

# reb

March 24, 2012

---

Hs.arms

*Hs.arms*

---

## Description

The data set gives the human chromosomal arms.

## Usage

```
data(Hs.arms)
```

## Format

The format is: chr [1:48] "1p" "1q" "2p" "2q" "3p" "3q" "4p" "4q" "5p" "5q" "6p" "6q" "7p" "7q" "8p" "8q" "9p" ...

## Source

International System of Human Cytogenetic Nomenclature (ISCN)

---

absMax

*Absolute Maxima*

---

## Description

Returns the absolute maxima of the input values.

## Usage

```
absMax(x)
```

## Arguments

x                    numeric argument

## Value

absMax returns the absolute maximum of all the values present in the arguments as double perserving the sign. Essentially `max(abs(x), na.rm=T)`.

**Author(s)**

Karl A. Dykema and Kyle A. Furge

**Examples**

```
absMax(c(1, 2, 3, 4))  
absMax(c(-1, -2, -3, -4))
```

---

buildChromCytoband *Construct a chromLocation object from a cytoband environment*

---

**Description**

Construct a chromLocation object from a cytoband environment. Human, Rat, and Mouse are currently possible.

**Usage**

```
buildChromCytoband(organism = "h")
```

**Arguments**

organism        character, "h" for human, "m" for mouse, and "r" for rat.

**Value**

a chromLocation object

**Author(s)**

Karl J. Dykema, <karl.dykema@vai.org> Kyle A. Furge, <kyle.furge@vai.org>

**See Also**

[buildChromLocation](#)

**Examples**

```
humanBands <- buildChromCytoband("h")  
humanBands@chromLocs[["1"]]
```

---

buildChromMap	<i>A function to generate an instantiation of a chromLocation class</i>
---------------	---

---

### Description

This function will take the name of a data package and build a chromLocation object representing regions of the genomes.

### Usage

```
buildChromMap(dataPkg, regions)
```

### Arguments

dataPkg	The name of the data package to be used (a.k.a generated by AnnBuilder or downloaded from Bioconductor web site)
regions	a character vector of genome regions to be generated

### Details

This function is related to the buildChromLocation function found in the 'annotate' library. However, this function can be used to build specialized chromLocation objects based on gene mapping information. For example, a chromLocation object can be build specifically for human chromosome 1 by supplying chromosomal band information, such as c("1p1", "1p2", "1p3", "1q1", "1q2", "1q3", "1q4"). Genes that map to these regions are isolated and a chromLocation object is returned. Note that genes are isolated by 'grep'ing genome mapping information. Therefore the number of genes that are able to placed into a defined genetic region (i.e. 1q4) is dependent on the quality of the mapping information in the annotation data source.

Unfortunately, not too many pre-built annotation packages are available for spotted arrays off the Bioconductor Metadata web set. Use AnnBuilder to make one or get one from your core.

### Value

A 'chromLocation' object representing the specified genomic regions and annotation data source

### Author(s)

Kyle A. Furge <kyle.furge@vai.org>

### See Also

[buildChromLocation](#)

### Examples

```
##
## NOTE: This requires an annotation package to work.
##       In this example it is hu6800
##
if (require(hu6800)) {

  library(Biobase)
```

```

library(annotate)

## Build a specific chrom arm

chr1q <- buildChromMap("hu6800", c("1q1", "1q2", "1q3", "1q4"))

## Build human data based on chrom arms

data(Hs.arms)
map <- buildChromMap("hu6800", Hs.arms)
}

```

---

cset2band

*cset2band*


---

## Description

This function will summarize gene expression data by cytogenetic band

## Usage

```
cset2band(exprs, genome, chr = "ALL", organism = NULL, FUN = isAbnormal, ...)
```

## Arguments

exprs	matrix of gene expression data or similar. The rownames must contain the gene identifiers
genome	an associated chromLoc annotation object
chr	a character vector specifying the chromosomes to analyze
organism	character, "h" for human, "m" for mouse, and "r" for rat.; defaults to NULL - loads from chromLocation object
FUN	function by which to aggregate/summarize each cytogenetic band
...	extra arguments passed on to the aggregate/summary function

## Details

This function loops through each band for a given organism and summarizes the data for genes that lie within each cytogenetic band based upon the input function. For example, a matrix of gene expression values could be used and the mean expression of each band be determined by passing the `mean` function. Alternative, DNA copy number gains or losses could be predicted using the `reb` function and regions of likely gain or losses be summarized by cytogenetic band using the `isAbnormal` function.

