*Italics means code*

-state- in a feature name can be replaced by all or some sleep states (wake.percent, NREM.percent, REM.percent and sleep.percent). Combined into one feature to reduce reading time (state.percent).

Animal.ID

Identification factor that connects the rat to its data.

It takes the first part of the file name by splitting the name and taking the first element of the list.

Mat.byTotal is a matrix which contains all the information about the files at the end of the script and gets written to \_byTotal.

*for (FileName in FileNames) {*

*names <- strsplit(FileName, "[[:punct:]]")*

*names <- sapply(names, function(x) return(x))*

*id <- names[1]*

*if (id.factor == TRUE & group.factor == TRUE) {*

*mat.byTotal[counter:counter,"Animal.ID"] <- id*

Grouping.Factor

Indicates which group the individual belongs to (Condition or Treatment)

It takes the second part of the file name

*for (FileName in FileNames) {*

*names <- strsplit(FileName, "[[:punct:]]")*

*names <- sapply(names, function(x) return(x))*

*group <- names[2]*

*if (id.factor == TRUE & group.factor == TRUE) {*

*mat.byTotal[counter:counter,"Grouping.Factor"] <- group*

Trial.Number

I added this column to the dataframe and added the values by taking it from the file name depending on if the file name had a additional number (trial number 5)

*if (length(names) == 10) {*

*trial\_number <- (paste(names[8], names[9], sep="\_"))*

*} else{*

*trial\_number <- (names[8])*

*}*

*if (id.factor == TRUE & group.factor == TRUE) {*

*mat.byTotal[counter:counter, 'Trial.Number'] <- trial\_number*

Sex

The gender of the corresponding rat. Is empty.

Cohort

Which cohort it belongs to. Is empty.

Age

The age of the rat. Is empty.

Total.Hours

Total number of hours in the file. Calculated by multiplying the length of the list containing the sleep sequency with the epoch.size (1) and dividing it by 3600 (from seconds to hours).

*mat.byTotal[,'Total.Hours'] <- length(scores)\*epoch.size/3600*

State.Epoch.Count

Amount of epochs that are scored as wake for example in that sleep sequence.

*Wake.Epoch.Count <- sum(scores == 1, na.rm = TRUE)*

*mat.byTotal[counter:counter,'Wake.Epoch.Count'] <- as.integer(Wake.Epoch.Count)*

State.Seconds

Number of seconds in that sleep state. The epoch count is 1, so it is the same as the wake count since it multiplies the epoch.count with the epoch size (1)

*Wake.Seconds <- Wake.Epoch.Count\*epoch.size*

*mat.byTotal[counter:counter,'Wake.Seconds'] <- as.integer(Wake.Seconds)*

State.Onset

Epoch indicating the first bout of that sleep state. Is calculated by taking the first result from the column containing the start epoch of each bout.

*NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= NREM.cutoff & y$values ==2,]*

*})*

*mat.byTotal[counter:counter,'NREMS.Onset'] <- NREM.Lat$starts[1] #NREM Onset*

*RLE example*

*lengths: int [1:30] 310 17 31 134 89 20 11 4 74 38 ...*

*values : num [1:30] 1 2 1 2 1 2 1 2 1 2 ...*

lengths = length bout

values = the sleep state

State.Offset

Last epoch of the last bout of the sleep state. Is calculated by taking the last element of the column containing the ends of the bouts.

*NREM.Offset <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths))*

*y[y$lengths >= NREM.cutoff & y$values ==2,]*

*})*

*NREM.Offset <- tail(NREM.Offset$ends, n = 1L)*

*mat.byTotal[counter:counter,'NREMS.Offset'] <- NREM.Offset #NREM Offset*

Latency.to.state

Amount of time in minutes before the first bout of the state. Calculated by subtracting one from the first epoch of the initial bout.

*NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= NREM.cutoff & y$values ==2,]*

*})*

*mat.byTotal[counter:counter,'Latency.to.NREMS'] <- ((NREM.Lat$starts[1]-1)\*epoch.size)/60 #Latency to NREM*

state.Percent

Percntage of the sleep sequence that is 1 (wake), 2 (NREM), 3 (REMS) or 4 (sleep) . Calculated by averaging a logical vector that indicates whether each state is Wake (True) or not (False).

*Wake.Percent <- mean(scores == 1, na.rm=TRUE)*

*mat.byTotal[counter:counter,'Wake.Percent'] <- as.numeric(Wake.Percent)*

Arousal.Count

Number of instances of wake of any length. Each sequenctial instance of wake is a count even if length is 1. Calculated by summing the number of values equal to 1 in a Run Length Encoding (RLE) object.

