

# Package ‘sigr’

October 14, 2022

**Type** Package

**Title** Succinct and Correct Statistical Summaries for Reports

**Version** 1.1.4

**Date** 2021-06-11

**URL** <https://github.com/WinVector/sigr/>,  
<https://winvector.github.io/sigr/>

**Maintainer** John Mount <jmount@win-vector.com>

**BugReports** <https://github.com/WinVector/sigr/issues>

**Description** Succinctly and correctly format statistical summaries of various models and tests (F-test, Chi-Sq-test, Fisher-test, T-test, and rank-significance). This package also includes empirical tests, such as Monte Carlo and bootstrap distribution estimates.

**License** GPL-2 | GPL-3

**RoxygenNote** 7.1.1

**Depends** R (>= 3.2.1)

**Imports** wrapr (>= 2.0.2), stats

**Suggests** pwr, parallel, knitr, rmarkdown, tinytest

**VignetteBuilder** knitr

**ByteCompile** true

**NeedsCompilation** no

**Author** John Mount [aut, cre],  
Nina Zumel [aut],  
Win-Vector LLC [cph]

**Repository** CRAN

**Date/Publication** 2021-06-12 04:40:03 UTC

**R topics documented:**

add_ROC_derived_columns . . . . .	4
as.character.sigr_statistic . . . . .	4
Bernoulli_diff_stat . . . . .	5
build_ROC_curve . . . . .	6
calcAUC . . . . .	7
calcDeviance . . . . .	8
calcSSE . . . . .	8
estimateDifferenceZeroCrossing . . . . .	9
find_area_q . . . . .	10
find_AUC_q . . . . .	10
find_matching_a1_1b . . . . .	11
find_matching_conditional_betas . . . . .	12
fit_beta_shapes . . . . .	13
format.sigr_statistic . . . . .	14
getRenderingFormat . . . . .	15
model_utility . . . . .	15
permTestAUC . . . . .	17
permutationScoreModel . . . . .	18
print.sigr_statistic . . . . .	19
render . . . . .	20
render.sigr_aucpairtest . . . . .	21
render.sigr_aucpermtest . . . . .	21
render.sigr_aucresamptest . . . . .	22
render.sigr_Bernoulli_diff_test . . . . .	23
render.sigr_binomtest . . . . .	24
render.sigr_chisqtest . . . . .	25
render.sigr_cohend . . . . .	26
render.sigr_cortest . . . . .	26
render.sigr_emptest . . . . .	27
render.sigr_fishtest . . . . .	28
render.sigr_ftest . . . . .	29
render.sigr_permtest . . . . .	30
render.sigr_pwr_htest . . . . .	31
render.sigr_significance . . . . .	31
render.sigr_tinterval . . . . .	32
render.sigr_ttest . . . . .	33
resampleScoreModel . . . . .	34
resampleScoreModelPair . . . . .	35
resampleTestAUC . . . . .	37
sensitivity_and_specificity_s12p12n . . . . .	38
sensitivity_from_specificity_q . . . . .	39
sigr . . . . .	40
testAUCpair . . . . .	41
TInterval . . . . .	42
TInterval.data.frame . . . . .	42
TInterval.numeric . . . . .	43

TIntervalS . . . . .	44
wrapBinomTest . . . . .	45
wrapBinomTest.data.frame . . . . .	45
wrapBinomTest.htest . . . . .	46
wrapBinomTest.logical . . . . .	47
wrapBinomTest.numeric . . . . .	48
wrapBinomTestS . . . . .	49
wrapChiSqTest . . . . .	50
wrapChiSqTest.anova . . . . .	51
wrapChiSqTest.data.frame . . . . .	52
wrapChiSqTest.glm . . . . .	53
wrapChiSqTest.summary.glm . . . . .	53
wrapChiSqTestImpl . . . . .	54
wrapCohenD . . . . .	55
wrapCohenD.data.frame . . . . .	55
wrapCohenD.numeric . . . . .	56
wrapCorTest . . . . .	57
wrapCorTest.data.frame . . . . .	57
wrapCorTest.htest . . . . .	58
wrapFisherTest . . . . .	59
wrapFisherTest.data.frame . . . . .	59
wrapFisherTest.htest . . . . .	61
wrapFisherTest.table . . . . .	62
wrapFTest . . . . .	63
wrapFTest.anova . . . . .	63
wrapFTest.data.frame . . . . .	64
wrapFTest.htest . . . . .	65
wrapFTest.lm . . . . .	66
wrapFTest.summary.lm . . . . .	67
wrapFTestezANOVA . . . . .	67
wrapFTestImpl . . . . .	68
wrapPWR . . . . .	69
wrapPWR.power.htest . . . . .	69
wrapSignificance . . . . .	70
wrapTTest . . . . .	71
wrapTTest.data.frame . . . . .	71
wrapTTest.htest . . . . .	72
wrapTTest.numeric . . . . .	73

---

```
add_ROC_derived_columns
```

*Add ROC derived columns.*

---

### Description

Add ROC columns derived from sensitivity and specificity.

### Usage

```
add_ROC_derived_columns(d, positive_prevalence)
```

### Arguments

`d` input data frame, must at least of columns Sensitivity and Specificity  
`positive_prevalence` scalar, the prevalence of the positive class or prior odds

### Value

extended data frame with more columns

### Examples

```
d <- data.frame(pred = 1:4, truth = c(TRUE,FALSE,TRUE,TRUE))
roc <- build_ROC_curve(d$pred, d$truth)
add_ROC_derived_columns(roc, mean(d$truth))
```

---

```
as.character.sigr_statistic
```

*as.character*

---

### Description

as.character

### Usage

```
## S3 method for class 'sigr_statistic'
as.character(x, ...)
```

### Arguments

`x` sigr wrapper to print  
`...` extra arguments for `sigr::render`

**Value**

formatted string

**Examples**

```
as.character(wrapSignificance(1/300))
```

---

Bernoulli\_diff\_stat     *Compute the distribution of differences of replacement samples of two Binomial or Bernoulli experiments.*

---

**Description**

Assuming  $\max(n_A, n_B) \% \min(n_A, n_B) == 0$ : compute the distribution of differences of weighted sums between  $\max(1, n_B/n_A) * \text{sum}(a)$  and  $\max(1, n_A/n_B) * \text{sum}(b)$  where  $a$  is a 0/1 vector of length  $n_A$  with each item 1 with independent probability  $(k_A+k_B)/(n_A+n_B)$ , and  $b$  is a 0/1 vector of length  $n_B$  with each item 1 with independent probability  $(k_A+k_B)/(n_A+n_B)$ . Then return the significance of a direct two-sided test that the absolute value of this difference is at least as large as the `test_rate_difference` (if supplied) or the empirically observed rate difference  $\text{abs}(n_B * k_A - n_A * k_B) / (n_A * n_B)$ . The idea is: under this scaling differences in success rates between the two processes are easily observed as differences in counts returned by the scaled processes. The method can be used to get the exact probability of a given difference under the null hypothesis that both the A and B processes have the same success rate  $(k_A+k_B)/(n_A+n_B)$ . When  $n_A$  and  $n_B$  don't divide evenly into to each other two calculations are run with the larger process is alternately padded and truncated to look like a larger or smaller experiment that meets the above conditions. This gives us a good range of significances.

**Usage**

```
Bernoulli_diff_stat(kA, nA, kB, nB, test_rate_difference, common_rate)
```

**Arguments**

<code>kA</code>	number of A successes observed.
<code>nA</code>	number of A experiments.
<code>kB</code>	number of B successes observed.
<code>nB</code>	number of B experiments.
<code>test_rate_difference</code>	numeric, difference in rate of A-B to test. Note: it is best to specify this prior to looking at the data.
<code>common_rate</code>	rate numeric, assumed null-rate.

**Details**

Note the intent is that we are measuring the results of an A/B test with  $\max(n_A, n_B) \% \min(n_A, n_B) == 0$  (no padding needed), or  $\max(n_A, n_B) \gg \min(n_A, n_B)$  (padding is small effect).

The idea of converting a rate problem into a counting problem follows from reading Wald's *Sequential Analysis*.

For very small p-values the calculation is sensitive to rounding in the observed ratio-difference, as an arbitrarily small change in test-rate can move an entire set of observed differences in or out of the significance calculation.

**Value**

Bernoulli difference test statistic.

**Examples**

```
Bernoulli_diff_stat(2000, 5000, 100, 200)
Bernoulli_diff_stat(2000, 5000, 100, 200, 0.1)
Bernoulli_diff_stat(2000, 5000, 100, 199)
Bernoulli_diff_stat(2000, 5000, 100, 199, 0.1)
Bernoulli_diff_stat(100, 200, 2000, 5000)

# sigr adjusts experiment sizes when lengths
# don't divide into each other.
Bernoulli_diff_stat(100, 199, 2000, 5000)
Bernoulli_diff_stat(100, 199, 2000, 5000)$pValue
```

---

build_ROC_curve	<i>calculate ROC curve.</i>
-----------------	-----------------------------

---

**Description**

Based on: <https://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html>

**Usage**

```
build_ROC_curve(modelPredictions, yValues, ..., na.rm = FALSE, yTarget = TRUE)
```

**Arguments**

modelPredictions	numeric predictions (not empty)
yValues	truth values (not empty, same length as model predictions)
...	force later arguments to bind by name.
na.rm	logical, if TRUE remove NA values.
yTarget	value considered to be positive.

