Package ‘spacesXYZ’

January 17, 2025

Type Package

Title CIE XYZ and some of Its Derived Color Spaces

Version 1.4-0

Encoding UTF-8

Date 2025-01-17

Maintainer Glenn Davis <gdavis@gluonics.com>

Description Functions for converting among CIE XYZ, xyY, Lab, and Luv.
Calculate Correlated Color Temperature (CCT) and the Planckian and daylight loci.
The XYZs of some standard illuminants and some standard linear chromatic adaptation transforms (CATs) are included.
Three standard color difference metrics are included.

License GPL (>= 3)

LazyLoad yes

LazyData yes

Depends R (>= 4.0.0)

Imports logger

Suggests knitr, rmarkdown, microbenchmark

Enhances colorSpec

Repository CRAN

VignetteBuilder knitr

BuildVignettes yes

NeedsCompilation no

Author Glenn Davis [aut, cre]

Date/Publication 2025-01-17 06:10:02 UTC

Contents

`spacesXYZ-package` *CIE XYZ and some of Its Derived Color Spaces*

Description

This package covers the basic CIE 1931 space, and derived spaces CIE xyY, Lab, and Luv. The equations are taken from Bruce Lindbloom's *CIE Color Calculator*. Color areas that are *not* covered are:

- spectral color data
- device color spaces, e.g. RGB and CMYK
- color order systems, e.g. Munsell, DIN, NCS, Ostwald, ...

The API

The API is small. There are functions to

- convert between CIE XYZ and other CIE spaces
- create and perform some standard chromatic adaptation transforms (CATs)
- compute 3 standard color difference ΔE metrics
- retrieve XYZ and xy of some standard illuminants

Other Packages

Package **colorscience** is a superset of this one.
Packages **colorspace** and **farver** have similar functionality, and are faster because they are in compiled C/C++.
Package **grDevices** also has similar functionality (in the function `convertColor()`), but is missing chromaticities xy, uv, and u'v'.

Logging

Logging is performed using the package **logger**. This is a powerful package that allows a separate configuration for logging from within **spacesXYZ**, and that is what I have done. When **spacesXYZ** is loaded, the logging threshold is changed from `INFO` to `WARN`. To change it back again, one can execute:

`logger::log_threshold( logger::INFO, namespace="spacesXYZ" ) # preferred`
or
`library( logger )`
`log_threshold( INFO, namespace="spacesXYZ" ) # not preferred`

The layout callback function is customized; it adds the name of the calling function to the message. To change it back again, one can execute:
`log_layout( layout_simple, namespace="spacesXYZ" )`
or to install ones own layout function, one can execute:
`log_layout( <your function>, namespace="spacesXYZ" ).`

The appender callback functions is also customized; it comes to an immediate stop if the log event level is `FATAL` or `ERROR`. To continue execution after such a log event, one can execute:
`log_appender( appender_console, namespace="spacesXYZ" )`

The formatter callback function is initialized to be `formatter_sprintf()`; this should not be changed.

Author(s)

Glenn Davis <gdavis@gluonics.com>

References

Lindbloom, Bruce. **CIE Color Calculator**. http://brucelindbloom.com/index.html?ColorCalculator.html

Lindbloom, Bruce. **Color Difference Calculator**. http://brucelindbloom.com/index.html?ColorDifferenceCalc.html

---

`adapt` *Chromatic Adaptation Functions*

Description

Adapt XYZ, xyY, Lab, or Luv from a source viewing enviroment with a given illuminant, to a target viewing environment with a different illuminant.

Usage

```
## S3 method for class 'CAT'
adaptXYZ( x, XYZ.src )

## S3 method for class 'CAT'
```

```
adaptxyY( x, xyY.src )
## S3 method for class 'CAT'
adaptLab( x, Lab.src )
## S3 method for class 'CAT'
adaptLuv( x, Luv.src )
```

Arguments

`x`	a **CAT** object as returned from `CAT()`
`XYZ.src`	an Nx3 matrix, or a vector that can be converted to such a matrix, by rows. Each row has an XYZ in the source viewing environment.
`xyY.src`	an Nx3 matrix, or a vector that can be converted to such a matrix, by rows. Each row has an xyY in the source viewing environment.
`Lab.src`	an Nx3 matrix, or a vector that can be converted to such a matrix, by rows. Each row has an Lab in the source viewing environment.
`Luv.src`	an Nx3 matrix, or a vector that can be converted to such a matrix, by rows. Each row has an Luv in the source viewing environment.

Details

`adaptXYZ()` is the most fundamental of the group; it simply multiplies each of the input XYZs by `x$M`.

`adaptxyY()` converts xyY.src to XYZ, calls `adaptXYZ()`, and then converts back to xyY.tgt. And it does an additional check: if the xy of xyY.src is equal to the xy of `x$source.xyY`, then the xy of the returned xyY.tgt is set to be the xy of `x$target.xyY`.

`adaptLab()` and `adaptLuv()` work in a similar way. When `Lab.src` is transformed to XYZ, the whitepoint is set to `x$source.XYZ`. And when the adapted XYZ is transformed to adapted Lab, the whitepoint is set to `target.XYZ`.

