Package 'astro'

February 19, 2015

Type Package

Title Astronomy Functions, Tools and Routines

Version 1.2

Date 2014-09-08

Author Lee Kelvin

Maintainer Lee Kelvin <lee.kelvin@uibk.ac.at>

Description The astro package provides a series of functions, tools and routines in everyday use within astronomy. Broadly speaking, one may group these functions into 7 main areas, namely: cosmology, FITS file manipulation, the Sersic function, plotting, data manipulation, statistics and general convenience functions and scripting tools.

License GPL (>= 2)

LazyLoad yes

Depends MASS,plotrix

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-08 18:25:15

R topics documented:

`aaxis` *Add a Scientific Axis to a Plot*

Description

Adds an axis to the current plot.

Usage

```
aaxis(side = 1, at = NULL, labels = TRUE, majticks = TRUE,
      minticks = TRUE, nmaj = NULL, nmin = NULL, unlog = FALSE,
      format = NULL, digits = NULL, las = NULL, mgp = NULL,
      tcl = NULL, dexcl = 0.2, ...)
```

Arguments

`side`	an integer specifying which side of the plot the axis is to be drawn on. The axis is placed as follows: 1=below, 2=left, 3=above and 4=right
`at`	the points at which the major tick-marks are to be drawn
`labels`	should axis annotation be added
`majticks`	should major tick marks be included
`minticks`	should minor tick marks be included
`nmaj`	number of major tick marks
`nmin`	number of minor tick marks between major marks
`unlog`	unlog axis annotation when the data is logged (base 10)
`format`	format for axes labelling (see 'formatC')
`digits`	number of digits for axes labels
`las,mgp,tcl`	standard 'par' plotting parameters
`dexcl`	distance from major tick marks within which no minor tick labels should be plotted
`...`	additional arguments to be passed to 'axis'

Details

The mid-level function 'aaxis' (astro:axis) is a wrapper around the R function 'axis'. It provides significant additional features which trivially allow the creation of figures more suited for a scientific audience. Notably, 'aaxis' allows minor tick-marks and logged axes to be created with the minimum of effort.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
par("mar"=c(5.1,4.1,2.1,4.1))
aplot(1:1000, log10(1:1000), unlog="y", type="l", format="p", side=1:3, col="red", lwd=2)
aaxis(4, nmaj=4, nmin=9)
mtext(bquote(paste(log[10]," y ")), side=4, line=2.5)
label("bottomright", txt="astro:axis (aaxis)", cex=2, lwd=0, bgcol=NULL)
```

---

abox *Draw a Thick Box Around a Plot*

Description

Draws a thick box (5 overlapped boxes) around a plot, in order to combat any anti-aliasing effects that may be in place.

Usage

```
abox(...)
```

Arguments

...	arguments to be passed to 'box'

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

acb *Add a Colourbar to a Plot*

Description

Adds a colourbar (colour legend) to a plot.

Usage

```
acb(zlim = NULL, zlab = NULL, unlog = FALSE, cex = 1,
    zcol = NULL, cbpos = 4, cbsep = 0, cbspan = 0, cbinset = 1,
    cbx1 = NULL, cbx2 = NULL, cby1 = NULL, cby2 = NULL,
    cblegend = NULL, cex.cb = 1, zline = 2.5, ...)
```

Arguments

zlim	lower/upper limits of the z-axis
zlab	colourbar label
unlog	unlog logged data
cex	expansion factor
zcol	colourbar colour palette
cbpos	colourbar position (1/2/3/4)
cbsep	separation of colourbar from main plot
cbspan	width/height of colourbar
cbinset	size of colourbar inset parallel to plotting axis
cbx1,cbx2,cby1,cby2	manual placement of colourbar (xlower, xupper, ylower, yupper)
cblegend	colourbar annotation
cex.cb	colourbar expansion factor
zline	colourbar label line
...	additional arguments to be passed to 'color.legend'

Details

The mid-level function 'acb' (astro:colourbar) is a wrapper around the 'plotrix' function 'color.legend'. It allows trivial creation and placement of colourbars, able to plug-in more readily with figures created with 'aplot'. The main advantage for using 'acb' over that within 'aplot' is for those cases when multi-panelled figures are being created, and the colourbar serves several sub-plots simultaneously.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
layout(cbind(c(1,2),c(3,3)), widths=c(5,1))
par("mar"=c(0,0,0,1))
par("oma"=c(3.1,3.1,3.1,2.1))
aplot(rnorm(1000), rnorm(1000), rnorm(1000), pch=17, zlim=c(-1,1), xlim=c(-3,3),
ylim=c(-3,3), labels=2:3)
grid()
label("topleft", txt="astro:colourbar (acb)", cex=2, lwd=0, bgcol=NULL)
aplot(rnorm(1000), rnorm(1000), rnorm(1000), pch=16, zlim=c(-1,1), xlim=c(-3,3),
ylim=c(-3,3), labels=1:2)
grid()
acb(zlim=c(-1,1), zlab="z-axis label")
```

---

acol *Convert a named colour*

Description

Convert a named colour (or vector of colours) into their RGBA equivalent(s).

Usage

```
acol(col, alpha = 0)
```

Arguments

col	named colour (can be a vector)
alpha	alpha transparency value

Value

A vector equal in length to 'col' of RGBA equivalent colours.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

age *Age of the Universe*

Description

Calculates the age of the Universe

Usage

```
age(z = 1, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Gyr")
```

Arguments

z	redshift
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	output units [Gyr/s]

Value

Age of the Universe in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

age2z *Redshift of the Universe at a given age*

Description

Calculates the redshift of the Universe at a given age

Usage

```
age2z(t = 1, steps = 10, H = 70, M = 0.3, L = 1-M, K = 1-M-L,
      units = "Gyr")
```

Arguments

t	time
steps	number of accuracy steps
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	input units [Gyr/s]

Details

This is a very crude inverse variation of the 'age' function. As such, the output result should be used as a guide only.

Value

Redshift of the Universe at a given age with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

akde2d *Two-Dimensional Kernel Density Estimation (Astro)*

Description

Two-dimensional kernel density estimation with an axis-aligned bivariate normal kernel, evaluated on a square grid. The function 'akde2d' (astro:kde2d) is a wrapper around the 'kde2d' function within the 'MASS' package. This function adds additional output to that function.

Usage

```
akde2d(x, y, n = 25, lims = c(range(x), range(y)), levels = seq(0,0.9,by=0.1),
       ...)
```

Arguments

x	x coordinate of data
y	y coordinate of data
n	number of grid points in each direction. Can be scalar or a length-2 integer vector
lims	the limits of the rectangle covered by the grid as 'c(xl, xu, yl, yu)'
levels	output levels containing the given percentiles of the data
...	arguments to be passed to 'kde2d' in the 'MASS' package

Details

Characters in the string will be stripped 'outside in', from left-to-right in the order they are given in the argument. See examples below for more detail.

Value

A list of four components.

x,y	the x and y coordinates of the grid points, vectors of length 'n'
z	an 'n[1]' by 'n[2]' matrix of the estimated density: rows correspond to the value of 'x', columns to the value of 'y'
l	percentile levels containing given fractions of the data

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Venables, W. N. and Ripley, B. D. (2002) _Modern Applied Statistics with S._ Fourth edition. Springer.

See Also

The astronomy package: astro.

Examples

```
# See 'kde2d' for further examples.
```

---

angdist *Angular Diameter Distance*

Description

Calculates angular diameter distance

Usage

```
angdist(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Mpc")
```

Arguments

z	redshift
c	speed of light (m/s)
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	output units [Mpc/ly/m]

Value

Angular diameter distance in indicated units to the given redshift with the given cosmology. Note that this function is only valid for values of K >= 0.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

angsize *Angular Size Conversion*

Description

Converts between angular and physical size

Usage

```
angsize(z = 1, r = 1, inp = "arcsec", out = "kpc", c = 3E8, H = 70,
        M = 0.3, L = 1-M, K = 1-M-L)
```

Arguments

z	redshift
r	radius
inp	input units
out	output units
c	speed of light (m/s)
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature

Details

Units available for conversion are: 'deg', 'rad' or 'arcsec' <==> 'm', 'pc', 'kpc' or 'Mpc'.

Value

Converted size in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

aplot *Scientific X-Y Plotting*

Description

Generic function for plotting of R objects.

