Package ‘Covid19Wastewater’

August 24, 2023

Title Prepare, Analyze, and Visualize Covid-19 Wastewater Data

Version 1.0.1

Date 2023-08-7

Description Intended to make the process of
analyzing epidemiological wastewater data easier and more insightful.
Includes tools for preparing, analyzing, and visualizing data. It
additionally includes Wisconsin’s Covid19 data.


License MIT + file LICENSE

Depends R (>= 4.3)

Imports data.table, dplyr, forecast, ggplot2, grDevices, gridExtra,
methods, partykit, patchwork, plotly, plyr, randomForest,
RcppRoll, reshape2, rlang, rsample, scales, signal, stats,
tidy, tidyselect, zoo

Suggests knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation yes

Author Marlin Lee [aut, cre] (<https://orcid.org/0009-0000-0448-9585>),
Kyllan Wunder [aut] (<https://orcid.org/0009-0008-8587-0967>),
Abe Megahed [aut] (<https://orcid.org/0009-0000-0890-2656>)

Maintainer Marlin Lee <mrlee6@wisc.edu>

Repository CRAN

Date/Publication 2023-08-24 20:10:02 UTC
R topics documented:

Aux_info_data ............................................ 3
bagging .............................................. 3
buildCaseAnalysisDF .................................. 4
buildRegressionEstimateTable ......................... 5
buildWasteAnalysisDF .................................. 6
Case_data .................................................. 6
classifyCaseRegression ................................. 7
classifyQuantileFlagRegression ......................... 8
classifyRegressionAnalysis ............................. 9
computeJumps ............................................ 10
computeRankQuantiles .................................. 10
countFlags ............................................... 11
Covariants_data ......................................... 12
Covid19Wastewater ...................................... 12
createCaseFlag ........................................... 13
Data_Description ......................................... 14
date_distance_calc ...................................... 14
date_distance_clamp ...................................... 15
date_distance_remove .................................... 16
DF_date_vector ........................................... 16
Example_data ............................................. 17
expand_formula .......................................... 17
expSmoothMod ............................................ 18
factorVecByVec .......................................... 18
flagOutliers .............................................. 19
Flag_From_Trend .......................................... 20
gen_INCMSE ............................................. 21
gen_OOB_pred ............................................ 21
heatmapcorfunc .......................................... 22
HFGCase_data ........................................... 23
HFGWaste_data ........................................... 23
InterceptorCase_data .................................... 23
loessSmoothMod .......................................... 24
makeQuantileColumns .................................... 24
OffsetDFMaker ............................................ 25
OffsetDF_Plot ............................................ 25
OffsetHeatmap .......................................... 26
OOB_MSE_num_trees ...................................... 28
Pop_data ................................................. 29
predict.random_linear_forest-method ................. 29
random_linear_forest .................................... 30
random_linear_forest-class ............................. 31
rankJumps ............................................... 31
removeOutliers .......................................... 32
sgolaySmoothMod ........................................ 33
show.random_linear_forest-method ...................... 33
**Aux_info_data**

Auxiliary data

**Description**

auxiliary information that has been split from the WasteWater_data. Can be re-merged using `merge(WasteWater_data, Aux_info_data, by="sample_id")`

**Usage**

data("Aux_info_data")

**Format**

Rows: 999 Columns: 17

---

**bagging**

*Bootstrap aggregating of dataset gen a list of dataframes using row resampling and column downsizing*

---

**Description**

Bootstrap aggregating of dataset gen a list of dataframes using row resampling and column downsizing

**Usage**

`bagging(data, num_bags, num_features = NULL, include_first = 0)`

**Arguments**

- `data` : data.frame info
- `num_bags` : number of resamplings done
- `num_features` : number of columns in each output data.frame
- `include_first` : auto include the first n rows

**Value**

return list of 2 list of data.frame given different bagged data
Examples

```
data(Example_data, package = "Covid19Wastewater")
head(bagging(Example_data, 10, 5, 2))
```

Description

Prep case data into right format

Usage

```
buildCaseAnalysisDF(
    df, 
    site_column = site, 
    date_column = date, 
    case_column = conf_case, 
    pop_column = pop 
)
```

Arguments

- **df**: case dataframe have columns: Date, pop, FirstConfirmed
- **site_column**: name of site column
- **date_column**: name of date column
- **case_column**: name of case column
- **pop_column**: name of pop column

Value

DF with a 7 day rolling sum and a population weighted case column

Examples

```
data(Case_data, package = "Covid19Wastewater")
data(Pop_data, package = "Covid19Wastewater")
buildCaseAnalysisDF(dplyr::left_join(head(Case_data), Pop_data))
```
**buildRegressionEstimateTable**

*Run DHS analysis at a top level*

**Description**

buildRegressionEstimateTable is used to create a data frame that has the predicted categorization as laid out by the DHS. For each RunOn var supplied it uses the var to create a 5 day lm fit and uses the percent change to bin the results into 5 categories, "major decrease", "moderate decrease", "fluctuating", "moderate increase", and "major increase". If the model P-value if over .3 the category is replaced with "no change".

