# Package: cubar (via r-universe)

December 7, 2024

**Title** Codon Usage Bias Analysis

**Version** 1.1.0

**Description** A suite of functions for rapid and flexible analysis of codon usage bias. It provides in-depth analysis at the codon level, including relative synonymous codon usage (RSCU), tRNA weight calculations, machine learning predictions for optimal or preferred codons, and visualization of codon-anticodon pairing. Additionally, it can calculate various gene- specific codon indices such as codon adaptation index (CAI), effective number of codons (ENC), fraction of optimal codons (Fop), tRNA adaptation index (tAI), mean codon stabilization coefficients (CSCg), and GC contents (GC/GC3s/GC4d). It also supports both standard and non-standard genetic code tables found in NCBI, as well as custom genetic code tables.

**License** MIT + file LICENSE

**URL** https://github.com/mt1022/cubar, https://mt1022.github.io/cubar/

**BugReports** https://github.com/mt1022/cubar/issues

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** bzip2

**Imports** Biostrings (>= 2.60.0), IRanges (>= 2.34.0), data.table (>= 1.14.0), ggplot2 (>= 3.3.5), rlang (>= 0.4.11)

**Depends** R (>= 4.1.0)

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Hong Zhang [aut, cre] (<https://orcid.org/0000-0002-4064-9432>), Mengyue Liu [aut], Bu Zi [aut]

**Maintainer** Hong Zhang <mt1022.dev@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-12-07 07:50:02 UTC

**Config/pak/sysreqs** libssl-dev

# Contents

## aa2codon *amino acids to codons*

### Description

A data.frame of mapping from amino acids to codons

### Usage

```
aa2codon
```

### Format

a data.frame with two columns: amino_acid, and codon.

**amino_acid** amino acid corresponding to the codon
**codon** codon identity

### Source

It is actually the standard genetic code.

### Examples

```
aa2codon
```

## check_cds *Quality control of CDS*

### Description

`check_cds` performs quality control of CDS sequences by filtering some peculiar sequences and optionally remove start or stop codons.

### Usage

```
check_cds(
  seqs,
  codon_table = get_codon_table(),
  min_len = 6,
  check_len = TRUE,
  check_start = TRUE,
  check_stop = TRUE,
  check_istop = TRUE,
  rm_start = TRUE,
  rm_stop = TRUE,
  start_codons = c("ATG")
)
```

### Arguments

| | |
|---|---|
| seqs | input CDS sequences |
| codon_table | codon table matching the genetic code of seqs |
| min_len | minimum CDS length in nt |
| check_len | check whether CDS length is divisible by 3 |
| check_start | check whether CDSs have start codons |
| check_stop | check whether CDSs have stop codons |
| check_istop | check internal stop codons |
| rm_start | whether to remove start codons |
| rm_stop | whether to remove stop codons |
| start_codons | vector of start codons |

### Value

DNAStringSet of filtered (and trimmed) CDS sequences

### Examples

```
# CDS sequence QC for a sample of yeast genes
s <- head(yeast_cds, 10)
print(s)
check_cds(s)
```

---

## codon_diff *Differential codon usage analysis*

---

### Description

codon_diff takes two set of coding sequences and perform differential codon usage analysis.

### Usage

```
codon_diff(seqs1, seqs2, codon_table = get_codon_table())
```

### Arguments

| | |
|---|---|
| seqs1 | DNAStringSet, or an object that can be coerced to a DNAStringSet |
| seqs2 | DNAStringSet, or an object that can be coerced to a DNAStringSet |
| codon_table | a table of genetic code derived from get_codon_table or create_codon_table. |

## Value

a data.table of the differential codon usage analysis. Global tests examine wthether a codon is used differently relative to all the other codons. Family tests examine whether a codon is used differently relative to other codons that encode the same amino acid. Subfamily tests examine whether a codon is used differently relative to other synonymous codons that share the same first two nucleotides. Odds ratio > 1 suggests a codon is used at higher frequency in `seqs1` than in `seqs2`.

