hu-neuro-pipeline

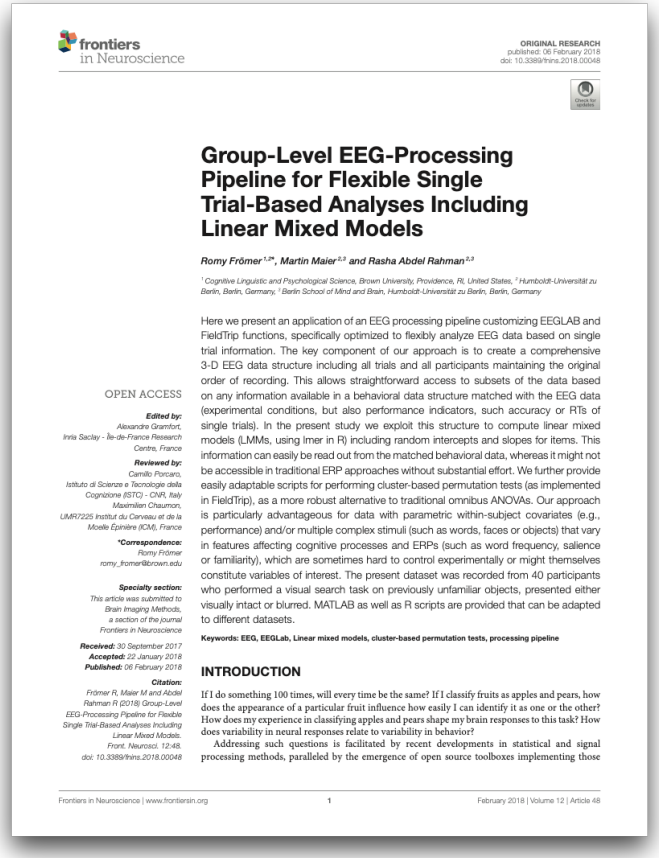
A Python implementation of the single trial EEG pipeline by Frömer et al. (*Front. Neurosci.*, 2018)

Alexander Enge

Neuro Lab @ Humboldt-Universität zu Berlin

2023-01-11

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:
Alexandra 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_froemer@brown.edu

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

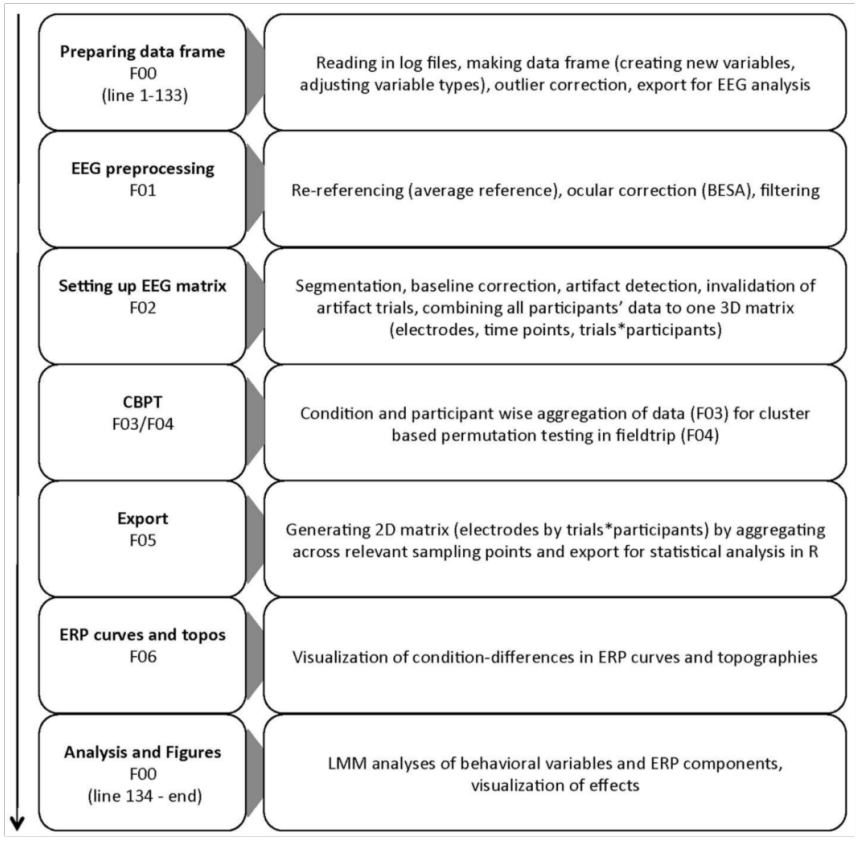
Received: *30 September 2017*
Accepted: *22 January 2018*
Published: *06 February 2018*