**Usage**

```r
buildRegressionEstimateTable(
  DataMod,
  RunOn = "sars_cov2_adj_load_log10",
  SplitOn = "site",
  DaysRegressed = 5,
  verbose = FALSE,
  PSigTest = TRUE
)
```

**Arguments**

- **DataMod**: The DF containing the col RunOn + date
- **RunOn**: The col names of the values we wish to run
- **SplitOn**: A category to separate to create independent TS data
- **DaysRegressed**: number of days used in each regression
- **verbose**: Bool on whether it should print out what group it is on
- **PSigTest**: When categorizing if it should reject high pVals

**Value**

A DF with the associated Date and DHS analysis

**Examples**

```r
library(dplyr)
data(Example_data, package = "Covid19Wastewater")
Example_data <- Example_data[Example_data$site == 'Janesville',]
Example_log_data <- mutate(Example_data, log_geo_mean = log10(geo_mean + 1))
head(buildRegressionEstimateTable(Example_log_data, SplitOn = "site",
                                  RunOn = "log_geo_mean"))
```
**buildWasteAnalysisDF**  
*Convert wastewater_data data to workset4 shape*

**Description**
This takes the wastewater_data dataframe and rename variables, calculates sars_cov2_adj_load_log10 column, and filters rows where average_flow_rate is NA.

**Usage**
```
buildWasteAnalysisDF(df)
```

**Arguments**
- `df`  
data frame object from data/wastewater_data.rda

**Value**

data frame

**Examples**
```
data(WasteWater_data, package = "Covid19Wastewater")
data(Pop_data, package = "Covid19Wastewater")
buildWasteAnalysisDF(dplyr::left_join(head(WasteWater_data), Pop_data))
```

---

**Case_data**  
*Case data*

**Description**
Data from the DHS of positive covid tests in Wisconsin cites.

**Usage**
```
data("Case_data")
```

**Format**

Rows: 33103  
Columns: 4
classifyCaseRegression

Create Case Flags based on regression slope

Description

Create Case Flags based on regression slope

Usage

```r
classifyCaseRegression(
  DF,
  slopeThreshold = 5,
  minSize = 200,
  per_changed_column = modeled_percentchange,
  model_sig_column = lmreg_sig
)
```

Arguments

- **DF**: dataframe that contains results of `buildRegressionEstimateTable`
- **slopeThreshold**: number threshold for `case_flag` flagging
- **minSize**: case threshold for `case_flag_plus_comm.threshold` flagging
- **per_changed_column**: column name of estimated percent change of signal
- **model.sig_column**: column name of significance of linear model that generated the percent change

Value

DF with an three extra column Category containing the case flags: `case_flag`: when the 7 day slope is above `slopeThreshold` `case_flag_plus_comm.threshold`: when `case_flag` and more then 200 cases `slope_switch_flag`: the first case flags in consecutive case flags

Examples

```r
example_DF <- data.frame(site = "madison", lmreg_slope = 5, value = 300)
classifyCaseRegression(example_DF)
```
classifyQuantileFlagRegression

Classify FlagRegression with rolling Quantile info

Description
Create wastewater flags based on the CDC classification defined in classifyRegressionAnalysis and
the quantile rank of the date.

Usage
classifyQuantileFlagRegression(
    DF,
    Pval = 0.3,
    model_sig_column = lmreg_sig,
    WW_column = pastKavg.wwlog10
)

Arguments
DF           dataframe that contains results of buildRegressionEstimateTable and makeQuan-
tileColumns
Pval         threshold needed for flag_nitile_Pval to flag
model_sig_column column name of significance of linear model that generated the percent change
WW_column    column containing ww data

Value
DF with three extra columns cdc_flag: when the CDC method labels as 'major increase' flag_nitile:
when the cdc flag and its in the top quantile flag_nitile_Pval: when the flag ntile and the regression
slope is less than Pval

Examples
data(Example_data, package = "Covid19Wastewater")
Example_data$modeled_percentchange = 0
Example_data$lmreg_sig = .01
Example_data$pastKavg.wwlog10 = 5
Example_data$ntile = 8
classifyQuantileFlagRegression(Example_data)
classifyRegressionAnalysis

description

adds the DHS classification scheme to data created by runRegressionAnalysis

usage

classifyRegressionAnalysis(
  DF,
  PSigTest = TRUE,
  per_changed_column = modeled_percentchange,
  model_sig_column = lmreg_sig
)

arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>dataframe that contains results of buildRegressionEstimateTable</td>
</tr>
<tr>
<td>PSigTest</td>
<td>Controls if we filter values with P-values&gt;.3</td>
</tr>
<tr>
<td>per_changed_column</td>
<td>column name of estimated percent change of signal</td>
</tr>
<tr>
<td>model_sig_column</td>
<td>column name of significance of linear model that generated the percent change</td>
</tr>
</tbody>
</table>

value

DF with an extra column Catagory containing the results of the DHS binning

examples

data(Example_data, package = "Covid19Wastewater")
Example_data$modeled_percentchange = 0
Example_data$lmreg_sig = .01
classifyRegressionAnalysis(Example_data)
computeJumps

compute first difference Jumps for N1 and N2

Description
compute first difference Jumps for N1 and N2

Usage
computeJumps(df, N1_column = N1, N2_column = N2, site_column = site)

