General-to-Specific Modelling (GETS) with User-Specified Estimators and Models

Genaro Sucarrat*

Department of Economics
BI Norwegian Business School

http://www.sucarrat.net/

Toulouse, 10 July 2019

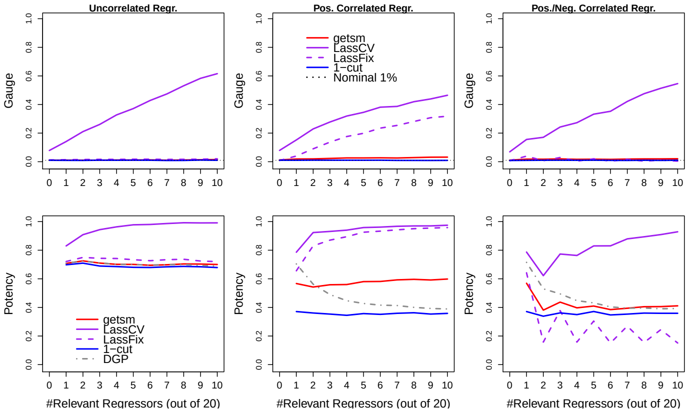
(Last updated: July 10, 2019)

* Based on joint work with Felix Pretis (Univ. of Victoria) and James Reade (Univ. of Reading)

What is General-to-Specific (GETS) modelling?

- Consider the linear regression $y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_{ik} x_{ik} + \epsilon_i$
- Which x's are relevant? That is, which β's are non-zero?
- Which x's are not relevant? That is, which β's are zero?
- GETS modelling combines well-known ingredients in a very well-thought through way. The ingredients are: Backwards elimination (along multiple paths), t-tests of the β's, multiple hypothesis tests of the β's (Wald-tests), goodness-of-fit measures (e.g. information criteria) and diagnostics tests
- The final model: A parsimonious model that contains the relevant variables, and – on average – a proportion of irrelevant variables equal to the regressor significance level α
- GETS modelling thus provides a comprehensive, systematic and cumulative approach to modelling that is ideally suited for conditional forecasting and scenario analysis more generally
- GETS modelling is *not* limited to linear regression
- The *R* package `gets`: provides GETS modelling methods, including the opportunity to user-specify estimators and models

GETS modelling vs. other algorithms



LassCV: Cross-validated Lasso, LassFix: Lasso with fixed penalty, DGP: significance in the DGP itself. Top row shows the false retention rate (gauge), bottom row shows the correct retention of relevant variables (potency). Columns show uncorrelated, positively correlated, and alternating positively and negatively correlated regressors.

Selected reading on GETS modelling:

- Hendry and Richard (1982): "On the Formulation of Empirical Models in Dynamic Econometrics", *Journal of Econometrics*
- Mizon (1995): "Progressive Modeling of Macroeconomic Time Series: The LSE Methodology", in Hoover (ed.) *Macroeconometrics. Developments, Tensions and Prospects*, Kluwer Academic Publishers
- Hoover and Perez (1999): "Data Mining Reconsidered: Encompassing and the General-to-Specific Approach to Specification Search", *Econometrics Journal*
- Hendry and Krolzig (1999): "Improving on 'Data Mining Reconsidered' by K.D. Hoover and S.J. Perez", *Econometrics Journal*
- Campos, Ericsson and Hendry (eds.) (2005): *General-to-Specific Modeling. Volumes 1 and 2*. Edward Elgar Publishing
- Hendry and Doornik (2014): *Empirical Model Discovery and Theory Evaluation*. The MIT Press
- Pretis, Reade and Sucarrat (2018): "Automated General-to-Specific (GETS) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks", *J.Stat.Software*

Why user-specified GETS modelling?

- If coded from scratch, then user-specified implementation of GETS modelling puts a large programming-burden on the user
- Also, GETS modelling is computationally intensive, since many models must be estimated and checked/diagnosed
- We provide a flexible and computationally efficient framework in *R* for the implementation of GETS modelling with user-specified estimators and models:

  – The *R* universe provides an enormous source of potential estimators and models to be used in GETS modelling

  – The user-specified estimators can, in principle, be implemented in external languages (e.g. C/C++, Fortran, Python, Java, Ox, STATA, EViews, MATLAB, etc.)

