vignette

Introduction to causalglm

causalglm is an R package for robust generalized linear models and interpretable causal inference for heterogeneous (or conditional) treatment effects. Specifically, causalglm very significantly relaxes the assumptions needed for useful causal estimates and correct inference by employing semi and nonparametric models and adaptive machine-learning through targeted maximum likelihood estimation (TMLE). See the writeup causalglm.pdf for a more theoretical overview of the methods implemented in this package.

The statistical data-structure used throughout this package is $O = (W, A, Y)$ where W represents a random vector of baseline (pretreatment) covariates/confounders, A is a usually binary treatment assignment with values in $c(0, 1)$, and Y is some outcome variable. For marginal structural models, we also consider a subvector $V \subset W$ that represents a subset of baseline variables that are of interest.

The estimands supported by causalglm are

1. Conditional average treatment effect (CATE) for arbitrary outcomes: $E[Y|A = 1, W] - E[Y|A = 0, W]$
2. Conditional odds ratio (OR) for binary outcomes: $\frac{P(Y=1|A=1,W)/P(Y=0|A=1,W)}{P(Y=1|A=0,W)/P(Y=0|A=0,W)}$
3. Conditional relative risk (RR) for binary, count or nonnegative outcomes: E[Y|A=1,W]/E[Y|A=0,W]
4. Conditional treatment-specific mean (TSM) : $E[Y|A = a, W$
5. Conditional average treatment effect among the treated (CATT) : the best approximation of E[Y|A=1,W] - E[Y|A=0,W] based on a user-specified formula/parametric model among the treated (i.e. observations with $A = 1$)

causalglm also supports the following marginal structural model estimands:

1. Marginal structural models for the CATE: E[CATE(W)|V] := E[E[Y|A=1,W] - E[Y|A=0,W]|V]
2. Marginal structural models for the RR: E[E[Y|A=1,W]|V]/E[E[Y|A=0,W]|V]
3. Marginal structural models for the TSM : $E[E[Y|A = a, W]|V]$
4. Marginal structural models for the CATT : $E[CATE(W)|V, A = 1] := E[E[Y|A = 1, W] - E[Y|A = 0, W]|V, A = 1]$

causalglm consists of four main functions:

1. spglm for semiparametric estimation of correctly specified parametric models for the CATE, RR and OR
2. npglm for robust nonparametric estimation for user-specified approximation models for the CATE, CATT, TSM, RR or OR
3. msmglm for robust nonparametric estimation for user-specified marginal structural models for the CATE, CATT, TSM or RR
4. causalglmnet for high dimensional confounders W (a custom wrapper function for spglm focused on big data where standard ML may struggle)

spglm is a semiparametric method which means that it assumes the user-specified parametric model is correct for inference. This method should be used if you are very confident in your parametric model. npglm is a nonparametric method that views the user-specified parametric model as an approximation or working-model for the true nonparametric estimand. The estimands are the best causal approximation of the true conditional estimand (i.e. projections). Because of this model agnostic view, npglm provides interpretable estimates and correct inference under no conditions. The user-specified parametric model need not be correct or even a good approximation for inference! npglm should be used if you believe your parametric model is a good approximation but are not very confident that it is correct. Also, it never hurts to run both spglm and npglm for robustness! If the parametric model is close to correct then the two methods should give similar estimates. Finally, msmglm deals with marginal structural models for the conditional treatment effect estimands. This method is useful if you are only interested in modeling the causal treatment effect as a function of a subset of variables V adjusting for all the available confounders W that remain. This allows for parsimonious causal modeling, still maximally adjusting for confounding. This function can be used to understand the causal variable importance of individual variables (by having V be a single variable) and allows for nice plots (see plot_msm).

Overview of features using `estimand = "CATE"` as an example

We will begin with the conditional average treatment effect estimand (CATE) and use it to illustrate the features of causalglm. Afterwards, we will go through all the other available estimands.

We will use the following simulated data throughout this part.

```
n <- 250
W1 <- runif(n, min = -1, max = 1)
W2 <- runif(n, min = -1, max = 1)
A <- rbinom(n, size = 1, prob = plogis((W1 + W2  )/3))
Y <- rnorm(n, mean = A * (1 + W1 + 2*W1^2) + sin(4 * W2) + sin(4 * W1), sd = 0.3)
data <- data.frame(W1, W2,A,Y)
```

spglm with CATE

All methods in causalglm have a similar argument setup. Mainly, they require a formula that specifies a parametric form for the conditional estimand, a data.frame with the data, and character vectors containing the names of the variables W, A and Y. The estimand is specified with the argument *estimand* and the learning method is specified with the *learning_method* argument.

```
formula <- ~ poly(W1, degree = 2, raw = T) # A correctly specified polynomial model of degree 2
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE", # Options are CATE, RR, OR
      learning_method = "HAL" # A bunch of options. Default is a custom semiparametric Highly Adaptive 
      )
```

```
## (max) epsilon: -9.836397e-04 max(abs(ED)): 4.563884e-17
```

`output` contains a `spglm` fit object. It contains estimates information and tlverse/tmle3 objects that store the fit likelihood, tmle_tasks, and target parameter objects. There are a number of extractor functions that should suffice for almost everyone. The `summary`, `coefs`, `print` and `predict` functions should be useful. They work as follows.

```
# Print tells you the object, estimand, and a fit formula/equation for the estimand
print(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.08 * (Intercept) + 1.06 * poly(W1, degree = 2, raw = T)1 + 1.92 * poly(W1, degree = 2, r
```

Summary provides the coefficient estimates (tmle_est), 95% confidence intervals (lower, upper), and p-values (p_value). The coef function provides pretty much the same thing as summary.

```r
summary(output)  # Summary gives you the estimates and inference
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.08 * (Intercept) + 1.06 * poly(W1, degree = 2, raw = T)1 + 1.92 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                          param tmle_est         se     lower    upper
## 1: CATE                    (Intercept) 1.076609 0.05023407 0.9781525 1.175066
## 2: CATE poly(W1, degree = 2, raw = T)1 1.055423 0.05658697 0.9445145 1.166331
## 3: CATE poly(W1, degree = 2, raw = T)2 1.918424 0.10683868 1.7090245 2.127824
##     Z_score p_value
## 1: 338.8675       0
## 2: 294.9036       0
## 3: 283.9136       0
```

The predict function allows you get individual-level treatment effect predictions and 95% prediction (confidence) intervals. Specifically, for each observation, the individual CATE estimate derived from the coefficient estimates is given and a 95% confidence interval + p-values for it.

```r
preds <- predict(output, data = data)
preds <- predict(output) # By default, training data is used.
head(preds)
```

```
##   (Intercept) poly(W1, degree = 2, raw = T)1 poly(W1, degree = 2, raw = T)2
## 1           1                      0.7040014                     0.49561794
## 2           1                      0.7065098                     0.49915605
## 3           1                      0.8113231                     0.65824519
## 4           1                     -0.7649594                     0.58516284
## 5           1                      0.2100699                     0.04412935
## 6           1                      0.7099265                     0.50399559
##    CATE(W)        se  CI_left CI_right  Z-score p-value
## 1 2.770434 0.7949916 2.671886 2.868982 55.10047       0
## 2 2.779869 0.7983337 2.680907 2.878832 55.05667       0
## 3 3.195692 0.9752680 3.074797 3.316588 51.80969       0
## 4 1.391845 1.0087457 1.266799 1.516890 21.81620       0
## 5 1.382981 0.7456640 1.290547 1.475414 29.32534       0
## 6 2.792760 0.8029552 2.693224 2.892295 54.99361       0
```

