Analyzing time series data using `clusterperm.lmer`

Cesko C. Voeten

12 September 2022

This vignette models ERP data, an example of *time series data*, using permutation testing. The `permutes` package implements two approaches, one based on package `lmPerm`, which is shown in the other vignette, and the other based on package `buildmer`, which is shown here. The `lmPerm` approach can do multivariate responses, but cannot run generalized linear models and only has very limited support for random effects (i.e.: the `Error()` functionality works to run ANOVA with random effects, but only using Type II SS). The `buildmer` approach can fit any model that function `buildmer` from package `buildmer` can (i.e.: `lm`, `glm`, `lmer`, and `glmer` models), but cannot fit multiple responses. Also, the `buildmer` approach can use the cluster-mass test by Maris & Oostenveld (2007), using the relevant machinery in package `permuco`.

Sample data

The dataset we will be working with is called `MMN`, which is an excerpt from a passive-oddball task conducted by Jager (in prep.). A proper method section with more details should eventually be found in that paper, but a brief and incomplete summary follows here in order to make it easier to work with the data. Jager (in prep.) presented Dutch learners of English with a stream of isolated vowels from English or from Dutch created by means of formant synthesis. Each of her six conditions (summarized in the below table; the package supplies data from two of these conditions) each having used four vowels: three of the same phonetic category (e.g. three realizations of the DRESS vowel with slightly different formant frequencies) and one of a different phonetic category (e.g. the TRAP vowel). The first set of vowels, termed 'standards', were presented frequently; the fourth vowel, the 'deviant', infrequently interrupted the stream of standards. Participants were instructed not to pay attention to this stream of sounds, and watched a silent movie to keep them occupied. EEG recordings of 30 electrodes (T7 and T8 were excluded) were recorded while the participants were exposed to the vowel stimuli. At the presentation of one of the deviant vowels, we expect to observe a negative deflection in the EEG signal about 200 milliseconds after stimulus onset, originating from frontal and central electrode sites. This effect is called the 'mismatch negativity'. A second effect called the 'P300' may also occur, but is ignored here. The data supplied with `permutes` are a subset of the vowel pairs tested by Jager (in prep.) consisting of the English vowel in DRESS presented as a standard vs. as a deviant, in both cases against the vowel from the Dutch word ZET.

The first few rows of the data look as follows:

```
library(permutes)
head(MMN)
```

```
##       Fp1     AF3      F7      F3    FC1    FC5     C3    CP1     CP5     P7
## 1 -0.0002 -1.7087  0.1357  0.3822 0.1538 1.1556 0.7552 0.7864 -0.0222 0.3399
## 2  0.2871 -2.2155  0.0740  0.3518 0.1051 1.4625 0.6329 0.7107 -0.0411 0.3134
## 3  0.4573 -2.4930  0.0070  0.2775 0.0810 1.6963 0.5497 0.6228 -0.0651 0.2984
## 4  0.5319 -2.4990 -0.0517  0.1462 0.0860 1.8356 0.5089 0.5332 -0.0914 0.2901
## 5  0.5321 -2.2408 -0.0890 -0.0346 0.1177 1.9058 0.4975 0.4489 -0.1169 0.2790
## 6  0.4875 -1.7814 -0.0951 -0.2321 0.1620 1.9712 0.4907 0.3710 -0.1400 0.2545
##       P3     Pz     PO3      O1      Oz      O2     PO4     P4     P8    CP6
## 1 0.4392 0.6500 -0.1648 -0.5426 -0.5382 -0.9072 -0.0193 0.7312 0.0102 1.0790
## 2 0.4209 0.5860 -0.1576 -0.5430 -0.5308 -0.8392  0.0938 0.7830 0.0822 1.1723
```

```
## 3 0.3585 0.5030 -0.1631 -0.5533 -0.5394 -0.7937  0.1809 0.7927 0.1162 1.1724
## 4 0.2612 0.4114 -0.1817 -0.5750 -0.5648 -0.7628  0.2506 0.7774 0.1246 1.1051
## 5 0.1417 0.3183 -0.2126 -0.6071 -0.6043 -0.7330  0.3125 0.7553 0.1226 1.0042
## 6 0.0138 0.2272 -0.2520 -0.6417 -0.6443 -0.6848  0.3767 0.7424 0.1276 0.9066
##      CP2     C4    FC6    FC2      F4      F8     AF4     Fp2     Fz     Cz
## 1 0.8142 1.3876 0.4432 0.7724 -0.1918  0.5416  0.0526 -0.1283 0.1105 0.9059
## 2 0.7216 1.4579 0.6214 0.7768 -0.1749  0.4926 -0.1035 -0.6480 0.0827 0.7886
## 3 0.5969 1.4392 0.7589 0.7753 -0.1024  0.3913 -0.2377 -1.2004 0.0861 0.6604
## 4 0.4643 1.3590 0.8526 0.7678  0.0054  0.2513 -0.3246 -1.7130 0.1086 0.5332
## 5 0.3446 1.2564 0.9184 0.7586  0.1225  0.1030 -0.3462 -2.1017 0.1352 0.4168
## 6 0.2484 1.1672 0.9760 0.7520  0.2162 -0.0195 -0.2996 -2.2827 0.1481 0.3172
##        Time Subject Session Deviant
## 1  1.171875       1       0       1
## 2  3.125000       1       0       1
## 3  5.078125       1       0       1
## 4  7.031250       1       0       1
## 5  8.984375       1       0       1
## 6 10.937500       1       0       1
```

```
nrow(MMN) #how many observations?
```

```
## [1] 22407
```

```
length(unique(MMN$Time)) #how many timepoints?
```

```
## [1] 231
```

