Package: SpecHelpers (via r-universe)

October 5, 2026

Type Package

Title Spectroscopy Related Utilities

Version 0.3.2

Date 2025-12-03

Description Utility functions for spectroscopy. 1. Functions to simulate spectra for use in teaching or testing. 2. Functions to process files created by 'LoggerPro' and 'SpectraSuite' software.

License GPL-3

Imports gsubfn, utils, stats, graphics, splancs

URL https://github.com/bryanhanson/SpecHelpers

ByteCompile TRUE

LazyData TRUE

Encoding UTF-8

BugReports https://github.com/bryanhanson/SpecHelpers/issues

Depends R (>= 3.0)

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Repository https://bryanhanson.r-universe.dev

Date/Publication 2025-12-03 20:36:24 UTC

RemoteUrl https://github.com/bryanhanson/spechelpers

RemoteRef HEAD

RemoteSha bc56896179497ad87fdb59c300bf8a58e4cc65bc

Contents

SpecHelpers-package 2
avgLambda 3
CIExyz 3

emSpectrum 4
gatherCsv 5
gatherSpecFiles 6
gaussCurve 7
getGamutValues 8
jSeq 8
lorentzCurve 9
makeSpec 10
multiplet 12
plot2DNMRspec 13
plotCIEchrom 15
plotNMRspec 17
prepCIEgradient 19
qMS 20
txt2csv 21

Index **22**

`SpecHelpers-package` *Spectroscopy Related Utilities* *Maintainer: Bryan A. Hanson* <hanson@depauw.edu>

Description

SpecHelpers: Spectroscopy Related Utilities

Details

Utility functions for spectroscopy. 1. Functions to simulate spectra for use in teaching or testing. 2. Functions to process files created by LoggerPro and SpectraSuite software.

Author(s)

Bryan A. Hanson, DePauw University, Greencastle Indiana USA

See Also

Useful links:

- https://github.com/bryanhanson/SpecHelpers
- Report bugs at https://github.com/bryanhanson/SpecHelpers/issues

avgLambda *Convert Wavelengths to Integer Values and Average Corresponding Absorbances*

Description

This function reads a csv file containing columns of wavelength and absorbances. It rounds the wavelengths to integers and replaces the absorbances corresponding to the rounded values with their averages. NOTE THAT ALL THE csv FILES IN THE CURRENT DIRECTORY ARE PROCESSED AND REPLACED WITH THE MODIFIED FILEs. You should use this function on a copy of the directory.

Usage

```
avgLambda()
```

Value

The original files are overwritten with the modified files.

Author(s)

Bryan A. Hanson, DePauw University

See Also

`gatherSpecFiles` which is the function the user should call.

CIExyz *Spectral Locus for the 1931 CIE chromaticity diagram*

Description

This data set gives wavelengths every 1.0 nm, along with the associated CIE xyz values for the spectral locus of the 1931 CIE chromaticity diagram. They are called xyz values here as they are called that in the original source, but they are also known as xyY or XYZ values.

Usage

```
CIExyz
```

Format

A data frame with 4400 observations each with the following 4 variables:

wavelength wavelength in nm

x x values

y y values

z z values

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

Source

Color Vision Research Lab. See https://cvrl.ioo.ucl.ac.uk/index.htm (Link can be very slow). Go to this URL, then choose 'NEW CIE XYZ...' In the new page that opens, go to 'New physiologically-relevant CIE x,y chromaticity coordinates (proposed)' and get the 2-deg coordinates at 1.0 nm resolution

See Also

plotCIEchrom for examples of this data in use.

Examples

```
data(CIExyz)
```

---

emSpectrum *Plot a pretty electromagnetic spectrum*

Description

This function plots an annotated electromagnetic spectrum. There are options to include annotations about the molecular effects and/or typical applications in technology.

Usage

```
emSpectrum(molecular = TRUE, applications = TRUE)
```

Arguments

molecular	Logical. Add annotations about molecular effects?
applications	Logical. Add annotations about applications?