*Arousal.Count <- with(rle(scores), {*

*sum(values == 1)*

*})*

*mat.byTotal[counter:counter,'Arousal.Count'] <- as.integer(Arousal.Count)*

Microarousal.Count

Number of wake bouts less than the wake cutoff value. Calculated by taking the sum of Wake bouts that have a length lower than the Wake.Cutoff

*Microarousal.Count <- with(rle(scores), {*

*sum(lengths < Wake.cutoff & values == 1)*

*})*

*}*

*mat.byTotal[counter:counter,'Microarousal.Count'] <- as.integer(Microarousal.Count)*

state.Bout.Count

Number of state bouts that are longer than the length cutoff for that sleep state

*Wake.Bout.Count <- with(rle(scores), {*

*sum(lengths >= Wake.cutoff & values == 1)*

*})*

*mat.byTotal[counter:counter,'Wake.Bout.Count'] <- as.integer(Wake.Bout.Count)*

Avg.state.Bout.Duration

Average length of time of a state bout. Calculated by taking the mean of the bout lengths that have the corresponding number as value

*Wake.Bout.Duration <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values)*

*y <- y[y$lengths >= Wake.cutoff & y$values ==1,]*

*mean(y$lengths, na.rm = TRUE)*

*})*

*mat.byTotal[counter:counter,'Avg.Wake.Bout.Duration'] <- (as.numeric((Wake.Bout.Duration\*epoch.size)))/60*

state.state.Transtion.Count

The number of transition made from an state (e.g. wake) sequence of any length to another state sequence (e.g. NREM) of any length. A dataframe is created with the original sleep sequence and another sequence where each state is shifted up by one using data.table::shift. It then checks if the original sequence has a sleep state of 1 and the shifted sequence has a sleep state of 2 in the same row. If so, it indicates that NREM sleep follows directly after the wake state.

Example of a shifted dataframe / sleep sequence:

Original

1, 2, 3, 4, 5

shifted

2 3 4 5 NA

*WN <- with(rle(scores), {*

*y <- data.frame(values = values, lead = data.table::shift(values, type ="lead"))*

*y[y$values == 1 & y$lead == 2,]*

*})*

*mat.byTotal[counter:counter,'Wake.NREMS.Transition.Count'] <- as.integer(length(WN$values[!is.na(WN$values)]))*

Propensity.to.enter.state

Average duration from the beginning of a wake bout to a NREMS, REMS, or sleep bout in minutes. Calculating by making another shifted dataframe and comparing taking the rows where the original sequence is wake and the shifted is NREM. Then it removes the rows that have empty column(s) with complete.cases. Lastly it calculates the duration for each entry by subtracting the starts value (Wake bout) from the ends value (NREM, REMS or Sleep bout) (plus 1 to include both the start and end times).

*Wake.NREMS.dat <- with(rle(scores),{*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= Propensity.cutoff & (y$values == 1 | y$values == 2),]*

*})*

*Wake.NREMS.dat['lead'] <- c(data.table::shift(Wake.NREMS.dat$values, type = "lead"))*

*Wake.NREMS.dat2 <- Wake.NREMS.dat[Wake.NREMS.dat$values == 1 & Wake.NREMS.dat$lead == 2,]*

*Wake.NREMS.dat2 <- Wake.NREMS.dat2 [complete.cases(Wake.NREMS.dat2 ), ]*

*Propensity.to.enter.NREMS <- (mean((Wake.NREMS.dat2$ends+1) - Wake.NREMS.dat2$starts))\*epoch.size*

*mat.byTotal[counter:counter,'Propensity.to.enter.NREMS'] <- as.numeric(Propensity.to.enter.NREMS)/60*

Propensity.to.Wake.from.state

Average duration from the beginning of a NREMS, REMS or Sleep bout to a wake bout in minutes. Calculated the same way as Propensity.to.enter.state, but with the start value being the NREMS bout and the end value being the Wake value

*NREMS.Wake.dat <- Wake.NREMS.dat[Wake.NREMS.dat$values == 2 & Wake.NREMS.dat$lead == 1,]*

*NREMS.Wake.dat <- NREMS.Wake.dat[complete.cases(NREMS.Wake.dat), ]*

*Propensity.to.Wake.from.NREMS <- (mean((NREMS.Wake.dat$ends+1) - NREMS.Wake.dat$starts))\*epoch.size*

*mat.byTotal[counter:counter,'Propensity.to.Wake.from.NREMS'] <- as.numeric(Propensity.to.Wake.from.NREMS)/60*

Sleep.Fragmentation.Index

Number of arousals (Number of instances of wake of any length) per minute of sleep. Calculated with Arousal Count / (Total Time in Sleep / 60)

Sleep.Fragmentation.Index <- as.integer(mat.byTotal[,'Arousal.Count']) / ((as.integer(mat.byTotal[,'Total.Sleep.Time']))/60)

Total.Sleep.Time

Total amount of time spent in NREMS + REMS.

