The hydrosanity Package

September 27, 2007

Type  Package

Title  Graphical user interface for exploring hydrological time series

Version  0.8.71

Date  2007-09-27

Author  Felix Andrews <felix@nfrac.org>

Maintainer  Felix Andrews <felix@nfrac.org>

Depends  R (>= 2.5.0), RGtk2, plotAndPlayGTK (>= 0.8.63), lattice (>= 0.16)

Imports  reshape, grid, grDevices, stats, utils

Suggests  sp, rgdal, tripack, gpclib, akima, oz, maps, mapdata, mapproj

Description  Hydrosanity provides a graphical user interface for exploring hydrological time series. It is designed to work with catchment surface hydrology data (rainfall and streamflow); but it could also be used with other time series data. Hydrological time series typically have many missing values, and varying accuracy of measurement (indicated by data quality codes). Furthermore, the spatial coverage of data varies over time. Much of the functionality of this package attempts to understand these complex data-sets, detect errors and characterise sources of uncertainty. The emphasis is on graphical methods. Hydrosanity’s Graphical User Interface was based on Rattle by Graham Williams.

License  GPL version 2 or newer

URL  http://hydrosanity.googlecode.com/

R topics documented:

aggregate.timeblob ..................................................... 2
arealSubPolygons .................................................. 3
gaps ................................................................. 4
grid.timeline.plot .................................................. 5
grid.timeseries.plot .................................................. 6
grid.xaxis.POSIXt ................................................... 8
aggregate.timeblob

**Aggregate to Longer Timesteps**

**Description**

A concise (1-5 lines) description of what the function does.

**Usage**

```r
## S3 method for class 'timeblob':
aggregate(x, by = "1 year", FUN = mean, fun.qual = c("worst", "median", "mode", "omit"), start.month = 1, ...)
```

**Arguments**

- `x`: Describe blob here
- `by`: Describe by here
- `FUN`: Describe FUN here
- `fun.qual`: Describe fun.qual here
- `start.month`: Describe start.month here
- `...`: Describe ... here

**Details**

If necessary, more details than the description above

**Value**

Describe the value returned If it is a LIST, use

- `comp1`: Description of 'comp1'
- `comp2`: Description of 'comp2'
- ...

---

Index

aggregate.timeblob 24
arealSubPolygons

Author(s)

who you are

References

put references to the literature/web site here

See Also

objects to See Also as help.

Examples

arealSubPolygons  Compute sub-regions closest to each point

Description

Compute the Voronoi mosaic of a set of points, and intersect it with a given polygon. This gives the sub-regions of the polygon for which each point is the closest.

Usage

arealSubPolygons(x, y = NULL, IDs = row.names(x), boundary, min.area.pct = 0.5)

Arguments

  x          x coordinate of points, or a list containing x and y.
  y          y coordinate of points.
  IDs        identification strings for each point.
  boundary   polygon coordinates (coerced to "gpc.poly" class).
  min.area.pct minimum percentage area of the polygon to be allocated to any one site. Sites which would have less area than this will be excluded.

Details

Voronoi mosaic, AKA Thiessen polygons, AKA Dirichlet tessellation.

Value

An object of class SpatialPolygons with ID slots from the original IDs argument.

Author(s)

Felix Andrews (felix@nfrac.org)
See Also

voronoi.mosaic, gpc.poly-class, SpatialPolygons

Examples

```r
## Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,  
##-- or do help(data=index) for the standard data sets.
```

---

gaps

Find Runs of Consecutive Missing Values, etc

Description

A concise (1-5 lines) description of what the function does.

