Package 'cmstatr'

July 22, 2025

Type Package

Title Statistical Methods for Composite Material Data

Version 0.10.0

Date 2024-11-18

Depends R (>= 3.3)

Description An implementation of the statistical methods commonly used for advanced composite materials in aerospace applications. This package focuses on calculating basis values (lower tolerance bounds) for material strength properties, as well as performing the associated diagnostic tests. This package provides functions for calculating basis values assuming several different distributions, as well as providing functions for non-parametric methods of computing basis values. Functions are also provided for testing the hypothesis that there is no difference between strength and modulus data from an alternate sample and that from a ``qualification'' or ``baseline'' sample. For a discussion of these statistical methods and their use, see the Composite Materials Handbook, Volume 1 (2012, ISBN: 978-0-7680-7811-4). Additional details about this package are available in the paper by Kloppenborg (2020, <doi:10.21105/joss.02265>).

URL https://www.cmstatr.net/, https://github.com/cmstatr/cmstatr

BugReports https://github.com/cmstatr/cmstatr/issues

License AGPL-3

Encoding UTF-8

LazyData true

Imports dplyr, generics, ggplot2, kSamples, MASS, purrr, rlang, stats, tibble, tidyr

Suggests knitr, lintr, rmarkdown, spelling, testthat, vdiffr

RoxygenNote 7.3.2

VignetteBuilder knitr

Language en-US

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation no

Author Stefan Kloppenborg [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1908-5214>),
Billy Cheng [ctb],
Ally Fraser [ctb],
Jeffrey Borlik [ctb],
Brice Langston [ctb],
Comtek Advanced Structures, Ltd. [fnd]

Maintainer Stefan Kloppenborg <stefan@kloppenborg.ca>

Repository CRAN

Date/Publication 2024-11-19 07:50:08 UTC

Contents

`ad_ksample` *Anderson–Darling K-Sample Test*

Description

This function performs an Anderson–Darling k-sample test. This is used to determine if several samples (groups) share a common (unspecified) distribution.

Usage

```
ad_ksample(data = NULL, x, groups, alpha = 0.025)
```

Arguments

`data`	a data.frame
`x`	the variable in the data.frame on which to perform the Anderson–Darling k-Sample test (usually strength)
`groups`	a variable in the data.frame that defines the groups
`alpha`	the significance level (default 0.025)

Details

This function is a wrapper for the ad.test function from the package `kSamples`. The method "exact" is specified in the call to `ad.test`. Refer to that package's documentation for details.

There is a minor difference in the formulation of the Anderson–Darling k-Sample test in CMH-17-1G, compared with that in the Scholz and Stephens (1987). This difference affects the test statistic and the critical value in the same proportion, and therefore the conclusion of the test is unaffected. When comparing the test statistic generated by this function to that generated by software that uses the CMH-17-1G formulation (such as ASAP, CMH17-STATS, etc.), the test statistic reported by this function will be greater by a factor of $(k-1)$, with a corresponding change in the critical value.

For more information about the difference between this function and the formulation in CMH-17-1G, see the vignette on the subject, which can be accessed by running `vignette("adktest")`

Value

Returns an object of class `adk`. This object has the following fields:

- `call` the expression used to call this function
- `data` the original data used to compute the ADK
- `groups` a vector of the groups used in the computation
- `alpha` the value of alpha specified
- `n` the total number of observations

- k the number of groups
- sigma the computed standard deviation of the test statistic
- ad the value of the Anderson–Darling k-Sample test statistic
- p the computed p-value
- reject_same_dist a boolean value indicating whether the null hypothesis that all samples come from the same distribution is rejected
- raw the original results returned from ad.test

References

F. W. Scholz and M. Stephens, "K-Sample Anderson–Darling Tests," Journal of the American Statistical Association, vol. 82, no. 399. pp. 918–924, Sep-1987.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Examples

```
library(dplyr)

carbon.fabric %>%
  filter(test == "WT") %>%
  filter(condition == "RTD") %>%
  ad_ksample(strength, batch)
##
## Call:
## ad_ksample(data = ., x = strength, groups = batch)
##
## N = 18          k = 3
## ADK = 0.912     p-value = 0.95989
## Conclusion: Samples come from the same distribution ( alpha = 0.025 )
```

---

anderson_darling *Anderson–Darling test for goodness of fit*

Description

Calculates the Anderson–Darling test statistic for a sample given a particular distribution, and determines whether to reject the hypothesis that a sample is drawn from that distribution.

Usage

```
anderson_darling_normal(data = NULL, x, alpha = 0.05)

anderson_darling_lognormal(data = NULL, x, alpha = 0.05)

anderson_darling_weibull(data = NULL, x, alpha = 0.05)
```

Arguments

data	a data.frame-like object (optional)
x	a numeric vector or a variable in the data.frame
alpha	the required significance level of the test. Defaults to 0.05.

Details

The Anderson–Darling test statistic is calculated for the distribution given by the user.

The observed significance level (OSL), or p-value, is calculated assuming that the parameters of the distribution are unknown; these parameters are estimate from the data.

The function anderson_darling_normal computes the Anderson–Darling test statistic given a normal distribution with mean and standard deviation equal to the sample mean and standard deviation.

The function anderson_darling_lognormal is the same as anderson_darling_normal except that the data is log transformed first.

The function anderson_darling_weibull computes the Anderson–Darling test statistic given a Weibull distribution with shape and scale parameters estimated from the data using a maximum likelihood estimate.

The test statistic, A, is modified to account for the fact that the parameters of the population are not known, but are instead estimated from the sample. This modification is a function of the sample size only, and is different for each distribution (normal/lognormal or Weibull). Several such modifications have been proposed. This function uses the modification published in Stephens (1974), Lawless (1982) and CMH-17-1G. Some other implementations of the Anderson-Darling test, such as the implementation in the nortest package, use other modifications, such as the one published in D'Agostino and Stephens (1986). As such, the p-value reported by this function may differ from the p-value reported by implementations of the Anderson–Darling test that use different modifiers. Only the unmodified test statistic is reported in the result of this function, but the modified test statistic is used to compute the OSL (p-value).

This function uses the formulae for observed significance level (OSL) published in CMH-17-1G. These formulae depend on the particular distribution used.

The results of this function have been validated against published values in Lawless (1982).

Value

an object of class anderson_darling. This object has the following fields.

- call the expression used to call this function
- dist the distribution used
- data a copy of the data analyzed
- n the number of observations in the sample
- A the Anderson–Darling test statistic
- osl the observed significance level (p-value), assuming the parameters of the distribution are estimated from the data
- alpha the required significance level for the test. This value is given by the user.
- reject_distribution a logical value indicating whether the hypothesis that the data is drawn from the specified distribution should be rejected

References

J. F. Lawless, *Statistical models and methods for lifetime data*. New York: Wiley, 1982.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

M. A. Stephens, "EDF Statistics for Goodness of Fit and Some Comparisons," Journal of the American Statistical Association, vol. 69, no. 347. pp. 730–737, 1974.

R. D'Agostino and M. Stephens, Goodness-of-Fit Techniques. New York: Marcel Dekker, 1986.

Examples

```
library(dplyr)

carbon.fabric %>%
  filter(test == "FC") %>%
  filter(condition == "RTD") %>%
  anderson_darling_normal(strength)
## Call:
## anderson_darling_normal(data = ., x = strength)
##
## Distribution:  Normal ( n = 18 )
## Test statistic:  A = 0.9224776
## OSL (p-value):  0.01212193  (assuming unknown parameters)
## Conclusion: Sample is not drawn from a Normal distribution (alpha = 0.05)
```

---

augment.mnr *Augment data with information from an* mnr *object*

Description

Augment accepts an mnr object (returned from the function maximum_normed_residual()) and a dataset and adds the column .outlier to the dataset. The column .outlier is a logical vector indicating whether each observation is an outlier.

When passing data into augment using the data argument, the data must be exactly the data that was passed to maximum_normed_residual.

Usage

```
## S3 method for class 'mnr'
augment(x, data = x$data, ...)
```

Arguments

x	an mnr object created by maximum_normed_residual()
data	a data.frame or tibble::tibble() containing the original data that was passed to maximum_normed_residual
...	Additional arguments. Not used. Included only to match generic signature.

Value

When `data` is supplied, `augment` returns `data`, but with one column appended. When `data` is not supplied, `augment` returns a new `tibble::tibble()` with the column `values` containing the original values used by `maximum_normed_residaul` plus one additional column. The additional column is:

- `.outler` a logical value indicating whether the observation is an outlier

See Also

`maximum_normed_residual()`

Examples

```
data <- data.frame(strength = c(80, 98, 96, 97, 98, 120))
m <- maximum_normed_residual(data, strength)

# augment can be called with the original data
augment(m, data)

##   strength .outlier
## 1       80    FALSE
## 2       98    FALSE
## 3       96    FALSE
## 4       97    FALSE
## 5       98    FALSE
## 6      120    FALSE

# or augment can be called without the orignal data and it will be
# reconstructed
augment(m)

## # A tibble: 6 x 2
##   values .outlier
##    <dbl> <lgl>
## 1     80 FALSE
## 2     98 FALSE
## 3     96 FALSE
## 4     97 FALSE
## 5     98 FALSE
## 6    120 FALSE
```

`basis` *Calculate basis values*

Description

Calculate the basis value for a given data set. There are various functions to calculate the basis values for different distributions. The basis value is the lower one-sided tolerance bound of a certain proportion of the population. For more information on tolerance bounds, see Meeker, et. al. (2017). For B-Basis, set the content of tolerance bound to $p = 0.90$ and the confidence level to $conf = 0.95$; for A-Basis, set $p = 0.99$ and $conf = 0.95$. While other tolerance bound contents and confidence levels may be computed, they are infrequently needed in practice.