*Total.Sleep.Time <- as.integer(mat.byTotal[,'NREMS.Seconds']) + as.integer(mat.byTotal[,'REMS.Seconds'])*

*#NREM.Seconds + REM.Seconds*

*mat.byTotal[,'Total.Sleep.Time'] <- as.integer(Total.Sleep.Time)*

Sleep.Efficiency

Total Sleep Time/Total Recording Time

*Sleep.Efficiency <- as.integer(mat.byTotal[,'Total.Sleep.Time']) / as.integer(length(scores)\*epoch.size)*

WASO

Total amount of wake after sleep onset (first sleep bout). Calculated with Total Recording Time - Total Sleep Time (aka Total Time Awake) - NREMS Latency (Amount of time before the first bout of NREMS).

*WASO <- length(scores)\*epoch.size - as.integer(mat.byTotal[,'Total.Sleep.Time']) - (as.integer(mat.byTotal[,'Latency.to.NREMS']))\*60*

*mat.byTotal[,'WASO'] <- as.numeric(WASO)/60*

WASF

Total amount of wake after sleep offset (last bout of sleep, NREMS or REMS). Calculated with Total Recording Time - Sleep Offset.

*WASF <- as.integer(length(scores)\*epoch.size) - (as.integer(mat.byTotal[,'Sleep.Offset'])\*epoch.size)*

*#Total.Recording.Time - Sleep.Offset*

*mat.byTotal[,'WASF'] <- WASF/60*

Latency.to.Wake.from.state

The time between the first NREM, REM or sleep bout and the next wake bout is calculated by creating a dataframe with the length and values of each bout, the lengths and values of the previous bout (shifted by one position), and the cumulative sum of the lengths for the end and start positions. The dataframe includes only rows where the current bout's length is at least the wake.cutoff, the previous bout's length is at least the state.cutoff, the current bout is wake, and the previous bout is either NREM, REM or sleep.

*Wake.from.NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, value.lag = data.table::shift(values), lengths.lag = data.table::shift(lengths),*

*ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= Wake.cutoff & y$lengths.lag >= NREM.cutoff & y$values ==1 & y$value.lag == 2,]*

*})*

*mat.byTotal[counter:counter,'Latency.to.Wake.from.NREMS'] <- ((Wake.from.NREM.Lat$starts[1]-1)\*epoch.size)/60 #Latency to Wake from NREM*

state.Onset.Bout.Length

Length of first bout of NREMS, REMS or sleep. Calculated by making a dataframe containing the lengths, values , start and end of each bout. The dataframe only contains the rows whose NREM bout lengths are larger or equal to the state.cutoff and whose value is the corresponding number. It then takes the length of the first bout in the dataframe.

*NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= NREM.cutoff & y$values ==2,]*

*})*

*mat.byTotal[counter:counter,'Latency.to.NREMS'] <- ((NREM.Lat$starts[1]-1)\*epoch.size)/60 #Latency to NREM*

state.Offset.Bout.Length

Length of last NREMS, REMS or sleep bout. Calculated by making a dataframe with the length, value and end of each bout, whose value is the corresponding number and length is greater or equal to the state.cutoff. It then obtains the last value of the ends column.

*NREM.Offset <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths))*

*y[y$lengths >= NREM.cutoff & y$values ==2,]*

*})*

*NREM.Offset <- tail(NREM.Offset$ends, n = 1L)*

*mat.byTotal[counter:counter,'NREMS.Offset.Bout.Length'] <- (NREM.Offset.Lengths\*epoch.size)/60 #NREM Offset Bout Length*

Post-state.Wake.Onset

Epoch indicating the first bout of wake after the first bout of NREMS, REMS or sleep. Calculated by creating a dataframe with the start epoch of each bout and the previous bout (shifted by one position). The dataframe only includes the rows where the length of the current is greater or equal to wake.cutoff, the length of the previous bout was greater or equal to state.cutoff, the value of the current bout equals wake and the previous bout equals to the number corresponded to the desired state. It then takes the first start epoch of the dataframe.

