Package 'philentropy'

May 23, 2018

Type Package

Title Similarity and Distance Quantification Between Probability Functions

Version 0.2.0

Date 2018-05-22

Maintainer Hajk-Georg Drost <hgd23@cam.ac.uk>

Description Computes 46 optimized distance and similarity measures for comparing probability functions. These comparisons between probability functions have their foundations in a broad range of scientific disciplines from mathematics to ecology. The aim of this package is to provide a core framework for clustering, classification, statistical inference, goodness-of-fit, non-parametric statistics, information theory, and machine learning tasks that are based on comparing univariate or multivariate probability functions.

Depends R (>= 3.1.2)

Imports Rcpp, dplyr, KernSmooth

License GPL-2

LazyData true

LinkingTo Rcpp

URL https://github.com/HajkD/philentropy

Suggests testthat, knitr

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Author Hajk-Georg Drost [aut, cre] (<https://orcid.org/0000-0002-1567-306X>)

Repository CRAN

Date/Publication 2018-05-22 19:06:08

R topics documented:

binned.kernel.est . . . 2
CE . . . 3
dist.diversity . . . 4
distance . . . 5
estimate.probability . . . 8
getDistMethods . . . 9
gJSD . . . 10
H . . . 11
JE . . . 12
JSD . . . 13
KL . . . 14
lin.cor . . . 16
MI . . . 17

Index **19**

`binned.kernel.est` *Kernel Density Estimation*

Description

This function implements an interface to the kernel density estimation functions provided by the **KernSmooth** package.

Usage

```
binned.kernel.est(data, kernel = "normal", bandwidth = NULL,
  canonical = FALSE, scalest = "minim", level = 2L, gridsize = 401L,
  range.data = range(data), truncate = TRUE)
```

Arguments

`data`	a numeric vector containing the sample on which the kernel density estimate is to be constructed.
`kernel`	character string specifying the smoothing kernel
`bandwidth`	the kernel bandwidth smoothing parameter.
`canonical`	a logical value indicating whether canonically scaled kernels should be used
`scalest`	estimate of scale. • `"stdev"` - standard deviation is used. • `"iqr"` - inter-quartile range divided by 1.349 is used. • `"minim"` - minimum of `"stdev"` and `"iqr"` is used.
`level`	number of levels of functional estimation used in the plug-in rule.
`gridsize`	the number of equally-spaced points over which binning is performed to obtain kernel functional approximation.

range.data	vector containing the minimum and maximum values of data at which to compute the estimate. The default is the minimum and maximum data values.
truncate	logical value indicating whether data with x values outside the range specified by range.data should be ignored.

Author(s)

Hajk-Georg Drost

References

Matt Wand (2015). KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995). R package version 2.23-14.

Henry Deng and Hadley Wickham (2011). Density estimation in R. http://vita.had.co.nz/papers/density-estimation.pdf.

CE *Shannon's Conditional-Entropy $H(X|Y)$*

Description

Compute Shannon's Conditional-Entropy based on the chain rule $H(X|Y) = H(X,Y) - H(Y)$ based on a given joint-probability vector $P(X,Y)$ and probability vector $P(Y)$.

Usage

```
CE(xy, y, unit = "log2")
```

Arguments

xy	a numeric joint-probability vector $P(X,Y)$ for which Shannon's Joint-Entropy $H(X,Y)$ shall be computed.
y	a numeric probability vector $P(Y)$ for which Shannon's Entropy $H(Y)$ (as part of the chain rule) shall be computed. It is important to note that this probability vector must be the probability distribution of random variable Y (P(Y) for which H(Y) is computed).
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

Details

This function might be useful to fastly compute Shannon's Conditional-Entropy for any given joint-probability vector and probability vector.

Value

Shannon's Conditional-Entropy in bit.

Note

Note that the probability vector P(Y) must be the probability distribution of random variable Y (P(Y) for which H(Y) is computed) and furthermore used for the chain rule computation of $H(X|Y) = H(X,Y) - H(Y)$.

