Package: apm (via r-universe)

November 14, 2026

Title Averaged Prediction Models

Version 0.1.1

Description In panel data settings, specifies set of candidate models, fits them to data from pre-treatment validation periods, and selects model as average over candidate models, weighting each by posterior probability of being most robust given its differential average prediction errors in pre-treatment validation periods. Subsequent estimation and inference of causal effect's bounds accounts for both model and sampling uncertainty, and calculates the robustness changepoint value at which bounds go from excluding to including 0. The package also includes a range of diagnostic plots, such as those illustrating models' differential average prediction errors and the posterior distribution of which model is most robust.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports stats, ggplot2 (>= 3.5.1), ggh4x (>= 0.2.8), ggrepel (>= 0.9.6), MASS, sandwich, pbapply (>= 1.7-2), fwb (>= 0.3.0), chk (>= 0.10.0)

Suggests parallel, knitr, rmarkdown

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LazyData true

VignetteBuilder knitr

URL https://github.com/tl2624/apm/, https://tl2624.github.io/apm/

BugReports https://github.com/tl2624/apm/issues

Repository https://tl2624.r-universe.dev

Date/Publication 2025-05-22 18:42:00 UTC

RemoteUrl https://github.com/tl2624/apm

RemoteRef HEAD

RemoteSha de1f2027a2edae4346fb91a009e6c7cf5cb316bf

Contents

apm_est 2
apm_mod 5
apm_pre 7
plot.apm_pre_fits 10
ptpdata 12
robustness_bound 13

Index **15**

`apm_est` *Estimate ATTs from models fits*

Description

`apm_est()` computes the ATTs from the models previously fit by `apm_pre()`, choosing the optimal one by minimizing the largest absolute average prediction error across validation times. Optionally, this process can be simulated to arrive at a distribution of ATTs that accounts for the uncertainty in selecting the optimal model. `plot()` plots the resulting ATT(s).

Usage

```
apm_est(
  fits,
  post_time,
  M = 0,
  R = 1000L,
  all_models = FALSE,
  cl = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'apm_est'
summary(object, level = 0.95, M = NULL, ...)

## S3 method for class 'apm_est'
plot(x, label = TRUE, size.weights = TRUE, ...)
```

Arguments

`fits`	an `apm_pre_fits` object; the output of a call to `apm_pre()`.
`post_time`	the value of the time variable considered post-treatment, for which the ATT is to be estimated.

M	the sensitivity parameter for set identification. For apm_est(), the default is 0, i.e., under point identification. For summary(), this can be set to one or more positive values to produce uncertainty bounds for each value. Only allowed when not set to 0 in the call to apm_est(). See Details.
R	the number of bootstrap iterations used to compute the sampling variance of the ATT. Default is 1000. More is better but takes longer.
all_models	logical; whether to compute ATTs for all models (TRUE) or just those with BMA weights greater than 0 (FALSE, default). This will not effect the final estimates but leaving as FALSE can speed up computation when some models have BMA weights of 0.
cl	a cluster object created by parallel::makeCluster(), an integer to indicate number of child-processes (ignored on Windows) for parallel evaluations, or "future" to use a future backend. NULL (default) refers to sequential evaluation. See fwb::fwb() for details and issues related to replicability.
verbose	logical; whether to print information about the progress of the estimation, including a progress bar. Default is TRUE.
...	other arguments passed to fwb::fwb().
level	the desired confidence level. Set to 0 to ignore sampling variation in computing the interval bounds. Default is .95.
x, object	an apm_est object; the output of a call to apm_est().
label	logical; whether to label the ATT estimates. Default is TRUE.
size.weights	logical; whether to size the points based on their BMA weights. Default is TRUE.

