Appendix A

Package 'Trends'

November 5, 2014

Version 0.2-1

Date 2014-11-05

Title Trend analysis of Monitoring Network Data

Author Jason C. Fisher and Linda C. Davis

Maintainer Jason C. Fisher <jfisher@usgs.gov>

Depends R (>= 3.1.0)

Imports survival

Suggests rgdal, knitr

SystemRequirements PDFtk Server (>= 2.02, optional)

Description This package is for identifying trends in data from multiple observation sites. A parametric survival regression model is fit to the observed data, both censored and uncensored.

License file LICENSE

URL https://github.com/jfisher-usgs/Trends

BugReports https://github.com/jfisher-usgs/Trends/issues

ByteCompile yes

VignetteBuilder knitr

R topics documented:

`DrawPlot` *Draw Single Plot*

Description

This function draws a single time-series plot on the active graphics device.

Usage

```
DrawPlot(d, model, plim = c(0.1, 0.9), xlim = NULL, ylim = NULL,
         main = NULL, ylab = "")
```

Arguments

`d`	`data.frame`; observations, where the first and second column is of class `Date` and `Surv`, respectively. The `Surv` component is of type *interval* and should not contain right-censored data.
`model`	`survreg`; a survival regression model.
`plim`	`numeric`; the percentile limits defining the confidence band of the regression model.
`xlim`	`Date`; the `x` limits of the plot.
`ylim`	`numeric`; the `y` limits of the plot.
`main`	`character`; a main title for the plot.
`ylab`	`character`; a label for the `y` axis.

Details

Uncensored data is drawn as points. Left- and interval-censored data is drawn as vertical lines with short-horizontal lines at the interval bounds. Note that the lower bound of left-censored data is placed at zero. The 50th percentile of the regression model is drawn as a curved line. The confidence band is drawn as a solid polygon.

Author(s)

J.C. Fisher

See Also

`RunAnalysis`, `predict.survreg`

Examples

```
# Create time-series data set with uncensored, left-censored, and interval-censored data
n <- 30
x <- as.Date(sort(sample(1:1000, n)), origin = as.Date("1992-02-15"))
time1 <- runif(n, min = 0, max = 100)
time2 <- time1 + runif(n, min = 0, max = 10)
time1[sample(1:n, 10)] <- NA
idxs <- sample(1:n, 10)
time1[idxs] <- time2[idxs]
y <- survival::Surv(time1, time2, type = "interval2")
d <- data.frame(x, y)

# Fit the data with a regression model
model <- survival::survreg(y ~ x, data = d)

# Plot the data and regression model
DrawPlot(d, model, plim = c(NA, 0.8), main = "Title", ylab = "Value")
```

`MergePDFs` *Merge PDF Files*

Description

This function combines Portable Document Format (PDF) files into a single new PDF file.

Usage

```
MergePDFs(path, pdfs, preserve.files = FALSE, open.file = FALSE)
```

Arguments

`path`	`character`; the path name of the folder containing the PDF files to merge.
`pdfs`	`character`; a vector of file names, if missing, all PDF files under `path` will be merged.
`preserve.files`	`logical`; if `TRUE` all individual PDF files are preserved after a merge is completed.
`open.file`	`logical`; if `TRUE` the merged PDF file is opened using your systems default PDF viewer.

Details

Names of the individual PDF files are used as bookmarks in the merged file. The merged file is placed one directory above the `path` folder.

Value

The name of the merged file is returned.

Note

Requires PDFtk Server, a cross-platform command-line tool for working with PDFs.