Value

`adaptXYZ()` returns an Nx3 matrix with adapted XYZ. Each row has an XYZ in the target viewing environment.
`adaptxyY()` returns an Nx3 matrix with adapted xyY. Each row has an xyY in the target viewing environment.

`adaptLab()` and `adaptLuv()` return adapted Lab and Luv respectively.

References

Hunt, R. W. G. **The Reproduction of Colour**. 6th Edition. John Wiley & Sons. 2004.

International Color Consortium. ICC.1:2001-04. File Format for Color Profiles. 2001.

Lindbloom, Bruce. Chromatic Adaptation. `http://brucelindbloom.com/Eqn_ChromAdapt.html`

Wikipedia. CIECAM02. `https://en.wikipedia.org/wiki/CIECAM02`

See Also

`CAT()`, `standardXYZ()`

Examples

```
# try the Bradford method
bCAT = CAT( 'D50', 'D65', method='bradford' )

adaptXYZ( bCAT, c(1,1,0.5) )
##                X         Y         Z
##  [1,] 0.9641191 0.9921559 0.6567701


adaptLab( bCAT, c(50,20,-10) )
##              L        a         b
##  [1,] 49.97396 20.84287 -10.19661      # as expected, there is a change

adaptLab( bCAT, c(40,0,0) )
##        L a b
##  [1,] 40 0 0   # but adaptLab() always preserves neutrals


adaptLuv( bCAT, c(40,0,0) )
##        L u v
##  [1,] 40 0 0   # and adaptLuv() also preserves neutrals



# try the scaling method - now XYZ are scaled independently
sCAT = CAT( 'D50', 'D65', method='scaling' )

adaptLab( sCAT, c(50,20,-10) )
##        L  a   b
##  [1,] 50 20 -10    with sCAT, adaptLab() is now the identity for *all* colors


adaptLuv( sCAT, c(50,-20,10) )
##        L         u        v
##  [1,] 50 -18.32244 11.29946    but adaptLuv() is NOT the identity for all colors
```

`adaptation` *Chromatic Adaptation Transforms (CATs)*

Description

Construct transforms from a source viewing enviroment with a given illuminant, to a target viewing environment with a different illuminant. Some standard linear von-Kries-based CAT methods are available.

Usage

```
CAT( source.XYZ, target.XYZ, method="Bradford" )
```

Arguments

`source.XYZ`	the XYZ of the illuminant in the source viewing environment. `source.XYZ` can also be a string with the name of a standard illuminant as in the function `standardXYZ()`.
`target.XYZ`	the XYZ of the illuminant in the target viewing environment. `target.XYZ` can also be a string with the name of a standard illuminant as in the function `standardXYZ()`.
`method`	the method used for the chromatic adaptation. Available methods are: `"Bradford"`, `"VonKries"`, `"MCAT02"`, `"Bianco+Schettini"`, and `"scaling"`; see **References**. Partial matching is enabled, and matching is case-insensitive. `method` can also be a 3x3 matrix, which is the cone response matrix used to construct a von-Kries-based CAT. The matrix must be invertible and map both `source.XYZ` and `target.XYZ` to the positive octant.

Value

`CAT()` returns an object with S3 class **CAT**, which can be passed to `adaptXYZ()`, `adaptxyY()`, `adaptLab()`, or `adaptLuv()`.

An object with S3 class **CAT** is a list with the following items:

method full name of the adaptation method, as in **Arguments**. If argument `method` is a 3x3 matrix, then this `method` is `NA`.

Ma 3x3 *cone response matrix* M_A for the method, as defined in *Lindbloom*

source.XYZ XYZ of the illuminant in the source viewing environment

source.xyY xyY of the illuminant in the source viewing environment

target.XYZ XYZ of the illuminant in the target viewing environment

target.xyY xyY of the illuminant in the target viewing environment

M 3x3 matrix defining the CAT. The matrix is written on the left and the source XYZ is written as a column vector on the right. This matrix depends continuously on source.XYZ and target.XYZ, and when these are equal, M is the identity. Therefore, when source.XYZ and target.XYZ are close, M is close to the identity. Compare with *Lindbloom*.

Note

Chromatic adaptation can be viewed as an Aristotelian Analogy of Proportions. For more about this, see the vignette **Chromatic Adaptation**.

References

Bianco, Simone and Raimondo Schettini. **Two new von Kries based chromatic adaptation transforms found by numerical optimization**. Color Research & Application. v. 35. i. 3. Jan 2010.

Hunt, R. W. G. **The Reproduction of Colour**. 6th Edition. John Wiley & Sons. 2004.

International Color Consortium. ICC.1:2001-04. File Format for Color Profiles. 2001.

Lindbloom, Bruce. Chromatic Adaptation. `http://brucelindbloom.com/Eqn_ChromAdapt.html`

Pascale, Danny. A Review of RGB Color Spaces ...from xyY to R'G'B'. https://babelcolor.com/index_htm_files/A%20review%20of%20RGB%20color%20spaces.pdf 2003.

