Package 'gWQS'

October 13, 2022

Type Package

Title Generalized Weighted Quantile Sum Regression

Version 3.0.4

Author Stefano Renzetti, Paul Curtin, Allan C Just, Ghalib Bello, Chris Gennings

Maintainer Stefano Renzetti <`stefano.renzetti88@gmail.com`>

Description Fits Weighted Quantile Sum (WQS) regression (Carrico et al. (2014) <doi:10.1007/s13253-014-0180-3>), a random subset implementation of WQS (Curtin et al. (2019) <doi:10.1080/03610918.2019.1577971>) and a repeated holdout validation WQS (Tanner et al. (2019) <doi:10.1016/j.mex.2019.11.008>) for continuous, binomial, multinomial, Poisson, quasi-Poisson and negative binomial outcomes.

Imports ggplot2, dplyr, stats, broom, rlist, MASS, reshape2, plotROC, knitr, kableExtra, nnet, future, future.apply, pscl, ggrepel, cowplot, Matrix

Suggests markdown

License GPL (>= 2)

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2021-05-20 09:30:02 UTC

R topics documented:

gwqs 2
gwqs_barplot 6
summary.gwqs 8
wqs_data 12

Index **14**

---

gwqs *Fitting Weighted Quantile Sum regression models*

Description

Fits Weighted Quantile Sum (WQS) regression (Carrico et al. (2014) doi: 10.1007/s132530140180-3), a random subset implementation of WQS (Curtin et al. (2019) doi: 10.1080/03610918.2019.1577971) and a repeated holdout validation WQS (Tanner et al. (2019) doi: 10.1016/j.mex.2019.11.008) for continuous, binomial, multinomial, Poisson, quasi-Poisson and negative binomial outcomes.

Usage

```
gwqs(formula, data, na.action, weights, mix_name, stratified, valid_var, b = 100,
             b1_pos = TRUE, b1_constr = FALSE, zero_infl = FALSE, q = 4,
         validation = 0.6, family = gaussian, signal = c("t2", "one", "abst", "expt"),
             rs = FALSE, n_vars = NULL,
          zilink = c("logit", "probit", "cloglog", "cauchit", "log"), seed = NULL,
             plan_strategy = "sequential",
             optim.method = c("BFGS", "Nelder-Mead", "CG", "SANN"),
             control = list(trace = FALSE, maxit = 2000, reltol = 1e-9), ...)

gwqsrh(formula, data, na.action, weights, mix_name, stratified, valid_var, rh = 100,
             b = 100, b1_pos = TRUE, b1_constr = FALSE, zero_infl = FALSE, q = 4,
               validation = 0.6, family = gaussian,
              signal = c("t2", "one", "abst", "expt"), rs = FALSE, n_vars = NULL,
          zilink = c("logit", "probit", "cloglog", "cauchit", "log"), seed = NULL,
               plan_strategy = "sequential",
               optim.method = c("BFGS", "Nelder-Mead", "CG", "SANN"),
               control = list(trace = FALSE, maxit = 2000, reltol = 1e-9), ...)
```

Arguments

`formula`	An object of class `formula` specifying the relationship to be tested. The `wqs` term must be included in `formula`, e.g. `y ~ wqs + ...`. To test for an interaction term with a continuous variable `a` or for a quadratic term we can specify the `formula` as below: `y ~ wqs*a + ...` and `y ~ wqs + I(wqs^2) + ...`, respectively.
`data`	The `data.frame` containing the variables to be included in the model.
`na.action`	`model.frame`. `na.omit` is the default.
`weights`	An optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector.
`mix_name`	A character vector listing the variables contributing to a mixture effect.
`stratified`	The character name of the variable for which you want to stratify for. It has to be a `factor`.

