



hu-neuro-pipeline

A Python implementation of the single trial EEG pipeline by
Frömer et al. (*Front. Neurosci.*, 2018)

Alexander Enge

Neuro Lab @ Humboldt-Universität zu Berlin

20/04/2022

Why Python?



Why MNE-Python?

- Versatile
 - EEG, MEG, ECoG, fNIRS
 - Preprocessing, statistics, time-frequency analysis, visualization, machine learning, connectivity, source localization, ...
- Open source
 - 288 contributors on GitHub as of April 2022
 - Funding: NIH, NSF, ERC, Google, Amazon, ...
- Community standards
 - Code review, automatic tests, user forum, office hours, ...

Why the Frömer et al. (2018) pipeline?

- Allows single trial analysis of ERP amplitudes
 - Treat items as random effects (Bürki et al., 2018)
 - Model trial and item level covariates
 - Include continuous predictor variables
 - Handle unbalanced designs via partial pooling
 - Weaker assumptions than ANOVA

Why this re-implementation?

- User friendly, e.g.:
 - No MATLAB license; can be called from within R
 - Outputs readily usable for mixed models and plotting
- New features, e.g.:
 - Time-frequency analysis
 - Automatic ocular correction (ICA) + bad channel detection
- Code standards + versioning (<https://github.com/alexenge/hu-neuro-pipeline/>)

And why not?

- More difficult to debug or modify
- Possibly not all features supported (e.g., RIDE)
- EEGLAB still more widely use than MNE-Python

Installation

For Python users:

```
# Install via the command line from the Python Packaging Index (PyPI)  
python3 -m pip install hu-neuro-pipeline
```

For R users:

```
# Install reticulate for interfacing with Python from R  
install.packages("reticulate")  
  
# Install the Miniconda Python distribution  
reticulate::install_miniconda()  
  
# Install the actual package  
reticulate::py_install("hu-neuro-pipeline", pip = TRUE, python_version = "3.8")
```

General usage

```
# Import the Python package  
pipeline <- reticulate::import("pipeline")  
  
# Run the pipeline  
res <- pipeline$group_pipeline(...)
```


A simple example

```
# Import the Python package
pipeline <- reticulate::import("pipeline")

# Run the pipeline
res <- pipeline$group_pipeline(
  # Input/output paths
  vhdr_files = "data/raw",
  log_files = "data/log",
  output_dir = "output",
  # Preprocessing options
  besa_files = "data/cali",
  # Epoching options
  triggers = c(201:208, 211:218),
  components = list(
    "name" = list("N2", "P3b"),
    "tmin" = list(0.25, 0.4),
    "tmax" = list(0.35, 0.55),
    "roi" = list(
      c("FC1", "FC2", "C1", "C2", "Cz"),
      c("CP3", "CP1", "CPz", "CP2", "CP4", "P3", "Pz", "P4", "P03", "P0z", "P04")
    )
  ),
  # Averaging options
  average_by = c("n_b", "DeviantPosRL", "n_b/DeviantPosRL")
)
```

Pipeline inputs

```
# Input/output paths  
vhdr_files = "data/raw",  
log_files = "data/log",  
output_dir = "output",
```

- Directory or list of raw EEG files (.vhdr)
- Directory or list of behavioral log files (.txt/.tsv/.csv)
- Output directory

Pipeline inputs

```
# Preprocessing options  
besa_files = "data/cali",
```

- Ocular correction with BESA: Directory path or list of BESA files (.matrix)
- Alternatively: independent component analysis (e.g., `tfr_method = "fastica"`)
- Default bandpass filter (0.1–40 Hz)

Pipeline inputs

```
# Epoching options
triggers = c(201:208, 211:218),
components = list(
  "name" = list("N2", "P3b"),
  "tmin" = list(0.25, 0.4),
  "tmax" = list(0.35, 0.55),
  "roi" = list(
    c("FC1", "FC2", "C1", "C2", "Cz"),
    c("CP3", "CP1", "CPz", "CP2", "CP4", "P3", "Pz", "P4", "P03", "P0z", "P04")
  )
),
```

