Covergence Clubs and Regression Trees

0686 - Spatial Economics

Nikolas, Philipp, Lukas & Daniel

Based on Postiglione, Benedetti, and Lafratta (2010)

17 Jänner, 2019

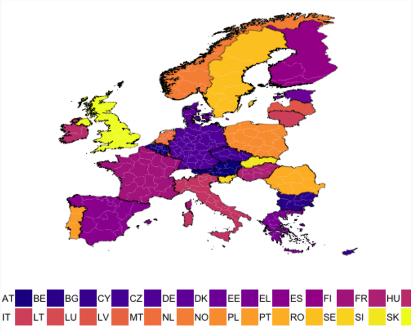
Data Recap

European Regional Database by Cambridge Econometrics

We limit the dataset:

- timeframe 2000-2015
- no Croatia (i.e. two fewer NUTS2 regions)

And use the full set of variables for our 273 regions.



Oh what a merry regression tree

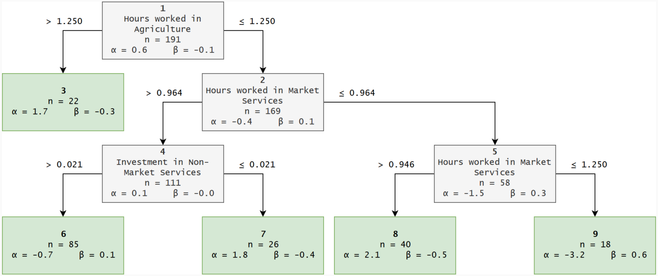
Split observations into clubs:

```
tree <- function(data, split_vars, end_criteria) {
  split <- find_best_split(...)
  if (!end_criteria) {
    return(list(tree(split$data1, ...),
                tree(split$data2, ...)))
  } else { # if(end_criteria)
    return(data)
  }
}
```

Regression Tree

We receive a recursive, tree-like data structure that is:

- hard to deal with (**a lot** of helper functions are necessary)
- nice



Regression Tree

Our results are comparable to partykit (Hothorn and Zeileis 2015).

Still there's the caveat of spatially filtering the data.

Motivation

Results

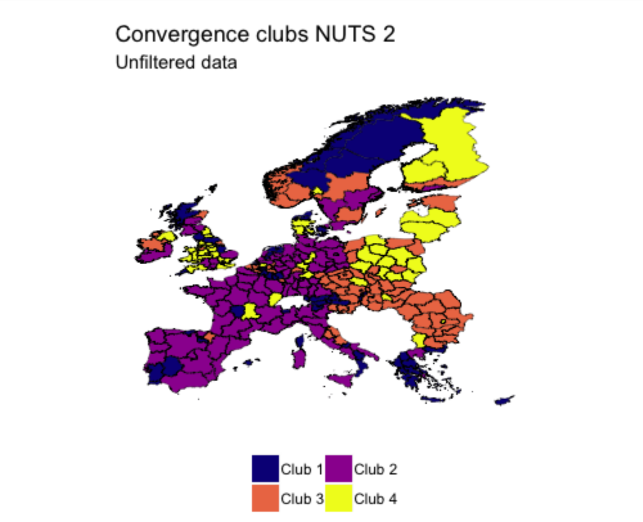
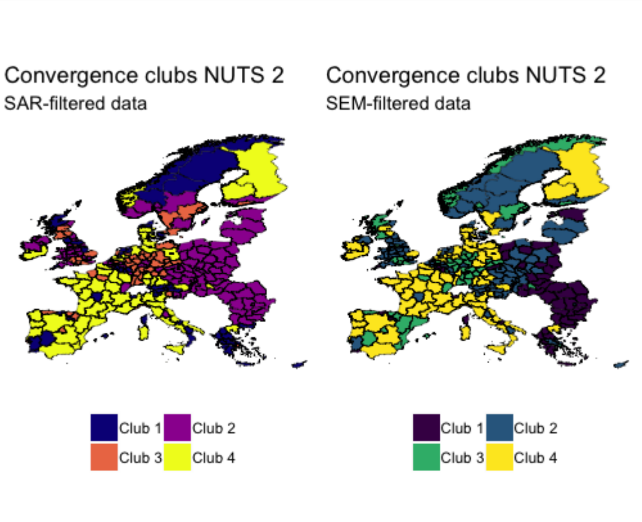

