Some High-level RAVE Functions

Zhengjia Wang

2023-11-06

RAVE is GUI-driven, but it is not an omni tool that provides everything. At some point, you will need to program by yourself. RAVE provides both high-level and low-level functions for you to load data. This tutorial mainly introduce high-level classes and functions.

- High-level functions require the subjects to be preprocessed using RAVE infrastructure.

Subject utility

You can get a subject class by

```
subject = raveio::as_rave_subject("demo/DemoSubject")
print(subject)
```

```
## RAVE subject <demo/DemoSubject>
```

The RAVE subject class provides access to all the meta information. For example,

```
# electrodes.csv
electrode_table = subject$get_electrode_table()
## Un-comment to print
# print(electrode_table)

# Trial epochs
subject$epoch_names
```

```
## [1] "auditory_onset"
```

```
epoch = subject$get_epoch("auditory_onset")
## Un-comment to print
# print(epoch$table)

# References
subject$reference_names
```

```
## [1] "_unsaved" "default"  "noref"
```

```
reference_table = subject$get_reference("default")
## Un-comment to print
# print(reference_table)
```

> Hint: Type `subject$` and wait or press `Tab` key in RStudio to get autocompletion for all possible fields and methods.

Electrode channels

Once electrode channel data is imported, you can create electrode instances:

```
# channel 14
e14 = raveio::new_electrode(subject, 14)
```

To load data from this channel, you will need to set which epoch and trial intervals. If the channel is referenced, you might also want to set reference channel as well.

```
# set loading epoch
epoch = subject$get_epoch("auditory_onset")
e14$set_epoch(epoch)

# set loading time window from 1s before onset and 2s after onset
e14$trial_intervals = c(-1, 2)

# set reference channel, using channel 13 as reference
ref = raveio::new_reference(subject, 13)
e14$set_reference(ref)
```

```
## <Reference electrode>
##   Project: demo
##   Subject: DemoSubject
##   Reference label: 13
##   Location type: iEEG
##   Signal type: LFP
##   Sample rates:
##     - Analog-trace (voltage): 2000
##     - Power-phase (coefficients): 100
##   Epoch:  auditory_onset
##   Trial windows:  list(c(-1, 2))
```

Now you can load voltage

```
voltage_data = e14$load_data("voltage")
str(dimnames(voltage_data))
```

```
## List of 3
##  $ Time     : num [1:6001] -1 -1 -0.999 -0.999 -0.998 ...
##  $ Trial    : num [1:287] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Electrode: int 14
```

In many scenarios, the loaded data could be too large for the computer memories. RAVE stores the loaded data as `FileArray` - a format that stores data on disk and loads data only when needed. To get the numerical values, simply use `[]` operator:

