SurvGPR Vignette

Aaron J. Molstad

5/28/2019

In this document, we show how to use the SurvGPR R package. First, we download the package from github.com/ajmolstad/SurvGRP.

```
library(devtools)
devtools::install_github("ajmolstad/SurvGPR")
```

Then, we create survival data from the Gaussian process regression model.

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.5.2  magrittr_1.5    tools_3.5.2     htmltools_0.3.6
##  [5] yaml_2.2.0      Rcpp_1.0.1      stringi_1.3.1   rmarkdown_1.11
##  [9] knitr_1.21      stringr_1.4.0   xfun_0.5        digest_0.6.18
## [13] evaluate_0.13
```

```
library(SurvGPR)
library(MASS)
# set dimesions
set.seed(20)
n = 500
p = 1000
q = 2
train.inds <- sample(1:n, 300)
test.inds <- c(1:n)[-train.inds]

# generate gene expression kernel
SigmaX <- matrix(.5, nrow=p, ncol=p)
diag(SigmaX) <- 1
X <- mvrnorm(n = n, mu = rep(0, p), Sigma = SigmaX, tol = 1e-6)
Kout <- as.matrix(dist(X, method="euclidean"))
K1 <- exp(-Kout/max(Kout[train.inds,train.inds]))
K1.full <- exp(-Kout/max(Kout))

Z <- cbind(rep(1, n), matrix(rnorm(n*q), nrow=n))
```

```
beta <- c(6.2, -0.5, -1.2)
Sigma <- 3*K1.full
G <- mvrnorm(n = 1, mu = rep(0, n), Sigma = Sigma, tol = 1e-6)

# generate failure times
log_time <- Z%*%beta + c(G) + rnorm(n, sd=sqrt(.5))
C <- log(rexp(n=n, rate=1/quantile(exp(log_time), .8)))
y <- pmin(log_time, C)
time <- exp(y)
status <- 1*(y != C)
```

Next, we fit the SurvGPR model using the following. Please note that this may take a few minutes to run.

```
time.train <- time[train.inds]
status.train <- status[train.inds]
Z.train <- Z[train.inds, ]
K.train <- array(K1[train.inds, train.inds], dim = c(length(train.inds), length(train.inds),
    1))
results <- SurvGPR(time = time.train, status = status.train, Z = Z.train, K = K.train,
    tol = 1e-07, max.iter.MM = 100, max.iter = 100, quiet = FALSE, max.samples = 1e+05,
    initializer = 0)
```

```
## 1 : ASE = 0.126 ; sigma2 = 2.72 ; resid = 10.73897 ; sk = 500
## 2 : ASE = 0.031 ; sigma2 = 2.903 ; resid = 0.93703 ; sk = 500
## 3 : ASE = 0.01 ; sigma2 = 2.973 ; resid = 0.0903 ; sk = 500
## 4 : ASE = 0.006 ; sigma2 = 2.974 ; resid = 0.00503 ; sk = 500
## 5 : ASE = 0.001 ; sigma2 = 3.008 ; resid = -0.00043 ; sk = 500
## Adding samples for ascent: 1000  samples
## 5 : ASE = 0.001 ; sigma2 = 3.023 ; resid = 0.00092 ; sk = 1000
## 6 : ASE = 0.001 ; sigma2 = 3.033 ; resid = -0.00036 ; sk = 1000
## Adding samples for ascent: 2000  samples
## 6 : ASE = 0.001 ; sigma2 = 3.033 ; resid = -0.00041 ; sk = 2000
## Adding samples for ascent: 4000  samples
## 6 : ASE = 0 ; sigma2 = 3.034 ; resid = 0.00028 ; sk = 4000
## 7 : ASE = 0 ; sigma2 = 3.032 ; resid = -0.00019 ; sk = 4000
## Adding samples for ascent: 8000  samples
## 7 : ASE = 0 ; sigma2 = 3.036 ; resid = -2e-05 ; sk = 8000
## Adding samples for ascent: 16000  samples
## 7 : ASE = 0 ; sigma2 = 3.037 ; resid = 0.00014 ; sk = 16000
## 8 : ASE = 0 ; sigma2 = 3.039 ; resid = 1e-05 ; sk = 16000
## Adding samples for ascent: 32000  samples
## 8 : ASE = 0 ; sigma2 = 3.041 ; resid = 7e-05 ; sk = 32000
## Adding samples for ascent: 64000  samples
## 8 : ASE = 0 ; sigma2 = 3.042 ; resid = 0.00012 ; sk = 64000
## 9 : ASE = 0 ; sigma2 = 3.042 ; resid = 0 ; sk = 64000
## Adding samples for ascent: 128000  samples
```

We can look at the estimated variance components:

```
results$sigma2
```

```
## [1] 3.0423070 0.4677258
```

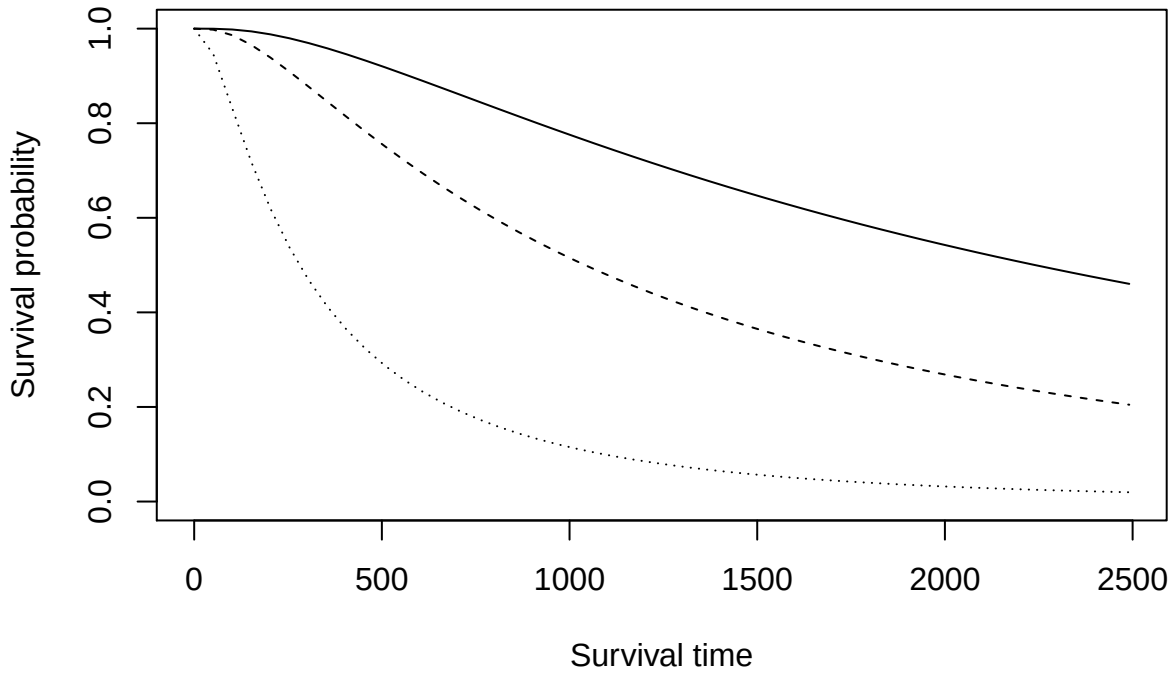
Finally, we can then predict for new subjects. Note that we can also get predicted survival probabilities at certain times (on the original scale, not log-scale) by specifying times int he `times` argument.