## Value

a matrix with rows representing cytogenetic bands, and columns representing individual samples.

## Author(s)

Karl Dykema

**Examples**

```

data(mcr.eset)
data(idiogramExample)

## Create a vector with the index of normal samples
norms <- grep("MNC",colnames(mcr.eset@exprs))

## Smooth the data using the default 'movbin' method,
## with the normal samples as reference and median centering
cset <- reb(mcr.eset@exprs,vai.chr,ref=norms,center=TRUE)

## Mask the result to remove noise
exprs <- cset[,-norms]
exprs[abs(exprs) < 1.96] <- NA

## Starting data
midiogram(exprs,vai.chr,method="i",col=.rwb,dlim=c(-4,4))

## Summarize each cytogenetic band
banded <- cset2band(exprs,vai.chr,FUN=mean,na.rm=TRUE)

## Create chromLocation object based on human cytobands
h.cyto <- buildChromCytoband(organism = "h")

## Plot all data using mideogram
midiogram(banded,h.cyto,method="i",col=.rwb,dlim=c(-4,4))

```

---

fromRevIsh

*Convert from revish strings to a matrix*


---

**Description**

This function will convert two lists of revish style strings to a matrix format.

**Usage**

```
fromRevIsh(enhList, dimList, chr, organism = "h")
```

**Arguments**

enhList	list of enhanced bands on each individual sample
dimList	list of diminished bands on each individual sample
chr	chromosome to examine
organism	character, "h" for human, "m" for mouse, and "r" for rat.

**Value**

A matrix is returned. The rownames of this matrix correspond to the major bands located on that chromosome, and the columns correspond to the sample names.

**Author(s)**

Karl J. Dykema, <karl.dykema@vai.org> Kyle A. Furge, <kyle.furge@vai.org>

**References**

MCR eset data was obtained with permission. See PMID: 15377468

**See Also**

[reb,revish](#)

**Examples**

```
mb.chr <- buildChromCytoband("h")

data(mcr.eset)
data(idiogramExample)
## Create a vector with the index of normal samples
norms <- grep("MNC", colnames(mcr.eset@exprs))
## Smooth the data using the default 'movbin' method, with the normal samples as reference
cset <- reb(mcr.eset@exprs, vai.chr, ref=norms, center=TRUE)
## Mask the cset to remove noise
exprs <- cset[, -norms]
exprs[abs(exprs) < 1.96] <- NA
## Extract the aberrations on the 5th chromosome
revish <- revish(exprs, vai.chr, "5")
## Convert back to matrix
reconverted <- fromRevIsh(revish[[1]], revish[[2]], "5")

layout(cbind(1, 2))
idiogram(cset[, -norms], vai.chr, "5", method="i", dlim=c(-2, 2), col=.rwb, main="chr 5 reb result")
idiogram(reconverted, mb.chr, "5", method="i", dlim=c(-1, 1), col=.rwb, main="chr 5 converted \n")
```

---

isAbnormal

*Is a band 'abnormal'?*

---

**Description**

Returns 1 or -1 indicating a chromosomal change based upon an input percentage.

**Usage**

```
isAbnormal(x, percent = 0.5)
```

**Arguments**

x	genomic data, can contain NA's
percent	numeric argument - a fraction or percentage

**Details**

This simple function is used by `cset2band`.

**Author(s)**

Karl Dykema

**See Also**

[cset2band](#)

**Examples**

```
#Not abnormal
isAbnormal(c(1, NA))
#Abnormal; +
isAbnormal(c(1, NA, 1))
#Abnormal; -
isAbnormal(c(1, NA, -1, -1, -1))
```

---

`mcr.eset`*Example exprSet and chromLocation objects*

---

**Description**

An example `exprSet` and a `chromLocation` object generated from an gene expression profiling experiment of leukemic and normal blood cells. Profiling was done on custom pin-printed cDNA arrays.