Author(s)

J.C. Fisher

See Also

RunAnalysis, system

Examples

```
# Create a temporary directory
dir.create(path <- file.path(tempdir(), "merge"))

# Write three single-page PDF files to the temporary directory
pdf(file.path(path, "f1.pdf"))
plot(seq_len(10), main = "f1a")
plot(sin, -pi, 2 * pi, main = "f1b")
plot(qnorm, col = "red", main = "f1c")
dev.off()
pdf(file.path(path, "f2.pdf"))
plot(table(rpois(100, 5)), type = "h", col = "yellow", main = "f2a")
dev.off()
pdf(file.path(path, "f3.pdf"))
plot(x <- sort(rnorm(47)), type = "s", col = "green", main = "f3a")
plot(x, main = "f3b")
dev.off()

# Merge PDF files into a single file and open it in your default viewer
MergePDFs(path, open.file = TRUE)

# Remove PDF files
unlink(path, recursive = TRUE)
```

ProcessConfig *Process Configuration*

Description

This function processes configuration records for data analysis.

Usage

```
ProcessConfig(config, processed.obs)
```

Arguments

config	data.frame; see 'Details' section.
processed.obs	list; see documentation for ProcessObs function for details.

Details

Required columns in the config data frame include: "Site_id", a unique site identifier; "Site_name", a local site name; and "Parameter_id", unique parameter identifier(s). The config data table is composed of character-class components. Multiple parameter identifiers can be specified in a single value of config$Parameter_id using a comma separator. Site and parameter identifiers not found in the processed.obs list are removed.

Value

Returns a data.frame object with the following components:

Site_id	numeric; a unique site identifier.
Site_name	character; a local site name.
Parameter_id	character; a unique parameter identifier.
row	integer; the row index in the config data table.

Author(s)

J.C. Fisher

See Also

RunAnalysis

ProcessObs *Process Observations*

Description

This function processes measurements from observation sites in a monitoring network.

Usage

```
ProcessObs(observations, parameters, detection.limits = NULL,
           date.fmt = "%Y-%m-%d")
```

Arguments

observations	data.frame; the observed data records, see 'Details' section.
parameters	data.frame; the parameter descriptions, see 'Details' section.
detection.limits	data.frame; the detection limits, see 'Details' section.
date.fmt	character; the date format used to convert character strings to Date-class.

Details

The observations, parameters, and detection.limits data tables are composed of character-class components.

Required columns in the observations data table include: "Site_id", a unique site identifier; "Site_name", a local site name; and "Date", the measurement date. Measured values and their uncertainty (reported as standard deviation) are located in the subsequent columns; a unique parameter identifier, or parameters standard deviation identifier, is specified for each of these column names. Measured values are checked for an optional character code in the first digit of each character string value. Character codes are identified using the following criteria: "<", below reporting level; "E", estimated value; "V", contaminated; and "U", undetectable. Values are stripped of their character code and converted to numeric-class. A warning is given if the character code is not recognized and its value set to NA. Measured values with character codes of "V" and "U" are also set to NA.

Required columns in the parameters data table include: "Parameter_id", a unique parameter identifier; "Parameter_name", the common parameter name; "Units", the units associated with the measured parameter values; and "sd", a column name in observations data table where the parameters standard deviation values are located.

A required column in the detection.limits data table is "Date", the date when the detection limit was first implemented. Detection limit values are located in subsequent columns; a unique parameter identifier is specified for each of these column names.

A measured value is converted to censored data under the following conditions: (1) the measured value is below the reporting level and represented as *left-censored* data; or (2) there is a standard deviation and detection limit associated with the measured value, therefore, it is represented as *interval-censored* data. The upper and lower bounds of interval-censored data are calculated by adding and subtracting three standard deviations from the measured value, respectively. Note that interval-censored data with a lower or upper bound less than the detection limit is represented as left-censored data. For left-censored data, the upper bound is set to the detection limit when its magnitude is less than the detection limit.

Value

Returns an object of class list with data.frame components corresponding to unique parameter identifiers. Each data table has the following components:

Site_id	numeric; a unique site identifier.
Site_name	character; a local site name.
Date	Date; the observation date.
code	factor; single-digit character code.
value	numeric; the measured value.
sd	numeric; the standard deviation of the measured value.
dl	numeric; the detection limit of the measured value.
surv	Surv; the observation as censored or uncensored data of *interval* type.