Usage

```r
gaps(x, max.length = Inf, internal.only = T)
expand.indices(info)
ris(x)
lastTime(x)
```

Arguments

- `x`
  Describe `x` here
- `internal.only`
  Describe `internal.only` here
- `max.length`
  Describe `max.length` here
- `info`
  Describe `info` here

Details

If necessary, more details than the description above

Value

Describe the value returned If it is a LIST, use

```r
comp1 Description of 'comp1'
comp2 Description of 'comp2'
...```
**grid.timeline.plot**

**Author(s)**

who you are

**References**

put references to the literature/web site here

**See Also**

objects to See Also as help,

**Examples**

```r
grid.timeline.plot  # Plot Timeline of Data and its Quality
```

**Description**

A concise (1-5 lines) description of what the function does.

**Usage**

```r
grid.timeline.plot(blob.list, xscale = NULL, colMap = NULL, barThickness = unit(1.2, "lines"), auto.key = TRUE, maxLabelChars = 20, pad = unit(1,"lines"), grill = TRUE, main = NULL, sub = TRUE, newpage = TRUE)
grid.timeline.bar <- function(blob, colMap = NULL, name = "timeline.bar", vp = NULL)
timelineColMapDefault(colMap = list(good="black", suspect=trellis.par.get("superpose.polygon")$col[1], disaccumulated=trellis.par.get("superpose.polygon")$col[3], imputed=trellis.par.get("superpose.polygon")$col[4]))
```

**Arguments**

- **blob.list**
  - Describe blob.list here
- **xscale**
  - Describe xscale here
- **colMap**
  - Describe colMap here
- **barThickness**
  - Describe thickness here
- **auto.key**
  - Describe auto.key here
- **maxLabelChars**
  - Describe maxLabelChars here
- **pad**
  - Describe pad here
- **grill**
  - Describe grill here
- **main**
  - Describe main here
- **sub**
  - Describe sub here
- **newpage**
  - Describe newpage here
- **blob**
  - Describe blob here
- **name**
  - Describe name here
- **vp**
  - Describe vp here
Details

If necessary, more details than the description above

Value

Describe the value returned If it is a LIST, use

comp1 Description of 'comp1'
comp2 Description of 'comp2'
...

Author(s)

who you are

References

put references to the literature/web site here

See Also

objects to See Also as help.

Examples

grid.timeseries.plot(blob.list, xscale = NULL, yscale = NULL, sameScales = T, logScale = F, qualTimeline = F, colMap = NULL, barThickness = unit(0.5,"lines"), auto.key = T, maxLabelChars = 20, pad = unit(1,"lines"), between = unit(0,"lines"), superPos = 1, newScale = T, main = NULL, sub = T, newpage = (superPos==1), nSuperpose = 1, gp=gpar(col=rep(trellis.par.get("superpose.line")$col, len=superPos)[superPos], lty=rep(trellis.par.get("superpose.line")$lty, len=superPos))

grid.timeseries.plot.superpose(superpose.blob.list, allSameScales = F, xscale = NULL, yscale = NULL, sameScales = T, logScale = F, qualTimeline = F, colMap = NULL, barThickness = unit(0.5,"lines"), auto.key = T, maxLabelChars = 20, pad = unit(1,"lines"), between = unit(0,"lines"), superPos = 1, newScale = T, main = NULL, sub = T, newpage = (superPos==1), nSuperpose = 1, gp=gpar(col=rep(trellis.par.get("superpose.line")$col, len=superPos)[superPos], lty=rep(trellis.par.get("superpose.line")$lty, len=superPos))

grid.timeseries.steps(blob, logScale = F, name = "timeseries", gp = NULL, vp = NULL)

Description

A concise (1-5 lines) description of what the function does.

Usage

grid.timeseries.plot(blob.list, xscale = NULL, yscale = NULL, sameScales = T, logScale = F, qualTimeline = F, colMap = NULL, barThickness = unit(0.5,"lines"), auto.key = T, maxLabelChars = 20, pad = unit(1,"lines"), between = unit(0,"lines"), superPos = 1, newScale = T, main = NULL, sub = T, newpage = (superPos==1), nSuperpose = 1, gp=gpar(col=rep(trellis.par.get("superpose.line")$col, len=superPos)[superPos], lty=rep(trellis.par.get("superpose.line")$lty, len=superPos))

