ARCHES WP4 Software documentation

François-Xavier Pineau
CDS Strasbourg

November 27, 2015
v0.6

Contents

List of Figures

List of Tables

1 Overview

1.1 Script principle

A script consists in a sequence of xmatches. You have first to load two tables before xmatching them (loading only one table will result in performing a self-xmatch on this table). You can then follow on by loading a new table and xmatching it with the result of the previous xmatch.
The *left* table is the one which is defined first, reading a script from top to bottom. So, for the first xmatch the *left* table is the first loaded table and the *right* table is the second loaded table. For the next xmatches, the result of the previous xmatch is always the *left* table and the new loaded table is the *right* table. To summarize this, the logical structure of a script is as follows:

create table_1
create table_2
xmatch table_1 (left) versus table_2 (right) –> table_1_2
create table_3
xmatch table_1_2 (left) versus table_3 (right) –> table_1_2_3
create table_4
xmatch table_1_2_3 (left) versus table_4 (right) –> table_1_2_3_4
...

The commands used to create a table are explained in §2 and the ones used to perform a xmatch are explained in §5.

Some xmatches (i.e. the probabilistic xmatch for more than two tables) requires all tables to be loaded. In this case, the logical struture becomes:

create table_1
create table_2
... create table_n
xmatch all tables from 1 to n –> table_1..._n
create table_n+1
xmatch table_1..._n (left) versus table_n+1 (right) –> table_1..._n+1
...

1.2 Stable columns

To be able to perform successive xmatches, we have defined a set of "stable" astrometric columns. The minimal set of "stable" columns consists in the two columns containing the positions, i.e. the right ascension (α) and the declination (δ). By "stable" we mean that from two input tables having each a set of α and δ columns, the xmatch creates a new table having only one set of α and δ columns. We will have to decide how we merge [1] the two sets of input "stable" columns to create the new set of "stable" columns.
All "stable" columns are "merged" while all "non-stable" columns are concatenated in the new table. §2.7 deals with the commands used to set the "stable" columns of input tables, §2.8.1 deals with the command used to set the "non-stable" columns, and §5.6 deals with the commands used to merge the input tables "stable" columns into the new table resulting from a xmatch.

1. "merge" has here a general meaning, we can e.g. decide not to merge but to keep the left (or the right) table "stable" columns.

We give here the list of the possible "stable" columns:

posRA, posDec positions, in decimal degrees;

posEpoch epoch at which the positions have been computed, in years;

ePosA, ePosB, ePosPA 1σ error ellipse on the positions, in arcseconds (ePosA, eposB) and degrees (ePosPA);

ePosEpoch epoch at which the positional error ellipse has been computed, in decimal Julian years;

pmRcd, pmDec proper motion (on $\alpha \cos \delta$ and δ), in mas/yr.

ePmA, ePmB, ePmPA error ellipse on the proper motion, in mas/yr and degrees;

extA, extB, extPA extension ellipse, in arcsec and degrees;

dMEC minimum enclosing cone diameter, in arcsec (result of a xmatch);

nPos number of positions used to compute the merged positions (result of a xmatch);

chi2Pos value of the positional χ^2 (result of a xmatch);

In the future, we may add the parallax (*plx*) and its error (*ePlx*), the radial velocity (*rv*) and its error (*eRv*), proper motion χ^2 parameters (*nPM* and *chi2PM*), ...

2 Loading and creating a table

2.1 Commands overview

Here is the list of the commands available to load a table. The order is important and we strongly recommend to use them in the same order as they appear here.
A '?' before a command means it is optional. Optional commands can become mandatory depending on the chosen xmatch algorithm, e.g. *set poserr* is needed to use the *chi2* xmatch.

get *LOADER options*
? get *LOADER options*
? ...
? union col=*id_column_name*
? cache *file_name* (fits|votable)
? save *file_name* (fits|votable|csv)
? buildmoc name=... norder=... ra=... dec=...
? save *moc_file_name* (fits|json)
? where *condition*
set pos ra=*expression* dec=*expression*
? set epoch year=*expression*
? set poserr type=*ERR_TYPE* param1=*expression* param2=... param3=...
? set pm rcd=*expression* dec=*expression*
? set pmerr type=*ERR_TYPE* param1=*expression* param2=... param3=...
? set extent type=*EXT_TYPE* param1=*expression* param2=... param3=...
? set cols (*|/*regexp*/|*col_name*|*expression as new_col_name*]) [, ...]
? addmeta *col_name* [datatype=...] [unit=...] [ucd=...] [precision=...] [...]
? addmeta ...
? ...
? prefix *prefix*
? save *file_name* (fits|votable|csv)

expression means a colname or a formula using or not one or several column name(s). The list of available formulas is given in appendix §A.

Each command is described in the next sections.

2.2 Command *get*: loads an input table

Input tables can come from :
– a file on a local drive
– a file accessible through a URL
– the VO Simple Cone Search protocol
– VizieR
– a RowCatFile (specific CDS binary file)

Several data loaders are available to load a file according to its origin. Each data loader has a name and a set of parameters which are described in the following sections.
You can chain *get* commands to load and concatenate tables having the same schema. See command *union* to merge tables possibly having common/duplicated rows.

2.2.1 The File data loader

Load a file on your local hard drive.

get FileLoader file=*filename*

– *filename* is the complete path of the file you want to load.

Supported formats are basically the formats supported by Topcat: VOTable, FITS, CSV,. . .
Because of its lack of metadata, we recommend not to use CSV.

2.2.2 The URL data loader

Load a file accessible through HTTP.

get URLLoader url=*http://your_url*

Supported formats are the same as the file loader ones.

2.2.3 The VO Simple Cone Search data loader

Load a table using the VO Simple Cone Search protocol.

get VOConeSearchLoader url=... ra=... dec=... sr=... [verb=(1|2|3)]

– *url* - the base URL of the cone search service;
– *ra* - right ascension of the cone center in decimal degrees;
– *dec* - declination of the cone center in decimal degrees;
– *sr* - search radius of the cone in decimal degrees;
– *verb* - ? verbosity must be 1, 2 or 3.

2.2.4 The VizieR data loader

Load a catalogue available through VizieR.
Two modes are available: the *allsky* mode and the *cone* mode.

Mode allsky Load all the given table sources.

get VizieRLoader tabname=*vizier_tab_name* mode=allsky [allcolumns]

WARNING: do not use the mode *allsky* with large catalogues!

Mode cone Load the table sources which are inside the given cone.

get VizieRLoader tabname=*vizier_tab_name* mode=cone center="*STRING*" radius=*REAL*(deg|arcmin|arcsec) [allcolumns]

– *center* - can be an object name or a position, e.g. *center="M 31"* or *center="12.345678 +87.654321"*;
– *radius* - radius of the cone search, in one of the 3 possible units, e.g. *radius=2.4arcmin*.

For both modes

– *tabname* - name of the table in VizieR, e.g. for 2MASS: *tabname=II/246/out*;
– *allcolumns* - ? load all columns available in VizieR, not only default ones.

Examples Allsky with the Downes catalogue:

get VizieRLoader tabname=V/123A/cv mode=allsky allcolumns

Cone with 2MASS:

get VizieRLoader tabname=II/246/out mode=cone center="12.256460 +56.878920" radius=2arcmin allcolumns

2.2.5 The RowCatFile (RCF) loader

Load a table from a query on a RowCatFile. Several modes are availables.

get RCFLoader tabname=STRING mode=*mode mode_args* filter=EXP limit=INT

In which *filter* and *limit* are optional:
- *filter* - a JAVA like expression to pre-filter sources
- *limit* - maximum number of rows to be returned

mode=allsky Load all sources the RCF file contains.

mode=allsky

mode=cone Load the file sources which are inside the given cone.

mode=cone center=RA+-DEC radius=REAL

- *center* - a position in decimal degrees or JNAME, e.g. *center="12.345678+87.654321"* or *JHHMMSS.S[+-]DDMMSS.S*;
- *radius* - radius of the cone search in degrees.

mode=zone Load the file sources which are inside the given zone (range in RA and DEC).

mode=zone minRA=REAL maxRA=REAL minDec=REAL maxDec=REAL

- *minRA* - smallest RA in decimal degrees;
- *maxRA* - highest RA in decimal degrees (minRA > maxRA if the zone contains RA=0);
- *minDec* - smallest Dec in decimal degrees;
- *maxDec* - highest Dec in decimal degrees.

mode=box Load the file sources which are inside the given box.

mode=box center==RA+-DEC width=REAL height=REAL pa=REAL

- *center* - center of the box in decimal degrees or JNAME, e.g. *center="12.345678+87.654321"* or *JHHMMSS.S[+-]DDMMSS.S*;
- *width* - width of the box in degrees;
- *height* - height of the box in degrees;
- *pa* - position angle in degrees.

mode=healpix Load the file sources which are inside the given HEALPix cell.

mode=healpix norder=INT ipix=LONG bnorder=INT

- *center* - center of the box in decimal degrees or JNAME, e.g. *center="12.345678+87.654321"* or *JHHMMSS.S[+-]DDMMSS.S*;
- *width* - width of the box in degrees;
- *height* - height of the box in degrees;
- *pa* - position angle in degrees.

mode=moc Load the file sources inside the given inline MOC or the given MOC file. (TO BE IMPLEMENTED, ask me by email if you need it).

2.3 Common *union*: merge the result of several *get* commands by removing duplicated rows

Perform the union of several tables loaded by chaining *get* commands, removing duplicated rows based on a columns containing a uniq identifier.

union col=*id_colname*

– *col* - name of the column containing a uniq identifier.

Remark: when several rows share a same identifier, only the first one is kept in output.