Usage

```
aplot(x, y = NULL, z = NULL, xlim = NULL, ylim = NULL, zlim = NULL,
      xlab = NULL, ylab = NULL, zlab = NULL, col = NULL, axes = TRUE,
      side = 1:4, labels = TRUE, majticks = TRUE, minticks = TRUE,
      nxmaj = NULL, nymaj = NULL, nxmin = NULL, nymin = NULL, xat = NULL,
      yat = NULL, log = "", unlog = FALSE, xformat = NULL, yformat = NULL,
      digits = 0, cex = 1, xlabpos = 1, ylabpos = 2, zcol = NULL,
      cb = FALSE, cbpos = 4, cbsep = 1.5, cbspan = 2, cbinset = 1,
      cbx1 = NULL, cbx2 = NULL, cby1 = NULL, cby2 = NULL, cblegend = NULL,
      cbsteps = 250, las = 0, mgp = c(2.5,0.5,0), tcl = 0.5, dexcl = 0.2,
      cex.axis = 1, cex.cb = 1, zline = mgp[1]+1, col.axes = "black",
      col.axis = "black", add = FALSE, type = NULL, bgcol = NULL, ...)
```

Arguments

x,y,z	the 'x', 'y' and 'z' arguments provide the x, y and z coordinates for the plot. Supplying the 'z' argument will colour each data point differently according to its z value.
xlim,ylim,zlim	the x, y and z limits of the plot in the form c(lower,upper)
xlab,ylab,zlab	the x, y and z axis labels
col	colour of the data points (will override z)
axes	plot axes
side	sides to plot axes [T/F or 1:4]
labels	sides to plot axes labels [T/F or 1:4]
majticks	plot major tick marks
minticks	plot minor tick marks
nxmaj	number of major tick marks on the x-axis
nymaj	number of major tick marks on the y-axis
nxmin	number of minor tick marks between major ticks (x)
nymin	number of minor tick marks between major ticks (y)
xat	position of x-axes major tick marks
yat	position of y-axes major tick marks
log	logged axes
unlog	unlog axes plotting logged data

xformat	format for x-axes labelling (see 'formatC')
yformat	format for y-axes labelling (see 'formatC')
digits	number of digits for axes labels
cex	expansion factor
xlabpos	label position (x)
ylabpos	label position (y)
zcol	z-axis colour palette
cb	plot colourbar
cbpos	position of colourbar
cbsep	separation of colourbar from plot
cbspan	width/height of colourbar
cbinset	size of inset of colourbar on axis parallel to plotting side
cbx1,cbx2,cby1,cby2	manual colourbar positioning (xlower, xupper, ylower, yupper
cblegend	colourbar legend
cbsteps	colourbar resolution
las,mgp,tcl	standard 'par' plotting parameters
dexcl	distance from major tick marks within which no minor tick labels should be plotted
cex.axis	expansion factor for axis annotation
cex.cb	expansion factor for colourbar annotation
zline	colourbar label line
col.axes	colour of axes
col.axis	colour of axes annotation
add	add the plot to an existing plot
type	plot type
bgcol	background colour
...	additional arguments to be passed to 'plot'

Details

The top-level function 'aplot' (astro:plot) is a wrapper around the R function 'plot'. It provides significant additional features which trivially allow the creation of figures more suited for a scientific audience. Notably, 'aplot' allows z-axis information to be displayed through the use of colourbars and provides improved axes (including minor-tick marks) through the use of the 'aaxis' function.

Note

Colourbar features are provided by the 'color.legend' function within the 'plotrix' package (author: Jim Lemon). 'aplot' provides a wrapper around this function, therefore, in order for colourbar features to function correctly, the 'plotrix' package must be installed.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
# example #1
layout(1)
par("mar"=c(5.1,4.1,2.1,2.1))
par("oma"=c(0,0,0,0))
aplot(sin, xlim=c(0,2*pi), ylim=c(-1.1,1.1), bgcol="lightgoldenrodyellow")
abline(h=0, col="grey75")
label("top", txt="Sine Function", lwd=0, bgcol="grey25", col="white")
label("bottomleft", txt="astro:label (label)", cex=2, lwd=0, bgcol=NULL)

# example #2
layout(1)
par("mar"=c(5.1,4.1,2.1,4.1))
par("oma"=c(0,0,0,0))
aplot(1:1000, log10(1:1000), unlog="y", type="l", yformat="p", side=1:3,
col="red", lwd=2)
aaxis(4, nmaj=4, nmin=9)
mtext(bquote(paste(log[10]," y ")), side=4, line=2.5)
label("bottomright", txt="astro:axis (aaxis)", cex=2, lwd=0, bgcol=NULL)

# example #3
layout(cbind(c(1,2),c(3,3)), widths=c(5,1))
par("mar"=c(0,0,0,1))
par("oma"=c(3.1,3.1,3.1,2.1))
aplot(rnorm(1000), rnorm(1000), rnorm(1000), pch=17, zlim=c(-1,1), xlim=c(-3,3),
ylim=c(-3,3), labels=2:3)
grid()
label("topleft", txt="astro:colourbar (acb)", cex=2, lwd=0, bgcol=NULL)
aplot(rnorm(1000), rnorm(1000), rnorm(1000), pch=16, zlim=c(-1,1), xlim=c(-3,3),
ylim=c(-3,3), labels=1:2)
grid()
acb(zlim=c(-1,1), zlab="z-axis label")

# example #4
layout(1)
par("mar"=c(5.1,4.1,4.1,5.1))
par("oma"=c(0,0,0,0))
aplot(rnorm(1000), rnorm(1000), rnorm(1000), cb=TRUE, zlim=c(-1,1), pch=16,
xlab="x-axis label", ylab="y-axis label", zlab="z-axis label", bgcol="grey95")
label("topleft", txt="astro:plot (aplot)", cex=2, lwd=0, bgcol=NULL)
```

aqbeta *Astro:Beta Distribution (quantile)*

Description

An astro wrapper around the 'qbeta' function. Useful for calculating robust upper and lower error boundaries for a given data set.

Usage

```
aqbeta(k, n, s = c(-1,1), p = NA, corr = TRUE, ...)
```

Arguments

k	number of successes
n	total number of trials
s	sigma values required
p	probability values required (see details)
corr	apply a one-sided correction for extreme values (k=0 / k=n)
...	arguments to be passed to 'qbeta'

Details

When 'p' is equal to <NA> (default), sigma confidence intervals are calculated according to the value of 's'. If a value of 'p' is given, probabilities of 'p' are calculated instead.

Author(s)

Ewan Cameron <dr.ewan.cameron@gmail.com>

Aaron Robotham <aaron.robotham@icrar.org>

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Cameron, E., 2011, Publications of the Astronomical Society of Australia (PASA), 28, 128

See Also

The astronomy package: astro.

astro *Astronomy Functions, Tools and Routines*

Description

A collection of commonly used astronomy functions, tools and routines.

Details

The astro package provides a series of functions, tools and routines in everyday use within astronomy.

Broadly speaking, one may group these functions into 7 main areas, namely: cosmology, FITS file manipulation, the Sersic function, plotting, data manipulation, statistics and general convenience functions and scripting tools. An overview of these sub-packages and their functions is as follows:

Cosmology:

Cosmology Calculator: coscalc,
Comoving distance (line of sight): comovdist.los,
Comoving distance (transverse): comovdist.trans,
Angular diameter distance: angdist,
Luminosity distance: lumdist,
Comoving volume: comovvol,
Lookback time: lookback,
Redshift at a given Lookback time: lookback2z,
Age of the Universe: age,
Redshift at a given age: age2z,
Angular size conversion: angsize

FITS:

Read FITS files (images/tables & headers): read.fits,
Read FITS header: read.fitshdr,
Read FITS key: read.fitskey,
Read FITS image: read.fitsim,
Read FITS table: read.fitstab,
Get FITS keyword value: get.fitskey,
Put FITS keyword value: put.fitskey,
Write FITS files: write.fits,
Write FITS header: write.fitshdr,
Write FITS key: write.fitskey,
Plot FITS images: plotfits,
Write large binary files: writeBin64,
Strip leading/trailing characters: strip

Sersic:
The Sersic function: sersic,
Convert between Sersic radii: convrad,
Convert between Sersic surface brightnesses: convmu,
Convert half-light radius to scalelength: re2h,
Convert scalelength to half-light radius: h2re,
Calculate a combined half-light radius: combrad,
Central Concentration: concen,
The Petrosian function: petro,
Petrosian index: petroindex,
Petrosian radius: petrorad,
The Kron function: kron,
Kron radius: kronrad

Plotting:
Astro plot: aplot,
Astro axis: aaxis,
Astro colourbar: acb,
Astro box: abox,
Astro alpha-transparency colour conversion: acol,
Ellipse: ellipse,
Plot labelling: label,
Plot shading: shade,
Shadowed text: shadowtext,
Cardinal points: cardinal,
Scale marks: scalemark

Data:
Astro kernel density estimator: akde2d,
The Gaussian Function: gauss,
The Schechter Function: schechter,
The Luminosity Density: lumdens,
Calculate Schechter Function Binned Variables: schechter.bin,
Fit to the Schechter Function: schechter.fit,
Error ellipses for a Schechter fit: schechter.ellipse

Statistics:
Chi-Squared Probability Density Function: chipdf,
Chi-Squared Cumulative Distribution Function: chicdf,
Chi-Squared P-Value: chipval,
Calculate the chi-squared statistic for a given p-value: chipval,
The incomplete gamma function: igamma,
Astro qbeta function: aqbeta,
Sigma-clipped mean: scmean

Miscellaneous:
Absolute Magnitude of the Sun: solar,
Central Angle: cenang,
Human-readable time strings: nicetime,
For-loop ETA: eta

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

`cardinal` *Add Cardinal Points to a Plot*

Description

Adds the four cardinal points to a figure, marking north and east.

Usage

```
cardinal(rotate = 0, pos = "bottomright", inset = 0.5,
         len = 2, gap = 1, col = "white", linecol = col,
         bg = "grey25", lwd = 1, invert = TRUE, south = TRUE,
         west = TRUE, textrot = TRUE, ...)
```

Arguments

`rotate`	rotate cardinal cross clockwise from 'up' (degrees)
`pos`	position of cross (bottomleft/topleft/topright/bottomright)
`inset`	inset cross from edge
`len`	length of cross arrows
`gap`	size of gap between arrowhead and text
`col`	colour of arrows/text
`linecol`	colour of cardinal cross
`bg`	shading colour
`lwd`	line width
`invert`	invert east-west
`south`	show the southern prong
`west`	show the western prong
`textrot`	rotate the text along with the cross
`...`	arguments to be passed to 'shadowtext'

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

cenang *Central Angle*

Description

Calculates the angle between two points on a sphere

Usage

