Landmarking 2.0, prothrombin data

Hein Putter

09 maart, 2021

Contents

1 Introduction

The purpose of this document is to implement and illustrate "landmarking 2.0", using $\hat{X}(t|s) = \mathrm{E}(X(t) \,|\, \overline{X}(s))$ as time-dependent covariate, as a companion to "Landmarking 2.0: Bridging the gap between joint models and landmarking". Notation in this document is as in the paper, in particular $X(t)$ is the biomarker, and $\overline{X}(s)$ denotes the collection of observed measurements of the process of the individual until the landmark time point s. I will use the prothrombin data from the CLS-1 trial, comparing prednisone with placebo for patients with liver cirrhosis.

2 CLS data

The CLS-1 trial data are stored in "cls.Rdata", with `cls1` containing the time-to-event data and `cls2` containing the longitudinal prothrombin measurements.

```
load("cls.Rdata")
head(cls1)
```

```
##   patid    survyrs status   treat
## 1     1  0.4134155      1 Placebo
## 2     2  6.7542779      1 Placebo
```

```
## 3     3 13.3935661      0 Placebo
## 4     4  0.7939767      1 Placebo
## 5     5  0.7501711      1 Placebo
## 6     6  0.7693361      1 Placebo
```

```
# Remove measurements *at* t=0, to avoid "immediate" treatment effects
cls2 <- subset(cls2, measyrs>0)
head(cls2)
```

```
##   patid   measyrs prothr   treat
## 2     1 0.2436687     31 Placebo
## 3     1 0.3805613     27 Placebo
## 5     2 0.6872005     73 Placebo
## 6     2 0.9609856     90 Placebo
## 7     2 1.1882272     64 Placebo
## 8     2 1.4428474     54 Placebo
```

```
# By removing subjects from cls2 that only have measurement at t=0,
# we have different sample sizes
pats1 <- sort(unique(cls1$patid))
pats2 <- sort(unique(cls2$patid))
n1 <- length(pats1)
n2 <- length(pats2)
n1
```

```
## [1] 488
```

```
n2
```

```
## [1] 446
```

```
"%w/o%" <- function(x, y) x[!x %in% y] #--  x without y
notincls2 <- pats1 %w/o% pats2 # these are the ones we loose
# They are all patients that died or were lost to follow-up within one year
subset(cls1, patid %in% notincls2)
```

```
##     patid    survyrs status      treat
## 10     10 0.14510609      1 Prednisone
## 17     19 0.08487337      1 Prednisone
## 21     24 0.10951403      1 Prednisone
## 44     60 0.04380561      1    Placebo
## 50     68 0.06570842      0 Prednisone
## 68     94 0.08761123      1 Prednisone
## 70     96 0.11225188      1    Placebo
## 74    103 0.26557153      1 Prednisone
## 95    131 0.06297057      1 Prednisone
## 123   167 0.63244353      0    Placebo
## 134   180 0.22450376      1    Placebo
## 138   184 0.03559206      1    Placebo
## 156   204 0.01095140      1 Prednisone
## 161   209 0.45722108      1    Placebo
```

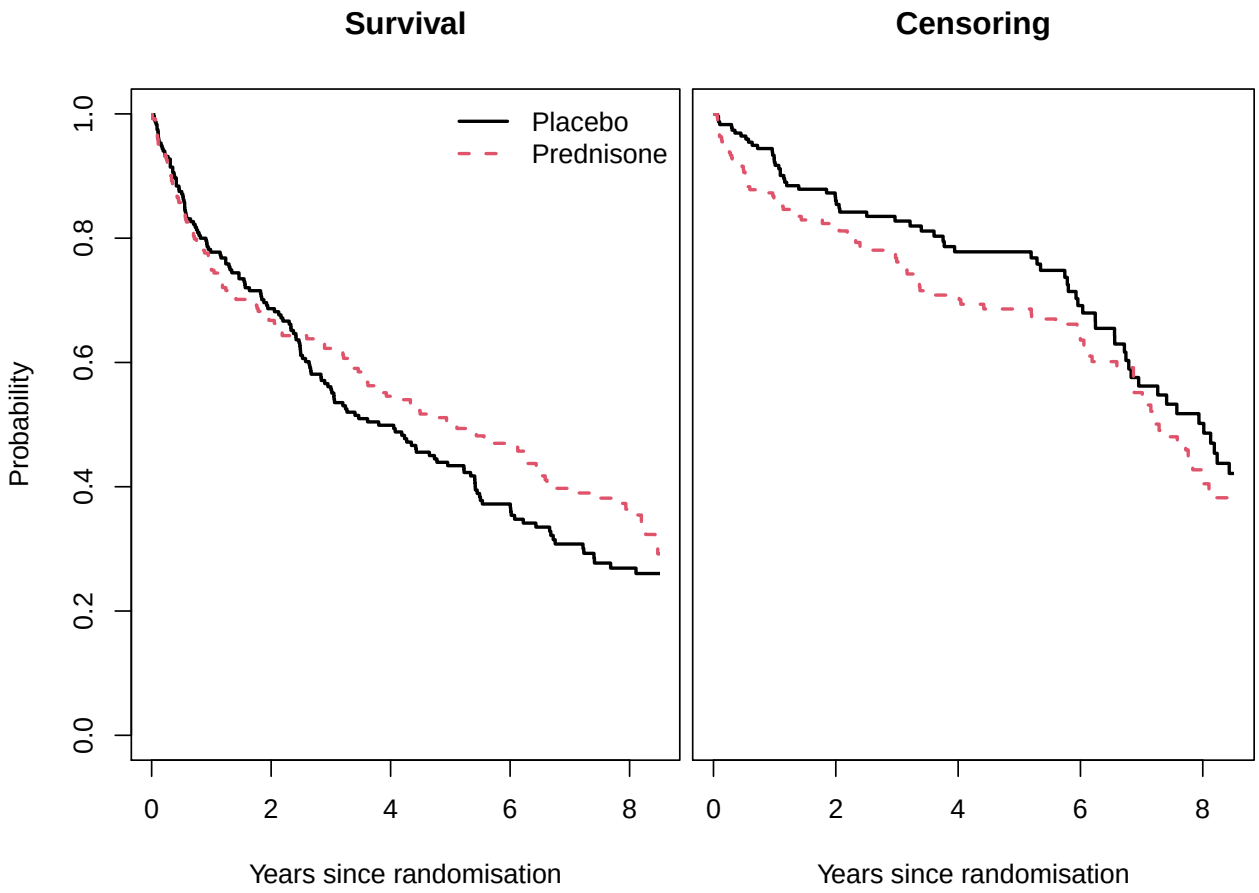
```
## 163   212 0.31211499      1    Placebo
## 174   224 0.06023272      0 Prednisone
## 189   242 0.07939767      0 Prednisone
## 203   256 0.15879535      1    Placebo
## 221   275 0.06570842      0    Placebo
## 225   280 0.17522245      1    Placebo
## 235   291 0.07392197      1    Placebo
## 250   307 0.06844627      0 Prednisone
## 251   309 0.08213552      0 Prednisone
## 262   321 0.09034908      1    Placebo
## 292   351 0.08213552      0 Prednisone
## 313   374 0.01368925      1 Prednisone
## 316   377 0.88432580      0 Prednisone
## 325   386 0.13689254      0 Prednisone
## 340   401 0.19986311      0 Prednisone
## 341   402 0.12320329      1    Placebo
## 362   425 0.08213552      0    Placebo
## 367   432 0.57221081      0    Placebo
## 389   456 0.09308693      0 Prednisone
## 415   482 0.09582478      0 Prednisone
## 420   487 0.07939767      1 Prednisone
## 432   500 0.09856263      0 Prednisone
## 448   516 0.10130048      0    Placebo
## 454   523 0.54209446      0 Prednisone
## 458   527 0.09582478      1 Prednisone
## 466   536 0.09856263      1 Prednisone
## 484   556 0.13141684      0 Prednisone
## 486   558 0.08213552      0    Placebo
```

```
# Since we are going to look at landmark predictions from 3 year onwards, this is OK

# For later attempts to make functions more generic
data1 <- cls1
data2 <- cls2
```

We start with a plot of the survival and censoring distributions by treatment, similar to Chapter 13 in the book "Dynamic Prediction in Clinical Survival Analysis".

```
# Kaplan-Meier plots for survival and censoring
cls.km <- survfit(formula = Surv(survyrs, status) ~ treat, data = cls1)
cls.cens <- survfit(formula = Surv(survyrs, status==0) ~ treat, data = cls1)

# Plot
layout(matrix(1:2, 1, 2),widths=c(10.25,9))
par(mar= c(5, 4, 4, 0.1) + 0.1)
plot(cls.km, mark.time= FALSE, conf.int=FALSE, lwd=2, xlim=c(0,8.5),
     xlab = "Years since randomisation", ylab = "Probability",
     col=1:2, lty=1:2)
legend("topright", levels(cls1$treat), lwd=2, col=1:2, lty=1:2, bty="n")
```

```
title(main="Survival")
par(mar= c(5, 0.1, 4, 1) + 0.1)
plot(cls.cens, mark.time=FALSE, conf.int=FALSE, lwd=2, xlim=c(0,8.5),
     xlab = "Years since randomisation", ylab = "", axes=FALSE,
     col=1:2, lty=1:2)
axis(1)
box()
title(main="Censoring")
```



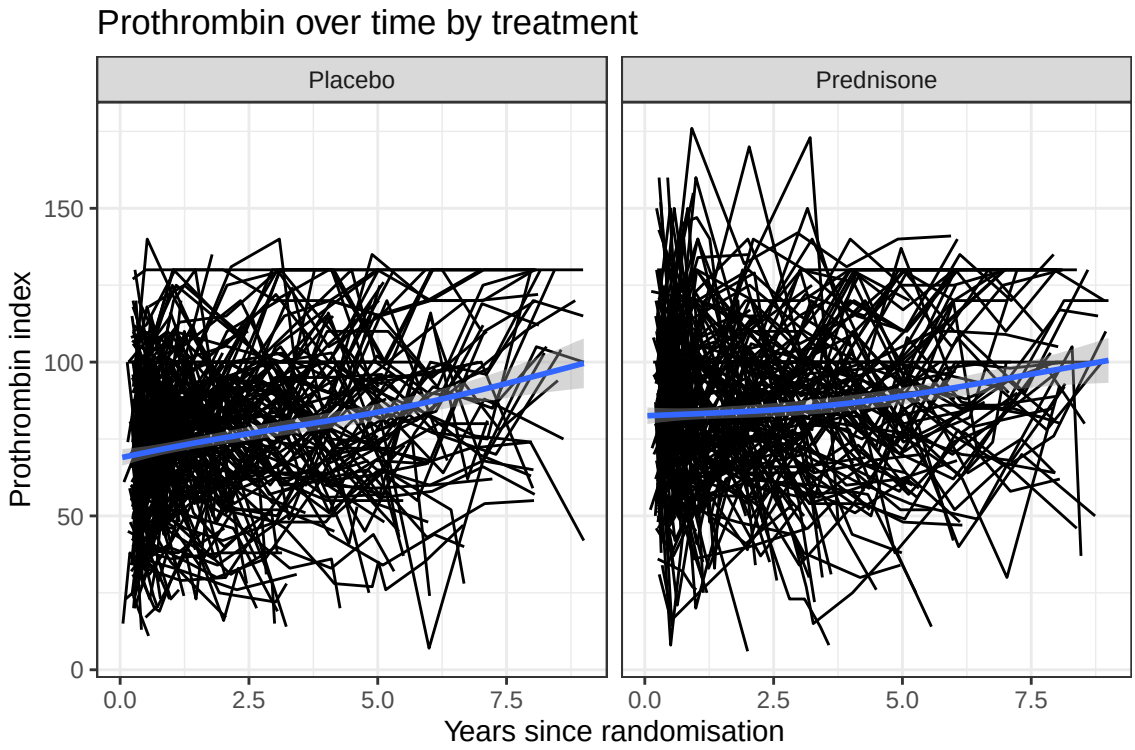
This shows that survival and censoring functions are not very different between the randomized treatment arms.

3 Modeling the longitudinal data

Here is a spaghetti plot of the prothrombin index over time, along with a loess curve, by treatment.

```
a <- ggplot(data = cls2, aes(x = measyrs, y = prothr, group = patid))

a + geom_line() + stat_smooth(aes(group=1)) +
  facet_grid(cols=vars(treat)) +
  xlim(0, 9) +
  theme_bw() +
```

```
  labs(title="Prothrombin over time by treatment") +
  xlab("Years since randomisation") +
  ylab("Prothrombin index")
```



I am fitting a Gaussian process to these data, assuming $X(t)$ has mean function $\mu(t) = \mathrm{E}X(t) = a + bt$, with separate intercept a and slope b per treatment, and covariance function $C(s,t) = \mathrm{cov}(X(s), X(t))$. I will assume that the structure of the covariance function is given by $C(s,t) = \sigma_1^2 + \sigma_2^2 \exp(-\lambda|s-t|) + \sigma_3^2 \mathbf{1}\{s=t\}$, with between subjects variance σ_1^2, within subjects variance σ_2^2, exponential decay parameter λ, and random error σ_3^2, all to be estimated from the data.

My sincere thanks go to Mike Sweeting for help with the following (remainder of this section). This model can be fitted in `nlme`: in `nlme`, a model with the corExp correlation structure and a "nugget" assumes the correlation of the errors within an individual (group) is $h(\epsilon(s), \epsilon(t); c_0, \rho) = (1-c_0)\exp(-|t-s|/\rho)$ if $s \neq t$, where c_0 is the nugget parameter and ρ is the range parameter (see "Mixed-Effects Models in S and S-PLUS"by Pinheiro and Bates). The function `lme` in `nlme` provides estimates of c_0, ρ and the full residual variance, σ_w^2. Therefore, when $s \neq t$ the covariance function is $\sigma_w^2(1-c_0)\exp(-|t-s|/\rho)$ and hence $\sigma_2^2 = \sigma_w^2(1-c_0)$ in the previous Gaussian process notation. When $s = t$ the correlation function is zero and the residual variance is $\sigma_w^2 c_0$, which corresponds to σ_3^2 in the previous Gaussian process representation.

```
exp1 <- corExp(form= ~ measyrs * treat | patid, nugget=TRUE)
exp1 <- Initialize(exp1, data2)
fitLME.GP <- lme(prothr ~  measyrs * treat, random = ~ 1 | patid,
                 data = data2, correlation = exp1)
summary(fitLME.GP)
```

```
## Linear mixed-effects model fit by REML
```