The first 30 columns are the 30 sampled EEG electrodes. The `Time` column denotes the moment from stimulus onset, in milliseconds, of each measurement type. `Subject` is the participant, and `Session` is the session of the experiment minus one, to make it a proper dummy indicating whether the datapoint is from the first (0) or the second session (1). Finally, `Deviant` is a dummy indicating whether the stimulus was a standard (0) or a deviant (1), explained in the next paragraph. Note that, contrary to the results and recommendations in Brysbaert (2007), Jager (in prep.) averaged over all her items belonging to the same condition in this experiment.

Time series data such as these are special because the data consist of multiple, *strongly correlated*, measurements of the same signal. This generates two problems. The first is a research problem: which portion of this time series do we want to analyze? The `permutes` package was designed to help researchers answer precisely this question. The second problem is of a statistical nature: how do we handle the strong autocorrelation present throughout this sampled window? Note that in the case of ERP data, these same two problems are additionally encountered in the *spatial* domain: what combination of electrodes ('region of interest') do we want to analyze, and how do we deal with the spatial correlation present in data measured from neighboring sites? However, in the case of the mismatch-negativity component, we have *a priori* reason to assume that it will be reflected most strongly in the `Fz` electrode. We will therefore only consider this electrode; for an analysis that includes all electrodes, see the other vignette.

Determining the window

The problem we face equates to what is known in the ERP literature as determining the *window*. The normal way to do this is to run a regression analysis on every individual timepoint, and take as the window the point where a large sequence of significant p-values occurs. If we plot time horizontally and p-value vertically, we can use this approach to select a time window.

It should be noted that this approach suffers from an extreme multiple-comparisons problem: we will be running 231 regressions! When compared to an asymptotic null distribution, the p-values will be spuriously significant in, expectedly, 12 random combinations. The permutation testing approach to time series data helps with this problem by *inferring* the null distribution from randomly perturbed data. When including

random effects, this is slightly complicated; `clusterperm.lmer` uses a simplified version of the algorithm by Lee & Braun (2012) for this.

The below code runs a permutation test series on the `MMN` data and plots the results as a heatmap. Note that permutation testing is not deterministic (the permutations are chosen randomly), so your results may be slightly different from mine.

```
perms <- clusterperm.lmer(Fz ~ Deviant * Session + (Deviant + Session | Subject),
              data=MMN,series.var=~Time,progress='text')
```

This takes a few seconds to run. We can speed it up by parallelizing the testing of the individual timepoints. On most systems (including Windows Rterm, but *not* Windows RGUI), we can add `outfile=''` to `makeCluster` and `progress=''` (or any other character string) to obtain some rudimentary progress indicator even when running in parallel. We do not do so now.