It is common to want to obtain multiple fits using multiple formulas. We recommend doing this with npglm since it always provides correct interpretable inference even when these models are wrong. It is computationally expensive to recall spglm for each formula since the machine-learning is redone. Instead, we can reuse the machine-learning fits from previous calls to spglm. Due to the semiparametric nature of spglm, the way this works for spglm differs from npglm and msmglm. For spglm, you can pass a previous spglm fit object through the `data` argument with a new formula. The previous fits will then automatically be reused. The catch for spglm is that the new formula must be a subset of the original formula from the previous fit. Thus, one should first fit the most complex formula that contains all terms of interest and then call spglm with the desired subformulas. Lets see how this works. Fortunately, npglm and msmglm also allow for reusing fits and they even work across estimands and for arbitrary formulas (not just subformulas).

```r
# Start with big formula
formula_full <- ~ poly(W1, degree = 3, raw = T)
output_full <- spglm(formula_full,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
```

```
      )
```

```
## (max) epsilon: -1.117314e-03 max(abs(ED)): 2.973594e-16
```

```
summary(output_full)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.08 * (Intercept) + 0.972 * poly(W1, degree = 3, raw = T)1 + 1.92 * poly(W1, degree = 3, r
##
## Coefficient estimates and inference:
##    type                          param  tmle_est          se      lower
## 1: CATE                    (Intercept) 1.0786759 0.05154435  0.9776508
## 2: CATE poly(W1, degree = 3, raw = T)1 0.9721665 0.15479626  0.6687714
## 3: CATE poly(W1, degree = 3, raw = T)2 1.9153390 0.10953202  1.7006602
## 4: CATE poly(W1, degree = 3, raw = T)3 0.1583614 0.23129669 -0.2949718
##        upper   Z_score p_value
## 1: 1.1797009 330.88717       0
## 2: 1.2755616  99.30021       0
## 3: 2.1300178 276.48689       0
## 4: 0.6116946  10.82555       0
```

```
# This will give a warning since the term names for `poly(W1, degree = 2, raw = T)` are not a subset of
# Use argument warn = FALSE to turn this off.
subformula <- ~ poly(W1, degree = 2, raw = T)  # one less degree
output<- spglm(subformula,
      data = output_full, # replace data with output_full
      estimand = "CATE" # No need to specify the variables again.
      )
```

```
## Warning in spglm(subformula, data = output_full, estimand = "CATE"): Terms of
## new formula could not be confirmed as subsets of original formula. Make sure
## this formula is truly a subformula or else the results may be unreliable..
```

```
## (max) epsilon: -1.085943e-03 max(abs(ED)): 7.184791e-17
```

```
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.08 * (Intercept) + 1.07 * poly(W1, degree = 2, raw = T)1 + 1.92 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                          param tmle_est          se     lower    upper
## 1: CATE                    (Intercept)  1.07548 0.05134515 0.9748456 1.176115
## 2: CATE poly(W1, degree = 2, raw = T)1  1.06708 0.05702840 0.9553067 1.178854
## 3: CATE poly(W1, degree = 2, raw = T)2  1.91969 0.10969591 1.7046901 2.134690
##     Z_score p_value
## 1: 331.1868       0
## 2: 295.8530       0
## 3: 276.7010       0
```

```
subformula <- ~ 1 + W1  # one less degree
output<- spglm(subformula,
      data = output_full, # replace data with output_full
      estimand = "CATE", warn = FALSE # No need to specify the variables again.
      )
```

```
## (max) epsilon: 8.178108e-04 max(abs(ED)): 5.246931e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.71 * (Intercept) + 1.05 * W1
##
## Coefficient estimates and inference:
##    type       param tmle_est         se     lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.706045 0.03381380 1.6397710 1.772319 797.7493       0
## 2: CATE          W1 1.045752 0.05703972 0.9339559 1.157547 289.8820       0
```

```r
subformula <- ~ 1  # one less degree
output<- spglm(subformula,
      data = output_full, # replace data with output_full
      estimand = "CATE", warn = FALSE # No need to specify the variables again.
      )
```

```
## (max) epsilon: 1.540166e-04 max(abs(ED)): 1.589354e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.74 * (Intercept)
##
## Coefficient estimates and inference:
##    type       param tmle_est         se    lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.735561 0.03390629 1.669106 1.802016 809.3373       0
```

```r
# That was fast! Look how different the estimates are when the model is misspecified! (npglm would do b
```

Currently all learning was done with HAL (default and recommended in most cases). There are a number of other options. All methods in this package require machine-learning of $P(A=1|W)$ (the propensity score) and $E[Y|A,W]$ (the conditinal mean outcome). For spglm, $E[Y|A,W]$ is learned in a semiparametric way. By default, the learning algorithm is provided the design matrix $cbind(W, A \cdot formula(W))$ where W is a matrix with columns being the baseline variable observations and $A \cdot formula(W)$ is a matrix with columns being the treatment interaction observations specified by the formula argument. Specifically, the design matrix is constructed as follows:

```r
formula <- ~ 1 + W1
AW <- model.matrix(formula, data)
design_mat_sp_Y <- as.matrix(cbind(data[,c("W1", "W2")],AW))
head(as.data.frame(design_mat_sp_Y))
```

```
##           W1         W2 (Intercept)         W1
## 1  0.7040014  0.9345973           1  0.7040014
## 2  0.7065098 -0.1191676           1  0.7065098
## 3  0.8113231 -0.4146595           1  0.8113231
## 4 -0.7649594  0.7453386           1 -0.7649594
## 5  0.2100699  0.7558070           1  0.2100699
## 6  0.7099265  0.2945271           1  0.7099265
```

Since the design matrix automatically contains the treatment interaction terms, additive learners like glm, glmnet or gam can in principle perform well (since they will model treatment interactions). Note that the final regression fit based on this design matrix will be projected onto the semiparametric model using glm.fit to ensure all model constraints are satisfied (this is not important and happens behind the scenes).

This learning method corresponds with the default argument specification `append\_design\_matrix = TRUE`. The other option `append\_design\_matrix = FALSE` performs treatment-stratified estimation. Specifically, the machine-learning algorithm is used to learn the placebo conditional mean $E[Y|A=0,W]$ by performing the regression of Y on W using only the observations with $A = 0$. Next, this initial estimator of $E[Y|A = 0, W]$ is used as an offset in a glm-type regression of Y on $A \cdot formula(W)$. This two-stage approach does not pool data across treatment arms and is thus not preferred.