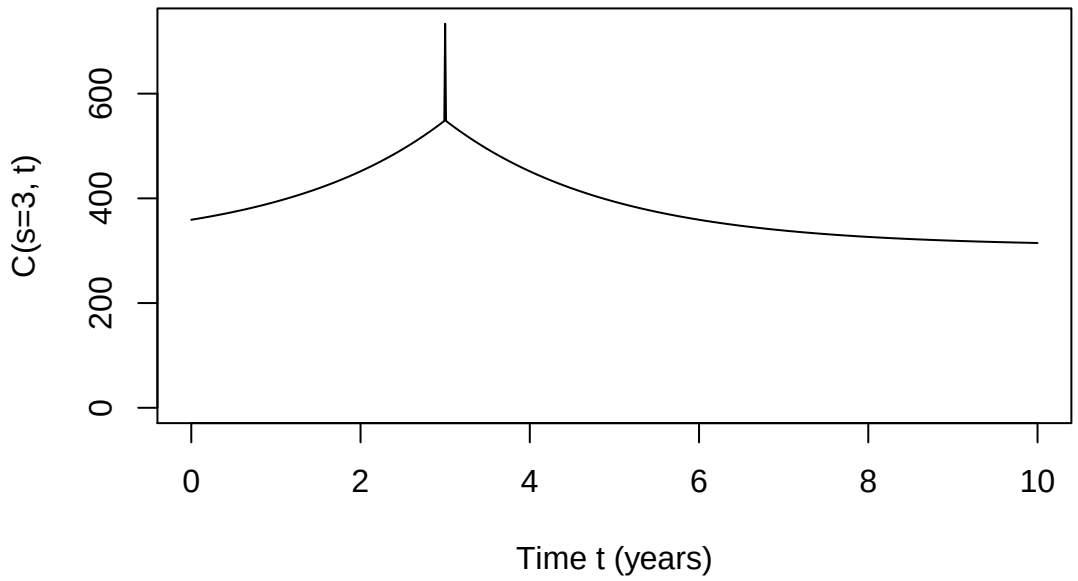
```
##  Data: data2
##        AIC      BIC    logLik
##   22161.93 22208.45 -11072.97
##
## Random effects:
##  Formula: ~1 | patid
##         (Intercept) Residual
## StdDev:    17.56127  20.6171
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~measyrs * treat | patid
##  Parameter estimate(s):
##     range    nugget
## 1.9263994 0.4335665
## Fixed effects: prothr ~ measyrs * treat
##                             Value Std.Error   DF  t-value p-value
## (Intercept)             69.02593 1.6925060 2033 40.78327  0.0000
## measyrs                  2.19188 0.3712409 2033  5.90419  0.0000
## treatPrednisone         11.54753 2.3584364  444  4.89627  0.0000
## measyrs:treatPrednisone -1.15937 0.5027266 2033 -2.30617  0.0212
##  Correlation:
##                         (Intr) mesyrs trtPrd
## measyrs                 -0.456
## treatPrednisone         -0.718  0.327
## measyrs:treatPrednisone  0.336 -0.738 -0.452
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -3.98653505 -0.55212111  0.01387547  0.60530123  3.37384918
##
## Number of Observations: 2481
## Number of Groups: 446
```

```r
# Extract and print estimated parameters
varcorr <- VarCorr(fitLME.GP)
rangenugget <- coef(fitLME.GP$modelStruct$corStruct, unconstrained = FALSE)
ss1 <- as.numeric(varcorr[1, 1])
nugget <- rangenugget[2]
range <- rangenugget[1]
ss2 <- as.numeric(varcorr[2, 1]) * (1-nugget)
ss3 <- as.numeric(varcorr[2, 1]) * nugget
bet <- fitLME.GP$coefficients$fixed
apl <- bet[1]; bpl <- bet[2]
apr <- bet[1] + bet[3]; bpr <- bet[2] + bet[4]
estimates <- c(apl, bpl, apr, bpr, ss1, ss2, ss3,  1/range)
names(estimates) <- c("apl", "bpl", "apr", "bpr", "ss1", "ss2", "ss3", "lambda")
print(estimates, digits=6)
```

```
##         apl         bpl         apr         bpr         ss1         ss2         ss3      lambda
##   69.025933    2.191877   80.573463    1.032504 308.398100 240.770982 184.293918    0.519103
```

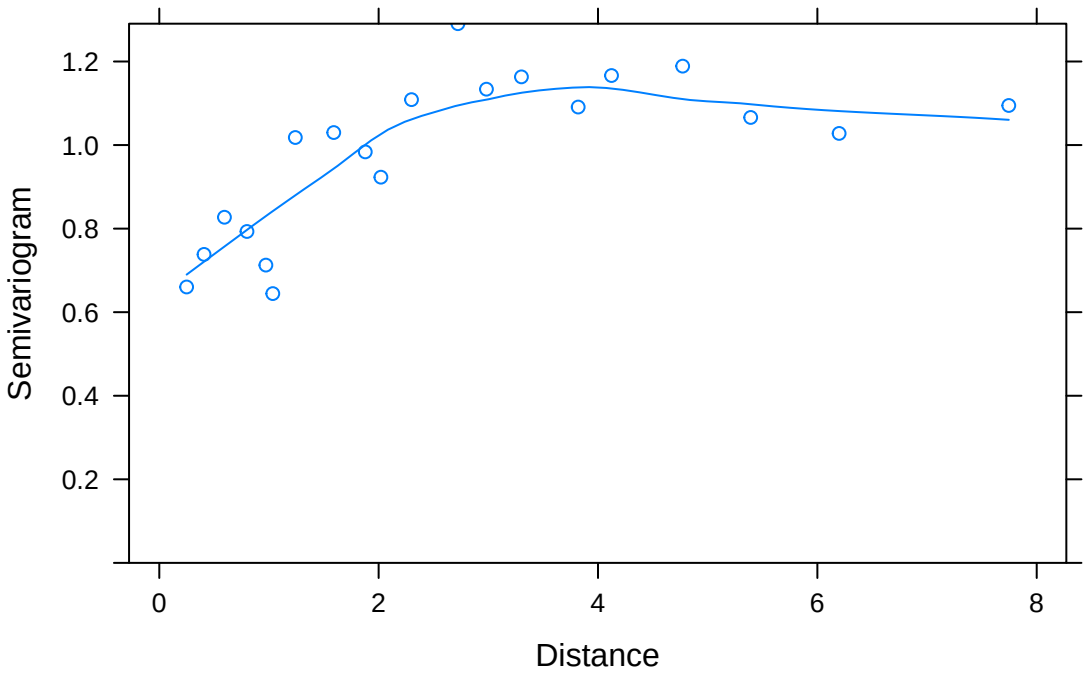
I will first make a plot of the covariance function $C(s,t)$ for a fixed s, varying t; I'm taking $s = 3$. The spike at $t = s$ is due to the random error that appears in the variance and not in the covariances.

```
covarf <- function(s, t, ss1, ss2, ss3, lambda)
  return(ss1 + ss2*exp(-lambda*abs(s-t)) + ss3*(s==t))
ss1 <- estimates[5]
ss2 <- estimates[6]
ss3 <- estimates[7]
lambda <- estimates[8]
# Let's look at a plot for s=3, and t varying
ttseq <- seq(0, 10, by=0.01)
covarseq <- covarf(s=3, t=ttseq, ss1=ss1, ss2=ss2,
                   ss3=ss3, lambda=lambda)
plot(ttseq, covarseq, type="l", ylim=c(0, max(covarseq)),
     xlab="Time t (years)", ylab="C(s=3, t)")
```



We can also look at the semivariogram of the residuals from an 'lme' model with independent error structure to visualise whether there is a correlation structure in the data:

```
fitLME <- lme(prothr ~  measyrs * treat, random = ~ 1 | patid, data = data2)
plot(Variogram(fitLME, form= ~ measyrs, maxDist = 10))
```



The semivariogram appears to increase with distance up to 2 years suggesting a correlation model may be needed. There is also a suggestion that a nugget is required (the semivariogram does not go to zero at distance zero).

```
exp2 <- corExp(form= ~ measyrs | patid, nugget=FALSE)
exp2 <- Initialize(exp2, data2)
fitLME.GP.noNugget <- lme(prothr ~  measyrs * treat, random = ~ 1 | patid,
                          data = data2, correlation = exp2)
```

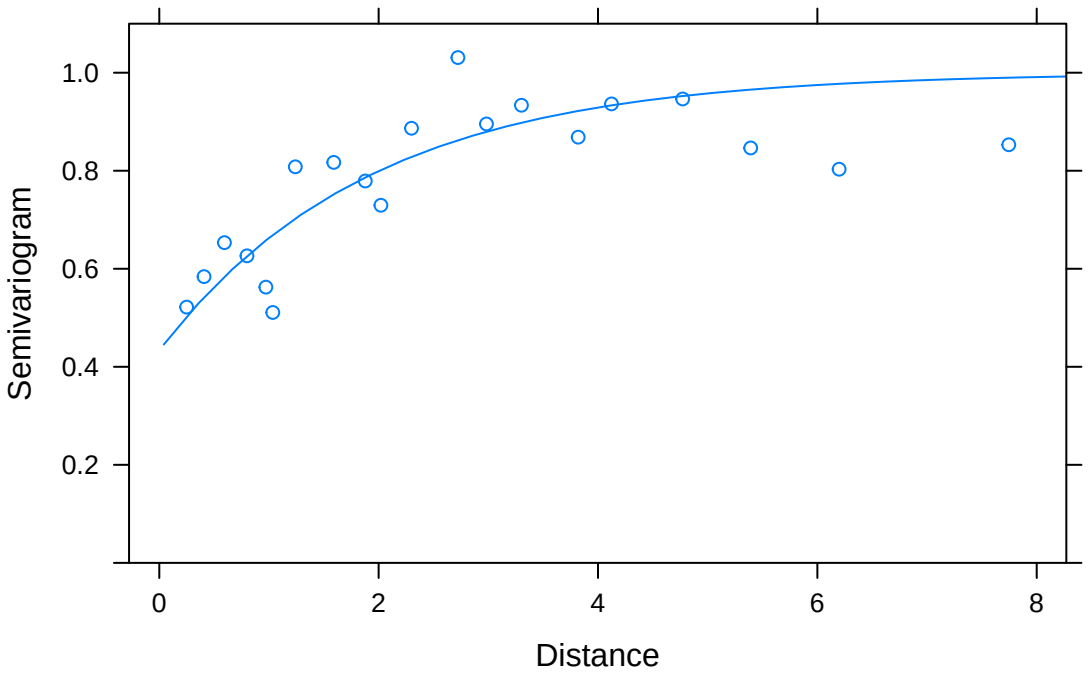
An ANOVA of the three models (independent errors, exponential no nugget, exponential with nugget) confirms this, with a much smaller AIC for the model with exponential correlation structure and a nugget effect:

```
anova(fitLME, fitLME.GP.noNugget, fitLME.GP)
```

```
##                    Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## fitLME                 1  6 22316.42 22351.31 -11152.21
## fitLME.GP.noNugget     2  7 22232.37 22273.07 -11109.18 1 vs 2 86.04941  <.0001
## fitLME.GP              3  8 22161.93 22208.45 -11072.97 2 vs 3 72.43995  <.0001
```

Finally, we can view the adequacy of the exponential correlation model by displaying the fitted semivariogram along with the sample variogram estimates. The result looks quite reasonable.

```
plot(Variogram(fitLME.GP, form = ~ measyrs, maxDist = 10), ylim=c(0, 1.1))
```



4 Procedures for dynamic prediction based on longitudinal markers

This long section will implement landmark and other models with the aim of obtaining dynamic predictions at some time point $s + w$ using all longitudinal information before the prediction time point s. Here, w is the window width, describing how far ahead in the future we want to predict. In this document I will set the landmark time point at $s = \text{LM} = 3$ years, and set the window width for the prediction at $w = 2$ years, implying that predictions are obtained for 5 years after randomisation. I am making everything semi-generic, by using `id`, `time1`, `status1`, `time2`, `marker` as the column names in the data indicating the subject id, time and status in the time-to-event data, time and marker in the longitudinal data. However, there are some specific points with this analysis (presence of treatment for instance) that make it difficult to make it completely generic. I have not attempted this.

```
LM <- 3
width <- 2
id <- "patid"
time1 <- "survyrs"
status1 <- "status"
time2 <- "measyrs"
marker <- "prothr"
```

When fitting the time-dependent Cox models and other models it is convenient to use all data, but when comparing predictive accuracy (in the form of Brier and Kullback-Leibler scores) of the resulting dynamic predictions, it is more appropriate to use cross-validation. Therefore, each of the sections coming up will consist of subsections "No cross-validation" (with models fitted on the complete data) and "Cross-validation" (based on leave-one-out cross-validation).

We will consider the following ways of obtaining these dynamic predictions:

- Based on last observed measurement (last observation carried forward, LOCF, Subsection 4.2)
- Based on the fixed value $\hat{X}(s|s)$ (landmarking 1.5, Subsection 4.3)
- Based on $\hat{X}(t|s)$ as predictable time-dependent covariate obtained from Gaussian process (landmarking 2.0, Subsection 4.1)
- Based on $\hat{X}(t|s)$ as predictable time-dependent covariate obtained from revival (landmarking 2.0 revival, Subsection 4.4). This subsection will also derive direct revival predictions.
- Based on joint model for longitudinal and time-to-event data (Subsection 4.5)

Since the emphasis in the paper is on landmarking 2.0, using $\hat{X}(t|s) = \mathrm{E}(X(t) \,|\, \overline{X}(s))$ as predictable time-dependent covariate in a time-dependent Cox model, I will start with this, leaving the competing (sometimes simpler, sometimes more involved) approaches for later.

4.1 Landmarking 2.0 based on Gaussian process

4.1.1 No cross-validation

It will be convenient to define some procedures that I will have to repeat quite often below as functions. The first of these, `makeLMdata`, extracts the relevant landmark data from the original data (`data1` and `data2`), both the time-to-event data and the longitudinal data (for the latter we separately output the measurements before and after the landmark time point s). The relevant time points after s are all event time points in the data between s and the failure or censoring time point of the patient.

```
makeLMdata <- function(data1, data2, id, time1, time2, LM)
{
  # data1 and data2 are the survival and longitudinal data, resp.
  # id, time1, time2 are the column names corresponding to the id
  #   (in both data sets), and time in data1 and data2, resp.,
  # LM is the landmark time point
  #
  # Landmark data 1
  data1LM <- data1[data1[[time1]] > LM, ]
  patLM <- sort(unique(data1LM[[id]]))
  nLM <- nrow(data1LM)
  # Landmark data 2 (marker data)
  data2LM <- data2[data2[[id]] %in% patLM, ]
  # Marker data before landmark (used for \overline{X}(t) later)
  data2LMbef <- data2LM[data2LM[[time2]] <= LM, ]
  # Marker data after landmark
  data2LMaft <- data2LM[data2LM[[time2]] > LM, ]
  return(list(data1LM=data1LM, data2LMbef=data2LMbef, data2LMaft=data2LMaft))
}
```

The second function, `mklongdata`, has as input the new landmark datasets, and uses the results of the fitted Gaussian process (through the argument `estimates`) to produce long (Andersen-Gill counting process) format data with the predictable time-dependent covariates $\hat{X}(t|s)$ at each of the relevant time points (`ttLM`). When fitting the time-dependent Cox model these would be all of the

event time points after the landmark time point s and before the prediction time horizon $s + w$ (using administrative censoring at $s + w$).

```
mklongdata <- function(data1LM, data2LMbef, id, ttLM,
                       time1, status1, time2, marker, estimates, LM, width)
{
  # data1LM: the landmark survival data set
  # data2LMbef: the set of observed measurements before LM (LM patients only)
  # id, time1, status1, time2, marker are column names of id, time in data1,
  #   status in data1, time in data2, and the biomarker
  # ttLM: a vector of time points after LM at which the predictable
  #   time-dependent covariates are calculated
  # estimates: a vector with parameter estimates of the Gaussian process
  # LM: the landmark time point
  # width: the prediction width
  #
  patLM <- sort(unique(data1LM[[id]]))
  nLM <- nrow(data1LM)
  # Prepare structure for time-dependent covariate \overline{X}(t)
  dataLM <- survSplit(Surv(data1LM[[time1]], data1LM[[status1]]) ~ .,
                      data=data1LM, cut=ttLM)
  dataLM$tstart[dataLM$tstart<LM] <- LM # repair tstart=0
  # Administrative censoring at LM+width
  whcens <- which(dataLM$tstop > LM+width)
  dataLM$tstop[whcens] <- LM+width
  dataLM$event[whcens] <- 0

  # These would have to change, depending on the particular setting
  meanf <- function(t, a, b) return(a + b*t)
  covarf <- function(s, t, ss1, ss2, ss3, lambda)
    return(ss1 + ss2*exp(-lambda*abs(s-t)) + ss3*(s==t))
  # Unpack estimates
  apl <- estimates[1]
  bpl <- estimates[2]
  apr <- estimates[3]
  bpr <- estimates[4]
  ss1 <- estimates[5]
  ss2 <- estimates[6]
  ss3 <- estimates[7]
  lambda <- estimates[8]

  dataLM$Xhats <- NA # save space for $\hat{X}(t|s)$

  for (i in 1:nLM) {
    whi <- which(dataLM[[id]]==patLM[i])
    if (length(whi) > 0) {
      dataLMi <- dataLM[whi, ]
      a <- ifelse(dataLMi$treat[1]=="Placebo", apl, apr)
```

```
      b <- ifelse(dataLMi$treat[1]=="Placebo", bpl, bpr)
      befi <- data2LMbef[data2LMbef[[id]]==patLM[i], ] # data of subject i before LM
      t1 <- befi[[time2]]
      t2 <- dataLMi$tstop
      mu1 <- unlist(lapply(t1, meanf, a=a, b=b)) # means before LM
      mu2 <- unlist(lapply(t2, meanf, a=a, b=b)) # means after LM
      S11 <- outer(t1, t1, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
      S12 <- outer(t1, t2, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
      Xhats <- mu2 + as.vector(t(S12) %*% solve(S11) %*% (befi[[marker]] - mu1))
      dataLM$Xhats[whi] <- Xhats # \hat{X}(t|s) for subject i
    } else cat("No longitudinal data found before LM for patient", patLM[i], "\n")
  }
  return(dataLM)
}
```