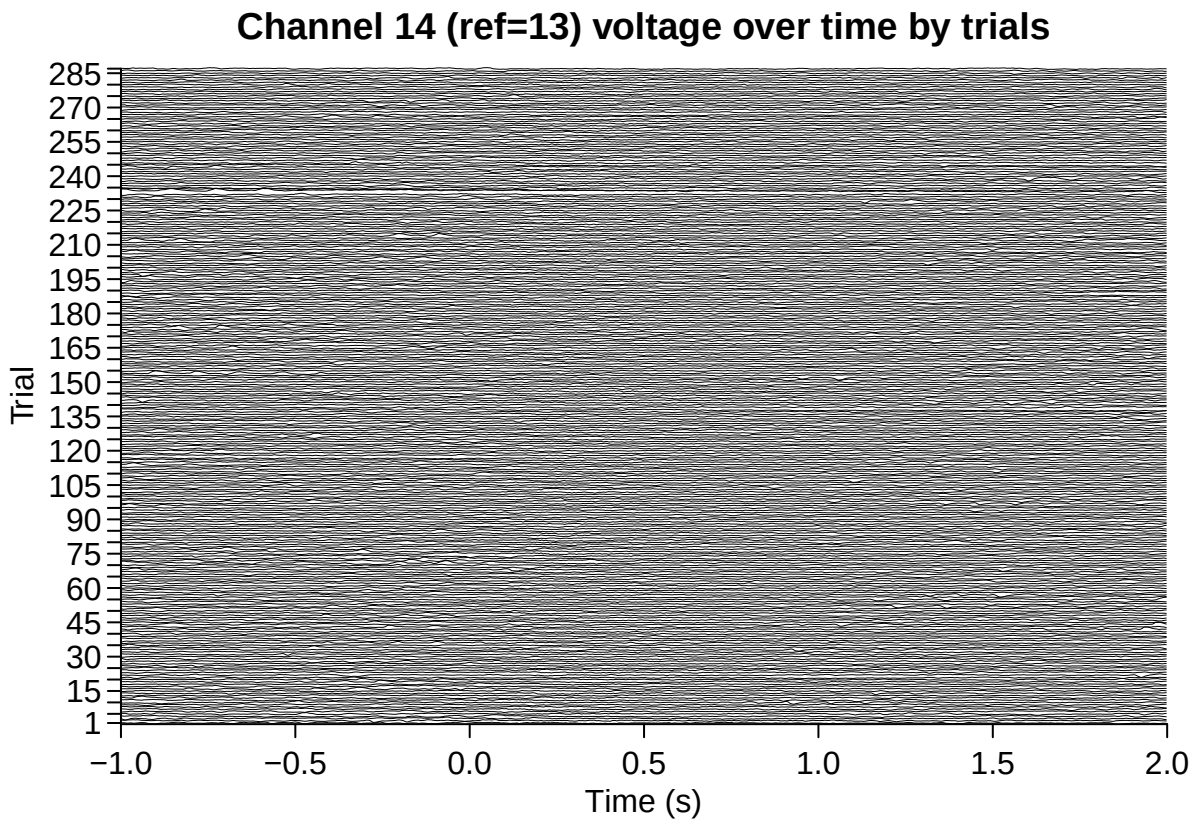
```
vdata = voltage_data[drop = FALSE]
# Snapshot the data
str(vdata)
```

```
##  num [1:6001, 1:287, 1] 92.6 87.8 54 83.1 116.4 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ Time     : chr [1:6001] "-1" "-0.9995" "-0.999" "-0.9985" ...
##   ..$ Trial    : chr [1:287] "1" "2" "3" "4" ...
##   ..$ Electrode: chr "14"
```

Let's plot the `vdata`

```
# Convert to trial x time
vdata_transform = t(vdata[,,1])
ravetools::plot_signals(
```

```
  vdata_transform,
  sample_rate = 2000,
  time_shift = -1,
  ylab = "Trial",
  main = "Channel 14 (ref=13) voltage over time by trials"
)
```



Pre-designed data repository

If you have finished RAVE preprocessing (Notch filter, wavelet, epoch, re-referencing), instead of loading channels one-by-one, you can use built-in data loading functions. These functions start with `raveio::prepare_*`. For example, if you want to

1. load the power spectrogram for channel 14 and 15
2. Baseline the data (baseline window: `-1~-0.5`)
3. Subset trial data on stimulus `drive_av`
4. Plot the average power percentage change

```
# ---- Load data ---------------------------------
repository = raveio::prepare_subject_power(
  subject,
  electrodes = c(14, 15),
  reference_name = "default",
  epoch_name = "auditory_onset",
  time_windows = c(-1, 2),
  verbose = FALSE
)

# ---- Baseline ----------------------------------
```

```
repository <- raveio::power_baseline(repository,
                                     baseline_windows = c(-1, -0.5),
                                     method = "percentage")

# ---- Get baseline'd power, and subset --------
trial_nums =
  repository$epoch$trials[ repository$epoch$table$Condition %in% c("drive_av") ]

sliced_data = subset(
  repository$power$baselined,
  Trial ~ Trial %in% trial_nums
)
str(sliced_data)
```

