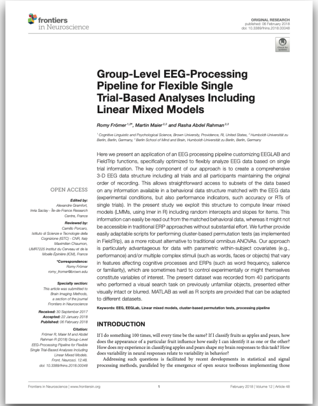
Analyzing single trial EEG data using the `hu-neuro-pipeline` package

Alexander Enge & Kirsten Stark

Neuro Lab @ Humboldt-Universität zu Berlin

2023-12-13

The Frömer et al. (2018) pipeline



frontiers in Neuroscience

ORIGINAL RESEARCH
published: 06 February 2018
doi: 10.3389/fnins.2018.00048

Group-Level EEG-Processing Pipeline for Flexible Single Trial-Based Analyses Including Linear Mixed Models

Romy Frömer[1,2*], *Martin Maier*[2,3] and *Rasha Abdel Rahman*[2,3]

[1] Cognitive Linguistic and Psychological Science, Brown University, Providence, RI, United States, [2] Humboldt-Universität zu Berlin, Berlin, Germany, [3] Berlin School of Mind and Brain, Humboldt-Universität zu Berlin, Berlin, Germany

Here we present an application of an EEG processing pipeline customizing EEGLAB and FieldTrip functions, specifically optimized to flexibly analyze EEG data based on single trial information. The key component of our approach is to create a comprehensive 3-D EEG data structure including all trials and all participants maintaining the original order of recording. This allows straightforward access to subsets of the data based on any information available in a behavioral data structure matched with the EEG data (experimental conditions, but also performance indicators, such accuracy or RTs of single trials). In the present study we exploit this structure to compute linear mixed models (LMMs, using lmer in R) including random intercepts and slopes for items. This information can easily be read out from the matched behavioral data, whereas it might not be accessible in traditional ERP approaches without substantial effort. We further provide easily adaptable scripts for performing cluster-based permutation tests (as implemented in FieldTrip), as a more robust alternative to traditional omnibus ANOVAs. Our approach is particularly advantageous for data with parametric within-subject covariates (e.g., performance) and/or multiple complex stimuli (such as words, faces or objects) that vary in features affecting cognitive processes and ERPs (such as word frequency, salience or familiarity), which are sometimes hard to control experimentally or might themselves constitute variables of interest. The present dataset was recorded from 40 participants who performed a visual search task on previously unfamiliar objects, presented either visually intact or blurred. MATLAB as well as R scripts are provided that can be adapted to different datasets.

Keywords: EEG, EEGLab, Linear mixed models, cluster-based permutation tests, processing pipeline

OPEN ACCESS

Edited by:
Alexandre Gramfort, Inria Saclay - Île-de-France Research Centre, France

Reviewed by:
Camillo Porcaro, Istituto di Scienze e Tecnologie della Cognizione (ISTC) - CNR, Italy
Maximilien Chaumon, UMR7225 Institut du Cerveau et de la Moelle Épinière (ICM), France

***Correspondence:**
Romy Frömer
romy_fromer@brown.edu

Specialty section:
This article was submitted to Brain Imaging Methods, a section of the journal Frontiers in Neuroscience

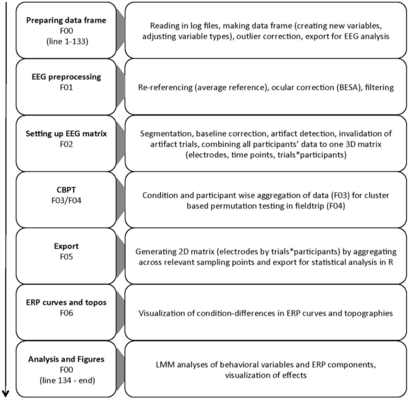
Received: *30 September 2017*
Accepted: *22 January 2018*
Published: *06 February 2018*

Citation:
Frömer R, Maier M and Abdel Rahman R (2018) Group-Level EEG-Processing Pipeline for Flexible Single Trial-Based Analyses Including Linear Mixed Models. Front. Neurosci. 12:48. doi: 10.3389/fnins.2018.00048

INTRODUCTION

If I do something 100 times, will every time be the same? If I classify fruits as apples and pears, how does the appearance of a particular fruit influence how easily I can identify it as one or the other? How does my experience in classifying apples and pears shape my brain responses to this task? How does variability in neural responses relate to variability in behavior?