```
cenang(a1, d1, a2, d2, units = "deg", method = "vincenty")
```

Arguments

a1	right ascension of point 1
d1	declination of point 1
a2	right ascension of point 2
d2	declination of point 2
units	input/output units [deg/rad]
method	angle calculation method (see below)

Details

The central angle describes the angle, from the origin, between two points lying on the surface of a sphere (e.g., the celestial sphere). Three commonly used methods employed in its calculation are: the Spherical Law of Cosines (sloc); the Haversine Formula (haversine), and; the Vincenty Formula (vincenty). The Spherical Law of Cosines suffers from severe rounding errors for small angles (theta < 1E-5). The Haversine Formula generally works well at all angles, but suffers from rounding errors for antipodal points. The Vincenty Formula is accurate for all angles, and is recommended. The three methods may be chosen by specifying 'sloc', 'haversine' or 'vincenty', or by their respective first letters: s, h or v.

Value

The central angle: the angle between two points lying on the surface of a sphere, with reference at the origin/centroid.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

http://en.wikipedia.org/wiki/Great-circle_distance

See Also

The astronomy package: astro.

---

chicdf *The Cumulative Distribution Function for the Chi-Squared Distribution*

Description

Calculates the CDF for the chi-squared distribution.

Usage

```
chicdf(X, k)
```

Arguments

X	a vector of input chi-squared values
k	the number of degrees of freedom

Details

The chi-squared distribution is the sum of the squares of k independent standard normal random variables, where k represents the number of degrees of freedom. Typically, k is estimated using the relation k = N - n, where N represents the number of data points (observations) in your data set, and n represents the number of fitted parameters in your model.

This function returns the cumulative distribution for a vector of given chi-squared values with an associated number k degrees of freedom.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
X = seq(0,8,len=1000)
kvals = c(1,2,3,4,6,9)
cols = c("yellow","green","turquoise","blue","purple","red")
aplot(NA, type="n", xlab=bquote(chi^2),
ylab=bquote(paste(F[k],"(",chi^2,")",sep="")), xlim=c(0,8),
ylim=c(0,1), main="Chi-Squared Cumulative Distribution Function")
grid(lty=1, col="grey90")
for(i in 1:length(kvals)){
    lines(X, chicdf(X=X, k=kvals[i]), lwd=3, col=cols[i])
```

```
}
legend("bottomright", legend=paste("k =",kvals), lwd=3, col=cols,
bty="o", bg=acol("white",alpha=0.7), inset=0.04, text.font=3,
box.col=NA)
```

---

chipdf *The Probability Density Function for the Chi-Squared Distribution*

Description

Calculates the PDF for the chi-squared distribution.

Usage

```
chipdf(X, k)
```

Arguments

X	a vector of input chi-squared values
k	the number of degrees of freedom

Details

The chi-squared distribution is the sum of the squares of k independent standard normal random variables, where k represents the number of degrees of freedom. Typically, k is estimated using the relation k = N - n, where N represents the number of data points (observations) in your data set, and n represents the number of fitted parameters in your model.

This function returns the probability density function for a vector of given chi-squared values with an associated number k degrees of freedom.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
X = seq(0,8,len=1000)
kvals = c(1,2,3,4,6,9)
cols = c("yellow","green","turquoise","blue","purple","red")
aplot(NA, type="n", xlab=bquote(chi^2),
ylab=bquote(paste(f[k],"(",chi^2,")",sep="")), xlim=c(0,8),
ylim=c(0,0.5), main="Chi-Squared Probability Density Function")
grid(lty=1, col="grey90")
for(i in 1:length(kvals)){
```

```
    lines(X, chipdf(X=X, k=kvals[i]), lwd=3, col=cols[i])
}
legend("topright", legend=paste("k =",kvals), lwd=3, col=cols,
bty="o", bg=acol("white",alpha=0.7), inset=0.04, text.font=3,
box.col=NA)
```

---

chipval *The P-Value for the Chi-Squared Distribution*

Description

Calculates the p-value for a given chi-squared statistic.

Usage

```
chipval(X, k)
```

Arguments

X	a vector of input chi-squared values
k	the number of degrees of freedom

Details

The chi-squared distribution is the sum of the squares of k independent standard normal random variables, where k represents the number of degrees of freedom. Typically, k is estimated using the relation k = N - n, where N represents the number of data points (observations) in your data set, and n represents the number of fitted parameters in your model.

The p-value represents the estimated probability of rejecting the null hypothesis. Here we assume the null hypothesis to be that the sample follows a chi-squared distribution as parameterised by the number of degrees of freedom k. Typically, if p >= 0.05, the data appear to be consistent with the null hypothesis, and if p < 0.05, there is significant evidence against the null hypothesis in favour of the alternative. These limits, albeit rather arbitrary, are nevertheless consistently used in the literature. Often, the p-value is (incorrectly) interpreted as the probability that the null hypothesis is true.

This function returns the p-value for a vector of given chi-squared values with an associated number k degrees of freedom.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
X = seq(0,20,by=1)
kvals = c(1,2,3,4,6,9)
par("mar"=c(1,1,3,1))
plot(NA, type="n", xlim=c(-0.5,6.5), ylim=c(21.5,-0.5), axes=FALSE,
xlab="", ylab="", xaxs="i", yaxs="i", main="P-Value Lookup Table")
text(x=0, y=0, labels=bquote(chi^2))
for(i in 1:length(kvals)){
    text(x=i, y=0, labels=paste("k =",kvals[i]))
}
abline(v=seq(0.5,length(kvals)-0.5,by=1), col="grey90", lwd=3)
for(i in 1:length(X)){
    text(x=0, y=i, labels=paste(X[i]))
}
abline(h=seq(0.5,length(X)-0.5,by=1), col="grey90", lwd=3)
cols = acol(colorRampPalette(c("green","red"))(100),alpha=0.5)
for(i in 1:length(X)){
    for(j in 1:length(kvals)){
        p = chipval(X=X[i], k=kvals[j])
        col = ((p^0.35)*(length(cols)-1))+1
        rect(xleft=j-0.5, xright=j+0.5, ybottom=i-0.5, ytop=i+0.5,
        col=cols[col], border=NA)
        text(x=j, y=i, labels=formatC(p,format="f",digits=3))
    }
}
par("mar"=c(5.1,4.1,4.1,2.1))
```

---

combrad *Combine Sersic Functions*

Description

Combine two or more Sersic functions and calculate the new implied total half-light radius

Usage

```
combrad(mag, re, n)
```

Arguments

mag	total magnitudes (lists of vectors)
re	half-light radii (lists of vectors)
n	Sersic indices (lists of vectors)

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

`comovdist.los` *Co-moving Distance (Line of Sight)*

Description

Calculates line-of-sight comoving distance

Usage

```
comovdist.los(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Mpc")
```

Arguments

`z`	redshift
`c`	speed of light (m/s)
`H`	Hubble constant (km/s/Mpc)
`M`	Omega M - matter
`L`	Omega L - energy
`K`	Omega K - curvature
`units`	output units [Mpc/ly/m]

Value

Line-of-sight comoving distance in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

`comovdist.trans` *Co-moving Distance (Transverse)*

Description

Calculates transverse comoving distance

Usage

```
comovdist.trans(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Mpc")
```

Arguments

`z`	redshift
`c`	speed of light (m/s)
`H`	Hubble constant (km/s/Mpc)
`M`	Omega M - matter
`L`	Omega L - energy
`K`	Omega K - curvature
`units`	output units [Mpc/ly/m]

Value

Transverse comoving distance in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: `astro`.

comovvol *Co-moving Volume*

Description

Calculates comoving volume

Usage

```
comovvol(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Gpc3")
```

Arguments

z	redshift
c	speed of light (m/s)
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	output units [Gpc3/Mpc3/ly3/m3]

Value

Comoving volume in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

concen *Central Concentration*

Description

Calculates the central concentration based on input Sersic parameters.

Usage

```
concen(a, n)
```

Arguments

a	some inner radius (0 < a < 1)
n	Sersic index

Details

This function calculates the central concentration of a galaxy dependent upon its Sersic index. In order for this calculation to succeed, the outer radial extent of the galaxy must be normalised to 1, and some arbitrary inner radius in the range 0 < a < 1 (for example, a = 1/3) must be chosen.

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

convmu *Convert Between Sersic Surface Brightnesses*

Description

Convert between Sersic surface brightnesses at different radii.

Usage

```
convmu(r, mu, n, re = 1, rmu = re)
```

Arguments

r	desired radius
mu	input surface brightness
n	Sersic index
re	half-light radius
rmu	radius at the input surface brightness

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

---

convrad *Convert Between Sersic Radii*

Description

Convert between Sersic radii defined by different flux fractions.

Usage

```
convrad(n, f = 0.9, r = 1, fr = 0.5)
```

Arguments

n	Sersic index
f	flux fraction at new radius
r	reference radius
fr	flux fraction at reference radius

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

`coscalc` *Calculates Cosmological Parameters*

Description

Calculates several commonly used cosmological parameters using a given cosmology.