```
K.test <- array(K1.full, dim = c(n, n, 1))
pred_results <- SurvGPR_Predict(results = results, barT = results$Tout, Z.full = Z,
    K.full = K.test, train.inds = train.inds, test.inds = test.inds, times = seq(0,
        max(time), length = 50))
str(pred_results)
```

```
## List of 3
##  $ test.inds: int [1:200] 4 6 7 12 13 21 23 25 29 40 ...
##  $ log.pred : num [1:200, 1] 7.71 6.95 5.64 5.34 3.76 ...
##  $ survFunc : num [1:200, 1:50] 1 1 1 1 1 1 1 1 1 1 ...
```

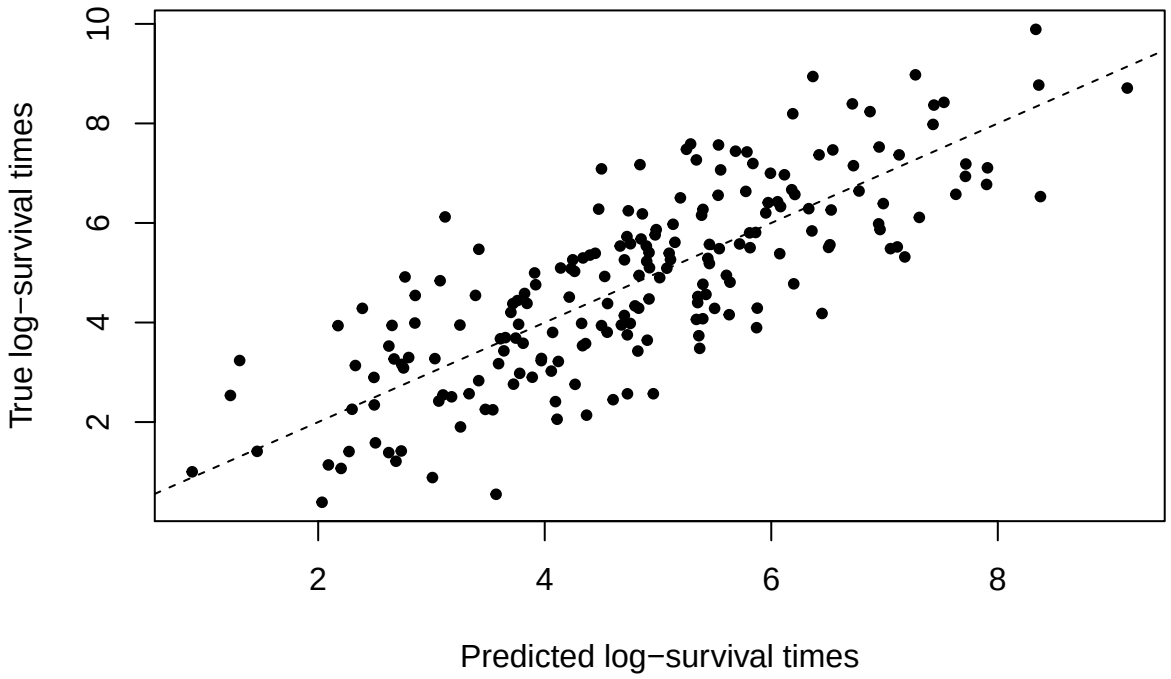
The output contains the predicted survival times on the log-scale, the testing indices, and the survival probabilities for all test set patients at the values given for `times`. We plot the estimated probabilites for three different subjects below.

```
plot(x = seq(0, max(time), length = 50), y = pred_results$survFunc[1, ], ylab = "Survival probability",
    xlab = "Survival time", type = "l", ylim = c(0, 1))
lines(x = seq(0, max(time), length = 50), y = pred_results$survFunc[2, ], lty = 2)
lines(x = seq(0, max(time), length = 50), y = pred_results$survFunc[3, ], lty = 3)
```



We also plot our predictions versus the true log-survival times.

```
plot(x = pred_results$log.pred, y = log_time[pred_results$test.inds], xlab = "Predicted log-survival ti
    ylab = "True log-survival times", pch = 20)
abline(0, 1, lty = 2)
```


True log−survival times
2
4
6
8
10
2
4
6
8
Predicted log−survival times