`valid_var`	A character value containing the name of the variable that identifies the validation and the training dataset. You previously need to create a variable in the dataset which is equal to 1 for the observations you want to include in the validation dataset, equal to 0 for the observation you want to include in the training dataset (use 0 also for the validation dataset if you want to train and validate the model on the same data) and equal to 2 if you want to keep part of the data for the predictive model.
`b`	Number of bootstrap samples used in parameter estimation.
`b1_pos`	A logical value that determines whether weights are derived from models where the beta values were positive or negative.
`b1_constr`	A logial value that determines whether to apply positive (if `b1_pos = TRUE`) or negative (if `b1_pos = FALSE`) constraints in the optimization function for the weight estimation.
`zero_infl`	A logical value (`TRUE` or `FALSE`) that allows to fit a zero inflated model in case `family = "poisson"` or `family = "negbin"`.
`q`	An `integer` to specify how mixture variables will be ranked, e.g. in quartiles (`q = 4`), deciles (`q = 10`), or percentiles (`q = 100`). If `q = NULL` then the values of the mixture variables are taken (these must be standardized).
`validation`	Percentage of the dataset to be used to validate the model. If `validation = 0` then the test dataset is used as validation dataset too.
`family`	A character value that allows to decide for the glm: `gaussian` for linear regression, `binomial` for logistic regression `"multinomial"` for multinomial regression, `poisson` for Poisson regression, `quasipoisson` for quasi-Poisson regression, `"negbin"` for negative binomial regression.
`signal`	Character identifying the signal function to be used when the average weights are estimated. It can take values from `"one"` to apply the identity, `"abst"` to apply the absolute value of the t-statistic, `"t2"` to apply the squared value of the t-statistic, `"expt"` to apply the exponential of the t-statistic as signal function.
`rs`	A logic value. If `rs = FALSE` then the bootstrap implementation of WQS is performed. If `rs = TRUE` then the random subset implementation of WQS is applied (see the "Details" and the vignette for further infromation).
`n_vars`	The number of mixture components to be included at each random subset step. If `rs = TRUE` and `n_vars = NULL` then the square root of the number of elements in the mixture is taken.
`zilink`	Character specification of link function in the binary zero-inflation model (you can choose among `"logit"`, `"probit"`, `"cloglog"`, `"cauchit"`, `"log"`).
`seed`	An `integer` value to fix the seed, if it is equal to `NULL` no seed is chosen.
`plan_strategy`	A character value that allows to choose the evaluation strategies for the `plan` function. You can choose among "sequential", "transparent", "multisession", "multicore", "multiprocess", "cluster" and "remote" (see `plan` help page for more details).
`optim.method`	A character identifying the method to be used by the `optim` function (you can choose among `"BFGS"`, `"Nelder-Mead"`, `"CG"`, `"SANN"`, `"BFGS"` is the default). See `optim` for details.

`control`	The control list of optimization parameters. See `optim` for details.
`...`	Additional arguments to be passed to the function
`rh`	Number of repeated holdout validations. This option is only available for `gwqsrh` function.

Details

`gWQS` uses the `glm` function in the **stats** package to fit the linear, logistic, the Poisson and the quasi-Poisson regression, while the `glm.nb` function from the **MASS** package is used to fit the negative binomial regression respectively. The `nlm` function from the **stats** package was used to optimize the log-likelihood of the multinomial regression.

The `optim` optimization function is used to estimate the weights at each bootstrap step.

The `seed` argument specifies a fixed seed through the `set.seed` function.

The `rs` term allows to choose the type of methodology between the bootstrap implementation (WQSBS) or the random subset implementation (WQSRS) of the WQS. The first method performs `b` bootstrapped samples to estimate the weights while the second creates `b` randomly-selected subset of the total predictor set. For further details please see the vignette ("How to use gWQS package") and the references below.

Value

`gwqs` return the results of the WQS regression as well as many other objects and datasets.

`fit`	The object that summarizes the output of the WQS model, reflecting a linear, logistic, multinomial, Poisson, quasi-Poisson or negative binomial regression depending on how the `family` parameter was specified. The summary function can be used to call and print fit data (not for multinomial regression).
`final_weights`	`data.frame` containing the final weights associated to each chemical.
`conv`	Indicates whether the solver has converged (0) or not (1 or 2).
`bres`	Matrix of estimated weights, mixture effect parameter estimates and the associated standard errors, statistics and p-values estimated for each bootstrap iteration.
`wqs`	Vector containing the wqs index for each subject.
`qi`	List of the cutoffs used to divide in quantiles the variables in the mixture
`bindex`	List of vectors containing the `rownames` of the subjects included in each bootstrap dataset.
`tindex`	Vector containing the rows used to estimate the weights in each bootstrap.
`vindex`	Vector containing the rows used to estimate the parameters of the final model.
`y_wqs_df`	`data.frame` containing the dependent variable values adjusted for the residuals of a fitted model adjusted for covariates (original values when `family = binomial` or `"multinomial"`) and the wqs index estimated values.
`family`	The family specified.