Usage

```
coscalc(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M,
        K = 1-M-L, dunit = "Mpc", vunit = "Gpc3", tunit = "Gyr",
        r = 1, inp = "arcsec", out = "kpc")
```

Arguments

`z`	redshift
`c`	speed of light (m/s)
`H`	Hubble constant (km/s/Mpc)
`M`	Omega M - matter
`L`	Omega L - energy
`K`	Omega K - curvature
`dunit`	outout distance units [Mpc/ly/m]
`vunit`	output volume units [Gpc3/ly3/m3]
`tunit`	output time units [Gyr/s]
`r`	radius/size
`inp`	input angular size units (see details)
`out`	output angular size units (see details)

Details

'coscalc' brings together several commonly used cosmological distance/volume/time measurements as given by several other functions in the 'astro' package.

Units available for angular size conversion are: 'deg', 'rad' or 'arcsec' <==> 'm', 'pc', 'kpc' or 'Mpc'.

Value

`z`	redshift
`codist.los`	comoving distance (line-of-sight)
`codist.trans`	comoving distance (transverse)
`angdist`	angular diameter distance
`lumdist`	luminosity distance

covol	comoving volume
lookback	lookback time
age	age of Universe at 'z'
angsize	angular size conversion

Author(s)

Lee Kelvin, Aaron Robotham

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

---

ellipse *Calculates Ellipse Coordinates*

Description

Calculates x,y cartesian position coordinates of an ellipse.

Usage

```
ellipse(xcen = 0, ycen = 0, a = 10, b = 5, e = 1-b/a, pa = 0)
```

Arguments

xcen	origin (x)
ycen	origin (y)
a	semi-major axis
b	semi-minor axis
e	ellipticity (1 - b/a)
pa	position angle (right=0, up=90)

Details

Note that 'b' is redundant if values of 'e' are given.

Value

x	ellipse coordinates along the x-axis
y	ellipse coordinates along the y-axis

Author(s)

Menaechmus, Euclid, Apollonius, Lee Kelvin

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

eta *Calculate ETA of a For-Loop*

Description

Calculates an ETA of a for-loop completing the loop, based on how many loops are left, and the start time of the first loop.

Usage

```
eta(i, total, start)
```

Arguments

i	current element
total	total number of elements
start	start processor time (seconds)

Details

The start time is best given by 'proc.time()[3]'.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

gauss *The Gaussian Function*

Description

This function calculates the Gaussian function for an input data range.

Usage

```
gauss(x, mean = 0, sigma = 1, norm = 1, bw = 0.1, ftype = "lin", ...)
```

Arguments

x	input values (typically a smooth data range)
mean	the mean(s) of the Gaussian function
sigma	the 1-sigma value
norm	the normalisation(s) of the Gaussian function
bw	integration bin width sizes
ftype	type of input data [lin/log/ln]
...	additional arguments to be passed to 'integrate'

Value

A vector of length equal to the length of data representing the number density *per dex* at each input data point. Note: to convert the final number densities into their original bin-width (e.g., per 0.5 dex) multiply the output of this function by the bin-width.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

get.fitskey *Get FITS Keyword Value From Header*

Description

A utility function to allow easy extraction of a FITS header value from an already loaded FITS header object.

Usage

```
get.fitskey(key, hdr)
```

Arguments

key	header keyword (may be a vector)
hdr	FITS header object

Value

A vector of data equal in length to the input key request. NA is returned where no keys have been found.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

h2re *Scalelength to Half-Light Radius*

Description

Convert scalelength to half-light radius.

Usage

```
h2re(n, h = 1)
```

Arguments

n	Sersic index
h	scalelength

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

`igamma` *The Incomplete Gamma Function*

Description

Calculates and returns the incomplete gamma function (specifically, the lower incomplete gamma function).

Usage

```
igamma(x, s)
```

Arguments

`x`	upper limit of integration
`s`	shape parameter

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

kron *Kron Function*

Description

Calculates Kron parameters based on input Sersic parameters.

Usage

```
kron(mag, n, e = 0, rk = 2.5, r=1e10, re = 1)
```

Arguments

mag	total magnitude
n	Sersic index
e	ellipticity (1 - b/a)
rk	integrate out to this many multiples of the Kron radius
r	radius at which to evaluate the Kron radius
re	half-light radius

Details

The Kron radius is defined as the first moment of the surface brightness light profile. Kron (1980) argued that apertures of multiple values of the Kron radius, Rk (typically, Rk = 2 or Rk = 2.5) would contain a sufficient amount of the total flux of the galaxy to be a useful measure of total flux. This is only true if one is able to evaluate the Kron radius at very large radii away from the galaxy centre (using very deep imaging data). If this is not possible (which typically it is not) the Kron radius will be an underestimate, and consequently, so will the Kron magnitude. This problem becomes most acute for those galaxies with high Sersic indices found within shallow imaging data.

Value

mag	magnitude within r
magdiff	difference between total magnitude and magnitude within r
mu	surface brightness at r
muavg	average surface brightness within r
inten	intensity at r
lum	luminosity within r
lumtot	total luminosity
lumfrac	fraction of total luminosity contained within r

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

---

kronrad *Kron Radius*

Description

Calculates the Kron radius based on input Sersic parameters.

Usage

```
kronrad(n, r = 1e10, re = 1)
```

Arguments

n	Sersic index
r	radius at which to evaluate the Kron radius
re	half-light radius

Details

The Kron radius is defined as the first moment of the surface brightness light profile. Kron (1980) argued that apertures of multiple values of the Kron radius, Rk (typically, Rk = 2 or Rk = 2.5) would contain a sufficient amount of the total flux of the galaxy to be a useful measure of total flux. This is only true if one is able to evaluate the Kron radius at very large radii away from the galaxy centre (using very deep imaging data). If this is not possible (which typically it is not) the Kron radius will be an underestimate, and consequently, so will the Kron magnitude. This problem becomes most acute for those galaxies with high Sersic indices found within shallow imaging data.

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

label *Add A Label To A Plot*

Description

Does exactly what it says on the tin.

Usage

```
label(pos = "topleft", lab = "label", txt = NULL, inset = 0.1,
      whitespace = 0.08, col = "black", bgcol = "white", bty = "n",
      bordercol = "black", lwd = 1, cex = 1, align = "center")
```

Arguments

pos	position of label (topleft, top, topright, ...)
lab	contents of the label
txt	see 'lab' (backwards compatibility)
inset	label box inset
whitespace	separation between text and box edge
col	colour
bgcol	background colour
bty	box type (b or n)
bordercol	border colour
lwd	line width of the border
cex	expansion factor
align	text align within box

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
par("mar"=c(5.1,4.1,2.1,2.1))
aplot(sin, xlim=c(0,2*pi), ylim=c(-1.1,1.1), bgcol="lightgoldenrodyellow")
abline(h=0, col="grey75")
label("top", txt="Sine Function", lwd=0, bgcol="grey25", col="white")
label("bottomleft", txt="astro:label (label)", cex=2, lwd=0, bgcol=NULL)
```

`lookback` *Lookback Time*

Description

Calculates the lookback time (light travel time)

Usage

```
lookback(z = 1, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Gyr")
```

Arguments

`z`	redshift
`H`	Hubble constant (km/s/Mpc)
`M`	Omega M - matter
`L`	Omega L - energy
`K`	Omega K - curvature
`units`	output units [Gyr/s]

Value

Lookback time in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: `astro`.

lookback2z *Redshift at a given Lookback Time*

Description

Calculates the lookback time (light travel time)

Usage

```
lookback2z(t = 1, steps = 10, H = 70, M = 0.3, L = 1-M, K = 1-M-L,
           units = "Gyr")
```

Arguments

t	time
steps	number of accuracy steps
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	input units [Gyr/s]

Details

This is a very crude inverse variation of the 'lookback' function. As such, the output result should be used as a guide only.

Value

Redshift at a given lookback time with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

lumdens *Luminosity Density*

Description

Calculate the luminosity density from a given set of input (Schechter) luminosity function inputs.

Usage

```
lumdens(knee, slope, norm, msun = solar("r"), mag = TRUE, log = FALSE)
```

Arguments

knee	The knee(s) of the luminosity distribution
slope	The faint-end slope(s) of the luminosity distribution
norm	The normalisation(s) of the luminosity distribution
msun	The absolute magnitude of the sun
mag	Are the input knee values in magnitudes?
log	Are the input knee values logged?

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

lumdist *Luminosity Distance*

Description

Calculates luminosity distance

Usage

```
lumdist(z = 1, c = 3E8, H = 70, M = 0.3, L = 1-M, K = 1-M-L, units = "Mpc")
```

Arguments

z	redshift
c	speed of light (m/s)
H	Hubble constant (km/s/Mpc)
M	Omega M - matter
L	Omega L - energy
K	Omega K - curvature
units	output units [Mpc/ly/m]

Value

Luminosity distance in indicated units to the given redshift with the given cosmology.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Hogg D. W., 1999, ArXiv Astrophysics e-prints, arXiv:astro-ph/9905116

See Also

The astronomy package: astro.

---

nicetime *Convert Seconds into a Human Readable Time*

Description

Converts an input raw time value, such as that given by 'proc.time', and converts it into a human readable time format.

Usage

```
nicetime(seconds)
```

Arguments

seconds	number of seconds

Value

The output character string will display the input time (in seconds) in units of days, hours, minutes and seconds, as appropriate.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

p2chi *The Chi-Squared Statistic for a Given P-Value*

Description

Calculates the chi-squared statistic for a given p-value.

Usage