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

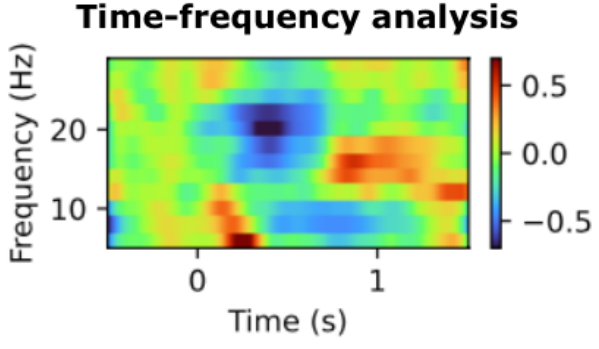
- Allows single trial analysis of ERP amplitudes
  - By-item random effects (Bürki et al., 2018)
  - Trial and item level covariates (Volpert-Esmond et al., 2021)
  - Continuous predictor variables
  - Unbalanced designs

Python implementation

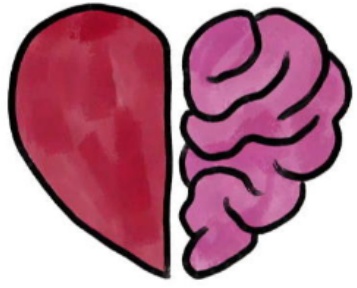
+

MNE

+



=



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

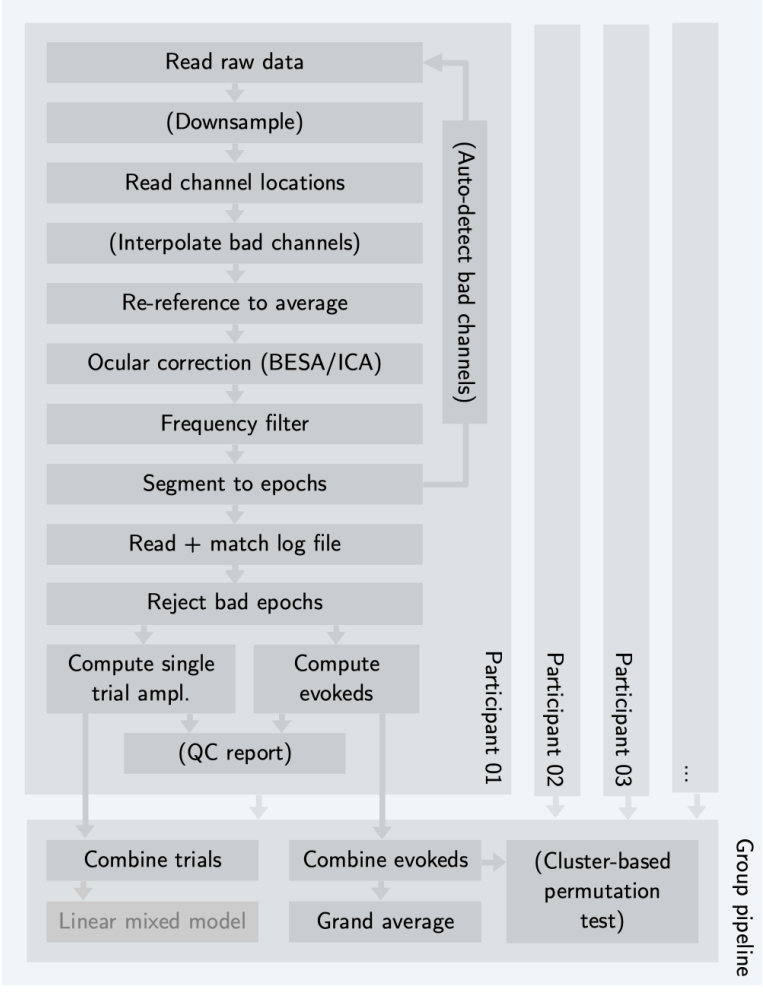
- Versatile
  - EEG, MEG, ECoG, fNIRS
  - Preprocessing, statistics, time-frequency analysis, visualization, machine learning, connectivity, source localization, ...
- Open source
  - 329 contributors on GitHub as of January 2023
  - Funding: NIH, NSF, ERC, Google, Amazon, ...
  - Code review, automated tests, user forum, office hours, ...

