

# Package: pooledpeaks (via r-universe)

November 25, 2024

**Title** Genetic Analysis of Pooled Samples

**Version** 1.0.7

**Description** Analyzing genetic data obtained from pooled samples. This package can read in Fragment Analysis output files, process the data, and score peaks, as well as facilitate various analyses, including cluster analysis, calculation of genetic distances and diversity indices, as well as bootstrap resampling for statistical inference. Specifically tailored to handle genetic data efficiently, researchers can explore population structure, genetic differentiation, and genetic relatedness among samples. We updated some functions from Covarrubias-Pazaran et al. (2016) <[doi:10.1186/s12863-016-0365-6](https://doi.org/10.1186/s12863-016-0365-6)> to allow for the use of new file formats and referenced the following to write our genetic analysis functions: Long et al. (2022) <[doi:10.1038/s41598-022-04776-0](https://doi.org/10.1038/s41598-022-04776-0)>, Jost (2008) <[doi:10.1111/j.1365-294x.2008.03887.x](https://doi.org/10.1111/j.1365-294x.2008.03887.x)>, Nei (1973) <[doi:10.1073/pnas.70.12.3321](https://doi.org/10.1073/pnas.70.12.3321)>, Foulley et al. (2006) <[doi:10.1016/j.livprodsci.2005.10.021](https://doi.org/10.1016/j.livprodsci.2005.10.021)>, Chao et al. (2008) <[doi:10.1111/j.1541-0420.2008.01010.x](https://doi.org/10.1111/j.1541-0420.2008.01010.x)>.

**URL** <https://github.com/kmkuesters/pooledpeaks>

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** ape, dplyr, Fragman, graphics, magrittr, pdftools, qpdf, rlang, stats, tibble, tidyr, utils

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**BugReports** <https://github.com/kmkuesters/pooledpeaks/issues>

**NeedsCompilation** no

**Author** Kathleen Kuesters [aut, cre, cph]  
(<<https://orcid.org/0000-0003-0238-7889>>), Jeffrey Long [aut],  
Jessica Blanton [aut], Walter Blank [ctb], Jeffrey Kovach  
[ctb], Ronald Blanton [ctb]

**Maintainer** Kathleen Kuesters <kathleen.kuesters@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-11-19 23:20:02 UTC

**Config/pak/sysreqs** libicu-dev libjpeg-dev libssl-dev  
libpoppler-cpp-dev

## Contents

|                                 |    |
|---------------------------------|----|
| AlRich . . . . .                | 3  |
| associate_dyes . . . . .        | 4  |
| BootStrap3 . . . . .            | 4  |
| check_fsa_v_batch . . . . .     | 5  |
| clean_scores . . . . .          | 6  |
| cluster . . . . .               | 7  |
| ClusterFromSamples . . . . .    | 8  |
| data_manipulation . . . . .     | 8  |
| DistCor . . . . .               | 9  |
| EmpiricalSE . . . . .           | 10 |
| fsa_batch_imp . . . . .         | 11 |
| fsa_metadata . . . . .          | 12 |
| GeneIdentityMatrix . . . . .    | 13 |
| GeneticDistanceMatrix . . . . . | 14 |
| GST . . . . .                   | 14 |
| JostD . . . . .                 | 15 |
| JostD_KK . . . . .              | 16 |
| lf_to_tdf . . . . .             | 16 |
| LoadData . . . . .              | 17 |
| MDSplot . . . . .               | 18 |
| PCDM . . . . .                  | 19 |
| preGST . . . . .                | 19 |
| preJostD . . . . .              | 20 |
| Rep_check . . . . .             | 20 |
| RWCDistanceMatrix . . . . .     | 21 |
| SampleOfLoci . . . . .          | 22 |
| score_markers_rev3 . . . . .    | 22 |
| TwoLevelGST . . . . .           | 25 |
| TypedLoci . . . . .             | 25 |

**Index**

**27**

---

|        |                                   |
|--------|-----------------------------------|
| AlRich | <i>Calculate Allelic Richness</i> |
|--------|-----------------------------------|

---

**Description**

This function calculates allelic richness based on provided genetic data.

**Usage**

```
AlRich(datafile = data.frame, n = matrix)
```

**Arguments**

|          |   |
|----------|---|
| datafile | A data frame containing the data as read in by <a href="#">LoadData</a>   |
| n        | A matrix representing the number of markers successfully genotyped like the output of the <a href="#">TypedLoci</a> function. |

**Value**

A vector containing the allelic richness for each locus.