Details

apm_est() estimates the ATT from each model and combines them to form the BMA-weighted estimate of the ATT. Uncertainty for the BMA-weighted ATT is computed by combining two variance components, one that account for sampling and another that accounts for model selection. The component due to sampling is computed by bootstrapping the process of fitting the outcome model for the post-treatment outcome identified by post_time and computing the difference between the observed outcome mean difference and the model-predicted outcome mean difference. The fractional weighted bootstrap as implemented in fwb::fwb() is used to ensure no units are dropped from the analysis. In each bootstrap sample, the BMA-weighted ATT estimate is computed as the weighted average of the ATTs computed from the models using the fixed BMA weights computed by apm_pre(), and the variance is computed as the empirical variance over the bootstrapped estimates. The variance component due to model selection is computed as the BMA-weighted variance of the original ATTs.

When M is greater than 0, bounds for set identification and their uncertainty are additionally computed. This involves bootstrapping the fitting of the pre-period models along with post-treatment models on order to compute the maximum absolute difference in average prediction errors for each model across validation periods. Each bootstrap sample produces a margin of error for each model computed as $M \times \delta_m$ where δ_m is the maximum absolute difference in average prediction errors for model m. Upper and lower bounds for the set-identified BMA-weighted ATT are computed as $\text{ATT}_m \pm M \times \delta_m$. The same procedure as above is then used to compute the variance of these bounds.

`summary()` displays the BMA-weighted ATT estimate, its standard error, and Wald confidence intervals. When `M` is greater than 0, bounds for the set-identified ATT are displayed in the confidence interval bound columns. The lower bound is computed as LB $- \sigma_{LB} Z_l$ and the upper bound as UB $+ \sigma_{UB} Z_l$, where LB and UB are the lower and upper bounds, σ_{LB} and σ_{UB} are their variances accounting for sampling and model selection, and Z_l is the critical Z-statistic for confidence level l. To display the set-identification bounds themselves, one should set `level = 0`.

Value

`apm_est()` returns an `apm_est` object, which contains the ATT estimates and their variance estimates. The following components are included:

BMA_att the BMA-weighted ATT

atts a 1-column matrix containing the ATT estimates from each model (when `all_models = FALSE`, only models with positive BMA weights are included)

BMA_var the total variance estimate for the BMA-weighted ATT incorporating the variance due to sampling and due to model selection

BMA_var_b the bootstrap-based component of the variance estimate for the BMA-weighted ATT due to sampling

BMA_var_m the component of the variance estimate for the BMA-weighted ATT due to model selection

M the value of the sensitivity parameter `M`

post_time the value supplied to `post_time`

observed_means a matrix of the observed outcome means at each pre-treatment validation period

pred_errors an array containing the average prediction errors for each model and each pre-treatment validation period

pred_error_diffs a matrix containing the difference in average prediction errors between groups for each model and each pre-treatment validation period

BMA_weights the BMA weights computed by `apm_pre()` (when `all_models = FALSE`, only positive BMA weights are included)

boot_out an `fwb` object containing the bootstrap results

`plot()` returns a `ggplot` object displaying the ATT for each model plotted against the maximum absolute difference in average prediction errors for that model. The model with the lowest maximum absolute difference in average prediction errors is displayed in red.

`summary()` produces a table with the BMA-weighted ATT, it's estimated standard error, and confidence interval limits. When `M` is greater than 0, additional rows for each value of `M` are included with the lower and upper bound. When `level` is greater than 0, these bounds include the uncertainty due to sampling and model selection; otherwise, they correspond to the set identification bounds for the ATT.

See Also

`apm_pre()` for computing the BMA weights; `fwb::fwb()` for the fractional weighted bootstrap.