Value

None. Side effect is a plot.

Note

The diagram is wider than a standard R graphics device. You should send it to a pdf or similar device with the width set to 11" or so.

Details

Obviously not to scale, but hopefully aesthetically pleasing!

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

Examples

```
dev.new(width = 10.5, height = 3)
emSpectrum()
emSpectrum(molecular = FALSE, applications = FALSE)
dev.off()
```

---

gatherCsv *Combine csv Files Containing Spectral Data into a Data Frame*

Description

This function processes csv files containing two columns, wavelength and absorbance (or intensity etc), into a data frame, which is then written out as a csv file. The files should have no header row.

Usage

```
gatherCsv()
```

Details

It is assumed that the csv files have already been cleaned up so that they contain only wavelength and absorbance data. The wavelength data column must be the same in all the files (as they would be if they came from the same instrument with the same settings).

Value

A data frame containing the wavelengths in the first column and the absorbances in the other columns, one per file, with the file name generating the column name. The data frame is written out in a file called "All Spec Files.csv".

Author(s)

Bryan A. Hanson, DePauw University

See Also

gatherSpecFiles which is the function the user should call.

---

gatherSpecFiles *Process LoggerPro Spectral Files into a Data Frame*

Description

This function will go through all the files of a specified format in a directory and convert them into a data frame with one column containing the wavelength information and the other columns the absorbances of each sample (file). The file names are used to create the column names in the data frame. Optionally, non-integer wavelengths in the file can be combined to give integer wavelengths. Keep in mind that this function specifically modifies formats written by LoggerPro. Each format, as it comes from LoggerPro, has various amounts of crap in it which has to be removed or modified.

Usage

```
gatherSpecFiles(type = "txt", intLambda = FALSE, ...)
```

Arguments

type	A character string giving the type of files to be processed. Currently, either "txt", "csv" or "cmbl" extensions can be processed.
intLambda	Logical. If TRUE, non-integer wavelengths that round to the same value will be combined and averaged and reported as integer values.
...	Other parameters to be passed downstream. Currently none possible.

Details

All files of a given extension in the directory will be processed, so make certain there are no extra files in the directory. The files will be modified and written back out as .csv files so look for the number of files in the directory to double. In the case of csv files, the original csv files will be overwritten. These files have no header row.

Value

A data frame containing the wavelengths in the first column and the absorbances in the other columns, one column per file, with column names generated from the file names.

Author(s)

Bryan A. Hanson, DePauw University

gaussCurve *Compute a Gaussian Curve*

Description

Computes the y values describing a Gaussian distribution given a range of x values and parameters for mu, sigma, and area. A tail may be introduced into the curve to simulate the behavior of some chromatography peaks.

Usage

```
gaussCurve(x, area, mu, sigma, tail)
```

Arguments

x	A vector of x values which will be used to compute the corresponding y values. Use enough to give good resolution.
area	The area of the peak, in arbitrary units.
mu	The position of the peak. Must fall in the range of x, of course.
sigma	The standard deviation of the peak.
tail	A value describing any tailing desired. If NA, no tailing is applied.

Value

A vector of y values corresponding to the x values supplied.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

lorentzCurve, makeSpec which uses this function to make either spectra or chromatograms.

Examples

```
### A pure Gaussian curve

myx <- seq(0, 100, length.out = 1000) # use lots of point for resolution
myy <- gaussCurve(x = myx, area = 1, mu = 40, sigma = 1.5, tail = NA)
plot(myx, myy, type = "l", main = "Pure Gaussian Curve")

### Now with tailing

myy2 <- gaussCurve(x = myx, area = 1, mu = 40, sigma = 1.5, tail = 0.1)
plot(myx, myy2, type = "l", main = "Gaussian Curve with Tailing")
```

getGamutValues *Look up gamut and white point values in the 1931 CIE system*

Description

These functions provide a simple way of storing white point and gamut data for use in drawing CIE chromaticity diagrams.