Having these two functions at our disposal, let's apply them to our setting.

```
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM = LM)
data1LM <- tmp$data1LM
data2LMbef <- tmp$data2LMbef
data2LMaft <- tmp$data2LMaft
patLM <- sort(unique(data1LM[[id]]))
nLM <- nrow(data1LM)

ttLM <- sort(unique(data1LM$survyrs[data1LM$status==1 & data1LM$survyrs <= LM+width]))

dataLM <- mklongdata(data1LM=data1LM, data2LMbef=data2LMbef, id="patid",
                     ttLM = ttLM,
                     time1="survyrs", status1="status",
                     time2="measyrs", marker="prothr",
                     estimates,
                     LM=LM, width=width)
```

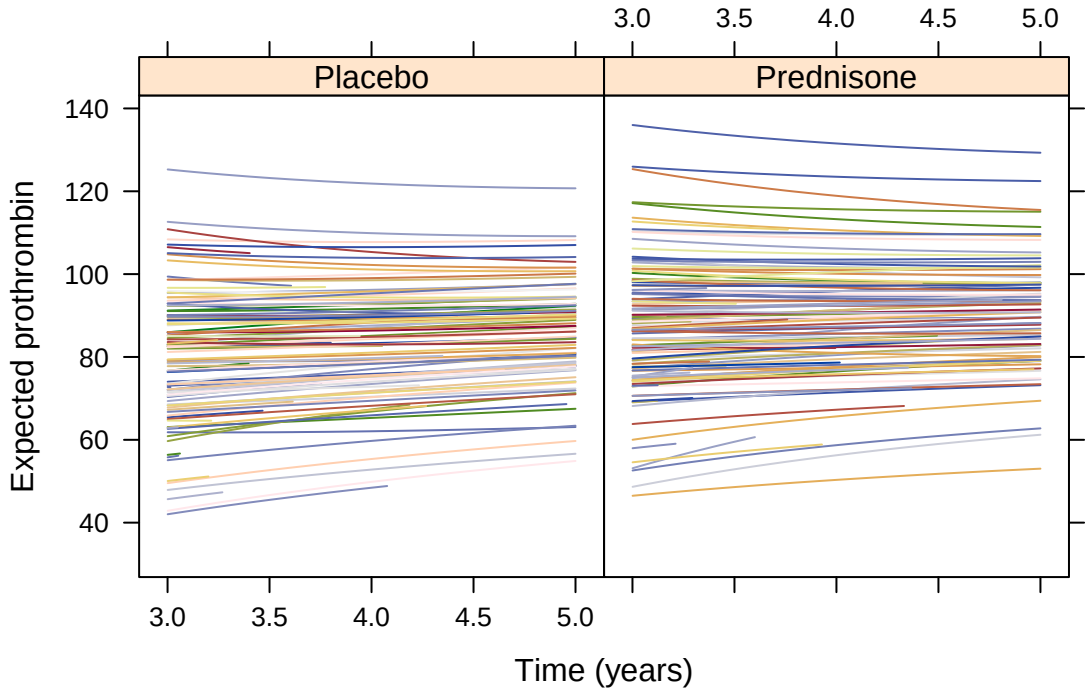
The data set for the time-dependent Cox analysis now looks like this (illustrating subject id 62, who died fairly soon after time s). They run until the event or censoring time (or horizon) for each subject.

```
subset(dataLM, patid==62)
```

```
##     patid  survyrs status      treat   tstart    tstop event    Xhats
## 871    62 3.211499      1 Prednisone 3.000000 3.000684     0 57.96927
## 872    62 3.211499      1 Prednisone 3.000684 3.014374     0 58.04217
## 873    62 3.211499      1 Prednisone 3.014374 3.049966     0 58.22977
## 874    62 3.211499      1 Prednisone 3.049966 3.052704     0 58.24408
## 875    62 3.211499      1 Prednisone 3.052704 3.060917     0 58.28693
## 876    62 3.211499      1 Prednisone 3.060917 3.123888     0 58.61063
## 877    62 3.211499      1 Prednisone 3.123888 3.200548     0 58.99350
## 878    62 3.211499      1 Prednisone 3.200548 3.211499     1 59.04722
```

Here is a plot of the trajectories of $\hat{X}(t|s)$ for the individuals, as calculated in the data:

```
xyplot(Xhats ~ tstop | treat, group = patid, data = dataLM,
  xlab = "Time (years)", ylab = "Expected prothrombin", col = cols, type = "l")
```



Fitting a Cox model with $\hat{X}(t|s)$ as time-dependent covariate is straightforward now and gives as result

```
ctd <- coxph(Surv(tstart, tstop, event) ~ Xhats, data=dataLM)
summary(ctd)
```

```
## Call:
## coxph(formula = Surv(tstart, tstop, event) ~ Xhats, data = dataLM)
##
##   n= 8974, number of events= 46
##
##            coef exp(coef)  se(coef)      z Pr(>|z|)
## Xhats -0.050564  0.950693  0.009991 -5.061 4.17e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##       exp(coef) exp(-coef) lower .95 upper .95
## Xhats    0.9507      1.052    0.9323    0.9695
##
## Concordance= 0.709  (se = 0.037 )
## Likelihood ratio test= 25.07  on 1 df,   p=6e-07
## Wald test            = 25.62  on 1 df,   p=4e-07
## Score (logrank) test = 25.39  on 1 df,   p=5e-07
```

4.1.2 Cross-validation

As mentioned before, to get an honest assessment of predictive accuracy, we use cross-validation. We loop over the individuals, fit the time-dependent Cox model (including modelling the Gaussian process) based on everyone but the left out individual and predict for the left out individual. The predictions are stored and saved.

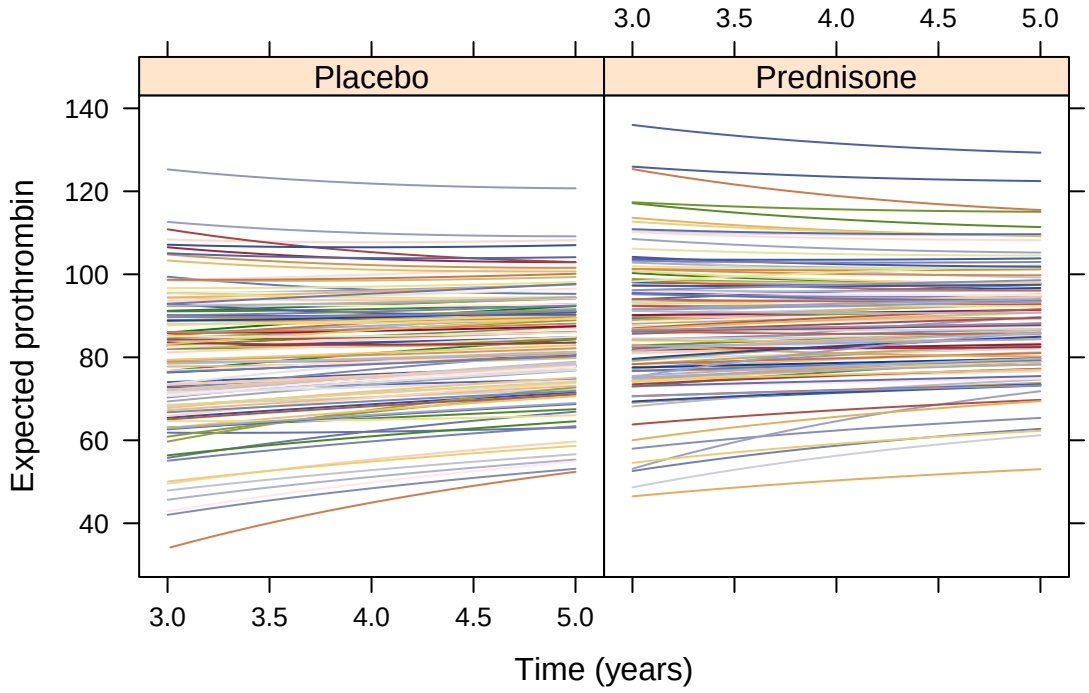
```
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
patLM <- sort(unique(tmp$data1LM$patid))
nLM <- nrow(data1LM)

predXhatCV <- rep(NA, nLM) # to contain the predictions
dataLM_all <- NULL

for (i in 1:nLM)
{
  pati <- patLM[i]

  #
  # Fit time-dependent Cox model on data with subject i excluded
  #
  cls1mini <- subset(cls1, patid != pati)
  cls2mini <- subset(cls2, patid != pati)

  tmp <- makeLMdata(data1=cls1mini, data2=cls2mini, id="patid",
                    time1="survyrs", time2="measyrs", LM=LM)
  data1LMmini <- tmp$data1LM
  data2LMbefmini <- tmp$data2LMbef
  data2mini <- cls2mini

  ttLMmini <- sort(unique(data1LMmini$survyrs[data1LMmini$status==1 &
                                                data1LMmini$survyrs <= LM+width]))
              # these time points will be used later

  # Refit Gaussian process, see Section 3
  exp1 <- corExp(form= ~ measyrs * treat | patid, nugget=TRUE)
  exp1 <- Initialize(exp1, data2mini)
  fitLME.GP <- lme(prothr ~  measyrs * treat, random = ~ 1 | patid,
                   data = data2mini, correlation = exp1)
  # Extract estimated parameters
  varcorr <- VarCorr(fitLME.GP)
  rangenugget <- coef(fitLME.GP$modelStruct$corStruct, unconstrained = FALSE)
  ss1 <- as.numeric(varcorr[1, 1])
  nugget <- rangenugget[2]
  range <- rangenugget[1]
  ss2 <- as.numeric(varcorr[2, 1]) * (1-nugget)
  ss3 <- as.numeric(varcorr[2, 1]) * nugget
```

```
bet <- fitLME.GP$coefficients$fixed
apl <- bet[1]
bpl <- bet[2]
apr <- bet[1] + bet[3]
bpr <- bet[2] + bet[4]
estimatesmini <- c(apl, bpl, apr, bpr, ss1, ss2, ss3,  1/range)
names(estimatesmini) <- c("apl", "bpl", "apr", "bpr", "ss1", "ss2", "ss3", "lambda")

# Make long format data containing the predictable time-dependent covariates
dataLMmini <- mklongdata(data1LM=data1LMmini, data2LMbef=data2LMbef, id="patid",
                     ttLM = ttLMmini,
                     time1="survyrs", status1="status",
                     time2="measyrs", marker="prothr",
                     estimates=estimatesmini,
                     LM=LM, width=width)

# Fit the landmark time-dependent Cox model
ctdmini <- coxph(Surv(tstart, tstop, event) ~ Xhats, data=dataLMmini)

# Extract coefficient and baseline (at mean of time-dependent covariate) hazard increments
betamini <- ctdmini$coef
Xhatsmn <- ctdmini$means
ctdminibh <- basehaz(ctdmini)
ctdminibh <- data.frame(time=ctdminibh$time, H0=ctdminibh$hazard)
ctdminibh$h0 <- diff(c(0, ctdminibh$H0))
ctdminibh <- subset(ctdminibh, h0>0)

#
# Apply time-dependent Cox model on data of subject i
#

cls1i <- subset(cls1, patid == pati)
cls2i <- subset(cls2, patid == pati)
tmpi <- makeLMdata(data1=cls1i, data2=cls2i, id=id, time1=time1, time2=time2, LM=LM)
data1LMi <- tmpi$data1LM
data1LMi[[time1]] <- LM + width # used for prediction, so time and status not to be used
data1LMi[[status1]] <- 0 # set time to horizon, status to 0

dataLMi <- mklongdata(data1LM=data1LMi, data2LMbef=tmpi$data2LMbef, id="patid",
                     ttLM = ttLMmini, # important to use same time points as in fit
                     time1="survyrs", status1="status",
                     time2="measyrs", marker="prothr",
                     estimates, # not used so doesn't matter what I fill in here
                     LM=LM, width=width)
dataLM_all <- rbind(dataLM_all, dataLMi) # save all Xhat values for plotting later

# Hazard increments for subject i at each time point (until horizon)
```

```
  HR <- exp(betamini * (dataLMi$Xhats[dataLMi$tstop != LM+width] - Xhatsmn))
  hi <- ctdminibh$h0 * HR # baseline increment times subject-specific hazard ratio

  # Predicted probability
  predXhatCV[i] <- exp(-sum(hi, na.rm=TRUE))
}

save(predXhatCV, file="predictionsCV.Rdata") # store
```

In the process we have stored the individual predicted values of the time-dependent covariate, this time ranging from s to $s + w$. Below is a spaghetti plot of these cross-validated $\hat{X}(t|s)$ trajectories.

```
xyplot(Xhats ~ tstop | treat, group = patid, data = dataLM_all,
  xlab = "Time (years)", ylab = "Expected prothrombin", col = cols, type = "l")
```



4.2 Last observed measurement (LOCF)

4.2.1 No cross-validation

Here it is useful to define a function returning a data frame with all subjects in the landmark data containing the last observed marker value, as well as a landmark Cox model based on those LOCF values.

```
fitlocf <- function(data1, data2, id, time1, status1, time2, marker, LM, width)
{
  # Fits landmark model with last observation
  # Input parameters are the same as used before
  #
  # Extract LM data, use makeLMdata
```

```
  tmp <- makeLMdata(data1=data1, data2=data2, id=id, time1=time1, time2=time2, LM=LM)
  data1LM <- tmp$data1LM
  data2LMbef <- tmp$data2LMbef
  data2LMaft <- tmp$data2LMaft
  # Extract last observation
  tmp <- data2LMbef[order(data2LMbef$patid, -data2LMbef$measyrs), ]
  locf <- tmp[!duplicated(tmp$patid), ] # first row of each subject contains last value
  locf <- merge(locf, data1LM[, c(id, time1, status1)],
                by=id, all.x=TRUE) # add time-to-event data
  # Administrative censoring at LM+width
  whcens <- which(locf[[time1]] > LM+width)
  locf[[time1]][whcens] <- LM+width
  locf[[status1]][whcens] <- 0
  locftime <- locf[[time1]]
  locfstatus <- locf[[status1]]
  locfmarker <- locf[[marker]]
  clocf <- coxph(Surv(locftime, locfstatus) ~ locfmarker)
  # summary(clocf)
  sf <- survfit(clocf, newdata=data.frame(locfmarker=locf[[marker]]))
  locf$predlastobs <- as.vector(unlist(summary(sf, times=LM+width)$surv))
  return(list(locf=locf, clocf=clocf))
}
```

The resulting Cox model is given by

```
dolocf <- fitlocf(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", status1="status",
                  time2="measyrs", marker="prothr", LM=LM, width=width)
locf <- dolocf$locf # survival data containing last observed marker value
clocf <- dolocf$clocf # landmark Cox model with locf marker value
print(summary(clocf))
```

```
## Call:
## coxph(formula = Surv(locftime, locfstatus) ~ locfmarker)
##
##   n= 229, number of events= 46
##
##                 coef exp(coef)  se(coef)      z Pr(>|z|)
## locfmarker -0.025991  0.974344  0.006301 -4.125 3.71e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##            exp(coef) exp(-coef) lower .95 upper .95
## locfmarker    0.9743      1.026    0.9624    0.9865
##
## Concordance= 0.677  (se = 0.041 )
## Likelihood ratio test= 17.2  on 1 df,   p=3e-05
## Wald test            = 17.02  on 1 df,   p=4e-05
```

```
## Score (logrank) test = 16.98  on 1 df,   p=4e-05
```

4.2.2 Cross-validation

The code below calculates and saves the resulting cross-validated predicted probabilities.

```
# Set up for cross-validation
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
data1LM <- tmp$data1LM
patLM <- sort(unique(data1LM$patid))
nLM <- nrow(data1LM)

predlocfCV <- rep(NA, nLM) # these will contain the predicted probabilities
for (i in 1:nLM) {
  pati <- patLM[i]

  # Fit time-dependent Cox model on data with subject i excluded
  cls1mini <- subset(cls1, patid != pati)
  cls2mini <- subset(cls2, patid != pati)

  dolocf <- fitlocf(data1=cls1mini, data2=cls2mini, id="patid",
                    time1="survyrs", status1="status",
                    time2="measyrs", marker="prothr", LM=LM, width=width)

  clocfmini <- dolocf$clocf # landmark Cox model with locf marker value

  # Apply model to subject i
  locfi <- subset(locf, patid==pati)
  locfi$locfmarker <- locfi$prothr
  sfi <- survfit(clocfmini, newdata=locfi)
  predlocfCV[i] <- summary(sfi, times = LM+width)$surv
}

# Add to image
resave(predlocfCV, file="predictionsCV.Rdata")
```

4.3 Landmarking 1.5

This procedure looks like landmarking 2.0, and we can reuse some of the functions defined there. It does fit the Gaussian process first, but after that it is slightly simpler, since it uses only $\hat{X}(s|s)$ and doesn't need time-dependent Cox models (and as a result doesn't need to make long format data for fitting it).

