Modelling long term survival with non-proportional hazards

Perperoglou, A.

Chapter 3

A fast routine for fitting Cox models with time varying effects

Abstract

The `S-plus` and `R` statistical packages have implemented a counting process setup to estimate Cox models with time varying effects of the covariates. The data set has to be re-arranged in a repeated measurement setting: the time is divided into small time intervals where a single event occurs and for each time interval, the covariate values and outcome in the interval for each subject still under observation are stacked to a large data set. This is the known (Tstart,Tstop] algorithm implemented in Therneau's Survival library (`S-plus`) which has been ported into an `R` package by Thomas Lumley.
However, the expansion of a data set leads to a larger set which can be hard to handle even with fast modern computers.
We propose the use of a fast and efficient algorithm, written in `R`, which works on the original data without the use of an expansion. The computations are done on the original data set, with significant less memory resources used. This improves the computational time by orders of magnitude. The algorithm can also fit reduced-rank Cox models with time varying effects.
We illustrate the method on a large data set of 2433 breast cancer patients, a smaller study of 358 ovarian cancer patients, and compare the computational times on simulated data of up to 10,000 cases with `SAS proc phreg` and `survival` package in `R`. For larger data sets our algorithm was several times faster, and was able to handle larger data sets then `SAS` and `R`.

3.1 Introduction

Cox proportional hazards model [24] has become the most common method to analyze time to event data. Consider information on a set of covariates as an $n \times p$ matrix X, where n is the number of cases and p is the number

of covariates, and let X_i a row vector of covariates for individual i. The Cox model specifies the hazard as

$$h(t|X_i) = h_0(t)\exp(X_i\beta) \tag{3.1}$$

where $h_0(t)$ is the unknown *baseline hazard*, and β is a $p \times 1$ vector of coefficients. Since the model assumes that the hazard ratio between two subjects with fixed covariates is constant it is also known as *proportional hazards* model.

To fit the model a conditional, or *partial likelihood* is used. Data are sorted according to their unique failure times (assuming that no ties are present) and at the time just prior to an event, say t_i, the set of individuals still at risk form the *risk set*, denoted by R_i. The partial likelihood is given as a product of the conditional probabilities, given an event, that individual i is the one that failed at time t_i. To estimate the model a Newton-Raphson algorithm is used, where the definition of the risk sets at each event time is an essential concept. The creation of the risk sets is done internally in most of the popular statistical software that fit proportional hazards models. However, it could also be done externally, by creating explicitly the risk set at each event time and stack them all together in a new data set. Given that, the estimation of a Cox model, is essentially the same as fitting a conditional logistic regression model stratified on the different risk sets.

In many medical studies, individuals are monitored throughout their follow up period, during which, the values of some covariates may change. For example, a covariate may record repeated measurements of blood pressure of a patient, or a transplant indicator variable. Variables whose values change over time are known as time dependent covariates. When time dependent covariates are present the creation of an expanded data set with stacked risk sets is essential. In such a setting, the covariate is denoted as $X_i(t)$ and equation 3.1 is extended to

$$h(t|X_i(t)) = h_0(t)\exp(X_i(t)\beta)$$

To fit this model one need values of $X(t)$ at all event times.

A different extension of the Cox model is the Cox model with time varying coefficients. Assume the effect of a covariate that changes over time, such as the effect of a treatment that might wash away. In this case the covariate itself is fixed but its effect is allowed to vary through time, and thus leading to *time varying effects* of the covariates. In such cases model 3.1 can be extended to

$$h(t|X_i) = h_0(t)\exp(X_i\beta(t)) \tag{3.2}$$

3.1. *Introduction*

The time varying effect for covariate j could be modeled by

$$\beta_j(t) = \sum_{k=1}^{q} \theta_{jk} f_k(t), j = 1, ..., p$$

where $f_k(t)$ is a function of time with $k = 1, ..., q$ and θ_{jk} is the coefficient for the k-th time function on covariate j.

