# French lexicon project: ex-Gaussian fits

*Guillaume A. Rousselet*

*2019-01-16*

## Contents



```
# dependencies
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 3.4.2
```

```
library(tibble)
```

```
## Warning: package 'tibble' was built under R version 3.4.3
```

```
library(retimes)
library(HDInterval)
```

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.14.2
##
```

```
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] HDInterval_0.1.3 retimes_0.1-2    tibble_1.4.2     cowplot_0.9.1
## [5] ggplot2_3.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.19     rstudioapi_0.8   bindr_0.1.1      knitr_1.17
##  [5] magrittr_1.5     tidyselect_0.2.4 munsell_0.4.3    colorspace_1.3-2
##  [9] R6_2.3.0         rlang_0.2.2      stringr_1.2.0    plyr_1.8.4
## [13] dplyr_0.7.6      tools_3.4.0      grid_3.4.0       gtable_0.2.0
## [17] withr_2.1.0      htmltools_0.3.6  assertthat_0.2.0 yaml_2.1.15
## [21] lazyeval_0.2.1   rprojroot_1.2    digest_0.6.12    crayon_1.3.4
## [25] bindrcpp_0.2.2   purrr_0.2.5      glue_1.3.0       evaluate_0.10.1
## [29] rmarkdown_1.8    stringi_1.1.6    compiler_3.4.0   pillar_1.3.0
## [33] scales_0.5.0     backports_1.1.1  pkgconfig_2.0.2
```

## Lexical decision dataset

Data from the French Lexicon Project. Click on "French Lexicon Project trial-level results with R scripts.zip". The `.RData` dataset was created by applying the script `/code/getflprtdata.Rmd`.

```
# get data - tibble = `flp`
load("./data/french_lexicon_project_rt_data.RData")
# columns =
#1 = participant
#2 = rt
#3 = acc = accuracy 0/1
#4 = condition = word/non-word
p.list <- unique(flp$participant)
Np <- length(p.list)
exg_param_w <- matrix(ncol=3, nrow=Np)
exg_param_nw <- matrix(ncol=3, nrow=Np)
```

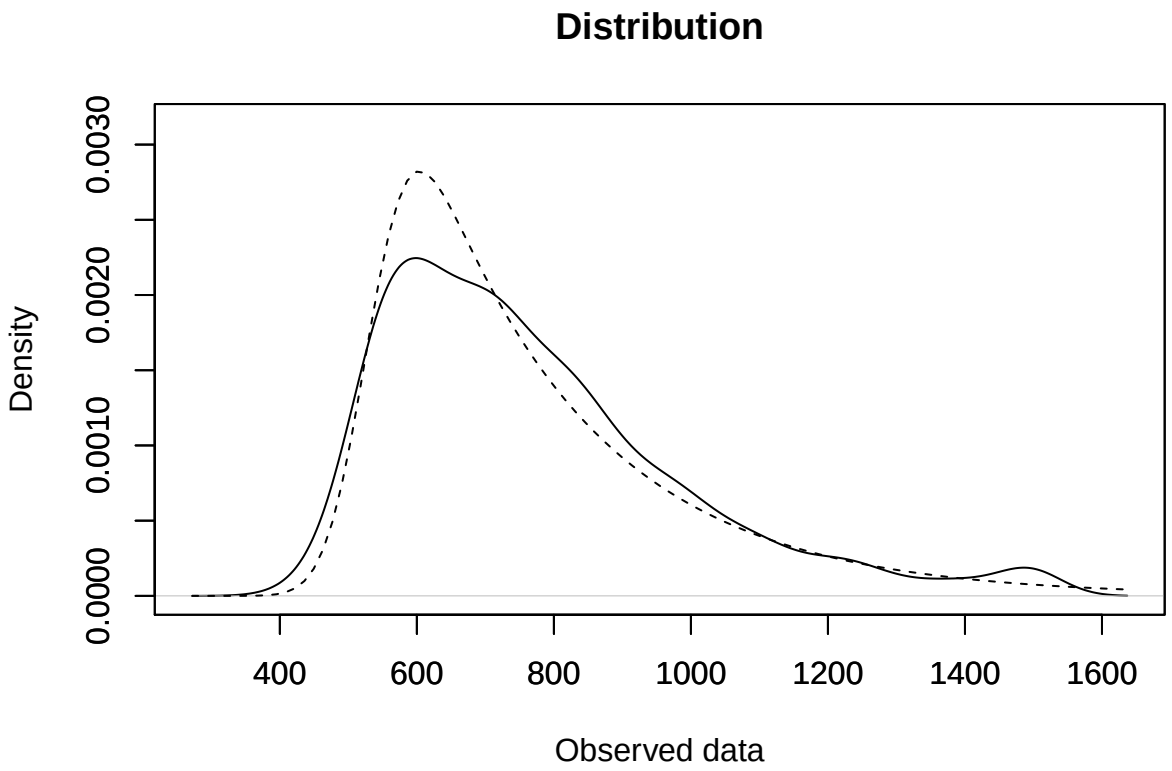
N = 959 participants.

## Check ex Gaussian fit

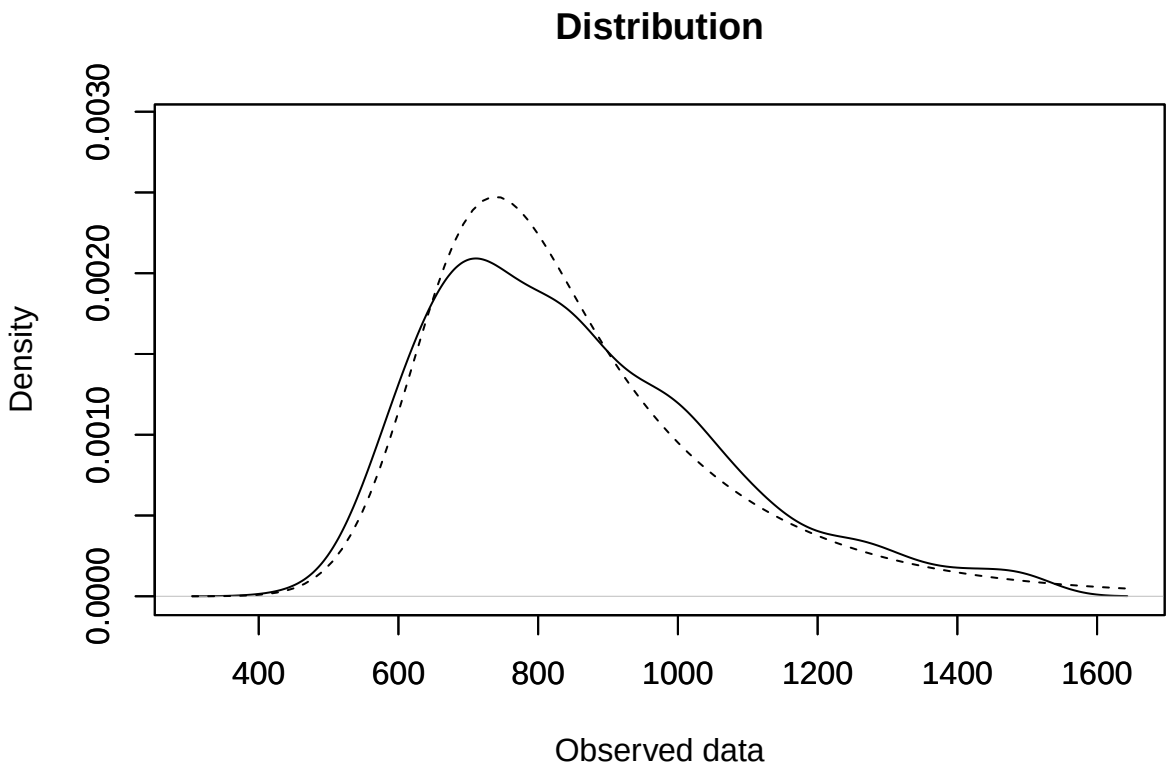
Word / non-word comparison:

```
# select participant
p.list <- unique(flp$participant)
sp <- p.list[1]
```

```r
# make KDE
flp.w <- sort(flp$rt[flp$participant==sp & flp$condition=="word"])
flp.nw <- sort(flp$rt[flp$participant==sp & flp$condition=="non-word"])
out <- timefit(flp.w, plot = TRUE)
```

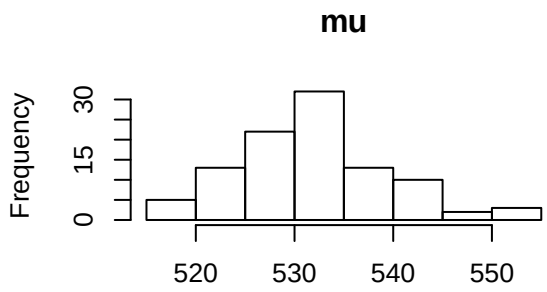


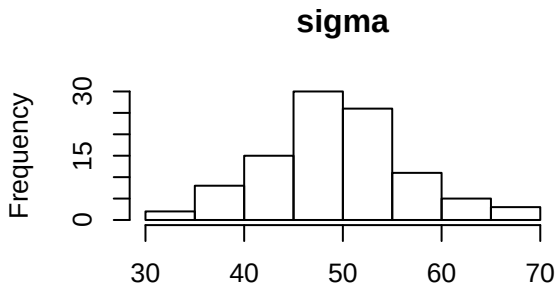

```r
out <- timefit(flp.nw, plot = TRUE)
```

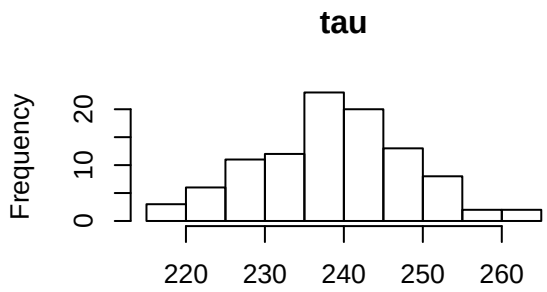


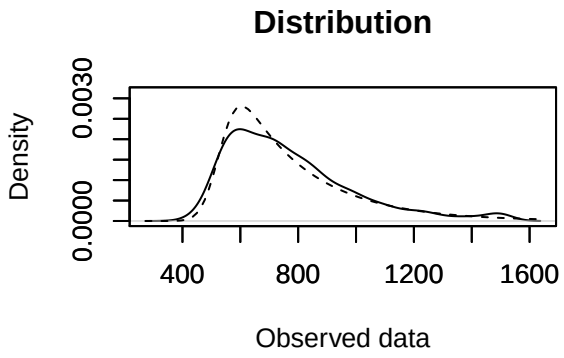

### Get distribution of parameters by resampling

```
out <- timefit(flp.w, plot = TRUE, iter = 100)
```

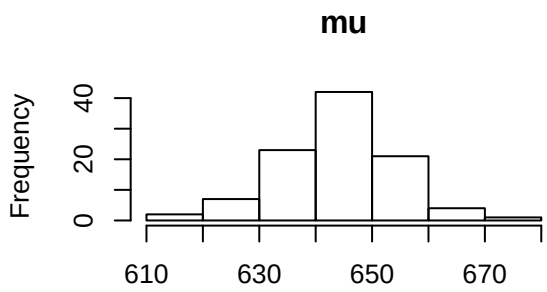


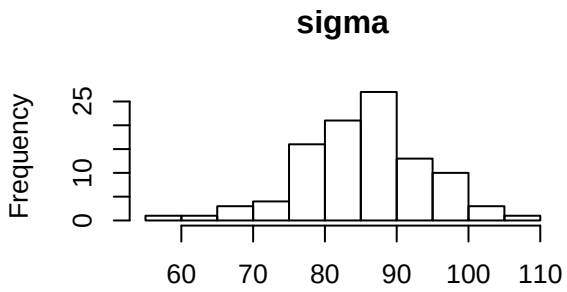



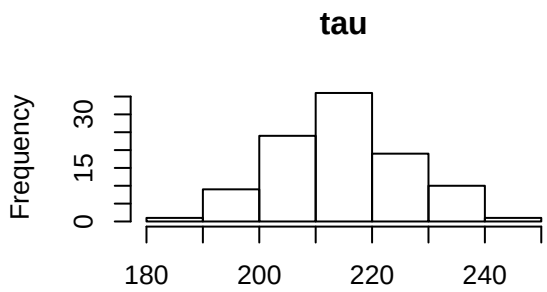



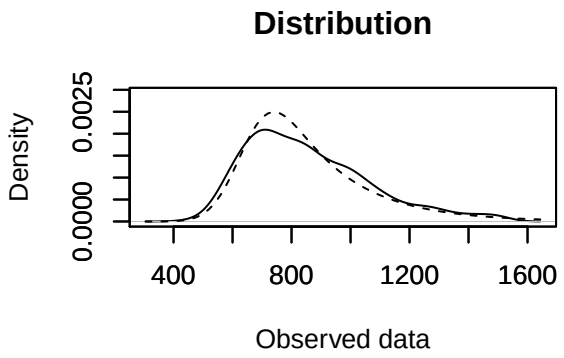