Addressing such questions is facilitated by recent developments in statistical and signal processing methods, paralleled by the emergence of open source toolboxes implementing those



The Frömer et al. (2018) pipeline

- Allows single trial analysis of ERP amplitudes
  - Random effects for items (Bürki et al., 2018)
  - Trial and item level covariates (Volpert-Esmond et al., 2021)
  - Continuous predictor variables
  - Unbalanced designs
  - Brain–behavior associations

Python implementation


frontiers in Neuroscience

ORIGINAL RESEARCH
published: 06 February 2018
doi: 10.3389/fnins.2018.00048

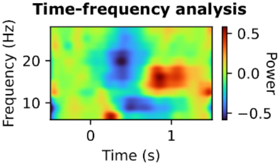
Group-Level EEG-Processing Pipeline for Flexible Single Trial-Based Analyses Including Linear Mixed Models

Romy Frömer[1,2*], Martin Maier[2,3] and Rasha Abdel Rahman[2,3]

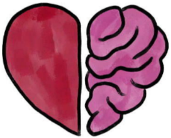
[1] Cognitive Linguistic and Psychological Science, Brown University, Providence, RI, United States, [2] Humboldt-Universität zu Berlin, Berlin, Germany, [3] Berlin School of Mind and Brain, Humboldt-Universität zu Berlin, Berlin, Germany

+

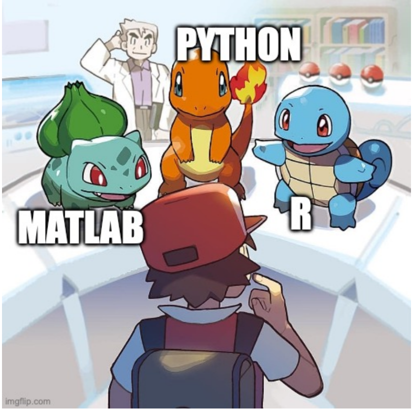
+



=



Python, I choose you!



Blog post: https://dominiquemakowski.github.io/post/2020-05-22-r_or_python
Online course: https://swcarpentry.github.io/python-novice-inflammation

MNE-Python

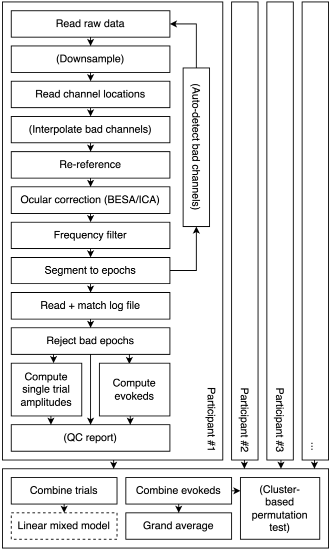
- Versatile
  - EEG, MEG, ECoG, fNIRS
  - Preprocessing, statistics, time-frequency analysis, visualization, machine learning, connectivity, source localization, ...
- Open source
  - $>$ 350 contributors on GitHub (December 2023)
  - Funded by NIH, NSF, ERC, Google, Amazon, ...
  - Code review, automated tests, user forum, office hours, ...

MEG + EEG ANALYSIS & VISUALIZATION

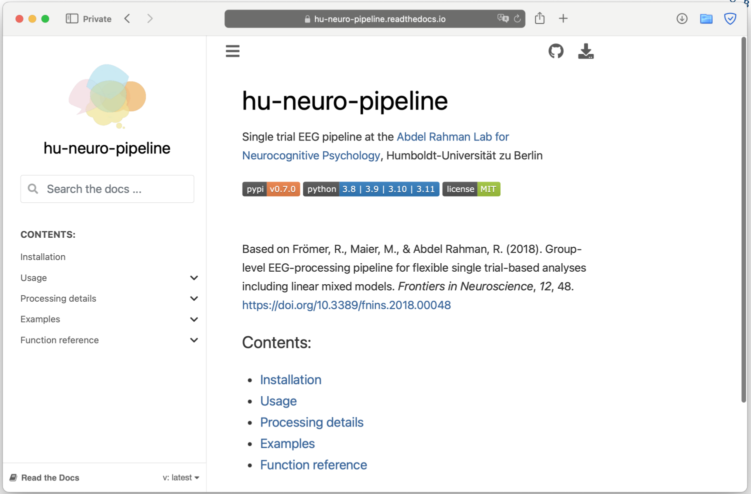
Python implementation

- No MATLAB required
- No Python skills required – can be called from R
- New features:
  - Time-frequency analysis
  - Fully automatic ocular correction (ICA)
  - Automatic bad channel detection
  - Automatic missing trial detection
  - Example datasets
- Code standards, documentation, version control
(https://github.com/alexenge/hu-neuro-pipeline/)