This model is usually fitted using software for time dependent covariates by introducing $p \times q$ pseudo time dependent covariates $X_j f_k(t)$. However, since these pseudo time dependent covariates vary continuously through time this is a computational challenge for large data sets. For example consider data on three patients (id=1,2,3) with information on one covariate (x with values x_1, x_2, x_3) and assume that we want to model the interaction of that covariate with three basis time functions $f_1(t), f_2(t)$ and $f_3(t)$. To include that interaction, every function of time must be evaluated at every subject's failure time, not just the time for the current subject. Using the command `expand.breakpoints` in `R` would lead to a data set of the following form:

`id`	`start`	`stop`	`status`	`X`	F_1	F_2	F_3
1	0	t_1	1	x_1	$f_1(t_1)$	$f_2(t_1)$	$f_3(t_1)$
2	0	t_1	0	x_2	$f_1(t_1)$	$f_2(t_1)$	$f_3(t_1)$
2	t_1	t_2	1	x_2	$f_1(t_2)$	$f_2(t_2)$	$f_3(t_2)$
3	0	t_1	0	x_3	$f_1(t_1)$	$f_2(t_1)$	$f_3(t_1)$
3	t_1	t_2	0	x_3	$f_1(t_2)$	$f_2(t_2)$	$f_3(t_2)$
3	t_2	t_3	1	x_3	$f_1(t_3)$	$f_2(t_3)$	$f_3(t_3)$

and a model with time varying effects could be fitted with the call:

```
coxph(Surv(start,stop,status)~X+X:F1+X:F2+X:F3)
```

The creation of an expanded data set with stacked risk sets can be avoided here, since these covariates are known and "predictable" at time $t = 0$. Thus, the creation of the risks sets can be done internally in order to save memory and computing time.

The purpose of this Chapter is to describe an efficient computation algorithm written in `R`, that can fit Cox models with time varying effects of the covariates, with the definition of the risk sets been done internally, without the use of an expanded data set. This saves memory, and decreases computation time by orders of magnitude. The routine can be adapted very easily to fit reduced- rank hazard models as described by Perperoglou, le Cessie and van

Houwelingen [77]. The structure of the Chapter is as follows: the basic theory will be briefly reviewed in Section 3.2, followed by a brief description of reduced-rank models. In Section 3.4 the software details will be discussed and in Section 3.5 we will present applications to real and simulated data. The Chapter closes with a discussion.

3.2 Cox model with time varying effects of the covariates

Assume information on p covariates and n cases that form the $n \times p$ matrix of covariates X, sorted on their survival/censoring time $t_1, t_2, ..., t_n$ and for simplicity assume that there are no ties present. Also assume that the effects of covariates may vary over time, and let F be the $n \times q$ matrix of time functions with $F_{ik} = f_k(t_i)$. Then, at event times t_j, model 3.2 can be written as

$$h(t_j|X_i) = h_0(t_j)\exp(X_i \Theta F_j') \tag{3.3}$$

where Θ is the matrix of unknown regression coefficients, which we call the *structure* matrix, and X_i and F_j denote the i-th row of X and j-th row of F, respectively. This notation is very useful, especially for representing reduced-rank models which are presented in Section 3.3. For example, in a simple case where there is information on two covariates x_1, x_2 which interact with one time function $f(t) = t$ model 3.3 is written as:

$$h(t|X) = h_0(t)\exp(\theta_{11}x_1 + \theta_{12}x_1 t + \theta_{21}x_2 + \theta_{22}x_2 t).$$

The partial likelihood is given by:

$$L(\Theta) = \prod_{\text{event times}} \frac{\exp(X_i \Theta F_i')}{\sum_{j \in R_i} \exp(X_j \Theta F_i')}$$

The numerator of the partial likelihood depends only on information of the case that experiences an event, while the denominator contains the information of all cases in the *risk set* R_i, that is the set of cases j that have observation time (censored or not) $t_j \geq t_i$. To maximize the likelihood, a Newton-Raphson algorithm is used. Define Θ_v =vec(Θ), which is a vectorization -by column- of the structure matrix. The score function is then given by:

$$U(\Theta_v) = \frac{\partial \ln(L(\Theta))}{\partial \Theta_v} = \sum_{\text{event times}} (X_i - \bar{X}_i(\Theta)) \otimes F_i$$

where $\otimes$ is the kronecker product and

$$\bar{X}_i(\Theta) = \sum_{j \in R_i} X_j \exp(X_j \Theta F_i') / \sum_{j \in R_i} \exp(X_j \Theta F_i')$$

the mean of covariate vectors X_j in risk sets R_i , weighted by $\exp(X_j\Theta F_i')$. Then the information matrix is given by

$$I(\Theta_v) = -\frac{\partial \ln(L(\Theta))}{\partial \Theta_v^2} = \sum_{\text{event times}} C_i(\Theta) \otimes (F_i F_i')$$

with $C_i(\Theta)$ the covariance matrix of covariate vectors X_j in risk sets R_i, weighted again by $\exp(X_j\Theta F_i')$. The covariance of Θ_v is given by inverting the information matrix.