**Examples**

```
genetic_data <- data.frame(  
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),  
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),  
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),  
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),  
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)  
)  
  
n_alleles <- matrix(c(  
  3, 3, 3,  
  3, 3, 3,  
  3, 3, 3  
) , nrow = 3, byrow = TRUE,  
  dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))  
  
AlRich(datafile=genetic_data,n=n_alleles)
```

---

associate\_dyes                      *Associate Dye Names in Batch Import Output*

---

### Description

This function associates dye info with fragman channel names. It was designed to be performed on any fsa formats after final columns are correctly imported.

### Usage

```
associate_dyes(x, y)
```

### Arguments

x                      The Output list of data frames from fsa\_batch\_imp.  
y                      The path to the folder from the current directory where the .fsa files that will be analyzed are stored.

### Value

The input dataframe with an added column assigning fluorescent dye colors.

### Examples

```
y <- system.file("extdata", package = "pooledpeaks")
x <- fsa_batch_imp(y, channels = 5, fourier = FALSE, saturated = FALSE ,
lets.pullup = FALSE, plotting = FALSE, rawPlot = FALSE,
llength = 3000, ulength = 80000 )
associate_dyes(x,y)
```

---

BootStrap3                      *Perform Bootstrap Analysis*

---

### Description

This function performs bootstrap analysis on genetic data.

### Usage

```
BootStrap3(A = data.frame, Rep = 20, Stat = 1)
```

### Arguments

A                      Data frame containing data as read in by [LoadData](#)  
Rep                      Number of bootstrap replicates.  
Stat                      Type of statistic to compute (1 for AlRich, 2 for TwoLevelGST)

**Value**

Either a matrix of AIRich statistics or a list containing various statistics computed using TwoLevel-GST.

**Examples**

```
genetic_data <- data.frame(  
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),  
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),  
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),  
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),  
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)  
)  
  
BootStrap3(A=genetic_data, Rep=10, Stat=1)  
BootStrap3(A=genetic_data, Rep=10, Stat=2)
```

---

check\_fsa\_v\_batch      *Check .fsa Version and Batch Information*

---

**Description**

This function analyzes .fsa files in a specified folder, providing a summary of their version and batch information.

**Usage**

```
check_fsa_v_batch(x)
```

**Arguments**

x                      The path to the folder from the current directory where the .fsa files that will be analyzed are stored.

**Value**

A written summary of how many .fsa files are in the folder and which version they are.

**Examples**

```
file_path <- system.file("extdata", package = "pooledpeaks")  
check_fsa_v_batch(x = file_path)
```

---

|              |                          |
|--------------|--------------------------|
| clean_scores | <i>Clean Scores Data</i> |
|--------------|--------------------------|

---

### Description

This function cleans the `score_markers_rev3` data by applying specified patterns and replacements to the ID and filename columns.

### Usage

```
clean_scores(
  scores_data,
  pattern1 = NULL,
  replacement1 = NULL,
  pattern2 = NULL,
  replacement2 = NULL,
  pattern3 = NULL,
  replacement3 = NULL
)
```

### Arguments

|                           |  |
|---------------------------|--|
| <code>scores_data</code>  | The list containing the output scores data from the <code>score_markers_rev3</code> .  |
| <code>pattern1</code>     | The first pattern to replace in the ID. This is intended to clean up the ID names for when the machine adds substrings to the names. For example <code>104.1a_FA060920_2020-06-09_C05.fsa.1</code> becomes <code>104.1a</code> using <code>pattern1 = "_FA.*"</code> and <code>replacement1 = ""</code>                                  |
| <code>replacement1</code> | Replacement for the first pattern.   |
| <code>pattern2</code>     | The second pattern to replace in the ID. See <code>pattern1</code> for more details.   |
| <code>replacement2</code> | Replacement for the second pattern.  |
| <code>pattern3</code>     | The pattern to replace in the file name. This is intended to clean up the file names for when the machine adds sub strings to the names. For example <code>104.1a_FA060920_2020-06-09_C05.fsa.1</code> becomes <code>104.1a_FA060920_2020-06-09_C05.fsa</code> using <code>pattern3 = "\.1*\$"</code> and <code>replacement3 = ""</code> |
| <code>replacement3</code> | Replacement for the file name pattern.   |

### Value

A cleaned long format data frame

### Examples

```
scores_data <- list(
  data.frame(Score = c(90, 85, 70), stringsAsFactors = FALSE),
  data.frame(Score = c(80, 75, 60), stringsAsFactors = FALSE)
)
rownames(scores_data[[1]]) <- c("104.1a_FA060920_2020-06-09_C05.fsa_Sa.1",
```

```

                                "105.2b_FA060920_2020-06-09_C05.fsa_Sa.1",
                                "106.3c_FA060920_2020-06-09_C05.fsa_Fa.1")
rownames(scores_data[[2]]) <- c("107.4d_FA060920_2020-06-09_C05.fsa_Sa.1",
                                "108.5e_FA060920_2020-06-09_C05.fsa_Sa.1",
                                "109.6f_SA060920_2020-06-09_C05.fsa_Fa.1")
clean_scores(scores_data,pattern1= "_SA.*", replacement1="",
pattern2= "_FA.*",replacement2="")

```

---

cluster

*K-means Clustering*


---

## Description

K-means Clustering

## Usage

```
cluster(RawData = data.frame, K = 2)
```

## Arguments

|         |   |
|---------|---|
| RawData | A data frame containing the raw data as read in by <a href="#">LoadData</a> |
| K       | An integer specifying the number of clusters.                               |

## Value

A list containing the results of the K-means cluster analysis, including cluster assignments and original data.