4.3.1 No cross-validation

```
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
```

```r
data1LM <- tmp$data1LM
patLM <- sort(unique(data1LM$patid))
nLM <- nrow(data1LM)
data2LMbef <- tmp$data2LMbef

meanf <- function(t, a, b) return(a + b*t)
covarf <- function(s, t, ss1, ss2, ss3, lambda)
  return(ss1 + ss2*exp(-lambda*abs(s-t)) + ss3*(s==t))

# First round is to calculate Xhat(s|s) for each subject in LM data
# Insert that into data1LM
data1LM$Xhats <- NA
for (i in 1:nLM)
{
  pati <- patLM[i]

  whi <- which(data1LM[[id]]==patLM[i])
  if (length(whi) > 0) {
    dataLMi <- data1LM[whi, ]
    a <- ifelse(data1LMi$treat[1]=="Placebo", apl, apr)
    b <- ifelse(data1LMi$treat[1]=="Placebo", bpl, bpr)
    befi <- data2LMbef[data2LMbef[[id]]==pati, ]
    t1 <- befi[[time2]]
    n1 <- length(t1)
    mu1 <- unlist(lapply(t1, meanf, a=a, b=b))
    mu2 <- meanf(t=LM, a=a, b=b)
    S11 <- outer(t1, t1, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
    S12 <- outer(t1, LM, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
    XhatLMi <- mu2 + as.vector(t(S12) %*% solve(S11) %*% (befi[[marker]] - mu1))
    data1LM$Xhats[i] <- XhatLMi
  } else cat("No longitudinal data found before LM for patient", patLM[i], "\n")
}

# Fit landmark Cox model with Xhat(s|s) with administrative censoring at LM+width
data1LM$survyrshor <- pmin(data1LM$survyrs, LM+width)
data1LM$statushor <- data1LM$status
data1LM$statushor[data1LM$survyrs > LM+width] <- 0
cXhats <- coxph(Surv(survyrshor, statushor) ~ Xhats, data=data1LM)
summary(cXhats)
```

```
## Call:
## coxph(formula = Surv(survyrshor, statushor) ~ Xhats, data = data1LM)
##
##   n= 229, number of events= 46
##
##            coef exp(coef)  se(coef)      z Pr(>|z|)
## Xhats -0.044692  0.956292  0.009089 -4.917 8.79e-07 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##       exp(coef) exp(-coef) lower .95 upper .95
## Xhats    0.9563      1.046    0.9394    0.9735
##
## Concordance= 0.702  (se = 0.038 )
## Likelihood ratio test= 23.65  on 1 df,   p=1e-06
## Wald test            = 24.18  on 1 df,   p=9e-07
## Score (logrank) test = 23.87  on 1 df,   p=1e-06
```

4.3.2 Cross-validation

Again, the cross-validated predicted probabilities are calculated and stored.

```
predXhatsCV <- rep(NA, nLM)
for (i in 1:nLM)
{

  pati <- patLM[i]

  #
  # Calculate Xhat(s|s) for all subjects in data minus subject i
  #
  cls1mini <- subset(cls1, patid != pati)
  cls2mini <- subset(cls2, patid != pati)

  tmp <- makeLMdata(data1=cls1mini, data2=cls2mini, id="patid",
                    time1="survyrs", time2="measyrs", LM=LM)
  data1LMmini <- tmp$data1LM
  data2LMbefmini <- tmp$data2LMbef
  data2mini <- cls2mini
  patLMmini <- data1LMmini$patid
  nLMmini <- nrow(data1LMmini)

  # Refit Gaussian process
  exp1 <- corExp(form= ~ measyrs * treat | patid, nugget=TRUE)
  exp1 <- Initialize(exp1, data2mini)
  fitLME.GP <- lme(prothr ~  measyrs * treat, random = ~ 1 | patid,
                   data = data2mini, correlation = exp1)
  # Extract estimated parameters
  varcorr <- VarCorr(fitLME.GP)
  rangenugget <- coef(fitLME.GP$modelStruct$corStruct, unconstrained = FALSE)
  ss1 <- as.numeric(varcorr[1, 1])
  nugget <- rangenugget[2]
  range <- rangenugget[1]
  ss2 <- as.numeric(varcorr[2, 1]) * (1-nugget)
  ss3 <- as.numeric(varcorr[2, 1]) * nugget
```

```
bet <- fitLME.GP$coefficients$fixed
apl <- bet[1]
bpl <- bet[2]
apr <- bet[1] + bet[3]
bpr <- bet[2] + bet[4]

# Loop for everyone in data1LMmini
for (j in 1:nLMmini) {
  patj <- patLMmini[j]
  whj <- which(data1LMmini[[id]]==patj)
  if (length(whj) > 0) {
    data1LMj <- data1LMmini[whj, ]
    a <- ifelse(data1LMj$treat[1]=="Placebo", apl, apr)
    b <- ifelse(data1LMj$treat[1]=="Placebo", bpl, bpr)
    befj <- data2LMbefmini[data2LMbefmini[[id]]==patj, ]
    t1 <- befj[[time2]]
    n1 <- length(t1)
    mu1 <- unlist(lapply(t1, meanf, a=a, b=b))
    mu2 <- meanf(t=LM, a=a, b=b)
    S11 <- outer(t1, t1, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
    S12 <- outer(t1, LM, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
    XhatLMj <- mu2 + as.vector(t(S12) %*% solve(S11) %*% (befj[[marker]] - mu1))
    data1LMmini$Xhats[j] <- XhatLMj
  }
}

# Administrative censoring at LM+width
data1LMmini$survyrshor <- pmin(data1LMmini$survyrs, LM+width)
data1LMmini$statushor <- data1LMmini$status
data1LMmini$statushor[data1LMmini$survyrs > LM+width] <- 0
cXhatsmini <- coxph(Surv(survyrshor, statushor) ~ Xhats, data=data1LMmini)

#
# Apply model to data of subject i
#
data1LMi <- subset(data1LM, patid==pati)
# Calculate Xhat(s|s) for subject i and add to data1LMi
a <- ifelse(data1LMi$treat[1]=="Placebo", apl, apr)
b <- ifelse(data1LMi$treat[1]=="Placebo", bpl, bpr)
befi <- data2LMbef[data2LMbef[[id]]==pati, ]
t1 <- befi[[time2]]
n1 <- length(t1)
mu1 <- unlist(lapply(t1, meanf, a=a, b=b))
mu2 <- meanf(t=LM, a=a, b=b)
S11 <- outer(t1, t1, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
S12 <- outer(t1, LM, covarf, ss1=ss1, ss2=ss2, ss3=ss3, lambda=lambda)
data1LMi$Xhats <- mu2 + as.vector(t(S12) %*% solve(S11) %*% (befi[[marker]] - mu1))
```

```
  sfi <- survfit(cXhatsmini, newdata=data1LMi)
  predXhatsCV[i] <- summary(sfi, times = LM+width)$surv
}

# Add new predictions to image
resave(predXhatsCV, file="predictionsCV.Rdata")
```

4.4 Revival

A new idea is to calculate $\hat{X}(t \mid s) = E(X(t)|\overline{X}(s), T > s)$ from the revival model. The model itself is described in detail in the paper by Dempsey & McCullagh (Lifetime Data Analysis 2018) and the landmarking 2.0 paper. Motivation for the models to be fitted is found in Dempsey & McCullagh (2018). Our approach differs from Dempsey & McCullagh (2018) in that we define a horizon τ and fits separate models for subjects that died before τ and for subjects that are alive and under follow-up at τ. Subjects censored before τ are not included in the models, but they are included in the predictions and their evaluations.

4.4.1 No cross-validation

Here I put down the necessary steps to calculate the predictable time-dependent covariate $\hat{X}(t \mid s)$ based on revival. Again it is convenient to put the code for estimating the revival model in a function. See the paper for explanation.

```
fitrevival <- function(data1, data2, tau)
{
  # data1 and data2 are the survival and longitudinal data, respectively
  # tau is the horizon
  data2 <- merge(data2, data1[, c("patid", "survyrs", "status")], by="patid", all=TRUE)
  # Define groups of "Dead" and "Survivor"
  data2$group <- NA
  data2$group[data2$status==1 & data2$survyrs<tau] <- 1
  data2$group[data2$survyrs>=tau] <- 0
  data2$group <- factor(data2$group, levels=0:1, labels=c("Survivor", "Dead"))

  year <- 365.25
  #
  # Fit longitudinal model in reverse time, for subjects that died before horizon
  #
  data <- subset(data2, group=="Dead")
  data$revyrs <- data$survyrs - data$measyrs
  data$lgtplus1 <- log(data$revyrs + 1/year)
  data$prednisone <- as.numeric(data$treat=="Prednisone")
  data <- subset(data, !is.na(revyrs)) # remove empty rows

  exp1 <- corExp(form= ~ revyrs | patid, nugget=TRUE)
  exp1 <- Initialize(exp1, data)
  fitLME.GP <- lme(prothr ~ prednisone + survyrs + revyrs + lgtplus1,
                   random = ~ 1 | patid, data = data, correlation = exp1)
```

```
  # Extract and save estimated parameters
  varcorr <- VarCorr(fitLME.GP)
  rangenugget <- coef(fitLME.GP$modelStruct$corStruct, unconstrained = FALSE)
  ss1 <- as.numeric(varcorr[1, 1])
  nugget <- rangenugget[2]
  range <- rangenugget[1]
  ss2 <- as.numeric(varcorr[2, 1]) * (1-nugget)
  ss3 <- as.numeric(varcorr[2, 1]) * nugget

  estimates <- c(fitLME.GP$coefficients$`fixed`, ss1, ss2, ss3, 1/range)
  names(estimates) <- c("alpha", "predn", "event_t", "u",
                        "log(u+1)", "ss1", "ss2", "ss3", "lambda")
  estimates1 <- estimates # save

  #
  # Fit longitudinal model in reverse time, for subjects that are alive at time tau
  #
  # data <- subset(data2, status==0 & survyrs>=tau & revyrs>=0)
  data <- subset(data2, group=="Survivor")
  data$revyrs <- tau - data$measyrs
  data <- subset(data, revyrs>0) # remove measurements after tau
  data$lgtplus1 <- log(data$revyrs + 1/year)
  data$prednisone <- as.numeric(data$treat=="Prednisone")

  exp1 <- corExp(form= ~ revyrs | patid, nugget=TRUE)
  exp1 <- Initialize(exp1, data)
  fitLME.GP <- lme(prothr ~ prednisone + revyrs + lgtplus1,
                   random = ~ 1 | patid, data = data, correlation = exp1)
  # Extract and save estimated parameters
  varcorr <- VarCorr(fitLME.GP)
  rangenugget <- coef(fitLME.GP$modelStruct$corStruct, unconstrained = FALSE)
  ss1 <- as.numeric(varcorr[1, 1])
  nugget <- rangenugget[2]
  range <- rangenugget[1]
  ss2 <- as.numeric(varcorr[2, 1]) * (1-nugget)
  ss3 <- as.numeric(varcorr[2, 1]) * nugget

  estimates <- c(fitLME.GP$coefficients$`fixed`, ss1, ss2, ss3, 1/range)
  names(estimates) <- c("alpha", "predn", "u", "log(u+1)",
                        "ss1", "ss2", "ss3", "lambda")
  estimates0 <- estimates # save

  return(list(estimates0=estimates0, estimates1=estimates1))
}
```

We take $\tau = 9$ years for our horizon, and obtain the following estimates.

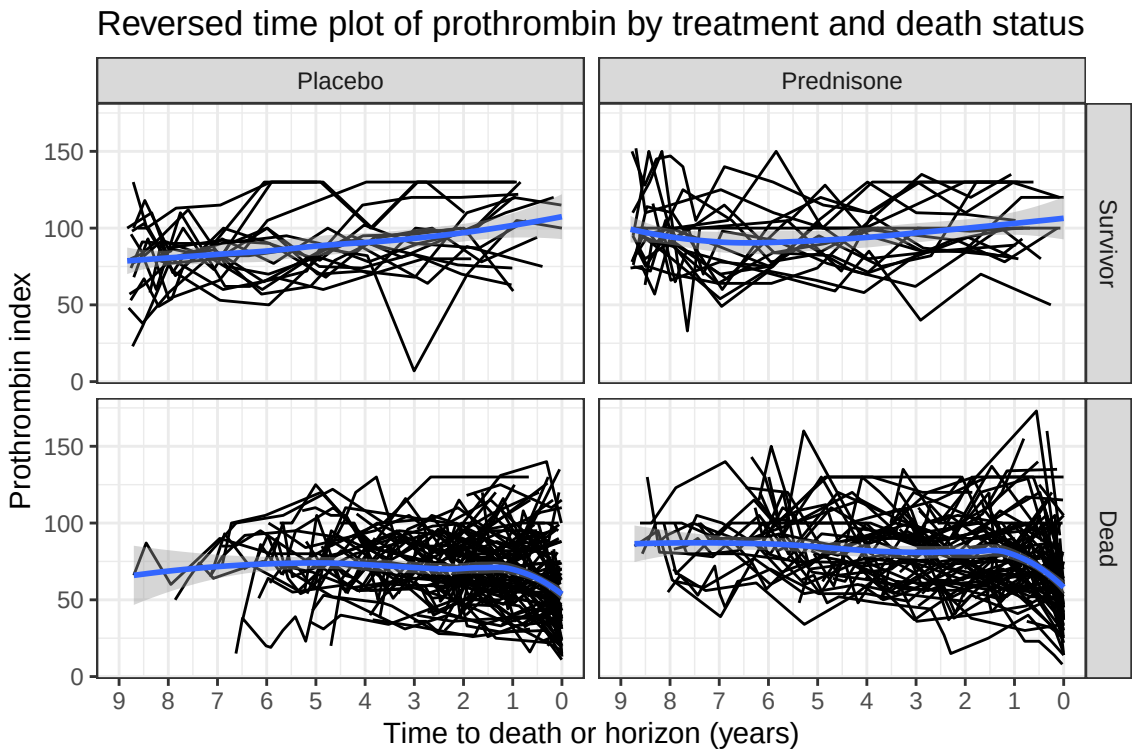
```
tau <- 9
estimates <- fitrevival(data1=cls1, data2=cls2, tau=tau)
print(estimates, digits=4)
```

```
## $estimates0
##    alpha    predn        u log(u+1)      ss1      ss2      ss3   lambda
##  95.8546   9.5292  -1.3879  -1.6472 202.4106 191.0657 201.3448   0.3519
##
## $estimates1
##    alpha    predn  event_t        u log(u+1)      ss1      ss2      ss3   lambda
##  66.3944   8.3655   1.7302  -1.7922   4.5783 221.5270 243.5791 161.8725   0.6157
```

```
save(estimates, file="estimates.Rdata")
```

Below is a "reverse spaghetti plot" of the biomarkers. It shows the biomarkers, along with a loess smoother, backwards in time from the time of death (for those that died within the horizon τ), or from the horizon τ (for those that were still alive at τ).

