

# ERGtools2 - Examples of how to create an ERGExam object from a set of CSV files

Moritz Lindner

22 Februar, 2025

## Contents

<b>Case 1: Raw data for each Recording are stored in a separate file</b>	<b>2</b>
Make the example data: Extract data from an ERGExam object and store into CSV file . . . . .	2
Create an ERGExam from the data just stored . . . . .	4
<b>Case 2: Average traces from all Recordings per step/stimulus condition</b>	<b>6</b>
Make the example data: Extract data from an ERGExam object and store into CSV file . . . . .	6
Create an ERGExam from the data just stored . . . . .	7
<b>Session info</b>	<b>12</b>

*FilePath: “/media/moritz/DATADISK/02\_Manuskripte/ERGtools2”*

```
require(ERGtools2)
```

```
## Lade nötiges Paket: ERGtools2
```

```
## Lade nötiges Paket: EPhysData
```

```
## Lade nötiges Paket: units
```

```
## udunits database from /usr/share/xml/udunits/udunits2.xml
```

```
## Registered S3 method overwritten by 'quantmod':
```

```
##   method             from
```

```
##   as.zoo.data.frame zoo
```

```
##
```

```
## Attache Paket: 'ERGtools2'
```

```
## Das folgende Objekt ist maskiert 'package:base':
```

```
##
```

```
##   Position
```

```
require(tidyr)
```

```
## Lade nötiges Paket: tidyr
```

```
require(stringr)
```

```
## Lade nötiges Paket: stringr
```

## Case 1: Raw data for each Recording are stored in a separate file

Make the example data: Extract data from an ERGexam object and store into CSV file

Metadata and stimulus information are stored in one file each, then each Recording of the exam is stored in a separate file, first column containing the time, and the sequential columns the individual trials

```
data(ERG)
MD <- Metadata(ERG)
Stim <- Stimulus(ERG)
dir.create("CVS_Example")
```

```
## Warning in dir.create("CVS_Example"): 'CVS_Example' existiert bereits
```

```
write.csv2(MD, paste0(getwd(), "/CVS_Example/Metadata.csv"), row.names =
  F) # may need adjustment depending on Operating System
write.csv2(Stim, paste0(getwd(), "/CVS_Example/Stimulus.csv"), row.names =
  F) # may need adjustment depending on Operating System
for (i in 1:nrow(MD)) {
  df <- as.data.frame(Subset(ERG,where=i),IncludeRejected = T)
  df <- df[,c("Trial","Time","Value")]
  df <- pivot_wider(
    df,
    names_from = "Trial",
    names_prefix = "Trial_",
    values_from = "Value"
  )
  fn <-
    paste0("ERG-Step-",
      MD$Step[i],
      "_Ch-",
      MD$Channel[i],
      "_Eye-",
      MD$Eye[i],
      ".csv")
  write.csv2(df, paste0(getwd(), "/CVS_Example/", fn), row.names = F, dec = ".") # may need adjustment
}
```

```
## Warning in write.csv2(df, paste0(getwd(), "/CVS_Example/", fn), row.names = F,
## : attempt to set 'dec' ignored
```

[illegible]

```
list.files(paste0(getwd(), "/CVS_Example/"))
```

```
head(read.csv2(paste0(getwd(), "/CVS_Example/"), fn), dec = "."))
```