```
p2chi(p, k, steps = 30, chimax = 1E31)
```

Arguments

p	a vector of input p-values
k	the number of degrees of freedom
steps	number of accuracy steps
chimax	the upper chi-squared search limit

Details

The chi-squared distribution is the sum of the squares of k independent standard normal random variables, where k represents the number of degrees of freedom. Typically, k is estimated using the relation k = N - n, where N represents the number of data points (observations) in your data set, and n represents the number of fitted parameters in your model.

The p-value represents the estimated probability of rejecting the null hypothesis. Here we assume the null hypothesis to be that the sample follows a chi-squared distribution as parameterised by the number of degrees of freedom k. Typically, if p >= 0.05, the data appear to be consistent with the null hypothesis, and if p < 0.05, there is significant evidence against the null hypothesis in favour of the alternative. These limits, albeit rather arbitrary, are nevertheless consistently used in the literature. Often, the p-value is (incorrectly) interpreted as the probability that the null hypothesis is true.

This function returns the chi-squared statistic for a vector of given p-values with an associated number k degrees of freedom. Increasing the number of accuracy steps will increase the final accuracy, however; steps = 30 (the default) should be sufficient for most tasks.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
X = seq(0,20,len=1000)
kvals = c(1,2,3,4,6,9)
cols = c("yellow","green","turquoise","blue","purple","red")
for(i in 1:length(kvals)){
    psig = p2chi(p=0.05, k=kvals[i])
    pgood = seq(0,psig,len=100)
    pbad = seq(psig,max(X),len=100)
    aplot(X, chipdf(X=X, k=kvals[i]), type="n", xlab=bquote(chi^2),
    ylab=bquote(paste(f[k],"(",chi^2,")",sep="")),
    ylim=c(0,min(0.5,max(chipdf(X=X, k=kvals[i])))),
    main="Chi-Squared Probability Density Function")
    polygon(x=c(pgood,rev(pgood)),
    y=c(chipdf(pgood, k=kvals[i]),rep(0,len=length(pgood))),
    col=acol("yellow",alpha=0.5), border=NA)
    polygon(x=c(pbad,rev(pbad)),
    y=c(chipdf(pbad, k=kvals[i]),rep(0,len=length(pbad))),
    col=acol("turquoise",alpha=0.5), border=NA)
    lines(X, chipdf(X=X, k=kvals[i]), lwd=3, col=cols[i])
    legend("topright",
    legend=c("p >= 0.05 (data consistent with null hypothesis)",
    "p < 0.05 (reject null hypothesis)"), bty="o",
    bg=acol("white",alpha=0.7), inset=0.04, text.font=3, box.col=NA,
    fill=acol(c("yellow","turquoise"),alpha=0.5), border=NA)
    legend("right", legend=paste("k =",kvals[i]), lwd=3, col=cols[i],
    bty="o", bg=acol("white",alpha=0.7), inset=0.04, text.font=3,
    box.col=NA)
}
```

---

petro *Petrosian Function*

Description

Calculates Petrosian parameters based on input Sersic parameters.

Usage

```
petro(mag, n, e = 0, rp = 3, i = 0.5)
```

Arguments

mag	total magnitude
n	Sersic index
e	ellipticity (1 - b/a)

rp	integrate out to this many multiples of the Petrosian radius
i	use this value of 1/petrosian index

Details

The Petrosian function describes the ratio between the average intensity within some projected radius and the intensity at that radius. The value of this ratio is known as the Petrosian index. Typically, a fixed value of 1/Petrosian index, i (usually i = 0.2 or i = 0.5) is chosen in order to define the Petrosian radius. The Petrosian magnitude is then defined as the magnitude lying within a given multiple of the Petrosian radius, Rp (typically, Rp = 2 or Rp = 3 for i = 0.2 and i = 0.5, respectively).

Value

mag	magnitude within r
magdiff	difference between total magnitude and magnitude within r
mu	surface brightness at r
muavg	average surface brightness within r
inten	intensity at r
lum	luminosity within r
lumtot	total luminosity
lumfrac	fraction of total luminosity contained within r

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

---

petroindex *Petrosian Index*

Description

Calculates the Petrosian index based on input Sersic parameters.

Usage

```
petroindex(r, n, re = 1)
```

Arguments

r	radius
n	Sersic index
re	half-light radius

Details

The Petrosian function describes the ratio between the average intensity within some projected radius and the intensity at that radius. The value of this ratio is known as the Petrosian index. Typically, a fixed value of 1/Petrosian index, i (usually i = 0.2 or i = 0.5) is chosen in order to define the Petrosian radius. The Petrosian magnitude is then defined as the magnitude lying within a given multiple of the Petrosian radius, Rp (typically, Rp = 2 or Rp = 3 for i = 0.2 and i = 0.5, respectively).

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

---

petrorad *Petrosian Radius*

Description

Calculates the Petrosian radius based on input Sersic parameters.

Usage

```
petrorad(n, i = 0.5, re = 1)
```

Arguments

n	Sersic index
i	use this value of 1/petrosian index
re	half-light radius

Details

The Petrosian function describes the ratio between the average intensity within some projected radius and the intensity at that radius. The value of this ratio is known as the Petrosian index. Typically, a fixed value of 1/Petrosian index, i (usually i = 0.2 or i = 0.5) is chosen in order to define the Petrosian radius. The Petrosian magnitude is then defined as the magnitude lying within a given multiple of the Petrosian radius, Rp (typically, Rp = 2 or Rp = 3 for i = 0.2 and i = 0.5, respectively).

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: `astro`.

`plotfits` *Plot FITS images*

Description

This function allows a FITS image to be converted into a PNG/X11 image, or output as a data array for use within a function. 3 colour RGB images may also be constructed by using multiple file inputs.

Usage

```
plotfits(input, hdu = 1, func = "atan", slide = c(0,0,0),
         scale = c(500,300,100), locut = 0, hicut = pi/2, invert = FALSE,
         method = 1, type = "x11", width = 5, height = 5, units = "in",
         res = 300, cen = c(NA,NA), xdim = NA, ydim = NA,
         file = "image.png")
```

Arguments

`input`	input file(s)
`hdu`	input hdu
`func`	scaling function for plot [lin/log/atan]
`slide`	offset counts from the origin by a given value
`scale`	scaling value for each image
`locut`	lower cut (black)
`hicut`	upper cut (white)

invert	invert the output greyscale/colours
method	counts -> image conversion method [1/2]
type	output type [dat/png/eps/pdf/x11 or bitmap types]
width	output width
height	output height
units	output units
res	output resolution
cen	centre of image
xdim	x width (pixels)
ydim	y width (pixels)
file	output file name

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

`put.fitskey` *Put FITS Keyword Value Into Header*

Description

A utility function to allow easy input of a FITS header value into an already loaded FITS header object.

Usage

```
put.fitskey(key, value, hdr)
```

Arguments

key	header keyword (may be a vector)
value	header value (may be a vector)
hdr	FITS header object

Value

The updated header object.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

re2h *Half-Light Radius to Scalelength*

Description

Convert half-light radius to scalelength.

Usage

```
re2h(n, re = 1)
```

Arguments

n	Sersic index
re	half-light radius

Author(s)

Lee Kelvin, <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: astro.

---

read.fits *Read FITS Files*

Description

The generic function 'read.fits' allows FITS binary tables and images (including headers) to be read directly into R.

Usage

```
read.fits(file, hdu = 0, comments = TRUE, strip = c(" ","'"," "),
          maxlines = 50000, xlo = NA, xhi = NA, ylo = NA, yhi = NA)
```

Arguments

file	file name
hdu	header and data unit to be read (0 = all)
comments	output header comments?
strip	lead/trail characters stripped from header 'value' data
maxlines	maximum number of header lines
xlo	lower x pixel sub-region (image only)
xhi	upper x pixel sub-region (image only)
ylo	lower y pixel sub-region (image only)
yhi	upper y pixel sub-region (image only)

Details

The strip argument uses the function 'strip', and removes leading/trailing characters in the order they are given in the argument. The default [c(" ","'"," ")] usually does a good job of removing all trace of FITS header formatting.

The high-level function 'read.fits' encompasses several low-level functions including '.read.fits.hdr', '.parse.fits.hdr', '.read.fits.image' and '.read.fits.table'. These low-level functions require specific inputs, and must be used in the correct order (particularly in the case of multi-HDU FITS files). For this reason, usage of these low-level functions is not advised in most cases.

Value

A list of length two, named '$hdr' and '$dat'. '$hdr' contains the header information for the file, whereas '$dat' contains the imaging or binary table data. Both '$hdr' and '$dat' contain a sub-list for each hdu present in the original file.

Each FITS extension containing a binary table (binary tables are never found in the primary FITS HDU) has additional sub-lists within the '$dat' list, named '$meta' and '$table'. These provide the table meta-data and actual table data itself, respectively.

$hdr	Header
$dat	Data Unit

Author(s)

Lee Kelvin, Andrew Harris, Angus Wright

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Code and inspiration from the FITSio package <http://cran.r-project.org/web/packages/FITSio/index.html>, written by Andrew Harris <harris at astro.umd.edu>.

See Also

The astronomy package: astro.