Author(s)

Hajk-Georg Drost

References

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

See Also

H, JE

Examples

```
CE(1:10/sum(1:10),1:10/sum(1:10))
```

`dist.diversity` *Distance Diversity between Probability Density Functions*

Description

This function computes all distance values between two probability density functions that are available in `getDistMethods` and returns a vector storing the corresponding distance measures. This vector is *named distance diversity vector*.

Usage

```
dist.diversity(x, p = 2)
```

Arguments

`x`	a numeric `data.frame` or `matrix` (storing probability vectors) or a numeric `data.frame` or `matrix` storing counts (if `est.prob` is specified).
`p`	power of the Minkowski distance.

Author(s)

Hajk-Georg Drost

Examples

```
dist.diversity(rbind(1:10/sum(1:10), 20:29/sum(20:29)))
```

`distance` *Distances between Probability Density Functions*

Description

This functions computes the distance/dissimilarity between two probability density functions.

Usage

```
distance(x, method = "euclidean", p = NULL, test.na = TRUE,
  unit = "log", est.prob = NULL)
```

Arguments

`x`	a numeric `data.frame` or `matrix` (storing probability vectors) or a numeric `data.frame` or `matrix` storing counts (if `est.prob` is specified).
`method`	a character string indicating whether the distance measure that should be computed.
`p`	power of the Minkowski distance.
`test.na`	a boolean value indicating whether input vectors should be tested for `NA` values. Faster computations if `test.na = FALSE`.
`unit`	a character string specifying the logarithm unit that should be used to compute distances that depend on log computations.
`est.prob`	method to estimate probabilities from a count vector. Default: `est.prob = NULL`.

Details

Here a distance is defined as a quantitative degree of how far two mathamatical objects are apart from eachother (Cha, 2007).

This function implements the following distance/similarity measures to quantify the distance between probability density functions:

- L_p Minkowski family
  - Euclidean : $d = sqrt(\sum |P_i - Q_i|^2)$
  - Manhattan : $d = \sum |P_i - Q_i|$
  - Minkowski : $d = (\sum |P_i - Q_i|^p)^1/p$
  - Chebyshev : $d = max|P_i - Q_i|$
- L_1 family
  - Sorensen : $d = \sum |P_i - Q_i| / \sum (P_i + Q_i)$