Additional attributes associated with the returned data frame include: Parameter_id, a unique parameter identifier; Parameter_name, a parameter name; and Units, the parameter units.

Author(s)

J.C. Fisher

See Also

RunAnalysis

ProcessWL *Process Water Levels*

Description

This function processes water levels for data analysis.

Usage

```
ProcessWL(water.levels, date.fmt = "%Y-%m-%d %H:%M")
```

Arguments

water.levels	data.frame; see 'Details' section.
date.fmt	character; the date format used to convert character strings to Date-class.

Details

Required columns in the water.levels data table include: "SITE_NO", a unique site identifier; "LEV_DT", the measurement date-time; "ALT_VA", referenced land-surface elevation; and "LEV_VA", depth below land surface to the water table. The water.levels data table is composed of character-class components.

Value

Returns a data.frame object with the following components:

Site_id	factor; a unique site identifier.
Date	Date; the measurement date.
Var	numeric; the water-level elevation.

Author(s)

J.C. Fisher

See Also

RunAnalysis

RunAnalysis *Run Trend Analysis*

Description

This function analyses observations for a significant trend.

Usage

```
RunAnalysis(processed.obs, processed.config, path, id, sdate = NA, edate = NA,
            control = survreg.control(iter.max = 100), sig.level = 0.05,
            graphics.type = "", merge.pdfs = TRUE, site.locations = NULL,
           is.seasonality = FALSE, explanatory.var = NULL, is.residual = FALSE,
            thin.obs.mo = NULL)
```

Arguments

processed.obs	list; see documentation for ProcessObs function for details.
processed.config	data.frame; see documentation for ProcessConfig function for details.
path	character; the path name of the folder where output data is written.
id	character; an analysis identifier that is used to construct output file names.
sdate, edate	Date or character; the start and end date corresponding to the period of interest. The required date format is YYYY-M-D (%Y-%m-%d).
control	list; the regression control values in the format produced by the survreg.control function.
sig.level	numeric; the significance level to be coupled with the *p*-value, see 'Value' section.
graphics.type	character; the graphics type for plot figures. The default is the 'active' device, typically the normal screen device. A file-based device can be selected by specifying either "pdf" or "postscript".
merge.pdfs	logical; if TRUE and graphics.type = "pdf" the figures are combined into a single PDF file, see documentation for MergePDFs function for details.
site.locations	SpatialPointsDataFrame; the geo-referenced site coordinates with a required data.frame component of "Site_id", a unique site identifier.
is.seasonality	logical; if TRUE, seasonal patterns are modeled by a trigonometric regression; as covariates in the trend model.
explanatory.var	data.frame; an explanatory variable added to the covariates of the trend model, see value from the ProcessWL function for the data table format. Note that explanatory variable values are linearly interpolated at sample dates in processed.obs.
is.residual	logical; if TRUE, the explanatory variable is transformed using its residuals from linear regression. Should be used when the explanatory variable is monotonically increasing or decreasing during the entire trend period. Requires specification of the explanatory.var argument.

`thin.obs.mo` `character`; the full name of a calendar month; if specified, data is thinned to one observation per year collected during this month. Allows verification that the variable sampling frequencies don't substantially affect the trend results. Thinning the data also could remove serial correlation in the more frequently sampled years.

Details

The `survreg` function is used to fit a parametric survival regression model to the observed data, both censored and uncensored. The specific class of survival model is known as the accelerated failure time (AFT) model. A maximum-likelihood estimation (MLE) method is used to estimate parameters in the AFT model. The MLE is solved by maximizing the log-likelihood using the Newton-Raphson method, an iterative root-finding algorithm. The likelihood function is dependent on the distribution of the observed data. Data is assumed to follow a log-normal distribution because most of the variables have values spanning two or more orders of magnitude. Note that if all observations are uncensored, the survival regression becomes identical to ordinary least squares regression.