grid.timeseries.plot.superpose(superpose.blob.list, allSameScales = F, xscale = NULL, yscale = NULL, sameScales = T, logScale = F, qualTimeline = F, colMap = NULL, barThickness = unit(0.5,"lines"), auto.key = T, maxLabelChars = 20, pad = unit(1,"lines"), between = unit(0,"lines"), superPos = 1, newScale = T, main = NULL, sub = T, newpage = (superPos==1), nSuperpose = 1, gp=gpar(col=rep(trellis.par.get("superpose.line")$col, len=superPos)[superPos], lty=rep(trellis.par.get("superpose.line")$lty, len=superPos))

grid.timeseries.steps(blob, logScale = F, name = "timeseries", gp = NULL, vp = NULL)
Arguments

- blob.list: Describe blob.list here
- xscale: Describe xscale here
-yscale: Describe yscale here
- sameScales: Describe sameScales here
- logScale: Describe logScale here
- qualTimeline: Describe qualTimeline here
- colMap: Describe colMap here
- barThickness: Describe thickness here
- auto.key: Describe auto.key here
- maxLabelChars: Describe maxLabelChars here
- pad: Describe pad here
- between: Describe between here
- superPos: Describe superPos here
- newScale: Describe newScale here
- main: Describe main here
- sub: Describe sub here
- newpage: Describe newpage here
- nSuperpose: Describe nSuperpose here
- gp: Describe gp here
- superpose.blob.list: Describe superpose.blob.list here
- allSameScales: Describe allSameScales here
- ...: Describe ... here
- blob: Describe blob here
- name: Describe blob here
- vp: Describe blob here

Details

If necessary, more details than the description above

Value

Describe the value returned. If it is a LIST, use

- comp1: Description of `comp1`
- comp2: Description of `comp2`
- ...

grid.xaxis.POSIXt

Author(s)
who you are

References
put references to the literature/web site here

See Also
objects to See Also as help,

Examples

---

grid.xaxis.POSIXt  Time Axis for Grid

---

Description
A concise (1-5 lines) description of what the function does.

Usage
grid.xaxis.POSIXt(lim = as.numeric(convertX(unit(c(0,1), "npc"), "native")), label

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lim</td>
<td>Describe lim here</td>
</tr>
<tr>
<td>label</td>
<td>Describe label here</td>
</tr>
<tr>
<td>draw</td>
<td>Describe draw here</td>
</tr>
<tr>
<td>name</td>
<td>Describe name here</td>
</tr>
<tr>
<td>...</td>
<td>Describe ... here</td>
</tr>
</tbody>
</table>

Details
If necessary, more details than the description above

Value
Describe the value returned If it is a LIST, use

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp1</td>
<td>Description of 'comp1'</td>
</tr>
<tr>
<td>comp2</td>
<td>Description of 'comp2'</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
grid.yaxis.log

Author(s)

who you are

References

put references to the literature/web site here

See Also

objects to See Also as help.

Examples

grid.yaxis.log Log Scale Axis for Grid/Lattice

Description

A concise (1-5 lines) description of what the function does.

Usage

grid.yaxis.log(logLim = as.numeric(convertY(unit(c(0,1), "npc"), "native")), label = T, draw = T, name = NULL, ...)
grid.xaxis.log(logLim = as.numeric(convertX(unit(c(0,1), "npc"), "native")), label = T, draw = T, name = NULL, ...)
lattice.y.prettylog(lim, ...) 
lattice.x.prettylog(lim, ...)
lattice.y.sqrt(lim, ...) 
lattice.x.sqrt(lim, ...)

Arguments

logLim Describe logLim here
label Describe label here
draw Describe draw here
name Describe name here
lim Describe lim here
... Describe ... here

Details

If necessary, more details than the description above
**Value**

Describe the value returned If it is a LIST, use