## Examples

```
yeast_exp_sorted <- yeast_exp[order(yeast_exp$fpkm),]
seqs1 <- yeast_cds[names(yeast_cds) %in% head(yeast_exp_sorted$gene_id, 1000)]
seqs2 <- yeast_cds[names(yeast_cds) %in% tail(yeast_exp_sorted$gene_id, 1000)]
cudiff <- codon_diff(seqs1, seqs2)
```

---

# codon_optimize *Optimize codons*

---

## Description

`codon_optimize` takes a coding sequence (without stop codon) and replace each codon to the corresponding synonymous optimal codon.

## Usage

```
codon_optimize(
  seq,
  optimal_codons,
  codon_table = get_codon_table(),
  level = "subfam"
)
```

## Arguments

| | |
|---|---|
| `seq` | DNAString, or an object that can be coerced to a DNAString. |
| `optimal_codons` | table optimze codons as generated by `est_optimal_codons`. |
| `codon_table` | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| `level` | "subfam" (default) or "amino_acid". Optimize codon usage at which level. |

## Value

a DNAString of the optimized coding sequence.

## Examples

```
cf_all <- count_codons(yeast_cds)
optimal_codons <- est_optimal_codons(cf_all)
seq <- 'ATGCTACGA'
codon_optimize(seq, optimal_codons)
```

---

`count_codons` *Count occurrences of different codons*

---

## Description

`count_codons` tabulates the occurrences of all the 64 codons in input CDSs

## Usage

```
count_codons(seqs, ...)
```

## Arguments

| | |
|---|---|
| `seqs` | CDS sequences, DNAStringSet. |
| `...` | additional arguments passed to `Biostrings::trinucleotideFrequency`. |

## Value

matrix of codon (column) frequencies of each CDS (row).

## Examples

```
# count codon occurrences
cf_all <- count_codons(yeast_cds)
dim(cf_all)
cf_all[1:5, 1:5]
count_codons(yeast_cds[1])
```

## create_codon_table *create custom codon table from a data frame*

### Description

create_codon_table creates codon table from data frame of aa to codon mapping.

### Usage

```
create_codon_table(aa2codon)
```

### Arguments

| | |
|---|---|
| aa2codon | a data frame with two columns: amino_acid (Ala, Arg, etc.) and codon. |

### Value

a data.table with four columns: aa_code, amino_acid, codon, and subfam.

### Examples

```
head(aa2codon)
create_codon_table(aa2codon = aa2codon)
```

## est_csc *Estimate Codon Stabilization Coefficient*

### Description

get_csc calculate codon occurrence to mRNA stability correlation coefficients (Default to Pearson's).

### Usage

```
est_csc(
  seqs,
  half_life,
  codon_table = get_codon_table(),
  cor_method = "pearson"
)
```

### Arguments

| | |
|---|---|
| seqs | CDS sequences of all protein-coding genes. One for each gene. |
| half_life | data.frame of mRNA half life (gene_id & half_life are column names). |
| codon_table | a table of genetic code derived from get_codon_table or create_codon_table. |
| cor_method | method name passed to 'cor.test' used for calculating correlation coefficients. |

## Value

data.table of optimal codons.

## References

Presnyak V, Alhusaini N, Chen YH, Martin S, Morris N, Kline N, Olson S, Weinberg D, Baker KE, Graveley BR, et al. 2015. Codon optimality is a major determinant of mRNA stability. Cell 160:1111-1124.

## Examples

```
# estimate yeast mRNA CSC
est_csc(yeast_cds, yeast_half_life)
```

---

# est_optimal_codons *Estimate optimal codons*

---

## Description

`est_optimal_codons` determine optimal codon of each codon family with binomial regression. Usage of optimal codons should correlate negatively with enc.