  – Main function for user-specified GETS: `getsFun`

  – `gets` method (S3), see Example 3:

```
mymodel <- lm(y ~ x)
gets(mymodel) # a gets.lm function applied to 'mymodel'
```

Outline

- GETS modelling in more detail
  - Implementation
  - Model selection properties

- User-specified GETS
  - The `getsFun` function
  - Example 1: Faster OLS (w/`Matrix` package)
  - Example 2: Regression with an ARMA-error (w/`arima`)
  - Example 3: A `gets` method (S3) for `lm`

- Conclusions
  - Summary
  - Outlook

GETS modelling

GETS modelling

Four ingredients:

- Backwards elimination (along multiple paths)
- Coefficient significance testing (individual and joint)
- Fit criteria (e.g. information criteria)
- Diagnostics testing

GETS modelling in 3 steps:

1. Formulate a General Unrestricted Model (GUM). Optional: They should pass the chosen diagnostics tests
2. Backwards elimination of insignificant regressors along multiple paths, while at each regressor removal: a) Test for joint insignificance and b) Check the diagnostics (optional)
3. Choose the best terminal model according to a fit criterion (e.g. an information criterion)

Example

- The starting model (i.e. the estimated GUM):

$$\underset{[p-val]}{y_t} = \underset{[0.07]}{\widehat{\beta}_1} x_{1t} + \underset{[0.02]}{\widehat{\beta}_2} x_{2t} + \underset{[0.26]}{\widehat{\beta}_3} x_{3t} + \widehat{\epsilon}_t$$

  P-values of two-sided t-tests in square brackets

- If we choose a 5% significance level, then deletion along two paths
- Path 1: Start by deleting x_{1t} to obtain

$$\underset{[p-val]}{y_t} = \underset{[0.00]}{\widehat{\beta}_2} x_{2t} + \underset{[0.22]}{\widehat{\beta}_3} x_{3t} + \widehat{\epsilon}_t$$

- Next, deleting x_{3t} gives

$$\underset{[p-val]}{y_t} = \underset{[0.00]}{\widehat{\beta}_2} x_{2t} + \widehat{\epsilon}_t,$$

  i.e. the terminal model of path 1, where the deletion path is $\{x_{1t}, x_{3t}\}$

Example (cont.)

- Recall the starting model (i.e. the estimated GUM):

$$\underset{[p-val]}{y_t} = \underset{[0.07]}{\widehat{\beta}_1} x_{1t} + \underset{[0.02]}{\widehat{\beta}_2} x_{2t} + \underset{[0.26]}{\widehat{\beta}_3} x_{3t} + \widehat{\epsilon}_t$$

- Path 2: Start by deleting x_{3t} to obtain

$$y_t = \underset{[0.03]}{\widehat{\beta}_1} x_{1t} + \underset{[0.00]}{\widehat{\beta}_2} x_{2t} + \widehat{\epsilon}_t$$

  i.e. the terminal model of path 2

- Summarised:

  Path 1 $= \{x_{1t}, x_{3t}\}$ with terminal model $= \{x_{2t}\}$

  Path 2 $= \{x_{3t}\}$ with terminal model $= \{x_{1t}, x_{2t}\}$

- The final model: The best among the terminals according to a fit-criterion, e.g. the Schwarz (1978) information criterion
- In addition: Diagnostics testing and multiple hypothesis testing ("Parsimonious Encompassing Tests") at each deletion (this increases power)

Model selection properties of GETS

Model selection properties of GETS modelling (as $T \to \infty$):

- *All the relevant regressors in the starting model (i.e. the GUM) will be retained in the final model*
- *On average $\alpha \cdot k$ irrelevant regressors will be retained, where α is the chosen significance level for the t-tests*

  Example: Suppose the starting model (i.e. the GUM) is

  $$y_t = \beta_1 x_{1t} + \cdots + \beta_k x_{tk} + \epsilon_t \qquad \text{with } k = 100 \text{ irrelevant regressors}$$

  Choosing $\alpha = 0.10$ means an average of $0.10 \cdot 100 = 10$ irrelevant regressors will be retained

  Choosing $\alpha = 0.05$ means an average of $0.05 \cdot 100 = 5$ irrelevant regressors will be retained

  Choosing $\alpha = 0.01$ means an average of $0.01 \cdot 100 = 1$ irrelevant regressors will be retained

User-specified GETS

The getsFun function

- `getsFun` undertakes GETS modelling with a user-specified estimator/model together with user-specified diagnostics (optional) and user-specified (optional) fit-criteria
- Main arguments:

  `y`: Left-hand side variable

  `x`: Regressor matrix

  `user.estimator`: A `list` containing the name of the user-specified estimator/model and further arguments to be passed on to the estimator
- The function w/three first arguments:

```
getsFun(y, x, user.estimator = list(name="ols",
tol=1e-07, LAPACK=FALSE, method=3), ...)
```

The getsFun function (cont.)