```
comp1 Description of `comp1`
comp2 Description of `comp2`
```

**Author(s)**

who you are

**References**

put references to the literature/web site here

**See Also**

objects to See Also as help.

**Examples**

```
hydrosanity-internal

*Internal Hydrosanity functions*

Description

Ignore these.

hydrosanity

*Start the Hydrosanity GUI*

Description

Start the Hydrosanity Graphical User Interface.

**Usage**

```
hydrosanity(project = NULL)
```

**Arguments**

```
project Path to a hydrosanity project file to open.
```
Details
...should list available documentation here.

Author(s)
Felix Andrews (felix@nfrac.org)

References

Package home page: http://hydrosanity.googlecode.com/

Examples
```r
## Not run:
hydrosanity()
## End(Not run)
```

Description
A concise (1-5 lines) description of what the function does.

Usage
```r
hydrosanity.caption(timelim, by, n, series = NA, x = unit(1, "npc") - unit(1, "mm"), y = unit(1, "mm"), just = c("right", "bottom"), gp = gpar(fontsize = 7, col = grey(0.5)))
```

Arguments
- `timelim` Describe `timelim` here
- `by` Describe `by` here
- `n` Describe `n` here
- `series` Describe `series` here
- `x` Describe `x` here
- `y` Describe `y` here
- `just` Describe `just` here
- `gp` Describe `gp` here
impute.timeblobs

Details
If necessary, more details than the description above

Value
Describe the value returned If it is a LIST, use

comp1 Description of 'comp1'
comp2 Description of 'comp2'
...

Author(s)
who you are

References
put references to the literature/web site here

See Also
objects to See Also as help,

Examples

```
impute.timeblobs blob.list, which.impute = names(blob.list), timelim = NULL, extend = F, extends = NA, method = ("distance", "correlation", "constant"), constant = ("mean", "zero", "extend"), trim = 0)
imputeGaps.timeblobs(blob.list, which.impute = names(blob.list), type = ("disaccumulated", "imputed"), fallBackToConstantDisaccum = T, maxGapLength = Inf, extend = F, ...)
quick.disaccumulate.timeblob(blob)
unimputeGaps.timeblobs(blob.list, timelim = NULL, type = c("imputed", "disaccumulated"))
```

Description
A concise (1-5 lines) description of what the function does.

Usage

impute.timeblobs(blob.list, which.impute = names(blob.list), timelim = NULL, extend = F, extends = NA, method = ("distance", "correlation", "constant"), constant = ("mean", "zero", "extend"), trim = 0)
imputeGaps.timeblobs(blob.list, which.impute = names(blob.list), type = ("disaccumulated", "imputed"), fallBackToConstantDisaccum = T, maxGapLength = Inf, extend = F, ...)
quick.disaccumulate.timeblob(blob)
unimputeGaps.timeblobs(blob.list, timelim = NULL, type = c("imputed", "disaccumulated"))
Arguments

blob.list  Describe blob.list here
which.impute  Describe which.impute here
timelim  Describe timelim here
extend  Describe extend here
withinTimeframe  Describe withinTimeframe here
method  Describe method here
constant  Describe constant here
trim  Describe trim here
type  Describe type here
fallBackToConstantDisaccum  Describe fallBackToConstantDisaccum here
maxGapLength  Describe maxGapLength here
...  Describe ... here
blob  Describe blob here

Details

If necessary, more details than the description above

Value

Describe the value returned If it is a LIST, use

cmp1  Description of 'comp1'
ncmp2  Description of 'comp2'
...

Author(s)

who you are

References

put references to the literature/web site here

See Also

objects to See Also as help,

Examples
panel.levelplot.interp

Panel functions for spatial layers

Description

Functions for plot-time spatial interpolation, which can be used as panel functions for levelplot. Also some generally useful spatial layers, assuming a latitude-longitude coordinate system.

Usage

panel.levelplot.mosaic <- function(x, y, z, subscripts = T,
  at = seq(min(z, na.rm = T), max(z, na.rm = T), length = 100),
  col.regions = regions$col)

panel.levelplot.interp(x, y, z, subscripts = T, xo.length = 40, yo.length = xo.length,
  linear = T, extrap = F, contour = F, region = T, at, ...)

panel.contourplot.interp(..., contour = T, region = F)

panel.worldmap(col = "black", ...)
panel.rivers(col = "blue", lty = "longdash", ...)
panel.cities(pch = 15, col = "black", ...)

prepanel.extend.10(...)

