

Modelling M/EEG data with Bayesian nonparametric multilevel models

Ladislav Nalborczyk¹ and Paul Bürkner²

¹Aix Marseille Univ, CNRS, LPL

²TU Dortmund University, Department of Statistics

Abstract

Time-resolved electrophysiological measurements such as those offered by magneto- or electro-encephalography (M/EEG) provide a unique window onto neural activity underlying cognitive process and how they unfold over time. Typically, we are interested in testing whether such measures differ across conditions and/or groups. The conventional approach consists in conducting mass-univariate statistics followed by some form of multiplicity correction (e.g., FDR, FWER) or cluster-based inference. However, these cluster-based methods have an important downside: they shift the focus of inference from the timepoint to the cluster level, thus preventing any conclusion to be made about the onset and offset of effects (e.g., differences across conditions). Here, we introduce a novel *model-based approach* for analysing one-dimensional M/EEG timeseries such as ERPs or decoding timecourses and their differences across conditions. This approach relies on Bayesian nonparametric multilevel modelling (multilevel generalised additive models or Gaussian processes), which outputs posterior probabilities of the effect being above chance at every timestep/voxel, while naturally taking into account the temporal dependencies and between-subject variability present in such data.

Keywords: EEG, MEG, generalised additive models, gaussian processes, mixed-effects models, Bayesian statistics, brms

Table of contents

Introduction	3
Introduction (in progress)	3
Previous modelling work	3
Generalised additive models	3
Gaussian process regression	3
Objectives	4

Ladislav Nalborczyk  <https://orcid.org/0000-0002-7419-9855>

Paul Bürkner  <https://orcid.org/0000-0001-5765-8995>

The authors have no conflicts of interest to disclose.

Correspondence concerning this article should be addressed to Ladislav Nalborczyk, Aix Marseille Univ, CNRS, LPL, 5 avenue Pasteur, 13100 Aix-en-Provence, France, email: ladislav.nalborczyk@cnrs.fr

15	Methods	4
16	M/EEG data simulation	4
17	Model fitting	6
18	Posterior probability of difference above 0	7
19	Error properties of the proposed approach	9
20	Using summary statistics of ERPs	10
21	Application to 1D decoding results (accuracy over time)	12
22	Application to 2D decoding results (cross-temporal generalisation)	18
23	Application to actual M/EEG data	19
24	Comparing the identified onsets/offsets to other approaches	19
25	Results	22
26	Discussion	23
27	Summary of the proposed approach	23
28	Increasing potential usage	23
29	Limitations and future directions	23
30	Conclusions	23
31	Packages	24
32	References	25
33	Mathematical formulation of the bivariate GAM	28
34	Threshold-free cluster enhancement	29
35	Using the R package and integration with MNE-Python	30

Modelling M/EEG data with Bayesian nonparametric multilevel models

Introduction (in progress)

Here are some useful references to be discussed (Combrisson & Jerbi, 2015; Ehinger & Dimigen, 2019; Frossard & Renaud, 2021a, 2022; Gramfort, 2013; Hayasaka, 2003; Luck & Gaspelin, 2017; Maris & Oostenveld, 2007; E. J. Pedersen et al., 2019; Pernet et al., 2015; Riutort-Mayol et al., 2023; Rousselet, 2025)... See also (Maris, 2011)... and (Rosenblatt et al., 2018) (history of cluster-based approaches and using a data split?)... Cluster failure (Eklund et al., 2016)...

In the following, we consider two approaches to modelling the non-linear timecourse of M/EEG ERPs or decoding performance: i) generalised additive models (GAMs) with thin-plate smoothing splines (Wood, 2003; Wood, 2004) and ii) Gaussian processes (GPs) with a smooth covariance kernel (Rasmussen & Williams, 2005) and low-rank approximation (Riutort-Mayol et al., 2023).

Previous modelling work

Disentangling overlapping processes (Skukies et al., 2024; Skukies & Ehinger, 2021)... Using Bayes factors (Teichmann, 2022)...

Generalised additive models

In generalised additive models (GAMs), the functional relationship between predictors and response variable is decomposed into a sum of low-dimensional non-parametric functions. A typical GAM has the following form:

$$y_i \sim \text{EF}(\mu_i, \phi)$$

$$g(\mu_i) = A_i + \mathbf{X}_i \gamma + \sum_{j=1}^J f_j(x_{ij})$$

where $y_i \sim \text{EF}(\mu_i, \phi)$ denotes that the observations y_i are distributed as some member of the exponential family of distributions (e.g., Gaussian, Gamma, Beta, Poisson) with mean μ_i and scale parameter ϕ ; $g(\cdot)$ is the link function, A is an offset, \mathbf{X}_i is the i th row of a parametric model matrix, γ is a vector of parameters for the parametric terms, f_j is a smooth function of covariate x_j . The smooth functions f_j are represented in the model via penalised splines basis expansions of the covariates, that are a weighted sum of basis functions:

$$f_j(x_{ij}) = \sum_{k=1}^K \beta_{jk} b_{jk}(x_{ij})$$

where β_{jk} is the weight (coefficient) associated with the k th basis function $b_{jk}()$ evaluated at the covariate value x_{ij} for the j th smooth function f_j . Splines' coefficients are penalised...

Gaussian process regression

A Gaussian process (GP) is a stochastic process that defines the distribution over a collection of random variables indexed by a continuous variable, that is $\{f(t) : t \in \mathcal{T}\}$ for some index set \mathcal{T} (Rasmussen & Williams, 2005; Riutort-Mayol et al., 2023). Whereas Bayesian linear regression outputs a distribution over the parameters of some predefined parametric model, the GP approach, in contrast, is a non-parametric approach, in that it finds a distribution over the possible functions that are consistent with the observed data. However, note that nonparametric does not mean there aren't parameters, it means that there are infinitely many parameters.

From [brms documentation](#): A GP is a stochastic process, which describes the relation between one or more predictors $x = (x_1, \dots, x_d)$ and a response $f(x)$, where d is the number of predictors. A GP is the generalization of the multivariate normal distribution to an infinite number of dimensions. Thus, it can be interpreted as a prior over functions. The values of $f()$ at any finite set of locations are jointly multivariate normal, with a covariance matrix defined by the covariance kernel $k_p(x_i, x_j)$, where p is the vector of parameters of the GP:

$$(f(x_1), \dots, f(x_n)) \sim \text{MVN}\left(0, (k_p(x_i, x_j))_{i,j=1}^n\right)$$

The smoothness and general behaviour of the function f depends only on the choice of covariance kernel, which ensures that values that are close together in the input space will be mapped to similar output values...

From this perspective, f is a realisation of an infinite dimensional normal distribution:

$$f \sim \text{Normal}(0, C(\lambda))$$

where C is a covariance kernel with hyperparameters λ that defines the covariance between two function values $f(t_1)$ and $f(t_2)$ for two time points t_1 and t_2 ([Rasmussen & Williams, 2005](#)). Similar to the different choices of the basis function for splines, different choices of the covariance kernel lead to different GPs. In this article, we consider the squared-exponential (a.k.a. radial basis function) kernel, which computes the squared distance between points and converts it into a measure of similarity. It is defined as:

$$C(\lambda) := C(t_1, t_2, \sigma, \gamma) := \sigma^2 \exp\left(-\frac{\|t_1 - t_2\|^2}{2\gamma^2}\right)$$

with hyperparameters $\lambda = (\sigma, \gamma)$, expressing the overall scale of GP and the length-scale, respectively ([Rasmussen & Williams, 2005](#)). The advantages of this kernel are that it is computationally efficient and (infinitely) smooth making it a reasonable choice for the purposes of the present article. Here again, λ hyperparameters are estimated from the data, along with all other model parameters.

Taken from <https://michael-franke.github.io/Bayesian-Regression/practice-sheets/10c-Gaussian-processes.html>: For a given vector \mathbf{x} , we can use the kernel to construct finite multi-variate normal distribution associated with it like so:

$$\mathbf{x} \mapsto_{GP} \text{MVNormal}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}))$$

where m is a function that specifies the mean for the distribution associated with \mathbf{x} . This mapping is essentially the Gaussian process: a systematic association of vectors of arbitrary length with a suitable multi-variate normal distribution.

Low-rank approximate Gaussian processes are of main interest in machine learning and statistics due to the high computational demands of exact Gaussian process models ([Riutort-Mayol et al., 2023](#))...

Objectives

...