## Usage

```
est_optimal_codons(
  cf,
  codon_table = get_codon_table(),
  level = "subfam",
  gene_score = NULL,
  fdr = 0.001
)
```

## Arguments

| | |
|---|---|
| `cf` | matrix of codon frequencies as calculated by `count_codons()`. |
| `codon_table` | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| `level` | "subfam" (default) or "amino_acid". For which level to determine optimal codons. |
| `gene_score` | a numeric vector of scores for genes. The order of values should match with gene orders in the codon frequency matrix. The length of the vector should be equal to the number of rows in the matrix. The scores could be gene expression levels (RPKM or TPM) that are optionally log-transformed (for example, with `log1p`). The opposite of ENC will be used by default if `gene_score` is not provided. |
| `fdr` | false discovery rate used to determine optimal codons. |

## Value

data.table of optimal codons.

## Examples

```
# perform binomial regression for optimal codon estimation
cf_all <- count_codons(yeast_cds)
codons_opt <- est_optimal_codons(cf_all)
codons_opt <- codons_opt[optimal == TRUE]
codons_opt
```

---

# est_rscu *Estimate RSCU*

---

## Description

est_rscu returns the RSCU value of codons

## Usage

```
est_rscu(
  cf,
  weight = 1,
  pseudo_cnt = 1,
  codon_table = get_codon_table(),
  level = "subfam"
)
```

## Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| weight | a vector of the same length as seqs that gives different weights to CDSs when count codons. for example, it could be gene expression levels. |
| pseudo_cnt | pseudo count to avoid dividing by zero. This may occur when only a few sequences are available for RSCU calculation. |
| codon_table | a table of genetic code derived from get_codon_table or create_codon_table. |
| level | "subfam" (default) or "amino_acid". For which level to determine RSCU. |

## Value

a data.table of codon info. RSCU values are reported in the last column.

## References

Sharp PM, Tuohy TM, Mosurski KR. 1986. Codon usage in yeast: cluster analysis clearly differentiates highly and lowly expressed genes. Nucleic Acids Res 14:5125-5143.

## Examples

```
# compute RSCU of all yeast genes
cf_all <- count_codons(yeast_cds)
est_rscu(cf_all)

# compute RSCU of highly expressed (top 500) yeast genes
heg <- head(yeast_exp[order(-yeast_exp$fpkm), ], n = 500)
cf_heg <- count_codons(yeast_cds[heg$gene_id])
est_rscu(cf_heg)
```

---

# est_trna_weight *Estimate tRNA weight w*

---

## Description

`est_trna_weight` compute the tRNA weight per codon for TAI calculation. This weight reflects relative tRNA availability for each codon.

## Usage

```
est_trna_weight(
  trna_level,
  codon_table = get_codon_table(),
  s = list(WC = 0, IU = 0, IC = 0.4659, IA = 0.9075, GU = 0.7861, UG = 0.6295)
)
```

## Arguments

| | |
|---|---|
| `trna_level` | named vector of tRNA level (or gene copy numbers), one value for each anticodon. vector names are anticodons. |
| `codon_table` | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| `s` | list of non-Waston-Crick pairing panelty. |

## Value

data.table of tRNA expression information.

## References

dos Reis M, Savva R, Wernisch L. 2004. Solving the riddle of codon usage preferences: a test for translational selection. Nucleic Acids Res 32:5036-5044.

## Examples

```
# estimate codon tRNA weight for yeasts
est_trna_weight(yeast_trna_gcn)
```

get_cai *Calculate CAI*

## Description

get_cai calculates Codon Adaptation Index (CAI) of each input CDS

## Usage

```
get_cai(cf, rscu, level = "subfam")
```

## Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| rscu | rscu table containing CAI weight for each codon. This table could be generated with est_rscu or prepared manually. |
| level | "subfam" (default) or "amino_acid". For which level to determine CAI. |

## Value

a named vector of CAI values

## References

Sharp PM, Li WH. 1987. The codon Adaptation Index–a measure of directional synonymous codon usage bias, and its potential applications. Nucleic Acids Res 15:1281-1295.