Examples

```
data("ptpdata")

# Combination of 4 models: 2 time trends, 2 lags
models <- apm_mod(list(crude_rate ~ 1),
                  lag = 0:1,
                  time_trend = 0:1)
models

# Fit the models to data; unit_var must be supplied for
# fixed effects
fits <- apm_pre(models,
                data = ptpdata,
                group_var = "group",
                time_var = "year",
                val_times = 2004:2007,
                unit_var = "state",
                nsim = 100,
                verbose = FALSE)

est <- apm_est(fits,
               post_time = 2008,
               M = 1,
               R = 20,
               verbose = FALSE)

est

# ATT estimate and bounds for M = 1
summary(est)

# Bounds for other values of M
summary(est, M = c(.5, 1, 1.5, 2))

# Set-ID bounds without uncertainty
summary(est, level = 0)

plot(est)
```

---

apm_mod *Generate models used to fit outcomes*

Description

apm_mod() generates a list of models characterized by a basic model formulas and other options (e.g., lags, families, etc.) that are supplied to apm_pre(). These values are completely crossed to create a grid of model specifications, and multiple sets of model specifications can be combined using c() (see Examples).

Usage

```
apm_mod(
  formula_list,
  family = "gaussian",
  lag = 0L,
  diff_k = 0L,
  log = FALSE,
  time_trend = 0L,
  fixef = FALSE,
  identiy_only_log = TRUE
)
```

Arguments

formula_list	a list of model formulas with the outcome on the left side and predictions (or just an intercept) on the right side.
family	a list of family specifications; see family() for allowable options. These will eventually be passed to glm() when fitting the models in apm_pre(). "negbin" can also be supplied to request a negative binomial model with a log link fit using MASS::glm.nb(). Default is "gaussian" to specify a linear model.
lag	a vector of integers indicating the desired outcome lags to be used as predictors. For example, a lag value of 3 means the outcome lagged once, twice, and three times will be included as predictors. Default is 0 for no lags.
diff_k	a vector of integers indicating the desired outcome lag to be used a an offset For example, a diff_k value of 1 means the prior time point's outcome will be included as an offset, equivalent to using the outcome minus its corresponding lag as the outcome of the corresponding model. Default is 0 for no lags. Any models with a diff_k value less than a lag value will be removed automatically. When used with a family with a log link, the lags are automatically log-transformed; an error will be thrown by apm_pre() if nonpositive values are present in the outcome.
log	a logical vector indicating whether the outcome should be log-transformed. Default is FALSE to use the original outcome. When lag or diff_k are greater than 0, the outcome lags will also be log-transformed if TRUE. When the family has a log link and diff_k is greater than zero, the lag in the offset will be log transformed.
time_trend	a vector of integers indicating the desired powers to be included in a time trend. For example, a time_trend value of 2 means the time variable and its square will be included as predictors in the model. A value of 0 (the default) means time is not included as a predictor.
fixef	a logical vector indicating whether unit fixed effects should be included as predictors. Default is FALSE to omit unit fixed effects.
identiy_only_log	logical; whether to omit any models in which log is TRUE but the link in the family specification corresponds to something other than "identity". Default is TRUE, and this should probably not be changed.

Value

An apm_models object, which is a list containing the full cross (less any omitted combinations) of the model features specified in the arguments, with each combination a list. These have a print() method and can be combined using c(). Each model is named automatically, but these can be set manually using names() as well. Models can be removed by setting their value to NULL; see Examples.

See Also

formula(), family()

Examples

```
data("ptpdata")

# Combination of 8 models: 1 baseline formulas,
# 2 families, 2 lags, 2 time trends
models1 <- apm_mod(crude_rate ~ 1,
                   family = list("gaussian", "quasipoisson"),
                   time_trend = 0:1,
                   lag = 0:1, fixef = TRUE)
models1

# Add a single other model with a square time trend
models2 <- apm_mod(crude_rate ~ 1,
                   family = "gaussian",
                   time_trend = 2,
                   fixef = FALSE)
models2

(models <- c(models1, models2))

# Remove a model
models[[4]] <- NULL
models
```

---

apm_pre *Fit validation models to pre-treatment data*

Description

apm_pre() fits models to the pre-treatment data to compute the observed prediction errors for each model in each period and compute the Bayesian model averaging (BMA) weights eventually used in apm_est() to estimate the treatment effect.

