Package 'ctsky'

December 10, 2015

Type Package

Title Compressing Radio Astronomy Data With Domain Inspired Statistical Models

Version 0.2

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, moments

License GPL (>=2)

LazyData TRUE

NeedsCompilation no

R topics documented:

ctsky-package *Compressing Radio Astronomy Data With Domain Inspired Statistical Models*

Description

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

Details

Package:	ctsky
Type:	Package
Title:	Compressing Radio Astronomy Data With Domain Inspired Statistical Models
Version:	0.2
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, moments
License:	GPL (>=2)
LazyData:	TRUE

Index of help topics:

approxdata	approximate data based on the models as derived with function getmodel
ctsky-package	Compressing Radio Astronomy Data With Domain Inspired Statistical Models
getmodel	derive models from data
getvis	generate visualisation from the data as store these in a pdf
studyshell	shell function for studying the data
testcompr	test data compression

The package holds a set of functions I developed as part of the path-finding project Compressing the sky into a large number of statistical models.

The package requires that MonetDB is installed with embedded R.

The function studyshell is mainly a sanity check for all the code as it reproduces all the performance evaluations The function getmodel is probably the most important function: It generates the models and tests for model assumptions.

Author(s)

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

Examples

```
## Not run:
library(MonetDB.R)
conn = dbConnect(MonetDB.R(),host="localhost", dbname="rsm", user="monetdb",
                            password="monetdb")
D = dbGetQuery(conn,paste("SELECT a1.runcat,a1.xtrsrc,i2.freq_eff,
                            i2.taustart_ts,x1.extract_type,x1.f_int,
                            x1.f_int_err,i2.band,i2.id
                            FROM assocxtrsource a1,
                            (SELECT t1.runcat
                            FROM
                            (SELECT a.runcat, i.band, count(*)
                            FROM assocxtrsource a, runningcatalog r,
                            extractedsource x, image i
                            WHERE a.runcat = r.id and a.xtrsrc = x.id and x.image = i.id
                            GROUP BY runcat, band having count(*) > 30) t1
                            GROUP BY t1.runcat) t2, runningcatalog r1,
                            extractedsource x1, image i2
                            WHERE
                            a1.runcat = r1.id and a1.runcat = t2.runcat and
                            a1.xtrsrc=x1.id and x1.image= i2.id
                            ORDER BY a1.runcat,a1.xtrsrc",sep=""))
dbDisconnect(conn)

D = D[order(D$runcat,D$band),]
studyshell(data=D,dependent="f_int",independent="freq_eff",group="band",id="runcat",
            resultpath= "~/CTS/results/",do.res=TRUE,CX2=0.95,minN = 30)


## End(Not run)
```

`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(),do.log=TRUE, dependent="f_int",independent="freq_eff",
group="band",id="runcat",modeltime=TRUE)
```

Arguments

`models`	dataframe of models as produced by function getmodel
`do.log`	Turn independent and devependent variables into log

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
modeltime	Whether to use constant of time series as a model (TRUE) or to use one model per source based on the assuming of a exponential relationship between source intensity and frequency (FALSE)

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,
                   do.log=TRUE,imidcol="imageid",xtrsrc="xtrsrc",modeltime=TRUE)
```

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
do.log	whether to take the log from the indpenedent and dependent variables
imidcol	Name of column in which image id is stored
xtrsrc	Name of column in which xtrsrc is stored
modeltime	Whether to use constant of time series as a model (TRUE) or to use one model per source based on the assuming of a exponential relationship between source intensity and frequency (FALSE)

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(),dependent="f_int",independent="freq_eff",
group="band",id="runcat",timecol="taustart_ts",do.log=TRUE,
do.bar=TRUE,error.low="err_low",error.up="err_up",deplabel="Intensity (Jy)")
```

Arguments

data	dataframe based on a merge between original data and the models
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
resultpath	directory where where the pdf with visualisations will be stored
timecol	Name of variable to indicate time
do.log	Turn independent and devependent variables into log
do.bar	whether to plot the error bars
error.low	lower boundary of error range
error.up	upper boundary of error range
deplabel	label name for the dependent variable in the visualisations

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=c(), resultpath= c(),CX2=0.95,minN = 30,
                  dependent="f_int",independent="freq_eff",group="band",id="runcat",
                  timecol="taustart_ts",mcmc=FALSE,include_ext1=FALSE,do.res=FALSE,
                      do.log=TRUE,imidcol="imageid",xtrsrc="xtrsrc",
                      error.low=c(),error.up=c(),prec=c(),
                      modeltime=TRUE,do.bar=TRUE,
                      deplabel="Intensity (Jy)")
```

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
timecol	Name of variable to indicate time
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
do.log	whether to take the log from the indpenedent and dependent variables
imidcol	Name of column in which image id is stored
xtrsrc	Name of column in which xtrsrc is stored
prec	Number of decimal places at which the data needs to be roudned. If left empty then function will estimate this from 10 percent of standard deviation in dependent variable
error.low	lower boundary of error range or uncertainty itself if error.up is not provided (see error.up)
error.up	upper boundary of error range (if not provided then it will assume that error.low equals the uncertainty itself and calculates from this the upper and lower boundary
modeltime	Whether to use constant of time series as a model (TRUE) or to use one model per source based on the assuming of a exponential relationship between source intensity and frequency (FALSE)
do.bar	whether to plot the error bars
deplabel	label name for the dependent variable in the visualisations

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