## Examples

```
# estimate CAI of yeast genes based on RSCU of highly expressed genes
heg <- head(yeast_exp[order(-yeast_exp$fpkm), ], n = 500)
cf_all <- count_codons(yeast_cds)
cf_heg <- cf_all[heg$gene_id, ]
rscu_heg <- est_rscu(cf_heg)
cai <- get_cai(cf_all, rscu_heg)
head(cai)
hist(cai)
```

## get_codon_table *get codon table by NCBI gene code ID*

### Description

`get_codon_table` creates a codon table based on the given id of genetic code in NCBI.

### Usage

```
get_codon_table(gcid = "1")
```

### Arguments

| | |
|---|---|
| `gcid` | a string of genetic code id. run `show_codon_tables()` to see available codon tables. |

### Value

a `data.table` with four columns: aa_code, amino_acid, codon, and subfam.

### Examples

```
# Standard genetic code
get_codon_table()

# Vertebrate Mitochondrial genetic code
get_codon_table(gcid = '2')
```

## get_cscg *Mean Codon Stabilization Coefficients*

### Description

`get_cscg` calculates Mean Codon Stabilization Coefficients of each CDS.

### Usage

```
get_cscg(cf, csc)
```

### Arguments

| | |
|---|---|
| `cf` | matrix of codon frequencies as calculated by `count_codons()`. |
| `csc` | table of Codon Stabilization Coefficients as calculated by `est_csc()`. |

### Value

a named vector of cscg values.

## References

Presnyak V, Alhusaini N, Chen YH, Martin S, Morris N, Kline N, Olson S, Weinberg D, Baker KE, Graveley BR, et al. 2015. Codon optimality is a major determinant of mRNA stability. Cell 160:1111-1124.

## Examples

```
# estimate CSCg of yeast genes
yeast_csc <- est_csc(yeast_cds, yeast_half_life)
cf_all <- count_codons(yeast_cds)
cscg <- get_cscg(cf_all, csc = yeast_csc)
head(cscg)
hist(cscg)
```

---

# get_dp *Deviation from Proportionality*

---

## Description

get_dp calculates Deviation from Proportionality of each CDS.

## Usage

```
get_dp(
  cf,
  host_weights,
  codon_table = get_codon_table(),
  level = "subfam",
  missing_action = "ignore"
)
```

## Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| host_weights | a named vector of tRNA weights for each codon that reflects the relative availability of tRNAs in the host organism. |
| codon_table | a table of genetic code derived from get_codon_table or create_codon_table. |
| level | "subfam" (default) or "amino_acid". If "subfam", the deviation is calculated at the codon subfamily level. Otherwise, the deviation is calculated at the amino acid level. |
| missing_action | Actions to take when no codon of a group were found in a CDS. Options are "ignore" (default), or "zero" (set codon proportions to 0). |

## Value

a named vector of dp values.

## References

Chen F, Wu P, Deng S, Zhang H, Hou Y, Hu Z, Zhang J, Chen X, Yang JR. 2020. Dissimilation of synonymous codon usage bias in virus-host coevolution due to translational selection. Nat Ecol Evol 4:589-600.

## Examples

```
# estimate DP of yeast genes
cf_all <- count_codons(yeast_cds)
trna_weight <- est_trna_weight(yeast_trna_gcn)
trna_weight <- setNames(trna_weight$w, trna_weight$codon)
dp <- get_dp(cf_all, host_weights = trna_weight)
head(dp)
hist(dp)
```

---

# get_enc *Calculate ENC*

---

## Description

get_enc computes ENC of each CDS

## Usage

```
get_enc(cf, codon_table = get_codon_table(), level = "subfam")
```

## Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| codon_table | codon_table a table of genetic code derived from get_codon_table or create_codon_table. |
| level | "subfam" (default) or "amino_acid". For which level to determine ENC. |

## Value

vector of ENC values, sequence names are used as vector names

## References

- Wright F. 1990. The 'effective number of codons' used in a gene. Gene 87:23-29. - Sun X, Yang Q, Xia X. 2013. An improved implementation of effective number of codons (NC). Mol Biol Evol 30:191-196.