```
out <- timefit(flp.nw, plot = TRUE, iter = 100)
```



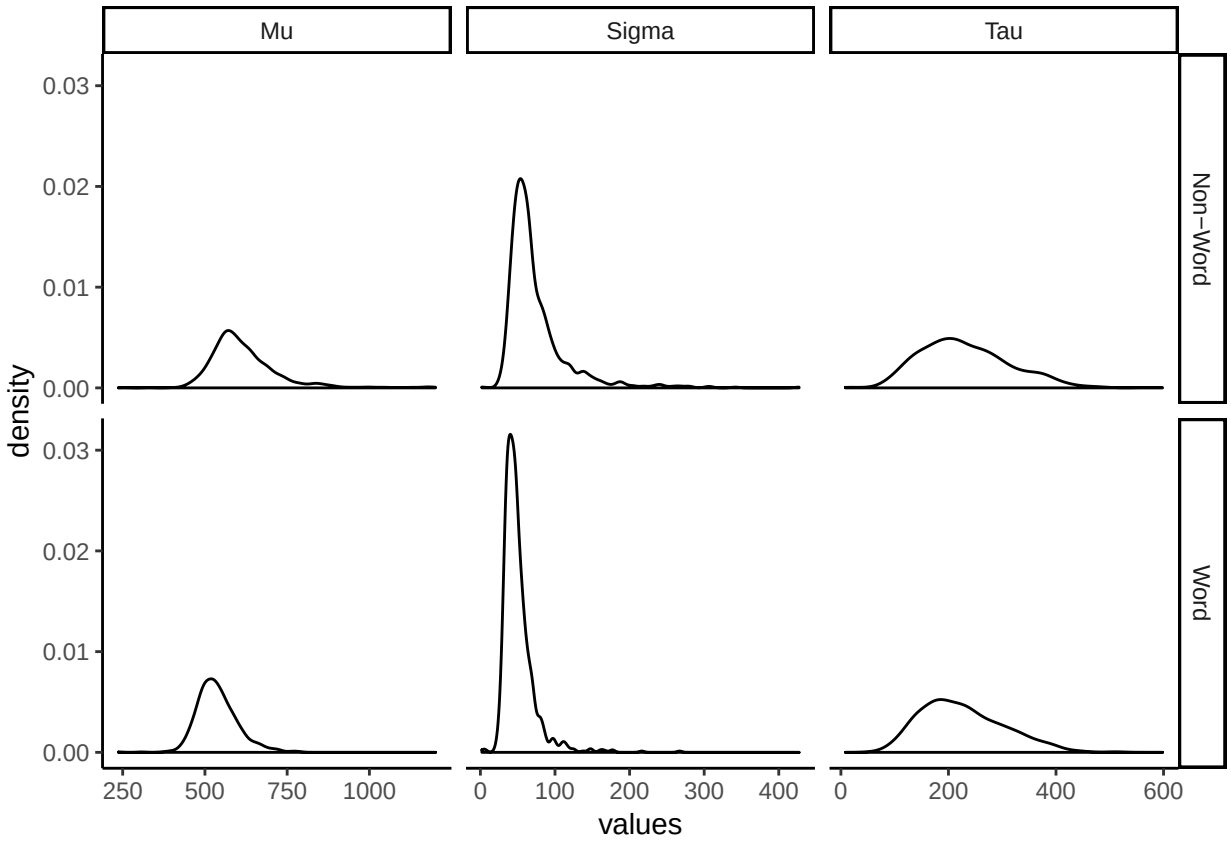



## Get parameters for all participants

```
for(P in 1:Np){
  # Word
  flp.w <- sort(flp$rt[flp$participant==p.list[P] & flp$condition=="word"])
  out <- timefit(flp.w)
  exg_param_w[P,] <- out@par

  # Non-Word
  flp.nw <- sort(flp$rt[flp$participant==p.list[P] & flp$condition=="non-word"])
  out <- timefit(flp.nw)
  exg_param_nw[P,] <- out@par
}

colnames(exg_param_w) <- c("mu", "sigma", "tau")
colnames(exg_param_nw) <- c("mu", "sigma", "tau")

save(exg_param_w,
     exg_param_nw,
     file = './data/flp_exg_param.RData')
```

## Illustrate parameters

```
load('./data/flp_exg_param.RData')

# create data frame
df <- tibble(values = c(as.vector(exg_param_w),as.vector(exg_param_nw)),
             Condition = c(rep("Word", Np*3), rep("Non-Word", Np*3)),
             Parameter = rep(rep(c("Mu", "Sigma", "Tau"), each = Np),2)
             )

# make plot
ggplot(df, aes(x = values)) + theme_classic() +
  geom_density() +
  facet_grid(rows = vars(Condition), cols = vars(Parameter), scales = "free_x")
```




## Compare parameters

### Mu

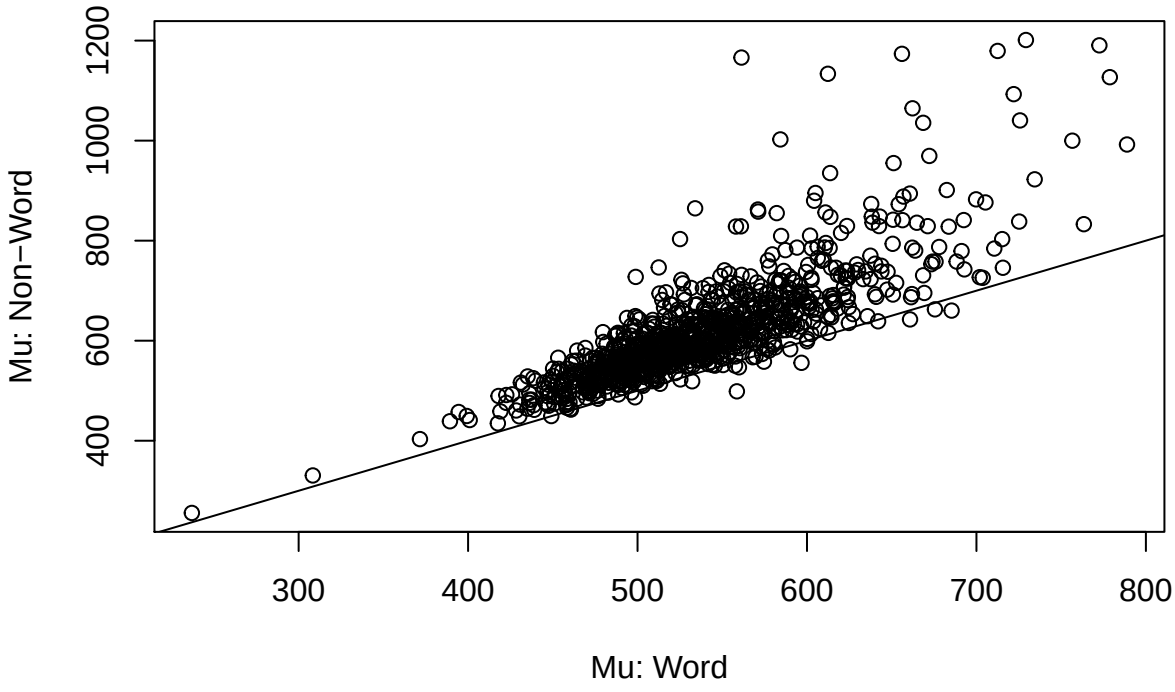
```
params <- c("Mu", "Sigma", "Tau")
paste("Word","Non-Word","Difference")
```