- List of numerical EEG triggers
- List of ERP component definitions:
 - name: Column names for each component
 - tmin + tmax: Onset and offset times (in s)
 - roi: List of channel names for each component

Pipeline inputs

```
# Averaging options  
average_by = c("n_b", "DeviantPosRL", "n_b/DeviantPosRL")
```

- List of column names (for main effects) and combinations of column names (for interaction effects, separated by "/")

More Pipeline inputs

- Downsampling (`downsample_sfreq`)
- Interpolate bad channels (`bad_channels`)
- Frequency filter (`highpass_freq`, `lowpass_freq`)
- Epoch duration (`epochs_tmin`, `epochs_tmax`)
- Baseline duration (`baseline`)
- Skip log file rows (`skip_log_rows`, `skip_log_conditions`)
- Threshold for artifact rejection (`reject_peak_to_peak`)

See <https://github.com/alexenge/hu-neuro-pipeline/blob/main/docs/inputs.md>

Pipeline outputs

Extract directly from the pipeline run:

```
trials <- res[[1]] # Single trial data frame  
evoked <- res[[2]] # Evoked data frame  
config <- res[[3]] # List of pipeline options
```

Or read from the output directory:

```
library(tidyverse)  
trials <- read_csv("output/trials.csv")  
evoked <- read_csv("output/ave.csv")  
config <- jsonlite::read_json("output/config.json")
```

See <https://github.com/alexenge/hu-neuro-pipeline/blob/main/docs/outputs.md>

Pipeline outputs

```
# Single trial data frame
```

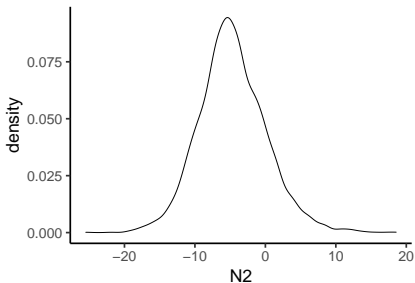
```
print(trials)
```

```
## # A tibble: 3,840 x 32
##   participa-1 VPNum-2 version   wdh lfdNr n_b   Stand-3 Deviant Objek-4 BedCo-5
##   <chr>         <dbl>   <dbl> <dbl> <dbl> <chr> <chr>   <chr>      <dbl> <chr>
## 1 05             5       1     1     1 norm- objekt- objekt-      8 gngf
## 2 05             5       1     1     2 blurr objekt- objekt-     10 un
## 3 05             5       1     1     3 blurr objekt- objekt-     10 un
## 4 05             5       1     1     4 norm- objekt- objekt-      3 un
## 5 05             5       1     1     5 norm- objekt- objekt-     15 unuf
## 6 05             5       1     1     6 norm- objekt- objekt-      7 unuf
## 7 05             5       1     1     7 blurr objekt- objekt-      2 un
## 8 05             5       1     1     8 norm- objekt- objekt-     16 gngf
## 9 05             5       1     1     9 norm- objekt- objekt-      9 gn
## 10 05            5       1     1    10 norm- objekt- objekt-     11 un
## # ... with 3,830 more rows, 22 more variables: BedCode_neu <chr>, bot <dbl>,
## #   DeviantPosRL <chr>, DeviantPosNR <dbl>, BedCodeRL <chr>, key <dbl>,
## #   ErrorCode <dbl>, RT <dbl>, Pos1 <chr>, Pos2 <chr>, Pos3 <chr>, Pos4 <chr>,
## #   Pos5 <chr>, Pos6 <chr>, Pos7 <chr>, Pos8 <chr>, Pos9 <chr>, Pos10 <chr>,
## #   Pos11 <chr>, Pos12 <chr>, N2 <dbl>, P3b <dbl>, and abbreviated variable
## #   names 1: participant_id, 2: VPNummer, 3: Standard, 4: Objektpaar,
## #   5: BedCode_alt
## # i Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable names
```