Usage

```
apm_pre(
  models,
  data,
  weights = NULL,
  group_var,
  time_var,
  val_times,
  unit_var,
  nsim = 1000L,
  cl = NULL,
  verbose = TRUE
)

## S3 method for class 'apm_pre_fits'
summary(object, order = NULL, ...)
```

Arguments

models	an apm_models object; the output of a call to apm_mod().
data	a dataset containing all the variables named in the supplied models (i.e., the outcome and any predictors) as well as any variable named below.
weights	an optional vector of weights (e.g., sampling weights) used to fit weighted regression models.
group_var	string; the name of the treatment variable in data defining the "to be treated" and "not to be treated" groups. The corresponding variable should take on values of 0 and 1 only.
time_var	string; the name of the variable in data containing the time variable.
val_times	a numeric vector corresponding to the pre-treatment times that will be used as validation times when select the model with the optimal average expected prediction error.
unit_var	string; the name of the unit ID variable in data.
nsim	the number of simulated draws from the joint posterior of the fitted models to use to compute the BMA weights. Default is 1000. More is better but takes longer.
cl	a cluster object created by parallel::makeCluster(), or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations. It can also be "future" to use a future backend. NULL (default) refers to sequential evaluation. See the cl argument of pbapply::pblapply() for details.
verbose	logical; whether to print information about the progress of the estimation, including a progress bar. Default is TRUE.
object	an apm_pre_fit object; the output of a call to apm_pre().
order	how to order the summary; NULL (the default) is the same ordering as the models supplied to apm_pre(), "weights" orders the models by their computed BMA

weights with the highest weights on top, and "errors" orders the models by their maximum absolute difference in prediction errors with the smallest errors on top.

... ignored.

Details

apm_pre() creates a grid of all models and all time points and fits all corresponding models. For each validation time supplied to val_times, each model is fit using all previous times. For example, for a validation time of 5, a model is fit with data only from periods 1-4.

lm(), glm(), or MASS::glm.nb() are used to fit the given models. The joint covariance matrix of all the coefficients is computed using the SUEST method described in Mize et al. (2019, p164), which is also used by the STATA command suest. This is equivalent to the covariance matrix computed by stacking the score equations for the models and fitting them using M-estimation and yields the equivalent of the HC0 covariance matrix for all within-model covariances. The covariance is clustered by unit_id.

To compute the BMA weights, random variate drawn from a multivariate normal distribution nsim times with mean vector equal to the concatenated model coefficients and covariance equal to the joint covariance matrix described above. For each iteration, the absolute average prediction errors are calculated for each model and validation period. A model is considered the "winner" if it its largest absolute average prediction error across validation periods is the smallest among all models. The BMA weight for each model is equal to the proportion of iterations in which that model was the "winner".

Value

apm_pre() returns an apm_pre_fits object, which is a list containing the models supplied to models, a grid of all fitted models, a list of all model fit objects, a list of all estimated coefficients, the joint covariance of the coefficients, the dataset supplied to data, and other components supplied to apm_pre().

summary() produces a data frame containing the BMA weights and maximum absolute difference in mean prediction errors for each model, ordered according order. An asterisk appears next to the model with the smallest error.

See Also

lm(),glm(), and MASS::glm.nb() for the functions used to fit the models; apm_est() to compute the ATT and its uncertainty; plot.apm_pre_fits() for plotting an apm_pre_fits object.