Our software uses the score functions and the information matrix to maximize the likelihood via a Newton-Raphson algorithm. By using the kronecker function the algorithm is very fast in finding the values of the coefficients that maximize the likelihood. To estimate the standard errors of the coefficients one has to invert the information matrix and compute the square root of diagonal values. At the end of the fitting algorithm the baseline hazard can be estimated as:

$$\hat{H}_0(t) = \sum_{t_i \le t} \frac{d_i}{\sum_{j \in R_i} \exp(X_j \Theta F_i')}$$

This is the known Breslow estimator given for covariate values $X = 0$.

3.3 Reduced-Rank Hazard Regression

Perperoglou, le Cessie and van Houwelingen [77] introduced the idea of reduced-rank regression to survival analysis with time varying coefficients. A reduced-rank model requires the matrix of regression coefficients Θ, as given in equation 3.3, to be of reduced-rank r, smaller than the number of covariates p and the number of time functions q. This can be achieved by writing the structure matrix as a product of two submatrices, $\Theta = B\Gamma'$, with B of size $p \times r$ and Γ of size $q \times r$. This factorization results in a rich class of models. When the rank=1 the model assumes that all time varying effects are common and shared among the covariates, resulting in a very parsimonious model. On the other hand, when r =min(p,q) the structure matrix is of full rank, and thus giving the saturated model, identical to the one in equation 3.3.

In their original paper, the authors proposed an alternating algorithm for estimating the model, which can be slightly modified here, so it can be fitted using the proposed routine in this Chapter. Starting with some random initial values of B and Γ, the scores and information matrix are estimated. Observe that the derivatives of Θ with respect to B and Γ are given by:

- $\frac{\partial \Theta_\nu}{\partial B_\nu} = T_B(\Gamma) = \Gamma \otimes I_p$, a $(p*q) \times (p*r)$ matrix given by the kronecker product of Γ with an identity matrix $I_{(p \times p)}$

- $\frac{\partial \Theta_\nu}{\partial \Gamma_\nu} = T_\Gamma(B) = I_q \otimes B$, a $(p*q) \times (q*r)$ matrix given by the kronecker product of B with an identity matrix $I_{(q \times q)}$.

Define B_ν =vec(B) and Γ_ν =vec(Γ'). At the first step, the B coefficients are updated by solving

$$\frac{\partial \ln(L(\Theta))}{\partial B_\nu} = T_B(\Gamma)' \frac{\partial \ln(L(\Theta))}{\partial \Theta_\nu} = 0$$

with second derivatives

$$\frac{\partial^2 \ln(L(\Theta))}{\partial B_\nu^2} = T_B(\Gamma)' \frac{\partial^2 \ln(L(\Theta))}{\partial \Theta_\nu^2} T_B(\Gamma)$$

Once the values of B have been updated the algorithm alternates to the estimation of Γ, by solving:

$$\frac{\partial \ln(L(\Theta))}{\partial \Gamma_\nu} = T_\Gamma(B)' \frac{\partial \ln(L(\Theta))}{\partial \Theta_\nu} = 0$$

where the matrix of the second derivatives is

$$\frac{\partial^2 \ln(L(\Theta))}{\partial \Gamma_\nu^2} = T_\Gamma(B)' \frac{\partial^2 \ln(L(\Theta))}{\partial \Theta_\nu^2} T_\Gamma(B)$$

The whole process alternates between the two steps until the likelihood stabilizes. The estimation of standard errors for a reduced-rank model is somewhat more complicated since the information matrix used in the fitting procedure is not of full rank. Furthermore, B and Γ are not identifiable. However, Θ is identifiable, but restricted by the rank$(\Theta) = r$. Hence, the covariance matrix of vec$(\hat{\Theta})$ is singular and it can be obtained from cov(vec$(\hat{\Theta})) = D(D'I(\Theta)D)^{-1}D'$, where

$$D = \begin{pmatrix} \frac{\partial \Theta_\nu}{\partial B_\nu} \\ \frac{\partial \Theta_\nu}{\partial \Gamma_\nu} \end{pmatrix}$$

This covariance matrix can be used to obtain confidence bands for the varying effects.

