

Decomposing the R -squared of a Regression Using the Shapley Value in SAS®

Charles D. Coleman, US Census Bureau

DISCLAIMER

Any views expressed are those of the author and not necessarily those of the U.S. Census Bureau.

ABSTRACT

After creating a linear regression model, quantifying the importance of each variable is often desired, either for academic purposes or to design the most effective interventions. This normally involves decomposing the variance, or, equivalently, R^2 . Using the Shapley Value, known in this literature as the LMV, it is possible to do such a decomposition. This decomposition can be applied to models with interactions and is robust against multicollinearity. It has the interpretation that it is the marginal contribution of each variable to R^2 .

This is implemented in macro %DECOMPOSE_R2 using PROCs GLM, IML, macro string functions and data step-like macro processing.

INTRODUCTION

Variable importance is a useful concept in statistical modelling. In the case of linear regression, it is each variable's contribution to explaining the variance, equivalently R^2 . The Shapley Value, known as the LMV in this context, can perform these decomposition. The resulting R^2 shares have the interpretation of being each variable's marginal contribution to R^2 . The LMV can be applied to models with interactions and is robust against multicollinearity. Macro %DECOMPOSE_R2 performs this decomposition and outputs for each variable its total and share of R^2 explained to the specified output dataset. This macro takes as its arguments, the input dataset of independent and dependent variables, without interactions, the dependent variable, a list of independent variables, an optional list of interactions and the output dataset. Each interaction term must contain only the variables already listed as being independent variables.

The macro first runs regressions of all subsets of the independent variables using PROC GLM, including interactions when present. This provides the set of R^2 values needed to calculate the LMV. When no interactions are present, the macro proceeds to the PROC IML call. When interactions are present, the macro identifies the regressions that contain all interactions containing each set of independent variables. This requires data step-like macro processing, nested loops and PROC SORT calls. The PROC IML call straightforwardly computes each variable's LMV and outputs it to an intermediate dataset. The variable names are added to this dataset, creating the output dataset.

This macro comes with some provisos. It is not a modeling aid: it should only be used to understand the final model. It also has an exponential run time, proportionate to 2^n , where n is the number of independent variables and interactions. Thus, it has a practical limit to the number of these variables.

VARIABLE IMPORTANCE

Variable importance is not a well-defined concept (Grömping, 2015). Its practical meaning varies across context. In regression and related contexts, it is taken as the share of variance, equivalently R^2 , explained by a variable. Grömping (2015) demonstrates that the LMV satisfies a large number of desirable criteria. The LMV gets its name from the authors Lindeman, Merenda and Gold (1980,119-121,125-127). The LMV itself has been rediscovered many times. Besides Lindeman et al. (1980), Kruskal (1987), Chevan and Sutherland (1991), Lipovetski and Conklin (2001) and Israeli (2007) have each independently proposed the LMV. Stufken (1992) demonstrated the identity of the LMV with the Shapley Value (Shapley, 1953).

LMV and other variable importance measures have been successfully used in fields from genetics to political science (Grömping, 2015). Fields (2003) successfully applied a flawed variant of the LMV (Israeli, 2007) to an industrial problem.

Theil (1971, 175-181) was perhaps the first to propose an R^2 decomposition. His method has the severe drawbacks of being incapable of handling interactions and having each variable's share of R^2 proportionate to t^2 (Theil 1971, 175). The example in this paper displays a regression with an interaction whose noninteraction t -values differ. Finally, Theil's method requires an ad hoc adjustment to handle correlations among the independent variables. A cursory search of the internet appears to indicate that his method is most widely taught, at least by econometricians.

THE SHAPLEY VALUE AND ITS REGRESSION INTERPRETATION

The Shapley Value (Shapley, 1953) for a player i in cooperative game theory measures i 's expected payoff. Equivalently, it is i 's expected marginal contribution. We use the latter interpretation. The Shapley Value for player i , $\varphi_i(v)$, is defined as

$$\varphi_i(v) = \sum_{S \subseteq N \setminus i} \frac{|S|!(n-|S|-1)!}{n!} [v(S \cup \{i\}) - v(S)] = \frac{1}{n!} \sum_R [v(P_i^R \cup \{i\}) - v(P_i^R)], \quad (1)$$

where v is payoff to coalition S , N is set of all n players, R is a player order and P_i^R is the set of players before i in order R . The empty coalition has 0 payoff, that is, $v(\emptyset) = 0$. The Efficiency Axiom implies

$$\sum_{i=1}^n \varphi_i(v) = v(N) \quad (2)$$

To decompose R^2 , $v(S) \equiv R^2(S)$, that is, we define $v(S)$ to be the R^2 of a regression including only the variables in S . From this, it follows that $v(N) = R^2$. $\varphi_i(v)$ is interpreted as the marginal contribution of variable i to overall R^2 .