Usage

```
getGamutValues(gamut)

getWhiteValues(white)
```

Arguments

gamut	A character string giving the name of the desired gamut. One of `c("Apple", "CIE", "Adobe", "sRGB", "NTSC", "SWOP")`.
white	The desired white point value. One of `c("D65", "E", "C", "D50")`.

Value

A data frame with columns x, y containing the vertices of the requested gamut in CIE chromaticity coordinates, or, for a white point, a data frame containing the coordinates of the requested white point.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

`plotCIEchrom` for examples of this function in use.

jSeq *Utility for Creating NMR Multiplets*

Description

This function creates sequences, centered on zero, which correspond to odd or even NMR multiplets. Not intended for direct use. Called by `plotNMRspec`.

Usage

```
jSeq(length.out)
```

Arguments

length.out	An integer giving the number of peaks in the sequence.

Value

A vector describing the spacing of the parts of an NMR multiplet in terms of multiples of the coupling constant, J.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

plotNMRspec which calls this function.

Examples

```
tmp <- jSeq(5) # a multiplet with an odd number of peaks
tmp
tmp <- jSeq(6) # an even number
tmp
```

---

lorentzCurve *Compute a Lorentzian Curve*

Description

Computes the y values describing a Lorentzian curve such as seen in an NMR peak. Requires a range of x values and parameters for peak position, area, and gamma (half the peak width at half-height).

Usage

```
lorentzCurve(x, x0, area, gamma)
```

Arguments

x	A vector of x values which will be used to compute the corresponding y values. Use enough to give good resolution.
x0	The position of the peak. Must fall in the range of x, of course.
area	The area of the peak, in arbitrary units.
gamma	HWHM, half-width at half-maximum. The peak "width" in units corresponding to x.

Value

A vector of y values corresponding to the x values supplied.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

gaussCurve, makeSpec, plotNMRspec and plot2DNMRspec for drawing NMR spectra.

Examples

```
myx <- seq(0, 100, length.out = 1000) # use lots of point for resolution
myy <- lorentzCurve(x = myx, area = 1, x0 = 40, gamma = 5)
plot(myx, myy, type = "l", main = "Pure Lorentzian Curve")
y = 0.5*max(myy)
x = seq(40, 45, 0.5)
points(x = x, y = rep(y, length(x)), col = "blue", type = "l")
text(x = 42, y = y + 0.005, labels = c("gamma"), col = "blue", srt = 90)
```

---

makeSpec *Draw a Chromatogram or Spectrum*

Description

This function creates a chromatogram or spectrum from a list of appropriate parameters describing the peaks. The individual curves are computed using the mathematical definition of either a Gaussian curve, possibly with tailing, or a Lorentzian curve. Gaussian curves are appropriate for simulating chromatograms or UV-Vis spectra, while Lorentzians are used for simulating NMR peaks. The function computes the individual curves as well as their sum (which is the whole chromatogram or spectrum). A plot can be made, which may display the separate underlying curves. If you want to draw NMR spectra, use plotNMRspec which is a much more natural interface to this function.

Usage

```
makeSpec(
  peak.list,
  x.range,
  plot = TRUE,
  curves = FALSE,
  type = "gauss",
  noise = 0,
  dd = 1,
  ...
)
```