3.4 Description of the software

The code is written in `R` (version 2.0.1)[81], and the package `coxvc_1-0-1.zip` and `coxvc_1-0-1.tar.gz` is available for download from the first authors website at
`http://clinicalresearch.nl/personalpage`. It is also possible to use the functions on `S-plus` (it was tested on `S-plus 6` for Windows [51]) with a few modifications. The basic structure can be seen graphically in figure 3.1. The code is divided into four component functions. On top of the figure, the main function `coxvc` is the starting point to define the formula and the data. Then, depending on the rank of the model, one of the functions `full.fit` or `rr.fit` is used. Within either of these functions, the details of the model are defined and then a fourth function `sumevents` computes the risks sets and essential parts of the score functions and information matrix.

In a bit more detail, the command line is:
`coxvc(formula, Ft, rank, iter.max=30, data=sys.parent())`
We explain here the basic arguments:

`formula:` A formula object, with the response on the left of a '$\sim$' operator, and the terms on the right. The response must be a survival object as returned by the 'Surv' function.

`Ft:` A numeric matrix containing the values of the time functions. The first column must be constant.

`rank:` Specifies the rank of the reduced-rank model. The maximum rank cannot exceed the minimum of the number of covariates or time functions.

`iter.max:` Maximum number of iterations, default is 30.

`coxvc` is the main function that is used for the definition of the variables and the model. Once the model has been specified the function calls either `full.fit` -the function for fitting a full rank model- or `rr.fit`, a function for fitting reduced-rank models.

For full rank models the code `full.fit` is simple and rather straightforward. The command line of the function is:
`full.fit(eventtimes,X,Ft,theta,iter.max,n,p,q)` with all the parts defined in the body of `coxvc`. Some explanation is given here:

`eventtimes:` Follow up time only where an event occurs.

`theta:` The structure matrix of coefficients.

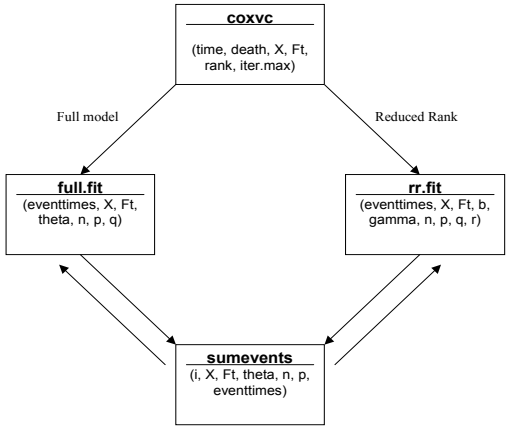

Figure 3.1: Data flow diagram of the `coxvc` program.

`n:` Number of cases.

`p:` Number of covariates.

`q:` Number of time. functions

The function calls function `sumevents` in which the definition of the risk sets takes place, along with the calculation of parts of the score functions and the second derivatives for event time i. Function `sumevents` is called in an `sapply` command and the results are returned as a list that has to be unlisted extracting the right components used in the scoring algorithm. The body of `full.fit` has a `while` loop in which the scoring takes place. At the end of the procedure the information matrix is solved to provide standard errors of the estimates. The output contains the estimated coefficients of the Θ matrix, their standard errors, and the value of the log likelihood.

For reduced-rank models, the function `rr.fit` is called which is very similar to the one used for full rank models. The command line is:
`rr.fit(eventtimes,X,Ftime,theta,iter.max,n,p,q,r)`
in which the rank r of the model has to be defined. This function divides the scoring algorithm in two steps, one for the estimation of B -first step- and a second step for the estimation of Γ. The `while` loop that contains the scoring comes to an end when the difference of the two different likelihoods from each step is very small. At the end of the procedure $\text{cov}(\hat{\Theta})$ is computed. In the case of reduced-rank models a generalized inverse matrix is needed (see end of Section 3.3). We use the function `ginv` from the package `MASS` [102]. For `S-plus` users this should be replaced with `ginverse`. The output of `rr.fit` contains the matrices B, Γ and Θ, a matrix of standard errors and the final likelihood of the model.

