# Package 'powdR'

October 14, 2022

**Type** Package

**Title** Full Pattern Summation of X-Ray Powder Diffraction Data

**Version** 1.3.0

**Date** 2021-08-13

**Maintainer** Benjamin Butler <benjamin.butler@hutton.ac.uk>

**Description** Full pattern summation of X-ray powder diffraction data as described in Chipera and Bish (2002) <doi:10.1107/S0021889802017405> and Butler and Hillier (2021) <doi:10.1016/j.cageo.2020.104662>. Derives quantitative estimates of crystalline and amorphous phase concentrations in complex mixtures.

**License** GPL-2 | file LICENSE

**URL** https://github.com/benmbutler/powdR

**BugReports** https://github.com/benmbutler/powdR/issues

**Depends** R (>= 3.2.0)

**Encoding** UTF-8

**LazyData** true

**Imports** plyr (>= 1.8.6), reshape (>= 0.8.8), plotly (>= 4.9.2.1), ggplot2 (>= 3.3.0), stats (>= 3.4.3), utils (>= 3.4.3), ggpubr (>= 0.2.5), shiny (>= 1.4.0.2), DT (>= 0.13), nnls (>= 1.4), shinyWidgets (>= 0.5.1), baseline (>= 1.2), tidyr (>= 1.0.2), FactoMineR (>= 2.3), factoextra (>= 1.0.7), rxylib (>= 0.2.6)

**Suggests** knitr, rmarkdown, bookdown

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Benjamin Butler [aut, cre],
Stephen Hillier [aut],
Dylan Beaudette [ctb],
Dennis Eberl [ctb]

**Repository** CRAN

**Date/Publication** 2021-08-13 15:20:02 UTC

# R topics documented:

---

`afps` *Automated full pattern summation*

---

## Description

`afps` returns estimates of phase concentrations using automated full pattern summation of X-ray powder diffraction data. It is designed for high-throughput cases involving mineral quantification from large reference libraries.

## Usage

```
afps(
  lib,
  smpl,
  harmonise,
  solver,
  obj,
  refs,
  std,
  force,
  std_conc,
  omit_std,
  closed,
  normalise,
  tth_align,
  align,
  manual_align,
  shift,
  tth_fps,
```

```
  lod,
  amorphous,
  amorphous_lod,
  weighting,
  ...
)
```

## Arguments

| | |
|---|---|
| lib | A powdRlib object representing the reference library. Created using the powdRlib constructor function. |
| smpl | A data frame. First column is 2theta, second column is counts |
| harmonise | logical parameter defining whether to harmonise the lib and smpl. Default = TRUE. Harmonises to the intersecting 2theta range at the coarsest resolution available using natural splines. |
| solver | The optimisation routine to be used. One of "BFGS", "Nelder-Mead",or "CG". Default = "BFGS". |
| obj | The objective function to minimise. One of "Delta", "R", "Rwp". Default = "Rwp". See Chipera and Bish (2002) and page 247 of Bish and Post (1989) for definitions of these functions. |
| refs | A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. If missing from the function call then all phases in the reference library will be used. |
| std | The phase ID (e.g. "QUA.1") to be used as internal standard. Must match an ID provided in the refs parameter. |
| force | An optional string of phase ID's or names specifying which phases should be forced to remain throughout the automated full pattern summation. The ID's or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. |
| std_conc | The concentration of the internal standard (if known) in weight percent. If unknown then either omit the argument from the function call of use std_conc = NA, in which case it will be assumed that all phases sum to 100 percent (default). |
| omit_std | A logical parameter to be used when the std_conc argument is defined. When omit_std = TRUE the phase concentrations are recomputed to account for value supplied in std_conc. Default = FALSE. |
| closed | A logical parameter to be used when the std_conc argument is defined and omit_std = TRUE. When closed = TRUE the internal standard concentration is removed and the remaining phase concentrations closed to sum to 100 percent. Default = FALSE. |
| normalise | deprecated. Please use the omit_std and closed arguments instead. |
| tth_align | A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. c(5,65)). If not defined, then the full range is used. |
| align | The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1. |

| | |
|---|---|
| manual_align | A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised. |
| shift | A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of selected phases. Default = 0. |
| tth_fps | A vector defining the minimum and maximum 2theta values to be used during automated full pattern summation (e.g. c(5,65)). If not defined, then the full range is used. |
| lod | Optional parameter used to define the limit of detection (in weight percent) of the internal standard (i.e. the phase provided in the std argument). The lod value is used to estimate the lod of other phases during the fitting process and hence remove reference patterns that are considered below detection limit. Default = 0.1. If lod = 0 then limits of detection are not computed. |
| amorphous | A character string of any phase IDs that should be treated as amorphous. These must match phases present in lib$phases$phase_id. |
| amorphous_lod | Optional parameter used to exclude amorphous phases if they are below this specified limit (percent). Must be between 0 and 100. Default = 0. |
| weighting | an optional 2 column data frame specifying the 2theta values in the first column and a numeric weighting vector in the second column that specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) when minimising the objective function defined in the obj argument. Use this weighting parameter with caution. The default is simply a weighting vector where all values are 1, which hence has no effect on the computed objective function. |
| ... | Other parameters passed to methods e.g. afps.powdRlib |

## Details

Applies automated full pattern summation to an XRPD measurement to quantify phase concentrations. Requires a powdRlib library of reference patterns with reference intensity ratios in order to derive mineral concentrations. Details provided in Butler and Hillier (2021).

## Value

a powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the count intensities of fitted XRPD pattern |
| measured | a vector of the count intensities of original XRPD measurement (aligned) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern |
| phases_grouped | the phases dataframe grouped and summed by phase_name |
| obj | named vector of the objective parameters summarising the quality of the fit |

`weighted_pure_patterns`
: a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit

`coefficients`
: a named vector of coefficients used to produce the fitted pattern

`inputs`
: a list of input arguments used in the function call

## References

Butler, B. M., Hillier, S., 2021.powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Comp. Geo. 147, 104662. doi:10.1016/j.cageo.2020.104662

Chipera, S.J., Bish, D.L., 2013. Fitting Full X-Ray Diffraction Patterns for Quantitative Analysis: A Method for Readily Quantifying Crystalline and Disordered Phases. Adv. Mater. Phys. Chem. 03, 47-53. doi:10.4236/ampc.2013.31A007

Chipera, S.J., Bish, D.L., 2002. FULLPAT: A full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns. J. Appl. Crystallogr. 35, 744-749. doi:10.1107/S0021889802017405

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

## Not run:
afps_sand <-  afps(lib = minerals,
                   smpl = soils$sandstone,
                   std = "QUA.2",
                   align = 0.2,
                   lod = 0.2,
                   amorphous = "ORG",
                   amorphous_lod = 1)

afps_lime <- afps(lib = minerals,
                  smpl = soils$limestone,
                  std = "QUA.2",
                  align = 0.2,
                  lod = 0.2,
                  amorphous = "ORG",
                  amorphous_lod = 1)

afps_granite <- afps(lib = minerals,
                     smpl = soils$granite,
                     std = "QUA.2",
                     align = 0.2,
                     lod = 0.2,
                     amorphous = "ORG",
```

```
                    amorphous_lod = 1)

#Alternatively run all 3 at once using lapply

afps_soils <- lapply(soils, afps,
                     lib = minerals,
                     std = "QUA.2",
                     align = 0.2,
                     lod = 0.2,
                     amorphous = "ORG",
                     amorphous_lod = 1)

#Automated quantification using the rockjock library

data(rockjock)
data(rockjock_mixtures)

#This takes a few minutes to run
rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1)

#Quantifying the same sample but defining the internal standard
#concentration (also takes a few minutes to run):
rockjock_a1s <- afps(lib = rockjock,
                     smpl = rockjock_mixtures$Mix1,
                     std = "CORUNDUM",
                     std_conc = 20,
                     align = 0.3,
                     lod = 1)


## End(Not run)
```

---

`afps.powdRlib` *Automated full pattern summation*

---

## Description

`afps` returns estimates of phase concentrations using automated full pattern summation of X-ray powder diffraction data. It is designed for high-throughput cases involving mineral quantification from large reference libraries.

## Usage