## Create an ERGExam from the data just stored

```
Metadata<-read.csv2(paste0(getwd(),"/CVS_Example/Metadata.csv"))
Stimulus<-read.csv2(paste0(getwd(),"/CVS_Example/Stimulus.csv"))

Recordings <- list() # make a list to store the recordings from each file (as EPhysData objects), this
for (i in 1:nrow(Metadata)) {
  fn <-
    paste0(
      "ERG-Step-",
      Metadata$Step[i],
      "_Ch-",
      Metadata$Channel[i],
      "_Eye-",
      Metadata$Eye[i],
      ".csv"
    )
  print(fn)
  # read the data for each recording, split into data and time trace and assign units
  df <- read.csv2(paste0(getwd(), "/CVS_Example/", fn), dec = ".")
  recording.data <- df[, str_detect(colnames(df), "Trial")]
  recording.data <- as_units(as.matrix(recording.data), "nV")
  time <- df[, "Time"]
  time <- as_units(time, "ms")
  # create the EPhysData object
  Recordings[[i]] <-
    newEPhysData(Data = recording.data, TimeTrace = time)
}
```

```
## [1] "ERG-Step-1_Ch-ERG_Eye-RE.csv"
## [1] "ERG-Step-1_Ch-ERG_Eye-LE.csv"
## [1] "ERG-Step-1_Ch-OP_Eye-RE.csv"
## [1] "ERG-Step-1_Ch-OP_Eye-LE.csv"
## [1] "ERG-Step-2_Ch-ERG_Eye-RE.csv"
## [1] "ERG-Step-2_Ch-ERG_Eye-LE.csv"
## [1] "ERG-Step-2_Ch-OP_Eye-RE.csv"
## [1] "ERG-Step-2_Ch-OP_Eye-LE.csv"
## [1] "ERG-Step-3_Ch-ERG_Eye-RE.csv"
## [1] "ERG-Step-3_Ch-ERG_Eye-LE.csv"
## [1] "ERG-Step-3_Ch-OP_Eye-RE.csv"
## [1] "ERG-Step-3_Ch-OP_Eye-LE.csv"
```

```
# An ERGExam object requires additional information on the subject (e.g. name, DOB) as well as on the exam
SubjectInfo <- list(Subject = "Test", DOB = as.Date("2000-01-01"))
ExamInfo <- list(ProtocolName = "TestProtocol", ExamDate = as.POSIXct("2024-01-01"))

# Now the ERGExam object is assembled
ImportedExam<-newERGExam(Data = Recordings,
                          Metadata = Metadata,
                          Stimulus = Stimulus,
                          Averaged = F,
                          ExamInfo = ExamInfo,
```

```

SubjectInfo = SubjectInfo)

validObject(ImportedExam)