These functions also perform some automated diagnostic tests of the data prior to calculating the basis values. These diagnostic tests can be overridden if needed.

Usage

```
basis_normal(
  data = NULL,
  x,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  override = c()
)

basis_lognormal(
  data = NULL,
  x,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  override = c()
)

basis_weibull(
  data = NULL,
  x,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  override = c()
)

basis_pooled_cv(
  data = NULL,
  x,
  groups,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  modcv = FALSE,
  override = c()
```

```
)

basis_pooled_sd(
  data = NULL,
  x,
  groups,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  modcv = FALSE,
  override = c()
)

basis_hk_ext(
  data = NULL,
  x,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  method = c("optimum-order", "woodward-frawley"),
  override = c()
)

basis_nonpara_large_sample(
  data = NULL,
  x,
  batch = NULL,
  p = 0.9,
  conf = 0.95,
  override = c()
)

basis_anova(data = NULL, x, groups, p = 0.9, conf = 0.95, override = c())
```

Arguments

data	a data.frame
x	the variable in the data.frame for which to find the basis value
batch	the variable in the data.frame that contains the batches.
p	the content of the tolerance bound. Should be 0.90 for B-Basis and 0.99 for A-Basis
conf	confidence level Should be 0.95 for both A- and B-Basis
override	a list of names of diagnostic tests to override, if desired. Specifying "all" will override all diagnostic tests applicable to the current method.
groups	the variable in the data.frame representing the groups
modcv	a logical value indicating whether the modified CV approach should be used. Only applicable to pooling methods.

`method` the method for Hanson–Koopmans nonparametric basis values. should be "optimum-order" for B-Basis and "woodward-frawley" for A-Basis.

Details

`data` is an optional argument. If `data` is given, it should be a `data.frame` (or similar object). When `data` is specified, the value of `x` is expected to be a variable within `data`. If `data` is not specified, `x` must be a vector.

When `modcv=TRUE` is set, which is only applicable to the pooling methods, the data is first modified according to the modified coefficient of variation (CV) rules. This modified data is then used when both calculating the basis values and also when performing the diagnostic tests. The modified CV approach is a way of adding extra variance to datasets with unexpectedly low variance.

`basis_normal` calculate the basis value by subtracting k times the standard deviation from the mean. k is given by the function `k_factor_normal()`. The equations in Krishnamoorthy and Mathew (2008) are used. `basis_normal` also performs a diagnostic test for outliers (using `maximum_normed_residual()`) and a diagnostic test for normality (using `anderson_darling_normal()`). If the argument `batch` is given, this function also performs a diagnostic test for outliers within each batch (using `maximum_normed_residual()`) and a diagnostic test for between batch variability (using `ad_ksample()`). The argument `batch` is only used for these diagnostic tests.

`basis_lognormal` calculates the basis value in the same way that `basis_normal` does, except that the natural logarithm of the data is taken.

`basis_lognormal` function also performs a diagnostic test for outliers (using `maximum_normed_residual()`) and a diagnostic test for normality (using `anderson_darling_lognormal()`). If the argument `batch` is given, this function also performs a diagnostic test for outliers within each batch (using `maximum_normed_residual()`) and a diagnostic test for between batch variability (using `ad_ksample()`). The argument `batch` is only used for these diagnostic tests.

`basis_weibull` calculates the basis value for data distributed according to a Weibull distribution. The confidence level for the content requested is calculated using the conditional method, as described in Lawless (1982) Section 4.1.2b. This has good agreement with tables published in CMH-17-1G. Results differ between this function and STAT17 by approximately 0.5\

`basis_weibull` function also performs a diagnostic test for outliers (using `maximum_normed_residual()`) and a diagnostic test for normality (using `anderson_darling_weibull()`). If the argument `batch` is given, this function also performs a diagnostic test for outliers within each batch (using `maximum_normed_residual()`) and a diagnostic test for between batch variability (using `ad_ksample()`). The argument `batch` is only used for these diagnostic tests.

`basis_hk_ext` calculates the basis value using the Extended Hanson–Koopmans method, as described in CMH-17-1G and Vangel (1994). For nonparametric distributions, this function should be used for samples up to n=28 for B-Basis and up to $n = 299$ for A-Basis. This method uses a pair of order statistics to determine the basis value. CMH-17-1G suggests that for A-Basis, the first and last order statistic is used: this is called the "woodward-frawley" method in this package, after the paper in which this approach is described (as referenced by Vangel (1994)). For B-Basis, another approach is used whereby the first and `j-th` order statistic are used to calculate the basis value. In this approach, the `j-th` order statistic is selected to minimize the difference between the tolerance limit (assuming that the order statistics are equal to the expected values from a standard normal distribution) and the population quantile for a standard normal distribution. This approach is described in Vangel (1994). This second method (for use when calculating B-Basis values) is

called "optimum-order" in this package. The results of basis_hk_ext have been verified against example results from the program STAT-17. Agreement is typically well within 0.2%.

Note that the implementation of hk_ext_z_j_opt changed after cmstatr version 0.8.0. This function is used internally by basis_hk_ext when method = "optimum-order". This implementation change may mean that basis values computed using this method may change slightly after version 0.8.0. However, both implementations seem to be equally valid. See the included vignette for a discussion of the differences between the implementation before and after version 0.8.0, as well as the factors given in CMH-17-1G. To access this vignette, run: vignette("hk_ext", package = "cmstatr")

basis_hk_ext also performs a diagnostic test for outliers (using maximum_normed_residual()) and performs a pair of tests that the sample size and method selected follow the guidance described above. If the argument batch is given, this function also performs a diagnostic test for outliers within each batch (using maximum_normed_residual()) and a diagnostic test for between batch variability (using ad_ksample()). The argument batch is only used for these diagnostic tests.

basis_nonpara_large_sample calculates the basis value using the large sample method described in CMH-17-1G. This method uses a sum of binomials to determine the rank of the ordered statistic corresponding with the desired tolerance limit (basis value). Results of this function have been verified against results of the STAT-17 program.

basis_nonpara_large_sample also performs a diagnostic test for outliers (using maximum_normed_residual()) and performs a test that the sample size is sufficiently large. If the argument batch is given, this function also performs a diagnostic test for outliers within each batch (using maximum_normed_residual()) and a diagnostic test for between batch variability (using ad_ksample()). The argument batch is only used for these diagnostic tests.

basis_anova calculates basis values using the ANOVA method. x specifies the data (normally strength) and groups indicates the group corresponding to each observation. This method is described in CMH-17-1G, but when the ratio of between-batch mean square to the within-batch mean square is less than or equal to one, the tolerance factor is calculated based on pooling the data from all groups. This approach is recommended by Vangel (1992) and by Krishnamoorthy and Mathew (2008), and is also implemented by the software CMH17-STATS and STAT-17. This function automatically performs a diagnostic test for outliers within each group (using maximum_normed_residual()) and a test for between group variability (using ad_ksample()) as well as checking that the data contains at least 5 groups. This function has been verified against the results of the STAT-17 program.

basis_pooled_sd calculates basis values by pooling the data from several groups together. x specifies the data (normally strength) and group indicates the group corresponding to each observation. This method is described in CMH-17-1G and matches the pooling method implemented in ASAP 2008.

basis_pooled_cv calculates basis values by pooling the data from several groups together. x specifies the data (normally strength) and group indicates the group corresponding to each observation. This method is described in CMH-17-1G.

basis_pooled_sd and basis_pooled_cv both automatically perform a number of diagnostic tests. Using maximum_normed_residual(), they check that there are no outliers within each group and batch (provided that batch is specified). They check the between batch variability using ad_ksample(). They check that there are no outliers within each group (pooling all batches) using maximum_normed_residual(). They check for the normality of the pooled data using anderson_darling_normal(). basis_pooled_sd checks for equality of variance of all data using levene_test() and basis_pooled_cv checks

for equality of variances of all data after transforming it using normalize_group_mean() using levene_test().

The object returned by these functions includes the named vector diagnostic_results. This contains all of the diagnostic tests performed. The name of each element of the vector corresponds with the name of the diagnostic test. The contents of each element will be "P" if the diagnostic test passed, "F" if the diagnostic test failed, "O" if the diagnostic test was overridden and NA if the diagnostic test was skipped (typically because an optional argument was not supplied).

The objects produced by the diagnostic tests are included in the named list diagnostic_obj. The name of each element in the list corresponds with the name of the test. This can be useful when evaluating diagnostic test failures.

The following list summarizes the diagnostic tests automatically performed by each function.

- basis_normal
  - outliers_within_batch
  - between_batch_variability
  - outliers
  - anderson_darling_normal
- basis_lognormal
  - outliers_within_batch
  - between_batch_variability
  - outliers
  - anderson_darling_lognormal
- basis_weibull
  - outliers_within_batch
  - between_batch_variability
  - outliers
  - anderson_darling_weibull
- basis_pooled_cv
  - outliers_within_batch
  - between_group_variability
  - outliers_within_group
  - pooled_data_normal
  - normalized_variance_equal
- basis_pooled_sd
  - outliers_within_batch
  - between_group_variability
  - outliers_within_group
  - pooled_data_normal
  - pooled_variance_equal
- basis_hk_ext
  - outliers_within_batch
  - between_batch_variability

    - outliers
    - sample_size
- basis_nonpara_large_sample
    - outliers_within_batch
    - between_batch_variability
    - outliers
    - sample_size
- basis_anova
    - outliers_within_group
    - equality_of_variance
    - number_of_groups

Value

an object of class basis This object has the following fields:

- call the expression used to call this function
- distribution the distribution used (normal, etc.)
- p the value of p supplied
- conf the value of $conf$ supplied
- modcv a logical value indicating whether the modified CV approach was used. Only applicable to pooling methods.
- data a copy of the data used in the calculation
- groups a copy of the groups variable. Only used for pooling and ANOVA methods.
- batch a copy of the batch data used for diagnostic tests
- modcv_transformed_data the data after the modified CV transformation
- override a vector of the names of diagnostic tests that were overridden. NULL if none were overridden
- diagnostic_results a named character vector containing the results of all the diagnostic tests. See the Details section for additional information
- diagnostic_obj a named list containing the objects produced by the diagnostic tests.
- diagnostic_failures a vector containing any diagnostic tests that produced failures
- n the number of observations
- r the number of groups, if a pooling method was used. Otherwise it is NULL.
- basis the basis value computed. This is a number except when pooling methods are used, in which case it is a data.frame.