```
## [1] "Word Non-Word Difference"
```

```
P <- 1
hdires <- round(hdi(exg_param_nw[,P]-exg_param_w[,P],credMass=0.80))
paste0(params[P],": ",round(median(exg_param_w[,P])),", ",
  round(median(exg_param_nw[,P])),", ",            round(median(exg_param_nw[,P])-median(exg_param_w[,P
           hdires[1],", ", hdires[2],"]")
```

```
## [1] "Mu: 530, 596, 65 [11, 117]"
```

```
plot(exg_param_w[,P],exg_param_nw[,P],
     xlab = paste0(params[P],": Word"),
     ylab = paste0(params[P],": Non-Word"))
abline(a = 0, b = 1)
```

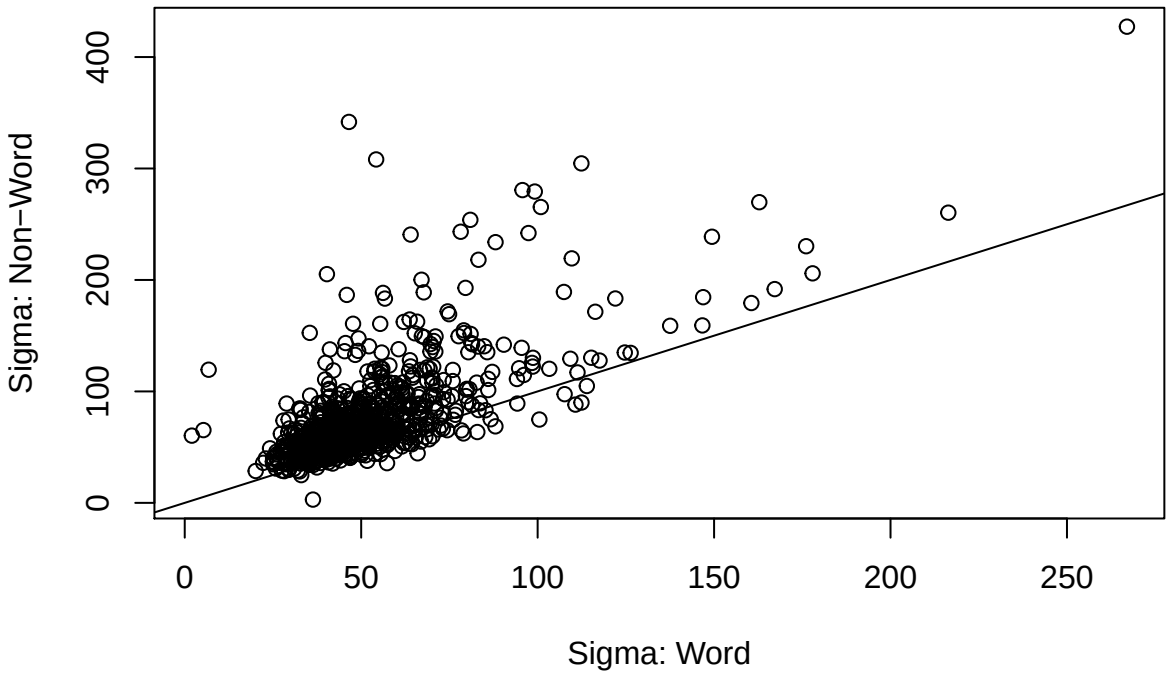



### Sigma

```
P <- 2
hdires <- round(hdi(exg_param_nw[,P]-exg_param_w[,P],credMass=0.80))
paste0(params[P],": ",round(median(exg_param_w[,P])),", ",
  round(median(exg_param_nw[,P])),", ",            round(median(exg_param_nw[,P])-median(exg_param_w[,P
            hdires[1],", ", hdires[2],"]")
```

```
## [1] "Sigma: 46, 62, 16 [-7, 38]"
```

```
plot(exg_param_w[,P],exg_param_nw[,P],
     xlab = paste0(params[P],": Word"),
     ylab = paste0(params[P],": Non-Word"))
abline(a = 0, b = 1)
```

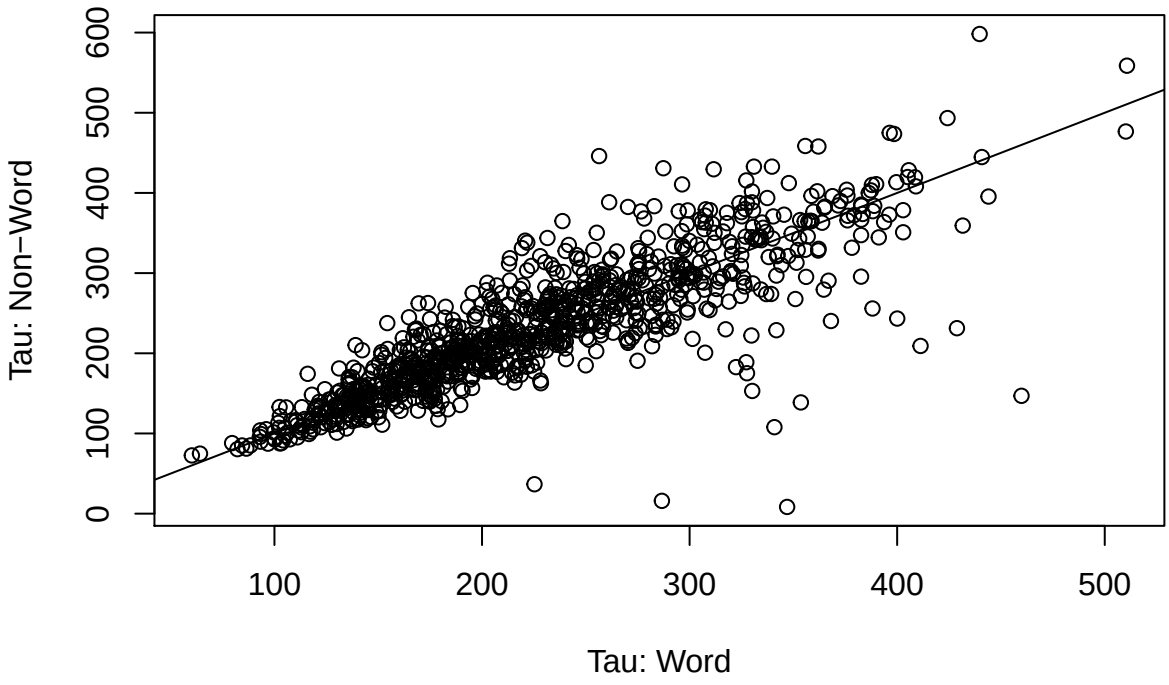



### Tau