  - Gower : $d = 1/d * \sum |P_i - Q_i|$
  - Soergel : $d = \sum |P_i - Q_i| / \sum max(P_i, Q_i)$
  - Kulczynski d : $d = \sum |P_i - Q_i| / \sum min(P_i, Q_i)$
  - Canberra : $d = \sum |P_i - Q_i| / (P_i + Q_i)$
  - Lorentzian : $d = \sum ln(1 + |P_i - Q_i|)$
- Intersection family
  - Intersection : $s = \sum min(P_i, Q_i)$
  - Non-Intersection : $d = 1 - \sum min(P_i, Q_i)$
  - Wave Hedges : $d = \sum |P_i - Q_i| / max(P_i, Q_i)$
  - Czekanowski : $d = \sum |P_i - Q_i| / \sum |P_i + Q_i|$
  - Motyka : $d = \sum min(P_i, Q_i) / (P_i + Q_i)$
  - Kulczynski s : $d = 1 / \sum |P_i - Q_i| / \sum min(P_i, Q_i)$
  - Tanimoto : $d = \sum (max(P_i, Q_i) - min(P_i, Q_i)) / \sum max(P_i, Q_i)$; equivalent to Soergel
  - Ruzicka : $s = \sum min(P_i, Q_i) / \sum max(P_i, Q_i)$; equivalent to 1 - Tanimoto = 1 - Soergel
- Inner Product family
  - Inner Product : $s = \sum P_i * Q_i$
  - Harmonic mean : $s = 2 * \sum (P_i * Q_i) / (P_i + Q_i)$
  - Cosine : $s = \sum (P_i * Q_i) / sqrt(\sum P_i^2) * sqrt(\sum Q_i^2)$
  - Kumar-Hassebrook (PCE) : $s = \sum (P_i * Q_i) / (\sum P_i^2 + \sum Q_i^2 - \sum (P_i * Q_i))$
  - Jaccard : $d = 1 - \sum (P_i * Q_i) / (\sum P_i^2 + \sum Q_i^2 - \sum (P_i * Q_i))$; equivalent to 1 - Kumar-Hassebrook
  - Dice : $d = \sum (P_i - Q_i)^2 / (\sum P_i^2 + \sum Q_i^2)$
- Squared-chord family
  - Fidelity : $s = \sum sqrt(P_i * Q_i)$
  - Bhattacharyya : $d = -ln \sum sqrt(P_i * Q_i)$
  - Hellinger : $d = 2 * sqrt(1 - \sum sqrt(P_i * Q_i))$
  - Matusita : $d = sqrt(2 - 2 * \sum sqrt(P_i * Q_i))$
  - Squared-chord : $d = \sum (sqrt(P_i) - sqrt(Q_i))^2$
- Squared L_2 family (X^2 squared family)
  - Squared Euclidean : $d = \sum (P_i - Q_i)^2$
  - Pearson X^2 : $d = \sum ((P_i - Q_i)^2 / Q_i)$
  - Neyman X^2 : $d = \sum ((P_i - Q_i)^2 / P_i)$
  - Squared X^2 : $d = \sum ((P_i - Q_i)^2 / (P_i + Q_i))$
  - Probabilistic Symmetric X^2 : $d = 2 * \sum ((P_i - Q_i)^2 / (P_i + Q_i))$
  - Divergence : X^2 : $d = 2 * \sum ((P_i - Q_i)^2 / (P_i + Q_i)^2)$
  - Clark : $d = sqrt(\sum (|P_i - Q_i| / (P_i + Q_i))^2)$
  - Additive Symmetric X^2 : $d = \sum (((P_i - Q_i)^2 * (P_i + Q_i)) / (P_i * Q_i))$
- Shannon's entropy family
  - Kullback-Leibler : $d = \sum P_i * log(P_i / Q_i)$

  - Jeffreys : $d = \sum(P_i - Q_i) * log(P_i/Q_i)$
  - K divergence : $d = \sum P_i * log(2 * P_i/P_i + Q_i)$
  - Topsoe : $d = \sum(P_i * log(2 * P_i/P_i + Q_i)) + (Q_i * log(2 * Q_i/P_i + Q_i))$
  - Jensen-Shannon : $d = 0.5 * (\sum P_i * log(2 * P_i/P_i + Q_i) + \sum Q_i * log(2 * Q_i/P_i + Q_i))$
  - Jensen difference : $d = \sum((P_i * log(P_i) + Q_i * log(Q_i)/2) - (P_i + Q_i/2) * log(P_i + Q_i/2))$
- Combinations
  - Taneja : $d = \sum(P_i + Q_i/2) * log(P_i + Q_i/(2 * sqrt(P_i * Q_i)))$
  - Kumar-Johnson : $d = \sum(P_i^2 - Q_i^2)^2/2 * (P_i * Q_i)^1.5$
  - Avg(L_1, L_n) : $d = \sum |P_i - Q_i| + max|P_i - Q_i|/2$

  In cases where `x` specifies a count matrix, the argument `est.prob` can be selected to first estimate probability vectors from input count vectors and second compute the corresponding distance measure based on the estimated probability vectors.

  The following probability estimation methods are implemented in this function:

  - `est.prob = "empirical"` : relative frequencies of counts.

Value

The following results are returned depending on the dimension of `x`:

- in case `nrow(x)` = 2 : a single distance value.
- in case `nrow(x)` > 2 : a distance `matrix` storing distance values for all pairwise probability vector comparisons.

Note

According to the reference in some distance measure computations invalid computations can occur when dealing with 0 probabilities.

In these cases the convention is treated as follows:

- division by zero - case `0/0`: when the divisor and dividend become zero, `0/0` is treated as `0`.
- division by zero - case `n/0`: when only the divisor becomes `0`, the corresponsning `0` is replaced by a small $\epsilon = 0.00001$.
- log of zero - case `0 * log(0)`: is treated as `0`.
- log of zero - case `log(0)`: zero is replaced by a small $\epsilon = 0.00001$.