**Usage**

```
data(mcr.eset)
```

**Source**

Lindvall C, Furge K, Bjorkholm M, Guo X, Haab B, Blennow E, Nordenskjold M, Teh BT. Combined genetic and transcriptional profiling of acute myeloid leukemia with normal and complex karyotypes. *Haematologica*. 2004 Sep;89(9):1072-81 PMID: 15377468

**Examples**

```
data(mcr.eset)
```

```
str(mcr.eset)
```

---

 movbin

*movbin*


---

## Description

This function analyzes ordered data series to identify regional biases using an moving (running) approximated binomial test.

## Usage

```
movbin(v, span=NULL, summarize=mean)
```

## Arguments

<code>v</code>	data vector
<code>span</code>	numeric vector. Each element is used to define the number of points to include when the approximated binomial test is applied to <code>v</code> . While mixed for the defaults, the span can be specified as fraction of the observation or actual sizes, but <i>not</i> a mixture - defaults to: <code>seq(25,length(v)*.3,by=5)</code>
<code>summarize</code>	function that is used to summarize the results from multiple spans. if NULL, a matrix with <code>length(span)</code> rows and <code>length(v)</code> columns is returned.

## Details

`movbin` applies a moving binomial test to sequential windows of elements of `v`. Within each span a z-score from an approximated binomial is computed such that  $z = (2 * r - n) / \sqrt{n}$  where `r` is the number of positive relative gene expression values and `n` is the number of non-zero values within each window.

For convenience, this function allows for the specification of multiple window sizes using the `span` argument. The result of a `movbin` call will generate a matrix with `length(span)` rows and `length(v)` columns. Each row of the matrix represents the data generated from each span. This matrix can be returned or the matrix from can be condensed to a single vector of length `v` by applying a summary function `summarize` to the matrix columns.

## Value

Either a matrix or a vector containing the summarized z-scores from the applied binomial test.

## Author(s)

Kyle A. Furge, Ph.D., <kyle.furge@vai.org> and Karl J. Dykema, <karl.dykema@vai.org>

## Examples

```
x <- c(rnorm(50,mean=1), rnorm(50,mean=-1), rnorm(100))
layout(1:2)
plot(x,type="h",ylim=c(-5,5))

## apply the approximated binomial with a single span
mb <- movbin(x,span=25,summarize=NULL)
lines(mb[1,])
```



```
## try a few different span ranges
mb <- movbin(x, span=c(10, 25, 50), summarize=NULL)
lines(mb[1,]) ## span of 10
lines(mb[2,]) ## span of 25
lines(mb[3,]) ## span of 50

## average the results from the different spans
plot(x, type="h", ylim=c(-5, 5))

mb <- movbin(x, span=c(10, 25, 50), summarize=mean)
lines(mb, col="blue")

mb <- movbin(x, span=c(10, 25, 50), summarize=median)
lines(mb, col="red")
```

---

 movt

*movt*


---

## Description

This function analyzes ordered data series to identify regional biases using an moving (running) approximated t-test.

## Usage

```
movt(v, span=NULL, summarize=mean)
```

## Arguments

<code>v</code>	data vector
<code>span</code>	numeric vector. Each element is used to define the number of points to include when the approximated binomial test is applied to <code>v</code> . While mixed for the defaults, the span can be specified as fraction of the observation or actual sizes, but <i>not</i> a mixture - defaults to: <code>seq(25, length(v)*.3, by=5)</code>
<code>summarize</code>	function that is used to summarize the results from multiple spans. if <code>NULL</code> , a matrix with <code>length(span)</code> rows and <code>length(v)</code> columns is returned.

## Details

`movt` acts very similar to `movbin`

## Value

Either a matrix or a vector containing the summarized z-scores from the applied t-test.