Methods

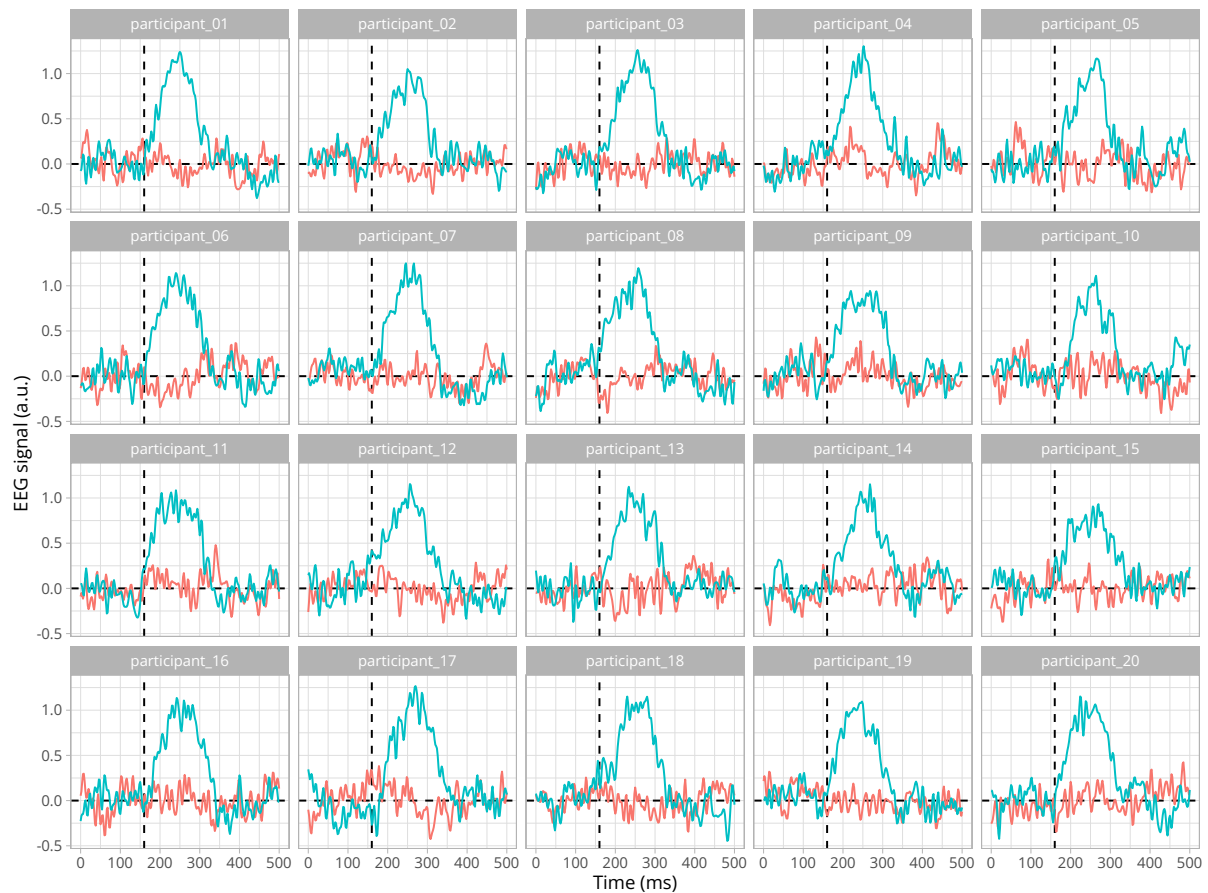
M/EEG data simulation

Following the approach used by Sassenhagen & Draschkow (2019) and Rousselet (2025), we simulated EEG data stemming from two conditions, one with noise only, and the other with noise + signal. As in previous studies, the noise was generated by superimposing 50 sinusoids

at different frequencies, following an EEG-like spectrum (see details and code in [Yeung et al., 2004](#)). As in [Rousselle \(2025\)](#), the signal was generated from truncated Gaussian with an objective onset at 160 ms, a peak at 250 ms, and an offset at 342 ms. We simulated this signal for 250 timesteps between 0 and 0.5s, akin to 500 Hz sampling rate. We simulated such data for a group of 20 participants with 50 trials per participant and condition.

Figure 1

Some ERPs in two conditions with 50 trials each, for a group of 20 participants.



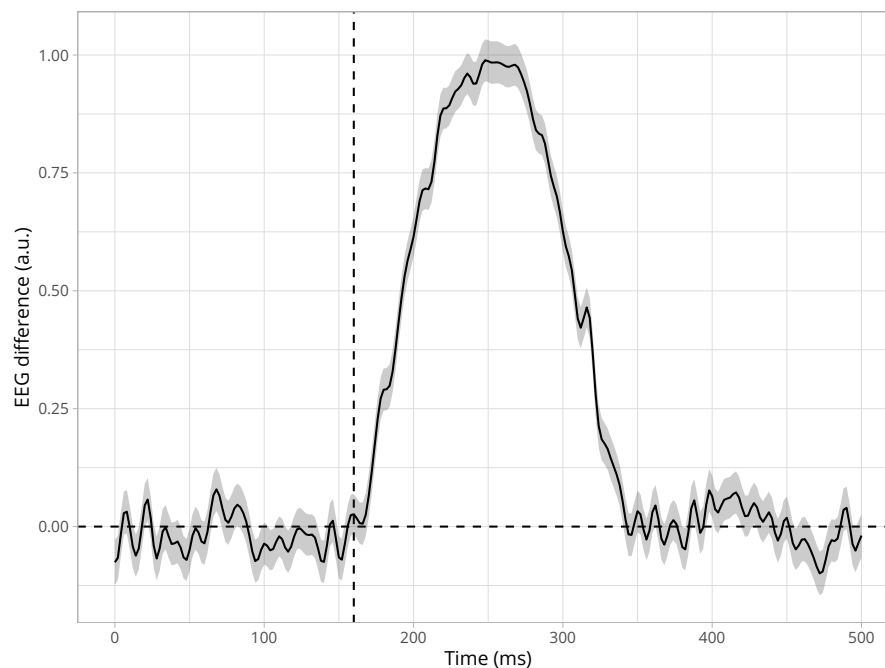
```
# averaging across participants
group_df <- raw_df %>%
  pivot_wider(names_from = condition, values_from = y, values_fn = mean) %>%
  mutate(y_diff = cond2 - cond1) %>%
  summarise(
    y_mean = mean(y_diff),
    y_sem = sd(y_diff) / sqrt(n() ),
    .by = x
  ) %>%
  mutate(lower = y_mean - y_sem, upper = y_mean + y_sem)

# plotting the data
group_df %>%
  ggplot(aes(x = x, y = y_mean) ) +
  geom_hline(yintercept = 0, linetype = 2) +
  geom_vline(xintercept = true_onset, linetype = 2) +
```

```
geom_ribbon(
  aes(ymin = lower, ymax = upper, colour = NULL),
  alpha = 0.25, show.legend = FALSE
) +
geom_line(show.legend = FALSE) +
labs(x = "Time (ms)", y = "EEG difference (a.u.)")
```

Figure 2

Group-level average difference between conditions (mean \pm standard error of the mean). The ‘true’ (population value of the) onset is indicated by the vertical dashed line.



78 Model fitting

79 Models were fitted using the `brms` package (Bürkner, 2017, 2018; Nalborczyk et al.,
80 2019)...

```
# defining a contrast for condition
contrasts(group_df$condition) <- c(-0.5, 0.5)

# fitting the GAM
gam <- brm(
  # cubic regression splines with k-1 basis functions
  # then we should try s(x, participant, bs = "fs")
  # that is, random factor smooth interactions...
  y ~ condition + s(x, bs = "cr", k = 20, by = condition),
  data = group_df,
  family = gaussian(),
  iter = 5000,
  chains = 4,
  cores = 4,
  file = "models/gam.rds"
)
```

81

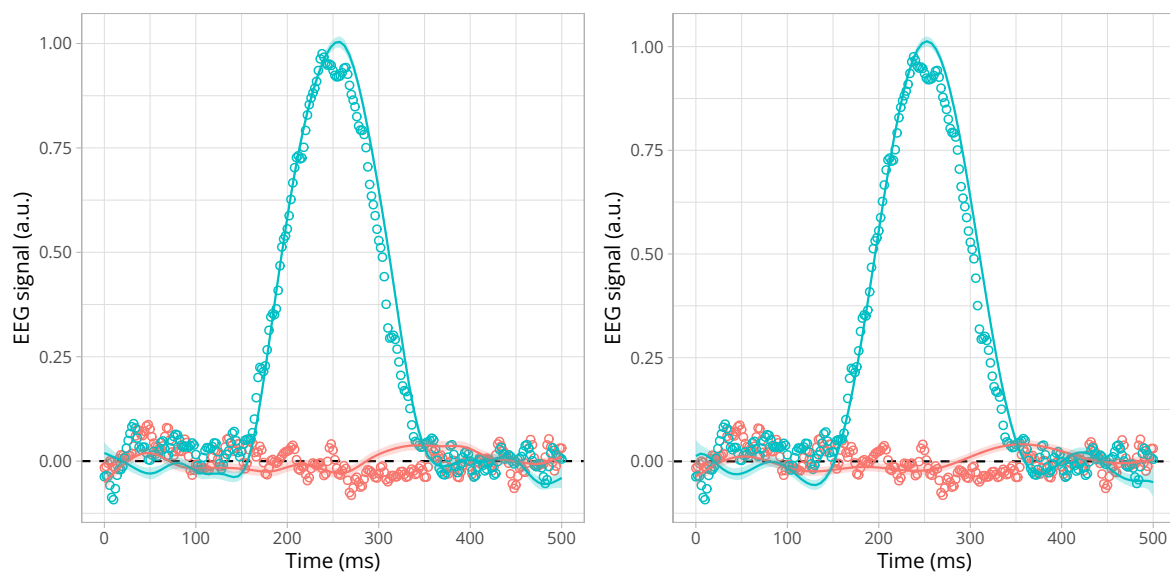
Now we fit the GP...

```
gp_model <- brm(
  # k refers to the number of basis functions for
  # computing Hilbert-space approximate GPs
  y ~ gp(x, k = 20, by = condition),
  # if k = NA (default), exact GPs are computed
  # y ~ gp(x, by = condition),
  data = group_df,
  family = gaussian(),
  control = list(adapt_delta = 0.99),
  iter = 5000,
  chains = 4,
  cores = 4,
  file = "models/gp.rds"
)
```

82

And we plot the posterior predictions...

```
# plotting the posterior predictions
plot_post_preds(model = gam, data = group_df) +
  plot_post_preds(model = gp_model, data = group_df)
```