**Value**

the ROC graph of Score (model score), Sensitivity, and Specificity. Guaranteed to have the (0, 0) and (1, 1) (1-Specificity,Sensitivity) endpoints.

**Examples**

```
sigr::build_ROC_curve(1:4, c(TRUE,FALSE,TRUE,TRUE))
```

---

calcAUC	<i>calculate AUC.</i>
---------	-----------------------

---

**Description**

Based on: <https://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html>

**Usage**

```
calcAUC(modelPredictions, yValues, ..., na.rm = FALSE, yTarget = TRUE)
```

**Arguments**

modelPredictions	numeric predictions (not empty), ordered (either increasing or decreasing)
yValues	truth values (not empty, same length as model predictions)
...	force later arguments to bind by name.
na.rm	logical, if TRUE remove NA values.
yTarget	value considered to be positive.

**Value**

area under curve

**Examples**

```
sigr::calcAUC(1:4, c(TRUE,FALSE,TRUE,TRUE)) # should be 2/3
```

calcDeviance                    *Calculate deviance.*

---

**Description**

Calculate deviance.

**Usage**

```
calcDeviance(pred, y, na.rm = FALSE, eps = 1e-06)
```

**Arguments**

pred	numeric predictions
y	logical truth
na.rm	logical, if TRUE remove NA values
eps	numeric, smoothing term

**Value**

deviance

**Examples**

```
sigr::calcDeviance(1:4,c(TRUE,FALSE,TRUE,TRUE))
```

---

calcSSE                         *Calculate sum of squared error.*

---

**Description**

Calculate sum of squared error.

**Usage**

```
calcSSE(pred, y, na.rm = FALSE)
```

**Arguments**

pred	numeric predictions
y	numeric truth
na.rm	logical, if TRUE remove NA values



**Value**

sum of squared error

**Examples**

```
sigr:::calcSSE(1:4,c(1,0,1,1))
```

---

`estimateDifferenceZeroCrossing`

*Studentized estimate of how often a difference is below zero.*

---

**Description**

Studentized estimate of how often a difference is below zero.

**Usage**

```
estimateDifferenceZeroCrossing(resampledDiffs, na.rm = FALSE)
```

**Arguments**

`resampledDiffs` numeric vector resampled observations

`na.rm` logical, if TRUE remove NA values

**Value**

estimated probability of seeing a re-sampled difference below zero.

**Examples**

```
set.seed(2352)
resampledDiffs <- rnorm(10)+1
estimateDifferenceZeroCrossing(resampledDiffs)
```

---

find_area_q	<i>Find area matching polynomial curve.</i>
-------------	---

---

**Description**

Based on <https://win-vector.com/2020/09/13/why-working-with-auc-is-more-powerful-than-one-might-think>

**Usage**

```
find_area_q(area, ..., n_points = 101)
```

**Arguments**

area	area to match
...	not used, force later arguments to bind by name
n_points	how many points to use to estimate area.

**Value**

q that such that curve  $1 - (1 - (1 - \text{Specificity})^q)^{1/q}$  matches area

**Examples**

```
find_area_q(0.75)
```

---

find_AUC_q	<i>Find area matching polynomial curve.</i>
------------	---

---

**Description**

Based on <https://win-vector.com/2020/09/13/why-working-with-auc-is-more-powerful-than-one-might-think>

**Usage**

```
find_AUC_q(
  modelPredictions,
  yValues,
  ...,
  na.rm = FALSE,
  yTarget = TRUE,
  n_points = 101
)
```

**Arguments**

modelPredictions	numeric predictions (not empty), ordered (either increasing or decreasing)
yValues	truth values (not empty, same length as model predictions)
...	force later arguments to bind by name.
na.rm	logical, if TRUE remove NA values.
yTarget	value considered to be positive.
n_points	number of points to use in estimates.

**Value**

q that such that curve  $1 - (1 - (1 - \text{ideal\_roc}\$Specificity)^q)^{1/q}$  matches area

**Examples**

```
d <- data.frame(pred = 1:4, truth = c(TRUE,FALSE,TRUE,TRUE))
q <- find_AUC_q(d$pred, d$truth)
roc <- build_ROC_curve(d$pred, d$truth)
ideal_roc <- data.frame(Specificity = seq(0, 1, length.out = 101))
ideal_roc$Sensitivity <- sensitivity_from_specificity_q(ideal_roc$Specificity, q)
# library(ggplot2)
# ggplot(mapping = aes(x = 1 - Specificity, y = Sensitivity)) +
#   geom_line(data = roc, color = "DarkBlue") +
#   geom_line(data = ideal_roc, color = "Orange") +
#   theme(aspect.ratio=1) +
#   ggtitle("example actual and ideal curve")
```

---

find\_matching\_a1\_1b     *Find beta-1 shape parameters matching the conditional distributions.*

---

**Description**

Based on <https://journals.sagepub.com/doi/abs/10.1177/0272989X15582210>. Fits a Beta(a, 1) distribution on positive examples and an Beta(1, b) distribution on negative examples.

**Usage**

```
find_matching_a1_1b(
  modelPredictions,
  yValues,
  ...,
  yTarget = TRUE,
  step_size = 0.001
)
```

```

find_ROC_matching_ab1(
  modelPredictions,
  yValues,
  ...,
  yTarget = TRUE,
  step_size = 0.001
)

```

### Arguments

modelPredictions	numeric predictions (not empty), ordered (either increasing or decreasing)
yValues	truth values (not empty, same length as model predictions)
...	force later arguments to bind by name.
yTarget	value considered to be positive.
step_size	size of steps in curve drawing

### Value

beta curve shape parameters

### Examples

```

d <- rbind(
  data.frame(x = rbeta(1000, shape1 = 6, shape2 = 4), y = TRUE),
  data.frame(x = rbeta(1000, shape1 = 2, shape2 = 5), y = FALSE)
)
find_ROC_matching_ab1(modelPredictions = d$x, yValues = d$y)
# should be near
# shape1_pos shape2_pos shape1_neg shape2_neg      a      b
# 3.985017 1.000000 1.000000 1.746613 3.985017 1.746613
#
# # How to land what you want as variables
# unpack[a, b] <-
#   find_matching_a1_1b(modelPredictions = d$x, yValues = d$y)

```

---

find\_matching\_conditional\_betas

*Find beta shape parameters matching the conditional distributions.*

---

### Description

Based on <https://win-vector.com/2020/09/13/why-working-with-auc-is-more-powerful-than-one-might-think>  
 Used to find one beta distribution on positive examples, and another on negative examples.

**Usage**

```
find_matching_conditional_betas(modelPredictions, yValues, ..., yTarget = TRUE)
```

```
find_ROC_matching_ab(modelPredictions, yValues, ..., yTarget = TRUE)
```

**Arguments**

```
modelPredictions      numeric predictions (not empty), ordered (either increasing or decreasing)
yValues                truth values (not empty, same length as model predictions)
...                   force later arguments to bind by name.
yTarget               value considered to be positive.
```

**Value**

beta curve shape parameters

**Examples**

```
d <- rbind(
  data.frame(x = rbeta(1000, shape1 = 6, shape2 = 4), y = TRUE),
  data.frame(x = rbeta(1000, shape1 = 2, shape2 = 3), y = FALSE)
)
find_matching_conditional_betas(modelPredictions = d$x, yValues = d$y)
# should be near
# shape1_pos shape2_pos shape1_neg shape2_neg
# 6          4          2          3
#
# # How to land all as variables
# unpack[shape1_pos, shape2_pos, shape1_neg, shape2_neg] <-
#   find_ROC_matching_ab(modelPredictions = d$x, yValues = d$y)
```

---

fit\_beta\_shapes      *Fit beta parameters from data.*

---

**Description**

Fit shape1, shape2 using the method of moments.

**Usage**

```
fit_beta_shapes(x)
```

**Arguments**

```
x                    numeric predictions
```

**Value**

beta shape1, shape2 parameters in a named list

**Examples**

```
x <- rbeta(1000, shape1 = 3, shape2 = 5.5)
fit_beta_shapes(x) # should often be near [3, 5.5]
```

---

format.sigr\_statistic *Format*

---

**Description**

Format

**Usage**

```
## S3 method for class 'sigr_statistic'
format(x, ...)
```

**Arguments**

x	sigr wrapper to print
...	extra arguments for sigr::render

**Value**

formatted string

**Examples**

```
format(wrapSignificance(1/300))
```

---

getRenderingFormat	<i>Detect rendering format (using knitr).</i>
--------------------	---

---

**Description**

Detect rendering format (using knitr).

**Usage**

```
getRenderingFormat()
```

**Value**

rendering format

**Examples**

```
getRenderingFormat()
```

---

model_utility	<i>Estimate model utility</i>
---------------	-------------------------------

---

**Description**

Compute the utility of a model score on a classification data set. For each threshold of interest we compute the utility of the classification rule of taking all items with model score greater than or equal to the threshold. The user specifies the outcome (a binary classification target), a model score (numeric), and the utility values (positive, negative, or zero) of each case: true positives, false positives, true negatives, and false negatives. What is returned is a table of model thresholds and the total value of using this model score plus the given threshold as a classification rule. NA is used to mark a threshold where no rows are selected.