Arguments

peak.list	For a Gaussian curve, a data frame with the following columns: mu, sd, area, tail. mu is the retention time (or center frequency). sd is the standard deviation (or peak width). area is the area under the peak. tail is the tailing parameter - use NA when a pure Gaussian with no tailing is desired. One row of the data frame contains data related to one peak. For a Lorentzian curve, a data frame with the following columns: x0, area, gamma. x0 is the center frequency or chemical shift. gamma is the half the peak width at half-height. area is the area under the peak.
x.range	A numeric vector of length 2 giving the retention time range (or frequency range) desired. Must make sense in light of the peak list given (i.e. a wider range, possibly much wider depending up the values of sd and tail), as these broaden the peaks.
plot	Logical; if TRUE, a plot is produced.
curves	Logical; if TRUE, the individual curves are plotted (provided plot = TRUE. Not very useful for NMR spectra, but great for showing, for instance, how shoulders arise on peaks in a chromatogram.
type	A character string. Use "gauss" to generate Gaussian curves (for chromatograms, or UV-Vis spectra). Use "lorentz" to generate Lorentzian curves as found in NMR spectra.
noise	A number giving the amount of noise to be added to the individual curves (the net spectrum has the noise from the individual spectra, it has no additional noise added to it). Value corresponds to the argument factor in function jitter.
dd	The density of data points per unit of x.range. The total number of data points used to create the spectrum or chromatogram is dd*abs(diff(x.range)) and thus it also depends on the units of x.range. This approach ensures that peaks are not distorted when changing x.range for the same peak.list.
...	Additional arguments to be passed downstream.

Value

A matrix containing the x values (retention times or frequencies) in the first row, and the complete chromatogram (spectrum) in the second row. Additional rows contain chromatograms (spectra) of the individual components. The row names of the data frame are character strings describing the chromatogram (spectrum) in that row. The matrix contains dd*abs(diff(x.range)) columns.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

gaussCurve, lorentzCurve, plotNMRspec and plot2DNMRspec, the preferred interfaces for drawing NMR spectra.

Examples

```
### A simple chromatogram

chrom <- data.frame(mu = c(2, 5, 11), sd = c(0.5, 1, 2),
area = c(1, 0.5, 1), tail =  c(NA, NA, 0.1))
ex1 <- makeSpec(chrom, x.range = c(0, 20), plot = TRUE, curves = TRUE,
dd = 5, main = "Chromatogram with Underlying Pure Curves")

### Faux ethyl group NMR with J = 0.1 ppm.
# Note that a much better
# NMR spectrum can be generated using plotNMRspec which also uses
# a more natural input format
#
spec <- data.frame(mu = c(3.5, 3.4, 3.3, 3.2, 1.4, 1.3, 1.2),
sd = rep(0.01, 7), tail =  rep(NA, 7),
area = c(1, 3, 3, 1, 1, 2, 1) * c(0.5, 0.5, 0.5, 0.5, 0.66, 0.66, 0.66))
ex2 <- makeSpec(spec, x.range = c(5, 0), plot = TRUE, curves = FALSE,
dd = 100, main = "Simulated 1H NMR of an Ethyl Group")
```

`multiplet` *Compute & Possibly Draw an NMR Multiplet with Optional Annotations*

Description

Serves as a teaching and self-study tool to understand complex NMR multiplets. Inspired by the Valiulin book (see the reference). One can draw a multiplet, and optionally draw the splitting tree along with annotations of the J values and guides connecting the tree to the peak maxima.

Usage

```
multiplet(
  J = c(15, 12, 2),
  pw = 0.5,
  plot = TRUE,
  plotJtree = TRUE,
  showJvalues = TRUE,
  showJtreeGuides = TRUE
)
```

Arguments

J	Numeric. A vector giving the coupling constants.
pw	Numeric. Half the peak width at half-maximum (HWHM). Passed to `makeSpec()`, and then `lorentzCurve()` where it is the gamma argument.
plot	Logical. Shall the multiplet be drawn?

plotJtree	Logical. Shall the Jtree be drawn? plot must also be TRUE in this case, and is set automatically if needed.
showJvalues	Logical. Should the J values be added to the plot? Only relevant if plotJtree = TRUE.
showJtreeGuides	Logical. Shall dotted guides be drawn between the last leaves of the Jtree and the peak maxima? Only relevant if plotJtree = TRUE.

Value

A matrix as produced by makeSpec().

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

Roman A. Valiulin *NMR Multiplet Interpretation*, 2nd Edition, de Gruyter, 2025.

