Package 'wqspt'

May 8, 2026

Title Permutation Test for Weighted Quantile Sum Regression

Version 1.0.2

Author Drew Day [aut, cre],
James Peng [aut],
Adam Szpiro [aut]

Maintainer Drew Day <dday612@gmail.com>

Description Implements a permutation test method for the weighted quantile sum (WQS) regression, building off the 'gWQS' package (Renzetti et al. <https://CRAN.R-project.org/package=gWQS>). Weighted quantile sum regression is a statistical technique to evaluate the effect of complex exposure mixtures on an outcome (Carrico et al. 2015 <doi:10.1007/s13253-014-0180-3>). The model features a statistical power and Type I error (i.e., false positive) rate trade-off, as there is a machine learning step to determine the weights that optimize the linear model fit. This package provides an alternative method based on a permutation test that should reliably allow for both high power and low false positive rate when utilizing WQS regression (Day et al. 2022 <doi:10.1289/EHP10570>).

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports rlang, gWQS, pbapply, ggplot2, mvtnorm, viridis, extraDistr, cowplot, methods, MASS, car, future, future.apply, pscl, reshape2, nnet

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2025-03-05 08:40:02 UTC

Contents

`wqspt_plot` *Plotting method for wqspt object*

Description

Generates plots to help visualize and summarize WQS permutation test results.

Usage

```
wqspt_plot(
  wqsptresults,
  FixedPalette = FALSE,
  InclKey = FALSE,
  AltMixName = NULL,
  AltOutcomeName = NULL,
  ViridisPalette = "D",
  StripTextSize = 14,
  AxisTextSize.Y = 12,
  AxisTextSize.X = 12,
  LegendTextSize = 14,
  LegendWidthIn = 0.4,
  LegendHeightIn = 0.4,
  PvalLabelSize = 5,
  HeatMapTextSize = 5
)
```

Arguments

`wqsptresults`	An object of class `wqs_pt`.
`FixedPalette`	If TRUE, the heatmap color key for the mixture weights has categorical cutoffs with the following categories: <0.1, 0.1 - <0.2, 0.2 - <0.3, and >= 0.3. If false, the heatmap color key is continuous and dependent on the weight values.
`InclKey`	If TRUE, a horizontal heatmap legend is included at the bottom of the full plot.
`AltMixName`	Defaults to NULL. If not NULL, these are alternative names for the mixture components to be displayed on the heatmap y axis.
`AltOutcomeName`	Defaults to NULL. If not NULL, this is an alternative name for the outcome to be displayed on the heatmap x axis.

ViridisPalette	Color palette to be used for the viridisLite package-based coloring of the heatmap, with possible values from 'A' to 'E'. Defaults to 'D'.
StripTextSize	Text size for the plot strip labels. Defaults to 14.
AxisTextSize.Y	Text size for the y axis text. Defaults to 12.
AxisTextSize.X	Text size for the x axis text. Defaults to 12.
LegendTextSize	Text size for the legend text. Defaults to 14.
LegendWidthIn	Width of the horizontal bottom legend in inches. Defaults to 0.4.
LegendHeightIn	Height of the horizontal bottom legend in inches. Defaults to 0.4.
PvalLabelSize	The geom_text size for the permutation test p-value label. Defaults to 5.
HeatMapTextSize	The geom_text size for the mixture weight heatmap labels. Defaults to 5.

Value

Returns a list with 4 objects.

FullPlot	Two plots stacked vertically: (1) A forest plot of the beta WQS coefficient with the naive confidence intervals as well as the permutation test p-value and (2) a heatmap of the WQS weights for each mixture component.
CoefPlot	Forest plot of the beta WQS coefficient with the naive confidence intervals as well as the permutation test p-value.
WtPlot	A heatmap of the WQS weights for each mixture component.
WtLegend	A legend for the weights in the WtPlot heatmap, saved as a TableGrob object.