Pipeline outputs

```
# Single trial N2 mean amplitudes
ggplot(trials, aes(x = N2)) +
  geom_density() +
  theme_classic(base_size = 30)
```

```
## Warning: Removed 7 rows containing non-finite values (stat_density).
```



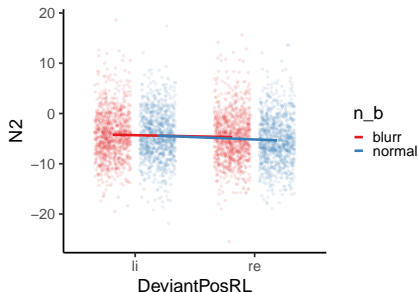
Pipeline outputs

```
# Linear mixed-effects model
form <- N2 ~ n_b * DeviantPosRL + (1 | participant_id)
mod <- lme4::lmer(form, trials)
summary(mod)

## Linear mixed model fit by REML ['lmerMod']
## Formula: N2 ~ n_b * DeviantPosRL + (1 | participant_id)
## Data: trials
##
## REML criterion at convergence: 22696.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -4.6856 -0.6130 -0.0002  0.6148  5.1121
##
## Random effects:
## Groups           Name          Variance Std.Dev.
## participant_id (Intercept)  2.287    1.512
## Residual                21.780    4.667
## Number of obs: 3833, groups: participant_id, 2
##
## Fixed effects:
##              Estimate Std. Error t value
## (Intercept)   -4.2285    1.0800  -3.915
## n_bnormal     -0.1796    0.2132  -0.842
## DeviantPosRLre -0.4587    0.2132  -2.151
## n_bnormal:DeviantPosRLre -0.4732    0.3015  -1.569
##
```

Pipeline outputs

```
# Single trial N2 mean amplitudes by condition
ggplot(trials, aes(x = DeviantPosRL, y = N2, color = n_b, group = n_b)) +
  geom_point(position = position_jitterdodge(0.3), alpha = 0.1) +
  stat_summary(
    geom = "line",
    size = 2.,
    position = position_dodge(0.75)
  ) +
  theme_classic(base_size = 30)
```