## Examples

```

genetic_data <- data.frame(
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)
)
cluster(RawData=genetic_data, K=2)

```

---

ClusterFromSamples      *Cluster From Samples*

---

### Description

Perform clustering on samples of loci from a data frame and calculate statistics.

### Usage

```
ClusterFromSamples(datafile = data.frame, numloci = 5, reps = 100)
```

### Arguments

datafile      A data frame containing the input data must be in LoadData style [LoadData](#).  
numloci      An integer specifying the number of loci to sample.  
reps      An integer specifying the number of repetitions.

### Value

A matrix containing statistics calculated from the clustering results.

### Examples

```
genetic_data <- data.frame(  
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),  
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),  
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),  
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),  
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)  
)
```

```
ClusterFromSamples(datafile=genetic_data, numloci=5, reps=10)
```

---

data\_manipulation      *Data Manipulation for Marker Data*

---

### Description

This function ensures that at least one peak for each sample is greater than a specified threshold (default: 500) and then formats the data frame for the next steps in the analysis.

### Usage

```
data_manipulation(marker, threshold = 500)
```



**Arguments**

|           |  |
|-----------|--|
| marker    | A data frame containing marker data, where each row represents a marker and each column represents a sample. |
| threshold | The threshold value for peak height. Peaks below this threshold will be replaced with 0.                     |

**Value**

A formatted data frame where at least one peak for each sample is greater than the specified threshold.

**Examples**

```
marker_data <- data.frame(  
  Sample1 = c(400, 600, 700,0),  
  Sample2 = c(450, 550, 480,0),  
  Sample3 = c(300, 200, 400,200),  
  Sample4 = c(0,0,0,0),  
  row.names=c(185,188,191,194)  
)  
data_manipulation(marker_data,threshold=500)
```

---

DistCor

*Distance Correlation*

---

**Description**

Calculate the correlation between expected and realized genetic distances and plot them.

**Usage**

```
DistCor(GD = matrix)
```

**Arguments**

|    |  |
|----|--|
| GD | A matrix containing the genetic distance data. |
|----|--|

**Value**

A plot showing the Expected Genetic Distance versus Realized Genetic Distance

**Examples**

```
genetic_distance_matrix <- matrix(c(0.316455, 0.2836333, 0.2760485,
0.2685221, 0.2797302,0.3202661,0.2836333, 0.3106084, 0.2867215, 0.2687472,
0.2596309, 0.2957862,0.2760485,0.2867215, 0.3338663, 0.297918, 0.3057039,
0.3153261,0.2685221, 0.2687472, 0.297918,0.3107094, 0.2753477, 0.3042383,
0.2797302, 0.2596309, 0.3057039, 0.2753477, 0.3761386, 0.3398558,0.3202661,
0.2957862, 0.3153261, 0.3042383, 0.3398558, 0.4402125),
nrow = 6, byrow = TRUE,dimnames = list(c("Sample1", "Sample2", "Sample3",
"Ind1", "Ind2", "Ind3"),
c("Sample1", "Sample2", "Sample3", "Ind1", "Ind2", "Ind3")))

DC<- DistCor(genetic_distance_matrix)
```

---

EmpiricalSE

*Calculate Empirical Standard Error*


---

**Description**

This function calculates the empirical standard error based on repeated sampling.

**Usage**

```
EmpiricalSE(datafile = data.frame, NLocs = 10)
```

**Arguments**

datafile            A data frame containing genetic data from the [LoadData](#)  
NLocs                Number of loci to sample in each iteration.

**Value**

A numeric vector containing the empirical standard error estimates.

**Examples**

```
genetic_data <- data.frame(
Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),
Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),
Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),
Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),
Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)
)
```

```
EmpiricalSE(datafile=genetic_data, NLocs=3)
```

fsa\_batch\_imp

*Batch Import of .fsa files***Description**

This function imports and extracts all of the information out of the .fsa files and combines them into one list type object. `fsa_batch_imp` is a modification of the original Fracman import script function, `storing_inds`. This revised script accommodates ABI's .fsa file format up to version 3. It retains Fracman functions for Fourier transformation, saturated peaks, and pull-up correction. Notable adjustments include updating channel parameters, utilizing Dyechannel count from the file directory, and streamlining the script by extracting data only from "DATA" tags. Major changes involve column selection for v3 formats and modifications to the "channel" parameter. Minor changes include allowing relative paths for the data directory, importing only .fsa files, and renaming channels with dye names. This revision ensures successful execution for any format version up to 3.