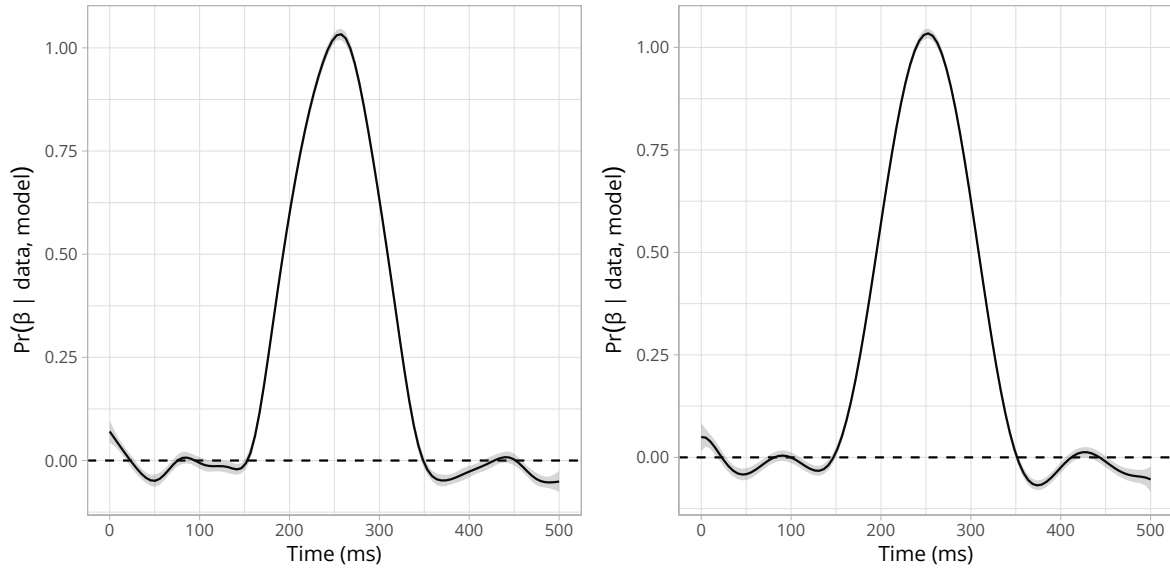
Figure 3*Posterior predictions of the GAM (left) and GP (right) models.*83 **Posterior probability of difference above 0**

84

We can retrieve the posterior probability of the slope for `condition` at each timestep.

Figure 4

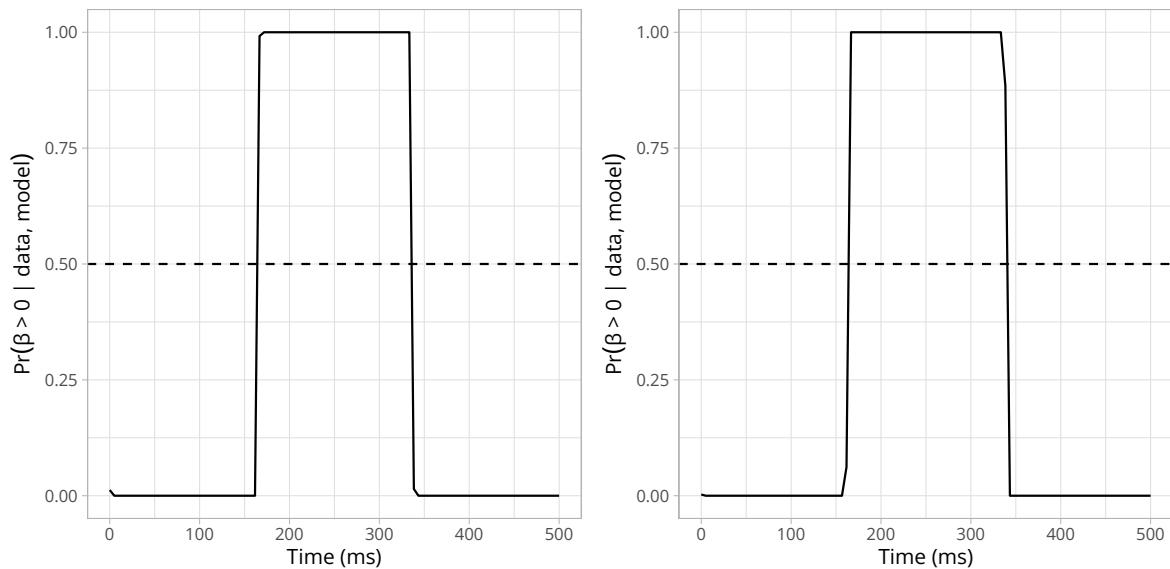
Slope for the difference between conditions according to the GAM (left) or the GP (right).



85 We can also compute the posterior probability of the slope for `condition` being above
 86 0 (Figure 5).

Figure 5

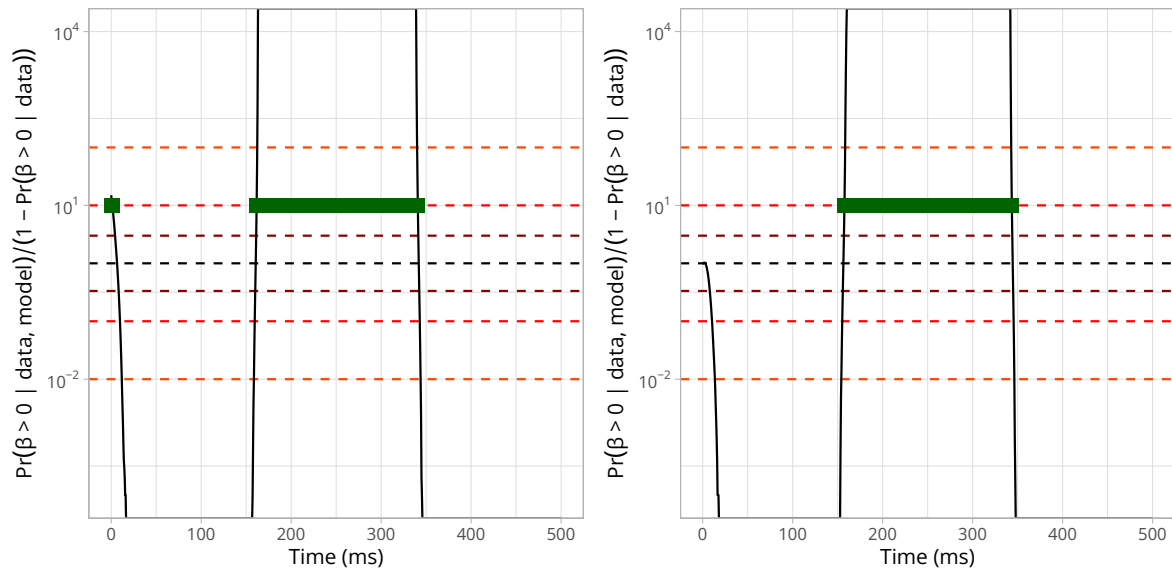
Posterior probability of the EEG signal being above $0+eps$ according to the GAM (left) and the GP (right).



87 We can also express this as the ratio of posterior probabilities (i.e., $p/(1-p)$) and visualise
 88 the timecourse of this ratio superimposed with the conventional BF thresholds (Figure 6).

Figure 6

Ratio of posterior probability according to the GAM (left) and the GP (right). Timesteps above threshold (>10) are highlighted in green.



Error properties of the proposed approach

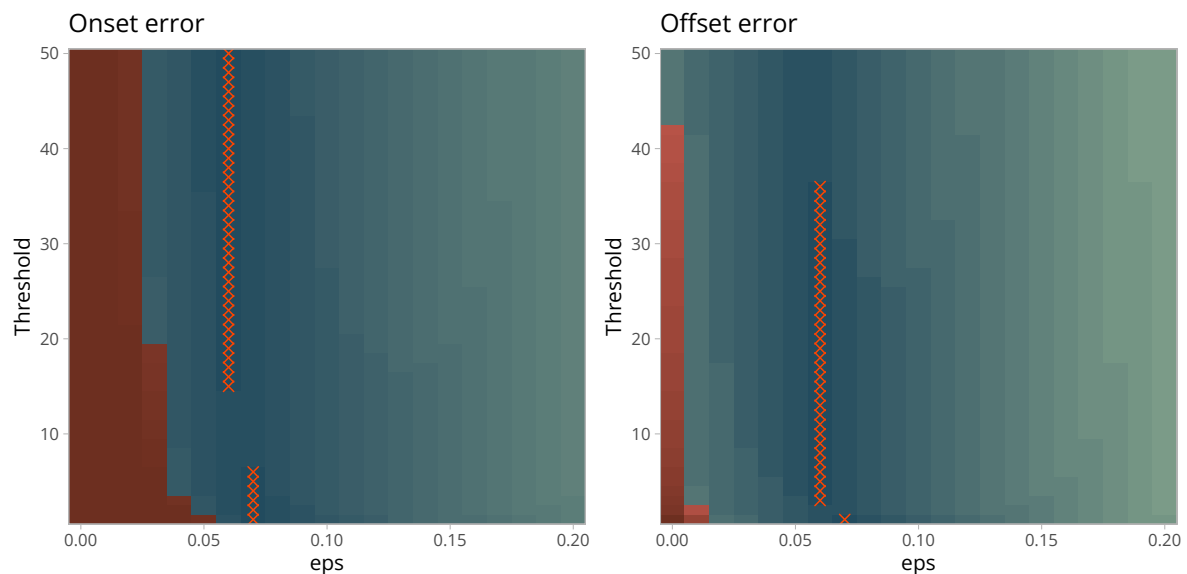
Below we are plotting the error function (where the error is computed as $|\hat{\theta} - \theta|$) for both the onset and offset values of the ERP difference, according to various **eps** and **threshold** values. Remember that the signal is generated from a truncated Gaussian defining an objective onset at 160 ms, a maximum at 250 ms, and an offset at 342 ms. Figure 7 shows that the GP model can *exactly* recover the true onset and offset values, given some reasonable choice of **eps** and **threshold** values.

	eps	threshold	estimated_onset	estimated_offset	error_onset	error_offset
1	0.06	15	160	342	0	0
2	0.06	16	160	342	0	0
3	0.06	17	160	342	0	0
4	0.06	18	160	342	0	0
5	0.06	19	160	342	0	0
6	0.06	20	160	342	0	0

	eps	threshold	estimated_onset	estimated_offset	error_onset	error_offset
1	0.06	3	159	342	1	0
2	0.06	4	159	342	1	0
3	0.06	5	159	342	1	0
4	0.06	6	159	342	1	0
5	0.06	7	159	342	1	0
6	0.06	8	159	342	1	0

Figure 7

Error function of onset (left) and offset (right) estimation according to various *eps* and threshold values (according to the GP model). Minimum error values are indicated by red crosses.