Now that we got the nitty and gritty details out of the way. Lets use some different algorithms. We see that glm and glmnet perform very badly because of model misspecification. (The true model is quite nonlinear in the noninteraction terms). This motivates using causalglm over conventional methods like glm.

```
formula <- ~ poly(W1, degree = 2, raw = T)
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "glm"
      )
```

```
## (max) epsilon: 2.379635e-02 max(abs(ED)): 1.173159e-16
```

```
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.06 * (Intercept) + 0.922 * poly(W1, degree = 2, raw = T)1 + 1.9 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                          param  tmle_est        se     lower    upper
## 1: CATE                    (Intercept) 1.0617760 0.1731852 0.7223392 1.401213
## 2: CATE poly(W1, degree = 2, raw = T)1 0.9223265 0.2077679 0.5151089 1.329544
## 3: CATE poly(W1, degree = 2, raw = T)2 1.8977910 0.4097479 1.0947000 2.700882
##     Z_score p_value
## 1: 96.93757       0
## 2: 70.19017       0
## 3: 73.23214       0
```

```
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "glmnet"
      )
```

```
## (max) epsilon: 1.215192e-02 max(abs(ED)): 4.771461e-16
```

```
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.07 * (Intercept) + 0.921 * poly(W1, degree = 2, raw = T)1 + 1.87 * poly(W1, degree = 2, r
##
## Coefficient estimates and inference:
##    type                          param  tmle_est        se     lower    upper
## 1: CATE                    (Intercept) 1.0699116 0.1731912 0.7304631 1.409360
## 2: CATE poly(W1, degree = 2, raw = T)1 0.9205476 0.2070159 0.5148039 1.326291
## 3: CATE poly(W1, degree = 2, raw = T)2 1.8736160 0.4083681 1.0732292 2.674003
##     Z_score p_value
## 1: 97.67695       0
## 2: 70.30927       0
## 3: 72.54354       0
```

```r
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "gam"
      )
```

```
## (max) epsilon: 7.003408e-04 max(abs(ED)): 3.806330e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.08 * (Intercept) + 1.05 * poly(W1, degree = 2, raw = T)1 + 1.9 * poly(W1, degree = 2, raw
##
## Coefficient estimates and inference:
##    type                            param tmle_est         se     lower    upper
## 1: CATE                      (Intercept) 1.077296 0.05339449 0.9726448 1.181947
## 2: CATE poly(W1, degree = 2, raw = T)1 1.048636 0.06104930 0.9289817 1.168291
## 3: CATE poly(W1, degree = 2, raw = T)2 1.895596 0.12044572 1.6595270 2.131666
##     Z_score p_value
## 1: 319.0132       0
## 2: 271.5902       0
## 3: 248.8425       0
```

```r
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "mars"
      )
```

```
## (max) epsilon: 1.647732e-02 max(abs(ED)): 1.295838e-16
```

```r
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.05 * (Intercept) + 1.04 * poly(W1, degree = 2, raw = T)1 + 1.92 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                            param tmle_est        se     lower    upper
## 1: CATE                      (Intercept) 1.052713 0.0606860 0.9337703 1.171655
## 2: CATE poly(W1, degree = 2, raw = T)1 1.040485 0.0695090 0.9042499 1.176720
## 3: CATE poly(W1, degree = 2, raw = T)2 1.917856 0.1396787 1.6440907 2.191621
##     Z_score p_value
## 1: 274.2782       0
## 2: 236.6818       0
## 3: 217.0980       0
```

```r
output <- spglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "xgboost"
      )
```

```
## (max) epsilon: 4.476031e-02 max(abs(ED)): 3.600592e-16
```

```r
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand CATE with formula:
## CATE(W) = 1.03 * (Intercept) + 1.11 * poly(W1, degree = 2, raw = T)1 + 1.98 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                            param tmle_est         se     lower    upper
## 1: CATE                      (Intercept) 1.033119 0.08697265 0.8626559 1.203582
## 2: CATE poly(W1, degree = 2, raw = T)1 1.113152 0.10461248 0.9081154 1.318189
## 3: CATE poly(W1, degree = 2, raw = T)2 1.976449 0.22234367 1.5406631 2.412234
##     Z_score p_value
## 1: 187.8182       0
## 2: 168.2445       0
## 3: 140.5500       0
```

npglm with CATE

npglm is a model-robust version of spglm that we personally recommend (at least as a robustness check). npglm works similarly to spglm. Fitting and extractor functions are pretty much the same.

```r
formula <- ~ poly(W1, degree = 2, raw = T)
output <- npglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
      )
```

```
## (max) epsilon: 8.475135e-03 max(abs(ED)): 6.259750e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.07 * (Intercept) + 1.05 * poly(W1, degree = 2, raw = T)1 + 1.88 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                            param tmle_est         se     lower    upper
## 1: CATE                      (Intercept) 1.068627 0.05150286 0.9676829 1.169570
## 2: CATE poly(W1, degree = 2, raw = T)1 1.047122 0.05765716 0.9341163 1.160128
## 3: CATE poly(W1, degree = 2, raw = T)2 1.875775 0.11836933 1.6437749 2.107774
##     Z_score p_value
## 1: 328.0686       0
## 2: 287.1535       0
## 3: 250.5598       0
```

```r
head(predict(output))
```

```
##   (Intercept) poly(W1, degree = 2, raw = T)1 poly(W1, degree = 2, raw = T)2
## 1           1                      0.7040014                     0.49561794
## 2           1                      0.7065098                     0.49915605
## 3           1                      0.8113231                     0.65824519
## 4           1                     -0.7649594                     0.58516284
## 5           1                      0.2100699                     0.04412935
## 6           1                      0.7099265                     0.50399559
##    CATE(W)        se  CI_left CI_right  Z-score p-value
## 1 2.735470 0.8874869 2.625456 2.845484 48.73489       0
```

```
## 2 2.744733 0.8917135 2.634195 2.855271 48.66814       0
## 3 3.152901 1.1077226 3.015586 3.290215 45.00381       0
## 4 1.365254 0.9818059 1.243548 1.486960 21.98659       0
## 5 1.371372 0.7668688 1.276310 1.466434 28.27511       0
## 6 2.757389 0.8975448 2.646128 2.868649 48.57489       0
```

npglm can reuse fits across both formulas and estimands with no restrictions. This is because the conditional mean and propensity score are learned fully nonparametrically (the previous semiparametric learning method no longer applies). The nice thing about npglm is that all models are viewed as approximations and thus each model below is interpretable as the best approximation. The intercept model is actually a nonparametric estimate for the marginal ATE! (See writeup.) Additionally, the inference for each model is correct (we don't require correctly specified parametric models!).

```
formula <- ~ 1 # We can start with simplest model. npglm does not care.
output_full <- npglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
      )
```

```
## (max) epsilon: 2.867249e-05 max(abs(ED)): 7.382983e-18
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.07 * (Intercept) + 1.05 * poly(W1, degree = 2, raw = T)1 + 1.88 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                          param tmle_est         se     lower    upper
## 1: CATE                    (Intercept) 1.068627 0.05150286 0.9676829 1.169570
## 2: CATE poly(W1, degree = 2, raw = T)1 1.047122 0.05765716 0.9341163 1.160128
## 3: CATE poly(W1, degree = 2, raw = T)2 1.875775 0.11836933 1.6437749 2.107774
##     Z_score p_value
## 1: 328.0686       0
## 2: 287.1535       0
## 3: 250.5598       0
```

```
formula <- ~  1 + W1
output <- npglm(formula,
      output_full,
      estimand = "CATE"
      )
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: -4.594136e-04 max(abs(ED)): 7.430168e-17
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.68 * (Intercept) + 0.969 * W1
##
## Coefficient estimates and inference:
##    type       param  tmle_est         se     lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.6838134 0.04820397 1.5893354 1.778291 552.3078       0
## 2: CATE          W1 0.9690411 0.09776292 0.7774293 1.160653 156.7249       0
```

```r
formula <- ~ poly(W1, degree = 2, raw = T)
output <- npglm(formula,
      output_full,
      estimand = "CATE"
      )
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: 4.691951e-03 max(abs(ED)): 2.018177e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.06 * (Intercept) + 1.04 * poly(W1, degree = 2, raw = T)1 + 1.89 * poly(W1, degree = 2, ra
##
## Coefficient estimates and inference:
##    type                          param tmle_est         se     lower    upper
## 1: CATE                    (Intercept) 1.058582 0.05074556 0.9591230 1.158042
## 2: CATE poly(W1, degree = 2, raw = T)1 1.035428 0.05677072 0.9241599 1.146697
## 3: CATE poly(W1, degree = 2, raw = T)2 1.892438 0.11683691 1.6634420 2.121434
##     Z_score p_value
## 1: 329.8349       0
## 2: 288.3804       0
## 3: 256.1012       0
```

```r
formula <- ~ poly(W1, degree = 3, raw = T)
output <- npglm(formula,
      output_full,
      estimand = "CATE"
      )
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: -1.068552e-02 max(abs(ED)): 2.560105e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.06 * (Intercept) + 0.908 * poly(W1, degree = 3, raw = T)1 + 1.89 * poly(W1, degree = 3, r
##
## Coefficient estimates and inference:
##    type                          param  tmle_est         se      lower
## 1: CATE                    (Intercept) 1.0610460 0.05111944  0.9608538
## 2: CATE poly(W1, degree = 3, raw = T)1 0.9083549 0.13564964  0.6424864
## 3: CATE poly(W1, degree = 3, raw = T)2 1.8881977 0.11658521  1.6596949
## 4: CATE poly(W1, degree = 3, raw = T)3 0.2152378 0.21634367 -0.2087880
##        upper   Z_score p_value
## 1: 1.1612383 328.18453       0
## 2: 1.1742233 105.87829       0
## 3: 2.1167005 256.07903       0
## 4: 0.6392636  15.73056       0
```

