This vignette is distributed with the R package **vcrpart**. Please cite as:
Bürgin, R. and G. Ritschard (2017). "Coefficient-Wise Tree-Based Varying Coefficient Regression with **vcrpart**." *Journal of Statistical Software*, **80**(6), 1–33. DOI 10.18637/jss.v080.i06

Coefficient-Wise Tree-Based Varying Coefficient Regression with vcrpart

Reto Bürgin
NCCR LIVES, Switzerland

Gilbert Ritschard
NCCR LIVES, Switzerland

Abstract

The tree-based TVCM algorithm and its implementation in the R package **vcrpart** is introduced for generalized linear models. The purpose of TVCM is to learn whether and how the coefficients of a regression model vary by moderating variables. A separate partition is built for each potentially varying coefficient, allowing the user to specify coefficient-specific sets of potential moderators, and allowing the algorithm to select moderators individually by coefficient. In addition to describing the algorithm, the TVCM is evaluated using a benchmark comparison and a simulation study and the R commands are demonstrated by means of empirical applications.

Keywords: regression trees, varying coefficient models, generalized linear models, statistical learning, R package, CART.

1. Introduction

When carrying out a regression analysis, researchers often wish to know whether and how moderating variables affect the coefficients of predictor variables. For example, medical scientists may be interested in how age or past illnesses moderate the effect of a clinical trial (e.g., Yusuf *et al.* 1991), and social scientists may examine the wage gap between genders separately for different labor sectors and countries (e.g., Arulampalam *et al.* 2007).

Varying coefficient models (e.g., Hastie and Tibshirani 1993) provide a semi-parametric approach for such moderated relations. Consider a response variable Y, where $g(E(Y|\cdot)) = \eta$, with g a known link function and η a predictor function of form:

$$\mathcal{M}_{\text{vc}} : \eta = X_1\beta_1(\mathbf{Z}_1) + \ldots + X_P\beta_P(\mathbf{Z}_P) \ , \tag{1}$$

where the variables X_p, $p = 1, \ldots, P$, are the P predictors with each having a varying coefficient $\beta_p(\mathbf{Z}_p)$. The vector $\mathbf{Z}_p = (Z_{p1}, \ldots, Z_{pL_p})^T$ associated to coefficient β_p stands for the L_p potential moderator variables specified for that coefficient. Model $\mathcal{M}_{\text{vc}}$ defines each coefficient β_p as a multivariate, nonparameterized function $\beta_p(\mathbf{Z}_p)$ of its associated set of moderators $\mathbf{Z}_p$. For example, if X_p is an indicator variable for some treatment and Z_p reflects the age, the term $\beta_p(Z_p)$ states that the treatment effect changes as a function of age. In principle, the moderator vectors $\mathbf{Z}_1, \ldots, \mathbf{Z}_P$, can have common moderators and can include some of the predictors $X_1, \ldots, X_P$. Model $\mathcal{M}_{\text{vc}}$ also covers two special cases. First, defining $\mathbf{Z}_p$ as a constant, for example by setting $\mathbf{Z}_p \equiv 1$, yields a "non-varying" coefficient for the predictor X_p.

Second, for a constant predictor $X_p \equiv 1$, the coefficient $\beta_p(\mathbf{Z}_p)$ becomes a "varying intercept", and its moderation provides a nonparametric estimate of the direct effects of the moderators $\mathbf{Z}_p$ on $E(Y|\cdot)$.

In varying coefficient regression, exogenous variables can act as predictor, moderator or both. The distinction between predictor and moderator arises naturally from the context. Predictors are the variables for which we are primarily interested to study the impact—coefficient—on the response variable. Moderators are variables that we think can moderate the impact of predictors, i.e., variables with which the coefficients of predictors may vary. Such moderators can serve to distinguish values of coefficients by groups of cases, to model continuous evolution of coefficients, or simply to improve the fit by allowing for interactions and non-linearities. Different examples illustrate the use of moderators in sections 4 and 5. See for example Hayes (2013) for a more in-depth discussion of moderation.

Various approaches have been considered to fit varying coefficient models, in particular with spline or kernel regression methods. See Fan and Zhang (2008) for an overview and the R (R Core Team 2016) packages **mgcv** (Wood 2006), **svcm** (Heim 2007), **mboost** (Hothorn *et al.* 2016), and **np** (Hayfield and Racine 2008) for software implementations. The tree-based approach considered here is a combination of linear models and recursive partitioning (e.g., Quinlan 1992; Alexander and Grimshaw 1996; Loh 2002), where Zeileis *et al.* (2008) and Wang and Hastie (2014) refer explicitly to the use of recursive partitioning to fit models of the form $\mathcal{M}_{\text{vc}}$ (1). Thus, it approximates the unknown varying coefficients with piecewise constant functions using recursive partitioning. The tree-based approach has certain drawbacks, particularly being a heuristic, and can be unstable for small changes in the data. However, it does have several advantages for statistical learning. Among others, the approach can handle many moderators, interactions between moderators, and nonlinearities, it treats moderators of different scales uniformly, and yields easily readable outcomes in the form of decision trees.

Both Zeileis *et al.* (2008) and Wang and Hastie (2014) propose approximating $\mathcal{M}_{\text{vc}}$ as follows: Let $\mathbf{X} = (X_1, \ldots, X_P)^\top$, $\mathbf{Z} = \{\mathbf{Z}_1 \cup \ldots \cup \mathbf{Z}_P\}$, and $\{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$ be a partition of the value space of the $\mathbf{Z}$ into M strata. Then, their piecewise constant approximation has the form

$$\widehat{\mathcal{M}}_{\text{tree}} : \eta = \sum_{m=1}^{M} 1\left(\mathbf{Z} \in \mathcal{B}_m\right) \mathbf{X}^\top \boldsymbol{\beta}_m \ . \tag{2}$$

Model $\widehat{\mathcal{M}}_{\text{tree}}$ (2) is linear and, consequently, standard estimation methods apply. The nonparametric task is to find a partition such that the varying coefficients $\beta_1(\mathbf{Z}), \ldots, \beta_P(\mathbf{Z})$ vary between the strata $\{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$, but are relatively constant within the strata. Since global partitioning is computationally too challenging, forward-stepwise algorithms are used that, in each iteration, split one of the current strata into two. The resulting partition can be visualized as a decision tree and, therefore, the strata $\mathcal{B}_m$ are referred to as terminal nodes, or simply to as nodes.

Here, we introduce the tree-based varying coefficient model (TVCM) algorithm of the R package **vcrpart** (Bürgin 2016). The TVCM algorithm allows us to approximate $\mathcal{M}_{\text{vc}}$ in a "coefficient-wise" manner. First, assuming that K out of the P predictors have varying-coefficients, we let $\mathbf{X}_0$ be the $P - K$ predictors with non-varying coefficients and $X_1, \ldots, X_K$ denote the remaining K predictors with corresponding moderator vectors $\mathbf{Z}_1, \ldots, \mathbf{Z}_K$. Further, we denote the value space of $\mathbf{Z}_k$ as $\mathcal{Z}_k = \mathcal{Z}_{k1} \times \ldots \times \mathcal{Z}_{kL_k}$ and denote a partition of $\mathcal{Z}_k$

into M_k nodes as $\{\mathcal{B}_{k1}, \ldots, \mathcal{B}_{kM_k}\}$. Then, the proposed approximation is:

$$\widehat{\mathcal{M}}_{\text{tvcm}} : \eta = \mathbf{X}_0^\top \boldsymbol{\beta}_0 + \sum_{k=1}^{K} \sum_{m=1}^{M_k} 1\left(\mathbf{Z}_k \in \mathcal{B}_{km}\right) X_k \beta_{km} \ . \tag{3}$$

Compared with $\widehat{\mathcal{M}}_{\text{tree}}$, the TVCM approximation $\widehat{\mathcal{M}}_{\text{tvcm}}$ assigns each varying coefficient a partition and includes non-varying coefficients. This allows us to specify parametrically known relations (the first term) and coefficient-specific sets of moderators (the second term). In addition, $\widehat{\mathcal{M}}_{\text{tvcm}}$ allows us to select moderators individually by varying coefficient. Furthermore, empirical evidence suggests (Sec. 4.1) that $\widehat{\mathcal{M}}_{\text{tvcm}}$ can build more accurate and more parsimonious fits than $\widehat{\mathcal{M}}_{\text{tree}}$ is able to do. A technical difference between the two approximations $\widehat{\mathcal{M}}_{\text{tree}}$ and $\widehat{\mathcal{M}}_{\text{tvcm}}$ is that the coefficients of $\widehat{\mathcal{M}}_{\text{tree}}$ are commonly obtained by fitting a local model on each of the M strata, while the approximation $\widehat{\mathcal{M}}_{\text{tvcm}}$ must be fitted as a closed model on all observations.

The remainder of this paper is organized as follows. In Section 2, we describe the basic algorithm that we apply to generalized linear models. In Section 3, we provide more detail and extend the basic algorithm. The algorithm is evaluated in Section 4 by comparing its performance with competing algorithms on a benchmark data set and its ability to retrieve an underlying generating model through a simulation study. Then, in Section 5, we present two applications. Finally, the concluding Section 6 addresses issues for further development.

2. The TVCM algorithm

Similar to classification and regression trees (CART, Breiman *et al.* 1984), TVCM involves two stages: The first stage (Sec. 2.2) builds K overly fine partitions; the second stage (Sec. 2.3) selects the final partitions by pruning.

To provide a consistent formulation, we restrict our consideration of TVCM to generalized linear models (GLMs). Therefore, Section 2.1 summarizes GLMs and introduces an illustrative example. Extensions to other model families are discussed in Section 3.3.

2.1. Generalized linear models

GLMs cover regression models for various types of responses, such as continuous data (the Gaussian model), count data (the Poisson model), and binary data (the logistic model). Denote the ith response of the training data $\mathcal{D}$ as y_i, with observations $i = 1, \ldots, N$, and the ith $P\times 1$ predictor vector as $\mathbf{x}_i$. Simple GLMs have densities of the form

$$f(y_i|\theta_i, \phi) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi)\right\} \ , \tag{4}$$

where θ_i is the natural parameter of the family, ϕ is the dispersion parameter, and $b(\cdot)$ and $c(\cdot)$ are family-specific functions. For example, the Poisson distribution has density $f(y_i) = \lambda_i^{y_i} e^{-\lambda_i}/k!$ and it can be derived that $\theta_i = \log\lambda_i$, $b(\theta_i) = e^{\theta_i} = \lambda_i$, $\phi = 1$, and $c(y_i, \phi) = \log y_i$. The predictor vector $\mathbf{x}_i$ is incorporated by defining the linear predictor

$$\mathcal{M}_{\text{glm}} : \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta} \ , \tag{5}$$

where $\boldsymbol{\beta}$ is the vector of unknown coefficients. This linear predictor η_i is linked with the conditional mean $\mu_i = E(y_i|\mathbf{x}_i)$ via $g(\mu_i) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$. The choice of $g(\cdot)$ depends on the specific model. A mathematically motivated choice is to specify $g(\cdot)$, such that $\theta_i = \eta_i$, usually called canonical link. For example, for the Poisson model, the canonical is $\log(\mu_i) = \eta_i$. Further details on GLMs can be found, for instance, in McCullagh and Nelder (1989).

Generalized linear models are generally fitted using maximum likelihood estimation (MLE), in other words, by maximizing the total log-likelihood of the training data w.r.t. $\boldsymbol{\beta}$ and ϕ:

$$\ell(\boldsymbol{\beta}, \phi) = \sum_{i=1}^{N} w_i \log f(y_i|\boldsymbol{\beta}, \phi) = \sum_{i=1}^{N} w_i \left(\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi) \right) \quad , \tag{6}$$

where w_i is the weight for observation i. The coefficients $\boldsymbol{\beta}$ enter into Equation 6 via $\theta_i = d(\mu_i) = d(g^{-1}(\mathbf{x}_i^\top \boldsymbol{\beta}))$, with $d(\cdot)$ a known function. To fit GLMs, we use the `glm` function of the **stats** package (see Chambers and Hastie 1992).

Gender gap in university admissions To illustrate R syntax and explanations, we consider the admission data of the UC Berkeley from 1973. The data consist of 4,526 observations on the response variable `Admit` (0 = rejected, 1 = admitted) and the covariates `Female` (0 = male, 1 = female) and `Dept` (departments A to F). The training data `UCBA` are prepared by

```
R> UCBA <- as.data.frame(UCBAdmissions)
R> UCBA$Admit <- 1 * (UCBA$Admit == "Admitted")
R> UCBA$Female <- 1 * (UCBA$Gender == "Female")
R> head(UCBA, 3)
```

```
  Admit Gender Dept Freq Female
1     1   Male    A  512      0
2     0   Male    A  313      0
3     1 Female    A   89      1
```

Each row of the data `UCBA` represents one combination of values in `Admit`, `Female`, and `Dept`. The column `Freq` gives the frequencies of the combinations.