```
## S3 method for class 'powdRlib'
afps(
  lib,
```

```
  smpl,
  harmonise,
  solver,
  obj,
  refs,
  std,
  force,
  std_conc,
  omit_std,
  closed,
  normalise,
  tth_align,
  align,
  manual_align,
  shift,
  tth_fps,
  lod,
  amorphous,
  amorphous_lod,
  weighting,
  ...
)
```

## Arguments

| | |
|---|---|
| lib | A powdRlib object representing the reference library. Created using the powdRlib constructor function. |
| smpl | A data frame. First column is 2theta, second column is counts |
| harmonise | logical parameter defining whether to harmonise the lib and smpl. Default = TRUE. Harmonises to the intersecting 2theta range at the coarsest resolution available using natural splines. |
| solver | The optimisation routine to be used. One of "BFGS", "Nelder-Mead",or "CG". Default = "BFGS". |
| obj | The objective function to minimise. One of "Delta", "R", "Rwp". Default = "Rwp". See Chipera and Bish (2002) and page 247 of Bish and Post (1989) for definitions of these functions. |
| refs | A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. If missing from the function call then all phases in the reference library will be used. |
| std | The phase ID (e.g. "QUA.1") to be used as internal standard. Must match an ID provided in the refs parameter. |
| force | An optional string of phase ID's or names specifying which phases should be forced to remain throughout the automated full pattern summation. The ID's or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. |

| | |
|---|---|
| std_conc | The concentration of the internal standard (if known) in weight percent. If unknown then either omit the argument from the function call of use std_conc = NA, in which case it will be assumed that all phases sum to 100 percent (default). |
| omit_std | A logical parameter to be used when the std_conc argument is defined. When omit_std = TRUE the phase concentrations are recomputed to account for value supplied in std_conc. Default = FALSE. |
| closed | A logical parameter to be used when the std_conc argument is defined and omit_std = TRUE. When closed = TRUE the internal standard concentration is removed and the remaining phase concentrations closed to sum to 100 percent. Default = FALSE. |
| normalise | deprecated. Please use the omit_std and closed arguments instead. |
| tth_align | A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. c(5,65)). If not defined, then the full range is used. |
| align | The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1. |
| manual_align | A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised. |
| shift | A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of selected phases. Default = 0. |
| tth_fps | A vector defining the minimum and maximum 2theta values to be used during automated full pattern summation (e.g. c(5,65)). If not defined, then the full range is used. |
| lod | Optional parameter used to define the limit of detection (in weight percent) of the internal standard (i.e. the phase provided in the std argument). The lod value is used to estimate the lod of other phases during the fitting process and hence remove reference patterns that are considered below detection limit. Default = 0.1. If lod = 0 then limits of detection are not computed. |
| amorphous | A character string of any phase IDs that should be treated as amorphous. These must match phases present in lib$phases$phase_id. |
| amorphous_lod | Optional parameter used to exclude amorphous phases if they are below this specified limit (percent). Must be between 0 and 100. Default = 0. |
| weighting | an optional 2 column data frame specifying the 2theta values in the first column and a numeric weighting vector in the second column that specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) when minimising the objective function defined in the obj argument. Use this weighting parameter with caution. The default is simply a weighting vector where all values are 1, which hence has no effect on the computed objective function. |
| ... | other arguments |

## Details

Applies automated full pattern summation to an XRPD measurement to quantify phase concentrations. Requires a `powdRlib` library of reference patterns with reference intensity ratios in order to derive mineral concentrations. Details provided in Butler and Hillier (2021).

## Value

a powdRafps object with components:

| | |
|---|---|
| `tth` | a vector of the 2theta scale of the fitted data |
| `fitted` | a vector of the count intensities of fitted XRPD pattern |
| `measured` | a vector of the count intensities of original XRPD measurement (aligned) |
| `residuals` | a vector of the residuals (measured minus fitted) |
| `phases` | a dataframe of the phases used to produce the fitted pattern |
| `phases_grouped` | the phases dataframe grouped and summed by phase_name |
| `obj` | named vector of the objective parameters summarising the quality of the fit |
| `weighted_pure_patterns` | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| `coefficients` | a named vector of coefficients used to produce the fitted pattern |
| `inputs` | a list of input arguments used in the function call |

## References

Butler, B. M., Hillier, S., 2021.powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Comp. Geo. 147, 104662. doi:10.1016/j.cageo.2020.104662

Bish, D.L., Post, J.E., 1989. Modern powder diffraction. Mineralogical Society of America.

Chipera, S.J., Bish, D.L., 2013. Fitting Full X-Ray Diffraction Patterns for Quantitative Analysis: A Method for Readily Quantifying Crystalline and Disordered Phases. Adv. Mater. Phys. Chem. 03, 47-53. doi:10.4236/ampc.2013.31A007

Chipera, S.J., Bish, D.L., 2002. FULLPAT: A full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns. J. Appl. Crystallogr. 35, 744-749. doi:10.1107/S0021889802017405

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

## Not run:
afps_sand <-  afps(lib = minerals,
```

```
                  smpl = soils$sandstone,
                  std = "QUA.2",
                  align = 0.2,
                  lod = 0.2,
                  amorphous = "ORG",
                  amorphous_lod = 1)

afps_lime <- afps(lib = minerals,
                  smpl = soils$limestone,
                  std = "QUA.2",
                  align = 0.2,
                  lod = 0.2,
                  amorphous = "ORG",
                  amorphous_lod = 1)

afps_granite <- afps(lib = minerals,
                     smpl = soils$granite,
                     std = "QUA.2",
                     align = 0.2,
                     lod = 0.2,
                     amorphous = "ORG",
                     amorphous_lod = 1)

#Alternatively run all 3 at once using lapply

afps_soils <- lapply(soils, afps,
                      lib = minerals,
                      std = "QUA.2",
                      align = 0.2,
                      lod = 0.2,
                      amorphous = "ORG",
                      amorphous_lod = 1)

#Automated quantification using the rockjock library

data(rockjock)
data(rockjock_mixtures)

#This takes a few minutes to run
rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1)

#Quantifying the same sample but defining the internal standard
#concentration (also takes a few minutes to run):
rockjock_a1s <- afps(lib = rockjock,
                     smpl = rockjock_mixtures$Mix1,
                     std = "CORUNDUM",
                     std_conc = 20,
                     align = 0.3,
                     lod = 1)
```

```
## End(Not run)
```

---

`afsis` *Africa Soil Information Service (AfSIS) XRPD reference library*

---

## Description

A `powdRlib` object of 21 pure reference patterns and associated reference intensity ratios for a range of common soil minerals. Data were collected on a Bruker D2 Phaser using Cu K-alpha radiation. All patterns have been normalised to 10,000 counts and reference intensity ratios transformed so that all are relative to that of corundum.

## Usage

```
afsis
```

## Format

A powdRlib object of 3 components

**xrd** A dataframe of all the count intensities of all reference patterns. Column names denote the unique phase ID of each reference pattern

**tth** A vector of the 2theta scale for all reference patterns in the library

**phases** A dataframe the phase IDs, names and reference intensity ratios (RIR)

---

`afsis_codes` *Original codes for the* `afsis` *reference patterns*

---

## Description

A data frame detailing the original codes associated with the `afsis` reference patterns prior to their addition to powdR.

## Usage

```
afsis_codes
```

## Format

An 2 column data frame. First column contains the phase IDs from `afsis$phase_id` and the second column the original IDs prior to the inclusion in powdR.

## afsis_regroup — *Regrouping structure for the Africa Soil Information Service (AfSIS) XRPD reference library*

### Description

A data frame containing an example re-grouping structure for the `afsis` reference library, which results in a slightly coarser description of clay minerals and Fe/Ti-(hydr)oxides in `powdRfps` or `powdRafps` objects when used with `regroup()`.

### Usage

```
afsis_regroup
```

### Format

A data frame with three columns:

**phase_id** the phase IDs present in `afsis$phases$phase_id`.

**phase_name_grouped** The phase names that constitute the first regrouping structure.

**phase_name_grouped2** The phase names that constitute the second regrouping structure

## align_xy — *Align XRPD data to a given standard*

### Description

See `?align_xy.XY` and `align_xy.multiXY` for method-specific details.

### Usage

```
align_xy(x, std, xmin, xmax, xshift, ...)
```

### Arguments

| | |
|---|---|
| `x` | an `XY` or `multiXY` object. |
| `std` | a dataframe of the chosen standard that each sample will be aligned to (column 1 = 2theta, column 2 = counts) |
| `xmin` | the minimum 2theta value used during alignment |
| `xmax` | the maximum 2theta value used during alignment |
| `xshift` | the maximum (positive and negative) 2theta shift that is allowed during alignment |
| `...` | other arguments |

## Value

an XY or multiXY object.

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:
#Create a standard quartz pattern to align to
quartz <- data.frame(tth = minerals$tth,
                     counts = minerals$xrd$QUA.1)

#Plot the main quartz peak prior to alignment
plot(soils, wavelength = "Cu",
     xlim = c(26,27),
     normalise = TRUE)

#align data
aligned <- align_xy(soils,
                    std = quartz,
                    xmin = 10,
                    xmax = 60,
                    xshift = 0.2)

#replot data
plot(aligned, wavelength = "Cu",
     xlim = c(26,27),
     normalise = TRUE)

#Alternatively try with a single XY object

unaligned <- as_multi_xy(list("quartz" = quartz,
                              "sandstone" = soils$sandstone))

plot(unaligned, wav = "Cu",
     xlim = c(26,27), normalise = TRUE)

sandstone_a <- align_xy(soils$sandstone,
                        std = quartz,
                        xmin = 10,
                        xmax = 60,
                        xshift = 0.3)

aligned <- as_multi_xy(list("quartz" = quartz,
                            "sandstone" = sandstone_a))

plot(aligned, wav = "Cu",
     xlim = c(26,27), normalise = TRUE)
```

```
## End(Not run)
```

---

`align_xy.multiXY` *Align XRPD data in a multiXY object to a given standard*

---

## Description

`align_xy.multiXY` takes a multiXY object and aligns each of the XY data frames within it to a given standard. An optimisation routine is used that computes a suitable linear shift. After all samples have been aligned, the function harmonises the data to a single 2theta scale.

## Usage

```
## S3 method for class 'multiXY'
align_xy(x, std, xmin, xmax, xshift, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `multiXY` object. |
| `std` | a dataframe of the chosen standard that each sample will be aligned to (column 1 = 2theta, column 2 = counts) |
| `xmin` | the minimum 2theta value used during alignment |
| `xmax` | the maximum 2theta value used during alignment |
| `xshift` | the maximum (positive and negative) 2theta shift that is allowed during alignment |
| `...` | other arguments |

## Value

a `multiXY` object.

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:
#Create a standard quartz pattern to align to
quartz <- data.frame(tth = minerals$tth,
                     counts = minerals$xrd$QUA.1)
```

```
#Plot the main quartz peak prior to alignment
plot(soils, wavelength = "Cu",
     xlim = c(26,27),
     normalise = TRUE)

#align data
aligned <- align_xy(soils,
                    std = quartz,
                    xmin = 10,
                    xmax = 60,
                    xshift = 0.2)

#replot data
plot(aligned, wavelength = "Cu",
     xlim = c(26,27),
     normalise = TRUE)


## End(Not run)
```

---

`align_xy.XY` *Align XRPD data in an XY object to a given standard*

---

## Description

`align_xy.XY` takes an XY object and aligns it to a given standard. An optimisation routine is used that computes a suitable linear shift.

## Usage

```
## S3 method for class 'XY'
align_xy(x, std, xmin, xmax, xshift, ...)
```

## Arguments

| | |
|---|---|
| `x` | an `XY` object. |
| `std` | a dataframe of the chosen standard that each sample is aligned to (column 1 = 2theta, column 2 = counts) |
| `xmin` | the minimum 2theta value used during alignment |
| `xmax` | the maximum 2theta value used during alignment |
| `xshift` | the maximum (positive and negative) 2theta shift that is allowed during alignment |
| `...` | other arguments |

## Value

an `XY` object.

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:

#Create a standard quartz pattern to align to
quartz <- data.frame(tth = minerals$tth,
                     counts = minerals$xrd$QUA.1)

unaligned <- as_multi_xy(list("quartz" = quartz,
                              "sandstone" = soils$sandstone))

plot(unaligned, wav = "Cu",
     xlim = c(26,27), normalise = TRUE)

sandstone_a <- align_xy(soils$sandstone,
                        std = quartz,
                        xmin = 10,
                        xmax = 60,
                        xshift = 0.3)

aligned <- as_multi_xy(list("quartz" = quartz,
                            "sandstone" = sandstone_a))

plot(aligned, wav = "Cu",
     xlim = c(26,27), normalise = TRUE)


## End(Not run)
```

---

# as_multi_xy *Create a multiXY object*

---

## Description

`as_multi_xy` takes a list or data frame of XRPD data and ensures that the data meet various requirements to create a multiXY object. Once a multiXY object has been created, it can easily be plotted using the associated `plot.multiXY` method.

## Usage

```
as_multi_xy(x, ...)
```

### Arguments

| | |
|---|---|
| x | a list or data frame of XRPD data |
| ... | other arguments |

### Value

a multiXY object.

### Examples

```
#EXAMPLE 1

#load soils data
data(soils)

#extract first two samples from the list
soils <- soils[c(1:2)]

#convert to multiXY
soils <- as_multi_xy(soils)

#EXAMPLE 2
#load the soils data
data(soils)

#Convert to data frame
soils_df <- multi_xy_to_df(soils,
                           tth = TRUE)

#Convert back to multiXY object
soils2 <- as_multi_xy(soils_df)
```

---

## as_multi_xy.data.frame

*Create a multiXY object from a list of XRPD data*

---

### Description

as_multi_xy.data.frame takes a data frame of XRPD data from multiple samples and ensures that it meets various requirements to create a multiXY object. Once a multiXY object has been created, it can easily be plotted using the associated plot.multiXY method.

### Usage

```
## S3 method for class 'data.frame'
as_multi_xy(x, ...)
```

### Arguments

| | |
|---|---|
| x | a data frame of XRPD data, with the first column as the 2theta axis and subsequent columns of count intensities. |
| ... | other arguments |

### Value

a multiXY object.