Arguments
df       DataFrame. needs Column n1_sars_cov2_conc, n2_sars_cov2_conc, site
N1_column N1 metric used in finding difference
N2_column N2 metric used in finding difference
site_column Grouping that makes each group a timeseries

Value
dataframe with 4 columns appended: delta(n1), delta(n2) from left and right

Examples
data(Example_data, package = "Covid19Wastewater")
Example_data$site = "Madison"
computeJumps(Example_data)

computeRankQuantiles

Description
Convert jumps from last step into a ordering quintile

Usage
computeRankQuantiles(df)

Arguments
df       dataframe. needs Column n1.jumpFromLeft, n1.jumpFromRight, n2.jumpFromLeft,
n2.jumpFromRight, site
First 4 gen from computeJumps
countFlags

Value

dataframe with 4 columns appended: ranks of each of the 4 jumps;

Examples

data(Example_data, package = "Covid19Wastewater")
Example_data$site = "Madison"
df_data <- computeJumps(Example_data)
ranked_data <- rankJumps(df_data)
computeRankQuantiles(ranked_data)

countFlags Create counts of flag data

Description

Takes a data frame with grouping columns and numeric flag DF where 1 means a flag

Usage

countFlags(DF, group = c("Site", "window", "quant"))

Arguments

DF data frame with grouping columns and numeric flag columns

group vector specifying what columns should be used for grouping

Value

DF with the number of flags of each type within groups

Examples

data(Example_data, package = "Covid19Wastewater")
countFlags(Example_data, group = c("site"))
**Covariants_data**  
*Covariants data*

**Description**

Dataframe containing info about the proportion of each COVID-19 variant every 2 weeks. This data is from [https://github.com/hodcroftlab/covariants/tree/master](https://github.com/hodcroftlab/covariants/tree/master)

**Usage**

```r
data("Covariants_data")
```

**Format**

- Rows: 69
- Columns: 33

---

**Covid19Wastewater**  
*Covid19Wastewater: A package for running Covid19 wastewater concentration analysis*

**Description**

This is an R package of utilities to perform wastewater data analysis for pathogenic surveillance and monitoring.

**Details**

This project is a collaboration between the University of Wisconsin’s [Data Science Institute (DSI)](https://datascience.wisc.edu), the [Wisconsin Department of Health Services (DHS)](https://www.dhs.wisconsin.gov/covid-19/wastewater.htm), and the [Wisconsin State Lab of Hygiene (SLH)](https://www.slh.wisc.edu/environmental/covid-19-wastewater/).

If you need help getting started view our [getting started guide](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/getting-started/Starting_Guide.md)

**Built in data**

View our data vignettes to learn more about our built in data. To view what each column name means view our [Data_Description](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/data/data_columns_discription.md)

This package includes traditional (case-based) and wastewater-based data about Wisconsin communities.

* High-Frequency Data: This dataset contains high-frequency testing data collected over a 7-day testing period. It includes 9 samples reported per day and is accompanied by corresponding case data. The dataset files are: [HFG_data.RData](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/HFG_data_case.pdf), [HFGCase_data.RData](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/HFGdata_case.pdf)

* Extra Information: The package also provides additional information in the form of dataframes. The extra info datasets are as follows: [Pop_data.RData](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/popl_data.pdf), [Covariants_data.RData](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/variant_data.pdf)

**Data Prep**

Learn how to prep our data here for analysis

The following are the data preparation tools used in this package:


* Smoothing: This vignette is an illustration of the smoothing techniques that are available in this package: [smoothing.Rmd](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/smoothing.pdf)

* Calculated Info: This vignette computed additional auxiliary information (geometric means, population normalizing, case averaging): [calculated_info.Rmd](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/calculated_info.pdf)

**Analysis tools**

Learn how to analyze using our methods or create your own.