The UCB admission data are a popular application to illustrate Simpson's paradox (see Bickel *et al.* 1975). The primary interest is the gender gap in the chance to be admitted. Let us first study this gap using the logistic regression model:

```
R> glmS.UCBA <- glm(formula = Admit ~ Female, data = UCBA,
+                   family = binomial(), weights = UCBA$Freq)
```

The estimated coefficients,

```
R> summary(glmS.UCBA)$coefficients[, -4]
```

```
            Estimate Std. Error z value
(Intercept)    -0.22     0.0388   -5.68
Female         -0.61     0.0639   -9.55
```

suggest that being female decreases the logit to be admitted significantly ($|z$ value$|> 2$). Now, let us extend the basic model `glmS.UCBA` with the `Dept` covariate by defining department-specific intercepts and department-specific gender gaps (without a global intercept):

```
R> glmL.UCBA <- glm(formula = Admit ~ -1 + Dept + Dept:Female,
+                   data = UCBA, family = binomial(),
+                   weights = UCBA$Freq)
```

```
R> summary(glmL.UCBA)$coefficients[, -4]
```

```
             Estimate Std. Error z value
DeptA           0.492     0.0717   6.859
DeptB           0.534     0.0875   6.097
DeptC          -0.536     0.1149  -4.659
DeptD          -0.704     0.1041  -6.764
DeptE          -0.957     0.1616  -5.922
DeptF          -2.770     0.2198 -12.602
DeptA:Female    1.052     0.2627   4.005
DeptB:Female    0.220     0.4376   0.503
DeptC:Female   -0.125     0.1439  -0.868
DeptD:Female    0.082     0.1502   0.546
DeptE:Female   -0.200     0.2002  -1.000
DeptF:Female    0.189     0.3052   0.619
```

In this second fit, the disadvantage for females disappears, and, in the case of department A, the gender gap is significantly positive (`DeptA:Female`: Estimate $= 1.05$, $|z$ value$|> 2$). The apparent disadvantage for females in `glmS.UCBA` arises, as the reader may know, from the tendency of females to apply to departments where the chances to be admitted are low.

The model `glmL.UCBA`, which uncovers the problem, can be seen as a full parametric varying coefficient model that defines the intercept and the gender gap as functions of the department. We will return to this example to investigate whether and how TVCM solves this problem.

2.2. Partitioning

The first stage to fit the approximate varying coefficient model $\widehat{\mathcal{M}}_{\text{tvcm}}$ (3) involves building a partition for each of the value spaces $\mathcal{Z}_k$, $k = 1, \ldots, K$ corresponding to the K varying coefficients. The resulting K partitions should be overly fine so that the 'optimal' partitions can be found in the subsequent pruning stage. Algorithm 1 provides a formal summary of this partitioning algorithm.

To partition the value spaces $\mathcal{Z}_1, \ldots, \mathcal{Z}_K$, Algorithm 1 uses a breadth-first search (e.g., Russell and Norvig 2003) that in each iteration fits the current model and splits one of the current terminal nodes into two. The split is selected by employing an exhaustive search over a set of candidate splits. These candidate splits are denoted by $\triangle_{kmlj}$ and refer in each case to a partition k; a node m; a moderator variable l; and the cutpoint j in the selected moderator. Algorithm 1 fits for each candidate split a corresponding (approximate) search-model $\widehat{\mathcal{M}}_{kmlj}$ (8) and selects the split that reduces the total $-2 \cdot$ log-likelihood training error the most.[1]

[1] In other words, we maximize the deviance from the current model.

Algorithm 1: The TVCM partitioning algorithm for generalized linear models.

Parameters: N_0 minimum node size, e.g., $N_0 = 30$
D_{min} minimum $-2 \cdot$ log-likelihood reduction, e.g., $D_{min} = 2$

Initialize $\mathcal{B}_{k1} \leftarrow \mathcal{Z}_{k1} \times \ldots \times \mathcal{Z}_{kL_k}$ and $M_k \leftarrow 1$ for all partitions $k = 1, \ldots, K$.

repeat

1 Compute $\hat{\ell}^{\widehat{\mathcal{M}}} = \max_{\{\boldsymbol{\beta}, \phi\}} \ell^{\widehat{\mathcal{M}}}(\boldsymbol{\beta}, \phi)$ of the current model

$$\widehat{\mathcal{M}} : \eta_i = \mathbf{x}_{i0}^\top \boldsymbol{\beta}_0 + \sum_{k=1}^{K} \sum_{m=1}^{M_k} 1\left(\mathbf{z}_{ik} \in \mathcal{B}_{km}\right) x_{ik} \beta_{km} \ . \tag{7}$$

using all observations $i = 1, \ldots, N$.

for *partitions* $k = 1$ *to* K **do**

for *nodes* $m = 1$ *to* M_k *and moderator variables* $l = 1$ *to* L_k **do**

foreach *unique candidate split* $\triangle_{kmlj}$, *in* $\{z_{kli} : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$ *that divides* $\mathcal{B}_{km}$ *into two nodes* $\{\mathcal{B}_{kmlj1}, \mathcal{B}_{kmlj2}\}$ *and satisfies* $\min_s \sum_i w_i 1(\mathbf{z}_{ik} \in \mathcal{B}_{kmljs}) \geq N_0$ **do**

2 Using only the observations $\{i : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$ of the node $\mathcal{B}_{km}$, compute $\hat{\ell}^{\widehat{\mathcal{M}}_{kmlj}} = \max_{\{\beta_1, \beta_2, \phi\}} \ell^{\widehat{\mathcal{M}}_{kmlj}}(\beta_1, \beta_2, \phi)$ of the approximate search model

$$\widehat{\mathcal{M}}_{kmlj} : \eta_i^{(s)} = \hat{\eta}_i + \sum_{s=1}^{2} 1\left(\mathbf{z}_{ik} \in \mathcal{B}_{kmljs}\right) x_{ik} \beta_s \ , \tag{8}$$

and compute the training error reduction $D_{kmlj} = -2\hat{\ell}^{\widehat{\mathcal{M}}} + 2\hat{\ell}^{\widehat{\mathcal{M}}_{kmlj}}_{\{i:\mathbf{z}_{ik} \in \mathcal{B}_{km}\}}$, where $\hat{\ell}^{\widehat{\mathcal{M}}}_{\{i:\mathbf{z}_{ik} \in \mathcal{B}_{km}\}}$ is the subtotal of $\hat{\ell}^{\widehat{\mathcal{M}}}$ for the observations of $\mathcal{B}_{km}$.

3 Split node $\mathcal{B}_{k'm'}$ by $\triangle_{k'm'l'j'}$ where $D_{k'm'l'j'} = \max D_{kmlj}$ and increase $M_{k'} \leftarrow M_{k'} + 1$.

until *no candidate split satisfies* N_0 *or* $D_{k'm'l'j'} < D_{min}$

The used search-model is approximate in that it keeps, in step 2, all parameters but those of the two newly created nodes as fixed, and it uses only the observations of the node to split. The reason for that is given below under 'Computational Details' on page 9. The algorithm iterates until (i) no candidate split provides daughter nodes with more than N_0 observations or (ii) the best split increases the $-2 \cdot$ log-likelihood training error by less than D_{min}.

When searching for a split, there can be differences in the number of candidate splits between partitions, nodes, and potential moderators. The $-2 \cdot$ log-likelihood reduction statistic is not 'standardized' to such differences and, therefore, Algorithm 1 tends to select partitions, nodes, and variables with many candidate splits (cf., Hothorn *et al.* 2006). This selection bias negatively affects interpretability and accuracy of the resulting trees. Reducing this bias is desirable and, therefore, a potential focus for further investigations.

The `tvcglm` function The `tvcglm` function implements Algorithm 1. For illustration, we fit a logistic TVCM to the UCB admission data. The following command specifies that both

the intercept and the gender gap vary across departments.

```
R> library("vcrpart")
R> vcmL.UCBA <-
+    tvcglm(formula = Admit ~ -1 + vc(Dept) + vc(Dept, by = Female),
+           data = UCBA, family = binomial(), weights = UCBA$Freq,
+           control = tvcglm_control(minsize = 30, mindev = 0.0,
+             cv = FALSE))
```

`tvcglm` treats the `data`, `family` and `weights` arguments in the same way as `glm`. Varying coefficients are specified with the '`vc`' operator and a separate partition is fitted for each `vc` term. The `by` argument of the `vc` operator specifies the predictor, while potential moderators are specified before the `by` argument as a comma separated list of variables. When no `by` argument is given, the `vc` term defines a varying intercept. Non-`vc` terms are treated as linear terms, as in `glm`. In the example above, '`vc(Dept)`' specifies a intercept varying with the variable `Dept`, and '`vc(Dept, by = Female)`' a varying coefficient for `Female` that varies with `Dept`. The predictors passed as `by` argument must be numeric in the current implementation. This is why we have defined (page 4) the `Female` variable for the `UCBA` data as '`UCBA$Female <- 1 * (UCBA$Gender == "Female")`'.

The control parameters are set by the `tvcglm_control` function. Above, '`minsize = 30`' specifies $N_0 = 30$ and '`mindev = 0`' specifies $D_{min} = 0$. We set $D_{min} = 0$ to obtain the largest possible tree and '`cv = FALSE`' to disable cross-validation (Sec. 2.3). Note that practice-oriented examples follow in sections 4 and 5, and details on arguments and dummy examples can be found in the help pages of the functions `tvcglm`, `vc` and `tvcglm_control`.

The two fitted partitions are shown in Figure 1, along with the nodewise coefficients. These plots were produced by the following commands:

```
R> plot(vcmL.UCBA, type = "coef", part = "A")
R> plot(vcmL.UCBA, type = "coef", part = "B")
```

As an option,[2], we could display the confidence intervals extracted from the underlying `glm` object. However, these intervals would not account for the model selection procedure and we do not show them here. Both partitions separate the departments fully and, therefore, the values of the coefficients of `vcmL.UCBA` shown in Figure 1 are exactly those obtained for the model `glmL.UCBA` on page 5. The partitioning process can be backtracked using the `splitpath` function. The following command summarizes the first iteration.

```
R> splitpath(vcmL.UCBA, steps = 1, details = TRUE)

Step: 1

Selected Split:
Partition: A
Node: 1
Variable: Dept
```

[2]See the `conf.int` argument of the `panel_coef` function.

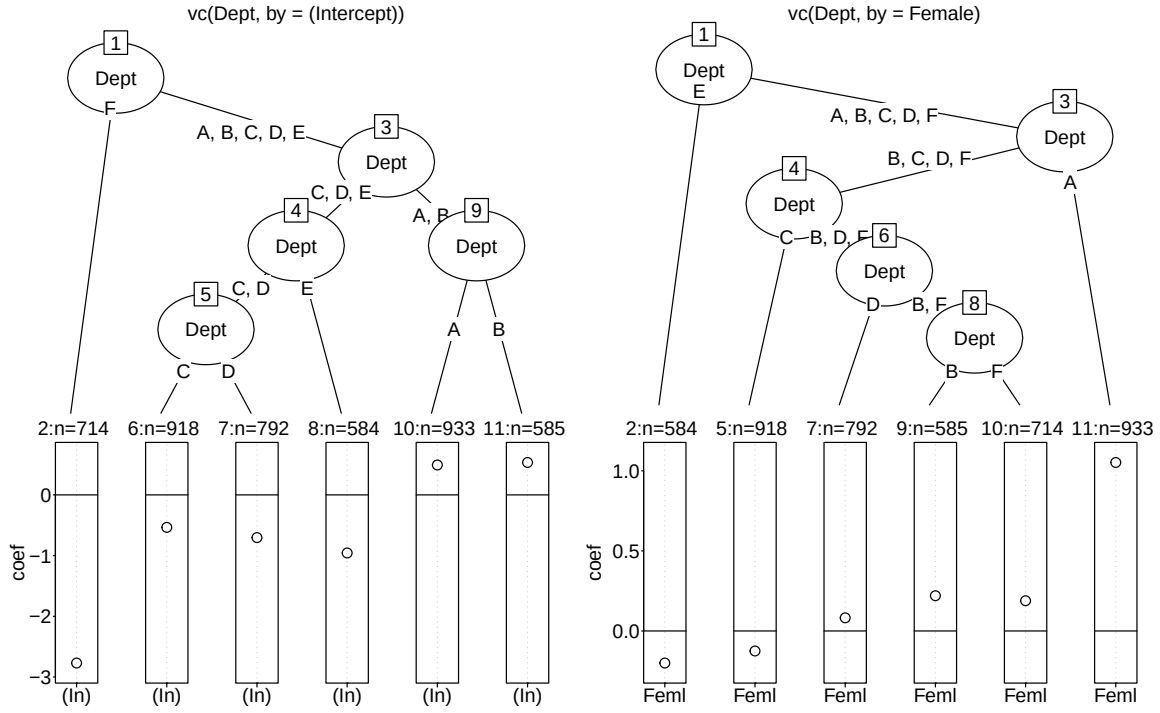

Figure 1: `vcmL.UCBA`: Fitted tree structures and nodewise coefficients. Left panel: The varying intercept. Right panel: The varying gender gap.

```
Cutpoint: {F}, {A, B, C, D, E}

Loss Reduction Statistics:
Partition: A Node: 1 Variable: Dept
  A B C D E F dev npar
1 0 0 0 0 0 1 444    1
2 0 0 0 0 1 1 409    1
3 0 0 0 1 1 1 384    1
4 0 0 1 1 1 1 416    1
5 0 1 1 1 1 1 226    1

Partition: B Node: 1 Variable: Dept
  A B C D E F   dev npar
1 0 0 0 0 0 1 130.6    1
2 0 0 0 0 1 1 124.7    1
3 0 0 1 0 1 1  76.8    1
4 0 0 1 1 1 1  99.3    1
5 0 1 1 1 1 1 121.0    1
```

Based on the training error reduction statistic D_{kmlj} (column `dev`), the algorithm selects the split $\{F\}$ vs $\{A, B, C, D, E\}$ for the varying intercept (partition A). The evaluated splits, listed in the lower part, show that only a subset of possible splits was evaluated. For example, the split $\{A, F\}$ vs $\{B, C, D\}$ was excluded from the search. This relates to the implemented

acceleration technique that orders the six categories A to F and treats the `Dept` as ordinal (details follow).