```

```
## [1] TRUE
```

```
# Compare (re)imported Exam to the original exam
```

```

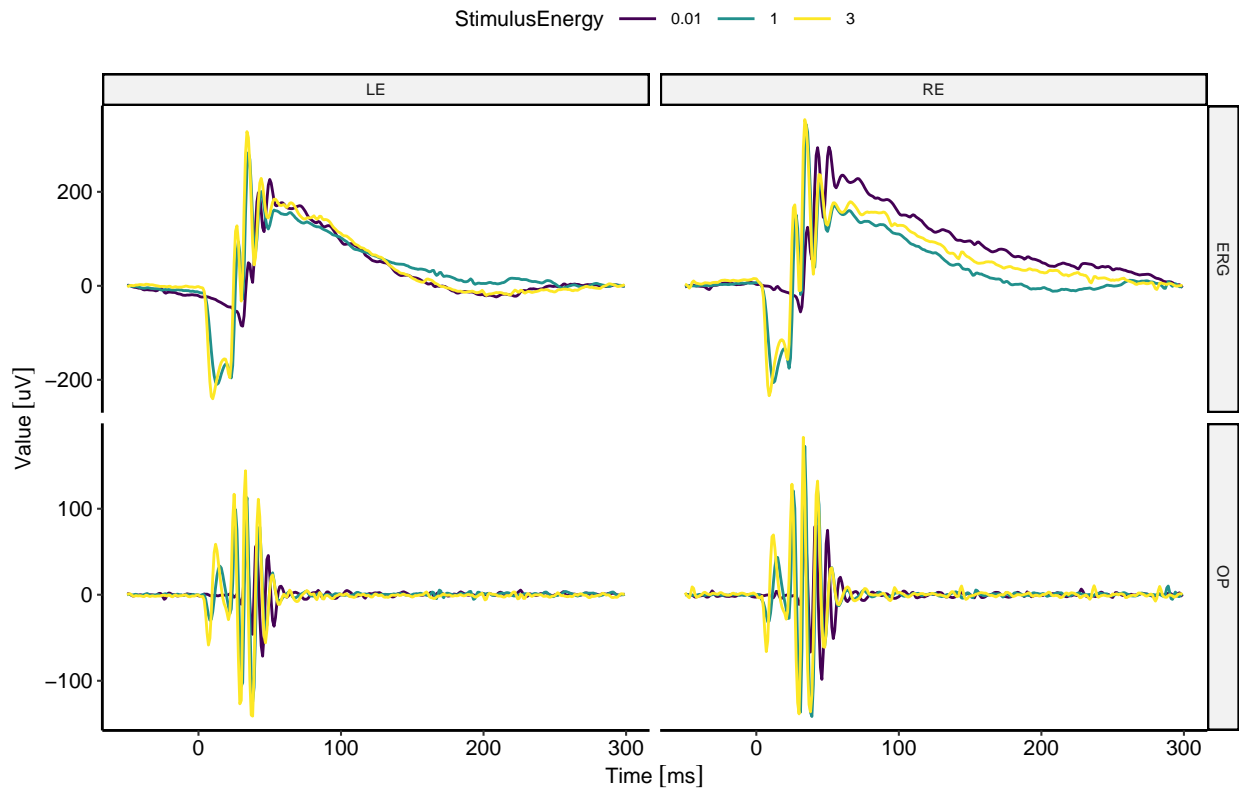
data(ERG)
Original<-ERG

ImportedExam<-SetStandardFunctions(ImportedExam)
Original<-SetStandardFunctions(Original)

ggERGExam(ImportedExam)

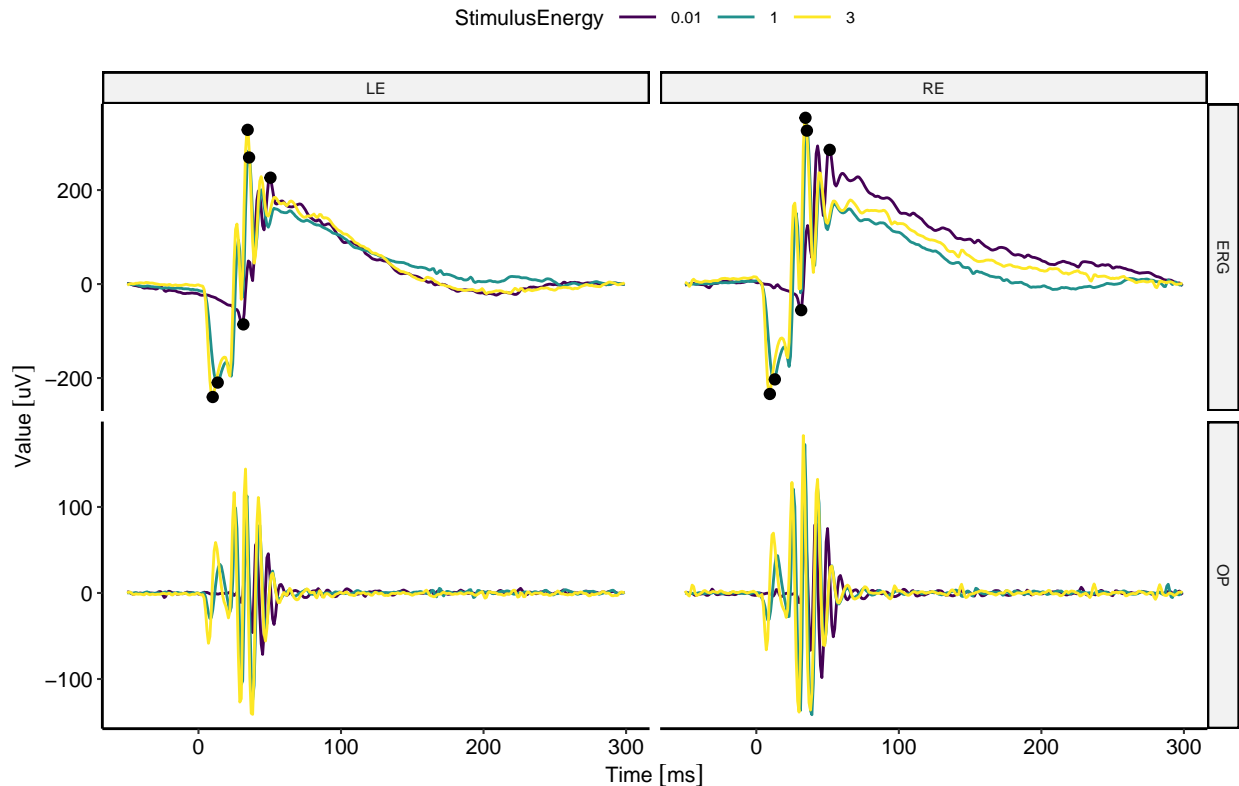
```

DA-Flash



```
ggERGExam(Original)
```

## DA-Flash



## Case 2: Average traces from all Recordings per step/stimulus condition

Make the example data: Extract data from an ERGexam object and store into CSV file