Arguments

x, y, z    x and y coordinates and z value to be interpolated.
subscripts used by Lattice for conditioning.
at    z values at which to indicate changes. May be omitted, see details below.
col.regions    a vector of colours representing levels. See levelplot.
xo.length, yo.length    resolution of interpolated surface. The plot region is divided into xo.length * yo.length cells.
linear    use bicubic rather than linear interpolation.
extrap    for linear=F, use spatial extrapolation outside the convex hull of the data.
contour    draw contour lines (passed to panel.levelplot.
region    draw shaded image (passed to panel.levelplot.
...    further arguments passed to panel.levelplot.
col, lty, pch    passed on to the usual drawing functions.
Details

For \texttt{panel.levelplot.interp}, if the \texttt{at} argument is missing it is taken as \texttt{pretty(z)} if \texttt{contour=T} and a 100-point sequence between the ranges of \textit{z} otherwise.

Value

Describe the value returned If it is a LIST, use

\begin{verbatim}
comp1 Description of 'comp1'
comp2 Description of 'comp2'
...
\end{verbatim}

Author(s)

Felix Andrews \{felix@nfrac.org\}

See Also

\texttt{levelplot.voronoi.mosaic.interp.map}

Examples

\begin{verbatim}
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random, 
##-- or do help(data=index) for the standard data sets.
\end{verbatim}

\begin{verbatim}
read.timeblob    Read Timeblob from a Text File
\end{verbatim}

Description

Read in a timeblob (time series) from a file.

Usage

\begin{verbatim}
read.timeblob(file, skip = 1, sep = ",", sitename = NULL, dataname = "Data",
dataCol = 2, qualCol = 3, extraCols = c(),
extraNames = paste("Extra", extraCols), readTimesFromFile = T,
timeCol = 1, timeFormat = "%d %b %Y", startTime = NA,
tz = "GMT", timeSeqBy = "days", timeOffset = NULL, ...)
\end{verbatim}
**Arguments**

- **file**: path to a file, or a connection.
- **skip**: number of lines in file before data begins.
- **sep**: column separator, see `read.table`.
- **sitename**: site name attribute.
- **dataname**: data name attribute.
- **dataCol**: file column number (starting from 1) containing data.
- **qualCol**: file column number (starting from 1) containing quality codes.
- **extraCols**: a vector of other file column numbers to read in.
- **extraNames**: names of the columns given above.
- **readTimesFromFile**: whether to read time series times from the file (column `timeCol`).
- **timeCol**: used when `readTimesFromFile = TRUE`. It should give the file column number (starting from 1) containing times.
- **timeFormat**: format of times in `timeCol`: see `strptime`.
- **startTime**: used when `readTimesFromFile = FALSE`. It should be (coercible to) a `POSIXt` object, or else be a list like `list(year=1,month=2,day=3)` (etc) where these give column numbers to read the start time components (from the first line in file).
- **tz**: timezone, see `POSIXt`.
- **timeSeqBy**: time step to go with `startTime`, passed to `seq.POSIXt`.
- **timeOffset**: a `difftime` object to add to the times.
- **...**: passed to `read.table`.

**Details**

**Value**

An object of class `timeblob`. It is just a `data.frame` with some extra attributes.

**Author(s)**

Felix Andrews (felix@nfrac.org)

**See Also**

`read.table`

**Examples**
**smooth.timeblob**  
*Apply smoothing filter to a timeseries object*

---

### Description

A concise (1-5 lines) description of what the function does.

### Usage

```r
smooth.timeblob(blob, by = "1 year")
```

### Arguments

- **blob**
  - *Describe blob here*

- **by**
  - *Describe by here*

### Details

If necessary, more details than the description above

### Value

Describe the value returned If it is a LIST, use

- **comp1**
  - Description of `comp1`

- **comp2**
  - Description of `comp2`

...  

### Author(s)

who you are

### References

put references to the literature/web site here

### See Also

objects to See Also as help,

### Examples
summaryMissing.timeblobs

*Summarise Missing Data in a Set of Timeblobs*

**Description**
A concise (1-5 lines) description of what the function does.

**Usage**
```r
summaryMissing.timeblobs(blob.list, timelim = NULL, timestep = NULL)
```

**Arguments**
- `blob.list`: Describe `blob.list` here
- `timelim`: Describe `timelim` here
- `timestep`: Describe `timestep` here

**Details**
If necessary, more details than the description above

**Value**
Describe the value returned If it is a LIST, use
- `comp1`: Description of `comp1`
- `comp2`: Description of `comp2`
...