MNE
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
- Code standards + version control
  (https://github.com/alexenge/hu-neuro-pipeline/)

Python implementation



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 Python (Miniconda distribution)
reticulate::install_miniconda()

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

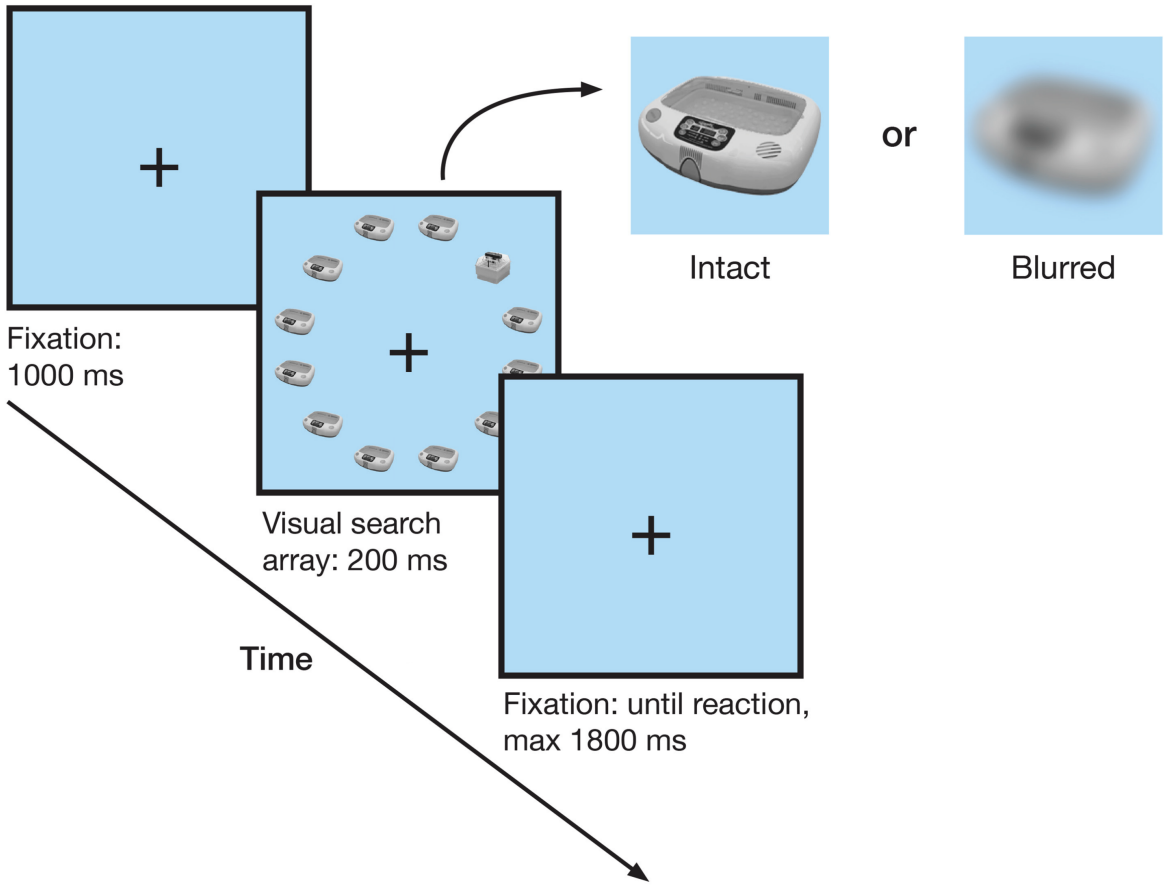
General usage


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

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

Minimal example


```r
# 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", "PO3", "POz", "PO4")
    )
  ),
  # Averaging options
  average_by = c("n_b", "DeviantPosRL", "n_b/DeviantPosRL")
)
```