2.4 Command *cache*: pass over the *get* command

Pass over the *get* command and load the given file if it exists, else execute the *get* command and save its result in the given file.

cache *filename* (votable|fits)

The *cache* command is very similar to the *save* command. But, contrary to the *save* command which overwrites existing files, the *cache* command saves the result of a *get* only if the given file does not exist. If the given file already exists, it is loaded instead of executing the *get* command.

Remarks:

– obviously, it is useless to use the *cache* command with the file data loader (see §2.2.1);

– after a *get* don't use both *save* and *cache* commands, or at least not in that order (*cache* must be first).

Warning: be carefull to assign different names to files associated with different *get* commands!

Example:

```
get VizieRLoader tabname=I/284/out mode=cone center="12.256460
    +20.878920" radius=10.2arcmin allcolumns
cache usnob1.cached.votable votable
```

At first execution, the *get* command is executed and its result is saved in the file *usnob1.cached.vot*.

At the second execution, the *get* command is not executed, the file *usnob1.cached.vot* is loaded instead (if it has not been removed manually in which case the *get* command is executed).

2.5 Command *where*: select rows

The *where* command removes all the sources of the loaded table which do not match the given condition. The syntax relies on the Java language and is thus very similar to the one used in TOPCAT.

where *expression*

Example 1 with 2MASS:

```
where Jmag - Hmag > -2 && Jmag - Hmag < 2
```

Example 2 with 2MASS:

```
where Jmag-Hmag < 3 && havDist(deg2rad(RAJ2000), deg2rad(DEJ2000),
deg2rad(12.345678), deg2rad(-87.654321)) < deg2rad(10 / 60.0)
```

Here, *havDist(...)* compute the angular distance between two points using the Haversine formula and *deg2rad* convert degrees into radians.
Example 3 with USNO-B.1:

```
where (pmRA == null || pmDE == null || sqrt(pmRA * pmRA + pmDE *
pmDE) < 100) && (VarType == null || !VarType.equals("SN"))
```

The list of availables constants and functions is available §A.

2.6 Command *buildmoc*: build a local MOC

Build a MOC.

buildmoc name=STRING norder=INT ra=*expression* dec=*expression*

in which:
- *name* - The name given to the MOC (to be used in a command *addcol moc*)
- *norder* - HEALPix level at which the MOC is built (see tab. 1);
- *ra* - right ascension of position in decimal degrees;
- *dec* - declination of the position in decimal degrees;
- *expression* - can be a column name or a more complex expression including (or not) column names.

Example:

```
buildmoc name=2MASS_MOC norder=11 ra=RAJ2000 dec=DEJ2000
```

WARNING: the value of the *norder* should be choosen so that the MOC resolution is high but without having empty cells in the area covered by the catalogue. So the *norder* you will put will probably depends on the galactic lattitude.

2.7 Command *set*: select columns which may be used in the xmatch

As explained in §1.2, there are two kinds of columns, and two ways to select them:
- columns (mainly astrometric ones) which are used in the xmatch process and are merged to be reused in the next xmatch (the "stable" columns);
- additional columns which are not directly used in the xmatch (flags, photometry, class, ...) and which are concatenated after each xmatch.

This section refers to the first type. For how to select the "non-stable" columns, see §2.8.

2.7.1 *set pos*

Sets the position to be used in the xmatch.

set pos ra=*expression* dec=*expression*

- *ra* - right ascension of position in decimal degrees;
- *dec* - declination of the position in decimal degrees;
- *expression* - can be a column name or a more complex expression including (or not) column names.

Example:

```
set pos ra=RAJ2000 dec=DEJ2000
```

2.7.2 *set epoch*

Sets the epoch of the positions (defined by *set pos*) to be used in the xmatch.

set epoch year=*expression*

- *year* - epoch of the positions, in decimal Julian years;
- *expression* - can be a column name or a more complex expression including (or not) column names.

Example USNO-B.1:

set epoch year=2000.0

Example 2MASS:

set epoch year=jd2epoch(JD)

Remark: *jd2epoch* is a function which converts Julian days into an epoch in years.

2.7.3 *set poserr*

Sets the errors of the positions (defined by *set pos*) to be used in the xmatch.

set poserr type=ERR_TYPE param1=*exp* param2=*exp* param3=*exp* epoch=*exp*

- *type* - one of the following error type: CIRCLE, ELLIPSE, RCD_DEC_ELLIPSE, COV_ELLIPSE, COR_ELLIPSE
- *param1* - radius or semi-major axis of the 1σ error ellipse or 1σ error on $\alpha \cos\delta$, in arcseconds;
- *param2* - ? semi-minor axis of the 1σ error ellipse or 1σ error on δ, in arcseconds;
- *param3* - ? position angle in degrees or covariance in arcsec^2 or correlation coeff;
- *epoch* - ? epoch of the positional error (useful if we want to take into account proper motions). E.g in USNO-B.1 positions have been computed at epoch 2000 using proper motions, but the errors on the proper motions have not been propagated and are not taken into account into the positional uncertainties.
- *exp* - can be a column name or a more complex expression including (or not) column names.

Example 2MASS:

set poserr type=ELLIPSE param1=errMaj param2=errMin param3=errPA epoch=jd2epoch(JD)

Example USNO-B.1:

set poserr type=RCD_DEC_ELLIPSE param1=sqrt(e_RAJ2000*e_RAJ2000 + 130*130)/1000.0 param2=sqrt(e_DEJ2000*e_DEJ2000 + 130*130)/1000.0 epoch=Epoch

Here, we convert the error from mas to arcsec and we add a systematic error of 0.13 arcseconds (do not take this value seriously).

type=CIRCLE

- *param1* - radius of the 1σ error circle, in arcsec.

type=ELLIPSE

- *param1* - semi-major axis of the 1σ error ellipse, in arcsec;
- *param2* - semi-minor axis of the 1σ error ellipse, in arcsec;
- *param3* - position angle of the 1σ error ellipse, in degrees.

type=RCD_DEC_ELLIPSE Equivalent to ELLIPSE with *param3=0*.

- *param1* - 1σ error on $\alpha \cos\delta$, in arcsec;
- *param2* - 1σ error on δ, in arcsec.

type=COV_ELLIPSE

- *param1* - 1σ error on $\alpha \cos\delta$, in arcsec;
- *param2* - 1σ error δ, in arcsec;
- *param3* - covariance of the 1σ errors, in arcsec^2.

type=COR_ELLIPSE

- *param1* - 1σ error on $\alpha \cos\delta$, in arcsec;
- *param2* - 1σ error δ, in arcsec;
- *param3* - correlation (ρ) between the 1σ errors.

2.7.4 *set pm*

Sets the proper motion to be used in the xmatch.

set pm rcd=*expression* dec=*expression*

- *rcd* - proper motion in $\alpha \cos\delta$, in mas/yr;
- *dec* - proper motion in δ, in mas/yr;
- *expression* - can be a column name or a more complex expression including (or not) column names.

Example on USNO-B.1:

set pm rcd=pmRA dec=pmDE

2.7.5 *set pmerr*

Sets the proper motion errors to be used in the xmatch.

set pmerr type=ERR_TYPE param1=*exp* param2=*exp* param3=*exp* epoch=*exp*

Parameters are exactly the same as the position error ones (see §2.7.3), except that units are mas/yr.
Example USNO-B.1:

set pmerr type=RCD_DEC_ELLIPSE param1=e_pmRA param2=e_pmDE

2.7.6 *set extent*

Sets the object extension to be used in the xmatch.

set pmerr type=ERR_TYPE param1=*exp* param2=*exp* param3=*exp* epoch=*exp*

Parameters are the same as the position error ones (see §2.7.3) but only ELLIPSE and CIRCLE are accepted. Units are arcseconds.

2.8 Selecting tables columns, defining and enriching their metadata

When you load a table, you have the possibility to select the columns you want to save, the non-selected ones being discarded. The selected columns (and their associated metadata) are kept in memory during the successive xmatches and are printed in the output result file. The software allows you to overwrite or to add a column metadata (see command *addmeta*, §2.8.2).

As in SQL you can also create new synthetic columns. In that case YOU HAVE TO define the new column metadata using the *addmeta* command (see §2.8.2).

2.8.1 *set cols*

Selects the columns of the input table you want to keep in the output.

set cols (*|/*regexp*/|*col_name*|*expression as new_col_name*]) [, ...]

in which

- * - select all the input table columns;
- *col_name* - select the column having the given name;
- *regexp* - select the input table columns which names are matched by the given regular expression;
- *expression as new_col_name* - create a new synthetic column from the given expression, *new_col_name* is the name of the new created column.

Example with 2MASS:

set cols 2MASS, RAJ2000, DEJ2000, JD, /err((Maj)|(Min)|(PA))/, /(e_)?[JHK]mag/, /.flg/, Jmag-Kmag as J-K

Here *2MASS, RAJ2000, DEJ2000, JD* are column names; *err((Maj)|(Min)|(PA))* is a regular expression which selects columns *errMaj, errMin, errPA* (we could have used the more concise regexp *err.+*; *(e_)?[JHK]mag* is a regular expression which selects columns *Jmag, e_Jmag, Hmag, e_Hmag, Kmag and e_Kmag*; *.flg* is a regular expression which selects columns *Qflg, Rflg, Bflg, Clfg, Xflg and Aflg* and *Jmag-Kmag as J-K* is a new syntethic column named *J-K* and which stores the difference between columns *Jmag* and *Kmag*.