Python implementation



Python implementation



Documentation at https://hu-neuro-pipeline.readthedocs.io

Installation

For Python users:

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

For R users:

```r
# Install reticulate for interfacing with Python from R
install.packages("reticulate")

# Install Python (Miniconda distribution)
reticulate::install_miniconda()

# Install the actual package from PyPI
reticulate::py_install("hu-neuro-pipeline", pip = TRUE, python_version = "3.8")
```

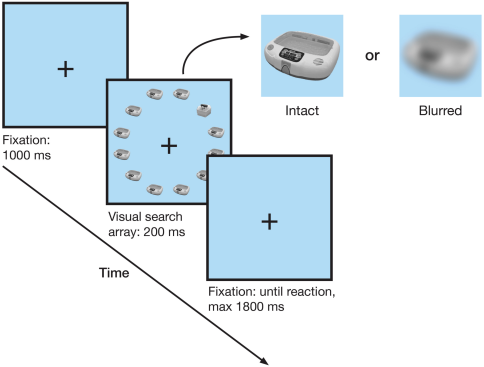
General usage

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

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

Minimal example

```r
ucap_paths <- pipeline$datasets$get_ucap(participants = 2, path = "data")
```



For details, see Frömer et al. (2018)

Minimal example

```r
# Run the pipeline
res <- pipeline$group_pipeline(
  # Input/output paths
  raw_files = "data/ucap/raw",
  log_files = "data/ucap/log",
  output_dir = "output",
  # Preprocessing options
  besa_files = "data/ucap/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 = list(
    blurr_left = "n_b == 'blurr' and DeviantPosRL == 'li' and RT > 200",
    blurr_right = "n_b == 'blurr' and DeviantPosRL == 're' and RT > 200",
    normal_right = "n_b == 'normal' and DeviantPosRL == 're' and RT > 200",
    normal_left = "n_b == 'normal' and DeviantPosRL == 're' and RT > 200"
  )
)
```

Pipeline inputs

```
# Input/output paths
raw_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",
```

- Directory path or list of BESA files (`.matrix`)
- Default bandpass filter (0.1–40 Hz)
- Default re-referencing (common average)

Pipeline inputs

```r
# 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`: Custom label for each component
  - `tmin` + `tmax`: Onset and offset times (in s)
  - `roi`: List of channel names for each component

Pipeline inputs

```r
# 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")
  )
),
```

- Default baseline correction (-0.2 – 0.0 s)
- Default rejection of bad epochs (peak-to-peak ampl. $>$ 200 μV)

Pipeline inputs

```
# Averaging options
average_by = list(
  blurr_left = "n_b == 'blurr' and DeviantPosRL == 'li' and RT > 200",
  blurr_right = "n_b == 'blurr' and DeviantPosRL == 're' and RT > 200",
  normal_right = "n_b == 'normal' and DeviantPosRL == 're' and RT > 200",
  normal_left = "n_b == 'normal' and DeviantPosRL == 're' and RT > 200"
)
```

- List with all (combinations of) conditions to create by-participant averages for:
  - List names are custom labels for each average
  - List values are query strings to select log file rows (trials/epochs)

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://hu-neuro-pipeline.readthedocs.io/en/latest/usage_inputs.html

Pipeline outputs

Extract directly from the pipeline run:

```r
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:

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

See https://hu-neuro-pipeline.readthedocs.io/en/latest/usage_outputs.html

Pipeline outputs

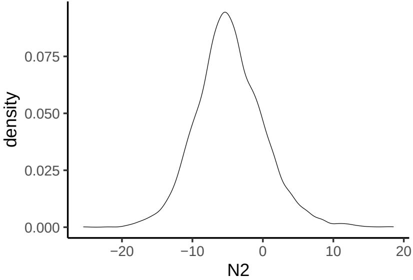
```r
# Single trial data frame
print(trials)
```

```
## # A tibble: 3,840 x 32
##    participant_id VPNummer version   wdh lfdNr n_b   Standard Deviant Objektpaar
##    <chr>             <dbl>   <dbl> <dbl> <dbl> <chr> <chr>    <chr>        <dbl>
##  1 05                    5       1     1     1 norm~ objekt7~ objekt~          8
##  2 05                    5       1     1     2 blurr objekt3~ objekt~         10
##  3 05                    5       1     1     3 blurr objekt3~ objekt~         10
##  4 05                    5       1     1     4 norm~ objekt2~ objekt~          3
##  5 05                    5       1     1     5 norm~ objekt8~ objekt~         15
##  6 05                    5       1     1     6 norm~ objekt6~ objekt~          7
##  7 05                    5       1     1     7 blurr objekt1~ objekt~          2
##  8 05                    5       1     1     8 norm~ objekt8~ objekt~         16
##  9 05                    5       1     1     9 norm~ objekt2~ objekt~          9
## 10 05                    5       1     1    10 norm~ objekt4~ objekt~         11
## # i 3,830 more rows
## # i 23 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>
```