Computational details

A breadth-first search can be computationally burdensome because it cycles in each iteration through all current nodes. Even so, we do not consider a depth-first search, which is more common for recursive partitioning and which evaluates only one node in each iteration, because it seems unclear whether the search sequence has consequences on the resulting partitions. To speed up the search, we use the approximate search model $\widehat{\mathcal{M}}_{kmlj}$ (8) to compute the training error reduction of split $\triangle_{kmlj}$, instead of using the following accurate search model

$$\widehat{\mathcal{M}}^*_{kmlj} : \eta_i^{(s)} = \mathbf{x}_{i0}^\top \boldsymbol{\gamma}_0 + \sum_{(k',m')\neq(k,m)} 1\left(\mathbf{z}_{k'i} \in \mathcal{B}_{k'm'}\right) x_{k'i}\gamma_{k'm'} + \\ + \sum_{s=1,2} 1\left(\mathbf{z}_{ik} \in \mathcal{B}_{kmljs}\right) x_{ik}\gamma_s \ . \quad (9)$$

The accurate search model, $\widehat{\mathcal{M}}^*_{kmlj}$ (9), requires using all observations $i = 1, \ldots, N$ and re-estimating all coefficients. By contrast, the approximate search model, $\widehat{\mathcal{M}}_{kmlj}$, uses only the observations $\{i : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$ of the node to split and incorporates the fitted values $\hat{\eta}_i$ of the current model $\widehat{\mathcal{M}}$ (7) as offsets. This reduces the optimization per considered split to three parameters, namely the coefficients β_1 and β_2 of the newly created nodes, and the dispersion parameter ϕ, because it cannot be fixed in `glm`. More specifically, the approximate model $\widehat{\mathcal{M}}_{kmlj}$ estimates the coefficients γ_s, $s = 1, 2$ of $\widehat{\mathcal{M}}^*_{kmlj}$ as $\hat{\gamma}_s = \hat{\beta}_{km} + \hat{\beta}_s$, with $\hat{\beta}_{km}$ retrieved from the current model $\widehat{\mathcal{M}}$. Further, in $\widehat{\mathcal{M}}_{kmlj}$ all the remaining coefficients of $\widehat{\mathcal{M}}^*_{kmlj}$, i.e., $\boldsymbol{\gamma}_0$ and $\gamma_{k'm'}$ for $(m', k') \neq (m, k)$, are kept fixed at the values estimated in part 1 (7). In our experience, the approximation is reliable, although it does not necessarily result in the same partitions that the accurate search would produce.[3] In particular, the approximation will tend to neglect splits that improve the fit through interplays with the temporarily fixed coefficients.

Eliminating split candidates and cleverly choosing the stopping parameters are further efficient acceleration techniques. We describe these techniques in more detail here.

Splits for ordered scales In Algorithm 1, the splits $\triangle_{kmlj}$ for continuous and ordinal moderators are defined as rules of the form $\{\text{Is } z_{kli} \leq \zeta_{kmlj}?\}$. The candidate cutpoints, $\{\zeta_{kml1}, \ldots\}$, are the unique values in set $\{z_{kli} : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$. Note that splits at boundaries may be omitted to respect the minimum node size criterion. To reduce the computational burden, we allow the set of candidate cutpoints to shrink to a prespecified cardinality N_S, which is $N_S = 9$ by default.[4] Specifically, the unique values of the (non-interpolated) quantiles of $\{z_{kli} : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$ are extracted at the N_S equidistant probabilities $(1, \ldots, N_S)/(N_S + 1)$. In cases of tied data, where this procedure potentially yields fewer than N_S splits, the number of equidistant probabilities is increased until the set of candidate splits has the desired size.

[3] The approximation can be disabled by setting '`fast = FALSE`' in `tvglm_control()`.

[4] See the `maxnumsplit` and `maxordsplit` arguments in `tvcglm_control`.

Splits for nominal scales The splits $\triangle_{kmlj}$ for nominal moderators are rules of the form $\{\text{Is } z_{kli} \in \zeta_{kmlj}?\}$, where ζ_{kmlj} are merged categories from the set $\{z_{kli} : \mathbf{z}_{ik} \in \mathcal{B}_{km}\}$. The number of unique candidate merges for C categories is 2^{C-1}, which increases exponentially with C. An approximation that restricts the number of splits to be linearly increasing with C determines a category order and then treats the moderator as ordinal. For CART, Breiman *et al.* (1984) propose to deduce such an order from the category-wise averages in the current node. Following this idea, we propose ordering, for each node, the categories by the category-wise estimated coefficients. This reduces the computational expenses to fitting the model that provides the category-wise coefficients, and fitting the at most $C - 1$ models that evaluate the ordinal splits. By default, the approximation is applied for $C \geq 5$.[5]

On page 7, we referred to the category ordering technique when demonstrating the `splitpath` function for the first iteration of partitioning. For instance, for partition B (the gender gap), we used the order $F < E < C < D < B < A$. The rank of a category corresponds to the first row where a '1' appears in the corresponding column. The category-wise coefficients can be estimated by using the model:

```
R> glmCW.UCBA <-
+    glm(formula = Admit ~ 1 + Dept:Female, family = binomial(),
+        data = UCBA,  weights = UCBA$Freq)
```

The model `glmCW.UCBA` substitutes the effect of `Female` of the current model, which is just the model `glmS.UCBA` (page 4), by an interaction term with `Dept` and `Female`. The category ordering is then obtained by ordering the estimated department-specific gender effects.

```
R> sort(coef(glmCW.UCBA)[-1])
```

```
DeptF:Female DeptE:Female DeptC:Female DeptD:Female DeptB:Female
      -2.361       -0.937       -0.440       -0.402        0.974
DeptA:Female
       1.764
```

Internally, our implementation uses an approximation technique to estimate category-wise coefficients, which is analogous to the technique used for approximating the search model $\widehat{\mathcal{M}}^*_{kmlj}$ (9).

Stopping criteria Algorithm 1 applies two stopping criteria. First, to have sufficient observations to estimate the coefficients nodewise, we require a minimum node size N_0. Here, $N_0 = 30$ seems a reasonable rule of thumb value, but can be modified according to the model. Second, to reduce the computational burden, we stop partitioning as soon as the maximal training error reduction falls below D_{min}. Large values of D_{min} yield rougher partitions and require less computation, and vice versa. Therefore, it is crucial to choose D_{min} to be small enough so that the optimal partitions are not overlooked. The default $D_{min} = 2$ was selected based on the forward-stepwise AIC algorithm (e.g., Venables and Ripley 2002), which also requires the total $-2 \cdot$ log-likelihood training error to decrease by at least 2 to continue. In

[5] See the `maxnomsplit` argument in `tvcglm_control`. After the transformation to the ordinal scale, the argument `maxordsplit` controls the effective number of evaluated splits.

Algorithm 2: The TVCM weakest-link pruning algorithm for generalized linear models.

Input: A fitted model $\widehat{\mathcal{M}}$ from Algorithm 1
Parameters: λ: the cost-complexity penalty, $\lambda \geq 0$
repeat
  forall *inner nodes* $\mathcal{B}^*_{kj}$ *of* $\widehat{\mathcal{M}}$, $k = 1, \ldots, K$ *and* $j = 1, \ldots, M_k - 1$ **do**
    Fit the model $\widehat{\mathcal{M}}_{kj}$ that collapses the inner node $\mathcal{B}^*_{kj}$ of $\widehat{\mathcal{M}}$.
    Compute the per-split increase of the training error $\bar{D}_{kj} = \frac{-2\hat{\ell}^{\widehat{\mathcal{M}}_{kj}} + 2\hat{\ell}^{\widehat{\mathcal{M}}}}{\sum_k M_k^{\widehat{\mathcal{M}}} - \sum_k M_k^{\widehat{\mathcal{M}}_{kj}}}$.
  if *any* $\bar{D}_{kj} \leq \lambda$ **then**
    Set $\widehat{\mathcal{M}} \leftarrow \widehat{\mathcal{M}}_{k'j'}$ with $\{k', j'\} = \arg\min_{k,j} \bar{D}_{kj}$
until *all* $\bar{D}_{kj} > \lambda$

our experience, $D_{min} = 2$ is small enough to capture the optimal partition, yet reduces the computational burden considerably. In Section 4.1, we evaluate the impact of N_0 and D_{min} on a real data application.

The `tvcglm_control` function also allows us to control classic tree growth parameters. These parameters, which can include the maximum number of terminal nodes and the maximal depth of the trees, can restrict the complexity of the final model.

2.3. Pruning

The pruning stage selects the final model by collapsing the inner nodes of the overly fine partitions produced by Algorithm 1. In other words, it cuts branches stemming from the same node. Here, we broadly follow the minimal cost-complexity pruning approach of Breiman *et al.* (1984, Chap. 8). Let $\widehat{\mathcal{M}}$ be a fitted model of form Equation 3, where the nodes $\mathcal{B}_{km}$ result from Algorithm 1. Define the cost-complexity error criterion by

$$\text{err}_\lambda(\widehat{\mathcal{M}}) := -2\hat{\ell}^{\widehat{\mathcal{M}}} + \lambda \sum_{k=1}^{K} (M_k^{\widehat{\mathcal{M}}} - 1), \quad \lambda \geq 0 \ . \tag{10}$$

In other words, we define the criterion as the total $-2 \cdot$ log-likelihood training error plus a tuning constant λ multiplied by the total number of splits. Here, λ trades off the in-sample performance and the complexity (i.e., the number of splits) of the model. When minimizing $\text{err}_\lambda(\widehat{\mathcal{M}})$, small choices of λ yield models with many splits, and vice versa. In general, λ is unknown and must be chosen adaptively from the data.

Pruning algorithm Pruning hierarchically collapses inner nodes of the initially overly fine partition to find the model that minimizes $\text{err}_\lambda(\widehat{\mathcal{M}})$, given λ. A global search that collapses multiple inner nodes simultaneously would be computationally too expensive and, therefore, we adopt the weakest link pruning algorithm of (Breiman *et al.* 1984). Algorithm 2 summarizes the implemented algorithm.

Each iteration in Algorithm 2 collapses the inner node that yields the smallest per-split increase in the total $-2 \cdot$ log-likelihood training error. The procedure starts with the model from the partitioning stage and continues until the smallest per-split increase is larger than

λ (i.e., all remaining collapses would increase $\text{err}_\lambda(\widehat{\mathcal{M}})$). The `prune` function implements Algorithm 2. For example, the fit for `vcmL.UCBA` on page 7 is pruned with $\lambda = 6$, as follows.

```
R> vcm.UCBA <- prune(vcmL.UCBA, cp = 6)
```

The pruning algorithm can be backtracked with the `prunepath` function.

```
R> prunepath(vcm.UCBA, steps = 1)
```

```
Step: 1
       part node loss npar nsplit dev
<none>           5167 12   10
1      A     1   5682  7    5     102.95129
2      A     3   5364  8    6      49.19850
3      A     4   5172 10    8       2.31499
4      A     5   5168 11    9       1.17896
5      A     9   5167 11    9       0.13536
6      B     1   5187  7    5       4.04086
7      B     3   5185  8    6       4.39381
8      B     4   5169  9    7       0.56149
9      B     6   5167 10    8       0.08252
10     B     8   5167 11    9       0.00341
```

The above R output provides various information about the first iteration of Algorithm 2, applied on the fit for `vcmL.UCBA`. The columns `part` and `node` identify the collapsed inner node, and `dev` shows the per-split increase of the training error. In the first iteration, the inner node 8 of partition B (the gender gap) yields the smallest $\bar{D}_{kj}$ and is therefore collapsed.