Wikipedia. CIECAM02. https://en.wikipedia.org/wiki/CIECAM02

See Also

standardXYZ(), adaptXYZ(), adaptxyY(), adaptLab(), adaptLuv()

Examples

```
D65toC = CAT( 'D65', 'C' )
D65toC

##  $method
##  [1] "Bradford"
##
##  $Ma
##          X       Y       Z
##  L  0.8951  0.2664 -0.1614
##  M -0.7502  1.7135  0.0367
##  S  0.0389 -0.0685  1.0296
##
##  $source.XYZ
##            X Y       Z
##  D65 0.95047 1 1.08883
##
##  $source.xyY
##              x         y Y
##  D65 0.3127266 0.3290231 1
##
##  $target.XYZ
##          X Y       Z
##  C 0.98074 1 1.18232
##
##  $target.xyY
##            x         y Y
##  C 0.3100605 0.3161496 1
##
##  $M
##              X            Y           Z
##  X 1.009778519  0.007041913 0.012797129
##  Y 0.012311347  0.984709398 0.003296232
##  Z 0.003828375 -0.007233061 1.089163878
##
##  attr(,"class")
##  [1] "CAT"  "list"


adaptXYZ( D65toC, c(1,1,0.5) )
##              X         Y         Z
##  [1,] 1.023219 0.9986689 0.5411773
```

`Correlated Color Temperature` *Compute Correlated Color Temperature (CCT) and Points on the Planckian Locus*

Description

Compute the CCT in Kelvin, of XYZ, xy, and uv, by multiple methods. And compute points on the Planckian locus.

The *reference* Planckian locus is defined spectrally - from the famous equation for the Planckian radiator (with $c_2 = 1.4388 \times 10^{-2}$) and from the tabulated CIE 1931 standard observer color matching functions, from 360 to 830nm in 1nm steps. The reference locus is a C^∞ curve, parameterized by temperature, in a 2D chromaticity space, usually either xy (1931) or uv (1960). Computing uv values (and derivatives) is lengthy because there are 471 wavelengths. An approximation to the reference locus is desirable.

The default locus approximation is a spline (using `stats::splinefun()` with `method="fmm"`) through the 31 uv locus points in *Robertson* and *Wyszecki & Stiles*. This spline does not appear in *Robertson*, but I think he would approve of it. It has C^2 continuity and good agreement with the reference locus. The maximum RMS error in the uv-plane is about 7.3×10^{-6} over the valid temperature interval [1667,∞] K. A similar piecewise-linear interpolating path has a maximum RMS error about 10 times larger. The 31 uv values in the table are accurate to the given 5 decimal places (but see **Note**), and the rounding to 5 places is a big limitation in accuracy. The locus is parameterized directly by reciprocal color temperature ($10^6/T$), and therefore indirectly by T. We call either of these the *native pameterization* of the locus. See Planckian Loci. The lines that are perpendicular to the locus are called the *native isotherms*.

The second available locus is a quintic spline through 65 points (knots), that were computed and saved with full precision. The maximum RMS error in the uv-plane is about 7.2×10^{-12} over the valid temperature interval [1000,∞] K. For this one the 1st and 2nd derivatives were also computed, so the normal vectors at the knots are accurate, and the curve is also C^2. See Planckian Loci. The lines that are perpendicular to the locus are also called the *native isotherms*.

Two more families of isotherms are available. The *Robertson* isotherms are tabulated just like the points on the locus, and a special linear interpolation is used for intermediate temperatures. The *McCamy* isotherms are defined by a single cubic rational function in xy, and no interpolation is necessary. Each isotherm family *induces* a slightly different parameterization of the locus - the temperature at a locus point is the temperature of the isotherm passing through that point.
The Robertson parameterization is only continuous of class C^0, but the *geometric continuity* class is G^2. The McCamy parameterization is as smooth as the locus itself, which is C^2.
For the Robertson parameterization the valid temperature interval is [1667,∞] K. For the McCamy parameterization the valid temperature interval is at most [1621,34530] K, and may be smaller depending on the locus.

Usage

```
CCTfromXYZ( XYZ, isotherms='robertson',  locus='robertson', strict=FALSE )
CCTfromxy( xy, isotherms='robertson',  locus='robertson', strict=FALSE )
CCTfromuv( uv, isotherms='robertson', locus='robertson', strict=FALSE )
```

```
planckLocus( temperature, locus='robertson', param='robertson', delta=0, space=1960 )
```

Arguments

`XYZ`	a numeric Mx3 matrix with XYZ tristimulus values (CIE 1931) in the rows, or a numeric vector that can be converted to such a matrix, by row.
`xy`	a numeric Mx2 matrix with xy chromaticity values (CIE 1931) in the rows, or a numeric vector that can be converted to such a matrix, by row.
`uv`	a numeric Mx2 matrix with uv chromaticity values (CIE UCS 1960) in the rows, or a numeric vector that can be converted to such a matrix, by row.
`isotherms`	A character vector whose elements match one of the available isotherm families: `'robertson'`, `'mccamy'`, and `'native'`. Matching is partial and case-insensitive. When more than one family is given, a matrix is returned, see **Value**. When `isotherms='native'` the isotherms are defined implicitly as lines perpendicular to the locus, see **Details**. The character `NA` (`NA_character_`) is taken as a synonym for `'native'`.
`locus`	valid values are `'robertson'` and `'precision'`, see above. Matching is partial and case-insensitive.
`strict`	The CIE considers the CCT of a chromaticity `uv` to be meaningful only if the distance from `uv` to the Planckian locus is less than or equal to 0.05 [in CIE UCS 1960]. If `strict=FALSE`, then this condition is ignored. Otherwise, the distance is computed along the corresponding isotherm, and if it exceeds 0.05 the returned CCT is set to `NA`.
`temperature`	a M-vector of temperatures (in K) at which to compute points on the Planckian locus, either for `uv`, `u'v'`, or `xy`; see `space`.
`param`	the desired parameterization of the locus. It can be either `'native'`, or a parameterization induced by the `'robertson'` or `'mccamy'` isotherms. The character `NA` (`NA_character_`) is taken as a synonym for `'native'`.
`delta`	a vector of offset distances in `uv` (CIE UCS 1960), along the corresponding isotherms, from the locus. Positive offsets are above the locus, and negative are below. `delta` can have length M or 1, where M is the length of `temperature`. If `delta` has length 1, that value is replicated to length M.
`space`	the year of the chromaticity space to return. Valid values are 1960 (the default `uv`), 1976 (`u'v'`), and 1931 (`xy`).

