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


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

# Run the pipeline
res <- pipeline$group_pipeline(...)
```

A simple example


```r
# 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
  ocular_correction = "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", "PO3", "POz", "PO4")
    )
  ),
  # 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
ocular_correction = "data/cali",
```

- Ocular correction:
  - Path or list of BESA files (`.matrix`) or
  - `"auto"` for independent component analysis (ICA)
- 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", "PO3", "POz", "PO4")
  )
),
```

- 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_tmin`, `baseline_tmax`)
- 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
evokeds <- res[[2]]  # Evokeds data frame
config <- res[[3]]   # List of pipeline options
```

Or read from the output directory:

```
library(tidyverse)
trials <- read_csv("output/trials.csv")
evokeds <- 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

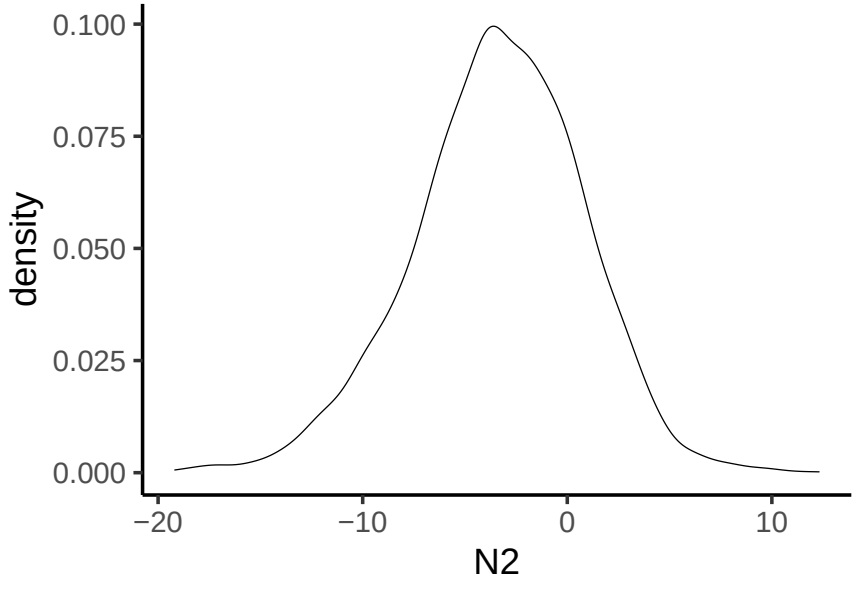

```
# Single trial data frame
print(trials)
```

```
## # A tibble: 3,840 x 33
##    participant_id VPNummer version   wdh lfdNr n_b   Standard Deviant Objektpaar
##    <chr>              <dbl>   <dbl> <dbl> <dbl> <chr> <chr>    <chr>        <dbl>
##  1 09                     9       1     1     1 norm~ objekt5~ objekt~          6
##  2 09                     9       1     1     2 blurr objekt5~ objekt~          6
##  3 09                     9       1     1     3 blurr objekt3~ objekt~          4
##  4 09                     9       1     1     4 blurr objekt4~ objekt~         11
##  5 09                     9       1     1     5 norm~ objekt3~ objekt~         10
##  6 09                     9       1     1     6 blurr objekt1~ objekt~          1
##  7 09                     9       1     1     7 blurr objekt7~ objekt~          8
##  8 09                     9       1     1     8 norm~ objekt3~ objekt~          4
##  9 09                     9       1     1     9 blurr objekt4~ objekt~         12
## 10 09                     9       1     1    10 norm~ objekt3~ objekt~          4
## # ... with 3,830 more rows, and 24 more variables: BedCode_alt <chr>,
## #   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>, P1 <dbl>
```

Pipeline outputs

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

```
## Warning: Removed 73 rows containing non-finite values (stat_density).
```