Pipeline outputs

```r
# 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()`).
```



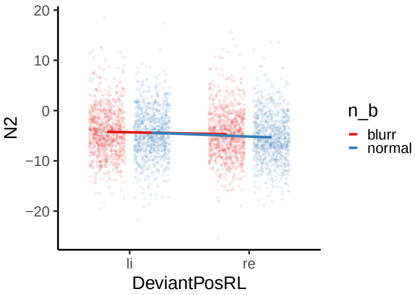
Pipeline outputs

```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: 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",
    linewidth = 2.0,
    position = position_dodge(0.75)
  ) +
  theme_classic(base_size = 30)
```



Pipeline outputs

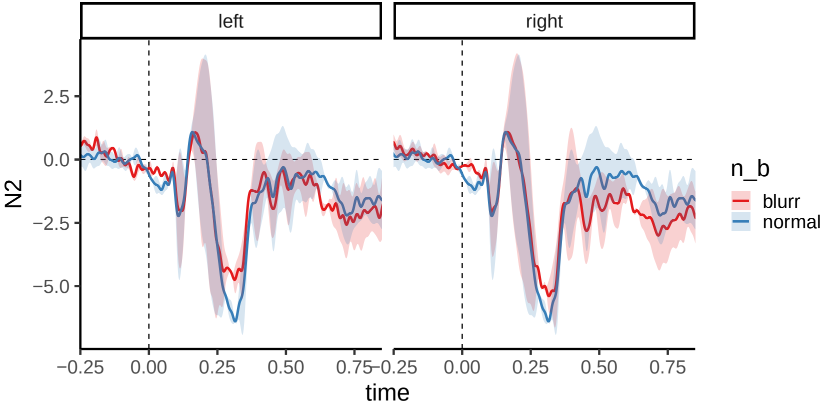
```r
# Evokeds by participant and condition
print(evokeds)
```

```
## # A tibble: 8,000 x 69
##    participant_id label      query   time    Fp1   Fpz   Fp2    AF7    AF3   AFz
##    <chr>          <chr>      <chr>  <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
##  1 05             blurr_left n_b ~ -0.5   -0.559 -1.41 -1.45 -0.617 -1.12  -1.49
##  2 05             blurr_left n_b ~ -0.498 -0.550 -1.37 -1.35 -0.640 -0.930 -1.45
##  3 05             blurr_left n_b ~ -0.496 -0.528 -1.34 -1.28 -0.666 -0.788 -1.43
##  4 05             blurr_left n_b ~ -0.494 -0.486 -1.31 -1.23 -0.689 -0.722 -1.42
##  5 05             blurr_left n_b ~ -0.492 -0.427 -1.29 -1.21 -0.701 -0.743 -1.40
##  6 05             blurr_left n_b ~ -0.49  -0.355 -1.27 -1.22 -0.698 -0.838 -1.39
##  7 05             blurr_left n_b ~ -0.488 -0.283 -1.26 -1.24 -0.675 -0.977 -1.38
##  8 05             blurr_left n_b ~ -0.486 -0.222 -1.26 -1.27 -0.633 -1.12  -1.37
##  9 05             blurr_left n_b ~ -0.484 -0.187 -1.26 -1.30 -0.577 -1.24  -1.36
## 10 05             blurr_left n_b ~ -0.482 -0.185 -1.26 -1.34 -0.515 -1.31  -1.35
## # i 7,990 more rows
## # i 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>, ...
```

Pipeline outputs

```r
# Time course plot with within-participant standard errors
evokeds |>
  separate_wider_delim(label, delim = "_", names = c("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(linewidth = 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