Details

Each of the isotherm families correspond to a parameterization of the locus. All this is designed so a round trip: temperature $\rightarrow$ uv $\rightarrow$ CCT (with the same choice of isotherm/parameterization) has neglible error.

When `isotherms='native'` the tangent line at a point on the locus is computed using the `deriv=1` argument to `stats::splinefun()` and the normal line - the isotherm at the point - is then easily computed from the tangent line.

When `isotherms='robertson'` or `isotherms='mccamy'` the locus curve has no effect on the computed CCT. The locus is only used when computing the distance from the given uv point to the locus (along the corresponding isotherm), and therefore only affects the decision whether the CCT is meaningful when `strict=TRUE`.

Value

`CCTfromXYZ()`, `CCTfromxy()`, and `CCTfromuv()` return a numeric vector of length M, or an MxN matrix. It returns a matrix iff `length(isotherms)` = `N` ≥ 2, and the column names are set to the isotherm family names. The names or rownames are set to the rownames of the input. In case of error, the element of the vector or matrix is set to `NA_real_`. In case there is an error in the arguments, the functions return `NULL`. In these functions, the locus is not used unless `isotherms='native'` or `strict=TRUE`.

`planckLocus()` returns an Mx2 matrix with chromaticies in the rows. The column names are set appropriately for the value of `space`. The row names are set from `temperature`. In case of a single error, both entries in the row are set to `NA_real_`. In case there is an error in the arguments, the functions return `NULL`.

Note

The lookup table on page 228 in *Wyszecki & Stiles* contains an error at 325 mired, which was corrected by Bruce Lindbloom (see **Source**).

Source

`http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_T.html`

References

McCamy, C. S. *Correlated color temperature as an explicit function of chromaticity coordinates*. Color Research & Application. Volume 17. Issue 2. pages 142-144. April 1992.

Robertson, A. R. Computation of correlated color temperature and distribution temperature. Journal of the Optical Society of America. 58. pp. 1528-1535 (1968).

Wyszecki, Günther and W. S. Stiles. **Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition.** John Wiley & Sons, 1982. Table 1(3.11). pp. 227-228.

See Also

`stats::splinefun()`, `colorSpec::computeCCT()`, `RobertsonLocus`, `PrecisionLocus`, the vignette **Correlated Color Temperature Isotherms**

Examples

```
# do a round trip and then compare
temperature = c(5003,6504)
uv  = planckLocus( temperature, delta=0.05 )
CCTfromuv( uv ) - temperature
##  2.772227e-05 5.094369e-05

# find some points on the daylight locus, and then their CCT
temperature = seq( 2000, 10000, by=1000 )
xy = daylightLocus( temperature )
```

```
cbind( xy, CCT=CCTfromxy(xy,iso='mccamy') )
##                 x         y      CCT
## D2000          NA        NA       NA
## D3000          NA        NA       NA
## D4000  0.3823436 0.3837663 4005.717
## D5000  0.3457410 0.3586662 4999.998
## D6000  0.3216915 0.3377984 5999.437
## D7000  0.3053570 0.3216459 6997.542
## D8000  0.2937719 0.3092195 7985.318
## D9000  0.2852645 0.2995816 8948.809
## D10000 0.2787996 0.2919672 9881.115

# compare all 3 different isotherms
CCTfromxy( xy, isotherms=c('robertson',NA,'mccamy') )
##        Robertson   native   McCamy
## D2000         NA       NA       NA
## D3000         NA       NA       NA
## D4000   4000.096 4000.062 4005.717
## D5000   4999.749 4999.608 4999.998
## D6000   5998.015 5999.242 5999.437
## D7000   6997.858 6998.258 6997.542
## D8000   7997.599 7996.985 7985.318
## D9000   8999.301 8993.811 8948.809
## D10000  9991.920 9992.672 9881.115

cbind( default=CCTfromxy(xy), prec.native=CCTfromxy(xy,locus='prec',iso=NA) )
##         default prec.native
## 2000K        NA          NA
## 3000K        NA          NA
## 4000K  4000.096    4000.052
## 5000K  4999.749    4999.767
## 6000K  5998.015    5999.097
## 7000K  6997.858    6997.857
## 8000K  7997.599    7997.951
## 9000K  8999.301    8995.835
## 10000K 9991.920    9992.839
```

daylight — *Compute Points on the Daylight Locus, in multiple Chromaticity Spaces*

Description

Compute points on the daylight locus, in multiple chromaticity spaces

Usage

```
daylightLocus( temperature, space=1931 )
```

Arguments

`temperature`	an M-vector of temperatures (in K) at which to compute points on the daylight locus. The valid temperatures range is 4000K to 25000K; outside this range the output is set to `NA`.
`space`	the year of the output chromaticity space desired - valid values are 1931, 1976 and 1960. The 1931 is the original and denoted by xy; the others are derived from it. The 1960 coordinates are usually denoted by uv, and the 1976 coordinates by u'v'. The only difference is that $v' = (3/2)v$.

