

Package ‘WRI’

December 16, 2025

Type Package

Title Wasserstein Regression and Inference

Version 0.2.1

Author Xi Liu [aut, cre],
Chao Zhang [aut],
Matthew Coleman [aut],
Alexander Petersen [aut]

Description Implementation of the methodologies described in 1) Alexander Petersen, Xi Liu and Afshin A. Divani (2021) <[doi:10.1214/20-aos1971](https://doi.org/10.1214/20-aos1971)>, including global F tests, partial F tests, intrinsic Wasserstein-infinity bands and Wasserstein density bands, and 2) Chao Zhang, Piotr Kokoszka and Alexander Petersen (2022) <[doi:10.1111/jtsa.12590](https://doi.org/10.1111/jtsa.12590)>, including estimation, prediction, and inference of the Wasserstein autoregressive models.

License GPL-2

Encoding UTF-8

LazyData true

LazyDataCompression xz

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.6.0)

Imports fdapace (>= 0.2.0), Rfast (>= 1.9.8), CVXR (>= 0.99.7), expm
(>= 0.999-4), ggplot2 (>= 3.2.1), gridExtra (>= 2.3), stats,
Rcpp (>= 1.0.3), mvtnorm (>= 1.1-0), methods, rlang, polynom

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 2.1.0), modeest

VignetteBuilder knitr

NeedsCompilation yes

Maintainer Xi Liu <xiliu@ucsb.edu>

Repository CRAN

Date/Publication 2025-12-16 08:40:02 UTC

Contents

confidenceBands	2
den2Q_qd	4
globalFtest	4
partialFtest	6
predict.WARp	7
print.summary.WRI	8
quan2den_qd	8
simulate_quantile_curves	9
strokeCTdensity	10
summary.WRI	10
WARp	11
warSim	13
wass_R2	13
wass_regress	14
Index	16

confidenceBands	<i>Confidence Bands for Wasserstein Regression</i>
-----------------	--

Description

Confidence Bands for Wasserstein Regression

Usage

```
confidenceBands(  
  wass_regress_res,  
  Xpred_df,  
  level = 0.95,  
  delta = 0.01,  
  type = "density",  
  figure = TRUE,  
  fig_num = NULL  
)
```

Arguments

wass_regress_res	an object returned by the wass_regress function
Xpred_df	k-by-p matrix (or dataframe, or named vector) used for prediction. Note that Xpred_df should have the same column names with Xfit_df used in wass_regress_res
level	confidence level
delta	boundary control value in density band computation. Must be a value in the interval (0, 1/2) (default: 0.01)

type	'density', 'quantile' or 'both' 'density': density function bands will be returned (and plotted if figure = TRUE) 'quantile': quantile function and CDF bands will be returned (and plotted if figure = TRUE) 'both': three kinds of bands, density function, quantile function and CDF bands will be returned (and plotted if figure = TRUE)
figure	logical; if TRUE, return a sampled plot (default: TRUE)
fig_num	the fig_num-th row of Xpred_df will be used for visualization of confidence bands. If NULL, then fig_num is randomly chosen (default: NULL)

Details

This function computes intrinsic confidence bands for Xpred_df if type = 'quantile' and density bands if type = 'density', and visualizes the confidence and/or density bands when figure = TRUE.

Value

a list containing the following lists:

den_list:	- fpred: k-by-m matrix, predicted density function at Xpred_df. - f_ux: k-by-m matrix, upper bound of confidence bands of density functions. - f_lx: k-by-m matrix, lower bound of confidence bands of density functions. - Qpred: k-by-m matrix, f_lx[i,], f_ux[i,] and fpred[i,] evaluated on Qpred[i,] vector.
quan_list:	- Qpred: k-by-m matrix of predicted quantile functions. - Q_ux: k-by-m matrix of upper bound of quantile functions. - Q_lx: k-by-m matrix of lower bound of quantile functions. - t_vec: a length m vector - common grid for all quantile functions.
cdf_list:	- fpred: k-by-m matrix, predicted density function. - Fpred: k-by-m matrix, predicted cumulative distribution functions. - F_ux: k-by-m matrix, upper bound of cumulative distribution functions. - F_lx: k-by-m matrix, lower bound of cumulative distribution functions. - Fsup: k-by-m matrix, fpred[i,], F_lx[i,], F_ux[i,] and Fpred[i,] evaluated on Fsup[i,] vector.