2.8.2 *addmeta*

Sets or overwrites the given metadata of the given column.

addmeta *col_name metatype*=val [, *metatype*=val] ...

in which

- *col_name* - name of the column we want to define or change metadata;
- *metatype* - the metadata we want to change can be: *datatype*, *arraysize*, *precision*, *unit*, *desc*, *ucd*, *utype*, *width*, *xtype*, *ref*, *type*, *linkValue*, *linkHref* ; To have more informations of the meaning of those metadata, see e.g. the IVOA document which defines VOTables;
- *val* - the new value to be set.

Example with 2MASS:

addmeta 2MASS datatype=char arraysize=17 width=17 ucd="meta.id;meta.main" desc="Src designation"
addmeta J-H datatype=double precision=2 width=5 unit=mag ucd="phot.color" desc="Color Jmag-Kmag"

The first line is not necessary (metadata already defined when we load the VOTable from VizieR) but the second one is mandatory since the column *J-H* is a synthetic column.
addmeta is also mandatory for non-synthetic columns if the data come from a file which does not contains any metadata (like CSV files).

2.9 Command *prefix*

Add the given prefix to the name of each column selected by *set cols*.

prefix *prefix*

Example with 2MASS:

prefix 2mass_

Will transform *RAJ2000* into *2mass_RAJ2000*, *Jmag* into *2mass_Jmag*, ...

2.10 Command *save*

Save the table into the given file at the given format.

save *filename* (votable|fits|csv)

Example with 2MASS:

save test/2mass.vot votable

Save the loaded 2MASS data as a VOTable into the file named *2mass.vot* in the sub-directory *test* of the current directory.

2.11 Command *cleartables*

Remove all tables: used to start a new script in the same file.

3 Generating synthetic tables

3.1 Commands overview

Here is the (small) list of commands available to generate and save synthetic tables.

synthetic *options*
? save *file_prefix* (fits|votable|csv)

When generating several synthetic tables, sources which are common in two or more tables have the same ID.

Each command is described in the next sections.

3.2 Command *synthetic*

The *synthetic* command generates one or several synthetic tables.

```
synthetic nTabs=INT seed=LONG preffix=true \
    geometry=GEOM geom_args \
    n[A-Z]=INT n[A-Z][B-Z]=INT n[A-Z][B-Z][C-Z]=INT ... \
    poserr[A-Z]type=ERR_TYPE \
    poserr[A-Z]mode=RAND_MODE rand_mode_args
```

Simple options (see next sections for the other options):

- *nTabs* - the number of synthetic tables to be generated
- *nA*, *nB*, ... - number of sources only in catalogue A, B, ... respectively
- *nAB*, *nAC*, ... - number of sources in common only in catalogues A and B, B and C, ... respectively
- *nABC*, ... - number of sources in common only in catalogues A and B and C, ...
- ...
- *seed* - ? seed used to initialize the random number generator
- *preffix* - ? add a preffix 'A_', 'B_', ... before tables ID column

Example for 3 tables:

```
synthetic nTabs=3 geometry=cone ra=22.51 dec=-68.94 r=0.25 \
    nA=1000 nB=2000 nC=300 \
    nAB=500 nAC=1500 nBC=780 \
    nABC=1200 \
    poserrAtype=CIRCLE poserrAmode=formula \
    paramA1="0.1+random()/2" \
    poserrBtype=RCD_DEC_ELLIPSE poserrBmode=function \
    paramB1func="0.8+log(x+0.1)/2" \
    paramB1xmin=0 paramB1xmax=2 paramB1ntep=100 \
    paramB2func="0.6+log(x+0.1)/2" \
    paramB2xmin=0 paramB2xmax=1.5 paramB1ntep=100 \
    poserrCtype=ELLIPSE poserrCmode=histogram \
    poserrCfile=sample.fits countC=counts \
    paramC1step=0.01 paramC1col=halfMaj \
    paramC2step=0.01 paramC2col=halfMin \
    paramC3step=1 paramC1col=posAng
```

3.2.1 Option *geometry=GEOM*

The *geometry* option define the region of the sky in which the positions of random sources are generated.

geometry=allsky Sources are uniformly distributed on the unit sphere. No additional option.

geometry=cone Sources are uniformly distributed on a given cone.

geometry=cone ra=DOUBLE dec=DOUBLE r=DOUBLE

- *ra* - right ascension of the cone center, in decimal degrees;
- *dec* - decliantion of the cone center, in decimal degrees;
- *r* - radius of the cone, in degrees.

Example:

geometry=cone ra=12.3647 dec=-68.4297 r=0.5

geometry=healpix WARNING: bug to be solved in HEALPix librairie!
Sources are uniformly distributed on the given HEALPix [2] cell:

geometry=healpix norder=INT ipix=LONG

- *norder* - norder of the HEALPix cell;
- *ipix* - index of the HEALPix cell.

Example:

geometry=healpix norder=3 ipix=542

geometry=moc WARNING: bug to be solved in HEALPix librairie!
Sources are uniformly distributed on the area of the unit sphere defined by the given MOC [3].

geometry=moc (moc=ASCII_MOC)|(mocfile=FILE)

- *moc* - the ASCII representation of a MOC;
- *mocfile* - name of the file (FITS, JSON, ...) containing the MOC.

Examples:

geometry=moc moc="1/1,3,4 2/4,25,12-14,21"
geometry=moc mocfile=mymoc.fits

3.2.2 Option *poserr[A-Z]type=ERR_TYPE*

The number of options in RAND_MODE depends on the error type. See §2.7.3 to obtain the liste of all possible error types.

3.2.3 Option *poserr[A-Z]mode=RAND_MODE*

The *poserr[A-Z]mode* defines the way the random positional error of sources of catalogue [A-Z] are computed.

poserr[A-Z]mode=formula Positional errors are computed from the given formula (wich may contains a call the the Java method *random()*):

poserr[A-Z]mode=formula param[A-Z][1-3]=EXP ...

Example with *poserrAtype=RCD_DEC_ELLIPSE*:

poserrAmode=formula paramA1="0.1+0.5*random()" paramA2=0.8

2. http://adsabs.harvard.edu/abs/2005ApJ...622..759G
3. http://ivoa.net/documents/MOC/

poserr[A-Z]mode=function Positional errors are randomly computed following the distribution of the given function (the software takes care of normalizing it to have a probability density function, pdf):

```
poserr[A-Z]mode=function
    param[A-Z][1-3]func=EXP
    param[A-Z][1-3]xmin=DOUBLE
    param[A-Z][1-3]xmax=DOUBLE
    param[A-Z][1-3]nstep=INT
```

in which:

- *param[A-Z][1-3]func* - the function (you must use the notation 'x' for the variable), MUST be always positive on the domain $[x_{\min}, x_{\max}]$ (if not, not a pdf after normalisation);
- *param[A-Z][1-3]xmin* - lower bound of the domain;
- *param[A-Z][1-3]xmax* - upper bound of the domain;
- *param[A-Z][1-3]nstep* - number of steps to be used in the numerical integration.

Example with *poserrAtype=CIRCLE*:

```
poserrAmode=function
    paramA1func="0.8+log(x+0.1)/2"
    paramA1xmin=0
    paramA1xmax=1.5
    paramA1nstep=100
```

Technically, the Trapezoidale rule is used to compute the numerical integrate I of the function f over the full domain $[x_{\min}, x_{\max}]$. Then, we pick a random number $r \in [0, 1[$. We use the Trapezoidale rule again from $x_0 = x_{\min}$ to $x_i = x_0 + ih$, with h which depends on the number n of steps $h = \frac{x_{\max} - x_{\min}}{n}$,such as:

$$I_{i-1} = \int_{x_0}^{x_{i-1}} f(x)\mathrm{d}x < rI \leq \int_{x_0}^{x_i} f(x)\mathrm{d}x = I_i$$

Then the random value choosen is $x_{\text{rand}} = x_{i-1} + h\frac{rI - I_{i-1}}{I_i - I_{i-1}}$.

poserrAmode=histogram Positional errors are randomly computed following the given histogram (possibly built on-the-fly) normalized to obtain a step pdf.

```
poserr[A-Z]mode=histogram
    poserr[A-Z]file=FILE
    param[A-Z][1-3]col=COLNAME|EXP
    param[A-Z][1-3]step=DOUBLE
    count[A-Z]=COLNAME|EXP
```

with:

- *poserr[A-Z]file* - file containing the histogram or the data to build the histogram;
- *param[A-Z][1-3]col* - expression or name of the column of the file containing the data of param[1-3];
- *param[A-Z][1-3]step* - step to be used for param[1-3];
- *count[A-Z]* - expression (e.g. 1) or name of the column of the file containing the counts (must return an integer).

Example with *poserrAtype=RCD_DEC_ELLIPSE*:

poserr[A-Z]mode=histogram
poserrAfile=histo.fits
paramA1step=0.01 paramA1col=0.5*(LOW_1+HIGH_1)
paramA1step=0.01 paramA1col=0.5*(LOW_2+HIGH_2)
countA=COUNT

Technically, we pick a random value $r \in [0, 1[$. We find a bin i such as $\sum_{i=0}^{i} count_i \leq r.count < \sum_{i=0}^{i+1} count_{i+1}$. Then, for each dimension d, we pick another random value $r_2 \in [0, step_d[$.

3.3 Command *save*

Save each generated table into a file having the given prefix and the given format.

save prefix=*fileprefix* suffix=*suffix* common=FILE format=(votable|fits|csv)

in which:

- *prefix* - prefix of the output files;
- *suffix* - suffix of the output files;
- *common* - ? name of the ouput file containing the exact positions before bluring;
- *format* - file format, votable, fits or csv.

The final name of the file will be: *prefix*[A-Z]*suffix*.
Example :

save prefix=test/synth suffix=.vot common=test/synthAll.vot format=votable

Save synthetic tables as VOTables into files named *synth[A-Z].vot* (synthAll.vot for the table containing unblurred positions) in the sub-directory *test* of the current directory.