Examples

```
# Example 3.1 from Valiulin, a ddt.
res <- multiplet(J = c(16.8, 10.1, 6.7, 6.7))
# Example 3.2 from Valiulin, a tt.
res <- multiplet(J = c(6.1, 6.1, 2.15, 2.15))
# Example 3.3 from Valiulin, a dddd.
res <- multiplet(J = c(12.7, 12.2, 10.0, 4.9))
# Some other nice examples
res <- multiplet(J = c(15, 12, 8, 7), pw = 0.25)
res <- multiplet(J = c(15, 8, 5, 2))
```

---

plot2DNMRspec *Draw a 2D NMR Spectrum*

Description

This function simulates 2D NMR spectra. Only 1st order coupling can be handled – there is currently no capacity for doublet of doublets and other such peaks. The field strength of the "instrument" is taken into account.

Usage

```
plot2DNMRspec(
  peaks,
  x.range = c(0, 12),
  MHz = 300,
  ppHz = 1,
```

```
  type = "COSY",
  M = NULL,
  levels = seq(0.5, 1, by = 0.1),
  ...
)
```

Arguments

peaks	A data frame with the following columns: delta, mult (multiplicity), J, area, pw. Multiplicity should be given by a number, so use 2 for a doublet. J is in Hz (use 0 for singlets). pw is the peak width at half-height in Hz.
x.range	A numeric vector of length 2 giving the ppm range desired. Must be increasing.
MHz	Integer. The operating frequency of the instrument, in MHz.
ppHz	Points per Hz: The number of data points per Hz to use in calculating the spectrum (passed as argument dd to makeSpec). The default (1) works well for 1H NMR spectra. Note that this function uses Hz internally so that the x.range, which is in ppm, is multiplied by Mhz before being sent to makeSpec, and once there, makeSpec will multiply it by ppHz. Thus the total data points used is ppHz * Mhz * abs(diff(x.range)). This approach ensures that peaks are not distorted when changing x.range for the same peak.list.
type	The type of 2D spectrum desired. One of c("COSY", "TOCSY").
M	An adjacency matrix indicating which peaks are coupled. The order of rows and columns must be the same as in peaks.
levels	A vector of levels for the contour plot. Must be in (0...1).
...	Parameters to be passed to the plotting function.

Value

Returns a matrix.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

makeSpec

Examples

```
### ethyl 2-ethyl-3-oxobutyrate
### Set up data

peaks1 <- data.frame(
#            A     B     C     D     E     F
 delta = c(4.20, 3.34, 2.23, 1.88, 1.28, 0.94),
 mult = c(4, 3, 1, 5, 3, 3),
 J = c(14, 14, 0, 14, 14, 14),
```

```
 area = c(2, 1, 3, 2, 3, 3),
 pw = c(2, 2, 2, 2, 2, 2))

#              A, B, C, D, E, F
AM <- matrix(c(0, 0, 0, 0, 1, 0,  # A
               0, 0, 0, 1, 0, 0,  # B
               0, 0, 0, 0, 0, 0,  # C
               0, 1, 0, 0, 0, 1,  # D
               1, 0, 0, 0, 0, 0,  # E
               0, 0, 0, 1, 0, 0), # F
      ncol = 6)

### 1D 1H NMR plot for reference
# CRAN checks will skip some examples to save time


jnk <- plotNMRspec(peaks = peaks1, x.range = c(0, 5), MHz = 500,
main = "1H NMR of ethyl 2-ethyl-3-oxobutyrate")

### 2D COSY plot

res <- plot2DNMRspec(peaks = peaks1, x.range = c(0, 5), MHz = 500, ppHz = 1, M = AM,
main = "COSY of ethyl 2-ethyl-3-oxobutyrate")

### 2D TOCSY plot

## Not run:

res <- plot2DNMRspec(peaks = peaks1, x.range = c(0, 5), MHz = 500, ppHz = 1,
levels = c(0.85, 0.9, 0.95), type = "TOCSY",
main = "TOCSY of ethyl 2-ethyl-3-oxobutyrate")

## End(Not run)
```

`plotCIEchrom` *Draw the 1931 CIE chromaticity diagram*

Description

This function draws the 1931 CIE chromaticity diagram with various decorations and annotations.