COMPUTING THE SHAPLEY VALUES

For each set of independent variables, S , and the interactions containing those variables, a linear regression is run and the value of R^2 is retained. Letting

$$\gamma(m, n) = m!(n - m - 1)!/n!, \quad (3)$$

the increase in R^2 from the contribution of adding variable i to S is computed by the γ -weighted difference $\gamma(m, n)[R^2(S \cup \{i\}) - R^2(S)] = m!(n - m - 1)! [R^2(S \cup \{i\}) - R^2(S)]/n!$. This reduces to the first summand in equation (1). These values are computed for every set S not containing i and summed. The sign and value of m depend on the number of variables in a regression and whether variable i is in the regression. Let k be the number of variables in a regression. If variable i is in the regression, R^2 is multiplied by $\gamma(k - 1, n)$. If not, the multiplier is $-\gamma(k, n)$.

CALLING THE MACRO

The macro call is %DECOMPOSE_R2 (DSIN=, DEPVAR=, INDVAR=, INTERACTIONS=, DSOUT=). &DSIN contains all variables used in the regression excluding interactions. &DEPVAR is the dependent variable. &INDVAR is a blank-separated list of independent variables. &INTERACTIONS is quote-enclosed list of variables expressed using the syntax of PROC GLM and the like, e.g., "X1*X2 X2*X3". All interacted variables must be included in &INDVAR. &DSOUT is the dataset in which the results are to be stored.

MACRO OUTPUT

Dataset &DSOUT contains three variables: VAR, R2EXPLAINED and R2SHARE, with one record per variable listed in &INDVAR. R2EXPLAINED is the amount of R^2 explained by VAR and R2SHARE = R2EXPLAINED / R^2 of the full regression.

INSIDE THE CODE

Because of the size and complexity of the code, only the major points are covered.

This macro makes extensive use of the `%sysfunc(countw())`, `%scan` and `%qscan` functions. The first counts words in a macro variable, that is, strings of characters separated by blanks. In this macro, the words are variable names or interaction products. Function `%scan()` is used to select words when they are delimited by blanks. Function `%qscan` with third argument `%str(*)` is used to select individual variables within interaction terms.

If interactions are present, the macro generates a list of interactions as variables `&INTERACTION1-&INTERACTION&NINTERACTIONS`, where `&NINTERACTIONS` is the number of interactions. PROC REG then runs all subsets regressions of `&INDVAR` and `&INTERACTION1-&INTERACTION&NINTERACTIONS`, if present. The `RSQUARE` option of PROC REG statement outputs R^2 to the `OUTEST=` dataset. The same option on the MODEL statement selects the maximum R^2 method of variable selection. Together, these options cause all subsets regressions to be performed and outputs R^2 of each to the `OUTEST=` dataset, in addition to coefficients and other data.

If any independent variable is accompanied by interactions, the macro finds the regression that contains all of its interactions. The first step is to locate the regressions that contain interaction terms that do not contain one or more of their independent variables. This involves using SAS's data step-like macro processing facilities. Similar to a data set, a dataset is opened and each observation is successively processed. The dataset must be closed when finished to avoid side effects. The code to open dataset ALL, output by PROC REG, is

```
%let id=%sysfunc(open(ALL));  
%let NObs=%sysfunc(attrn(&id,NOBS));  
%syscall set(id);
```

`%sysfunc(open(ALL))` opens dataset ALL for macro access, returning `&id` as its dataset identifier. The next statement counts the observations in dataset ALL for subsequent loop control. `%syscall set(id)` is the macro equivalent of the SET command within a data step.