## Examples

```
# estimate ENC of yeast genes
cf_all <- count_codons(yeast_cds)
enc <- get_enc(cf_all)
head(enc)
hist(enc)
```

---

`get_fop` *Fraction of optimal codons (Fop)*

---

## Description

`get_fop` calculates the fraction of optimal codons (Fop) of each CDS.

## Usage

```
get_fop(cf, op = NULL, codon_table = get_codon_table(), ...)
```

## Arguments

| | |
|---|---|
| `cf` | matrix of codon frequencies as calculated by `count_codons()`. |
| `op` | a character vector of optimal codons. Can be determined automatically by running `est_optimal_codons`. |
| `codon_table` | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| `...` | other arguments passed to `est_optimal_codons`. |

## Value

a named vector of fop values.

## References

Ikemura T. 1981. Correlation between the abundance of Escherichia coli transfer RNAs and the occurrence of the respective codons in its protein genes: a proposal for a synonymous codon choice that is optimal for the E. coli translational system. J Mol Biol 151:389-409.

## Examples

```
# estimate Fop of yeast genes
cf_all <- count_codons(yeast_cds)
fop <- get_fop(cf_all)
head(fop)
hist(fop)
```

## get_gc *GC contents*

### Description

Calculate GC content of the whole sequences.

### Usage

```
get_gc(cf)
```

### Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by ‘count_codons()‘. |

### Value

a named vector of GC contents.

### Examples

```
# estimate GC content of yeast genes
cf_all <- count_codons(yeast_cds)
gc <- get_gc(cf_all)
head(gc)
hist(gc)
```

## get_gc3s *GC contents at synonymous 3rd codon positions*

### Description

Calculate GC content at synonymous 3rd codon positions.

### Usage

```
get_gc3s(cf, codon_table = get_codon_table(), level = "subfam")
```

### Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by `count_codons()`. |
| codon_table | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| level | "subfam" (default) or "amino_acid". For which level to determine GC content at synonymous 3rd codon positions. |

## Value

a named vector of GC3s values.

## References

Peden JF. 2000. Analysis of codon usage.

## Examples

```
# estimate GC3s of yeast genes
cf_all <- count_codons(yeast_cds)
gc3s <- get_gc3s(cf_all)
head(gc3s)
hist(gc3s)
```

---

# get_gc4d *GC contents at 4-fold degenerate sites*

---

## Description

Calculate GC content at synonymous position of codons (using four-fold degenerate sites only).

## Usage

```
get_gc4d(cf, codon_table = get_codon_table(), level = "subfam")
```

## Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| codon_table | a table of genetic code derived from get_codon_table or create_codon_table. |
| level | "subfam" (default) or "amino_acid". For which level to determine GC contents at 4-fold degenerate sites. |

## Value

a named vector of GC4d values.

## Examples

```
# estimate GC4d of yeast genes
cf_all <- count_codons(yeast_cds)
gc4d <- get_gc4d(cf_all)
head(gc4d)
hist(gc4d)
```

## get_tai *Calculate TAI*

### Description

get_tai calculates tRNA Adaptation Index (TAI) of each CDS

### Usage

```
get_tai(cf, trna_w)
```

### Arguments

| | |
|---|---|
| cf | matrix of codon frequencies as calculated by count_codons(). |
| trna_w | tRNA weight for each codon, can be generated with est_trna_weight(). |

### Value

a named vector of TAI values

### References

dos Reis M, Savva R, Wernisch L. 2004. Solving the riddle of codon usage preferences: a test for translational selection. Nucleic Acids Res 32:5036-5044.