Choosing λ The per-split penalty λ is generally unknown and, hence, must be chosen adaptively from the data. To do so, the validation-set or cross-validation methods are suitable (e.g., Breiman *et al.* 1984) and computationally fast. The validation-set method works as follows. First, divide the training data $\mathcal{D}$ randomly into a subtraining set $\mathcal{D}_1$ and a validation set $\mathcal{D}_2$, e.g., with a ratio of 3 : 1. Second, replicate the fit with Algorithm 1 based on $\mathcal{D}_1$. Third, repeatedly prune the new fit with increasing λ values and compute the validation error each time an inner node is collapsed. This yields two sequences, $\{\lambda_1 = 0, \ldots, \lambda_S, \lambda_{S+1} = \infty\}$ and $\{\overline{\text{err}}_1^{\mathcal{D}_2}, \ldots, \overline{\text{err}}_S^{\mathcal{D}_2}\}$, where $\overline{\text{err}}_s^{\mathcal{D}_2} = \frac{-2}{\sum_{i \in \mathcal{D}_2} w_i} \sum_{i \in \mathcal{D}_2} w_i \log f_{\widehat{\mathcal{M}}}(y_i | \mathbf{x}_i, \mathbf{z}_i)$ is the average[6] prediction error on $\mathcal{D}_2$ of the new model pruned by λ values in interval $[\lambda_s, \lambda_{s+1})$. We retain the estimate for λ,

$$\hat{\lambda} = \frac{\lambda_{s'} + \lambda_{s'+1}}{2} \qquad \text{with} \qquad s' = \arg \min_{s \in \{1,\ldots,S\}} \overline{\text{err}}_s^{\mathcal{D}_2} \ . \tag{11}$$

This is the center of the interval $[\lambda_{s'}, \lambda_{s'+1})$ that minimizes the validation error $\overline{\text{err}}^{\mathcal{D}_2}$. The estimation potentially yields $\hat{\lambda} = \infty$, in which case no split is necessary. Cross-validation methods repeat the validation-set method to include the entire data. In particular, cross-validation combines the obtained sequences $\{\lambda_1, \ldots, \lambda_{S+1}\}$ to a finer grid and averages the errors $\overline{\text{err}}_s^{\mathcal{D}_2}$ accordingly.

[6] We use the average to avoid having the validation error depend on the number of observations in $\mathcal{D}_2$.

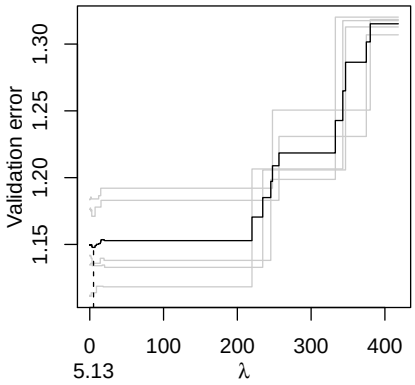

Figure 2: `cv.UCBA`: Validation errors in function of λ from 5-fold cross-validating fits for `vcmL.UCBA`. Black solid line: The cross-validated error. Gray solid lines: The validation errors on individual validation sets. Vertical dotted line: The estimated value for λ.

The `cvloss` function implements the validation-set and the cross-validation methods to estimate $\hat{\lambda}$. By default, 5-fold cross-validation is used. To estimate λ for the `UCBA` data, we use the commands:

```
R> cv.UCBA <- cvloss(vcmL.UCBA,
+                    folds = folds_control(weights = "freq", seed = 13))
```

The argument '`weights = "freq"`' indicates that the weights of `vcmL.UCBA` represent counts rather than unit-specific weights (default). The `seed` argument is used to control the random generator when creating the cross-validation folds, which allows the results to be replicated. If available, the `cvloss` function processes the validation sets parallelized.

The black solid line in Figure 2 shows the per-split penalty λ against the cross-validated error of fits for `vcmL.UCBA`, which is minimal with $\overline{\text{err}}^{\text{cv}}_{28} = 1.148$ at $\hat{\lambda} = 5.1$. The original fit for `vcm.UCBA` can be pruned by $\hat{\lambda} = 5.1$ with the command:

```
R> vcm.UCBA <- prune(vcmL.UCBA, cp = cv.UCBA$cp.hat)
```

The varying coefficients of the model obtained from pruning with $\hat{\lambda} = 5.1$ are shown in Figure 3. Both the varying intercept and the varying gender gap are split into three strata. The final model collapses several departments. For example, in the right panel, we see that the departments B, C, D, and F share the same gender gap. By contrast, the large negative intercept in department F and the large gender gap in department A remain detached.

Alternatives to $\hat{\lambda}$ (11) could be considered. For example, Breiman *et al.* (1984, Chap. 3) propose the 1-SE rule to decrease the variance of $\hat{\lambda}$. We prefer $\hat{\lambda}$ for its simple form, but with `cvloss` and `prune`, we provide the tools to use these alternative rules.

3. Details and extensions

In Section 2, we explained the basic parts of the TVCM algorithm. This section describes the algorithm in more detail and explains how TVCM can be extended to other model classes.

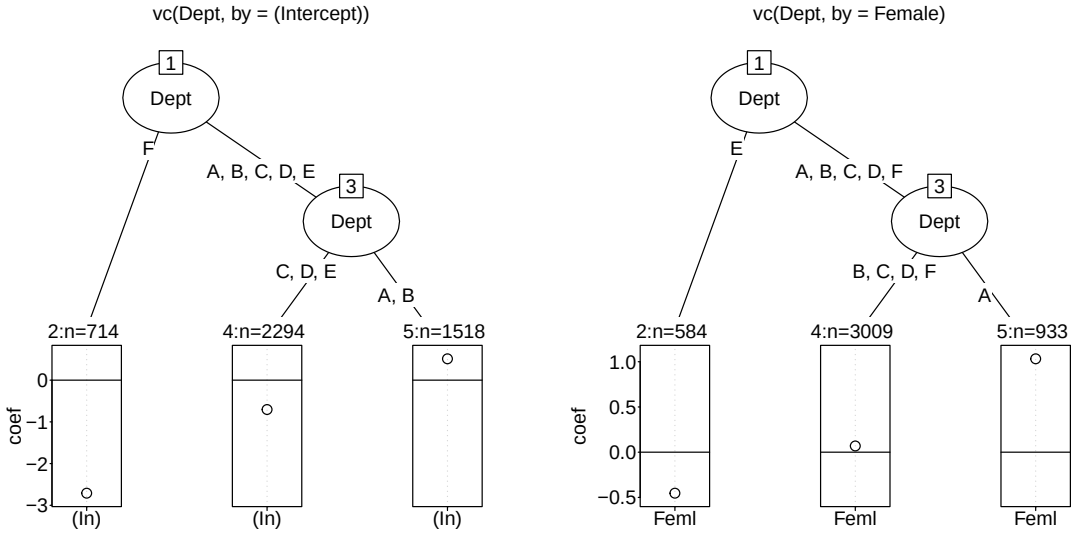

Figure 3: `vcm.UCBA`: Pruned tree structures and nodewise coefficient plots. Left panel: The varying intercept. Right panel: The varying gender gap.

3.1. Mean-centering the predictors of the search model

A useful technique to improve the split selection with the approximate search model $\widehat{\mathcal{M}}_{kmlj}$ (8) is to mean-center its predictors. That is, we substitute the values x_{ik} in Equation 8 with the values $\tilde{x}_{ik} = x_{ik} - N^{-1} \sum_{i=1}^{N} x_{ik}$. The advantage of this is most visible in the case of a pure interaction. Consider Figure 4. In both panels the slope of a predictor x varies between two groups, A and B. In the left panel, the TVCM partitioning algorithm tries to uncover this moderation when x is not centered and the current model specifies a global intercept and a global slope for x. The search model uses the fitted values of the current model (solid line) as offsets and incorporates separate slopes for each group. This restricts the slopes to pass through the origin, and hence the fit (dotted and dashed lines) do not really identify the moderation pattern. The right panel shows that, in this scenario, the moderation pattern is perfectly identified by using the same search procedure, but when x is mean-centered.

The centering trick is applied by default, but can be disabled with the control argument `center`. Note that the output model is not affected by the mean-centering technique, because it is applied only to the search model $\widehat{\mathcal{M}}_{kmlj}$ (8). Further, the trick is not necessary when using the accurate search model $\widehat{\mathcal{M}}^{*}_{kmlj}$ (9) that does not use the offset values.

3.2. Additive expansion of multivariate varying coefficients

An additional feature of TVCM is the possibility to avoid the interactions between moderators by means of an additive expansion of varying coefficients. Although this restriction may prevent the algorithm from finding the very best model, coefficient functions with one moderator may be easier to understand while still performing quite well as will be seen in Section 4.1.

So far, we have implicitly assumed that the predictors $X_1, \ldots, X_P$ of model $\mathcal{M}_{\text{vc}}$ (1) differ

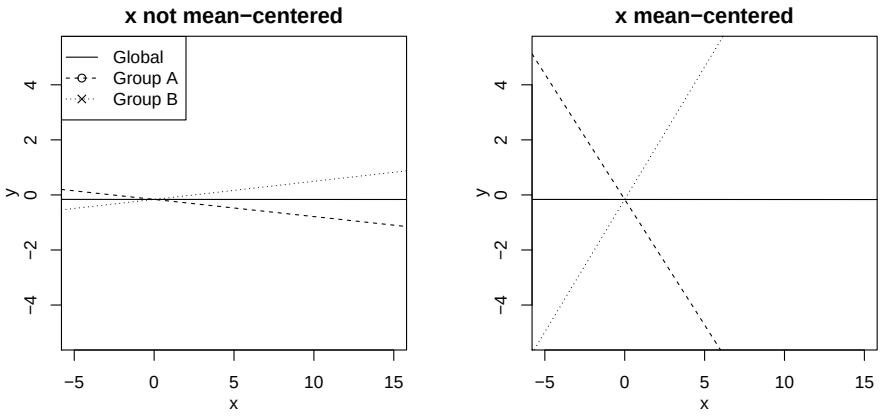

Figure 4: Illustration of the split point search without (left) und with (right) mean-centering the predictors. In this artificial scenario the slope of a predictor x with $\bar{x} = 10$ varies between two groups A (circles) and B (crosses). The solid lines show the slope of the global model and the dotted and dashed lines the group specific slopes of the search model with an intercept set equal to that of the global model.

from one another. To suppress interactions between moderators, we expand the multivariate varying coefficients into additive, moderator-wise components. First, consider a multivariate varying coefficient term $x_{ip}\beta_p(\mathbf{z}_{ip}) = x_{ip}\beta_p(z_{ip1}, \ldots, z_{ipL_p})$, possibly $x_{ip} = 1$ for all i. The additive expansion is

$$x_{ip}\beta(\mathbf{z}_{ip}) \quad \longrightarrow \quad x_{ip}\beta_{p0} + x_{ip}\beta_{p1}(z_{ip1}) + \ldots + x_{ip}\beta_{ipL_p}(z_{ipL_p}) \ . \tag{12}$$

Here, we decompose $x_{ip}\beta(\mathbf{z}_p)$ into the 'isolated' contributions of the individual moderators, including a global term $x_{ip}\beta_{p0}$. In this expansion, the individual varying coefficients $\beta_{pl}(z_{ipl})$, $l = 1, \ldots, L_p$ act as local contributions to the global coefficient β_{p0}. To identify the additive expansion of Equation 12, we mean-center the approximations for $\beta_{p1}(\cdot), \ldots, \beta_{pL_p}(\cdot)$ using node-weighted sum contrasts that restrict the sample-average of the coefficient functions to zero. That is, we approximate $\beta_{pl}(\cdot)$ with the piecewise constant function $\sum_{m=1}^{M_{pl}} 1(z_{ipl} \in \mathcal{B}_{plm})\beta_{plm}$, and estimate the coefficients $\beta_{pl1}, \ldots, \beta_{plM_{pl}}$ subject to

$$\sum_{i=1}^{N} \sum_{m=1}^{M_{pl}} 1(z_{ipl} \in \mathcal{B}_{plm}) w_i \beta_{plm} = 0 \ . \tag{13}$$

The nodewise-weighted sum contrasts are computed with the `contr.wsum` function of **vcrpart**. We also considered extending the additive expansion with second- and higher-order interactions between moderators. However, such an extension likely needs further considerations for the partitioning algorithm.

3.3. Extension to other model classes

To extend the scope of the algorithm, the TVCM requires functions to extract the training and validation errors from fitted models of the considered model class. The training error is required for partitioning (Algorithm 1) and pruning (Algorithm 2), and the need for extracting validation errors arises from cross-validating λ. Both errors refer to loss functions. We

suggest to use the loss function retained for estimating the coefficients, even though different functions could be specified in **vcrpart**. For GLMs, we use as the training error the total $-2 \cdot$ log-likelihood loss on the training sample, which can be extracted from the coefficient estimation. Then, as the validation error, we use the average $-2 \cdot$ log-likelihood loss of predictions on validation sets, which can be extracted using the `predict` and `family` functions. Using the $-2 \cdot$ log-likelihood loss for both the training and validation errors synchronizes the criteria for estimating the coefficients, selecting the split, pruning, and choosing λ. The same, or similar implementation could be considered for other likelihood-based regression models.