Minimal example



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",
```

- Directory path or list of BESA files (`.matrix`)
- 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`)
- 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

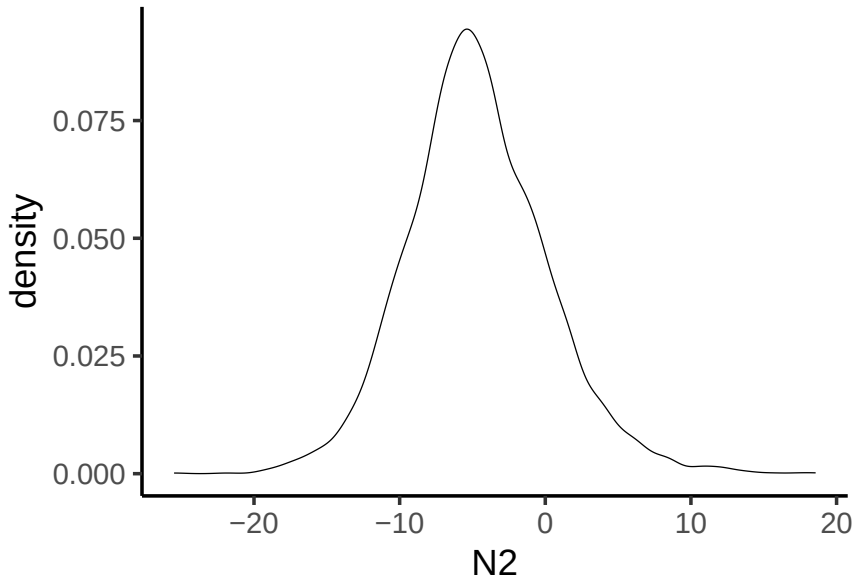
```
# Single trial data frame
print(trials)
```

```
## # A tibble: 3,840 x 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
```

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()‘).
```



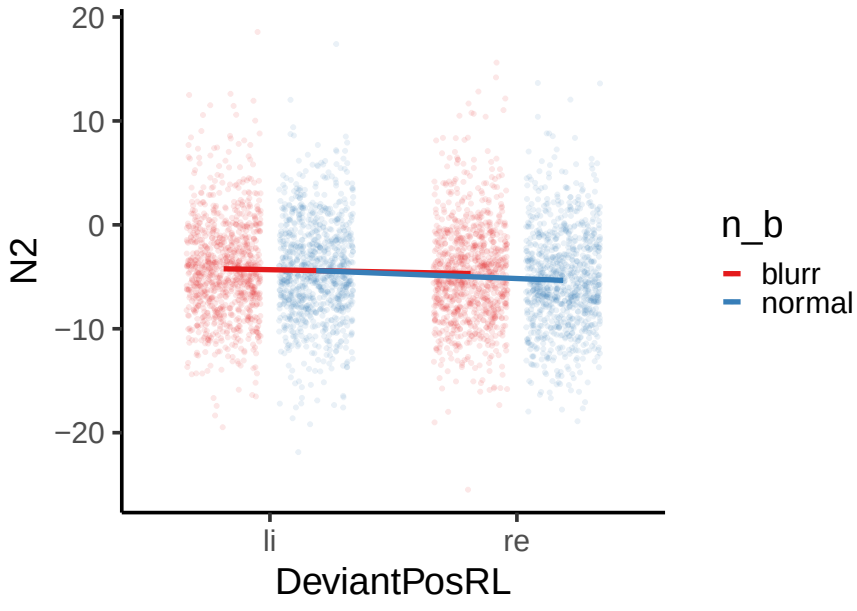
Pipeline outputs


```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