```r
# List of pipeline options
names(config)
```

```
##  [1] "raw_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] "ref_channels"         "besa_files"           "ica_method"
## [16] "ica_n_components"     "highpass_freq"        "lowpass_freq"
## [19] "triggers"             "triggers_column"      "epochs_tmin"
## [22] "epochs_tmax"          "baseline"             "skip_log_rows"
## [25] "skip_log_conditions"  "reject_peak_to_peak"  "components"
## [28] "average_by"           "perform_tfr"          "tfr_subtract_evoked"
## [31] "tfr_freqs"            "tfr_cycles"           "tfr_mode"
## [34] "tfr_baseline"         "tfr_components"       "perm_contrasts"
## [37] "perm_tmin"            "perm_tmax"            "perm_channels"
## [40] "perm_fmin"            "perm_fmax"            "n_jobs"
## [43] "vhdr_files"           "auto_rejected_epochs" "package_versions"
```

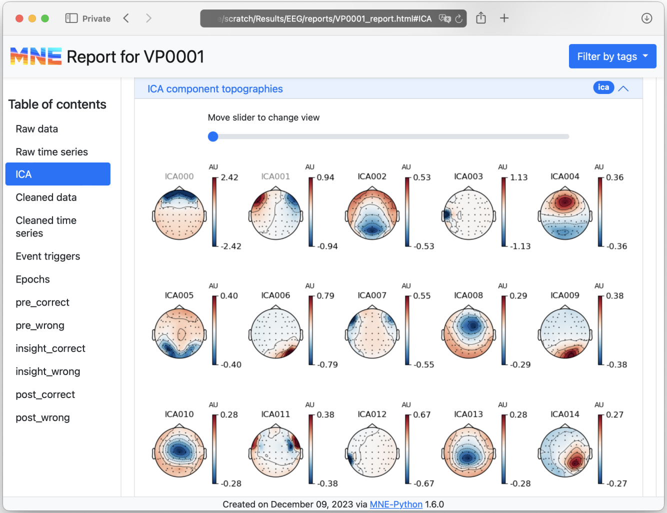
```r
# Number of rejected epochs per participant
lengths(config$auto_rejected_epochs)
```

```
## 05 07
##  7  0
```

More pipeline outputs

- Cleaned continuous data (`clean_dir`)
- Epoched data (`epochs_dir`)
- Automated QC reports (`reports_dir`)

QC reports


/scratch/Results/EEG/reports/VP0001_report.html#ICA
Private
Report for VP0001
Filter by tags
Table of contents
Raw data
Raw time series
ICA
Cleaned data
Cleaned time series
Event triggers
Epochs
pre_correct
pre_wrong
insight_correct
insight_wrong
post_correct
post_wrong
ICA component topographies
ica
Move slider to change view
ICA000 ICA001 ICA002 ICA003 ICA004
ICA005 ICA006 ICA007 ICA008 ICA009
ICA010 ICA011 ICA012 ICA013 ICA014
Created on December 09, 2023 via MNE-Python 1.6.0

Cluster-based permutation tests

```r
# Permutation test input
perm_contrasts = list(
  c("blurr_left", "normal_left"),
  c("blurr_right", "normal_right")
)
```

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