The **vcrpart** package also provides implementations for the baseline-category and the cumulative-link models to allow for regression with nominal and ordinal responses. Both these models are multivariate extensions of GLMs (cf., Fahrmeir and Tutz 2001, Chap. 3). Therefore, we can adopt the definitions for the training and validation errors for GLMs (Bürgin and Ritschard 2015).

4. Empirical evaluation

Here, we present two empirical evaluations of the TVCM algorithm. Section 4.1 evaluates the performance of TVCM with different stopping criteria by comparing it with alternative tree-based algorithms on a benchmark data set. Section 4.2 presents a simulation study to assess the ability of TVCM to identify an underlying data generating varying coefficient process.

From here on and unless specifically stated otherwise, the following default control parameters and fixed seed for creating cross-validation folds will be used.

```
R> control <- tvcglm_control(folds = folds_control(seed = 13))
```

4.1. Benchmarking TVCM with the Pima Indians diabetes data

To evaluate the TVCM algorithm, we consider the Pima Indians diabetes data of Smith *et al.* (1988). These data are available from the UC Irvine machine learning repository (Bache and Lichman 2013) and record diabetes tests of 768 Pima Indian women, along with eight covariates. Here, we use the `PimaIndiansDiabetes2` data of the R package **mlbench** (Leisch and Dimitriadou 2010) containing a version of the original data corrected for physical impossibilities, such as zero values for blood pressure. We exclude the two variables `tricepts` and `insulin` and omit cases with missing values of the remaining data by listwise deletion, which is also the default option of `tvcglm`, to avoid expanding the discussion to the missing value problem. The `Pima` data, prepared by the following commands, include 724 observations on the seven variables listed in Table 1.

```
R> library("mlbench")
R> data("PimaIndiansDiabetes2")
R> Pima <- na.omit(PimaIndiansDiabetes2[, -c(4, 5)])
```

For this evaluation, we follow Zeileis *et al.* (2006) and model the response variable `diabetes` using a logistic model with a varying intercept and a varying slope for `glucose` in the predictor function. The remaining covariates 3–7 of Table 1 are used as potential moderators for both varying coefficients. The described model is fitted with the command

	Variable	Label	Scale (Unit)	Range
1	Diabetes	`diabetes`	Binary	Negative, Positive
2	Plasma glucose concentration	`glucose`	Continuous	$[44, 199]$
3	Number of times pregnant	`pregnant`	Continuous	$[0, 17]$
4	Diastolic blood pressure	`pressure`	Cont. $(mmHg)$	$[24, 122]$
5	Body mass index	`mass`	Cont. (kg/m^2)	$[18.2, 67.1]$
6	Diabetes pedigree function	`pedigree`	Continuous	$[0.08, 2.42]$
7	Age	`age`	Cont. (years)	$[21, 81]$

Table 1: Variables of the `Pima` data.

```
R> vcm.Pima.1 <-
+    tvcglm(diabetes ~ -1 + vc(pregnant, pressure, mass, pedigree, age) +
+           vc(pregnant, pressure, mass, pedigree, age, by = glucose),
+           data = Pima, family = binomial(), control = control)
```

where the first `vc` term specifies the varying intercept and the second term specifies the varying slope for `glucose`. We use '`-1`' to remove the global intercept so that the fitted varying intercepts represent local intercepts. Keeping the global intercept would produce the same fit. However, the fitted varying intercepts would represent local contributions to the global intercept. The alternative additive expansion introduced in Section 3.2 is fitted using the command:

```
R> vcm.Pima.2 <-
+    tvcglm(diabetes ~ 1 + glucose +
+           vc(pregnant) + vc(pregnant, by = glucose) +
+           vc(pressure) + vc(pressure, by = glucose) +
+           vc(mass) + vc(mass, by = glucose) +
+           vc(pedigree) + vc(pedigree, by = glucose) +
+           vc(age) + vc(age, by = glucose),
+           data = Pima, family = binomial(), control = control)
```

The additive expansion includes a global intercept and a global slope for `glucose`, which implies that the remaining varying coefficients, which consist of moderator-wise varying intercepts and varying slopes for `glucose`, represent local contributions.

Zeileis *et al.* (2006) fit the same varying coefficient model using the model-based recursive partitioning algorithm (MOB, Zeileis *et al.* 2008), which is based on the single-tree approximation $\mathcal{M}_{\text{tree}}$ (2). First, we compare the fit for `vcm.Pima.1` with the fit based on MOB to discuss the structural differences between the two approximations $\mathcal{M}_{\text{tree}}$ and $\mathcal{M}_{\text{tvcm}}$.

The left panel in Figure 5 shows the fit for `vcm.Pima.1` and the right panel the fit based on the MOB algorithm. Note that the **partykit** plot function generates by default spinograms and not coefficient plots for the leaves of the MOB tree. For the sake of comparison, we have replaced here the default with coefficient plots.[7] The structural difference with MOB is that the TVCM fits separate partitions for the varying intercept and the varying slope for `glucose`, while MOB algorithm fits a common partition for the two varying coefficients. Interestingly, the tree of the varying intercept from the TVCM is identical to the tree from

[7]The code for generating the plot with spinograms is given in the supplementary R-script.

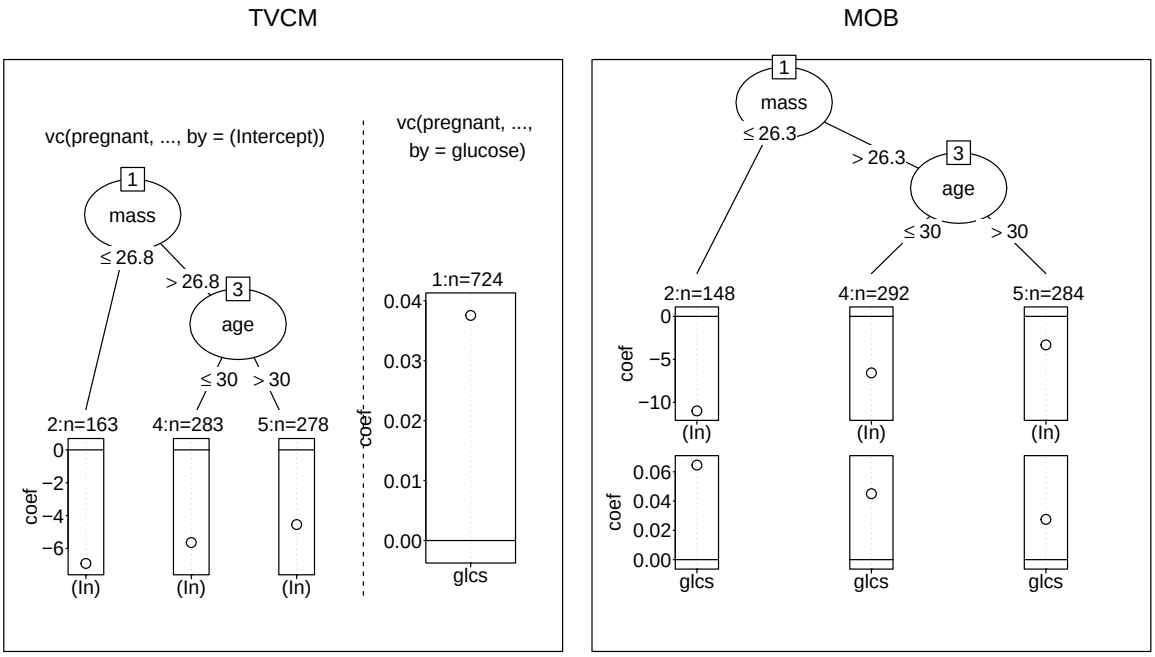

Figure 5: Fitted tree structures and nodewise coefficient plots. Left panel: The varying intercept (left) and the varying slope for `glucose` (right, no split) of the fit for `vcm.Pima.1`. Right panel: The fit based on MOB (cf., Zeileis *et al.* 2006).

the MOB algorithm. By contrast, the TVCM does not retain splits for the slope of `glucose`. This illustrates the flexibility of the TVCM in adapting to situations in which coefficient functions differ. If a single partition for all varying coefficients is accurate, then the TVCM can fit the same partition multiple times. Otherwise, it can tailor the partition individually for each varying coefficient. As a result, the TVCM potentially produces more parsimonious and/or more accurate fits than does the $\mathcal{M}_{\text{tree}}$ approximation.

Comparison with competing tree-based algorithms To evaluate the performance of the TVCM, we extend the benchmark study of Zeileis *et al.* (2006) who, using the same `Pima` data, compare MOB with the conditional inference tree (CTree, Hothorn *et al.* 2006), CART (Breiman *et al.* 1984), the logistic model tree (LMT, Landwehr *et al.* 2005), and C4.5 (Quinlan 1993) algorithms.[8] The MOB and CTree algorithms are implemented in the R **partykit** package (Hothorn and Zeileis 2014) (and **party**), CART in **rpart** (Therneau *et al.* 2015), and LMT and C4.5 in **RWeka** (Hornik *et al.* 2009). We denote CART as RPART and C4.5 as J4.8 because the corresponding R implementations are slightly modified versions.

The performance comparison is made with the `Pima` data and relies on 250 bootstrap samples with replacement (as in Zeileis *et al.* 2006). For each bootstrap sample, we fit a model with each algorithm to predict the excluded observations. In the case of the TVCM, we fit five models on each bootstrap sample to compare the fits for `tvcm.Pima.1` and `tvcm.Pima.2` and to evaluate the sensitivity of fits for `tvcm.Pima.1` to changes from the defaults for N_0 (the

[8]Here, we do not consider the quick, unbiased, efficient, statistical tree algorithm (QUEST Loh and Shih 1997) included in the study by Zeileis *et al.* (2006).

	Misclassification			Complexity		Time	
	Boot	Boot-Mean	Orig	Boot	Orig	Boot	Orig
TVCM	0.246	0.247	0.232	10	6	39.29	25.85
TVCM (additive)	0.241	0.242	0.200	14	18	49.10	22.39
TVCM ($N_0 = 50$)	0.243	0.244	0.225	12	6	12.96	9.25
TVCM ($D_{min} = 50$)	0.249	0.250	0.250	2	2	0.83	0.57
TVCM ($N_S = 19$)	0.246	0.246	0.232	10	6	45.61	50.56
MOB	0.254	0.253	0.238	23	8	1.19	1.00
CTree	0.256	0.258	0.222	19	15	0.03	0.03
RPART	0.260	0.259	0.199	27	15	0.01	0.05
LMT	0.279	0.280	0.222	63	1	0.15	0.85
J4.8	0.279	0.280	0.213	89	11	0.03	0.05

Table 2: Performances for the `Pima` data. Boot: Medians (and means under Boot-Mean) of results from fits on 250 bootstrap samples. Orig: Results on the original data. Misclassification: Misclassification errors. Complexity: The number of coefficients plus the number of splits. Time: Computation time in seconds.

minimum node size), D_{min} (the minimum training error reduction), and N_S (the maximum number of splits). For the competitors, we employ the default control parameters. Three comparison measures are considered: Misclassification, the median and mean 0-1 loss on excluded data; complexity, the median of the number of coefficients plus the number of splits; and time, the median computation time. Furthermore, with each algorithm, we fit a model on the original data. To run the simulation, we use a computer with an Intel Xeon 3.50GHz processor.

Table 2 shows that the TVCM outperforms the competitors in terms of performance and complexity. That is, it builds smaller models with slightly better predictive performance than the other algorithms. By contrast, the TVCM performs worst in terms of computational time because it evaluates far more candidate models than do the competitors. Increasing N_0 and D_{min} accelerates the burden significantly, with surprisingly little effect on the performance. Apparently, in this application, it is not necessary to grow very large trees in the partitioning stage to produce an accurate fit. Furthermore, the difference between the multivariate varying coefficient specification and the additive expansion is negligibly small in this application.

Figure 6 shows averages of 250 pairwise differences between the competitors and the TVCM. The confidence intervals for the averages are based on the Student's t-distribution. Several average differences are significant in favor of the TVCM. It may be that the CTree, RPART, LMT, and J4.8 algorithms perform worse because they merely use piecewise constant regression functions, whereas the TVCM and the MOB algorithm include a (prespecified) slope for `glucose`.

4.2. Simulation study

Here, we use simulated data to evaluate the ability of TVCM to identify an underlying data generating process with coefficients varying across coefficient-wise partitions. In particular, we consider a scenario where the varying coefficients of the data generating model depend on different moderator variables, in which case we would expect coefficient-wise partitioning to perform better than a single partitioning approach.