## Author(s)

Kyle A. Furge, Ph.D., <kyle.furge@vai.org> and Karl J. Dykema, <karl.dykema@vai.org>

## See Also

[movbin](#)

**Examples**

```
x <- c(rnorm(50,mean=1),rnorm(50,mean=-1),rnorm(100))
layout(1:2)
plot(x,type="h",ylim=c(-5,5))

## apply the approximated binomial with a single span
mb <- movbin(x,span=25,summarize=NULL)
lines(mb[1,])

## try a few different span ranges
mb <- movt(x,span=c(10,25,50),summarize=NULL)
lines(mb[1,]) ## span of 10
lines(mb[2,]) ## span of 25
lines(mb[3,]) ## span of 50

## average the results from the different spans
plot(x,type="h",ylim=c(-5,5))

mb <- movt(x,span=c(10,25,50),summarize=mean)
lines(mb,col="blue")

mb <- movt(x,span=c(10,25,50),summarize=median)
lines(mb,col="red")

mb <- movt(x,span=c(10,25,50),summarize=max)
lines(mb,col="green")
```

---

naMean

*Wrapper function for the arithmetic mean*


---

**Description**

Simple call to mean with the `na.rm` option set to TRUE.

**Usage**

```
naMean(x)
```

**Arguments**

`x` An R object

**Value**

The arithmetic mean of the values in `x`.

**Examples**

```
mean(c(1,2,3,NA),na.rm=TRUE)
naMean(c(1,2,3,NA))
```

---

regmap	<i>image function wrapper</i>
--------	-------------------------------

---

## Description

A simple wrapper around the image function

## Usage

```
regmap(m, scale=c(-6, 6), na.color=par("bg"), ...)
```

## Arguments

<code>m</code>	a matrix
<code>scale</code>	Include a graph scale showing this range of values ‘image’ function
<code>na.color</code>	the color to draw over NA values
<code>...</code>	additional paramters to ‘image’

## Details

A small wrapper around the ‘image’ function to display genome region summary statistics. Additional parameters will be passed along to the image function.

The scale argument is a two-element vector that provides a floor and ceiling for the matrix and allows a crude scale bar to be included on the lower potion of the graph.

For other colors consider using the `geneploater` (`dChip.colors`) or `marrayPlots` (`maPalette`) library functions (i.e. `regmap(m, col=dChipColors(50))`)

## Author(s)

Kyle A. Furge

## See Also

[image, summarizeByRegion](#)

## Examples

```
m <- matrix(rnorm(6*4), ncol=6)
colnames(m) <- c(1:6)
rownames(m) <- c("1p", "1q", "2p", "2q")
regmap(m, scale=c(-1, 1))
```

---

 revish

*Creation of CGH (reverse in situ hybridization) style character strings*


---

### Description

This function returns a two lists of character strings. These two lists correspond to the enhanced and diminished chromosomal bands.

### Usage

```
revish(cset, genome, chr, organism = NULL)
```

### Arguments

cset	expression set containing cytogenetic predictions, see <a href="#">reb</a>
genome	chromLocation object containing annotation information
chr	chromosome to examine
organism	if NULL, determination of the host organism will be retrieved from the <code>organism</code> slot of the <code>chromLocation</code> object. Otherwise "h", "r", or "m" can be used to specify <b>h</b> uman, <b>r</b> at, or <b>m</b> ouse chromosome information

### Value

enh	list of enhanced bands on each individual sample
dim	list of diminished bands on each individual sample

### Author(s)

Karl J. Dykema, <karl.dykema@vai.org> Kyle A. Furge, <kyle.furge@vai.org>

### References

Furge KA, Dykema KJ, Ho C, Chen X. Comparison of array-based comparative genomic hybridization with gene expression-based regional expression biases to identify genetic abnormalities in hepatocellular carcinoma. *BMC Genomics*. 2005 May 9;6(1):67. PMID: 1588246

MCR eset data was obtained with permission. See PMID: 15377468

### See Also

[reb](#)

### Examples

```
data(idiogramExample)
ix <- abs(colo.eset) > .225
colo.eset[ix] <- NA
idiogram(colo.eset,ucsf.chr,"14",method="i",dlim=c(-1,1),col=.rwb)
revlist<- revish(colo.eset,ucsf.chr,"14")
str(revlist)
```

---

rmAmbigMappings	<i>Remove genes that map to multiple chromosomes from a chromLocation object</i>
-----------------	--

---

### Description

Due to the automated probe annotation, a subset of probes can be “confidently” mapped to multiple chromosomes on the genome.