```
## # A tibble: 3,954 x 6
##    contrast                   time channel t_obs cluster p_val
##    <chr>                     <dbl> <chr>   <dbl> <chr>   <dbl>
##  1 blurr_left - normal_left 0     T7      735.  pos_204     1
##  2 blurr_left - normal_left 0     TP9      13.1 pos_204     1
##  3 blurr_left - normal_left 0.002 AF8     -17.3 neg_52      1
##  4 blurr_left - normal_left 0.002 FC3     381.  pos_26      1
##  5 blurr_left - normal_left 0.002 C3      115.  pos_26      1
##  6 blurr_left - normal_left 0.002 TP9      91.4 pos_204     1
##  7 blurr_left - normal_left 0.002 TP7      15.6 pos_204     1
##  8 blurr_left - normal_left 0.004 AF8     -20.3 neg_52      1
##  9 blurr_left - normal_left 0.004 F3       23.1 pos_27      1
## 10 blurr_left - normal_left 0.004 TP9      53.8 pos_204     1
## # i 3,944 more rows
```

Artifact correction

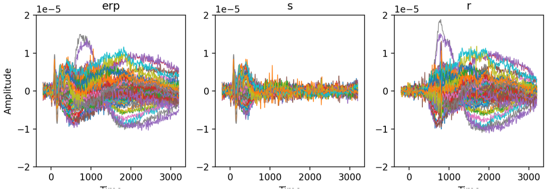
- **Multiple source eye correction (MSEC)**
  - Requires `.matrix` files from BESA

Artifact correction

- **Independent component analysis (ICA)**
  - Different algorithms available (e.g., `ica_method = "fastica"`)
  - Can specify initial number of principal components with `ica_n_components`
  - Automatic detection + exclusion of eye movement components based on correlation with HEOG and VEOG (see https://mne.tools/stable/generated/mne.preprocessing.ICA.html#mne.preprocessing.ICA.find_bads_eog)
  - Verify in QC reports
  - Other selection methods (manual selection, ICLabel) not yet implemented

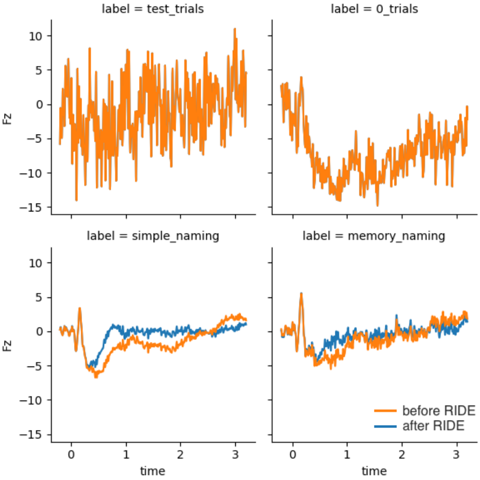
Artifact correction

- Coming soon: **Residue iteration decomposition (RIDE)**
  - For correcting speech artifacts