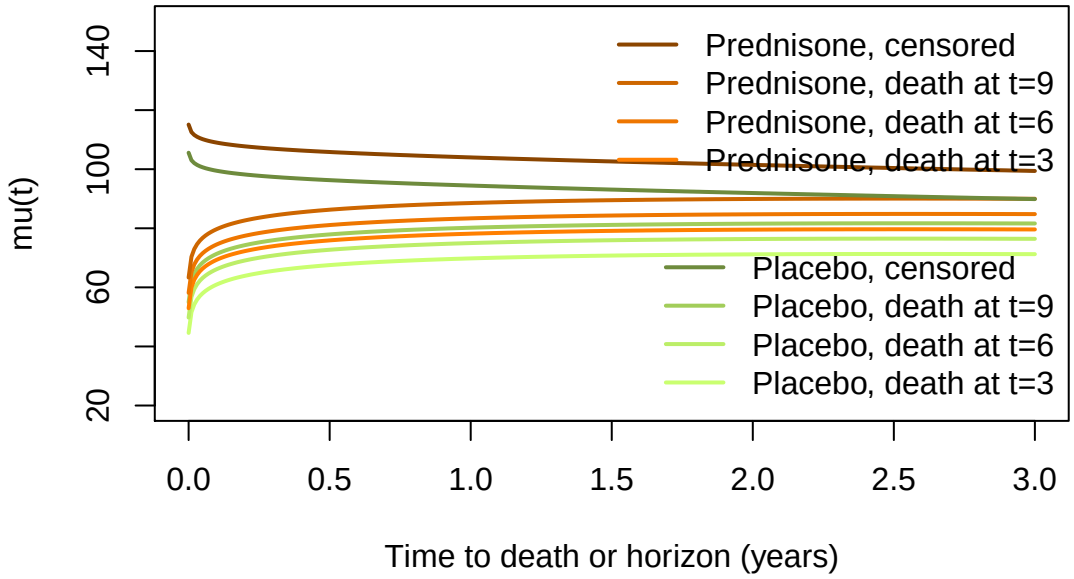
```
cls2plus <- merge(cls2, cls1[, c("patid", "survyrs", "status")], by="patid", all=TRUE)
cls2plus$revyrs <- cls2plus$survyrs - cls2plus$measyrs
cls2plus$revyrs[cls2plus$status==0] <- tau - cls2plus$measyrs[cls2plus$status==0]
cls2plus <- subset(cls2plus, !is.na(treat))
cls2plus$statuscat <- factor(cls2plus$status,
                             levels=0:1, labels=c("Survivor", "Dead"))
cls2plus <- subset(cls2plus, (status==1 & survyrs<tau & revyrs>=0) |
                     (status==0 & survyrs>=tau & revyrs>=0))

a <- cls2plus %>%
  ggplot(aes(x = revyrs, y = prothr, group = patid)) +
  geom_line() +
  stat_smooth(aes(group=1)) +
  facet_grid(cols=vars(treat), rows=vars(statuscat)) +
  # xlim(0, 9) +
  scale_x_continuous(trans = "reverse", limits = c(9, 0),
                     breaks = 0:9, labels=0:9) +
  theme_bw() +
  labs(title="Reversed time plot of prothrombin by treatment and death status") +
  xlab("Time to death or horizon (years)") +
  ylab("Prothrombin index")
a
```



In order to demonstrate the fit of the model I will now make a plot of the mean function (in reverse time) for both treatments, for patients that died at 3, 6, and 9 years, and for patients alive at $\tau = 9$ years.

```
ttseq <- seq(0, 3, by=0.01)
year <- 365.25
# Calculate mean for the patients that died before tau
estimates1 <- estimates$estimates1
alp <- estimates1[1]
contr.predn <- estimates1[2]
b0 <- estimates1[3]
b1 <- estimates1[4]
b2 <- estimates1[5]
meanseq03 <- alp + b0*3 + b1*ttseq + b2*log(ttseq + 1/year)
meanseq06 <- alp + b0*6 + b1*ttseq + b2*log(ttseq + 1/year)
meanseq09 <- alp + b0*9 + b1*ttseq + b2*log(ttseq + 1/year)
meanseq13 <- alp + contr.predn + b0*3 + b1*ttseq + b2*log(ttseq + 1/year)
meanseq16 <- alp + contr.predn + b0*6 + b1*ttseq + b2*log(ttseq + 1/year)
meanseq19 <- alp + contr.predn + b0*9 + b1*ttseq + b2*log(ttseq + 1/year)
# Calculate mean for the patients that were alive at tau
estimates0 <- estimates$estimates0
alp <- estimates0[1]
contr.predn <- estimates0[2]
b1 <- estimates0[3]
b2 <- estimates0[4]
meanseq0 <- alp + b1*ttseq + b2*log(ttseq + 1/year)
meanseq1 <- alp + contr.predn + b1*ttseq + b2*log(ttseq + 1/year)
```

```
# Do the plot
plot(ttseq, meanseq03, type="l", lwd=2, col=colors()[86],
     ylim=c(20, 150),
     xlab="Time to death or horizon (years)", ylab="mu(t)")
lines(ttseq, meanseq06, type="l", lwd=2, col=colors()[87])
lines(ttseq, meanseq09, type="l", lwd=2, col=colors()[88])
lines(ttseq, meanseq13, type="l", lwd=2, col=colors()[91])
lines(ttseq, meanseq16, type="l", lwd=2, col=colors()[92])
lines(ttseq, meanseq19, type="l", lwd=2, col=colors()[93])
lines(ttseq, meanseq0, type="l", lwd=2, col=colors()[89])
lines(ttseq, meanseq1, type="l", lwd=2, col=colors()[94])
legend("bottomright", rev(c("Placebo, death at t=3", "Placebo, death at t=6", "Placebo, death a
       lwd=2, col=colors()[89:86], bty="n")
legend("topright", rev(c("Prednisone, death at t=3", "Prednisone, death at t=6", "Prednisone, d
       lwd=2, col=colors()[94:91], bty="n")
```



Direct revival dynamic prediction probabilities of dying at time $u > s$ are given by Bayes' rule:

$$P(T = u \,|\, T > s, \overline{X}(s)) = \frac{P(\overline{X}(s) \,|\, T = u, T > s) \cdot P(T = u \,|\, T > s)}{\sum_{u' > s} P(\overline{X}(s) \,|\, T = u', T > s) \cdot P(T = u' \,|\, T > s)}.$$

Note that $P(\overline{X}(s) \,|\, T = u, T > s)$ can be simplified to $P(\overline{X}(s) \,|\, T = u)$. Here we want to extract the conditional expectations of $X(t)$, given the observed history until time $\overline{X}(s)$, and given $T \geq t$. This conditional expectation is given by

$$E\left(X(t) \,|\, T \geq t, \overline{X}(s)\right) = \int_t^\tau E\left(X(t) \,|\, T = u, \overline{X}(s)\right) P(T = u \,|\, T \geq t, \overline{X}(s)) du.$$

For $P(T = u \,|\, T \geq t, \overline{X}(s))$ we again Bayes' rule, giving

$$P(T = u \,|\, T \geq t, \overline{X}(s)) = \frac{P(\overline{X}(s) \,|\, T = u) \cdot P(T = u \,|\, T \geq t)}{\sum_{u' \geq t} P(\overline{X}(s) \,|\, T = u') \cdot P(T = u' \,|\, T \geq t)}.$$

Furthermore, $E\left(X(t)\,|\,T=u,\overline{X}(s)\right)$ can be obtained from the joint distribution of $(\overline{X}(s),\overline{X}(s,u])$, given $T=u$. Here $\overline{X}(s,u]$ refers to the vector of $X(t)$'s for the event times t in $(s,u]$. If we denote this distribution as multivariate normal with mean vector $\begin{pmatrix}\mu_s\\ \mu_{su}\end{pmatrix}$, and covariance matrix $\begin{pmatrix}\Sigma_{ss} & \Sigma_{s,su}\\ \Sigma_{su,s} & \Sigma_{su,su}\end{pmatrix}$, then we obtain

$$E\left(\overline{X}(s,u]\,|\,T=u,\overline{X}(s)\right)=\mu_{su}+\Sigma_{su,s}\Sigma_{ss}^{-1}\left(\overline{X}(s)-\mu_s\right).$$

That leaves us with the following procedure: loop over event time points $u>s$, including $u=\tau$

- Calculate $P(T=u\,|\,T\geq t,\overline{X}(s))$
- Calculate conditional expectations and variances, given $T=u$, of $(\overline{X}(s),\overline{X}(s,u])$, yielding expectation vector $\begin{pmatrix}\mu_s\\ \mu_{su}\end{pmatrix}$ and covariance matrix $\begin{pmatrix}\Sigma_{ss} & \Sigma_{s,su}\\ \Sigma_{su,s} & \Sigma_{su,su}\end{pmatrix}$
- Calculate $E\left(\overline{X}(s,u]\,|\,T=u,\overline{X}(s)\right)=\mu_{su}+\Sigma_{su,s}\Sigma_{ss}^{-1}\left(\overline{X}(s)-\mu_s\right)$
- Combine elements $E\left(X(t)\,|\,T=u,\overline{X}(s)\right)$ of these with $P(T=u\,|\,T\geq t,\overline{X}(s))$ and sum over $u\in(t,\tau]$ to obtain $E\left(X(t)\,|\,T\geq t,\overline{X}(s)\right)$

The code below implements a function that, given estimates of the revival process (in `estimates`) and given an estimate of the marginal distribution of T (estimated below by separate Kaplan-Meiers by treatment), calculates, for subject `pat`, the predictable time-dependent covariate $\hat{X}(t|s)$ for all event time points $>s$, the landmark time point. .

```
# Overall Kaplan-Meier (for time points only)
sfkm <- survfit(Surv(data1[[time1]], data1[[status1]]) ~ 1)
km <- summary(sfkm)
# Separate Kaplan-Meiers per treatment group
sfkmtreat <- survfit(Surv(data1[[time1]], data1[[status1]]) ~ data1[["treat"]])

doXhats <- function(LM, data1LM, data2LM, pat, estimates,
                    sfkm, sfkmtreat, data1i, data2i) {
  # Input parameters are the same as in previous functions, with in addition
  # sfkm: overall Kaplan-Meier
  # sfkmtreat: Kaplan-Meier per treatment arm
  # data1i, data2i: only used with cross-validation to contain
  #   survival and longitudinal data of subject i (estimates in that
  #   case obtained from all data expect subject i)
  #
  # Calculates E(X(t) | T >= t, \overline{X}(s))
  # Workflow:
  #   A: Direct revival probabilities P( T=t | T > s, \overline{X}(s) )
  #   B: E( X(t) | T=u, \overline{X}(s) )
  #   C: P( T=u | \overline{X}(s) )
  #
```

```r
estimates0 <- estimates$estimates0
estimates1 <- estimates$estimates1
if (missing(data1i)) data1i <- data1LM[data1LM[[id]] == pat, ]
if (missing(data2i)) data2i <- data2LM[data2LM[[id]] == pat, ]
ti <- data1i[[time1]]

# Use sfkm first just to extract time points
km <- summary(sfkm)
tt <- km$time[km$time > LM]
tt <- tt[tt < tau]
nt <- length(tt)

#
#   --- A ---
#
# P(T = t | T > s) first, for direct revival probabilities
#
# Since we have separate KM's for the two treatments, we will have two
# conditional probability distributions
if (data1i$treat=="Placebo") { # The placebo arm
  sfkm <- sfkmtreat[1] # overwrite sfkm to now mean survival of appropriate treatment group
} else { # The prednisone arm
  sfkm <- sfkmtreat[2]
}
survLM <- summary(sfkm, times=LM)$surv
tmp <- summary(sfkm, times=tt)$surv / survLM
condprob <- -diff(c(1, tmp))
condprob <- data.frame(time=tt, condprob=condprob)
# Note that these conditional probabilities do not sum up to 1
# The remainder is reserved for tau, the probality that P(T > tau | T > LM)
condprobtau <- 1 - sum(condprob$condprob)

# Marker values
markeri <- data2i[[marker]]
ni <- nrow(data2i)
Sigma1 <- matrix(1, ni, ni)
Sigma3 <- diag(ni)

#
# P(T=t | T > s, \overline{X}(s))
#
# We first need to make a further step, in going from P(T=t | T>s), which is
# currently contained in condprob and condprobtau, to P(T=t | T>s, \overline{X}(s))
# In the process we will save those predicted probabilities, since they are
# of interest in themselves; these are the original revival dynamic predictions
#
# In the process, we save P(\overline{X}(s) | T = t) and P(\overline{X}(s) | T > tau);
```

```r
# in PXs_given_T; these are used later as well
PXs_given_T <- rep(NA, nt+1)

# First loop over the death time points
# (this will use the longitudinal model based on the patients that died)
a0 <- estimates1[1]
if (data1i$treat=="Prednisone")
  a0 <- a0 + estimates1[2] # if prednisone patient, add predn contrast
b0 <- estimates1[3]
b1 <- estimates1[4]
b2 <- estimates1[5]
ss1 <- estimates1[6]
ss2 <- estimates1[7]
ss3 <- estimates1[8]
lam <- estimates1[9]
meanf <- function(t, ti, a0, b0, b1, b2)
  return( a0 + b0*ti + b1*t + b2*log(t+1/year) )
for (j in 1:nt) {
  revtime <- tt[j] - data2i[[time2]]
  Sigma2 <- exp( -lam*abs(outer(revtime, revtime, "-")) )
  Sigma <- ss1 * Sigma1 + ss2 * Sigma2 + ss3 * Sigma3
  PXs_given_T[j] <- dmvnorm(markeri,
                    mean=meanf(revtime, ti=ti, a0=a0, b0=b0, b1=b1, b2=b2),
                    sigma=Sigma)
}
# And for tau (longitudinal model based on the surviving patients)
a0 <- estimates0[1]
if (data1i$treat=="Prednisone")
  a0 <- a0 + estimates0[2] # if prednisone patient, add predn contrast
b1 <- estimates0[3]
b2 <- estimates0[4]
ss1 <- estimates0[5]
ss2 <- estimates0[6]
ss3 <- estimates0[7]
lam <- estimates0[8]
meanf <- function(t, a0, b1, b2) return( a0 + b1*t + b2*log(t+1/year) )
revtime <- tau - data2i[[time2]]
Sigma2 <- exp( -lam*abs(outer(revtime, revtime, "-")) )
Sigma <- ss1 * Sigma1 + ss2 * Sigma2 + ss3 * Sigma3
PXs_given_T[nt+1] <- dmvnorm(markeri, mean=meanf(revtime, a0=a0, b1=b1, b2=b2), sigma=Sigma)

# For direct revival probs multiply with P(T = t | T > s) and P(T > tau | T > s)
revprobs <- PXs_given_T * c(condprob$condprob, condprobtau)
# Finally, obtain all probabilities by dividing by numerator
revprobs <- revprobs / sum(revprobs)

#
```

```r
# End direct revival probabilities, we now continue with deriving
# E( X(t) | T >= t, \overline{X}(s) )
#
#   --- B ---
#
# Matrix res (upper diagonal) will contain E( X(t) | T=u, \overline{X}(s) ),
# for s < t \leq u, with t (rows) and u (columns) varying over all
# event time points > s, including the horizon tau (so length nt+1)
res <- matrix(0, nt+1, nt+1)
# First loop over the death time points
# (this will use the longitudinal model based on the patients that died)
a0 <- estimates1[1]
if (data1i$treat=="Prednisone")
  a0 <- a0 + estimates1[2] # if prednisone patient, add predn contrast
b0 <- estimates1[3]
b1 <- estimates1[4]
b2 <- estimates1[5]
ss1 <- estimates1[6]
ss2 <- estimates1[7]
ss3 <- estimates1[8]
lam <- estimates1[9]
meanf <- function(t, ti, a0, b0, b1, b2)
  return( a0 + b0*ti + b1*t + b2*log(t+1/365.25) )
tbef <- data2i[[time2]]
nbef <- length(tbef)
for (j in 1:nt) { # Loop over u, with s < u < tau
  u <- tt[j]
  # Calculate E(X(t) | T=u, \overline{X}(s)), for all s < t \leq u
  # First calculate full joint distribution of (Xbef, Xaft)
  taft <- tt[tt<=u]
  naft <- length(taft)
  revtime <- u - c(tbef, taft)
  ni <- nbef + naft
  Sigma1 <- matrix(1, ni, ni)
  Sigma3 <- diag(ni)
  Sigma2 <- exp( -lam*abs(outer(revtime, revtime, "-")) )
  Sigma <- ss1 * Sigma1 + ss2 * Sigma2 + ss3 * Sigma3
  mu <- meanf(revtime, ti=ti, a0=a0, b0=b0, b1=b1, b2=b2)
  mu1 <- mu[1:nbef]
  mu2 <- mu[(nbef+1):ni]
  Sigma11 <- Sigma[1:nbef, 1:nbef]
  Sigma21 <- Sigma[(nbef+1):ni, 1:nbef]
  # Finally the conditional expectation
  tmp <- mu2 + Sigma21 %*% solve(Sigma11) %*% (markeri - mu1)
  # Store
  res[(1:j), j] <- tmp
}
```

```r
# And for tau (longitudinal model based on the surviving patients)
a0 <- estimates0[1]
if (data1i$treat=="Prednisone")
  a0 <- a0 + estimates0[2] # if prednisone patient, add predn contrast
b1 <- estimates0[3]
b2 <- estimates0[4]
ss1 <- estimates0[5]
ss2 <- estimates0[6]
ss3 <- estimates0[7]
lam <- estimates0[8]
meanf <- function(t, a0, b1, b2) return( a0 + b1*t + b2*log(t+1/365.25) )
u <- tau
taft <- c(tt[tt<=u], tau)
naft <- length(taft)
revtime <- u - c(tbef, taft)
ni <- nbef + naft
Sigma1 <- matrix(1, ni, ni)
Sigma3 <- diag(ni)
Sigma2 <- exp( -lam*abs(outer(revtime, revtime, "-")) )
Sigma <- ss1 * Sigma1 + ss2 * Sigma2 + ss3 * Sigma3
mu <- meanf(revtime, a0=a0, b1=b1, b2=b2)
mu1 <- mu[1:nbef]
mu2 <- mu[(nbef+1):ni]
Sigma11 <- Sigma[1:nbef, 1:nbef]
Sigma21 <- Sigma[(nbef+1):ni, 1:nbef]
# Finally the conditional expectation
tmp <- mu2 + Sigma21 %*% solve(Sigma11) %*% (markeri - mu1)
# Store
res[, nt+1] <- c(tmp)

#
# Matrix res (upper diagonal) now contains E( X(t) | T=u, \overline{X}(s) ),
# for s < t \leq u, with t (rows) and u (columns) varying over all
# event time points > s, including the horizon tau (so length nt+1)
#
# Now to obtain \hat{X}(t | s), by multiplying, for each u,
# E( X(t) | T=u, \overline{X}(s) ) with P(T=u | T>=t, \overline{X}(s) )
# (or P(T > tau | T>t, \overline{X}(s))), and sum over u.
#
#   --- C ---
#
# These probabilities P(T=u | T>=t, \overline{X}(s) ) are calculated
# similarly to the direct revival probabilities; the only difference is that
# conditioning is on T>=t, rather than T>s
#
mu <- rep(NA, nt+1) # save space; single index represents t in \hat{X}(t | s)
for (j in 1:nt) { # index is over t < tau
```

```
    # First P(T = u | T >= t)
    survt <- summary(sfkm, times=tt[j] - 1e-6)$surv
    tmp <- summary(sfkm, times=tt[tt >= tt[j]])$surv / survt
    condprob <- -diff(c(1, tmp))
    condprob <- data.frame(time=tt[tt >= tt[j]], condprob=condprob)
    condprobtau <- 1 - sum(condprob$condprob) # remainder is prob that P(T > tau | T >= t)
    # Multiply each P(T = u | T >= t) with P(\overline{X}(s) | T = u)
    # to obtain P(T=u | T>=t, \overline{X}(s) ); save in pp
    pp <- PXs_given_T[j:(nt+1)] * c(condprob$condprob, condprobtau)
    pp <- pp / sum(pp)
    # Finally multiply E( X(t) | T=u, \overline{X}(s) ) with P(T=u | T>=t, \overline{X}(s) )
    mu[j] <- sum(res[j, (j:(nt+1))] * pp)
  }
  # Finally, t = tau
  mu[nt+1] <- res[nt+1, nt+1]
  return(list(tt=tt, res=res, revprobs=revprobs, pat=pat, LM=LM,
              tbef=tbef, Xbef=markeri,
              Xhat=data.frame(time=c(tt, tau), mu=mu)))
}
```