The following are the analysis tools used in this package:

* DHS Methods: This code comes from our collaborators at the DHS and is an example of of their flagging method that is used on their [dashboard](https://www.dhs.wisconsin.gov/covid-19/wastewater.htm)

* Time Series Offset: This vignette demonstrates how to use the offset analysis tools to find the temporal offset in days between wastewater and case measurements: [time_series_offset.Rmd](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/time_series_offset.pdf)

* Linear Forest: This package contains a Linear Forest modeling tool for finding an optimal breakdown of linear components. This is most useful for looking for cofactors to the N1 vs case data: [linear_forest.Rmd](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/vignettes/linear_forest.pdf)

View more in our github repo: [https://github.com/UW-Madison-DSI/Covid19Wastewater](https://github.com/UW-Madison-DSI/Covid19Wastewater)

---

**createCaseFlag**

Create Case flags

---

**Description**

Create case flags created by the DHS to warn of increasing cases

**Usage**

createCaseFlag(DF)
**Arguments**

**DF**  
Input data frame

**Value**

data frame with columns:  
site: Location of flag  
date: date of flag  
case_flag: when the 7 day slope is above 5  
case_flag_plus_comm.threshold: when case flag and more than 200 cases  
slope_switch_flag: the first case flags in consecutive case flags

**Examples**

data(Example_data, package = "Covid19Wastewater")  
data(Pop_data, package = "Covid19Wastewater")  
Example_data <- Example_data[Example_data$site == 'Janesville',]  
createCaseFlag(dplyr::left_join(Example_data, Pop_data))

---

**Description**

Data Description This package contains a lot of data with many column names, here is a list of them all: [https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/data/data_columns_discription.md](https://github.com/UW-Madison-DSI/Covid19Wastewater/blob/main/docs/data/data_columns_discription.md)

---

**Usage**

date_distance_calc(DF, base_date_vec, vecNames)

**Arguments**

**DF**  
DF to extract vector from

**base_date_vec**  
date flag vector to pull each distance from

**vecNames**  
a vector of column vec names each with binary flagging info
**Value**

DF containing the distance of each term to the base vector

**Examples**

```r
data("Example_data", package = "Covid19Wastewater")
Example_data$Late_date <- sample(Example_data$date)
Example_data$Late_date[sample(1:length(Example_data), length(Example_data) / 3)] <- NA
head(date_distance_calc(Example_data, "date", "Late_date"))
```

**Description**

remove distances above threshold

**Usage**

`date_distance_clamp(DF, vecNames, thresh)`

**Arguments**

- **DF**: source dataframe
- **vecNames**: column names in DF to be modified
- **thresh**: max distance not reduced to thresh from data

**Value**

DF with distances above threshold clamped to threshold

**Examples**

```r
data("Example_data", package = "Covid19Wastewater")
Example_data$Late_date <- sample(Example_data$date)
Example_data$Late_date[sample(1:length(Example_data), length(Example_data) / 3)] <- NA
df <- date_distance_calc(Example_data, "date", "Late_date")
```
**date_distance_remove**

*remove distances above threshold*

**Description**

remove distances above threshold

**Usage**

date_distance_remove(DF, vecNames, thresh)

**Arguments**

- **DF**: source dataframe
- **vecNames**: column names in DF to be modified
- **thresh**: max distance not removed from data

**Value**

DF with distances above threshold removed

**Examples**

data("Example_data", package = "Covid19Wastewater")
Example_data$Late_date <- sample(Example_data$date)
Example_data$Late_date[sample(1:length(Example_data), length(Example_data) / 3)] <- NA
df <- date_distance_calc(Example_data, "date", "Late_date")
date_distance_remove(df, "Late_date", 21)

**DF_date_vector**

*DF_date_vector*

**Description**

DF_date_vector

**Usage**

DF_date_vector(DF, date_vec, flag_vecs)

**Arguments**

- **DF**: DF containing the other vectors as columns
- **date_vec**: column vec name with date data type
- **flag_vecs**: a vector of column vec names each with binary flagging info
Example_data

Value
an equivalent DF where the 1 in the flag info is replaced with the date

Examples
data("Example_data", package = "Covid19Wastewater")
Example_data$flag = 1
head(DF_date_vector(Example_data, "date", "flag"))

---

Description
Toy example of full data contains 3 sites and roughly 500 entries (waste and case merged together)

Usage
data("Example_data")

Format
Rows: 500 Columns: 69

---

expand_formula

Expand formula for increased info takes a formula with shape A ~ B | C and convert . to its real representation

Description
Expand formula for increased info takes a formula with shape A ~ B | C and convert . to its real representation

Usage
expand_formula(X, data)

Arguments
X formula object like A ~ B | C
data data object to extract . info from

Value
formula object
expSmoothMod

Add a column of the smoothed values using exponential smoothing

**Description**

expSmoothMod Add a column of the smoothed values using exponential smoothing

**Usage**

```r
expSmoothMod(DF, InVar, OutVar, alpha = "guess", beta = "guess", Filter = NULL)
```

**Arguments**

- **DF**
  The DF we are to add an exponential smoothing column to
- **InVar**
  The column to be smoothed
- **OutVar**
  The name of the new column
- **alpha**
  The alpha fed into forecast exponential smoothing. if it equals "guess" then it if found using parameterGuess
- **beta**
  The beta fed into forecast exponential smoothing. if it equals "guess" then it if found using parameterGuess
- **Filter**
  Prefilter using the value of a Filter col