Value

a numeric Mx2 matrix with xy, uv, or u'v' coordinates in the rows. The colnames of the output are set appropriately.

The names of the temperature are copied to the rownames of the output, unless these names are `NULL` when the temperatures followed by 'K' are used.

If the input is invalid, the function returns `NULL`.

References

Wyszecki, Günther and W. S. Stiles. **Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition.** John Wiley & Sons, 1982. pp. 145-146.

See Also

`uvfromxy()`

Examples

```
# find some points on the daylight locus, and then their CCT
temp = seq( 2000, 10000, by=1000 )
xy = daylightLocus( temp )
cbind( xy, CCT=CCTfromxy(xy) )
##                x         y      CCT
## D2000         NA        NA       NA
## D3000         NA        NA       NA
## D4000  0.3823436 0.3837663 4000.096
## D5000  0.3457410 0.3586662 4999.749
## D6000  0.3216915 0.3377984 5998.015
## D7000  0.3053570 0.3216459 6997.858
## D8000  0.2937719 0.3092195 7997.599
## D9000  0.2852645 0.2995816 8999.301
## D10000 0.2787996 0.2919672 9991.920
```

DeltaE *Calculate the Color Difference between Two Colors*

Description

Calculate Standard CIE Color Differences between two Colors

Usage

```
DeltaE( Lab1, Lab2, metric=1976 )
```

Arguments

`Lab1`	a numeric Nx3 matrix with Lab values in the rows, or a vector that can be converted to such a matrix, by row. `Lab1` can also be a numeric 3-vector with a single Lab, and it is then replicated to match the size of `Lab2`.
`Lab2`	a numeric Nx3 matrix with Lab values in the rows, or a vector that can be converted to such a matrix, by row. `Lab2` can also be a numeric 3-vector with a single Lab, and it is then replicated to match the size of `Lab1`.
`metric`	a vector of color metric specifiers. Valid values are `'1976'`, `'1994'`, and `'2000'`, which refer to the year the metric was recommended by the CIE. They can also be given as integers, as shown. These metrics are often denoted ΔE_{1976}, ΔE_{1994}, and ΔE_{2000}. When more than one metric is given, a matrix is returned, see **Value**.

Value

`DeltaE()` returns a numeric vector of length N, or an NxM matrix. It returns a matrix iff `length(metric)=` M$\geq$ 2; the column names are set to the metric names. The elements of the output are the pairwise differences, i.e. between row i of `Lab1` and row i of `Lab2`. The names or rownames are set to the rownames of one of the input matrices.

For `metric=1976` the distance is simply the Euclidean distance between the two points in Lab, see *Hunt* p. 111.
For `metric=1994` the symmetric variant is used, see *Hunt* p. 670. There is an asymmetric variant which is not available in this package. The weighting coefficients are for **graphic arts** (not for **textiles**).
For `metric=2000` the distance is insanely complicated, see *Hunt* p. 671.
All these metrics are *symmetric*, which means that swapping `Lab1` and `Lab2` does not change the result.

References

Hunt, R. W. G. **The Reproduction of Colour**. 6th Edition. John Wiley & Sons. 2004.

Examples

```
DeltaE( c(50,0,0),  c(51,2,2,  52,10,11,  46,-13,16)  )
## [1]  3 15 21

path = system.file( "extdata/ciede2000testdata.txt", package='spacesXYZ' )
df   = read.table( path, sep='\t', quote='', head=TRUE )
Lab1 = as.matrix( df[ , 1:3 ] )
Lab2 = as.matrix( df[ , 4:6 ] )
cbind( Lab1, Lab2, DeltaE( Lab1, Lab2, metric=c(1976,2000) ) )[ 1:10, ]

##      LAB_L_REF LAB_A_REF LAB_B_REF LAB_L_SAM LAB_A_SAM LAB_B_SAM DeltaE.1976 DeltaE.2000
##  [1,]   50.0000    2.6772  -79.7751   50.0000    0.0000  -82.7485   4.0010633   2.0424597
##  [2,]   50.0000    3.1571  -77.2803   50.0000    0.0000  -82.7485   6.3141501   2.8615102
##  [3,]   50.0000    2.8361  -74.0200   50.0000    0.0000  -82.7485   9.1776999   3.4411906
##  [4,]   50.0000   -1.3802  -84.2814   50.0000    0.0000  -82.7485   2.0627008   0.9999989
##  [5,]   50.0000   -1.1848  -84.8006   50.0000    0.0000  -82.7485   2.3695707   1.0000047
##  [6,]   50.0000   -0.9009  -85.5211   50.0000    0.0000  -82.7485   2.9152927   1.0000130
##  [7,]   50.0000    0.0000    0.0000   50.0000   -1.0000    2.0000   2.2360680   2.3668588
##  [8,]   50.0000   -1.0000    2.0000   50.0000    0.0000    0.0000   2.2360680   2.3668588
##  [9,]   50.0000    2.4900   -0.0010   50.0000   -2.4900    0.0009   4.9800004   7.1791720
## [10,]   50.0000    2.4900   -0.0010   50.0000   -2.4900    0.0010   4.9800004   7.1791626
```

