Package: AQEval (via r-universe)

October 1, 2024

Title Air Quality Evaluation

Version 0.5.7

Date 2024-04-03

Description Developed for use by those tasked with the routine detection, characterisation and quantification of discrete changes in air quality time-series, such as identifying the impacts of air quality policy interventions. The main functions use signal isolation then break-point/segment (BP/S) methods based on 'strucchange' and 'segmented' methods to detect and quantify change events (Ropkins & Tate, 2021, <doi:10.1016/j.scitotenv.2020.142374>).

Maintainer Karl Ropkins <k.ropkins@its.leeds.ac.uk>

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports openair, dplyr, loa, ggplot2, strucchange, segmented, mgcv, tidyr, lubridate, purrr, ggtext, stats

RoxygenNote 7.2.3

URL https://github.com/karlropkins/AQEval, https://karlropkins.github.io/AQEval/

BugReports https://github.com/karlropkins/AQEval/issues

NeedsCompilation no

Author Karl Ropkins [aut, cre] (<https://orcid.org/0000-0002-0294-6997>), Anthony Walker [aut] (<https://orcid.org/0000-0003-3979-0989>), James Tate [aut]

Repository CRAN

Date/Publication 2024-04-03 18:53:01 UTC

Contents

`AQEval` *Air Quality Evaluation*

Description

R AQEval: R code for the analysis of discrete change in Air Quality time-series.

AQEval

`AQEval` was developed for use by those tasked with the routine detection, characterisation and quantification of discrete changes in air quality time-series.

The main functions, `quantBreakPoints` and `quantBreakSegments`, use break-point/segment (BP/S) methods based on the consecutive use of methods in the `strucchange` and `segmented` R packages to first detection (as break-points) and then characterise and quantify (as segments), discrete changes in air-quality time-series.

`AQEval` functions adopt an `openair`-friendly approach using function and data structures that many in the air quality research community are already familiar with. Most notably, most functions expect supplied data to be time-series, to be supplied as a single `data.frame` (or similar R object), and for time-series to be identified by column names. The main functions are typically structured expect first the `data.frame`, then the name of the pollutant to be used, then other arguments:

```
function(data, "polluant.name", ...)
```

```
output <- function(data, "polluant.name", ...)
```

Author(s)

Karl Ropkins

References

Ropkins et al (In Prep).

See Also

For more about data structure and an example data set, see AQEval.data

For more about the main functions, see quantBreakPoints and quantBreakSegments

---

AQEval.data *AQEval Example data*

Description

Data packaged with AQEval for use with example code.

Usage

```
aq.data
```

Format

(26280x6) 'tbl_df' objects

date Time-series of POSIX class date and time records.

no2 Time-series of nitrogen dioxide measurements from local site.

bg.no2 Time-series of nitrogen dioxide measurements from nearby background site.

ws Time-series of local wind speed measurements.

wd Time-series of local wind direction measurements.

air_temp Time-series of local air temperature measurements.

Details

Most of functions in AQEval adopt the openair convention of assuming supplied data is a single data.frame or similar. The data frame was initially adopted for two reasons:

- Firstly, air quality data collected and archived in numerous formats and keeping the import requirements simple minimises the frustrations associated with data importation.
- Secondly, restricting the user to work with a single data format greatly simplifies data management for those less familiar with programming environments.

As part of this work several openair coding conventions were adopted, most importantly that data sets should include a column named date of POSIX class data-and-time-stamps (DateTimeClasses). This and other conventions, such as the use of ws and wd for numeric wind speed and direction data-series, and site and code for character or factor monitoring site name and identifier code, are now commonplace for many working with R in the air quality research community, and many air quality archives provide data in (or support import functions that convert their own data structures to) this openair-friendly structure.

Source

Air quality and meteorological data packaged for use with AQEval Examples.

Time-series sources:

- **date** Date-and-time-stamp of POSIX class (DateTimeClasses).
- **no2** Nitrogen dioxide downloaded from King's College London Archive using importKCL function in openair.
- **bg.no2** Nitrogen dioxide downloaded from the Automatic Urban and Rural Network Archive using importAURN function in openair.
- **ws**, **wd**, **air_temp** Wind speed, wind direction and air temperature downloaded from NOAA's Integrated Surface Database using importNOAA function in worldmet.