Examples

```
library(gWQS)

# mixture names
PCBs <- names(wqs_data)[1:10] #10 of the original 34 for a quick example

#quick example WQSPT model
perm_test_res <- wqs_full_perm(formula = yLBX ~ wqs, data = wqs_data,
                               mix_name = PCBs, q = 10, b_main = 4,
                               b_perm = 4, b1_pos = TRUE, b_constr = FALSE,
                               niter = 3, seed = 16,
                               plan_strategy = "multicore",
                               stop_if_nonsig = FALSE)

#plot with continuous heatmap for mixture weights, omitting the heatmap
# legend on the plot
wqsptplot <- wqspt_plot(perm_test_res)

#plot the full plot with the WQS coefficient forest plot above & the mixture
# weight heatmap below
wqsptplot$FullPlot

#plot with binned heatmap for mixture weights
```

```
wqsptplot <- wqspt_plot(perm_test_res, InclKey = TRUE, LegendWidthIn = 0.8,
FixedPalette = TRUE)

#plot the full plot with the WQS coefficient forest plot above & the mixture
# weight heatmap below
wqsptplot$FullPlot
```

`wqs_full_perm` *Full wrapper WQS permutation test*

Description

`wqs_full_perm` is a full wrapper function that is a full implementation of the Weighted Quantile Sum (WQS) regression method followed by the permutation test to determine the significance of the WQS coefficient.

Usage

```
wqs_full_perm(
  formula,
  data,
  mix_name,
  q = 10,
  b_main = 1000,
  b_perm = 200,
  b1_pos = TRUE,
  b_constr = FALSE,
  rs = FALSE,
  niter = 200,
  seed = NULL,
  family = "gaussian",
  plan_strategy = "multicore",
  stop_if_nonsig = FALSE,
  stop_thresh = 0.05,
  nworkers = NULL,
  ...
)
```

Arguments

`formula`	An object of class formula. The wqs term must be included in the formula (e.g., `y ~ wqs + ...`).
`data`	The `data.frame` to be used in the WQS regression run. This can be of class `data.frame` or it can be a tibble from the tidyverse.

mix_name	A vector with the mixture column names.
q	An integer to indicate the number of quantiles to split the mixture variables.
b_main	The number of bootstraps for the main WQS regression run.
b_perm	The number of bootstraps for the iterated permutation test WQS regression runs and the reference WQS regression run (only for linear WQS regression and only when b_mean != b_perm).
b1_pos	A logical value that indicates whether beta values should be positive or negative.
b_constr	Logical value that determines whether to apply positive or negative constraints in the optimization function for the weight optimization. Note that this won't guarantee that the iterated b1 values in the weight optimization are only positive (if b1_pos = TRUE) or only negative (if b1_pos = FALSE) as seen in the bres matrix output by the gwqs models (i.e., column bres$b1), but it does substantially increase the probability that those b1 values will be constrained to be either positive or negative. This defaults to FALSE.
rs	A logical value indicating whether random subset implementation should be performed.
niter	Number of permutation test iterations.
seed	An integer to fix the seed. This will only impact the the initial WQS regression run and not the permutation test iterations. The default setting is NULL, which means no seed is used for the initial WQS regression. The seed will be saved in the "gwqs_main" object as "gwqs_main$seed".
family	A description of the error distribution and link function to be used in the model. This can be a character string naming a family function (e.g., "binomial") or a family object (e.g., binomial(link="logit")). Currently validated families include gaussian(link="identity") for linear regression, binomial() with any accepted link function (e.g., "logit" or "probit"), poisson(link = "log"), quasipoisson(link = "log"), or "negbin" for negative binomial. The "multinomial" family is not yet supported.
plan_strategy	Evaluation strategy for the plan function. You can choose among "sequential", "multisession", "multicore", and "cluster". This defaults to "multicore". See the future::plan documentation for full details.
stop_if_nonsig	if TRUE, the function will not proceed with the permutation test if the main WQS regression run produces nonsignificant p-value.
stop_thresh	numeric p-value threshold required in order to proceed with the permutation test, if stop_if_nonsig = TRUE.
nworkers	(optional) If the plan_strategy is not "sequential", this argument defines the number of parallel processes to use, which can be critical when using a high-performance computing (HPC) cluster. This should be an integer value. The default behavior for gWQS::gwqs is to use all detected cores on a machine, but for many HPC use scenarios, this would call in cores that have not been allotted by the HPC scheduler, resulting in the submitted job being halted. For example, if one has requested 14 cores on a 28-core HPC queue, one would want to set nworkers = 14. If nworkers was greater than 14 in that case, the HPC job would be terminated. This argument defaults to NULL, in which case length(future::availableWorkers()) will be used to determine the number of parallel processes to use.

... (optional) Additional arguments to pass to the gWQS::gwqs function.