### Examples

```
# calculate TAI of yeast genes based on genomic tRNA copy numbers
w <- est_trna_weight(yeast_trna_gcn)
cf_all <- count_codons(yeast_cds)
tai <- get_tai(cf_all, w)
head(tai)
hist(tai)
```

## human_mt *human mitochondrial CDS sequences*

### Description

CDSs of 13 protein-coding genes in the human mitochondrial genome extracted from ENSEMBL Biomart

### Usage

```
human_mt
```

### Format

a DNAStringSet of 13 sequences

### Source

<https://www.ensembl.org/index.html>

### Examples

```
head(human_mt)
```

---

## plot_ca_pairing *Plot codon-anticodon pairing relationship*

---

### Description

`plot_ca_pairing` show possible codon-anticodons pairings

### Usage

```
plot_ca_pairing(codon_table = get_codon_table(), plot = TRUE)
```

### Arguments

| | |
|---|---|
| `codon_table` | a table of genetic code derived from `get_codon_table` or `create_codon_table`. |
| `plot` | whether to plot the pairing relationship |

### Value

a data.table of codon info and RSCU values

### Examples

```
ctab <- get_codon_table(gcid = '2')
pairing <- plot_ca_pairing(ctab)
head(pairing)
```

## rev_comp *Reverse complement*

### Description

rev_comp creates reverse complemented version of the input sequence

### Usage

```
rev_comp(seqs)
```

### Arguments

| | |
|---|---|
| seqs | input sequences, DNAStringSet or named vector of sequences |

### Value

reverse complemented input sequences as a DNAStringSet.

### Examples

```
# reverse complement of codons
rev_comp(Biostrings::DNAStringSet(c('TAA', 'TAG')))
```

## seq_to_codons *Convert CDS to codons*

### Description

seq_to_codons converts a coding sequence to a vector of codons

### Usage

```
seq_to_codons(seq)
```

### Arguments

| | |
|---|---|
| seq | DNAString, or an object that can be coerced to a DNAString |

### Value

a character vector of codons

### Examples

```
# convert a CDS sequence to a sequence of codons
seq_to_codons('ATGTGGTAG')
seq_to_codons(yeast_cds[[1]])
```

---

## show_codon_tables *show available codon tables*

---

### Description

show_codon_tables print a table of available genetic code from NCBI through Biostrings::GENETIC_CODE_TABLE.

### Usage

```
show_codon_tables()
```

### Value

No return value (NULL). Available codon tables will be printed out directly.

### Examples

```
# print available NCBI codon table IDs and descriptions.
show_codon_tables()
```

---

## slide *slide window interval generator*

---

### Description

slide generates a data.table with start, center, and end columns for a sliding window analysis.

### Usage

```
slide(from, to, step = 1, before = 0, after = 0)
```

### Arguments

| | |
|---|---|
| from | integer, the start of the sequence |
| to | integer, the end of the sequence |
| step | integer, the step size |
| before | integer, the number of values before the center of a window |
| after | integer, the number of values after the center of a window |

## Value

data.table with start, center, and end columns

## Examples

```
slide(1, 10, step = 2, before = 1, after = 1)
```

---

`slide_apply` *apply a cub index to a sliding window*

---

## Description

`slide_apply` applies a function to a sliding window of codons.

## Usage

```
slide_apply(seq, .f, step = 1, before = 0, after = 0, ...)
```

## Arguments

| | |
|---|---|
| `seq` | DNAString, the sequence |
| `.f` | function, the codon index calculation function to apply, for example, `get_enc`. |
| `step` | integer, the step size in number of codons |
| `before` | integer, the number of codons before the center of a window |
| `after` | integer, the number of codons after the center of a window |
| `...` | additional arguments to pass to the function `.f` |

## Value

data.table with start, center, end, and codon usage index columns

## Examples

```
slide_apply(yeast_cds[[1]], get_enc, step = 1, before = 10, after = 10)
```

## slide_codon *sliding window of codons*

### Description

slide_codon generates a data.table with start, center, and end columns for a sliding window analysis of codons.

### Usage

```
slide_codon(seq, step = 1, before = 0, after = 0)
```

### Arguments

| | |
|---|---|
| seq | DNAString, the sequence |
| step | integer, the step size |
| before | integer, the number of codons before the center of a window |
| after | integer, the number of codons after the center of a window |

### Value

data.table with start, center, and end columns

### Examples

```
x <- Biostrings::DNAString('ATCTACATAGCTACGTAGCTCGATGCTAGCATGCATCGTACGATCGTCGATCGTAG')
slide_codon(x, step = 3, before = 1, after = 1)
```

## slide_plot *plot sliding window codon usage*

### Description

slide_plot visualizes codon usage in sliding window.