A `%do` loop, indexed by macro variable `&L`, then sequentially processes each of observations in dataset ALL. command `%let rc=%sysfunc(fetchobs(&id,&L))` loads the `&L`'th observation into macro variables bearing the names of the dataset's fields with the values for that observation. The rest of the loop is too complex to describe in detail, so it is summarized with highlights. The first nested `%do` loop within this block builds the list of independent variables excluded from the current regression. The statement `"%if %index(&&&CURRENTVAR,.) = 1 %then %let EXCLUDEVARLIST = &EXCLUDEVARLIST. &CURRENTVAR"` first tests whether the value of `&CURRENTVAR` is missing. Three ampersands are required before `CURRENTVAR`. The last ampersand selects macro variable `&CURRENTVAR`. The preceding two ampersands are resolved into a single one and extract the value of `&CURRENTVAR`. The `%index` function locates the location of `'.'` in `&&&CURRENTVAR`. It is only first when missing. A missing value indicates that the corresponding independent variable is excluded from the regression, so it is appended to `&EXCLUDEVARLIST`.

The second nested `%do` loop scans the interactions for occurrences of variables in `&EXCLUDEVARLIST`. If one is found, macro variable `&DROP` is set equal to 1 and the loop ends. The loop control variables are incremented by the code. Each nested loop ends whenever `&DROP = 1` or it has run out of arguments. If `&DROP = 1`, every nested loop terminates.

The main loop ends by again extracting the `&L`'th observation of dataset ALL, this time into dataset L and using `&DROP` to determine whether to put a drop flag on it. The current observation has variable D set to 1 if `&DROP = 1`, missing otherwise, is appended to dataset ALL0. After the main `%do` loop completes, the statement `%let RC=%sysfunc(close(&id))` closes dataset ALL to macro processing. This is necessary to prevent side effects.

The next step is to retain only the regressions with all appropriate interactions. Daststep `ALLOA` drops all observations marked for dropping and creates a set of key variables for sorting. Each independent variable is represented by a key variable (`KEY1-KEY&NVAR`, where `&NVAR` is the number of independent variables). If a variable is missing, its key is set to '1', '2' otherwise. This is an arbitrary convention to mark the absence or presence of an independent variable, respectively. The first PROC SORT sorts by `KEY1-KEY&NVAR` and descending `_RSQ_` (R^2) to create one block of records for each set of keys, with the regression with highest R^2 first. The next PROC SORT uses the `NODUPKEY` option to select the first record of each block. This record has the highest R^2 within the key group and no unwanted interactions, so its R^2 is the R^2 of the nonmissing variables. In terms of the Shapley value, letting the nonmissing variables be S , $_RSQ_ = R^2(S)$.

If the regression has no interaction, all of the foregoing macro processing is replaced by simple copies.

A PROC IML call produces the Shapley values. It creates a matrix of indicators of the presence of independent variables to select the appropriate weights. For each variable, the R^2 of each regression is multiplied by its weight. The weights are summed for each variable as variable `R2EXPLAINED`. The largest R^2 is the R^2 of the whole regression, so `R2EXPLAINED` is divided by it to form `R2SHARE`. These variables are output to dataset `R2EXPLAINED`.

The final task is to put the variable names onto the observations and output `&DSOUT`. Data step `&DSOUT` executes the statement `VAR = scan("&indvar", _n_)` to assign the `_N_`'th independent variable name to variable `VAR`. A final PROC DATASETS call deletes the intermediate datasets created by the macro.

AN EXAMPLE

A real-world linear regression using PROC REG produced the results (t -values in parentheses)

$$Y = 102.22 - 1.09 X1 - 1.23 X2 + 0.06 X1 \cdot X2. \quad (4)$$

(121.52) (22.17) (13.67) (13.81)

JMP produced the centered specification (i.e., the interaction term is a multiple of the centered independent variables) of (4), where the overbars indicate averages:

$$Y = 92.41 - 0.51 X1 - 0.15 X2 + 0.06 (X1 - \overline{X1})(X2 - \overline{X2}), \quad (5)$$

(262.68) (27.46) (4.15) (13.81)

with t -values are in parentheses. Equations (4) and (5) have different t -values for all independent variables except the interactions. This underscores the unusability of Theil's (1971) incremental contribution method, as the incremental contributions are proportionate to t^2 , which differ between these specifications. Table 1 displays the LMV decomposition of regression (4), whose interaction is a function of only the independent variables.