### Examples

```
#load the soils data
data(soils)

#Convert to data frame
soils_df <- multi_xy_to_df(soils,
                           tth = TRUE)

#Convert back to multiXY object
soils2 <- as_multi_xy(soils_df)
```

---

## as_multi_xy.list *Create a multiXY object from a list of XRPD data*

---

### Description

as_multi_xy.list takes a list of XRPD data and ensures that they meet various requirements to create a multiXY object. These requirements include that each item in the list contains 2 columns of numeric data in a data frame. as_multi_xy.list also checks that all names are unique. Once a multiXY object has been created, it can easily be plotted using the associated plot.multiXY method.

### Usage

```
## S3 method for class 'list'
as_multi_xy(x, ...)
```

### Arguments

| | |
|---|---|
| x | a list of XRPD data frames (column 1 = 2theta, column 2 = counts) |
| ... | other arguments |

### Value

a multiXY object.

## Examples

```
#' #load soils data
data(soils)

#extract first two samples from the list
soils <- soils[c(1:2)]

#convert to multiXY
soils <- as_multi_xy(soils)
```

---

as_xy *Create an XY object*

---

## Description

as_xy takes a data frame of XY XRPD data and ensures that it meets the criteria for an XY object. These requirements include that the data contains 2 columns of numeric data in a dataframe. Once an XY object has been created, it can easily be plotted using the associated plot.XY method.

## Usage

```
as_xy(x)
```

## Arguments

| | |
|---|---|
| x | a data frame (column 1 = 2theta, column 2 = counts) |

## Value

an XY object.

## Examples

```
# Load soils xrd data
data(rockjock_mixtures)

xy <- as_xy(rockjock_mixtures$Mix1)

class(xy)

## Not run:
plot(xy, wavelength = "Cu")
plot(xy, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

bkg *Fit a background to XRPD data*

## Description

bkg fits a background to X-Ray Powder Diffraction data

## Usage

```
bkg(xrd, lambda, hwi, it, int)
```

## Arguments

| | |
|---|---|
| xrd | an xy data frame of the data to fit a background to. First column is the 2theta scale, second column is count intensities |
| lambda | second derivative penalty for primary smoothing. Default = 0.5. |
| hwi | Half width of local windows. Default = 25. |
| it | Number of iterations in suppression loop. Default = 50. |
| int | Number of buckets to divide the data into. Default = round(nrow(xrd)/4). |

## Details

A wrapper for the baseline.fillPeaks in the baseline package.

## Value

a powdRbkg object consisting of of 3 vectors

| | |
|---|---|
| tth | The 2theta axis of the measurement |
| counts | The count intensities of the measurement |
| background | The count intensities of the fitted background |

## Examples

```
data(soils)
## Not run:
fit_bkg <- bkg(soils$granite)
plot(bkg)

## End(Not run)
```

close_quant *Close the phase concentration data within a* powdRfps *or* powdRafps *object*

## Description

close_quant closes the quantitative data within a powdRfps or powdRafps object (derived from fps() and afps(), respectively) by ensuring that the composition sums to 100 percent. See also ?close_quant.powdRfps and ?close_quant.powdRafps.

## Usage

```
close_quant(x, ...)
```

## Arguments

| | |
|---|---|
| x | A powdRfps or powdRafps object.. |
| ... | other arguments |

## Value

a powdRfps or powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (fitted vs measured) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
```

```
                    refs = c("ORDERED_MICROCLINE",
                             "LABRADORITE",
                             "KAOLINITE_DRY_BRANCH",
                             "MONTMORILLONITE_WYO",
                             "ILLITE_1M_RM30",
                             "CORUNDUM"),
                   std = "CORUNDUM",
                   align = 0.3,
                   std_conc = 20)

sum(rockjock_1$phases$phase_percent)

rockjock_1c <- close_quant(rockjock_1)

sum(rockjock_1c$phases$phase_percent)

rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1,
                    std_conc = 20)

sum(rockjock_a1$phases$phase_percent)

rockjock_a1c <- close_quant(rockjock_a1)

sum(rockjock_a1c$phases$phase_percent)


## End(Not run)
```

---

`close_quant.powdRafps` *Close the phase concentration data within a* `powdRafps` *object*

---

## Description

`close_quant` closes the quantitative data within a `powdRafps` object (derived from `afps()`) by ensuring that the composition sums to 100 percent.

## Usage

```
## S3 method for class 'powdRafps'
close_quant(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `powdRafps` object derived from `afps()`. |
| `...` | other arguments |

## Value

a powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (fitted vs measured) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1,
                    std_conc = 20)

sum(rockjock_a1$phases$phase_percent)

rockjock_a1c <- close_quant(rockjock_a1)

sum(rockjock_a1c$phases$phase_percent)


## End(Not run)
```

---

# close_quant.powdRfps *Close the phase concentration data within a* powdRfps *object*

---

## Description

close_quant closes the quantitative data within a powdRfps object (derived from fps()) by ensuring that the composition sums to 100 percent.

## Usage

```
## S3 method for class 'powdRfps'
close_quant(x, ...)
```

## Arguments

| | |
|---|---|
| x | A powdRfps object derived from fps(). |
| ... | other arguments |

## Value

a powdRfps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (fitted vs measured) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
                  refs = c("ORDERED_MICROCLINE",
                           "LABRADORITE",
                           "KAOLINITE_DRY_BRANCH",
                           "MONTMORILLONITE_WYO",
                           "ILLITE_1M_RM30",
                           "CORUNDUM"),
                 std = "CORUNDUM",
                 align = 0.3,
                 std_conc = 20)

sum(rockjock_1$phases$phase_percent)
```

```
rockjock_1c<- close_quant(rockjock_1)

sum(rockjock_1c$phases$phase_percent)


## End(Not run)
```

---

`delta` *Calculate the Delta value for a fitted pattern*

---

## Description

`delta` computes the absolute difference between a measured and fitted pattern. See equation for Delta in section 2.1 of Butler and Hillier (2021).

## Usage

```
delta(measured, fitted, weighting)
```

## Arguments

| | |
|---|---|
| `measured` | a vector of count intensities for a measured pattern |
| `fitted` | a vector of count intensities for a fitted pattern |
| `weighting` | an optional weighting vector of the same length as those specified in `measured` and `fitted`, which specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) from the calculation. Use with caution. Default is simply a weighting vector where all values are 1, which hence has no effect on the computed value. |

## Value

a single numeric value

## References

Butler, B.M., Hillier, S., 2021. powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Computers and Geosciences. 147, 104662. doi:10.1016/j.cageo.2020.104662

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:
```

```
#Produce a fit
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)

delta(measured = fps_sand$measured,
      fitted = fps_sand$fitted)


## End(Not run)
```

---

`extract_xy` *Import and extract XY data from proprietary files*

---

## Description

`extract_xy` is a wrapper for `read_xyData` of the `rxylib` package, which extracts the xy data from various proprietary formats of X-ray powder diffraction data using the `xylib` C++ library. For more information see `?rxylib` and `?rxylib::read_xyData`.

## Usage

```
extract_xy(files)
```

## Arguments

| | |
|---|---|
| `files` | path of the file(s) to be imported. |

## Value

If only one path is supplied then an XY data frame with 2 columns is returned, the first being the 2theta axis and the second being the count intensities. If more than one path is supplied then a multiXY list is returned, with each item in the list being an XY data frame as already described.

## Examples

```
#load example RAW file
file <- system.file("extdata/D5000/RAW/D5000_1.RAW", package = "powdR")
raw1 <- extract_xy(file)

#Load multiple RAW files
files <- dir(system.file("extdata/D5000/RAW", package = "powdR"),
             full.names = TRUE)
raw_list <- extract_xy(files)

class(raw_list)
```

```
## Not run:
plot(raw_list, wavelength = "Cu")
plot(raw_list, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

---

`fps` *Full pattern summation*

---

## Description

`fps` returns estimates of phase concentrations using full pattern summation of X-ray powder diffraction data.

## Usage

```
fps(
  lib,
  smpl,
  harmonise,
  solver,
  obj,
  refs,
  std,
  force,
  std_conc,
  omit_std,
  normalise,
  closed,
  tth_align,
  align,
  manual_align,
  tth_fps,
  shift,
  remove_trace,
  weighting,
  ...
)
```

## Arguments

| | |
|---|---|
| `lib` | A `powdRlib` object representing the reference library. Created using the `powdRlib` constructor function. |
| `smpl` | A data frame. First column is 2theta, second column is counts |

harmonise — logical parameter defining whether to harmonise the lib and smpl. Default = TRUE. When TRUE the function will harmonise the lib and smpl to the intersecting 2theta range at the coarsest resolution available using natural splines.

solver — The optimisation routine to be used. One of "BFGS", "Nelder-Mead","CG", "NNLS". Default = "BFGS".

obj — The objective function to minimise when "BFGS", "Nelder-Mead", or "CG" are used as the solver argument. One of "Delta", "R", "Rwp". Default = "Rwp". See Chipera and Bish (2002) and page 247 of Bish and Post (1989) for definitions of these functions.

refs — A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. If missing from the function call then all phases in the reference library will be used.

std — The phase ID (e.g. "QUA.1") to be used as an internal standard. Must match an ID provided in the refs parameter.

force — An optional string of phase IDs or names specifying which phases should be forced to remain throughout the automated full pattern summation. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns.

std_conc — The concentration of the internal standard (if known) in weight percent. If unknown then omit the argument from the function call or use std_conc = NA (default), n which case it will be assumed that all phases sum to 100 percent.

omit_std — A logical parameter to be used when the std_conc argument is defined. When omit_std = TRUE the phase concentrations are recomputed to account for the value supplied in std_conc. Default = FALSE.

normalise — deprecated. Please use the omit_std and closed arguments instead.

closed — A logical parameter to be used when the std_conc argument is defined and omit_std = TRUE. When closed = TRUE the internal standard concentration is removed and the remaining phase concentrations closed to sum to 100 percent. Default = FALSE.

tth_align — A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. c(5,65)). If not defined, then the full range is used.

align — The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1.

manual_align — A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised.

tth_fps — A vector defining the minimum and maximum 2theta values to be used during full pattern summation (e.g. c(5,65)). If not defined, then the full range is used.

shift — A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of reference patterns. Default = 0.

| | |
|---|---|
| remove_trace | A single numeric value representing the limit for the concentration of trace phases to be retained, i.e. any mineral with an estimated concentration below remove_trace will be omitted. Default = 0. |
| weighting | an optional 2 column data frame specifying the 2theta values in the first column and a numeric weighting vector in the second column that specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) when minimising the objective function defined in the obj argument. Use this weighting parameter with caution. The default is simply a weighting vector where all values are 1, which hence has no effect on the computed objective function. |
| ... | Other parameters passed to methods e.g. fps.powdRlib |

## Details

Applies full pattern summation (Chipera & Bish, 2002, 2013; Eberl, 2003) to an XRPD measurement to quantify phase concentrations. Requires a powdRlib library of reference patterns with reference intensity ratios in order to derive mineral concentrations. Details provided in Butler and Hillier (2021).

## Value

a powdRfps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## References

Butler, B. M., Hillier, S., 2021.powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Comp. Geo. 147, 104662. doi:10.1016/j.cageo.2020.104662

Chipera, S.J., Bish, D.L., 2013. Fitting Full X-Ray Diffraction Patterns for Quantitative Analysis: A Method for Readily Quantifying Crystalline and Disordered Phases. Adv. Mater. Phys. Chem. 03, 47-53. doi:10.4236/ampc.2013.31A007

Chipera, S.J., Bish, D.L., 2002. FULLPAT: A full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns. J. Appl. Crystallogr. 35, 744-749. doi:10.1107/S0021889802017405

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

#Since the reference library is relatively small,
#the whole library can be used at once to get an
#estimate of the phases within each sample.
## Not run:
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)

fps_lime <- fps(lib = minerals,
                smpl = soils$limestone,
                refs = minerals$phases$phase_id,
                std = "QUA.1",
                align = 0.2)

fps_granite <- fps(lib = minerals,
                   smpl = soils$granite,
                   refs = minerals$phases$phase_id,
                   std = "QUA.1",
                   align = 0.2)

#Alternatively run all 3 at once using lapply

fps_soils <- lapply(soils, fps,
                    lib = minerals,
                    std = "QUA.2",
                    refs = minerals$phases$phase_id,
                    align = 0.2)

#Using the rockjock library:

data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
                  refs = c("ORDERED_MICROCLINE",
                           "LABRADORITE",
                           "KAOLINITE_DRY_BRANCH",
                           "MONTMORILLONITE_WYO",
                           "ILLITE_1M_RM30",
```

```
                    "CORUNDUM"),
                std = "CORUNDUM",
                align = 0.3)

#Alternatively you can specify the internal standard
#concentration if known:
rockjock_1s <- fps(lib = rockjock,
                smpl = rockjock_mixtures$Mix1,
                refs = c("ORDERED_MICROCLINE",
                         "LABRADORITE",
                         "KAOLINITE_DRY_BRANCH",
                         "MONTMORILLONITE_WYO",
                         "ILLITE_1M_RM30",
                         "CORUNDUM"),
                std = "CORUNDUM",
                std_conc = 20,
                align = 0.3)


## End(Not run)
```

