Estimation of Behavioral Entropy Rate : R-Software

ccber: an R Package for the Estimation of Behavioral Entropy Rate - Developed for the Conte Center @ UCI

See reference: *Early Life Exposure to Unpredictable Maternal Sensory Signals Influences Cognitive Development: A Cross-Species Approach* Elysia Davis, Stephanie Stour, Jenny Molet, Brian Vegetabile, Laura Glynn, Curt Sandman, Kevin Heins, Hal Stern, and Tallie Baram: To appear in the Proceedings of the National Academy of Sciences

Installing ccber

The package devtools is required to install this R package from this Github repository. Install this package first if it is not already installed.

```
install.packages('devtools', dependencies = TRUE)
```

Once that package has been installed, use the following to install ccber

```
devtools::install_github('bvegetabile/ccber')
```

Load the package to begin analysis!

```
library('ccber')
```

Using ccber with Observer Files

To use ccber to estimate behavioral entropy rate, see the Conte Center website for a description of how to set up files and record observations. The input files for these functions are described there...

Running a single file

To run ccber with a single file use the following function

```
ber_analyze_file(f_loc,
                 plot_all=F,
                 plots_to_file=F,
                 tactile_padding = 1.0,
                 auditory_padding = 1.0,
                 behavior_types=list("mom_auditory_types" = c('Vocal'),
                                     "mom_tactile_types" = c('TouchBaby',
                                                             'HoldingBaby'),
                                     "mom_visual_types" = c('ManipulatingObject'),
                                     "baby_visual_types" = c('LookAtMomActivity'),
                                     "missing_types" = c('CantTellHolding',
                                                         'ActivityNotVisible',
                                                         'CantTellLooking')),
                 missing_threshold = 0.1)
```

The variables are described below:

- `f_loc` : Location of the file to be analyzed
- `plot_all` : Logical variable to plot diagnostic plots for this individual. Defaults to `FALSE`
- `plots_to_file` : Logical variable which plots diagnostic plots to a file. Currently unused.
- `tactile_padding` : Padding to be applied to the "event" types of the Observer software. Padding is right-adjusted and Defaults to 1 second.
- `auditory_padding` : Padding to be applied to the "event" types of the Observer software. Padding is right-adjusted and Defaults to 1 second.
- `behavior_types` : List specifying the "auditory", "tactile", and "visual" behaviors to subset on. The "missing_types" are description of tags that define missing data.

- `missing_threshold` : Proportion value that defines what the threshold of missingness is to include the file or not. Set to 0.1 reflecting 10% of missingness is acceptable based upon the tags in `behavior_types`.

Running on a Directory of Files

To expedite processing of many files an additionally function is provided to analyze an entire directory of Excel files. The function `ber_analyze_dir` is similar to the previous function and takes as input `dir_loc`.

```
ber_analyze_dir(dir_loc,
                tactile_padding = 1.0,
                auditory_padding = 1.0,
                behavior_types=list("mom_auditory_types" = c('Vocal'),
                                    "mom_tactile_types" = c('TouchBaby',
                                                            'HoldingBaby'),
                                    "mom_visual_types" = c('ManipulatingObject'),
                                    "baby_visual_types" = c('LookAtMomActivity'),
                                    "missing_types" = c('CantTellHolding',
                                                        'ActivityNotVisible',
                                                        'CantTellLooking')),
                missing_threshold = 0.1)
```

The inputs that are described in the previous section are the same and are passed as input to multiple calls of `ber_analyze_file`.

An Example of How to Estimate Entropy Rate using `ccber`

Consider the following transition matrix of a first-order Markov chain with three states,

```
P = matrix(c(0.2, 0.3, 0.5,
             0.7, 0.1, 0.2,
             0.2, 0.2, 0.6), 3,3, byrow = T)
```

We can simulate from a Markov process with this using the function `SimulateMarkovChain`

```
mc_chain <- SimulateMarkovChain(trans_mat = P, n_sims = 5000)
head(mc_chain, n = 20)
```

```
##  [1] 2 3 1 3 3 2 1 3 2 1 3 1 1 3 1 2 1 3 3 2
```

From this we can calculate a matrix of transition counts

```
tc <- CalcTransitionCounts(mc_chain)
tc
```

```
##      [,1] [,2] [,3]
## [1,]  301  452  754
```

```
## [2,]  735  106  225
## [3,]  472  507 1447
```

And then estimate a transition matrix,

```
tm <- CalcTransitionMatrix(tc)
tm
```

```
##           [,1]       [,2]      [,3]
## [1,] 0.1997346 0.29993364 0.5003318
## [2,] 0.6894934 0.09943715 0.2110694
## [3,] 0.1945589 0.20898599 0.5964551
```

which agrees fairly well with the true `P`. Additionally we can estimate the stationary distribution of the process in a one of two ways. The first way is an empirical estimate from the observed sequence.

```
emp_sm <- CalcEmpiricalStationary(mc_chain, state_space = 1:3)
emp_sm
```

```
##        [,1]   [,2]   [,3]
## [1,] 0.3016 0.2132 0.4852
```

The second way is an eigendecomposition of the observed transition matrix, though the preferred method is through the empirical estimation procedure.

```
eig_sm <- CalcEigenStationary(tm)
eig_sm
```

```
## [1] 0.3015764 0.2130719 0.4853517
```

Using both the stationary distribution estimate and the estimate of the transition matrix, the entropy rate of the process can be estimated using the following commands

```
entrate1 <- CalcMarkovEntropyRate(tm, emp_sm)
entrate1
```

```
## [1] 1.366048
```

```
entrate2 <- CalcMarkovEntropyRate(tm, eig_sm)
entrate2
```

```
## [1] 1.366071
```

Both of these values agree very well with the true entropy rate,

```
true_entropy_rate <- CalcMarkovEntropyRate(P, CalcEigenStationary(P))
true_entropy_rate
```

```
## [1] 1.360979
```

There's a Quicker Way Than That...

If the method of estimation for the stationary distrbution is known, a more simple function is provided to estimate the entropy rate as well.

Both of these values agree very well with the true entropy rate,

```
quicker_estimate <- CalcEntropyRate(mc_chain,
                                    state_space = 1:3,
                                    stat_method = "Empirical")
quicker_estimate
```

```
## [1] 1.366048
```