Variable	R^2	Share of R^2 (%)
X1	.205	79.7
X2	.052	20.3
Total	.257	100.0

Table 1. LMV Decomposition of Regressions (4) and (5)

Specification (4) was used, as its interaction is a function of only the independent variables.

CONCLUSION

This macro provides the LMV, equivalently, Shapley value, decomposition of the R^2 of a regression to understand variable importance. The results can aid the understanding of a regression model and to

design interventions to improve outcomes. The macro uses several SAS facilities: macro string functions, macro data step-like functions and PROC IML.

REFERENCES

- Chevan, A., and Sutherland, M. 1991. "Hierarchical Partitioning." *The American Statistician*, 90-96.
- Fields, G.S. 2003. "Regression-Based Decompositions: A New Tool for Managerial Decision-Making." Accessed December 2, 2016. <http://www.ilr.cornell.edu/extension/files/20031203100122-pub1248.pdf>.
- Grömping, U. 2015. "Variable Importance in Regression Models," *WIREs Computational Statistics*, 7:137–152. DOI: 10.1002/wics.1346.
- Hart, S. and A. Mas-Colell. 1996. "Bargaining and value," *Econometrica*. 64:357–380.
- Israeli, O. 2007. "A Shapley-based decomposition of the R-square of a linear regression." *Journal of Economic Inequality*. 5:199-212. DOI: 10.1007/s10888-006-9036-6.
- Lindeman R.H., Merenda P.F. and R.Z. Gold (1980), *Introduction to Bivariate and Multivariate Analysis*, Glenview, IL: Scott Foresman.
- Lipovetski, S. and M. Conklin. 2001. "Analysis of regression in game theory approach." *Stochastic Models in Business and Industry*. 17:319-330.
- Shapley L.S. 1953. "A Value for n -person Games." In *Contributions to the Theory of Games*, volume II. H.W. Kuhn and A.W. Tucker, editors. Princeton, NJ: Princeton University Press.
- Stufken J. 1992. "On hierarchical partitioning." *The American Statistician*. 46:70–71.
- Theil, H. 1971. *Principles of Econometrics*. New York, NY: John Wiley and Sons.

ACKNOWLEDGMENTS

I'd like to thank Suzanne Dorinski for peer review and useful comments, and participants at an earlier presentation for their comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Charles D. Coleman
Economic Statistical Methods Division
US Census Bureau
CENHQ 5H482C
301-763-6068
charles.d.coleman@census.gov

APPENDIX: MACRO %DECOMPOSE_R2

```
%macro DECOMPOSE_R2(DSIN=, DEPVAR=, INDVAR=, INTERACTIONS=, DSOUT=);
```

```
/*
```

This macro decomposes the R^2 of a linear regression according to the Shapley decomposition of Lindeman R.H., Merenda P.F. and R.Z. Gold (1980).

The linear regression has be of the form $\&DEPVAR = a + B_1 * \&INDVAR * B_2 * \&INTERACTIONS$, where a is the intercept, B_1 and B_2 are vectors of coefficients, $\&INDVAR$ is the set of independent variables excluding interactions and $\&INTERACTIONS$ is the optional set of interactions among $\&INDVAR$.

Groups of dummy variables are not recommended, as the results will not be interpretable. A single yes/no dummy, however, is permissible.

Macro call:

```
%DECOMPOSE_R2(DSIN=<input dataset>, DEPVAR=<dependent variable>, INDVAR=<independent variables>,  
INTERACTIONS=<interactions among &INDVAR, &DSOUT>
```

```
&INDVAR should be listed singly without quotes: X1 X2 X3
```

```
&INTERACTIONS should be listed using PROC GLM and similar PROC conventions, enclosed in single or double quotes:
```

```
"X1*X2 X2*X3 X3*X4"
```

Output:

Dataset $\&DSOUT$ contains 3 variables: $\&INDVAR$, the independent variables whose contribution to R^2 is calculated, $R2EXPLAINED$, the amount of R^2 explained by $\&INDVAR$, and $R2SHARE$, the share of total R^2 explained by $\&INDVAR$.

This macro should only be used after a model has been estimated. It is not meant to estimate models.