Pipeline outputs

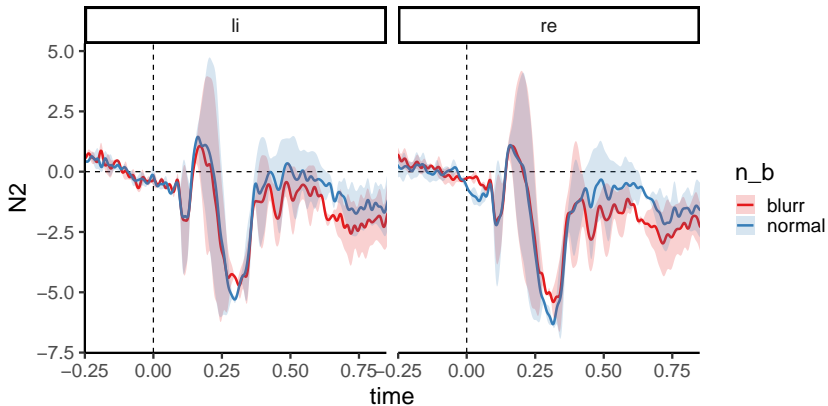
```
# Evoked by participant and condition
print(evokeds)
```

```
## # A tibble: 16,000 x 70
##   particip-1 avera-2 n_b Devia-3 time Fp1 Fpz Fp2 AF7 AF3 AFz
##   <chr> <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 05 n_b norm~ <NA> -0.5 -0.751 -1.49 -1.35 -1.14 -1.94 -1.86
## 2 05 n_b norm~ <NA> -0.498 -0.779 -1.46 -1.30 -1.16 -2.04 -1.93
## 3 05 n_b norm~ <NA> -0.496 -0.809 -1.43 -1.25 -1.15 -2.11 -2.00
## 4 05 n_b norm~ <NA> -0.494 -0.836 -1.40 -1.21 -1.13 -2.14 -2.06
## 5 05 n_b norm~ <NA> -0.492 -0.859 -1.39 -1.17 -1.10 -2.14 -2.10
## 6 05 n_b norm~ <NA> -0.49 -0.876 -1.39 -1.15 -1.05 -2.10 -2.12
## 7 05 n_b norm~ <NA> -0.488 -0.886 -1.42 -1.14 -1.01 -2.06 -2.12
## 8 05 n_b norm~ <NA> -0.486 -0.893 -1.46 -1.15 -0.976 -2.00 -2.10
## 9 05 n_b norm~ <NA> -0.484 -0.896 -1.51 -1.17 -0.952 -1.95 -2.09
## 10 05 n_b norm~ <NA> -0.482 -0.898 -1.56 -1.19 -0.942 -1.90 -2.07
## # ... with 15,990 more rows, 59 more variables: AF4 <dbl>, AF8 <dbl>, F9 <dbl>,
## # F7 <dbl>, F5 <dbl>, F3 <dbl>, Fz <dbl>, F4 <dbl>, F6 <dbl>, F8 <dbl>,
## # F10 <dbl>, FT7 <dbl>, FC5 <dbl>, FC3 <dbl>, FC1 <dbl>, FC2 <dbl>,
## # FC4 <dbl>, FC6 <dbl>, FT8 <dbl>, T7 <dbl>, C5 <dbl>, C3 <dbl>, C1 <dbl>,
## # Cz <dbl>, C2 <dbl>, C4 <dbl>, C6 <dbl>, T8 <dbl>, TP9 <dbl>, TP7 <dbl>,
## # CP5 <dbl>, CP3 <dbl>, CP1 <dbl>, CPz <dbl>, CP2 <dbl>, CP4 <dbl>,
## # CP6 <dbl>, TP8 <dbl>, TP10 <dbl>, P7 <dbl>, P5 <dbl>, P3 <dbl>, ...
## # i Use 'print(n = ...)' to see more rows, and 'colnames()' to see all variable names
```

Pipeline outputs

```
# Evoked by participant/condition
evoked %>%
  filter(average_by == "n_b/DeviantPosRL") %>%
  Rmisc::summarySEwithin(
    measurevar = "N2",
    withinvars = c("time", "n_b", "DeviantPosRL"),
    idvar = "participant_id"
  ) %>%
  mutate(time = as.numeric(levels(time))[time]) %>%
  ggplot(aes(
    x = time,
    y = N2,
    ymin = N2 - se,
    ymax = N2 + se,
    color = n_b,
    fill = n_b
  )) +
  facet_wrap(~DeviantPosRL) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_vline(xintercept = 0, linetype = "dashed") +
  geom_line(size = 1) +
  geom_ribbon(color = NA, alpha = 0.2) +
  coord_cartesian(xlim = c(-0.2, 0.8)) +
  theme_classic(base_size = 20)
```

Pipeline outputs



Pipeline outputs

List of pipeline options

```
names(config)
```

```
## [1] "vhdr_files"          "log_files"          "output_dir"
## [4] "clean_dir"           "epochs_dir"         "report_dir"
## [7] "to_df"               "downsample_sfreq"   "veog_channels"
## [10] "heog_channels"       "montage"            "bad_channels"
## [13] "besa_files"          "ica_method"         "ica_n_components"
## [16] "highpass_freq"       "lowpass_freq"       "triggers"
## [19] "triggers_column"     "epochs_tmin"        "epochs_tmax"
## [22] "baseline"            "skip_log_rows"      "skip_log_conditions"
## [25] "reject_peak_to_peak" "components"         "average_by"
## [28] "perform_tfr"         "tfr_subtract_evoked" "tfr_freqs"
## [31] "tfr_cycles"          "tfr_mode"           "tfr_baseline"
## [34] "tfr_components"      "perm_contrasts"     "perm_tmin"
## [37] "perm_tmax"           "perm_channels"      "perm_fmin"
## [40] "perm_fmax"           "n_jobs"             "auto_rejected_epochs"
```