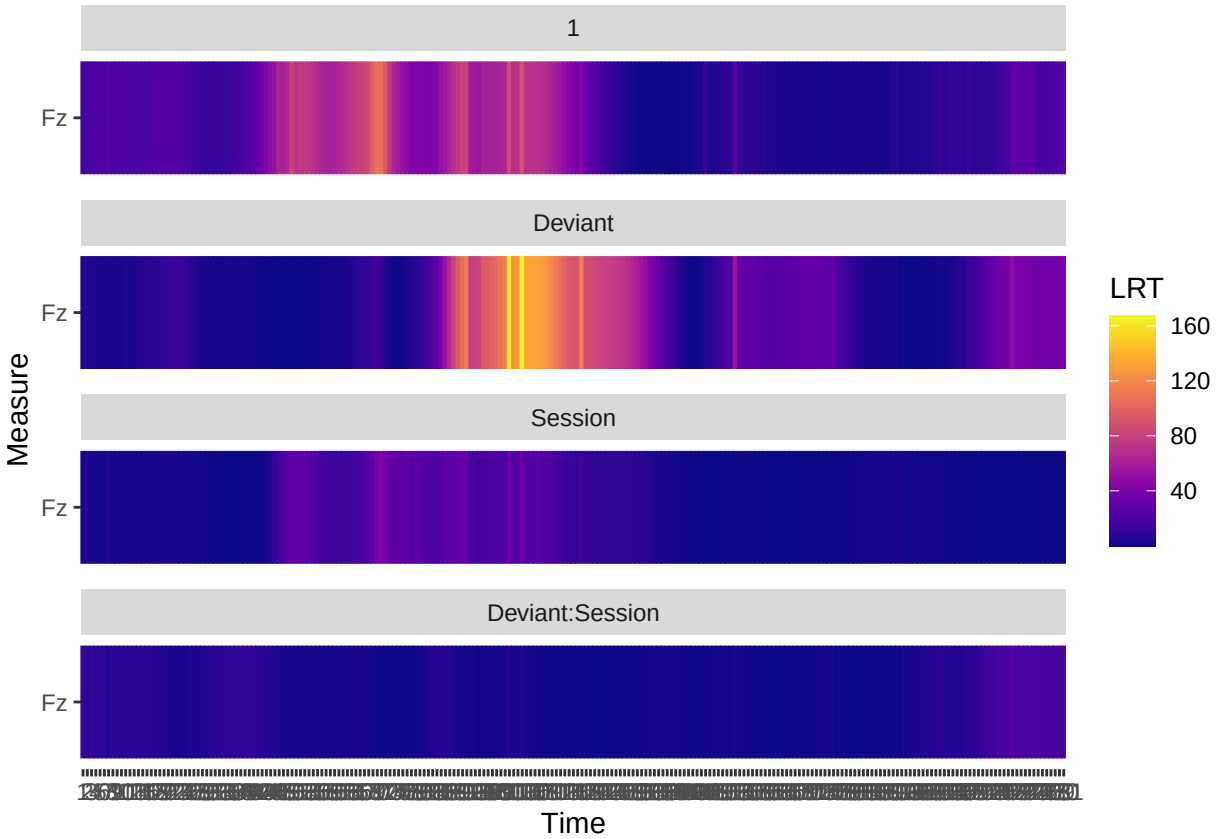
```
library(doParallel)
cl <- makeCluster(2) #or more
registerDoParallel(cl)
perms <- clusterperm.lmer(Fz ~ Deviant * Session + (Deviant + Session | Subject),
              data=MMN,series.var=~Time,parallel=TRUE)
```

We can then plot the results:

```
plot(perms)
```

```
## Loading required namespace: ggplot2
```

