User guide to the ePCR R-package

Teemu Daniel Laajala
teelaa@utu.fi

May 29, 2018

Contents

1 Description of the ePCR package

ePCR is an R-package intended for the survival analysis of advanced prostate cancer. This document is a basic introduction to the functionality of ePCR and a general overview to the possible analysis workflows for clinical trial or hospital registry cohorts. The approach leverages ensemble-driven usage of single Cox regression based regression models named *ePCR*, which was the top performing approach in the DREAM 9.5 Prostate Cancer Challenge [2].

The latest version of ePCR is available in the Comprehensive R Archive Network (CRAN, http://cran.r-project.org/). CRAN mirrors are by default available in the installation of R, and the ePCR package is installable using the R terminal command: `install.packages("ePCR")`. This should prompt the user to select a nearby CRAN mirror, after which the installation of ePCR and its dependencies are automatically performed. After the `install.packages`-call, the ePCR package can be loaded with either command `library("ePCR")` or `require("ePCR")`.

The following notation is used in the document: R commands, package names and function names are written in `typewriter font`. The notation of format `pckgName::funcName` indicates that the function `funcName` is called from the package `pckgName`, which is prominently used in the underlying R code due to package namespaces. This document as well as other useful PDFs can be inspected using the `browseVignettes` function for any package in R.

2 Loading the example clinical cohorts into the R session

The ePCR-package is provided with two example hospital registry datasets. These datasets represent confidential hospital registry cohorts, to which kernel density estimation was fitted. Illustrative virtual patients were then generated from the kernel estimates and are provided here in the example datasets. Please see the accompanying ePCR publication for further details on the two Turku University Hospital cohorts [1], and the Synapse site for DREAM 9.5 PCC for accessing the original DREAM data. The exemplifying datasets can be loaded into an R session using:

```
> require(ePCR)
> require(survival)
> # Kernel density simulated patients from Turku University Hospital (TYKS)
> # Data consists of TEXT cohort (text-search found patients)
> # and MEDI (patients identified using medication and few keywords)
> data(TYKSSIMU)
> # The following data matrices x and survival responses y become available
> # head(xTEXTSIMU); head(yTEXTSIMU)
> # head(xMEDISIMU); head(yMEDISIMU)
```

3 Fitting new ePCR S4-objects

It is important to disginguish between the PSP and PEP objects, which represent a single penalized Cox regression model and an ensemble of Cox regression models,

respectively. PSP objects are penalized/regularized Cox regression models fitted to a particular dataset by exploring its $\{\lambda, \alpha\}$ parameter space. Notice that the sequence of λ is dependent on the $\alpha \in [0, 1]$. The regularized/penalized fitting procedure in ePCR is provided by the glmnet-package [3], although custom cross-validation and other supporting functionality is provided independently.

After fitting suitable candidate PSP-objects (Penalized Single Predictors), these will be aggregated to the ensemble structure PEP (Penalized Ensemble Predictor). The key input to PEP-constructor are the PSP intended for the use of the ensemble. We will start off by introducing the fine-tuning and fitting of *PSP*s. For this purpose the generic S4-class contructor *new* will be called with the main parameter indicating that we wish to construct a *PSP*-object.

3.1 PSP-objects

The key attributes provided for the PSP-constructor are the following parameters (see `?'PSP-class'` in R for further documentation):

- `x`: The input data matrix where rows corresponding to patients and columns to potential predictors.
- `y`: The '`surv`'-class response vector as required by Cox regression and `glmnet` in survival prediction.
- `seeds`: An integer vector or a single value for setting the random seed for cross-validation. Notice that this is highly suggested for reproducibility. If a multiple seed integers are provided, the cross-validation will be conducted separately for each. This will smoothen the cross-validation surface, but will take multiply the computational time required to fit a model.
- `score`: The scoring function utilized in evaluating the generalization ability of the fitted model in cross-validation; readily implemented scoring functions include `score.iAUC` and `score.cindex`, but custom scoring functions are also allowed. Appendix 3 elaborates on the evaluation metrics.
- `alphaseq`: Sequence of alpha values. The extreme ends $\alpha = 1$ is LASSO regression and $\alpha = 0$ is Ridge Regression. $\alpha \in]0, 1[$ is generally referred to as Elastic Net. Notice that LASSO and Ridge Regression have noticeably different characteristics as they utilizeo only the L_1 and L_2 norms, respectively; for example, a Ridge Regression model will never have its coefficients exactly zero. Furthermore, for co-linear predictors LASSO tends to pick a single one, while Ridge Regression picks multiple ones and spreads the overall effect over these predictors. Depending on the ultimate prediction purpose, one may prefer one or the other and can tailor `alphaseq` to suit their needs. By default we suggest utilizing an evenly spaced `alphaseq` over $[0, 1]$ at least for preliminary search.
- `nlambda`: Number of λ tested as a function of the corresponding α. By default *glmnet* suggested 100 values which are picked from a feasible range between model including all coefficients and converged model where no further penalization is possible.