```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

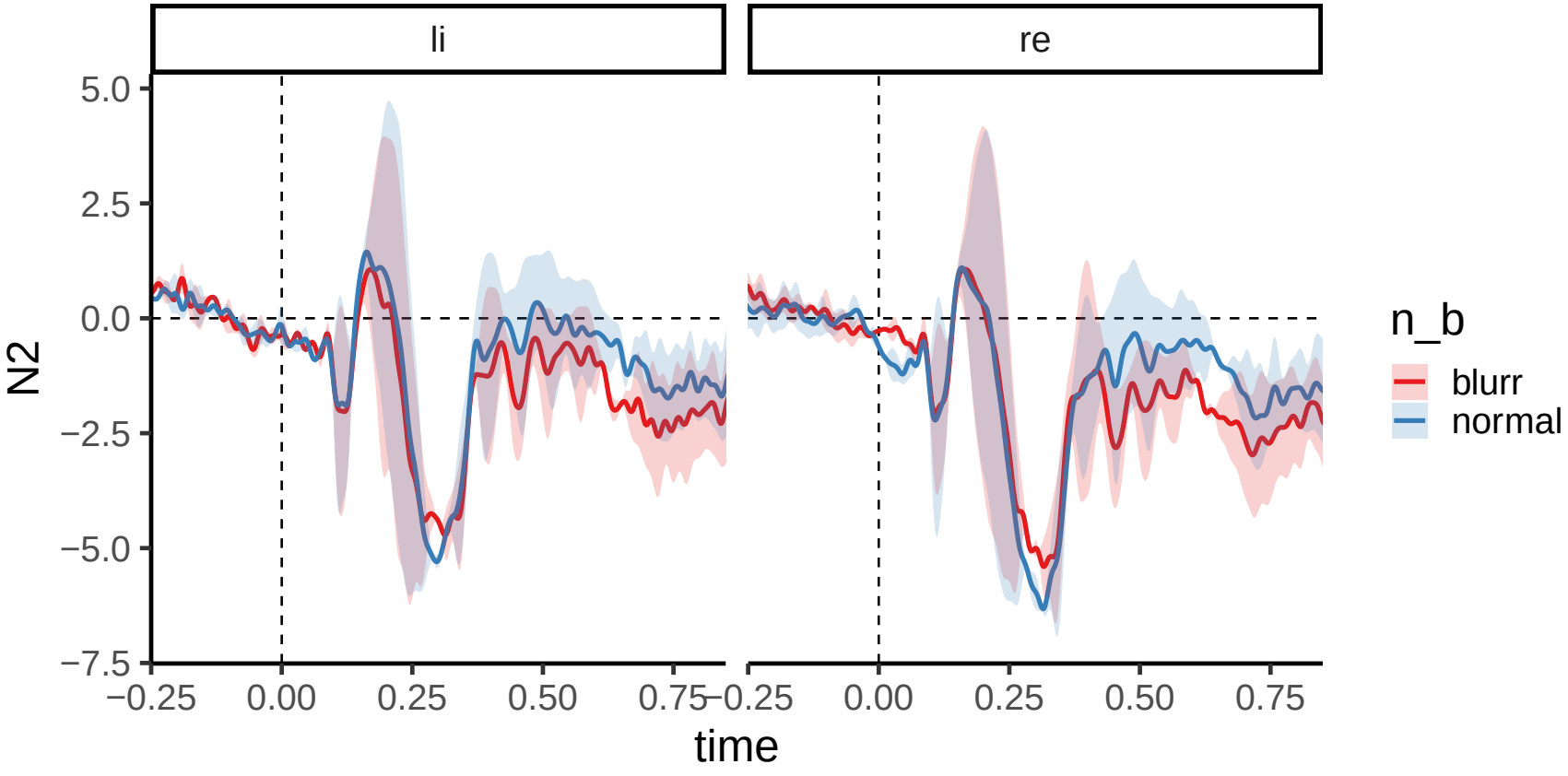

```
# Evokeds by participant and condition
print(evokeds)
```

```
## # A tibble: 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>, ...
```

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] "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
```

More pipeline outputs

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

Cluster-based permutation tests