Examples

```
require(astro)

# create fake data
dat1 = matrix(rnorm(100*50),100,50)
dat2 = matrix(rnorm(50*25),25,50)

# create multi-HDU FITS image
write.fits(list(dat1,dat2), file="astro.fits")
grep("astro.fits", dir(), value=TRUE)

# read FITS image
x = read.fits("astro.fits")
summary(x)

# show keywords in primary header
x$hdr[[1]][,"key"]

# add keywords into secondary header
write.fitskey(key=c("A","COMMENT"), value=c("B","N/A"), file="astro.fits",
comment=c("C","astro.fits created by the 'astro' package"), hdu=2)

# print values of 'NAXIS1' and 'A' from secondary header
read.fitskey(c("NAXIS1","A"), "astro.fits", hdu=2)

# create a plot
x = read.fits("astro.fits")
layout(cbind(c(1,2),c(1,3)), widths=c(1,2))
par("mar"=c(3.1,3.1,1.1,1.1))
im1 = x$dat[[1]]
im2 = x$dat[[2]]
image(1:dim(im1)[1], 1:dim(im1)[2], im1, asp=1, xlab="", ylab="")
label("topleft", txt="HDU 1", cex=2, lwd=0)
box()
image(1:dim(im2)[1], 1:dim(im2)[2], im2, asp=1, xlab="", ylab="",
col=rainbow(1000))
label("topleft", txt="HDU 2", cex=2, lwd=0)
box()
par("mar"=c(3.1,0,1.1,1.1))
aplot(sin, type="n", axes=FALSE, xlab="", ylab="")
hdr = x$hdr[[2]]
ktxt = paste("** astro.fits: HDU 2 Header **\n\nKey\n-----\n",
paste(hdr[,"key"],collapse="\n",sep=""),collapse="",sep="")
vtxt = paste("\n\nValue\n-----\n",paste(hdr[,"value"],collapse="\n",sep=""),
collapse="",sep="")
mtxt = paste("\n\nComment\n-----\n",paste(hdr[,"comment"],collapse="\n",sep=""),
collapse="",sep="")
label("topleft", txt=ktxt, align="left", bty="n")
label("topleft", txt=vtxt, align="left", bty="n", inset=c(1,0.08))
label("topleft", txt=mtxt, align="left", bty="n", inset=c(2,0.08))
label("bottom", txt="note: 'astro.fits' has been automatically deleted",
```

```
bty="n", col="blue", cex=1.5)

unlink("astro.fits")
```

`read.fitshdr` *Read FITS Header*

Description

The mid-level function 'read.fitshdr' allows FITS headers to be read directly into R.

Usage

```
read.fitshdr(file, hdu = 1, comments = TRUE,
             strip = c(" ","'"," "), maxlines = 50000)
```

Arguments

`file`	file name
`hdu`	header and data unit to be read
`comments`	output header comments?
`strip`	lead/trail characters stripped from header 'value' data
`maxlines`	maximum number of header lines

Details

The mid-level function 'read.fitshdr' is a wrapper around 'read.fits', and provides a more simplistic output of that routine.

Value

A matrix of data corresponding to the original FITS image header, with two columns ('key' and 'value') and an optional third column ('comment'), should comments = TRUE (default).

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

`read.fitsim` *Read FITS Image*

Description

The mid-level function 'read.fitsim' allows FITS images to be read directly into R.

Usage

```
read.fitsim(file, hdu = 1, maxlines = 50000, xlo = NA, xhi = NA,
            ylo = NA, yhi = NA)
```

Arguments

`file`	file name
`hdu`	header and data unit to be read
`maxlines`	maximum number of header lines
`xlo`	lower x pixel sub-region (image only)
`xhi`	upper x pixel sub-region (image only)
`ylo`	lower y pixel sub-region (image only)
`yhi`	upper y pixel sub-region (image only)

Details

The mid-level function 'read.fitsim' is a wrapper around 'read.fits', and provides a more simplistic output of that routine.

Value

A matrix of data corresponding to the original FITS image.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

`read.fitskey` *Read FITS Header Keyword*

Description

The mid-level function 'read.fitskey' allows FITS header keywords to be read directly into R.

Usage

```
read.fitskey(key, file, hdu = 1, comments = FALSE,
             strip = c(" ","'"," "), maxlines = 50000)
```

Arguments

`key`	header keyword (may be a vector)
`file`	file name
`hdu`	header and data unit to be read
`comments`	output header comments?
`strip`	lead/trail characters stripped from header 'value' data
`maxlines`	maximum number of header lines

Details

The mid-level function 'read.fitskey' is a wrapper around 'read.fits', and provides a more simplistic output of that routine.

Value

A vector of data equal in length to the input key request. NA is returned where no keys have been found.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

`read.fitstab` *Read FITS Table*

Description

The mid-level function 'read.fitstab' allows FITS binary tables to be read directly into R.

Usage

```
read.fitstab(file, hdu = 2, strip = c(" ","'"," "), maxlines = 50000)
```

Arguments

`file`	file name
`hdu`	header and data unit to be read
`strip`	lead/trail characters stripped from header 'value' data
`maxlines`	maximum number of header lines

Details

The mid-level function 'read.fitstab' is a wrapper around 'read.fits', and provides a more simplistic output of that routine. Note that the FITS table is usually stored in the 2nd HDU of a FITS file, hence the default setting of hdu for this function.

Value

A matrix of data corresponding to the original FITS table, with column names labelled appropriately.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

`scalemark` *Add a Scalemark to a Plot*

Description

Adds a scalemark to a figure, denoting angular size.

Usage

```
scalemark(len = "AUTO", txt = "AUTO", pixsize = NA,
          col = "white", linecol = col, bg = "grey25",
          pos = "bottomleft", inset = 0.1, cex = 0.9, lwd = 1.2)
```

Arguments

`len`	angular size to represent ('AUTO' = automatic best fit)
`txt`	angular size text ('AUTO' = automatic as above)
`pixsize`	pixel scale (arcsec/pixel)
`col`	colour
`linecol`	line colour
`bg`	shadow colour
`pos`	position (bottomleft/topleft/topright/bottomright)
`inset`	inset into plot
`cex`	text expansion factor
`lwd`	line widths

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

schechter *The Schechter Function*

Description

This function calculates the single or double Schechter function, most usually associated with the luminosity function.

Usage

```
schechter(x, knee, slope, norm, bw = 0.1, mag = FALSE, log = FALSE, ...)
```

Arguments

x	input values
knee	the knees(s) of the Schechter function (L_star/M_star)
slope	the slope(s) of the Schechter function (alpha)
norm	the normalisation(s) of the Schechter function (phi_star)
bw	integration bin width sizes
mag	are the input data magnitudes?
log	are the input data logged?
...	additional arguments to be passed to 'integrate'

Value

A vector of length equal to the length of data representing the number density *per dex* at each input data point. Note: to convert the final number densities into their original bin-width (e.g., per 0.5 dex) multiply the output of this function by the bin-width.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Blanton M. R., Lupton R. H., Schlegel D. J., Strauss M. A., Brinkmann J., Fukugita M., Loveday J., 2005, ApJ, 631, 208

Driver S. P., Popescu C. C., Tuffs R. J., Graham A. W., Liske J., Baldry I., 2008, ApJ, 678, L101

Baldry I. K. Driver S. P., Loveday J., et al., 2012, MNRAS, 421, 621

See Also

The astronomy package: astro.

Examples

```
#
#
# Driver et al. 2008, Figure 2 (magnitude)
#
#

# setup input magnitudes
mag = seq(-24,-17,len=100)

# calculate number densities
num = schechter(mag, bw = 0.5, mag = TRUE, log = FALSE, knee = -21.32,
slope = -1.32, norm = 4.8E-3)

# plot
aplot(mag, log10(0.5*num), las=1, ylim=c(-6,-2), type="l", xaxs="i", yaxs="i",
nxmin=1, xlab="Magnitude / mag",
ylab=bquote(paste("log ", phi, " [ 0.5mag ]", " "^-1)),
main="Driver et al. 2008, Figure 2")
label("bottomright", lab=bquote(paste("M* = -21.32     ",
alpha, " = -1.32     ", phi, "* = 4.8x", 10^{-3})), inset=0.1)

#
#
# Baldry et al. 2012, Figure 13 (stellar mass)
#
#

# setup input masses
mass = seq(7,11.6,len=100)

# calculate number densities
num = schechter(mass, bw = 0.1, mag = FALSE, log = TRUE, knee = 10.66,
slope = c(-0.35,-1.47), norm = c(3.96E-3,0.79E-3))

# plot
aplot(log10(mass), num, las=1, ylim=c(1e-5,2e-1), type="l", log="y", xaxs="i",
yaxs="i", xlab="log (M / Msun)",
ylab=bquote(paste("number density (", dex^{-1}, " ", , Mpc^{-3}, ")")),
yformat="p", main="Baldry et al. 2012, Figure 13")
label("bottomleft",
lab=bquote(paste("log M* = 10.66   ", alpha[1], " = -0.35   ",
phi, "*", ""[1], " = 3.96x", 10^{-3}, "   ", alpha[2], " = -1.47   ",
phi, "*", ""[2], " = 0.79x", 10^{-3})), inset=0.1)
```

---

schechter.bin *Calculate Schechter Function Binned Variables*

Description

This function calculates a Schechter function fit to a set of input data.