References

J. F. Lawless, Statistical Models and Methods for Lifetime Data. New York: John Wiley & Sons, 1982.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

M. Vangel, "One-Sided Nonparametric Tolerance Limits," Communications in Statistics - Simulation and Computation, vol. 23, no. 4. pp. 1137–1154, 1994.

K. Krishnamoorthy and T. Mathew, Statistical Tolerance Regions: Theory, Applications, and Computation. Hoboken: John Wiley & Sons, 2008.

W. Meeker, G. Hahn, and L. Escobar, Statistical Intervals: A Guide for Practitioners and Researchers, Second Edition. Hoboken: John Wiley & Sons, 2017.

M. Vangel, "New Methods for One-Sided Tolerance Limits for a One-Way Balanced Random-Effects ANOVA Model," Technometrics, vol. 34, no. 2. Taylor & Francis, pp. 176–185, 1992.

See Also

hk_ext_z_j_opt()

k_factor_normal()

transform_mod_cv()

maximum_normed_residual()

anderson_darling_normal()

anderson_darling_lognormal()

anderson_darling_weibull()

ad_ksample()

normalize_group_mean()

Examples

```
library(dplyr)

# A single-point basis value can be calculated as follows
# in this example, three failed diagnostic tests are
# overridden.

res <- carbon.fabric %>%
  filter(test == "FC") %>%
  filter(condition == "RTD") %>%
  basis_normal(strength, batch,
               override = c("outliers",
                            "outliers_within_batch",
                            "anderson_darling_normal"))
print(res)

## Call:
## basis_normal(data = ., x = strength, batch = batch,
##     override = c("outliers", "outliers_within_batch",
```

```
##    "anderson_darling_normal"))
##
## Distribution:  Normal  ( n = 18 )
## The following diagnostic tests were overridden:
##     `outliers`,
##     `outliers_within_batch`,
##     `anderson_darling_normal`
## B-Basis:   ( p = 0.9 , conf = 0.95 )
## 76.94656

print(res$diagnostic_obj$between_batch_variability)

## Call:
## ad_ksample(x = x, groups = batch, alpha = 0.025)
##
## N = 18           k = 3
## ADK = 1.73       p-value = 0.52151
## Conclusion: Samples come from the same distribution ( alpha = 0.025 )

# A set of pooled basis values can also be calculated
# using the pooled standard deviation method, as follows.
# In this example, one failed diagnostic test is overridden.
carbon.fabric %>%
  filter(test == "WT") %>%
  basis_pooled_sd(strength, condition, batch,
                  override = c("outliers_within_batch"))

## Call:
## basis_pooled_sd(data = ., x = strength, groups = condition,
##                 batch = batch, override = c("outliers_within_batch"))
##
## Distribution:  Normal - Pooled Standard Deviation  ( n = 54, r = 3 )
## The following diagnostic tests were overridden:
##     `outliers_within_batch`
## B-Basis:   ( p = 0.9 , conf = 0.95 )
## CTD  127.6914
## ETW  125.0698
## RTD  132.1457
```

`calc_cv_star` *Calculate the modified CV from the CV*

Description

This function calculates the modified coefficient of variation (CV) based on a (unmodified) CV. The modified CV is calculated based on the rules in CMH-17-1G. Those rules are:

- For CV < 4\%, CV* = 6\%
- For 4\% <= CV < 8\%, CV* = CV / 2 + 4\%
- For CV > 8\%, CV* = CV

Usage

```
calc_cv_star(cv)
```

Arguments

cv	The CV to modify

Value

The value of the modified CV

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

See Also

cv()

Examples

```
# The modified CV for values of CV smaller than 4% is 6%
calc_cv_star(0.01)
## [1] 0.06

# The modified CV for values of CV larger than 8% is unchanged
calc_cv_star(0.09)
## [1] 0.09
```

---

carbon.fabric *Sample data for a generic carbon fabric*

Description

Datasets containing sample data that is typical of a generic carbon fabric prepreg. This data is used in several examples within the cmstatr package. This data is fictional and should only be used for learning how to use this package.

Usage

```
carbon.fabric

carbon.fabric.2
```

Format

An object of class `data.frame` with 216 rows and 5 columns.

An object of class `data.frame` with 177 rows and 9 columns.

`condition_summary` *Produce basis summary statistics for each (environmental) condition*

Description

Produces a `data.frame` containing the sample size and mean for each condition. If a reference condition (e.g. "RTD") is specified, the ratio of each condition mean value to the mean value for the reference condition is also calculated. If a `basis` object returned by one of the `basis_pooled` functions is given as an argument, this table also contains the basis value for each condition.

Usage

```
condition_summary(data, ...)

## S3 method for class 'data.frame'
condition_summary(data, x, condition, ref_condition = NULL, ...)

## S3 method for class 'basis'
condition_summary(data, ref_condition = NULL, ...)
```

Arguments

`data`	a `data.frame` or a `basis` object
`...`	not used
`x`	the column name of the variable of interest (usually strength)
`condition`	the column name for the condition
`ref_condition`	a `character` representing the reference condition

Value

a `data.frame`

Examples

```
library(dplyr)
carbon.fabric.2 %>%
  filter(test == "WT") %>%
  condition_summary(strength, condition, "RTD")

##   condition  n     mean mean_fraction
## 1       CTD 19 135.4719     0.9702503
## 2       RTD 28 139.6257     1.0000000
```

```
## 3       ETW 18 134.1009     0.9604312
## 4      ETW2 21 130.1545     0.9321673

carbon.fabric %>%
  filter(test == "FT") %>%
  basis_pooled_sd(strength, condition, batch) %>%
  condition_summary("RTD")

##   condition  n     mean mean_fraction    basis basis_fraction
## 1       RTD 18 127.6211     1.0000000 116.8894      1.0000000
## 2       ETW 18 117.8080     0.9231072 107.0762      0.9160476
## 3       CTD 18 125.9629     0.9870063 115.2311      0.9858133
```

---

cv *Calculate the coefficient of variation*

Description

The coefficient of variation (CV) is the ratio of the standard deviation to the mean of a sample. This function takes a vector of data and calculates the CV.

Usage

```
cv(x, na.rm = FALSE)
```

Arguments

x	a vector
na.rm	logical. Should missing values be removed?

Value

The calculated CV

Examples

```
set.seed(15)  # make this example reproducible
x <- rnorm(100, mean = 100, sd = 5)
cv(x)
## [1] 0.04944505

# the cv function can also be used within a call to dplyr::summarise
library(dplyr)
carbon.fabric %>%
filter(test == "WT") %>%
  group_by(condition) %>%
  summarise(mean = mean(strength), cv = cv(strength))
```

```
## # A tibble: 3 x 3
##   condition  mean     cv
##   <chr>     <dbl>  <dbl>
## 1 CTD        137. 0.0417
## 2 ETW        135. 0.0310
## 3 RTD        142. 0.0451
```

`equiv_change_mean` *Equivalency based on change in mean value*

Description

Checks for change in the mean value between a qualification data set and a sample. This is normally used to check for properties such as modulus. This function is a wrapper for a two-sample t–test.

Usage

```
equiv_change_mean(
  df_qual = NULL,
  data_qual = NULL,
  n_qual = NULL,
  mean_qual = NULL,
  sd_qual = NULL,
  data_sample = NULL,
  n_sample = NULL,
  mean_sample = NULL,
  sd_sample = NULL,
  alpha,
  modcv = FALSE
)
```

Arguments

`df_qual`	(optional) a data.frame containing the qualification data. Defaults to NULL.
`data_qual`	(optional) a vector of observations from the "qualification" data to which equivalency is being tested. Or the column of `df_qual` that contains this data. Defaults to NULL
`n_qual`	the number of observations in the qualification data to which the sample is being compared for equivalency
`mean_qual`	the mean from the qualification data to which the sample is being compared for equivalency
`sd_qual`	the standard deviation from the qualification data to which the sample is being compared for equivalency
`data_sample`	a vector of observations from the sample being compared for equivalency

n_sample	the number of observations in the sample being compared for equivalency
mean_sample	the mean of the sample being compared for equivalency
sd_sample	the standard deviation of the sample being compared for equivalency
alpha	the acceptable probability of a Type I error
modcv	a logical value indicating whether the modified CV approach should be used. Defaults to FALSE

Details

There are several optional arguments to this function. Either (but not both) data_sample or all of n_sample, mean_sample and sd_sample must be supplied. And, either (but not both) data_qual (and also df_qual if data_qual is a column name and not a vector) or all of n_qual, mean_qual and sd_qual must be supplied. If these requirements are violated, warning(s) or error(s) will be issued.

This function uses a two-sample t-test to determine if there is a difference in the mean value of the qualification data and the sample. A pooled standard deviation is used in the t-test. The procedure is per CMH-17-1G.

If modcv is TRUE, the standard deviation used to calculate the thresholds will be replaced with a standard deviation calculated using the Modified Coefficient of Variation (CV) approach. The Modified CV approach is a way of adding extra variance to the qualification data in the case that the qualification data has less variance than expected, which sometimes occurs when qualification testing is performed in a short period of time. Using the Modified CV approach, the standard deviation is calculated by multiplying CV_star * mean_qual where mean_qual is either the value supplied or the value calculated by mean(data_qual) and $CV*$ is determined using calc_cv_star().