**Usage**

```
model_utility(  
  d,  
  model_name,  
  outcome_name,  
  ...,  
  outcome_target = TRUE,  
  true_positive_value_column_name = "true_positive_value",  
  false_positive_value_column_name = "false_positive_value",  
  true_negative_value_column_name = "true_negative_value",  
  false_negative_value_column_name = "false_negative_value"  
)
```

**Arguments**

d	A data.frame containing all data and outcome values.
model_name	Name of the column containing model predictions.
outcome_name	Name of the column containing the truth values.
...	Not used, forces later argument to be specified by name.
outcome_target	truth value considered to be TRUE.
true_positive_value_column_name	column name of per-row values of true positive cases. Only used on positive instances.
false_positive_value_column_name	column name of per-row values of false positive cases. Only used on negative instances.
true_negative_value_column_name	column name of per-row values of true negative cases. Only used on negative instances.
false_negative_value_column_name	column name of per-row values of false negative cases. Only used on positive instances.

**Details**

A worked example can be found here: <https://github.com/WinVector/sigr/blob/main/extras/UtilityExample.md>.

**Value**

data.frame of all threshold values.

**Examples**

```
d <- data.frame(
  predicted_probability = c(0, 0.5, 0.5, 0.5),
  made_purchase = c(FALSE, TRUE, FALSE, FALSE),
  false_positive_value = -5,    # acting on any predicted positive costs $5
  true_positive_value = 95,    # revenue on a true positive is $100 minus action cost
  true_negative_value = 0.001, # true negatives have no value in our application
                                # but just give ourselves a small reward for being right
  false_negative_value = -0.01 # adding a small notional tax for false negatives,
                                # don't want our competitor getting these accounts.
)

values <- model_utility(d, 'predicted_probability', 'made_purchase')
best_strategy <- values[values$total_value >= max(values$total_value), ][1, ]
t(best_strategy)
```

```
# a bigger example
```



```

d <- data.frame(
  predicted_probability = stats::runif(100),
  made_purchase = sample(c(FALSE, TRUE), replace = TRUE, size = 100),
  false_positive_value = -5, # acting on any predicted positive costs $5
  true_positive_value = 95, # revenue on a true positive is $100 minus action cost
  true_negative_value = 0.001, # true negatives have no value in our application
                                # but just give ourselves a small reward for being right
  false_negative_value = -0.01 # adding a small notional tax for false negatives,
                                # don't want our competitor getting these accounts.
)

values <- model_utility(d, 'predicted_probability', 'made_purchase')

# plot the estimated total utility as a function of threshold
plot(values$threshold, values$total_value)

best_strategy <- values[values$total_value >= max(values$total_value), ][1, ]
t(best_strategy)

# without utilities example

d <- data.frame(
  predicted_probability = c(0, 0.5, 0.5, 0.5),
  made_purchase = c(FALSE, TRUE, FALSE, FALSE))
model_utility(d, 'predicted_probability', 'made_purchase')

```

---

permTestAUC

*Perform AUC permutation test.*


---

## Description

Estimate significance of AUC by permutation test.

## Usage

```

permTestAUC(
  d,
  modelName,
  yName,
  yTarget = TRUE,
  ...,
  na.rm = FALSE,
  returnScores = FALSE,
  nrep = 100,
  parallelCluster = NULL
)

```

**Arguments**

**d** data.frame  
**modelName** character model column name  
**yName** character outcome column name  
**yTarget** target to match to y  
**...** extra arguments (not used)  
**na.rm** logical, if TRUE remove NA values  
**returnScores** logical if TRUE return detailed permutedScores  
**nrep** number of permutation repetitions to estimate p values.  
**parallelCluster** (optional) a cluster object created by package parallel or package snow

**Value**

AUC statistic

**Examples**

```

set.seed(25325)
d <- data.frame(x1=c(1,2,3,4,5,6,7,7),
                y=c(FALSE,TRUE,FALSE,FALSE,
                    TRUE,TRUE,FALSE,TRUE))
permTestAUC(d,'x1','y',TRUE)

```

---

permutationScoreModel	<i>Empirical cance</i>	<i>permutation of</i>	<i>test of</i>	<i>signifi- &gt;=</i>
	<i>scoreFn(modelValues,yValues)</i>			
	<i>scoreFn(modelValues,perm(yValues)).</i>			

---

**Description**

Treat permutation re-samples as similar to bootstrap replications.

**Usage**

```

permutationScoreModel(
  modelValues,
  yValues,
  scoreFn,
  ...,
  na.rm = FALSE,
  returnScores = FALSE,
  nRep = 100,
  parallelCluster = NULL
)

```

**Arguments**

modelValues	numeric array of predictions.
yValues	numeric/logical array of outcomes, dependent, or truth values
scoreFn	function with signature scoreFn(modelValues,yValues) returning scalar numeric score.
...	not used, forces later arguments to be bound by name
na.rm	logical, if TRUE remove NA values
returnScores	logical if TRUE return detailed permutedScores
nRep	integer number of repetitions to perform
parallelCluster	optional snow-style parallel cluster.

**Value**

summaries

**Examples**

```
set.seed(25325)
y <- 1:5
m <- c(1,1,2,2,2)
cor.test(m,y,alternative='greater')
f <- function(modelValues,yValues) cor(modelValues,yValues)
permutationScoreModel(m,y,f)
```

---

print.sigr\_statistic *Print*

---

**Description**

Print

**Usage**

```
## S3 method for class 'sigr_statistic'
print(x, ...)
```

**Arguments**

x	sigr wrapper to print
...	extra arguments for sigr::render and print

**Value**

formatted string

**Examples**

```
print(wrapSignificance(1/300))
```

---

render	<i>Format summary roughly in "APA Style" ( American Psychological Association ).</i>
--------	--

---

**Description**

Format summary roughly in "APA Style" ( American Psychological Association ).

**Usage**

```
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	sigr summary statistic
...	extra arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

**See Also**

[render.sigr\\_significance](#), [render.sigr\\_ftest](#)

---

```
render.sigr_aucpairtest
    Format an AUC-test (quality of a probability score)
```

---

**Description**

Format an AUC-test (quality of a probability score)

**Usage**

```
## S3 method for class 'sigr_aucpairtest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped AUC test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

```
render.sigr_aucpermttest
    Format an AUC-test (quality of a probability score)
```

---

**Description**

Format an AUC-test (quality of a probability score)

**Usage**

```
## S3 method for class 'sigr_aucpermttest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped AUC test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

```
render.sigr_aucresamptest
```

*Format an AUC-test (quality of a probability score)*

---

**Description**

Format an AUC-test (quality of a probability score)

**Usage**

```
## S3 method for class 'sigr_aucresamptest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped AUC test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

```
render.sigr_Bernoulli_diff_test
```

*Format sigr\_Bernoulli\_diff\_test (test of difference of Bernoulli processes).*

---

**Description**

Format sigr\_Bernoulli\_diff\_test (test of difference of Bernoulli processes).

**Usage**

```
## S3 method for class 'sigr_Bernoulli_diff_test'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped cor.test.
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

**Examples**

```
Bernoulli_diff_stat(2000, 5000, 100, 200)
Bernoulli_diff_stat(2000, 5000, 100, 200, 0.1)
Bernoulli_diff_stat(2000, 5000, 100, 199)
Bernoulli_diff_stat(2000, 5000, 100, 199, 0.1)
```

---

render.sigr\_binomtest *Format binom.test (test of rate of a Binomial/Bernoulli experiment).*

---

**Description**

Format binom.test (test of rate of a Binomial/Bernoulli experiment).

**Usage**

```
## S3 method for class 'sigr_binomtest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped binom.test.
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string



## Examples

```
bt <- binom.test(7, 10, 0.5)
wrapBinomTest(bt)
```

---

render.sigr\_chisqtest *Format a chi-square test (quality of categorical prediction)*

---

## Description

Format a chi-square test (quality of categorical prediction)

## Usage

```
## S3 method for class 'sigr_chisqtest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

## Arguments

statistic	wrapped T-test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

## Value

formatted string

---

render.sigr\_cohend      *Format Cohen-D (effect size between groups)*

---

### Description

Format Cohen-D (effect size between groups)

### Usage

```
## S3 method for class 'sigr_cohend'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 1,
  pSmallCutoff = 0
)
```

### Arguments

statistic	CohenD-approximation
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

### Value

formatted string

---

render.sigr\_cortest      *Format cor.test (test of liner correlation).*

---

### Description

Format cor.test (test of liner correlation).

**Usage**

```
## S3 method for class 'sigr_cortest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped cor.test.
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
ct <- cor.test(d$x,d$y)
wrapCorTest(ct)
```

---

render.sigr\_emptest    *Format an empirical test (quality of categorical prediction)*

---

**Description**

Format an empirical test (quality of categorical prediction)

**Usage**

```
## S3 method for class 'sigr_emptest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped T-test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

```
render.sigr_fishertest
```

*Format fisher.test (test of categorical independence).*

---

**Description**

Format fisher.test (test of categorical independence).