Usage

```
schechter.bin(data, vmax = NA, range = range(data), lim1 = NA,
              lim2 = NA, numlim = 1, volume = max(vmax), bw = 0.1,
              null = 1E-9)
```

Arguments

data	input data vector
vmax	vector of maximum comoving volumes within which object could lie
range	data range of interest
lim1	lower data limit for fitting
lim2	upper data limit for fitting
numlim	lower number (per bin) limit for fitting
volume	total volume across which the data has been collected (default 1 if vmax = NA)
bw	bin width sizes
null	value of null

Value

bins	bin boundaries
binmid	bin midpoints
binlo	bin lower boundaries
binhi	bin upper boundaries
num	number per bin (note: if using vmax values, these are weights)
den	density per bin
err	error per bin
errlo	lower error limit
errhi	upper error limit
fitbinmid	bin midpoints after data limits applied
fitbinlo	bin lower boundaries after data limits applied
fitbinhi	bin upper boundaries after data limits applied
fitnum	number per bin after data limits applied (note: if using vmax values, these are weights)
fitden	density per bin after data limits applied
fiterr	error per bin after data limits applied
fiterrlo	lower error limit after data limits applied
fiterrhi	upper error limit after data limits applied

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Blanton M. R., Lupton R. H., Schlegel D. J., Strauss M. A., Brinkmann J., Fukugita M., Loveday J., 2005, ApJ, 631, 208

Driver S. P., Popescu C. C., Tuffs R. J., Graham A. W., Liske J., Baldry I., 2008, ApJ, 678, L101

Baldry I. K. Driver S. P., Loveday J., et al., 2012, MNRAS, 421, 621

See Also

The astronomy package: `astro`.

---

schechter.ellipse *Calculate Error Matrices for a Schechter Function Fit*

Description

This function calculates ellipsoidal error matrices for a given Schechter Function fit.

Usage

```
schechter.ellipse(data, vmax = NA, knee, slope, norm, chi2,
                  datarange = NA, kneerange = c(-24,-16),
                  sloperange = c(-2,1.5), kneeofflims = NA,
                  slopeofflims = NA, kneestep = 0.5, slopestep = 0.1,
                  kneesteps = NA, slopesteps = NA, lim1 = NA, lim2 = NA,
                  numlim = 1, method = "nlminb", volume = max(vmax),
                  bw = 0.1, mag = FALSE, log = FALSE, null = 1E-9, ...)
```

Arguments

`data`	input data vector
`vmax`	vector of maximum comoving volumes within which object could lie
`knee`	the knees(s) of the Schechter function (L_star/M_star)
`slope`	the slope(s) of the Schechter function (alpha)
`norm`	the normalisation(s) of the Schechter function (phi_star)
`chi2`	the full chi2 result from this fit (not reduced)
`datarange`	the range across which the data is evaluated
`kneerange`	range of knee values
`sloperange`	range of slope values
`kneeofflims`	alternative to kneerange, vector length 2 describing limit offsets from knees

slopeofflims	alternative to sloperange, vector length 2 describing limit offsets from slopes
kneestep	the matrix step in knee values
slopestep	the matrix step in slope values
kneesteps	alternative to kneestep, the number of steps in the matrix
slopesteps	alternative to slopestep, the number of steps in the matrix
lim1	lower data limit for fitting
lim2	upper data limit for fitting
numlim	lower number (per bin) limit for fitting
method	choice of 'nlminb' (recommended) or one of 'optim's minimisation methods (e.g., 'Nelder-Mead')
volume	total volume across which the data has been collected (default 1 if vmax = NA)
bw	bin width sizes
mag	are the input data magnitudes?
log	are the input data logged?
null	value of null
...	additional arguments to be passed to 'integrate'

Value

knees	knee bin midpoints
slopes	slope bin midpoints
res1	result matrix for knee1/slope1
res2	result matrix for knee2/slope2
s1	1 sigma chi2 limit (min+2.30)
s2	2 sigma chi2 limit (min+6.17)
s3	3 sigma chi2 limit (min+11.8)
kneelo1	lower knee error (all sigmas)
kneehi1	upper knee error (all sigmas)
slopelo1	lower slope error (all sigmas)
slopehi1	upper slope error (all sigmas)
kneelo2	lower secondary knee error (all sigmas)
kneehi2	upper secondary knee error (all sigmas)
slopelo2	lower secondary slope error (all sigmas)
slopehi2	upper secondary slope error (all sigmas)

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

schechter.fit *Fit to the Schechter Function*

Description

This function calculates a Schechter function fit to a set of input data.

Usage

```
schechter.fit(data, vmax = NA, knee, slope, norm, knee.alt = NA,
              slope.alt = NA, norm.alt = NA, kneelo = -Inf,
              slopelo = -Inf, normlo = 0, kneehi = Inf, slopehi = Inf,
              normhi = Inf, fixk1 = FALSE, fixs1 = FALSE, fixn1 = FALSE,
              fixk2 = FALSE, fixs2 = FALSE, fixn2 = FALSE,
              range = range(data), lim1 = NA, lim2 = NA, numlim = 1,
              method = "nlminb", volume = max(vmax), bw = 0.1,
              mag = FALSE, log = FALSE, null = 1E-9, error = "jack",
              subvol = 10, sampnum = subvol, msun = solar("r"))
```

Arguments

data	input data vector
vmax	vector of maximum comoving volumes within which object could lie
knee	the knees(s) of the Schechter function (L_star/M_star)
slope	the slope(s) of the Schechter function (alpha)
norm	the normalisation(s) of the Schechter function (phi_star)
knee.alt	alternative knees(s) of the Schechter function (L_star/M_star)
slope.alt	alternative slope(s) of the Schechter function (alpha)
norm.alt	alternative normalisation(s) of the Schechter function (phi_star)
kneelo	a lower bound on the knee parameter
slopelo	a lower bound on the slope parameter
normlo	a lower bound on the norm parameter
kneehi	an upper bound on the knee parameter
slopehi	an upper bound on the slope parameter
normhi	an upper bound on the norm parameter
fixk1	fix the first knee?
fixs1	fix the first slope?
fixn1	fix the first normalisation?
fixk2	fix the second knee?
fixs2	fix the second slope?
fixn2	fix the second normalisation?

range	data range of interest
lim1	lower data limit for fitting
lim2	upper data limit for fitting
numlim	lower number (per bin) limit for fitting
method	choice of 'nlminb' (recommended) or one of 'optim's minimisation methods (e.g., 'Nelder-Mead')
volume	total volume across which the data has been collected (default 1 if vmax = NA)
bw	bin width sizes
mag	are the input data magnitudes?
log	are the input data logged?
null	value of null
error	parameter error estimation method [jack/boot]
subvol	number of sub-volumes to split the input data into
sampnum	number of samplings to be made for bootstrapping method
msun	absolute magnitude of the sun (to be used in calculation of the luminosity density should mag=TRUE)

Value

binmid	bin midpoints
num	number per bin
den	density per bin
err	error per bin
errlo	lower error limit
errhi	upper error limit
par	best fit parameters
parlo	lower error estimates on recovered fit parameters
parhi	upper error estimates on recovered fit parameters
j	luminosity density
jlo	lower luminosity density limit
jhi	upper luminosity density limit
chi2	chi2 value
dof	number of degrees of freedom
rchi2	reduced chi2
pval	probability of observing chi2 value by chance [significant: pval < 0.05]
denlim	lower density limit used
hessian	optim hessian output

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Blanton M. R., Lupton R. H., Schlegel D. J., Strauss M. A., Brinkmann J., Fukugita M., Loveday J., 2005, ApJ, 631, 208

Driver S. P., Popescu C. C., Tuffs R. J., Graham A. W., Liske J., Baldry I., 2008, ApJ, 678, L101

Baldry I. K. Driver S. P., Loveday J., et al., 2012, MNRAS, 421, 621

See Also

The astronomy package: `astro`.

`scmean` *Sigma-Clipped Mean*

Description

Calculates a sigma-clipped mean

Usage

```
scmean(x, mult = 3, loop = 10)
```

Arguments

`x`	a vector of data
`mult`	multiples of 1-sigma with which to clip the data (about the mean)
`loop`	number of loops clipping the data

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: `astro`.

`sersic` *Sersic Function*

Description

Calculates Sersic parameters

Usage

```
sersic(mag, re, n, e = 0, r = re)
```

Arguments

`mag`	total magnitude
`re`	half-light radius
`n`	Sersic index
`e`	ellipticity (1 - b/a)
`r`	radius of interest

Value

`mag`	magnitude within r
`magdiff`	difference between total magnitude and magnitude within r
`mu`	surface brightness at r
`muavg`	average surface brightness within r
`inten`	intensity at r
`lum`	luminosity within r
`lumtot`	total luminosity
`lumfrac`	fraction of total luminosity contained within r

Author(s)

Lee Kelvin, Aaron Robotham

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Graham A. W., Driver S. P., 2005, PASA, 22, 118

See Also

The astronomy package: `astro`.