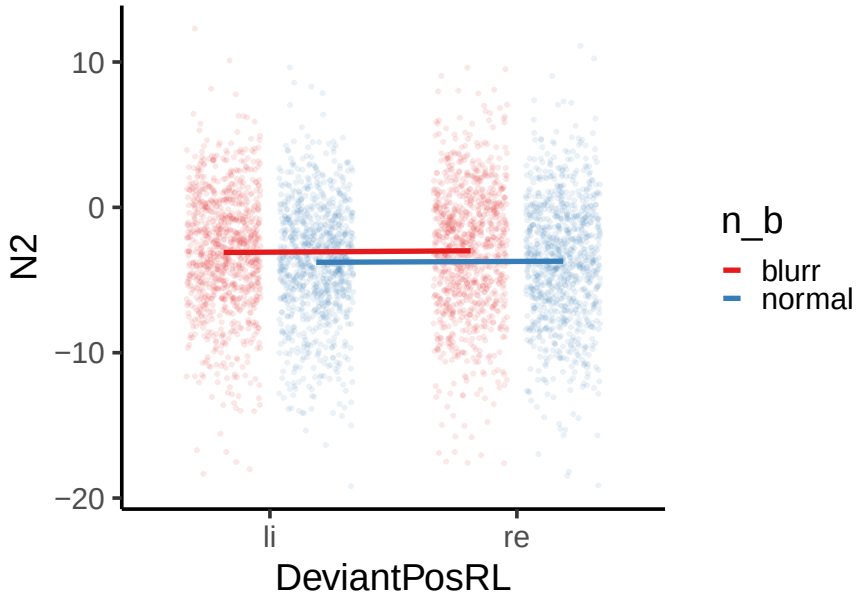
Pipeline outputs


```r
# 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: 21350.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.6932 -0.6409  0.0206  0.6501  3.9292
##
## Random effects:
##  Groups         Name        Variance Std.Dev.
##  participant_id (Intercept)  1.141   1.068
##  Residual                   16.900   4.111
## Number of obs: 3767, groups:  participant_id, 2
##
## Fixed effects:
##                            Estimate Std. Error t value
## (Intercept)                -3.09231    0.76708  -4.031
## n_bnormal                  -0.67552    0.18932  -3.568
## DeviantPosRLre              0.11436    0.18952   0.603
## n_bnormal:DeviantPosRLre -0.04093    0.26792  -0.153
##
```

Pipeline outputs

```r
# 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

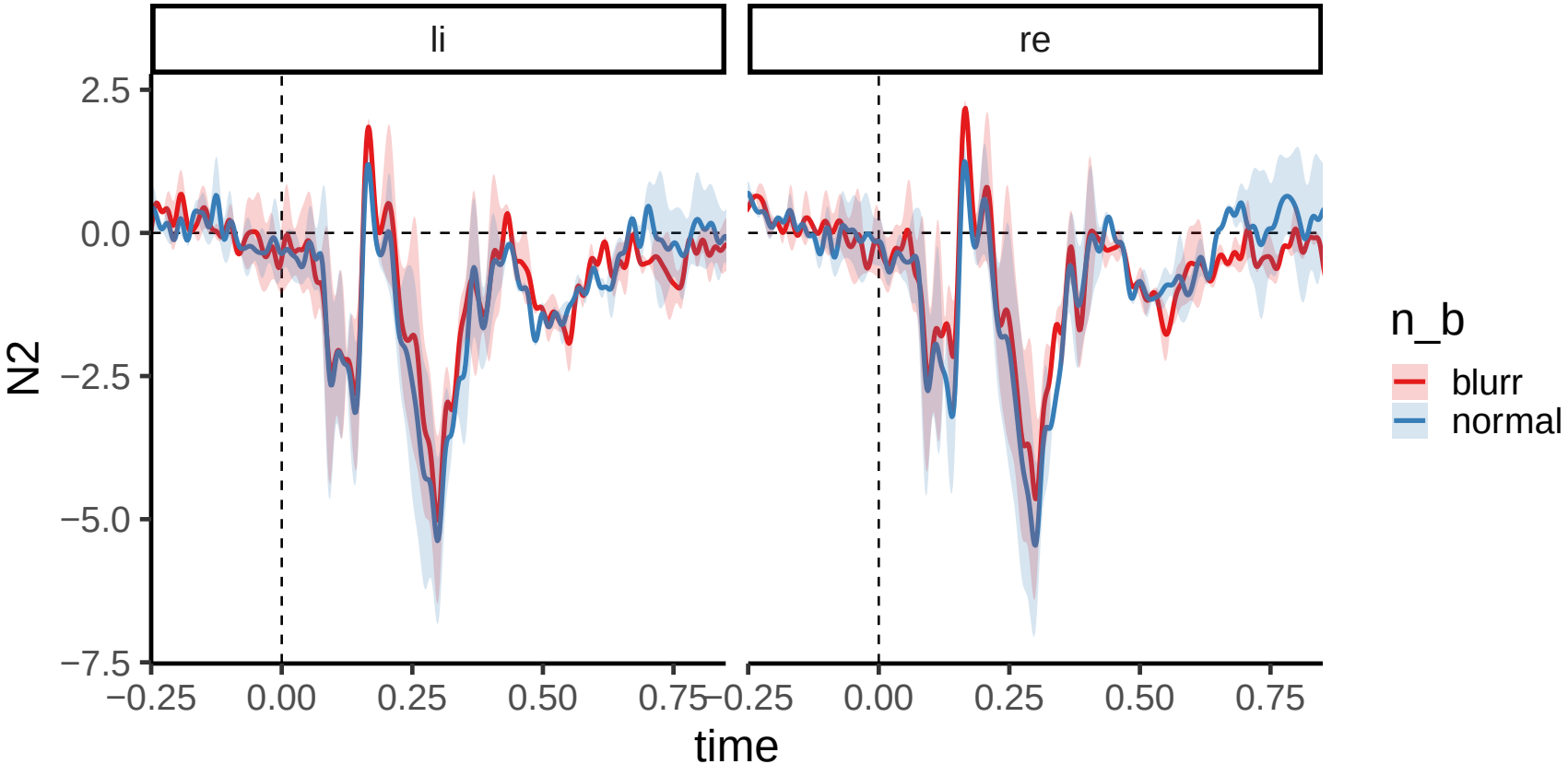