Metadata and stimulus information are stored in one file each, then each Recording of the exam is stored in a separate file, first column containing the time, and the sequential columns the individual trials

```
unlink(paste0(getwd(), "/CVS_Example/"),recursive=T)
rm(list = ls())
data(ERG)
ERG<- Subset(ERG, Raw=F) # make object to store the averaged traces instead of raw data
MD <- Metadata(ERG)
Stim <- Stimulus(ERG)
dir.create("CVS_Example")
write.csv2(MD, paste0(getwd(), "/CVS_Example/Metadata.csv"), row.names =
  F) # may need adjustment depending on Operating System
write.csv2(Stim, paste0(getwd(), "/CVS_Example/Stimulus.csv"), row.names =
  F) # may need adjustment depending on Operating System
for (i in 1:nrow(Stim)) {
  df <- as.data.frame(Subset(ERG,where=list(Step=Stim$Step[[i]])),IncludeRejected = T)
  df <- df[,c("Channel","Eye","Time","Value")]
}
```

```

df <- pivot_wider(
  df,
  names_from = c("Channel", "Eye"),
  names_prefix = "Value_",
  values_from = "Value"
)
fn <-
  paste0("ERG-Step-",
        Stim$Step[[i]],
        ".csv")
write.csv2(df, paste0(getwd(), "/CVS_Example/"), fn), row.names = F, dec = ".") # may need adjustment
}

```

```

## Warning in write.csv2(df, paste0(getwd(), "/CVS_Example/"), fn), row.names = F,
## : attempt to set 'dec' ignored
## Warning in write.csv2(df, paste0(getwd(), "/CVS_Example/"), fn), row.names = F,
## : attempt to set 'dec' ignored
## Warning in write.csv2(df, paste0(getwd(), "/CVS_Example/"), fn), row.names = F,
## : attempt to set 'dec' ignored

```

This is the content of the directory:

```
list.files(paste0(getwd(), "/CVS_Example/"))
```

```

## [1] "ERG-Step-1.csv" "ERG-Step-2.csv" "ERG-Step-3.csv" "Metadata.csv"
## [5] "Stimulus.csv"

```

This is how the data in each file looks like, note the column names contain the channel identifier (ERG/OP) and the eye identifier (RE/LE)

```
head(read.csv2(paste0(getwd(), "/CVS_Example/"), fn), dec = ".")
```

```

##      Time Value_ERG_RE Value_ERG_LE Value_OP_RE Value_OP_LE
## 1 -50.0    3797.9881   -2465.3628   2375.0339   -225.8795
## 2 -49.5    3014.5190   -1755.6906   1136.0682    360.1678
## 3 -49.0    2254.7980    -606.4151     32.0034   1081.8208
## 4 -48.5    1034.7684    172.9521  -1032.9668   1366.6543
## 5 -48.0    -827.4936    585.9339  -2169.7392   1138.6010
## 6 -47.5   -3086.9021    906.4600  -3207.4051    669.2753

```

```
rm(list = ls())
```

## Create an ERGExam from the data just stored

```

Metadata<-read.csv2(paste0(getwd(), "/CVS_Example/Metadata.csv"))
Stimulus<-read.csv2(paste0(getwd(), "/CVS_Example/Stimulus.csv"))

```

```

Recordings <- list() # make a list to store the recordings from each file (as EPhysData objects), this
for (i in 1:nrow(Stimulus)) {

```

```

fn <-
  paste0("ERG-Step-",
        Stimulus$Step[[i]],
        ".csv")
print(fn)
# read the data for each step, split into data and time trace and assign units
df <- read.csv2(paste0(getwd(), "/CVS_Example/", fn), dec = ".")
trace.data <- df[, str_detect(colnames(df), "Value")]
trace.data <- as_units(as.matrix(trace.data), "nV")
time <- df[, "Time"]
time <- as_units(time, "ms")
# create the EPhysData object
current.metadata<-Metadata[Metadata$Step==Stimulus$Step[[i]],]
for (j in 1:nrow(current.metadata)){ # for each Channel and Eye that has data from under the current
  current.column.name <-
    paste0("Value_",
          current.metadata$Channel[[j]],
          "_",
          current.metadata$Eye[[j]])
  current.trace<-trace.data[,current.column.name]
  Recordings<-append(Recordings, newEPhysData(Data = current.trace, TimeTrace = time))
}
}