causalglmnet with CATE

causalglmnet is a wrapper for spglm that uses the LASSO with glmnet for all estimation. This is made for high dimensional settings. It is used in the same way as spglm.

```r
formula <- ~ poly(W1, degree = 3, raw = T)
output <- causalglmnet(formula,
```

```
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE"
      )
```

```
## (max) epsilon: 3.282416e+00 max(abs(ED)): 5.816153e-15
```

```
summary(output)
```

```
## A causalglm fit object obtained from causalglmnet for the estimand CATE with formula:
## CATE(W) = 1.13 * (Intercept) + 0.126 * poly(W1, degree = 3, raw = T)1 + 1.65 * poly(W1, degree = 3, r
##
## Coefficient estimates and inference:
##    type                          param  tmle_est        se        lower
## 1: CATE                    (Intercept) 1.1268623 0.1501016  0.832668589
## 2: CATE poly(W1, degree = 3, raw = T)1 0.1259956 0.4524698 -0.760828893
## 3: CATE poly(W1, degree = 3, raw = T)2 1.6543729 0.3456009  0.977007532
## 4: CATE poly(W1, degree = 3, raw = T)3 1.3913367 0.7123614 -0.004866021
##       upper   Z_score    p_value
## 1: 1.421056 118.70132 0.0000e+00
## 2: 1.012820   4.40287 1.0683e-05
## 3: 2.331738  75.68826 0.0000e+00
## 4: 2.787539  30.88175 0.0000e+00
```

msmglm with CATE

msmglm is for learning marginal structural models (e.g. marginal estimands like the ATE, ATT, and marginal relative risk). It operates in the same way as npglm. It is also a nonparametrically robust method that does not require correct model specification and estimates the best approximation. The only difference is that the marginal covariate(s) of interest V need to be specified. It also has a useful plotting feature that displays 95% confidence bands (only if V is one-dimensional). This method is used if you have many confounders W for which to adjust but only care about the treatment effect association with a subset of variables V. This can be used to build causal predictors that only utilize a handful of variables.

```
formula <-  ~ poly(W1, degree = 3, raw = T)
output <- msmglm(formula,
      data,
      V = "W1",
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
      )
```

```
## (max) epsilon: 4.129264e-02 max(abs(ED)): 4.631018e-16
```

```
summary(output)
```

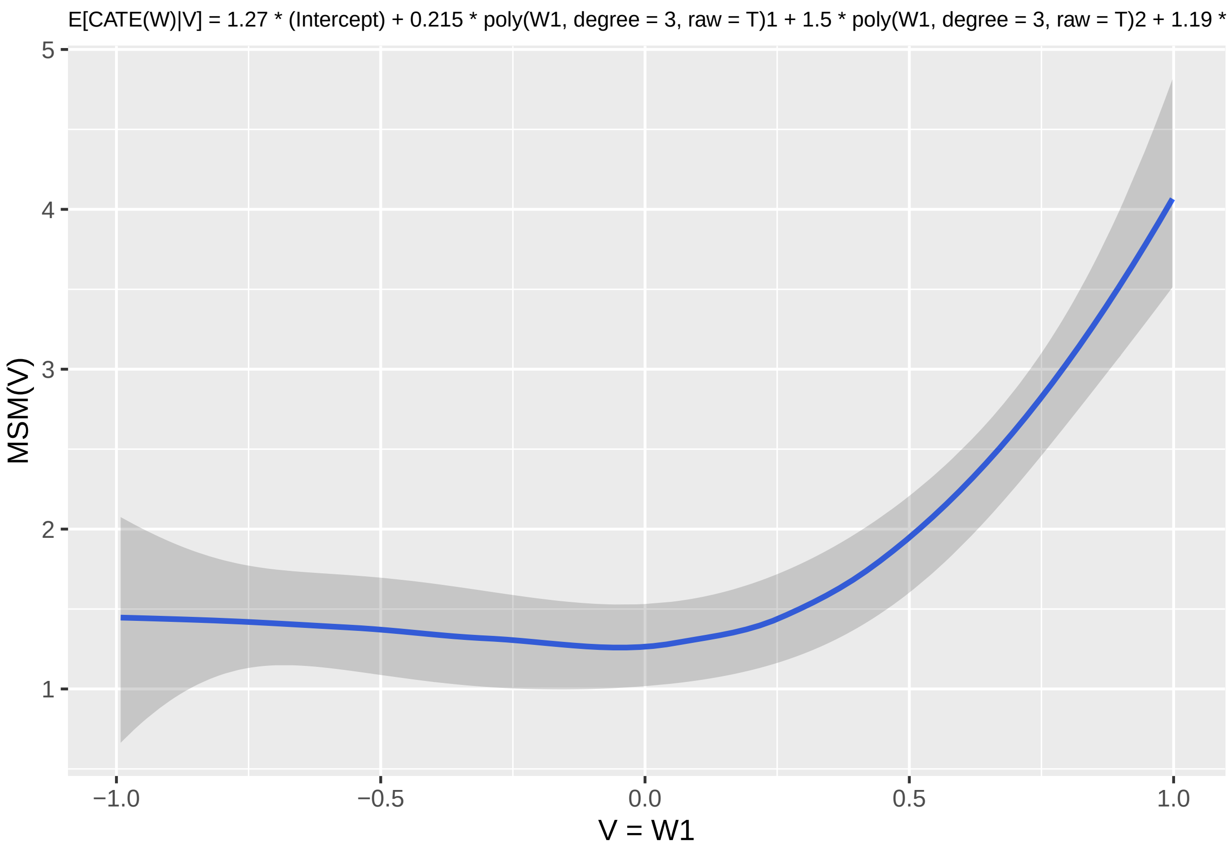
```
## A causalglm fit object obtained from msmglm for the estimand CATE with formula:
## E[CATE(W)|V] = 1.27 * (Intercept) + 0.215 * poly(W1, degree = 3, raw = T)1 + 1.5 * poly(W1, degree =
##
## Coefficient estimates and inference:
##    type                          param  tmle_est        se       lower
## 1: CATE                    (Intercept) 1.2743693 0.1311173  1.01738405
## 2: CATE poly(W1, degree = 3, raw = T)1 0.2153617 0.3715142 -0.51279283
## 3: CATE poly(W1, degree = 3, raw = T)2 1.4960846 0.2990497  0.90995799
## 4: CATE poly(W1, degree = 3, raw = T)3 1.1922956 0.5654499  0.08403418
```

```
##        upper    Z_score p_value
## 1: 1.5313545 153.675701       0
## 2: 0.9435162   9.165645       0
## 3: 2.0822112  79.101154       0
## 4: 2.3005571  33.339557       0
```

```r
plot_msm(output)
```

```
## Loading required package: ggplot2
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
formula <-  ~ 1 + W1 # Best linear approximation
output <- msmglm(formula,
      data,
      V = "W1",
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
      )
```

```
## (max) epsilon: -2.168065e-03 max(abs(ED)): 6.822320e-17
```

