Package: snapshot (via r-universe)

June 2, 2026

Type Package

Title Gadget N-body cosmological simulation code snapshot format 1 and 2 I/O utilities

Version 2.0.1

Date 2020-10-05

Author Aaron Robotham & Federico Stasyszyn

Maintainer Aaron Robotham <aaron.robotham@uwa.edu.au>

Description Functions for reading and writing Gadget N-body format 1 and 2 snapshots. The Gadget code is popular in astronomy for running N-body / hydrodynamical cosmological and merger simulations. To find out more about Gadget see the main distribution page at www.mpa-garching.mpg.de/gadget/. Format 1 specific functions end with a ``.1" suffix and format 2 specific functions end with a ``.2" suffix. Generic functions have neither suffix.

License GPL-3

Depends R (>= 2.13)

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

Date/Publication 2020-10-05 10:18:18 UTC

RemoteUrl https://github.com/asgr/snapshot

RemoteRef HEAD

RemoteSha 80082c98709892d8eecf9d1f64b70136290832e5

Contents

`snapshot-package` *Gadget N-body cosmological simulation code snapshot format 1 and 2 I/O utilities*

Description

Functions for reading and writing Gadget N-body format 1 and 2 snapshots. The Gadget code is popular in astronomy for running N-body / hydrodynamical cosmological and merger simulations. To find out more about Gadget see the main distribution page at www.mpa-garching.mpg.de/gadget/. Format 1 specific functions end with a ".1" suffix and format 2 specific functions end with a ".2" suffix. Generic functions have neither suffix.

Details

Package:	snapshot
Type:	Package
Version:	2.0.1
Date:	2020-10-05
License:	GPL-3

Author(s)

Aaron Robotham & Federico Stasyszyn

Maintainer: Aaron Robotham <aaron.robotham@uwa.edu.au>

Examples

```
## Not run:
temp=snap.read.1('snapshot_XXX')
temp$part[,'x']=temp$part[,'x']+10
snap.write.1(temp$part,temp$head,'snapshot_XXX_mod')

## End(Not run)
```

snap.add.head.1 *Add header information to particle data*

Description

Function to add required header information to a Gadget format 1 particle data.frame. This has sensible defaults for a small galaxy merger style simulation

Usage

```
snap.add.head.1(part, Npart = 2, Massarr = 0, Time = 0, z = 0, FlagSfr = 0,
FlagFeedback = 0, FlagCooling = 0, BoxSize = 0, OmegaM = 0, OmegaL = 0,
h = 1, FlagAge = 0, FlagMetals = 0, NallHW = 0, flag_entr_ics = 0)
```

Arguments

part	Strictly speaking 'part' is passed through the function, but to make this a useful object 'part' should be a data.frame containing the main particle level information. Columns required are:

ID	particle ID
x	x position in units of Mpc
y	y position in units of Mpc
z	z position in units of Mpc
vx	x velocity in units of km/s
vy	y velocity in units of km/s
vz	z velocity in units of km/s
Mass	particle mass in units of Msun

Npart	The index on the Npart vector that should contain the particle number, where: gas [1] / collisionless particles [2:6]. The actual value is calculated based on the part data.frame provided with 'part', Nall is also calculated based on this number and not given as an option since the same index as Npart must be used
Massarr	The mass of the particles in the particle index provided to Npart
Time	Time of snapshot in units of km/s and kpc so 1 unit is ~10 Gyrs
z	Redshift of snapshot
FlagSfr	Star formation turned on/off
FlagFeedback	Feedback turned on/off
FlagCooling	Cooling turned on/off
BoxSize	Size of simulation box edge length in units of kpc
OmegaM	Omega matter of the simulation
OmegaL	Omega lambda of the simulation
h	Hubble constant divided by 100 used in the simulation

FlagAge	Stellar ages on/off
FlagMetals	Stellar metallacities on/off
NallHW	Tell Gadget to use large integers in the particle index provided to Npart- not usually necessary
flag_entr_ics	Entropy for gas on/off

Details

Nall is calculated based on Npart, and therfore it cannot be specified via an input argument. This increases the likelihood that a legal Gadget header will be produced.