Examples

```
data("ptpdata")

# Combination of 8 models: 2 baseline formulas,
# 2 families, 2 lags
models <- apm_mod(crude_rate ~ 1,
                  family = list("gaussian", "quasipoisson"),
                  time_trend = 0:1,
                  lag = 0:1)
```

```
models

# Fit the models to data
fits <- apm_pre(models, data = ptpdata,
                group_var = "group",
                time_var = "year",
                val_times = 1999:2007,
                unit_var = "state",
                nsim = 200,
                verbose = FALSE)

fits

summary(fits)

plot(fits, type = "weights")

plot(fits, type = "error")
```

`plot.apm_pre_fits` *Plot outputs of* `apm_pre()`

Description

`plot()` displays the Bayesian model averaging (BMA) weights for each model (computed by `apm_fit()` as the posterior probability of selection) and the distribution of the difference in average prediction errors.

Usage

```
## S3 method for class 'apm_pre_fits'
plot(
  x,
  type = "weights",
  abs = TRUE,
  ncol = 4L,
  clip_at = 15,
  model = ".optimal",
  ...
)
```

Arguments

`x`	an `apm_pre_fits` object; the output of a call to `apm_pre()`.
`type`	which values to plot: allowable options include `"weights"` to plot the BMA weights/posterior probabilities (default), `"errors"` to plot the difference in average predictions errors for all models across validation periods, `"predict"` to plot the time series and model predictions for each model, and `"corrected"` to

plot the corrected predictions for the treated group for each model. Abbreviations allowed.

`abs` — `logical`; when `type = "errors"`, whether to plot the differences in average prediction errors in absolute value (`TRUE`, default) or not (`FALSE`).

`ncol` — when `type` is `"errors"`, `"predict"`, or `"corrected"`, the number of columns to use to display the plots. Default is 4.

`clip_at` — when `type = "errors"`, the value (in robust z-score units) at which to clip the y-axis of the plot to prevent outliers from distorting it. Default is 15. Set to `Inf` to prevent clipping.

`model` — string; when `type = "predict"` or `type = "corrected"`, the model(s) to plot. Allowable values include `".optimal"` to plot the model with the smallest maximum absolute difference in average prediction errors, `".all"` to plot all models (excluding the BMA-weighted predictions), or the names of one or more specific models. Abbreviations allowed.

`...` — ignored.

Details

When `type = "weights"`, `plot()` displays a bar plot with a bar for each model with height equal to the BMA weight/posterior probability of selection for that model. (Note that the plot margins can sometimes cut off the model names; use `theme(plot.margins =)` after loading `ggplot2` to extend the left margin of the plot to ensure all text is visible. Alternatively, the axis text can be rotated using `theme(axis.text.x =)`.)

When `type = "errors"`, `plot()` displays a lattice of bar plots with a plot for each model displaying the difference in average prediction errors for each validation period. The period with the largest difference in average prediction errors will be shaded black. The model with the smallest maximum absolute difference in average prediction errors will have a gray label.

When `type = "predict"`, `plot()` displays a lattice of line plots with a plot for each model displaying the observed and predicted outcomes for each validation period under each model. The observed outcomes are displayed as points, while the predicted outcomes are displayed as lines.

When `type = "corrected"`, `plot()` displays a lattice of line plots with a plot for each model displaying the observed and corrected predictions for the treated group for each validation period under each model. The observed outcomes are displayed as points, while the corrected predictions are displayed as lines. Corrected predictions are computed as the observed outcome in the treated group minus the prediction error in the treated group plus the prediction error in the control group.

Value

A `ggplot` object, which can be manipulated using `ggplot2` syntax (after loading `ggplot2`).

See Also

`apm_pre()` to to compute the difference in average prediction errors and BMA weights; `ggplot2::geom_col()`, which is used to create the plots.

Examples

```
data("ptpdata")

# Combination of 8 models: 2 baseline formulas,
# 2 families, 2 lags
models <- apm_mod(crude_rate ~ 1,
                  family = "gaussian",
                  time_trend = 0:1,
                  lag = 0:1,
                  diff_k = 0:1)
models

# Fit the models to data
fits <- apm_pre(models, data = ptpdata,
                group_var = "group",
                time_var = "year",
                val_times = 1999:2007,
                unit_var = "state",
                nsim = 50,
                verbose = FALSE)
fits

plot(fits, type = "weights")

plot(fits, type = "error", ncol = 2)

plot(fits, type = "predict", model = ".optimal")

plot(fits, type = "corrected", model = ".optimal")
```

---

ptpdata *Dataset on Annual Homicide Rates*

Description

A dataset of homicide rates across 9 states from 1994 to 2008. Missouri repealed its permit-to-purchase (PTP) law in 2007.