Examples

```
alpha = 2
beta = 1
n = 50
x1 = runif(n)
t_vec = unique(c(seq(0, 0.05, 0.001), seq(0.05, 0.95, 0.05), seq(0.95, 1, 0.001)))
set.seed(1)
quan_obs = simulate_quantile_curves(x1, alpha, beta, t_vec)
Xfit_df = data.frame(x1 = x1)
res = wass_regress(rightside_formula = ~., Xfit_df = Xfit_df,
```

```

        Ytype = 'quantile', Ymat = quan_obs, Sup = t_vec)
confidence_Band = confidenceBands(res, Xpred_df = data.frame(x1 = c(-0.5,0.5)),
type = 'both', fig_num = 2)

data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve
xpred = predictor[2:3, ]

res = wass_regress(rightside_formula = ~., Xfit_df = predictor,
Ytype = 'density', Ymat = densityCurves, Sup = dSup)
confidence_Band = confidenceBands(res, Xpred_df = xpred, type = 'density', fig_num = 1)

```

den2Q_qd	<i>convert density function to quantile and quantile density function</i>
----------	---

Description

convert density function to quantile and quantile density function

Usage

```
den2Q_qd(densityCurves, dSup, t_vec)
```

Arguments

densityCurves	n-by-m matrix of density curves
dSup	length m vector contains the common support grid of the density curves
t_vec	common grid for quantile functions

globalFtest	<i>global F test for Wasserstein regression</i>
-------------	---

Description

global F test for Wasserstein regression

Usage

```

globalFtest(
  wass_regress_res,
  alpha = 0.05,
  permutation = FALSE,
  numPermu = 200,
  bootstrap = FALSE,
  numBoot = 200
)

```

Arguments

wass_regress_res	an object returned by the wass_regress function
alpha	type one error rate
permutation	logical; perform permutation global F test (default: FALSE)
numPermu	number of permutation samples if permutation = TRUE
bootstrap	logical; bootstrap global F test (default: FALSE)
numBoot	number of bootstrap samples if bootstrap = TRUE

Details

four methods used to compute p value of global F test

- truncated: asymptotic inference, p-value is obtained by truncating the infinite summation of eigenvalues into the first K terms, where the first K terms explain more than 99.99% of the variance.
- satterthwaite: asymptotic inference, p-value is computed using Satterthwaite's approximation method of mixtures of chi-square.
- permutation: resampling technique; Wasserstein SSR is used as the F statistic.
- bootstrap: resampling technique; Wasserstein SSR is used as the F statistic.

Value

a list containing the following fields:

- wasserstein.F_stat: the Wasserstein F statistic value in Satterthwaite method.
- chisq_df: the degree of freedom of the null chi-square distribution.
- summary_df: a dataframe containing the following columns:
 - method: methods used to compute p value, see details
 - statistic: the test statistics
 - critical_value: critical value
 - p_value: p value of global F test

Examples

```
data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve

res = wass_regress(rightside_formula = ~., Xfit_df = predictor,
Ytype = 'density', Ymat = densityCurves, Sup = dSup)
globalF_res = globalFtest(res, alpha = 0.05, permutation = TRUE, numPermu = 200)
```

partialFtest	<i>partial F test for Wasserstein regression</i>
--------------	--

Description

partial F test for Wasserstein regression

Usage

```
partialFtest(reduced_res, full_res, alpha = 0.05)
```

Arguments

reduced_res	a reduced model list returned by the wass_regress function
full_res	a full model list returned by the wass_regress function
alpha	type one error rate

Details

two methods used to compute p value using asymptotic distribution of F statistic

- truncated: asymptotic inference, p-value is obtained by truncating the infinite summation of eigenvalues into the first K terms, where the first K terms explain more than 99.99% of the variance.
- satterthwaite: asymptotic inference, p-value is computed using Satterthwaite approximation method of mixtures of chi-square.