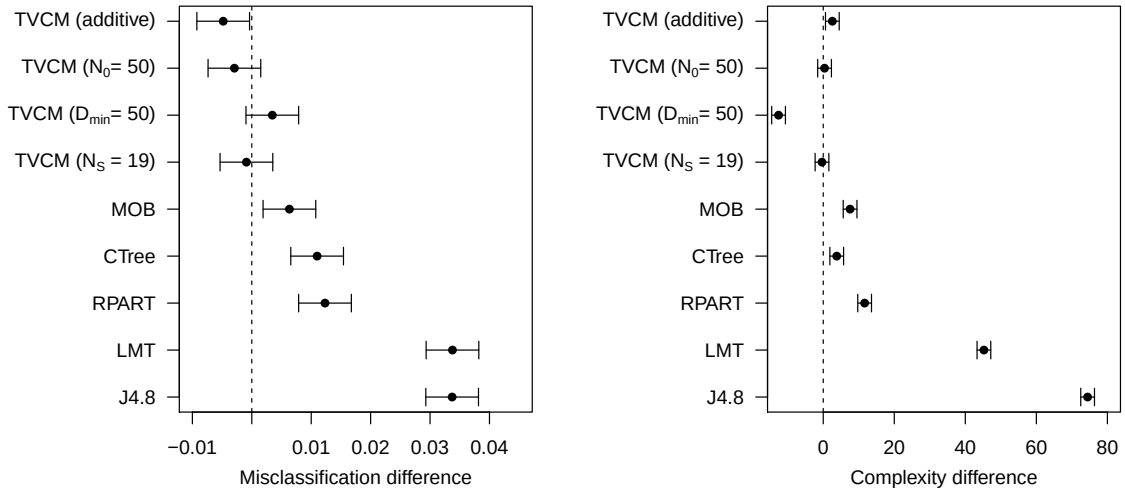

Figure 6: Performances for the `Pima` data relative to TVCM with default control parameters. Left panel: Average differences in misclassification errors. Right panel: Average differences in complexity.

The study consists in generating multiple datasets by means of a simulation model, and fitting a model with the coefficient-wise and a model with the single partitioning approach on each of the datasets. The two approaches are then compared by means of four performance measures. For each dataset, we draw N realizations of a normally distributed predictor variable $X \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$, and N realizations of each of 6 independent Bernoulli moderator variables $Z_0, \ldots, Z_5$ with $Z_l \overset{i.i.d.}{\sim} \text{Bin}(1, 0.5)$. Using these data, we draw N responses $y_1, \ldots, y_N$ from the simulation model:

$$\mathcal{M}_{\text{sim}} : y_i = \left[-1 + 2 \cdot 1_{(z_{0i}>0)}\right] + \left[1 - 2 \cdot 1_{(z_{1i}>0)}\right] \cdot x_i + \varepsilon_i, \quad \varepsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0,1) \tag{14}$$

The model $\mathcal{M}_{\text{sim}}$ (14) has a varying intercept that depends on Z_0, and a varying slope for X that depends on Z_1. The additional four variables X_2 to X_5 act as noise variables that we expect to not be selected. Since both the varying intercept and the varying slope are threshold functions, they can be reconstructed by recursive partitioning. Coefficient-wise partitioning can reconstruct the model with two splits, a first split for the varying intercept in Z_0 and a second split for the varying slope for X in Z_1. The single partition approach needs to perform three splits. Either, it splits the root node by Z_0 and both child nodes by Z_1, or it splits the root node by Z_1 and both child nodes by Z_1. The sample size N is varied by steps of 50 between 50 and 500, and 2000 runs are made for each N, which makes a total of 20,000 simulated datasets.

Both the model with coefficient-wise partitions and the model with a single partition are fitted with the `tvcglm` function. Using for both models `tvcglm` allows to isolate the comparison between the coefficient-wise and single partition approaches from other differences between the algorithms. For example, `glmtree` could be used to fit the single partition model, but then the performance differences would also relate to implementation-specific differences between TVCM and MOB such as the splitting criterion. Apart from computational details, the procedure used by `tvcglm` to fit single partition models is equivalent to that of Wang and

Hastie (2014). Here are the R-commands used to fit the two models, where the data frame `simdata` presents a placeholder for a generated dataset.

```
R> z <- c("z0", "z1", "z2", "z3", "z4", "z5")
R> control.sim <-
+      tvcglm_control(mindev = min(N / 200, 2), minsize = min(N / 20, 30))

R> vcm.sim.1 <- # coefficient-wise partitions model
+      tvcglm(y ~ -1 + vc(z) + vc(z, by = x), data = simdata,
+             family = gaussian(), control = control.sim)

R> vcm.sim.2 <- # single partition model
+      tvcglm(y ~ -1 + vc(z, by = x, intercept = TRUE),
+             data = simdata, family = gaussian(), control = control.sim)
```

As can be seen from the code for `vcm.sim.2`, a single `vc` term in the formula with the arguments '`intercept = TRUE`' and '`by = x`' allows to fit a common partition for the varying intercept and the slope for X. The minimum node size N_0 and the minimum training error reduction D_{min} arguments (cf., Algorithm 1) are specified such that identifying the true model and selecting noise variable remains possible for small sample sizes. For this purpose—to some extent arbitrarily chosen—truncated linear functions are used. For $N = 200$ for example, the retained functions set the minimum node size (`minsize`) as $N_0 = 10$ and the minimum required deviance improvement (`mindev`) as $D_{min} = 1$.

Performance measures To evaluate the fitted models, four frequency measures are considered: (i) Identified underlying model, the proportion of runs for which the exact underlying model was identified; (ii) nested underlying model, the proportion of runs for which the underlying model is nested in the fitted model; (iii) true moderators selected, the proportion of runs for which all true moderators were selected; and (iv) false variable selections, the proportion of runs that selected noise moderators.

Results Figure 7 contrasts the performances of coefficient-wise partitioning to the single partition approach. All four measures tend to a same asymptote for both approaches with, however, a faster convergence for the coefficient-wise partitioning. For our generated data, the chances to identify the exact generating model tends to about 85% when the sample size N becomes large ($N \geq 150$ for coefficient-wise partitioning and $N \geq 300$ for single partitioning), while the chances to end up with a model that includes the generating process as a nested model tends to 1. This shows that the presence of the noise variables leads in some cases to overfitting. Similar conclusions hold for the two other indicators: When N becomes large both approaches select at least all true moderators together with, in 20% of the cases, some noise variables. In sum, the above simulation study indicates that coefficient-wise partitioning is better than single partitioning in retrieving the exact generating process with a sample of moderated size N, while both approaches look equivalent when N becomes large (≥ 350). It also shows that, when $N \geq 200$, the methods tend to select a set of moderators that include at least all true moderators.

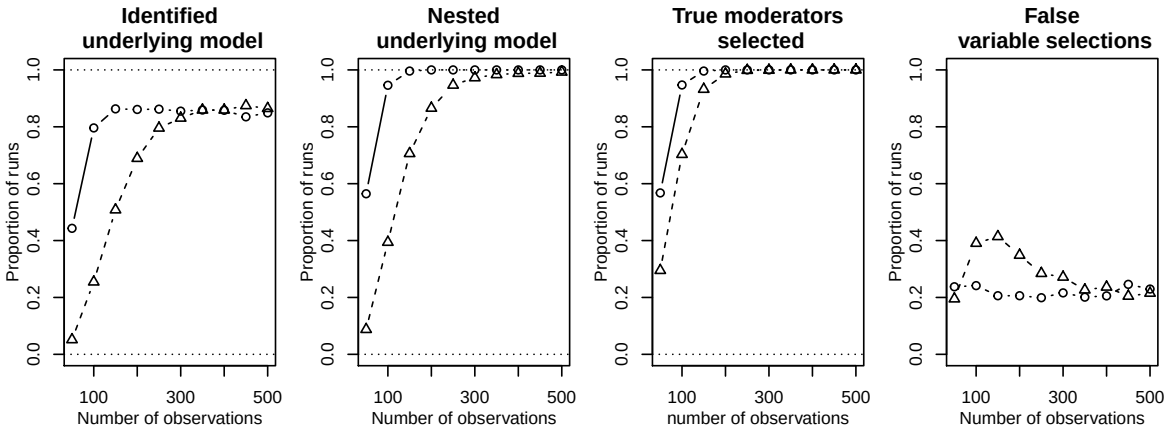

Figure 7: Performance comparison of coefficient-wise partitioning (solid lines) and single partitioning (dashed lines).

5. Applications

We now illustrate the usage of the **vcrpart** library to tackle moderated regression problems in social science research. Two applications are presented. We demonstrate the general multivariate varying coefficient specification in Section 5.1 and the additive expansion in Section 5.2.

5.1. The racial wage gap

The first application is a study of the racial wage gap and, more specifically, a study of whether the wage gap varies across strata. A suitable data set to examine the issue is the `Schooling` data of the R package **Ecdat** (Croissant 2015). The `Schooling` data are a cross-section of 3,010 men prepared by Card (1993) from the 1976 wave of the US National Longitudinal Survey of Young Men (NLSYM).[9] Table 3 describes the variables of the `Schooling` data, where `lwage76` is the response variable and the dummy `black` the predictor of interest.The data were prepared as follows.

```
R> library("Ecdat")
R> data("Schooling")
R> Schooling <- Schooling[c(19, 21, 7, 28, 9, 14, 17, 18, 20, 23, 2, 4)]
R> Schooling$black <- 1 * (Schooling$black == "yes")
```

A standard model for wage is provided by the Mincer equation (Mincer 1974), stating that schooling and working experience are the principal predictors for wage. Therefore, a (Gaussian) linear model that predicts `lwage76` by `ed76`, `exp76` (linear and squared), and `black` seems a suitable base model for the examination of the racial wage gap.

Since the literature (e.g., Ashenfelter and Card 1999) has widely discussed the endogeneity problem in regressing wages on schooling, we implement an instrumental variable (IV) approach using college proximity (`nearc4`) as the instrument for schooling (`ed76`). This instrument, which we computed with

[9] See http://davidcard.berkeley.edu/data_sets.html.

	Variable	Label	Scale (Unit)	Values
1	Logarithm wage per hour 1976	`lwage76`	Cont. (¢/h)	[4.6, 9]
2	Is person black?	`black`	Binary	0=No, 1=Yes
3	Education in 1976	`ed76`	Continuous	[1, 18]
4	Working experience in 1976	`exp76`	Continuous	[0, 23]
5	Age in 1976	`age76`	Cont. (years)	[24, 34]
6	Lived with mom/ dad at age 14?	`momdad14`	Binary	No, Yes
7	Lived in south in 1966?	`south66`	Binary	No, Yes
8	Lived in south in 1976?	`south76`	Binary	No, Yes
9	Mother-father education class	`famed`	Continuous	[1, 9]
10	Enrolled in 1976?	`enroll76`	Binary	No, Yes
11	Lived in smsa in 1976?	`smsa76`	Binary	No, Yes
12	Grew up near 4-yr college?	`nearc4`	Binary	No, Yes

Table 3: The subset of used variables of the `Schooling` data.

```
R> Schooling$ed76.IV <- fitted(lm(ed76 ~ nearc4, Schooling))
```

has been proposed and evaluated by Card (1993). We rely on their evaluation and do not go into detail, because the endogeneity problem is not the point of this illustration.

Using the instrument `ed76.IV` for `ed76`, we fit the intended base model that includes the Mincer equation and the interesting `black` dummy in the predictor function with

```
R> lm.School <- lm(lwage76 ~ ed76.IV + exp76 + I(exp76^2) + black,
+                  data = Schooling)

             Estimate Std. Error t value
(Intercept)  3.90434   0.263306   14.83
ed76.IV      0.16422   0.019645    8.36
exp76        0.05483   0.007184    7.63
I(exp76^2)  -0.00243   0.000351   -6.92
black       -0.31594   0.018065  -17.49
```

The fit reveals that `black` has a significant (t value > 2) negative impact on `lwage76`.

The aim of this application is to illustrate how the TVCM could be used to study moderations on the effect of `black`. To do this, we consider the covariates of 3 to 11 of Table 3 as potential moderators. Furthermore, we want to account for the direct effects of the covariates 5 to 11 on wage, which are those covariates not included in `lm.School`. To integrate these two extensions, we replace the constant coefficient of `black` with a varying coefficient and we replace the global intercept with a varying intercept. However, we continue to assume the Mincer equation and, therefore, use the same specification for the direct effects of `ed76.IV` and `exp76` as in `lm.School`. To fit the described extended model, we use the following formula.

```
R> f.School <- lwage76 ~ -1 + ed76.IV + exp76 + I(exp76^2) +
+    vc(age76, momdad14, south66, south76, famed, enroll76, smsa76) +
+    vc(ed76.IV, exp76, age76, momdad14, south66,
+       south76, famed, enroll76, smsa76, by = black)
```

Then, we fit the varying coefficient model using

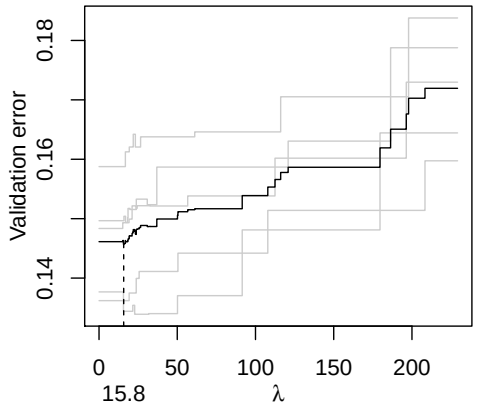

Figure 8: `vcm.School`: 5-fold cross-validated error in function of λ.

```
R> vcm.School <- tvcglm(formula = f.School, data = Schooling,
+                       family = gaussian(), control = control)
```

Figure 8 shows the 5-fold cross-validated error as a function of λ. The estimated $\hat{\lambda} = 15.76$ is almost at the minimum of the evaluated λ values. If we decrease the control parameter `mindev` to 0.5, then $\hat{\lambda}$ will be situated in a clear dump at about 6. Compared to the tree structures below for which $\hat{\lambda} = 15.76$ was used for pruning, $\lambda = 6$ yields additional splits for the varying intercept, yet the same tree structure for the varying coefficient of `black`.