Note that the modified CV option should only be used if that data passes the Anderson–Darling test.

Value

- call the expression used to call this function
- alpha the value of alpha passed to this function
- n_sample the number of observations in the sample for which equivalency is being checked. This is either the value n_sample passed to this function or the length of the vector data_sample.
- mean_sample the mean of the observations in the sample for which equivalency is being checked. This is either the value mean_sample passed to this function or the mean of the vector data-sample.
- sd_sample the standard deviation of the observations in the sample for which equivalency is being checked. This is either the value mean_sample passed to this function or the standard deviation of the vector data-sample.
- n_qual the number of observations in the qualification data to which the sample is being compared for equivalency. This is either the value n_qual passed to this function or the length of the vector data_qual.
- mean_qual the mean of the qualification data to which the sample is being compared for equivalency. This is either the value mean_qual passed to this function or the mean of the vector data_qual.

- `sd_qual` the standard deviation of the qualification data to which the sample is being compared for equivalency. This is either the value `mean_qual` passed to this function or the standard deviation of the vector `data_qual`.
- `modcv` logical value indicating whether the equivalency calculations were performed using the modified CV approach
- `sp` the value of the pooled standard deviation. If `modecv = TRUE`, this pooled standard deviation includes the modification to the qualification CV.
- `t0` the test statistic
- `t_req` the t-value for $\alpha/2$ and $df = n1 + n2 - 2$
- `threshold` a vector with two elements corresponding to the minimum and maximum values of the sample mean that would result in a pass
- `result` a character vector of either "PASS" or "FAIL" indicating the result of the test for change in mean

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

See Also

`calc_cv_star()`

`stats::t.test()`

Examples

```
equiv_change_mean(alpha = 0.05, n_sample = 9, mean_sample = 9.02,
                  sd_sample = 0.15785, n_qual = 28, mean_qual = 9.24,
                  sd_qual = 0.162, modcv = TRUE)
```

```
## Call:
## equiv_change_mean(n_qual = 28, mean_qual = 9.24, sd_qual = 0.162,
##                   n_sample = 9, mean_sample = 9.02, sd_sample = 0.15785,
##                   alpha = 0.05,modcv = TRUE)
##
## For alpha = 0.05
## Modified CV used
##                   Qualification        Sample
##           Number         28                9
##             Mean       9.24             9.02
##               SD      0.162          0.15785
##           Result               PASS
##    Passing Range       8.856695 to 9.623305
```

`equiv_mean_extremum` *Test for decrease in mean or minimum individual*

Description

This test is used when determining if a new process or manufacturing location produces material properties that are "equivalent" to an existing dataset, and hence the existing basis values are applicable to the new dataset. This test is also sometimes used for determining if a new batch of material is acceptable. This function determines thresholds based on both minimum individual and mean, and optionally evaluates a sample against those thresholds. The joint distribution between the sample mean and sample minimum is used to generate these thresholds. When there is no true difference between the existing ("qualification") and the new population from which the sample is obtained, there is a probability of α of falsely concluding that there is a difference in mean or variance. It is assumed that both the original and new populations are normally distributed. According to Vangel (2002), this test provides improved power compared with a test of mean and standard deviation.

Usage

```
equiv_mean_extremum(
  df_qual = NULL,
  data_qual = NULL,
  mean_qual = NULL,
  sd_qual = NULL,
  data_sample = NULL,
  n_sample = NULL,
  alpha,
  modcv = FALSE
)
```

Arguments

`df_qual`	(optional) a data.frame containing the qualification data. Defaults to NULL.
`data_qual`	(optional) a vector of observations from the "qualification" data to which equivalency is being tested. Or the column of `df_qual` that contains this data. Defaults to NULL
`mean_qual`	(optional) the mean from the "qualification" data to which equivalency is being tested. Defaults to NULL
`sd_qual`	(optional) the standard deviation from the "qualification" data to which equivalency is being tested. Defaults to NULL
`data_sample`	(optional) a vector of observations from the sample for which equivalency is being tested. Defaults to NULL
`n_sample`	(optional) the number of observations in the sample for which equivalency will be tested. Defaults to NULL
`alpha`	the acceptable probability of a type I error

`modcv` (optional) a boolean value indicating whether a modified CV should be used. Defaults to FALSE, in which case the standard deviation supplied (or calculated from `data_qual`) will be used directly.

Details

This function is used to determine acceptance limits for a sample mean and sample minimum. These acceptance limits are often used to set acceptance limits for material strength for each lot of material, or each new manufacturing site. When a sample meets the criteria that its mean and its minimum are both greater than these limits, then one may accept the lot of material or the new manufacturing site.

This procedure is used to ensure that the strength of material processed at a second site, or made with a new batch of material are not degraded relative to the data originally used to determine basis values for the material. For more information about the use of this procedure, see CMH-17-1G or PS-ACE 100-2002-006.

There are several optional arguments to this function. However, you can't omit all of the optional arguments. You must supply either `data_sample` or `n_sample`, but not both. You must also supply either `data_qual` (and `df_qual` if `data_qual` is a variable name and not a vector) or both `mean_qual` and `sd_qual`, but if you supply `data_qual` (and possibly `df_qual`) you should not supply either `mean_qual` or `sd_qual` (and visa-versa). This function will issue a warning or error if you violate any of these rules.

If `modcv` is TRUE, the standard deviation used to calculate the thresholds will be replaced with a standard deviation calculated using the Modified Coefficient of Variation (CV) approach. The Modified CV approach is a way of adding extra variance to the qualification data in the case that the qualification data has less variance than expected, which sometimes occurs when qualification testing is performed in a short period of time. Using the Modified CV approach, the standard deviation is calculated by multiplying `CV_star * mean_qual` where `mean_qual` is either the value supplied or the value calculated by `mean(data_qual)` and $CV*$ is the value computed by `calc_cv_star()`.

Value

Returns an object of class `equiv_mean_extremum`. This object is a list with the following named elements:

- `call` the expression used to call this function
- `alpha` the value of alpha passed to this function
- `n_sample` the number of observations in the sample for which equivalency is being checked. This is either the value `n_sample` passed to this function or the length of the vector `data_sample`.
- `k1` the factor used to calculate the minimum individual threshold. The minimum individual threshold is calculated as $W_{min} = qual\,mean - k_1 \cdot qual\,sd$
- `k2` the factor used to calculate the threshold for mean. The threshold for mean is calculated as $W_{mean} = qual\,mean - k_2 \cdot qual\,sd$
- `modcv` logical value indicating whether the acceptance thresholds are calculated using the modified CV approach
- `cv` the coefficient of variation of the qualification data. This value is not modified, even if `modcv=TRUE`

- `cv_star` The modified coefficient of variation. If `modcv=FALSE`, this will be `NULL`
- `threshold_min_indiv` The calculated threshold value for minimum individual
- `threshold_mean` The calculated threshold value for mean
- `result_min_indiv` a character vector of either "PASS" or "FAIL" indicating whether the data from `data_sample` passes the test for minimum individual. If `data_sample` was not supplied, this value will be `NULL`
- `result_mean` a character vector of either "PASS" or "FAIL" indicating whether the data from `data_sample` passes the test for mean. If `data_sample` was not supplied, this value will be `NULL`
- `min_sample` The minimum value from the vector `data_sample`. if `data_sample` was not supplied, this will have a value of `NULL`
- `mean_sample` The mean value from the vector `data_sample`. If `data_sample` was not supplied, this will have a value of `NULL`

References

M. G. Vangel. Lot Acceptance and Compliance Testing Using the Sample Mean and an Extremum, Technometrics, vol. 44, no. 3. pp. 242–249. 2002.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Federal Aviation Administration, "Material Qualification and Equivalency for Polymer Matrix Composite Material Systems," PS-ACE 100-2002-006, Sep. 2003.

See Also

`k_equiv()`

`calc_cv_star()`

Examples

```
equiv_mean_extremum(alpha = 0.01, n_sample = 6,
                    mean_qual = 100, sd_qual = 5.5, modcv = TRUE)
##
## Call:
## equiv_mean_extremum(mean_qual = 100, sd_qual = 5.5, n_sample = 6,
##     alpha = 0.01, modcv = TRUE)
##
## Modified CV used: CV* = 0.0675 ( CV = 0.055 )
##
## For alpha = 0.01 and n = 6
## ( k1 = 3.128346 and k2 = 1.044342 )
##                   Min Individual   Sample Mean
##      Thresholds:    78.88367        92.95069
```

geom_jitter_failure_mode *Jittered points showing (possibly multiple) failure modes*

Description

The geom_jitter_failure_mode is very similar to ggplot2::geom_jitter() except that a failure mode variable specified as the color and/or shape aesthetic is parsed to separate multiple failure modes and plot them separately. For example, if an observation has the failure mode "LAT/LAB", two points will be plotted, one with the failure mode "LAT" and the second with the failure mode "LAB".

Usage

```
geom_jitter_failure_mode(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "jitter",
  ...,
  width = NULL,
  height = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  sep = "[/, ]+"
)
```

Arguments

mapping	Set of aesthetic mapping created by aes(). See ggplot2::geom_jitter() for additional details.
data	The data to be displayed by this layer. See ggplot2::geom_jitter() for additional details.
stat	The statistical transformation to use on the data for this layer. See ggplot2::geom_jitter() for additional details.
position	A position adjustment to use on the data for this layer. See ggplot2::geom_jitter() for additional details.
...	Other arguments passed on to layer()'s params argument. See ggplot2::geom_jitter() for additional details.
width	The amount of horizontal jitter. See ggplot2::geom_jitter() for additional details.
height	The amount of vertical jitter. See ggplot2::geom_jitter() for additional details.

na.rm	If FALSE, the default, missing values are removed with warning. See ggplot2::geom_jitter() for additional details.
show.legend	NA, the default, indicates that nay aesthetics are mapped. See ggplot2::geom_jitter() for additional details.
inherit.aes	if FALSE, overrides the default aesthetics. See ggplot2::geom_jitter() for additional details.
sep	A regular expression indicating the character(s) separating multiple failure modes. By default "[/,]+"

Details

The variable specified for the aesthetics shape and color are passed to the function separate_failure_modes() to parse the failure modes and separate multiple failure modes separated by character(s) specified in the regular expression given in the parameter sep. By default, multiple failure modes are expected to be separated by spaces, commas or forward slashes, but this can be overridden.