References

Regarding openair and openair-friendly data structuring, see:

Carslaw, D. C. and K. Ropkins (2012), openair — an R package for air quality data analysis. Environmental Modelling & Software. Volume 27-28, 52-61, DOI doi:10.1016/j.envsoft.2011.09.008

Ropkins, K. and D.C. Carslaw (2012), openair-Data Analysis Tools for the Air Quality Community. R Journal, 4(1). URL https://journal.r-project.org/archive/2012/RJ-2012-003/RJ-2012-003.pdf

Regarding worldmet, see:

David Carslaw (2021), worldmet: Import Surface Meteorological Data from NOAA Integrated Surface Database (ISD). R package version 0.9.5. URL https://CRAN.R-project.org/package=worldmet

See Also

DateTimeClasses

openair: functions importAURN and importKCL

worldmet: function importNOAA (See References)

Examples

```
#data set used in AQEval Examples
dim(aq.data)
head(aq.data)
with(aq.data, plot(date, no2, type="l"))
```

`calculate.stats` *Some functions to calculate statistics*

Description

Calculate data set statistics for selected time intervals.

Usage

```
calcDateRangeStat(
  data,
  from = NULL,
  to = NULL,
  stat = NULL,
  pollutant = NULL,
  ...,
  method = 2
)

calcRollingDateRangeStat(
  data,
  range = "year",
  res = "day",
  stat = NULL,
  pollutant = NULL,
  from = NULL,
  to = NULL,
  ...,
  method = 2
)
```

Arguments

`data`	(data.frame, tibble, etc) Data set containing data statistic to be calculated for, and `date` column of date/time records.
`from`	(various) Start date(s) to subsample from when calculating statistic, by default end of supplied `data` date range.
`to`	(various) End date(s) to subsample to when calculating statistic, by default end of supplied `data` date range.
`stat`	(function) Statistic to be applied to selected data, by default `mean(pollutant, na.rm=TRUE)`.
`pollutant`	(character) The name(s) of data-series to analyse in `data`, by default all columns in supplied data except `date`.
`...`	extra arguments.
`method`	(numeric) Method to use when calculating statistic.

range	(character) For calcRollingDateRange, the range the rolling date windows, by default 'year' for annual statistic calculations.
res	(character) For calcRollingDateRange, the resolution to calculate the rolling statistic at, by default 'day' to calculate this once per day.

Value

These functions return data.frames of function outputs.

Note

These functions are in development and likely to change significantly in future versions, please handle with care.

---

find.breaks *find and test break-points*

Description

Finding and testing break-points in conventionally formatted air quality data sets.

Usage

```
findBreakPoints(data, pollutant, h = 0.15, ...)

testBreakPoints(data, pollutant, breaks, ...)
```

Arguments

data	Data source, typically a data.frame or similar, containing data-series to apply function to and a paired time-stamped data-series, called date.
pollutant	Name of time-series, assumed to be a column in date.
h	(findBreakPoints only) The data/time window size to use when looking for breaks in a supplied time-series, expressed as proportion of time-series (0-1), default 0.15.
...	other parameters
breaks	(testBreakPoints only) data.frame of The break-points and confidence intervals, typically a findBreakPoints output.

Details

findBreakPoints uses methods from strucchange package (see references) and modifications as suggested by the main author of strucchange to handle missing cases to find potential breaks-points in a supplied time-series.

testBreakPoints tests and identifies most likely break-points using methods proposed for use with quantBreakPoints and quantBreakSegments and conventionally formatted air quality data sets.

Value

`findBreakPoints` returns a `data.frame` of found break-points.

`testBreakPoints` return a likely break-point/segment report.

References

Regarding `strucchange` methods see `breakpoints`, and:

Achim Zeileis, Friedrich Leisch, Kurt Hornik and Christian Kleiber (2002). strucchange: An R Package for Testing for Structural Change in Linear Regression Models. Journal of Statistical Software, 7(2), 1-38. URL https://www.jstatsoft.org/v07/i02/.

Achim Zeileis, Christian Kleiber, Walter Kraemer and Kurt Hornik (2003). Testing and Dating of Structural Changes in Practice. Computational Statistics & Data Analysis, 44, 109-123.