Example: Linear regression

$$y_t = \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + \epsilon_t, \qquad t = 1, \ldots, n$$

Code:

```
library(gets) #load library (if necessary)

n = 40 #number of observations
k = 20 #number of Xs

set.seed(123) #for reproducability
y = 0.1*rnorm(n) #generate Y
x = matrix(rnorm(n*k), n, k) #create matrix of Xs

#do gets w/default estimator (ols):
getsFun(y, x)
```

The getsFun function (cont.)

Some of the output:

```
18 path(s) to search
Searching: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

$time.started
[1] "Wed Jun 26 16:16:47 2019"

$time.finished
[1] "Wed Jun 26 16:16:47 2019"

$no.of.estimations
[1] 308

$paths
$paths[[1]]
[1]  1 15  6  7  3 14 11 16  4  2  8 12  5  9 20 19 13

$paths[[2]]
[1]  2 16 11  7  6 15 14  8  4 12 20 19  1  3  9  5 13
```

The getsFun function (cont.)

```
  .
  .
  .

$paths[[18]]
[1] 20 16  7 15  6 14 11  8  4 19  2 12  1  3  9  5 13

$terminals.results
         info(sc)     logl  n k
spec 1: -2.090464 47.34259 40 3
spec 2: -2.075247 45.19382 40 2

$best.terminal
[1] 1

$specific.spec
[1] 10 17 18
```

The getsFun function (cont.)

The user-specified estimator/model should:

- Be of the form `myEstimator(y, x, ...)`, where `y` is a vector and `x` is a matrix
- Return a `list` with a minimum of six entries:

  `coefficients` (the coefficient estimates)

  `vcov` (the coefficient covariance matrix)

  `df` (degrees of freedom, used for the t-statistics)

  `logl` (a goodness-of-fit value, e.g. the log-likelihood)

  `n` (number of observations)

  `k` (number of parameters)
- The estimator must be able to handle `NULL` regressor-matrices (i.e. `is.null(x)=TRUE` or `NCOL(x)=0`)

The `getsFun` function (cont.)

User-specified diagnostics (optional):

- Use the `user.diagnostics` argument
- The argument should be a `list` with first entry `name="myDiagnosticsFunction"` (say)

User-specified Goodness-of-Fit function (optional):

- Use the `gof.function` and `gof.method` arguments
- The former should be a `list` with first entry `name="myGofFunction"` (say)
- The latter should be either `"min"` (default) or `"max"`

Examples

- Example 1: Faster OLS

  – using the `Matrix` package to build a faster OLS estimator

- Example 2: Regression with an ARMA-error

  – using the `arima` function to automatically search for breaks (location shifts) in a time-series

- Example 3: A `gets` method (S3) for `lm`

  – enables us to to do:

```
mymodel <- lm(y ~ x)
gets(mymodel)
```

Example 1: Faster OLS

- There are packages and routines that can be used to make OLS faster, e.g. the `Matrix` package
- The code below creates a new function, `olsFaster`, which is essentially a copy of `ols(y, x, method=3)` from our gets package, but based on routines from the `Matrix` package
- `microbenchmark` suggests a speed improvement of 10%

The code:

```
library(Matrix)
olsFaster <- function(y, x){
  out <- list()
  out$n <- length(y)
  if (is.null(x)){ out$k <- 0 }else{ out$k <- NCOL(x) }
  out$df <- out$n - out$k
```

Example 1: Faster OLS (cont.)

```
  if (out$k > 0) {
    x <- as(x, "dgeMatrix")
    out$xpy <- crossprod(x, y)
    out$xtx <- crossprod(x)
    out$coefficients <- as.numeric(solve(out$xtx,out$xpy))
    out$xtxinv <- solve(out$xtx)
    out$fit <- out$fit <- as.vector(x %*% out$coefficients)
  }else{ out$fit <- rep(0, out$n) }
  out$residuals <- y - out$fit
  out$residuals2 <- out$residuals^2
  out$rss <- sum(out$residuals2)
  out$sigma2 <- out$rss/out$df
  if (out$k > 0) { out$vcov <- as.matrix(out$sigma2 * out$xtxinv) }
  out$logl <-
    -out$n * log(2 * out$sigma2 * pi)/2 - out$rss/(2 * out$sigma2)
  return(out)
}
```