**Value**

A DF with an extra col with a exp smoothed version of InVar

**Examples**

```r
data("Example_data", package = "Covid19Wastewater")
Example_data <- Example_data[Example_data$site == "Green Bay",]
expSmoothMod(Example_data, "N1", "expN1")
```

factorVecByVec

Get ordering for plotting based on factoring vector

**Description**

Get ordering for plotting based on factoring vector

**Usage**

```r
factorVecByVec(FactorDF, OrderDF, FacVar, NumVar)
```
flagOutliers

**Arguments**

- **FactorDF**: DF containing FacVar
- **OrderDF**: DF containing NumVar
- **FacVar**: Column given ordering
- **NumVar**: Column used to find ordering

**Value**

FactorDF with FacVar being a factor ordered by NumVar

**Examples**

```r
data(Example_data, package = "Covid19Wastewater")
head(factorVecByVec(Example_data, Example_data, "site", "N1"))
```

---

flagOutliers

*Create column with Boolean based on a threshold*

**Description**

Create column with Boolean based on a threshold

**Usage**

```r
flagOutliers(DF, threshold, col, FlaggedOutlier = FlaggedOutlier)
```

**Arguments**

- **DF**: Dataframe containing Column column ranked_quantile_data
- **threshold**: a numeric used to flag if its an outlier
- **col**: column being flagged based on threshold
- **FlaggedOutlier**: name of flag column

**Value**

DF Dataframe with the extra column of if its flagged an outlier

**Examples**

```r
data(Example_data, package = "Covid19Wastewater")
Example_data$site = "Madison"
df_data <- computeJumps(Example_data)
ranked_data <- rankJumps(df_data)
ranked_quantile_data <- computeRankQuantiles(ranked_data)
flagOutliers(ranked_quantile_data, 9, MeasureRank, FlaggedOutlier)
```
Flag values as outliers based on error from estimated trend. This function can be done within group if the data fed into it was grouped.

**Description**

Flag values as outliers based on error from estimated trend. This function can be done within group if the data fed into it was grouped.

**Usage**

```
Flag_From_Trend(DF, base_data, trend_data, sd_degree = 2.5)
```

**Arguments**

- **DF**: Dataframe containing selected columns
- **base_data**: Name of column containing Raw data
- **trend_data**: Name of generated trend data
- **sd_degree**: Threshold to flag deviant values as an outlier

**Value**

DF with a new column 'flagged_outlier' that contains if the column is an outlier

**Examples**

```r
library(dplyr)
data("Example_data", package = "Covid19Wastewater")
smoothing_df <- Example_data%>%
    select(site, date, N1, N2)%>%
    filter(N1 != 0, N2 != 0)%>%
    mutate(N1 = log(N1), N2 = log(N2), N12_avg = (N1 + N2) / 2)
df_data <- loessSmoothMod(smoothing_df, "N12_avg", "N12_avg_loess", Filter = NULL)
head(df_data%>%
    group_by(site)%>%
    Flag_From_Trend( N12_avg, N12_avg_loess)%>%
    select(site, date, N12_avg, flagged_outlier))
```
**gen_INCMSE**

*get increased mean square error for each column*

**Description**

get increased mean square error for each column

**Usage**

```r
gen_INCMSE(tree_model)
```

**Arguments**

- `tree_model` random_linear_forest object you want the MSE of

**Value**

data.frame containing each column and its MSE increase

**Examples**

```r
data(Example_data, package = "Covid19Wastewater")
model <- random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
gen_INCMSE(model)
```

---

**gen_OOB_pred**

*get OOB predictions of the training dataset returns the predictions of each row of the input data using only trees not trained on the row*

**Description**

get OOB predictions of the training dataset returns the predictions of each row of the input data using only trees not trained on the row

**Usage**

```r
gen_OOB_pred(tree_model, incMSE = NA, resid = FALSE)
```

**Arguments**

- `tree_model` random_linear_forest object you want the OOB predictions of
- `incMSE` if its a numeric scramble the given column
- `resid` if True return the residuals of the model instead
Value

data.frame or vector of oob predictions

Examples

data(Example_data, package = "Covid19Wastewater")
model <- random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
gen_OOB_pred(model, resid = TRUE)

heatmapcorfunc

Outputs a heatmap where the color is the r squared of wastewater data and center day + x many future days and y many past days Helps inform Offset Analysis

Description

Outputs a heatmap where the color is the r squared of wastewater data and center day + x many future days and y many past days Helps inform Offset Analysis

Usage

heatmapcorfunc(cordata, length = 14, case_column = conf_case)

Arguments

cordata     DF with geo_mean and conf_case columns
length      the length of the time window for the results / 2
case_column name of case column

Value

ggplot plot object (heatmap)

Examples

data(Example_data, package = "Covid19Wastewater")
heatmapcorfunc(Example_data)
### HFGCase_data