```

```

## [1] "ERG-Step-1.csv"
## [1] "ERG-Step-2.csv"
## [1] "ERG-Step-3.csv"

```

```

# An ERGExam object requires additional information on the subject (e.g. name, DOB) as well as on the exam
SubjectInfo <- list(Subject = "Test", DOB = as.Date("2000-01-01"))
ExamInfo <- list(ProtocolName = "TestProtocol", ExamDate = as.POSIXct("2024-01-01"))

```

```

# Now the ERGExam object is assembled
ImportedExam<-newERGExam(Data = Recordings,
                        Metadata = Metadata,
                        Stimulus = Stimulus,
                        Averaged = F,
                        ExamInfo = ExamInfo,
                        SubjectInfo = SubjectInfo)

```

```

validObject(ImportedExam)

```

```

## [1] TRUE

```

```

# Compare (re)imported Exam to the original exam

```

```

data(ERG)
Original<-ERG

```

```

ImportedExam<-SetStandardFunctions(ImportedExam)

```

```

## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.

```

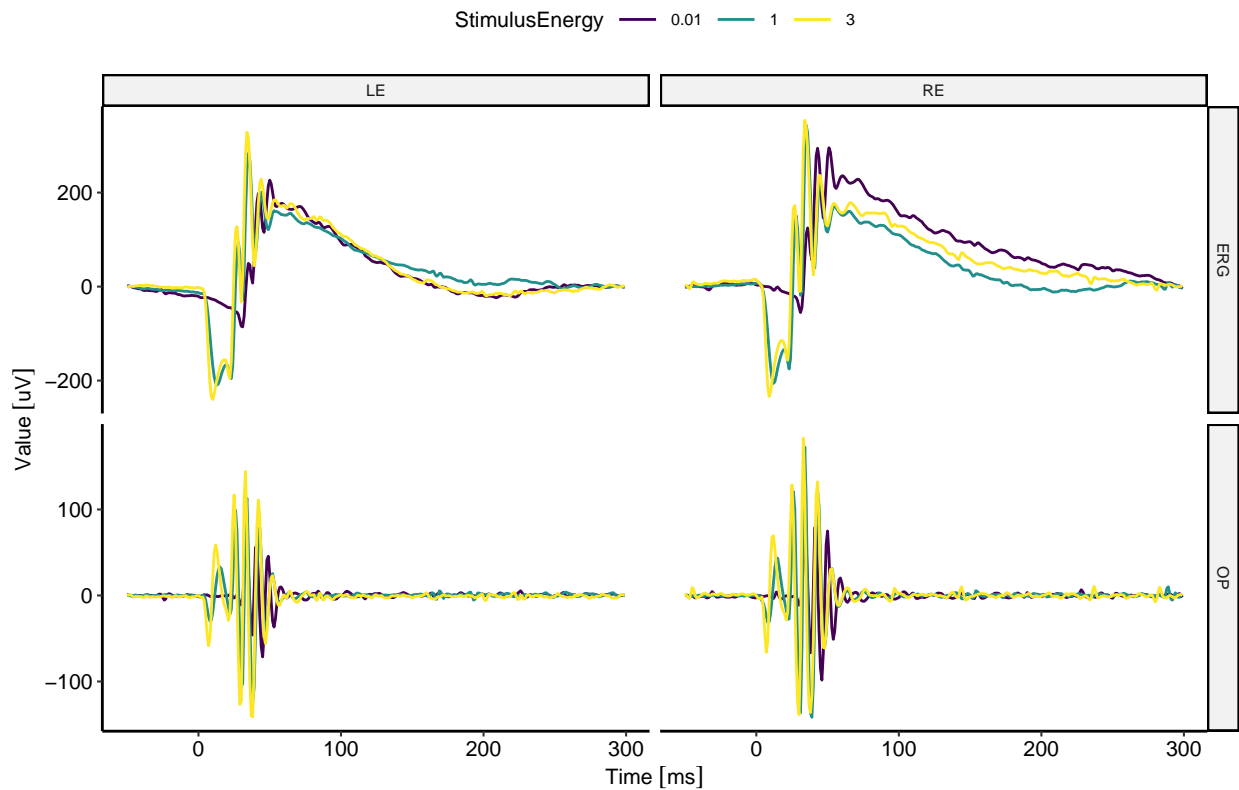


```
## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.
## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.
## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.
## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.
## Can't set a Rejected function for 'X', because 'X' contains only one trial. Keeping it.
```

```
Original<-SetStandardFunctions(Original)
```