- `folds`: Number of folds in the cross-validation (minimum 3, maximum n obs = LOO-CV).

For the sake of the example, we will construct an `ePCR` model ensemble that consists of two `PSP`-objects; one from the medication curated cohort and other from the text search cohort. We will leave out a small portion of medication and text search patients for a small test set, to later evaluate the generalization ability of the ensemble. Notice however that this is not a proper evaluation as the patients are not from an independent source, and therefore give an optimistic view to the generalization capability of the model(s).

```
> testset <- 1:30
> # Medication cohort fit
> # Leaving out patients into a separate test set using negative indices
> psp_medi <- new("PSP",
+         # Input data matrix x (example data loaded previously)
+         x = xMEDISIMU[-testset,],
+         # Response vector, 'surv'-object
+         y = yMEDISIMU[-testset,"surv"],
+         # Seeds for reproducibility
+         seeds = c(1,2),
+         # If user wishes to run the CV binning multiple times,
+         # this is possible by averaging over them for smoother CV heatmap.
+         cvrepeat = 2,
+         # Using the concordance-index as prediction accuracy in CV
+         score = score.cindex,
+         # Alpha sequence
+         alphaseq = seq(from=0, to=1, length.out=51),
+         # Using glmnet's default nlambda of 100
+         nlambda = 100,
+         # Running the nominal 10-fold cross-validation
+         folds = 10,
+         # x.expand slot is a function that would allow interaction terms
+         # For the sake of the simplicity we will consider identity function
+         x.expand = function(x) { as.matrix(x) }
+ )
```

The parameters for the second `PSP` are similar to the one above. Notice that with the PSP-members, user can tailor multiple parameters to best suit the data.

```
> # Text run similar to above
> # Leaving out patients into a separate test set using negative indices
> psp_text <- new("PSP",
+         x = xTEXTSIMU[-testset,],
+         y = yTEXTSIMU[-testset,"surv"],
+         seeds = c(3,4),
+         cvrepeat = 2,
+         score = score.cindex,
+         alphaseq = seq(from=0, to=1, length.out=51),
+         nlambda = 100,
```

```
+         folds = 10,
+         x.expand = function(x) { as.matrix(x) }
+ )

> # Taking a look on the show-method for PSP:
> psp_medi

PSP ePCR object
N observations:  120
Optimal alpha:  0.1
Optimal lambda:  0.7693556
Optimal lambda index:  14

> class(psp_medi)

[1] "PSP"
attr(,"package")
[1] "ePCR"
```

Various diagnostic and utility functions are provided for `PSP` objects. Here are some convenient example calls in R for the object fitted above, where the heatmaps for the optimized PSP surfaces are highly useful:

Noticeably, the cross-validation surface suggests different optimized penalization parameters for the two ensemble members. This most likely stems from systematic differences in the two cohorts, to which end the `ePCR` methodology offers an ensemble-driven alternative to account for differences between patient substrata.

In addition to providing the CV-grid, the identified optimal parameters are available for downstream analyses:

```
> # Identified optimal PEP parameters according to CV
> psp_medi@optimum

      Alpha  AlphaIndex      Lambda LambdaIndex
  0.1000000   6.0000000   0.7693556  14.0000000

> psp_text@optimum

      Alpha  AlphaIndex      Lambda LambdaIndex
 0.12000000  7.00000000  0.05569486 46.00000000

> # Various other object slots are available in the S4-slots indicated by @:
> slotNames(psp_medi)

 [1] "description" "features"    "strata"      "alphaseq"    "cvfolds"
 [6] "nlambda"     "cvmean"      "cvmedian"    "cvstdev"     "cvmin"
[11] "cvmax"       "score"       "cvrepeat"    "impute"      "optimum"
[16] "seed"        "x"           "x.expand"    "y"           "fit"
[21] "criterion"   "dictionary"  "regAUC"
```

```
> # Plot the CV-surface of the fitted PSP:
> plot(psp_medi,
+         # Showing only every 10th row and column name (propagated to heatcv-function)
+         by.rownames=10, by.colnames=10,
+         # Adjust main title and tilt the bias of the color key legend (see ?heatcv)
+         main="C-index CV for psp_medi", bias=0.2)
```