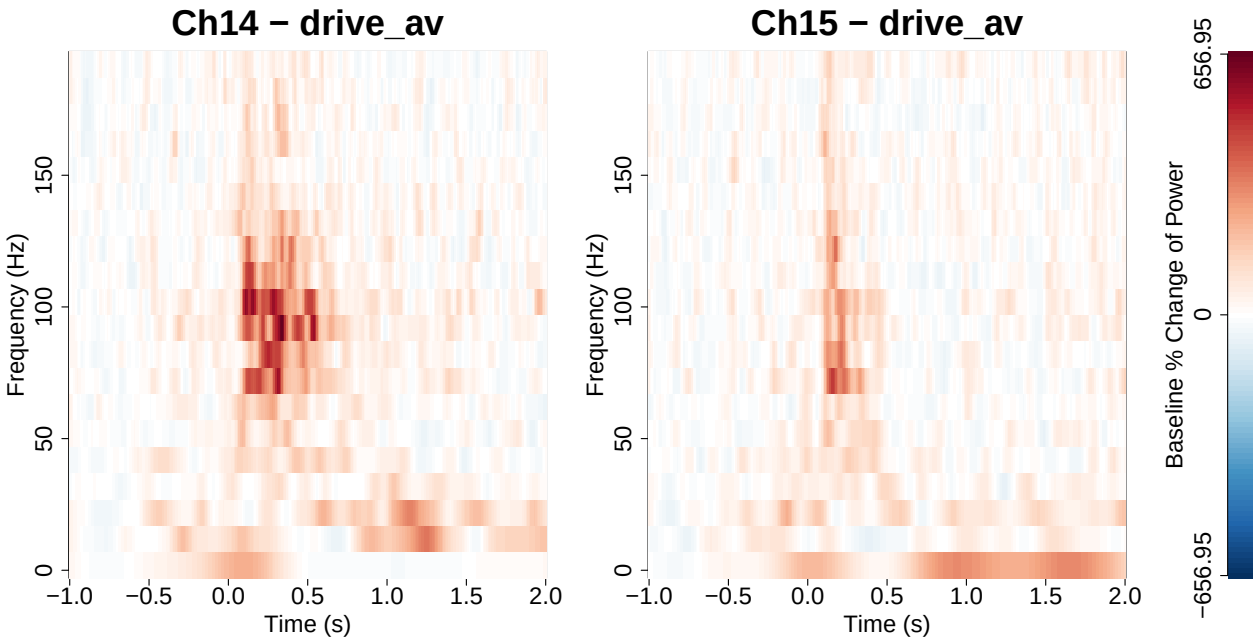
```
##  num [1:20, 1:301, 1:15, 1:2] 133.3 227.6 66.4 -64.9 -72.4 ...
##  - attr(*, "dimnames")=List of 4
##   ..$ Frequency: chr [1:20] "2" "12" "22" "32" ...
##   ..$ Time     : chr [1:301] "-1" "-0.99" "-0.98" "-0.97" ...
##   ..$ Trial    : chr [1:15] "10" "22" "43" "67" ...
##   ..$ Electrode: chr [1:2] "14" "15"
```

```
# ---- Collapse by trial and electrodes and plot
# keep = c(2, 1, 4) means keep the 1st (Frequency), 2nd (Time) ,
#   and 4th (Electrode channel) dimensions
#   and transpose the first two, so the result is
#     Time x Frequency x Electrode
avg_data = raveio::collapse2(x = sliced_data,
                             keep = c(2, 1, 4),
                             method = "mean")

# plot: set up layout
ravebuiltins:::layout_heat_maps(k = 2, max_col = 2)
```

```
## Warning: replacing previous import 'lme4::lmer' by 'lmerTest::lmer' when
## loading 'ravebuiltins'
```

```
par(cex.lab = 2, cex.main = 3, cex.axis = 2,
    mar = c(4.1, 6.1, 4.1, 1.1))

# color palette
pal = ravebuiltins::expand_heatmap("BlueWhiteRed")

# plotting range
zlim = max(avg_data) * c(-1, 1)

# draw channel 14
image(x = repository$time_points,
      y = repository$frequency,
      z = avg_data[,,1], col = pal, zlim = zlim,
      xlab = "Time (s)", ylab = "Frequency (Hz)",
      main = "Ch14 - drive_av")

# draw channel 15
image(x = repository$time_points,
      y = repository$frequency,
```

```
      z = avg_data[,,2], col = pal, zlim = zlim,
      xlab = "Time (s)", ylab = "Frequency (Hz)",
      main = "Ch15 - drive_av")

# draw legend
legend_ticks = seq(zlim[[1]], zlim[[2]], length.out = length(pal))
image(y = legend_ticks, z = matrix(legend_ticks, nrow = 1),
      col = pal, axes = FALSE, xlab = "", ylab = "Baseline % Change of Power")
axis(side = 2, at = c(zlim, 0), labels = c(sprintf("%.2f", zlim), "0"))
```



3D viewer

Here is a simple example for plotting the brain (more tutorials on 3D viewers will come soon)

```
brain = raveio::rave_brain(subject)
plot(brain)
```