In practice only function `coxvc` has to be used in order for the model to be defined. The function prepares the data in the appropriate form and calls one of the internal functions to estimate the model. The returned value at the end is an object of a class `coxvc` for the full model, or `coxrr` for the reduced-rank model. The package contains routines for summarizing and printing the models fitted by `coxvc`. Note that since the package was build mainly for fitting reduced-rank models, all the covariates are supposed to interact with the same time functions. However, when the model is of full rank it would be useful to allow one covariate to interact with say, a linear function of time, and another covariate to interact with another function. This could be achieved by imposing some restrictions on the estimation method, and forcing some of the γ's to be zero. It is the authors' intention to provide an update to the present package just for that reason.

3.5 Applications

Survival of breast cancer patients

From 1981 up to 2002, 2433 women with operable breast cancer were treated in IASO Woman's Hospital, in Athens Greece. All of these cases had either a mastectomy or a breast conserving surgery, and the diagnosis of breast cancer was confirmed by histological examination after surgery. More than half of the cases (1405 women) had received a full circle of 6-month chemotherapy following the operation, 1814 patients were radiated to the area of the chest and axilla to prevent a local recurrence, and 1810 cases received hormonal treatment, which consisted of 20mg of Tamoxifen each day, for at least five

Table 3.1: Characteristics of 2433 breast cancer patients. T represents tumour size (T1=0-20mm, T2=21-50mm, T3=>50mm), Ln is the number of positive lymph nodes, and G represents tumour gradind (Bloom-Richardson classification, GI=grade 1, GII= grade 2, GIII= grade 3)

	n	%
T1	1155	47.5
T2	1123	46.1
T3	155	6.4
Ln-	1137	46.7
Ln+_{1-3}	484	19.8
Ln+_{4-9}	428	17.6
$\text{Ln+}_{>9}$	384	15.8
GI	331	13.6
GII	1470	60.4
GIII	632	26.0

years after surgery. The mean age of the patients was 56 years (from 23 up to 98) and 602 patient died within the follow up period. More information about the cases characteristics can be found in table 3.1. The data are available on request from the first author.

We consider seven covariates that may be important for the prognosis of breast cancer: the age of the patient at the time of diagnosis, the tumor size (measured in mm), the number of positive lymph nodes, the Bloom-Richardson tumor grading, a binary covariate denoting whether the patient has had a chemotherapy, another binary covariate for radiotherapy following surgery and a binary variable denoting whether the patient received hormonal treatment. We assumed that all of these variables may show a time varying effect especially since the patients have been under observation for a long time, and there were several cases that could considered long term survivors. As time functions second degree b-splines were considered with 3 interior knots, at 1st, 2nd and 3rd quantile of survival time. That led to a matrix of time functions $F = [f_1(t), f_2(t), f_3(t), f_4(t), f_5(t), f_6(t)]$ with $f_1(t)$ a vector of one and $f_2(t), f_3(t), ...$ the first and second b-spline function, and so forth. The model assumed that all seven covariates interact with the six time functions, which results in 42

Table 3.2: Partial log-likelihood, number of estimated parameters, AIC and time of computation for different rank models.

	log-likelihood	parameters	IC	time of computation(s)
rank=1	-4118.57	12	-4130.57	42.74
rank=2	-4092.22	22	-4114.22	38.83
rank=3	-4083.92	30	-4113.92	34.75
rank=4	-4081.40	36	-4117.40	42.74
rank=5	-4079.69	40	-4119.69	16.97
rank=6	-4079.62	42	-4121.62	13.89

parameters to be estimated in total. We fitted the model described in `S-plus` on a Pentium M, 1.86GHz with 1024MB of memory in 13.89 seconds. It was not possible to fit the same model using the standard `survival` package in `R` since the memory requirements for the expansion and handling of the data set exceeded the available computer memory.

We fitted all possible reduced-rank models in the data set of breast cancer patients. The calls are given here:

```
attach(iaso)
Ft <- cbind(rep(1,nrow(iaso)),bs(time,knots=c(24,48,96),degree=2))
#creates matrix of time functions
fit <- coxvc(Surv(time,death)~age+mass+positive+grade+chemo+radio
           +horm,Ft,rank=6)
```

Table 3.2 presents the five different models, from rank=1 up to full rank, along with the partial likelihood, the number of free parameters, the Akaike's Information criterion and the time (in seconds) required to fit the models. According to AIC, the best model is rank=3.