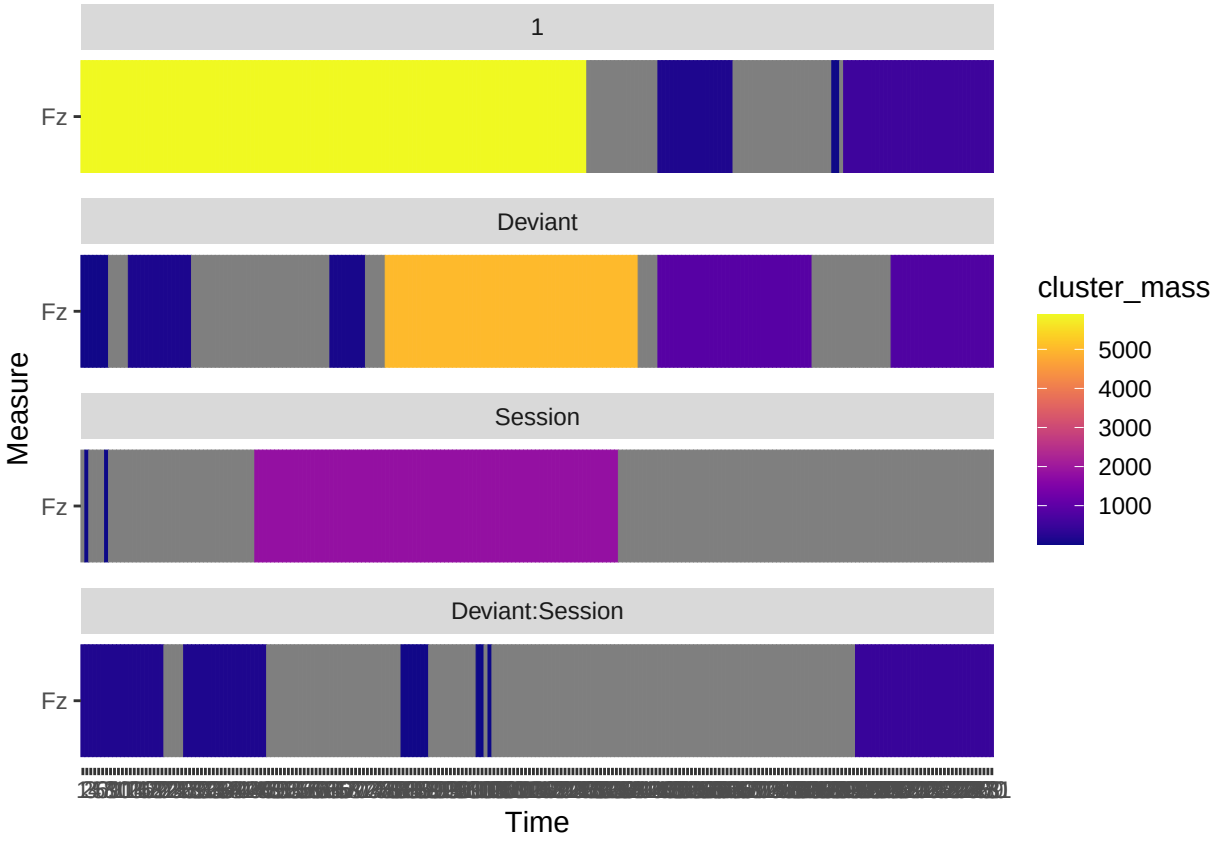
```
## Loading required namespace: viridis
```



By default, the `plot` method from `permutes` plots likelihood-ratio tests, as this is what the permutation procedure by Lee & Braun (2012) operates on. However, these permutation p-values aren't useful in and of

themselves, at least if following Maris & Oostenveld (2007). Instead, one should use the permuted LRTs to compute a cluster-mass statistic. This can also be plotted:

```
plot(perms,type='cluster_mass')
```



This suggests some clear windows, of which the strongest one is found for the `Deviant` effect. We will focus on this window. Because `permutes` objects are also normal data frames, this is easily achieved.

```
head(perms) #what do we have?
```

```
##   Time Measure          Factor       LRT       beta          t cluster
## 1    1      Fz               1 18.635655  1.1175222  3.0594588       1
## 2    1      Fz         Deviant  4.577659 -0.3979960 -0.7706906       1
## 3    1      Fz         Session  2.634262  0.4890000  1.0367109       0
## 4    1      Fz Deviant:Session 11.915262 -1.1857400 -1.6624008       1
## 5    2      Fz               1 19.947771  1.1143481  3.6544409       1
## 6    2      Fz         Deviant  5.847704 -0.5217075 -0.9588454       1
##   cluster_mass          p p.cluster_mass
## 1   5881.47977 0.00000000    0.000999001
## 2     32.94252 0.05494505    1.000000000
## 3           NA 0.12887113             NA
## 4    163.42550 0.00000000    0.000999001
## 5   5881.47977 0.00000000    0.000999001
## 6     32.94252 0.03496503    1.000000000
```

```
perms2 <- perms[perms$Factor == 'Deviant' & !is.na(perms$cluster_mass),]
perms2 <- perms2[perms2$cluster_mass == max(perms2$cluster_mass),]
print(perms2$Time)
```

```
##  [1] 78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96
## [20] 97  98  99  100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
## [39] 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134
## [58] 135 136 137 138 139 140 141
## 231 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 231
```

For technical reasons, the `Time` variable was coerced to a factor. We see that *contiguous* permutation significance was achieved from the 78th to the 141 level of this factor. The below code shows that this corresponds to a window from 151 ms to 275 ms post-stimulus-onset.