**Author(s)**
who you are

**References**
put references to the literature/web site here

**See Also**
objects to See Also as `help`

**Examples**
**Synchronise Time Series (Timeblobs)**

**Description**

A concise (1-5 lines) description of what the function does.

**Usage**

```r
sync.timeblobs(blob.list, timestep = NULL, timelim = NULL, extractColumn = "Data")
syncTo.timeblobs(blob.list, blob, extractColumn = "Data")
matchtimes.timeblob(blob, times)
common.timestep.timeblobs(blob.list, default = "DSTdays")
```

**Arguments**

- `blob.list` Describe blob.list here
- `timestep` Describe timestep here
- `timelim` Describe timelim here
- `extractColumn` Describe extractColumn here
  - `blob` Describe blob here
  - `times` Describe times here
  - `default` Describe default here

**Details**

If necessary, more details than the description above

**Value**

Describe the value returned If it is a LIST, use

- `comp1` Description of `comp1`
- `comp2` Description of `comp2`
...

**Author(s)**

who you are
References
put references to the literature/web site here

See Also
objects to See Also as help.

Examples

```r
timeblob

Time Series Object (Timeblob) with Times and Data

Description
A concise (1-5 lines) description of what the function does.

Usage
timeblob(Time, Data, Qual = NULL, extras = NULL, timestep = NULL, sitename = "Unknown", dataname = "Data", role = "OTHER")
is.timeblob(x)
lapply.timeblob.data(blob.list, FUN, ...)
sapply.timeblob.data(blob.list, FUN, ...)

Arguments
- **Time** Describe Time here
- **Data** Describe Data here
- **Qual** Describe Qual here
- **extras** Describe extras here
- **timestep** Describe timestep here
- **sitename** Describe sitename here
- **dataname** Describe dataname here
- **role** Describe role here
- **x** Describe x here
- **blob.list** Describe blob.list here
- **FUN** Describe FUN here
- **...** Describe ... here

Details
If necessary, more details than the description above
**truncMonth**

**Value**
Describe the value returned. If it is a LIST, use

<table>
<thead>
<tr>
<th>comp1</th>
<th>Description of 'comp1'</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp2</td>
<td>Description of 'comp2'</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Author(s)**
who you are

**References**
put references to the literature/web site here

**See Also**
objects to See Also as help.

**Examples**

```
truncMonth  Round Down to a Month, Year or Decade
```

**Description**
A concise (1-5 lines) description of what the function does.

**Usage**

```
truncMonth(x)  
truncYear(x)   
truncDecade(x) 
```

**Arguments**

| x | Describe x here |

**Details**
If necessary, more details than the description above
Value

Describe the value returned If it is a LIST, use

comp1  Description of 'comp1'
comp2  Description of 'comp2'
...

Author(s)

who you are

References

put references to the literature/web site here

See Also

objects to See Also as help.

Examples


---

**window.timeblob**  
*Time Windows of Time Series (Timeblobs)*

---

Description

A concise (1-5 lines) description of what the function does.

Usage

```r
## S3 method for class 'timeblob':
window(x, start = NULL, end = NULL, inclusive = F, return.indices = F, extend = F, ...)