```
P <- 3
hdires <- round(hdi(exg_param_nw[,P]-exg_param_w[,P],credMass=0.80))
paste0(params[P],": ",round(median(exg_param_w[,P])),", ",
  round(median(exg_param_nw[,P])),", ",            round(median(exg_param_nw[,P])-median(exg_param_w[,P
            hdires[1],", ", hdires[2],"]")
```

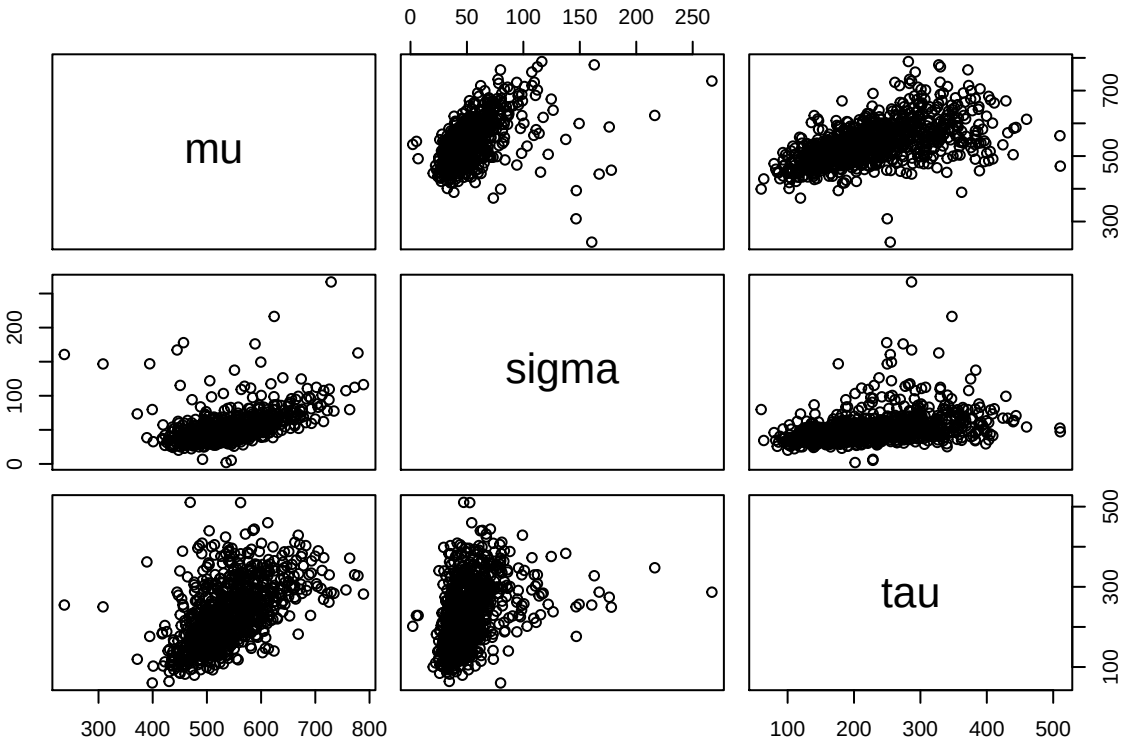
```
## [1] "Tau: 216, 221, 5 [-34, 52]"
```

```
plot(exg_param_w[,P],exg_param_nw[,P],
     xlab = paste0(params[P],": Word"),
     ylab = paste0(params[P],": Non-Word"))
abline(a = 0, b = 1)
```



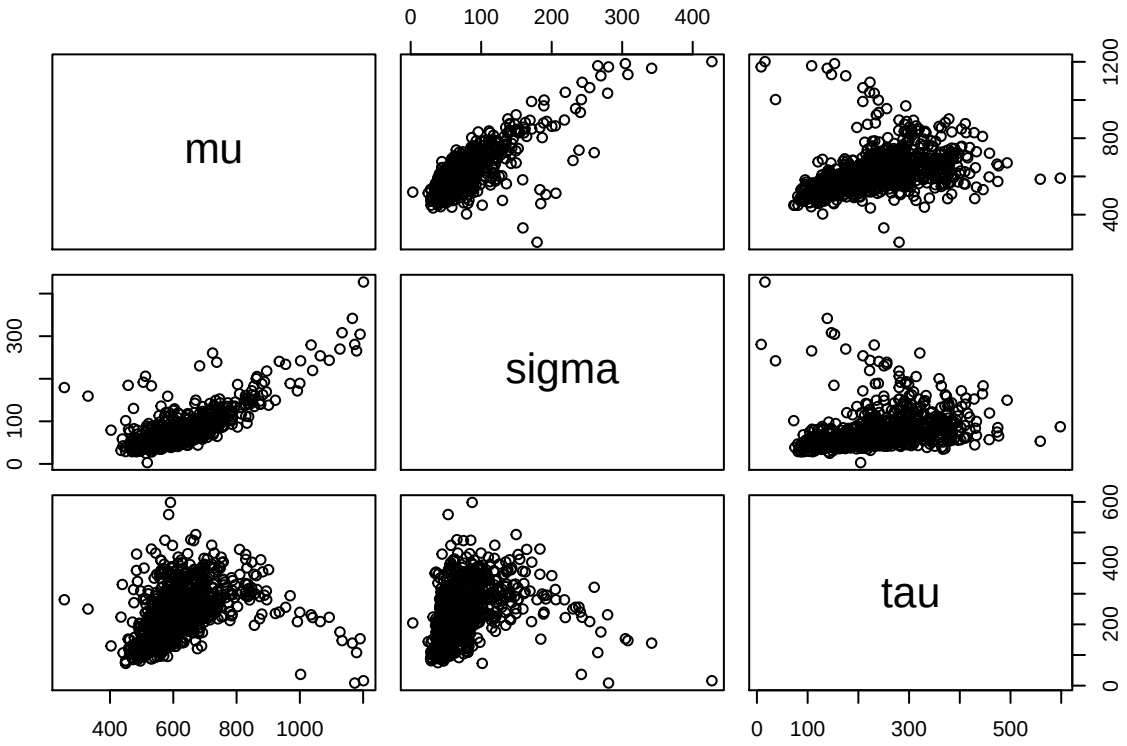

### Parameter correlations: Word

```
pairs(exg_param_w)
```




## Parameter correlations: Non-Word

```
pairs(exg_param_nw)
```




## Covariance matrices

```
# covmat_w <- cov(exg_param_w)
cov(exg_param_w)
```

```
##              mu    sigma       tau
## mu    3802.2621 602.8309 2560.3318
## sigma  602.8309 467.3606  612.0562
## tau   2560.3318 612.0562 5813.0170
```

```
# covmat_nw <- cov(exg_param_nw)
cov(exg_param_nw)
```

```
##              mu    sigma      tau
## mu    11126.824 3370.122 3114.721
## sigma  3370.122 1693.188  974.327
## tau    3114.721  974.327 6855.443
```

## Correlation matrices

```
# covmat_w <- cov(exg_param_w)
cov2cor(cov(exg_param_w))
```

```
##              mu     sigma       tau
## mu    1.0000000 0.4522186 0.5445960
## sigma 0.4522186 1.0000000 0.3713338
## tau   0.5445960 0.3713338 1.0000000
```