Value

wqs_full_perm returns an object of class wqs_perm, which contains three sublists:

perm_test List containing:

- pval: permutation test p-value
- (linear regression only) testbeta1: reference WQS regression coefficient beta1 value
- (linear regression only) betas: Vector of beta values from each permutation test run
- (logistic regression only) testpval: test reference p-value
- (logistic regression only) permpvals: p-values from the null models

gwqs_main Main gWQS object (same as model input). This will now include an additional object "seed" that returns the seed used for this main WQS regression.

gwqs_perm Permutation test reference gWQS object (NULL if model family != "gaussian" or if same number of bootstraps are used in permutation test WQS regression runs as in the main run).

Examples

```
library(gWQS)

# mixture names
PCBs <- names(wqs_data)[1:5]
# Only using 1st 5 of the original 34 exposures for this quick example

# quick example with only 3 bootstraps each WQS regression iteration, and
# only 2 iterations

perm_test_res <- wqs_full_perm(formula = yLBX ~ wqs, data = wqs_data,
                               mix_name = PCBs, q = 10, b_main = 3,
                               b_perm = 3, b1_pos = TRUE, b_constr = FALSE,
                               niter = 2, seed = 16,
                               plan_strategy = "multicore",
                               stop_if_nonsig = FALSE)

# Note: The default values of b_main = 1000, b_perm = 200, and niter = 200
# are the recommended parameter values. This example has a lower b_main,
# b_perm, and niter in order to serve as a shorter test run.
```

wqs_pt *WQS permutation test*

Description

wqs_pt takes a gwqs object as an input and runs the permutation test (Day et al. 2022) to obtain an estimate for the p-value significance for the WQS coefficient.

Usage

```
wqs_pt(
  model,
  niter = 200,
  boots = NULL,
  b1_pos = TRUE,
  b_constr = FALSE,
  rs = FALSE,
  plan_strategy = "multicore",
  seed = NULL,
  nworkers = NULL,
  ...
)
```

Arguments

model	A gwqs object as generated from the gWQS package.
niter	Number of permutation test iterations.
boots	Number of bootstrap samples for each permutation test WQS regression iteration. If boots is not specified, then we will use the same bootstrap count for each permutation test WQS regression iteration as was specified in the main WQS regression run.
b1_pos	A logical value that indicates whether beta values should be positive or negative.
b_constr	Logical value that determines whether to apply positive or negative constraints in the optimization function for the weight optimization. Note that this won't guarantee that the iterated b1 values in the weight optimization are only positive (if b1_pos = TRUE) or only negative (if b1_pos = FALSE) as seen in the bres matrix output by the gwqs models (i.e., column bres$b1), but it does substantially increase the probability that those b1 values will be constrained to be either positive or negative. This defaults to FALSE.
rs	A logical value indicating whether random subset implementation should be performed.
plan_strategy	Evaluation strategy for the plan function. You can choose among "sequential", "transparent", "multisession", "multicore", and "cluster". This defaults to "multicore". See the future::plan documentation for full details.

seed	(optional) Random seed for the permutation test WQS reference run. This should be the same random seed as used for the main WQS regression run. This seed will be saved in the "gwqs_perm" object as gwqs_perm$seed. This defaults to NULL.
nworkers	(optional) If the plan_strategy is not "sequential", this argument defines the number of parallel processes to use, which can be critical when using a high-performance computing (HPC) cluster. This should be an integer value. The default behavior for gWQS::gwqs is to use all detected cores on a machine, but for many HPC use scenarios, this would call in cores that have not been allotted by the HPC scheduler, resulting in the submitted job being halted. For example, if one has requested 14 cores on a 28-core HPC queue, one would want to set nworkers = 14. If nworkers was greater than 14 in that case, the HPC job would be terminated. This argument defaults to NULL, in which case length(future::availableWorkers()) will be used to determine the number of parallel processes to use.
...	(optional) Additional arguments to pass to the gWQS::gwqs function.