Value

a dataframe containing the following columns:

- method: methods used to compute p value, see details
- statistic: the test statistics
- critical_value: critical value
- p_value: p value of global F test

Examples

```
data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve

full_res <- wass_regress(rightside_formula = ~., Xfit_df = predictor,
  Ymat = densityCurves, Ytype = 'density', Sup = dSup)
reduced_res <- wass_regress(~ log_b_vol + b_shapInd + midline_shift + B_TimeCT, Xfit_df = predictor,
  Ymat = densityCurves, Ytype = 'density', Sup = dSup)
partialFtable = partialFtest(reduced_res, full_res, alpha = 0.05)
```

predict.WARp	<i>Prediction by WAR(p) models</i>
--------------	------------------------------------

Description

a method of the WARp class which produces a one-step ahead prediction by WAR(p) models

Usage

```
## S3 method for class 'WARp'  
predict(object, dSup, expSup, ...)
```

Arguments

object	A WARp object, the output of WARp().
dSup	Optional, a numeric vector, the grid over which forecasted cdf/pdf is evaluated. Should be supplied/ignored with expSup together.
expSup	Optional, a numeric vector, the grid over the Exponential map is applied, dSup should cover and be denser than expSup. Should be supplied/ignored with dSup together.
...	Further arguments passed to or from other methods.

Value

A list of:

pred.cdf	predicted cdf
pred.pdf	predicted pdf
dSup	support of the predicted cdf/pdf

References

Wasserstein Autoregressive Models for Density Time Series, Chao Zhang, Piotr Kokoszka, Alexander Petersen, 2022

See Also

[WARp](#)

<code>print.summary.WRI</code>	<i>print the summary of WRI object</i>
--------------------------------	--

Description

print the summary of WRI object

Usage

```
## S3 method for class 'summary.WRI'
print(x, ...)
```

Arguments

<code>x</code>	a 'summary.WRI' object
<code>...</code>	further arguments passed to or from other methods.

<code>quan2den_qd</code>	<i>convert density function to quantile and quantile density function</i>
--------------------------	---

Description

convert density function to quantile and quantile density function

Usage

```
quan2den_qd(quantileCurves, t_vec)
```

Arguments

<code>quantileCurves</code>	n-by-m matrix of quantile curves
<code>t_vec</code>	length m vector contains the common support grid of the quantile curves

`simulate_quantile_curves`*Simulate quantile curves*

Description

This function simulates quantile curves used as a toy example

Usage

```
simulate_quantile_curves(x1, alpha, beta, t_vec)
```

Arguments

<code>x1</code>	n-by-1 predictor vector
<code>alpha</code>	parameter in location transformation
<code>beta</code>	parameter in variance transformation
<code>t_vec</code>	a length m vector - common grid for all quantile functions

Value

`quan_obs`: n-by-m matrix of quantile functions

References

Wasserstein F-tests and confidence bands for the Frechet regression of density response curves, Alexander Petersen, Xi Liu and Afshin A. Divani, 2019

Examples

```
alpha = 2
beta = 1
n = 100
x1 = runif(n)
t_vec = unique(c(seq(0, 0.05, 0.001), seq(0.05, 0.95, 0.05), seq(0.95, 1, 0.001)))
quan_obs = simulate_quantile_curves(x1, alpha, beta, t_vec)
```

strokeCTdensity	<i>Stroke data: clinical, radiological scalar variables and density curves of the hematoma of 393 stroke patients</i>
-----------------	---

Description

Stroke data: clinical, radiological scalar variables and density curves of the hematoma of 393 stroke patients

Format

a list of the following three fields:

densityCurve: 393-by-101 head CT hematoma densities as distributional response

densitySupport: length 101 common support vector

predictors: 393-by-9 matrix containing 9 scalar predictors

References

Wasserstein F-tests and confidence bands for the Frechet regression of density response curves, Alexander Petersen, Xi Liu and Afshin A. Divani, 2019

summary.WRI	<i>Summary Function of Wasserstein Regression Model</i>
-------------	---

Description

Summary Function of Wasserstein Regression Model

Usage

```
## S3 method for class 'WRI'
summary(object, ...)
```

Arguments

object	an object returned by the wass_regress function
...	further arguments passed to or from other methods.

