SurvGPR Vingette

Aaron Molstad

5/28/2019

In this document, we show briefly how to use the SurvGPR R package.

First, we download the package from github.com/ajmolstad/SurvGRP.

```
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_1.0.0      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(devtools)
devtools::install_github("ajmolstad/SurvGPR", force=TRUE)
```

```
## Downloading GitHub repo ajmolstad/SurvGPR@master
```

```
## mvtnorm      (1.0-8       -> 1.0-10     ) [CRAN]
## Rcpp         (1.0.0       -> 1.0.1      ) [CRAN]
## RcppArmad... (0.9.200.7.0 -> 0.9.400.3.0) [CRAN]
## sandwich     (2.5-0       -> 2.5-1      ) [CRAN]
## zoo          (1.8-4       -> 1.8-6      ) [CRAN]
```

```
## Installing 5 packages: mvtnorm, Rcpp, RcppArmadillo, sandwich, zoo
```

```
##
##   There is a binary version available but the source version is
##   later:
##     binary source needs_compilation
## zoo  1.8-5  1.8-6              TRUE
##
##
## The downloaded binary packages are in
##  /var/folders/tb/28vr2j9567vf2d2ywz5zb5hw0000gn/T//Rtmp3PajI6/downloaded_packages
```

```
## installing the source package 'zoo'
```

```
##
   checking for file ‘/private/var/folders/tb/28vr2j9567vf2d2ywz5zb5hw0000gn/T/Rtmp3PajI6/remotes79604f

v  checking for file ‘/private/var/folders/tb/28vr2j9567vf2d2ywz5zb5hw0000gn/T/Rtmp3PajI6/remotes79604f
##

-  preparing ‘SurvGPR’:
##

   checking DESCRIPTION meta-information ...

v  checking DESCRIPTION meta-information
##

-  checking for LF line-endings in source and make files and shell scripts
##

-  checking for empty or unneeded directories
##

-  building ‘SurvGPR_0.1.tar.gz’
##


##
```

```
library(SurvGPR)
library(MASS)
```

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

```
# set dimesions
set.seed(10)
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(5, -0.5, -1.2)
Sigma <- 3*K1.full
eo <- eigen(Sigma)
Sigmasqrt <- eo$vec%*%diag(sqrt(eo$val))%*%t(eo$vec)
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-7, max.
```

```
## 1 : ASE = 0.057 ; sigma2 = 1.164 ; resid = 3.54926 ; sk = 500
## 2 : ASE = 0.013 ; sigma2 = 1.188 ; resid = 0.2373 ; sk = 500
## 3 : ASE = 0.002 ; sigma2 = 1.251 ; resid = 0.0056 ; sk = 500
## 4 : ASE = 0.001 ; sigma2 = 1.248 ; resid = -0.00127 ; sk = 500
## Adding samples for ascent: 1000  samples
## 4 : ASE = 0.001 ; sigma2 = 1.248 ; resid = -0.00129 ; sk = 1000
## Adding samples for ascent: 2000  samples
## 4 : ASE = 0.001 ; sigma2 = 1.249 ; resid = -8e-05 ; sk = 2000
## Adding samples for ascent: 4000  samples
## 4 : ASE = 0.001 ; sigma2 = 1.249 ; resid = 0.00083 ; sk = 4000
## 5 : ASE = 0 ; sigma2 = 1.248 ; resid = -7e-05 ; sk = 4000
## Adding samples for ascent: 8000  samples
## 5 : ASE = 0 ; sigma2 = 1.249 ; resid = -7e-05 ; sk = 8000
## Adding samples for ascent: 16000  samples
## 5 : ASE = 0 ; sigma2 = 1.25 ; resid = 1e-05 ; sk = 16000
## Adding samples for ascent: 32000  samples
## 5 : ASE = 0 ; sigma2 = 1.251 ; resid = 1e-04 ; sk = 32000
## 6 : ASE = 0 ; sigma2 = 1.252 ; resid = -1e-05 ; sk = 32000
## Adding samples for ascent: 64000  samples
## 6 : ASE = 0 ; sigma2 = 1.252 ; resid = 2e-05 ; sk = 64000
## Adding samples for ascent: 128000  samples
```

We can look at the estimated variance components:

```
results$sigma2
```

```
## [1] 1.2523320 0.8485367
```

Finally, we can then predict for new subjects:

```
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, t
```