This can cause some confusion if you are trying to perform certain types of data analysis.

This function examines a `chromLocation` object and removes probes that map to multiple chromosomes.

### Usage

```
rmAmbigMappings(cL)
```

### Arguments

`cL`                    an existing `chromLocation` object

### Value

A `chromLocation` object

### Author(s)

Kyle A. Furge

### See Also

[buildChromLocation](#)

### Examples

```
if (require(hu6800)) {  
  library(Biobase)  
  library(annotate)  
  
  ## Build a specific chrom arm  
  
  cL <- buildChromLocation("hu6800")  
  cleanCL <- rmAmbigMappings(cL)  
}
```

---

smoothByRegion      *reb*

---

### Description

This function “smooths” gene expression data to assist in the identification of regional expression biases.

### Usage

```
reb(eset, genome, chrom = "ALL", ref = NULL, center = FALSE,
    agrfun=absMax, method = c("movbin", "supsmu", "lowess", "movt"), ...)
```

### Arguments

<code>eset</code>	the expression set to analyze
<code>genome</code>	an associated <code>chromLoc</code> annotation object
<code>chrom</code>	a character vector specifying the chromosomes to analyze
<code>ref</code>	a vector containing the index of reference samples from which to make comparisons. Defaults to <code>NULL</code> (internally referenced samples)
<code>center</code>	boolean - re-center gene expression matrix columns. Helpful if <code>ref</code> is used
<code>aggrfun</code>	a function to summarize/aggregates gene expression values that map to the same locations. Defaults to the maximum absolute value <code>absMax</code> . If <code>NULL</code> , all values are included.
<code>method</code>	smoothing function to use - either "supmu", "lowess", "movbin" or "movt".
<code>...</code>	additional paramaters to pass along to the smoothing function

### Details

`reb` returns an `eset` that contains predictions of regional expression bias using data smoothing approaches. The `exprSet` is separated into subsets based on the `genome` `chromLocation` object and the gene expression data within the subsets is organized by genomic location and smoothed. In addition, the `approx` function is used to estimate data between any missing values. This was implemented so the function follows the ‘principles of least astonishment’.

Smoothing approaches are most straightforwardly applied by comparing a set of test samples to a set of control samples. For single color experiments, the control samples can be specified using the `ref` argument and the comparisons are generated internal to the `reb` function. This argument can also be used for two-color experiments provided both the test and control samples were run against a common reference.

If multiple clones map to the same genomic locus the `aggrfun` argument can be used to summarize the overlapping expression values to a single summarized value. This is can be helpful in two situations. First, the `supsum` and `lowess` smoothing functions do not allow for duplicate values. Currently, if duplicate values are found and these smoothing functions are used, the duplicate values are simply discard. Second, if 50 copies of the actin gene are present on a the array and actin changes expression under a given condition, it may appear as though a regional expression bias exists as 50 values within a region change expression. Summarizing the 50 expression values to a single value can partially correct for this effect.

The `idiogram` package can be used to plot the regional expression bias.

**Value**

An exprSet

**Author(s)**

Kyle A. Furge, <kyle.furge@vai.org> Karl J. Dykema, <karl.dykema@vai.org>

**References**

Furge KA, Dykema KJ, Ho C, Chen X. Comparison of array-based comparative genomic hybridization with gene expression-based regional expression biases to identify genetic abnormalities in hepatocellular carcinoma. *BMC Genomics*. 2005 May 9;6(1):67. PMID: 1588246

MCR eset data was obtained with permission. See PMID: 15377468

**See Also**

[movbin](#), [idiogram](#)

**Examples**

```
# The mcr.eset is a two-color gene expression exprSet
# with cytogenetically complex (MCR) and normal
# control (MNC) samples which are a pooled-cell line reference.

data("mcr.eset")
data(idiogramExample)

## Create a vector with the index of normal samples
norms <- grep("MNC", colnames(mcr.eset@exprs))

## Smooth the data using the default 'movbin' method,
## with the normal samples as reference

cset <- reb(mcr.eset@exprs, vai.chr, ref=norms, center=TRUE)

## Display the results with midiogram
midiogram(cset@exprs[, -norms], vai.chr, method="i", dlim=c(-5, 5), col=.rwb)
```