Value

a list containing the following fields:

<code>call</code>	function call of the Wasserstein regression
<code>r.square</code>	Wasserstein R^2 , the Wasserstein coefficient of determination
<code>global_wasserstein_F_stat</code>	Wasserstein global F test statistic from the Satterthwaite method
<code>global_F_pvalue</code>	p value of global F test
<code>global_wasserstein_F_df</code>	degrees of freedom of satterthwaite approximated sampling distribution used in global F test
<code>partial_F_table</code>	Partial F test for individual effects

Examples

```
data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve

res <- wass_regress(rightside_formula = ~., Xfit_df = predictor,
Ymat = densityCurves, Ytype = 'density', Sup = dSup)
summary(res)
```

 WARp

WAR(p) models: estimation and forecast

Description

this function produces an object of the WARp class which includes WAR(p) model parameter estimates and relevant quantities (see output list)

Usage

```
WARp(quantile, quantile.grid, p)
```

Arguments

<code>quantile</code>	A matrix containing all the sample quantile functions. Columns represent time indices and rows represent evaluation grid.
<code>quantile.grid</code>	A numeric vector, the grid over which quantile functions are evaluated.
<code>p</code>	A positive integer, the order of the fitted WAR(p) model.

Details

This function takes in a density time series in the form of the corresponding quantile functions as the main input. If the quantile series is not readily available, a general practice is to estimate density functions from samples, then use `dens2quantile` from the `fdadensity` package to convert density time series to quantile series.

Value

A WARp object of:

<code>coef</code>	estimated AR parameters of the fitted WAR(p) model
<code>coef.cov</code>	covariance matrix of <code>coef</code>
<code>acvf</code>	Wasserstein autocovariance function values
<code>Wass.mean</code>	Wasserstein mean quantile function
<code>quantile</code>	a matrix containing all the sample quantile functions (columns represent time indices and rows represent evaluation grid)
<code>quantile.grid</code>	quantile function grid that is utilized in calculation
<code>order</code>	a positive integer, the order of the fitted WAR(p) model

References

Wasserstein Autoregressive Models for Density Time Series, Chao Zhang, Piotr Kokoszka, Alexander Petersen, 2022

Examples

```
# Simulate a density time series represented in quantile functions
# warSimData$sample.ts: A sample TS of quantile functions of length 100, taken from
# the simulation experiments in Section 4 of Zhang et al. 2022.

# warSimData$quantile.grid: The grid over which quantile functions in sample.ts are evaluated.

warSimData <- warSim()

p <- 3
dSup <- seq(-2, 2, 0.02)
expSup <- seq(-2, 2, 0.1)

# Estimation: fit a WAR(3) model
WARp_obj <- WARp(warSimData$sample.ts, warSimData$quantile.grid, p)

# Forecast: one-step-ahead forecast
forecast_1 <- predict(WARp_obj) # dSup and expSup are chosen automatically
forecast_2 <- predict(WARp_obj, dSup, expSup) # dSup and expSup are chosen by user

# Plots
par(mfrow=c(1,2))

plot(forecast_1$dSup, forecast_1$pred.cdf, type="l", xlab="dSup", ylab="cdf")
```

```
plot(forecast_1$dSup, forecast_1$pred.pdf, type="l", xlab="dSup", ylab="pdf")

plot(forecast_2$dSup, forecast_2$pred.cdf, type="l", xlab="dSup", ylab="cdf")
plot(forecast_2$dSup, forecast_2$pred.pdf, type="l", xlab="dSup", ylab="pdf")
```

warSim	<i>Generate simulation data</i>
--------	---------------------------------

Description

Generate WAR(p) simulation data sets: samples simulated from a WAR(3) model similar to the specification in Section 4 of the referenced paper.