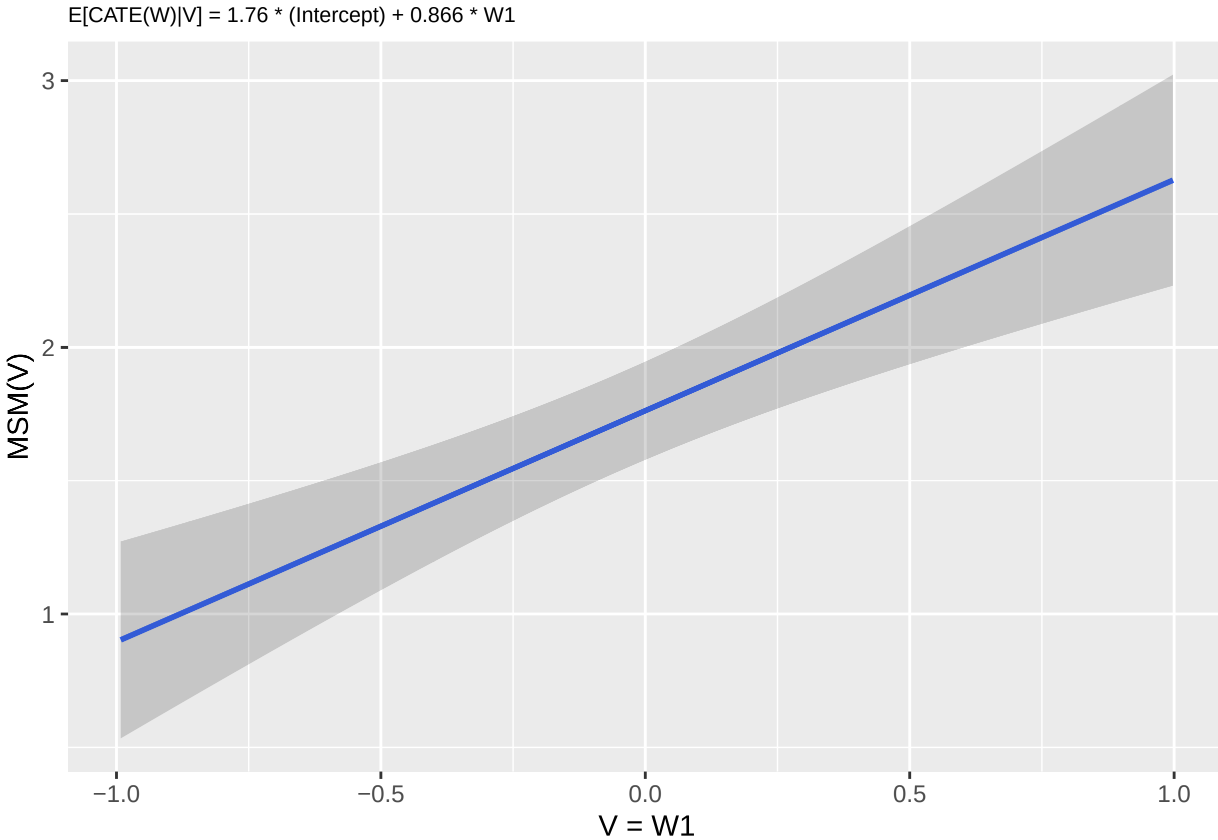
```r
summary(output)
```

```
## A causalglm fit object obtained from msmglm for the estimand CATE with formula:
## E[CATE(W)|V] = 1.76 * (Intercept) + 0.866 * W1
##
## Coefficient estimates and inference:
##    type       param  tmle_est         se      lower     upper   Z_score p_value
```

```
## 1: CATE (Intercept) 1.7623920 0.09402283 1.5781107 1.946673 296.37338       0
## 2: CATE          W1 0.8662941 0.17198018 0.5292191 1.203369  79.64471       0
```

```r
plot_msm(output)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
# This gives a nonparametric estimate for the marginal ATE
formula <-  ~ 1
output <- msmglm(formula,
      data,
      V = "W1",
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      learning_method = "HAL"
      )
```

```
## (max) epsilon: 7.516807e-04 max(abs(ED)): 1.865175e-17
```

```r
summary(output)
```

```
## A causalglm fit object obtained from msmglm for the estimand CATE with formula:
## E[CATE(W)|V] = 1.74 * (Intercept)
##
## Coefficient estimates and inference:
##    type       param tmle_est         se    lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.741025 0.09905414 1.546883 1.935168 277.9089       0
```

Learning other estimands.

All of the vignette discussed so far can be applied to other estimands by specifying a different "estimand" argument.

Let us begin with npglm (msmglm acts in the same exact way). Both npglm and msmglm support the CATE, OR, RR, CATT and TSM

```
n <- 250
W1 <- runif(n, min = -1, max = 1)
W2 <- runif(n, min = -1, max = 1)
A <- rbinom(n, size = 1, prob = plogis((W1 + W2  )/3))
Y <- rnorm(n, mean = A * (1 + W1 + 2*W1^2) + sin(4 * W2) + sin(4 * W1), sd = 0.3)
data <- data.frame(W1, W2,A,Y)
# CATE
formula = ~ poly(W1, degree = 2, raw = TRUE)
output <- npglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE")
```

```
## (max) epsilon: 1.522343e-02 max(abs(ED)): 1.550947e-16
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.04 * (Intercept) + 0.902 * poly(W1, degree = 2, raw = TRUE)1 + 2.09 * poly(W1, degree = 2
##
## Coefficient estimates and inference:
##    type                             param  tmle_est         se     lower
## 1: CATE                       (Intercept) 1.0418577 0.05751445 0.9291314
## 2: CATE poly(W1, degree = 2, raw = TRUE)1 0.9020606 0.06398551 0.7766513
## 3: CATE poly(W1, degree = 2, raw = TRUE)2 2.0896941 0.12570843 1.8433101
##       upper  Z_score p_value
## 1: 1.154584 286.4187       0
## 2: 1.027470 222.9072       0
## 3: 2.336078 262.8381       0
```

```
# CATT, lets reuse fit
output <- npglm(formula,
      output,
      estimand = "CATT")
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: 2.314991e-02 max(abs(ED)): 2.300590e-16
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATT with formula:
## CATT(W) = 1.05 * (Intercept) + 0.905 * poly(W1, degree = 2, raw = TRUE)1 + 2.06 * poly(W1, degree = 2
##
## Coefficient estimates and inference:
##    type                             param  tmle_est         se     lower
## 1: CATT                       (Intercept) 1.0458915 0.05470906 0.9386637
## 2: CATT poly(W1, degree = 2, raw = TRUE)1 0.9045589 0.07348656 0.7605279
## 3: CATT poly(W1, degree = 2, raw = TRUE)2 2.0633212 0.12926115 1.8099740
##       upper  Z_score p_value
## 1: 1.153119 302.2716       0
```

```
## 2: 1.048590 194.6251       0
## 3: 2.316668 252.3881       0
```

```
# TSM, note this provides a list of npglm objects for each level of `A`.
outputs <- npglm(formula,
      output,
      estimand = "TSM")
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: 2.536119e-02 max(abs(ED)): 6.775136e-17
```

```
summary(outputs[[1]])
```

```
## A causalglm fit object obtained from npglm for the estimand TSM with formula:
## TSM(W) = 1.12 * E[Y_{A=1}]: (Intercept) + 1.31 * E[Y_{A=1}]: poly(W1, degree = 2, raw = TRUE)1 + 1.82
##
## Coefficient estimates and inference:
##    type                                         param tmle_est         se
## 1:  TSM                        E[Y_{A=1}]: (Intercept) 1.116153 0.09793815
## 2:  TSM E[Y_{A=1}]: poly(W1, degree = 2, raw = TRUE)1 1.311449 0.11478525
## 3:  TSM E[Y_{A=1}]: poly(W1, degree = 2, raw = TRUE)2 1.819316 0.23298786
##        lower    upper  Z_score p_value
## 1: 0.9241979 1.308108 180.1947       0
## 2: 1.0864736 1.536424 180.6488       0
## 3: 1.3626686 2.275964 123.4653       0
```

```
summary(outputs[[2]])
```

```
## A causalglm fit object obtained from npglm for the estimand TSM with formula:
## TSM(W) = 0.0739 * E[Y_{A=0}]: (Intercept) + 0.411 * E[Y_{A=0}]: poly(W1, degree = 2, raw = TRUE)1 + -
##
## Coefficient estimates and inference:
##    type                                         param   tmle_est         se
## 1:  TSM                        E[Y_{A=0}]: (Intercept)  0.0738934 0.09099383
## 2:  TSM E[Y_{A=0}]: poly(W1, degree = 2, raw = TRUE)1  0.4110514 0.10387451
## 3:  TSM E[Y_{A=0}]: poly(W1, degree = 2, raw = TRUE)2 -0.2723207 0.20675991
##         lower     upper  Z_score p_value
## 1: -0.1044512 0.2522380 12.83996       0
## 2:  0.2074611 0.6146417 62.56870       0
## 3: -0.6775627 0.1329212 20.82497       0
```

