Package ‘ctsky’

November 9, 2015

Type Package

Title Compressing Radio Astronomy Data With Domain Inspired Statistical Models

Version 0.1

Date 2015-11-09

Author Vincent T van Hees [aut, cre],
H. Muelheisen [ctb]

Maintainer Vincent T van Hees <`v.vanhees@esciencecenter.nl`>

Description Fits a linear regression model in log space and uses this to aid compression.

Suggests nortest, MCMCpack

License GPL (>=2)

LazyData TRUE

R topics documented:

`ctsky-package` *a package to model radio astronomy data*

Description

To work with this code you need to have MonetDB with R integration. For up to date information on how to install this see online MonetDB documentationinstalled. The commands I used:

```
hg clone http://dev.monetdb.org/hg/MonetDB/ MonetDB cd MonetDB hg update Jul2015
./bootstrap ./configure prefix=/some/install/dir enable-rintegration=yes enable-optimize
make -j clean install into ~/.bashrc: export PATH=$PATH:/some/install/dir/bin
```

Next step is to either work in R with the code directly or to call the R functions from within MonetDB. Please see the bash scripts I developed as an example. A more generic introduction on using R from within MonetDB can be found on the MonetDB website

Details

Package:	ctsky
Type:	Package
Version:	0.1
Date:	2015-11-09
License:	GPL (>=2)

Author(s)

- Vincent T van Hees <v.vanhees@esciencecenter.nl>
- Hannes Muelheisen
- Bart Scheers

approxdata — *approximate data based on the models as derived with function getmodel*

Description

This function takes the models dataframe as produced by getmodel and uses it to replicate the original data based on the model, which will be an approximation.

Usage

```
approxdata(models=c())
```

Arguments

`models`	dataframe of models as produced by function getmodel

Value

A dataframe with the same size as the dataframe from which models was derived

Examples

```
## Not run:
rep.data = approxdata(models=models)

## End(Not run)
```

getmodel *derive models from data*

Description

This function takes the data and derives one models per source (the subset of data identified by id). Here, the function converts the data to log space and then fits a linear model. This is what is assumed to work in radio astronomogy. Additionally the function tests whether the dependent values are normally distributed per frequency band.

Usage

```
getmodel(data=c(),dependent=c(),independent=c(),group=c(),id=c(),
                   mcmc=FALSE,include_ext1=FALSE,minN=30,do.res=FALSE,CX2=0.95)
```

Arguments

data	dataframe
dependent	name of dependent variable as stored in data (source intensity)
independent	name of independent variable as stored in data (frequency)
group	name of grouping variable, band (frequency band) in the case radio astronomy
id	name of variable as stored in data for identifying the subsets of data for which a model needs to be developed. In radio astronomy this is the source identifier named runcat
mcmc	If TRUE then use Markov Chain Monte Carlo regression instead of ordinary least squares
include_ext1	If true then keep the datapoints in radio astronomy for which the extract type equals 1. Extract type equals 1 data points are interpolations and not real data. A model is likely to be more accurate if these datapoints are ommited
minN	minimum number of measurements required per group (frequency band)
do.res	if true then also output the function residuals as a seperate dataframe
CX2	Conficende interval for for Chi-square test, default = 0.95

Value

Dataframe with models and if do.res is set to TRUE it also includes the model residuals

Examples

```
## Not run:
V = getmodel(data=D,dependent="f_int",independent="freq_eff",group="band",id="runcat",
                 mcmc=FALSE,include_ext1=FALSE,minN=minN,doinR=doin.R,CX2=CX2)

## End(Not run)
```

getvis *generate visualisation from the data as store these in a pdf*

Description

This function takes a merge between the original data and the models to create a visualation of both models and data

Usage

```
getvis(data=c(),resultpath  = c())
```

Arguments

data	dataframe based on a merge between original data and the models
resultpath	directory where where the pdf with visualisations will be stored

Value

No output is stored

Examples

```
## Not run:
getvis(data=mydata2)

## End(Not run)
```

studyshell *shell function for studying the data*

Description

Shell function for investigating the impact of model representations on compression, reproduction of residuals, approximation of original data. This function is mainly as a sanity check that the procedure is doing what it is supposed to do

Usage

```
studyshell(data=D, resultpath= "~/CTS/results/",CX2=0.95,minN = 30,
                    dependent="f_int",independent="freq_eff",group="band",id="runcat",
                         mcmc=FALSE,include_ext1=FALSE,do.res=FALSE)
```

Arguments

`data` dataframe including the variables (columns): source identifiers (runcat, xtrsrc), independent variable (freq_eff), timestamps (taustart_ts), data type (extract_type), independent variables (f_int), 1-sigma error in the independent variable (f_int_err), grouping variable (band), image id (id)

`resultpath` directory where code stores the results

`CX2` Confidence interval for the chi-square test

`minN` Minimum number of measurements per frequency band

`dependent` Name of the dependent variable in the model (f_int in our case)

`independent` Name of the independent variable in the model (freq_eff in our case)

`group` Grouping variable (band in our case)

`id` Identifier of the image, this is mainly used as a key to aid tracing back the origin of the data

`mcmc` If TRUE then use Markov Chain Monte Carlo regression instead of ordinary least squares

`include_ext1` If true then keep the datapoints in radio astronomy for which the extract type equals 1. Extract type equals 1 data points are interpolations and not real data. A model is likely to be more accurate if these datapoints are ommited

`do.res` if true then also output the function residuals as a seperate dataframe

Examples

```
## Not run:
studyshell(data=D,resultpath= "~/CTS/results/",CX2=0.95)

## End(Not run)
```

---

testcompr *test data compression*

Description

This function takes a vector and then tests the compression of the vector using nine different compression techniques: gzip low (level 1), high (level 9) or default (level NA) bzip2 low (level 1), high (level 9) or default (level NA) xz low (level 1), high (level 9) or default (level NA)

Usage

```
testcompr(data=c(),path=c())
```

Arguments

data	vector of data
path	directory that is used for temporarily creating files

Examples

```
## Not run:
result = testcompr(data=D,path=filepath)

## End(Not run)
```

Index