The rank 3 model has 30 free parameters to describe the time varying effects of the covariates, compared with the full rank model that has 42. In figure 3.2 the effects of the covariates under the rank=3 model and the full rank model are illustrated. There are some small differences in the effects of the chemotherapy variable, with its effect shrunk towards zero in the reduced-rank model, as well as the effect of the hormonal treatment in the start of the

follow up period. However, the figure reveals that the time varying effects can be described adequately with less parameters.

Following one of the referees suggestions we also fitted an approximation to the full rank model. The 200 month time frame was converted into 33 six-months interval. That way we used a coarser grid of break points, making the stack and split approach work on a smaller data set. The result can be seen in Figure 3.3. It can be seen that the approximation to the full model performs reasonable for some covariates at the first few months but the differences become more important as time passes by.

Survival of ovarian cancer patients

To compare the results of our routine with the standard `coxph` function we analyzed data from a smaller study. The data set consists of 358 patients suffering from ovarian cancer, originally analyzed by Verweij and van Houwelingen [104], and it is included as the demonstration data set in the `coxvc` package. We have information on three covariates measured at the start of the treatment, the Karnofsky index, a categorical variable that measures the ability of the patients (from 1 up to 4), Figo status, a variable denoting the stage according to International Federation of Gynecology and Obstetrics, taking values 0 or 1, and the diameter of the residual tumor after the surgery, measured into four categories from small to larger.

For fitting a full rank model we used the following calls:

```
data(ova); attach(ova)
Ft <- cbind(rep(1,nrow(ova)),bs(time,df=3))
coxvc(Surv(time,death)~karn+diam+figo, Ft, rank=3, data=ova)
```

The model was fitted in 0.79 seconds and the results are given in table (3.3). In order to fit a model with time varying effects of the covariates we first had to expand the data to create the different risk sets using the function `expand.breakpoints` created by Kathy Russel. The function is available on the internet from the following url:
`www.biostat.wustl.edu/archives/html/s-news/1998-11/msg00139.html`.
The calls used to create the model follow:

```
breakpoints <- sort(unique(time))
ova.exp <- expand.breakpoints(ova,breakpoints, tevent="time",
           status="death", index="x", Zvar=FALSE)
attach(ova.exp)
```