Value

part — Strictly speaking 'part' is passed through the function, but to make this a useful object 'part' should be a data.frame containing the main particle level information. Assuming 'part' has been given a sensible input, columns provided are:

ID	particle ID
x	x position in units of Mpc
y	y position in units of Mpc
z	z position in units of Mpc
vx	x velocity in units of km/s
vy	y velocity in units of km/s
vz	z velocity in units of km/s
Mass	particle mass in units of Msun

head — A list containing various header information as list elements. These are:

Npart	Vector of length 6 containing the number of particles in this snapshot file, where: gas [1] / collisionless particles [2:6]
Massarr	Vector of length 6 containing the particle masses for the respective particle types in Npart
Time	Time of snapshot in units of km/s and kpc so 1 unit is ~10 Gyrs
z	Redshift of snapshot
FlagSfr	Star formation turned on/off
Nall	Vector of length 6 containing the number of particles in all snapshot files, where: gas [1] / collisionless particles [2:6]
FlagFeedback	Feedback turned on/off
FlagCooling	Cooling turned on/off
NumFiles	Number of files per snapshot- usually 1
BoxSize	Size of simulation box edge length in units of kpc
OmegaM	Omega matter of the simulation
OmegaL	Omega lambda of the simulation
h	Hubble constant divided by 100 used in the simulation
FlagAge	Stellar ages on/off
FlagMetals	Stellar metallacities on/off
NallHW	Tell Gadget to use large integers for the respective particle types in Npart

	- not usually necessary
flag_entr_ics	Entropy for gas on/off

Author(s)

Aaron Robotham

See Also

snap.write.1, snap.read.1, snap.write.block.2,snap.read.2, snap.gen.param

Examples

```
## Not run:
tempadd=snap.add.head.1(temp$part)

## End(Not run)
```

---

snap.clean.fields.2 *Clean out a Gadget format 2 snapshot file*

Description

Function to read format 2 blocks and write back out. It is usefull to clean the DM particles.

Usage

```
snap.clean.fields.2(file, gas)
```

Arguments

file	INPUT file name to be read
gas	OPTIONAL: write only the GAS particles.

Author(s)

Federico Stasyszyn

See Also

snap.write.block.2, snap.add.head.1, snap.write.head.2, snap.select.type.2, snap.gen.param

Examples

```
#None yet!
```

snap.gen.param *Generates a Gadget paramter file*

Description

Function to generator a legal Gadget parameter setup file. This has a sensible selection of defaults chosen for fairly small (non Cosmological) simulations.

Usage

```
snap.gen.param(ParamFile = "galaxy.param", ParamBase = "./HernTest/",
InitCondFile = "./HernStart.gdt", OutputDir = "./HernTest/", EnergyFile = "energy.txt",
InfoFile = "info.txt", TimingsFile = "timings.txt", CpuFile = "cpu.txt",
RestartFile = "restart", SnapshotFileBase = "snapshot",
OutputListFilename = "parameterfiles/output_list.txt", TimeLimitCPU = 36000,
ResubmitOn = 0, ResubmitCommand = "my-scriptfile", ICFormat = 1, SnapFormat = 1,
ComovingIntegrationOn = 0, TypeOfTimestepCriterion = 0, OutputListOn = 0,
PeriodicBoundariesOn = 0, TimeBegin = 0, TimeMax = 0.001, Omega0 = 0, OmegaLambda = 0,
OmegaBaryon = 0, HubbleParam = 1, BoxSize = 0, TimeBetSnapshot = 1e-05,
TimeOfFirstSnapshot = 0, CpuTimeBetRestartFile = 36000, TimeBetStatistics = 0.05,
NumFilesPerSnapshot = 1, NumFilesWrittenInParallel = 1, ErrTolIntAccuracy = 0.025,
CourantFac = 0.3, MaxSizeTimestep = 0.1, MinSizeTimestep = 0, ErrTolTheta = 0.5,
TypeOfOpeningCriterion = 1, ErrTolForceAcc = 0.005, TreeDomainUpdateFrequency = 0.1,
DesNumNgb = 32, MaxNumNgbDeviation = 8, ArtBulkViscConst = 1, InitGasTemp = 0,
MinGasTemp = 100, PartAllocFactor = 3.0, TreeAllocFactor = 4.8, BufferSize = 25,
UnitLength_in_cm = 3.085678e+21, UnitMass_in_g = 1.989e+43,
UnitVelocity_in_cm_per_s = 1e+05, GravityConstantInternal = 0,
MinGasHsmlFractional = 0.25, SofteningGas = 1e-04, SofteningHalo = 1e-04,
SofteningDisk = 0.4, SofteningBulge = 0.8, SofteningStars = 0, SofteningBndry = 0.1,
SofteningGasMaxPhys = 1e-04, SofteningHaloMaxPhys = 1e-04, SofteningDiskMaxPhys = 0.4,
SofteningBulgeMaxPhys = 0.8, SofteningStarsMaxPhys = 0, SofteningBndryMaxPhys = 0.1,
MaxRMSDisplacementFac = 0.2, NFWConcentration = 10, VirialMass = 200, FlatRadius = 1e-05,
DeltaVir = 200, addNFW = FALSE)
```