Details

To use wqs_pt, we first need to run an initial WQS regression run while setting validation = 0. We will use this gwqs object as the model argument for the wqs_pt function. Note that permutation test has so far only been validated for linear WQS regression (i.e., family = "gaussian") or logistic WQS regression (i.e., family = binomial(link = "logit")), though the permutation test algorithm should also work for all WQS GLMs. Therefore, this function accepts gwqs objects made with the following families: "gaussian" or gaussian(link = "identity"), "binomial" or binomial() with any accepted link function (e.g., "logit" or "probit"), "poisson" or poisson(link="log"), "negbin" for negative binomial, and "quasipoisson" or quasipoisson(link="log"). This function cannot currently accommodate gwqs objects made with the "multinomial" family, and it is not currently able to accommodate stratified weights or WQS interaction terms (e.g., y ~ wqs * sex).

The argument boots is the number of bootstraps for the WQS regression run in each permutation test iteration. Note that we may elect a bootstrap count boots lower than that specified in the model object for the sake of efficiency. If boots is not specified, then we will use the same bootstrap count in the permutation test WQS regression runs as that specified in the model argument.

The arguments b1_pos and rs should be consistent with the inputs chosen in the model object. The seed should ideally be consistent with the seed set in the model object for consistency, though this is not required.

Value

wqs_pt returns an object of class "wqs_pt", which contains:

perm_test	List containing: (1) pval: permutation test p-value, (2) (linear WQS regression only) testbeta1: reference WQS coefficient beta1 value, (3) (linear WQS regression only) betas: Vector of beta values from each permutation test run, (4) (WQS GLM only) testpval: test reference p-value, (5) (WQS GLM only) permpvals: p-values from the null models.
gwqs_main	Main gWQS object (same as model input).

gwqs_perm — Permutation test reference gWQS object (NULL if model `family != "gaussian"` or if same number of bootstraps are used in permutation test WQS regression runs as in the main run).

References

Day, D. B., Sathyanarayana, S., LeWinn, K. Z., Karr, C. J., Mason, W. A., & Szpiro, A. A. (2022). A permutation test-based approach to strengthening inference on the effects of environmental mixtures: comparison between single index analytic methods. Environmental Health Perspectives, 130(8).

Day, D. B., Collett, B. R., Barrett, E. S., Bush, N. R., Swan, S. H., Nguyen, R. H., ... & Sathyanarayana, S. (2021). Phthalate mixtures in pregnancy, autistic traits, and adverse childhood behavioral outcomes. Environment International, 147, 106330.

Loftus, C. T., Bush, N. R., Day, D. B., Ni, Y., Tylavsky, F. A., Karr, C. J., ... & LeWinn, K. Z. (2021). Exposure to prenatal phthalate mixtures and neurodevelopment in the Conditions Affecting Neurocognitive Development and Learning in Early childhood (CANDLE) study. Environment International, 150, 106409.

Examples

```
library(gWQS)

# mixture names
PCBs <- names(wqs_data)[1:5]
 # Only using 1st 5 of the original 34 exposures for this quick example

# create reference wqs object with 4 bootstraps
wqs_main <- gwqs(yLBX ~ wqs, mix_name = PCBs, data = wqs_data, q = 10,
                 validation = 0, b = 3, b1_pos = TRUE, b_constr = FALSE,
                 plan_strategy = "multicore", family = "gaussian", seed = 16)
# Note: We recommend niter = 1000 for the main WQS regression. This example
# has a lower number of bootstraps to serve as a shorter test run.

# run the permutation test
perm_test_res <- wqs_pt(wqs_main, niter = 2, b1_pos = TRUE)

# Note: The default value of niter = 200 is the recommended parameter value.
# This example has a lower niter in order to serve as a shorter test run.
```

---

wqs_sim — *WQS simulated dataset generator*

Description

`wqs_sim` generates a simulated dataset of mixture components, covariates, and outcomes based on an initial set of specifications.

Usage

```
wqs_sim(
  nmix = 10,
  ncovrt = 10,
  nobs = 500,
  ntruewts = 10,
  ntruecovrt = 5,
  vcov = 0,
  eps = 1,
  truewqsbeta = NULL,
  truebeta0 = NULL,
  truewts = NULL,
  truegamma = NULL,
  rnd_wqsbeta_dir = "none",
  seed = 101,
  q = 10,
  family = "gaussian"
)
```