4 Computing a density map

You can compute density maps of the tables you have built.
For this you can use the *densmap* command after having created a table – via commands *get*, *set pos*, ... (see §2) – or after having performed a xmatch (see §5).

4.1 Commands overview

densmap *algo* ... mode=[cone|allsky] [...] nthreads=[INT|max] norder=INT
save *filename* fits

4.2 Command *densamp*

Create a density map based on the HEALPix tessellation:

densmap *algo algo_options* mode=[cone|allsky] [*mode_options*]
nthreads=[INT|max] norder=INT

Common options:

- *nthreads=[INT|max]* - number of threads to be used to compute the density map:
  - *max* - automatically sets to the maximum number of threads available on the machine;
  - *INT* - from 1 to *max* (automatically set to *max* if greater than *max*).
- *norder=INT* - the wanted HEALPix norder (directly linked to the map resolution, see tab. 1).

Option *mode*:

- *mode=allsky* - for small allsky catalogues (be careful at the *norder* not to use too much memory).
- *mode=cone* - for a table which is the result of a cone search.
  - *autodetect* - automatically find the center and the radius of the cone;
  - *center* - can be an object name or a position, e.g. *center="M 31"* or *center="12.345678 +87.654321"*;
  - *radius* - radius of the cone search, in one of the 3 possible units (deg, arcmin or arcsec), e.g. *radius=2.4arcmin*.

Example: see §4.4.

norder	$N_{pix} = 12 \times 2^{norder}$	$\theta_{pix} = \Omega_{pix}^{1/2}$
0	12	58°.6
1	48	29°.3
2	192	14°.7
3	768	7°.33
4	3072	3°.66
5	12 288	1°.83
6	49 152	55′.0
7	196 608	27′.5
8	786 432	13′.7
9	3 145 728	6′.87
10	12 582 912	3′44
11	50 331 648	1′72
12	201 326 592	51”.5
13	805 306 368	25”.8
14	3.22×10^{9}	12”.9
15	1.29×10^{10}	6”.44
16	5.15×10^{10}	3”.22
17	2.06×10^{11}	1”.61
⋮	⋮	⋮
29	3.46×10^{18}	$3”.93 \times 10^{-4}$

Table 1: Number of pixel on the full sky and approximative pixel side size according to the HEALPix norder (see tab. 1 of Górski et al. (2004), astro-ph/0409513).

The next section explains the different algorithms available.

4.2.1 *densmap knn*

Computes the local density by k-nearest neighbour averaging.

densmap knn k=INT ...

Options:

– *k*: number of nearest neighbours used to compute the local density

Example: see §4.4.

4.2.2 *densmap kernsmooth*

Computes the local density by fixed bandwidth kernel smoothing (using the Epanechnikov function).

densmap knn h=REAL[deg|arcmin|arcsec] ...

Options:

– *h*: kernel bandwidth, in one of the 3 possible units (deg, arcmin or arcsec).

Example: see §4.4.

4.2.3 *densmap balloon*

Computes the local density by balloon estimator: variable bandwidth kernel smoothing in which the bandwidth of an estimation is here the distance to the k^{th} nearest neighbour.

densmap balloon k=INT ...

Options:

– *k*: number of nearest neighbours used to compute the bandwidth of the kernel smoothing

Example: see §4.4.

4.2.4 *densmap samplepoint*

Computes the local density by sample point density estimator: variable kernel smoothing in which each object bandwidth is the distance to the k^{th} nearest neighbour.

densmap samplepoint k=INT ...

Options:

– *k*: number of nearest neighbours used to compute the local density

Example: see §4.4.

4.3 Command *save*

Save the density map into the given file at the given format (only *fits* allowed so far).

save *filename* fits

Example:

```
save usnob1.densmap.samplepoint50.auto.fits fits
```

4.4 Example script and results

The following script computes 8 density maps on USNO-B1.0 data.

```
# Load table
get VizieRLoader tabname=I/284/out mode=cone center="12.256460 +20.878920"
    radius=10.2arcmin allcolumns
cache usnob1.viz.cone.12d256460p20d878920.10d2arcmin.allcols.vot votable
# Build table
where !Flags.matches(".s.")
set pos ra=RAJ2000 dec=DEJ2000
save usnob1.vot votable
# Compute and save 8 density maps
densmap knn k=50 mode=cone center="12.256460 +20.878920" radius=10.2arcmin
        nthreads=max norder=16
save usnob1.densmap.knn50.noauto.fits fits
densmap knn k=50 mode=cone autodetect nthreads=max norder=16
save usnob1.densmap.knn50.auto.fits fits
densmap balloon k=50 mode=cone center="12.256460 +20.878920"
        radius=10.2arcmin nthreads=max norder=16
save usnob1.densmap.balloon50.noauto.fits fits
densmap balloon k=50 mode=cone autodetect nthreads=max norder=16
save usnob1.densmap.balloon50.auto.fits fits
densmap kernsmooth h=3arcmin mode=cone center="12.256460 +20.878920"
        radius=10.2arcmin nthreads=max norder=16
save usnob1.densmap.kernsmooth3arcmin.noauto.fits fits
```

```
densmap kernsmooth h=3arcmin mode=cone autodetect nthreads=max norder=16
save usnob1.densmap.kernsmooth3arcmin.auto.fits fits
densmap samplepoint k=50 mode=cone center="12.256460 +20.878920"
        radius=10.2arcmin nthreads=max norder=16
save usnob1.densmap.samplepoint50.noauto.fits fits
densmap samplepoint k=50 mode=cone autodetect nthreads=max norder=16
save usnob1.densmap.samplepoint50.auto.fits fits
```

The results for 4 density maps are displayed on fig. 1.

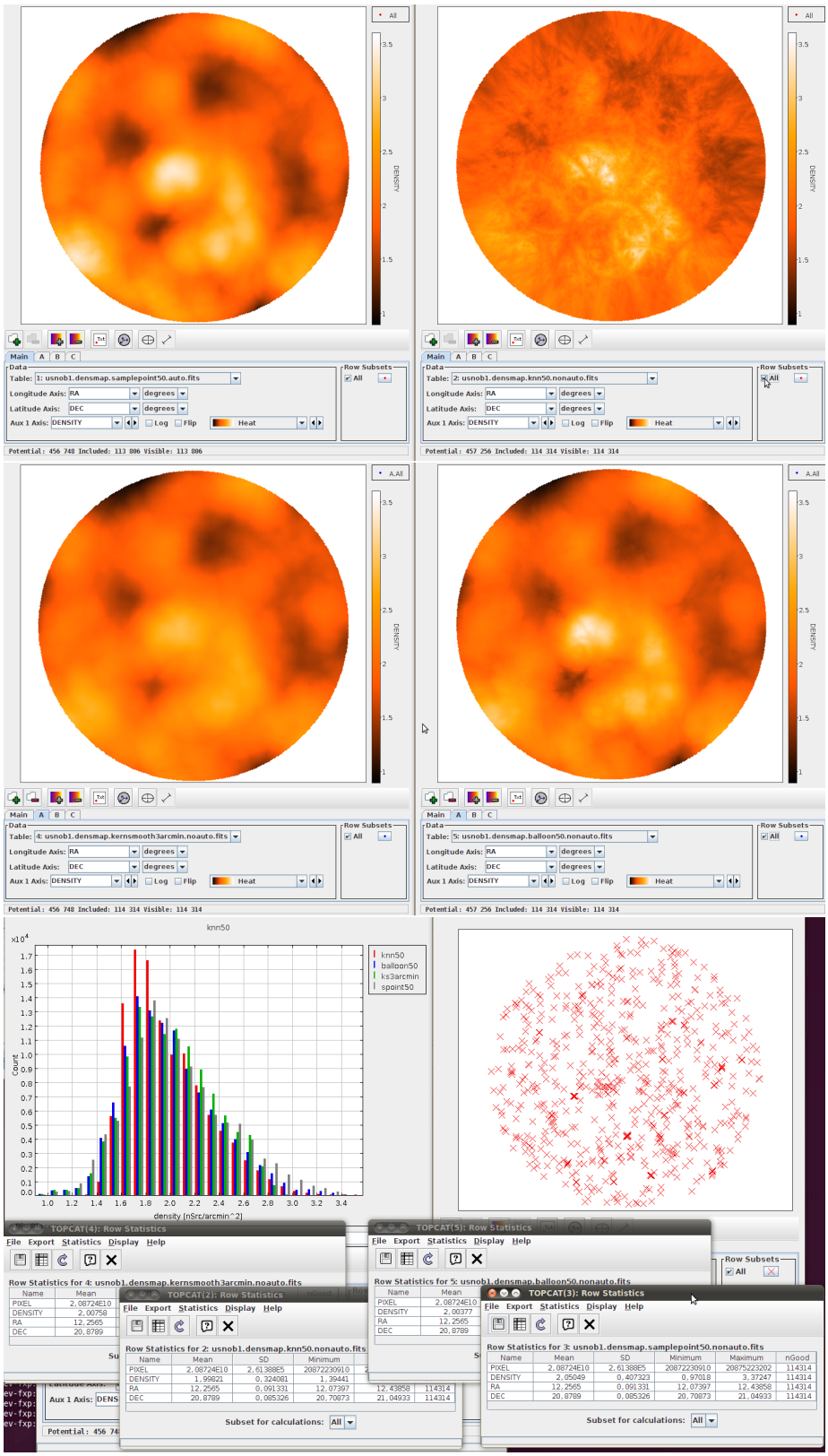

Figure 1 – Visual results of the above script in TOPCAT. Top left: sample point estimator. Top right: knn. Middle left: fixed bandwidth kernel smoothing. Middle right: balloon estimator. Bottom left: density histograms in $arcmin^{-2}$. Bottom right: point distribution on the sky.