Regarding missing data handling, see:

URL: https://stackoverflow.com/questions/43243548/strucchange-not-reporting-breakdates.

Regarding `testBreakPoints`, see:

Ropkins et al (In Prep).

See Also

`find.breaks`.

`find.near` *find nearby sites*

Description

Function to find nearest locations in a reference by latitude and longitude.

Usage

```
findNearLatLon(lat, lon = NULL, nmax = 10, ..., ref = NULL, units = "m")

findNearSites(
  lat,
  lon,
  pollutant = "no2",
  site.type = "rural background",
  nmax = 10,
  ...,
  ref = NULL,
  units = "m"
)
```

Arguments

lat, lon	(numeric) The supplied latitude and longitude.
nmax	(numeric) The maximum number of nearest sites to report, by default 10.
...	Other parameters, currently ignored.
ref	(data.frame or similar) The look-up table to use when identifying nearby locations, and expected to contain latitude, longitude and any required location identifier data-series. By default, findNearSites uses openair importMeta output if this is not supplied but this is a required input for findNearLatLon.
units	(character) The units to use when reporting distances to near locations; current options m.
pollutant	(character) For findNearSites only, the pollutant of interest, by default NO2.
site.type	(character) For findNearSites only, the monitoring site type, by default Rural Background.

Details

If investigating air quality in a particular location, for example a UK Clean Air Zone (https://www.gov.uk/guidance/driving-in-a-clean-air-zone), you may wish to locate an appropriate rural background air quality monitoring station. findNearSites locates air quality monitoring sites with openly available data such as that available from the UK AURN network (https://uk-air.defra.gov.uk/networks/network-info?view=aurn)

Value

find.near returns data.frame of near site meta data.

Note

This function uses haversine formula to account to the Earth's surface curvature, and uses 6371 km as the radius of earth.

Examples

```
#find rural background NO2 monitoring sites
#near latitude = 50, longitude = -1

#not run: requires internet
## Not run:
findNearSites(lat = 50, lon = -1)

## End(Not run)
```

`isolate.signal` *isolateContribution*

Description

Environmental time-series signal processing: Contribution isolation based on background subtraction, deseasonalisation and/or deweathering.

Usage

```
isolateContribution(
  data,
  pollutant,
  background = NULL,
  deseason = TRUE,
  deweather = TRUE,
  method = 2,
  add.term = NULL,
  formula = NULL,
  output = "mean",
  ...
)
```

Arguments

`data`	Data source, typically `data.frame` (or similar), containing all time-series to be used when applying signal processing.
`pollutant`	The column name of the `data` time-series to be signal processed.
`background`	(optional) if supplied, the background time-series to use as a background correction. See below.
`deseason`	logical or character vector, if `TRUE` (default), the `pollutant` is deseasonalised using `day.hour` and `year.day` frequency terms, all calculate from the `data` time stamp, assumed to be `date` in `data`. Other options: `FALSE` to turn off deseasonalisation; or a character vector of frequency terms if user-defining. See below.
`deweather`	logical or character vector, if `TRUE` (default), the data is deweathered using wind speed and direction, assumed to be `ws` and `wd` in `data`). Other options: `FALSE` to turn off deweathering; or a character vector of `data` column names if user-defining. See below.
`method`	numeric, contribution isolation method (default 2). See Note.
`add.term`	extra terms to add to the contribution isolation model; ignore for now (in development).
`formula`	(optional) Signal isolate model formula; this allows user to set the signal isolation model formula directly, but means function arguments `background`, `deseason` and `deweather` will be ignored.

output	output options; currently, 'mean', 'model', and 'all'; but please note these are in development and may be subject to change.
...	other arguments; ignore for now (in development)

Details

isolateContribution estimates and subtracts pollutant variance associated with factors that may hinder break-point/segment analysis:

- **Background Correction** If applied, this fits the supplied background time-series as a spline term: s(background).
- **Seasonality** If applied, this fits regular frequency terms, e.g. day.hour, year.day, as spline terms, default TRUE is equivalent to s(day.hour) and s(year.day). All terms are calculated from date column in data.
- **Weather** If applied, this fits time-series of identified meteorological measurements, e.g. wind speed and direction (ws and wd in data). If both ws and wd are present these are fitted as a tensor term te(ws, wd). Other deweathering terms, if included, are fitted as spline term s(term). The default TRUE is equivalent to te(ws, wd).