Examples

```
r = seq(0,8,len=500)
ns = c(0.5,1,2,4,10)
col = hsv(seq(2/3,0,len=length(ns)))

layout(c(1,2))
par("mar"=c(0,4.1,0,2.1))
par("oma"=c(5.1,0,2.1,0))

# surface brightness plot
mu = convmu(r=r, mu=20, re=1, n=ns[1])
aplot(r,abs(mu), ylim=c(28,12), type="l", nxmaj=5, nxmin=1, nymin=4, xlab="",
ylab=bquote(paste(mu," / mag ", arcsec^{-2})), xaxs="i", yaxs="i", las=1,
col=col[1], labels=2)
for(i in 2:length(ns)){
    mu = convmu(r=r, mu=20, re=1, n=ns[i])
    lines(r, abs(mu), col=col[i])
}
label("top", txt=bquote(paste(mu["e"]," = 20")), bty="n", inset=0.3)
text(0.9,17,labels="n = 10")
text(2,25,labels="n = 0.5")
legend("topright", legend=c(0.5,1,2,4,10), col=col, bty="n", lty=1, inset=0.03)
box()

# magnitude difference plot
md = sersic(mag=0, re=1, n=ns[1], r=r)$magdiff
aplot(r,abs(md), ylim=c(0,3), type="l", nxmaj=5, nxmin=1, nymaj=4, nymin=4,
xlab="", ylab="m (< r) / mag", xaxs="i", yaxs="i", las=1, col=col[1])
for(i in 2:length(ns)){
    md = sersic(mag=0, re=1, n=ns[i], r=r)$magdiff
    lines(r, abs(md), col=col[i])
}
abline(h=abs(sersic(mag=0, re=1, n=1, r=1)$magdiff), lty=2)
label("top", txt=bquote(paste(m[tot]," = 0")), bty="n", inset=0.3)
text(7,0.4,labels="n = 10")
text(1,2,labels="n = 0.5")
mtext(bquote(paste("r / ",r[e],sep="")), side=1, line=2.5)
box()
```

---

shade *Adds a Shaded Region to a Figure*

Description

A convenience wrapper around the 'polygon' function.

Usage

```
shade(x, y1, y2, col = hsv(alpha=0.5), border = NA, ...)
```

Arguments

x	x data
y1	y data (limit 1)
y2	y data (limit 2)
col	fill colour
border	border colour
...	arguments to be passed to 'polygon'

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

shadowtext *Add Shadowed Text to a Plot*

Description

Add text with a shadowed background to a plot.

Usage

```
shadowtext(x, y = NULL, labels, col = "white", bg = "grey25",
           theta = (1:8/4)*pi, r1 = 0.06, r2 = 0.04, ...)
```

Arguments

x,y	numeric vectors of coordinates where the text labels should be written
labels	a character vector or expression specifying the text to be written
col	foreground text colour
bg	background text colour
theta	position angle offsets for the background shading
r1	shadow width
r2	shadow height
...	additional arguments to be passed to 'text'

Details

'shadowtext' works by plotting the text initially at positions slightly offset from the central position in a darker background colour, and then over-plots the desired text (twice, to avoid anti-aliasing effects).

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

solar *Display the Absolute Magnitude of the Sun*

Description

A quick reference for the absolute magnitude of the sun, in either AB or Vega.

Usage

```
solar(band = "r", vega = FALSE, source = FALSE)
```

Arguments

band	The band of interest (case sensitive)
vega	Output Vega magnitude instead?
source	Show the source of information?

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

strip *Strip Leading/Trailing Characters*

Description

Strips leading/trailing characters from the input string. Particularly useful for extracting information from FITS file headers which are embedded in a complex string, for example.

Usage

```
strip(x, strip = " ")
```

Arguments

x	input string
strip	character to be stripped (may be a vector)

Details

Characters in the string will be stripped 'outside in', from left-to-right in the order they are given in the argument. See examples below for more detail.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Examples

```
require("astro")

x = "   'lee'   "

strip(x, strip=" ")
#[1] "'lee'"

strip(x, strip=c(" ","'"))
#[1] "lee"

strip(x, strip=c("'"," "))
#[1] "'lee'"
```

---

write.fits *Write FITS Files*

Description

The generic function 'write.fits' allows FITS images (including headers) to be written directly from R.

Usage

```
write.fits(x, file = "star.fits", type = "single", hdu = 0)
```

Arguments

x	input data (may be a list, see details)
file	file name
type	data format type [single/double/auto]
hdu	write a specific hdu from input list 'x' (0 = all/NULL)

Details

'write.fits' will write out the data in object x into the named file as a FITS image. x can be in the form of a named list such as that output by 'read.fits', or simply a matrix (or more simply, a vector) of data.

This function allows custom headers to be used when creating the FITS image. Checks for FITS-critical keywords will be made prior to writing the image, and, if necessary, these keywords will be forcefully changed/removed/added in order to comply with the FITS standard should the header provided violate these rules.

The high-level function 'write.fits' encompasses several low-level functions including '.dummy.fits.hdr', '.check.fits.hdr' and '.make.fits.hdr'. These low-level functions require specific inputs, and must be used in the correct order (particularly in the case of multi-HDU FITS files). For this reason, usage of these low-level functions is not advised in most cases.

Note that writing of FITS binary tables is not yet implemented.

Author(s)

Lee Kelvin, Andrew Harris, Aaron Robotham

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

References

Code and inspiration from the FITSio package <http://cran.r-project.org/web/packages/FITSio/index.html>, written by Andrew Harris <harris at astro.umd.edu>.

See Also

The astronomy package: astro.

Examples

```
require(astro)

# create fake data
dat1 = matrix(rnorm(100*50),100,50)
dat2 = matrix(rnorm(50*25),25,50)

# create multi-HDU FITS image
write.fits(list(dat1,dat2), file="astro.fits")
grep("astro.fits", dir(), value=TRUE)
```

```
# read FITS image
x = read.fits("astro.fits")
summary(x)

# show keywords in primary header
x$hdr[[1]][,"key"]

# add keywords into secondary header
write.fitskey(key=c("A","COMMENT"), value=c("B","N/A"), file="astro.fits",
comment=c("C","astro.fits created by the 'astro' package"), hdu=2)

# print values of 'NAXIS1' and 'A' from secondary header
read.fitskey(c("NAXIS1","A"), "astro.fits", hdu=2)

# create a plot
x = read.fits("astro.fits")
layout(cbind(c(1,2),c(1,3)), widths=c(1,2))
par("mar"=c(3.1,3.1,1.1,1.1))
im1 = x$dat[[1]]
im2 = x$dat[[2]]
image(1:dim(im1)[1], 1:dim(im1)[2], im1, asp=1, xlab="", ylab="")
label("topleft", txt="HDU 1", cex=2, lwd=0)
box()
image(1:dim(im2)[1], 1:dim(im2)[2], im2, asp=1, xlab="", ylab="",
col=rainbow(1000))
label("topleft", txt="HDU 2", cex=2, lwd=0)
box()
par("mar"=c(3.1,0,1.1,1.1))
aplot(sin, type="n", axes=FALSE, xlab="", ylab="")
hdr = x$hdr[[2]]
ktxt = paste("** astro.fits: HDU 2 Header **\n\nKey\n-----\n",paste(hdr[,"key"],
collapse="\n",sep=""),collapse="",sep="")
vtxt = paste("\n\nValue\n-----\n",paste(hdr[,"value"],collapse="\n",sep=""),
collapse="",sep="")
mtxt = paste("\n\nComment\n-----\n",paste(hdr[,"comment"],collapse="\n",sep=""),
collapse="",sep="")
label("topleft", txt=ktxt, align="left", bty="n")
label("topleft", txt=vtxt, align="left", bty="n", inset=c(1,0.08))
label("topleft", txt=mtxt, align="left", bty="n", inset=c(2,0.08))
label("bottom", txt="note: 'astro.fits' has been automatically deleted",
bty="n", col="blue", cex=1.5)

unlink("astro.fits")
```

`write.fitshdr` *Write FITS Headers*

Description

The mid-level function 'write.fitshdr' allows FITS headers to be written directly from R.

Usage

```
write.fitshdr(hdr, file, hdu = 1)
```

Arguments

hdr	input header (see details)
file	file name
hdu	header and data unit to be written to

Details

Input headers ('hdr') must contain at least two columns, one named 'key' and one named 'value' (a 'comment' column is optional). For an example of the format expected, see 'read.fitshdr' and 'read.fits'.

The mid-level function 'write.fitshdr' is a wrapper around 'read.fits' and 'write.fits', and provides a shortcut to updating the header of an already existent FITS file.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

---

write.fitskey *Write FITS Header Keywords*

Description

The mid-level function 'write.fitskey' allows FITS header keywords to be written directly from R.

Usage

```
write.fitskey(key, value, file, comment = "", hdu = 1)
```

Arguments

key	header keyword (may be a vector)
value	header value (may be a vector)
file	file name
comment	header comments (may be a vector)
hdu	header and data unit to be written to

Details

The mid-level function 'write.fitskey' is a wrapper around 'read.fits' and 'write.fits', and provides a shortcut to updating individual keywords of an already existent FITS file.

Author(s)

Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

`writeBin64` *Transfer Binary Data To a Connection*

Description

Write binary data to a connection.

Usage

```
writeBin64(object, con, size = NA_integer_,
           endian = .Platform$endian, useBytes = FALSE)
```

Arguments

`object`	An R object to be written to the connection.
`con`	A connection object or a character string naming a file or a raw vector.
`size`	integer. The number of bytes per element in the byte stream. The default, 'NA_integer_', uses the natural size. Size changing is not supported for raw and complex vectors.
`endian`	The endian-ness ('big' or 'little') of the target system for the file. Using 'swap' will force swapping endian-ness.
`useBytes`	See 'writeLines'.

Details

An extension of the standard R function writeBin. This 64-bit extension splits data into 2^30 chunks, and writes each out sequentially, overcoming the 2^31-1 limit of the original function.

Author(s)

Aaron Robotham

Maintainer: Lee Kelvin <lee.kelvin@uibk.ac.at>

See Also

The astronomy package: astro.

Index