This macro does not (yet) enable grouping independent variables.

NOTE: This macro's computation time goes with the FACTORIAL of the independent variables, including interactions. Therefore, it is not recommended for >10 independent variables.

References:

Developed by Chuck Coleman, April 11, 2013, December 7, 2016

```
*/
```

```
/* Strip the quotes from &INTERACTIONS, put into variable &UINTERACTIONS */
```

```
%if &INTERACTIONS ne %then %let UINTERACTIONS = %substr(&INTERACTIONS,2,%length(&INTERACTIONS)-2);
```

```
/* Count independent variables */
```

```
%let NVAR = %sysfunc(countw(&INDVAR,%str( )));
```

```
/* Count interactions */
```

```

%let NINTERACTIONS = %sysfunc(countw(&INTERACTIONS,%str( )));

/* Generate interaction variables */

%if &NINTERACTIONS > 0 %then %do;

    data INDATA;
        set &DSIN;
        %do I = 1 %to &NINTERACTIONS;
            INTERACTION&I = %qscan(&UINTERACTIONS,&I,%str( ));
        %end;
    run;

%end;
%else %do;

    data INDATA;
        set &DSIN;
    run;

%end;

/* Fit all models */

proc reg data=INDATA outest=ALL rsquare noprint;
    model &DEPVAR = &INDVAR %if &NINTERACTIONS > 0 %then INTERACTION1-INTERACTION&NINTERACTIONS; / selection = RSQUARE;
quit;

%if &NINTERACTIONS > 0 %then %do;

    /* Discard models with interactions that are not included in the independent variables */

    /* Make dataset ALL macro accessible */

    %let id=%sysfunc(open(ALL)); /* Open dataset ALL for macro access */
    %let NObs=%sysfunc(attrn(&id,NOBS)); /* Count the observations in dataset ALL, put into &NOBS */
    %syscall set(id); /* Macro equivalent of "set ALL" */

    /* Loop through each observation of ALL */

    %do L = 1 %to &NObs;

        %let rc=%sysfunc(fetchobs(&id,&L)); /* Read observation &L of dataset ALL */

        /* Build exclusion variable list */

        %let EXCLUDEVARLIST = ;
        %do I = 1 %to &NVAR;
            /* Select independent variable &I */
            %let CURRENTVAR = %scan(&INDVAR,&I);

```

```

        %if %index(&&CURRENTVAR,.) = 1 %then %let EXCLUDEVARLIST = &EXCLUDEVARLIST. &CURRENTVAR;
    %end;
    %let NEXCLUDEVARS = %sysfunc(countw(&EXCLUDEVARLIST,%str( )));

    /* If &NEXCLUDEVARS > 0, identify interactions, and, therefore, observations to drop */

    %let DROP = 0;          /* Initialize &DROP to default of not dropping */
    /* If variables are to be exclude, identify interactions containing them for dropping */
    %if &NEXCLUDEVARS > 0 %then %do;
        /* Because of %do %until construct, have to manually initialize and increment index variables */
        %let I = 0;
        %do %until (&I = &NINTERACTIONS or &DROP = 1);
            %let I = %eval(&I + 1);
            %if %index(&&INTERACTION&I,.) > 1 %then %do; /* If &&INTERACTION&I is nonmissing */
                /* Find corresponding interaction specification */
                %let CURRENTINTERACTION = %qscan(&UINTERACTIONS,&I,%str( ));
                /* Count variables in interaction, always >= 2 */
                %let NINTVARS = %sysfunc(count(&CURRENTINTERACTION,%str(*)));
                %let J = 0;
                /* This loop operates on individual variables contained in current interaction. */
                %do %until (&J = &NINTVARS or &DROP = 1);
                    %let J = %eval(&J + 1);
                    /* Identify individual variable */
                    %let CURRENTINTVAR = %qscan(&CURRENTINTERACTION,&J,%str(*));
                    %let K = 0;
                    /* Loop through &EXCLUDEVARLIST */
                    %do %until (&K = &NEXCLUDEVARS or &DROP = 1);
                        %let K = %eval(&K + 1);
                        /* Sequentially select from &EXCLUDEVARLIST */
                        %let CURRENTEXCLUDEVAR = %scan(&EXCLUDEVARLIST,&K);
                        /* If we have a match, an interaction contains a variable not in current
                        regression. Set &DROP = 1 and abort loop using %do %until logic */
                        %if &CURRENTEXCLUDEVAR = &CURRENTINTVAR %then %let DROP = 1;
                    %end;
                %end;
            %end;
        %end;
    %end;

    /* Read &L'th observation of dataset ALL, set value for D according whether observation should be
    dropped. */

    data L1;
        set ALL;
        if _N_ = &L;
            %if &DROP = 1 %then D = 1;
            %else D = .;;
    run;

    %end;
    %else %do;