---

fromPolar *Convert CIE Lab and Luv from Polar Form to Rectangular Form*

Description

Convert the Polar Form of CIE Lab and Luv to Rectangular Form

Usage

```
LabfromLCHab( LCHab )
LuvfromLCHuv( LCHuv )
```

Arguments

`LCHab`	a numeric Nx3 matrix with CIE LCHab coordinates in the rows, or a vector that can be converted to such a matrix, by row. The hue angle H must be in degrees.
`LCHuv`	a numeric Nx3 matrix with CIE LCHuv coordinates in the rows, or a vector that can be converted to such a matrix, by row. The hue angle H must be in degrees.

Value

`LabfromLCHab()`	returns a numeric Nx3 matrix with CIE Lab coordinates in the rows.
`LuvfromLCHuv()`	returns a numeric Nx3 matrix with CIE Luv coordinates in the rows.

In both cases, the rownames are copied from input to output. If the input is invalid, the functions return `NULL`.

References

Wikipedia. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

See Also

`LCHabfromLab()`, `LCHuvfromLuv()`

Examples

```
LabfromLCHab( c(50,10,45) )
##        L        a        b
##  [1,] 50 7.071068 7.071068     #  on line with slope 1
```

---

fromXYZ *Convert from XYZ to other Color Spaces*

Description

Convert from XYZ to other Color Spaces

Usage

```
xyYfromXYZ( XYZ )
LabfromXYZ( XYZ, white )
LuvfromXYZ( XYZ, white )
```

Arguments

`XYZ`	a numeric Nx3 matrix with CIE XYZ coordinates in the rows, or a vector that can be converted to such a matrix, by row.
`white`	a numeric 3-vector giving the XYZ of reference white; all 3 numbers must be positive. `white` can also be a character string with the name of a standard illuminant, which is passed to `standardXYZ()` to get the XYZ.

Value

`xyYfromXYZ()`	returns a numeric Nx3 matrix with CIE xyY coordinates in the rows. If the sum `X+Y+Z==0`, `xy` are set to `NA`.
`LabfromXYZ()`	returns a numeric Nx3 matrix with CIE Lab coordinates in the rows
`LuvfromXYZ()`	returns a numeric Nx3 matrix with CIE Luv coordinates in the rows

In all cases, the rownames are copied from input to output. If the input is invalid, the functions return `NULL`.

References

Wikipedia. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

See Also

standardXYZ()

Examples

```
D65 = standardXYZ( 'D65' )

xyYfromXYZ( D65 )
##              x         y Y
##  D65 0.3127266 0.3290231 1     # probably not familiar

round( xyYfromXYZ(D65), 4 )
##           x     y Y
##  D65 0.3127 0.329 1        # probably more familiar


LabfromXYZ( 0.18*D65, D65 )   # 18% gray card
##             L a b
##  D65 49.49611 0 0          # exactly neutral, and L is about 50


D50 = standardXYZ( 'D50' )

LabfromXYZ( D50, D65 )
##        L        a        b
## D50 100 2.399554 17.65321  # D50 is far from neutral (yellowish) in D65 viewing environment
```

Planckian Loci *Planckian Loci - stored as Lookup Tables*

Description

RobertsonLocus	the table from Robertson, with 31 points from 0 to 600 mired
PrecisionLocus	a precomputed table, with 65 points from 0 to 1000 mired

Format

Both objects are data.frames with these columns

mired	the reciprocal temperature $10^6/T$
u	the u chromaticity, in 1960 CIE
v	the v chromaticity, in 1960 CIE

The PrecisionLocus data.frame has these additional columns:

up	the 1st derivative of u with respect to mired
vp	the 1st derivative of v with respect to mired
upp	the 2nd derivative of u with respect to mired
vpp	the 2nd derivative of v with respect to mired

Details

For RobertsonLocus, the values are taken from *Wyszecki & Stiles*. The lookup table on page 228 contains an error at 325 mired, which was corrected by Bruce Lindbloom (see **Source**).

For PrecisionLocus, the chromaticity values u and v are computed from first principles, from the famous equation for the Planckian radiator (with $c_2 = 1.4388 \times 10^{-2}$) and from the tabulated CIE 1931 standard observer color matching functions, by summing from 360 to 830nm. Let β denote the reciprocal temperature $10^6/T$. We think of u as a function $u(\beta)$. The column up is $u'(\beta)$, and upp is $u''(\beta)$. And similarly for v. The derivatives are computed from first principles, by summing the derivatives of the Planckian formula from 360 to 830nm. This includes the limiting case $\beta = 0$.

When this package is loaded (during .onLoad()), cubic splines are computed from RobertsonLocus, using stats::splinefun() with method="fmm"). And quintic splines are computed from PrecisionLocus. Both splines are C^2 continuous.