Arguments

ParamFile	Name for the paramter file
ParamBase	Base file path for the paramter file
InitCondFile	Full path of file containing initial conditions
OutputDir	Base directory in which to put the major Gadget outputs, including snapshots etc
EnergyFile	Name to give energy file
InfoFile	Name to give info file
TimingsFile	Name to give timings file
CpuFile	Name to give CPU file

`RestartFile` Name to give restart file

`SnapshotFileBase`
Base name for snapshots, appended by snapshot number

`OutputListFilename`
Name of file containing output times / expansion factors

`TimeLimitCPU` Max CPU time to use for Gadget run

`ResubmitOn` Flag to tell super-computer there is a resubmit file

`ResubmitCommand`
Specific to super-computer resubmit command

`ICFormat` Initial conditions format: PUT OPTIONS IN TABLE HERE

`SnapFormat` Snapshot format: PUT OPTIONS IN TABLE HERE

`ComovingIntegrationOn`
Allow for expansion of Universe

`TypeOfTimestepCriterion`
Type of particle integrator- leave at 0

`OutputListOn` Flag to tell it to use OutputListFilename as input

`PeriodicBoundariesOn`
Flag to turn on/off periodic box boundaries, only needed for large cosmological runs

`TimeBegin` Time at the beginning of simulation

`TimeMax` Max time to evolve particles to

`Omega0` Total energy density

`OmegaLambda` Cosmological constant energy density

`OmegaBaryon` Baryonic energy density

`HubbleParam` Value of H0/100 to be used

`BoxSize` Length of box edge (important for cosmological runs only)

`TimeBetSnapshot`
Time between snapshots

`TimeOfFirstSnapshot`
Time at which to output first snapshot

`CpuTimeBetRestartFile`
How often to output full restart file

`TimeBetStatistics`
Time between energy.txt updates

`NumFilesPerSnapshot`
How many files to split snapshots over

`NumFilesWrittenInParallel`
How many files to split snapshots over (probably ignore)

`ErrTolIntAccuracy`
Orbital integration accuracy

`CourantFac` Limit on time step compared to sound crossing time for hydro runs

`MaxSizeTimestep`
: Maximum time step allowed

`MinSizeTimestep`
: Minimum time step allowed

`ErrTolTheta` Controls the accurary of integration (smaller is closer to direct N-body)

`TypeOfOpeningCriterion`
: Barnes-Hut or modified opening criteria (probably ignore)

`ErrTolForceAcc` Only used for modified opening criterion (use default)

`TreeDomainUpdateFrequency`
: How often should a tree be constructed

`DesNumNgb` Number of neighbours to use for denisty estimation in SPH

`MaxNumNgbDeviation`
: How much tolerance is allowed when finding neighbours

`ArtBulkViscConst`
: Artificial viscosity term (use default)

`InitGasTemp` Initial gas temperature

`MinGasTemp` Minimum gas temperature allowed in the run

`PartAllocFactor`
: Memory buffer per particle per processor

`TreeAllocFactor`
: Memory buffer for tree calculation

`BufferSize` Total memory buffer between processors

`UnitLength_in_cm`
: Assumed IC distance units in cm (default assumes Kpc for input)