```
# covmat_nw <- cov(exg_param_nw)
cov2cor(cov(exg_param_nw))
```

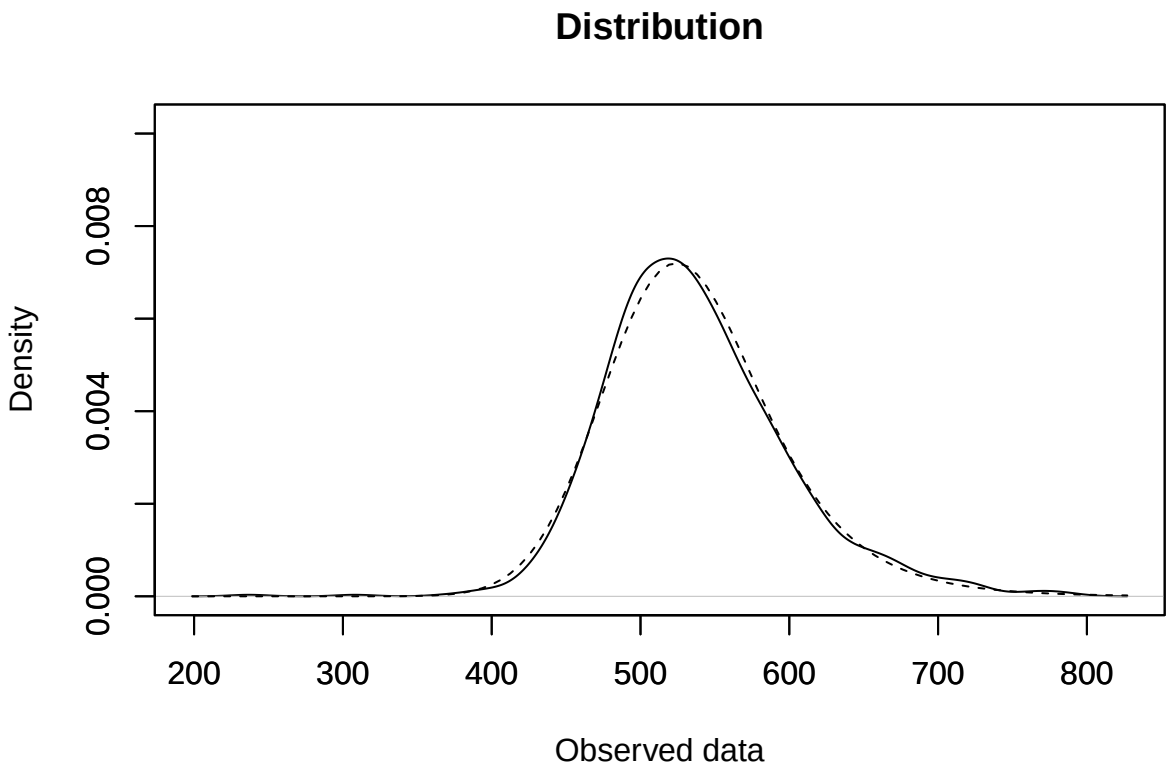
```
##              mu     sigma       tau
## mu    1.0000000 0.7764392 0.3566282
## sigma 0.7764392 1.0000000 0.2859792
## tau   0.3566282 0.2859792 1.0000000
```

# ExG fit of ExG parameters

## Word

### Mu

```
exg_param_w_meta <- list()
exg_param_nw_meta <- list()

out <- timefit(exg_param_w[,1], plot = TRUE)
```



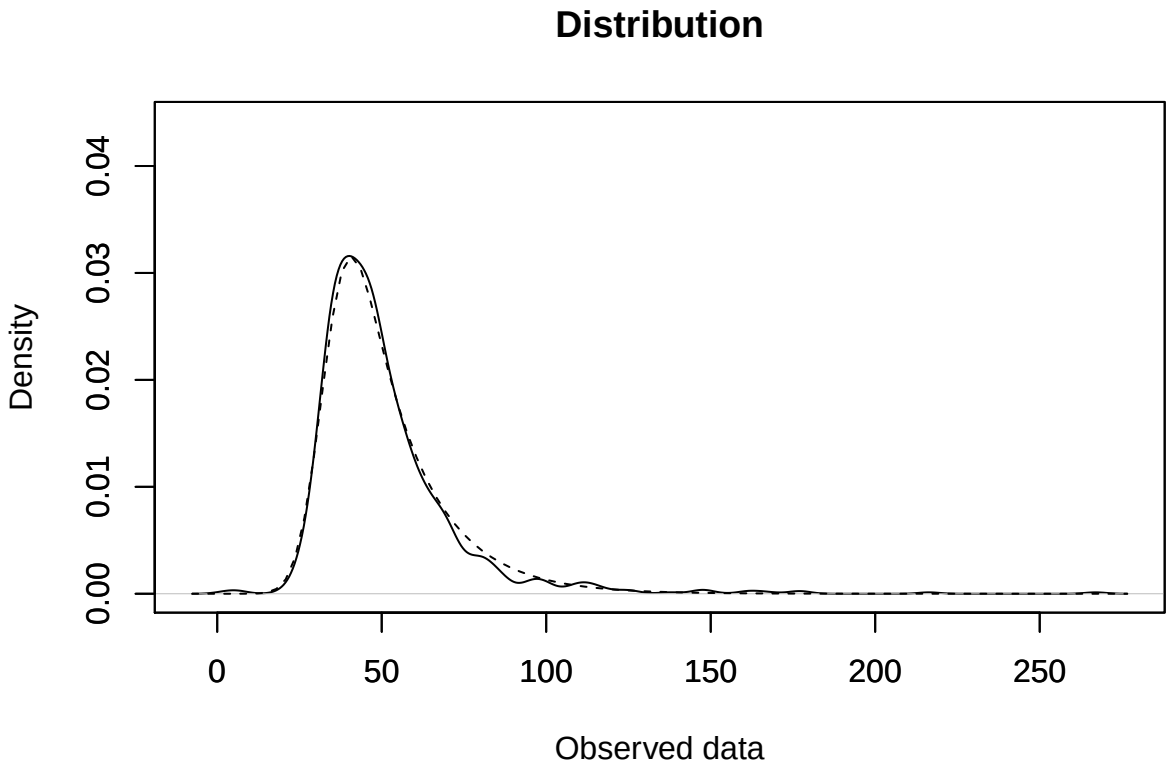