5 Matching two tables

The xmatch part consists in building a new table by cross-comparing the positions of the objects of the two input tables. It can be seen as a SQL join in which the join condition is a positional criteria. Compared to SQL, we have added join operators [4]. As explained in §1.2, we also have to merge the stable columns.

5.1 Commands overview

? switchtables
xmatch *algo algo_options* join=*join_type*
? addconstraint *condition*
? merger toepoch *type* ? merger pos *type*
? merger dist *type*
? merger chi2pos *type*
? merger epoch *type*
? merger poserr *type*
? merger poserrepoch *type*
? merger pm *type*
? merger pmerr *type*
? merger extent *type*
? addcol *type options*
? addcols *type options*
? save *filename* (fits|votable) ? buildmoc *mocname* ? save *MOCfilename* (fits|json)

Example:

xmatch chi2 nStep=1 nMax=1 completeness=0.997 join=inner
merger pos ML
merger pmerr right

5.2 Command *switchtables*

To be used when only two tables are loaded. Switch the two input tables so the right table becomes the left table and the left table becomes the right table.

5.3 Command *xmatch*

The command you are waiting for ;)

xmatch *algo algo_options* join=*join_type*

The next sections explain the possible algorithms (*algo*) and their options, while §6.4.1 details the available join types (*join_type*).

5.3.1 *xmatch knn*

Performs a simple *k*-nearest neighbours xmatch.

xmatch knn k=... dMax=... join=*join_type*

4. the result of those new operators can also be obtained in SQL from its 4 joins (full, left, right, inner) plus a constraint or from a more complex query

Options:
- *k=INTEGER* - maximum number of nearest neighbours;
- *dMax=DOUBLE* - distance upper limit, in arcseconds (can be *Double.MAX_VALUE*, see Java documentation).

Example:

```
xmatch knn k=2 dMax=4.25 join=left
```

5.3.2 *xmatch knn_kd*

Same as *xmatch knn* (§5.3.1) but using a *kd*-tree instead of a *M*-tree. The main interest consists in comparing performances and cross-validating both methods.

5.3.3 *xmatch knn_rpm*

Same as *xmatch knn* (§5.3.1) but taking into account the right table proper motions.
WARNING: you must have set the proper motion parameters in the right table and the epoch in both catalogues (see §2.7).
WARNING: after the xmatch, the right sources positions are no more the original ones but the ones computed at epoch the mean of the left sources epoches.

5.3.4 *xmatch knn_bpm*

Same as *xmatch knn* (§5.3.1) but taking into account both tables proper motions.
WARNING: you must have set the proper motions and the epochs in both catalogues (see §2.7).

5.3.5 *xmatch cone*

Performs a simple cone-search xmatch.

```
xmatch cone dMax=... join=join_type
```

Options:
- *dMax=DOUBLE* - distance upper limit, in arcseconds.

Example:

```
xmatch cone dMax=4.25 join=full
```

5.3.6 *xmatch cone_kd*

Same as *xmatch cone* (§5.3.5) but using a *kd*-tree instead of a *M*-tree. The main interest consists in comparing performances and cross-validating both methods.

5.3.7 *xmatch cone_lpm*

Same as *xmatch cone* (§5.3.5) but taking into account the left table proper motions.
WARNING: you must have set the proper motion parameters in the left table and the epoch in both catalogues (see §2.7).

5.3.8 *xmatch cone_rpm*

Same as *xmatch cone* (§5.3.5) but taking into account the right table proper motions.
WARNING: you must have set the proper motion parameters in the right table and the epoch in both catalogues (see §2.7).

5.3.9 *xmatch cone_bpm*

Performs a simple cone-search xmatch taking into account both catalogues proper motions.

xmatch cone_bpm dMax=... epoch=... join=*join_type*

Options:
- *dMax=DOUBLE* - distance upper limit, in arcseconds.
- *epoch=DOUBLE* - epoch at which positions are computed to perform the xmatch.

Example:

xmatch cone_bpm dMax=4.25 epoch=2000.0 join=full

WARNING: you must have set the proper motions and the epochs in both catalogues (see §2.7).

5.3.10 *xmatch chi2*

Performs a chi2 xmatch (the xmatch you are probably waiting for :).

xmatch chi2 nStep=... nMax=... completeness=... join=*join_type*

Options:
- *nStep=INT* - number of xmatch done using chi2 (=1 at first xmatch);
- *nMax=INT* - total number of xmatches you plan to perform using chi2;
- *completeness=DOUBLE* - wanted chi2 completeness (must not be highest than at previous steps).

Example:

xmatch chi2 nStep=2 nMax=3 completeness=0.9973 join=inner

You are here performing the second chi2 xmatch out of 3. A completeness of 0.9973 is equivalent to the 3σ criteria in 1 dimension.
WARNING: you must have set the positional errors on both catalogues (see §2.7).

5.3.11 *xmatch proba2_v1*

Performs a chi2 xmatch between two catalogues and computes probabilities for each association. The computed probabilities are based on positional coincidence only and take into account the sky density of sources.

xmatch proba2_v1 completeness=... area=...

Options:
- *completeness=DOUBLE* - wanted chi2 completeness
- *area=DOUBLE* - the sky surface area of the two tables, in radians^{-2}, e.g. the cone surface for a cone search.

Example (for a cone of radius=8.0arcmin):

xmatch proba2_v1 completeness=0.9973 area=0.0000170

Remark: so far, you can choose neither the JOIN type (INNER) nor mergers (a default merger is used). It does not yet take into account proper motions.
WARNING: this version

- Assumes that the positional errors are correct, i.e. neither underestimated not overestimated, so that the χ distribution really follows the Rayleigh distribution.
- Does not care about multi-candidates so the probabilities are symmetric (A x B probabilities = B x A probabilities): in other words, each match are treated individually.

5.3.12 *xmatch proba2_v2*

Idem as *proba2_v1* but we take into account the fact that the first catalogue sources can have multiple candidates, and only one candidate can be the true counterpart.
WARNING: the probabilities are no more symmetric!

5.3.13 *xmatch proba2_v3*

Performs a chi2 xmatch between two catalogues and computes probabilities for each association. The computed probabilities are based on positional coincidence only and take into account local sky densities of sources.
This version of the algorithm computes density maps to compute local priors.

xmatch proba2_v3 completeness=... densmethod=(knn|kernsmooth|balloon|samplepoint) *densmap_params*

Options:

- *completeness=DOUBLE* - wanted chi2 completeness
- *densmethod=STRING* - the method used to compute density maps (*algo* in §4.2)
- *densmap_params* - parameters of the density map method (see §4.2)

Example:

xmatch proba1_v3 completeness=0.9973 densmethod=kernsmooth h=6arcmin
norder=14 mode=cone center="30.9253 -7.78128" radius=15.0arcmin nthreads=max

5.3.14 *xmatch chi2_lpm*

Same as *xmatch chi2* (§5.3.10) but taking into account the left table proper motions.
WARNING: you must have set the proper motion – possibly proper motion error and positional error epoch – parameters in the left table and the epoch in both catalogues (see §2.7).

5.3.15 *xmatch chi2_rpm*

Same as *xmatch chi2* (§5.3.10) but taking into account the right table proper motions.
WARNING: you must have set the proper motion – possibly proper motion error and positional error epoch – parameters in the right table and the epoch in both catalogues (see §2.7).

5.3.16 *xmatch chi2_bpm*

Performs a chi2 xmatch taking into account both tables proper motions.

xmatch chi2_bpm nStep=... nMax=... completeness=... epoch=... join=*join_type*

Options:

- *nStep=INT* - number of xmatch done using chi2 (=1 at first xmatch);
- *nMax=INT* - total number of xmatches you plan to perform using chi2;
- *completeness=DOUBLE* - wanted chi2 completeness;
- *epoch=DOUBLE* - epoch at which the positions and their errors are computed to perform the xmatch.

Example:

```
xmatch chi2 nStep=2 nMax=3 completeness=0.9973 epoch=2000.0 join=inner
```

WARNING: you must have set the proper motions – possibly proper motion errors and positional errors epoch – and the epochs in both catalogues (see §2.7).

5.3.17 *xmatch ext_l*

Performs a xmatch associating objects of the right table which overlap the left table extended objects.

```
xmatch ext_l k=... join=join_type
```

Options:

- *k=DOUBLE* - ellipse extension factor (most of the time = 1), i.e. radius of the matching ellipse considering the input extension has a radius of 1.

Example:

```
xmatch ext_l k=1.2 join=left_bar
```

Command used if you want to remove from the right catalogue the sources intersecting the left catalogue extended sources, considering 1.2 times their extension.
WARNING: you must have set the extension on the left catalogue (see §2.7).

5.3.18 *xmatch ext_r*

Performs a xmatch associating objects of the left table which overlap the right table extended objects.

```
xmatch ext_r k=... join=join_type
```

Options:

- *k=DOUBLE* - ellipse extension factor (most of the time = 1), i.e. radius of the matching ellipse considering the input extension has a radius of 1.

Example:

```
xmatch ext_r k=1.2 join=right_bar
```

Command used if you want to remove from the left catalogue the sources intersecting the right catalogue extended sources, considering 1.2 time their extension.
WARNING: you must have set the extension on the right catalogue (see §2.7).