  - Based on iterative separation of stimulus- and response-related ERP components (Ouyang et al., 2011, 2015, 2016)
  - Subtract response-related component from single trials using their voice onset times

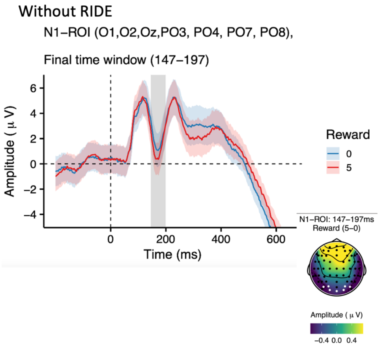


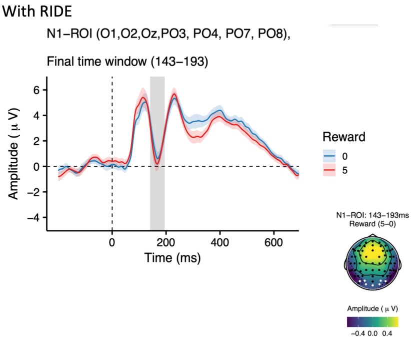
See https://github.com/kirstenstark/eeg-ride

Artifact correction



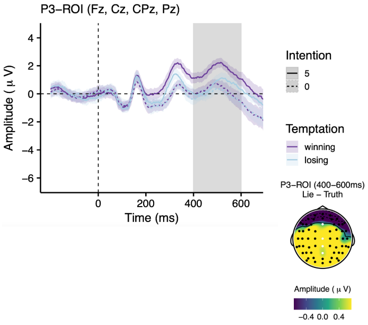
Artifact correction



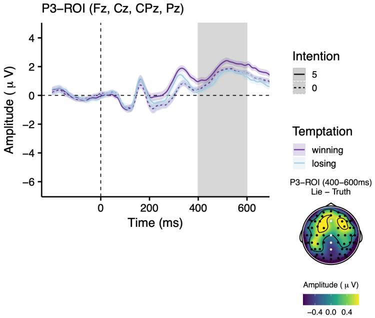


Artifact correction

Without RIDE

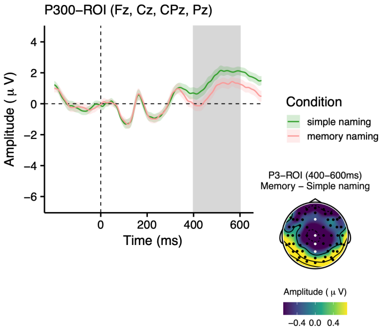


With RIDE

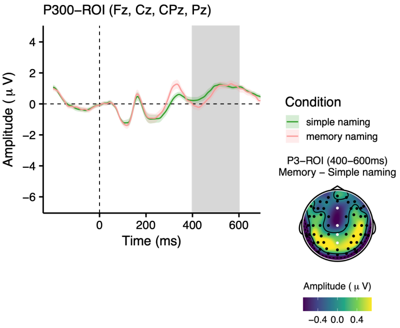


Artifact correction

Without RIDE

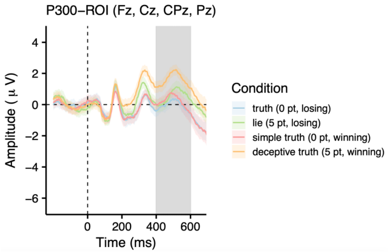


With RIDE

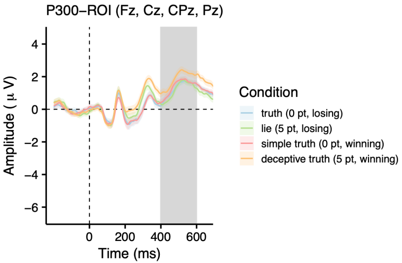


Artifact correction

Without RIDE



With RIDE

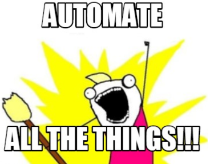


Artifact rejection

- Per-channel peak-to-peak amplitude threshold via `reject_peak_to_peak` (default: `200.0`)
- In addition to or instead of BESA or ICA

Repairing bad channels

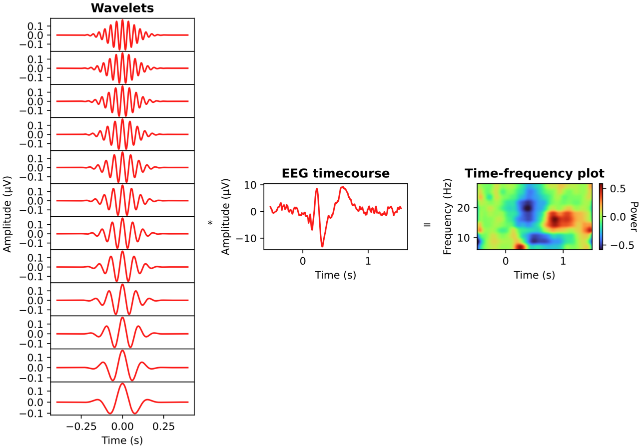
- Pass participant-specific vectors of bad channel labels
  - E.g., `bad_channels = list("05" = c("C3", "P7"), ...)`
- Uses spherical spline interpolation
- Experimental: Automatic bad channel detection (`bad_channels = "auto"`)
  - Based on channel *SD*s across epochs



Detecting missing epochs

- Requires log file column (`triggers_column`) with the EEG trigger for every trial
- Pipeline magically detects and deletes log file trials with missing EEG

Time-frequency analysis



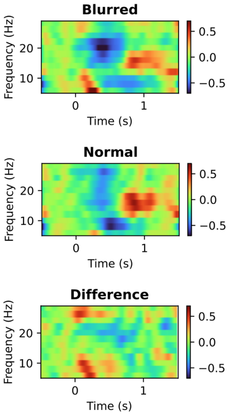
See https://github.com/alexenge/tfr-workshop

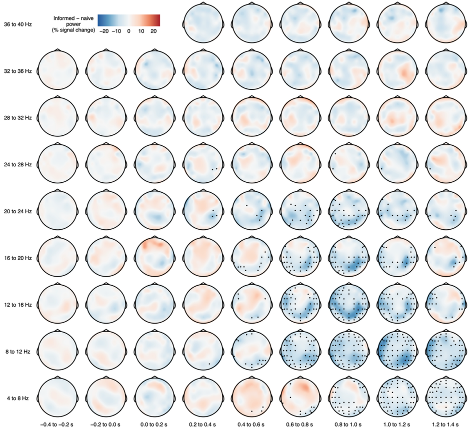
Time-frequency analysis