**Usage**

```
## S3 method for class 'sigr_fishertest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped Fisher test
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x=c('b','a','a','a','b','b','b'),
               y=c('1','1','1','2','2','2','2'))
ft <- fisher.test(table(d))
wrapFisherTest(ft)
```

---

render.sigr\_ftest      *Format an F-test*

---

**Description**

Format an F-test

**Usage**

```
## S3 method for class 'sigr_ftest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

render.sigr\_permtest *Format an empirical test (quality of categorical prediction)*

---

**Description**

Format an empirical test (quality of categorical prediction)

**Usage**

```
## S3 method for class 'sigr_permtest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped T-test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summary.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

render.sigr\_pwr\_hstest *Format a pwr-test*

---

**Description**

Format a pwr-test

**Usage**

```
## S3 method for class 'sigr_pwr_hstest'  
render(  
  statistic,  
  ...,  
  format,  
  statDigits = 4,  
  sigDigits = 4,  
  pLargeCutoff = 1,  
  pSmallCutoff = 1e-05  
)
```

**Arguments**

statistic	wrapped test from pwr package
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

render.sigr\_significance  
*Format a significance*

---

**Description**

Format a significance

**Usage**

```
## S3 method for class 'sigr_significance'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped significance
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries (not used in significance reports).
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

**Examples**

```
cat(render(wrapSignificance(1/300), format='html'))
```

---

render.sigr\_tinterval *Format a Student-T tolerance-style interval around an estimate of a mean.*

---

**Description**

Report sample size (n), sample mean, bias-corrected standard deviation estimate (assuming normality, using a chi-square distribution correction from [https://en.wikipedia.org/wiki/Unbiased\\_estimation\\_of\\_standard\\_deviation#Bias\\_correction](https://en.wikipedia.org/wiki/Unbiased_estimation_of_standard_deviation#Bias_correction)), and a Student t-test tolerance-style confidence interval.



**Usage**

```
## S3 method for class 'sigr_tinterval'  
render(  
  statistic,  
  ...,  
  format,  
  statDigits = 4,  
  sigDigits = 4,  
  pLargeCutoff = 0.05,  
  pSmallCutoff = 1e-05  
)
```

**Arguments**

statistic	wrapped TInterval.
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

**Examples**

```
set.seed(2018)  
d <- rnorm(100) + 3.2  
TInterval(d)
```

---

render.sigr\_ttest      *Format a T-test (difference in means by group)*

---

**Description**

Format a T-test (difference in means by group)

**Usage**

```
## S3 method for class 'sigr_ttest'
render(
  statistic,
  ...,
  format,
  statDigits = 4,
  sigDigits = 4,
  pLargeCutoff = 0.05,
  pSmallCutoff = 1e-05
)
```

**Arguments**

statistic	wrapped T-test
...	not used, force use of named binding for later arguments
format	if set the format to return ("html", "latex", "markdown", "ascii")
statDigits	integer number of digits to show in summaries.
sigDigits	integer number of digits to show in significances.
pLargeCutoff	value to declare non-significance at or above.
pSmallCutoff	smallest value to print

**Value**

formatted string

---

resampleScoreModel	<i>Studentized</i>	<i>bootstrap</i>	<i>variance</i>	<i>estimate</i>	<i>for</i>
	<i>scoreFn(yValues,modelValues).</i>				

---

**Description**

Studentized bootstrap variance estimate for scoreFn(yValues,modelValues).

**Usage**

```
resampleScoreModel(
  modelValues,
  yValues,
  scoreFn,
  ...,
  na.rm = FALSE,
  returnScores = FALSE,
  nRep = 100,
  parallelCluster = NULL
)
```

**Arguments**

modelValues	numeric array of predictions (model to test).
yValues	numeric/logical array of outcomes, dependent, or truth values
scoreFn	function with signature <code>scoreFn(modelValues,yValues)</code> returning scalar numeric score.
...	not used, forces later arguments to be bound by name
na.rm	logical, if TRUE remove NA values
returnScores	logical if TRUE return detailed resampledScores
nRep	integer number of repetitions to perform
parallelCluster	optional snow-style parallel cluster.

**Value**

summaries

**Examples**

```

set.seed(25325)
y <- 1:5
m1 <- c(1,1,2,2,2)
cor.test(m1,y,alternative='greater')
f <- function(modelValues,yValues) {
  if((sd(modelValues)<=0)|| (sd(yValues)<=0)) {
    return(0)
  }
  cor(modelValues,yValues)
}
s <- sigr::resampleScoreModel(m1,y,f)
print(s)
z <- (s$observedScore-0)/s$sd # should check size of z relative to bias!
pValue <- pt(z,df=length(y)-2,lower.tail=FALSE)
pValue

```

---

resampleScoreModelPair

*Studentized bootstrap test of strength of  
scoreFn(yValues,modelValues) > scoreFn(yValues,modelValues).*

---

**Description**

Studentized bootstrap test of strength of `scoreFn(yValues,modelValues) > scoreFn(yValues,modelValues)` sampled with replacement.

**Usage**

```
resampleScoreModelPair(
  model1Values,
  model2Values,
  yValues,
  scoreFn,
  ...,
  na.rm = FALSE,
  returnScores = FALSE,
  nRep = 100,
  parallelCluster = NULL,
  sameSample = FALSE
)
```

**Arguments**

model1Values	numeric array of predictions (model to test).
model2Values	numeric array of predictions (reference model).
yValues	numeric/logical array of outcomes, dependent, or truth values
scoreFn	function with signature scoreFn(modelValues,yValues) returning scalar numeric score.
...	not used, forces later arguments to be bound by name.
na.rm	logical, if TRUE remove NA values
returnScores	logical if TRUE return detailed resampledScores.
nRep	integer number of repetitions to perform.
parallelCluster	optional snow-style parallel cluster.
sameSample	logical if TRUE use the same sample in computing both scores during bootstrap replication (else use independent samples).

**Details**

True confidence intervals are harder to get right (see "An Introduction to the Bootstrap", Bradely Efron, and Robert J. Tibshirani, Chapman & Hall/CRC, 1993.), but we will settle for simple p-value estimates.

**Value**

summaries

**Examples**

```
set.seed(25325)
y <- 1:5
m1 <- c(1,1,2,2,2)
```

```

m2 <- c(1,1,1,1,2)
cor(m1,y)
cor(m2,y)
f <- function(modelValues,yValues) {
  if((sd(modelValues)<=0)||(sd(yValues)<=0)) {
    return(0)
  }
  cor(modelValues,yValues)
}
resampleScoreModelPair(m1,m2,y,f)

```

---

resampleTestAUC	<i>Wrap AUC resampling test results.</i>
-----------------	--

---

### Description

Estimate significance of AUC by resampling test.

### Usage

```

resampleTestAUC(
  d,
  modelName,
  yName,
  yTarget = TRUE,
  ...,
  na.rm = FALSE,
  returnScores = FALSE,
  nrep = 100,
  parallelCluster = NULL
)

```

### Arguments

d	data.frame
modelName	character model column name
yName	character outcome column name
yTarget	target to match to y
...	extra arguments (not used)
na.rm	logical, if TRUE remove NA values
returnScores	logical if TRUE return detailed resampledScores.
nrep	number of permutation repetitions to estimate p values.
parallelCluster	(optional) a cluster object created by package parallel or package snow.

**Value**

AUC statistic

**Examples**

```
set.seed(25325)
d <- data.frame(x1=c(1,2,3,4,5,6,7,7),
                y=c(FALSE,TRUE,FALSE,FALSE,
                    TRUE,TRUE,FALSE,TRUE))
resampleTestAUC(d, 'x1', 'y', TRUE)
```

---

sensitivity\_and\_specificity\_s12p12n

*Compute the shape1\_pos, shape2\_pos, shape1\_neg, shape2\_neg graph.*

---

**Description**

Compute specificity and sensitivity given specificity and model fit parameters.

**Usage**

```
sensitivity_and_specificity_s12p12n(
  Score,
  ...,
  shape1_pos,
  shape2_pos,
  shape1_neg,
  shape2_neg
)
```

**Arguments**

Score	vector of sensitivities to evaluate
...	force later arguments to bind by name.
shape1_pos	beta shape1 parameter for positive examples
shape2_pos	beta shape2 parameter for positive examples
shape1_neg	beta shape1 parameter for negative examples
shape2_neg	beta shape1 parameter for negative examples

**Value**

Score, Specificity and Sensitivity data frame

**Examples**

```

library(wrapr)

empirical_data <- rbind(
  data.frame(
    Score = rbeta(1000, shape1 = 3, shape2 = 2),
    y = TRUE),
  data.frame(
    Score = rbeta(1000, shape1 = 5, shape2 = 4),
    y = FALSE)
)

unpack[shape1_pos = shape1, shape2_pos = shape2] <-
  fit_beta_shapes(empirical_data$Score[empirical_data$y])

shape1_pos
shape2_pos

unpack[shape1_neg = shape1, shape2_neg = shape2] <-
  fit_beta_shapes(empirical_data$Score[!empirical_data$y])

shape1_neg
shape2_neg

ideal_roc <- sensitivity_and_specificity_s12p12n(
  seq(0, 1, 0.1),
  shape1_pos = shape1_pos,
  shape1_neg = shape1_neg,
  shape2_pos = shape2_pos,
  shape2_neg = shape2_neg)

empirical_roc <- build_ROC_curve(
  modelPredictions = empirical_data$Score,
  yValues = empirical_data$y
)