5.3.19 *xmatch ext_b*

Performs a xmatch associating overlapping objects of both catalogues.

```
xmatch ext_b kl=... kr=... join=join_type
```

Options:

- *kl=DOUBLE* - left ellipse extension factor (most of the time = 1), i.e. radius of the matching ellipse considering the input extension has a radius of 1;

– *kr=DOUBLE* - right ellipse extension factor (most of the time = 1), i.e. radius of the matching ellipse considering the input extension has a radius of 1.

Example:

```
xmatch ext_r kl=1.2 kr=1.3 join=inner
```

WARNING: you must have set the extension on both catalogues (see §2.7).

5.4 Command *xmatch*, option *join*

To illustrate the different join possibilities, let us define two tables (tab. 2). The matching positions of the left and the right tables have the same name.

src_id	position
left_1	pos_a
left_2	pos_b
left_3	pos_c

src_id	position
right_1	pos_d
right_2	pos_b
right_3	pos_b
right_4	pos_c

Table 2: Left and right tables to be joined based on their positions.

For a definition of the SQL joins, you can also have a look here:
– inner join: http://www.w3schools.com/sql/sql_join_inner.asp
– left join: http://www.w3schools.com/sql/sql_join_left.asp
– right join: http://www.w3schools.com/sql/sql_join_right.asp
– full join: http://www.w3schools.com/sql/sql_join_full.asp

5.4.1 *join=inner*

The classical xmatch join : returns only the matching sources from the left and right catalogues (defined in Tab. 2).

src_id_left	position	src_id_right
left_2	pos_b	right_2
left_2	pos_b	right_3
left_3	pos_c	right_4

Table 3: Result of left table INNER JOIN right table

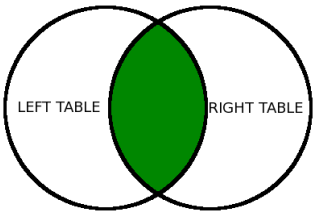

Figure 2 – Venn diagram of the INNER JOIN

5.4.2 *join=left*

Returns all sources from the left catalogue, with the matching sources in the right catalogue (defined in Tab. 2).

src_id_left	position	src_id_right
left_1	pos_a	null
left_2	pos_b	right_2
left_2	pos_b	right_3
left_3	pos_c	right_4

Table 4: Result of left table LEFT JOIN right table

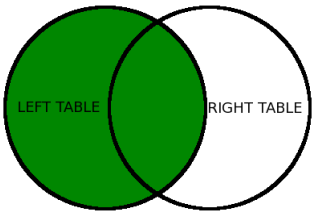

Figure 3 – Venn diagram of the LEFT JOIN

5.4.3 *join=right*

Returns all sources from the right catalogue, with the matching sources in the left catalogue (defined in Tab. 2).

src_id_left	position	src_id_right
left_2	pos_b	right_2
left_2	pos_b	right_3
left_3	pos_c	right_4
null	pos_d	right_1

Table 5: Result of left table RIGHT JOIN right table

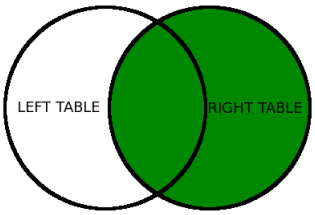

Figure 4 – Venn diagram of the RIGHT JOIN

5.4.4 *join=full*

Returns all sources from the left and from the right catalogues (defined in Tab. 2).

src_id_left	position	src_id_right
left_1	pos_a	null
left_2	pos_b	right_2
left_2	pos_b	right_3
left_3	pos_c	right_4
null	pos_d	right_1

Table 6: Result of left table FULL JOIN right table

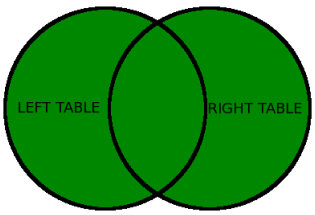

Figure 5 – Venn diagram of the FULL JOIN

5.4.5 *join=lleft*

Returns all sources from the left catalogue plus the sources of the left catalogue with the matching sources in the right catalogue (defined in Tab. 2).

src_id_left	position	src_id_right
left_1	pos_a	null
left_2	pos_b	null
left_2	pos_b	right_2
left_2	pos_b	right_3
left_3	pos_c	null
left_3	pos_c	right_4

Table 7: Result of left table LLEFT JOIN right table

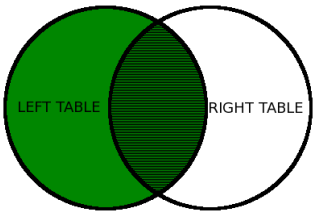

Figure 6 – Venn diagram of the LLEFT JOIN

5.4.6 *join=rright*

Returns all sources from the right catalogue plus the sources of the right catalogue with the matching sources in the left catalogue (defined in Tab. 2).

src_id_left	position	src_id_right
null	pos_b	right_2
left_2	pos_b	right_2
null	pos_b	right_3
left_2	pos_b	right_3
null	pos_c	right_4
left_3	pos_c	right_4
null	pos_d	right_1

Table 8: Result of left table RRIGHT JOIN right table

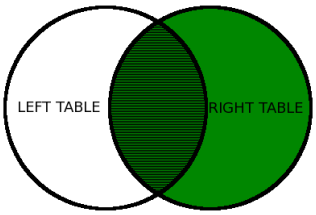

Figure 7 – Venn diagram of the RRIGHT JOIN

5.4.7 *join=ffull*

Returns all sources from the left and from the right catalogues plus the matching sources of both catalogues (defined in Tab. 2).

src_id_left	position	src_id_right
left_1	pos_a	null
left_2	pos_b	null
null	pos_b	right_2
left_2	pos_b	right_2
null	pos_b	right_3
left_2	pos_b	right_3
left_3	pos_c	null
null	pos_c	right_4
left_3	pos_c	right_4
null	pos_d	right_1

Table 9: Result of left table FFULL JOIN right table

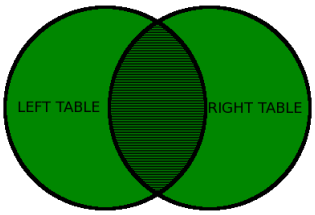

Figure 8 – Venn diagram of the FFULL JOIN

5.4.8 *join=inner_bar*

Returns the result of the full join minus the result of the inner join (defined in Tab. 2).
Said differently, its returns all sources from both the left and the right catalogue which do not match.

src_id_left	position	src_id_right
left_1	pos_a	null
null	pos_d	right_1

Table 10: Result of left table INNER_BAR JOIN right table

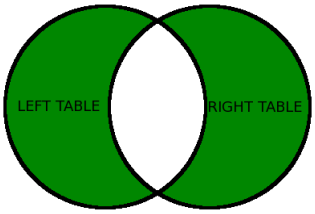

Figure 9 – Venn diagram of the INNER_BAR JOIN

5.4.9 *join=left_bar*

Returns the result of the full join minus the result of the left join. Columns of the left catalogue are then removed (since they are all null) (defined in Tab. 2).
Said differently, it returns all sources from the right catalogue which do not match any of the left catalogue sources.

position	src_id_right
pos_d	right_1

Table 11: Result of left table LEFT_BAR JOIN right table

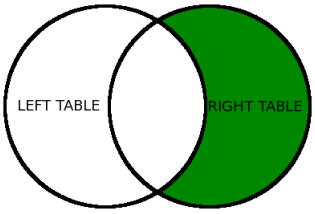

Figure 10 – Venn diagram of the LEFT_BAR JOIN

5.4.10 *join=right_bar*

Returns the result of the full join minus the result of the right join. Columns of the right catalogue are then removed (since they are all null) (defined in Tab. 2).
Said differently, it returns all sources from the left catalogue which do not match any of the right catalogue sources.

src_id_left	position
left_1	pos_a

Table 12: Result of left table RIGHT_BAR JOIN right table

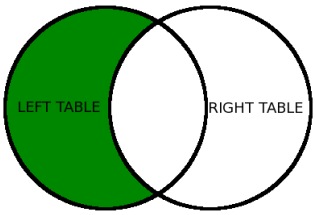

Figure 11 – Venn diagram of the RIGHT_BAR JOIN

5.5 Command *addconstraint*

Add a condition which must be satisfied to associate two objects during the xmatch.

addconstraint *condition*

Example:

addconstraint (sdss9.r - 2mass.Jmag) > -2 && (sdss9.r - 2mass.Jmag < 2)

5.6 Command *merge*

The merge command is used to invoke mergers. Mergers are used to set the "stable" columns (see §1.2) of the new table resulting from a xmatch. A merger can set one or several columns at the same time. For example, the *merge pos LM* sets the positions, the positional errors, the value of the positional chi2 and the number of sources matched by chi2; while *merge pos left* only sets the positions.

The general syntax is:

merge *label algo*

The next section details the available labels and, for each label, the algorithms you can choose.

5.6.1 *merger toepoch*

This merger is special and has to be used when you perform xmatches taking into account proper motions. It does not directly set columns but tells at which epoch the astrometry of sources having a proper motion has to be computed before being merged.

merge toepoch *type*

in which *type* can be:

- org or o - keep original data (do not take into account proper motion);
- left or l - compute the astrometry of the right source (which must has a proper motion) at the epoch of the left source;

– right or r - compute the astrometry of the left source (which must has a proper motion) at the epoch of the right source;
– bothEPOCH or bEPOCH - compute the astrometry of both sources (which must have proper motions) at the given epoch EPOCH (e.g. EPOCH=2000.0).