Value

This function returns a `data.frame` object with the following components:

`Site_id`	`numeric`; a unique site identifier.
`Site_name`	`character`; a local site name.
`Parameter_id`	`character`; a unique parameter identifier.
`Parameter_name`	`character`; a common parameter name.
`sdate, edate`	`Date`; the start and end date corresponding to the period of interest.
`n`	`integer`; the number of observations in the analysis.
`nmissing`	`integer`; the number of missing values.
`nexact`	`integer`; the number of exact (uncensored) observations.
`nleft`	`integer`; the number of left-censored observations.
`ninterval`	`integer`; the number of interval-censored observations.
`nbelow.rl`	`integer`; the number of observations that are below the reporting level.
`min, max`	`numeric`; the minimum and maximum.
`median`	`numeric`; the median.
`mean, sd`	`numeric`; the mean and standard deviation, set to `NA` if censored data is present.
`iter`	`numeric`; the number of Newton-Raphson iterations required for convergence. If `NA`, the regression failed or ran out of iterations and did not converge.
`slope`	`numeric`; the slope of the linear trend over time in percent change per year.
`std.err`	`numeric`; the standard error for the linear trend over time in percent change per year.
`p`	`numeric`; the *p*-value for the linear trend over time.
`p.model`	`numeric`; the *p*-value for the parametric survival regression model.

`trend` `character`; significant trends are indicated by a *p*-value (`p`) less than or equal to the significance level. The sign of the `slope` indicates whether the significant trend is positive (`"+"`) or negative (`"-"`). *p*-values greater than the significance level are specified as having no significant trend (`"none"`).

If arguments `path` and `id` are specified, the returned data table of summary statistics (described above) is written to an external text file. If in addition a file-based graphics type is selected, plots are drawn to external files.

Author(s)

J.C. Fisher and L.C. Davis

See Also

DrawPlot

Examples

```
# Specify global arguments for reading table formatted data in a text file
read.args <- list(header = TRUE, sep = "\t", colClasses = "character",
                  na.strings = "", fill = TRUE, strip.white = TRUE,
                  comment.char = "", flush = TRUE, stringsAsFactors = FALSE)

# Read input files
path.in <- system.file("extdata", "SIR2014", package = "Trends")
file <- file.path(path.in, "Observations.tsv")
observations <- do.call(read.table, c(list(file), read.args))
file <- file.path(path.in, "Parameters.tsv")
parameters <- do.call(read.table, c(list(file), read.args))
file <- file.path(path.in, "Detection_Limits.tsv")
detection.limits <- do.call(read.table, c(list(file), read.args))
file <- file.path(path.in, "Config_VOC.tsv")
config <- do.call(read.table, c(list(file), read.args))

# Process observations
processed.obs <- ProcessObs(observations, parameters, detection.limits,
                            date.fmt = "%m/%d/%Y")

# Plot data for a single parameter at a specific site
d <- processed.obs[["P32102"]]
d <- d[d$Site_id == "433002113021701", c("Date", "surv")]
DrawPlot(d, main = "RWMC Production", ylab = "Carbon Tetrachloride")

# Configure sites, parameters, and duration for analysis
processed.config <- tail(ProcessConfig(config, processed.obs))

# Run analysis
stats <- RunAnalysis(processed.obs, processed.config,
                     sdate = "1987-01-01", edate = "2012-12-31")
```

Index

Processing Instructions for the 2014 Trend Analysis

Jason C. Fisher

November 5, 2014

If R (version $\geq$ 3.1) is not already installed on your computer, download and install the latest binary distribution from the Comprehensive R Archive Network (CRAN). Open an **R** session and install the required packages with the following commands:

```
repos <- c("http://cran.us.r-project.org", "http://jfisher-usgs.github.com/R")
update.packages(ask = FALSE, repos = repos[1])
install.packages("Trends", repos = repos, dependencies = TRUE, type = "both")
```

Support for merging Portable Document Format (PDF) files into a new file requires PDFtk Server, a cross-platform command-line tool for working with PDFs; download and install this software.