# # should look very similar
# library(ggplot2)
# ggplot(mapping = aes(x = 1 - Specificity, y = Sensitivity)) +
#   geom_line(data = empirical_roc, color='DarkBlue') +
#   geom_line(data = ideal_roc, color = 'Orange')

```

---

sensitivity\_from\_specificity\_q

*Compute the q-graph.*

---

**Description**

Based on: <https://blog.revolutionanalytics.com/2016/08/roc-curves-in-two-lines-of-code.html>

**Usage**

```
sensitivity_from_specificity_q(Specificity, q)
```

**Arguments**

Specificity      vector of sensitivities to evaluate  
q                shape parameter for  $1 - (1 - (1 - \text{Specificity})^q)^{1/q}$

**Value**

Sensitivity

**Examples**

```
sensitivity_from_specificity_q(seq(0, 1, 0.1), 0.61)
```

---

sigr

*sigr: Format Significance Summaries for Reports*

---

**Description**

Succinctly format significance summaries of various models and tests (F-test, Chi-Sq-test, Fisher-test, T-test, and rank-significance). The main purpose is unified reporting and planning of experimental results, working around issue such as the difficulty of extracting model summary facts (such as with 'lm'/'glm'). This package also includes empirical tests, such as bootstrap estimates.

**Details**

To learn more about sigr, please start with the vignette: `vignette('sigrFormatting', 'sigr')`



---

testAUCpair	<i>Test AUC pair results.</i>
-------------	-------------------------------

---

### Description

Estimate significance of difference in two AUCs by resampling.

### Usage

```
testAUCpair(  
  d,  
  model1Name,  
  model2Name,  
  yName,  
  yTarget = TRUE,  
  ...,  
  na.rm = FALSE,  
  returnScores = FALSE,  
  nrep = 100,  
  parallelCluster = NULL  
)
```

### Arguments

d	data.frame
model1Name	character model 1 column name
model2Name	character model 2 column name
yName	character outcome column name
yTarget	target to match to y
...	extra arguments (not used)
na.rm	logical, if TRUE remove NA values
returnScores	logical if TRUE return detailed resampledScores
nrep	number of re-sample repetition to estimate p value.
parallelCluster	(optional) a cluster object created by package parallel or package snow

### Value

AUC pair test

**Examples**

```
set.seed(25325)
d <- data.frame(x1=c(1,2,3,4,5,6,7,7),
                x2=1,
                y=c(FALSE,TRUE,FALSE,FALSE,
                    TRUE,TRUE,FALSE,TRUE))
testAUCpair(d,'x1','x2','y',TRUE)
```

---

TInterval

*Wrap TInterval (test of Binomial/Bernoulli rate).*


---

**Description**

Wrap TInterval (test of Binomial/Bernoulli rate).

**Usage**

```
TInterval(x, ...)
```

**Arguments**

x                    numeric, data.frame or test.  
...                    extra arguments

**See Also**

[TIntervals](#), [TInterval.numeric](#), [TInterval.data.frame](#)

---

TInterval.data.frame

*Student-T tolerance-style interval around an estimate of a mean from a data.frame.*


---

**Description**

Student-T tolerance-style interval around an estimate of a mean from a data.frame.

**Usage**

```
## S3 method for class 'data.frame'
TInterval(x, ColumnName, ..., conf.level = 0.95, na.rm = FALSE)
```

**Arguments**

x	data.frame
ColumnName	character name of measurement column
...	extra arguments passed to TInterval
conf.level	confidence level to draw interval
na.rm	logical, if TRUE remove NA values

**Value**

wrapped stat

**See Also**

[TInterval](#), [TIntervals](#), [TInterval.numeric](#), [TInterval.data.frame](#)

**Examples**

```
set.seed(2018)
d <- data.frame(x = rnorm(100) + 3.2)
TInterval(d, "x")
```

---

TInterval.numeric	<i>Student-T tolerance-style interval around an estimate of a mean from observations.</i>
-------------------	---

---

**Description**

Student-T tolerance-style interval around an estimate of a mean from observations.

**Usage**

```
## S3 method for class 'numeric'
TInterval(x, ..., conf.level = 0.95, na.rm = FALSE)
```

**Arguments**

x	logical, vector of observations.
...	extra arguments passed to TInterval
conf.level	confidence level to draw interval
na.rm	logical, if TRUE remove NA values

**Value**

wrapped stat

**See Also**

[TInterval](#), [TIntervals](#), [TInterval.numeric](#), [TInterval.data.frame](#)

**Examples**

```
set.seed(2018)
d <- rnorm(100) + 3.2
TInterval(d)
```

---

TIntervals	<i>Student-T tolerance-style interval around an estimate of a mean from summary.</i>
------------	--

---

**Description**

Student-T tolerance-style interval around an estimate of a mean from summary.

**Usage**

```
TIntervals(
  sample_size,
  sample_mean,
  sample_var,
  ...,
  nNA = 0,
  conf.level = 0.95
)
```

**Arguments**

sample_size	numeric scalar integer, size of sample.
sample_mean	numeric scalar, mean of sample.
sample_var	numeric scalar, variance of sample (Bessel-corrected).
...	extra arguments passed to TInterval.
nNA	number of NAs seen.
conf.level	confidence level to draw interval

**Value**

wrapped stat

**See Also**

[TInterval](#), [TIntervals](#), [TInterval.numeric](#), [TInterval.data.frame](#)

## Examples

```
set.seed(2018)
d <- rnorm(100) + 3.2
TIntervalS(length(d), mean(d), stats::var(d))
```

---

wrapBinomTest	<i>Wrap binom.test (test of Binomial/Bernoulli rate).</i>
---------------	---

---

## Description

Wrap binom.test (test of Binomial/Bernoulli rate).

## Usage

```
wrapBinomTest(x, ...)
```

## Arguments

x	numeric, data.frame or test.
...	extra arguments

## See Also

[wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#),  
[wrapBinomTest.data.frame](#)

---

wrapBinomTest.data.frame	<i>Wrap binom.test (test of Binomial/Bernoulli rate).</i>
--------------------------	---

---

## Description

Wrap binom.test (test of Binomial/Bernoulli rate).

## Usage

```
## S3 method for class 'data.frame'
wrapBinomTest(
  x,
  ColumnName,
  SuccessValue = TRUE,
  ...,
  p = NA,
```

```

alternative = c("two.sided", "less", "greater"),
conf.level = 0.95,
na.rm = FALSE
)

```

### Arguments

x	data.frame
ColumnName	character name of measurement column
SuccessValue	value considered a success (positive)
...	extra arguments passed to binom.test
p	number, hypothesized probability of success.
alternative	passed to <a href="#">binom.test</a>
conf.level	passed to <a href="#">binom.test</a>
na.rm	logical, if TRUE remove NA values

### Value

wrapped stat

### See Also

[wrapBinomTest](#), [wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#), [wrapBinomTest.data.frame](#)

### Examples

```

d <- data.frame(x = c(rep(0, 3), rep(1, 7)))
wrapBinomTest(d, "x", 1, p = 0.5)
d <- data.frame(x = c(rep(0, 15), rep(1, 35)))
wrapBinomTest(d, "x", 1, p = 0.5)

```

---

wrapBinomTest.htest    *Wrap binom.test (test of Binomial/Bernoulli rate).*

---

### Description

Wrap binom.test (test of Binomial/Bernoulli rate).

### Usage

```

## S3 method for class 'htest'
wrapBinomTest(x, ...)

```

**Arguments**

x                    binom.test result  
 ...                  not used, just for argument compatibility

**Value**

wrapped stat

**See Also**

[wrapBinomTest](#), [wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#),  
[wrapBinomTest.data.frame](#)

**Examples**

```
bt <- binom.test(7, 10, 0.5)
wrapBinomTest(bt)
```

---

wrapBinomTest.logical    *Wrap binom.test (test of Binomial/Bernoulli rate).*

---

**Description**

Wrap binom.test (test of Binomial/Bernoulli rate).

**Usage**

```
## S3 method for class 'logical'
wrapBinomTest(
  x,
  ...,
  p = NA,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  na.rm = FALSE
)
```

**Arguments**

x                    logical, vector of trials.  
 ...                  extra arguments passed to binom.test  
 p                    number, hypothesized probability of success.  
 alternative        passed to [binom.test](#)  
 conf.level        passed to [binom.test](#)  
 na.rm              logical, if TRUE remove NA values

**Value**

wrapped stat

**See Also**

[wrapBinomTest](#), [wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#), [wrapBinomTest.data.frame](#)

**Examples**