**Usage**

```
fsa_batch_imp(
  folder,
  channels = NULL,
  fourier = TRUE,
  saturated = TRUE,
  lets.pullup = FALSE,
  plotting = FALSE,
  rawPlot = FALSE,
  llength = 3000,
  ulength = 80000
)
```

**Arguments**

|                          |   |
|--------------------------|---|
| <code>folder</code>      | The path to the folder from the current directory where the .fsa files that will be analyzed are stored.  |
| <code>channels</code>    | The number of dye channels expected, including the ladder.  |
| <code>fourier</code>     | True/False Should fourier transformation be applied.  |
| <code>saturated</code>   | True/False whether to Check and correct for saturated peaks.  |
| <code>lets.pullup</code> | True/False Applying pull up correction to the samples to decrease noise from channel to channel. The default is FALSE, please do not change this. |
| <code>plotting</code>    | True/False Should plots be drawn of all channels after data cleaning.   |
| <code>rawPlot</code>     | True/False indicating whether a plot should be drawn of all vectors.  |
| <code>llength</code>     | A numeric value for the minimum number of indexes in each channel.  |
| <code>ulength</code>     | A numeric value for the maximum number fo indexes in each channel.  |

**Value**

Output is a LIST where each element of the list is a DATAFRAME with the channels in columns for each FSA file

**Examples**

```
file_path <- system.file("extdata", package = "pooledpeaks")
fsa_batch_imp(file_path, channels = 5, fourier = FALSE, saturated = FALSE ,
lets.pullup = FALSE,plotting = FALSE, rawPlot = FALSE)
```

---

|              |                          |
|--------------|--------------------------|
| fsa_metadata | <i>Retrieve Metadata</i> |
|--------------|--------------------------|

---

**Description**

Retrieves basic info from .fsa files about the sample and run,and aggregates multiple samples in a single object.

**Usage**

```
fsa_metadata(x)
```

**Arguments**

x                    The path to the folder from the current directory where the .fsa files that will be analyzed are stored.

**Value**

A data frame that contains the metadata of the machine and run extracted from the .fsa file. One row for each .fsa file in directory x and the following columns: retrieved\_sample\_name, batch\_container\_name, fsa\_version, user, run\_start\_date, run\_start\_time, machine\_type,machineN\_serial.

**Examples**

```
file_path <- system.file("extdata", package = "pooledpeaks")
fsa_metadata(x = file_path)
```

---

|                    |                             |
|--------------------|-----------------------------|
| GeneIdentityMatrix | <i>Gene Identity Matrix</i> |
|--------------------|-----------------------------|

---

## Description

Using the number of typed loci, this function calculates the gene identity between all possible pairwise combinations between individuals for all markers creating a matrix.

## Usage

```
GeneIdentityMatrix(RawData = data.frame, LociGenotyped = matrix)
```

## Arguments

**RawData** A data frame containing the input data must be in LoadData style [LoadData](#).  
**LociGenotyped** The Output from the TypedLoci function

## Value

The Gene Identity Matrix

## Examples

```
genetic_data <- data.frame(  
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),  
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),  
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),  
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),  
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)  
)  
  
n_alleles <- matrix(c(  
  3, 3, 3,  
  3, 3, 3,  
  3, 3, 3  
) , nrow = 3, byrow = TRUE,  
  dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))  
  
GeneIdentityMatrix(RawData=genetic_data,LociGenotyped=n_alleles)
```

---

 GeneticDistanceMatrix *Genetic Distance Matrix*


---

**Description**

This function calculates the genetic distance matrix from a given gene identity matrix.

**Usage**

```
GeneticDistanceMatrix(J = matrix)
```

**Arguments**

J                    The Gene Identity Matrix created using [GeneIdentityMatrix](#)

**Value**

The Genetic Distance Matrix

**Examples**

```
gene_identity_matrix <- matrix(c(
  0.3164550, 0.2836333, 0.2760485,
  0.2836333, 0.3106084, 0.2867215,
  0.2760485, 0.2867215, 0.3338663
), nrow = 3, byrow = TRUE,
dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))

GeneticDistanceMatrix(gene_identity_matrix)
```

---

 GST

*Nei's GST*


---

**Description**

This function calculates GST (Nei's standard genetic distance) measure from a gene identity matrix.