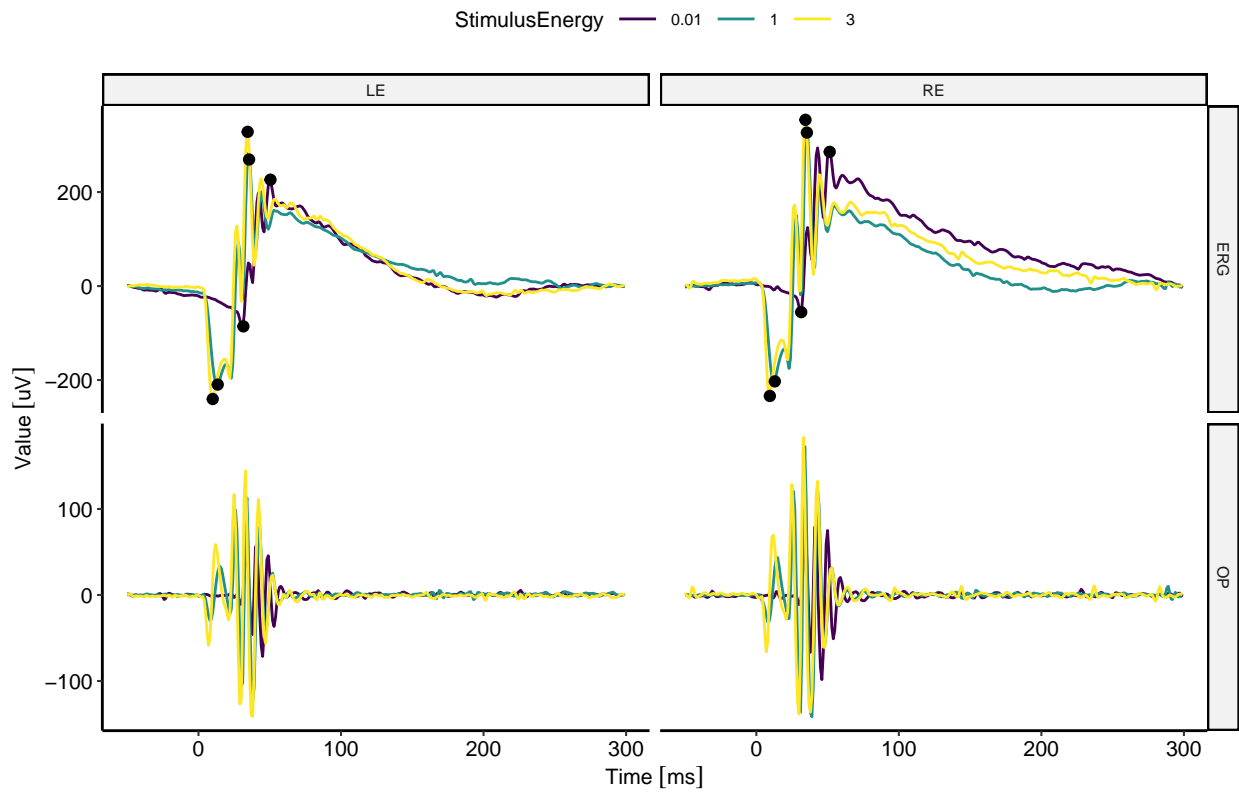
```
ggERGExam(ImportedExam)
```

DA-Flash



```
ggERGExam(Original)
```

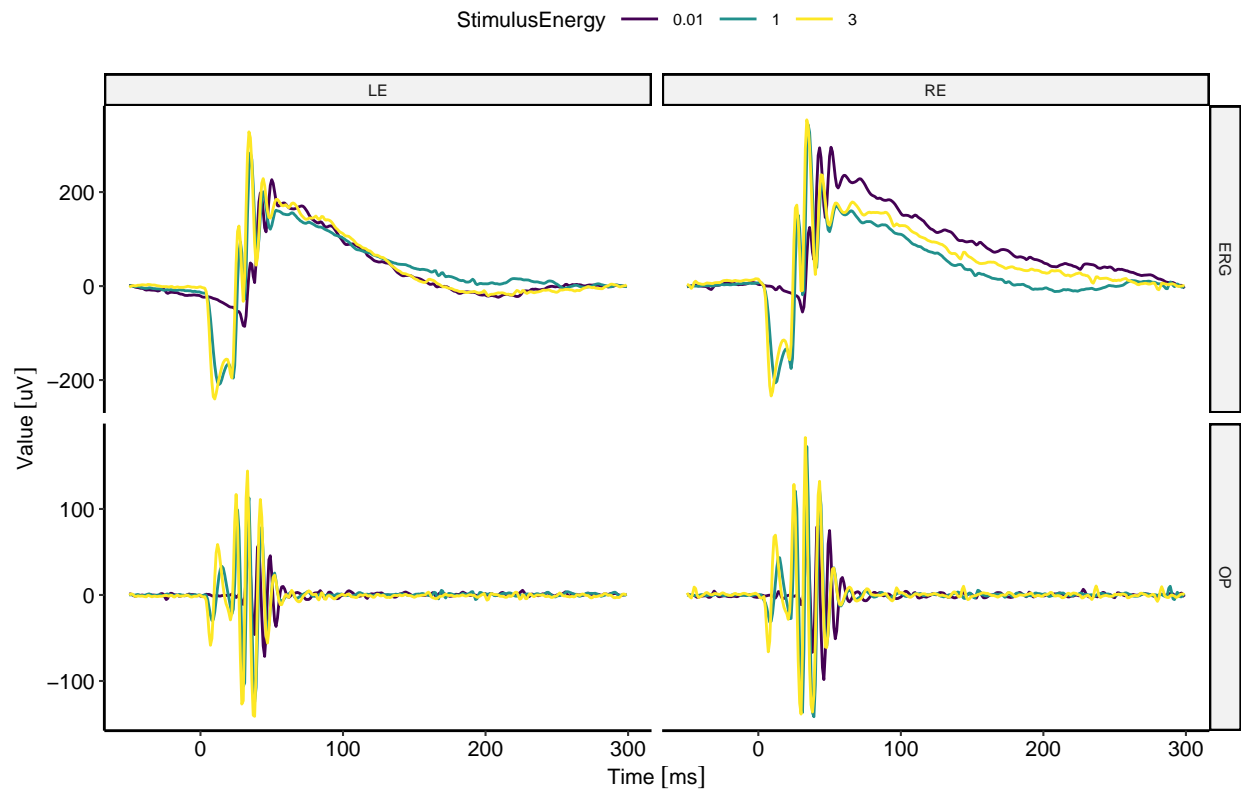
## DA-Flash



*# Do some stuff with the imported exam (See other