```
x = c(rep(FALSE, 3), rep(TRUE, 7))
wrapBinomTest(x)
x = c(rep(FALSE, 15), rep(TRUE, 35))
wrapBinomTest(x)
```

---

wrapBinomTest.numeric *Wrap binom.test (test of Binomial/Bernoulli rate).*

---

**Description**

Wrap binom.test (test of Binomial/Bernoulli rate).

**Usage**

```
## S3 method for class 'numeric'
wrapBinomTest(
  x,
  SuccessValue = TRUE,
  ...,
  p = NA,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95,
  na.rm = FALSE
)
```

**Arguments**

x	numeric, vector of trials.
SuccessValue	value considered a success (positive)
...	extra arguments passed to binom.test
p	number, hypothesized probability of success.
alternative	passed to <a href="#">binom.test</a>
conf.level	passed to <a href="#">binom.test</a>
na.rm	logical, if TRUE remove NA values



**Value**

wrapped stat

**See Also**

[wrapBinomTest](#), [wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#), [wrapBinomTest.data.frame](#)

**Examples**

```
x = c(rep(0, 3), rep(1, 7))
wrapBinomTest(x, 1)
x = c(rep(0, 15), rep(1, 35))
wrapBinomTest(x, 1)
```

---

wrapBinomTestS

*Wrap binom.test (test of Binomial/Bernoulli rate) from summary.*

---

**Description**

Wrap binom.test (test of Binomial/Bernoulli rate) from summary.

**Usage**

```
wrapBinomTestS(
  x,
  n,
  ...,
  p = NA,
  alternative = c("two.sided", "less", "greater"),
  conf.level = 0.95
)
```

**Arguments**

x	numeric scalar, number of successes.
n	numeric scalar, number of trials.
...	extra arguments passed to binom.test
p	number, hypothesized probability of success.
alternative	passed to <a href="#">binom.test</a>
conf.level	passed to <a href="#">binom.test</a>

**Value**

wrapped stat

**See Also**

[wrapBinomTest](#), [wrapBinomTest.htest](#), [wrapBinomTestS](#), [wrapBinomTest.logical](#), [wrapBinomTest.numeric](#), [wrapBinomTest.data.frame](#)

**Examples**

```
wrapBinomTestS(3, 7, p = 0.5)
wrapBinomTestS(300, 700, p = 0.5)
```

---

wrapChiSqTest	<i>Wrap quality of a categorical prediction roughly in "APA Style" ( American Psychological Association ).</i>
---------------	--

---

**Description**

Wrap quality of a categorical prediction roughly in "APA Style" ( American Psychological Association ).

**Usage**

```
wrapChiSqTest(x, ...)
```

**Arguments**

x	numeric, data.frame or lm where to get model or data to score.
...	extra arguments

**See Also**

[wrapChiSqTestImpl](#), [wrapChiSqTest.glm](#), and [wrapChiSqTest.data.frame](#)

---

wrapChiSqTest.anova     *Format ChiSqTest from anova of logistic model.*

---

## Description

Format ChiSqTest from anova of logistic model.

## Usage

```
## S3 method for class 'anova'  
wrapChiSqTest(x, ...)
```

## Arguments

x	result from stats::anova(stats::glm(family=binomial))
...	extra arguments (not used)

## Value

list of formatted string and fields

## Examples

```
d <- data.frame(x1= c(1,2,3,4,5,6,7,7),  
               x2= c(1,0,3,0,5,0,7,0),  
               y= c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE))  
model <- glm(y~x1+x2, data=d, family=binomial)  
summary(model)  
render(wrapChiSqTest(model),  
       pLargeCutoff=1, format='ascii')  
anov <- anova(model)  
print(anov)  
lapply(sigr::wrapChiSqTest(anov),  
       function(ti) {  
         sigr::render(ti,  
                      pLargeCutoff= 1,  
                      pSmallCutoff= 0,  
                      statDigits=4,  
                      sigDigits=4,  
                      format='ascii')  
       })
```

---

```
wrapChiSqTest.data.frame
```

*Format ChiSqTest from data.*

---

## Description

Format ChiSqTest from data.

## Usage

```
## S3 method for class 'data.frame'
wrapChiSqTest(
  x,
  predictionColumnName,
  yColumnName,
  ...,
  yTarget = TRUE,
  nParameters = 1,
  meany = mean(x[[yColumnName]] == yTarget),
  na.rm = FALSE
)
```

## Arguments

<code>x</code>	data frame containing columns to compare
<code>predictionColumnName</code>	character name of prediction column
<code>yColumnName</code>	character name of column containing dependent variable
<code>...</code>	extra arguments (not used)
<code>yTarget</code>	y value to consider positive
<code>nParameters</code>	number of variables in model
<code>meany</code>	(optional) mean of y
<code>na.rm</code>	logical, if TRUE remove NA values

## Value

wrapped test

## Examples

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE))
model <- glm(y~x, data=d, family=binomial)
summary(model)
d$pred <- predict(model,type='response',newdata=d)
```

```
render(wrapChiSqTest(d, 'pred', 'y'), pLargeCutoff=1)
```

---

```
wrapChiSqTest.glm      Format ChiSqTest from model.
```

---

### Description

Format ChiSqTest from model.

### Usage

```
## S3 method for class 'glm'
wrapChiSqTest(x, ...)
```

### Arguments

```
x          glm logistic regression model (glm(family=binomial))
...        extra arguments (not used)
```

### Value

wrapped test

### Examples

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
                y=c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE))
model <- glm(y~x,data=d,family=binomial)
summary(model)
render(wrapChiSqTest(model),pLargeCutoff=1,format='ascii')
```

---

```
wrapChiSqTest.summary.glm
      Format ChiSqTest from model summary.
```

---

### Description

Format ChiSqTest from model summary.

### Usage

```
## S3 method for class 'summary.glm'
wrapChiSqTest(x, ...)
```

**Arguments**

x                   summary(glm(family=binomial)) object.  
 ...                extra arguments (not used)

**Value**

wrapped test

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
  y=c(TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,FALSE))
model <- glm(y~x,data=d,family=binomial)
sum <- summary(model)
render(wrapChiSqTest(sum),pLargeCutoff=1,format='ascii')
```

---

wrapChiSqTestImpl	<i>Format quality of a logistic regression roughly in "APA Style" ( American Psychological Association ).</i>
-------------------	---

---

**Description**

Format quality of a logistic regression roughly in "APA Style" ( American Psychological Association ).

**Usage**

```
wrapChiSqTestImpl(df.null, df.residual, null.deviance, deviance)
```

**Arguments**

df.null            null degrees of freedom.  
 df.residual       residual degrees of freedom.  
 null.deviance     null deviance  
 deviance          residual deviance

**Value**

wrapped statistic

**Examples**

```
wrapChiSqTestImpl(df.null=7,df.residual=6,
  null.deviance=11.09035,deviance=10.83726)
```

---

wrapCohenD	<i>Wrap Cohen's D (effect size between groups).</i>
------------	---

---

**Description**

Wrap Cohen's D (effect size between groups).

**Usage**

```
wrapCohenD(x, ...)
```

**Arguments**

x	numeric, data.frame or test.
...	extra arguments

**See Also**

[wrapCohenD.data.frame](#)

---

wrapCohenD.data.frame	<i>Wrap Cohen's D (effect size between groups).</i>
-----------------------	---

---

**Description**

Wrap Cohen's D (effect size between groups).

**Usage**

```
## S3 method for class 'data.frame'
wrapCohenD(x, Column1Name, Column2Name, ..., na.rm = FALSE)
```

**Arguments**

x	data.frame
Column1Name	character column 1 name
Column2Name	character column 2 name
...	extra arguments (not used)
na.rm	if TRUE remove NAs

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x = c(1,1,2,2,3,3,4,4),
               y = c(1,2,3,4,5,6,7,7))
render(wrapCohenD(d, 'x', 'y'))
```

---

wrapCohenD.numeric      *Wrap Cohen's D (effect size between groups).*

---

**Description**

Wrap Cohen's D (effect size between groups).

**Usage**

```
## S3 method for class 'numeric'
wrapCohenD(x, treatment, ..., na.rm = FALSE)
```

**Arguments**

x	numeric reference or control measurements
treatment	numeric treatment or group-2 measurements
...	extra arguments (not used)
na.rm	if TRUE remove NAs

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x = c(1,1,2,2,3,3,4,4),
               y = c(1,2,3,4,5,6,7,7))
render(wrapCohenD(d$x, d$y))
```



---

wrapCorTest	<i>Wrap cor.test (test of liner correlation).</i>
-------------	---

---

**Description**

Wrap cor.test (test of liner correlation).

**Usage**

```
wrapCorTest(x, ...)
```

**Arguments**

x	numeric, data.frame or test.
...	extra arguments

**See Also**

[wrapCorTest.htest](#), and [wrapCorTest.data.frame](#)

---

wrapCorTest.data.frame	<i>Wrap cor.test (test of liner correlation).</i>
------------------------	---

---

**Description**

Wrap cor.test (test of liner correlation).