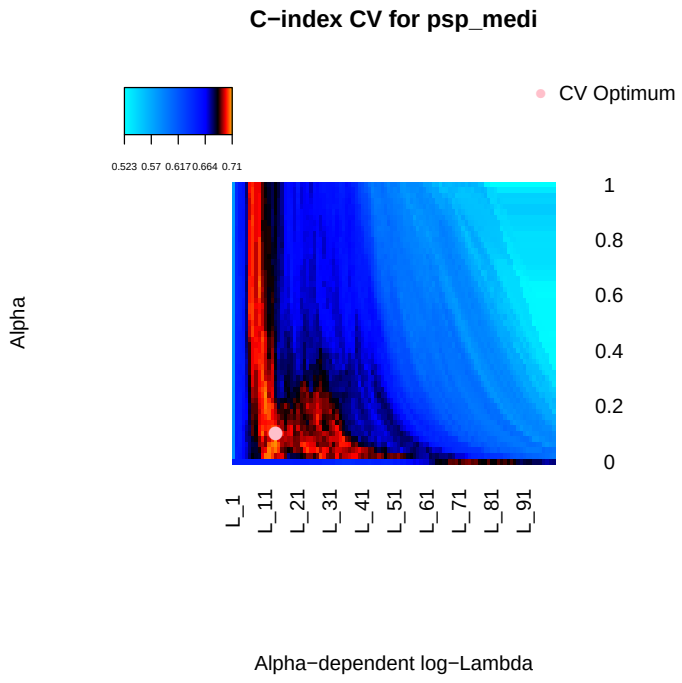

Figure 1: Cross-validation surface of the $\{\alpha, \lambda\}$ grid for the medication cohort, with concordance index as the performance metric.

```
> plot(psp_text,
+         # Showing only every 10th row and column name (propagated to heatcv-function)
+         by.rownames=10, by.colnames=10,
+         # Adjust main title and tilt the bias of the color key legend (see ?heatcv)
+         main="C-index CV for psp_text", bias=0.2)
```

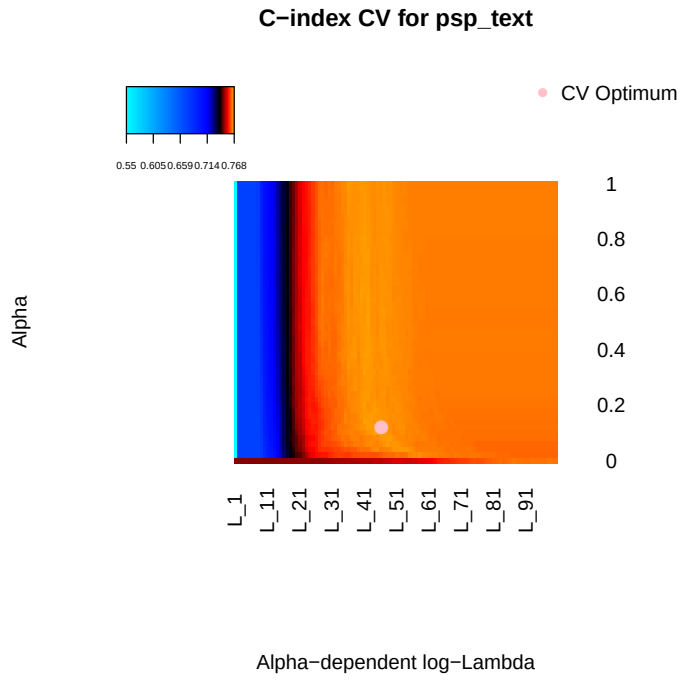

Figure 2: Cross-validation surface of the $\{\alpha, \lambda\}$ grid for the text search cohort.