Arguments

nmix	Number of mixture components in simulated dataset.
ncovrt	Number of covariates in simulated dataset.
nobs	Number of observations in simulated dataset.
ntruewts	Number of mixture components that have a non-zero association with the outcome (i.e., are not noise).
ntruecovrt	Number of covariates that have a non-zero association with the outcome (i.e., are not noise).
vcov	This parameter relates to the variance-covariance matrix of the simulated independent variables (i.e., the m exposure mixture components and z covariates). This is either a variance-covariance matrix of dimensions (m + z) x (m + z) or a single value. If this is a single value, the variance- covariance matrix will have ones on the diagonal and that single value will be all the off-diagonal values. For example, if this input were 0.4 and there were two mixture components and no covariates, the variance-covariance matrix would be `matrix(c(1, 0.4, 0.4, 1), nrow = 2, ncol = 2)`. The default value is 0, giving a variance-covariance matrix with variances of 1 and covariances of 0.
eps	Dispersion parameter. If the family is "gaussian", this corresponds to the residual standard deviation. If the family is "binomial" or "poisson", this parameter is ignored. If the family is "negbin", this represents the "size" parameter of the negative binomial distribution (see the documentation for the `rnbinom` function for more details).
truewqsbeta	Simulated WQS beta_1 value. If NULL, then this value will be randomly sampled depending on the parameter `rnd_wqsbeta_dir`.
truebeta0	Simulated beta_0 value. If NULL, then this value will be randomly sampled from a standard normal distribution.

truewts	Simulated vector of mixture weights. If NULL, then this value will be randomly sampled from a Dirichlet distribution with a vector of alpha values all equal to 1 (see the documentation for the extraDistr::rdirichlet function documentation for more details).
truegamma	Simulated gamma vector. If NULL, then this value will be randomly sampled from a standard normal distribution.
rnd_wqsbeta_dir	Direction of randomly sampled truewqsbeta (if truewqsbeta = NULL). The options are "positive", "negative", or NULL. If "positive" or "negative", the truewqsbeta will be sampled from a standard half normal distribution in either of those respective directions. If NULL, then truewqsbeta will be sampled from a standard normal distribution.
seed	Random seed. This defaults to 101.
q	Number of quantiles. This defaults to 10.
family	Family for the generative model creating the outcome vector. Options include "gaussian" or gaussian(link = "identity") for a continuous outcome, "binomial" or binomial() with any accepted link function for a binary outcome, and finally for count outcomes this can be "poisson" or poisson(link="log") for the Poisson distributed outcome values, or "negbin" for negative binomial distributed outcome values.

Value

wqs_perm returns a list of:

weights	Simulated weights.
coef	Simulated beta coefficients.
Data	Simulated dataset.
etahat	predicted linear predictor (eta) values from the data generating model.
wqs	Weighted quantile sum vector (quantile-transformed mixture components multiplied by weights).
modmat	Model matrix.
Xq	Quantile-transformed mixture components.

Examples

```
# For these examples, we only run a GLM using the simulated dataset
# including the simulated WQS vector just to show that the user-specified
# coefficients for beta1 and beta0 are returned. An example of running
# the full permutation test WQS regression for the simulated dataset
# (for which the WQS vector would be determined by the model)
# with the "gaussian" family is shown as well.

wqsform<-formula(paste0("y~wqs+",paste(paste0("C",1:10),collapse="+")))

testsim_gaussian<-
  wqs_sim(truewqsbeta=0.2,truebeta0=-2,
```

```
          truewts=c(rep(0.15,5),rep(0.05,5)),family="gaussian")
Dat<-testsim_gaussian$Data
Dat$wqs<-testsim_gaussian$wqs
summary(glm(wqsform,data=Dat,family="gaussian"))$coef[1:2,]

perm_test_res <- wqs_full_perm(formula = wqsform, data = testsim_gaussian$Data,
                               mix_name = paste0("T",1:10), q = 10, b_main = 5,
                               b_perm = 5, b1_pos = TRUE, b_constr = FALSE,
                               niter = 4, seed = 16, plan_strategy = "multicore",
                               stop_if_nonsig = FALSE)

# Note: The default values of b_main = 1000, b_perm = 200, and niter = 200
# are the recommended parameter values. This example has a lower b_main,
# b_perm, and niter in order to serve as a shorter example run.


testsim_logit<-
  wqs_sim(truewqsbeta=0.2,truebeta0=-2,
          truewts=c(rep(0.15,5),rep(0.05,5)),family="binomial")
Dat<-testsim_logit$Data
Dat$wqs<-testsim_logit$wqs
summary(glm(wqsform,data=Dat,family="binomial"))$coef[1:2,]
```

Index