**Usage**

```
## S3 method for class 'data.frame'
wrapCorTest(
  x,
  Column1Name,
  Column2Name,
  ...,
  alternative = c("two.sided", "less", "greater"),
  method = c("pearson", "kendall", "spearman"),
  exact = NULL,
  conf.level = 0.95,
  continuity = FALSE,
  na.rm = FALSE
)
```

**Arguments**

x	data.frame
Column1Name	character column 1 name
Column2Name	character column 2 name
...	extra arguments passed to cor.test
alternative	passed to <code>cor.test</code>
method	passed to <code>cor.test</code>
exact	passed to <code>cor.test</code>
conf.level	passed to <code>cor.test</code>
continuity	passed to <code>cor.test</code>
na.rm	logical, if TRUE remove NA values

**Value**

wrapped stat

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
                y=c(1,1,2,2,3,3,4,4))
wrapCorTest(d, 'x', 'y')
```

---

wrapCorTest.htest      *Wrap cor.test (test of liner correlation).*

---

**Description**

Wrap cor.test (test of liner correlation).

**Usage**

```
## S3 method for class 'htest'
wrapCorTest(x, ...)
```

**Arguments**

x	cor.test result
...	extra arguments (not used)

**Value**

wrapped stat

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
ct <- cor.test(d$x,d$y)
wrapCorTest(ct)
```

---

wrapFisherTest	<i>Wrap fisher.test (test of categorical independence).</i>
----------------	---

---

**Description**

Wrap fisher.test (test of categorical independence).

**Usage**

```
wrapFisherTest(x, ...)
```

**Arguments**

x	numeric, data.frame or test.
...	extra arguments

**See Also**

[wrapFisherTest.htest](#), and [wrapFisherTest.data.frame](#)

---

wrapFisherTest.data.frame	<i>Wrap fisher.test (test of categorical independence).</i>
---------------------------	---

---

**Description**

Wrap fisher.test (test of categorical independence).

**Usage**

```
## S3 method for class 'data.frame'
wrapFisherTest(
  x,
  Column1Name,
  Column2Name,
  ...,
  na.rm = FALSE,
  workspace = 2e+05,
  hybrid = FALSE,
  control = list(),
  or = 1,
  alternative = "two.sided",
  conf.int = TRUE,
  conf.level = 0.95,
  simulate.p.value = FALSE,
  B = 2000
)
```

**Arguments**

x	data.frame
Column1Name	character column 1 name
Column2Name	character column 2 name
...	extra arguments (not used)
na.rm	logical, if TRUE remove NA values
workspace	passed to <a href="#">fisher.test</a>
hybrid	passed to <a href="#">fisher.test</a>
control	passed to <a href="#">fisher.test</a>
or	passed to <a href="#">fisher.test</a>
alternative	passed to <a href="#">fisher.test</a>
conf.int	passed to <a href="#">fisher.test</a>
conf.level	passed to <a href="#">fisher.test</a>
simulate.p.value	passed to <a href="#">fisher.test</a>
B	passed to <a href="#">fisher.test</a>

**Value**

wrapped test.

## Examples

```
d <- data.frame(x=c('b','a','a','a','b','b','b'),
               y=c('1','1','1','2','2','2','2'))
wrapFisherTest(d, 'x', 'y')
```

---

wrapFisherTest.htest    *Wrap fisher.test (test of categorical independence).*

---

## Description

Wrap fisher.test (test of categorical independence).

## Usage

```
## S3 method for class 'htest'
wrapFisherTest(x, ...)
```

## Arguments

x	fisher.test result
...	extra arguments (not used)

## Value

wrapped test.

## Examples

```
d <- data.frame(x=c('b','a','a','a','b','b','b'),
               y=c('1','1','1','2','2','2','2'))
ft <- fisher.test(table(d))
wrapFisherTest(ft)
```

---

wrapFisherTest.table    *Wrap fisher.test (test of categorical independence).*

---

### Description

Wrap fisher.test (test of categorical independence).

### Usage

```
## S3 method for class 'table'
wrapFisherTest(
  x,
  ...,
  workspace = 2e+05,
  hybrid = FALSE,
  control = list(),
  or = 1,
  alternative = "two.sided",
  conf.int = TRUE,
  conf.level = 0.95,
  simulate.p.value = FALSE,
  B = 2000
)
```

### Arguments

x	data.frame
...	extra arguments (not used)
workspace	passed to <a href="#">fisher.test</a>
hybrid	passed to <a href="#">fisher.test</a>
control	passed to <a href="#">fisher.test</a>
or	passed to <a href="#">fisher.test</a>
alternative	passed to <a href="#">fisher.test</a>
conf.int	passed to <a href="#">fisher.test</a>
conf.level	passed to <a href="#">fisher.test</a>
simulate.p.value	passed to <a href="#">fisher.test</a>
B	passed to <a href="#">fisher.test</a>

### Value

wrapped test.

**Examples**

```
d <- data.frame(x=c('b', 'a', 'a', 'a', 'b', 'b', 'b'),
                y=c('1', '1', '1', '2', '2', '2', '2'))
t <- table(d)
wrapFisherTest(t)
```

---

wrapFTest	<i>Wrap F-test (significance identity relation).</i>
-----------	--

---

**Description**

Wrap F-test (significance identity relation).

**Usage**

```
wrapFTest(x, ...)
```

**Arguments**

x	numeric, data.frame or lm where to get model or data to score.
...	extra arguments

**See Also**

[wrapFTestImpl](#), [wrapFTest.lm](#), and [wrapFTest.data.frame](#)

---

wrapFTest.anova	<i>Wrap quality statistic of a linear relation from anova.</i>
-----------------	--

---

**Description**

Wrap quality statistic of a linear relation from anova.

**Usage**

```
## S3 method for class 'anova'
wrapFTest(x, ...)
```

**Arguments**

x	result from stats::anova(stats::lm())
...	extra arguments (not used)

**Value**

list of formatted string and fields

**Examples**

```
d <- data.frame(x1 = c(1,2,3,4,5,6,7,7),
               x2 = c(1,0,3,0,5,6,0,7),
               y = c(1,1,2,2,3,3,4,4))
model <- lm(y~x1+x2, data=d)
summary(model)
sigr::wrapFTest(model)
anov <- stats::anova(model)
print(anov)
lapply(sigr::wrapFTest(anov),
       function(ti) {
         sigr::render(ti,
                      pLargeCutoff= 1,
                      pSmallCutoff= 0,
                      statDigits=4,
                      sigDigits=4,
                      format='ascii')
       })
```

---

wrapFTest.data.frame    *Wrap quality statistic of identity relation from data.*

---

**Description**

Wrap quality statistic of identity relation from data.

**Usage**

```
## S3 method for class 'data.frame'
wrapFTest(
  x,
  predictionColumnName,
  yColumnName,
  nParameters = 1,
  meany = mean(x[[yColumnName]]),
  ...,
  na.rm = FALSE,
  format = NULL
)
```



**Arguments**

x	data frame containing columns to compare
predictionColumnName	character name of prediction column
yColumnName	character name of column containing dependent variable
nParameters	number of variables in model
meany	(optional) mean of y
...	extra arguments (not used)
na.rm	logical, if TRUE remove NA values
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx")

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
model <- lm(y~x,data=d)
summary(model)
d$pred <- predict(model,newdata=d)
sigr::wrapFTest(d,'pred','y')
```

---

wrapFTest.htest	<i>Wrap F-test (ratio of variances).</i>
-----------------	--

---

**Description**

Wrap F-test (ratio of variances).

**Usage**

```
## S3 method for class 'htest'
wrapFTest(x, ..., format = NULL)
```

**Arguments**

x	lm model
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)

**Value**

formatted string

**Examples**

```
v <- var.test(c(1,2,3,4,5,6,7,7), c(1, 1, 2))
sigr::wrapFTest(v)
```

---

wrapFTest.lm

*Wrap quality statistic of identity r regression.*

---

**Description**

Wrap quality statistic of identity r regression.

**Usage**

```
## S3 method for class 'lm'
wrapFTest(x, ..., format = NULL)
```

**Arguments**

x	lm model
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)

**Value**

formatted string

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
                y=c(1,1,2,2,3,3,4,4))
model <- lm(y~x,data=d)
summary(model)
sigr::wrapFTest(model)
```

---

wrapFTest.summary.lm *Wrap quality statistic of linear regression summary.*

---

### Description

Wrap quality statistic of linear regression summary.

### Usage

```
## S3 method for class 'summary.lm'
wrapFTest(x, ..., format = NULL)
```

### Arguments

x	summary.lm summary(lm()) object
...	extra arguments (not used)
format	if set the format to return ("html", "latex", "markdown", "ascii", "docx", ...)

