

# Package ‘trqwe’

December 13, 2017

**Type** Package

**Title** Performance oriented statistical metrics and utility functions.

**Version** 0.1

**Date** 2016-01-01

**Author** Travers Ching

**Maintainer** Travers Ching <traversc@gmail.com>

**Description** Performance oriented statistical metrics and utility functions.

**License** GPL2.0

**LazyLoad** yes

**Imports** Rcpp (>= 0.11.0)

**LinkingTo** Rcpp

**RoxygenNote** 6.0.1.9000

## R topics documented:

add_colnames . . . . .	2
add_rownames . . . . .	3
allDups . . . . .	3
append<- . . . . .	4
bioc . . . . .	4
chop . . . . .	5
cindex . . . . .	5
cosineDist . . . . .	6
f1score . . . . .	6
fastAUC . . . . .	7
fastPR . . . . .	8
fastReadLines . . . . .	8
fastROC . . . . .	9
fpkmFromCounts . . . . .	10
head2 . . . . .	10
install . . . . .	11
logit . . . . .	11
make_percent . . . . .	12
matrixFactor . . . . .	12
mccscore . . . . .	13
mcreadRDS . . . . .	13

mcsaveRDS . . . . .	14
mcsplitapply . . . . .	14
mgrepl . . . . .	15
nelson_aelen_surv . . . . .	16
posteriorBalance . . . . .	17
prepend<- . . . . .	17
reload . . . . .	18
reloadtrqwe . . . . .	18
se . . . . .	19
sigmoid . . . . .	19
statsCallback . . . . .	20
tablec . . . . .	20
tail2 . . . . .	21
TCGA_barcode . . . . .	21
timePrompt . . . . .	22
topn . . . . .	22
trqwe_KNN . . . . .	23
varSizes . . . . .	23
ww_test . . . . .	24
%Q% . . . . .	25
<b>Index</b>	<b>26</b>

---

add_colnames	<i>Set colnames of data.frame or matrix</i>
--------------	---

---

**Description**

Set colnames of data.frame or matrix and return it, for use with pipes

**Usage**

add\_colnames(df, colnames)

**Arguments**

- df                      data.frame or matrix
- colnames              colnames to add

**Value**

df with colnames added

---

add_rownames	<i>Set rownames of data.frame or matrix</i>
--------------	---

---

**Description**

Set rownames of data.frame or matrix and return it, for use with pipes

**Usage**

```
add_rownames(df, rownames)
```

**Arguments**

df	data.frame or matrix
rownames	rownames to add

**Value**

df with rownames added

---

allDups	<i>All duplicates.</i>
---------	------------------------

---

**Description**

Finds all duplicates in a vector including first instances of duplicates.

**Usage**

```
allDups(vec)
```

**Arguments**

vec	A vector.
-----	-----------

**Value**

A boolean vector of the same length as vec. TRUE if the element is duplicated, FALSE if not.

**Examples**

```
allDups(sample(1:100, size=100, replace=T))
```

---

append<-	<i>Append to a vector.</i>
----------	----------------------------

---

**Description**

Appends the 2nd argument to the 1st.

**Usage**

```
append(x) <- value
```

**Arguments**

x	A vector.
value	The element to append.

**Examples**

```
x <- 1:5
append(x) <- 6
print(x)

[1] 1 2 3 4 5 6
```

---

bioc	<i>Install Bioconductor package.</i>
------	--------------------------------------

---

**Description**

Utility function for installing packages from bioconductor easily.

**Usage**

```
bioc(package)
```

**Arguments**

package	unquoted package name.
---------	------------------------

**Examples**

```
\code{bioc(DESeq2)}
```

---

`chop`*Cleans leading and trailing whitespace.*

---

**Description**

Removes leading and trailing whitespace in a vector of strings.

**Usage**

```
chop(x)
```

**Arguments**

`x` A character vector.

**Value**

A vector with leading and trailing whitespace removed.