`call`	The matched call.
`formula`	The formula supplied.
`mix_name`	The vector of variable names used to identify the elements in the mixture.
`q`	The method used to rank varibales included in the mixture.
`n_levels`	The number of levels of the of the dependent variable when a multinomial regression is ran.
`zero_infl`	If a zero inflated model was ran (`TRUE`) or not (`FALE`)
`zilink`	The chosen link function when a zero inflated model was ran.
`levelnames`	The name of each level when a multinomial regression is ran.
`data`	The data used in the WQS analysis.
`objfn_values`	The vector of the b values of the objective function corresponding to the optima values
`optim_messages`	The vector of character strings giving any additional information returned by the optimizer, or NULL.
`gwqslist`	List of the output from the `rh` WQS models.
`coefmat`	Matrix containing the parameter estimates from each repeated holdout WQS model.
`wmat`	Matrix containing the weight estimates from each repeated holdout WQS model.

Author(s)

Stefano Renzetti, Paul Curtin, Allan C Just, Ghalib Bello, Chris Gennings

References

Carrico C, Gennings C, Wheeler D, Factor-Litvak P. Characterization of a weighted quantile sum regression for highly correlated data in a risk analysis setting. J Biol Agricul Environ Stat. 2014:1-21. ISSN: 1085-7117. doi: 10.1007/s1325301401803.

Czarnota J, Gennings C, Colt JS, De Roos AJ, Cerhan JR, Severson RK, Hartge P, Ward MH, Wheeler D. 2015. Analysis of environmental chemical mixtures and non-Hodgkin lymphoma risk in the NCI-SEER NHL study. Environmental Health Perspectives, doi: 10.1289/ehp.1408630.

Czarnota J, Gennings C, Wheeler D. 2015. Assessment of weighted quantile sum regression for modeling chemical mixtures and cancer risk. Cancer Informatics, 2015:14(S2) 159-171 doi: 10.4137/CIN.S17295.

Brunst KJ, Sanchez Guerra M, Gennings C, et al. Maternal Lifetime Stress and Prenatal Psychological Functioning and Decreased Placental Mitochondrial DNA Copy Number in the PRISM Study. Am J Epidemiol. 2017;186(11):1227-1236. doi: 10.1093/aje/kwx183.

Curtin P, Kellogg J, Cech N, Gennings C. 2019. A random subset implementation of weighted quantile sum (WQSRS) regression for analysis of high-dimensional mixtures, Communications in

Statistics - Simulation and Computation. doi: 10.1080/03610918.2019.1577971.

Tanner EM, Bornehag CG, Gennings C. Repeated holdout validation for weighted quantile sum regression. MethodsX. 2019 Nov 22;6:2855-2860. doi: 10.1016/j.mex.2019.11.008.

See Also

glm, glm.nb, multinom, zeroinfl.

Examples

```
# we save the names of the mixture variables in the variable
# "toxic_chems"
toxic_chems = names(wqs_data)[1:34]

# To run a linear model and save the results in the variable
# "results". This linear model (family = gaussian) will
# rank/standardize variables in deciles (q = 10), perform a
# 40/60 split of the data for training/validation
# (validation = 0.6), and estimate weights over 2 bootstrap
# samples (b = 2; in practical applications at least 100
# bootstraps should be used). Weights will be derived from
# mixture effect parameters that are positive (b1_pos = TRUE).
# A unique seed was specified (seed = 2016) so this model will
# be reproducible, and plots describing the variable weights
# and linear relationship will be generated as output
# (plots = TRUE). In the end tables describing the weights
# values and the model parameters with the respectively
# statistics are generated in the plots window (tables = TRUE):
results = gwqs(yLBX ~ wqs, mix_name = toxic_chems,
               data = wqs_data, q = 10, validation = 0.6,
               b = 2, b1_pos = TRUE, b1_constr = FALSE,
               family = gaussian, seed = 2016)

# to test the significance of the covariates
summary(results)
```

`gwqs_barplot` *Plots and tables functions*

Description

Functions that allow to generate plots and tables helping in visualizing and summarise Weighted Quantile Sum (WQS) regression results.