Load the Trends package into the current **R** session:

```
library(Trends)
```

Set the full path name to the directory containing input files; list all files in this directory:

```
list.files(path.in <- system.file("extdata", "SIR2014", package = "Trends"))
```

```
#  [1] "Config_Field.tsv"     "Config_NWQL.tsv"      "Config_RAD.tsv"
#  [4] "Config_VOC.tsv"       "Detection_Limits.tsv" "Observations.tsv"
#  [7] "Parameters.tsv"       "Site_Locations.dbf"   "Site_Locations.prj"
# [10] "Site_Locations.shp"   "Site_Locations.shx"   "Tables.xlsx"
# [13] "Water_Levels.tsv"
```

Set the full path name to the output directory; if it doesn't already exist, create this directory:

```
path.out <- file.path(getwd(), paste0("Trends_", format(Sys.time(), "%Y%m%d%H%M%S")))
dir.create(path = path.out, showWarnings = FALSE)
```

Set the graphics type for output figures to a PDF file format:

```
graphics.type <- "pdf"
```

Specify global arguments for reading table formatted data from a text file:

```
read.args <- list(header = TRUE, sep = "\t", colClasses = "character", na.strings = "",
                  fill = TRUE, strip.white = TRUE, comment.char = "", flush = TRUE,
                  stringsAsFactors = FALSE)
```

Read observational data from a text file:

```
file <- file.path(path.in, "Observations.tsv")
observations <- do.call(read.table, c(list(file), read.args))
```

Read parameter descriptions from a text file:

```
file <- file.path(path.in, "Parameters.tsv")
parameters <- do.call(read.table, c(list(file), read.args))
```

Read detection limits from a text file:

```
file <- file.path(path.in, "Detection_Limits.tsv")
detection.limits <- do.call(read.table, c(list(file), read.args))
```

Process observational data:

```
processed.obs <- ProcessObs(observations, parameters, detection.limits,
                            date.fmt = "%m/%d/%Y")
```

Read geo-referenced site locations from a point shapefile:

```
site.locations <- rgdal::readOGR(path.in, layer = "Site_Locations", verbose = FALSE)
```

Read water levels from a text file:

```
file <- file.path(path.in, "Water_Levels.tsv")
water.levels <- do.call(read.table, c(list(file), read.args))
```

Process water-level data:

```
processed.wl <- ProcessWL(water.levels, date.fmt = "%Y-%m-%d %H:%M")
```

Run trend analysis for National Water Quality Laboratory (NWQL) parameters from 1989 through 2012:

```
file <- file.path(path.in, "Config_NWQL.tsv")
config <- do.call(read.table, c(list(file), read.args))
processed.config <- ProcessConfig(config, processed.obs)
RunAnalysis(processed.obs, processed.config, path = path.out,
            id = "Stats_NWQL_1989-2012", sdate = "1989-01-01", edate = "2012-12-31",
            site.locations = site.locations, graphics.type = graphics.type)
```

Run trend analysis for field parameters (pH, Specific Conductance, and Temperature) from 1989 through 2012:

```
file <- file.path(path.in, "Config_Field.tsv")
config <- do.call(read.table, c(list(file), read.args))
processed.config <- ProcessConfig(config, processed.obs)
RunAnalysis(processed.obs, processed.config, path = path.out,
            id = "Stats_Field_1989-2012", sdate = "1989-01-01", edate = "2012-12-31",
            site.locations = site.locations, graphics.type = graphics.type)
```

