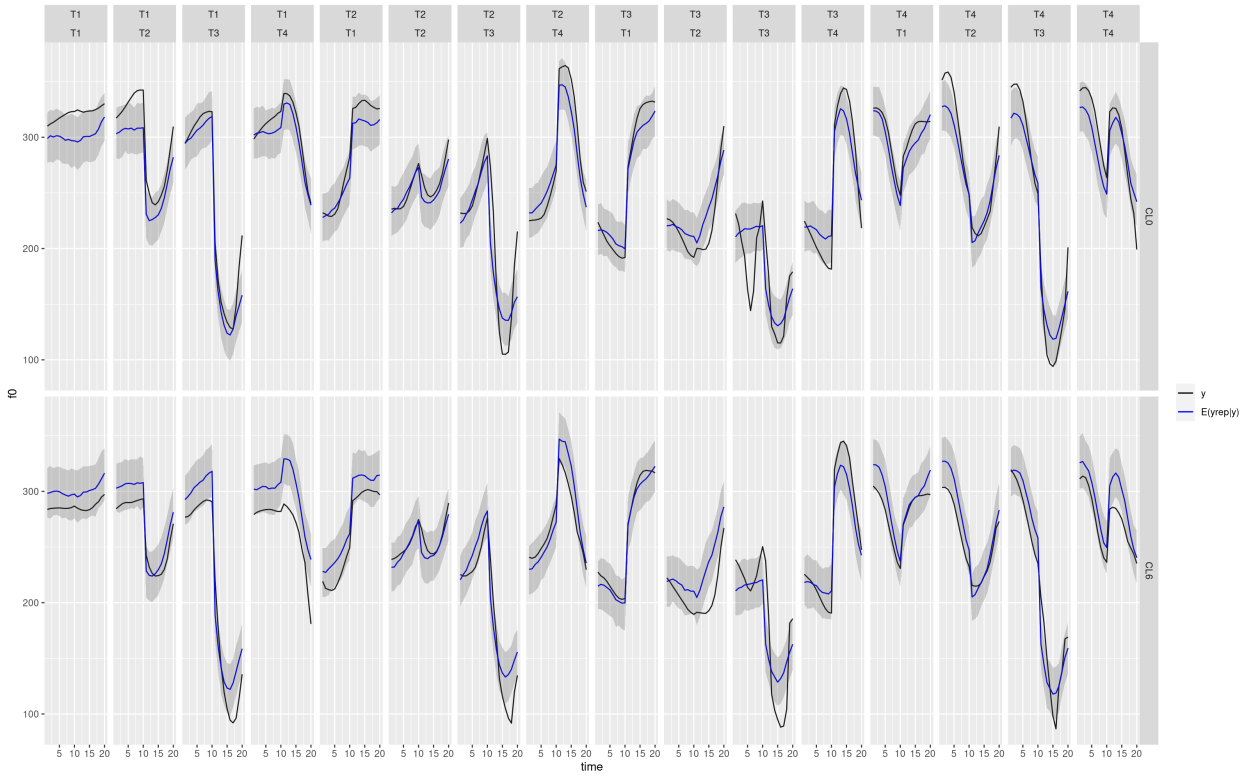

```
ppc_subject("S2", mandarino, ypred, lmin, lmax)
```



```
ppc_subject("S3", mandarino, ypred, lmin, lmax)
```




```
ppc_subject("S4", mandarino, ypred, lmin, lmax)
```



```
# ppc_subject("S5", mandarino, ypred, lmin, lmax)
# ppc_subject("S6", mandarino, ypred, lmin, lmax)
# ppc_subject("S7", mandarino, ypred, lmin, lmax)
# ppc_subject("S8", mandarino, ypred, lmin, lmax)
# ppc_subject("S9", mandarino, ypred, lmin, lmax)
# ppc_subject("S10", mandarino, ypred, lmin, lmax)
# ppc_subject("S11", mandarino, ypred, lmin, lmax)
# ppc_subject("S12", mandarino, ypred, lmin, lmax)

ppc_full(mandarino, ypred)
```