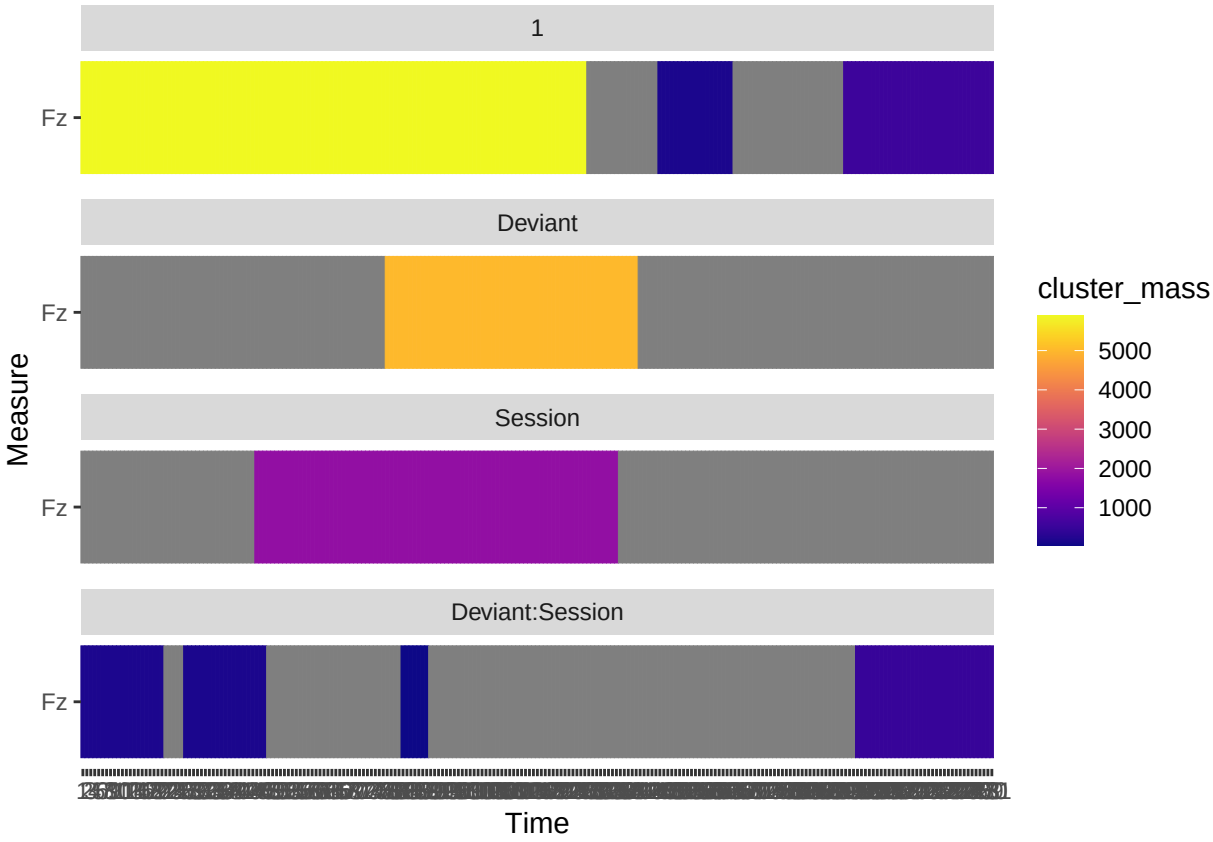
```
print(unique(MMN$Time)[c(78,141)])
```

```
## [1] 151.5625 274.6094
```

Interpreting the results

We can now proceed in two ways. The first way follows a literal interpretation of Maris & Oostenveld (2007), asking for the p-values corresponding to the cluster-mass test, *and that is the entirety of the analysis*. We can obtain this by reading the results themselves through `print(perms)` (`permutes` objects are also normal data frames), or by specifying the `sig` option to `plot` so as to only plot those results that reached cluster-mass significance:

```
plot(perms,type='cluster_mass',sig='p.cluster_mass')
```



This confirms significance for the chosen factor in the chosen window, as well as some other factor–window combinations that we will not discuss.

Alternatively, we can use an approach more common in psycholinguistics, which is to use the found window as input for our *actual* analysis, which can be a linear mixed-effects model. We will average our data along the aforementioned window, and then run a *permutation linear mixed-effects model* to analyze the data, so

that we can easily obtain robust p-values. The advantage of these permutation p-values is, again, that we could run multiple such models without requiring any Bonferroni correction. For the present single model, however, an ordinary linear mixed-effects model would also suffice.