Usage

```
plotCIEchrom(
  gradient = NULL,
  colSpace = "sRGB",
  ex = 1,
  opts = c("D65", "specLocus", "purples"),
  title = NULL,
  ...
)
```

Arguments

gradient	Character: either "sl", NULL, or a data frame with columns x and y. If NULL, no gradient is drawn. If "sl" a gradient filling the entire spectral locus is drawn. If a data frame, the vertices should specify a polygon to be filled with the gradient (see the examples for convenient ways to specify the gradient).
colSpace	Character string giving the color space to use for drawing the gradient. One of c("sRGB", "Apple RGB"). Apple RGB is mainly of historical interest; no physical devices use it at this time.
ex	Numeric. The 'exposure' to use. The exposure must be used with **extreme care**. Larger values of exposure make the white point whiter in the plot, and lightens colors near the spectral locus (driving some off the plot!). The purpose is to alter the aesthetics of the plot - that is, to make the white "whiter" so that it looks "right". The effect of exposure will vary with the display device.
opts	A character vector of options to be employed. One or more of c("D65", "D50", "C", "E", "specLocus", "purples", "Munsell", "sRGB", "SWOP", "Apple", "NTSC", "Adobe", "CIE"). The first few of these are reference white points. "specLocus" and "purples" cause the spectral locus and line of purples to be labeled. "Munsell" causes the approximate Munsell hues to be marked along the spectral locus at the appropriate wavelength. The last few options cause the requested gamut to be outlined.
title	A character string to be plotted at the top of the diagram. If NULL, the title defaults to "1931 CIE Chromaticity Diagram". If no title is desired, set it to an empty string.
...	Additional arguments to be passed downstream, to grid functions.

Value

A plot is drawn using grid graphics.

Warning

The appearance of the color gradient will vary with the device, surface and incident light used to view it and is not likely correct anywhere. **The appearance varies strongly with exposure**.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

References

For opts = "Munsell" the Munsell designation by wavelength are taken from Romney & Indow doi:10.1073/pnas.162368999

Examples

```
require("grid")
plotCIEchrom() # no gradient
## These are a too slow for CRAN checks:
```

```
## Not run:
plotCIEchrom(gradient = "sl") # basic plot
# Notice there is not much yellow in that plot.  Increase
# the exposure to bring in some yellow, at the expense of some blues:
plotCIEchrom(gradient = "sl", ex = 1.4)
# Next show a gradient for the CMYK printing process
# and outline the colors a typical monitor can display.
plotCIEchrom(gradient = getGamutValues("SWOP"), opts = c("D65", "SWOP", "sRGB"))

## End(Not run)
```

`plotNMRspec` *Create and Plot an NMR Spectrum*

Description

This function simulates simple NMR spectra. Only 1st order coupling can be handled – there is currently no capacity for doublet of doublets and other such peaks. The field strength of the "instrument" is taken into account.

Usage

```
plotNMRspec(
  peaks,
  x.range = c(12, 0),
  MHz = 300,
  ppHz = 1,
  nuclei = "1H",
  pkLabs = TRUE,
  lab.pos = NULL,
  plot = TRUE,
  ...
)
```

Arguments

`peaks`	A data frame with the following columns: delta, mult (multiplicity), J, area, pw. Multiplicity should be given by a number, so use 2 for a doublet. J is in Hz (use 0 for singlets). pw is the peak width at half-height in Hz.
`x.range`	A numeric vector of length 2 giving the ppm range desired.
`MHz`	Integer. The operating frequency of the instrument, in MHz.
`ppHz`	Integer, but numeric works too! Points per Hz: The number of data points per Hz to use in calculating the spectrum (passed as argument `dd` to `makeSpec`). The default (1) works well for 1H NMR spectra. For 13C NMR spectra, where the peaks are very narrow, one may need to increase the data density so that enough points define the peaks (a value of 4 is a good starting point). See Details.