Usage

```
warSim()
```

Value

A list of:

sample.ts	one simulation run chosen from sample.ts.full
sample.ts.full	1000 simulation runs, each of which consists of a sample time series (of length 100) of quantile functions generated by a WAR(3) model as specified by the reference paper
quantile.grid	the grid over which the quantile functions in sample.ts.full are evaluated

References

Wasserstein Autoregressive Models for Density Time Series, Chao Zhang, Piotr Kokoszka, Alexander Petersen, 2022

wass_R2	<i>Compute Wasserstein Coefficient of Determination</i>
---------	---

Description

Compute Wasserstein Coefficient of Determination

Usage

```
wass_R2(wass_regress_res)
```

Arguments

wass_regress_res
an object returned by the wass_regress function

Value

Wasserstein R^2 , the Wasserstein coefficient of determination

References

Frechet regression for random objects with Euclidean predictors, Alexander Petersen and Hans-Georg Müller, 2019

Examples

```
data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve

res = wass_regress(rightside_formula = ~., Xfit_df = predictor,
Ymat = densityCurves, Ytype = 'density', Sup = dSup)
wass_r2 = wass_R2(res)
```

wass_regress

Perform Frechet Regression with the Wasserstein Distance

Description

Perform Frechet Regression with the Wasserstein Distance

Usage

```
wass_regress(rightside_formula, Xfit_df, Ytype, Ymat, Sup = NULL)
```

Arguments

rightside_formula
a right-side formula

Xfit_df
n-by-p matrix (or dataframe) of predictor values for fitting (do not include a column for the intercept)

Ytype
'quantile' or 'density'

Ymat
one of the following matrices:

- if Ytype = 'quantile' Ymat is an n-by-m matrix of the observed quantile functions. Ymat[i, :] is a 1-by-m vector of quantile function values on grid Sup.

- if `Ytype = 'density'` `Ymat` is an `n`-by-`m` matrix of the observed density functions. `Ymat[i, :]` is a 1-by-`m` vector of density function values on grid `Sup`.
- `Sup` one of the following vectors:
- if `Ytype = 'quantile'` `Sup` is a length `m` vector - common grid for all quantile functions in `Ymat` (default: `seq(0, 1, length.out = ncol(Ymat))`).
 - if `Ytype = 'density'` `Sup` is a length `m` vector - common grid for all density functions in `Ymat` (default: `seq(0, 1, length.out = ncol(Ymat))`).

Value

a list containing the following objects:

<code>call</code>	function call
<code>rformula</code>	rightside_formula
<code>predictor_names</code>	names of predictors as the colnames given in the <code>xfit</code> matrix or dataframe.
<code>Qfit</code>	<code>n</code> -by- <code>m</code> matrix of fitted quantile functions.
<code>xfit</code>	design matrix in quantile fitting.
<code>Xfit_df</code>	<code>n</code> -by- <code>p</code> matrix (or dataframe) of predictor values for fitting
<code>Yobs</code>	a list containing the following matrices: • <code>Qobs</code>: <code>n</code>-by-<code>m</code> matrix of the observed quantile functions. • <code>qobs</code>: <code>n</code>-by-<code>m</code> matrix of the observed quantile density functions. • <code>qobs_prime</code>: <code>n</code>-by-<code>m</code> matrix of the first derivative of the observed quantile density functions. • <code>fobs</code>: <code>n</code>-by-<code>m</code> matrix of the observed density functions.
<code>t_vec</code>	a length <code>m</code> vector - common grid for all quantile functions in <code>Qobs</code> .

References

Wasserstein F-tests and confidence bands for the Frechet regression of density response curves, Alexander Petersen, Xi Liu and Afshin A. Divani, 2019

Examples

```
data(strokeCTdensity)
predictor = strokeCTdensity$predictors
dSup = strokeCTdensity$densitySupport
densityCurves = strokeCTdensity$densityCurve

res1 = wass_regress(rightside_formula = ~., Xfit_df = predictor,
  Ytype = 'density', Ymat = densityCurves, Sup = dSup)
res2 = wass_regress(rightside_formula = ~ log_b_vol * weight, Xfit_df = predictor,
  Ytype = 'density', Ymat = densityCurves, Sup = dSup)
```

Index

`confidenceBands`, [2](#)

`den2Q_qd`, [4](#)

`globalFtest`, [4](#)

`partialFtest`, [6](#)

`predict.WARp`, [7](#)

`print.summary.WRI`, [8](#)

`quan2den_qd`, [8](#)

`simulate_quantile_curves`, [9](#)

`strokeCTdensity`, [10](#)

`summary.WRI`, [10](#)

`WARp`, [7](#), [11](#)

`warSim`, [13](#)

`wass_R2`, [13](#)

`wass_regress`, [14](#)