Using the supplied arguments, it builds a signal (mgcv) GAM model, calculates, and returns the mean-centred residuals as an estimate of the isolated local contribution.

Value

isolateContribution returns a vector of predictions of the pollutant time-series after the requested signal isolation.

Note

method was included as part of method development and testing work, and retained for now. Please ignore for now.

Author(s)

Karl Ropkins

References

Regarding mgcv GAM fitting methods, see Wood (2017) for general introduction and package documentation regarding coding (mgcv):

Wood, S.N. (2017) Generalized Additive Models: an introduction with R (2nd edition), Chapman and Hall/CRC.

Regarding isolateContribution, see:

Ropkins et al (In Prep).

See Also

Regarding seasonal terms and frequency analysis, see also stl and spectralFrequency.

mgcv, gam.

Examples

```
#fitting a simple deseasonalisation, deweathering
#and background correction (dswb) model to no2:

aq.data$dswb.no2 <- isolateContribution(aq.data,
                        "no2", background="bg.no2")

#compare at 7 day resolution:
temp <- openair::timeAverage(aq.data, "7 day")

#without dswb
quantBreakPoints(temp, "no2", test=FALSE, h=0.1)

#with dswb
quantBreakPoints(temp, "dswb.no2", test=FALSE, h=0.1)
```

---

other.aq.models *Other Air Quality Models*

Description

Other packaged Air Quality Models.

Usage

```
fitNearSiteModel(data, pollutant = "no2", y, x = "rest", elements = NULL, ...)
```

Arguments

data	data.frame (or similar) containing data-series to be modelled; this is expected to contain 'date', 'site' and pollutant of interest data-series.
pollutant	The name of the pollutant (in data) to model, by default 'NO2'.
y	The name of the monitor site to be modelled, assumed to be one several names in the site column of data.
x	The other sites to use when building the model, the default 'rest' uses all supplied sites except 'y'.
elements	The number of inputs to use in the site models, can be any number up to length of x or combination thereof; by default this is set as length(x):1
...	extra arguments.

Details

fitNearSiteModel builds an air quality model for one location using air quality data from nearby sites.

Value

data with model output added as additional column.

quantify.breaks *quantify break-point/segments*

Description

Quantify either break-points or break-segment methods for pollutant time-series

Usage

```
quantBreakPoints(
  data,
  pollutant,
  breaks,
  ylab = NULL,
  xlab = NULL,
  pt.col = c("lightgrey", "darkgrey"),
  line.col = "red",
  break.col = "blue",
  event = NULL,
  show = c("plot", "report"),
  ...
)

quantBreakSegments(
  data,
  pollutant,
  breaks,
  ylab = NULL,
  xlab = NULL,
  pt.col = c("lightgrey", "darkgrey"),
  line.col = "red",
  break.col = "blue",
  event = NULL,
  seg.method = 2,
  seg.seed = 12345,
  show = c("plot", "report"),
  ...
)
```

Arguments

data	Data source, typically a data.frame or similar, containing data-series to model and a paired time-stamp data-series, named date.
pollutant	The name of the data-series to break-point or break-segment model.
breaks	(Optional) The break-points and confidence intervals to use when building either break-point or break-segment models. If not supplied these are build using findBreakPoints and supplied arguments.

ylab	Y-label term, by default pollutant.
xlab	X-label term, by default date.
pt.col	Point fill and line colours for plot, defaults lightgrey and darkgrey.
line.col	Line colour for plot, default red.
break.col	Break-point/segment colour for plot, default blue.
event	An optional list of plot terms for an event marker, applied to a vertical line and text label. List items include: x the event date (YYYY-MM-DD format) require for both line and label; y by default 0.9 x y-plot range; label the label text, required for label; line.size the line width, by default 0.5; font.size the text size, by default 5; and, hjust the label left/right justification, 0 left, 0.5 centre, 1 right (default). See also examples below.
show	What to show before returning the break-point quantification mode, by default plot and report.
...	other parameters
seg.method	(quantBreakSegments only) the break-segment fitting method to use.
seg.seed	(quantBreakSegments only) the seed setting to use when fitting break-segments, default 12345.