Example:

merge toepoch both1995.42

5.6.2 *merge pos*

Merges at least the positions.

merge pos *algo*

in which *algo* can be:

– left or l - keep left table positions;
– right or r - keep the right table positions;
– mean or m - the new position is the centroid of the left and right table positions. It also sets *dMEC* with the distance between the left and right positions.
– MEC or mec - new positions are the center of the minimum enclosing cone (computed taking into account all the positions of the previous xmatched tables). It also sets *dMEC* with the value of the diameter of the minimum enclosing cone (MEC).
– besterr - sets the positions of the sources having the smallest positional uncertainties, comparing the semi-major axis (the two xmatched tables must have positional errors). It also sets the new positional errors (the minimum ones).
– ML or ml or chi2 - new positions are the one estimated by a maximum likelihood, based on the individual positional errors. It means that to use this merger both left and right table should have positional errors. The merger also sets the new positional errors (error on the ML estimate), *nPos* and *chi2Pos*.

WARNING: issue to be solved with *ML* and *besterr*: always sets the error epoch even when there is no such information in both the right and the left tables.

5.6.3 *merge epoch*

Sets the epoch.

merge epoch *algo*

in which *algo* can be:

– left or l - keep left value;
– right or r - keep right value;
– preferLeft or lr - keep left value, or right value if left value is null;
– preferRight or rl - keep right value, or left value if right value is null;
– highest or h - keep the highest value;
– smallest or s - keep the smallest value;
– mean or m - keep the mean between the left and the right value.

5.6.4 *merge poserr*

Merges the positional error. If the positional error also has an epoch, use *merge poserrepoch* (§5.6.5).

merge poserr *type*

in which *type* can be:
- left or l - keep left value;
- right or r - keep right value;

Remark: to actually merge positional errors, see *merge pos LM* in §5.6.2.

5.6.5 *merge poserrepoch*

Merges the positional error and its epoch:

merge poserrepoch *type*

in which *type* can be:
- left or l - keep left value;
- right or r - keep right value;

Remark: to actually merge positional errors, see *merge pos LM* in §5.6.2.

5.6.6 *merger pm*

Merges the proper motion (if you have informations about the error, use *merger pmerr* §5.6.7)

merge pm *algo*

in which *algo* can be:
- left or l - keep left value;
- right or r - keep right value;
- preferLeft or lr - keep left value, or right value if left value is null;
- preferRight or rl - keep right value, or left value if right value is null;

5.6.7 *merger pmerr*

Merges the proper motion and associated errors.

merge pmerr *algo*

in which *algo* can be:
- left or l - keep left value;
- right or r - keep right value;
- preferLeft or lr - keep left value, or right value if left value is null;
- preferRight or rl - keep right value, or left value if right value is null;
- besterr - if both left and right table have PM with error, keep the one having the smallest error;

5.6.8 *merger extent*

Merges the source's extensions.

merge ext *algo*

- left or l - keep left value;
- right or r - keep right value;
- preferLeft or lr - keep left value, or right value if left value is null;
- preferRight or rl - keep right value, or left value if right value is null;
- smallest or s - if both left and right sources have an extension, keep the one having the smallest semi-major axis;
- highest or h - if both left and right sources have an extension, keep the one having the highest semi-major axis.

5.6.9 *merger dist*

Merges the distances *dMEC*:

merge dist *type*

in which *type* can be:
- left or l - keep left value;
- right or r - keep right value;
- l2r - compute a new value which is the distance from the left position to the right position;
- MEC or mec - compute the diameter of the minimum enclosing cone (taking into account all sources which have been merged during the successive xmatches).

Remarks:
- DON'T USE *merger dist* WHEN you use:
  - *merge pos mean*, see §5.6.2;
  - *merge pos ML*, see §5.6.2.
- using MEC during intermediate xmatches may be useless as the new values takes into account previous positions. It is useful only if you save intermediary results.
- DON'T USE MEC when you have xmatches with proper motion because the positions used to compute the MEC do not take into account proper motions!

5.6.10 *merge chi2pos*

Merges the *chi2Pos* and *nPos* value:

merge chi2pos *type*

in which *type* can be:
- left or l - keep left value;
- right or r - keep right value;

Remark: DON'T USE *merger chi2pos* WHEN you use *merger pos LM*, see §5.6.2.

5.7 Command *addconstraint*

The command *addconstraint* is similar to the *where* command (see §2.5) but applies on the result of a xmatch.
It select the rows matching the given *expression*:

addconstraint *expression*

Example with SDSS and 2MASS:

addconstraint sdss9.rmag - 2mass.Jmag < 3

5.8 Command *addcol*

The command *addcol* add a new column to the output table.

addcol *type* name=... ?append *options*

With:
- name - the name to be given to the column
- append - put the column at the end of the table, hence put it just after 'stable' columns.

In which *type* can be:

- moc moc=*mocname*: add a column telling if the position is out of (0) is in (1) or is on the border (2) of the given moc (you must have previously generated the moc using the *buildmoc* command)
- maxProba: store the highest of all probabilities (algo probaN_v1 only)
- maxProbaLabel: store the label of the columns which contains the highest of all probabilities (algo probaN_v1 only)
- limitmag: To BE DOCUMENTED

Example in one script:

```
...
buildmoc name=SDSS9_MOC ra=RAJ2000 dec=DEJ2000 norder=11
...
addcol moc moc=SDSS9_MOC name=isSDSSCovered
addcol maxProba name=maxProba
addcol maxProbaLabel name=maxProbaLabel
...
```

5.9 Command *addcols*

Similar to *addcol* except that it adds several new columns.

addcols *type options* name=... ?append *options*

In which *type options* can be:

- maxProbas prefix=PREFIX n=N: generates $2N$ columns storing the N highest probabilities and their N associated labels (algo probaN_v1 only)
- sumProbas1* n=N: N is the number of catalogues, generates $N-1$ columns containing the marginalized probabilties of association with the source of the first catalogue (specific use case!)

5.10 Command *save*

Save the table into the given file at the given format.

save *filename* (votable|fits|csv)

Example with 2MASS:

```
save test/2mass.vot votable
```

Save the loaded 2MASS data as a VOTable into the file named *2mass.vot* in the sub-directory *test* of the current directory.

5.11 Command *buildmoc*

See §2.6

6 Matching multiple tables computing probabilities

This section describes how to cross-match multiple tables if one want to compute probabilities of identification taking into account a full set of catalogues: In this case, the xmatch process cannot be iterative any longer.

6.1 Principle

The structure of a script as presented in §1.1 is slightly modified.
Here, you have first to load *n* tables and then to xmatch them. The logical structure of such a script is then:

```
create table_1
create table_2
create table_3
...
xmatch table_1 versus table_2 versus table_3 versus ... ...
```

WARNING: in ouput, positions and associated errors are estimated by maximum likelihood using candidates from each catalogue, even if some or all probabilities of associations are low.

6.2 *xmatch proba3_v1*

Performs a χ^2 xmatch between three catalogues and computes probabilities for each association.
Usage:

```
xmatch proba3_v1 completeness=... area=...
```

Options:

- *completeness=DOUBLE* - wanted chi2 completeness
- *area=DOUBLE* - the sky surface area of the tables, in radians^{-2}, e.g. the cone surface for a cone search.

Info:

- probabilities are based on the final χ value, not on χ_1 and χ_2 ($\chi^2 = \chi_1^2 + \chi_2^2$)
- probabilities are based on a model (no systematics, errors neither underestimated nor overestimated, ...), verify you data fit the model!

In the output file, probabilities are:

- *probaABC*: probability the 3 associated sources have to be from a same 'true' source;
- *probaAB_C*: probability sources A and B have to be from a same 'true' source and C is from a different 'true' source;
- *probaAC_B*: probability sources A and C have to be from a same 'true' source and B is from a different 'true' source;
- *probaBC_A*: probability sources B and C have to be from a same 'true' source and A is from a different 'true' source;
- *probaA_B_C*: probability the 3 sources have to be from 3 different 'real' sources.
- $probaAB = probaABC + probaAB\_C$
- $probaAC = probaABC + probaAC\_B$
- $probaBC = probaABC + probaBC\_A$

6.3 *xmatch proba4_v1*

Same as *proba3_v1* for four catalogues, but here probabilities are based on χ_1, χ_2 and χ_3.
In the output file, probabilities are:

- *probaABCD*: the 4 associated sources are from a same 'true' source;
- *probaABC_D*: A, B and C from a same 'true' source, D from a different 'true' source
- *probaABD_C*: A, B and D from a same 'true' source, C from a different 'true' source
- *probaACD_B*: A, C and D from a same 'true' source, B from a different 'true' source
- *probaBCD_A*: B, C and D from a same 'true' source, A from a different 'true' source
- *probaAB_C_D*: A and B from a same 'true' source, C and D from 2 other different 'true' sources
- *probaAC_B_D*: A and C from a same 'true' source, B and D from 2 other different 'true' sources
- *probaAD_B_C*: A and D from a same 'true' source, B and C from 2 other different 'true' sources
- *probaBC_A_D*: B and D from a same 'true' source, A and D from 2 other different 'true' sources
- *probaBD_A_C*: B and C from a same 'true' source, A and C from 2 other different 'true' sources
- *probaCD_A_B*: C and B from a same 'true' source, A and B from 2 other different 'true' sources
- *probaAB_CD*: A and B from a same 'true' source, C and D from another 'true' sources
- *probaAC_BD*: A and C from a same 'true' source, B and D from another 'true' sources
- *probaAD_BC*: A and D from a same 'true' source, B and C from another 'true' sources
- *probaA_B_C_D*: the 4 sources are from 4 different 'true' sources
- *probaAB = probaABCD + probaABC_D + probaABD_C + probaAB_C_D + probaAB_CD*
- *probaAC = probaABCD + probaABC_D + probaACD_B + probaAC_B_D + probaAC_BD*
- *probaAD = probaABCD + probaABD_C + probaACD_B + probaAD_B_C + probaAD_BC*
- *probaBC = probaABCD + probaABC_D + probaBCD_A + probaBC_A_D + probaAD_BC*
- *probaBD = probaABCD + probaABD_C + probaBCD_A + probaBD_A_C + probaAC_BD*
- *probaCD = probaABCD + probaACD_B + probaBCD_A + probaCD_A_B + probaAB_CD*

6.4 *xmatch probaN_v1*

Performs a χ^2 xmatch between N catalogues and compute probabilities for each association.