We also make a function, `makeplot`, that plots the resulting $\hat{X}(t|s)$, along with its components $E\left(X(t)\,|\,T=u, \overline{X}(s)\right)$, based on the output of `doXhats`.

```
makeplot <- function(mures, width=Inf) {
  # Makes a plot of the \hat{X}(t|s), along with its components and the measurements before s
  # Input is mures, obtained from a call of doXhats
  pat <- mures$pat
  res <- mures$res
  revprobs <- mures$revprobs
  tt <- mures$tt
  LM <- mures$LM
  tbef <- mures$tbef
  Xbef <- mures$Xbef
  mu <- mures$Xhat$mu

  nt <- length(tt)

  #
  # Layout: large plot with prothrombin above, smaller one with probabilities below
  #
  layout(matrix(1:2, 2, 1), widths=c(1), heights=c(2, 1))

  #
  # Prothrombin
  #
  par(mar = c(0.1, 4, 2, 1))
  # Actual observations first
  plot(tbef, Xbef, type="b", pch=1, xlim=c(0, tau), ylim=c(20, 140),
```

```
       ylab="Prothrombin", axes=FALSE)
  axis(2)
  box()
  abline(v=LM, lty=3)
  # Conditional expectations given event-free at tau
  lines(tt[1:(nt+1)], res[1:(nt+1), nt+1], type="l", col="#2ca1db") # blue
  # Conditional expectations given event at tj < tau
  for (j in nt:1)
    lines(tt[1:j], res[1:j, j], type="l", col="#dfb78e") # 8
  # Overall prediction estimate (thicker)
  lines(c(tt, tau), mu, lwd=2, col="#112047")
  title(main = paste("Patient", pat))

  #
  # Conditional probabilities at the bottom
  #
  par(mar = c(3, 4, 0, 1))
  plot(c(LM, tt, tau), 1 - cumsum(c(0, revprobs)), type="s",
       xlim=c(0, tau), xlab="Years since randomisation",
       ylab="Survival")
  abline(v=LM, lty=3)

  return(invisible())
}
```

Finally, let's illustrate this for a couple of patients.

```
data1LM <- data1[data1[[time1]] > LM, ]
dim(data1LM)
```

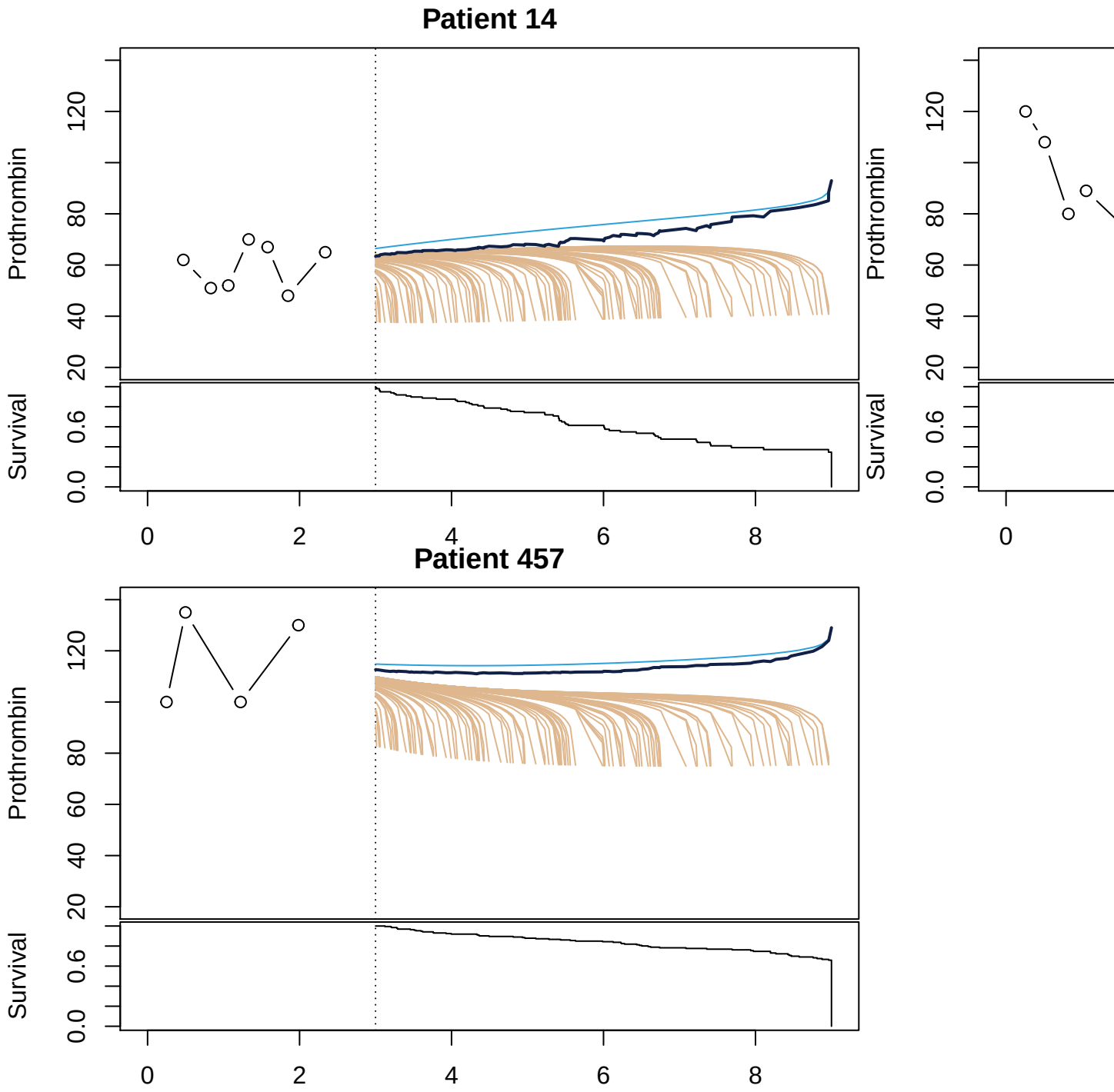
```
## [1] 229   4
```

```
table(data1LM$treat)
```

```
##
##    Placebo Prednisone
##        110        119
```

```
data2LM <- data2[data2[[time2]] <= LM, ]
data2LM <- data2LM[!is.na(data2LM[[id]]), ]
pats <- sort(unique(data1LM[[id]]))

for (k in c(5, 30, 177)) {
  pat <- pats[k]
  muk <- doXhats(LM, data1LM, data2LM, pat, estimates,
                 sfkm, sfkmtreat)
  makeplot(muk)
  # pdf(paste("Xhatsrev", pats[k], ".pdf", sep="")) # save pdf for the paper
  # makeplot(muk)
  # dev.off()
```

```
}
```



Now, to use the $\hat{X}(t|s)$ based on revival in landmarking 2.0, we need a function, similar to `mklongdata` for the landmarking 2.0 based on the Gaussian process.

```
mklongdata_revival <- function(data1LM, data2LMbef, id, ttLM, time1, status1, time2,
                               marker, estimates, sfkm, sfkmtreat, LM, width)
{
  # Similar to mklongdata, input parameters are the same as mklongdata and doXhats
```

```
  patLM <- sort(unique(data1LM[[id]]))
  nLM <- nrow(data1LM)
  # Prepare structure for time-dependent covariate \overline{X}(t)
  dataLM <- survSplit(Surv(data1LM[[time1]], data1LM[[status1]]) ~ .,
                      data=data1LM, cut=ttLM)
  dataLM$tstart[dataLM$tstart<LM] <- LM # repair tstart=0
  # Administrative censoring at LM+width
  whcens <- which(dataLM$tstop > LM+width)
  dataLM$tstop[whcens] <- LM+width

  estimates0 <- estimates$estimates0
  estimates1 <- estimates$estimates1
  dataLM$Xhat <- NA
  for (i in 1:length(patLM)) {
    # deb(i, method="cat")
    pat <- patLM[i]
    Xhat <- doXhats(LM, data1LM, data2LM, pat, estimates,
                    sfkm, sfkmtreat)$Xhat
    whid <- which(dataLM[[id]]==pat)
    nid <- length(whid)
    whLMw <- which(Xhat$time <= LM + width)
    nLMw <- length(whLMw)
    # deb(c(nid, nLMw), method="cat")
    if (nid < nLMw) dataLM$Xhat[whid] <- Xhat$mu[whLMw[1:nid]]
    else if (nid == nLMw + 1) { # LM + width not included in time points of whid
      dataLM$Xhat[whid[-nid]] <- Xhat$mu[whLMw]
      dataLM$Xhat[whid[nid]] <- Xhat$mu[whLMw[nLMw]]
    } else
      dataLM$Xhat[whid] <- Xhat$mu[whLMw]
  }

  return(dataLM)
}
```

Now we apply this to the CLS data.

```
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
data1LM <- tmp$data1LM
data2LMbef <- tmp$data2LMbef
data2LMaft <- tmp$data2LMaft
patLM <- sort(unique(data1LM[[id]]))
nLM <- nrow(data1LM)
ttLM <- sort(unique(data1LM$survyrs[data1LM$status==1 & data1LM$survyrs <= LM+width]))

dataLM_revival <- mklongdata_revival(data1LM=data1LM, data2LMbef=data2LMbef, id="patid",
                      ttLM = ttLM,
                      time1="survyrs", status1="status",
```

```
                    time2="measyrs", marker="prothr",
                    estimates, sfkm, sfkmtreat,
                    LM=LM, width=width)
```

Now I will fit the time-dependent Cox model based on the $\hat{X}(t|s)$ obtained above by revival.

```
ctdrev <- coxph(Surv(tstart, tstop, event) ~ Xhat, data=dataLM_revival)
summary(ctdrev)
```

```
## Call:
## coxph(formula = Surv(tstart, tstop, event) ~ Xhat, data = dataLM_revival)
##
##   n= 8974, number of events= 113
##
##           coef exp(coef)  se(coef)      z Pr(>|z|)
## Xhat -0.040884  0.959941  0.006901 -5.925 3.13e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## Xhat    0.9599      1.042     0.947     0.973
##
## Concordance= 0.68  (se = 0.029 )
## Likelihood ratio test= 34.63  on 1 df,   p=4e-09
## Wald test            = 35.1  on 1 df,   p=3e-09
## Score (logrank) test = 34.64  on 1 df,   p=4e-09
```

4.4.2 Cross-validation

Finally, I am going to obtain the (cross-validated) predicted probabilities obtained from revival landmarking 2.0.