3.2 PEP-objects

Once the PSP-objects have been constructed, they are aggregated to the corresponding Penalized Ensemble Predictor (PEP). The PEP objects aggregate PSP objects from various data slices or optimization criteria, and create an ensemble predictor that averages over the provided single predictors. As such, its most important input is the list of desired PSP-objects:

```
> pep_tyks <- new("PEP",
+         # The main input is the list of PSP objects
+         PSPs = list(psp_medi, psp_text)
+ )
> # These PSPs were constructed using the example code above.
> pep_tyks

Penalized Ensemble Predictor
Count of PSPs:  2

> class(pep_tyks)

[1] "PEP"
attr(,"package")
[1] "ePCR"

> slotNames(pep_tyks)

[1] "PSPs"        "description" "features"    "dictionary"  "predens"
[6] "prednorm"
```

3.3 Predictions based on PEP/PSP-objects

```
> # Conduct naive test set evaluation
> xtest <- rbind(xMEDISIMU[testset,], xTEXTSIMU[testset,])
> ytest <- rbind(yMEDISIMU[testset,], yTEXTSIMU[testset,])
> # Perform survival prediction based on the PEP-ensemble we've created
> xpred <- predict(pep_tyks, newx=as.matrix(xtest), type="ensemble")
> # Construct a survival object using the Surv-package
> ytrue <- Surv(time = ytest[,"surv"][,"time"], event = ytest[,"surv"][,"status"])
> # Test c-index between our constructed ensemble prediction and true response
> tyksscore <- score.cindex(pred = xpred, real = ytrue)
> print(paste("TYKS example c-index:", round(tyksscore, 4)))

[1] "TYKS example c-index: 0.7463"
```

4 Using the provided DREAM and TYKS ePCR-models

The ePCR R-package comes with readily fitted *ePCR*-ensembles from the work by Guinney, Wang, Laajala et al [2] as well as from hospital registry cohorts. Due to data confidentiality issues, the original data matrices or responses are

not provided in the S4-objects (although normally they would be in the slots `@x` and `@y`, respectively).

In order to gain access to the original data by Guinney et al., the processed data can be accessed as raw `.csv` files or R workspaces at:
https://www.synapse.org/#!Synapse:syn4227610/files/

Accessing the Turku University Hospital registry cohort requires a research permit and users are encouraged to contact the Center for Clinical Informatics (Arho.Virkki@tyks.fi) for further information.

Despite not providing the original data matrices, the ensemble model fits and their coefficients as a function of $\{\lambda, \alpha\}$ are fully functional. They are therefore suitable for conducting predictions for future patients or for studying effect within the estimated models/ensembles. These model objects can be loaded in `ePCR` using:

```
> data(ePCRmodels)
> class(DREAM)

[1] "PEP"
attr(,"package")
[1] ".GlobalEnv"

> class(TYKS)

[1] "PEP"
attr(,"package")
[1] "ePCR"
```

The `DREAM` S4-object is the top-performing mCRPC OS-predicting ensemble from Guinney et al. [2], while the TYKS models are fitted to the original Turku University Hospital cohorts. These model objects can be used for prediction similarly to the novel S4 PEP-object created in above sections. As an example, if we utilize the DREAM model trained on controlled clinical trials on the TYKS hospital registry patients, the OS prediction can be conducted using:

```
> # Create a DREAM-matching data input matrix from our xtest and the full data matrix
> xtemp <- conforminput(DREAM, xtest)
> # Predict survival for our hospital registry example dataset
> dreampred <- predict(DREAM,
+         # Providing full new data and average prediction over the ensemble members
+         newx=xtemp, type="ensemble",
+         # Defining that we don't want any further data matrix feature extraction
+         # The call to conforminput above already formatted the input data
+         x.expand = as.matrix
+ )
```

Notice that we utilize the helper function `conforminput` for feature extraction/creation, as multiple interaction variables were introduced in the original DREAM data matrix and the dimensions would not match in the regression task otherwise.

The following error message is quite commonly encountered when first using pre-built models to new data:

```
Error in newx %*% nbeta :
  Cholmod error 'X and/or Y have wrong dimensions'
  at file ../MatrixOps/cholmod_sdmult.c, line 90
```

It is prompted by the `glmnet`-package's C/Fortran implementation, if the β coefficients do not conform to the provided dimensions of the new data matrix X. For this purpose, the new data should have equal number of columns (variables) using data processing (functions such as `conforminput` or the S4-slot in a `PEP`-object called `x.expand`).

The concordance index for the DREAM prediction for the hospital registry patients in this example is:

```
> # Test c-index between the DREAM ensemble prediction and TYKS true response
> dreamscore <- score.cindex(pred = dreampred, real = ytrue)
> print(paste("DREAM example c-index:", round(dreamscore, 4)))

[1] "DREAM example c-index: 0.611"
```

5 Appendices

5.1 Appendix 1 - *ePCR* functionality

Table 1: This table bundles together various functions in a brief manner, and the modeling task in which they may be useful. The open-source code enables the user to use readily fitted model ensembles or construct their own. The functions presented below are a short summary of functions that the user may want to call on their own and does not include the package's internal functions. Self-sufficient examples for using the functions are provided the R-package manual, in the *Examples*-section of each function.