### Usage

```
slide_plot(windt, index_name = "Index")
```

### Arguments

| | |
|---|---|
| windt | data.table, the sliding window codon usage generated by slide_apply. |
| index_name | character, the name of the index to display. |

## Value

ggplot2 plot.

## Examples

```
sw <- slide_apply(yeast_cds[[1]], get_enc, step = 1, before = 10, after = 10)
slide_plot(sw)
```

---

# yeast_cds *yeast CDS sequences*

---

## Description

CDSs of all protein-coding genes in Saccharomyces_cerevisiae

## Usage

```
yeast_cds
```

## Format

a DNAStringSet of 6600 sequences

## Source

<https://ftp.ensembl.org/pub/release-107/fasta/saccharomyces_cerevisiae/cds/Saccharomyces_cerevisiae.R64-1-1.cds.all.fa.gz>

## Examples

```
head(yeast_cds)
```

---

# yeast_exp *yeast mRNA expression levels*

---

## Description

Yeast mRNA FPKM determined from rRNA-depleted (RiboZero) total RNA-Seq libraries. RUN1_0_WT and RUN2_0_WT (0 min after RNA Pol II repression) were averaged and used here.

## Usage

```
yeast_exp
```

## Format

a data.frame with 6717 rows and three columns:

**gene_id** gene ID

**gene_name** gene name

**fpkm** mRNA expression level in Fragments per kilobase per million reads

## Source

<https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE57385>

## References

Presnyak V, Alhusaini N, Chen YH, Martin S, Morris N, Kline N, Olson S, Weinberg D, Baker KE, Graveley BR, et al. 2015. Codon optimality is a major determinant of mRNA stability. Cell 160:1111-1124.

## Examples

```
head(yeast_exp)
```

---

`yeast_half_life` *Half life of yeast mRNAs*

---

## Description

Half life of yeast mRNAs in Saccharomyces_cerevisiae calculated from rRNA-deleted total RNAs by Presnyak et al.

## Usage

```
yeast_half_life
```

## Format

a data.frame with 3888 rows and three columns:

**gene_id** gene id

**gene_name** gene name

**half_life** mRNA half life in minutes

## Source

<https://doi.org/10.1016/j.cell.2015.02.029>

## References

Presnyak V, Alhusaini N, Chen YH, Martin S, Morris N, Kline N, Olson S, Weinberg D, Baker KE, Graveley BR, et al. 2015. Codon optimality is a major determinant of mRNA stability. Cell 160:1111-1124.

## Examples

```
head(yeast_half_life)
```

---

`yeast_trna` *yeast tRNA sequences*

---

## Description

Yeast tRNA sequences obtained from gtRNAdb.

## Usage

```
yeast_trna
```

## Format

a RNAStringSet with a length of 275.

## Source

<http://gtrnadb.ucsc.edu/genomes/eukaryota/Scere3/sacCer3-mature-tRNAs.fa>

## References

Chan PP, Lowe TM. 2016. GtRNAdb 2.0: an expanded database of transfer RNA genes identified in complete and draft genomes. Nucleic Acids Res 44:D184-189.

## Examples

```
yeast_trna
```

yeast_trna_gcn *yeast tRNA gene copy numbers (GCN)*

## Description

Yeast tRNA gene copy numbers (GCN) by anticodon obtained from gtRNAdb.

## Usage

```
yeast_trna_gcn
```

## Format

a named vector with a length of 41. Value names are anticodons.

## Source

<http://gtrnadb.ucsc.edu/genomes/eukaryota/Scere3/sacCer3-mature-tRNAs.fa>

## References

Chan PP, Lowe TM. 2016. GtRNAdb 2.0: an expanded database of transfer RNA genes identified in complete and draft genomes. Nucleic Acids Res 44:D184-189.

## Examples

```
yeast_trna_gcn
```

# Index