**Usage**

```
GST(J = matrix, pairwise = TRUE)
```

**Arguments**

J                    A square matrix representing a gene identity matrix.  
 pairwise           Logical indicating whether to calculate pairwise GST. If set to FALSE, must not contain any missing data.

**Value**

If `pairwise = TRUE`, returns a matrix of pairwise GST values. If `pairwise = FALSE`, returns the overall GST value.

**Examples**

```
gene_identity_matrix <- matrix(c(
  0.3164550, 0.2836333, 0.2760485,
  0.2836333, 0.3106084, 0.2867215,
  0.2760485, 0.2867215, 0.3338663
), nrow = 3, byrow = TRUE,
dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))

GST(J=gene_identity_matrix, pairwise=TRUE)
GST(J=gene_identity_matrix, pairwise=FALSE)
```

---

JostD

*Calculate Jost's D*


---

**Description**

This function calculates Jost's D measure from a gene identity matrix.

**Usage**

```
JostD(J = matrix, pairwise = TRUE)
```

**Arguments**

|                       |   |
|-----------------------|---|
| <code>J</code>        | A gene identity matrix.   |
| <code>pairwise</code> | Logical indicating whether to calculate pairwise Jost's D. If <code>pairwise=FALSE</code> , must not have any missing data. |

**Value**

If `pairwise = TRUE`, returns a matrix of pairwise Jost's D values. If `pairwise = FALSE`, returns the overall Jost's D value.

**Examples**

```
gene_identity_matrix <- matrix(c(
  0.3164550, 0.2836333, 0.2760485,
  0.2836333, 0.3106084, 0.2867215,
  0.2760485, 0.2867215, 0.3338663
), nrow = 3, byrow = TRUE,
dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))

JostD(J=gene_identity_matrix, pairwise=TRUE)
JostD(J=gene_identity_matrix, pairwise=FALSE)
```

---

|          |   |
|----------|---|
| JostD_KK | <i>Pairwise Jost D between replicates</i> |
|----------|---|

---

**Description**

This function calculates Jost's D between two columns, specifically designed for comparing duplicate samples based on allele frequencies.

**Usage**

```
JostD_KK(Ni1, Ni2)
```

**Arguments**

|     |  |
|-----|--|
| Ni1 | Vector containing the allele frequencies of the first duplicate sample.  |
| Ni2 | Vector containing the allele frequencies of the second duplicate sample. |

**Value**

The calculated Jost's D value.

---

|           |                            |
|-----------|----------------------------|
| lf_to_tdf | <i>Transform LF to TDF</i> |
|-----------|----------------------------|

---

**Description**

This function transforms a data frame from LF (long format) to TDF (table format), performing various data manipulation steps such as spreading data across columns, removing NA and/or 0 columns, merging ID allele heights within each replicate, transposing the table, converting from character to numeric class, and replacing empty data with "0".

**Usage**

```
lf_to_tdf(x)
```

**Arguments**

|   |  |
|---|--|
| x | A data frame in LF format ideally coming out of the clean_scores function. |
|---|--|

**Value**

A transformed data frame in TDF format.



**Examples**

```
scores<- data.frame(ID=c("104.1a","105.2b","106.3c","107.4d","108.5e",
"109.6f"),
filename=c("104.1a_FA060920_2020-06-09_C05.fsa_Sa.1",
"105.2b_FA060920_2020-06-09_C05.fsa_Sa.1",
"106.3c_FA060920_2020-06-09_C05.fsa_Fa.1",
"107.4d_FA060920_2020-06-09_C05.fsa_Sa.1",
"108.5e_FA060920_2020-06-09_C05.fsa_Sa.1",
"109.6f_SA060920_2020-06-09_C05.fsa_Fa.1"),
hei=c(2000,3000,4000,5000,2500, 1000),
pos=c(2000,3000,4000,5000,2500, 1000),
wei=c(290,285,280,275,270,260),
row.names= c("104.1a_FA060920_2020-06-09_C05.fsa_Sa.1",
"105.2b_FA060920_2020-06-09_C05.fsa_Sa.1",
"106.3c_FA060920_2020-06-09_C05.fsa_Fa.1",
"107.4d_FA060920_2020-06-09_C05.fsa_Sa.1",
"108.5e_FA060920_2020-06-09_C05.fsa_Sa.1",
"109.6f_SA060920_2020-06-09_C05.fsa_Fa.1"))

lf_to_tdf(scores)
```

LoadData

*Load Genetic Data***Description**

This function imports data for genetic analysis.

**Usage**

```
LoadData(datafile = NULL)
```

**Arguments**

datafile            The path to your datafile. The format of your data should be .txt or .csv.

**Value**

A data frame containing the imported data formatted in the way necessary for downstream population genetic functions.