```
# Permutation test input
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
```

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 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

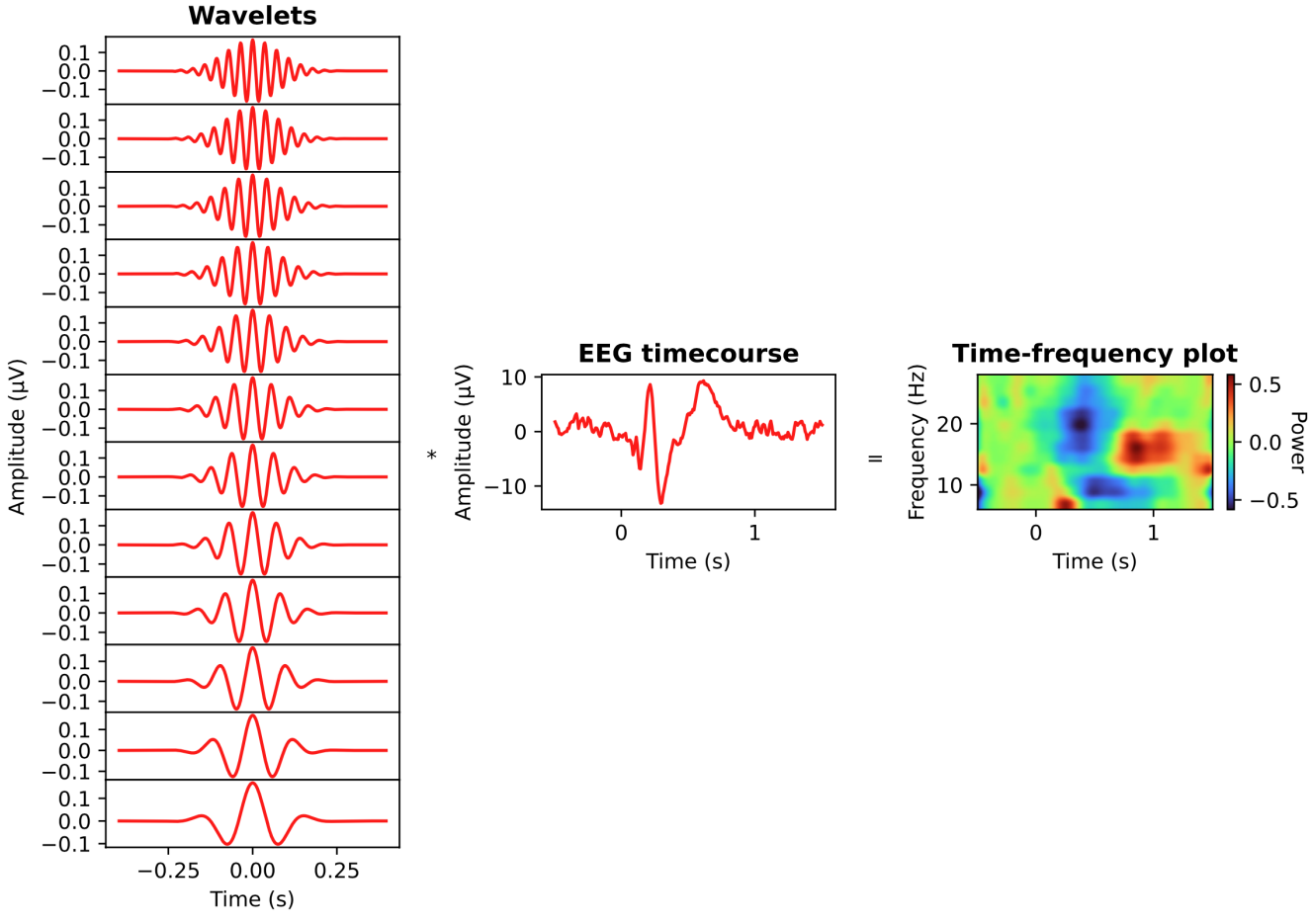
Repairing bad channels

- 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

Detect missing epochs

- Requires log file column with the EEG trigger for every trial
- Specify name of this column as `triggers_column = ...`
- Pipeline magically detects and deletes log file trials with missing EEG

Time-frequency analysis