Using summary statistics of ERPs

Next we fit a hierarchical GAM using summary statistics of ERPs (mean and SD) at the participant level (similar to what is done in meta-analysis).

```
# averaging across participants
summary_df <- raw_df %>%
  summarise(
    eeg = mean(y),
    eeg_sd = sd(y),
    .by = c(participant, condition, x)
  )

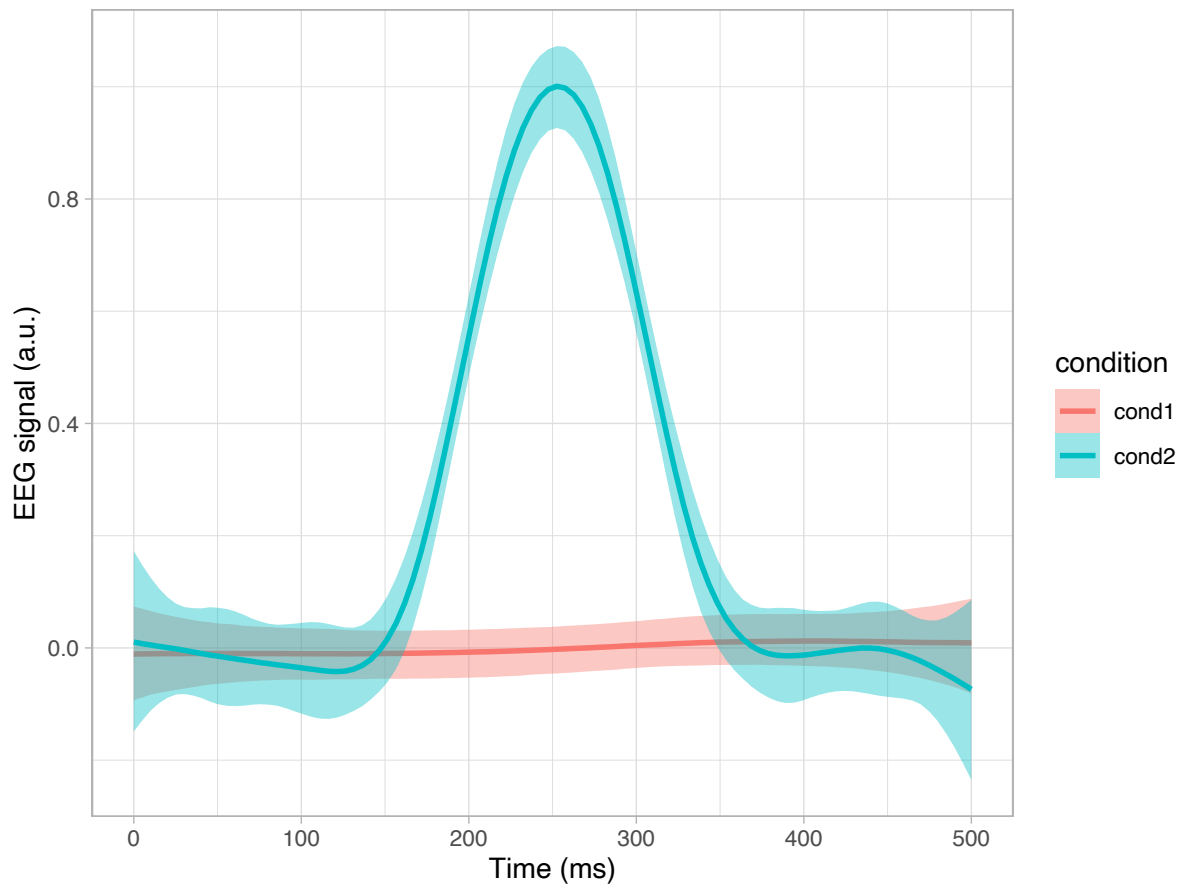
# defining a contrast for condition
contrasts(summary_df$condition) <- c(-0.5, 0.5)

# fitting the GAM
meta_gam <- brm(
  # using by-participant SD of ERPs across trials
  eeg | se(eeg_sd) ~
    condition + s(x, bs = "cr", k = 10, by = condition) +
    (1 | participant),
  data = summary_df,
  family = gaussian(),
  iter = 5000,
  chains = 4,
  cores = 4,
  file = "models/meta_gam.rds"
)
```

```
# plotting the posterior predictions
plot(
  conditional_effects(x = meta_gam, effect = "x:condition"),
  points = FALSE, theme = theme_light(), plot = FALSE
)[[1]] +
  labs(x = "Time (ms)", y = "EEG signal (a.u.)")
```

Figure 8

Hierarchical GAM posterior predictions.



```
# fitting the GP
meta_gp <- brm(
  # using by-participant SD of ERPs across trials
  eeg | se(eeg_sd) ~
    condition + gp(x, k = 20, by = condition) +
    (1 | participant),
  data = summary_df,
  family = gaussian(),
  control = list(adapt_delta = 0.95, max_treedepth = 20),
  iter = 5000,
  chains = 4,
  cores = 4,
  file = "models/meta_gp.rds"
)
```

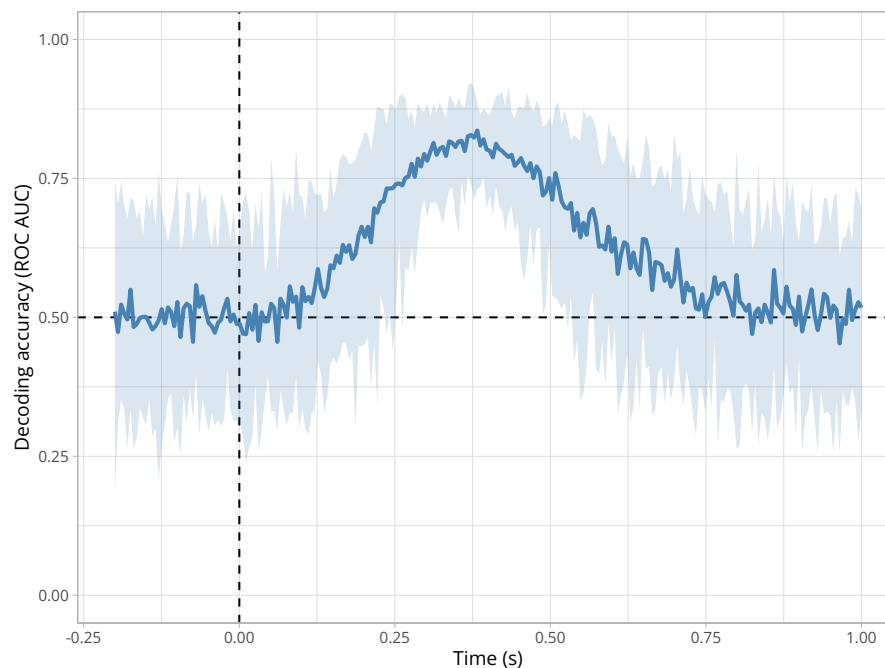
```
# plotting the posterior predictions
plot(
  conditional_effects(x = meta_gp, effect = "x:condition"),
  points = FALSE, theme = theme_light(), plot = FALSE
)[[1]] +
  labs(x = "Time (ms)", y = "EEG signal (a.u.)")
```

113 Application to 1D decoding results (accuracy over time)

114 Assume we have M/EEG data and we conducted time-resolved multivariate pattern
 115 analysis (MVPA), also known as decoding. As a result, we have a timecourse of decoding
 116 accuracies (e.g., ROC AUC), bounded between 0 and 1, per participant (Figure 9).

Figure 9

Exemplary (simulated) group-level average timecourse of binary decoding accuracy (ROC AUC).



```
# loading the reticulate package
library(reticulate)

# importing the numpy and mne python modules
np <- import("numpy")
mne <- import("mne")
sklearn <- import("sklearn")

# defining the function in R (it will be executed in Python)
mne_decoding <- function (X, labels, ncores = 1) {

  # converting R dataframe to NumPy array (reshaping if needed)
  # X should be a matrix before conversion
  X_np <- np$array(X)
```

```

if (length(dim(X_np) ) == 2) {

  # adding a second dimension (channels) if missing
  X_np <- np$expand_dims(X_np, axis = as.integer(1) )

}

# defining the classifier
clf <- sklearn$linear_model$LogisticRegression(solver = "liblinear")

# sliding the estimator on all time frames
time_decod <- mne$decoding$SlidingEstimator(
  clf, n_jobs = as.integer(ncores),
  scoring = "roc_auc", verbose = TRUE
)

# or using N-fold cross-validation
scores <- mne$decoding$cross_val_multiscore(
  time_decod,
  X_np,
  labels,
  cv = as.integer(4),
  n_jobs = as.integer(ncores),
  verbose = TRUE
)

# returning the scores (averaged over CV folds)
return (scores)

}

# listing all participants
participants <- unique(raw_df$participant)

# initialising empty decoding results
group_decoding_scores <- data.frame()

# running decoding for each participant
for (ppt in participants) {

  # printing progress
  print(ppt)

  # retrieve data from one participant
  ppt_data <- raw_df %>%
    filter(participant == ppt) %>%
    select(-participant) %>%
    pivot_wider(names_from = x, values_from = y) %>%
    select(-condition, -trial)

  # extracting the labels

