Package 'kssa'

June 21, 2022

Title Known Sub-Sequence Algorithm

Version 0.0.1

Maintainer Iván Felipe Benavides <pipeben@gmail.com>

Description Implements the Known Sub-Sequence Algorithm <doi:10.1016/j.aaf.2021.12.013>, which helps to automatically identify and validate the best method for missing data imputation in a time series. Supports the comparison of multiple state-of-the-art algorithms.

License AGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.0

URL https://github.com/pipeben/kssa

BugReports https://github.com/pipeben/kssa/issues

Depends R (>= 4.0)

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports magrittr, ggplot2, rlang, methods, forecast, imputeTS, stats, zoo, Metrics, dplyr, missMethods

Date 2022-06-18

NeedsCompilation no

Author Iván Felipe Benavides [aut, cre, cph] (<https://orcid.org/0000-0002-1139-3909>), Steffen Moritz [aut] (<https://orcid.org/0000-0002-0085-1804>), Brayan-David Aroca-Gonzalez [aut] (<https://orcid.org/0000-0002-7365-5740>), Jhoana Romero [aut] (<https://orcid.org/0000-0002-1834-3461>), Marlon Santacruz [aut] (<https://orcid.org/0000-0003-2242-742X>), John-Josephraj Selvaraj [aut] (<https://orcid.org/0000-0002-9195-4883>)

Repository CRAN

Date/Publication 2022-06-21 19:40:02 UTC

R topics documented:

`get_imputations` *get_imputations function*

Description

Function to get imputations from methods compared by kssa

Usage

```
get_imputations(x_ts, methods = "all", seed = 1234)
```

Arguments

`x_ts`	A ts object with missing data to be imputed
`methods`	A string or string vector indicating the method or methods You can choose between the following: • "all" - get imputed values for all methods - Default • "auto.arima" - State space representation of an ARIMA model • "StructTS" - State space representation of a structural model • "seadec" - Seasonal decomposition with Kalman smoothing • "linear_i" - Linear interpolation • "spline_i" - Spline interpolation • "stine_i" - Stineman interpolation • "simple_ma" - Simple moving average • "linear_ma" - Linear moving average • "exponential_ma" - Exponential moving average • "locf" - Last observation carried forward • "stl" - Seasonal and trend decomposition with Loess For further details on these imputation methods please check packages `imputeTS` and `forecast`
`seed`	Numeric. Any number

Value

A list of imputed time series with the selected methods

Examples

```
# Example 1: Get imputed values for airgap_na_ts with the methods of

library("imputeTS")
library("kssa")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)

# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

my_imputations <- get_imputations(airgap_na_ts, methods = "all")

# my_imputations contains the imputed time series with all methods.
# Access it and choose the one from the best method for your purposes

my_imputations$seadec
plot.ts(my_imputations$seadec)



# Example 2: Get imputed values for airgap_na_ts using only a subset of algorithms

library("imputeTS")
library("kssa")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)

# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

my_imputations <- get_imputations(airgap_na_ts, methods = c("linear_i", "locf"))

# my_imputations contains the imputed time series with all applied
# methods (locf and linear interpolation).
# Access it and choose the one from the best method for your purposes

my_imputations$locf
plot.ts(my_imputations$locf)
```

---

kssa *kssa Algorithm*

Description

Run the Known Sub-Sequence Algorithm to compare the performance of imputation methods on a time series of interest

Usage

```
kssa(
  x_ts,
  start_methods,
  actual_methods,
  segments = 5,
  iterations = 10,
  percentmd = 0.2,
  seed = 1234
)
```

Arguments

x_ts	Time series object ts containing missing data (NA)
start_methods	String vector. The method or methods to start the algorithm. Same as for actual_methods
actual_methods	The imputation methods to be compared and validated. It can be a string vector containing the following You can choose between the following: • "all" - compare among all methods automatically - Default • "auto.arima" - State space representation of an ARIMA model • "StructTS" - State space representation of a structural model • "seadec" - Seasonal decomposition with Kalman smoothing • "linear_i" - Linear interpolation • "spline_i" - Spline interpolation • "stine_i" - Stineman interpolation • "simple_ma" - Simple moving average • "linear_ma" - Linear moving average • "exponential_ma" - Exponential moving average • "locf" - Last observation carried forward • "stl" - Seasonal and trend decomposition with Loess For further details on these imputation methods please check packages imputeTS and forecast
segments	Integer. Into how many segments the time series will be divided
iterations	Integer. How many iterations to run
percentmd	Numeric. Percentage of missing data. Must match with the true percentage of missing data in x_ts
seed	Numeric. Random seed to choose

Value

A list of results to be plotted with function kssa_plot for easy interpretation

References

Benavides, I. F., Santacruz, M., Romero-Leiton, J. P., Barreto, C., & Selvaraj, J. J. (2022). Assessing methods for multiple imputation of systematic missing data in marine fisheries time series with a new validation algorithm. Aquaculture and Fisheries. Full text publication.