To run: `getsFun(y, x, user.estimator=list(name="olsFaster"))`

Example 2: Regression with an ARMA-error

Example: Linear regression w/deterministic regressors (X's) and ARMA(1,1)-error

$$\begin{aligned} y_t &= \beta_1 x_{t1} + \cdots + \beta_k x_{tk} + \epsilon_t, \\ \epsilon_t &= \phi_1 \epsilon_{t-1} + \theta_1 u_{t-1} + u_t, \qquad u_t \sim N(0,1) \end{aligned}$$

The Data Generating Process (DGP):

$$y_t = 4 \cdot 1(t \geq 30) + \epsilon_t, \qquad \epsilon_t = 0.4\epsilon_{t-1} + 0.1u_{t-1} + u_t$$

Note: This is a re-parametrisation of an ARMA(1,1) w/location-shift:

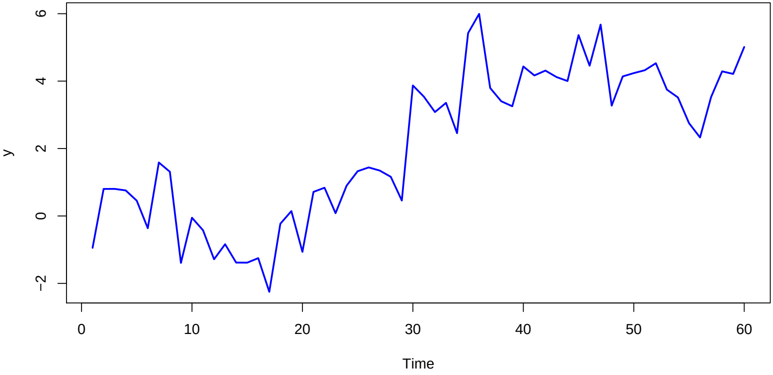
$$y_t = \beta_t^* + \phi_1 y_{t-1} + \theta_1 u_{t-1} + u_t, \qquad y_t = \text{inflation (say)}$$

R code for the DGP:

```
set.seed(123) #for reproducability
eps = arima.sim(list(ar=0.4, ma=0.1), 60) #epsilon
x = coredata(sim(eps, which.ones=25:35)) #11 step-dummies
y = 4*x[,"sis30"] + eps #the dgp
plot(y, col="blue", lwd=2)
```

User-specified GETS: Example 2 (cont.)

The DGP:



Example 2: Regression with an ARMA-error (cont.)

A user-specified estimator (example):

```
myEstimator <- function(y, x){

  tmp = arima(y, order=c(1,0,1), xreg=x)

  #rename and re-organise:
  result = list()
  result$coefficients = tmp$coef[-c(1:3)]
  result$vcov = tmp$var.coef
  result$vcov = result$vcov[-c(1:3),-c(1:3)]
  result$logl = tmp$loglik
  result$n = tmp$nobs
  result$k = NCOL(x)
  result$df = result$n - result$k

  return(result)
}

##try estimator:
myEstimator(y, x)
```

Example 2: Regression with an ARMA-error (cont.)

The GUM (i.e. the general starting model):

$$y_t = \sum_{i=1}^{11} \beta_i \cdot 1_{\{t \geq 24+i\}} + \epsilon_t, \qquad \epsilon_t = \phi_1 \epsilon_{t-1} + \theta_1 u_{t-1} + u_t$$

Do GETS modelling with `myEstimator`:

```
##estimate the gum and then do gets:
getsFun(y, x, user.estimator=list(name="myEstimator"))
```

Example 3: A gets method (S3) for `lm` (cont.)

Aim: To be able to do...

```
mymodel <- lm(y ~ x)
gets(mymodel)
```

Accordingly, we need to make `gets.lm`:

```
gets.lm <- function(object, ...){

  ##make y:
  y <- as.vector(object$model[,1])
  yName <- names(object$model)[1]

  ##make x:
  x <- as.matrix(object$model[,-1])
  xNames <- colnames(x)
```

Example 3: A gets method (S3) for `lm` (cont.)

```
if(NCOL(x)==0){
  x <- NULL; xNames <- NULL
}else{
  if(is.null(xNames)){
    xNames <- paste0("X", 1:NCOL(x))
    colnames(x) <- xNames
  }
}

##is there an intercept?
if(length(coef(object))>0){
  cTRUE <- names(coef(object))[1] == "(Intercept)"
  if(cTRUE){
    x <- cbind(rep(1,NROW(y)),x)
    xNames <- c("(Intercept)", xNames)
    colnames(x) <- xNames
  }
}
```