```

```

labels <- raw_df %>%
  filter(participant == ppt) %>%
  select(-participant) %>%
  pivot_wider(names_from = x, values_from = y) %>%
  pull(condition) %>%
  as.numeric()

# extracting the timesteps
timesteps <- raw_df %>%
  filter(participant == ppt) %>%
  select(-participant) %>%
  pull(x) %>%
  unique()

# running the decoding
decoding_scores <- data.frame(
  mne_decoding(X = ppt_data, labels = labels-1)
) %>%
# computing the average over CV folds
summarise(across(where(is.numeric), mean) )

# appending to previous results
group_decoding_scores <- bind_rows(group_decoding_scores, decoding_scores)
}

# saving the scores
saveRDS(object = group_decoding_scores, file = "results/decoding_scores.rds")

# importing the decoding scores
group_decoding_scores <- readRDS(file = "results/decoding_scores.rds")

# extracting the timesteps
timesteps <- raw_df %>%
  # filter(participant == ppt) %>%
  # select(-participant) %>%
  pull(x) %>%
  unique()

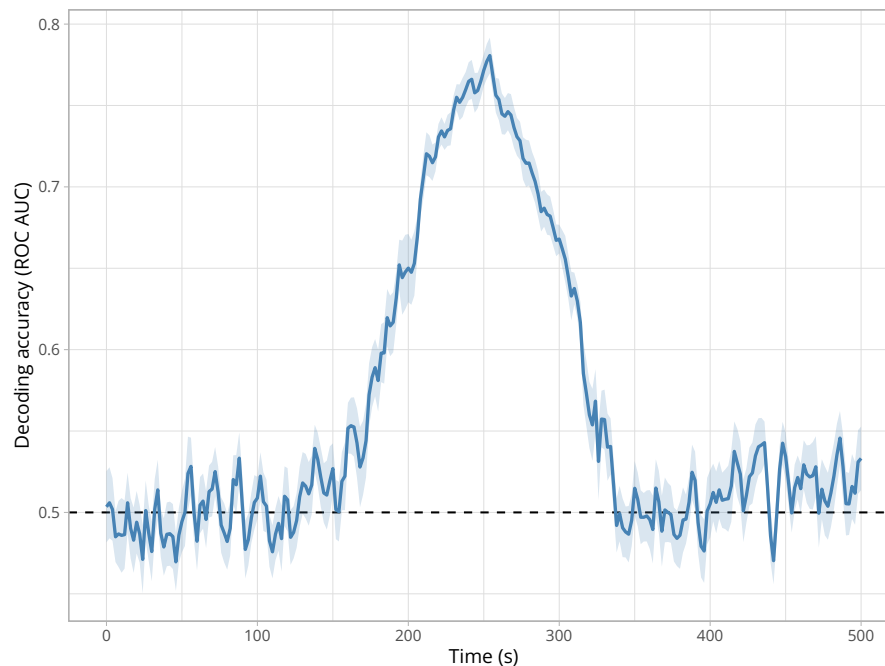
# plotting it
group_decoding_scores %>%
  t() %>%
  data.frame() %>%
  mutate(
    mean = rowMeans(across(1:20) ),
    se = apply(across(1:20), 1, function (x) sd(x) / sqrt(length(x) ) )
  ) %>%
  mutate(lower = mean - se, upper = mean + se) %>%
  mutate(time = timesteps) %>%
  ggplot(aes(x = time, y = mean) ) +
  geom_hline(yintercept = 0.5, linetype = "dashed") +

```

```
geom_ribbon(aes(ymin = lower, ymax = upper), fill = "steelblue", alpha = 0.2) +
geom_line(color = "steelblue", size = 0.8) +
labs(x = "Time (s)", y = "Decoding accuracy (ROC AUC)")
```

Figure 10

Exemplary group-level average timecourse of binary decoding accuracy (ROC AUC). Decoding was performed using MNE-Python.



117 Now, we want to *test* whether the group-level average decoding accuracy is above chance
 118 (i.e., 0.5) at each timestep. We use a similar GAM/GP as previously, but we replace the Normal
 119 likelihood function by a Beta one to account for the bounded nature of AUC values (between 0
 120 and 1).

```
# fitting the GAM
decoding_gam <- brm(
  auc ~ s(time, bs = "cr", k = 10),
  data = decoding_data,
  family = Beta(),
  iter = 5000,
  chains = 4,
  cores = 4,
  file = "models/decoding_gam.rds"
)
```

```
# fitting the GP
decoding_gp <- brm(
  auc ~ gp(time, k = 20),
  data = decoding_data,
  family = Beta(),
  control = list(adapt_delta = 0.9),
```

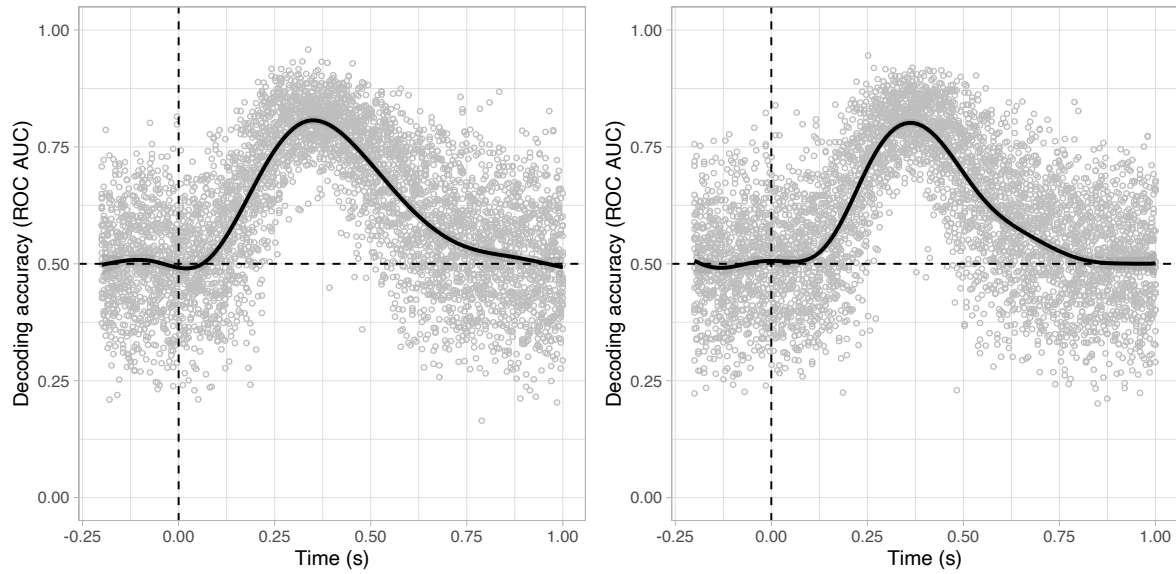
```

iter = 5000,
chains = 4,
cores = 4,
file = "models/decoding_gp.rds"
)

```

Figure 11

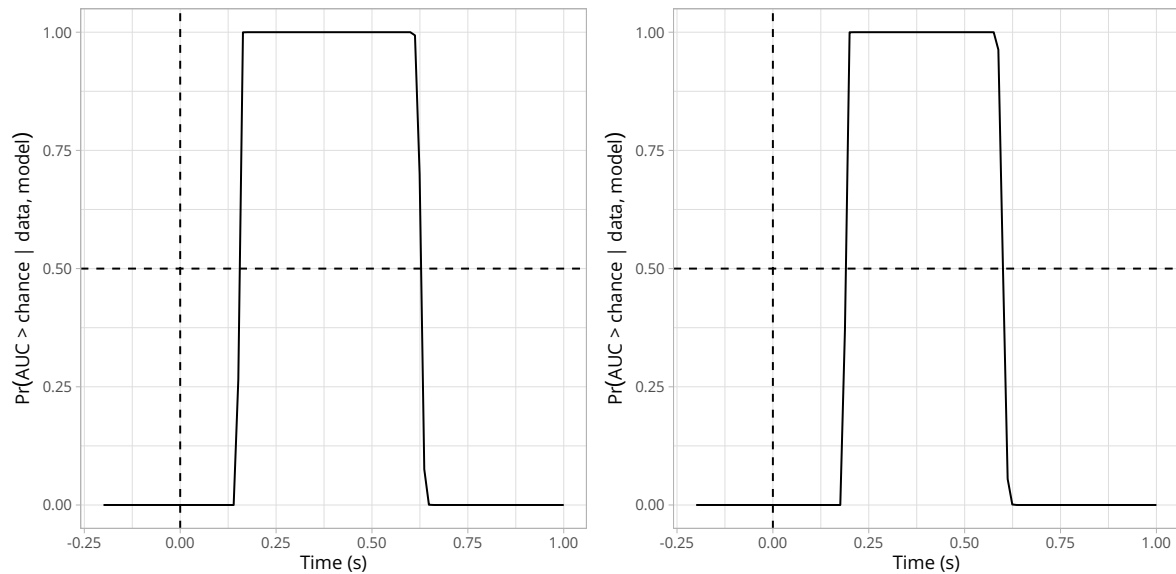
Posterior predictions of the GAM (left) and GP (right) fitted on decoding accuracy over time.



121 Next, we plot the posterior probability of decoding accuracy being above chance level
 122 (plus some epsilon) (Figure 12).

Figure 12

Posterior probability of decoding accuracy being above chance level according to the GAM (left) or the GP (right).



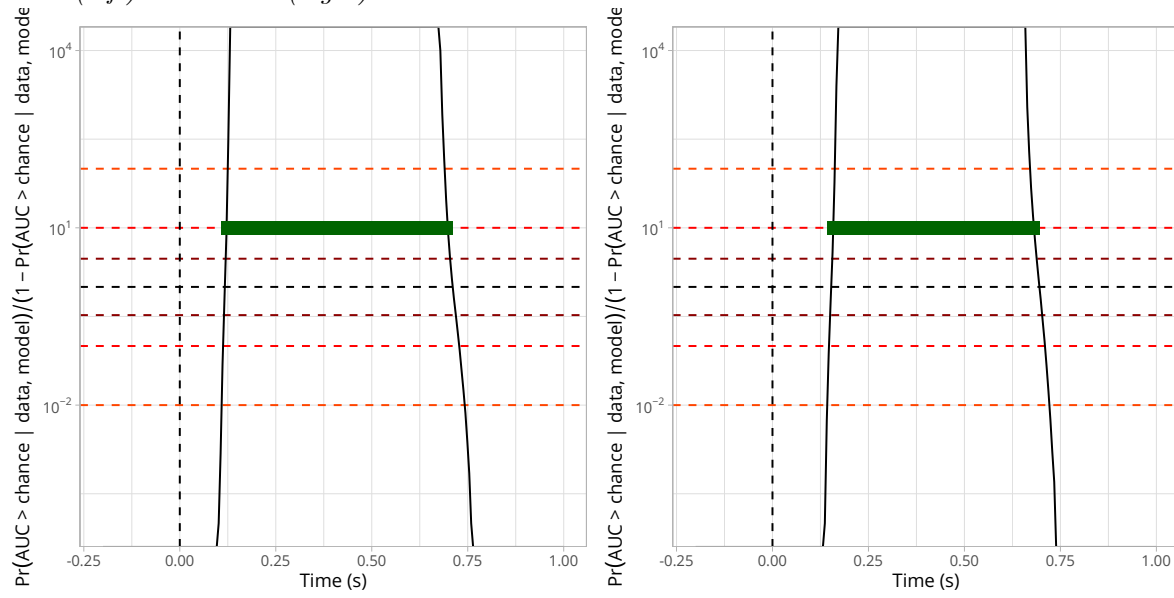

```

123   cluster_onset cluster_offset
124   1      0.1263598      0.6937238
125   cluster_onset cluster_offset
126   1      0.1615063      0.6786611

```

Figure 13

Ratio of posterior probabilities of decoding accuracy being above chance level according to the GAM (left) or the GP (right).