If both shape and color aesthetics are specified, both must be identical.

See Also

separate_failure_modes()

ggplot2::geom_jitter()

Examples

```
library(dplyr)
library(ggplot2)
carbon.fabric.2 %>%
  filter(test == "WT") %>%
  ggplot(aes(x = condition, y = strength)) +
  geom_boxplot() +
  geom_jitter_failure_mode(aes(color = failure_mode, shape = failure_mode))
```

---

glance.adk *Glance at a* adk *(Anderson–Darling k-Sample) object*

Description

Glance accepts an object of type adk and returns a tibble::tibble() with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'adk'
glance(x, ...)
```

Arguments

x	an adk object
...	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row `tibble::tibble()` with the following columns:

- `alpha` the significance level for the test
- `n` the sample size for the test
- `k` the number of samples
- `sigma` the computed standard deviation of the test statistic
- `ad` the test statistic
- `p` the p-value of the test
- `reject_same_dist` whether the test concludes that the samples are drawn from different populations

See Also

`ad_ksample()`

Examples

```
x <- c(rnorm(20, 100, 5), rnorm(20, 105, 6))
k <- c(rep(1, 20), rep(2, 20))
a <- ad_ksample(x = x, groups = k)
glance(a)

## A tibble: 1 x 7
##   alpha     n     k sigma    ad       p reject_same_dist
##   <dbl> <int> <int> <dbl> <dbl>   <dbl> <lgl>
## 1 0.025    40     2 0.727  4.37 0.00487 TRUE
```

`glance.anderson_darling` *Glance at an* `anderson_darling` *object*

Description

Glance accepts an object of type `anderson_darling` and returns a `tibble::tibble()` with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'anderson_darling'
glance(x, ...)
```

Arguments

x	an anderson_darling object
...	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row tibble::tibble() with the following columns:

- dist the distribution used
- n the number of observations in the sample
- A the Anderson–Darling test statistic
- osl the observed significance level (p-value), assuming the parameters of the distribution are estimated from the data
- alpha the required significance level for the test. This value is given by the user.
- reject_distribution a logical value indicating whether the hypothesis that the data is drawn from the specified distribution should be rejected

See Also

anderson_darling()

Examples

```
x <- rnorm(100, 100, 4)
ad <- anderson_darling_weibull(x = x)
glance(ad)

## # A tibble: 1 x 6
##   dist        n     A        osl alpha reject_distribution
##   <chr>   <int> <dbl>      <dbl> <dbl> <lgl>
## 1 Weibull   100  2.62 0.00000207  0.05 TRUE
```

---

glance.basis *Glance at a basis object*

Description

Glance accepts an object of type basis and returns a tibble::tibble() with one row of summaries for each basis value.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'basis'
glance(x, include_diagnostics = FALSE, ...)
```

Arguments

x	a basis object
include_diagnostics	a logical value indicating whether to include columns for diagnostic tests. Default FALSE.
...	Additional arguments. Not used. Included only to match generic signature.

Details

For the pooled basis methods (basis_pooled_cv and basis_pooled_sd), the tibble::tibble() returned by glance will have one row for each group included in the pooling. For all other basis methods, the resulting tibble will have a single row.

If include_diagnostics=TRUE, there will be additional columns corresponding with the diagnostic tests performed. These column(s) will be of type character and will contain a "P" if the diagnostic test passed, a "F" if the diagnostic test failed, an "O" if the diagnostic test was overridden or NA if the test was not run (typically because an optional argument was not passed to the function that computed the basis value).

Value

A tibble::tibble() with the following columns:

- p the the content of the tolerance bound. Normally 0.90 or 0.99
- conf the confidence level. Normally 0.95
- distribution a string representing the distribution assumed when calculating the basis value
- modcv a logical value indicating whether the modified CV approach was used. Only applicable to pooling methods.
- n the sample size
- r the number of groups used in the calculation. This will be NA for single-point basis values
- basis the basis value

See Also

basis()

Examples

```
set.seed(10)
x <- rnorm(20, 100, 5)
b <- basis_normal(x = x)
glance(b)
```

```
## # A tibble: 1 x 7
##       p  conf distribution modcv     n r     basis
##   <dbl> <dbl> <chr>        <lgl> <int> <lgl> <dbl>
## 1   0.9  0.95 Normal       FALSE    20 NA     92.0


glance(b, include_diagnostics = TRUE)

## # A tibble: 1 x 11
##        p  conf distribution modcv     n r     basis outliers_within. . .
##    <dbl> <dbl> <chr>        <lgl> <int> <lgl> <dbl> <chr>
##  1   0.9  0.95 Normal       FALSE    20 NA     92.0 NA
## # . . .  with 3 more variables: between_batch_variability <chr>,
## #   outliers <chr>, anderson_darling_normal <chr>
```

`glance.equiv_change_mean` *Glance at a* `equiv_change_mean` *object*

Description

Glance accepts an object of type `equiv_change_mean` and returns a `tibble::tibble()` with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'equiv_change_mean'
glance(x, ...)
```

Arguments

`x`	a `equiv_change_mean` object returned from `equiv_change_mean()`
`...`	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row `tibble::tibble()` with the following columns:

- `alpha` the value of alpha passed to this function
- `n_sample` the number of observations in the sample for which equivalency is being checked. This is either the value `n_sample` passed to this function or the length of the vector `data_sample`.
- `mean_sample` the mean of the observations in the sample for which equivalency is being checked. This is either the value `mean_sample` passed to this function or the mean of the vector `data-sample`.

- sd_sample the standard deviation of the observations in the sample for which equivalency is being checked. This is either the value mean_sample passed to this function or the standard deviation of the vector data-sample.
- n_qual the number of observations in the qualification data to which the sample is being compared for equivalency. This is either the value n_qual passed to this function or the length of the vector data_qual.
- mean_qual the mean of the qualification data to which the sample is being compared for equivalency. This is either the value mean_qual passed to this function or the mean of the vector data_qual.
- sd_qual the standard deviation of the qualification data to which the sample is being compared for equivalency. This is either the value mean_qual passed to this function or the standard deviation of the vector data_qual.
- modcv logical value indicating whether the equivalency calculations were performed using the modified CV approach
- sp the value of the pooled standard deviation. If modecv = TRUE, this pooled standard deviation includes the modification to the qualification CV.
- t0 the test statistic
- t_req the t-value for $\alpha/2$ and $df = n1 + n2 - 2$
- threshold_min the minimum value of the sample mean that would result in a pass
- threshold_max the maximum value of the sample mean that would result in a pass
- result a character vector of either "PASS" or "FAIL" indicating the result of the test for change in mean

See Also

equiv_change_mean()

Examples

```
x0 <- rnorm(30, 100, 4)
x1 <- rnorm(5, 91, 7)
eq <- equiv_change_mean(data_qual = x0, data_sample = x1, alpha = 0.01)
glance(eq)
```

```
## # A tibble: 1 x 14
##   alpha n_sample mean_sample sd_sample n_qual mean_qual sd_qual modcv
##   <dbl>    <int>       <dbl>     <dbl>  <int>     <dbl>   <dbl> <lgl>
## 1  0.01        5        85.8      9.93     30      100.    3.90 FALSE
## # ... with 6 more variables: sp <dbl>, t0 <dbl>, t_req <dbl>,
## #   threshold_min <dbl>, threshold_max <dbl>, result <chr>
```

`glance.equiv_mean_extremum` *Glance at an* `equiv_mean_extremum` *object*

Description

Glance accepts an object of type `equiv_mean_extremum` and returns a `tibble::tibble()` with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'equiv_mean_extremum'
glance(x, ...)
```

Arguments

`x`	an equiv_mean_extremum object returned from `equiv_mean_extremum()`
`...`	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row `tibble::tibble()` with the following columns:

- `alpha` the value of alpha passed to this function
- `n_sample` the number of observations in the sample for which equivalency is being checked. This is either the value `n_sample` passed to this function or the length of the vector `data_sample`.
- `modcv` logical value indicating whether the acceptance thresholds are calculated using the modified CV approach
- `threshold_min_indiv` The calculated threshold value for minimum individual
- `threshold_mean` The calculated threshold value for mean
- `result_min_indiv` a character vector of either "PASS" or "FAIL" indicating whether the data from `data_sample` passes the test for minimum individual. If `data_sample` was not supplied, this value will be `NULL`
- `result_mean` a character vector of either "PASS" or "FAIL" indicating whether the data from `data_sample` passes the test for mean. If `data_sample` was not supplied, this value will be `NULL`
- `min_sample` The minimum value from the vector `data_sample`. if `data_sample` was not supplied, this will have a value of `NULL`
- `mean_sample` The mean value from the vector `data_sample`. If `data_sample` was not supplied, this will have a value of `NULL`

See Also

`equiv_mean_extremum()`

Examples

```
x0 <- rnorm(30, 100, 4)
x1 <- rnorm(5, 91, 7)
eq <- equiv_mean_extremum(data_qual = x0, data_sample = x1, alpha = 0.01)
glance(eq)
```