`UnitMass_in_g` Assumed mass of provided IC mass units in grams (default assumes 1e10 Msun for input)

`UnitVelocity_in_cm_per_s`
: Assumed velocity of provided units in cm/s (default assumes km/s)

`GravityConstantInternal`
: Internal units for g

`MinGasHsmlFractional`
: Minimum multiplicitive factor for smoothing length in hyrdo gas

`SofteningGas` Softening to use for gas particles

`SofteningHalo` Softening to use for halo particles

`SofteningDisk` Softening to use for disk particles

`SofteningBulge` Softening to use for bulge particles

`SofteningStars` Softening to use for star particles

`SofteningBndry` Softening to use for boundary particles

`SofteningGasMaxPhys`
: Physical softening to use for gas particles (only relevant for Cosmo run)

`SofteningHaloMaxPhys`
: Physical softening to use for halo particles (only relevant for Cosmo run)

`SofteningDiskMaxPhys`
: Physical softening to use for disk particles (only relevant for Cosmo run)

`SofteningBulgeMaxPhys`
: Physical softening to use for bulge particles (only relevant for Cosmo run)

`SofteningStarsMaxPhys`
: Physical softening to use for star particles (only relevant for Cosmo run)

`SofteningBndryMaxPhys`
: Physical softening to use for boundary particles (only relevant for Cosmo run)

`MaxRMSDisplacementFac`
: Biggest distance that a particle can move in a time step

`NFWConcentration`
: Concentration of analytic NFW profile, addNFW must be set to TRUE

`VirialMass`
: Mass within virial radius of analytic NFW profile, addNFW must be set to TRUE

`FlatRadius`
: Forces the NFW profile to be cored (not cusped), addNFW must be set to TRUE

`DeltaVir`
: Virial overdensity of NFW profile, addNFW must be set to TRUE

`addNFW`
: Logic determining whether the analyic NFW specific paramters be added to the setup file? See above

Value

No value returned, called for the side-effect of writing out a Gadget parameter setup file.

Author(s)

Aaron Robotham

See Also

`snap.write.1`, `snap.read.1`, `snap.write.head.2`, `snap.write.block.2`, `snap.read.2`

Examples

```
## Not run:
snap.gen.param('example.param','Demo/Example1/')

## End(Not run)
```

`snap.read.1` *Read in a Gadget format 1 snapshot file*

Description

This function allows the user to read in format 1 Gadget binaries. It keeps the particle information and header information in separate components of a list.

Usage

```
snap.read.1(file, thin=1, verbose=FALSE)
```

Arguments

`file`	The full path to the Gadget snapshot to be read in.
`thin`	Scalar. How much should the particle data be thinned? '`thin`'=1 means all data is read in, '`thin`'=10 means every 10th particle is read in. Larger numbers will create more sub-sampled data, and will hugely increase the read-in speed and reduce the memory required. This is useful for making images where only a small fraction of Gadget particles are required.
`verbose`	Logical. If TRUE then function will print out the current reading processes. If FALSE the read is silent.

Details

When using thinning you will generally only see a speed-up in reading times when the multiple is quite large (over 500), assuming the initial full snapshot ('`thin`'=1) can be fully read into RAM. This is because of the extra overheads involved in stop-starting the scanning and reading when thinning. If the initial snapshot cannot be read into RAM then the speed up will be much larger and witnessed with much smaller multiples (basically whatever allows the data to be comfortably read into RAM).

Value

`part` A data.frame containing the main particle level information. Columns included are:

ID	particle ID
x	x position in units of Mpc
y	y position in units of Mpc
z	z position in units of Mpc
vx	x velocity
vy	y velocity
vz	z velocity
Mass	particle mass in units of Msun

head A list containing various header information as list elements. These are:

Npart	Vector of length 6 containing the number of particles in this snapshot file, where: gas [1] / collisionless particles [2:6]
Massarr	Vector of length 6 containing the particle masses for the respective particle types in Npart
Time	Time of snapshot in units of km/s and kpc so 1 unit is ~10 Gyrs
z	Redshift of snapshot
FlagSfr	Star formation turned on/off
Nall	Vector of length 6 containing the number of particles in all snapshot files, where: gas [1] / collisionless particles [2:6]
FlagFeedback	Feedback turned on/off
FlagCooling	Cooling turned on/off
NumFiles	Number of files per snapshot- usually 1
BoxSize	Size of simulation box edge length in units of kpc
OmegaM	Omega matter of the simulation
OmegaL	Omega lambda of the simulation
h	Hubble constant divided by 100 used in the simulation
FlagAge	Stellar ages on/off
FlagMetals	Stellar metallacities on/off
NallHW	Tell Gadget to use large integers for the respective particle types in Npart - not usually necessary
flag_entr_ics	Entropy for gas on/off

Author(s)

Aaron Robotham

See Also

snap.write.1,snap.add.head.1,snap.gen.param

Examples

```
## Not run:
temp=snapread('somepath/snapshot_XXX')

## End(Not run)
```

---

snap.read.2 *Read in a Gadget format 2 snapshot file*

Description

This function opens the file and reads the header (also checks that the blocks are well written). Then goes trought the full file, checking the blocks. If the block label matches it returns that block.

Usage

```
snap.read.2(file, what, ndim, type, debug, gas, thin=1)
```

Arguments

file	File name of the snapshot file to be read.
what	Name of the block to be read.
ndim	OPTIONAL: Dimensions of the block.
type	OPTIONAL: Type of data.
debug	OPTIONAL: add extra information. If debug = 1, then it shows all the blocks present in the file.
gas	OPTIONAL: read only gas files.
thin	How much should the particle data be thinned? 'thin'=1 means all data is read in, 'thin'=10 means every 10th particle is read in. Larger numbers will create more sub-sampled data, and will hugely increase the read-in speed and reduce the memory required. This is useful for making images where only a small fraction of Gadget particles are required.

Details

Opens the file, and read header (also checks that the blocks are well written). Then goes through the full file, checking the blocks. If the block label matches 'what' then it returns that block DATA.

Value

Returns the DATA block from the file.

Author(s)

Federico Stasyszyn & Aaron Robotham

See Also

snap.write.block.2, snap.add.head.1, snap.write.head.2, snap.select.type.2, snap.gen.param

Examples

```
#None yet!
```

snap.select.type.2 *Returns information from the common Gadget format 2 snapshot LABELS*

Description

A convenience function. Given some commonly used Gadget format 2 snashot labels, it return the data type and dimensions.

Usage

```
snap.select.type.2(what)
```

Arguments

what	What is the name of label that we are looking for.

Value

If the label is found the dimentions and the type of data. If not found Type=FALSE and Ndim=0. The same is returned if HEADER.

Author(s)

Federico Stasyszyn & Aaron Robotham

See Also

snap.write.block.2, snap.add.head.1, snap.write.head.2, snap.read.2, snap.gen.param

Examples

```
#None yet!
```

snap.strip.2 *Strip out particles from a Gadget format 2 snapshot.*

Description

A convenience function to strip out different particle types and arrange as separate list elements.

Usage

```
snap.strip.2(snap, type=6)
```

Arguments

snap	The Gadget format-2 snapshot data.
type	Vector; the particle types to strip out into separate lists.

Value

A list of particles data.frames as requested, named paste("T",type,sep="").

Author(s)

Federico Stasyszyn

See Also

snap.write.block.2, snap.write.head.2, snap.read.2, snap.add.head.1, snap.gen.param

Examples

```
#None yet!
```

---

snap.write.1 *Write out a Gadget format 1 snapshot file*

Description

This function allows the user to write format 1 Gadget snapshots. It can write the particle information and header information, which are provided as separate R objects.