## S3 method for class 'timeblob':
start(x, ...)
## S3 method for class 'timeblob':
end(x, ...)

start.timeblobs(x, ...)
ed timeblobs(x, ...)
timelim.timeblobs(x)
```
window.timeblob

Arguments

- x Describe x here
- start Describe start here
- end Describe end here
- inclusive Describe inclusive here
- return.indices Describe return.indices here
- extend Describe extend here
- ... Describe ... here

Details

- If necessary, more details than the description above

Value

- Describe the value returned If it is a LIST, use
  - comp1 Description of 'comp1'
  - comp2 Description of 'comp2'
  - ... returns length 0 if blob is empty

Author(s)

- who you are

References

- put references to the literature/web site here

See Also

- objects to See Also as help.

Examples
Index

*Topic aplot
  hydrosanity.caption, 11
  panel.levelplot.interp, 14

*Topic dplot
  grid.xaxis.POSIXt, 8
  grid.yaxis.log, 9
  panel.levelplot.interp, 14

*Topic environment
  hydrosanity, 10

*Topic internal
  hydrosanity-internal, 10

*Topic spatial
  arealSubPolygons, 3

*Topic ts
  aggregate.timeblob, 2
  gaps, 4
  grid.timeline.plot, 5
  grid.timeseries.plot, 6
  grid.xaxis.POSIXt, 8
  impute.timeblobs, 12
  read.timeblob, 15
  smooth.timeblob, 17
  summaryMissing.timeblobs, 18
  sync.timeblobs, 19
  timeblob, 20
  truncMonth, 21
  window.timeblob, 22

addToLog (hydrosanity-internal), 10
aggregate.timeblob, 2
arealSubPolygons, 3
common.timestep.timeblobs
  (sync.timeblobs), 19

data.frame, 16
difftime, 16
end.timeblob (window.timeblob), 22
end.timeblobs (window.timeblob), 22
expand.indices (gaps), 4
gaps, 4
gpc.poly-class, 3
grid.timeline.bar
  (grid.timeline.plot), 5
grid.timeline.plot, 5
grid.timeseries.plot, 6
grid.timeseries.steps
  (grid.timeseries.plot), 6
grid.xaxis.log (grid.yaxis.log), 9
grid.xaxis.POSIXt, 8
grid.yaxis.log, 9
help, 2, 5, 6, 8–10, 12, 13, 17, 18, 20–23
hydrosanity, 10
hydrosanity-internal, 10
hydrosanity.caption, 11
impute.timeblobs, 12
imputeGaps.timeblobs
  (impute.timeblobs), 12
interp, 15
is.timeblob (timeblob), 20
lapply.timeblob.data (timeblob), 20
lastTime (gaps), 4
lattice.x.prettylog
  (grid.yaxis.log), 9
lattice.x.sqrt (grid.yaxis.log), 9
lattice.y.prettylog
  (grid.yaxis.log), 9
lattice.y.sqrt (grid.yaxis.log), 9
levelplot, 14, 15
map, 15
matchtimes.timeblob
  (sync.timeblobs), 19
INDEX

panel.cities
  (panel.levelplot.interp), 14
panel.contourplot.interp
  (panel.levelplot.interp), 14
panel.levelplot.interp, 14
panel.levelplot.mosaic
  (panel.levelplot.interp), 14
panel.rivers
  (panel.levelplot.interp), 14
panel.worldmap
  (panel.levelplot.interp), 14
POSIXt, 16
prepanel.extend.10
  (panel.levelplot.interp), 14
prepanel.qqmath.fix
  (hydrosanity-internal), 10
quick.disaccumulate.timeblob
  (impute.timeblobs), 12
read.table, 16
read.timeblob, 15
readGDAL_fixed
  (hydrosanity-internal), 10
rises (gaps), 4
sapply.timeblob.data (timeblob), 20
select.sites.BOM.AU
  (hydrosanity-internal), 10
seq.POSIXt, 16
smooth.timeblob, 17
SpatialPolygons, 3
sqrtPalette
  (hydrosanity-internal), 10
sqrtPretty
  (hydrosanity-internal), 10
start.timeblob (window.timeblob), 22
start.timeblobs
  (window.timeblob), 22
strptime, 16
summaryMissing.timeblobs, 18
sync.timeblobs, 19
syncTo.timeblobs
  (sync.timeblobs), 19
timeblob, 20
timeline.timeblobs
  (window.timeblob), 22
timelineColMapDefault
  (grid.timeline.plot), 5
truncDecade (truncMonth), 21
truncMonth, 21
truncYear (truncMonth), 21
unimputeGaps.timeblobs
  (impute.timeblobs), 12
voronoi.mosaic, 3, 15
waterQuarters
  (hydrosanity-internal), 10
window.timeblob, 22