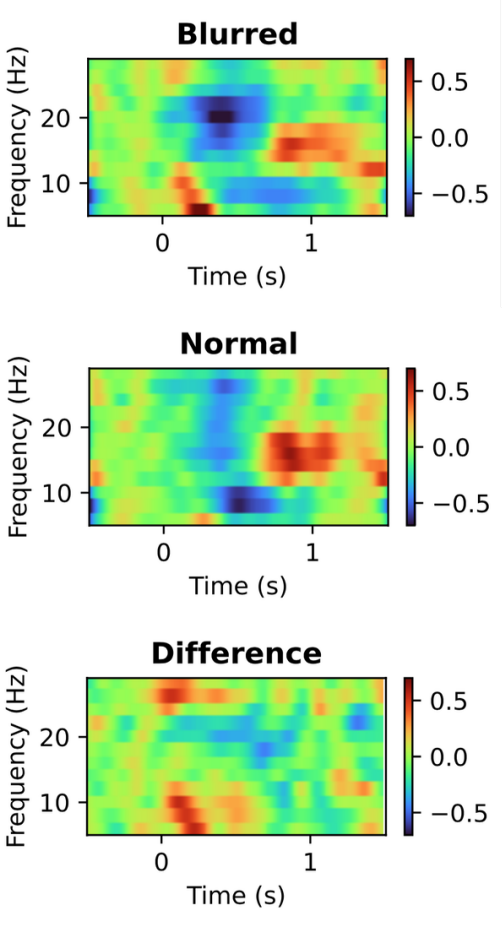
See https://github.com/alexenge/tfr-workshop

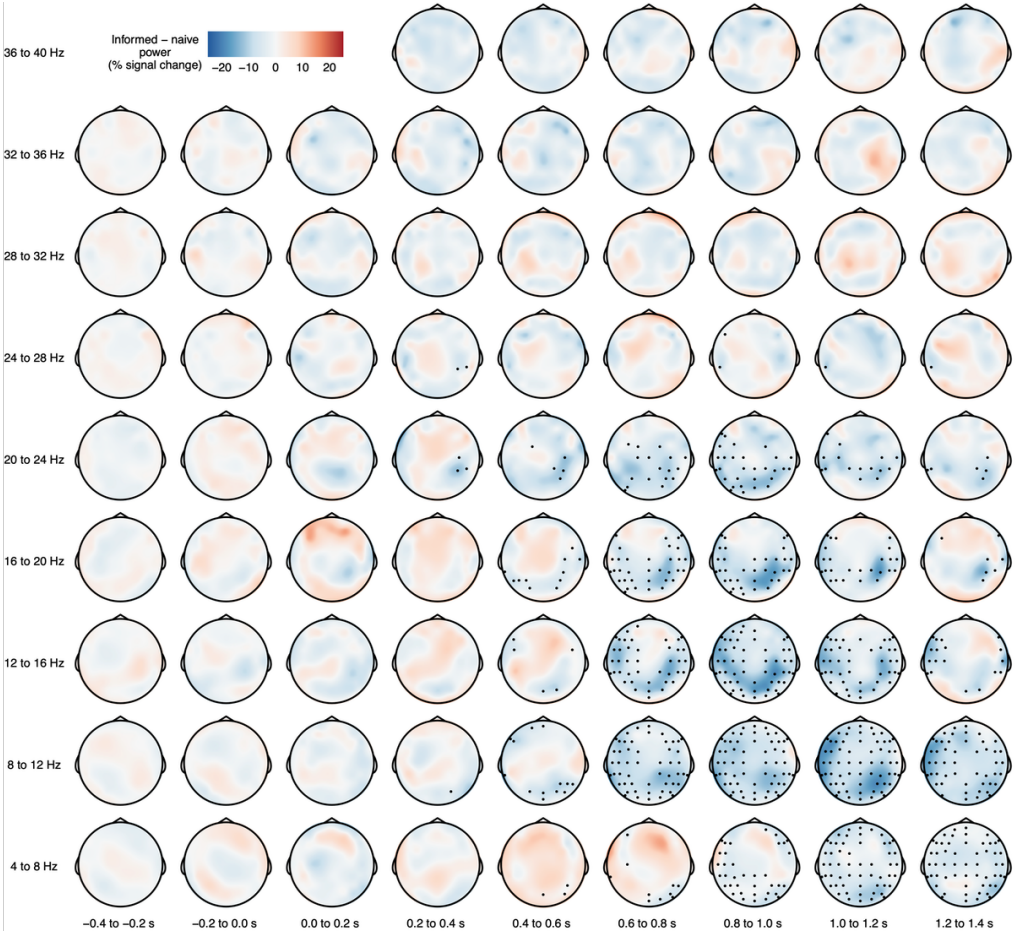
Time-frequency analysis

```
# 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
- Can additionally specify:
  - Morlet frequencies (`tfr_freqs`, default 4, 5, 6, . . . , 40 Hz)
  - Numbers 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





Plans

- Improve documentation
- Unit tests
- RIDE correction for speech artifacts
- 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

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