```
# Time-frequency analysis options
perform_tfr = TRUE,
tfr_components = list(
  "name" = list("alpha"),
  "tmin" = list(0.0), "tmax" = list(0.2),
  "fmin" = list(8.0), "fmax" = list(14.0),
  "roi" = list(c("PO9", "PO7", "PO3", "POz", "PO4", "PO8", "PO10", "O1", "Oz", "O2"))
)
```

- `tfr_components` extracts single trial power values
- Additional options:
  - Morlet frequencies (`tfr_freqs`, default 4, 5, 6, . . . , 40 Hz)
  - Morlet no. of cycles (`tfr_cycles`, default 2, 2.5, 3, . . . , 20)
  - Baseline window (`tfr_baseline`, default -450 ms to -50 ms)
  - Baseline method (`tfr_method`, default percent signal change)

Time-frequency analysis





See Enge et al. (2023)

Example datasets


```r
# Download example data from UCAP
pipeline$datasets$get_ucap(
  participants = 10,
  path = "data"
)

# Download example data from ERP CORE
pipeline$datasets$get_erpcore("N170")
pipeline$datasets$get_erpcore("MMN")
pipeline$datasets$get_erpcore("N2pc")
pipeline$datasets$get_erpcore("N400")
pipeline$datasets$get_erpcore("P3")
pipeline$datasets$get_erpcore("ERN")
```

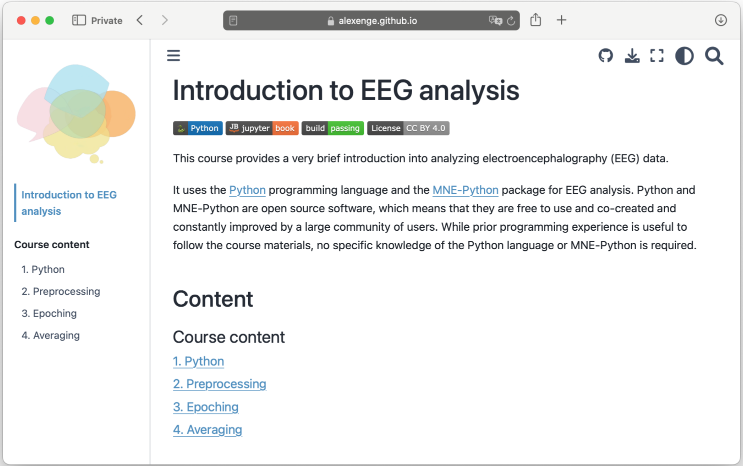
See Kappenman et al. (2021)

Plans

- Enhance documentation (examples, boilerplate, preprint)
- Unit tests
- Mixed models with `pymer4` or `bambi`
- Better permutation tests (Frossard & Renaud, 2021, 2022)
- BIDS interface
- Your ideas + contributions?

See https://github.com/alexenge/hu-neuro-pipeline/issues

Learning/teaching EEG analysis


Private
alexenge.github.io

Introduction to EEG analysis

Course content

1. Python
2. Preprocessing
3. Epoching
4. Averaging

Introduction to EEG analysis

Python | jupyter book | build passing | License CC BY 4.0

This course provides a very brief introduction into analyzing electroencephalography (EEG) data.

It uses the Python programming language and the MNE-Python package for EEG analysis. Python and MNE-Python are open source software, which means that they are free to use and co-created and constantly improved by a large community of users. While prior programming experience is useful to follow the course materials, no specific knowledge of the Python language or MNE-Python is required.

Content

Course content

1. Python
2. Preprocessing
3. Epoching
4. Averaging

See https://alexenge.github.io/intro-to-eeg

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

Enge, A., Süß, F., & Rahman, R. A. (2023). Instant effects of semantic information on visual perception. *Journal of Neuroscience*, *43*(26), 4896–4906. https://doi.org/10.1523/JNEUROSCI.2038-22.2023

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

Kappenman, E. S., Farrens, J. L., Zhang, W., Stewart, A. X., & Luck, S. J. (2021). ERP CORE: An open resource for human event-related potential research. *NeuroImage*, *225*, 117465. https://doi.org/10.1016/j.neuroimage.2020.117465

Ouyang, G., Herzmann, G., Zhou, C., & Sommer, W. (2011). Residue iteration decomposition (RIDE): A new method to separate ERP components on the basis of latency variability in single trials. *Psychophysiology*, *48*(12), 1631–1647. https://doi.org/10.1111/j.1469-8986.2011.01269.x

Ouyang, G., Sommer, W., & Zhou, C. (2015). A toolbox for residue iteration decomposition (RIDE)—A method for the decomposition, reconstruction, and single trial analysis of event related potentials. *Journal of Neuroscience Methods*, *250*, 7–21. https://doi.org/10.1016/j.jneumeth.2014.10.009

Ouyang, G., Sommer, W., Zhou, C., Aristei, S., Pinkpank, T., & Abdel Rahman, R. (2016). Articulation artifacts during overt language production in event-related brain potentials: Description and correction. *Brain Topography*, *29*(6), 791–813. https://doi.org/10.1007/s10548-016-0515-1

Volpert-Esmond, H. I., Page-Gould, E., & Bartholow, B. D. (2021). Using multilevel models for the analysis of event-related potentials. *International Journal of Psychophysiology*, *162*, 145–156. https://doi.org/10.1016/j.ijpsycho.2021.02.006