```
out@par
```

```
##        mu     sigma       tau
## 493.74768  43.26326  43.86880
```

```
exg_param_w_meta$mu <- out@par
```

**Sigma**

```
out <- timefit(exg_param_w[,2], plot = TRUE)
```



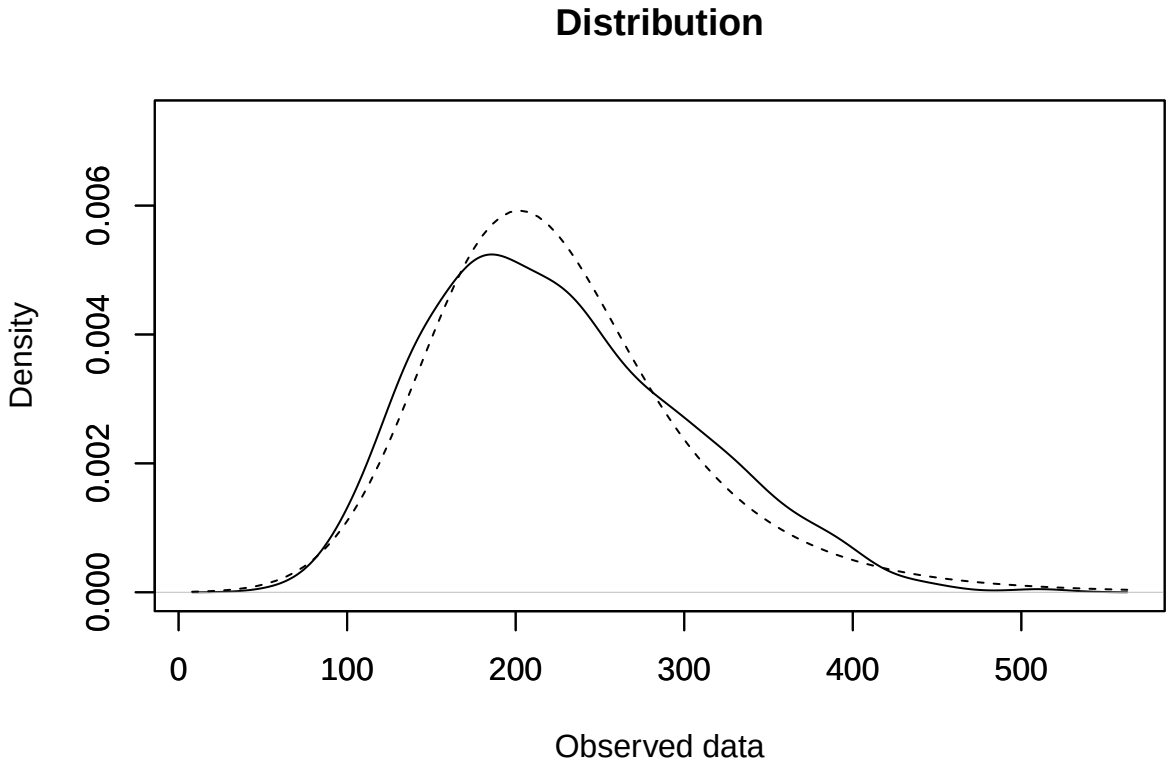

```
out@par
```

```
##        mu     sigma       tau
## 33.253308  6.431256 17.299312
```

```
exg_param_w_meta$sigma <- out@par
```

**Tau**

```
out <- timefit(exg_param_w[,3], plot = TRUE)
```




```
out@par
```

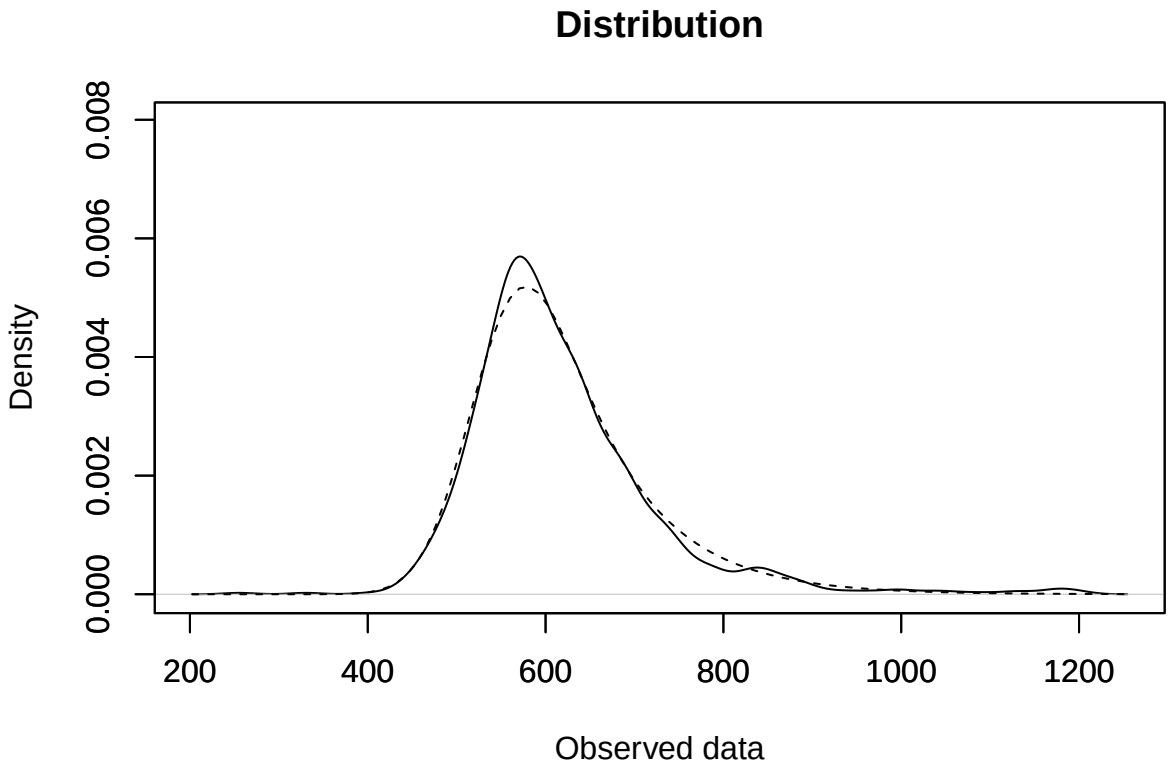
```
##        mu     sigma       tau
## 162.67433  48.00298  63.54590
```