```
# Evokeds by participant and condition
print(evokeds)
```

```
## # A tibble: 12,000 x 71
##    participant_id average_by n_b      time     Fp1    Fpz    Fp2   AF7   AF3
##    <chr>          <chr>      <chr>   <dbl>   <dbl>  <dbl>  <dbl> <dbl> <dbl>
##  1 09             n_b        normal -0.5    0.790  -0.478 -0.539 0.353 0.459
##  2 09             n_b        normal -0.498  0.765  -0.438 -0.623 0.405 0.544
##  3 09             n_b        normal -0.496  0.648  -0.376 -0.647 0.446 0.537
##  4 09             n_b        normal -0.494  0.466  -0.315 -0.626 0.477 0.460
##  5 09             n_b        normal -0.492  0.252  -0.277 -0.577 0.492 0.347
##  6 09             n_b        normal -0.49   0.0393 -0.273 -0.512 0.487 0.237
##  7 09             n_b        normal -0.488 -0.145  -0.300 -0.439 0.458 0.163
##  8 09             n_b        normal -0.486 -0.286  -0.345 -0.357 0.402 0.140
##  9 09             n_b        normal -0.484 -0.380  -0.389 -0.265 0.318 0.162
## 10 09             n_b        normal -0.482 -0.437  -0.413 -0.160 0.210 0.206
## # ... with 11,990 more rows, and 62 more variables: AFz <dbl>, 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>, ...
```

Pipeline outputs


```r
# Evokeds by participant/condition
evokeds %>%
  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] "ocular_correction"   "highpass_freq"       "lowpass_freq"
## [16] "triggers"            "triggers_column"     "epochs_tmin"
## [19] "epochs_tmax"         "baseline"            "skip_log_rows"
## [22] "skip_log_conditions" "reject_peak_to_peak" "components"
## [25] "average_by"          "perform_tfr"         "tfr_subtract_evoked"
## [28] "tfr_freqs"           "tfr_cycles"          "tfr_baseline"
## [31] "tfr_components"      "perm_contrasts"      "perm_tmin"
## [34] "perm_tmax"           "perm_channels"       "perm_fmin"
## [37] "perm_fmax"           "n_jobs"              "rejected_epochs"
```

```
# Number of rejected epochs per participant
lengths(config$rejected_epochs)
```

```
## 09 47
## 66  7
```

Automated QC reports


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

Cluster-based permutation tests


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

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

```
## # A tibble: 4,991 x 6
##    contrast        time channel t_obs cluster p_val
##    <chr>          <dbl> <chr>   <dbl> <chr>   <dbl>
##  1 blurr - normal 0     FT7      58.3 pos_1       1
##  2 blurr - normal 0     P6      -14.8 neg_92      1
##  3 blurr - normal 0.002 T8      -90.9 neg_94      1
##  4 blurr - normal 0.002 CP1     -17.6 neg_101     1
##  5 blurr - normal 0.004 T8      -22.0 neg_94      1
##  6 blurr - normal 0.004 CP4     -26.1 neg_95      1
##  7 blurr - normal 0.004 CP6     -18.0 neg_95      1
##  8 blurr - normal 0.006 T8      -30.6 neg_94      1
##  9 blurr - normal 0.006 P7       71.9 pos_101     1
## 10 blurr - normal 0.008 TP9     110.  pos_102     1
## # ... with 4,981 more rows
```

Automated tools

- Reject bad epochs (`reject_peak_to_peak = 200`)
  - Using per-channel peak-to-peak amplitudes
- Ocular correction (`ocular_correction = "auto"`)
  - FastICA (Hyvärinen, 1999) + correlation with HEOG/VEOG
- 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 `pymer4` (?)
- 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