Both the OR and RR estimands provide the original coefficient estimates and their exponential transforms. This is because the parametric model/formula is actually for the log RR and log OR (that is log-linear models). The predict function gives the exponential of the linear predictor (so actually predicts the OR and RR).

```
# odds ratio
n <- 250
W <- runif(n, min = -1,  max = 1)
A <- rbinom(n, size = 1, prob = plogis(W))
Y <- rbinom(n, size =  1, prob = plogis(A + A * W + W + sin(5 * W)))
data <- data.frame(W, A, Y)
output <-
  npglm(
    ~1+W,
    data,
    W = c("W"), A = "A", Y = "Y",
```

```
    estimand = "OR"
  )
```

```
## risk_change: -1.180519e-04 (max) epsilon: 2.499999e-02 max(abs(ED)): 6.295175e-03
## risk_change: -9.680710e-07 (max) epsilon: 2.624654e-03 max(abs(ED)): 3.495454e-04
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand OR with formula:
## log OR(W) = 0.898 * (Intercept) + 0.125 * W
##
## Coefficient estimates and inference:
##    type       param  tmle_est        se      lower    upper  psi_exp lower_exp
## 1:   OR (Intercept) 0.8981221 0.2879931  0.3336661 1.462578 2.454989 1.3960769
## 2:   OR           W 0.1254387 0.4886152 -0.8322294 1.083107 1.133646 0.4350782
##    upper_exp   Z_score     p_value
## 1:  4.317075 49.308678 0.0000e+00
## 2:  2.953842  4.059146 4.9253e-05
```

```
output <-
  spglm(
    ~1+W,
    data,
    W = c("W"), A = "A", Y = "Y",
    estimand = "OR"
  )
```

```
## risk_change: -2.480462e-05 (max) epsilon: 2.719917e-02 max(abs(ED)): 8.301994e-04
```

```
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand OR with formula:
## log OR(W) = 0.897 * (Intercept) + 0.135 * W
##
## Coefficient estimates and inference:
##    type       param  tmle_est        se      lower    upper  psi_exp lower_exp
## 1:   OR (Intercept) 0.8966970 0.2910512  0.3262471 1.467147 2.451492 1.3857577
## 2:   OR           W 0.1352182 0.4890799 -0.8233608 1.093797 1.144787 0.4389539
##    upper_exp   Z_score     p_value
## 1:  4.336844 48.713156 0.0000e+00
## 2:  2.985589  4.371447 1.2343e-05
```

```
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand OR with formula:
## log OR(W) = 0.897 * (Intercept) + 0.135 * W
##
## Coefficient estimates and inference:
##    type       param  tmle_est        se      lower    upper  psi_exp lower_exp
## 1:   OR (Intercept) 0.8966970 0.2910512  0.3262471 1.467147 2.451492 1.3857577
## 2:   OR           W 0.1352182 0.4890799 -0.8233608 1.093797 1.144787 0.4389539
##    upper_exp   Z_score     p_value
## 1:  4.336844 48.713156 0.0000e+00
## 2:  2.985589  4.371447 1.2343e-05
```

```
# relative risk
n <- 250
```

```r
W <- runif(n, min = -1,  max = 1)
A <- rbinom(n, size = 1, prob = plogis(W))
Y <- rpois(n, lambda = exp( A * (1 + W + 2*W^2)  + sin(5 * W)))
data <- data.frame(W, A, Y)
formula = ~ poly(W, degree = 2, raw = TRUE)
output <-
  npglm(
    formula,
    data,
    W = "W", A = "A", Y = "Y",
    estimand = "RR",
    verbose = FALSE
  )
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand RR with formula:
## log RR(W) = 1.13 * (Intercept) + 0.814 * poly(W, degree = 2, raw = TRUE)1 + 1.74 * poly(W, degree = 2
##
## Coefficient estimates and inference:
##    type                              param  tmle_est        se     lower    upper
## 1:   RR                        (Intercept) 1.1272164 0.1686676 0.7966339 1.457799
## 2:   RR poly(W, degree = 2, raw = TRUE)1 0.8142522 0.1931903 0.4356063 1.192898
## 3:   RR poly(W, degree = 2, raw = TRUE)2 1.7404999 0.4104832 0.9359675 2.545032
##     psi_exp lower_exp upper_exp   Z_score p_value
## 1: 3.087052  2.218062  4.296492 105.66850       0
## 2: 2.257487  1.545900  3.296622  66.64134       0
## 3: 5.700192  2.549679 12.743639  67.04225       0
```

```r
output <-
  spglm(
    formula,
    data,
    W = "W", A = "A", Y = "Y",
    estimand = "RR",
    verbose = FALSE
  )
summary(output)
```

```
## A causalglm fit object obtained from spglm for the estimand RR with formula:
## log RR(W) = 1.07 * (Intercept) + 0.875 * poly(W, degree = 2, raw = TRUE)1 + 1.87 * poly(W, degree = 2
##
## Coefficient estimates and inference:
##    type                              param  tmle_est        se     lower    upper
## 1:   RR                        (Intercept) 1.0675756 0.1509340 0.7717504 1.363401
## 2:   RR poly(W, degree = 2, raw = TRUE)1 0.8747098 0.1717774 0.5380323 1.211387
## 3:   RR poly(W, degree = 2, raw = TRUE)2 1.8742303 0.3461052 1.1958765 2.552584
##     psi_exp lower_exp upper_exp   Z_score p_value
## 1: 2.908320  2.163550  3.909466 111.83598       0
## 2: 2.398179  1.712634  3.358141  80.51336       0
## 3: 6.515802  3.306455 12.840240  85.62189       0
```

```r
output <-
  msmglm(
    formula,
    data,
```

```
    V = "W",
    W = "W", A = "A", Y = "Y",
    estimand = "RR",
    verbose = FALSE
  )
summary(output)
```

```
## A causalglm fit object obtained from msmglm for the estimand RR with formula:
## log E[RR(W)|V] = 1.13 * (Intercept) + 0.814 * poly(W, degree = 2, raw = TRUE)1 + 1.74 * poly(W, degr
##
## Coefficient estimates and inference:
##    type                            param  tmle_est        se     lower    upper
## 1:   RR                      (Intercept) 1.1264602 0.1677209 0.7977332 1.455187
## 2:   RR poly(W, degree = 2, raw = TRUE)1 0.8140069 0.1909705 0.4397116 1.188302
## 3:   RR poly(W, degree = 2, raw = TRUE)2 1.7412471 0.4057276 0.9460356 2.536459
##     psi_exp lower_exp upper_exp   Z_score p_value
## 1: 3.084718  2.220502  4.285285 106.19366       0
## 2: 2.256933  1.552260  3.281505  67.39564       0
## 3: 5.704453  2.575479 12.634847  67.85719       0
```

Custom learners with sl3

We refer to the documentation of the tlverse/sl3 package for how learners work. To specify custom learners for the propensity score use the argument sl3_learner_A and to specify custom learners for the outcome conditional mean use the argument sl3_learner_Y. For spglm, keep in mind the argument "append_design_matrix" when choosing learners. A good rule of thumb for spglm is to think of sl3_learner_Y as a learner for $E[Y|A = 0, W]$. For msmglm and npglm, the learning is fully nonparametric and the regression is performed how you would expect (a standard design matrix containing W and A is passed to the learner). For msmglm and npglm, make sure the learner models interactions, specifically treatment interactions, as these are crucial for fitting the conditional treatment effect estimands well.

```
library(sl3)
lrnr_A <- Lrnr_gam$new()
lrnr_Y <- Lrnr_xgboost$new(max_depth = 4)
lrnr_Y <- Lrnr_cv$new(lrnr_Y, full_fit = TRUE) #cross-fit xgboost

n <- 250
W1 <- runif(n, min = -1, max = 1)
W2 <- runif(n, min = -1, max = 1)
A <- rbinom(n, size = 1, prob = plogis((W1 + W2  )/3))
Y <- rnorm(n, mean = A * (1 + W1 + 2*W1^2) + sin(4 * W2) + sin(4 * W1), sd = 0.3)
data <- data.frame(W1, W2,A,Y)
# CATE
formula = ~ poly(W1, degree = 2, raw = TRUE)
output <- npglm(formula,
      data,
      W = c("W1", "W2"), A = "A", Y = "Y",
      estimand = "CATE",
      sl3_Learner_A = lrnr_A,
      sl3_Learner_Y = lrnr_Y)
```

```
## (max) epsilon: 6.742920e-02 max(abs(ED)): 4.786935e-16
```

Other arguments

See the documentation for other arguments for all methods. We note that the remaining arguments will likely not be needed for the average user.