nuclei	Character. One of c("1H", "13C"). Controls the spacing of the tick marks and labeling of the peaks.
pkLabs	Logical. If TRUE, and nuclei = 1H, the integral is drawn next to the peak. If FALSE, no labels are drawn.
lab.pos	A vector of label positions as along as the number of rows in peaks (the number of peaks in the spectrum). A numeric vector where 2 = left and 4 = right. This adjusts the positions of the labels to be either left or right of the peak as a way to avoid overlaps. The order must correspond to the order in peaks.
plot	Logical: Shall a plot be made?
...	Other parameters to be passed downstream. These may affect the plot. You can also include noise = some number to add noise (passed through to makeSpec). In this case, warnings are raised from the plotting routines, but they can be ignored.

Value

Returns a data frame of the type produced by makeSpec. See there for details. x values are in Hz.

Details

Note that this function uses Hz internally so that the x.range, which is in ppm, is multiplied by Mhz before being sent to makeSpec, and once there, makeSpec will multiply it by ppHz. Thus the total data points used is floor(ppHz * Mhz * abs(diff(x.range))). This approach ensures that peaks are not distorted when changing x.range for the same peak.list.

Note that ppHz can be numeric as well, due to the use of floor. This can be useful: if you wanted your simulated NMR spectrum to be composed of exactly 16384 data points as real data might be, you can call the function with ppHz specified like ppHz = 2^14/(12*500) and it works!

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

lorentzCurve, makeSpec

Examples

```
### A simulated 1H NMR spectrum

peaks1 <- data.frame(
 delta = c(1.3, 3.75, 3.9, 10.2),
 mult = c(3, 4, 2, 1),
 J = c(14, 14, 14, 0),
 area = c(3, 2, 1, 1),
 pw = c(2, 2, 2, 10))

res <- plotNMRspec(peaks1, x.range = c(12, 0), MHz = 500,
 main = "500 MHz Simulated 1H NMR Spectrum")
```

```
### Compare to the same data at 200 MHz and plot together

par(mfrow = c(2,1))
res <- plotNMRspec(peaks1, x.range = c(12, 0), MHz = 500,
 main = "500 MHz Simulated 1H NMR Spectrum")
res <- plotNMRspec(peaks1, x.range = c(12, 0), MHz = 200,
 main = "200 MHz Simulated 1H NMR Spectrum")
par(mfrow = c(1,1))

### Zoom in to show off

par(mfrow = c(2,1))
res <- plotNMRspec(peaks1, x.range = c(4.5, 1), MHz = 500,
 main = "500 MHz Simulated 1H NMR Spectrum")
res <- plotNMRspec(peaks1, x.range = c(4.5, 1), MHz = 200,
 main = "200 MHz Simulated 1H NMR Spectrum")
par(mfrow = c(1,1))

### A simulated 13C NMR spectrum

# This is substantially slower due to the large
# chemical shift range

peaks2 <- data.frame(
 delta = c(160, 155, 145, 143, 135, 60, 32),
 mult = rep(1, 7),
 J = rep(1, 7),
 area = c(0.1, 0.3, 0.3, 1, 1, 0.5, 0.5),
 pw = rep(1, 7))

res <- plotNMRspec(peaks2, x.range = c(180, 0), MHz = 200,
 main = "200 MHz Simulated 13C NMR Spectrum", ppHz = 4,
 pkLabs = FALSE, nuclei = "13C")

# Try repeating the above with ppHz = 1; note the peaks heights are not quite right
# as there are not enough data points to define the peak properly.
```

`prepCIEgradient` *Compute a gradient to fill the CIE chromaticity diagram*

Description

This function creates a gradient to fill the CIE chromaticity diagram.