**Examples**

```
file<-system.file("extdata", "Multiplex_frequencies.txt",
package = "pooledpeaks")
LoadData(file)
```

---

MDSplot

*Multi Dimensional Scaling (MDS) Plot*


---

**Description**

Generate a multidimensional scaling (MDS) plot from genetic distance data.

**Usage**

```
MDSplot(
  distance = matrix,
  pcs = c(1, 2),
  PF = NULL,
  y = c("dodgerblue", "red", "turquoise3", "purple", "olivedrab3")
)
```

**Arguments**

|          |  |
|----------|--|
| distance | A matrix containing the genetic distance data.                 |
| pcs      | A numeric vector specifying the principal coordinates to plot. |
| PF       | A factor vector specifying population labels.                  |
| y        | A character vector specifying colors for population labels.    |

**Value**

The output is the MDS plot for the samples for the specified principal coordinates.

**Examples**

```
genetic_distance_matrix <- matrix(c(
  0, 0.2836333, 0.2760485, 0.2685221, 0.2797302, 0.3202661,
  0.2836333, 0, 0.2867215, 0.2687472, 0.2596309, 0.2957862,
  0.2760485, 0.2867215, 0, 0.297918, 0.3057039, 0.3153261,
  0.2685221, 0.2687472, 0.297918, 0, 0.2753477, 0.3042383,
  0.2797302, 0.2596309, 0.3057039, 0.2753477, 0, 0.3398558,
  0.3202661, 0.2957862, 0.3153261, 0.3042383, 0.3398558, 0),
  nrow = 6, byrow = TRUE, dimnames = list(c("Sample1", "Sample2",
  "Sample3", "Ind1", "Ind2", "Ind3"),
  c("Sample1", "Sample2", "Sample3", "Ind1", "Ind2", "Ind3")))

MDSplot(distance=genetic_distance_matrix, pcs=c(1,3))
```

---

PCDM

*Post-consolidation Data Manipulation*


---

**Description**

This function manipulates consolidated marker data and egg count data to prepare them for further analysis.

**Usage**

```
PCDM(consolidated_marker = data.frame, eggcount = data.frame, marker_name)
```

**Arguments**

`consolidated_marker` A data frame containing consolidated marker data.  
`eggcount` A data frame containing egg count data.  
`marker_name` A string specifying the marker name.

**Value**

A dataframe containing the allele frequencies and eggcounts for each sample.

**Examples**

```
marker_data <- data.frame(
  Sample1 = c(400, 600, 700),
  Sample2 = c(450, 550, 480),
  Sample3 = c(300, 200, 500),
  row.names=c(185,188,191)
)

eggs<-data.frame(
  ID=c("Sample1","Sample2","Sample3"),n=c(3000,400,50))

PCDM(consolidated_marker=marker_data, eggcount= eggs,"SMMS2")
```

---

preGST

*Pre GST Calculation*


---

**Description**

This function calculates the GST from a gene identity matrix.

**Usage**

```
preGST(G = matrix)
```

**Arguments**

G                    The gene identity matrix

**Value**

The GST value.

preJostD                    *Calculate Pre-Jost's D*

**Description**

This function calculates the pre-Jost's D measure from a gene identity matrix.

**Usage**

```
preJostD(G = matrix)
```

**Arguments**

G                    A square matrix representing a gene identity matrix.

**Value**

The Jost's D value.

Rep\_check                    *Replicate Check for Duplicate Samples*

**Description**

This function checks for duplicate samples in the input data frame and calculates the average peak heights for each sample. If the Jost's D between duplicate samples exceeds 0.05, it flags those samples.

**Usage**

```
Rep_check(df)
```

**Arguments**

df                    The input data frame containing peak heights for each sample.

**Value**

A data frame containing the average peak heights for each sample, with flagged samples where duplicates have a Jost's D exceeding 0.05.

**Examples**

```
marker_data <- data.frame(
  Sample.1a = c(400, 600, 700),
  Sample.1b = c(420, 606, 710),
  Sample.2a = c(450, 550, 480),
  Sample.2b = c(500, 540, 480),
  Sample.3a = c(300, 200, 500),
  Sample.3b = c(290, 100, 400),
  row.names=c(185,188,191)
)

Rep_check(marker_data)
```

---

RWCDistanceMatrix

*Random Walk Covariance Distance Matrix*


---

**Description**

This function calculates the RWC (Random Walk Covariance) distance matrix from a given matrix of genetic distances.

**Usage**

```
RWCDistanceMatrix(J = matrix)
```

**Arguments**

J                      The Genetic Distance Matrix calculated using [GeneticDistanceMatrix](#)

**Value**

A matrix representing the distance matrix calculated using the Random Walk Covariance method.