```

```

        /* If no variables to exclude from interactions, simply copy */

        data L1;
            set ALL;
            if _N_ = &L;
                D = .;
        run;

    %end;

    /* Append dataset L1 to dataset ALL0*/

    proc append base=ALL0 data=L1 force;
    run;

%end;

%let RC=%sysfunc(close(&id)); /* Close dataset ALL to macro processing. This prevents side effects. */

/* Drop observations identified to drop, create KEY variables for sorting */

data ALL0A;
    set ALL0;
    where D ne 1;
    array INDVAR &INDVAR;
    array KEYS KEY1-KEY&NVAR;
    do I = 1 to dim(INDVAR);
        if INDVAR[I] = . then KEYS[I] = '1';
        else KEYS[I] = '2';
    end;
    drop I D;
run;

/* Sort ALL0A for later processing, making, for each set of keys, record with largest value of R^2 first record of
group. */

proc sort data=ALL0A out=ALL0B;
    by KEY1-KEY&NVAR descending _RSQ_;
run;

/* For each set of keys, pull out observation with largest R2: this observation contains all permitted interactions
*/

proc sort nodupkey data=ALL0B out=ALL1;
    by KEY1-KEY&NVAR;
run;

%end;
%else %do;

    /* Straight copy */

```

```

data ALL1;
    set ALL;
    keep &INDVAR _RSQ_;
run;

%end;

/* Compute R2 shares */

proc iml;
    use ALL1;
    read all var{&INDVAR} into INDVAR;
    read all var{_RSQ_} into RSQUARE;
    INDICATORS = INDVAR ^= .;      * If a variable is present in a regression, INDICATORS[I,J] = 1, 0 otherwise.;
    M = nrow(INDVAR);
    N = ncol(INDVAR);
    WEIGHTS = j(M,N,0);
    * Take row sums of INDICATORS: these are the number of independent variables in each regression.;
    MM = INDICATORS[,+];
    MM1 = MM + 1;

    /* Initialize GAMMA weights for efficiency */

    GAMMA = j(N,1,0);
    FACTN = fact(N);      * Make constant for N! ;
    do I = 0 to N-1;
        GAMMA[I+1,] = fact(I) * fact(N - I - 1) / FACTN;
    end;

    /* Build weight matrix */

    do J = 1 to N;
        do I = 1 to M;
            if INDICATORS[I,J] = 1 then WEIGHTS[I,J] = GAMMA[MM[I,],];
            else WEIGHTS[I,J] = -GAMMA[MM1[I,],];
        end;
    end;

    WEIGHTEDR2 = WEIGHTS # shape(RSQUARE,N,M)`;      * Multiply each R^2 by corresponding weight (L&C, 2001);
    R2EXPLAINED = WEIGHTEDR2[+,]`;
    * R^2 explained by each variable is the column sum of the weighted R^2s (transposed for output);
    R2FULL = max(RSQUARE);      * R^2 of full regression is largest R^2 of all regressions;
    R2SHARE = R2EXPLAINED / R2FULL;      * Share of R^2 explained by each variable is its explained R2 divided by R2FULL;
    create R2INTERMEDIATE var{R2EXPLAINED R2SHARE};      * Output results;
    append;

quit;

/* Add variable names to dataset and output */

data &DSOUT;

```

```

        length VAR $32;
        set R2INTERMEDIATE;
        VAR = scan("&indvar",_n_);
run;

proc print data=&DSOUT;
run;

/* Clean up: delete temporary datasets */

proc datasets lib=WORK nolist;
    delete ALL ALL0A ALL0B ALL1 TEMP R2INTERMEDIATE L1 INDATA;
run;

%mend DECOMPOSE_R2;

```