Effects of categorical treatments with npglm and msmglm

For `msmglm` and `npglm`, the CATE, CATT, TSM and RR can be learned for categorical treatments relative to a control treatment. To do this, you need to specify the arguments treatment_level and control_level. The estimands are then user-specified parametric models in W for

$$W \mapsto E[Y|A = a, W] - E[Y|A = 0, W]$$

$$W \mapsto E[Y|A = a, W]$$

$$W \mapsto E[Y|A = a, W]/E[Y|A = 0, W]$$

where a is the specified treatment level.

```
n <- 250
V <- runif(n, min = -1, max = 1)
W <- runif(n, min = -1, max = 1)
A <- rbinom(n, size = 1, prob = 0.66*plogis(W))
A[A==1] <- 2
A[A==0] <- rbinom(n, size = 1, prob = plogis(W))
```

```
## Warning in A[A == 0] <- rbinom(n, size = 1, prob = plogis(W)): number of items
## to replace is not a multiple of replacement length
```

```
table(A)
```

```
## A
##  0  1  2
## 91 72 87
```

```
Y <- rnorm(n, mean = A * (1 + W  ) + W , sd = 0.5)
data <- data.table(W,A,Y)

output_init <- npglm(~1+W, data, W = "W", A = "A", Y = "Y", estimand = "CATE", learning_method = "mars"
```

```
## (max) epsilon: 1.544953e-02 max(abs(ED)): 4.198378e-17
```

```
summary(output_init)
```

```
## A causalglm fit object obtained from npglm for the estimand CATE with formula:
## CATE(W) = 1.01 * (Intercept) + 0.988 * W
##
## Coefficient estimates and inference:
##    type       param  tmle_est         se     lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.0115954 0.08044989 0.8539165 1.169274 198.8160       0
## 2: CATE           W 0.9875495 0.13076011 0.7312644 1.243835 119.4136       0
```

```
output <- msmglm(~1+W, data, V = "W", W = "W", A = "A", Y = "Y", estimand = "CATE", learning_method = "m
```

```
## (max) epsilon: 1.544953e-02 max(abs(ED)): 4.198378e-17
```

```
summary(output)
```

```
## A causalglm fit object obtained from msmglm for the estimand CATE with formula:
## E[CATE(W)|V] = 1.01 * (Intercept) + 0.988 * W
```

```
##
## Coefficient estimates and inference:
##    type       param  tmle_est         se     lower    upper  Z_score p_value
## 1: CATE (Intercept) 1.0115954 0.08044989 0.8539165 1.169274 198.8160       0
## 2: CATE           W 0.9875495 0.13076011 0.7312644 1.243835 119.4136       0
```

```
# Reuse fits
output <- npglm(~1+W, output_init , estimand = "CATT",   treatment_level = 2, control_level = 0)
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: 2.021694e-02 max(abs(ED)): 6.788060e-17
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand CATT with formula:
## CATT(W) = 2.02 * (Intercept) + 1.9 * W
##
## Coefficient estimates and inference:
##    type       param tmle_est         se    lower    upper  Z_score p_value
## 1: CATT (Intercept) 2.017564 0.07707207 1.866505 2.168622 413.9046       0
## 2: CATT           W 1.901519 0.12656576 1.653454 2.149583 237.5496       0
```

```
output <- npglm(~1+W, output_init , estimand = "TSM",   treatment_level = c(0,1,2))
```

```
## [1] "Reusing previous fit..."
## (max) epsilon: 2.149722e-02 max(abs(ED)): 1.329457e-16
```

```
lapply(output, summary)
```

```
## A causalglm fit object obtained from npglm for the estimand TSM with formula:
## TSM(W) = -0.0235 * E[Y_{A=0}]: (Intercept) + 1.09 * E[Y_{A=0}]: W
##
## Coefficient estimates and inference:
##    type                   param    tmle_est         se      lower      upper
## 1:  TSM E[Y_{A=0}]: (Intercept) -0.02345778 0.05473499 -0.1307364 0.08382083
## 2:  TSM           E[Y_{A=0}]: W  1.09247342 0.08579828  0.9243119 1.26063496
##       Z_score   p_value
## 1:   6.776287 1.233e-11
## 2: 201.327131 0.000e+00
## A causalglm fit object obtained from npglm for the estimand TSM with formula:
## TSM(W) = 0.989 * E[Y_{A=1}]: (Intercept) + 2.08 * E[Y_{A=1}]: W
##
## Coefficient estimates and inference:
##    type                   param  tmle_est         se     lower    upper
## 1:  TSM E[Y_{A=1}]: (Intercept) 0.9890342 0.05933185 0.8727459 1.105323
## 2:  TSM           E[Y_{A=1}]: W 2.0819848 0.10021200 1.8855729 2.278397
##     Z_score p_value
## 1: 263.5685       0
## 2: 328.4943       0
## A causalglm fit object obtained from npglm for the estimand TSM with formula:
## TSM(W) = 2 * E[Y_{A=2}]: (Intercept) + 2.97 * E[Y_{A=2}]: W
##
## Coefficient estimates and inference:
##    type                   param tmle_est         se    lower    upper  Z_score
## 1:  TSM E[Y_{A=2}]: (Intercept) 2.000139 0.05294942 1.896360 2.103918 597.2677
## 2:  TSM           E[Y_{A=2}]: W 2.972092 0.07997225 2.815349 3.128835 587.6151
##    p_value
```

```
## 1:       0
## 2:       0
```

```
## $`E[Y_{A=0}]`
##    type                    param    tmle_est         se      lower      upper
## 1:  TSM E[Y_{A=0}]: (Intercept) -0.02345778 0.05473499 -0.1307364 0.08382083
## 2:  TSM           E[Y_{A=0}]: W  1.09247342 0.08579828  0.9243119 1.26063496
##       Z_score   p_value
## 1:   6.776287 1.233e-11
## 2: 201.327131 0.000e+00
##
## $`E[Y_{A=1}]`
##    type                    param  tmle_est         se     lower    upper
## 1:  TSM E[Y_{A=1}]: (Intercept) 0.9890342 0.05933185 0.8727459 1.105323
## 2:  TSM           E[Y_{A=1}]: W 2.0819848 0.10021200 1.8855729 2.278397
##     Z_score p_value
## 1: 263.5685       0
## 2: 328.4943       0
##
## $`E[Y_{A=2}]`
##    type                    param tmle_est         se    lower    upper  Z_score
## 1:  TSM E[Y_{A=2}]: (Intercept) 2.000139 0.05294942 1.896360 2.103918 597.2677
## 2:  TSM           E[Y_{A=2}]: W 2.972092 0.07997225 2.815349 3.128835 587.6151
##    p_value
## 1:       0
## 2:       0
##
## $estimand
##    Length     Class      Mode
##         1 character character
##
## $levels_A
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.5     1.0     1.0     1.5     2.0
```

```r
n <- 250
V <- runif(n, min = -1, max = 1)
W <- runif(n, min = -1, max = 1)
A <- rbinom(n, size = 1, prob = 0.66*plogis(W))
A[A==1] <- 2
A[A==0] <- rbinom(n, size = 1, prob = plogis(W))
```

```
## Warning in A[A == 0] <- rbinom(n, size = 1, prob = plogis(W)): number of items
## to replace is not a multiple of replacement length
```

```r
table(A)
```

```
## A
##  0  1  2
## 81 93 76
```

```r
Y <- rpois(n, lambda = exp( A * (1 + W)  + sin(5 * W)))
data <- data.table(W,A,Y)

output_init <- npglm(~1+W, data, W = "W", A = "A", Y = "Y", estimand = "RR", learning_method = "gam", t
```

```
## risk_change: -4.403048e-02 (max) epsilon: 2.499999e-02 max(abs(ED)): 1.148280e+00
```

```
## risk_change: -2.638082e-02 (max) epsilon: 2.499999e-02 max(abs(ED)): 6.724864e-01
## risk_change: -1.239399e-02 (max) epsilon: 2.499999e-02 max(abs(ED)): 1.108231e-01
## risk_change: -4.260323e-04 (max) epsilon: 3.160725e-03 max(abs(ED)): 2.425985e-02
## risk_change: -5.650654e-05 (max) epsilon: 2.151528e-03 max(abs(ED)): 3.721015e-03
```

```
summary(output_init)
```

```
## A causalglm fit object obtained from npglm for the estimand RR with formula:
## log RR(W) = 0.912 * (Intercept) + 1.33 * W
##
## Coefficient estimates and inference:
##    type       param  tmle_est        se     lower    upper  psi_exp lower_exp
## 1:   RR (Intercept) 0.9123236 0.1771783 0.5650605 1.259587 2.490102  1.759554
## 2:   RR           W 1.3277945 0.3712439 0.6001698 2.055419 3.772714  1.822428
##    upper_exp  Z_score p_value
## 1:  3.523965 81.41574       0
## 2:  7.810111 56.55116       0
```

```
output <- npglm(~1+W, output_init , estimand = "RR",   treatment_level = 2, control_level = 0)
```

```
## [1] "Reusing previous fit..."
## risk_change: -1.931557e-01 (max) epsilon: 2.499999e-02 max(abs(ED)): 8.372314e-01
## risk_change: -1.900186e-02 (max) epsilon: 9.438307e-03 max(abs(ED)): 9.964378e-02
## risk_change: -1.140377e-03 (max) epsilon: 7.106237e-04 max(abs(ED)): 2.156471e-01
## risk_change: -7.885511e-04 (max) epsilon: 2.719563e-03 max(abs(ED)): 9.315535e-02
## risk_change: -3.401372e-04 (max) epsilon: 4.965813e-04 max(abs(ED)): 9.718894e-02
## risk_change: -1.499144e-04 (max) epsilon: 1.205192e-03 max(abs(ED)): 4.082228e-02
## risk_change: -6.840305e-05 (max) epsilon: 2.327098e-04 max(abs(ED)): 4.444077e-02
## risk_change: -2.820244e-05 (max) epsilon: 5.117513e-04 max(abs(ED)): 1.657583e-02
## risk_change: -1.323319e-05 (max) epsilon: 1.010039e-04 max(abs(ED)): 2.056839e-02
## risk_change: -5.725552e-06 (max) epsilon: 2.276880e-04 max(abs(ED)): 7.188890e-03
## risk_change: -2.711307e-06 (max) epsilon: 4.532755e-05 max(abs(ED)): 9.551378e-03
## risk_change: -1.204161e-06 (max) epsilon: 1.037767e-04 max(abs(ED)): 3.235590e-03
```

```
summary(output)
```

```
## A causalglm fit object obtained from npglm for the estimand RR with formula:
## log RR(W) = 1.88 * (Intercept) + 2.17 * W
##
## Coefficient estimates and inference:
##    type       param tmle_est        se    lower    upper  psi_exp lower_exp
## 1:   RR (Intercept) 1.883185 0.1541914 1.580976 2.185395 6.574413  4.859695
## 2:   RR           W 2.169397 0.4951681 1.198885 3.139909 8.753004  3.316418
##    upper_exp   Z_score p_value
## 1:  8.894161 193.10913       0
## 2: 23.101756  69.27178       0
```