The fitted varying intercept and varying wage gap are shown in Figure 9. The varying intercept on the left consists of 9 terminal nodes. The tree structure suggests that, in particular, `age76` and `smsa76` (standard metropolitan statistical area) have strong direct effects on wage. We do not study the varying intercept in detail because it was mainly implemented to allow the study of the racial wage gap while controlling for the direct effects of the considered variables.

The varying racial wage gap, shown in the right panel of Figure 9, includes three strata. It turns out that the gap is particularly negative for people who live in a southern state and have high working experience. For people who live in the north or with a low working experience (equal or less than 9 years) in the south, the gap is smaller. However, the negative gap remains.

The estimated non-varying coefficients for `ed76.IV` and `exp76` (linear, squared) can be extracted using the `summary` or the `coef` functions, for example, with

```
R> coef(vcm.School)$fe

   ed76.IV      exp76 I(exp76^2)
   0.05723    0.00865   -0.00190
```

5.2. The effect of parental leave duration on return to work

As the last application, we consider an example from the literature on the effects of welfare reforms. Specifically, we investigate the effect of the 1990 reform of the Austrian parental-leave (PL) system. Before 1990, working women had the right to stay off work after childbirth

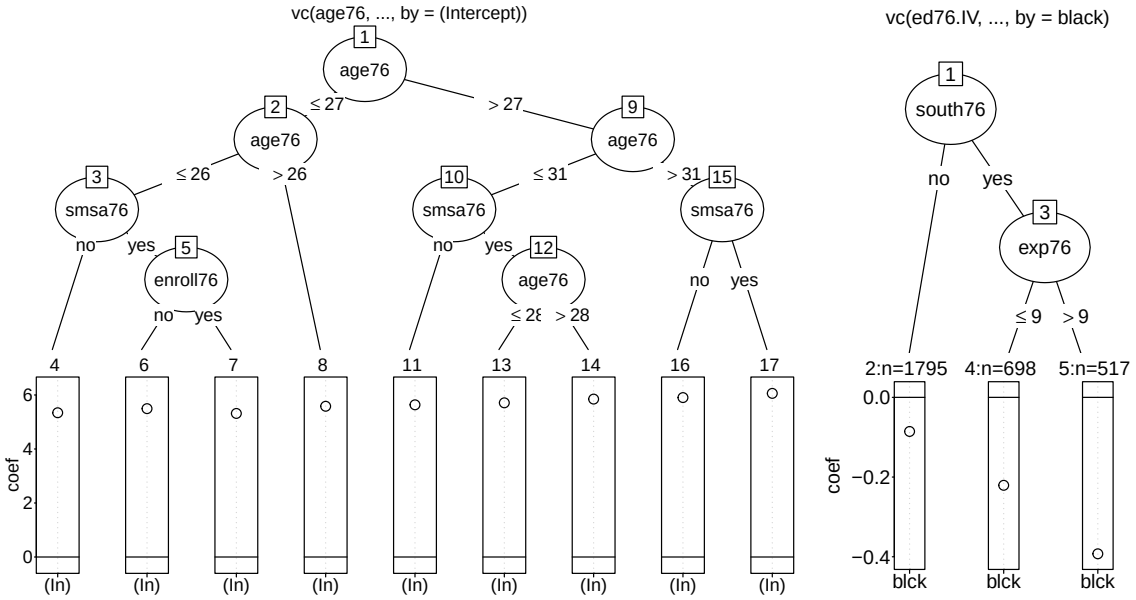

Figure 9: `vcm.School`: Fitted tree structures and nodewise coefficient plots. Left panel: The varying intercept. Right panel: The varying race gap.

up to 12 months and, thereafter, return to the same (or similar) job at the same employer. The 1990 reform extended the duration of this leave from 12 months to 24 months. Lalive and Zweimüller (2009) investigated the effect of the 1990 PL reform on various fertility and labor market outcomes, based on linear regression models and using the Austrian Social Security Database (ASSD). They provide a background to the Austrian PL system and describe the data subset.[10] Here, using the same data, we reanalyze the effect of the reform on whether women returned to work at least once in the 10 years after childbirth.

The subset of the ASSD data includes 6,180 women who gave birth in June or July 1990 and were eligible to access the Austrian PL system. With **vcrpart**, the data are loaded by

```
R> data("PL")
```

The interesting PL reform dummy is `july`. A '0' in `july` means childbirth in June 1990, which is the last month under the old PL system, and a '1' indicates a birth in July 1990, which is the first month under the new PL system. The response variable `uncj10` is binary, where '1' refers to women who returned to work at least once in the 10 years after the considered childbirth. Both `july` and `uncj10` are summarized in Table 4, along with eight further covariates used as moderators.

First, we consider a simple logistic model for `uncj10` with only `july` in the predictor function.

```
R> glm.PL <- glm(uncj10 ~ july, data = PL, family = binomial)
```

[10]The data subset is available from https://sites.google.com/site/rafaellalive/research.

	Variable	Label	Scale	Range
1	Whether returned to work 0-120 months after childbirth	`uncj10`	Binary	0 = No, 1 = Yes
2	Whether childbirth was in July	`july`	Binary	0 = June, 1 = July
3	Years employed before birth	`workExp`	Continuous	[0, 17.5]
4	Years unemployed before birth	`unEmpl`	Continuous	[0, 5.8]
5	Daily earnings at birth	`laborEarnings`	Cont. (€/d)	[0, 1510.7]
6	Whether white collar worker	`whiteCollar`	Binary	no, yes
7	Daily earnings 1989	`wage`	Cont. (€/d)	[0, 98.6]
8	Age	`age`	Ordinal	1, [15−19]; . . . ; 5, [35−44]
9	Industry	`industry.SL`	Nominal	20 categories
10	Region	`region.SL`	Nominal	9 categories

Table 4: The subset of used variables of the `PL` data.

```
            Estimate Std. Error z value
(Intercept)    1.840     0.0535   34.40
july          -0.234     0.0713   -3.28
```

The estimated effect of `july` is -0.23 (corresponding to an odds ratio of $e^{-0.23} = 0.79$) and is significant ($|z \text{ value}| > 2$). This means that the 1990 PL reform decreases the logit for returning to work.

The aim of this application is to investigate whether and how the effect of the PL reform is moderated by covariates 3 to 10 of Table 4, which include for example age and region. Furthermore, we want to study such moderation by considering the direct effects of the moderators. To implement this, we use the additive expansion for multivariate varying coefficients introduced in Section 3.2. The additive expansion is restrictive because it ignores interactions between moderators. However, in applied regression analysis it is common to limit the scope on first-order interactions between the predictor of interest and further covariates (e.g., Cox 1984). For each considered moderator, the intended model adds varying intercepts and varying coefficients for `july` to the simple model `glm.PL`, and is specified by the formula

```
R> f.PL <- uncj10  ~  1 + july + vc(age) + vc(age, by = july) +
+       vc(workExp) + vc(workExp, by = july) +
+       vc(unEmpl) + vc(unEmpl, by = july) +
+       vc(laborEarnings) + vc(laborEarnings, by = july) +
+       vc(whiteCollar) + vc(whiteCollar, by = july) +
+       vc(wage) + vc(wage, by = july) +
+       vc(industry.SL) + vc(industry.SL, by = july) +
+       vc(region.SL) + vc(region.SL, by = july)
```

Note that we keep the global intercept and the global effect of `july` as global references for the individual varying coefficients. The model is fitted with

```
R> vcm.PL <- tvcglm(formula = f.PL, family = binomial(),
+                       data = PL, control = control)
```

Figure 10 shows that the 5-fold cross-validated error is smallest at $\hat{\lambda} = 13.49$. The error curve is relatively flat on the right of the minimum.

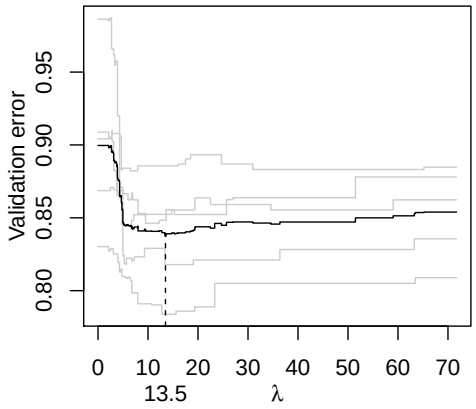

Figure 10: `vcm.PL`: 5-fold cross-validated error in function of λ.

Figure 11 renders the fitted varying coefficients with at least one split. The top row shows the varying intercepts, which are contributions to the global intercept $\hat{\beta}^{(\texttt{Intercept})} = 1.93$, and are interpreted as direct effects on the logits for return to work. The result suggests that women with low working experience (≤ 2.2 years) and a high wage (> 45.8€/d) have increased logits to return to work, and that there are also differences between industries. Specifically, working in an industry corresponding to Node 3, which includes service industries such as social insurance, education and bank industries, has a positive direct effect on return to work. The global effect of the PL reform on the logits for return to work is estimated to be $\hat{\beta}^{\texttt{july}} = -0.23$. The moderation effects of the two selected variables, working experience and region, are shown in the bottom row of Figure 11. From the nodewise coefficient plots, we learn that low working experience (≤ 5.3 years) increases $\hat{\beta}^{\texttt{july}}$ by 0.22, and living in Vienna (W) or Lower Austria (NOe) increases $\hat{\beta}^{\texttt{july}}$ by 0.27. These positive contributions imply that the effect of the PL reform locally surpasses zero, especially for those women who combine the two characteristics low working experience and living in Vienna or Lower Austria.

6. Discussion and outlook

In this study, we showed how to use the TVCM algorithm for varying coefficient regression, as well as its implementation in the R package **vcrpart**. Unlike alternative tree-based algorithms, the TVCM can build a separate partition for each varying coefficient. Thus, it allows us to select moderators individually for each varying coefficient and to specify coefficient-specific sets of moderators. Furthermore, Section 4 provides empirical evidence showing that TVCM builds slightly more accurate and/or more parsimonious models than competing tree-based algorithms. It also assesses the ability of TVCM to identify an underlying data model. In addition to the description of the TVCM, we discussed the model specification, provided R commands and demonstrated the scope of TVCM by applying the algorithm to different data sets.

It is worth mentioning here that TVCM could in some situations wrongly identify a moderator effect in place of a main effect of the variable.[11] Such spurious moderator effects could appear when the moderator has in truth a main effect on the response while the variable is not

[11] Thanks to the reviewer who pointed that out.