**Examples**

```
chop(c("  hello ", " 123 \t"))
```

```
[1] "hello" "123"
```

---

`cindex`*Concordance Index.*

---

**Description**

Calculates the concordance index from the results of a censored survival model. Very fast compared to other packages.

**Usage**

```
cindex(probs, time, event)
```

**Arguments**

`probs` The prognostic score of each patient.

`time` The time of each patient.

`event` The death of each patient: 1 for a patient death, 0 for censored.

**Value**

The concordance index.

---

cosineDist	<i>Cosine distance.</i>
------------	-------------------------

---

**Description**

Calculates the cosine distance of rows of a matrix.

**Usage**

```
cosineDist(x)
```

**Arguments**

`x`                      A matrix.

**Value**

Cosine distance as a dist object.

**See Also**

<http://stackoverflow.com/questions/2535234/find-cosine-similarity-in-r>

**Examples**

```
rbind(c(1,1,1,0,0,0), c(0,0,0,1,1,1), c(1,1,1,0,0,1)) -> x
cosineDist(x)
      1      2
2 1.0000000
3 0.1339746 0.7113249
```

---

f1score	<i>F1 score.</i>
---------	------------------

---

**Description**

Calculates F1 score from the results of a classification model.

**Usage**

```
f1score(probs, class)
```

**Arguments**

`probs`                      A numeric vector where 1 is predicted positive and 0 is predicted negative.  
`class`                      A numeric vector where 1 is positive and 0 is negative.

**Value**

The F1 score.

**See Also**

[https://en.wikipedia.org/wiki/F1\\_score](https://en.wikipedia.org/wiki/F1_score)

---

fastAUC	<i>Fast AUC</i>
---------	-----------------

---

**Description**

This function calculates the Area Under the Receiver-Operator Curve from the results of a classification model.

**Usage**

```
fastAUC(probs, class, method = "trqwe")
```

**Arguments**

probs	A numeric vector of probabilities or likelihoods for each data point.
class	A numeric vector where 1 is positive and 0 is negative.

**Value**

The AUC.

**See Also**

Reference <https://stat.ethz.ch/pipermail/r-help/2005-September/079872.html>

**Examples**

```
library(AppliedPredictiveModeling)
data(abalone)
library(glmnet)
class <- factor(as.numeric(abalone$Type == "M"))
data <- data.matrix(abalone[, -1])
fit <- cv.glmnet(data, class, family="binomial")
probs <- predict(fit, newx=data)[,1]
fastAUC(probs, class)
```

---

fastPR	<i>Precision-Recall curve</i>
--------	-------------------------------

---

**Description**

Calculates the points in a Precision-Recall from the results of a classification model.

**Usage**

```
fastPR(probs, class)
```

**Arguments**

probs	A numeric vector of probabilities or likelihoods for each data point.
class	A numeric vector where 1 is positive and 0 is negative.

**Value**

a list containing the ROC curve.

**Examples**

```
library(AppliedPredictiveModeling)
data(abalone)
library(glmnet)
class <- factor(as.numeric(abalone$Type == "M"))
data <- data.matrix(abalone[, -1])
fit <- cv.glmnet(data, class, family="binomial")
probs <- predict(fit, newx=data)[,1]
with(fastPR(probs, class), plot(recall, precision, type="l"))
```

---

fastReadLines	<i>Fast readLines.</i>
---------------	------------------------

---

**Description**

Replacement for readLines, faster.

**Usage**

```
fastReadLines(fname, newlinechar = "\n", nchars = NULL)
```

**Arguments**

fname	Filename to read.
newlinechar	The new line character in the file.

**Value**

A vector containing all the lines in the file.



**Examples**

```
library(microbenchmark)
writeLines(replicate(100000, sample(letters, size=100, replace=T)), con="/tmp/temp.txt")
microbenchmark(fastReadLines("/tmp/temp.txt"),
               readLines("/tmp/temp.txt"), times=3)[,c("expr", "mean"), drop=F]

Unit: milliseconds
      expr      min       lq     mean  median      uq
fastReadLines("/tmp/temp.txt") 335.3874 335.9188 336.5900 336.4502 337.1912
  readLines("/tmp/temp.txt") 724.9136 725.4523 727.2444 725.9911 728.4098
```

---

fastROC	<i>Fast ROC</i>
---------	-----------------

---

**Description**

Calculates the points in a Receiver-Operator Curve from the results of a classification model.