**Examples**

```
genetic_distance_matrix <- matrix(c(0.316455, 0.2836333, 0.2760485,
0.2685221, 0.2797302,0.3202661,0.2836333, 0.3106084, 0.2867215, 0.2687472,
0.2596309, 0.2957862,0.2760485,0.2867215, 0.3338663, 0.297918, 0.3057039,
0.3153261,0.2685221, 0.2687472, 0.297918,0.3107094, 0.2753477, 0.3042383,
0.2797302, 0.2596309, 0.3057039, 0.2753477, 0.3761386,0.3398558,0.3202661,
0.2957862, 0.3153261, 0.3042383, 0.3398558, 0.4402125),
nrow = 6, byrow = TRUE, dimnames = list(c("Sample1", "Sample2", "Sample3",
"Ind1", "Ind2", "Ind3"),
```

```
c("Sample1", "Sample2", "Sample3", "Ind1", "Ind2", "Ind3"))
RWCDistanceMatrix(genetic_distance_matrix)
```

---

|              |                       |
|--------------|-----------------------|
| SampleOfLoci | <i>Sample Of Loci</i> |
|--------------|-----------------------|

---

### Description

An internal function that supports ClusterFromSamples. Sample loci from a dataset based on the number of loci specified.

### Usage

```
SampleOfLoci(aaax = data.frame, NLoci = max(aaax[, 1]))
```

### Arguments

|       |   |
|-------|---|
| aaax  | A data frame containing the input data must be in LoadData style <a href="#">LoadData</a> . |
| NLoci | An integer specifying the number of loci to sample.   |

### Value

A data frame containing the sampled loci.

---

|                    |                              |
|--------------------|------------------------------|
| score_markers_rev3 | <i>Score Markers Wrapper</i> |
|--------------------|------------------------------|

---

### Description

This is a revision of the Fragman script score.markers, for the original instructions and parameters, run '?score.markers'. This revision designates separate parameters for Left and Right search windows.

### Usage

```
score_markers_rev3(
  my.inds,
  channel = 1,
  n.inds = NULL,
  panel = NULL,
  shift = 0.8,
  ladder,
  channel.ladder = NULL,
  ploidy = 2,
  left.cond = c(0.6, 3),
  right.cond = 0.35,
```

```

warn = FALSE,
windowL = 0.5,
windowR = 0.5,
init.thresh = 200,
ladd.init.thresh = 200,
method = "iter2",
env = parent.frame(),
my.palette = NULL,
plotting = FALSE,
plotdir = "plots_scoring",
pref = 3
)

```

### Arguments

|                  |  |
|------------------|--|
| my.inds          | The list output from the <code>fsa_batch_imp</code> or <code>storing.inds</code> function that contains the channel information from the individuals that you want to score.   |
| channel          | The number of the channel you wish to analyze. Typically 1 is blue, 2 is green, 3 yellow, and 4 red.   |
| n.inds           | (optional) A vector specifying which fsa files to score.   |
| panel            | A vector containing the expected allele sizes for this marker.   |
| shift            | All peaks at that distance from the tallest peak will be ignored and be considered noise.  |
| ladder           | A vector containing the expected peaks for your ladder.  |
| channel.ladder   | The channel number where your ladder can be found.   |
| ploidy           | The name is a relic of the fact that <code>Fragman::score.markers</code> was originally written for plants. In the context of pooled egg samples it is used to specify the number of possible alleles in the marker.   |
| left.cond        | The first part is a percentile (0-1) that corresponds to the height that a peak to the left of the tallest peak must be in order to be considered real. The second argument is a number of base pairs that a peak to the left of the tallest peak must be away to be considered as real. |
| right.cond       | A percentile (0-1) that corresponds to the height that a peak to the right of the tallest peak must be in order to be real.  |
| warn             | TRUE/FAISE Do you want to receive warnings when detecting the ladder?  |
| windowL          | the window means that all peaks closer by that distance to the left of the panel peaks will be accounted as peaks.   |
| windowR          | the window means that all peaks closer by that distance to the right of the panel peaks will be accounted as peaks.  |
| init.thresh      | A value that sets a minimum intensity in order for a peak to be called.  |
| ladd.init.thresh | We don't recommend messing with this parameter unless your ladder has special circumstances. See <a href="#">Fragman::score.markers</a>  |

|            |   |
|------------|---|
| method     | In cases where samples weren't sized using the <code>info.ladder.attach</code> function, this technique steps in to identify ladder peaks. You have three method options using an argument: "cor" explores all potential peak combinations and thoroughly searches for correlations to identify the correct peaks corresponding to expected DNA weights; "ci" constructs confidence intervals to identify peaks meeting specified conditions from earlier arguments; "iter2" applies an iterative strategy to identify the most likely peaks aligning with your ladder expectations. The default method is "iter2." |
| env        | Please do not change this parameter, it is used to detect the users environment.  |
| my.palette | (optional) A character vector specifying which colors to use for the output RFU plots.  |
| plotting   | TRUE/FALSE Do you want to create pdf output plots?  |
| plotdir    | The name of the directory where output pdf plots should be stored.  |
| pref       | The number of plots to be drawn in the output plot.   |