Convergence clubs NUTS 2
Unfiltered data
Club 1
Club 2
Club 3
Club 4

Table 1: Regression results using unfiltered data

	Dependent variable:			
	GDP p.c. growth rate 2000-15			
	(1)	(2)	(3)	(4)
Constant	-1.139^{***}	-0.265	1.769^{***}	2.922^{***}
	(0.323)	(0.360)	(0.146)	(0.147)
Initial GDP p.c.	0.120^{***}	0.035	-0.159^{***}	-0.275^{***}
	(0.032)	(0.036)	(0.016)	(0.015)
Observations	63	92	67	51
Residual Std. Error	0.118 (df = 61)	0.105 (df = 90)	0.129 (df = 65)	0.086 (df = 49)
Note:				*p<0.1; **p<0.05; ***p<0.01

Results


Convergence clubs NUTS 2
SAR-filtered data
Club 1
Club 2
Club 3
Club 4
Convergence clubs NUTS 2
SEM-filtered data
Club 1
Club 2
Club 3
Club 4

Results

Table 2: Regression results using SAR-filtered data

	Dependent variable:			
	GDP p.c. growth rate 2000-15			
	(1)	(2)	(3)	(4)
Constant	-1.174^{***}	1.445^{***}	1.296^{***}	-0.037
	(0.343)	(0.122)	(0.383)	(0.470)
Initial GDP p.c.	0.109^{***}	-0.142^{***}	-0.128^{***}	-0.003
	(0.034)	(0.013)	(0.037)	(0.047)
Observations	63	97	55	58
Residual Std. Error	0.125 (df = 61)	0.124 (df = 95)	0.073 (df = 53)	0.110 (df = 56)
Note:			$^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$	

Results

Table 3: Regression results using SEM-filtered data

	Dependent variable:			
	GDP p.c. growth rate 2000-15			
	(1)	(2)	(3)	(4)
Constant	-0.039^{**}	0.088^{***}	0.016	-0.021
	(0.018)	(0.014)	(0.020)	(0.022)
Initial GDP p.c.	-0.277^{***}	-0.265^{***}	-0.061^{**}	-0.132^{***}
	(0.022)	(0.026)	(0.028)	(0.047)
Observations	55	89	59	70
Residual Std. Error	0.117 (df = 53)	0.120 (df = 87)	0.086 (df = 57)	0.106 (df = 68)
Note:				*p<0.1; **p<0.05; ***p<0.01

- club-plots
- some first LM vs. SAR vs. SEM comparisons

Implementation I

1. Grow the tree

w_tree	list [2]	List of length 2
[[1]]	list [2]	List of length 2
df	list [63 x 18] (S3: data.frame	A data.frame with 63 rows and 18 columns
node	list [1]	List of length 1
[[2]]	list [2]	List of length 2
[[1]]	list [2]	List of length 2
[[1]]	list [2]	List of length 2
df	list [67 x 18] (S3: data.frame	A data.frame with 67 rows and 18 columns
node	list [3]	List of length 3
[[2]]	list [2]	List of length 2
df	list [51 x 18] (S3: data.frame	A data.frame with 51 rows and 18 columns
node	list [3]	List of length 3
[[2]]	list [2]	List of length 2
df	list [92 x 18] (S3: data.frame	A data.frame with 92 rows and 18 columns
node	list [2]	List of length 2
[[1]]	list [3]	List of length 3
pval	double [1]	1.335496e-23
name	character [1]	'emp_ind'
value	double [1]	-0.7057793
[[2]]	list [3]	List of length 3
pval	double [1]	7.152432e-12
name	character [1]	'inv_nms'
value	double [1]	-0.04690463