Author(s)

Hajk-Georg Drost

References

Sung-Hyuk Cha. (2007). *Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions*. International Journal of Mathematical Models and Methods in Applied Sciences 4: 1.

See Also

getDistMethods, estimate.probability, dist.diversity

Examples

```
# Simple Examples

# receive a list of implemented probability distance measures
getDistMethods()

## compute the euclidean distance between two probability vectors
distance(rbind(1:10/sum(1:10), 20:29/sum(20:29)), method = "euclidean")

## compute the euclidean distance between all pairwise comparisons of probability vectors
ProbMatrix <- rbind(1:10/sum(1:10), 20:29/sum(20:29),30:39/sum(30:39))
distance(ProbMatrix, method = "euclidean")

# compute distance matrix without testing for NA values in the input matrix
distance(ProbMatrix, method = "euclidean", test.na = FALSE)

# Specialized Examples

CountMatrix <- rbind(1:10, 20:29, 30:39)

## estimate probabilities from a count matrix
distance(CountMatrix, method = "euclidean", est.prob = "empirical")

## compute the euclidean distance for count data
## NOTE: some distance measures are only defined for probability values,
distance(CountMatrix, method = "euclidean")

## compute the Kullback-Leibler Divergence with different logarithm bases:
### case: unit = log (Default)
distance(ProbMatrix, method = "kullback-leibler", unit = "log")

### case: unit = log2
distance(ProbMatrix, method = "kullback-leibler", unit = "log2")

### case: unit = log10
distance(ProbMatrix, method = "kullback-leibler", unit = "log10")
```

---

estimate.probability *Estimate Probability Vectors From Count Vectors*

Description

This function takes a numeric count vector and returns estimated probabilities of the corresponding counts.

The following probability estimation methods are implemented in this function:

- empirical :
-
-

Usage

```
estimate.probability(x, method = "empirical")
```

Arguments

x	a numeric vctor storing count values.
method	a character string specifying the estimation method tht should be used to estimate probabilities from input counts.

Value

a numeric probability vector.

Author(s)

Hajk-Georg Drost

Examples

```
# generate a count vector
x <- runif(100)

# generate a probability vector from corresponding counts
# method = "empirical"
x.prob <- estimate.probability(x, method = "empirical")
```

---

getDistMethods *Get method names for* distance

Description

This function returns the names of the methods that can be applied to compute distances between probability density functions using the distance function.

Usage

```
getDistMethods()
```

Author(s)

Hajk-Georg Drost

Examples

```
getDistMethods()
```

---

gJSD *Generalized Jensen-Shannon Divergence*

Description

This function computes the Generalized Jensen-Shannon Divergence of a probability matrix.

Usage

```
gJSD(x, unit = "log2", weights = NULL)
```

Arguments

x	a probability matrix.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
weights	a numeric vector specifying the weights for each distribution in x. Default: weights = NULL; in this case all distributions are weighted equally.

Details

Function to compute the Generalized Jensen-Shannon Divergence

Value

The Jensen-Shannon divergence between all possible combinations of comparisons.

Author(s)

Hajk-Georg Drost

See Also

KL, H, JSD, CE, JE

Examples

```
Prob <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the Generalized JSD comparing the PS probability matrix
gJSD(Prob)
```

H *Shannon's Entropy $H(X)$*

Description

Compute the Shannon's Entropy $H(X) = -\sum P(X) * log2(P(X))$ based on a given probability vector $P(X)$.

Usage

```
H(x, unit = "log2")
```

Arguments

x	a numeric probability vector $P(X)$ for which Shannon's Entropy $H(X)$ shall be computed.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

Details

This function might be useful to fastly compute Shannon's Entropy for any given probability vector.

Value

a numeric value representing Shannon's Entropy in bit.

Author(s)

Hajk-Georg Drost

References

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

See Also

JE, CE, KL, JSD, gJSD

Examples

```
H(1:10/sum(1:10))
```

JE *Shannon's Joint-Entropy $H(X,Y)$*

Description

This funciton computes Shannon's Joint-Entropy $H(X,Y) = -\sum\sum P(X,Y) * log2(P(X,Y))$ based on a given joint-probability vector $P(X,Y)$.