Details

quantBreakPoints and quantBreakSegments both use strucchange methods to identify potential break-points in time-series, and then quantify these as conventional break-points or break-segments, respectively:

- **Finding Break-points** Using the strucchange methods of Zeileis and colleagues and independent change detection model, the functions apply a rolling-window approach, assuming the first window (or data subset) is without change, building a statistical model of that, advancing the window, building a second model and comparing these, and so on, to identify the most likely points of change in a larger data-series. See also findBreakPoints
- **Quantifying Break-points** Using the supplied break-points to build a break-point model.
- **Quantifying Break-segments** Using the confidence regions for the supplied break-points as the starting points to build a break-segment model.

Value

Both functions use the show argument to control which elements of the functions outputs are shown but also invisible return a list of all outputs which can caught using, e.g.:

```
brk.mod <- quantBreakPoints(data, pollutant)
```

Note

AQEval function quantBreakSegments is currently running segmented v.1.3-4 while we evaluate latest version, v.1.4-0.

Author(s)

Karl Ropkins

References

Regarding strucchange methods see in-package documentation, e.g. breakpoints, and:

Achim Zeileis, Friedrich Leisch, Kurt Hornik and Christian Kleiber (2002). strucchange: An R Package for Testing for Structural Change in Linear Regression Models. Journal of Statistical Software, 7(2), 1-38. URL https://www.jstatsoft.org/v07/i02/.

Achim Zeileis, Christian Kleiber, Walter Kraemer and Kurt Hornik (2003). Testing and Dating of Structural Changes in Practice. Computational Statistics & Data Analysis, 44, 109-123. DOI doi:10.1016/S01679473(03)000306.

Regarding segmented methods see in-package documentation, e.g. segmented, and:

Vito M. R. Muggeo (2003). Estimating regression models with unknown break-points. Statistics in Medicine, 22, 3055-3071. DOI 10.1002/sim.1545.

Vito M. R. Muggeo (2008). segmented: an R Package to Fit Regression Models with Broken-Line Relationships. R News, 8/1, 20-25. URL https://cran.r-project.org/doc/Rnews/.

Vito M. R. Muggeo (2016). Testing with a nuisance parameter present only under the alternative: a score-based approach with application to segmented modelling. J of Statistical Computation and Simulation, 86, 3059-3067. DOI 10.1080/00949655.2016.1149855.

Vito M. R. Muggeo (2017). Interval estimation for the breakpoint in segmented regression: a smoothed score-based approach. Australian & New Zealand Journal of Statistics, 59, 311-322. DOI 10.1111/anzs.12200.

Regarding break-points/segment methods, see:

Ropkins et al (In Prep).

See Also

timeAverage in openair, breakpoints in strucchange, and segmented in segmented.

Examples

```
#using openair timeAverage to covert 1-hour data to 1-day averages

temp <- openair::timeAverage(aq.data, "1 day")

#break-points

quantBreakPoints(temp, "no2", h=0.3)

#break-segments

quantBreakSegments(temp, "no2", h=0.3)

#addition examples (not run)
## Not run:
#in-call plot modification
#removing x axis label
#recolouring break line and
#adding an event marker
quantBreakPoints(temp, "no2", h=0.3,
```

```
        xlab="", break.col = "red",
        event=list(label="Event expected here",
                  x="2002-08-01", col="grey"))

## End(Not run)
```

`spectral.analysis` *Spectral Analysis*

Description

Time-series spectral frequency analysis.

Usage

```
spectralFrequency(data, pollutant, ...)
```

Arguments

`data`	`data.frame` holding data to be analysed, expected to contain a timestamp data-series called `date` and a measurement time-series to be analysed identified using the `pollutant` argument.
`pollutant`	The name of the time-series, typically pollutant measurements, to be analysed.
`...`	extra arguments.

Details

`spectralFrequency` producing a time frequency analysis of the requested `pollutant`.

Value

`spectralFrequency` uses the `show` argument to control which elements of the functions outputs are shown but also invisibly returns a `list` of all outputs which can caught using, e.g.:

`sfa.mod <- spectralFrequency(data, pollutant)`

Examples

```
spectralFrequency(aq.data, "no2")
```

Index