---

`fps.powdRlib` *Full pattern summation*

---

## Description

`fps.powdRlib` returns estimates of phase concentrations using full pattern summation of X-ray powder diffraction data.

## Usage

```
## S3 method for class 'powdRlib'
fps(
  lib,
  smpl,
  harmonise,
  solver,
  obj,
  refs,
  std,
  force,
  std_conc,
  omit_std,
  normalise,
  closed,
  tth_align,
  align,
  manual_align,
  tth_fps,
```

```
  shift,
  remove_trace,
  weighting,
  ...
)
```

## Arguments

| | |
|---|---|
| lib | A powdRlib object representing the reference library. Created using the powdRlib constructor function. |
| smpl | A data frame. First column is 2theta, second column is counts |
| harmonise | logical parameter defining whether to harmonise the lib and smpl. Default = TRUE. When TRUE the function will harmonise the lib and smpl to the intersecting 2theta range at the coarsest resolution available using natural splines. |
| solver | The optimisation routine to be used. One of "BFGS", "Nelder-Mead","CG", "NNLS". Default = "BFGS". |
| obj | The objective function to minimise when "BFGS", "Nelder-Mead", or "CG" are used as the solver argument. One of "Delta", "R", "Rwp". Default = "Rwp". See Chipera and Bish (2002) and page 247 of Bish and Post (1989) for definitions of these functions. |
| refs | A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. If missing from the function call then all phases in the reference library will be used. |
| std | The phase ID (e.g. "QUA.1") to be used as an internal standard. Must match an ID provided in the refs parameter. |
| force | An optional string of phase IDs or names specifying which phases should be forced to remain throughout the automated full pattern summation. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. |
| std_conc | The concentration of the internal standard (if known) in weight percent. If unknown then omit the argument from the function call or use std_conc = NA (default), n which case it will be assumed that all phases sum to 100 percent. |
| omit_std | A logical parameter to be used when the std_conc argument is defined. When omit_std = TRUE the phase concentrations are recomputed to account for the value supplied in std_conc. Default = FALSE. |
| normalise | deprecated. Please use the omit_std and closed arguments instead. |
| closed | A logical parameter to be used when the std_conc argument is defined and omit_std = TRUE. When closed = TRUE the internal standard concentration is removed and the remaining phase concentrations closed to sum to 100 percent. Default = FALSE. |
| tth_align | A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. c(5,65)). If not defined, then the full range is used. |
| align | The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1. |

| | |
|---|---|
| manual_align | A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised. |
| tth_fps | A vector defining the minimum and maximum 2theta values to be used during full pattern summation (e.g. c(5,65)). If not defined, then the full range is used. |
| shift | A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of reference patterns. Default = 0. |
| remove_trace | A single numeric value representing the limit for the concentration of trace phases to be retained, i.e. any mineral with an estimated concentration below remove_trace will be omitted. Default = 0. |
| weighting | an optional 2 column data frame specifying the 2theta values in the first column and a numeric weighting vector in the second column that specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) when minimising the objective function defined in the obj argument. Use this weighting parameter with caution. The default is simply a weighting vector where all values are 1, which hence has no effect on the computed objective function. |
| ... | other arguments |

## Details

Applies full pattern summation (Chipera & Bish, 2002, 2013; Eberl, 2003) to an XRPD sample to quantify phase concentrations. Requires a powdRlib library of reference patterns with reference intensity ratios in order to derive mineral concentrations. Details provided in Butler and Hillier (2021)

## Value

a powdRfps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## References

Butler, B. M., Hillier, S., 2021.powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Comp. Geo. 147, 104662. doi:10.1016/j.cageo.2020.104662

Bish, D.L., Post, J.E., 1989. Modern powder diffraction. Mineralogical Society of America.

Chipera, S.J., Bish, D.L., 2013. Fitting Full X-Ray Diffraction Patterns for Quantitative Analysis: A Method for Readily Quantifying Crystalline and Disordered Phases. Adv. Mater. Phys. Chem. 03, 47-53. doi:10.4236/ampc.2013.31A007

Chipera, S.J., Bish, D.L., 2002. FULLPAT: A full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns. J. Appl. Crystallogr. 35, 744-749. doi:10.1107/S0021889802017405

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

#Since the reference library is relatively small,
#the whole library can be used at once to get an
#estimate of the phases within each sample.
## Not run:
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)

fps_lime <- fps(lib = minerals,
                smpl = soils$limestone,
                refs = minerals$phases$phase_id,
                std = "QUA.1",
                align = 0.2)

fps_granite <- fps(lib = minerals,
                   smpl = soils$granite,
                   refs = minerals$phases$phase_id,
                   std = "QUA.1",
                   align = 0.2)

#Alternatively run all 3 at once using lapply

fps_soils <- lapply(soils, fps,
                    lib = minerals,
                    std = "QUA.2",
                    refs = minerals$phases$phase_id,
                    align = 0.2)
```

```
#Using the rockjock library:

data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
                  refs = c("ORDERED_MICROCLINE",
                           "LABRADORITE",
                           "KAOLINITE_DRY_BRANCH",
                           "MONTMORILLONITE_WYO",
                           "ILLITE_1M_RM30",
                           "CORUNDUM"),
                  std = "CORUNDUM",
                  align = 0.3)

#Alternatively you can specify the internal standard
#concentration if known:
rockjock_1s <- fps(lib = rockjock,
                   smpl = rockjock_mixtures$Mix1,
                   refs = c("ORDERED_MICROCLINE",
                            "LABRADORITE",
                            "KAOLINITE_DRY_BRANCH",
                            "MONTMORILLONITE_WYO",
                            "ILLITE_1M_RM30",
                            "CORUNDUM"),
                   std = "CORUNDUM",
                   std_conc = 20,
                   align = 0.3)


## End(Not run)
```

---

`fps_lm` *Full pattern summation using linear regression*

---

## Description

`fps_lm` returns a simple fit of a given pattern using linear regression, where coefficients may be either positive or negative. Does not return quantitative data. For quantitative results use `fps` or `afps`.

## Usage

```
fps_lm(
  lib,
  smpl,
  harmonise,
  refs,
```

```
  std,
  tth_align,
  align,
  manual_align,
  tth_fps,
  shift,
  p,
  ...
)
```

## Arguments

| | |
|---|---|
| lib | A powdRlib object representing the reference library. Created using the powdRlib constructor function. |
| smpl | A data frame. First column is 2theta, second column is counts |
| harmonise | logical parameter defining whether to harmonise the lib and smpl. Default = TRUE. When TRUE the function harmonises the lib and smpl data to the intersecting 2theta range at the coarsest resolution available using natural splines. |
| refs | A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. If missing from the function call then all phases in the reference library will be used. |
| std | The phase ID (e.g. "QUA.1") to be used as internal standard. Must match an ID provided in the refs parameter. |
| tth_align | A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. c(5,65)). If not defined, then the full range is used. |
| align | The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1. |
| manual_align | A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised. |
| tth_fps | A vector defining the minimum and maximum 2theta values to be used during full pattern summation (e.g. c(5,65)). If not defined, then the full range is used. |
| shift | A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of selected phases. Default = 0. |
| p | a numeric parameter between 0 and 1 specifying the p-value limit for coefficients. Any reference patterns with a p-value greater than this value will be omitted from the linear regression and results recomputed. Must be greater than 0.000001 but no greater than 1. |
| ... | Other arguments |

## Details

Requires a powdRlib library of reference patterns. Mineral concentrations are not quantified and therefore reference intensity ratios are not required.

## Value

a powdRlm object with components:

| | |
|---|---|
| `tth` | a vector of the 2theta scale of the fitted data |
| `fitted` | a vector of the fitted XRPD pattern |
| `measured` | a vector of the original XRPD measurement (aligned) |
| `residuals` | a vector of the residuals (fitted vs measured) |
| `phases` | a dataframe of the phases used to produce the fitted pattern |
| `phases_grouped` | the phases dataframe grouped by phase_name and summed |
| `weighted_pure_patterns` | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| `coefficients` | a named vector of coefficients used to produce the fitted pattern |
| `inputs` | a list of input arguments used in the function call |

## Examples

```
data(rockjock)
data(rockjock_mixtures)

#Compute the PCA and loadings
x1 <- xrpd_pca(rockjock_mixtures,
               mean_center = TRUE,
               bin_size = 1,
               root_transform = 1)

## Not run:
fps_lm_out <- fps_lm(rockjock,
                     smpl = data.frame("x" = x1$loadings$tth,
                                       "y" = x1$loadings$Dim.1),
                     refs = rockjock$phases$phase_id,
                     std = "QUARTZ",
                     align = 0.3,
                     p = 0.01)

plot(fps_lm_out,
     wavelength = "Cu",
     interactive = TRUE,
     group = TRUE)


## End(Not run)
```

`fps_lm.powdRlib` *Full pattern summation using linear regression*

## Description

`fps_lm.powdRlib` returns a simple fit of a given pattern using linear regression, where coefficients may be either positive or negative. Does not return quantitative data. For quantitative results use `fps` or `afps`.

## Usage