Examples

```
# Example 1: Compare all imputation methods

library("kssa")
library("imputeTS")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)

# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

# Apply the kssa algorithm with 5 segments, 10 iterations, 20% of missing data,
# compare among all available methods in the package.
# Remember that percentmd must match with
# the real percentage of missing data in the input time series

results_kssa <- kssa(airgap_na_ts,
  start_methods = "all",
  actual_methods = "all",
  segments = 5,
  iterations = 10,
  percentmd = 0.2
)

# Print and check results
results_kssa

# For an easy interpretation of kssa results
# please use function kssa_plot




# Example 2: Compare only locf and linear imputation

library("kssa")
library("imputeTS")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
```

```
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)

# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

# Apply the kssa algorithm with 5 segments, 10 iterations, 20% of missing data,
# compare among all applied methods (locf and linear interpolation).
# Remember that percentmd must match with
# the real percentage of missing data in the input time series

results_kssa <- kssa(airgap_na_ts,
  start_methods = c("locf", "linear_i"),
  actual_methods = c("locf", "linear_i"),
  segments = 5,
  iterations = 10,
  percentmd = 0.2
)

# Print and check results
results_kssa

# For an easy interpretation of kssa results
# please use function kssa_plot
```

`kssa_plot` *kssa_plot function*

Description

Function to plot the results of kssa for easy interpretation

Usage

```
kssa_plot(results, type, metric)
```

Arguments

`results`	An object with results produced with function `kssa`
`type`	A character value with the type of plot to show. It can be "summary" or "complete".
`metric`	A character with the performance metric to be plotted. It can be "rmse", "mase," "cor", or "smape" • "rmse" - Root Mean Squared Error (default choice) • "mase" - Mean Absolute Scaled Error • "smape" - Symmetric Mean Absolute Percentage Error • "cor" - Pearson correlation coefficient For further details on these metrics please check package Metrics

Value

A plot of kssa results in which imputation methods are ordered from lower to higher (left to right) error.

Examples

```
# Example 1: Plot the results from comparing all imputation methods

library("kssa")
library("imputeTS")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)

# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

# Apply the kssa algorithm with 5 segments,
# 10 iterations, 20% of missing data, and
# compare among all available methods in the package.
# Remember that percentmd must match with
# the real percentage of missing data in the input time series

results_kssa <- kssa(airgap_na_ts,
  start_methods = "all",
  actual_methods = "all",
  segments = 5,
  iterations = 10,
  percentmd = 0.2
)

kssa_plot(results_kssa, type = "complete", metric = "rmse")

# Conclusion: Since kssa_plot is ordered from lower to
# higher error (left to right), method 'linear_i' is the best to
# impute missing data in airgap_na_ts. Notice that method 'locf' is the worst

# To obtain imputations with the best method, or any method of preference
# please use function get_imputations




# Example 2: Plot the results when only applying locf and linear interpolation

library("kssa")
library("imputeTS")

# Create 20% random missing data in tsAirgapComplete time series from imputeTS
airgap_na <- missMethods::delete_MCAR(as.data.frame(tsAirgapComplete), 0.2)
```

```
# Convert to time series object
airgap_na_ts <- ts(airgap_na, start = c(1959, 1), end = c(1997, 12), frequency = 12)

# Apply the kssa algorithm with 5 segments,
# 10 iterations, 20% of missing data, and compare among all
# applied methods (locf and linear interpolation).
# Remember that percentmd must match with
# the real percentage of missing data in the input time series

results_kssa <- kssa(airgap_na_ts,
  start_methods = c("linear_i", "locf"),
  actual_methods = c("linear_i", "locf"),
  segments = 5,
  iterations = 10,
  percentmd = 0.2
)

kssa_plot(results_kssa, type = "complete", metric = "rmse")
```

Index