Run trend analysis for radiation-related (RAD) parameters from 1981 through 2012:

```
file <- file.path(path.in, "Config_RAD.tsv")
config <- do.call(read.table, c(list(file), read.args))
processed.config <- ProcessConfig(config, processed.obs)
RunAnalysis(processed.obs, processed.config, path = path.out,
            id = "Stats_RAD_1981-2012", sdate = "1981-01-01", edate = "2012-12-31",
            site.locations = site.locations, graphics.type = graphics.type)
```

Run trend analysis for volatile organic compound (VOC) parameters from 1987 through 2012:

```
file <- file.path(path.in, "Config_VOC.tsv")
config <- do.call(read.table, c(list(file), read.args))
processed.config <- ProcessConfig(config, processed.obs)
RunAnalysis(processed.obs, processed.config, path = path.out,
```

```
            id = "Stats_VOC_1987-2012", sdate = "1987-01-01", edate = "2012-12-31",
            site.locations = site.locations, graphics.type = graphics.type)
```

Run trend analysis for Carbon Tetrachloride (P32102), a VOC parameter, at the RWMC Production well from 2005 through 2012:

```
rec <- with(processed.config, Parameter_id == "P32102" & Site_id == "433002113021701")
RunAnalysis(processed.obs, processed.config[rec, ], path = path.out,
            id = "Stats_RWMC_CCl4_2005-2012",sdate = "2005-01-01", edate = "2012-12-31",
            graphics.type = graphics.type)
```

Run trend analysis for Chloride (CL), a NWQL parameter, at the USGS 114, PW 8, USGS 87, USGS 35, and USGS 39 wells from 1989 through 2012:

```
file <- file.path(path.in, "Config_NWQL.tsv")
config <- do.call(read.table, c(list(file), read.args))
processed.config <- ProcessConfig(config, processed.obs)
site.ids <- c("433318112555001", "433456112572001", "433013113024201",
              "433339112565801", "433343112570001")
rec <- with(processed.config, Parameter_id == "CL" & Site_id %in% site.ids)
RunAnalysis(processed.obs, processed.config[rec, ], path = path.out,
            id = "Test",sdate = "1989-01-01", edate = "2012-12-31",
            graphics.type = graphics.type)
```

Account for residual water-level variability in the trend model.

```
RunAnalysis(processed.obs, processed.config[rec, ], path = path.out,
            id = "Test_resWLs",sdate = "1989-01-01", edate = "2012-12-31",
            graphics.type = graphics.type, explanatory.var = processed.wl,
            is.residual = TRUE)
```

Account for seasonal variability in the trend model.

```
RunAnalysis(processed.obs, processed.config[rec, ], path = path.out,
            id = "Test_Seas",sdate = "1989-01-01", edate = "2012-12-31",
            graphics.type = graphics.type, is.seasonality = TRUE)
```

Account for residual water-level variability and seasonal variability in the trend model.

```
RunAnalysis(processed.obs, processed.config[rec, ], path = path.out,
            id = "Test_resWLs_Seas",sdate = "1989-01-01", edate = "2012-12-31",
            graphics.type = graphics.type, is.seasonality = TRUE,
            explanatory.var = processed.wl, is.residual = TRUE)
```

To reprocess the 2014 water quality data, evaluate **R** code extracted from this vignette using the following command:

```
source(system.file("doc", "Trends-process.R", package = "Trends"), echo = TRUE)
list.files(path.out, full.names = TRUE, recursive = TRUE)  # path names of output files
```

Total processing time for this vignette was 2 minutes. Version information about **R** and loaded packages is as follows:

- R version 3.1.2 (2014-10-31), `x86_64-w64-mingw32`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Trends 0.2-1, rgdal 0.9-1, sp 1.0-15

- Loaded via a namespace (and not attached): evaluate 0.5.5, formatR 1.0, grid 3.1.2, highr 0.4, knitr 1.7, lattice 0.20-29, splines 3.1.2, stringr 0.6.2, survival 2.37-7, tools 3.1.2