---

summarizeByRegion *Compute Summary Statistics of Genome Regions*

---

**Description**

Splits the data into subsets based on genome mapping information, computes summary statistics for each region, and returns the results in a convenient form. (cgma stands for Comparative Genomic Microarray Analysis)

This function supplies a t.test function at the empirically derived significance threshold (p.value = 0.005)

**Usage**

```
cgma(eset, genome, chrom="ALL", ref=NULL, center=TRUE, aggrfun=NULL, p.value=0.005,
```

**Arguments**

<code>eset</code>	an <code>exprSet</code> object
<code>genome</code>	an <code>chromLocation</code> object, such as on produced by <code>buildChromLocation</code> or <code>buildChromMap</code>
<code>chrom</code>	a character vector specifying the chromosomes to analyze
<code>ref</code>	a vector containing the index of reference samples from which to make comparisons. Defaults to <code>NULL</code> (internally referenced samples)
<code>center</code>	boolean - re-center gene expression matrix columns. Helpful if <code>ref</code> is used
<code>aggrfun</code>	a function to summarize/aggregates gene expression values that map to the same locations. If <code>NULL</code> , all values are included. Also see <code>absMax</code>
<code>p.value</code>	<code>p.value</code> cutoff, <code>NA</code> for all results, or <code>TRUE</code> for all t.stats and <code>p.values</code>
<code>FUN</code>	function by which to summarize the data
<code>verbose</code>	boolean - print verbose output during execution?
<code>explode</code>	boolean - explode summary matrix into a full expression set?
<code>...</code>	further arguments pass to or used by the function

**Details**

Gene expression values are separated into subsets that based on the 'chromLocation' object argument. For example, `buildChromMap` can be used to produce a 'chromLocation' object composed of the genes that populate human chromosome 1p and chromosome 1q. The gene expression values from each of these regions are extracted from the 'exprSet' and a summary statistic is computed for each region.

`cgma` is most straightforwardly used to identify regional gene expression biases when comparing a test sample to a reference sample. For example, a number of simple tests can be used to determine if a genomic region contains a disproportionate number of positive or negative log transformed gene expression ratios. The presence of such a regional expression bias can indicates an underlying genomic abnormality.

If multiple clones map to the same genomic locus the `aggregate.by.loc` argument can be used to include a summary value for the overlapping expression values rather than include all of the individual gene expression values. For example, if 50 copies of the actin gene are on a particular array and actin changes expression under a given condition, it may appear as though a regional expression bias exists as 50 values in a small region change expression.

`regmap` is usually the best way to plot results of this function. `idiogram` can also be used if you set the "explode" argument to `TRUE`.

`buildChromLocation.2` can be used to create a `chromLocation` object in which the genes can be divided a number of different ways. Separating the data by chromosome arm was the original intent. If you use `buildChromLocation.2` with the "arms" argument to build your `chromLocation` object, set the "chrom" argument to "arms" in this function.

**Value**

`m` A matrix of summary statistics



**Author(s)**

Kyle A. Furge

**References**

Crawley and Furge, Genome Biol. 2002;3(12):RESEARCH0075. Epub 2002 Nov 25.

**See Also**

[buildChromMap,tBinomTest,regmap,buildChromLocation.2](#)