**Usage**

```
fastROC(probs, class)
```

**Arguments**

probs	A numeric vector of probabilities or likelihoods for each data point.
class	A numeric vector where 1 is positive and 0 is negative.

**Value**

a list containing the ROC curve.

**Examples**

```
library(AppliedPredictiveModeling)
data(abalone)
library(glmnet)
class <- factor(as.numeric(abalone$Type == "M"))
data <- data.matrix(abalone[, -1])
fit <- cv.glmnet(data, class, family="binomial")
probs <- predict(fit, newx=data)[,1]
with(fastROC(probs, class), plot(fpr, tpr, type="l"))
```

---

fpkmFromCounts	<i>Calculate FPKM.</i>
----------------	------------------------

---

### Description

Calculates FPKM from a count matrix.

### Usage

```
fpkmFromCounts(mat, gene_lengths, uq_norm = T)
```

### Arguments

mat	An integer matrix representing the counts of a RNA-Seq dataset.
gene_lengths	A named vector with the length of each gene.
uq_norm	If TRUE, will perform upper-quartile normalization.

### Value

A matrix containing FPKM with the same dimensions as mat.

### See Also

FPKM-UQ - A normalized read count in which gene expression values, in FPKM, are divided by the 75th percentile value.

[https://gdc-docs.nci.nih.gov/Data/Bioinformatics\\_Pipelines/Expression\\_mRNA\\_Pipeline/](https://gdc-docs.nci.nih.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/)  
<http://vinaykmittal.blogspot.com/2013/10/fpkmrpkm-normalization-caveat-and-upper.html>

---

head2	<i>Upper-left corner of matrix.</i>
-------	-------------------------------------

---

### Description

Shortcut function for previewing a matrix or data.frame by displaying the upper-left corner. Similar to head.

### Usage

```
head2(x, n = 10, ncols = 10)
```

### Arguments

x	A wide matrix or data.frame.
n	Number of lines to display. Default 10.
ncols	Number of columns to display. Default 10.

### Value

n by ncols subset of the matrix taken from the upper-left corner.

---

install	<i>Install CRAN package.</i>
---------	------------------------------

---

**Description**

Utility function for installing packages from CRAN easily.

**Usage**

```
install(package, repos = "http://cran.us.r-project.org")
```

**Arguments**

package	unquoted package name.
---------	------------------------

**Examples**

```
\code{install(Rcpp)}
```

---

logit	<i>Logit Transformation.</i>
-------	------------------------------

---

**Description**

Calculates the results of the Logit Transformation.

**Usage**

```
logit(x)
```

**Arguments**

x	Input to the function (e.g., probabilities from 0 to 1).
---	--

**Value**

Logit(x)

---

make_percent	<i>Make percentage from number</i>
--------------	------------------------------------

---

**Description**

Make percentage from number

**Usage**

```
make_percent(x, digits = 2)
```

**Arguments**

x	A number.
digits	Number of decimal places, default 2.

**Value**

A percentage.

**Examples**

```
make_percent(0.424, 2)
```

---

matrixFactor	<i>Matrix Factor Design.</i>
--------------	------------------------------

---

**Description**

From a factor, returns a design matrix with a column for each level.

**Usage**

```
matrixFactor(x, names = NULL)
```

**Arguments**

x	A factor.
names	The name of each instance in the resulting matrix (i.e., the rownames).

**Value**

The design matrix.

**Examples**

```
matrixFactor(factor(letters))
```

---

mccscore	<i>Matthew's correlation coefficient</i>
----------	--

---

**Description**

Calculates Matthew's correlation coefficient. Requires the Rmpfr package.