Effects of a continuous treatment with contglm

The function `contglm` supports treatment effects for continuous treatments. Currently, only the CATE estimand is supported. Specifically, `contglm` computes estimates and nonparametric inference for the best approximation of the true CATE $E[Y|A=a,W] - E[Y|A=0,W]$ with respect to the parametric working model $E[Y|A=a,W] - E[Y|A=0,W] = 1(a>0) \cdot \beta^T \underline{f}(W) + a \cdot \beta^T \underline{g}(W)$ where $\underline{f}(W)$ and $\underline{g}(W)$ are user-specified parametric models. $\underline{f}(W)$ is specified with the argument `formula\_binary` and captures the treatment effect caused by being treated or not treated $(1(A>0))$. $\underline{g}(W)$ is specified with the argument `formula_continuous` and captures the treatment effect caused by dosage of continuous effects in the treatment

A. Note A should be a nonnegative treatment value with $A = 0$ being the placebo group and $A > 0$ being a continuous or ordered numeric dose value.

Thus, unlike other functions, both the argument `formula\_continuous` and `formula\_binary` need to be specified.

```
n <- 1000
W <- runif(n, min = -1, max = 1)
Abinary <- rbinom(n ,size = 1, plogis(W))
A <- rgamma(n, shape = 1, rate = exp(W))
A <- A * Abinary
quantile(A)
```

```
##        0%       25%       50%       75%      100%
## 0.0000000 0.0000000 0.0000000 0.6058994 9.8877696
```

```
Y <- rnorm(n, mean =   (A>0) + A * (1 + W  ) + W , sd = 0.5)
data <- data.table(W,A,Y)
out <- contglm(formula_continuous = ~1+W, formula_binary = ~1, data =data, W = "W", A = "A", Y = "Y")
```

```
## (max) epsilon: -6.973032e-03 max(abs(ED)): 3.901046e-17
```

```
summary(out)
```

```
## A causalglm fit object obtained from contglm for the estimand CATE with formula:
## contCATE(A,W) = 0.987 * 1(A>0)*(Intercept) + 1 * A*(Intercept) + 1.02 * A*W
##
## Coefficient estimates and inference:
##        type               param  tmle_est         se     lower    upper  Z_score
## 1: contCATE 1(A>0)*(Intercept) 0.9868226 0.04896798 0.8908471 1.082798 637.2750
## 2: contCATE      A*(Intercept) 1.0003411 0.03730014 0.9272342 1.073448 848.0815
## 3: contCATE                A*W 1.0171211 0.06641432 0.8869514 1.147291 484.2960
##    p_value
## 1:       0
## 2:       0
## 3:       0
```

```
# The CATE predictions are now a function of `A`
head(predict(out))
```

```
##   1(A>0)*(Intercept) A*(Intercept)        A*W  CATE(W)       se  CI_left
## 1                  0      0.000000  0.0000000 0.000000 0.000000 0.000000
## 2                  1      1.270437  1.0079589 3.282909 2.741117 3.113013
## 3                  1      2.863632 -0.1938386 3.654274 2.616693 3.492089
## 4                  0      0.000000  0.0000000 0.000000 0.000000 0.000000
## 5                  0      0.000000  0.0000000 0.000000 0.000000 0.000000
## 6                  0      0.000000  0.0000000 0.000000 0.000000 0.000000
##   CI_right  Z-score p-value
## 1 0.000000      NaN     NaN
## 2 3.452805 37.87314       0
## 3 3.816458 44.16196       0
## 4 0.000000      NaN     NaN
## 5 0.000000      NaN     NaN
## 6 0.000000      NaN     NaN
```

Variable importance