example script for more details)*

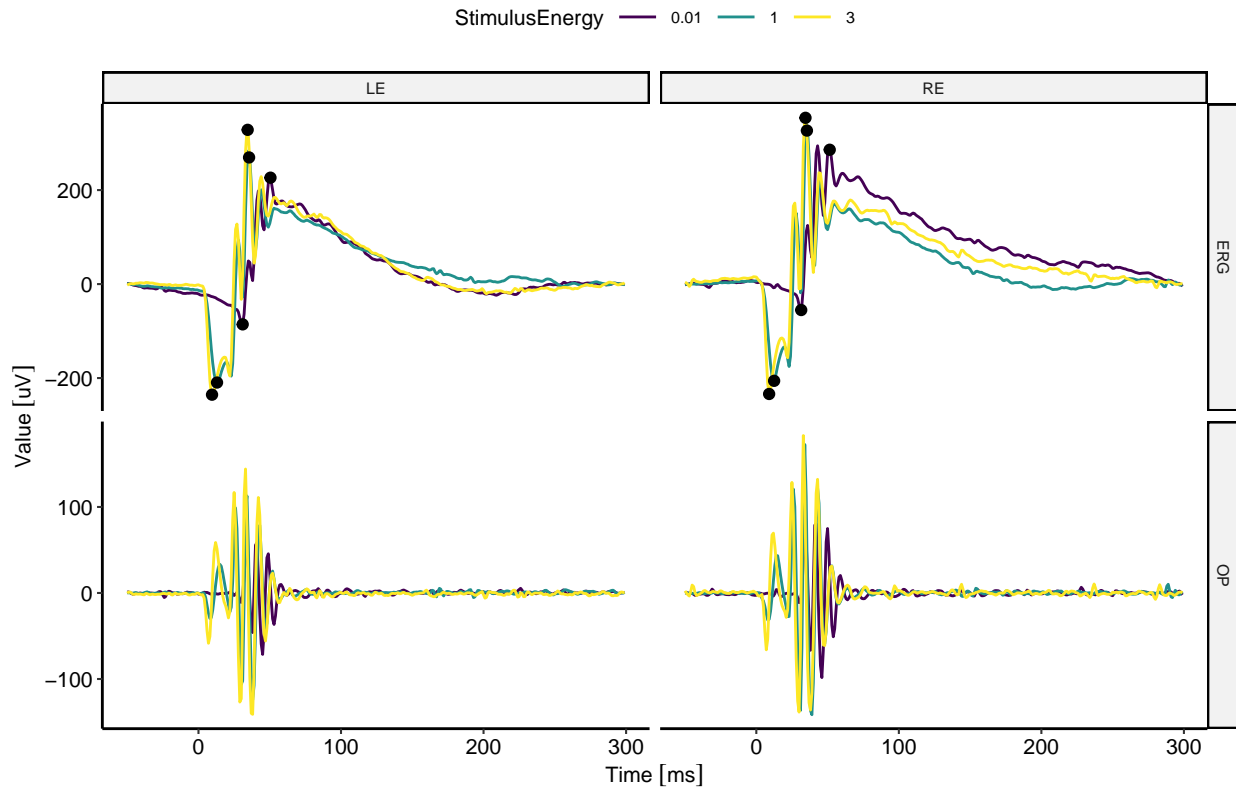
```
ImportedExam_Subset<-DropRecordings(ImportedExam, where=list(Background="LA")) # drop all recordings ac
ggERGExam(ImportedExam_Subset)
```