### Value

The `score_markers_rev3` function will return a list containing three variables: `$pos`, `$hei`, and `$wei`. These correspond to the index position for the intensities, the intensity of each peak, and the weight in base pairs based on the ladder respectively. If `plotting = TRUE`, a pdf file will also have been created in the specified directory. This pdf file allows you to visually inspect how all of the peaks were scored.

### Examples

```
file_path <- system.file("extdata", package = "pooledpeaks")
mock_fsa_batch_imp_output <- fsa_batch_imp(file_path, channels = 5,
fourier = FALSE, saturated = FALSE, lets.pullup = FALSE,
plotting = FALSE, rawPlot = FALSE)
panel <- c(176,179,182,185,188,191,194,197,200,203,206)
ladder <- c( 140, 160, 180, 200, 214, 220,240, 250, 260, 280, 300, 314)
mock_fsa_batch_imp_output <- associate_dyes(mock_fsa_batch_imp_output,
file_path)
score_markers_rev3(my.inds = mock_fsa_batch_imp_output,
channel = 1,
channel.ladder = 5,
panel = "panel",
ladder = ladder,
init.thresh = 200,
ploidy = length(panel),
plotting = FALSE)
```



---

TwoLevelGST

*Calculate Two-Level GST*


---

**Description**

This function calculates two-level GST (Nei's standard gene identity) measure from a gene identity matrix.

**Usage**

```
TwoLevelGST(G = matrix)
```

**Arguments**

G                    A square matrix representing a gene identity matrix.

**Value**

A list containing the components of two-level GST including within-group gene identity, between-group gene identity, and GST values.

**Examples**

```
gene_identity_matrix <- matrix(c(
  0.3164550, 0.2836333, 0.2760485,
  0.2836333, 0.3106084, 0.2867215,
  0.2760485, 0.2867215, 0.3338663
), nrow = 3, byrow = TRUE,
dimnames = list(paste0("Sample", 1:3), paste0("Sample", 1:3)))

TwoLevelGST(G=gene_identity_matrix)
```

---

TypedLoci

*Typed Loci*


---

**Description**

This function calculates the number of loci successfully genotyped by each individual included in our data set

**Usage**

```
TypedLoci(datafile = data.frame)
```

**Arguments**

datafile            A data frame containing the input data must be in LoadData style [LoadData](#).

**Value**

A matrix representing processed data.

**Examples**

```
genetic_data <- data.frame(  
  Locus = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2),  
  Locus_allele = c("Marker1", "n", 1, 2, 3, "Marker2", "n", 1, 2, 3),  
  Sample1 = c(NA, 10, 0.5, 0.5, 0, NA, 10, 0.2, 0.3, 0.5),  
  Sample2 = c(NA, 20, 0.1, 0.2, 0.7, NA, 20, 0.3, 0.4, 0.3),  
  Sample3 = c(NA, 30, 0.3, 0.4, 0.3, NA, 30, 0.4, 0.2, 0.4)  
)  
TypedLoci(datafile=genetic_data)
```

# Index

AlRich, [3](#)  
associate\_dyes, [4](#)  
  
BootStrap3, [4](#)  
  
check\_fsa\_v\_batch, [5](#)  
clean\_scores, [6](#)  
cluster, [7](#)  
ClusterFromSamples, [8](#)  
  
data\_manipulation, [8](#)  
DistCor, [9](#)  
  
EmpiricalSE, [10](#)  
  
Fragman::score.markers, [23](#)  
fsa\_batch\_imp, [11](#)  
fsa\_metadata, [12](#)  
  
GeneIdentityMatrix, [13](#), [14](#)  
GeneticDistanceMatrix, [14](#), [21](#)  
GST, [14](#)  
  
JostD, [15](#)  
JostD\_KK, [16](#)  
  
lf\_to\_tdf, [16](#)  
LoadData, [3](#), [4](#), [7](#), [8](#), [10](#), [13](#), [17](#), [22](#), [25](#)  
  
MDSplot, [18](#)  
  
PCDM, [19](#)  
preGST, [19](#)  
preJostD, [20](#)  
  
Rep\_check, [20](#)  
RWCDistanceMatrix, [21](#)  
  
SampleOfLoci, [22](#)  
score\_markers\_rev3, [22](#)  
  
TwoLevelGST, [25](#)  
TypedLoci, [3](#), [25](#)