```
## S3 method for class 'powdRlib'
fps_lm(
  lib,
  smpl,
  harmonise,
  refs,
  std,
  tth_align,
  align,
  manual_align,
  tth_fps,
  shift,
  p,
  ...
)
```

## Arguments

| | |
|---|---|
| `lib` | A `powdRlib` object representing the reference library. Created using the `powdRlib` constructor function. |
| `smpl` | A data frame. First column is 2theta, second column is counts |
| `harmonise` | logical parameter defining whether to harmonise the `lib` and `smpl`. Default = `TRUE`. When `TRUE` the function harmonises the `lib` and `smpl` data to the intersecting 2theta range at the coarsest resolution available using natural splines. |
| `refs` | A character string of reference pattern IDs or names from the specified library. The IDs or names supplied must be present within the `lib$phases$phase_id` or `lib$phases$phase_name` columns. If missing from the function call then all phases in the reference library will be used. |
| `std` | The phase ID (e.g. "QUA.1") to be used as internal standard. Must match an ID provided in the `refs` parameter. |
| `tth_align` | A vector defining the minimum and maximum 2theta values to be used during alignment (e.g. `c(5,65)`). If not defined, then the full range is used. |
| `align` | The maximum shift that is allowed during initial 2theta alignment (degrees). Default = 0.1. |

| | |
|---|---|
| manual_align | A logical operator denoting whether to optimise the alignment within the negative/position 2theta range defined in the align argument, or to use the specified value of the align argument for alignment of the sample to the standards. Default = FALSE, i.e. alignment is optimised. |
| tth_fps | A vector defining the minimum and maximum 2theta values to be used during full pattern summation (e.g. c(5,65)). If not defined, then the full range is used. |
| shift | A single numeric value denoting the maximum (positive or negative) shift, in degrees 2theta, that is allowed during the shifting of selected phases. Default = 0. |
| p | a numeric parameter between 0 and 1 specifying the p-value limit for coefficients. Any reference patterns with a p-value greater than this value will be omitted from the linear regression and results recomputed. Must be greater than 0.000001 but no greater than 1. |
| ... | Other arguments |

## Details

Requires a powdRlib library of reference patterns. Mineral concentrations are not quantified and therefore reference intensity ratios are not required.

## Value

a powdRlm object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the count intensities of the fitted XRPD pattern |
| measured | a vector of the original count intensities of the XRPD measurement (aligned) |
| residuals | a vector of the residuals (fitted vs measured) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
data(rockjock)
data(rockjock_mixtures)

#Compute the PCA and loadings
x1 <- xrpd_pca(rockjock_mixtures,
               mean_center = TRUE,
               bin_size = 1,
               root_transform = 1)
```

```
## Not run:
fps_lm_out <- fps_lm(rockjock,
                     smpl = data.frame("x" = x1$loadings$tth,
                                       "y" = x1$loadings$Dim.1),
                     refs = rockjock$phases$phase_id,
                     std = "QUARTZ",
                     align = 0.3,
                     p = 0.01)

plot(fps_lm_out,
     wavelength = "Cu",
     interactive = TRUE,
     group = TRUE)


## End(Not run)
```

---

`interpolate` *Interpolate an XY, multiXY or powdRlib object to a given 2theta scale.*

---

## Description

`interpolate` takes an XY, multiXY or powdRlib object and interpolates the data onto a new 2theta scale using a natural spline. See additional help via `?interpolate.XY`, `?interpolate.multiXY` or `?interpolate.powdRlib`.

## Usage

```
interpolate(x, new_tth, ...)
```

## Arguments

| | |
|---|---|
| `x` | an `XY` or `multiXY` object. |
| `new_tth` | a numeric vector of the new 2theta scale. |
| `...` | other arguments |

## Value

an `XY` or `multiXY` object.

## Examples

```
#Define a new 2theta scale:
data(rockjock_mixtures)
tth <- seq(10, 60, 0.04)

#interpolate multiXY object of data onto new scale
```

```
i1 <- interpolate(rockjock_mixtures, new_tth = tth)

#interpolate XY object onto new scale
i2 <- interpolate(rockjock_mixtures$Mix1, new_tth = tth)

#interpolate powdRlib object onto new scale
i3 <- interpolate(minerals, new_tth = tth)
```

---

`interpolate.multiXY` *Interpolate a multiXY object onto a given 2theta scale.*

---

## Description

`interpolate` takes a `multiXY` object, which may contain XY data frames with varying 2theta scales, and interpolates all data frames onto the same scale using cubic splines.

## Usage

```
## S3 method for class 'multiXY'
interpolate(x, new_tth, ...)
```

## Arguments

| | |
|---|---|
| `x` | a `multiXY` object. |
| `new_tth` | a numeric vector of the new 2theta scale. |
| `...` | other arguments |

## Value

a `multiXY` object.

## Examples

```
data(rockjock_mixtures)

#Define a new 2theta scale:
tth <- seq(10, 60, 0.04)

#interpolate data onto new scale
i1 <- interpolate(rockjock_mixtures, new_tth = tth)
```

## interpolate.powdRlib *Interpolate a* powdRlib *object onto a given 2theta scale.*

### Description

interpolate takes a powdRlib object and interpolates the data onto a new 2theta scale using a cubic spline.

### Usage

```
## S3 method for class 'powdRlib'
interpolate(x, new_tth, ...)
```

### Arguments

| | |
|---|---|
| x | a powdRlib object. |
| new_tth | a numeric vector of the new 2theta scale. |
| ... | other arguments |

### Value

a powdRlib object.

### Examples

```
data(minerals)

#Define a new 2theta scale:
tth <- seq(10, 60, 0.04)

#interpolate data onto new scale
i1 <- interpolate(minerals, new_tth = tth)
```

## interpolate.XY *Interpolate an* XY *object onto a given 2theta scale.*

### Description

interpolate takes an XY object and interpolates the data onto a new 2theta scale using a cubic spline.

### Usage

```
## S3 method for class 'XY'
interpolate(x, new_tth, ...)
```

### Arguments

| | |
|---|---|
| x | an XY object. |
| new_tth | a numeric vector of the new 2theta scale. |
| ... | other arguments |

### Value

an XY object.

### Examples

```
data(rockjock_mixtures)

#Define a new 2theta scale:
tth <- seq(10, 60, 0.04)

#interpolate data onto new scale
i1 <- interpolate(rockjock_mixtures$Mix1, new_tth = tth)
```

---

## merge.powdRlib *Merge two powdRlib objects*

---

### Description

merge.powdRlib allows two powdRlib objects (which must have) the same 2theta scale) to be merged into a single powdRlib object.

### Usage

```
## S3 method for class 'powdRlib'
merge(x, y, ...)
```

### Arguments

| | |
|---|---|
| x | a powdRlib object. |
| y | a powdRlib object |
| ... | other arguments |

### Value

a powdRlib object.

## Examples

```
#Load the minerals library
data(minerals)

#Load the rockjock library
data(rockjock)

#interpolate minerals library onto same 2theta as rockjock
minerals_i <- interpolate(minerals, new_tth = rockjock$tth)

#merge the libraries
merged_lib <- merge(rockjock, minerals_i)
```

---

# minerals *An example powdRlib reference library*

---

## Description

This powdRlib object, built using the powdRlib constructor function, contains a range of measured XRPD data (collected using Cu K-alpha radiation) along with their reference intensity ratios. The library is designed for simple examples only and can be used with the soils data for relatively fast tests of fps and afps.

## Usage

```
minerals
```

## Format

A powdRlib object of 3 components

**xrd** A dataframe of all the count intensities of all reference patterns. Column names denote the unique phase ID of each reference pattern

**tth** A vector of the 2theta scale for all reference patterns in the library

**phases** A dataframe the phase IDs, names and reference intensity ratios (RIR)

---

# minerals_phases *Example phases table for a reference library*

---

## Description

A data frame of associated phase information for the minerals_xrd data. Together these two data frames can be combined with the powdRlib constructor function to create an example reference library (see ?powdRlib). Use the same layout to create custom reference libraries.

### Usage

```
minerals_phases
```

### Format

A 3 column data frame consisting of:

**phase_id** A string defining the unique phase IDs that should match those defined as column names of the minerals table (e.g. `minerals_xrd`).

**phase_name** A string defining the mineral group that each reference pattern belongs to.

**rir** A vector defining the reference intensity ratios of each reference pattern.

---

## `minerals_regroup` *Example regrouping structure for the* `minerals` *data*

---

### Description

Example regrouping structure for the `minerals` data

### Usage

```
minerals_regroup
```

### Format

A 2 column data frame.

First column contains the unique phase IDs of all phases in the `minerals` data. Second column contains the grouping structure for the data (Non-clay, Clay or Amorphous).

---

## `minerals_xrd` *Example xrd table for a reference library*

---

### Description

A table of 14 reference patterns and their corresponding two theta scale that can be combined with the `minerals_phases` table to create a `powdRlib` object using the `powdRlib` constructor function. Use the same layout to create custom reference libraries.

### Usage

```
minerals_xrd
```

### Format

A dataframe

The first column defines the two theta scale, and remaining columns are individual reference patterns of pure minerals or amorphous phases. Each column title should be a unique mineral ID

multi_xy_to_df *Convert a multiXY object to a data frame.*

## Description

multi_xy_to_df converts multiXY objects to a column-wise data frame.

## Usage

```
multi_xy_to_df(x, tth, ...)
```

## Arguments

| | |
|---|---|
| x | a multiXY object. |
| tth | a logical value denoting whether the 2theta scale is appended as the first column. Default = TRUE. |
| ... | other arguments |

## Value

A data.frame.

## Examples

```
#Load the minerals library
data(soils)

soils_df1 <- multi_xy_to_df(soils, tth = TRUE)
soils_df2 <- multi_xy_to_df(soils, tth = FALSE)
```

multi_xy_to_df.multiXY *Convert a multiXY object to a data frame.*

## Description

multi_xy_to_df.multiXY converts multiXY objects to a column-wise data frame.

## Usage

```
## S3 method for class 'multiXY'
multi_xy_to_df(x, tth, ...)
```

### Arguments

| | |
|---|---|
| x | a multiXY object. |
| tth | a logical value denoting whether the 2theta scale is appended as the first column. Default = TRUE. |
| ... | other arguments |

### Value

A data.frame.

### Examples

```
#Load the minerals library
data(soils)

soils_df1 <- multi_xy_to_df(soils, tth = TRUE)
soils_df2 <- multi_xy_to_df(soils, tth = FALSE)
```

---

## omit_std — *Omit the internal standard from phase concentration data within a* powdRfps *or* powdRafps *object*

---

### Description

omit_std adjusts phase concentrations in a powdRfps or powdRafps object (derived from fps() and afps(), respectively) by removing the concentrations of the internal standard. Relevant information for the calculation is automatically extracted from x$inputs$std and x$inputs$std_conc. For more information see ?omit_std.powdRfps and omit_std.powdRafps.