### Value

formatted string

### Examples

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
                y=c(1,1,2,2,3,3,4,4))
model <- lm(y~x,data=d)
sum <- summary(model)
sigr::wrapFTest(sum)
```

---

wrapFTestezANOVA *Wrap quality statistic of a linear relation from ezANOVA (package ez).*

---

### Description

Please see <https://github.com/WinVector/sigr/issues/1#issuecomment-322311947> for an example.

### Usage

```
wrapFTestezANOVA(x, ...)
```

**Arguments**

x                    list result from ezANOVA (package ez).  
...                    extra arguments (not used)

**Value**

list of formatted string and fields

---

wrapFTestImpl                    *Wrap F-test (significance of identity relation).*

---

**Description**

Wrap F-test (significance of identity relation).

**Usage**

```
wrapFTestImpl(numdf, dendf, FValue, ..., format = NULL)
```

**Arguments**

numdf                degrees of freedom 1.  
dendf                degrees of freedom 2.  
FValue                observed F test statistic  
...                    not used, force later arguments to bind by name  
format                optional, suggested format

**Value**

wrapped statistic

**Examples**

```
wrapFTestImpl(numdf=2, dendf=55, FValue=5.56)
```

---

wrapPWR	<i>Wrap pwr test (difference in means by group).</i>
---------	--

---

**Description**

Wrap pwr test (difference in means by group).

**Usage**

```
wrapPWR(x, ...)
```

**Arguments**

x	test from pwr package
...	extra arguments

**See Also**

[pwr.2p.test](#)

---

wrapPWR.power.htest	<i>Wrap pwr test.</i>
---------------------	-----------------------

---

**Description**

Wrap pwr test.

**Usage**

```
## S3 method for class 'power.htest'  
wrapPWR(x, ...)
```

**Arguments**

x	pwr test result
...	extra arguments (not used)

**Value**

formatted string and fields

**Examples**

```
if(require("pwr", quietly = TRUE)) {  
  # Example from pwr package  
  # Exercise 6.1 p. 198 from Cohen (1988)  
  test <- pwr::pwr.2p.test(h=0.3,n=80,sig.level=0.05,alternative="greater")  
  wrapPWR(test)  
}
```

---

wrapSignificance	<i>Wrap a significance</i>
------------------	----------------------------

---

**Description**

Wrap a significance

**Usage**

```
wrapSignificance(significance, symbol = "p")
```

**Arguments**

significance	numeric the significance value.
symbol	the name of the value (e.g. "p", "t", ...).

**Value**

wrapped significance

**Examples**

```
wrapSignificance(1/300)
```

---

wrapTTest	<i>Wrap t.test (difference in means by group).</i>
-----------	--

---

**Description**

Wrap t.test (difference in means by group).

**Usage**

```
wrapTTest(x, ...)
```

**Arguments**

x	numeric, data.frame or test.
...	extra arguments

**See Also**

[wrapTTest.htest](#), and [wrapTTest.data.frame](#)

---

wrapTTest.data.frame	<i>Wrap t.test (difference in means by group).</i>
----------------------	--

---

**Description**

Wrap t.test (difference in means by group).

**Usage**

```
## S3 method for class 'data.frame'
wrapTTest(
  x,
  Column1Name,
  Column2Name,
  ...,
  y = NULL,
  alternative = c("two.sided", "less", "greater"),
  mu = 0,
  paired = FALSE,
  var.equal = FALSE,
  conf.level = 0.95,
  na.rm = FALSE
)
```

**Arguments**

x	data.frame
Column1Name	character column 1 name
Column2Name	character column 2 name
...	extra arguments passed to ttest
y	passed to <code>t.test</code>
alternative	passed to <code>t.test</code>
mu	passed to <code>t.test</code>
paired	passed to <code>t.test</code>
var.equal	passed to <code>t.test</code>
conf.level	passed to <code>t.test</code>
na.rm	logical, if TRUE remove NA values

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
render(wrapTTest(d, 'x', 'y'), pLargeCutoff=1)
# confirm p not order dependent
render(wrapTTest(d, 'y', 'x'), pLargeCutoff=1)
```

---

wrapTTest.htest	<i>Wrap t.test (difference in means by group).</i>
-----------------	--

---

**Description**

Wrap t.test (difference in means by group).

**Usage**

```
## S3 method for class 'htest'
wrapTTest(x, ...)
```

**Arguments**

x	t.test result
...	extra arguments (not used)



**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
tt <- t.test(d$x,d$y)
render(wrapTTest(tt),pLargeCutoff=1)
# confirm not rescaling, as a correlation test would
render(wrapTTest(t.test(d$x,2*d$y)),pLargeCutoff=1)
```

---

wrapTTest.numeric	<i>Wrap t.test (difference in means by group).</i>
-------------------	--

---

**Description**

Wrap t.test (difference in means by group).

**Usage**

```
## S3 method for class 'numeric'
wrapTTest(
  x,
  pop2,
  ...,
  y = NULL,
  alternative = c("two.sided", "less", "greater"),
  mu = 0,
  paired = FALSE,
  var.equal = FALSE,
  conf.level = 0.95,
  na.rm = FALSE
)
```

**Arguments**

x	numeric population 1
pop2	numeric population 2
...	extra arguments passed to ttest
y	passed to <a href="#">t.test</a>
alternative	passed to <a href="#">t.test</a>
mu	passed to <a href="#">t.test</a>
paired	passed to <a href="#">t.test</a>

<code>var.equal</code>	passed to <code>t.test</code>
<code>conf.level</code>	passed to <code>t.test</code>
<code>na.rm</code>	logical, if TRUE remove NA values

**Value**

formatted string and fields

**Examples**

```
d <- data.frame(x=c(1,2,3,4,5,6,7,7),
               y=c(1,1,2,2,3,3,4,4))
render(wrapTTest(d$x, d$y), pLargeCutoff=1)
# confirm p not order dependent
render(wrapTTest(d$y, d$x), pLargeCutoff=1)
```

# Index

add\_ROC\_derived\_columns, 4  
as.character.sigr\_statistic, 4

Bernoulli\_diff\_stat, 5  
binom.test, 46–49  
build\_ROC\_curve, 6

calcAUC, 7  
calcDeviance, 8  
calcSSE, 8  
cor.test, 58

estimateDifferenceZeroCrossing, 9

find\_area\_q, 10  
find\_AUC\_q, 10  
find\_matching\_a1\_1b, 11  
find\_matching\_conditional\_betas, 12  
find\_ROC\_matching\_ab  
    (find\_matching\_conditional\_betas),  
    12  
find\_ROC\_matching\_ab1  
    (find\_matching\_a1\_1b), 11  
fisher.test, 60, 62  
fit\_beta\_shapes, 13  
format.sigr\_statistic, 14

getRenderingFormat, 15

model\_utility, 15

permTestAUC, 17  
permutationScoreModel, 18  
print.sigr\_statistic, 19  
pwr.2p.test, 69

render, 20  
render.sigr\_aucpairtest, 21  
render.sigr\_aucpermttest, 21  
render.sigr\_aucresampptest, 22  
render.sigr\_Bernoulli\_diff\_test, 23

render.sigr\_binomtest, 24  
render.sigr\_chisqtest, 25  
render.sigr\_cohend, 26  
render.sigr\_cortest, 26  
render.sigr\_emptest, 27  
render.sigr\_fishertest, 28  
render.sigr\_ftest, 20, 29  
render.sigr\_permtest, 30  
render.sigr\_pwr\_hptest, 31  
render.sigr\_significance, 20, 31  
render.sigr\_tinterval, 32  
render.sigr\_ttest, 33  
resampleScoreModel, 34  
resampleScoreModelPair, 35  
resampleTestAUC, 37

sensitivity\_and\_specificity\_s12p12n,  
    38  
sensitivity\_from\_specificity\_q, 39  
sigr, 40

t.test, 72–74  
testAUCpair, 41  
TInterval, 42, 43, 44  
TInterval.data.frame, 42, 42, 43, 44  
TInterval.numeric, 42, 43, 43, 44  
TIntervalS, 42–44, 44

wrapBinomTest, 45, 46–50  
wrapBinomTest.data.frame, 45, 45, 46–50  
wrapBinomTest.htest, 45, 46, 46, 47–50  
wrapBinomTest.logical, 45–47, 47, 48–50  
wrapBinomTest.numeric, 45–48, 48, 49, 50  
wrapBinomTestS, 45–49, 49, 50  
wrapChiSqTest, 50  
wrapChiSqTest.anova, 51  
wrapChiSqTest.data.frame, 50, 52  
wrapChiSqTest.glm, 50, 53  
wrapChiSqTest.summary.glm, 53  
wrapChiSqTestImpl, 50, 54

- wrapCohenD, [55](#)
- wrapCohenD.data.frame, [55](#), [55](#)
- wrapCohenD.numeric, [56](#)
- wrapCorTest, [57](#)
- wrapCorTest.data.frame, [57](#), [57](#)
- wrapCorTest.htest, [57](#), [58](#)
- wrapFisherTest, [59](#)
- wrapFisherTest.data.frame, [59](#), [59](#)
- wrapFisherTest.htest, [59](#), [61](#)
- wrapFisherTest.table, [62](#)
- wrapFTest, [63](#)
- wrapFTest.anova, [63](#)
- wrapFTest.data.frame, [63](#), [64](#)
- wrapFTest.htest, [65](#)
- wrapFTest.lm, [63](#), [66](#)
- wrapFTest.summary.lm, [67](#)
- wrapFTestezANOVA, [67](#)
- wrapFTestImpl, [63](#), [68](#)
- wrapPWR, [69](#)
- wrapPWR.power.htest, [69](#)
- wrapSignificance, [70](#)
- wrapTTest, [71](#)
- wrapTTest.data.frame, [71](#), [71](#)
- wrapTTest.htest, [71](#), [72](#)
- wrapTTest.numeric, [73](#)