Source

http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_T.html

References

Robertson, A. R. Computation of correlated color temperature and distribution temperature. Journal of the Optical Society of America. 58. pp. 1528-1535. 1968.

Wyszecki, Günther and W. S. Stiles. **Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition.** John Wiley & Sons, 1982. Table 1(3.11). pp. 227-228.

See Also

CCTfromuv(), planckLocus()

Examples

```
RobertsonLocus[ 1:10, ]
##    mired       u       v
## 1      0 0.18006 0.26352
## 2     10 0.18066 0.26589
## 3     20 0.18133 0.26846
## 4     30 0.18208 0.27119
## 5     40 0.18293 0.27407
## 6     50 0.18388 0.27709
## 7     60 0.18494 0.28021
## 8     70 0.18611 0.28342
## 9     80 0.18740 0.28668
## 10    90 0.18880 0.28997
```

```
PrecisionLocus[ 1:10, ]
##    mired         u         v           up           vp          upp          vpp
## 1      0 0.1800644 0.2635212 5.540710e-05 0.0002276279 7.115677e-07 1.977793e-06
## 2     10 0.1806553 0.2658948 6.291429e-05 0.0002469232 7.900243e-07 1.873208e-06
## 3     20 0.1813253 0.2684554 7.120586e-05 0.0002649377 8.679532e-07 1.722425e-06
## 4     30 0.1820820 0.2711879 8.026143e-05 0.0002812384 9.423039e-07 1.531723e-06
## 5     40 0.1829329 0.2740733 9.002982e-05 0.0002954676 1.010028e-06 1.309700e-06
## 6     50 0.1838847 0.2770894 1.004307e-04 0.0003073613 1.068393e-06 1.066350e-06
## 7     60 0.1849432 0.2802122 1.113592e-04 0.0003167582 1.115240e-06 8.120582e-07
## 8     70 0.1861132 0.2834161 1.226923e-04 0.0003235990 1.149155e-06 5.566812e-07
## 9     80 0.1873980 0.2866757 1.342971e-04 0.0003279171 1.169532e-06 3.088345e-07
## 10    90 0.1887996 0.2899664 1.460383e-04 0.0003298241 1.176525e-06 7.543963e-08
```

standardXYZ   *Query the Standardized XYZ and xy values of Standard Illuminants and Whitepoints*

Description

In careful calcuations with standard illuminants and whitepoints, it is often helpful to have the ’official’ values of XYZ and xy, i.e. with the right number of decimal places.

Usage

```
standardXYZ( name )

standardxy( name )
```

Arguments

`name`	a subvector of `c('A','B','C','C.NBS','C.NTSC','C.JOSA','D50','D50.ICC', 'D55','D60','D65','D75','E','F2','F7','F11','ACES','DCI')`, which are the names of some standard illuminants and white points. Matching is partial and case-insensitive. `name` can also be `NULL` or `'*'` which means to return *all* available data.

Details

All XYZ values are taken from the ASTM publication in **References**, except `B` which is taken from *Wyszecki & Stiles* and `D50.ICC` which is taken from ICC publications.

xy values were taken from *CIE*, *BT.709*, *SMPTE EG 432-1*, and *TB-2018-001*. For D65 the values in *CIE* and *BT.709* disagree; the former has 5 digits and the latter has 4. We have selected the value in *BT.709* (page 3) since is far more commonly used. Three of the Illuminant C variants are rarely used and obsolete.

Value

`standardXYZ()` returns an Mx3 matrix where M is the length of `name`. But if `name` is `NULL` or `'*'`, M is the number of records available. Each row filled with the official XYZ, but if the illuminant name is not recognized or if there is no data, the row is all `NA`s. The output XYZ is normalized so that `Y=1`. The matrix `rownames` are set to the full illuminant names, and `colnames` to `c('X','Y','Z')`.

Similarly, `standardxy()` returns an Mx2 matrix with `colnames` set to `c('x','y')`.

Warning

The returned XYZs are normalized so that Y=1. In other color domains, it is common to normalize so that Y=100; in these cases be sure to multiply by 100.

Some illuminants have no standard XYZ available and some have no standard xy. In these cases, the rows are filled with NAs.

References

ASTM E 308 - 01. Standard Practice for Computing the Colors of Objects by Using the CIE System. 2001.

BT.709. Parameter values for the HDTV standards for production and international programme exchange. June 2015.

CIE 015:2004 - Colorimetry, 3rd edition. International Commission on Illumination (CIE). Vienna Austria. Technical Report. 2004.

Günther Wyszecki and W. S. Stiles. Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition. John Wiley & Sons, 1982. Table I(3.3.8) p. 769.

TB-2018-001. **Derivation of the ACES White Point CIE Chromaticity Coordinates**. The Academy of Motion Picture Arts and Sciences. Science and Technology Council. Academy Color Encoding System (ACES) Project. June 15, 2018.

SMPTE RP 431-2. **D-Cinema Quality - Reference Projector and Environment for the Display of DCDM in Review Rooms and Theaters**. 2011.