Usage:

xmatch probaN_v1 completeness=... joins=... area=... ?meth=median

Options:

- *completeness=DOUBLE* - wanted chi2 completeness
- *area=DOUBLE* - the sky surface area of the tables, in radians^{-2}, e.g. the cone surface for a cone search
- *joins=*`[ILRFJMSG][ILRFJMSG()]*` - define the way the successive tables are joined (for more information, see §6.4.1)
- *? meth=median* - use the mediane instead of the mean to estimate priors (we recall that the mediane is a robust estimation of the mean so it is much less sensitive to outliers)
- *? keep=largestnid* - to be used with option *ids*, if the set of not-null identifiers of a row is a subset of the not-null identifiers of another rows, remove it
- *? ids=ID_A_COLNAME,ID_B_COLNAME,...* - the list of IDs to take into account for option *keep=largestnid*

In the output file the number of probabilities you obtain is equal to:

$$n_{proba} = \sum_{K=2}^{N} C_N^K B(K) \tag{1}$$

In which $C_N^K = \frac{N!}{K!(N-K)!}$ (K-combination) is the number of possible xmatches involving K catalogues among N and $B(K)$ is the Bell number with reprensents the number of possible partitions of a set of K sources.

$$B(K) = \sum_{p=0}^{K-1} C_{N-1}^p B(p-1) \tag{2}$$

Example:

xmatch probaN_v1 joins=I(F) completeness=0.9973 area=0.00003828

6.4.1 Parameter *joins*

You can use 4 differents letters which are associated to the four basic join types:

- *I*: means Inner join, see §5.4.1;
- *L*: means Left join, see §5.4.2;
- *R*: means Right join, see §5.4.3;
- *F*: means Full join, see §5.4.4;
- *J*: means IInner join, see §**??**;
- *M*: means LLeft join, see §5.4.5;
- *S*: means RRight join, see §5.4.6;
- *G*: means FFull join, see §5.4.7.

Operations are made from left to right except if you use parentheses '(' and ')'.
Examples:

- *joins=I(L)* ⇔ 1 I (2 L 3): performs the inner join between table 1 and the result of the left join between table 2 and table 3;
- *joins=IF* ⇔ (1 I 2) F 3: performs the full join between the result of the inner join of table 1 with table 2 and table 3
- *joins=I(FF)L* ⇔ (1 I (2 F 3 F 4)) L 5 ⇔ tab1 INNER JOIN (tab2 FULL JOIN tab3 FULL JOIN tab4) LEFT JOIN tab5

7 One example script (more to come)

This script makes the union (full join) of the SDSS and the 2MASS sources, xmatching them with a chi2 xmatch. The result is then xmatched with the USNO-B.1 (inner join). So each output row contains one of the three possibilities:

1. the match of one USNO-B.1 and one 2MASS source;
2. the match of one USNO-B.1 and one SDSS source;
3. the match of one USNO-B.1, one 2MASS and one SDSS source.

Remarks:

- This script does not take into account proper motions, even if it should to perform an accurate xmatch with the USNO-B.1 (in that case, we should have started by xmatching the USNOB-1.0).
- We add here to the USNO-B.1 positional errors a systematic of 0.13 arcsecond; don't take this value too seriously.

```
#### Load table 1 ####
get VizieRLoader tabname=V/139/sdss9 mode=cone center="12.256460 +20.878920" radius=10arcmin allcolumns
where mode==1 && Q==3
set pos ra=RAJ2000 dec=DEJ2000
set epoch year=ObsDate
set poserr type=RCD_DEC_ELLIPSE param1=e_RAJ2000  param2=e_DEJ2000  epoch=ObsDate
set cols SDSS9,objID,/.+J2000/,ObsDate,flags,/(e_)?[ugriz]mag/,cl,spCl,subClass,zsp
prefix sdss9
save test/sdss9.vot votable
#### Load table 2 ####
get VizieRLoader tabname=II/246/out mode=cone center="12.256460 +20.878920" radius=10.2arcmin allcolumns
where Qflg.matches("[ABC][ABCDU]{2}")
set pos ra=RAJ2000 dec=DEJ2000
set epoch year=jd2epoch(JD)
set poserr type=ELLIPSE param1=errMaj param2=errMin param3=errPA epoch=jd2epoch(JD)
set cols 2MASS,RAJ2000,DEJ2000,JD,/err((Maj)|(Min)|(PA))/,/(e_)?[JHK]mag/,/.flg/,Jmag-Hmag as J-H,Hmag-Kmag as H-K,Jmag-Kmag as J-K
addmeta J-H datatype=double precision=2 width=5 unit=mag ucd="phot.color" desc="Color Jmag-Kmag"
addmeta H-K datatype=double precision=2 width=5 unit=mag ucd="phot.color" desc="Color Kmag-Hmag"
addmeta J-K datatype=double precision=2 width=5 unit=mag ucd="phot.color" desc="Color Jmag-Hmag"
prefix 2mass
save test/2mass.vot votable
#### Perform xmatch of table 1 vs table 2 ####
xmatch chi2 nStep=1 nMax=2 completeness=0.997 join=full
merge pos ML
merge epoch left
save test/sdss9_vs_2mass.vot votable
#### Load table 3 ####
get VizieRLoader tabname=I/284/out mode=cone center="12.256460 +20.878920" radius=5arcmin allcolumns
```

```
where !Flags.matches(".s.")
set pos ra=RAJ2000 dec=DEJ2000
set epoch year=2000.0
set poserr type=RCD_DEC_ELLIPSE param1=sqrt(e_RAJ2000*e_RAJ2000 + 130*130)/1000.0 param2=sqrt(e_DEJ2000*e_DEJ2000 + 130*130)/1000.0
    epoch=Epoch
set pm rcd=pmRA dec=pmDE
set pmerr type=RCD_DEC_ELLIPSE param1=e_pmRA param2=e_pmDE
set cols USNO-B1.0,/(e_)?((RA)|(DE))J2000/,Epoch,/(e_)?pm((RA)|(DE))/,(int) sqrt(pmRA * pmRA + pmDE * pmDE) as tanVeloc,/.*mag/,Flags
addmeta tanVeloc datatype=int unit=mas/yr ucd=phys.veloc.transverse desc="Tangential velocity"
prefix usnob1
save test/usnob1.vot votable
#### Perform xmatch of previous result vs table 3 ####
xmatch chi2 nStep=2 nMax=2 completeness=0.997 join=inner
merge pos ML
merge epoch left
merge pmerr right
merge dist mec
save test/sdss9_vs_2mass_vs_usnob1.vot votable
```

A List of constants and functions available in *expression*

A.1 List of math constants

PI - 3.14159265358979323846

E - 2.7182818284590452354

A.2 List of math functions

A.2.1 General purpose functions

Most of them come from *java.lang.Math*, see:
http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

double toDegrees(double) - converts radians to degrees

double toRadians(double) - converts degrees to radians

double sin(double) - returns the sine of the given angle in radians

double cos(double) - returns the cosine of the given angle in radians

double tan(double) - returns the tangent of the given angle in radians

double sinh(double) - returns the hyperbolic sine of a double value

double cosh(double) - returns the hyperbolic cosine of a double value

double tanh(double) - returns the hyperbolic tangent of a double value

double asin(double) - returns the arc sine of a value; the returned angle is in the range -PI/2 through PI/2

double acos(double) - returns the arc cosine of a value; the returned angle is in the range 0.0 through PI

double atan(double) - returns the arc tangent of a value; the returned angle is in the range -PI/2 through PI/2

double atan2(double, double) - returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta)

double exp(double) - returns Euler's number e raised to the power of a double value

double expm1(double) - returns $e^x - 1$

double log(double) - returns the natural logarithm (base e) of a double value

double log10(double) - returns the base 10 logarithm of a double value

double log1p(double) - returns the natural logarithm of the sum of the argument and 1

double sqrt(double) - returns the positive square root of a double value

double cbrt(double) - returns the cube root of a double value.

double ceil(double) - returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer

double pow(double, double) - returns the value of the first argument raised to the power of the second argument

double abs(double) - returns the absolute value of a double value

double floor(double) - returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer

double round(double) - returns the closest long to the argument, with ties rounding up.

double min(double, double) - returns the smallest of two double values

double max(double, double) - returns the greatest of two double values

double hypot(double x, double) - returns $\sqrt{x^2 + y^2}$ without intermediate overflow or underflow

double random() - returns a random value uniformly ditributed $\in [0, 1[$

A.2.2 Astronomy specific functions

More specific to astronomy:

double jd2epoch(double) - converts julian days to epoch: $epoch = jd/365.25 - 4713$

double havDist(double ra1, double dec1, double ra2, double dec2) - returns the Haversine distance; input parameters in radians, result in radians

double deg2rad(double) - converts degrees to radians

double arcmin2rad(double) - converts arc minutes to radians

double arcsec2ra(double) - converts arc seconds to radians

double mas2rad(double) - converts milli arc seconds to radians

double hms2deg(String) - converts from sexagesimal hours to decimal degrees

double dms2deg(String) - converts from sexagesimal degrees to decimal degrees

A.3 List of string functions

All the function available in Java, see:
http://docs.oracle.com/javase/7/docs/api/java/lang/String.html