Example 3: A gets method (S3) for lm (cont.)

```
  ##do gets:
  myspecific <- getsFun(y, x, ...)

  ##which are the retained regressors?:
  retainedXs <- xNames[myspecific$specific.spec]
  cat("Retained regressors:\n ", retainedXs, "\n")

  ##return result
  return(myspecific)

} #close gets.lm function
```

Do gets: `gets(mymodel)`

Conclusions

Summary

- General-to-Specific (GETS) modelling provides a comprehensive, systematic and cumulative approach to modelling ideally suited for conditional forecasting and policy analysis
- User-specified implementation of these methods, however, puts a large programming-burden on the user, and may require substantial computing power
- We develop a flexible and computationally efficient framework for the implementation of GETS methods with user-specified estimators and models:

  – The *R* universe provides an enormous source of potential estimators and models that can be used in GETS modelling

  – Main function for user-specified GETS: `getsFun`

  – The user-specified estimators can, in principle, be implemented in external languages (e.g. C/C++, Fortran, Python, Java, Ox, STATA, EViews, MATLAB, etc.) by letting `getsFun` call functions externally

  – `gets` S3 method

Outlook

- Our software is continuously being maintained and improved
- Some of the items on our 'to do list':
  - User-specified Indicator Saturation (ISAT)
  - Additional ISAT features
  - Simpler parallel computing
  - Faster search and computing
- The developments reflect, to some extent, our research interests – and time!

Thanks!

sucarrat.net/R/gets
https://CRAN.R-project.org/package=gets
github.com/gsucarrat/gets

References:

Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B 57*, 289–300.

Campos, J., D. F. Hendry, and N. R. Ericsson (Eds.) (2005). *General-to-Specific Modeling. Volumes 1 and 2*. Cheltenham: Edward Elgar Publishing.

Doornik, J. (2009). Autometrics. In J. L. Castle and N. Shephard (Eds.), *The Methodology and Practice of Econometrics: A Festschrift in Honour of David F. Hendry*, pp. 88–121. Oxford: Oxford University Press.

Hendry, D. F. and J. Doornik (2014). *Empirical Model Discovery and Theory Evaluation*. London: The MIT Press.

Hendry, D. F., S. Johansen, and C. Santos (2007). Automatic selection of indicators in a fully saturated regression. *Computational Statistics 20*, 3–33. DOI 10.1007/s00180-007-0054-z.

Hendry, D. F. and H.-M. Krolzig (1999). Improving on 'Data Mining Reconsidered' by K.D. Hoover and S.J. Perez. *Econometrics Journal 2*, 202–219.

Hendry, D. F. and H.-M. Krolzig (2001). *Automatic Econometric Model Selection using PcGets*. London: Timberlake Consultants Press.

Hendry, D. F. and J.-F. Richard (1982). On the Formulation of Empirical Models in Dynamic Econometrics. *Journal of Econometrics 20*, 3–33.

Hoover, K. D. and S. J. Perez (1999). Data Mining Reconsidered: Encompassing and the General-to-Specific Approach to Specification Search. *Econometrics Journal 2*, 167–191. Dataset and code: http://www.csus.edu/indiv/p/perezs/Data/data.htm.

Mizon, G. (1995). Progressive Modeling of Macroeconomic Time Series: The LSE Methodology. In K. D. Hoover (Ed.), *Macroeconometrics. Developments, Tensions and Prospects*, pp. 107–169. Kluwer Academic Publishers.

Pretis, F., J. Reade, and G. Sucarrat (2018). Automated General-to-Specific (GETS) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks. *Journal of Statistical Software 86*, 1–44.

Saville, D. (1990). Multiple Comparison Procedures: The Practical Solution. *The American Statistician 44*, 174–180.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics 6*, 461–464.

Sucarrat, G. (2011). *AutoSEARCH: An R Package for Automated Financial Modelling*.

Sucarrat, G. (2014). *gets: General-to-Specific (GETS) Model Selection*. R package version 0.1. http://cran.r-project.org/web/packages/gets/.

Sucarrat, G. and Á. Escribano (2012). Automated Model Selection in Finance: General-to-Specific Modelling of the Mean and Volatility Specifications. *Oxford Bulletin of Economics and Statistics 74*, 716–735.