R documentation

of '`./man/dfxco2.Rd`' etc.

February 13, 2018

`dfxco2` *Detrended XCO2 seasonal cycles for GOSAT-ACOSv3.5 and DGVM simulations*

Description

The detrended XCO2 seasonal cycle data presented herein, are the result of signal decomposition of XCO2 daily means by TransCom Region for 2009-2012. Land fluxes (NBP) underwent simulated atmospheric transport, using JAMSTEC's ACTM. XCO2 was then sampled via co-location to GOSAT observations. Both GOSAT and simulated XCO2 underwent signal decomposition, using the ccgcrv algoritm <https://www.esrl.noaa.gov/gmd/ccgg/mbl/crvfit/crvfit.html>.

Usage

```
data(dfxco2)
```

Format

A data.frame with 100136 rows and 16 variables:

- year: year (integer)
- month: month (integer)
- day: day (integer)
- func: values of the full function (harmonic+trend+short-term & long-term digital filter) (float)
- poly: values of the polynomial part of the function (float)
- smooth: values of the short-term smoothed curve; function + short-term filter of residualas (float)
- trend: values of the trend curve; polynomial plus long-term filter of residuals (float)
- detrend: values of the original data points minus the trend curve (float)
- smcycle: values of the smoothed, detrended annual cycle; smooth - trend (float)
- harm: values of the harmonic part of the function (float)
- smres: values of the short-term smoothed residuals from the function (float)
- trres: values of the long-term smoothed residual from the function (float)
- trres: values of the growth rate; the first derivative of the trend curve (float)

References

Calle, L., B. Poulter, & P.K. Patra (2018)

Examples

```
data(dfxco2)
```

`segmentTS.1matchsignal` *Match Signals*

Description

This function matches the number of events in the time series. It is based on Ehret and Zehe's (2011) conception of event-type signals. But for our purposes, we treat the full time series as a single event and break up the time series into segements categorized as separate signals. This function is generally not used as we set all data points as the same 'event', but we keep this function here for future application.

Usage

```
segmentTS.1matchsignal(obs.evnt, sim.evnt, limit4match = 0)
```

Arguments

`obs.evnt`	data.frame object with variables of start time (decimal.date), end time (decimal.date), match (integer)
`sim.evnt`	data.frame, as above, but for simulated data
`limit4match`	number to search the vector in obs.evnt,sim.evnt for similar events (set to zero)

Value

a list with both obs.evnt,sim.evnt with an additional variable for the matching index; used in segmentTS.2catsignal

---

`segmentTS.2catsignal` *Categorize Signals*

Description

This function takes in the time-series data for observed and simulated from segmentTS.1matchsignal. Categorizes portions of the curve into clear signals trough, up, no-event, down, or peak based on a difference equation to determine first and second derivatives.

Usage

```
segmentTS.2catsignal(dat, lolim = -999)
```

Arguments

`dat`	data.frame object with variables of data value; variables derived from segmentTS.1matchsignal.
`lolim`	lower limit for consideration of matching; set to low value so that all values potentially match.

Value

data.frame object with variable ’pos’, categorized as above; used in segmentTS.2eqsignal,

---

`segmentTS.3eqsignal` *Equalize Signals*

Description

This function takes in the time-series data for observed and simulated from segmentTS.2catsignal. Attempts to equalize the number of peaks and troughs in simulated time-series, to match number of signals in the observed time-series. This fn defines the boundary positions for the segments in the full time-series. Modify the criteria for removing false peaks/troughs in this section. i.e., remove short-term signals and focus on seasonal patterns of rise and fall. Also, set individual criteria to smooth signal characterization for different simulated time-series.

Usage

```
segmentTS.3eqsignal(obs.evnt, sim.evnt, val.mindays = 250,
  manual_removal = NULL)
```

Arguments

`obs.evnt`	data.frame object with variables derived from segmentTS.2catsignal
`sim.evnt`	data.frame, variables as in obs.evnt, but for simulated data.
`val.mindays`	integer number of timesteps (days) between peaks troughs; helps to remove false peaks and troughs.
`manual_removal`	section of code identifying which peak or trough to remove from the time-series. By visual inspection, if the first peak/trough is a false peak/trough, then pass code as obs.peak[c(-1),] or obs.trough[c(-1),]. Add multiple removals via obs.peak[c(-1,-2,...),]. Pass code for multiple signals with semi-colons as in manual_removal="obs.peak[c(-1),]; sim.peak[c(-1),]"

Value

list object with two data.frames with number of peaks,troughs equalized. The data frame only contains the vector positions of the peaks and troughs.

`segmentTS.4segdist` *Segment Distance*

Description

This function takes in the full time-series data for observed and simulated from segmentTS.1matchsignal. The boundaries of the segments in the time-series are defined using the positions of the major peaks and troughs from fn segmentTS.3eqsignal(). Each segment is then passed to segmentTS.4segdist to calculate the point-by-point segment statistics. Matches similar signals in the two time-series. Distance statistics are simulation minus observation.

Usage

```
segmentTS.4segdist(obs.seg, sim.seg)
```

Arguments

`obs.seg`	data.frame object with variables derived from segmentTS.1matchsignal
`sim.seg`	data.frame object, variables as in obs.seg, but for simulated data

Value

list object with 4 outputs: time-series of the matching times and values (poly_t,poly) and the distance statistics (dist_tdiff,dist_vdiff)

`segmentTS.mkdf` *Make data frames for event/signal type of time-series data*

Description

This function makes the empty data frames for storing time-series event/signal data.

Usage

```
segmentTS.mkdf(df.obs, df.sim, func.var, date.var)
```

Arguments

`df.obs`	data.frame object based on ccgcrv output for observational data. At minimum, requires variables of values and date (YYYY-MM-DD)
`df.sim`	data.frame object based on ccgcrv output for simulated data At minimum, requires variables of values and date (YYYY-MM-DD)
`func.var`	name of variable for data values. Must have same name in both df.obs and df.sim.
`date.var`	name of variable for dates. Must have same name in both df.obs and df.sim.

Value

a list object with two data.frames, df.obs.evnt, df.sim.evnt formatted with new variable names

segmentTS.statsplots *Segment-based Statistics & Figures*

Description

This function collates segment-based statistics based on the observed and simulated time-series.

Usage

```
segmentTS.statsplots(df.obs.evnt, df.sim.evnt, ls.evnt.pos, obs.name = "obs",
  sim.name = "sim", time.units = "days", val.units = NULL,
  save.plot = FALSE, outDir = getwd(), region.name = NULL)
```

Arguments

`df.obs.evnt`	data.frame object for full time-series of observational data, from segmentTS.1matchsignal.
`df.sim.evnt`	data.frame object for full time-series of simulated data, from segmentTS.1matchsignal.
`ls.evnt.pos`	list object from segmentTS.3eqsignal with positions of the peaks and troughs in the full time-series. Defines the boundaries of the segments.
`obs.name`	name of observational data for table (string).
`sim.name`	name of simulated data for table (string).
`time.units`	units of time for calculating period length in the time-series; default is days.
`val.units`	units of the value for the variable in the time-series.
`save.plot`	save plot as a pdf (TRUE/FALSE); default is FALSE; default out is the current working directory.
`outDir`	location to save plot; default location is getwd()).
`region.name`	name of underlying region for table (string). Default is 'null_region'; not important unless evaluating multiple regions. At minimum, requires variables of values and date (YYYY-MM-DD)

Value

a data.frame object segment-based statistical summaries for all segments, obs and sim.

Index