## DA-Flash



```
ImportedExam<-AutoPlaceMarkers(ImportedExam)
ggERGExam(ImportedExam)
```

## DA-Flash



## Session info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.5 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
##  [1] LC_CTYPE=de_DE.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8      LC_COLLATE=de_DE.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8  LC_MESSAGES=de_DE.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
```

```

## other attached packages:
## [1] stringr_1.5.1    tidyr_1.3.1      ERGtools2_0.8.5 EPhysData_0.9.7
## [5] units_0.8-0
##
## loaded via a namespace (and not attached):
## [1] httr_1.4.7          bit64_4.5.2      jsonlite_1.8.9    viridisLite_0.4.2
## [5] hdf5r_1.3.11        carData_3.0-5    Formula_1.2-5     shiny_1.9.1
## [9] TTR_0.24.4          highr_0.11       yaml_2.3.10       pillar_1.9.0
## [13] backports_1.5.0     lattice_0.20-45  glue_1.8.0        digest_0.6.37
## [17] promises_1.3.0      ggsignif_0.6.4   colorspace_2.1-1  EPhysMethods_0.3.1
## [21] htmltools_0.5.8.1  httpuv_1.6.15    pkgconfig_2.0.3   broom_1.0.7
## [25] purrr_1.0.2         xtable_1.8-4     scales_1.3.0      later_1.3.2
## [29] pracma_2.4.4        tibble_3.2.1     farver_2.1.2      generics_0.1.3
## [33] car_3.1-3           ggplot2_3.5.1    ggpubr_0.6.0      DT_0.33
## [37] withr_3.0.1         shinyjs_2.1.0    lazyeval_0.2.2    cli_3.6.3
## [41] quantmod_0.4.26     magrittr_2.0.3   mime_0.12          evaluate_1.0.1
## [45] fansi_1.0.6         MASS_7.3-55      rstatix_0.7.2     xts_0.14.1
## [49] tools_4.1.2         data.table_1.16.2 lifecycle_1.0.4    plotly_4.10.4.9000
## [53] munsell_0.5.1       compiler_4.1.2   signal_1.8-1       rlang_1.1.4
## [57] grid_4.1.2          rstudioapi_0.13  htmlwidgets_1.6.4 labeling_0.4.3
## [61] rmarkdown_2.28      gtable_0.3.5     abind_1.4-8        curl_5.2.3
## [65] R6_2.5.1            gridExtra_2.3    zoo_1.8-12         knitr_1.48
## [69] dplyr_1.1.4         fastmap_1.2.0    bit_4.5.0          utf8_1.2.4
## [73] stringi_1.8.4       Rcpp_1.0.13      vctrs_0.6.5        tidyselect_1.2.1
## [77] xfun_0.48

```