*Wake.from.NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, value.lag = data.table::shift(values), lengths.lag = data.table::shift(lengths),*

*ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= Wake.cutoff & y$lengths.lag >= NREM.cutoff & y$values ==1 & y$value.lag == 2,]*

*})*

*mat.byTotal[counter:counter,'Post-NREMS.Wake.Onset'] <- Wake.from.NREM.Lat$starts[1] #Post-NREMS Wake Onset*

Post-state.Wake.Onset.Bout.Length

Length of the first bout of wake following the first NREMS, REMS or sleep bout. Calculated by creating a dataframe with the length and values of each bout and the previous bout (shifted by one position). The dataframe only includes the rows where the length of the current is greater or equal to wake.cutoff, the length of the previous bout was greater or equal to state.cutoff, the value of the current bout equals wake and the previous bout equals the corresponding number. It then takes the first length of the dataframe.

*Wake.from.NREM.Lat <- with(rle(scores), {*

*y <- data.frame(lengths = lengths, values = values, value.lag = data.table::shift(values), lengths.lag = data.table::shift(lengths), ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= Wake.cutoff & y$lengths.lag >= NREM.cutoff & y$values ==1 & y$value.lag == 2,]*

*})*

*mat.byTotal[counter:counter,'Post-NREMS.Wake.Onset.Bout.Length'] <- (Wake.from.NREM.Lat$lengths[1]\*epoch.size)/60 #Post-NREMS Wake Onset Bout Length*

Avg.state.Interbout.Interval

The average duration between bouts of the same state is calculated by creating a dataframe that includes the start times of each wake, NREM, or REM bout longer than the cutoff and matching the respective sleep state. A shifted dataframe is then created to calculate the intervals between the start of each bout and the end of the previous bout, and the mean of these intervals is computed.

*Wake.dat <- with(rle(scores),{*

*y <- data.frame(lengths = lengths, values = values, ends = cumsum(lengths), starts = cumsum(lengths) - lengths + 1)*

*y[y$lengths >= Wake.cutoff & y$values ==1,]*

*})*

*Wake.Interbout.Intervals <- data.frame(IBI = data.table::shift(Wake.dat$starts, type = "lead") - Wake.dat$ends - 1)*

*Wake.Interbout.Interval.Average <- data.frame(IBIA = mean(Wake.Interbout.Intervals))*

*mat.byTotal[counter:counter,'Avg.Wake.Interbout.Interval'] <- as.numeric((Wake.Interbout.Interval.Average$IBIA[1]\*epoch.size)/60)*

W/N/RBC (Wake or NREMS or REMS Bout Count)

Wake or NREMS or REMS bout count of set length determined by epoch size (1). Calculated by calculating all the multiplications and adding them to a list. Next, create a dataframe with the lengths of the bouts and calculate the sum of rows where the value matches the sleep state and the length meets one of the specified conditions.

|  |  |  |
| --- | --- | --- |
| Group | condition | Our dataset |
| 1 | Bouts == epoch size | Number of bouts with a length of 1 |
| 2 | Bouts >= epoch size x 2 & < epoch size x 4 | Number of bouts with a length of 2 and 3 |
| 3 | Bouts >= epoch size x 4 & < epoch size x 8 | Number of bouts with a length of 4 up until 7 |
| 4 | Bouts >= epoch size x 8 & < epoch size x 16 | Number of bouts with a length of 8 up until 15 |
| 5 | Bouts >= epoch size x 16 & < epoch size x 32 | Number of bouts with a length of 16 up until 31 |
| 6 | Bouts >= epoch size x 32 & < epoch size x 64 | Number of bouts with a length of 32 up until 63 |
| 7 | Bouts >= epoch size x 64 & < epoch size x 128 | Number of bouts with a length of 64 up until 127 |
| 8 | Bouts >= epoch size x 128 & < epoch size x 256 | Number of bouts with a length of 128 up until 255 |
| 9 | Bouts >= epoch size\*256 | Number of bouts with a length equal or greater than 256 |

*a = epoch.size*

*for(i in 1:9){*

*a[i+1] = 2\*a[i]*

*}*

*a <- a/epoch.size*

*WBC.group1 <- with(rle(scores), {*

*sum(values == 1 & lengths <= a[1])*

*})*

*mat.byTotal[counter:counter, 'WBC.group1'] <- as.integer(WBC.group1)*