Implementation II

2. Fell the tree

u_nodes	list [4]	List of length 4
[[1]]	list [1 x 4] (S3: data.frame)	A data.frame with 1 rows and 4 columns
pval	character [1]	'1.33549573594655e-23'
name	character [1]	'emp_ind'
value	character [1]	'-0.70577934346498'
direction	character [1]	'leq'
[[2]]	list [2 x 4] (S3: data.frame)	A data.frame with 2 rows and 4 columns
pval	character [2]	'1.33549573594655e-23' '7.15243206573357e-12'
name	character [2]	'emp_ind' 'inv_nms'
value	character [2]	'-0.70577934346498' '-0.0469046310811765'
direction	character [2]	'gre' 'gre'
[[3]]	list [3 x 4] (S3: data.frame)	A data.frame with 3 rows and 4 columns
pval	character [3]	'1.33549573594655e-23' '7.15243206573357e-12' '2.28459268934336e-08'
name	character [3]	'emp_ind' 'inv_nms' 'emp_nms'
value	character [3]	'-0.70577934346498' '-0.0469046310811765' '-0.311514464909221'
direction	character [3]	'gre' 'leq' 'leq'
[[4]]	list [3 x 4] (S3: data.frame)	A data.frame with 3 rows and 4 columns
pval	character [3]	'1.33549573594655e-23' '7.15243206573357e-12' '2.28459268934336e-08'
name	character [3]	'emp_ind' 'inv_nms' 'emp_nms'
value	character [3]	'-0.70577934346498' '-0.0469046310811765' '-0.311514464909221'
direction	character [3]	'gre' 'leq' 'gre'

Implementation II

```
untree <- function(nodes, simplify = FALSE){
  out <- list()
  lumberjack <- function(nodes){
    # ...
    parent <- parent.frame()
    pos <- length(parent$out) + 1
    # if(...){ ...
    }else if(term_leq){
      parent$out[[pos]] <- leq
      Recall(gre)
    }
  #...} # lumberjack
  lumberjack(nodes)
  return(out)
}
```

Implementation III

3. Plan the furniture

```
> plan
$plan
[1] "emp_ind <= -0.70577934346498"
[2] "emp_ind > -0.70577934346498"
[3] "emp_ind > -0.70577934346498 & inv_nms > -0.0469046310811765"
[4] "emp_ind > -0.70577934346498 & inv_nms <= -0.0469046310811765"
[5] "emp_ind > -0.70577934346498 & inv_nms <= -0.0469046310811765 & emp_nms <= -0.311514464909221"
[6] "emp_ind > -0.70577934346498 & inv_nms <= -0.0469046310811765 & emp_nms > -0.311514464909221"

$terminal
[1] "emp_ind <= -0.70577934346498"
[2] "emp_ind > -0.70577934346498 & inv_nms > -0.0469046310811765"
[3] "emp_ind > -0.70577934346498 & inv_nms <= -0.0469046310811765 & emp_nms <= -0.311514464909221"
[4] "emp_ind > -0.70577934346498 & inv_nms <= -0.0469046310811765 & emp_nms > -0.311514464909221"
```

```
cumPaste <- function(vec, collps = NULL){
  return(sapply(vec, function(x)
    paste(vec[1:which(vec == x)], collapse = collps)))
}
```

Implementation IV

4. Build the furniture

- E.g. for terminal nodes:

dat	list [4]	List of length 4
df1	list [63 x 18] (S3: data.frame	A data.frame with 63 rows and 18 columns
df2	list [92 x 18] (S3: data.frame	A data.frame with 92 rows and 18 columns
df3	list [67 x 18] (S3: data.frame	A data.frame with 67 rows and 18 columns
df4	list [51 x 18] (S3: data.frame	A data.frame with 51 rows and 18 columns

`lapply(dat, ...)` desired regression function

Computational concerns

- Looping
  - For each splitting variable
    - For each value in variable
- Rcpp (Eddelbuettel and Balamuta 2017)

1. Write function in C++

```
// [[Rcpp::export]]
NumericVector get_var_stat(arma::rowvec & y,
                           arma::mat & X, arma::vec & Z,
                           double min_obs){ // ... }
```

2. Source in R

```
Rcpp::sourceCpp("get_var_stat.cpp")
```

Literature

Eddelbuettel, Dirk, and James Joseph Balamuta. 2017. "Extending extitR with extitC++: A Brief Introduction to extitRcpp." *PeerJ Preprints* 5 (August): e3188v1. doi:10.7287/peerj.preprints.3188v1.

Hothorn, Torsten, and Achim Zeileis. 2015. "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research* 16: 3905–9. http://jmlr.org/papers/v16/hothorn15a.html.

Postiglione, Paolo, Roberto Benedetti, and Giovanni Lafratta. 2010. "A Regression Tree Algorithm for the Identification of Convergence Clubs." *Computational Statistics & Data Analysis* 54 (11). Elsevier: 2776–85.