**Usage**

```
mccscore(probs, class)
```

**Arguments**

probs	A numeric vector where 1 is predicted positive and 0 is predicted negative.
class	A numeric vector where 1 is positive and 0 is negative.

**Value**

The MCC score.

**See Also**

[https://en.wikipedia.org/wiki/Matthews\\_correlation\\_coefficient](https://en.wikipedia.org/wiki/Matthews_correlation_coefficient)

---

mcreadRDS	<i>Multi-threaded readRDS</i>
-----------	-------------------------------

---

**Description**

Uses the pigz utility to improve loading large R objects. This is compatible with saveRDS and readRDS functions. Requires pigz (sudo apt-get install pigz on Ubuntu).

**Usage**

```
mcreadRDS(file, mc.cores = min(parallel::detectCores(), 4))
```

**Arguments**

file	The filename of the rds object.
mc.cores	How many cores to use in pigz. The program does not seem to benefit after more than about 4 cores.

**Value**

The R object.

**See Also**

<http://stackoverflow.com/questions/28927750/>

**Examples**

```
x <- sample(1e4, 1e7, replace=T)
saveRDS(x, file="temp.Rds")
xmc <- mcreadRDS("temp.Rds")
```

---

mcsaveRDS

*Multi-threaded saveRDS*


---

**Description**

Uses the pigz utility to improve saving large R objects. This is compatible with saveRDS and readRDS functions. Requires pigz (sudo apt-get install pigz on Ubuntu).

**Usage**

```
mcsaveRDS(object, file, mc.cores = min(parallel::detectCores(), 4))
```

**Arguments**

object	An r object to save.
file	The filename to save to.
mc.cores	How many cores to use in pigz. The program does not seem to benefit after more than about 4 cores.

**See Also**

<http://stackoverflow.com/questions/28927750/>

**Examples**

```
x <- sample(1e4, 1e7, replace=T)
y <- sample(1e4, 1e7, replace=T)
microbenchmark(mcsaveRDS(x, file="temp.Rds"), saveRDS(y, file="temp2.Rds"))
```

Unit: seconds

	expr	min	lq	mean	median	uq
mcsaveRDS(x, file = "temp.Rds")		1.908310	1.908310	1.908310	1.908310	1.908310
saveRDS(y, file = "temp2.Rds")		6.271499	6.271499	6.271499	6.271499	6.271499

---

mcsplitapply

*Parallel split-matrix loop.*


---

**Description**

Splits a matrix into subsets based on a factor, and applies a function to each subset. Typical use case: sum exon count data to gene count data.

**Usage**

```
mcsplitapply(mat, f, func, mc.cores = 4, .combine = rbind, ...)
```

**Arguments**

<code>mat</code>	The matrix.
<code>f</code>	A factor of length equal to <code>nrow(mat)</code> . The levels of this factor will split the matrix into subsets.
<code>func</code>	The function to apply to each subset.
<code>mc.cores</code>	The number of cores to use.
<code>.combine</code>	The function to combine the results with. Default is <code>rbind</code> . Use <code>NA</code> to return a list.

**Value**

A list or a combined object depending on the `.combine` parameter.

**Examples**

```
library(pasilla)
library(DEXSeq)
library(trqwe)
data(pasillaDEXSeqDataSet)
exon_counts <- counts(dxd)
f <- rowData(dxd)$groupID
gene_counts <- mcsplitapply(exon_counts, f, colSums)
```

---

mgrepl

---

*Multiple grepl.*


---

**Description**

Takes in a list of regex patterns and returns true if any pattern matches.

**Usage**

```
mgrepl(patterns, x, ...)
```

**Arguments**

<code>x</code>	A vector of strings to search.
<code>pattern</code>	A vector of regex patterns.

**Value**

A vector of the same length as `x`, `TRUE` if any pattern matches.