Examples

```
standardXYZ( c('a','d50','D50.ICC','D65') )
#                 X Y         Z
# A       1.0985000 1 0.3558500
# D50     0.9642200 1 0.8252100
# D50.ICC 0.9642029 1 0.8249054
# D65     0.9504700 1 1.0888300

standardxy( c('a','D65','D60','D60.ACES','E','F2') )
#                x       y
# A        0.44758 0.40745
# D65      0.31270 0.32900
# D60      0.32163 0.33774
# D60.ACES 0.32168 0.33767
# E        0.33333 0.33333
# F2            NA      NA
```

toPolar *Convert CIE Lab and Luv to Polar Form*

Description

Convert the Rectangular Form of CIE Lab and Luv to Polar Form

Usage

```
LCHabfromLab( Lab )
LCHuvfromLuv( Luv )
```

Arguments

`Lab`	a numeric Nx3 matrix with CIE Lab coordinates in the rows, or a vector that can be converted to such a matrix, by row.
`Luv`	a numeric Nx3 matrix with CIE Luv coordinates in the rows, or a vector that can be converted to such a matrix, by row.

Value

a numeric Nx3 matrix with LCH coordinates in the rows. The lightness L is simply copied from input to output. The chroma C corresponds to radius in polar coordinates, and the hue H corresponds to theta. H is in degrees, not radians. The rownames are copied from input to output.

References

Wikipedia. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

See Also

`LabfromLCHab()`, `LuvfromLCHuv()`

Examples

```
LCHabfromLab( c(50,0,0) )   #  a neutral gray
##        L Cab Hab
##  [1,] 50   0   0         # Hue is undefined, but set to 0

LCHabfromLab( c(50,0,20) )
##        L Cab Hab
##  [1,] 50  20  90         # 90 degrees, on yellow axis


LCHabfromLab( c(50,0,-20) )
##        L Cab Hab
##  [1,] 50  20 270         # 270 degrees, on blue axis
```

---

toXYZ *Convert other Color Spaces to XYZ*

Description

Convert other Color Spaces to XYZ

Usage

```
XYZfromxyY( xyY )
XYZfromLab( Lab, white )
XYZfromLuv( Luv, white )
```

Arguments

xyY	a numeric Nx3 matrix with CIE xyY coordinates in the rows, or a vector that can be converted to such a matrix, by row.
Lab	a numeric Nx3 matrix with CIE Lab coordinates in the rows, or a vector that can be converted to such a matrix, by row.
Luv	a numeric Nx3 matrix with CIE Luv coordinates in the rows, or a vector that can be converted to such a matrix, by row.
white	a numeric 3-vector giving the XYZ of reference white. white can also be a character string with the name of a standard illuminant, which is passed to standardXYZ() to get the XYZ.

Value

a numeric Nx3 matrix with XYZ coordinates in the rows. The rownames are copied from input to output.
In XYZfromxyY() if y==0 then X and Z are set to NA. Exception: Y==0 is treated as a special case (pure black); x and y are ignored, and XYZ are all set to 0.

References

Wikipedia. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

See Also

standardXYZ()

Examples

```
XYZfromxyY(c(0.310897, 0.306510, 74.613450))
##              X        Y        Z
##  [1,] 75.68137 74.61345 93.13427
```

```
XYZfromLab( c(50,2,-3), 'D50' )
##                X         Y         Z
##  [1,] 0.1813684 0.1841865 0.1643335
```

---

uvfrom *Convert from XYZ or xy to Uniform Chromaticity Spaces*

Description

Convert from XYZ or xy to Uniform Chromaticity Spaces

Usage

```
uvfromXYZ( XYZ, space=1976 )

uvfromxy( xy, space=1976 )
```

Arguments

`XYZ`	a numeric Nx3 matrix with CIE XYZ coordinates in the rows, or a vector that can be converted to such a matrix, by row.
`xy`	a numeric Nx2 matrix with CIE xy coordinates in the rows, or a vector that can be converted to such a matrix, by row.
`space`	the year of the space of output uv desired - valid spaces are 1976 and 1960. The default `space=1976` has largely superseded `space=1960`, but the latter is still used for computing Correlated Color Temperature; see `CCTfromuv()`. The 1976 space is used in the function `LuvfromXYZ()`. The 1960 coordinates are usually denoted by uv, and the 1976 coordinates by u'v'. The only difference is that $v' = (3/2)v$.

Value

a numeric Nx2 matrix with u'v' (or uv) coordinates in the rows.

For `uvfromXYZ()`, if $X + 15Y + 3Z \le 0$, uv are set to `NA`.

For `uvfromxy()`, if $-2x + 12y + 3 \le 0$, uv are set to `NA`.

The rownames are copied from input to output. If the input is invalid, the function returns `NULL`.

References

Wikipedia. CIE 1931 color space. https://en.wikipedia.org/wiki/CIE_1931_color_space

Wikipedia. CIE 1960 color space. https://en.wikipedia.org/wiki/CIE_1960_color_space

See Also

`LuvfromXYZ()`, `standardXYZ()`

Examples

```
# locate some standard illuminants on the 1976 UCS diagram
uvfromXYZ( standardXYZ( c('C','D50','D65','E')  ) )

##             u'        v'
##  C   0.2008921 0.4608838
##  D50 0.2091601 0.4880734
##  D65 0.1978398 0.4683363
##  E   0.2105263 0.4736842
```

Index