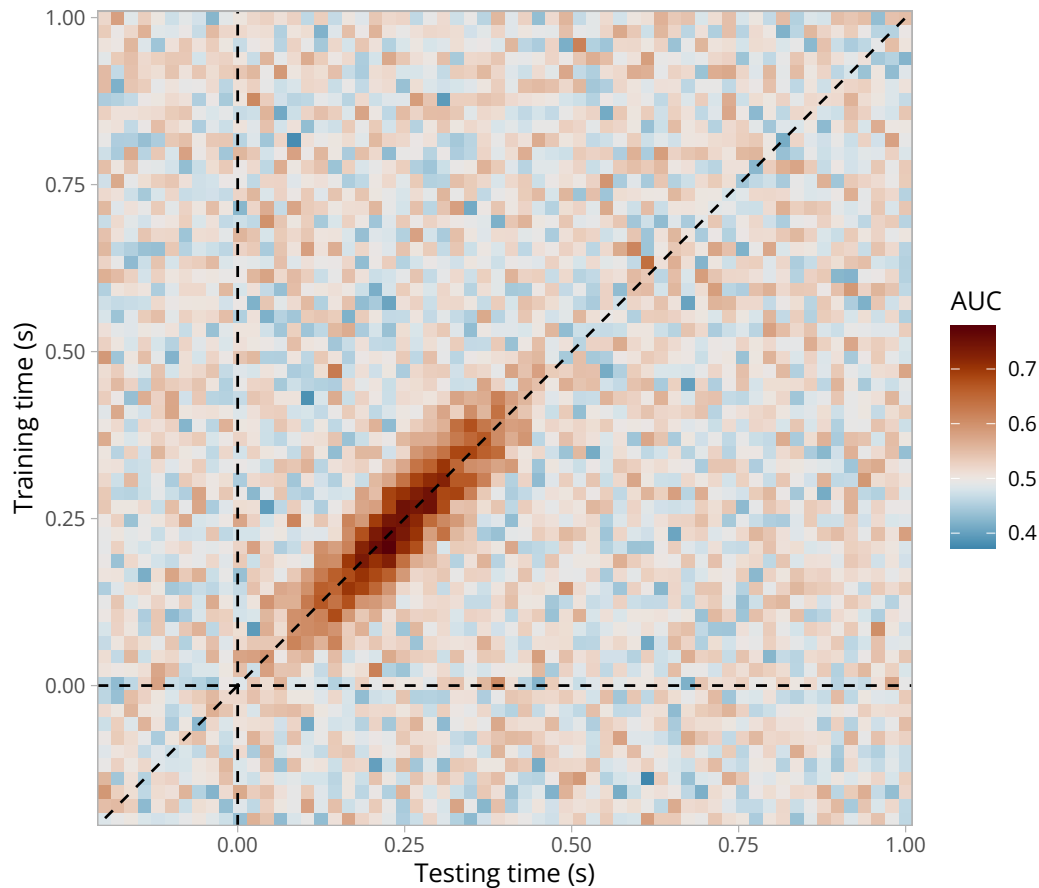


Application to 2D decoding results (cross-temporal generalisation)

Assume we have M/EEG data and we have conducted cross-temporal generalisation analyses (King & Dehaene, 2014). As a result, we have a 2D matrix where each element contains the decoding accuracy (e.g., ROC AUC) of a classifier trained at timestep t_{training_i} and tested at timestep t_{testing_j} (Figure 14).

Figure 14

Exemplary (simulated) group-level average cross-temporal generalisation matrix of decoding performance (ROC AUC).



Now, we want to test whether and when decoding performance is above chance level (0.5 for a binary decoding task). These two models are computationally heavier to fit (more observations and 2D smooth functions)...

```
# fitting a GAM with two temporal dimensions
timegen_gam <- brm(
  # 2D thin-plate spline (tp) to model smooth interactions between training and testing t
  # auc ~ s(train_time, test_time, bs = "tp", k = 10),
  auc ~ t2(train_time, test_time, bs = "tp", k = 10),
  data = timegen_data,
  family = Beta(),
  iter = 2000,
  chains = 4,
  cores = 4,
  file = "models/timegen_gam_t2.rds"
```

```

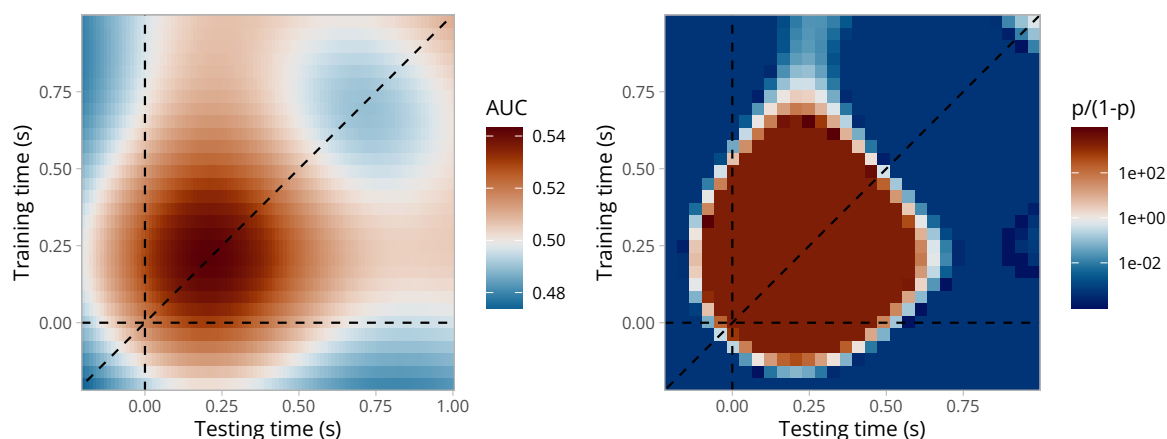
)

# fitting a GP with two temporal dimensions
timegen_gp <- brm(
  auc ~ gp(train_time, test_time, k = 20),
  # data = timegen_data %>% mutate(auc_c = scale(x = auc, center = TRUE, scale = FALSE) )
  data = timegen_data,
  family = Beta(),
  control = list(adapt_delta = 0.95, max_treedepth = 20),
  iter = 2000,
  chains = 4,
  cores = 4,
  file = "models/timegen_gp.rds"
)

```

Figure 15

Posterior probability of decoding accuracy being above chance level (2D GAM).



Application to actual M/EEG data

Assessing the reliability of the proposed approach (in comparison to other methods) using some sort of split-half reliability (Rosenblatt et al., 2018)?

Comparing the identified onsets/offsets to other approaches

From Rousselet (2024): Five methods for multiple comparison correction were considered: two FDR methods; two cluster-based methods; and the maximum statistics. The two FDR methods were BH95 (Benjamini & Hochberg, 1995) and BY01 (Benjamini & Yekutieli, 2001), which were applied to the permutation p-values using the `p.adjust` function in R. The first cluster-based inference was implemented using a cluster-sum statistic of squared t-values...

Comparing performance of the GAM and GP models with:

- 145 • uncorrected univariate tests
- 146 • univariate tests + Bonferroni
- 147 • univariate tests + BH95 (Benjamini-Hochberg, 1995) aka FDR
- 148 • univariate tests + BY01 (Benjamini-Yekutieli, 2001)
- 149 • cluster-based permutation test (cluster sum)
- 150 • cluster-based permutation test (cluster depth)
- 151 • threshold-free cluster enhancement
- 152 • change point

153 We used the R package `permuco` v 1.1.3 (Frossard & Renaud, 2021b) and the R package
 154 `changepoint` v 2.2.4 (Killick et al., 2022a)...

Figure 16

Timecourse of squared t -values and s -values (continuous measure of evidence given by $-\log_2(p\text{-value})$) with true onset and onset identified using the raw (uncorrected) p -values or the corrected p -values (BH, BY, Bonferroni, or Holm).