Usage

```
JE(x, unit = "log2")
```

Arguments

x	a numeric joint-probability vector $P(X,Y)$ for which Shannon's Joint-Entropy $H(X,Y)$ shall be computed.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

Value

a numeric value representing Shannon's Joint-Entropy in bit.

Author(s)

Hajk-Georg Drost

References

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

See Also

H, CE, KL, JSD, gJSD, distance

Examples

```
JE(1:100/sum(1:100))
```

JSD *Jensen-Shannon Divergence*

Description

This function computes a distance matrix or distance value based on the Jensen-Shannon Divergence with equal weights.

Usage

```
JSD(x, test.na = TRUE, unit = "log2", est.prob = NULL)
```

Arguments

x	a numeric data.frame or matrix (storing probability vectors) or a numeric data.frame or matrix storing counts (if est.prob = TRUE). See distance for details.
test.na	a boolean value specifying whether input vectors shall be tested for NA values.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
est.prob	method to estimate probabilities from a count vector. Default: est.prob = NULL.

Details

Function to compute the Jensen-Shannon Divergence JSD(P ‖ Q) between two probability distributions P and Q with equal weights $\pi_1 = \pi_2 = 1/2$.

The Jensen-Shannon Divergence JSD(P ‖ Q) between two probability distributions P and Q is defined as:

$$JSD(P||Q) = 0.5 * (KL(P||R) + KL(Q||R))$$

where $R = 0.5 * (P + Q)$ denotes the mid-point of the probability vectors P and Q, and KL(P ‖ R), KL(Q ‖ R) denote the Kullback-Leibler Divergence of P and R, as well as Q and R.

General properties of the Jensen-Shannon Divergence:

- 1) JSD is non-negative.
- 2) JSD is a symmetric measure JSD(P ‖ Q) = JSD(Q ‖ P).
- 3) JSD = 0, if and only if P = Q.

Value

a distance value or matrix based on JSD computations.

Author(s)

Hajk-Georg Drost

References

Lin J. 1991. "Divergence Measures Based on the Shannon Entropy". IEEE Transactions on Information Theory. (33) 1: 145-151.

Endres M. and Schindelin J. E. 2003. "A new metric for probability distributions". IEEE Trans. on Info. Thy. (49) 3: 1858-1860.

See Also

KL, H, CE, gJSD, distance

Examples

```
# Jensen-Shannon Divergence between P and Q
P <- 1:10/sum(1:10)
Q <- 20:29/sum(20:29)
x <- rbind(P,Q)
JSD(x)

# Jensen-Shannon Divergence between P and Q using different log bases
JSD(x, unit = "log2") # Default
JSD(x, unit = "log")
JSD(x, unit = "log10")

# Jensen-Shannon Divergence Divergence between count vectors P.count and Q.count
P.count <- 1:10
Q.count <- 20:29
x.count <- rbind(P.count,Q.count)
JSD(x, est.prob = "empirical")

# Example: Distance Matrix using JSD-Distance

Prob <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the KL matrix of a given probability matrix
JSDMatrix <- JSD(Prob)

# plot a heatmap of the corresponding JSD matrix
heatmap(JSDMatrix)
```

KL *Kullback-Leibler Divergence*

Description

This function computes the Kullback-Leibler divergence of two probability distributions P and Q.

Usage

```
KL(x, test.na = TRUE, unit = "log2", est.prob = NULL)
```

Arguments

x	a numeric data.frame or matrix (storing probability vectors) or a numeric data.frame or matrix storing counts (if est.prob = TRUE). See distance for details.
test.na	a boolean value indicating whether input vectors should be tested for NA values.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
est.prob	method to estimate probabilities from a count vector. Default: est.prob = NULL.

Details

$$KL(P||Q) = \sum P(P) * log2(P(P)/P(Q)) = H(P,Q) - H(P)$$

where H(P,Q) denotes the joint entropy of the probability distributions P and Q and H(P) denotes the entropy of probability distribution P. In case P = Q then KL(P,Q) = 0 and in case P != Q then KL(P,Q) > 0.