### Usage

```
omit_std(x, ...)
```

### Arguments

| | |
|---|---|
| x | A powdRfps or powdRafps object.. |
| ... | other arguments |

### Value

a powdRfps or powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (measured minus fitted) |

| | |
|---|---|
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
                  refs = c("ORDERED_MICROCLINE",
                           "LABRADORITE",
                           "KAOLINITE_DRY_BRANCH",
                           "MONTMORILLONITE_WYO",
                           "ILLITE_1M_RM30",
                           "CORUNDUM"),
                 std = "CORUNDUM",
                 align = 0.3,
                 std_conc = 20)

rockjock_1o <- omit_std(rockjock_1)

rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1,
                    std_conc = 20)

rockjock_a1o <- omit_std(rockjock_a1)


## End(Not run)
```

---

`omit_std.powdRafps` *Omit the internal standard from phase concentration data within a* `powdRafps` *object*

---

## Description

omit_std.powdRafps adjusts phase concentrations in a powdRafps object by removing the concentrations of the internal standard. Relevant information for the calculation is automatically extracted from x$inputs$std and x$inputs$std_conc.

## Usage

```
## S3 method for class 'powdRafps'
omit_std(x, ...)
```

## Arguments

| | |
|---|---|
| x | A powdRafps object derived from afps(). |
| ... | other arguments |

## Value

a powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the fitted XRPD pattern |
| measured | a vector of the original XRPD measurement (aligned and harmonised) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| phases_grouped | the phases dataframe grouped by phase_name and concentrations summed |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_a1 <- afps(lib = rockjock,
                    smpl = rockjock_mixtures$Mix1,
                    std = "CORUNDUM",
                    align = 0.3,
                    lod = 1,
                    std_conc = 20)

rockjock_a1o <- omit_std(rockjock_a1)
```

```
## End(Not run)
```

---

`omit_std.powdRfps` *Omit the internal standard from phase concentration data within a* `powdRfps` *object*

---

## Description

`omit_std.powdRfps` adjusts phase concentrations in a `powdRfps` object by removing the concentrations of the internal standard. Relevant information for the calculation is automatically extracted from `x$inputs$std` and `x$inputs$std_conc`.

## Usage

```
## S3 method for class 'powdRfps'
omit_std(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `powdRfps` object derived from `fps()`. |
| `...` | other arguments |

## Value

a `powdRfps` object with components:

| | |
|---|---|
| `tth` | a vector of the 2theta scale of the fitted data |
| `fitted` | a vector of the fitted XRPD pattern |
| `measured` | a vector of the original XRPD measurement (aligned and harmonised) |
| `residuals` | a vector of the residuals (measured minus fitted) |
| `phases` | a dataframe of the phases used to produce the fitted pattern and their concentrations |
| `phases_grouped` | the phases dataframe grouped by phase_name and concentrations summed |
| `obj` | named vector of the objective parameters summarising the quality of the fit |
| `weighted_pure_patterns` | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| `coefficients` | a named vector of coefficients used to produce the fitted pattern |
| `inputs` | a list of input arguments used in the function call |

## Examples

```
## Not run:
data(rockjock)
data(rockjock_mixtures)

rockjock_1 <- fps(lib = rockjock,
                  smpl = rockjock_mixtures$Mix1,
                  refs = c("ORDERED_MICROCLINE",
                           "LABRADORITE",
                           "KAOLINITE_DRY_BRANCH",
                           "MONTMORILLONITE_WYO",
                           "ILLITE_1M_RM30",
                           "CORUNDUM"),
                 std = "CORUNDUM",
                 align = 0.3,
                 std_conc = 20)

rockjock_1o <- omit_std(rockjock_1)


## End(Not run)
```

---

# plot.multiXY *Plotting a multiXY object*

---

## Description

plot.multiXY is designed to provide easy, adaptable plots of multiple XRPD patterns.

## Usage

```
## S3 method for class 'multiXY'
plot(x, wavelength, xlim, normalise, interactive, ...)
```

## Arguments

| | |
|---|---|
| x | a multiXY object |
| wavelength | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| xlim | A numeric vector providing limits of the x-axis (E.g. c(10, 60)). Defaults to full x-axis unless specified. |
| normalise | Logical. If TRUE then count intensities will be normalised to a minimum of zero and maximum of 1. Default = FALSE. |
| interactive | Logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| ... | other arguments |

## Details

Plots can be made interactive using the logical `interactive` argument.

## Examples

```
# Load the minerals library
data(rockjock_mixtures)
## Not run:
plot(as_multi_xy(rockjock_mixtures), wavelength = "Cu")
plot(as_multi_xy(rockjock_mixtures), wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

---

`plot.powdRafps` *Plotting elements of a powdRafps object*

---

## Description

`plot.powdRafps` is designed to provide easy, adaptable plots of full pattern summation outputs produced from `afps`.

## Usage

```
## S3 method for class 'powdRafps'
plot(x, wavelength, mode, group, xlim, show_excluded, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | a powdRafps object |
| `wavelength` | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| `mode` | One of "fit", "residuals" or "both" defining whether to plot the fitted patterns, the residuals of the fit, or both, respectively. Default = "fit". |
| `group` | A logical parameter used to specify whether the plotted data are grouped according to the phase name. Default = FALSE. |
| `xlim` | A numeric vector providing limits of the x-axis (E.g. `c(10, 60)`). Defaults to full x-axis unless specified. |
| `show_excluded` | A logical value specifying whether the areas excluded from the fitting are identified in the plot as grey rectangles. Default = `TRUE`. |
| `interactive` | logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| `...` | other arguments |

## Details

When seeking to inspect the results from full pattern summation, interactive plots are particularly useful and can be specified with the `interactive` argument.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

## Not run:
afps_sand <-  afps(lib = minerals,
                   smpl = soils$sandstone,
                   std = "QUA.1",
                   amorphous = "ORG",
                   align = 0.2,
                   lod = 0.1)

plot(afps_sand, wavelength = "Cu")
plot(afps_sand, wavelength = "Cu", interactive = TRUE)


## End(Not run)
```

---

`plot.powdRbkg` *Plotting a powdRbkg object*

---

## Description

`plot.powdRbkg` is designed to provide quick plots to inspect the fitted backgrounds obtained from `bkg`.

## Usage

```
## S3 method for class 'powdRbkg'
plot(x, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | a powdRbkg object |
| `interactive` | Logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| `...` | other arguments |

## Details

The only mandatory argument is x, which must be a powdRbkg object. Plots can be made interactive using the logical `interactive` argument.

## Examples

```
# Load the minerals library
data(minerals)

## Not run:
plot(minerals, interactive = TRUE)

## End(Not run)
```

---

`plot.powdRfps` *Plotting elements of a powdRfps object*

---

## Description

`plot.powdRfps` is designed to provide easy, adaptable plots of full pattern summation outputs produced from `fps`.

## Usage

```
## S3 method for class 'powdRfps'
plot(x, wavelength, mode, group, xlim, show_excluded, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | a powdRfps object |
| `wavelength` | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| `mode` | One of "fit", "residuals" or "both" defining whether to plot the fitted patterns, the residuals of the fit, or both, respectively. Default = "fit". |
| `group` | A logical parameter used to specify whether the plotted data are grouped according to the phase name. Default = FALSE. |
| `xlim` | A numeric vector providing limits of the x-axis (E.g. `c(10, 60)`). Defaults to full x-axis unless specified. |
| `show_excluded` | A logical value specifying whether the areas excluded from the fitting are identified in the plot as grey rectangles. Default = `TRUE`. |
| `interactive` | logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| `...` | other arguments |

## Details

When seeking to inspect the results from full pattern summation, interactive plots are particularly useful and can be specified with the `interactive` argument.

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

## Not run:
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)

plot(fps_sand, wavelength = "Cu")
plot(fps_sand, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

---

# plot.powdRlib *Plotting elements of a powdRlib object*

---

## Description

`plot.powdRlib` is designed to provide easy, adaptable plots of an XRPD reference library built using the `powdRlib` constructor function.

## Usage

```
## S3 method for class 'powdRlib'
plot(x, wavelength, refs, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | a powdRlib object |
| `wavelength` | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| `refs` | a character string of reference pattern id's to be plotted |
| `interactive` | Logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| `...` | other arguments |

## Details

Plots can be made interactive using the logical `interactive` argument.

## Examples

```
# Load the minerals library
data(minerals)
## Not run:
plot(minerals, wavelength = "Cu", refs = "ALB")
plot(minerals, wavelength = "Cu", refs = "ALB", interactive = TRUE)

## End(Not run)
```

---

`plot.powdRlm` *Plotting elements of a powdRlm object*

---

## Description

`plot.powdRlm` is designed to provide easy, adaptable plots of full pattern summation outputs produced from `fps_lm`.

## Usage

```
## S3 method for class 'powdRlm'
plot(x, wavelength, mode, xlim, group, show_excluded, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | a powdRlm object |
| `wavelength` | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| `mode` | One of "fit", "residuals" or "both" defining whether to plot the fitted patterns, the residuals of the fit, or both, respectively. Default = "fit". |
| `xlim` | A numeric vector providing limits of the x-axis (E.g. `c(10, 60)`). Defaults to full x-axis unless specified. |
| `group` | A logical parameter used to specify whether the plotted data are grouped according to the phase name. Default = FALSE. |
| `show_excluded` | A logical value specifying whether the areas excluded from the fitting are identified in the plot as grey rectangles. Default = `TRUE`. |
| `interactive` | logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| `...` | other arguments |

## Details

When seeking to inspect the results from full pattern summation, interactive plots are particularly useful and can be specified with the `interactive` argument.

## Examples

```
data(rockjock)
data(rockjock_mixtures)

#Compute the PCA and loadings
x1 <- xrpd_pca(rockjock_mixtures,
               mean_center = TRUE,
               bin_size = 1,
               root_transform = 1)

## Not run:
fps_lm_out <- fps_lm(rockjock,
                     smpl = data.frame("x" = x1$loadings$tth,
                                       "y" = x1$loadings$Dim.1),
                     refs = rockjock$phases$phase_id,
                     std = "QUARTZ",
                     align = 0.3,
                     p = 0.01)

plot(fps_lm_out,
     wavelength = "Cu",
     interactive = TRUE,
     group = TRUE)


## End(Not run)
```

---

`plot.XY` *Plotting an XY object*

---

## Description

`plot.XY` is designed to provide easy, adaptable plots of an XRPD pattern.

## Usage

```
## S3 method for class 'XY'
plot(x, wavelength, xlim, normalise, interactive, ...)
```

## Arguments

| | |
|---|---|
| `x` | an XY object |

| | |
|---|---|
| wavelength | One of "Cu", "Co" or a custom numeric value defining the wavelength (in Angstroms). Used to compute d-spacings.When "Cu" or "Co" are supplied, wavelengths of 1.54056 or 1.78897 are used, respectively. |
| xlim | A numeric vector providing limits of the x-axis (E.g. `c(10, 60)`). Defaults to full x-axis unless specified. |
| normalise | Logical. If TRUE then count intensities will be normalised to a minimum of zero and maximum of 1. Default = `FALSE`. |
| interactive | Logical. If TRUE then the output will be an interactive ggplotly object. If FALSE then the output will be a ggplot object. |
| ... | other arguments |

## Details

Plots can be made interactive using the logical `interactive` argument.