True onset: 160ms, Uncorrected onset: 38ms, BH onset: 38ms, BY onset: 166ms,
 Bonf onset: 168ms, Holm onset: 166ms

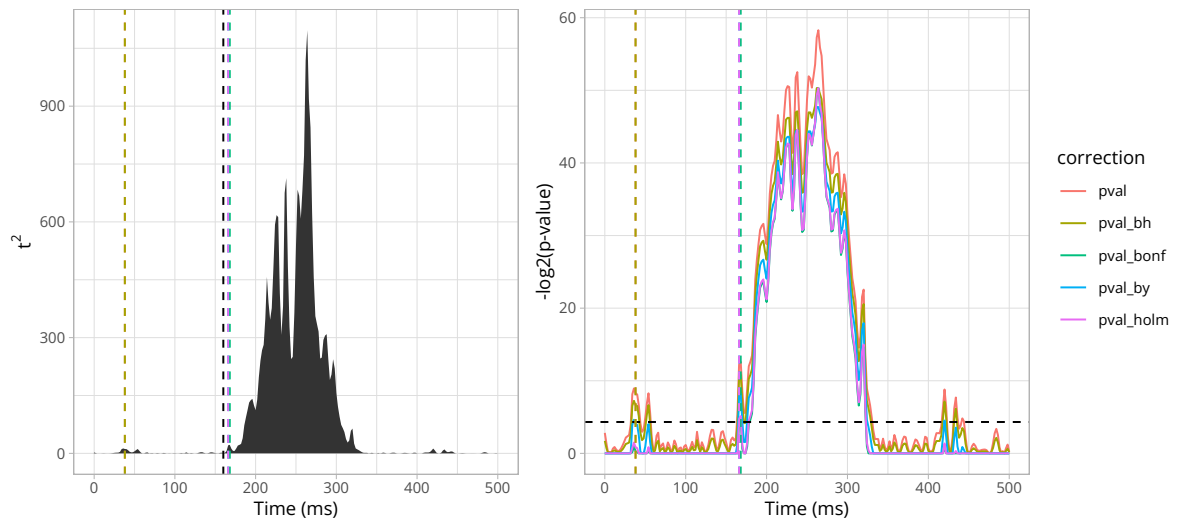
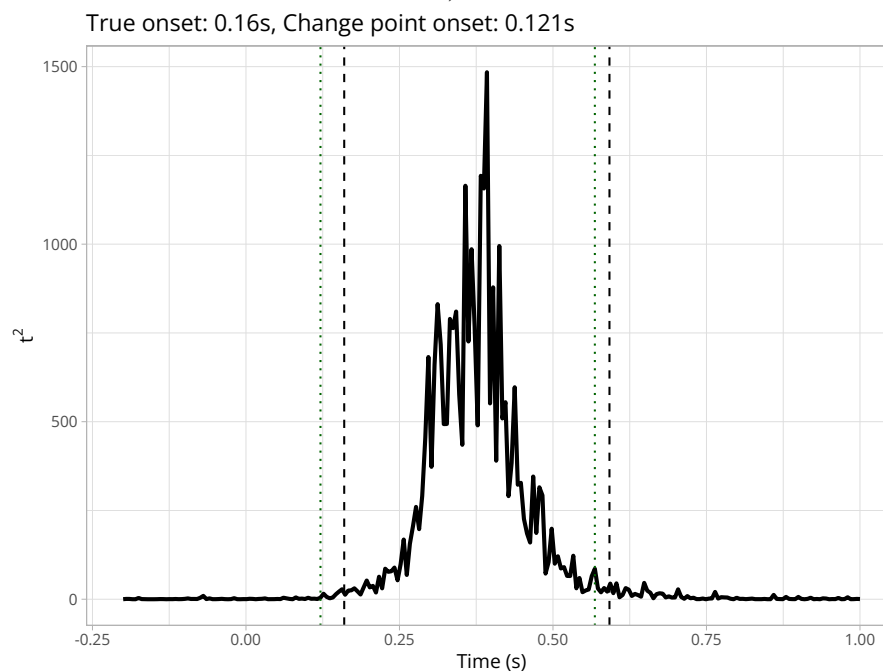


Figure 17

T-values timecourse with true onset/offset and onset/offset identified using the changepoint package (binary segmentation method, in green).



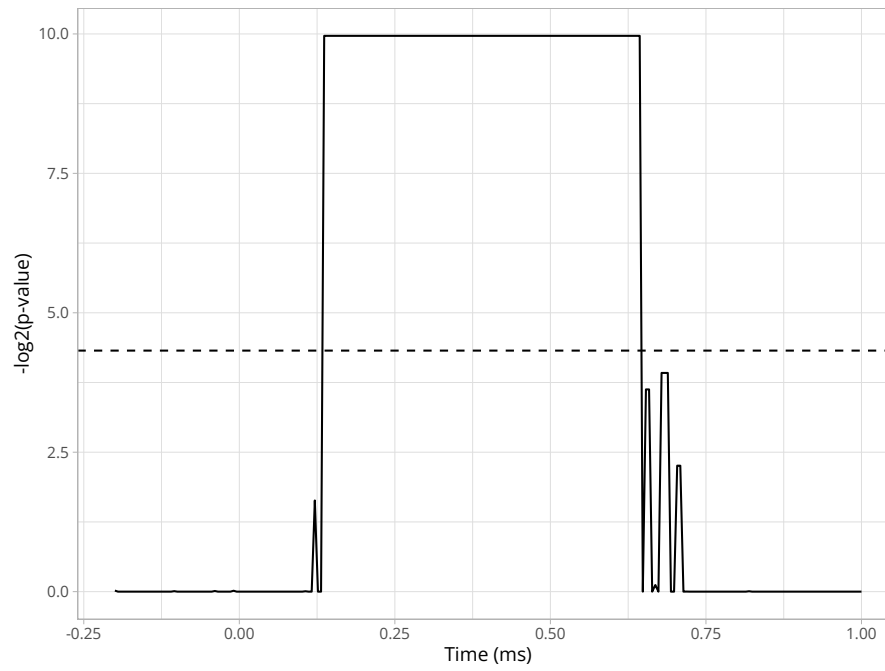
155 Now cluster-based permutations using MNE-Python ([Gramfort, 2013](#)) via the R package
 156 `reticulate` v 1.35.0 ([Ushey et al., 2024](#))...

```
# importing the decoding scores
p_values_df <- readRDS(file = "results/mne_permutation_decoding_scores.rds")

# plotting the s-values (-log2(p-values))
p_values_df %>%
  mutate(sval = -log2(pval) ) %>%
  ggplot(aes(x = time, y = sval) ) +
  geom_line(linewidth = 0.5) +
  geom_hline(yintercept = -log2(0.05), linetype = 2) +
  # geom_area(position = "identity") +
  labs(
    x = "Time (ms)",
    y = "-log2(p-value)"
  )
```

Figure 18

Cluster-based permutation tests via MNE-Python.



Results

157

158

...

Discussion

Summary of the proposed approach

Increasing potential usage

Prepare a wrapper R package and show how to call it in Python and integrate it with MNE-Python ([Gramfort, 2013](#)) pipelines...

Limitations and future directions

Conclusions

Packages

170

171 We used R version 4.2.3 ([R Core Team, 2023](#)) and the following R packages: brms v.
172 2.22.0 ([Bürkner, 2017, 2018, 2021](#)), changepoint v. 2.2.4 ([Killick et al., 2022b](#); [Killick & Eckley,](#)
173 [2014](#)), grateful v. 0.2.10 ([Rodriguez-Sanchez & Jackson, 2023](#)), knitr v. 1.45 ([Xie, 2014, 2015,](#)
174 [2023](#)), MetBrewer v. 0.2.0 ([Mills, 2022](#)), pakret v. 0.2.2 ([Gallou, 2024](#)), patchwork v. 1.2.0
175 ([T. L. Pedersen, 2024](#)), rmarkdown v. 2.29 ([Allaire et al., 2024](#); [Xie et al., 2018, 2020](#)), scales
176 v. 1.3.0 ([Wickham et al., 2023](#)), scico v. 1.5.0 ([T. L. Pedersen & Crameri, 2023](#)), tidybayes v.
177 3.0.6 ([Kay, 2023](#)), tidyverse v. 2.0.0 ([Wickham et al., 2019](#)).

References

- Allaire, J., Xie, Y., Dervieux, C., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., & Iannone, R. (2024). *rmarkdown: Dynamic documents for r*. <https://github.com/rstudio/rmarkdown>
- Bürkner, P.-C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1–28. <https://doi.org/10.18637/jss.v080.i01>
- Bürkner, P.-C. (2018). Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 10(1), 395–411. <https://doi.org/10.32614/RJ-2018-017>
- Bürkner, P.-C. (2021). Bayesian item response modeling in R with brms and Stan. *Journal of Statistical Software*, 100(5), 1–54. <https://doi.org/10.18637/jss.v100.i05>
- Combrisson, E., & Jerbi, K. (2015). Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy. *Journal of Neuroscience Methods*, 250, 126–136. <https://doi.org/10.1016/j.jneumeth.2015.01.010>
- Ehinger, B. V., & Dimigen, O. (2019). Unfold: An integrated toolbox for overlap correction, non-linear modeling, and regression-based EEG analysis. *PeerJ*, 7, e7838. <https://doi.org/10.7717/peerj.7838>
- Eklund, A., Nichols, T. E., & Knutsson, H. (2016). Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences*, 113(28), 7900–7905. <https://doi.org/10.1073/pnas.1602413113>
- Frossard, J., & Renaud, O. (2021a). Permutation Tests for Regression, ANOVA, and Comparison of Signals: The **permuco** Package. *Journal of Statistical Software*, 99(15). <https://doi.org/10.18637/jss.v099.i15>
- Frossard, J., & Renaud, O. (2021b). Permutation tests for regression, ANOVA, and comparison of signals: The permuco package. *Journal of Statistical Software*, 99(15), 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>
- Gallou, A. (2024). *pakret: Cite “R” packages on the fly in “R Markdown” and “Quarto”*. <https://CRAN.R-project.org/package=pakret>
- Gramfort, A. (2013). MEG and EEG data analysis with MNE-python. *Frontiers in Neuroscience*, 7. <https://doi.org/10.3389/fnins.2013.00267>
- Hayasaka, S. (2003). Validating cluster size inference: Random field and permutation methods. *NeuroImage*, 20(4), 2343–2356. <https://doi.org/10.1016/j.neuroimage.2003.08.003>
- Kay, M. (2023). *tidybayes: Tidy data and geoms for Bayesian models*. <https://doi.org/10.5281/zenodo.1308151>
- Killick, R., & Eckley, I. A. (2014). changepoint: An R package for changepoint analysis. *Journal of Statistical Software*, 58(3), 1–19. <https://www.jstatsoft.org/article/view/v058i03>
- Killick, R., Haynes, K., & Eckley, I. A. (2022a). *changepoint: An R package for changepoint analysis*. <https://CRAN.R-project.org/package=changepoint>
- Killick, R., Haynes, K., & Eckley, I. A. (2022b). *changepoint: An R package for changepoint analysis*. <https://CRAN.R-project.org/package=changepoint>
- King, J.-R., & Dehaene, S. (2014). Characterizing the dynamics of mental representations: the temporal generalization method. *Trends in Cognitive Sciences*, 18(4), 203–210. <https://doi.org/10.1016/j.tics.2014.01.002>
- Luck, S. J., & Gaspelin, N. (2017). How to get statistically significant effects in any ERP experiment (and why you shouldn’t). *Psychophysiology*, 54(1), 146–157. <https://doi.org/10.1111/psyp.12639>
- Maris, E. (2011). Statistical testing in electrophysiological studies. *Psychophysiology*, 49(4), 549–565. <https://doi.org/10.1111/j.1469-8986.2011.01320.x>

- Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of EEG- and MEG-data. *Journal of Neuroscience Methods*, 164(1), 177–190. <https://doi.org/10.1016/j.jneumeth.2007.03.024>
- Mills, B. R. (2022). *MetBrewer: Color palettes inspired by works at the metropolitan museum of art*. <https://CRAN.R-project.org/package=MetBrewer>
- Nalborczyk, L., Batailler, C., Løevenbruck, H., Vilain, A., & Bürkner, P.-C. (2019). An Introduction to Bayesian Multilevel Models Using brms: A Case Study of Gender Effects on Vowel Variability in Standard Indonesian. *Journal of Speech, Language, and Hearing Research*, 62(5), 1225–1242. https://doi.org/10.1044/2018_jslhr-s-18-0006
- Pedersen, E. J., Miller, D. L., Simpson, G. L., & Ross, N. (2019). Hierarchical generalized additive models in ecology: An introduction with mgcv. *PeerJ*, 7, e6876. <https://doi.org/10.7717/peerj.6876>
- Pedersen, T. L. (2024). *patchwork: The composer of plots*. <https://CRAN.R-project.org/package=patchwork>
- Pedersen, T. L., & Cramer, F. (2023). *scico: Colour palettes based on the scientific colour-maps*. <https://CRAN.R-project.org/package=scico>
- Pernet, C. R., Latinus, M., Nichols, T. E., & Rousselet, G. A. (2015). Cluster-based computational methods for mass univariate analyses of event-related brain potentials/fields: A simulation study. *Journal of Neuroscience Methods*, 250, 85–93. <https://doi.org/10.1016/j.jneumeth.2014.08.003>
- R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. <https://doi.org/10.7551/mitpress/3206.001.0001>
- Riutort-Mayol, G., Bürkner, P.-C., Andersen, M. R., Solin, A., & Vehtari, A. (2023). Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *Statistics and Computing*, 33(1), 17. <https://doi.org/10.1007/s11222-022-10167-2>
- Rodriguez-Sanchez, F., & Jackson, C. P. (2023). *grateful: Facilitate citation of r packages*. <https://pakillo.github.io/grateful/>
- Rosenblatt, J. D., Finos, L., Weeda, W. D., Solari, A., & Goeman, J. J. (2018). All-Resolutions Inference for brain imaging. *NeuroImage*, 181, 786–796. <https://doi.org/10.1016/j.neuroimage.2018.07.060>
- Rousselet, G. A. (2025). Using cluster-based permutation tests to estimate MEG/EEG onsets: How bad is it? *European Journal of Neuroscience*, 61(1), e16618. <https://doi.org/10.1111/ejn.16618>
- Sassenhagen, J., & Draschkow, D. (2019). Cluster-based permutation tests of MEG/EEG data do not establish significance of effect latency or location. *Psychophysiology*, 56(6). <https://doi.org/10.1111/psyp.13335>
- Skukies, R., & Ehinger, B. (2021). Modelling event duration and overlap during EEG analysis. *Journal of Vision*, 21(9), 2037. <https://doi.org/10.1167/jov.21.9.2037>
- Skukies, R., Schepers, J., & Ehinger, B. (2024, December 9). *Brain responses vary in duration - modeling strategies and challenges*. <https://doi.org/10.1101/2024.12.05.626938>
- Smith, S., & Nichols, T. (2009). Threshold-free cluster enhancement: Addressing problems of smoothing, threshold dependence and localisation in cluster inference. *NeuroImage*, 44(1), 83–98. <https://doi.org/10.1016/j.neuroimage.2008.03.061>
- Teichmann, L. (2022). An empirically driven guide on using bayes factors for m/EEG decoding. *Aperture Neuro*, 2, 1–10. <https://doi.org/10.52294/apertureneuro.2022.2.maoc6465>
- Ushey, K., Allaire, J., & Tang, Y. (2024). *Reticulate: Interface to 'python'*. <https://CRAN.R-project.org/package=reticulate>
- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M.,

- Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43), 1686. <https://doi.org/10.21105/joss.01686>
- Wickham, H., Pedersen, T. L., & Seidel, D. (2023). *scales: Scale functions for visualization*. <https://CRAN.R-project.org/package=scales>
- Wood, S. N. (2003). Thin Plate Regression Splines. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 65(1), 95–114. <https://doi.org/10.1111/1467-9868.00374>
- Wood, S. N. (2004). Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models. *Journal of the American Statistical Association*, 99(467), 673–686. <https://doi.org/10.1198/016214504000000980>
- Xie, Y. (2014). knitr: A comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Chapman; Hall/CRC.
- Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. <https://yihui.org/knitr/>
- Xie, Y. (2023). *knitr: A general-purpose package for dynamic report generation in r*. <https://yihui.org/knitr/>
- Xie, Y., Allaire, J. J., & Golemund, G. (2018). *R markdown: The definitive guide*. Chapman; Hall/CRC. <https://bookdown.org/yihui/rmarkdown>
- Xie, Y., Dervieux, C., & Riederer, E. (2020). *R markdown cookbook*. Chapman; Hall/CRC. <https://bookdown.org/yihui/rmarkdown-cookbook>
- Yeung, N., Bogacz, R., Holroyd, C. B., & Cohen, J. D. (2004). Detection of synchronized oscillations in the electroencephalogram: An evaluation of methods. *Psychophysiology*, 41(6), 822–832. <https://doi.org/10.1111/j.1469-8986.2004.00239.x>

Appendix A

Mathematical formulation of the bivariate GAM

To model cross-temporal generalisation matrices of decoding performance (ROC AUC), we extended the initial (decoding) GAM to take into account the bivariate temporal distribution of AUC values, thus producing naturally smoothed estimates (timecourses) of AUC values and posterior probabilities. This model can be written as follows:

$$\begin{aligned} \text{AUC}_i &\sim \text{Beta}(\mu_i, \phi) \\ g(\mu_i) &= f(\text{train}_i, \text{test}_i) \end{aligned}$$

where we assume that AUC values come from a Beta distribution with two parameters μ and ϕ . We can think of $f(\text{train}_i, \text{test}_i)$ as a surface (a smooth function of two variables) that we can model using a 2-dimensional splines. Let $\mathbf{s}_i = (\text{train}_i, \text{test}_i)$ be some pair of training and testing samples, and let $\mathbf{k}_m = (\text{train}_m, \text{test}_m)$ denote the m^{th} knot in the domain of train_i and test_i . We can then express the smooth function as:

$$f(\text{train}_i, \text{test}_i) = \alpha + \sum_{m=1}^M \beta_m b_m(\tilde{s}_i, \tilde{k}_m)$$

Note that $b_m(\cdot)$ is a basis function that maps $R \times R \rightarrow R$. A popular bivariate basis function uses *thin-plate splines*, which extend to $\mathbf{s}_i \in \mathbb{R}^d$ and ∂l_g penalties. These splines are designed to interpolate and approximate smooth surfaces over two dimensions (hence the “bivariate” term). For $d = 2$ dimensions and $l = 2$ (smoothness penalty involving second order derivative):

$$f(\tilde{s}_i) = \alpha + \beta_1 x_i + \beta_2 z_i + \sum_{m=1}^M \beta_{2+m} b_m(\tilde{s}_i, \tilde{k}_m)$$

using the the radial basis function given by:

$$b_m(\tilde{s}_i, \tilde{k}_m) = \left\| \tilde{s}_i - \tilde{k}_m \right\|^2 \log \left\| \tilde{s}_i - \tilde{k}_m \right\|$$

where $\|\mathbf{s}_i - \mathbf{k}_m\|$ is the Euclidean distance between the covariate \mathbf{s}_i and the knot location \mathbf{k}_m .

Appendix B

Threshold-free cluster enhancement

Cluster-based permutation approaches require defining a cluster-forming threshold (e.g., a t - or f -value) as the initial step of the algorithm. As different cluster-forming thresholds lead to clusters with different spatial or temporal extent, this threshold modulates the sensitivity of the subsequent permutation test. The threshold-free cluster enhancement method (TFCE) was introduced by Smith & Nichols (2009) to overcome this arbitrary threshold.

In brief, the TFCE method works as follows. Instead of picking an arbitrary cluster-forming threshold (e.g., $t = 2$), we try all (or many) possible thresholds in a given range and check whether a given timestep/voxel belongs to a significant cluster under any of the set of thresholds... Then, instead of using cluster mass, we use a weighted average between the cluster extend (e , how broad is the cluster, that is, how many connected samples it contains) and the cluster height (h , how high is the cluster, that is, how large is the test statistic) according to the formula:

$$\text{TFCE} = \int_h e(h)^E h^H dh$$

Where... the parameters E and H are set a priori and control the influence of the extend and height on the TFCE. Then, p -value for timestep/voxel i is computed by comparing it TFCE with the null distribution of TFCE values. For each permuted signal, we keep the maximal value over the whole signal for the null distribution of the TFCE.... But see Sassenhagen & Draschkow (2019)...

Appendix C

Using the R package and integration with MNE-Python

338 Explain how to use the R package and to integrate it with MNE epochs...

```
# to-do adding some code here...
```