Number of rejected epochs per participant

```
lengths(config$auto_rejected_epochs)
```

```
## 05 07
```

```
## 7 0
```

Automated QC reports

```
# Input/output paths  
report_dir = "output/qc_reports",
```


Cluster-based permutation tests

```
# Permutation test options
perm_contrasts = list(
  c("blurr", "normal"),
  c("blurr/re", "blurr/li"),
  c("normal/re", "normal/li")
)
```

```
# Permutation test outputs
clusters <- read_csv("output/clusters.csv") # or clusters <- res[[4]]
print(na.omit(clusters))
```

```
## # A tibble: 5,748 x 6
##   contrast      time channel  t_obs cluster p_val
##   <chr>      <dbl> <chr>   <dbl> <chr>   <dbl>
## 1 blurr - normal 0      AF3    -18.1 neg_282 1
## 2 blurr - normal 0      FT7     52.8 pos_1 1
## 3 blurr - normal 0      C6     -30.0 neg_281 1
## 4 blurr - normal 0      POz     15.3 pos_96 1
## 5 blurr - normal 0.002 F10   -24.5 neg_280 1
## 6 blurr - normal 0.002 FT7    20.1 pos_1 1
## 7 blurr - normal 0.002 FC3    17.5 pos_95 1
## 8 blurr - normal 0.002 FC1    22.1 pos_95 1
## 9 blurr - normal 0.002 C6    -154. neg_281 1
## 10 blurr - normal 0.002 CPz    16.6 pos_94 1
## # ... with 5,738 more rows
## # i Use 'print(n = ...)' to see more rows
```

Automated tools

- Reject bad epochs (`reject_peak_to_peak = 200.0`)
 - Using per-channel peak-to-peak amplitudes
- Ocular correction (`ica_method = "fastica"`)
 - FastICA (Hyvärinen, 1999) + correlation with HEOG/VEOG
 - Can specify different methods + number of principal components (via `ica_n_components`)
- Interpolate bad channels (`bad_channel = "auto"`)
 - Based on per-channel standard error across epochs

One more thing

```
# Auto-match log files to triggers  
triggers_column = "trigger",
```

- Have such a column?
Great!
- If not, create in R and pass
data frames as log_files

Plans

- Improve documentation + tests
- More detailed QC reports
- Mixed models with `pymr4` (?)
- Better permutation tests (Frossard & Renaud, 2021, 2022)
- BIDS interface
- Your ideas + contributions?

Thanks



References

- Bürki, A., Frossard, J., & Renaud, O. (2018). Accounting for stimulus and participant effects in event-related potential analyses to increase the replicability of studies. *Journal of Neuroscience Methods*, 309, 218–227. <https://doi.org/10.1016/j.jneumeth.2018.09.016>
- Frömer, R., Maier, M., & Abdel Rahman, R. (2018). Group-level EEG-processing pipeline for flexible single trial-based analyses including linear mixed models. *Frontiers in Neuroscience*, 12, 48. <https://doi.org/10.3389/fnins.2018.00048>
- Frossard, J., & Renaud, O. (2021). Permutation tests for regression, ANOVA, and comparison of signals: The permuco package. *Journal of Statistical Software*, 99, 1–32. <https://doi.org/10.18637/jss.v099.i15>
- Frossard, J., & Renaud, O. (2022). The cluster depth tests: Toward point-wise strong control of the family-wise error rate in massively univariate tests with application to M/EEG. *NeuroImage*, 247, 118824. <https://doi.org/10.1016/j.neuroimage.2021.118824>
- Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3), 626–634. <https://doi.org/10.1109/72.761722>