**Examples**

```
x <- fastReadLines("http://textfiles.com/ufo/ufobooks.ufo", newlinechar="\r\n", 1e5)
head(x[mgrepl(c("ALIENS", "UFO"), x)])
```

```
[1] "                                UFOLOGY BOOKS (REVISION 2.1 343 books)"
[2] "      H. S. Stewart on the  subject of UFO's. The list is alphabetic"
[3] "      Tom Mickus's most excellent board UFONET I.  (416-237-1204)"
[4] "      Bill Adler                *  LETTERS TO THE AIR FORCE ON UFOS  1967"
[5] "      Gordon W. Allen           OVERLORDS OLYMPIANS AND THE UFO  1974"
[6] "      Robert B. Beard           FLYING SAUCERS, UFO'S AND EXTRA"
```

---

nelson_aelen_surv	<i>Nelson Aalen estimator</i>
-------------------	-------------------------------

---

**Description**

Nelson–Aalen estimator is an estimator of the cumulative hazard function in survival data. It can be used to compare the overall risks of two groups or used to estimate the number of deaths before a certain time.

**Usage**

```
nelson_aelen_surv(time, event)
```

**Arguments**

time	The time of each patient.
event	The death of each patient: 1 for a patient death, 0 for censored.

**Value**

A list containing the cumulative hazard function.

**See Also**

[#https://en.wikipedia.org/wiki/Nelson-Aalen\\_estimator](https://en.wikipedia.org/wiki/Nelson-Aalen_estimator)

**Examples**

```
library(survival)
data(veteran)
with(nelson_aelen_surv(veteran$time, veteran$status), plot(ti, Hi, type="b"))
```



---

posteriorBalance	<i>Posterior probability adjustment.</i>
------------------	--

---

### Description

Adjusts the posterior probability of a classifier based on unbalanced datasets. In classification model where the negative data is randomly under-sampled and all the positive data is used, the adjustment factor (beta) is  $p(s=1|-) = p(+)/p(-)$ . I.e., the probability that a negative datapoint is selected in the classifier.  $\text{beta} \sim N+/N-$ .

### Usage

```
posteriorBalance(probs, beta = NULL, Nplus, Nminus)
```

### Arguments

probs	The original posterior probability.
beta	The adjustment factor.
Nplus	The number of positive examples in the real data. Only used if beta is NULL.
Nminus	The number of negative examples in the real data. Only used if beta is NULL.

### Value

The adjusted posterior probability.

### See Also

Dal Pozzolo, Andrea, et al. "Calibrating probability with undersampling for unbalanced classification." Computational Intelligence, 2015 IEEE Symposium Series on. IEEE, 2015.

---

prepend<-	<i>Prepend to a vector.</i>
-----------	-----------------------------

---

### Description

Prepends the 2nd argument to the 1st.

### Usage

```
prepend(x) <- value
```

### Arguments

x	A vector.
value	The element to append.

**Examples**

```
x <- 1:5
prepend(x) <- 6
print(x)

[1] 6 1 2 3 4 5
```

---

**reload**
*Reload a package.*


---

**Description**

Unload and reload a package and sets the namespace search order.

**Usage**

```
reload(package, pos = 2)
```

**Arguments**

package	Unquoted package name.
pos	Namespace search position.

**Examples**

```
\code{reload(trqwe)}
```

---

**reloadtrqwe**
*Unload and reload trqwe.*


---

**Description**

Unload and reload trqwe. Shortcut for reload(trqwe)

**Usage**

```
reloadtrqwe()
```

---

se	<i>Standard error.</i>
----	------------------------

---

**Description**

Calculates the standard error of a sampling distribution.

**Usage**

```
se(x)
```

**Arguments**

x	A vector.
---	-----------

**Value**

The standard error of x.

**Examples**

```
x <- rnorm(1e3)
se(x)
[1] 0.03192027
```

---

sigmoid	<i>Sigmoid Function.</i>
---------	--------------------------

---

**Description**

Calculates the results of the sigmoid function.

**Usage**

```
sigmoid(x)
```

**Arguments**

probs	x Input to the function.
-------	--------------------------

**Value**

Sigmoid(x)

---

statsCallback	<i>Variable information.</i>
---------------	------------------------------

---

### Description

Automatically stores basic information of variables in the previous command. This function adds a callback which reports information on previous variables and stores this information in `.stats`.