```
## # A tibble: 1 x 9
##   alpha n_sample modcv threshold_min_indiv threshold_mean
##   <dbl>    <int> <lgl>               <dbl>          <dbl>
## 1  0.01        5 FALSE                86.2           94.9
## # ... with 4 more variables: result_min_indiv <chr>, result_mean <chr>,
## #   min_sample <dbl>, mean_sample <dbl>
```

---

glance.levene *Glance at a* levene *object*

Description

Glance accepts an object of type levene and returns a tibble::tibble() with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'levene'
glance(x, ...)
```

Arguments

x	a levene object returned from levene_test()
...	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row tibble::tibble() with the following columns:

- alpha the value of alpha specified
- modcv a logical value indicating whether the modified CV approach was used.
- n the total number of observations
- k the number of groups
- f the value of the F test statistic
- p the computed p-value
- reject_equal_variance a boolean value indicating whether the null hypothesis that all samples have the same variance is rejected

See Also

levene_test()

Examples

```
df <- data.frame(
  groups = c(rep("A", 5), rep("B", 6)),
  strength = c(rnorm(5, 100, 6), rnorm(6, 105, 7))
)
levene_result <- levene_test(df, strength, groups)
glance(levene_result)

## # A tibble: 1 x 7
##   alpha modcv     n     k      f     p reject_equal_variance
##   <dbl> <lgl> <int> <int>  <dbl> <dbl> <lgl>
## 1  0.05 FALSE    11     2 0.0191 0.893 FALSE
```

---

glance.mnr *Glance at a* mnr *(maximum normed residual) object*

Description

Glance accepts an object of type mnr and returns a tibble::tibble() with one row of summaries.

Glance does not do any calculations: it just gathers the results in a tibble.

Usage

```
## S3 method for class 'mnr'
glance(x, ...)
```

Arguments

x	An mnr object
...	Additional arguments. Not used. Included only to match generic signature.

Value

A one-row tibble::tibble() with the following columns:

- mnr the computed MNR test statistic
- alpha the value of alpha used for the test
- crit the critical value given the sample size and the significance level
- n_outliers the number of outliers found

See Also

maximum_normed_residual()

Examples

```
x <- c(rnorm(20, 100, 5), 10)
m <- maximum_normed_residual(x = x)
glance(m)

## # A tibble: 1 x 4
##     mnr alpha  crit n_outliers
##   <dbl> <dbl> <dbl>      <dbl>
## 1  4.23  0.05  2.73          1
```

`hk_ext` *Calculate values related to Extended Hanson–Koopmans tolerance bounds*

Description

Calculates values related to Extended Hanson–Koopmans tolerance bounds as described by Vangel (1994).

Usage

```
hk_ext_z(n, i, j, p, conf)

hk_ext_z_j_opt(n, p, conf)
```

Arguments

`n`	the sample size
`i`	the first order statistic (1 <= i < j)
`j`	the second order statistic (i < j <= n)
`p`	the content of the tolerance bound (normally 0.90 or 0.99)
`conf`	the confidence level (normally 0.95)

Details

Hanson (1964) presents a nonparametric method for determining tolerance bounds based on consecutive order statistics. Vangel (1994) extends this method using non-consecutive order statistics.

The extended Hanson–Koopmans method calculates a tolerance bound (basis value) based on two order statistics and a weighting value `z`. The value of `z` is based on the sample size, which order statistics are selected, the desired content of the tolerance bond and the desired confidence level.

The function `hk_ext_z` calculates the weighting variable `z` based on selected order statistics `i` and `j`. Based on this value `z`, the tolerance bound can be calculated as:

$$S = zX_{(i)} + (1 - z)X_{(j)}$$

Where $X_{(i)}$ and $X_{(j)}$ are the `i-th` and `j-th` ordered observation.

The function `hk_ext_z_j_opt` determines the value of `j` and the corresponding value of `z`, assuming `i=1`. The value of `j` is selected such that the computed tolerance limit is nearest to the desired population quantile for a standard normal distribution when the order statistics are equal to the expected value of the order statistics for the standard normal distribution.

Value

For `hk_ext_z`, the return value is a numeric value representing the parameter `z` (denoted as `k` in CMH-17-1G).

For `hk_ext_z_j_opt`, the return value is named list containing `z` and `k`. The former is the value of `z`, as defined by Vangel (1994), and the latter is the corresponding order statistic.

References

M. Vangel, "One-Sided Nonparametric Tolerance Limits," Communications in Statistics - Simulation and Computation, vol. 23, no. 4. pp. 1137–1154, 1994.

D. L. Hanson and L. H. Koopmans, "Tolerance Limits for the Class of Distributions with Increasing Hazard Rates," The Annals of Mathematical Statistics, vol. 35, no. 4. pp. 1561–1570, 1964.

See Also

`basis_hk_ext()`

Examples

```
# The factors from Table 1 of Vangel (1994) can be recreated
# using the hk_ext_z function. For the sample size n=21,
# the median is the 11th ordered observation. The factor
# required for calculating the tolerance bound with a content
# of 0.9 and a confidence level of 0.95 based on the median
# and first ordered observation can be calculated as follows.
hk_ext_z(n = 21, i = 1, j = 11, p = 0.9, conf = 0.95)

## [1] 1.204806

# The hk_ext_z_j_opt function can be used to refine this value
# of z by finding an optimum value of j, rather than simply
# using the median. Here, we find that the optimal observation
# to use is the 10th, not the 11th (which is the median).
hk_ext_z_j_opt(n = 21, p = 0.9, conf = 0.95)

## $z
## [1] 1.217717
##
## $j
## [1] 10
```

k_equiv *k-factors for determining acceptance based on sample mean and an extremum*

Description

k-factors for determining acceptance based on sample mean and an extremum

Usage

```
k_equiv(alpha, n)
```

Arguments

alpha	the acceptable probability of a type I error
n	the number of observations in the sample to test

Details

The k-factors returned by this function are used for determining whether to accept a new dataset.

This function is used as part of the procedure for determining acceptance limits for a sample mean and sample minimum. These acceptance limits are often used to set acceptance limits for material strength for each lot of material, or each new manufacturing site. When a sample meets the criteria that its mean and its minimum are both greater than these limits, then one may accept the lot of material or the new manufacturing site.

This procedure is used to ensure that the strength of material processed at a second site, or made with a new batch of material are not degraded relative to the data originally used to determine basis values for the material. For more information about the use of this procedure, see CMH-17-1G or PS-ACE 100-2002-006.

According to Vangel (2002), the use of mean and extremum for this purpose is more powerful than the use of mean and standard deviation.

The results of this function match those published by Vangel within 0.05\ by Vangel are identical to those published in CMH-17-1G.

This function uses numerical integration and numerical optimization to find values of the factors k_1 and k_2 based on Vangel's saddle point approximation.

The value n refers to the number of observations in the sample being compared with the original population (the qualification sample is usually assumed to be equal to the population statistics).

The value of $alpha$ is the acceptable probability of a type I error. Normally, this is set to 0.05 for material or process equivalency and 0.01 when setting lot acceptance limits. Though, in principle, this parameter can be set to any number between 0 and 1. This function, however, has only been validated in the range of $1e-5 \leq alpha \leq 0.5$.

Value

a vector with elements c(k1, k2). k1 is for testing the sample extremum. k2 is for testing the sample mean

References

M. G. Vangel. Lot Acceptance and Compliance Testing Using the Sample Mean and an Extremum, Technometrics, vol. 44, no. 3. pp. 242–249. 2002.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Federal Aviation Administration, "Material Qualification and Equivalency for Polymer Matrix Composite Material Systems," PS-ACE 100-2002-006, Sep. 2003.

See Also

`equiv_mean_extremum()`

Examples

```
qual_mean <- 100
qual_sd <- 3.5
k <- k_equiv(0.01, 5)
print("Minimum Individual Acceptance Limit:")
print(qual_mean - qual_sd * k[1])
print("Minimum Average Acceptance Limit:")
print(qual_mean - qual_sd * k[2])

## [1] "Minimum Individual Acceptance Limit:"
## [1] 89.24981
## [1] "Minimum Average Acceptance Limit:"
## [1] 96.00123
```

---

k_factor_normal *Calculate k factor for basis values (kB, kA) with normal distribution*

Description

The factors returned by this function are used when calculating basis values (one-sided confidence bounds) when the data are normally distributed. The basis value will be equal to $\bar{x} - ks$, where $\bar{x}$ is the sample mean, s is the sample standard deviation and k is the result of this function. This function is internally used by `basis_normal()` when computing basis values.

Usage

```
k_factor_normal(n, p = 0.9, conf = 0.95)
```

Arguments

n	the number of observations (i.e. coupons)
p	the desired content of the tolerance bound. Should be 0.90 for B-Basis and 0.99 for A-Basis
conf	confidence level. Should be 0.95 for both A- and B-Basis

Details

This function calculates the k factors used when determining A- and B-Basis values for normally distributed data. To get kB, set the content of the tolerance bound to `p = 0.90` and the confidence level to `conf = 0.95`. To get kA, set `p = 0.99` and `conf = 0.95`. While other tolerance bound contents and confidence levels may be computed, they are infrequently needed in practice.

The k-factor is calculated using equation 2.2.3 of Krishnamoorthy and Mathew (2008).

This function has been validated against the kB tables in CMH-17-1G for each value of n from $n = 2$ to $n = 95$. It has been validated against the kA tables in CMH-17-1G for each value of n from $n = 2$ to $n = 75$. Larger values of n also match the tables in CMH-17-1G, but R emits warnings that "full precision may not have been achieved." When validating the results of this function against the tables in CMH-17-1G, the maximum allowable difference between the two is 0.002. The tables in CMH-17-1G give values to three decimal places.

For more information about tolerance bounds in general, see Meeker, et. al. (2017).