The KL divergence is a non-symmetric measure of the directed divergence between two probability distributions P and Q. It only fulfills the *positivity* property of a *distance metric*.

Because of the relation KL(P||Q) = H(P,Q) - H(P), the Kullback-Leibler divergence of two probability distributions P and Q is also named *Cross Entropy* of two probability distributions P and Q.

Value

The Kullback-Leibler divergence of probability vectors.

Author(s)

Hajk-Georg Drost

References

Cover Thomas M. and Thomas Joy A. 2006. Elements of Information Theory. *John Wiley & Sons*.

See Also

H, CE, JSD, gJSD, distance

Examples

```
# Kulback-Leibler Divergence between P and Q
P <- 1:10/sum(1:10)
Q <- 20:29/sum(20:29)
x <- rbind(P,Q)
KL(x)

# Kulback-Leibler Divergence between P and Q using different log bases
KL(x, unit = "log2") # Default
KL(x, unit = "log")
KL(x, unit = "log10")

# Kulback-Leibler Divergence between count vectors P.count and Q.count
P.count <- 1:10
Q.count <- 20:29
x.count <- rbind(P.count,Q.count)
KL(x, est.prob = "empirical")

# Example: Distance Matrix using KL-Distance

Prob <- cbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the KL matrix of a given probability matrix
KLMatrix <- KL(Prob)

# plot a heatmap of the corresponding KL matrix
heatmap(KLMatrix)
```

---

lin.cor *Linear Correlation*

Description

This function computed the linear correlation between two vectors or a correlation matrix for an input matrix.

The following methods to compute linear correlations are implemented in this function:

Usage

```
lin.cor(x, y = NULL, method = "pearson", test.na = FALSE)
```

Arguments

x	a numeric vector, matrix, or data.frame.
y	a numeric vector that should be correlated with x.
method	the method to compute the linear correlation between x and y.
test.na	a boolean value indicating whether input data should be checked for NA values.

Details

- `method = "pearson"` : Pearson's correlation coefficient (centred).
- `method = "pearson2"` : Pearson's uncentred correlation coefficient.
- `method = "sq_pearson"` . Squared Pearson's correlation coefficient.
- `method = "kendall"` : Kendall's correlation coefficient.
- `method = "spearman"` : Spearman's correlation coefficient.

Further Details:

- *Pearson's correlation coefficient (centred)* :

Author(s)

Hajk-Georg Drost

`MI` *Shannon's Mutual Information* $I(X, Y)$

Description

Compute Shannon's Mutual Information based on the identity $I(X,Y) = H(X) + H(Y) - H(X,Y)$ based on a given joint-probability vector $P(X,Y)$ and probability vectors $P(X)$ and $P(Y)$.

Usage

```
MI(x, y, xy, unit = "log2")
```

Arguments

`x`	a numeric probability vector $P(X)$.
`y`	a numeric probability vector $P(Y)$.
`xy`	a numeric joint-probability vector $P(X,Y)$.
`unit`	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

Details

This function might be useful to fastly compute Shannon's Mutual Information for any given joint-probability vector and probability vectors.

Value

Shannon's Mutual Information in bit.

Author(s)

Hajk-Georg Drost

References

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

See Also

H, JE, CE

Examples

```
MI( x = 1:10/sum(1:10), y = 20:29/sum(20:29), xy = 1:10/sum(1:10) )
```

Index

binned.kernel.est, 2

CE, 3, *10–12*, *14, 15*, *18*

dist.diversity, 4, *8*
distance, 5, *9*, *12–15*

estimate.probability, *8*, 8

getDistMethods, *4*, *8*, 9
gJSD, 10, *11, 12*, *14, 15*

H, *4*, *10*, 11, *12*, *14, 15*, *18*

JE, *4*, *10, 11*, 12, *18*
JSD, *10–12*, 13, *15*

KL, *10–12*, *14*, 14

lin.cor, 16

MI, 17