Usage

```
snap.write.1(part, head, file)
```

Arguments

part	A data.frame containing the main particle level information. Columns required are:

ID	particle ID
x	x position in units of Mpc
y	y position in units of Mpc
z	z position in units of Mpc
vx	x velocity in units of km/s
vy	y velocity in units of km/s
vz	z velocity in units of km/s
Mass	particle mass in units of Msun

`head` A list containing various header information as list elements. These are:

Npart	Vector of length 6 containing the number of particles in this snapshot file, where: gas [1] / collisionless particles [2:6]
Massarr	Vector of length 6 containing the particle masses for the respective particle types in Npart
Time	Time of snapshot in units of km/s and kpc so 1 unit is ~10 Gyrs
z	Redshift of snapshot
FlagSfr	Star formation turned on/off
Nall	Vector of length 6 containing the number of particles in all snapshot files, where: gas [1] / collisionless particles [2:6]
FlagFeedback	Feedback turned on/off
FlagCooling	Cooling turned on/off
NumFiles	Number of files per snapshot- usually 1
BoxSize	Size of simulation box edge length in units of kpc
OmegaM	Omega matter of the simulation
OmegaL	Omega lambda of the simulation
h	Hubble constant divided by 100 used in the simulation
FlagAge	Stellar ages on/off
FlagMetals	Stellar metallacities on/off
NallHW	Tell Gadget to use large integers for the respective particle types in Npart - not usually necessary
flag_entr_ics	Entropy for gas on/off

`file` The full path to the Gadget snapshot to be created.

Value

No value returned, called for the side-effect of writing out a binary Gadget file.

Author(s)

Aaron Robotham

See Also

`snap.read.1`, `snap.add.head.1`, `snap.gen.param`

Examples

```
## Not run:
temp=snap.write.1(snap$part,snap$head,'somepath/snapshot_XXX')

## End(Not run)
```

`snap.write.block.2` *Function to add blocks to an existing Gadget snapshot format 2 file.*

Description

Function to add blocks to an existing format 2 snapshot file. First it checks the data input and uses `snap.select.type.2` to define block properties. Then it writes the LABEL block and the DATA block.

Usage

```
snap.write.block.2(file, label, inp, ndim, type)
```

Arguments

`file`	Name of the file to be opened to append the blocks.
`label`	LABEL of the block to be added.
`inp`	Input array of data. In general a data frame if dim of 3, then x,y,z are expected in the data frame.
`ndim`	OPTIONAL: dimensions of the input array usually 3 or 1 dimentions.
`type`	OPTIONAL/DEPRECATED.

Value

No value returned. Called for the side-effect of writing out a binary Gadget format-2 snapshot file.

Note

Note that you need already a file with a header. You can generate it with `snap.write.head.2`.

Author(s)

Federico Stasyszyn

See Also

`snap.read.2`, `snap.add.head.1`, `snap.write.head.2`, `snap.gen.param`

Examples

```
#None yet!
```

`snap.write.head.2` *Function to start creating a gadget format 2 snapshot file with a header*

Description

Function to start creating a Gadget format 2 snapshot file from a given header. Writes the label block and the header.

Usage

```
snap.write.head.2(head, file)
```

Arguments

`head`	A list containing various header information as list elements. These are:
Npart	Vector of length 6 containing the number of particles in this snapshot file, where: gas [1] / collisionless particles [2:6]
Massarr	Vector of length 6 containing the particle masses for the respective particle types in Npart
Time	Time of snapshot in units of km/s and kpc so 1 unit is ~10 Gyrs
z	Redshift of snapshot
FlagSfr	Star formation turned on/off
Nall	Vector of length 6 containing the number of particles in all snapshot files, where: gas [1] / collisionless particles [2:6]
FlagFeedback	Feedback turned on/off
FlagCooling	Cooling turned on/off
NumFiles	Number of files per snapshot- usually 1
BoxSize	Size of simulation box edge length in units of kpc
OmegaM	Omega matter of the simulation
OmegaL	Omega lambda of the simulation
h	Hubble constant divided by 100 used in the simulation
FlagAge	Stellar ages on/off
FlagMetals	Stellar metallacities on/off
NallHW	Tell Gadget to use large integers for the respective particle types in Npart - not usually necessary
flag_entr_ics	Entropy for gas on/off
`file`	The full path to the Gadget snapshot to be created.

Details

Basically, it opens the file to be written. Writes the label blocks and then the header itself.

Note

With this function one starts the process of writing a snapshot in Gadget format-2.

Author(s)

Federico Stasyszyn

See Also

snap.write.block.2, snap.add.head.1, snap.select.type.2, snap.read.2, snap.gen.param

Examples

```
#None yet!
```

Index