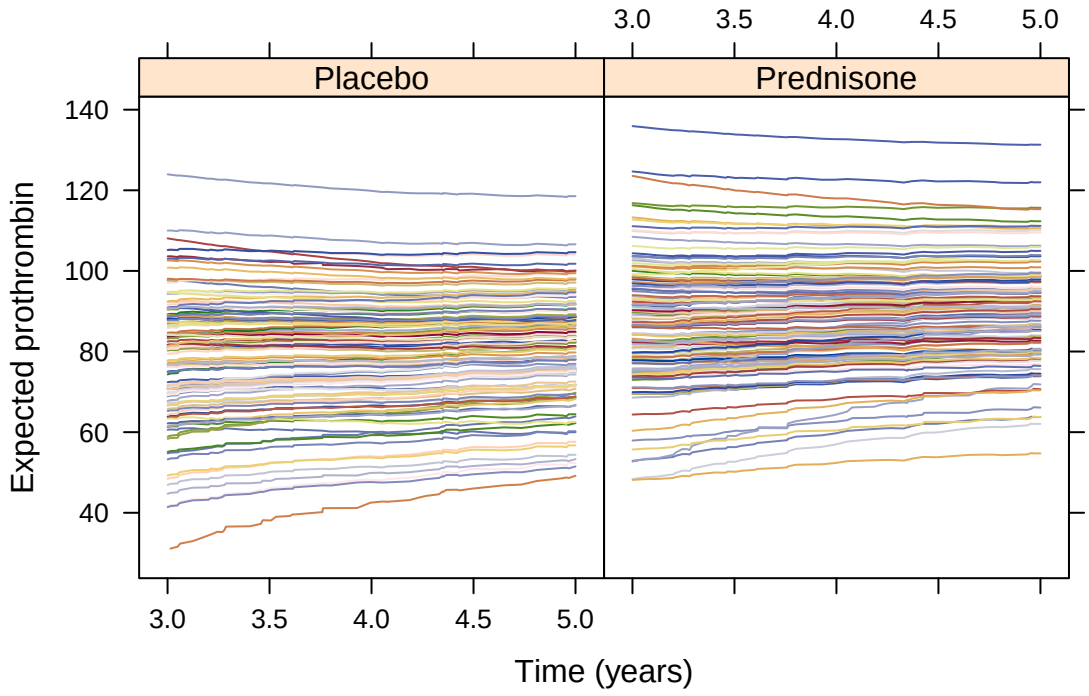
```
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
data1LM <- tmp$data1LM
patLM <- sort(unique(data1LM$patid))
nLM <- nrow(data1LM)

predrevCV <- predXhatrevCV <- rep(NA, nLM)
dataLMrev_all <- NULL

revprobsall <- list()

# pb <- progress_bar$new(total = nLM) # progress bar
for (i in 1:nLM)
{
  # pb$tick() # show progress
  # deb(i, method="cat")
  pati <- patLM[i]
```

```
#
# Fit time-dependent Cox model on data with subject i excluded
#
cls1mini <- subset(cls1, patid != pati)
cls2mini <- subset(cls2, patid != pati)

tmp <- makeLMdata(data1=cls1mini, data2=cls2mini, id="patid",
                  time1="survyrs", time2="measyrs", LM=LM)
data1LMmini <- tmp$data1LM
data2LMbefmini <- tmp$data2LMbef
data2mini <- cls2mini

ttLMmini <- sort(unique(data1LMmini$survyrs[data1LMmini$status==1 &
                                            data1LMmini$survyrs <= LM+width]))

# Refit revival model
estimates <- fitrevival(data1=cls1mini, data2=cls2mini, tau=tau)

# Overall Kaplan-Meier (for time points)
sfkm <- survfit(Surv(data1LM[[time1]], data1LM[[status1]]) ~ 1)
km <- summary(sfkm)
# Separate Kaplan-Meiers per treatment group
sfkmtreat <- survfit(Surv(data1LM[[time1]], data1LM[[status1]]) ~ data1LM[["treat"]])

patLMmini <- sort(unique(data1LMmini[[id]]))
nLMmini <- nrow(data1LMmini)
ttLMmini <- sort(unique(data1LMmini$survyrs[data1LMmini$status==1 &
                                            data1LMmini$survyrs <= LM+width]))

dataLMmini <- mklongdata_revival(data1LM=data1LMmini, data2LMbef=data2LMbefmini,
                                 id="patid", ttLM = ttLMmini,
                                 time1="survyrs", status1="status",
                                 time2="measyrs", marker="prothr",
                                 estimates=estimates, sfkm=sfkm, sfkmtreat=sfkmtreat,
                                 LM=LM, width=width)

ctdmini <- coxph(Surv(tstart, tstop, event) ~ Xhat, data=dataLMmini)

# Extract coefficient and baseline hazard (at mean of time-dependent covariate)
betamini <- ctdmini$coef
Xhatsmn <- ctdmini$means
ctdminibh <- basehaz(ctdmini)
ctdminibh <- data.frame(time=ctdminibh$time, H0=ctdminibh$hazard)
ctdminibh$h0 <- diff(c(0, ctdminibh$H0))
ctdminibh <- subset(ctdminibh, h0>0)

#
```

```
  # Apply time-dependent Cox model on data of subject i
  #

  cls1i <- subset(cls1, patid == pati)
  cls2i <- subset(cls2, patid == pati)
  tmpi <- makeLMdata(data1=cls1i, data2=cls2i, id=id, time1=time1, time2=time2, LM=LM)
  data1LMi <- tmpi$data1LM
  data1LMi[[time1]] <- LM + width # used for prediction, so time and status not to be used
  data1LMi[[status1]] <- 0
  data2LMbefi <- tmpi$data2LMbef

  # Catch and store the direct revival dynamic prediction probabilities
  tmp <- doXhats(LM, data1LMmini, data2LMbefmini, pati, estimates,
                 sfkm, sfkmtreat, data1i = data1LMi, data2i = data2LMbefi)
  tmptt <- c(tmp$tt, tau)
  predrevCV[i] <- 1 - sum(tmp$revprobs[tmptt<=LM+width])
  revprobsall[[i]] <- data.frame(id=tmp$pat, time=tmptt, revprobs=tmp$revprobs)

  # Make long format data containing the predictable time-dependent covariate
  dataLMi <- mklongdata_revival(data1LM=data1LMi, data2LMbef=data2LMbefi,
                                id="patid", ttLM = ttLMmini, # use same times as fit
                                time1="survyrs", status1="status",
                                time2="measyrs", marker="prothr",
                                estimates=estimates,
                                sfkm=sfkm, sfkmtreat=sfkmtreat,
                                LM=LM, width=width)
  dataLMrev_all <- rbind(dataLMrev_all, dataLMi)

  # Hazard at each time point
  # hi <- ctdminibh$h0 * exp(betamini * (dataLMi$Xhat[dataLMi$tstop != LM+width] - Xhatsmn))
  hi <- ctdminibh$h0 * exp(betamini * (dataLMi$Xhat - Xhatsmn))

  # Predicted probability
  predXhatrevCV[i] <- exp(-sum(hi, na.rm=TRUE))
}

save(revprobsall, file="revprobsall.Rdata")

# Save to image
resave(predrevCV, predXhatrevCV, file="predictionsCV.Rdata")
```

Just like landmarking 2.0 based on the Gaussian process, we have stored the individual predicted values of the time-dependent covariate, ranging from s to $s + w$. Below is a spaghetti plot of these cross-validated $\hat{X}(t|s)$ trajectories. They show similar patterns, but are more variable.

```
xyplot(Xhat ~ tstop | treat, group = patid, data = dataLMrev_all,
  xlab = "Time (years)", ylab = "Expected prothrombin", col = cols, type = "l")
```



4.5 Joint model

A final approach to consider is of course the joint model. Luckily, Dimitris Rizopoulos has published a joint model analysis of the CLS trial data (http://www.drizopoulos.com/Rpgm/JM_sampe_analysis.R). We will simply follow this analysis with a very small adaptation, using what we need only.

4.5.1 No cross-validation

```
load("cls.Rdata")
cls2 <- merge(cls2, cls1[, c("patid", "survyrs", "status")], by="patid")

# Remove measurements *at* t=0, to avoid "immediate" treatment effects
cls2 <- subset(cls2, measyrs>0)
"%w/o%" <- function(x, y) x[!x %in% y] #--  x without y
notincls2 <- pats1 %w/o% pats2 # these are the ones we loose

cls1 <- subset(cls1, !(patid %in% notincls2))
cls2 <- subset(cls2, !(patid %in% notincls2)) # not necessary
lmeFit <- lme(prothr ~ treat * measyrs, random = ~ measyrs | patid,
              data = cls2)
summary(lmeFit)
```

```
## Linear mixed-effects model fit by REML
##  Data: cls2
##        AIC     BIC    logLik
##   22213.88 22260.4 -11098.94
##
## Random effects:
```

```
##  Formula: ~measyrs | patid
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev    Corr
## (Intercept) 19.699250 (Intr)
## measyrs      3.890785 -0.087
## Residual    16.906695
##
## Fixed effects: prothr ~ treat * measyrs
##                            Value Std.Error   DF  t-value p-value
## (Intercept)             69.96525 1.5421329 2033 45.36915  0.0000
## treatPrednisone         11.77235 2.1675305  444  5.43123  0.0000
## measyrs                  1.37471 0.4698402 2033  2.92591  0.0035
## treatPrednisone:measyrs -1.51087 0.6543025 2033 -2.30913  0.0210
##  Correlation:
##                         (Intr) trtPrd mesyrs
## treatPrednisone         -0.711
## measyrs                 -0.303  0.216
## treatPrednisone:measyrs  0.218 -0.293 -0.718
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -4.915898968 -0.544916316 -0.001231867  0.534752113  3.863705916
##
## Number of Observations: 2481
## Number of Groups: 446
```

```
survFit <- coxph(Surv(survyrs, status) ~ treat + cluster(patid),
                 data = cls1, x = TRUE) # strata(treat) gives error later in survfitJM
summary(survFit)
```

```
## Call:
## coxph(formula = Surv(survyrs, status) ~ treat, data = cls1, x = TRUE,
##     cluster = patid)
##
##   n= 446, number of events= 270
##
##                    coef exp(coef) se(coef) robust se      z Pr(>|z|)
## treatPrednisone -0.1059    0.8995   0.1219    0.1219 -0.869    0.385
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## treatPrednisone    0.8995      1.112    0.7083     1.142
##
## Concordance= 0.511  (se = 0.017 )
## Likelihood ratio test= 0.76  on 1 df,   p=0.4
## Wald test            = 0.75  on 1 df,   p=0.4
## Score (logrank) test = 0.76  on 1 df,   p=0.4,   Robust = 0.76  p=0.4
##
##   (Note: the likelihood ratio and score tests assume independence of
```

```
##      observations within a cluster, the Wald and robust score tests do not).
```

```
fitJoint.pw <- jointModel(lmeFit, survFit, timeVar = "measyrs",
                          method = "piecewise-PH-aGH")
summary(fitJoint.pw)
```

```
##
## Call:
## jointModel(lmeObject = lmeFit, survObject = survFit, timeVar = "measyrs",
##     method = "piecewise-PH-aGH")
##
## Data Descriptives:
## Longitudinal Process     Event Process
## Number of Observations: 2481 Number of Events: 270 (60.5%)
## Number of Groups: 446
##
## Joint Model Summary:
## Longitudinal Process: Linear mixed-effects model
## Event Process: Relative risk model with piecewise-constant
##      baseline risk function
## Parameterization: Time-dependent
##
##    log.Lik      AIC      BIC
##  -11798.15 23630.31 23700.01
##
## Variance Components:
##               StdDev    Corr
## (Intercept)  20.1835  (Intr)
## measyrs       4.1894 -0.0518
## Residual     16.7390
##
## Coefficients:
## Longitudinal Process
##                            Value Std.Err z-value p-value
## (Intercept)              70.5386  1.5224 46.3338 <0.0001
## treatPrednisone          11.3116  2.1369  5.2936 <0.0001
## measyrs                   0.4108  0.4628  0.8878  0.3746
## treatPrednisone:measyrs -1.2511  0.6236 -2.0061  0.0448
##
## Event Process
##                    Value Std.Err  z-value p-value
## treatPrednisone   0.2291  0.1309   1.7496  0.0802
## Assoct           -0.0412  0.0037 -11.1217 <0.0001
## log(xi.1)         1.1184  0.2654   4.2145
## log(xi.2)         1.3257  0.2799   4.7354
## log(xi.3)         0.8091  0.2865   2.8238
## log(xi.4)         1.0399  0.2878   3.6129
## log(xi.5)         0.9620  0.2993   3.2137
```

```
## log(xi.6)        1.4662  0.3437   4.2666
## log(xi.7)        2.1301  0.3907   5.4528
##
## Integration:
## method: (pseudo) adaptive Gauss-Hermite
## quadrature points: 3
##
## Optimization:
## Convergence: 0
```

```
data1 <- cls1
data2 <- cls2
```

4.5.2 Cross-validation

We now obtain the predicted probabilities for the joint model, based on cross-validation.

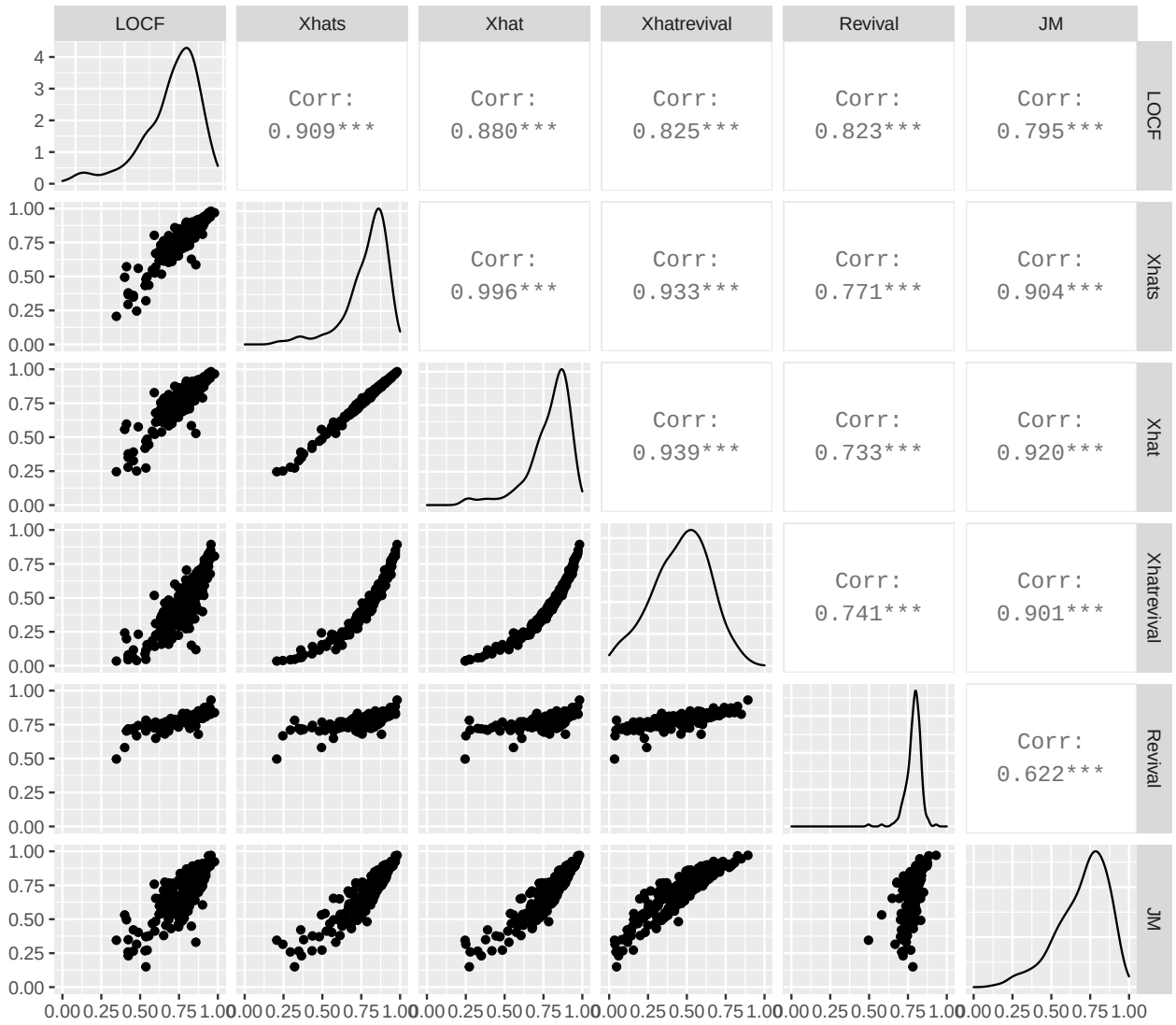
```
#
# Predictions for all patients in landmark data, cross-validated
#

# Landmark data (marker data)
data1LM <- data1[data1[[time1]] > LM, ]
patLM <- sort(unique(data1LM[[id]]))
nLM <- nrow(data1LM)

data2LM <- data2[data2[[id]] %in% patLM, ]

predJMCV <- rep(NA, nLM)
# pb <- progress_bar$new(
#   format = "(:spin) [:bar] :percent eta: :eta",
#   total = nLM) # progress bar

for (i in 1:nLM) {
#   pb$tick()
  pat <- patLM[i]

  #
  # Fit JM on data without subject i
  #
  data1mini <- subset(data1, patid != pat)
  data2mini <- subset(data2, patid != pat)

  # LME and Cox model
  lmeFit <- lme(prothr ~ treat * measyrs, random = ~ measyrs | patid,
                data = data2mini)
  survFit <- coxph(Surv(survyrs, status) ~ treat + cluster(patid),
                   data = data1mini, x = TRUE) # strata(treat) gives error later in survfitJM
```

```
  # Joint model
  fitJoint.pw <- jointModel(lmeFit, survFit, timeVar = "measyrs",
                            method = "piecewise-PH-aGH")

  data2LMi <- subset(data2LM, patid==pat)
  data2LMbefi <- data2LMi[data2LMi[[time2]] <= LM, ]
  data2LMbefi$id <- data2LMbefi[[id]]

  #
  # Obtain predicted probabilites for subject i
  #
  sfJM <- survfitJM(fitJoint.pw, newdata=data2LMbefi, survTimes=LM+width)
  predJMCV[i] <- unlist(sfJM)[2]
}

# Save to image
resave(predJMCV, file="predictionsCV.Rdata")
```