Previous versions of `permutes` produced `(g)lm`, `gam`, or `(g)lmer` objects here; the current version does not, instead producing the same output as the other permutation functions. For this reason, it is recommended to start by fitting a linear mixed model, and to then compute the permutation p-values separately. The linear mixed model can be fitted as follows:

```
library(buildmer)
data <- MMN[MMN$Time > 151 & MMN$Time < 275,]
data <- aggregate(Fz ~ Deviant + Session + Subject,data,mean)
lmm <- buildmer(Fz ~ Deviant * Session + (Deviant + Session | Subject),data,
                buildmerControl=list(direction='order',ddf='lme4'))
```

```
## Determining predictor order

## Fitting via lm: Fz ~ 1

## Currently evaluating LRT for: Deviant, Session

## Fitting via lm: Fz ~ 1 + Deviant

## Fitting via lm: Fz ~ 1 + Session

## Updating formula: Fz ~ 1 + Deviant

## Currently evaluating LRT for: Session

## Fitting via lm: Fz ~ 1 + Deviant + Session

## Updating formula: Fz ~ 1 + Deviant + Session

## Currently evaluating LRT for: Deviant:Session

## Fitting via lm: Fz ~ 1 + Deviant + Session + Deviant:Session

## Updating formula: Fz ~ 1 + Deviant + Session + Deviant:Session

## Fitting via gam, with REML: Fz ~ 1 + Deviant + Session +
##     Deviant:Session

## Currently evaluating LRT for: 1 | Subject

## Fitting via lmer, with REML: Fz ~ 1 + Deviant + Session +
##     Deviant:Session + (1 | Subject)

## Updating formula: Fz ~ 1 + Deviant + Session + Deviant:Session + (1 |
##     Subject)

## Currently evaluating LRT for: Deviant | Subject, Session | Subject

## Fitting via lmer, with REML: Fz ~ 1 + Deviant + Session +
##     Deviant:Session + (1 + Deviant | Subject)

## Fitting via lmer, with REML: Fz ~ 1 + Deviant + Session +
##     Deviant:Session + (1 + Session | Subject)

## Updating formula: Fz ~ 1 + Deviant + Session + Deviant:Session + (1 +
##     Deviant | Subject)

## Currently evaluating LRT for: Session | Subject

## Fitting via lmer, with REML: Fz ~ 1 + Deviant + Session +
##     Deviant:Session + (1 + Deviant + Session | Subject)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
## Ending the ordering procedure due to having reached the maximal
##     feasible model - all higher models failed to converge. The types of
##     convergence failure are: Singular fit
```

where the control argument `direction` specifies that we want to `order` the model terms to obtain the maximal feasible model. In `buildmer`, the default is `c('order','backward')`, which will also perform backward stepwise elimination. `permutes` by default modifies this to use `order` only, however, if you specify an explicit `buildmerControl` argument, all defaults specified by `permutes` will be overridden. We hence need to explicitly specify that we only want the term-ordering step, and that we do not also want backward stepwise elimination. The `ddf='lme4'`, also a default set by `permutes` that deviates from those in `buildmer`, saves a little computation time by not coercing the model to an `lmerTest` model, from which denominator degrees of freedom can be computed. Since we are doing permutation tests, we don't need to calculate any degrees of freedom.

From the above, we observe that the highest random-effects structure that this model could support was `(1 + Deviant | Subject)`. Given that the data only contain 95 observations—due to Jager's averaging over items—this is not surprising. Keeping this in mind, we can use function `perm.lmer` to obtain the corresponding permutation p-values:

```
perm <- perm.lmer(Fz ~ Deviant * Session + (Deviant + Session | Subject),data)
```

It does not matter here whether we specify `(Deviant + Session | Subject)` or only `(Deviant | Subject)`; the permutation tests are based on the maximal *feasible* model in any case. The permutation p-values (and corresponding regression coefficients) are provided as a simple dataframe, which we can print:

```
print(perm)
```

```
## data frame with 0 columns and 0 rows
```

The main conclusion to be drawn from these results is that there is a significant mismatch negativity of -1.91 microvolts. No evidence is found for a change in this MMN over the two sessions of the experiment.

This concludes our expository foray into these data.

References

Brysbaert, M. (2007). The language-as-fixed-effect-fallacy: Some simple SPSS solutions to a complex problem.

Jager, L. (in prep.).

Lee, O. E., & Braun, T. M. (2012). Permutation tests for random effects in linear mixed models. *Biometrics*, *68*(2), 486–493. https://doi.org/10.1111/j.1541-0420.2011.01675.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