Usage

```
ptpdata
```

Format

A dataframe of 7 variables with 135 observations (9 states, 15 years).

Details

details for dataset2

state the name of the state; states present include Arkansas, Illinois, Iowa, Kansas, Kentucky, Missouri, Nebraska, Oklahoma, and Tennessee.

year the year of each observation. Years range from 1994 to 2008.

deaths the number of homicide deaths in the corresponding state in the corresponding year.

crude_rate the homicide rate in the corresponding state in the corresponding year.

age_adj_rate the age-adjusted homicide rate in the corresponding state in the corresponding year.

group whether each observation belongs to the "to-be-treated" group; 1 for Missouri and 0 for all other states.

treat whether each observation is treated; 1 for Missouri in 2008 and 0 otherwise.

---

robustness_bound *Compute the robustness changepoint*

Description

`robustness_bound()` computes the value of the sensitivity parameter M at which the robustness bounds change from excluding to including an ATT of 0.

Usage

```
robustness_bound(object, level = 0.95)
```

Arguments

`object`	an `apm_est` object; the output of a call to `apm_est()`. `M` must have been set to a nonzero value to use `robustness_bound()`.
`level`	the desired confidence level. Set to 0 to ignore sampling variation in computing the interval bounds. Default is .95.

Value

A single number corresponding to the changepoint value of M. If there is no positive value of M for which the interval bounds cross 0, `NA` will be returned.

See Also

`summary.apm_est()` for examining the ATT and bounds for a given value of `M`; `uniroot()` for the function that finds the changepoint value of `M`.

Examples

```
data("ptpdata")

# Combination of 4 models: 2 time trends, 2 lags
models <- apm_mod(list(crude_rate ~ 1),
                  lag = 0:1,
                  time_trend = 0:1)
models

# Fit the models to data; unit_var must be supplied for
# fixed effects
fits <- apm_pre(models,
                data = ptpdata,
                group_var = "group",
                time_var = "year",
                val_times = 2004:2007,
                unit_var = "state",
                nsim = 100,
                verbose = FALSE)

est <- apm_est(fits,
               post_time = 2008,
               M = 1,
               R = 20,
               verbose = FALSE)

est

# ATT estimate and bounds for M = 1
summary(est)

#Changepoint value of M ignoring estimation uncertainty
(M <- robustness_bound(est, level = 0))

summary(est, level = 0, M = M)

#Changepoint value of M accounting for estimation uncertainty
(M <- robustness_bound(est, level = .95))

summary(est, level = .95, M = M)
```

Index

∗ **datasets**
ptpdata, 12

apm_est, 2
apm_est(), *7*, *9*, *13*
apm_mod, 5
apm_mod(), *8*
apm_pre, 7
apm_pre(), *2–6*, *10, 11*

family(), *6, 7*
formula(), *7*
fwb::fwb(), *3, 4*

ggplot2::geom_col(), *11*
glm(), *6*, *9*

lm(), *9*

MASS::glm.nb(), *6*, *9*

names(), *7*

parallel::makeCluster(), *3*, *8*
pbapply::pblapply(), *8*
plot.apm_est *(*apm_est*)*, 2
plot.apm_pre_fits, 10
plot.apm_pre_fits(), *9*
ptpdata, 12

robustness_bound, 13

summary.apm_est *(*apm_est*)*, 2
summary.apm_est(), *13*
summary.apm_pre_fits *(*apm_pre*)*, 7

uniroot(), *13*