5 Comparison

To compare the three different predictions, we will first put them into one data frame and plot them against each other.

```
load("predictionsCV.Rdata")
preds <- cbind(patLM, predlocfCV, predXhatsCV, predXhatCV, predXhatrevCV,
               predrevCV, predJMCV)
preds <- as.data.frame(preds)
names(preds) <- c("patid", "LOCF", "Xhats", "Xhat", "Xhatrevival", "Revival", "JM")

pm <- ggpairs(
  preds,
  columns = c(2:7),
  diag = list(continous = "barDiag", discrete="barDiag")
)
pm2 <- pm
for (i in 2:pm$nrow) {
  pm2[i, i] <- pm[i, i] + scale_x_continuous(limits = c(0, 1))
  for (j in 1:(i-1)) {
    pm2[i,j] <- pm[i,j] +
      scale_x_continuous(limits = c(0, 1)) +
      scale_y_continuous(limits = c(0, 1))
  }
}
print(pm2)
```



In order to compare the predictive information of the different methods, I am going to transform the original predicted probabilities using the complementary log-log transformation. I am then going to enter each of the transformed cross-validated dynamic prediction probabilities in a univariate proportional hazards model in the landmark data, using administrative censoring at the horizon.

```
# For this we need the outcome data
preds <- merge(preds, data1LM[, c(id, time1, status1)], by=id, all=TRUE)
preds$statushor <- preds[[status1]]
preds$statushor[preds[[time1]] > LM+width] <- 0
preds$timehor <- preds[[time1]]
preds$timehor[preds[[time1]] > LM+width] <- LM+width

cloglog <- function(x) log(-log(x))
preds$cloglogLOCF <- cloglog(preds$LOCF)
preds$cloglogXhats <- cloglog(preds$Xhats)
preds$cloglogXhat <- cloglog(preds$Xhat)
preds$cloglogXhatrevival <- cloglog(preds$Xhatrevival)
```

```
preds$cloglogRevival <- cloglog(preds$Revival)
preds$cloglogJM <- cloglog(preds$JM)
coxph(Surv(timehor, statushor) ~ cloglogLOCF, data=preds)
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogLOCF, data = preds)
##
##                coef exp(coef) se(coef)     z        p
## cloglogLOCF 0.8577    2.3578   0.2382 3.601 0.000317
##
## Likelihood ratio test=12.99  on 1 df, p=0.0003129
## n= 229, number of events= 46
```

```
coxph(Surv(timehor, statushor) ~ cloglogXhats, data=preds)
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhats, data = preds)
##
##                 coef exp(coef) se(coef)     z        p
## cloglogXhats 0.8864    2.4263   0.1980 4.477 7.58e-06
##
## Likelihood ratio test=19.3  on 1 df, p=1.118e-05
## n= 229, number of events= 46
```

```
coxph(Surv(timehor, statushor) ~ cloglogXhat, data=preds)
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhat, data = preds)
##
##                coef exp(coef) se(coef)   z       p
## cloglogXhat 0.8739    2.3963   0.1942 4.5 6.8e-06
##
## Likelihood ratio test=19.45  on 1 df, p=1.035e-05
## n= 229, number of events= 46
```

```
coxph(Surv(timehor, statushor) ~ cloglogXhatrevival, data=preds)
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival,
##     data = preds)
##
##                       coef exp(coef) se(coef)     z        p
## cloglogXhatrevival 1.1679    3.2152   0.2558 4.566 4.98e-06
##
## Likelihood ratio test=20.3  on 1 df, p=6.632e-06
## n= 229, number of events= 46
```

```
coxph(Surv(timehor, statushor) ~ cloglogRevival, data=preds)
```

```
## Call:
```

```
## coxph(formula = Surv(timehor, statushor) ~ cloglogRevival, data = preds)
##
##                  coef exp(coef) se(coef)     z        p
## cloglogRevival  2.717    15.137    0.622 4.369 1.25e-05
##
## Likelihood ratio test=17.69  on 1 df, p=2.598e-05
## n= 229, number of events= 46
```

```
coxph(Surv(timehor, statushor) ~ cloglogJM, data=preds)
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogJM, data = preds)
##
##             coef exp(coef) se(coef)     z        p
## cloglogJM 0.9292    2.5324   0.2237 4.153 3.28e-05
##
## Likelihood ratio test=17.68  on 1 df, p=2.613e-05
## n= 229, number of events= 46
```

The most powerful univariate prediction is obtained by `Xhatrevival`. Now a series of bivariate Cox models with `Xhatrevival` and each of the other predictions.

```
# Xhatrevival selected
c1 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival, data=preds)
c2 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival + cloglogLOCF, data=preds)
c2
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival +
##     cloglogLOCF, data = preds)
##
##                       coef exp(coef) se(coef)      z       p
## cloglogXhatrevival  1.3348    3.7993   0.4719  2.829 0.00467
## cloglogLOCF        -0.1810    0.8344   0.4317 -0.419 0.67499
##
## Likelihood ratio test=20.47  on 2 df, p=3.59e-05
## n= 229, number of events= 46
```

```
anova(c1, c2) # LRT
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(timehor, statushor)
##  Model 1: ~ cloglogXhatrevival
##  Model 2: ~ cloglogXhatrevival + cloglogLOCF
##    loglik  Chisq Df P(>|Chi|)
## 1 -232.41
## 2 -232.32 0.1733  1    0.6772
```

```
c2 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival + cloglogXhats, data=preds)
c2
```

```
## Call:
```

```
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival +
##     cloglogXhats, data = preds)
##
##                       coef exp(coef) se(coef)      z     p
## cloglogXhatrevival  1.6623    5.2713   1.5833  1.050 0.294
## cloglogXhats       -0.3903    0.6769   1.2342 -0.316 0.752
##
## Likelihood ratio test=20.4  on 2 df, p=3.724e-05
## n= 229, number of events= 46
```

```
anova(c1, c2)
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(timehor, statushor)
##  Model 1: ~ cloglogXhatrevival
##  Model 2: ~ cloglogXhatrevival + cloglogXhats
##    loglik Chisq Df P(>|Chi|)
## 1 -232.41
## 2 -232.35   0.1  1    0.7519
```

```
c2 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival + cloglogXhat, data=preds)
c2
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival +
##     cloglogXhat, data = preds)
##
##                       coef exp(coef) se(coef)      z     p
## cloglogXhatrevival  2.3707   10.7048   2.2180  1.069 0.285
## cloglogXhat        -0.9253    0.3964   1.6950 -0.546 0.585
##
## Likelihood ratio test=20.6  on 2 df, p=3.371e-05
## n= 229, number of events= 46
```

```
anova(c1, c2)
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(timehor, statushor)
##  Model 1: ~ cloglogXhatrevival
##  Model 2: ~ cloglogXhatrevival + cloglogXhat
##    loglik  Chisq Df P(>|Chi|)
## 1 -232.41
## 2 -232.25 0.2991  1    0.5845
```

```
c2 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival + cloglogRevival, data=preds)
c2
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival +
##     cloglogRevival, data = preds)
##
```

```
##                       coef exp(coef) se(coef)     z      p
## cloglogXhatrevival 0.7845    2.1912   0.3816 2.056 0.0398
## cloglogRevival     1.3499    3.8570   0.9821 1.374 0.1693
##
## Likelihood ratio test=22.06  on 2 df, p=1.624e-05
## n= 229, number of events= 46
```

```
anova(c1, c2)
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(timehor, statushor)
##  Model 1: ~ cloglogXhatrevival
##  Model 2: ~ cloglogXhatrevival + cloglogRevival
##    loglik  Chisq Df P(>|Chi|)
## 1 -232.41
## 2 -231.53 1.7595  1    0.1847
```

```
c2 <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival + cloglogJM, data=preds)
c2
```

```
## Call:
## coxph(formula = Surv(timehor, statushor) ~ cloglogXhatrevival +
##     cloglogJM, data = preds)
##
##                       coef exp(coef) se(coef)     z     p
## cloglogXhatrevival 1.0373    2.8215   0.6341 1.636 0.102
## cloglogJM          0.1210    1.1286   0.5379 0.225 0.822
##
## Likelihood ratio test=20.35  on 2 df, p=3.817e-05
## n= 229, number of events= 46
```

```
anova(c1, c2)
```

```
## Analysis of Deviance Table
##  Cox model: response is  Surv(timehor, statushor)
##  Model 1: ~ cloglogXhatrevival
##  Model 2: ~ cloglogXhatrevival + cloglogJM
##    loglik  Chisq Df P(>|Chi|)
## 1 -232.41
## 2 -232.38 0.0507  1    0.8219
```

After having added `Xhatrevival`, `Revival` has the highest LRT value, but it does not reach statistical significance.

Finally, let's calculate the calibrated prediction probabilities for `Revival` and `Xhatrevival`, and for those and the others calculate the cross-validated Brier and Kullbach-Leibler scores.

```
preds$cloglogpredrevCV <- log(-log(preds$Revival))
crev <- coxph(Surv(timehor, statushor) ~ cloglogRevival, data=preds)
preds$RevivalCalibrated <- as.vector(summary(survfit(crev, newdata=preds),
                                             times=LM+width)$surv)
```

```
crev <- coxph(Surv(timehor, statushor) ~ cloglogXhatrevival, data=preds)
preds$XhatrevivalCalibrated <- as.vector(summary(survfit(crev, newdata=preds),
                                                 times=LM+width)$surv)

# Include predictions without covariates (null model) for comparison, to calculate
# percentage of prediction error reduction.
# Because of cross-validation these will not be the same; the predicted value for
# subject i will be the Kaplan-Meier estimate in the landmarking data with
# subject i excluded, say KM(-i). A quick way to calculate them is using
# pseudo-observations in the pseudo package; idea is that
# pseudo_i = n*KM - (n-1)*KM(-i), hence KM(-i) = (n*KM - pseudo_i) / (n-1)
tmp <- makeLMdata(data1=cls1, data2=cls2, id="patid",
                  time1="survyrs", time2="measyrs", LM = LM)
data1LM <- tmp$data1LM
nLM <- nrow(data1LM)
KMw <- summary(survfit(Surv(survyrs, status) ~ 1, data=data1LM), times=LM+width)$surv
pseudovals <- as.numeric(pseudosurv(data1LM$survyrs, data1LM$status, LM+width)$pseudo)
preds$null <- (nLM*KMw - pseudovals) / (nLM-1)

eps <- 1e-06
notusable <- which(data1LM[[time1]] < LM + width & data1LM[[status1]]==0)
usable <- which(!(data1LM[[time1]] < LM + width & data1LM[[status1]]==0))

sfcens <- survfit(Surv(data1LM[[time1]], data1LM[[status1]]==0) ~ 1)

Brier <- function(y, pred) return((y-pred)^2)
KL <- function(y, pred) return( -(y*log(pred) + (1-y)*log(1-pred)) )

# Calculation of Brier and Kullback-Leibler scores
evaltimes <- pmin(preds[[time1]] - eps, LM + width) # data have to be ordered
preds <- preds[order(evaltimes), ]
y <- as.numeric(preds$statushor == 1)
usable <- as.numeric(!(preds[[time1]] < LM + width & preds$statushor==0))
IPCW <- 1 / (summary(sfcens, times=evaltimes)$surv)
PercRedPredErr <- function(Score1, Score0) (Score0 - Score1) / Score0

tbl <- matrix(0, 7, 4)
# Brier scores
Brier0 <- mean( Brier(y, 1-preds$null) * IPCW * usable )
tbl[1, 1] <-  round(Brier0, 4)
tmp <- mean( Brier(y, 1-preds$JM) * IPCW * usable )
tbl[2, 1] <- round(tmp, 4)
tbl[2, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
tmp <- mean( Brier(y, 1-preds$RevivalCalibrated) * IPCW * usable )
tbl[3, 1] <- round(tmp, 4)
tbl[3, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
tmp <- mean( Brier(y, 1-preds$LOCF) * IPCW * usable )
```

```
tbl[4, 1] <- round(tmp, 4)
tbl[4, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
tmp <- mean( Brier(y, 1-preds$Xhats) * IPCW * usable )
tbl[5, 1] <- round(tmp, 4)
tbl[5, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
tmp <- mean( Brier(y, 1-preds$Xhat) * IPCW * usable )
tbl[6, 1] <- round(tmp, 4)
tbl[6, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
tmp <- mean( Brier(y, 1-preds$XhatrevivalCalibrated) * IPCW * usable )
tbl[7, 1] <- round(tmp, 4)
tbl[7, 2] <- round(100 * PercRedPredErr(tmp, Brier0), 1)
#
# Kullback-Leibler scores
#
KL0 <- mean( KL(y, 1-preds$null) * IPCW * usable )
tbl[1, 3] <-  round(KL0, 4)
tmp <- mean( KL(y, 1-preds$JM) * IPCW * usable )
tbl[2, 3] <- round(tmp, 4)
tbl[2, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)
tmp <- mean( KL(y, 1-preds$RevivalCalibrated) * IPCW * usable )
tbl[3, 3] <- round(tmp, 4)
tbl[3, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)
tmp <- mean( KL(y, 1-preds$LOCF) * IPCW * usable )
tbl[4, 3] <- round(tmp, 4)
tbl[4, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)
tmp <- mean( KL(y, 1-preds$Xhats) * IPCW * usable )
tbl[5, 3] <- round(tmp, 4)
tbl[5, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)
tmp <- mean( KL(y, 1-preds$Xhat) * IPCW * usable )
tbl[6, 3] <- round(tmp, 4)
tbl[6, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)
tmp <- mean( KL(y, 1-preds$XhatrevivalCalibrated) * IPCW * usable )
tbl[7, 3] <- round(tmp, 4)
tbl[7, 4] <- round(100 * PercRedPredErr(tmp, KL0), 1)

tbl <- as.data.frame(tbl)
row.names(tbl) <- c("Null", "JM", "Revival", "LOCF", "Xhats", "Xhat", "XhatRevival")
names(tbl) <- c("Brier", "Brier_PercRedPredErr", "KL", "KL_PercRedPredErr")
tbl
```

```
##              Brier Brier_PercRedPredErr     KL KL_PercRedPredErr
## Null        0.1683                 0.0 0.5206               0.0
## JM          0.1647                 2.2 0.5040               3.2
## Revival     0.1565                 7.0 0.4858               6.7
## LOCF        0.1585                 5.8 0.4932               5.3
## Xhats       0.1549                 8.0 0.4797               7.9
## Xhat        0.1549                 8.0 0.4791               8.0
```

```
## XhatRevival 0.1536                8.7 0.4751                8.7
```