Value

the calculated factor

References

K. Krishnamoorthy and T. Mathew, Statistical Tolerance Regions: Theory, Applications, and Computation. Hoboken: John Wiley & Sons, 2008.

W. Meeker, G. Hahn, and L. Escobar, Statistical Intervals: A Guide for Practitioners and Researchers, Second Edition. Hoboken: John Wiley & Sons, 2017.

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

See Also

`basis_normal()`

Examples

```
kb <- k_factor_normal(n = 10, p = 0.9, conf = 0.95)
print(kb)

## [1] 2.35464

# This can be used to caclulate the B-Basis if
# the sample mean and sample standard deviation
# is known, and data is assumed to be normally
# distributed

sample_mean <- 90
sample_sd <- 5.2
print("B-Basis:")
print(sample_mean - sample_sd * kb)

## [1] B-Basis:
```

```
## [1] 77.75587
```

`levene_test` *Levene's Test (Median) for Equality of Variance*

Description

This function performs the Levene's test for equality of variance using the median. This is also known as the Brown-Forsythe test.

Usage

```
levene_test(data = NULL, x, groups, alpha = 0.05, modcv = FALSE)
```

Arguments

`data`	a data.frame
`x`	the variable in the data.frame or a vector on which to perform the Levene's test (usually strength)
`groups`	a variable in the data.frame that defines the groups
`alpha`	the significance level (default 0.05)
`modcv`	a logical value indicating whether the modified CV approach should be used.

Details

This function performs the Levene's test for equality of variance using median (also known as the Brown-Forsythe test). The data is transformed as follows:

$$w_{ij} = |x_{ij} - m_i|$$

Where m_i is median of the ith group. An F-Test is then performed on the transformed data.

When `modcv=TRUE`, the data from each group is first transformed according to the modified coefficient of variation (CV) rules before performing Levene's test.

Value

Returns an object of class `levene`. This object has the following fields:

- `call` the expression used to call this function
- `data` the original data supplied by the user
- `groups` a vector of the groups used in the computation
- `alpha` the value of alpha specified
- `modcv` a logical value indicating whether the modified CV approach was used.
- `n` the total number of observations

- k the number of groups
- f the value of the F test statistic
- p the computed p-value
- reject_equal_variance a boolean value indicating whether the null hypothesis that all samples have the same variance is rejected
- modcv_transformed_data the data after the modified CV transformation

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

NIST/SEMATECH e-Handbook of Statistical Methods, https://www.itl.nist.gov/div898/handbook/eda/section3/eda35a.htm, 2024.

Brown, M. B. and Forsythe, A. B. (1974), Journal of the American Statistical Association, 69, pp. 364-367.

See Also

calc_cv_star()

transform_mod_cv()

Examples

```
library(dplyr)

carbon.fabric.2 %>%
  filter(test == "FC") %>%
  levene_test(strength, condition)
##
## Call:
## levene_test(data = ., x = strength, groups = condition)
##
## n = 91          k = 5
## F = 3.883818    p-value = 0.00600518
## Conclusion: Samples have unequal variance ( alpha = 0.05 )
```

---

maximum_normed_residual

Detect outliers using the maximum normed residual method

Description

This function detects outliers using the maximum normed residual method described in CMH-17-1G. This method identifies a value as an outlier if the absolute difference between the value and the sample mean divided by the sample standard deviation exceeds a critical value.

Usage

```
maximum_normed_residual(data = NULL, x, alpha = 0.05)
```

Arguments

data	a data.frame
x	the variable in the data.frame for which to find the MNR or a vector if data=NULL. This must include at least 3 observations.
alpha	the significance level for the test. Defaults to 0.05

Details

data is an optional argument. If data is given, it should be a data.frame (or similar object). When data is specified, the value of x is expected to be a variable within data. If data is not specified, x must be a vector.

The maximum normed residual test is a test for outliers. The test statistic is given in CMH-17-1G. Outliers are identified in the returned object.

The maximum normed residual test statistic is defined as:

$$MNR = max\frac{|x_i - \bar{x}|}{s}$$

When the value of the MNR test statistic exceeds the critical value defined in Section 8.3.3.1 of CMH-17-1G, the corresponding value is identified as an outlier. It is then removed from the sample, and the test statistic is computed again and compared with the critical value corresponding with the new sample. This process is repeated until no values are identified as outliers.

Value

an object of class mnr This object has the following fields:

- call the expression used to call this function
- data the original data used to compute the MNR
- alpha the value of alpha given by the user
- mnr the computed MNR test statistic
- crit the critical value given the sample size and the significance level
- outliers a data.frame containing the index and value of each of the identified outliers
- n_outliers the number of outliers found

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Examples

```
library(dplyr)

carbon.fabric.2 %>%
  filter(test=="FC" & condition=="ETW2" & batch=="A") %>%
  maximum_normed_residual(strength)

## Call:
## maximum_normed_residual(data = ., x = strength)
##
## MNR =  1.958797  ( critical value = 1.887145 )
##
## Outliers ( alpha = 0.05 ):
##   Index  Value
##       6  44.26

carbon.fabric.2 %>%
  filter(test=="FC" & condition=="ETW2" & batch=="B") %>%
  maximum_normed_residual(strength)

## Call:
## maximum_normed_residual(data = ., x = strength)
##
## MNR =  1.469517  ( critical value = 1.887145 )
##
## No outliers detected ( alpha = 0.05 )
```

---

nested_data_plot *Create a plot of nested sources of variation*

Description

Creates a plot showing the breakdown of variation within a sample. This function uses `ggplot2` internally.

Usage

```
nested_data_plot(
  dat,
  x,
  groups = c(),
  stat = "mean",
  ...,
  y_gap = 1,
  divider_color = "grey50",
  point_args = list(),
  dline_args = list(),
  vline_args = list(),
```

```
  hline_args = list(),
  label_args = list(),
  connector_args = list()
)
```

Arguments

dat	a data.frame or similar object
x	the variable within dat to plot. Most often this would be a strength or modulus variable.
groups	a vector of variables to group the data by
stat	a function for computing the central location for each group. This is normally "mean" but could be "median" or another function.
...	extra options. See Details.
y_gap	the vertical gap between grouping variables
divider_color	the color of the lines between grouping variables. Or NULL to omit these lines.
point_args	arguments to pass to ggplot2::geom_point when plotting individual data points.
dline_args	arguments to pass to ggplot2::geom_segment when plotting the horizontal lines between data points.
vline_args	arguments to pass to ggplot2::geom_segment when plotting vertical lines
hline_args	arguments to pass to ggplot2::geom_segment when plotting horizontal lines connecting levels in groups
label_args	arguments to pass to ggplot2::geom_label when plotting labels
connector_args	arguments to pass to ggplot2::geom_point when plotting the connection between the vertical lines and the horizontal lines connecting levels in groups

Details

Extra options can be included to control aesthetic options. The following options are supported. Any (or all) can be set to a single variable in the data set.

- color: Controls the color of the data points.
- fill: Controls the fill color of the labels. When a particular label is associated with data points with more than one level of the supplied variable, the fill is omitted.

Examples

```
library(dplyr)
carbon.fabric.2 %>%
  filter(test == "WT" & condition == "RTD") %>%
  nested_data_plot(strength,
                   groups = c(batch, panel))

# Labels can be filled too
carbon.fabric.2 %>%
  filter(test == "WT" & condition == "RTD") %>%
```

```
nested_data_plot(strength,
                 groups = c(batch, panel),
                 fill = batch)
```

`nonpara_binomial_rank` *Rank for distribution-free tolerance bound*

Description

Calculates the rank order for finding distribution-free tolerance bounds for large samples. This function should only be used for computing B-Basis for samples larger than 28 or A-Basis for samples larger than 298. This function is used by `basis_nonpara_large_sample()`.

Usage

```
nonpara_binomial_rank(n, p, conf)
```

Arguments

`n`	the sample size
`p`	the desired content for the tolerance bound
`conf`	the confidence level for the desired tolerance bound

Details

This function uses the sum of binomial terms to determine the rank of the ordered statistic that corresponds with the desired tolerance limit. This approach does not assume any particular distribution. This approach is described by Guenther (1969) and by CMH-17-1G.

The results of this function have been verified against the tables in CMH-17-1G and agreement was found for all sample sizes published in CMH-17-1G for both A- and B-Basis, as well as the sample sizes `n+1` and `n-1`, where `n` is the sample size published in CMH-17-1G.

The tables in CMH-17-1G purportedly list the smallest sample sizes for which a particular rank can be used. That is, for a sample size one less than the `n` published in the table, the next lowest rank would be used. In some cases, the results of this function disagree by a rank of one for sample sizes one less than the `n` published in the table. This indicates a disagreement in that sample size at which the rank should change. This is likely due to numerical differences in this function and the procedure used to generate the tables. However, the disagreement is limited to sample 6500 for A-Basis; no discrepancies have been identified for B-Basis. Since these sample sizes are uncommon for composite materials testing, and the difference between subsequent order statistics will be very small for samples this large, this difference will have no practical effect on computed tolerance bounds.

Value

The rank corresponding with the desired tolerance bound

References

W. Guenther, "Determination of Sample Size for Distribution-Free Tolerance Limits," Jan. 1969. Available online: https://www.duo.uio.no/handle/10852/48686

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

See Also

basis_nonpara_large_sample()

Examples

```
nonpara_binomial_rank(n = 1693, p = 0.99, conf = 0.95)
## [1] 11

# The above example indicates that for a sample of 1693 observations,
# the A-Basis is best approximated as the 11th ordered observation.
# In the example below, the same ordered observation would also be used
# for a sample of size 1702.

nonpara_binomial_rank(n = 1702, p = 0.99, conf = 0.95)
## [1] 11
```

---

normalize_group_mean *Normalize values to group means*

Description

This function computes the mean of each group, then divides each observation by its corresponding group mean. This is commonly done when pooling data across environments.