Functions	Names	Usage
S4-class ePCR construction		
`PEP` and `PSP`	Class initializers	`new("PSP", ...)`, `new("PEP", ...)`
S4 R default overrides		
`predict.PEP`, `print.PEP`, `show.PSP`	Override default functions for PEP-objects	Called when the default names are used on `PEP` S4-objects
`predict.PSP`, `coef.PSP`, `plot.PSP`, `print.PSP`, `show.PSP`, `print.PSP`	Override default functions for PSP-objects	Called when the default names are used on `PEP` S4-objects
Data/ensemble examples		
`TYKS`	Hospital registry cohort `x` and `y` from Turku University Hospital	`data(TYKS)`
`ePCRmodels`	Readily fitted `ePCR` S4-ensembles (DREAM and TYKS)	`data(ePCRmodels)`
Data processing		
conforminput	Try conform dimensions of input X and model β	`conforminput(psp, xinput)`
zt	z-score transformation, with NA-imputation	`apply(x, MARGIN=2, FUN=zt)`
interact.all, interact.part	Functions for feature extraction from existing data x	`interact.all(x)`
PSP-functions		
PSP.BOX, PSP.CSP, PSP.KM, PSP.NA, PSP.PCA	Diagnostics, plotting and prediction functions for PSP-objects	`PSP.funcname(psp, ...)`
Model fitting and evaluation		
cv, cv.alpha, cv.grid	Cross-validation (CV) functionality embedded into the constructors	Embedded to PSP construction
score.cindex, score.iAUC	Scoring functions for CV	Embedded to PSP construction and exemplified in this guide
Model diagnostics		
bootstrapRegCoefs	Significance of regularized model coefficients through bootstrapping	`bootstrapRegCoefs(psp, ...)`
integrateRegCurve	Integrate Area Under/Over the regularized model coefficint curves as a function of λ	`integrateRegCurve(psp, ...)`
Survival time-to-event prediction		
NelsonAalen	Heuristic Cox-Oakes extension for Nelson-Aalen for individual survival	See package documentation
TimeSurvProb	Obtain individualized time-to-event predictions where cumulative survival probability reaches 0.5	See package documentation

5.2 Appendix 2 - Comparison of the present *ePCR*-package and original DREAM code

Table 2: Description and novelty of functions, S4-classes, datasets and ePCR-models provided in the *ePCR*-package in comparison with R code published in Synapse after the DREAM 9.5 mCRPC Challenge [2].

Classes/Functions	Summary	Purpose	Novelty comparison
S4-classes			
PEP	Penalized Ensemble Predictor	S4-class ensembles consisting of PSP-members	Novel
PSP	Penalized Single Predictor	S4-class for constructing single predictors	Novel
S4 R default overrides			
predict.PEP	predict-override	Overrides default.predict for the PEP-class; gives ensemble risk prediction	Improved
print.PEP	print-override	Overrides default.print for the PEP-class; prints ensemble description and key characteristics	Novel
coef.PSP	coef-override	Overrides default.coef for the PSP-class; extracts single predictors non-zero coefficients at optimum	Novel
plot.PSP	plot-override	Overrides default.plot for the PSP-class; plots the α, λ optimization grid based on cross-validation	Novel
predict.PSP	predict-override	Overrides default.predict for the PSP-class; gives a single member's risk predictions	Novel
print.PP	print-override	Overrides default.print for the PSP-class; prints details of a single predictor's characteristics	Novel
S4 specific functions			
PSP.BOX	PSP-object diagnostics	Boxplot of a single variable in a PSP in respect to strata	Novel
PSP.CSP	PSP-object survival	Cumulative survival probabilities at $F(t)$	Novel
PSP.KM	PSP-object survival	Kaplan-Meier with division at a given cutoff point within [0,1]	Novel
PSP.NA	PSP-object survival	Individualized Nelson-Aalen with time-to-event prediction at point $t = F^{-1}(0.5)$	Novel
PSP.PCA	PSP-object diagnostics	Principal Component plot of a single PSP, showing 2 principal axes	Novel
Readily fitted S4-objects			
DREAM	PEP S4-class	The DREAM challenge top-performing ePCR model generalized in this package	Improved
TYKS	PEP S4-class	Novel hospital registry ePCR ensemble, which includes all predictors	Novel
TYKS_reduced	PEP S4-class	Novel hospital registry ePCR ensemble, reduced set of predictors	Novel
Generalized functions			
bootstrapRegCoefs	Bootstrap regularized coefficients	Evaluate significance of β_i over bootstrapped datasets with bias	Improved