### Usage

```
statsCallback()
```

### Examples

```
statsCallback()
my_data <- VADeaths
.stats
> [1] "dim: 5 4, length: 20, class: matrix, typeof: double"
```

---

tablec	<i>Fast C++ tabulation.</i>
--------	-----------------------------

---

### Description

Takes in a character, integer or factor vector and tabulates the number of times each element appears.

### Usage

```
tablec(x, sort = F)
```

### Arguments

<code>x</code>	A character, integer or factor vector. NAs are allowed.
<code>sort</code>	TRUE if the result names should be sorted alphanumerically.

### Value

A integer vector of counts of each element.

### Examples

```
x <- factor(sample(1e5, 1e8, replace=T))
microbenchmark(table(x), tablec(x), times=3)
```

```
Unit: milliseconds
      expr      min       lq      mean   median      uq      max
table(x) 9777.6457 10479.0949 10717.3382 11180.544 11187.1844 11193.8246
tablec(x)  678.0364   685.9467   713.1181   693.857   730.6589   767.4608
```

```
x <- sample(letters, 1e8, replace=T)
microbenchmark(table(x), tablec(x), tablec(x,sort=T), times=3)

Unit: seconds
      expr      min       lq     mean  median      uq      max
table(x) 5.778514 5.829125 5.855560 5.879737 5.894083 5.908430
tablec(x) 1.589360 1.589381 1.589516 1.589402 1.589594 1.589786
tablec(x, sort = T) 1.589386 1.590520 1.591824 1.591655 1.593044 1.594432
```

---

tail2	<i>Lower-left corner of matrix.</i>
-------	-------------------------------------

---

### Description

Shortcut function for previewing the bottom of a matrix or data.frame by displaying the lower-left corner. Similar to tail.

### Usage

```
tail2(x, n = 10, ncols = 10)
```

### Arguments

x	A wide matrix or data.frame.
n	Number of lines to display. Default 10.
ncols	Number of columns to display. Default 10.

### Value

n by ncols subset of the matrix taken from the lower-left corner.

---

TCGA_barcode	<i>Parse TCGA barcode.</i>
--------------	----------------------------

---

### Description

Taking in a full TCGA sample barcode, or any subset of the barcode, and return extracted values.

### Usage

```
TCGA_barcode(x, what = "patient")
```

### Arguments

x	TCGA barcode, or a vector of barcodes.
what	Which information to return.

### Value

The specified information contained in the barcode of the same length as x.

**Examples**

```
TCGA_barcode(c("TCGA-02-0001-01C-01D-0182-01", "TCGA-02-0001-11C-01D-0182-01"), what="tissue")

[1] "01" "11"
```

---

timePrompt

*Time profiling.*


---

**Description**

Reports time in seconds to the R prompt of the previous command. This function adds a callback which saves the running of individual commands and reports the time in seconds on the next line.

**Usage**

```
timePrompt()
```

**Examples**

```
> timePrompt()
0.000s> x <- sample(1:10, size=1e8, replace=T)
1.240s>
Note - this time is not accurate if child processes or multithreading is involved.
```

---

topn

*Highest elements in a vector.*


---

**Description**

Finds the top elements in a vector very quickly. Equivalent of `-sort(-x, partial=1:n)`

**Usage**

```
topn(x, n = 100, value = F, lowest = F)
```

**Arguments**

x	A numeric vector.
n	The number of top elements to return.
value	If TRUE, returns the values of the top elements. If FALSE, returns the indices.
lowest	If TRUE, returns the lowest elements instead of the highest.

**Value**

A vector containing the indices or the values of the top elements.