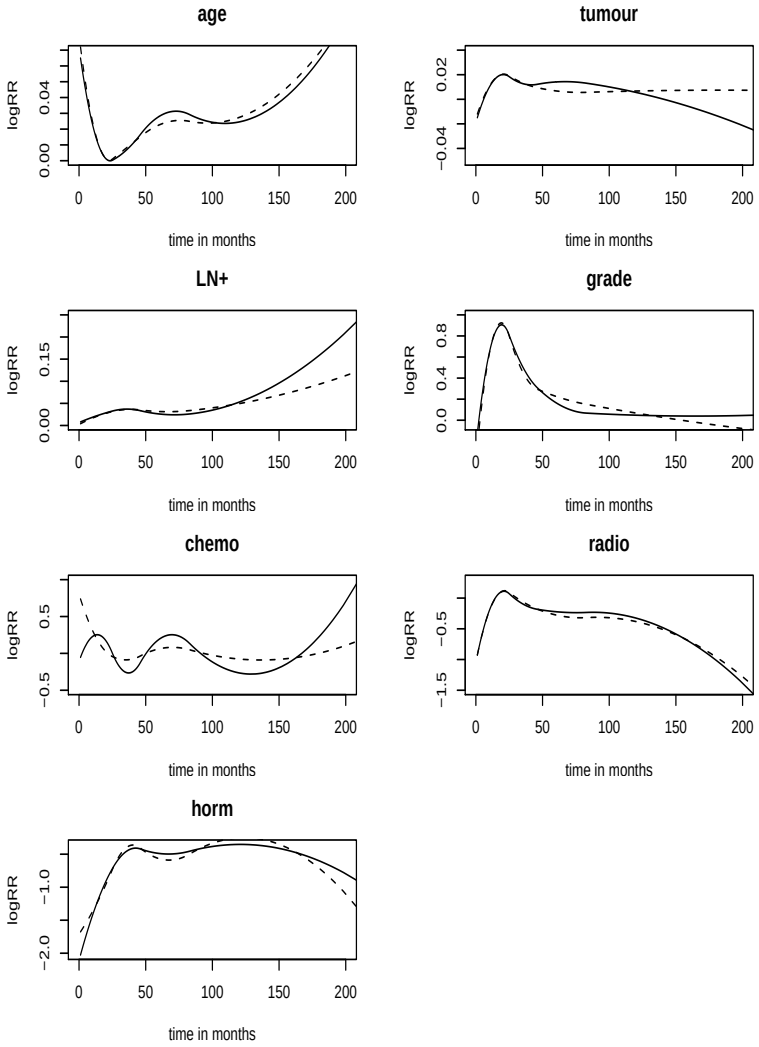

Figure 3.2: Effects of the covariates through time under the rank=3 (dashed line) and the full rank (solid line) model. Time in months.

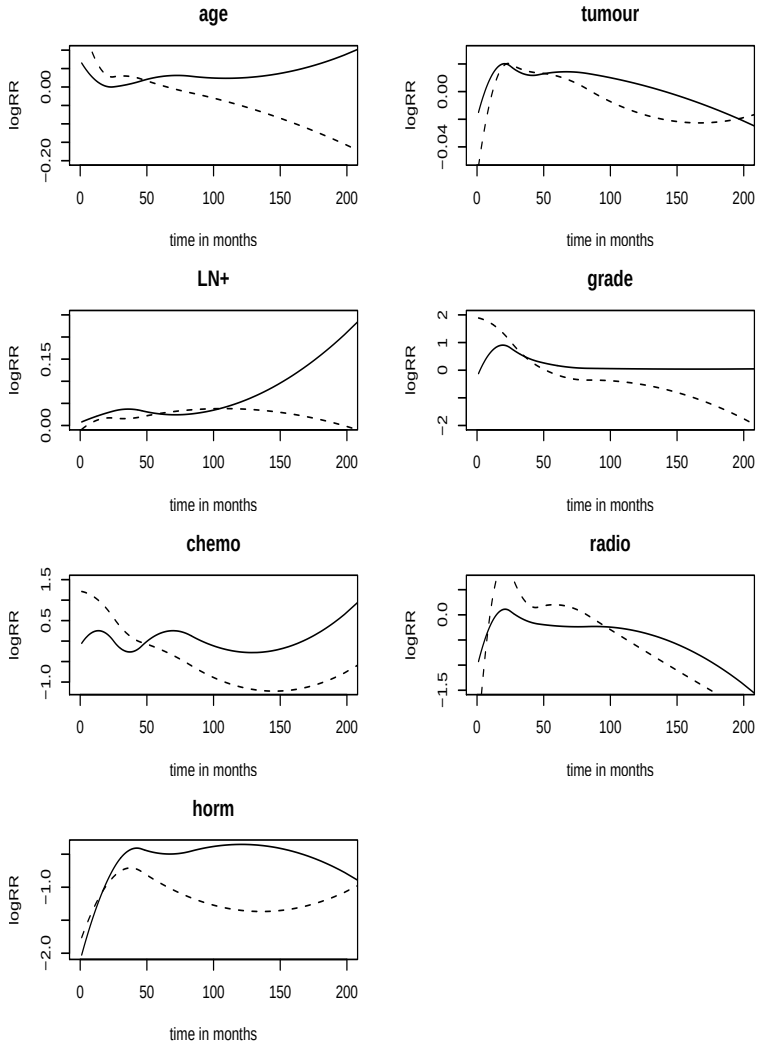

Figure 3.3: Effects of the covariates through time under the full rank model (solid line) and the approximated model (dashed line). Time in months.

Table 3.3: Results from full rank model using the function `coxvc` in the ovarian data set.

	coef	exp(coef)	se(coef)	z	p
karn	0.5562	1.744	0.152	3.6538	0.00026
diam	0.1990	1.220	0.168	1.1857	0.24000
figo	0.5011	1.650	0.392	1.2784	0.20000
karn:f1(t)	-1.3696	0.254	0.730	-1.8761	0.06100
diam:f1(t)	0.0426	1.044	0.696	0.0612	0.95000
figo:f1(t)	0.5433	1.722	1.797	0.3022	0.76000
karn:f2(t)	0.4969	1.644	1.396	0.3559	0.72000
diam:f2(t)	0.4651	1.592	1.072	0.4337	0.66000
figo:f2(t)	-2.2802	0.102	3.201	-0.7122	0.48000
karn:f3(t)	-1.6764	0.187	2.150	-0.7796	0.44000
diam:f3(t)	-1.5500	0.212	1.546	-1.0029	0.32000
figo:f3(t)	3.4807	32.482	4.358	0.7987	0.42000

```
Ftt <- cbind(rep(1,nrow(ova.exp)), bs(Tstop,df=3))
coxph(Surv(Tstart,Tstop,death)~karn:Ftt+diam:Ftt+figo:Ftt,
      data=ova.exp)
```