**Description**
Data from the DHS of positive covid tests for HFG data

**Usage**
data("HFGCase_data")

**Format**
Rows: 788 Columns: 6

### HFGWaste_data

**Description**
Data from the DHS of HFG covid wastewater concentration

**Usage**
data("HFGWaste_data")

**Format**
Rows: 3078 Columns: 18

### InterceptorCase_data

**Description**
Data from the DHS of positive covid tests for Madison interceptors

**Usage**
data("InterceptorCase_data")

**Format**
Rows: 3288 Columns: 5
**loessSmoothMod**

*loessSmoothMod Add a column of the smoothed values using Loess*

---

**Description**

loessSmoothMod Add a column of the smoothed values using Loess

**Usage**

```r
loessSmoothMod(
  DF,
  InVar = "N1",
  OutVar = "Loess",
  Span = "guess",
  Filter = NULL
)
```

**Arguments**

- **DF**: DF we are adding the loess smooth col to
- **InVar**: The column to be smoothed
- **OutVar**: The name of the new column
- **Span**: The span fed into loess smoothing. if it equals "guess" then it if found using parameterGuess
- **Filter**: Prefilter using the value of a Filter col

**Value**

A DF with an extra col with a loesss smoothed version of InVar

**Examples**

```r
data(Example_data, package = "Covid19Wastewater")
head(loessSmoothMod(Example_data))
```

---

**makeQuantileColumns**

*Add many combo of rolling quantile columns to dataframe have info for each quant window combo*

---

**Description**

Add many combo of rolling quantile columns to dataframe have info for each quant window combo
Usage

```
makeQuantileColumns(DF, quants, windows, column = "N1")
```

Arguments

DF : Dataframe containing columns: site: what site the data is from date: date variable communicating the day the measurement is from *column: DF needs to contain a column with the same name as the string in the variable column

quants : vector containing the different quantiles

windows : vector containing the different windows

column : string name of column in DF

Value

DF with added columns window: what window group the row is in quant: what quantile group the row is in ntile: a rolling quantile of the data pastKavg.wwlog10: a mean of the last K days

Examples

```
data("Example_data", package = "Covid19Wastewater")
Example_data$site = "madison"
makeQuantileColumns(Example_data, .5, 6, column = "geo_mean")
makeQuantileColumns(Example_data, c(.5, .75), c(2,5), column = "geo_mean")
```

---

OffsetDFMaker

**Returns a dataframe with the multiple ways to analyze how offset the Wastewater is from cases data**

Description

Returns a dataframe with the multiple ways to analyze how offset the Wastewater is from cases data

Usage

```
OffsetDFMaker(
  length, startdate, enddate, casesdf, wastedf, N1_column = N1, N2_column = N2, site_column = site, date_column = date, case_column = conf_case
)
```
### Arguments

- **length**
  - The length of the time window for the results / 2

- **startdate**
  - First day for both data sets

- **enddate**
  - Last day for both data sets

- **casesdf**
  - DF with case data columns needed (data, conf_case)

- **wastedf**
  - DF with Wastewater data columns needed (date, N1, N2)

- **N1_column**
  - name of N1 column

- **N2_column**
  - name of N2 column

- **site_column**
  - name of site column

- **date_column**
  - name of date column

- **case_column**
  - name of case column

### Value

DF with the columns: number of days wastewater is offset, geo mean of (n1, n2) / confirmed cases, previous with rolling case average, MSE rolling average, Pearson correlation, Kendall correlation, Spearman correlation, R squared.

### Examples

```r
data(WasteWater_data, package = "Covid19Wastewater")
data("Case_data", package = "Covid19Wastewater")
# Will output a df from -10 to +10 days using all of the data from 2020-08-01 to 2023-01-01
OffsetDFMaker(10, as.Date("2020-08-01"), as.Date("2023-01-01"), Case_data, WasteWater_data)
```

### Description

Given output from OffsetDFMaker returns a 2x3 grid of all the plots with highlighted values

### Usage

```r
OffsetDF_Plot(data, title)
```

### Arguments

- **data**
  - Output from OffsetDFMaker

- **title**
  - Title you want (appears at bottom)

### Value

`ggplot` plot object
OffsetHeatmap

Examples

data(WasteWater_data, package = "Covid19Wastewater")
data("Case_data", package = "Covid19Wastewater")
OffsetDFMaker_Output <- OffsetDFMaker(10, as.Date("2020-08-01"),
                                      as.Date("2023-01-01"),
                                      Case_data, WasteWater_data)
OffsetDF_Plot(OffsetDFMaker_Output,"All Wisconsin data over all time")

OffsetHeatmap (method, timePeriods, waste_df, case_df, pop_df, covarstarts, covarends, covarnames, list, week, lod = FALSE, N1_column = N1, N2_column = N2, site_column = site, date_column = date, case_column = conf_case, pop_column = pop)

Description

Outputs a heatmap of the offset for variant / time windows and population size / region