Usage

```
normalize_group_mean(x, group)
```

Arguments

x	the variable containing the data to normalized
group	the variable containing the groups

Details

Computes the mean for each group, then divides each value by the mean for the corresponding group.

Value

Returns a vector of normalized values

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Examples

```
library(dplyr)
carbon.fabric.2 %>%
filter(test == "WT") %>%
  select(condition, strength) %>%
  mutate(condition_norm = normalize_group_mean(strength, condition)) %>%
  head(10)
```

```
##    condition strength condition_norm
## 1        CTD  142.817      1.0542187
## 2        CTD  135.901      1.0031675
## 3        CTD  132.511      0.9781438
## 4        CTD  135.586      1.0008423
## 5        CTD  125.145      0.9237709
## 6        CTD  135.203      0.9980151
## 7        CTD  128.547      0.9488832
## 8        CTD  127.709      0.9426974
## 9        CTD  127.074      0.9380101
## 10       CTD  126.879      0.9365706
```

---

normalize_ply_thickness

Normalizes strength values to ply thickness

Description

This function takes a vector of strength values and a vector of measured thicknesses, and a nominal thickness and returns the normalized strength.

Usage

```
normalize_ply_thickness(strength, measured_thk, nom_thk)
```

Arguments

strength	the strength to be normalized. Either a vector or a numeric
measured_thk	the measured thickness of the samples. Must be the same length as strength
nom_thk	the nominal thickness. Must be a single numeric value.

Details

It is often necessary to normalize strength values so that variation in specimen thickness does not unnecessarily increase variation in strength. See CMH-17-1G, or other references, for information about the cases where normalization is appropriate.

Either cured ply thickness or laminate thickness may be used for `measured_thk` and `nom_thk`, as long as the same decision made for both values.

The formula applied is:

$$normalized\,value = test\,value\frac{t_{measured}}{t_{nominal}}$$

If you need to normalize based on fiber volume fraction (or another method), you will first need to calculate the nominal cured ply thickness (or laminate thickness). Those calculations are outside the scope of this documentation.

Value

The normalized strength values

References

"Composite Materials Handbook, Volume 1. Polymer Matrix Composites Guideline for Characterization of Structural Materials," SAE International, CMH-17-1G, Mar. 2012.

Examples

```
library(dplyr)

carbon.fabric.2 %>%
select(thickness, strength) %>%
  mutate(normalized_strength = normalize_ply_thickness(strength,
                                                        thickness,
                                                        0.105)) %>%
  head(10)
```

```
##    thickness strength normalized_strength
## 1      0.112  142.817            152.3381
## 2      0.113  135.901            146.2554
## 3      0.113  132.511            142.6071
## 4      0.112  135.586            144.6251
## 5      0.113  125.145            134.6799
## 6      0.113  135.203            145.5042
## 7      0.113  128.547            138.3411
## 8      0.113  127.709            137.4392
## 9      0.113  127.074            136.7558
## 10     0.114  126.879            137.7543
```

`separate_failure_modes` *Separate multiple failure modes into multiple rows*

Description

For a `data.frame` containing a column with (some) multiple failure modes, this function expands the `data.frame` by repeating each row with multiple failure modes so that each row contains only a single failure mode.

Usage

```
separate_failure_modes(data, failure_mode, sep = "[/, ]+")
```

Arguments

`data`	a `data.frame`
`failure_mode`	the column in `data` containing the failure modes
`sep`	a regular expression with the character(s) separating individual failure modes. Default `"[/, ]+"`.

Details

When multiple failure modes are reported, they are commonly reported in the format "LGM/GIT" or "LGM,GIT". This function will separate these multiple failure modes into multiple rows.

This can be useful when counting the number of coupons exhibited each failure mode.

Examples

```
library(dplyr)
data.frame(strength = c(101, 102), fm = c("LGM/GIT", "LGM")) %>%
  separate_failure_modes(fm)
##
## # A tibble: 3 × 2
##   strength fm
##      <dbl> <chr>
## 1      101 LGM
## 2      101 GIT
## 3      102 LGM
```

stat_esf *Empirical Survival Function*

Description

The empirical survival function (ESF) provides a visualization of a distribution. This is closely related to the empirical cumulative distribution function (ECDF). The empirical survival function is simply ESF = 1 - ECDF.

Usage

```
stat_esf(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  n = NULL,
  pad = FALSE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by `aes()`.
data	The data to be displayed in this layer. This has the same usage as a `ggplot2` `stat` function.
geom	The geometric object to use to display the data.
position	Position argument
show.legend	Should this layer be included in the legends?
inherit.aes	If `FALSE`, overrides the default aesthetic, rather than combining with them.
n	If `NULL`, do not interpolated. Otherwise, the number of points to interpolate.
pad	If `TRUE`, pad the ESF with additional points `(-Inf, 0)` and `(0, Inf)`.
...	Other arguments to pass on to `layer`.

`stat_normal_surv_func` *Normal Survival Function*

Description

The Normal survival function provides a visualization of a distribution. A normal curve is fit based on the mean and standard deviation of the data, and the survival function of this normal curve is plotted. The survival function is simply one minus the CDF.

Usage

```
stat_normal_surv_func(
  mapping = NULL,
  data = NULL,
  geom = "smooth",
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  n = 100,
  pad = FALSE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`.
`data`	The data to be displayed in this layer. This has the same usage as a `ggplot2 stat` function.
`geom`	The geometric object to use to display the data.
`position`	Position argument
`show.legend`	Should this layer be included in the legends?
`inherit.aes`	If `FALSE`, overrides the default aesthetic, rather than combining with them.
`n`	If `NULL`, do not interpolated. Otherwise, the number of points to interpolate.
`pad`	If `TRUE`, pad the ESF with additional points `(-Inf, 0)` and `(0, Inf)`.
`...`	Other arguments to pass on to `layer`.

`transform_mod_cv` *Transforms data according to the modified CV rule*

Description

Transforms data according to the modified coefficient of variation (CV) rule. This is used to add additional variance to datasets with unexpectedly low variance, which is sometimes encountered during testing of new materials over short periods of time.

Two versions of this transformation are implemented. The first version, `transform_mod_cv()`, transforms the data in a single group (with no other structure) according to the modified CV rules.

The second version, `transform_mod_cv_ad()`, transforms data that is structured according to both condition and batch, as is commonly done for the Anderson–Darling k-Sample and Anderson-Darling tests when pooling across environments.

Usage

```
transform_mod_cv_ad(x, condition, batch)

transform_mod_cv(x)
```

Arguments

`x`	a vector of data to transform
`condition`	a vector indicating the condition to which each observation belongs
`batch`	a vector indicating the batch to which each observation belongs

Details

The modified CV transformation takes the general form:

$$\frac{S_i^*}{S_i}(x_{ij} - \bar{x_i}) + \bar{x_i}$$

Where S_i^* is the modified standard deviation (mod CV times mean) for the ith group; S_i is the standard deviation for the ith group, $\bar{x_i}$ is the group mean and x_{ij} is the observation.

`transform_mod_cv()` takes a vector containing the observations and transforms the data. The equation above is used, and all observations are considered to be from the same group.

`transform_mod_cv_ad()` takes a vector containing the observations plus a vector containing the corresponding conditions and a vector containing the batches. This function first calculates the modified CV value from the data from each condition (independently). Then, within each condition, the transformation above is applied to produce the transformed data x'. This transformed data is further transformed using the following equation.

$$x''_{ij} = C(x'_{ij} - \bar{x_i}) + \bar{x_i}$$

Where:

$$C = \sqrt{\frac{SSE^*}{SSE'}}$$

$$SSE^* = (n-1)(CV^*\bar{x})^2 - \sum(n_i(\bar{x_i} - \bar{x})^2)$$

$$SSE' = \sum(x'_{ij} - \bar{x_i})^2$$

Value

A vector of transformed data

See Also

calc_cv_star()

cv()

Examples

```
# Transform data according to the modified CV transformation
# and report the original and modified CV for each condition

library(dplyr)
carbon.fabric %>%
filter(test == "FT") %>%
  group_by(condition) %>%
  mutate(trans_strength = transform_mod_cv(strength)) %>%
  head(10)
```

```
## # A tibble: 10 x 6
## # Groups:   condition [1]
##    id          test  condition batch strength trans_strength
##    <chr>       <chr> <chr>     <int>    <dbl>          <dbl>
##  1 FT-RTD-1-1 FT    RTD           1     126.           126.
##  2 FT-RTD-1-2 FT    RTD           1     139.           141.
##  3 FT-RTD-1-3 FT    RTD           1     116.           115.
##  4 FT-RTD-1-4 FT    RTD           1     132.           133.
##  5 FT-RTD-1-5 FT    RTD           1     129.           129.
##  6 FT-RTD-1-6 FT    RTD           1     130.           130.
##  7 FT-RTD-2-1 FT    RTD           2     131.           131.
##  8 FT-RTD-2-2 FT    RTD           2     124.           124.
##  9 FT-RTD-2-3 FT    RTD           2     125.           125.
## 10 FT-RTD-2-4 FT    RTD           2     120.           119.
```

```
# The CV of this transformed data can be computed to verify
# that the resulting CV follows the rules for modified CV

carbon.fabric %>%
  filter(test == "FT") %>%
  group_by(condition) %>%
```

```
  mutate(trans_strength = transform_mod_cv(strength)) %>%
  summarize(cv = sd(strength) / mean(strength),
            mod_cv = sd(trans_strength) / mean(trans_strength))
```

```
## # A tibble: 3 x 3
##   condition     cv mod_cv
##   <chr>      <dbl>  <dbl>
## 1 CTD       0.0423 0.0612
## 2 ETW       0.0369 0.0600
## 3 RTD       0.0621 0.0711
```

Index