```
exg_param_w_meta$tau <- out@par
```

## Non-Word

### Mu

```
out <- timefit(exg_param_nw[,1], plot = TRUE)
```



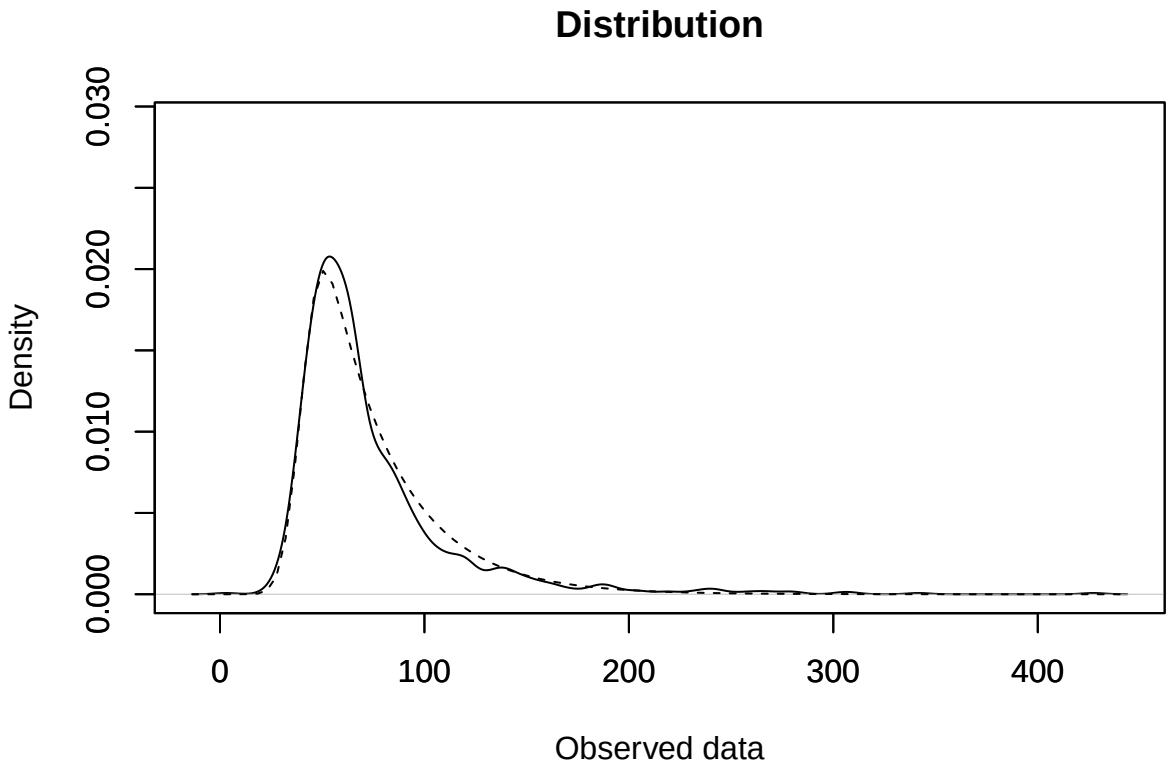

```
out@par
```

```
##        mu     sigma       tau
## 531.33178  48.32347  86.29880
```

```
exg_param_nw_meta$mu <- out@par
```

**Sigma**

```
out <- timefit(exg_param_nw[,2], plot = TRUE)
```



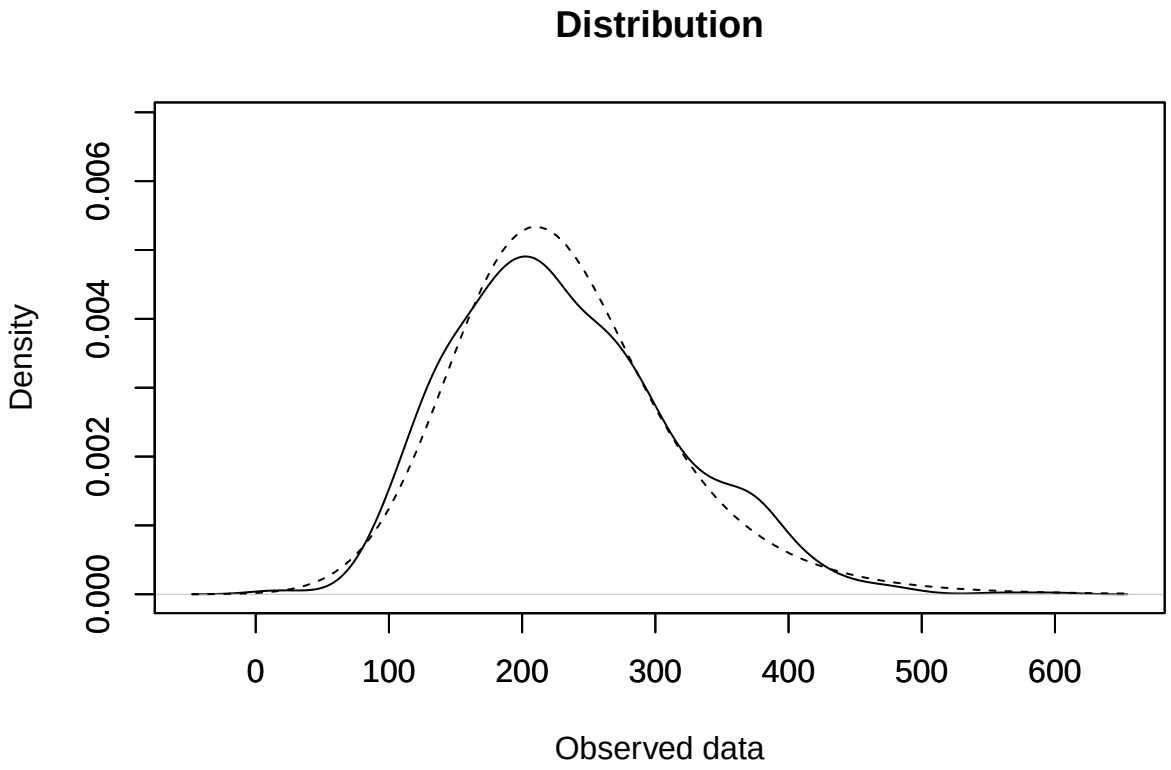

```
out@par
```

```
##       mu    sigma      tau
## 40.47351  7.34384 33.30768
```

```
exg_param_nw_meta$sigma <- out@par
```

**Tau**

```
out <- timefit(exg_param_nw[,3], plot = TRUE)
```




```
out@par
```

```
##        mu     sigma       tau
## 168.07342  56.56520  63.15946
```

```
exg_param_nw_meta$tau <- out@par
```

```
save(exg_param_w,
     exg_param_nw,
     exg_param_w_meta,
     exg_param_nw_meta,
     file = './data/flp_exg_param.RData')
```

## Simulate RT distributions using exG parameters

### Sample participants' parameters with replacement

```
Np.total <- nrow(exg_param_w)
Np <- 20 # participants
Nt <- 100 # trials
rt_data <- matrix(ncol=Np, nrow=Nt)

# Resample participants' triads
bootsample <- sample(Np.total, Np, replace = TRUE)

# Generate random trials for each participant
# based on their triads of ex Gaussian parameters
for(P in 1:Np){
```

```
  rt_data[,P] <- rexgauss(Nt,
                      mu = exg_param_w[bootsample[P],1],
                      sigma = exg_param_w[bootsample[P],2],
                      tau = exg_param_w[bootsample[P],3])
}
```

### Illustrate results

```
df <- tibble(rt = as.vector(rt_data),
             Participant = factor(rep(seq(1,Np),each = Nt)))

ggplot(df, aes(x = rt, colour = Participant)) +
  geom_density()
```