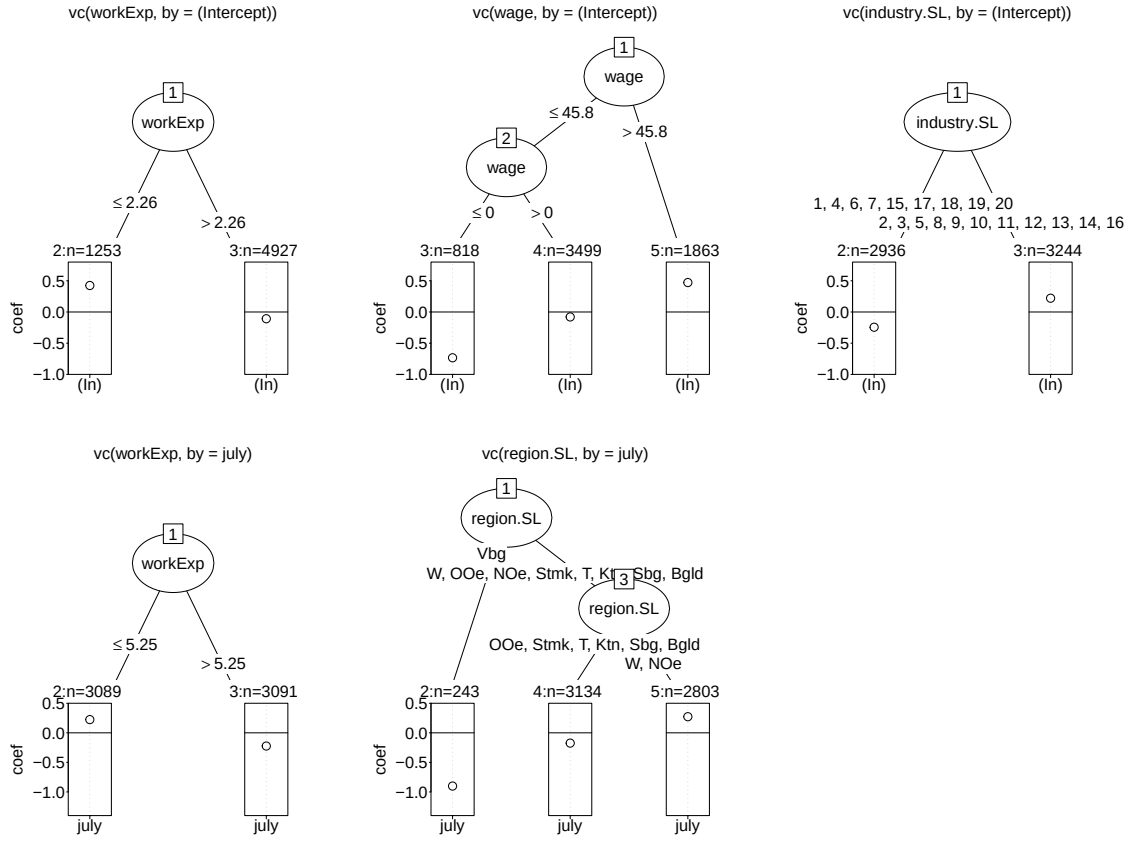

Figure 11: `vcm.PL`: Fitted varying coefficients with at least one split. Top row: The varying intercepts. Bottom row: The varying PL reform effects. The coefficients are contributions to the global intercept $\hat{\beta}^{\texttt{(Intercept)}} = 1.93$ resp. the global PL reform effect $\hat{\beta}^{\texttt{july}} = -0.23$.

specified as a predictor, nor as a moderator of the intercept. Such potential moderators may, when one of the specified predictors remains almost constant, reflect their own main effect on the response through their interaction with that predictor. Therefore, it would be good practice to test, before drawing any definite conclusion, that every identified moderator effect does not change significantly when the moderator is also entered as a predictor.

Further research should investigate the theoretical properties of the TVCM in more detail. This could include simulation studies and/or comparisons with smoothing splines and/or kernel regression techniques, in line with the comparison study of Wang and Hastie (2014). The simulation study described in Section 4.2 is a first attempt in that direction. It suggests that the performance of TVCM for identifying an underlying data model improves with increasing numbers of observations, and that TVCM is more powerful than the single-partition approach in the case where the coefficient functions differ from each others.

There also is potential for improving the TVCM. This could include: (i) Developing an unbiased selection procedure for partitioning; (ii) decreasing the computational time; (iii) refining the pruning criterion; and (iv) stabilizing the algorithm.

Improvement (i) requires finding an alternative criterion that does not tend to select par-

titions, nodes, and moderators with many split candidates (cf., Sec. 2.2). At the outset, we considered implementing the score-based coefficient constancy tests of Zeileis and Hornik (2007), used in the MOB algorithm. We were particularly interested into these tests because they would have allowed to select the partition, the node and the moderator based on the scores of the current model $\widehat{\mathcal{M}}$ (7), without estimating search models. However, we discarded the idea because the tests work under the condition that the predictors of the model are stationary and ergodic (cf., Andrews 1993) with respect to the tested moderator, which seems difficult to control when partitioning coefficient-wise. Another adjustment would be to derive the distribution of the maximally selected likelihood ratio statistics $D_{k'm'l'j'}$ of Algorithm 1. This would allow us to select the split based on p-values, which eliminates the dependence of the selection on the number of splits in the moderator variable. Andrews (1993) developes the distribution of maximally selected likelihood ratio statistics, however, again under the stationarity assumption. Indeed, the stationarity assumption could be resolved by using bootstrap techniques (e.g., Jouini 2008), but such techniques are computationally complex. Finally, F- or χ^2-type tests, such as those proposed in Loh and Shih (1997), could be implemented. For example, Brandmaier *et al.* (2012) implement such tests for building structural equation model trees, and they show that their implementation reduces the variable selection bias substantially.

With regard to point (ii), the TVCM seems more time consuming than the alternative algorithms (cf., Sec. 4.1), although we integrated several acceleration techniques and parallelized the cross-validation. This hindrance, which might be relevant for big data applications, could be partly solved by rewriting the bottleneck functions in a low-level programming language. With regard to improvement (iii), we could consider refining the cost-complexity criterion of Equation 10, which assumes that the 'optimism' of the training error linearly increases with each split. Ye (1998) showed that this assumption is violated for CART, and the same probably applies to the TVCM. Ye (1998) and Efron (2004) provide more accurate solutions using resampling techniques, though these solutions are highly time consuming. Finally, with regard to improvement (iv), to stabilize the algorithm regarding perturbations to the data and to improve the accuracy, we provide with the `fvcglm` function an implementation of the random forest ensemble algorithm (Breiman 2001) for the TVCM. We have not addressed this implementation here so as to focus on the original parts of our work.

Along with `tvcglm`, `tvcglm_control`, `splitpath`, `prunepath`, and `plot`, this study introduced the main functions for the fitting and diagnosis of coefficient-wise tree-based varying coefficient models. Additional functions provided by **vcrpart**, such as `summary` and `predict`, are described in the reference manual.

Acknowledgments

This publication results from research work carried out within the framework of the Swiss National Center of Competence in Research LIVES, which is financed by the Swiss National Science Foundation. The authors are grateful to the Swiss National Science Foundation for its financial support.

References

Alexander WP, Grimshaw SD (1996). "Treed Regression." *Journal of Computational and Graphical Statistics*, **5**(2), 156–175.

Andrews DWK (1993). "Tests for Parameter Instability and Structural Change with Unknown Change Point." *Econometrica*, **61**(4), 821–56.

Arulampalam W, Booth AL, Bryan ML (2007). "Is There A Glass Ceiling Over Europe? Exploring the Gender Pay Gap Across the Wage Distribution." *Industrial and Labor Relations Review*, **60**(2), 163–186.

Ashenfelter O, Card D (eds.) (1999). *Handbook of Labor Economics*, volume 3. Elsevier, Amsterdam, Netherlands.

Bache K, Lichman M (2013). "UCI Machine Learning Repository." URL http://archive.ics.uci.edu/ml.

Bickel PJ, Hammel EA, O'Connell JW (1975). "Sex Bias in Graduate Admissions: Data from Berkeley." *Science*, **187**(4175), 398–403.

Brandmaier AM, von Oertzen T, McArdle JJ, Lindenberger U (2012). "Structural Equation Model Trees." *Psychological Methods*, **18**(1), 71–86.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32.

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984). *Classification and Regression Trees*. Wadsworth, New York, USA.

Bürgin R (2016). ***vcrpart**: Tree-Based Varying Coefficient Regression for Generalized Linear and Ordinal Mixed Models*. R-package version-0.4-1, URL http://cran.r-project.org/web/packages/vcrpart/.

Bürgin R, Ritschard G (2015). "Tree-Based Varying Coefficient Regression for Longitudinal Ordinal Responses." *Computational Statistics & Data Analysis*, **86**, 65–80. doi:10.1016/j.csda.2015.01.003.

Card D (1993). "Using Geographic Variation in College Proximity to Estimate the Return to Schooling." *Technical report*, National Bureau of Economic Research.

Chambers JM, Hastie T (1992). *Statistical Models in S*. Advanced Books & Software. Wadsworth & Brooks/Cole, Pacific Grove, USA.

Cox DR (1984). "Interaction." *International Statistical Review / Revue Internationale de Statistique*, **52**(1), 1–24.

Croissant Y (2015). ***Ecdat**: Data Sets for Econometrics*. R-package version-0.2-9, URL http://CRAN.R-project.org/package=Ecdat.

Efron B (2004). "The Estimation of Prediction Error; Covariance Penalties and Cross-Validation." *Journal of the American Statistical Association*, **99**(467), 619–642.

Fahrmeir L, Tutz G (2001). *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer Series in Statistics, 2 edition. Springer-Verlag, New York, USA.

Fan J, Zhang W (2008). "Statistical Methods with Varying Coefficient Models." *Statistics and Its Interface*, **1**(1), 179–195.

Hastie T, Tibshirani R (1993). "Varying-Coefficient Models." *Journal of the Royal Statistical Society B*, **55**(4), 757–796.

Hayes AF (2013). *Introduction to Mediation, Moderation, and Conditional Process Analysis: A Regression-Based Approach*. Guilford Press, New York, USA.

Hayfield T, Racine JS (2008). "Nonparametric Econometrics: The np Package." *Journal of Statistical Software*, **27**(5), 1–32.

Heim S (2007). "svcm: 2D and 3D Space-Varying Coefficient Models in R." *R package reference manual*, CRAN. 0.1.2, URL http://cran.r-project.org/package=svcm.

Hornik K, Buchta C, Zeileis A (2009). "Open-Source Machine Learning: R Meets Weka." *Computational Statistics*, **24**(2), 225–232.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2016). ***mboost**: Model-Based Boosting*. R-package version-2.6-0, URL http://CRAN.R-project.org/package=mboost.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Hothorn T, Zeileis A (2014). "partykit: A Modular Toolkit for Recursive Partytioning in R." In *Working Papers in Economics and Statistics, Research Platform Empirical and Experimental Economics*, 2014-10. Universität Innsbruck.

Jouini J (2008). "Bootstrap Methods for Single Structural Change Tests: Power versus Corrected Size and Empirical Illustration." *Statistical Papers*, **51**(1), 85–109.

Lalive R, Zweimüller J (2009). "Does Parental Leave Affect Fertility and Return-to-Work? Evidence from Two Natural Experiments." *The Quarterly Journal of Economics*, **124**(3), 1363–1402.

Landwehr N, Hall M, Frank E (2005). "Logistic Model Trees." *Machine Learning*, **95**(1-2), 161–205.

Leisch F, Dimitriadou E (2010). ***mlbench**: Machine Learning Benchmark Problems*. R-package version-2.1-1, URL http://CRAN.R-project.org/package=mlbench.

Loh WY (2002). "Regression Trees with Unbiased Variable Selection and Interaction Detection." *Statistica Sinica*, **12**(2), 361–386.

Loh WY, Shih YS (1997). "Split Selection Methods for Classification Trees." *Statistica Sinica*, **7**, 815–840.

McCullagh P, Nelder J (1989). *Generalized Linear Models*. Monographs on Statistics and Applied Probability, 2nd edition. Chapman and Hall, London, UK.

Mincer JA (1974). *Schooling, Experience, and Earnings*. NBER Books. National Bureau of Economic Research, Inc.

Quinlan JR (1992). "Learning with Continuous Classes." In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific, Singapore.

Quinlan JR (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

R Core Team (2016). *R: A Language and Environment for Statistical Computing, Version 3.3.1*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org.

Russell SJ, Norvig P (2003). *Artificial Intelligence: A Modern Approach*. 3 edition. Pearson Education Inc., New Jersey, USA.

Smith J, Everhart J, Dickson W, Knowler W, Johannes R (1988). "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus." In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261–265. American Medical Informatics Association.

Therneau T, Atkinson B, Ripley BD (2015). ***rpart**: Recursive Partitioning and Regression Trees*. R-package version-4.1-10, URL http://CRAN.R-project.org/package=rpart.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Statistics and Computing, 4 edition. Springer-Verlag, New York.

Wang JC, Hastie T (2014). "Boosted Varying-Coefficient Regression Models for Product Demand Prediction." *Journal of Computational and Graphical Statistics*, **23**(2), 361–382.

Wood S (2006). *Generalized Additive Models: An Introduction with R*. Texts in Statistical Science. Chapman and Hall, Boca Raton, USA.

Ye J (1998). "On Measuring and Correcting the Effects of Data Mining and Model Selection." *Journal of the American Statistical Association*, **93**(441), 120–313.

Yusuf S, Wittes J, Probstfield J, Tyroler HA (1991). "Analysis and Interpretation of Treatment Effects in Subgroups of Patients in Randomized Clinical Trials." *Journal of the American Medical Association*, **266**(1), 93–98.

Zeileis A, Hornik K (2007). "Generalized M-Fluctuation Tests for Parameter Instability." *Statistica Neerlandica*, **61**(4), 488–508.

Zeileis A, Hothorn T, Hornik K (2006). "Evaluating Model-Based Trees in Practice." In *Research Report Series*, 32. Wirtschaftsuniversität Wien. URL http://epub.wu.ac.at/1484/1/document.pdf.

Zeileis A, Hothorn T, Hornik K (2008). "Model-Based Recursive Partitioning." *Journal of Computational and Graphical Statistics*, **17**(2), 492–514.

Affiliation:

Reto Bürgin
Swiss National Center in Competence in Research LIVES
Südbahnhofstrasse 2
CH-3007 Bern, Switzerland
E-mail: rbuergin@gmx.ch

Gilbert Ritschard
Swiss National Center in Competence in Research LIVES
IDESO, Centre Acacias 4
Route des Acacias 54
CH-1227 Carouge, Switzerland
E-mail: Gilbert.Ritschard@unige.ch