Usage

```
gwqs_barplot(object, tau, ...)

gwqs_scatterplot(object, ...)

gwqs_fitted_vs_resid(object, sumtype = c("norm", "perc"), ...)

gwqs_levels_scatterplot(object, ...)

gwqs_ROC(object, newdata, sumtype = c("norm", "perc"), ...)

gwqsrh_boxplot(object, tau, ...)

gwqs_summary_tab(object, sumtype = c("norm", "perc"), ...)

gwqs_weights_tab(object, ...)

selectdatavars(data, na.action, formula, mix_name, ...)

gwqs_rank(data, mix_name, q)
```

Arguments

`object`	An object of class "gwqs" as returned by `gwqs`.
`tau`	A number identifying the cutoff for the significant weights. Is tau is missing then reciprocal of the number of elements in the mixture is considered. To avoid printing the threshold line set `tau` = `NULL`.
`...`	Further arguments to be passed to the function.
`sumtype`	Type of summary statistic to be used: "norm" takes the mean of the estimated parameters on the validation sets and the 95 as the parameters estimates and the 2.5, 97.5 percentiles as CI. This option is only available for objects of class `gwqsrh`.
`newdata`	A data frame in which to look for variables with which to predict and generate the ROC curve.
`data`	Dataset from which you want to select the variables you are interested in.
`na.action`	Allows to choose what action has to be taken to deal with NAs.
`formula`	Formula used in the model to specify the dependent and independent variables.
`mix_name`	Vector containing element names included in the mixture.
`q`	An `integer` to specify how mixture variables will be ranked, e.g. in quartiles (`q` = `4`), deciles (`q` = `10`), or percentiles (`q` = `100`).

Details

The `gwqs_barplot`, `gwqs_scatterplot`, `gwqs_fitted_vs_resid`, `gwqs_levels_scatterplot`, `gwqs_ROC` and `gwqsrh_boxplot` functions produce five figures through the `ggplot` function.

The `gwqs_summary_tab` and `gwqs_weights_tab` functions produce two tables in the viewr pane through the use of the `kable` and `kable_styling` functions.

The `gwqs_barplot`, `gwqs_scatterplot` plots are available for all family types while `gwqs_fitted_vs_resid` is not available when `family = binomial` or `"multinomial"`. `gwqs_levels_scatterplot` plot is only available when `family = "multinomial"` and `gwqs_ROC` when `family = binomial`. All these plots can also be applied to the objects of class `gwqsrh`. For these objects an additional plot is available through the function `gwqs_boxplot`.

The `gwqs_rank` function allows to split the variables selected through the vector `mix_name` in quantiles (depending by the value assigned to `q`).

Value

All the plot functions print the output in the Plots pane while the table functions print the output in the Viewer pane.

`Qm`	The matrix containing the quantiled variables of the elements included in the mixture.
`qi`	A list of vectors containing the cut points used to determine the quantiled variables.

Author(s)

Stefano Renzetti, Paul Curtin, Allan C Just, Ghalib Bello, Chris Gennings

Examples

```
toxic_chems = names(wqs_data)[1:34]
results = gwqs(yLBX ~ wqs, mix_name = toxic_chems, data = wqs_data, q = 4, validation = 0.6,
               b = 1, b1_pos = TRUE, b1_constr = FALSE, family = gaussian)

# barplot
gwqs_barplot(results)

# scatterplot
gwqs_scatterplot(results)

# fitted values vs rediduals scatterplot
gwqs_fitted_vs_resid(results)
```

`summary.gwqs` *Methods for gwqs objects*

Description

Methods for extracting information from fitted Weighted Quantile Sum (WQS) regression model objects of class "gwqs".