## Examples

```
# Load the minerals library
data(rockjock_mixtures)
## Not run:
plot(rockjock_mixtures$Mix1, wavelength = "Cu")
plot(rockjock_mixtures$Mix1, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

---

powdR *powdR: Full Pattern Summation of X-Ray Powder Diffraction Data*

---

## Description

An implementation of the full pattern summation approach to quantitative mineralogy from X-ray powder diffraction data (Chipera & Bish, 2002, 2013; Eberl, 2003; Butler & Hillier 2021).

## Author(s)

Benjamin Butler, The James Hutton Institute, Aberdeen, UK

## References

Butler, B. M., Hillier, S., 2021.powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Comp. Geo. 147, 104662. doi:10.1016/j.cageo.2020.104662

Chipera, S.J., Bish, D.L., 2013. Fitting Full X-Ray Diffraction Patterns for Quantitative Analysis: A Method for Readily Quantifying Crystalline and Disordered Phases. Adv. Mater. Phys. Chem. 03, 47-53. doi:10.4236/ampc.2013.31A007

Chipera, S.J., Bish, D.L., 2002. FULLPAT: A full-pattern quantitative analysis program for X-ray powder diffraction using measured and calculated patterns. J. Appl. Crystallogr. 35, 744-749. doi:10.1107/S0021889802017405

Eberl, D.D., 2003. User's guide to ROCKJOCK - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

---

`powdRlib` *Create an XRPD reference library*

---

## Description

A constructor function for creating a `powdRlib` object from two tables of data. The resulting `powdRlib` object is required when using `fps` or `afps`.

## Usage

```
powdRlib(xrd_table, phases_table, check_names)
```

## Arguments

| | |
|---|---|
| `xrd_table` | A data frame of the count intensities of the XRPD reference patterns, all scaled to same maximum intensity, with their 2theta axis as the first column. |
| `phases_table` | A data frame of the required data (phase ID, phase name, and reference intensity ratio) for each reference pattern. |
| `check_names` | A logical argument defining whether the column names in the data supplied in `xrd_table` are syntactically valid variable names and are not duplicated. Default = `TRUE`. |

## Value

a powdRlib object with components:

| | |
|---|---|
| `xrd` | a data frame of the count intensities of the reference patterns |
| `tth` | a vector of the 2theta axis |
| `phases` | a 3 column data frame of the IDs, names and reference intensity ratios of the reference pattern |

## Examples

```
#load an example xrd_table
data(minerals_xrd)
#load an example phases_table
data(minerals_phases)

#Create a reference library object
xrd_lib <- powdRlib(xrd_table = minerals_xrd,
                    phases_table = minerals_phases)
```

r *Calculate the R value for a fitted pattern*

## Description

r computes the difference between a measured and fitted pattern. See equation for R in section 2.1 of Butler and Hillier (2021).

## Usage

```
r(measured, fitted, weighting)
```

## Arguments

| | |
|---|---|
| measured | a vector of count intensities for a measured pattern |
| fitted | a vector of count intensities for a fitted pattern |
| weighting | an optional weighting vector of the same length as those specified in measured and fitted, which specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) from the calculation. Use with caution. Default is simply a weighting vector where all values are 1, which hence has no effect on the computed value. |

## Value

a single numeric value

## References

Butler, B.M., Hillier, S., 2021. powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Computers and Geosciences. 147, 104662. doi:10.1016/j.cageo.2020.104662

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:
#Produce a fit
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)
```

```
r(measured = fps_sand$measured,
  fitted = fps_sand$fitted)


## End(Not run)
```

---

read_xy *Read ASCII XY data*

---

## Description

read_xy is a wrapper for read.csv that is designed for space separated XRPD data.

## Usage

```
read_xy(files, header, sep)
```

## Arguments

| | |
|---|---|
| files | path of the file(s) to be imported. |
| header | a logical value indicating whether the file contains the names of the variables as its first line. Default = FALSE. |
| sep | the field separator character. Values on each line of the file are separated by this character. Default = " ", indicating space separated format. |

## Value

If only one path is supplied then an XY data frame with 2 columns is returned, the first being the 2theta axis and the second being the count intensities. If more than one path is supplied then a multiXY list is returned, with each item in the list being an XY data frame as already described.

## Examples

```
#load example XY file
file <- system.file("extdata/D5000/xy/D5000_1.xy", package = "powdR")
xy <- read_xy(file)

#Load multiple XY files
files <- dir(system.file("extdata/D5000/xy", package = "powdR"),
             full.names = TRUE)
xy_list <- read_xy(files)

## Not run:
plot(xy_list, wavelength = "Cu")
plot(xy_list, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

regroup *regroup*

## Description

regroup allows an alternative grouping structure to be applied to powdRfps and powdRafps objects. For more details see ?regroup.powdRfps or ?regroup.powdRafps.

## Usage

```
regroup(x, ...)
```

## Arguments

| | |
|---|---|
| x | A powdRfps or powdRafps object |
| ... | Other parameters passed to methods e.g. fps.powdRlib |

## Details

powdRfps and powdRafps objects contain a data frame called phases_grouped that summarises phase concentrations based on defined mineral groups from the powdRlib reference library. regroup allows you to change this grouping structure by supplying new group identities.

## Value

a powdRfps or powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the count intensities of fitted XRPD pattern |
| measured | a vector of the count intensities of original XRPD measurement (aligned) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern |
| phases_grouped | the phases dataframe grouped and summed by phase_name |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
#Load the minerals library
data(minerals)

#Load the soils data
data(soils)

#Load the regrouping structure
data(minerals_regroup)

## Not run:
fps_sandstone <- fps(lib = minerals,
                     smpl = soils$sandstone,
                     refs = minerals$phases$phase_id,
                     std = "QUA.1",
                     align = 0.2)

fps_sandstone_regrouped <- regroup(fps_sandstone,
                                   minerals_regroup)

fps_sandstone_regrouped$phases_grouped


## End(Not run)
```

---

`regroup.powdRafps` *regroup*

---

## Description

`regroup.powdRafps` allows an alternative grouping structure to be applied to `powdRafps` objects.

## Usage

```
## S3 method for class 'powdRafps'
regroup(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `powdRafps` object |
| `y` | A data frame. First column contains the phase IDs covering all those present in `x$phases$phase_id`. Second column contains the desired grouping of each phase. |
| `...` | other arguments |

## Details

`powdRafps` objects contain a data frame called `phases_grouped` that summarises phase concentrations based on defined mineral groups from the `powdRlib` reference library. `regroup` allows you to change this grouping structure by supplying new group identities.

## Value

a powdRafps object with components:

| | |
|---|---|
| tth | a vector of the 2theta scale of the fitted data |
| fitted | a vector of the count intensities of fitted XRPD pattern |
| measured | a vector of the count intensities of original XRPD measurement (aligned) |
| residuals | a vector of the residuals (measured minus fitted) |
| phases | a dataframe of the phases used to produce the fitted pattern |
| phases_grouped | the phases dataframe grouped and summed by phase_name |
| obj | named vector of the objective parameters summarising the quality of the fit |
| weighted_pure_patterns | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| coefficients | a named vector of coefficients used to produce the fitted pattern |
| inputs | a list of input arguments used in the function call |

## Examples

```
#Load the minerals library
data(minerals)

# Load the soils data
data(soils)

#Load the regrouping structure
data(minerals_regroup)

## Not run:
afps_sandstone <- afps(lib = minerals,
                       smpl = soils$sandstone,
                       std = "QUA.2",
                       align = 0.2,
                       lod = 0.2,
                       amorphous = "ORG",
                       amorphous_lod = 1)

afps_sandstone_regrouped <- regroup(afps_sandstone,
                                    minerals_regroup)

afps_sandstone_regrouped$phases_grouped

## End(Not run)
```

`regroup.powdRfps` *regroup*

## Description

`regroup.powdRfps` allows an alternative grouping structure to be applied to `powdRfps` objects.

## Usage

```
## S3 method for class 'powdRfps'
regroup(x, y, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `powdRfps` object |
| `y` | A data frame. First column contains the phase IDs covering all those present in `x$phases$phase_id`. Second column contains the desired grouping of each phase. |
| `...` | other arguments |

## Details

`powdRfps` objects contain a data frame called `phases_grouped` that summarises phase concentrations based on defined mineral groups from the `powdRlib` reference library. `regroup` allows you to change this grouping structure by supplying new group identities.

## Value

a `powdRfps` object with components:

| | |
|---|---|
| `tth` | a vector of the 2theta scale of the fitted data |
| `fitted` | a vector of the count intensities of fitted XRPD pattern |
| `measured` | a vector of the count intensities of original XRPD measurement (aligned) |
| `residuals` | a vector of the residuals (measured minus fitted) |
| `phases` | a dataframe of the phases used to produce the fitted pattern |
| `phases_grouped` | the phases dataframe grouped and summed by phase_name |
| `obj` | named vector of the objective parameters summarising the quality of the fit |
| `weighted_pure_patterns` | a dataframe of reference patterns used to produce the fitted pattern. All patterns have been weighted according to the coefficients used in the fit |
| `coefficients` | a named vector of coefficients used to produce the fitted pattern |
| `inputs` | a list of input arguments used in the function call |

## Examples

```
#Load the minerals library
data(minerals)

#Load the soils data
data(soils)

#Load the regrouping structure
data(minerals_regroup)

## Not run:
fps_sandstone <- fps(lib = minerals,
                     smpl = soils$sandstone,
                     refs = minerals$phases$phase_id,
                     std = "QUA.1",
                     align = 0.2)

fps_sandstone_regrouped <- regroup(fps_sandstone,
                                   minerals_regroup)

fps_sandstone_regrouped$phases_grouped


## End(Not run)
```

---

`rockjock` *RockJock reference library*

---

## Description

A `powdRlib` object of 168 pure reference patterns from the RockJock library (Cu K-alpha radiation) along with reference intensity ratios. Note that compared to same library supplied with RockJock the powdR patterns have been normalised to 10,000 counts and reference intensity ratios transformed so that all are relative to that of corundum, which has been set to a value of 1.0. Can be used with the `fps()` and `afps()` functions for quantitative analysis. Example mixtures for testing the `rockjock` library with known concentrations are available in the `rockjock_mixtures` data. See `?rockjock_mixtures`.

## Usage

```
rockjock
```

## Format

A powdRlib object of 3 components

**xrd** A dataframe of all the count intensities of all reference patterns. Column names denote the unique phase ID of each reference pattern

**tth** A vector of the 2theta scale for all reference patterns in the library

**phases** A dataframe the phase IDs, names and reference intensity ratios (RIR)

### Author(s)

Dennis Eberl

### References

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

---

## rockjock_mixtures *RockJock synthetic mixtures*

---

### Description

A multiXY list containing 8 XRPD measurements (Cu K-alpha radiation) of synthetic mixtures that can be used to assess accuracy of quantitative analysis from the `fps()` and `afps()` functions. The mixtures contain various amounts of quartz (QUARTZ standard of the `rockjock` library), K-feldspar (ORDERED_MICROCLINE), plagioclase (LABRADORITE), kaolinite (KAOLINITE_DRY_BRANCH), dioctahedral smectite (MONTMORILLIONITE_WYO), illite (ILLITE_1M_RM30) and corundum (CORUNDUM).

### Usage

```
rockjock_mixtures
```

### Format

A multiXY list of 8 components, each comprised of two columns. Column `tth` specifies the 2theta axis and `counts` specifies the count intensities. The mixtures have to following compositions that are also tabulated in the `rockjock_weights` data.

**Mix1** Contains: 4 % K-feldspar, 20 % plagioclase, 12 % kaolinite, 36 % dioctahedral smectite, 8 % illite and 20 % corundum.

**Mix2** Contains: 4 % quartz, 8 % K-feldspar, 36 % plagioclase, 20 % kaolinite, 12 % illite and 20 % corundum.

**Mix3** Contains: 8 % quartz, 12 % K-feldspar, 36 % kaolinite, 4 % dioctahedral smectite, 20 % illite and 20 % corundum.