Usage

OffsetHeatmap(  
  method,  
  timePeriods,  
  waste_df,  
  case_df,  
  pop_df,  
  covarstarts,  
  covarends,  
  covarnames,  
  list,  
  week,  
  lod = FALSE,  
  N1_column = N1,  
  N2_column = N2,  
  site_column = site,  
  date_column = date,  
  case_column = conf_case,  
  pop_column = pop  
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Which analysis definds the offset (r squared, pearson, r squared offset, pearson offset, kendall offset, spearman offset)</td>
</tr>
<tr>
<td>timePeriods</td>
<td>Size of time windows in months (if 0 uses variants)</td>
</tr>
<tr>
<td>waste_df</td>
<td>DF of waste data must include: date, N1, N2</td>
</tr>
<tr>
<td>case_df</td>
<td>DF of case data must include: date, conf_case</td>
</tr>
<tr>
<td>pop_df</td>
<td>dataframe where region and population info is stored</td>
</tr>
</tbody>
</table>
covarstarts start of each split period
covarends end of each split period
covarnames name of each split group
list y axis bins (population size or regions)
week if true applies 7-day smoothing to case data
lod if true removes all values below LOD (default false)
N1_column name of N1 column
N2_column name of N2 column
site_column name of site column
date_column name of date column
case_column name of case column
pop_column name of pop column

Value

ggplot plot object

Examples

data(WasteWater_data, package = "Covid19Wastewater")
data("Case_data", package = "Covid19Wastewater")
data(Pop_data, package = "Covid19Wastewater")
covarstarts <- c(as.Date("2020-08-17"),
as.Date("2021-03-29"))
covarends <- c(as.Date("2021-01-18"),
as.Date("2021-05-24"))
covarnames <- c("Robin1",
"Alpha.V1")
OffsetHeatmap("kendall_offset", 0, WasteWater_data, Case_data, Pop_data,
covarstarts, covarends, covarnames, "pop", TRUE, TRUE)

---

OOB_MSE_num_trees get OOB MSE vs number of forest in trees

Description

get OOB MSE vs number of forest in trees

Usage

OOB_MSE_num_trees(tree_model)

Arguments

tree_model random_linear_forest model to calculate the OOB mean squared error
Value
get dataframe of number of trees and OOB MSE

Examples

```r
data(Example_data, package = "Covid19Wastewater")
model <- random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
OOB_MSE_num_trees(model)
```

---

**Pop_data**

* Sewer shed population data

Description
dataframe containing population information about each sewer shed in Wisconsin

Usage
data("Pop_data")

Format
Rows: 89 Columns: 2

---

**predict,random_linear_forest-method**

* predict new data from random_linear_forest models

Description
predict new data from random_linear_forest models

Usage
```r
## S4 method for signature 'random_linear_forest'
predict(object, new_data, ...)
```

Arguments
- **object** random_linear_forest being used
- **new_data** data.frame.
- **...** extra parameters ignored

Value
vector of predictions for each row
Examples

```r
data(Example_data, package = "Covid19Wastewater")
model <- random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
predict(model, Example_data)
```

random_linear_forest  Fitting linear random forest

Description

This uses the linear tree model from party kit and bootstrapping to create linear random forests that work like a random forest but with the linear dynamics permuted with a linear method.

Usage

```r
random_linear_forest(
  data,
  num_tree,
  model_formula,
  num_features = NULL,
  na.action = na.roughfix,
  max_depth = 5,
  importance = FALSE,
  verbose = FALSE
)
```

Arguments

- **data**: a dataframe containing the variables in the model
- **num_tree**: numeric, the number of trees in the random forest.
- **model_formula**: an object of class "formula": a symbolic description of the model to be fitted.
- **num_features**: number of tree features in each tree. if left NULL rounded up square of the number of columns
- **na.action**: passed to lmtree to handle missing data
- **max_depth**: the max depth of each tree in the forest
- **importance**: controls if the importance should be calculated and stored
- **verbose**: If true it prints training progress

Value

random_linear_forest object trained using given data

Examples

```r
data(Example_data, package = "Covid19Wastewater")
random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
```
random_linear_forest-class

random_linear_forest model class using a random forest of linear forest models

Description
random_linear_forest model class using a random forest of linear forest models

Value
random_linear_forest object

Slots
  formula  formula used to create trees and linear models
  models   list of each tree model in model
  data     data.frame of the input data
  resid    numeric vector of residuals
  inbag_data  list of data.frame each model trained on
  oob_data  list of data.frame containing info model not trained on
  oob_resid numeric vector of oob residuals
  inc_mse   % increased mean squared error

Examples
  data("Example_data", package = "Covid19Wastewater")
  random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)

rankJumps

Description
Convert jumps from last step into a ordering

Usage
  rankJumps(df)

Arguments
  df          DataFrame, needs Column n1.jumpFromLeft, n1.jumpFromRight, n2.jumpFromLeft, n2.jumpFromRight, site
              First 4 gen from computeJumps
Value
dataframe with 4 columns appended: ranks of each of the 4 jumps;