Usage

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

## S3 method for class 'gwqsrh'
summary(object, sumtype = c("norm", "perc"), ...)

## S3 method for class 'gwqs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'gwqsrh'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.gwqs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.gwqsrh'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'gwqs'
predict(
  object,
  newdata,
  type = c("link", "response", "prob", "count", "zero", "class", "probs", "terms"),
  ...
)

## S3 method for class 'gwqsrh'
predict(
  object,
  newdata,
  sumtype = c("norm", "perc"),
  type = c("link", "response", "prob", "count", "zero", "class", "probs", "terms"),
  ...
)

## S3 method for class 'gwqs'
coef(object, ...)

## S3 method for class 'gwqsrh'
coef(object, sumtype = c("norm", "perc"), ...)

## S3 method for class 'gwqs'
vcov(object, model = c("full", "count", "zero"), ...)

## S3 method for class 'gwqsrh'
vcov(object, model = c("full", "count", "zero"), ...)
```

```
## S3 method for class 'gwqs'
fitted(object, type = c("prob", "response"), ...)

## S3 method for class 'gwqsrh'
fitted(object, sumtype = c("norm", "perc"), type = c("prob", "response"), ...)

## S3 method for class 'gwqs'
residuals(
  object,
  type = c("deviance", "pearson", "working", "response", "partial"),
  ...
)

## S3 method for class 'gwqsrh'
residuals(
  object,
  sumtype = c("norm", "perc"),
  type = c("pearson", "response"),
  ...
)
```

Arguments

`object, x`	An object of class "gwqs" as returned by gwqs.
`...`	Further arguments to be passed.
`sumtype`	Type of summary statistic to be used: "norm" takes the mean of the estimated parameters on the validation sets and the 95 as the parameters estimates and the 2.5, 97.5 percentiles as CI. This option is only available for objects of class `gwqsrh`.
`digits`	The number of significant digits to use when printing.
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
`type`	Character specifying the type of predictions, fitted values or residuals, respectively. For details see below.
`model`	Character specifying for which component of the model the varance-covariance matrix should be extracted when zero_infl = TRUE.

Details

A set of standard extractor functions for fitted model objects is available for objects of class "gwqs", including methods to the generic functions print and summary which print the estimated coefficients along with some further information. As usual, the summary method returns an object of class "summary.gwqs" containing the relevant summary statistics which can subsequently be printed using the associated print method.

The methods for coef and vcov by default return a single vector of coefficients (a matrix when family = "multinomial") and their associated covariance matrix, respectively. By setting the model argument, the estimates for the corresponding model components can be extracted.

Both the `fitted` and predict methods can compute fitted responses. The latter sets the default on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. Type can be equal to "prob", "count" or "zero" when zero_infl = T to estimate the predicted density (i.e., probabilities for the observed counts), the predicted mean from the count component (without zero inflation) and the predicted probability for the zero component. Type = "class" allow to predict the dependent variable categories when family = "multinomial". The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.

The residuals method allows to extracts model residuals from the objects of class "gwqs".

Value

All these methods return the classic output as for the corresponding glm, glm.nb, multinom and zeroinfl classes. Only the predict method gives a different output made of the following values.

`df_pred`	A data.frame containing the dependent varible and the predicted values.
`Q`	The matrix containing the new dataset quantiled variables of the elements included in the mixture.
`qi`	A list of vectors containing the cut points used to determine the quantiled variables.
`wqs`	The vetor containing the wqs index built on the new dataset.

Author(s)

Stefano Renzetti, Paul Curtin, Allan C Just, Ghalib Bello, Chris Gennings

Examples

```
toxic_chems = names(wqs_data)[1:34]
set.seed(1234)
rws <- sample(1:500, 150)
results = gwqs(yLBX ~ wqs, mix_name = toxic_chems, data = wqs_data[-rws,], q = 4, validation = 0.6,
               b = 1, b1_pos = TRUE, b1_constr = FALSE, family = gaussian)