**Mix4** Contains: 12 % quartz, 20 % K-feldspar, 4 % plagioclase, 8 % dioctahedral smectite, 36 % illite and 20 % corundum.

**Mix5** Contains: 20 % quartz, 36 % K-feldspar, 8 % plagioclase, 4 % kaolinite, 12 % dioctahedral smectite and 20 % corundum.

**Mix6** Contains: 36 % quartz, 12 % plagioclase, 8 % kaolinite, 20 % dioctahedral smectite, 4 % illite and 20 % corundum.

**Mix7** Contains: 8 % K-feldspar, 40 % plagioclase, 4 % kaolinite, 12 % dioctahedral smectite, 16 % illite and 20 % corundum.

**Mix8** Contains: 8 % quartz, 4 % K-feldspar, 4 % plagioclase, 24 % dioctahedral smectite, 40 % illite and 20 % corundum.

### Author(s)

Dennis Eberl

### References

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

---

## rockjock_regroup *Regrouping structure for the rockjock reference library*

---

### Description

A data frame containing an example re-grouping structure for the `rockjock` reference library, which results in a slightly coarser description of clay minerals and Fe/Ti-(hydr)oxides in `powdRfps` or `powdRafps` objects when used with `regroup()`.

### Usage

```
rockjock_regroup
```

### Format

A data frame with three columns:

**phase_id** the phase IDs present in `afsis$phases$phase_id`.

**phase_name_grouped** The phase names that constitute the first regrouping structure.

**phase_name_grouped2** The phase names that constitute the second regrouping structure

---

## rockjock_weights *Mineral concentrations of the* `rockjock_mixtures` *data*

---

### Description

A dataframe summarising the weighed mineral concentrations of the `rockjock_mixtures` data, all in units of weight percent.

### Usage

```
rockjock_weights
```

### Format

An 8 column dataframe, with each row detailing the composition of a sample.

## Author(s)

Dennis Eberl

## References

Eberl, D.D., 2003. User's guide to RockJock - A program for determining quantitative mineralogy from powder X-ray diffraction data. Boulder, CA.

---

`run_bkg` *Run the background fitting shiny app*

---

## Description

A wrapper for `shiny::runApp` to start the `powdR` background fitting Shiny app.

## Usage

```
run_bkg(...)
```

## Arguments

| | |
|---|---|
| `...` | further arguments to pass to `shiny::runApp` |

## Examples

```
## Not run:

run_powdR()


## End(Not run)
```

---

`run_powdR` *Run the powdR shiny app*

---

## Description

A wrapper for `shiny::runApp` to start the Shiny app for `powdR`.

## Usage

```
run_powdR(...)
```

### Arguments

| | |
|---|---|
| ... | further arguments to pass to shiny::runApp |

### Examples

```
## Not run:

run_powdR()


## End(Not run)
```

---

## rwp *Calculate the Rwp value for a fitted pattern*

---

### Description

rwp computes the difference between a measured and fitted pattern. See equation for Rwp in section 2.1 of Butler and Hillier (2021).

### Usage

```
rwp(measured, fitted, weighting)
```

### Arguments

| | |
|---|---|
| measured | a vector of count intensities for a measured pattern |
| fitted | a vector of count intensities for a fitted pattern |
| weighting | an optional weighting vector of the same length as those specified in measured and fitted, which specifies areas of the pattern to either emphasise (values > 1) or omit (values = 0) from the calculation. Use with caution. Default is simply a weighting vector where all values are 1, which hence has no effect on the computed value. |

### Value

a single numeric value

### References

Butler, B.M., Hillier, S., 2021. powdR: An R package for quantitative mineralogy using full pattern summation of X-ray powder diffraction data. Computers and Geosciences. 147, 104662. doi:10.1016/j.cageo.2020.104662

## Examples

```
# Load soils xrd data
data(soils)

#Load minerals library
data(minerals)

## Not run:
#Produce a fit
fps_sand <-  fps(lib = minerals,
                 smpl = soils$sandstone,
                 refs = minerals$phases$phase_id,
                 std = "QUA.1",
                 align = 0.2)

rwp(measured = fps_sand$measured,
    fitted = fps_sand$fitted)


## End(Not run)
```

---

`soils` *Example soil XRPD data*

---

## Description

3 soil samples from different parent materials measured by XRPD (Cu K-alpha radiation)

## Usage

```
soils
```

## Format

A multiXY list of 3 XY dataframes (named according to parent material type), with each XY dataframe containing two columns of:

**tth** The 2theta measurement intervals

**counts** The count intensities

subset.powdRlib *Subset a powdRlib object*

## Description

subset.powdRlib is designed to provide an easy way of subsetting a powdRlib object by defining the phase ID's that the user wishes to either keep or remove.

## Usage

```
## S3 method for class 'powdRlib'
subset(x, refs, mode, ...)
```

## Arguments

| | |
|---|---|
| x | a powdRlib object. |
| refs | a string of the phase IDs or names of reference patterns to be subset. The ID's or names supplied must be present within the lib$phases$phase_id or lib$phases$phase_name columns. |
| mode | denotes whether the phase IDs or names defined in the refs argument are retained ("keep") or removed ("remove"). |
| ... | other arguments |

## Value

A powdRlib object.

## Examples

```
#Load the minerals library
data(minerals)

minerals_keep <- subset(minerals,
                        refs = c("QUA.1", "QUA.2"),
                        mode = "keep")

minerals_remove <- subset(minerals,
                          refs = c("QUA.1", "QUA.2"),
                          mode = "remove")
```

summarise_mineralogy *Summarise the mineralogy from multiple powdRfps and powdRafps outputs*

## Description

summarise_mineralogy creates a summary table of quantified mineral concentrations across a given dataset using a list of multiple powdRfps or powdRafps derived from fps() and afps(), respectively.

## Usage

```
summarise_mineralogy(x, type, order, rwp, r, delta)
```

## Arguments

| | |
|---|---|
| x | a list of powdRfps or powdRafps objects. |
| type | a string specifying whether the table uses all phase ID's, or summarises them according to the phase name. One of "all" or "grouped". |
| order | a logical operator denoting whether the columns of the resulting summary table are ordered in descending order according to the summed abundance of each phase across the dataset. |
| rwp | a logical operator denoting whether to include the Rwp value as the final column in the output. This provides an objective measure of the difference between the fitted and measured patterns. |
| r | a logical operator denoting whether to include the R value as the final column in the output. This provides an objective measure of the difference between the fitted and measured patterns. |
| delta | a logical operator denoting whether to include the Delta value as the final column in the output. This provides an objective measure of the difference between the fitted and measured patterns. |

## Value

A data frame

## Examples

```
data(minerals)
data(soils)

## Not run:
multiple_afps <- lapply(soils, afps,
                        lib = minerals,
                        std = "QUA.1",
                        align = 0.2,
```

```
                        lod = 0.1,
                        amorphous = "ORG",
                        amorphous_lod = 1)

sm1 <- summarise_mineralogy(multiple_afps,
                            type = "all",
                            order = TRUE)

sm2 <- summarise_mineralogy(multiple_afps,
                            type = "grouped",
                            order = TRUE)

sm3 <- summarise_mineralogy(multiple_afps,
                            type = "grouped",
                            order = TRUE,
                            rwp = TRUE)

## End(Not run)
```

---

`tth_transform` *Transform a two theta axis between wavelengths*

---

## Description

`tth_transform` converts the two theta axis from one wavelength to another via Bragg's law. Use this function with caution if intending the apply `fps()` or `afps()` to wavelength transformed samples or libraries because background signals can vary with wavelength which may therefore affect the quality of the fit.

## Usage

```
tth_transform(tth, from, to)
```

## Arguments

| | |
|---|---|
| `tth` | the 2theta vector to be transformed |
| `from` | numeric value defining the wavelength (Angstroms) to transform from |
| `to` | numeric value defining the wavelength (Angstroms) to transform to |

## Value

a transformed 2theta vector

## Examples

```
data(soils)
sandstone2 <- soils$sandstone

#Convert from Cu (1.54056 Angstroms) to Co (1.78897 Angstroms)
sandstone2$tth <- tth_transform(sandstone2$tth,
                                from = 1.54056,
                                to = 1.78897)

sandstone_list <- as_multi_xy(list("sandstone" = soils$sandstone,
                                   "sandstone2" = sandstone2))
#plot the change
plot(sandstone_list, wavelength = "Cu")

#Alternatively convert the 2theta axis of a library
data(minerals)

minerals2 <- minerals
minerals2$tth <- tth_transform(minerals2$tth,
                                from = 1.54056,
                                to = 1.78897)

#Plot the difference
plot(x = minerals$tth, y = minerals$xrd$QUA.1,
     type = "l", xlim = c(0, 85))
lines(x = minerals2$tth, y = minerals2$xrd$QUA.1,
      col = "red")
```

---

# xrpd_pca *PCA of XRPD data*

---

## Description

xrpd_pca is used to apply principal component analysis to X-ray powder diffraction data.

## Usage

```
xrpd_pca(x, mean_center, bin_size, root_transform, components)
```

## Arguments

| | |
|---|---|
| x | A multiXY list containing the XRPD data, where each item in the list is a 2 column XY dataframe defining the x (2theta) and y (counts) axes of each measurement. Each item in the list must have a name corresponding to a unique sample ID. |
| mean_center | A logical argument defining whether mean centering is applied to the XRPD data (default = TRUE). |
| bin_size | An integer between 1 and 10 defining whether to bin the XRPD data to a lower resolution. This bin_size defines the number of data points used in each bin. |

| | |
|---|---|
| root_transform | An integer between 1 and 8 defining the root transform to apply to the XRPD data |
| components | An integer defining the number of principal components to include in the output. Must be at least 1 less than the number of XRPD patterns in the dataset (the default). |

## Details

Applies data pre-treatment and principal components analysis to XRPD data based based on the protocols detailed in Butler et al. (2020).

## Value

a list with components:

| | |
|---|---|
| coords | a dataframe containing the sample ID's for each sample and the PCA coordinates for each dimension |
| loadings | a dataframe containing the 2theta axis and the loading of each dimension |
| eig | a dataframe summarising the variance explained by each dimension |

## References

Butler, B.M., Sila, A.M., Shepherd, K.D., Nyambura, M., Gilmore, C.J., Kourkoumelis, N., Hillier, S., 2019. Pre-treatment of soil X-ray powder diffraction data for cluster analysis. Geoderma 337, 413-424. doi:10.4236/ampc.2013.31A007

## Examples

```
data(rockjock_mixtures)

x1 <- xrpd_pca(rockjock_mixtures,
               mean_center = TRUE,
               bin_size = 1,
               root_transform = 1)

#Plot the loading of dimension 1

plot(x = x1$loadings$tth,
     y = x1$loadings$Dim.1,
     type = "l")

## Not run:
#Fit loading 1 to the rockjock library
f1 <- fps_lm(rockjock,
             smpl = data.frame("tth" = x1$loadings$tth,
                               "counts" = x1$loadings$Dim.1),
             refs = rockjock$phases$phase_id,
             std = "QUARTZ",
             align = 0,
             p = 0.05)
```

```
plot(f1, wavelength = "Cu", interactive = TRUE)

## End(Not run)
```

# Index