Usage

```
prepCIEgradient(vertices = NULL, colSpace = "sRGB", ex = 1, ...)
```

Arguments

vertices	The vertices of a polygon that is to be filled with the gradient.
colSpace	Character. The color space model to use.
ex	Numeric. The exposure factor. This shifts the gradient. Be extremely careful with this. See plotCIEchrom for full details.
...	Arguments to be passed downstream.

Value

An array containing the data needed to draw the gradient.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

See Also

plotCIEchrom for examples of this function in use.

---

qMS *Draw a Simple Mass Spectrum Showing the Parent Ion*

Description

Given a molecular formula, this function computes the mass of the parent ion, including any M + n peaks due to Br or Cl, and plots it. Intended to draw the parent ion region for small organic molecules, especially those with Br or Cl.

Usage

```
qMS(f = NULL, xlab = "m/z", ylab = "intensity", main = "Mass Spectrum", ...)
```

Arguments

f	A character string giving the molecular formula of the molecule of interest. Order of elements does not matter. Elements should be given as their atomic symbols, e.g. "Br" not "br".
xlab	A character string giving the x axis label.
ylab	A character string giving the y axis label.
main	A character string giving the title of the plot.
...	Additional arguments to be passed downstream.

Details

The function currently accepts formulas containing C, H, N, O, Br and Cl in any quantities.

Value

Draws a plot. Returns a data frame giving the peak masses and relative intensites.

Author(s)

Bryan A. Hanson, DePauw University. <hanson@depauw.edu>

Examples

```
ms <- qMS(f = "C5H8BrCl", xlim = c(150, 200), main = "Parent Ion of C5H8BrCl")
```

---

txt2csv *Utility Functions to Clean and Convert Spectral Files to csv*

Description

These functions clean out extraneous information from exported spectral data files and then write them out in csv format. `txt2csv` and `cmbl2csv` handle files exported by LoggerPro software. `sstab2csv` handles files exported by Spectra Suite software. Not directly called by the user.

Usage

```
txt2csv(in.file = "", out.file = "")
```

Arguments

`in.file`	The name of the input file.
`out.file`	The name of the output file.

Details

Extraneous text at the beginning of the file is removed. In the case of cmbl files, lines containing "Z2" or ">" are removed. Absorbances marked as "Z1" are replaced with zero. The data are initially in one long column; the wavelength and absorbances are reunited into two columns.

Value

A modifed file in csv format.

Author(s)

Bryan A. Hanson, DePauw University.

See Also

`gatherSpecFiles` which is the function the user should call.

Index

∗ **data**
  CIExyz, 3
∗ **distributions**
  lorentzCurve, 9
∗ **hplot**
  plotCIEchrom, 15
∗ **package**
  SpecHelpers-package, 2
∗ **utilities**
  avgLambda, 3
  emSpectrum, 4
  gatherCsv, 5
  gatherSpecFiles, 6
  gaussCurve, 7
  getGamutValues, 8
  jSeq, 8
  lorentzCurve, 9
  makeSpec, 10
  multiplet, 12
  plot2DNMRspec, 13
  plotNMRspec, 17
  prepCIEgradient, 19
  qMS, 20
  txt2csv, 21

avgLambda, 3

CIExyz, 3
cmbl2csv *(*txt2csv*)*, 21

emSpectrum, 4

gatherCsv, 5
gatherSpecFiles, *3*, 6, *6*, *21*
gaussCurve, 7, *10, 11*
getGamutValues, 8
getWhiteValues *(*getGamutValues*)*, 8

jSeq, 8

lorentzCurve, *7*, 9, *11*, *18*
lorentzCurve(), *12*

makeSpec, *7*, 10, *10*, *14*, *18*
makeSpec(), *12, 13*
multiplet, 12

plot2DNMRspec, *10, 11*, 13
plotCIEchrom, *4*, *8*, 15, *20*
plotNMRspec, *8–11*, 17
prepCIEgradient, 19

qMS, 20

SpecHelpers *(*SpecHelpers-package*)*, 2
SpecHelpers-package, 2
sstab2csv *(*txt2csv*)*, 21

txt2csv, 21