**Examples**

```
##
## NOTE: This requires an annotation package to work.
##       In this example packages "hu6800" and "golubEsets" are used.
##       They can be downloaded from http://www.bioconductor.org
##       "hu6800" is under MetaData, "golubEsets" is under Experimental Data.

if(require(hu6800) && require(golubEsets)) {
  data(Golub_Train)
  cloc <- buildChromMap("hu6800",c("1p","1q","2p","2q","3p","3q"))

  ## For one-color expression data
  ## compare the ALL samples to the AML samples
  ## not particularly informative in this example

  aml.ix <- which(Golub_Train$"ALL.AML" == "AML")
  bias <- cgm(aeset=Golub_Train,ref=aml.ix,genome=cloc)
  regmap(bias,col=.rwb)
} else print("This example requires the hu6800 and golubEsets data
packages.")

## A more interesting example

## The mcr.eset is a two-color gene expression exprSet
## where cytogenetically complex (MCR),
## cytogenetically simple (CN) leukemia samples
## and normal control (MNC) samples were profiled against
## a pooled-cell line reference
## The MCR eset data was obtained with permission. See PMID: 15377468

## Notice the diminished expression on chromosome 5 in the MCR samples
## and the enhanced expression on chromosome 11
## This reflects chromosome gains and losses as validated by CGH

data("mcr.eset")
data(idiogramExample)
norms <- grep("MNC",colnames(mcr.eset@exprs))
bias <- cgm(mcr.eset@exprs,vai.chr,ref=norms)
regmap(bias,col=topo.colors(50))
```

---

tBinomTest	<i>binomial t-test</i>
------------	------------------------

---

**Description**

Binomial t-test

**Usage**

```
tBinomTest(x, trim=.1)
```

**Arguments**

x	numeric argument
trim	trim at?

**Value**

bla bla bla

**Author(s)**

Karl A. Dykema and Kyle A. Furge

**Examples**

```
cat("this is an example")
```

---

writeGFF3	<i>Output of a GFF compliant table describing the enhanced and diminished chromosomal bands.</i>
-----------	--

---

**Description**

This function writes out a GFF compliant tab delimited file for intergration with genome browsers.

**Usage**

```
writeGFF3(cset, genome, chr, file.prefix = "temp.gff", organism = NULL)
```

**Arguments**

cset	expression set containing cytogenetic predictions, see <a href="#">reb</a>
genome	chromLocation object containing annotation information
chr	chromosome to examine
file.prefix	character string - name of the output file, defaults to "temp.gff"
organism	if NULL, determination of the host organism will be retrieved from the organism slot of the chromLocation object. Otherwise "h", "r", or "m" can be used to specify <b>h</b> uman, <b>r</b> at, or <b>m</b> ouse chromosome information

**Value**

writeGFF3 returns an invisible list of character vectors.

**Author(s)**

Karl J. Dykema, <karl.dykema@vai.org> Kyle A. Furge, <kyle.furge@vai.org>

**References**

Furge KA, Dykema KJ, Ho C, Chen X. Comparison of array-based comparative genomic hybridization with gene expression-based regional expression biases to identify genetic abnormalities in hepatocellular carcinoma. *BMC Genomics*. 2005 May 9;6(1):67. PMID: 1588246

MCR eset data was obtained with permission. See PMID: 15377468

**See Also**

[reb](#)

**Examples**

```
data(idiogramExample)
ix <- abs(coło.eset) > .225
coło.eset[ix] <- NA
idiogram(coło.eset,ucsf.chr,"14",method="i",dlim=c(-1,1),col=.rwb)
gffmat <- writeGFF3(coło.eset,ucsf.chr,"14",NULL)
gffmat[1:4,]
```

# Index

## \*Topic **arith**

isAbnormal, 6

## \*Topic **datasets**

Hs.arms, 1

mcr.eset, 7

## \*Topic **manip**

absMax, 1

buildChromCytoband, 2

buildChromMap, 3

cset2band, 4

fromRevIsh, 5

movbin, 8

movt, 9

naMean, 10

regmap, 11

revish, 12

rmAmbigMappings, 13

smoothByRegion, 14

summarizeByRegion, 15

tBinomTest, 18

writeGFF3, 18

reb (*smoothByRegion*), 14

regmap, 11, 17

revish, 6, 12

rmAmbigMappings, 13

smoothByRegion, 14

summarizeByRegion, 11, 15

tBinomTest, 17, 18

writeGFF3, 18

absMax, 1

buildChromCytoband, 2

buildChromLocation, 2, 3, 13

buildChromLocation.2, 16, 17

buildChromMap, 3, 17

cgma (*summarizeByRegion*), 15

cset2band, 4, 7

fromRevIsh, 5

Hs.arms, 1

image, 11

isAbnormal, 6

mcr.eset, 7

movbin, 8, 9, 15

movt, 9

naMean, 10

reb, 6, 12, 18, 19