**See Also**

<http://stackoverflow.com/questions/18450778/>

**Examples**

```
naive_top <- function(x, n) {
  -sort(-x, partial=1:n)
}
x <- runif(1e7)
microbenchmark(naive_top(x,100), topn(x,100,value=T), times=10)

Unit: milliseconds
      expr      min       lq     mean   median      uq
naive_top(x, 100) 1070.0180 1071.5951 1075.964 1072.3520 1073.9989
topn(x, 100, value = T) 433.6682 433.8882 434.771 434.4986 435.6029
```

---

trqwe_KNN	<i>Fast binary KNN classifier</i>
-----------	-----------------------------------

---

**Description**

Fast binary KNN classifier

**Usage**

```
trqwe_KNN(distmat, train_idx, test_idx, classes, K, mc.cores = 1)
```

**Arguments**

distmat	A NxN pre-computed distance matrix
train_idx	Train indices
test_idx	Test indices
classes	vector length N, 1 or 0
K	Number of nearest neighbors parameter
mc.cores	Number of threads to use

**Value**

A prediction vector for the test set based on the class labels of the train set.

---

varSizes	<i>Size of R objects.</i>
----------	---------------------------

---

**Description**

Prints out the size of all R objects in the environment.

**Usage**

```
varSizes(env = globalenv(), units = "KB")
```

**Arguments**

<code>env</code>	The environment to search (default global environment).
<code>units</code>	Units to print out for each variable.

**Value**

A data.frame containing the size of each object.

---

<code>ww_test</code>	<i>Wald-Wolfowitz Runs Tests for Randomness</i>
----------------------	---

---

**Description**

This is the k-category asymptotic Z Test with continuity correction. Imagine rolling a die multiple times to obtain a sequence of rolls. This statistic tests whether there exists a "run" within the sequence where one particular number comes up more times in a row than expected randomly. If the test is significant, it can be concluded that the die rolls are not independent.

**Usage**

```
ww_test(x)
```

**Arguments**

<code>x</code>	A vector of items, coerced into a factor.
----------------	---

**Value**

A p-value for the statistical test.

**See Also**

Reference [https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Analysis\\_of\\_Runs.pdf](https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Analysis_of_Runs.pdf)

**Examples**

```
set.seed(1)
ww_test(sample(2, 100, replace=T))
ww_test(c(sample(6, 90, replace=T), rep(1, 10)))
```



---

%Q%	<i>Concatenate strings.</i>
-----	-----------------------------

---

### Description

Concatenates two strings.

### Usage

a %Q% b

### Arguments

a	First string.
b	Second string.

### Value

The concatenated string.

### Examples

```
'Hello ' %Q% 'World'
```

```
[1] "Hello World"
```

# Index

[%Q%](#), [25](#)

[add\\_colnames](#), [2](#)  
[add\\_rownames](#), [3](#)  
[allDups](#), [3](#)  
[append<-](#), [4](#)

[bioc](#), [4](#)

[chop](#), [5](#)  
[cindex](#), [5](#)  
[cosineDist](#), [6](#)

[f1score](#), [6](#)  
[fastAUC](#), [7](#)  
[fastPR](#), [8](#)  
[fastReadLines](#), [8](#)  
[fastROC](#), [9](#)  
[fpkmFromCounts](#), [10](#)

[head2](#), [10](#)

[install](#), [11](#)

[logit](#), [11](#)

[make\\_percent](#), [12](#)  
[matrixFactor](#), [12](#)  
[mccscore](#), [13](#)  
[mcreadRDS](#), [13](#)  
[mcsaveRDS](#), [14](#)  
[mcsplitapply](#), [14](#)  
[mgrepl](#), [15](#)

[nelson\\_aelen\\_surv](#), [16](#)

[posteriorBalance](#), [17](#)  
[prepend<-](#), [17](#)

[reload](#), [18](#)  
[reloadtrqwe](#), [18](#)

[se](#), [19](#)  
[sigmoid](#), [19](#)  
[statsCallback](#), [20](#)

[tablec](#), [20](#)  
[tail2](#), [21](#)  
[TCGA\\_barcode](#), [21](#)  
[timePrompt](#), [22](#)  
[topn](#), [22](#)  
[trqwe\\_KNN](#), [23](#)

[varSizes](#), [23](#)

[ww\\_test](#), [24](#)