Examples
data(Example_data, package = "Covid19Wastewater")
Example_data$site = "Madison"
df_data <- computeJumps(Example_data)
rankJumps(df_data)

removeOutliers  Add column with NA values where the data was flagged

Description
Add column with NA values where the data was flagged

Usage
removeOutliers(DF, Messure, Filtcol, outputColName)

Arguments
DF  DF containing the columns Measure and Filtcol
Messure  The original measurement we want to keep inliers for
Filtcol  the column containing the Boolean info needed to remove outliers
outputColName  the name for the clean column

Value
DF with new column without the flagged values

Examples
data(Example_data, package = "Covid19Wastewater")
Example_data$site = "Madison"
df_data <- computeJumps(Example_data)
ranked_data <- rankJumps(df_data)
ranked_quantile_data <- computeRankQuantiles(ranked_data)
classied_data <- flagOutliers(ranked_quantile_data, 9)
removeOutliers(classied_data, sars_cov2_adj_load_log10, FlaggedOutlier, sars_adj_log10_Filtered)
**Description**

sgolaySmoothMod Add a column of the smoothed values using sgolayfilt

**Usage**

sgolaySmoothMod(DF, InVar, OutVar, poly = 5, n = "guess", Filter = NULL)

**Arguments**

- **DF**: dataframe containing the columns specified below
- **InVar**: The column to be smoothed
- **OutVar**: The name of the new column
- **poly**: The degree of the polynomial fit
- **n**: The number of points per polynomial fed into sgolayfilt. if it equals "guess" then it is found using parameterGuess
- **Filter**: Prefilter using the value of a Filter col

**Value**

DF with an extra col with a sgolayfilt smoothed version of InVar

**Examples**

data(Example_data, package = "Covid19Wastewater")
Example_data <- Example_data[Example_data$site == "Green Bay",]
Covid19Wastewater::sgolaySmoothMod(WasteWater_data,"N1","sgolayN1")

**show,random_linear_forest-method**

*display form for random_linear_forest class*

**Description**

display form for random_linear_forest class

**Usage**

## S4 method for signature 'random_linear_forest'
show(object)
Arguments

object  object with class random_linear_forest

Value

Prints to output a summary of the model

Examples

```r
data(Example_data, package = "Covid19Wastewater")
random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
```

```r
model <- random_linear_forest(Example_data, 2, PMMoV ~ N1 + N2 | pcr_type)
summary(model)
```
**VariantPlot**

*Shows each variant in proportion to the others in 2 week time periods*

**Description**

Shows each variant in proportion to the others in 2 week time periods

**Usage**

```r
VariantPlot(covar)
```

**Arguments**

`covar` Covariant data frame

**Value**

`ggplotly` object

**Examples**

```r
data(Covariants_data, package = "Covid19Wastewater")
VariantPlot(Covariants_data)
```

---

**WasteWater_data**

*Wastewater data set*

**Description**

Data from the DHS of wastewater prevalence in Wisconsin cites

**Usage**

```r
data("WasteWater_data")
```

**Format**

Rows: 11,084 Columns: 62
Index

* datasets
  Aux_info_data, 3
  Case_data, 6
  Covariants_data, 12
  Example_data, 17
  HFGCase_data, 23
  HFGWaste_data, 23
  InterceptorCase_data, 23
  Pop_data, 29
  WasteWater_data, 35

Aux_info_data, 3

bagging, 3
buildCaseAnalysisDF, 4
buildRegressionEstimateTable, 5
buildWasteAnalysisDF, 6

Case_data, 6
classifyCaseRegression, 7
classifyQuantileFlagRegression, 8
classifyRegressionAnalysis, 9
computeJumps, 10
computeRankQuantiles, 10
countFlags, 11
Covariants_data, 12
Covid19Wastewater, 12
Covid19Wastewater-package
  (Covid19Wastewater), 12
createCaseFlag, 13

Data_Description, 14
date_distance_calc, 14
date_distance_clamp, 15
date_distance_remove, 16
DF_date_vector, 16

Example_data, 17
expand_formula, 17
expSmoothMod, 18

factorVecByVec, 18
Flag_From_Trend, 20
flagOutliers, 19
gen_INCMSE, 21
gen_OOB_pred, 21
heatmapcorfunc, 22
HFGCase_data, 23
HFGWaste_data, 23
InterceptorCase_data, 23
loessSmoothMod, 24
makeQuantileColumns, 24
OffsetDF_Plot, 26
OffsetDFMaker, 25
OffsetHeatmap, 27
OOB_MSE_num_trees, 28

Pop_data, 29
predict, random_linear_forest-method, 29
random_linear_forest, 30
random_linear_forest-class, 31
rankJumps, 31
removeOutliers, 32

sgolaySmoothMod, 33
show, random_linear_forest-method, 33
summary, random_linear_forest-method, 34

VariantPlot, 35
WasteWater_data, 35