# to test the significance of the covariates
summary(results)

# extract regression coefficients
coef(results)

# estimate variance-covariance matrix
vcov(results)

# estimate fitted values
fitted(results)

# estimate regression residuals
residuals(results)
```

```
# estimate predicted values on the left part of wqs_data
pred_res <- predict(results, wqs_data[rws,])
pred_res$df_pred
```

`wqs_data` *Exposure concentrations of 34 PCB (simulated dataset)*

Description

We created the ‘wqs_data‘ dataset to show how to use this function. These data reflect 59 exposure concentrations simulated from a distribution of 34 PCB exposures and 25 phthalate biomarkers measured in subjects participating in the NHANES study (2001-2002). Additionally, 8 outcome measures were simulated applying different distributions and fixed beta coefficients to the predictors. In particular ‘y‘ and ‘yLBX‘ were simulated from a normal distribution, ‘ybin‘ and ‘ybinLBX‘ from a binomial distribution, ‘ymultinom‘ and ‘ymultinomLBX‘ from a multinomial distribution and ‘ycount‘ and ‘ycountLBX‘ from a Poisson distribution. The regression coefficients used to generate the outcomes ‘yLBX‘, ‘ybinLBX‘ and ‘ycountLBX‘ were set to:
LBX105LA = 0.3
LBX138LA = 0.6
LBX157LA = 0.2
LBXD02LA = 0.45
LBXD04LA = 0.15
LBXF06LA = 0.3
LBXF07LA = 0.45
then the following terms were added to generate the variables ‘y‘, ‘ybin‘ and ‘ycount‘:
URXMC1 = 0.15
URXMOH = 0.45
URXP02 = 0.2
URXP10 = 0.3
URXUCR = 0.2
All the remaining coefficients were set to 0.
The coefficients to generate ‘ymultinomLBX‘ were set as below:
level B:
LBX138LA = 0.8
LBXD04LA = 0.2
level C:
LBX105LA = 0.4
LBX157LA = 0.3
LBXD02LA = 0.6
LBXF06LA = 0.4
LBXF07LA = 0.6
and the following terms were added for ‘ymultinom‘:
level B:
URXMC1 = 0.2
URXP02 = 0.3
URXP10 = 0.4

URXUCR = 0.3
level C:
URXMOH = 0.6
The ‘sex‘ variable was also simulated to allow to adjust for a covariate in the model. This dataset can thus be used to test the ‘gWQS‘ package by analyzing the mixed effect of the 59 simulated PCBs on the continuous, binary or count outcomes, with adjustments for covariates.

Usage

```
wqs_data
```

Format

A data frame with 500 rows and 68 variables

Details

y continuous outcome generated considerig all the predictors

yLBX continuous outcome generated considerig only PCBs

ybin binary outcome generated considerig all the predictors

ybinLBX binary outcome generated considerig only PCBs

ymultinom multinomial outcome generated considerig all the predictors

ymultinomLBX multinomial outcome generated considerig only PCBs

ycount count outcome generated considerig all the predictors

ycountLBX count outcome generated considerig only PCBs

sex covariate, gender of the subject

LBX 34 exposure concentrations of PCB

URX 25 exposure concentrations of phthalates ...

Index

∗ **datasets**
    wqs_data, 12

coef, *10*
coef.gwqs (summary.gwqs), 8
coef.gwqsrh (summary.gwqs), 8

fitted.gwqs (summary.gwqs), 8
fitted.gwqsrh (summary.gwqs), 8

ggplot, *7*
glm, *6*
glm.nb, *6*
gwqs, 2
gwqs_barplot, 6
gwqs_fitted_vs_resid (gwqs_barplot), 6
gwqs_levels_scatterplot (gwqs_barplot), 6
gwqs_rank (gwqs_barplot), 6
gwqs_ROC (gwqs_barplot), 6
gwqs_scatterplot (gwqs_barplot), 6
gwqs_summary_tab (gwqs_barplot), 6
gwqs_weights_tab (gwqs_barplot), 6
gwqsrh (gwqs), 2
gwqsrh_boxplot (gwqs_barplot), 6

kable, *8*
kable_styling, *8*

model.frame, *2*
multinom, *6*

optim, *3*, *4*

plan, *3*
predict, *11*
predict.gwqs (summary.gwqs), 8
predict.gwqsrh (summary.gwqs), 8
print.gwqs (summary.gwqs), 8
print.gwqsrh (summary.gwqs), 8
print.summary.gwqs (summary.gwqs), 8
print.summary.gwqsrh (summary.gwqs), 8

residuals, *11*
residuals.gwqs (summary.gwqs), 8
residuals.gwqsrh (summary.gwqs), 8

selectdatavars (gwqs_barplot), 6
summary.gwqs, 8
summary.gwqsrh (summary.gwqs), 8

vcov, *10*
vcov.gwqs (summary.gwqs), 8
vcov.gwqsrh (summary.gwqs), 8

wqs_data, 12

zeroinfl, *6*