conforminput	Conform input data matrix and model object dimensions	Creates missing columns to the input matrix through interactions and checks that all requires fields for β coefficients are present	Novel
cv	Generate cross-validation bins	General purpose cross-validation binning function	Improved
cv.alpha	Perform cross-validation over λ for a given α	α-specific CV run, called by the grid-function	Improved
cv.grid	Perform cross-validation over grid λ, α	Wrapped function that explores the whole parameter space	Improved
heatcv	Plot heatmaps for cross-validation grids	Function for visualizing the CV surface and local/global maxima	Improved
integrateRegCurve	Integrate area under/over regularization curve	Evaluate importance of β_i over whole range of λ	Improved
interact.all	Interactions between all columns in the data	Extract novel variables from the provided data x	Re-used from DREAM
interact.part	Interactions between selected columns in the data	Extract novel variables from the provided data x	Re-used from DREAM
meanrank	Compute mean rank over PEP-members	Aggregated ensemble prediction over the members	Re-used from DREAM
NelsonAalen	Cox-Oakes extension of the Nelson-Aalen estimates for a Cox model	Estimating individual survival curves using a heuristic method instead of population survival	Novel
normriskrank	Normalized risk ranks over PEP-members	Normalized risk ranks to range [0,1] after averaging over ensemble members	Improved
score.cindex	Concordance-index prediction performance	Scoring function for cross-validation or otherwise inspecting prediction performance	Re-used from DREAM
score.iAUC	Time-integrated AUC prediction performance	Scoring function for cross-validation or otherwise inspecting prediction performance	Re-used from DREAM
TimeSurvProb	Predict survival based on cumulative survival probabilities	Predict survival time using `NelsonAalen` by identifying first time point where $F^{-1}(t) = 0.5$	Novel
zt	Extended *scale*-function	Performs z-score transformation $\frac{x-\mu_x}{\sigma_x}$ with additional functionality such as NA-imputation	Re-used from DREAM
Attached datasets			
TYKSSIMU	Representative hospital registry data from TYKS	usage: *data(TYKSSIMU)*	Novel
xMEDISIMU	x data matrix of medication cohort	loaded with *TYKSSIMU*	Novel
xTEXTSIMU	x data matrix of text-search cohort	loaded with *TYKSSIMU*	Novel
yMEDISIMU	y responses of medication cohort	loaded with *TYKSSIMU*	Novel
yTEXTSIMU	y responses of text-search cohort	loaded with *TYKSSIMU*	Novel

5.3 Appendix 3 - Abbreviations and terminology

- *c-index*: Concordance index - scoring metric computed over all possible pairs of observations in survival prediction. It is the fraction of pairs in the data, where the observation with the higher survival time has the higher probability of survival predicted by the model. For further details, see `timROC` R-package and its function `timeROC` documentation.

- *iAUC*: Integrated Area Under Curve (over follow-up time) - scoring metric computed within a given time-interval of AUCs at discrete time points in survival prediction. The integrated area over the defined time interval is sensitive to the study follow-up times and therefore is sensitive to the choice of interval. Counter-intuitively, completely randomized iAUC may therefore be therefore be different from 0.5 in contrast to normal AUC. For further details, see `timeROC` R-package and its `timeROC` together with the R-package `Bolstad2` and its `sintegral` function documentation.

- *OS*: Overall Survival - Clinical follow-up time for patients until death, last observed date of being alive (lost in follow-up) or pre-determined end of the study. OS is typically computed starting from the set date of initiating an intervention in clinical trials.

*: equal contribution

References

[1] Laajala TD*, Murtojärvi M*, Virkki A, Aittokallio T. *ePCR*: an R-package for survival and time-to-event prediction in advanced prostate cancer, applied to real-world patient cohorts. *Manuscript submitted.*

[2] Guinney J*, Wang T*, Laajala TD*, et al. Prediction of overall survival for patients with metastatic castration-resistant prostate cancer: development of a prognostic model through a crowdsourced challenge with open clinical trial data. Lancet Oncol. 2017 Jan;18(1):132-142. doi: 10.1016/S1470-2045(16)30560-5.

[3] Simon N, Friedman J, Hastie T, Tibshirani R. Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. J Stat Softw. 2011 Mar; 39(5): 1–13. doi: 10.18637/jss.v039.i05