To expand the data set 1.14 seconds were needed, and 1.42 seconds for fitting the model on the expanded data. The results from the model are almost identical to the ones given in table (3.3), with only small differences in the third decimal digit of the `z` values in some of the covariates. Overall, our procedure fitted the model even in a small data set faster than `coxph` with very similar results.

Simulated data

To test the speed of the algorithm we simulated survival data of different size. In each case, two covariates were created, one dichotomous and one continuous (X_1 and X_2). The follow up time was simulated from an exponential distribution with rate equal to $\exp(\beta_1 X_1 + \beta_2 X_2)$ with $\beta_1 = 1$ and $\beta_2 = 0.5$. We assumed that there were no censored cases.

Following the classical `R` approach we first had to expand the simulated data

Table 3.4: Time of estimation (in seconds) for simulated data sets of size n using the new algorithm (`coxvc`), `coxph` command in `R` and `proc phreg` in `SAS`. Time to fit coxph also includes the time to create an expanded data set using expand.breakpoints function.(-) Not enough memory to fit the model.

n	coxvc	R coxph	SAS phreg
100	0.15	0.16	0.15
250	0.54	1.34	0.20
500	1.06	9.41	0.39
750	1.93	19.98	1.03
1000	3.32	48.76	2.98
1500	6.94	165.80	6.09
2000	11.24	-	9.50
3000	23.74	-	18.93
5000	44.22	-	318.29
10000	156.83	-	784.23

set and then fit the model using the following code:

```
breakpoints <- sort(unique(time))
dat.exp <- expand.breakpoints(sim.data,breakpoints,index="id",
           status=death,tevent="time", Zvar=FALSE)
fit <- coxph(Surv(Tstart,Tstop,death)~ x1+x2+x1:Tstop+x2:Tstop,
       data=dat.exp)
```

Using our package `coxvc` we fitted the model on the original data set as follows:

```
Ft <- cbind(rep(1,nrow(sim.data)),time)
fit <- coxvc(Surv(time,death)~x1+x2,Ft,rank=2,data=sim.data)
```

while in `SAS` we used `proc phreg` to fit the model. Table 3.4 illustrates the times for fitting full rank models on different sizes of data using the new approach versus the standard `coxph` command from `survival` package in `R` (which is based on an `S-plus` library written by T. Therneau [91] and translated into a package by T. Lumley [64]) and `SAS` [83] `proc phreg`.

For a small data set of 100 cases and using our proposed `coxvc` command, 0.15 seconds were needed to estimate the model, while for a data set of 10000

cases with no censoring, the procedure required 156.83 seconds (less than 3 minutes) to give the result. In the case of `coxph`, the time that was required to expand the data set -using the function `expand.breakpoints`- is also included in the table. Since the data set has to be expanded, `S-plus` fails to fit models with number of events that exceed 1500. On the other hand, `SAS` is quicker than our algorithm in small data sets, but as the number of events increases the routine becomes slower.

3.6 Discussion

We have presented a simple yet very efficient algorithm for fitting Cox models with time dependent effects of the covariates. The main merit of our function is that it works directly on the original data set allowing for time varying models to be fitted on a large number of cases. With respect to estimation time, the routine is equivalent to `proc phreg` used in `SAS` when applied to small data sets, but performs better as the number of cases increases, and much faster than the routine used in `S-plus` and `R`. In simulated examples the computational time was decreased by orders of magnitude, and our routine was able to fit models on large data, where `SAS` was slower and `R` failed due to memory problems.

One of the referees raised a concern about the precision of our routine since we use different methods for inverting matrices. In our experience so far with the presented examples and the simulated data sets, the results are very similar in the cases where we could compare our method with the standard `coxph` function. We have also distributed our package among colleagues for testing and use and none of them reported any problems. The package was also tested on later versions of `R` and it works without any problems.